Направления развития современной вычислительной техники.

Направления:
1.Эволюционная модификация ЭВМ Фон Неймана
Характерны:
-последовательная обработка информации
-увеличение производительности достигается за счет улучшения технологических характеристик, т.е. повышается тактовая частота, увеличивается разрядность и т.д.

2.Параллельная обработка информации
Улучшение производительности достигается за счет:
-увеличения количества обрабатывающих элементов
-за счет архитектурных решений по связи между вычислительной техникой
-улучшение характеристик отдельных вычислительных элементов

Механические и электромеханические приборы для вычислений.

1. Абак (появился 2000 лет. до.н.э.)
-механизм вычислений основан на позиционировании элементов относительно друг друга(счеты)
2. В конце 17в. Появились перфокарты
3. В 1818 г. Английский ученый Бэббидж разработал механическую вычислительную машину. Близкую по архитектуре современной ЭВМ. Полностью ее реализовать не удалось из-за несовершенства технологического прогресса того времени
4. В 19 в. Было разработано множество специализированных вычислительных устройств, решающих задачи в определенной области.
5. В 1939 г. немецкий ученый Цузе разработал серийную универсальную вычислительную машину Z1. Позже появились ее модификации Z2 и Z3. Цузе использовал электро-механический рельеф для памяти и вычислений.
! Электронный прибор содержит элемента с нелинейной вольтамперной характеристикой.
6. В 1918 г. отечественный ученый Бонч Бруевич разработал электронный прибор ТРИГГЕР, состоящий из двух ламп, имеющих обратную связь. Триггер способен находиться в 2-ух устойчивых состояниях и менять свое состояние под внешним воздействием.
7. В 1930 годах на цепочке триггеров удалось собрать электронный счетчик автономных частиц, который стал первым электронно-вычислительным прибором.

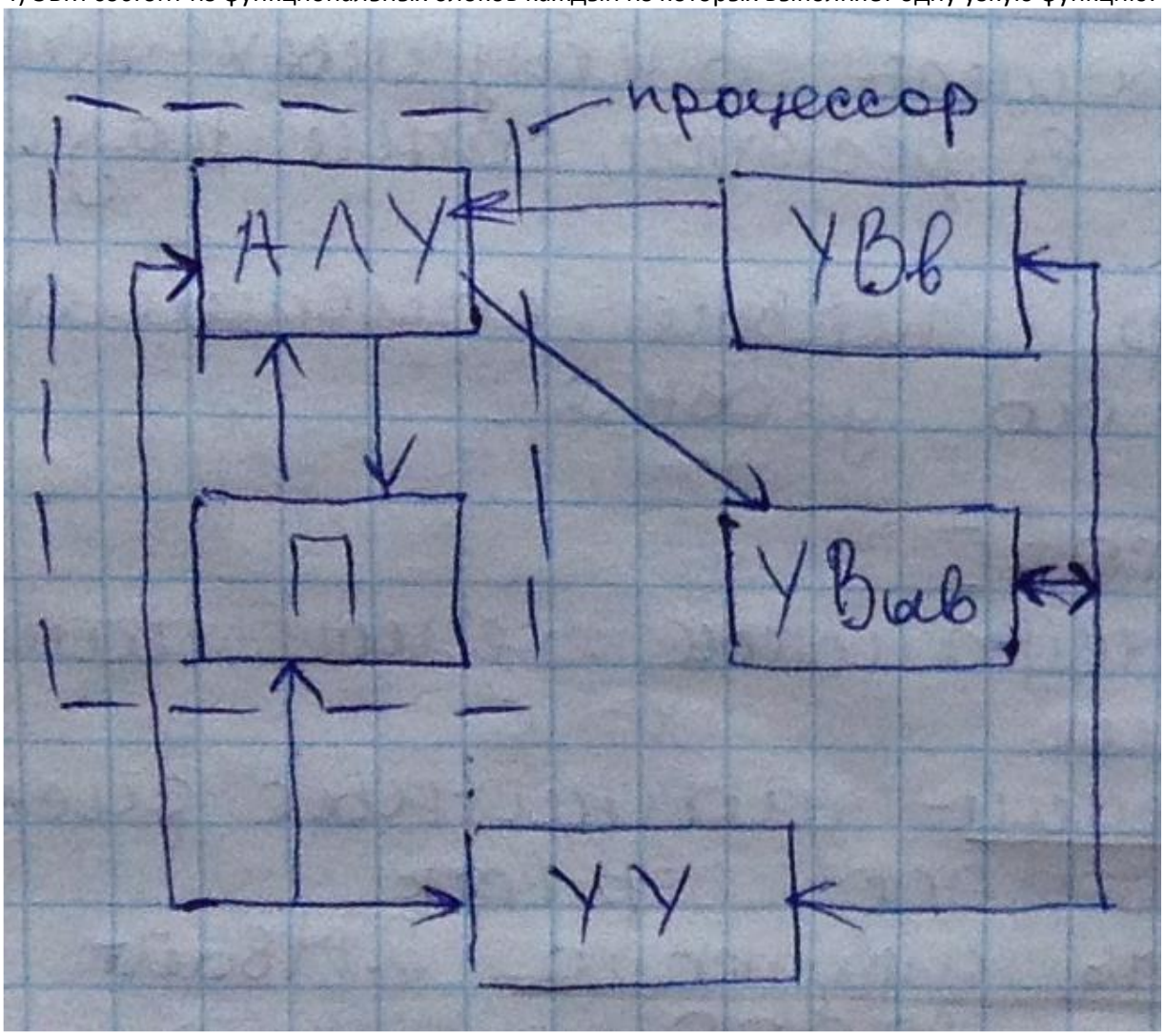
Поколение ЭВМ.

1 поколение:
1945-1955 гг.
В качестве базового элемента-электронная лампа.
В качестве носителя информации-перфокарта. Стоимость такой ЭВМ более 10 млн.дол.
Производительность: порядка 10 000 операций в секунду.
Каждая ЭВМ содержала порядка 20 000 ламп. Многие лампы имели водное охлаждение. В среднем каждые 20 минут одна из ламп выходила из строя. Для обслуживания вычислительного процесса требовалось порядка 40 инженеров. Такие ЭВМ использовались в военных целях. S=200 кв.м. Программировалась в машинных кодах.
2 поколение:
1955-1965 гг.
В качестве базового элемента-транзистор.
В качестве носителя информации-магнитная лента.
Производительность: порядка 100 000 операций в секунду.
Стоимость >100 тыс.долларов. Такие ЭВМ использовались в научных целях, надежность выросла в десятки, сотни раз. S=3 кв.м. программирование на ассемблер, появился первые компиляторы языков высокого уровня.
3 поколение:
1965-1975 гг.
Базовые элементы-интегральные схемы малой степени интеграции.
Носитель информации-магнитные дискеты.
Производительность: 1 млн. операций в секунду.
Оперативная память загружала 2 Мбайт.
Стоимость порядка 10 000 дол, что позволило использовать их на многих предприятиях.
Активно развивались языки программирования высокого уровня (C, Pascal, Fortran…).
4 поколение: 1975- до наших дней.
Тактовая частота- ГГц
Базовые элементы-интегральные схемы большой и сверхбольшой степени интеграции.

ЭВМ Фон Неймана.

В 1940 г. разработал архитектуру, согласно которой строятся современные вычислительные машины.
Помимо архитектуры Фон Нейман разработал основные принципы работы ЭВМ:

1) Информация обрабатывается последовательно
2) Программа хранится в памяти ЭВМ
3) Программа модифицируется средствами ЭВМ
4) ЭВМ состоит из функциональных блоков каждый из которых выполняет одну узкую функцию.

