Оглавление

Классификация ЭВМ. Основные характеристики ЭВМ (1.10-1.12)

Классификация ЭВМ

Классификация ЭВМ по назначению:

Общего назначения

- Супер ЭВМ
- Минисупер ЭВМ
- Мэйнфреймы
- Серверы
- Рабочие станции
- Персональные компьютеры
- Ноутбуки
- Портативные компьютеры
- ...

Специализированные

...

Классификация ЭВМ по структуре:

- Однопроцессорные
- Многопроцессорные

Классификация ЭВМ по режимам работы:

- Однопрограммные
- Мультипрограммные
- Мультипрограммные в составе систем
- ЭВМ в системах реального времени

Классификация ЭВМ по количеству потоков команд и данных:

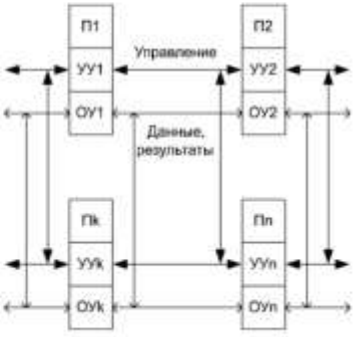
- ЭВМ с одним потоком команд и одним потоком данных (ОКОД, SISD);
- ЭВМ с одним потоком команд и многими потоками данных (ОКМД, SIMD);
- ЭВМ с многими потоками команд и одним потоком данных (МКОД, MISD);
- ЭВМ с многими потоками команд и многими потоками данных (МКМД, MIMD).

ОКОД, SISD

ОКМД, SIMD

МКОД, MISD

МКМД, MIMD



Основные характеристики ЭВМ

- **Эффективность**
- **Производительность**
- **Надежность**
- **Стоимость**
- **Энергопотребление**

Общий коэффициент эффективности

$$Э := \frac{P}{C_{ЭВМ} + C_{эксплуатации}}$$

Э - Общий коэффициент эффективности,
P - Производительность,
$C_{ЭВМ}$ - Стоимость ЭВМ,
$C_{Эксплуатации}$ - Стоимость эксплуатации.

$$Э' := \frac{P}{C_{ЭВМ}}$$

$$C_{ЭВМ} >> C_{Эксплуатации}$$

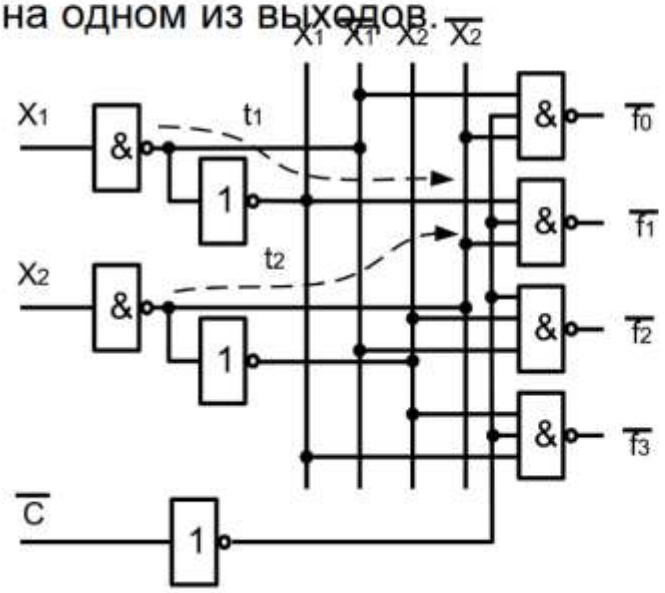
$$Э := \frac{P \cdot K_{н}}{C_{ЭВМ} + C_{Эксплуатации}}$$

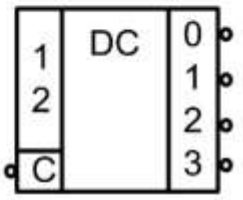
Э' - Эффективность без учета эксплуатационных издержек.
Эн - Эффективность с учетом эксплуатационной надежности.

Мультиплексоры и дешифраторы (2.19-2.21)

Дешифраторы

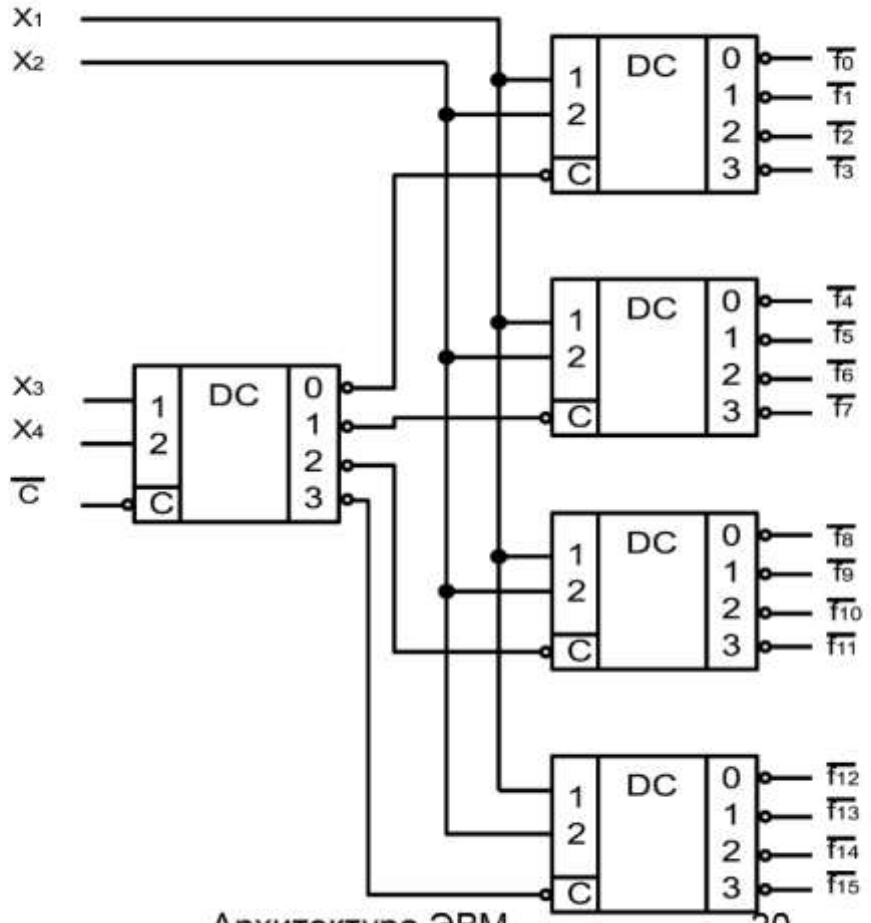
Дешифратором называется комбинационная схема, преобразующая код, подаваемый на входы, в сигнал на одном из выходов.





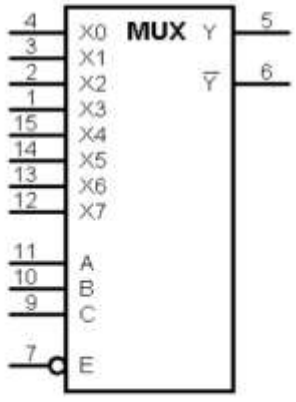
Статический риск: кратковременное изменение сигнала, который должен остаться неизменным.
Динамический риск: многократное переключение элемента вместо ожидаемого однократно.

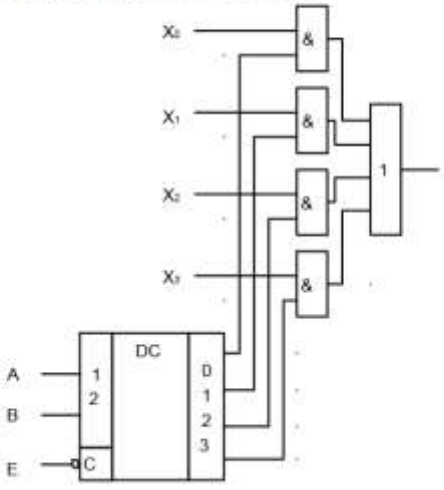
Наращивание размерности дешифраторов



Мультиплексоры

Мультиплексором называется комбинационная схема, осуществляющая передачу сигнала с одной из входных информационных линий на выход.





Счётчики (2.11-2.14)

Счетчики

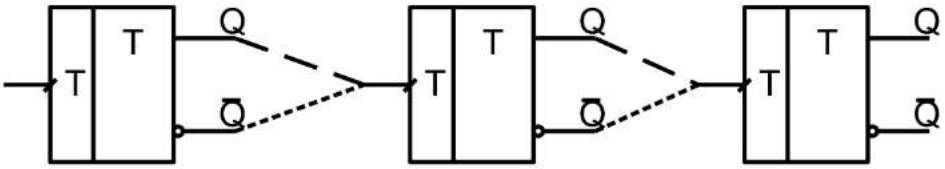
Счетчиком называется узел ЭВМ, предназначенный для подсчета входных сигналов.
Модуль счета: число возможных состояний счетчика.

Классификация счетчиков.
По способу счета: суммирующие, вычитающие, реверсивные.
По модулю счета: двоичные, десятичные, ….
По способу распространения переноса: с параллельным переносом, с последовательным переносом, с групповой структурой.
По способу синхронизации: асинхронные, синхронные.
По режиму работы: для подсчета входных сигналов, для деления частоты.

Таблица состояний

$X_{сч}$	Q4	Q3	Q2	Q1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
		...		перенос
15	1	1	1	1

— — Обратный счет



------- Прямой счет

Счетчик с последовательным переносом

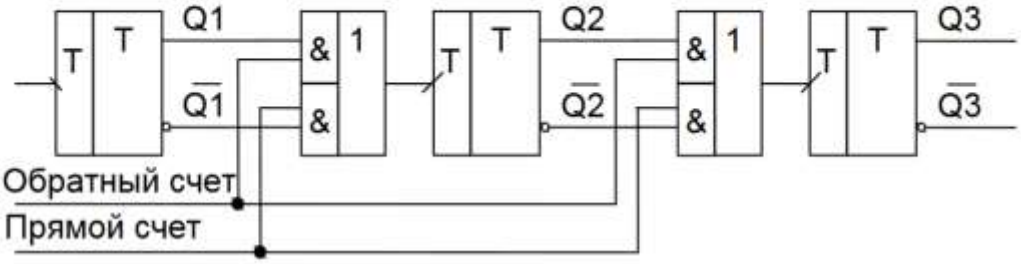
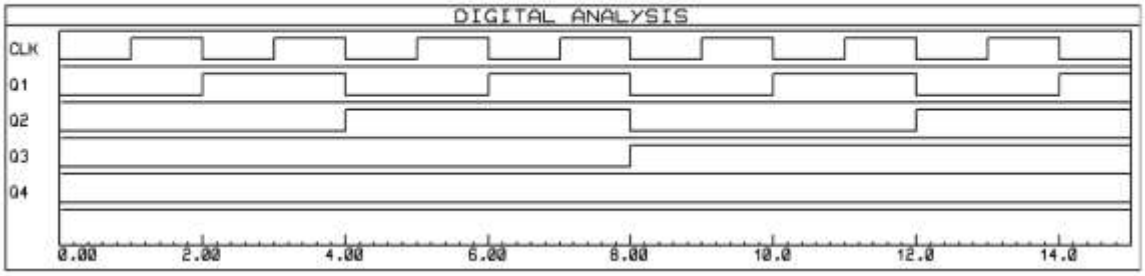


Диаграмма работы (прямой счет)



Счетчик с параллельным переносом

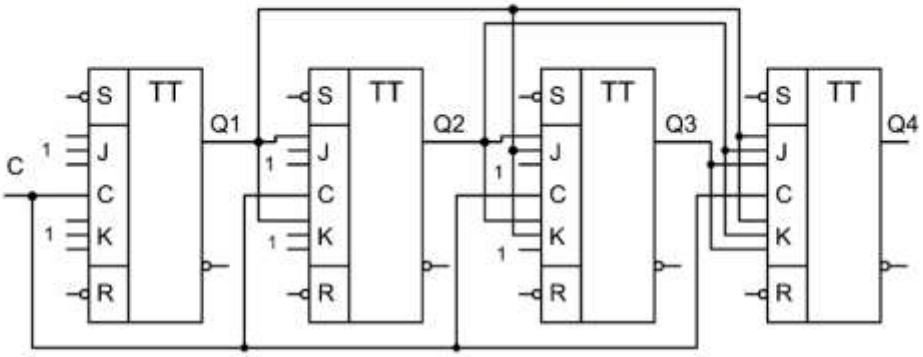
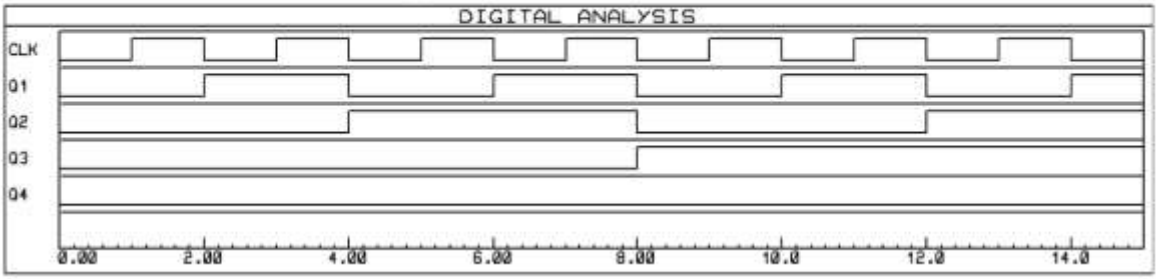


Диаграмма работы



Триггеры (RS, T, D, JK) (2.1-2.5)

III. Элементы и узлы ЭВМ

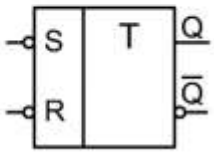
Триггеры

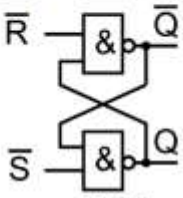
Триггер – логический элемент, который может находиться в одном из двух устойчивых состояний.

