**1)Что такое микропроцессорная система? Особенности традиционной цифровой системы.**

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.

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.

**2) Что такое микропроцессорная система? Особенности системы на гибкой логике?**

В системах гибкой логики изменение условий влияет только на изменение программы, в соответствии с которой работает система. Такая система является программируемой. Для сложных алгоритмов обработки информации.

Характерной особенностью традиционных цифровых систем является тот факт, что алгоритм обработки и хранения информации в ней жёстко связан со схемотехникой системы, то есть для конкретно поставленной задачи разрабатывается и реализуется конкретная электронная схема. Любое изменение исходных условий задачи повлечет за собой и изменение её схемотехнического решения, т.е. изменение алгоритма функционирования системы возможно только путём изменения её структуры. Такие схемы называют схемами жёсткой логики.

Таким образом, любая система жёсткой логики представляет собой специализированную электронную систему, разработанную и настроенную на решение одной или нескольких заранее известных задач.

Преимуществом систем жёсткой логики является их высокое быстродействие, так как такие системы никогда не имеют аппаратной избыточности, а скорость выполнения алгоритмов определяется в ней только быстродействием отдельных логических элементов.

Самым большим недостатком цифровой системы на жёсткой логике является тот факт, что при изменении условий задачи схему нужно проектировать и изготавливать заново.

Преодолеть этот недостаток позволяют электронные системы гибкой логики, которые могут легко адаптироваться под любую за-дачу, перестраиваясь с одного алгоритма на другой без изменения электронной схемы. В таких системах изменение условий влияет только на изменение программы, в соответствии с которой работает система (Рис. 1.2).

Такая система является программируемой (перепрограммируемой). Именно к системе гибкой логики и относятся микропроцессорные системе.

Конечно, аппаратно такая схема может быть избыточна, так как должна функционировать и для решения самой простой, и для решения самой сложной задачи. А решение трудной задачи требует гораздо больше аппаратных средств, чем решение простой.

чем проще решается задача, тем больше избыточность. Такая избыточность ведет, с одной стороны, к увеличению стоимости схемы, увеличению потребляемой мощности, с другой стороны - к существенному уменьшению быстродействия.

Из вышеизложенного можно сделать следующий вывод: системы жесткой логики должны применяться там, где решаемая задача не меняется длительное время, где требуется высокое быстродействие, а алгоритмы обработки информации просты. Системы гибкой логики должны применяться там, где часто меняются решаемые задачи, высокое быстродействие не слишком важно, а алгоритмы обработки информации сложны.

**3) Преимущества и недостатки систем на жесткой и гибкой логике?**

Преимущества жёсткой логики:

* Высокое быстродействие (нет аппаратной избыточности)

Недостатки ж.л.:

* при изменении условий задачи схему нужно проектировать и изготавливать заново.

Преимущества гибкой логики:

* могут легко адаптироваться под любую за-дачу, перестраиваясь с одного алгоритма на другой без изменения электронной схемы

Недостатки г.л.:

* низкое быстродействие

**4) Что такое Большая интегральная схема? Какие бывают процессоры по количеству БИС?**

Большая интегральная схема (БИС) — это полупроводниковый прибор, у которого на одной пластине из полупроводникового материала площадью в несколько квадратных миллиметров размещается до миллиона и более компонентов: диодов, транзисторов, резисторов. конденсаторов, соединенных между собой некоторым наперед заданным образом.

По числу больших интегральных схем (БИС) в микропроцессорном комплекте различают микропроцессоры однокристальные, многокристальные и многокристальные секционные. Процессоры даже самых простых ЭВМ имеют сложную функциональную структуру, содержат большое количество электронных элементов и множество разветвленных связей.

**5)Виды процессоров по назначению? Преимущества и недостатки?**

- **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.

**6)Виды процессоров по виду обрабатываемых сигналов? Преимущества и недостатки?**

- **Цифровые** – т.е. работающие с числовыми данными.

- **Аналоговые**– предназначены для обработки аналоговых сигналов и имеющие в качестве входных и выходных данных аналоговые сигналы. По сути, все современные аналоговые МП являются цифровыми сигнальными МП, имеющими на входе встроенные аналого-цифровые преобразователи (АЦП), а на выходе – встроенные цифро-аналоговые преобразователи (ЦАП).

**7)Виды процессоров по временной организации работы? Преимущества и недостатки?**

- **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.

