Санкт-Петербургский Государственный Университет
Математико-механический факультет
Кафедра системного программирования

Модуль сбора информации о производительности процессоров Intel с использованием PMU для профайлера ядра ОС MS WS2008R2

Курсовая работа студента 345 группы
Серко Сергея Анатольевича

Научный руководитель ……..... М.В.Баклановский
ст. преподаватель /подпись/

Санкт-Петербург
2012г.

Оглавление

Введение.

В последние годы темпы увеличения производительности процессоров значительно снизились: возможности их совершенствования ограничены существующими технологиями, тактовая частота почти достигла своего предельного значения. В силу этого наблюдается повышенный интерес к оптимизации, повышению производительности программного обеспечения. Особенно нуждаются в высокопроизводительных программах компании, занятые в сфере хранения и управления большими объемами данных. Их системы хранения данных высокой производительности должны максимально выгодно распоряжаться имеющимися ресурсами. Зачастую, в программных продуктах, используемых в подобных системах, встречаются так называемые «бутылочные горлышки» или «горячие» точки – участки программы, требующие сравнительно большего количества памяти, времени исполнения и/или других ресурсов. Такие места должны быть найдены и, по возможности, улучшены или устранены.

Для обнаружения этих «горячих» точек, выявления причин замедления, простоя, утечки ресурсов и просто оценки производительности ПО необходимы специальные программы – профайлеры (профилировщики, анализаторы производительности). Их основной задачей является исследование характера поведения программы во всех её точках посредствам замера и фиксирования различных параметров системы, таких как:

- время исполнения программы или функции (в тактах процессора, секундах, миллисекундах...)
- количество промахов кэша различных уровней
- критические секции
- количество неверно угаданных переходов
- объем использующейся памяти
- стеки вызовов функций
- количество переключений контекста (context switches)

и др.

В текущее время рынок анализаторов производительности полон разнообразными по своей функциональности решениями. Наиболее известными и широко используемыми являются:

- Intel® VTune™ Amplifier XE

  Данный профайлер, созданный компанией Intel, эволюционировал из продукта Intel® VTune™ Performance Analyzer с добавлением функциональности Intel® Thread Profiler. Хоть он и работает на процессорах от AMD, но большая часть анализов доступна только для владельцев процессоров Intel. Полный набор функциональности доступен под ОС Microsoft Windows, под ОС Linux профайлер работает с ограничениями.

- AMD CodeAnalyst Performance Analyzer

  CodeAnalyst создан компанией AMD. Он ориентирован на процессоры AMD, хотя работает также и на процессорах компании Intel. Существуют версии как для ОС Microsoft Windows, так и для ОС Linux.

- Windows Performance Analysis Toolkit

  WPAT был создан компанией Microsoft и, как следствие, специализирован под ОС Microsoft Windows. Данный пакет включает в себя Windows Performance Recorder (WPR), основанный на Event Tracing for Windows, и Windows Performance Analyzer – утилиту для обработки информации, собранной при помощи WPR. Xperf – предшественник WPR – также поставляется в данном пакете утилит.

- AQtime Pro

  Создан компанией SmartBear. Работает на процессорах фирм AMD и Intel под руководством ОС Microsoft Windows.

Не смотря на то, что данные продукты пользуются большой популярностью на рынке, зачастую их возможностей не хватает для решения ряда задач. Например, все анализаторы производительности, упомянутые выше, нацелены на профилирование программ, работающих с уровнем привилегий User-mode, хотя некоторые и имеют достаточную функциональность для считывания данных из ядра операционной системы. Довольно часто необходимая функциональность может быть достигнута лишь при

использовании совокупности продуктов, что не только занимает лишнее время у программиста, но и к тому же довольно накладно, ввиду высокой стоимости профайлеров.
В связи с отсутствием подходящего инструмента для профилирования кода, исполняющегося в ядре ОС, в сентябре 2011 года компания EMC предложила научно-исследовательский проект, связанный с разработкой профайлера ядра ОС Microsoft Windows Server 2008 R2 x64 Edition.

