Registers	ID	CTRL
MSR_PERF_FIXED_CTR0	0x309	MSR_PERF_FIXED_CTR_CTRL (0x38D)
MSR_PERF_FIXED_CTR1	0x30A	MSR_PERF_FIXED_CTR_CTRL (0x38D)
MSR_PERF_FIXED_CTR2	0x30B	MSR_PERF_FIXED_CTR_CTRL (0x38D)
MSR_PKG_ENERGY_STATUS	0x611	NA
MSR_PP0_ENERGY_STATUS	0x639	NA
MSR_PP1_ENERGY_STATUS	0x641	NA
MSR_DRAM_ENERGY_STATUS	0x619	NA
MSR_PMC0	0x0C1	MSR_PERFEVTSEL0 (0x186)
MSR_PMC1	0x0C2	MSR_PERFEVTSEL1 (0x187)
MSR_PMC2	0x0C3	MSR_PERFEVTSEL2 (0x188)
MSR_PMC3	0x0C4	MSR_PERFEVTSEL3 (0x189)

17.4.8.1 LBR Stack and Intel® 64 Processors

LBR MSRs are 64-bits. In 64-bit mode, last branch records store the full address. Outside of 64-bit mode, the upper 32-bits of branch addresses will be stored as 0.

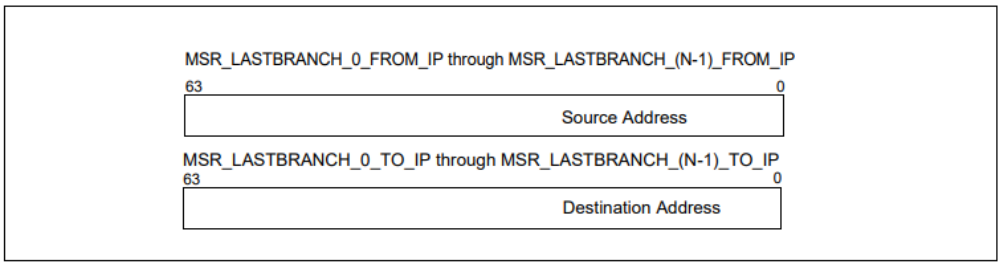

Figure 17-4. 64-bit Address Layout of LBR MSR

Software should query an architectural MSR IA32_PERF_CAPABILITIES[5:0] about the format of the address that is stored in the LBR stack. Four formats are defined by the following encoding:

- **000000B (32-bit record format) —** Stores 32-bit offset in current CS of respective source/destination,
- **000001B (64-bit LIP record format) —** Stores 64-bit linear address of respective source/destination,
- **000010B (64-bit EIP record format) —** Stores 64-bit offset (effective address) of respective source/destination.
- **000011B (64-bit EIP record format) and Flags** — Stores 64-bit offset (effective address) of respective source/destination. Misprediction info is reported in the upper bit of 'FROM' registers in the LBR stack. See LBR stack details below for flag support and definition.
- **000100B (64-bit EIP record format), Flags and TSX** — Stores 64-bit offset (effective address) of respective source/destination. Misprediction and TSX info are reported in the upper bits of 'FROM' registers in the LBR stack.

14.10.3 Package RAPL Domain

The MSR interfaces defined for the package RAPL domain are:

- MSR_PKG_POWER_LIMIT allows software to set power limits for the package and measurement attributes associated with each limit,
- MSR_PKG_ENERGY_STATUS reports measured actual energy usage,
- MSR_PKG_POWER_INFO reports the package power range information for RAPL usage.

MSR_PKG_PERF_STATUS can report the performance impact of power limiting, but its availability may be model-specific.

