APPENDIX A
PERFORMANCE-MONITORING EVENTS

This appendix lists the performance-monitoring events that can be monitored with the Intel 64 or IA-32 processors. The ability to monitor performance events and the events that can be monitored in these processors are mostly model-specific, except for architectural performance events, described in Section A.1.

Non-architectural performance events (i.e. model-specific events) are listed for each generation of microarchitecture:

- Section A.2 - Processors based on Intel Core microarchitecture
- Section A.3 - Intel Core Solo and Intel Core Duo processors
- Section A.4 - Processors based on Intel NetBurst microarchitecture
- Section A.5 - Pentium M family processors
- Section A.6 - P6 family processors
- Section A.7 - Pentium processors

NOTE

These performance-monitoring events are intended to be used as guides for performance tuning. The counter values reported by the performance-monitoring events are approximate and believed to be useful as relative guides for tuning software. Known discrepancies are documented where applicable.

A.1 ARCHITECTURAL PERFORMANCE-MONITORING EVENTS

Architectural performance events are introduced in Intel Core Solo and Intel Core Duo processors. They are also supported on processors based on Intel Core microarchitecture. Table A-1 lists pre-defined architectural performance events that can be configured using general-purpose performance counters and associated event-select registers.

Table A-1. Architectural Performance Events

Event Num.	Event Mask Mnemonic	Umask Value	Description	Comment
3CH	UnHalted Core Cycles	00H	Unhalted core cycles	
3CH	UnHalted Reference Cycles	01H	Unhalted reference cycles	Measures bus cycle[1]

Table A-1. Architectural Performance Events

Event Num.	Event Mask Mnemonic	Umask Value	Description	Comment
C0H	Instruction Retired	00H	Instruction retired	
2EH	LLC Reference	4FH	LL cache references	
2EH	LLC Misses	41H	LL cache misses	
C4H	Branch Instruction Retired	00H	Branch instruction retired	
C5H	Branch Misses Retired	00H	Mispredicted Branch Instruction retired	

NOTES:

1. Implementation of this event in Intel Core 2 processor family, Intel Core Duo, and Intel Core Solo processors measures bus clocks.

A.2 PERFORMANCE MONITORING EVENTS FOR INTEL® XEON® PROCESSOR 3000, 3200, 5100, 5300 SERIES AND INTEL® CORE™2 DUO PROCESSORS

Processors based on Intel Core microarchitecture support architectural and non-architectural performance-monitoring events.

Fixed-function performance counters are introduced first on processors based on Intel Core microarchitecture. Table A-2 lists pre-defined performance events that can be counted using fixed-function performance counters.

Table A-2. Fixed-Function Performance Counter and Pre-defined Performance Events

Fixed-Function Performance Counter	Address	Event Mask Mnemonic	Description
MSR_PERF_FIXED_CTR0	309H	Instr_Retired.Any	This event counts the number of instructions that retire execution. For instructions that consist of multiple micro-ops, this event counts the retirement of the last micro-op of the instruction. The counter continue counting during hardware interrupts, traps, and inside interrupt handlers

Table A-2. Fixed-Function Performance Counter and Pre-defined Performance Events (Contd.)

Fixed-Function Performance Counter	Address	Event Mask Mnemonic	Description
MSR_PERF_FIXED_CTR1	30AH	CPU_CLK_UNHALTED.CORE	This event counts the number of core cycles while the core is not in a halt state. The core enters the halt state when it is running the HLT instruction. This event is a component in many key event ratios. The core frequency may change from time to time due to transitions associated with Enhanced Intel SpeedStep Technology or TM2. For this reason this event may have a changing ratio with regards to time. When the core frequency is constant, this event can approximate elapsed time while the core was not in halt state.
MSR_PERF_FIXED_CTR2	30BH	CPU_CLK_UNHALTED.REF	This event counts the number of reference cycles when the core is not in a halt state. The core enters the halt state when it is running the HLT instruction or the MWAIT instruction. This event is not affected by core frequency changes (e.g., P states, TM2 transitions) but counts at the same frequency as the time stamp counter. This event can approximate elapsed time while the core was not in halt state. This event has a constant ratio with the CPU_CLK_UNHALTED.BUS event.

Table A-3 lists general-purpose non-architectural performance-monitoring events supported in processors based on Intel Core microarchitecture. For convenience, Table A-3 also includes architectural events and describes minor model-specific behavior where applicable. Software must use a general-purpose performance counter to count events listed in Table A-3.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture

