Микропроцессоры

Процессором называется устройство, осуществляющее обработку данных и программное управление этим процессом.

Процессор занимает центральное место в структуре ЭВМ, так как он осуществляет управление взаимодействием всех устройств, входящих в состав ЭВМ .

Логические основы ЭВМ, элементы и узлы

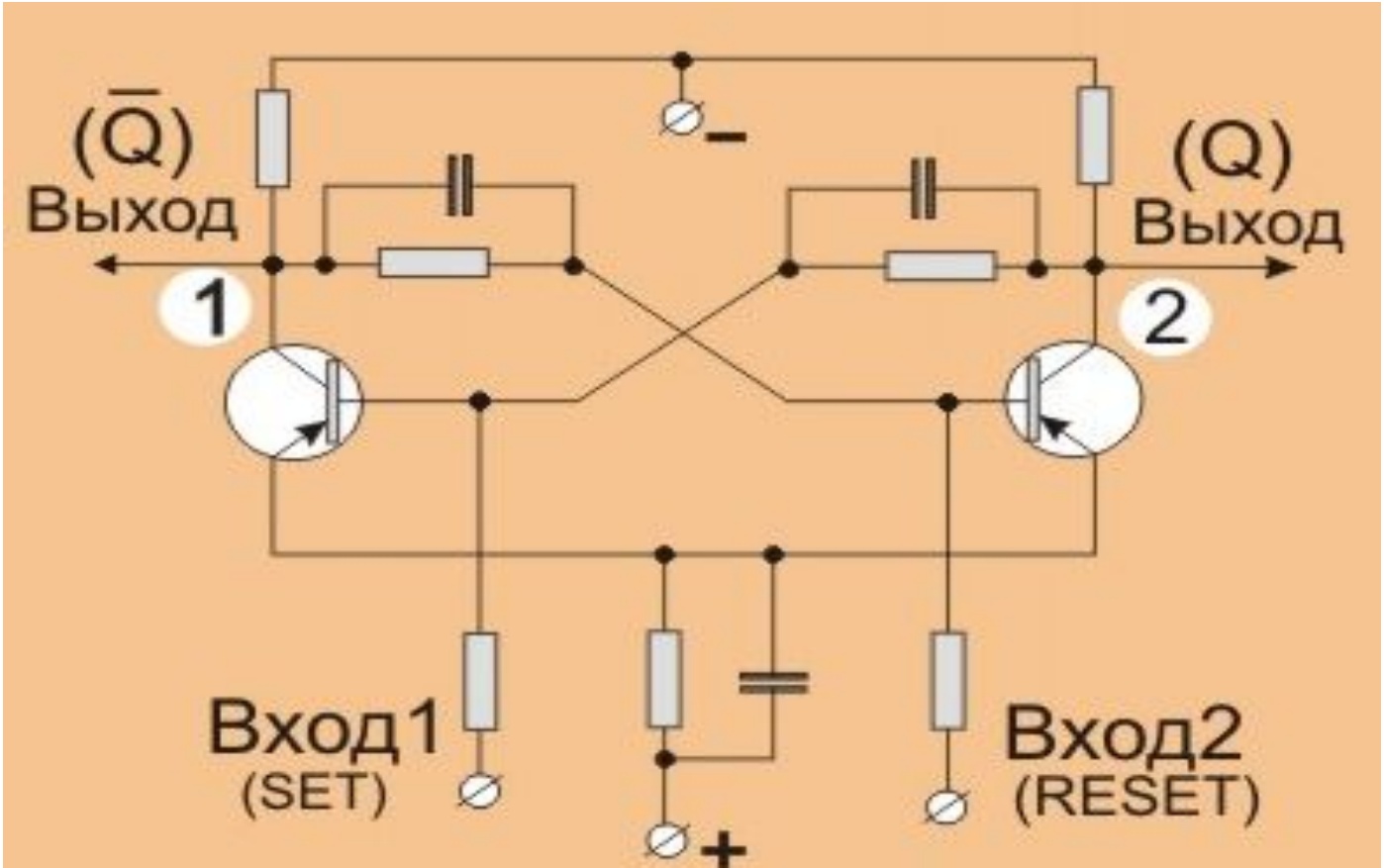
- **Основа: алгебра логики**

Алгебра логики – это раздел математической логики, значения всех элементов (функций и аргументов) которой определены в двухэлементном множестве: 0 и 1. Алгебра логики оперирует логическими высказываниями.

- **Основные технические элементы: триггеры, сумматоры, регистры, шифраторы, дешифраторы, мультиплексоры, демультиплексоры, счетчики.**

Триггеры

Простейшая электронная схема триггера состоит из двух усилительных каскадов. Выход каждого из каскадов подключен к входу другого через резисторы. Номиналы этих резисторов подобраны так, что каскад с полностью открытым транзистором, уверенно запирает транзистор другого каскада.



Пример работы процессора

В качестве примера, иллюстрирующего работу микро ЭВМ, рассмотрим процедуру, для реализации которой нужно выполнить следующую последовательность элементарных операций:

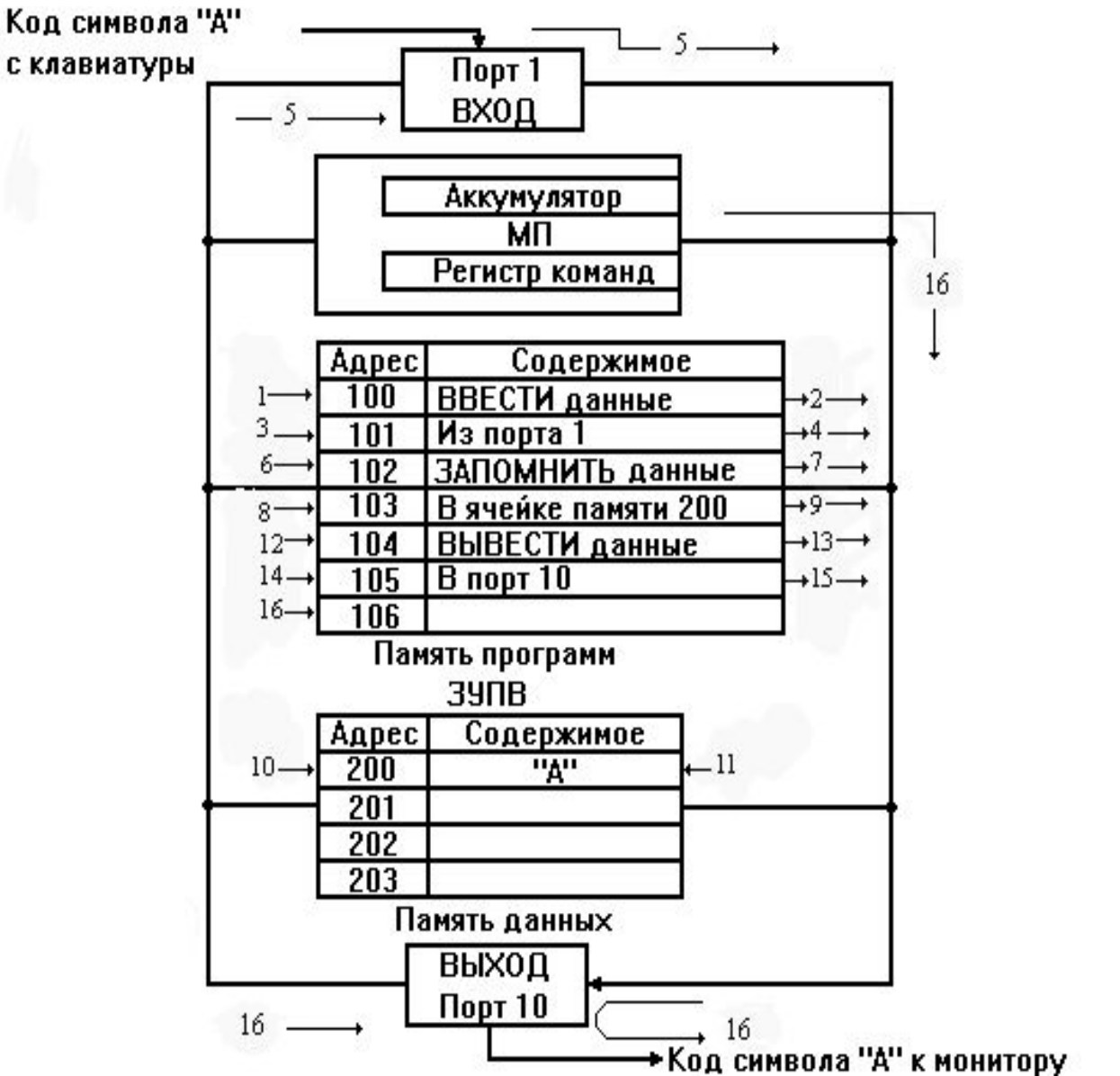
1. Нажать клавишу с буквой "А" на клавиатуре.
2. Поместить букву "А" в память микроЭВМ.
3. Вывести букву "А" на экран дисплея.