**8)Структура процессора?**

Центральный процессор состоит из 3-х частей:

1. Ядро процессора, которое выполняет основную работу. Оно позволяет читать, расшифровывать, выполнять и отправлять инструкции. Ядро состоит из следующих частей:

* Арифметико-логическое устройство (АЛУ). Выполняет основные математические и логические операции. Все вычисления производятся в двоичной системе.
* Устройство управления (УУ). Управляет работой CPU с помощью электрических сигналов. От него зависит согласованность работы всех частей процессора и его связь с внешними устройствами.

Каждое ядро может выполнять только одну задачу, хоть и за долю секунды. Одноядерный процессор выполняет каждую задачу последовательно. Для современного объёма операций этого мало, поэтому ценятся CPU с более чем одним ядром, чтобы выполнять несколько задач одновременно. Например, двухъядерный выполняет две задачи одновременно, трехъядерный ― три и т. д.

1. Запоминающее устройство. Это небольшая внутренняя память центрального процессора. Она состоит из регистров и кеш-памяти. В регистрах хранятся текущие команды, данные, промежуточные результаты операции. В кеш-память загружаются часто используемые команды и данные из оперативной памяти. Обратиться в кеш быстрее, чем в оперативную память, поэтому объём кеш-памяти влияет на скорость выполнения запросов.
2. Шины ― это каналы, по которым передаётся информация. Они как рельсы для перевозки данных.

Главной характеристикой процессора является производительность. Она зависит от двух параметров: **тактовая частота**и**разрядность.**

**Тактовая частота** ― число выполненных операций в секунду. Измеряется в мегагерцах (МГц — миллион тактов в секунду ) и гигагерцах (ГГц — миллиард тактов в секунду). Чем больше тактовая частота, тем быстрее работает машина.

**Разрядность** ― количество информации (байт), которое можно передать за такт. Разрядность процессора бывает 8, 16, 32, 64 бита. Современные процессоры 32-х и 64-битные.

**9)Типы архитектур процессора? По набору команд? По организации адресного пространства?**

Все микропроцессоры можно разделить на следующие группы:

МП с гарвардской архитектурой;

МП с фоннеймановской архитектурой;

МП типа CISC (Complex Instruction Set Computing) с полным набором команд;

МП типа RISC (Reduced Instruction Set Computing) с сокращенным набором команд;

МП типа MISC (Minimum Instruction Set Computing) с минимальным набором команд и весьма высоким быстродействием (в настоящее время эти модели находятся в стадии разработки).

**10)Особенности шинной структуры связей? Преимущества и недостатки?**

Суть шинной структуры связей сводится к следующему. 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.

**Преимущества шинной топологий.**  
1. она надежно работает в сетях, проста в использование, и понятна.   
2. шина требует меньше кабеля чем в других топологиях, следовательно, она дешевле.   
3. она легка масштабирована.   
4. для расширения шинной топологии можно использовать повторитель (Repeater) усиливает сигналы и позволяет передавать его но большие расстояния.  
**Недостатки шинной топологии.**  
1. при большом кол-ве ПК, мощность передачи инф. значительно снижается.   
2. трудность диагностики такой сети.

**11)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.

Для перемещения информации недостаточно объединить устройства проводниками электрических сигналов. Их передача должна соответствовать определённым правилам – протоколу. Он определяет скорость передачи, приоритетность разных типов информации и задач, её адресацию, отвечает за целостность.

Системная шина (магистраль) включает в себя шину данных, адреса и управления. По каждой их них передается своя информация: по шине данных - данные, адреса - соответственно, адрес (устройств и ячеек памяти), управления - управляющие сигналы для устройств.

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.

Логически системная магистраль представлена тремя уровнями.