Первые полноценные системы для профайлинга начали появляться в начале 70-х годов. Для платформ IBM/360 и IBM/370 были созданы средства, которые использовали прерывания по таймеру для сохранения состояния программы, хранимого в специальном контрольном регистре Program status word. Более поздние профайлеры умели собирать стек вызовов и перечислять все функции профилируемой программы. Но их основной функцией являлось измерение времени, в течение которого программа находится на процессоре, с целью обнаружения «горячих» точек. Однако такой подход не мог дать ответ на вопрос, почему происходит замедление, он лишь мог указать место, где оно происходит. Например, программа могла «затормозить» из-за многочисленных промахов кэша, информация о которых была недоступна программистам.

Специально для нужд оптимизации разработчиками процессоров была введена возможность мониторинга производительности – сбора информации о количестве микроархитектурных событий, которые описывают тонкости работы процессора. Для получения этой информации используются специальные счетчики производительности. Выбор событий, на которые должны реагировать счетчики, и действий, которые следует выполнить в случае их переполнения, а так же запуск и остановка подсчета осуществляются модификацией специальных конфигурационных регистров.

В процессорах Intel регистры-счетчики, называемые Performance Monitoring Counters (PMC), и их конфигурационные регистры являются Model-Specific (MSR) и включены в модуль Performance Monitoring Unit (PMU). В процессорах ARM регистры-счетчики называются Performance Monitor Registers, конфигурацию которых содержит в себе регистр Performance Monitor Control Register. Модуль процессора ARM, занимающийся мониторингом производительности называется Performance Monitor Unit. AMD не дает какого-либо специального названия счетчикам производительности, называя их просто Performance Counters. То же касается модуля мониторинга производительности.

Ввиду того, что системы, для которых разрабатывается профайлер, работают именно на процессорах Intel, было принято решение ограничиться детальным рассмотрением моделей процессоров исключительно этой архитектуры. Подробнее о PMC, MSR и PMU будет рассказано позже.

Наиболее важными для программиста-оптимизатора являются показания счетчиков UNCORE – среды, состоящей из процессора, за исключением его ядер, и компонент, разделяемых ядрами. В UNCORE входят разделяемый кэш последнего уровня (LLC), чипсет доступа к памяти (северный мост), интерфейс взаимодействия процессоров (в мультипроцессорных системах) и др. Именно UNCORE счетчики занимаются измерением таких критичных для производительности показателей, как попадания/промахи L3 кэша, статистика подкачки DRAM, общая пропускная способность памяти. Их показания могут быть использованы профайлером, для определения «горячих» точек программы.

Данная курсовая работа является частью проекта по разработке профайлера ядра ОС Microsoft Windows Server 2008 R2 x64 Edition. В ней описывается технология получения информации о производительности процессоров Intel с использованием Performance Monitoring Unit, рассматриваются разнообразие Model Specific счётчиков, а также описывается архитектура PMU.

Постановка задачи

Как уже упоминалось выше, профайлеру для анализа производительности нужно получить информацию о состоянии процессора в определенные моменты времени. Состояние процессора характеризуется значением некоторых параметров производительности, в частности, числом тех или иных микроархитектурных событий, произошедших к моменту времени. Передо мной была поставлена задача разработки модуля профайлера, позволяющего программировать счетчики на подсчет количества возникших Performance Monitoring событий, и впоследствии получать их показатели.

Отдельный шаг на пути создания модуля – определить модели процессоров, обладающих возможностью сбора информации об их производительности, а также изучить архитектуру модуля, предоставляющего эту возможность, методы конфигурирования счетчиков и сбора их показателей.

Решение задачи

Исследование архитектуры

Возможность считать микроархитектурные события была введена Intel вместе с внедрением в процессор модуля Performance Monitoring Unit. PMU состоит из набора MSR – Model Specific Registers – специальных контрольных регистров процессора, позволяющих изменять/контролировать его поведение. Доступ к ним осуществляется инструкциями RDMSR/WRMSR для чтения и записи соответственно. В общем случае набор MSR в PMU включает в себя:

- регистры-счетчики – для замера параметров производительности;
- регистры выбора фиксируемых параметров – в них задаются параметры производительности, измеряемые счетчиками;
- глобальные конфигурационные регистры – для точной настройки параметров измерения производительности, запуска/остановки счетчиков и пр.

У PMU старых процессоров было мало общего между собой, однако PMU-интерфейс современных процессоров начинает развиваться в сторону более высокой степени архитектурной стабильности. Это развитие можно пронаблюдать, начиная с процессоров Intel Core Solo и Intel Core Duo. В них было введено 2 класса мониторинга производительности: non-architectural и architectural. Возможности первого класса расширяются, меняются от модели к модели, тогда как возможности второго строго ограничены и являются общими для всего модельного ряда. Стоит заметить, что второй класс значительно уступает первому по количеству доступных для мониторинга параметров.

