**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
* одиночные цифровые сигналы,
* цифровые коды,
* последовательности цифровых кодов.

Характерная особенность: алгоритмы обработки и хранения информации в ней жестко связаны со схемотехникой системы, т.е изменение этих алгоритмов возможно только путем изменения структуры системы.

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

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 -- высокое быстродействие, скорость выполнения алгоритмов определяется в ней только быстродействием отдельных логических элементов.

Самым большим недостатком цифровой системы на жёсткой логике -- при изменении условий задачи схему нужно проектировать и изготавливать заново.

Преодолеть этот недостаток позволяют электронные системы гибкой логики, которые могут легко адаптироваться под любую задачу, перестраиваясь с одного алгоритма на другой без изменения электронной схемы.

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

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?**

См. 2

**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.

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

1. **Универсальные.**

 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.

2. Специализированные.

Среди **специализированных микропроцессоров (МП)** можно выделить:

а) микроконтроллеры, ориентированные на выполнение сложных последовательностей логических операций.

б) математические МП (предназначены для повышения производительности при выполнении арифметических операций).

в) МП для обработки данных в различных областях применений.

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

- **Цифровые** – работают с числовыми данными. У цифровых преимуществ и недостатков нет.

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

Преимущество аналогового: дает достаточно точный сигнал. Недостаток – аналоговый сигнал трудно распознавать/принимать.

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

Делятся на:

а) синхронные (начало и конец выполнения операций задаются устройством управления)

б) асинхронные (начало выполнения каждой следующей операции определить по сигналу фактического окончания выполнения предыдущей операции, роль естественного распределителя работ принимает на себя память)

**8)Структура процессора?**

Центральный процессор состоит из 3-х частей:

1. Ядро процессора.

-читает, расшифровывает, выполняет и отправляет инструкции.

Состоит из:

* Арифметико-логическое устройство (АЛУ). Выполняет основные математические и логические операции.
* Устройство управления (УУ). Управляет работой процессора с помощью электрических сигналов.

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.

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.

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. 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?**

При шинной структуре связей все сигналы между устройствами передаются по одним и тем же линиям связи, но в разное время.

Причем передача по всем линиям связи может осуществляться в обоих направлениях. В результате количество линий связи существенно сокращается, а правила обмена упрощаются.

**Преимущества шинной топологий.**  
1. она надежно работает в сетях, проста в использовании 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 -- подсистема, обеспечивающая обмен данными между структурными компонентами компьютера. Физически представлена набором разноуровневых проводников, интерфейсов, линий связи.

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.

Стандартная шина ввода/вывода – интерфейс для подсоединения к прочим шинам медленного оборудования: мышка, клавиатура, звуковое оборудование.

Системная магистраль построена по модульному принципу. Модульный принцип даёт возможность заменять устаревшие и повреждённые комплектующие, расширять функциональность ПК за счёт добавления новых устройств. Логически системная магистраль представлена тремя уровнями.

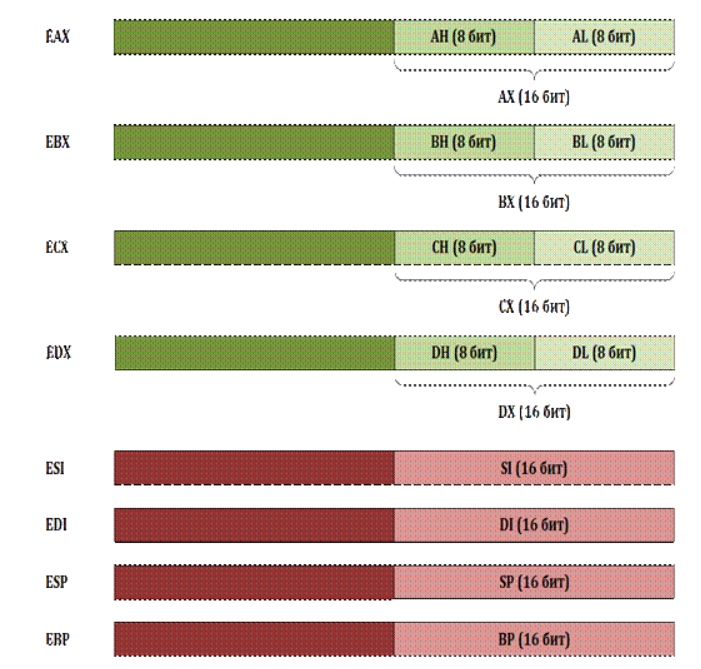
**12)Режимы обмена информацией на системной магистрали?**

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, уменьшения на единицу. Этим командам требуется один или два входных операнда.

Логические команды производят над операндами логические операции, например, логическое И, логическое ИЛИ, исключающее ИЛИ, очистку, инверсию, разнообразные сдвиги (вправо, влево, арифметический сдвиг, циклический сдвиг).

