



Arm[®] C1-Premium Core

Telemetry Specification

Non-Confidential

Copyright © 2024–2025 Arm Limited (or its affiliates).
All rights reserved.

Issue 02

109950_0200_02_en



Arm® C1-Premium Core Telemetry Specification

This document is Non-Confidential.

Copyright © 2024–2025 Arm Limited (or its affiliates). All rights reserved.

This document is protected by copyright and other intellectual property rights.

Arm only permits use of this document if you have reviewed and accepted [Arm's Proprietary Notice](#) found at the end of this document.

This document (109950_0200_02_en) was issued on 2025-09-10. There might be a later issue at <https://developer.arm.com/documentation/109950>

See also: [Proprietary notice](#) | [Product and document information](#) | [Useful resources](#)

Start reading

If you prefer, you can skip to [the start of the content](#).

Intended audience

This specification is useful for engineers to collect and analyze Arm® C1-Premium core telemetry data to gain insights about a system's performance. Architects and system designers can also use it for resource characterization and platform tuning.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on <https://support.developer.arm.com>.

To provide feedback on the document, fill the following survey: <https://developer.arm.com/documentation-feedback-survey>.

Contents

1. Overview of the C1-Premium Telemetry methodology.....	6
1.1 Documentation and resources.....	7
2. C1-Premium microarchitecture and its telemetry features.....	8
3. CPU performance analysis methodology.....	11
3.1 Stage 1: Topdown analysis.....	13
3.1.1 Topdown Level 1 metric group.....	15
3.1.2 Topdown Frontend metric group.....	17
3.1.3 Topdown Backend metric group.....	19
3.1.4 Topdown SME2 metric group.....	22
3.2 Stage 2: Microarchitecture exploration.....	23
3.3 Core memory operations.....	31
4. C1-Premium Telemetry cheat-sheets and lookup tables.....	32
4.1 Metrics cheat sheet for C1-Premium.....	32
4.2 PMU events cheat sheet for C1-Premium.....	35
4.3 Metrics lookup table for C1-Premium.....	39
4.4 PMU events lookup table for C1-Premium.....	47
5. Metrics by metric group in C1-Premium.....	69
5.1 Topdown_Backend metrics for C1-Premium.....	70
5.2 Topdown_CME metrics for C1-Premium.....	78
5.3 Cycle_Accounting metrics for C1-Premium.....	80
5.4 Operation_Mix metrics for C1-Premium.....	83
5.5 Topdown_L1 metrics for C1-Premium.....	89
5.6 Topdown_Frontend metrics for C1-Premium.....	92
5.7 General metrics for C1-Premium.....	97
5.8 MPKI metrics for C1-Premium.....	98
5.9 Miss_Ratio metrics for C1-Premium.....	105
5.10 SVE_Effectiveness metrics for C1-Premium.....	112
5.11 FP_Arithmetic_Intensity metrics for C1-Premium.....	114
5.12 FP_Precision_Mix metrics for C1-Premium.....	116

5.13 Branch_Effectiveness metrics for C1-Premium.....	117
5.14 ITLB_Effectiveness metrics for C1-Premium.....	120
5.15 DTLB_Effectiveness metrics for C1-Premium.....	126
5.16 L1I_Cache_Effectiveness metrics for C1-Premium.....	133
5.17 L1D_Cache_Effectiveness metrics for C1-Premium.....	134
5.18 L2D_Cache_Effectiveness metrics for C1-Premium.....	136
5.19 L2I_Cache_Effectiveness metrics for C1-Premium.....	138
5.20 L3_Cache_Effectiveness metrics for C1-Premium.....	139
5.21 LL_Cache_Effectiveness metrics for C1-Premium.....	141
5.22 Rename_Effectiveness metrics for C1-Premium.....	143
5.23 IQ_Effectiveness metrics for C1-Premium.....	145
5.24 MCQ_Effectiveness metrics for C1-Premium.....	147
5.25 Port_Utilization metrics for C1-Premium.....	148
5.26 Prefetcher_Effectiveness metrics for C1-Premium.....	151
5.27 Atomics_Effectiveness metrics for C1-Premium.....	156
5.28 Average_Latency metrics for C1-Premium.....	159
5.29 Bus_Effectiveness metrics for C1-Premium.....	162
5.30 System_Memory_Effectiveness metrics for C1-Premium.....	164
6. PMU events by functional group in C1-Premium.....	167
6.1 Bus (BUS) events for C1-Premium.....	168
6.2 Chain (CHAIN) events for C1-Premium.....	171
6.3 Exception (EXCEPTION) events for C1-Premium.....	171
6.4 L1D_Cache (L1D CACHE) events for C1-Premium.....	176
6.5 L1I_Cache (L1I CACHE) events for C1-Premium.....	184
6.6 L2_Cache (L2 CACHE) events for C1-Premium.....	190
6.7 L3_Cache (L3 CACHE) events for C1-Premium.....	202
6.8 LL_Cache (LL CACHE) events for C1-Premium.....	205
6.9 Memory (MEMORY) events for C1-Premium.....	207
6.10 Retired (RETIRED) events for C1-Premium.....	212
6.11 SPE (SPE) events for C1-Premium.....	225
6.12 Spec_Operation (SPEC OPERATION) events for C1-Premium.....	228
6.13 FP_Operation (FP OPERATION) events for C1-Premium.....	245
6.14 Stall (STALL) events for C1-Premium.....	247
6.15 General (GENERAL) events for C1-Premium.....	267
6.16 TLB (TLB) events for C1-Premium.....	274

6.17 SVE (SVE) events for C1-Premium.....291

6.18 Non_PMU (NON_PMU) events for C1-Premium..... 297

6.19 TRACE (TRACE) events for C1-Premium..... 298

6.20 CPU_Debug (CPU_DEBUG) events for C1-Premium..... 301

6.21 Coherency (COHERENCY) events for C1-Premium.....303

7. Supplemental performance debug PMU events..... 306

Proprietary notice.....307

Product and document information..... 309

Product status.....309

Revision history..... 309

Conventions.....310

Useful resources..... 313

1. Overview of the C1-Premium Telemetry methodology

The Arm® C1-Premium Telemetry Specification describes the Topdown methodology, derived metrics, and Performance Monitoring Unit (PMU) events supported by the Arm C1-Premium unit.

The Arm® C1-Premium unit is also known as the processor or co-processor if an Arm® C1-SME2 unit is included in the configuration.



This specification is applicable to all releases of the product. The C1-SME2 unit (previously known as the CME unit) is also referred to as the SME2 unit throughout this specification. However, some instances of the term CME remain in this specification.

This specification implements the framework provided by the [Arm® CPU Telemetry Solution Topdown Methodology Specification](#), which is referred to as the Architecture Specification. The reader is expected to read this document in conjunction with the Architecture Specification.

Arm Telemetry framework

This specification outlines the telemetry features implemented for the Arm C1-Premium and follows the Arm Telemetry framework for CPUs defined in the Architecture Specification.

The following list provides a brief description of the Telemetry framework:

Events

Hardware performance monitoring events implemented by the product that contain raw data read from the registers or memory buffers.

Metrics

Derived mathematical relationships between events that provide insight into the system behavior. They are developed to abstract hardware details of the events from consumers of the telemetry data.

Metric groups

Group of metrics that can be analyzed together to investigate a bottleneck scenario or a specific resource in a given system.

Methodology

Actionable guidance, such as Arm Topdown methodology, to explain how to consume the different metrics and events for a specific usage model. Decision tree with a group of metrics that can be analyzed hierarchically to investigate a bottleneck scenario or a specific resource in a given system.

Tool support for profiling and monitoring

This specification is also available in a machine-readable format (JSON) to be consumed by profiling and monitoring tools. The JSON schema implements the Arm Telemetry framework from the Architecture Specification.

1.1 Documentation and resources

Arm products include a set of documents.

The documentation and resources for C1-Premium consist of:

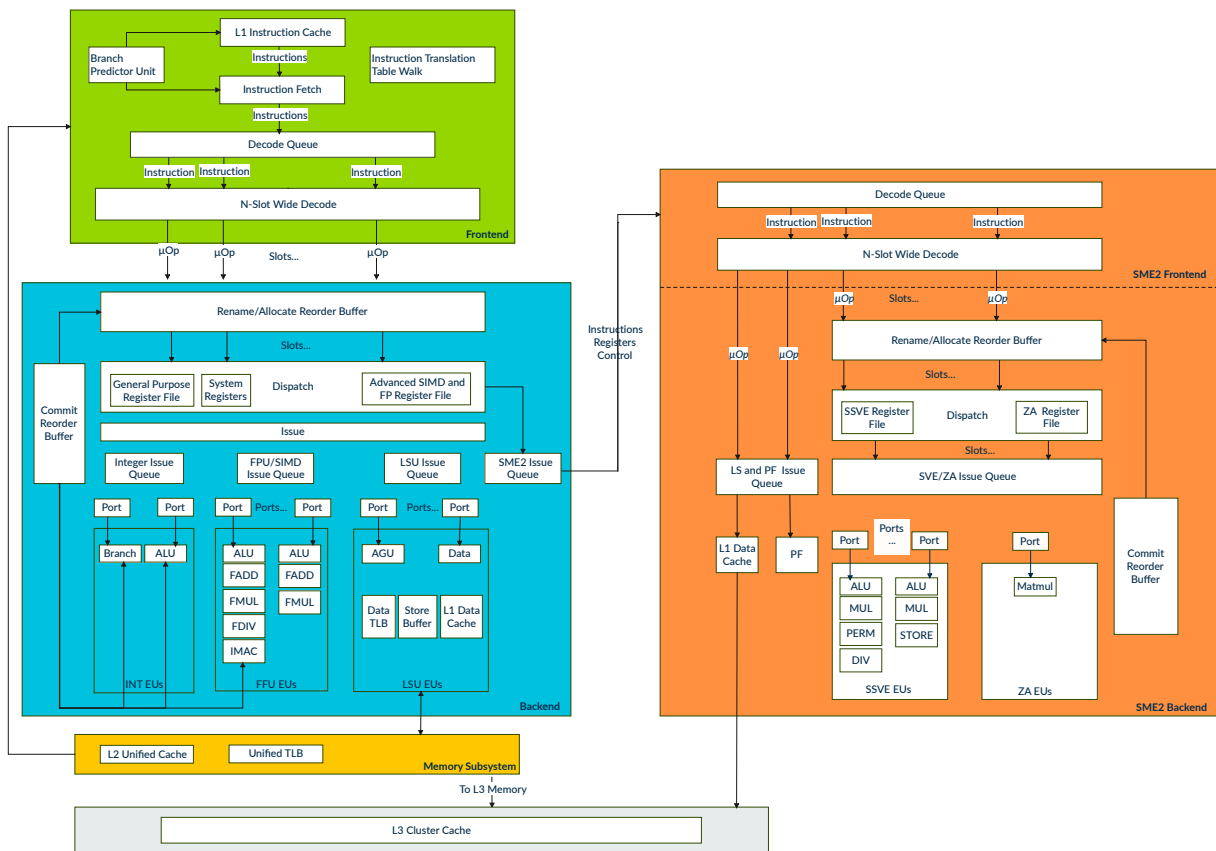
- [Arm® Telemetry on Arm Developer](#)
- [Arm® C1-Premium Core Technical Reference Manual](#)
- [Arm® C1-Scalable Matrix Extension 2 Telemetry Specification](#)

2. C1-Premium microarchitecture and its telemetry features

C1-Premium is a super pipelined superscalar processor that has an in-order frontend and out-of-order backend.

The following figure shows the microarchitecture details of C1-Premium.

Figure 2-1: C1-Premium core microarchitecture



The frontend of the core pipeline is comprised of the instruction fetch and decode units. The frontend also includes a branch predictor unit that predicts branch target addresses and fetches instructions ahead of the pipeline. This unit helps to hide latencies caused by control flow bubbles in the pipeline.

The fetch unit can fetch multiple instructions for each cycle, which gets stored in a decode queue. The decode queue sends multiple instructions per cycle for decoding, whose bandwidth is determined by the number of available decode slots. The decode unit decomposes the Arm architecture instructions into micro-operations, also known as micro-ops or μ ops. This unit

decodes more than one micro-operation for each cycle. These micro-ops are then fed to the rename unit for organization of out-of-order execution in the backend of the core pipeline. The instruction bandwidth is determined by the number of renamed slots available in the microarchitecture. The number of slots can be discovered in the PMMIR_EL1 registers SLOTS field. From a microarchitecture standpoint, the rename unit is considered the boundary between the frontend and backend of the core pipeline.

The backend of the core has a scheduler that orchestrates the operations to be executed when the issue queue associated with the operation can accept the operation. The issue queue sends operations for execution when the execution unit is free and the source operands are ready. Once the execution is complete, the results are sent to the commit Reorder Buffer (ROB) from where the instructions are retired when the speculated execution is confirmed. The backend of the core executes the operations out-of-order and stores results with the help of the reorder buffer. The dispatch unit tracks dependencies between operations and determines the operand availability for the execution of operations. Register renaming occurs at this stage to mitigate data dependency hazards.

In the dispatch unit, issue queues are employed for:

- Queuing the micro-operations (μ ops) to assigned ports
- Managing dependencies between operations
- Tracking operand availability for execution

Each execution port supports different categories of operations. After the execution of operations, the ROB is updated with execution results status. Completed operations are retired architecturally in program order. Operations are flushed when the predicted program flow changes due to mispredictions or exceptions. The typical operation types supported by different execution units (EU) are:

- Integer EU, executes branches and arithmetic instructions
- Floating-Point Unit (FPU) and Single Instruction Multiple Data (SIMD) EU, executes the floating-point instruction and vector instructions
- Load Store Unit (LSU), executes load, store, and atomic instructions
- Scalable Matrix Extension 2 (SME2), executes scalable matrix (SME) instructions

The SME2 co-processor is a shared unit that implements SME and SME2 instructions. It is implemented inside an Arm® C1-DSU (DSU) cluster. The SME2 co-processor is shared between all the cores and is accessible through this DSU cluster. The DSU behaves as a full interconnect with shared L3 support and full coherency between the cores and the SME2 co-processor. CPU performance for SME2 workloads may be impacted by CPU managed resources or by SME2 managed resources. This specification describes the methodology to analyze CPU performance that is related to SME2 unit bottlenecks due to CPU resources. For more information about the performance analysis methodology for the resources that are managed by the SME2 co-processor, see [Arm® C1-Scalable Matrix Extension 2 Telemetry Specification](#).

The memory subsystem of the core handles the execution of load and store operations which rely heavily on the memory hierarchy levels.

C1-Premium has dedicated, unique L1 instruction and data caches for each core in the complex. The L2 cache is a unified cache that is shared between cores in a complex and used for both instruction and data. The caches are set associative and the sizes are configurable for each implementation. The cache line size for this core is 64 bytes. The L2 cache of a complex connects to the DynamIQ™ Shared Unit (DSU) cluster logic and an optional L3 cluster cache.

C1-Premium system configurations

C1-Premium connects to the compute cluster through the DSU which supports a shared L3 cache across all cores in the cluster. The DSU also supports a snoop filter for coherency management of shared data between the processors. It is possible for the platform to support other caches downstream in the system interconnects.

It is always best to check with the Silicon Provider for details on the system configuration for the underlying system, including the cache sizes.

PMU capabilities of C1-Premium

The C1-Premium implements version 3.8 of the Performance Monitors Extension, FEAT_PMUv3p8, and Arm version 8.8 debug architecture, FEAT_Debugv8p8.

For more information, see [Arm® Architecture Reference Manual for A-profile architecture](#).

The C1-Premium PMU can support 6 or 31 programmable performance counters, depending on your configuration, and one fixed function counter to count CPU cycles.

3. CPU performance analysis methodology

The Arm Topdown methodology for the C1-Premium enables you to use PMU events, metrics, and metric groups to identify potential bottlenecks that could negatively impact the performance of the core in your design.

The methodology is conducted in two stages.

Stage 1: Topdown analysis

The first stage is to perform Topdown analysis to detect and identify any performance bottlenecks in the CPU, and provide direction for further analysis at Stage 2. Stage 1 uses hierarchical pipeline stall-related metrics. For more information, see [Stage 1: Topdown analysis](#).

Stage 2: Microarchitecture Exploration

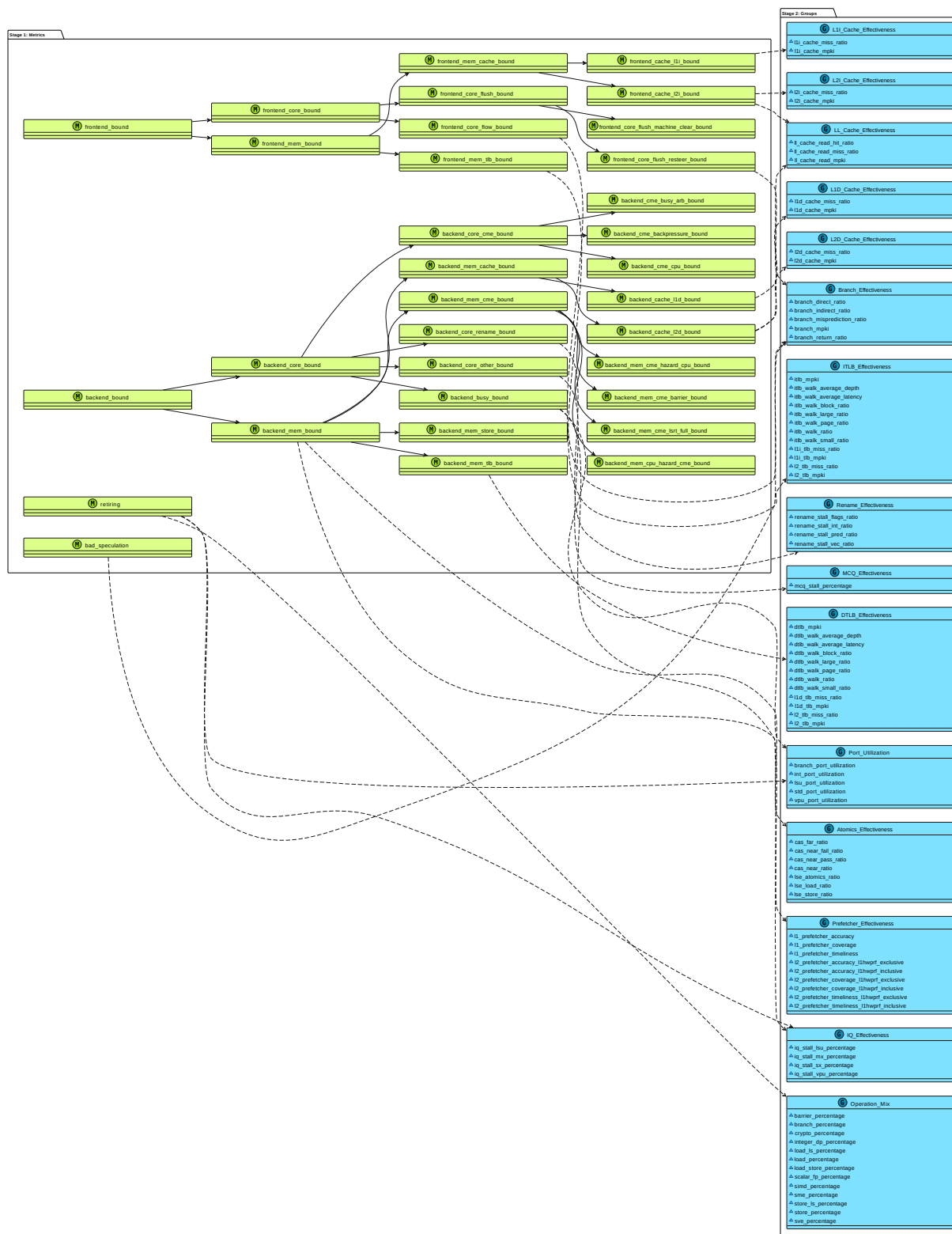
The second stage is to conduct microarchitecture exploration to further analyze bottlenecked CPU resources, based on the Stage 1 findings. Stage 2 uses a set of CPU resource-effectiveness metrics. For more information, see [Stage 2: Microarchitecture Exploration](#).

For more information about our approach to performance analysis and the standardized telemetry framework, see [Arm® CPU Telemetry Solution Topdown Methodology Specification](#).

We recommend collecting all metrics that are in Stage 1 and Stage 2 Topdown analysis for workload characterization. We provide a recommended set of microarchitecture exploration metric groups for further analysis, for hotspots detected in Stage 1. All Stage 2 metrics can be used to derive further insights into the overall microarchitecture behavior during the execution of the application under investigation. These metrics can be used independently of Stage 1.

The following figure gives an overview of the Topdown methodology tree for C1-Premium. It shows Stage 1 metrics, and Stage 2 metric groups and metrics. Stage 1 covers the stall-related metrics for Topdown analysis for bottleneck identification. Stage 2 covers the microarchitecture exploration metric groups for root cause analysis.

Figure 3-1: Topdown methodology overview for C1-Premium





The industry-standard metrics Misses Per Kilo Instructions (MPKI) and Miss Ratios are metric groups that are defined in Stage 2, but they are not included in the hierarchical pipeline stall analysis of Stage 1, as these are a standard set of metrics recommended for analysis and characterization rather than hierarchical analysis specified as topdown.

3.1 Stage 1: Topdown analysis

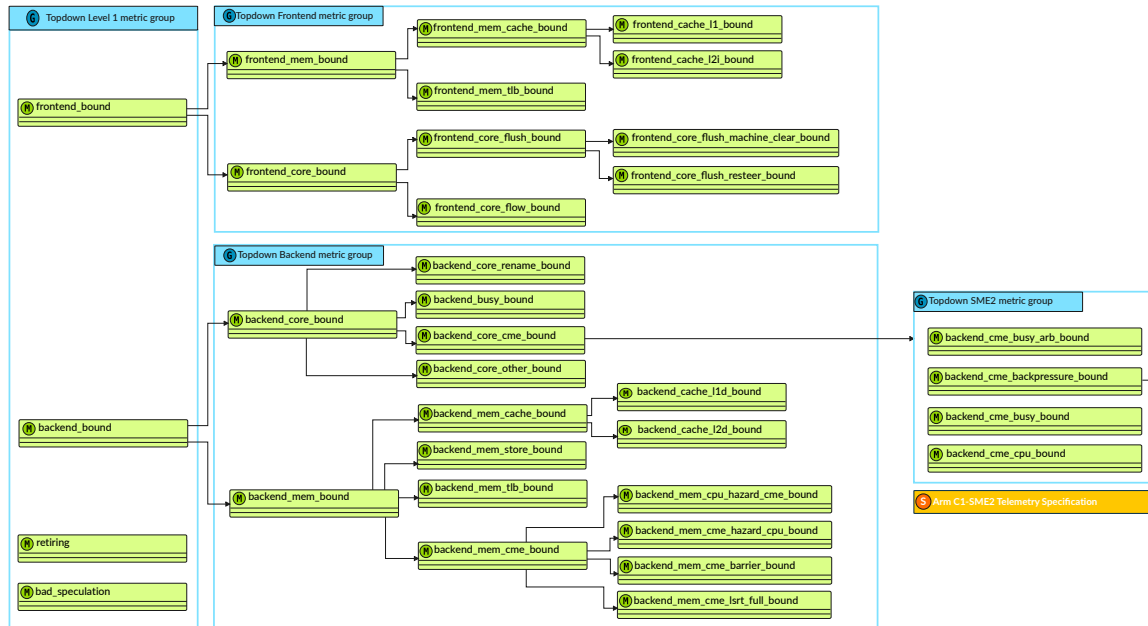
The objective of the Topdown analysis is to identify potential bottlenecks in the key blocks of the processor.

Each block can be further broken down to units and subunits within the block that all have different performance characteristics. For example, control flow and data flow issues can be fixed differently in software after root cause analysis.

Results from a Topdown analysis can indicate:

- Which metric groups and metrics to analyze next.
- Areas where software improvements can be made in the current design.
- Existing microarchitectural limitations that can inform future hardware improvements, better system configuration, or tuning decisions at a platform level.

The following figure shows the metric groups and metrics in the Stage 1 Topdown methodology tree for C1-Premium, which supports up to four levels of hierarchical pipeline stall accounting.

Figure 3-2: C1-Premium Topdown methodology Stage 1 overview

C1-Premium has four Stage 1 metric groups.

Metric Group: Topdown Level 1

The [Topdown Level 1 metric group](#) contains the first set of metrics to begin Topdown analysis of application performance. These metrics provide the percentage distribution of processor pipeline utilization.

For more information about the metrics in this group and the associated formulas and events, see [Topdown_L1](#).

Metric Group: Topdown Frontend

The [Topdown Frontend metric group](#) contains a set of metrics to analyze a frontend bound workload.

For more information about the metrics in this group and the associated formulas and events, see [Topdown_Frontend](#).

Metric Group: Topdown Backend

The [Topdown Backend metric group](#) contains a set of metrics to analyze a backend bound workload.

For more information about the metrics in this group and the associated formulas and events, see [Topdown_Backend](#).

Metric Group: Topdown SME2

The [Topdown SME2 metric group](#) contains a set of metrics to analyze a SME2 bound workload.

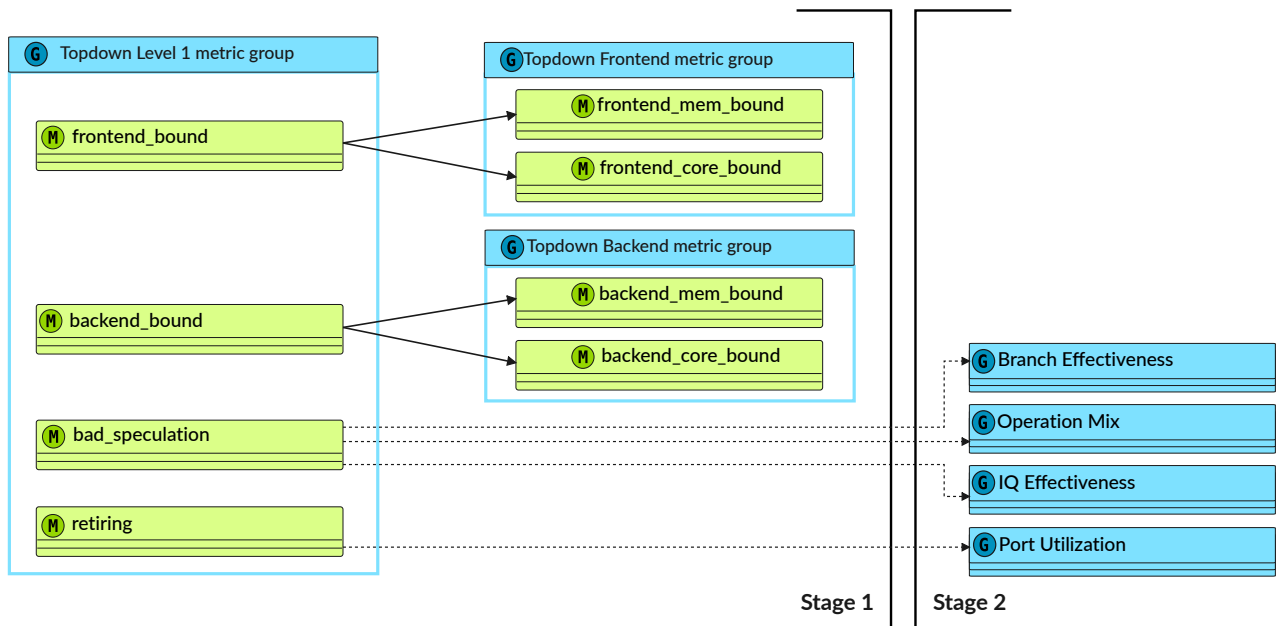
For more information about the metrics in this group and the associated formulas and events, see [Topdown_CME](#).

3.1.1 Topdown Level 1 metric group

The [Topdown_L1](#) metric group contains the first set of metrics to begin Topdown analysis of application performance. These metrics provide the percentage distribution of processor pipeline utilization.

The following figure shows the metrics for Topdown_L1 and the next steps in the methodology.

Figure 3-3: C1-Premium Metric Group: Topdown_L1



Topdown Level 1 metric group has four primary metrics [Frontend Bound metric](#), [Backend Bound metric](#), [Bad Speculation metric](#), and [Retiring metric](#).

Frontend Bound metric

The [frontend_bound](#) metric measures pipeline inefficiency due to slots that were stalled because of resource constraints in the frontend.

To analyze the data further, use the next step Stage 1 [Topdown Frontend metric group](#).

Backend Bound metric

The [backend_bound](#) metric measures pipeline inefficiency due to slots that were stalled because of resource constraints in the backend.

To analyze the data further, use the next step Stage 1 [Topdown Backend metric group](#).

Bad Speculation metric

The [bad_speculation](#) metric is the percentage of total slots that executed operations and did not retire due to a pipeline flush.

This indicates cycles that were utilized but inefficiently. This metric measures pipeline inefficiency caused by flushes in the pipeline, due to branch mispredictions or machine clears. For workloads that demonstrate high bad speculation rates, the next step is to understand the branch performance in the workload.

Some key reasons for pipeline inefficiency are the same reasons that have been identified in the `frontend_core_flush_bound` metric. This metric is part of the `frontend_core_bound` metric within the [Topdown Frontend metric group section test](#).

To analyze the data further, use the following next steps where applicable:

- Stage 1 [frontend_core_flush_bound](#) metric
- Stage 2 [Branch Effectiveness metric group](#)

Retiring metric

The [retiring](#) metric is the percentage of total slots that retired operations, which indicates cycles that were utilized efficiently. This metric covers the total slots that were utilized efficiently by the processor.

A high retirement rate can also mean that there is an opportunity for further performance optimization. Scalar code that shows high retirement can be optimized by vectorization if there is data parallelism. In some cases, the code that is executed can be made redundant or optimized. When you analyze high retirement rates, evaluation of the instruction or operation mix is a recommended next step in the characterization of execution units that are heavily utilized.

To analyze the data further, use the following next steps where applicable:

- Stage 2 [Operation Mix metric group](#)
- Stage 2 [Issue Queue Effectiveness metric group](#)
- Stage 2 [Port Utilization metric group](#)

Topdown Level 1 implementation criteria

This metric group has criteria that apply to the implementation of the methodology.

- The sum of the metrics in Topdown_L1 equals 100% of the total execution cycles in the core implementation.
- The total execution bandwidth of the core is the same as the number of execution slots for operations.
- Slots are the number of ports in the rename unit, which is considered the boundary between the frontend and backend of the core.
- A frontend stall is counted when there are no micro-ops to rename.

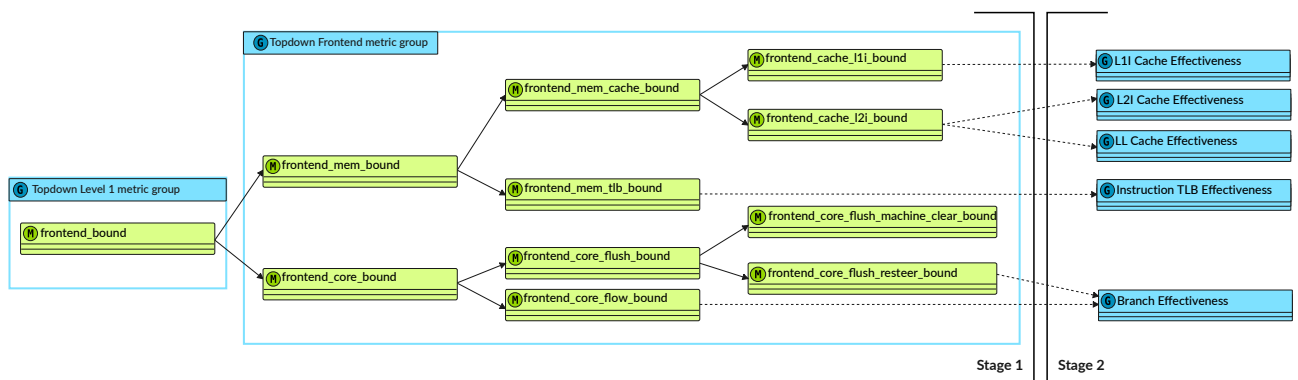
- A backend stall is counted when there are micro-ops in the rename unit, but they cannot dispatch to the backend due to backend resource constraints.
- In a cycle where there are no micro-ops to rename and no micro-ops dispatched to the backend, the CPU counts a frontend stall.
- The definition of the backend stall is linked to the slot that does not dispatch although there are micro-ops available in the rename unit.

3.1.2 Topdown Frontend metric group

The [Topdown_Frontend](#) metric group contains a set of metrics to analyze a frontend bound workload. The metric group divides frontend bound into memory bound and core bound.

The following figure shows the metrics for the Topdown_Frontend metric group and the next steps in the methodology.

Figure 3-4: C1-Premium Metric Group: Topdown_Frontend



The Topdown Frontend metric group has two primary metrics, [Frontend Memory Bound metric](#) and [Frontend Core Bound metric](#).

Frontend Memory Bound metric

The [frontend_mem_bound](#) metric is the percentage of total cycles stalled in the frontend due to frontend core resource constraints related to the instruction fetch latency issues caused by memory access components.

This metric breaks down into the following metrics:

- [frontend_mem_cache_bound](#)
 - [frontend_cache_l1i_bound](#)
 - [frontend_cache_l2i_bound](#)
- [frontend_mem_tlb_bound](#)

`frontend_mem_*_bound` stall metrics are derived ratios of Translation Lookaside Buffer (TLB) and Cache stalls such as:

- `frontend_mem_11i_bound` is a ratio of frontend memory bound cycles waiting on an L1 miss.
- `frontend_mem_12i_bound` is a ratio of frontend memory bound cycles waiting on an L2 miss.
- `frontend_mem_tlb_bound` is a ratio of frontend memory bound cycles waiting on a TLB miss.



Note

Some frontend memory bound stall reasons are not captured by the `frontend_mem_tlb_bound`, `frontend_mem_11i_bound`, or `frontend_mem_12i_bound` metrics. The unaccounted stall reasons are primarily related to microarchitectural stalls that are sometimes required because of L1I cache hits.

The following implementation criteria apply to these metrics:

- The Frontend Memory Cache Bound metric is implemented as the number of cycles of either the Frontend Memory L1I Bound metric or the Frontend Memory L2I Bound metric.
- The Frontend Memory L1I Bound metric counts when waiting on an L1 miss but not waiting on an L2 miss.

To analyze the data further, use the following next steps where applicable:

- For `frontend_cache_11i_bound`, use Stage 2 [Level 1 Instruction Cache Effectiveness metric group](#).
- For `frontend_cache_12i_bound`, use Stage 2 [Level 2 Instruction Unified Cache Effectiveness metric group](#) and Stage 2 [Last Level Cache Effectiveness metric group](#).
- For `frontend_mem_tlb_bound`, use Stage 2 [Instruction TLB Effectiveness metric group](#).

Frontend Core Bound metric

The `frontend_core_bound` metric is the percentage of total cycles stalled in the frontend due to frontend core resource constraints not related to instruction fetch latency issues caused by memory access components.

Reasons for a stall in the frontend due to the core units are as follows:

- Flush: Refers to the frontend resteer that cause PC updates, which can either be:
 - A flush that is caused by mispredictions
 - Other machine clears due to microarchitectural reasons such as exceptions and memory hazards.
- Flow: Refers to the frontend structural flow stall that is caused in the decode unit fetches because of a branch prediction unit that does not provide predictions on time.



Note

Some frontend core bound stall reasons are not captured by the `frontend_core_flush_bound` and `frontend_core_flow_bound` metrics. Examples of these stalls include, but are not limited to:

- Stalls that are required due to the microarchitecture of the branch predictor and instruction cache interaction.

- Stalls that are generated for improved power efficiency.
-

This metric breaks down into the following metrics:

- [frontend_core_flush_bound](#)
- [frontend_core_flush_resteer_bound](#)
- [frontend_core_flush_machine_clear_bound](#)
- [frontend_core_flow_bound](#)

To analyze the data further, use the next step Stage 2 [Branch Effectiveness metric group](#).

Topdown Frontend implementation criteria

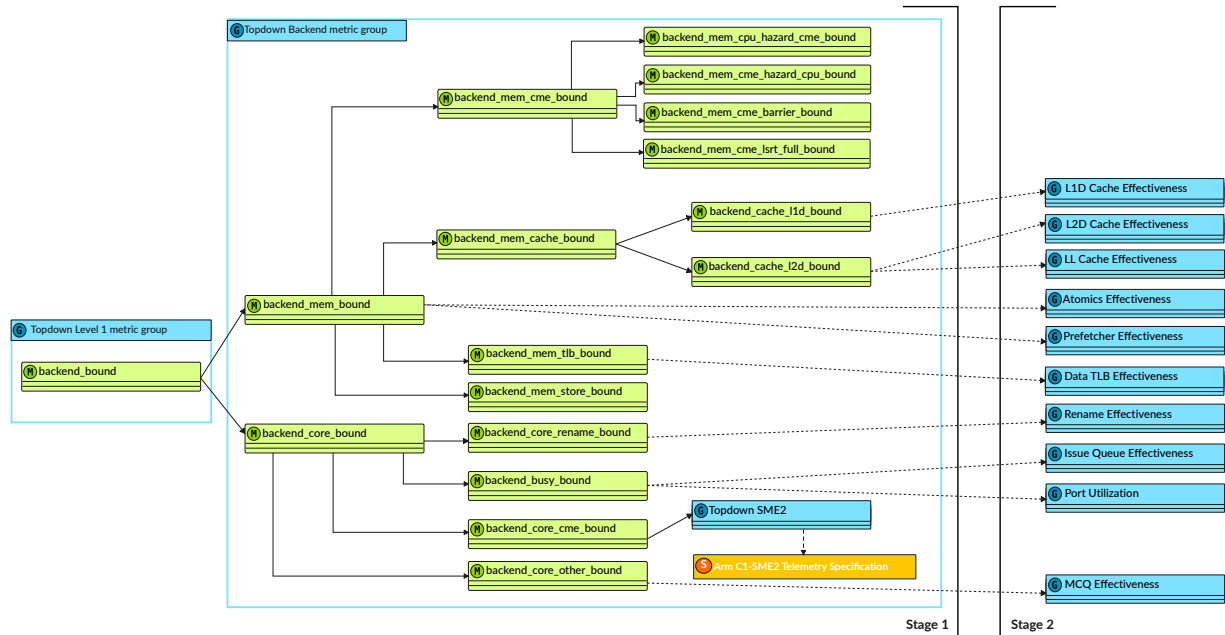
This metric group has criteria that apply to the implementation of the methodology.

- The implementation of the [Frontend Core Bound](#) metric in C1-Premium is the difference between the [Frontend Bound](#) and [Frontend Memory Bound](#) metrics. This means that the cycle was stalled in the frontend and the stall was not caused by the frontend memory units. This derivation is mathematically correct but misses a direct count of the [Frontend Core Bound](#) metric from an implementation standpoint.
- Due to the implementation, the sum of the Frontend Core Bound and Frontend Memory Bound metrics is equal to 100% of the Frontend Bound metric.

3.1.3 Topdown Backend metric group

The [Topdown_Backend](#) metric group contains a set of metrics to analyze a backend bound workload. The metric group divides backend bound into memory bound and core bound.

The following figure shows the metrics for the Topdown_Backend metric group and the next steps in the methodology.

Figure 3-5: C1-Premium Metric Group: Topdown_Backend

Topdown Backend metric group has two primary metrics, [Backend Memory Bound metric](#) and [Backend Core Bound metric](#).

Backend Memory Bound metric

The [backend_mem_bound](#) metric is the percentage of total cycles stalled in the backend due to backend core resource constraints related to memory access latency issues caused by memory access components.

[backend_mem_bound](#) counts the stall cycles in which no operation can be sent to the dispatch unit and a memory operation is being executed. Additional stalls are introduced due to SME2 unit and CPU sharing the CPU memory subsystem. The SME2 unit and CPU are synchronized through the Load Store Registers (LSRT) block, causing both execution units to have dependencies on data accesses. New stalls can be analyzed using metrics starting with [backend_mem_cme_bound](#).

This metric breaks down into the following metrics:

- [backend_mem_cache_bound](#)
 - [backend_cache_l1d_bound](#)
 - [backend_cache_l2d_bound](#)
- [backend_mem_tlb_bound](#)
- [backend_mem_store_bound](#)
- [backend_mem_cme_bound](#)
 - [backend_mem_cme_barrier_bound](#)

- [backend_mem_cme_lsrt_full_bound](#)
- [backend_mem_cpu_hazard_cme_bound](#)
- [backend_mem_cme_hazard_cpu_bound](#)

`backend_mem_*_bound` stall metrics are derived ratios of TLB and Cache stalls such as:

- `backend_mem_11d_bound` is a ratio of backend memory bound cycles waiting on an L1 data cache miss.
- `backend_mem_12d_bound` is a ratio of backend memory bound cycles waiting on an L2 unified cache miss.
- `backend_mem_tlb_bound` is a ratio of backend memory bound cycles waiting on data for a TLB miss or walk.

The following implementation criteria apply to these metrics:

- The Backend Memory Cache Bound metric is implemented as the number of cycles of either the Backend Memory L1D Bound metric or the Backend Memory L2D Bound metric.
- The Backend Memory L1D Bound metric counts when waiting on an L1 miss but not waiting on an L2 miss.
- The Backend Memory Store Bound metric also counts backend memory bound stall cycles while stores are outstanding.

To analyze the data further, use the following steps where applicable:

- For `backend_cache_11d_bound`, use Stage 2 [Level 1 Data Cache Effectiveness metric group](#).
- For `backend_cache_12d_bound`, use Stage 2 [Level 2 Data Unified Cache Effectiveness metric group](#) and Stage 2 [Last Level Cache Effectiveness metric group](#).
- For `backend_mem_tlb_bound`, use Stage 2 [Data TLB Effectiveness metric group](#).
- For `backend_mem_bound`, use Stage 2 [Atomics Effectiveness metric group](#) and Stage 2 [Prefetcher Effectiveness metric group](#).

There is no further analysis for:

- `backend_mem_store_bound`
- `backend_mem_cme_barrier_bound`
- `backend_mem_cpu_hazard_cme_bound`
- `backend_mem_cme_hazard_cme_bound`
- `backend_mem_cme_lsrt_full_bound`

Backend Core Bound metric

The [backend_core_bound](#) metric is the percentage of total cycles stalled in the backend due to backend core resource constraints not related to instruction fetch latency issues caused by memory access components.

Reasons for a stall in the backend due to the core units are as follows:

- **Rename:** Refers to rename stalls caused by the register renaming unit that sends operations to the dispatch unit.
- **Dispatch Busy:** Refers to execution stalls caused by any of the following:
 - Issue queue unit stalls
 - Execution port starvation
 - Data dependency issues for operations to make further progress in execution
- **Other:** Refers to stalls caused by other backend microarchitectural resources being unavailable, for example, program counter tracking.
- **CME:** Refers to stalls caused by a Scalable Matrix Extension 2 (SME2) resource availability, either waiting for arbitration or waiting for execution to complete.

This metric breaks down into the following metrics:

- [backend_core_cme_bound](#)
- [backend_core_rename_bound](#)
- [backend_core_other_bound](#)
- [backend_busy_bound](#)

To analyze the data further, use the following steps where applicable:

- For [backend_core_cme_bound](#), use [Topdown SME2 metric group](#).
- For [backend_core_rename_bound](#), use Stage 2 [Rename Effectiveness metric group](#).
- For [backend_busy_bound](#), use Stage 2 [Issue Queue Effectiveness metric group](#) and the Stage 2 [Port Utilization metric group](#).
- For [backend_core_other_bound](#) use Stage 2 [Main Commit Queue Effectiveness metric group](#).

Topdown Backend implementation criteria

This metric group has criteria that apply to the implementation of the methodology.

- The implementation of the Backend Core Bound metric in C1-Premium is the difference between the Backend Bound and Backend Memory Bound metrics. This means that the cycle was stalled in the backend and the stall was not caused by the backend memory units. This derivation is mathematically correct but misses a direct count of the Backend Core Bound metric from an implementation standpoint.
- Due to the implementation, the sum of the Backend Core Bound and Backend Memory Bound metrics is equal to 100% of the Backend Bound metric.

3.1.4 Topdown SME2 metric group

The [Topdown SME2](#) metric group contains a set of metrics to analyze an SME2 bound workload.

This metric breaks down into the following metrics:

- [backend_cme_backpressure_bound](#)

- [backend_cme_busy_arb_bound](#)
- [backend_cme_cpu_bound](#)

To analyze the data further, use the following next steps where applicable:

- For `backend_cme_backpressure_bound`, analyze the bottlenecks that are caused by the resources that the SME2 unit manages, as described in the [Arm® C1-Scalable Matrix Extension 2 Telemetry Specification](#).
- There is no further analysis for:
 - `backend_cme_busy_arb_bound`
 - `backend_cme_cpu_bound`

Topdown SME2 implementation criteria

This metric group has criteria that apply to the implementation of the methodology for the SME2 unit.

For more information, see the [Arm® C1-Scalable Matrix Extension 2 Telemetry Specification](#).

3.2 Stage 2: Microarchitecture exploration

Topdown analysis Stage 2 helps to locate the hot spots in the code and conduct root cause analysis. This stage reveals more details about the CPU component that caused the bottleneck and was identified during the Stage 1 process.

Stage 2 contains metric groups for microarchitecture exploration targeted at CPU resource effectiveness and other relevant metrics. Metrics in this category are designed for users who are interested in the microarchitectural characteristics of key CPU components.

The following figure shows the Stage 2 metric groups for C1-Premium.

Figure 3-6: C1-Premium Topdown methodology Stage 2 overview

Stage 2 Groups	
⑥ L1I_Cache_Effectiveness l1i_cache_mпки l1i_cache_miss_ratio	⑥ DTLB_Effectiveness dtlb_mпки dtlb_walk_ratio dtlb_walk_large_ratio dtlb_walk_small_ratio dtlb_walk_page_ratio dtlb_walk_block_ratio l1d_tlb_mпки l1d_tlb_miss_ratio l2_tlb_mпки l2_tlb_miss_ratio dtlb_walk_average_depth dtlb_walk_average_latency
⑥ L1D_Cache_Effectiveness l1d_cache_mпки l1d_cache_miss_ratio	⑥ Port_Utilization vpu_port_utilization branch_port_utilization int_port_utilization lsu_port_utilization std_port_utilization
⑥ L2I_Cache_Effectiveness l2i_cache_mпки l2i_cache_miss_ratio	⑥ Atomics_Effectiveness cas_near_fail_ratio cas_near_pass_ratio cas_near_ratio cas_far_ratio lse_atomics_ratio lse_load_ratio lse_store_ratio
⑥ L2D_Cache_Effectiveness l2d_cache_mпки l2d_cache_miss_ratio	⑥ Prefetcher_Effectiveness l1_prefetcher_coverage l1_prefetcher_accuracy l1_prefetcher_timeliness l2_prefetcher_coverage_l1hwprf_exclusive l2_prefetcher_accuracy_l1hwprf_exclusive l2_prefetcher_timeliness_l1hwprf_exclusive l2_prefetcher_coverage_l1hwprf_inclusive l2_prefetcher_accuracy_l1hwprf_inclusive l2_prefetcher_timeliness_l1hwprf_inclusive
⑥ L3_Cache_Effectiveness l3_cache_mпки l3_cache_miss_ratio	⑥ IQ_Effectiveness iq_stall_lsu_percentage iq_stall_sx_percentage iq_stall_mx_percentage iq_stall_vpu_percentage
⑥ LL_Cache_Effectiveness ll_cache_read_mпки ll_cache_read_miss_ratio ll_cache_read_hit_ratio	⑥ Operation_Mix load_percentage store_percentage integer_dp_percentage simd_percentage scalar_fp_percentage branch_percentage crypto_percentage
⑥ ITLB_Effectiveness itlb_mпки itlb_walk_ratio itlb_walk_large_ratio itlb_walk_small_ratio itlb_walk_page_ratio itlb_walk_block_ratio l1i_tlb_mпки l1i_tlb_miss_ratio l2_tlb_mпки l2_tlb_miss_ratio itlb_walk_average_depth itlb_walk_average_latency	⑥ Bus_Effectiveness bus_access_average_count bus_read_requests_average_latency
⑥ System_Memory_Effectiveness system_dram_mem_hit_ratio system_l3_cache_hit_ratio system_llc_cache_hit_ratio system_peer_cluster_cache_hit_ratio	⑥ Average_Latency bus_read_requests_average_latency dtlb_walk_average_latency instruction_fetch_average_latency itlb_walk_average_latency load_average_latency
⑥ Branch_Effectiveness branch_mпки branch_misprediction_ratio branch_direct_ratio branch_indirect_ratio branch_return_ratio	⑥ SVE_Effectiveness sve_predicate_empty_percentage sve_predicate_full_percentage sve_predicate_partial_percentage
⑥ Cycle_Accounting backend_stalled_cycles cme_alloc_cycles_ratio cme_arb_pending_cycles frontend_stalled_cycles sm_active_cycles_ratio za_active_cycles_ratio	⑥ FP_Precision_Mix fp16_percentage fp32_percentage fp64_percentage
⑥ Rename_Effectiveness rename_stall_vec_ratio rename_stall_pred_ratio rename_stall_flags_ratio rename_stall_int_ratio	
⑥ MCQ_Effectiveness mcq_stall_percentage	
⑥ FP_Arithmetic_Intensity fp_ops_per_cycle nonsve_fp_ops_per_cycle sve_fp_ops_per_cycle	

In C1-Premium there are two metric groups based on industry standard metrics, Misses Per Kilo Instructions (MPKI) and Miss Ratios, plus several other Stage 2 metric groups.

Level 1 Data Cache Effectiveness metric group

The [L1D_Cache_Effectiveness](#) metric group contains metrics to evaluate the effectiveness of the L1 Data Cache on the processor.

The metrics in this metric group include cache misses per thousand instructions and miss to access ratio.

Level 1 Instruction Cache Effectiveness metric group

The [L1I_Cache_Effectiveness](#) metric group contains metrics to evaluate the effectiveness of the L1 Instruction Cache on the processor.

The metrics in this metric group include cache misses per thousand instructions and miss to access ratio.

Level 2 Data Unified Cache Effectiveness metric group

The [L2D_Cache_Effectiveness](#) metric group contains metrics to evaluate the effectiveness of the L2 Unified Cache data accesses on the processor.

The metrics in this metric group include cache misses per thousand instructions and miss to access ratio.

Level 2 Instruction Unified Cache Effectiveness metric group

The [L2I_Cache_Effectiveness](#) metric group contains metrics to evaluate the effectiveness of the L2 Unified Cache instruction accesses on the processor.

The metrics in this metric group include cache misses per thousand instructions and miss to access ratio.

Level 3 Cache Effectiveness metric group

The [L3_Cache_Effectiveness](#) metric group contains metrics to evaluate the effectiveness of the L3 Unified Cache on this processor.

Last Level Cache Effectiveness metric group

The [LL_Cache_Effectiveness](#) metric group contains metrics to evaluate the effectiveness of the Last Level Cache on the processor.

In systems that support a shared System Level Cache (SLC) in the interconnect that is configured to count Last Level Cache events:

- The [Last level cache access, read](#) event counts total SLC accesses made by the core.
- The [Last level cache miss, read](#) event counts accesses missed at the SLC.

The [ll_cache_read_mpki](#) and [ll_cache_read_miss_ratio](#) metrics can be used to analyze the last level read behavior.

Another useful metric to measure the SLC hit percentage for read traffic is [ll_cache_read_hit_ratio](#).

Last level cache events do not have a write variant in C1-Premium because the SLC is only used as an eviction cache for the core. In addition, all the writes complete early at the interconnect when the transaction is acknowledged but always completed.

Data TLB Effectiveness and Instruction TLB Effectiveness metric groups

The [DTLB_Effectiveness](#) and [ITLB_Effectiveness](#) metric groups contain metrics to evaluate the effectiveness of the data TLB and instruction TLB, respectively, on the processor.

An important performance evaluation step is to check the virtual memory system performance, which affects the instruction fetch performance in the frontend and memory access performance on the data side.

The processor translates a virtual address into a physical address for any instruction or data memory access before it accesses the respective cache.



A program's view of memory is the virtual address, but the processor works with the physical address when it accesses cache or memory.

Virtual to physical mappings are defined in the page translation tables which reside in system memory. Access to these tables requires one or more memory operations that take many cycles to complete and are known as a translation table walk. However, TLBs cache these translation table walks, which significantly reduces the number of accesses to system memory and makes translations faster.

Several key metrics can be used to evaluate the TLB effectiveness and the cost of latency that is specifically caused by translation table walks:

- The [itlb_mpki](#) and [dtlb_mpki](#) metrics provide the rate of TLB Walks per kilo instructions for instruction and data accesses, respectively. These metrics help to evaluate and correlate the TLB efficiency with respect to the total number of instructions.
- The [dtlb_walk_ratio](#) metric provides the ratio of DTLB Walks to the overall TLB lookups made by the program. This metric is the same as the [DTLB_WALK](#) and [MEM_ACCESS](#) events because every MEM_ACCESS causes a [L1D_TLB](#) access.
- The [itlb_walk_ratio](#) metric provides a percentage of ITLB walks to the overall TLB lookups initiated from the instruction side.

Metrics are defined to provide insight into the depth of translation tables walks that utilize the [ITLB_STEP](#) and [DTLB_STEP](#) events. These counters count all memory accesses to both stage 1 and stage 2 translation tables for a translation table walk. Address translation mappings with an excessive number of steps per translation might impact performance.

Additionally, metrics are defined to provide ratios of walks that result in blocks versus pages which utilize [ITLB_WALK_BLOCK](#), [ITLB_WALK_PAGE](#), [DTLB_WALK_BLOCK](#), and [DTLB_WALK_PAGE](#) events.



The implementation for this CPU defines BLOCK/LARGE to be greater to or equal to 2MB, and PAGE/SMALL to be less than 2MB.

Atomics Effectiveness metric group

The [Atomics_Effectiveness](#) metric group contains metrics to evaluate the effectiveness of atomics execution on the processor.

The metric group includes metrics for CAS atomics, as well as LSE atomics. The metric group provides metrics to analyze passing and failing atomic operations.

Average Latency metric group

The [Average_Latency](#) metric group contains metrics that provide average latencies of costly memory related operations by the processor that can take a variable number of cycles to execute.

The metrics in this metric group utilize PMU counters that aggregate the number of outstanding operations per cycle such as [MEM_ACCESS_RD_PERCYC](#) to derive the average latency.

Bus Effectiveness metric group

The [Bus_Effectiveness](#) metric group contains metrics to evaluate the effectiveness of bus transactions issued by this processor.

Branch Effectiveness metric group

The [Branch_Effectiveness](#) metric group contains metrics to evaluate the effectiveness of branch instruction execution on the processor.

Branch mispredictions are costly in a deeply pipelined CPU, causing pipeline flushes and wasted cycles. Although modern CPUs have optimized branch prediction units, there are many use cases that are branch heavy and hard to predict.

Two key performance metrics can be used for a high-level evaluation of the branch execution performance regarding the overall program execution:

- The [branch_mпки](#) metric provides total branch mispredictions per kilo instructions.
- The [branch_misprediction_ratio](#) metric can indicate the ratio of branches that were mispredicted to the overall branches.

Branch prediction units work differently depending on the branch type. The main components are:

Branch History Table (BHT)

Stores the history of conditional branches, taken or not.

Branch Target Buffer (BTB)

Stores the target address for indirect branches.

Return Address Stack (RAS)

Stores the function return branches.

C1-Premium supports the following three events that categorize the immediate, indirect, and return branches executed, respectively:

- [BR_IMMED_SPEC](#)
- [BR_INDIRECT_SPEC](#)
- [BR_RETURN_SPEC](#)

You can get a breakdown of the branch type to help investigate each subunit within the branch prediction unit.

Operation Mix metric group

The [Operation_Mix](#) metric group provides the distribution of micro-operation types executed for the program.

The C1-Premium microarchitecture has a variety of execution units that can process more than seven types of operations:

- Branch
- Single-cycle integers
- Multicycle integers
- Load Store Unit with address generation
- Advanced FP/SIMD operations
- Scalable Vector Extension (SVE)
- Scalable Matrix Extension (SME)

The operations that are issued to these execution units can be counted by the PMU events in the Operation Mix metric group.



Load/store instructions that behave as both a load and a store are reflected once in the [load_store_percentage](#).

Prefetcher Effectiveness metric group

The [Prefetcher_Effectiveness](#) metric group contains metrics to evaluate the effectiveness of the L1 and L2 data prefetchers on the processor.

These metrics include:

- Coverage, a measure of eliminated prefetches
- Accuracy, a measure of useful prefetches
- Timeliness, a measure of appropriate timing of prefetches

SVE Effectiveness metric group

The [SVE_Effectiveness](#) metric group provides metrics to evaluate the effectiveness of predicated SVE instruction execution on this processor.

Floating Point Precision metric group

The [FP_Precision_Mix](#) metric group provides metrics to evaluate tmix of precision of floating point instruction execution on this processor.

Floating Point Arithmetic Intensity metric group

The [FP_Arithmetic_Intensity](#) metric group provides metrics to evaluate the effectiveness of floating point operation execution on this processor.

The metrics evaluate operations by computation and vector lanes, metrics are provided for scalable vector operations and non scalable vector operations.

Issue Queue Effectiveness metric group

The [IQ_Effectiveness](#) metric group provides relative stall cycles due to issue queue fullness by operation type.

C1-Premium may have multiple issue queues for an operation type.

Main Commit Queue Effectiveness metric group

The [MCQ_Effectiveness](#) metric group provides relative stall information for out of order speculation depth stalls.

The main commit queue is responsible for maintaining instruction retirement in program order. Although this queue is generally sized to be a non-limiting resource, it can be limited in some cases.

Port Utilization metric group

The [Port_Utilization](#) metric group provides relative utilization of execution ports for the program. The utilization metrics provide average operation execution per cycle for different execution ports.

Rename Effectiveness metric group

The [Rename_Effectiveness](#) metric group provides relative stall ratios for register renaming by register type.

Register renaming is grouped by flag, integer, vector, and predicate registers. A lack of available physical registers in any of these groups may result in a backend stall until a physical register becomes available.



Note

The implementation for this CPU includes the following cycles in the metric calculations:

- Cycles which are fully stalled due to register renaming
- Cycles in which some operations were issued but not all operations were issued due to unavailable registers

System Memory Effectiveness metric group

The [System_Memory_Effectiveness](#) metric group contains a set of metrics to analyze a system memory bound workload. They provide hierarchical data hits in system memory resources internal or external to the processor.

This metric group provides memory access metrics for DRAM, L3 Cache, last level Cache, and peer cluster cache.

For systems with multiple sockets or SoCs, C1-Premium supports the `REMOTE_ACCESS` event, which counts the memory transactions that were completed by a subordinate source from another chip.

Cycle Accounting metric group

The [Cycle_Accounting](#) metric group contains a set of metrics that measure the percentage of processor cycles stalled in either frontend or backend of the processor.

General metric group

The [General](#) metric group contains general CPU metrics for performance analysis such as Instructions Per Cycle (IPC).

Misses Per Kilo Instructions metric group

The [MPKI](#) metric group contains metrics for different CPU resources that can be measured as misses per kilo instructions. These metrics can be used to normalize the misses in CPU components against the total instructions executed. The primary components are branches, caches, and TLBs.

MPKI is an industry-standard metric that can also help with comparisons across different implementations of the Arm architecture, because instructions retired should count the same on all AArch64-based microarchitectures.



The MPKI metric group is a Stage 2 metric but is not included in the methodology decision tree.

Miss Ratio metric group

The [Miss_Ratio](#) metric group contains metrics to measure miss ratios of different processor resources. These metrics can be used to calculate the ratio of misses in CPU components against the total accesses in those components. The primary components are branches, caches, and TLBs.

Miss Ratio metrics provide insights into the efficiency of each CPU component in the pipeline and help to find the root cause of issues.



The Miss_Ratio metric group is a Stage 2 metric but is not included in the methodology decision tree.

3.3 Core memory operations

The `MEM_ACCESS` event counts the total number of memory operations that were issued by the Load Store Unit (LSU) of the core.

Because these operations are looked up in the `L1D_CACHE` first, both the `L1D_CACHE` and `MEM_ACCESS` events count at the same rate.

C1-Premium also supports two additional events, `MEM_ACCESS_RD` and `MEM_ACCESS_WR` that can provide the read and write traffic breakdown respectively. These events are not the same as the `LD_SPEC` and `ST_SPEC` events because they count memory operations that are speculatively issued but not always executed.

4. C1-Premium Telemetry cheat-sheets and lookup tables

The cheat-sheets and lookup tables enable you to find and access metrics and events in different ways.

Cheat-sheets

Both metrics and events are listed by metric groups.

Lookup tables

Metrics are listed alphabetically, with the related events, and metric groups.

Events are listed by code number, with the related metrics, metric groups, and functional groups.

4.1 Metrics cheat sheet for C1-Premium

Metrics are listed in their respective metric groups. Some metrics are used in more than one metric group.

C1-Premium specification provides the following types of metrics:

- Total implemented Common metrics: 145

Topdown Backend (17)	Topdown SME2 (3)	Cycle Accounting (6)
<ul style="list-style-type: none"> • backend_busy_bound • backend_cache_l1d_bound • backend_cache_l2d_bound • backend_core_bound • backend_core_cme_bound • backend_core_other_bound • backend_core_rename_bound • backend_core_spec_throttle_bound • backend_mem_bound • backend_mem_cache_bound • backend_mem_cme_barrier_bound • backend_mem_cme_bound • backend_mem_cme_hazard_cpu_bound • backend_mem_cme_lsrt_full_bound • backend_mem_cpu_hazard_cme_bound • backend_mem_store_bound • backend_mem_tlb_bound 	<ul style="list-style-type: none"> • backend_cme_backpressure_bound • backend_cme_busy_arb_bound • backend_cme_cpu_bound 	<ul style="list-style-type: none"> • backend_stalled_cycles • cme_alloc_cycles_ratio • cme_arb_pending_ratio • frontend_stalled_cycles • sm_active_cycles_ratio • za_active_cycles_ratio

Speculative Operation Mix (13)	Topdown Level 1 (4)	Topdown Frontend (11)
<ul style="list-style-type: none"> barrier_percentage branch_percentage crypto_percentage integer_dp_percentage load_ls_percentage load_percentage load_store_percentage scalar_fp_percentage simd_percentage sme_percentage store_ls_percentage store_percentage sve_percentage 	<ul style="list-style-type: none"> backend_bound bad_speculation frontend_bound retiring 	<ul style="list-style-type: none"> frontend_cache_l1i_bound frontend_cache_l2i_bound frontend_core_bound frontend_core_flow_bound frontend_core_flush_bound frontend_core_flush_machine_clear_bound frontend_core_flush_resteer_bound frontend_core_spec_throttle_bound frontend_mem_bound frontend_mem_cache_bound frontend_mem_tlb_bound
General (1)	Misses Per Kilo Instructions (14)	Miss Ratio (12)
<ul style="list-style-type: none"> ipc 	<ul style="list-style-type: none"> branch_mpki dtlb_mpki itlb_mpki l1d_cache_demand_mpki l1d_cache_mpki l1d_tlb_mpki l1i_cache_mpki l1i_tlb_mpki l2_tlb_mpki l2d_cache_demand_mpki l2d_cache_mpki l2i_cache_mpki l3_cache_mpki ll_cache_read_mpki 	<ul style="list-style-type: none"> branch_misprediction_ratio dtlb_walk_ratio itlb_walk_ratio l1d_cache_miss_ratio l1d_tlb_miss_ratio l1i_cache_miss_ratio l1i_tlb_miss_ratio l2_tlb_miss_ratio l2d_cache_miss_ratio l2i_cache_miss_ratio l3_cache_miss_ratio ll_cache_read_miss_ratio
SVE Effectiveness (4)	Floating Point Arithmetic Intensity (3)	Floating Point Precision (3)
<ul style="list-style-type: none"> sve_predicate_empty_percentage sve_predicate_full_percentage sve_predicate_partial_percentage sve_predicate_percentage 	<ul style="list-style-type: none"> fp_ops_per_cycle nonsve_fp_ops_per_cycle sve_fp_ops_per_cycle 	<ul style="list-style-type: none"> fp16_percentage fp32_percentage fp64_percentage

Branch Effectiveness (5)	Instruction TLB Effectiveness (12)	Data TLB Effectiveness (12)
<ul style="list-style-type: none"> branch_direct_ratio branch_indirect_ratio branch_misprediction_ratio branch_mpki branch_return_ratio 	<ul style="list-style-type: none"> itlb_mpki itlb_walk_average_depth itlb_walk_average_latency itlb_walk_block_ratio itlb_walk_large_ratio itlb_walk_page_ratio itlb_walk_ratio itlb_walk_small_ratio l1i_tlb_miss_ratio l1i_tlb_mpki l2_tlb_miss_ratio l2_tlb_mpki 	<ul style="list-style-type: none"> dtlb_mpki dtlb_walk_average_depth dtlb_walk_average_latency dtlb_walk_block_ratio dtlb_walk_large_ratio dtlb_walk_page_ratio dtlb_walk_ratio dtlb_walk_small_ratio l1d_tlb_miss_ratio l1d_tlb_mpki l2_tlb_miss_ratio l2_tlb_mpki
L1 Instruction Cache Effectiveness (2)	L1 Data Cache Effectiveness (3)	L2D Data Unified Cache Effectiveness (3)
<ul style="list-style-type: none"> l1i_cache_miss_ratio l1i_cache_mpki 	<ul style="list-style-type: none"> l1d_cache_demand_mpki l1d_cache_miss_ratio l1d_cache_mpki 	<ul style="list-style-type: none"> l2d_cache_demand_mpki l2d_cache_miss_ratio l2d_cache_mpki
L2 Instruction Unified Cache Effectiveness (2)	L3 Unified Cache Effectiveness (2)	Last Level Cache Effectiveness (3)
<ul style="list-style-type: none"> l2i_cache_miss_ratio l2i_cache_mpki 	<ul style="list-style-type: none"> l3_cache_miss_ratio l3_cache_mpki 	<ul style="list-style-type: none"> ll_cache_read_hit_ratio ll_cache_read_miss_ratio ll_cache_read_mpki
Register Rename Effectiveness (4)	Issue Queue Effectiveness (4)	Main Commit Queue Effectiveness (1)
<ul style="list-style-type: none"> rename_stall_flags_ratio rename_stall_int_ratio rename_stall_pred_ratio rename_stall_vec_ratio 	<ul style="list-style-type: none"> iq_stall_lsu_percentage iq_stall_mx_percentage iq_stall_sx_percentage iq_stall_vpu_percentage 	<ul style="list-style-type: none"> mcq_stall_percentage

Execution Unit Effectiveness (5)	Prefetcher Effectiveness (9)	Atomics Effectiveness (7)
<ul style="list-style-type: none"> branch_port_utilization int_port_utilization lsu_port_utilization std_port_utilization vpu_port_utilization 	<ul style="list-style-type: none"> l1_prefetcher_accuracy l1_prefetcher_coverage l1_prefetcher_timeliness l2_prefetcher_accuracy_l1hwprf_exclusive l2_prefetcher_accuracy_l1hwprf_inclusive l2_prefetcher_coverage_l1hwprf_exclusive l2_prefetcher_coverage_l1hwprf_inclusive l2_prefetcher_timeliness_l1hwprf_exclusive l2_prefetcher_timeliness_l1hwprf_inclusive 	<ul style="list-style-type: none"> cas_far_ratio cas_near_fail_ratio cas_near_pass_ratio cas_near_ratio lse_atomics_ratio lse_load_ratio lse_store_ratio
Average Latency (5)	Bus Effectiveness (2)	System Memory Effectiveness (4)
<ul style="list-style-type: none"> bus_read_requests_average_latency dtlb_walk_average_latency instruction_fetch_average_latency itlb_walk_average_latency load_average_latency 	<ul style="list-style-type: none"> bus_access_average_count bus_read_requests_average_latency 	<ul style="list-style-type: none"> system_dram_mem_hit_ratio system_l3_cache_hit_ratio system_llc_cache_hit_ratio system_peer_cluster_cache_hit_ratio

4.2 PMU events cheat sheet for C1-Premium

Events are listed in their respective metric groups. Some events are not used in the Methodology, therefore are not shown in the cheat sheet.