Второй класс мониторинга производительности представлен технологией Architectural Performance Monitoring. Она развивалась поэтапно: существует 3 версии Architectural Performance Monitoring – Version 1-3 соответственно. Каждая последующая включает в себя все возможности предыдущих. В версии 1 был представлен набор из нескольких Model Specific регистров выбора событий (IA32_PERFEVTSELx MSRs), результат возникновения которых передается в сопоставленные им регистры-счетчики мониторинга производительности – Performance Monitoring Counters (IA32_PMCx MSRs). Структура IA32_PERFEVTSELx (впрочем, как и адреса всех представленных регистров) является неизменной в независимости от микроархитектур. (приложение 1)

Каждый логический процессор имеет свой собственный набор IA32_PERFEVTSELx и IA32_PMCx MSR – конфигурации и счетчики не являются общими для логических процессоров, разделяющих процессорное ядро. Число счетчиков, доступных для мониторинга в логическом процессоре; число бит, поддерживаемых каждым IA32_PMCx, а так же число архитектурных событий, поддерживаемых логическим процессором, можно узнать с помощью механизма CPUID.

Вторая версия APM ввела три fixed-function PMC (IA32_FIXED_CTR0 - IA32_FIXED_CTR2), каждый из которых может считать лишь одно архитектурное событие:

- INST_RETIRED.ANY – количество выполненных инструкций
- CPU_CLK_UNHALTED.CORE – количество циклов ядра процессора, не находящегося в состоянии ожидания.
- CPU_CLK_UNHALTED.REF – количество «эталонных» циклов ядра процессора, не находящегося в состоянии ожидания (считаются циклы на исходной частоте процессора)

соответственно. Для их настройки был введён конфигурационный регистр (IA32_FIXED_CTR_CTRL). Его структура, как и структура IA32_PERFEVTSELx неизменна для процессора любой микроархитектуры, поддерживающего AMP v.2. Он включает в себя наборы 4-битных полей, каждое из которых управляет одним из счетчиков с фиксированной функцией (приложение 2).

Так же во второй версии APM был упрощен механизм программирования событий посредством ввода трех архитектурных MSR:

- IA32_PERF_GLOBAL_CTRL – позволяет программисту включить/выключить все или любую комбинацию PMC с помощью одной инструкции WRMSR.
- IA32_PERF_GLOBAL_OVF_CTRL – позволяет программному обеспечению обнулить индикаторы переполнения любого набора счетчиков базового назначения или фиксированной функции, используя WRMSR. Эту операцию следует выполнять, когда устанавливаются новые значения полей выбора

событий и UMASK, при перезагрузке значений счетчиков для продолжения работы, при завершении счета событий.

- IA32_PERF_GLOBAL_STATUS – предоставляет возможность проверить однобитовый статус состояния условий переполнения каждого счетчика. Значения 1 и 0 бит с 32 по 34 сообщают статус переполнения соответствующего счетчика.

В APM Version 3 доступны все возможности предыдущих версий. Кроме того, в версии 3 были представлены усовершенствования для поддержки ядер процессоров, состоящих более чем из одного логического процессора, например, поддерживающих технологию Intel Hyper-Threading.

Как было сказано выше, список архитектурно-определенных событий очень мал и состоит всего из 7 событий: UnHalted Core Cycles, Instruction Retired, UnHalted Reference Cycles, LLC Reference, LLC Misses, Branch Instruction Retired, Branch Misses Retired (их описание может быть найдено на стр. 797 Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3).

Класс non-architectural возможностей значительно шире architectural за счет того, что структура PMU, события, методы их конфигурации и мониторинга, адреса регистров-счетчиков, регистров выбора событий и конфигурационных регистров разнятся от микроархитектуры к микроархитектуре, от семейства к семейству. Выделяют различия в подходах к мониторингу производительности процессоров микроархитектур Intel® Core™, Intel® Atom™, Intel® Microarchitecture Code Name Nehalem, Intel® Microarchitecture Code Name Westmere, Intel® Microarchitecture Code Name Sandy Bridge.