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. Переходы могут быть условными и безусловными. Именно эти команды позволяют строить сложные алгоритмы обработки информации.

**15)Методы адресации?**

Количество методов адресации в различных процессорах может быть от 4 до 16.

а) Непосредственная адресация предполагает, что операнд (входной) находится в памяти непосредственно за кодом команды. Операнд обычно представляет собой константу, которую надо куда-то переслать, к чему-то прибавить и т.д.

б) Прямая (она же абсолютная) адресация предполагает, что операнд находится в памяти по адресу, код которого находится внутри программы сразу же за кодом команды.

в) Регистровая адресация предполагает, что операнд находится во внутреннем регистре процессора.

г) Косвенная адресация предполагает, что во внутреннем регистре процессора находится не сам операнд, а его адрес в памяти.

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

Сегментация памяти - это метод, 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. Сегменты или разделы 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, 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). 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). Некоторые сегменты могут быть разделены между программами.

Первоначально сегментация была предназначена для повышения надежности систем, выполняющих несколько процессов одновременно. 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.

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

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.

Преимущество:

- такая система позволяет быстрее решить задачу, с которой не справляется один компьютер за нужное время.

Основной недостаток:

- 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?**

Общая классификация архитектур ЭВМ по признакам наличия [параллелизма](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%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) в потоках команд и данных была предложена [Майклом Флинном](https://ru.wikipedia.org/wiki/%D0%A4%D0%BB%D0%B8%D0%BD%D0%BD,_%D0%9C%D0%B0%D0%B9%D0%BA%D0%BB_(%D1%83%D1%87%D1%91%D0%BD%D1%8B%D0%B9)) в 1966 году[[1]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%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%D1%85_%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%D1%85_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC#cite_note-1) и расширена в 1972 году[[2]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%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%D1%85_%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%D1%85_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC#cite_note-2). Все разнообразие архитектур ЭВМ в этой таксономии сводится к четырём классам:

а) [SISD](https://ru.wikipedia.org/wiki/SISD) (Single Instruction stream over a Single Data stream)  — Вычислительная система с одиночным потоком команд и одиночным потоком данных

б) [SIMD](https://ru.wikipedia.org/wiki/SIMD) (Single Instruction, Multiple Data) — Вычислительная система с одиночным потоком команд и множественным потоком данных (типичные представители - векторные архитектуры)

в) [MISD](https://ru.wikipedia.org/wiki/MISD) (Multiple Instruction Single Data) — Вычислительная система со множественным потоком команд и одиночным потоком данных (реальных систем-представителей не существует)

г) [MIMD](https://ru.wikipedia.org/wiki/MIMD) (Multiple Instruction Multiple Data) — Вычислительная система со множественным потоком команд и множественным потоком данных

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

Системами с общей памятью называют системы, в которых несколько процессоров  имеют общую оперативную память.

Преимущество:

- не требуется обмена данными.

Недостаток:

- плохо масштабируется.

Вычислительная система с распределенной памятью:система, в которой каждый  процессор обладает собственной локальной памятью, а общая память отсутствует.  Обмен информацией между составляющими системы сети посредством обмена сообщениями.

Преимущество:

- хорошая масштабируемость.

Недостатки

- отсутствие общей памяти снижает скорость межпроцессорного обмена.

- каждый процессор может использовать только ограниченный объем локального банка памяти.

**20) Способы распараллеливания?**

Первый способ — представление алгоритма задачи в виде частично-упорядоченной последовательности выполняемых работ. Затем 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?**

Dataflow – потоковая архитектура, в которой 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 - оператор или набор операторов.

Очередной узел начинает выполняться 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 объединяются простейшим коммутатором, причем для адресации токенов служат их метки.

**22)Аппаратная реализация Dataflow?**

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

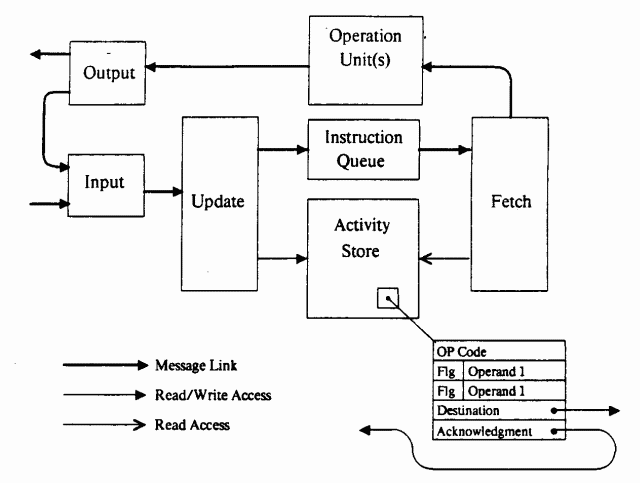
Исполнительное устройство служит для выполнения инструкций и формирования токенов с результатами операций. Оно включает в себя память команд, доступную только для чтения.