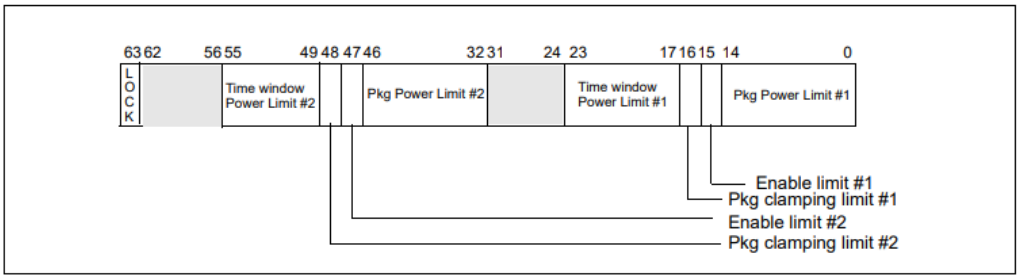

Figure 14-36. MSR_PKG_POWER_LIMIT Register

MSR_PKG_POWER_LIMIT allows a software agent to define power limitation for the package domain. Power limitation is defined in terms of average power usage (Watts) over a time window specified in MSR_PKG_POWER_LIMIT. Two power limits can be specified, corresponding to time windows of different sizes. Each power limit provides independent clamping control that would permit the processor cores to go below OS-requested state to meet the power limits. A lock mechanism allow the software agent to enforce power limit settings. Once the lock bit is set, the power limit settings are static and un-modifiable until next RESET.

The bit fields of MSR_PKG_POWER_LIMIT (Figure 14-36) are:

- **Package Power Limit #1**(bits 14:0): Sets the average power usage limit of the package domain corresponding to time window # 1. The unit of this field is specified by the "Power Units" field of MSR_RAPL_POWER_UNIT.

MSR_PKG_ENERGY_STATUS is a read-only MSR. It reports the actual energy use for the package domain. This MSR is updated every ~1msec. It has a wraparound time of around 60 secs when power consumption is high, and may be longer otherwise.

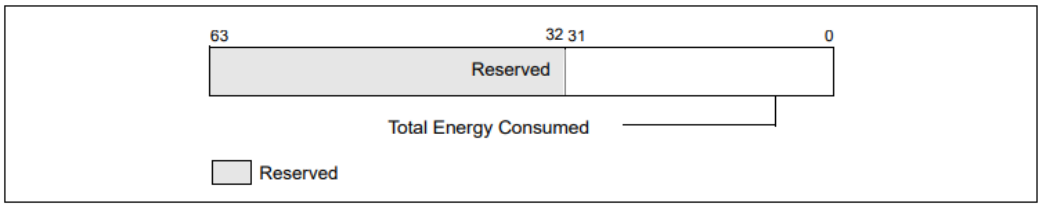

Figure 14-37. MSR_PKG_ENERGY_STATUS MSR

- **Total Energy Consumed** (bits 31:0): The unsigned integer value represents the total amount of energy consumed since that last time this register is cleared. The unit of this field is specified by the "Energy Status Units" field of MSR_RAPL_POWER_UNIT.

- **Total Energy Consumed** (bits 31:0): The unsigned integer value represents the total amount of energy consumed since that last time this register is cleared. The unit of this field is specified by the "Energy Status Units" field of MSR_RAPL_POWER_UNIT.

MSR_PKG_POWER_INFO is a read-only MSR. It reports the package power range information for RAPL usage. This MSR provides maximum/minimum values (derived from electrical specification), thermal specification power of the package domain. It also provides the largest possible time window for software to program the RAPL interface.

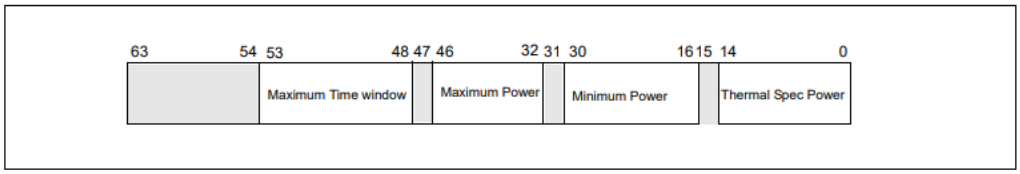

Figure 14-38. MSR_PKG_POWER_INFO Register

MSR_DRAM_ENERGY_STATUS is a read-only MSR. It reports the actual energy use for the DRAM domain. This MSR is updated every ~1msec.

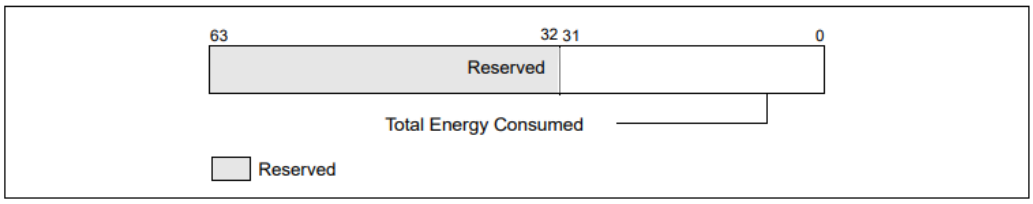

Figure 14-45. MSR_DRAM_ENERGY_STATUS MSR

- **Total Energy Consumed** (bits 31:0): The unsigned integer value represents the total amount of energy consumed since that last time this register is cleared. The unit of this field is specified by the "Energy Status Units" field of MSR_RAPL_POWER_UNIT.