C1-Premium specification provides the following types of PMU events:

- Total implemented Common events: 311
- Total Implemented Product ImpDef events: 61
- PMU Only events : 61
- ETE Only events : 3

Topdown Backend (17)	Topdown SME2 (4)	Cycle Accounting (8)
<ul style="list-style-type: none"> IMP_STALL_BACKEND_CPUBOUND_OTHER IMP_STALL_BACKEND_SPEC_THROT STALL_BACKEND STALL_BACKEND_BUSY STALL_BACKEND_BUSY_CME STALL_BACKEND_CPUBOUND STALL_BACKEND_L1D STALL_BACKEND_MEM STALL_BACKEND_MEMBOUND STALL_BACKEND_MEM_CME STALL_BACKEND_MEM_CME_BARRIER STALL_BACKEND_MEM_CME_HZ_ON_CPU STALL_BACKEND_MEM_CME_LSRT_FULL STALL_BACKEND_MEM_CPU_HZ_ON_CME STALL_BACKEND_RENAME STALL_BACKEND_ST STALL_BACKEND_TLB 	<ul style="list-style-type: none"> STALL_BACKEND_BUSY_CME STALL_BACKEND_BUSY_CMEBOUND STALL_BACKEND_BUSY_CME_ARB STALL_BACKEND_BUSY_CME_CPUBOUND 	<ul style="list-style-type: none"> CPU_CYCLES CYCLES_ARB_PENDING_CME CYCLES_CME_ALLOC IMP_WFX_CLOCK_CYCLES SM_ACTIVE_CYCLES STALL_BACKEND STALL_FRONTEND ZA_ACTIVE
Speculative Operation Mix (14)	Topdown Level 1 (8)	Topdown Frontend (12)
<ul style="list-style-type: none"> ASE_SPEC CRYPTO_SPEC DMB_SPEC DP_SPEC DSB_SPEC INST_SPEC ISB_SPEC LDST_SPEC LD_SPEC PC_WRITE_SPEC SME_INST_SPEC ST_SPEC SVE_SPEC VFP_SPEC 	<ul style="list-style-type: none"> CPU_CYCLES IMP_WFX_CLOCK_CYCLES OP_RETIRED OP_SPEC STALL_FRONTEND_FLUSH STALL_SLOT STALL_SLOT_BACKEND STALL_SLOT_FRONTEND 	<ul style="list-style-type: none"> IMP_STALL_FRONTEND_FLUSH_CLEAR IMP_STALL_FRONTEND_FLUSH_RESTEER IMP_STALL_FRONTEND_SPEC_THROT IMP_WFX_CLOCK_CYCLES STALL_FRONTEND STALL_FRONTEND_CPUBOUND STALL_FRONTEND_FLOW STALL_FRONTEND_FLUSH STALL_FRONTEND_L1I STALL_FRONTEND_MEM STALL_FRONTEND_MEMBOUND STALL_FRONTEND_TLB

General (3)	Misses Per Kilo Instructions (17)	Miss Ratio (22)
<ul style="list-style-type: none"> CPU_CYCLES IMP_WFX_CLOCK_CYCLES INST_RETIRED 	<ul style="list-style-type: none"> BR_MIS_PRED_RETIRED DTLB_WALK INST_RETIRED ITLB_WALK L1D_CACHE_REFILL L1D_CACHE_REFILL_RD L1D_CACHE_REFILL_WR L1D_TLB_REFILL L1I_CACHE_REFILL L1I_TLB_REFILL L2D_CACHE_REFILL L2D_CACHE_REFILL_RD L2D_CACHE_REFILL_WR L2D_TLB_REFILL L2I_CACHE_REFILL L3D_CACHE_REFILL LL_CACHE_MISS_RD 	<ul style="list-style-type: none"> BR_MIS_PRED_RETIRED BR_RETIRED DTLB_WALK ITLB_WALK L1D_CACHE L1D_CACHE_REFILL L1D_TLB L1D_TLB_REFILL L1I_CACHE L1I_CACHE_REFILL L1I_TLB L1I_TLB_REFILL L2D_CACHE L2D_CACHE_REFILL L2D_TLB L2D_TLB_REFILL L2I_CACHE L2I_CACHE_REFILL L3D_CACHE L3D_CACHE_REFILL LL_CACHE_MISS_RD LL_CACHE_RD
SVE Effectiveness (5)	Floating Point Arithmetic Intensity (4)	Floating Point Precision (4)
<ul style="list-style-type: none"> INST_SPEC SVE_PRED_EMPTY_SPEC SVE_PRED_FULL_SPEC SVE_PRED_PARTIAL_SPEC SVE_PRED_SPEC 	<ul style="list-style-type: none"> CPU_CYCLES FP_FIXED_OPS_SPEC FP_SCALE_OPS_SPEC IMP_WFX_CLOCK_CYCLES 	<ul style="list-style-type: none"> FP_DP_SPEC FP_HP_SPEC FP_SP_SPEC INST_SPEC

Branch Effectiveness (6)	Instruction TLB Effectiveness (12)	Data TLB Effectiveness (12)
<ul style="list-style-type: none"> BR_IMMED_RETIRED BR_IND_RETIRED BR_MIS_PRED_RETIRED BR_RETIRED BR_RETURN_RETIRED INST_RETIRED 	<ul style="list-style-type: none"> INST_RETIRED ITLB_STEP ITLB_WALK ITLB_WALK_BLOCK ITLB_WALK_LARGE ITLB_WALK_PAGE ITLB_WALK_PERCYC ITLB_WALK_SMALL L1I_TLB L1I_TLB_REFILL L2D_TLB L2D_TLB_REFILL 	<ul style="list-style-type: none"> DTLB_STEP DTLB_WALK DTLB_WALK_BLOCK DTLB_WALK_LARGE DTLB_WALK_PAGE DTLB_WALK_PERCYC DTLB_WALK_SMALL INST_RETIRED L1D_TLB L1D_TLB_REFILL L2D_TLB L2D_TLB_REFILL
L1 Instruction Cache Effectiveness (3)	L1 Data Cache Effectiveness (5)	L2D Data Unified Cache Effectiveness (5)
<ul style="list-style-type: none"> INST_RETIRED L1I_CACHE L1I_CACHE_REFILL 	<ul style="list-style-type: none"> INST_RETIRED L1D_CACHE L1D_CACHE_REFILL L1D_CACHE_REFILL_RD L1D_CACHE_REFILL_WR 	<ul style="list-style-type: none"> INST_RETIRED L2D_CACHE L2D_CACHE_REFILL L2D_CACHE_REFILL_RD L2D_CACHE_REFILL_WR
L2 Instruction Unified Cache Effectiveness (3)	L3 Unified Cache Effectiveness (3)	Last Level Cache Effectiveness (3)
<ul style="list-style-type: none"> INST_RETIRED L2I_CACHE L2I_CACHE_REFILL 	<ul style="list-style-type: none"> INST_RETIRED L3D_CACHE L3D_CACHE_REFILL 	<ul style="list-style-type: none"> INST_RETIRED LL_CACHE_MISS_RD LL_CACHE_RD
Register Rename Effectiveness (4)	Issue Queue Effectiveness (5)	Main Commit Queue Effectiveness (2)
<ul style="list-style-type: none"> IMP_STALL_BACKEND_RENAME_FRF IMP_STALL_BACKEND_RENAME_GRF IMP_STALL_BACKEND_RENAME_PDRF IMP_STALL_BACKEND_RENAME_VRF 	<ul style="list-style-type: none"> IMP_STALL_BACKEND_IQ_LS IMP_STALL_BACKEND_IQ_MX IMP_STALL_BACKEND_IQ_SX IMP_STALL_BACKEND_IQ_VX STALL_BACKEND_BUSY 	<ul style="list-style-type: none"> IMP_STALL_BACKEND_MCQ STALL_BACKEND_CPUBOUND

Execution Unit Effectiveness (7)	Prefetcher Effectiveness (13)	Atomics Effectiveness (7)
<ul style="list-style-type: none"> CPU_CYCLES IMP_OP_BRU_ISSUE IMP_OP_DPU_ISSUE IMP_OP_LSU_ISSUE IMP_OP_STD_ISSUE IMP_OP_VPU_ISSUE IMP_WFX_CLOCK_CYCLES 	<ul style="list-style-type: none"> IMP_L2_CACHE_PREFETCH_LATE IMP_L2_CACHE_PREFETCH_USEFUL L1D_CACHE_HIT_RW_FHWPRF L1D_CACHE_REFILL_HWPRF L1D_CACHE_REFILL_RD L1D_CACHE_REFILL_WR L1D_LFB_HIT_RW_FHWPRF L2D_CACHE_HIT_RW_FHWPRF L2D_CACHE_REFILL L2D_CACHE_REFILL_HWPRF L2D_CACHE_REFILL_RD L2D_CACHE_REFILL_WR L2D_LFB_HIT_RW_FHWPRF 	<ul style="list-style-type: none"> CAS_NEAR_PASS CAS_NEAR_SPEC CAS_SPEC LDST_SPEC LSE_LDST_SPEC LSE_LD_SPEC LSE_ST_SPEC
Average Latency (10)	Bus Effectiveness (5)	System Memory Effectiveness (7)
<ul style="list-style-type: none"> BUS_REQ_RD BUS_REQ_RD_PERCYC DTLB_WALK DTLB_WALK_PERCYC INST_FETCH INST_FETCH_PERCYC ITLB_WALK ITLB_WALK_PERCYC MEM_ACCESS MEM_ACCESS_RD_PERCYC 	<ul style="list-style-type: none"> BUS_ACCESS_RD BUS_ACCESS_WR BUS_REQ BUS_REQ_RD BUS_REQ_RD_PERCYC 	<ul style="list-style-type: none"> DSNP_HIT IMP_DRAM_ACCESS ISNP_HIT_RD L2D_CACHE_REFILL L2I_CACHE_REFILL L3D_CACHE_HIT LL_CACHE_HIT

4.3 Metrics lookup table for C1-Premium

All metrics are listed alphabetically, with the related events, and metric groups. Some metrics are used in more than one metric group, in that case they are listed multiple times so that you can jump to the most relevant metric group for your requirements.

Table 4-21: Metrics listed by name, with related events and metric groups

Metric Name	Formula from Events	Metric Groups
backend_bound	$\text{STALL_SLOT_BACKEND} / (10 * (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES})) * 100$	<ul style="list-style-type: none"> Topdown_L1
backend_busy_bound	$\text{STALL_BACKEND_BUSY} / \text{STALL_BACKEND} * 100$	<ul style="list-style-type: none"> Topdown_Backend
backend_cache_l1d_bound	$\text{STALL_BACKEND_L1D} / (\text{STALL_BACKEND_L1D} + \text{STALL_BACKEND_MEM}) * 100$	<ul style="list-style-type: none"> Topdown_Backend

Metric Name	Formula from Events	Metric Groups
backend_cache_l2d_bound	$\text{STALL_BACKEND_MEM} / (\text{STALL_BACKEND_L1D} + \text{STALL_BACKEND_MEM}) * 100$	<ul style="list-style-type: none"> Topdown_Backend
backend_cme_backpressure_bound	$\text{STALL_BACKEND_BUSY_CMEBOUND} / \text{STALL_BACKEND_BUSY_CME} * 100$	<ul style="list-style-type: none"> Topdown_CME
backend_cme_busy_arb_bound	$\text{STALL_BACKEND_BUSY_CME_ARB} / \text{STALL_BACKEND_BUSY_CME} * 100$	<ul style="list-style-type: none"> Topdown_CME
backend_cme_cpu_bound	$\text{STALL_BACKEND_BUSY_CME_CPUBOUND} / \text{STALL_BACKEND_BUSY_CME} * 100$	<ul style="list-style-type: none"> Topdown_CME
backend_core_bound	$\text{STALL_BACKEND_CPUBOUND} / \text{STALL_BACKEND} * 100$	<ul style="list-style-type: none"> Topdown_Backend
backend_core_cme_bound	$\text{STALL_BACKEND_BUSY_CME} / \text{STALL_BACKEND_CPUBOUND} * 100$	<ul style="list-style-type: none"> Topdown_Backend
backend_core_other_bound	$\text{IMP_STALL_BACKEND_CPUBOUND_OTHER} / \text{STALL_BACKEND_CPUBOUND} * 100$	<ul style="list-style-type: none"> Topdown_Backend
backend_core_rename_bound	$\text{STALL_BACKEND_RENAME} / \text{STALL_BACKEND_CPUBOUND} * 100$	<ul style="list-style-type: none"> Topdown_Backend
backend_core_spec_throttle_bound	$\text{IMP_STALL_BACKEND_SPEC_THROT} / \text{STALL_BACKEND_CPUBOUND} * 100$	<ul style="list-style-type: none"> Topdown_Backend
backend_mem_bound	$\text{STALL_BACKEND_MEMBOUND} / \text{STALL_BACKEND} * 100$	<ul style="list-style-type: none"> Topdown_Backend
backend_mem_cache_bound	$(\text{STALL_BACKEND_L1D} + \text{STALL_BACKEND_MEM}) / \text{STALL_BACKEND_MEMBOUND} * 100$	<ul style="list-style-type: none"> Topdown_Backend
backend_mem_cme_barrier_bound	$\text{STALL_BACKEND_MEM_CME_BARRIER} / \text{STALL_BACKEND_MEM_CME} * 100$	<ul style="list-style-type: none"> Topdown_Backend
backend_mem_cme_bound	$\text{STALL_BACKEND_MEM_CME} / \text{STALL_BACKEND_MEMBOUND} * 100$	<ul style="list-style-type: none"> Topdown_Backend
backend_mem_cme_hazard_cpu_bound	$\text{STALL_BACKEND_MEM_CME_HZ_ON_CPU} / \text{STALL_BACKEND_MEM_CME} * 100$	<ul style="list-style-type: none"> Topdown_Backend
backend_mem_cme_lsrt_full_bound	$\text{STALL_BACKEND_MEM_CME_LSRT_FULL} / \text{STALL_BACKEND_MEM_CME} * 100$	<ul style="list-style-type: none"> Topdown_Backend
backend_mem_cpu_hazard_cme_bound	$\text{STALL_BACKEND_MEM_CPU_HZ_ON_CME} / \text{STALL_BACKEND_MEM_CME} * 100$	<ul style="list-style-type: none"> Topdown_Backend
backend_mem_store_bound	$\text{STALL_BACKEND_ST} / \text{STALL_BACKEND_MEMBOUND} * 100$	<ul style="list-style-type: none"> Topdown_Backend
backend_mem_tlb_bound	$\text{STALL_BACKEND_TLB} / \text{STALL_BACKEND_MEMBOUND} * 100$	<ul style="list-style-type: none"> Topdown_Backend
backend_stalled_cycles	$\text{STALL_BACKEND} / (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES}) * 100$	<ul style="list-style-type: none"> Cycle_Accounting

Metric Name	Formula from Events	Metric Groups
bad_speculation	$(1 - (\text{STALL_SLOT} - 10 * \text{IMP_WFX_CLOCK_CYCLES}) / (10 * (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES}))) * (1 - \text{OP_RETIRED} / \text{OP_SPEC}) * 100 + \text{STALL_FRONTEND_FLUSH} / (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES}) * 100$	<ul style="list-style-type: none"> Topdown_L1
barrier_percentage	$(\text{ISB_SPEC} + \text{DSB_SPEC} + \text{DMB_SPEC}) / \text{INST_SPEC} * 100$	<ul style="list-style-type: none"> Operation_Mix
branch_direct_ratio	$\text{BR_IMMED_RETIRED} / \text{BR_RETIRED}$	<ul style="list-style-type: none"> Branch_Effectiveness
branch_indirect_ratio	$\text{BR_IND_RETIRED} / \text{BR_RETIRED}$	<ul style="list-style-type: none"> Branch_Effectiveness
<ul style="list-style-type: none"> branch_misprediction_ratio in Branch_Effectiveness branch_misprediction_ratio in Miss_Ratio 	$\text{BR_MIS_PRED_RETIRED} / \text{BR_RETIRED}$	<ul style="list-style-type: none"> Branch_Effectiveness Miss_Ratio
<ul style="list-style-type: none"> branch_mpki in Branch_Effectiveness branch_mpki in MPKI 	$\text{BR_MIS_PRED_RETIRED} / \text{INST_RETIRED} * 1000$	<ul style="list-style-type: none"> Branch_Effectiveness MPKI
branch_percentage	$\text{PC_WRITE_SPEC} / \text{INST_SPEC} * 100$	<ul style="list-style-type: none"> Operation_Mix
branch_port_utilization	$\text{IMP_OP_BRU_ISSUE} / (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES})$	<ul style="list-style-type: none"> Port_Utilization
branch_return_ratio	$\text{BR_RETURN_RETIRED} / \text{BR_RETIRED}$	<ul style="list-style-type: none"> Branch_Effectiveness
bus_access_average_count	$(\text{BUS_ACCESS_RD} + \text{BUS_ACCESS_WR}) / \text{BUS_REQ}$	<ul style="list-style-type: none"> Bus_Effectiveness
<ul style="list-style-type: none"> bus_read_requests_average_latency in Average_Latency bus_read_requests_average_latency in Bus_Effectiveness 	$\text{BUS_REQ_RD_PERCYC} / \text{BUS_REQ_RD}$	<ul style="list-style-type: none"> Average_Latency Bus_Effectiveness
cas_far_ratio	$1 - \text{CAS_NEAR_SPEC} / \text{CAS_SPEC}$	<ul style="list-style-type: none"> Atomics_Effectiveness
cas_near_fail_ratio	$1 - \text{CAS_NEAR_PASS} / \text{CAS_NEAR_SPEC}$	<ul style="list-style-type: none"> Atomics_Effectiveness
cas_near_pass_ratio	$\text{CAS_NEAR_PASS} / \text{CAS_NEAR_SPEC}$	<ul style="list-style-type: none"> Atomics_Effectiveness
cas_near_ratio	$\text{CAS_NEAR_SPEC} / \text{CAS_SPEC}$	<ul style="list-style-type: none"> Atomics_Effectiveness
cme_alloc_cycles_ratio	$\text{CYCLES_CME_ALLOC} / \text{CPU_CYCLES} * 100$	<ul style="list-style-type: none"> Cycle_Accounting
cme_arb_pending_ratio	$\text{CYCLES_ARB_PENDING_CME} / \text{CPU_CYCLES} * 100$	<ul style="list-style-type: none"> Cycle_Accounting
crypto_percentage	$\text{CRYPTO_SPEC} / \text{INST_SPEC} * 100$	<ul style="list-style-type: none"> Operation_Mix
<ul style="list-style-type: none"> dtlb_mpki in DTLB_Effectiveness dtlb_mpki in MPKI 	$\text{DTLB_WALK} / \text{INST_RETIRED} * 1000$	<ul style="list-style-type: none"> DTLB_Effectiveness MPKI
dtlb_walk_average_depth	$\text{DTLB_STEP} / \text{DTLB_WALK}$	<ul style="list-style-type: none"> DTLB_Effectiveness
<ul style="list-style-type: none"> dtlb_walk_average_latency in Average_Latency dtlb_walk_average_latency in DTLB_Effectiveness 	$\text{DTLB_WALK_PERCYC} / \text{DTLB_WALK}$	<ul style="list-style-type: none"> Average_Latency DTLB_Effectiveness
dtlb_walk_block_ratio	$\text{DTLB_WALK_BLOCK} / \text{L1D_TLB}$	<ul style="list-style-type: none"> DTLB_Effectiveness
dtlb_walk_large_ratio	$\text{DTLB_WALK_LARGE} / \text{L1D_TLB}$	<ul style="list-style-type: none"> DTLB_Effectiveness
dtlb_walk_page_ratio	$\text{DTLB_WALK_PAGE} / \text{L1D_TLB}$	<ul style="list-style-type: none"> DTLB_Effectiveness

Metric Name	Formula from Events	Metric Groups
<ul style="list-style-type: none"> dtlb_walk_ratio in DTLB_Effectiveness dtlb_walk_ratio in Miss_Ratio 	DTLB_WALK / L1D_TLB	<ul style="list-style-type: none"> DTLB_Effectiveness Miss_Ratio
dtlb_walk_small_ratio	DTLB_WALK_SMALL / L1D_TLB	<ul style="list-style-type: none"> DTLB_Effectiveness
fp16_percentage	FP_HP_SPEC / INST_SPEC * 100	<ul style="list-style-type: none"> FP_Precision_Mix
fp32_percentage	FP_SP_SPEC / INST_SPEC * 100	<ul style="list-style-type: none"> FP_Precision_Mix
fp64_percentage	FP_DP_SPEC / INST_SPEC * 100	<ul style="list-style-type: none"> FP_Precision_Mix
fp_ops_per_cycle	(FP_SCALE_OPS_SPEC + FP_FIXED_OPS_SPEC) / (CPU_CYCLES - IMP_WFX_CLOCK_CYCLES)	<ul style="list-style-type: none"> FP_Arithmetic_Intensity
frontend_bound	((STALL_SLOT_FRONTEND - 10 * IMP_WFX_CLOCK_CYCLES) / (10 * (CPU_CYCLES - IMP_WFX_CLOCK_CYCLES)) - (STALL_FRONTEND_FLUSH - IMP_WFX_CLOCK_CYCLES) / (CPU_CYCLES - IMP_WFX_CLOCK_CYCLES)) * 100	<ul style="list-style-type: none"> Topdown_L1
frontend_cache_l1i_bound	STALL_FRONTEND_L1I / (STALL_FRONTEND_L1I + STALL_FRONTEND_MEM) * 100	<ul style="list-style-type: none"> Topdown_Frontend
frontend_cache_l2i_bound	STALL_FRONTEND_MEM / (STALL_FRONTEND_L1I + STALL_FRONTEND_MEM) * 100	<ul style="list-style-type: none"> Topdown_Frontend
frontend_core_bound	(STALL_FRONTEND_CPUBOUND - IMP_WFX_CLOCK_CYCLES) / (STALL_FRONTEND - IMP_WFX_CLOCK_CYCLES) * 100	<ul style="list-style-type: none"> Topdown_Frontend
frontend_core_flow_bound	STALL_FRONTEND_FLOW / STALL_FRONTEND_CPUBOUND * 100	<ul style="list-style-type: none"> Topdown_Frontend
frontend_core_flush_bound	(STALL_FRONTEND_FLUSH - IMP_WFX_CLOCK_CYCLES) / (STALL_FRONTEND_CPUBOUND - IMP_WFX_CLOCK_CYCLES) * 100	<ul style="list-style-type: none"> Topdown_Frontend
frontend_core_flush_machine_clear_bound	(IMP_STALL_FRONTEND_FLUSH_CLEAR - IMP_WFX_CLOCK_CYCLES) / (STALL_FRONTEND_FLUSH - IMP_WFX_CLOCK_CYCLES) * 100	<ul style="list-style-type: none"> Topdown_Frontend
frontend_core_flush_resteer_bound	IMP_STALL_FRONTEND_FLUSH_RESTEER / STALL_FRONTEND_FLUSH * 100	<ul style="list-style-type: none"> Topdown_Frontend
frontend_core_spec_throttle_bound	IMP_STALL_FRONTEND_SPEC_THROT / STALL_FRONTEND_CPUBOUND * 100	<ul style="list-style-type: none"> Topdown_Frontend
frontend_mem_bound	STALL_FRONTEND_MEMBOUND / STALL_FRONTEND * 100	<ul style="list-style-type: none"> Topdown_Frontend
frontend_mem_cache_bound	(STALL_FRONTEND_L1I + STALL_FRONTEND_MEM) / STALL_FRONTEND_MEMBOUND * 100	<ul style="list-style-type: none"> Topdown_Frontend
frontend_mem_tlb_bound	STALL_FRONTEND_TLB / STALL_FRONTEND_MEMBOUND * 100	<ul style="list-style-type: none"> Topdown_Frontend

Metric Name	Formula from Events	Metric Groups
frontend_stalled_cycles	$(\text{STALL_FRONTEND} - \text{IMP_WFX_CLOCK_CYCLES}) / (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES}) * 100$	<ul style="list-style-type: none"> Cycle_Accounting
instruction_fetch_average_latency	$\text{INST_FETCH_PERCYC} / \text{INST_FETCH}$	<ul style="list-style-type: none"> Average_Latency
int_port_utilization	$\text{IMP_OP_DPU_ISSUE} / (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES})$	<ul style="list-style-type: none"> Port_Utilization
integer_dp_percentage	$\text{DP_SPEC} / \text{INST_SPEC} * 100$	<ul style="list-style-type: none"> Operation_Mix
ipc	$\text{INST_RETIRED} / (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES})$	<ul style="list-style-type: none"> General
iq_stall_lsu_percentage	$\text{IMP_STALL_BACKEND_IQ_LS} / \text{STALL_BACKEND_BUSY} * 100$	<ul style="list-style-type: none"> IQ_Effectiveness
iq_stall_mx_percentage	$\text{IMP_STALL_BACKEND_IQ_MX} / \text{STALL_BACKEND_BUSY} * 100$	<ul style="list-style-type: none"> IQ_Effectiveness
iq_stall_sx_percentage	$\text{IMP_STALL_BACKEND_IQ_SX} / \text{STALL_BACKEND_BUSY} * 100$	<ul style="list-style-type: none"> IQ_Effectiveness
iq_stall_vpu_percentage	$\text{IMP_STALL_BACKEND_IQ_VX} / \text{STALL_BACKEND_BUSY} * 100$	<ul style="list-style-type: none"> IQ_Effectiveness
<ul style="list-style-type: none"> itlb_mpki in ITLB_Effectiveness itlb_mpki in MPKI 	$\text{ITLB_WALK} / \text{INST_RETIRED} * 1000$	<ul style="list-style-type: none"> ITLB_Effectiveness MPKI
itlb_walk_average_depth	$\text{ITLB_STEP} / \text{ITLB_WALK}$	<ul style="list-style-type: none"> ITLB_Effectiveness
<ul style="list-style-type: none"> itlb_walk_average_latency in Average_Latency itlb_walk_average_latency in ITLB_Effectiveness 	$\text{ITLB_WALK_PERCYC} / \text{ITLB_WALK}$	<ul style="list-style-type: none"> Average_Latency ITLB_Effectiveness
itlb_walk_block_ratio	$\text{ITLB_WALK_BLOCK} / \text{L1I_TLB}$	<ul style="list-style-type: none"> ITLB_Effectiveness
itlb_walk_large_ratio	$\text{ITLB_WALK_LARGE} / \text{L1I_TLB}$	<ul style="list-style-type: none"> ITLB_Effectiveness
itlb_walk_page_ratio	$\text{ITLB_WALK_PAGE} / \text{L1I_TLB}$	<ul style="list-style-type: none"> ITLB_Effectiveness
<ul style="list-style-type: none"> itlb_walk_ratio in ITLB_Effectiveness itlb_walk_ratio in Miss_Ratio 	$\text{ITLB_WALK} / \text{L1I_TLB}$	<ul style="list-style-type: none"> ITLB_Effectiveness Miss_Ratio
itlb_walk_small_ratio	$\text{ITLB_WALK_SMALL} / \text{L1I_TLB}$	<ul style="list-style-type: none"> ITLB_Effectiveness
l1_prefetcher_accuracy	$(\text{L1D_CACHE_HIT_RW_FHWPRF} + \text{L1D_LFB_HIT_RW_FHWPRF}) / (\text{L1D_CACHE_REFILL_HWPFRF} + \text{L1D_LFB_HIT_RW_FHWPRF})$	<ul style="list-style-type: none"> Prefetcher_Effectiveness
l1_prefetcher_coverage	$(\text{L1D_CACHE_HIT_RW_FHWPRF} + \text{L1D_LFB_HIT_RW_FHWPRF}) / (\text{L1D_CACHE_HIT_RW_FHWPRF} + \text{L1D_CACHE_REFILL_RD} + \text{L1D_CACHE_REFILL_WR})$	<ul style="list-style-type: none"> Prefetcher_Effectiveness
l1_prefetcher_timeliness	$\text{L1D_CACHE_HIT_RW_FHWPRF} / (\text{L1D_CACHE_HIT_RW_FHWPRF} + \text{L1D_LFB_HIT_RW_FHWPRF})$	<ul style="list-style-type: none"> Prefetcher_Effectiveness
<ul style="list-style-type: none"> l1d_cache_demand_mpki in L1D_Cache_Effectiveness l1d_cache_demand_mpki in MPKI 	$(\text{L1D_CACHE_REFILL_RD} + \text{L1D_CACHE_REFILL_WR}) / \text{INST_RETIRED} * 1000$	<ul style="list-style-type: none"> L1D_Cache_Effectiveness MPKI

Metric Name	Formula from Events	Metric Groups
<ul style="list-style-type: none"> l1d_cache_miss_ratio in L1D_Cache_Effectiveness l1d_cache_miss_ratio in Miss_Ratio 	$L1D_CACHE_REFILL / L1D_CACHE$	<ul style="list-style-type: none"> L1D_Cache_Effectiveness Miss_Ratio
<ul style="list-style-type: none"> l1d_cache_mpki in L1D_Cache_Effectiveness l1d_cache_mpki in MPKI 	$L1D_CACHE_REFILL / INST_RETIRED * 1000$	<ul style="list-style-type: none"> L1D_Cache_Effectiveness MPKI
<ul style="list-style-type: none"> l1d_tlb_miss_ratio in DTLB_Effectiveness l1d_tlb_miss_ratio in Miss_Ratio 	$L1D_TLB_REFILL / L1D_TLB$	<ul style="list-style-type: none"> DTLB_Effectiveness Miss_Ratio
<ul style="list-style-type: none"> l1d_tlb_mpki in DTLB_Effectiveness l1d_tlb_mpki in MPKI 	$L1D_TLB_REFILL / INST_RETIRED * 1000$	<ul style="list-style-type: none"> DTLB_Effectiveness MPKI
<ul style="list-style-type: none"> l1i_cache_miss_ratio in L1I_Cache_Effectiveness l1i_cache_miss_ratio in Miss_Ratio 	$L1I_CACHE_REFILL / L1I_CACHE$	<ul style="list-style-type: none"> L1I_Cache_Effectiveness Miss_Ratio
<ul style="list-style-type: none"> l1i_cache_mpki in L1I_Cache_Effectiveness l1i_cache_mpki in MPKI 	$L1I_CACHE_REFILL / INST_RETIRED * 1000$	<ul style="list-style-type: none"> L1I_Cache_Effectiveness MPKI
<ul style="list-style-type: none"> l1i_tlb_miss_ratio in ITLB_Effectiveness l1i_tlb_miss_ratio in Miss_Ratio 	$L1I_TLB_REFILL / L1I_TLB$	<ul style="list-style-type: none"> ITLB_Effectiveness Miss_Ratio
<ul style="list-style-type: none"> l1i_tlb_mpki in ITLB_Effectiveness l1i_tlb_mpki in MPKI 	$L1I_TLB_REFILL / INST_RETIRED * 1000$	<ul style="list-style-type: none"> ITLB_Effectiveness MPKI
l2_prefetcher_accuracy_l1hwprf_exclusive	$(L2D_CACHE_HIT_RW_FWWPRF + L2D_LFB_HIT_RW_FWWPRF) / (L2D_CACHE_REFILL_HWPRF - IMP_L2_CACHE_PREFETCH_USEFUL - IMP_L2_CACHE_PREFETCH_LATE + L2D_LFB_HIT_RW_FWWPRF + L2D_CACHE_REFILL_HWPRF)$	<ul style="list-style-type: none"> Prefetcher_Effectiveness
l2_prefetcher_accuracy_l1hwprf_inclusive	$(IMP_L2_CACHE_PREFETCH_USEFUL + IMP_L2_CACHE_PREFETCH_LATE) / L2D_CACHE_REFILL_HWPRF$	<ul style="list-style-type: none"> Prefetcher_Effectiveness
l2_prefetcher_coverage_l1hwprf_exclusive	$(L2D_CACHE_HIT_RW_FWWPRF + L2D_LFB_HIT_RW_FWWPRF) / (L2D_CACHE_REFILL_RD + L2D_CACHE_REFILL_WR)$	<ul style="list-style-type: none"> Prefetcher_Effectiveness
l2_prefetcher_coverage_l1hwprf_inclusive	$(IMP_L2_CACHE_PREFETCH_USEFUL + IMP_L2_CACHE_PREFETCH_LATE) / (IMP_L2_CACHE_PREFETCH_USEFUL + IMP_L2_CACHE_PREFETCH_LATE + L2D_CACHE_REFILL - L2D_CACHE_REFILL_HWPRF)$	<ul style="list-style-type: none"> Prefetcher_Effectiveness
l2_prefetcher_timeliness_l1hwprf_exclusive	$L2D_CACHE_HIT_RW_FWWPRF / (L2D_CACHE_HIT_RW_FWWPRF + L2D_LFB_HIT_RW_FWWPRF)$	<ul style="list-style-type: none"> Prefetcher_Effectiveness
l2_prefetcher_timeliness_l1hwprf_inclusive	$IMP_L2_CACHE_PREFETCH_USEFUL / (IMP_L2_CACHE_PREFETCH_USEFUL + IMP_L2_CACHE_PREFETCH_LATE)$	<ul style="list-style-type: none"> Prefetcher_Effectiveness

Metric Name	Formula from Events	Metric Groups
<ul style="list-style-type: none"> l2_tlb_miss_ratio in DTLB_Effectiveness l2_tlb_miss_ratio in ITLB_Effectiveness l2_tlb_miss_ratio in Miss_Ratio 	$L2D_TLB_REFILL / L2D_TLB$	<ul style="list-style-type: none"> DTLB_Effectiveness ITLB_Effectiveness Miss_Ratio
<ul style="list-style-type: none"> l2_tlb_mpki in DTLB_Effectiveness l2_tlb_mpki in ITLB_Effectiveness l2_tlb_mpki in MPKI 	$L2D_TLB_REFILL / INST_RETIRED * 1000$	<ul style="list-style-type: none"> DTLB_Effectiveness ITLB_Effectiveness MPKI
<ul style="list-style-type: none"> l2d_cache_demand_mpki in L2D_Cache_Effectiveness l2d_cache_demand_mpki in MPKI 	$(L2D_CACHE_REFILL_RD + L2D_CACHE_REFILL_WR) / INST_RETIRED * 1000$	<ul style="list-style-type: none"> L2D_Cache_Effectiveness MPKI
<ul style="list-style-type: none"> l2d_cache_miss_ratio in L2D_Cache_Effectiveness l2d_cache_miss_ratio in Miss_Ratio 	$L2D_CACHE_REFILL / L2D_CACHE$	<ul style="list-style-type: none"> L2D_Cache_Effectiveness Miss_Ratio
<ul style="list-style-type: none"> l2d_cache_mpki in L2D_Cache_Effectiveness l2d_cache_mpki in MPKI 	$L2D_CACHE_REFILL / INST_RETIRED * 1000$	<ul style="list-style-type: none"> L2D_Cache_Effectiveness MPKI
<ul style="list-style-type: none"> l2i_cache_miss_ratio in L2I_Cache_Effectiveness l2i_cache_miss_ratio in Miss_Ratio 	$L2I_CACHE_REFILL / L2I_CACHE$	<ul style="list-style-type: none"> L2I_Cache_Effectiveness Miss_Ratio
<ul style="list-style-type: none"> l2i_cache_mpki in L2I_Cache_Effectiveness l2i_cache_mpki in MPKI 	$L2I_CACHE_REFILL / INST_RETIRED * 1000$	<ul style="list-style-type: none"> L2I_Cache_Effectiveness MPKI
<ul style="list-style-type: none"> l3_cache_miss_ratio in L3_Cache_Effectiveness l3_cache_miss_ratio in Miss_Ratio 	$L3D_CACHE_REFILL / L3D_CACHE$	<ul style="list-style-type: none"> L3_Cache_Effectiveness Miss_Ratio
<ul style="list-style-type: none"> l3_cache_mpki in L3_Cache_Effectiveness l3_cache_mpki in MPKI 	$L3D_CACHE_REFILL / INST_RETIRED * 1000$	<ul style="list-style-type: none"> L3_Cache_Effectiveness MPKI
ll_cache_read_hit_ratio	$(LL_CACHE_RD - LL_CACHE_MISS_RD) / LL_CACHE_RD$	<ul style="list-style-type: none"> LL_Cache_Effectiveness
<ul style="list-style-type: none"> ll_cache_read_miss_ratio in LL_Cache_Effectiveness ll_cache_read_miss_ratio in Miss_Ratio 	$LL_CACHE_MISS_RD / LL_CACHE_RD$	<ul style="list-style-type: none"> LL_Cache_Effectiveness Miss_Ratio
<ul style="list-style-type: none"> ll_cache_read_mpki in LL_Cache_Effectiveness ll_cache_read_mpki in MPKI 	$LL_CACHE_MISS_RD / INST_RETIRED * 1000$	<ul style="list-style-type: none"> LL_Cache_Effectiveness MPKI
load_average_latency	$MEM_ACCESS_RD_PERCYC / MEM_ACCESS$	<ul style="list-style-type: none"> Average_Latency
load_ls_percentage	$LD_SPEC / LDST_SPEC * 100$	<ul style="list-style-type: none"> Operation_Mix
load_percentage	$LD_SPEC / INST_SPEC * 100$	<ul style="list-style-type: none"> Operation_Mix
load_store_percentage	$LDST_SPEC / INST_SPEC * 100$	<ul style="list-style-type: none"> Operation_Mix
lse_atomics_ratio	$LSE_LDST_SPEC / LDST_SPEC$	<ul style="list-style-type: none"> Atomics_Effectiveness
lse_load_ratio	$LSE_LD_SPEC / LSE_LDST_SPEC$	<ul style="list-style-type: none"> Atomics_Effectiveness
lse_store_ratio	$LSE_ST_SPEC / LSE_LDST_SPEC$	<ul style="list-style-type: none"> Atomics_Effectiveness

Metric Name	Formula from Events	Metric Groups
lsu_port_utilization	$\text{IMP_OP_LSU_ISSUE} / (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES})$	<ul style="list-style-type: none"> Port_Utilization
mcq_stall_percentage	$\text{IMP_STALL_BACKEND_MCQ} / \text{STALL_BACKEND_CPUBOUND} * 100$	<ul style="list-style-type: none"> MCQ_Effectiveness
nonsve_fp_ops_per_cycle	$\text{FP_FIXED_OPS_SPEC} / (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES})$	<ul style="list-style-type: none"> FP_Arithmetic_Intensity
rename_stall_flags_ratio	$\text{IMP_STALL_BACKEND_RENAME_FRF} / (\text{IMP_STALL_BACKEND_RENAME_VRF} + \text{IMP_STALL_BACKEND_RENAME_FRF} + \text{IMP_STALL_BACKEND_RENAME_GRF} + \text{IMP_STALL_BACKEND_RENAME_PDRF})$	<ul style="list-style-type: none"> Rename_Effectiveness
rename_stall_int_ratio	$\text{IMP_STALL_BACKEND_RENAME_GRF} / (\text{IMP_STALL_BACKEND_RENAME_VRF} + \text{IMP_STALL_BACKEND_RENAME_FRF} + \text{IMP_STALL_BACKEND_RENAME_GRF} + \text{IMP_STALL_BACKEND_RENAME_PDRF})$	<ul style="list-style-type: none"> Rename_Effectiveness
rename_stall_pred_ratio	$\text{IMP_STALL_BACKEND_RENAME_PDRF} / (\text{IMP_STALL_BACKEND_RENAME_VRF} + \text{IMP_STALL_BACKEND_RENAME_FRF} + \text{IMP_STALL_BACKEND_RENAME_GRF} + \text{IMP_STALL_BACKEND_RENAME_PDRF})$	<ul style="list-style-type: none"> Rename_Effectiveness
rename_stall_vec_ratio	$\text{IMP_STALL_BACKEND_RENAME_VRF} / (\text{IMP_STALL_BACKEND_RENAME_VRF} + \text{IMP_STALL_BACKEND_RENAME_FRF} + \text{IMP_STALL_BACKEND_RENAME_GRF} + \text{IMP_STALL_BACKEND_RENAME_PDRF})$	<ul style="list-style-type: none"> Rename_Effectiveness
retiring	$(1 - (\text{STALL_SLOT} - 10 * \text{IMP_WFX_CLOCK_CYCLES}) / ((\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES}) * 10)) * (\text{OP_RETIRED} / \text{OP_SPEC}) * 100$	<ul style="list-style-type: none"> Topdown_L1
scalar_fp_percentage	$\text{VFP_SPEC} / \text{INST_SPEC} * 100$	<ul style="list-style-type: none"> Operation_Mix
simd_percentage	$\text{ASE_SPEC} / \text{INST_SPEC} * 100$	<ul style="list-style-type: none"> Operation_Mix
sm_active_cycles_ratio	$\text{SM_ACTIVE_CYCLES} / \text{CPU_CYCLES} * 100$	<ul style="list-style-type: none"> Cycle_Accounting
sme_percentage	$\text{SME_INST_SPEC} / \text{INST_SPEC} * 100$	<ul style="list-style-type: none"> Operation_Mix
std_port_utilization	$\text{IMP_OP_STD_ISSUE} / (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES})$	<ul style="list-style-type: none"> Port_Utilization
store_ls_percentage	$\text{ST_SPEC} / \text{LDST_SPEC} * 100$	<ul style="list-style-type: none"> Operation_Mix
store_percentage	$\text{ST_SPEC} / \text{INST_SPEC} * 100$	<ul style="list-style-type: none"> Operation_Mix
sve_fp_ops_per_cycle	$\text{FP_SCALE_OPS_SPEC} / (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES})$	<ul style="list-style-type: none"> FP_Arithmetic_Intensity
sve_percentage	$\text{SVE_SPEC} / \text{INST_SPEC} * 100$	<ul style="list-style-type: none"> Operation_Mix
sve_predicate_empty_percentage	$\text{SVE_PRED_EMPTY_SPEC} / \text{SVE_PRED_SPEC} * 100$	<ul style="list-style-type: none"> SVE_Effectiveness
sve_predicate_full_percentage	$\text{SVE_PRED_FULL_SPEC} / \text{SVE_PRED_SPEC} * 100$	<ul style="list-style-type: none"> SVE_Effectiveness
sve_predicate_partial_percentage	$\text{SVE_PRED_PARTIAL_SPEC} / \text{SVE_PRED_SPEC} * 100$	<ul style="list-style-type: none"> SVE_Effectiveness
sve_predicate_percentage	$\text{SVE_PRED_SPEC} / \text{INST_SPEC} * 100$	<ul style="list-style-type: none"> SVE_Effectiveness

Metric Name	Formula from Events	Metric Groups
system_dram_mem_hit_ratio	$\text{IMP_DRAM_ACCESS} / (\text{L2D_CACHE_REFILL} + \text{L2I_CACHE_REFILL})$	<ul style="list-style-type: none"> System_Memory_Effectiveness
system_l3_cache_hit_ratio	$\text{L3D_CACHE_HIT} / (\text{L2D_CACHE_REFILL} + \text{L2I_CACHE_REFILL})$	<ul style="list-style-type: none"> System_Memory_Effectiveness
system_llc_cache_hit_ratio	$\text{LL_CACHE_HIT} / (\text{L2D_CACHE_REFILL} + \text{L2I_CACHE_REFILL})$	<ul style="list-style-type: none"> System_Memory_Effectiveness
system_peer_cluster_cache_hit_ratio	$(\text{DSNP_HIT} + \text{ISNP_HIT_RD}) / (\text{L2D_CACHE_REFILL} + \text{L2I_CACHE_REFILL})$	<ul style="list-style-type: none"> System_Memory_Effectiveness
vpu_port_utilization	$\text{IMP_OP_VPU_ISSUE} / (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES})$	<ul style="list-style-type: none"> Port_Utilization
za_active_cycles_ratio	$\text{ZA_ACTIVE} / \text{CPU_CYCLES} * 100$	<ul style="list-style-type: none"> Cycle_Accounting

4.4 PMU events lookup table for C1-Premium

All events are listed in event code order, with the related metrics, metric groups, and functional groups. Some events are not used in the Methodology, however, they are all listed for completeness.

Summary of Events:

- Total Possible Common events: 825
- Total implemented Common events: 311
 - Common : Architectural-defined events: 245
 - Common : Implementation-defined events: 66
- Total Implemented Product ImpDef events: 61
- PMU Only Events : 61
- ETE Only Events : 3

Table 4-22: Events listed by Event Code, with related Metrics, Metric Groups, and Functional Groups

Code, Mnemonic	Metrics	Metric Groups	Functional Groups
0x0000, SW_INCR	-	-	<ul style="list-style-type: none"> Retired
0x0001, L1I_CACHE_REFILL	<ul style="list-style-type: none"> l1i_cache_mpki in L1I_Cache_Effectiveness l1i_cache_mpki in MPKI l1i_cache_miss_ratio in L1I_Cache_Effectiveness l1i_cache_miss_ratio in Miss_Ratio 	<ul style="list-style-type: none"> L1I_Cache_Effectiveness MPKI Miss_Ratio 	<ul style="list-style-type: none"> L1I_Cache

Code, Mnemonic	Metrics	Metric Groups	Functional Groups
0x0002, L1I_TLB_REFILL	<ul style="list-style-type: none"> l1i_tlb_mpki in ITLB_Effectiveness l1i_tlb_mpki in MPKI l1i_tlb_miss_ratio in ITLB_Effectiveness l1i_tlb_miss_ratio in Miss_Ratio 	<ul style="list-style-type: none"> ITLB_Effectiveness MPKI Miss_Ratio 	<ul style="list-style-type: none"> TLB
0x0003, L1D_CACHE_REFILL	<ul style="list-style-type: none"> l1d_cache_mpki in L1D_Cache_Effectiveness l1d_cache_mpki in MPKI l1d_cache_miss_ratio in L1D_Cache_Effectiveness l1d_cache_miss_ratio in Miss_Ratio 	<ul style="list-style-type: none"> L1D_Cache_Effectiveness MPKI Miss_Ratio 	<ul style="list-style-type: none"> L1D_Cache
0x0004, L1D_CACHE	<ul style="list-style-type: none"> l1d_cache_miss_ratio in L1D_Cache_Effectiveness l1d_cache_miss_ratio in Miss_Ratio 	<ul style="list-style-type: none"> L1D_Cache_Effectiveness Miss_Ratio 	<ul style="list-style-type: none"> L1D_Cache
0x0005, L1D_TLB_REFILL	<ul style="list-style-type: none"> l1d_tlb_mpki in DTLB_Effectiveness l1d_tlb_mpki in MPKI l1d_tlb_miss_ratio in DTLB_Effectiveness l1d_tlb_miss_ratio in Miss_Ratio 	<ul style="list-style-type: none"> DTLB_Effectiveness MPKI Miss_Ratio 	<ul style="list-style-type: none"> TLB

Code, Mnemonic	Metrics	Metric Groups	Functional Groups
0x0008, INST_RETIRED	<ul style="list-style-type: none"> ipc branch_mpki in Branch_Effectiveness branch_mpki in MPKI itlb_mpki in ITLB_Effectiveness itlb_mpki in MPKI l1i_tlb_mpki in ITLB_Effectiveness l1i_tlb_mpki in MPKI dtlb_mpki in DTLB_Effectiveness dtlb_mpki in MPKI l1d_tlb_mpki in DTLB_Effectiveness l1d_tlb_mpki in MPKI l2_tlb_mpki in DTLB_Effectiveness l2_tlb_mpki in ITLB_Effectiveness l2_tlb_mpki in MPKI l1i_cache_mpki in L1I_Cache_Effectiveness l1i_cache_mpki in MPKI l1d_cache_demand_mpki in L1D_Cache_Effectiveness l1d_cache_demand_mpki in MPKI l1d_cache_mpki in L1D_Cache_Effectiveness l1d_cache_mpki in MPKI l2i_cache_mpki in L2I_Cache_Effectiveness l2i_cache_mpki in MPKI l2d_cache_demand_mpki in L2D_Cache_Effectiveness l2d_cache_demand_mpki in MPKI l2d_cache_mpki in L2D_Cache_Effectiveness l2d_cache_mpki in MPKI l3_cache_mpki in L3_Cache_Effectiveness l3_cache_mpki in MPKI ll_cache_read_mpki in LL_Cache_Effectiveness ll_cache_read_mpki in MPKI 	<ul style="list-style-type: none"> Branch_Effectiveness DTLB_Effectiveness General ITLB_Effectiveness L1D_Cache_Effectiveness L1I_Cache_Effectiveness L2D_Cache_Effectiveness L2I_Cache_Effectiveness L3_Cache_Effectiveness LL_Cache_Effectiveness MPKI 	<ul style="list-style-type: none"> Retired

Code, Mnemonic	Metrics	Metric Groups	Functional Groups
0x0009, EXC_TAKEN	-	-	<ul style="list-style-type: none"> Exception
0x000A, EXC_RETURN	-	-	<ul style="list-style-type: none"> Exception
0x000B, CID_WRITE_RETIRED	-	-	<ul style="list-style-type: none"> Retired
0x000C, PC_WRITE_RETIRED	-	-	<ul style="list-style-type: none"> Retired
0x000D, BR_IMMED_RETIRED	<ul style="list-style-type: none"> branch_direct_ratio 	<ul style="list-style-type: none"> Branch_Effectiveness 	<ul style="list-style-type: none"> Retired
0x000E, BR_RETURN_RETIRED	<ul style="list-style-type: none"> branch_return_ratio 	<ul style="list-style-type: none"> Branch_Effectiveness 	<ul style="list-style-type: none"> Retired
0x0010, BR_MIS_PRED	-	-	<ul style="list-style-type: none"> Spec_Operation
0x0011, CPU_CYCLES	<ul style="list-style-type: none"> frontend_stalled_cycles backend_stalled_cycles frontend_bound backend_bound retiring bad_speculation ipc sve_fp_ops_per_cycle nonsve_fp_ops_per_cycle fp_ops_per_cycle branch_port_utilization int_port_utilization vpu_port_utilization lsu_port_utilization std_port_utilization sm_active_cycles_ratio za_active_cycles_ratio cme_alloc_cycles_ratio cme_arb_pending_ratio 	<ul style="list-style-type: none"> Cycle_Accounting FP_Arithmetic_Intensity General Port_Utilization Topdown_L1 	<ul style="list-style-type: none"> General
0x0012, BR_PRED	-	-	<ul style="list-style-type: none"> Spec_Operation
0x0013, MEM_ACCESS	<ul style="list-style-type: none"> load_average_latency 	<ul style="list-style-type: none"> Average_Latency 	<ul style="list-style-type: none"> Memory
0x0014, L1I_CACHE	<ul style="list-style-type: none"> l1i_cache_miss_ratio in L1I_Cache_Effectiveness l1i_cache_miss_ratio in Miss_Ratio 	<ul style="list-style-type: none"> L1I_Cache_Effectiveness Miss_Ratio 	<ul style="list-style-type: none"> L1I_Cache
0x0015, L1D_CACHE_WB	-	-	<ul style="list-style-type: none"> L1D_Cache
0x0016, L2D_CACHE	<ul style="list-style-type: none"> l2d_cache_miss_ratio in L2D_Cache_Effectiveness l2d_cache_miss_ratio in Miss_Ratio 	<ul style="list-style-type: none"> L2D_Cache_Effectiveness Miss_Ratio 	<ul style="list-style-type: none"> L2_Cache

Code, Mnemonic	Metrics	Metric Groups	Functional Groups
0x0017, L2D_CACHE_REFILL	<ul style="list-style-type: none"> l2d_cache_mпки in L2D_Cache_Effectiveness l2d_cache_mпки in MPKI l2d_cache_miss_ratio in L2D_Cache_Effectiveness l2d_cache_miss_ratio in Miss_Ratio l2_prefetcher_coverage_l1hwprf_inclusive system_l3_cache_hit_ratio system_peer_cluster_cache_hit_ratio system_llc_cache_hit_ratio system_dram_mem_hit_ratio 	<ul style="list-style-type: none"> L2D_Cache_Effectiveness MPKI Miss_Ratio Prefetcher_Effectiveness System_Memory_Effectiveness 	<ul style="list-style-type: none"> L2_Cache
0x0018, L2D_CACHE_WB	-	-	<ul style="list-style-type: none"> L2_Cache
0x0019, BUS_ACCESS	-	-	<ul style="list-style-type: none"> Bus
0x001A, MEMORY_ERROR	-	-	<ul style="list-style-type: none"> Memory
0x001B, INST_SPEC	<ul style="list-style-type: none"> load_store_percentage load_percentage store_percentage integer_dp_percentage simd_percentage scalar_fp_percentage barrier_percentage branch_percentage crypto_percentage sve_percentage sve_predicate_percentage fp16_percentage fp32_percentage fp64_percentage sme_percentage 	<ul style="list-style-type: none"> FP_Precision_Mix Operation_Mix SVE_Effectiveness 	<ul style="list-style-type: none"> Spec_Operation
0x001C, TTBR_WRITE_RETIRED	-	-	<ul style="list-style-type: none"> Retired
0x001D, BUS_CYCLES	-	-	<ul style="list-style-type: none"> Bus
0x001E, CHAIN	-	-	<ul style="list-style-type: none"> Chain
0x0020, L2D_CACHE_ALLOCATE	-	-	<ul style="list-style-type: none"> L2_Cache

Code, Mnemonic	Metrics	Metric Groups	Functional Groups
0x0021, BR_RETIRED	<ul style="list-style-type: none"> branch_direct_ratio branch_indirect_ratio branch_return_ratio branch_misprediction_ratio in Branch_Effectiveness branch_misprediction_ratio in Miss_Ratio 	<ul style="list-style-type: none"> Branch_Effectiveness Miss_Ratio 	<ul style="list-style-type: none"> Retired
0x0022, BR_MIS_PRED_RETIRED	<ul style="list-style-type: none"> branch_mpki in Branch_Effectiveness branch_mpki in MPKI branch_misprediction_ratio in Branch_Effectiveness branch_misprediction_ratio in Miss_Ratio 	<ul style="list-style-type: none"> Branch_Effectiveness MPKI Miss_Ratio 	<ul style="list-style-type: none"> Retired
0x0023, STALL_FRONTEND	<ul style="list-style-type: none"> frontend_stalled_cycles frontend_core_bound frontend_mem_bound 	<ul style="list-style-type: none"> Cycle_Accounting Topdown_Frontend 	<ul style="list-style-type: none"> Stall
0x0024, STALL_BACKEND	<ul style="list-style-type: none"> backend_stalled_cycles backend_core_bound backend_mem_bound backend_busy_bound 	<ul style="list-style-type: none"> Cycle_Accounting Topdown_Backend 	<ul style="list-style-type: none"> Stall
0x0025, L1D_TLB	<ul style="list-style-type: none"> dtlb_walk_ratio in DTLB_Effectiveness dtlb_walk_ratio in Miss_Ratio dtlb_walk_large_ratio dtlb_walk_small_ratio dtlb_walk_page_ratio dtlb_walk_block_ratio l1d_tlb_miss_ratio in DTLB_Effectiveness l1d_tlb_miss_ratio in Miss_Ratio 	<ul style="list-style-type: none"> DTLB_Effectiveness Miss_Ratio 	<ul style="list-style-type: none"> TLB
0x0026, L1I_TLB	<ul style="list-style-type: none"> itlb_walk_ratio in ITLB_Effectiveness itlb_walk_ratio in Miss_Ratio itlb_walk_large_ratio itlb_walk_small_ratio itlb_walk_page_ratio itlb_walk_block_ratio l1i_tlb_miss_ratio in ITLB_Effectiveness l1i_tlb_miss_ratio in Miss_Ratio 	<ul style="list-style-type: none"> ITLB_Effectiveness Miss_Ratio 	<ul style="list-style-type: none"> TLB

Code, Mnemonic	Metrics	Metric Groups	Functional Groups
0x0027, L2I_CACHE	<ul style="list-style-type: none"> l2i_cache_miss_ratio in L2I_Cache_Effectiveness l2i_cache_miss_ratio in Miss_Ratio 	<ul style="list-style-type: none"> L2I_Cache_Effectiveness Miss_Ratio 	<ul style="list-style-type: none"> L2_Cache
0x0028, L2I_CACHE_REFILL	<ul style="list-style-type: none"> l2i_cache_mpki in L2I_Cache_Effectiveness l2i_cache_mpki in MPKI l2i_cache_miss_ratio in L2I_Cache_Effectiveness l2i_cache_miss_ratio in Miss_Ratio system_l3_cache_hit_ratio system_peer_cluster_cache_hit_ratio system_llc_cache_hit_ratio system_dram_mem_hit_ratio 	<ul style="list-style-type: none"> L2I_Cache_Effectiveness MPKI Miss_Ratio System_Memory_Effectiveness 	<ul style="list-style-type: none"> L2_Cache
0x0029, L3D_CACHE_ALLOCATE	-	-	<ul style="list-style-type: none"> L3_Cache
0x002A, L3D_CACHE_REFILL	<ul style="list-style-type: none"> l3_cache_mpki in L3_Cache_Effectiveness l3_cache_mpki in MPKI l3_cache_miss_ratio in L3_Cache_Effectiveness l3_cache_miss_ratio in Miss_Ratio 	<ul style="list-style-type: none"> L3_Cache_Effectiveness MPKI Miss_Ratio 	<ul style="list-style-type: none"> L3_Cache
0x002B, L3D_CACHE	<ul style="list-style-type: none"> l3_cache_miss_ratio in L3_Cache_Effectiveness l3_cache_miss_ratio in Miss_Ratio 	<ul style="list-style-type: none"> L3_Cache_Effectiveness Miss_Ratio 	<ul style="list-style-type: none"> L3_Cache
0x002D, L2D_TLB_REFILL	<ul style="list-style-type: none"> l2_tlb_mpki in DTLB_Effectiveness l2_tlb_mpki in ITLB_Effectiveness l2_tlb_mpki in MPKI l2_tlb_miss_ratio in DTLB_Effectiveness l2_tlb_miss_ratio in ITLB_Effectiveness l2_tlb_miss_ratio in Miss_Ratio 	<ul style="list-style-type: none"> DTLB_Effectiveness ITLB_Effectiveness MPKI Miss_Ratio 	<ul style="list-style-type: none"> TLB
0x002F, L2D_TLB	<ul style="list-style-type: none"> l2_tlb_miss_ratio in DTLB_Effectiveness l2_tlb_miss_ratio in ITLB_Effectiveness l2_tlb_miss_ratio in Miss_Ratio 	<ul style="list-style-type: none"> DTLB_Effectiveness ITLB_Effectiveness Miss_Ratio 	<ul style="list-style-type: none"> TLB
0x0031, REMOTE_ACCESS	-	-	<ul style="list-style-type: none"> Memory

Code, Mnemonic	Metrics	Metric Groups	Functional Groups
0x0032, LL_CACHE	-	-	<ul style="list-style-type: none"> LL_Cache
0x0034, DTLB_WALK	<ul style="list-style-type: none"> dtlb_mpki in DTLB_Effectiveness dtlb_mpki in MPKI dtlb_walk_ratio in DTLB_Effectiveness dtlb_walk_ratio in Miss_Ratio dtlb_walk_average_depth dtlb_walk_average_latency in Average_Latency dtlb_walk_average_latency in DTLB_Effectiveness 	<ul style="list-style-type: none"> Average_Latency DTLB_Effectiveness MPKI Miss_Ratio 	<ul style="list-style-type: none"> TLB
0x0035, ITLB_WALK	<ul style="list-style-type: none"> itlb_mpki in ITLB_Effectiveness itlb_mpki in MPKI itlb_walk_ratio in ITLB_Effectiveness itlb_walk_ratio in Miss_Ratio itlb_walk_average_depth itlb_walk_average_latency in Average_Latency itlb_walk_average_latency in ITLB_Effectiveness 	<ul style="list-style-type: none"> Average_Latency ITLB_Effectiveness MPKI Miss_Ratio 	<ul style="list-style-type: none"> TLB
0x0036, LL_CACHE_RD	<ul style="list-style-type: none"> ll_cache_read_miss_ratio in LL_Cache_Effectiveness ll_cache_read_miss_ratio in Miss_Ratio ll_cache_read_hit_ratio 	<ul style="list-style-type: none"> LL_Cache_Effectiveness Miss_Ratio 	<ul style="list-style-type: none"> LL_Cache
0x0037, LL_CACHE_MISS_RD	<ul style="list-style-type: none"> ll_cache_read_mpki in LL_Cache_Effectiveness ll_cache_read_mpki in MPKI ll_cache_read_miss_ratio in LL_Cache_Effectiveness ll_cache_read_miss_ratio in Miss_Ratio ll_cache_read_hit_ratio 	<ul style="list-style-type: none"> LL_Cache_Effectiveness MPKI Miss_Ratio 	<ul style="list-style-type: none"> LL_Cache
0x0039, L1D_CACHE_LMISS_RD	-	-	<ul style="list-style-type: none"> L1D_Cache
0x003A, OP_RETIRED	<ul style="list-style-type: none"> retiring bad_speculation 	<ul style="list-style-type: none"> Topdown_L1 	<ul style="list-style-type: none"> Retired
0x003B, OP_SPEC	<ul style="list-style-type: none"> retiring bad_speculation 	<ul style="list-style-type: none"> Topdown_L1 	<ul style="list-style-type: none"> Spec_Operation
0x003C, STALL	-	-	<ul style="list-style-type: none"> Stall
0x003D, STALL_SLOT_BACKEND	<ul style="list-style-type: none"> backend_bound 	<ul style="list-style-type: none"> Topdown_L1 	<ul style="list-style-type: none"> Stall

Code, Mnemonic	Metrics	Metric Groups	Functional Groups
0x003E, STALL_SLOT_FRONTEND	<ul style="list-style-type: none"> frontend_bound 	<ul style="list-style-type: none"> Topdown_L1 	<ul style="list-style-type: none"> Stall
0x003F, STALL_SLOT	<ul style="list-style-type: none"> retiring bad_speculation 	<ul style="list-style-type: none"> Topdown_L1 	<ul style="list-style-type: none"> Stall
0x0040, L1D_CACHE_RD	-	-	<ul style="list-style-type: none"> L1D_Cache
0x0041, L1D_CACHE_WR	-	-	<ul style="list-style-type: none"> L1D_Cache
0x0042, L1D_CACHE_REFILL_RD	<ul style="list-style-type: none"> l1d_cache_demand_mпки in L1D_Cache_Effectiveness l1d_cache_demand_mпки in MPKI l1_prefetcher_coverage 	<ul style="list-style-type: none"> L1D_Cache_Effectiveness MPKI Prefetcher_Effectiveness 	<ul style="list-style-type: none"> L1D_Cache
0x0043, L1D_CACHE_REFILL_WR	<ul style="list-style-type: none"> l1d_cache_demand_mпки in L1D_Cache_Effectiveness l1d_cache_demand_mпки in MPKI l1_prefetcher_coverage 	<ul style="list-style-type: none"> L1D_Cache_Effectiveness MPKI Prefetcher_Effectiveness 	<ul style="list-style-type: none"> L1D_Cache
0x0044, L1D_CACHE_REFILL_INNER	-	-	<ul style="list-style-type: none"> L1D_Cache
0x0045, L1D_CACHE_REFILL_OUTER	-	-	<ul style="list-style-type: none"> L1D_Cache
0x0046, L1D_CACHE_WB_VICTIM	-	-	<ul style="list-style-type: none"> L1D_Cache
0x0047, L1D_CACHE_WB_CLEAN	-	-	<ul style="list-style-type: none"> L1D_Cache
0x0048, L1D_CACHE_INVALID	-	-	<ul style="list-style-type: none"> L1D_Cache
0x004C, L1D_TLB_REFILL_RD	-	-	<ul style="list-style-type: none"> TLB
0x004D, L1D_TLB_REFILL_WR	-	-	<ul style="list-style-type: none"> TLB
0x004E, L1D_TLB_RD	-	-	<ul style="list-style-type: none"> TLB
0x004F, L1D_TLB_WR	-	-	<ul style="list-style-type: none"> TLB
0x0050, L2D_CACHE_RD	-	-	<ul style="list-style-type: none"> L2_Cache
0x0051, L2D_CACHE_WR	-	-	<ul style="list-style-type: none"> L2_Cache
0x0052, L2D_CACHE_REFILL_RD	<ul style="list-style-type: none"> l2d_cache_demand_mпки in L2D_Cache_Effectiveness l2d_cache_demand_mпки in MPKI l2_prefetcher_coverage_l1hwrpf_exclusive 	<ul style="list-style-type: none"> L2D_Cache_Effectiveness MPKI Prefetcher_Effectiveness 	<ul style="list-style-type: none"> L2_Cache
0x0053, L2D_CACHE_REFILL_WR	<ul style="list-style-type: none"> l2d_cache_demand_mпки in L2D_Cache_Effectiveness l2d_cache_demand_mпки in MPKI l2_prefetcher_coverage_l1hwrpf_exclusive 	<ul style="list-style-type: none"> L2D_Cache_Effectiveness MPKI Prefetcher_Effectiveness 	<ul style="list-style-type: none"> L2_Cache
0x0056, L2D_CACHE_WB_VICTIM	-	-	<ul style="list-style-type: none"> L2_Cache

