## Что такое микропроцессорная система? Особенности традиционной цифровой системы.

Микропроцессорная система может рассматриваться как частный случай электронной системы, предназначенной для обработки входных сигналов и выдачи выходных сигналов (рис. 1.1). В качестве входных и выходных сигналов при этом могут использоваться аналоговые сигналы, одиночные цифровые сигналы, цифровые коды, последовательности цифровых кодов. Внутри системы может производиться хранение, накопление сигналов (или информации), но суть от этого не меняется. Если система цифровая (а микропроцессорные системы относятся к разряду цифровых), то входные аналоговые сигналы преобразуются в последовательности кодов выборок с помощью АЦП, а выходные аналоговые сигналы формируются из последовательности кодов выборок с помощью ЦАП. Обработка и хранение информации производятся в цифровом виде. Характерная особенность традиционной цифровой системы состоит в том, что алгоритмы обработки и хранения информации в ней жестко связаны со схемотехникой системы. То есть изменение этих алгоритмов возможно только путем изменения структуры системы, замены электронных узлов, входящих в систему, и/или связей между ними. Например, если нам нужна дополнительная операция суммирования, то необходимо добавить в структуру системы лишний сумматор. Или если нужна дополнительная функция хранения кода в течение одного такта, то мы должны добавить в структуру еще один регистр. Естественно, это практически невозможно сделать в процессе эксплуатации, обязательно нужен новый производственный цикл проектирования, изготовления, отладки всей системы. Именно поэтому традиционная цифровая система часто называется системой на «жесткой логике».

## Что такое микропроцессорная система? Особенности системы на гибкой логике?

Недостатки традиционной цифровой системы:

· для каждой новой задачи ее надо проектировать и изготавливать заново. (длительный, дорогостоящий, требующий высокой квалификации исполнителей процесс).

Возникла потребность в системе, которая могла бы легко адаптироваться под любую задачу, перестраиваться с одного алгоритма работы на другой без изменения аппаратуры. Задавать тот или иной алгоритм работы такой системы мы могли бы путем ввода в систему некой управляющей информации - программы. Данная система обладает свойством универсальности, или является программируемой, не «жесткой», а «гибкой». Именно это и обеспечивает микропроцессорная система.

Рассмотри особенности микропроцессорных систем:

1. Избыточность универсальных систем, увеличение стоимости, снижение надежности, увеличение потребляемой мощности и т.д.

*Решение максимально трудной задачи требует гораздо больше средств, чем решение простой задачи. Поэтому сложность универсальной системы должна быть такой, чтобы обеспечивать решение самой трудной задачи, а при решении простой задачи система будет работать далеко не в полную силу, будет использовать не все свои ресурсы. И чем*

*проще решаемая задача, тем больше избыточность, и тем менее оправданной становится универсальность.*

2. Снижение быстродействия универсальных систем.

*Оптимизировать универсальную систему так, чтобы каждая новая задача решалась максимально быстро, попросту невозможно. Общее правило таково: чем больше универсальность, гибкость, тем меньше быстродействие. Более того, для универсальных систем не существует таких задач (пусть даже и самых простых), которые бы они решали с максимально возможным быстродействием.*

Вывод: Цифровые системы (на "жесткой логике") используются при решении задачи, которая не меняется длительное время, где требуется самое высокое быстродействие, где алгоритмы обработки информации предельно просты. Микропроцессорные системы (универсальные, программируемые) оптимально использовать при решении часто меняющихся задач, где высокое быстродействие не слишком важно, где алгоритмы обработки информации сложные.

1. Преимущества и недостатки систем на жесткой и гибкой логике?

Микропроцессорная система - электронная система, предназначенная для обработки входных сигналов и выдачи выходных сигналов. В качестве входных и выходных сигналов при этом могут использоваться

* аналоговые сигналы (входные аналоговые сигналы преобразуются в последовательности кодов выборок с помощью АЦП, выходные аналоговые сигналы формируются из последовательности кодов выборок с помощью ЦАП),
* одиночные цифровые сигналы,
* цифровые коды,
* последовательности цифровых кодов.

Внутри системы производится хранение, накопление сигналов (или информации)

Обработка и хранение информации производятся в цифровом виде.

В цифровой системе алгоритмы обработки и хранения информации жестко связаны со схемотехникой системы (изменение алгоритмов возможно только путем изменения структуры системы, замены электронных узлов, входящих в систему, и/или связей между ними - например, дополнительная операция суммирования: добавить в цифровую систему сумматор, дополнительная функция хранения кода в течение одного такта - добавить регистр. Естественно, это практически невозможно сделать в процессе эксплуатации, обязательно нужен новый производственный цикл проектирования, изготовления, отладки всей системы. Именно поэтому цифровая система часто называется системой на "жесткой логике" - специализированная система, настроенная исключительно на одну задачу или (реже) на несколько близких, заранее известных задач.

Преимущества:

· отсутствие аппаратурной избыточности, то есть каждый ее элемент обязательно работает в полную силу (конечно, если эта система грамотно спроектирована).

· обеспечивает максимально высокое быстродействие, так как скорость выполнения алгоритмов обработки информации определяется в ней только быстродействием отдельных логических элементов и выбранной схемой путей прохождения информации. (логические элементы обладают максимальным на данный момент быстродействием)

Недостаток

· для каждой новой задачи ее надо проектировать и изготавливать заново. (длительный, дорогостоящий, требующий высокой квалификации исполнителей процесс).

Возникла потребность в системе, которая могла бы легко адаптироваться под любую задачу, перестраиваться с одного алгоритма работы на другой без изменения аппаратуры. Задавать тот или иной алгоритм работы такой системы мы могли бы путем ввода в систему некой управляющей информации - программы. Данная система обладает свойством универсальности, или является программируемой, не «жесткой», а «гибкой». Именно это и обеспечивает микропроцессорная система.

