Performance Monitoring

Chris Dahnken

Intel SSG EMEA HPCTC

Legal Disclaimer

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Optimization Notice

Optimization Notice

Intel® compilers, associated libraries and associated development tools may include or utilize options that optimize for instruction sets that are available in both Intel® and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel compilers, including some that are not specific to Intel micro-architecture, are reserved for Intel microprocessors. For a detailed description of Intel compiler options, including the instruction sets and specific microprocessors they implicate, please refer to the “Intel® Compiler User and Reference Guides” under “Compiler Options." Many library routines that are part of Intel® compiler products are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel® compiler products offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

Intel® compilers, associated libraries and associated development tools may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (Intel® SSSE3) instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors.

While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel® and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving to offer the best performance of any compiler or library; please let us know if you find we do not.

Notice revision #20101101

Performance Monitoring

The Performance Monitoring Unit (PMU)

Ways to measure performance

- time – the ultimate measure!
- gprof – break down execution time on a functional level. Careful, changes execution time!
- counter monitoring – hardware supported measurement that allows to determine which hardware component or functional unit is under stress or starves. Doesn't change exec time!

Performance Counter Monitoring

- Performance counter monitoring is a HW supported method count events appearing in the hardware
- Events are situations the HW designers allow us to see. E.g. the CPU could experience “I tried to load data, but it was not in the last level cache” (aka a LLC cache miss). This is an event that can be measured.