Event Num	Umask Value	Event Name	Definition	Description and Comment
03H	02H	LOAD_BLOCK.STA	Loads blocked by a preceding store with unknown address	This event indicates that loads are blocked by preceding stores. A load is blocked when there is a preceding store to an address that is not yet calculated. The number of events is greater or equal to the number of load operations that were blocked. If the load and the store are always to different addresses, check why the memory disambiguation mechanism is not working. To avoid such blocks, increase the distance between the store and the following load so that the store address is known at the time the load is dispatched.
03H	04H	LOAD_BLOCK.STD	Loads blocked by a preceding store with unknown data	This event indicates that loads are blocked by preceding stores. A load is blocked when there is a preceding store to the same address and the stored data value is not yet known. The number of events is greater or equal to the number of load operations that were blocked. To avoid such blocks, increase the distance between the store and the dependant load, so that the store data is known at the time the load is dispatched.
03H	08H	LOAD_BLOCK. OVERLAP_STORE	Loads that partially overlap an earlier store, or 4-Kbyte aliased with a previous store	This event indicates that loads are blocked due to a variety of reasons. Some of the triggers for this event are when a load is blocked by a preceding store, in one of the following: ▪ Some of the loaded byte locations are written by the preceding store and some are not. ▪ The load is from bytes written by the preceding store, the store is aligned to its size and either: ▪ The load's data size is one or two bytes and it is not aligned to the store. ▪ The load's data size is of four or eight bytes and the load is misaligned.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
				▪ The load is from bytes written by the preceding store, the store is misaligned and the load is not aligned on the beginning of the store. ▪ The load is split over an eight byte boundary (excluding 16-byte loads). ▪ The load and store have the same offset relative to the beginning of different 4-KByte pages. This case is also called 4-KByte aliasing. In all these cases the load is blocked until after the blocking store retires and the stored data is committed to the cache hierarchy.
03H	10H	LOAD_BLOCK. UNTIL_RETIRE	Loads blocked until retirement	This event indicates that load operations were blocked until retirement. The number of events is greater or equal to the number of load operations that were blocked. This includes mainly uncacheable loads and split loads (loads that cross the cache line boundary) but may include other cases where loads are blocked until retirement.
03H	20H	LOAD_BLOCK.L1D	Loads blocked by the L1 data cache	This event indicates that loads are blocked due to one or more reasons. Some triggers for this event are: ▪ The number of L1 data cache misses exceeds the maximum number of outstanding misses supported by the processor. This includes misses generated as result of demand fetches, software prefetches or hardware prefetches. ▪ Cache line split loads. ▪ Partial reads, such as reads to un-cacheable memory, I/O instructions and more. ▪ A locked load operation is in progress. The number of events is greater or equal to the number of load operations that were blocked.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
04H	01H	SB_DRAIN_ CYCLES	Cycles while stores are blocked due to store buffer drain	This event counts every cycle during which the store buffer is draining. This includes: ▪ Serializing operations such as CPUID ▪ Synchronizing operations such as XCHG ▪ Interrupt acknowledgment ▪ Other conditions, such as cache flushing
04H	02H	STORE_BLOCK. ORDER	Cycles while store is waiting for a preceding store to be globally observed	This event counts the total duration, in number of cycles, which stores are waiting for a preceding stored cache line to be observed by other cores. This situation happens as a result of the strong store ordering behavior, as defined in "Memory Ordering," Chapter 7, *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A.* The stall may occur and be noticeable if there are many cases when a store either misses the L1 data cache or hits a cache line in the Shared state. If the store requires a bus transaction to read the cache line then the stall ends when snoop response for the bus transaction arrives.
04H	08H	STORE_BLOCK. SNOOP	A store is blocked due to a conflict with an external or internal snoop.	This event counts the number of cycles the store port was used for snooping the L1 data cache and a store was stalled by the snoop. The store is typically resubmitted one cycle later.
06H	00H	SEGMENT_REG_ LOADS	Number of segment register loads	This event counts the number of segment register load operations. Instructions that load new values into segment registers cause a penalty.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
				This event indicates performance issues in 16-bit code. If this event occurs frequently, it may be useful to calculate the number of instructions retired per segment register load. If the resulting calculation is low (on average a small number of instructions are executed between segment register loads), then the code's segment register usage should be optimized. As a result of branch misprediction, this event is speculative and may include segment register loads that do not actually occur. However, most segment register loads are internally serialized and such speculative effects are minimized.
07H	00H	SSE_PRE_EXEC. NTA	Streaming SIMD Extensions (SSE) Prefetch NTA instructions executed	This event counts the number of times the SSE instruction prefetchNTA is executed. This instruction prefetches the data to the L1 data cache.
07H	01H	SSE_PRE_EXEC.L1	Streaming SIMD Extensions (SSE) PrefetchT0 instructions executed	This event counts the number of times the SSE instruction prefetchT0 is executed. This instruction prefetches the data to the L1 data cache and L2 cache.
07H	02H	SSE_PRE_EXEC.L2	Streaming SIMD Extensions (SSE) PrefetchT1 and PrefetchT2 instructions executed	This event counts the number of times the SSE instructions prefetchT1 and prefetchT2 are executed. These instructions prefetch the data to the L2 cache.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
07H	03H	SSE_PRE_ EXEC.STORES	Streaming SIMD Extensions (SSE) Weakly-ordered store instructions executed	This event counts the number of times SSE non-temporal store instructions are executed.
08H	01H	DTLB_MISSES. ANY	Memory accesses that missed the DTLB	This event counts the number of Data Table Lookaside Buffer (DTLB) misses. The count includes misses detected as a result of speculative accesses. Typically a high count for this event indicates that the code accesses a large number of data pages.
08H	02H	DTLB_MISSES .MISS_LD	DTLB misses due to load operations	This event counts the number of Data Table Lookaside Buffer (DTLB) misses due to load operations. This count includes misses detected as a result of speculative accesses.
08H	04H	DTLB_MISSES.L0_ MISS_LD	L0 DTLB misses due to load operations	This event counts the number of level 0 Data Table Lookaside Buffer (DTLB0) misses due to load operations. This count includes misses detected as a result of speculative accesses. Loads that miss that DTLB0 and hit the DTLB1 can incur two-cycle penalty.
08H	08H	DTLB_MISSES. MISS_ST	TLB misses due to store operations	This event counts the number of Data Table Lookaside Buffer (DTLB) misses due to store operations. This count includes misses detected as a result of speculative accesses. Address translation for store operations is performed in the DTLB1.
09H	01H	MEMORY_ DISAMBIGUATION. RESET	Memory disambiguation reset cycles	This event counts the number of cycles during which memory disambiguation misprediction occurs. As a result the execution pipeline is cleaned and execution of the mispredicted load instruction and all succeeding instructions restarts.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
				This event occurs when the data address accessed by a load instruction, collides infrequently with preceding stores, but usually there is no collision. It happens rarely, and may have a penalty of about 20 cycles.
09H	02H	MEMORY_DISAMBIGUATION.SUCCESS	Number of loads successfully disambiguated.	This event counts the number of load operations that were successfully disambiguated. Loads are preceded by a store with an unknown address, but they are not blocked.
0CH	01H	PAGE_WALKS.COUNT	Number of page-walks executed	This event counts the number of page-walks executed due to either a DTLB or ITLB miss. The page walk duration, PAGE_WALKS.CYCLES, divided by number of page walks is the average duration of a page walk. The average can hint whether most of the page-walks are satisfied by the caches or cause an L2 cache miss.
0CH	02H	PAGE_WALKS.CYCLES	Duration of page-walks in core cycles	This event counts the duration of page-walks in core cycles. The paging mode in use typically affects the duration of page walks. Page walk duration divided by number of page walks is the average duration of page-walks. The average can hint at whether most of the page-walks are satisfied by the caches or cause an L2 cache miss.
10H	00H	FP_COMP_OPS_EXE	Floating point computational micro-ops executed	This event counts the number of floating point computational micro-ops executed.
11H	00H	FP_ASSIST	Floating point assists	This event counts the number of floating point operations executed that required micro-code assist intervention. Assists are required in the following cases: ▪ Streaming SIMD Extensions (SSE) instructions:

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
				▪ Denormal input when the DAZ (Denormals Are Zeros) flag is off ▪ Underflow result when the FTZ (Flush To Zero) flag is off ▪ X87 instructions: ▪ NaN or denormal are loaded to a register or used as input from memory ▪ Division by 0 ▪ Underflow output
12H	00H	MUL	Multiply operations executed	This event counts the number of multiply operations executed. This includes integer as well as floating point multiply operations.
13H	00H	DIV	Divide operations executed	This event counts the number of divide operations executed. This includes integer divides, floating point divides and square-root operations executed.
14H	00H	CYCLES_DIV _BUSY	Cycles the divider busy	This event counts the number of cycles the divider is busy executing divide or square root operations. The divide can be integer, X87 or Streaming SIMD Extensions (SSE). The square root operation can be either X87 or SSE.
18H	00H	IDLE_DURING _DIV	Cycles the divider is busy and all other execution units are idle.	This event counts the number of cycles the divider is busy (with a divide or a square root operation) and no other execution unit or load operation is in progress. Load operations are assumed to hit the L1 data cache. This event considers only micro-ops dispatched after the divider started operating.
19H	00H	DELAYED_ BYPASS.FP	Delayed bypass to FP operation	This event counts the number of times floating point operations use data immediately after the data was generated by a non-floating point execution unit. Such cases result in one penalty cycle due to data bypass between the units.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
19H	01H	DELAYED_ BYPASS.SIMD	Delayed bypass to SIMD operation	This event counts the number of times SIMD operations use data immediately after the data was generated by a non-SIMD execution unit. Such cases result in one penalty cycle due to data bypass between the units.
19H	02H	DELAYED_ BYPASS.LOAD	Delayed bypass to load operation	This event counts the number of delayed bypass penalty cycles that a load operation incurred. When load operations use data immediately after the data was generated by an integer execution unit, they may (pending on certain dynamic internal conditions) incur one penalty cycle due to delayed data bypass between the units.
21H	See Table 18-7	L2_ADS.(Core)	Cycles L2 address bus is in use	This event counts the number of cycles the L2 address bus is being used for accesses to the L2 cache or bus queue. It can count occurrences for this core or both cores.
23H	See Table 18-7	L2_DBUS_BUSY _RD.(Core)	Cycles the L2 transfers data to the core	This event counts the number of cycles during which the L2 data bus is busy transferring data from the L2 cache to the core. It counts for all L1 cache misses (data and instruction) that hit the L2 cache. This event can count occurrences for this core or both cores.
24H	Com-bined mask from Table 18-7 and Table 18-9	L2_LINES_IN. (Core, Prefetch)	L2 cache misses	This event counts the number of cache lines allocated in the L2 cache. Cache lines are allocated in the L2 cache as a result of requests from the L1 data and instruction caches and the L2 hardware prefetchers to cache lines that are missing in the L2 cache. This event can count occurrences for this core or both cores. It can also count demand requests and L2 hardware prefetch requests together or separately.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
25H	See Table 18-7	L2_M_LINES_IN. (Core)	L2 cache line modifications	This event counts whenever a modified cache line is written back from the L1 data cache to the L2 cache. This event can count occurrences for this core or both cores.
26H	See Table 18-7 and Table 18-9	L2_LINES_OUT. (Core, Prefetch)	L2 cache lines evicted	This event counts the number of L2 cache lines evicted. This event can count occurrences for this core or both cores. It can also count evictions due to demand requests and L2 hardware prefetch requests together or separately.
27H	See Table 18-7 and Table 18-9	L2_M_LINES_OUT.(Core, Prefetch)	Modified lines evicted from the L2 cache	This event counts the number of L2 modified cache lines evicted. These lines are written back to memory unless they also exist in a modified-state in one of the L1 data caches. This event can count occurrences for this core or both cores. It can also count evictions due to demand requests and L2 hardware prefetch requests together or separately.
28H	Com-bined mask from Table 18-7 and Table 18-10	L2_IFETCH.(Core, Cache Line State)	L2 cacheable instruction fetch requests	This event counts the number of instruction cache line requests from the IFU. It does not include fetch requests from uncacheable memory. It does not include ITLB miss accesses. This event can count occurrences for this core or both cores. It can also count accesses to cache lines at different MESI states.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
29H	Combined mask from Table 18-7, Table 18-9, and Table 18-10	L2_LD.(Core, Prefetch, Cache Line State)	L2 cache reads	This event counts L2 cache read requests coming from the L1 data cache and L2 prefetchers. The event can count occurrences: ▪ for this core or both cores ▪ due to demand requests and L2 hardware prefetch requests together or separately ▪ of accesses to cache lines at different MESI states
2AH	See Table 18-7 and Table 18-10	L2_ST.(Core, Cache Line State)	L2 store requests	This event counts all store operations that miss the L1 data cache and request the data from the L2 cache. The event can count occurrences for this core or both cores. It can also count accesses to cache lines at different MESI states.
2BH	See Table 18-7 and Table 18-10	L2_LOCK.(Core, Cache Line State)	L2 locked accesses	This event counts all locked accesses to cache lines that miss the L1 data cache. The event can count occurrences for this core or both cores. It can also count accesses to cache lines at different MESI states.
2EH	See Table 18-7, Table 18-9, and Table 18-10	L2_RQSTS.(Core, Prefetch, Cache Line State)	L2 cache requests	This event counts all completed L2 cache requests. This includes L1 data cache reads, writes, and locked accesses, L1 data prefetch requests, instruction fetches, and all L2 hardware prefetch requests. This event can count occurrences: ▪ for this core or both cores. ▪ due to demand requests and L2 hardware prefetch requests together, or separately ▪ of accesses to cache lines at different MESI states

