**Вопрос №1:**

Какие блоки составляют конвейер МП 80286?

* BU (Bus Unit) – шинный блок (считывание из памяти и портов ввода/вывода);
* IU (Instruction Unit) - командный блок (дешифрация команд);
* EU (Executive Unit) - исполнительный блок (выполнение команд);
* AU (Address Unit) - адресный блок (вычисляет все адреса, формирует физический адрес).

**Вопрос №2:**

Какой блок и почему был добавлен в конвейер МП Intel-486?

PF (Prefetch) - предвыборка команд. Этот блок позволил проводить подготовку к выполнению следующей команды во время выполнения текущей.

**Вопрос №3:**

Что понимают под суперскалярной архитектурой?

Наличие более одного конвейера для обработки команд.

**Вопрос №4:**

Какие способы обработки данных объединяет термин «динамическое исполнение программы»?

* Глубокое предсказание ветвлений (с вероятностью >90% можно предсказать 1015 ближайших переходов)
* Анализ потока данных (на 20-30 шагов вперед просмотреть программу и определить зависимость команд по данным или ресурсам).
* Опережающее исполнение команд (МП P6 может выполнять команды в порядке, отличном от их следования в программе).

**Вопрос №5:**

В чем состоит внутренняя RISC-архитектура ЦП Pentium Pro?

**Подсистема упорядоченной предварительной обработки**

К устройствам этой подсистемы относятся:

* Модуль и буфер предсказания переходов (*Branch Target Buffer, BTB*) — предсказывают переходы и хранят таблицу истории переходов. Для предсказания используются как динамический, так и статический методы. Последний используется в том случае, если динамическое предсказание невозможно (в таблице переходов отсутствует необходимая информация).
* Декодер инструкций (*Instruction Decoder*) — преобразует CISC-инструкции x86 в последовательность RISC-микроопераций, исполняемых процессором. Включает два декодера простых инструкций (*Simple*), обрабатывающих команды, которые могут быть выполнены одной микрооперацией, и декодер сложных инструкций (*Complex*), обрабатывающего команды, для которых нужно несколько (до четырёх) микроопераций.
* Планировщик последовательностей микроопераций (*Microcode sequencer*) — хранит последовательности микроопераций, используемые при декодировании сложных инструкций x86, требующих более четырёх микроопераций.
* Блок вычисления адреса следующей инструкции (*Next IP Unit*) — вычисляет адрес инструкции (англ. *instruction pointer, IP*), которая должна быть обработана следующей, на основании информации о прерываниях и таблицы переходов.
* Блок выборки инструкций (*Instruction Fetch Unit, IFU*) — осуществляет выборку инструкций из памяти по адресам, подготовленным блоком вычисления адреса следующей инструкции.

**Ядро исполнения с изменением последовательности**

Исполнение с изменением последовательности, при котором меняется очерёдность исполнения инструкций, так, чтобы это не приводило к изменению результата, позволяет ускорить работу за счёт более оптимального распределения запросов к вспомогательным блокам и минимизации их простоев. К устройствам организации исполнения с изменением последовательности относятся:

* Таблица назначения регистров (*Register Alias Table*) — задаёт соответствие между регистрами архитектуры x86/IA32 (Intel Architecture 32-bit) и внутренними регистрами, используемыми при исполнении микроопераций.
* Буфер переупорядочивания микроопераций (*Reorder Buffer*) — обеспечивает выполнение микроопераций в оптимальной с точки зрения производительности последовательности.
* Станция-резервуар (*Reservation Station*) — содержит инструкции, отправляемые на исполнительные устройства.

К исполнительным устройствам ядра относятся:

* Арифметическо-логические устройства, АЛУ (*Arithmetic Logic Unit, ALU*) — выполняют целочисленные операции.
* Блок арифметики с плавающей запятой (*Floating Point Unit, FPU*) — выполняет операции над числами с плавающей точкой. Процессоры Pentium III и выше имеют также блок, осуществляющий исполнение инструкций SSE (*SIMD FPU*).
* Блок генерации адресов (*Address Generation Unit, AGU*) — вычисляет адреса данных, используемых инструкциями, и формирует запросы к кешу для загрузки/выгрузки этих данных.

