10 Performance Monitoring

AMD Athlon™ 64 and AMD Opteron™ Processors support the performance-monitoring features introduced in earlier processor implementations. These features allow the selection of events to be monitored, and include a set of corresponding counter registers that track the frequency of monitored events. Software tools can use these features to identify performance bottlenecks, such as sections of code that have high cache-miss rates or frequently mis-predicted branches. This information can then be used as a guide for improving or eliminating performance problems through software optimizations or hardware-design improvements.

For a general description of how to use these performance monitoring features, refer to the "Debug and Performance Resources" section in Volume 2: System Programming of the *AMD64 Architecture Programmers Manual*, order# 24593.

10.1 Performance Counters

The processor provides four 48-bit performance counters. Each counter can monitor a different event. The accuracy of the counters is not ensured. Some events are reserved. When a reserved event is selected, the results are undefined.

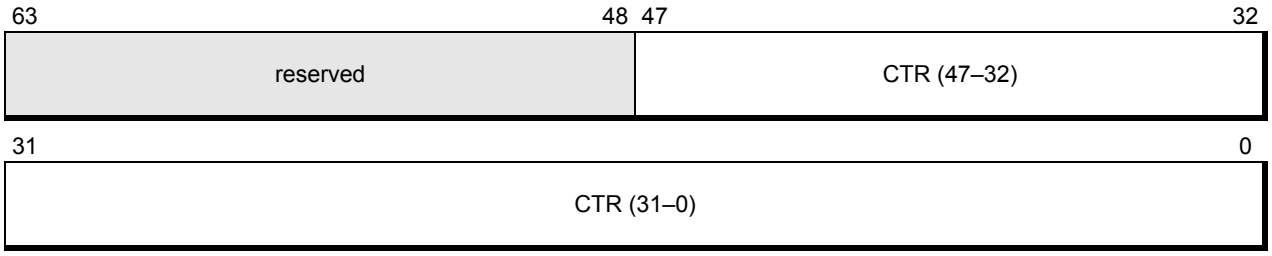
Performance counters are used to count specific processor events, such as data-cache misses, or the duration of events, such as the number of clocks it takes to return data from memory after a cache miss. During event counting, the processor increments the counter when it detects an occurrence of the event. During duration measurement, the processor counts the number of processor clocks it takes to complete an event. Each performance counter can be used to count one event, or measure the duration of one event at a time.

In addition to RDMSR instruction, the PerfCtr*n* registers can be read using a special *read performance-monitoring counter* instruction, RDPMC. The RDPMC instruction loads the contents of the PerfCtr*n* register specified by the ECX register, into the EDX register and the EAX register.

Writing the performance counters can be useful if software wants to count a specific number of events, and then trigger an interrupt when that count is reached. An interrupt can be triggered when a performance counter overflows. Software should use the WRMSR instruction to load the count as a two's-complement negative number into the performance counter. This causes the counter to overflow after counting the appropriate number of times.

The performance counters are not assured of producing identical measurements each time they are used to measure a particular instruction sequence, and they should not be used to take measurements of very small instruction sequences. The RDPMC instruction is not serializing, and it can be executed out-of-order with respect to other instructions around it. Even when bound by serializing instructions, the system environment at the time the instruction is executed can cause events to be counted before the counter value is loaded into EDX:EAX.

PerfCtr0–PerfCtr3 Registers C001_0004h, C001_0005h, C001_0006h, C001_0007h



Bit	Name	Function	R/W
63–48	Reserved	RAZ	R
47–0	CTR	Counter value	R

10.2 Performance Event-Select Registers

Performance Event-Select registers (PerfEvtSel*i*) are used to specify the events counted by the performance counters, and to control other aspects of their operation. Each performance counter supported by the implementation has a corresponding event-select register that controls its operation. This section shows the format of the PerfEvtSel*n* registers.

To accurately start counting with the write that enables the counter, disable the counter when changing the event and then enable the counter with a second MSR write.

The edge count mode will increment the counter when a transition happens on the monitored event. If the event selected is changed without disabling the counter, an extra edge will be falsely detected when the first event is a static 0 and the second event is a static one. To avoid this false edge detection, disable the counter when changing the event and then enable the counter with a second MSR write.

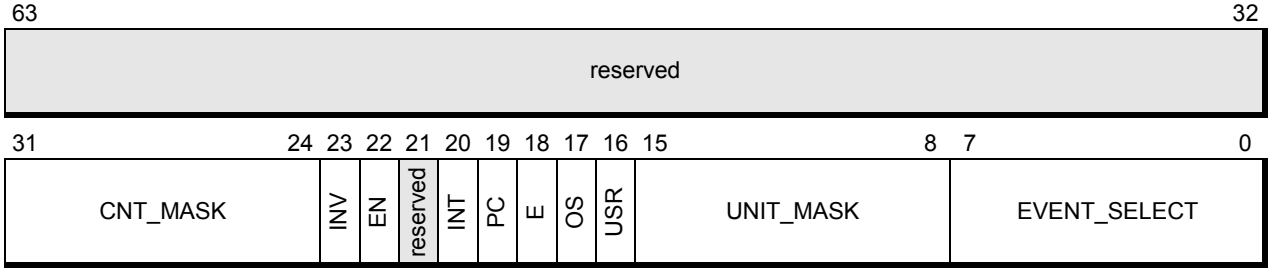
The EVENT_SELECT field specifies the event or event duration in a processor unit to be counted by the corresponding PerfCtr*i* register. The UNIT_MASK field is used to further specify or qualify a monitored event. The events that can be counted are implementation dependent. For more up-to-date information on the events in the latest processor revisions, refer to the *Revision Guide for the AMD Athlon™ 64 and AMD Opteron™ Processors,* order# 25759.

