Performance Monitor Counters for AMD Family 1Ah Model 00h-0Fh Processors

Legal Notices

The information contained herein is for informational purposes only, and is subject to change without notice. While every precaution has been taken in the preparation of these materials, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of these materials, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by these materials. Terms and limitations applicable to the purchase or use of AMD’s products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale.

Trademarks:
AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc.
CXL is a trademark of Compute Express Link Consortium, Inc.
Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

Reverse engineering or disassembly is prohibited.

USE OF THIS PRODUCT IN ANY MANNER THAT COMPLIES WITH THE MPEG ACTUAL OR DE FACTO VIDEO AND/OR AUDIO STANDARDS IS EXPRESSLY PROHIBITED WITHOUT ALL NECESSARY LICENSES UNDER APPLICABLE PATENTS. SUCH LICENSES MAY BE ACQUIRED FROM VARIOUS THIRD PARTIES INCLUDING, BUT NOT LIMITED TO, IN THE MPEG PATENT PORTFOLIO, WHICH LICENSE IS AVAILABLE FROM MPEG LA, L.L.C., 6312 S. FIDDLERS GREEN CIRCLE, SUITE 400E, GREENWOOD VILLAGE, COLORADO 80111.

List of Chapters

Table of Contents

List of Tables

1 Performance Monitor Counters

1.1 RDPMC Assignments

There are six core performance event counters per thread, six performance events counters per L3 complex and sixteen Data Fabric performance events counters mapped to the RDPMC instruction as follows:

- The RDPMC[5:0] instruction accesses core events. See 1.4 [Core Performance Monitor Counters].
- The RDPMC[9:6, 1B:10] instruction accesses data fabric events.
- The RDPMC[F:A] instruction accesses L3 cache events. See 1.5 [L3 Cache Performance Monitor Counters].

1.2 Performance Measurement

This section contains AMD's recommended method for collecting microarchitecture performance common to software optimization. This may require combining multiple performance event selections. Table 1 [Guidance for Common Performance Statistics with Complex Event Selects] lists formulas for collecting common performance statistics.

- The term Event is the full value written to Core::X86::Msr::PERF_CTL0..5.
    - Core PMC select bits [63:36,31:16] are at the user's discretion, (i.e., they are not part of the event selection).
- The term L3Event is the full value written to Core::X86::Msr::ChL3PmcCfg.
- The term DFEvent is the full value written to Core::X86::Msr::DF_PERF_CTL.

Some UnitMask fields are not disclosed, but may be used by 1.2 [Performance Measurement].

Table 1: Guidance for Common Performance Statistics with Complex Event Selects

Description	Equation
Branch Prediction	
Execution-Time Branch Misprediction Ratio (Non-Speculative).	Event[0x4300C3] / Event[0x4300C2]
Basic Caching	
All Data Cache Accesses	Event[0x430729]
All L2 Cache Accesses	Event[0x43F160] + Event[0x431F70] + Event[0x431F71] + Event[0x431F72]
L2 Cache Access from L1 Instruction Cache Miss (including prefetch)	Event[0x431060]
L2 Cache Access from L1 Data Cache Miss (including Prefetch)	Event[0x43E060]
L2 Cache Access from L2 Cache HWPF	Event[0x431F70] + Event[0x431F71] + Event[0x431F72]
All L2 Cache Misses	Event[0x430964] + Event[0x431F71] + Event[0x431F72]
L2 Cache Miss from L1 Instruction Cache Miss	Event[0x430164]
L2 Cache Miss from L1 Data Cache Miss	Event[0x430864]
L2 Cache Miss from L2 Cache HWPF	Event[0x431F71] + Event[0x431F72]
All L2 Cache Hits	Event[0x43f664] + Event[0x431f70]
L2 Cache Hit from L1 Instruction Cache Miss	Event[0x430664]
L2 Cache Hit from L1 Data Cache Miss	Event[0x43F064]
L2 Cache Hit from L2 Cache HWPF	Event[0x431F70]
L3 Cache Accesses	L3Event[0x0300C0000040FF04]
L3 Miss (includes cacheline state change requests)	L3Event[0x0300C00000400104]

Average L3 Cache Read Miss Latency (in core clocks)	L3Event[0x0303C00000403FAC]*10/ L3Event[0x0303C00000403FAD]
Op Cache (64B) Fetch Miss Ratio	Event[0x20043048F] / Event[0x20043078F]
Instruction Cache (32B) Fetch Miss Ratio	Event[0x10043188E] / Event[0x100431F8E]
Advanced Caching	
L1 Data Cache Fills from DRAM or IO in any NUMA node	Event[0x434844]
L1 Data Cache Fills from a different NUMA node	Event[0x435044]
L1 Data Cache Fills from within the same CCX	Event[0x430344]
L1 Data Cache Fills from another CCX cache in any NUMA node	Event[0x431444]
L1 Data Cache Fills All	Event[0x435F44]
Demand L1 Data Cache Fills from local L2	Event[0x430143]
Demand L1 Data Cache Fills from local L3 or different L2 in same CCX	Event[0x430243]
Demand L1 Data Cache Fills from another CCX cache in the same NUMA node	Event[0x430443]
Demand L1 Data Cache Fills from DRAM or MMIO in the same NUMA node	Event[0x430843]
Demand L1 Data Cache Fills from another CCX cache in a different NUMA node	Event[0x431043]
Demand L1 Data Cache Fills from Remote Memory or IO	Event[0x434043]
64B lines written per WCB close	Event[0x430150] / Event[0x432063]
TLBs	
L1 ITLB Misses	Event[0x430084] + Event[0x430785]
L2 ITLB Misses & Instruction page walk	Event[0x430785]
L1 DTLB Misses	Event[0x43FF45]
L2 DTLB Misses & Data page walk	Event[0x43F045]
All TLBs Flushed	Event[0x43FF78]
Stalls	
Macro-ops Dispatched	Event[0x4307AA]
Mixed SSE/AVX Stalls	Event[0x430E0E]
Macro-ops Retired	Event[0x4300C1]

Table 2: Guidance for Pipeline Utilization Analysis Statistics

Name	Description	Equation
	Level 1	
Total Dispatch Slots	Up to 6 instructions can be dispatched in one cycle.	8 * Event[430076]
Frontend Bound	Fraction of dispatch slots that remained unused because the frontend did not supply enough instructions/ops.	Event[1004301A0] / Total Dispatch Slots
Bad Speculation	Fraction of dispatched ops that did not retire.	(Event[4307AA] – Event[4300C1]) / Total Dispatch Slots
Backend Bound	Fraction of dispatch slots that remained unused because of backend stalls.	Event[100431EA0] / Total Dispatch Slots
SMT contention	Fraction of unused dispatch slots because the other thread was selected.	Event[1004360A0] / Total Dispatch Slots

Retiring	Fraction of dispatch slots used by ops that retired.	Event[4300C1] / Total Dispatch Slots
	Level 2	
Frontend Bound - Latency	Fraction of dispatch slots that remained unused because of a latency bottleneck in the frontend, such as Instruction Cache or ITLB misses.	8 * Event[1064301A0] / Total Dispatch Slots
Frontend Bound - BW	Fraction of dispatch slots that remained unused because of a bandwidth bottleneck in the frontend, such as decode bandwidth or Op Cache fetch bandwidth.	Event[1004301A0] – (8 * Event[1064301A0]) / Total Dispatch Slots
Bad Speculation – Mispredicts	Fraction of dispatched ops that were flushed due to branch mispredicts.	Bad Speculation * Event[4300C3] / (Event[4300C3] + Event[43019F])
Bad Speculation - Pipeline Restarts	Fraction of dispatched ops that were flushed due to pipeline restarts (resyncs).	Bad Speculation * Event[43019F]) / (Event[4300C3] + Event[430796]))
Backend Bound - Memory	Fraction of dispatched slots that remained unused because of stalls due to the memory subsystem.	Backend Bound * (Event[43A2D6] / Event[4302D6])
Backend Bound – CPU	Fraction of dispatched slots that remained unused because of stalls not related to the memory subsystem.	Backend Bound * (1 - (Event[43A2D6] / Event[4302D6]))
Retiring - Fastpath	Fraction of dispatch slots used by fastpath ops that retired.	Retiring * (Event[4300C1] – Event[1004300C2]) / Event[4300C1]
Retiring - Microcode	Fraction of dispatch slots used by microcode ops that retired.	Retiring * Event[1004300C2] / Event[4300C1]

1.3 Large Increment per Cycle Events

Table 3: PMC_Definitions

Term	Description
MergeEvent	A PMC event that is capable of counter increments greater than 15, thus requiring merging a pair of even/odd performance monitors.