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
2EH	41H	L2_RQSTS.SELF. DEMAND.I_STATE	L2 cache demand requests from this core that missed the L2	This event counts all completed L2 cache demand requests from this core that miss the L2 cache. This includes L1 data cache reads, writes, and locked accesses, L1 data prefetch requests, and instruction fetches. This is an architectural performance event.
2EH	4FH	L2_RQSTS.SELF. DEMAND.MESI	L2 cache demand requests from this core	This event counts all completed L2 cache demand requests from this core. This includes L1 data cache reads, writes, and locked accesses, L1 data prefetch requests, and instruction fetches. This is an architectural performance event.
30H	See Table 18-7, Table 18-9, and Table 18-10	L2_REJECT_BUSQ.(Core, Prefetch, Cache Line State)	Rejected L2 cache requests	This event indicates that a pending L2 cache request that requires a bus transaction is delayed from moving to the bus queue. Some of the reasons for this event are: ▪ The bus queue is full. ▪ The bus queue already holds an entry for a cache line in the same set. The number of events is greater or equal to the number of requests that were rejected. ▪ for this core or both cores. ▪ due to demand requests and L2 hardware prefetch requests together, or separately. ▪ of accesses to cache lines at different MESI states.
32H	See Table 18-7	L2_NO_REQ.(Core)	Cycles no L2 cache requests are pending	This event counts the number of cycles that no L2 cache requests were pending from a core. When using the BOTH_CORE modifier, the event counts only if none of the cores have a pending request. The event counts also when one core is halted and the other is not halted. The event can count occurrences for this core or both cores.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
3AH	00H	EIST_TRANS	Number of Enhanced Intel SpeedStep Technology (EIST) transitions	This event counts the number of transitions that include a frequency change, either with or without voltage change. This includes Enhanced Intel SpeedStep Technology (EIST) and TM2 transitions. The event is incremented only while the counting core is in C0 state. Since transitions to higher-numbered CxE states and TM2 transitions include a frequency change or voltage transition, the event is incremented accordingly.
3BH	C0H	THERMAL_TRIP	Number of thermal trips	This event counts the number of thermal trips. A thermal trip occurs whenever the processor temperature exceeds the thermal trip threshold temperature. Following a thermal trip, the processor automatically reduces frequency and voltage. The processor checks the temperature every millisecond and returns to normal when the temperature falls below the thermal trip threshold temperature.
3CH	00H	CPU_CLK_ UNHALTED. CORE_P	Core cycles when core is not halted	This event counts the number of core cycles while the core is not in a halt state. The core enters the halt state when it is running the HLT instruction. This event is a component in many key event ratios. The core frequency may change due to transitions associated with Enhanced Intel SpeedStep Technology or TM2. For this reason, this event may have a changing ratio in regard to time. When the core frequency is constant, this event can give approximate elapsed time while the core not in halt state. This is an architectural performance event.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
3CH	01H	CPU_CLK_ UNHALTED.BUS	Bus cycles when core is not halted	This event counts the number of bus cycles while the core is not in the halt state. This event can give a measurement of the elapsed time while the core was not in the halt state. The core enters the halt state when it is running the HLT instruction. The event also has a constant ratio with CPU_CLK_UNHALTED.REF event, which is the maximum bus to processor frequency ratio. Non-halted bus cycles are a component in many key event ratios.
3CH	02H	CPU_CLK_ UNHALTED.NO _OTHER	Bus cycles when core is active and the other is halted	This event counts the number of bus cycles during which the core remains non-halted and the other core on the processor is halted. This event can be used to determine the amount of parallelism exploited by an application or a system. Divide this event count by the bus frequency to determine the amount of time that only one core was in use.
40H	See Table 18-10	L1D_CACHE_LD. (Cache Line State)	L1 cacheable data reads	This event counts the number of data reads from cacheable memory. Locked reads are not counted.
41H	See Table 18-10	L1D_CACHE_ST. (Cache Line State)	L1 cacheable data writes	This event counts the number of data writes to cacheable memory. Locked writes are not counted.
42H	See Table 18-10	L1D_CACHE_ LOCK.(Cache Line State)	L1 data cacheable locked reads	This event counts the number of locked data reads from cacheable memory.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
42H	10H	L1D_CACHE_ LOCK_DURATION	Duration of L1 data cacheable locked operation	This event counts the number of cycles during which any cache line is locked by any locking instruction. Locking happens at retirement and therefore the event does not occur for instructions that are speculatively executed. Locking duration is shorter than locked instruction execution duration.
43H	10H	L1D_ALL_REF	All references to the L1 data cache	This event counts all references to the L1 data cache, including all loads and stores with any memory types. The event counts memory accesses only when they are actually performed. For example, a load blocked by unknown store address and later performed is only counted once. The event includes non-cacheable accesses, such as I/O accesses.
43H	02H	L1D_ALL_ CACHE_REF	L1 Data cacheable reads and writes	This event counts the number of data reads and writes from cacheable memory, including locked operations. This event is a sum of: ▪ L1D_CACHE_LD.MESI ▪ L1D_CACHE_ST.MESI ▪ L1D_CACHE_LOCK.MESI
45H	0FH	L1D_REPL	Cache lines allocated in the L1 data cache	This event counts the number of lines brought into the L1 data cache.
46H	00H	L1D_M_REPL	Modified cache lines allocated in the L1 data cache	This event counts the number of modified lines brought into the L1 data cache.
47H	00H	L1D_M_EVICT	Modified cache lines evicted from the L1 data cache	This event counts the number of modified lines evicted from the L1 data cache, whether due to replacement or by snoop HITM intervention.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
48H	00H	L1D_PEND_ MISS	Total number of outstanding L1 data cache misses at any cycle	This event counts the number of outstanding L1 data cache misses at any cycle. An L1 data cache miss is outstanding from the cycle on which the miss is determined until the first chunk of data is available. This event counts: ▪ all cacheable demand requests ▪ L1 data cache hardware prefetch requests ▪ requests to write through memory ▪ requests to write combine memory Uncacheable requests are not counted. The count of this event divided by the number of L1 data cache misses, L1D_REPL, is the average duration in core cycles of an L1 data cache miss.
49H	01H	L1D_SPLIT.LOADS	Cache line split loads from the L1 data cache	This event counts the number of load operations that span two cache lines. Such load operations are also called split loads. Split load operations are executed at retirement.
49H	02H	L1D_SPLIT. STORES	Cache line split stores to the L1 data cache	This event counts the number of store operations that span two cache lines.
4BH	00H	SSE_PRE_ MISS.NTA	Streaming SIMD Extensions (SSE) Prefetch NTA instructions missing all cache levels	This event counts the number of times the SSE instructions prefetchNTA were executed and missed all cache levels. Due to speculation an executed instruction might not retire. This instruction prefetches the data to the L1 data cache.
4BH	01H	SSE_PRE_ MISS.L1	Streaming SIMD Extensions (SSE) PrefetchT0 instructions missing all cache levels	This event counts the number of times the SSE instructions prefetchT0 were executed and missed all cache levels. Due to speculation executed instruction might not retire. The prefetchT0 instruction prefetches data to the L2 cache and L1 data cache.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
4BH	02H	SSE_PRE_ MISS.L2	Streaming SIMD Extensions (SSE) PrefetchT1 and PrefetchT2 instructions missing all cache levels	This event counts the number of times the SSE instructions prefetchT1 and prefetchT2 were executed and missed all cache levels. Due to speculation, an executed instruction might not retire. The prefetchT1 and PrefetchNT2 instructions prefetch data to the L2 cache.
4CH	00H	LOAD_HIT_PRE	Load operations conflicting with a software prefetch to the same address	This event counts load operations sent to the L1 data cache while a previous Streaming SIMD Extensions (SSE) prefetch instruction to the same cache line has started prefetching but has not yet finished.
4EH	10H	L1D_PREFETCH. REQUESTS	L1 data cache prefetch requests	This event counts the number of times the L1 data cache requested to prefetch a data cache line. Requests can be rejected when the L2 cache is busy and resubmitted later or lost. All requests are counted, including those that are rejected.
60H	See Table 18-7 and Table 18-8	BUS_REQUEST_ OUTSTANDING. (Core and Bus Agents)	Outstanding cacheable data read bus requests duration	This event counts the number of pending full cache line read transactions on the bus occurring in each cycle. A read transaction is pending from the cycle it is sent on the bus until the full cache line is received by the processor. The event counts only full-line cacheable read requests from either the L1 data cache or the L2 prefetchers. It does not count Read for Ownership transactions, instruction byte fetch transactions, or any other bus transaction.