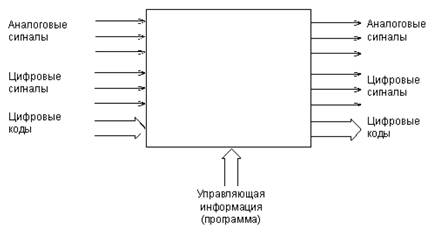


Рис. Микропроцессорная система

Рассмотри особенности микропроцессорных систем:

1. Избыточность универсальных систем, увеличение стоимости, снижение надежности, увеличение потребляемой мощности и т.д.

*Решение максимально трудной задачи требует гораздо больше средств, чем решение простой задачи. Поэтому сложность универсальной системы должна быть такой, чтобы обеспечивать решение самой трудной задачи, а при решении простой задачи система будет работать далеко не в полную силу, будет использовать не все свои ресурсы. И чем проще решаемая задача, тем больше избыточность, и тем менее оправданной становится универсальность.*

2. Снижение быстродействия универсальных систем.

*Оптимизировать универсальную систему так, чтобы каждая новая задача решалась максимально быстро, попросту невозможно. Общее правило таково: чем больше универсальность, гибкость, тем меньше быстродействие. Более того, для универсальных систем не существует таких задач (пусть даже и самых простых), которые бы они решали с максимально возможным быстродействием.*

Вывод: Цифровые системы (на "жесткой логике") используются при решении задачи, которая не меняется длительное время, где требуется самое высокое быстродействие, где алгоритмы обработки информации предельно просты. Микропроцессорные системы (универсальные, программируемые) оптимально использовать при решении часто меняющихся задач, где высокое быстродействие не слишком важно, где алгоритмы обработки информации сложные.

*За последние десятилетия быстродействие микропроцессорных систем сильно выросло (на несколько порядков). К тому же большой объем выпуска микросхем для этих систем привел к резкому снижению их стоимости. В результате область применения цифровых систем (на "жесткой логике") резко сузилась.*

Появились программируемые системы, предназначенные для решения одной задачи или нескольких близких задач (ПЛИС – программируемые логические интегральные микросхемы). Они удачно совмещают в себе как достоинства цифровых систем, так и программируемых систем, обеспечивая сочетание достаточно высокого быстродействия и необходимой гибкости. Так что вытеснение "жесткой логики" продолжается.

## Что такое Большая интегральная схема? Какие бывают процессоры по количеству БИС?

**Интегра́льная** (engl. Integrated circuit, IC, microcircuit, microchip, silicon chip, or chip), (**микро**)**схе́ма** (**ИС, ИМС, м/сх**), **чип**, **микрочи́п** ([англ.](https://dic.academic.ru/dic.nsf/ruwiki/6161) *chip* — щепка, обломок, фишка) — [микроэлектронное](https://dic.academic.ru/dic.nsf/ruwiki/195253) устройство — электронная схема произвольной сложности, изготовленная на [полупроводниковом](https://dic.academic.ru/dic.nsf/ruwiki/19202) кристалле (или плёнке) и помещённая в неразборный корпус. Часто под **интегральной схемой** (ИС) понимают собственно кристалл или плёнку с электронной схемой, а под **микросхемой** (МС) — ИС, заключённую в корпус.

Больша́я интегра́льная схе́ма (БИС) — [интегральная схема](https://megabook.ru/article/%d0%98%d0%bd%d1%82%d0%b5%d0%b3%d1%80%d0%b0%d0%bb%d1%8c%d0%bd%d0%b0%d1%8f%20%d1%81%d1%85%d0%b5%d0%bc%d0%b0) соответствующая большой степени интеграции (плотности упаковки (размещения) на одном кристалле ([чипе](https://megabook.ru/article/%d0%a7%d0%b8%d0%bf)) элементов интегральных схем). Число схемных элементов в БИС от 103 до 104. БИС используются в электронной аппаратуре как функционально законченные узлы устройств вычислительной техники, автоматики, измерительной техники и др. В настоящее время наряду с БИС актуальными являются схемы со сверхбольшой степенью интеграции (СБИС), характеризующиеся плотностью упаковки от 104 до 106 элементов на кристалл и ультрабольшой степенью интеграции (УБИС) — до 109 элементов на кристалл.

По числу больших интегральных схем (БИС) в микропроцессорном комплекте различают микропроцессоры однокристальные, многокристальные и многокристальные секционные.

Однокристальные микропроцессоры получаются при реализации всех аппаратных средств процессора в виде одной БИС или СБИС (сверхбольшой интегральной схемы). По мере увеличения степени интеграции элементов в кристалле и числа выводов корпуса параметры однокристальных микропроцессоров улучшаются. Однако возможности однокристальных микропроцессоров ограничены аппаратными ресурсами кристалла и корпуса. Для получения многокристального микропроцессора необходимо провести разбиение его логической структуры на функционально законченные части и реализовать их в виде БИС (СБИС). Функциональная законченность БИС многокристального микропроцессора означает, что его части выполняют заранее определенные функции и могут работать автономно.

Многокристальные секционные микропроцессоры получаются в том случае, когда в виде БИС реализуются части (секции) логической структуры процессора при функциональном разбиении ее вертикальными плоскостями (рис. 1,б). Для построения многоразрядных микропроцессоров при параллельном включении секций БИС в них добавляются средства "стыковки".

## Виды процессоров по назначению? Преимущества и недостатки?

По назначению различают универсальные и специализированные микропроцессоры.

Универсальные микропроцессоры могут быть применены для решения широкого круга разнообразных задач. При этом их эффективная производительность слабо зависит от проблемной специфики решаемых задач. Специализация МП, т.е. его проблемная ориентация на ускоренное выполнение определенных функций позволяет резко увеличить эффективную производительность при решении только определенных задач.