Command line perf monitoring

Architectural perf events

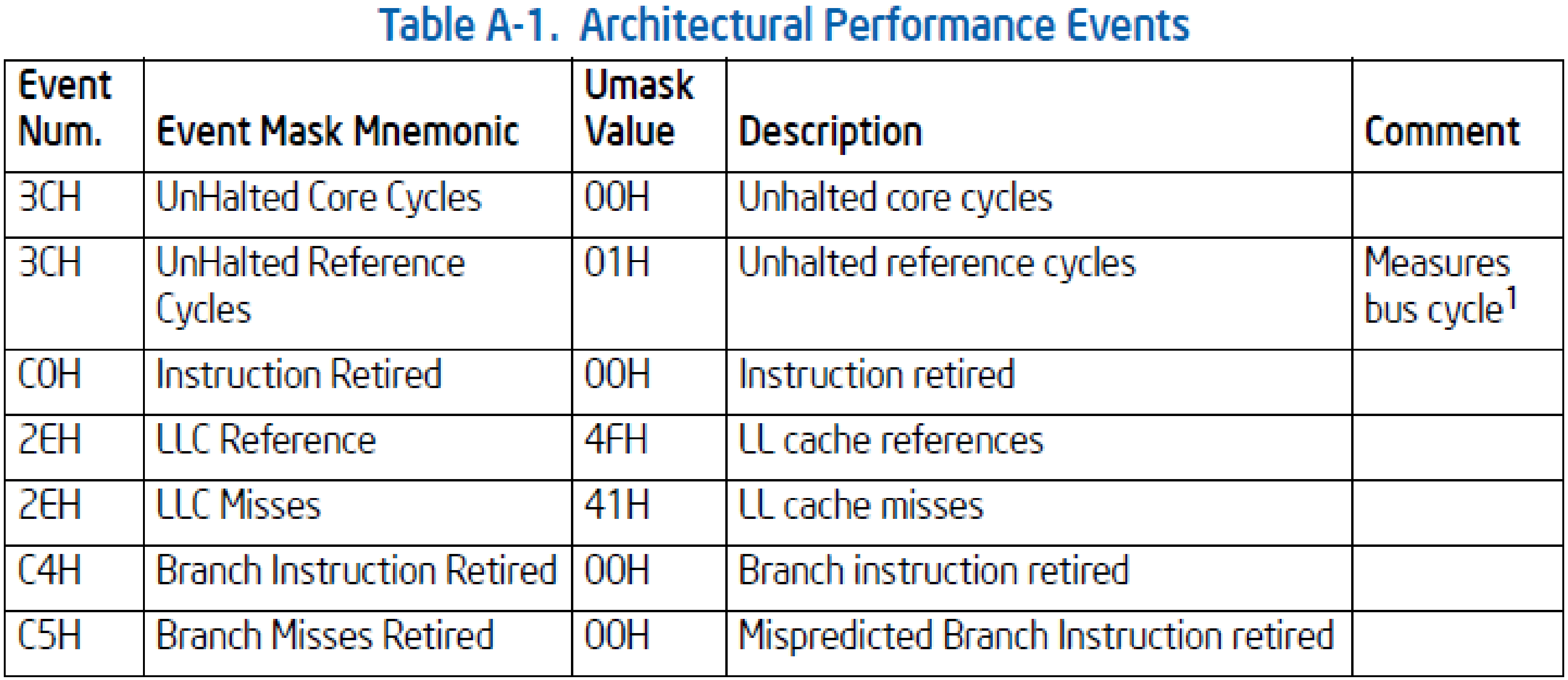


Table A-1. Architectural Performance Events

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]
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	

There are another ~1000 non-architectural performance events!

Performance Counter Monitoring

- There are
  - 4 freely programmable
  - 3 fix function

  performance counter registers on current Intel CPUs
- There are 4 configuration registers, that let us specify which events are counted in the programmable counters
- There is 1 configuration register that lets us enable and disable performance monitoring.

Performance Counter Monitoring

Just in case: What is a register?

- A register is a series of latches (HW bits) with 64 bit width (some might differ, e.g. SSE) the CPU can access very quickly

- A Model Specific Register (MSR) is a register that is not guaranteed to be there in the next CPU!

Performance Counter Monitoring

How can I influence/change/write/read MSRs?

- MSRs can only be written in Ring 0 (supervisor mode). You can read MSRs as user.

You need to be

- root/sudoer to write

or

- a driver that does the access for you

Performance Counter Monitoring

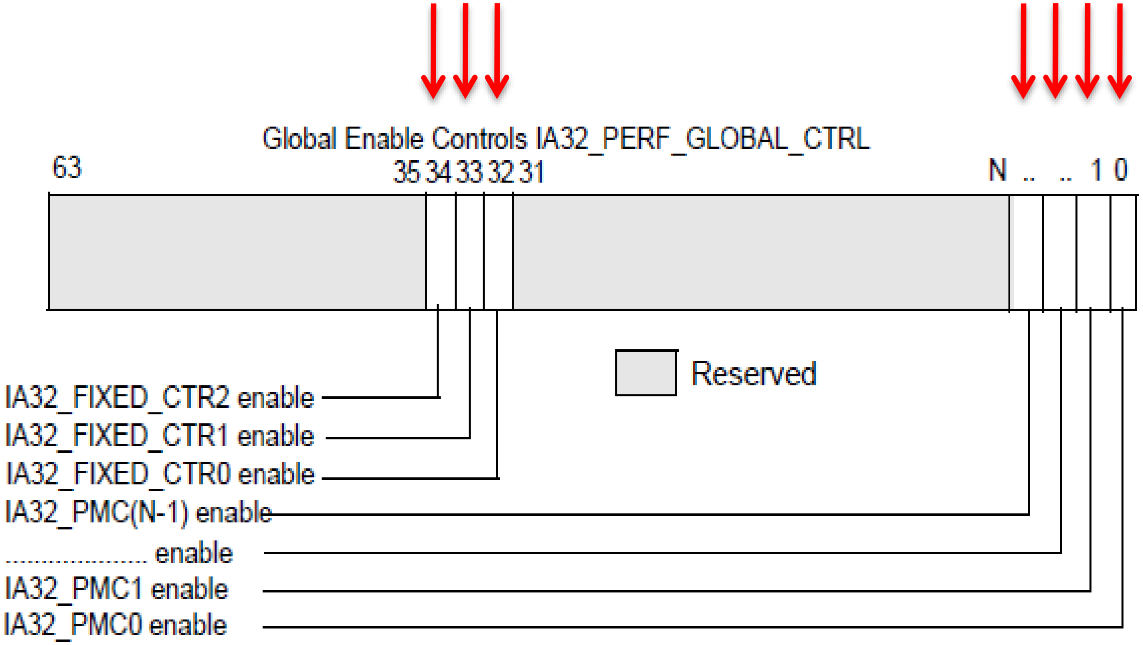
Software to read and write MSRs (linux)

- msr-tools package: rdmsr, wrmsr, msr.ko
- perf (linux system tool)
- Intel Vtune, Amplifier XE, PTU (http://software.intel.com/en-us/articles/intel-vtune-amplifier-xe/)
- Oprofile (http://oprofile.sourceforge.net/news/)
- Likwid (http://code.google.com/p/likwid/)