Это типичная процедура ввода-запоминания-вывода, рассмотрение которой дает возможность пояснить принципы использования некоторых устройств, входящих в микроЭВМ.

Схема

Команды уже загружены в первые шесть ячеек памяти. Хранимая программа содержит следующую цепочку команд:

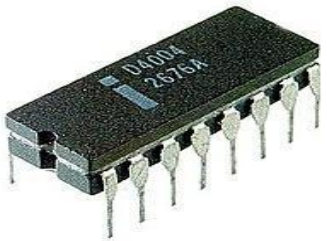
1. Ввести данные из порта ввода 1.
2. Запомнить данные в ячейке памяти 200.
3. Переслать данные в порт вывода 10.



Схема

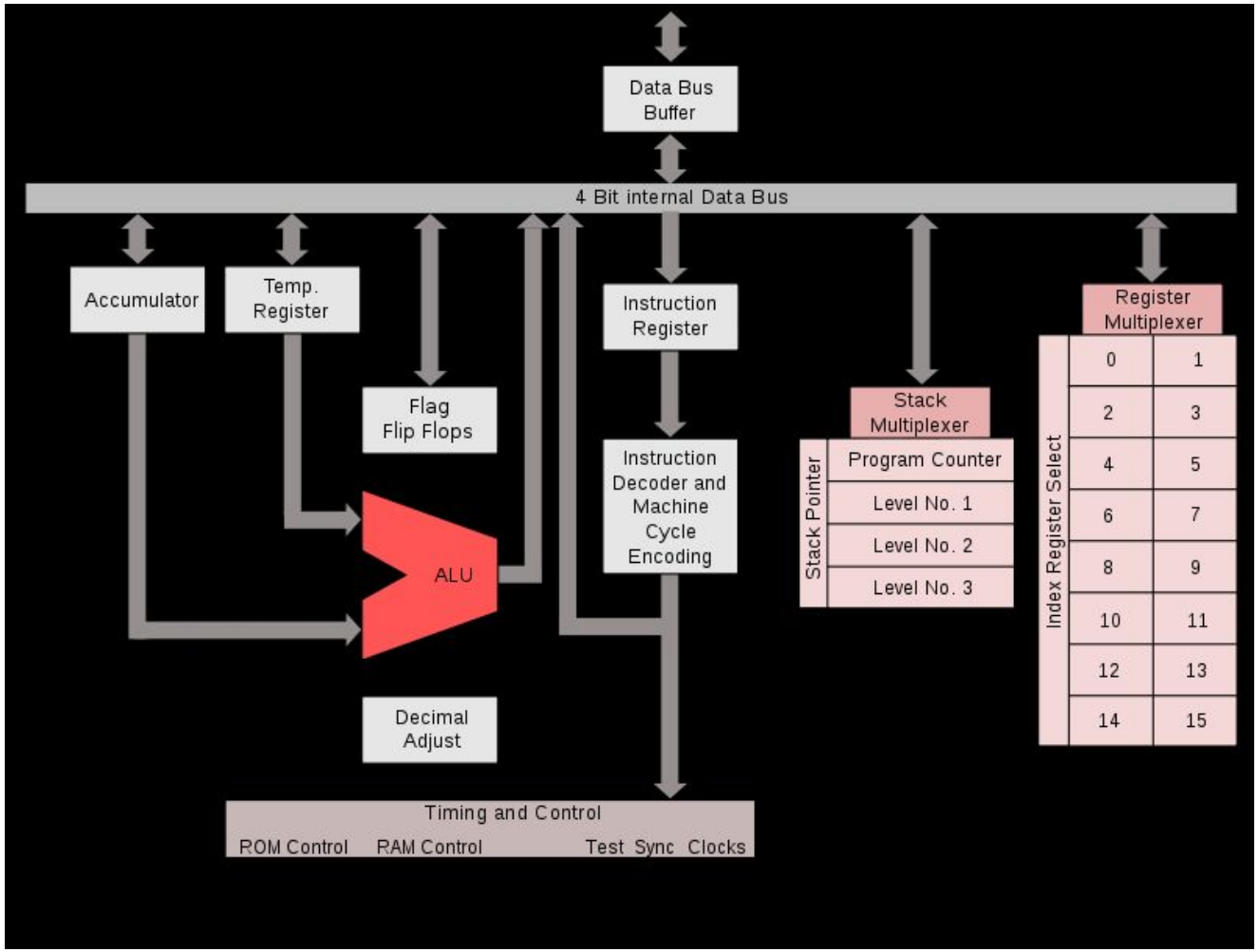
На рисунке в памяти программ записано шесть команд. Это связано с тем, что команда обычно разбивается на части. Первая часть команды 1 в приведенной выше программе - команда ввода данных. Во второй части команды 1 указывается, откуда нужно ввести данные (из порта 1). Первая часть команды, предписывающая конкретное действие, называется кодом операции (КОП), а вторая часть - операндом. Код операции и операнд размещаются в отдельных ячейках памяти программ. КОП хранится в ячейке 100, а код операнда - в ячейке 101 (порт 1); последний указывает откуда нужно взять информацию.

История



Первым общедоступным микропроцессором был 4-разрядный Intel 4004, созданный в 1971 году корпорацией Intel. Он содержал 2300 транзисторов, работал с тактовой частотой 92,6 кГц и стоил 200 $. Кристалл представлял собой 4 -х разрядный процессор с классической архитектурой ЭВМ гарвардского типа и изготавливался по передовой технологии с проектными нормами 10 мкм. Электрическая схема прибора насчитывала 2300 транзисторов. МП работал на тактовой частоте 750 кГц при длительности цикла команд 10,8 мкс. Чип i4004 имел адресный стек (счетчик команд и три регистра стека), блок РОНов (регистры сверхоперативной памяти), 4 - разрядное параллельное АЛУ, аккумулятор, регистр команд с дешифратором команд и схемой управления, а также схему связи с внешними устройствами.

4004. Блок-схема

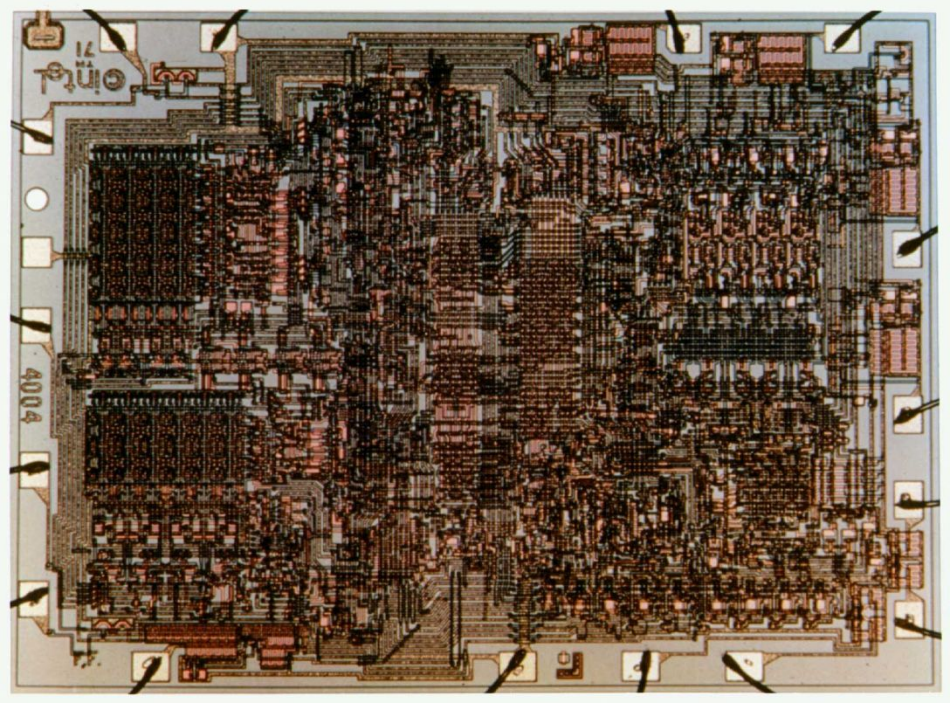


История

Все эти функциональные узлы объединялись между собой 4 - разрядной ШД. Память команд достигала 4 кб, а РОН ЦП насчитывал шестнадцать 4 -х разрядных регистров, которые можно было использовать и как восемь 8 -ми разрядных. Такая организация РОНов сохранена и в последующих МП фирмы Intel. Три регистра стека обеспечивали три уровня вложения подпрограмм. В систему его команд входило всего 46 инструкций. Кристалл располагал весьма ограниченными средствами ввода/вывода, а в системе команд отсутствовали операции логической обработки данных (И, ИЛИ, ИСКЛЮЧАЮЩЕЕ ИЛИ), в связи с чем их приходилось реализовывать с помощью специальных подпрограмм.