Готовность входных данных узла определяется по наличию набора токенов с одинаковыми метками. Для поиска таких наборов и служит устройство сопоставления. Оно реализуется на базе ассоциативной памяти. Используется либо «настоящяя», аппаратная ассоциативная память, либо структуры, работающие аналогично, например, [хэш-таблицы](http://en.wikipedia.org/wiki/Hash_table).

**23) Статическая 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 потоковый компьютер, созданный в Массачусетском технологическом институте в 1974 году. Машина состояла из множества обрабатывающих элементов, связанных коммуникационной сетью. Схема одного элемента показана на рисунке:



Роль устройства сопоставления здесь выполняла память взаимодействий. 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 считывало код операции, и данные отправлялись на обработку в исполнительное устройство.

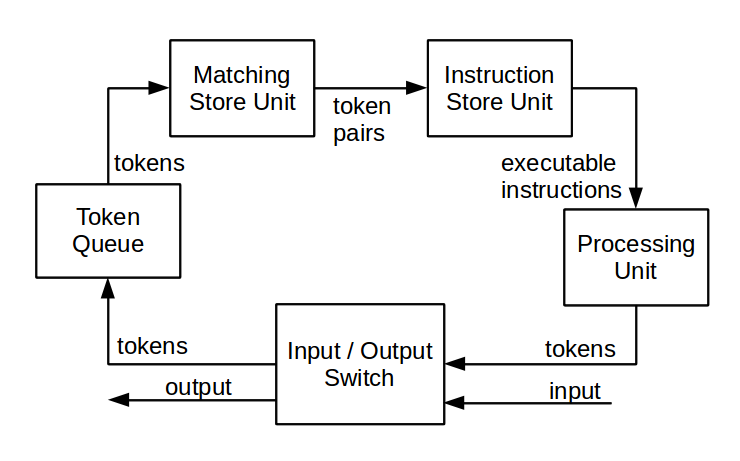
**24)Динамическая Dataflow с помеченными токенами?**

В динамической потоковой архитектуре 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). 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. 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?**

Динамические потоковые вычислительные системы - системы, в которых каждый узел может иметь множество экземпляров. Для того, чтобы различать токены, адресованные в разные экземпляры одного узла, в структуру токена вводится дополнительное поле - контекст, называемый «цветом значения». По сравнению со статической архитектурой появляется возможность параллельного исполнения различных итераций цикла. Одной из первых реализаций динамической потоковой архитектуры была система Manchester Dataflow Machine, созданная в 1980 году.  
  
Производительность потоковых систем существенно возрастает, если они в состоянии поддерживать дополнительный уровень параллелизма, соответствующий одновременному выполнению отдельных итераций цикла или параллельной обработке пар элементов в векторных операциях. Кроме того, в современных языках программирования активно используются так называемые реентерабельные процедуры, когда в памяти хранится только одна копия кода процедуры, но эта копия является повторно входимой (реентерабельной). Это означает, что к процедуре можно еще раз обратиться, не дожидаясь завершения действий в соответствии с предыдущим входом в данную процедуру. Отсюда желательно, чтобы все обращения к реентерабельной процедуре также обрабатывались параллельно. Задача обеспечения дополнительного уровня параллелизма решается в динамических потоковых ВС и реализуется двумя вариантами архитектуры потоковой ВС: архитектуры с помеченными токенами и архитектуры с явно адресуемыми токенами.  
В основе механизма явной адресации токенов лежит наблюдение: все токены в одной и той же итерации цикла и в одном и том же вхождении в реентерабельную процедуру имеют идентичный тег (цвет). При инициализации очередной итерации цикла или очередном обращении к процедуре формируется так называемый кадр токенов, содержащий токены, относящиеся к данной итерации или данному обращению, то есть токены с одинаковыми тегами. Использование конкретных ячеек внутри кадра задается на этапе компиляции. Каждому кадру выделяется отдельная область в специальной памяти кадров (frame memory), причем раздача памяти под каждый кадр происходит уже на этапе выполнения программы.  
В схеме с явной адресацией токенов любое вычисление полностью описывается указателем команды (IP, Instruction Pointer) и указателем кадра (FP, Frame Pointer). Токен выглядит следующим образом: <v, <FP,IP».  
Каждому слову в памяти кадров придан бит наличия, единичное значение которого удостоверяет, что в ячейке находится токен, ждущий согласования, то есть что одно из искомых значений операндов уже имеется. Как и в архитектуре с окрашенными токенами, определено, что вершины могут иметь максимум две входных дуги.

**26)Гибридные Dataflow?**

«Чистые» потоковые архитектуры имели много слабых мест:

* Dataflow-машины давали огромные возможности для параллелизма выполнения, однако 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 программируемых логических интегральных схем (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.   
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.

 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.