Начиная с микроархитектуры Nehalem, события класса non-architectural делятся на два типа – события, происходящие в ядре (CORE), и события в окружении – в разделяемой ядрами подсистеме из L3 кэша, логики Intel QuickPath и контроллера памяти (UNCORE). Для каждого из этих двух типов выделяются отдельные счетчики, конфигурационные регистры. Например, в процессорах микроархитектуры Nehalem доступны четыре CORE счетчика общего назначения (IA32_PMCx) вместе с соответствующими регистрами конфигурации (IA32_PERFEVTSELx), 3 счетчика фиксированной функции (IA32_FIXED_CTRx) с их конфигурационным регистром (IA32_FIXED_CTR_CTRL) а так же восемь 48-битных

UNCORE счетчиков общего назначения (MSR_UNCORE_PerfCntrx), каждый из которых ассоциирован с конфигурационным MSR выбора событий (MSR_UNCORE_PerfEvtSelx). Имеются MSR, аналогичные контрольным регистрам APM v.2 IA32_PERF_GLOBAL_CTRL, IA32_PERF_GLOBAL_OVF_CTRL, IA32_PERF_GLOBAL_STATUS, выполняющие соответствующие конфигурационные функции для UNCORE счетчиков (MSR_UNCORE_PERF_GLOBAL_CTRL, MSR_UNCORE_PERF_GLOBAL_STATUS, MSR_UNCORE_PERF_GLOBAL_OVF_CTRL), и другие.

В силу того, что структура PMU, а так же число архитектурных событий, доступных для мониторинга, зависят микроархитектуры, модели и семейства процессора, появилась необходимость определять эти свойства процессора. Как оказалось, эту информацию для современных процессоров можно получить посредствам механизма CPUID.

Инструкция CPUID вызванная со значением параметра 1 в регистре EAX возвращает в EAX идентификационную информацию о процессоре:

- Model (4-7 биты)
- Family (8-11 биты)
- Extended Model (6-19 биты)
- Extended Family (20-27 биты)

Чтобы посчитать номер семейства процессора, необходимо сложить Family и Extended Family. Номер модели считается по более сложной формуле: к результату побитового сдвига на 4 влево Extended Model необходимо прибавить значение поля Model. Микроархитектуру, кодовое имя процессора и некоторые другие характеристики, соответствующие паре семейство-модель можно посмотреть в приложении 3.

Реализация

В силу того, что на целевых системах используются процессоры L5630, X5680 микроархитектуры Westmere, было решено для начала реализовать прототип модуля сбора информации для процессоров именно этой архитектуры. Алгоритм его работы таков:

1. Проверить, является ли процессор, на котором запущен профайлер, процессором Intel L5630 или Intel X5680 с помощью механизма CPUID: Family Number = 0x06, Model Number = 0x2C.
2. Отключить счетчики, выставив 0 в соответствующих битах IA32_PERF_GLOBAL_CTRL, MSR_UNCORE_PERF_GLOBAL_CTRL, IA32_FIXED_CTR_CTRL, а так же обнулив флаг EN регистров IA32_PERFEVTSELx и MSR_UNCORE_PerfEvtSelx.
3. Отчистить статус переполнения, записав 1 в соответствующие биты регистров IA32_PERF_GLOBAL_OVF_CTRL, MSR_UNCORE_PERF_GLOBAL_OVF_CTRL
4. Сконфигурировать события, записав коды выбранных событий и параметры их подсчета в IA32_PERFEVTSELx и MSR_UNCORE_PerfEvtSelx, обнулив при этом очередь занятости UNCORE счетчиков.
5. Обнулить счетчики, заполнив 0 регистры IA32_PMCx, MSR_UNCORE_PerfCntrx, IA32_FIXED_CTRx.
6. Включить счетчики, выставив 1 в соответствующих битах IA32_PERF_GLOBAL_CTRL, MSR_UNCORE_PERF_GLOBAL_CTRL и IA32_FIXED_CTR_CTRL, а так же установив флаг EN в регистрах IA32_PERFEVTSELx и MSR_UNCORE_PerfEvtSelx.
7. По запросу остановить счетчики так же, как это делалось в пункте 2.
8. Получить их значения с помощью инструкции RDMSR, где в качестве параметра в EAX используется адрес того или иного счетчика.
9. Если необходим дальнейший мониторинг производительности – перейти к пункту 5. Если нет – завершить работу.

