Информатика

Лекция 9

Hyper-Threading

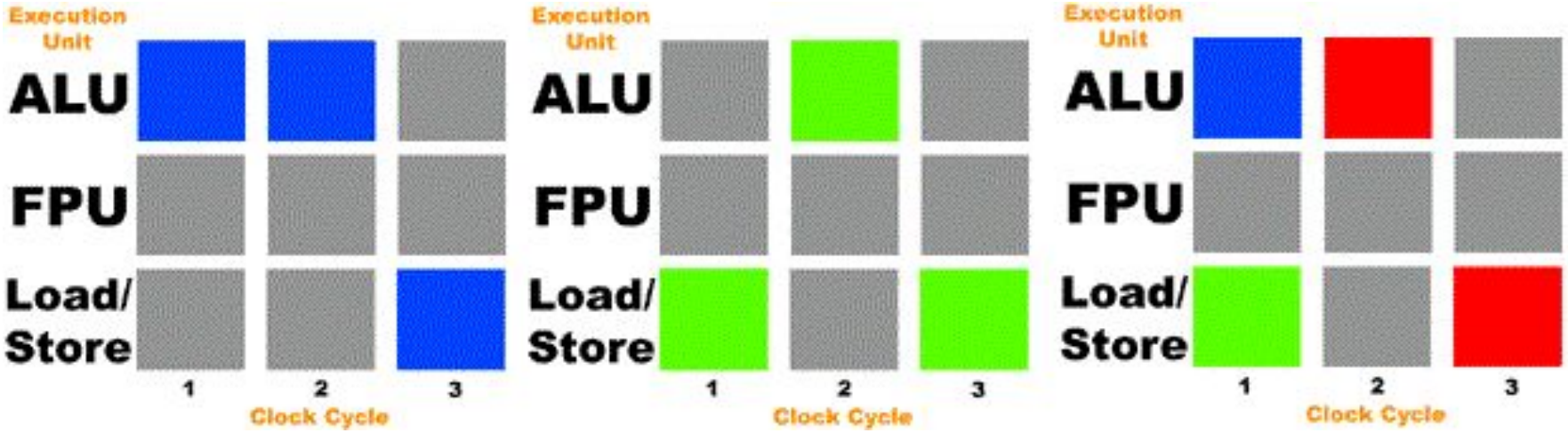
- Технология Intel® Hyper-Threading (Intel® HT) обеспечивает более эффективное использование ресурсов процессора, позволяя выполнять несколько потоков на каждом ядре.

- В отношении производительности эта технология повышает пропускную способность процессоров, улучшая общее быстродействие многопоточных приложений.

Hyper-Threading

- HT позволяет одному физическому ядру обрабатывать одновременно несколько (обычно два) логических потока.
- Процессор, поддерживающий гиперпоточность: может хранить информацию сразу о нескольких выполняющихся потоках;
- Содержит по одному набору регистров (то есть блоков быстрой памяти внутри процессора) и по одному контроллеру прерываний (то есть встроенному блоку процессора, отвечающему за возможность последовательной обработки запросов о наступлении какого-либо события, требующего немедленного внимания, от разных устройств) на каждый логический процессор.

Hyper-Threading



- Как видно на картинке выше при выполнении одной задачи процессор не занят на 100% — какие-то блоки процессора не нужны в данной задаче, где-то ошибается модуль предсказания, где-то происходит ошибка обращения к кэшу — в общем и целом при выполнении задачи процессор редко бывает занят больше, чем на 70%.
- Технология HT как раз передает незанятым блокам процессора вторую задачу, и получается что одновременно на одном ядре обрабатываются две задачи.

Hyper-Threading

- Удвоения производительности не происходит по понятным причинам — очень часто получается так, что двум задачам нужен один и тот же вычислительный блок в процессоре, и тогда мы видим простой: пока одна задача обрабатывается, выполнение второй на это время просто останавливается.
- В итоге время, затраченное процессором с HT на две задачи, оказывается больше времени, требуемого на вычисление самой тяжелой задачи, но меньше того времени, которое нужно для последовательного вычисления обеих задач.