История

Модуль i4004 не имел возможности останова (команды HALT) и обработки прерываний. Цикл команды процессора состоял из 8 тактов задающего генератора. Была мультиплексированная ША (шина адреса)/ШД (шина да н-ных), адрес 12 - разрядный передавался по 4 разряда.



История. 8080

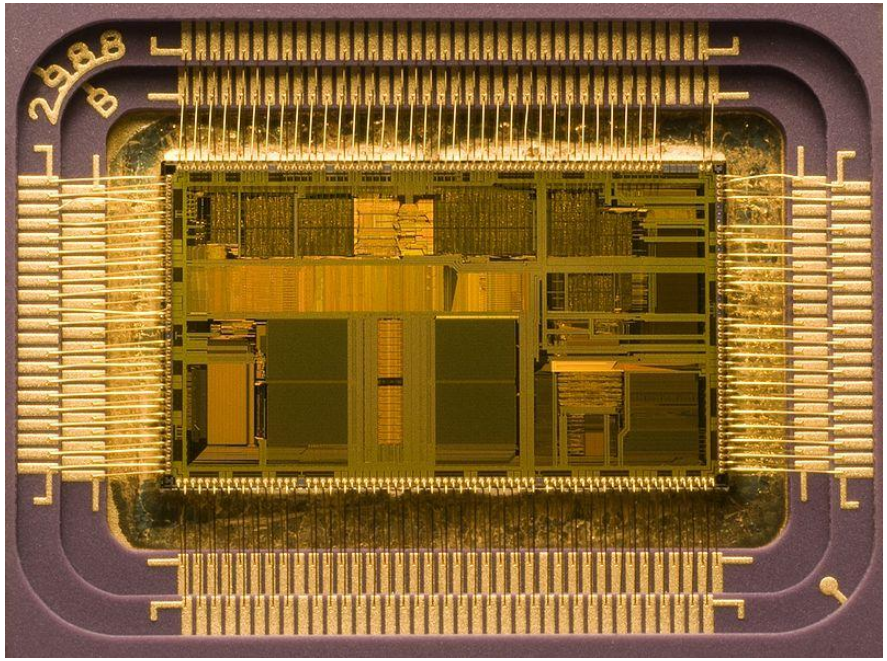
Далее его сменили 8-разрядный Intel 8080 и 16-разрядный 8086, заложившие основы архитектуры всех современных настольных процессоров. Система команд процессора Intel 8086 состояла из 98 команд (и более 3800 их вариаций): 19 команд передачи данных, 38 команд их обработки, 24 команды перехода и 17 команд управления процессором.

История

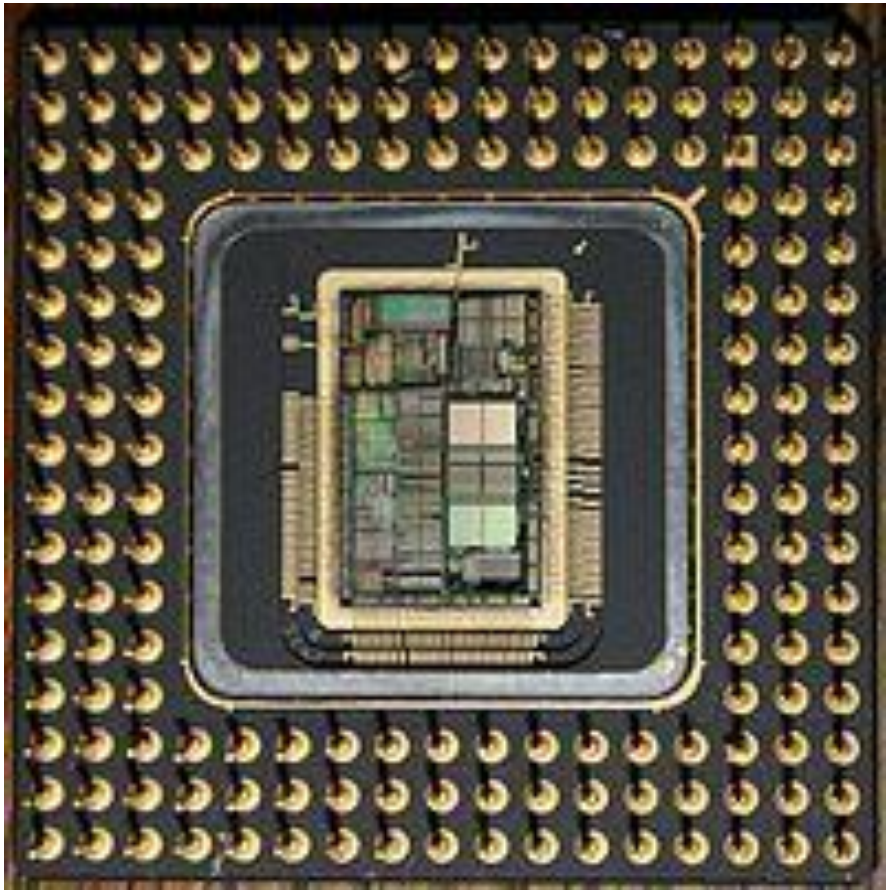
В процессоре 80286 появился защищённый режим с 24-битной адресацией, до 16 Мб памяти. Процессор Intel 80386 (1985) и привнёс улучшенный защищённый режим, 32-битную адресацию, позволившую использовать до 4 Гб оперативной памяти и поддержку механизма виртуальной памяти. Позже последовал процессор 80486. Новое поколение процессоров Intel началось серией Pentium в 1993 году. В модели P5 появилась возможность выполнять сразу две команды, изменился процесс кэширования, в 2 раза повысилась пропускная способность 64-разрядной шины. Дальнейшее совершенствование процессоров продолжалось в сторону увеличения тактовой част а встроенного кеша и разрядности.

Микропроцессор 80486

Кристалл микропроцессора Intel 80486DX2



Расположение кристалла в корпусе микропроцессора



Планарная технология

- **Планарная технология** — совокупность технологических операций, используемых при изготовлении планарных (плоских, поверхностных) полупроводниковых приборов и интегральных микросхем.

Планарная технология

- ПЛАНА́РНАЯ ТЕХНОЛО́ГИЯ (от лат. planus – плоский, ровный), совокупность способов изготовления полупроводниковых приборов и интегральных схем путём формирования их структур с одной стороны пластины (подложки), вырезанной из монокристалла. П. т. основана на создании в приповерхностном слое подложки областей с разл. типами проводимости или с разной концентрацией примесей, в совокупности образующих структуру ПП прибора или интегральной схемы. Наибольшее распространение в качестве полупроводникового материала для подложек в П. т. получил монокристаллич. кремний.

