Глава 3
ПРОЦЕССОРЫ: МИКРОАРХИТЕКТУРЫ И ПРОГРАММИРОВАНИЕ

Архитектура компьютерных систем

Вспоминаем функциональное устройство процессора

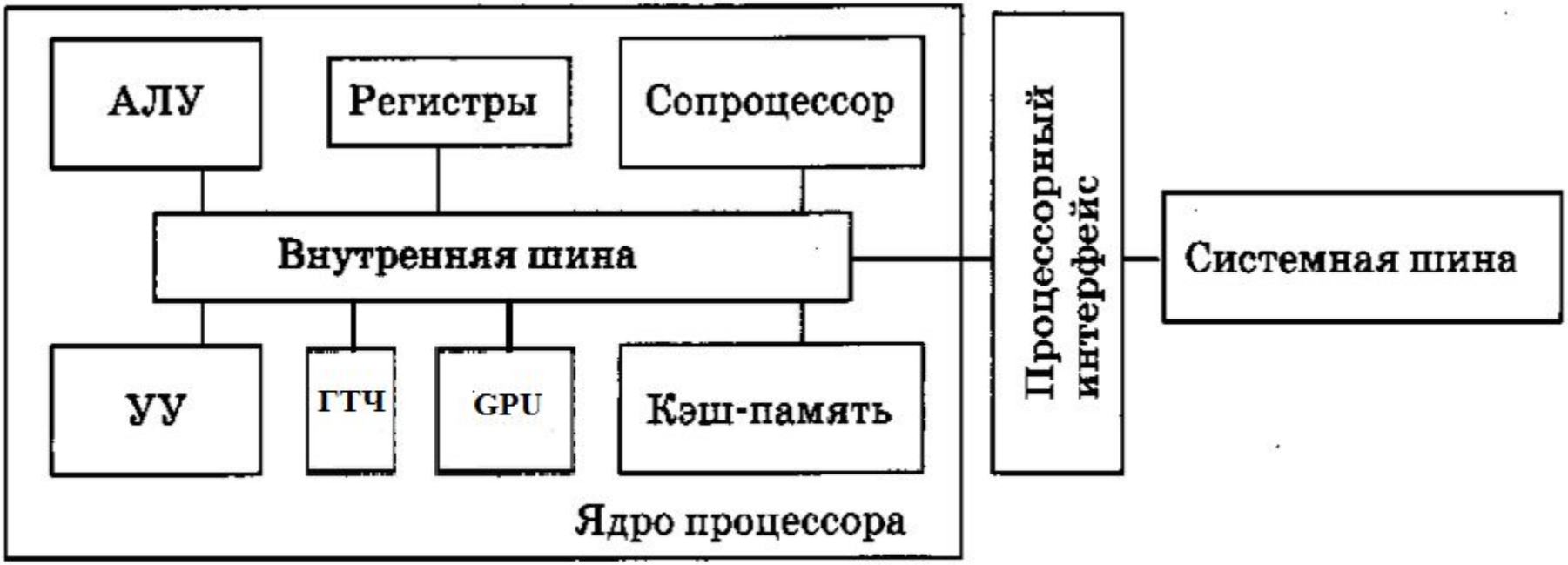
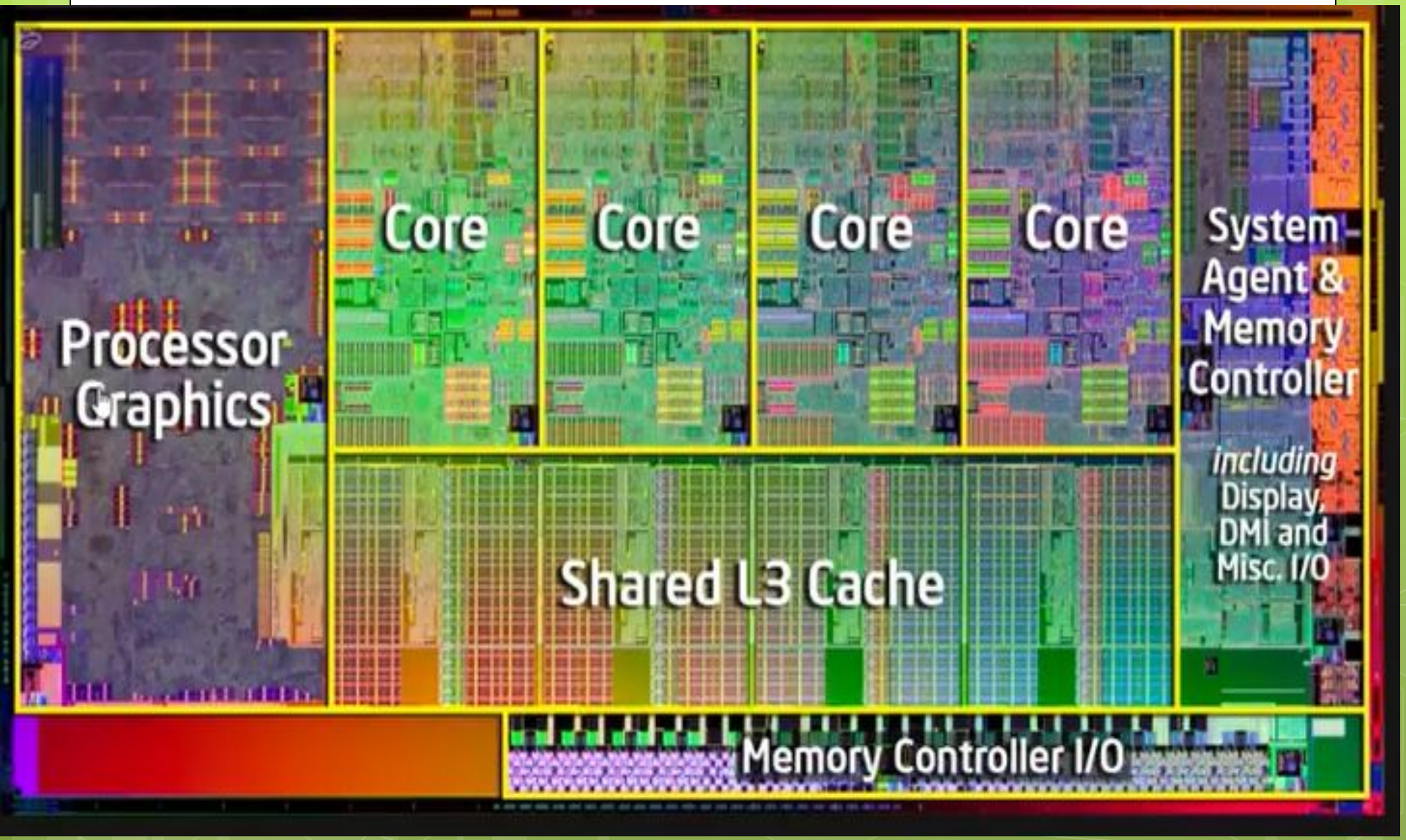