Повышение производительности

- Кристалл процессора с поддержкой HT физически больше кристалла процессора без HT в среднем на 5% (именно столько занимают дополнительные блоки регистров и контроллеры прерываний)
- Поддержка HT позволяет нагрузить процессор на 90-95%
- В сравнении с 70% без HT мы получаем, что прирост в лучшем случае будет 20-30% — цифра достаточно большая.

Понижение производительности

- Однако не все так хорошо: бывает, что прироста производительности от HT нет вообще, и даже бывает так, что HT ухудшает производительность процессора.
- Это бывает по многим причинам:

- **Нехватка кэш-памяти.** К примеру в современных четырехядерных i5 находится 6 мб кэша L3 - по 1.5 мб на ядро. В четырехядерных i7 с HT кэша уже 8 мб, но так как логических ядер 8, то мы получаем уже только 1 мб на ядро — при вычислениях некоторым программам этого объема может не хватать, что приводит к падению производительности.

Понижение производительности

- **Отсутствие оптимизации ПО.** Самая основная проблема — программы считают логические ядра физическими, из-за чего при параллельном выполнении задач на одном ядре часто возникают задержки из-за обращения задач к одному и тому же вычислительному блоку, что в итоге сводит сводит прирост производительности от HT на нет.

  Зависимость данных. Вытекает из предыдущего пункта — для выполнения одной задачи требуется результат другой, а она еще не выполнена. И опять же мы получаем простой, снижение загрузки на процессор и небольшой прирост от HT.

Программы, умеющие работать с гиперпоточностью

- Таких много, ибо для вычислений HT это достоинство — тепловыделение практически не растет, процессор особо больше не становится, а при правильной оптимизации можно получить прирост до 30%.
- Поэтому ее поддержку быстро внедрили в те программы, где легко можно сделать распараллеливание нагрузки — в архиваторы (WinRar), программы для 2D/3D моделирования (3ds Max, Maya), программы для обработки фото и видео (Sony Vegas, Photoshop, Corel Draw).

Программы, плохо работающие с гиперпоточностью

- Традиционно это большинство игр — их обычно бывает трудно грамотно распараллелить, поэтому зачастую четырех физических ядер на высоких частотах (i5) более чем хватает для игр, распараллелить которые под 8 логических ядер в i7 оказывается непосильной задачей.
- Стоит учитывать и то, что есть фоновые процессы, и если процессор не поддерживает HT, то их обработка ложится на физические ядра, что может замедлить игру. Тут i7 с HT оказывается в выигрыше — все фоновые задачи традиционно имеют пониженный приоритет.

Пределы адресуемой памяти

- Хотя 64-разрядный процессор теоретически мог бы адресоваться к 16 экзабайтам памяти (2^{64}), в настоящее время Win64 поддерживает 16 Тб - значение, которое представлено 44 разрядами.

- Почему же нельзя задействовать все 64 разряда, чтобы адресоваться к 16 экзабайтам памяти?

- Сама архитектура (но не современное оборудование) допускает расширение до 52 разрядов (4 петабайтов).