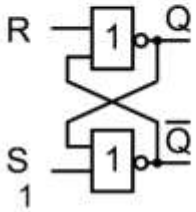
S, J – входы установки триггера в «1».
R, K – входы установки триггера в «0».
T – счетный вход триггера.
D – информационный вход триггера D
C – вход синхронизации
Q – прямой выход триггера
$\overline{Q}$ – инверсный выход триггера

Триггеры
- по логике:
RS, D, T, JK
- по способу приема:
Асинхронные,
Синхронные,
Одноступенчатые,
Двухступенчатые

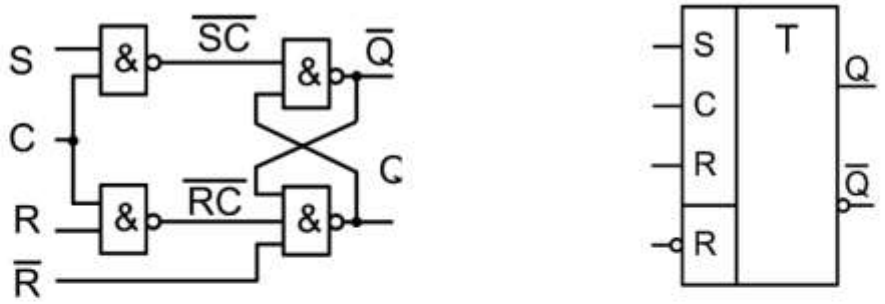
Одноступенчатый асинхронный RS-триггер



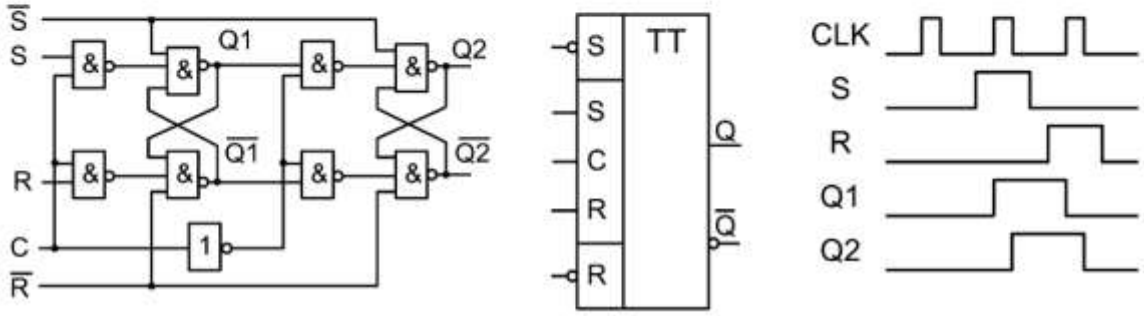




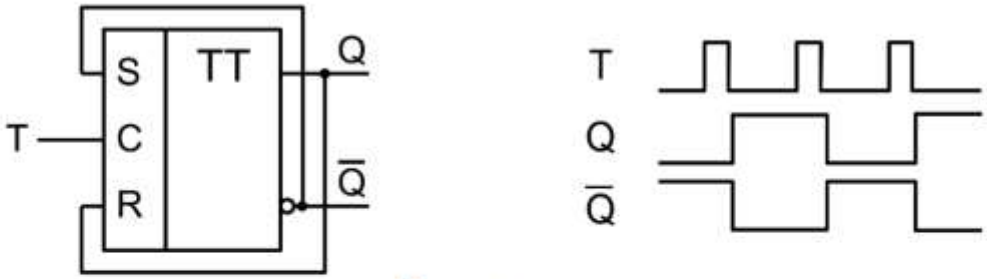
Одноступенчатый синхронный RS-триггер



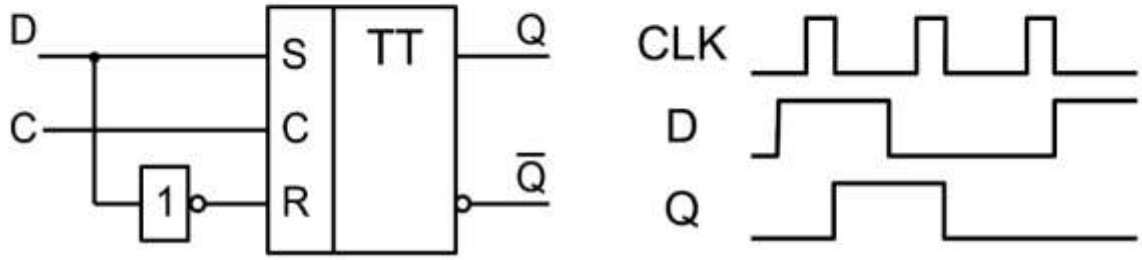
Двухступенчатый синхронный RS-триггер



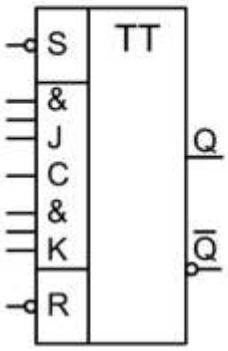
T-триггер

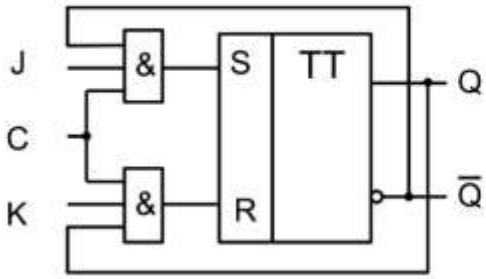


D-триггер



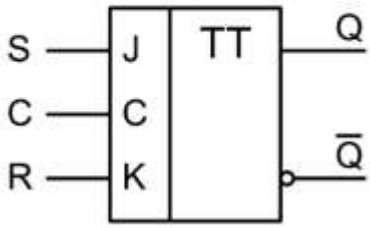
JK-триггер



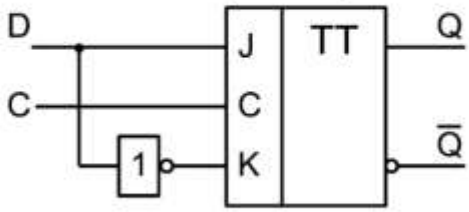


J(t)	K(t)	Q(t+1)	Режим
0	0	Q(t)	Хранение
0	1	0	Установка «0»
1	0	1	Установка «1»
1	1	$\overline{Q(t)}$	Инверсия

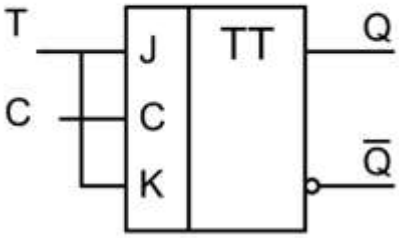
RS-триггер на основе JK-триггера



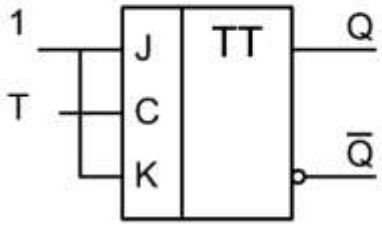
D-триггер на основе JK-триггера



Синхронный T-триггер на основе JK-триггера



Асинхронный T-триггер на основе JK-триггера

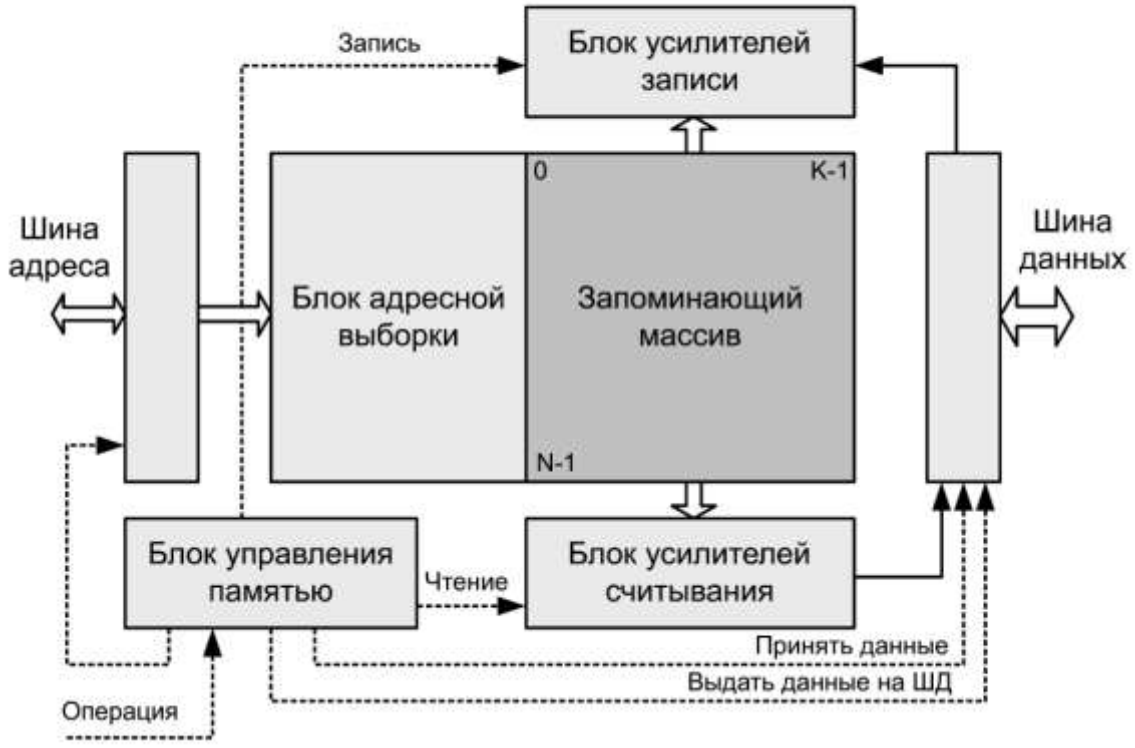


Методы организации доступа в запоминающие устройства (адресная, стековая и ассоциативная организации доступа) (3.2,3.4-3.6)

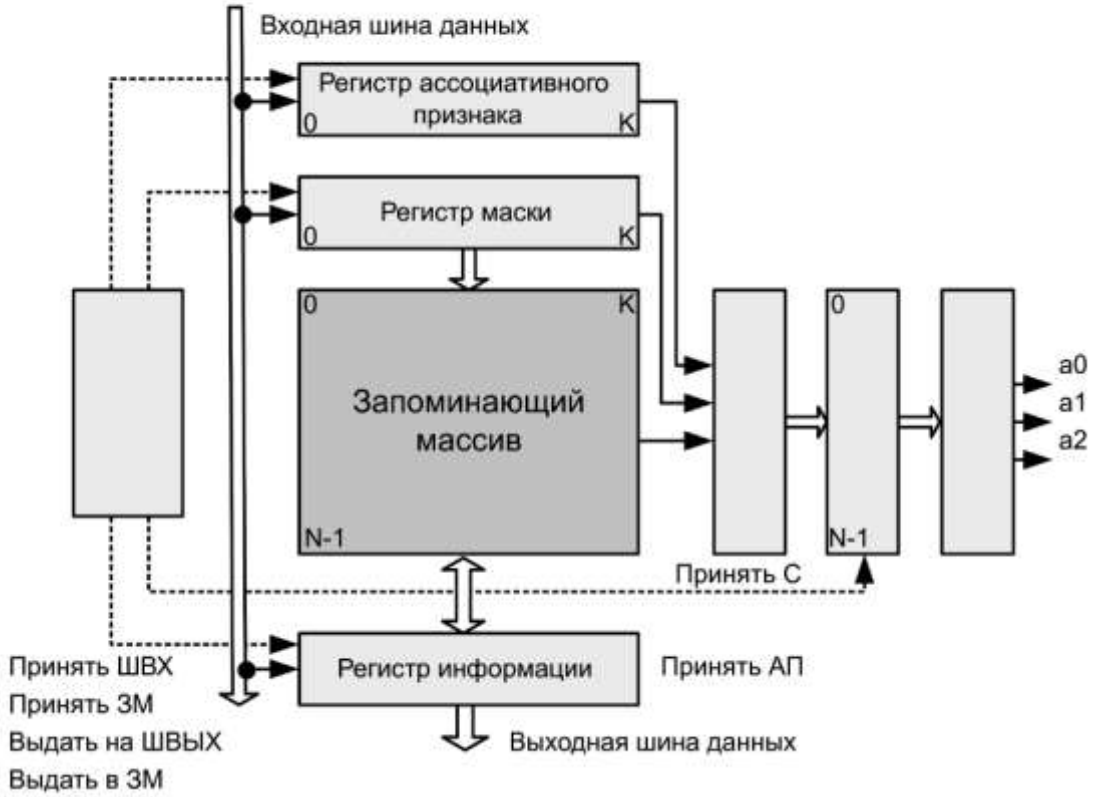
Классификация запоминающих устройств по способу доступа.

- Адресные ЗУ
  - Постоянные ЗУ, ПЗУ (ROM)
  - ЗУ с произвольным доступом (RAM)
- Ассоциативные ЗУ
  - Полностью ассоциативные ЗУ
  - Ассоциативные ЗУ с прямым размещением
  - Наборно-ассоциативные ЗУ
- Последовательные ЗУ
  - FIFO
  - LIFO
  - Файловые
  - Циклические

Обобщенная схема адресного ЗУ

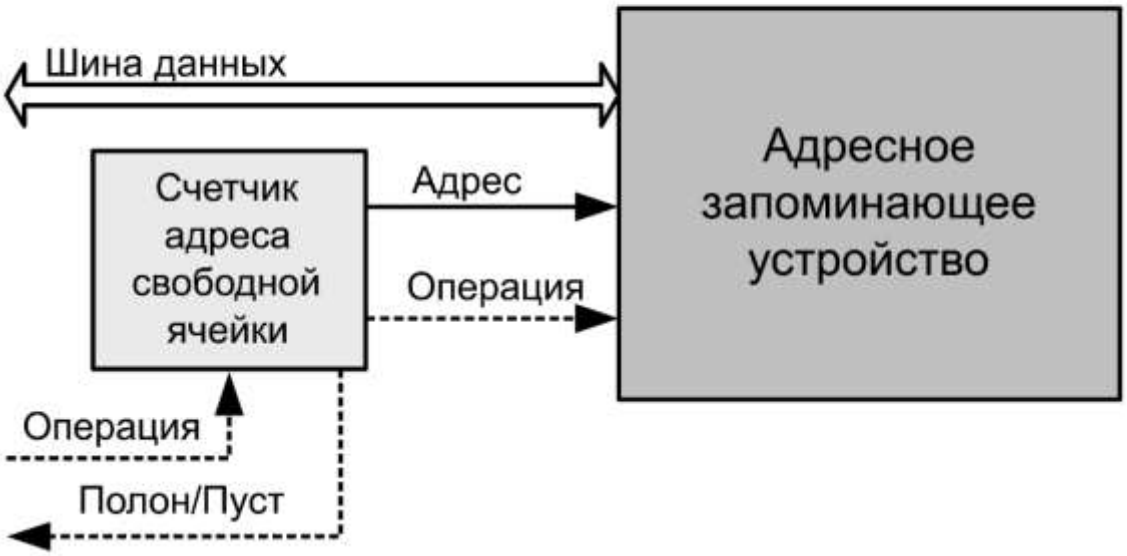


Обобщенная схема ассоциативного ЗУ



Обобщенная схема последовательного ЗУ

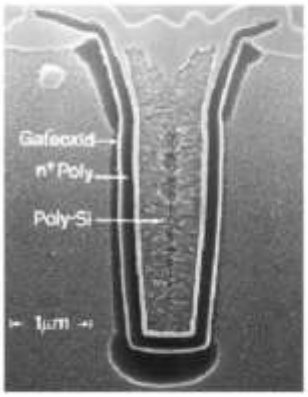
Стек (память типа LIFO)



Динамические запоминающие устройства с произвольной выборкой. ЗЯ динамической памяти. (3.27-3.29)

