Урок 2. Процессоры.

1. Набор команд процессора

Сегодня мы начинаем изучать важнейший компонент современного компьютера - процессор. Давайте в первую очередь определимся, чем можно характеризовать некий процессор.

Пожалуй, важнейший параметр процессора - набор команд, который он умеет исполнять. Разумеется, все процессоры, на базе которых строится PC - совместимый компьютер, должны уметь исполнять одинаковый набор команд. Напомню, что компьютерная программы - не что иное, как последовательность некоторых команд, и, естественно, каждую из этих команд должен уметь исполнить процессор. Процессоры, на базе которых строятся другие (не PC) компьютеры, исполняют свои, совсем другие наборы команд. Как называют тот набор команд, который исполняет PC? Для того чтобы ответить на этот вопрос, нам придется немного заглянуть в историю.

В июне 1978 года, уже упоминавшаяся нами фирма Intel выпустила процессор, который назвала i8086, который (точнее его близкого наследника) IBM и применила в своих первых PC. Затем Intel выпустила процессор i80186, который обладал повышенной производительностью. В феврале 1982 выходит новый процессор i80286, обладающий рядом нововведений, относительно предыдущих процессоров и имеющий снова более высокую производительность, но при этом, само собой разумеется, совместимый по командам с прошлыми процессорами. В июне 1988 появляется новое поколение процессоров фирмы Intel, и эти процессоры маркируются i80386. Наконец в 1991 выходит процессор i80486, обладающий еще более высокой производительностью. Естественно, все перечисленные процессоры, устанавливавшиеся в PC, умели выполнять одинаковый набор команд, иначе они не были бы совместимы между собой программно (т.е. программы, запускающиеся на одном процессоре, не запускались бы на другом). А набор команд, исполняемый всеми этими процессорами, принято называть по серии, которой нумеровались выходившие процессоры. Такой набор команд принято называть **x86** . Таким образом, процессоры, устанавливаемые в PC, называют **x86 - процессорами**, а саму архитектуру PC нередко называют **x86 - архитектурой**.

Помимо такой важнейшей архитектурной особенности, как набор команд, процессор, разумеется характеризуют быстродействием. В каких единицах приято измерять быстродействие процессора? Этот вопрос не имеет однозначного ответа. Быстродействие зависит от разных факторов, и мы сейчас рассмотрим эти факторы.

Элементы архитектуры процессора

2. Тактовая частота

Быстродействие процессора во многом зависит от тактовой частоты, обычно измеряемое в **мегагерцах** (**МГц**). Тактовая частота определяется параметрами кварцевого резонатора, представляющего собой кристалл кварца в оловянной оболочке. Под воздействие электрического напряжения в кристалле кварца возникают колебания электрического тока с частой, определяемой формой и размерами кристалла. Частота этого переменного тока и называется тактовой частотой. Наименьшей единицей времени для

процессора, как для логического устройства является период тактовой частоты или просто такт. На каждую операцию (выполнение команды) процессор затрачивает некоторое количество тактов. Естественно, чем выше тактовая частота процессора, тем производительнее он работает, так как в единицу времени происходит большее количество тактов и выполняется большее количество команд. Естественно, более новые процессоры работают на все более высоких тактовых частотах (это достигается, в частности, улучшением технологии их изготовления) показываю большую производительность.

Но тактовая частота не единственный фактор, определяющий производительность процессора. Ведь количество тактов, затрачиваемое на выполнение команд тоже можно менять. И если первые x86 процессоры на выполнение одной команды тратили в среднем около 12 тактов, в 286 и 386 этот показатель в среднем составлял около 4,5 тактов, в 486 - около 2 тактов, то в процессорах Pentium в среднем выполняется одна команда за такт. Ну а процессоры семейства Pentium 4 могут выполнять несколько команд одновременно (за счет параллельного исполнения команд).

Различное количество тактов, затрачиваемое процессорами на выполнение команд, затрудняет их сравнение с использованием только лишь тактовой частоты. Гораздо удобнее использовать для измерения производительности среднее количество операций, выполняемое за один такт работы процессора. Есть единица измерения этого количества – **MIPS** – million integer per second, миллионы целочисленных операций в секунду, и **MFLOPS** - million floating per second, миллионы вещественных операций в секунду. Этими величинами измеряется производительность разных блоков процессора – **ALU** и **FPU**. ALU – арифметико-логическое устройство, выполняет обработку целых чисел, это основной вычислительный блок процессора, именно он определяет используемый основной набор команд, напомним, что для рассматриваемых нами процессоров – это **x86**. A FPU – floating-point unit, обрабатывает вещественные числа (с плавающей запятой), это дополнительный блок, который также называют математическим сопроцессором, о нём мы поговорим позже.

Как вы уже наверное догадались теоретически сравнивать два процессора нужно рассматривая быстродействие и тактовую частоту работы в совокупности: чем меньше тактов затрачивает в среднем процессор на исполнение команды, тем выше его эффективность (производительность) даже при неизменной тактовой частоте. Например: 486 процессор (в среднем 2 такта на команду) на частоте 133 МГц работает даже медленнее, чем процессор Pentium (в среднем 1 такт на команду) на частоте 75МГц. Оценивать реальную производительность процессора в сравнении с другими весьма непросто, и нужно понимать, что такое сравнение во многом зависит от той задачи, которую процессор решает.

Тактовые частоты современных процессоров лежат в диапазоне от 1,5 ГГц до 3,8 ГГц, и более.

Перечисленные выше характеристики во многом отражают эффективность процессора. Но есть еще целый ряд характеристик, описывающих внутреннюю архитектуру процессора, и сейчас мы рассмотри некоторые важнейшие из них.

3. Шина данных процессора

Одной из самых общих характеристик процессора является разрядность его шины данных и шины адреса. (Вспомните, что такое шина).

Когда говорят о шине процессора, обычно имеют ввиду шину данных, которая является набором соединений, для передачи и приема данных. Чем больше сигналов одновременно поступает на шину, тем больше данных по ней передается за определенный интервал времени, и тем быстрее она работает. Разрядность шины данных подобна количеству полос автомагистрали - чем больше полос, тем больше поток машин, чем шире шина данных, тем больше данных за одинаковые промежутки времени по ней передается. В процессоре 286 для приема и передачи двоичных данных используется 16 соединений, поэтому их шина данных считается 16-разрядной. У 32-х разрядных процессоров (например, 486), таких соединений вдвое больше, поэтому за единицу времени они передают и

получают вдвое больше данных, чем 16-и разрядные процессоры - разумеется, эффективность выше. Современные процессоры (начиная с Pentium) имеют 64-х разрядную шину данных, поэтому они могут передавать в системную память по 64 бита за один такт. Такая реализация позволяет ускорить обмен данными между быстрым процессором и относительно медленным ОЗУ при неизменной рабочей частоте последнего за счёт повышения пропускной способности шины данных (вспомните, что такое пропускная способность шины).