Рис. 2.22. Схема состава микропроцессора

Процессор под микроскопом



Введение

Вспомним…

Процессор и оперативная память образуют центральное устройство (ЦУ) ЭВМ.

Процессором является совокупность устройств, которая регулирует, управляет процессом обработки данных.

Сопряжение процессора, памяти и внешних устройств осуществляется через систему интерфейсов, реализующих коммутационно-коммуникационные функции.

3.1. Общее представление о структуре и архитектуре процессоров

Системы команд

Основные команды ЭВМ классифицируются следующим образом:

- **по функциям (выполняемым операциям),**
- **направлению приема-передачи информации,**
- **адресности.**

Очевидна связь таких параметров ЦУ, как длина адресного пространства, адресность, разрядность.

Увеличение разрядности позволяет увеличить адресность команды и длину адреса (т. е. объем памяти, доступной данной команде, например, в 32-разрядной машине, можно адресовать до 4 Гбайт ОП).

Увеличение адресности, в свою очередь, приводит к повышению быстродействия обработки (за счет снижения числа требуемых команд).

Классы процессоров

В зависимости от набора и порядка выполнения команд процессоры подразделяются на четыре класса:

CISC
(Complex Instruction Set Computer)

Классическая архитектура процессоров, которая начала свое развитие в 1940-х гг. с появлением первых компьютеров.

В CISC ЦП использует микропрограммы для выполнения большого набора разноформатных команд с использованием многочисленных способов адресации, для этого требуется наличие сложных электронных цепей для декодирования и исполнения.

Типичным примером CISC являются процессоры Intel x86 (в частности, семейство Pentium).

Количество команд: более 200 команд разной степени сложности.

Размер команд: от 1 до 15 байт,

Адресация: более 10 различных способов адресации.

Такое многообразие выполняемых команд и способов адресации позволяет программисту реализовать наиболее **эффективные алгоритмы решения различных задач.**

Однако при этом существенно **усложняется структура процессора**, особенно его устройства управления, что приводит к **увеличению размеров и стоимости кристалла**, снижению производительности.

В то же время анализ работы процессоров показал, что в течение примерно 80 % времени выполняется лишь 20 % общего набора команд. Поэтому была поставлена задача оптимизации выполнения небольшого по числу, но часто используемых команд.

RISC (Redused Instuction Set Computer)

Архитектура отличается использованием ограниченного набора команд фиксированного формата.

Первый процессор RISC был создан корпорацией IBM в 1979 г.

Современные RISC-процессоры обычно реализуют около 100 команд, имеющих фиксированный формат длиной 4 байта.

Значительно сокращается число используемых способов адресации.

В результате процессор на 20—30 % реже обращается к оперативной памяти, что также повышает скорость обработки данных.

Упростилась топология процессора, выполняемого в виде одной интегральной схемы, сократились сроки ее разработки, она стала дешевле.

Упрощается структура процессора, сокращаются его размеры и стоимость, значительно повышается производительность.

Начиная с процессора Pentium, корпорация Intel начала внедрять элементы RISC-технологий в свои изделия.

CISC или RISC?

В то время, как в процессоре CISC для выполнения одной команды необходимо в большинстве случаев десять тактов и более,

процессоры RISC близки к тому, чтобы выполнять по одной команде в каждом такте.