The performance counter registers can be used to track events in the Northbridge. When a Northbridge event is selected using one of the Performance Event-Select registers in either core of a dual core processor, that Performance Event-Select register and the corresponding Performance Counter register cannot be used by the other core. Monitoring of Northbridge events should only be performed by one core in a dual core processor.

Care must also be taken when measuring Northbridge or other non-processor-specific events under conditions where the processor may go into Halt mode during the measurement period. For instance, one may wish to monitor DRAM traffic due to DMA activity from a disk or graphics adaptor. This entails running some event counter monitoring code on the processor, where such code accesses the

counters at the beginning and end of the measurement period, or may even sample them periodically throughout the measurement period. Such code typically gives up the processor during each measurement interval. If there is nothing else for the OS to run on that particular processor at that time, it may halt the processor until it is needed. Under these circumstances, the clock for the counter logic will be stopped, hence the counters will not count the events of interest. To prevent this, simply run a low-priority background process that will keep the processor busy during the period of interest.

PerfEvtSel0–PerfEvtSel3 Registers — C001_0000h, C001_0001h, C001_0002h, C001_0003h



)

Bit	Name	Function	R/W
63–32	reserved	RAZ	
31–24	CNT_MASK	Counter Mask	R/W
23	INV	Invert Counter Mask	R/W
22	EN	Enable Counter	R/W
21	reserved	SBZ	
20	INT	Enable APIC Interrupt	R/W
19	PC	Pin Control	R/W
18	E	Edge Detect	R/W
17	OS	Operating-System Mode	R/W
16	USR	User Mode	R/W
15–8	UNIT_MASK	Event Qualification	R/W
7–0	EVENT_SELECT	Unit and Event Selection	R/W

Field Descriptions

Event Selection (EVENT_SELECT)—Bits 7–0. The EVENT_SELECT field specifies the event or event duration in a processor unit to be counted by the corresponding PerfCtr*i* register.

Event Qualification (UNIT_MASK)—Bits 15–8. Each UNIT_MASK bit further specifies or qualifies the event specified by EVENT_MASK. All events selected by UNIT_MASK are simultaneously monitored. Unless otherwise stated, the UNIT_MASK values shown may be combined (logically ORed) to select any desired combination of the sub-events for a given event. In some cases, certain combinations can result in misleading counts, or the UNIT_MASK value is an ordinal rather than a bit mask. These situations are described where

applicable, or should be obvious from the event descriptions. For events where no UNIT_MASK table is shown, the UNIT_MASK is not applicable and may be set to zeros.

User Mode (USR)—Bit 16. Count events in user mode (at CPL > 0).

Operating-System Mode (OS)—Bit 17. Count events in operating-system mode (at CPL = 0).

Edge Detect (E)—Bit 18. 1=Edge detection. 0=Level detection.

Pin Control (PC)—Bit 19. With a value of 1, toggles the performance monitor pin PM*i* when PerfCtr*i* register overflows. With a value of 0, toggles the performance monitor pin PM*i* each time PerfCtr*i* register is incremented.

Enable APIC Interrupt (INT)—Bit 20. Enables APIC interrupt when PerfCtr*i* register overflows.

Enable Counter (EN)—Bit 22. Enables performance monitor counter.

Invert Counter Mask (INV)—Bit 23. See CNT_MASK.

Counter Mask (CNT_MASK)—Bits 31–24. a) When CNT_MASK is 0, the corresponding PerfCtr*i* register is incremented by the number of events occurring in a clock cycle. Maximum number of events in one cycle is 3. b) When CNT_MASK values are from 1 to 3 and INV is 0, the corresponding PerfCtr*i* register is incremented by 1, if the number of events occurring in a clock cycle is greater than or equal to CNT_MASK. When CNT_MASK values are from 1 to 3 and INV is 1, the corresponding PerfCtr*i* register is incremented by 1, if the number of events occurring in a clock cycle is less than CNT_MASK. c) CNT_MASK values from 4 to 255 are reserved.

10.2.1 Performance Monitor Events

Note: ***Speculative vs. Retired****: Several events may include speculative activity, meaning they may be associated with false-path instructions that are ultimately discarded due to a branch misprediction. Events associated with Retire reflect actual program execution. For events where the distinction may matter, these are explicitly labeled as one or the other.*

10.2.1.1 Floating Point Unit Events

Event Select: 00h - Dispatched FPU Operations

The number of operations (uops) dispatched to the FPU execution pipelines. This event reflects how busy the FPU pipelines are. This includes all operations done by x87, MMX and SSE instructions, including moves. Each increment represents a one-cycle dispatch event; packed 128-bit SSE operations count as two ops; scalar operations count as one. Speculative. (See also event CBh).

Note: Since this event includes non-numeric operations it is not suitable for measuring MFLOPs.