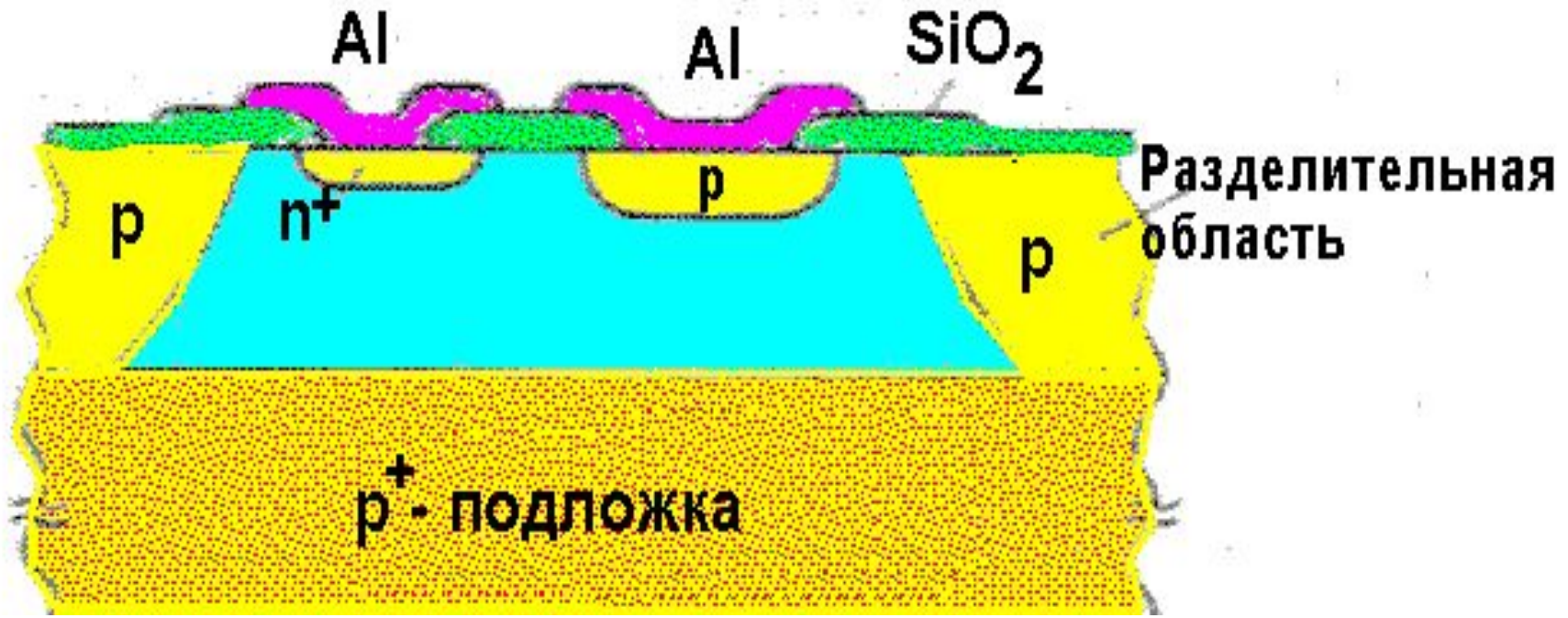
Планарная технология

- Области структур создаются локальным введением в подложку примесей (посредством диффузии или ионной имплантации), осуществляемым через маску (обычно из плёнки оксида кремния SiO_2), формируемую на рабочей стороне подложки при помощи фотолитографии. Последовательно проводя процессы окисления (образование плёнки SiO_2), фотолитографии (создание маски) и введения примесей, можно получить в приповерхностном слое подложки легированную область любой требуемой конфигурации, а также внутри области с одним типом проводимости (уровнем концентрации примесей) создать область с др. типом проводимости

Планарная технология

- Все области имеют выход на одну сторону подложки, что позволяет через окна в плёнке SiO_2 осуществить их коммутацию в соответствии с заданной схемой посредством плёночных металлич. (как правило, из Al) проводников, формируемых также с помощью фотолитографии. Плёнка SiO_2, помимо использования её в качестве маски, защищает выходящие на поверхность p – n переходы как в процессе их формирования, так и при эксплуатации ПП приборов и микросхем.

Планарная технология



Планарная технология

Основные технологические операции, используемые в планарной технологии, основаны на процессе литографии (фотолитографии). Применяются следующие способы:

оптическая фотолитография (стандартная), λ=310—450 нм;

ультрафиолетовая фотолитография на эксимерных лазерах, λ=248 нм, λ=193 нм

фотолитография в глубоком ультрафиолете, λ=100—10 нм;

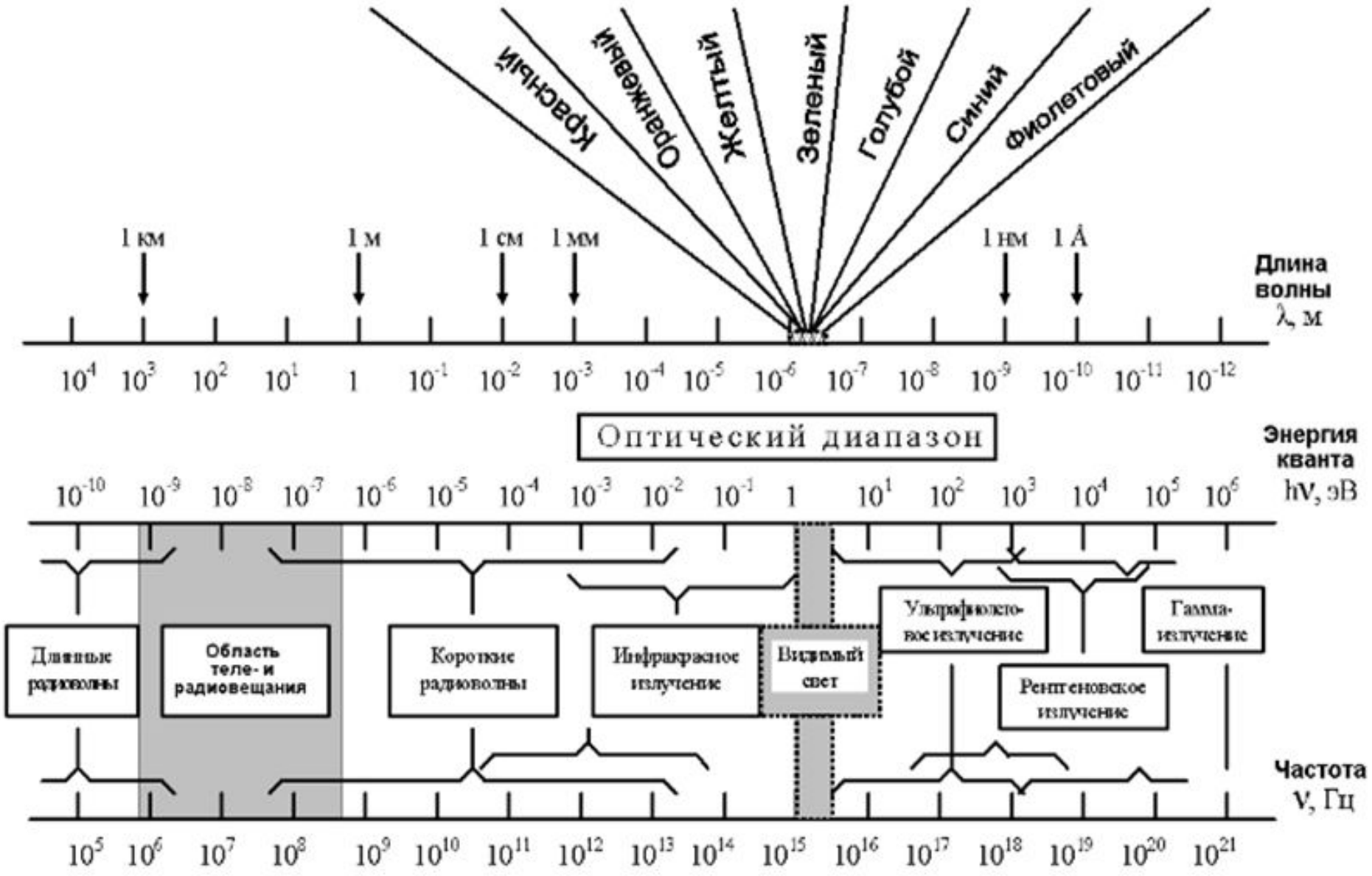
рентгеновская литография, λ=0,1—10 нм;

электронная литография;

ионно-лучевая литография;

нанопечатная литография.