The maximum increment for a regular performance event is 15 (i.e., a 4-bit event). However some event types can have a larger increments every cycle.

An option is provided for merging a pair of even/odd performance monitors to acquire an accurate count. First the odd numbered Core::X86::Msr::PERF_CTL0..5 is programmed with the event Core::X86::Pmc::Core::Merge (PMCxFFF) with the enable bit (En) turned on and with the remaining bits off. Then the corresponding even numbered Core::X86::Msr::PERF_CTL0..5 is programmed with the desired PMC event. Both the odd and even numbered counter need to be enabled in Core::X86::Msr::PerfCntrGlobalCtl for the merged counter to count. The performance monitor combines the count value to an 8-bit increment event and extends the counter to a 64-bit counter.

Software wanting to preload a value to a merged counter pair writes the high-order 16-bit value to the low-order 16 bits of the odd counter and then writes the low-order 48-bit value to the even counter. Reading the even counter of the merged

counter pair returns the full 64-bit value.

If an even performance monitor is programmed with the event Core::X86::Pmc::Core::Merge the Read results are undetermined. If an even performance monitor is programmed with a non-merge-able event (i.e., a 4-bit event) while the corresponding odd performance monitor is programmed as Merge, the Read results are undetermined. When discontinuing use of a merged counter pair, clear the Merge event from the odd performance monitor.

1.4 Core Performance Monitor Counters

This section provides the core performance counter events that may be selected through Core::X86::Msr::PERF_CTL0[EventSelect[11:8],EventSelect[7:0],UnitMask]. See Core::X86::Msr::PERF_CTR. See Core::X86::Msr::PERF_LEGACY_CTL0..3 and Core::X86::Msr::PERF_LEGACY_CTR.

1.4.1 Floating-Point (FP) Events

PMCx002 [FP retired x87 uops] (Core::X86::Pmc::Core::Retired_x87_FP_Ops)

Read-write.

Number of retired x87 arithmetic operations. Can be used to calculate x87 FLOPs.

PMCx002; PMC=0000_0000h

Bits	Description
7:3	Reserved.
2	**DivSqrROps**. Read-write. x87 Divide or square root uops.
1	**MulOps**. Read-write. x87 Multiply uops.
0	**AddSubOps**. Read-write. x87 Add/subtract uops.

PMCx003 [FP retired SSE and AVX FLOPs] (Core::X86::Pmc::Core::Retired_SSE_AVX_FLOPs)

Read-write.

Number of SSE and AVX floating point arithmetic operations retired. Number of arithmetic operations retired is dependent on number of uops retired, data size (scalar/128/256/512), data type (BF16/FP16/FP32/FP64) and type of operation (add/sub/mul/mac/...). Use MergeEvent feature for accurate results.

PMCx003; PMC=0000_0000h

Bits	Description
7:5	**FlopTypeSel**. Read-write. Mask for specifying FLOP type. **ValidValues**: **Value** / **Description** 0h: All types. 1h: B Float 16. 2h: Scalar single. 3h: Packed single. 4h: Scalar double. 5h: Packed double. 7h-6h: Reserved.
4	Reserved.
3	**MacFLOPs**. Read-write. Each MAC operation count as 2 FLOPs. bfloat MAC operations are not included in this event.
2	**DivFLOPs**. Read-write. Divide/square root FLOPs. Does not provide a useful count without use of the MergeEvent feature.
1	**MultFLOPs**. Read-write. Multiply FLOPs. Does not provide a useful count without use of the MergeEvent feature.
0	**AddSubFLOPs**. Read-write. Add/subtract FLOPs. Does not provide a useful count without use of the MergeEvent feature.

PMCx008 [FP uops retired by size] (Core::X86::Pmc::Core::Retired_FP_uOps)

Read-write.

Report number of FP uops retired by size. Can be used to determine how vectorized code is and how much MMX / x87 content is in the code.

PMCx008; PMC=0000_0000h

Bits	Description
7:6	Reserved.
5	**Pack512uOpsRetired**. Read-write. Packed 512-bit uops retired.
4	**Pack256uOpsRetired**. Read-write. Packed 256-bit uops retired.
3	**Pack128uOpsRetired**. Read-write. Packed 128-bit uops retired.
2	**ScalaruOpsRetired**. Read-write. Scalar uops retired.
1	**MMXuOpsRetired**. Read-write. MMX uops retired.
0	**x87uOpsRetired**. Read-write. x87 uops retired.

PMCx00A [FP uops retired sorted by vector or scalar] (Core::X86::Pmc::Core::FP_Ops_Retired)

Read-write.

Number of FP uops retired of selected type sorted by vector (AVX/SSE packed) or scalar (x87, AVX/SSE scalar). Can be used to profile FP codes.

PMCx00A; PMC=0000_0000h

Bits	Description
7:4	**VectorFpOpType**. Read-write. select a vector FP uop type to count or 0 for none. **ValidValues**:

Value	Description
0h	None selected.
1h	Add.
2h	Subtract.
3h	Multiply.
4h	Multiply accumulate.
5h	Divide.
6h	Square root.
7h	Compare.
8h	Convert.
9h	Blend.
Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.
Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily thought to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.
Ch	BFloat.
Dh	Logical.
Eh	Other uops not included in previous groups.
Fh	Select all fp type uops.

Bits	Description
3:0	**ScalarFpOpType**. Read-write. select scalar FP uop type to count or 0 for none. **ValidValues**:

Value	Description
0h	None selected.
1h	Add.
2h	Subtract.
3h	Multiply.
4h	Multiply accumulate.
5h	Divide.
6h	Square root.
7h	Compare.
8h	Convert.
9h	Blend.
Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.
Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily thought to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.
Ch	BFloat.
Dh	Logical.
Eh	Other uops not included in previous groups.
Fh	Select all fp type uops.

PMCx00B [FP executed integer type uops sorted by vector or scalar] (Core::X86::Pmc::Core::INT_Ops_Retired)

Read-write.

Number of integer uops executed in the FP retired of selected type sorted by vector (SSE/AVX) or scalar (MMX). Can be used to profile vector INT / MMX codes.

PMCx00B; PMC=0000_0000h

Bits	Description
7:4	**SseAvxOpType**. Read-write. select SSE/AVX vector INT uop type to count or 0 for none.
	ValidValues:

Value	Description
0h	None selected.
1h	Add.
2h	Subtract.
3h	Multiply.
4h	Multiply accumulate.
5h	AES.
6h	SHA.
7h	Compare.
8h	Convert or pack.
9h	Shift or rotate.
Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.
Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily though to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.
Ch	VNNI.
Dh	Logical.
Eh	Other uops not included in previous groups.
Fh	Select all int type uops.

Bits	Description
3:0	**MmxOpType**. Read-write. select MMX INT scalar uop type to count or 0 for none.
	ValidValues:

Value	Description
0h	None selected.
1h	Add.
2h	Subtract.
3h	Multiply.
4h	Multiply accumulate.
5h	AES.
6h	SHA.
7h	Compare.
8h	Convert or pack.
9h	Shift or rotate.
Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.
Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily though to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.
Ch	VNNI.
Dh	Logical.
Eh	Other uops not included in previous groups.
Fh	Select all int type uops.

PMCx00C [FP uops retired sorted by packed 128 or packed 256] (Core::X86::Pmc::Core::Packed_FP_Ops_Retired)

Read-write.

Number of FP uops retired of selected type sorted by 128-bit packed dest (XMM) or 256-bit packed dest (YMM). Can be used to profile FP codes.