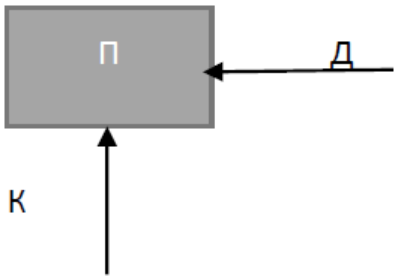


Арифметико-логическое устройство (АЛУ)
Память (П)
Устройство ввода (УВВ)
Устройство вывода (УВыв)
Управляющее устройство (УУ).(Координирует работу всех устройств в составе ЭВМ).
АЛУ+П – процессор

Типы архитектур вычислительных систем.

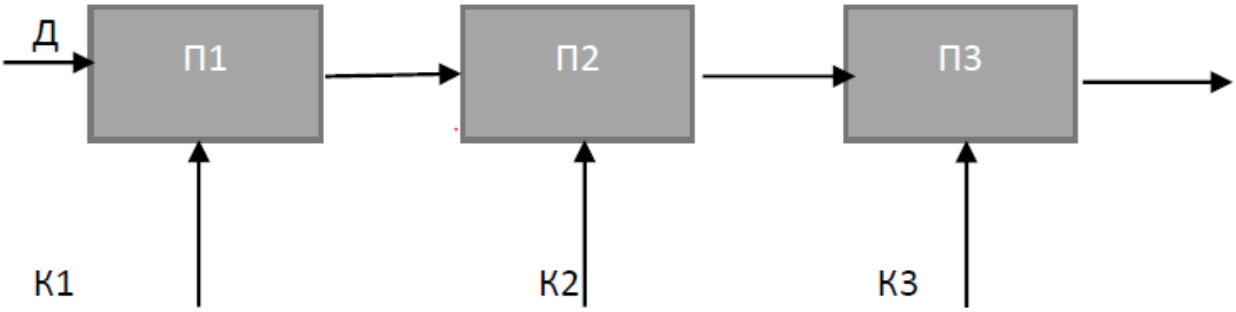
Классификацию систем предложил Флин.

1.SISD(ОКОД)-одиночный поток команд, одиночный поток данных.



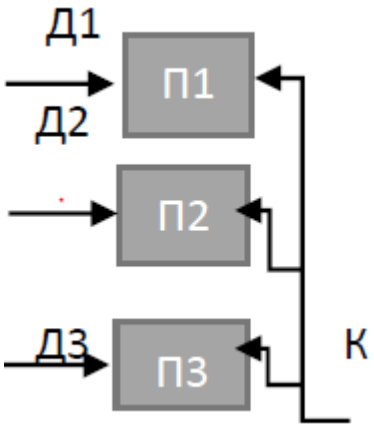
ЭВМ Фон Неймана попадает под эту архитектуру.

2.MISD (МКОД)-множественный поток команд, одиночный поток данных.



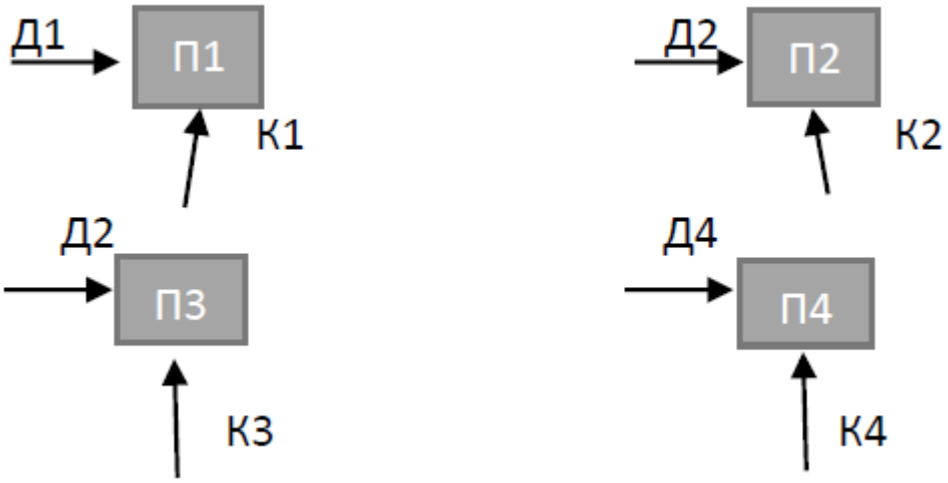
Данный тип архитектуры соответствует конвейерной обработки данных.

3.SIMD(ОКМД)-одиночный поток команд, множественный поток данных.



Данная архитектура соответствует векторной обработки данных (графические операции).

4.MIMD(МКМД)-множественный поток команд, множественный поток данных.



Из данного типа архитектуры можно получить любую из рассмотренных ранее.
Архитектура MIMD наиболее эффективна (перспективна), но при этом сложна в организации.
К параллельным вычислителям относятся все типы архитектур кроме 1-ой.

Методы обмена информации между внешними устройствами и процессором.

1.Непосредственный обмен.
Инициатор обмена-процессор.
Управление обменом осуществляет процессор.
Суть метода: при необходимости обмена данными процессор обращается к устройству и начинает обмен. Состояние устройства при этом не учитывается. Предполагается, что устройство всегда готово к обмену.
Плюсы: высокая скорость обмена.
Минусы: высокий риск потери данных.
Примеры устройств: перемычки, светодиоды.

2.Обмен по опросу готовности.
Инициатор обмена-процессор.
Управление также осуществляет процессор.
При необходимости обмена процессор обращается к устройству и переходит в режим ожидания готовности. В этом режиме процессор выполняет холостые такты. После получения сигнала готовности процессор начинает обмен данными.
Минусы: простой процессора во время ожидания.
Плюсы: существенно снижается риск потери данных.
Этот метод используется для обмена с устройствами, имеющими высокую готовность. Например, цифро-аналоговые и аналого-цифровые преобразователи.

3.Обмен по запросу на прерывание.

Инициатор-внешнее устройство.
Управление обменом осуществляет процессор.
При необходимости обмена внешнее устройство посылает запрос процессору и ожидает подтверждение начала обмена. Если прерывания в процессоре разрешены, то процессор сохраняет свое состояние в стеке и переходит к выполнению подпрограммы, обозначенной как обработчик данного прерывания. В этой подпрограмме предусмотрены все процедуры для выполнения обмена. По окончанию обмена процессор восстанавливает свое состояние из стека и продолжает выполнение прерванной программы. Это самый распространённый способ обмена данными с процессором.
Минусы: большие затраты на выполнение операции.
Плюсы: минимальный риск потери данных; за счёт буферизации, скорость работы внешнего устройства не влияет на производительность.

4.Прямой доступ к памяти.
Инициатор обмена-внешнее устройство.
Управление обмена-осуществляет контроллер прямого доступа к памяти.
При необходимости обмена внешнее устройство обращается к контроллеру, который формирует запрос к процессору. Процессор завершает выполнение текущей операции, выдает сигнал подтверждения контроллеру и отключается от внешних шин. Контроллер формирует адреса, и запись идет непосредственно из внешнего

устройства в оперативную память, либо наоборот. По окончанию обмена контроллер передает управление процессору, и процессор продолжает выполнение программы.
Плюсы: высокая скорость обмена.
Минусы: процессор исключен из вычислительного процесса.
Такой обмен используется для взаимодействия с быстрыми устройствами.
Видеокарта, жёсткий диск, устройства, требующие быстрой передачи большого объёма данных.

Количественные характеристики памяти

1.Объём(ёмкость).
Показывает потенциальное количество информации, которое можно сохранить в памяти. Единица измерения: бит, байт и их производные.

2.Временные характеристики.
1) Определяется параметрами: $t_{сч}=t_{п}+t_{ч}+t_{р}$
$t_{сч}$-время считывания
$t_{п}$-время поиска
$t_{ч}$-время на чтение
$t_{р}$-время регенерации
2) Временные характеристики: $t_{зп}=t_{п}+t_{зп}$
$t_{п}$-время поиска
$t_{зп}$-время записи

3.Надежность.
Определяется вероятностью потери данных при чтении, записи и хранении.

4.Энергозависимость.
Определяет способность памяти сохранять значения при отключении питания.