Таким образом мы определили, с помощью какой шины процессор связан с оперативной памятью и от разрядности этой шины конечно же зависит производительность процессора. Теперь давайте разберемся, как процессор обрабатывает полученные из оперативной памяти данные.

4. Шины и регистры процессора

Хоть процессор и получает данные из оперативной памяти с помощью шины некоторой ширины, это не значит, что внутри он может обрабатывать данные такой же разрядности. Давайте разберемся, как это происходит.

Количество битов данных, которые может обработать процессор за один прием, характеризуется разрядностью внутренних регистров.

Регистр - это по существу ячейка памяти внутри процессора, например, процессор может складывать числа, записанные в двух разных регистрах, а результат записывать в третий регистр. Разрядность регистров описывает разрядность обрабатываемых процессором данных. Разрядность регистров определяет также характеристики программного обеспечения и команд, выполняемых процессором. Например, процессоры с 32-разрядными внутренними регистрами могут выполнять 32-разрядные команды, которые обрабатывают данные 32-разрядными порциями, а процессоры с 16-разрядными регистрами этого делать не могут.

В современных процессорах для PC внутренние регистры являются 32 или 64 разрядными, причем 64-битных существует несколько вариантов – IA-64 для процессоров Itanium, . В некоторых старых процессорах разрядность внутренней шины данных (а шина состоит из линий передачи данных и регистров), больше, чем разрядность внешней шины данных, той, которую мы с Вами уже обсуждали (которая связывает процессор с оперативной памятью). Обычно такие процессоры являются более дешевыми вариантами своих старших собратьев. Такая архитектура (внутренняя шина и регистры вдвое шире внешней) позволяет проектировать и создавать, например, недорогие 16-разрядные материнские платы, устанавливая в них 32-разрядные процессоры, обеспечивая таким образом 32-разрядную совместимость процессора при 16 разрядном обмене с памятью. Но такой способ удешевления системы остался в прошлом и в настоящее время совершенно не применяется.

В современных процессорах все обстоит наоборот: внешняя шина данных, как мы уже говорили, 64-разрядная, а регистры и внутренняя шина процессора по-прежнему 32-разрядны. Странная ситуация, не правда ли? Но странной она кажется лишь до того момента, как мы узнаем, что в современном процессоре (например, Pentium) для обработки информации, поступающей по внешней 64-разрядной шине данных, ALU состоит из двух 32-разрядных блоков, которые называются конвейерами (подробнее в п.6), а 64-разрядная внешняя шина данных позволяет быстрее наполнить регистры процессора. Такая архитектура, применяющая для обработки поступивших данных несколько, называется **суперскалярной** и применяется сегодня во всех современных процессорах.

У процессоров с 64 разрядными регистрами ситуация с внешней и внутренней шинами несколько иная, например 64-битные процессоры фирмы AMD вообще имеют встроенный контроллер памяти, который работает с 64-битными модулями, и ширина шины памяти у них зависит от количества этих контроллеров.

Шина адреса. Шина адреса представляет собой набор проводников, по которым контроллеру памяти передается адрес ячейки памяти, в которую или из которой

пересылаются данные. По каждому проводнику передается один бит адреса, соответствующий одной цифре в адресе. Увеличение количества проводников (разрядов шины) используемых для формирования адреса, позволяет увеличить количество адресуемых ячеек. Разрядность шины адреса определяет максимальный объем памяти, адресуемой процессором. В компьютерах применяется, как Вы знаете, двоичная системы счисления. Если, например, разрядность шины адреса составила бы всего один бит (один провод для передачи данных), то по этому проводу можно было бы передать всего два значения (логический нуль - нет напряжения, логическая единица - есть напряжение) и таким образом можно было бы адресоваться к двум ячейкам памяти. Такой бы процессор поддерживал обмен только с двумя байтами оперативной памяти! Использование двух бит для задания адреса позволило бы адресоваться уже к 4-м байтам памяти (00, 01, 10, 11 на шине - на четыре разных адреса можно указать). Вообще, количество разных значений, принимаемое n-разрядным двоичным числом равно 2 в степени n. Соответственно, при ширине шины адреса n бит, количество разных ячеек памяти, к которым можно адресоваться составляет 2 в степени n, поэтому говорят, что процессор поддерживает 2 в степени n байт оперативной памяти, или говорят, что адресное пространство процессора равно 2 в степени n байт. Например: процессор 8086 имел адресную шину 20 бит. Тогда он мог адресовать (2 в степени 20=1048576) байт оперативной памяти, т.е. 1 Мбайт. Таким образом, максимальный объем оперативной памяти, поддерживаемой процессором 8086 составляет 1 Мбайт. 286-ой процессор имел адресную шину равную 24 битам, адресуя таким образом уже 16 Мбайт (обратите внимание: каждый новый бит в шине адреса, увеличивает объем адресуемой памяти вдвое, это естественно, если вспомнить формулу Объем памяти = 2 в степени разрядность шины). Современные процессоры имеют адресную шину равной 36 бит, что соответствует поддерживаемой оперативной памяти объемом 64 Гбайт!

Шины данных и адреса независимы, и разработчики микросхем выбирают их разрядность по своему усмотрению. Разрядность этих шин является показателем возможностей процессора: разрядность шины данных определяет возможности процессора быстро обмениваться информацией, разрядность адресной шины определяет объем поддерживаемой процессором памяти.

5. Частота и множитель процессора

Вы наверняка знаете, что современный процессор работает сегодня на частотах порядка 2…3 ГГц. С другой стороны оперативная память работает на сегодня на гораздо более низких частотах. Более того, не только оперативная память, но и чипсет работает на невысокой относительно процессора частоте. На материнской плате вообще нет компонентов, которые работали бы быстрее, чем системная шина и чипсет (точнее его северный мост). Впрочем, это и не удивительно.

