**Лекция №9**

**Тема «Технологии повышения производительности процессоров»**

**Цель: рассмотреть основные технологии повышения производительности процессоров**

**Конвейерная обработка команд**

Обработка команды, или цикл процессора может быть разделена на несколько основных этапов (микрокоманд), которых как минимум пять (выборка, декодирование, чтение исходных данных, выполнение, запись результата).

Каждая операция требует для своего выполнения времени, равного такту генератора процессора. Все этапы команды задействуются только один раз и всегда в одном и том же порядке – одна за другой. Это, в частности, означает, что если логическая схема первой микрокоманды выполнила свою работу и передала результаты второй, то для выполнения текущей команды она больше не понадобится, и, следовательно, может приступить к выполнению следующей команды.

Такая технология обработки команд носит название конвейерной (pipeline), обработки. Каждая часть устройства называется ступенью (стадией) конвейера, а общее число ступеней – длиной линии конвейера.

**Конвейеризация**осуществляет многопоточную параллельную обработку команд, так что в каждый момент одна из команд считывается, другая декодируется и т. д., и всего в обработке одновременно находится пять команд. Таким образом, на выходе конвейера на каждом такте процессора появляется результат обработки одной команды (одна команда в один такт).

Приведенный пример процессора (5 микроопераций) является гипотетическим – в реальных ЦП конвейер обработки команд сложнее и включает большее количество ступеней. Причина увеличения длины конвейера заключается в том, что многие команды являются довольно сложными и не могут быть выполнены за один такт процессора, особенно при высоких тактовых частотах. Поэтому каждая из упомянутых пяти стадий обработки команд в свою очередь может состоять из нескольких ступеней конвейера.

С ростом числа линий конвейера и увеличением числа ступеней на линии увеличивается пропускная способность процессора при неизменной тактовой частоте. Наоборот, чем больше ступеней насчитывается в конвейере, тем меньшая работа выполняется за такт и тем выше можно поднимать частоту процессора.

**Суперскаляризация.** Процессоры с несколькими линиями конвейера получили название суперскалярных. Pentium — первый суперскалярный процессор Intel. Здесь две линии, что позволяет ему при одинаковых частотах быть вдвое производительней i80486, выполняя сразу две инструкции за такт.

Наряду с конвейером команд, используются конвейеры данных. Сочетание этих двух конвейеров дает возможность достичь очень высокой производительности на определенных классах задач, особенно если используется несколько различных конвейерных процессоров, способных работать одновременно и независимо друг от друга.

**Операции над вещественными числами (с плавающей запятой)**

**Сопроцессоры.**Для расширения вычислительных возможностей центрального процессора — выполнения арифметических операций, вычисления основных математических функции (тригонометрических, показательных, логарифмических) и т. д. — в состав ЭВМ добавляется математический сопроцессор. Применение сопроцессора повышает производительность вычислений в сотни раз. В разных поколениях процессоров он назывался по-разному — FPU (Floating Point Unit — блок чисел/операций с плавающей точкой — БПЗ) или NPX (Numeric Processor extension — числовое расширение процессора).

Для процессоров 386 и ниже сопроцессор был отдельной микросхемой, подключаемой к локальной нише основного процессора. В любом случае сопроцессор исполняет только свои специфические команды, а всю работу по декодированию инструкций и доставке данных осуществляет ЦП.

**Блоки операций с плавающей запятой.**С программной точки зрения сопроцессор и процессор выглядят как единое целое. В современных (486+) процессорах БПЗ располагается на одном кристалле с центральным процессором.

**Увеличение разрядности систем**

В «романтические» 1980-е годы соответствие между типом ЭВМ и ее разрядностью имело простейший вид:

• микроЭВМ — 8 разрядов;

• миниЭВМ — 16 разрядов;

• большие ЭВМ — 32 разряда;

• сверхбольшие (супер) ЭВМ — 64 разряда.

В процессе развития микропроцессоров Intel рубежи в 16 и 32 разряда (IA-32) были преодолены довольно быстро, а в районе 2004 г. произошел переход и на 64-разрядные архитектуры в процессорах Intel и AMD.