5.Удельная стоимость.
Определяет цену одной единицы обмена.

6.Разрядность.
Определяет количество бит, которое может быть считано или записано за одно обращение.

7.Масса, габаритные показатели.
Размер, вес.

Характеристики производительности.

1.Тактовая частота.

$$f_{\text{Т}} = \frac{1}{\tau_{\text{п}}}$$

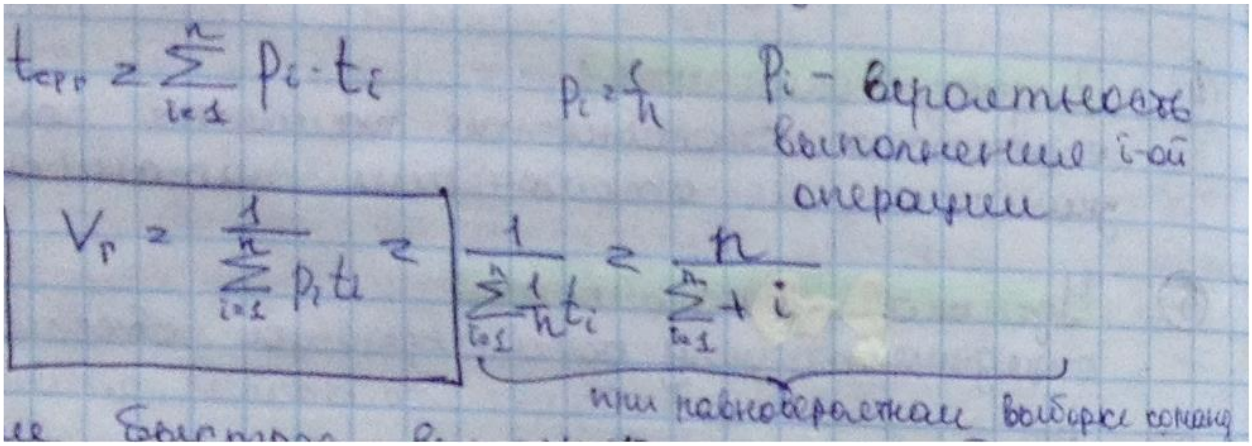
Определяет время одной элементарной операции.
Измеряется ГЦ и производных ГЦ.
Данная характеристика подходит для сравнения только процессоров с одной архитектурой.

2.Среднее быстродействие (tcp) или номинальное быстродействие.

$$t_{\text{ср}} = \frac{\sum_{i=1}^{n} t_i}{n}, v_{\text{ср}} = \frac{1}{t_{\text{ср}}}$$

Определяет среднее время выполнения одной операции
номинальное быстродействие (вторая формула).
n-количество операций.
t_i-время выполнения i-ой операции.
Характеристика существенно более объективна чем 1-ая.

3.Быстродействие по Гиббсону.
Предложил учитывать вероятности выполнения каждой команды.



Чем быстрее выполняется наиболее вероятные команды, тем выше производительность.

4.Использование универсальных тестов.

Тесты выпускаются независимыми компаниями. Тесты показывают производительность. Единица измерения которых является:

-FLOPS-количество операций с плавающей точкой за 1с.

-MOPS-количество операций над целыми числами за 1с.

И производные от этих единиц.

Современные высокопроизводительные вычислительные системы достигают производительности Петофлопсы. Если ЭВМ планируется использовать для специализированных задач, то целесообразно разработать свой набор тестов для оценки соответствия заданным требованиям.

Формат машинных команд.

Процессор ЭВМ выполняет инструкции, представленные в двоичном виде, каждая инструкция содержит информацию о необходимом действии, операнды, над которыми следует провести это действие, информацию о том, куда следует поместить результат и адрес следующей операции или команды.

В общем виде машинная команда представляет собой такую конструкцию:

|КОП|A1|A2|A3|A4| и содержит 5 полей:

1: КОП –код операции.

2: A1-адрес 1-го операнда.

3: A2-адрес 2-го операнда.

4: A3-адрес приемника результата.

5: A4-адрес следующей команды.

Поля в машинной команде могут иметь как плавающий, так и фиксированный размер.

Если хотя бы одно поле в машинной команде не имеет фиксированной величины, то такой формат будет плавающим.

В большинстве случаев машинные команды располагаются в памяти последовательно, чтобы вычислить адрес следующей команды достаточно к адресу текущей команды прибавить ее длину. Поэтому адресное поле A4 обычно отсутствует и вводят специальные команды передачи управления-такие команды называют 3-х адресными:

|КОП|A1|A2|A3|.

В качестве приемника результата чаще всего выступает внутренний регистр процессора, называемый аккумулятором: AX.

В этом случае адресное поле A3 не требуется. Такие команды называются 2-х адресные:

|КОП|A1|A2|.

При этом предусмотрены команды сохранения содержимого регистра в память.

Множество команд имеет только один операнд, либо берут 2-ой операнд из внутренних регистров процессора, в этом случае отпадает необходимость адресного поля A2:

|КОП|A1|.

При этом предусматриваются команды загрузки данных из памяти во внутренний регистр.

Существуют команды, которые не требуют операндов, либо берут их из внутренних регистров или из стека. В этом случае адреса операнд не требуются, такие команды называют безадресные: |КОП|.

В современных процессорах чаще всего используются одноадресные команды, реже двухадресные и безадресные, 3-х и 4-х адресные команды практически не используются.

Способы адресации.

В машинной команде содержится адресный код при этом ячейки памяти имеют физический адрес.

Адресный код, который содержится в команде обозначают Ак-командный адрес.
Аф-физический адрес ячейки памяти.
В большинстве случаев физический и командный адреса отличаются, т.е. Аф неравно Ак.
Алгоритм получения физического адреса по командам называют способами адресации.
$А_ф = f(А_к)$.
Все способы адресации делятся на две группы:
1.неявные.
В машинной команде не содержится адресного поля.
Способы адресации:
1) операнд (Q++)
2) адрес: например, мы удалили поле А3 из формата команды, подразумевая, что в качестве приемника результат будет выступать регистр аккумулятора.
2.явные.
1) прямая адресация (физический адрес совпадает с командным, т.е. Аф=Ак).
2) непосредственная адресация (в адресном поле содержится сам операнд, а не адрес).
3) косвенная адресация (машинная команда содержит адрес ячейки памяти, в которой хранится операнд $А_ф = M[(А_к)]$ М-массив памяти.).
4) автоинкрементная и автодекрементная адресация (после выполнения операции адрес автоматически увеличивается или уменьшается на заданную величину. Такой способ адресации удобно использовать в циклах).
5) базовая адресация (физический адрес формируется как сумма базового адреса и командного адреса, т.е. Аф=Аб+Ак, при этом базовый адрес хранится в одном из внутренних регистров процессора, а командный адрес содержится в команде. Такой способ адресации используется при работе с массивами, при этом Аб-начало расположения массива в памяти определяет, а Ак определяет физический адрес конкретного элемента массива в памяти.
6) базово-индексная адресация (физический адрес формируется как сумма 3-х слагаемых Аф=Аб+Аи+Ак, используется при работе с двумерными массивами. Аи-индексный адрес, Ак-командный адрес.).
7) стековая адресация (в случае стековой адресации работа ведется только с одним адресом, который указывает на вершину стека).
8) укороченная адресация (старшие биты адреса подразумеваются или берутся из внутреннего регистра процессора: ||Ак|).
9) регистровая адресация (адресный код содержит номер регистра, с которым необходимо осуществить операцию).
Способ адресации определяется кодом операции.

RISC и CISC процессоры.