PMCx00C; PMC=0000_0000h

Bits	Description
7:4	**Fp256OpType**. Read-write. select a 256-bit packed FP uop type to count or 0 for none.

ValidValues:

Value	Description
0h	None selected.
1h	Add.
2h	Subtract.
3h	Multiply.
4h	Multiply accumulate.
5h	Divide.
6h	Square root.
7h	Compare.
8h	Convert.
9h	Blend.
Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.
Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily thought to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.
Ch	BFloat.
Dh	Logical.
Eh	Other uops not included in previous groups.
Fh	Select all fp type uops.

Bits	Description
3:0	**Fp128OpType**. Read-write. select 128-bit packed FP uop type to count or 0 for none.

ValidValues:

Value	Description
0h	None selected.
1h	Add.
2h	Subtract.
3h	Multiply.
4h	Multiply accumulate.
5h	Divide.
6h	Square root.
7h	Compare.
8h	Convert.
9h	Blend.
Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.
Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily thought to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.
Ch	BFloat.
Dh	Logical.
Eh	Other uops not included in previous groups.
Fh	Select all fp type uops.

PMCx00D [FP executed packed integer uops sorted by packed 128 or packed 256] (Core::X86::Pmc::Core::Packed_INT_Ops_Retired)

Read-write.

Number of integer uops executed in FP retired of selected type sorted by 128-bit packed dest (XMM) or 256-bit packed dest (YMM). Can be used to profile FP codes.

PMCx00D; PMC=0000_0000h

Bits	Description
7:4	**Int256OpType**. Read-write. select a 256-bit packed INT uop type to count or 0 for none.

ValidValues:

Value	Description
0h	None selected.
1h	Add.
2h	Subtract.
3h	Multiply.
4h	Multiply accumulate.
5h	AES.
6h	SHA.
7h	Compare.
8h	Convert or pack.
9h	Shift or rotate.
Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.
Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily though to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.
Ch	VNNI.
Dh	Logical.
Eh	Other uops not included in previous groups.
Fh	Select all int type uops.

Bits	Description
3:0	**Int128OpType**. Read-write. select 128-bit packed INT uop type to count or 0 for none.

ValidValues:

Value	Description
0h	None selected.
1h	Add.
2h	Subtract.
3h	Multiply.
4h	Multiply accumulate.
5h	AES.
6h	SHA.
7h	Compare.
8h	Convert or pack.
9h	Shift or rotate.
Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.
Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily though to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.
Ch	VNNI.
Dh	Logical.
Eh	Other uops not included in previous groups.
Fh	Select all int type uops.

PMCx00E [FP Dispatch Faults] (Core::X86::Pmc::Core::FP_Dispatch_Faults)

Read-write.

Number of FP dispatch faults triggered by type. Dispatch fill/spill faults occur when FP either does not have the data needed to operate on in its local registers (fill), or FP needs to empty out upper register data for proper SSE merging behavior when executing AVX code (spill).

PMCx00E; PMC=0000_0000h

Bits	Description
7:4	Reserved.
3	**YmmSpillFault**. Read-write. YMM spill fault
2	**YmmFillFault**. Read-write. YMM fill fault
1	**XmmFillFault**. Read-write. XMM Fill fault
0	**x87FillFault**. Read-write. x87 Fill fault

1.4.2 Load/Store (LS) Events

PMCx024 [Bad Status 2] (Core::X86::Pmc::Core::Bad_Status_2_STLI)

Read-write.

Store To Load Interlock (STLI) are loads that were unable to complete because of a possible match with an older store, and the older store could not do Store To Load Forwarding (STLF) for some reason.

PMCx024; PMC=0000_0000h

Bits	Description
7:2	Reserved.
1	**StliOther**. Read-write. Store-to-load conflicts: A load was unable to complete due to a non-forwardable conflict with an older store. Most commonly, a load's address range partially but not completely overlaps with an uncompleted older store. Software can avoid this problem by using same-size and same-alignment loads and stores when accessing the same data. Vector/SIMD code is particularly susceptible to this problem; software should construct wide vector stores by manipulating vector elements in registers using shuffle/blend/swap instructions prior to storing to memory, instead of using narrow element-by-element stores.
0	Reserved.

PMCx025 [Retired Lock Instructions] (Core::X86::Pmc::Core::Retired_Lock_Instructions)

Read-write.

Counts retired atomic read-modify-write instructions with a LOCK prefix.

PMCx025; PMC=0000_0000h

Bits	Description
7:5	Reserved.
4:0	**LockInstructions**. Read-write. Specifies type of lock instructions counted **ValidValues**:

Value	Description
00h	Reserved.
01h	BusLock: Non-cacheable or cacheline-misaligned lock.
1Eh-02h	Reserved.
1Fh	AnyLock: Counts all lock instructions.

PMCx026 [Retired CLFLUSH Instructions] (Core::X86::Pmc::Core::CLFLUSH)

Read-write.

The number of retired CLFLUSH instructions. This is a non-speculative event.

PMCx026; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx027 [Retired CPUID Instructions] (Core::X86::Pmc::Core::CPUID)

Read-write.

The number of CPUID instructions retired.

PMCx027; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx029 [LS Dispatch] (Core::X86::Pmc::Core::LS_Dispatch)

Read-write.

Counts the number of operations dispatched to the LS unit. Unit Masks events are ADDed.

PMCx029; PMC=0000_0000h

Bits	Description
7:3	Reserved.
2	**LdOpSt**. Read-write. Dispatch of a single op that performs a load from and store to the same memory address.
1	**PureSt**. Read-write. Dispatch of a single op that performs a memory store.
0	**PureLd**. Read-write. Dispatch of a single op that performs a memory load.

PMCx02B [SMIs Received] (Core::X86::Pmc::Core::SMI_or_SMM_cycles)

Reset: 00h.

Counts the number of System Management Interrupts (SMIs) received.

PMCx02B; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx02C [Interrupts Taken] (Core::X86::Pmc::Core::Interrupts_Taken)

Read-write.

Counts the number of interrupts taken.

PMCx02C; PMC=0000_0000h

Bits	Description
7:1	Reserved.
0	**NumInterrupts**. Read-write. Number of interrupts taken. This event is also counted when UnitMask[7:0]=0.

PMCx035 [Store to Load Forward] (Core::X86::Pmc::Core::Store_to_Load_Forward)

Read-write.

Number of STLF hits.

PMCx035; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx037 [Store Globally Visible Cancels 2] (Core::X86::Pmc::Core::Store_Globally_Visible_Cancels_2)

Read-write.

Counts reasons why a Store Coalescing Buffer (SCB) commit is canceled.

PMCx037; PMC=0000_0000h

Bits	Description
7:1	Reserved.
0	**OlderStVisibleDepCancel**. Read-write. Older SCB we are waiting on to become globally visible was unable to become globally visible.

PMCx041 [LS MAB Allocates by Type] (Core::X86::Pmc::Core::LS_MAB_Allocates_by_Type)

Read-write.

Counts when an LS pipe allocates a Miss Address Buffer (MAB) entry to make a miss request.

PMCx041; PMC=0000_0000h

Bits	Description
7	Reserved.
6:0	**LsMabAllocation**. Read-write. **ValidValues**: Value / Description 06h-00h: Reserved. 07h: Load Store Allocations 08h: Hardware Prefetcher Allocations 0Eh-09h: Reserved. 0Fh: All Allocations 7Fh-10h: Reserved.

PMCx043 [Demand Data Cache Fills by Data Source] (Core::X86::Pmc::Core::Demand_DC_Fills_by_Data_Source)

Read-write.

Counts fills into the DC that were initiated by demand ops, per data source.

PMCx043; PMC=0000_0000h

Bits	Description
7	**AlternateMemories_NearFar**. Read-write. Requests that return from Extension Memory.
6	**DramIO_Far**. Read-write. Requests that target another NUMA node and return from DRAM or MMIO.
5	Reserved.
4	**NearFarCache_Far**. Read-write. Requests that target another NUMA node and return from another CCX's cache.
3	**DramIO_Near**. Read-write. Requests that target the same NUMA node and return from DRAM or MMIO.
2	**NearFarCache_Near**. Read-write. Requests that target the same NUMA node and return from another CCX's cache.
1	**LocalCcx**. Read-write. Data returned from L3 or different L2 in the same CCX.
0	**LocalL2**. Read-write. Data returned from local L2.