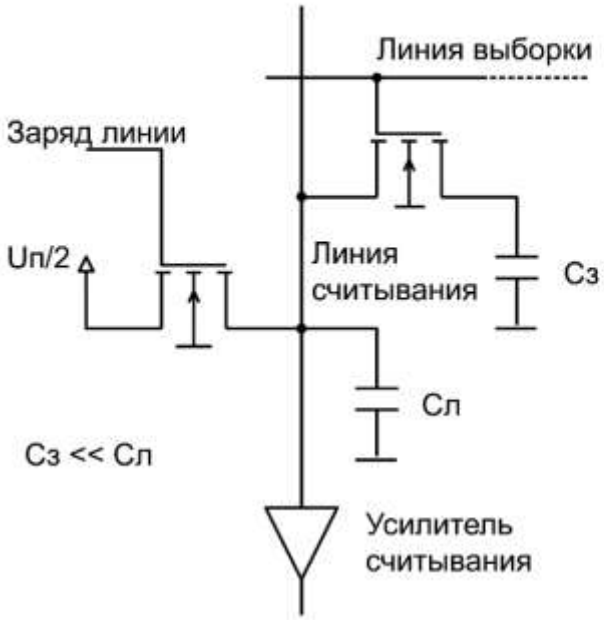
Динамические ЗУ с произвольной выборкой (DRAM)

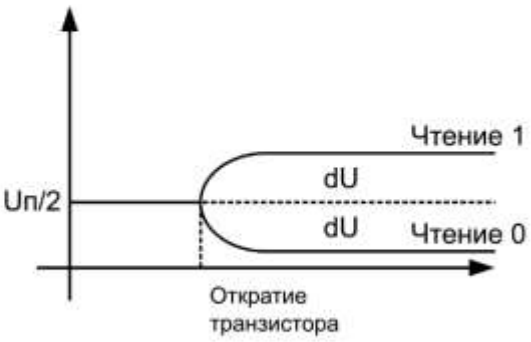
DRAM для обращения по произвольным адресам
DRAM, RLDRAM
DRAM, оптимизированные для обращения по последовательным адресам:
FPM DRAM, EDO DRAM, BEDO DRAM, SDRAM, DDR SDRAM, RDRAM



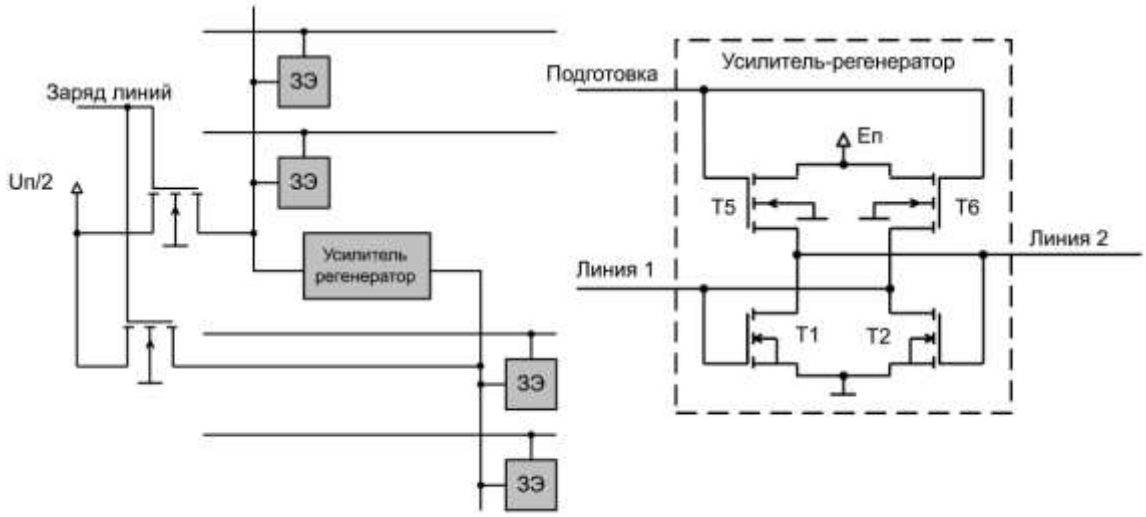
При выборке строки все Сз подключаются к линиям считывания.
После считывания необходимо произвести обратную запись информации – регенерацию.

Процесс считывания в DRAM





Принцип действия усилителя-регенератора



Диаграммы работы DRAM,FPM DRAM,SDRAM,DDR SDRAM (3.34-3.37)

Диаграмма работы FPM DRAM памяти

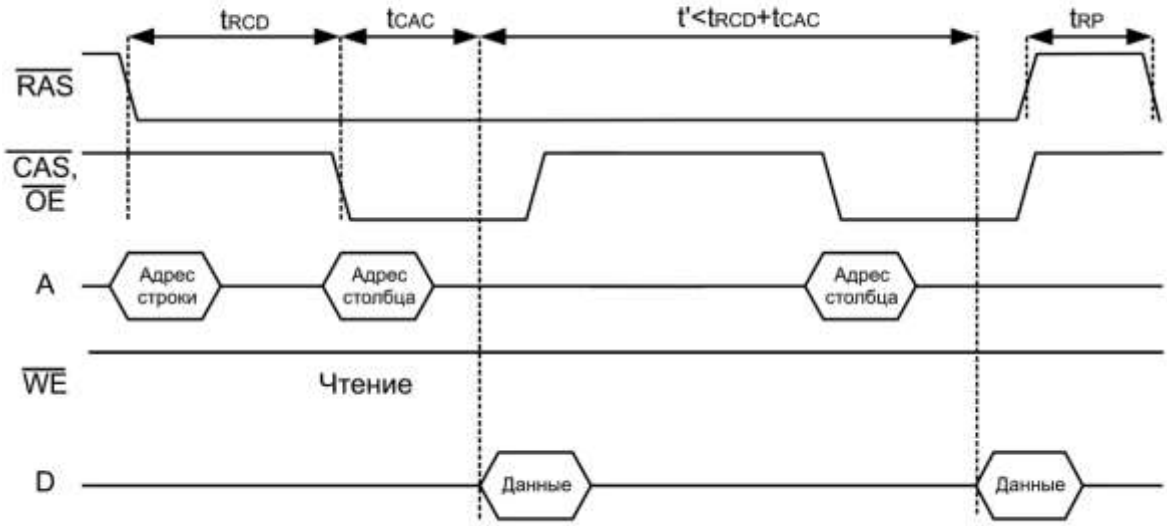


Диаграмма работы BEDO DRAM памяти

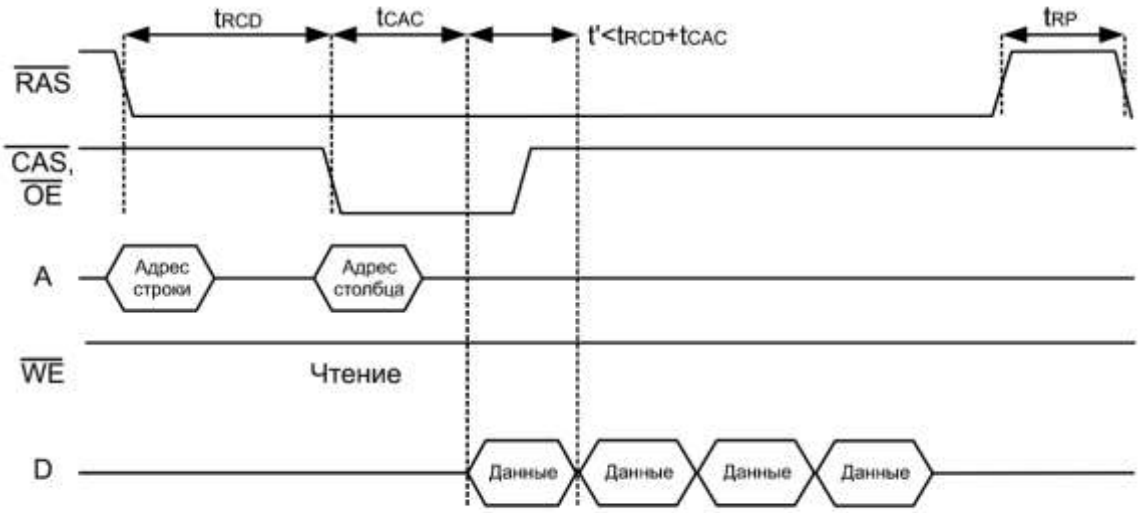


Диаграмма работы SDRAM памяти

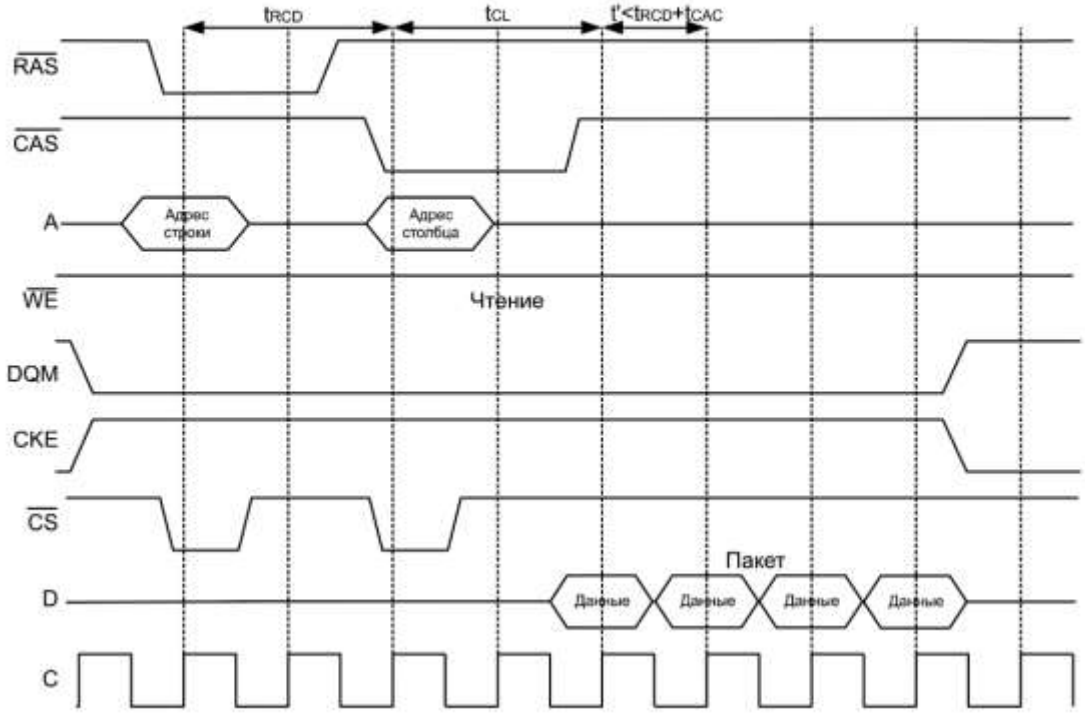
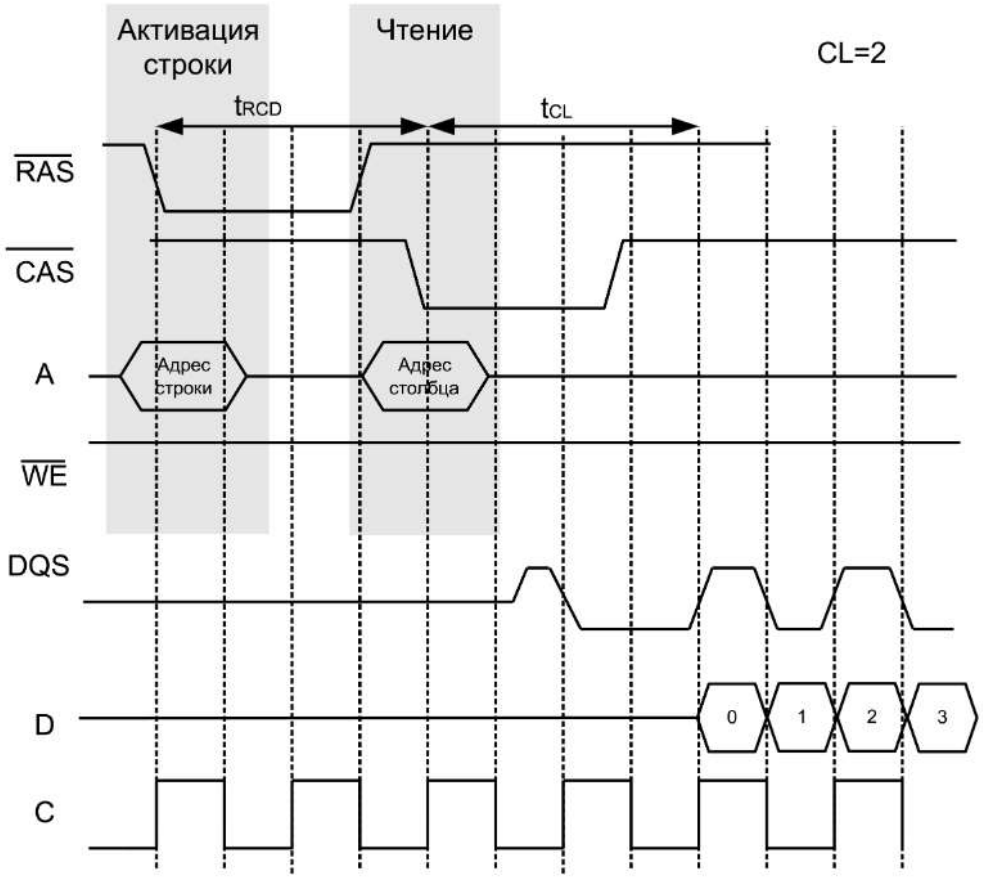


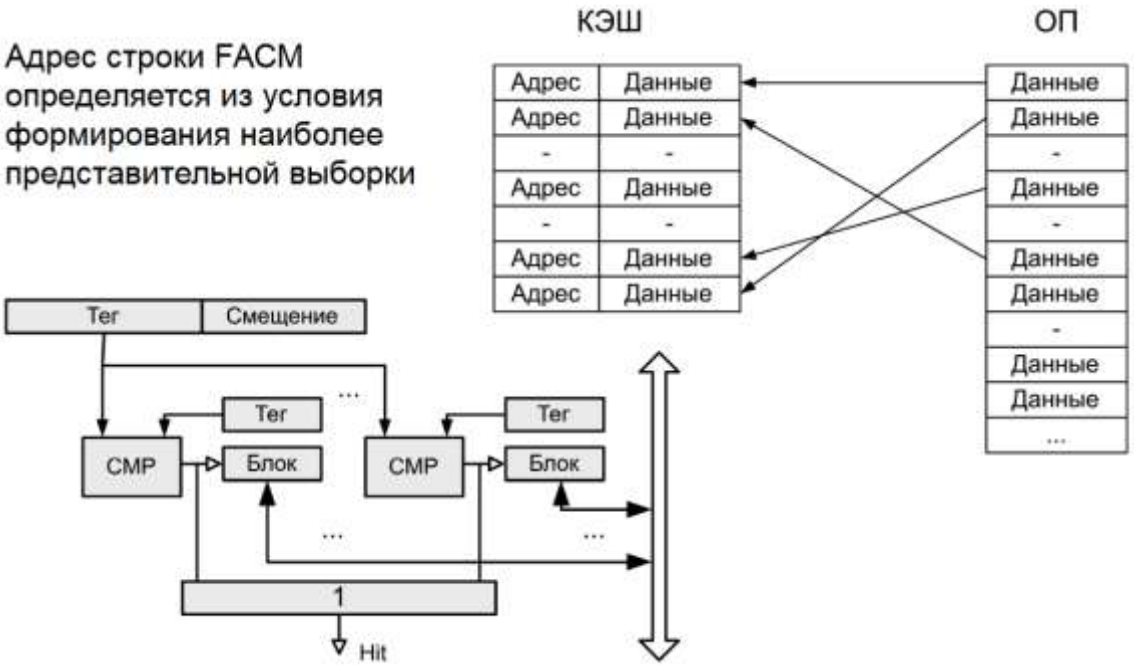
Диаграмма работы DDR SDRAM памяти



Кэш с произвольной загрузкой, прямым размещением и наборно-ассоциативный кэш (3.78-3.80)