61H	See Table 18-8.	BUS_BNR_DRV. (Bus Agents)	Number of Bus Not Ready signals asserted	This event counts the number of Bus Not Ready (BNR) signals that the processor asserts on the bus to suspend additional bus requests by other bus agents.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
				A bus agent asserts the BNR signal when the number of data and snoop transactions is close to the maximum that the bus can handle. To obtain the number of bus cycles during which the BNR signal is asserted, multiply the event count by two. While this signal is asserted, new transactions cannot be submitted on the bus. As a result, transaction latency may have higher impact on program performance.
62H	See Table 18-8	BUS_DRDY_ CLOCKS.(Bus Agents)	Bus cycles when data is sent on the bus	This event counts the number of bus cycles during which the DRDY (Data Ready) signal is asserted on the bus. The DRDY signal is asserted when data is sent on the bus. With the 'THIS_AGENT' mask this event counts the number of bus cycles during which this agent (the processor) writes data on the bus back to memory or to other bus agents. This includes all explicit and implicit data writebacks, as well as partial writes. With the 'ALL_AGENTS' mask, this event counts the number of bus cycles during which any bus agent sends data on the bus. This includes all data reads and writes on the bus.
63H	See Table 18-7 and Table 18-8	BUS_LOCK_ CLOCKS.(Core and Bus Agents)	Bus cycles when a LOCK signal asserted	This event counts the number of bus cycles, during which the LOCK signal is asserted on the bus. A LOCK signal is asserted when there is a locked memory access, due to: ▪ uncacheable memory ▪ locked operation that spans two cache lines ▪ page-walk from an uncacheable page table Bus locks have a very high performance penalty and it is highly recommended to avoid such accesses.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
64H	See Table 18-7	BUS_DATA_ RCV.(Core)	Bus cycles while processor receives data	This event counts the number of bus cycles during which the processor is busy receiving data.
65H	See Table 18-7 and Table 18-8	BUS_TRANS_BRD.(Core and Bus Agents)	Burst read bus transactions	This event counts the number of burst read transactions including: ▪ L1 data cache read misses (and L1 data cache hardware prefetches) ▪ L2 hardware prefetches by the DPL and L2 streamer ▪ IFU read misses of cacheable lines. It does not include RFO transactions.
66H	See Table 18-7 and Table 18-8.	BUS_TRANS_RFO.(Core and Bus Agents)	RFO bus transactions	This event counts the number of Read For Ownership (RFO) bus transactions, due to store operations that miss the L1 data cache and the L2 cache. It also counts RFO bus transactions due to locked operations.
67H	See Table 18-7 and Table 18-8.	BUS_TRANS_WB. (Core and Bus Agents)	Explicit writeback bus transactions	This event counts all explicit writeback bus transactions due to dirty line evictions. It does not count implicit writebacks due to invalidation by a snoop request.
68H	See Table 18-7 and Table 18-8	BUS_TRANS_ IFETCH.(Core and Bus Agents)	Instruction-fetch bus transactions	This event counts all instruction fetch full cache line bus transactions.
69H	See Table 18-7 and Table 18-8	BUS_TRANS_ INVAL.(Core and Bus Agents)	Invalidate bus transactions	This event counts all invalidate transactions. Invalidate transactions are generated when: ▪ A store operation hits a shared line in the L2 cache. ▪ A full cache line write misses the L2 cache or hits a shared line in the L2 cache.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
6AH	See Table 18-7 and Table 18-8	BUS_TRANS_ PWR.(Core and Bus Agents)	Partial write bus transaction	This event counts partial write bus transactions.
6BH	See Table 18-7 and Table 18-8	BUS_TRANS _P.(Core and Bus Agents)	Partial bus transactions	This event counts all (read and write) partial bus transactions.
6CH	See Table 18-7 and Table 18-8	BUS_TRANS_IO.(C ore and Bus Agents)	IO bus transactions	This event counts the number of completed I/O bus transactions as a result of IN and OUT instructions. The count does not include memory mapped IO.
6DH	See Table 18-7 and Table 18-8	BUS_TRANS_ DEF.(Core and Bus Agents)	Deferred bus transactions	This event counts the number of deferred transactions.
6EH	See Table 18-7 and Table 18-8	BUS_TRANS_ BURST.(Core and Bus Agents)	Burst (full cache-line) bus transactions	This event counts burst (full cache line) transactions including: ▪ Burst reads ▪ RFOs ▪ Explicit writebacks ▪ Write combine lines
6FH	See Table 18-7 and Table 18-8	BUS_TRANS_ MEM.(Core and Bus Agents)	Memory bus transactions	This event counts all memory bus transactions including: ▪ Burst transactions ▪ Partial reads and writes - invalidate transactions The BUS_TRANS_MEM count is the sum of BUS_TRANS_BURST, BUS_TRANS_P and BUS_TRANS_IVAL.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
70H	See Table 18-7 and Table 18-8	BUS_TRANS_ ANY.(Core and Bus Agents)	All bus transactions	This event counts all bus transactions. This includes: ▪ Memory transactions ▪ IO transactions (non memory-mapped) ▪ Deferred transaction completion ▪ Other less frequent transactions, such as interrupts
77H	See Table 18-7 and Table 18-11	EXT_SNOOP. (Bus Agents, Snoop Response)	External snoops	This event counts the snoop responses to bus transactions. Responses can be counted separately by type and by bus agent. With the 'THIS_AGENT' mask, the event counts snoop responses from this processor to bus transactions sent by this processor. With the 'ALL_AGENTS' mask the event counts all snoop responses seen on the bus.
78H	See Table 18-7 and Table 18-12	CMP_SNOOP.(Core, Snoop Type)	L1 data cache snooped by other core	This event counts the number of times the L1 data cache is snooped for a cache line that is needed by the other core in the same processor. The cache line is either missing in the L1 instruction or data caches of the other core, or is available for reading only and the other core wishes to write the cache line. The snoop operation may change the cache line state. If the other core issued a read request that hit this core in E state, typically the state changes to S state in this core. If the other core issued a read for ownership request (due a write miss or hit to S state) that hits this core's cache line in E or S state, this typically results in invalidation of the cache line in this core. If the snoop hits a line in M state, the state is changed at a later opportunity.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
				These snoops are performed through the L1 data cache store port. Therefore, frequent snoops may conflict with extensive stores to the L1 data cache, which may increase store latency and impact performance.
7AH	See Table 18-8	BUS_HIT_DRV. (Bus Agents)	HIT signal asserted	This event counts the number of bus cycles during which the processor drives the HIT# pin to signal HIT snoop response.
7BH	See Table 18-8	BUS_HITM_DRV. (Bus Agents)	HITM signal asserted	This event counts the number of bus cycles during which the processor drives the HITM# pin to signal HITM snoop response.
7DH	See Table 18-7	BUSQ_EMPTY. (Core)	Bus queue empty	This event counts the number of cycles during which the core did not have any pending transactions in the bus queue. It also counts when the core is halted and the other core is not halted. This event can count occurrences for this core or both cores.
7EH	See Table 18-7 and Table 18-8	SNOOP_STALL_ DRV.(Core and Bus Agents)	Bus stalled for snoops	This event counts the number of times that the bus snoop stall signal is asserted. To obtain the number of bus cycles during which snoops on the bus are prohibited, multiply the event count by two. During the snoop stall cycles, no new bus transactions requiring a snoop response can be initiated on the bus. A bus agent asserts a snoop stall signal if it cannot response to a snoop request within three bus cycles.
7FH	See Table 18-7	BUS_IO_WAIT. (Core)	IO requests waiting in the bus queue	This event counts the number of core cycles during which IO requests wait in the bus queue. With the SELF modifier this event counts IO requests per core. With the BOTH_CORE modifier, this event increments by one for any cycle for which there is a request from either core.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
80H	00H	L1I_READS	Instruction fetches	This event counts all instruction fetches, including uncacheable fetches that bypass the Instruction Fetch Unit (IFU).
81H	00H	L1I_MISSES	Instruction Fetch Unit misses	This event counts all instruction fetches that miss the Instruction Fetch Unit (IFU) or produce memory requests. This includes uncacheable fetches. An instruction fetch miss is counted only once and not once for every cycle it is outstanding.
82H	02H	ITLB.SMALL_MISS	ITLB small page misses	This event counts the number of instruction fetches from small pages that miss the ITLB.
82H	10H	ITLB.LARGE_MISS	ITLB large page misses	This event counts the number of instruction fetches from large pages that miss the ITLB.
82H	40H	ITLB.FLUSH	ITLB flushes	This event counts the number of ITLB flushes. This usually happens upon CR3 or CR0 writes, which are executed by the operating system during process switches.
82H	12H	ITLB.MISSES	ITLB misses	This event counts the number of instruction fetches from either small or large pages that miss the ITLB.
83H	02H	INST_QUEUE.FULL	Cycles during which the instruction queue is full	This event counts the number of cycles during which the instruction queue is full. In this situation, the core front-end stops fetching more instructions. This is an indication of very long stalls in the back-end pipeline stages.
86H	00H	CYCLES_L1I_ MEM_STALLED	Cycles during which instruction fetches stalled	This event counts the number of cycles for which an instruction fetch stalls, including stalls due to any of the following reasons: ▪ instruction Fetch Unit cache misses ▪ instruction TLB misses ▪ instruction TLB faults

