Open-Source Register Reference For AMD Family 17h Processors

Legal Notices

List of Chapters

Table of Contents

List of Figures

List of Tables

1 Open Source Register Reference

The Open Source Register Reference (OSRR) is provided as an additional processor programming reference for Family 17h processors. This document provides details of various registers to aide in optimizing the performance and operation of AMD's 17h Family processors.

1.1 Intended Audience

This document provides targeted processor register definitions and associated design notes. It is intended to aide software designers and programmers involved in the development of operating system kernel modules and performance metric mechanisms.

The processor revision is specified by CPUID_Fn00000001_EAX (FamModStep) or CPUID_Fn80000001_EAX (FamModStepExt). This document uses a revision letter instead of specific model numbers. Where applicable, the processor stepping is indicated after the revision letter. All behavior marked with a revision letter apply to future revisions unless they are superseded by a change in a later revision. See the revision guide in 1.2 [Reference Documents] for additional information about revision determination.

1.2 Reference Documents

Table 1: Reference Documents Listing

Term	Description
docAPM1	AMD64 Architecture Programmer's Manual Volume 1: Application Programming, order# 24592.
docAPM2	AMD64 Architecture Programmer's Manual Volume 2: System Programming, order# 24593.
docAPM3	AMD64 Architecture Programmer's Manual Volume 3: Instruction-Set Reference, order# 24594.
docAPM4	AMD64 Architecture Programmer's Manual Volume 4: 128-Bit and 256-Bit Media Instructions, order# 26568.
docAPM5	AMD64 Architecture Programmer's Manual Volume 5: 64-Bit Media and x87 Floating-Point Instructions, order# 26569.
docACPI	Advanced Configuration and Power Interface (ACPI) Specification. http://www.acpi.info.
docAPML	Advanced Platform Management Link (APML) Specification, order #55719.
docIOMMU	AMD I/O Virtualization TechnologySpecification, order# 48882.
docI2C	I2C Bus Specification. http://www.nxp.com/documents/user_manual/UM10204.pdf
docJEDEC	JEDEC Standards. http://www.jedec.org.
docPCIe	PCI Express® Specification. http://www.pcisig.org.
docPCIlb	PCI Local Bus Specification. http://www.pcisig.org.
docRAS	RAS Feature Enablement for AMD Family 17h Models 00h-0Fh, order# 55987.
docRevG	Revision Guide for AMD Family 17h Models 00h-0Fh Processors, order# 55449.
docAM4	Socket AM4 Processor Functional Data Sheet, order# 55509.

1.2.1 Documentation Conventions

When referencing information found in external documents listed in Reference Documents, the "=>" operator is used. This notation represents the item to be searched for in the reference document. For example:

docExDoc => Header1 => Header2

is to have the reader use the search facility when opening referenced document "docExDoc" and search for "Header2". "Header2" may appear more than once in "docExDoc", therefore, referencing the one that follows "Header1". In that case, the easiest way to get to Header2 is to use the search to locate Header1, then again to locate "Header2".

1.3 Conventions

1.3.1 Numbering

- Binary numbers: Binary numbers are indicated either by appending a “b" at the end (e.g., 0110b) or by verilog syntax (e.g., 4'b110).
- Hexadecimal numbers: Hexadecimal numbers are indicated by appending an “h" to the end (e.g., 45F8h) or by verilog syntax (e.g., 16'h45F8).
- Decimal numbers: A number is decimal if not specified to be binary or hex.
- Exception: Physical register mnemonics are implied to be hex without the h suffix.
- Underscores in numbers: Underscores are used to break up numbers to make them more readable. They do not imply any operation (e.g., 0110_1100).

1.3.2 Arithmetic And Logical Operators

In this document, formulas generally follow Verilog conventions for logic equations.

Table 2: Arithmetic and Logical Operator Definitions

Operator	Definition
{}	Concatenation. Curly brackets are used to indicate a group of bits that are concatenated together. Each set of bits is separated by a comma (e.g., {Addr[3:2], Xlate[3:0]} represents a 6-bit values; the two MSBs are Addr[3:2] and the four LSBs are Xlate[3:0]).
|	Bitwise OR (e.g., 01b | 10b == 11b).
||	Logical OR (e.g., 01b || 10b == 1b). It treats a multi-bit operand as 1 if >= 1 and produces a 1-bit result.
&	Bitwise AND (e.g., 01b & 10b == 00b).
&&	Logical AND (e.g., 01b && 10b == 1b). It treats a multi-bit operand as 1 if >= 1 and produces a 1-bit result.
^	Bitwise exclusive-OR (e.g., 01b ^ 10b == 11b). Sometimes used as “raised to the power of" as well, as indicated by the context in which it is used (e.g., 2^2 == 4).
~	Bitwise NOT (also known as one's complement). (e.g., ~10b == 01b).
!	Logical NOT (e.g., !10b == 0b). It treats a multi-bit operand as 1 if >= 1 and produces a 1-bit result.
<, <=, >, >=, ==, !=	Relational. Less than, Less than or equal, greater, greater than or equal, equal, and not equal.
+, -, *, /, %	Arithmetic. Addition, subtraction, multiplication, division, and modulus.
<<	Bitwise left shift. Shift left first operand by the number of bits specified by the 2nd operand (e.g., 01b << 01b == 10b).
>>	Bitwise right shift. Shift right first operand by the number of bits specified by the 2nd operand (e.g., 10b >> 01b == 01b).
?:	Ternary conditional (e.g., condition ? value if true : value if false).

Table 3: Function Definitions

Term	**Description**
ABS	ABS(integer expression): Remove sign from signed value.
FLOOR	FLOOR(integer expression): Rounds real number down to nearest integer.
CEIL	CEIL(real expression): Rounds real number up to nearest integer.
MIN	MIN(integer expression list): Picks minimum integer or real value of comma separated list.
MAX	MAX(integer expression list): Picks maximum integer or real value of comma separated list.
COUNT	COUNT(integer expression): Returns the number of binary 1's in the integer.
ROUND	ROUND(real expression): Rounds to the nearest integer; halfway rounds away from zero.
UNIT	UNIT(register field reference): Input operand is a register field reference that contains a valid values table that defines a value with a unit (e.g., clocks, ns, ms, etc). This function takes the value in the register field and returns the value associated with the unit (e.g., If the field had a valid value definition where 1010b was defined as 5 ns). Then if the field had the value of 1010b, then UNIT() would return the value 5.
POW	POW(base, exponent): POW(x,y) returns the value x to the power of y.

1.3.2.1 Operator Precedence and Associativity

This document follows C operator precedence and associativity. The following table lists operator precedence (highest to lowest). Their associativity indicates in what order operators of equal precedence in an expression are applied. Parentheses are also used to group subexpressions to force a different precedence; such parenthetical expressions can be nested and are evaluated from inner to outer (e.g., “X = A || !B && C" is the same as “X = A || ((!B) && C)").

Table 4: Operator Precedence and Associativity

Operator	Description	Associativity
!, ~	Logical negation/bitwise complement	right to left
*, /, %	Multiplication/division/modulus	left to right
+, -	Addition/subtraction	left to right
<<, >>	Bitwise shift left, Bitwise shift right	left to right
< , <=, >, >=, ==, !=	Relational operators	left to right
&	Bitwise AND	left to right
^	Bitwise exclusive OR	left to right
|	Bitwise inclusive OR	left to right
&&	Logical AND	left to right
||	Logical OR	left to right
?:	Ternary conditional	right to left

1.3.3 Register Mnemonics

A register mnemonic is a short name that uniquely refers to a register, either all instances of that register, some instances, or a single instance.

Every register instance can be expressed in 2 forms, logical and physical, as defined below.

Table 5: Register Mnemonic Definitions

Term	**Description**
logical mnemonic	The register mnemonic format that describes the register functionally, what namespace to which the register belongs, a name for the register that connotes its function, and optionally, named parameters that indicate the different function of each instance (e.g., Link::Phy::PciDevVendIDF3). See 1.3.3.1 [Logical Mnemonic].

physical mnemonic	The register mnemonic that is formed based on the physical address used to access the register (e.g., D18F3x00). See 1.3.3.2 [Physical Mnemonic].

1.3.3.1 Logical Mnemonic

The logical mnemonic format consists of a register namespace, a register name, and optionally a register instance specifier (e.g., register namespace::register name register instance specifier).

For Unb::PciDevVendIDF3:
- The register namespace is Unb, which is the UNB IP register namespace.
- The register name is PciDevVendIDF3, which reads as PCICFG device and vendor ID in Function 3.
- There is no register instance specifier because there is just a single instance of this register.

For Dct::Phy::CalMisc2_dct[1:0]_chiplet[BCST,3:0]_pad[BCST,11:0]:
- The register namespace is Dct::Phy, which is the DCT PHY register namespace.
- The register name is CalMisc2, which reads as miscellaneous calibration register 2.
- The register instance specifier is _dct[1:0]_chiplet[BCST,3:0]_pad[BCST,11:0], which indicates that there are 2 DCTPHY instances, each IP for this register has 5 chiplets (0-3 and BCST), and for each chiplet 13 pads (0-11 and BCST). This register has 130 instances. (2*5*13)

Table 6: Logical Mnemonic Definitions

Term	Description
register namespace	A namespace for which the register name must be unique. A register namespace indicates to which IP it belongs and an IP may have multiple namespaces. A namespace is a string that supports a list of "::" separated names. The convention is for the list of names to be hierarchical, with the most significant name first and the least significant name last (e.g., Link::Phy::Rx is the RX component in the Link PHY).
register name	A name that cannotes the function of the register.
register instance specifier	The register instance specifier exists when there is more than one instance for a register. The register instance specifier consists of one or more register instance parameter specifier (e.g., The register instance specifier _dct[1:0]_chiplet[BCST,3:0]_pad[BCST,11:0] consists of 3 register instance parameter specifiers, _dct[1:0], _chiplet[BCST,3:0], and _pad[BCST,11:0]).
register instance parameter specifier	A register instance parameter specifier is of the form _register parameter name[register parameter value list] (e.g., The register instance parameter specifier _dct[1:0] has a register parameter name of dct (The DCT PHY instance name) and a register parameter value list of "1:0" or 2 instances of DCT PHY).
register parameter name	A register parameter name is the name of the number of instances at some level of the logical hierarchy (e.g., The register parameter name dct specifies how many instances of the DCT PHY exist).
register parameter value list	The register parameter value list is the logical name for each instance of the register parameter name (e.g., For _dct[1:0], there are 2 DCT PHY instances, with the logical names 0 and 1, but it should be noted that the logical names 0 and 1 can correspond to physical values other than 0 and 1). It is the purpose of the AddressMappingTable to map these register parameter values to physical address values for the register.