Code, Mnemonic	Metrics	Metric Groups	Functional Groups
0x0057, L2D_CACHE_WB_CLEAN	-	-	<ul style="list-style-type: none"> L2_Cache
0x0058, L2D_CACHE_INVALID	-	-	<ul style="list-style-type: none"> L2_Cache
0x005C, L2D_TLB_REFILL_RD	-	-	<ul style="list-style-type: none"> TLB
0x005D, L2D_TLB_REFILL_WR	-	-	<ul style="list-style-type: none"> TLB
0x005E, L2D_TLB_RD	-	-	<ul style="list-style-type: none"> TLB
0x005F, L2D_TLB_WR	-	-	<ul style="list-style-type: none"> TLB
0x0060, BUS_ACCESS_RD	<ul style="list-style-type: none"> bus_access_average_count 	<ul style="list-style-type: none"> Bus_Effectiveness 	<ul style="list-style-type: none"> Bus
0x0061, BUS_ACCESS_WR	<ul style="list-style-type: none"> bus_access_average_count 	<ul style="list-style-type: none"> Bus_Effectiveness 	<ul style="list-style-type: none"> Bus
0x0066, MEM_ACCESS_RD	-	-	<ul style="list-style-type: none"> Memory
0x0067, MEM_ACCESS_WR	-	-	<ul style="list-style-type: none"> Memory
0x0068, UNALIGNED_LD_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x0069, UNALIGNED_ST_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x006A, UNALIGNED_LDST_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x006C, LDREX_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x006D, STREX_PASS_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x006E, STREX_FAIL_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x006F, STREX_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x0070, LD_SPEC	<ul style="list-style-type: none"> load_ls_percentage load_percentage 	<ul style="list-style-type: none"> Operation_Mix 	<ul style="list-style-type: none"> Spec_Operation
0x0071, ST_SPEC	<ul style="list-style-type: none"> store_ls_percentage store_percentage 	<ul style="list-style-type: none"> Operation_Mix 	<ul style="list-style-type: none"> Spec_Operation
0x0072, LDST_SPEC	<ul style="list-style-type: none"> load_store_percentage load_ls_percentage store_ls_percentage lse_atomics_ratio 	<ul style="list-style-type: none"> Atomics_Effectiveness Operation_Mix 	<ul style="list-style-type: none"> Spec_Operation
0x0073, DP_SPEC	<ul style="list-style-type: none"> integer_dp_percentage 	<ul style="list-style-type: none"> Operation_Mix 	<ul style="list-style-type: none"> Spec_Operation
0x0074, ASE_SPEC	<ul style="list-style-type: none"> simd_percentage 	<ul style="list-style-type: none"> Operation_Mix 	<ul style="list-style-type: none"> Spec_Operation
0x0075, VFP_SPEC	<ul style="list-style-type: none"> scalar_fp_percentage 	<ul style="list-style-type: none"> Operation_Mix 	<ul style="list-style-type: none"> Spec_Operation
0x0076, PC_WRITE_SPEC	<ul style="list-style-type: none"> branch_percentage 	<ul style="list-style-type: none"> Operation_Mix 	<ul style="list-style-type: none"> Spec_Operation
0x0077, CRYPTO_SPEC	<ul style="list-style-type: none"> crypto_percentage 	<ul style="list-style-type: none"> Operation_Mix 	<ul style="list-style-type: none"> Spec_Operation
0x0078, BR_IMMED_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x0079, BR_RETURN_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x007A, BR_INDIRECT_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x007C, ISB_SPEC	<ul style="list-style-type: none"> barrier_percentage 	<ul style="list-style-type: none"> Operation_Mix 	<ul style="list-style-type: none"> Spec_Operation
0x007D, DSB_SPEC	<ul style="list-style-type: none"> barrier_percentage 	<ul style="list-style-type: none"> Operation_Mix 	<ul style="list-style-type: none"> Spec_Operation
0x007E, DMB_SPEC	<ul style="list-style-type: none"> barrier_percentage 	<ul style="list-style-type: none"> Operation_Mix 	<ul style="list-style-type: none"> Spec_Operation
0x007F, CSDB_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x0081, EXC_UNDEF	-	-	<ul style="list-style-type: none"> Exception
0x0082, EXC_SVC	-	-	<ul style="list-style-type: none"> Exception

Code, Mnemonic	Metrics	Metric Groups	Functional Groups
0x0083, EXC_PABORT	-	-	• Exception
0x0084, EXC_DABORT	-	-	• Exception
0x0086, EXC_IRQ	-	-	• Exception
0x0087, EXC_FIQ	-	-	• Exception
0x0088, EXC_SMC	-	-	• Exception
0x008A, EXC_HVC	-	-	• Exception
0x008B, EXC_TRAP_PABORT	-	-	• Exception
0x008C, EXC_TRAP_DABORT	-	-	• Exception
0x008D, EXC_TRAP_OTHER	-	-	• Exception
0x008E, EXC_TRAP_IRQ	-	-	• Exception
0x008F, EXC_TRAP_FIQ	-	-	• Exception
0x0090, RC_LD_SPEC	-	-	• Spec_Operation
0x0091, RC_ST_SPEC	-	-	• Spec_Operation
0x00A0, L3D_CACHE_RD	-	-	• L3_Cache
0x010B, IMP_L2_CACHE_PREFETCH_LATE	<ul style="list-style-type: none"> l2_prefetcher_coverage_l1hwprf_inclusive l2_prefetcher_accuracy_l1hwprf_inclusive l2_prefetcher_timeliness_l1hwprf_inclusive l2_prefetcher_accuracy_l1hwprf_exclusive 	• Prefetcher_Effectiveness	• L2_Cache
0x010F, IMP_MMU_L2_TLB_REFILL_PREFETCH_READ	-	-	• TLB
0x0120, IMP_CT_FLUSH	-	-	• CPU_Debug
0x0121, IMP_CT_FLUSH_MEM_HAZARD	-	-	• CPU_Debug
0x0122, IMP_CT_FLUSH_BAD_BRANCH	-	-	• CPU_Debug
0x0124, IMP_CT_FLUSH_ISB	-	-	• CPU_Debug
0x0125, IMP_CT_FLUSH_OTHER	-	-	• CPU_Debug
0x0127, IMP_LS_RAR_HAZARD	-	-	• CPU_Debug
0x0128, IMP_LS_RAW_HAZARD	-	-	• CPU_Debug
0x0158, IMP_STALL_BACKEND_RENAME_FRF	<ul style="list-style-type: none"> rename_stall_vec_ratio rename_stall_flags_ratio rename_stall_int_ratio rename_stall_pred_ratio 	• Rename_Effectiveness	• Stall

Code, Mnemonic	Metrics	Metric Groups	Functional Groups
0x0159, IMP_STALL_BACKEND_RENAME_GRF	<ul style="list-style-type: none"> rename_stall_vec_ratio rename_stall_flags_ratio rename_stall_int_ratio rename_stall_pred_ratio 	<ul style="list-style-type: none"> Rename_Effectiveness 	<ul style="list-style-type: none"> Stall
0x015A, IMP_STALL_BACKEND_RENAME_VRF	<ul style="list-style-type: none"> rename_stall_vec_ratio rename_stall_flags_ratio rename_stall_int_ratio rename_stall_pred_ratio 	<ul style="list-style-type: none"> Rename_Effectiveness 	<ul style="list-style-type: none"> Stall
0x015C, IMP_STALL_BACKEND_IQ_SX	<ul style="list-style-type: none"> iq_stall_sx_percentage 	<ul style="list-style-type: none"> IQ_Effectiveness 	<ul style="list-style-type: none"> Stall
0x015D, IMP_STALL_BACKEND_IQ_MX	<ul style="list-style-type: none"> iq_stall_mx_percentage 	<ul style="list-style-type: none"> IQ_Effectiveness 	<ul style="list-style-type: none"> Stall
0x015E, IMP_STALL_BACKEND_IQ_LS	<ul style="list-style-type: none"> iq_stall_lsu_percentage 	<ul style="list-style-type: none"> IQ_Effectiveness 	<ul style="list-style-type: none"> Stall
0x015F, IMP_STALL_BACKEND_IQ_VX	<ul style="list-style-type: none"> iq_stall_vpu_percentage 	<ul style="list-style-type: none"> IQ_Effectiveness 	<ul style="list-style-type: none"> Stall
0x0160, IMP_STALL_BACKEND_MCQ	<ul style="list-style-type: none"> mcq_stall_percentage 	<ul style="list-style-type: none"> MCQ_Effectiveness 	<ul style="list-style-type: none"> Stall
0x0179, IMP_L2_CACHE_PREFETCH_USEFUL	<ul style="list-style-type: none"> l2_prefetcher_coverage_l1hwprf_inclusive l2_prefetcher_accuracy_l1hwprf_inclusive l2_prefetcher_timeliness_l1hwprf_inclusive l2_prefetcher_accuracy_l1hwprf_exclusive 	<ul style="list-style-type: none"> Prefetcher_Effectiveness 	<ul style="list-style-type: none"> L2_Cache
0x0198, IMP_L2_CHI_RX_CBUSY_0	-	-	<ul style="list-style-type: none"> General
0x0199, IMP_L2_CHI_RX_CBUSY_1	-	-	<ul style="list-style-type: none"> General
0x019A, IMP_L2_CHI_RX_CBUSY_2	-	-	<ul style="list-style-type: none"> General
0x019B, IMP_L2_CHI_RX_CBUSY_3	-	-	<ul style="list-style-type: none"> General
0x019C, IMP_L2_CHI_RX_CBUSY_MT	-	-	<ul style="list-style-type: none"> General
0x01B8, IMP_L2D_CACHE_L1HWPRF	-	-	<ul style="list-style-type: none"> L2_Cache
0x01B9, IMP_L2D_CACHE_REFILL_L1HWPRF	-	-	<ul style="list-style-type: none"> L2_Cache

Code, Mnemonic	Metrics	Metric Groups	Functional Groups
0x0225, IMP_WFX_CLOCK_CYCLES	<ul style="list-style-type: none"> frontend_stalled_cycles backend_stalled_cycles frontend_bound backend_bound retiring bad_speculation frontend_core_bound frontend_core_flush_machine_clear_bound frontend_core_flush_bound ipc sve_fp_ops_per_cycle nonsve_fp_ops_per_cycle fp_ops_per_cycle branch_port_utilization int_port_utilization vpu_port_utilization lsu_port_utilization std_port_utilization 	<ul style="list-style-type: none"> Cycle_Accounting FP_Arithmetic_Intensity General Port_Utilization Topdown_Frontend Topdown_L1 	<ul style="list-style-type: none"> General
0x0399, IMP_STALL_BACKEND_RENAME_PDRF	<ul style="list-style-type: none"> rename_stall_vec_ratio rename_stall_flags_ratio rename_stall_int_ratio rename_stall_pred_ratio 	<ul style="list-style-type: none"> Rename_Effectiveness 	<ul style="list-style-type: none"> Stall
0x1338, IMP_STALL_BACKEND_CPUBOUND_OTHER	<ul style="list-style-type: none"> backend_core_other_bound 	<ul style="list-style-type: none"> Topdown_Backend 	<ul style="list-style-type: none"> Stall
0x3000, IMP_OP_BRU_ISSUE	<ul style="list-style-type: none"> branch_port_utilization 	<ul style="list-style-type: none"> Port_Utilization 	<ul style="list-style-type: none"> General
0x3001, IMP_OP_DPU_ISSUE	<ul style="list-style-type: none"> int_port_utilization 	<ul style="list-style-type: none"> Port_Utilization 	<ul style="list-style-type: none"> General
0x3002, IMP_OP_VPU_ISSUE	<ul style="list-style-type: none"> vpu_port_utilization 	<ul style="list-style-type: none"> Port_Utilization 	<ul style="list-style-type: none"> General
0x3003, IMP_OP_LSU_ISSUE	<ul style="list-style-type: none"> lsu_port_utilization 	<ul style="list-style-type: none"> Port_Utilization 	<ul style="list-style-type: none"> General
0x3004, IMP_OP_STD_ISSUE	<ul style="list-style-type: none"> std_port_utilization 	<ul style="list-style-type: none"> Port_Utilization 	<ul style="list-style-type: none"> General
0x3005, IMP_STALL_FRONTEND_SPEC_THROT	<ul style="list-style-type: none"> frontend_core_spec_throttle_bound 	<ul style="list-style-type: none"> Topdown_Frontend 	<ul style="list-style-type: none"> Stall
0x3006, IMP_STALL_FRONTEND_FLUSH_CLEAR	<ul style="list-style-type: none"> frontend_core_flush_machine_clear_bound 	<ul style="list-style-type: none"> Topdown_Frontend 	<ul style="list-style-type: none"> Stall
0x3007, IMP_STALL_FRONTEND_FLUSH_RESTEER	<ul style="list-style-type: none"> frontend_core_flush_resteer_bound 	<ul style="list-style-type: none"> Topdown_Frontend 	<ul style="list-style-type: none"> Stall
0x3008, IMP_DRAM_ACCESS	<ul style="list-style-type: none"> system_dram_mem_hit_ratio 	<ul style="list-style-type: none"> System_Memory_Effectiveness 	<ul style="list-style-type: none"> Memory
0x3009, IMP_STALL_BACKEND_SPEC_THROT	<ul style="list-style-type: none"> backend_core_spec_throttle_bound 	<ul style="list-style-type: none"> Topdown_Backend 	<ul style="list-style-type: none"> Stall

Code, Mnemonic	Metrics	Metric Groups	Functional Groups
0x3200, STALL_BACKEND_BUSY_CME	<ul style="list-style-type: none"> backend_core_cme_bound backend_cme_cpu_bound backend_cme_backpressure_bound backend_cme_busy_arb_bound 	<ul style="list-style-type: none"> Topdown_Backend Topdown_CME 	<ul style="list-style-type: none"> Stall
0x3201, STALL_BACKEND_BUSY_CMEBOUND	<ul style="list-style-type: none"> backend_cme_backpressure_bound 	<ul style="list-style-type: none"> Topdown_CME 	<ul style="list-style-type: none"> Stall
0x3202, STALL_BACKEND_BUSY_CME_ARB	<ul style="list-style-type: none"> backend_cme_busy_arb_bound 	<ul style="list-style-type: none"> Topdown_CME 	<ul style="list-style-type: none"> Stall
0x3203, STALL_BACKEND_BUSY_CME_CPUBOUND	<ul style="list-style-type: none"> backend_cme_cpu_bound 	<ul style="list-style-type: none"> Topdown_CME 	<ul style="list-style-type: none"> Stall
0x320c, STALL_BACKEND_MEM_CME_LSRT_FULL	<ul style="list-style-type: none"> backend_mem_cme_lsrt_full_bound 	<ul style="list-style-type: none"> Topdown_Backend 	<ul style="list-style-type: none"> Stall
0x320d, STALL_BACKEND_MEM_CME_BARRIER	<ul style="list-style-type: none"> backend_mem_cme_barrier_bound 	<ul style="list-style-type: none"> Topdown_Backend 	<ul style="list-style-type: none"> Stall
0x320e, STALL_BACKEND_MEM_CME_HZ_ON_CPU	<ul style="list-style-type: none"> backend_mem_cme_hazard_cpu_bound 	<ul style="list-style-type: none"> Topdown_Backend 	<ul style="list-style-type: none"> Stall
0x320f, STALL_BACKEND_MEM_CPU_HZ_ON_CME	<ul style="list-style-type: none"> backend_mem_cpu_hazard_cme_bound 	<ul style="list-style-type: none"> Topdown_Backend 	<ul style="list-style-type: none"> Stall
0x3210, STALL_BACKEND_MEM_CME	<ul style="list-style-type: none"> backend_mem_cme_bound backend_mem_cme_hazard_cpu_bound backend_mem_cpu_hazard_cme_bound backend_mem_cme_lsrt_full_bound backend_mem_cme_barrier_bound 	<ul style="list-style-type: none"> Topdown_Backend 	<ul style="list-style-type: none"> Stall
0x3212, SM_ACTIVE_CYCLES	<ul style="list-style-type: none"> sm_active_cycles_ratio 	<ul style="list-style-type: none"> Cycle_Accounting 	<ul style="list-style-type: none"> General
0x3213, CYCLES_CME_ALLOC	<ul style="list-style-type: none"> cme_alloc_cycles_ratio 	<ul style="list-style-type: none"> Cycle_Accounting 	<ul style="list-style-type: none"> General
0x3214, CYCLES_ARB_PENDING_CME	<ul style="list-style-type: none"> cme_arb_pending_ratio 	<ul style="list-style-type: none"> Cycle_Accounting 	<ul style="list-style-type: none"> General
0x3215, CYCLES_CME_RECONNECT_PENDING	-	-	<ul style="list-style-type: none"> General
0x3216, ARB_CME_COUNT	-	-	<ul style="list-style-type: none"> General
0x3217, RECONNECT_CME_COUNT	-	-	<ul style="list-style-type: none"> General
0x3218, OP_CME_ISSUE	-	-	<ul style="list-style-type: none"> Spec_Operation
0x3219, SSVE_INST_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation

Code, Mnemonic	Metrics	Metric Groups	Functional Groups
0x321a, SSVE_SPEC	-	-	• Spec_Operation
0x3234, SSVE_PRED_SPEC	-	-	• SVE
0x3235, SSVE_PRED_EMPTY_SPEC	-	-	• SVE
0x3236, SSVE_PRED_FULL_SPEC	-	-	• SVE
0x3237, SSVE_PRED_NOT_FULL_SPEC	-	-	• SVE
0x3238, SSVE_PRED_PARTIAL_SPEC	-	-	• SVE
0x4000, SAMPLE_POP	-	-	• SPE
0x4001, SAMPLE_FEED	-	-	• SPE
0x4002, SAMPLE_FILTRATE	-	-	• SPE
0x4003, SAMPLE_COLLISION	-	-	• SPE
0x4004, CNT_CYCLES	-	-	• General
0x4005, STALL_BACKEND_MEM	<ul style="list-style-type: none"> • backend_mem_cache_bound • backend_cache_l1d_bound • backend_cache_l2d_bound 	• Topdown_Backend	• Stall
0x4006, L1I_CACHE_LMISS	-	-	• L1_Cache
0x4009, L2D_CACHE_LMISS_RD	-	-	• L2_Cache
0x400A, L2I_CACHE_LMISS	-	-	• L2_Cache
0x400B, L3D_CACHE_LMISS_RD	-	-	• L3_Cache
0x400C, TRB_WRAP	-	-	• TRACE
0x400D, PMU_OVFS	-	-	• Non_PMU
0x400E, TRB_TRIG	-	-	• TRACE
0x400F, PMU_HOVFS	-	-	• Non_PMU
0x4010, TRCEXTOUT0	-	-	• TRACE
0x4011, TRCEXTOUT1	-	-	• TRACE
0x4012, TRCEXTOUT2	-	-	• TRACE
0x4013, TRCEXTOUT3	-	-	• TRACE
0x4018, CTI_TRIGOUT4	-	-	• TRACE
0x4019, CTI_TRIGOUT5	-	-	• TRACE
0x401A, CTI_TRIGOUT6	-	-	• TRACE
0x401B, CTI_TRIGOUT7	-	-	• TRACE
0x4020, LDST_ALIGN_LAT	-	-	• Memory
0x4021, LD_ALIGN_LAT	-	-	• Memory
0x4022, ST_ALIGN_LAT	-	-	• Memory
0x4024, MEM_ACCESS_CHECKED	-	-	• Memory
0x4025, MEM_ACCESS_CHECKED_RD	-	-	• Memory

Code, Mnemonic	Metrics	Metric Groups	Functional Groups
0x4026, MEM_ACCESS_CHECKED_WR	-	-	<ul style="list-style-type: none"> Memory
0x8004, SIMD_INST_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x8005, ASE_INST_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x8006, SVE_INST_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation SVE
0x8014, FP_HP_SPEC	<ul style="list-style-type: none"> fp16_percentage 	<ul style="list-style-type: none"> FP_Precision_Mix 	<ul style="list-style-type: none"> FP_Operation
0x8018, FP_SP_SPEC	<ul style="list-style-type: none"> fp32_percentage 	<ul style="list-style-type: none"> FP_Precision_Mix 	<ul style="list-style-type: none"> FP_Operation
0x801C, FP_DP_SPEC	<ul style="list-style-type: none"> fp64_percentage 	<ul style="list-style-type: none"> FP_Precision_Mix 	<ul style="list-style-type: none"> FP_Operation
0x8040, INT_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x8056, SVE_SPEC	<ul style="list-style-type: none"> sve_percentage 	<ul style="list-style-type: none"> Operation_Mix 	<ul style="list-style-type: none"> Spec_Operation
0x8057, ASE_SVE_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x8074, SVE_PRED_SPEC	<ul style="list-style-type: none"> sve_predicate_percentage sve_predicate_full_percentage sve_predicate_partial_percentage sve_predicate_empty_percentage 	<ul style="list-style-type: none"> SVE_Effectiveness 	<ul style="list-style-type: none"> SVE
0x8075, SVE_PRED_EMPTY_SPEC	<ul style="list-style-type: none"> sve_predicate_empty_percentage 	<ul style="list-style-type: none"> SVE_Effectiveness 	<ul style="list-style-type: none"> SVE
0x8076, SVE_PRED_FULL_SPEC	<ul style="list-style-type: none"> sve_predicate_full_percentage 	<ul style="list-style-type: none"> SVE_Effectiveness 	<ul style="list-style-type: none"> SVE
0x8077, SVE_PRED_PARTIAL_SPEC	<ul style="list-style-type: none"> sve_predicate_partial_percentage 	<ul style="list-style-type: none"> SVE_Effectiveness 	<ul style="list-style-type: none"> SVE
0x8078, SVE_UNPRED_SPEC	-	-	<ul style="list-style-type: none"> SVE
0x8079, SVE_PRED_NOT_FULL_SPEC	-	-	<ul style="list-style-type: none"> SVE
0x8080, SVE_LDST_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x8081, SVE_LD_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x8082, SVE_ST_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x8083, SVE_PRF_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x8087, PRF_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x80BC, SVE_LDFF_SPEC	-	-	<ul style="list-style-type: none"> SVE
0x80BD, SVE_LDFF_FAULT_SPEC	-	-	<ul style="list-style-type: none"> SVE
0x80C0, FP_SCALE_OPS_SPEC	<ul style="list-style-type: none"> sve_fp_ops_per_cycle fp_ops_per_cycle 	<ul style="list-style-type: none"> FP_Arithmetic_Intensity 	<ul style="list-style-type: none"> FP_Operation
0x80C1, FP_FIXED_OPS_SPEC	<ul style="list-style-type: none"> nonsve_fp_ops_per_cycle fp_ops_per_cycle 	<ul style="list-style-type: none"> FP_Arithmetic_Intensity 	<ul style="list-style-type: none"> FP_Operation
0x80E3, ASE_SVE_INT8_SPEC	-	-	<ul style="list-style-type: none"> SVE
0x80E7, ASE_SVE_INT16_SPEC	-	-	<ul style="list-style-type: none"> SVE
0x80EB, ASE_SVE_INT32_SPEC	-	-	<ul style="list-style-type: none"> SVE

Code, Mnemonic	Metrics	Metric Groups	Functional Groups
0x80EF, ASE_SVE_INT64_SPEC	-	-	• SVE
0x8108, BR_IMMED_TAKEN_RETIRED	-	-	• Retired
0x810C, BR_INDNR_TAKEN_RETIRED	-	-	• Retired
0x8110, BR_IMMED_PRED_RETIRED	-	-	• Retired
0x8111, BR_IMMED_MIS_PRED_RETIRED	-	-	• Retired
0x8112, BR_IND_PRED_RETIRED	-	-	• Retired
0x8113, BR_IND_MIS_PRED_RETIRED	-	-	• Retired
0x8114, BR_RETURN_PRED_RETIRED	-	-	• Retired
0x8115, BR_RETURN_MIS_PRED_RETIRED	-	-	• Retired
0x8116, BR_INDNR_PRED_RETIRED	-	-	• Retired
0x8117, BR_INDNR_MIS_PRED_RETIRED	-	-	• Retired
0x8118, BR_TAKEN_PRED_RETIRED	-	-	• Retired
0x8119, BR_TAKEN_MIS_PRED_RETIRED	-	-	• Retired
0x811A, BR_SKIP_PRED_RETIRED	-	-	• Retired
0x811B, BR_SKIP_MIS_PRED_RETIRED	-	-	• Retired
0x811C, BR_PRED_RETIRED	-	-	• Retired
0x811D, BR_IND_RETIRED	• branch_indirect_ratio	• Branch_Effectiveness	• Retired
0x8120, INST_FETCH_PERCYC	• instruction_fetch_average_latency	• Average_Latency	• Memory
0x8121, MEM_ACCESS_RD_PERCYC	• load_average_latency	• Average_Latency	• Memory
0x8124, INST_FETCH	• instruction_fetch_average_latency	• Average_Latency	• Memory
0x8125, BUS_REQ_RD_PERCYC	• bus_read_requests_average_latency in Average_Latency • bus_read_requests_average_latency in Bus_Effectiveness	• Average_Latency • Bus_Effectiveness	• Bus
0x8128, DTLB_WALK_PERCYC	• dtlb_walk_average_latency in Average_Latency • dtlb_walk_average_latency in DTLB_Effectiveness	• Average_Latency • DTLB_Effectiveness	• TLB

Code, Mnemonic	Metrics	Metric Groups	Functional Groups
0x8129, ITLB_WALK_PERCYC	<ul style="list-style-type: none"> itlb_walk_average_latency in Average_Latency itlb_walk_average_latency in ITLB_Effectiveness 	<ul style="list-style-type: none"> Average_Latency ITLB_Effectiveness 	<ul style="list-style-type: none"> TLB
0x812A, SAMPLE_FEED_BR	-	-	<ul style="list-style-type: none"> SPE
0x812B, SAMPLE_FEED_LD	-	-	<ul style="list-style-type: none"> SPE
0x812C, SAMPLE_FEED_ST	-	-	<ul style="list-style-type: none"> SPE
0x812D, SAMPLE_FEED_OP	-	-	<ul style="list-style-type: none"> SPE
0x812E, SAMPLE_FEED_EVENT	-	-	<ul style="list-style-type: none"> SPE
0x812F, SAMPLE_FEED_LAT	-	-	<ul style="list-style-type: none"> SPE
0x8130, L1D_TLB_RW	-	-	<ul style="list-style-type: none"> TLB
0x8131, L1I_TLB_RD	-	-	<ul style="list-style-type: none"> TLB
0x8132, L1D_TLB_PRFM	-	-	<ul style="list-style-type: none"> TLB
0x8133, L1I_TLB_PRFM	-	-	<ul style="list-style-type: none"> TLB
0x8134, DTLB_HWUPD	-	-	<ul style="list-style-type: none"> TLB
0x8135, ITLB_HWUPD	-	-	<ul style="list-style-type: none"> TLB
0x8136, DTLB_STEP	<ul style="list-style-type: none"> dtlb_walk_average_depth 	<ul style="list-style-type: none"> DTLB_Effectiveness 	<ul style="list-style-type: none"> TLB
0x8137, ITLB_STEP	<ul style="list-style-type: none"> itlb_walk_average_depth 	<ul style="list-style-type: none"> ITLB_Effectiveness 	<ul style="list-style-type: none"> TLB
0x8138, DTLB_WALK_LARGE	<ul style="list-style-type: none"> dtlb_walk_large_ratio 	<ul style="list-style-type: none"> DTLB_Effectiveness 	<ul style="list-style-type: none"> TLB
0x8139, ITLB_WALK_LARGE	<ul style="list-style-type: none"> itlb_walk_large_ratio 	<ul style="list-style-type: none"> ITLB_Effectiveness 	<ul style="list-style-type: none"> TLB
0x813A, DTLB_WALK_SMALL	<ul style="list-style-type: none"> dtlb_walk_small_ratio 	<ul style="list-style-type: none"> DTLB_Effectiveness 	<ul style="list-style-type: none"> TLB
0x813B, ITLB_WALK_SMALL	<ul style="list-style-type: none"> itlb_walk_small_ratio 	<ul style="list-style-type: none"> ITLB_Effectiveness 	<ul style="list-style-type: none"> TLB
0x813C, DTLB_WALK_RW	-	-	<ul style="list-style-type: none"> TLB
0x813D, ITLB_WALK_RD	-	-	<ul style="list-style-type: none"> TLB
0x813E, DTLB_WALK_PRFM	-	-	<ul style="list-style-type: none"> TLB
0x813F, ITLB_WALK_PRFM	-	-	<ul style="list-style-type: none"> TLB
0x8140, L1D_CACHE_RW	-	-	<ul style="list-style-type: none"> L1D_Cache
0x8141, L1I_CACHE_RD	-	-	<ul style="list-style-type: none"> L1I_Cache
0x8142, L1D_CACHE_PRFM	-	-	<ul style="list-style-type: none"> L1D_Cache
0x8143, L1I_CACHE_PRFM	-	-	<ul style="list-style-type: none"> L1I_Cache
0x8144, L1D_CACHE_MISS	-	-	<ul style="list-style-type: none"> L1D_Cache
0x8145, L1I_CACHE_HWPRF	-	-	<ul style="list-style-type: none"> L1I_Cache
0x8146, L1D_CACHE_REFILL_PRFM	-	-	<ul style="list-style-type: none"> L1D_Cache
0x8147, L1I_CACHE_REFILL_PRFM	-	-	<ul style="list-style-type: none"> L1I_Cache
0x8148, L2D_CACHE_RW	-	-	<ul style="list-style-type: none"> L2_Cache
0x8149, L2I_CACHE_RD	-	-	<ul style="list-style-type: none"> L2_Cache
0x814A, L2D_CACHE_PRFM	-	-	<ul style="list-style-type: none"> L2_Cache
0x814B, L2I_CACHE_PRFM	-	-	<ul style="list-style-type: none"> L2_Cache
0x814C, L2D_CACHE_MISS	-	-	<ul style="list-style-type: none"> L2_Cache

Code, Mnemonic	Metrics	Metric Groups	Functional Groups
0x814E, L2D_CACHE_REFILL_PRFM	-	-	<ul style="list-style-type: none"> L2_Cache
0x814F, L2I_CACHE_REFILL_PRFM	-	-	<ul style="list-style-type: none"> L2_Cache
0x8150, L3D_CACHE_RW	-	-	<ul style="list-style-type: none"> L3_Cache
0x8151, L3D_CACHE_PRFM	-	-	<ul style="list-style-type: none"> L3_Cache
0x8152, L3D_CACHE_MISS	-	-	<ul style="list-style-type: none"> L3_Cache
0x8153, L3D_CACHE_REFILL_PRFM	-	-	<ul style="list-style-type: none"> L3_Cache
0x8154, L1D_CACHE_HWPRF	-	-	<ul style="list-style-type: none"> L1D_Cache
0x8155, L2D_CACHE_HWPRF	-	-	<ul style="list-style-type: none"> L2_Cache
0x8158, STALL_FRONTEND_MEMBOUND	<ul style="list-style-type: none"> frontend_mem_bound frontend_mem_cache_bound frontend_mem_tlb_bound 	<ul style="list-style-type: none"> Topdown_Frontend 	<ul style="list-style-type: none"> Stall
0x8159, STALL_FRONTEND_L1I	<ul style="list-style-type: none"> frontend_mem_cache_bound frontend_cache_l1i_bound frontend_cache_l2i_bound 	<ul style="list-style-type: none"> Topdown_Frontend 	<ul style="list-style-type: none"> Stall
0x815B, STALL_FRONTEND_MEM	<ul style="list-style-type: none"> frontend_mem_cache_bound frontend_cache_l1i_bound frontend_cache_l2i_bound 	<ul style="list-style-type: none"> Topdown_Frontend 	<ul style="list-style-type: none"> Stall
0x815C, STALL_FRONTEND_TLB	<ul style="list-style-type: none"> frontend_mem_tlb_bound 	<ul style="list-style-type: none"> Topdown_Frontend 	<ul style="list-style-type: none"> Stall
0x8160, STALL_FRONTEND_CPUBOUND	<ul style="list-style-type: none"> frontend_core_bound frontend_core_spec_throttle_bound frontend_core_flush_bound frontend_core_flow_bound 	<ul style="list-style-type: none"> Topdown_Frontend 	<ul style="list-style-type: none"> Stall
0x8161, STALL_FRONTEND_FLOW	<ul style="list-style-type: none"> frontend_core_flow_bound 	<ul style="list-style-type: none"> Topdown_Frontend 	<ul style="list-style-type: none"> Stall
0x8162, STALL_FRONTEND_FLUSH	<ul style="list-style-type: none"> frontend_bound bad_speculation frontend_core_flush_resteer_bound frontend_core_flush_machine_clear_bound frontend_core_flush_bound 	<ul style="list-style-type: none"> Topdown_Frontend Topdown_L1 	<ul style="list-style-type: none"> Stall
0x8164, STALL_BACKEND_MEMBOUND	<ul style="list-style-type: none"> backend_mem_bound backend_mem_cache_bound backend_mem_tlb_bound backend_mem_store_bound backend_mem_cme_bound 	<ul style="list-style-type: none"> Topdown_Backend 	<ul style="list-style-type: none"> Stall

Code, Mnemonic	Metrics	Metric Groups	Functional Groups
0x8165, STALL_BACKEND_L1D	<ul style="list-style-type: none"> backend_mem_cache_bound backend_cache_l1d_bound backend_cache_l2d_bound 	<ul style="list-style-type: none"> Topdown_Backend 	<ul style="list-style-type: none"> Stall
0x8167, STALL_BACKEND_TLB	<ul style="list-style-type: none"> backend_mem_tlb_bound 	<ul style="list-style-type: none"> Topdown_Backend 	<ul style="list-style-type: none"> Stall
0x8168, STALL_BACKEND_ST	<ul style="list-style-type: none"> backend_mem_store_bound 	<ul style="list-style-type: none"> Topdown_Backend 	<ul style="list-style-type: none"> Stall
0x816A, STALL_BACKEND_CPUBOUND	<ul style="list-style-type: none"> backend_core_bound backend_core_spec_throttle_bound backend_core_other_bound backend_core_rename_bound mcq_stall_percentage backend_core_cme_bound 	<ul style="list-style-type: none"> MCQ_Effectiveness Topdown_Backend 	<ul style="list-style-type: none"> Stall
0x816B, STALL_BACKEND_BUSY	<ul style="list-style-type: none"> backend_busy_bound iq_stall_lsu_percentage iq_stall_sx_percentage iq_stall_mx_percentage iq_stall_vpu_percentage 	<ul style="list-style-type: none"> IQ_Effectiveness Topdown_Backend 	<ul style="list-style-type: none"> Stall
0x816D, STALL_BACKEND_RENAME	<ul style="list-style-type: none"> backend_core_rename_bound 	<ul style="list-style-type: none"> Topdown_Backend 	<ul style="list-style-type: none"> Stall
0x8170, CAS_NEAR_FAIL	-	-	<ul style="list-style-type: none"> Spec_Operation
0x8171, CAS_NEAR_PASS	<ul style="list-style-type: none"> cas_near_pass_ratio cas_near_fail_ratio 	<ul style="list-style-type: none"> Atomics_Effectiveness 	<ul style="list-style-type: none"> Spec_Operation
0x8172, CAS_NEAR_SPEC	<ul style="list-style-type: none"> cas_near_ratio cas_far_ratio cas_near_pass_ratio cas_near_fail_ratio 	<ul style="list-style-type: none"> Atomics_Effectiveness 	<ul style="list-style-type: none"> Spec_Operation
0x8173, CAS_FAR_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x8174, CAS_SPEC	<ul style="list-style-type: none"> cas_near_ratio cas_far_ratio 	<ul style="list-style-type: none"> Atomics_Effectiveness 	<ul style="list-style-type: none"> Spec_Operation
0x8175, LSE_LD_SPEC	<ul style="list-style-type: none"> lse_load_ratio 	<ul style="list-style-type: none"> Atomics_Effectiveness 	<ul style="list-style-type: none"> Spec_Operation
0x8176, LSE_ST_SPEC	<ul style="list-style-type: none"> lse_store_ratio 	<ul style="list-style-type: none"> Atomics_Effectiveness 	<ul style="list-style-type: none"> Spec_Operation
0x8177, LSE_LDST_SPEC	<ul style="list-style-type: none"> lse_atomics_ratio lse_load_ratio lse_store_ratio 	<ul style="list-style-type: none"> Atomics_Effectiveness 	<ul style="list-style-type: none"> Spec_Operation
0x8179, BRNL_INDNR_TAKEN_RETIRED	-	-	<ul style="list-style-type: none"> Retired
0x817A, BL_TAKEN_RETIRED	-	-	<ul style="list-style-type: none"> Retired
0x817B, BRNL_TAKEN_RETIRED	-	-	<ul style="list-style-type: none"> Retired
0x817C, BL_IND_TAKEN_RETIRED	-	-	<ul style="list-style-type: none"> Retired

Code, Mnemonic	Metrics	Metric Groups	Functional Groups
0x817D, BRNL_IND_TAKEN_RETIRED	-	-	<ul style="list-style-type: none"> Retired
0x817E, BL_IMMED_TAKEN_RETIRED	-	-	<ul style="list-style-type: none"> Retired
0x817F, BRNL_IMMED_TAKEN_RETIRED	-	-	<ul style="list-style-type: none"> Retired
0x8180, BR_UNCOND_RETIRED	-	-	<ul style="list-style-type: none"> Retired
0x8181, BR_COND_RETIRED	-	-	<ul style="list-style-type: none"> Retired
0x8188, DTLB_WALK_BLOCK	<ul style="list-style-type: none"> dtlb_walk_block_ratio 	<ul style="list-style-type: none"> DTLB_Effectiveness 	<ul style="list-style-type: none"> TLB
0x8189, ITLB_WALK_BLOCK	<ul style="list-style-type: none"> itlb_walk_block_ratio 	<ul style="list-style-type: none"> ITLB_Effectiveness 	<ul style="list-style-type: none"> TLB
0x818A, DTLB_WALK_PAGE	<ul style="list-style-type: none"> dtlb_walk_page_ratio 	<ul style="list-style-type: none"> DTLB_Effectiveness 	<ul style="list-style-type: none"> TLB
0x818B, ITLB_WALK_PAGE	<ul style="list-style-type: none"> itlb_walk_page_ratio 	<ul style="list-style-type: none"> ITLB_Effectiveness 	<ul style="list-style-type: none"> TLB
0x818D, BUS_REQ_RD	<ul style="list-style-type: none"> bus_read_requests_average_latency in Average_Latency bus_read_requests_average_latency in Bus_Effectiveness 	<ul style="list-style-type: none"> Average_Latency Bus_Effectiveness 	<ul style="list-style-type: none"> Bus
0x818E, BUS_REQ_WR	-	-	<ul style="list-style-type: none"> Bus
0x818F, BUS_REQ	<ul style="list-style-type: none"> bus_access_average_count 	<ul style="list-style-type: none"> Bus_Effectiveness 	<ul style="list-style-type: none"> Bus
0x8190, ISNP_HIT_RD	<ul style="list-style-type: none"> system_peer_cluster_cache_hit_ratio 	<ul style="list-style-type: none"> System_Memory_Effectiveness 	<ul style="list-style-type: none"> Coherency
0x81A4, DSNP_HIT_HWPRF	-	-	<ul style="list-style-type: none"> Coherency
0x81B4, DSNP_HIT	<ul style="list-style-type: none"> system_peer_cluster_cache_hit_ratio 	<ul style="list-style-type: none"> System_Memory_Effectiveness 	<ul style="list-style-type: none"> Coherency
0x81BC, L1D_CACHE_REFILL_HWPRF	<ul style="list-style-type: none"> l1_prefetcher_accuracy 	<ul style="list-style-type: none"> Prefetcher_Effectiveness 	<ul style="list-style-type: none"> L1D_Cache
0x81BD, L2D_CACHE_REFILL_HWPRF	<ul style="list-style-type: none"> l2_prefetcher_coverage_l1hwprf_inclusive l2_prefetcher_accuracy_l1hwprf_inclusive l2_prefetcher_accuracy_l1hwprf_exclusive 	<ul style="list-style-type: none"> Prefetcher_Effectiveness 	<ul style="list-style-type: none"> L2_Cache
0x81C0, L1I_CACHE_HIT_RD	-	-	<ul style="list-style-type: none"> L1I_Cache
0x81C1, L2I_CACHE_HIT_RD	-	-	<ul style="list-style-type: none"> L2_Cache
0x81D0, L1I_CACHE_HIT_RD_FPRFM	-	-	<ul style="list-style-type: none"> L1I_Cache
0x81E0, L1I_CACHE_HIT_RD_FHWPRF	-	-	<ul style="list-style-type: none"> L1I_Cache
0x81EC, L1D_CACHE_HIT_RW_FHWPRF	<ul style="list-style-type: none"> l1_prefetcher_coverage l1_prefetcher_accuracy l1_prefetcher_timeliness 	<ul style="list-style-type: none"> Prefetcher_Effectiveness 	<ul style="list-style-type: none"> L1D_Cache

Code, Mnemonic	Metrics	Metric Groups	Functional Groups
0x81ED, L2D_CACHE_HIT_RW_FHWPRF	<ul style="list-style-type: none"> l2_prefetcher_coverage_l1hwprf_exclusive l2_prefetcher_accuracy_l1hwprf_exclusive l2_prefetcher_timeliness_l1hwprf_exclusive 	<ul style="list-style-type: none"> Prefetcher_Effectiveness 	<ul style="list-style-type: none"> L2_Cache
0x8200, L1I_CACHE_HIT	-	-	<ul style="list-style-type: none"> L1I_Cache
0x8206, L3D_CACHE_HIT	<ul style="list-style-type: none"> system_l3_cache_hit_ratio 	<ul style="list-style-type: none"> System_Memory_Effectiveness 	<ul style="list-style-type: none"> L3_Cache
0x8207, LL_CACHE_HIT	<ul style="list-style-type: none"> system_llc_cache_hit_ratio 	<ul style="list-style-type: none"> System_Memory_Effectiveness 	<ul style="list-style-type: none"> LL_Cache
0x8208, L1I_CACHE_HIT_PRFM	-	-	<ul style="list-style-type: none"> L1I_Cache
0x8209, L2I_CACHE_HIT_PRFM	-	-	<ul style="list-style-type: none"> L2_Cache
0x8240, L1I_LFB_HIT_RD	-	-	<ul style="list-style-type: none"> L1I_Cache
0x8244, L1D_LFB_HIT_RD	-	-	<ul style="list-style-type: none"> L1D_Cache
0x8248, L1D_LFB_HIT_WR	-	-	<ul style="list-style-type: none"> L1D_Cache
0x824C, L1D_LFB_HIT_RW	-	-	<ul style="list-style-type: none"> L1D_Cache
0x824D, L2D_LFB_HIT_RW	-	-	<ul style="list-style-type: none"> L2_Cache
0x8250, L1I_LFB_HIT_RD_FPRFM	-	-	<ul style="list-style-type: none"> L1I_Cache
0x8260, L1I_LFB_HIT_RD_FHWPRF	-	-	<ul style="list-style-type: none"> L1I_Cache
0x826C, L1D_LFB_HIT_RW_FHWPRF	<ul style="list-style-type: none"> l1_prefetcher_coverage l1_prefetcher_accuracy l1_prefetcher_timeliness 	<ul style="list-style-type: none"> Prefetcher_Effectiveness 	<ul style="list-style-type: none"> L1D_Cache
0x826D, L2D_LFB_HIT_RW_FHWPRF	<ul style="list-style-type: none"> l2_prefetcher_coverage_l1hwprf_exclusive l2_prefetcher_accuracy_l1hwprf_exclusive l2_prefetcher_timeliness_l1hwprf_exclusive 	<ul style="list-style-type: none"> Prefetcher_Effectiveness 	<ul style="list-style-type: none"> L2_Cache
0x8284, L1D_CACHE_PRF	-	-	<ul style="list-style-type: none"> L1D_Cache
0x8285, L2D_CACHE_PRF	-	-	<ul style="list-style-type: none"> L2_Cache
0x835D, SE_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x835E, SME_INST_SPEC	<ul style="list-style-type: none"> sme_percentage 	<ul style="list-style-type: none"> Operation_Mix 	<ul style="list-style-type: none"> Spec_Operation
0x835F, SE_INST_SPEC	-	-	<ul style="list-style-type: none"> Spec_Operation
0x8380, ZA_ACTIVE	<ul style="list-style-type: none"> za_active_cycles_ratio 	<ul style="list-style-type: none"> Cycle_Accounting 	<ul style="list-style-type: none"> General

5. Metrics by metric group in C1-Premium

Metrics are measured using different combinations of PMU events. They are organized into groups that can be analyzed together for a use case. To calculate the metrics, two or more PMU counters are programmed with the events listed for the metric. The counters are read at the same time to determine the metric value.

Summary:

- Total metrics: 145

Metrics for C1-Premium are grouped into the following metric groups:

- [Topdown_Backend](#), Topdown Backend (17 metrics)
- [Topdown_CME](#), Topdown SME2 (3 metrics)
- [Cycle_Accounting](#), Cycle Accounting (6 metrics)
- [Operation_Mix](#), Speculative Operation Mix (13 metrics)
- [Topdown_L1](#), Topdown Level 1 (4 metrics)
- [Topdown_Frontend](#), Topdown Frontend (11 metrics)
- [General](#), General (1 metrics)
- [MPKI](#), Misses Per Kilo Instructions (14 metrics)
- [Miss_Ratio](#), Miss Ratio (12 metrics)
- [SVE_Effectiveness](#), SVE Effectiveness (4 metrics)
- [FP_Arithmetic_Intensity](#), Floating Point Arithmetic Intensity (3 metrics)
- [FP_Precision_Mix](#), Floating Point Precision (3 metrics)
- [Branch_Effectiveness](#), Branch Effectiveness (5 metrics)
- [ITLB_Effectiveness](#), Instruction TLB Effectiveness (12 metrics)
- [DTLB_Effectiveness](#), Data TLB Effectiveness (12 metrics)
- [L1I_Cache_Effectiveness](#), L1 Instruction Cache Effectiveness (2 metrics)
- [L1D_Cache_Effectiveness](#), L1 Data Cache Effectiveness (3 metrics)
- [L2D_Cache_Effectiveness](#), L2D Data Unified Cache Effectiveness (3 metrics)
- [L2I_Cache_Effectiveness](#), L2 Instruction Unified Cache Effectiveness (2 metrics)
- [L3_Cache_Effectiveness](#), L3 Unified Cache Effectiveness (2 metrics)
- [LL_Cache_Effectiveness](#), Last Level Cache Effectiveness (3 metrics)
- [Rename_Effectiveness](#), Register Rename Effectiveness (4 metrics)
- [IQ_Effectiveness](#), Issue Queue Effectiveness (4 metrics)
- [MCQ_Effectiveness](#), Main Commit Queue Effectiveness (1 metrics)
- [Port_Utilization](#), Execution Unit Effectiveness (5 metrics)

- [Prefetcher_Effectiveness](#), Prefetcher Effectiveness (9 metrics)
- [Atomics_Effectiveness](#), Atomics Effectiveness (7 metrics)
- [Average_Latency](#), Average Latency (5 metrics)
- [Bus_Effectiveness](#), Bus Effectiveness (2 metrics)
- [System_Memory_Effectiveness](#), System Memory Effectiveness (4 metrics)

5.1 Topdown_Backend metrics for C1-Premium

Topdown Backend. This metric group contains a set of metrics to analyze a backend bound workload.

Summary of metrics in Topdown_Backend:

- Total metrics: 17

Table 5-1: Topdown_Backend metrics summary

Metric	Name	Description
backend_busy_bound	Backend Busy Bound	This metric is the percentage of total cycles stalled in the backend due to issue queues being...
backend_cache_l1d_bound	Backend Cache L1D Bound	This metric is the percentage of total cycles stalled in the backend due to memory access latency...
backend_cache_l2d_bound	Backend Cache L2D Bound	This metric is the percentage of total cycles stalled in the backend due to memory access latency...
backend_core_bound	Backend Core Bound	This metric is the percentage of total cycles stalled in the backend due to backend core resource...
backend_core_cme_bound	Backend Core SME2 Bound	This metric is the percentage of total cycles stalled in the backend due to the resource...
backend_core_other_bound	Backend Core Other Bound	This metric is the percentage of total cycles stalled in the backend as some other resource is...
backend_core_rename_bound	Backend Core Rename Bound	This metric is the percentage of total cycles stalled in the backend as the rename unit registers...
backend_core_spec_throttle_bound	Backend Core Specuation Throttle Bound	This metric is the percentage of total cycles stalled in the backend as the processor is stalling...
backend_mem_bound	Backend Memory Bound	This metric is the percentage of total cycles stalled in the backend due to backend core resource...
backend_mem_cache_bound	Backend Memory Cache Bound	This metric is the percentage of total cycles stalled in the backend due to memory latency issues...
backend_mem_cme_barrier_bound	Backend SME2 LSRT Barrier Bound	This metric is the percentage of total cycles stalled in the backend as the SME2 unit is...
backend_mem_cme_bound	Backend Memory SME2 Bound	This metric is the percentage of total cycles stalled in the backend caused by load or store...
backend_mem_cme_hazard_cpu_bound	Backend SME2 Memory Hazard CPU Bound	This metric is the percentage of total cycles stalled in the backend due to a SME2 LSRT hazard...
backend_mem_cme_lsrt_full_bound	Backend SME2 LSRT Full Bound	This metric is the percentage of total cycles stalled in the backend as the SME2 unit is busy due...

Metric	Name	Description
backend_mem_cpu_hazard_cme_bound	Backend CPU Memory Hazard SME2 Bound	This metric is the percentage of total cycles stalled in the backend as the SME2 unit is busy due...
backend_mem_store_bound	Backend Memory Store Bound	This metric is the percentage of total cycles stalled in the backend due to memory write pending...
backend_mem_tlb_bound	Backend Memory TLB Bound	This metric is the percentage of total cycles stalled in the backend due to memory access latency...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

backend_busy_bound, Backend Busy Bound, metric

This metric is the percentage of total cycles stalled in the backend due to issue queues being full to accept operations for execution.

Units

This unit is expressed as percent of cycles.

Formula

$\text{STALL_BACKEND_BUSY} / \text{STALL_BACKEND} * 100$

Related telemetry artifacts

Events

[STALL_BACKEND](#)

[STALL_BACKEND_BUSY](#)

Metric group

[Topdown_Backend](#)

Methodology

Stage 1

backend_cache_l1d_bound, Backend Cache L1D Bound, metric

This metric is the percentage of total cycles stalled in the backend due to memory access latency issues caused by level 1 data cache misses.

Units

This unit is expressed as percent of cycles.

Formula

$\text{STALL_BACKEND_L1D} / (\text{STALL_BACKEND_L1D} + \text{STALL_BACKEND_MEM}) * 100$

Related telemetry artifacts

Events

[STALL_BACKEND_L1D](#)

[STALL_BACKEND_MEM](#)

Metric group

[Topdown_Backend](#)

Methodology

Stage 1

backend_cache_l2d_bound, Backend Cache L2D Bound, metric

This metric is the percentage of total cycles stalled in the backend due to memory access latency issues caused by level 2 data cache misses.

Units

This unit is expressed as percent of cycles.

Formula

$$\text{STALL_BACKEND_MEM} / (\text{STALL_BACKEND_L1D} + \text{STALL_BACKEND_MEM}) * 100$$

Related telemetry artifacts**Events**

STALL_BACKEND_L1D
STALL_BACKEND_MEM

Metric group

Topdown_Backend

Methodology

Stage 1

backend_core_bound, Backend Core Bound, metric

This metric is the percentage of total cycles stalled in the backend due to backend core resource constraints not related to instruction fetch latency issues caused by memory access components.

Units

This unit is expressed as percent of cycles.

Formula

$$\text{STALL_BACKEND_CPUBOUND} / \text{STALL_BACKEND} * 100$$

Related telemetry artifacts**Events**

STALL_BACKEND
STALL_BACKEND_C PUBOUND

Metric group

Topdown_Backend

Methodology

Stage 1

backend_core_cme_bound, Backend Core SME2 Bound, metric

This metric is the percentage of total cycles stalled in the backend due to the resource constraints to dispatch to the SME2 unit.

Units

This unit is expressed as percent of cycles.

Formula
$$\text{STALL_BACKEND_BUSY_CME} / \text{STALL_BACKEND_CPUBOUND} * 100$$
Related telemetry artifacts**Events**

[STALL_BACKEND_BUSY_CME](#)
[STALL_BACKEND_CPubound](#)

Metric group

[Topdown_Backend](#)

Methodology

Stage 1

backend_core_other_bound, Backend Core Other Bound, metric

This metric is the percentage of total cycles stalled in the backend as some other resource is unavailable.

Units

This unit is expressed as percent of cycles.

Formula
$$\text{IMP_STALL_BACKEND_CPUBOUND_OTHER} / \text{STALL_BACKEND_CPUBOUND} * 100$$
Related telemetry artifacts**Events**

[IMP_STALL_BACKEND_CPubound_OTHER](#)
[STALL_BACKEND_CPubound](#)

Metric group

[Topdown_Backend](#)

Methodology

Stage 1

backend_core_rename_bound, Backend Core Rename Bound, metric

This metric is the percentage of total cycles stalled in the backend as the rename unit registers are unavailable.

Units

This unit is expressed as percent of cycles.

Formula
$$\text{STALL_BACKEND_RENAME} / \text{STALL_BACKEND_CPUBOUND} * 100$$
Related telemetry artifacts**Events**

[STALL_BACKEND_CPubound](#)
[STALL_BACKEND_RENAME](#)

Metric group[Topdown_Backend](#)**Methodology**

Stage 1

backend_core_spec_throttle_bound, Backend Core Specuation Throttle Bound, metric

This metric is the percentage of total cycles stalled in the backend as the processor is stalling due to power throttling linked to low confidence branches

Units

This unit is expressed as percent of cycles.

Formula
$$\text{IMP_STALL_BACKEND_SPEC_THROT} / \text{STALL_BACKEND_CPUBOUND} * 100$$
Related telemetry artifacts**Events**[IMP_STALL_BACKEND_SPEC_THROT](#)
[STALL_BACKEND_CPubound](#)**Metric group**[Topdown_Backend](#)**Methodology**

Stage 1

backend_mem_bound, Backend Memory Bound, metric

This metric is the percentage of total cycles stalled in the backend due to backend core resource constraints related to memory access latency issues caused by memory access components.

Units

This unit is expressed as percent of cycles.

Formula
$$\text{STALL_BACKEND_MEMBOUND} / \text{STALL_BACKEND} * 100$$
Related telemetry artifacts**Events**[STALL_BACKEND](#)
[STALL_BACKEND_MEMBOUND](#)**Metric group**[Topdown_Backend](#)**Methodology**

Stage 1

backend_mem_cache_bound, Backend Memory Cache Bound, metric

This metric is the percentage of total cycles stalled in the backend due to memory latency issues caused by data cache misses.

Units

This unit is expressed as percent of cycles.

Formula

$$\frac{(\text{STALL_BACKEND_L1D} + \text{STALL_BACKEND_MEM})}{\text{STALL_BACKEND_MEMBOUND}} * 100$$

Related telemetry artifacts**Events**

STALL_BACKEND_L1D
STALL_BACKEND_MEM
STALL_BACKEND_MEMBOUND

Metric group

Topdown_Backend

Methodology

Stage 1

backend_mem_cme_barrier_bound, Backend SME2 LSRT Barrier Bound, metric

This metric is the percentage of total cycles stalled in the backend as the SME2 unit is busy because a CPU barrier waits for SME2 load/store transaction completion.

Units

This unit is expressed as percent of cycles.

Formula

$$\frac{\text{STALL_BACKEND_MEM_CME_BARRIER}}{\text{STALL_BACKEND_MEM_CME}} * 100$$

Related telemetry artifacts**Events**

STALL_BACKEND_MEM_CME
STALL_BACKEND_MEM_CME_BARRIER

Metric group

Topdown_Backend

Methodology

Stage 1

backend_mem_cme_bound, Backend Memory SME2 Bound, metric

This metric is the percentage of total cycles stalled in the backend caused by load or store address hazards caused by the SME2 unit memory execution dependency. SME2 unit and PE share the CPU memory subsystem and are synchronized via the LSRT block, causing memory execution units in them to have dependencies on data accesses.

Units

This unit is expressed as percent of cycles.

Formula

$$\frac{\text{STALL_BACKEND_MEM_CME}}{\text{STALL_BACKEND_MEMBOUND}} * 100$$

Related telemetry artifacts**Events**

[STALL_BACKEND_MEMBOUND](#)
[STALL_BACKEND_MEM_CME](#)

Metric group

[Topdown_Backend](#)

Methodology

Stage 1

backend_mem_cme_hazard_cpu_bound, Backend SME2 Memory Hazard CPU Bound, metric

This metric is the percentage of total cycles stalled in the backend due to a SME2 LSRT hazard causing SME2 unit instructions to stall. SME2 unit and PE share the CPU memory subsystem and are synchronized via the LSRT block, causing both the execution units to have dependencies on data accesses.

Units

This unit is expressed as percent of cycles.

Formula

$\text{STALL_BACKEND_MEM_CME_HZ_ON_CPU} / \text{STALL_BACKEND_MEM_CME} * 100$

Related telemetry artifacts**Events**

[STALL_BACKEND_MEM_CME](#)
[STALL_BACKEND_MEM_CME_HZ_ON_CPU](#)

Metric group

[Topdown_Backend](#)

Methodology

Stage 1

backend_mem_cme_lsrt_full_bound, Backend SME2 LSRT Full Bound, metric

This metric is the percentage of total cycles stalled in the backend as the SME2 unit is busy due to the LSRT being full. SME2 unit and PE share the CPU memory subsystem and are synchronized via the LSRT block, causing both execution units to have dependencies on data accesses.

Units

This unit is expressed as percent of cycles.

Formula

$\text{STALL_BACKEND_MEM_CME_LSRT_FULL} / \text{STALL_BACKEND_MEM_CME} * 100$

Related telemetry artifacts**Events**

[STALL_BACKEND_MEM_CME](#)
[STALL_BACKEND_MEM_CME_LSRT_FULL](#)

Metric group

Topdown_Backend

Methodology

Stage 1

backend_mem_cpu_hazard_cme_bound, Backend CPU Memory Hazard SME2 Bound, metric

This metric is the percentage of total cycles stalled in the backend as the SME2 unit is busy due to SME2 LSRT hazard causing CPU instructions to stall. SME2 unit and PE share the CPU memory subsystem and are synchronized via the LSRT block, causing both execution units to have dependencies on data accesses.

Units

This unit is expressed as percent of cycles.

Formula

$$\text{STALL_BACKEND_MEM_CPU_HZ_ON_CME} / \text{STALL_BACKEND_MEM_CME} * 100$$
Related telemetry artifacts**Events**

STALL_BACKEND_MEM_CME

STALL_BACKEND_MEM_CPU_HZ_ON_CME

Metric group

Topdown_Backend

Methodology

Stage 1

backend_mem_store_bound, Backend Memory Store Bound, metric

This metric is the percentage of total cycles stalled in the backend due to memory write pending caused by stores stalled in the pre-commit stage.

Units

This unit is expressed as percent of cycles.

Formula

$$\text{STALL_BACKEND_ST} / \text{STALL_BACKEND_MEMBOUND} * 100$$
Related telemetry artifacts**Events**

STALL_BACKEND_MEMBOUND

STALL_BACKEND_ST

Metric group

Topdown_Backend

Methodology

Stage 1

backend_mem_tlb_bound, Backend Memory TLB Bound, metric

This metric is the percentage of total cycles stalled in the backend due to memory access latency issues caused by data TLB misses.

Units

This unit is expressed as percent of cycles.

Formula

$$\text{STALL_BACKEND_TLB} / \text{STALL_BACKEND_MEMBOUND} * 100$$

Related telemetry artifacts**Events**

[STALL_BACKEND_MEMBOUND](#)
[STALL_BACKEND_TLB](#)

Metric group

[Topdown_Backend](#)

Methodology

Stage 1

5.2 Topdown_CME metrics for C1-Premium

Topdown SME2. This metric group contains a set of metrics to analyse an SME2 bound workload.

Summary of metrics in Topdown_CME:

- Total metrics: 3

Table 5-2: Topdown_CME metrics summary

Metric	Name	Description
backend_cme_backpressure_bound	Backend SME2 Backpressure Bound	This metric is the percentage of total cycles stalled in the backend as the SME2 unit is busy due...
backend_cme_busy_arb_bound	Backend SME2 Arbitration Bound	This metric is the percentage of total cycles stalled in the backend because an SME2 unit is...
backend_cme_cpu_bound	Backend SME2 CPU Bound	This metric is the percentage of total cycles stalled in the SME2 unit of the backend due to...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

backend_cme_backpressure_bound, Backend SME2 Backpressure Bound, metric

This metric is the percentage of total cycles stalled in the backend as the SME2 unit is busy due to other reasons.

Units

This unit is expressed as percent of cycles.

Formula

$$\text{STALL_BACKEND_BUSY_CMEBOUND} / \text{STALL_BACKEND_BUSY_CME} * 100$$

Related telemetry artifacts**Events**

STALL_BACKEND_BUSY_CME
STALL_BACKEND_BUSY_CMEBOUND

Metric group

Topdown_CME

Methodology

Stage 1

backend_cme_busy_arb_bound, Backend SME2 Arbitration Bound, metric

This metric is the percentage of total cycles stalled in the backend because an SME2 unit is busy. The instruction cannot be sent to the SME2 unit because it is waiting for arbitration.

Units

This unit is expressed as percent of cycles.

Formula

$$\text{STALL_BACKEND_BUSY_CME_ARB} / \text{STALL_BACKEND_BUSY_CME} * 100$$

Related telemetry artifacts**Events**

STALL_BACKEND_BUSY_CME
STALL_BACKEND_BUSY_CME_ARB

Metric group

Topdown_CME

Methodology

Stage 1

backend_cme_cpu_bound, Backend SME2 CPU Bound, metric

This metric is the percentage of total cycles stalled in the SME2 unit of the backend due to dependency to the other units of CPU for execution.

Units

This unit is expressed as percent of cycles.

Formula

$$\text{STALL_BACKEND_BUSY_CME_CPUBOUND} / \text{STALL_BACKEND_BUSY_CME} * 100$$

Related telemetry artifacts**Events**

STALL_BACKEND_BUSY_CME
STALL_BACKEND_BUSY_CME_C PUBOUND

Metric group
Topdown_CME

Methodology
Stage 1

5.3 Cycle_Accounting metrics for C1-Premium

Cycle Accounting. This metric group contains a set of metrics that measure the percentage of processor cycles stalled in either frontend or backend of the processor.

Summary of metrics in Cycle_Accounting:

- Total metrics: 6

Table 5-3: Cycle_Accounting metrics summary

Metric	Name	Description
backend_stalled_cycles	Backend Stalled Cycles	This metric is the percentage of cycles that were stalled due to resource constraints in the...
cme_alloc_cycles_ratio	SME2 Allocation Cycles Ratio	This metric is measures the ratio of cycles where the CPU had an SME2 unit allocated to it, such...
cme_arb_pending_ratio	SME2 Arbitration Pending Cycles Ratio	This metric is measures the ratio of cycles where the CPU is in arbitration while attempting to...
frontend_stalled_cycles	Frontend Stalled Cycles	This metric is the percentage of cycles that were stalled due to resource constraints in the...
sm_active_cycles_ratio	SM Active Cycles Ratio	This metric is measures the ratio of cycles when PSTATE.SM was enabled to the total number of CPU...
za_active_cycles_ratio	ZA Active Cycles Ratio	This metric is measures the ratio of cycles when PSTATE.ZA was enabled to the total number of CPU...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

backend_stalled_cycles, Backend Stalled Cycles, metric

This metric is the percentage of cycles that were stalled due to resource constraints in the backend unit of the processor.

Units
This unit is expressed as percent of cycles.