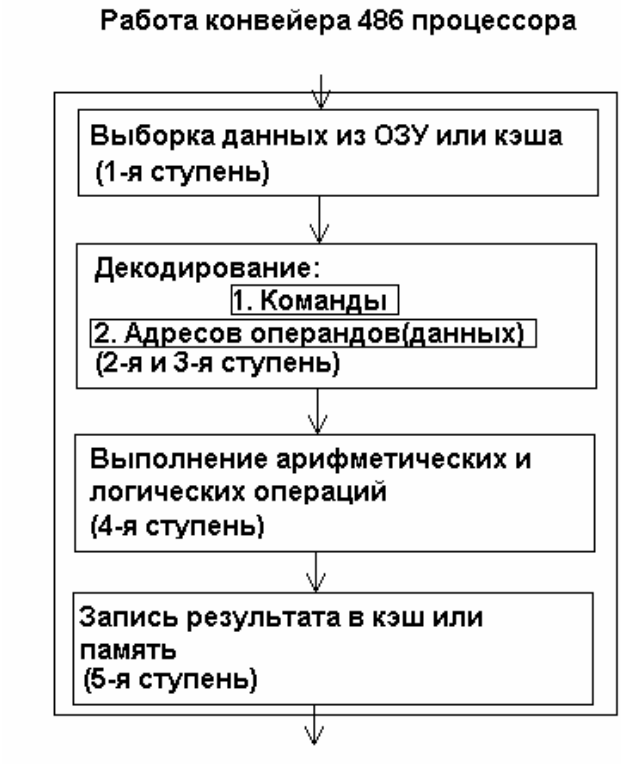
Для того, чтобы, имея систему с 2,0 ГГц процессором, улучшить ее до 2,8 ГГц, нам достаточно лишь заменить процессор (естественно, учитывая при этом целый ряд факторов, но обо всем этом мы поговорим позже, давайте пока рассуждать упрощенно). Итак, вы извлекаете из процессорного гнезда 2,0 ГГц процессор и устанавливаете 2,8 ГГц. Вопрос: а материнская плата работает с новым процессором в прежнем режиме, в том же, в котором она работала и с процессором 2,0 ГГц? Если плата теперь работает втрое быстрее, то спрашивается: если для того, чтобы ускорить процессор его нужно заменить, значит, чтобы ускорить во столько же раз материнскую плату ее, наверное, тоже нужно заменить на втрое более производительную? Т.е. простой заменой процессора систему нельзя улучшить? Нужно заменять материнскую плату, на новой плате должен быть другой, способный работать на утроенной частоте чипсет, нужно заменить и оперативную память? Если бы все было так, как мы сейчас описали, то компьютер вообще бы практически не подлежал бы улучшению (upgrade), а можно было бы лишь заменить его практически целиком.

Но это не так. В большинстве систем предел скорости на которой работает чипсет, составляет 800 МГц (200 МГц QDR). Тогда возникает главный вопрос: каким образом в ОДНОЙ И ТОЙ ЖЕ материнской плате могут работать и 2,0 ГГц и 2,8 ГГц процессора? Если

частота системной шины никогда не превышает 800 МГц, как процессор работает на более высокой частоте? Ответ кроется в том, что процессор НЕ работает на частоте системной шины (на этой частоте работает вся материнская плата, чипсет, память, но не процессор). Процессор лишь использует для своей работы частоту системной шины. Дело в том, что процессор УМНОЖАЕТ эту частоту на некоторый множитель, таким образом получая результирующую частоту, на которой и работает. Например: Процессор, работающий на частоте 2,0 ГГц использует системную шину на частоте 200 МГц QDR, умножая ее на 10, а процессор, работающий на частоте 2,8 ГГц использует ту же частоту системной шины, умножая ее на 14.

Таким образом, в одну и ту же материнскую плату можно вставить различные процессоры, работающие на разных частотах за счет того, что частота системной шины используется одна и та же (или меньше, например: 2,0 ГГц процессора = 200 МГц системной шины х 10), а процессор производит в себе умножение частоты поданной на него системной шины на некоторый, закрепленный в конкретном процессоре, множитель. Еще один важнейший момент. Какая система быстрее? Процессор 2 ГГц = 100 МГц х 20 или 2 ГГц = 200 МГц х 10? Иными словами, влияет ли частота системной шины на производительность системы при неизменной частоте процессора? Естественно влияет. Разумеется, компьютер, у которого системная шина работает на частоте 200 МГц, при прочих равных условиях будет работать быстрее системы с тем же процессором, но на частоте системной шины 100 МГц. Ведь и память, и прочие компоненты предпочтительнее использовать на более высокой частоте (естественно, если это позволяет их спецификация, если они, иными словами, для такой частоты предназначены). Т.е. компьютер производительнее не только потому, что в нем установлен более быстродействующий процессор; производительность так же зависит и от частоты системной шины, на которой, в свою очередь, работает оперативная память. Важное замечание: естественно, производители чипсетов (а именно от чипсета зависит, какую частоту системной шины поддерживает материнская плата, от чипсета почти все зависит :)) постоянно стремятся увеличить частоту системной шины, поддерживаемую их продуктами, так как даже при очень быстром процессоре система не может быстрее обмениваться данными с памятью, чем позволяет системная шина и это ограничивает производительность всей системы. И, изучая с Вами в ближайшее время чипсеты, мы увидим, какое значение придается поддержке в сегодняшнем чипсете максимально возможной частоте системной шины и типу поддерживаемой памяти, так как, в конечном счете, борьба за быструю системную шины - это во многом борьба за ускорение обмена память - процессор.

6. Конвейерное исполнение



Современные процессоры обрабатывают данные конвейерным способом, т.е. одновременно в процессоре на различных стадиях происходит выполнение нескольких команд - одна команда практически близка к исполнению, другая - в середине процесса исполнения, третья только входит на конвейер и т.д., на рисунке пример первого из применяемых в PC конвейера. Таким образом удается значительно повысить производительность, так как все части процессора одновременно задействуются, и каждая часть процессора выполняет свой маленьких объем работы. При этом вводят понятие "количество стадий конвейера" - число элементарных задач на которые разбивается выполнение команды – на рисунке выделена каждая стадия. При этом если

стадий мало, то недостаточно полно используются все части процессора одновременно (страдает производительность), и невозможно использовать большие тактовые частоты. А если стадий много, то появляется возможность использовать более высокие частоты процессора, но при этом увеличиваются задержки в процессе исполнения команд (тоже страдает производительность). Таким образом, производители процессоров выбирают некоторое оптимальное количество стадий конвейера, при котором процессор показывает наивысшую производительность. Количество стадий в конвейерах современных процессоров от 15 до 30, при этом производители столкнулись с ещё одной проблемой – хоть частоты процессоров одних производителей благодаря длинным конвейерам (30 стадий) выросли более 3-х гигагерц, но их производительность сравнима с процессорами конкурентов, у которых конвейер 15 стадий, и частоты намного меньше.

Также, при работе конвейера возникает проблема, связанная с исполнением команд ветвления (условных переходов): если в процессор поступила команда ветвления, то ответ на вопрос о том, какие следующие команды должны быть исполнены будет известен ТОЛЬКО после того, как исполниться команда ветвления, но пока она не исполнена, на конвейер могли бы поступить другие команды, которые бы постепенно исполнялись! Получается, что пока не будет выполнена команда ветвления, конвейер должен простаивать! И чем длиннее конвейер, тем дольше он простаивает. Решением этой проблемы стало **динамическое исполнение** команд.