PMU MSRs

Control Registers

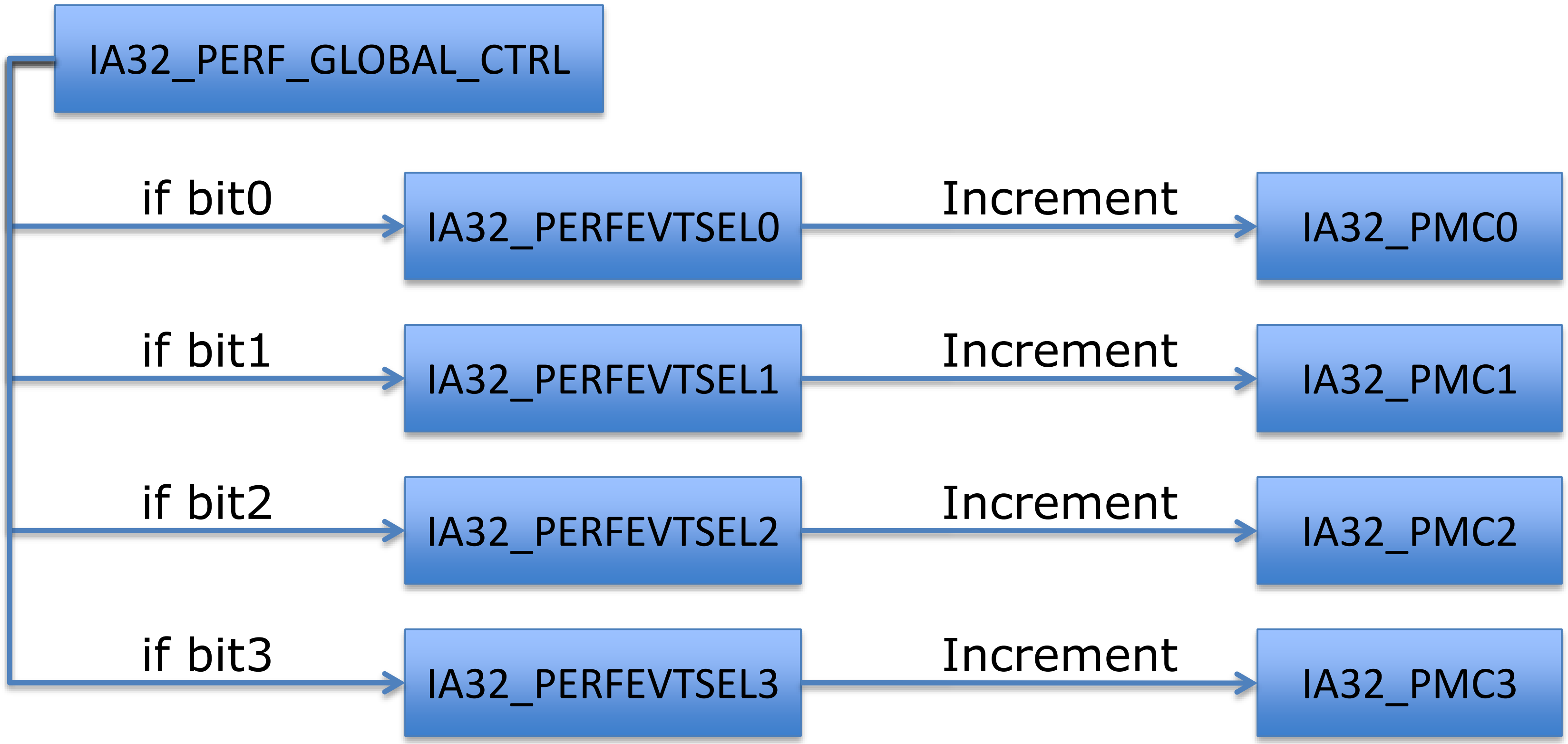
- IA32_PERF_GLOBAL_CTRL (0x38F/911)



```
wrmsr 911 –d 30064771087
```

PMU MSRs

General purpose performance counters



IA32_PERF_GLOBAL_CTRL

if bit0 → IA32_PERFEVTSEL0 → Increment → IA32_PMC0

if bit1 → IA32_PERFEVTSEL1 → Increment → IA32_PMC1

if bit2 → IA32_PERFEVTSEL2 → Increment → IA32_PMC2

if bit3 → IA32_PERFEVTSEL3 → Increment → IA32_PMC3

IA32_PERFEVTSELx specifies the number of the event to be obeserved (and a number of modifiers)