Планарная технология



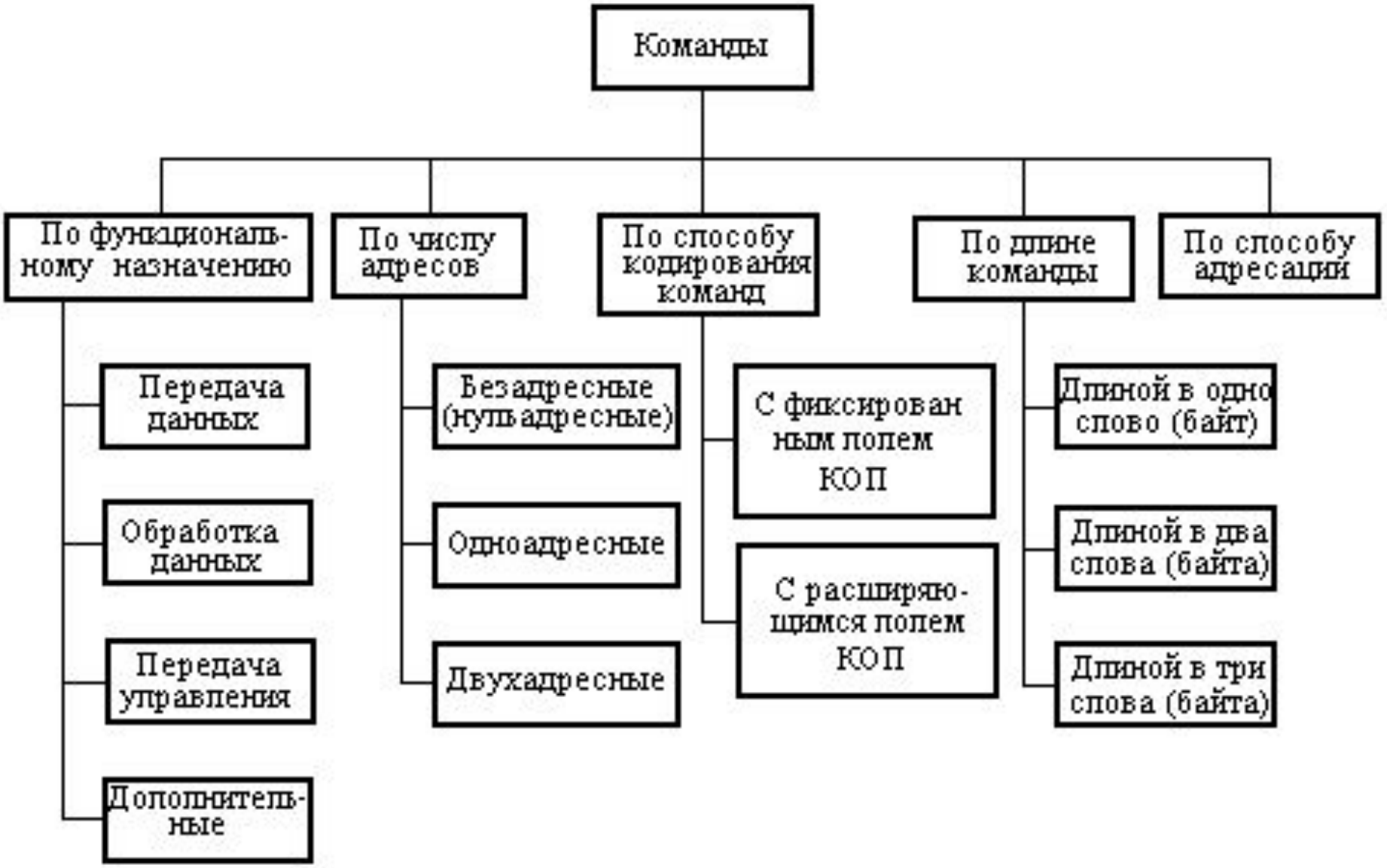
Функции микропроцессора

- вычисление адресов операндов или команд;
- выборка и дешифрация команд из оперативной памяти (ОП);
- выборка данных из ОП, регистров микропроцессорной памяти (МПП) и регистров адаптеров внешних устройств (ВУ);
- прием и обработка запросов и команд от адаптеров на обслуживание ВУ;
- выработка управляющих сигналов для всех прочих узлов и блоков ПК.

Основные параметры микропроцессоров

- разрядность - разрядностью внутренних регистров, над которыми одновременно могут выполняться операции;
- адресное пространство - максимальное количество ячеек основной памяти, которое может быть непосредственно адресовано микропроцессором;
- рабочая тактовая частота – частота тактового генератора, определяющая внутреннее быстродействие процессора;
- кэш-память - память внутри процессора, работающая с частотой ядра;
- состав инструкций - перечень, вид и тип команд исполняемых данным процессором;
- архитектура - тип архитектуры построения процессора;

Классификация команд микропроцессора



Набор команд

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

- **CISC** (Complex Instruction Set Command)
- **RISC** (Reduced Instruction Set Command)
- **VLIW** (Very Length Instruction Word)
- **MISC** (Minimum Instruction Set Command)

МП типа CISC

Характеризуется следующим набором свойств:

- нефиксированное значение длины команды;
- арифметические действия кодируются в одной команде;
- небольшое число регистров, каждый из которых выполняет строго определённую функцию.

Типичными представителями CISC-архитектуры являются процессоры на основе команд x86. Благодаря распространённости процессоров архитектур x86 CISC-системы доминируют в сегментах персональных компьютеров.

МП типа CISC

При этом поздние x86, хотя и CISC-совместимы, но являются процессорами с RISC-ядром, и в формальном смысле считаются гибридными. В таких гибридных CISC-процессорах CISC-инструкции преобразовываются в набор внутренних RISC-команд, при этом одна команда x86 может порождать несколько RISC-команд, исполнение команд происходит на суперскалярном конвейере одновременно по несколько штук.
Основной недостаток CISC-архитектуры в сравнении с RISC — более сложный подход к распараллеливанию вычислений.

Суперскалярный процессор — процессор, поддерживающий так называемый параллелизм на уровне инструкций за счёт включения в состав его вычислительного ядра нескольких одинаковых

МП типа RISC

Вычисления с сокращённым набором команд – это процессоры по следующему принципу: более компактные и простые инструкции выполняются быстрее. Ранние RISC-процессоры даже не имели команд умножения и деления. Идея создания пришла после того как в 1970-х годах в IBM обнаружили, что многие из функциональных особенностей традиционных ЦП игнорировались программистами, кроме того, поскольку некоторые сложные операции использовались редко, они как правило были медленнее, чем те же действия, выполняемые набором простых команд. Это происходило из-за того что создатели процессоров тратили гораздо меньше времени на улучшение сложных команд, чем на улучшение простых. Первые RISC-процессоры были разработаны в начале 1980-х годов в Стэнфордском и Калифорнийском университетах США. Они выполняли небольшой (50-100) набор команд,

Характерные особенности RISC - процессоров

1. Одинаковая длина команд (упрощает выборку из памяти);
2. Использование большого количество регистров -снижает использование ОП;
3. 2-3 способа адресации, в основном регистровая.
4. Сокращенный набор команд - 50-100 команд (позволяет обойтись без схемы микропрограммного управления);
5. Простые способы адресации памяти (обеспечивает отсутствие сложных вычислений адреса);
6. Отсутствие совмещенной операции чтения/записи с обработкой данных;
7. Необходимость соответствующей компиляции программ для повышения эффективности;
8. Несовместимость с набором команд CISC МП .

В настоящее время CISC и RISC сливаются, т.к. большинство CISC МП основаны на ядре RISC.

Достоинства и недостатки

Достоинства:

— высокая тактовая частота;

— высокая скорость выполнения команд;

—уменьшение площади кристалла :МП POWER PC — 121 мм^2 (Apple), Pentium— 292 мм^2 (Intel)

— уменьшение мощности потребления: МП POWER PC — 8,5 Вт, Pentium — 16 Вт.

— уменьшение стоимости.

Недостатки:

— необходимость моделирования сложных команд;

МП типа VLIW

Архитектура МП с несколькими вычислительными устройствами. Одна инструкция процессора содержит несколько операций, которые выполняются параллельно. В суперскалярных процессорах также есть несколько вычислительных модулей, но задача распределения между ними работы решается аппаратно. Это сильно усложняет процессор и может быть чревато ошибками. В процессорах VLIW задача распределения решается во время компиляции и в инструкциях явно указано, какое вычислительное устройство должно выполнять какую команду.

VLIW можно считать продолжением идеологии RISC, расширяющей её на архитектуры с несколькими вычислительными модулями. Так же, как в RISC, в инструкции явно указывается, что именно должен делать каждый модуль процессора. Из-за этого длина

МП типа VLIW

Подход VLIW **сильно упрощает архитектуру процессора**, перекладывая задачу распределения вычислительных устройств на компилятор. Поскольку отсутствуют большие и сложные узлы, **значительно снижается энергопотребление**.
В то же время, код для VLIW обладает невысокой плотностью. Из-за большого количества пустых инструкций для простаивающих устройств программы для VLIW-процессоров могут быть гораздо длиннее, чем аналогичные программы для традиционных архитектур.

Из-за сложных внутренних зависимостей кода, программирование на уровне машинных кодов для VLIW-архитектур вручную практически невозможно. Приходится полагаться на оптимизацию компилятора, который сам может содержать ошибки.

МП типа MISC

Процессор, работающий с **минимальным набором длинных команд**.

Увеличение разрядности процессоров привело к идее укладки нескольких команд в одно большое слово. Это позволило использовать возросшую производительность компьютера и его возможность обрабатывать одновременно несколько потоков данных.

Процессоры, образующие «компьютеры с минимальным набором команд» MISC, как и процессоры RISC, характеризуются **небольшим числом чаще всего встречающихся команд**.

Вместе с этим, принцип «**очень длинных слов команд**» VLIW обеспечивает выполнение группы команд за один цикл работы процессора.

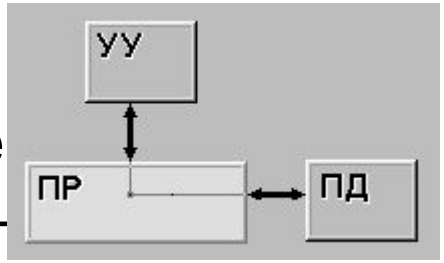
Таким образом, архитектура **MISC объединила вместе** суперскалярную RISC и VLIW концепции. Компоненты такого

Классификация Флинна

В 1966 г. Флинном был предложен подход к классификации архитектур вычислительных систем. В его основу было положено понятие потока, под которым понимается последовательность элементов, команд или данных, обрабатываемая процессором.
Соответствующая система классификации основана на рассмотрении числа потоков инструкций и потоков данных и описывает четыре архитектурных класса.