Динамическое исполнение - совокупность трех методов обработки информации в процессоре, таких как: Предсказание ветвлений, анализ потока команд, упреждающее выполнение. Динамическое выполнение - важнейшее архитектурное преимущество современных процессоров, оказывает серьезное влияние на производительность. Давайте вкратце рассмотрим, что такое динамическое выполнение.

Предсказание ветвлений (branch prediction)

С помощью этого метода можно выяснить, каким будет поток управления программы через несколько команд ветвления. При использовании специального механизма процессор может предсказать переходы или ветвления в потоке команд. В процессорах прежних поколений инструкция перехода приостанавливала конвейер (выборку инструкций) до исполнения собственно перехода, на чем, естественно, терялась производительность. Предсказание переходов направляет поток выборки и декодирования по одной из ветвей. Статический метод предсказания работает по схеме, заложенной в процессор, считая, что переходы по одним условиям, вероятнее всего, произойдут, а по другим — нет. Динамическое предсказание опирается на предысторию вычислительного процесса — для каждого конкретного случая перехода накапливается статистика поведения, и переход предсказывается, основываясь именно на ней. Этот метод позволяет существенно ускорить выполнение программы, но при этом возникает проблема – в случае, если предсказание окажется неверным, всё содержимое конвейера тоже окажется неверным, и его придется очищать. А чем длиннее конвейер, тем больше будет потеря времени в результате его очистки, поэтому производители процессоров постоянно совершенствуют механизм предсказания ветвлений.

Анализ потока команд (out-of-order execution)

Это средство анализирует и планирует выполнение команд в оптимальной последовательности, независимо от их первоначального порядка в программе. Свойственный RISC-архитектуре, данный метод теперь реализуется и для процессоров x86. При этом изменяется порядок внутренних манипуляций данными, а внешние (шинные) операции ввода-вывода и записи в память выполняются, конечно же, в порядке, предписанном программным кодом. Однако эта способность процессора в наибольшей степени может блокироваться несовершенством программного кода (особенно 16-битных приложений), если он генерируется без учета возможности изменения порядка исполнения инструкций. Процессор рассматривает команды, из которых состоит выполняемая программа, и определяет, доступны ли они для обработки или же зависят от других команд,

которые следует выполнить предварительно. Затем процессор определяет оптимальную последовательность обработки и выполняет команды наиболее эффективным способом.

Упреждающее выполнение (data forwarding)

Подразумевает начало исполнения инструкции до готовности всех операндов. При этом выполняются все возможные действия, и декодированная инструкция с одним операндом помещается в исполнительное устройство, где дожидается готовности второго операнда, выходящего с другого конвейера. С помощью этого метода процессор просматривает стоящие на очереди команды и выполняет те из них, к которым вероятно потребуется обратиться позже. Таким образом ряд команд процессор может выполнить заранее, а затем пользоваться результатами произведенных вычислений позже.

7. Сопроцессор, дополнительные наборы команд.

Набор команд x86, который мы с Вами уже обсуждали, имеет следующую особенность - он ориентирован на работу с целыми числами. Как быть, если процессору нужно извлечь квадратный корень, найти синус или логарифм? Естественно, что процессор может справиться с такой задачей, но, учитывая то, что он ориентирован на вычисления с целыми числами, выполнение такой операции займет у него много тактов.

В то же время Intel для всех своих процессоров разрабатывает так называемый сопроцессор - кристалл, который тоже умеет выполнять команды, но не x86, а другие, и поддерживаемый сопроцессором набор команд (называемый **x87**) ориентирован на работу с числами с плавающей запятой, таким образом, он (сопроцессор) перечисленные выше задачи как раз и призван решать. На заре развития PC считалось, что сопроцессор нужен небольшому количеству пользователей (действительно, зачем пользователю текстового редактора математические вычисления) и устанавливался в систему дополнительно; при желании пользователь мог отдельно приобрести сопроцессор и установить его в специальное гнездо на материнской плате. Сегодня ситуация полностью изменилась. Сегодня сопроцессор нужен абсолютно всем пользователям: "в наш век мультимедиа" когда компьютер все больше и больше способен обрабатывать реалистичную 3-х мерную графику, математические расчеты становятся неотъемлемым атрибутом любого мультимедиа приложения (например, современной игры). Поэтому достаточно давно сопроцессор стали устанавливать вместе с процессором на один кристалл. Действительно, раз сопроцессор все равно нужен всем пользователям, то его разумно встроить в процессор, а не изготавливать отдельно: разумеется, это будет дешевле.

На заре развития PC считалось, что сопроцессор нужен небольшому количеству пользователей (действительно, зачем пользователю текстового редактора математические вычисления) и устанавливался в систему дополнительно; при желании пользователь мог отдельно приобрести сопроцессор и установить его в специальное гнездо на материнской плате. Сегодня ситуация полностью изменилась. Сегодня сопроцессор нужен абсолютно всем пользователям: "в наш век мультимедиа" когда компьютер все больше и больше способен обрабатывать реалистичную 3-х мерную графику, математические расчеты становятся неотъемлемым атрибутом любого мультимедиа приложения (например, современной игры). Поэтому достаточно давно сопроцессор стали устанавливать вместе с процессором на один кристалл. Действительно, раз сопроцессор все равно нужен всем пользователям, то его разумно встроить в процессор, а не изготавливать отдельно: разумеется, это будет дешевле.

Таким образом, любой современный процессор поддерживает два основных набора команд: **x86** и **x87**. Могут ли производители процессоров отказаться от этих наборов команд? Нет! Тогда получившаяся система уже не будет программно совместима с PC, так как программное обеспечение, написанное для PC на таком процессоре, уже не будет работать! Поддержка этих двух наборов команд - залог программной совместимости. А могут ли производители процессоров создавать и добавлять в процессор новые наборы

команд? Конечно. Но что нужно для того, чтобы программы стали работать эффективнее на процессоре с новым набором команд? Будет ли у старого приложения, которое проектировали еще до разработки нового набора команд какие-то преимущества от исполнения на процессоре с новым набором команд? Нет! Ведь программа - это не что иное, как последовательность команд процессору. Если в программе нет ни одной новой команды (а откуда ей там взяться, если в момент написания программы новых команд еще не придумали), то естественно никакой пользы из того, что процессор может исполнять новые команды, старое приложение не извлечет. Поэтому, когда в процессор добавляют новый набор команд, нужно понимать, что пока разработчики программного обеспечения не начнут писать программы с учетом нового набора команд НИКАКОЙ пользы от него не будет. А старым программам от нового набора команд уже никогда не будет пользы (разве что авторы перепишут старую программу с учетом нового набора команд).

Процессор, изначально, задумывался как универсальное устройство, которое могло с помощью изменения программы, решать любые задачи, будь то решение математических уравнений, игра, программа набора текста и др. Но со временем, к некоторым областям стали предъявляться особые требования по быстродействию. Для решения, некоторых узких задач, были введены дополнительные наборы команд - расширения основного набора x86.