1.3.3.2 Physical Mnemonic

The physical register mnemonic format varies by the access method. The following table describes the supported physical register mnemonic formats.

Table 7: Physical Mnemonic Definitions

Term	Description
PCICFG	The PCICFG, or PCI defined configuration space, physical register mnemonic format is of the form DXFYxZZZ.
BAR	The BAR, or base address register, physical register mnemonic format is of the form PREFIXxZZZ.
MSR	The MSR, or x86 model specific register, physical register mnemonic format is of the form MSRXXXX_XXXX, where XXXX_XXXX is the hexadecimal MSR number. This space is accessed through x86 defined RDMSR and WRMSR instructions.
PMC	The PMC, or x86 performance monitor counter, physical register mnemonic format is any of the forms {PMCxXXX, L2IPMCxXXX, NBPMCxXXX}, where XXX is the performance monitor select.
CPUID	The CPUID, or x86 processor identification state, physical register mnemonic format is of the form CPUID FnXXXX_XXXX_EiX[_xYYY], where XXXX_XXXX is the hex value in the EAX and YYY is the hex value in ECX.

1.3.4 Register Format

A register is a group of register instances that have the same field format (same bit indices and field names).

1.3.4.1 A Register is a group of Register Instances

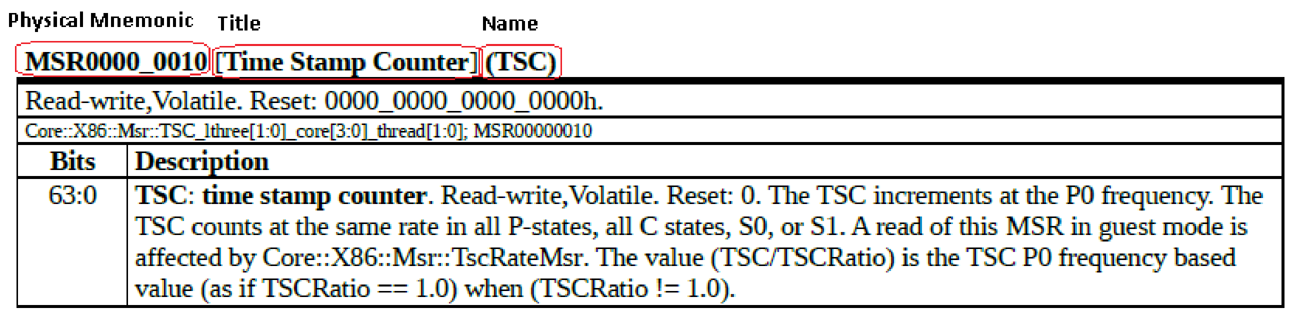
All instances of a register:

- Have the same:
  - Field bit indices and names
  - Field titles, descriptions, valid values.
  - Register title
  - Register description
- Fields may have different: (instance specific)
  - Access Type. See 1.3.4.10 [Field Access Type].
  - Reset. See 1.3.4.11 [Field Reset].
  - Init. See 1.3.4.12 [Field Initialization].
  - Check. See 1.3.4.13 [Field Check].

1.3.4.2 Register Physical Mnemonic, Title, and Name

A register definition is identified by a table that starts with a heavy bold line. The information above the bold line in order is:

1. The physical mnemonic of the first instance of the register
   - A register that has multiple instances, may have instances that have different access methods, each with it's own physical mnemonic format.
   - This text is not intended to represent the physical mnemonics of all instances of the register. It is only a visual aid to identify a register when scanning down a list, for readers that prefer to find registers by physical mnemonic.
   - The first instance physical mnemonic is the physical mnemonic of the first instance of the list of instances, sorted in lexical/alphabetical order.
2. The register title in brackets.
3. The register name in parenthesis.


Physical Mnemonic
Title
Name

MSR0000_0010 [Time Stamp Counter] (TSC)

Read-write,Volatile. Reset: 0000_0000_0000_0000h.

Core::X86::Msr::TSC_lthree[1:0]_core[3:0]_thread[1:0]; MSR00000010

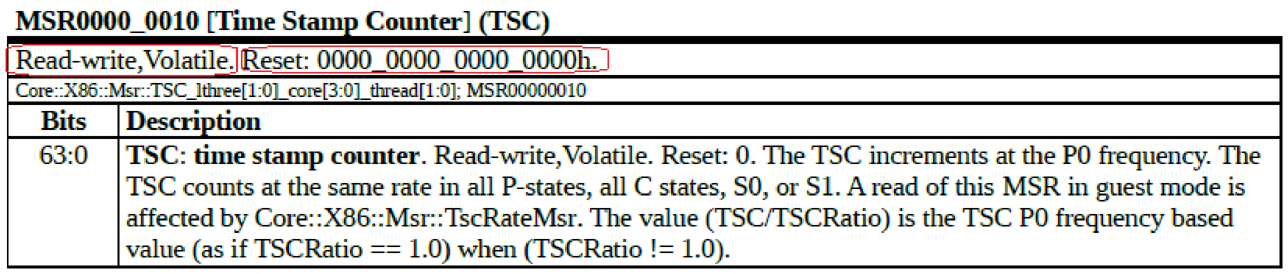
Bits	Description
63:0	**TSC: time stamp counter**. Read-write,Volatile. Reset: 0. The TSC increments at the P0 frequency. The TSC counts at the same rate in all P-states, all C states, S0, or S1. A read of this MSR in guest mode is affected by Core::X86::Msr::TscRateMsr. The value (TSC/TSCRatio) is the TSC P0 frequency based value (as if TSCRatio == 1.0) when (TSCRatio != 1.0).

Figure 1: Register Physical Mnemonic, Title, and Name

1.3.4.3 Full Width Register Attributes

The first line that follows the bold line contains the attributes that apply to all fields of the register. This row is rendered as a convenience to the reader and replicates content that exists in the register field.

- AccessType: If all non-reserved fields of a register have the same access type, then the access type is rendered in this row.
  - The supported access types are specified by 1.3.4.10 [Field Access Type].
  - The example figure shows that the access type "Read-write,Volatile" applies to all non-reserved fields of the register.
- Reset: If all non-reserved fields of a register have a constant reset, then the full width register reset is rendered in this row. The example figure shows the reset "0000000000000000h".
  - The value zero (0) is assumed for display purposes for all reserved fields.
- If none of the above content is rendered, then this row of the register is not rendered.



MSR0000_0010 [Time Stamp Counter] (TSC)

Read-write,Volatile. Reset: 0000_0000_0000_0000h.

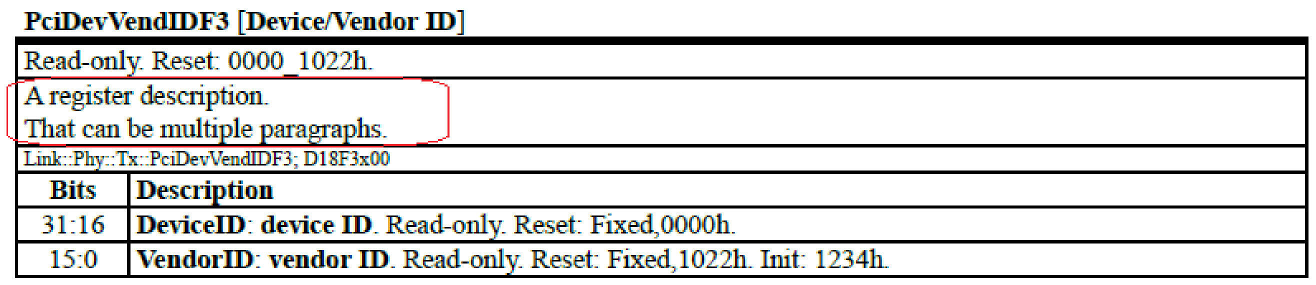
Core::X86::Msr::TSC_lthree[1:0]_core[3:0]_thread[1:0]; MSR00000010

Bits	Description
63:0	**TSC: time stamp counter**. Read-write,Volatile. Reset: 0. The TSC increments at the P0 frequency. The TSC counts at the same rate in all P-states, all C states, S0, or S1. A read of this MSR in guest mode is affected by Core::X86::Msr::TscRateMsr. The value (TSC/TSCRatio) is the TSC P0 frequency based value (as if TSCRatio == 1.0) when (TSCRatio != 1.0).

Figure 2: Full Width Register Attributes

1.3.4.4 Register Description

The register description is optional and appears after the "full width register attributes" row and before the "register instance table" rows. The register description can be one or more paragraphs.


PciDevVendIDF3 [Device/Vendor ID]

Read-only. Reset: 0000_1022h.	
A register description. That can be multiple paragraphs.	
Link::Phy::Tx::PciDevVendIDF3; D18F3x00	
Bits	**Description**
31:16	**DeviceID**: **device ID**. Read-only. Reset: Fixed,0000h.
15:0	**VendorID**: **vendor ID**. Read-only. Reset: Fixed,1022h. Init: 1234h.

Figure 3: Register Description

1.3.4.5 Register Instance Table

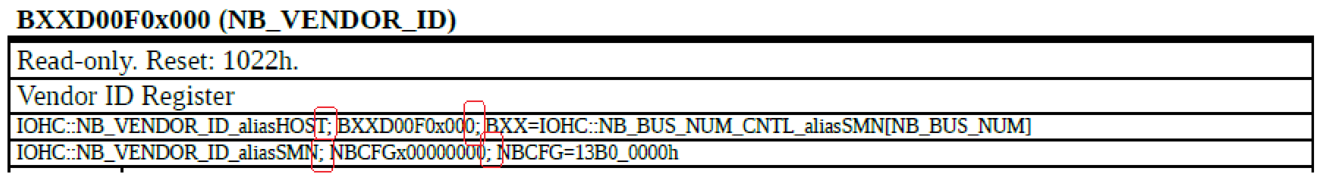
The zero or more rows of 8-pt font before the Bits/Description row is the register instance table.

The register instance table can generally be described as follows:
- Each row describes the access method of one or more register instances.
- If a row describes two or more instances, then the logical instance range, left to right, corresponds to the physical range, left to right.
- The absence of register instance rows indicates that the register exists for documentation purposes, and no access method is described for the register.

Because there are multiple access methods for all the registers, each of the following subsections describes an aspect of the register instance table in isolation.

1.3.4.5.1 Content Ordering in a Row

Content in a register instance table row is ordered as follows:
- The text up to the first semicolon is the logical mnemonic.
    - See 1.3.3.1 [Logical Mnemonic].
- The text after the first semicolon is the physical mnemonic.
    - See 1.3.3.2 [Physical Mnemonic].
- Optionally, content after the physical mnemonic provides additional information about the access method for the register instances in the row.