Среди специализированных микропроцессоров можно выделить различные микроконтроллеры, ориентированные на выполнение сложных последовательностей логических операций, математические МП, предназначенные для повышения производительности при выполнении арифметических операций за счет, например, матричных методов их выполнения, МП для обработки данных в различных областях применений и т. д. С помощью специализированных МП можно эффективно решать новые сложные задачи параллельной обработки данных. Например, конволюция позволяет осуществить более сложную математическую обработку сигналов, чем широко используемые методы корреляции. Последние в основном сводятся к сравнению всего двух серий данных: входных, передаваемых формой сигнала, и фиксированных опорных и к определению их подобия. Конволюция дает возможность в реальном масштабе времени находить соответствие для сигналов изменяющейся формы путем сравнения их с различными эталонными сигналами, что, например, может позволить эффективно выделить полезный сигнал на фоне шума.

## Виды процессоров по виду обрабатываемых сигналов? Преимущества и недостатки?

По виду обрабатываемых входных сигналов различают цифровые и аналоговые микропроцессоры. Сами микропроцессоры цифровые устройства, однако могут иметь встроенные аналого-цифровые и цифро-аналоговые преобразователи. Поэтому входные аналоговые сигналы передаются в МП через преобразователь в цифровой форме, обрабатываются и после обратного преобразования в аналоговую форму поступают на выход. С архитектурной точки зрения такие микропроцессоры представляют собой аналоговые функциональные преобразователи сигналов и называются аналоговыми микропроцессорами. Они выполняют функции любой аналоговой схемы (например, производят генерацию колебаний, модуляцию, смещение, фильтрацию, кодирование и декодирование сигналов в реальном масштабе времени и т.д., заменяя сложные схемы, состоящие из операционных усилителей, катушек индуктивности, конденсаторов и т.д.). При этом применение аналогового микропроцессора значительно повышает точность обработки аналоговых сигналов и их воспроизводимость, а также расширяет функциональные возможности за счет программной "настройки" цифровой части микропроцессора на различные алгоритмы обработки сигналов.

Обычно в составе однокристальных аналоговых МП имеется несколько каналов аналого-цифрового и цифро-аналогового преобразования. В аналоговом микропроцессоре разрядность обрабатываемых данных достигает 24 бит и более, большое значение уделяется увеличению скорости выполнения арифметических операций.

Отличительная черта аналоговых микропроцессоров способность к переработке большого объема числовых данных, т. е. к выполнению операций сложения и умножения с большой скоростью при необходимости даже за счет отказа от операций прерываний и переходов. Аналоговый сигнал, преобразованный в цифровую форму, обрабатывается в реальном масштабе времени и передается на выход обычно в аналоговой форме через цифро-аналоговый преобразователь. При этом согласно теореме Котельникова частота квантования аналогового сигнала должна вдвое превышать верхнюю частоту сигнала.

Сравнение цифровых микропроцессоров производится сопоставлением времени выполнения ими списков операций. Сравнение же аналоговых микропроцессоров производится по количеству эквивалентных звеньев аналого-цифровых фильтров рекурсивных фильтров второго порядка. Производительность аналогового микропроцессора определяется его способностью быстро выполнять операции умножения: чем быстрее осуществляется умножение, тем больше эквивалентное количество звеньев фильтра в аналоговом преобразователе и тем более сложный алгоритм преобразования цифровых сигналов можно задавать в микропроцессоре.

Одним из направлений дальнейшего совершенствования аналоговых микропроцессоров является повышение их универсальности и гибкости. Поэтому вместе с повышением скорости обработки большого объема цифровых данных будут развиваться средства обеспечения развитых вычислительных процессов обработки цифровой информации за счет реализации аппаратных блоков прерывания программ и программных переходов.

## Виды процессоров по временной организации работы? Преимущества и недостатки?

По характеру временной организации работы микропроцессоры делят на синхронные и асинхронные.

Синхронные микропроцессоры - микропроцессоры, в которых начало и конец выполнения операций задаются устройством управления (время выполнения операций в этом случае не зависит от вида выполняемых команд и величин операндов).

Асинхронные микропроцессоры позволяют начало выполнения каждой следующей операции определить по сигналу фактического окончания выполнения предыдущей операции. Для более эффективного использования каждого устройства микропроцессорной системы в состав асинхронно работающих устройств вводят электронные цепи, обеспечивающие автономное функционирование устройств. Закончив работу над какой-либо операцией, устройство вырабатывает сигнал запроса, означающий его готовность к выполнению следующей операции. При этом роль естественного распределителя работ принимает на себя память, которая в соответствии с заранее установленным приоритетом выполняет запросы остальных устройств по обеспечению их командной информацией и данными.

## Структура процессора?

По организации структуры микропроцессорных систем различают микроЭВМ одно- и многомагистральные.

В одномагистральных микроЭВМ все устройства имеют одинаковый интерфейс и подключены к единой информационной магистрали, по которой передаются коды данных, адресов и управляющих сигналов.

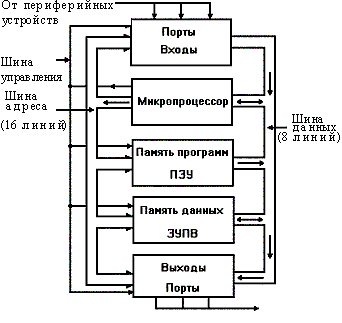
В многомагистральных микроЭВМ устройства группами подключаются к своей информационной магистрали. Это позволяет осуществить одновременную передачу информационных сигналов по нескольким (или всем) магистралям. Такая организация систем усложняет их конструкцию, однако увеличивает производительность.

По количеству выполняемых программ различают одно- и многопрограммные микропроцессоры.

В однопрограммных микропроцессорах выполняется только одна программа. Переход к выполнению другой программы происходит после завершения текущей программы.

В много- или мультипрограммных микропроцессорах одновременно выполняется несколько (обычно несколько десятков) программ. Организация мультипрограммной работы микропроцессорных управляющих систем позволяет осуществить контроль за состоянием и управлением большим числом источников или приемников информации.