Преимущества 64-битной архитектуры микропроцессоров главным образом относятся к памяти. Если взять два идентичных микропроцессора, и один из них будет 32-битным, а другой — 64-битным, то последний сможет адресовать намного больший объем памяти, чем 32-битный (264против 232). Известны следующие архитектуры на 64 разряда (64-bit architecture).

**IA-64.**Спецификация «IA-64» означает «Архитектура Intel, 64 бита», но связь с IA-32 — только по названию. Архитектура IA-64 не совместима непосредственно с набором команд IA-32. Здесь появляется полностью отличный набор команд, а также используются принципы VLIW вместо выполнения вне естественного порядка. IA-64 — архитектура, используемая линией процессоров Itanium.

**AMD64.** Набор команд AMD64, первоначально названный х86-64, в значительной степени построен на основе IA-32 и таким образом обеспечивает наследственность семейства х86. При расширении набора команд AMD воспользовался возможностью, чтобы очистить часть его от ряда «устаревших» команд — наследия «16-разрядных времен».

**EM64T** (Extended Memory 64-bit Technology, или Intel 64) — набор команд, (ранее известный как Yamhill), объявленный Intel в феврале 2004 г., в подражание AMD64. ЕМ64Т в целом совместим с кодами, написанными для AMD64, хотя и имеет ряд недостатков сравнительно с AMD64.

Поскольку AMD64 и ЕМТ64 почти не различаются, для ссылки на них используются нейтральные названия — х86-64, х86\_64 (Linux и Apple's Mac OS X), х64 (Microsoft и Sun Microsystems).

**Векторная обработка (SIMD-команды)**

SIMD — поток данных, обрабатываемых одной командой. В последовательных расширениях системы команд х86, выполненных Intel и AMD, все более полно используются принципы обработки одной командой вектора (потока) данных.

ММХ (MultiMedia extension) — архитектура системы команд, непосредственно предназначенных для задач мультимедиа, связи и графических приложений, которые часто используют сложные алгоритмы, исполняющие одинаковые операции на большом количестве типов данных (байты, слова и двойные слова). При этом было достигнуто общее повышение производительности на 10—20 %, а в программах обработки мультимедиа — до 60 %.

Архитектура 3DNow! впервые реализована в процессорах AMD K6-2 (май 1998 г.). Технология 3DNow! включает 21 дополнительную команду, новые типы данных для поддержки высокопроизводительной обработки ЗD-графики и звука.

SSE (или SIMD-FP) — система команд Streaming SIMD Extensions — SIMD-расширение, предложенное Intel в 1999 г. в Pentium III (ядро Katmai), отсюда вариант названия — KNI (Katmai New Instructions). Это 70 новых команд, в том числе:

• для повышения эффективности операций с плавающей запятой

• дополнение ранее введенных команд ММХ;

• повышение производительности кэш-памяти L1 при работе с мультимедийными данными.

SSE2 — введенный с Pentium IV набор команд является существенным развитием SSE, оперирует с теми же самыми регистрами и обратно совместим с SSE процессора Pentium III. SSE2 представляет собой симбиоз ММХ и SSE и позволяет работать с любыми типами данных, вмещающимися в 128-битовые регистры.

SSE3 — набор команд, также известный как Prescott New Instructions (PNI), является третьей версией команд SSE для IA-32. Intel использует SSE3 с начала 2004 г. в ЦП Pentium IV Prescott. В апреле 2005 г. AMD также включает SSE3 в ЦП Athlon 64. SSE3 содержит 13 дополнительных по отношению к SSE2 команд.

**Динамическое исполнение (dynamic execution technology)**

Динамическое исполнение — технология обработки данных процессором, обеспечивающая более эффективную работу процессора за счет манипулирования данными, а не просто линейного исполнения списка инструкций.

**Предсказание ветвлений.** С большой точностью (более 90 %) процессор предсказывает, в какой области памяти можно найти следующие инструкции. Это оказывается возможным, поскольку в процессе исполнения инструкции процессор просматривает программу на несколько шагов вперед.