BXXD00F0x000 (NB_VENDOR_ID)

Read-only. Reset: 1022h.
Vendor ID Register
IOHC::NB_VENDOR_ID_aliasHOST; BXXD00F0x000; BXX=IOHC::NB_BUS_NUM_CNTL_aliasSMN[NB_BUS_NUM]
IOHC::NB_VENDOR_ID_aliasSMN; NBCFGx00000000; NBCFG=13B0_0000h

Figure 4: Register Instance Table: Content Ordering in a Row

1.3.4.5.2 Multiple Instances Per Row

Multiple instances in a row is represented by a single dimension “range" in the logical mnemonic and the physical mnemonic.

The single dimension order of instances is the same for both the logical and physical mnemonic. The first logical mnemonic is associated with the first physical mnemonic, so forth for the 2nd, up until the last.

- Brackets indicates a list, most significant to least significant.
- The “:" character indicates a continuous range between 2 values.
- The “," character separates non-contiguous values.
- There are some cases where more than one logical mnemonic maps to a single physical mnemonic.

Note that it is implied that the MSR {lthree,core,thread} parameters are not part of a range.

Example:
NAMESP::REGNAME_inst[BLOCK[5:0],BCST]_aliasHOST; FFF1x00000088_x[000[B:6]_0001,00000000]

- There are 7 instances.
- NAMESP is the namespace.
- 6 instances are represented by the sub-range 000[B:6]_0001.
- _instBCST corresponds to FFF1x00000088_x00000000.
- _inst BLOCK 0 corresponds to FFF1x00000088_x00060001.
- ...
- _inst BLOCK 5 corresponds to FFF1x00000088_x000B0001.

1.3.4.5.3 MSR Access Method

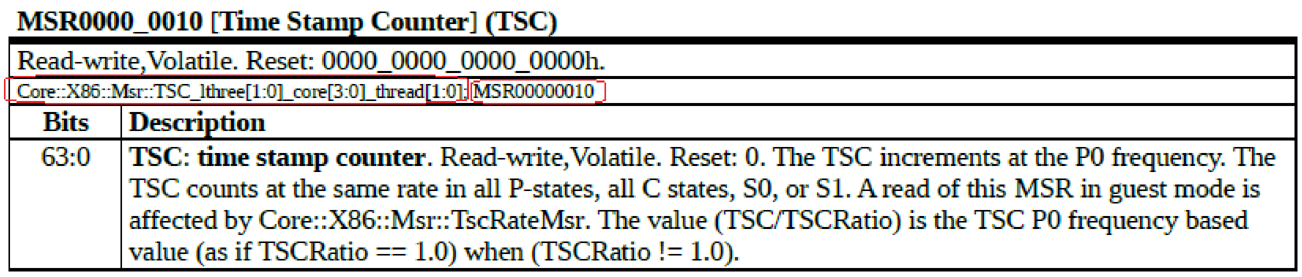
The MSR parameters {lthree,core,thread} are implied by the identity of the core on which the RDMSR/WRMSR is being executed, and therefore are not represented in the physical mnemonic.

MSRs that are:

- per-thread have the {lthree,core,thread} parameters.
- per-core do not have the thread parameter.
- per-L3 do not have the {core,thread} parameters.
- common to all L3's do not have the {lthree,core,thread} parameters.

1.3.4.5.3.1 MSR Per-Thread Example

An MSR that is per-thread has all three {lthree,core,thread} parameters and all instances have the same physical mnemonic.



MSR0000_0010 [Time Stamp Counter] (TSC)

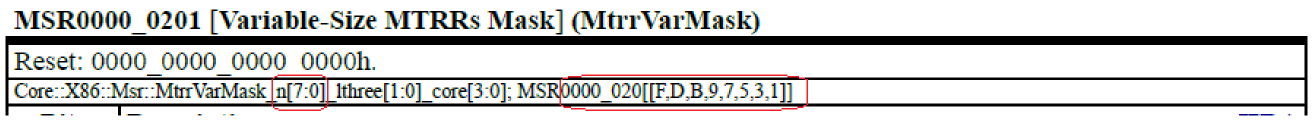
Read-write,Volatile. Reset: 0000_0000_0000_0000h.	
Core::X86::Msr::TSC_lthree[1:0]_core[3:0]_thread[1:0]; MSR00000010	
Bits	**Description**
63:0	**TSC: time stamp counter**. Read-write,Volatile. Reset: 0. The TSC increments at the P0 frequency. The TSC counts at the same rate in all P-states, all C states, S0, or S1. A read of this MSR in guest mode is affected by Core::X86::Msr::TscRateMsr. The value (TSC/TSCRatio) is the TSC P0 frequency based value (as if TSCRatio == 1.0) when (TSCRatio != 1.0).

Figure 5: Register Instance Table: MSR Example

1.3.4.5.3.2 MSR Range Example

An MSR can exist as a range for a parameter other than the {lthree,core,thread} parameters.

In the following example the n parameter is a range. The _n0 value corresponds to MSR0000_0201, and so on.



MSR0000_0201 [Variable-Size MTRRs Mask] (MtrrVarMask)

Reset: 0000_0000_0000_0000h.
Core::X86::Msr::MtrrVarMask_n[7:0]_lthree[1:0]_core[3:0]; MSR0000_020[[F,D,B,9,7,5,3,1]]

Figure 6: Register Instance Table: MSR Range Example

1.3.4.5.4 BAR Access Method

The BAR access method is indicated by a physical mnemonic that has the form PREFIXxNUMBER.

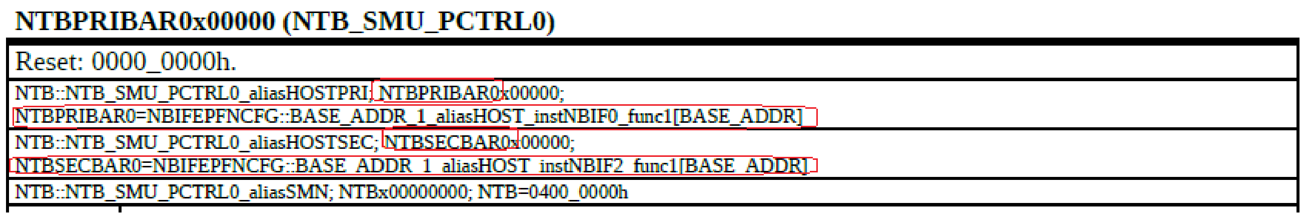
- Example: APICx0000. The BAR prefix is “APIC".

The BAR prefix represents either a constant or an expression that consists of a register reference.

1.3.4.5.4.1 BAR as a Register Reference

A relocatable BAR is when the base of an IP is not a constant.

- The prefix NTBPRIBAR0 represents the base of the IP, the value of which comes from the register NBIFEPFNCFG::BASE_ADDR_1_aliasHOST_instNBIF0_func1[BASE_ADDR].



NTBPRIBAR0x00000 (NTB_SMU_PCTRL0)

Reset: 0000_0000h.
NTB::NTB_SMU_PCTRL0_aliasHOSTPRI; NTBPRIBAR0x00000; NTBPRIBAR0=NBIFEPFNCFG::BASE_ADDR_1_aliasHOST_instNBIF0_func1[BASE_ADDR]
NTB::NTB_SMU_PCTRL0_aliasHOSTSEC; NTBSECBAR0x00000; NTBSECBAR0=NBIFEPFNCFG::BASE_ADDR_1_aliasHOST_instNBIF2_func1[BASE_ADDR]
NTB::NTB_SMU_PCTRL0_aliasSMN; NTBx00000000; NTB=0400_0000h

Figure 7: Register Instance Table: BAR as Register Reference

1.3.4.5.5 PCICFG Access Method

The PCICFG access method is indicated by a physical mnemonic that has the form DXXFXxNUMBER. There are 2 cases:

- Bus omitted and implied to be 00h.
- Bus represented as BXX and indicates that the bus is indicated by a register field.

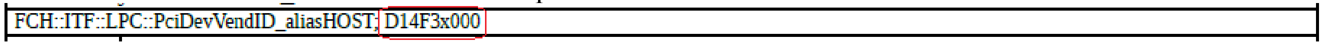
Example:

- Example: D18F000h. (The bus, when omitted, is implied to be 00h)
- Example: BXXD0F000h. (The bus as an expression that includes a register reference)

1.3.4.5.5.1 PCICFG Bus Implied to be 00h

Example:

- The absence of a B before the D14 implies that the bus is 0.



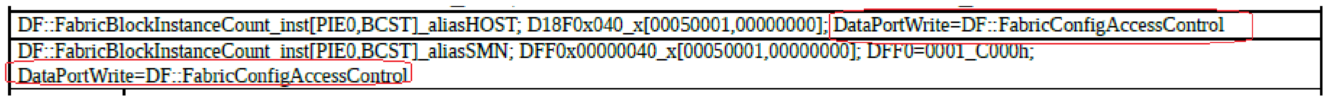
FCH::ITF::LPC::PciDevVendID_aliasHOST; D14F3x000

Figure 8: Register Instance Table: Bus Implied to be 00h

1.3.4.5.6 Data Port Access Method

A data port requires that the data port select be written before the register is accessed via the data port.

Example:
- The data port select value follows the "_x".
- The data port select register follows the "DataPortWrite=".



DF::FabricBlockInstanceCount_inst[PIE0,BCST]_aliasHOST; D18F0x040_x[00050001,00000000]; DataPortWrite=DF::FabricConfigAccessControl
DF::FabricBlockInstanceCount_inst[PIE0,BCST]_aliasSMN; DFF0x00000040_x[00050001,00000000]; DFF0=0001_C000h; DataPortWrite=DF::FabricConfigAccessControl

Figure 9: Register Instance Table: Data Port Select

1.3.4.6 Register Field Format

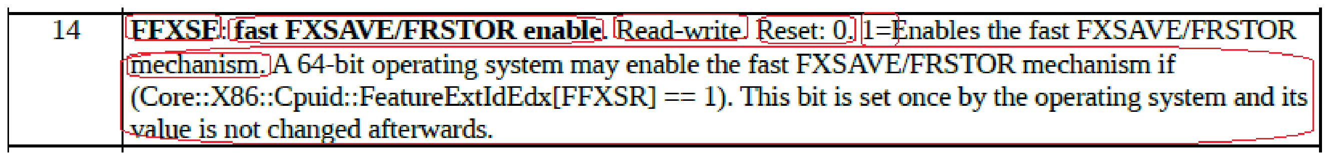
The register field definition are all rows that follow the Bits/Description row. Each field row represents the definition of a bit range, with the bit ranges ordered from most to least significant. There are 2 columns, with the left column defining the field bit range, and the right column containing the field definition.