Следует также иметь в виду, что благодаря своей простоте процессоры RISC не патентуются. Это также способствует их быстрой разработке и широкому производству.

Процессор MISC

Работает с минимальным набором длинных команд и характеризуется небольшим набором чаще всего встречающихся команд.

Порядок выполнения команд распределяется таким образом, чтобы в максимальной степени загрузить маршруты, по которым проходят потоки данных.

Таким образом, архитектура MISC объединила вместе суперскалярную (многопоточную) и VLIW концепции. Компоненты процессора просты и работают с высокими скоростями.

VLIW (Very Large Instruction Word)

архитектура, которая появилась относительно недавно (в 1990-х гг.). Ее особенностью является использование очень длинных команд (до 128 бит и более), отдельные поля которых содержат коды, обеспечивающие выполнение различных операций.

Дорожная карта Супер-ЭВМ на базе архитектуры Эльбрус 2021-2027

РСК

Эльбрус-8СВ
28нм

5 поколение

0.29 TF FP64
8я @ 1.5 ГГц
ПСП 76 ГБ/с

Эльбрус-16С*
16нм

6 поколение

0.75 TF FP64
16я @ 2 ГГц
48 MB L2 + L3
ПСП 204 ГБ/с

Разработка:
МП Эльбрус-32С,
Сервер с воздушным охл.
Сервер с жидкостным охл.
Операционная система
Компилятор
Математические библиотеки
Разработка интерконнекта (ВНИИЭФ)

Адаптация СПО к архитектуре Эльбрус

Эльбрус-32С*
6нм

7 поколение

3 TF FP64
> 48я @ 2.5 ГГц
> 64 MB L2 + L3
ПСП > 300 ГБ/с**
(A64FX: 1ТБ/с)

Сервер 4хЭ8СВ

1.16 TF FP64
<500 Вт (СЖО)
44 лезвия
50 ТФ (ЦИАМ)
идёт поставка

Сервер 4хЭ16С

3 TF FP64 2U
<1000 Вт (воздух)
готов к серии

Сервер 4хЭ16С

3 TF FP64
<800 Вт (СЖО)
готов к серии

Сервер до 4хЭ32С

До 12 TF FP64
<1000 Вт (СЖО)
Опытные образцы

(*ТТ улучшены)

(**можно увеличить по запросу заказчика)

СуперЭВМ на Э8СВ

1 ПФ (864 лезвия,
6 шкафов)
возможна поставка

(*ТТ улучшены)

СуперЭВМ на Э16С

10 ПФ
(~3500 лезвий,
35 шкафов)
возможна поставка

СуперЭВМ на Э16С

+40 ПФ
(~14000 лезвий,
140 шкафов)
возможна поставка в теч. 2 лет

**СуперЭВМ н
Э32С**

100 ПФ
(~33000 МП,
16500 лезвий
по 2 МП
165 шкафов)
возможна
поставка в теч
2 лет

Компилятор

- В давние времена, когда компьютеры были большими, время доступа к памяти было небольшим, но и памяти было мало, поэтому была актуальна экономия регистров. Алгоритмы минимизации использования регистров разрабатывались еще основоположником оптимизирующей компиляции А. П. Ершовым и до сих пор не утратили своего значения — все современные оптимизирующие компиляторы оптимизируют распределение переменных по регистрам.

Специальный компилятор планирования перед выполнением прикладной программы проводит ее анализ. По множеству ветвей последовательности операций определяет группу команд, которые могут выполняться параллельно.

Каждая такая группа образует одну **сверхдлинную команду**. Это позволяет решать две важные задачи:

Во-первых, в течение одного такта выполнять группу коротких («обычных») команд,

Во-вторых — упростить структуру процессора. Этим технология VLIW отличается от суперскалярности (здесь отбор групп одновременно выполняемых команд происходит непосредственно в ходе выполнения прикладной программы, а не заранее, из-за этого усложняется структура процессора и замедляется скорость его работы).

К VLIW-типу можно отнести процессор **Elbrus**, объявленный **российской** компанией «Эльбрус».

3.2. Технологии повышения производительности процессоров и эффективности ЭВМ

Конвейерная обработка команд

Обработка команды, или цикл процессора, может быть разделена на несколько основных этапов (микрокоманд) ,

МикроКоманд минимум пять (выборка, декодирование, чтение исходных данных, выполнение, запись результата).

Каждая операция требует для своего выполнения времени, равного такту генератора процессора (tick of the internal clock).

Конвейерная обработка команд

Все команды в таких процессорах следуют одна за другой – это носит название конвейерной (pipeline) обработки.

Каждая часть устройства называется ступенью (стадией) к о н в е й е р а.