Набор инструкций 3DNow!/Enhanced 3DNow!

Технология 3DNow! разработана фирмой AMD для процессоров K6-2, конкурента Pentium MMX и Pentium II. Направлена эта технология, как видно из названия, на ускорение работы с трехмерной графикой, а также другие вычисления связанные с потоковыми видео- и аудиоданными, хотя по сути является просто математическим набором команд. Основная цель, которую преследовала AMD при создании набора команд 3DNow! – создание альтернативы x87, т.к. сопроцессоры, которые применялись фирмой AMD для своих процессоров, были недостаточно производительными в сравнении с сопроцессорами Intel, разработчиком x87, а трёхмерная графика и игры являются основными потребителями математических команд. И если заметить, именно игры для большинства пользователей являются двигателем к выбору аппаратной части компьютера, поэтому AMD сделали рискованный маркетинговый ход – внедрили новый набор команд в свои процессоры, и начали активно преподносить свои процессоры как наиболее оптимизированные под игровой процесс. Как мы уже говорили, успех того или иного набора команд целиком зависит от его поддержки программным обеспечением, поэтому для успеха своих процессоров AMD сделали ставку именно на игры. Ее дальнейшее развитие - Enhanced 3DNow!, появилась в процессорах Athlon и Duron.

Технология MMX

В зависимости от контекста MMX может означать multi-media extensions (мультимедийные расширения) или matrix math extensions (матричные математические расширения). Технология MMX начала использоваться в старших моделях процессоров Pentium пятого поколения в качестве расширения, благодаря которому ускоряется компрессия/декомпрессия видеоданных, манипулирование изображением, шифрование и выполнение операций ввода-вывода - почти все операции, используемые во многих современных программах. MMX это расширение основного набора команд на 57 новых команд, а также во введении новой возможности выполнения команд, называемой одиночный поток команд - множественный поток данных (Single Instruction - Multiple Data, SIMD).

Набор инструкций SSE/SSE2/SSE3

SSE (Streaming SIMD Extensions - потоковые расширения SIMD), обновление технологии MMX, появились в Pentium III. Содержит 70 новых инструкций для работы с графикой и звуком в дополнение к существующим командам MMX. Инструкции SSE подобны инструкциям MMX и предварительно назывались MMX-2. Операции с плавающей точкой SSE

реализованы в виде отдельного модуля в процессоре. Новые инструкции SSE позволяют более эффективно работать с трехмерной графикой, потоками аудио- и видеоданных, приложениями распознавания речи. Набор команд SSE2, который появились в Pentium 4 - это опять расширенный на новые 144 команды набор команд SSE, в первую очередь направлен на потоковые вычисления. И SSE3, появившийся в последних моделях Pentium 4, также представляет собой очередное расширение SSE.

8. Кэш – память

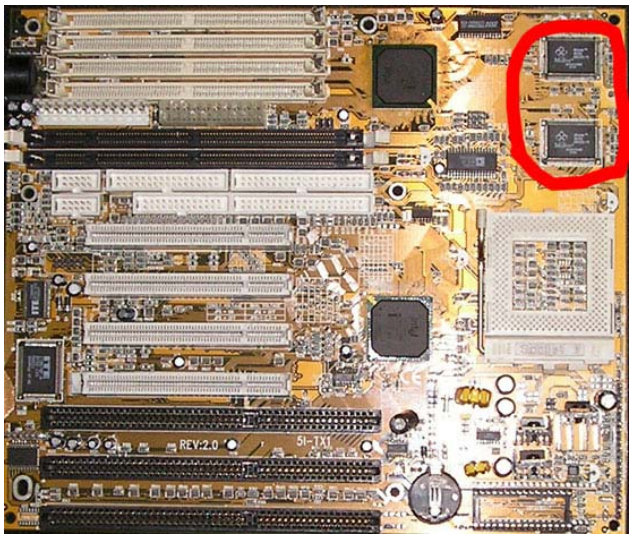
К важнейшим параметрам процессора относится так называемая кэш-память. Представим себе, как происходит обмен информацией между процессором и памятью. В большинстве случаев оперативная память не удовлетворяет потребностям современных процессоров в плане пропускной способности по данным, т.к работает на значительно более низких частотах. Современный процессор работает на частотах до 3 ГГц. Естественно, что при обмене с памятью процессор достаточно долгое время будет ждать прихода новых порций данных и таким образом простаивать. Для того чтобы этого избежать, между памятью и процессором устанавливают дополнительно небольшой объем очень быстрой памяти, работающей без задержек на частоте процессора. Такая память и называется **кэш–память**, и если вы помните из предыдущего урока, она имеет тип **SRAM**. В современном процессоре встроено некоторое количество такой памяти (от 32 Кб до 3 Мб) и эта память обеспечивает снижение простоев процессора при операциях с оперативной памятью. Та кэш-память, которая установлена непосредственно в кристалле процессора, называется кэш-памятью первого уровня (или L1, Level1).

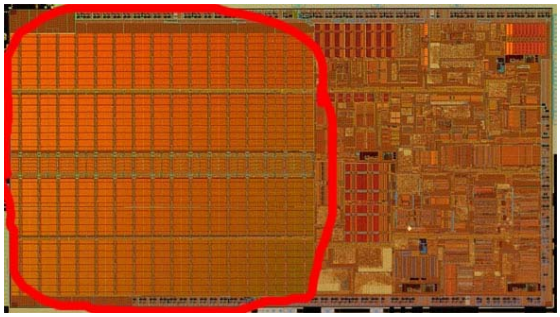
В переводе слово кэш (cache) означает «тайный склад», «тайник» («заначка»). Тайна этого склада заключается в его «прозрачности» — для программы он не представляет собой дополнительной адресуемой области памяти. Кэш является дополнительным быстродействующим хранилищем копий блоков информации из основной памяти, вероятность обращения к которым в ближайшее время велика. Кэш не может хранить копию всей основной памяти, поскольку его объем во много раз меньше основной памяти. Он хранит лишь ограниченное количество блоков данных и каталог (**cache directory**) — список их текущего соответствия областям основной памяти. Кроме того, кэшироваться может не вся память, доступная процессору.