Архитектура типичной небольшой вычислительной системы на основе микроЭВМ показана на рис. 2.1 Такая микроЭВМ содержит все 5 основных блоков цифровой машины: устройство ввода информации, управляющее устройство (УУ), арифметико-логическое устройство (АЛУ) (входящие в состав микропроцессора), запоминающие устройства (ЗУ) и устройство вывода информации.

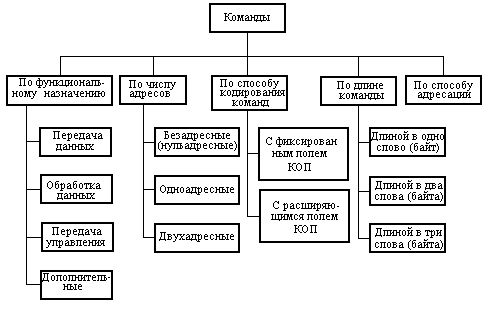


## Типы архитектур процессора? По набору команд? По организации адресного пространства?

Режим адресации памяти - это процедура или схема преобразования адресной информации об операнде в его исполнительный адрес.

Все способы адресации памяти можно разделить на:  
1) прямой, когда исполнительный адрес берется непосредственно из команды или вычисляется с использованием значения, указанного в команде, и содержимого какого-либо регистра (прямая адресация, регистровая, базовая, индексная и т.д.);  
2) косвенный, который предполагает, что в команде содержится значение косвенного адреса, т.е. адреса ячейки памяти, в которой находится окончательный исполнительный адрес (косвенная адресация).

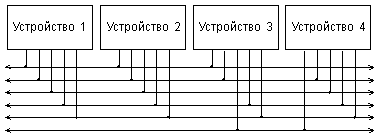
основные типы архитектурных решений, выделяя связь со способами адресации памяти.  
1. Регистровая архитектура определяется наличием достаточно большого регистрового файла внутри МП. Команды получают возможность обратиться к операндам, расположенным в одной из двух запоминающих сред: оперативной памяти или регистрах. Размер регистра обычно фиксирован и совпадает с размером слова, физически реализованного в оперативной памяти. К любому регистру можно обратиться непосредственно, поскольку регистры представлены в виде массива запоминающих элементов - регистрового файла. Типичным является выполнение арифметических операций только в регистре, при этом команда содержит два операнда (оба операнда в регистре или один операнд в регистре, а второй в оперативной памяти).



Мб еще risc cisc

## Особенности шинной структуры связей? Преимущества и недостатки?

При шинной структуре связей все сигналы между устройствами передаются по одним и тем же линиям связи, но в разное время (это называется мультиплексированной передачей). Причем передача по всем линиям связи может осуществляться в обоих направлениях (так называемая двунаправленная передача). В результате количество линий связи существенно сокращается, а правила обмена (протоколы) упрощаются. Группа линий связи, по которым передаются сигналы или коды как раз и называется шиной (англ. bus).



Понятно, что при шинной структуре связей легко осуществляется пересылка всех информационных потоков в нужном направлении, например, их можно пропустить через один процессор, что очень важно для микропроцессорной системы. Однако при шинной структуре связей вся информация передается по линиям связи последовательно во времени, по очереди, что снижает быстродействие системы по сравнению с классической структурой связей.

Большое достоинство шинной структуры связей состоит в том, что все устройства, подключенные к *шине*, должны принимать и передавать информацию *по* одним и тем же правилам (протоколам обмена информацией *по* *шине* ). Соответственно, все узлы, отвечающие за обмен с *шиной* в этих устройствах, должны быть единообразны, унифицированы.

Существенный недостаток шинной структуры связан с тем, что все устройства подключаются к каждой линии связи параллельно. Поэтому любая неисправность любого устройства может вывести из строя всю систему, если она портит линию связи. *По* этой же причине *отладка* системы с шинной структурой связей довольно сложна и обычно требует специального оборудования.

## Что такое системная магистраль? Что в нее входит?

Информационная магистраль ЭВМ – это унифицированная подсистема, обеспечивающая обмен данными между структурными компонентами компьютера. Физически представлена набором разноуровневых проводников, интерфейсов, линий связи. Унификация заключается в том, что устройства одинаково подключены к магистрали, обмениваются информацией по единому протоколу.

В состав главной информационной магистрали входят следующие шины:

* Системная или процессорная (FSB) – применяется микросхемами чипсета для пересылки данных между центральным процессором и видеокартой, оперативной памятью.

* Кэш-памяти – организовывает обмен между кэшем и процессором. Современные ЦП оснащаются встроенной кэш-памятью с целью повышения быстродействия шины.

* Памяти – интерфейс для связи между CPU и оперативной памятью.

* Ввода/вывода – объединяет интерфейсы внешнего оборудования.
* Последний вид шин подразделяют на локальные и стандартные.
* Локальная шина – интерфейс для объединения быстродействующего оборудования (видеоадаптер, сетевая карта) с центральным процессором. Преимущественно это PCI-e.
* Стандартная шина ввода/вывода – интерфейс для подсоединения к прочим шинам медленного оборудования: мышка, клавиатура, звуковое оборудование. Благодаря архитектуре, поддерживает параллельное подключение нескольких внешних устройств.
* Системная магистраль построена по модульному принципу – это организация системы таким образом, что позволяет без затрат подключать к шине и отключать от неё модули, не оказывая отрицательного воздействия на компьютер. Модульный принцип даёт возможность заменять устаревшие и повреждённые комплектующие, расширять функциональность ПК за счёт добавления новых устройств: второй видеокарты, накопителей, планок оперативной памяти. Процессор сам организовывает и согласовывает их взаимодействие.

## Режимы обмена информацией на системной магистрали?

1. При синхронном обмене процессор заканчивает обмен данными самостоятельно, через раз и навсегда установленный временной интервал выдержки (tвыд), то есть без учета интересов устройства-исполнителя; 2. При асинхронном обмене процессор заканчивает обмен только тогда, когда устройство-исполнитель подтверждает выполнение операции специальным сигналом (так называемый режим handshake — рукопожатие).