- MSR_PP0_ENERGY_STATUS/MSR_PP1_ENERGY_STATUS report actual energy usage on a power plane.
- MSR_PP0_POLICY/MSR_PP1_POLICY allow software to adjust balance for respective power plane.

MSR_PP0_PERF_STATUS can report the performance impact of power limiting, but it is not available in client platforms.

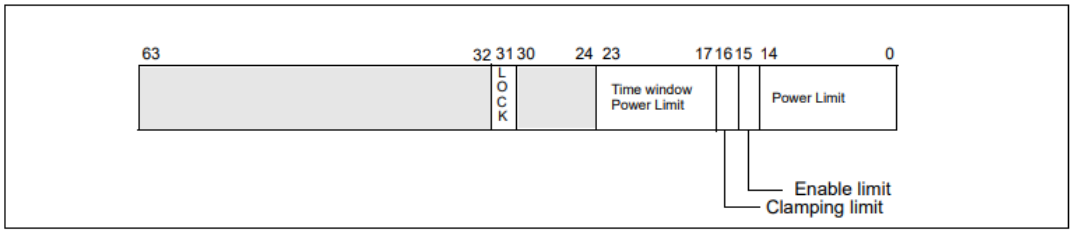

Figure 14-40. MSR_PP0_POWER_LIMIT/MSR_PP1_POWER_LIMIT Register

— For the fixed-function counters; enabling PMI is done by setting the 3rd bit in the corresponding 4-bit control field of the MSR_PERF_FIXED_CTR_CTRL register (see Figure 19-1) or IA32_FIXED_CTR_CTRL MSR (see Figure 19-2).

- The PEBS buffer is almost full and reaches the interrupt threshold

MSR_PP0_PERF_STATUS can report the performance impact of power limiting, but it is not available in client platforms.

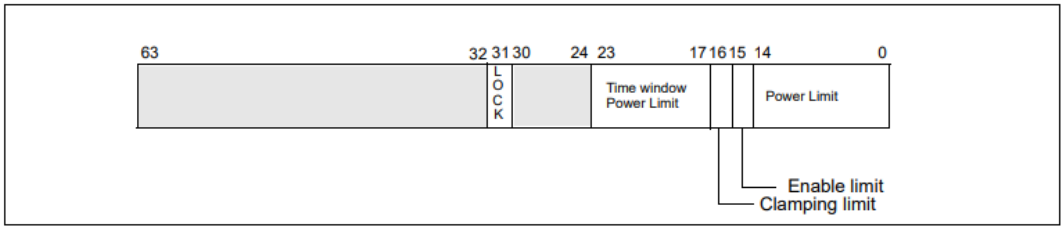

Figure 14-40. MSR_PP0_POWER_LIMIT/MSR_PP1_POWER_LIMIT Register

The IA32_MCG_STATUS MSR describes the current state of the processor after a machine-check exception has occurred (see Figure 15-3).

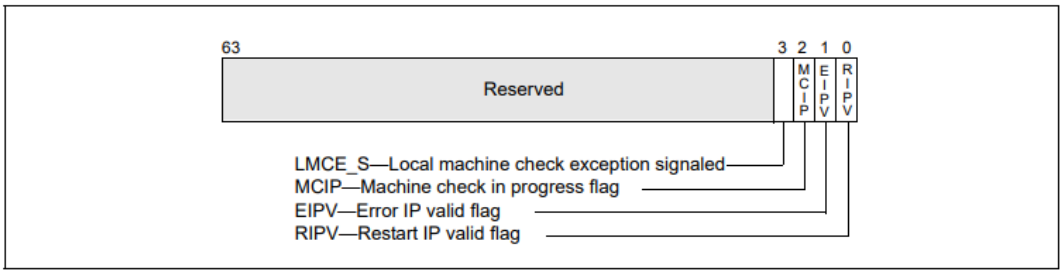

Figure 15-3. IA32_MCG_STATUS Register

Where:

- **RIPV (restart IP valid) flag, bit 0** — Indicates (when set) that program execution can be restarted reliably at the instruction pointed to by the instruction pointer pushed on the stack when the machine-check exception is generated. When clear, the program cannot be reliably restarted at the pushed instruction pointer.
- **EIPV (error IP valid) flag, bit 1** — Indicates (when set) that the instruction pointed to by the instruction pointer pushed onto the stack when the machine-check exception is generated is directly associated with the error. When this flag is cleared, the instruction pointed to may not be associated with the error.