При каждом обращении к памяти контроллер кэш-памяти по каталогу проверяет, есть ли действительная копия затребованных данных в кэше. Если она там есть, то это случай **кэш-попадания (cache hit)**, и данные берутся из кэш-памяти. Если действительной копии там нет, это случай **кэш-промаха (cache miss)**, и данные берутся из основной памяти. В соответствии с алгоритмом кэширования блок данных, считанный из основной памяти, при определенных условиях заместит один из блоков кэша. От интеллектуальности алгоритма замещения зависит процент попаданий и, следовательно, эффективность кэширования. Поиск блока в списке должен производиться достаточно быстро, чтобы «задумчивостью» в принятии решения не свести на нет выигрыш от применения быстродействующей памяти. Обращение к основной памяти может начинаться одновременно с поиском в каталоге, а в случае попадания — прерываться (архитектура **Look aside**). Это экономит время, но лишние обращения к основной памяти ведут к увеличению энергопотребления. Другой вариант: обращение к внешней памяти начинается только после фиксации промаха (архитектура **Look Through**), при этом теряется по крайней мере один такт процессора, зато экономится энергия.

В современных компьютерах кэш обычно строится по двухуровневой схеме. **Первичный кэш (L1 Cache)** встроен во все процессоры начиная с 486, это внутренний кэш. Объем этого кэша невелик (от 8 до 64 Кбайт). Для повышения производительности для данных и команд часто используется раздельный кэш (так называемая Гарвардская архитектура — противоположность Принстонской, использующей общую память для команд и данных).

Вторичный кэш (L2 Cache) для процессоров 486 и Pentium является внешним, т.е. устанавливается на системной плате, его чипы выделены на рисунке слева, а в Pentium pro и всех последующих процессорах располагается в одной упаковке с ядром и подключается специальной внутренней шиной к процессору, или вообще является частью кристалла процессора, на рисунке справа – фотография кристалла процессора, и выделен кэш 2-го уровня.





Разница в расположении кэша очень существенная – в случае, если он находится на материнской плате, он работает на частоте системной (внешней) шины процессора, и доступ к нему осуществляется опять же через эту шину. А в случае с подключением кэша отдельной шиной, частота его работы и производительность этой шины не будет зависеть от системной шины, соответственно и производительность будет намного выше.

Кэш-контроллер должен обеспечивать **когерентность (coherency)** — согласованность данных кэш-памяти обоих уровней с данными в основной памяти, при том условии, что обращение к этим данным может производиться не только процессором, но и другими активными (busmaster) адаптерами, подключенными к шинам (PCI, VLB, ISA и т. д.). Следует также учесть, что процессоров может быть несколько, и у каждого может быть свой внутренний кэш. Контроллер кэша оперирует **строками (cache line)** фиксированной длины. Строка может хранить копию блока основной памяти, размер которого, естественно, совпадает с длиной строки. С каждой строкой кэша связана информация об адресе скопированного в нее блока основной памяти и об ее состоянии. Строка может быть действительной (valid) — это означает, что в текущий момент времени она достоверно отражает соответствующий блок основной памяти, или недействительной. Информация о том какой именно блок занимает данную строку (то есть старшая часть адреса или номер страницы), и о ее состоянии называется **тегом (tag)** и хранится в связанной с данной строкой ячейке специальной **памяти тегов (tag RAM)** . В операциях обмена с основной памятью обычно строка участвует целиком (несекторированный кэш), для процессоров 486 и старше длина строки совпадает с объемом данных, передаваемых за один пакетный цикл (для 486 это 4x4=16 байт, для Pentium 4x8=32 байт). Возможен и вариант секторированного (sectored) кэша, при котором одна строка содержит несколько смежных ячеек — секторов, размер которых соответствует минимальной порции обмена данных кэша с основной памятью. При этом в записи каталога, соответствующей каждой строке, должны раниться биты действительности для каждого сектора данной строки. Секторирование позволяет экономить память, необходимую для ранения каталога при увеличении объема кэша, поскольку большее количество бит каталога отводится под тег и выгоднее использовать дополнительные биты действительности, чем увеличивать глубину индекса (количество элементов) каталога.

Строки кэша под отображение блока памяти выделяются при промахах операций чтения, в P6 строки заполняются и при записи. Запись блока, не имеющего копии в кэше, производится в основную память (для повышения быстродействия запись может производиться через буфер отложенной записи). Поведение кэш-контроллера при операции

записи в память, когда копия затребованной области находится в некоторой строке кэша, определяется его алгоритмом, или политикой записи (**Write Policy**) . Существуют две основные политики записи данных из кэша в основную память: сквозная запись **WT (Write Through)** и обратная запись **WB (Write Back)**.

Политика WT предусматривает выполнение каждой операции записи (даже однобайтной), попадающей в кэшированный блок, одновременно и в строку кэша, и в основную память. При этом процессору при каждой операции записи придется выполнять относительно длительную запись в основную память. Алгоритм достаточно прост в реализации и легко обеспечивает целостность данных за счет постоянного совпадения копий данных в кэше и основной памяти. Для него не нужно хранить признаки присутствия и модифицированности — вполне достаточно только информации тега (при этом считается, что любая строка всегда отражает какой-либо блок, а какой именно — указывает тег). Но эта простота оборачивается низкой эффективностью записи. Существуют варианты этого алгоритма с применением отложенной буферированной записи, при которой данные в основную память переписываются через FIFO-буфер во время свободных тактов шины.

Политика WB позволяет уменьшить количество операций записи на шине основной памяти. Если блок памяти, в который должна производиться запись, отображен в кэше, то физическая запись сначала будет произведена в эту действительную строку кэша, и она будет отмечена как грязная (dirty), или модифицированная, то есть требующая выгрузки в основную память. Только после этой выгрузки (записи в основную память) строка станет чистой (clean), и ее можно будет использовать для кэширования других блоков без потери целостности данных. В основную память данные переписываются только целой строкой. Эта выгрузка контроллером может откладываться до наступления крайней необходимости (обращение к кэшированной памяти другим абонентом, замещение в кэше новыми данными) или выполняться в свободное время после модификации всей строки.

9. Технология изготовления

Под технологией изготовления понимают размер элементов в кристалле процессора. Когда о процессоре говорят, что он изготовлен с помощью определенного технологического процесса, например 0,25 мкм, то имеют ввиду, что размер одного отдельно взятого транзистора в кристалле процессора равен 0,25 мкм. В чем преимущество использования более мелкого технологического процесса? Таких преимуществ целый ряд.

Во-первых, чем мельче технология, тем меньшую площадь занимает ядро процессора и тем больше самих кристаллов может изготовить производитель из одной кремниевой пластины. Т.е., процессор, изготовленный по более мелкой технологии, имеет низшую себестоимость, чем процессор, изготовленный с применением более крупного техпроцесса и, таким образом, его розничная цена для пользователя будет ниже. Но этим не ограничиваются достоинства мелких техпроцессов. Далее, процессор, сделанный по более мелкой технологии, может работать на более высоких частотах (если вы изучали физику, то понимаете почему, если нет - придется верить на слово :)). Стало быть, рост частоты процессора возможен только в рамках существующего техпроцесса, затем нужно переходить на новый, более мелкий. И это еще не все. Чем мельче элементы процессора, тем меньшего напряжения питания требует кристалл.