Так как инструкции RDMSR и WRMSR, с помощью которых осуществляется чтение/запись Model Specific регистров, доступны только из кольца 0, пришлось задуматься над тем, каким методом можно проникнуть в ядро ОС. В ходе исследования были выявлены 3 метода: модификация системных файлов, эксплойты, драйверы. Модификация системных файлов работает до первого обновления, поэтому этот метод нас не устраивает. Эксплойты – это уязвимости, дыры в системе. Они могут быть обнаружены и легко устранены

«заплатками» (патчами). Драйверы – программы, работающие с уровнем привилегий ring0, т.е. имеющие доступ к ядру. Они не очень сложно устанавливаются в систему, их работоспособность почти не зависит от её обновления – именно поэтому было решено реализовать модуль сбора информации о производительности процессора в виде драйвера.

Заключение

В ходе работы была рассмотрена возможность процессоров Intel сбора информации о производительности – Performance Monitoring. Подробно была изучена архитектура процессорного модуля Performance Monitoring Unit, его non-architectural и architectural возможности. Детально разобраны CORE и UNCORE классы событий.

Итогом работы является прототип модуля профайлера, позволяющий для процессоров Intel® Xeon® Processor L5630 и Intel® Xeon® Processor X5680 задать микроархитектурные события для подсчета счетчиками PMU и впоследствии получать показатели этих регистров-счетчиков.

Дальнейшее поле деятельности заключается в дельнейшем изучении методов мониторинга производительности, например, технологии Precise Event Based Sampling (PEBS). Также, одним из направлений дальнейшей работы является изучение схожих с PMU модулей процессоров архитектур AMD и ARM. Одной из основных дальнейших целей является создания полноценного модуля, который будет собирать информацию о производительности любого процессора Intel, содержащего в себе модуль PMU.

Список используемых источников

1. A Study on Performance Monitoring Counters in x86-Architecture. Shibdas Bandyopadhyay
   http://www.cise.ufl.edu/~sb3/files/pmc.pdf
2. ARM11 performance monitor unit
   http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dai0195b/index.html
3. Basic Performance Measurements for AMD Athlon™ 64, AMD Opteron™ and AMD Phenom™ Processors. Paul J. Drongowski
   http://developer.amd.com/Assets/intro_to_ca_v3_final.pdf
4. BIOS and Kernel Developer's Guide for AMD AthlonTM 64 and AMD OpteronTM Processors
   http://support.amd.com/us/Processor_TechDocs/26094.PDF
5. IBM S/360 System Calls. Mark Smotherman
   http://www.cs.clemson.edu/~mark/syscall/s360.html
6. Intel® 64 and IA-32 Architectures Software Developer's Manual. Combined Volumes: 1, 2A, 2B, 2C, 3A, 3B and 3C
   http://download.intel.com/products/processor/manual/325462.pdf
7. Intel® Microarchitecture Codename Nehalem Performance Monitoring Unit Programming Guide (Nehalem Core PMU)
   http://software.intel.com/file/30320
8. Intel® Processor Identification and the CPUID Instruction
   http://www.intel.com/content/dam/www/public/us/en/documents/application-notes/processor-identification-cpuid-instruction-note.pdf
9. Performance Analysis Guide for Intel® Core™ i7 Processor and Intel® Xeon™ 5500 processors . Dr David Levinthal PhD
   http://software.intel.com/sites/products/collateral/hpc/vtune/performance_analysis_guide.pdf
10. Performance Monitoring Unit Sharing Guide. Peggy Irelan and Shihjong Kuo
    http://software.intel.com/file/30388

11. Processor Performance Counter Monitoring. Dr. Roman Dementiev. http://epic.hpi.uni-potsdam.de/pub/Home/TuKLecture2010/Dementiev_Processor__Performance__Counter_Monitoring_by_Roman_Dementiev_14-07-2010.pdf
12. Reading Intel Uncore Performance Counters from User Space. Ryan Cox http://tech.ryancox.net/2010/11/reading-intel-uncore-performance.html
13. Исследование и тестирование семлирующего метода профайлинга на примере профилировщика производительности Intel® VTune™ Amplifier XE. Одеров Р., Серко С. (в печати)
14. Теоретический анализ и разработка методик оценки достоверности информации, получаемой современными профайлерами. Булычев Иван

Приложения

Приложение 1. Структура IA32_PERFEVTSELx (APM v.3)