Достоинства синхронного обмена — более простой протокол обмена, меньшее количество управляющих сигналов. Недостатки — отсутствие гарантии, что исполнитель выполнил требуемую операцию, а также высокие требования к быстродействию исполнителя.

Достоинства асинхронного обмена — более надежная пересылка данных, возможность работы с самыми разными по быстродействию исполнителями. Недостаток — необходимость формирования сигнала подтверждения всеми исполнителями, то есть дополнительные аппаратурные затраты.

## Регистры общего назначения? Как организованы?

Регистры общего назначения физически находятся в микропроцессоре внутри арифметико-логического устройства (АЛУ), поэтому их еще называют регистрами АЛУ.

**eax/ax/ah/al** (Accumulator register) — регистр-аккумулятор.

      Применяется для хранения промежуточных данных.

**ebx/bx/bh/bl** (Base register) — базовый регистр.

       Применяется для хранения базового адреса некоторого объекта в памяти.

**ecx/cx/ch/cl** (Count register) — регистр-счетчик.

       Применяется в командах, производящих некоторые повторяющиеся действия.

Его использование часто неявно и скрыто в алгоритме работы соответствующей команды. Например, команда организации цикла loop кроме передачи управления команде, находящейся по некоторому адресу, анализирует и уменьшает на единицу значение регистра ecx/cx.

**edx/dx/dh/dl** (Data register) — регистр данных.

        Применяется для хранения промежуточных данных.

Обратим внимание, можно использовать только весь регистр целиком (например, eax), младшую 16-битную часть (например, ax) и 8-битные части этих регистров (например, ah, al).

!!! Старшие 16 бит этих регистров как самостоятельные объекты недоступны.

Регистры **esi/si** и **edi/di** используются для поддержки так называемых цепочечных операций, то есть операций, производящих последовательную обработку цепочек элементов (можно сравнить со строками в языке Pascal).

**esi/si** (Source Index register) — индекс источника.

         Этот регистр в цепочечных операциях содержит текущий адрес элемента в цепочке-источнике (первоначальная строка).

**edi/di**(Destination Index register) — индекс приемника (получателя).

         Этот регистр в цепочечных операциях содержит текущий адрес в цепочке-приемнике (результирующая строка).

Регистры **ebp/bp** и **esp/sp** предназначены для работы со стеком.

**esp/sp** (Stack Pointer register) — регистр указателя стека.

          Содержит указатель вершины стека в текущем сегменте стека.

**ebp/bp** (Base Pointer register) — регистр указателя базы кадра стека.

          Регистр предназначен для организации произвольного доступа к данным внутри стека.

## Классификация команд?

При наличии статистических данных можно разработать (выбрать) ЭВМ с эффективной системой команд. Одним из подходов к достижению данной цели является разработка команд длиной в одно слово и кодирование их таким образом, чтобы разряды таких коротких команд использовать оптимально, что позволит сократить время реализации программы и ее длину.

Другим подходом к оптимизации системы команд является использование микроинструкций. В этом случае отдельные биты или группы бит команды используются для кодирования нескольких элементарных операций, которые выполняются в одном командном цикле. Эти элементарные операции не требуют обращения к памяти, а последовательность их реализации определяется аппаратной логикой.

Сокращение времени выполнения программ и емкости памяти достигается за счет увеличения сложности логики управления.

Важной характеристикой команды является ее формат, определяющий структурные элементы команды, каждый из которых интерпретируется определенные образом при ее выполнении. Среди таких элементов (полей) команды выделяют следующие: код операции, определяющий выполняемое действие; адрес ячейки памяти, регистра процессора, внешнего устройства; режим адресации; операнд при использовании непосредственной адресации; код анализируемых признаков для команд условного перехода.

Классификация команд по основным признакам представлена на рис. 2.4. Важнейшим структурным элементом формата любой команды является код операции (КОП), определяющей действие, которое должно быть выполнено. Большое число КОП в процессоре очень важно, так как аппаратная реализация команд экономит память и время. Но при выборе ЭВМ необходимо концентрировать внимание на полноте операций с конкретными типами данных, а не только на числе команд, на доступных режимах адресации. Число бит, отводимое под КОП, является функцией полного набора реализуемых команд.

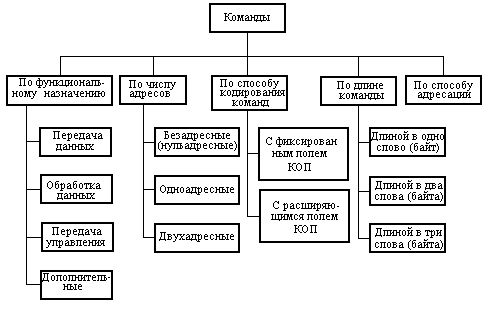


Рис. 2.4. Классификация команд.

При использовании фиксированного числа бит под КОП для кодирования всех m команд необходимо в поле КОП выделить двоичных разрядов. Однако, учитывая ограниченную длину слова мини- и микроЭВМ, различное функциональное назначение команд, источники и приемники результатов операций, а также то, что не все команды содержат адресную часть для обращения к памяти и периферийным устройствам, в малых ЭВМ для кодирования команд широко используется принцип кодирования с переменным числом бит под поле КОП для различных групп команд.

В некоторых командах необходим только один операнд и они называются однооперандными (или одноадресными) командами в отличие от двухоперандных (или двухадресных), в которых требуются два операнда. При наличии двух операндов командой обычно изменяется только один из них. Так как информация берется только из одной ячейки, эту ячейку называются источником; ячейка, содержимое которой изменяется, называется приемником.