Это обеспечивает значительное повышение производительности. Например, программный цикл, состоящий из пересылки, сравнения, сложения и перехода в 80486 DX, выполняется за шесть тактов синхронизации, а в — Pentium за два (команды пересылки и сложения, а также сравнения и перехода сочетаются и предсказывается переход).

**Внеочередное выполнение (выполнение вне естественного порядка**— **out-of-orderexecution).**Процессор анализирует поток команд и составляет график исполнения инструкций в оптимальной последовательности, независимо от порядка их следования в тексте программы, просматривая декодированные инструкции и определяя, готовы ли они к непосредственному исполнению или зависят от результата других инструкций. Далее процессор определяет оптимальную последовательность выполнения и исполняет инструкции наиболее эффективным образом.

**Выполнение по предположению (спекулятивное — speculative) —**процессор выполняет инструкции (до пяти инструкций одновременно) по мере их поступления в оптимизированной последовательности (спекулятивно). Поскольку выполнение инструкций происходит на основе предсказания ветвлений, результаты сохраняются как предположительные («спекулятивные»). На конечном этапе порядок инструкций восстанавливается.

**Предикация.**Обычный компилятор транслирует оператор ветвления (например, if-then-else) вблоки машинного кода, расположенные последовательно в потоке. Обычный процессор, в зависимости от исхода условия, исполняет один из этих базовых блоков, пропуская все другие. Более развитые процессоры пытаются прогнозировать исход операции и предварительно выполняют предсказанный блок. При этом в случае ошибки много тактов тратится впустую. Сами блоки зачастую весьма малы — две или три команды, — а ветвления встречаются в коде в среднем каждые шесть команд. Такая структура кода делает крайне сложным его параллельное выполнение.

При использовании предикации компилятор, обнаружив оператор ветвления в исходной программе, анализирует все возможные ветви (блоки) и помечает их метками или предикатами (predicate). После этого он определяет, какие из них могут быть выполнены параллельно (из соседних, независимых ветвей).

В процессе выполнения программы ЦП выбирает команды, которые взаимно независимы и распределяет их на параллельную обработку. Если ЦП обнаруживает оператор ветвления, он не пытается предсказать переход, а начинает выполнять все возможные ветви программы.

Таким образом, могут быть обработаны все ветви программы, но без записи полученного результата. В определенный момент процессор, наконец «узнает» о реальном исходе условного оператора, записывает в память результат «правильной ветви» и отменяет остальные результаты.

В то же время, если компилятор не «отметил» ветвление, процессор действует как обычно — пытается предсказать путь ветвления и т. д. Испытания показали, что описанная технология позволяет устранить более половины ветвлений в типичной программе, и, следовательно, уменьшить более чем в 2 раза число возможных ошибок в предсказаниях.

Опережающее чтение (предварительная загрузка данных, чтение по предположению) разделяет загрузку данных в регистры и их реальное использование, избегая ситуации, когда процессору приходится ожидать прихода данных, чтобы начать их обработку.

Прежде всего, компилятор анализирует программу, определяя команды, которые требуют приема данных из оперативной памяти. Там, где это возможно, он вставляет команды опережающего чтения и парную команду контроля опережающего чтения (speculative check). В то же время компилятор переставляет команды таким образом, чтобы ЦП мог их обрабатывать параллельно.

В процессе работы ЦП встречает команду опережающего чтения и пытается выбрать данные из памяти. Может оказаться, что они еще не готовы (результат работы блока команд, который еще не выполнился). Обычный процессор в этой ситуации выдает сообщение об ошибке, однако система откладывает «сигнал тревоги» до момента прихода процесса в точку «команда проверки опережающего чтения». Если к этому моменту все предшествующие подпроцессы завершены и данные считаны, то обработка продолжается, в противном случае вырабатывается сигнал прерывания.

Возможность располагать команду предварительной загрузки до ветвления очень существенна, так как позволяет загружать данные задолго до момента использования (напомним, что в среднем каждая шестая команда является командой ветвления).