**Подсистема упорядоченного завершения**

* Регистровый файл (*Register File*) — хранит результаты операций (состояние регистров IA32 для исполняемых инструкций).
* Буфер переупорядочивания памяти (*Memory Reorder Buffer*) — управляет порядком записи данных в память для предотвращения записи неверных данных из-за изменения порядка выполнения инструкций.
* Блок завершения (*Retirement Unit*) — выдаёт результаты исполнения инструкций в той последовательности, в которой они поступили на исполнение.

**Подсистема памяти**

Подсистема памяти осуществляет взаимодействие с оперативной памятью. К этой подсистеме относятся:

* Кэш первого уровня для данных (*Level 1 Data Cache, L1D*) — память с малым временем доступа объёмом 8 (для Pentium Pro) или 16 (для более новых процессоров) [килобайт](https://ru.wikipedia.org/wiki/%D0%9A%D0%B8%D0%BB%D0%BE%D0%B1%D0%B0%D0%B9%D1%82), предназначенная для хранения данных.
* Кэш первого уровня для инструкций (*Level 1 Instruction Cache, L1I*) — память с малым временем доступа объёмом 8 (Pentium Pro) или 16 килобайт, предназначенная для хранения инструкций.
* Кэш второго уровня (*Level 2 Cache, L2*). Память с малым временем доступа объёмом 128, 256, 512, 1024 или 2048 килобайт. Ширина шины L2 составляет 64 или 256 (для процессоров на ядре [Coppermine](https://ru.wikipedia.org/wiki/Pentium_III" \l "Coppermine" \o "Pentium III) и выше) бит. Процессоры Celeron на ядре Covington кэша второго уровня не имеют.
* Блок шинного интерфейса (*Bus Interface Unit*) — управляет системной шиной.

**Вопрос №6:**

В работе какого процессора наблюдается отклонение от принципов фон Неймана? В чем это проявляется?

**Вопрос №7:**

В чем состоит преимущество использования двойной независимой шины?

Двойная независимая шина увеличивает пропускную способность подсистемы памяти. Одна из шин, Back side bus, соединяет процессор с кэш-памятью второго уровня. Вторая, Front side bus, соединяет процессор с северным мостом набора микросхем.

**Вопрос №9:**

Какие особенности имеет Net Burst-архитектура?

**Гиперконвейеризация**

Процессоры Pentium 4 на ядрах Willamette и Northwood имеют конвейер глубиной 20 стадий, а процессоры на ядрах Prescott и Cedar Mill — 31 стадию. При этом стадии декодирования инструкций не учитываются: в связи с применением кэша последовательностей микроопераций декодер вынесен за пределы конвейера. Это позволяет процессорам Pentium 4 достигать более высоких тактовых частот по сравнению с процессорами, имеющими более короткий конвейер при одинаковой технологии производства. Так, например, максимальная тактовая частота процессоров [Pentium III](https://ru.wikipedia.org/wiki/Pentium_III" \o "Pentium III) на ядре Coppermine (180 [нм](https://ru.wikipedia.org/wiki/%D0%9D%D0%BC" \o "Нм). технология) составляет 1133 МГц, а процессоры Pentium 4 на ядре Willamette способны работать на частоте, превышающей 2000 МГц.

Для минимизации влияния неверно предсказанных переходов в процессорах архитектуры NetBurst используется увеличенный по сравнению с предшественниками буфер предсказания ветвлений и новый алгоритм предсказания ветвлений, что позволило достичь высокой точности предсказания (около 94 %) в процессорах на ядре Willamette. В последующих ядрах механизм предсказания ветвлений подвергался модернизациям, повышавшим точность предсказания.

**Кэш последовательностей микроопераций**

Процессоры архитектуры NetBurst, как и большинство современных [x86](https://ru.wikipedia.org/wiki/X86)-совместимых процессоров, являются CISC-процессорами с RISC-ядром: перед исполнением сложные инструкции x86 преобразуются в более простой набор внутренних инструкций (микроопераций), что позволяет повысить скорость обработки команд. Однако, вследствие того, что инструкции x86 имеют переменную длину и не имеют фиксированного формата, их декодирование связано с существенными временными затратами.