UNIT_MASK	
01h	Add pipe ops

UNIT_MASK	
02h	Multiply pipe ops
04h	Store pipe ops
08h	Add pipe load ops
10h	Multiply pipe load ops
20h	Store pipe load ops

Event Select: 01h - Cycles with no FPU Ops Retired

The number of cycles in which no FPU operations were retired. Invert this (set the Invert control bit in the event select MSR) to count cycles in which at least one FPU operation was retired.

Event Select: 02h - Dispatched Fast Flag FPU Operations

The number of FPU operations that use the fast flag interface (e.g. FCOMI, COMISS, COMISD, UCOMISS, UCOMISD). Speculative.

10.2.1.2 Load/Store Unit and TLB Events

Event Select: 20h - Segment Register Loads

The number of segment register loads performed.

UNIT_MASK	
01h	ES
02h	CS
04h	SS
08h	DS
10h	FS
20h	GS
40h	HS

Event Select: 21h - Pipeline Restart Due to Self-Modifying Code

The number of pipeline restarts that were caused by self-modifying code (a store that hits any instruction that's been fetched for execution beyond the instruction doing the store).

Event Select: 22h - Pipeline Restart Due to Probe Hit

The number of pipeline restarts caused by an invalidating probe hitting on a speculative out-of-order load.

Event Select: 23h - LS Buffer 2 Full

The number of cycles that the LS2 buffer is full. This buffer holds stores waiting to retire as well as requests that missed the data cache and are waiting on a refill. This condition will stall further data cache accesses, although such stalls may be overlapped to some extent by independent instruction execution.

Event Select: 24h - Locked Operations

This event covers locked operations performed and their execution time. The execution time represented by the cycle counts is typically overlapped to a large extent with other instructions. The *non-speculative cycles* event is suitable for event-based profiling of lock operations that tend to miss in the cache.

UNIT_MASK	
01h	The number of locked instructions executed
02h	The number of cycles spent in speculative phase
04h	The number of cycles spent in non-speculative phase (including cache miss penalty)

Event Select: 65h - Memory Requests by Type

These events reflect accesses to uncacheable (UC) or write-combining (WC) memory regions (as defined by MTRR or PAT settings) and Streaming Store activity to WB memory. Both the WC and Streaming Store events reflect Write Combining buffer flushes, not individual store instructions. WC buffer flushes which typically consist of one 64-byte write to the system for each flush (assuming software typically fills a buffer before it gets flushed). A partially-filled buffer will require two or more smaller writes to the system. The WC event reflects flushes of WC buffers that were filled by stores to WC memory or streaming stores to WB memory. The Streaming Store event reflects only flushes due to streaming stores (which are typically only to WB memory). The difference between counts of these two events reflects the true amount of write events to WC memory.

UNIT_MASK	
01h	Requests to non-cacheable (UC) memory
02h	Requests to write-combining (WC) memory or WC buffer flushes to WB memory
80h	Streaming store (SS) requests

10.2.1.3 Data Cache Events

Event Select: 40h - Data Cache Accesses

The number of accesses to the data cache for load and store references. This may include certain microcode scratchpad accesses, although these are generally rare. Each increment represents an eight-byte access, although the instruction may only be accessing a portion of that. Speculative.

Event Select: 41h - Data Cache Misses

The number of data cache references which missed in the data cache. Speculative.

Except in the case of streaming stores, only the first miss for a given line is included - access attempts by other instructions while the refill is still pending are not included in this event. So in the absence of streaming stores, each event reflects one 64-byte cache line refill, and counts of this event are the same as, or very close to, the combined count for event 42h.

Streaming stores however will cause this event for every such store, since the target memory is not refilled into the cache. Hence this event should not be used as an indication of data cache refill activity - event 42h should be used for such measurements. (See event 65h for an indication of streaming store activity.) A large difference between events 41h (with all UNIT_MASK bits set) and 42h would be due mainly to streaming store activity.

Event Select: 42h - Data Cache Refills from L2 or System

The number of data cache refills satisfied from the L2 cache (and/or the system), per the UNIT_MASK. UNIT_MASK bits 4:1 allow a breakdown of refills from the L2 by coherency state. UNIT_MASK bit 0 reflects refills which missed in the L2, and provides the same measure as the combined sub-events of event 43h. Each increment reflects a 64-byte transfer. Speculative.

UNIT_MASK	
01h	Refill from System
02h	Shared-state line from L2
04h	Exclusive-state line from L2
08h	Owned-state line from L2
10h	Modified-state line from L2

Event Select: 43h - Data Cache Refills from System

The number of L1 cache refills satisfied from the system (system memory or another cache), as opposed to the L2. The UNIT_MASK selects lines in one or more specific coherency states. Each increment reflects a 64-byte transfer. Speculative.

UNIT_MASK	
01h	Invalid
02h	Shared
04h	Exclusive
08h	Owned
10h	Modified

Event Select: 44h - Data Cache Lines Evicted

The number of L1 data cache lines written to the L2 cache or system memory, having been displaced by L1 refills. The UNIT_MASK may be used to count only victims in specific coherency states. Each increment represents a 64-byte transfer. Speculative.

In most cases, L1 victims are moved to the L2 cache, displacing an older cache line there. Lines brought into the data cache by PrefetchNTA instructions, however, are evicted directly to system memory (if dirty) or invalidated (if clean). There is no provision for measuring this component by itself. The Invalid case (UNIT_MASK value 01h) reflects the replacement of lines that would have been invalidated by probes for write operations from another processor or DMA activity.