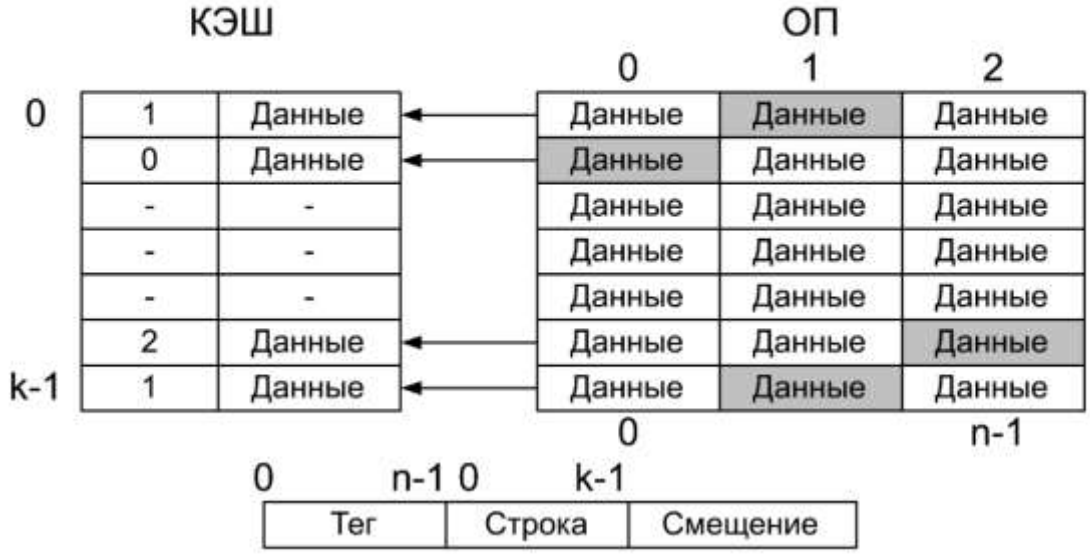
Произвольная загрузка
(Fully associated cache memory, FACM).

Адрес строки FACM определяется из условия формирования наиболее представительной выборки

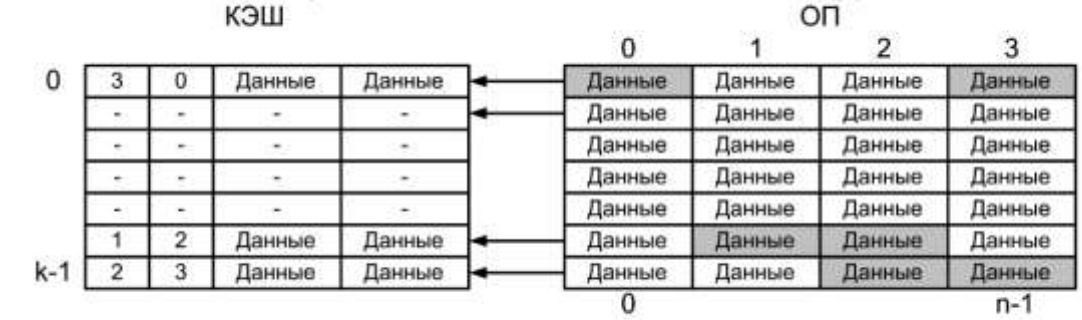


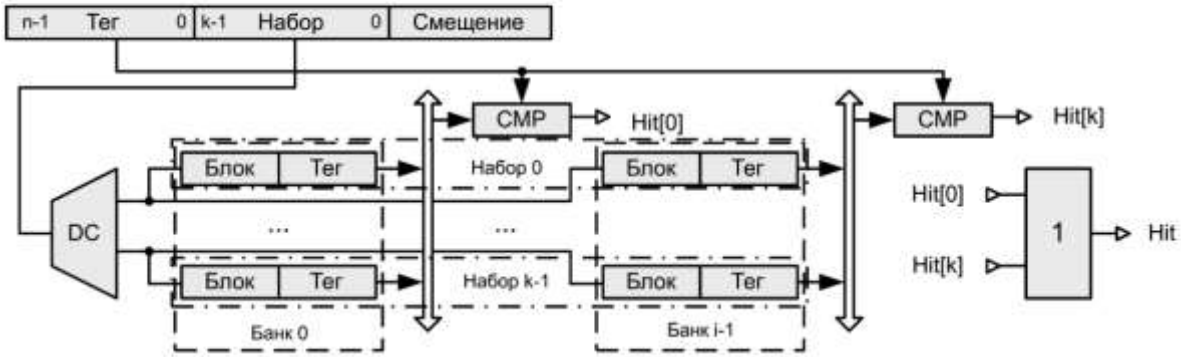
Прямое размещение.

Адрес строки однозначно определяется по тегу (i = t mod k).



Наборно-ассоциативная кэш-память
(Set associated cache memory)





Страничная, сегментная и сегментно-страничная организация виртуальной памяти (3.83-3.87)

Виртуальная память

Механизм виртуализации адресного пространства позволяет:

-Увеличить объем адресуемой памяти.

-Использовать физическую память различного объема.

-Возложить на аппаратную составляющую механизмы доступа к ВЗУ

-Сгладить разрыв в производительности ОП и ВЗУ.

-Ускоряет доступ к данным по последовательным адресам.

-Способствует реализации защиты памяти.

Виртуальные системы строятся по трем принципам:

-Системы с блоками различного размера (сегментная организация).

-Системы с блоками одинакового размера (страничная организация).

-Смешанные системы (сегментно-страничная организация).

Страничная организация

Программа отображается в память равными блоками – страницами. Преобразование логического адреса в физический осуществляется с помощью таблицы страниц.

Преобразование логического адреса в физический реализуется в устройстве управления памятью (Memory Manage Unit), который определяет, находится ли страница в физической памяти (попадение).

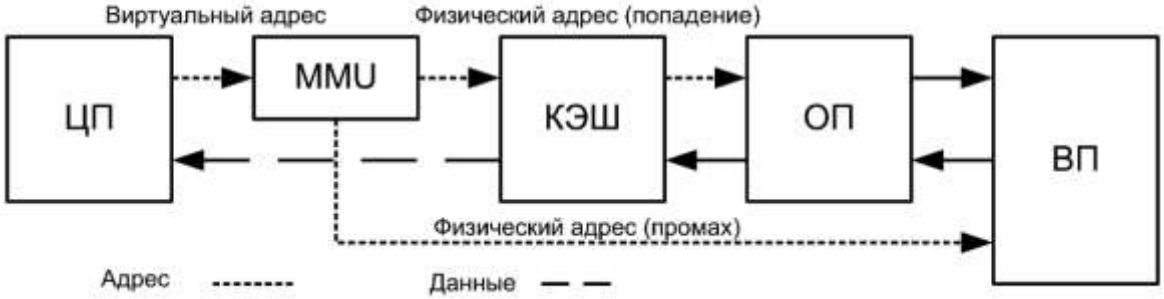
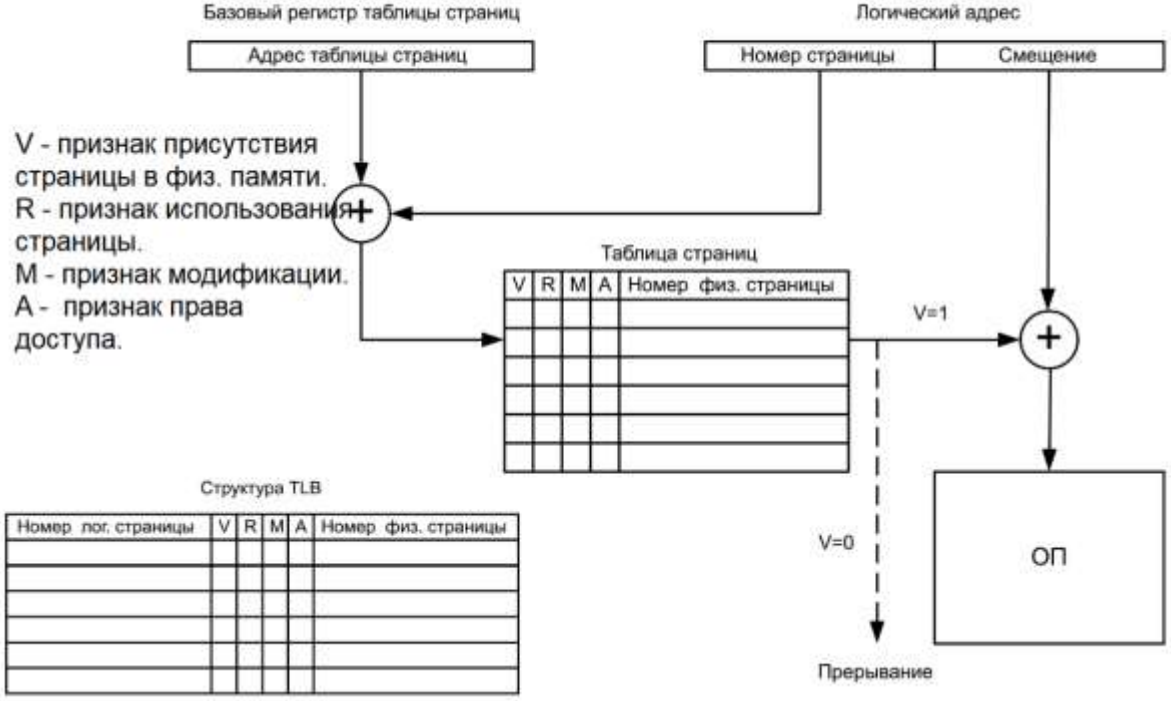
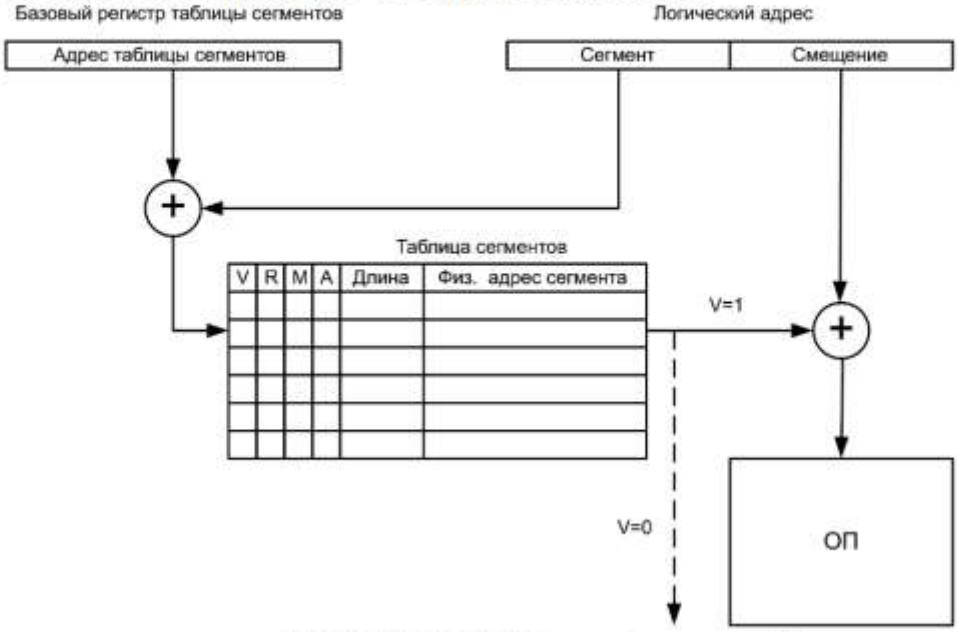


Схема страничного преобразования



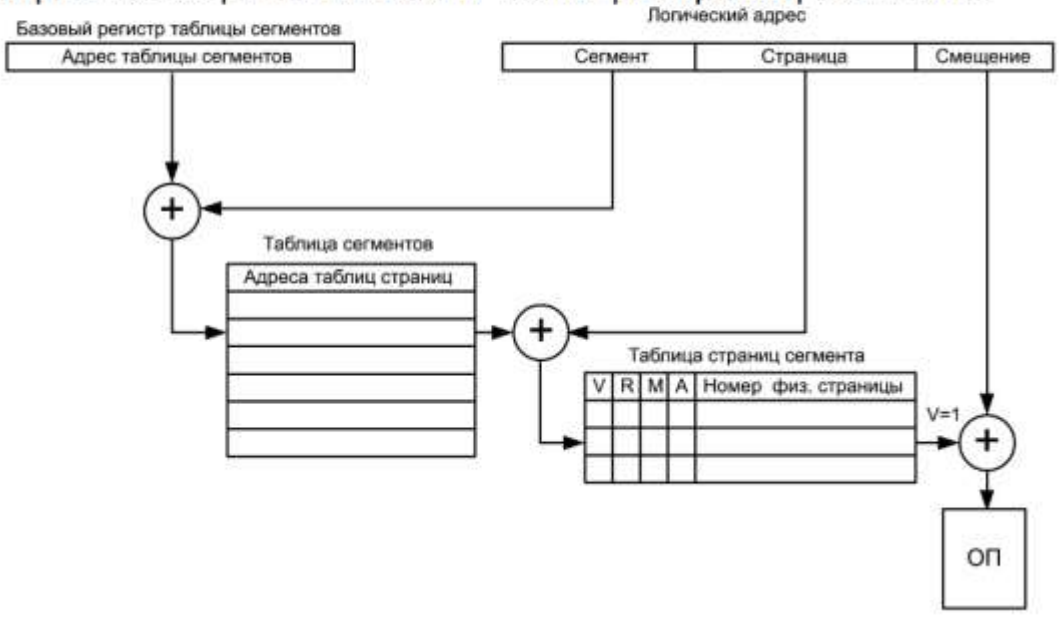
Сегментная организация

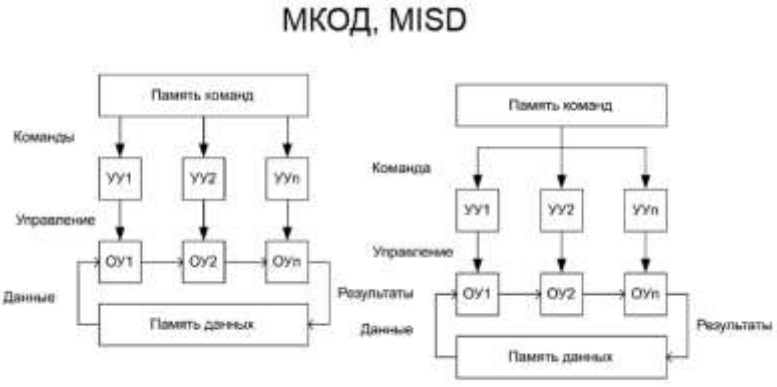
Программа отображается в память блоками различного размера –сегментами. Преобразование логического адреса в физический осуществляется с помощью таблицы сегментов.