PMCx044 [Any Data Cache Fills by Data Source] (Core::X86::Pmc::Core::Any_DC_Fills_by_Data_Source)

Read-write.

Counts all fills into the DC, per data source.

PMCx044; PMC=0000_0000h

Bits	Description
7	**AlternateMemories_NearFar**. Read-write. Requests that return from Extension Memory.
6	**DramIO_Far**. Read-write. Requests that target another NUMA node and return from DRAM or MMIO.
5	Reserved.
4	**NearFarCache_Far**. Read-write. Requests that target another NUMA node and return from another CCX's cache.
3	**DramIO_Near**. Read-write. Requests that target the same NUMA node and return from DRAM or MMIO.
2	**NearFarCache_Near**. Read-write. Requests that target the same NUMA node and return from another CCX's cache.
1	**LocalCcx**. Read-write. Data returned from L3 or different L2 in the same CCX.
0	**LocalL2**. Read-write. Data returned from local L2.

PMCx045 [L1 DTLB Reloads] (Core::X86::Pmc::Core::L1_DTLB_Reloads)

Read-write.

Counts L1DTLB reloads

PMCx045; PMC=0000_0000h

Bits	Description
7	**TlbReload1GL2Miss**. Read-write. DTLB reload to a 1G page that missed in the L2DTLB.
6	**TlbReload2ML2Miss**. Read-write. DTLB reload to a 2M page that missed in the L2DTLB.
5	**TlbReloadCoalescedPageMiss**. Read-write. DTLB reload to a coalesced page that missed in the L2DTLB.
4	**TlbReload4KL2Miss**. Read-write. DTLB reload to a 4K page that missed in the L2DTLB.
3	**TlbReload1GL2Hit**. Read-write. DTLB reload to a 1G page that hit in the L2DTLB.
2	**TlbReload2ML2Hit**. Read-write. DTLB reload to a 2M page that hit in the L2DTLB.
1	**TlbReloadCoalescedPageHit**. Read-write. DTLB reload to a coalesced page that hit in the L2DTLB.
0	**TlbReload4KL2Hit**. Read-write. DTLB reload to a 4K page that hit in the L2DTLB.

PMCx047 [Misaligned Load Flows] (Core::X86::Pmc::Core::Misaligned_Load_Flows)

Read-write.

The number of misaligned load flows.

PMCx047; PMC=0000_0000h

Bits	Description
7:2	Reserved.
1	**MA4K**. Read-write. The number of 4KB misaligned (i.e., page crossing) loads or LdOpSt.
0	**MA64**. Read-write. The number of 64B misaligned (i.e., cacheline crossing) loads or LdOpSt.

PMCx04B [Prefetch Instructions Dispatched] (Core::X86::Pmc::Core::Software_Prefetch_Dispatched)

Read-write.

Software Prefetch Instructions Dispatched (speculative)

PMCx04B; PMC=0000_0000h

Bits	Description
7:3	Reserved.
2	**PREFETCHNTA**. Read-write. PrefetchNTA instruction. See docAPM3 PREFETCHlevel.
1	**PREFETCHW**. Read-write. PrefetchW instruction. See docAPM3 PREFETCHlevel.
0	**PREFETCH**. Read-write. PrefetchT0, T1, and T2 instructions. See docAPM3 PREFETCHlevel.

PMCx050 [Write Combining Buffer Close] (Core::X86::Pmc::Core::WCB_Close)

Read-write.

Counts events that cause a Write Combining Buffer (WCB) entry to close.

PMCx050; PMC=0000_0000h

Bits	Description
7:1	Reserved.
0	**FullLine64B**. Read-write. All 64 bytes of the WCB entry have been written.

PMCx052 [Ineffective Software Prefetches] (Core::X86::Pmc::Core::Ineffective_Software_Prefetches)

Read-write.

The number of software prefetches that did not fetch data outside of the processor core.

PMCx052; PMC=0000_0000h

Bits	Description
7:2	Reserved.
1	**MabHit**. Read-write. Software PREFETCH instruction saw a match on an already-allocated miss request.
0	**DcHit**. Read-write. Software PREFETCH instruction saw a DC hit.

PMCx059 [Software Prefetch Data Cache Fills by Data Source] (Core::X86::Pmc::Core::Software_Prefetch_Data_Cache_Fills)

Read-write.

Counts fills into the DC that were initiated by software prefetch instructions, per data source.

PMCx059; PMC=0000_0000h

Bits	Description
7	**AlternateMemories_NearFar**. Read-write. Requests that return from Extension Memory.
6	**DramIO_Far**. Read-write. Requests that target another NUMA node and return from DRAM or MMIO.
5	Reserved.
4	**NearFarCache_Far**. Read-write. Requests that target another NUMA node and return from another CCX's cache.
3	**DramIO_Near**. Read-write. Requests that target the same NUMA node and return from DRAM or MMIO.
2	**NearFarCache_Near**. Read-write. Requests that target the same NUMA node and return from another CCX's cache.
1	**LocalCcx**. Read-write. Data returned from L3 or different L2 in the same CCX.
0	**LocalL2**. Read-write. Data returned from local L2.

PMCx05A [Hardware Prefetch Data Cache Fills by Data Source] (Core::X86::Pmc::Core::Hardware_Prefetch_Data_Cache_Fills)

Read-write.

Counts fills into the DC that were initiated by hardware prefetches, per data source.

PMCx05A; PMC=0000_0000h

Bits	Description
7	**AlternateMemories_NearFar**. Read-write. Requests that return from Extension Memory.
6	**DramIO_Far**. Read-write. Requests that target another NUMA node and return from DRAM or MMIO.
5	Reserved.
4	**NearFarCache_Far**. Read-write. Requests that target another NUMA node and return from another CCX's cache.
3	**DramIO_Near**. Read-write. Requests that target the same NUMA node and return from DRAM or MMIO.
2	**NearFarCache_Near**. Read-write. Requests that target the same NUMA node and return from another CCX's cache.
1	**LocalCcx**. Read-write. Data returned from L3 or different L2 in the same CCX.
0	**LocalL2**. Read-write. Data returned from local L2.

PMCx05F [**Allocated DC misses**] **(Core::X86::Pmc::Core::Allocated_DC_misses)**

Read-write.

Counts the number of in-flight DC misses each cycle.

PMCx05F; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx076 [**Cycles Not in Halt**] **(Core::X86::Pmc::Core::Cycles_Not_in_Halt)**

Read-write.

Counts cycles when the thread is not in a HALTed state

PMCx076; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx078 [**All TLB Flushes**] **(Core::X86::Pmc::Core::TLB_Flush_Events)**

Read-write.

TLB flush events.

PMCx078; PMC=0000_0000h

Bits	Description
7:0	**All**. Read-write. All TLB Flushes **ValidValues**: Value: FEh-00h — Reserved. Value: FFh — Counts all TLB Flushes

PMCx120 [**P0 Freq Cycles not in Halt**] **(Core::X86::Pmc::Core::P0_frequency_Cycles_Not_in_Halt)**

Read-write.

Counts cycles not in Halt, at the P0 P-state frequency, regardless of the current Pstate.

PMCx120; PMC=0000_0000h

Bits	Description
7:1	Reserved.
0	**P0_frequency_Cycles_Not_in_Halt**. Read-write. Counts at the P0 frequency (same as Core::X86::Msr::MPERF) when not in Halt.

1.4.3 Instruction Cache (IC) and Branch Prediction (BP) Events

Note: All instruction cache events are speculative events unless specified otherwise.

PMCx082 [**Instruction Cache Refills From L2**] **(Core::X86::Pmc::Core::Instruction_Cache_Refills_from_L2)**

Read-write.

The number of 64 byte instruction cache lines fulfilled from the L2 cache.