Formula
 $STALL_BACKEND / (CPU_CYCLES - IMP_WFX_CLOCK_CYCLES) * 100$

Related telemetry artifacts

Events
CPU_CYCLES
IMP_WFX_CLOCK_CYCLES
STALL_BACKEND

Metric group[Cycle_Accounting](#)**Methodology**

Stage 2

cme_alloc_cycles_ratio, SME2 Allocation Cycles Ratio, metric

This metric is measures the ratio of cycles where the CPU had an SME2 unit allocated to it, such that the SME and Streaming SVE state of the CPU is held in that SME2 unit to the total number of CPU cycles.

Units

This unit is expressed as percent of cycles.

Formula
$$\text{CYCLES_CME_ALLOC} / \text{CPU_CYCLES} * 100$$
Related telemetry artifacts**Events**[CPU_CYCLES](#)[CYCLES_CME_ALLOC](#)**Metric group**[Cycle_Accounting](#)**Methodology**

Stage 2

cme_arb_pending_ratio, SME2 Arbitration Pending Cycles Ratio, metric

This metric is measures the ratio of cycles where the CPU is in arbitration while attempting to access an SME2 unit to the total number of CPU cycles.

Units

This unit is expressed as percent of cycles.

Formula
$$\text{CYCLES_ARB_PENDING_CME} / \text{CPU_CYCLES} * 100$$
Related telemetry artifacts**Events**[CPU_CYCLES](#)[CYCLES_ARB_PENDING_CME](#)**Metric group**[Cycle_Accounting](#)**Methodology**

Stage 2

frontend_stalled_cycles, Frontend Stalled Cycles, metric

This metric is the percentage of cycles that were stalled due to resource constraints in the frontend unit of the processor.

Units

This unit is expressed as percent of cycles.

Formula

$$\frac{(\text{STALL_FRONTEND} - \text{IMP_WFX_CLOCK_CYCLES})}{(\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES})} * 100$$

Related telemetry artifacts**Events**

CPU_CYCLES
IMP_WFX_CLOCK_CYCLES
STALL_FRONTEND

Metric group

Cycle_Accounting

Methodology

Stage 2

sm_active_cycles_ratio, SM Active Cycles Ratio, metric

This metric is measures the ratio of cycles when PSTATE.SM was enabled to the total number of CPU cycles.

Units

This unit is expressed as percent of cycles.

Formula

$$\text{SM_ACTIVE_CYCLES} / \text{CPU_CYCLES} * 100$$

Related telemetry artifacts**Events**

CPU_CYCLES
SM_ACTIVE_CYCLES

Metric group

Cycle_Accounting

Methodology

Stage 2

za_active_cycles_ratio, ZA Active Cycles Ratio, metric

This metric is measures the ratio of cycles when PSTATE.ZA was enabled to the total number of CPU cycles.

Units

This unit is expressed as percent of cycles.

Formula

$$\text{ZA_ACTIVE} / \text{CPU_CYCLES} * 100$$

Related telemetry artifacts**Events**

CPU_CYCLES

ZA_ACTIVE

Metric group

Cycle_Accounting

Methodology

Stage 2

5.4 Operation_Mix metrics for C1-Premium

Speculative Operation Mix. This metric group provides the distribution of micro-operation types executed for the program.

Summary of metrics in Operation_Mix:

- Total metrics: 13

Table 5-4: Operation_Mix metrics summary

Metric	Name	Description
barrier_percentage	Barrier Operations Percentage	This metric measures instruction and data barrier operations as a percentage of operations...
branch_percentage	Branch Operations Percentage	This metric measures branch operations as a percentage of operations speculatively executed.
crypto_percentage	Crypto Operations Percentage	This metric measures crypto operations as a percentage of operations speculatively executed.
integer_dp_percentage	Integer Operations Percentage	This metric measures scalar integer operations as a percentage of operations speculatively executed.
load_ls_percentage	Load Operations Percentage of Load/Store Operations	This metric measures load operations as a percentage of load and store operations speculatively...
load_percentage	Load Operations Percentage	This metric measures load operations as a percentage of operations speculatively executed.
load_store_percentage	Load/Store Operations Percentage	This metric measures load and store operations as a percentage of operations speculatively executed.
scalar_fp_percentage	Floating Point Operations Percentage	This metric measures scalar floating point operations as a percentage of operations speculatively...
simd_percentage	Advanced SIMD Operations Percentage	This metric measures advanced SIMD operations as a percentage of total operations speculatively...
sme_percentage	SME Operations Percentage	This metric measures Scalable Matrix extension data processing operations as a percentage of...
store_ls_percentage	Store Operations Percentage of Load/Store Operations	This metric measures store operations as a percentage of load and store operations speculatively...

Metric	Name	Description
store_percentage	Store Operations Percentage	This metric measures store operations as a percentage of operations speculatively executed.
sve_percentage	SVE Operations Percentage	This metric measures scalable vector operations as a percentage of operations speculatively...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

barrier_percentage, Barrier Operations Percentage, metric

This metric measures instruction and data barrier operations as a percentage of operations speculatively executed.

Units

This unit is expressed as percent of operations.

Formula

$$(\text{ISB_SPEC} + \text{DSB_SPEC} + \text{DMB_SPEC}) / \text{INST_SPEC} * 100$$

Related telemetry artifacts

Events

[DMB_SPEC](#)
[DSB_SPEC](#)
[INST_SPEC](#)
[ISB_SPEC](#)

Metric group

[Operation_Mix](#)

Methodology

Stage 2

branch_percentage, Branch Operations Percentage, metric

This metric measures branch operations as a percentage of operations speculatively executed.

Units

This unit is expressed as percent of operations.

Formula

$$\text{PC_WRITE_SPEC} / \text{INST_SPEC} * 100$$

Related telemetry artifacts

Events

[INST_SPEC](#)
[PC_WRITE_SPEC](#)

Metric group

[Operation_Mix](#)

Methodology

Stage 2

crypto_percentage, Crypto Operations Percentage, metric

This metric measures crypto operations as a percentage of operations speculatively executed.

Units

This unit is expressed as percent of operations.

Formula

$$\text{CRYPTO_SPEC} / \text{INST_SPEC} * 100$$

Related telemetry artifacts**Events**

CRYPTO_SPEC
INST_SPEC

Metric group

Operation_Mix

Methodology

Stage 2

integer_dp_percentage, Integer Operations Percentage, metric

This metric measures scalar integer operations as a percentage of operations speculatively executed.

Units

This unit is expressed as percent of operations.

Formula

$$\text{DP_SPEC} / \text{INST_SPEC} * 100$$

Related telemetry artifacts**Events**

DP_SPEC
INST_SPEC

Metric group

Operation_Mix

Methodology

Stage 2

load_ls_percentage, Load Operations Percentage of Load/Store Operations, metric

This metric measures load operations as a percentage of load and store operations speculatively executed.

Units

This unit is expressed as percent of load/store operations.

Formula

$$\text{LD_SPEC} / \text{LDST_SPEC} * 100$$

Related telemetry artifacts**Events**

LDST_SPEC
LD_SPEC

Metric group

Operation_Mix

Methodology

Stage 2

load_percentage, Load Operations Percentage, metric

This metric measures load operations as a percentage of operations speculatively executed.

Units

This unit is expressed as percent of operations.

Formula

$$\text{LD_SPEC} / \text{INST_SPEC} * 100$$

Related telemetry artifacts**Events**

INST_SPEC
LD_SPEC

Metric group

Operation_Mix

Methodology

Stage 2

load_store_percentage, Load/Store Operations Percentage, metric

This metric measures load and store operations as a percentage of operations speculatively executed.

Units

This unit is expressed as percent of operations.

Formula

$$\text{LDST_SPEC} / \text{INST_SPEC} * 100$$

Related telemetry artifacts**Events**

INST_SPEC
LDST_SPEC

Metric group

Operation_Mix

Methodology

Stage 2

scalar_fp_percentage, Floating Point Operations Percentage, metric

This metric measures scalar floating point operations as a percentage of operations speculatively executed.

Units

This unit is expressed as percent of operations.

Formula

$$\text{VFP_SPEC} / \text{INST_SPEC} * 100$$

Related telemetry artifacts**Events**

INST_SPEC

VFP_SPEC

Metric group

Operation_Mix

Methodology

Stage 2

simd_percentage, Advanced SIMD Operations Percentage, metric

This metric measures advanced SIMD operations as a percentage of total operations speculatively executed.

Units

This unit is expressed as percent of operations.

Formula

$$\text{ASE_SPEC} / \text{INST_SPEC} * 100$$

Related telemetry artifacts**Events**

ASE_SPEC

INST_SPEC

Metric group

Operation_Mix

Methodology

Stage 2

sme_percentage, SME Operations Percentage, metric

This metric measures Scalable Matrix extension data processing operations as a percentage of operations speculatively executed.

Units

This unit is expressed as percent of operations.

Formula

$$\text{SME_INST_SPEC} / \text{INST_SPEC} * 100$$

Related telemetry artifacts**Events**

INST_SPEC
SME_INST_SPEC

Metric group

Operation_Mix

Methodology

Stage 2

store_Is_percentage, Store Operations Percentage of Load/Store Operations, metric

This metric measures store operations as a percentage of load and store operations speculatively executed.

Units

This unit is expressed as percent of load/store operations.

Formula

$$\text{ST_SPEC} / \text{LDST_SPEC} * 100$$

Related telemetry artifacts**Events**

LDST_SPEC
ST_SPEC

Metric group

Operation_Mix

Methodology

Stage 2

store_percentage, Store Operations Percentage, metric

This metric measures store operations as a percentage of operations speculatively executed.

Units

This unit is expressed as percent of operations.

Formula

$$\text{ST_SPEC} / \text{INST_SPEC} * 100$$

Related telemetry artifacts**Events**

INST_SPEC
ST_SPEC

Metric group

Operation_Mix

Methodology
Stage 2

sve_percentage, SVE Operations Percentage, metric

This metric measures scalable vector operations as a percentage of operations speculatively executed.

Units
This unit is expressed as percent of operations.

Formula
$$\text{SVE_SPEC} / \text{INST_SPEC} * 100$$

Related telemetry artifacts

- Events
- INST_SPEC
 - SVE_SPEC
- Metric group
- Operation_Mix
- Methodology
- Stage 2

5.5 Topdown_L1 metrics for C1-Premium

Topdown Level 1. This metric group contains the first set of metrics to begin topdown analysis of application performance, which provide the percentage distribution of processor pipeline utilization.

Summary of metrics in Topdown_L1:

- Total metrics: 4

Table 5-5: Topdown_L1 metrics summary

Metric	Name	Description
backend_bound	Backend Bound	This metric is the percentage of total slots that were stalled due to resource constraints in the...
bad_speculation	Bad Speculation	This metric is the percentage of total slots that executed operations and didn't retire due to a...
frontend_bound	Frontend Bound	This metric is the percentage of total slots that were stalled due to resource constraints in the...
retiring	Retiring	This metric is the percentage of total slots that retired operations, which indicates cycles that...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

backend_bound, Backend Bound, metric

This metric is the percentage of total slots that were stalled due to resource constraints in the backend of the processor.

Units

This unit is expressed as percent of slots.

Formula

$$\text{STALL_SLOT_BACKEND} / (10 * (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES})) * 100$$

Related telemetry artifacts**Events**

CPU_CYCLES
IMP_WFX_CLOCK_CYCLES
STALL_SLOT_BACKEND

Metric group

Topdown_L1

Methodology

Stage 1

bad_speculation, Bad Speculation, metric

This metric is the percentage of total slots that executed operations and didn't retire due to a pipeline flush. This indicates cycles that were utilized but inefficiently.

Units

This unit is expressed as percent of slots.

Formula

$$(1 - (\text{STALL_SLOT} - 10 * \text{IMP_WFX_CLOCK_CYCLES}) / (10 * (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES}))) * (1 - \text{OP_RETIRED} / \text{OP_SPEC}) * 100 + \text{STALL_FRONTEND_FLUSH} / (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES}) * 100$$

Related telemetry artifacts**Events**

CPU_CYCLES
IMP_WFX_CLOCK_CYCLES
OP_RETIRED
OP_SPEC
STALL_FRONTEND_FLUSH
STALL_SLOT

Metric group

Topdown_L1

Methodology

Stage 1

frontend_bound, Frontend Bound, metric

This metric is the percentage of total slots that were stalled due to resource constraints in the frontend of the processor.

Units

This unit is expressed as percent of slots.

Formula

$$\frac{((\text{STALL_SLOT_FRONTEND} - 10 * \text{IMP_WFX_CLOCK_CYCLES}) / (10 * (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES})) - (\text{STALL_FRONTEND_FLUSH} - \text{IMP_WFX_CLOCK_CYCLES}) / (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES})) * 100}{}$$

Related telemetry artifacts**Events**

CPU_CYCLES
IMP_WFX_CLOCK_CYCLES
STALL_FRONTEND_FLUSH
STALL_SLOT_FRONTEND

Metric group

Topdown_L1

Methodology

Stage 1

retiring, Retiring, metric

This metric is the percentage of total slots that retired operations, which indicates cycles that were utilized efficiently.

Units

This unit is expressed as percent of slots.

Formula

$$\frac{(1 - (\text{STALL_SLOT} - 10 * \text{IMP_WFX_CLOCK_CYCLES}) / ((\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES}) * 10)) * (\text{OP_RETIRED} / \text{OP_SPEC}) * 100}{}$$

Related telemetry artifacts**Events**

CPU_CYCLES
IMP_WFX_CLOCK_CYCLES
OP_RETIRED
OP_SPEC
STALL_SLOT

Metric group

Topdown_L1

Methodology

Stage 1

5.6 Topdown_Frontend metrics for C1-Premium

Topdown Frontend. This metric group contains a set of metrics to analyse a frontend bound workload.

Summary of metrics in Topdown_Frontend:

- Total metrics: 11

Table 5-6: Topdown_Frontend metrics summary

Metric	Name	Description
frontend_cache_l1i_bound	Frontend Cache L1I Bound	This metric is the percentage of total cycles stalled in the frontend due to memory access...
frontend_cache_l2i_bound	Frontend Cache L2I Bound	This metric is the percentage of total cycles stalled in the frontend due to memory access...
frontend_core_bound	Frontend Core Bound	This metric is the percentage of total cycles stalled in the frontend due to frontend core...
frontend_core_flow_bound	Frontend Core Flow Bound	This metric is the percentage of total cycles stalled in the frontend as the decode unit is...
frontend_core_flush_bound	Frontend Core Flush Bound	This metric is the percentage of total cycles stalled in the frontend as the processor is...
frontend_core_flush_machine_clear_bound	Frontend Core Flush Machine Clear Bound	This metric is the percentage of total cycles stalled in the frontend as the processor is...
frontend_core_flush_resteer_bound	Frontend Core Flush Branch Resteer Bound	This metric is the percentage of total cycles stalled in the frontend as the processor is...
frontend_core_spec_throttle_bound	Frontend Core Specuation Throttle Bound	This metric is the percentage of total cycles stalled in the frontend as the processor is...
frontend_mem_bound	Frontend Memory Bound	This metric is the percentage of total cycles stalled in the frontend due to frontend core...
frontend_mem_cache_bound	Frontend Mem Cache Bound	This metric is the percentage of total cycles stalled in the frontend due to instruction fetch...
frontend_mem_tlb_bound	Frontend Mem TLB Bound	This metric is the percentage of total cycles stalled in the frontend due to instruction fetch...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

frontend_cache_l1i_bound, Frontend Cache L1I Bound, metric

This metric is the percentage of total cycles stalled in the frontend due to memory access latency issues caused by level 1 instruction cache misses.

Units

This unit is expressed as percent of cycles.

Formula

$$\text{STALL_FRONTEND_L1I} / (\text{STALL_FRONTEND_L1I} + \text{STALL_FRONTEND_MEM}) * 100$$

Related telemetry artifacts**Events**

STALL_FRONTEND_L1I
STALL_FRONTEND_MEM

Metric group

Topdown_Frontend

Methodology

Stage 1

frontend_cache_l2i_bound, Frontend Cache L2I Bound, metric

This metric is the percentage of total cycles stalled in the frontend due to memory access latency issues caused by level 2 instruction cache misses.

Units

This unit is expressed as percent of cycles.

Formula

$$\text{STALL_FRONTEND_MEM} / (\text{STALL_FRONTEND_L1I} + \text{STALL_FRONTEND_MEM}) * 100$$
Related telemetry artifacts**Events**

STALL_FRONTEND_L1I
STALL_FRONTEND_MEM

Metric group

Topdown_Frontend

Methodology

Stage 1

frontend_core_bound, Frontend Core Bound, metric

This metric is the percentage of total cycles stalled in the frontend due to frontend core resource constraints not related to instruction fetch latency issues caused by memory access components.

Units

This unit is expressed as percent of cycles.

Formula

$$(\text{STALL_FRONTEND_CPUBOUND} - \text{IMP_WFX_CLOCK_CYCLES}) / (\text{STALL_FRONTEND} - \text{IMP_WFX_CLOCK_CYCLES}) * 100$$
Related telemetry artifacts**Events**

IMP_WFX_CLOCK_CYCLES
STALL_FRONTEND
STALL_FRONTEND_C PUBOUND

Metric group

Topdown_Frontend

Methodology

Stage 1

frontend_core_flow_bound, Frontend Core Flow Bound, metric

This metric is the percentage of total cycles stalled in the frontend as the decode unit is awaiting input from the branch prediction unit.

Units

This unit is expressed as percent of cycles.

Formula
$$\text{STALL_FRONTEND_FLOW} / \text{STALL_FRONTEND_CPUBOUND} * 100$$
Related telemetry artifacts**Events**

STALL_FRONTEND_CPUBOUND

STALL_FRONTEND_FLOW

Metric group

Topdown_Frontend

Methodology

Stage 1

frontend_core_flush_bound, Frontend Core Flush Bound, metric

This metric is the percentage of total cycles stalled in the frontend as the processor is recovering from a pipeline flush caused by bad speculation or other machine restears.

Units

This unit is expressed as percent of cycles.

Formula
$$(\text{STALL_FRONTEND_FLUSH} - \text{IMP_WFX_CLOCK_CYCLES}) / (\text{STALL_FRONTEND_CPUBOUND} - \text{IMP_WFX_CLOCK_CYCLES}) * 100$$
Related telemetry artifacts**Events**

IMP_WFX_CLOCK_CYCLES

STALL_FRONTEND_CPUBOUND

STALL_FRONTEND_FLUSH

Metric group

Topdown_Frontend

Methodology

Stage 1

frontend_core_flush_machine_clear_bound, Frontend Core Flush Machine Clear Bound, metric

This metric is the percentage of total cycles stalled in the frontend as the processor is recovering from a pipeline flush caused by machine resteeers other than bad speculation.

Units

This unit is expressed as percent of cycles.

Formula

$$\frac{(\text{IMP_STALL_FRONTEND_FLUSH_CLEAR} - \text{IMP_WFX_CLOCK_CYCLES})}{(\text{STALL_FRONTEND_FLUSH} - \text{IMP_WFX_CLOCK_CYCLES})} * 100$$

Related telemetry artifacts**Events**

IMP_STALL_FRONTEND_FLUSH_CLEAR
IMP_WFX_CLOCK_CYCLES
STALL_FRONTEND_FLUSH

Metric group

Topdown_Frontend

Methodology

Stage 1

frontend_core_flush_resteer_bound, Frontend Core Flush Branch Resteer Bound, metric

This metric is the percentage of total cycles stalled in the frontend as the processor is recovering from a pipeline flush caused by bad speculation.

Units

This unit is expressed as percent of cycles.

Formula

$$\text{IMP_STALL_FRONTEND_FLUSH_RESTEER} / \text{STALL_FRONTEND_FLUSH} * 100$$

Related telemetry artifacts**Events**

IMP_STALL_FRONTEND_FLUSH_RESTEER
STALL_FRONTEND_FLUSH

Metric group

Topdown_Frontend

Methodology

Stage 1

frontend_core_spec_throttle_bound, Frontend Core Specuation Throttle Bound, metric

This metric is the percentage of total cycles stalled in the frontend as the processor is stalling due to power throttling linked to low confidence branches

Units

This unit is expressed as percent of cycles.

Formula

$$\text{IMP_STALL_FRONTEND_SPEC_THROT} / \text{STALL_FRONTEND_CPUBOUND} * 100$$

Related telemetry artifacts**Events**

IMP_STALL_FRONTEND_SPEC_THROT
STALL_FRONTEND_CPubound

Metric group

Topdown_Frontend

Methodology

Stage 1

frontend_mem_bound, Frontend Memory Bound, metric

This metric is the percentage of total cycles stalled in the frontend due to frontend core resource constraints related to the instruction fetch latency issues caused by memory access components.

Units

This unit is expressed as percent of cycles.

Formula

$$\text{STALL_FRONTEND_MEMBOUND} / \text{STALL_FRONTEND} * 100$$

Related telemetry artifacts**Events**

STALL_FRONTEND
STALL_FRONTEND_MEMBOUND

Metric group

Topdown_Frontend

Methodology

Stage 1

frontend_mem_cache_bound, Frontend Mem Cache Bound, metric

This metric is the percentage of total cycles stalled in the frontend due to instruction fetch latency issues caused by instruction cache misses.

Units

This unit is expressed as percent of cycles.

Formula

$$(\text{STALL_FRONTEND_L1I} + \text{STALL_FRONTEND_MEM}) / \text{STALL_FRONTEND_MEMBOUND} * 100$$

Related telemetry artifacts**Events**

STALL_FRONTEND_L1I
STALL_FRONTEND_MEM

STALL_FRONTEND_MEMBOUND

Metric group
Topdown_Frontend
Methodology

Stage 1

frontend_mem_tlb_bound, Frontend Mem TLB Bound, metric

This metric is the percentage of total cycles stalled in the frontend due to instruction fetch latency issues caused by instruction TLB misses.

Units
This unit is expressed as percent of cycles.

Formula
$$\text{STALL_FRONTEND_TLB} / \text{STALL_FRONTEND_MEMBOUND} * 100$$

Related telemetry artifacts

Events
STALL_FRONTEND_MEMBOUND
STALL_FRONTEND_TLB

Metric group
Topdown_Frontend

Methodology
Stage 1

5.7 General metrics for C1-Premium

General. This metric group contains general CPU metrics for performance analysis.

Summary of metrics in General:

- Total metrics: 1

Table 5-7: General metrics summary

Metric	Name	Description
ipc	Instructions Per Cycle	This metric measures the number of instructions retired per cycle.

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

ipc, Instructions Per Cycle, metric

This metric measures the number of instructions retired per cycle.

Units
This unit is expressed as per cycle.

Formula

$$\text{INST_RETIRED} / (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES})$$
Related telemetry artifacts**Events**

CPU_CYCLES
IMP_WFX_CLOCK_CYCLES
INST_RETIRED

Metric group

General

Methodology

Stage 2

5.8 MPKI metrics for C1-Premium

Misses Per Kilo Instructions. This metric group contains metrics for different CPU resources that can be measured as misses per kilo instructions.

Summary of metrics in MPKI:

- Total metrics: 14

Table 5-8: MPKI metrics summary

Metric	Name	Description
branch_mpki	Branch MPKI	This metric measures the number of branch mispredictions per thousand instructions executed.
dtlb_mpki	DTLB MPKI	This metric measures the number of data TLB Walks per thousand instructions executed.
itlb_mpki	ITLB MPKI	This metric measures the number of instruction TLB Walks per thousand instructions executed.
l1d_cache_demand_mpki	L1D Cache Demand MPKI	This metric measures the number of level 1 data cache demand accesses missed per thousand...
l1d_cache_mpki	L1D Cache MPKI	This metric measures the number of level 1 data cache accesses missed per thousand instructions...
l1d_tlb_mpki	L1 Data TLB MPKI	This metric measures the number of level 1 data TLB accesses missed per thousand instructions...
l1i_cache_mpki	L1I Cache MPKI	This metric measures the number of level 1 instruction cache accesses missed per thousand...
l1i_tlb_mpki	L1 Instruction TLB MPKI	This metric measures the number of level 1 instruction TLB accesses missed per thousand...
l2_tlb_mpki	L2 Unified TLB MPKI	This metric measures the number of level 2 unified TLB accesses missed per thousand instructions...
l2d_cache_demand_mpki	L2D Cache Demand MPKI	This metric measures the number of level 2 unified cache data demand accesses missed per thousand...

Metric	Name	Description
l2d_cache_mпки	L2D Cache MPKI	This metric measures the number of level 2 unified cache data accesses missed per thousand...
l2i_cache_mпки	L2I Cache MPKI	This metric measures the number of level 2 unified cache instruction accesses missed per thousand...
l3_cache_mпки	L3 Cache MPKI	This metric measures the number of level 3 unified cache accesses missed per thousand...
ll_cache_read_mпки	LL Cache Read MPKI	This metric measures the number of last level cache read accesses missed per thousand...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

branch_mпки, Branch MPKI, metric

This metric measures the number of branch mispredictions per thousand instructions executed.

Units

This unit is expressed as mпки.

Formula

$\text{BR_MIS_PRED_RETIRED} / \text{INST_RETIRED} * 1000$

Related telemetry artifacts

Events

[BR_MIS_PRED_RETIRED](#)
[INST_RETIRED](#)

Metric group

[MPKI](#)

Other metric group: [Branch_Effectiveness](#)

Methodology

Stage 2

dtlb_mпки, DTLB MPKI, metric

This metric measures the number of data TLB Walks per thousand instructions executed.

Units

This unit is expressed as mпки.

Formula

$\text{DTLB_WALK} / \text{INST_RETIRED} * 1000$

Related telemetry artifacts

Events

[DTLB_WALK](#)
[INST_RETIRED](#)

Metric group

[MPKI](#)

Other metric group: [DTLB_Effectiveness](#)

Methodology

Stage 2

itlb_mpki, ITLB MPKI, metric

This metric measures the number of instruction TLB Walks per thousand instructions executed.

Units

This unit is expressed as mpki.

Formula

$$\text{ITLB_WALK} / \text{INST_RETIRED} * 1000$$

Related telemetry artifacts

Events

[INST_RETIRED](#)

[ITLB_WALK](#)

Metric group

[MPKI](#)

Other metric group: [ITLB_Effectiveness](#)

Methodology

Stage 2

l1d_cache_demand_mpki, L1D Cache Demand MPKI, metric

This metric measures the number of level 1 data cache demand accesses missed per thousand instructions executed.

Units

This unit is expressed as mpki.

Formula

$$(\text{L1D_CACHE_REFILL_RD} + \text{L1D_CACHE_REFILL_WR}) / \text{INST_RETIRED} * 1000$$

Related telemetry artifacts

Events

[INST_RETIRED](#)

[L1D_CACHE_REFILL_RD](#)

[L1D_CACHE_REFILL_WR](#)

Metric group

[MPKI](#)

Other metric group: [L1D_Cache_Effectiveness](#)

Methodology

Stage 2

l1d_cache_mpki, L1D Cache MPKI, metric

This metric measures the number of level 1 data cache accesses missed per thousand instructions executed.

Units

This unit is expressed as mpki.

Formula

$$\text{L1D_CACHE_REFILL} / \text{INST_RETIRED} * 1000$$

Related telemetry artifacts**Events**

[INST_RETIRED](#)
[L1D_CACHE_REFILL](#)

Metric group

[MPKI](#)
Other metric group: [L1D_Cache_Effectiveness](#)

Methodology

Stage 2

l1d_tlb_mpki, L1 Data TLB MPKI, metric

This metric measures the number of level 1 data TLB accesses missed per thousand instructions executed.

Units

This unit is expressed as mpki.

Formula

$$\text{L1D_TLB_REFILL} / \text{INST_RETIRED} * 1000$$

Related telemetry artifacts**Events**

[INST_RETIRED](#)
[L1D_TLB_REFILL](#)

Metric group

[MPKI](#)
Other metric group: [DTLB_Effectiveness](#)

Methodology

Stage 2

l1i_cache_mpki, L1I Cache MPKI, metric

This metric measures the number of level 1 instruction cache accesses missed per thousand instructions executed.

Units

This unit is expressed as mpki.

Formula

$$\text{L1I_CACHE_REFILL} / \text{INST_RETIRED} * 1000$$

Related telemetry artifacts**Events**

[INST_RETIRED](#)
[L1I_CACHE_REFILL](#)

Metric group

[MPKI](#)
Other metric group: [L1I_Cache_Effectiveness](#)

Methodology

Stage 2

[l1i_tlb_mпки](#), L1 Instruction TLB MPKI, metric

This metric measures the number of level 1 instruction TLB accesses missed per thousand instructions executed.

Units

This unit is expressed as mпки.

Formula

$$\text{L1I_TLB_REFILL} / \text{INST_RETIRED} * 1000$$

Related telemetry artifacts**Events**

[INST_RETIRED](#)
[L1I_TLB_REFILL](#)

Metric group

[MPKI](#)
Other metric group: [ITLB_Effectiveness](#)

Methodology

Stage 2

[l2_tlb_mпки](#), L2 Unified TLB MPKI, metric

This metric measures the number of level 2 unified TLB accesses missed per thousand instructions executed.

Units

This unit is expressed as mпки.

Formula

$$\text{L2D_TLB_REFILL} / \text{INST_RETIRED} * 1000$$

Related telemetry artifacts**Events**

[INST_RETIRED](#)

[L2D_TLB_REFILL](#)**Metric group**[MPKI](#)Other metric group: [DTLB_Effectiveness](#)Other metric group: [ITLB_Effectiveness](#)**Methodology**

Stage 2

l2d_cache_demand_mпки, L2D Cache Demand MPKI, metric

This metric measures the number of level 2 unified cache data demand accesses missed per thousand instructions executed. Note that cache accesses in this cache are either data memory access or instruction fetch as this is a unified cache.

Units

This unit is expressed as mпки.

Formula

$$(\text{L2D_CACHE_REFILL_RD} + \text{L2D_CACHE_REFILL_WR}) / \text{INST_RETIRED} * 1000$$

Related telemetry artifacts**Events**[INST_RETIRED](#)[L2D_CACHE_REFILL_RD](#)[L2D_CACHE_REFILL_WR](#)**Metric group**[MPKI](#)Other metric group: [L2D_Cache_Effectiveness](#)**Methodology**

Stage 2

l2d_cache_mпки, L2D Cache MPKI, metric

This metric measures the number of level 2 unified cache data accesses missed per thousand instructions executed. Note that cache accesses in this cache are either data memory access or instruction fetch as this is a unified cache.

Units

This unit is expressed as mпки.

Formula

$$\text{L2D_CACHE_REFILL} / \text{INST_RETIRED} * 1000$$

Related telemetry artifacts**Events**[INST_RETIRED](#)[L2D_CACHE_REFILL](#)

Metric group

MPKI

Other metric group: [L2D_Cache_Effectiveness](#)**Methodology**

Stage 2

l2i_cache_mпки, L2I Cache MPKI, metric

This metric measures the number of level 2 unified cache instruction accesses missed per thousand instructions executed. Note that cache accesses in this cache are either data memory access or instruction fetch as this is a unified cache.

Units

This unit is expressed as mпки.

Formula
$$\text{L2I_CACHE_REFILL} / \text{INST_RETIRED} * 1000$$
Related telemetry artifacts**Events**[INST_RETIRED](#)[L2I_CACHE_REFILL](#)**Metric group**

MPKI

Other metric group: [L2I_Cache_Effectiveness](#)**Methodology**

Stage 2

l3_cache_mпки, L3 Cache MPKI, metric

This metric measures the number of level 3 unified cache accesses missed per thousand instructions executed. Note that cache accesses in this cache are either data memory access or instruction fetch as this is a unified cache.

Units

This unit is expressed as mпки.

Formula
$$\text{L3D_CACHE_REFILL} / \text{INST_RETIRED} * 1000$$
Related telemetry artifacts**Events**[INST_RETIRED](#)[L3D_CACHE_REFILL](#)**Metric group**

MPKI

Other metric group: [L3_Cache_Effectiveness](#)

Methodology
Stage 2

ll_cache_read_mпки, LL Cache Read MPKI, metric

This metric measures the number of last level cache read accesses missed per thousand instructions executed.

Units

This unit is expressed as mпки.

Formula

$$\text{LL_CACHE_MISS_RD} / \text{INST_RETIRED} * 1000$$

Related telemetry artifacts

Events

INST_RETIRED
LL_CACHE_MISS_RD

Metric group

MPKI
Other metric group: LL_Cache_Effectiveness

Methodology
Stage 2

5.9 Miss_Ratio metrics for C1-Premium

Miss Ratio. This metric group contains metrics to measure miss ratios of different processor resources.

Summary of metrics in Miss_Ratio:

- Total metrics: 12

Table 5-9: Miss_Ratio metrics summary

Metric	Name	Description
branch_misprediction_ratio	Branch Misprediction Ratio	This metric measures the ratio of branches mispredicted to the total number of branches...
dtlb_walk_ratio	DTLB Walk Ratio	This metric measures the ratio of data TLB Walks to the total number of data TLB accesses. This...
itlb_walk_ratio	ITLB Walk Ratio	This metric measures the ratio of instruction TLB Walks to the total number of instruction TLB...
l1d_cache_miss_ratio	L1D Cache Miss Ratio	This metric measures the ratio of level 1 data cache accesses missed to the total number of level...
l1d_tlb_miss_ratio	L1 Data TLB Miss Ratio	This metric measures the ratio of level 1 data TLB accesses missed to the total number of level 1...
l1i_cache_miss_ratio	L1I Cache Miss Ratio	This metric measures the ratio of level 1 instruction cache accesses missed to the total number...

Metric	Name	Description
l1i_tlb_miss_ratio	L1 Instruction TLB Miss Ratio	This metric measures the ratio of level 1 instruction TLB accesses missed to the total number of...
l2_tlb_miss_ratio	L2 Unified TLB Miss Ratio	This metric measures the ratio of level 2 unified TLB accesses missed to the total number of...
l2d_cache_miss_ratio	L2D Cache Miss Ratio	This metric measures the ratio of level 2 cache data accesses missed to the total number of level...
l2i_cache_miss_ratio	L2I Cache Miss Ratio	This metric measures the ratio of level 2 cache instruction accesses missed to the total number...
l3_cache_miss_ratio	L3 Cache Miss Ratio	This metric measures the ratio of level 3 cache accesses missed to the total number of level 3...
ll_cache_read_miss_ratio	LL Cache Read Miss Ratio	This metric measures the ratio of last level cache read accesses missed to the total number of...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

branch_misprediction_ratio, Branch Misprediction Ratio, metric

This metric measures the ratio of branches mispredicted to the total number of branches architecturally executed. This gives an indication of the effectiveness of the branch prediction unit.

Units

This unit is expressed as per branch.

Formula

[BR_MIS_PRED_RETIRED](#) / [BR_RETIRED](#)

Related telemetry artifacts

Events

[BR_MIS_PRED_RETIRED](#)

[BR_RETIRED](#)

Metric group

[Miss_Ratio](#)

Other metric group: [Branch_Effectiveness](#)

Methodology

Stage 2

dtlb_walk_ratio, DTLB Walk Ratio, metric

This metric measures the ratio of data TLB Walks to the total number of data TLB accesses. This gives an indication of the effectiveness of the data TLB accesses.

Units

This unit is expressed as per tlb access.

Formula

[DTLB_WALK](#) / [L1D_TLB](#)

Related telemetry artifacts**Events**[DTLB_WALK](#)[L1D_TLB](#)**Metric group**[Miss_Ratio](#)Other metric group: [DTLB_Effectiveness](#)**Methodology**

Stage 2

itlb_walk_ratio, ITLB Walk Ratio, metric

This metric measures the ratio of instruction TLB Walks to the total number of instruction TLB accesses. This gives an indication of the effectiveness of the instruction TLB accesses.

Units

This unit is expressed as per tlb access.

Formula
$$\text{ITLB_WALK} / \text{L1I_TLB}$$
Related telemetry artifacts**Events**[ITLB_WALK](#)[L1I_TLB](#)**Metric group**[Miss_Ratio](#)Other metric group: [ITLB_Effectiveness](#)**Methodology**

Stage 2

l1d_cache_miss_ratio, L1D Cache Miss Ratio, metric

This metric measures the ratio of level 1 data cache accesses missed to the total number of level 1 data cache accesses. This gives an indication of the effectiveness of the level 1 data cache.

Units

This unit is expressed as per cache access.

Formula
$$\text{L1D_CACHE_REFILL} / \text{L1D_CACHE}$$
Related telemetry artifacts**Events**[L1D_CACHE](#)[L1D_CACHE_REFILL](#)

Metric group[Miss_Ratio](#)Other metric group: [L1D_Cache_Effectiveness](#)**Methodology**

Stage 2

l1d_tlb_miss_ratio, L1 Data TLB Miss Ratio, metric

This metric measures the ratio of level 1 data TLB accesses missed to the total number of level 1 data TLB accesses. This gives an indication of the effectiveness of the level 1 data TLB.

Units

This unit is expressed as per tlb access.

Formula
$$\text{L1D_TLB_REFILL} / \text{L1D_TLB}$$
Related telemetry artifacts**Events**[L1D_TLB](#)[L1D_TLB_REFILL](#)**Metric group**[Miss_Ratio](#)Other metric group: [DTLB_Effectiveness](#)**Methodology**

Stage 2

l1i_cache_miss_ratio, L1I Cache Miss Ratio, metric

This metric measures the ratio of level 1 instruction cache accesses missed to the total number of level 1 instruction cache accesses. This gives an indication of the effectiveness of the level 1 instruction cache.

Units

This unit is expressed as per cache access.

Formula
$$\text{L1I_CACHE_REFILL} / \text{L1I_CACHE}$$
Related telemetry artifacts**Events**[L1I_CACHE](#)[L1I_CACHE_REFILL](#)**Metric group**[Miss_Ratio](#)Other metric group: [L1I_Cache_Effectiveness](#)

Methodology

Stage 2

I1i_tlb_miss_ratio, L1 Instruction TLB Miss Ratio, metric

This metric measures the ratio of level 1 instruction TLB accesses missed to the total number of level 1 instruction TLB accesses. This gives an indication of the effectiveness of the level 1 instruction TLB.

Units

This unit is expressed as per tlb access.

Formula
$$L1I_TLB_REFILL / L1I_TLB$$
Related telemetry artifacts**Events**[L1I_TLB](#)[L1I_TLB_REFILL](#)**Metric group**[Miss_Ratio](#)

Other metric group: [ITLB_Effectiveness](#)

Methodology

Stage 2

I2_tlb_miss_ratio, L2 Unified TLB Miss Ratio, metric

This metric measures the ratio of level 2 unified TLB accesses missed to the total number of level 2 unified TLB accesses. This gives an indication of the effectiveness of the level 2 TLB.

Units

This unit is expressed as per tlb access.

Formula
$$L2D_TLB_REFILL / L2D_TLB$$
Related telemetry artifacts**Events**[L2D_TLB](#)[L2D_TLB_REFILL](#)**Metric group**[Miss_Ratio](#)

Other metric group: [DTLB_Effectiveness](#)

Other metric group: [ITLB_Effectiveness](#)

Methodology

Stage 2

L2d_cache_miss_ratio, L2D Cache Miss Ratio, metric

This metric measures the ratio of level 2 cache data accesses missed to the total number of level 2 data cache accesses. This gives an indication of the effectiveness of data accesses in the level 2 cache, which is a unified cache that stores both data and instruction. Note that cache accesses in this cache are either data memory access or instruction fetch as this is a unified cache.

Units

This unit is expressed as per cache access.

Formula

$$\text{L2D_CACHE_REFILL} / \text{L2D_CACHE}$$

Related telemetry artifacts

Events

[L2D_CACHE](#)

[L2D_CACHE_REFILL](#)

Metric group

[Miss_Ratio](#)

Other metric group: [L2D_Cache_Effectiveness](#)

Methodology

Stage 2

L2i_cache_miss_ratio, L2I Cache Miss Ratio, metric

This metric measures the ratio of level 2 cache instruction accesses missed to the total number of level 2 cache instruction accesses. This gives an indication of the effectiveness of instruction accesses in the level 2 cache, which is a unified cache that stores both data and instruction. Note that cache accesses in this cache are either data memory access or instruction fetch as this is a unified cache.

Units

This unit is expressed as per cache access.

Formula

$$\text{L2I_CACHE_REFILL} / \text{L2I_CACHE}$$

Related telemetry artifacts

Events

[L2I_CACHE](#)

[L2I_CACHE_REFILL](#)

Metric group

[Miss_Ratio](#)

Other metric group: [L2I_Cache_Effectiveness](#)

Methodology

Stage 2

L3_cache_miss_ratio, L3 Cache Miss Ratio, metric

This metric measures the ratio of level 3 cache accesses missed to the total number of level 3 cache accesses. This gives an indication of the effectiveness of the level 3 cache, which is a unified cache that stores both data and instruction. Note that cache accesses in this cache are either data memory access or instruction fetch as this is a unified cache.

Units

This unit is expressed as per cache access.

Formula

$L3D_CACHE_REFILL / L3D_CACHE$

Related telemetry artifacts

Events

[L3D_CACHE](#)

[L3D_CACHE_REFILL](#)

Metric group

[Miss_Ratio](#)

Other metric group: [L3_Cache_Effectiveness](#)

Methodology

Stage 2

ll_cache_read_miss_ratio, LL Cache Read Miss Ratio, metric

This metric measures the ratio of last level cache read accesses missed to the total number of last level cache accesses. This gives an indication of the effectiveness of the last level cache for read traffic. Note that cache accesses in this cache are either data memory access or instruction fetch as this is a system level cache.

Units

This unit is expressed as per cache access.

Formula

$LL_CACHE_MISS_RD / LL_CACHE_RD$

Related telemetry artifacts

Events

[LL_CACHE_MISS_RD](#)

[LL_CACHE_RD](#)

Metric group

[Miss_Ratio](#)

Other metric group: [LL_Cache_Effectiveness](#)

Methodology

Stage 2

5.10 SVE_Effectiveness metrics for C1-Premium

SVE Effectiveness. This metric group contains metrics to evaluate the effectiveness of predicated SVE instruction execution on this processor.

Summary of metrics in SVE_Effectiveness:

- Total metrics: 4

Table 5-10: SVE_Effectiveness metrics summary

Metric	Name	Description
sve_predicate_empty_percentage	SVE Empty Predicate Percentage	This metric measures scalable vector operations with no active predicates as a percentage of sve...
sve_predicate_full_percentage	SVE Full Predicate Percentage	This metric measures scalable vector operations with all active predicates as a percentage of sve...
sve_predicate_partial_percentage	SVE Partial Predicate Percentage	This metric measures scalable vector operations with at least one active predicates as a...
sve_predicate_percentage	SVE Predicate Percentage	This metric measures scalable vector operations with predicates as a percentage of operations...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

sve_predicate_empty_percentage, SVE Empty Predicate Percentage, metric

This metric measures scalable vector operations with no active predicates as a percentage of sve predicated operations speculatively executed.

Units

This unit is expressed as percent of operations.

Formula

$$\text{SVE_PRED_EMPTY_SPEC} / \text{SVE_PRED_SPEC} * 100$$

Related telemetry artifacts

Events

[SVE_PRED_EMPTY_SPEC](#)

[SVE_PRED_SPEC](#)

Metric group

[SVE_Effectiveness](#)

Methodology

Stage 2

sve_predicate_full_percentage, SVE Full Predicate Percentage, metric

This metric measures scalable vector operations with all active predicates as a percentage of sve predicated operations speculatively executed.

Units

This unit is expressed as percent of operations.

Formula

$$\text{SVE_PRED_FULL_SPEC} / \text{SVE_PRED_SPEC} * 100$$

Related telemetry artifacts**Events**

SVE_PRED_FULL_SPEC
SVE_PRED_SPEC

Metric group

SVE_Effectiveness

Methodology

Stage 2

sve_predicate_partial_percentage, SVE Partial Predicate Percentage, metric

This metric measures scalable vector operations with at least one active predicates as a percentage of sve predicated operations speculatively executed.

Units

This unit is expressed as percent of operations.

Formula

$$\text{SVE_PRED_PARTIAL_SPEC} / \text{SVE_PRED_SPEC} * 100$$

Related telemetry artifacts**Events**

SVE_PRED_PARTIAL_SPEC
SVE_PRED_SPEC

Metric group

SVE_Effectiveness

Methodology

Stage 2

sve_predicate_percentage, SVE Predicate Percentage, metric

This metric measures scalable vector operations with predicates as a percentage of operations speculatively executed.

Units

This unit is expressed as percent of operations.

Formula

$$\text{SVE_PRED_SPEC} / \text{INST_SPEC} * 100$$

Related telemetry artifacts**Events**

INST_SPEC
SVE_PRED_SPEC

Metric group
SVE_Effectiveness

Methodology
Stage 2

5.11 FP_Arithmetic_Intensity metrics for C1-Premium

Floating Point Arithmetic Intensity. This metric group contains metrics to evaluate the effectiveness of floating point instruction execution on this processor.

Summary of metrics in FP_Arithmetic_Intensity:

- Total metrics: 3

Table 5-11: FP_Arithmetic_Intensity metrics summary

Metric	Name	Description
fp_ops_per_cycle	Floating Point Operations per Cycle	This metric measures floating point operations per cycle in any precision performed by any...
nonsve_fp_ops_per_cycle	Non-SVE Floating Point Operations per Cycle	This metric measures floating point operations per cycle in any precision performed by an...
sve_fp_ops_per_cycle	SVE Floating Point Operations per Cycle	This metric measures floating point operations per cycle in any precision performed by SVE...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

fp_ops_per_cycle, Floating Point Operations per Cycle, metric

This metric measures floating point operations per cycle in any precision performed by any instruction. Operations are counted by computation and by vector lanes, fused computations such as multiply-add count as twice per vector lane for example.

Units

This unit is expressed as operations per cycle.

Formula

$$(FP_SCALE_OPS_SPEC + FP_FIXED_OPS_SPEC) / (CPU_CYCLES - IMP_WFX_CLOCK_CYCLES)$$

Related telemetry artifacts

Events

CPU_CYCLES
FP_FIXED_OPS_SPEC
FP_SCALE_OPS_SPEC
IMP_WFX_CLOCK_CYCLES

Metric group

FP_Arithmetic_Intensity

Methodology

Stage 2

nonsve_fp_ops_per_cycle, Non-SVE Floating Point Operations per Cycle, metric

This metric measures floating point operations per cycle in any precision performed by an instruction that is not an SVE instruction. Operations are counted by computation and by vector lanes, fused computations such as multiply-add count as twice per vector lane for example.

Units

This unit is expressed as operations per cycle.

Formula

$$\text{FP_FIXED_OPS_SPEC} / (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES})$$

Related telemetry artifacts**Events**

CPU_CYCLES

FP_FIXED_OPS_SPEC

IMP_WFX_CLOCK_CYCLES

Metric group

FP_Arithmetic_Intensity

Methodology

Stage 2

sve_fp_ops_per_cycle, SVE Floating Point Operations per Cycle, metric

This metric measures floating point operations per cycle in any precision performed by SVE instructions. Operations are counted by computation and by vector lanes, fused computations such as multiply-add count as twice per vector lane for example.

Units

This unit is expressed as operations per cycle.

Formula

$$\text{FP_SCALE_OPS_SPEC} / (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES})$$

Related telemetry artifacts**Events**

CPU_CYCLES

FP_SCALE_OPS_SPEC

IMP_WFX_CLOCK_CYCLES

Metric group

FP_Arithmetic_Intensity

Methodology

Stage 2

5.12 FP_Precision_Mix metrics for C1-Premium

Floating Point Precision. This metric group contains metrics to evaluate the precision of floating point instruction execution on this processor.

Summary of metrics in FP_Precision_Mix:

- Total metrics: 3

Table 5-12: FP_Precision_Mix metrics summary

Metric	Name	Description
fp16_percentage	Half Precision Floating Point Percentage	This metric measures half-precision floating point operations as a percentage of operations...
fp32_percentage	Single Precision Floating Point Percentage	This metric measures single-precision floating point operations as a percentage of operations...
fp64_percentage	Double Precision Floating Point Percentage	This metric measures double-precision floating point operations as a percentage of operations...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

fp16_percentage, Half Precision Floating Point Percentage, metric

This metric measures half-precision floating point operations as a percentage of operations speculatively executed.

Units

This unit is expressed as percent of operations.

Formula

$$\text{FP_HP_SPEC} / \text{INST_SPEC} * 100$$

Related telemetry artifacts

Events

FP_HP_SPEC
INST_SPEC

Metric group

FP_Precision_Mix

Methodology

Stage 2

fp32_percentage, Single Precision Floating Point Percentage, metric

This metric measures single-precision floating point operations as a percentage of operations speculatively executed.

Units

This unit is expressed as percent of operations.

Formula

$$FP_SP_SPEC / INST_SPEC * 100$$

Related telemetry artifacts

Events

FP_SP_SPEC
INST_SPEC

Metric group

FP_Precision_Mix

Methodology

Stage 2

fp64_percentage, Double Precision Floating Point Percentage, metric

This metric measures double-precision floating point operations as a percentage of operations speculatively executed.

Units

This unit is expressed as percent of operations.

Formula

$$FP_DP_SPEC / INST_SPEC * 100$$

Related telemetry artifacts

Events

FP_DP_SPEC
INST_SPEC

Metric group

FP_Precision_Mix

Methodology

Stage 2

5.13 Branch_Effectiveness metrics for C1-Premium

Branch Effectiveness. This metric group contains metrics to evaluate the effectiveness of branch instruction execution on this processor.

Summary of metrics in Branch_Effectiveness:

- Total metrics: 5

Table 5-13: Branch_Effectiveness metrics summary

Metric	Name	Description
branch_direct_ratio	Branch Direct Ratio	This metric measures the ratio of direct branches retired to the total number of branches...

Metric	Name	Description
branch_indirect_ratio	Branch Indirect Ratio	This metric measures the ratio of indirect branches retired, including function returns, to the...
branch_misprediction_ratio	Branch Misprediction Ratio	This metric measures the ratio of branches mispredicted to the total number of branches...
branch_mpki	Branch MPKI	This metric measures the number of branch mispredictions per thousand instructions executed.
branch_return_ratio	Branch Return Ratio	This metric measures the ratio of branches retired that are function returns to the total number...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

branch_direct_ratio, Branch Direct Ratio, metric

This metric measures the ratio of direct branches retired to the total number of branches architecturally executed.

Units

This unit is expressed as per branch.

Formula

[BR_IMMED_RETIRED](#) / [BR_RETIRED](#)

Related telemetry artifacts

Events

[BR_IMMED_RETIRED](#)

[BR_RETIRED](#)

Metric group

[Branch_Effectiveness](#)

Methodology

Stage 2

branch_indirect_ratio, Branch Indirect Ratio, metric

This metric measures the ratio of indirect branches retired, including function returns, to the total number of branches architecturally executed.

Units

This unit is expressed as per branch.

Formula

[BR_IND_RETIRED](#) / [BR_RETIRED](#)

Related telemetry artifacts

Events

[BR_IND_RETIRED](#)

[BR_RETIRED](#)

Metric group[Branch_Effectiveness](#)**Methodology**

Stage 2

branch_misprediction_ratio, Branch Misprediction Ratio, metric**

This metric measures the ratio of branches mispredicted to the total number of branches architecturally executed. This gives an indication of the effectiveness of the branch prediction unit.

Units

This unit is expressed as per branch.

Formula
$$\text{BR_MIS_PRED_RETIRED} / \text{BR_RETIRED}$$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts**Events**[BR_MIS_PRED_RETIRED](#)[BR_RETIRED](#)**Metric group**[Branch_Effectiveness](#)Other metric group: [Miss_Ratio](#)**Methodology**

Stage 2

branch_mпки, Branch MPKI, metric**

This metric measures the number of branch mispredictions per thousand instructions executed.

Units

This unit is expressed as mпки.

Formula
$$\text{BR_MIS_PRED_RETIRED} / \text{INST_RETIRED} * 1000$$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts**Events**[BR_MIS_PRED_RETIRED](#)[INST_RETIRED](#)**Metric group**[Branch_Effectiveness](#)Other metric group: [MPKI](#)

Methodology
Stage 2

branch_return_ratio, Branch Return Ratio, metric

This metric measures the ratio of branches retired that are function returns to the total number of branches architecturally executed.

Units

This unit is expressed as per branch.

Formula

$$\text{BR_RETURN_RETIRED} / \text{BR_RETIRED}$$

Related telemetry artifacts

Events

BR_RETIRED
BR_RETURN_RETIRED

Metric group

Branch_Effectiveness

Methodology

Stage 2

5.14 ITLB_Effectiveness metrics for C1-Premium

Instruction TLB Effectiveness. This metric group contains metrics to evaluate the effectiveness of instruction TLB on this processor.

Summary of metrics in ITLB_Effectiveness:

- Total metrics: 12

Table 5-14: ITLB_Effectiveness metrics summary

Metric	Name	Description
itlb_mпки	ITLB MPKI	This metric measures the number of instruction TLB Walks per thousand instructions executed.
itlb_walk_average_depth	ITLB Walk Average Depth of Accesses	This metric measures the average depth of the instruction TLB walks for an instruction TLB refill
itlb_walk_average_latency	ITLB Walk Average Latency	This metric measures the average latency of instruction TLB walks in CPU cycles
itlb_walk_block_ratio	ITLB Walk Block Ratio	This metric measures the ratio of instruction TLB Walks that returned a block to the total number...
itlb_walk_large_ratio	ITLB Walk Large Page Ratio	This metric measures the ratio of instruction TLB Walks that returned large page to the total...
itlb_walk_page_ratio	ITLB Walk Page Ratio	This metric measures the ratio of instruction TLB Walks that returned a page to the total number...

Metric	Name	Description
itlb_walk_ratio	ITLB Walk Ratio	This metric measures the ratio of instruction TLB Walks to the total number of instruction TLB...
itlb_walk_small_ratio	ITLB Walk Small Page Ratio	This metric measures the ratio of instruction TLB Walks that returned small page to the total...
l1i_tlb_miss_ratio	L1 Instruction TLB Miss Ratio	This metric measures the ratio of level 1 instruction TLB accesses missed to the total number of...
l1i_tlb_mпки	L1 Instruction TLB MPKI	This metric measures the number of level 1 instruction TLB accesses missed per thousand...
l2_tlb_miss_ratio	L2 Unified TLB Miss Ratio	This metric measures the ratio of level 2 unified TLB accesses missed to the total number of...
l2_tlb_mпки	L2 Unified TLB MPKI	This metric measures the number of level 2 unified TLB accesses missed per thousand instructions...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

itlb_mпки, ITLB MPKI, metric**

This metric measures the number of instruction TLB Walks per thousand instructions executed.

Units

This unit is expressed as mпки.

Formula

$$\text{ITLB_WALK} / \text{INST_RETIRED} * 1000$$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts

Events

[INST_RETIRED](#)
[ITLB_WALK](#)

Metric group

[ITLB_Effectiveness](#)
Other metric group: [MPKI](#)

Methodology

Stage 2

itlb_walk_average_depth, ITLB Walk Average Depth of Accesses, metric

This metric measures the average depth of the instruction TLB walks for an instruction TLB refill

Units

This unit is expressed as page accesses.

Formula

$$\text{ITLB_STEP} / \text{ITLB_WALK}$$

Related telemetry artifacts**Events**

[ITLB_STEP](#)
[ITLB_WALK](#)

Metric group

[ITLB_Effectiveness](#)

Methodology

Stage 2

itlb_walk_average_latency, ITLB Walk Average Latency, metric

This metric measures the average latency of instruction TLB walks in CPU cycles

Units

This unit is expressed in cycles..

Formula

[ITLB_WALK_PERCYC](#) / [ITLB_WALK](#)

Related telemetry artifacts**Events**

[ITLB_WALK](#)
[ITLB_WALK_PERCYC](#)

Metric group

[ITLB_Effectiveness](#)

Other metric group: [Average_Latency](#)

Methodology

Stage 2

itlb_walk_block_ratio, ITLB Walk Block Ratio, metric

This metric measures the ratio of instruction TLB Walks that returned a block to the total number of instruction TLB accesses. Block size is any memory block larger than the page granule size set by the system.

Units

This unit is expressed as per tlb access.

Formula

[ITLB_WALK_BLOCK](#) / [L1I_TLB](#)

Related telemetry artifacts**Events**

[ITLB_WALK_BLOCK](#)
[L1I_TLB](#)

Metric group

[ITLB_Effectiveness](#)

Methodology

Stage 2

itlb_walk_large_ratio, ITLB Walk Large Page Ratio, metric

This metric measures the ratio of instruction TLB Walks that returned large page to the total number of instruction TLB accesses.

Units

This unit is expressed as per tlb access.

Formula
$$\text{ITLB_WALK_LARGE} / \text{L1I_TLB}$$
Related telemetry artifacts**Events**

[ITLB_WALK_LARGE](#)
[L1I_TLB](#)

Metric group

[ITLB_Effectiveness](#)

Methodology

Stage 2

itlb_walk_page_ratio, ITLB Walk Page Ratio, metric

This metric measures the ratio of instruction TLB Walks that returned a page to the total number of instruction TLB accesses. Page size is determined by the page granule size set by the system.

Units

This unit is expressed as per tlb access.

Formula
$$\text{ITLB_WALK_PAGE} / \text{L1I_TLB}$$
Related telemetry artifacts**Events**

[ITLB_WALK_PAGE](#)
[L1I_TLB](#)

Metric group

[ITLB_Effectiveness](#)

Methodology

Stage 2

itlb_walk_ratio, ITLB Walk Ratio, metric**

This metric measures the ratio of instruction TLB Walks to the total number of instruction TLB accesses. This gives an indication of the effectiveness of the instruction TLB accesses.

Units

This unit is expressed as per tlb access.

Formula
$$\text{ITLB_WALK} / \text{L1I_TLB}$$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts**Events**[ITLB_WALK](#)[L1I_TLB](#)**Metric group**[ITLB_Effectiveness](#)Other metric group: [Miss_Ratio](#)**Methodology**

Stage 2

itlb_walk_small_ratio, ITLB Walk Small Page Ratio, metric

This metric measures the ratio of instruction TLB Walks that returned small page to the total number of instruction TLB accesses.

Units

This unit is expressed as per tlb access.

Formula
$$\text{ITLB_WALK_SMALL} / \text{L1I_TLB}$$
Related telemetry artifacts**Events**[ITLB_WALK_SMALL](#)[L1I_TLB](#)**Metric group**[ITLB_Effectiveness](#)**Methodology**

Stage 2

l1i_tlb_miss_ratio, L1 Instruction TLB Miss Ratio, metric**

This metric measures the ratio of level 1 instruction TLB accesses missed to the total number of level 1 instruction TLB accesses. This gives an indication of the effectiveness of the level 1 instruction TLB.

Units

This unit is expressed as per tlb access.

Formula
$$\text{L1I_TLB_REFILL} / \text{L1I_TLB}$$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts**Events**

[L1I_TLB](#)
[L1I_TLB_REFILL](#)

Metric group

[ITLB_Effectiveness](#)
 Other metric group: [Miss_Ratio](#)

Methodology

Stage 2

`l1i_tlb_mпки`, L1 Instruction TLB MPKI, metric**

This metric measures the number of level 1 instruction TLB accesses missed per thousand instructions executed.

Units

This unit is expressed as mпки.

Formula

$$\frac{\text{L1I_TLB_REFILL}}{\text{INST_RETIRED}} * 1000$$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts**Events**

[INST_RETIRED](#)
[L1I_TLB_REFILL](#)

Metric group

[ITLB_Effectiveness](#)
 Other metric group: [MPKI](#)

Methodology

Stage 2

`l2_tlb_miss_ratio`, L2 Unified TLB Miss Ratio, metric**

This metric measures the ratio of level 2 unified TLB accesses missed to the total number of level 2 unified TLB accesses. This gives an indication of the effectiveness of the level 2 TLB.

Units

This unit is expressed as per tlb access.

Formula

$$\frac{\text{L2D_TLB_REFILL}}{\text{L2D_TLB}}$$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts**Events**

[L2D_TLB](#)
[L2D_TLB_REFILL](#)

Metric group

[ITLB_Effectiveness](#)
 Other metric group: [DTLB_Effectiveness](#)
 Other metric group: [Miss_Ratio](#)

Methodology

Stage 2

[l2_tlb_mпки](#), L2 Unified TLB MPKI, metric**

This metric measures the number of level 2 unified TLB accesses missed per thousand instructions executed.

Units

This unit is expressed as mпки.

Formula

[L2D_TLB_REFILL](#) / [INST_RETIRED](#) * 1000

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts**Events**

[INST_RETIRED](#)
[L2D_TLB_REFILL](#)

Metric group

[ITLB_Effectiveness](#)
 Other metric group: [DTLB_Effectiveness](#)
 Other metric group: [MPKI](#)

Methodology

Stage 2

5.15 [DTLB_Effectiveness](#) metrics for C1-Premium

Data TLB Effectiveness. This metric group contains metrics to evaluate the effectiveness of data TLB on this processor.

Summary of metrics in [DTLB_Effectiveness](#):

- Total metrics: 12

Table 5-15: DTLB_Effectiveness metrics summary

Metric	Name	Description
dtlb_mpki	DTLB MPKI	This metric measures the number of data TLB Walks per thousand instructions executed.
dtlb_walk_average_depth	DTLB Walk Average Depth of Accesses	This metric measures the average depth of the data TLB walks for a data TLB refill
dtlb_walk_average_latency	DTLB Walk Average Latency	This metric measures the average latency of data TLB walks in CPU cycles
dtlb_walk_block_ratio	DTLB Walk Block Ratio	This metric measures the ratio of data TLB Walks that returned a block to the total number of...
dtlb_walk_large_ratio	DTLB Walk Large Page Ratio	This metric measures the ratio of data TLB Walks that returned large page to the total number of...
dtlb_walk_page_ratio	DTLB Walk Page Ratio	This metric measures the ratio of data TLB Walks that returned a page to the total number of data...
dtlb_walk_ratio	DTLB Walk Ratio	This metric measures the ratio of data TLB Walks to the total number of data TLB accesses. This...
dtlb_walk_small_ratio	DTLB Walk Small Page Ratio	This metric measures the ratio of data TLB Walks that returned small page to the total number of...
l1d_tlb_miss_ratio	L1 Data TLB Miss Ratio	This metric measures the ratio of level 1 data TLB accesses missed to the total number of level 1...
l1d_tlb_mpki	L1 Data TLB MPKI	This metric measures the number of level 1 data TLB accesses missed per thousand instructions...
l2_tlb_miss_ratio	L2 Unified TLB Miss Ratio	This metric measures the ratio of level 2 unified TLB accesses missed to the total number of...
l2_tlb_mpki	L2 Unified TLB MPKI	This metric measures the number of level 2 unified TLB accesses missed per thousand instructions...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

dtlb_mpki, DTLB MPKI, metric**

This metric measures the number of data TLB Walks per thousand instructions executed.

Units

This unit is expressed as mpki.

Formula

$$\text{DTLB_WALK} / \text{INST_RETIRED} * 1000$$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts

Events

[DTLB_WALK](#)
[INST_RETIRED](#)

Metric group

[DTLB_Effectiveness](#)
Other metric group: [MPKI](#)

Methodology

Stage 2

dtlb_walk_average_depth, DTLB Walk Average Depth of Accesses, metric

This metric measures the average depth of the data TLB walks for a data TLB refill

Units

This unit is expressed as page table accesses.

Formula
$$\text{DTLB_STEP} / \text{DTLB_WALK}$$
Related telemetry artifacts**Events**[DTLB_STEP](#)[DTLB_WALK](#)**Metric group**[DTLB_Effectiveness](#)**Methodology**

Stage 2

dtlb_walk_average_latency, DTLB Walk Average Latency, metric

This metric measures the average latency of data TLB walks in CPU cycles

Units

This unit is expressed in cycles..

Formula
$$\text{DTLB_WALK_PERCYC} / \text{DTLB_WALK}$$
Related telemetry artifacts**Events**[DTLB_WALK](#)[DTLB_WALK_PERCYC](#)**Metric group**[DTLB_Effectiveness](#)Other metric group: [Average_Latency](#)**Methodology**

Stage 2

dtlb_walk_block_ratio, DTLB Walk Block Ratio, metric

This metric measures the ratio of data TLB Walks that returned a block to the total number of data TLB accesses. Block size is any memory block larger than the page granule size set by the system.

Units

This unit is expressed as per tlb access.

Formula
$$\text{DTLB_WALK_BLOCK} / \text{L1D_TLB}$$
Related telemetry artifacts**Events**

[DTLB_WALK_BLOCK](#)
[L1D_TLB](#)

Metric group

[DTLB_Effectiveness](#)

Methodology

Stage 2

dtlb_walk_large_ratio, DTLB Walk Large Page Ratio, metric

This metric measures the ratio of data TLB Walks that returned large page to the total number of data TLB accesses.

Units

This unit is expressed as per tlb access.

Formula
$$\text{DTLB_WALK_LARGE} / \text{L1D_TLB}$$
Related telemetry artifacts**Events**

[DTLB_WALK_LARGE](#)
[L1D_TLB](#)

Metric group

[DTLB_Effectiveness](#)

Methodology

Stage 2

dtlb_walk_page_ratio, DTLB Walk Page Ratio, metric

This metric measures the ratio of data TLB Walks that returned a page to the total number of data TLB accesses. Page size is determined by the page granule size set by the system.

Units

This unit is expressed as per tlb access.

Formula
$$\text{DTLB_WALK_PAGE} / \text{L1D_TLB}$$
Related telemetry artifacts**Events**

[DTLB_WALK_PAGE](#)
[L1D_TLB](#)

Metric group[DTLB_Effectiveness](#)**Methodology**

Stage 2

dtlb_walk_ratio, DTLB Walk Ratio, metric**

This metric measures the ratio of data TLB Walks to the total number of data TLB accesses. This gives an indication of the effectiveness of the data TLB accesses.

Units

This unit is expressed as per tlb access.