- **SISD = Single Instruction Single Data**
- **MISD = Multiple Instruction Single Data**
- **SIMD = Single Instruction Multiple Data**
- **MIMD = Multiple Instruction Multiple Data**

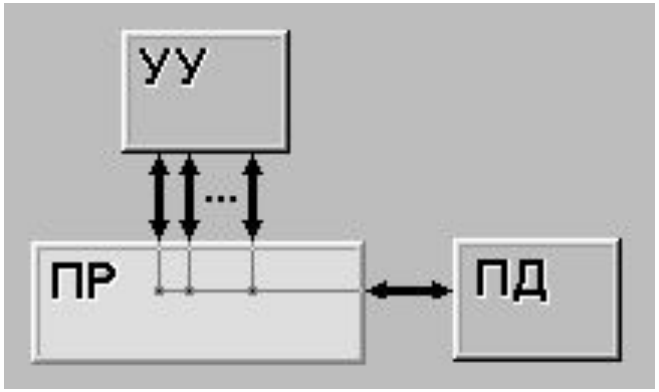
Одиночный поток команд и одиночный поток данных



К этому классу относятся последовательные компьютерные системы, которые имеют один центральный процессор, способный обрабатывать только один поток последовательно исполняемых инструкций. В настоящее время практически все высокопроизводительные системы имеют более одного центрального процессора, однако каждый из них выполняет несвязанные потоки инструкций, что делает такие системы комплексами SISD-систем, действующих на разных пространствах данных. Для увеличения скорости обработки команд и скорости выполнения арифметических операций может применяться конвейерная обработка. В случае векторных систем векторный поток данных следует рассматривать как поток из одиночных неделимых

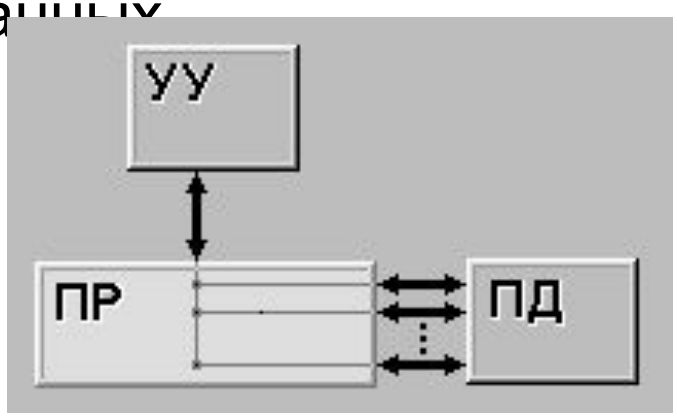
Множественный поток команд и одиночный поток данных

Ни Флинн, ни другие специалисты в области архитектуры компьютеров до некоторого времени не могли представить убедительный пример реально существующей вычислительной системы, построенной на данном принципе. Ряд исследователей относят конвейерные машины к данному классу, однако это не нашло окончательного признания в научном сообществе. Появившиеся многоядерные компьютеры можно отнести к данному классу.



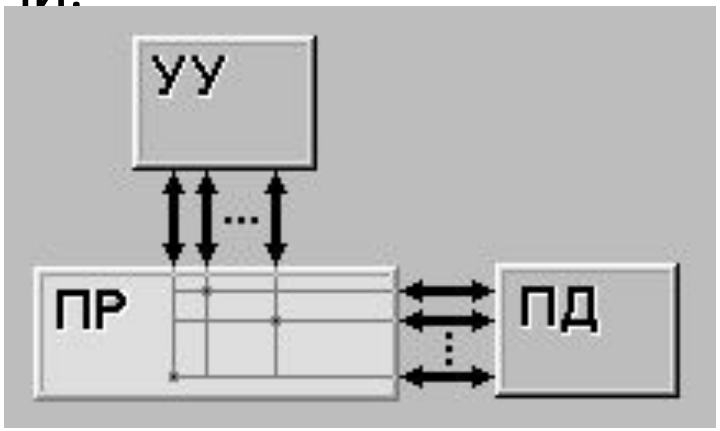
Одиночный поток команд и множественный поток данных

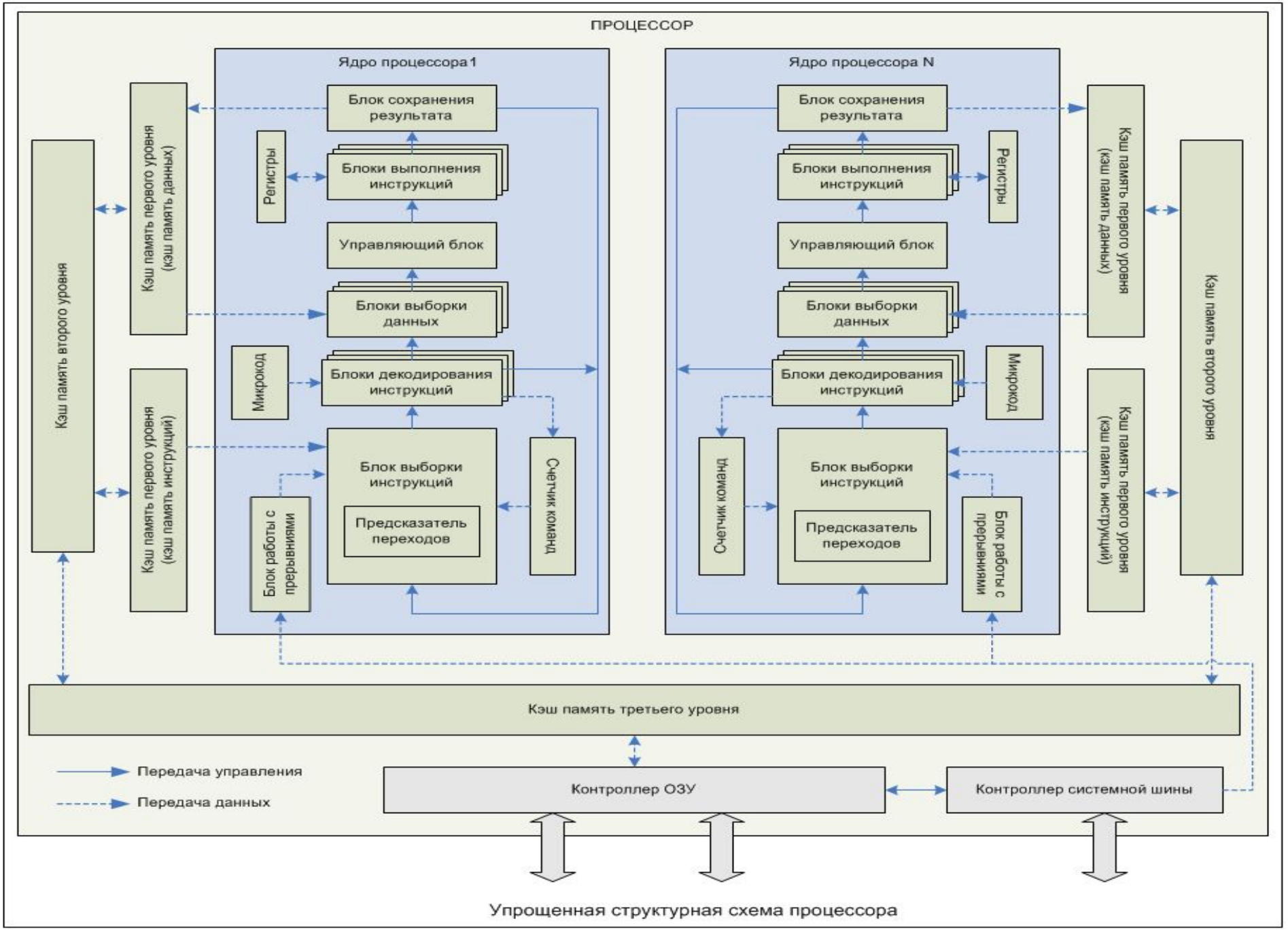
В таких системах единое управляющее устройство контролирует множество процессорных элементов. Каждый процессорный элемент получает от устройства управления в каждый фиксированный момент времени одинаковую команду и выполняет ее над своими локальными данными. В этот же класс можно включить и векторно-конвейерные машины. В этом случае каждый элемент вектора надо рассматривать как отдельный элемент потока данных



Множественный поток команд и множественный поток данных

Эти машины параллельно выполняют несколько потоков инструкций над различными потоками данных. В отличие от упомянутых выше многопроцессорных SISD-машин, команды и данные связаны, потому что они представляют различные части одной и той же задачи. Например, MIMD-системы могут параллельно выполнять множество подзадач с целью сокращения времени выполнения основной задачи.