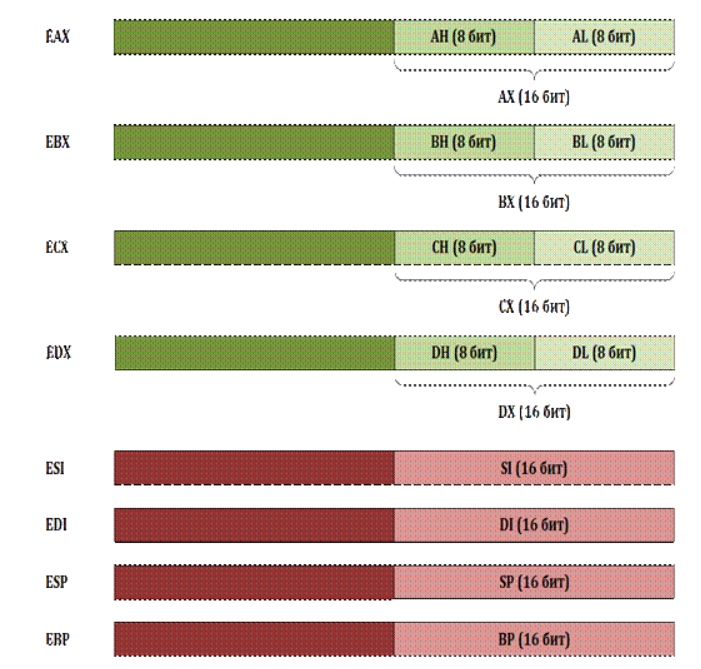
**12)Режимы обмена информацией на системной магистрали?**

Режимы работы МПС Практически любая развитая МПС (в том числе и компьютер) поддерживает три основных режима обмена по магистрали: программный обмен информацией; обмен с использованием прерываний (Interrupts); обмен с использованием прямого доступа к памяти (ПДП, DMA — Direct Memory Access).

**13)Регистры общего назначения? Как организованы?**

Регистры – это специальные ячейки памяти, расположенные непосредственно в процессоре. Работа с регистрами выполняется намного быстрее, чем с ячейками оперативной памяти, поэтому регистры активно используются как в программах на языке ассемблера, так и компиляторами языков высокого уровня.

К регистрам общего назначения относится группа из 8 регистров, которые можно использовать в программе на языке ассемблера. Все регистры имеют размер 32 бита и могут быть разделены на 2 или более частей.



Как видно из рисунка, регистры ESI, EDI, ESP и EBP позволяют обращаться к младшим 16 битам по именам SI, DI, SP и BP соответственно, а регистры EAX, EBX, ECX и EDX позволяют обращаться как к младшим 16 битам (по именам AX, BX, CX и DX), так и к двум младшим байтам по отдельности (по именам AH/AL, BH/BL, CH/CL и DH/DL).

Названия регистров происходят от их назначения:

* EAX/AX/AH/AL (accumulator register) – аккумулятор;
* EBX/BX/BH/BL (base register) –регистр базы;
* ECX/CX/CH/CL (counter register) – счётчик;
* EDX/DX/DH/DL (data register) – регистр данных;
* ESI/SI (source index register) – индекс источника;
* EDI/DI (destination index register) – индекс приёмника (получателя);
* ESP/SP (stack pointer register) – регистр указателя стека;
* EBP/BP (base pointer register) – регистр указателя базы кадра стека.

**14)Классификация команд?**

В общем случае система команд процессора включает в себя следующие четыре основные группы команд:

* команды пересылки данных;
* арифметические команды ;
* логические команды ;
* команды переходов.

Команды пересылки данных не требуют выполнения никаких операций над операндами. Операнды просто пересылаются (точнее, копируются) из источника (Source) в приемник (Destination). Источником и приемником могут быть внутренние регистры процессора, ячейки памяти или устройства ввода/вывода. АЛУ в данном случае не используется.

Арифметические команды выполняют операции сложения, вычитания, умножения, деления, увеличения на единицу (инкрементирования), уменьшения на единицу (декрементирования) и т.д. Этим командам требуется один или два входных операнда. Формируют команды один выходной операнд.

Логические команды производят над операндами логические операции, например, логическое И, логическое ИЛИ, исключающее ИЛИ, очистку, инверсию, разнообразные сдвиги (вправо, влево, арифметический сдвиг, циклический сдвиг). Этим командам, как и арифметическим, требуется один или два входных операнда, и формируют они один выходной операнд.

Наконец, команды переходов предназначены для изменения обычного порядка последовательного выполнения команд. С их помощью организуются переходы на подпрограммы и возвраты из них, всевозможные циклы, ветвления программ, пропуски фрагментов программ и т.д. Команды переходов всегда меняют содержимое счетчика команд. Переходы могут быть условными и безусловными. Именно эти команды позволяют строить сложные алгоритмы обработки информации.

В соответствии с результатом каждой выполненной команды устанавливаются или очищаются биты регистра состояния процессора ( PSW ). Но надо помнить, что не все команды изменяют все имеющиеся в PSW флаги. Это определяется особенностями каждого конкретного процессора.