Пределы адресуемой памяти

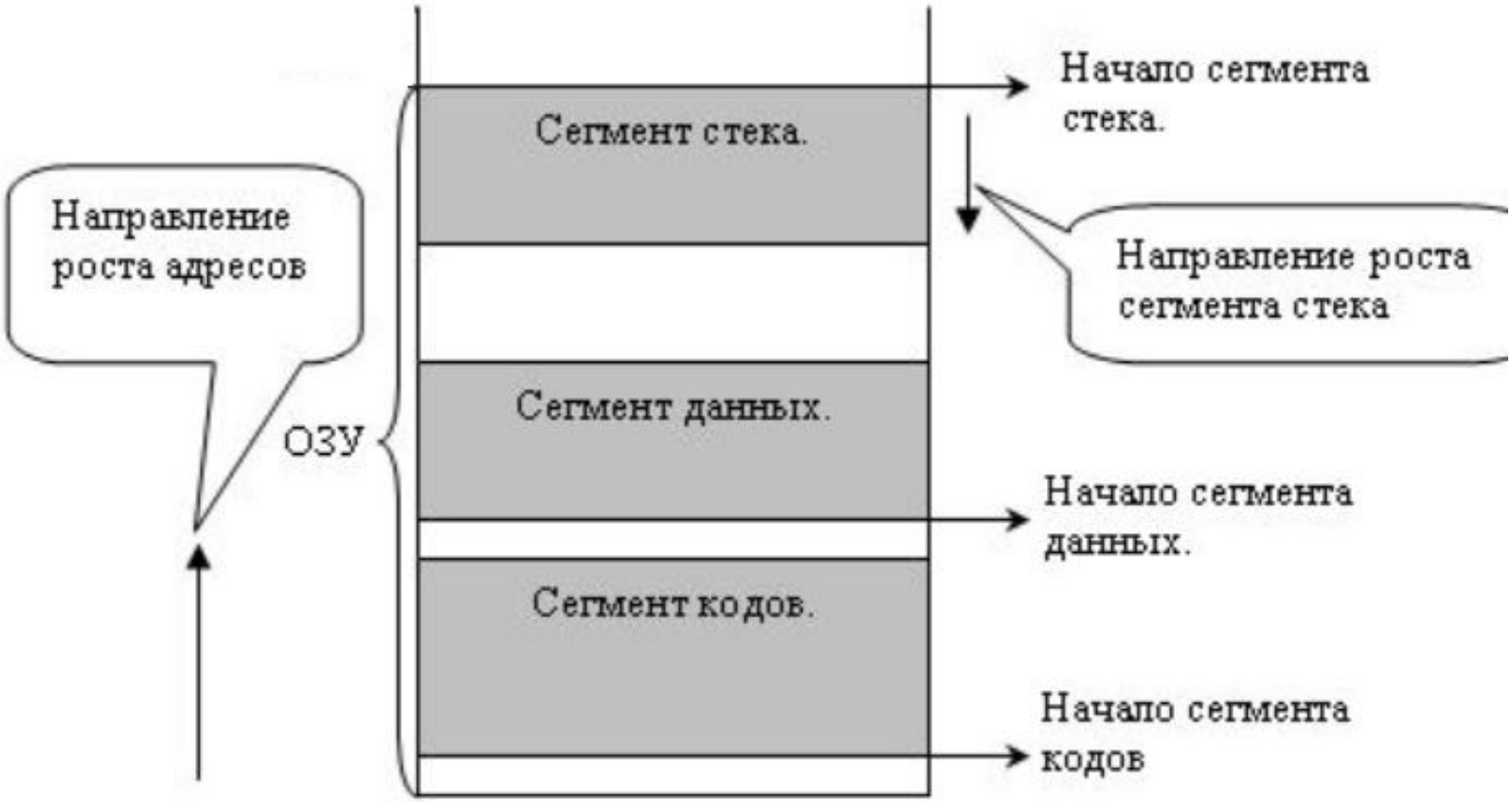
- Как и в Win32, адресуемая память делится на области пользовательского режима и режима ядра.
- Каждому процессу выделяется собственное уникальное пространство максимальным размером 8 Тб в нижней части памяти, а код режима ядра размещается в верхних 8 Тб и разделяется всеми процессами.

Адресация

- Как известно программист, когда пишет программы работает не с физическим адресом, а только с логическим. И то если он программирует на ассемблере.
- В том же Си ячейки памяти от программиста уже скрыты указателями, для его же удобства, но если грубо говорить указатель это другое представление логического адреса памяти, а в Java и указателей нет ☹.