Четырехбитный КОП (биты 15-12) кодирует ряд двухоперандных операций, приведенных в таблице 1. Биты (11-6) и (5-0) для команд данного типа определяют адреса источника и приемника данных. Как видно из таблицы, комбинации 0000 и 1000 поля КОП определяют группы одноадресных команд (рис 1,б). КОП 1 (биты 15-12), соответствующий кодам 0000 и 1000, определяет группу одноадресных команд, а КОП 2 (биты 11-6) кодирует конкретную операцию команд данной группы. Таким образом, команды, использующие один операнд, кодируются 10-битным КОП (биты 15-6).

Наиболее гибкая команда требует до четырех операндов. Например, команда сложения может указывать адреса слагаемых, адрес результата и адрес следующей команды. Если для задания адреса требуется 16 бит, то четырехоперандная команда займет 8 байт памяти, не учитывая код операции. Следовательно, получится медленнодействующая ЭВМ с огромной памятью. Поэтому в большинстве микроЭВМ любой команде требуется не более двух операндов. Это достигается следующими приемами:  
1. Адрес следующей команды указывается только в командах переходов; в остальных случаях очередная команда выбирается из ячеек памяти, следующих за выполненной командой.  
2. Использование ячейки, в которой находится один из операндов, для запоминания результата (например, сумма запоминается в ячейки первого операнда).

Локализацию и обращение к операндам обеспечивают режимы адресации. При введении нескольких режимов адресации необходимо отвести в команде биты, указывающие режимы адресации для каждого операнда. Если предусмотрено восемь режимов адресации, то для задания каждого из них нужно три бита.

Почти во всех форматах команд первые биты отводятся для кода операции, но далее форматы команд разных ЭВМ сильно отличаются друг от друга. Остальные биты должны определять операнды или их адреса, и поэтому они используются для комбинации режимов, адресов регистров, адресов памяти, относительных адресов и непосредственных операндов. Обычно длина команды варьируется от 1 до 3 и даже 6 байт.

## Методы адресации?

Для взаимодействия с различными модулями в ЭВМ должны быть средства идентификации ячеек внешней памяти, ячеек внутренней памяти, регистров МП и регистров устройств ввода/вывода. Поэтому каждой из запоминающих ячеек присваивается адрес, т.е. однозначная комбинация бит. Количество бит определяет число идентифицируемых ячеек. Обычно ЭВМ имеет различные адресные пространства памяти и регистров МП, а иногда - отдельные адресные пространства регистров устройств ввода/вывода и внутренней памяти. Кроме того, память хранит как данные, так и команды. Поэтому для ЭВМ разработано множество способов обращения к памяти, называемых режимами адресации.

Режим адресации памяти - это процедура или схема преобразования адресной информации об операнде в его исполнительный адрес.

Все способы адресации памяти можно разделить на:  
1) прямой, когда исполнительный адрес берется непосредственно из команды или вычисляется с использованием значения, указанного в команде, и содержимого какого-либо регистра (прямая адресация, регистровая, базовая, индексная и т.д.);  
2) косвенный, который предполагает, что в команде содержится значение косвенного адреса, т.е. адреса ячейки памяти, в которой находится окончательный исполнительный адрес (косвенная адресация).

В каждой микроЭВМ реализованы только некоторые режимы адресации, использование которых, как правило, определяется архитектурой МП.

## Сегментирование памяти? Для чего нужно?

**Сегментация памяти** представляет собой [систему операционного](https://hmn.wiki/ru/Operating_system)[управления памятью](https://hmn.wiki/ru/Memory_management_(operating_systems)) методика деления [компьютера](https://hmn.wiki/ru/Computer) «S [первичной памяти](https://hmn.wiki/ru/Primary_memory) в **сегменты** или **секции** . В [компьютерной системе,](https://hmn.wiki/ru/Computer_architecture) использующей сегментацию, ссылка на ячейку памяти включает в себя значение, которое идентифицирует сегмент, и [смещение](https://hmn.wiki/ru/Offset_(computer_science)) (ячейку памяти) в этом сегменте. Сегменты или разделы также используются в [объектных файлах](https://hmn.wiki/ru/Object_file) скомпилированных программ, когда они [связаны](https://hmn.wiki/ru/Linker_(computing)) вместе в [образ программы](https://hmn.wiki/ru/Program_image) и когда изображение [загружается](https://hmn.wiki/ru/Loader_(computing)) в память.

Сегменты обычно соответствуют естественным разделам программы, таким как отдельные процедуры или таблицы данных [[1],](https://hmn.wiki/ru/Memory_segmentation#cite_note-holt1961-1) поэтому сегментация обычно более заметна для программиста, чем [разбиение по](https://hmn.wiki/ru/Paging) страницам. [[2]](https://hmn.wiki/ru/Memory_segmentation#cite_note-englander-2) Различные сегменты могут быть созданы для разных программных [модулей](https://hmn.wiki/ru/Module_(programming)) или для разных классов использования памяти, таких как [сегменты](https://hmn.wiki/ru/Data_segment)[кода](https://hmn.wiki/ru/Code_segment) и [данных](https://hmn.wiki/ru/Data_segment) . [[3]](https://hmn.wiki/ru/Memory_segmentation#cite_note-glaser1965-3) Некоторые сегменты могут использоваться программами совместно. [[1]](https://hmn.wiki/ru/Memory_segmentation#cite_note-holt1961-1)[[2]](https://hmn.wiki/ru/Memory_segmentation#cite_note-englander-2)

Первоначально сегментация была изобретена как метод, с помощью которого [системное программное обеспечение](https://hmn.wiki/ru/System_software) могло изолировать различные программные [процессы](https://hmn.wiki/ru/Process_(computing)) ( [задачи](https://hmn.wiki/ru/Task_(computing)) ) и данные, которые они используют. Это было предназначено для повышения надежности систем, одновременно выполняющих несколько процессов. [[4]](https://hmn.wiki/ru/Memory_segmentation#cite_note-:0-4) В [архитектуре x86-64](https://hmn.wiki/ru/X86-64) это считается устаревшим, и большая часть современного системного программного обеспечения на базе x86-64 не использует сегментацию памяти. Вместо этого они обрабатывают программы и их данные, используя [подкачку памяти,](https://hmn.wiki/ru/Paging) которая также служит способом защиты памяти. Однако большинство реализаций x86-64 по-прежнему поддерживают его из соображений обратной совместимости. [[4]](https://hmn.wiki/ru/Memory_segmentation#cite_note-:0-4)