Formula
$$\text{DTLB_WALK} / \text{L1D_TLB}$$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts**Events**[DTLB_WALK](#)[L1D_TLB](#)**Metric group**[DTLB_Effectiveness](#)Other metric group: [Miss_Ratio](#)**Methodology**

Stage 2

dtlb_walk_small_ratio, DTLB Walk Small Page Ratio, metric

This metric measures the ratio of data TLB Walks that returned small page to the total number of data TLB accesses.

Units

This unit is expressed as per tlb access.

Formula
$$\text{DTLB_WALK_SMALL} / \text{L1D_TLB}$$
Related telemetry artifacts**Events**[DTLB_WALK_SMALL](#)[L1D_TLB](#)**Metric group**[DTLB_Effectiveness](#)**Methodology**

Stage 2

l1d_tlb_miss_ratio, L1 Data TLB Miss Ratio, metric**

This metric measures the ratio of level 1 data TLB accesses missed to the total number of level 1 data TLB accesses. This gives an indication of the effectiveness of the level 1 data TLB.

Units

This unit is expressed as per tlb access.

Formula

[L1D_TLB_REFILL](#) / [L1D_TLB](#)

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts**Events**

[L1D_TLB](#)

[L1D_TLB_REFILL](#)

Metric group

[DTLB_Effectiveness](#)

Other metric group: [Miss_Ratio](#)

Methodology

Stage 2

l1d_tlb_mпки, L1 Data TLB MPKI, metric**

This metric measures the number of level 1 data TLB accesses missed per thousand instructions executed.

Units

This unit is expressed as mпки.

Formula

[L1D_TLB_REFILL](#) / [INST_RETIRED](#) * 1000

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts**Events**

[INST_RETIRED](#)

[L1D_TLB_REFILL](#)

Metric group

[DTLB_Effectiveness](#)

Other metric group: [MPKI](#)

Methodology

Stage 2

`l2_tlb_miss_ratio*`, L2 Unified TLB Miss Ratio, metric**

This metric measures the ratio of level 2 unified TLB accesses missed to the total number of level 2 unified TLB accesses. This gives an indication of the effectiveness of the level 2 TLB.

Units

This unit is expressed as per tlb access.

Formula

`L2D_TLB_REFILL` / `L2D_TLB`

*** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts**Events**

`L2D_TLB`

`L2D_TLB_REFILL`

Metric group

`DTLB_Effectiveness`

Other metric group: `ITLB_Effectiveness`

Other metric group: `Miss_Ratio`

Methodology

Stage 2

`l2_tlb_mпки*`, L2 Unified TLB MPKI, metric**

This metric measures the number of level 2 unified TLB accesses missed per thousand instructions executed.

Units

This unit is expressed as mpki.

Formula

`L2D_TLB_REFILL` / `INST_RETIRED` * 1000

*** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts**Events**

`INST_RETIRED`

`L2D_TLB_REFILL`

Metric group

`DTLB_Effectiveness`

Other metric group: `ITLB_Effectiveness`

Other metric group: `MPKI`

Methodology

Stage 2

5.16 L1I_Cache_Effectiveness metrics for C1-Premium

L1 Instruction Cache Effectiveness. This metric group contains metrics to evaluate the effectiveness of L1 Instruction cache on this processor.

Summary of metrics in L1I_Cache_Effectiveness:

- Total metrics: 2

Table 5-16: L1I_Cache_Effectiveness metrics summary

Metric	Name	Description
l1i_cache_miss_ratio	L1I Cache Miss Ratio	This metric measures the ratio of level 1 instruction cache accesses missed to the total number...
l1i_cache_mpki	L1I Cache MPKI	This metric measures the number of level 1 instruction cache accesses missed per thousand...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

[l1i_cache_miss_ratio](#)** , L1I Cache Miss Ratio, metric

This metric measures the ratio of level 1 instruction cache accesses missed to the total number of level 1 instruction cache accesses. This gives an indication of the effectiveness of the level 1 instruction cache.

Units

This unit is expressed as per cache access.

Formula

$$\frac{\text{L1I_CACHE_REFILL}}{\text{L1I_CACHE}}$$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts

Events

[L1I_CACHE](#)
[L1I_CACHE_REFILL](#)

Metric group

[L1I_Cache_Effectiveness](#)
Other metric group: [Miss_Ratio](#)

Methodology

Stage 2

[l1i_cache_mpki](#)** , L1I Cache MPKI, metric

This metric measures the number of level 1 instruction cache accesses missed per thousand instructions executed.

Units

This unit is expressed as mpki.

Formula

$$L1I_CACHE_REFILL / INST_RETIRED * 1000$$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts

Events

[INST_RETIRED](#)
[L1I_CACHE_REFILL](#)

Metric group

[L1I_Cache_Effectiveness](#)
Other metric group: [MPKI](#)

Methodology

Stage 2

5.17 L1D_Cache_Effectiveness metrics for C1-Premium

L1 Data Cache Effectiveness. This metric group contains metrics to evaluate the effectiveness of L1 Data Cache on this processor.

Summary of metrics in L1D_Cache_Effectiveness:

- Total metrics: 3

Table 5-17: L1D_Cache_Effectiveness metrics summary

Metric	Name	Description
l1d_cache_demand_mпки	L1D Cache Demand MPKI	This metric measures the number of level 1 data cache demand accesses missed per thousand...
l1d_cache_miss_ratio	L1D Cache Miss Ratio	This metric measures the ratio of level 1 data cache accesses missed to the total number of level...
l1d_cache_mпки	L1D Cache MPKI	This metric measures the number of level 1 data cache accesses missed per thousand instructions...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

[l1d_cache_demand_mпки](#)**, L1D Cache Demand MPKI, metric

This metric measures the number of level 1 data cache demand accesses missed per thousand instructions executed.

Units

This unit is expressed as mpki.

Formula

$$(\text{L1D_CACHE_REFILL_RD} + \text{L1D_CACHE_REFILL_WR}) / \text{INST_RETIRED} * 1000$$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts**Events**

[INST_RETIRED](#)
[L1D_CACHE_REFILL_RD](#)
[L1D_CACHE_REFILL_WR](#)

Metric group

[L1D_Cache_Effectiveness](#)
 Other metric group: [MPKI](#)

Methodology

Stage 2

l1d_cache_miss_ratio, L1D Cache Miss Ratio, metric**

This metric measures the ratio of level 1 data cache accesses missed to the total number of level 1 data cache accesses. This gives an indication of the effectiveness of the level 1 data cache.

Units

This unit is expressed as per cache access.

Formula

$$\text{L1D_CACHE_REFILL} / \text{L1D_CACHE}$$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts**Events**

[L1D_CACHE](#)
[L1D_CACHE_REFILL](#)

Metric group

[L1D_Cache_Effectiveness](#)
 Other metric group: [Miss_Ratio](#)

Methodology

Stage 2

l1d_cache_mпки, L1D Cache MPKI, metric**

This metric measures the number of level 1 data cache accesses missed per thousand instructions executed.

Units

This unit is expressed as mпки.

Formula

$$\text{L1D_CACHE_REFILL} / \text{INST_RETIRED} * 1000$$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts

Events

[INST_RETIRED](#)
[L1D_CACHE_REFILL](#)

Metric group

[L1D_Cache_Effectiveness](#)
Other metric group: [MPKI](#)

Methodology

Stage 2

5.18 L2D_Cache_Effectiveness metrics for C1-Premium

L2D Data Unified Cache Effectiveness. This metric group contains metrics to evaluate the effectiveness of data access in L2 Unified Cache on this processor.

Summary of metrics in L2D_Cache_Effectiveness:

- Total metrics: 3

Table 5-18: L2D_Cache_Effectiveness metrics summary

Metric	Name	Description
l2d_cache_demand_mпки	L2D Cache Demand MPKI	This metric measures the number of level 2 unified cache data demand accesses missed per thousand...
l2d_cache_miss_ratio	L2D Cache Miss Ratio	This metric measures the ratio of level 2 cache data accesses missed to the total number of level...
l2d_cache_mпки	L2D Cache MPKI	This metric measures the number of level 2 unified cache data accesses missed per thousand...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

[l2d_cache_demand_mпки](#)**, L2D Cache Demand MPKI, metric

This metric measures the number of level 2 unified cache data demand accesses missed per thousand instructions executed. Note that cache accesses in this cache are either data memory access or instruction fetch as this is a unified cache.

Units

This unit is expressed as mпки.

Formula

$$(\text{L2D_CACHE_REFILL_RD} + \text{L2D_CACHE_REFILL_WR}) / \text{INST_RETIRED} * 1000$$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts

Events

[INST_RETIRED](#)
[L2D_CACHE_REFILL_RD](#)
[L2D_CACHE_REFILL_WR](#)

Metric group

[L2D_Cache_Effectiveness](#)
 Other metric group: [MPKI](#)

Methodology

Stage 2

l2d_cache_miss_ratio**, L2D Cache Miss Ratio, metric

This metric measures the ratio of level 2 cache data accesses missed to the total number of level 2 data cache accesses. This gives an indication of the effectiveness of data accesses in the level 2 cache, which is a unified cache that stores both data and instruction. Note that cache accesses in this cache are either data memory access or instruction fetch as this is a unified cache.

Units

This unit is expressed as per cache access.

Formula

$\frac{\text{L2D_CACHE_REFILL}}{\text{L2D_CACHE}}$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts

Events

[L2D_CACHE](#)
[L2D_CACHE_REFILL](#)

Metric group

[L2D_Cache_Effectiveness](#)
 Other metric group: [Miss_Ratio](#)

Methodology

Stage 2

l2d_cache_mпки**, L2D Cache MPKI, metric

This metric measures the number of level 2 unified cache data accesses missed per thousand instructions executed. Note that cache accesses in this cache are either data memory access or instruction fetch as this is a unified cache.

Units

This unit is expressed as mпки.

Formula

$$\text{L2D_CACHE_REFILL} / \text{INST_RETIRED} * 1000$$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts

Events

[INST_RETIRED](#)
[L2D_CACHE_REFILL](#)

Metric group

[L2D_Cache_Effectiveness](#)
Other metric group: [MPKI](#)

Methodology

Stage 2

5.19 L2I_Cache_Effectiveness metrics for C1-Premium

L2 Instruction Unified Cache Effectiveness. This metric group contains metrics to evaluate the effectiveness of instruction access in L2 Unified Cache on this processor.

Summary of metrics in L2I_Cache_Effectiveness:

- Total metrics: 2

Table 5-19: L2I_Cache_Effectiveness metrics summary

Metric	Name	Description
l2i_cache_miss_ratio	L2I Cache Miss Ratio	This metric measures the ratio of level 2 cache instruction accesses missed to the total number...
l2i_cache_mпки	L2I Cache MPKI	This metric measures the number of level 2 unified cache instruction accesses missed per thousand...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

[l2i_cache_miss_ratio](#)**, **L2I Cache Miss Ratio**, **metric**

This metric measures the ratio of level 2 cache instruction accesses missed to the total number of level 2 cache instruction accesses. This gives an indication of the effectiveness of instruction accesses in the level 2 cache, which is a unified cache that stores both data and instruction. Note that cache accesses in this cache are either data memory access or instruction fetch as this is a unified cache.

Units

This unit is expressed as per cache access.

Formula

$$\text{L2I_CACHE_REFILL} / \text{L2I_CACHE}$$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts

Events

[L2I_CACHE](#)

[L2I_CACHE_REFILL](#)

Metric group

[L2I_Cache_Effectiveness](#)

Other metric group: [Miss_Ratio](#)

Methodology

Stage 2

`l2i_cache_mпки`, L2I Cache MPKI, metric**

This metric measures the number of level 2 unified cache instruction accesses missed per thousand instructions executed. Note that cache accesses in this cache are either data memory access or instruction fetch as this is a unified cache.

Units

This unit is expressed as mпки.

Formula

$\text{L2I_CACHE_REFILL} / \text{INST_RETIRED} * 1000$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts

Events

[INST_RETIRED](#)

[L2I_CACHE_REFILL](#)

Metric group

[L2I_Cache_Effectiveness](#)

Other metric group: [MPKI](#)

Methodology

Stage 2

5.20 L3_Cache_Effectiveness metrics for C1-Premium

L3 Unified Cache Effectiveness. This metric group contains metrics to evaluate the effectiveness of L3 Unified Cache on this processor.

Summary of metrics in L3_Cache_Effectiveness:

- Total metrics: 2

Table 5-20: L3_Cache_Effectiveness metrics summary

Metric	Name	Description
l3_cache_miss_ratio	L3 Cache Miss Ratio	This metric measures the ratio of level 3 cache accesses missed to the total number of level 3...
l3_cache_mпки	L3 Cache MPKI	This metric measures the number of level 3 unified cache accesses missed per thousand...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

[l3_cache_miss_ratio](#)**, L3 Cache Miss Ratio, metric

This metric measures the ratio of level 3 cache accesses missed to the total number of level 3 cache accesses. This gives an indication of the effectiveness of the level 3 cache, which is a unified cache that stores both data and instruction. Note that cache accesses in this cache are either data memory access or instruction fetch as this is a unified cache.

Units

This unit is expressed as per cache access.

Formula

[L3D_CACHE_REFILL](#) / [L3D_CACHE](#)

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts

Events

[L3D_CACHE](#)

[L3D_CACHE_REFILL](#)

Metric group

[L3_Cache_Effectiveness](#)

Other metric group: [Miss_Ratio](#)

Methodology

Stage 2

[l3_cache_mпки](#)**, L3 Cache MPKI, metric

This metric measures the number of level 3 unified cache accesses missed per thousand instructions executed. Note that cache accesses in this cache are either data memory access or instruction fetch as this is a unified cache.

Units

This unit is expressed as mпки.

Formula

[L3D_CACHE_REFILL](#) / [INST_RETIRED](#) * 1000

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts

Events

[INST_RETIRED](#)
[L3D_CACHE_REFILL](#)

Metric group

[L3_Cache_Effectiveness](#)
Other metric group: [MPKI](#)

Methodology

Stage 2

5.21 LL_Cache_Effectiveness metrics for C1-Premium

Last Level Cache Effectiveness. This metric group contains metrics to evaluate the effectiveness of Last Level Cache on this processor.

Summary of metrics in LL_Cache_Effectiveness:

- Total metrics: 3

Table 5-21: LL_Cache_Effectiveness metrics summary

Metric	Name	Description
ll_cache_read_hit_ratio	LL Cache Read Hit Ratio	This metric measures the ratio of last level cache read accesses hit in the cache to the total...
ll_cache_read_miss_ratio	LL Cache Read Miss Ratio	This metric measures the ratio of last level cache read accesses missed to the total number of...
ll_cache_read_mпки	LL Cache Read MPKI	This metric measures the number of last level cache read accesses missed per thousand...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

[ll_cache_read_hit_ratio](#), LL Cache Read Hit Ratio, metric

This metric measures the ratio of last level cache read accesses hit in the cache to the total number of last level cache accesses. This gives an indication of the effectiveness of the last level cache for read traffic. Note that cache accesses in this cache are either data memory access or instruction fetch as this is a system level cache.

Units

This unit is expressed as per cache access.

Formula

$$(\text{LL_CACHE_RD} - \text{LL_CACHE_MISS_RD}) / \text{LL_CACHE_RD}$$

Related telemetry artifacts

Events

[LL_CACHE_MISS_RD](#)

[LL_CACHE_RD](#)**Metric group**[LL_Cache_Effectiveness](#)**Methodology**

Stage 2

ll_cache_read_miss_ratio, LL Cache Read Miss Ratio, metric**

This metric measures the ratio of last level cache read accesses missed to the total number of last level cache accesses. This gives an indication of the effectiveness of the last level cache for read traffic. Note that cache accesses in this cache are either data memory access or instruction fetch as this is a system level cache.

Units

This unit is expressed as per cache access.

Formula

$$\text{LL_CACHE_MISS_RD} / \text{LL_CACHE_RD}$$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts**Events**[LL_CACHE_MISS_RD](#)[LL_CACHE_RD](#)**Metric group**[LL_Cache_Effectiveness](#)Other metric group: [Miss_Ratio](#)**Methodology**

Stage 2

ll_cache_read_mpk*, LL Cache Read MPKI, metric**

This metric measures the number of last level cache read accesses missed per thousand instructions executed.

Units

This unit is expressed as mpki.

Formula

$$\text{LL_CACHE_MISS_RD} / \text{INST_RETIRED} * 1000$$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts**Events**[INST_RETIRED](#)[LL_CACHE_MISS_RD](#)

Metric group

[LL_Cache_Effectiveness](#)

Other metric group: [MPKI](#)

Methodology

Stage 2

5.22 Rename_Effectiveness metrics for C1-Premium

Register Rename Effectiveness. This metric group provides relative stall cycles for register renaming by register type.

Summary of metrics in Rename_Effectiveness:

- Total metrics: 4

Table 5-22: Rename_Effectiveness metrics summary

Metric	Name	Description
rename_stall_flags_ratio	Flag Register Rename Ratio	This metric measures the percentage of rename stall cycles due to flag register rename availability
rename_stall_int_ratio	Integer Register Rename Ratio	This metric measures the percentage of rename stall cycles due to GPR register rename availability
rename_stall_pred_ratio	Predicate Register Rename Ratio	This metric measures the percentage of rename stall cycles due to predicate register rename...
rename_stall_vec_ratio	Vector Register Rename Ratio	This metric measures the percentage of rename stall cycles due to vector register rename...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

rename_stall_flags_ratio, Flag Register Rename Ratio, metric

This metric measures the percentage of rename stall cycles due to flag register rename availability

Units

This unit is expressed as cycles per rename stall cycle.

Formula

$$\frac{\text{IMP_STALL_BACKEND_RENAME_FRF}}{(\text{IMP_STALL_BACKEND_RENAME_VRF} + \text{IMP_STALL_BACKEND_RENAME_FRF} + \text{IMP_STALL_BACKEND_RENAME_GRF} + \text{IMP_STALL_BACKEND_RENAME_PDRF})}$$

Related telemetry artifacts

Events

- [IMP_STALL_BACKEND_RENAME_FRF](#)
- [IMP_STALL_BACKEND_RENAME_GRF](#)
- [IMP_STALL_BACKEND_RENAME_PDRF](#)
- [IMP_STALL_BACKEND_RENAME_VRF](#)

Metric group

Rename_Effectiveness

Methodology

Stage 2

rename_stall_int_ratio, Integer Register Rename Ratio, metric

This metric measures the percentage of rename stall cycles due to GPR register rename availability

Units

This unit is expressed as cycles per rename stall cycle.

Formula

$$\frac{\text{IMP_STALL_BACKEND_RENAME_GRF}}{(\text{IMP_STALL_BACKEND_RENAME_VRF} + \text{IMP_STALL_BACKEND_RENAME_FRF} + \text{IMP_STALL_BACKEND_RENAME_GRF} + \text{IMP_STALL_BACKEND_RENAME_PDRF})}$$
Related telemetry artifacts**Events**

IMP_STALL_BACKEND_RENAME_FRF
IMP_STALL_BACKEND_RENAME_GRF
IMP_STALL_BACKEND_RENAME_PDRF
IMP_STALL_BACKEND_RENAME_VRF

Metric group

Rename_Effectiveness

Methodology

Stage 2

rename_stall_pred_ratio, Predicate Register Rename Ratio, metric

This metric measures the percentage of rename stall cycles due to predicate register rename availability

Units

This unit is expressed as cycles per rename stall cycle.

Formula

$$\frac{\text{IMP_STALL_BACKEND_RENAME_PDRF}}{(\text{IMP_STALL_BACKEND_RENAME_VRF} + \text{IMP_STALL_BACKEND_RENAME_FRF} + \text{IMP_STALL_BACKEND_RENAME_GRF} + \text{IMP_STALL_BACKEND_RENAME_PDRF})}$$
Related telemetry artifacts**Events**

IMP_STALL_BACKEND_RENAME_FRF
IMP_STALL_BACKEND_RENAME_GRF
IMP_STALL_BACKEND_RENAME_PDRF
IMP_STALL_BACKEND_RENAME_VRF

Metric group
Rename_Effectiveness

Methodology
Stage 2

rename_stall_vec_ratio, Vector Register Rename Ratio, metric
This metric measures the percentage of rename stall cycles due to vector register rename availability

Units
This unit is expressed as cycles per rename stall cycles.

Formula
$$\frac{\text{IMP_STALL_BACKEND_RENAME_VRF}}{(\text{IMP_STALL_BACKEND_RENAME_VRF} + \text{IMP_STALL_BACKEND_RENAME_FRF} + \text{IMP_STALL_BACKEND_RENAME_GRF} + \text{IMP_STALL_BACKEND_RENAME_PDRF})}$$

Related telemetry artifacts
Events
IMP_STALL_BACKEND_RENAME_FRF
IMP_STALL_BACKEND_RENAME_GRF
IMP_STALL_BACKEND_RENAME_PDRF
IMP_STALL_BACKEND_RENAME_VRF

Metric group
Rename_Effectiveness

Methodology
Stage 2

5.23 IQ_Effectiveness metrics for C1-Premium

Issue Queue Effectiveness. This metric group provides relative stall cycles by operation type.

Summary of metrics in IQ_Effectiveness:

- Total metrics: 4

Table 5-23: IQ_Effectiveness metrics summary

Metric	Name	Description
iq_stall_lsu_percentage	Load/Store IQ Stall Percentage	This metric measures the percentage of backend stall cycles where the load or store operations IQ...
iq_stall_mx_percentage	Integer MX IQ Stall Percentage	This metric measures the percentage of backend stall cycles where the complex integer operations...
iq_stall_sx_percentage	Integer SX IQ Stall Percentage	This metric measures the percentage of backend stall cycles where the integer operations IQ could...

Metric	Name	Description
iq_stall_vpu_percentage	Vector IQ Stall Percentage	This metric measures the percentage of backend stall cycles where the vector operations IQ could...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

iq_stall_lsu_percentage, Load/Store IQ Stall Percentage, metric

This metric measures the percentage of backend stall cycles where the load or store operations IQ could not accept new instructions

Units

This unit is expressed as percent of backend stalls.

Formula

$\text{IMP_STALL_BACKEND_IQ_LS} / \text{STALL_BACKEND_BUSY} * 100$

Related telemetry artifacts

Events

[IMP_STALL_BACKEND_IQ_LS](#)
[STALL_BACKEND_BUSY](#)

Metric group

[IQ_Effectiveness](#)

Methodology

Stage 2

iq_stall_mx_percentage, Integer MX IQ Stall Percentage, metric

This metric measures the percentage of backend stall cycles where the complex integer operations IQ could not accept new instructions

Units

This unit is expressed as percent of backend stalls.

Formula

$\text{IMP_STALL_BACKEND_IQ_MX} / \text{STALL_BACKEND_BUSY} * 100$

Related telemetry artifacts

Events

[IMP_STALL_BACKEND_IQ_MX](#)
[STALL_BACKEND_BUSY](#)

Metric group

[IQ_Effectiveness](#)

Methodology

Stage 2

iq_stall_sx_percentage, Integer SX IQ Stall Percentage, metric

This metric measures the percentage of backend stall cycles where the integer operations IQ could not accept new instructions

Units

This unit is expressed as percent of backend stalls.

Formula

$$\text{IMP_STALL_BACKEND_IQ_SX} / \text{STALL_BACKEND_BUSY} * 100$$

Related telemetry artifacts**Events**

IMP_STALL_BACKEND_IQ_SX
STALL_BACKEND_BUSY

Metric group

IQ_Effectiveness

Methodology

Stage 2

iq_stall_vpu_percentage, Vector IQ Stall Percentage, metric

This metric measures the percentage of backend stall cycles where the vector operations IQ could not accept new instructions

Units

This unit is expressed as percent of backend stalls.

Formula

$$\text{IMP_STALL_BACKEND_IQ_VX} / \text{STALL_BACKEND_BUSY} * 100$$

Related telemetry artifacts**Events**

IMP_STALL_BACKEND_IQ_VX
STALL_BACKEND_BUSY

Metric group

IQ_Effectiveness

Methodology

Stage 2

5.24 MCQ_Effectiveness metrics for C1-Premium

Main Commit Queue Effectiveness. This metric group provides relative stall information for out of order speculation depth stalls.

Summary of metrics in MCQ_Effectiveness:

- Total metrics: 1

Table 5-24: MCQ_Effectiveness metrics summary

Metric	Name	Description
mcq_stall_percentage	Commit Queue Stall Percentage	This metric measures the percentage of backend stall cycles where no commit queue resource was...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

mcq_stall_percentage, Commit Queue Stall Percentage, metric

This metric measures the percentage of backend stall cycles where no commit queue resource was available

Units

This unit is expressed as percent of backend stalls.

Formula

$$\text{IMP_STALL_BACKEND_MCQ} / \text{STALL_BACKEND_CPUBOUND} * 100$$

Related telemetry artifacts

Events

[IMP_STALL_BACKEND_MCQ](#)
[STALL_BACKEND_C PUBOUND](#)

Metric group

[MCQ_Effectiveness](#)

Methodology

Stage 2

5.25 Port_Utilization metrics for C1-Premium

Execution Unit Effectiveness. This metric group provides relative utilization of execution pipes for the program.

Summary of metrics in Port_Utilization:

- Total metrics: 5

Table 5-25: Port_Utilization metrics summary

Metric	Name	Description
branch_port_utilization	Branch Execution Unit Utilization	This metric measures the average number of branch operations executed per cycle
int_port_utilization	Integer Execution Unit Utilization	This metric measures the average number of integer operations executed per cycle
lsu_port_utilization	Load/Store Address Execution Unit Utilization	This metric measures the average number of load/store operations executed per cycle
std_port_utilization	Integer Store Data Execution Unit Utilization	This metric measures the average number of integer store data operations executed per cycle

Metric	Name	Description
vpu_port_utilization	Vector Execution Unit Utilization	This metric measures the average number of vector operations executed per cycle

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

branch_port_utilization, Branch Execution Unit Utilization, metric

This metric measures the average number of branch operations executed per cycle

Units

This unit is expressed as operations per cycle.

Formula

$$\text{IMP_OP_BRU_ISSUE} / (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES})$$

Related telemetry artifacts

Events

[CPU_CYCLES](#)
[IMP_OP_BRU_ISSUE](#)
[IMP_WFX_CLOCK_CYCLES](#)

Metric group

[Port_Utilization](#)

Methodology

Stage 2

int_port_utilization, Integer Execution Unit Utilization, metric

This metric measures the average number of integer operations executed per cycle

Units

This unit is expressed as operations per cycle.

Formula

$$\text{IMP_OP_DPU_ISSUE} / (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES})$$

Related telemetry artifacts

Events

[CPU_CYCLES](#)
[IMP_OP_DPU_ISSUE](#)
[IMP_WFX_CLOCK_CYCLES](#)

Metric group

[Port_Utilization](#)

Methodology

Stage 2

lsu_port_utilization, Load/Store Address Execution Unit Utilization, metric

This metric measures the average number of load/store operations executed per cycle

Units

This unit is expressed as operations per cycle.

Formula

$$\text{IMP_OP_LSU_ISSUE} / (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES})$$

Related telemetry artifacts**Events**

CPU_CYCLES
IMP_OP_LSU_ISSUE
IMP_WFX_CLOCK_CYCLES

Metric group

Port_Utilization

Methodology

Stage 2

std_port_utilization, Integer Store Data Execution Unit Utilization, metric

This metric measures the average number of integer store data operations executed per cycle

Units

This unit is expressed as operations per cycle.

Formula

$$\text{IMP_OP_STD_ISSUE} / (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES})$$

Related telemetry artifacts**Events**

CPU_CYCLES
IMP_OP_STD_ISSUE
IMP_WFX_CLOCK_CYCLES

Metric group

Port_Utilization

Methodology

Stage 2

vpu_port_utilization, Vector Execution Unit Utilization, metric

This metric measures the average number of vector operations executed per cycle

Units

This unit is expressed as operations per cycle.

Formula

$$\text{IMP_OP_VPU_ISSUE} / (\text{CPU_CYCLES} - \text{IMP_WFX_CLOCK_CYCLES})$$

Related telemetry artifacts**Events**

CPU_CYCLES
IMP_OP_VPU_ISSUE
IMP_WFX_CLOCK_CYCLES

Metric group

Port_Utilization

Methodology

Stage 2

5.26 Prefetcher_Effectiveness metrics for C1-Premium

Prefetcher Effectiveness. L1 and L2 prefetcher effectiveness metrics.

Summary of metrics in Prefetcher_Effectiveness:

- Total metrics: 9

Table 5-26: Prefetcher_Effectiveness metrics summary

Metric	Name	Description
l1_prefetcher_accuracy	L1D Prefetcher Accuracy	This metric measures L1D cache prefetcher accuracy
l1_prefetcher_coverage	L1D Prefetcher Coverage	This metric measures L1D cache prefetcher coverage
l1_prefetcher_timeliness	L1D Prefetcher Timeliness	This metric measures L1D cache prefetcher timeliness
l2_prefetcher_accuracy_l1hwprf_exclusive	L2 Cache Prefetcher Accuracy (L1 HW prefetch exclusive)	This metric measures L2D cache prefetcher accuracy excluding L1 hardware prefetch requests
l2_prefetcher_accuracy_l1hwprf_inclusive	L2 Cache Prefetcher Accuracy (L1 HW prefetch inclusive)	This metric measures L2D cache prefetcher accuracy treating L1 hardware prefetches as demand...
l2_prefetcher_coverage_l1hwprf_exclusive	L2 Cache Prefetcher Coverage (L1 HW prefetch exclusive)	This metric measures L2D cache prefetcher coverage excluding L1 hardware prefetch requests
l2_prefetcher_coverage_l1hwprf_inclusive	L2 Cache Prefetcher Coverage (L1 HW prefetch inclusive)	This metric measures L2D cache prefetcher coverage treating L1 hardware prefetches as demand...
l2_prefetcher_timeliness_l1hwprf_exclusive	L2 Cache Prefetcher Timeliness (L1 HW prefetch exclusive)	This metric measures L2D cache prefetcher timeliness excluding L1 hardware prefetch requests
l2_prefetcher_timeliness_l1hwprf_inclusive	L2 Cache Prefetcher Timeliness (L1 HW prefetch inclusive)	This metric measures L2D cache prefetcher timeliness treating L1 hardware prefetches as demand...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

l1_prefetcher_accuracy, L1D Prefetcher Accuracy, metric

This metric measures L1D cache prefetcher accuracy

Units

This unit is expressed as useful prefetches per total prefetches.

Formula

$$\frac{(L1D_CACHE_HIT_RW_FWWPRF + L1D_LFB_HIT_RW_FWWPRF)}{(L1D_CACHE_REFILL_HWPRF + L1D_LFB_HIT_RW_FWWPRF)}$$

Related telemetry artifacts**Events**

L1D_CACHE_HIT_RW_FWWPRF
L1D_CACHE_REFILL_HWPRF
L1D_LFB_HIT_RW_FWWPRF

Metric group

Prefetcher_Effectiveness

Methodology

Stage 2

l1_prefetcher_coverage, L1D Prefetcher Coverage, metric

This metric measures L1D cache prefetcher coverage

Units

This unit is expressed as useful prefetches per demand misses.

Formula

$$\frac{(L1D_CACHE_HIT_RW_FWWPRF + L1D_LFB_HIT_RW_FWWPRF)}{(L1D_CACHE_HIT_RW_FWWPRF + L1D_CACHE_REFILL_RD + L1D_CACHE_REFILL_WR)}$$

Related telemetry artifacts**Events**

L1D_CACHE_HIT_RW_FWWPRF
L1D_CACHE_REFILL_RD
L1D_CACHE_REFILL_WR
L1D_LFB_HIT_RW_FWWPRF

Metric group

Prefetcher_Effectiveness

Methodology

Stage 2

l1_prefetcher_timeliness, L1D Prefetcher Timeliness, metric

This metric measures L1D cache prefetcher timeliness

Units

This unit is expressed as timely prefetches per useful prefetches.

Formula

$$L1D_CACHE_HIT_RW_FWWPRF / (L1D_CACHE_HIT_RW_FWWPRF + L1D_LFB_HIT_RW_FWWPRF)$$

Related telemetry artifacts**Events**

L1D_CACHE_HIT_RW_FHWPRF
L1D_LFB_HIT_RW_FHWPRF

Metric group

Prefetcher_Effectiveness

Methodology

Stage 2

l2_prefetcher_accuracy_l1hwprf_exclusive, L2 Cache Prefetcher Accuracy (L1 HW prefetch exclusive), metric

This metric measures L2D cache prefetcher accuracy excluding L1 hardware prefetch requests

Units

This unit is expressed as useful prefetches per total prefetches.

Formula

$$\frac{(L2D_CACHE_HIT_RW_FHWPRF + L2D_LFB_HIT_RW_FHWPRF)}{(L2D_CACHE_REFILL_HWPRF - IMP_L2_CACHE_PREFETCH_USEFUL - IMP_L2_CACHE_PREFETCH_LATE + L2D_LFB_HIT_RW_FHWPRF + L2D_CACHE_REFILL_HWPRF)}$$
Related telemetry artifacts**Events**

IMP_L2_CACHE_PREFETCH_LATE
IMP_L2_CACHE_PREFETCH_USEFUL
L2D_CACHE_HIT_RW_FHWPRF
L2D_CACHE_REFILL_HWPRF
L2D_LFB_HIT_RW_FHWPRF

Metric group

Prefetcher_Effectiveness

Methodology

Stage 2

l2_prefetcher_accuracy_l1hwprf_inclusive, L2 Cache Prefetcher Accuracy (L1 HW prefetch inclusive), metric

This metric measures L2D cache prefetcher accuracy treating L1 hardware prefetches as demand requests

Units

This unit is expressed as useful prefetches per total prefetches.

Formula

$$\frac{(IMP_L2_CACHE_PREFETCH_USEFUL + IMP_L2_CACHE_PREFETCH_LATE)}{L2D_CACHE_REFILL_HWPRF}$$

Related telemetry artifacts**Events**

IMP_L2_CACHE_PREFETCH_LATE
IMP_L2_CACHE_PREFETCH_USEFUL
L2D_CACHE_REFILL_HWPRF

Metric group

Prefetcher_Effectiveness

Methodology

Stage 2

I2_prefetcher_coverage_l1hwprf_exclusive, L2 Cache Prefetcher Coverage (L1 HW prefetch exclusive), metric

This metric measures L2D cache prefetcher coverage excluding L1 hardware prefetch requests

Units

This unit is expressed as useful prefetches per demand misses.

Formula

$$\frac{(L2D_CACHE_HIT_RW_FWPRF + L2D_LFB_HIT_RW_FWPRF)}{(L2D_CACHE_REFILL_RD + L2D_CACHE_REFILL_WR)}$$
Related telemetry artifacts**Events**

L2D_CACHE_HIT_RW_FWPRF
L2D_CACHE_REFILL_RD
L2D_CACHE_REFILL_WR
L2D_LFB_HIT_RW_FWPRF

Metric group

Prefetcher_Effectiveness

Methodology

Stage 2

I2_prefetcher_coverage_l1hwprf_inclusive, L2 Cache Prefetcher Coverage (L1 HW prefetch inclusive), metric

This metric measures L2D cache prefetcher coverage treating L1 hardware prefetches as demand requests

Units

This unit is expressed as useful prefetches per demand misses.

Formula

$$\frac{(IMP_L2_CACHE_PREFETCH_USEFUL + IMP_L2_CACHE_PREFETCH_LATE)}{(IMP_L2_CACHE_PREFETCH_USEFUL + IMP_L2_CACHE_PREFETCH_LATE + L2D_CACHE_REFILL - L2D_CACHE_REFILL_HWPRF)}$$

Related telemetry artifacts**Events**

IMP_L2_CACHE_PREFETCH_LATE
 IMP_L2_CACHE_PREFETCH_USEFUL
 L2D_CACHE_REFILL
 L2D_CACHE_REFILL_HWPRF

Metric group

Prefetcher_Effectiveness

Methodology

Stage 2

l2_prefetcher_timeliness_l1hwprf_exclusive, L2 Cache Prefetcher Timeliness (L1 HW prefetch exclusive), metric

This metric measures L2D cache prefetcher timeliness excluding L1 hardware prefetch requests

Units

This unit is expressed as timely prefetches per useful prefetches.

Formula

$$\frac{\text{L2D_CACHE_HIT_RW_FWWPRF}}{\text{L2D_CACHE_HIT_RW_FWWPRF} + \text{L2D_LFB_HIT_RW_FWWPRF}}$$
Related telemetry artifacts**Events**

L2D_CACHE_HIT_RW_FWWPRF
 L2D_LFB_HIT_RW_FWWPRF

Metric group

Prefetcher_Effectiveness

Methodology

Stage 2

l2_prefetcher_timeliness_l1hwprf_inclusive, L2 Cache Prefetcher Timeliness (L1 HW prefetch inclusive), metric

This metric measures L2D cache prefetcher timeliness treating L1 hardware prefetches as demand requests

Units

This unit is expressed as timely prefetches per useful prefetches.

Formula

$$\frac{\text{IMP_L2_CACHE_PREFETCH_USEFUL}}{\text{IMP_L2_CACHE_PREFETCH_USEFUL} + \text{IMP_L2_CACHE_PREFETCH_LATE}}$$
Related telemetry artifacts**Events**

IMP_L2_CACHE_PREFETCH_LATE

IMP_L2_CACHE_PREFETCH_USEFUL

Metric group

Prefetcher_Effectiveness

Methodology

Stage 2

5.27 Atomics_Effectiveness metrics for C1-Premium

Atomics Effectiveness. This metric group contains metrics to evaluate the effectiveness of atomics execution on this processor.

Summary of metrics in Atomics_Effectiveness:

- Total metrics: 7

Table 5-27: Atomics_Effectiveness metrics summary

Metric	Name	Description
cas_far_ratio	Compare and Swap Far Ratio	This metric measures the ratio of compare and swap instructions that did not execute locally to...
cas_near_fail_ratio	Compare and Swap Near Fail Ratio	This metric measures the ratio of failed compare and swap instructions speculatively executed...
cas_near_pass_ratio	Compare and Swap Near Pass Ratio	This metric measures the ratio of passed compare and swap instructions speculatively executed...
cas_near_ratio	Compare and Swap Near Ratio	This metric measures the ratio of compare and swap instructions speculatively executed locally to...
lse_atomics_ratio	LSE Atomics Ratio	This metric measures the ratio of LSE atomics instructions speculatively executed locally to the...
lse_load_ratio	LSE Load Ratio	This metric measures the ratio of LSE load instructions speculatively executed to the total LSE...
lse_store_ratio	LSE Store Ratio	This metric measures the ratio of LSE store instructions speculatively executed to the total LSE...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

cas_far_ratio, Compare and Swap Far Ratio, metric

This metric measures the ratio of compare and swap instructions that did not execute locally to the PE to the total compare and swap instructions.

Units

This unit is expressed as per compare and swap instruction.

Formula

1 - CAS_NEAR_SPEC / CAS_SPEC

Related telemetry artifacts**Events**[CAS_NEAR_SPEC](#)[CAS_SPEC](#)**Metric group**[Atomics_Effectiveness](#)**Methodology**

Stage 2

cas_near_fail_ratio, Compare and Swap Near Fail Ratio, metric

This metric measures the ratio of failed compare and swap instructions speculatively executed locally to the PE that do not update the location accessed to the total near compare and swap instructions.

Units

This unit is expressed as per near compare and swap instruction.

Formula
$$1 - \text{CAS_NEAR_PASS} / \text{CAS_NEAR_SPEC}$$
Related telemetry artifacts**Events**[CAS_NEAR_PASS](#)[CAS_NEAR_SPEC](#)**Metric group**[Atomics_Effectiveness](#)**Methodology**

Stage 2

cas_near_pass_ratio, Compare and Swap Near Pass Ratio, metric

This metric measures the ratio of passed compare and swap instructions speculatively executed locally to the PE that updated the location accessed to the total near compare and swap instructions.

Units

This unit is expressed as per near compare and swap instruction.

Formula
$$\text{CAS_NEAR_PASS} / \text{CAS_NEAR_SPEC}$$
Related telemetry artifacts**Events**[CAS_NEAR_PASS](#)[CAS_NEAR_SPEC](#)

Metric group[Atomics_Effectiveness](#)**Methodology**

Stage 2

cas_near_ratio, Compare and Swap Near Ratio, metric

This metric measures the ratio of compare and swap instructions speculatively executed locally to the PE to the total compare and swap instructions.

Units

This unit is expressed as per compare and swap instruction.

Formula
$$\text{CAS_NEAR_SPEC} / \text{CAS_SPEC}$$
Related telemetry artifacts**Events**[CAS_NEAR_SPEC](#)[CAS_SPEC](#)**Metric group**[Atomics_Effectiveness](#)**Methodology**

Stage 2

lse_atomics_ratio, LSE Atomics Ratio, metric

This metric measures the ratio of LSE atomics instructions speculatively executed locally to the PE to the total load and store instructions.

Units

This unit is expressed as per load/store instruction.

Formula
$$\text{LSE_LDST_SPEC} / \text{LDST_SPEC}$$
Related telemetry artifacts**Events**[LDST_SPEC](#)[LSE_LDST_SPEC](#)**Metric group**[Atomics_Effectiveness](#)**Methodology**

Stage 2

lse_load_ratio, LSE Load Ratio, metric

This metric measures the ratio of LSE load instructions speculatively executed to the total LSE instructions.

Units

This unit is expressed as per lse instruction.

Formula

$LSE_LD_SPEC / LSE_LDST_SPEC$

Related telemetry artifacts**Events**

[LSE_LDST_SPEC](#)

[LSE_LD_SPEC](#)

Metric group

[Atomics_Effectiveness](#)

Methodology

Stage 2

lse_store_ratio, LSE Store Ratio, metric

This metric measures the ratio of LSE store instructions speculatively executed to the total LSE instructions.

Units

This unit is expressed as per lse instruction.

Formula

$LSE_ST_SPEC / LSE_LDST_SPEC$

Related telemetry artifacts**Events**

[LSE_LDST_SPEC](#)

[LSE_ST_SPEC](#)

Metric group

[Atomics_Effectiveness](#)

Methodology

Stage 2

5.28 Average_Latency metrics for C1-Premium

Average Latency. This metric group contains metrics that provide average latencies of costly memory related operations by the CPU that can have variable number of cycles

Summary of metrics in Average_Latency:

- Total metrics: 5

Table 5-28: Average_Latency metrics summary

Metric	Name	Description
bus_read_requests_average_latency	Bus Read Request Average Latency	This metric measures the average latency of read requests made by this processor on the system...
dtlb_walk_average_latency	DTLB Walk Average Latency	This metric measures the average latency of data TLB walks in CPU cycles
instruction_fetch_average_latency	Instruction Fetch Average Latency	This metric measures the average latency of instruction fetches in CPU cycles
itlb_walk_average_latency	ITLB Walk Average Latency	This metric measures the average latency of instruction TLB walks in CPU cycles
load_average_latency	Load Operation Average Latency	This metric measures the average latency of load operations in CPU cycles

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

bus_read_requests_average_latency, Bus Read Request Average Latency, metric

This metric measures the average latency of read requests made by this processor on the system bus in CPU cycles

Units

This unit is expressed in cycles..

Formula

[BUS_REQ_RD_PERCYC](#) / [BUS_REQ_RD](#)

Related telemetry artifacts

Events

[BUS_REQ_RD](#)

[BUS_REQ_RD_PERCYC](#)

Metric group

[Average_Latency](#)

Other metric group: [Bus_Effectiveness](#)

Methodology

Stage 2

dtlb_walk_average_latency, DTLB Walk Average Latency, metric**

This metric measures the average latency of data TLB walks in CPU cycles

Units

This unit is expressed in cycles..

Formula

[DTLB_WALK_PERCYC](#) / [DTLB_WALK](#)

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts**Events**

[DTLB_WALK](#)
[DTLB_WALK_PERCYC](#)

Metric group

[Average_Latency](#)
Other metric group: [DTLB_Effectiveness](#)

Methodology

Stage 2

instruction_fetch_average_latency, Instruction Fetch Average Latency, metric

This metric measures the average latency of instruction fetches in CPU cycles

Units

This unit is expressed in cycles..

Formula

$\text{INST_FETCH_PERCYC} / \text{INST_FETCH}$

Related telemetry artifacts**Events**

[INST_FETCH](#)
[INST_FETCH_PERCYC](#)

Metric group

[Average_Latency](#)

Methodology

Stage 2

itlb_walk_average_latency, ITLB Walk Average Latency, metric**

This metric measures the average latency of instruction TLB walks in CPU cycles

Units

This unit is expressed in cycles..

Formula

$\text{ITLB_WALK_PERCYC} / \text{ITLB_WALK}$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts**Events**

[ITLB_WALK](#)
[ITLB_WALK_PERCYC](#)

Metric group

[Average_Latency](#)

Other metric group: [ITLB_Effectiveness](#)

Methodology

Stage 2

load_average_latency, Load Operation Average Latency, metric

This metric measures the average latency of load operations in CPU cycles

Units

This unit is expressed in cycles..

Formula

$$\text{MEM_ACCESS_RD_PERCYC} / \text{MEM_ACCESS}$$

Related telemetry artifacts

Events

[MEM_ACCESS](#)

[MEM_ACCESS_RD_PERCYC](#)

Metric group

[Average_Latency](#)

Methodology

Stage 2

5.29 Bus_Effectiveness metrics for C1-Premium

Bus Effectiveness. This metric group contains metrics to evaluate the effectiveness of bus transactions issued by this processor

Summary of metrics in Bus_Effectiveness:

- Total metrics: 2

Table 5-29: Bus_Effectiveness metrics summary

Metric	Name	Description
bus_access_average_count	Bus Access Average Count	This metric measures the average count of bus accesses taken for read and write requests made by...
bus_read_requests_average_latency	Bus Read Request Average Latency	This metric measures the average latency of read requests made by this processor on the system...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

bus_access_average_count, Bus Access Average Count, metric

This metric measures the average count of bus accesses taken for read and write requests made by this processor

Units

This unit is expressed as bus accesses.

Formula

$$(\text{BUS_ACCESS_RD} + \text{BUS_ACCESS_WR}) / \text{BUS_REQ}$$

Related telemetry artifacts**Events**

[BUS_ACCESS_RD](#)
[BUS_ACCESS_WR](#)
[BUS_REQ](#)

Metric group

[Bus_Effectiveness](#)

Methodology

Stage 2

bus_read_requests_average_latency, Bus Read Request Average Latency, metric**

This metric measures the average latency of read requests made by this processor on the system bus in CPU cycles

Units

This unit is expressed in cycles..

Formula

$$\text{BUS_REQ_RD_PERCYC} / \text{BUS_REQ_RD}$$

** This metric is used in multiple metric groups. See the following for more information.

Related telemetry artifacts**Events**

[BUS_REQ_RD](#)
[BUS_REQ_RD_PERCYC](#)

Metric group

[Bus_Effectiveness](#)
Other metric group: [Average_Latency](#)

Methodology

Stage 2

5.30 System_Memory_Effectiveness metrics for C1-Premium

System Memory Effectiveness. This metric group contains a set of metrics to analyze a system memory bound workload, providing hierarchical data hits in system memory resources internal or external to the processor.

Summary of metrics in System_Memory_Effectiveness:

- Total metrics: 4

Table 5-30: System_Memory_Effectiveness metrics summary

Metric	Name	Description
system_dram_mem_hit_ratio	System DRAM Hit Ratio	This metric measures the ratio of DRAM hits to the total memory accesses that missed in the...
system_l3_cache_hit_ratio	System L3 Cache Hit Ratio	This metric measures the ratio of system L3 cache hits to the total memory accesses that...
system_llc_cache_hit_ratio	System Last Level Cache Hit Ratio	This metric measures the ratio of system last level cache hits to the total memory accesses that...
system_peer_cluster_cache_hit_ratio	System Peer Cluster Cache Hit Ratio	This metric measures the ratio of peer cluster cache hits to the total memory accesses that...

For a complete list of the metrics in C1-Premium, see [Metrics cheat sheet for C1-Premium](#) and [Metrics lookup table for C1-Premium](#).

system_dram_mem_hit_ratio, System DRAM Hit Ratio, metric

This metric measures the ratio of DRAM hits to the total memory accesses that missed in the private L2 cache of the core. This metric indicates that the workload is memory bound obtaining data from the local DRAM.

Units

This unit is expressed as per l2 cache refill.

Formula

$$\text{IMP_DRAM_ACCESS} / (\text{L2D_CACHE_REFILL} + \text{L2I_CACHE_REFILL})$$

Related telemetry artifacts

Events

[IMP_DRAM_ACCESS](#)

[L2D_CACHE_REFILL](#)

[L2I_CACHE_REFILL](#)

Metric group

[System_Memory_Effectiveness](#)

Methodology

Stage 2

system_l3_cache_hit_ratio, System L3 Cache Hit Ratio, metric

This metric measures the ratio of system L3 cache hits to the total memory accesses that missed in the private L2 cache of the core. This metric indicates that the workload is memory bound obtaining data from L3 cache.

Units

This unit is expressed as per l2 cache refill.

Formula

$$\text{L3D_CACHE_HIT} / (\text{L2D_CACHE_REFILL} + \text{L2I_CACHE_REFILL})$$

Related telemetry artifacts**Events**

L2D_CACHE_REFILL

L2I_CACHE_REFILL

L3D_CACHE_HIT

Metric group

System_Memory_Effectiveness

Methodology

Stage 2

system_llc_cache_hit_ratio, System Last Level Cache Hit Ratio, metric

This metric measures the ratio of system last level cache hits to the total memory accesses that missed in the private L2 cache of the core. This metric indicates that the workload is memory bound obtaining data from system's last level cache.

Units

This unit is expressed as per l2 cache refill.

Formula

$$\text{LL_CACHE_HIT} / (\text{L2D_CACHE_REFILL} + \text{L2I_CACHE_REFILL})$$

Related telemetry artifacts**Events**

L2D_CACHE_REFILL

L2I_CACHE_REFILL

LL_CACHE_HIT

Metric group

System_Memory_Effectiveness

Methodology

Stage 2

system_peer_cluster_cache_hit_ratio, System Peer Cluster Cache Hit Ratio, metric

This metric measures the ratio of peer cluster cache hits to the total memory accesses that missed in the private L2 cache of the core. This metric indicates that the workload is memory bound obtaining data from peer cores in the compute cluster.

Units

This unit is expressed as per I2 cache refill.

Formula

$$(\text{DSNP_HIT} + \text{ISNP_HIT_RD}) / (\text{L2D_CACHE_REFILL} + \text{L2I_CACHE_REFILL})$$

Related telemetry artifacts**Events**

DSNP_HIT
ISNP_HIT_RD
L2D_CACHE_REFILL
L2I_CACHE_REFILL

Metric group

System_Memory_Effectiveness

Methodology

Stage 2

6. PMU events by functional group in C1-Premium

The Performance Monitoring Unit (PMU) collects events through an event interface from other units in the design. These events are used as triggers for event counters. Not all of the possible events are used in the Methodology, however, they are all listed for completeness.

C1-Premium provides the following types of PMU events:

- Total implemented Common events: 311
- Total Implemented Product ImpDef events: 61
- PMU Only events : 61
- ETE Only events : 3

PMU events for C1-Premium are grouped into the following functional groups:

- [Bus](#), BUS (8 events)
- [Chain](#), CHAIN (1 events)
- [Exception](#), EXCEPTION (15 events)
- [L1D_Cache](#), L1D CACHE (25 events)
- [L1I_Cache](#), L1I CACHE (15 events)
- [L2_Cache](#), L2 CACHE (34 events)
- [L3_Cache](#), L3 CACHE (10 events)
- [LL_Cache](#), LL CACHE (4 events)
- [Memory](#), MEMORY (15 events)
- [Retired](#), RETIRED (35 events)
- [SPE](#), SPE (10 events)
- [Spec_Operation](#), SPEC OPERATION (53 events)
- [FP_Operation](#), FP OPERATION (5 events)
- [Stall](#), STALL (44 events)
- [General](#), GENERAL (20 events)
- [TLB](#), TLB (39 events)
- [SVE](#), SVE (18 events)
- [Non_PMU](#), NON_PMU (2 events)
- [TRACE](#), TRACE (10 events)
- [CPU_Debug](#), CPU_DEBUG (7 events)
- [Coherency](#), COHERENCY (3 events)

6.1 Bus (BUS) events for C1-Premium

Bus transaction related events.

Summary of events in Bus:

- Total implemented Common events: 8
- Total Implemented Product ImpDef events: 0
- PMU Only events : 0
- ETE Only events : 0

Table 6-1: Bus events summary

Code	Mnemonic	Name	Description
0x0019	BUS_ACCESS	Bus access	Counts memory transactions issued by the CPU to the external bus, including snoop requests and...
0x001D	BUS_CYCLES	Bus cycle	Counts bus cycles in the CPU. Bus cycles represent a clock cycle in which a transaction could be...
0x0060	BUS_ACCESS_RD	Bus access, read	Counts memory read transactions seen on the external bus. Each beat of data is counted individually.
0x0061	BUS_ACCESS_WR	Bus access, write	Counts memory write transactions seen on the external bus. Each beat of data is counted...
0x8125	BUS_REQ_RD_PERCYC	Bus read transactions in progress	Counts memory read transaction requests issued by the CPU to the external bus in progress on a...
0x818D	BUS_REQ_RD	Bus request, read	Counts memory read transaction requests issued by the CPU to the external bus.
0x818E	BUS_REQ_WR	Bus request, write	Counts memory write transaction requests issued by the CPU to the external bus.
0x818F	BUS_REQ	Bus request	Counts memory transaction requests issued by the CPU to the external bus.

For a complete list of the events in C1-Premium, see [PMU events cheat sheet for C1-Premium](#) and [PMU events lookup table for C1-Premium](#).

0x0019 [BUS_ACCESS](#), Bus access, event

Counts memory transactions issued by the CPU to the external bus, including snoop requests and snoop responses. Each beat of data is counted individually.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Bus](#)

0x001D BUS_CYCLES, Bus cycle, event

Counts bus cycles in the CPU. Bus cycles represent a clock cycle in which a transaction could be sent or received on the interface from the CPU to the external bus. Since that interface is driven at the same clock speed as the CPU, this event is a duplicate of CPU_CYCLES.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Bus](#)

0x0060 BUS_ACCESS_RD, Bus access, read, event

Counts memory read transactions seen on the external bus. Each beat of data is counted individually.

Related telemetry artifacts**Metrics**

- [bus_access_average_count](#)

Metric groups

[Bus_Effectiveness](#)

Functional groups

[Bus](#)

0x0061 BUS_ACCESS_WR, Bus access, write, event

Counts memory write transactions seen on the external bus. Each beat of data is counted individually.

Related telemetry artifacts**Metrics**

- [bus_access_average_count](#)

Metric groups

[Bus_Effectiveness](#)

Functional groups

[Bus](#)

0x8125 BUS_REQ_RD_PERCYC, Bus read transactions in progress, event

Counts memory read transaction requests issued by the CPU to the external bus in progress on a processor cycle.

Related telemetry artifacts**Metrics**

- [bus_read_requests_average_latency](#) in [Average_Latency](#)
- [bus_read_requests_average_latency](#) in [Bus_Effectiveness](#)

Metric groups[Average_Latency](#)[Bus_Effectiveness](#)**Functional groups**[Bus](#)**0x818D BUS_REQ_RD, Bus request, read, event**

Counts memory read transaction requests issued by the CPU to the external bus.

Related telemetry artifacts**Metrics**

- [bus_read_requests_average_latency](#) in [Average_Latency](#)
- [bus_read_requests_average_latency](#) in [Bus_Effectiveness](#)

Metric groups[Average_Latency](#)[Bus_Effectiveness](#)**Functional groups**[Bus](#)**0x818E BUS_REQ_WR, Bus request, write, event**

Counts memory write transaction requests issued by the CPU to the external bus.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[Bus](#)**0x818F BUS_REQ, Bus request, event**

Counts memory transaction requests issued by the CPU to the external bus.

Related telemetry artifacts**Metrics**

- [bus_access_average_count](#)

Metric groups[Bus_Effectiveness](#)**Functional groups**[Bus](#)

6.2 Chain (CHAIN) events for C1-Premium

Chain related events.

Summary of events in Chain:

- Total implemented Common events: 1
- Total Implemented Product ImpDef events: 0
- PMU Only events : 0
- ETE Only events : 0

Table 6-2: Chain events summary

Code	Mnemonic	Name	Description
0x001E	CHAIN	Chain a pair of event counters	For odd-numbered counters, this event increments the count by one for each overflow of the...

For a complete list of the events in C1-Premium, see [PMU events cheat sheet for C1-Premium](#) and [PMU events lookup table for C1-Premium](#).

0x001E CHAIN, Chain a pair of event counters, event

For odd-numbered counters, this event increments the count by one for each overflow of the preceding even-numbered counter. For even-numbered counters, there is no increment. This event is used when the even/odd pairs of registers are used as a single counter.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Chain](#)

6.3 Exception (EXCEPTION) events for C1-Premium

Exception related events.

Summary of events in Exception:

- Total implemented Common events: 15
- Total Implemented Product ImpDef events: 0
- PMU Only events : 0
- ETE Only events : 0

Table 6-3: Exception events summary

Code	Mnemonic	Name	Description
0x0009	EXC_TAKEN	Exception taken	Counts any taken architecturally visible exceptions such as IRQ, FIQ, SError, and other...
0x000A	EXC_RETURN	Instruction architecturally executed, Condition code check pass, exception return	Counts any architecturally executed exception return instructions. For example: AArch64: ERET
0x0081	EXC_UNDEF	Exception taken, other synchronous	Counts the number of synchronous exceptions which are taken locally that are due to attempting to...
0x0082	EXC_SVC	Exception taken, Supervisor Call	Counts SVC exceptions taken locally.
0x0083	EXC_PABORT	Exception taken, Instruction Abort	Counts synchronous exceptions that are taken locally and caused by Instruction Aborts.
0x0084	EXC_DABORT	Exception taken, Data Abort or SError	Counts exceptions that are taken locally and are caused by data aborts or SErrors. Conditions...
0x0086	EXC_IRQ	Exception taken, IRQ	Counts IRQ exceptions including the virtual IRQs that are taken locally.
0x0087	EXC_FIQ	Exception taken, FIQ	Counts FIQ exceptions including the virtual FIQs that are taken locally.
0x0088	EXC_SMC	Exception taken, Secure Monitor Call	Counts SMC exceptions take to EL3.
0x008A	EXC_HVC	Exception taken, Hypervisor Call	Counts HVC exceptions taken to EL2.
0x008B	EXC_TRAP_PABORT	Exception taken, Instruction Abort not Taken locally	Counts exceptions which are traps not taken locally and are caused by Instruction Aborts. For...
0x008C	EXC_TRAP_DABORT	Exception taken, Data Abort or SError not Taken locally	Counts exceptions which are traps not taken locally and are caused by Data Aborts or SError...
0x008D	EXC_TRAP_OTHER	Exception taken, other traps not Taken locally	Counts the number of synchronous trap exceptions which are not taken locally and are not SVC,...
0x008E	EXC_TRAP_IRQ	Exception taken, IRQ not Taken locally	Counts IRQ exceptions including the virtual IRQs that are not taken locally.
0x008F	EXC_TRAP_FIQ	Exception taken, FIQ not Taken locally	Counts FIQs which are not taken locally but taken from EL0, EL1, or EL2 to EL3 (which would be...

For a complete list of the events in C1-Premium, see [PMU events cheat sheet for C1-Premium](#) and [PMU events lookup table for C1-Premium](#).

0x0009 EXC_TAKEN, Exception taken, event

Counts any taken architecturally visible exceptions such as IRQ, FIQ, SError, and other synchronous exceptions. Exceptions are counted whether or not they are taken locally.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Exception](#)

0x000A EXC_RETURN, Instruction architecturally executed, Condition code check pass, exception return, event

Counts any architecturally executed exception return instructions. For example: AArch64: ERET

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Exception](#)

0x0081 EXC_UNDEF, Exception taken, other synchronous, event

Counts the number of synchronous exceptions which are taken locally that are due to attempting to execute an instruction that is **UNDEFINED**. Attempting to execute instruction bit patterns that have not been allocated. Attempting to execute instructions when they are disabled. Attempting to execute instructions at an inappropriate Exception level. Attempting to execute an instruction when the value of PSTATE.IL is 1.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Exception](#)

0x0082 EXC_SVC, Exception taken, Supervisor Call, event

Counts SVC exceptions taken locally.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Exception](#)

0x0083 EXC_PABORT, Exception taken, Instruction Abort, event

Counts synchronous exceptions that are taken locally and caused by Instruction Aborts.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Exception](#)

0x0084 EXC_DABORT, Exception taken, Data Abort or SError, event

Counts exceptions that are taken locally and are caused by data aborts or SErrors. Conditions that could cause those exceptions are attempting to read or write memory where the MMU generates a fault, attempting to read or write memory with a misaligned address, interrupts from the nSEI inputs and internally generated SErrors.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[Exception](#)**0x0086 EXC_IRQ, Exception taken, IRQ, event**

Counts IRQ exceptions including the virtual IRQs that are taken locally.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[Exception](#)**0x0087 EXC_FIQ, Exception taken, FIQ, event**

Counts FIQ exceptions including the virtual FIQs that are taken locally.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[Exception](#)**0x0088 EXC_SMC, Exception taken, Secure Monitor Call, event**

Counts SMC exceptions take to EL3.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[Exception](#)**0x008A EXC_HVC, Exception taken, Hypervisor Call, event**

Counts HVC exceptions taken to EL2.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[Exception](#)**0x008B EXC_TRAP_PABORT, Exception taken, Instruction Abort not Taken locally, event**

Counts exceptions which are traps not taken locally and are caused by Instruction Aborts. For example, attempting to execute an instruction with a misaligned PC.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

Exception

0x008C EXC_TRAP_DABORT, Exception taken, Data Abort or SError not Taken locally, event

Counts exceptions which are traps not taken locally and are caused by Data Aborts or SError interrupts. Conditions that could cause those exceptions are:

1. Attempting to read or write memory where the MMU generates a fault,
2. Attempting to read or write memory with a misaligned address,
3. Interrupts from the SEI input.
4. internally generated SErrors.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

Exception

0x008D EXC_TRAP_OTHER, Exception taken, other traps not Taken locally, event

Counts the number of synchronous trap exceptions which are not taken locally and are not SVC, SMC, HVC, data aborts, Instruction Aborts, or interrupts.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

Exception

0x008E EXC_TRAP_IRQ, Exception taken, IRQ not Taken locally, event

Counts IRQ exceptions including the virtual IRQs that are not taken locally.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

Exception

0x008F EXC_TRAP_FIQ, Exception taken, FIQ not Taken locally, event

Counts FIQs which are not taken locally but taken from EL0, EL1, or EL2 to EL3 (which would be the normal behavior for FIQs when not executing in EL3).

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

Exception

6.4 L1D_Cache (L1D CACHE) events for C1-Premium

L1 data cache related events.

Summary of events in L1D_Cache:

- Total implemented Common events: 25
- Total Implemented Product ImpDef events: 0
- PMU Only events : 0
- ETE Only events : 0

Table 6-4: L1D_Cache events summary

Code	Mnemonic	Name	Description
0x0003	L1D_CACHE_REFILL	Level 1 data cache refill	Counts level 1 data cache refills caused by speculatively executed load or store operations that...
0x0004	L1D_CACHE	Level 1 data cache access	Counts level 1 data cache accesses from any load/store operations. Atomic operations that resolve...
0x0015	L1D_CACHE_WB	Level 1 data cache write-back	Counts write-backs of dirty data from the L1 data cache to the L2 cache. This occurs when either...
0x0039	L1D_CACHE_LMISS_RD	Level 1 data cache long-latency read miss	Counts cache line refills into the level 1 data cache from any memory read operations, that...
0x0040	L1D_CACHE_RD	Level 1 data cache access, read	Counts level 1 data cache accesses from any load operation. Atomic load operations that resolve...
0x0041	L1D_CACHE_WR	Level 1 data cache access, write	Counts level 1 data cache accesses generated by store operations. This event also counts accesses...
0x0042	L1D_CACHE_REFILL_RD	Level 1 data cache refill, read	Counts level 1 data cache refills caused by speculatively executed load instructions where the...
0x0043	L1D_CACHE_REFILL_WR	Level 1 data cache refill, write	Counts level 1 data cache refills caused by speculatively executed store instructions where the...
0x0044	L1D_CACHE_REFILL_INNER	Level 1 data cache refill, inner	Counts level 1 data cache refills where the cache line data came from caches inside the immediate...
0x0045	L1D_CACHE_REFILL_OUTER	Level 1 data cache refill, outer	Counts level 1 data cache refills for which the cache line data came from outside the immediate...