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
87H	00H	ILD_STALL	Instruction Length Decoder stall cycles due to a length changing prefix	This event counts the number of cycles during which the instruction length decoder uses the slow length decoder. Usually, instruction length decoding is done in one cycle. When the slow decoder is used, instruction decoding requires 6 cycles. The slow decoder is used in the following cases: ▪ operand override prefix (66H) preceding an instruction with immediate data ▪ address override prefix (67H) preceding an instruction with a modr/m in real, big real, 16-bit protected or 32-bit protected modes To avoid instruction length decoding stalls, generate code using imm8 or imm32 values instead of imm16 values. If you must use an imm16 value, store the value in a register using "mov reg, imm32" and use the register format of the instruction.
88H	00H	BR_INST_EXEC	Branch instructions executed	This event counts all executed branches (not necessarily retired). This includes only instructions and not micro-op branches. Frequent branching is not necessarily a major performance issue. However frequent branch mispredictions may be a problem.
89H	00H	BR_MISSP_EXEC	Mispredicted branch instructions executed	This event counts the number of mispredicted branch instructions that were executed.
8AH	00H	BR_BAC_ MISSP_EXEC	Branch instructions mispredicted at decoding	This event counts the number of branch instructions that were mispredicted at decoding.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
8BH	00H	BR_CND_EXEC	Conditional branch instructions executed.	This event counts the number of conditional branch instructions executed, but not necessarily retired.
8CH	00H	BR_CND_ MISSP_EXEC	Mispredicted conditional branch instructions executed	This event counts the number of mispredicted conditional branch instructions that were executed.
8DH	00H	BR_IND_EXEC	Indirect branch instructions executed	This event counts the number of indirect branch instructions that were executed.
8EH	00H	BR_IND_MISSP _EXEC	Mispredicted indirect branch instructions executed	This event counts the number of mispredicted indirect branch instructions that were executed.
8FH	00H	BR_RET_EXEC	RET instructions executed	This event counts the number of RET instructions that were executed.
90H	00H	BR_RET_ MISSP_EXEC	Mispredicted RET instructions executed	This event counts the number of mispredicted RET instructions that were executed.
91H	00H	BR_RET_BAC_ MISSP_EXEC	RET instructions executed mispredicted at decoding	This event counts the number of RET instructions that were executed and were mispredicted at decoding.
92H	00H	BR_CALL_EXEC	CALL instructions executed	This event counts the number of CALL instructions executed
93H	00H	BR_CALL_ MISSP_EXEC	Mispredicted CALL instructions executed	This event counts the number of mispredicted CALL instructions that were executed.
94H	00H	BR_IND_CALL_ EXEC	Indirect CALL instructions executed	This event counts the number of indirect CALL instructions that were executed.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
97H	00H	BR_TKN_ BUBBLE_1	Branch predicted taken with bubble 1	The events BR_TKN_BUBBLE_1 and BR_TKN_BUBBLE_2 together count the number of times a taken branch prediction incurred a one-cycle penalty. The penalty incurs when: ▪ Too many taken branches are placed together. To avoid this, unroll loops and add a non-taken branch in the middle of the taken sequence. ▪ The branch target is unaligned. To avoid this, align the branch target.
98H	00H	BR_TKN_ BUBBLE_2	Branch predicted taken with bubble 2	The events BR_TKN_BUBBLE_1 and BR_TKN_BUBBLE_2 together count the number of times a taken branch prediction incurred a one-cycle penalty. The penalty incurs when: ▪ Too many taken branches are placed together. To avoid this, unroll loops and add a non-taken branch in the middle of the taken sequence. ▪ The branch target is unaligned. To avoid this, align the branch target.
A0H	00H	RS_UOPS_ DISPATCHED	Micro-ops dispatched for execution	This event counts the number of micro-ops dispatched for execution. Up to six micro-ops can be dispatched in each cycle.
A1H	01H	RS_UOPS_ DISPATCHED.PORT 0	Cycles micro-ops dispatched for execution on port 0	This event counts the number of cycles for which micro-ops dispatched for execution. Each cycle, at most one micro-op can be dispatched on the port. Issue Ports are described in *Intel® 64 and IA-32 Architectures Optimization Reference Manual.*
A1H	02H	RS_UOPS_ DISPATCHED.PORT 1	Cycles micro-ops dispatched for execution on port 1	This event counts the number of cycles for which micro-ops dispatched for execution. Each cycle, at most one micro-op can be dispatched on the port.
A1H	04H	RS_UOPS_ DISPATCHED.PORT 2	Cycles micro-ops dispatched for execution on port 2	This event counts the number of cycles for which micro-ops dispatched for execution. Each cycle, at most one micro-op can be dispatched on the port.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
A1H	08H	RS_UOPS_ DISPATCHED.PORT 3	Cycles micro-ops dispatched for execution on port 3	This event counts the number of cycles for which micro-ops dispatched for execution. Each cycle, at most one micro-op can be dispatched on the port.
A1H	10H	RS_UOPS_ DISPATCHED.PORT 4	Cycles micro-ops dispatched for execution on port 4	This event counts the number of cycles for which micro-ops dispatched for execution. Each cycle, at most one micro-op can be dispatched on the port.
A1H	20H	RS_UOPS_ DISPATCHED.PORT 5	Cycles micro-ops dispatched for execution on port 5	This event counts the number of cycles for which micro-ops dispatched for execution. Each cycle, at most one micro-op can be dispatched on the port.
AAH	01H	MACRO_INSTS. DECODED	Instructions decoded	This event counts the number of instructions decoded (but not necessarily executed or retired).
AAH	08H	MACRO_INSTS. CISC_DECODED	CISC Instructions decoded	This event counts the number of complex instructions decoded. Complex instructions usually have more than four micro-ops. Only one complex instruction can be decoded at a time.
ABH	01H	ESP.SYNCH	ESP register content synchron-ization	This event counts the number of times that the ESP register is explicitly used in the address expression of a load or store operation, after it is implicitly used, for example by a push or a pop instruction. ESP synch micro-op uses resources from the rename pipe-stage and up to retirement. The expected ratio of this event divided by the number of ESP implicit changes is 0,2. If the ratio is higher, consider rearranging your code to avoid ESP synchronization events.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
ABH	02H	ESP.ADDITIONS	ESP register automatic additions	This event counts the number of ESP additions performed automatically by the decoder. A high count of this event is good, since each automatic addition performed by the decoder saves a micro-op from the execution units. To maximize the number of ESP additions performed automatically by the decoder, choose instructions that implicitly use the ESP, such as PUSH, POP, CALL, and RET instructions whenever possible.
B0H	00H	SIMD_UOPS_EXEC	SIMD micro-ops executed (excluding stores)	This event counts all the SIMD micro-ops executed. It does not count MOVQ and MOVD stores from register to memory.
B1H	00H	SIMD_SAT_UOP_ EXEC	SIMD saturated arithmetic micro-ops executed	This event counts the number of SIMD saturated arithmetic micro-ops executed.
B3H	01H	SIMD_UOP_ TYPE_EXEC.MUL	SIMD packed multiply micro-ops executed	This event counts the number of SIMD packed multiply micro-ops executed.
B3H	02H	SIMD_UOP_TYPE_ EXEC.SHIFT	SIMD packed shift micro-ops executed	This event counts the number of SIMD packed shift micro-ops executed.
B3H	04H	SIMD_UOP_TYPE_ EXEC.PACK	SIMD pack micro-ops executed	This event counts the number of SIMD pack micro-ops executed.
B3H	08H	SIMD_UOP_TYPE_ EXEC.UNPACK	SIMD unpack micro-ops executed	This event counts the number of SIMD unpack micro-ops executed.
B3H	10H	SIMD_UOP_TYPE_ EXEC.LOGICAL	SIMD packed logical micro-ops executed	This event counts the number of SIMD packed logical micro-ops executed.
B3H	20H	SIMD_UOP_TYPE_ EXEC.ARITHMETIC	SIMD packed arithmetic micro-ops executed	This event counts the number of SIMD packed arithmetic micro-ops executed.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
C0H	00H	INST_RETIRED. ANY_P	Instructions retired	This event counts the number of instructions that retire execution. For instructions that consist of multiple micro-ops, this event counts the retirement of the last micro-op of the instruction. The counter continue counting during hardware interrupts, traps, and inside interrupt handlers. INST_RETIRED.ANY_P is an architectural performance event.
C0H	01H	INST_RETIRED. LOADS	Instructions retired, which contain a load	This event counts the number of instructions retired that contain a load operation.
C0H	02H	INST_RETIRED. STORES	Instructions retired, which contain a store	This event counts the number of instructions retired that contain a store operation.
C0H	04H	INST_RETIRED. OTHER	Instructions retired, with no load or store operation	This event counts the number of instructions retired that do not contain a load or a store operation.
C1H	01H	X87_OPS_ RETIRED.FXCH	FXCH instructions retired	This event counts the number of FXCH instructions retired. Modern compilers generate more efficient code and are less likely to use this instruction. If you obtain a high count for this event consider recompiling the code.
C1H	FEH	X87_OPS_ RETIRED.ANY	Retired floating-point computational operations (precise event)	This event counts the number of floating-point computational operations retired. It counts: ▪ floating point computational operations executed by the assist handler ▪ sub-operations of complex floating-point instructions like transcendental instructions