PMU MSRs

Selecting events

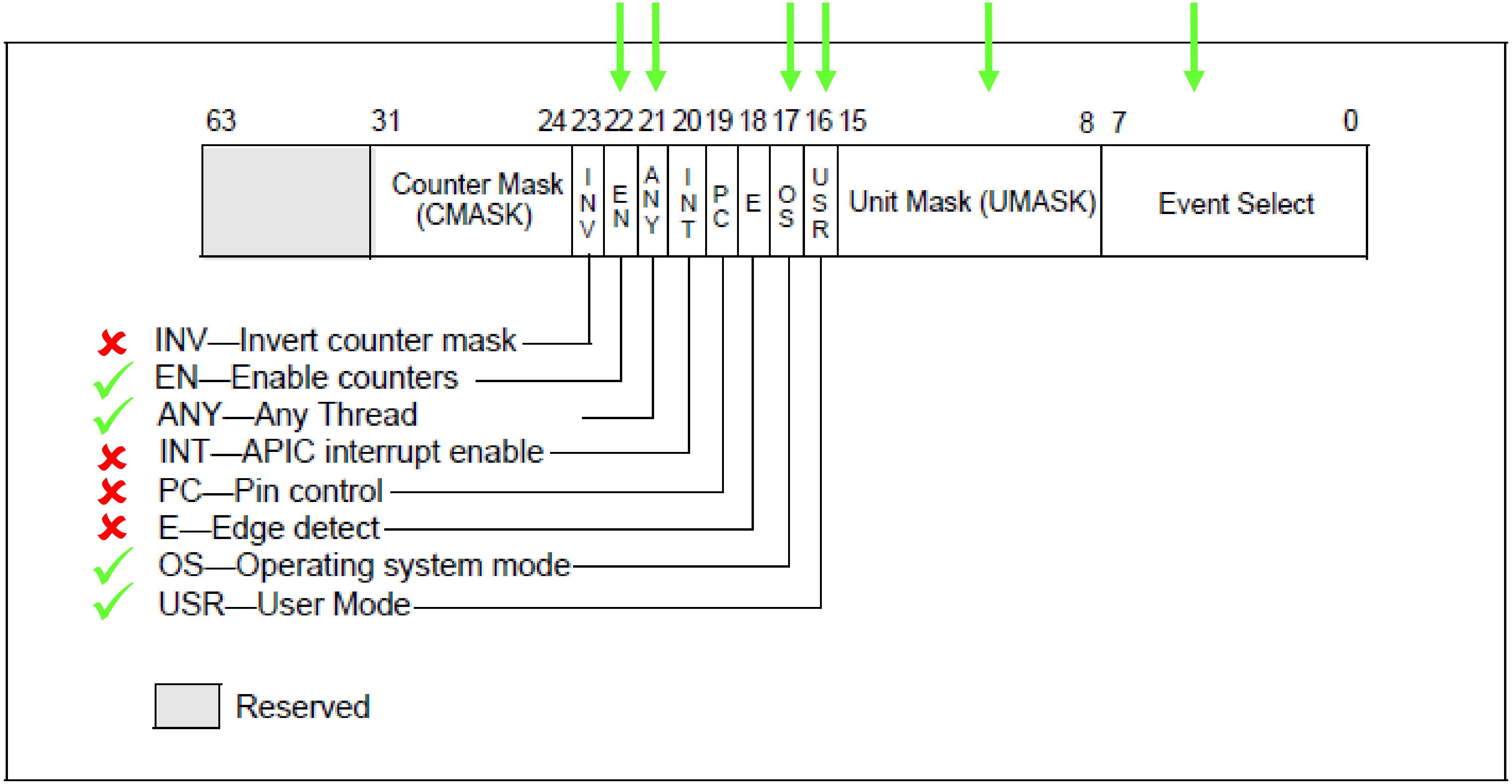
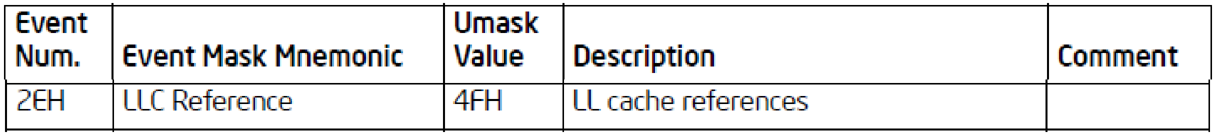