До начала 1980 г. развитие процессоров происходило за счет увеличения количества поддерживаемых команд усложнения способов адресации, все производители старались приблизить систему команд процессора к операторам и конструкциям языков программирования высокого уровня. В частности, некоторые процессоры аппаратно поддерживали численное интегрирование, разложения в ряд Фурье и т.д. После проведенных исследований американскими учеными выяснилось, что 98% времени процессор выполняет самые простые операции, такие как сдвиг, выгрузка из регистров в память, загрузка из памяти в регистр, на остальные сложные операции приходится очень малая доля времени.
Те процессоры, которые разрабатывались до этих исследований, они имели CISC-архитектуру, которая расшифровывается как «расширенный набор команд».
Отличительной особенностью CISC процессоров является:
1. Плавающий формат машинных команд.
2. Поддержка множества способов адресации.
3. Большое количество возможных кодов операций.
4. Сравнительно малое количество внутренних регистров.

В результате проведенных исследований появились RISC процессора, или процессоры с сокращенным набором команд, задача которых ставилась при разработке - максимально быстрое выполнение самых простых команд, сложные команды RISC процессоры не поддерживают. К тому же, развитие языков программирования высокого уровня существенно опережало развитие процессоров и смысла соответствовать этим языкам не было.
Сложные конструкции языков программирования сложного уровня преобразовались в совокупность простых с помощью компилятора. При этом разработка компиляторов для RISC процессоров усложнилась на порядки.
Отличительные особенности RISC процессоров:
1. Малое количество поддерживаемых команд.

2. Фиксированный формат машинных команд.
3. Малое количество поддерживаемых способов адресации.
4. Все операции выполняются над содержимым внутренних регистров, результат также помещается во внутренний регистр. При этом предусмотрены команды перемещения из памяти во внутренний регистр и из внутреннего регистра в память.
5. Большое количество внутренних регистров.

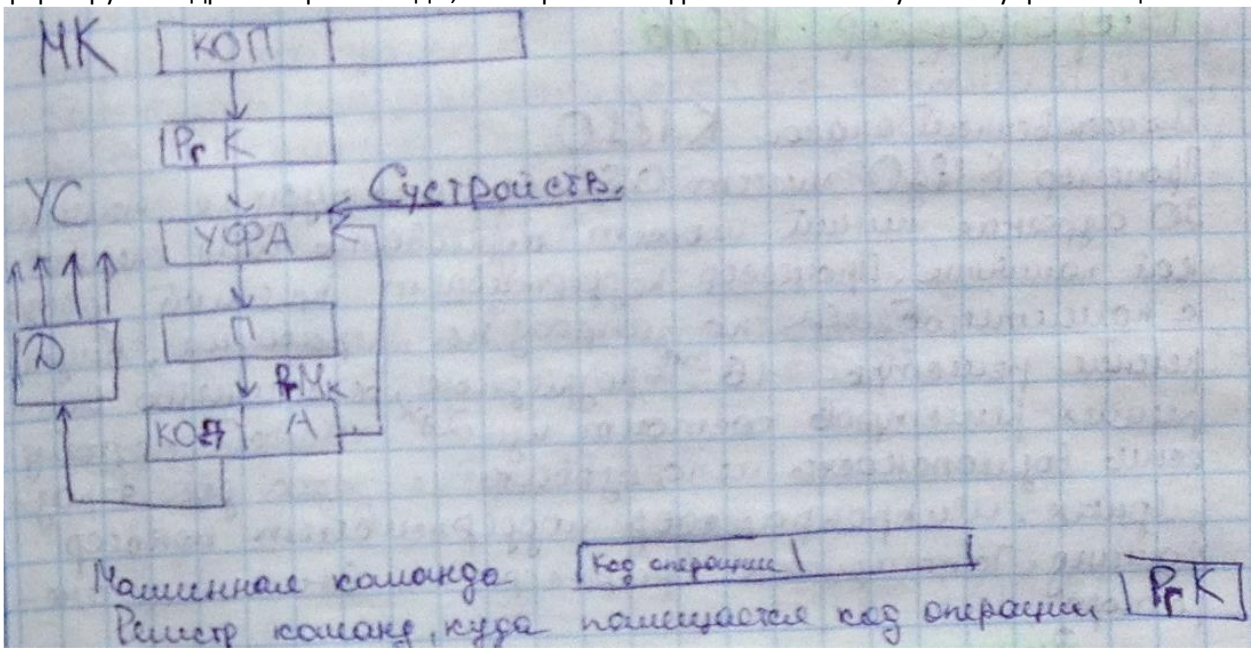
CISC процессора продолжают до сих пор производиться из-за большого количества программного обеспечения написанного под эту архитектуру. Производители процессоров часто вынуждены обеспечивать интерфейс CISC процессора, ставить преобразователи и внутри выполнять вычисления RISC процессоров.

Жесткое и микропрограммное устройство управления.

Жесткое управление- представляет собой заранее спроектированную схему, которая формирует управляющие сигналы, в зависимости от состояния устройств, и выполняемые операции.
Достоинства жесткого управления:
- высокая скорость работы
- простота реализации
Недостатки:
- невозможность модификации

Микропрограммное устройство управления.

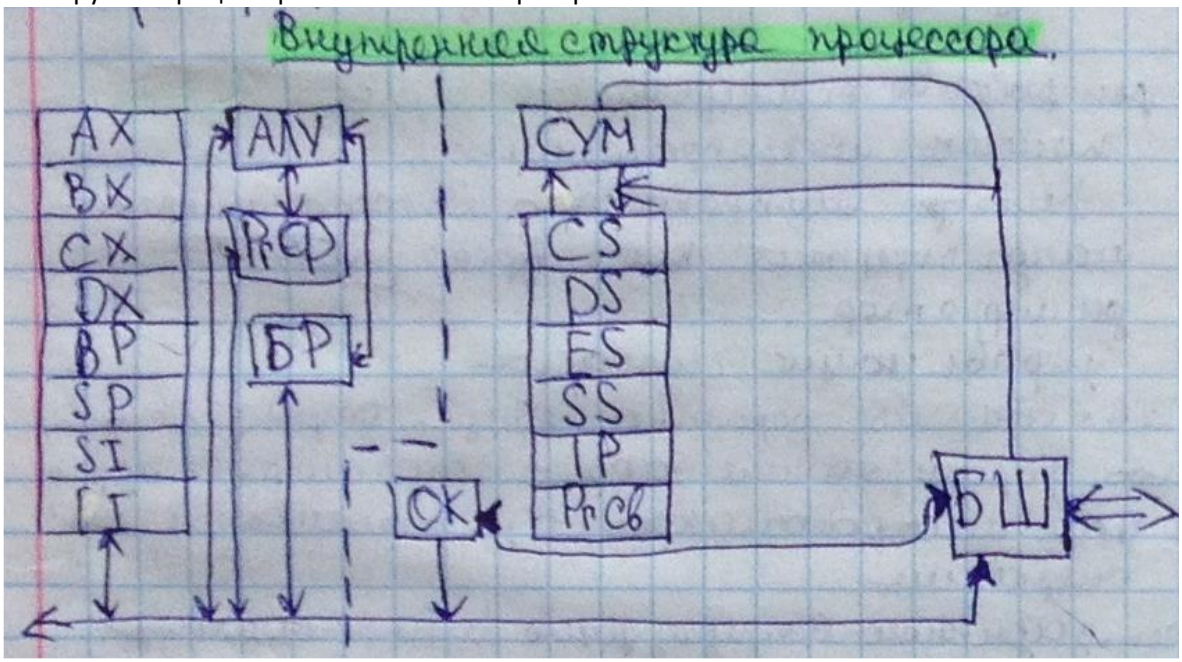
Управляющие сигналы формируются на основе микропрограммы хранящейся в памяти. Устройство управления получает в качестве исходных данных код операции, состояния устройств, и на основе этих данных формируется адрес микрокоманды, в который зашифрована вся совокупность управляющих сигналов.