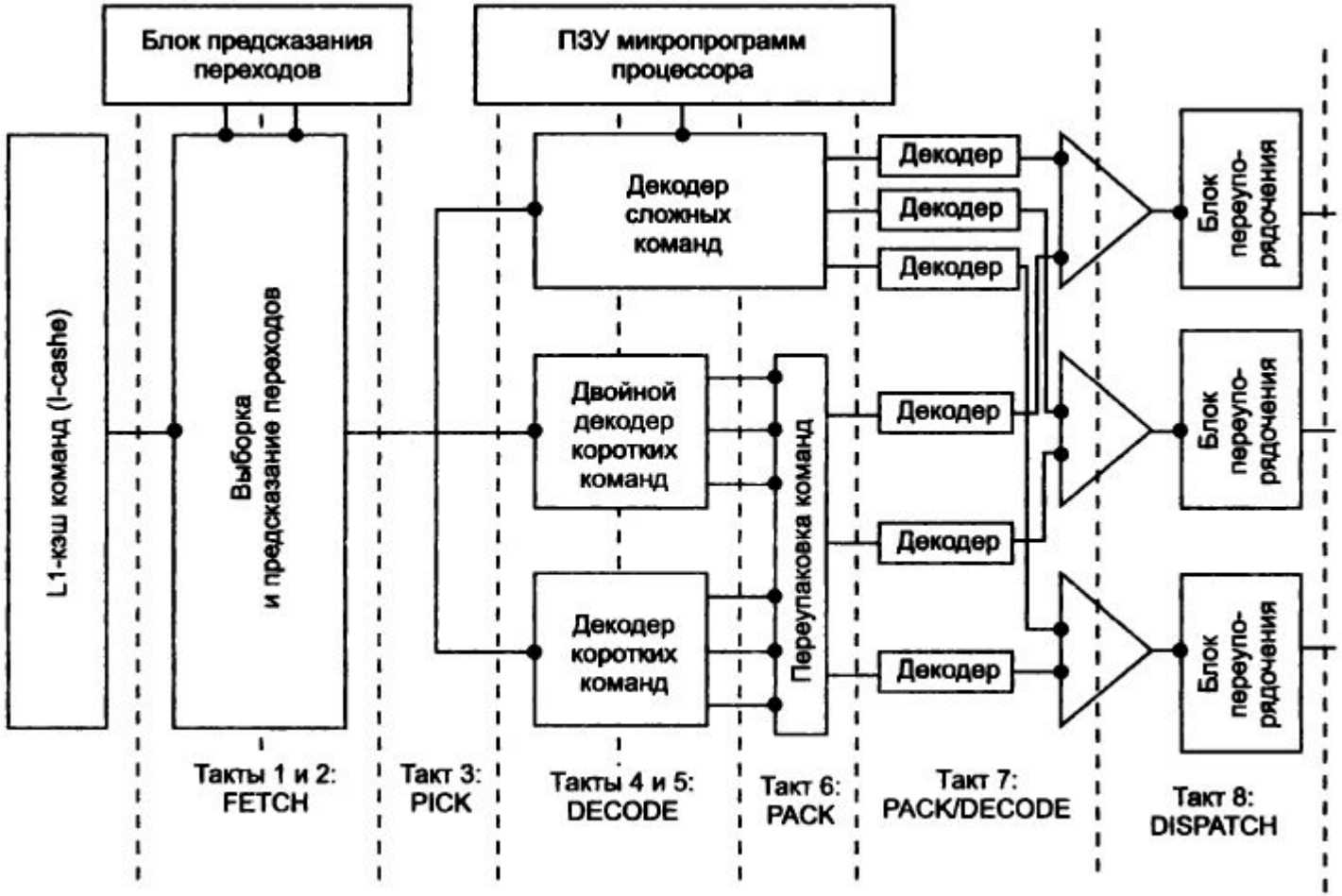
Общее число ступеней — длиной линии конвейера.