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
				This event does not count: ▪ floating-point computational operations that cause traps or assists. ▪ floating-point loads and stores. When this event is captured with the precise event mechanism, the collected samples contain the address of the instruction that was executed immediately after the instruction that caused the event.
C2H	01H	UOPS_RETIRED. LD_IND_BR	Fused load+op or load+indirect branch retired	This event counts the number of retired micro-ops that fused a load with another operation. This includes: ▪ Fusion of a load and an arithmetic operation, such as with the following instruction: ADD EAX, [EBX] where the content of the memory location specified by EBX register is loaded, added to EXA register, and the result is stored in EAX. ▪ Fusion of a load and a branch in an indirect branch operation, such as with the following instructions: ▪ JMP [RDI+200] ▪ RET ▪ Fusion decreases the number of micro-ops in the processor pipeline. A high value for this event count indicates that the code is using the processor resources effectively.
C2H	02H	UOPS_RETIRED. STD_STA	Fused store address + data retired	This event counts the number of store address calculations that are fused with store data emission into one micro-op. Traditionally, each store operation required two micro-ops. This event counts fusion of retired micro-ops only. Fusion decreases the number of micro-ops in the processor pipeline. A high value for this event count indicates that the code is using the processor resources effectively.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
C2H	04H	UOPS_RETIRED. MACRO_FUSION	Retired instruction pairs fused into one micro-op	This event counts the number of times CMP or TEST instructions were fused with a conditional branch instruction into one micro-op. It counts fusion by retired micro-ops only. Fusion decreases the number of micro-ops in the processor pipeline. A high value for this event count indicates that the code uses the processor resources more effectively.
C2H	07H	UOPS_RETIRED. FUSED	Fused micro-ops retired	This event counts the total number of retired fused micro-ops. The counts include the following fusion types: ▪ Fusion of load operation with an arithmetic operation or with an indirect branch (counted by event UOPS_RETIRED.LD_IND_BR) ▪ Fusion of store address and data (counted by event UOPS_RETIRED.STD_STA) ▪ Fusion of CMP or TEST instruction with a conditional branch instruction (counted by event UOPS_RETIRED.MACRO_FUSION) Fusion decreases the number of micro-ops in the processor pipeline. A high value for this event count indicates that the code is using the processor resources effectively.
C2H	08H	UOPS_RETIRED. NON_FUSED	Non-fused micro-ops retired	This event counts the number of micro-ops retired that were not fused.
C2H	0FH	UOPS_RETIRED. ANY	Micro-ops retired	This event counts the number of micro-ops retired. The processor decodes complex macro instructions into a sequence of simpler micro-ops. Most instructions are composed of one or two micro-ops.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
				Some instructions are decoded into longer sequences such as repeat instructions, floating point transcendental instructions, and assists. In some cases micro-op sequences are fused or whole instructions are fused into one micro-op. See other UOPS_RETIRED events for differentiating retired fused and non-fused micro-ops.
C3H	01H	MACHINE_ NUKES.SMC	Self-Modifying Code detected	This event counts the number of times that a program writes to a code section. Self-modifying code causes a sever penalty in all Intel 64 and IA-32 processors.
C3H	04H	MACHINE_NUKES. MEM_ORDER	Execution pipeline restart due to memory ordering conflict or memory disambiguation misprediction	This event counts the number of times the pipeline is restarted due to either multi-threaded memory ordering conflicts or memory disambiguation misprediction. A multi-threaded memory ordering conflict occurs when a store, which is executed in another core, hits a load that is executed out of order in this core but not yet retired. As a result, the load needs to be restarted to satisfy the memory ordering model. See Chapter 7, "Multiple-Processor Management" in the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A.* To count memory disambiguation mispredictions, use the event MEMORY_DISAMBIGUATION.RESET.
C4H	00H	BR_INST_RETIRED. ANY	Retired branch instructions	This event counts the number of branch instructions retired. This is an architectural performance event.
C4H	01H	BR_INST_RETIRED. PRED_NOT_ TAKEN	Retired branch instructions that were predicted not-taken	This event counts the number of branch instructions retired that were correctly predicted to be not-taken.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
C4H	02H	BR_INST_RETIRED. MISPRED_NOT_ TAKEN	Retired branch instructions that were mispredicted not-taken	This event counts the number of branch instructions retired that were mispredicted and not-taken.
C4H	04H	BR_INST_RETIRED. PRED_TAKEN	Retired branch instructions that were predicted taken	This event counts the number of branch instructions retired that were correctly predicted to be taken.
C4H	08H	BR_INST_RETIRED. MISPRED_TAKEN	Retired branch instructions that were mispredicted taken	This event counts the number of branch instructions retired that were mispredicted and taken.
C4H	0CH	BR_INST_RETIRED. TAKEN	Retired taken branch instructions	This event counts the number of branches retired that were taken.
C5H	00H	BR_INST_RETIRED. MISPRED	Retired mispredicted branch instructions. (precise event)	This event counts the number of retired branch instructions that were mispredicted by the processor. A branch misprediction occurs when the processor predicts that the branch would be taken, but it is not, or vice-versa. This is an architectural performance event.
C6H	01H	CYCLES_INT_ MASKED	Cycles during which interrupts are disabled	This event counts the number of cycles during which interrupts are disabled.
C6H	02H	CYCLES_INT_ PENDING_AND _MASKED	Cycles during which interrupts are pending and disabled	This event counts the number of cycles during which there are pending interrupts but interrupts are disabled.
C7H	01H	SIMD_INST_ RETIRED.PACKED_ SINGLE	Retired SSE packed-single instructions	This event counts the number of SSE packed-single instructions retired.
C7H	02H	SIMD_INST_ RETIRED.SCALAR_ SINGLE	Retired SSE scalar-single instructions	This event counts the number of SSE scalar-single instructions retired.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
C7H	04H	SIMD_INST_ RETIRED.PACKED_ DOUBLE	Retired SSE2 packed-double instructions	This event counts the number of SSE2 packed-double instructions retired.
C7H	08H	SIMD_INST_ RETIRED.SCALAR_ DOUBLE	Retired SSE2 scalar-double instructions	This event counts the number of SSE2 scalar-double instructions retired.
C7H	10H	SIMD_INST_ RETIRED.VECTOR	Retired SSE2 vector integer instructions	This event counts the number of SSE2 vector integer instructions retired.
C7H	1FH	SIMD_INST_ RETIRED.ANY	Retired Streaming SIMD instructions (precise event)	This event counts the overall number of SIMD instructions retired. To count each type of SIMD instruction separately, use the following events: ▪ SIMD_INST_RETIRED.PACKED_SINGLE ▪ SIMD_INST_RETIRED.SCALAR_SINGLE ▪ SIMD_INST_RETIRED.PACKED_DOUBLE ▪ SIMD_INST_RETIRED.SCALAR_DOUBLE ▪ and SIMD_INST_RETIRED.VECTOR When this event is captured with the precise event mechanism, the collected samples contain the address of the instruction that was executed immediately after the instruction that caused the event.
C8H	00H	HW_INT_RCV	Hardware interrupts received	This event counts the number of hardware interrupts received by the processor.
C9H	00H	ITLB_MISS_ RETIRED	Retired instructions that missed the ITLB	This event counts the number of retired instructions that missed the ITLB when they were fetched.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
CAH	01H	SIMD_COMP_ INST_RETIRED. PACKED_SINGLE	Retired computational SSE packed-single instructions	This event counts the number of computational SSE packed-single instructions retired. Computational instructions perform arithmetic computations (for example: add, multiply and divide). Instructions that perform load and store operations or logical operations, like XOR, OR, and AND are not counted by this event.
CAH	02H	SIMD_COMP_ INST_RETIRED. SCALAR_SINGLE	Retired computational SSE scalar-single instructions	This event counts the number of computational SSE scalar-single instructions retired. Computational instructions perform arithmetic computations (for example: add, multiply and divide). Instructions that perform load and store operations or logical operations, like XOR, OR, and AND are not counted by this event.
CAH	04H	SIMD_COMP_ INST_RETIRED. PACKED_DOUBLE	Retired computational SSE2 packed-double instructions	This event counts the number of computational SSE2 packed-double instructions retired. Computational instructions perform arithmetic computations (for example: add, multiply and divide). Instructions that perform load and store operations or logical operations, like XOR, OR, and AND are not counted by this event.
CAH	08H	SIMD_COMP_INST_ RETIRED.SCALAR_ DOUBLE	Retired computational SSE2 scalar-double instructions	This event counts the number of computational SSE2 scalar-double instructions retired. Computational instructions perform arithmetic computations (for example: add, multiply and divide). Instructions that perform load and store operations or logical operations, like XOR, OR, and AND are not counted by this event.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
CBH	01H	MEM_LOAD_ RETIRED.L1D _MISS	Retired loads that miss the L1 data cache (precise event)	This event counts the number of retired load operations that missed the L1 data cache. This includes loads from cache lines that are currently being fetched, due to a previous L1 data cache miss to the same cache line. This event counts loads from cacheable memory only. The event does not count loads by software prefetches. When this event is captured with the precise event mechanism, the collected samples contain the address of the instruction that was executed immediately after the instruction that caused the event.