УФА-устройство формирования адреса
П-память микропрограмм
РгМк- регистр микропрограмм, каждая микрокоманда содержит код и адрес (КОД|А)
Д-дишифратор
УС-управляющие сигналы
Сустройств-сигналы обратной связи от устройств
Устройство формирования адреса (УФА) формирует адрес следующей микрокоманды в зависимости от:
-кода операции
-адреса, заданного в предыдущей микрокоманде
-сигнала об обратной связи со всех устройств в ЭВМ
Адрес подается на память микрокоманд, где из заданной ячейки извлекается необходимая микрокоманда, состоящая из 2-ух полей (кода и адреса).
Поле адреса используется для формирования следующего адреса микрокоманды в УФА, а код содержит информацию о необходимых управляющих сигналах, которые специальным образом зашифрованы.
Дешифратор на основе кода формирует управляющие сигналы для всех устройств.
Если необходимо изменить поведение микропрограммного устройства управления достаточно изменить содержимое памяти микрограммы.

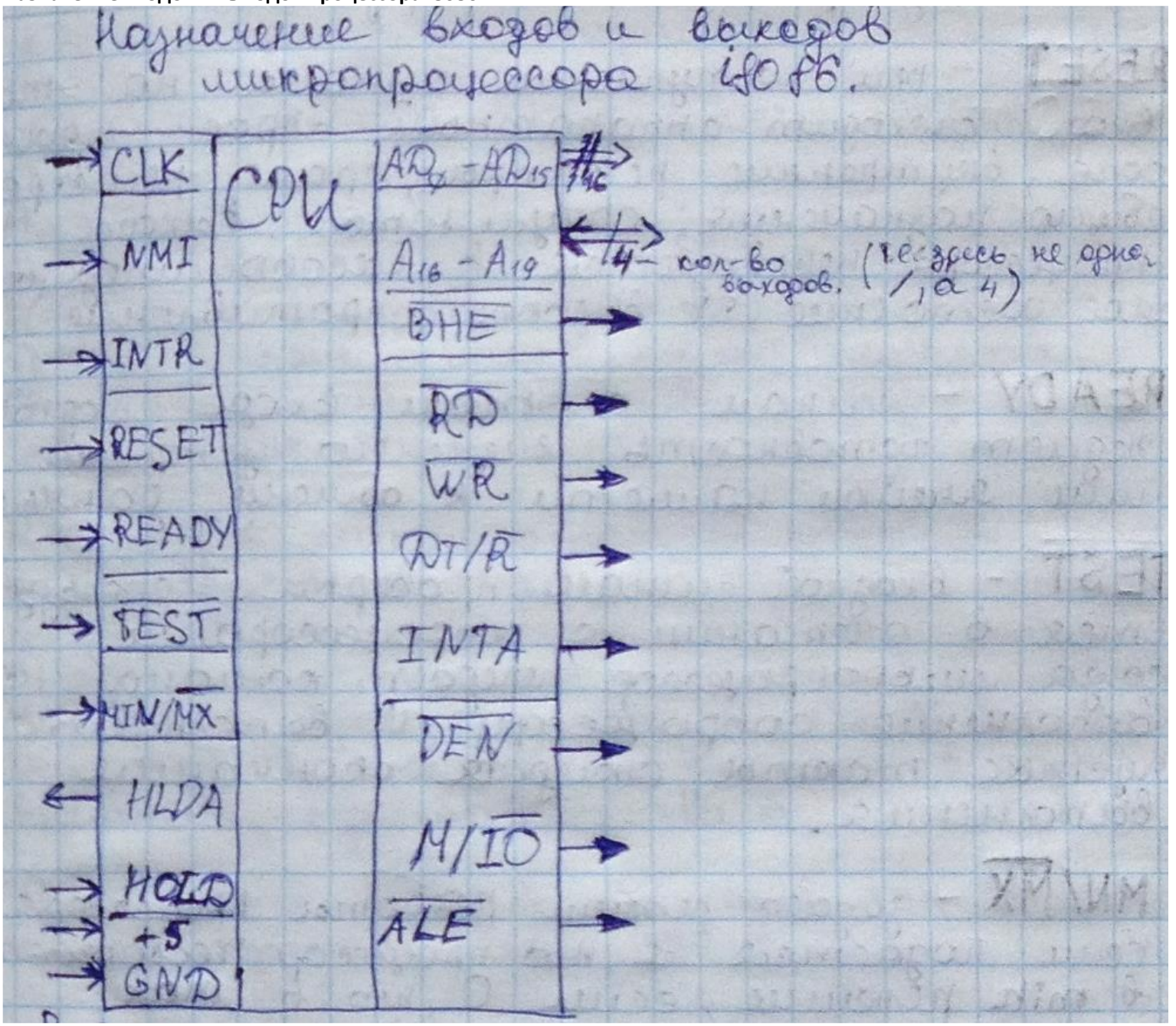
Микропроцессор i8086.

Отечественный аналог К1810.
Процессор К1810 имеет CISC архитектуру, с помощью 20 адресных линий может адресовать 2Мб оперативной памяти. Процессор поддерживает прямой доступ к памяти, обмен по запросу на прерывание, внутренние регистры 16-ти разрядные, большинство внутренних регистров состоит из 2-ух частей, поэтому есть возможность использовать из как два 8-ми разрядных. Микропроцессор поддерживает конвеер команд. Тактируется процессор от внешнего генератора.



Процессор состоит из 2-ух базовых частей: левая часть-операционные устройства, правая часть-шинный интерфейс. Операционное устройство выполняет вычисления и логические операции, шинный интерфейс занимается формированием адресов и обменом данными с другими устройствами.
(OV) Операционное устройство состоит из регистров общего назначения (AX, BX, CX, DX, SP, SI, PI), арифметико-логического устройства (АЛУ), регистры флагов (РгФ), буферных регистров (БР).
(ШИ) Шинный интерфейс содержит сумматор (СУМ), который обеспечивает сегментную организацию памяти, 4 сегментных регистра (CS, DS, ES, SS), счетчик команд IP, регистр связи (РгСв), очередь команд (ОК), буфер шины (БШ-обеспечивает электрическое сопряжение).
Операционное устройство и шинный интерфейс могут работать параллельно, пока ОУ выполняет инструкцию, ШИ может считывать следующие команды и заполнять ими очередь. Таким образом повышается общее быстродействие.

Назначение входов и выходов процессора i8086



CLK-вход синхронизаций, на который подаются тактовые импульсы, задающие частоту процессора.
NMI-вход немаскируемых запросов на прерывание. При поступлении запроса на этот вход он обрабатывается независимо от текущего состояния процессора.
INTR-вход маскируемых запросов на прерывание. При поступлении запроса на этот вход процессор проверяет состояние флага, разрешающего обработку прерывания, и если прерывание разрешены, то запрос обрабатывается, запрещены-игнорируется.
RESET- при поступлении сигнала на этот вход происходит аппаратный сброс процессора, внутренние регистры, кроме регистров общего назначения, обнуляются. Выходы процессора переводятся в высоко-импедантсное состояние (т.е. высокое сопротивление).
READY-сигнал на этом входе подтверждает готовность внешнего устройства или ячейки памяти к обмену данными.
TEST-входной сигнал проверки. Используется в сочетании с сопроцессорами, когда микропроцессор передает команду на выполнение сопроцессору и выполняет пустые такты, ожидая окончания выполнения.
MN/MX-задает режим работы процессора. Если подается 1, то процессор работает в min режиме, если 0, то в max. Min режим используется в однопроцессорных конфигурациях, max в сложных многопроцессорных. В max режиме большинство входов и выходов меняют свое назначение.
HOLD-вход запроса на прямой доступ к памяти. При поступлении сигнала процессор завершает выполнение текущей операции, передает управление контроллеру прямого доступа к памяти и отключается от внешних шин.
+5-питание (+5В).
GND-заземление (отрицательный контакт питания).
Выходы:

AD0-AD15-совмещённая шина адреса и данных. В одни промежутки времени передаются адреса, в другие данные, т.е. используется временное разделение шины.
A16-A19-старшие разряды шины адреса.
BHE-используется для разрешения подключения старшего банка памяти.16-ти разрядная шина может использоваться и как 8-разрадная.
RD-0-ой сигнал означает, что процессор осуществляет операцию чтения из памяти или внешнего устройства.
WR-0-й сигнал означает, что процессор осуществляет операцию записи в порт внешнего устройства или в память.
DT/R-1 на выходе означает, что идет передача данных, 0- означает, что идет прием.
INTA-используется для подтверждения обработки запроса на прерывание со входа INTR.
HLDA-выход подтверждения запроса на прямой доступ к памяти.
DEN- 0-ой сигнал свидетельствует о том, что по совмещенной шине передаются данные.
ALE-при выставлении адреса на совмещенную шину адреса и данных на данном выходе формируется страбирующий импульс (нет четкого закона, четких событий). Этот импульс используется в схеме разделения совмещенной шины на две независимые.
M/IO-выход (М- память, IO- порты ввода/вывода) разделения адресных пространств памяти и внешних устройств.