UNIT_MASK	
01h	Invalid
02h	Shared
04h	Exclusive
08h	Owned
10h	Modified

Event Select: 45h - L1 DTLB Miss and L2 DLTB Hit

The number of data cache accesses that miss in the L1 DTLB and hit in the L2 DTLB. Speculative.

Event Select: 46h - L1 DTLB and L2 DLTB Miss

The number of data cache accesses that miss in both the L1 and L2 DTLBs. Speculative.

Event Select: 47h - Misaligned Accesses

The number of data cache accesses that are misaligned. These are accesses which cross an eight-byte boundary. They incur an extra cache access (reflected in event 40h), and an extra cycle of latency on reads. Speculative.

Event Select: 48h - Microarchitectural Late Cancel of an Access

Event Select: 49h - Microarchitectural Early Cancel of an Access

Event Select: 4Ah - Single-bit ECC Errors Recorded by Scrubber

The number of single-bit errors corrected by either of the error detection/correction mechanisms in the data cache.

UNIT_MASK	
01h	Scrubber error
02h	Piggyback scrubber errors

Event Select: 4Bh - Prefetch Instructions Dispatched

The number of prefetch instructions dispatched by the decoder. Speculative. Such instructions may or may not cause a cache line transfer. Any Dcache and L2 accesses, hits and misses by prefetch instructions are included in those types of events.

UNIT_MASK	
01h	Load (Prefetch, PrefetchT0/T1/T2)
02h	Store (PrefetchW)
04h	NTA (PrefetchNTA)

Event Select: 4Ch - DCACHE Misses by Locked Instructions

The number of data cache misses incurred by locked instructions. (The total number of locked instructions may be obtained from event 24h.)

Such misses may be satisfied from the L2 or system memory, but there is no provision for distinguishing between the two. When used for event-based profiling, this event tends to occur very close to the offending instructions. (See also event 24h.) This event is also included in the basic Dcache miss event (41h).

UNIT_MASK	
02h	Data cache misses by locked instructions

10.2.1.4 L2 Cache and System Interface Events

Event Select: 67h - Data Prefetcher

These events reflect requests made by the data prefetcher. UNIT_MASK bit 1 counts total prefetch requests, while bit 0 counts requests where the target block is found in the L2 or data cache. The difference between the two represents actual data read (in units of 64-byte cache lines) from the system by the prefetcher. This is also included in the count of event 7Fh, UNIT_MASK bit 0 (combined with other L2 fill events).

UNIT_MASK	
01h	Cancelled prefetches
02h	Prefetch attempts

Event Select: 6Ch - System Read Responses by Coherency State

The number of responses from the system for cache refill requests. The UNIT_MASK may be used to select specific cache coherency states. Each increment represents one 64-byte cache line transferred from the system (DRAM or another cache, including another core on the same node) to the data cache, instruction cache or L2 cache (for data prefetcher and TLB table walks). Modified-state

responses may be for Dcache store miss refills, PrefetchW software prefetches, hardware prefetches for a store-miss stream, or Change-to-Dirty requests that get a dirty (Owned) probe hit in another cache. Exclusive responses may be for any Icache refill, Dcache load miss refill, other software prefetches, hardware prefetches for a load-miss stream, or TLB table walks that miss in the L2 cache. Shared responses may be for any of those that hit a clean line in another cache.

UNIT_MASK	
01h	Exclusive
02h	Modified
04h	Shared

Event Select: 6Dh - Quadwords Written to System

The number of quadword (8-byte) data transfers from the processor to the system. These may be part of a 64-byte cache line writeback or a 64-byte dirty probe hit response, each of which would cause eight increments, or a partial or complete Write Combining buffer flush (Sized Write), which could cause from one to eight increments.

UNIT_MASK	
01h	Quadword write transfer

Event Select: 7Dh - Requests to L2 Cache

The number of requests to the L2 cache for Icache or Dcache fills, or page table lookups for the TLB. These events reflect only read requests to the L2. Writes to the L2 are indicated by event 7Fh. These include some amount of retries associated with address or resource conflicts. Such retries tend to occur more as the L2 gets busier, and in certain extreme cases (such as large block moves that overflow the L2) these extra requests can dominate the event count.

These extra requests are not a direct indication of performance impact - they simply reflect opportunistic accesses that don't complete. But because of this, they are not a good indication of actual cache line movement. The Icache and Dcache miss and refill events (81h, 82h, 83h, 41h, 42h, 43h) provide a more accurate indication of this, and are the preferred way to measure such traffic.

UNIT_MASK	
01h	IC fill
02h	DC fill
04h	TLB fill (page table walks)
08h	Tag snoop request
10h	Cancelled request

Event Select: 7Eh - L2 Cache Misses

The number of requests that miss in the L2 cache. This may include some amount of speculative activity, as well as some amount of retried requests as described in event 7Dh. The IC-fill-miss and DC-fill-miss events tend to mirror the Icache and Dcache refill-from-system events (83h and 43h, respectively), and tend to include more speculative activity than those events.