Упрощенная структурная схема процессора

Ядро процессора

Каждое ядро процессора состоит из нескольких функциональных блоков:

- блока выборки инструкций;
- блоков декодирования инструкций;
- блоков выборки данных;
- управляющего блока;
- блоков выполнения инструкций;
- блоков сохранения результатов;
- блока работы с прерываниями;
- ПЗУ, содержащего микрокод;
- набора регистров;
- счетчика команд.

Блоки выборки инструкций и предсказатель перехода

- **Блок выборки инструкций** осуществляет считывание инструкций по адресу, указанному в счетчике команд. Обычно, за такт он считывает несколько инструкций. Количество считываемых инструкций обусловлено количеством блоков декодирования, так как необходимо на каждом такте работы максимально загрузить блоки декодирования. Для того чтобы блок выборки инструкций работал оптимально, в ядре процессора имеется предсказатель переходов.
- **Предсказатель переходов** пытается определить, какая последовательность команд будет выполняться после совершения перехода. Это необходимо, чтобы после условного перехода максимально нагрузить конвейер ядра процессора.

Блоки декодирования

Это блоки, которые занимаются декодированием инструкций, т.е. определяют, что надо сделать процессору, и какие дополнительные данные нужны для выполнения инструкции. Задача эта для большинства современных процессоров, построенных на базе концепции CISC, – очень сложная. Дело в том, что длина инструкций и количество операндов – нефиксированные, и это сильно усложняет жизнь разработчикам процессоров и делает процесс декодирования нетривиальной задачей.

Часто отдельные сложные команды приходится заменять микрокодом – серией простых инструкций, в совокупности выполняющих то же действие, что и одна сложная инструкция. Набор микрокода прошит в ПЗУ, встроенном в процессоре. В современных процессорах, обычно, бывает 2-4 блока декодирования инструкций

Блоки выборки данных и управляющий блок

- **Блоки выборки данных** осуществляют выборку данных из КЭШ-памяти или ОЗУ, необходимых для выполнения текущих инструкций. Обычно, каждое процессорное ядро содержит несколько блоков выборки данных.
- **Управляющий блок** на основании декодированных инструкций управляет работой блоков выполнения инструкций, распределяет нагрузку между ними, обеспечивает своевременное и верное выполнение инструкций. Это один из наиболее важных блоков ядра процессора.

Блоки выполнения инструкций

Включают в себя несколько разнотипных блоков:

- ALU – арифметическое логическое устройство;
- FPU – устройство по выполнению операций с плавающей точкой;
- Блоки для обработки расширения наборов инструкций.

Дополнительные инструкции используются для ускорения обработки потоков данных, шифрования и дешифрования, кодирования видео и так далее. Для этого в ядро процессора вводят дополнительные регистры и наборы логики.

Популярными расширениями наборов инструкция являются:

- MMX (Multimedia Extensions) – набор инструкций, разработанный компанией Intel, для ускорения кодирования и декодирования потоковых аудио и видео-данных;
- SSE (Streaming SIMD Extensions) – набор инструкций, разработанный компанией Intel, для выполнения одной и той же последовательности операций над множеством данных с распараллеливанием вычислительного процесса;
- ATA (Application Targeted Accelerator) – набор инструкций, разработанный компанией Intel, для ускорения работы специализированного программного обеспечения и снижения энергопотребления при работе с такими программами;
- 3DNow – набор инструкций, разработанный компанией AMD, для расширения возможностей набора инструкций MMX;
- AES (Advanced Encryption Standard) – набор инструкций

Блоки сохранения результатов и работы с прерываниями

- **Блок сохранения результатов** обеспечивает запись результата выполнения инструкции в ОЗУ по адресу, указанному в обрабатываемой инструкции.
- **Блок работы с прерываниями.** Работа с прерываниями позволяет процессору своевременно реагировать на события, прерывать ход работы программы и выполнять требуемые от него действия. Благодаря наличию прерываний, процессор способен к псевдопараллельной работе, т.е. к, так называемой, многозадачности.

Блок работы с прерываниями

Обработка прерываний происходит следующим образом. Процессор перед началом каждого цикла работы проверяет наличие запроса на прерывание. Если есть прерывание для обработки, процессор сохраняет в стек адрес инструкции, которую он должен был выполнить, и данные, полученные после выполнения последней инструкции, и переходит к выполнению функции обработки прерывания.

После окончания выполнения функции обработки прерывания, из стека считываются сохраненные в него данные, и процессор возобновляет выполнение восстановленной задачи.

Регистры

Регистры – сверхбыстрая оперативная память (доступ к регистрам в несколько раз быстрее доступа к КЭШ-памяти) небольшого объема (несколько сотен байт), входящая в состав процессора, для временного хранения промежуточных результатов выполнения инструкций. Регистры процессора делятся на два типа: регистры общего назначения и специальные регистры.

Регистры общего назначения используются при выполнении арифметических и логических операций, или операций дополнительных наборов инструкций (MMX, SSE и т.д.).

Регистры специального назначения содержат системные данные, необходимые для работы процессора. К таким регистрам относятся, например, регистры управления, регистры системных адресов, регистры отладки и т.д. Доступ к этим регистрам жестко регламентирован.

Счетчик команд

- **Счетчик команд** – регистр, содержащий адрес команды, которую процессор начнет выполнять на следующем такте работы.

Цикл работы ядра процессора

1. Блок выборки инструкций проверяет наличие прерываний. Если прерывание есть, то данные регистров и счетчика команд заносятся в стек, а в счетчик команд заносится адрес команды обработчика прерываний. По окончанию работы функции обработки прерываний, данные из стека будут восстановлены.
2. Блок выборки инструкций из счетчика команд считывает адрес команды, предназначенной для выполнения. По этому адресу из КЭШ-памяти или ОЗУ считывается команда. Полученные данные передаются в блок декодирования.
3. Блок декодирования команд расшифровывает команду, при необходимости используя для интерпретации команды записанный в ПЗУ микрокод. Если это команда перехода, то в счетчик команд записывается адрес перехода и управление передается в блок выборки инструкций (пункт 1), иначе счетчик команд увеличивается на размер команды (для процессора с длинной команды 32 бита – на 4) и передает управление в блок

Цикл работы ядра процессора

4. Блок выборки данных считывает из КЭШ-памяти или ОЗУ требуемые для выполнения команды данные и передает управление планировщику.
5. Управляющий блок определяет, какому блоку выполнения инструкций обработать текущую задачу, и передает управление этому блоку.
6. Блоки выполнения инструкций выполняют требуемые командой действия и передают управление блоку сохранения результатов.
7. При необходимости сохранения результатов в ОЗУ, блок сохранения результатов выполняет требуемые для этого действия и передает управление блоку выборки инструкций (пункт 1).

Описанный выше цикл называется процессом (именно

Технологии повышения производительности ядра процессора

Увеличение производительности ядра процессора, за счет поднятия тактовый частоты, имеет жесткое ограничение. Увеличение тактовой частоты влечет за собой повышение температуры процессора, энергопотребления и снижение стабильности его работы и срока службы.

Поэтому разработчики процессоров применяют различные архитектурные решения, позволяющие увеличить производительность процессоров без увеличения тактовой частоты.

Рассмотрим основные способы повышения производительности процессоров.

Конвейеризация

Каждая инструкция, выполняемая процессором, последовательно проходит все блоки ядра, в каждом из которых совершается своя часть действий, необходимых для выполнения инструкции. Если приступать к обработке новой инструкции только после завершения работы над первой инструкцией, то большая часть блоков ядра процессора в каждый момент времени будет простаивать, а, следовательно, возможности процессора будут использоваться не полностью.

Рассмотрим пример, в котором процессор будет выполнять программу, состоящую из пяти инструкций (К1–К5), без использования принципа конвейеризации. Для упрощения примера примем, что каждый блок ядра процессора выполняет инструкцию за 1 такт.