CPU Instructions

CPUID.0AH:EAX[bits 7:0]

Description

The ID flag (bit 21) in the EFLAGS register indicates support for the CPUID instruction. If a software procedure can set and clear this flag, the processor executing the procedure supports the CPUID instruction. This instruction operates the same in non-64-bit modes and 64-bit mode.

CPUID returns processor identification and feature information in the EAX, EBX, ECX, and EDX registers. The instruction's output is dependent on the contents of the EAX register upon execution (in some cases, ECX as well). For example, the following pseudocode loads EAX with 00H and causes CPUID to return a Maximum Return Value and the Vendor Identification String in the appropriate registers:

```
MOV EAX, 00H

CPUID
```

CPUID.EAX = 05H (* Returns MONITOR/MWAIT leaf. *)

CPUID.EAX = 0AH (* Returns Architectural Performance Monitoring leaf. *) CPUID.EAX = 0BH (* Returns Extended Topology Enumeration leaf. *)[2] CPUID.EAX =1FH (* Returns V2 Extended Topology Enumeration leaf. *)[2]

1. **On Intel 64 processors, CPUID clears the high 32 bits of the RAX/RBX/RCX/RDX registers in all modes.**
2. **CPUID leaf 1FH is a preferred superset to leaf 0BH. Intel recommends first checking for the existence of CPUID leaf 1FH before using leaf 0BH.**

CPUID.EAX = 80000008H (* Returns linear/physical address size data. *)

CPUID.EAX = 8000000AH (* INVALID: Returns same information as CPUID.EAX = 0BH. *)

- **Event select field (bits 0 through 7) —** Selects the event logic unit used to detect microarchitectural conditions (see Table 19-1, for a list of architectural events and their 8-bit codes). The set of values for this field is defined architecturally; each value corresponds to an event logic unit for use with an architectural performance event. The number of architectural events is queried using CPUID.0AH:EAX. A processor may support only a subset of pre-defined values.

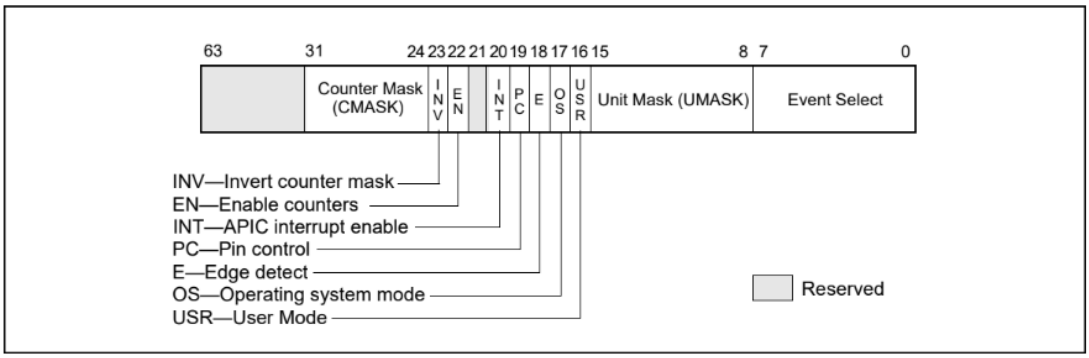

Figure 19-1. Layout of IA32_PERFEVTSELx MSRs

- **Unit mask (UMASK) field (bits 8 through 15)** — These bits qualify the condition that the selected event logic unit detects. Valid UMASK values for each event logic unit are specific to the unit. For each architectural performance event, its corresponding UMASK value defines a specific microarchitectural condition.

  A pre-defined microarchitectural condition associated with an architectural event may not be applicable to a given processor. The processor then reports only a subset of pre-defined architectural events. Pre-defined architectural events are listed in Table 19-1; support for pre-defined architectural events is enumerated using CPUID.0AH:EBX.
- **USR (user mode) flag (bit 16) —** Specifies that the selected microarchitectural condition is counted when the logical processor is operating at privilege levels 1, 2 or 3. This flag can be used with the OS flag.
- **OS (operating system mode) flag (bit 17) —** Specifies that the selected microarchitectural condition is counted when the logical processor is operating at privilege level 0. This flag can be used with the USR flag.
- **E (edge detect) flag (bit 18) —** Enables (when set) edge detection of the selected microarchitectural condition. The logical processor counts the number of deasserted to asserted transitions for any condition that can be expressed by the other fields. The mechanism does not permit back-to-back assertions to be distinguished.