- Event select field (биты 0-7) – поле выбора события для мониторинга
- Unit mask (UMASK) field (биты 8-15) – эти биты определяют условие, при котором выбранное событие детектируется модулем PMU. Для каждого из микроархитектурных событий соответствующее значение UMASK определяет специфическое микроархитектурное условие.
- USR (user mode) flag (бит 16) – если установлен, возникновение микроархитектурного события инкрементирует счетчик только в том случае, когда процессор работает с уровнем привилегий 1, 2 или 3. Этот флаг может быть использован одновременно с флагом OS.
- OS (operating system mode) flag (бит 17) – если установлен, событие считает только в том случае, когда процессор работает с уровнем привилегий 0. Данный флаг может быть использован вместе с флагом USR
- E (edge detect) flag (бит 18) – когда установлен, счетчики обнаруживают «границы» выбранного микроархитектурного условия. Этот механизм позволяет измерять время, проведённое процессором в определённом состоянии.
- PC (pin control) flag (бит 19) – Каждый счетчик имеет связанный с ним внешний контакт (PMi), который может быть использован для индикации состояния счетчика внешнему оборудованию. Если флаг PC установлен, логический процессор переключает ножки/контакты PMi и инкрементирует счетчик, когда случаются performance-monitoring события. Если флаг обнулён, процессор переключает PMi ножки/контакты, когда счетчик переполняется. Переключение контактов/ножек определяется, как подача сигнала на ножку на один тик и последующее прекращение подачи сигнала.
- INT (APIC interrupt enable) flag (бит 20) – когда установлен, логический процессор генерирует исключение по переполнению счетчика.
- AT (AnyThread) Flag (бит 21) – Если этот флаг сброшен в 0, тогда счетчик будет инкрементироваться только по возникновению события в логическом процессоре, к которому привязан этот счетчик. В противном случае считаются возникновения события на любом логическом процессоре.

- EN (Enable Counters) Flag (бит 22) – когда установлен, соответствующим PMC производится измерение производительности процессора, когда сброшен – соответствующий PMC отключен.
- INV (invert) flag (бит 23) – когда установлен, результат сравнения с CMASK инвертируется.
- Counter mask (CMASK) field (биты 24-31) – когда в этом поле записано значение отличное от 0, логический процессор сравнивает эту маску с количеством событий, обнаруженных в течение одного цикла. Если это количество больше либо равно значению маски, счетчик увеличивается на единицу, в противном случае значение счетчика не меняется. Эта маска необходима для характеристики микроархитектурных условий, которые могут произойти несколько раз за один цикл. Если маска сброшена в 0, тогда на каждой итерации цикла счетчик увеличивается на число произошедших на данной итерации событий.

Приложение 2. Структура IA32_FIXED_CTR_CTRL (AMP v.2)

- Enable field (младшие 2 бита каждого из 4-битных полей) – когда установлен 0ой бит, счетчик фиксированной функции инкрементируется в случае возникновения соответствующего события в 0ом кольце. В случае, когда установлен 1ый бит, детектируются возникновения соответствующего события в остальных кольцах. Сбрасывание обоих битов в 0 остановит мониторинг производительности. Запись значения 11 позволит детектировать события при работе процессора с любым уровнем привилегий.
- AT (AnyThread) Flag (3ий бит каждого из 4-битных полей) – Если этот флаг сброшен в 0, тогда счетчик будет инкрементироваться только по возникновению события в логическом процессоре, к которому привязан этот счетчик. В противном случае считаются возникновения события на любом логическом процессоре.
- PMI field (4ый бит каждого из 4-битных полей) — когда установлен, логический процессор генерирует исключение по переполнению соответствующего счетчика.

Приложение 3. Структура IA32_PERF_GLOBAL_CTRL

- EN (enable) flags (биты 0-N) – если i-ый бит выставлен в 1, то i-ый счетчик работает, в противном случае – нет.
- ENF (enable fixed-function) flags (биты 32-34) – аналогично для счетчиков фиксированных функций.