Figure 30-6. Layout of IA32_PERFEVTSELx MSRs Supporting Architectural Performance Monitoring Version 3

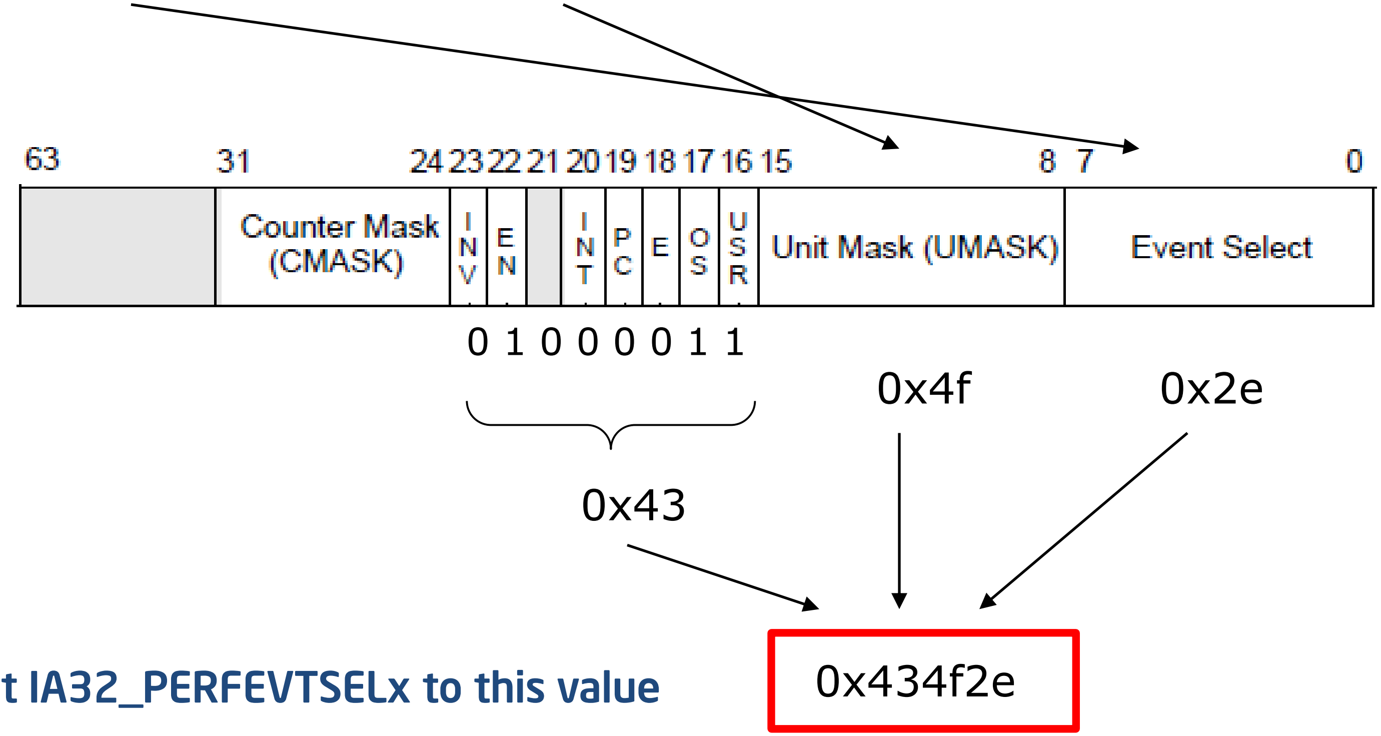
PMU MSRs

Selecting general purpose events

Intel® 64 and IA-32 Architectures Software Developer's Manual
Volume 3B: System Programming Guide, Part 2



Event Num.	Event Mask Mnemonic	Umask Value	Description	Comment
2EH	LLC Reference	4FH	LL cache references	