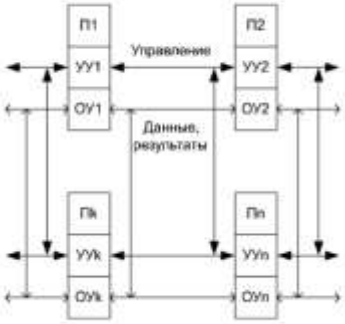


Сегментно-страничная организация памяти

Программа отображается в память блоками различного размера –сегментами, каждый из которых целое число страниц. Преобразование логического адреса в физический осуществляе помощью таблицы сегментов и таблицы страниц сегмента.







I.Принципы построения и архитектура ЭВМ

- Общие принципы построения современных ЭВМ.
- Основные тенденции развития ЭВМ.
- Классификация архитектур системы команд (СК).
- RISC, CISC, VLIW архитектура.
- Типы команд.
- Форматы команд.
- Способы адресации.

Общие принципы построения современных ЭВМ

Принципы Фон-Неймана

-Двоичное кодирование информации
-Программное управление
-Адресность памяти
-Однородность памяти

ОКОД, SISD

-Гарвардская архитектура (ОП для хранения команд и ОП для хранения данных)
Принстонская архитектура (ОП для хранения команд и данных)

Структура современных ЭВМ с архитектурой Фон-Неймана

-Центральное процессорное устройство (ЦПУ).
- Арифметико-логическое устройство (АЛУ)
-Устройство управления (УУ)
-Регистры общего назначения (РОН)
-Основная память
-Система ввода-вывода
-Внешние устройства
-Внешняя память
-Система передачи информации
-Система синхронизации
-Система прерываний
-Система прямого доступа к памяти
-Система подвода питания/земли и система энергосбережения
-Система повышения отказоустойчивости

Компьютеры с «не Фон-Неймовской» архитектурой

Нейрокомпьютеры — устройство переработки информации на основе принципов работы естественных нейронных систем.

Когнитивный компьютинг — вычислительная технология, основанная на имитации процесса познания на нейросинаптических структурах

Компьютеры, управляются потоком данных - выполнение каждой операции производится при готовности всех её операндов, при этом последовательность выполнения команд заранее не задаётся

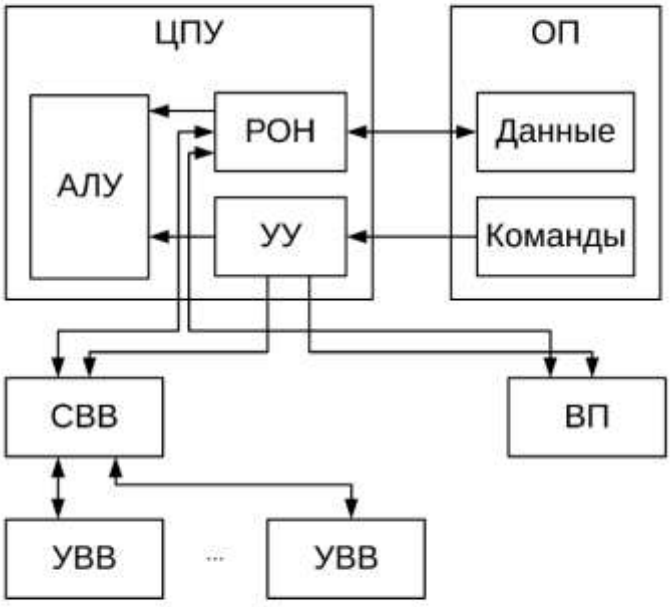
Квантовые компьютеры - вычислительное устройство, работающее на основе квантовой механики. Квантовый компьютер принципиально отличается от классических компьютеров, работающих на основе классической механики.

TrueNorth neurosynaptic computer chip

TrueNorth chip (08.2014):
- Не Фон-Неймановская архитектура
- 5.4 миллиарда транзисторов
- 4,096 нейросинаптических ядра
- Миллион нейронов и 256 миллионов синапсов (связей между нейронами)
- Произведен по технологии 28nm
- Потребляет 70mW

ЭВМ с непосредственными связями и магистральной структурой. Основные тенденции развития ЭВМ (4.5,4.6,4.9)

ЭВМ с непосредственными связями



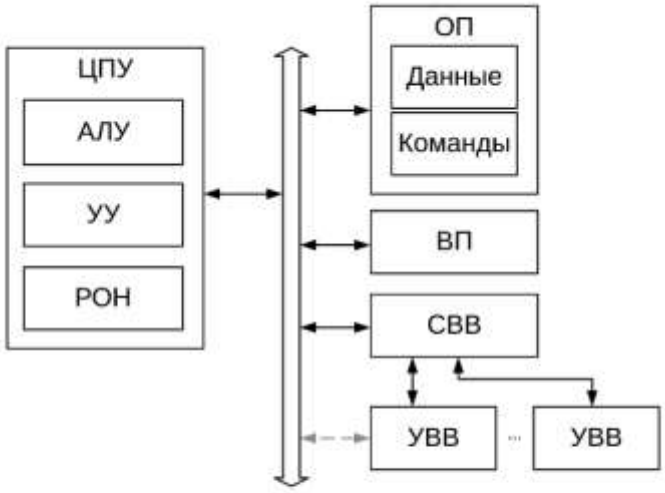
(+) При построении оптимальных линий связи вычислительная машина обладает максимальным быстродействием.

(-) Ограничение на количество выводов микросхем не позволяет организовать широкие шины.

(-) Канал между ОП и ЦПУ является узким местом.

(-) Реконфигурация системы требует изменения характеристик линий связи.

ЭВМ с магистральной структурой



(+) Общая шина позволяет легко реконфигурировать систему.

(-) Шина является узким местом.

- Шина, используемая всеми устройствами системы для передачи данных называется системной.
- Для разгрузки системной шины используют иерархию шин.
- По назначению, разделяют шины адреса, шины данных и шины управления.

Архитектура системы команд

В команде указывается, какую операцию выполнять (КОП), над какими операндами выполнять операцию, а также куда поместить операнд.

RISC – Reduced Instruction Set Computer; CISC – Complex Instruction Set Computer; VLIW – Very Long Instruction Word; ROSC - Removed Operand Set Computer

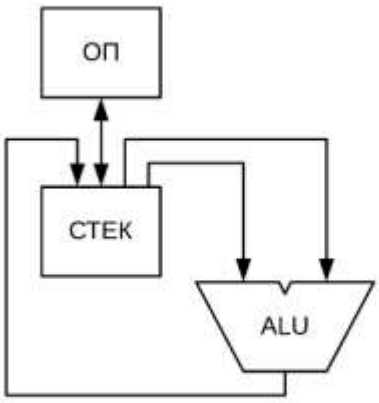
RISC, CISC, VLIW архитектура(4.10-4.11)

Сравнение CISC, RISC и VLIW архитектур СК

Характеристика	CISC	RISC	VLIW
Длина команды	Различная	Одинаковая	Одинаковая
Расположение полей в командах	Различное	Одинаковое	Одинаковое
Количество регистров	Малое. Регистры специализи-рованные	Большое. Регистры универсальные	Большое. Регистры универсальные
Доступ к памяти	Кодируется в команде. Выполняется по микрокоманде	Выполняется по специальной команде	Выполняется по специальной команде
Длительность выполнения команд	Различная	Одинаковая (для большинства команд)	Различная

Стековая архитектура СК

(+) При размещении операндов в стековой памяти (LIFO) архитектура команд упрощается (большое количество действий выполняется аппаратно)



Операции:
- занесение в стек (PUSH);
- извлечение из стека (POP);
- выполнение действий на стеком (извлечение операндов из вершины стека, выполнение действий, помещение результата в вершину стека)

Для выполнение арифметических операций их преобразуют к постфиксной форме (Польской записи).

Пример: a = a + b * (c -d); Постфиксная форма: abcd-*+;
Действия: PUSH a; PUSH b; PUSH c; PUSH d; SUB; MUL; ADD; POP a.

(-) Отсутствие прямого доступа к памяти ограничивает область применения.
(-) Сложность организации параллельной обработки.

Назначение и обобщенная структура процессорного устройства. Микропроцессор. Классификация микропроцессорных СБИС (5.4-5.5)

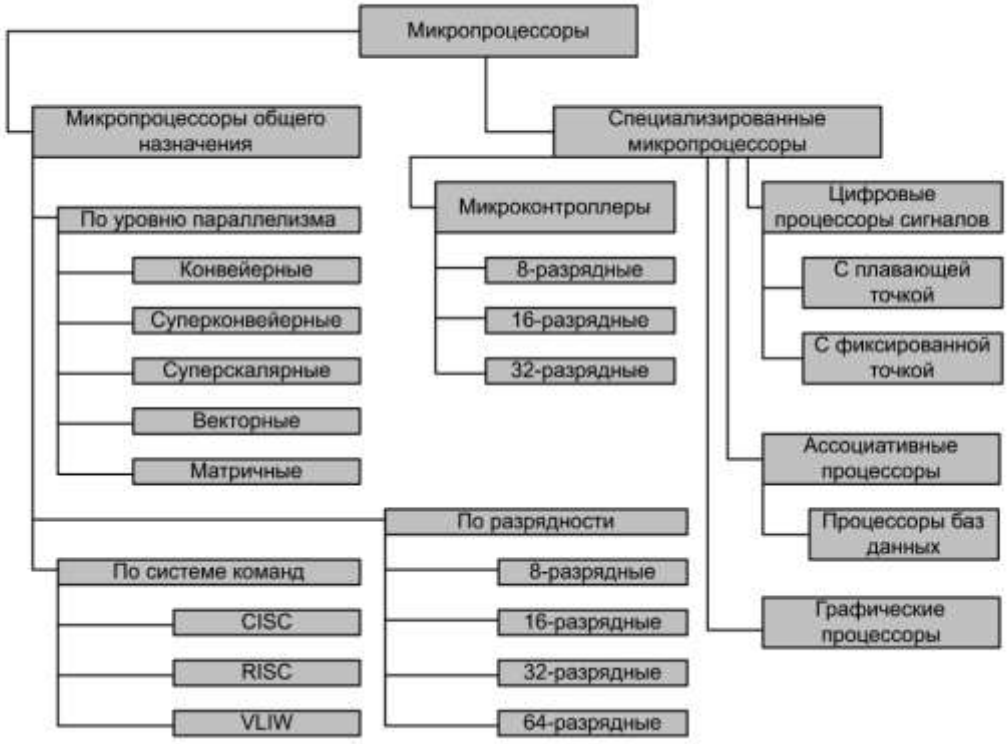
Процессором (процессорным ядром) называется устройство ЭВМ, непосредственно осуществляющее процесс переработки информации и управление им в соответствии с заданным алгоритмом, который, как правило, представлен программой.

ЭВМ может содержать несколько процессоров. Процессор, управляющий вычислительным процессом, называется центральным.

Микропроцессором называется функционально законченное устройство, представляющее собой вариант процессора (или нескольких процессорных ядер) современной ЭВМ и реализованное в виде одной или нескольких СБИС.

Микропроцессорный комплект представляет собой совокупность микропроцессора и специализированных ИС, совместимых по временным, электрическим и конструктивным параметрам, совместное использование которых позволяет реализовать основные функции ЭВМ.

Классификация процессорных устройств



Обобщенная структура универсального процессорного устройства

Архитектурные особенности:
-Конвейерное исполнение команд.

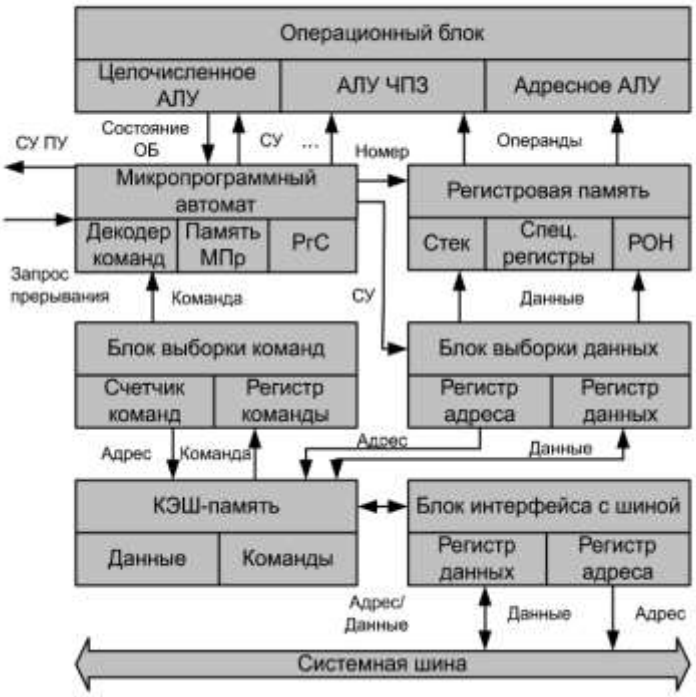
-Внутренняя КЭШ-память.

-Целочисленное АЛУ.

-Устройство выполнения операций над числами с плавающей запятой.

-Обработка прерываний от ПУ.

-Поддержка мультипроцессорной обработки.



Форматы команд. Типы команд.(4.18-4.20)

Типы команд.

- Команды пересылки данных.
  - регистр-регистр
  - регистр-память
  - память-память
- Команды арифметической и логической обработки (сложение, вычитание, умножение, деление, инкремент, декремент, сравнение, операции над ЧПЗ, логические операции, операции сдвига).
  Сдвиг: логический, арифметический, циклический, циклический через дополнительным разряд.
- Команды работы со строками (могут быть реализованы набором других команд, однако удобны при работе с символьной информацией).
- Команды векторной обработки (позволяет выполнять однотипные действия над большим количеством однородных данных). Пример арифметики с насыщением:

  1011 0111 1010
+ 0001 1001 1000
  1100 1111 1111
- Команды преобразования: служат для табличного преобразования данных из одной системы кодов в другую (2-10 <-> 2)

•Команды ввода/вывода. Служат для управления, проверки состояния и обмена данными с периферийными устройствами.

- Команды вывода в порт
- Команды ввода из порта.

•Команды управления потоком команд. Данные команды служат для указания очередности выполняемых команд.

Вычисление адреса очередной команды может выполняться несколькими способами:

- увеличением адреса на длину исполненной (естественный порядок).
- изменением адреса на длину следующей (перешагивание)
- изменением адреса на значение, указанное в текущей команде (короткий переход).
- непосредственное указание следующей команды (длинный переход).

Перечисленные команды могут выполняться лишь по некоторому условию (уловные переходы).

Команды условного перехода составляют 80% команд управления.

Команды безусловного перехода: вызовы и возвраты из процедур, и.т.д.

Форматы команд.

Операционная часть	Адресная часть

1. Четырехадресная команда.

КОП	1 операнд	2 операнд	результат	Адр след ком.

2. Трехадресная команда

КОП	1 операнд	2 операнд	результат

3. Двухадресная команда.