There are 2 field definition formats, simple and complex. If the description can be described in the simple one paragraph format then the simple format is used, else the complex format is used.

1.3.4.7 Simple Register Field Format

The simple register format compresses all content into a single paragraph with the following implied order:
1. Field name (required)
   - Allowed to be Reserved. See 1.3.4.9 [Field Name is Reserved].
   - "FFXSE" in the example figure.
2. Field title
   - "fast FXSAVE/FRSTOR enable" in the example figure.
3. Field Access Type. See 1.3.4.10 [Field Access Type].
   - In the example figure the access type is "Read-write".
4. Field Reset. See 1.3.4.11 [Field Reset].
   - In the example figure the reset is warm reset and "0".
5. Field Init. See 1.3.4.12 [Field Initialization].
6. Field Check. See 1.3.4.13 [Field Check].
7. Field Valid Values, if the valid values are single bit (E.g. 0=, 1=). See 1.3.4.14 [Field Valid Values].
   - In the example figure the 1= definition begins with "Enables" and ends with "mechanism".
   - In the example figure there is no 0= definition.
8. Field description, if it is a single paragraph.
   - In the example figure the field description begins with "This is" and ends with "afterwards".

All fields that don't exist are omitted.



14	**FFXSE**: **fast FXSAVE/FRSTOR enable**. Read-write. Reset: 0. 1=Enables the fast FXSAVE/FRSTOR mechanism. A 64-bit operating system may enable the fast FXSAVE/FRSTOR mechanism if (Core::X86::Cpuid::FeatureExtIdEdx[FFXSR] == 1). This bit is set once by the operating system and its value is not changed afterwards.

Figure 10: Simple Register Field Example

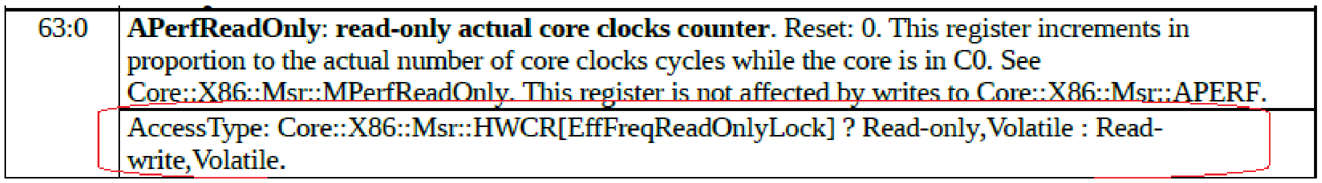
1.3.4.8 Complex Register Field Format

Content that can't be expressed in the single paragraph format is broken out to a separate sub-row (a definition column row).

Additional sub-rows are added in the following order:

1. Complex expression for {Reset,AccessType,Init,Check}.
2. Instance specific {Reset,AccessType,Init,Check} values.
3. Description, if more than 1 paragraph.
4. Valid values, if more than 0=/1=. Or a Valid bit table. (see figure)

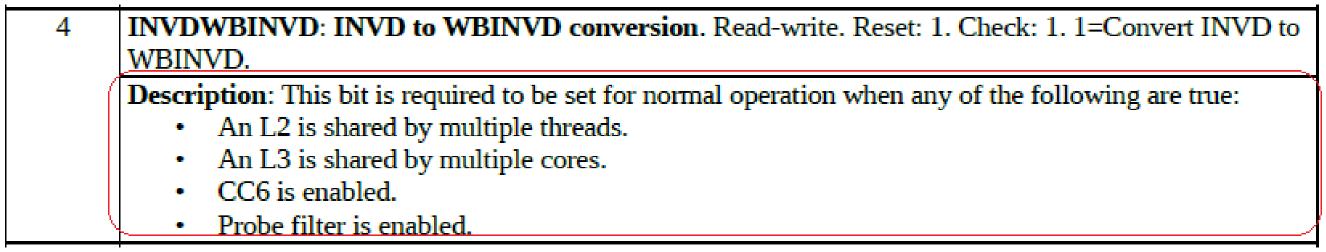
The following figure highlights a complex access type specification.



63:0	**APerfReadOnly**: **read-only actual core clocks counter**. Reset: 0. This register increments in proportion to the actual number of core clocks cycles while the core is in C0. See Core::X86::Msr::MPerfReadOnly. This register is not affected by writes to Core::X86::Msr::APERF.
	AccessType: Core::X86::Msr::HWCR[EffFreqReadOnlyLock] ? Read-only,Volatile : Read-write,Volatile.

Figure 11: Register Field Sub-Row for {Reset,AccessType,Init,Check}

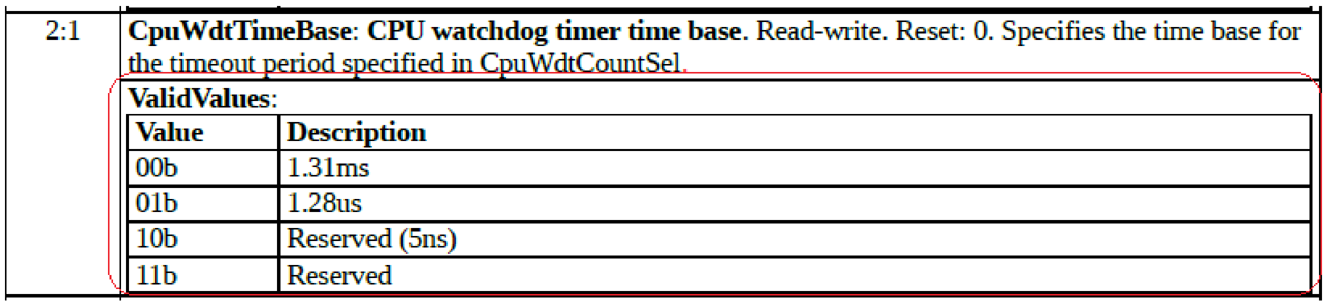
The following figure highlights a complex description specification.



4	**INVDWBINVD**: **INVD to WBINVD conversion**. Read-write. Reset: 1. Check: 1. 1=Convert INVD to WBINVD.
	Description: This bit is required to be set for normal operation when any of the following are true: • An L2 is shared by multiple threads. • An L3 is shared by multiple cores. • CC6 is enabled. • Probe filter is enabled.

Figure 12: Register Field Sub-Row for Description

The following figure highlights a complex valid value table, used either when the field is more than 1 bit or when the definition is more than a single sentence.



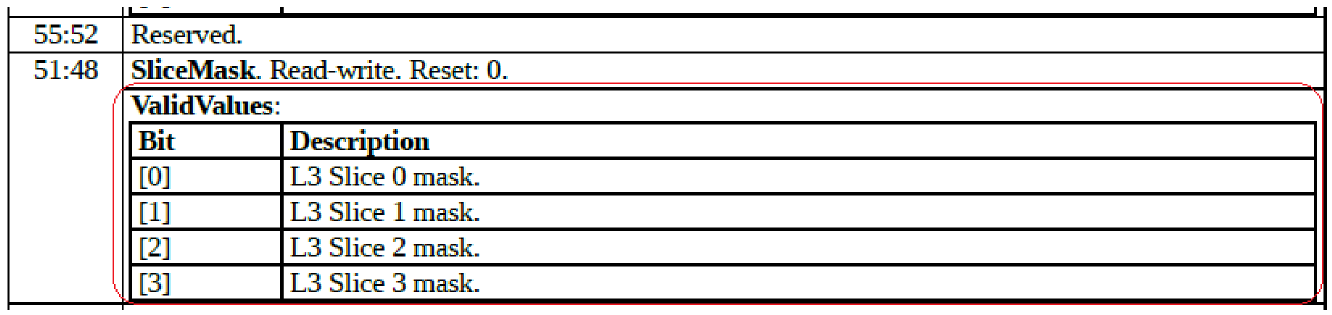
2:1 **CpuWdtTimeBase**: **CPU watchdog timer time base**. Read-write. Reset: 0. Specifies the time base for the timeout period specified in CpuWdtCountSel.

ValidValues:

Value	**Description**
00b	1.31ms
01b	1.28us
10b	Reserved (5ns)
11b	Reserved

Figure 13: Register Field Sub-Row for Valid Value Table

The following figure highlights a valid bit table which is used when each bit has a specific function.



55:52	Reserved.
51:48	**SliceMask**. Read-write. Reset: 0.
	ValidValues:

Bit	Description
[0]	L3 Slice 0 mask.
[1]	L3 Slice 1 mask.
[2]	L3 Slice 2 mask.
[3]	L3 Slice 3 mask.

Figure 14: Register Field Sub-Row for Valid Bit Table

1.3.4.9 Field Name is Reserved

When a register field name is Reserved, and it does not explicitly specify an access type, then the implied access type is "Reserved-write-as-read".

- The Reserved-write-as-read access type is:
    - Reads must not depend on the read value.
    - Writes must only write the value that was read.

1.3.4.10 Field Access Type

The AccessType keyword is optional and specifies the access type for a register field. The access type for a field is a comma separated list of the following access types.

Table 8: AccessType Definitions

Term	Description
Read-only	Readable; writes are ignored.
Read-write	Readable and writable.
Read	Readable; must be associated with one of the following {Write-once, Write-1-only, Write-1-to-clear, Error-on-write}.
Write-once	Capable of being written once; all subsequent writes have no effect. If not associated with Read, then reads are undefined.
Write-only	Writable. Reads are undefined.
Write-1-only	Writing a 1 sets to a 1; Writing a 0 has no effect. If not associated with Read, then reads are undefined.
Write-1-to-clear	Writing a 1 clears to a 0; Writing a 0 has no effect. If not associated with Read, then reads are undefined.
Write-0-only	Writing a 0 clears to a 0; Writing a 1 has no effect. If not associated with Read, then reads are undefined.
Error-on-read	Error occurs on read.
Error-on-write	Error occurs on write.
Error-on-write-0	Error occurs on bitwise write of 0.
Error-on-write-1	Error occurs on bitwise write of 1.
Inaccessible	Not readable or writable (e.g., Hide ? Inaccessible : Read-Write).
Configurable	Indicates that the access type is configurable as described by the documentation.
Unpredictable	The behavior of both reads and writes is unpredictable.
Reserved-write-as-1	Reads are undefined. Must always write 1.

Reserved-write-as-0	Reads are undefined. Must always write 0.
Volatile	Indicates that a register field value may be modified by hardware, firmware, or microcode when fetching the first instruction and/or might have read or write side effects. No read may depend on the results of a previous read and no write may be omitted based on the value of a previous read or write.