**15)Методы адресации?**

Под методом адресации понимают способ, посредством которого команда указывает микропроцессору источники обрабатываемых кодов – операндов и приемник полученного результата. Наибольшее значение методы адресации имеют для команд пересылки и арифметико–логической обработки. Поэтому далее вопрос излагается применительно только к этим группам команд.

В командах учебного микропроцессора используется пять методов адресации: неявный, регистровый, косвенный, прямой и непосредственный. Одна команда может использовать несколько методов адресации (например два для адресации операндов и еще один для адресации результата).

Команды учебного микропроцессора, использующие неявную, регистровую, косвенную адресацию, а также любое их сочетание всегда имеют однобайтовый формат. Если же в числе других используется непосредственная или прямая адресация, то соответствующая команда имеет двух– или трехбайтовый формат, в зависимости от разрядности используемого адреса или данных.

Под неявной адресацией понимают жесткую фиксацию регистров СОЗУ – источников операндов или приемников результата, принятую в ряде команд микропроцессора. Признаком такой адресации является отсутствие у программиста каких–либо альтернатив в выборе соответствующего источника или приемника кода. Так, например, все команды сдвига оперируют только с содержимым аккумулятора А. Во всех операциях с двумя байтовыми операндами аккумулятор является как источником одного из операндов, так и приемником результата.

При регистровой адресации источником операнда или приемником результата является один из регистров микропроцессора или регистровая пара. Возможность указать, какой именно из некоторого множества регистров следует использовать в конкретном случае, здесь предоставлена программисту. Имена используемых регистров обязательно фигурируют в буквенной части мнемоники соответствующей команды.

Неявная и регистровая адресация оперируют только с регистрами микропроцессора и не позволяют обращаться к ячейкам памяти ОЗУ и ПЗУ. Для такого обращения нужен адрес соответствующей ячейки.

Косвенная адресация рассматривает в качестве адреса ячейки шестнадцатеричный двоичный код, содержащийся в какой–либо регистровой паре или указателе стека SP. Особенно часто в качестве регистра косвенной адресации в учебном микропроцессоре используется регистровая пара HL.

При прямой адресации адрес источника или приемника кода в памяти или порте ввода–вывода указывается в цифровой части мнемоники в виде двухбайтового кода ADR или однобайтового – N.

Иногда оказывается удобнее указать в команде не адрес операнда, а сам операнд в виде однобайтового слова данных D8 или двухбайтового – D16. В таком случае говорят о непосредственной адресации соответствующего операнда.

**16)Сегментирование памяти? Для чего нужно?**

Сегментация памяти - это метод управления памятью [операционной системы](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Operating_system) , позволяющий разделить [основную память](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Computer)[компьютера](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Primary_memory) на сегменты или секции. В [компьютерной системе](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Computer_architecture), использующей сегментацию, ссылка на ячейку памяти включает значение, которое идентифицирует сегмент и [смещение](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Offset_(computer_science)) (ячейку памяти) внутри этого сегмента. Сегменты или разделы также используются в [объектных файлах](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Object_file) скомпилированных программ, когда они [связаны](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Linker_(computing)) вместе в [образ программы](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Program_image) и когда образ [загружается](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Loader_(computing)) в память.

Сегменты обычно соответствуют естественным подразделениям программы, таким как отдельные подпрограммы или таблицы данных[[1]](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Segmented_memory#cite_note-holt1961-1), поэтому сегментация обычно более заметна программисту, чем только [подкачка](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Paging).[[2]](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Segmented_memory#cite_note-englander-2) Различные сегменты могут быть созданы для разных программных [модулей](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Module_(programming)) или для разных классов использования памяти, таких как [сегменты](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Code_segment)кода[и данных](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Data_segment).[[3]](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Segmented_memory#cite_note-glaser1965-3) Некоторые сегменты могут быть разделены между программами.[[1]](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Segmented_memory#cite_note-holt1961-1)[[2]](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Segmented_memory#cite_note-englander-2)

Первоначально сегментация была изобретена как метод, с помощью которого [системное программное](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/System_software) обеспечение могло изолировать различные программные [процессы](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Process_(computing)) ([задачи](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Task_(computing))) и данные, которые они используют. Она была предназначена для повышения надежности систем, выполняющих несколько процессов одновременно.[[4]](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Segmented_memory#cite_note-:0-4) В [архитектуре x86-64](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/X86-64) она считается устаревшей, и большинство современных системных программ на базе x86-64 не используют сегментацию памяти. Вместо этого они обрабатывают программы и их данные, используя [подкачку памяти](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Paging), которая также служит способом защиты памяти. Однако большинство реализаций x86-64 все еще поддерживают его по причинам обратной совместимости.[[4]](https://translated.turbopages.org/proxy_u/en-ru.ru.d6422ed6-628e3890-14c69700-74722d776562/https/en.wikipedia.org/wiki/Segmented_memory#cite_note-:0-4)