Ну и что, скажете Вы? Какое пользователю до этого дело? Чем меньше напряжение питания процессора, тем меньше мощности он потребляет и соответственно меньше тепла выделяет. А потребление мощности современного процессора достаточно высоко, особенно на больших частотах, и процессор, потребляя, нередко как половина электрической лампочки, так же неслабо как лампочка и греется. Соответственно, чем меньше питание, тем меньше процессор потребляет электрической энергии и тем меньше греется. А это в свою очередь еще один фактор, позволяющий увеличивать частоту процессора. Ведь чем меньше процессор греется, тем на более высокой частоте его можно использовать, и он не будет перегреваться. В общем, важность вопроса о техпроцессе, с помощью которого изготовлен процессор, трудно переоценить.

Технология изготовления современных процессоров - 0,09 мкм или 90 нм, и многие производители заявляют о скором переходе на новый, 65 нм техпроцесс.

10. Режимы работы процессора

Все 32-разрядные и более поздние процессоры Intel, начиная с 386, могут выполнять программы в нескольких режимах. Режимы процессора предназначены для выполнения программ в различных средах; в разных режимах работы возможности чипа не одинаковы, потому, что команды выполняются по-разному. В зависимости от режима процессора изменяется схема управления памятью системы и задачами.

Процессоры могут работать в трех режимах: реальном, защищенном и виртуальном реальном режиме (реальном внутри защищенного).

Реальный режим

В первоначальном IBM PC использовался процессор 8088, который мог выполнять 16-разрядные команды, используя 16-разрядные внутренние регистры и адресовать только 1 Мб памяти, используя 20-и разрядную шину адреса. Все программное обеспечение PC первоначально было предназначено для этого процессора, оно было разработано на основе 16-разрядной системы команд и модели памяти, объемом 1 Мб. Например DOS, все программное обеспечение DOS написано в расчете на 16-разрядные команды.

Более поздние процессоры, например 286, могли также выполнять те же самые 16-разрядные команды, что и первоначальный 8088, но намного быстрее. Другими словами процессор 286 был полностью совместим с первоначальным 8088. 16-разрядный режим, в котором выполнялись команды процессоров 8088 и 80286 был назван реальным режимом. Все программы, выполняющиеся в реальном режиме, должны использовать только 16-разрядные команды и 20-разрядный адрес. Для программного обеспечения такого типа используется однозадачный режим, т.е. одновременно должна выполняться только одна программа. Нет никакой встроенной защиты для предотвращения перезаписи ячеек памяти, занятых одной программой или даже самой операционной системой, другими программами: это означает, что при выполнении нескольких программ вполне могут быть испорчены данные или код одной из программ, что может привести к остановке системы.

Защищенный режим

Первым 32-разрядным процессором, предназначенным для PC, был 386-ой. Этот чип мог выполнять абсолютно новую 32-разрядную систему команд. Для того чтобы полностью использовать преимущество этой новой системы команд, были необходимы 32-разрядная операционная система и 32-разрядные приложения. Этот новый режим называли защищенным, так как выполняющиеся в нем программы защищены от перезаписи используемых ими областей памяти другими программами.

Такая защита делает систему более надежной, так как уже ни одна программа с ошибками не сможет повредить другие программы или операционную систему. Зная, что разработка новых операционных систем и приложений, использующих преимущество 32-разрядного защищенного режима, займет некоторое время, Intel предусмотрела в процессоре 386 обратно совместимый реальный режим. Благодаря этому процессор 386 мог выполнять обычные 16-разрядные приложения и операционные системы. Причем они выполнялись намного быстрее, чем на любом процессоре предыдущего поколения. Для большинства пользователей этого было достаточно: они не востребовали 32-разрядные системы и приложения и довольствовались тем, что уже имеющиеся у них 16-разрядные программы работали быстрее. К сожалению, из-за этого 386 процессор так никогда и не использовался в защищенном режиме, и, стало быть, все преимущества такого режима терялись. Когда современный высокопроизводительный процессор работает в реальном режиме, то он напоминает чудовищно ускоренный 8088! Т.е процессор может хоть и с огромной (по сравнению с оригинальным 8088) скоростью, но все же выполнять только 16-разрядные приложения и адресоваться только к памяти размером 1 Мб, с которой мог

работать 8088. Поэтому были необходимы новые операционные системы и новые приложения, которые могли бы на современных процессорах выполняться в защищенном 32-разрядном режиме. Однако пользователи сопротивлялись всем попыткам перехода к 32-разрядной среде. Для них это означало, что нужно, по крайней мере частично, отказываться от старого программного обеспечения, а пользователю это не подходило. Только в августе 1995 года (спустя 10 лет после выхода первого 32-разрядного процессора) наконец появилась первая пользовательская 32-разрядная операционная система Windows 95, да и то, пользователи приняли ее во многом потому, что она частично 16-разрядная и поэтому без труда исполняет как новые 32-разрядные программы, так и старые, 16-разрядные. Именно для такой обратной совместимости Windows 95 использовала третий режим процессора:

Виртуальный реальный режим

Виртуальный реальный, по существу, является режимом выполнения 16-разрядной среды (реальный режим), который реализован внутри 32-разрядного защищенного режима. Выполняя команды в окне DOS в Windows 95/98, вы создаете виртуальный сеанс реального режима. Поскольку защищенный режим является подлинно многозадачным, фактически можно выполнять несколько сеансов реального режима, причем в каждом сеансе собственное программное обеспечение выполняется на собственном виртуальном компьютере. И все эти приложения могут выполняться одновременно, даже во время выполнения других 32-разрядных программ. Следует обратить внимание на то, что любая программа, выполняющаяся в виртуальном реальном режиме, может обращаться к памяти, объемом до 1 Мб, причем для каждой такой программы это будет как бы первый и единственный мегабайт памяти в системе. Виртуальное реальное окно полностью имитирует среду процессора 8088и если не учитывать быстродействие, программное обеспечение в виртуальном реальном режиме выполняется так, как выполнялось бы на самых первых PC в реальном режиме. При запуске каждого 16-разрядного приложения Windows 95/98 создает так называемую виртуальную машину DOS, выдает ей 1 Мб памяти и на этой машине 16-разрядное приложение выполняется. Следует обратить внимание на то, что все процессоры при включении начинают работать в реальном режиме, и только при старте 32-разрядной операционной системы происходит переключение в 32-разрядный режим.

Так же следует отметить, что не любое 16-разрядное приложение будет корректно работать в виртуальном реальном режиме. Например, диагностические программы для обслуживание аппаратного обеспечения делают вещи, не предусмотренные в виртуальном реальном режиме (в первую очередь пытаются напрямую работать с аппаратурой). Такие программы нельзя запускать из Windows 95/98, для запуска таких программ необходимо стартовать компьютер с операционной системой DOS и выполнять эти приложения в настоящем, а не виртуальном реальном режиме.