1.3.4.10.1 Conditional Access Type Expression

The ternary operator can be used to express an access type that is conditional on an expression that can contain any of the following:

- A register field value
- A constant
- A definition

1.3.4.11 Field Reset

The Reset keyword is optional and specifies the value for a register field at the time that hardware exits reset, before firmware initialization initiates.

Unless preceded by one of the following prefixes, the reset value is called warm reset and the value is applied at both warm and cold reset.

Table 9: Reset Type Definitions

Type	Description
Cold	Cold reset. The value is applied only at cold reset.
Fixed	The value applies at all time.

1.3.4.12 Field Initialization

The Init keyword is optional and specifies an initialization recommendation for a register field.

If present, then there is an optional prefix that specifies the owner of the initialization. See Table 10 [Init Type Definitions].

- Example: Init: BIOS,2'b0. //A initialization recommendation for a field to be programmed by BIOS.

Table 10: Init Type Definitions

Type	Description
BIOS	Initialized by AMD provided AMD Generic Encapsulated Software Architecture (AGESA™) x86 software.
SBIOS	Initialized by OEM or IBV provided x86 software, also called Platform BIOS.

1.3.4.13 Field Check

The Check keyword is optional and specifies the value that is recommended for firmware/software to write for a register field. It is a recommendation, not a requirement, and may not under all circumstances be what software programs.

1.3.4.14 Field Valid Values

A register can optionally have either a valid values table or a valid bit table:

- A valid values table specifies the definition for specific field values.
- A valid bit table specifies the definition for specific field bits.

2 Performance Monitor Counters

When selecting an event for which not all UnitMask bits are defined, the undefined UnitMask bits should be set to zero.

2.1 RDPMC Assignments

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

- The RDPMC[5:0] instruction accesses core events. See 2.3 [Core Performance Monitor Counters].
- The RDPMC[F:A] instruction accesses L3 cache events. See 2.4 [L3 Cache Performance Monitor Counters].

2.2 Large Increment per Cycle Events

The maximum increment for a regular performance event is 15 (i.e., a 4-bit event). However some event types can have a large increment every cycle (example: Core::X86::Pmc::Core::FpRetSseAvxOps).

An option is provided for merging a pair of even/odd performance monitors to acquire an accurate count. The even performance monitor is programmed with the desired event (example: Core::X86::Pmc::Core::FpRetSseAvxOps). The odd performance monitor is programmed with the event Core::X86::Pmc::Core::Merge with the enable bit (En) turned off. The performance monitor combines the count value to an 8-bit increment event and extends the counter to a 64-bit counter.

Software wanting to preload a value to a merged counter pair writes the high-order 16-bit value to the low-order 16 bits of the odd counter and then writes the low-order 48-bit value to the even counter. Reading the even counter of the merged counter pair returns the full 64-bit value.

If an even performance monitor is programmed with the event Core::X86::Pmc::Core::Merge the read results are undetermined. If an even performance monitor is programmed with a non-merge-able event (i.e., less then 5-bit event) while the corresponding odd performance monitor is programmed as Merge, the read results are undetermined. When discontinuing use of a merged counter pair, clear the Merge event from the odd performance monitor.

PMCxFFF [Merge] (Merge)

See 2.2 [Large Increment per Cycle Events].	
Core::X86::Pmc::Core::Merge; PMCxFFF	
Bits	**Description**
7:0	Reserved.

2.3 Core Performance Monitor Counters

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

2.3.1 Floating Point (FP) Events

PMCx000 [FPU Pipe Assignment] (FpuPipeAssignment)

Read-only. Reset: 00h.
The number of operations (uOps) and dual-pipe uOps dispatched to each of the 4 FPU execution pipelines. This event

reflects how busy the FPU pipelines are and may be used for workload characterization. This includes all operations performed by x87, MMX™, and SSE instructions, including moves. Each increment represents a one-cycle dispatch event. This event is a speculative event. (See Core::X86::Pmc::Core::ExRetMmxFpInstr). Since this event includes non-numeric operations it is not suitable for measuring MFLOPS.

Core::X86::Pmc::Core::FpuPipeAssignment; PMCx000

Bits	Description
7	**Dual3**: **Total number multi-pipe uOps assigned to Pipe 3**. Read-only. Reset: 0.
6	**Dual2**: **Total number multi-pipe uOps assigned to Pipe 2**. Read-only. Reset: 0.
5	**Dual1**: **Total number multi-pipe uOps assigned to Pipe 1**. Read-only. Reset: 0.
4	**Dual0**: **Total number multi-pipe uOps assigned to Pipe 0**. Read-only. Reset: 0.
3	**Total3**: **Total number uOps assigned to Pipe 3**. Read-only. Reset: 0.
2	**Total2**: **Total number uOps assigned to Pipe 2**. Read-only. Reset: 0.
1	**Total1**: **Total number uOps assigned to Pipe 1**. Read-only. Reset: 0.
0	**Total0**: **Total number uOps assigned to Pipe 0**. Read-only. Reset: 0.

PMCx001 [FP Scheduler Empty] (FpSchedEmpty)

This is a speculative event. The number of cycles in which the FPU scheduler is empty. Note that some Ops like FP loads bypass the scheduler. Invert this (Core::X86::Msr::PERF_CTL[Inv] == 1) to count cycles in which at least one FPU operation is present in the FPU.

Core::X86::Pmc::Core::FpSchedEmpty; PMCx001

Bits	Description
7:0	Reserved.

PMCx002 [Retired x87 Floating Point Operations] (FpRetx87FpOps)

Read-write. Reset: 00h.

The number of x87 floating-point Ops that have retired. The number of events logged per cycle can vary from 0 to 8.

Core::X86::Pmc::Core::FpRetx87FpOps; PMCx002

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

PMCx003 [Retired SSE/AVX Operations] (FpRetSseAvxOps)

Read-write. Reset: 00h.

This is a retire-based event. The number of retired SSE/AVX FLOPS. The number of events logged per cycle can vary from 0 to 64. This event can count above 15. See 2.2 [Large Increment per Cycle Events].

Core::X86::Pmc::Core::FpRetSseAvxOps; PMCx003

Bits	Description
7	**DpMultAddFlops**: **Double precision multiply-add FLOPS**. Read-write. Reset: 0. Multiply-add counts as 2 FLOPS.
6	**DpDivFlops**: **Double precision divide/square root FLOPS**. Read-write. Reset: 0.
5	**DpMultFlops**: **Double precision multiply FLOPS**. Read-write. Reset: 0.
4	**DpAddSubFlops**: **Double precision add/subtract FLOPS**. Read-write. Reset: 0.
3	**SpMultAddFlops**: **Single precision multiply-add FLOPS**. Read-write. Reset: 0. Multiply-add counts as 2 FLOPS.
2	**SpDivFlops**: **Single-precision divide/square root FLOPS**. Read-write. Reset: 0.
1	**SpMultFlops**: **Single-precision multiply FLOPS**. Read-write. Reset: 0.
0	**SpAddSubFlops**: **Single-precision add/subtract FLOPS**. Read-write. Reset: 0.

PMCx004 [Number of Move Elimination and Scalar Op Optimization] (FpNumMovElimScalOp)

Read-write. Reset: 00h.	
This is a dispatch based speculative event, and is useful for measuring the effectiveness of the Move elimination and Scalar code optimization schemes.	
Core::X86::Pmc::Core::FpNumMovElimScalOp; PMCx004	
Bits	**Description**
7:4	Reserved.
3	**Optimized**: **Number of Scalar Ops optimized**. Read-write. Reset: 0.
2	**OptPotential**: **Number of Ops that are candidates for optimization (have Z-bit either set or pass)**. Read-write. Reset: 0.
1	**SseMovOpsElim**: **Number of SSE Move Ops eliminated**. Read-write. Reset: 0.
0	**SseMovOps**: **Number of SSE Move Ops**. Read-write. Reset: 0.

PMCx005 [Retired Serializing Ops] (FpRetiredSerOps)

Read-write. Reset: 00h.	
The number of serializing Ops retired.	
Core::X86::Pmc::Core::FpRetiredSerOps; PMCx005	
Bits	**Description**
7:4	Reserved.
3	**X87CtrlRet**: **x87 control word mispredict traps due to mispredictions in RC or PC, or changes in mask bits**. Read-write. Reset: 0.
2	**X87BotRet**: **x87 bottom-executing uOps retired**. Read-write. Reset: 0.
1	**SseCtrlRet**: **SSE control word mispredict traps due to mispredictions in RC, FTZ or DAZ, or changes in mask bits**. Read-write. Reset: 0.
0	**SseBotRet**: **SSE bottom-executing uOps retired**. Read-write. Reset: 0.

2.3.2 LS Events

PMCx024 [Bad Status 2] (LsBadStatus2)

Read-write. Reset: 00h.	
Store To Load Interlock (STLI) are loads that were unable to complete because of a possible match with an older store, and the older store could not do STLF for some reason. There are a number of reasons why this occurs, and this perfmon organizes them into three major groups.	
Core::X86::Pmc::Core::LsBadStatus2; PMCx024	
Bits	**Description**
7:3	Reserved.
2	**StlfNoData**. Read-write. Reset: 0. The load is capable of forwarding from an older store (i.e. the address match/overlap between the load and the older store) was good and everything works from an address perspective, but the store's data has not been produced by EX or FP yet so it can't be forwarded.
1	**StliOther**. Read-write. Reset: 0. All the other reasons. The most common among these is that there is only a partial overlap between the store and the load, for example there's an 8B store to address A and a 16B load starting at address A. STLF can't be performed in this case because only some of the load's data is coming from the store, so the load gets StliOther. Another StliOther case is if the load hits a non-cacheable store that's sitting in the non-cacheable buffers (WCBs).
0	**StliNoState**. Read-write. Reset: 0. The STLF is validated using DC way instead of an address compare. The store that wants to STLF is required to be a DC hit and have a valid DC way. The STLF candidate store is chosen based on address bits 11:0 overlap, and the DC way of that store is compared to the way of the load. If the store is in a DC miss state, then it doesn't have a valid DC way and so cannot validate STLF. The load gets StliNoState and can't complete.

PMCx025 [Locks] (LsLocks)

Read-write. Reset: 00h.	
Unit Masks ORed.	
Core::X86::Pmc::Core::LsLocks; PMCx025	
Bits	**Description**
7:4	Reserved.
3	**SpecLockMapCommit**. Read-write. Reset: 0.
2	**SpecLock**. Read-write. Reset: 0.
1	**NonSpecLock**. Read-write. Reset: 0.
0	**BusLock**. Read-write. Reset: 0.

PMCx026 [Retired CLFLUSH Instructions] (LsRetClClush)

The number of retired CLFLUSH instructions. This is a non-speculative event.	
Core::X86::Pmc::Core::LsRetClClush; PMCx026	
Bits	**Description**
7:0	Reserved.