**17)Архитектура параллельных вычислительных систем? Преимущества и недостатки?**

Применение параллельных вычислительных систем (ПВС) и суперкомпьютеров является стратегическим направлением развития вычислительной техники. Если один компьютер не справляется с решением задачи за нужное время, то попробуем взять два, три, десять компьютеров и заставим их одновременно работать над различными частями общей задачи, надеясь получить соответствующее ускорение.

Объединение компьютеров в единую систему повлекло за собой множество следствий. Чтобы обеспечить отдельные компьютеры работой, необходимо исходную задачу разделить на фрагменты, которые можно выполнять независимо друг от друга. Так стали возникать специальные численные методы, допускающие возможность подобного разделения. Чтобы описать способ одновременного выполнения разных фрагментов задачи на разных компьютерах, потребовались специальные языки программирования, специальные операционные системы и т. д. Постепенно такие слова, как «одновременный», «независимый» и похожие на них стали заменяться одним словом «параллельный». Всё это синонимы, если иметь в виду описание каких-то процессов, действий, фактов, состояний, не связанных друг с другом. Ничего другого слова «параллелизм» и «параллельный» в областях, относящихся к компьютерам, не означают.

Основной недостаток заключается в том, что при каждом обращении к памяти затрачивается много времени на ожидание кругового путешествия по сети, поэтому каждый процессор должен обеспечить высокую пропускную способность.

**18)Классификация параллельных вычислительных систем?**

Основные классы параллельных ВС  
  
Основной характеристикой при классификации [3] параллельных вычислительных систем является способ организации памяти :  
  
общая память - все процессора работают в едином адресном пространстве с равноправным доступом к памяти  
распределенная память - каждый процессор имеет собственную локальную памятью, и прямой доступ к этой памяти других процессоров невозможен.  
  
Можно выделить следующие классы параллельных систем :  
  
SMP - симметричные мультипроцессорные системы. SMP  
обычно состоит из нескольких одинаковых процессоров и массива общей памяти. Все процессоры имеют доступ к любой точке памяти с одинаковой скоростью. Наличие общей памяти упрощает взаимодействие процессоров между собой, однако накладывает сильные ограничения на их число (обычно не более 32).  
  
PVP - параллельные векторные системы.  
( NEC SX-4/SX-5, CRAY J90/T90)  
Основным признаком PVP-систем является наличие специальных векторно-конвейерных процессоров. Как правило, несколько таких процессоров работают одновременно над общей памятью (аналогично SMP) в рамках многопроцессорных конфигураций.  
  
MPP - системы массового параллелизма.  
(МВС-1000, IBM RS/6000 SP2, SGI/CRAY T3E)  
MPP состоит из нескольких однородных вычислительных узлов; Каждый такой узел имеет свою локальную память (прямой доступ к памяти других узлов невозможен), один или несколько центральных процессоров( иногда - жесткий диск). Узлы обычно связаны специальной высокоскоростной сетью. Общее число процессоров в таких системах может достигать нескольких тысяч.  
  
NUMA - системы с неоднородным доступом к памяти.  
(SGI Origin2000, Sun HPC 10000)  
NUMA (Non Uniform Memory Access) представляет собой нечто среднее между SMP и MPP. В NUMA память физически распределена, но логически общедоступна. Масштабируемость NUMA-систем ограничивается объемом адресного пространства и возможностями операционной системы.  
  
Кластеры - дешевый вариант MPP.  
( Avalon, Beowulf)  
Обычно это сеть из серийных PC или рабочих станций общего назначения, которая объединяется в ''виртуальную многопроцессорную машину''. Для связи узлов используется одна из стандартных сетевых технологий (Ethernet, Myrinet).

**19)Системы с общей и распределенной памятью? Преимущества и недостатки?**

Системами с общей памятью называют системы, в которых несколько процессоров имеют общую оперативную память. Чаще всего встречающиеся системы этого типа — компьютеры с многоядерными процессорами (multi-core). Преимущества таких систем:· Не требуется обмена данными: данные, помещённые в память одним процессором, автоматически становятся доступными другим процессорам. Соответственно, система не должна тратить время на пересылку данных.