В связи с этим, при разработке архитектуры NetBurst было принято решение отказаться от традиционной кэш-памяти инструкций первого уровня, хранящей команды x86, в пользу кэша последовательностей микроопераций, хранящего последовательности микроопераций в соответствии с предполагаемым порядком их исполнения. Ёмкость trace cache составляла около 12 тыс. микроопераций. Такая организация кэш-памяти позволила также снизить временные затраты на выполнение условных переходов и на выборку инструкций.

**АЛУ и механизм ускоренного исполнения целочисленных операций**

Так как основной целью разработки архитектуры NetBurst было повышение производительности за счёт достижения высоких тактовых частот, возникла необходимость увеличения темпа выполнения основных целочисленных операций. Для достижения этой цели АЛУ процессоров архитектуры NetBurst разделено на несколько блоков: «медленное АЛУ» ,способное выполнять большое количество целочисленных операций, и два «быстрых АЛУ», выполняющих только простейшие целочисленные операции (например, сложение). Выполнение операций на «быстрых АЛУ» происходит последовательно в три этапа: сначала вычисляются младшие разряды результата, затем старшие, после чего могут быть получены флаги.

«Быстрые АЛУ», обслуживающие их планировщики, а также регистровый синхронизируются по половине такта процессора, таким образом, эффективная частота их работы вдвое превышает частоту ядра. Эти блоки образуют механизм ускоренного выполнения целочисленных операций.

В процессорах на ядрах Willamette и Norhtwood «быстрые АЛУ» способны выполнять лишь те операции, которые обрабатывают операнды в направлении от младших разрядов к старшим. При этом результат вычисления младших разрядов может быть получен через половину такта. Таким образом, эффективная задержка составляет половину такта. В процессорах на ядрах Willamette и Norhtwood отсутствуют блоки целочисленного умножения и сдвига, а данные операции выполняются другими блоками (в частности, блоком инструкций MMX).

В процессорах на ядре Prescott (а также более новых ядрах) присутствует блок целочисленного умножения, а «быстрые АЛУ» способны выполнять операции сдвига. Эффективная задержка операций, исполняемых «быстрыми АЛУ», возросла по сравнению с процессорами на ядре Norhtwood и составляет один такт.

**Система повторного исполнения микроопераций**

Основной задачей планировщиков микроопераций является определение готовности микроопераций к исполнению и передача их на конвейер. Вследствие большого числа стадий конвейера планировщики вынуждены отправлять микрооперации на исполнительные блоки до того, как завершится выполнение предыдущих микроопераций. Это обеспечивает оптимальную загрузку исполнительных блоков процессора и позволяет избежать потери производительности в том случае, если данные, необходимые для выполнения микрооперации, находятся в кэш-памяти первого уровня, регистровом файле или могут быть переданы, минуя регистровый файл.

При определении готовности новых микроопераций к передаче на исполнительные блоки планировщику необходимо определить время выполнения тех предыдущих микроопераций, результатом которых являются данные, необходимые для выполнения новых микроопераций. В том случае, если время выполнения заранее не определено, планировщик для его определения использует наименьшее время её выполнения (так, например, если для выполнения некоторой микрооперации необходимо загрузить данные из памяти, планировщик при передаче этой микрооперации на конвейер будет исходить из предположения, что необходимые данные находятся в кэш-памяти данных первого уровня и будут получены через количество тактов, равное сумме латентности кэш-памяти первого уровня и количества тактов, которое займёт передача микрооперации от планировщика до исполнительного блока).

Если оценка времени, необходимого для получения данных, оказалась верной, микрооперация выполняется успешно. В том случае, если данные не были получены вовремя, проверка корректности результата заканчивается неудачей. При этом микрооперация, результат выполнения которой оказался некорректен, ставится в специальную очередь, а затем вновь направляется планировщиком на исполнение.

Существуют такие неблагоприятные ситуации, в которых повторное исполнение микроопераций может привести к взаимоблокировкам. Выход из таких ситуаций осуществляется прекращением передачи новых микроопераций на исполнительные блоки и направлением переисполняемых микроопераций в специальный буфер для того, чтобы они могли освободить конвейер.

Несмотря на то, что повторное исполнение микроопераций приводит к значительным потерям производительности, применение данного механизма позволяет в случае ошибочного исполнения микроопераций избежать останова и сброса конвейера, который приводил бы к более серьёзным потерям.