UNIT_MASK	
01h	IC fill
02h	DC fill (includes possible replays, whereas event 41h does not)
04h	TLB page table walk

Event Select: 7Fh - L2 Fill/Writeback

The number of lines written into the L2 cache due to victim writebacks from the Icache or Dcache, TLB page table walks and the hardware data prefetcher (UNIT_MASK bit 0); or writebacks of dirty lines from the L2 to the system (UNIT_MASK bit 1). Each increment represents a 64-byte cache line transfer.

Note: *Victim writebacks from the Dcache may be measured separately using event 44h. However, this is not quite the same as the Dcache component of event 7Fh, the main difference being PrefetchNTA lines. When these are evicted from the Dcache due to replacement, they are written out to system memory (if dirty) or simply invalidated (if clean), rather than being moved to the L2 cache.*

UNIT_MASK	
01h	L2 fills (victims from L1 caches, TLB page table walks and data prefetches)

10.2.1.5 Instruction Cache Events

All instruction cache events are speculative.

Event Select: 80h - Instruction Cache Fetches

The number of instruction cache accesses by the instruction fetcher. Each access is an aligned 16 byte read, from which a varying number of instructions may be decoded.

Event Select: 81h - Instruction Cache Misses

The number of instruction fetches that miss in the instruction cache. This is typically equal to or very close to the sum of events 82h and 83h. Each miss results in a 64-byte cache line refill.

Event Select: 82h - Instruction Cache Refills from L2

The number of instruction cache refills satisfied from the L2 cache. Each increment represents one 64-byte cache line transfer.

Event Select: 83h - Instruction Cache Refills from System

The number of instruction cache refills from system memory (or another cache). Each increment represents one 64-byte cache line transfer.

Event Select: 84h - L1 ITLB Miss, L2 ITLB Hit

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

Event Select: 85h - L1 ITLB Miss, L2 ITLB Miss

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

Event Select: 86h - Pipeline Restart Due to Instruction Stream Probe

The number of pipeline restarts caused by invalidating probes that hit on the instruction stream currently being executed. This would happen if the active instruction stream was being modified by another processor in an MP system - typically a highly unlikely event.

Event Select: 87h - Instruction Fetch Stall

The number of cycles the instruction fetcher is stalled. This may be for a variety of reasons such as branch predictor updates, unconditional branch bubbles, far jumps and cache misses, among others. May be overlapped by instruction dispatch stalls or instruction execution, such that these stalls don't necessarily impact performance.

Event Select: 88h - Return Stack Hits

The number of near return instructions (RET or RET Iw) that get their return address from the return address stack (i.e. where the stack has not gone empty). This may include cases where the address is incorrect (return mispredicts). This may also include speculatively executed false-path returns. Return mispredicts are typically caused by the return address stack underflowing. However, they may also be caused by an imbalance in calls vs. returns, such as doing a call but then popping the return address off the stack.

Note: *This event cannot be reliably compared with events C9h and CAh (such as to calculate percentage of return mispredicts due to an empty return address stack), since it may include speculatively executed false-path returns that are not included in those retire-time events.*

Event Select: 89h - Return Stack Overflows

The number of (near) call instructions that cause the return address stack to overflow. When this happens, the oldest entry is discarded. This count may include speculatively executed calls.

10.2.1.6 Execution Unit Events

Event Select: 26h - Retired CLFLUSH Instructions

The number of CLFLUSH instructions retired.

Event Select: 27h - Retired CPUID Instructions.

The number of CPUID instructions retired.

Event Select: 76h - CPU Clocks not Halted

The number of clocks that the CPU is not in a halted state (due to STPCLK or a HALT instruction).

Note: *This event allows system idle time to be automatically factored out from IPC (or CPI) measurements, providing the OS halts the CPU when going idle. If the OS goes into an idle loop rather than halting, such calculations will be influenced by the IPC of the idle loop.*

Event Select: C0h - Retired Instructions

The number of instructions retired (execution completed and architectural state updated). This count includes exceptions and interrupts – each exception or interrupt is counted as one instruction.

Event Select: C1h - Retired uops

The number of micro-ops retired. This includes all processor activity (instructions, exceptions, interrupts, microcode assists, etc.).

Event Select: C2h - Retired Branch Instructions

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

Event Select: C3h - Retired Mispredicted Branch Instructions

The number of branch instructions retired, of any type, that were not correctly predicted. This includes those for which prediction is not attempted (far control transfers, exceptions and interrupts).

Event Select: C4h - Retired Taken Branch Instructions

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

Event Select: C5h - Retired Taken Branch Instructions Mispredicted

The number of retired taken branch instructions that were mispredicted.

Event Select: C6h - Retired Far Control Transfers

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.

Event Select: C7h - Retired Branch Resyncs

The number of resync branches. These reflect pipeline restarts due to certain microcode assists and events such as writes to the active instruction stream, among other things. Each occurrence reflects a restart penalty similar to a branch mispredict. Relatively rare.

Event Select: C8h - Retired Near Returns

The number of near return instructions (RET or RET Iw) retired.

Event Select: C9h - Retired Near Returns Mispredicted

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.

Event Select: CAh - Retired Indirect Branches Mispredicted

The number of indirect branch instructions retired where the target address was not correctly predicted.

Event Select: CBh - Retired MMX/FP Instructions

The number of MMX, SSE or X87 instructions retired. The UNIT_MASK allows the selection of the individual classes of instructions as given in the table. Each increment represents one complete instruction.