В системе, использующей сегментацию, адреса памяти компьютера состоят из идентификатора сегмента и смещения внутри сегмента. [[3]](https://hmn.wiki/ru/Memory_segmentation#cite_note-glaser1965-3) Блок [управления](https://hmn.wiki/ru/Memory_management_unit) аппаратной [памятью](https://hmn.wiki/ru/Memory_management_unit) (MMU) отвечает за преобразование сегмента и смещения в [физический адрес](https://hmn.wiki/ru/Physical_address) и за выполнение проверок, чтобы убедиться, что преобразование может быть выполнено, и что ссылка на этот сегмент и смещение разрешены.

Каждый сегмент имеет длину и связанный с ним набор разрешений (например, *чтение* , *запись* , *выполнение* ). [[3]](https://hmn.wiki/ru/Memory_segmentation#cite_note-glaser1965-3)[процесс](https://hmn.wiki/ru/Process_(computing)) разрешено только сделать ссылку в сегмент , если тип ссылки разрешено разрешениями, и если смещение внутри сегмента находится в пределах диапазона , определенного по длине сегмента. В противном случае возникает [аппаратное исключение,](https://hmn.wiki/ru/Hardware_exception) например [ошибка сегментации](https://hmn.wiki/ru/Segmentation_fault) .

Сегменты также могут использоваться для реализации [виртуальной памяти](https://hmn.wiki/ru/Virtual_memory) . В этом случае с каждым сегментом связан флаг, указывающий, присутствует он в основной памяти или нет. Если осуществляется доступ к сегменту, которого нет в основной памяти, возникает исключение, и [операционная система](https://hmn.wiki/ru/Operating_system) считывает сегмент в память из вторичного хранилища.

## Архитектура параллельных вычислительных систем? Преимущества и недостатки?

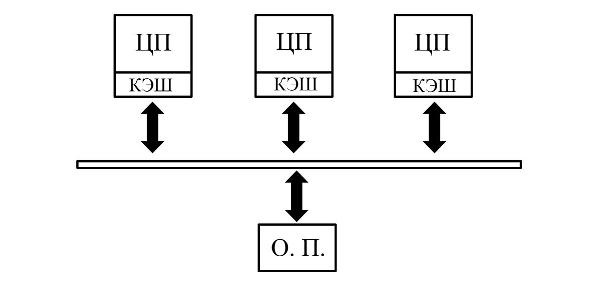
**Параллельные вычислительные системы** — это физические компьютерные, а также программные системы, реализующие тем или иным способом [параллельную обработку данных](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F) на многих вычислительных узлах.[[1]](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B#cite_note-1)

Например, для быстрой [сортировки массива](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2%D0%B0) на двухпроцессорной машине можно разделить массив пополам и сортировать каждую половину на отдельном процессоре. Сортировка каждой половины может занять разное время, поэтому необходима [синхронизация](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)).

*Основная статья:*[***Параллельные вычисления***](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F)

Идея [распараллеливания](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) вычислений основана на том, что большинство задач может быть разделено на набор меньших задач, которые могут быть решены одновременно. Обычно параллельные вычисления требуют координации действий. Параллельные вычисления существуют в нескольких формах: параллелизм на уровне битов, параллелизм на уровне инструкций, параллелизм данных, параллелизм задач. Параллельные вычисления использовались много лет в основном в высокопроизводительных вычислениях, но в последнее время к ним возрос интерес вследствие существования физических ограничений на рост тактовой частоты процессоров. Параллельные вычисления стали доминирующей парадигмой в [архитектуре компьютеров](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%B0), в основном в форме [многоядерных процессоров](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D1%8F%D0%B4%D0%B5%D1%80%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80).[[2]](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B#cite_note-View-Power-2)

Писать программы для параллельных систем сложнее, чем для последовательных[[3]](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B#cite_note-3), так как конкуренция за ресурсы представляет новый класс потенциальных ошибок в программном обеспечении ([багов](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%B0%D1%8F_%D0%BE%D1%88%D0%B8%D0%B1%D0%BA%D0%B0)), среди которых [состояние гонки](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D0%B5_%D0%B3%D0%BE%D0%BD%D0%BA%D0%B8) является самой распространённой. Взаимодействие и [синхронизация](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D0%B2&action=edit&redlink=1) между процессами представляют большой барьер для получения высокой производительности параллельных систем. В последние годы также стали рассматривать вопрос о потреблении электроэнергии параллельными компьютерами.[[4]](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B#cite_note-4) Характер увеличения скорости программы в результате распараллеливания объясняется законами [Амдала](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BA%D0%BE%D0%BD_%D0%90%D0%BC%D0%B4%D0%B0%D0%BB%D0%B0) и [Густавсона](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BA%D0%BE%D0%BD_%D0%93%D1%83%D1%81%D1%82%D0%B0%D0%B2%D1%81%D0%BE%D0%BD%D0%B0_%E2%80%94_%D0%91%D0%B0%D1%80%D1%81%D0%B8%D1%81%D0%B0" \o "Закон Густавсона — Барсиса).



## Классификация параллельных вычислительных систем?

**Параллелизм на уровне битов**[[править](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B&veaction=edit&section=2) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B&action=edit&section=2)]

*Основная статья:****[Bit-level parallelism](https://ru.wikipedia.org/wiki/Bit-level_parallelism" \o "Bit-level parallelism)***

Эта форма параллелизма основана на увеличении размера [машинного слова](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D1%81%D0%BB%D0%BE%D0%B2%D0%BE). Увеличение размера машинного слова уменьшает количество операций, необходимых процессору для выполнения действий над переменными, чей размер превышает размер машинного слова. К примеру: на 8-битном процессоре нужно сложить два 16-битных целых числа. Для этого вначале нужно сложить младшие 8 бит чисел, затем сложить старшие 8 бит и к результату их сложения прибавить значение [флага переноса](https://ru.wikipedia.org/wiki/%D0%A4%D0%BB%D0%B0%D0%B3_%D0%BF%D0%B5%D1%80%D0%B5%D0%BD%D0%BE%D1%81%D0%B0). Итого 3 инструкции. С 16-битным процессором можно выполнить эту операцию одной инструкцией.