Конвейеризация. Пример

Такты	Выборка инструкции	Декодирование инструкции	Выборка данных	Выполнение инструкции	Сохранение результата
1	K1	-	-	-	-
2	-	K1	-	-	-
3	-	-	K1	-	-
4	-	-	-	K1	-
5	-	-	-	-	K1
6	K2	-	-	-	-
7	-	K2	-	-	-
8	-	-	K2	-	-
9	-	-	-	K2	-
10	-	-	-	-	K2
11	K3	-	-	-	-
12	-	K3	-	-	-
13	-	-	K3	-	-
14	-	-	-	K3	-
15	-	-	-	-	K3
16	K4	-	-	-	-
17	-	K4	-	-	-
18	-	-	K4	-	-
19	-	-	-	K4	-
20	-	-	-	-	K4
21	K5	-	-	-	-
22	-	K5	-	-	-
23	-	-	K5	-	-
24	-	-	-	K5	-
25	-	-	-	-	K5

Конвейеризация

- Как видно, для выполнения пяти инструкций процессору понадобилось 25 тактов. При этом в каждом такте четыре из пяти блоков ядра процессора простаивали, т.е. процессор использовал всего 20% своего потенциала. В реальных процессорах все сложнее. Разные блоки процессора решают разные по сложности задачи. Сами инструкции тоже отличаются друг от друга по сложности. Но в ситуация остается такой же.
- Для решения этой проблемы во всех современных процессорах выполнение инструкций построено по принципу конвейера, то есть по мере освобождения блоков ядра, они загружаются обработкой следующей инструкции.
- Рассмотрим пример выполнения той же программы, состоящей из пяти инструкций, но с использованием принципа конвейеризации

Конвейеризация. Пример

Такты	Выборка инструкции	Декодирование инструкции	Выборка данных	Выполнение инструкции	Сохранение результата
1	K1	-	-	-	-
2	K2	K1	-	-	-
3	K3	K2	K1	-	-
4	K4	K3	K2	K1	-
5	K5	K4	K3	K2	K1
6	-	K5	K4	K3	K2
7	-	-	K5	K4	K3
8	-	-	-	K5	K4
9	-	-	-	-	K5

Та же программа была выполнена за 9 тактов, что почти 2.8 раза быстрее. Максимальная загрузка процессора была получена на 5 такте. В этот момент использовались все блоки ядра процессора. Так как процессор выполняет команды непрерывно, то он мог бы быть занят на 100%, при этом, чем длиннее был бы конвейер, тем больший выигрыш в производительности был бы получен. Но на практике это

Конвейеризация

Во-первых, реальный поток команд, обрабатываемый процессором – не последовательный. В нем часто встречаются переходы. При этом пока команда условного перехода не будет обработана полностью, конвейер не сможет начать выполнение новой команды, так как не знает, по какому адресу она находится. После условного перехода конвейер приходиться наполнять заново. И чем длиннее конвейер, тем дольше это происходит. В результате, прирост производительности от введения конвейера снижается.

Для уменьшения влияния условных переходов на работу конвейера, в ядро процессора вводятся блоки предсказания условных переходов. Основная задача этих блоков – определить, когда будет совершен условный переход и какие команды будут выполнены после

Конвейеризация

Если условный переход удалось предсказать, то выполнение инструкций по новому адресу начинается раньше, чем будет закончена обработка команды условного перехода. По статистике, точность блоков предсказания условных переходов в современных процессорах превышает 90%, что позволяет делать достаточно длинные и при этом хорошо наполняемые конвейеры. Во-вторых, часто обрабатываемые инструкции – взаимосвязаны, то есть одна из инструкций требует в качестве исходных данных результата выполнения другой инструкции.
В этом случае она может быть выполнена только после полного завершения обработки первой инструкции. Однако современные процессоры могут анализировать код на несколько инструкций вперед и, например, параллельно с первой инструкцией обработать третью инструкцию,

Конвейеризация

В большинстве современных процессорах задача анализа взаимосвязи инструкций и составления порядка их обработки ложится на плечи процессора, что неминуемо ведет к снижению его быстродействия и увеличению стоимости.
Однако все большую популярность получает статическое планирование, когда порядок выполнения программы процессором определяется на этапе компиляции программы. В этом случае инструкции, которые можно выполнить параллельно, объединяются компилятором в одну длинную команду, в которой все инструкции заведомо параллельны. Процессоры, работающие с такими инструкциями, построены на базе архитектура VLIW (Very long instruction word).

Суперскалярность

Суперскалярность – архитектура вычислительного ядра, при которой наиболее нагруженные блоки могут входить в нескольких экземплярах. Скажем, в ядре процессора блок выборки инструкций может нагружать сразу несколько блоков декодирования.
В этом случае блоки, выполняющие более сложные действия и работающие дольше, за счет параллельной обработки сразу нескольких инструкций не будут задерживать весь конвейер.
Однако параллельное выполнение инструкций возможно, только если эти инструкции – независимые.

Многоядерность

Подавляющее большинство современных процессоров имеют два и более ядра. Мы практически получаем несколько процессоров, способных независимо решать каждый свои задачи, при этом возрастает производительность. Однако прирост производительности далеко не всегда оправдывает ожидания.
Во-первых не все программы поддерживают распределение вычислений на несколько ядер. Можно программы разделять между ядрами, чтобы на каждом ядре работал свой набор независимых программ. Но это дает выигрыш в производительности до тех пор, пока не появляется программа, требующая ресурсов больше, чем может дать одно ядро.

Многоядерность

Во-вторых, усложняется работа с памятью, так как ядер – много, и всем им требуется доступ к ОЗУ. Требуется сложный механизм, определяющий очередность доступа ядер процессора к памяти и к другим ресурсам ЭВМ.
В-третьих, возрастает энергопотребление, а, следовательно, увеличивается тепловыделение и требуется мощная система охлаждения.
В-четвертых, себестоимость производства многоядерных процессоров высокая.
Несмотря на все недостатки, применение многоядерных процессоров дает значительный прирост производительности.

Технология Hyper-Threading

Технология Intel Hyper-threading позволяет каждому ядру процессора выполнять две задачи одновременно, делая из одного реального ядра два виртуальных. Это возможно из-за того, что в таких ядрах сохраняется состояние сразу двух потоков, так как у ядра есть свой набор регистров, свой счетчик команд и свой блок работы с прерываниями для каждого потока. В результате, операционная система видит такое ядро, как два отдельных ядра, и будет с ними работать так же, как работала бы с двуядерным процессором.

Однако остальные элементы ядра для обоих потоков – общие, и делятся между ними. Кроме этого, когда по какой-либо причине один из потоков освобождает элементы конвейера, другой поток использует свободные блоки.

Технология Hyper-Threading

Большинство программ не могут полностью нагрузить процессор, так как некоторые, в основном, используют несложные целочисленные вычисления, практически не задействуя блок FPU. Другие же программы, например 3D-студия, требуют массу расчетов с использованием чисел с плавающей точкой, но при этом освобождая некоторые другие исполнительные блоки и так далее.
К тому же практически во всех программах – много условных переходов и зависимых переменных. В результате, использование технологии Hyper-threading может дать существенный прирост производительности, способствуя максимальной загрузке конвейера ядра.

Технология Hyper-Threading

Естественно, прирост производительности будет меньше, чем от использования нескольких физических ядер, так как потоки используют общие блоки одного конвейера и часто вынуждены ждать освобождения требуемого блока. К тому же большинство процессоров уже имеют несколько физических ядер, и при использовании технологии Hyper-threading виртуальных ядер может стать слишком много, особенно, если процессор содержит четыре и больше физических ядра. Может сложиться ситуация, когда параллельно будут работать два схожих потока, часто использующие одни и те же блоки. В таком случае прирост производительности будет минимален. В результате, технология Hyper-Threading очень зависима от типа нагрузки на процессор и может дать хороший прирост производительности, а может быть практически бесполезной.

Технология Turbo Boost

Производительность большинства современных процессоров в поднять, разогнать – заставить работать на частотах, превышающих номинальную.
Частота процессора рассчитывается, как частота системной шины, умноженная на некий коэффициент, называемый множителем. Например, процессор Core i7-970 работает с системной шиной DMI на базовой частоте – 133 МГц, и имеет множитель – 24. Таким образом, тактовая частота ядра процессора составит: 133 Мгц*24= 3192 Мгц. Если в настройках BIOS увеличить множитель или поднять тактовую частоту системной шины, то тактовая частота процессора увеличится, а, соответственно, увеличится и его производительность. Однако процесс этот – далеко небезопасный. Из-за разгона процессор может работать нестабильно или вообще выйти из строя.

Технология Turbo Boost

Процессоры с технологией Turbo Boost могут сами динамически, на короткий промежуток времени, повышать тактовую частоту, тем самым, увеличивая свою производительность. При этом процессор контролирует все параметры своей работы: напряжение, силу тока, температуру и т.д., не допуская сбоев и тем более выхода из строя. Например, процессор может отключить неиспользуемые ядра, тем самым, понизив общую температуру, а взамен увеличить тактовую частоту остальных ядер. Применение технологии Turbo Boost позволяет значительно поднять производительность процессора, особенно, при работе с однопоточными приложениями.