CBH	02H	MEM_LOAD_ RETIRED.L1D_ LINE_MISS	L1 data cache line missed by retired loads (precise event)	This event counts the number of load operations that miss the L1 data cache and send a request to the L2 cache to fetch the missing cache line. That is the missing cache line fetching has not yet started. The event count is equal to the number of cache lines fetched from the L2 cache by retired loads. This event counts loads from cacheable memory only. The event does not count loads by software prefetches. The event might not be counted if the load is blocked (see LOAD_BLOCK events).
				When this event is captured with the precise event mechanism, the collected samples contain the address of the instruction that was executed immediately after the instruction that caused the event.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
CBH	04H	MEM_LOAD_ RETIRED.L2_MISS	Retired loads that miss the L2 cache (precise event)	This event counts the number of retired load operations that missed the L2 cache. This event counts loads from cacheable memory only. It does not count loads by software prefetches. When this event is captured with the precise event mechanism, the collected samples contain the address of the instruction that was executed immediately after the instruction that caused the event.
CBH	08H	MEM_LOAD_ RETIRED.L2_LINE_ MISS	L2 cache line missed by retired loads (precise event)	This event counts the number of load operations that miss the L2 cache and result in a bus request to fetch the missing cache line. That is the missing cache line fetching has not yet started. This event count is equal to the number of cache lines fetched from memory by retired loads. This event counts loads from cacheable memory only. The event does not count loads by software prefetches. The event might not be counted if the load is blocked (see LOAD_BLOCK events). When this event is captured with the precise event mechanism, the collected samples contain the address of the instruction that was executed immediately after the instruction that caused the event.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
CBH	10H	MEM_LOAD_ RETIRED.DTLB_ MISS	Retired loads that miss the DTLB (precise event)	This event counts the number of retired loads that missed the DTLB. The DTLB miss is not counted if the load operation causes a fault. This event counts loads from cacheable memory only. The event does not count loads by software prefetches. When this event is captured with the precise event mechanism, the collected samples contain the address of the instruction that was executed immediately after the instruction that caused the event.
CCH	01H	FP_MMX_TRANS_ TO_MMX	Transitions from Floating Point to MMX Instructions	This event counts the first MMX instructions following a floating-point instruction. Use this event to estimate the penalties for the transitions between floating-point and MMX states.
CCH	02H	FP_MMX_TRANS_ TO_FP	Transitions from MMX Instructions to Floating Point Instructions	This event counts the first floating-point instructions following any MMX instruction. Use this event to estimate the penalties for the transitions between floating-point and MMX states.
CDH	00H	SIMD_ASSIST	SIMD assists invoked	This event counts the number of SIMD assists invoked. SIMD assists are invoked when an EMMS instruction is executed, changing the MMX state in the floating point stack.
CEH	00H	SIMD_INSTR_ RETIRED	SIMD Instructions retired	This event counts the number of SIMD instructions that retired.
CFH	00H	SIMD_SAT_INSTR_ RETIRED	Saturated arithmetic instructions retired	This event counts the number of saturated arithmetic SIMD instructions that retired.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
D2H	01H	RAT_STALLS. ROB_READ_PORT	ROB read port stalls cycles	This event counts the number of cycles when ROB read port stalls occurred, which did not allow new micro-ops to enter the out-of-order pipeline. Note that, at this stage in the pipeline, additional stalls may occur at the same cycle and prevent the stalled micro-ops from entering the pipe. In such a case, micro-ops retry entering the execution pipe in the next cycle and the ROB-read-port stall is counted again.
D2H	02H	RAT_STALLS. PARTIAL_CYCLES	Partial register stall cycles	This event counts the number of cycles instruction execution latency became longer than the defined latency because the instruction uses a register that was partially written by previous instructions.
D2H	04H	RAT_STALLS. FLAGS	Flag stall cycles	This event counts the number of cycles during which execution stalled due to several reasons, one of which is a partial flag register stall. A partial register stall may occur when two conditions are met: ▪ an instruction modifies some, but not all, of the flags in the flag register ▪ the next instruction, which depends on flags, depends on flags that were not modified by this instruction
D2H	08H	RAT_STALLS. FPSW	FPU status word stall	This event indicates that the FPU status word (FPSW) is written. To obtain the number of times the FPSW is written divide the event count by 2. The FPSW is written by instructions with long latency; a small count may indicate a high penalty.
D2H	0FH	RAT_STALLS. ANY	All RAT stall cycles	This event counts the number of stall cycles due to conditions described by: ▪ RAT_STALLS.ROB_READ_PORT ▪ RAT_STALLS.PARTIAL ▪ RAT_STALLS.FLAGS ▪ RAT_STALLS.FPSW.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
D4H	01H	SEG_RENAME_ STALLS.ES	Segment rename stalls - ES	This event counts the number of stalls due to the lack of renaming resources for the ES segment register. If a segment is renamed, but not retired and a second update to the same segment occurs, a stall occurs in the front-end of the pipeline until the renamed segment retires.
D4H	02H	SEG_RENAME_ STALLS.DS	Segment rename stalls - DS	This event counts the number of stalls due to the lack of renaming resources for the DS segment register. If a segment is renamed, but not retired and a second update to the same segment occurs, a stall occurs in the front-end of the pipeline until the renamed segment retires.
D4H	04H	SEG_RENAME_ STALLS.FS	Segment rename stalls - FS	This event counts the number of stalls due to the lack of renaming resources for the FS segment register. If a segment is renamed, but not retired and a second update to the same segment occurs, a stall occurs in the front-end of the pipeline until the renamed segment retires.
D4H	08H	SEG_RENAME_ STALLS.GS	Segment rename stalls - GS	This event counts the number of stalls due to the lack of renaming resources for the GS segment register. If a segment is renamed, but not retired and a second update to the same segment occurs, a stall occurs in the front-end of the pipeline until the renamed segment retires.
D4H	0FH	SEG_RENAME_ STALLS.ANY	Any (ES/DS/FS/GS) segment rename stall	This event counts the number of stalls due to the lack of renaming resources for the ES, DS, FS, and GS segment registers. If a segment is renamed but not retired and a second update to the same segment occurs, a stall occurs in the front-end of the pipeline until the renamed segment retires.
D5H	01H	SEG_REG_ RENAMES.ES	Segment renames - ES	This event counts the number of times the ES segment register is renamed.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
D5H	02H	SEG_REG_ RENAMES.DS	Segment renames - DS	This event counts the number of times the DS segment register is renamed.
D5H	04H	SEG_REG_ RENAMES.FS	Segment renames - FS	This event counts the number of times the FS segment register is renamed.
D5H	08H	SEG_REG_ RENAMES.GS	Segment renames - GS	This event counts the number of times the GS segment register is renamed.
D5H	0FH	SEG_REG_ RENAMES.ANY	Any (ES/DS/FS/GS) segment rename	This event counts the number of times any of the four segment registers (ES/DS/FS/GS) is renamed.
DCH	01H	RESOURCE_ STALLS.ROB_FULL	Cycles during which the ROB full	This event counts the number of cycles when the number of instructions in the pipeline waiting for retirement reaches the limit the processor can handle. A high count for this event indicates that there are long latency operations in the pipe (possibly load and store operations that miss the L2 cache, and other instructions that depend on these cannot execute until the former instructions complete execution). In this situation new instructions can not enter the pipe and start execution.
DCH	02H	RESOURCE_ STALLS.RS_FULL	Cycles during which the RS full	This event counts the number of cycles when the number of instructions in the pipeline waiting for execution reaches the limit the processor can handle. A high count of this event indicates that there are long latency operations in the pipe (possibly load and store operations that miss the L2 cache, and other instructions that depend on these cannot execute until the former instructions complete execution). In this situation new instructions can not enter the pipe and start execution.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
DCH	04	RESOURCE_ STALLS.LD_ST	Cycles during which the pipeline has exceeded load or store limit or waiting to commit all stores	This event counts the number of cycles while resource-related stalls occur due to: ▪ The number of load instructions in the pipeline reached the limit the processor can handle. The stall ends when a loading instruction retires. ▪ The number of store instructions in the pipeline reached the limit the processor can handle. The stall ends when a storing instruction commits its data to the cache or memory. ▪ There is an instruction in the pipe that can be executed only when all previous stores complete and their data is committed in the caches or memory. For example, the SFENCE and MFENCE instructions require this behavior.
DCH	08H	RESOURCE_ STALLS.FPCW	Cycles stalled due to FPU control word write	This event counts the number of cycles while execution was stalled due to writing the floating-point unit (FPU) control word.
DCH	10H	RESOURCE_ STALLS.BR_MISS_C LEAR	Cycles stalled due to branch misprediction	This event counts the number of cycles after a branch misprediction is detected at execution until the branch and all older micro-ops retire. During this time new micro-ops cannot enter the out-of-order pipeline.
DCH	1FH	RESOURCE_ STALLS.ANY	Resource related stalls	This event counts the number of cycles while resource-related stalls occurs for any conditions described by the following events: ▪ RESOURCE_STALLS.ROB_FULL ▪ RESOURCE_STALLS.RS_FULL ▪ RESOURCE_STALLS.LD_ST ▪ RESOURCE_STALLS.FPCW ▪ RESOURCE_STALLS.BR_MISS_CLEAR
E0H	00H	BR_INST_ DECODED	Branch instructions decoded	This event counts the number of branch instructions decoded.