КОП	1 операнд	2 оп-д/результат

Характерна для CISC-архитектуры

4. Аккумуляторная архитектура

КОП	1 операнд

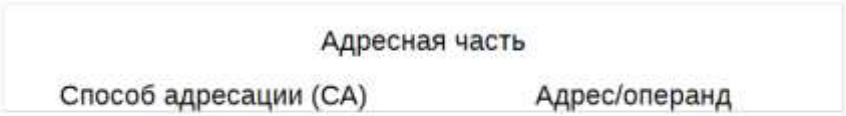
Второй операнд хранится в аккумуляторе. Данный формат команд характерен для RISC-архитектур.

5. Нульоперандная команда.

КОП

Способы адресации: непосредственная, прямая, регистровая, неявная, косвенная, косвенная регистровая (4.21-4.26)

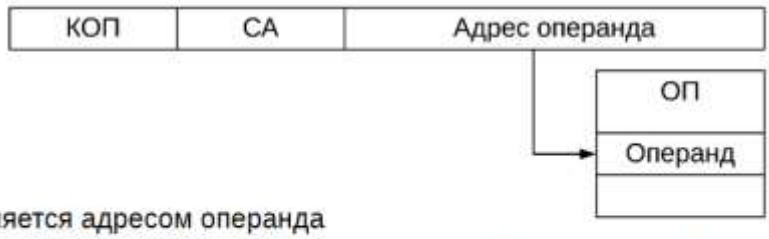
Способы адресации



Непосредственная адресация

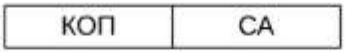
Вместо адреса команда содержит непосредственно операнд.
(+) команда выполняется быстро
(-) непосредственный операнд может не войти в команду

Прямая адресация



Адрес в команде является адресом операнда
(+) если операнд находится в памяти, то это самый быстрый способ указать на него
(-) заранее определенный адрес влияет на переносимость программы.
(-) Адрес занимает много места

Неявная адресация

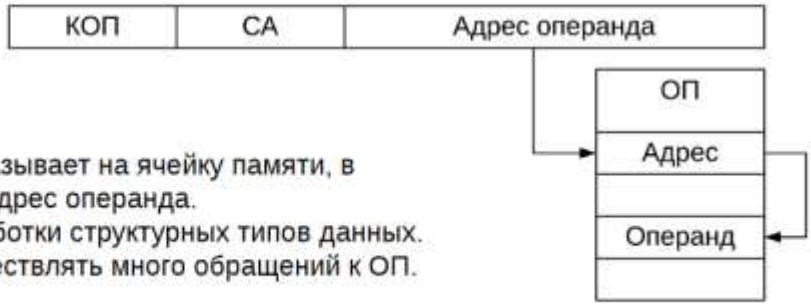


Операнд подразумевается (следует из КОП).
(+) Команда занимает мало места
(-) только такие командах нельзя использовать для построение всей системы команд.

Регистровая адресация

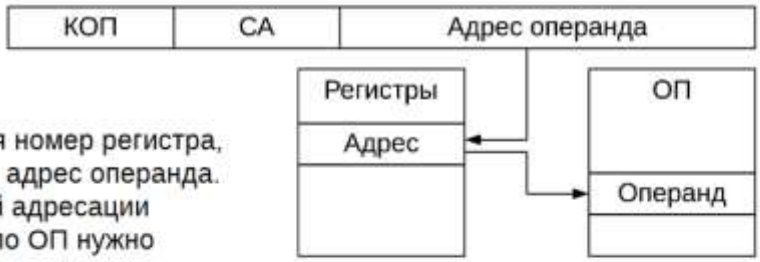
Адрес в команде указывает не на ячейку ОП, а на регистр.
(+) Быстрее прямой адресации
(-) Количество регистров ограничено

Косвенная адресация



Адрес в команде указывает на ячейку памяти, в которой находится адрес операнда.
(+) удобна для обработки структурных типов данных.
(-) приходится осуществлять много обращений к ОП.

Косвенная регистровая адресация

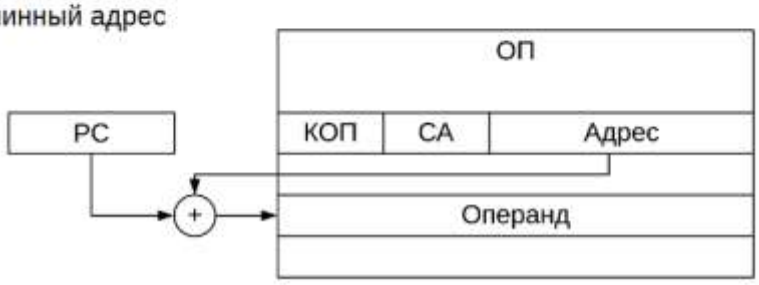


В команде содержится номер регистра, в котором содержится адрес операнда.
(+) быстрее косвенной адресации
(-) для перемещения по ОП нужно менять содержимое регистра

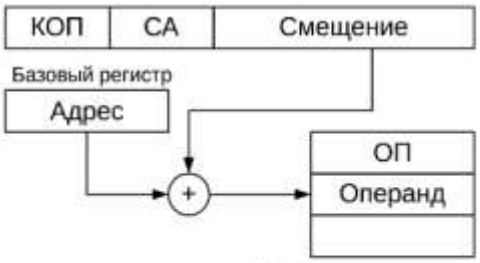
Относительная адресация

Адрес вычисляется относительно счётчика команд

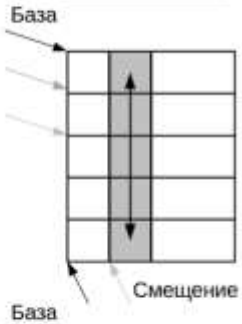
(+) Код переносим, команды занимают мало места
(-) Может понадобиться длинный адрес



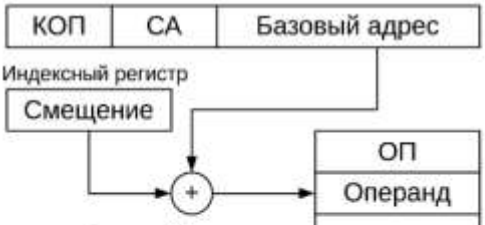
Базовая регистровая адресация

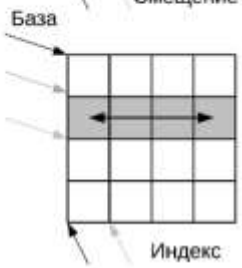


Адрес в команде представляет собой смещение, которое складывается со значением в базовом регистре для получения адреса операнда
(+) Удобна для работы со структурами данных, размещаемых динамически.
(-) Переносимость меньше, чем у относительной адресации



Индексная регистровая адресация





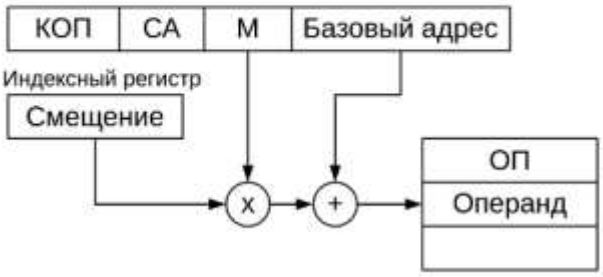
В поле адреса команды содержится базовый адрес, складываемый со значением смещения в индексном регистре.
(+) Удобна для работы со структурами данных, размещаемых динамически.
(-) Переносимость меньше, чем у относительной адресации

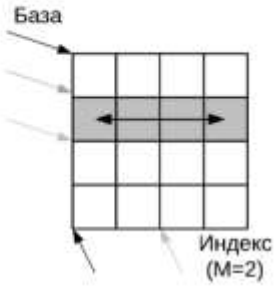
Автоинкрементная/автодекрементная адресация

Разновидность регистровой индексной или базовой адресации. До или после выполнения команды значение базового или индексного регистра увеличивается/уменьшается на единицу.
(+) Способ адресации удобен для команд обработки строк.
(-) Автоматическое изменение часто требуется выполнять на величину, большую единицы.

Индексная адресация с масштабированием

Индексный регистр умножается на масштаб M и суммируется с базовым адресом из команды.
(+) Удобен для модификации адреса на величину M.
(-) Вычисление адреса замедляется, т.к. требуется выполнять умножение.

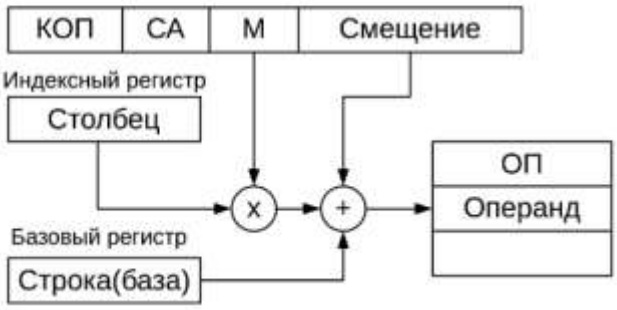




Базовая индексная адресация с масштабированием

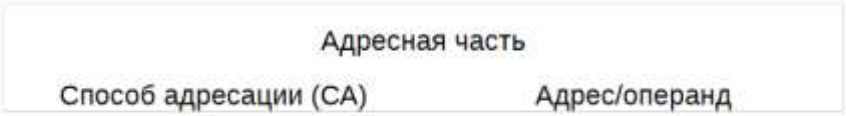
Адрес определяется по формуле Адрес=Индекс*Масштаб+База+Смещение.

(+) Базовая индексная адресация с масштабированием часто используется при обращении к системным таблицам, находящимся в ОП (таблица дескрипторов, таблицы страниц, таблица векторов прерываний и т.д.)
(-) Ограниченное на величину M (M=1,2,4,8).



Способы адресации со смещением: относительная, базовая регистровая, индексная, автоинкрементная и автодекрементная, индексная с масштабированием (4.21-4.26)

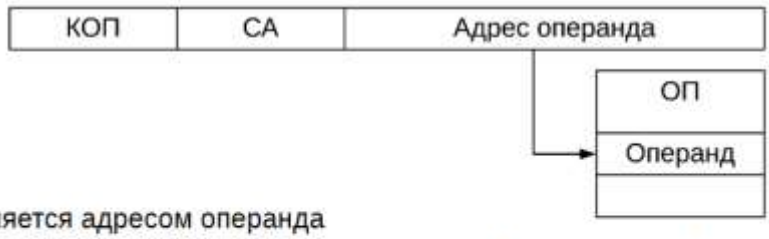
Способы адресации



Непосредственная адресация

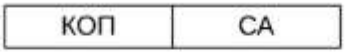
Вместо адреса команда содержит непосредственно операнд.
(+) команда выполняется быстро
(-) непосредственный операнд может не войти в команду

Прямая адресация



Адрес в команде является адресом операнда
(+) если операнд находится в памяти, то это самый быстрый способ указать на него
(-) заранее определенный адрес влияет на переносимость программы.
(-) Адрес занимает много места

Неявная адресация

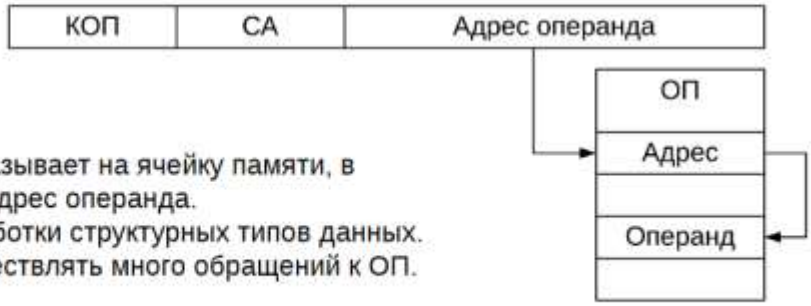


Операнд подразумевается (следует из КОП).
(+) Команда занимает мало места
(-) только такие командах нельзя использовать для построение всей системы команд.

Регистровая адресация

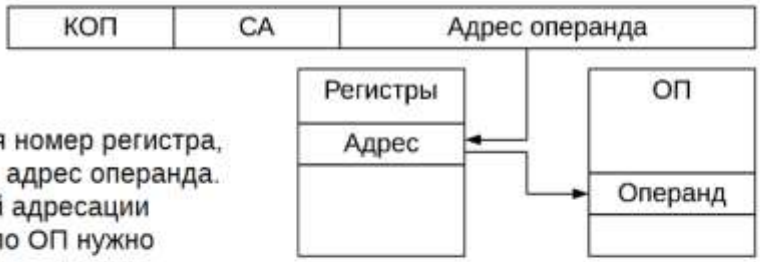
Адрес в команде указывает не на ячейку ОП, а на регистр.
(+) Быстрее прямой адресации
(-) Количество регистров ограничено

Косвенная адресация



Адрес в команде указывает на ячейку памяти, в которой находится адрес операнда.
(+) удобна для обработки структурных типов данных.
(-) приходится осуществлять много обращений к ОП.

Косвенная регистровая адресация

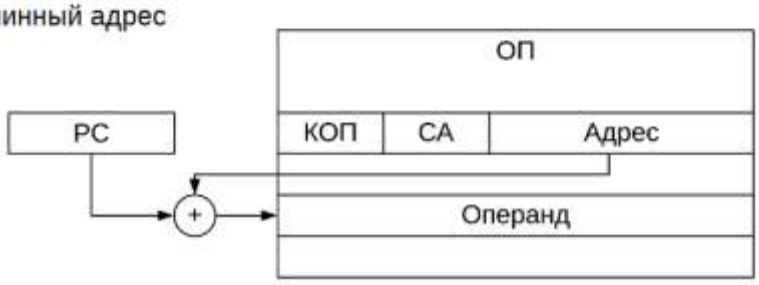


В команде содержится номер регистра, в котором содержится адрес операнда.
(+) быстрее косвенной адресации
(-) для перемещения по ОП нужно менять содержимое регистра

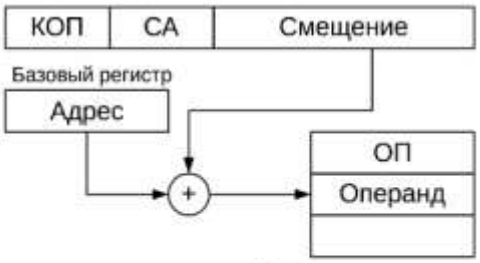
Относительная адресация

Адрес вычисляется относительно счётчика команд

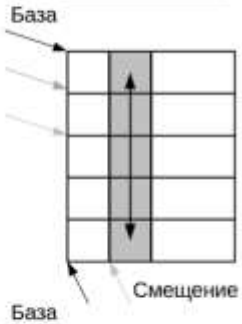
(+) Код переносим, команды занимают мало места
(-) Может понадобиться длинный адрес



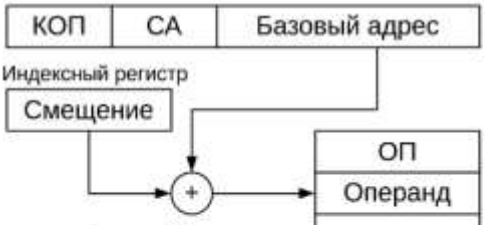
Базовая регистровая адресация

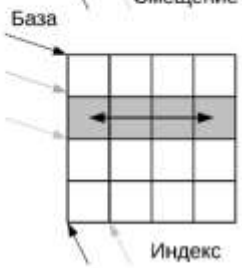


Адрес в команде представляет собой смещение, которое складывается со значением в базовом регистре для получения адреса операнда
(+) Удобна для работы со структурами данных, размещаемых динамически.
(-) Переносимость меньше, чем у относительной адресации



Индексная регистровая адресация





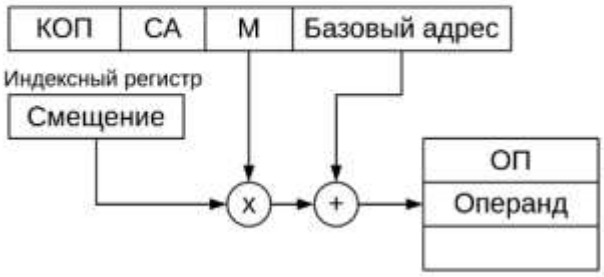
В поле адреса команды содержится базовый адрес, складываемый со значением смещения в индексном регистре.
(+) Удобна для работы со структурами данных, размещаемых динамически.
(-) Переносимость меньше, чем у относительной адресации

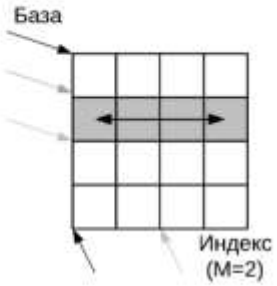
Автоинкрементная/автодекрементная адресация

Разновидность регистровой индексной или базовой адресации. До или после выполнения команды значение базового или индексного регистра увеличивается/уменьшается на единицу.
(+) Способ адресации удобен для команд обработки строк.
(-) Автоматическое изменение часто требуется выполнять на величину, большую единицы.