Конвейеризация

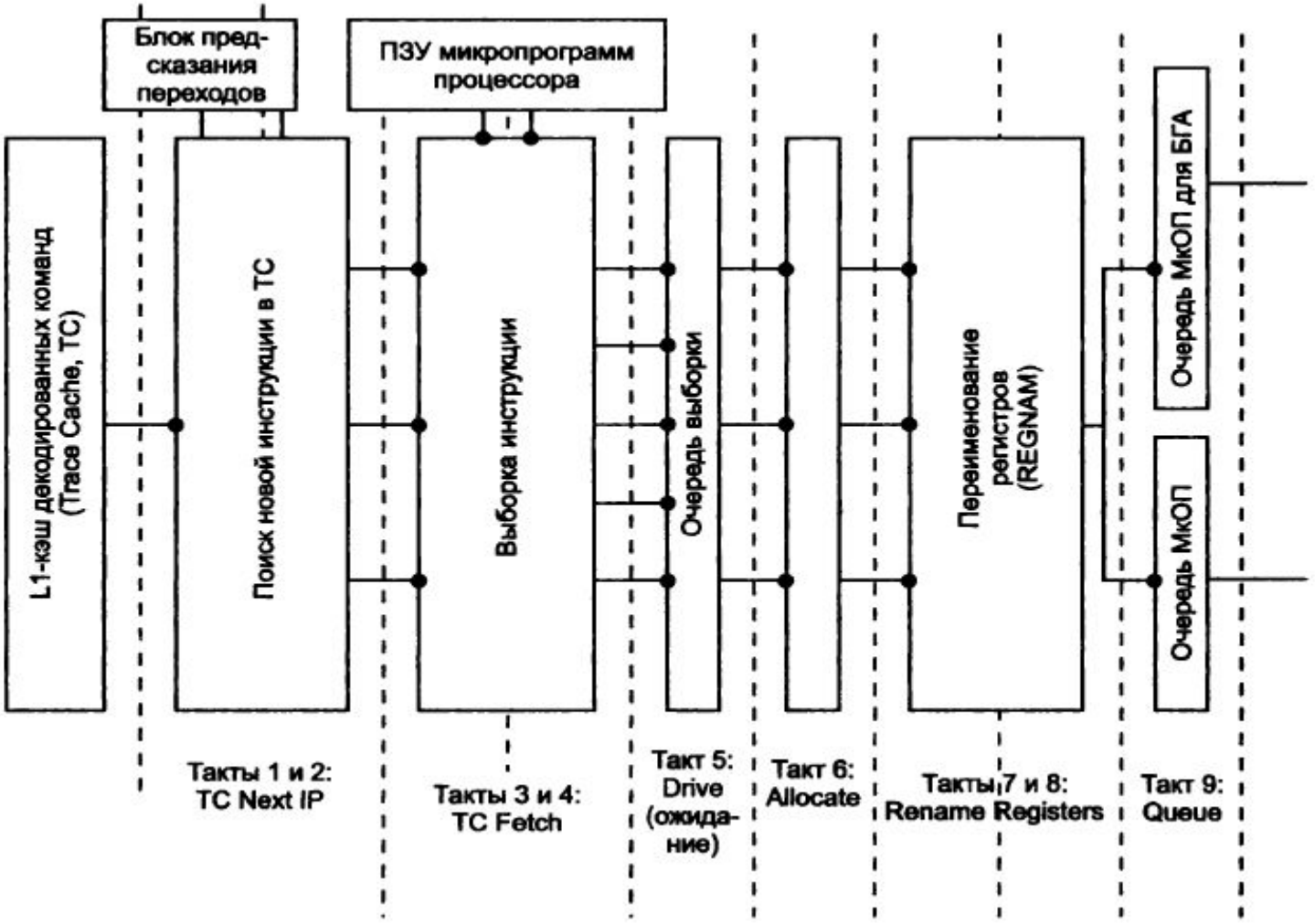
Конвейеризация *технология,* осуществляющая многопоточную параллельную обработку команд.

В каждый момент одна из команд считывается, другая декодируется и т. д., и всего в обработке одновременно находится пять команд.

Гипотетическая конвейеризация



Конвейеризация реальная



Пояснение к схемам

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.

Свойства конвейеризации

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.

У него две линии, что позволяет ему при одинаковых частотах быть вдвое производительней i80486, выполняя сразу две инструкции за такт

Суперскалярный процессор

Суперскалярный процессор - Процессор, поддерживающий так называемый параллелизм на уровне инструкций (то есть, процессор, способный выполнять несколько инструкций одновременно).

Во многих вычислительных системах, 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.

Сопроцессоры

Для расширения вычислительных возможностей центрального процессора — выполнения арифметических операций *над вещественными числами (с плавающей запятой)*, вычисления основных математических функций (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.

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 1980-е годы соответствие между типом ЭВМ и ее разрядностью имело простейший вид:

- микроЭВМ — 8 разрядов;
- мини-ЭВМ — 16 разрядов;
- большие ЭВМ — 32 разряда;
- сверхбольшие (супер) ЭВМ — 64 разряда.

В районе 2004 г. произошел переход на 64-разрядные архитектуры в процессорах Intel и AMD.

Преимущества 64-битовой архитектуры микропроцессоров главным образом относятся к памяти.

Если взять два идентичных микропроцессора, и один из них будет 32-битовым, а другой — 64-битовым, то последний сможет адресовать намного больший объем памяти, чем 32-битовый (2^{64} против 2^{32}).

Использование GPU

GPU - отдельное устройство персонального компьютера, выполняющее графический рендеринг.

GPU это вспомогательный микрочип, который берёт часть вычислительных операций на себя вместо процессора. И за счёт специализированной архитектуры, GPU лучше подходит для проведения расчётов с плавающей точкой, тогда как CPU больше ориентирован на работу в многопоточном режиме.

Видеокарта GPU способна быстро проводить расчёты, где используется одна или схожая формула (например, вычисление точки затенения графики при попадании тени на текстуру).