11. Многопроцессорные системы

Довольно давно возникла идея использования нескольких процессоров, вместо одного. Т.е. руководствуясь принципом «одна голова хорошо, а две лучше», производители процессоров решили найти способ увеличения производительности компьютера, не изменяя при этом параметров самих процессоров. Таким образом, стали появляться системы с 2-мя, 4-мя и более процессорами. По идее производительность таких систем должна быть больше в 2, 4 и т.д. раз, но на практике это совсем не так.

При создании многопроцессорных систем надо учитывать множество условий:

- Материнская плата должна уметь работать с несколькими процессорами, т.е. на ней должен быть соответствующий чипсет, и необходимое число разъёмов для процессоров.
- Сам процессор должен поддерживать работу в многопроцессорных системах, такие современные процессоры часто имеют в маркировке индекс **MP**.

- Операционные системы и программное обеспечение также должны уметь работать с несколькими процессорами, в следующем курсе, мы будем ещё упоминать об этом, когда будем рассматривать возможности операционных систем.

Многопроцессорные системы дают хороший прирост производительности, только тогда, когда на них выполняется специализированное программное обеспечение. Компьютер с несколькими процессорами работает под управлением операционной системы, которая может распределять разные задачи по разным процессорам компьютера. А программы, которые выполняются на таком компьютере, должны состоять из нескольких потоков, которые можно выполнять независимо друг от друга. За счет этих возможностей будет увеличиваться производительность. Соответственно, если операционная система и программное обеспечение не отвечает этим требованиям, то и выигрыша от применения многопроцессорное системы не будет никакого, и и её производительность будет такой же, что и у однопроцессорной системы.

Существуют два режима работы многопроцессорных систем – симметричный и несимметричный. Рассмотрим их подробнее:

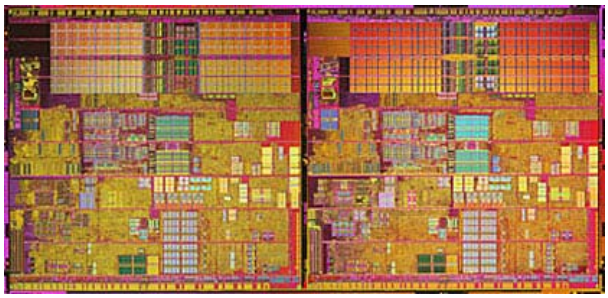
- **AMP, Asymmetric Multiprocessing**. Асимметричная многопроцессорность. В этом режиме операционная система выполняется на одних процессорах, а приложения выполняются другими процессорами. Такой режим в определённых условиях является неэффективным, т.к. нагрузка распределяется неравномерно, поэтому сейчас чаще всего применяется второй режим.
- **SMP, Symmetric Multiprocessing**. Симметричная многопроцессорность. В этом режиме и операционная система, и приложения выполняются любым процессором, в зависимости от его загрузки, что обеспечивает большую гибкость при распределении нагрузки, и соответственно большую производительность. Для работы в таком режиме был разработан специальный протокол, который называется **APIC**.

У многопроцессорных систем есть кроме множества достоинств, и явный недостаток – стоимость. Материнские платы для многопроцессорных систем стоят намного дороже однопроцессорных, да и покупка нескольких процессоров тоже не является дешевым удовольствием. Поэтому производители стали разрабатывать более дешевые варианты многопроцессорности.

Первый из этих вариантов, который появился в процессорах Pentium 4 на ядре Northwood, это технология **Hyper Threading (HT)**, которая представляет из себя "логическую многопроцессорность". В чём её особенность?

Как вы уже знаете, процессор состоит из нескольких блоков, выполняющих различные задачи, например 2 конвейера ALU, из которых один основной а второй вспомогательный, т.е. загружен не всегда, 1 FPU и блок загружающий и выгружающий команды и данные из памяти. Представьте ситуацию, что во время выполнения программы возникает ситуация, когда на протяжении работы нескольких стадий конвейера загружены 60% только ALU, FPU не загружен вообще и блок загрузки–выгрузки занят на 30%. В среднем выйдет процент использования ресурсов процессора примерно 30%. Но это только конкретный случай, возможны и масса других ситуаций, когда процессор будет использовать большее, или меньшее число своих блоков, и во многом это зависит от программного обеспечения. Сама Intel заявляет, что блоки процессора Pentium 4 загружены в среднем не более чем на 35%. Поэтому они и предложили технологию, которая при небольших изменениях ядра (увеличение примерно на 10%), позволяет использовать простаивающие блоки процессора в качестве дополнительного, 2-го процессора. Для этого был добавлен ещё один блок регистров, и программы работают с таким процессором не как с одним, и как с двумя. Конечно, такой подход не даст полноценной альтернативы многопроцессорным системам, но в зависимости от конкретного применения HT, прирост производительности может достигать 30%, хотя для домашнего пользователя он может

быть равным и 0%, т.к. задачи выполняемые ими, как правило не вообще требуют многопроцессорности.



Также, с уменьшением технологического процесса производства процессоров, появилась возможность создавать на одном кристалле, который помещается в стандартный корпус процессора 2 и более процессоров. Такие процессоры называются **многоядерными**. А такую архитектуру называют **CMP, Cellular MultiProcessing**, сотовая мультипроцессорность. Особенностью, и отличием такой архитектуры от SMP, является использование несколькими ядрами общего пространства памяти.

Производители процессоров в настоящее время выпускают 2-ядерные процессоры, и в их планах на ближайшее будущее выпуск процессоров с 4-мя ядрами и более. На фото – изображение двухъядерного процессора Pentium D 840, и можно чётко различить два одинаковых ядра.

12. NX-bit

Многие современные процессоры, в основном x86-64, вдобавок к описанным технологиям, содержат ещё одну интересную технологию – **NX-bit**, защищающую программы от ситуаций, когда, используя переполнение буфера, одни программы могут вставлять свой код в область данных других программ, и выполнять его. Эта особенность используется многими современными вирусами для заражения компьютеров. А технология **NX-bit, или Non-eXecute bit** позволяет защититься от подобного способа заражения на аппаратном уровне. Особенностью этой технологии также является необходимость поддержки со стороны операционной системы, и судя по отзывам, она далеко не всегда помогает, а в некоторых случаях даже мешает нормальной работе компьютера, ведь ни сама технология, ни программное обеспечение, использующее её далеко не идеальны, и стоит надеяться на то, что в будущем все подобные проблемы будут решены.

В следующем уроке мы с вами рассмотрим основные модели процессоров которые применялись недавно, применяются сейчас и будут применяться в ближайшем будущем.