Индексная адресация с масштабированием

Индексный регистр умножается на масштаб M и суммируется с базовым адресом из команды.
(+) Удобен для модификации адреса на величину M.
(-) Вычисление адреса замедляется, т.к. требуется выполнять умножение.

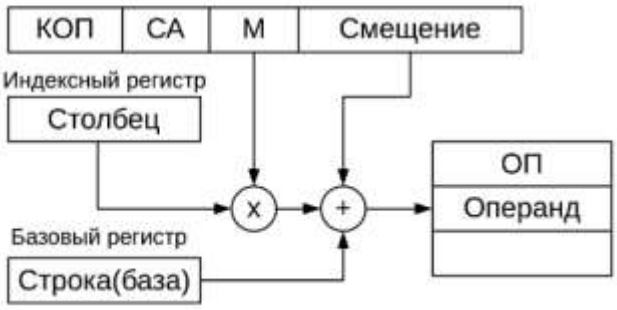




Базовая индексная адресация с масштабированием

Адрес определяется по формуле Адрес=Индекс*Масштаб+База+Смещение.

(+) Базовая индексная адресация с масштабированием часто используется при обращении к системным таблицам, находящимся в ОП (таблица дескрипторов, таблицы страниц, таблица векторов прерываний и т.д.)
(-) Ограниченное на величину M (M=1,2,4,8).



Архитектура конвейерного суперскалярного процессора. Проблема условных переходов (5.8,5.9,5.12)

Конфликты в конвейере (риски)

1. Структурный риск

   Команды одновременно обращаются к одному и тому же ресурсу (например, к ОП).

2. Риск по данным

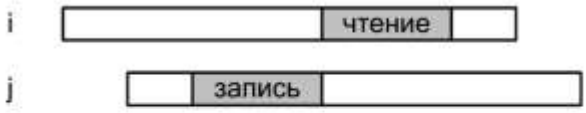
   Команды имеют зависимость по данным.

   O(i) – множество ячеек, изменяемых командой i;
   I(j) – множество ячеек, читаемых командой j.

А) Чтение после записи (ЧПЗ).

$$O(i) \cap I(j) \neq \varnothing$$

Б) Запись после чтения (ЗПЧ).



$$I(i) \cap O(j) \neq \varnothing$$

В) Запись после записи (ЗПЗ).

$$O(i) \cap O(j) \neq \varnothing$$

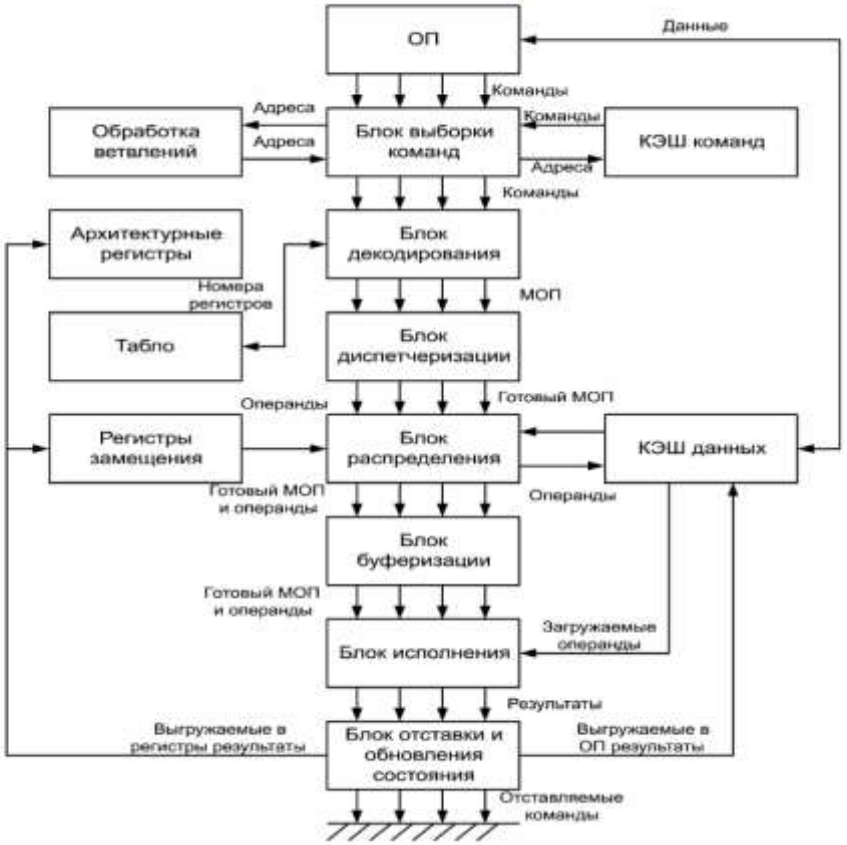
3. Риск по управлению.

   Из-за наличия команд перехода (10-20% потока команд) возможна неоднозначность при выборе очередной инструкции.

   Потери в лучшем случае: сброс всех поступивших команд за время декодирования команды ветвления.
   Потери в худшем случае: сброс всех поступивших команд за время декодирования, выборки операндов и исполнения команды ветвления.

Временные потери при обработке команд переходов

Обобщенная схема суперкскалярного суперконвейерного процессора



Архитектура конвейерного суперскалярного процессора. Статическое и динамическое предсказание переходов(5.12-5.14)

Способы устранения конфликтов по управлению

-Дублирование ступеней конвейера для обработки обеих ветвей

-Оптимизация кода на этапе компиляции с целью увеличения полезной нагрузки на дублированные ступени конвейера.

-Предсказание переходов.

Способы предсказания переходов

Точность предсказания: отношение числа правильно предсказанных переходов к их общему количеству.

Эффективность алгоритмов предсказания зависит от использования статистических данных, накопленных:
-заранее при компиляции и тестовых прогонах (статическое предсказание переходов);
-полученных в процессе исполнения программы (динамическое предсказание переходов).
-На основе статического и динамического подходов.

Стратегии статического предсказания переходов

-Переход происходит всегда (60-70%).
-Переход не происходит никогда (50%).
-Переход выполняется по результатам профилирования (75%).
-Переход определяется по коду операции (75%).
-Переход выполняется исходя из направления (85%).
-При первом выполнении переход имеет место всегда (90%).

Стратегии динамического предсказания переходов

-Одноуровневое предсказание: использует Шаблонную Таблицы Истории (Pettern History Table).

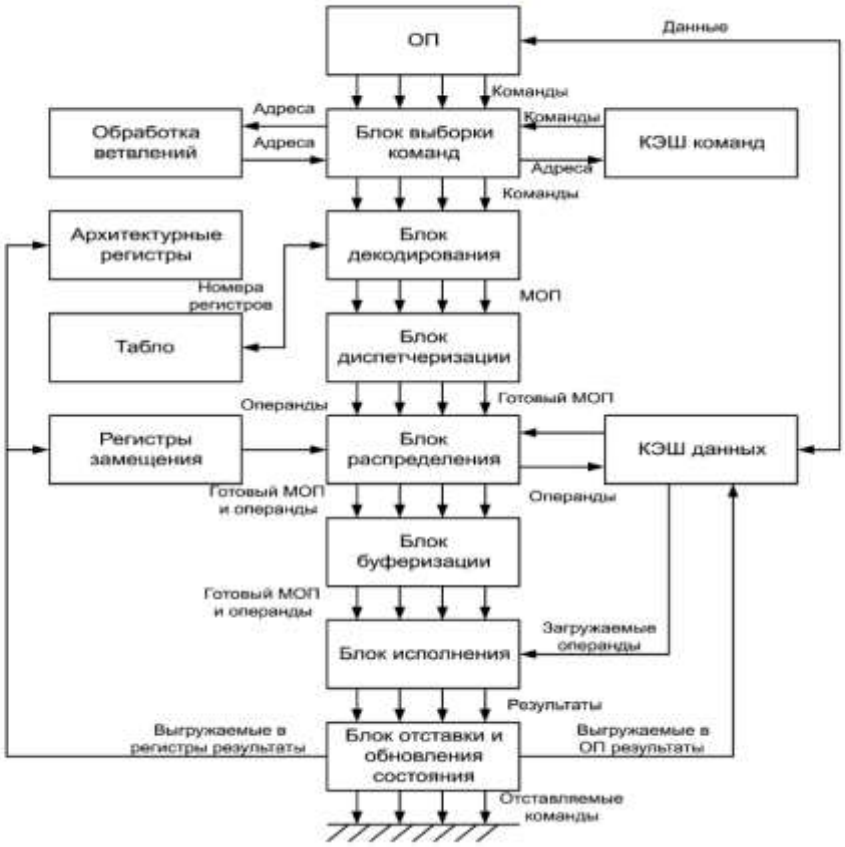
Выборка информации может происходить: по адресу команды перехода; по истории всех команд перехода; по истории исполнения только предсказываемой команды перехода.

Алгоритм предсказания зависит от размера строк PHT. При хранении одного бита переход предсказывается в соответствии с предыдущим итогом выполнения команды (точность ~78%).

При хранении двух бит учитывается переход для двух последних исполнений команды (точность ~82%).

- Таблица меток перехода (Branch Target Buffer)
-Двухуровневое предсказание.
-Гибридное предсказание

Обобщенная схема суперкскалярного суперконвейерного процессора



Архитектура конвейерного суперскалярного процессора. Конфликты в конвейере. Регистры замещения.(5.7,5.8,5.10,5.11)

Конфликты в конвейере (риски)

1. Структурный риск

   Команды одновременно обращаются к одному и тому же ресурсу (например, к ОП).

2. Риск по данным

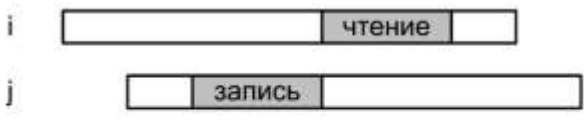
   Команды имеют зависимость по данным.

   O(i) – множество ячеек, изменяемых командой i;
   I(j) – множество ячеек, читаемых командой j.

А) Чтение после записи (ЧПЗ).

$O(i) \cap I(j) \neq \varnothing$

Б) Запись после чтения (ЗПЧ).



$I(i) \cap O(j) \neq \varnothing$

В) Запись после записи (ЗПЗ).

$O(i) \cap O(j) \neq \varnothing$

3. Риск по управлению.

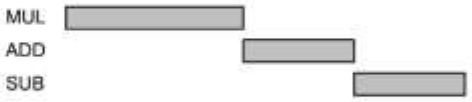
Из-за наличия команд перехода (10-20% потока команд) возможна неоднозначность при выборе очередной инструкции.

Потери в лучшем случае: сброс всех поступивших команд за время декодирования команды ветвления.
Потери в худшем случае: сброс всех поступивших команд за время декодирования, выборки операндов и исполнения команды ветвления.

Способы устранения конфликтов по данным, находящихся в регистрах

Пример 1:

```
MUL BX        ; (DX:AX = AX * BX)
ADD AX,BX     ; (AX = AX + BX)
SUB BX,2      ; (BX = BX - 2)
```



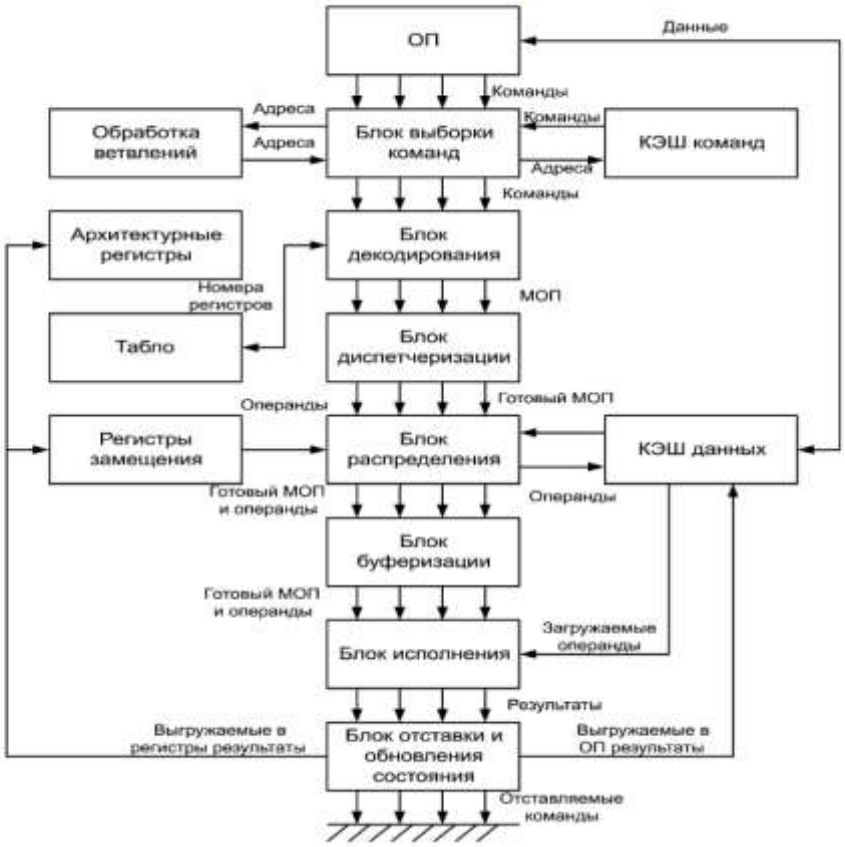
Правило:

Каждый новый результат записывается в новый регистр замещения.

```
MUL BX        ; (DX':AX' = AX * BX)
ADD AX,BX     ; (AX'' = AX' + BX)
SUB BX,2      ; (BX' = BX - 2)
```

Конфликт типа ЧПЗ по данных, находящимся в регистрах, может быть устранен с помощью бита достоверности

Обобщенная схема суперкскалярного суперконвейерного процессора



Арифметико-логические устройства (АЛУ). Структура АЛУ для целочисленного умножения (6.6-6.8)

Устройства целочисленного умножения

Умножение сводится к последовательному формированию частных произведений и их сложению.