Системы с общей (разделяемой) оперативной памятью образуют современный класс ВС — многопроцессорных супер-ЭВМ. Одинаковый доступ всех процессоров к программам и данным представляет широкие возможности организации параллельного вычислительного процесса (параллельных вычислений). Отсутствуют потери реальной производительности на межпроцессорный (между задачами, процессами и т.д.) обмен данными

(Вычислительная система с распределенной памятью (distributed memory) – система, в которой каждый процессор обладает собственной локальной памятью, а общая память отсутствует. Обмен информацией между составляющими системы обеспечивается с помощью коммуникационной сети посредством обмена сообщениями. Часто такие системы объединяют отдельные ВМ. Данный вид ВС называют слабо связанными (loosely coupled systems).

Системы с распределенной памятью образуют вычислительные комплексы (ВК) — коллективы ЭВМ с межмашинным обменом для совместного решения задач ([рис. 1.5б](https://intuit.ru/studies/courses/80/80/lecture/2431?page=2#image.1.5)). В ВК объединяются вычислительные средства систем управления, решающие специальные наборы задач, взаимосвязанных по данным. Принято говорить, что такие ВК выполняют распределенные вычисления, а сами ВК называют распределенными ВК.

Другое, противоположное воплощение принципа МИМД — масспроцессорные или высокопараллельные архитектуры, объединяющие сотни — тысячи — десятки тысяч процессоров.

В современных супер-ЭВМ наметилась тенденция объединения двух принципов: общей (распределяемой) и распределенной (локальной) оперативной памяти (ЛОП). Такая структура используется в проекте МВК "Эльбрус-3" и "Эльбрус-3М"

**20)Способы распараллеливания?**

Первый способ — представление алгоритма задачи в виде частично-упорядоченной последовательности выполняемых работ. Затем в результате диспетчирования реализуется оптимальный план выполнения работ в ВС при ограничениях на время выполнения всего алгоритма или за минимальное время.

Основой является представление алгоритма граф-схемой G, отражающей информационные связи между работами (задачами, процессами, процедурами, операторами, макрокомандами и т.д.), на которые разбит алгоритм. Граф G — взвешенный, ориентированный, без контуров.

Второй способ распараллеливания — по информации — используется тогда, когда можно распределить обрабатываемую информацию между процессорами для обработки по идентичным алгоритмам (по одному алгоритму).

**21)Dataflow архитектура? Преимущества и недостатки?**

В архитектурах с управлением потоком данных (Dataflow) [[01]](https://habr.com/ru/post/122479/#ref-01) отсутствует понятие «последовательность инструкций», нет Instruction Pointer'а, отсутствует даже адресуемая память в привычном нам смысле. Программа в потоковой системе — это не набор команд, а вычислительный граф. Каждый узел графа представляют собой оператор или набор операторов, а ветви отражают зависимости узлов по данным. Очередной узел начинает выполняеться как только доступны все его входные данные. В этом состоит один из основных принципов dataflow: исполнение инструкций по готовности данных.

Одним из основных достоинств dataflow-архитектуры является ее масштабируемость: не составляет труда собрать систему, содержащую множество устройств сопоставления и исполнительных устройств. Устройства объедиянются простейшим коммутатором, причем для адресации токенов служат их метки. Весь диапазон номеров узлов просто распределяется равномерно между устройствами. Никаких дополнительных мер для синхронизации вычислительного процесса, в отличие от многопроцессорной controlflow-архитектуры, не требуется.

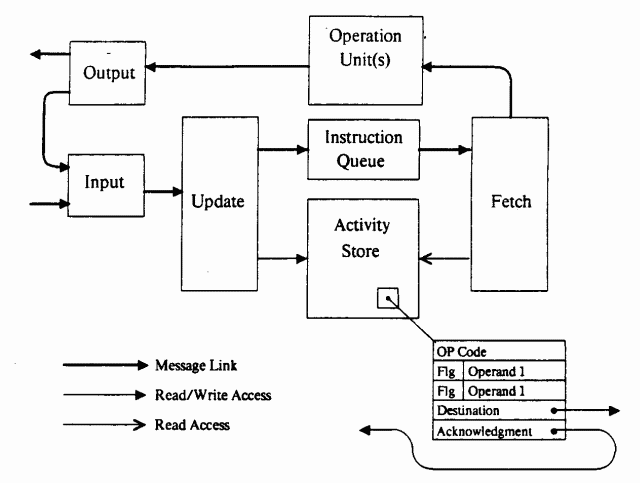
**22)Аппаратная реализация Dataflow?**

В потоковых машинах данные передаются и хранятся в виде т.н. токенов (token). Токен — это структура, содержащая собственно передаваемое значение и метку — указатель узла назначения. Простейшая потоковая вычислительная система состоит из двух устройств: исполнительного (execution unit) и устройства сопоставления (matching unit)