Исторически 4-битные микропроцессоры были заменены 8-битными, затем появились 16-битные и 32-битные. 32-битные процессоры долгое время были стандартом в повседневных вычислениях. С появлением технологии [x86-64](https://ru.wikipedia.org/wiki/AMD64) для этих целей стали использовать 64-битные процессоры.

**Параллелизм на уровне инструкций**[[править](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B&veaction=edit&section=3) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B&action=edit&section=3)]

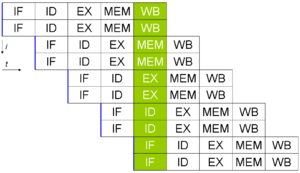
*Основная статья:*[***Параллелизм на уровне команд***](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D0%B8%D0%B7%D0%BC_%D0%BD%D0%B0_%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5_%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4)

Компьютерная программа — это, по существу, поток инструкций, выполняемых процессором. Но можно изменить порядок этих инструкций, распределить их по группам, которые будут выполняться параллельно, без изменения результата работы всей программы. Данный приём известен как параллелизм на уровне инструкций. Продвижения в развитии параллелизма на уровне инструкций в архитектуре компьютеров происходили с середины 1980-х до середины 1990-х.

[](https://commons.wikimedia.org/wiki/File:Fivestagespipeline.png?uselang=ru)

Классический пример пятиступенчатого конвейера на [RISC](https://ru.wikipedia.org/wiki/RISC)-машине (IF = выборка инструкции, ID = декодирование инструкции, EX = выполнение инструкции, MEM = доступ к памяти, WB = запись результата в регистры).

Современные процессоры имеют многоступенчатый [конвейер команд](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BD%D0%B2%D0%B5%D0%B9%D0%B5%D1%80). Каждой ступени конвейера соответствует определённое действие, выполняемое процессором в этой инструкции на этом этапе. Процессор с N ступенями конвейера может иметь одновременно до N различных инструкций на разном уровне законченности. Классический пример процессора с конвейером — это [RISC](https://ru.wikipedia.org/wiki/RISC)-процессор с 5-ю ступенями: выборка инструкции из памяти (IF), декодирование инструкции (ID), выполнение инструкции (EX), доступ к памяти (MEM), запись результата в регистры (WB). Процессор [Pentium 4](https://ru.wikipedia.org/wiki/Pentium_4) имеет конвейер в 31 ступень[[5]](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B#cite_note-5).

[](https://commons.wikimedia.org/wiki/File:Superscalarpipeline.png?uselang=ru)

Пятиступенчатый конвейер суперскалярного процессора, способный выполнять две инструкции за цикл. Может иметь по две инструкции на каждой ступени конвейера, максимум 10 инструкций могут выполняться одновременно.

Некоторые процессоры, дополнительно к использованию конвейеров, обладают возможностью выполнять несколько инструкций одновременно, что даёт дополнительный параллелизм на уровне инструкций. Возможна реализация данного метода при помощи [суперскалярности](https://ru.wikipedia.org/wiki/%D0%A1%D1%83%D0%BF%D0%B5%D1%80%D1%81%D0%BA%D0%B0%D0%BB%D1%8F%D1%80%D0%BD%D0%BE%D1%81%D1%82%D1%8C" \o "Суперскалярность), когда инструкции могут быть сгруппированы вместе для параллельного выполнения (если в них нет [зависимости между данными](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B2%D0%B8%D1%81%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) (завимости по данным)). Также возможны реализации с использованием явного параллелизма на уровне инструкций: [VLIW](https://ru.wikipedia.org/wiki/VLIW) и [EPIC](https://ru.wikipedia.org/wiki/EPIC_(%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80)).

**Параллелизм данных**[[править](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B&veaction=edit&section=4) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B&action=edit&section=4)]

*Основная статья:*[***Векторизация (параллельные вычисления)***](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_(%D0%BF%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F))

Основная идея подхода, основанного на параллелизме данных, заключается в том, что одна операция выполняется сразу над всеми элементами массива данных. Различные фрагменты такого массива обрабатываются на [векторном процессоре](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80) или на разных процессорах параллельной машины. Распределением данных между процессорами занимается программа. Векторизация или распараллеливание в этом случае чаще всего выполняется уже на этапе компиляции — перевода исходного текста программы в машинные команды. Роль программиста в этом случае обычно сводится к заданию настроек векторной или параллельной оптимизации [компилятору](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80), директив параллельной компиляции, использованию специализированных языков для параллельных вычислений.

**Параллелизм задач**[[править](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B&veaction=edit&section=5) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B&action=edit&section=5)]

*Основная статья:*[***Параллелизм на уровне потоков***](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D0%B8%D0%B7%D0%BC_%D0%BD%D0%B0_%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5_%D0%BF%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%B2&action=edit&redlink=1)

Стиль программирования, основанный на параллелизме задач, подразумевает, что вычислительная задача разбивается на несколько относительно самостоятельных подзадач и каждый процессор загружается своей собственной подзадачей.

## Системы с общей и распределенной памятью? Преимущества и недостатки?

## Способы распараллеливания?

## Dataflow архитектура? Преимущества и недостатки?

## Аппаратная реализация Dataflow?

## Статическая Dataflow?

## Динамическая Dataflow с помеченными токенами?

## Динамическая Dataflow с явно адресуемыми токенами?

## Гибридные Dataflow?

## Реконфигурируемые системы? Преимущества и недостатки?