По способу формирования частных произведений:
умножение со старших разрядов множителя со сдвигом влево,
умножение с младших разрядов множителя со сдвигом вправо.

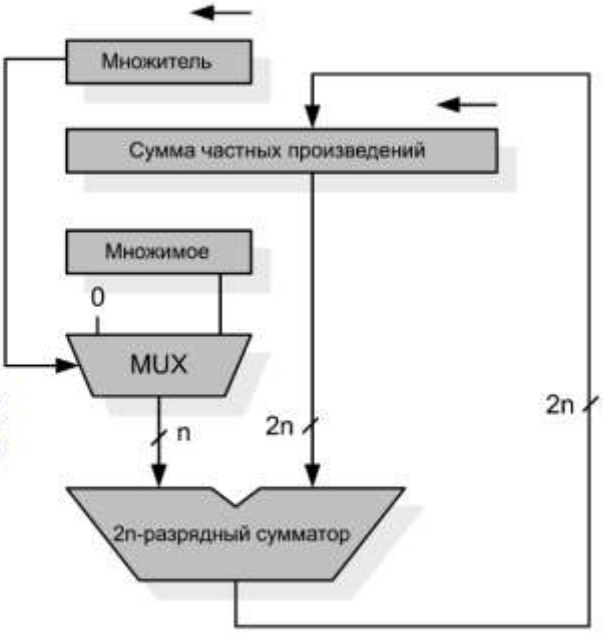
По способу накопления частных произведений: матричные умножители, древовидные умножители.

Способы ускорения работы устройств умножения:
- сокращение количества частных произведений;
- обработка нескольких разрядов множителя за такт;
- параллельное вычисление нескольких СЧП;
- конвейеризация умножителей.

Умножение со старших разрядов множителя со сдвигом влево

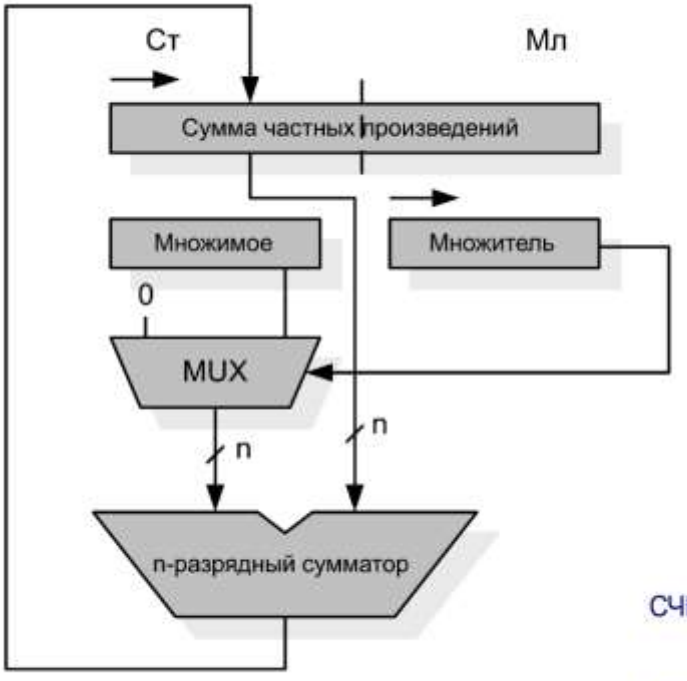
Старший разряд множителя определяет очередное частное произведение (ЧП), которое складывается с накопленной суммой частных произведений (СЧП). После этого СЧП и множитель сдвигаются на один разряд влево.

	A		1	1	0	6
x	B		1	0	1	5
	ЧП0		1	1	0	
	СЧП0		1	1	0	
	<-СЧП0	1	1	0	0	
	ЧП1		0	0	0	
	СЧП1=СЧП0+ЧП1	1	1	0	0	
	<-СЧП1	1	1	0	0	0
	ЧП2		1	1	0	
	СЧП2=СЧП1+ЧП2	1	1	1	1	0	30



(-) 2-n разрядный сумматор и шины данных.

Умножение с младших разрядов множителя со сдвигом вправо



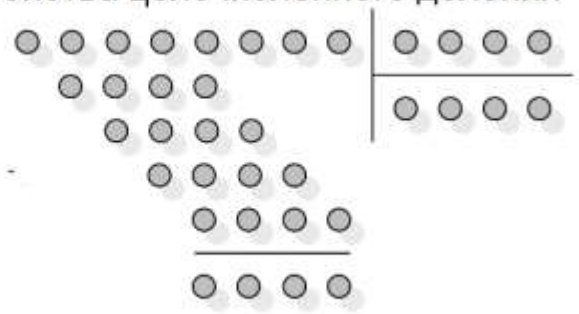
(+) n-разрядный сумматор и шины данных.

Младший разряд множителя определяет очередное частное произведение (ЧП), которое складывается с накопленной суммой частных произведений (СЧП). После этого СЧП и множитель сдвигаются на один разряд вправо.

	A	1	1	0		6
x	B	1	0	1		5
	ЧП0	1	1	0		
	СЧП0	1	1	0		
	СЧП0->	0	1	1	0	
	ЧП1	0	0	0		
	СЧП1=СЧП0+ЧП1	0	1	1	0	
	СЧП1->	0	0	1	1	0
	ЧП2	1	1	0		
	СЧП2=СЧП1+ЧП2	1	1	1	1	0	30

Деление с восстановлением и без восстановления остатка. Структура арифметико-логического устройства для целочисленного деления (6.13-6.14)

Устройства целочисленного деления



Деление с восстановлением остатка

17

Делимое			0	1	0	0	0	1	0 1 1	3
ЧО			0	1	0	0	0	1	1 0 1	5
<-ЧО		0	1	0	0	0	1	0		
-Делитель	-		0	1	1					
ЧО>0		0	0	0	1	0	1	0		
<-ЧО		0	0	0	1	0	1	0	0	
-Делитель	-			0	1	1				
ЧО<0		1	1	1	1	1	1	0	0	
+Делитель				0	1	1				
Восст. ЧО		0	0	0	1	0	1	0	0	
<-ЧО	0	0	0	1	0	1	0	0	0	
-Делитель	-			0	1	1				
ЧО>0	0	0	0	0	1	0	0	0	0	

Алгоритм:
1) ЧО = Делимое;
2) ЧО = ЧО*2;
3) ЧО = ЧО – Делитель * 2^n;
4) Если ЧО<0 то
   Ч<-0,
   ЧО = ЧО + Делитель * 2^n
   иначе Ч<-1;
5) Если все цифры то конец
   иначе пункт 2.

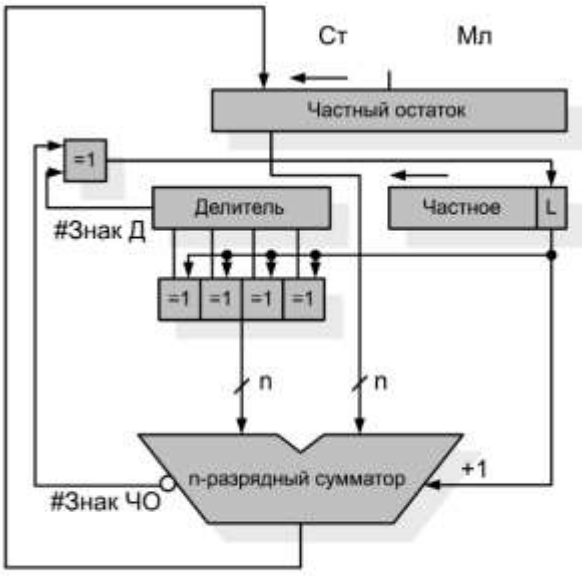
Деление без восстановления остатка

17

Делимое			0	1	0	0	0	1	0 1 1	3
ЧО			0	1	0	0	0	1	1 0 1	5
<-ЧО		0	1	0	0	0	1	0		
-Делитель	-		0	1	1					
ЧО>0		0	0	0	1	0	1	0		
<-ЧО		0	0	0	1	0	1	0	0	
-Делитель	-			0	1	1				
ЧО<0		1	1	1	1	1	1	0	0	
<-ЧО	1	1	1	1	1	1	0	0	0	
+Делитель	+			0	1	1				
ЧО>0	0	0	0	0	1	0	0	0	0	

Алгоритм:
1) ЧО = Делимое*2;
2) ЧО = ЧО – Делитель * 2^n;
3) ЧО = ЧО*2;
4) Если ЧО<0 то
   Ч<-0,
   ЧО = ЧО + Делитель * 2^n
   иначе
   Ч<-1;
   ЧО = ЧО – Делитель * 2^n
5) Если все цифры то конец
   иначе пункт 3.

Схема АЛУ для целочисленного деления



Организация операций сложения, вычитания, умножения и деления над числами с плавающей запятой (6.16-6.19)

Операции над числами с плавающей запятой.

1. Подготовительный этап.

- Разделение упакованного ЧПЗ на группы М,П,З.
- Проверка на специальное числовое значение.

2. Выполнение операции.

- Приведение порядков.
- Определение знака результата.
- Определение мантиссы результата.
- Определение порядка результата.
- Проверка на переполнение, потери значимости мантиссы, потери значимости порядка, неточности, деления на 0.

3. Заключительный этап.

- Проверка на специальное числовое значение.
- Нормализация результата.
- Проверка на переполнение, потери значимости мантиссы, потери значимости порядка, неточности.
- Упаковка полей З,П,М в ЧПЗ.

Организация операций сложения и вычитания над числами с плавающей запятой.

1. Подготовительный этап
2. Определение меньшего из двух порядков и проведение операции выравнивания порядков (сдвиг вправо на разность порядков).
3. Проверка на потерю значимости одного операнда (неточность).
4. Определение результирующего порядка как максимума.
5. Сложение мантисс и определение знака результата.
6. Проверка на переполнение мантиссы. Если да, то сдвигаем мантиссу вправо и увеличиваем порядок на 1.
7. Проверка на переполнение порядка.
8. Заключительный этап.

Организация операций умножения чисел с плавающей запятой.

1. Подготовительный этап
2. Проверка (М1=0 или М2=0). Если да, то Р=0.
3. Определение порядка результата: Пр = П1+П2-С.
4. Проверка на переполнение порядка.
5. Определение мантиссы результата: Мр = М1*М2.
6. Определение знака результата.
7. Заключительный этап.

Организация операций деления чисел с плавающей запятой.

1. Подготовительный этап
2. Проверка (М1=0 или М2=0). Если деление на ноль, то +/-бесконечность или ошибка.
3. Определение порядка результата: Пр = П1-П2+С.
4. Проверка на переполнение порядка.
5. Определение мантиссы результата: Мр = М1*(1/М2).
6. Определение знака результата.
7. Заключительный этап.

Аппаратные методы ускоренного умножения: матричные умножители, умножители по схеме Уоллеса (6.6,6.9,6.11)

Устройства целочисленного умножения

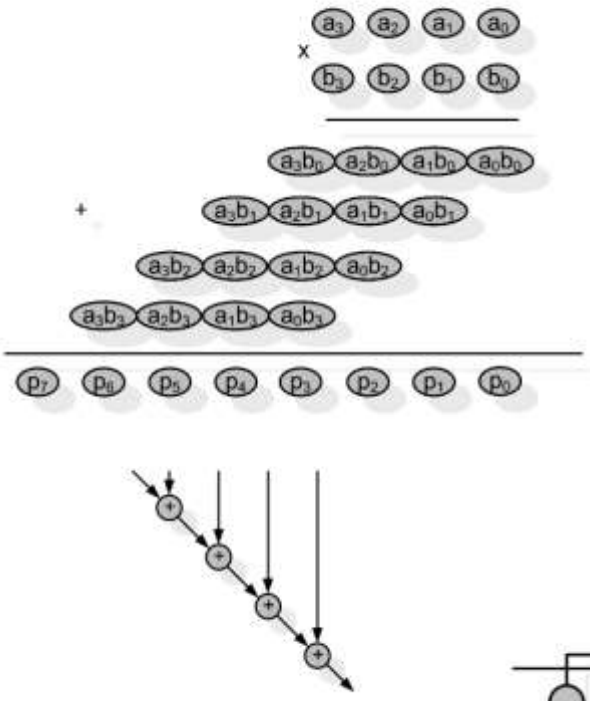
Умножение сводится к последовательному формированию частных произведений и их сложению.

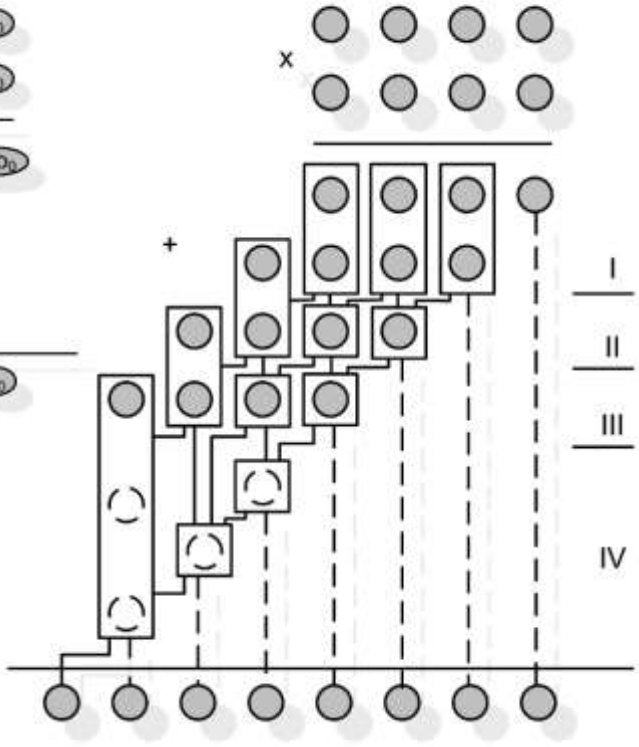
По способу формирования частных произведений:
умножение со старших разрядов множителя со сдвигом влево,
умножение с младших разрядов множителя со сдвигом вправо.

По способу накопления частных произведений: матричные умножители, древовидные умножители.

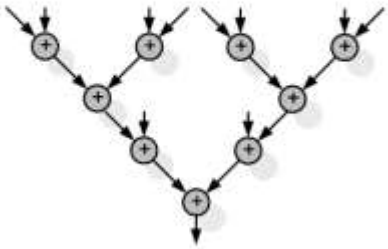
Способы ускорения работы устройств умножения:
- сокращение количества частных произведений;
- обработка нескольких разрядов множителя за такт;
- параллельное вычисление нескольких СЧП;
- конвейеризация умножителей.

Матричные умножители





Древовидные умножители (схема Уоллеса)



Матричные умножители