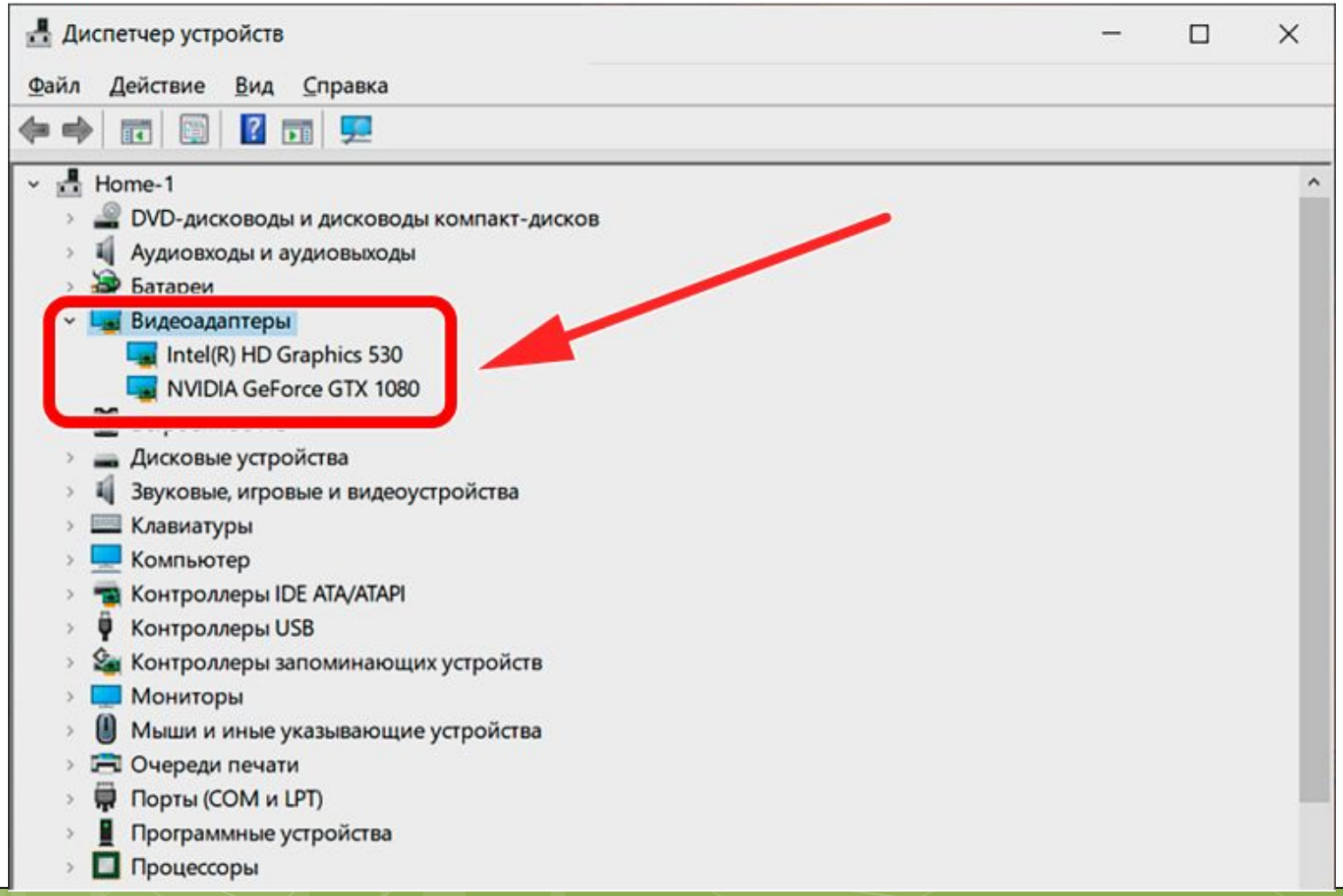
Центральный процессор же ориентирован на проведение расчётов сразу в несколько потоков, когда пользователь работает одновременно с большим количеством приложений.

В 95% случаев графический процессор используется для обработки графики. Он же отвечает за вывод изображения, отрисовку интерфейса операционной системы и каждой из запущенных программ.

Графический процессор интегрированный в CPU

Как по внешнему виду понять встроенная видео карта или внешняя

Для чего нужна внешняя видеокарта при встроенном GPU



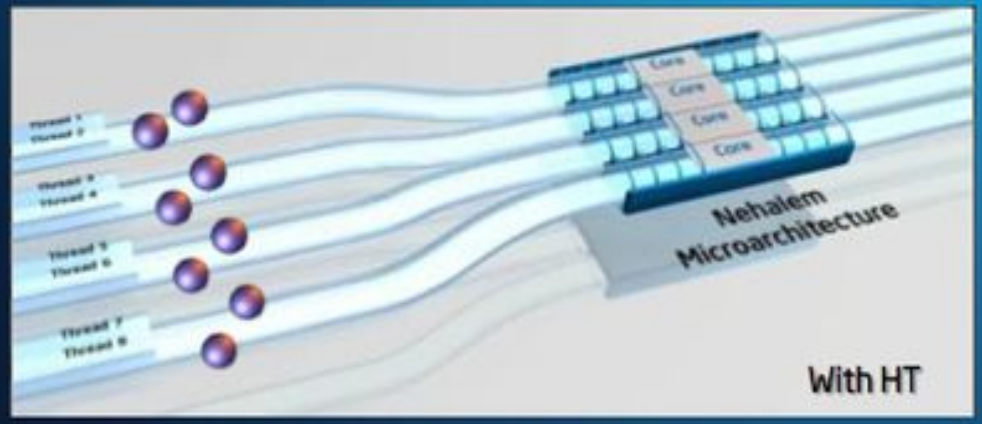
Использование виртуализации процессора

Hyper-threading (англ. *hyper-threading* — гиперпоточность, ***HTT*** или *HT*)

— технология, разработанная компанией *Intel* реализует идею «одновременной многопоточности».