Code	Mnemonic	Name	Description
0x0046	L1D_CACHE_WB_VICTIM	Level 1 data cache write-back, victim	Counts dirty cache line evictions from the level 1 data cache caused by a new cache line...
0x0047	L1D_CACHE_WB_CLEAN	Level 1 data cache write-back, cleaning and coherency	Counts write-backs from the level 1 data cache that are a result of a coherency operation made by...
0x0048	L1D_CACHE_INVALID	Level 1 data cache invalidate	Counts each explicit invalidation of a cache line in the level 1 data cache caused by: <ul style="list-style-type: none"> Cache...
0x8140	L1D_CACHE_RW	Level 1 data cache demand access	Counts level 1 data demand cache accesses from any load or store operation. Near atomic...
0x8142	L1D_CACHE_PRFM	Level 1 data cache software preload	Counts level 1 data cache accesses from software preload or prefetch instructions.
0x8144	L1D_CACHE_MISS	Level 1 data cache demand access miss	Counts cache line misses in the level 1 data cache.
0x8146	L1D_CACHE_REFILL_PRFM	Level 1 data cache refill, software preload	Counts level 1 data cache refills where the cache line access was generated by software preload...
0x8154	L1D_CACHE_HWPRF	Level 1 data cache hardware prefetch	Counts level 1 data cache accesses from any load/store operations generated by the hardware...
0x81BC	L1D_CACHE_REFILL_HWPRF	Level 1 data cache refill, hardware prefetch	Counts level 1 data cache refills where the cache line is requested by a hardware prefetcher.
0x81EC	L1D_CACHE_HIT_RW_FHWPRF	Level 1 data cache demand access first hit, fetched by hardware prefetcher	Counts first level 1 data demand cache hit from any load or store operation where the cache line...
0x8244	L1D_LFB_HIT_RD	Level 1 data cache demand line-fill buffer hit, read	Counts load data accesses that access the level 1 data cache and hit in a line that is in the...
0x8248	L1D_LFB_HIT_WR	Level 1 data cache demand access line-fill buffer hit, write	Counts store data accesses that access the level 1 data cache and hit in a line that is in the...
0x824C	L1D_LFB_HIT_RW	Level 1 data cache demand access line-fill buffer hit	Counts load or store data accesses that access the level 1 data cache and hit in a line that is...
0x826C	L1D_LFB_HIT_RW_FHWPRF	Level 1 data cache demand access line-fill buffer first hit, recently fetched by hardware prefetcher	Counts first load or store data accesses that access the level 1 data cache and hit in a line...
0x8284	L1D_CACHE_PRF	Level 1 data cache, preload or prefetch hit	Counts level 1 data cache accesses from software preload or prefetch instructions or hardware...

For a complete list of the events in C1-Premium, see [PMU events cheat sheet for C1-Premium](#) and [PMU events lookup table for C1-Premium](#).

0x0003 L1D_CACHE_REFILL, Level 1 data cache refill, event

Counts level 1 data cache refills caused by speculatively executed load or store operations that missed in the level 1 data cache. This event only counts one event per cache line.

Related telemetry artifacts

Metrics

- [l1d_cache_mpki](#) in [L1D_Cache_Effectiveness](#)
- [l1d_cache_mpki](#) in [MPKI](#)
- [l1d_cache_miss_ratio](#) in [L1D_Cache_Effectiveness](#)
- [l1d_cache_miss_ratio](#) in [Miss_Ratio](#)

Metric groups

[L1D_Cache_Effectiveness](#)

[MPKI](#)

[Miss_Ratio](#)

Functional groups

[L1D_Cache](#)

0x0004 L1D_CACHE, Level 1 data cache access, event

Counts level 1 data cache accesses from any load/store operations. Atomic operations that resolve in the CPUs caches (near atomic operations) counts as both a write access and read access. Each access to a cache line is counted including the multiple accesses caused by single instructions such as LDM or STM. Each access to other level 1 data or unified memory structures, for example refill buffers, write buffers, and write-back buffers, are also counted.

Related telemetry artifacts

Metrics

- [l1d_cache_miss_ratio](#) in [L1D_Cache_Effectiveness](#)
- [l1d_cache_miss_ratio](#) in [Miss_Ratio](#)

Metric groups

[L1D_Cache_Effectiveness](#)

[Miss_Ratio](#)

Functional groups

[L1D_Cache](#)

0x0015 L1D_CACHE_WB, Level 1 data cache write-back, event

Counts write-backs of dirty data from the L1 data cache to the L2 cache. This occurs when either a dirty cache line is evicted from L1 data cache and allocated in the L2 cache or dirty data is written to the L2 and possibly to the next level of cache. This event counts both victim cache line evictions and cache write-backs from snoops or cache maintenance operations. The following cache operations are not counted:

1. Invalidations which do not result in data being transferred out of the L1 (such as evictions of clean data),
2. Full line writes which write to L2 without writing L1, such as write streaming mode.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L1D_Cache](#)

0x0039 L1D_CACHE_LMISS_RD, Level 1 data cache long-latency read miss, event

Counts cache line refills into the level 1 data cache from any memory read operations, that incurred additional latency.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L1D_Cache](#)

0x0040 L1D_CACHE_RD, Level 1 data cache access, read, event

Counts level 1 data cache accesses from any load operation. Atomic load operations that resolve in the CPUs caches counts as both a write access and read access.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L1D_Cache](#)

0x0041 L1D_CACHE_WR, Level 1 data cache access, write, event

Counts level 1 data cache accesses generated by store operations. This event also counts accesses caused by a DC ZVA (data cache zero, specified by virtual address) instruction. Near atomic operations that resolve in the CPUs caches count as a write access and read access.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L1D_Cache](#)

0x0042 L1D_CACHE_REFILL_RD, Level 1 data cache refill, read, event

Counts level 1 data cache refills caused by speculatively executed load instructions where the memory read operation misses in the level 1 data cache. This event only counts one event per cache line.

Related telemetry artifacts**Metrics**

- [l1d_cache_demand_mpki in L1D_Cache_Effectiveness](#)

- [l1d_cache_demand_mpki](#) in MPKI
- [l1_prefetcher_coverage](#)

Metric groups[L1D_Cache_Effectiveness](#)[MPKI](#)[Prefetcher_Effectiveness](#)**Functional groups**[L1D_Cache](#)**0x0043 L1D_CACHE_REFILL_WR, Level 1 data cache refill, write, event**

Counts level 1 data cache refills caused by speculatively executed store instructions where the memory write operation misses in the level 1 data cache. This event only counts one event per cache line.

Related telemetry artifacts**Metrics**

- [l1d_cache_demand_mpki](#) in [L1D_Cache_Effectiveness](#)
- [l1d_cache_demand_mpki](#) in MPKI
- [l1_prefetcher_coverage](#)

Metric groups[L1D_Cache_Effectiveness](#)[MPKI](#)[Prefetcher_Effectiveness](#)**Functional groups**[L1D_Cache](#)**0x0044 L1D_CACHE_REFILL_INNER, Level 1 data cache refill, inner, event**

Counts level 1 data cache refills where the cache line data came from caches inside the immediate cluster of the core.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[L1D_Cache](#)**0x0045 L1D_CACHE_REFILL_OUTER, Level 1 data cache refill, outer, event**

Counts level 1 data cache refills for which the cache line data came from outside the immediate cluster of the core, like an SLC in the system interconnect or DRAM.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[L1D_Cache](#)**0x0046 L1D_CACHE_WB_VICTIM, Level 1 data cache write-back, victim, event**

Counts dirty cache line evictions from the level 1 data cache caused by a new cache line allocation. This event does not count evictions caused by cache maintenance operations.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[L1D_Cache](#)**0x0047 L1D_CACHE_WB_CLEAN, Level 1 data cache write-back, cleaning and coherency, event**

Counts write-backs from the level 1 data cache that are a result of a coherency operation made by another CPU. Event count includes cache maintenance operations.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[L1D_Cache](#)**0x0048 L1D_CACHE_INVALID, Level 1 data cache invalidate, event**

Counts each explicit invalidation of a cache line in the level 1 data cache caused by:

- Cache Maintenance Operations (CMO) that operate by a virtual address.
- Broadcast cache coherency operations from another CPU in the system.

This event does not count for the following conditions:

1. A cache refill invalidates a cache line.
2. A CMO which is executed on that CPU and invalidates a cache line specified by set/way.

Note that CMOs that operate by set/way cannot be broadcast from one CPU to another.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[L1D_Cache](#)**0x8140 L1D_CACHE_RW, Level 1 data cache demand access, event**

Counts level 1 data demand cache accesses from any load or store operation. Near atomic operations that resolve in the CPUs caches counts as both a write access and read access.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L1D_Cache](#)

0x8142 L1D_CACHE_PRFM, Level 1 data cache software preload, event

Counts level 1 data cache accesses from software preload or prefetch instructions.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L1D_Cache](#)

0x8144 L1D_CACHE_MISS, Level 1 data cache demand access miss, event

Counts cache line misses in the level 1 data cache.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L1D_Cache](#)

0x8146 L1D_CACHE_REFILL_PRFM, Level 1 data cache refill, software preload, event

Counts level 1 data cache refills where the cache line access was generated by software preload or prefetch instructions.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L1D_Cache](#)

0x8154 L1D_CACHE_HWPRF, Level 1 data cache hardware prefetch, event

Counts level 1 data cache accesses from any load/store operations generated by the hardware prefetcher.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L1D_Cache](#)

0x81BC L1D_CACHE_REFILL_HWPRF, Level 1 data cache refill, hardware prefetch, event

Counts level 1 data cache refills where the cache line is requested by a hardware prefetcher.

Related telemetry artifacts

Metrics

- [l1_prefetcher_accuracy](#)

Metric groups

[Prefetcher_Effectiveness](#)

Functional groups

[L1D_Cache](#)

0x81EC L1D_CACHE_HIT_RW_FHWPRF, Level 1 data cache demand access first hit, fetched by hardware prefetcher, event

Counts first level 1 data demand cache hit from any load or store operation where the cache line was fetched by a hardware prefetcher.

Related telemetry artifacts

Metrics

- [l1_prefetcher_coverage](#)
- [l1_prefetcher_accuracy](#)
- [l1_prefetcher_timeliness](#)

Metric groups

[Prefetcher_Effectiveness](#)

Functional groups

[L1D_Cache](#)

0x8244 L1D_LFB_HIT_RD, Level 1 data cache demand line-fill buffer hit, read, event

Counts load data accesses that access the level 1 data cache and hit in a line that is in the process of being loaded into the level 1 data cache.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L1D_Cache](#)

0x8248 L1D_LFB_HIT_WR, Level 1 data cache demand access line-fill buffer hit, write, event

Counts store data accesses that access the level 1 data cache and hit in a line that is in the process of being loaded into the level 1 data cache.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L1D_Cache](#)

0x824c L1D_LFB_HIT_RW, Level 1 data cache demand access line-fill buffer hit, event

Counts load or store data accesses that access the level 1 data cache and hit in a line that is in the process of being loaded into the level 1 data cache.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L1D_Cache](#)

0x826c L1D_LFB_HIT_RW_FHWPRF, Level 1 data cache demand access line-fill buffer first hit, recently fetched by hardware prefetcher, event

Counts first load or store data accesses that access the level 1 data cache and hit in a line that is in the process of being loaded into the level 1 data cache.

Related telemetry artifacts**Metrics**

- [l1_prefetcher_coverage](#)
- [l1_prefetcher_accuracy](#)
- [l1_prefetcher_timeliness](#)

Metric groups

[Prefetcher_Effectiveness](#)

Functional groups

[L1D_Cache](#)

0x8284 L1D_CACHE_PRF, Level 1 data cache, preload or prefetch hit, event

Counts level 1 data cache accesses from software preload or prefetch instructions or hardware prefetcher.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L1D_Cache](#)

6.5 L1I_Cache (L1I CACHE) events for C1-Premium

L1 instruction cache related events.

Summary of events in L1I_Cache:

- Total implemented Common events: 15
- Total Implemented Product ImpDef events: 0

- PMU Only events : 0
- ETE Only events : 0

Table 6-5: L1I_Cache events summary

Code	Mnemonic	Name	Description
0x0001	L1I_CACHE_REFILL	Level 1 instruction cache refill	Counts cache line refills in the level 1 instruction cache caused by a missed instruction fetch...
0x0014	L1I_CACHE	Level 1 instruction cache access	Counts instruction fetches which access the level 1 instruction cache. Instruction cache accesses...
0x4006	L1I_CACHE_LMISS	Level 1 instruction cache long-latency miss	Counts cache line refills into the level 1 instruction cache, that incurred additional latency.
0x8141	L1I_CACHE_RD	Level 1 instruction cache demand fetch	Counts demand instruction fetches which access the level 1 instruction cache.
0x8143	L1I_CACHE_PRFM	Level 1 instruction cache software preload	Counts instruction fetches generated by software preload or prefetch instructions which access...
0x8145	L1I_CACHE_HWPRF	Level 1 instruction cache hardware prefetch	Counts instruction fetches which access the level 1 instruction cache generated by the hardware...
0x8147	L1I_CACHE_REFILL_PRFM	Level 1 instruction cache refill, software preload	Counts cache line refills in the level 1 instruction cache caused by a missed instruction fetch...
0x81C0	L1I_CACHE_HIT_RD	Level 1 instruction cache demand fetch hit	Counts demand instruction fetches that access the level 1 instruction cache and hit in the L1...
0x81D0	L1I_CACHE_HIT_RD_FPRFM	Level 1 instruction cache demand fetch first hit, fetched by software preload	Counts demand instruction fetches that access the level 1 instruction cache that hit in the L1...
0x81E0	L1I_CACHE_HIT_RD_FHWPRF	Level 1 instruction cache demand fetch first hit, fetched by hardware prefetcher	Counts demand instruction fetches generated by hardware prefetch that access the level 1...
0x8200	L1I_CACHE_HIT	Level 1 instruction cache hit	Counts instruction fetches that access the level 1 instruction cache and hit in the level 1...
0x8208	L1I_CACHE_HIT_PRFM	Level 1 instruction cache software preload hit	Counts instruction fetches generated by software preload or prefetch instructions that access the...
0x8240	L1I_LFB_HIT_RD	Level 1 instruction cache demand fetch line-fill buffer hit	Counts demand instruction fetches that access the level 1 instruction cache and hit in a line...
0x8250	L1I_LFB_HIT_RD_FPRFM	Level 1 instruction cache demand fetch line-fill buffer first hit, recently fetched by software preload	Counts demand instruction fetches generated by software prefetch instructions that access the...
0x8260	L1I_LFB_HIT_RD_FHWPRF	Level 1 instruction cache demand fetch line-fill buffer first hit, recently fetched by hardware prefetcher	Counts demand instruction fetches generated by hardware prefetch that access the level 1...

For a complete list of the events in C1-Premium, see [PMU events cheat sheet for C1-Premium](#) and [PMU events lookup table for C1-Premium](#).

0x0001 L1I_CACHE_REFILL, Level 1 instruction cache refill, event

Counts cache line refills in the level 1 instruction cache caused by a missed instruction fetch. Instruction fetches may include accessing multiple instructions, but the single cache line allocation is counted once.

Related telemetry artifacts

Metrics

- [l1i_cache_mpki](#) in [L1I_Cache_Effectiveness](#)
- [l1i_cache_mpki](#) in [MPKI](#)
- [l1i_cache_miss_ratio](#) in [L1I_Cache_Effectiveness](#)
- [l1i_cache_miss_ratio](#) in [Miss_Ratio](#)

Metric groups

[L1I_Cache_Effectiveness](#)

[MPKI](#)

[Miss_Ratio](#)

Functional groups

[L1I_Cache](#)

0x0014 L1I_CACHE, Level 1 instruction cache access, event

Counts instruction fetches which access the level 1 instruction cache. Instruction cache accesses caused by cache maintenance operations are not counted.

Related telemetry artifacts

Metrics

- [l1i_cache_miss_ratio](#) in [L1I_Cache_Effectiveness](#)
- [l1i_cache_miss_ratio](#) in [Miss_Ratio](#)

Metric groups

[L1I_Cache_Effectiveness](#)

[Miss_Ratio](#)

Functional groups

[L1I_Cache](#)

0x4006 L1I_CACHE_LMISS, Level 1 instruction cache long-latency miss, event

Counts cache line refills into the level 1 instruction cache, that incurred additional latency.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[L1I_Cache](#)**0x8141 L1I_CACHE_RD, Level 1 instruction cache demand fetch, event**

Counts demand instruction fetches which access the level 1 instruction cache.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[L1I_Cache](#)**0x8143 L1I_CACHE_PRFM, Level 1 instruction cache software preload, event**

Counts instruction fetches generated by software preload or prefetch instructions which access the level 1 instruction cache.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[L1I_Cache](#)**0x8145 L1I_CACHE_HWPRF, Level 1 instruction cache hardware prefetch, event**

Counts instruction fetches which access the level 1 instruction cache generated by the hardware prefetcher.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[L1I_Cache](#)**0x8147 L1I_CACHE_REFILL_PRFM, Level 1 instruction cache refill, software preload, event**

Counts cache line refills in the level 1 instruction cache caused by a missed instruction fetch generated by software preload or prefetch instructions. Instruction fetches may include accessing multiple instructions, but the single cache line allocation is counted once.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[L1I_Cache](#)

0x81c0 L1I_CACHE_HIT_RD, Level 1 instruction cache demand fetch hit, event

Counts demand instruction fetches that access the level 1 instruction cache and hit in the L1 instruction cache.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L1I_Cache](#)

0x81d0 L1I_CACHE_HIT_RD_FPRFM, Level 1 instruction cache demand fetch first hit, fetched by software preload, event

Counts demand instruction fetches that access the level 1 instruction cache that hit in the L1 instruction cache and the line was requested by a software prefetch.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L1I_Cache](#)

0x81e0 L1I_CACHE_HIT_RD_FHWPRF, Level 1 instruction cache demand fetch first hit, fetched by hardware prefetcher, event

Counts demand instruction fetches generated by hardware prefetch that access the level 1 instruction cache and hit in the L1 instruction cache.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L1I_Cache](#)

0x8200 L1I_CACHE_HIT, Level 1 instruction cache hit, event

Counts instruction fetches that access the level 1 instruction cache and hit in the level 1 instruction cache. Instruction cache accesses caused by cache maintenance operations are not counted.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L1I_Cache](#)

0x8208 L1I_CACHE_HIT_PRFM, Level 1 instruction cache software preload hit, event

Counts instruction fetches generated by software preload or prefetch instructions that access the level 1 instruction cache and hit in the level 1 instruction cache.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L1I_Cache](#)

0x8240 L1I_LFB_HIT_RD, Level 1 instruction cache demand fetch line-fill buffer hit, event

Counts demand instruction fetches that access the level 1 instruction cache and hit in a line that is in the process of being loaded into the level 1 instruction cache.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L1I_Cache](#)

0x8250 L1I_LFB_HIT_RD_FPRFM, Level 1 instruction cache demand fetch line-fill buffer first hit, recently fetched by software preload, event

Counts demand instruction fetches generated by software prefetch instructions that access the level 1 instruction cache and hit in a line that is in the process of being loaded into the level 1 instruction cache.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L1I_Cache](#)

0x8260 L1I_LFB_HIT_RD_FHWPRF, Level 1 instruction cache demand fetch line-fill buffer first hit, recently fetched by hardware prefetcher, event

Counts demand instruction fetches generated by hardware prefetch that access the level 1 instruction cache and hit in a line that is in the process of being loaded into the level 1 instruction cache.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L1I_Cache](#)

6.6 L2_Cache (L2 CACHE) events for C1-Premium

L2 unified cache related events.

Summary of events in L2_Cache:

- Total implemented Common events: 30
- Total Implemented Product ImpDef events: 4
- PMU Only events : 4
- ETE Only events : 0

Table 6-6: L2_Cache events summary

Code	Mnemonic	Name	Description
0x0016	L2D_CACHE	Level 2 data cache access	Counts accesses to the level 2 cache due to data accesses. Level 2 cache is a unified cache for...
0x0017	L2D_CACHE_REFILL	Level 2 data cache refill	Counts cache line refills into the level 2 cache. Level 2 cache is a unified cache for data and...
0x0018	L2D_CACHE_WB	Level 2 data cache write-back	Counts write-backs of data from the L2 cache to outside the CPU. This includes snoops to the L2...
0x0020	L2D_CACHE_ALLOCATE	Level 2 data cache allocation without refill	Counts level 2 cache line allocates that do not fetch data from outside the level 2 data or...
0x0027	L2I_CACHE	Level 2 instruction cache access	Counts accesses to the level 2 cache due to instruction accesses. Level 2 cache is a unified...
0x0028	L2I_CACHE_REFILL	Level 2 instruction cache refill	Counts cache line refills into the level 2 cache. Level 2 cache is a unified cache for data and...
0x0050	L2D_CACHE_RD	Level 2 data cache access, read	Counts level 2 data cache accesses due to memory read operations. Level 2 cache is a unified...
0x0051	L2D_CACHE_WR	Level 2 data cache access, write	Counts level 2 cache accesses due to memory write operations. Level 2 cache is a unified cache...
0x0052	L2D_CACHE_REFILL_RD	Level 2 data cache refill, read	Counts refills for memory accesses due to memory read operation counted by L2D_CACHE_RD. Level 2...
0x0053	L2D_CACHE_REFILL_WR	Level 2 data cache refill, write	Counts refills for memory accesses due to memory write operation counted by L2D_CACHE_WR. Level 2...
0x0056	L2D_CACHE_WB_VICTIM	Level 2 data cache write-back, victim	Counts evictions from the level 2 cache because of a line being allocated into the L2 cache.
0x0057	L2D_CACHE_WB_CLEAN	Level 2 data cache write-back, cleaning and coherency	Counts write-backs from the level 2 cache that are a result of either: 1. Cache maintenance...

Code	Mnemonic	Name	Description
0x0058	L2D_CACHE_INVALID	Level 2 data cache invalidate	Counts each explicit invalidation of a cache line in the level 2 cache by cache maintenance...
0x010B	IMP_L2_CACHE_PREFETCH_LATE	Late prefetch requests to L2 cache	Counts level 2 cache demand accesses for which prefetch requests were late. This event counts any...
0x0179	IMP_L2_CACHE_PREFETCH_USEFUL	L2 cache lookups for lines allocated by prefetch	Counts level 2 cache lookups for load store instructions including store, software prefetch and...
0x01B8	IMP_L2D_CACHE_L1HWPRF	L2D cache access due to L1 hardware prefetch	Counts level 2 cache accesses due to level 1 data cache hardware prefetcher.
0x01B9	IMP_L2D_CACHE_REFILL_L1HWPRF	L2D cache refill due to L1 hardware prefetch	Counts level 2 cache refills where the cache line is requested by a level 1 data cache hardware...
0x4009	L2D_CACHE_LMISS_RD	Level 2 data cache long-latency read miss	Counts cache line refills into the level 2 unified cache from any memory read operations that...
0x400A	L2I_CACHE_LMISS	Level 2 instruction cache long-latency miss	Counts cache line refills into the level 2 unified cache from any instruction read operations...
0x8148	L2D_CACHE_RW	Level 2 data cache demand access	Counts level 2 cache demand accesses from any load/store operations. Level 2 cache is a unified...
0x8149	L2I_CACHE_RD	Level 2 instruction cache demand fetch	Counts level 2 cache accesses that are due to a demand instruction cache access.
0x814A	L2D_CACHE_PRFM	Level 2 data cache software preload	Counts level 2 data cache accesses generated by software preload or prefetch instructions.
0x814B	L2I_CACHE_PRFM	Level 2 instruction cache software preload	Counts level 2 instruction cache accesses generated by software preload or prefetch instructions.
0x814C	L2D_CACHE_MISS	Level 2 data cache demand access miss	Counts cache line misses in the level 2 cache. Level 2 cache is a unified cache for data and...
0x814E	L2D_CACHE_REFILL_PRFM	Level 2 data cache refill, software preload	Counts refills due to accesses generated as a result of software preload or prefetch instructions...
0x814F	L2I_CACHE_REFILL_PRFM	Level 2 instruction cache refill, software preload	Counts level 2 instruction cache refills generated by software preload or prefetch instructions.
0x8155	L2D_CACHE_HWPRF	Level 2 data cache hardware prefetch	Counts level 2 data cache accesses generated by L2D hardware prefetchers.
0x81BD	L2D_CACHE_REFILL_HWPRF	Level 2 data cache refill, hardware prefetch	Counts level 2 data cache refills where the cache line is requested by a hardware prefetcher.
0x81C1	L2I_CACHE_HIT_RD	Level 2 instruction cache demand fetch hit	Counts level 2 cache hits that are due to a demand instruction cache access.
0x81ED	L2D_CACHE_HIT_RW_FHWPRF	Level 2 data cache demand access first hit, fetched by hardware prefetcher	Counts first level 2 data demand cache hit from any load or store operation where the cache line...

Code	Mnemonic	Name	Description
0x8209	L2I_CACHE_HIT_PRFM	Level 2 instruction cache software preload hit	Counts level 2 instruction cache hits generated by software preload or prefetch instructions.
0x824D	L2D_LFB_HIT_RW	Level 2 data cache demand access line-fill buffer hit	Counts load or store data accesses that access the level 2 data cache and hit in a line that is...
0x826D	L2D_LFB_HIT_RW_FHWPRF	Level 2 data cache demand access line-fill buffer first hit, recently fetched by hardware prefetcher	Counts first load or store data access that accesses the level 2 cache and hit in a line that is...
0x8285	L2D_CACHE_PRF	Level 2 data cache, preload or prefetch hit	Counts level 2 data cache accesses from software preload or prefetch instructions or hardware...

For a complete list of the events in C1-Premium, see [PMU events cheat sheet for C1-Premium](#) and [PMU events lookup table for C1-Premium](#).

0x0016 L2D_CACHE, Level 2 data cache access, event

Counts accesses to the level 2 cache due to data accesses. Level 2 cache is a unified cache for data and instruction accesses. Accesses are for misses in the first level data cache or translation resolutions due to accesses. This event also counts write back of dirty data from level 1 data cache to the L2 cache.

Related telemetry artifacts

Metrics

- [l2d_cache_miss_ratio](#) in [L2D_Cache_Effectiveness](#)
- [l2d_cache_miss_ratio](#) in [Miss_Ratio](#)

Metric groups

[L2D_Cache_Effectiveness](#)

[Miss_Ratio](#)

Functional groups

[L2_Cache](#)

0x0017 L2D_CACHE_REFILL, Level 2 data cache refill, event

Counts cache line refills into the level 2 cache. Level 2 cache is a unified cache for data and instruction accesses. Accesses are for misses in the level 1 data cache or translation resolutions due to accesses.

Related telemetry artifacts

Metrics

- [l2d_cache_mpki](#) in [L2D_Cache_Effectiveness](#)
- [l2d_cache_mpki](#) in [MPKI](#)
- [l2d_cache_miss_ratio](#) in [L2D_Cache_Effectiveness](#)
- [l2d_cache_miss_ratio](#) in [Miss_Ratio](#)
- [l2_prefetcher_coverage_l1hwrpf_inclusive](#)

- [system_l3_cache_hit_ratio](#)
- [system_peer_cluster_cache_hit_ratio](#)
- [system_llc_cache_hit_ratio](#)
- [system_dram_mem_hit_ratio](#)

Metric groups[L2D_Cache_Effectiveness](#)[MPKI](#)[Miss_Ratio](#)[Prefetcher_Effectiveness](#)[System_Memory_Effectiveness](#)**Functional groups**[L2_Cache](#)**0x0018 L2D_CACHE_WB, Level 2 data cache write-back, event**

Counts write-backs of data from the L2 cache to outside the CPU. This includes snoops to the L2 (from other CPUs) which return data even if the snoops cause an invalidation. L2 cache line invalidations which do not write data outside the CPU and snoops which return data from an L1 cache are not counted. Data would not be written outside the cache when invalidating a clean cache line.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[L2_Cache](#)**0x0020 L2D_CACHE_ALLOCATE, Level 2 data cache allocation without refill, event**

Counts level 2 cache line allocates that do not fetch data from outside the level 2 data or unified cache.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[L2_Cache](#)**0x0027 L2I_CACHE, Level 2 instruction cache access, event**

Counts accesses to the level 2 cache due to instruction accesses. Level 2 cache is a unified cache for data and instruction accesses. Accesses are for misses in the first level instruction cache.

Related telemetry artifacts**Metrics**

- [l2i_cache_miss_ratio](#) in [L2I_Cache_Effectiveness](#)

- [l2i_cache_miss_ratio](#) in [Miss_Ratio](#)

Metric groups[L2I_Cache_Effectiveness](#)[Miss_Ratio](#)**Functional groups**[L2_Cache](#)**0x0028 L2I_CACHE_REFILL, Level 2 instruction cache refill, event**

Counts cache line refills into the level 2 cache. Level 2 cache is a unified cache for data and instruction accesses. Accesses are for misses in the level 1 instruction cache.

Related telemetry artifacts**Metrics**

- [l2i_cache_mpki](#) in [L2I_Cache_Effectiveness](#)
- [l2i_cache_mpki](#) in [MPKI](#)
- [l2i_cache_miss_ratio](#) in [L2I_Cache_Effectiveness](#)
- [l2i_cache_miss_ratio](#) in [Miss_Ratio](#)
- [system_l3_cache_hit_ratio](#)
- [system_peer_cluster_cache_hit_ratio](#)
- [system_llc_cache_hit_ratio](#)
- [system_dram_mem_hit_ratio](#)

Metric groups[L2I_Cache_Effectiveness](#)[MPKI](#)[Miss_Ratio](#)[System_Memory_Effectiveness](#)**Functional groups**[L2_Cache](#)**0x0050 L2D_CACHE_RD, Level 2 data cache access, read, event**

Counts level 2 data cache accesses due to memory read operations. Level 2 cache is a unified cache for data and instruction accesses, accesses are for misses in the level 1 data cache or translation resolutions due to accesses.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[L2_Cache](#)

0x0051 L2D_CACHE_WR, Level 2 data cache access, write, event

Counts level 2 cache accesses due to memory write operations. Level 2 cache is a unified cache for data and instruction accesses, accesses are for misses in the level 1 data cache or translation resolutions due to accesses.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L2_Cache](#)

0x0052 L2D_CACHE_REFILL_RD, Level 2 data cache refill, read, event

Counts refills for memory accesses due to memory read operation counted by L2D_CACHE_RD. Level 2 cache is a unified cache for data and instruction accesses, accesses are for misses in the level 1 data cache or translation resolutions due to accesses.

Related telemetry artifacts**Metrics**

- [l2d_cache_demand_mpki](#) in [L2D_Cache_Effectiveness](#)
- [l2d_cache_demand_mpki](#) in [MPKI](#)
- [l2_prefetcher_coverage_l1hwprf_exclusive](#)

Metric groups

[L2D_Cache_Effectiveness](#)

[MPKI](#)

[Prefetcher_Effectiveness](#)

Functional groups

[L2_Cache](#)

0x0053 L2D_CACHE_REFILL_WR, Level 2 data cache refill, write, event

Counts refills for memory accesses due to memory write operation counted by L2D_CACHE_WR. Level 2 cache is a unified cache for data and instruction accesses, accesses are for misses in the level 1 data cache or translation resolutions due to accesses.

Related telemetry artifacts**Metrics**

- [l2d_cache_demand_mpki](#) in [L2D_Cache_Effectiveness](#)
- [l2d_cache_demand_mpki](#) in [MPKI](#)
- [l2_prefetcher_coverage_l1hwprf_exclusive](#)

Metric groups

[L2D_Cache_Effectiveness](#)

[MPKI](#)

[Prefetcher_Effectiveness](#)

Functional groups

[L2_Cache](#)

0x0056 L2D_CACHE_WB_VICTIM, Level 2 data cache write-back, victim, event

Counts evictions from the level 2 cache because of a line being allocated into the L2 cache.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L2_Cache](#)

0x0057 L2D_CACHE_WB_CLEAN, Level 2 data cache write-back, cleaning and coherency, event

Counts write-backs from the level 2 cache that are a result of either:

1. Cache maintenance operations,
2. Snoop responses or,
3. Direct cache transfers to another CPU due to a forwarding snoop request.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L2_Cache](#)

0x0058 L2D_CACHE_INVALID, Level 2 data cache invalidate, event

Counts each explicit invalidation of a cache line in the level 2 cache by cache maintenance operations that operate by a virtual address, or by external coherency operations. This event does not count if either:

1. A cache refill invalidates a cache line or,
2. A Cache Maintenance Operation (CMO), which invalidates a cache line specified by set/way, is executed on that CPU.

CMOs that operate by set/way cannot be broadcast from one CPU to another.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L2_Cache](#)

0x010B IMP_L2_CACHE_PREFETCH_LATE, Late prefetch requests to L2 cache, event

Counts level 2 cache demand accesses for which prefetch requests were late. This event counts any lookups in the L2 cache that occur after a prefetch request was generated by the prefetcher,

but before the prefetched cache line is allocated into the cache. It indicates that the prefetch was useful but not on time.

Related telemetry artifacts

Metrics

- [l2_prefetcher_coverage_l1hwprf_inclusive](#)
- [l2_prefetcher_accuracy_l1hwprf_inclusive](#)
- [l2_prefetcher_timeliness_l1hwprf_inclusive](#)
- [l2_prefetcher_accuracy_l1hwprf_exclusive](#)

Metric groups

[Prefetcher_Effectiveness](#)

Functional groups

[L2_Cache](#)

0x0179 IMP_L2_CACHE_PREFETCH_USEFUL, L2 cache lookups for lines allocated by prefetch, event

Counts level 2 cache lookups for load store instructions including store, software prefetch and L1 data prefetch that completed with cache hit on a prefetched line in L2 cache. This event counts L2 cache hits due to a data access on a line that was prefetched by hardware. It does not count if the cache line was not accessed by the CPU but instead was accessed by a snoop request from another CPU.

Related telemetry artifacts

Metrics

- [l2_prefetcher_coverage_l1hwprf_inclusive](#)
- [l2_prefetcher_accuracy_l1hwprf_inclusive](#)
- [l2_prefetcher_timeliness_l1hwprf_inclusive](#)
- [l2_prefetcher_accuracy_l1hwprf_exclusive](#)

Metric groups

[Prefetcher_Effectiveness](#)

Functional groups

[L2_Cache](#)

0x01B8 IMP_L2D_CACHE_L1HWPRF, L2D cache access due to L1 hardware prefetch, event

Counts level 2 cache accesses due to level 1 data cache hardware prefetcher.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L2_Cache](#)

0x01B9 IMP_L2D_CACHE_REFILL_L1HWPRF, L2D cache refill due to L1 hardware prefetch, event

Counts level 2 cache refills where the cache line is requested by a level 1 data cache hardware prefetcher.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L2_Cache](#)

0x4009 L2D_CACHE_LMISS_RD, Level 2 data cache long-latency read miss, event

Counts cache line refills into the level 2 unified cache from any memory read operations that incurred additional latency.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L2_Cache](#)

0x400A L2I_CACHE_LMISS, Level 2 instruction cache long-latency miss, event

Counts cache line refills into the level 2 unified cache from any instruction read operations that incurred additional latency.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L2_Cache](#)

0x8148 L2D_CACHE_RW, Level 2 data cache demand access, event

Counts level 2 cache demand accesses from any load/store operations. Level 2 cache is a unified cache for data and instruction accesses, accesses are for misses in the level 1 data cache or translation resolutions due to accesses.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L2_Cache](#)

0x8149 L2I_CACHE_RD, Level 2 instruction cache demand fetch, event

Counts level 2 cache accesses that are due to a demand instruction cache access.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L2_Cache](#)

0x814A L2D_CACHE_PRFM, Level 2 data cache software preload, event

Counts level 2 data cache accesses generated by software preload or prefetch instructions.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L2_Cache](#)

0x814B L2I_CACHE_PRFM, Level 2 instruction cache software preload, event

Counts level 2 instruction cache accesses generated by software preload or prefetch instructions.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L2_Cache](#)

0x814C L2D_CACHE_MISS, Level 2 data cache demand access miss, event

Counts cache line misses in the level 2 cache. Level 2 cache is a unified cache for data and instruction accesses. Accesses are for misses in the level 1 data cache or translation resolutions due to accesses.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L2_Cache](#)

0x814E L2D_CACHE_REFILL_PRFM, Level 2 data cache refill, software preload, event

Counts refills due to accesses generated as a result of software preload or prefetch instructions as counted by L2D_CACHE_PRFM.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L2_Cache](#)

0x814F L2I_CACHE_REFILL_PRFM, Level 2 instruction cache refill, software preload, event

Counts level 2 instruction cache refills generated by software preload or prefetch instructions.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L2_Cache](#)

0x8155 L2D_CACHE_HWPRF, Level 2 data cache hardware prefetch, event

Counts level 2 data cache accesses generated by L2D hardware prefetchers.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L2_Cache](#)

0x81BD L2D_CACHE_REFILL_HWPRF, Level 2 data cache refill, hardware prefetch, event

Counts level 2 data cache refills where the cache line is requested by a hardware prefetcher.

Related telemetry artifacts**Metrics**

- [l2_prefetcher_coverage_l1hwprf_inclusive](#)
- [l2_prefetcher_accuracy_l1hwprf_inclusive](#)
- [l2_prefetcher_accuracy_l1hwprf_exclusive](#)

Metric groups

[Prefetcher_Effectiveness](#)

Functional groups

[L2_Cache](#)

0x81C1 L2I_CACHE_HIT_RD, Level 2 instruction cache demand fetch hit, event

Counts level 2 cache hits that are due to a demand instruction cache access.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L2_Cache](#)

0x81ED L2D_CACHE_HIT_RW_FHWPRF, Level 2 data cache demand access first hit, fetched by hardware prefetcher, event

Counts first level 2 data demand cache hit from any load or store operation where the cache line was fetched by a hardware prefetcher. For the L2D_CACHE_HIT events, an L2 cache hit is only counted if the operation is satisfied internally by L2 (i.e. no bus request, no snoop to lower level).

Related telemetry artifacts**Metrics**

- [l2_prefetcher_coverage_l1hwprf_exclusive](#)
- [l2_prefetcher_accuracy_l1hwprf_exclusive](#)
- [l2_prefetcher_timeliness_l1hwprf_exclusive](#)

Metric groups

[Prefetcher_Effectiveness](#)

Functional groups

[L2_Cache](#)

0x8209 L2I_CACHE_HIT_PRFM, Level 2 instruction cache software preload hit, event

Counts level 2 instruction cache hits generated by software preload or prefetch instructions.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L2_Cache](#)

0x824D L2D_LFB_HIT_RW, Level 2 data cache demand access line-fill buffer hit, event

Counts load or store data accesses that access the level 2 data cache and hit in a line that is in the process of being loaded into the level 2 data cache.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L2_Cache](#)

0x826D L2D_LFB_HIT_RW_FHWPRF, Level 2 data cache demand access line-fill buffer first hit, recently fetched by hardware prefetcher, event

Counts first load or store data access that accesses the level 2 cache and hit in a line that is in the process of being loaded into the level 2 cache.

Related telemetry artifacts**Metrics**

- [l2_prefetcher_coverage_l1hwprf_exclusive](#)
- [l2_prefetcher_accuracy_l1hwprf_exclusive](#)

- [l2_prefetcher_timeliness_l1hwprf_exclusive](#)

Metric groups[Prefetcher_Effectiveness](#)**Functional groups**[L2_Cache](#)**0x8285 L2D_CACHE_PRF, Level 2 data cache, preload or prefetch hit, event**

Counts level 2 data cache accesses from software preload or prefetch instructions or hardware prefetcher.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[L2_Cache](#)

6.7 L3_Cache (L3 CACHE) events for C1-Premium

L3 unified cache related events.

Summary of events in L3_Cache:

- Total implemented Common events: 10
- Total Implemented Product ImpDef events: 0
- PMU Only events : 0
- ETE Only events : 0

Table 6-7: L3_Cache events summary

Code	Mnemonic	Name	Description
0x0029	L3D_CACHE_ALLOCATE	Level 3 data cache allocation without refill	Counts level 3 cache line allocates that do not fetch data from outside the level 3 data or...
0x002A	L3D_CACHE_REFILL	Level 3 data cache refill	Counts level 3 accesses that receive data from outside the L3 cache.
0x002B	L3D_CACHE	Level 3 data cache access	Counts level 3 cache accesses. Level 3 cache is a unified cache for data and instruction...
0x00A0	L3D_CACHE_RD	Level 3 data cache access, read	Counts level 3 cache accesses caused by any memory read operation. Level 3 cache is a unified...
0x400B	L3D_CACHE_LMISS_RD	Level 3 data cache long-latency read miss	Counts any cache line refill into the level 3 cache from memory read operations that incurred...
0x8150	L3D_CACHE_RW	Level 3 data cache demand access	Counts level 3 cache accesses caused by any memory read or write operation. Level 3 cache is a...
0x8151	L3D_CACHE_PRFM	Level 3 data cache software preload	Counts level 3 cache accesses generated by hardware or software prefetches.

Code	Mnemonic	Name	Description
0x8152	L3D_CACHE_MISS	Level 3 data cache demand access miss	Counts level 3 cache accesses that missed in the level 3 cache.
0x8153	L3D_CACHE_REFILL_PRFM	Level 3 data cache refill, software preload	Counts cacheable reads generated by hardware or software prefetches that receive data from...
0x8206	L3D_CACHE_HIT	Level 3 data cache hit	Counts each access counted by L3D_CACHE that hits in the Level 3 cache. Level 3 cache is a...

For a complete list of the events in C1-Premium, see [PMU events cheat sheet for C1-Premium](#) and [PMU events lookup table for C1-Premium](#).

0x0029 L3D_CACHE_ALLOCATE, Level 3 data cache allocation without refill, event

Counts level 3 cache line allocates that do not fetch data from outside the level 3 data or unified cache. For example, allocates due to streaming stores.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[L3_Cache](#)

0x002A L3D_CACHE_REFILL, Level 3 data cache refill, event

Counts level 3 accesses that receive data from outside the L3 cache.

Related telemetry artifacts

Metrics

- [l3_cache_mpki](#) in [L3_Cache_Effectiveness](#)
- [l3_cache_mpki](#) in [MPKI](#)
- [l3_cache_miss_ratio](#) in [L3_Cache_Effectiveness](#)
- [l3_cache_miss_ratio](#) in [Miss_Ratio](#)

Metric groups

[L3_Cache_Effectiveness](#)

[MPKI](#)

[Miss_Ratio](#)

Functional groups

[L3_Cache](#)

0x002B L3D_CACHE, Level 3 data cache access, event

Counts level 3 cache accesses. Level 3 cache is a unified cache for data and instruction accesses. Accesses are for misses in the lower level caches or translation resolutions due to accesses.

Related telemetry artifacts

Metrics

- [l3_cache_miss_ratio](#) in [L3_Cache_Effectiveness](#)

- [l3_cache_miss_ratio](#) in [Miss_Ratio](#)

Metric groups[L3_Cache_Effectiveness](#)[Miss_Ratio](#)**Functional groups**[L3_Cache](#)**0x00A0 L3D_CACHE_RD, Level 3 data cache access, read, event**

Counts level 3 cache accesses caused by any memory read operation. Level 3 cache is a unified cache for data and instruction accesses. Accesses are for misses in the lower level caches or translation resolutions due to accesses.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[L3_Cache](#)**0x400B L3D_CACHE_LMISS_RD, Level 3 data cache long-latency read miss, event**

Counts any cache line refill into the level 3 cache from memory read operations that incurred additional latency.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[L3_Cache](#)**0x8150 L3D_CACHE_RW, Level 3 data cache demand access, event**

Counts level 3 cache accesses caused by any memory read or write operation. Level 3 cache is a unified cache for data and instruction accesses. Accesses are for misses in the lower level caches or translation resolutions due to accesses.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[L3_Cache](#)**0x8151 L3D_CACHE_PRFM, Level 3 data cache software preload, event**

Counts level 3 cache accesses generated by hardware or software prefetches.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[L3_Cache](#)**0x8152 L3D_CACHE_MISS, Level 3 data cache demand access miss, event**

Counts level 3 cache accesses that missed in the level 3 cache.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[L3_Cache](#)**0x8153 L3D_CACHE_REFILL_PRFM, Level 3 data cache refill, software preload, event**

Counts cacheable reads generated by hardware or software prefetches that receive data from outside the L3 cache.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[L3_Cache](#)**0x8206 L3D_CACHE_HIT, Level 3 data cache hit, event**

Counts each access counted by L3D_CACHE that hits in the Level 3 cache. Level 3 cache is a unified cache for data and instruction accesses. Accesses are for misses in the lower level caches or translation resolutions due to accesses.

Related telemetry artifacts**Metrics**

- [system_l3_cache_hit_ratio](#)

Metric groups[System_Memory_Effectiveness](#)**Functional groups**[L3_Cache](#)

6.8 LL_Cache (LL CACHE) events for C1-Premium

Last Level Cache related events.

Summary of events in LL_Cache:

- Total implemented Common events: 4
- Total Implemented Product ImpDef events: 0
- PMU Only events : 0

- ETE Only events : 0

Table 6-8: LL_Cache events summary

Code	Mnemonic	Name	Description
0x0032	LL_CACHE	Last level cache access	Counts transactions that were returned from outside the core cluster. This event counts...
0x0036	LL_CACHE_RD	Last level cache access, read	Counts read transactions that were returned from outside the core cluster. This event counts for...
0x0037	LL_CACHE_MISS_RD	Last level cache miss, read	Counts read transactions that were returned from outside the core cluster but missed in the...
0x8207	LL_CACHE_HIT	Last level cache hit	Counts each access counted by LL_CACHE that hits in the Last level cache. This event counts...

For a complete list of the events in C1-Premium, see [PMU events cheat sheet for C1-Premium](#) and [PMU events lookup table for C1-Premium](#).

0x0032 LL_CACHE, Last level cache access, event

Counts transactions that were returned from outside the core cluster. This event counts transactions for external last level cache when the system register CPUECTLR.EXTLLC bit is set.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[LL_Cache](#)

0x0036 LL_CACHE_RD, Last level cache access, read, event

Counts read transactions that were returned from outside the core cluster. This event counts for external last level cache when the system register CPUECTLR.EXTLLC bit is set. This event counts read transactions returned from outside the core if those transactions are either hit in the system level cache or missed in the SLC and are returned from any other external sources.

Related telemetry artifacts

Metrics

- [ll_cache_read_miss_ratio](#) in [LL_Cache_Effectiveness](#)
- [ll_cache_read_miss_ratio](#) in [Miss_Ratio](#)
- [ll_cache_read_hit_ratio](#)

Metric groups

[LL_Cache_Effectiveness](#)

[Miss_Ratio](#)

Functional groups

[LL_Cache](#)

0x0037 LL_CACHE_MISS_RD, Last level cache miss, read, event

Counts read transactions that were returned from outside the core cluster but missed in the system level cache. This event counts for external last level cache when the system register CPUECTLR.EXTLLC bit is set. This event counts read transactions returned from outside the core if those transactions are missed in the System level Cache. The data source of the transaction is indicated by a field in the CHI transaction returning to the CPU. This event does not count reads caused by cache maintenance operations.

Related telemetry artifacts**Metrics**

- [ll_cache_read_mpki](#) in [LL_Cache_Effectiveness](#)
- [ll_cache_read_mpki](#) in [MPKI](#)
- [ll_cache_read_miss_ratio](#) in [LL_Cache_Effectiveness](#)
- [ll_cache_read_miss_ratio](#) in [Miss_Ratio](#)
- [ll_cache_read_hit_ratio](#)

Metric groups[LL_Cache_Effectiveness](#)[MPKI](#)[Miss_Ratio](#)**Functional groups**[LL_Cache](#)**0x8207 LL_CACHE_HIT, Last level cache hit, event**

Counts each access counted by LL_CACHE that hits in the Last level cache. This event counts transactions for external last level cache when the system register CPUECTLR.EXTLLC bit is set.

Related telemetry artifacts**Metrics**

- [system_llc_cache_hit_ratio](#)

Metric groups[System_Memory_Effectiveness](#)**Functional groups**[LL_Cache](#)

6.9 Memory (MEMORY) events for C1-Premium

Memory system related events.

Summary of events in Memory:

- Total implemented Common events: 14

- Total Implemented Product ImpDef events: 1
- PMU Only events : 1
- ETE Only events : 0

Table 6-9: Memory events summary

Code	Mnemonic	Name	Description
0x0013	MEM_ACCESS	Data memory access	Counts memory accesses issued by the CPU load store unit, where those accesses are issued due to...
0x001A	MEMORY_ERROR	Local memory error	Counts any detected correctable or uncorrectable physical memory errors (ECC or parity) in...
0x0031	REMOTE_ACCESS	Access to another socket in a multi-socket system	Counts accesses to another chip, which is implemented as a different CMN mesh in the system. If...
0x0066	MEM_ACCESS_RD	Data memory access, read	Counts memory accesses issued by the CPU due to load operations. The event counts any memory load...
0x0067	MEM_ACCESS_WR	Data memory access, write	Counts memory accesses issued by the CPU due to store operations. The event counts any memory...
0x3008	IMP_DRAM_ACCESS	Count of loads that were completed at DRAM	Counts access where the data was sourced from the DRAM.
0x4020	LDST_ALIGN_LAT	Access with additional latency from alignment	Counts the number of memory read and write accesses in a cycle that incurred additional latency,...
0x4021	LD_ALIGN_LAT	Load with additional latency from alignment	Counts the number of memory read accesses in a cycle that incurred additional latency, due to the...
0x4022	ST_ALIGN_LAT	Store with additional latency from alignment	Counts the number of memory write access in a cycle that incurred additional latency, due to the...
0x4024	MEM_ACCESS_CHECKED	Checked data memory access	Counts the number of memory read and write accesses counted by MEM_ACCESS that are tag checked by...
0x4025	MEM_ACCESS_CHECKED_RD	Checked data memory access, read	Counts the number of memory read accesses in a cycle that are tag checked by the Memory Tagging...
0x4026	MEM_ACCESS_CHECKED_WR	Checked data memory access, write	Counts the number of memory write accesses in a cycle that is tag checked by the Memory Tagging...
0x8120	INST_FETCH_PERCYC	Event in progress, INST FETCH	Counts number of instruction fetches outstanding per cycle, which will provide an average latency...
0x8121	MEM_ACCESS_RD_PERCYC	Event in progress, MEM ACCESS RD	Counts the number of outstanding loads or memory read accesses per cycle.
0x8124	INST_FETCH	Instruction memory access	Counts Instruction memory accesses that the PE makes.

For a complete list of the events in C1-Premium, see [PMU events cheat sheet for C1-Premium](#) and [PMU events lookup table for C1-Premium](#).

0x0013 MEM_ACCESS, Data memory access, event

Counts memory accesses issued by the CPU load store unit, where those accesses are issued due to load or store operations. This event counts memory accesses no matter whether the data is received from any level of cache hierarchy or external memory. If memory accesses are broken up into smaller transactions than what were specified in the load or store instructions, then the event counts those smaller memory transactions.

Related telemetry artifacts

Metrics

- [load_average_latency](#)

Metric groups

[Average_Latency](#)

Functional groups

[Memory](#)

0x001A MEMORY_ERROR, Local memory error, event

Counts any detected correctable or uncorrectable physical memory errors (ECC or parity) in protected CPUs RAMs. On the core, this event counts errors in the caches (including data and tag rams). Any detected memory error (from either a speculative and abandoned access, or an architecturally executed access) is counted. Note that errors are only detected when the actual protected memory is accessed by an operation.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Memory](#)

0x0031 REMOTE_ACCESS, Access to another socket in a multi-socket system, event

Counts accesses to another chip, which is implemented as a different CMN mesh in the system. If the CHI bus response back to the core indicates that the data source is from another chip (mesh), then the counter is updated. If no data is returned, even if the system snoops another chip/mesh, then the counter is not updated.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Memory](#)

0x0066 MEM_ACCESS_RD, Data memory access, read, event

Counts memory accesses issued by the CPU due to load operations. The event counts any memory load access, no matter whether the data is received from any level of cache hierarchy or external memory. The event also counts atomic load operations. If memory accesses are broken up by the load/store unit into smaller transactions that are issued by the bus interface, then the event counts those smaller transactions.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Memory](#)

0x0067 MEM_ACCESS_WR, Data memory access, write, event

Counts memory accesses issued by the CPU due to store operations. The event counts any memory store access, no matter whether the data is located in any level of cache or external memory. The event also counts atomic load and store operations. If memory accesses are broken up by the load/store unit into smaller transactions that are issued by the bus interface, then the event counts those smaller transactions.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Memory](#)

0x3008 IMP_DRAM_ACCESS, Count of loads that were completed at DRAM, event

Counts access where the data was sourced from the DRAM.

Related telemetry artifacts**Metrics**

- [system_dram_mem_hit_ratio](#)

Metric groups

[System_Memory_Effectiveness](#)

Functional groups

[Memory](#)

0x4020 LDST_ALIGN_LAT, Access with additional latency from alignment, event

Counts the number of memory read and write accesses in a cycle that incurred additional latency, due to the alignment of the address and the size of data being accessed, which results in store crossing a single cache line.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Memory](#)

0x4021 LD_ALIGN_LAT, Load with additional latency from alignment, event

Counts the number of memory read accesses in a cycle that incurred additional latency, due to the alignment of the address and size of data being accessed, which results in load crossing a single cache line.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Memory](#)

0x4022 ST_ALIGN_LAT, Store with additional latency from alignment, event

Counts the number of memory write access in a cycle that incurred additional latency, due to the alignment of the address and size of data being accessed incurred additional latency.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Memory](#)

0x4024 MEM_ACCESS_CHECKED, Checked data memory access, event

Counts the number of memory read and write accesses counted by MEM_ACCESS that are tag checked by the Memory Tagging Extension (MTE). This event is implemented as the sum of MEM_ACCESS_CHECKED_RD and MEM_ACCESS_CHECKED_WR

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Memory](#)

0x4025 MEM_ACCESS_CHECKED_RD, Checked data memory access, read, event

Counts the number of memory read accesses in a cycle that are tag checked by the Memory Tagging Extension (MTE).

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Memory](#)

0x4026 MEM_ACCESS_CHECKED_WR, Checked data memory access, write, event

Counts the number of memory write accesses in a cycle that is tag checked by the Memory Tagging Extension (MTE).

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Memory](#)

0x8120 INST_FETCH_PERCYC, Event in progress, INST FETCH, event

Counts number of instruction fetches outstanding per cycle, which will provide an average latency of instruction fetch.

Related telemetry artifacts**Metrics**

- [instruction_fetch_average_latency](#)

Metric groups

[Average_Latency](#)

Functional groups

[Memory](#)

0x8121 MEM_ACCESS_RD_PERCYC, Event in progress, MEM ACCESS RD, event

Counts the number of outstanding loads or memory read accesses per cycle.

Related telemetry artifacts**Metrics**

- [load_average_latency](#)

Metric groups

[Average_Latency](#)

Functional groups

[Memory](#)

0x8124 INST_FETCH, Instruction memory access, event

Counts Instruction memory accesses that the PE makes.

Related telemetry artifacts**Metrics**

- [instruction_fetch_average_latency](#)

Metric groups

[Average_Latency](#)

Functional groups

[Memory](#)

6.10 Retired (RETIRED) events for C1-Premium

Retired instruction and operation events.

Summary of events in Retired:

- Total implemented Common events: 35
- Total Implemented Product ImpDef events: 0
- PMU Only events : 0
- ETE Only events : 0

Table 6-10: Retired events summary

Code	Mnemonic	Name	Description
0x0000	SW_INCR	Instruction architecturally executed, Condition code check pass, software increment	Counts software writes to the PMSWINC_ELO (software PMU increment) register. The PMSWINC_ELO...
0x0008	INST_RETIRED	Instruction architecturally executed	Counts instructions that have been architecturally executed.
0x000B	CID_WRITE_RETIRED	Instruction architecturally executed, Condition code check pass, write to CONTEXTIDR	Counts architecturally executed writes to the CONTEXTIDR_EL1 register, which usually contain the...
0x000C	PC_WRITE_RETIRED	Instruction architecturally executed, Condition code check pass, Software change of the PC	Counts branch instructions that caused a change of Program Counter, which effectively causes a...
0x000D	BR_IMMED_RETIRED	Branch instruction architecturally executed, immediate	Counts architecturally executed direct branches.
0x000E	BR_RETURN_RETIRED	Branch instruction architecturally executed, procedure return, taken	Counts architecturally executed procedure returns.
0x001C	TTBR_WRITE_RETIRED	Instruction architecturally executed, Condition code check pass, write to TTBR	Counts architectural writes to TTBR0/1_EL1. If virtualization host extensions are enabled (by...
0x0021	BR_RETIRED	Instruction architecturally executed, branch	Counts architecturally executed branches, whether the branch is taken or not. Instructions that...
0x0022	BR_MIS_PRED_RETIRED	Branch instruction architecturally executed, mispredicted	Counts branches counted by BR_RETIRED which were mispredicted and caused a pipeline flush.
0x003A	OP_RETIRED	Micro-operation architecturally executed	Counts micro-operations that are architecturally executed. This is a count of number of...
0x8108	BR_IMMED_TAKEN_RETIRED	Branch instruction architecturally executed, immediate, taken	Counts architecturally executed direct branches that were taken.
0x810C	BR_INDNR_TAKEN_RETIRED	Branch instruction architecturally executed, indirect excluding procedure return, taken	Counts architecturally executed indirect branches excluding procedure returns that were taken.
0x8110	BR_IMMED_PRED_RETIRED	Branch instruction architecturally executed, predicted immediate	Counts architecturally executed direct branches that were correctly predicted.
0x8111	BR_IMMED_MIS_PRED_RETIRED	Branch instruction architecturally executed, mispredicted immediate	Counts architecturally executed direct branches that were mispredicted and caused a pipeline flush.
0x8112	BR_IND_PRED_RETIRED	Branch instruction architecturally executed, predicted indirect	Counts architecturally executed indirect branches including procedure returns that were correctly...
0x8113	BR_IND_MIS_PRED_RETIRED	Branch instruction architecturally executed, mispredicted indirect	Counts architecturally executed indirect branches including procedure returns that were...
0x8114	BR_RETURN_PRED_RETIRED	Branch instruction architecturally executed, predicted procedure return	Counts architecturally executed procedure returns that were correctly predicted.
0x8115	BR_RETURN_MIS_PRED_RETIRED	Branch instruction architecturally executed, mispredicted procedure return	Counts architecturally executed procedure returns that were mispredicted and caused a pipeline...

Code	Mnemonic	Name	Description
0x8116	BR_INDNR_PRED_RETIRE	Branch instruction architecturally executed, predicted indirect excluding procedure return	Counts architecturally executed indirect branches excluding procedure returns that were correctly...
0x8117	BR_INDNR_MIS_PRED_RETIRE	Branch instruction architecturally executed, mispredicted indirect excluding procedure return	Counts architecturally executed indirect branches excluding procedure returns that were...
0x8118	BR_TAKEN_PRED_RETIRE	Branch instruction architecturally executed, predicted branch, taken	Counts architecturally executed branches that were taken and were correctly predicted.
0x8119	BR_TAKEN_MIS_PRED_RETIRE	Branch instruction architecturally executed, mispredicted branch, taken	Counts architecturally executed branches that were taken and were mispredicted causing a pipeline...
0x811A	BR_SKIP_PRED_RETIRE	Branch instruction architecturally executed, predicted branch, not taken	Counts architecturally executed branches that were not taken and were correctly predicted.
0x811B	BR_SKIP_MIS_PRED_RETIRE	Branch instruction architecturally executed, mispredicted branch, not taken	Counts architecturally executed branches that were not taken and were mispredicted causing a...
0x811C	BR_PRED_RETIRE	Branch instruction architecturally executed, predicted branch	Counts branch instructions counted by BR_RETIRE which were correctly predicted.
0x811D	BR_IND_RETIRE	Instruction architecturally executed, indirect branch	Counts architecturally executed indirect branches including procedure returns.
0x8179	BRNL_INDNR_TAKEN_RETIRE	Branch instruction architecturally executed, indirect branch without link excluding procedure return, taken	Counts architecturally executed indirect branch without link instructions, excluding return...
0x817A	BL_TAKEN_RETIRE	Branch instruction architecturally executed, branch with link, taken	Counts architecturally executed branch with link instructions that were taken.
0x817B	BRNL_TAKEN_RETIRE	Branch instruction architecturally executed, branch without link, taken	Counts architecturally executed branch without link instructions that were taken.
0x817C	BL_IND_TAKEN_RETIRE	Branch instruction architecturally executed, indirect branch with link, taken	Counts architecturally executed indirect branch with link instructions that were taken.
0x817D	BRNL_IND_TAKEN_RETIRE	Branch instruction architecturally executed, indirect branch without link, taken	Counts architecturally executed indirect branch without link instructions that were taken.
0x817E	BL_IMMED_TAKEN_RETIRE	Branch instruction architecturally executed, direct branch with link, taken	Counts architecturally executed direct branch with link instructions that were taken.
0x817F	BRNL_IMMED_TAKEN_RETIRE	Branch instruction architecturally executed, direct branch without link, taken	Counts architecturally executed direct branch without link instructions that were taken.
0x8180	BR_UNCOND_RETIRE	Branch instruction architecturally executed, unconditional branch	Counts architecturally executed unconditional branch instructions.
0x8181	BR_COND_RETIRE	Branch instruction architecturally executed, conditional branch	Counts architecturally executed conditional branch instructions.

For a complete list of the events in C1-Premium, see [PMU events cheat sheet for C1-Premium](#) and [PMU events lookup table for C1-Premium](#).

0x0000 SW_INCR, Instruction architecturally executed, Condition code check pass, software increment, event

Counts software writes to the PMSWINC_ELO (software PMU increment) register. The PMSWINC_ELO register is a manually updated counter for use by application software.

This event could be used to measure any user program event, such as accesses to a particular data structure (by writing to the PMSWINC_ELO register each time the data structure is accessed).

To use the PMSWINC_ELO register and event, developers must insert instructions that write to the PMSWINC_ELO register into the source code.

Since the SW_INCR event records writes to the PMSWINC_ELO register, there is no need to do a read/increment/write sequence to the PMSWINC_ELO register.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Retired](#)

0x0008 INST_RETIRED, Instruction architecturally executed, event

Counts instructions that have been architecturally executed.

Related telemetry artifacts**Metrics**

- [ipc](#)
- [branch_mpki](#) in [Branch_Effectiveness](#)
- [branch_mpki](#) in [MPKI](#)
- [itlb_mpki](#) in [ITLB_Effectiveness](#)
- [itlb_mpki](#) in [MPKI](#)
- [l1i_tlb_mpki](#) in [ITLB_Effectiveness](#)
- [l1i_tlb_mpki](#) in [MPKI](#)
- [dtlb_mpki](#) in [DTLB_Effectiveness](#)
- [dtlb_mpki](#) in [MPKI](#)
- [l1d_tlb_mpki](#) in [DTLB_Effectiveness](#)
- [l1d_tlb_mpki](#) in [MPKI](#)
- [l2_tlb_mpki](#) in [DTLB_Effectiveness](#)
- [l2_tlb_mpki](#) in [ITLB_Effectiveness](#)
- [l2_tlb_mpki](#) in [MPKI](#)
- [l1i_cache_mpki](#) in [L1I_Cache_Effectiveness](#)
- [l1i_cache_mpki](#) in [MPKI](#)

- [l1d_cache_demand_mpki](#) in [L1D_Cache_Effectiveness](#)
- [l1d_cache_demand_mpki](#) in [MPKI](#)
- [l1d_cache_mpki](#) in [L1D_Cache_Effectiveness](#)
- [l1d_cache_mpki](#) in [MPKI](#)
- [l2i_cache_mpki](#) in [L2I_Cache_Effectiveness](#)
- [l2i_cache_mpki](#) in [MPKI](#)
- [l2d_cache_demand_mpki](#) in [L2D_Cache_Effectiveness](#)
- [l2d_cache_demand_mpki](#) in [MPKI](#)
- [l2d_cache_mpki](#) in [L2D_Cache_Effectiveness](#)
- [l2d_cache_mpki](#) in [MPKI](#)
- [l3_cache_mpki](#) in [L3_Cache_Effectiveness](#)
- [l3_cache_mpki](#) in [MPKI](#)
- [ll_cache_read_mpki](#) in [LL_Cache_Effectiveness](#)
- [ll_cache_read_mpki](#) in [MPKI](#)

Metric groups

[Branch_Effectiveness](#)
[DTLB_Effectiveness](#)
[General](#)
[ITLB_Effectiveness](#)
[L1D_Cache_Effectiveness](#)
[L1I_Cache_Effectiveness](#)
[L2D_Cache_Effectiveness](#)
[L2I_Cache_Effectiveness](#)
[L3_Cache_Effectiveness](#)
[LL_Cache_Effectiveness](#)
[MPKI](#)

Functional groups

[Retired](#)

0x000B CID_WRITE_RETIRED, Instruction architecturally executed, Condition code check pass, write to CONTEXTIDR, event

Counts architecturally executed writes to the CONTEXTIDR_EL1 register, which usually contain the kernel PID and can be output with hardware trace.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Retired](#)

0x000c PC_WRITE_RETIRED, Instruction architecturally executed, Condition code check pass, Software change of the PC, event

Counts branch instructions that caused a change of Program Counter, which effectively causes a change in the control flow of the program.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Retired](#)

0x000d BR_IMMED_RETIRED, Branch instruction architecturally executed, immediate, event

Counts architecturally executed direct branches.

Related telemetry artifacts**Metrics**

- [branch_direct_ratio](#)

Metric groups

[Branch_Effectiveness](#)

Functional groups

[Retired](#)

0x000e BR_RETURN_RETIRED, Branch instruction architecturally executed, procedure return, taken, event

Counts architecturally executed procedure returns.

Related telemetry artifacts**Metrics**

- [branch_return_ratio](#)

Metric groups

[Branch_Effectiveness](#)

Functional groups

[Retired](#)

0x001c TTBR_WRITE_RETIRED, Instruction architecturally executed, Condition code check pass, write to TTBR, event

Counts architectural writes to TTBR0/1_EL1. If virtualization host extensions are enabled (by setting the HCR_EL2.E2H bit to 1), then accesses to TTBR0/1_EL1 that are redirected to TTBR0/1_EL2, or accesses to TTBR0/1_EL12, are counted. TTBRn registers are typically updated when the kernel is swapping user-space threads or applications.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Retired](#)

0x0021 BR_RETIREd, Instruction architecturally executed, branch, event

Counts architecturally executed branches, whether the branch is taken or not. Instructions that explicitly write to the PC are also counted. Note that exception generating instructions, exception return instructions and context synchronization instructions are not counted.

Related telemetry artifacts

Metrics

- [branch_direct_ratio](#)
- [branch_indirect_ratio](#)
- [branch_return_ratio](#)
- [branch_misprediction_ratio](#) in [Branch_Effectiveness](#)
- [branch_misprediction_ratio](#) in [Miss_Ratio](#)

Metric groups

[Branch_Effectiveness](#)

[Miss_Ratio](#)

Functional groups

[Retired](#)

0x0022 BR_MIS_PRED_RETIREd, Branch instruction architecturally executed, mispredicted, event

Counts branches counted by BR_RETIREd which were mispredicted and caused a pipeline flush.