- Рассмотрим адресное пространство программного режима 32 битного процессора.
- Адресное пространство этого режима будет состоять из 2^{32} ячеек памяти пронумерованных от 0 и до 2^{32}-1.
- **Логический адрес --> Линейный (виртуальный)--> Физический**

Структура сегментов программы



Линейный адрес

- Линейный адрес вычисляется по формуле:

- **Линейный адрес=Базовый адрес сегмента(на слайде это начало сегмента) + смещение**

Сегмент кода

- Базовый адрес сегмента кода берется из регистра CS. Значение смещения для сегмента кода берется из регистра EIP, в котором хранится адрес инструкции, после исполнения которой, значение EIP увеличивается на размер этой команды.
- Если команда занимает 4 байта, то значение EIP увеличивается на 4 байта и будет указывать уже на следующую инструкцию.
- Все это делается автоматически без участия программиста.

Сегмент данных

- Данные загружаются в регистры DS, ES, FS, GS
  Это значит что сегментов данных может быть до 4х.
- На слайде он один.

- Смещение внутри сегмента данных задается как операнд команды. По умолчанию используется сегмент на который указывает регистр DS.

Страничная память

- **Страничная память** — способ организации виртуальной памяти, при котором единицей отображения виртуальных адресов на физические является регион постоянного размера (т. н. *страница*). Типичный размер страницы — 4096 байт, для некоторых архитектур — до 128 КБ.
- Поддержка такого режима присутствует в большинстве 32-битных и 64-битных процессоров. Такой режим является классическим для почти всех современных ОС, в том числе Windows и семейства UNIX.
- Широкое использование такого режима началось с процессора VAX и ОС VMS с конца 70-х годов (по некоторым сведениям, первая реализация).
- В семействе x86 поддержка появилась с поколения 386, оно же первое 32-битное поколение.

Решаемые задачи

1. поддержка изоляции процессов и защиты памяти путём создания своего собственного виртуального адресного пространства для каждого процесса
2. поддержка изоляции области ядра от кода пользовательского режима
3. поддержка памяти «только для чтения» и неисполняемой памяти
4. поддержка отгрузки давно не используемых страниц в область подкачки на диске
5. поддержка отображённых в память файлов, в том числе загрузочных модулей
6. поддержка разделяемой между процессами памяти.

Концепции

- Адрес, используемый в машинном коде, то есть значение указателя, называется «виртуальный адрес».
- Адрес, выставляемый процессором на шину, называется «линейный адрес» (который позже преобразовывается в физический).

- Число записей в одной таблице ограничено и зависит от размера записи и размера страницы. Используется многоуровневая организация таблиц, часто 2 или 3 уровня, иногда 4 уровня (для 64-разрядных архитектур).
- В случае 2 уровней используется «директория» страниц, в которой хранятся записи, указывающие на физические адреса таблиц страниц. В таблицах содержатся записи, указывающие на страницы данных.
- При использовании 3-х уровневой организации, добавляется супер-директория, хранящая записи, указывающие на несколько директорий.

- Старшие биты виртуального адреса указывают на номер записи в директории, средние — номер записи в таблице, младшие (адрес внутри страницы) попадают в физический адрес без трансляции.
- Формат записей таблиц, их размер, размер страницы и организация таблиц зависит от типа процессора, а иногда и от режима его работы.

Page Table Entry (PTE)

- Исторически x86 использует 32-битные PTE, 32-битные виртуальные адреса, 4KB-страницы, 1024 записи в таблице, двухуровневые таблицы.

- Старшие 10 бит виртуального адреса — номер записи в директории, следующие 10 — номер записи в таблице, младшие 12 — адрес внутри страницы.

Соображения безопасности

- Первоначально архитектура x86 не имела флага «страница недоступна на исполнение» (NX).
- Поддержка данного флага появилась в архитектуре x86 как часть режима PAE (Physical Address Extension) в поколении Pentium 4, под большим давлением со стороны специалистов по безопасности.
- Установка данного флага на страницах стека и кучи (heap) позволяет реализовать аппаратно защиту от исполнения данных, что делает невозможной работу многих разновидностей вредоносного ПО, в том числе, например, злонамеренную эксплуатацию многих брешей в Internet Explorer.