HTT является развитием технологии суперпоточности.

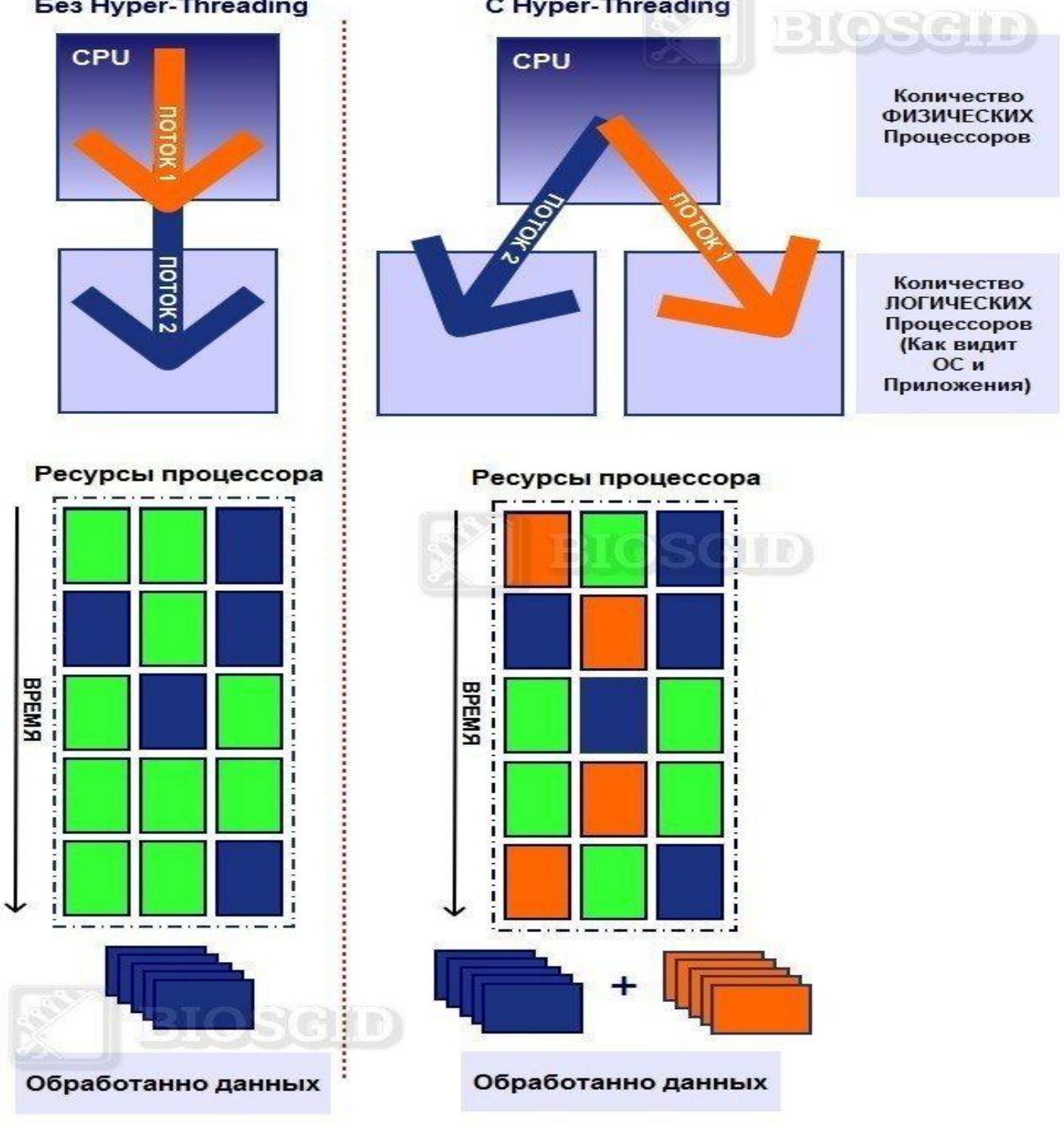
Intel® Hyper-Threading Technology For Better Multitasking[3]



2 thread engines per core, enabling 8-way processing in 4-core systems. It means that a quad-core processor could run up to eight threads simultaneously

- Helps you do more and wait less
- Like having a freeway with multiple car lanes instead of just one
- With multiple lanes cars more cars can move more quickly

intel


Без Hyper-Threading
CPU
ПОТОК 1
ПОТОК 2
С Hyper-Threading
CPU
ПОТОК 2
ПОТОК 1
Количество ФИЗИЧЕСКИХ Процессоров
Количество ЛОГИЧЕСКИХ Процессоров (Как видит ОС и Приложения)
Ресурсы процессора
ВРЕМЯ
Обработанно данных
Ресурсы процессора
ВРЕМЯ
+
Обработанно данных

Архитектуры на 64 разряда (64-bit architecture).

IA-64. **Спецификация IA-64 означает «Архитектура Intel, 64 бита».**

Появляется полностью отличный набор команд, Используются принципы VLIW.

IA-64 — архитектура, используемая линией процессоров Itanium.