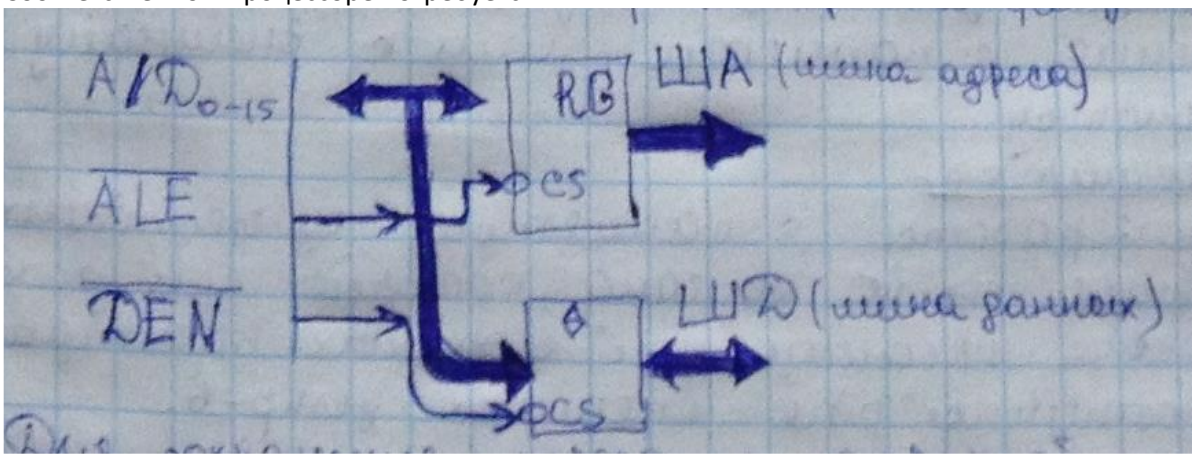
Программная модель процессора i8086.

С точки зрения ПО процессор представляет набор внутренних регистров.
Все регистры процессора 16-разрядные и делятся на группы:
1 группа регистры общего назначения (в данном процессоре 8 регистров общего назначения, которые участвуют в операциях обмена памятью, имеют набор команд для чтения и записи, помимо того, каждый имеет свою специфическую функцию).
AX-будучи 16-разрядными он может использоваться как два 8-разрядных: **|AL|Al|**.
Этот регистр называют аккумулятором в него обычно сохраняется результат выполненной операции.
BX-также может использоваться как два 8-разрядных. Специфическое назначение-может использоваться как указатель на ячейку памяти, т.е. в нем может храниться часть адреса ячейки памяти, к которой происходит обращение.
CX-может использоваться как два 8-разрядных. Специфическая функция-используется как счетчик при выполнении операции цикла. При выполнении очередной операции цикла происходит автоматическое уменьшение содержимого регистра CX и переход по указанному адресу, если содержимое регистра CX не равно нулю.
DX-может быть использован как два 8-разрядных. Это единственный регистр, который может быть использован как указатель на порт ввода/вывода.
SI-может использоваться как дополнительный указатель на ячейку памяти. В большинстве случаев этот регистр используется для работы со строками.
DI- используется как указатель на ячейку памяти. Они отличаются использованием сегментных регистров.
BP- адресует ячейку памяти относительно сегментов стека. Используется для работы с локальными переменными и параметрами процедур.
SP-является указателем на вершину стека. Наименее общий из регистров общего назначения, т.к. иных механизмов для работы со стеками нет.
Таким образом и 8 регистров общего назначения -4 используется как два 8-разрядных; - остальные как 16-ти разрядные.
2 группа: сегментные регистры
Процессор i8086 использует 20 адресных линий при этом его внутренние регистры 16-разрядны. Для формирования 20 разрядов используется сегментация. В результате физической адрес, по которому происходит обращение обозначается следующей формулой:
Аф=(сек.рег.) *16+смешение. Сек.рег.-содержимое сегментного регистра.
*16 эквивалентно сдвигу влево на 4 разряда. Память имеет циклическую организацию, такую же организацию имеет и сегменты, так что выход за пределы памяти и сегмента невозможен.
CS-является указателем сегмента кода. С помощью него формируется адрес следующей команды. В качестве смещения в пределах этого сегмента используется содержимое регистра IP, которой является счетчиком машинных команд. Таким образом адрес следующей команды можно получить следующим образом: **Ак=(CS) *16+IP.**
DS-содержит указатель на начало расположения в памяти сегмента данных. В качестве смещения могут использоваться регистры общего назначения, например, BX, DI.
Таким образом, адрес данных можно получить по формуле: **Ад=(DS) *16+DI.**

ES содержит указатель на дополнительный сегмент, который в большинстве случаев является дополнительным сегментом данных.
SS-содержит указатель на сегмент стека. Адрес стека: **Ac=(SS) *16+SP**
Каждый сегмент весит 64 килобайта. Теоретически сегменты могут пересекаться и даже накладываться. Чтобы перейти к другом сегменту необходимо сменить содержимое сегментного регистра специальной командой.
Отдельной группой выступает регистр флагов. Рег.Ф.
Флаги управляют поведением процессора и содержат признаки выполнения предыдущей операции.
IP-счетчик команд, используется для формирования адреса следующей команды совместно с сегментным регистром **CS.**
Рег.Ф-регистр флагов, данный регистр содержит 16 бит, при этом каждый бит в регистре является самостоятельным флагом. Флаги используются либо для управления поведением процессор, либо содержат информацию о результатах выполнения предыдущей операции.
OF-флаг переполнения, в нем устанавливается 1, если в результате предыдущей операции произошел выход за границы разрядной сетки. (пример деление на 0).
DF-флаг направления, определяет порядок обработки цепочек байт в соответствующих команд, соответственно от больших к меньшим или от меньших к большим.
IF – флаг разрешения маскирует прерывания. Если данный флаг установлен в 0, то прерывания на входе INTR игнорируются, если 1, то запросы на прерывание на входе INTR продолжаются.
TF-флаг трассировки, если установлен в 1, то процессор переходит в пошаговый режим выполнения команд, и после каждой команды генерирует внутренние прерывания в обработке этого прерывания, например, можно получить содержимое всех внутренних регистров процессора.
SF-флаг знака, он устанавливается в 1, если результат последней операции отрицательный.
ZF- флаг нулевого результата, 1-если результат последней операции равен 0.
AF-флаг меж тетрадного переноса, флаг устанавливается. Флаг устанавливается в 1, если осуществляется заем или перенос между четверками бит. Данный флаг используется в 2-ых-10-ых операциях.
PF-флаг четности результатов.
CF-флаг переносов, устанавливается в 1 при переносе или заёме между байтами или словами.
Флаги признаков последней операции используются в условных конструкциях.

Схема демультиплексирования шины адреса и данных.