PMCx027 [Retired CPUID Instructions] (LsRetCpuid)

The number of CPUID instructions retired.	
Core::X86::Pmc::Core::LsRetCpuid; PMCx027	
Bits	**Description**
7:0	Reserved.

PMCx029 [LS Dispatch] (LsDispatch)

Read-write. Reset: 00h.	
Counts the number of operations dispatched to the LS unit. Unit Masks ADDed.	
Core::X86::Pmc::Core::LsDispatch; PMCx029	
Bits	**Description**
7:3	Reserved.
2	**LdStDispatch**: **Load-op-Stores**. Read-write. Reset: 0.
1	**StoreDispatch**. Read-write. Reset: 0.
0	**LdDispatch**. Read-write. Reset: 0.

PMCx02B [SMIs Received] (LsSmiRx)

Counts the number of SMIs received.	
Core::X86::Pmc::Core::LsSmiRx; PMCx02B	
Bits	**Description**
7:0	Reserved.

PMCx035 [Store to Load Forward] (LsSTLF)

Number of STLF hits.	
Core::X86::Pmc::Core::LsSTLF; PMCx035	
Bits	**Description**
7:0	Reserved.

PMCx037 [Store Commit Cancels 2] (LsStCommitCancel2)

Read-write. Reset: 00h.	
Core::X86::Pmc::Core::LsStCommitCancel2; PMCx037	
Bits	**Description**
7:1	Reserved.
0	**StCommitCancelWcbFull**. Read-write. Reset: 0.

PMCx040 [Data Cache Accesses] (LsDcAccesses)

The number of accesses to the data cache for load and store references. This may include certain microcode scratchpad accesses, although these are generally rare. Each increment represents an eight-byte access, although the instruction may only be accessing a portion of that. This event is a speculative event.
Core::X86::Pmc::Core::LsDcAccesses; PMCx040

Bits	Description
7:0	Reserved.

PMCx041 [MAB Allocation by Pipe] (LsMabAllocPipe)

Read-write. Reset: 00h.
Core::X86::Pmc::Core::LsMabAllocPipe; PMCx041

Bits	Description
7:5	Reserved.
4	**TlbPipeEarly**. Read-write. Reset: 0.
3	**HwPf**. Read-write. Reset: 0.
2	**TlbPipeLate**. Read-write. Reset: 0.
1	**StPipe**. Read-write. Reset: 0.
0	**DataPipe**. Read-write. Reset: 0.

PMCx043 [Data Cache Refills from System] (LsRefillsFromSys)

Read-write. Reset: 00h.
Demand Data Cache Fills by Data Source.
Core::X86::Pmc::Core::LsRefillsFromSys; PMCx043

Bits	Description
7	Reserved.
6	**LS_MABRESP_RMT_DRAM**. Read-write. Reset: 0. DRAM or IO from different die.
5	Reserved.
4	**LS_MABRESP_RMT_CACHE**. Read-write. Reset: 0. Hit in cache; Remote CCX and the address's Home Node is on a different die.
3	**LS_MABRESP_LCL_DRAM**. Read-write. Reset: 0. DRAM or IO from this thread's die.
2	Reserved.
1	**LS_MABRESP_LCL_CACHE**. Read-write. Reset: 0. Hit in cache; local CCX (not Local L2), or Remote CCX and the address's Home Node is on this thread's die.
0	**MABRESP_LCL_L2**. Read-write. Reset: 0. Local L2 hit.

PMCx045 [L1 DTLB Miss] (LsL1DTlbMiss)

Read-write. Reset: 00h.
Core::X86::Pmc::Core::LsL1DTlbMiss; PMCx045

Bits	Description
7	**TlbReload1GL2Miss**. Read-write. Reset: 0.
6	**TlbReload2ML2Miss**. Read-write. Reset: 0.
5	**TlbReload32KL2Miss**. Read-write. Reset: 0.
4	**TlbReload4KL2Miss**. Read-write. Reset: 0.
3	**TlbReload1GL2Hit**. Read-write. Reset: 0.
2	**TlbReload2ML2Hit**. Read-write. Reset: 0.
1	**TlbReload32KL2Hit**. Read-write. Reset: 0.
0	**TlbReload4KL2Hit**. Read-write. Reset: 0.

PMCx046 [Tablewalker allocation] (LsTablewalker)

Read-write. Reset: 00h.
Core::X86::Pmc::Core::LsTablewalker; PMCx046

Bits	Description

7:4	Reserved.
3	**PerfMonTablewalkAllocIside1**. Read-write. Reset: 0.
2	**PerfMonTablewalkAllocIside0**. Read-write. Reset: 0.
1	**PerfMonTablewalkAllocDside1**. Read-write. Reset: 0.
0	**PerfMonTablewalkAllocDside0**. Read-write. Reset: 0.

PMCx047 [Misaligned loads] (LsMisalAccesses)

Core::X86::Pmc::Core::LsMisalAccesses; PMCx047

Bits	Description
7:0	Reserved.

PMCx04B [Prefetch Instructions Dispatched] (LsPrefInstrDisp)

Read-write. Reset: 00h.

Software Prefetch Instructions Dispatched.

Core::X86::Pmc::Core::LsPrefInstrDisp; PMCx04B

Bits	Description
7:3	Reserved.
2	**PrefetchNTA**. Read-write. Reset: 0.
1	**StorePrefetchW**. Read-write. Reset: 0.
0	**LoadPrefetchW**: **Prefetch, Prefetch_T0_T1_T2**. Read-write. Reset: 0.

PMCx052 [Ineffective Software Prefetchs] (LsInefSwPref)

Read-write. Reset: 00h.

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

Core::X86::Pmc::Core::LsInefSwPref; PMCx052

Bits	Description
7:2	Reserved.
1	**MabMchCnt**. Read-write. Reset: 0.
0	**DataPipeSwPfDcHit**. Read-write. Reset: 0.

PMCx059 [Software Prefetch Data Cache Fills] (LsSwPfDcFills)

Read-write. Reset: 00h.

Software Prefetch Data Cache Fills by Data Source

Core::X86::Pmc::Core::LsSwPfDcFills; PMCx059

Bits	Description
7	Reserved.
6	**LS_MABRESP_RMT_DRAM**. Read-write. Reset: 0. DRAM or IO from different die.
5	Reserved.
4	**LS_MABRESP_RMT_CACHE**. Read-write. Reset: 0. Hit in cache; Remote CCX and the address's Home Node is on a different die.
3	**LS_MABRESP_LCL_DRAM**. Read-write. Reset: 0. DRAM or IO from this thread's die.
2	Reserved.
1	**LS_MABRESP_LCL_CACHE**. Read-write. Reset: 0. Hit in cache; local CCX (not Local L2), or Remote CCX and the address's Home Node is on this thread's die.
0	**MABRESP_LCL_L2**. Read-write. Reset: 0. Local L2 hit.

PMCx05A [Hardware Prefetch Data Cache Fills] (LsHwPfDcFills)

Read-write. Reset: 00h.

Hardware Prefetch Data Cache Fills by Data Source

Core::X86::Pmc::Core::LsHwPfDcFills; PMCx05A

Bits	Description
7	Reserved.

6	**LS_MABRESP_RMT_DRAM**. Read-write. Reset: 0. DRAM or IO from different die.
5	Reserved.
4	**LS_MABRESP_RMT_CACHE**. Read-write. Reset: 0. Hit in cache; Remote CCX and the address's Home Node is on a different die.
3	**LS_MABRESP_LCL_DRAM**. Read-write. Reset: 0. DRAM or IO from this thread's die.
2	Reserved.
1	**LS_MABRESP_LCL_CACHE**. Read-write. Reset: 0. Hit in cache; local CCX (not Local L2), or Remote CCX and the address's Home Node is on this thread's die.
0	**MABRESP_LCL_L2**. Read-write. Reset: 0. Local L2 hit.

PMCx05B [Table Walker Data Cache Fills by Data Source] (LsTwDcFills)

Read-write. Reset: 00h.

Core::X86::Pmc::Core::LsTwDcFills; PMCx05B

Bits	Description
7	Reserved.
6	**LS_MABRESP_RMT_DRAM**. Read-write. Reset: 0. DRAM or IO from different die.
5	Reserved.
4	**LS_MABRESP_RMT_CACHE**. Read-write. Reset: 0. Hit in cache; Remote CCX and the address's Home Node is on a different die.
3	**LS_MABRESP_LCL_DRAM**. Read-write. Reset: 0. DRAM or IO from this thread's die.
2	Reserved.
1	**LS_MABRESP_LCL_CACHE**. Read-write. Reset: 0. Hit in cache; local CCX (not Local L2), or Remote CCX and the address's Home Node is on this thread's die.
0	**MABRESP_LCL_L2**. Read-write. Reset: 0. Local L2 hit.

PMCx076 [Cycles not in Halt] (LsNotHaltedCyc)

Core::X86::Pmc::Core::LsNotHaltedCyc; PMCx076

Bits	Description
7:0	Reserved.

2.3.3 IC and BP Events

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

PMCx080 [32 Byte Instruction Cache Fetch] (IcFw32)

The number of 32B fetch windows transferred from IC pipe to DE instruction decoder (includes non-cacheable and cacheable fill responses).

Core::X86::Pmc::Core::IcFw32; PMCx080

Bits	Description
7:0	Reserved.

PMCx081 [32 Byte Instruction Cache Misses] (IcFw32Miss)

The number of 32B fetch windows tried to read the L1 IC and missed in the full tag.

Core::X86::Pmc::Core::IcFw32Miss; PMCx081

Bits	Description
7:0	Reserved.

PMCx082 [Instruction Cache Refills from L2] (IcCacheFillL2)

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

Core::X86::Pmc::Core::IcCacheFillL2; PMCx082

Bits	Description
7:0	Reserved.

PMCx083 [Instruction Cache Refills from System] (IcCacheFillSys)

The number of 64 byte instruction cache line fulfilled from system memory or another cache.	
Core::X86::Pmc::Core::IcCacheFillSys; PMCx083	
Bits	**Description**
7:0	Reserved.

PMCx084 [L1 ITLB Miss, L2 ITLB Hit] (BpL1TlbMissL2Hit)

The number of instruction fetches that miss in the L1 ITLB but hit in the L2 ITLB.	
Core::X86::Pmc::Core::BpL1TlbMissL2Hit; PMCx084	
Bits	**Description**
7:0	Reserved.

PMCx085 [L1 ITLB Miss, L2 ITLB Miss] (BpL1TlbMissL2Miss)

The number of instruction fetches that miss in both the L1 and L2 TLBs.	
Core::X86::Pmc::Core::BpL1TlbMissL2Miss; PMCx085	
Bits	**Description**
7:0	Reserved.