Note: *Since this event includes non-numeric instructions, it is not suitable for measuring MFLOPS.*

UNIT_MASK	
01h	x87 instructions
02h	MMX™ and 3DNow!™ instructions
04h	Packed SSE and SSE2 instructions
08h	Scalar SSE and SSE2 instructions

Event Select: CCh - Retired Fastpath Double op Instructions

UNIT_MASK	
01h	With low op in position 0
02h	With low op in position 1
04h	With low op in position 2

Event Select: CDh - Interrupts-Masked Cycles

The number of processor cycles where interrupts are masked (EFLAGS.IF = 0). Using edge-counting with this event gives the number of times IF is cleared. Dividing the cycle-count value by this value gives the average length of time that interrupts are disabled on each instance. Compare the edge count with event CFh to determine how often interrupts are disabled for interrupt handling vs. other reasons (e.g., critical sections).

Event Select: CEh - Interrupts-Masked Cycles with Interrupt Pending

The number of processor cycles where interrupts are masked (EFLAGS.IF = 0) and an interrupt is pending. Using edge-counting with this event and comparing the resulting count with the edge count for event CDh gives the proportion of interrupts for which handling is delayed due to prior interrupts being serviced, critical sections, etc. The cycle count value gives the total amount of time for such delays. The cycle count divided by the edge count gives the average length of each such delay.

Event Select: CFh - Interrupts Taken

The number of hardware interrupts taken. This does not include software interrupts (INT n instruction).

Event Select: D0h - Decoder Empty

The number of processor cycles where the decoder has nothing to dispatch (typically waiting on an instruction fetch that missed the Icache or for the target fetch after a branch mispredict).

Event Select: D1h - Dispatch Stalls

The number of processor cycles where the decoder is stalled for any reason (has one or more instructions ready but can't dispatch them due to resource limitations in execution). This is the combined effect of events D2h - DAh, some of which may overlap. This event reflects the net stall cycles. The more common stall conditions (events D5h, D6h, D7h, D8h, and to a lesser extent D2) may overlap considerably. The occurrence of these stalls is highly dependent on the nature of the code being executed (instruction mix, memory reference patterns, etc.).

Event Select: D2h - Dispatch Stall for Branch Abort to Retire

The number of processor cycles the decoder is stalled waiting for the pipe to drain after a mispredicted branch. This stall occurs if the corrected target instruction reaches the dispatch stage before the pipe has emptied. See also event D1h.

Event Select: D3h - Dispatch Stall for Serialization

The number of processor cycles the decoder is stalled due to a serializing operation, which waits for the execution pipeline to drain. Relatively rare; mainly associated with system instructions. See also event D1h.

Event Select: D4h - Dispatch Stall for Segment Load

The number of processor cycles the decoder is stalled due to a segment load instruction being encountered while execution of a previous segment load operation is still pending. Relatively rare except in 16-bit code. See also event D1h.

Event Select: D5h - Dispatch Stall for Reorder Buffer Full

The number of processor cycles the decoder is stalled because the reorder buffer is full. May occur simultaneously with certain other stall conditions. See event D1h.

Event Select: D6h - Dispatch Stall for Reservation Station Full

The number of processor cycles the decoder is stalled because a required integer unit reservation stations is full. May occur simultaneously with certain other stall conditions. See event D1h.

Event Select: D7h - Dispatch Stall for FPU Full

The number of processor cycles the decoder is stalled because the scheduler for the Floating Point Unit is full. This condition can be caused by a lack of parallelism in FP-intensive code, or by cache misses on FP operand loads (which could also show up as event D8h instead, depending on the nature of the instruction sequences). May occur simultaneously with certain other stall conditions. See event D1h

Event Select: D8h - Dispatch Stall for LS Full

The number of processor cycles the decoder is stalled because the Load/Store Unit is full. This generally occurs due to heavy cache miss activity. May occur simultaneously with certain other stall conditions. See event D1h.

Event Select: D9h - Dispatch Stall Waiting for All Quiet

The number of processor cycles the decoder is stalled waiting for all outstanding requests to the system to be resolved. Relatively rare; associated with certain system instructions and types of interrupts. May partially overlap certain other stall conditions; see event D1h.

Event Select: DAh - Dispatch Stall for Far Transfer or Resync to Retire

The number of processor cycles the decoder is stalled waiting for the execution pipeline to drain before dispatching the target instructions of a far control transfer or a Resync (an instruction stream restart associated with certain microcode assists). Relatively rare; does not overlap with other stall conditions. See also event D1h.

Event Select: DBh - FPU Exceptions

The number of floating point unit exceptions for microcode assists. The UNIT_MASK may be used to isolate specific types of exceptions.

UNIT_MASK	
01h	x87 reclass microfaults
02h	SSE retype microfaults
04h	SSE reclass microfaults
08h	SSE and x87 microtraps

Event Select: DCh - DR0 Breakpoint Matches

The number of matches on the address in breakpoint register DR0, per the breakpoint type specified in DR7. The breakpoint does not have to be enabled. Each instruction breakpoint match incurs an overhead of about 120 cycles. Load/store breakpoint matches do not incur any overhead.

Event Select: DDh - DR1 Breakpoint Matches

The number of matches on the address in breakpoint register DR1. See notes for event DCh.

Event Select: DEh - DR2 Breakpoint Matches

The number of matches on the address in breakpoint register DR2. See notes for event DCh.