PMCx082; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx083 [**Instruction Cache Refills from System**] **(Core::X86::Pmc::Core::Instruction_Cache_Refills_from_System)**

Read-write.

The number of 64 byte instruction cache line fulfilled from system memory or another cache.

PMCx083; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx084 [**L1 ITLB Miss, L2ITLB Hit**] **(Core::X86::Pmc::Core::L1_ITLB_Miss_L2_ITLB_Hit)**

Read-write.

The number of instruction fetches that miss in the L1 ITLB but hit in the L2 ITLB.

PMCx084; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx085 [**L1 ITLB Miss, L2 ITLB Miss**] **(Core::X86::Pmc::Core::ITLB_Reload_from_Page_Table_walk)**

Read-write.

The number of instruction fetches that miss in both the L1 ITLB and L2 ITLB.

PMCx085; PMC=0000_0000h

Bits	Description
7:4	Reserved.
3	**Coalesced_4k**. Read-write. Walk for >4k Coalesced page (implemented as 16k)
2	**walk_1G**. Read-write. Walk for 1G page
1	**walk_2M**. Read-write. Walk for 2M page
0	**walk_4K**. Read-write. Walk to 4k page

PMCx08B [**BP Pipe Correction or Cancel**] **(Core::X86::Pmc::Core::BP_Correct)**

Reset: 00h.

The Branch Predictor flushed its own pipeline due to internal conditions such as a second level prediction structure. Does not count the number of bubbles caused by these internal flushes.

PMCx08B; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx08E [**Variable Target Predictions**] **(Core::X86::Pmc::Core::Variable_Target_Predictions)**

Read-write.

The number of times a branch used the indirect predictor to make a prediction.

PMCx08E; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx091 [**Early Redirects**] **(Core::X86::Pmc::Core::Decoder_Overrides_Existing_Branch_Prediction_Speculative)**

Reset: 00h.

Number of times that an Early Redirect is sent to Branch Predictor. This happens when either the decoder or dispatch logic is able to detect that the Branch Predictor needs to be redirected.

PMCx091; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx094 [ITLB Instruction Fetch Hits] (Core::X86::Pmc::Core::ITLB_Hits)

Read-write.

The number of instruction fetches that hit in the L1ITLB.

PMCx094; PMC=0000_0000h

Bits	Description
7:3	Reserved.
2	**IF1G**. Read-write. L1 Instruction TLB Hit (1G page size)
1	**IF2M**. Read-write. L1 Instruction TLB Hit (2M page size)
0	**IF4K**. Read-write. L1 Instruction TLB Hit (4k or 16k coalesced page size)

PMCx09F [BP Redirects] (Core::X86::Pmc::Core::BP_redirects)

Read-write.

Counts redirects of the branch predictor. To support legacy software, counts both EX mispredict and resyncs when unit_mask[7:0] is set to 0.

PMCx09F; PMC=0000_0000h

Bits	Description
7:2	Reserved.
1	**ExRedir**. Read-write. Mispredict redirect from EX (execution-time)
0	**Resync**. Read-write. Resync redirect (Retire-time) from RT

PMCx188 [Fetch IBS events] (Core::X86::Pmc::Core::Fetch_IBS_events)

Read-write.

Counts significant Fetch IBS State transitions.

PMCx188; PMC=0000_0000h

Bits	Description
7:5	Reserved.
4	**SampleVal**. Read-write. Counts the number of valid Fetch Instruction Based Sampling (fetch IBS) samples that were collected. Each valid sample also created an IBS interrupt.
3	**SampleFiltered**. Read-write. Counts the number of Fetch IBS tagged fetches that were discarded due to IBS filtering. When a tagged fetch is discarded the Fetch IBS facility will automatically tag a new fetch.
2	**SampleDiscarded**. Read-write. Counts when the Fetch IBS facility discards an IBS tagged fetch for reasons other than IBS filtering. When a tagged fetch is discarded the Fetch IBS facility will automatically tag a new fetch.
1	**FetchTagged**. Read-write. Counts the number of fetches tagged for Fetch IBS. Not all tagged fetches create an IBS interrupt and valid fetch sample.
0	Reserved.

PMCx18E [IC Tag Hit and Miss Events] (Core::X86::Pmc::Core::IC_Tag_Hit_Miss_events)

Read-write.

Counts the number of microtag and full tag events as selected by unit mask.

PMCx18E; PMC=0000_0000h

Bits	Description
7:5	Reserved.
4:0	**IcAccessTypes**. Read-write. Instruction Cache accesses.
	ValidValues:

Value	Description
06h-00h	Reserved.
07h	Instruction Cache Hit.
17h-08h	Reserved.
18h	Instruction Cache Miss.
1Eh-19h	Reserved.
1Fh	All Instruction Cache Accesses.

PMCx28F [Op Cache Hit or Miss] (Core::X86::Pmc::Core::Op_Cache_hit_miss)

Read-write.

Counts Op Cache micro-tag hit/miss events.

PMCx28F; PMC=0000_0000h

Bits	Description
7:3	Reserved.
2:0	**OpCacheAccesses**. Read-write. OpCacheAccesses
	ValidValues:

Value	Description
2h-0h	Reserved.
3h	Op Cache Hit.
4h	Op Cache Miss.
6h-5h	Reserved.
7h	All Op Cache accesses.

1.4.4 DE Events

PMCx0A9 [Op Queue Empty] (Core::X86::Pmc::Core::Dispatch_Empty)

Reset: 00h.

Cycles where the Op Queue is empty.

PMCx0A9; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx0AA [**Source of Op Dispatched From Decoder**]
(Core::X86::Pmc::Core::Source_of_Op_Dispatched_From_Decoder)

Read-write.

Counts the number of ops dispatched from the decoder classified by op source.

PMCx0AA; PMC=0000_0000h

Bits	Description
7:2	Reserved.
1	**Op_Cache**. Read-write. Count of ops dispatched from OpCache
0	**x86_decoder**. Read-write. Count of ops dispatched from x86 decoder

PMCx0AB [**Types of Ops Dispatched From Decoder**]
(Core::X86::Pmc::Core::Types_of_Ops_Dispatched_From_Decoder)

Read-write.

Counts the number of ops dispatched from the decoder classified by op type. The UnitMask value encodes which types of ops are counted.

PMCx0AB; PMC=0000_0000h

Bits	Description
7:5	Reserved.
4:0	**DispOpType**. Read-write. DispOpType. **ValidValues**:

Value	Description
03h-00h	Reserved.
04h	Any FP dispatch.
07h-05h	Reserved.
08h	Any Integer dispatch.
1Fh-09h	Reserved.

PMCx0AE [**Dynamic Tokens Dispatch Stall Cycles 1**]
(Core::X86::Pmc::Core::Dispatch_Stall_Cycles_Dynamic_Tokens_Part_1)

Read-write.

Cycles where a dispatch group is valid but does not get dispatched due to a Token Stall. UnitMask bits select the stall types included in the count.

PMCx0AE; PMC=0000_0000h

Bits	Description
7	Reserved.
6	**FPSchRsrcStall**. Read-write. FP NSQ token stall
5	Reserved.
4	**TakenBrnchBufferRsrc**. Read-write. taken branch buffer resource stall.
3	Reserved.
2	**StoreQueueRsrcStall**. Read-write. STQ Tokens unavailable
1	**LoadQueueRsrcStall**. Read-write. Load Queue Token Stall.
0	**IntPhyRegFileRsrcStall**. Read-write. Integer Physical Register File resource stall.

PMCx0AF [Dynamic Tokens Dispatch Stall Cycles 2] (Core::X86::Pmc::Core::Dispatch_Stall_Cycles_Dynamic_Tokens_Part_2)

Read-write.

Cycles where a dispatch group is valid but does not get dispatched due to a token stall. UnitMask bits select the stall types included in the count.

PMCx0AF; PMC=0000_0000h

Bits	Description
7:6	Reserved.
5	**RetQ**. Read-write. Retire queue tokens unavailable
4:3	Reserved.
2	**EX_Flush_recovery**. Read-write. Integer Execution flush recovery pending
1	**AGTokens**. Read-write. Agen tokens unavailable
0	**ALTokens**. Read-write. ALU tokens unavailable

PMCx1A0 [No_Dispatch_per_Slot] (Core::X86::Pmc::Core::No_Dispatch_per_Slot)

Read-write.