Related telemetry artifacts

Metrics

- [branch_mpki](#) in [Branch_Effectiveness](#)
- [branch_mpki](#) in [MPKI](#)
- [branch_misprediction_ratio](#) in [Branch_Effectiveness](#)
- [branch_misprediction_ratio](#) in [Miss_Ratio](#)

Metric groups

[Branch_Effectiveness](#)

[MPKI](#)

[Miss_Ratio](#)

Functional groups

[Retired](#)

0x003A OP_RETIRE, Micro-operation architecturally executed, event

Counts micro-operations that are architecturally executed. This is a count of number of micro-operations retired from the commit queue in a single cycle.

Related telemetry artifacts**Metrics**

- [retiring](#)
- [bad_speculation](#)

Metric groups

[Topdown_L1](#)

Functional groups

[Retired](#)

0x8108 BR_IMMED_TAKEN_RETIRE, Branch instruction architecturally executed, immediate, taken, event

Counts architecturally executed direct branches that were taken.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Retired](#)

0x810C BR_INDNR_TAKEN_RETIRE, Branch instruction architecturally executed, indirect excluding procedure return, taken, event

Counts architecturally executed indirect branches excluding procedure returns that were taken.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Retired](#)

0x8110 BR_IMMED_PRED_RETIRE, Branch instruction architecturally executed, predicted immediate, event

Counts architecturally executed direct branches that were correctly predicted.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Retired](#)

0x8111 BR_IMMED_MIS_PRED_RETIRE, Branch instruction architecturally executed, mispredicted immediate, event

Counts architecturally executed direct branches that were mispredicted and caused a pipeline flush.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

Retired

0x8112 BR_IND_PRED_RETIRE, Branch instruction architecturally executed, predicted indirect, event

Counts architecturally executed indirect branches including procedure returns that were correctly predicted.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

Retired

0x8113 BR_IND_MIS_PRED_RETIRE, Branch instruction architecturally executed, mispredicted indirect, event

Counts architecturally executed indirect branches including procedure returns that were mispredicted and caused a pipeline flush.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

Retired

0x8114 BR_RETURN_PRED_RETIRE, Branch instruction architecturally executed, predicted procedure return, event

Counts architecturally executed procedure returns that were correctly predicted.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

Retired

0x8115 BR_RETURN_MIS_PRED_RETIRE, Branch instruction architecturally executed, mispredicted procedure return, event

Counts architecturally executed procedure returns that were mispredicted and caused a pipeline flush.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

Retired

0x8116 BR_INDNR_PRED_RETIRE, Branch instruction architecturally executed, predicted indirect excluding procedure return, event

Counts architecturally executed indirect branches excluding procedure returns that were correctly predicted.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

Retired

0x8117 BR_INDNR_MIS_PRED_RETIRE, Branch instruction architecturally executed, mispredicted indirect excluding procedure return, event

Counts architecturally executed indirect branches excluding procedure returns that were mispredicted and caused a pipeline flush.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

Retired

0x8118 BR_TAKEN_PRED_RETIRE, Branch instruction architecturally executed, predicted branch, taken, event

Counts architecturally executed branches that were taken and were correctly predicted.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

Retired

0x8119 BR_TAKEN_MIS_PRED_RETIRE, Branch instruction architecturally executed, mispredicted branch, taken, event

Counts architecturally executed branches that were taken and were mispredicted causing a pipeline flush.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

Retired

0x811A BR_SKIP_PRED_RETIRED, Branch instruction architecturally executed, predicted branch, not taken, event

Counts architecturally executed branches that were not taken and were correctly predicted.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

Retired

0x811B BR_SKIP_MIS_PRED_RETIRED, Branch instruction architecturally executed, mispredicted branch, not taken, event

Counts architecturally executed branches that were not taken and were mispredicted causing a pipeline flush.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

Retired

0x811C BR_PRED_RETIRED, Branch instruction architecturally executed, predicted branch, event

Counts branch instructions counted by BR_RETIRE which were correctly predicted.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

Retired

0x811D BR_IND_RETIRED, Instruction architecturally executed, indirect branch, event

Counts architecturally executed indirect branches including procedure returns.

Related telemetry artifacts

Metrics

- [branch_indirect_ratio](#)

Metric groups

[Branch_Effectiveness](#)

Functional groups

Retired

0x8179 BRNL_INDNR_TAKEN_RETIREd, Branch instruction architecturally executed, indirect branch without link excluding procedure return, taken, event

Counts architecturally executed indirect branch without link instructions, excluding return instructions, that were taken.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Retired](#)

0x817A BL_TAKEN_RETIREd, Branch instruction architecturally executed, branch with link, taken, event

Counts architecturally executed branch with link instructions that were taken.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Retired](#)

0x817B BRNL_TAKEN_RETIREd, Branch instruction architecturally executed, branch without link, taken, event

Counts architecturally executed branch without link instructions that were taken.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Retired](#)

0x817C BL_IND_TAKEN_RETIREd, Branch instruction architecturally executed, indirect branch with link, taken, event

Counts architecturally executed indirect branch with link instructions that were taken.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Retired](#)

0x817D BRNL_IND_TAKEN_RETIREd, Branch instruction architecturally executed, indirect branch without link, taken, event

Counts architecturally executed indirect branch without link instructions that were taken.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Retired](#)

0x817E BL_IMMED_TAKEN_RETIRE, Branch instruction architecturally executed, direct branch with link, taken, event

Counts architecturally executed direct branch with link instructions that were taken.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Retired](#)

0x817F BRNL_IMMED_TAKEN_RETIRE, Branch instruction architecturally executed, direct branch without link, taken, event

Counts architecturally executed direct branch without link instructions that were taken.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Retired](#)

0x8180 BR_UNCOND_RETIRE, Branch instruction architecturally executed, unconditional branch, event

Counts architecturally executed unconditional branch instructions.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Retired](#)

0x8181 BR_COND_RETIRE, Branch instruction architecturally executed, conditional branch, event

Counts architecturally executed conditional branch instructions.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Retired](#)

6.11 SPE (SPE) events for C1-Premium

SPE related events.

Summary of events in SPE:

- Total implemented Common events: 10
- Total Implemented Product ImpDef events: 0
- PMU Only events : 0
- ETE Only events : 0

Table 6-11: SPE events summary

Code	Mnemonic	Name	Description
0x4000	SAMPLE_POP	Statistical Profiling sample population	Counts statistical profiling sample population, the count of all operations that could be sampled...
0x4001	SAMPLE_FEED	Statistical Profiling sample taken	Counts statistical profiling samples taken for sampling.
0x4002	SAMPLE_FILTRATE	Statistical Profiling sample taken and not removed by filtering	Counts statistical profiling samples taken which are not removed by filtering.
0x4003	SAMPLE_COLLISION	Statistical Profiling sample collided with previous sample	Counts statistical profiling samples that have collided with a previous sample and so therefore...
0x812A	SAMPLE_FEED_BR	Statistical Profiling sample taken, branch	Counts statistical profiling samples taken which are branches.
0x812B	SAMPLE_FEED_LD	Statistical Profiling sample taken, load	Counts statistical profiling samples taken which are loads or load atomic operations.
0x812C	SAMPLE_FEED_ST	Statistical Profiling sample taken, store	Counts statistical profiling samples taken which are stores or store atomic operations.
0x812D	SAMPLE_FEED_OP	Statistical Profiling sample taken, matching operation type	Counts statistical profiling samples taken which are matching any operation type filters supported.
0x812E	SAMPLE_FEED_EVENT	Statistical Profiling sample taken, matching events	Counts statistical profiling samples taken which are matching event packet filter constraints.
0x812F	SAMPLE_FEED_LAT	Statistical Profiling sample taken, exceeding minimum latency	Counts statistical profiling samples taken which are exceeding minimum latency set by operation...

For a complete list of the events in C1-Premium, see [PMU events cheat sheet for C1-Premium](#) and [PMU events lookup table for C1-Premium](#).

0x4000 [SAMPLE_POP](#), Statistical Profiling sample population, event

Counts statistical profiling sample population, the count of all operations that could be sampled but may or may not be chosen for sampling.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[SPE](#)

0x4001 SAMPLE_FEED, Statistical Profiling sample taken, event

Counts statistical profiling samples taken for sampling.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[SPE](#)

0x4002 SAMPLE_FILTRATE, Statistical Profiling sample taken and not removed by filtering, event

Counts statistical profiling samples taken which are not removed by filtering.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[SPE](#)

0x4003 SAMPLE_COLLISION, Statistical Profiling sample collided with previous sample, event

Counts statistical profiling samples that have collided with a previous sample and so therefore not taken.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[SPE](#)

0x812A SAMPLE_FEED_BR, Statistical Profiling sample taken, branch, event

Counts statistical profiling samples taken which are branches.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[SPE](#)

0x812B SAMPLE_FEED_LD, Statistical Profiling sample taken, load, event

Counts statistical profiling samples taken which are loads or load atomic operations.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[SPE](#)**0x812C SAMPLE_FEED_ST, Statistical Profiling sample taken, store, event**

Counts statistical profiling samples taken which are stores or store atomic operations.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[SPE](#)**0x812D SAMPLE_FEED_OP, Statistical Profiling sample taken, matching operation type, event**

Counts statistical profiling samples taken which are matching any operation type filters supported.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[SPE](#)**0x812E SAMPLE_FEED_EVENT, Statistical Profiling sample taken, matching events, event**

Counts statistical profiling samples taken which are matching event packet filter constraints.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[SPE](#)**0x812F SAMPLE_FEED_LAT, Statistical Profiling sample taken, exceeding minimum latency, event**

Counts statistical profiling samples taken which are exceeding minimum latency set by operation latency filter constraints.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[SPE](#)

6.12 Spec_Operation (SPEC OPERATION) events for C1-Premium

Speculatively executed operations related events.

Summary of events in Spec_Operation:

- Total implemented Common events: 50
- Total Implemented Product ImpDef events: 3
- PMU Only events : 3
- ETE Only events : 0

Table 6-12: Spec_Operation events summary

Code	Mnemonic	Name	Description
0x0010	BR_MIS_PRED	Branch instruction speculatively executed, mispredicted or not predicted	Counts branches which are speculatively executed and mispredicted.
0x0012	BR_PRED	Predictable branch instruction speculatively executed	Counts all speculatively executed branches.
0x001B	INST_SPEC	Operation speculatively executed	Counts operations that have been speculatively executed.
0x003B	OP_SPEC	Micro-operation speculatively executed	Counts micro-operations speculatively executed. This is the count of the number of...
0x0068	UNALIGNED_LD_SPEC	Unaligned access, read	Counts unaligned memory read operations issued by the CPU. This event counts unaligned accesses...
0x0069	UNALIGNED_ST_SPEC	Unaligned access, write	Counts unaligned memory write operations issued by the CPU. This event counts unaligned accesses...
0x006A	UNALIGNED_LDST_SPEC	Unaligned access	Counts unaligned memory operations issued by the CPU. This event counts unaligned accesses (as...
0x006C	LDREX_SPEC	Exclusive operation speculatively executed, Load-Exclusive	Counts Load-Exclusive operations that have been speculatively executed. For example: LDREX, LDX
0x006D	STREX_PASS_SPEC	Exclusive operation speculatively executed, Store-Exclusive pass	Counts store-exclusive operations that have been speculatively executed and have successfully...
0x006E	STREX_FAIL_SPEC	Exclusive operation speculatively executed, Store-Exclusive fail	Counts store-exclusive operations that have been speculatively executed and have not successfully...
0x006F	STREX_SPEC	Exclusive operation speculatively executed, Store-Exclusive	Counts store-exclusive operations that have been speculatively executed.
0x0070	LD_SPEC	Operation speculatively executed, load	Counts speculatively executed load operations including Single Instruction Multiple Data (SIMD)...
0x0071	ST_SPEC	Operation speculatively executed, store	Counts speculatively executed store operations including Single Instruction Multiple Data (SIMD)...
0x0072	LDST_SPEC	Operation speculatively executed, load or store	Counts load and store operations that have been speculatively executed.
0x0073	DP_SPEC	Operation speculatively executed, integer data processing	Counts speculatively executed logical or arithmetic instructions such as MOV/MVN operations.
0x0074	ASE_SPEC	Operation speculatively executed, Advanced SIMD data processing	The counter counts each operation counted by INST_SPEC that is an Advanced SIMD...

Code	Mnemonic	Name	Description
0x0075	VFP_SPEC	Operation speculatively executed, scalar floating-point	Counts speculatively executed floating point operations. This event does not count operations...
0x0076	PC_WRITE_SPEC	Operation speculatively executed, Software change of the PC	Counts speculatively executed operations which cause software changes of the PC. Those operations...
0x0077	CRYPTO_SPEC	Operation speculatively executed, Cryptographic instruction	Counts speculatively executed cryptographic operations except for PMULL and VMULL operations.
0x0078	BR_IMMED_SPEC	Branch speculatively executed, immediate branch	Counts direct branch operations which are speculatively executed.
0x0079	BR_RETURN_SPEC	Branch speculatively executed, procedure return	Counts procedure return operations (RET, RETAA and RETAB) which are speculatively executed.
0x007A	BR_INDIRECT_SPEC	Branch speculatively executed, indirect branch	Counts indirect branch operations including procedure returns, which are speculatively executed....
0x007C	ISB_SPEC	Barrier speculatively executed, ISB	Counts ISB operations that are executed.
0x007D	DSB_SPEC	Barrier speculatively executed, DSB	Counts DSB operations that are speculatively issued to Load/Store unit in the CPU.
0x007E	DMB_SPEC	Barrier speculatively executed, DMB	Counts DMB operations that are speculatively issued to the Load/Store unit in the CPU. This event...
0x007F	CSDB_SPEC	Barrier speculatively executed, CSDB	Counts CSDB operations that are speculatively issued to the Load/Store unit in the CPU. This...
0x0090	RC_LD_SPEC	Release consistency operation speculatively executed, Load-Acquire	Counts any load acquire operations that are speculatively executed. For example: LDAR, LDARH, LDARB
0x0091	RC_ST_SPEC	Release consistency operation speculatively executed, Store-Release	Counts any store release operations that are speculatively executed. For example: STLR, STLRH, STLRB
0x3218	OP_CME_ISSUE	SME operation issued	The counter counts each operation that was issued to a streaming mode compute unit. The...
0x3219	SSVE_INST_SPEC	Operation speculatively executed, Streaming SVE, including load and store	The counter counts each instruction counted by SVE_INST_SPEC when the CPU executes in Streaming...
0x321a	SSVE_SPEC	Operation speculatively executed, Streaming SVE	The counter counts each operation counted by SVE_SPEC specifically in Streaming mode.
0x8004	SIMD_INST_SPEC	Operation speculatively executed, SIMD	Counts speculatively executed operations that are SIMD or SVE vector operations or Advanced SIMD...
0x8005	ASE_INST_SPEC	Operation speculatively executed, Advanced SIMD	The counter counts each Speculatively executed operation due to an A64 Advanced...
0x8006	SVE_INST_SPEC	Operation speculatively executed, SVE, including load and store	The counter counts each Speculatively executed operation due to an SVE instruction. It is...
0x8040	INT_SPEC	Integer operation speculatively executed	Counts speculatively executed integer arithmetic operations.
0x8056	SVE_SPEC	Operation speculatively executed, SVE	The counter counts each operation counted by INST_SPEC that is a scalable vector data processing...
0x8057	ASE_SVE_SPEC	Operation speculatively executed, Advanced SIMD or SVE data processing	The counter counts each operation counted by INST_SPEC that is an Advanced SIMD or scalable...
0x8080	SVE_LDST_SPEC	Operation speculatively executed, SVE load, store, or prefetch	The counter counts each Speculatively executed load, store, or prefetch operation counted by...

Code	Mnemonic	Name	Description
0x8081	SVE_LD_SPEC	Operation speculatively executed, SVE load	The counter counts each Speculatively executed operation that reads from memory due to an...
0x8082	SVE_ST_SPEC	Operation speculatively executed, SVE store	The counter counts each Speculatively executed operation that writes to memory due to an...
0x8083	SVE_PRF_SPEC	Operation speculatively executed, SVE prefetch	The counter counts each Speculatively executed prefetch operation due to any of the following...
0x8087	PRF_SPEC	Operation speculatively executed, Prefetch	Counts speculatively executed operations that prefetch memory. For example: Scalar: PRFM, SVE:...
0x8170	CAS_NEAR_FAIL	Atomic memory Operation speculatively executed, Compare and Swap fail	Counts compare and swap instructions that executed locally to the PE and did not update the...
0x8171	CAS_NEAR_PASS	Atomic memory Operation speculatively executed, Compare and Swap pass	Counts compare and swap instructions that executed locally to the PE and updated the location...
0x8172	CAS_NEAR_SPEC	Atomic memory Operation speculatively executed, Compare and Swap near	Counts compare and swap instructions that executed locally to the PE.
0x8173	CAS_FAR_SPEC	Atomic memory Operation speculatively executed, Compare and Swap far	Counts compare and swap instructions that did not execute locally to the PE.
0x8174	CAS_SPEC	Atomic memory Operation speculatively executed, Compare and Swap	Counts the total compare and swap instructions that were executed.
0x8175	LSE_LD_SPEC	Atomic memory Operation speculatively executed, load	Counts the total atomic memory instructions that return a value that were speculatively executed.
0x8176	LSE_ST_SPEC	Atomic memory Operation speculatively executed, store	Counts the total atomic memory instructions that do not return a value that were speculatively...
0x8177	LSE_LDST_SPEC	Atomic memory Operation speculatively executed, load or store	Counts the total atomic memory instructions that were speculatively executed.
0x835D	SE_SPEC	Operation speculatively executed, Advanced SIMD, SVE or SME data processing	The counter counts each operation counted by INST_SPEC that is an Advanced SIMD, scalable...
0x835E	SME_INST_SPEC	Operation speculatively executed, SME	The counter counts each speculatively executed operation counted by SE_INST_SPEC that is...
0x835F	SE_INST_SPEC	Operation speculatively executed, Advanced SIMD, SVE, SME	The counter counts each speculatively executed operation counted by INST_SPEC that is classified...

For a complete list of the events in C1-Premium, see [PMU events cheat sheet for C1-Premium](#) and [PMU events lookup table for C1-Premium](#).

0x0010 BR_MIS_PRED, Branch instruction speculatively executed, mispredicted or not predicted, event

Counts branches which are speculatively executed and mispredicted.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

0x0012 BR_PRED, Predictable branch instruction speculatively executed, event

Counts all speculatively executed branches.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

0x001B INST_SPEC, Operation speculatively executed, event

Counts operations that have been speculatively executed.

Related telemetry artifacts**Metrics**

- [load_store_percentage](#)
- [load_percentage](#)
- [store_percentage](#)
- [integer_dp_percentage](#)
- [simd_percentage](#)
- [scalar_fp_percentage](#)
- [barrier_percentage](#)
- [branch_percentage](#)
- [crypto_percentage](#)
- [sve_percentage](#)
- [sve_predicate_percentage](#)
- [fp16_percentage](#)
- [fp32_percentage](#)
- [fp64_percentage](#)
- [sme_percentage](#)

Metric groups

[FP_Precision_Mix](#)

[Operation_Mix](#)

[SVE_Effectiveness](#)

Functional groups

[Spec_Operation](#)

0x003B OP_SPEC, Micro-operation speculatively executed, event

Counts micro-operations speculatively executed. This is the count of the number of micro-operations dispatched in a cycle.

Related telemetry artifacts

Metrics

- [retiring](#)
- [bad_speculation](#)

Metric groups

[Topdown_L1](#)

Functional groups

[Spec_Operation](#)

0x0068 UNALIGNED_LD_SPEC, Unaligned access, read, event

Counts unaligned memory read operations issued by the CPU. This event counts unaligned accesses (as defined by the actual instruction), even if they are subsequently issued as multiple aligned accesses. The event does not count preload operations (PLD, PLI).

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

0x0069 UNALIGNED_ST_SPEC, Unaligned access, write, event

Counts unaligned memory write operations issued by the CPU. This event counts unaligned accesses (as defined by the actual instruction), even if they are subsequently issued as multiple aligned accesses.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

0x006A UNALIGNED_LDST_SPEC, Unaligned access, event

Counts unaligned memory operations issued by the CPU. This event counts unaligned accesses (as defined by the actual instruction), even if they are subsequently issued as multiple aligned accesses.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

0x006C LDREX_SPEC, Exclusive operation speculatively executed, Load-Exclusive, event

Counts Load-Exclusive operations that have been speculatively executed. For example: LDREX, LDX

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

0x006D STREX_PASS_SPEC, Exclusive operation speculatively executed, Store-Exclusive pass, event

Counts store-exclusive operations that have been speculatively executed and have successfully completed the store operation.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

0x006E STREX_FAIL_SPEC, Exclusive operation speculatively executed, Store-Exclusive fail, event

Counts store-exclusive operations that have been speculatively executed and have not successfully completed the store operation.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

0x006F STREX_SPEC, Exclusive operation speculatively executed, Store-Exclusive, event

Counts store-exclusive operations that have been speculatively executed.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

0x0070 LD_SPEC, Operation speculatively executed, load, event

Counts speculatively executed load operations including Single Instruction Multiple Data (SIMD) load operations.

Related telemetry artifacts**Metrics**

- [load_ls_percentage](#)
- [load_percentage](#)

Metric groups[Operation_Mix](#)**Functional groups**[Spec_Operation](#)**0x0071 ST_SPEC, Operation speculatively executed, store, event**

Counts speculatively executed store operations including Single Instruction Multiple Data (SIMD) store operations.

Related telemetry artifacts**Metrics**

- [store_ls_percentage](#)
- [store_percentage](#)

Metric groups[Operation_Mix](#)**Functional groups**[Spec_Operation](#)**0x0072 LDST_SPEC, Operation speculatively executed, load or store, event**

Counts load and store operations that have been speculatively executed.

Related telemetry artifacts**Metrics**

- [load_store_percentage](#)
- [load_ls_percentage](#)
- [store_ls_percentage](#)
- [lse_atomics_ratio](#)

Metric groups[Atomics_Effectiveness](#)[Operation_Mix](#)**Functional groups**[Spec_Operation](#)**0x0073 DP_SPEC, Operation speculatively executed, integer data processing, event**

Counts speculatively executed logical or arithmetic instructions such as MOV/MVN operations.

Related telemetry artifacts**Metrics**

- [integer_dp_percentage](#)

Metric groups[Operation_Mix](#)

Functional groups[Spec_Operation](#)**0x0074 ASE_SPEC, Operation speculatively executed, Advanced SIMD data processing, event**

The counter counts each operation counted by INST_SPEC that is an Advanced SIMD data-processing operation. It does not count SVE operations counted by SVE_SPEC, or SME operations counted by SME_SPEC.

Related telemetry artifacts**Metrics**

- [simd_percentage](#)

Metric groups[Operation_Mix](#)**Functional groups**[Spec_Operation](#)**0x0075 VFP_SPEC, Operation speculatively executed, scalar floating-point, event**

Counts speculatively executed floating point operations. This event does not count operations that move data to or from floating point (vector) registers.

Related telemetry artifacts**Metrics**

- [scalar_fp_percentage](#)

Metric groups[Operation_Mix](#)**Functional groups**[Spec_Operation](#)**0x0076 PC_WRITE_SPEC, Operation speculatively executed, Software change of the PC, event**

Counts speculatively executed operations which cause software changes of the PC. Those operations include all taken branch operations.

Related telemetry artifacts**Metrics**

- [branch_percentage](#)

Metric groups[Operation_Mix](#)**Functional groups**[Spec_Operation](#)

0x0077 CRYPTO_SPEC, Operation speculatively executed, Cryptographic instruction, event

Counts speculatively executed cryptographic operations except for PMULL and VMULL operations.

Related telemetry artifacts**Metrics**

- [crypto_percentage](#)

Metric groups

[Operation_Mix](#)

Functional groups

[Spec_Operation](#)

0x0078 BR_IMMED_SPEC, Branch speculatively executed, immediate branch, event

Counts direct branch operations which are speculatively executed.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

0x0079 BR_RETURN_SPEC, Branch speculatively executed, procedure return, event

Counts procedure return operations (RET, RETAA and RETAB) which are speculatively executed.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

0x007A BR_INDIRECT_SPEC, Branch speculatively executed, indirect branch, event

Counts indirect branch operations including procedure returns, which are speculatively executed. This includes operations that force a software change of the PC, other than exception-generating operations and direct branch instructions. Some examples of the instructions counted by this event include BR Xn, RET, etc...

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

0x007C ISB_SPEC, Barrier speculatively executed, ISB, event

Counts ISB operations that are executed.

Related telemetry artifacts**Metrics**

- [barrier_percentage](#)

Metric groups

[Operation_Mix](#)

Functional groups

[Spec_Operation](#)

0x007D DSB_SPEC, Barrier speculatively executed, DSB, event

Counts DSB operations that are speculatively issued to Load/Store unit in the CPU.

Related telemetry artifacts**Metrics**

- [barrier_percentage](#)

Metric groups

[Operation_Mix](#)

Functional groups

[Spec_Operation](#)

0x007E DMB_SPEC, Barrier speculatively executed, DMB, event

Counts DMB operations that are speculatively issued to the Load/Store unit in the CPU. This event does not count implied barriers from load acquire/store release operations.

Related telemetry artifacts**Metrics**

- [barrier_percentage](#)

Metric groups

[Operation_Mix](#)

Functional groups

[Spec_Operation](#)

0x007F CSDB_SPEC, Barrier speculatively executed, CSDB, event

Counts CSDB operations that are speculatively issued to the Load/Store unit in the CPU. This event does not count implied barriers from load acquire/store release operations.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

0x0090 RC_LD_SPEC, Release consistency operation speculatively executed, Load-Acquire, event

Counts any load acquire operations that are speculatively executed. For example: LDAR, LDARH, LDARB

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

0x0091 RC_ST_SPEC, Release consistency operation speculatively executed, Store-Release, event

Counts any store release operations that are speculatively executed. For example: STLR, STLRH, STLRB

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

0x3218 OP_CME_ISSUE, SME operation issued, event

The counter counts each operation that was issued to a streaming mode compute unit.

The definition of which operations are issued to an SME2 unit is **IMPLEMENTATION DEFINED**. The maximum value by which the counter could increment by in a single cycle is **IMPLEMENTATION DEFINED**.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

0x3219 SSVE_INST_SPEC, Operation speculatively executed, Streaming SVE, including load and store, event

The counter counts each instruction counted by SVE_INST_SPEC when the CPU executes in Streaming mode.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

0x321a SSVE_SPEC, Operation speculatively executed, Streaming SVE, event

The counter counts each operation counted by SVE_SPEC specifically in Streaming mode.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

0x8004 SIMD_INST_SPEC, Operation speculatively executed, SIMD, event

Counts speculatively executed operations that are SIMD or SVE vector operations or Advanced SIMD non-scalar operations.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

0x8005 ASE_INST_SPEC, Operation speculatively executed, Advanced SIMD, event

The counter counts each Speculatively executed operation due to an A64 Advanced SIMD instruction.

It is **IMPLEMENTATION DEFINED** whether the counter counts operations due to Advanced SIMD scalar instructions.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

0x8006 SVE_INST_SPEC, Operation speculatively executed, SVE, including load and store, event

The counter counts each Speculatively executed operation due to an SVE instruction.

It is **IMPLEMENTATION DEFINED** whether the counter counts operations due to non-SIMD SVE instructions.

Instructions classified as SME instructions and counted by SME_INST_SPEC are not counted by this event.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

SVE

0x8040 INT_SPEC, Integer operation speculatively executed, event

Counts speculatively executed integer arithmetic operations.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

0x8056 SVE_SPEC, Operation speculatively executed, SVE, event

The counter counts each operation counted by INST_SPEC that is a scalable vector data processing operation.

It does not count:

- SME operations counted by SME_SPEC.
- Neon operation counted by ASE_SPEC
- Load/store operations

Related telemetry artifacts

Metrics

- [sve_percentage](#)

Metric groups

[Operation_Mix](#)

Functional groups

[Spec_Operation](#)

0x8057 ASE_SVE_SPEC, Operation speculatively executed, Advanced SIMD or SVE data processing, event

The counter counts each operation counted by INST_SPEC that is an Advanced SIMD or scalable vector data processing operation.

It does not count:

- SME operations counted by SME_SPEC.
- Load/store operations

See ASE_SPEC and SVE_SPEC for these classifications.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[Spec_Operation](#)**0x8080 SVE_LDST_SPEC, Operation speculatively executed, SVE load, store, or prefetch, event**

The counter counts each Speculatively executed load, store, or prefetch operation counted by any of SVE_LD_SPEC, SVE_PRF_SPEC, or SVE_ST_SPEC.

This includes Load/Store operations to Z register but does not count Load/Store operations to ZT and ZA registers.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[Spec_Operation](#)**0x8081 SVE_LD_SPEC, Operation speculatively executed, SVE load, event**

The counter counts each Speculatively executed operation that reads from memory due to an SVE load instruction.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[Spec_Operation](#)**0x8082 SVE_ST_SPEC, Operation speculatively executed, SVE store, event**

The counter counts each Speculatively executed operation that writes to memory due to an SVE store instruction.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[Spec_Operation](#)**0x8083 SVE_PRF_SPEC, Operation speculatively executed, SVE prefetch, event**

The counter counts each Speculatively executed prefetch operation due to any of the following A64 instructions:

- SVE: PRFB, PRFD, PRFH, or PRFW.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[Spec_Operation](#)**0x8087 PRF_SPEC, Operation speculatively executed, Prefetch, event**

Counts speculatively executed operations that prefetch memory. For example: Scalar: PRFM, SVE: PRFB, PRFD, PRFH, or PRFW.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[Spec_Operation](#)**0x8170 CAS_NEAR_FAIL, Atomic memory Operation speculatively executed, Compare and Swap fail, event**

Counts compare and swap instructions that executed locally to the PE and did not update the location accessed.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[Spec_Operation](#)**0x8171 CAS_NEAR_PASS, Atomic memory Operation speculatively executed, Compare and Swap pass, event**

Counts compare and swap instructions that executed locally to the PE and updated the location accessed.

Related telemetry artifacts**Metrics**

- [cas_near_pass_ratio](#)
- [cas_near_fail_ratio](#)

Metric groups[Atomics_Effectiveness](#)**Functional groups**[Spec_Operation](#)**0x8172 CAS_NEAR_SPEC, Atomic memory Operation speculatively executed, Compare and Swap near, event**

Counts compare and swap instructions that executed locally to the PE.

Related telemetry artifacts**Metrics**

- [cas_near_ratio](#)
- [cas_far_ratio](#)
- [cas_near_pass_ratio](#)
- [cas_near_fail_ratio](#)

Metric groups[Atomics_Effectiveness](#)**Functional groups**[Spec_Operation](#)**0x8173 CAS_FAR_SPEC, Atomic memory Operation speculatively executed, Compare and Swap far, event**

Counts compare and swap instructions that did not execute locally to the PE.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[Spec_Operation](#)**0x8174 caS_SPEC, Atomic memory Operation speculatively executed, Compare and Swap, event**

Counts the total compare and swap instructions that were executed.

Related telemetry artifacts**Metrics**

- [cas_near_ratio](#)
- [cas_far_ratio](#)

Metric groups[Atomics_Effectiveness](#)**Functional groups**[Spec_Operation](#)**0x8175 LSE_LD_SPEC, Atomic memory Operation speculatively executed, load, event**

Counts the total atomic memory instructions that return a value that were speculatively executed.

Related telemetry artifacts**Metrics**

- [lse_load_ratio](#)

Metric groups[Atomics_Effectiveness](#)**Functional groups**[Spec_Operation](#)**0x8176 LSE_ST_SPEC, Atomic memory Operation speculatively executed, store, event**

Counts the total atomic memory instructions that do not return a value that were speculatively executed.

Related telemetry artifacts**Metrics**

- [lse_store_ratio](#)

Metric groups[Atomics_Effectiveness](#)**Functional groups**[Spec_Operation](#)**0x8177 LSE_LDST_SPEC, Atomic memory Operation speculatively executed, load or store, event**

Counts the total atomic memory instructions that were speculatively executed.

Related telemetry artifacts**Metrics**

- [lse_atomics_ratio](#)
- [lse_load_ratio](#)
- [lse_store_ratio](#)

Metric groups[Atomics_Effectiveness](#)**Functional groups**[Spec_Operation](#)**0x835D SE_SPEC, Operation speculatively executed, Advanced SIMD, SVE or SME data processing, event**

The counter counts each operation counted by INST_SPEC that is an Advanced SIMD, scalable vector extension, or scalable matrix extension data-processing operation.

See ASE_SVE_SPEC and SME_SPEC for these classifications.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[Spec_Operation](#)

0x835E SME_INST_SPEC, Operation speculatively executed, SME, event

The counter counts each speculatively executed operation counted by SE_INST_SPEC that is classified as an SME operation.

Operations due to the following instructions are counted as SME operations:

- Data-processing operations involving the ZA and ZT registers.
- Load and store operations involving the ZA and ZT registers.

Operations due to instructions added by FEAT_SME which involve the SVE registers but do not involve any ZA or ZT registers are counted as SVE data-processing operations.

Related telemetry artifacts**Metrics**

- [sme_percentage](#)

Metric groups

[Operation_Mix](#)

Functional groups

[Spec_Operation](#)

0x835F SE_INST_SPEC, Operation speculatively executed, Advanced SIMD, SVE, SME, event

The counter counts each speculatively executed operation counted by INST_SPEC that is classified as an Advanced SIMD, scalable vector extension, or scalable matrix extension operation.

See ASE_SVE_INST_SPEC and SME_INST_SPEC for these classifications

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

6.13 FP_Operation (FP OPERATION) events for C1-Premium

Speculatively executed floating-point events.

Summary of events in FP_Operation:

- Total implemented Common events: 5
- Total Implemented Product ImpDef events: 0
- PMU Only events : 0
- ETE Only events : 0

Table 6-13: FP_Operation events summary

Code	Mnemonic	Name	Description
0x8014	FP_HP_SPEC	Floating-point operation speculatively executed, half precision	Counts speculatively executed half precision floating point operations.
0x8018	FP_SP_SPEC	Floating-point operation speculatively executed, single precision	Counts speculatively executed single precision floating point operations.
0x801C	FP_DP_SPEC	Floating-point operation speculatively executed, double precision	Counts speculatively executed double precision floating point operations.
0x80C0	FP_SCALE_OPS_SPEC	Scalable floating-point element ALU operations speculatively executed	Counts speculatively executed scalable single precision floating point operations.
0x80C1	FP_FIXED_OPS_SPEC	Non-scalable floating-point element ALU operations speculatively executed	Counts speculatively executed non-scalable single precision floating point operations.

For a complete list of the events in C1-Premium, see [PMU events cheat sheet for C1-Premium](#) and [PMU events lookup table for C1-Premium](#).

0x8014 FP_HP_SPEC, Floating-point operation speculatively executed, half precision, event

Counts speculatively executed half precision floating point operations.

Related telemetry artifacts

Metrics

- [fp16_percentage](#)

Metric groups

[FP_Precision_Mix](#)

Functional groups

[FP_Operation](#)

0x8018 FP_SP_SPEC, Floating-point operation speculatively executed, single precision, event

Counts speculatively executed single precision floating point operations.

Related telemetry artifacts

Metrics

- [fp32_percentage](#)

Metric groups

[FP_Precision_Mix](#)

Functional groups

[FP_Operation](#)

0x801C FP_DP_SPEC, Floating-point operation speculatively executed, double precision, event

Counts speculatively executed double precision floating point operations.

Related telemetry artifacts**Metrics**

- [fp64_percentage](#)

Metric groups

[FP_Precision_Mix](#)

Functional groups

[FP_Operation](#)

0x80c0 FP_SCALE_OPS_SPEC, Scalable floating-point element ALU operations speculatively executed, event

Counts speculatively executed scalable single precision floating point operations.

Related telemetry artifacts**Metrics**

- [sve_fp_ops_per_cycle](#)
- [fp_ops_per_cycle](#)

Metric groups

[FP_Arithmetic_Intensity](#)

Functional groups

[FP_Operation](#)

0x80c1 FP_FIXED_OPS_SPEC, Non-scalable floating-point element ALU operations speculatively executed, event

Counts speculatively executed non-scalable single precision floating point operations.

Related telemetry artifacts**Metrics**

- [nonsve_fp_ops_per_cycle](#)
- [fp_ops_per_cycle](#)

Metric groups

[FP_Arithmetic_Intensity](#)

Functional groups

[FP_Operation](#)

6.14 Stall (STALL) events for C1-Premium

Stall related events.

Summary of events in Stall:

- Total implemented Common events: 21

- Total Implemented Product ImpDef events: 23
- PMU Only events : 23
- ETE Only events : 0

Table 6-14: Stall events summary

Code	Mnemonic	Name	Description
0x0023	STALL_FRONTEND	No operation sent for execution due to the frontend	Counts cycles when frontend could not send any micro-operations to the rename stage because of...
0x0024	STALL_BACKEND	No operation sent for execution due to the backend	Counts cycles whenever the rename unit is unable to send any micro-operations to the backend of...
0x003C	STALL	No operation sent for execution	Counts cycles when no operations are sent to the rename unit from the frontend or from the rename...
0x003D	STALL_SLOT_BACKEND	No operation sent for execution on a Slot due to the backend	Counts slots per cycle in which no operations are sent from the rename unit to the backend due to...
0x003E	STALL_SLOT_FRONTEND	No operation sent for execution on a Slot due to the frontend	Counts slots per cycle in which no operations are sent to the rename unit from the frontend due...
0x003F	STALL_SLOT	No operation sent for execution on a Slot	Counts slots per cycle in which no operations are sent to the rename unit from the frontend or...
0x0158	IMP_STALL_BACKEND_RENAME_FRF	Backend rename stall due to no available flag registers	Counts the number of backend rename stall cycles due to the flag registers being full (FRF), that...
0x0159	IMP_STALL_BACKEND_RENAME_GRF	Backend rename stall due to no available general purpose registers	Counts the number of backend rename stall cycles due to the general purpose registers being full...
0x015A	IMP_STALL_BACKEND_RENAME_VRF	Backend rename stall due to no available vector registers	Counts the number of backend rename stall cycles due to the vector registers being full (VRF),...
0x015C	IMP_STALL_BACKEND_IQ_SX	Backend dispatch stall due to no room for operations in simple integer issue queues	Counts the number of dispatch stall cycles due to simple integer issue queues (SX IQ) which...
0x015D	IMP_STALL_BACKEND_IQ_MX	Backend dispatch stall due to no room for operations in complex integer issue queues	Counts the number of dispatch stall cycles due to complex integer issue queues (MX IQ) which...
0x015E	IMP_STALL_BACKEND_IQ_LS	Backend dispatch stall due to no room for operations in load store issue queues	Counts the number of dispatch stall cycles due to load store issue queues (LS IQ) which cannot...
0x015F	IMP_STALL_BACKEND_IQ_VX	Backend dispatch stall due to no room for operations in vector issue queues	Counts the number of dispatch stall cycles due to vector issue queues (VX IQ) which cannot take...
0x0160	IMP_STALL_BACKEND_MCQ	Backend dispatch stage stall, due to full commit queue	Counts cycles where the dispatch stage is stalled because the commit queue (MCQ) is full and...

Code	Mnemonic	Name	Description
0x0399	IMP_STALL_BACKEND_RENAME_PDRF	Backend rename stall due to no available predicate registers	Counts the number of backend rename stall cycles due to the predicate registers being full...
0x1338	IMP_STALL_BACKEND_CPUBOUND_OTHER	Stall backend cpubound other than renames or issue queue	Counts cycles when backend is stalled even when operations are available from the frontend but at...
0x3005	IMP_STALL_FRONTEND_SPEC_THROT	Stall frontend cycles due to power throttling linked to low confidence branches	Counts cycles when the frontend did not send any micro-operations to the rename stage as the...
0x3006	IMP_STALL_FRONTEND_FLUSH_CLEAR	Stall frontend flush cycles due to architectural or microarchitectural flushes	Counts cycles when the frontend could not send any micro-operations to the rename stage as the...
0x3007	IMP_STALL_FRONTEND_FLUSH_RESTEER	Stall frontend flush cycles due to flush for branch mispredictions	Counts cycles when the frontend could not send any micro-operations to the rename stage as the...
0x3009	IMP_STALL_BACKEND_SPEC_THROT	Stall backend cycles due to power throttling linked to low confidence branches	Counts cycles when the backend did not accept any micro-operations as the backend is actively...
0x3200	STALL_BACKEND_BUSY_CME	Backend stall cycles, SME2 unit busy	The counter counts each PE cycle counted by STALL_BACKEND_CPUBOUND when the PEs backend was...
0x3201	STALL_BACKEND_BUSY_CMEBOUND	Backend stall cycles, SME2 unit backpressure	The counter counts each CPU cycle counted by STALL_BACKEND_BUSY_CME when the SME2 unit causes...
0x3202	STALL_BACKEND_BUSY_CME_ARB	Backend stall cycles, SME2 unit arbitration	The counter counts each CPU cycle counted by STALL_BACKEND_BUSY_CME when oldest SME instruction...
0x3203	STALL_BACKEND_BUSY_CME_CPUBOUND	Backend stall cycles, SME2 unit stalled by CPU	The counter counts each PE cycle counted by STALL_BACKEND_BUSY_CME when not counted by...
0x320c	STALL_BACKEND_MEM_CME_LSRT_FULL	Backend stall cycles, SME2 unit stalled on LSRT full	The counter counts each CPU cycle counted by STALL_BACKEND_MEMBOUND when at least one Streaming...
0x320d	STALL_BACKEND_MEM_CME_BARRIER	Backend stall cycles, barrier stalled by SME2 unit	The counter counts each CPU cycle counted by STALL_BACKEND_MEMBOUND when a barrier instruction is...
0x320e	STALL_BACKEND_MEM_CME_HZ_ON_CPU	Backend stall cycles, SME2 unit stalled by CPU memory hazard	The counter counts each CPU cycle counted by STALL_BACKEND_MEMBOUND when at least one Streaming...
0x320f	STALL_BACKEND_MEM_CPU_HZ_ON_CME	Backend stall cycles, CPU stalled by SME2 unit memory hazard	The counter counts each CPU cycle counted by STALL_BACKEND_MEMBOUND when at least one CPU...
0x3210	STALL_BACKEND_MEM_CME	Backend stall cycles, SME2 unit stalled on memory hazard or LSRT full	The counter counts each CPU cycle counted by STALL_BACKEND_MEMBOUND when at least one Streaming...
0x4005	STALL_BACKEND_MEM	Memory stall cycles	Counts cycles when the backend is stalled because there is a demand data miss in the last level...

Code	Mnemonic	Name	Description
0x8158	STALL_FRONTEND_MEMBOUND	Frontend stall cycles, memory bound	Counts cycles when the frontend could not send any micro-operations to the rename stage due to...
0x8159	STALL_FRONTEND_L1I	Frontend stall cycles, level 1 instruction cache	Counts cycles when the frontend is stalled because there is an instruction fetch request pending...
0x815B	STALL_FRONTEND_MEM	Frontend stall cycles, last level PE cache or memory	Counts cycles when the frontend is stalled because there is an instruction fetch request pending...
0x815C	STALL_FRONTEND_TLB	Frontend stall cycles, TLB	Counts when the frontend is stalled on any TLB misses being handled. This event also counts the...
0x8160	STALL_FRONTEND_CPUBOUND	Frontend stall cycles, processor bound	Counts cycles when the frontend could not send any micro-operations to the rename stage due to...
0x8161	STALL_FRONTEND_FLOW	Frontend stall cycles, flow control	Counts cycles when the frontend could not send any micro-operations to the rename stage due to...
0x8162	STALL_FRONTEND_FLUSH	Frontend stall cycles, flush recovery	Counts cycles when the frontend could not send any micro-operations to the rename stage as the...
0x8164	STALL_BACKEND_MEMBOUND	Backend stall cycles, memory bound	Counts cycles when the backend could not accept any micro-operations due to resource constraints...
0x8165	STALL_BACKEND_L1D	Backend stall cycles, level 1 data cache	Counts cycles when the backend is stalled because there is a demand data miss in the level 1 data...
0x8167	STALL_BACKEND_TLB	Backend stall cycles, TLB	Counts cycles when the backend is stalled on any demand TLB misses being handled.
0x8168	STALL_BACKEND_ST	Backend stall cycles, store	Counts cycles when the backend is stalled and there is a store that has not reached the...
0x816A	STALL_BACKEND_CPUBOUND	Backend stall cycles, processor bound	Counts cycles when the backend could not accept any micro-operations due to any resource...
0x816B	STALL_BACKEND_BUSY	Backend stall cycles, backend busy	Counts cycles when the backend could not accept any micro-operations because the issue queues are...
0x816D	STALL_BACKEND_RENAME	Backend stall cycles, rename full	Counts cycles when backend is stalled even when operations are available from the frontend but at...

For a complete list of the events in C1-Premium, see [PMU events cheat sheet for C1-Premium](#) and [PMU events lookup table for C1-Premium](#).

0x0023 STALL_FRONTEND, No operation sent for execution due to the frontend, event

Counts cycles when frontend could not send any micro-operations to the rename stage because of frontend resource stalls caused by fetch memory latency or branch prediction flow stalls. STALL_FRONTEND_SLOTS counts SLOTS during the cycle when this event counts.

Related telemetry artifacts

Metrics

- [frontend_stalled_cycles](#)
- [frontend_core_bound](#)
- [frontend_mem_bound](#)

Metric groups

[Cycle_Accounting](#)
[Topdown_Frontend](#)

Functional groups

[Stall](#)

0x0024 STALL_BACKEND, No operation sent for execution due to the backend, event

Counts cycles whenever the rename unit is unable to send any micro-operations to the backend of the pipeline because of backend resource constraints. Backend resource constraints can include issue stage fullness, execution stage fullness, or other internal pipeline resource fullness. All the backend slots were empty during the cycle when this event counts.

Related telemetry artifacts

Metrics

- [backend_stalled_cycles](#)
- [backend_core_bound](#)
- [backend_mem_bound](#)
- [backend_busy_bound](#)

Metric groups

[Cycle_Accounting](#)
[Topdown_Backend](#)

Functional groups

[Stall](#)

0x003c STALL, No operation sent for execution, event

Counts cycles when no operations are sent to the rename unit from the frontend or from the rename unit to the backend for any reason (either frontend or backend stall). This event is the sum of STALL_FRONTEND and STALL_BACKEND

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Stall](#)

0x003D STALL_SLOT_BACKEND, No operation sent for execution on a Slot due to the backend, event

Counts slots per cycle in which no operations are sent from the rename unit to the backend due to backend resource constraints. STALL_BACKEND counts during the cycle when STALL_SLOT_BACKEND counts at least 1.

Related telemetry artifacts

Metrics

- [backend_bound](#)

Metric groups

[Topdown_L1](#)

Functional groups

[Stall](#)

0x003E STALL_SLOT_FRONTEND, No operation sent for execution on a Slot due to the frontend, event

Counts slots per cycle in which no operations are sent to the rename unit from the frontend due to frontend resource constraints.

Related telemetry artifacts

Metrics

- [frontend_bound](#)

Metric groups

[Topdown_L1](#)

Functional groups

[Stall](#)

0x003F STALL_SLOT, No operation sent for execution on a Slot, event

Counts slots per cycle in which no operations are sent to the rename unit from the frontend or from the rename unit to the backend for any reason (either frontend or backend stall). STALL_SLOT is the sum of STALL_SLOT_FRONTEND and STALL_SLOT_BACKEND.

Related telemetry artifacts

Metrics

- [retiring](#)
- [bad_speculation](#)

Metric groups

[Topdown_L1](#)

Functional groups

[Stall](#)

0x0158 IMP_STALL_BACKEND_RENAME_FRF, Backend rename stall due to no available flag registers, event

Counts the number of backend rename stall cycles due to the flag registers being full (FRF), that is, no flag registers are available. This event counts when an operation is available to be sent to the backend but cannot be sent because all physical flag registers are in use. Flag registers store condition codes such as zero, carry, overflow, and negative flags. These registers are used for conditional execution, comparisons, and arithmetic operations.

Related telemetry artifacts

Metrics

- [rename_stall_vec_ratio](#)
- [rename_stall_flags_ratio](#)
- [rename_stall_int_ratio](#)
- [rename_stall_pred_ratio](#)

Metric groups

[Rename_Effectiveness](#)

Functional groups

[Stall](#)

0x0159 IMP_STALL_BACKEND_RENAME_GRF, Backend rename stall due to no available general purpose registers, event

Counts the number of backend rename stall cycles due to the general purpose registers being full (GRF), that is, no general purpose registers are available. This event counts when an operation is available to be sent to the backend but cannot be sent because all physical general purpose registers are in use. These registers store temporary data operands and computational results. Their availability is crucial to sustain instruction throughput.

Related telemetry artifacts

Metrics

- [rename_stall_vec_ratio](#)
- [rename_stall_flags_ratio](#)
- [rename_stall_int_ratio](#)
- [rename_stall_pred_ratio](#)

Metric groups

[Rename_Effectiveness](#)

Functional groups

[Stall](#)

0x015A IMP_STALL_BACKEND_RENAME_VRF, Backend rename stall due to no available vector registers, event

Counts the number of backend rename stall cycles due to the vector registers being full (VRF), that is, no vector registers are available. This event counts when an operation is available to be sent to

the backend but cannot be sent because all physical vector registers are in use. Vector registers store Single Instruction, Multiple DATA (SIMD) and Scalable Vector Extension (SVE) operations. Their availability is critical to accelerate parallel computations.

Related telemetry artifacts

Metrics

- [rename_stall_vec_ratio](#)
- [rename_stall_flags_ratio](#)
- [rename_stall_int_ratio](#)
- [rename_stall_pred_ratio](#)

Metric groups

[Rename_Effectiveness](#)

Functional groups

[Stall](#)

0x015C IMP_STALL_BACKEND_IQ_SX, Backend dispatch stall due to no room for operations in simple integer issue queues, event

Counts the number of dispatch stall cycles due to simple integer issue queues (SX IQ) which cannot take any operations for execution. This event counts when the oldest operation is waiting to issue to the SX IQ but it is full.

Related telemetry artifacts

Metrics

- [iq_stall_sx_percentage](#)

Metric groups

[IQ_Effectiveness](#)

Functional groups

[Stall](#)

0x015D IMP_STALL_BACKEND_IQ_MX, Backend dispatch stall due to no room for operations in complex integer issue queues, event

Counts the number of dispatch stall cycles due to complex integer issue queues (MX IQ) which cannot take any operations for execution. This event counts when the oldest operation is waiting to issue to the MX IQ but it is full.

Related telemetry artifacts

Metrics

- [iq_stall_mx_percentage](#)

Metric groups

[IQ_Effectiveness](#)

Functional groups

[Stall](#)

0x015E IMP_STALL_BACKEND_IQ_LS, Backend dispatch stall due to no room for operations in load store issue queues, event

Counts the number of dispatch stall cycles due to load store issue queues (LS IQ) which cannot take any operations for execution. This event counts when the oldest operation is waiting to issue to the LS IQ but it is full.

Related telemetry artifacts**Metrics**

- [iq_stall_lsu_percentage](#)

Metric groups

[IQ_Effectiveness](#)

Functional groups

[Stall](#)

0x015F IMP_STALL_BACKEND_IQ_VX, Backend dispatch stall due to no room for operations in vector issue queues, event

Counts the number of dispatch stall cycles due to vector issue queues (VX IQ) which cannot take any operations for execution. This event counts when the oldest operation is waiting to issue to the VX IQ but it is full.

Related telemetry artifacts**Metrics**

- [iq_stall_vpu_percentage](#)

Metric groups

[IQ_Effectiveness](#)

Functional groups

[Stall](#)

0x0160 IMP_STALL_BACKEND_MCQ, Backend dispatch stage stall, due to full commit queue, event

Counts cycles where the dispatch stage is stalled because the commit queue (MCQ) is full and cannot take any operations for execution. The commit queue is responsible for managing the final stage of instruction execution, where instructions are retired in program order after completing execution.

Related telemetry artifacts**Metrics**

- [mcq_stall_percentage](#)

Metric groups

[MCQ_Effectiveness](#)

Functional groups

[Stall](#)

0x0399 IMP_STALL_BACKEND_RENAME_PDRF, Backend rename stall due to no available predicate registers, event

Counts the number of backend rename stall cycles due to the predicate registers being full (PDRF), that is, no predicate registers are available.

Related telemetry artifacts**Metrics**

- [rename_stall_vec_ratio](#)
- [rename_stall_flags_ratio](#)
- [rename_stall_int_ratio](#)
- [rename_stall_pred_ratio](#)

Metric groups

[Rename_Effectiveness](#)

Functional groups

[Stall](#)

0x1338 IMP_STALL_BACKEND_CPUBOUND_OTHER, Stall backend cpubound other than renames or issue queue, event

Counts cycles when backend is stalled even when operations are available from the frontend but at least one is not ready to be sent to the backend because a cpu resource other than renames or issue queues.

Related telemetry artifacts**Metrics**

- [backend_core_other_bound](#)

Metric groups

[Topdown_Backend](#)

Functional groups

[Stall](#)

0x3005 IMP_STALL_FRONTEND_SPEC_THROT, Stall frontend cycles due to power throttling linked to low confidence branches, event

Counts cycles when the frontend did not send any micro-operations to the rename stage as the frontend is actively throttle throughput based on speculation around low confidence branches

Related telemetry artifacts**Metrics**

- [frontend_core_spec_throttle_bound](#)

Metric groups

[Topdown_Frontend](#)

Functional groups

[Stall](#)

0x3006 IMP_STALL_FRONTEND_FLUSH_CLEAR, Stall frontend flush cycles due to architectural or microarchitectural flushes, event

Counts cycles when the frontend could not send any micro-operations to the rename stage as the frontend is recovering from a machine flush due to taken exceptions or other micro-architectural flushes not related to branch mispredictions.

Related telemetry artifacts**Metrics**

- [frontend_core_flush_machine_clear_bound](#)

Metric groups

[Topdown_Frontend](#)

Functional groups

[Stall](#)

0x3007 IMP_STALL_FRONTEND_FLUSH_RESTEER, Stall frontend flush cycles due to flush for branch mispredictions, event

Counts cycles when the frontend could not send any micro-operations to the rename stage as the frontend is recovering from a resteer due to branch mispredictions.

Related telemetry artifacts**Metrics**

- [frontend_core_flush_resteer_bound](#)

Metric groups

[Topdown_Frontend](#)

Functional groups

[Stall](#)

0x3009 IMP_STALL_BACKEND_SPEC_THROT, Stall backend cycles due to power throttling linked to low confidence branches, event

Counts cycles when the backend did not accept any micro-operations as the backend is actively throttle throughput based on speculation around low confidence branches

Related telemetry artifacts**Metrics**

- [backend_core_spec_throttle_bound](#)

Metric groups

[Topdown_Backend](#)

Functional groups

[Stall](#)

0x3200 STALL_BACKEND_BUSY_CME, Backend stall cycles, SME2 unit busy, event

The counter counts each PE cycle counted by STALL_BACKEND_CPUBOUND when the PEs backend was stalled due SME instructions not able to progress, typically:

- When waiting for SME2 unit arbitration
- Because of SME2 unit backpressure
- Because instructions cannot be sent to the SME2 unit due to dependencies, commit, etc.

Related telemetry artifacts

Metrics

- [backend_core_cme_bound](#)
- [backend_cme_cpu_bound](#)
- [backend_cme_backpressure_bound](#)
- [backend_cme_busy_arb_bound](#)

Metric groups

[Topdown_Backend](#)

[Topdown_CME](#)

Functional groups

[Stall](#)

0x3201 STALL_BACKEND_BUSY_CMEBOUND, Backend stall cycles, SME2 unit backpressure, event

The counter counts each CPU cycle counted by STALL_BACKEND_BUSY_CME when the SME2 unit causes backpressure and does not accept instructions.

Notes:

- This event counts independently of oldest instruction being ready to be sent to the SME2 unit
- This event does not count when the CPU is waiting for arbitration, and STALL_BACKEND_BUSY_CME_ARB is counting.

Related telemetry artifacts

Metrics

- [backend_cme_backpressure_bound](#)

Metric groups

[Topdown_CME](#)

Functional groups

[Stall](#)

0x3202 STALL_BACKEND_BUSY_CME_ARB, Backend stall cycles, SME2 unit arbitration, event

The counter counts each CPU cycle counted by STALL_BACKEND_BUSY_CME when oldest SME instruction cannot be sent because it is waiting for arbitration.

Related telemetry artifacts**Metrics**

- [backend_cme_busy_arb_bound](#)

Metric groups[Topdown_CME](#)**Functional groups**[Stall](#)**0x3203 STALL_BACKEND_BUSY_CME_CPUBOUND, Backend stall cycles, SME2 unit stalled by CPU, event**

The counter counts each PE cycle counted by STALL_BACKEND_BUSY_CME when not counted by STALL_BACKEND_BUSY_CME_BOUND or STALL_BACKEND_BUSY_CME_ARB.

This can typically be due to:

- Instruction waiting for commit point being reached
- A register dependency prevents the SSVE oldest instruction to be sent
- A control packet needs to be sent

Related telemetry artifacts**Metrics**

- [backend_cme_cpu_bound](#)

Metric groups[Topdown_CME](#)**Functional groups**[Stall](#)**0x320c STALL_BACKEND_MEM_CME_LSRT_FULL, Backend stall cycles, SME2 unit stalled on LSRT full, event**

The counter counts each CPU cycle counted by STALL_BACKEND_MEMBOUND when at least one Streaming SVE instruction is waiting for an entry being freed to be allocated into the LSRT.

Related telemetry artifacts**Metrics**

- [backend_mem_cme_lsrt_full_bound](#)

Metric groups[Topdown_Backend](#)**Functional groups**[Stall](#)

0x320d STALL_BACKEND_MEM_CME_BARRIER, Backend stall cycles, barrier stalled by SME2 unit, event

The counter counts each CPU cycle counted by STALL_BACKEND_MEMBOUND when a barrier instruction is executed and waits for completions from SME load/store transactions.

This includes effect due to store-release or load-acquire semantic.

Related telemetry artifacts**Metrics**

- [backend_mem_cme_barrier_bound](#)

Metric groups

[Topdown_Backend](#)

Functional groups

[Stall](#)

0x320e STALL_BACKEND_MEM_CME_HZ_ON_CPU, Backend stall cycles, SME2 unit stalled by CPU memory hazard, event

The counter counts each CPU cycle counted by STALL_BACKEND_MEMBOUND when at least one Streaming SVE load/store instruction is waiting for resolution from an address hazard. This could be used to detect cases for which the CPU and SME2 unit are making overlapping accesses, that is, both are accessing the same location, and the SME2 unit cannot accept the operation to preserve the ordering of memory effects for the location required by the architecture.

Related telemetry artifacts**Metrics**

- [backend_mem_cme_hazard_cpu_bound](#)

Metric groups

[Topdown_Backend](#)

Functional groups

[Stall](#)

0x320f STALL_BACKEND_MEM_CPU_HZ_ON_CME, Backend stall cycles, CPU stalled by SME2 unit memory hazard, event

The counter counts each CPU cycle counted by STALL_BACKEND_MEMBOUND when at least one CPU load/store instruction is waiting for resolution from an address hazard. This could be used to detect cases for which the CPU and SME2 unit are making overlapping accesses, that is, both are accessing the same location, and the CPU cannot execute the operation to preserve the ordering of memory effects for the location required by the architecture.

Related telemetry artifacts**Metrics**

- [backend_mem_cpu_hazard_cme_bound](#)

Metric groups

[Topdown_Backend](#)

Functional groups

[Stall](#)

0x3210 STALL_BACKEND_MEM_CME, Backend stall cycles, SME2 unit stalled on memory hazard or LSRT full, event

The counter counts each CPU cycle counted by STALL_BACKEND_MEMBOUND when at least one Streaming SVE instruction is waiting for an entry being freed to be allocated into the LSRT or an address hazard to be resolved, or at least one CPU load/store instruction is waiting for resolution from an address hazard caused by Streaming SVE instruction.

Related telemetry artifacts

Metrics

- [backend_mem_cme_bound](#)
- [backend_mem_cme_hazard_cpu_bound](#)
- [backend_mem_cpu_hazard_cme_bound](#)
- [backend_mem_cme_lsrt_full_bound](#)
- [backend_mem_cme_barrier_bound](#)

Metric groups

[Topdown_Backend](#)

Functional groups

[Stall](#)

0x4005 STALL_BACKEND_MEM, Memory stall cycles, event

Counts cycles when the backend is stalled because there is a demand data miss in the last level core cache.

Related telemetry artifacts

Metrics

- [backend_mem_cache_bound](#)
- [backend_cache_l1d_bound](#)
- [backend_cache_l2d_bound](#)

Metric groups

[Topdown_Backend](#)

Functional groups

[Stall](#)

0x8158 STALL_FRONTEND_MEMBOUND, Frontend stall cycles, memory bound, event

Counts cycles when the frontend could not send any micro-operations to the rename stage due to resource constraints in the memory resources.

Related telemetry artifacts

Metrics

- [frontend_mem_bound](#)
- [frontend_mem_cache_bound](#)
- [frontend_mem_tlb_bound](#)

Metric groups

[Topdown_Frontend](#)

Functional groups

[Stall](#)

0x8159 STALL_FRONTEND_L1I, Frontend stall cycles, level 1 instruction cache, event

Counts cycles when the frontend is stalled because there is an instruction fetch request pending in the level 1 instruction cache.

Related telemetry artifacts

Metrics

- [frontend_mem_cache_bound](#)
- [frontend_cache_l1i_bound](#)
- [frontend_cache_l2i_bound](#)

Metric groups

[Topdown_Frontend](#)

Functional groups

[Stall](#)

0x815B STALL_FRONTEND_MEM, Frontend stall cycles, last level PE cache or memory, event

Counts cycles when the frontend is stalled because there is an instruction fetch request pending in the last level core cache.

Related telemetry artifacts

Metrics

- [frontend_mem_cache_bound](#)
- [frontend_cache_l1i_bound](#)
- [frontend_cache_l2i_bound](#)

Metric groups

[Topdown_Frontend](#)

Functional groups

[Stall](#)

0x815c STALL_FRONTEND_TLB, Frontend stall cycles, TLB, event

Counts when the frontend is stalled on any TLB misses being handled. This event also counts the TLB accesses made by hardware prefetches.

Related telemetry artifacts**Metrics**

- [frontend_mem_tlb_bound](#)

Metric groups

[Topdown_Frontend](#)

Functional groups

[Stall](#)

0x8160 STALL_FRONTEND_CPUBOUND, Frontend stall cycles, processor bound, event

Counts cycles when the frontend could not send any micro-operations to the rename stage due to resource constraints in the CPU resources excluding memory resources.

Related telemetry artifacts**Metrics**

- [frontend_core_bound](#)
- [frontend_core_spec_throttle_bound](#)
- [frontend_core_flush_bound](#)
- [frontend_core_flow_bound](#)

Metric groups

[Topdown_Frontend](#)

Functional groups

[Stall](#)

0x8161 STALL_FRONTEND_FLOW, Frontend stall cycles, flow control, event

Counts cycles when the frontend could not send any micro-operations to the rename stage due to resource constraints in the branch prediction unit.

Related telemetry artifacts**Metrics**

- [frontend_core_flow_bound](#)

Metric groups

[Topdown_Frontend](#)

Functional groups

[Stall](#)

0x8162 STALL_FRONTEND_FLUSH, Frontend stall cycles, flush recovery, event

Counts cycles when the frontend could not send any micro-operations to the rename stage as the frontend is recovering from a machine flush or resteer. Example scenarios that cause a flush include branch mispredictions, taken exceptions, micro-architectural flush etc.

Related telemetry artifacts**Metrics**

- [frontend_bound](#)
- [bad_speculation](#)
- [frontend_core_flush_resteer_bound](#)
- [frontend_core_flush_machine_clear_bound](#)
- [frontend_core_flush_bound](#)

Metric groups[Topdown_Frontend](#)[Topdown_L1](#)**Functional groups**[Stall](#)**0x8164 STALL_BACKEND_MEMBOUND, Backend stall cycles, memory bound, event**

Counts cycles when the backend could not accept any micro-operations due to resource constraints in the memory resources.

Related telemetry artifacts**Metrics**

- [backend_mem_bound](#)
- [backend_mem_cache_bound](#)
- [backend_mem_tlb_bound](#)
- [backend_mem_store_bound](#)
- [backend_mem_cme_bound](#)

Metric groups[Topdown_Backend](#)**Functional groups**[Stall](#)**0x8165 STALL_BACKEND_L1D, Backend stall cycles, level 1 data cache, event**

Counts cycles when the backend is stalled because there is a demand data miss in the level 1 data cache.

Related telemetry artifacts**Metrics**

- [backend_mem_cache_bound](#)

- [backend_cache_l1d_bound](#)
- [backend_cache_l2d_bound](#)

Metric groups[Topdown_Backend](#)**Functional groups**[Stall](#)**0x8167 STALL_BACKEND_TLB, Backend stall cycles, TLB, event**

Counts cycles when the backend is stalled on any demand TLB misses being handled.

Related telemetry artifacts**Metrics**

- [backend_mem_tlb_bound](#)

Metric groups[Topdown_Backend](#)**Functional groups**[Stall](#)**0x8168 STALL_BACKEND_ST, Backend stall cycles, store, event**

Counts cycles when the backend is stalled and there is a store that has not reached the pre-commit stage.

Related telemetry artifacts**Metrics**

- [backend_mem_store_bound](#)

Metric groups[Topdown_Backend](#)**Functional groups**[Stall](#)**0x816A STALL_BACKEND_CPUBOUND, Backend stall cycles, processor bound, event**

Counts cycles when the backend could not accept any micro-operations due to any resource constraints in the CPU excluding memory resources.