Event Select: DFh - DR3 Breakpoint Matches

The number of matches on the address in breakpoint register DR3. See notes for event DCh.

10.2.1.7 Memory Controller Events

Event Select: E0h - DRAM Accesses

The number of memory accesses performed by the local DRAM controller. The UNIT_MASK may be used to isolate the different DRAM page access cases. Page miss cases incur an extra latency to open a page; page conflict cases incur both a page-close as well as page-open penalties. These penalties may be overlapped by DRAM accesses for other requests and don't necessarily represent lost DRAM bandwidth. The associated penalties are as follows:

Page miss = Trcd (DRAM RAS-to-CAS delay)

Page conflict = Trp + Trcd (DRAM row-precharge time plus RAS-to-CAS delay)

Each DRAM access represents one 64-byte block of data transferred if the DRAM is configured for 64-byte granularity, or one 32-byte block if the DRAM is configured for 32-byte granularity. (The latter is only applicable to single-channel DRAM systems, which may be configured either way.)

UNIT_MASK	
01h	Page hit
02h	Page Miss
04h	Page Conflict

Event Select: E1 - Memory Controller Page Table Overflows

The number of page table overflows in the local DRAM controller. This table maintains information about which DRAM pages are open. An overflow occurs when a request for a new page arrives when the maximum number of pages are already open. Each occurrence reflects an access latency penalty equivalent to a page conflict.

Event Select: E3 - Memory Controller Turnarounds

The number of turnarounds on the local DRAM data bus. The UNIT_MASK may be used to isolate the different cases. These represent lost DRAM bandwidth, which may be calculated as follows (in bytes per occurrence):

DIMM turnaround = DRAM_width_in_bytes * 2 edges_per_memclk * 2

R/W turnaround = DRAM_width_in_bytes * 2 edges_per_memclk * 1

R/W turnaround = DRAM_width_in_bytes * 2 edges_per_memclk * (Tcl-1)

where DRAM_width_in_bytes is 8 or 16 (for single- or dual-channel systems), and Tcl is the CAS latency of the DRAM in memory system clock cycles (where the memory clock for DDR-400, or PC3200 DIMMS, for example, would be 200 MHz).

UNIT_MASK	
01h	DIMM (chip select) turnaround
02h	Read to write turnaround
04h	Write to read turnaround

Event Select: E4h - Memory Controller Bypass Counter Saturation

UNIT_MASK	
01h	Memory controller high priority bypass
02h	Memory controller low priority bypass
04h	DRAM controller interface bypass
08h	DRAM controller queue bypass

Event Select: E5 - Sized Blocks

Sized Read/Write activity: The Sized Read/Write events reflect 32- or 64-byte transfers (as opposed to other sizes which could be anywhere between 1 and 64 bytes), from either the processor or the Hostbridge (on any node in an MP system). Such accesses from the processor would be due only to write combining buffer flushes, where 32-byte accesses would reflect flushes of partially-filled buffers. Event 65h provides a count of sized write requests associated with WC buffer flushes; comparing that with counts for these events (providing there is very little Hostbridge activity at the same time) will give an indication of how efficiently the write combining buffers are being used. Event 65h may also be useful in factoring out WC flushes when comparing these events with the Upstream Requests component of event ECh.

UNIT_MASK	
04h	32-byte Sized Writes (Revision D and later revisions)
08h	64-byte Sized Writes (Revision D and later revisions)
10h	32-byte Sized Reads (Revision D and later revisions)
20h	64-byte Sized Reads (Revision D and later revisions)

Event Select: E8h - ECC Errors

UNIT_MASK	
80h	Number of correctable and Uncorrectable DRAM ECC errors (Revision E)

Event Select: E9h - CPU/IO Requests to Memory/IO Revision E

These events reflect request flow between units and nodes, as selected by the UNIT_MASK. The UNIT_MASK is divided into two fields: request type (CPU or I/O access to I/O or Memory) and source/target location (local vs. remote). One or more requests types must be enabled with UNIT_MASK[3:0], and at least one source and one target location must be selected with UNIT_MASK[7:4]. Each event reflects a request of the selected type(s) going from the selected source(s) to the selected target(s).

Not all possible paths are supported. The following table shows the UNIT_MASK values that are valid for each request type.

Source/Target	CPU to Mem	CPU to I/O	I/O to Mem	I/O to I/O
Local -> Local	A8h	A4h	A2h	A1h
Local -> Remote	98h	94h	92h	91h
Remote -> Local	-	64h	-	61h
Remote -> Remote	-	-	-	-

Any of the mask values shown may be logically ORed to combine the events. For instance, local CPU requests to both local and remote nodes would be A8h | 98h = B8h. Any CPU to any I/O would be A4h | 94h | 64h = F4h (but remote CPU to remote I/O requests would not be included).