Counts the number of dispatch slots (each cycle) that remained unused for reasons selected by UnitMask.

PMCx1A0; PMC=0000_0000h

Bits	Description
7:0	**StallReason**. Read-write. **ValidValues**:

Value	Description
00h	Reserved.
01h	Counts dispatch slots left empty because the front-end did not supply ops.
1Dh-02h	Reserved.
1Eh	Counts ops unable to dispatch due to back-end stalls.
5Fh-1Fh	Reserved.
60h	Counts ops unable to dispatch because the dispatch cycle was granted to the other SMT thread.
FFh-61h	Reserved.

PMCx1A2 [Dispatch Additional Resource Stalls] (Core::X86::Pmc::Core::Additional_Resource_Stalls)

Read-write.

This PMC event counts additional resource stalls that are not captured by Dispatch_Stall_Cycle_Dynamic_Tokens_Part_1 or Dispatch_Stall_Cycles_Dynamic_Tokens_Part_2.

PMCx1A2; PMC=0000_0000h

Bits	Description
7:0	**Stall**. Read-write. **ValidValues**:

Value	Description
2Fh-00h	Reserved.
30h	Counts additional cycles dispatch is stalled due to the lack of dispatch resources.
FFh-31h	Reserved.

PMCxFFF [Merge] (Core::X86::Pmc::Core::Merge)

See 1.3 [Large Increment per Cycle Events].

PMCxFFF

Bits	Description
7:0	Reserved.

1.4.5 EX (SC) Events

PMCx0C0 [Retired Instructions] (Core::X86::Pmc::Core::Retired_Instructions)

Read-write.

The number of instructions retired.

PMCx0C0; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx0C1 [Retired Macro-Ops] (Core::X86::Pmc::Core::Retired_Macro_Ops)

Read-write.

The number of macro-ops retired.

PMCx0C1; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx0C2 [Retired Branch Instructions] (Core::X86::Pmc::Core::Retired_Branch_Instructions)

Read-write.

The number of branch instructions retired. This includes all types of architectural control flow changes, including exceptions and interrupts.

PMCx0C2; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx0C3 [Retired Branch Instructions Mispredicted.] (Core::X86::Pmc::Core::Retired_Branch_Instructions_Mispredicted)

Read-write.

The number of retired branch instructions, that were mispredicted. Note that only EX mispredicts are counted.

PMCx0C3; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx0C4 [Retired Taken Branch Instructions] (Core::X86::Pmc::Core::Retired_Taken_Branch_Instructions)

Read-write.

The number of taken branches that were retired. This includes all types of architectural control flow changes, including exceptions and interrupts.

PMCx0C4; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx0C5 [**Retired Taken Branch Instructions Mispredicted.**] **(Core::X86::Pmc::Core::Retired_Taken_Branch_Instructions_Mispredicted)**

Read-write.

The number of retired taken branch instructions that were mispredicted. Note that only EX mispredicts are counted.

PMCx0C5; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx0C6 [**Retired Far Control Transfers**] **(Core::X86::Pmc::Core::Retired_Far_Control_Transfers)**

Read-write.

The number of far control transfers retired including far call/jump/return, IRET, SYSCALL and SYSRET, plus exceptions and interrupts. Far control transfers are not subject to branch prediction.

PMCx0C6; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx0C8 [**Retired Near Return Branch Instructions**] **(Core::X86::Pmc::Core::Retired_Near_Return_Branch_Instructions)**

Read-write.

The number of near return instructions (RET [C3] or RET Iw [C2]) retired.

PMCx0C8; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx0C9 [**Retired Near Return Branch Instructions Mispredicted**] **(Core::X86::Pmc::Core::Retired_Near_Return_Branch_Instructions_Mispredicted)**

Read-write.

The number of near returns retired that were not correctly predicted by the return address predictor. Each such mispredict incurs the same penalty as a mispredicted conditional branch instruction. Note that only EX mispredicts are counted .

PMCx0C9; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx0CA [**Retired Indirect Branch Instructions Mispredicted**] **(Core::X86::Pmc::Core::Retired_Indirect_Branch_Instructions_Mispredicted)**

Read-write.

The number of indirect branches retired that were not correctly predicted. Each such mispredict incurs the same penalty as a mispredicted conditional branch instruction. Note that only EX mispredicts are counted .

PMCx0CA; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx0CB [Retired MMX FP Instructions] (Core::X86::Pmc::Core::Retired_MMX_FP_Instructions)

Read-write.

The number of MMX, SSE or x87 instructions retired. The UnitMask allows the selection of the individual classes of instructions as given in the table. Each increment represents one complete instruction. Since this event includes non-numeric instructions it is not suitable for measuring MFLOPs

PMCx0CB; PMC=0000_0000h

Bits	Description
7:3	Reserved.
2	**SSE**. Read-write. SSE instructions (SSE, SSE2, SSE3, SSSE3, SSE4A, SSE41, SSE42, AVX).
1	**MMX**. Read-write. MMX instructions
0	**X87**. Read-write. x87 instructions

PMCx0CC [Retired Indirect Branch Instructions] (Core::X86::Pmc::Core::Retired_Indirect_Branch_Instructions)

Read-write.

The number of indirect branches retired.

PMCx0CC; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx0D1 [Retired Conditional Branch Instructions] (Core::X86::Pmc::Core::Retired_Conditional_Branch_Instructions)

Read-write.

Count of conditional branch instructions that retired

PMCx0D1; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx0D3 [Div Cycles Busy count] (Core::X86::Pmc::Core::Div_Cycles_Busy_count)

Read-write.

Counts cycles when the divider is busy

PMCx0D3; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx0D4 [Div Op Count] (Core::X86::Pmc::Core::Div_Op_Count)

Read-write.

Counts number of divide ops

PMCx0D4; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx0D6 [Cycles with no retire] (Core::X86::Pmc::Core::Cycles_with_no_retire)

Read-write.

This event counts cycles when the hardware thread does not retire any ops for reasons selected by UnitMask[4:0]. UnitMask events [4:0] are mutually exclusive. If multiple reasons apply for a given cycle, the lowest numbered UnitMask event is counted.

PMCx0D6; PMC=0000_0000h

Bits	Description
7:5	Reserved.
4	**ThreadNotSelected**. Read-write. The number cycles where ops could have retired (i.e. did not fall into the sub-events [0]...[3]) but did not retire because the thread arbitration did not select the thread for retire.
3	**Other**. Read-write. The number of cycles where ops could have retired (self and older ops are complete), but were stopped from retirement for other reasons: retire breaks, traps, faults, etc.
2	Reserved.
1	**NotCompleteSelf**. Read-write. The number of cycles where the oldest retire slot did not have its completion bits set.
0	**Empty**. Read-write. The number of cycles when there were no valid ops in the retire queue. This may be caused by front-end bottlenecks or pipeline redirects.

PMCx1C1 [Retired Microcoded Instructions] (Core::X86::Pmc::Core::Retired_Microcoded_Instructions)

Read-write.

The number of retired microcoded instructions.

PMCx1C1; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx1C2 [Retired Microcode Ops] (Core::X86::Pmc::Core::Retired_Microcode_Ops)

Read-write.

The number of microcode ops that have retired.

PMCx1C2; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx1C7 [Retired Conditional Branch Instructions Mispredicted] (Core::X86::Pmc::Core::Retired_Conditional_Branch_Instructions_Mispredicted)

Read-write.

The number of retired conditional branch instructions that were not correctly predicted because of a branch direction mismatch.

PMCx1C7; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx1C8 [Retired Unconditional Branch Instructions Mispredicted] (Core::X86::Pmc::Core::Retired_Unconditional_Branch_Instructions_Mispredicted)

Read-write.

The number of retired unconditional indirect branch instructions that were mispredicted.

PMCx1C8; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx1C9 [Retired Unconditional Branch Instructions] (Core::X86::Pmc::Core::Retired_Unconditional_Branch_Instructions)

Read-write.