Set IA32_PERFEVTSELx to this value

PMU MSRs

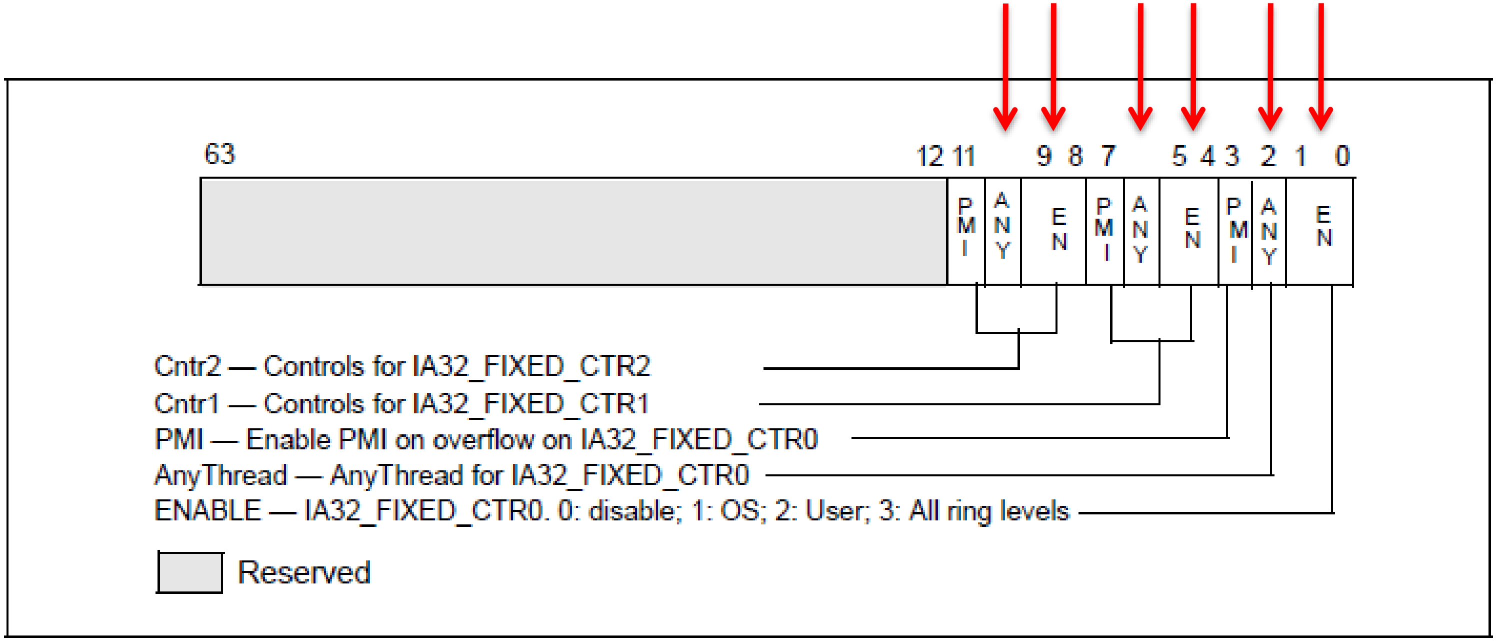
Fixed function performance counters

Additionally to the 4 freely programmable performance counters Nehalem (and later CPUs) offer three fixed function performance counters that will observe