Приложение 4. Структура IA32_PERF_GLOBAL_STATUS

- Overflow status (биты 0-N) – если i-ый бит выставлен в 1, значит i-ый счетчик переполнился.
- Fixed-function overflow status (биты 32-34) – аналогично для счетчиков фиксированных функций.
- OvfBuffer (бит 62) – используется при Precise Event-Based Sampling
- CondChgd (бит 63) – сигнализирует о том, что как минимум в одном из счетчиков произошло переполнение

Приложение 5. Структура IA32_PERF_GLOBAL_OVF_CTRL

Структура аналогична IA32_PERF_GLOBAL_STATUS, запись в соответствующие биты 1 сбросит статус переполнения счетчика – обнулит соответствующий бит в IA32_PERF_GLOBAL_STATUS.

Приложение 6. Структура MSR_UNCORE_PERFEVTSELx

- Event Select (биты 0-7) – аналогично IA32_PERFEVTSELx
- Unit Mask (биты 8-15) – аналогично IA32_PERFEVTSELx
- OCC_CTR_RST (бит 17) – когда установлен, обнуляет очередь занятости
- Edge Detect (бит 18) – аналогично IA32_PERFEVTSELx бит 18
- PMI (бит 20) – аналогично IA32_PERFEVTSELx бит 19
- EN (бит 22) – аналогично IA32_PERFEVTSELx бит 22
- INV (бит 23) – аналогично IA32_PERFEVTSELx бит 23
- Counter Mask (биты 24-31) – аналогично IA32_PERFEVTSELx

Приложение 7. Структура MSR_UNCORE_PERF_GLOBAL_CTRL

- EN_PCx (бит x, x = 0, 7): выставление в 1 запускается UNCORE счетчик общего назначения MSR_UNCORE_PerfCntrx.
- EN_FC0 (бит 32) – аналогично для UNCORE счетчика фиксированной функции MSR_UNCORE_FixedCntr0.
- EN_PMI_COREx (бит x, x = 0, 3 в случае наличия 4х ядер) – когда установлен, n-ое ядро процессора программируется на прием сигнала прерывания порожденного любым UNCORE счетчиком.
- PMI_FRZ (bit 63) – когда установлен, все UNCORE счетчики отключаются, как только хотя бы один из них вызывает прерывание. При выходе из обработчика прерывания программное обеспечение должно заново запустить счетчики, установив в 1 EN_PCx и EN_FC0 биты в регистре MSR_UNCORE_PERF_GLOBAL_CTRL.

Приложение 8. Структура MSR_UNCORE_PERF_GLOBAL_STATUS

- OVF_PCx (бит x, x = 0, 7) – если выставлен, сообщает о переполнении в UNCORE счетчике общего назначения MSR_UNCORE_PerfCntrx.
- OVF_FC0 (бит 32) – аналогично для UNCORE счетчика фиксированной функции.
- OVF_PMI (бит 61) – сигнализирует о переполнении UNCORE счетчика, в результате которого был порожден запрос на прерывание.
- CHG (bit 63) – сообщает о том, что хотя бы один бит регистра MSR_UNCORE_PERF_GLOBAL_STATUS изменил своё состояние.

Приложение 9. Структура MSR_UNCORE_PERF_GLOBAL_OVF_CTRL

Структура аналогична MSR_UNCORE_PERF_GLOBAL_STATUS, запись в соответствующие биты 1 сбросит статус переполнения счетчика

Приложение 10. Сводная таблица процессоров Intel

Process Tech.	Microarch. Codename	Processor Codename	Processor Signature	Family Num	Model Num	Intel® Brand Name(s)	Intel® Brand Processor Number
22 nm	IvyBridge	IvyBridge	0x306Ax	0x06	0x3A	Xeon™ E3	E3-12xxV2
32 nm	SandyBridge	SandyBridge	0x206Ax		0x2A	Core™ i3 Core™ i5 Core™ i7 Core™ i7 Extreme Celeron™ Desktop Celeron™ Mobile Pentium™ Desktop Pentium™ Mobile Xeon™ E3	i3-21xx/23xx-T/M/E/UE i5-23xx/24xx /25xx-T/S/M/K i7-2xxx-S/K/M/QM /LE/UE/QE i7-29xxXM G4xx, G5xx 8xx, B8xx 350, G6xx, G6xxT, G8xx 9xx, B9xx E3-12xx
		SandyBridge-E	0x206Dx		0x2D	Core™ i7 Core™ i7 Extreme	I7-3820/3930K i7-3960X
		SandyBridge-EN				Xeon™ E5	E5-24xx
		SandyBridge-EP				Xeon™ E5	E5-16xx, 26xx/L/W
	Westmere	Arrandale	0x2065x		0x25	Celeron™ Mobile Pentium™ Mobile Core™ i3 Core™ i5 Core™ i7	P4xxx, U3xxx P6xxx, U5xxx i3-3xxE, i3-3xxM, i3-3xxUM i5-4xxM/UM, i5-5xxE/M/UM i7-6xxE/LE/UE/M/LM/UM
		Clarksdale				Pentium™ Desktop Core™ i3 Core™ i5 Xeon™3000	G69xx i3-5xx i5-6xx, i5-6xxK L34xx
		Gulftown	0x206Cx		0x2C	Core™ i7 Core™ i7 Extreme Xeon™ 3000	i7-9xx i7-9xxX W36xx
		Westmere-EP				Xeon™ 3000 Xeon™ 5000	W36xx L56xx, E56xx, X56xx
		Westmere-EX	0x206Fx		0x2F	Xeon™ E7	E7-2xxx, E7-48xx, E7-88xx