Исполнительное устройство служит для выполнения инструкций и формирования токенов с результатами операций. Как правило, оно включает в себя память команд, доступную только для чтения. Готовность входных данных узла определяется по наличию набора токенов с одинаковыми метками. Для поиска таких наборов и служит устройство сопоставления. Обычно оно реализуется на базе ассоциативной памяти. Используется либо «настоящяя», аппаратная ассоциативная память ([CAM](http://en.wikipedia.org/wiki/Content-addressable_memory) — content-addressable memory), либо структуры, работающие аналогично, например, [хэш-таблицы](http://en.wikipedia.org/wiki/Hash_table).

**23)Статическая Dataflow?**

Описанная выше схема называется статической (static dataflow). В ней каждый вычислительный узел представлен в единственном экземпляре, число узлов заранее известно, также заранее известно число токенов, циркулирующих в системе. В качестве примера реализации статической архитектуры можно привести MIT Static Dataflow Machine [[12]](https://habr.com/ru/post/122479/#ref-12) — потоковый компьютер, созданный в Массачусетском технологическом институте в 1974 году. Машина состояла из множества обрабатывающих элементов (Processing Element), связанных коммуникационной сетью. Схема одного элемента показана на рисунке:



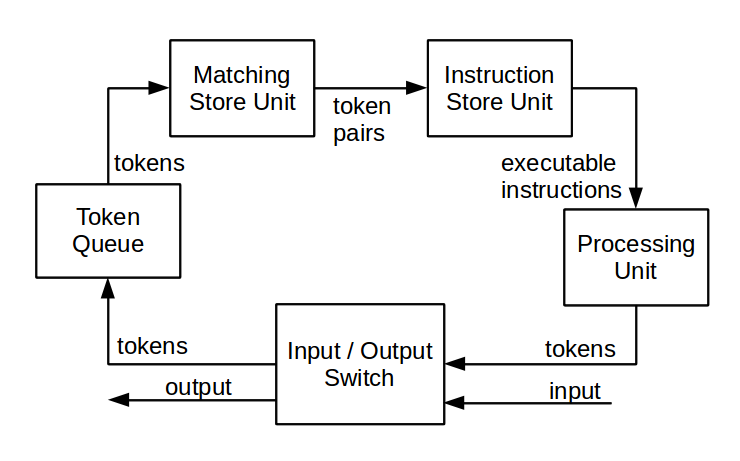
Роль устройства сопоставления здесь выполняла память взаимодействий (activity store). В ней хранились пары токенов вместе с адресом узла назначения, флагами готовности и кодом операции. Любой вычислительный узел в этой архитектуре имел только два входа и состоял из одного оператора. При обнаружении готовности обоих операндов устройство выборки (fetch unit) считывало код операции, и данные отправлялись на обработку в исполнительное устройство (operation unit).

**24)Динамическая Dataflow с помеченными токенами?**

В динамической потоковой архитектуре (dynamic dataflow) каждый узел может иметь множество экземпляров. Для того, чтобы различать токены, адресованные в разные экземпляры одного узла, в структуру токена вводится дополнительное поле — контекст. Сопоставление токенов теперь ведется не только по меткам, но и по значениям контекста. По сравнению со статической архитектурой появляется целый ряд новых возможностей.

* **Рекурсия.** Узел может направлять данные в свою копию, которая будет отличаться контекстом (но при этом иметь ту же метку).
* **Поддержка процедур.** Процедурой в рамках данной модели вычислений будет последовательность узлов, связанных между собой и имеющая входы и выходы. Можно одновременно вызывать несколько экземпляров одной и той же процедуры, которые будут отличаться контекстом.
* **Распараллеливание циклов.** Если между итерациями цикла нет зависимости по данным, можно обрабатывать сразу все итерации одновременно. Номер итерации, как вы уже наверное догадались, будет содержаться в поле контекста.