PMCx087 [Instruction Pipe Stall] (IcFetchStall)

Read-write. Reset: 00h.	
Core::X86::Pmc::Core::IcFetchStall; PMCx087	
Bits	**Description**
7:3	Reserved.
2	**IcStallAny**. Read-write. Reset: 0. IC pipe was stalled during this clock cycle for any reason (nothing valid in pipe ICM1).
1	**IcStallDqEmpty**. Read-write. Reset: 0. IC pipe was stalled during this clock cycle (including IC to OC fetches) due to DQ empty.
0	**IcStallBackPressure**. Read-write. Reset: 0. IC pipe was stalled during this clock cycle (including IC to OC fetches) due to back-pressure.

PMCx08A [L1 BTB Correction] (BpL1BTBCorrect)

Core::X86::Pmc::Core::BpL1BTBCorrect; PMCx08A	
Bits	**Description**
7:0	Reserved.

PMCx08B [L2 BTB Correction] (BpL2BTBCorrect)

Core::X86::Pmc::Core::BpL2BTBCorrect; PMCx08B	
Bits	**Description**
7:0	Reserved.

PMCx08C [Instruction Cache Lines Invalidated] (IcCacheInval)

Read-write. Reset: 00h.	
The number of instruction cache lines invalidated. A non-SMC event is CMC (cross modifying code), either from the other thread of the core or another core.	
Core::X86::Pmc::Core::IcCacheInval; PMCx08C	
Bits	**Description**
7:2	Reserved.
1	**L2InvalidatingProbe**. Read-write. Reset: 0. IC line invalidated due to L2 invalidating probe (external or LS).
0	**FillInvalidated**. Read-write. Reset: 0. IC line invalidated due to overwriting fill response.

PMCx099 [ITLB Reloads] (BpTlbRel)

The number of ITLB reload requests.
Core::X86::Pmc::Core::BpTlbRel; PMCx099

Bits	Description
7:0	Reserved.

PMCx28A [OC Mode Switch] (IcOcModeSwitch)

Read-write. Reset: 00h.	
Core::X86::Pmc::Core::IcOcModeSwitch; PMCx28A	
Bits	**Description**
7:2	Reserved.
1	**OcIcModeSwitch**: **OC to IC mode switch**. Read-write. Reset: 0.
0	**IcOcModeSwitch**: **IC to OC mode switch**. Read-write. Reset: 0.

2.3.4 DE Events

PMCx0AF [Dynamic Tokens Dispatch Stall Cycles 0] (DeDisDispatchTokenStalls0)

Read-write. Reset: 00h.	
Cycles where a dispatch group is valid but does not get dispatched due to a token stall.	
Core::X86::Pmc::Core::DeDisDispatchTokenStalls0; PMCx0AF	
Bits	**Description**
7	Reserved.
6	**RetireTokenStall**: **RETIRE Tokens unavailable**. Read-write. Reset: 0.
5	**AGSQTokenStall**: **AGSQ Tokens unavailable**. Read-write. Reset: 0.
4	**ALUTokenStall**: **ALU tokens total unavailable**. Read-write. Reset: 0.
3	**ALSQ3_0_TokenStall**. Read-write. Reset: 0.
2	**ALSQ3TokenStall**: **ALSQ 3 Tokens unavailable**. Read-write. Reset: 0.
1	**ALSQ2TokenStall**: **ALSQ 2 Tokens unavailable**. Read-write. Reset: 0.
0	**ALSQ1TokenStall**: **ALSQ 1 Tokens unavailable**. Read-write. Reset: 0.

2.3.5 EX (SC) Events

PMCx0C0 [Retired Instructions] (ExRetInstr)

Core::X86::Pmc::Core::ExRetInstr; PMCx0C0	
Bits	**Description**
7:0	Reserved.

PMCx0C1 [Retired Uops] (ExRetCops)

The number of uOps retired. This includes all processor activity (instructions, exceptions, interrupts, microcode assists, etc.). The number of events logged per cycle can vary from 0 to 4.	
Core::X86::Pmc::Core::ExRetCops; PMCx0C1	
Bits	**Description**
7:0	Reserved.

PMCx0C2 [Retired Branch Instructions] (ExRetBrn)

The number of branch instructions retired. This includes all types of architectural control flow changes, including exceptions and interrupts.	
Core::X86::Pmc::Core::ExRetBrn; PMCx0C2	
Bits	**Description**
7:0	Reserved.

PMCx0C3 [Retired Branch Instructions Mispredicted] (ExRetBrnMisp)

The number of branch instructions retired, of any type, that were not correctly predicted. This includes those for which

prediction is not attempted (far control transfers, exceptions and interrupts).	
Core::X86::Pmc::Core::ExRetBrnMisp; PMCx0C3	
Bits	**Description**
7:0	Reserved.

PMCx0C4 [Retired Taken Branch Instructions] (ExRetBrnTkn)

The number of taken branches that were retired. This includes all types of architectural control flow changes, including exceptions and interrupts.	
Core::X86::Pmc::Core::ExRetBrnTkn; PMCx0C4	
Bits	**Description**
7:0	Reserved.

PMCx0C5 [Retired Taken Branch Instructions Mispredicted] (ExRetBrnTknMisp)

The number of retired taken branch instructions that were mispredicted.	
Core::X86::Pmc::Core::ExRetBrnTknMisp; PMCx0C5	
Bits	**Description**
7:0	Reserved.

PMCx0C6 [Retired Far Control Transfers] (ExRetBrnFar)

The number of far control transfers retired including far call/jump/return, IRET, SYSCALL and SYSRET, plus exceptions and interrupts. Far control transfers are not subject to branch prediction.	
Core::X86::Pmc::Core::ExRetBrnFar; PMCx0C6	
Bits	**Description**
7:0	Reserved.

PMCx0C7 [Retired Branch Resyncs] (ExRetBrnResync)

The number of resync branches. These reflect pipeline restarts due to certain microcode assists and events such as writes to the active instruction stream, among other things. Each occurrence reflects a restart penalty similar to a branch mispredict. This is relatively rare.	
Core::X86::Pmc::Core::ExRetBrnResync; PMCx0C7	
Bits	**Description**
7:0	Reserved.

PMCx0C8 [Retired Near Returns] (ExRetNearRet)

The number of near return instructions (RET or RET Iw) retired.	
Core::X86::Pmc::Core::ExRetNearRet; PMCx0C8	
Bits	**Description**
7:0	Reserved.

PMCx0C9 [Retired Near Returns Mispredicted] (ExRetNearRetMispred)

The number of near returns retired that were not correctly predicted by the return address predictor. Each such mispredict incurs the same penalty as a mispredicted conditional branch instruction.	
Core::X86::Pmc::Core::ExRetNearRetMispred; PMCx0C9	
Bits	**Description**
7:0	Reserved.

PMCx0CA [Retired Indirect Branch Instructions Mispredicted] (ExRetBrnIndMisp)

Core::X86::Pmc::Core::ExRetBrnIndMisp; PMCx0CA	
Bits	**Description**
7:0	Reserved.

PMCx0CB [Retired MMX™/FP Instructions] (ExRetMmxFpInstr)

Read-write. Reset: 00h.

The number of MMX, SSE or x87 instructions retired. The UnitMask allows the selection of the individual classes of instructions as given in the table. Each increment represents one complete instruction. Since this event includes non-numeric instructions it is not suitable for measuring MFLOPS.	
Core::X86::Pmc::Core::ExRetMmxFpInstr; PMCx0CB	
Bits	**Description**
7:3	Reserved.
2	**SseInstr**. Read-write. Reset: 0. SSE instructions (SSE, SSE2, SSE3, SSSE3, SSE4A, SSE41, SSE42, AVX).
1	**MmxInstr**. Read-write. Reset: 0. MMX instructions.
0	**X87Instr**: **x87 instructions**. Read-write. Reset: 0.

PMCx0D1 [Retired Conditional Branch Instructions] (ExRetCond)

Core::X86::Pmc::Core::ExRetCond; PMCx0D1	
Bits	**Description**
7:0	Reserved.

PMCx0D3 [Div Cycles Busy count] (ExDivBusy)

Core::X86::Pmc::Core::ExDivBusy; PMCx0D3	
Bits	**Description**
7:0	Reserved.

PMCx0D4 [Div Op Count] (ExDivCount)

Core::X86::Pmc::Core::ExDivCount; PMCx0D4	
Bits	**Description**
7:0	Reserved.

PMCx1CF [Tagged IBS Ops] (ExTaggedIbsOps)

Read-write. Reset: 00h.	
Core::X86::Pmc::Core::ExTaggedIbsOps; PMCx1CF	
Bits	**Description**
7:3	Reserved.
2	**IbsCountRollover**. Read-write. Reset: 0. Number of times an op could not be tagged by IBS because of a previous tagged op that has not retired.
1	**IbsTaggedOpsRet**: **Number of Ops tagged by IBS that retired**. Read-write. Reset: 0.
0	**IbsTaggedOps**: **Number of Ops tagged by IBS**. Read-write. Reset: 0.

PMCx1D0 [Retired Fused Branch Instructions] (ExRetFusBrnchInst)

The number of fused retired branch instructions retired per cycle. The number of events logged per cycle can vary from 0 to 3.	
Core::X86::Pmc::Core::ExRetFusBrnchInst; PMCx1D0	
Bits	**Description**
7:0	Reserved.

2.3.6 L2 Cache Events.

PMCx060 [Requests to L2 Group1] (L2RequestG1)

Read-write. Reset: 00h.	
Core::X86::Pmc::Core::L2RequestG1; PMCx060	
Bits	**Description**
7	**RdBlkL**. Read-write. Reset: 0.
6	**RdBlkX**. Read-write. Reset: 0.
5	**LsRdBlkC_S**. Read-write. Reset: 0.

4	**CacheableIcRead**. Read-write. Reset: 0.
3	**ChangeToX**. Read-write. Reset: 0.
2	**PrefetchL2**. Read-write. Reset: 0. Assume core should also count these and allow the breakdown between H/W vs. S/W and LS vs. IC.
1	**L2HwPf**. Read-write. Reset: 0.
0	**OtherRequests**. Read-write. Reset: 0. Events covered by Core::X86::Pmc::Core::L2RequestG2.

PMCx061 [Requests to L2 Group2] (L2RequestG2)

Read-write. Reset: 00h.

Multi-events in that LS and IF requests can be received simultaneous.