**Технология Hyper-Threading (HT)**

Здесь реализуется разделение времени на аппаратном уровне, разбивая физический процессор на два логических процессора, каждый из которых использует ресурсы чипа — ядро, кэш-память, шины, исполнительное устройство. Благодаря НТ многопроцессная операционная система использует один процессор как два, и выдает одновременно два потока команд. Смысл технологии заключается в том, что в большинстве случаев исполнительные устройства процессора далеки от полной загруженности. От передачи на выполнение вдвое большего потока команд повышается загрузка исполнительных устройств.

Специалисты Intel оценивают повышение эффективности в 30 % НТ-процессоров при использовании многопрограммных ОС и обычных прикладных программ.

**Многоядерные процессоры**

Многоядерная архитектура предполагает размещение двух или более основных вычислительных агрегатов в пределах единственного процессора. Этот многоядерный процессор имеет единственный интерфейс с системной платой, но операционные системы «видят» каждое из его ядер как дискретный логический процессор со всеми связанными ресурсами. Это отличает их от технологии Hyper-Threading (где отдельные процессы выполняются единственным ядром) и существующие ресурсы используются более эффективно.

Разделяя вычислительную нагрузку, выполняемую единственным ядром в традиционных процессорах между многими ядрами, многоядерный процессор может выполнить большую работу в пределах отдельного цикла ЭВМ. Чтобы реализовать это увеличение эффективности, соответствующее программное обеспечение должно поддерживать это распараллеливание. Эти функциональные возможности называют «параллелизмом уровня подпроцесса (нити)» или «threading». Приложения и операционные системы, которые поддерживают это, упоминаются как мульти-подпро-цессные или «multi-threaded».

**Другие технологии**

**Технологии «невыполнимых битов»**(No-eXecute bit). Бит «NX» (63-й бит адреса) позволяет операционной системе определить, какие страницы адреса могут содержать исполняемые коды, а какие – нет. Попытка обратиться к NX-адресу как к исполняемой программе вызывает событие «нарушение защиты памяти», подобное попытке обратиться к памяти «только для чтения» или к области размещения ОС. Этим может быть запрещено выполнение программного кода, находящегося в некоторых страницах памяти, таким образом предотвращая вирусные или хакерские атаки. Обозначение «NX-bit» используется AMD, Intel использует выражение «XD-bit» (eXecute Disable bit).

**Технологии энергосбережения**

**OnNow**PC — способ управления энергопотреблением системы, который заключается в значительном уменьшении потребления электрической энергии, но так, чтобы система в любой момент времени была готова к работе без перезагрузки ОС (например, как готов телевизор, включаемый с помощью удаленного пульта). Система при включении остается способной реагировать на внешние события: нажатие кнопки пользователем, сигнал из сети. Это обеспечивается тем, что небольшая часть системы остается постоянно включенной. Технология OnNow PC требует выполнения следующих условий:

• операционная система берет на себя управление энергопотреблением;

• все устройства, входящие в систему, должны допускать возможность эффективного регулирования потребления ими электрической энергии;

• должен быть предусмотрен ряд определяемых операционной системой последовательных энергетических состояний, переходящих одно в другое.

**Интеллектуальное управление электропитанием**(Intel Intelligent Power Capability) — уменьшение потребления энергии путем включения именно тех логических блоков, которые требуются в данный момент.

**Enhanced Intel Speed STep (EIST)**— идентичен механизму, осуществленному в процессорах Intel мобильных ПК который позволяет процессору уменьшать его тактовую частоту, когда не требуется высокая загрузка, таким образом значительно сокращая нагрев центрального процессора и потребление мощности.

**Контрольные вопросы**

1. Какие бывают типы процессоров?
2. В чем заключается конвейерная обработка команд?
3. В чем заключается суперскаляризация?
4. В чем заключается векторная обработкя?
5. В чем заключается технология Hyper-Threading (HT)?
6. Что такое многоядерные процессоры
7. Назовите основные технологии повышения производительности процессоров