Одной из первых реализаций динамической потоковой архитектуры была система Manchester Dataflow Machine (1980 год) [[13]](https://habr.com/ru/post/122479/#ref-13). Машина содержала аппаратные средства для организации рекурсий, вызова процедур, раскрытия циклов, копирования и объединения ветвей вычислительного графа. Также в отдельный модуль была вынесена память команд (instruction store unit). На рисунке показана схема одного элемента машины:



Динамическая dataflow-архитектура, по сравнению со статической, демонстрирует более высокую производительность, за счет лучшего параллелизма вычислений. Кроме того, она дает больше возможностей для программиста. С другой стороны, динамическая система сложнее по аппаратной реализации, особенно это касается устройств сопоставления и блоков формирования контекста токенов.

**25)Динамическая Dataflow с явно адресуемыми токенами?**

**26)Гибридные Dataflow?**

«Чистые» потоковые архитектуры, подобные описанным MIT Static Dataflow Machine и Manchester Dataflow Machine, к сожалению, имели много слабых мест:

* Dataflow-машины давали огромные возможности для параллелизма выполнения. Обратной стороной этого преимущества было то, что на последовательных участках вычислительного графа они показывали резкое падение производительности.
* Загрузка исполнительных устройств была далека от максимально возможной. Б*о*льшая часть машинного времени тратилась на поиск соответствия операндов, выборку инструкций, а исполнительное устройство все это время простаивало, выполняя лишь по одной инструкции на каждую пару токенов.
* Трудным было конструирование устройств сопоставления. Ассоциативная память сложнее, дороже, медленнее, занимает больше места и потребляет больше энергии, по сравнению с обычной оперативной памятью такого же объема.
* Сам принцип управления потоком данных не позволял организовать эффективный конвейер. Почти все устройства работали асинхронно, требовались буферы и очереди в линиях связи.
* По сравнению с классической многопроцессорной архитектурой, в dataflow-машинах значительно выше была нагрузка на коммутационную сеть. Ведь фактически, каждая операция требовала пересылки двух токенов.

В попытках решить перечисленные проблемы стали появляться гибридные архитектуры, сочетающие в себе элементы как архитектур потока данных, так и потока управления.

**27)Реконфигурируемые системы? Преимущества и недостатки?**

Реконфигурируемые вычислительные системы (РВС) — это системы, имеющие возможность менять свою модель вычислений, или, иначе говоря, позволяющие вносить существенные изменения в свою аппаратную часть.

Развитие технологий ПЛИС сделало возможным принципиально новый подход к архитектуре dataflow. Что, если собрать машину, ориентированную на решение одной конкретной задачи? Если реализовать непосредственно на схемотехническом уровне нужный вычислительный граф, можно добиться потрясающих результатов. Вместо устройств сложных и медленных сопоставления можно использовать безусловное перенаправление данных от одного функционального модуля к другому. Сами исполнительные устройства тоже можно «заточить» под нужную задачу: выбрать тип арифметики, разрядность, нужный набор поддерживаемых операций.  
Разумеется, подобная машина будет очень узкоспециализированной, но ведь достоинство ПЛИС как раз в возможности неоднократного перепрограммирования. Таким образом, под каждую отдельную задачу собирается нужная архитектура. Некоторые системы позволяют даже осуществлять перенастройку прямо в процессе работы. Реконфигурируемые системы на базе FPGA-микросхем в настоящее время выпускаются серийно в самых разных форматах — от блока «ускорителя» для ПК до системы производительностью порядка нескольких Тфлопс [[31]](https://habr.com/ru/post/122530/#ref-31).  
Из недостатков реконфигурируемых архитектур можно выделить следующие:

* Принципиальная однозадачность. Для запуска новой задачи требуется остановка системы и перепрограммирование ПЛИС, входящих в ее состав.
* Сложность программирования. Программирование каждой задачи включает в себя синтез всей вычислительной архитектуры под данную задачу.
* Избыточная аппаратная сложность. Обратной стороной гибкости ПЛИС является наличие на кристалле большого процента элементов, которые непосредственно не участвуют в вычислениях, а служат только для реконфигурации. Тем не менее, эти элементы потребляют энергию и выделяют тепло во время работы, что ухудшает энергетические показатели эффективности системы (Гфлопс/Вт).

 Одним из недостатков крупнозернистых архитектур является то, что они имеют тенденцию терять часть своего использования и производительности, если им нужно выполнять меньшие вычисления, чем обеспечивает их гранулярность, например, для добавления одного бита к функциональному блоку шириной четыре бита будет потрачено три бита.