- Instructions Retired
- CPU Cycles
- Reference CPU Cycles

PMU MSRs

Fixed function performance counters



Cntr2 — Controls for IA32_FIXED_CTR2

Cntr1 — Controls for IA32_FIXED_CTR1

PMI — Enable PMI on overflow on IA32_FIXED_CTR0

AnyThread — AnyThread for IA32_FIXED_CTR0

ENABLE — IA32_FIXED_CTR0. 0: disable; 1: OS; 2: User; 3: All ring levels

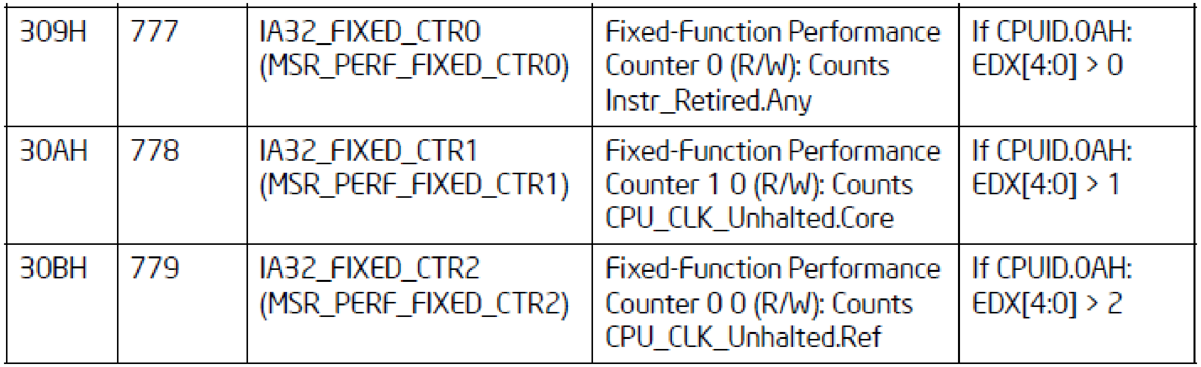
▢ Reserved

Figure 30-7. Layout of IA32_FIXED_CTR_CTRL MSR Supporting Architectural Performance Monitoring Version 3

PMU MSRs

Fixed function performance counters

- Event counts can now be seen in
- IA32_FIXED_CTR0: Instructions Retired
- IA32_FIXED_CTR1: Unhalted CPU Cycles
- IA32_FIXED_CTR2: Reference CPU Cycles



309H	777	IA32_FIXED_CTR0 (MSR_PERF_FIXED_CTR0)	Fixed-Function Performance Counter 0 (R/W): Counts Instr_Retired.Any	If CPUID.0AH: EDX[4:0] > 0
30AH	778	IA32_FIXED_CTR1 (MSR_PERF_FIXED_CTR1)	Fixed-Function Performance Counter 1 0 (R/W): Counts CPU_CLK_Unhalted.Core	If CPUID.0AH: EDX[4:0] > 1
30BH	779	IA32_FIXED_CTR2 (MSR_PERF_FIXED_CTR2)	Fixed-Function Performance Counter 0 0 (R/W): Counts CPU_CLK_Unhalted.Ref	If CPUID.0AH: EDX[4:0] > 2

PMU MSRs

Summary

- Intel CPUs can monitor particular events during execution
- 4 freely programmable counters and 3 fixed function counters measure the occurrence of the given events at the same time
- In order observe event counts
  - Enable event monitoring in the control register
  - Program the event into the event select register
  - Reset/Read the count from the counter register
- READ: 253669 - Intel 64 and IA-32 Architectures Software Developers Manual Volume 3B System Programming Guide Part 2

intel®

Software

Appendix

Command line performance monitoring

Command line perf monitoring

- With rdmsr and wrmsr we can now easily do performance monitoring in a lightweight way.
- Before going on to feature burden apps like Vtune, let’s now do a quick performance analysis directly from the command line

Command line perf monitoring

Architectural perf events

- Intel doesn’t guarantee the consistency of the performance monitoring unit
- Events can change with each model
- There are a number of events that Intel guarantees always to be present
- These are called Architectural performance events