PMCx1C9; PMC=0000_0000h

Bits	Description
7:0	Reserved.

PMCx1CF [Tagged IBS Ops] (Core::X86::Pmc::Core::Tagged_IBS_Ops)

Read-write.

Counts Op IBS related events

PMCx1CF; PMC=0000_0000h

Bits	Description
7:3	Reserved.
2	**IbsCountRollover**. Read-write. Number of times an op could not be tagged by IBS because of a previous tagged op that has not yet signaled interrupt.
1	**IbsTaggedOpsRet**. Read-write. Number of Ops tagged by IBS that retired
0	**IbsTaggedOps**. Read-write. Number of Ops tagged by IBS

PMCx1D0 [Retired Fused Instructions] (Core::X86::Pmc::Core::Retired_fused_instructions)

Reset: 00h.

Counts retired fused instructions.

PMCx1D0; PMC=0000_0000h

Bits	Description
7:0	Reserved.

1.4.6 L2 Cache Events

PMCx060 [Requests to L2 Group1] (Core::X86::Pmc::L2::L2RequestG1)

Read-write.

All L2 Cache Requests (Breakdown 1 - Common)

PMCx060; PMC=0000_0000h

Bits	Description
7	**RdBlkL**. Read-write. Data Cache Reads (including hardware and software prefetch).
6	**RdBlkX**. Read-write. Data Cache Stores
5	**LsRdBlkC_S**. Read-write. Data Cache Shared Reads
4	**CacheableIcRead**. Read-write. Instruction Cache Reads.
3	Reserved.
2	**LsPrefetchL2Cmd**. Read-write.
1	**L2HwPf**: **L2 Prefetcher**. Read-write. All prefetches accepted by L2 pipeline, hit or miss. Types of PF and L2 hit/miss broken out in a separate perfmon event
0	**Group2**. Read-write. MiscRequests. Read-write. Various Noncacheable requests. Non-cached Data Reads, Non-cached Instruction Reads, Self-modifying code checks.

PMCx061 [Requests to L2 Group2] (Core::X86::Pmc::L2::L2RequestG2)

Read-write.

All L2 Cache Requests (Breakdown 2 - Rare).

PMCx061; PMC=0000_0000h

Bits	Description
7	Reserved.
6	**LsRdSized**. Read-write. LS sized read, coherent non-cacheable.
5	**LsRdSizedNC**. Read-write. LS sized read, non-coherent, non-cacheable.
4:0	Reserved.

PMCx063 [Write Combining Buffer Requests] (Core::X86::Pmc::L2::L2WcbReq)

Read-write.

Write Combining Buffer operations. For information on Write Combining see docAPM2 sections: Memory System, Memory Types, Buffering and Combining Memory Writes.

PMCx063; PMC=0000_0000h

Bits	Description
7:6	Reserved.
5	**WcbClose**. Read-write. Write Combining Buffer close
4:0	Reserved.

PMCx064 [Core to L2 Cacheable Request Access Status] (Core::X86::Pmc::L2::L2CacheReqStat)

Read-write.

L2 Cache Request Outcomes (not including L2 Prefetch).

PMCx064; PMC=0000_0000h

Bits	Description
7	**LsRdBlkCS**: **Data Cache Shared Read Hit in L2**. Read-write. LsRdBlkCS
6	**LsRdBlkLHitX**: **Data Cache Read Hit in L2**. Read-write. Modifiable
5	**LsRdBlkLHitS**: **Data Cache Read Hit Non-Modifiable Line in L2**. Read-write.
4	**LsRdBlkX**: **Data Cache Store Hit in L2**. Read-write.
3	**LsRdBlkC**: **Data Cache Req Miss in L2**. Read-write.
2	**IcFillHitX**: **Instruction Cache Hit Modifiable Line in L2**. Read-write. IcFillHitX
1	**IcFillHitS**: **Instruction Cache Hit Non-Modifiable Line in L2.**. Read-write.
0	**IcFillMiss**: **Instruction Cache Req Miss in L2**. Read-write. IcFillMiss

PMCx070 [L2 Prefetch Hit in L2] (Core::X86::Pmc::L2::L2PfHitL2)

Read-write.

Counts all L2 prefetches accepted by L2 pipeline which hit in the L2 cache.

PMCx070; PMC=0000_0000h

Bits	Description
7:0	**Prefetches**. Read-write. **ValidValues**:

Value	Description
1Eh-00h	Reserved.
1Fh	Counts requests generated from L2 Hardware Prefetchers.
DFh-20h	Reserved.
E0h	Counts requests generated from L1 DC Hardware Prefetchers.
FEh-E1h	Reserved.
FFh	Counts requests generated from L1 DC and L2 Hardware Prefetchers.

PMCx071 [L2 Prefetcher Hits in L3] (Core::X86::Pmc::L2::L2PfMissL2HitL3)

Read-write.

Counts all L2 prefetches accepted by the L2 pipeline which miss the L2 cache and hit the L3.

PMCx071; PMC=0000_0000h

Bits	Description
7:0	**Prefetches**. Read-write. L2Stream **ValidValues**:

Value	Description
1Eh-00h	Reserved.
1Fh	Counts requests generated from L2 Hardware Prefetchers.
DFh-20h	Reserved.
E0h	Counts requests generated from L1 DC Hardware Prefetchers.
FEh-E1h	Reserved.
FFh	Counts requests generated from L1 DC and L2 Hardware Prefetchers.

PMCx072 [L2 Prefetcher Misses in L3] (Core::X86::Pmc::L2::L2PfMissL2L3)

Read-write.

Counts all L2 prefetches accepted by the L2 pipeline which miss the L2 and the L3 caches

PMCx072; PMC=0000_0000h

Bits	Description
7:0	**Prefetches**. Read-write. L2Stream **ValidValues**:

Value	Description
1Eh-00h	Reserved.
1Fh	Counts requests generated from L2 Hardware Prefetchers.
DFh-20h	Reserved.
E0h	Counts requests generated from L1 DC Hardware Prefetchers.
FEh-E1h	Reserved.
FFh	Counts requests generated from L1 DC and L2 Hardware Prefetchers.

PMCx165 [L2 Fill Response Source] (Core::X86::Pmc::L2::L2FillRspSrc)

Read-write.

Counts fill responses based on their source. Selecting an event mask of 0xfe will count all L3 responses.
This will count all L3 responses to fill requests.
This event is similar to LS PMC 0x44

PMCx165; PMC=0000_0000h

Bits	Description
7	**AlternateMemories_NearFar**. Read-write. "Requests that return from Extension Memory"
6	**DramIO_Far**. Read-write. Requests that target another NUMA node and return from either DRAM or MMIO from another NUMA node, either from the same or different NUMA node.
5	Reserved.
4	**NearFarCache_Far**. Read-write. Requests that target another NUMA node and return from another CCX's cache.
3	**DramIO_Near**. Read-write. Requests that target the same NUMA node and return from either DRAM or MMIO from the same NUMA node.
2	**NearFarCache_Near**. Read-write. Requests that target the same NUMA node and return from another CCX's cache.
1	**LocalCcx**. Read-write. Data returned from L3 or different L2 in the same CCX.
0	Reserved.

1.5 L3 Cache Performance Monitor Counters

This section provides the core performance counter events that may be selected through Core::X86::Msr::ChL3PmcCfg.

- When in non-SMT mode, thread 0 must be selected for events that don't ignore ThreadMask.

1.5.1 L3 Cache PMC Events

L3PMCx04 [L3 tag lookup state] (Core::X86::Pmc::L3::L3LookupState)

Read-write.

All L3 Requests.

L3PMCx04; L3PMC=0000_0000h

Bits	Description
7:0	**L3LookupMask**. Read-write. L3 Request Types **ValidValues**:

Value	Description
00h	Reserved.
01h	L3 Miss
FDh-02h	Reserved.
FEh	L3 Hit
FFh	All coherent accesses to L3

L3PMCxAC [L3 XiSampledLatency] (Core::X86::Pmc::L3::L3_XiSampledLatency)

Read-write.