Схема демультиплексирования позволяет из одной объединенной шины адреса и данных получить независимые шины адреса и данных.
Соответственно в процессоре потребуется:



-Для хранения адреса на адресной шине требуется регистр.
-При появлении 0-го сигнала входе CS регистр запоминает то, что присутствует на входе.
Для передачи данных используется шинный формирователь (обозначается как ромбик, только по средине минус так сказать).
Шинный формирователь при отсутствии 0-го сигнала на входе CS обеспечивает гальваническую развязку своего входа и выхода. Если на вход CS подается 0, то шинный формирователь соединяет вход с выходом.

Организация адресного пространства портов ввода/вывода и памяти.

Существует 3 вариант организации адресных пространств:
1.разделение:
В этом случае используется выход M\IO, который однозначно определяет принадлежность выставленного адреса. Адресные пространства полностью независимы.
Достоинства:
1) Все адреса доступны для памяти и для внешних устройств.

2) Во время выполнения команды однозначно известно осуществляется обращение к памяти или к внешнему устройству.

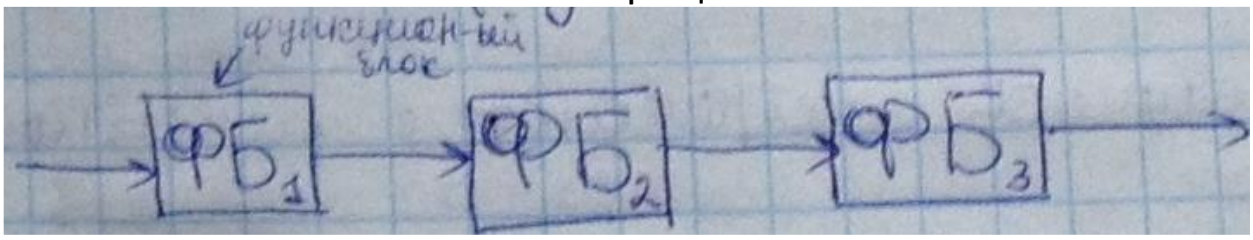
Недостатки:
1) Для работы с памятью существует большое количество команд, которые при данном способе организации адресных пространств недоступны для внешних устройств.
2) Существенное снижение количества способов адресации доступных для внешних устройств.

2.совмещение адресных пространств:
На выходе M/IO всегда присутствует 1, т.е. процессор считает, что он всегда работает с памятью, при этом часть адресного пространства отводится под порты ввода/вывода.
Достоинства:
1) Доступны все команды и способы адресации для портов внешних устройств.

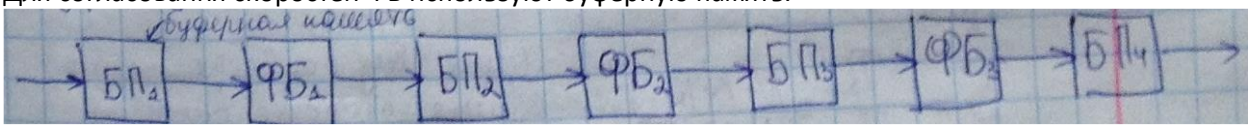
Недостатки:
1) Часть адресного пространства съедается портами, соответственно эта память недоступна для адресации.

3.Гибридное использование адресного пространства:
Часть портов располагается в адресном пространстве памяти и к ним можно обращаться как к памяти, др. часть портов имеет выделенное адресное пространство, и не отнимает пространство памяти.

Конвейеризация вычислений.



Для согласования скоростей ФБ используют буферную память.

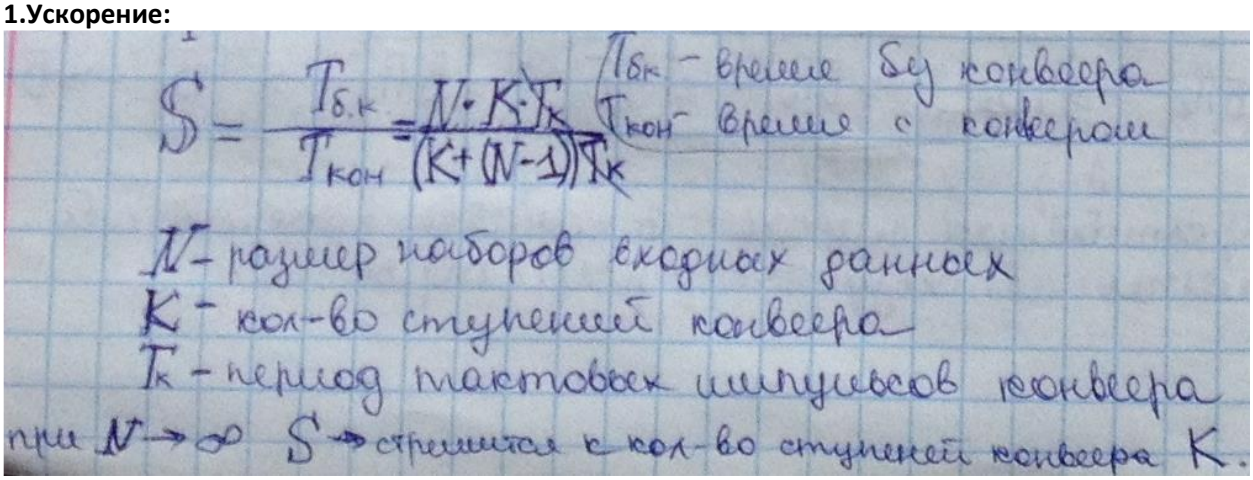


В простейшем случае в качестве простейшей памяти используется регистры.
Конвейеры делятся на 2 вида: синхронные и асинхронные.
В синхронных конвейерах начало выполнения любой операции определяется тактовыми импульсами.
В асинхронных обработка осуществляется по мере готовности исходных данных.
Период тактовых импульсов определяется max временем обработки самым медленным функциональным блоком и временем записи в буферную память.
Конвейеры бывают как линейные, так и нелинейные.
В линейном конвейере ФБ соединяются последовательно.
В нелинейном конвейере возможны дополнительные связи при этом допускаются обратные связи.

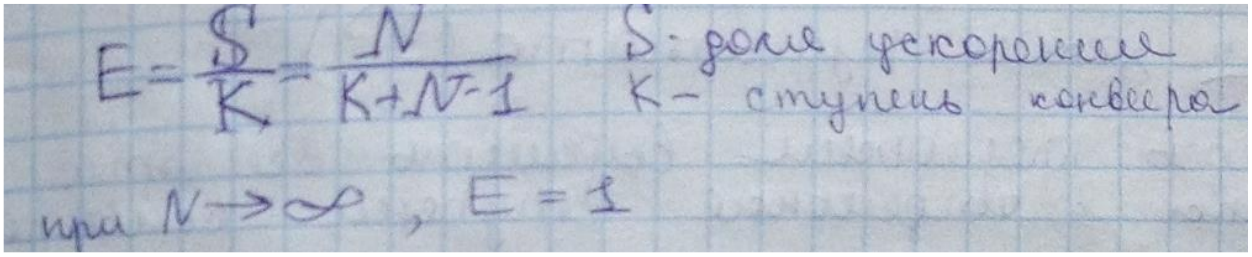
!!В большинстве случаев встречаются линейные конвейеры.

Метрики эффективности конвеера.

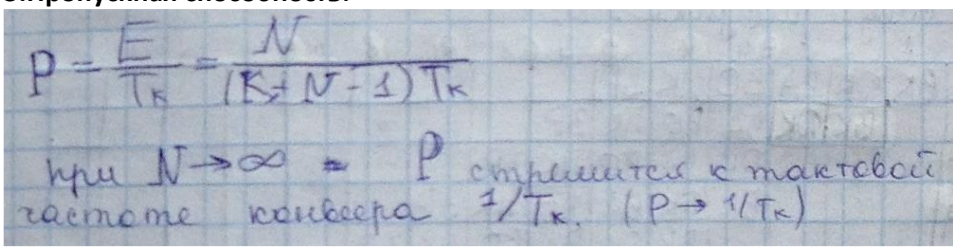
1.Ускорение:



2.Эффективность конвейера:



3.Пропускная способность:



Конвейер команд.