Related telemetry artifacts**Metrics**

- [backend_core_bound](#)
- [backend_core_spec_throttle_bound](#)
- [backend_core_other_bound](#)
- [backend_core_rename_bound](#)
- [mcq_stall_percentage](#)

- [backend_core_cme_bound](#)

Metric groups[MCQ_Effectiveness](#)[Topdown_Backend](#)**Functional groups**[Stall](#)**0x816B STALL_BACKEND_BUSY, Backend stall cycles, backend busy, event**

Counts cycles when the backend could not accept any micro-operations because the issue queues are full to take any operations for execution.

Related telemetry artifacts**Metrics**

- [backend_busy_bound](#)
- [iq_stall_lsu_percentage](#)
- [iq_stall_sx_percentage](#)
- [iq_stall_mx_percentage](#)
- [iq_stall_vpu_percentage](#)

Metric groups[IQ_Effectiveness](#)[Topdown_Backend](#)**Functional groups**[Stall](#)**0x816D STALL_BACKEND_RENAME, Backend stall cycles, rename full, event**

Counts cycles when backend is stalled even when operations are available from the frontend but at least one is not ready to be sent to the backend because no rename register is available.

Related telemetry artifacts**Metrics**

- [backend_core_rename_bound](#)

Metric groups[Topdown_Backend](#)**Functional groups**[Stall](#)

6.15 General (GENERAL) events for C1-Premium

General CPU related events.

Summary of events in General:

- Total implemented Common events: 3
- Total Implemented Product ImpDef events: 17
- PMU Only events : 17
- ETE Only events : 0

Table 6-15: General events summary

Code	Mnemonic	Name	Description
0x0011	CPU_CYCLES	Cycle	Counts CPU clock cycles (not timer cycles). The clock measured by this event is defined as the...
0x0198	IMP_L2_CHI_RX_CBUSY_0	Received RXDAT or RXRSP responses, with CBusy 0	Counts the number of RXDAT or RXRSP responses received with a CBusy value of 0. Note that if an...
0x0199	IMP_L2_CHI_RX_CBUSY_1	Received RXDAT or RXRSP responses, with CBusy 1	Counts the number of RXDAT or RXRSP responses received with a CBusy value of 1. Note that if an...
0x019A	IMP_L2_CHI_RX_CBUSY_2	Received RXDAT or RXRSP responses, with CBusy 2	Counts the number of RXDAT or RXRSP responses received with a CBusy value of 2. Note that if an...
0x019B	IMP_L2_CHI_RX_CBUSY_3	Received RXDAT or RXRSP responses, with CBusy 3	Counts the number of RXDAT or RXRSP responses received with a CBusy value of 3. Note that if an...
0x019C	IMP_L2_CHI_RX_CBUSY_MT	Received RXDAT or RXRSP responses, with CBusy multi-threaded set	Counts the number of RXDAT or RXRSP responses received with CBusy multi-threaded set. Note that...
0x0225	IMP_WFX_CLOCK_CYCLES	Cycles in WFX awake handling snoop	Counts cycles in WFX wait state awake handling external snoop traffic.
0x3000	IMP_OP_BRU_ISSUE	Branch operation issued	This event counts each resolution from the branch execution pipelines.
0x3001	IMP_OP_DPU_ISSUE	Integer operation issued	This event counts each resolution from the integer execution pipelines.
0x3002	IMP_OP_VPU_ISSUE	Vector/float operation issued	This event counts each resolution from the vector execution pipelines.
0x3003	IMP_OP_LSU_ISSUE	Load/store operation issued	This event counts each resolution from the load store execution pipelines.
0x3004	IMP_OP_STD_ISSUE	Store data operation issued	This event counts each resolution for store data operations.
0x3212	SM_ACTIVE_CYCLES	Cycles PSTATE.SM enabled	The counter counts each cycle counted by CPU_CYCLES when PSTATE.SM was enabled.
0x3213	CYCLES_CME_ALLOC	Cycles SME2 unit allocated	The counter counts each PE cycle counted by CPU_CYCLES where the CPU had an granted arbitration...
0x3214	CYCLES_ARB_PENDING_CME	Cycles SME2 unit arbitration pending	The counter counts each PE cycle counted by CPU_CYCLES where the CPU is in waiting for...
0x3215	CYCLES_CME_RECONNECT_PENDING	Cycles SME2 unit reconnect pending	The counter counts each PE cycle counted by CYCLES_ARB_PENDING_CME where the CPU is in waiting...

Code	Mnemonic	Name	Description
0x3216	ARB_CME_COUNT	SME2 unit arbitration requests	The counter counts the number of times an arbitration to SME2 unit is requested, either an...
0x3217	RECONNECT_CME_COUNT	SME2 unit reconnect requests	The counter counts the number of times the CPU needs to request arbitration following...
0x4004	CNT_CYCLES	Constant frequency cycles	Increments at a constant frequency equal to the rate of increment of the System Counter, CNTPCT_ELO.
0x8380	ZA_ACTIVE	PSTATE.ZA active cycles	The counter counts each cycle counted by CPU_CYCLES when PSTATE.ZA was enabled.

For a complete list of the events in C1-Premium, see [PMU events cheat sheet for C1-Premium](#) and [PMU events lookup table for C1-Premium](#).

0x0011 CPU_CYCLES, Cycle, event

Counts CPU clock cycles (not timer cycles). The clock measured by this event is defined as the physical clock driving the CPU logic.

Related telemetry artifacts

Metrics

- [frontend_stalled_cycles](#)
- [backend_stalled_cycles](#)
- [frontend_bound](#)
- [backend_bound](#)
- [retiring](#)
- [bad_speculation](#)
- [ipc](#)
- [sve_fp_ops_per_cycle](#)
- [nonsve_fp_ops_per_cycle](#)
- [fp_ops_per_cycle](#)
- [branch_port_utilization](#)
- [int_port_utilization](#)
- [vpu_port_utilization](#)
- [lsu_port_utilization](#)
- [std_port_utilization](#)
- [sm_active_cycles_ratio](#)
- [za_active_cycles_ratio](#)
- [cme_alloc_cycles_ratio](#)
- [cme_arb_pending_ratio](#)

Metric groups[Cycle_Accounting](#)[FP_Arithmetic_Intensity](#)[General](#)[Port_Utilization](#)[Topdown_L1](#)**Functional groups**[General](#)**0x0198 IMP_L2_CHI_RX_CBUSY_0, Received RXDAT or RXRSP responses, with CBusy 0, event**

Counts the number of RXDAT or RXRSP responses received with a CBusy value of 0. Note that if an RXDAT flit and RXRSP flit, both with a CBusy value of 0 arrive at the same time this event increments only once.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[General](#)**0x0199 IMP_L2_CHI_RX_CBUSY_1, Received RXDAT or RXRSP responses, with CBusy 1, event**

Counts the number of RXDAT or RXRSP responses received with a CBusy value of 1. Note that if an RXDAT flit and RXRSP flit, both with a CBusy value of 1 arrive at the same time this event increments only once.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[General](#)**0x019A IMP_L2_CHI_RX_CBUSY_2, Received RXDAT or RXRSP responses, with CBusy 2, event**

Counts the number of RXDAT or RXRSP responses received with a CBusy value of 2. Note that if an RXDAT flit and RXRSP flit, both with a CBusy value of 2 arrive at the same time this event increments only once.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[General](#)

0x019B IMP_L2_CHI_RX_CBUSY_3, Received RXDAT or RXRSP responses, with CBusy 3, event

Counts the number of RXDAT or RXRSP responses received with a CBusy value of 3. Note that if an RXDAT flit and RXRSP flit, both with a CBusy value of 3 arrive at the same time this event increments only once.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[General](#)

0x019C IMP_L2_CHI_RX_CBUSY_MT, Received RXDAT or RXRSP responses, with CBusy multi-threaded set, event

Counts the number of RXDAT or RXRSP responses received with CBusy multi-threaded set. Note that if an RXDAT flit and RXRSP flit, both with a CBusy multi-threaded set arrive at the same time this event increments only once.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[General](#)

0x0225 IMP_WFX_CLOCK_CYCLES, Cycles in WFx awake handling snoop, event

Counts cycles in WFx wait state awake handling external snoop traffic.

Related telemetry artifacts**Metrics**

- [frontend_stalled_cycles](#)
- [backend_stalled_cycles](#)
- [frontend_bound](#)
- [backend_bound](#)
- [retiring](#)
- [bad_speculation](#)
- [frontend_core_bound](#)
- [frontend_core_flush_machine_clear_bound](#)
- [frontend_core_flush_bound](#)
- [ipc](#)
- [sve_fp_ops_per_cycle](#)
- [nonsve_fp_ops_per_cycle](#)
- [fp_ops_per_cycle](#)

- [branch_port_utilization](#)
- [int_port_utilization](#)
- [vpu_port_utilization](#)
- [lsu_port_utilization](#)
- [std_port_utilization](#)

Metric groups

[Cycle_Accounting](#)
[FP_Arithmetic_Intensity](#)
[General](#)
[Port_Utilization](#)
[Topdown_Frontend](#)
[Topdown_L1](#)

Functional groups

[General](#)

0x3000 IMP_OP_BRU_ISSUE, Branch operation issued, event

This event counts each resolution from the branch execution pipelines.

Related telemetry artifacts**Metrics**

- [branch_port_utilization](#)

Metric groups

[Port_Utilization](#)

Functional groups

[General](#)

0x3001 IMP_OP_DPU_ISSUE, Integer operation issued, event

This event counts each resolution from the integer execution pipelines.

Related telemetry artifacts**Metrics**

- [int_port_utilization](#)

Metric groups

[Port_Utilization](#)

Functional groups

[General](#)

0x3002 IMP_OP_VPU_ISSUE, Vector/float operation issued, event

This event counts each resolution from the vector execution pipelines.

Related telemetry artifacts**Metrics**

- [vpu_port_utilization](#)

Metric groups

[Port_Utilization](#)

Functional groups

[General](#)

0x3003 IMP_OP_LSU_ISSUE, Load/store operation issued, event

This event counts each resolution from the load store execution pipelines.

Related telemetry artifacts**Metrics**

- [lsu_port_utilization](#)

Metric groups

[Port_Utilization](#)

Functional groups

[General](#)

0x3004 IMP_OP_STD_ISSUE, Store data operation issued, event

This event counts each resolution for store data operations.

Related telemetry artifacts**Metrics**

- [std_port_utilization](#)

Metric groups

[Port_Utilization](#)

Functional groups

[General](#)

0x3212 SM_ACTIVE_CYCLES, Cycles PSTATE.SM enabled, event

The counter counts each cycle counted by CPU_CYCLES when PSTATE.SM was enabled.

Related telemetry artifacts**Metrics**

- [sm_active_cycles_ratio](#)

Metric groups

[Cycle_Accounting](#)

Functional groups

[General](#)

0x3213 CYCLES_CME_ALLOC, Cycles SME2 unit allocated, event

The counter counts each PE cycle counted by CPU_CYCLES where the CPU had an granted arbitration to the SME2 unit, such that the SME2 and Streaming SVE state of the CPU is held in that SME2 unit. It does not count cycles during which a CPU has requested arbitration and is waiting for acknowledgement.

Related telemetry artifacts**Metrics**

- [cme_alloc_cycles_ratio](#)

Metric groups

[Cycle_Accounting](#)

Functional groups

[General](#)

0x3214 CYCLES_ARB_PENDING_CME, Cycles SME2 unit arbitration pending, event

The counter counts each PE cycle counted by CPU_CYCLES where the CPU is in waiting for arbitration while attempting to access the SME2 unit. It can be due an initial arbitration request or contention.

Related telemetry artifacts**Metrics**

- [cme_arb_pending_ratio](#)

Metric groups

[Cycle_Accounting](#)

Functional groups

[General](#)

0x3215 CYCLES_CME_RECONNECT_PENDING, Cycles SME2 unit reconnect pending, event

The counter counts each PE cycle counted by CYCLES_ARB_PENDING_CME where the CPU is in waiting for arbitration due to contention, when it was previously arbitrated to a SME2 unit but arbitration was granted to another CPU.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[General](#)

0x3216 ARB_CME_COUNT, SME2 unit arbitration requests, event

The counter counts the number of times an arbitration to SME2 unit is requested, either an initial request, or a reconnect request due to contention.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[General](#)

0x3217 RECONNECT_CME_COUNT, SME2 unit reconnect requests, event

The counter counts the number of times the CPU needs to request arbitration following a disconnect due to contention.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[General](#)

0x4004 CNT_CYCLES, Constant frequency cycles, event

Increments at a constant frequency equal to the rate of increment of the System Counter, CNTPCT_ELO.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[General](#)

0x8380 ZA_ACTIVE, PSTATE.ZA active cycles, event

The counter counts each cycle counted by CPU_CYCLES when PSTATE.ZA was enabled.

Related telemetry artifacts**Metrics**

- [za_active_cycles_ratio](#)

Metric groups

[Cycle_Accounting](#)

Functional groups

[General](#)

6.16 TLB (TLB) events for C1-Premium

TLB and MMU related events.

Summary of events in TLB:

- Total implemented Common events: 38

- Total Implemented Product ImpDef events: 1
- PMU Only events : 1
- ETE Only events : 0

Table 6-16: TLB events summary

Code	Mnemonic	Name	Description
0x0002	L1I_TLB_REFILL	Level 1 instruction TLB refill	Counts level 1 instruction TLB refills from any Instruction fetch. If there are multiple misses...
0x0005	L1D_TLB_REFILL	Level 1 data TLB refill	Counts level 1 data TLB accesses that resulted in TLB refills. If there are multiple misses in...
0x0025	L1D_TLB	Level 1 data TLB access	Counts level 1 data TLB accesses caused by any memory load or store operation. Note that load or...
0x0026	L1I_TLB	Level 1 instruction TLB access	Counts level 1 instruction TLB accesses, whether the access hits or misses in the TLB. This event...
0x002D	L2D_TLB_REFILL	Level 2 data TLB refill	Counts level 2 TLB refills caused by memory operations from both data and instruction fetch,...
0x002F	L2D_TLB	Level 2 data TLB access	Counts level 2 TLB accesses except those caused by TLB maintenance operations.
0x0034	DTLB_WALK	Data TLB access with at least one translation table walk	Counts number of demand data translation table walks caused by a miss in the L2 TLB and...
0x0035	ITLB_WALK	Instruction TLB access with at least one translation table walk	Counts number of instruction translation table walks caused by a miss in the L2 TLB and...
0x004C	L1D_TLB_REFILL_RD	Level 1 data TLB refill, read	Counts level 1 data TLB refills caused by memory read operations. If there are multiple misses in...
0x004D	L1D_TLB_REFILL_WR	Level 1 data TLB refill, write	Counts level 1 data TLB refills caused by data side memory write operations. If there are...
0x004E	L1D_TLB_RD	Level 1 data TLB access, read	Counts level 1 data TLB accesses caused by memory read operations. This event counts whether the...
0x004F	L1D_TLB_WR	Level 1 data TLB access, write	Counts any L1 data side TLB accesses caused by memory write operations. This event counts whether...
0x005C	L2D_TLB_REFILL_RD	Level 2 data TLB refill, read	Counts level 2 TLB refills caused by memory read operations from both data and instruction fetch...
0x005D	L2D_TLB_REFILL_WR	Level 2 data TLB refill, write	Counts level 2 TLB refills caused by memory write operations from both data and instruction fetch...
0x005E	L2D_TLB_RD	Level 2 data TLB access, read	Counts level 2 TLB accesses caused by memory read operations from both data and instruction fetch...

Code	Mnemonic	Name	Description
0x005F	L2D_TLB_WR	Level 2 data TLB access, write	Counts level 2 TLB accesses caused by memory write operations from both data and instruction...
0x010F	IMP_MMU_L2_TLB_REFILL_PREFETCH_READ	Level 2 data TLB refill, read, prefetch	Counts the number of L2 Unified TLB refills that occur because of a prefetched read, including...
0x8128	DTLB_WALK_PERCYC	Event in progress, DTLB WALK	Counts the number of data translation table walks in progress per cycle.
0x8129	ITLB_WALK_PERCYC	Event in progress, ITLB WALK	Counts the number of instruction translation table walks in progress per cycle.
0x8130	L1D_TLB_RW	Level 1 data TLB demand access	Counts level 1 data TLB demand accesses caused by memory read or write operations. This event...
0x8131	L1I_TLB_RD	Level 1 instruction TLB demand access	Counts level 1 instruction TLB demand accesses whether the access hits or misses in the TLB.
0x8132	L1D_TLB_PRFM	Level 1 data TLB software preload	Counts level 1 data TLB accesses generated by software prefetch or preload memory accesses. Load...
0x8133	L1I_TLB_PRFM	Level 1 instruction TLB software preload	Counts level 1 instruction TLB accesses generated by software preload or prefetch instructions....
0x8134	DTLB_HWUPD	Data TLB hardware update of translation table	Counts number of memory accesses triggered by a data translation table walk and performing an...
0x8135	ITLB_HWUPD	Instruction TLB hardware update of translation table	Counts number of memory accesses triggered by an instruction translation table walk and...
0x8136	DTLB_STEP	Data TLB translation table walk, step	Counts number of memory accesses triggered by a demand data translation table walk and performing...
0x8137	ITLB_STEP	Instruction TLB translation table walk, step	Counts number of memory accesses triggered by an instruction translation table walk and...
0x8138	DTLB_WALK_LARGE	Data TLB large page translation table walk	Counts number of demand data translation table walks caused by a miss in the L2 TLB and yielding...
0x8139	ITLB_WALK_LARGE	Instruction TLB large page translation table walk	Counts number of instruction translation table walks caused by a miss in the L2 TLB and yielding...
0x813A	DTLB_WALK_SMALL	Data TLB small page translation table walk	Counts number of data translation table walks caused by a miss in the L2 TLB and yielding a small...
0x813B	ITLB_WALK_SMALL	Instruction TLB small page translation table walk	Counts number of instruction translation table walks caused by a miss in the L2 TLB and yielding...
0x813C	DTLB_WALK_RW	Data TLB demand access with at least one translation table walk	Counts number of demand data translation table walks caused by a miss in the L2 TLB and...

Code	Mnemonic	Name	Description
0x813D	ITLB_WALK_RD	Instruction TLB demand access with at least one translation table walk	Counts number of demand instruction translation table walks caused by a miss in the L2 TLB and...
0x813E	DTLB_WALK_PRFM	Data TLB software preload access with at least one translation table walk	Counts number of software prefetches or preloads generated data translation table walks caused by...
0x813F	ITLB_WALK_PRFM	Instruction TLB software preload access with at least one translation table walk	Counts number of software prefetches or preloads generated instruction translation table walks...
0x8188	DTLB_WALK_BLOCK	Data TLB block translation table walk	Counts number of demand data translation table walks caused by a miss in the L2 TLB and yielding...
0x8189	ITLB_WALK_BLOCK	Instruction TLB block translation table walk	Counts number of instruction translation table walks caused by a miss in the L2 TLB and yielding...
0x818A	DTLB_WALK_PAGE	Data TLB page translation table walk	Counts number of demand data translation table walks caused by a miss in the L2 TLB and yielding...
0x818B	ITLB_WALK_PAGE	Instruction TLB page translation table walk	Counts number of instruction translation table walks caused by a miss in the L2 TLB and yielding...

For a complete list of the events in C1-Premium, see [PMU events cheat sheet for C1-Premium](#) and [PMU events lookup table for C1-Premium](#).

0x0002 L1I_TLB_REFILL, Level 1 instruction TLB refill, event

Counts level 1 instruction TLB refills from any Instruction fetch. If there are multiple misses in the TLB that are resolved by the refill, then this event only counts once. This event will not count if the translation table walk results in a fault (such as a translation or access fault), since there is no new translation created for the TLB.

Related telemetry artifacts

Metrics

- [l1i_tlb_mpki](#) in [ITLB_Effectiveness](#)
- [l1i_tlb_mpki](#) in [MPKI](#)
- [l1i_tlb_miss_ratio](#) in [ITLB_Effectiveness](#)
- [l1i_tlb_miss_ratio](#) in [Miss_Ratio](#)

Metric groups

[ITLB_Effectiveness](#)

[MPKI](#)

[Miss_Ratio](#)

Functional groups

[TLB](#)

0x0005 L1D_TLB_REFILL, Level 1 data TLB refill, event

Counts level 1 data TLB accesses that resulted in TLB refills. If there are multiple misses in the TLB that are resolved by the refill, then this event only counts once. This event counts for refills caused by preload instructions or hardware prefetch accesses. This event counts regardless of whether the miss hits in L2 or results in a translation table walk. This event will not count if the translation table walk results in a fault (such as a translation or access fault), since there is no new translation created for the TLB. This event will not count on an access from an AT(address translation) instruction.

Related telemetry artifacts

Metrics

- [l1d_tlb_mpki](#) in [DTLB_Effectiveness](#)
- [l1d_tlb_mpki](#) in [MPKI](#)
- [l1d_tlb_miss_ratio](#) in [DTLB_Effectiveness](#)
- [l1d_tlb_miss_ratio](#) in [Miss_Ratio](#)

Metric groups

[DTLB_Effectiveness](#)

[MPKI](#)

[Miss_Ratio](#)

Functional groups

[TLB](#)

0x0025 L1D_TLB, Level 1 data TLB access, event

Counts level 1 data TLB accesses caused by any memory load or store operation. Note that load or store instructions can be broken up into multiple memory operations. This event does not count TLB maintenance operations.

Related telemetry artifacts

Metrics

- [dtlb_walk_ratio](#) in [DTLB_Effectiveness](#)
- [dtlb_walk_ratio](#) in [Miss_Ratio](#)
- [dtlb_walk_large_ratio](#)
- [dtlb_walk_small_ratio](#)
- [dtlb_walk_page_ratio](#)
- [dtlb_walk_block_ratio](#)
- [l1d_tlb_miss_ratio](#) in [DTLB_Effectiveness](#)
- [l1d_tlb_miss_ratio](#) in [Miss_Ratio](#)

Metric groups

[DTLB_Effectiveness](#)

[Miss_Ratio](#)

Functional groups

TLB

0x0026 L1I_TLB, Level 1 instruction TLB access, event

Counts level 1 instruction TLB accesses, whether the access hits or misses in the TLB. This event counts both demand accesses and prefetch or preload generated accesses.

Related telemetry artifacts

Metrics

- [itlb_walk_ratio](#) in [ITLB_Effectiveness](#)
- [itlb_walk_ratio](#) in [Miss_Ratio](#)
- [itlb_walk_large_ratio](#)
- [itlb_walk_small_ratio](#)
- [itlb_walk_page_ratio](#)
- [itlb_walk_block_ratio](#)
- [l1i_tlb_miss_ratio](#) in [ITLB_Effectiveness](#)
- [l1i_tlb_miss_ratio](#) in [Miss_Ratio](#)

Metric groups

[ITLB_Effectiveness](#)

[Miss_Ratio](#)

Functional groups

TLB

0x002D L2D_TLB_REFILL, Level 2 data TLB refill, event

Counts level 2 TLB refills caused by memory operations from both data and instruction fetch, except for those caused by TLB maintenance operations and hardware prefetches.

Related telemetry artifacts

Metrics

- [l2_tlb_mpki](#) in [DTLB_Effectiveness](#)
- [l2_tlb_mpki](#) in [ITLB_Effectiveness](#)
- [l2_tlb_mpki](#) in [MPKI](#)
- [l2_tlb_miss_ratio](#) in [DTLB_Effectiveness](#)
- [l2_tlb_miss_ratio](#) in [ITLB_Effectiveness](#)
- [l2_tlb_miss_ratio](#) in [Miss_Ratio](#)

Metric groups

[DTLB_Effectiveness](#)

[ITLB_Effectiveness](#)

[MPKI](#)

[Miss_Ratio](#)

Functional groups

TLB

0x002F L2D_TLB, Level 2 data TLB access, event

Counts level 2 TLB accesses except those caused by TLB maintenance operations.

Related telemetry artifacts**Metrics**

- [l2_tlb_miss_ratio](#) in [DTLB_Effectiveness](#)
- [l2_tlb_miss_ratio](#) in [ITLB_Effectiveness](#)
- [l2_tlb_miss_ratio](#) in [Miss_Ratio](#)

Metric groups[DTLB_Effectiveness](#)[ITLB_Effectiveness](#)[Miss_Ratio](#)**Functional groups**

TLB

0x0034 DTLB_WALK, Data TLB access with at least one translation table walk, event

Counts number of demand data translation table walks caused by a miss in the L2 TLB and performing at least one memory access. Translation table walks are counted even if the translation ended up taking a translation fault for reasons different than EPD, EOPD and NFD. Note that partial translations that cause a translation table walk are also counted. Also note that this event counts walks triggered by software preloads, but not walks triggered by hardware prefetchers, and that this event does not count walks triggered by TLB maintenance operations.

Related telemetry artifacts**Metrics**

- [dtlb_mpki](#) in [DTLB_Effectiveness](#)
- [dtlb_mpki](#) in [MPKI](#)
- [dtlb_walk_ratio](#) in [DTLB_Effectiveness](#)
- [dtlb_walk_ratio](#) in [Miss_Ratio](#)
- [dtlb_walk_average_depth](#)
- [dtlb_walk_average_latency](#) in [Average_Latency](#)
- [dtlb_walk_average_latency](#) in [DTLB_Effectiveness](#)

Metric groups[Average_Latency](#)[DTLB_Effectiveness](#)[MPKI](#)[Miss_Ratio](#)

Functional groups

TLB

0x0035 ITLB_WALK, Instruction TLB access with at least one translation table walk, event

Counts number of instruction translation table walks caused by a miss in the L2 TLB and performing at least one memory access. Translation table walks are counted even if the translation ended up taking a translation fault for reasons different than EPD, EOPD and NFD. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.

Related telemetry artifacts

Metrics

- [itlb_mpki](#) in [ITLB_Effectiveness](#)
- [itlb_mpki](#) in [MPKI](#)
- [itlb_walk_ratio](#) in [ITLB_Effectiveness](#)
- [itlb_walk_ratio](#) in [Miss_Ratio](#)
- [itlb_walk_average_depth](#)
- [itlb_walk_average_latency](#) in [Average_Latency](#)
- [itlb_walk_average_latency](#) in [ITLB_Effectiveness](#)

Metric groups

[Average_Latency](#)
[ITLB_Effectiveness](#)
[MPKI](#)
[Miss_Ratio](#)

Functional groups

TLB

0x004c L1D_TLB_REFILL_RD, Level 1 data TLB refill, read, event

Counts level 1 data TLB refills caused by memory read operations. If there are multiple misses in the TLB that are resolved by the refill, then this event only counts once. This event counts for refills caused by preload instructions or hardware prefetch accesses. This event counts regardless of whether the miss hits in L2 or results in a translation table walk. This event will not count if the translation table walk results in a fault (such as a translation or access fault), since there is no new translation created for the TLB. This event will not count on an access from an Address Translation (AT) instruction.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

TLB

0x004D L1D_TLB_REFILL_WR, Level 1 data TLB refill, write, event

Counts level 1 data TLB refills caused by data side memory write operations. If there are multiple misses in the TLB that are resolved by the refill, then this event only counts once. This event counts for refills caused by preload instructions or hardware prefetch accesses. This event counts regardless of whether the miss hits in L2 or results in a translation table walk. This event will not count if the table walk results in a fault (such as a translation or access fault), since there is no new translation created for the TLB. This event will not count with an access from an Address Translation (AT) instruction.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

TLB

0x004E L1D_TLB_RD, Level 1 data TLB access, read, event

Counts level 1 data TLB accesses caused by memory read operations. This event counts whether the access hits or misses in the TLB. This event does not count TLB maintenance operations.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

TLB

0x004F L1D_TLB_WR, Level 1 data TLB access, write, event

Counts any L1 data side TLB accesses caused by memory write operations. This event counts whether the access hits or misses in the TLB. This event does not count TLB maintenance operations.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

TLB

0x005C L2D_TLB_REFILL_RD, Level 2 data TLB refill, read, event

Counts level 2 TLB refills caused by memory read operations from both data and instruction fetch except for those caused by TLB maintenance operations or hardware prefetches.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

TLB

0x005D L2D_TLB_REFILL_WR, Level 2 data TLB refill, write, event

Counts level 2 TLB refills caused by memory write operations from both data and instruction fetch except for those caused by TLB maintenance operations.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

TLB

0x005E L2D_TLB_RD, Level 2 data TLB access, read, event

Counts level 2 TLB accesses caused by memory read operations from both data and instruction fetch except for those caused by TLB maintenance operations.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

TLB

0x005F L2D_TLB_WR, Level 2 data TLB access, write, event

Counts level 2 TLB accesses caused by memory write operations from both data and instruction fetch except for those caused by TLB maintenance operations.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

TLB

0x010F IMP_MMU_L2_TLB_REFILL_PREFETCH_READ, Level 2 data TLB refill, read, prefetch, event

Counts the number of L2 Unified TLB refills that occur because of a prefetched read, including hardware and software prefetched reads. It counts both data and instructions.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

TLB

0x8128 DTLB_WALK_PERCYC, Event in progress, DTLB WALK, event

Counts the number of data translation table walks in progress per cycle.

Related telemetry artifacts**Metrics**

- [dtlb_walk_average_latency](#) in [Average_Latency](#)
- [dtlb_walk_average_latency](#) in [DTLB_Effectiveness](#)

Metric groups

[Average_Latency](#)
[DTLB_Effectiveness](#)

Functional groups

[TLB](#)

0x8129 ITLB_WALK_PERCYC, Event in progress, ITLB WALK, event

Counts the number of instruction translation table walks in progress per cycle.

Related telemetry artifacts**Metrics**

- [itlb_walk_average_latency](#) in [Average_Latency](#)
- [itlb_walk_average_latency](#) in [ITLB_Effectiveness](#)

Metric groups

[Average_Latency](#)
[ITLB_Effectiveness](#)

Functional groups

[TLB](#)

0x8130 L1D_TLB_RW, Level 1 data TLB demand access, event

Counts level 1 data TLB demand accesses caused by memory read or write operations. This event counts whether the access hits or misses in the TLB. This event does not count TLB maintenance operations.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[TLB](#)

0x8131 L1I_TLB_RD, Level 1 instruction TLB demand access, event

Counts level 1 instruction TLB demand accesses whether the access hits or misses in the TLB.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[TLB](#)

0x8132 L1D_TLB_PRFM, Level 1 data TLB software preload, event

Counts level 1 data TLB accesses generated by software prefetch or preload memory accesses. Load or store instructions can be broken into multiple memory operations. This event does not count TLB maintenance operations.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[TLB](#)

0x8133 L1I_TLB_PRFM, Level 1 instruction TLB software preload, event

Counts level 1 instruction TLB accesses generated by software preload or prefetch instructions. This event counts whether the access hits or misses in the TLB. This event does not count TLB maintenance operations.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[TLB](#)

0x8134 DTLB_HWUPD, Data TLB hardware update of translation table, event

Counts number of memory accesses triggered by a data translation table walk and performing an update of a translation table entry. Memory accesses are counted even if the translation ended up taking a translation fault for reasons different than EPD, EOPD and NFD. Note that this event counts accesses triggered by software preloads, but not accesses triggered by hardware prefetchers.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[TLB](#)

0x8135 ITLB_HWUPD, Instruction TLB hardware update of translation table, event

Counts number of memory accesses triggered by an instruction translation table walk and performing an update of a translation table entry. Memory accesses are counted even if the translation ended up taking a translation fault for reasons different than EPD, EOPD and NFD.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[TLB](#)

0x8136 DTLB_STEP, Data TLB translation table walk, step, event

Counts number of memory accesses triggered by a demand data translation table walk and performing a read of a translation table entry. Memory accesses are counted even if the translation ended up taking a translation fault for reasons different than EPD, EOPD and NFD. Note that this event counts accesses triggered by software preloads, but not accesses triggered by hardware prefetchers.

Related telemetry artifacts

Metrics

- [dtlb_walk_average_depth](#)

Metric groups

[DTLB_Effectiveness](#)

Functional groups

[TLB](#)

0x8137 ITLB_STEP, Instruction TLB translation table walk, step, event

Counts number of memory accesses triggered by an instruction translation table walk and performing a read of a translation table entry. Memory accesses are counted even if the translation ended up taking a translation fault for reasons different than EPD, EOPD and NFD.

Related telemetry artifacts

Metrics

- [itlb_walk_average_depth](#)

Metric groups

[ITLB_Effectiveness](#)

Functional groups

[TLB](#)

0x8138 DTLB_WALK_LARGE, Data TLB large page translation table walk, event

Counts number of demand data translation table walks caused by a miss in the L2 TLB and yielding a large page. The set of large pages is defined as all pages with a final size higher than or equal to 2MB. Translation table walks that end up taking a translation fault are not counted, as the page size would be undefined in that case. If DTLB_WALK_BLOCK is implemented, then it is an alias for this event in this family. Note that partial translations that cause a translation table walk are also counted. Also note that this event counts walks triggered by software preloads, but not walks triggered by hardware prefetchers, and that this event does not count walks triggered by TLB maintenance operations.

Related telemetry artifacts

Metrics

- [dtlb_walk_large_ratio](#)

Metric groups

[DTLB_Effectiveness](#)

Functional groups

TLB

0x8139 ITLB_WALK_LARGE, Instruction TLB large page translation table walk, event

Counts number of instruction translation table walks caused by a miss in the L2 TLB and yielding a large page. The set of large pages is defined as all pages with a final size higher than or equal to 2MB. Translation table walks that end up taking a translation fault are not counted, as the page size would be undefined in that case. In this family, this is equal to ITLB_WALK_BLOCK event. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.

Related telemetry artifacts

Metrics

- [itlb_walk_large_ratio](#)

Metric groups

[ITLB_Effectiveness](#)

Functional groups

TLB

0x813A DTLB_WALK_SMALL, Data TLB small page translation table walk, event

Counts number of data translation table walks caused by a miss in the L2 TLB and yielding a small page. The set of small pages is defined as all pages with a final size lower than 2MB. Translation table walks that end up taking a translation fault are not counted, as the page size would be undefined in that case. If DTLB_WALK_PAGE event is implemented, then it is an alias for this event in this family. Note that partial translations that cause a translation table walk are also counted. Also note that this event counts walks triggered by software preloads, but not walks triggered by hardware prefetchers, and that this event does not count walks triggered by TLB maintenance operations.

Related telemetry artifacts

Metrics

- [dtlb_walk_small_ratio](#)

Metric groups

[DTLB_Effectiveness](#)

Functional groups

TLB

0x813B ITLB_WALK_SMALL, Instruction TLB small page translation table walk, event

Counts number of instruction translation table walks caused by a miss in the L2 TLB and yielding a small page. The set of small pages is defined as all pages with a final size lower than 2MB. Translation table walks that end up taking a translation fault are not counted, as the page size would be undefined in that case. In this family, this is equal to ITLB_WALK_PAGE event. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.

Related telemetry artifacts

Metrics

- [itlb_walk_small_ratio](#)

Metric groups

[ITLB_Effectiveness](#)

Functional groups

[TLB](#)

0x813C DTLB_WALK_RW, Data TLB demand access with at least one translation table walk, event

Counts number of demand data translation table walks caused by a miss in the L2 TLB and performing at least one memory access. Translation table walks are counted even if the translation ended up taking a translation fault for reasons different than EPD, EOPD and NFD. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[TLB](#)

0x813D ITLB_WALK_RD, Instruction TLB demand access with at least one translation table walk, event

Counts number of demand instruction translation table walks caused by a miss in the L2 TLB and performing at least one memory access. Translation table walks are counted even if the translation ended up taking a translation fault for reasons different than EPD, EOPD and NFD. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[TLB](#)

0x813E DTLB_WALK_PRFM, Data TLB software preload access with at least one translation table walk, event

Counts number of software prefetches or preloads generated data translation table walks caused by a miss in the L2 TLB and performing at least one memory access. Translation table walks are counted even if the translation ended up taking a translation fault for reasons different than EPD, EOPD and NFD. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[TLB](#)

0x813F ITLB_WALK_PRFM, Instruction TLB software preload access with at least one translation table walk, event

Counts number of software prefetches or preloads generated instruction translation table walks caused by a miss in the L2 TLB and performing at least one memory access. Translation table walks are counted even if the translation ended up taking a translation fault for reasons different than EPD, EOPD and NFD. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[TLB](#)

0x8188 DTLB_WALK_BLOCK, Data TLB block translation table walk, event

Counts number of demand data translation table walks caused by a miss in the L2 TLB and yielding a block descriptor at level 1 or level 2 of the translation. This means that the walk yielded a final size higher than or equal to 2MB. Translation table walks that end up taking a translation fault are not counted, as the page size would be undefined in that case. In this family, this is an alias to DTLB_WALK_LARGE event. Note that partial translations that cause a translation table walk are also counted. Also note that this event counts walks triggered by software preloads, but not walks triggered by hardware prefetchers, and that this event does not count walks triggered by TLB maintenance operations.

Related telemetry artifacts

Metrics

- [dtlb_walk_block_ratio](#)

Metric groups

[DTLB_Effectiveness](#)

Functional groups

[TLB](#)

0x8189 ITLB_WALK_BLOCK, Instruction TLB block translation table walk, event

Counts number of instruction translation table walks caused by a miss in the L2 TLB and yielding a block descriptor at level 1 or level 2 of the translation. Translation table walks that end up taking a translation fault are not counted, as the page size would be undefined in that case. In this family, this is equal to ITLB_WALK_LARGE event. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.

Related telemetry artifacts

Metrics

- [itlb_walk_block_ratio](#)

Metric groups

[ITLB_Effectiveness](#)

Functional groups

[TLB](#)

0x818A DTLB_WALK_PAGE, Data TLB page translation table walk, event

Counts number of demand data translation table walks caused by a miss in the L2 TLB and yielding a page descriptor at level 3 of the translation. This means that the walk yielded a final size lower than 2MB. Translation table walks that end up taking a translation fault are not counted, as the page size would be undefined in that case. In this family, this is an alias to DTLB_WALK_SMALL event. Note that partial translations that cause a translation table walk are also counted. Also note that this event counts walks triggered by software preloads, but not walks triggered by hardware prefetchers, and that this event does not count walks triggered by TLB maintenance operations.

Related telemetry artifacts

Metrics

- [dtlb_walk_page_ratio](#)

Metric groups

[DTLB_Effectiveness](#)

Functional groups

[TLB](#)

0x818B ITLB_WALK_PAGE, Instruction TLB page translation table walk, event

Counts number of instruction translation table walks caused by a miss in the L2 TLB and yielding a page descriptor at level 3 of the translation. Translation table walks that end up taking a translation fault are not counted, as the page size would be undefined in that case. In this family, this is equal to ITLB_WALK_SMALL event. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.

Related telemetry artifacts

Metrics

- [itlb_walk_page_ratio](#)

Metric groups

[ITLB_Effectiveness](#)

Functional groups

[TLB](#)

6.17 SVE (SVE) events for C1-Premium

SVE related events.

Summary of events in SVE:

- Total implemented Common events: 13
- Total Implemented Product ImpDef events: 5
- PMU Only events : 5
- ETE Only events : 0

Table 6-17: SVE events summary

Code	Mnemonic	Name	Description
0x3234	SSVE_PRED_SPEC	Operation speculatively executed, Streaming SVE predicated	Counts operations counted by SVE_PRED_SPEC, but in Streaming mode only. Note: this counts SME...
0x3235	SSVE_PRED_EMPTY_SPEC	Operation speculatively executed, Streaming SVE predicated with no active predicates	Counts operations counted by SVE_PRED_EMPTY_SPEC, but in Streaming mode only.
0x3236	SSVE_PRED_FULL_SPEC	Operation speculatively executed, Streaming SVE predicated with all active predicates	Counts speculatively executed predicated SVE operations with all predicate elements active,...
0x3237	SSVE_PRED_NOT_FULL_SPEC	Operation speculatively executed, Streaming SVE predicated with no active or partially active predicates	Counts speculatively executed predicated SVE operations with at least one non active predicate...
0x3238	SSVE_PRED_PARTIAL_SPEC	Operation speculatively executed, Streaming SVE predicated with partially active predicates	Counts speculatively executed predicated SVE operations with at least one but not all active...
0x8006	SVE_INST_SPEC	Operation speculatively executed, SVE, including load and store	The counter counts each Speculatively executed operation due to an SVE instruction. It is...
0x8074	SVE_PRED_SPEC	Operation speculatively executed, SVE predicated	Counts speculatively executed predicated SVE operations.
0x8075	SVE_PRED_EMPTY_SPEC	Operation speculatively executed, SVE predicated with no active predicates	Counts speculatively executed predicated SVE operations with no active predicate elements.
0x8076	SVE_PRED_FULL_SPEC	Operation speculatively executed, SVE predicated with all active predicates	Counts speculatively executed predicated SVE operations with all predicate elements active.
0x8077	SVE_PRED_PARTIAL_SPEC	Operation speculatively executed, SVE predicated with partially active predicates	Counts speculatively executed predicated SVE operations with at least one but not all active...
0x8078	SVE_UNPRED_SPEC	Operation speculatively executed, SVE unpredicated	The counter counts each Speculatively executed SIMD data-processing, load, or store operation...
0x8079	SVE_PRED_NOT_FULL_SPEC	SVE predicated operations speculatively executed with no active or partially active predicates	Counts speculatively executed predicated SVE operations with at least one non active predicate...
0x80BC	SVE_LDFF_SPEC	Operation speculatively executed, SVE first-fault load	Counts speculatively executed SVE first fault or non-fault load operations.

Code	Mnemonic	Name	Description
0x80BD	SVE_LDFF_FAULT_SPEC	Operation speculatively executed, SVE first-fault load which set FFR bit to 0b0	Counts speculatively executed SVE first fault or non-fault load operations that clear at least...
0x80E3	ASE_SVE_INT8_SPEC	Integer operation speculatively executed, Advanced SIMD or SVE 8-bit	Counts speculatively executed Advanced SIMD or SVE integer operations with the largest data type...
0x80E7	ASE_SVE_INT16_SPEC	Integer operation speculatively executed, Advanced SIMD or SVE 16-bit	Counts speculatively executed Advanced SIMD or SVE integer operations with the largest data type...
0x80EB	ASE_SVE_INT32_SPEC	Integer operation speculatively executed, Advanced SIMD or SVE 32-bit	Counts speculatively executed Advanced SIMD or SVE integer operations with the largest data type...
0x80EF	ASE_SVE_INT64_SPEC	Integer operation speculatively executed, Advanced SIMD or SVE 64-bit	Counts speculatively executed Advanced SIMD or SVE integer operations with the largest data type...

For a complete list of the events in C1-Premium, see [PMU events cheat sheet for C1-Premium](#) and [PMU events lookup table for C1-Premium](#).

0x3234 SSVE_PRED_SPEC, Operation speculatively executed, Streaming SVE predicated, event

Counts operations counted by SVE_PRED_SPEC, but in Streaming mode only.

Note: this counts SME operations requiring PSTATE.ZA to be set, including 2D operations.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[SVE](#)

0x3235 SSVE_PRED_EMPTY_SPEC, Operation speculatively executed, Streaming SVE predicated with no active predicates, event

Counts operations counted by SVE_PRED_EMPTY_SPEC, but in Streaming mode only.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[SVE](#)

0x3236 SSVE_PRED_FULL_SPEC, Operation speculatively executed, Streaming SVE predicated with all active predicates, event

Counts speculatively executed predicated SVE operations with all predicate elements active, specifically in Streaming mode.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[SVE](#)

0x3237 SSVE_PRED_NOT_FULL_SPEC, Operation speculatively executed, Streaming SVE predicated with no active or partially active predicates, event

Counts speculatively executed predicated SVE operations with at least one non active predicate elements, specifically in Streaming mode.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[SVE](#)

0x3238 SSVE_PRED_PARTIAL_SPEC, Operation speculatively executed, Streaming SVE predicated with partially active predicates, event

Counts speculatively executed predicated SVE operations with at least one but not all active predicate elements, specifically in streaming mode.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[SVE](#)

0x8006 SVE_INST_SPEC, Operation speculatively executed, SVE, including load and store, event

The counter counts each Speculatively executed operation due to an SVE instruction.

It is **IMPLEMENTATION DEFINED** whether the counter counts operations due to non-SIMD SVE instructions.

Instructions classified as SME instructions and counted by SME_INST_SPEC are not counted by this event.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Spec_Operation](#)

[SVE](#)

0x8074 SVE_PRED_SPEC, Operation speculatively executed, SVE predicated, event

Counts speculatively executed predicated SVE operations.

Related telemetry artifacts**Metrics**

- [sve_predicate_percentage](#)
- [sve_predicate_full_percentage](#)
- [sve_predicate_partial_percentage](#)
- [sve_predicate_empty_percentage](#)

Metric groups

[SVE_Effectiveness](#)

Functional groups

[SVE](#)

0x8075 SVE_PRED_EMPTY_SPEC, Operation speculatively executed, SVE predicated with no active predicates, event

Counts speculatively executed predicated SVE operations with no active predicate elements.

Related telemetry artifacts**Metrics**

- [sve_predicate_empty_percentage](#)

Metric groups

[SVE_Effectiveness](#)

Functional groups

[SVE](#)

0x8076 SVE_PRED_FULL_SPEC, Operation speculatively executed, SVE predicated with all active predicates, event

Counts speculatively executed predicated SVE operations with all predicate elements active.

Related telemetry artifacts**Metrics**

- [sve_predicate_full_percentage](#)

Metric groups

[SVE_Effectiveness](#)

Functional groups

[SVE](#)

0x8077 SVE_PRED_PARTIAL_SPEC, Operation speculatively executed, SVE predicated with partially active predicates, event

Counts speculatively executed predicated SVE operations with at least one but not all active predicate elements.

Related telemetry artifacts**Metrics**

- [sve_predicate_partial_percentage](#)

Metric groups[SVE_Effectiveness](#)**Functional groups**[SVE](#)**0x8078 SVE_UNPRED_SPEC, Operation speculatively executed, SVE unpredicated, event**

The counter counts each Speculatively executed SIMD data-processing, load, or store operation due to an SVE instruction without a Governing predicate operand.

This counts the SME operations requiring PSTATE.ZA to be set. It does not count Advanced SIMD operations.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[SVE](#)**0x8079 SVE_PRED_NOT_FULL_SPEC, SVE predicated operations speculatively executed with no active or partially active predicates, event**

Counts speculatively executed predicated SVE operations with at least one non active predicate elements.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[SVE](#)**0x80BC SVE_LDFF_SPEC, Operation speculatively executed, SVE first-fault load, event**

Counts speculatively executed SVE first fault or non-fault load operations.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[SVE](#)

0x80BD SVE_LDFF_FAULT_SPEC, Operation speculatively executed, SVE first-fault load which set FFR bit to 0b0, event

Counts speculatively executed SVE first fault or non-fault load operations that clear at least one bit in the FFR.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[SVE](#)

0x80E3 ASE_SVE_INT8_SPEC, Integer operation speculatively executed, Advanced SIMD or SVE 8-bit, event

Counts speculatively executed Advanced SIMD or SVE integer operations with the largest data type an 8-bit integer.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[SVE](#)

0x80E7 ASE_SVE_INT16_SPEC, Integer operation speculatively executed, Advanced SIMD or SVE 16-bit, event

Counts speculatively executed Advanced SIMD or SVE integer operations with the largest data type a 16-bit integer.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[SVE](#)

0x80EB ASE_SVE_INT32_SPEC, Integer operation speculatively executed, Advanced SIMD or SVE 32-bit, event

Counts speculatively executed Advanced SIMD or SVE integer operations with the largest data type a 32-bit integer.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[SVE](#)

0x80EF ASE_SVE_INT64_SPEC, Integer operation speculatively executed, Advanced SIMD or SVE 64-bit, event

Counts speculatively executed Advanced SIMD or SVE integer operations with the largest data type a 64-bit integer.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[SVE](#)

6.18 Non_PMU (NON_PMU) events for C1-Premium

Non-PMU related events.

Summary of events in Non_PMU:

- Total implemented Common events: 2
- Total Implemented Product ImpDef events: 0
- PMU Only events : 0
- ETE Only events : 2

Table 6-18: Non_PMU events summary

Code	Mnemonic	Name	Description
0x400D	PMU_OVFS	PMU overflow, counters accessible to EL1 and EL0	This event is generated each time an event causes a PMEVCTNR<n>_EL1 counter overflow when...
0x400F	PMU_HOVFS	PMU overflow, counters reserved for use by EL2	This event is generated each time an event causes a PMEVCTNR<n>_EL1 counter overflow when...

For a complete list of the events in C1-Premium, see [PMU events cheat sheet for C1-Premium](#) and [PMU events lookup table for C1-Premium](#).

0x400D PMU_OVFS, PMU overflow, counters accessible to EL1 and EL0, event

This event is generated each time an event causes a PMEVCTNR<n>_EL1 counter overflow when PMINTENSET_EL1[n] is set to 1, for each implemented PMU counter n in the range 0 ≤ n < UInt(MDCR_EL2.HPMN), and the Cycle Counter (n = 31).

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Non_PMU](#)

0x400F PMU_HOVFS, PMU overflow, counters reserved for use by EL2, event

This event is generated each time an event causes a PMEVTCTNR<n>_EL1 counter overflow when PMINTENSET_EL1[n] is set to 1, for each implemented PMU counter n in the range $UInt(MDCR_EL2.HPMN) \leq n < UInt(PMCR_ELO.N)$. This event is not transmitted to a PE Trace Unit while $TRCFR_EL2.E2TRE == 0b0$.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

Non_PMU

6.19 TRACE (TRACE) events for C1-Premium

Trace related events.

Summary of events in TRACE:

- Total implemented Common events: 10
- Total Implemented Product ImpDef events: 0
- PMU Only events : 0
- ETE Only events : 1

Table 6-19: TRACE events summary

Code	Mnemonic	Name	Description
0x400C	TRB_WRAP	Trace buffer current write pointer wrapped	This event is generated each time the current write pointer is wrapped to the base pointer.
0x400E	TRB_TRIG	Trace buffer Trigger Event	This event is generated when a Trace Buffer Extension Trigger Event occurs.
0x4010	TRCEXTOUT0	Trace unit external output 0	This event is generated each time an event is signaled by ETE external event 0.
0x4011	TRCEXTOUT1	Trace unit external output 1	This event is generated each time an event is signaled by ETE external event 1.
0x4012	TRCEXTOUT2	Trace unit external output 2	This event is generated each time an event is signaled by ETE external event 2.
0x4013	TRCEXTOUT3	Trace unit external output 3	This event is generated each time an event is signaled by ETE external event 3.
0x4018	CTI_TRIGOUT4	Cross-trigger Interface output trigger 4	This event is generated each time an event is signaled on CTI output trigger 4.
0x4019	CTI_TRIGOUT5	Cross-trigger Interface output trigger 5	This event is generated each time an event is signaled on CTI output trigger 5.
0x401A	CTI_TRIGOUT6	Cross-trigger Interface output trigger 6	This event is generated each time an event is signaled on CTI output trigger 6.
0x401B	CTI_TRIGOUT7	Cross-trigger Interface output trigger 7	This event is generated each time an event is signaled on CTI output trigger 7.

For a complete list of the events in C1-Premium, see [PMU events cheat sheet for C1-Premium](#) and [PMU events lookup table for C1-Premium](#).

0x400c TRB_WRAP, Trace buffer current write pointer wrapped, event

This event is generated each time the current write pointer is wrapped to the base pointer.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[TRACE](#)

0x400e TRB_TRIG, Trace buffer Trigger Event, event

This event is generated when a Trace Buffer Extension Trigger Event occurs.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[TRACE](#)

0x4010 TRCEXTOUT0, Trace unit external output 0, event

This event is generated each time an event is signaled by ETE external event 0.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[TRACE](#)

0x4011 TRCEXTOUT1, Trace unit external output 1, event

This event is generated each time an event is signaled by ETE external event 1.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[TRACE](#)

0x4012 TRCEXTOUT2, Trace unit external output 2, event

This event is generated each time an event is signaled by ETE external event 2.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

TRACE

0x4013 TRCEXTOUT3, Trace unit external output 3, event

This event is generated each time an event is signaled by ETE external event 3.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

TRACE

0x4018 CTI_TRIGOUT4, Cross-trigger Interface output trigger 4, event

This event is generated each time an event is signaled on CTI output trigger 4.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

TRACE

0x4019 CTI_TRIGOUT5, Cross-trigger Interface output trigger 5, event

This event is generated each time an event is signaled on CTI output trigger 5.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

TRACE

0x401A CTI_TRIGOUT6, Cross-trigger Interface output trigger 6, event

This event is generated each time an event is signaled on CTI output trigger 6.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

TRACE

0x401B CTI_TRIGOUT7, Cross-trigger Interface output trigger 7, event

This event is generated each time an event is signaled on CTI output trigger 7.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

TRACE

6.20 CPU_Debug (CPU_DEBUG) events for C1-Premium

CPU performance debug related events.

Summary of events in CPU_Debug:

- Total implemented Common events: 0
- Total Implemented Product ImpDef events: 7
- PMU Only events : 7
- ETE Only events : 0

Table 6-20: CPU_Debug events summary

Code	Mnemonic	Name	Description
0x0120	IMP_CT_FLUSH	Pipeline flushes in the commit unit	Counts the number of pipeline flushes in the commit unit. These flushes refer to architectural...
0x0121	IMP_CT_FLUSH_MEM_HAZARD	Non-speculative pipeline flushes in the commit unit due to memory hazards	Counts the number of non-speculative pipeline flushes in the commit unit caused by memory...
0x0122	IMP_CT_FLUSH_BAD_BRANCH	Pipeline flushes in the commit unit due to non-branch instructions predicted as branch	Counts the number of pipeline flushes in the commit unit caused by a non-branch instruction...
0x0124	IMP_CT_FLUSH_ISB	Pipeline flushes in the commit unit due to Instruction Synchronization Barrier or similar	Counts the number of pipeline flushes in the commit unit caused by an Instruction Synchronization...
0x0125	IMP_CT_FLUSH_OTHER	Pipeline flushes in the commit unit due to other hazards	Counts the number of pipeline flushes in the commit unit that are caused by rare instruction...
0x0127	IMP_LS_RAR_HAZARD	Generated load store hazards due to a Read-After-Read ordering hazard	Counts any load store detected hazards that are generated due to a Read-After-Read (RAR) ordering...
0x0128	IMP_LS_RAW_HAZARD	Generated load store hazards due to a Read-After-Write ordering hazard	Counts any load store detected hazards that are generated due to a Read-After-Write (RAW)...

For a complete list of the events in C1-Premium, see [PMU events cheat sheet for C1-Premium](#) and [PMU events lookup table for C1-Premium](#).

0x0120 IMP_CT_FLUSH, Pipeline flushes in the commit unit, event

Counts the number of pipeline flushes in the commit unit. These flushes refer to architectural and microarchitectural flushes that are resolved when an instruction is no longer speculative. It also includes speculative branch redirects for mispredicted branches. These redirects are not necessarily full pipeline flushes, and they may occur before the branch is the oldest instruction in flight.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[CPU_Debug](#)**0x0121 IMP_CT_FLUSH_MEM_HAZARD, Non-speculative pipeline flushes in the commit unit due to memory hazards, event**

Counts the number of non-speculative pipeline flushes in the commit unit caused by memory hazards, including Read-After-Write, Write-After- Read, and Write-After-Write hazards.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[CPU_Debug](#)**0x0122 IMP_CT_FLUSH_BAD_BRANCH, Pipeline flushes in the commit unit due to non-branch instructions predicted as branch, event**

Counts the number of pipeline flushes in the commit unit caused by a non-branch instruction predicted as a branch. This type of flush occurs when the branch predictor incorrectly identifies a regular instruction as a branch, which causes speculative execution to follow an incorrect control flow path. When the CPU later detects this misprediction, it must flush the pipeline and restart execution from the correct instruction. Counts flushes due to non-branch instructions predicted as a branch.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[CPU_Debug](#)**0x0124 IMP_CT_FLUSH_ISB, Pipeline flushes in the commit unit due to Instruction Synchronization Barrier or similar, event**

Counts the number of pipeline flushes in the commit unit caused by an Instruction Synchronization Barrier (ISB) or an instruction that causes similar side effects. An ISB is a barrier instruction used to ensure that all previous instructions have completed before any subsequent instructions are fetched and executed. It is commonly used when switching execution contexts, modifying system registers, or updating the instruction-related state, for example, after self-modifying code. ISBs are necessary for correctness. However, they introduce pipeline stalls, because they force instruction fetch and decode stages to restart.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups[CPU_Debug](#)

0x0125 IMP_CT_FLUSH_OTHER, Pipeline flushes in the commit unit due to other hazards, event

Counts the number of pipeline flushes in the commit unit that are caused by rare instruction sequences or microarchitecture states.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[CPU_Debug](#)

0x0127 IMP_LS_RAR_HAZARD, Generated load store hazards due to a Read-After-Read ordering hazard, event

Counts any load store detected hazards that are generated due to a Read-After-Read (RAR) ordering hazard. This hazard occurs when a load operation is incorrectly speculated or executed out-of-order relative to an earlier load. It leads to potential data inconsistencies. To maintain memory ordering correctness, the CPU invalidates the speculatively executed load instructions and reissues them in the correct order. It flushes the pipeline and execution stalls.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[CPU_Debug](#)

0x0128 IMP_LS_RAW_HAZARD, Generated load store hazards due to a Read-After-Write ordering hazard, event

Counts any load store detected hazards that are generated due to a Read-After-Write (RAW) ordering hazard. This hazard occurs when a younger load instruction executes before an older store to the same memory location. It leads to incorrect data being read. The CPU detects such violations and triggers a load store flush. It clears the pipeline and re-executes affected instructions to ensure a correct execution order.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[CPU_Debug](#)

6.21 Coherency (COHERENCY) events for C1-Premium

Coherency related events.

Summary of events in Coherency:

- Total implemented Common events: 3

- Total Implemented Product ImpDef events: 0
- PMU Only events : 0
- ETE Only events : 0

Table 6-21: Coherency events summary

Code	Mnemonic	Name	Description
0x8190	ISNP_HIT_RD	Snoop hit, demand instruction fetch	Counts each instruction access that is satisfied by a snoop request that hits in a cache outside...
0x81A4	DSNP_HIT_HWPRF	Snoop hit, hardware data prefetch	Counts each data hardware prefetch that is satisfied by a snoop request that hits in a cache...
0x81B4	DSNP_HIT	Snoop hit, data	Counts each data access that is satisfied by a snoop request that hits in a cache outside of the...

For a complete list of the events in C1-Premium, see [PMU events cheat sheet for C1-Premium](#) and [PMU events lookup table for C1-Premium](#).

0x8190 ISNP_HIT_RD, Snoop hit, demand instruction fetch, event

Counts each instruction access that is satisfied by a snoop request that hits in a cache outside of the cache hierarchy of this PE

Related telemetry artifacts

Metrics

- [system_peer_cluster_cache_hit_ratio](#)

Metric groups

[System_Memory_Effectiveness](#)

Functional groups

[Coherency](#)

0x81A4 DSNP_HIT_HWPRF, Snoop hit, hardware data prefetch, event

Counts each data hardware prefetch that is satisfied by a snoop request that hits in a cache outside of the cache hierarchy of this PE.

Related telemetry artifacts

There are no related metrics or metric groups because this event is not used in this Telemetry Specification.

Functional groups

[Coherency](#)

0x81B4 DSNP_HIT, Snoop hit, data, event

Counts each data access that is satisfied by a snoop request that hits in a cache outside of the cache hierarchy of this PE.

Related telemetry artifacts

Metrics

- [system_peer_cluster_cache_hit_ratio](#)

Metric groups

System_Memory_Effectiveness

Functional groups

Coherency

7. Supplemental performance debug PMU events

C1-Premium implements an additional set of events that are available for use in debugging the performance behaviors of the processor. These events are not guaranteed to have the same level of accuracy as the architected PMU counters. Any use of these events must take this variable accuracy into account.

The following table shows the events.

Table 7-1: Supplemental performance debug PMU events for C1-Premium

Code	Mnemonic	Description
0x120	IMP_CT_FLUSH	Flushes including architectural, microarchitectural, and branch redirects
0x121	IMP_CT_FLUSH_MEM	Flushes due to memory hazards, including Read-After-Write, Write-After-Read, and Write-After-Write hazards
0x122	IMP_CT_FLUSH_BAD_BRANCH	Flushes due to non-branch instructions predicted as a branch
0x124	IMP_CT_FLUSH_ISB	Flushes due to an Instruction Synchronization Barrier (ISB) or an instruction that causes similar side-effects or similar side-effects
0x125	IMP_CT_FLUSH_OTHER	Flushes due to other hazards
0x127	IMP_LS_RAR	Loadstore detected nuke due to read-after-read ordering hazard
0x128	IMP_LS_RAW	Loadstore detected nuke due to read-after-write ordering hazard

Proprietary Notice

This document is protected by copyright and other related rights and the use or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm Limited ("Arm"). No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether the subject matter of this document infringes any third party patents.

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm's view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. In addition, you are responsible for any applications which are used in conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

This document may include technical inaccuracies or typographical errors. THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, any patents, copyrights, trade secrets, trademarks, or other rights.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted use, duplication, or disclosure of this document complies fully with any relevant

export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of this document shall prevail.

The validity, construction and performance of this notice shall be governed by English Law.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. Please follow Arm’s trademark usage guidelines at <https://www.arm.com/company/policies/trademarks>. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

Product and document information

Read the information in these sections to understand the release status of the product and documentation, and the conventions used in Arm documents.

Product status

All products and services provided by Arm require deliverables to be prepared and made available at different levels of completeness. The information in this document indicates the appropriate level of completeness for the associated deliverables.

Product completeness status

The information in this document is Final, that is for a developed product.

Revision history

These sections can help you understand how the document has changed over time.

Document release information

The Document history table gives the issue number and the released date for each released issue of this document.

Document history

Issue	Date	Confidentiality	Change
0200-02	10 September 2025	Non-Confidential	Second issue for all revisions of Arm® C1-Premium.
0100-01	30 August 2024	Confidential	First issue for all revisions of Arm® C1-Premium.

Change history

The Change history tables describe the technical changes between released issues of this document in reverse order. Issue numbers match the revision history in [Document release information](#) on page 309.

Table 2: Issue 0100-01

Change	Location
First release for all revisions of Arm® C1-Premium.	-

Table 3: Differences between Issue 0100-01 and 0200-02

Change	Location
Second issue for all revisions of Arm® C1-Premium.	-
Previous release issue numbering and change information realigned for consistency. Corresponding release date information remains unchanged.	-
Editorial changes	Throughout the document
Added l1d_cache_demand_mпки and l2d_cache_demand_mпки	Metrics lookup table for C1-Premium
Expanded frontend_core_spec_throttle_bound and backend_core_spec_throttle_bound information added	Topdown Frontend metric group Topdown Backend metric group
System Memory Effectiveness section added	System Memory Effectiveness metric group
Additional metrics, events, and metric groups added throughout the specification	-

Conventions

The following subsections describe conventions used in Arm documents.

Glossary

The Arm Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm Glossary for more information: developer.arm.com/glossary.

Typographic conventions

Arm documentation uses typographical conventions to convey specific meaning.

Convention	Use
<i>italic</i>	Citations.
bold	Terms in descriptive lists, where appropriate.
monospace	Text that you can enter at the keyboard, such as commands, file and program names, and source code.
monospace <u>underline</u>	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: <pre>MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2></pre>
SMALL CAPITALS	Terms that have specific technical meanings as defined in the <i>Arm® Glossary</i> . For example, IMPLEMENTATION DEFINED , IMPLEMENTATION SPECIFIC , UNKNOWN , and UNPREDICTABLE .



We recommend the following. If you do not follow these recommendations your system might not work.



Your system requires the following. If you do not follow these requirements your system will not work.



You are at risk of causing permanent damage to your system or your equipment, or of harming yourself.



This information is important and needs your attention.



This information might help you perform a task in an easier, better, or faster way.

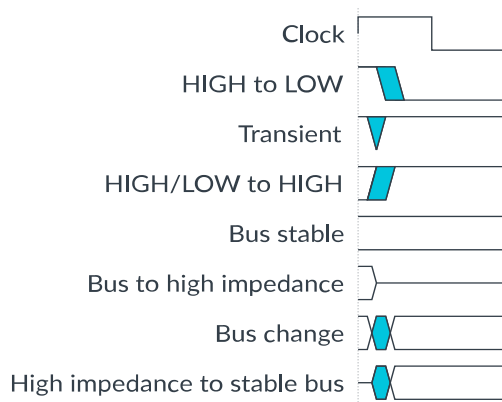


This information reminds you of something important relating to the current content.

Timing diagrams

The following figure explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.

Figure 1: Key to timing diagram conventions

Signals

The signal conventions are:

Signal level

The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:

- HIGH for active-HIGH signals.
- LOW for active-LOW signals.

Lowercase n

At the start or end of a signal name, n denotes an active-LOW signal.

Useful resources

This document contains information that is specific to this product. See the following resources for other useful information.

Arm documents are available on developer.arm.com/documentation.

Confidential documents are only available to licensees, when logged in. Each document link in the tables below provides direct access to the online version of the document.

Arm product resources	Document ID	Confidentiality
Arm® C1-Premium Core Technical Reference Manual	109950	Confidential
Arm® C1-Scalable Matrix Extension 2 Telemetry Specification	108821	Non-Confidential
Arm® CPU Telemetry Solution Topdown Methodology Specification	109542	Non-Confidential
Arm® Telemetry Solution GitLab repository	–	Non-Confidential
Arm® Telemetry on Arm Developer	–	Non-Confidential

Arm architecture and specifications	Document ID	Confidentiality
Arm® Architecture Reference Manual for A-profile architecture	DDI 0487	Non-Confidential