When used in conjunction with L3_XiSampledLatencyRequests, this PMC Event will measure the average memory latency (excluding MMIO) observed by this CCX.
Configure two PMCs with the L3_XiSampledLatency and L3_XiSampledLatencyRequests events and use the following equation to identify the observed latency.

Average Sampled Latency = L3_XiSampledLatency/L3_XiSampledLatencyRequests * 10ns

Some ChL3PmcCfg fields must be programmed as follows to ensure that these events accurately measure latency: ChL3PmcCfg[EnAllSources]=0x1.

Other ChL3PmcCfg fields can be used to filter the measured latency based on originating thread (EnAllCores, CoreID) and Data Source (UnitMask).
To measure average latency from all threads to all Data Sources, use the following configuration: ChL3PmcCfg[EnAllCores]=0x1, ChL3PmcCfg[ThreadMask]=0x3, and ChL3PmcCfg[UnitMask]=0xFF.

L3PMCxAC; L3PMC=0000_0000h

Bits	Description
7:6	Reserved.
5	**Ext_Far**. Read-write. Requests that target another NUMA node and return from Extension Memory (CXL™)
4	**Ext_Near**. Read-write. Requests that target the same NUMA node and return from Extension Memory (CXL)
3	**NearCache_FarCache_Far**. Read-write. Requests that target another NUMA node and return from another CCX's cache.
2	**NearCache_FarCache_Near**. Read-write. Requests that target the same NUMA node and return from another CCX's cache.
1	**Dram_Far**. Read-write. Requests that target another NUMA node and return from DRAM
0	**Dram_Near**. Read-write. Requests that target the same NUMA node and return from DRAM

L3PMCxAD [L3 XiSampledLatencyRequests] (Core::X86::Pmc::L3::L3_XiSampledLatencyRequests)

Read-write.

When used in conjunction with L3_XiSampledLatency, this PMC Event will measure the average memory latency (excluding MMIO) observed by this CCX.
Configure two PMCs with the L3_XiSampledLatency and L3_XiSampledLatencyRequests events and use the following equation to identify the observed latency.

Average Sampled Latency = L3_XiSampledLatency/L3_XiSampledLatencyRequests * 10ns

Some ChL3PmcCfg fields must be programmed as follows to ensure that these events accurately measure latency: ChL3PmcCfg[EnAllSources]=0x1.

Other ChL3PmcCfg fields can be used to filter the measured latency based on originating thread (EnAllCores, CoreID) and Data Source (UnitMask).
To measure average latency from all threads to all Data Sources, use the following configuration: ChL3PmcCfg[EnAllCores]=0x1, ChL3PmcCfg[ThreadMask]=0x3, and ChL3PmcCfg[UnitMask]=0xFF.

L3PMCxAD; L3PMC=0000_0000h

Bits	Description
7:6	Reserved.
5	**Ext_Far**. Read-write. Requests that target another NUMA node and return from Extension Memory (CXL)
4	**Ext_Near**. Read-write. Requests that target the same NUMA node and return from Extension Memory (CXL)
3	**NearCache_FarCache_Far**. Read-write. Requests that target another NUMA node and return from another CCX's cache.
2	**NearCache_FarCache_Near**. Read-write. Requests that target the same NUMA node and return from another CCX's cache.
1	**Dram_Far**. Read-write. Requests that target another NUMA node and return from DRAM
0	**Dram_Near**. Read-write. Requests that target the same NUMA node and return from DRAM

2 UMC Performance Monitors

The UMC provides Performance Monitor counters that software can use to count specific events that occur within the DDR channel. Each UMC provides multiple 48-bit performance counters. The UMC provides one dedicated clock counter. All UMC performance monitor events can be counted on the remaining counters.

UMC Performance Monitors support the following features:

- 48-bit MEMCLK counter
- 48-bit programmable performance counters

The UMC includes both subchannels in all events counted. The programmable counters can be configured to monitor a variety of performance parameters. The counters support masking functions.

2.1 UMC Performance Monitor Events

UMCPMCx00000000 [Counts MemClk cycles] (UMC::PMC::MEMCLK)

Read-write. Reset: 0000_0000h.

_umc0_instUMCWPHY0; UMCPMCx00000000; UMCPMC=0000_0000h

Bits	Description
31	**Enable**. Read-write. Reset: 0. Enable/Start Counter. Disabling counter will freeze.
30:8	Reserved.
7:0	**Eventselect**. Read-write. Reset: 00h. Event select: programmed to 0x0

UMCPMCx00000005 [ACTIVATE command sent. Cannot be used with SPM.] (UMC::PMC::ACTCMD)

Read-write. Reset: 0000_0000h.

_umc0_instUMCWPHY0; UMCPMCx00000005; UMCPMC=0000_0000h

Bits	Description
31	**Enable**. Read-write. Reset: 0. Enable/Start Counter. Disabling counter will freeze.
30:10	Reserved.
9:8	**RdWrMask**. Read-write. Reset: 0h.
	Description: Mask either reads or writes or None 00 - None 01 - Mask Wr 10 - Mask Rd 11 - Reserved
7:0	**Eventselect**. Read-write. Reset: 00h. Event select: programmed to 0x5

UMCPMCx00000006 [**PRECHARGE command sent (For a page conflict in DCQ) . Cannot be used with SPM.**] **(UMC::PMC::PCHGCMD)**

Read-write. Reset: 0000_0000h.

_umc0_instUMCWPHY0; UMCPMCx00000006; UMCPMC=0000_0000h

Bits	Description
31	**Enable**. Read-write. Reset: 0. Enable/Start Counter. Disabling counter will freeze.
30:10	Reserved.
9:8	**RdWrMask**. Read-write. Reset: 0h.
	Description: Mask either reads or writes or None 00 - None 01 - Mask Wr 10 - Mask Rd 11 - Reserved
7:0	**Eventselect**. Read-write. Reset: 00h. Event select: programmed to 0x6

UMCPMCx0000000A [**DDR5: CAS command sent.**] **(UMC::PMC::CASCMD)**

Read-write. Reset: 0000_0000h.

_umc0_instUMCWPHY0; UMCPMCx0000000A; UMCPMC=0000_0000h

Bits	Description
31	**Enable**. Read-write. Reset: 0. Enable/Start Counter. Disabling counter will freeze.
30:10	Reserved.
9:8	**RdWrMask**. Read-write. Reset: 0h.
	Description: Mask either reads or writes or None 00 - None 01 - Mask Wr 10 - Mask Rd 11 - Reserved
7:0	**Eventselect**. Read-write. Reset: 00h. Event select: programmed to 0xA

UMCPMCx00000014 [**Number of clocks data bus is utilized. Cannot be used with SPM.**] **(UMC::PMC::DATASLOTCLKS)**

Read-write. Reset: 0000_0000h.

_umc0_instUMCWPHY0; UMCPMCx00000014; UMCPMC=0000_0000h

Bits	Description
31	**Enable**. Read-write. Reset: 0. Enable/Start Counter. Disabling counter will freeze.
30:10	Reserved.
9:8	**RdWrMask**. Read-write. Reset: 0h.
	Description: Mask either reads or writes or None 00 - None 01 - Mask Wr 10 - Mask Rd 11 - Reserved
7:0	**Eventselect**. Read-write. Reset: 00h. Event select: programmed to 0x14

List of Namespaces

Namespace	Heading(s)
Core::X86::Pmc::Core	1.3 [Large Increment per Cycle Events] 1.4.1 [Floating-Point (FP) Events] 1.4.2 [Load/Store (LS) Events] 1.4.3 [Instruction Cache (IC) and Branch Prediction (BP) Events] 1.4.4 [DE Events] 1.4.5 [EX (SC) Events]
Core::X86::Pmc::L2	1.4.6 [L2 Cache Events]
Core::X86::Pmc::L3	1.5.1 [L3 Cache PMC Events]
UMC::PMC	2.1 [UMC Performance Monitor Events]

List of Definitions

MergeEvent: A PMC event that is capable of counter increments greater than 15, thus requiring merging a pair of even/odd performance monitors.