  This mechanism allows software to measure not only the fraction of time spent in a particular state, but also the average length of time spent in such a state (for example, the time spent waiting for an interrupt to be serviced).
- **PC (pin control) flag (bit 19) —** Beginning with Sandy Bridge microarchitecture, this bit is reserved (not writeable). On processors based on previous microarchitectures, the logical processor toggles the PM*i* pins and increments the counter when performance-monitoring events occur; when clear, the processor toggles the PM*i* pins when the counter overflows. The toggling of a pin is defined as assertion of the pin for a single bus clock followed by deassertion.

Table 19-1. UMask and Event Select Encodings for Pre-Defined Architectural Performance Events

Bit Position CPUID.AH.EBX	Event Name	UMask	Event Select
0	UnHalted Core Cycles	00H	3CH
1	Instruction Retired	00H	C0H
2	UnHalted Reference Cycles[1]	01H	3CH
3	LLC Reference	4FH	2EH
4	LLC Misses	41H	2EH
5	Branch Instruction Retired	00H	C4H
6	Branch Misses Retired	00H	C5H
7	Topdown Slots	01H	A4H

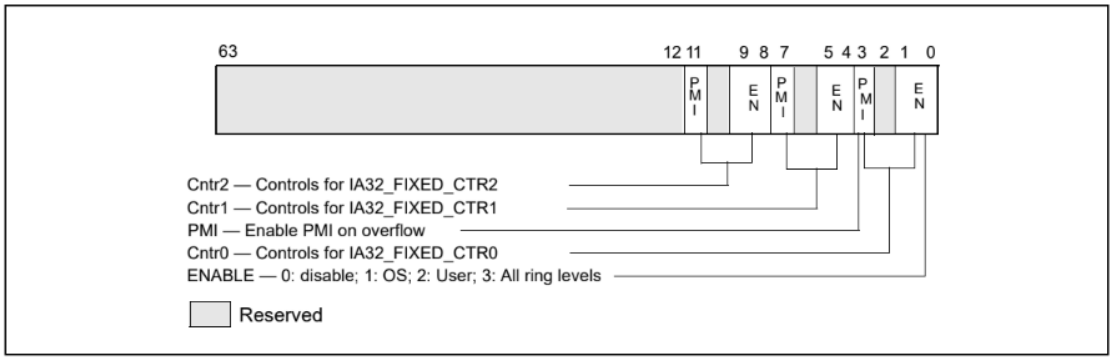

Figure 19-2. Layout of IA32_FIXED_CTR_CTRL MSR

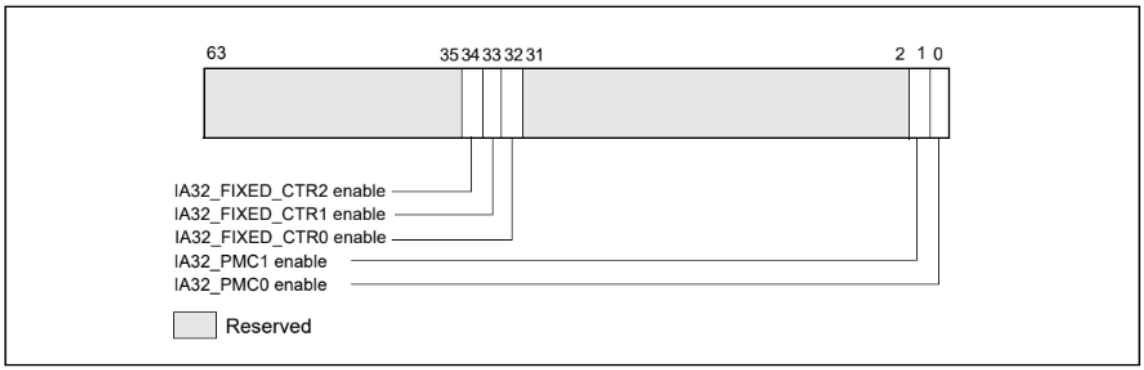

Figure 19-3. Layout of IA32_PERF_GLOBAL_CTRL MSR