Является самым распространенным конвейером во всех современных процессорах. Впервые конвейер команд был предложен в 1956 г. академиком Лебедевым.

Цикл команды представляет собой последовательность операции.

1 этап: выборка команды (ВК). Команда читается из памяти и заносится во внутренний регистр процессора.

2 этап: дешифрация команды (декодирование) (ДК). Определяется код операции и определяются способы адресации операнда.

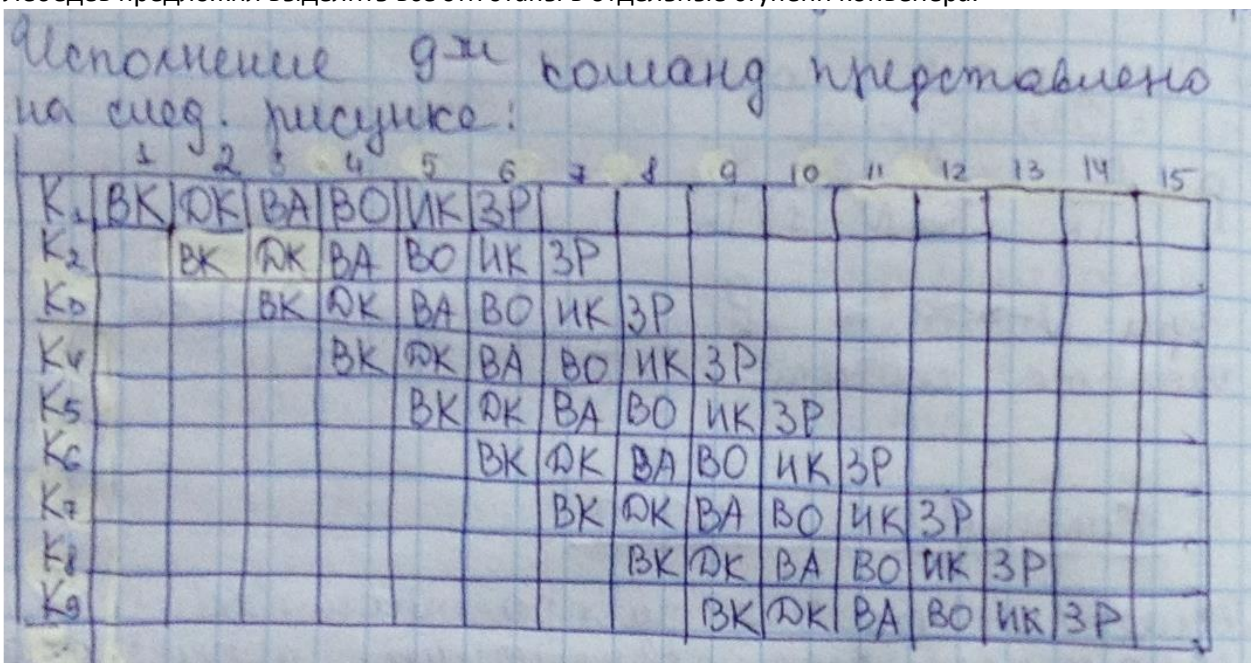
3 этап: вычисление адресов операндов (ВА).

4 этап: выборка операндов (ВО). Команды читаются из памяти во внутренний регистр процессора.

5 этап: исполнение команды (ИК).

6 этап: запись результата (ЗР).

Лебедев предложил выделить все эти этапы в отдельные ступени конвейера.



Выполнение 9-ти команд на конвейере заняло 14 тактов. Без конвейера потребовалось бы 54 такта. В реальности такого существенного увеличения производительности достичь не удается.

Конфликтные ситуации в конвейере не позволяют достичь потенциальной производительности. Конфликты в конвейере также показывают рисками. Существует 3 причины конфликтов:

1) Структурный риск:

- несколько команд пытаются обратиться к одному и тому же ресурсу. (чаще всего этим ресурсом является память). Т.к. многие команды не предполагают обращения к памяти, влияние структурного риска на производительность не велико.

2) Риск по данным:

- две команды на разных ступенях конвейера работают с одной и той же переменной. Риск по данным является типичным для конвейера.

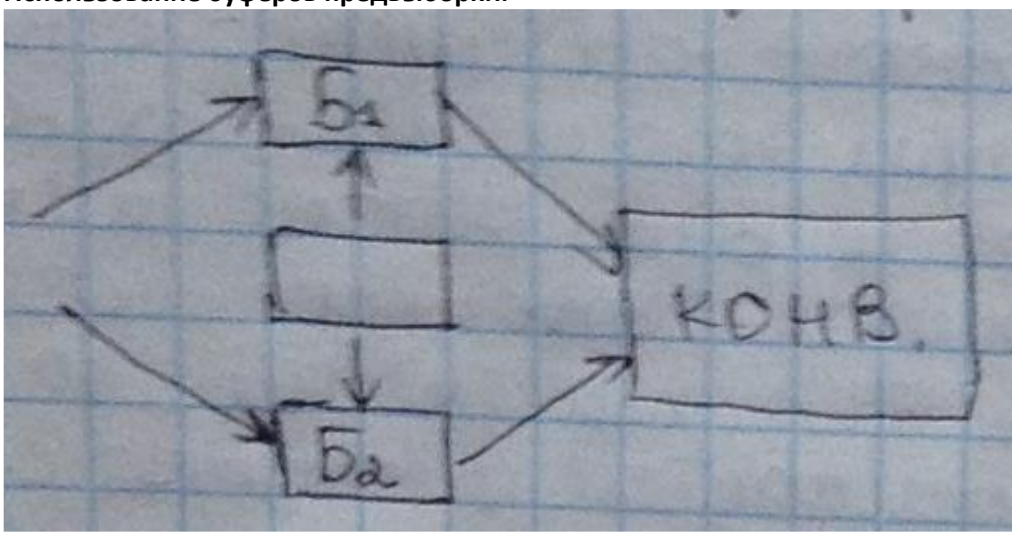
3) Риск по управлению:

- вызван неоднозначностью выполнения следующей команды после команды перехода (условного и безусловного). Это самая большая проблема для производительности конвейера.
Проблемы, связанные с безусловным переходом, решаются анализом команды. На этапе ее выборки, т.к. переход однозначный и адрес перехода известен.
Основную проблему составляет условный переход, т.к. до выполнения команды неизвестна необходимость перехода.

Методы решения проблем условного перехода.

Для устранения и частичного сокращения издержек используется следующие подходы:

1. **Использование буферов предвыборки.**



На этапе декодирования команды распознаются. Наличие условного перехода. Соответственно, становятся известны адреса, по которым следует переходить. В дальнейшем команды из памяти считываются в 2 буфера. В 1-ый буфер считываются команды, которые будут выполняться при отсутствии перехода. Во 2-ой буфер при его наличии.
Когда становиться известен факт перехода содержимое одного из буферов отбрасывается, содержимое второго буфера выполняется.

2. Множественные потоки:

Часть ступеней конвейера дублируются. Аналогично 1-му подходу используется результаты одной из ветвей.

2. **Задержанный переход:**

Данный подход предполагает выполнение команд, следующих за командой условного перехода как будто такой команды и не было, естественно это имеет смысл только если следующие команды будут полезными. Основная нагрузка кладется на компиляторе, который старается вставить после команды условного перехода команды, которые будут выполнены в любом случае независимо от перехода.

4. Предсказание переходов:

Данный метод является наиболее эффективным способом борьбы с рисками по управлению. Идея такого подхода заключается в том, что еще до появления команды условного перехода либо сразу же при ее поступлении на конвейер делается предположение о наиболее вероятном исходе такой команды. Последующие команды подаются на конвейер согласно сделанному предположению. При ошибочном предсказании конвейер необходимо вернуть к состоянию до выборки ненужных команд. Цена ошибки предсказания достаточно высока (в плане производительности). Вводится понятие «точность предсказания». В современных подходах точность предсказания превышает 85%.