Усовершенствования:

- **в 16 раз увеличено количество РОН и ПЗ (теперь по 128);**
- **механизм переименования/ротации регистров, чтобы сохранять значения в регистрах при вызове функций.**

AMD64

Набор команд AMD64 (первоначально названный x86-64), в значительной степени построен на основе IA-32 и таким образом обеспечивает наследственность семейства x86.

При расширении набора команд AMD воспользовалась возможностью, чтобы очистить часть его от ряда «устаревших» команд — наследия «16-разрядных времен».

Усовершенствования:

- **в 2 раза увеличено количество РОН и SSE (теперь по 16);**
- **РОН — теперь действительно регистры общего назначения и ничем больше не ограничены.**

Архитектура использована в ЦП Athlon 64, Athlon 64 X2, Athlon 64 FX, Opteron, Turion 64, Turion 64 X2, Sempron («Palermo», «Manila»).

EM64T

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

Intel начала использовать набор EM64T, начиная с ЦП Xeon (ядро Nocona) в конце 2004 г., а затем вышла с ним на рынок настольных ПК в начале 2005 г. (Pentium IV, версия E0). EMT/Intel 64 используется в ЦП архитектуры Intel NetBurst — Xeon («Nocona»), Celeron D («Prescott» и далее), Pentium 4 («Prescott» и далее), Pentium D, Pentium Extreme Edition и архитектуры Intel Core — Xeon («Woodcrest»), Intel Core 2.

Векторная обработка (SIMD-команды)

В классификации Г. Флинна имеется рубрика SIMD — поток данных, обрабатываемых одной командой.

Процессоры, реализующие такую обработку, именуют потоковыми процессорами.

Могут быть определены как однопотоковые (Single-streaming processor — SSP), так и многопотоковые процессоры (Multi-Streaming Processor — MSP).

В таких системах единое управляющее устройство контролирует множество процессорных элементов. Каждый процессорный элемент получает от устройства управления в каждый фиксированный момент времени одинаковую команду и выполняет ее над своими локальными данными.

MMX

MMX (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).

Анализ участков таких программ с большим объемом вычислений показал, что такие приложения имеют следующие общие свойства, определившие выбор системы команд и структуры данных:

- **небольшая разрядность целочисленных данных (например, 8-разрядные пиксели для графики или 16-разрядное представление речевых сигналов);**
- **небольшая длина циклов, но большое число их повторений;**
- **большой объем вычислений и значительный удельный вес операций умножения и накопления;**
- **существенный параллелизм операций в программах.**

Это и определило новую структуру данных и расширение системы команд. При этом было достигнуто общее повышение производительности на 10—20 %, а в программах обработки мультимедиа — до 60 %.

MMX-команды используются в ЦП, начиная с Pentium MMX (AMD K6) и имеют следующий синтаксис:

Архитектура 3DNow!

впервые реализована в процессорах AMD K6-2 (май 1998 г.). Технология 3DNow! включает 21 дополнительную команду, новые типы данных и использует регистры ммп (ммо—мм7) для поддержки высокопроизводительной обработки 3D-графики и звука.

В то время как архитектура MMX предполагает целочисленную арифметику, векторные команды 3DNow! Параллельно обрабатывают две пары 32-разрядных вещественных операндов одинарной точности

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.

Внеочередное выполнение

Внеочередное выполнение **(выполнение вне естественного порядка — out-of-order execution). 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.

Переименование (ротация) регистров

Переименование (ротация) регистров (register rename)

Чтобы избежать пересылок данных между регистрами в соответствующей команде изменяется адрес регистра, содержащего данные, участвующие в следующей операции.

Поэтому вместо пересылки данных в регистр-источник осуществляется переименование регистра с данными как источника.

Выполнение по предположению

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

Предикация (predication) **— одновременное исполнение нескольких ветвей программы вместо предсказания переходов (выполнения наиболее вероятного);**

- Если в исходной программе встречается условное ветвление (по статистике - через каждые 6 команд), то команды из разных ветвей помечаются разными предикатными регистрами, далее они выполняются совместно, но их результаты не записываются, пока значения предикатных регистров неопределены. Когда, наконец, вычисляется условие ветвления, предикатный регистр, соответствующий "правильной" ветви, устанавливается в 1, а другой — в 0.

Техника, подобная предикации, используется в RISC процессорах

Опережающее чтение данных

Опережающее чтение данных (speculative loading), т. е. загрузка данных в регистры с опережением, до того, как определилось реальное ветвление программы (переход управления).

Эти возможности осуществляются комбинированно — при компиляции и выполнении программы.


1. Компилятор сканирует исходную программу и обнаруживает чтение данных (команда 8, после перехода). Она удаляется. Помещается опережающее чтение перед переходом. После перехода вставляется проверка
Команда 1
Команда 2
Чтение с опережением
Команда 3 (переход)
2. В момент исполнения программы эта команда считывает данные, которые могут понадобиться в одной из ветвей
5. Реально команда чтения выполняется перед ветвлением программы
Команда 4
Команда 5
Команда 6
3. Если исполнение идет по данной ветви, то поскольку данные уже считаны, команда 8 реально выполняться не будет
Команда 7
Команда 8 (чтение)
4. Проверка действительности считанного данного
Команда проверки
Команда 9 (обработка)