Command line perf monitoring

Architectural perf events

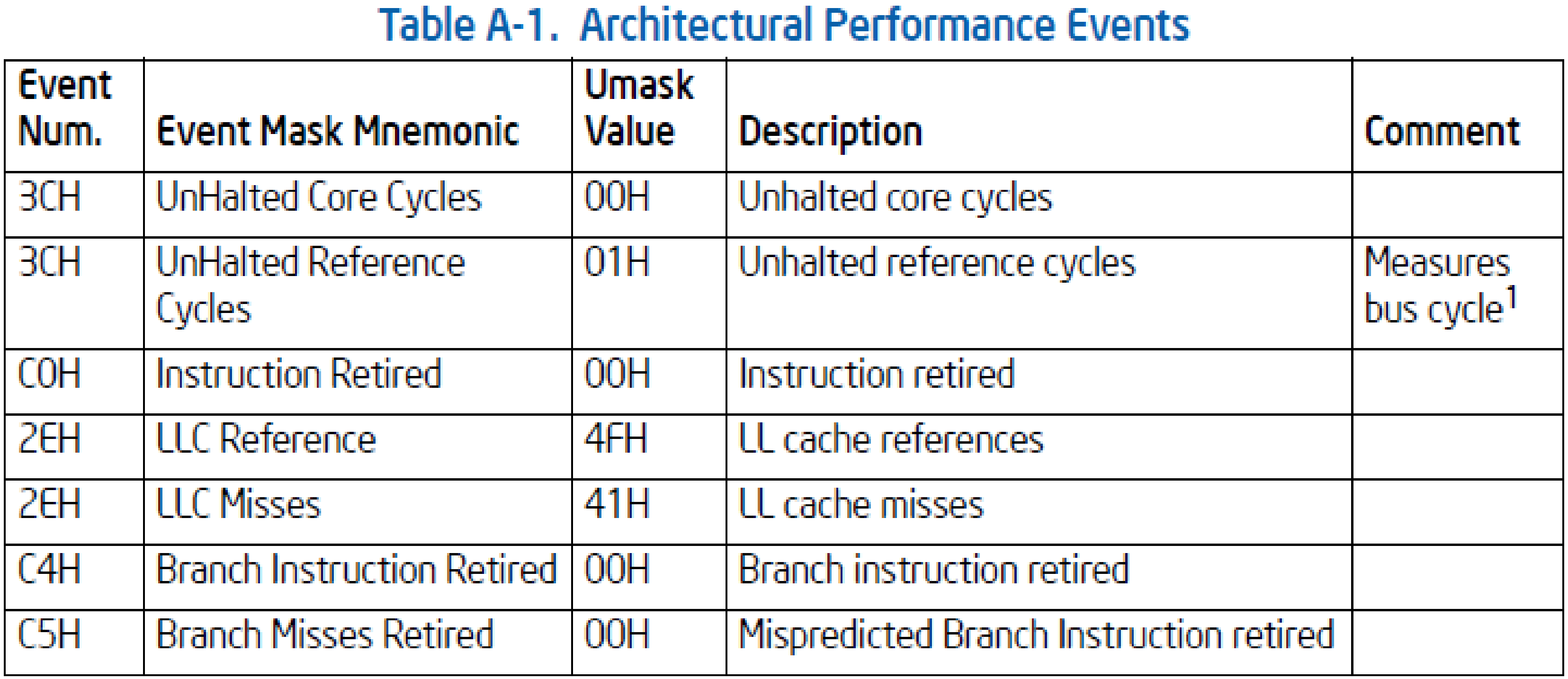


Table A-1. Architectural Performance Events

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]
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	

Command line perf monitoring

Architectural perf events

```
for c in 0 1 2 3; do
#Enable perf mon in IA32_PERF_GLOBAL_CTRL
 wrmsr -p $c 911 30064771087
#Enable fixed perf m. in IA32_FIXED_CTR_CTRL
 wrmsr -p $c 909 819
#IA32_PERFEVTSEL0(390 dec) for total branches
 wrmsr -p $c 390  0x4700c4
 #IA32_PERFEVTSEL1 (391 dec) for branch miss.
 wrmsr -p $c 391  0x4700c5
done
```

Command line perf monitoring

Architectural perf events

```
while[ 1 ]; do
  for c in 0 1 2 3; do
    wrmsr -p $c  193 0  #reset counter 0
    wrmsr -p $c  194 0  #reset counter 1
  done
  sleep 5
  for c in 0 1 2 3; do
   BRREF=`rdmsr -p $c  193` #read counter 0
   BRMISS=`rdmsr -p $c  194` #read counter 1
   echo $c `echo $BRMISS/$BRREF | bc -l`
  done
done
```

Command line perf monitoring

Advanced Example

```
proc   CPI        LLC misses    Branch misspred
----------------------------------------------------
0      1.35079      .30312        .00057
1      1.36037      .29671        .00082
2      1.35890      .30145        .00090
3      1.36869      .29146        .00049
4      1.36068      .29906        .00107
5      1.36653      .29260        .00047
6      1.35879      .30083        .00096
7      1.37438      .29364        .00044
```

app with cache blocking

```
proc   CPI        LLC misses    Branch misspred
----------------------------------------------------
0      .49865     .99099        .01582
1      .46621     .99247        .01260
2      .49805     .99210        .01281
3      .46907     .98990        .01363
4      .49935     .99038        .01508
5      .46850     .99154        .01374
6      .49610     .99049        .01602
7      .46863     .99231        .01207
```

app without cache blocking

Command line perf monitoring

Summary

- You can do this for all kind of events you are interested in
- Provides immediate feedback
- Ideal for first view assessments
- Ideal for command line work