Process Tech.	Microarch. Codename	Processor Codename	Processor Signature	Family Num	Model Num	Intel® Brand Name(s)	Intel® Brand Processor Number
45 nm	Nehalem	Clarksfield	0x106Ex	0x06	0x1E	Core™ i7 Core™ i7 Extreme	i7-7xxQM, i7-8xxQM i7-9xxXM
		Lynnfield				Core™ i5 Core™ i7 Xeon™ 3000	i5-7xx, i5-7xxS i7-8xx, i7-8xxS, i7-8xxK X34xx
		Jasper Forest				Xeon™ 5000 Celeron™ Desktop	LC55xx, EC55xx P10xx
		Bloomfield	0x106Ax		0x1A	Core™ i7 Extreme Core™ i7 Xeon™ 3000	17-965/975 i7-9x0 W35xx
		Nehalem-EP				Xeon™ 5000	L55xx, E55xx, X55xx,
		Nehalem-EX	0x206Ex		0x2E	Xeon™ 7000 Xeon™ 6000	L75xx, E75xx, X75xx E65xx, X65xx
	Penryn	Yorkfield	0x1067x		0x17	Core™ 2 Quad Core™ 2 Extreme Xeon™ 3000	Q9xxx, Q8xxx, !9xxxS QX9xxxб, L33xx, X3350
		Wolfdale				Celeron™ Desktop Core™ 2 Duo Pentium™ Xeon™ 5000/3000	E3xxx E7xxx, E8xxx E5xxx, E6xxx, E6xxxK L52xx, E31xx
		Penryn				Core™ 2 Duo Mobile Celeron™ M	P7xxx, P9xxx, SL9xxx 722
		Harpertown (DP)				Xeon™ 5000	L54xx, E54xx, X54xx
		Dunnington (MP)	0x106Dx		0x1D	Xeon™ 7000	L74xx, E74xx, Q7xx

Process Tech.	Microarch. Codename	Processor Codename	Processor Signature	Family Num	Model Num	Intel® Brand Name(s)	Intel® Brand Processor Number
65 nm	Merom	Clovertown	0x006Fx	0x06	0x0F	Xeon™ 5000	E53xx, L53xx, X53xx
		Kentsfield				Xeon™ 3000 Core™ 2 Quad Core™ 2 Extreme	X32xx Q6600 QX6xxx
		Conroe				Xeon™ 3000 Pentium™ Core™ 2 Duo Core™ 2 Extreme Celeron™ Desktop	30xx E21xx E43xx, E6xxx X6800 E1600
		Merom				Core™ 2 Duo M Pentium™ Mobile Core™ 2 Extreme M	L7xxx,T5xxx,T7xxx,U7xxx T3200 X7xxx
		Woodcrest				Xeon™ 5000	51xx
		Merom Conroe	0x1066x		0x16	Celeron™ Desktop Celeron™ Mobile	4xx 5xx
	Presler	Cedar Mill	0x0066x	0x0 F	0x06	Pentium™ 4	3xx, 6xx
		Presler				Pentium™ D	9xx
90 nm	Prescott	Nocona Irwindale	0x0063x 0x0064x		0x03/ 0x04	Xeon™	
		Prescott				Celeron™ D Pentium™ 4	3xx 5xx
	Dothan	Dothan	0x0060x	0x06	0x0D	Celeron™ M Pentium™ Mobile	3xx 7xx