Table A-3. Non-Architectural Performance Events in Processors Based on Intel Core Microarchitecture (Contd.)

Event Num	Umask Value	Event Name	Definition	Description and Comment
E4H	00H	BOGUS_BR	Bogus branches	This event counts the number of byte sequences that were mistakenly detected as taken branch instructions. This results in a BACLEAR event. This occurs mainly after task switches.
E6H	00H	BACLEARS	BACLEARS asserted	This event counts the number of times the front end is resteered, mainly when the BPU cannot provide a correct prediction and this is corrected by other branch handling mechanisms at the front and. This can occur if the code has many branches such that they cannot be consumed by the BPU. Each BACLEAR asserted costs approximately 7 cycles of instruction fetch. The effect on total execution time depends on the surrounding code.
F0	00H	PREF_RQSTS_UP	Upward prefetches issued from DPL	This event counts the number of upward prefetches issued from the Data Prefetch Logic (DPL) to the L2 cache. A prefetch request issued to the L2 cache cannot be cancelled and the requested cache line is fetched to the L2 cache.
F8	00H	PREF_RQSTS_DN	Downward prefetches issued from DPL.	This event counts the number of downward prefetches issued from the Data Prefetch Logic (DPL) to the L2 cache. A prefetch request issued to the L2 cache cannot be cancelled and the requested cache line is fetched to the L2 cache.

A.3 PERFORMANCE MONITORING EVENTS FOR INTEL® CORE™ SOLO AND INTEL® CORE™ DUO PROCESSORS

Table A-4 lists non-architectural performance events for Intel Core Duo processors. If a non-architectural event requires qualification in core specificity, it is indicated in the comment column. Table A-4 also applies to Intel Core Solo processors; bits in the unit mask corresponding to core-specificity are reserved and should be 00B.