Note: *It is not possible to tell from these events how much data is going in which direction, as there is no distinction between reads and writes. Also, particularly for I/O, the requests may be for varying amounts of data, anywhere from one to sixty-four bytes. Event E5h provides an indication of 32- and 64-byte read and write transfers for such requests (although from the target point of view). For a direct measure of the amount and direction of data flowing between nodes, use events F6h, F7h and F8h.*

UNIT_MASK	
01h	I/O to I/O
02h	I/O to Mem
04h	CPU to I/O
08h	CPU to Mem
10h	To remote node
20h	To local node
40h	From remote node
80h	From local node

Event Select: EAh - Cache Block Commands — Revision E

The number of requests made to the system for cache line transfers or coherency state changes, by request type. Each increment represents one cache line transfer, except for Change-to-Dirty. If a Change-to-Dirty request hits on a line in another processor's cache that is in the Owned state, the

request causes a cache line transfer, otherwise no data transfer is associated with Change-to-Dirty requests.

UNIT_MASK	
01h	Victim Block (Writeback)
04h	Read Block (Dcache load miss refill)
08h	Read Block Shared (Icache refill)
10h	Read Block Modified (Dcache store miss refill)
20h	Change to Dirty (first store to clean block already in cache)

Event Select: EBh - Sized Commands

The number of Sized Read/Write commands handled by the System Request Interface (local processor and hostbridge interface to the system). These commands may originate from the processor or hostbridge. Typical uses of the various Sized Read/Write commands are given in the UNIT_MASK table. See also event E5h, which covers commonly-used block sizes for these requests, and event ECh, which provides a separate measure of Hostbridge accesses.

UNIT_MASK		Typical Usage
01h	NonPosted SzWr Byte (1-32 bytes)	Legacy or mapped I/O, typically 1–4 bytes
02h	NonPosted SzWr Dword (1-16 dwords)	Legacy or mapped I/O, typically 1 Dword
04h	Posted SzWr Byte (1-32 bytes)	Sub-cache-line DMA writes, size varies; also flushes of partially-filled Write Combining buffer
08h	Posted SzWr Dword (1-16 dwords)	Block-oriented DMA writes, often cache-line sized; also processor Write Combining buffer flushes
10h	SzRd Byte (4 bytes)	Legacy or mapped I/O
20h	SzRd Dword (1-16 dwords)	Block-oriented DMA reads, typically cache-line size
40h	RdModWr	

Event Select: ECh - Probe Responses and Upstream Requests

This covers two unrelated sets of events — cache probe results and requests received by the Hostbridge from devices on non-coherent links.

Probe results: These events reflect the results of probes sent from a memory controller to local caches. They provide an indication of the degree data and code is shared between processors (or moved between processors due to process migration). The dirty-hit events indicate the transfer of a 64-byte cache line to the requestor (for a read or cache refill) or the target memory (for a write). The system bandwidth used by these, in terms of bytes per unit of time, may be calculated as 64 times the event count, divided by the elapsed time. Sized writes to memory that cover a full cache line do not

incur this cache line transfer — they simply invalidate the line and are reported as clean hits. Cache line transfers occur for Change2Dirty requests that hit cache lines in the Owned state. (Such cache lines are counted as Modified-state refills for event 6Ch, System Read Responses.)

Upstream requests: The upstream read and write events reflect requests originating from a device on a local non-coherent HyperTransport link. The two read events allow display refresh traffic in a UMA system to be measured separately from other DMA activity. Display refresh traffic will typically be dominated by 64-byte transfers. Non-display-related DMA accesses may be anywhere from 1 to 64 bytes in size, but may be dominated by a particular size such as 32 or 64 bytes, depending on the nature of the devices. Event E5h can provide a measure of 32- and 64-byte accesses by the hostbridge (possibly combined with write combining buffer flush activity from the processor, although that can be factored out via event 65h).

UNIT_MASK	
01h	Probe miss
02h	Probe hit clean
04h	Probe hit dirty without memory cancel (probed by Sized Write or Change2Dirty)
08h	Probe hit dirty with memory cancel (probed by DMA read or cache refill request)
10h	Upstream display refresh reads
20h	Upstream non-display refresh reads
40h	Upstream writes (Revision D and later revisions)

Event Select: EEh - GART Events

These events reflect GART activity, and in particular allow one to calculate the GART TLB miss ratio as GART_miss_count divided by GART_aperture_hit_count. GART aperture accesses are typically from I/O devices as opposed to the processor and are generally from a 3D graphics accelerator, but can be from other devices when the GART is used as an IO MMU).

UNIT_MASK	
01h	GART aperture hit on access from CPU
02h	GART aperture hit on access from I/O
04h	GART miss

10.2.1.8 Crossbar Events

10.2.1.9 HyperTransport™ Interface Events

Event Select: F6h - HyperTransport Link 0 Transmit Bandwidth
F7h - HyperTransport Link 1 Transmit Bandwidth
F8h - HyperTransport Link 2 Transmit Bandwidth

The number of Dwords transmitted (or unused, in the case of Nops) on the outgoing side of the HyperTransport links. The sum of all four subevents (all four UNIT_MASK bits set) directly reflects the maximum transmission rate of the link. Link utilization may be calculated by dividing the combined Command, Data and Buffer Release count (UNIT_MASK 07h) by that value plus the Nop count (UNIT_MASK 08h). Bandwidth in terms of bytes per unit time for any one component or combination of components is calculated by multiplying the count by four and dividing by elapsed time.

The Data event provides a direct indication of the flow of data around the system. Translating this link-based view into a source/target node based view requires knowledge of the system layout (i.e. which links connect to which nodes).

UNIT_MASK	
01h	Command Dword sent
02h	Data Dword sent
04h	Buffer release Dword sent
08h	Nop Dword sent (idle)