Core::X86::Pmc::Core::L2RequestG2; PMCx061

Bits	Description
7	**Group1**. Read-write. Reset: 0. All Group 1 commands not in unit0.
6	**LsRdSized**. Read-write. Reset: 0. RdSized, RdSized32, RdSized64.
5	**LsRdSizedNC**. Read-write. Reset: 0. RdSizedNC, RdSized32NC, RdSized64NC.
4	**IcRdSized**. Read-write. Reset: 0.
3	**IcRdSizedNC**. Read-write. Reset: 0.
2	**SmcInval**. Read-write. Reset: 0.
1	**BusLocksOriginator**. Read-write. Reset: 0.
0	**BusLocksResponses**. Read-write. Reset: 0.

PMCx062 [L2 Latency] (L2Latency)

Read-write. Reset: 00h.

Total cycles spent waiting for L2 fills to complete from L3 or memory, divided by four. This may be used to calculate average latency by multiplying this count by four and then dividing by the total number of L2 fills (unit mask Core::X86::Pmc::Core::L2RequestG1 == FEh). Event counts are for both threads. To calculate average latency, the number of fills from both threads must be used.

Core::X86::Pmc::Core::L2Latency; PMCx062

Bits	Description
7:1	Reserved.
0	**L2CyclesWaitingOnFills**. Read-write. Reset: 0.

PMCx063 [LS to L2 WBC requests] (L2WbcReq)

Read-write. Reset: 00h.

Core::X86::Pmc::Core::L2WbcReq; PMCx063

Bits	Description
7	Reserved.
6	**WcbWrite**. Read-write. Reset: 0.
5	**WcbClose**. Read-write. Reset: 0.
4	**CacheLineFlush**. Read-write. Reset: 0.
3	**I_LineFlush**. Read-write. Reset: 0.
2	**ZeroByteStore**. Read-write. Reset: 0. This becomes WriteNoData at SDP; this count does not include DVM Sync Ops and bus locks which are counted in Core::X86::Pmc::Core::L2RequestG2.
1	**LocalIcClr**: **Local IC Clear**. Read-write. Reset: 0.
0	**CLZero**: **Cache Line Zero**. Read-write. Reset: 0.

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

Read-write. Reset: 00h.

This event does not count accesses to the L2 cache by the L2 prefetcher, but it does count accesses by the L1 prefetcher.

Core::X86::Pmc::Core::L2CacheReqStat; PMCx064

Bits	Description

7	**LsRdBlkCS**: **LS ReadBlock C/S Hit**. Read-write. Reset: 0.
6	**LsRdBlkLHitX**: **LS Read Block L Hit X**. Read-write. Reset: 0.
5	**LsRdBlkLHitS**: **LsRdBlkL Hit Shared**. Read-write. Reset: 0.
4	**LsRdBlkX**: **LsRdBlkX/ChgToX Hit X**. Read-write. Reset: 0. Count RdBlkX finding Shared as a Miss.
3	**LsRdBlkC**: **LS Read Block C S L X Change to X Miss**. Read-write. Reset: 0.
2	**IcFillHitX**: **IC Fill Hit Exclusive Stale**. Read-write. Reset: 0.
1	**IcFillHitS**: **IC Fill Hit Shared**. Read-write. Reset: 0.
0	**IcFillMiss**: **IC Fill Miss**. Read-write. Reset: 0.

PMCx06D [Cycles with fill pending from L2] (L2FillPending)

Read-write. Reset: 00h.	
Total cycles spent with one or more fill requests in flight from L2.	
Core::X86::Pmc::Core::L2FillPending; PMCx06D	
Bits	**Description**
7:1	Reserved.
0	**L2FillBusy**. Read-write. Reset: 0.

2.4 L3 Cache Performance Monitor Counters

This section provides the core performance counter events that may be selected through Core::X86::Msr::ChL3PmcCfg. See Core::X86::Msr::ChL3Pmc.

2.4.1 L3 Cache PMC Events

L3PMCx01 [L3 Cache Accesses] (L3RequestG1)

Read-write. Reset: 00h.	
Core::X86::Pmc::L3::L3RequestG1; L3PMCx01	
Bits	**Description**
7	**Caching**: **L3 cache accesses**. Read-write. Reset: 0.
6:0	Reserved.

L3PMCx06 [L3 Miss] (L3CombClstrState)

Read-write. Reset: 00h.	
Core::X86::Pmc::L3::L3CombClstrState; L3PMCx06	
Bits	**Description**
7:1	Reserved.
0	**RequestMiss**: **L3 miss**. Read-write. Reset: 0.

3 MSR Registers

See 1.3.3 [Register Mnemonics] for a description of the register naming convention. MSRs are accessed through x86 WRMSR and RDMSR instructions.

MSRC001_0130 [Guest Host Communication Block] (GHCB)

Read-write. Reset: 0000_0000_0000_0000h.	
If Core::X86::Msr::GHCB is accessed in hypervisor mode, #GP is generated.	
Core::X86::Msr::GHCB_lthree[1:0]_core[3:0]_thread[1:0]; MSRC0010130	
Bits	**Description**
63:0	**GHCBPA**. Read-write. Reset: 0. Guest physical address of GHCB.

MSRC001_0131 [SEV Status] (SEV_Status)

Read,Error-on-write. Reset: 0000_0000_0000_0000h.	
Core::X86::Msr::SEV_Status_lthree[1:0]_core[3:0]_thread[1:0]; MSRC0010131	
Bits	**Description**
63:2	Reserved.
1	**SevEsEnabled**. Read,Error-on-write. Reset: 0. 1=The guest was launched with the Sev-ES feature enabled in VMCB offset 90h.
0	**SevEnabled**. Read,Error-on-write. Reset: 0. 1=The guest was launched with SEV feature enabled in VMCB offset 90h.

List of Namespaces

Namespace	**Heading(s)**
Core::X86::Msr	3 [MSR Registers]
Core::X86::Pmc::Core	2.2 [Large Increment per Cycle Events] 2.3.1 [Floating Point (FP) Events] 2.3.2 [LS Events] 2.3.3 [IC and BP Events] 2.3.4 [DE Events] 2.3.5 [EX (SC) Events] 2.3.6 [L2 Cache Events.]
Core::X86::Pmc::L3	2.4.1 [L3 Cache PMC Events]

List of Definitions

ABS: ABS(integer expression): Remove sign from signed value.
BAR: The BAR, or base address register, physical register mnemonic format is of the form PREFIXxZZZ.
CEIL: CEIL(real expression): Rounds real number up to nearest integer.
Configurable: Indicates that the access type is configurable as described by the documentation.
COUNT: COUNT(integer expression): Returns the number of binary 1's in the integer.
CPUID: The CPUID, or x86 processor identification state, physical register mnemonic format is of the form CPUID FnXXXX_XXXX_EiX[_xYYY], where XXXX_XXXX is the hex value in the EAX and YYY is the hex value in ECX.
docACPI: Advanced Configuration and Power Interface (ACPI) Specification. http://www.acpi.info.
docAM4: Socket AM4 Processor Functional Data Sheet, order# 55509.
docAPM1: AMD64 Architecture Programmer's Manual Volume 1: Application Programming, order# 24592.
docAPM2: AMD64 Architecture Programmer's Manual Volume 2: System Programming, order# 24593.
docAPM3: AMD64 Architecture Programmer's Manual Volume 3: Instruction-Set Reference, order# 24594.
docAPM4: AMD64 Architecture Programmer's Manual Volume 4: 128-Bit and 256-Bit Media Instructions, order# 26568.
docAPM5: AMD64 Architecture Programmer's Manual Volume 5: 64-Bit Media and x87 Floating-Point Instructions, order# 26569.
docAPML: Advanced Platform Management Link (APML) Specification, order #55719.
docI2C: I2C Bus Specification. http://www.nxp.com/documents/user_manual/UM10204.pdf
docIOMMU: AMD I/O Virtualization TechnologySpecification, order# 48882.
docJEDEC: JEDEC Standards. http://www.jedec.org.
docPCIe: PCI Express® Specification. http://www.pcisig.org.
docPCIlb: PCI Local Bus Specification. http://www.pcisig.org.
docRAS: RAS Feature Enablement for AMD Family 17h Models 00h-0Fh, order# 55987.
docRevG: Revision Guide for AMD Family 17h Models 00h-0Fh Processors, order# 55449.
Error-on-read: Error occurs on read.
Error-on-write: Error occurs on write.
Error-on-write-0: Error occurs on bitwise write of 0.
Error-on-write-1: Error occurs on bitwise write of 1.
FLOOR: FLOOR(integer expression): Rounds real number down to nearest integer.
Inaccessible: Not readable or writable (e.g., Hide ? Inaccessible : Read-Write).
MAX: MAX(integer expression list): Picks maximum integer or real value of comma separated list.
MIN: MIN(integer expression list): Picks minimum integer or real value of comma separated list.
MSR: The MSR, or x86 model specific register, physical register mnemonic format is of the form MSRXXXX_XXXX, where XXXX_XXXX is the hexadecimal MSR number. This space is accessed through x86 defined RDMSR and WRMSR instructions.
PCICFG: The PCICFG, or PCI defined configuration space, physical register mnemonic format is of the form DXFYxZZZ.
PMC: The PMC, or x86 performance monitor counter, physical register mnemonic format is any of the forms {PMCxXXX, L2IPMCxXXX, NBPMCxXXX}, where XXX is the performance monitor select.
POW: POW(base, exponent): POW(x,y) returns the value x to the power of y.
Reserved-write-as-0: Reads are undefined. Must always write 0.
Reserved-write-as-1: Reads are undefined. Must always write 1.
ROUND: ROUND(real expression): Rounds to the nearest integer; halfway rounds away from zero.
UNIT: UNIT(register field reference): Input operand is a register field reference that contains a valid values table that defines a value with a unit (e.g., clocks, ns, ms, etc). This function takes the value in the register field and returns the value associated with the unit (e.g., If the field had a valid value definition where 1010b was defined as 5 ns). Then if the field had the value of 1010b, then UNIT() would return the value 5.
Unpredictable: The behavior of both reads and writes is unpredictable.
Volatile: Indicates that a register field value may be modified by hardware, firmware, or microcode when fetching the first instruction and/or might have read or write side effects. No read may depend on the results of a previous read and no write may be omitted based on the value of a previous read or write.
Write-0-only: Writing a 0 clears to a 0; Writing a 1 has no effect. If not associated with Read, then reads are undefined.
Write-1-only: Writing a 1 sets to a 1; Writing a 0 has no effect. If not associated with Read, then reads are undefined.
Write-1-to-clear: Writing a 1 clears to a 0; Writing a 0 has no effect. If not associated with Read, then reads are undefined.
Write-once: Capable of being written once; all subsequent writes have no effect. If not associated with Read, then reads are undefined.

Memory Map - MSR

Physical Mnemonic	**Namespace**
C001_0130...C001_0131	Core::X86::Msr