



Учебная дисциплина



Вычислительные средства








АСОИУ



(5 семестр)

# Программа курса

---

-  Лекции – 51 час.
-  Лабораторные работы – 34 часа  
(две пары раз в две недели)
-  Курсовая работа (зачёт с оценкой)
-  РК 1 и РК2
-  Итоговый экзамен

# Основная учебная литература

---



В.Л. Бройдо, О.П. Ильина.

Архитектура ЭВМ и систем. Учебник  
для ВУЗов. - СПб.: Питер, 2005. –  
718 с.



Э. Танненбаум. Архитектура  
компьютера. 4-е издание. – СПб.:  
Питер, 2006. – 699 с

# Основная учебная литература

---



К. Хамакер, Э. Вранешич, С. Заки.  
Организация ЭВМ. 5-е издание. –  
СПб.: Питер; Киев: Издательская  
группа ВНУ, 2003. – 848 с.

# Основная учебная литература

---



Бойко В.И. Схемотехника электронных систем. Цифровые устройства. СПб.: «БХВ-Петербург», 2004. – 512 с.

# Дополнительная учебная литература

---



М. Гук, Аппаратные интерфейсы ПК. Энциклопедия. – Питер 2003. – 523 с.



Новожилов О.П., Архитектура ЭВМ и систем: учеб. Пособие для бакалавров / Новожилов О.П., М. : Юрайт, 2012. – 527 с.

# Дополнительная учебная литература

---



Столингс. Структурная организация и архитектура компьютерных систем. 5-е В издание. – СПб.: Питер, 2002. – 896 с.




Б.Ф. Томпсон, Р.Б. Томпсон. Железо ПК: Энциклопедия. 3-е издание. – СПб.: Питер, 2004. – 960 с.

# Классификация вычислительных средств АСОИУ.

---


 Термины и определения:

 **Вычислительная система** — это совокупность одного или нескольких компьютеров или процессоров, программного обеспечения и периферийного оборудования, организованная для совместного выполнения информационно-вычислительных процессов.





# Классификация вычислительных средств АСОИУ.

---

 **Электронная вычислительная машина (ЭВМ), компьютер** — комплекс технических средств, предназначенных для автоматической обработки информации в процессе решения вычислительных и информационных задач.





## *Основные принципы построения, закладываемые при создании ВС:*

---

-  • возможность работы в разных режимах;
-  • модульность структуры технических и программных средств, что позволяет совершенствовать и модернизировать вычислительные системы без коренных их переделок;


## *Основные принципы построения, закладываемые при создании ВС:*

---

-  • унификация и стандартизация технических и программных решений;
-  • иерархия в организации управления процессами;
-  • способность систем к адаптации, самонастройке и самоорганизации;
-  • обеспечение необходимым сервисом пользователей при выполнении вычислений

# Классификация ВС

---

 **По назначению** вычислительные системы делят на:

**универсальные**

 **специализированные.**

# Классификация ВС

---



***По типу***

***построения*** вычислительные  
системы разделяются на:



**многомашинные**



**многопроцессорные**


# Классификация ВС



**Многомашинные вычислительные системы (ММС)** появились раньше, чем многопроцессорные. Основные отличия ММС заключаются в организации связей и обмена информацией между ЭВМ комплекса. **Многомашинная ВС** содержит некоторое число компьютеров, информационно взаимодействующих между собой.

# Классификация ВС

---

 В *многомашинных ВС* каждый компьютер работает под управлением своей операционной системы (ОС).



# Классификация ВС


---

- Информационное взаимодействие компьютеров в многомашинной ВС может быть организовано на уровне:
  - процессоров;
  - оперативной памяти;
  - каналов связи.





# Многопроцессорные системы

---

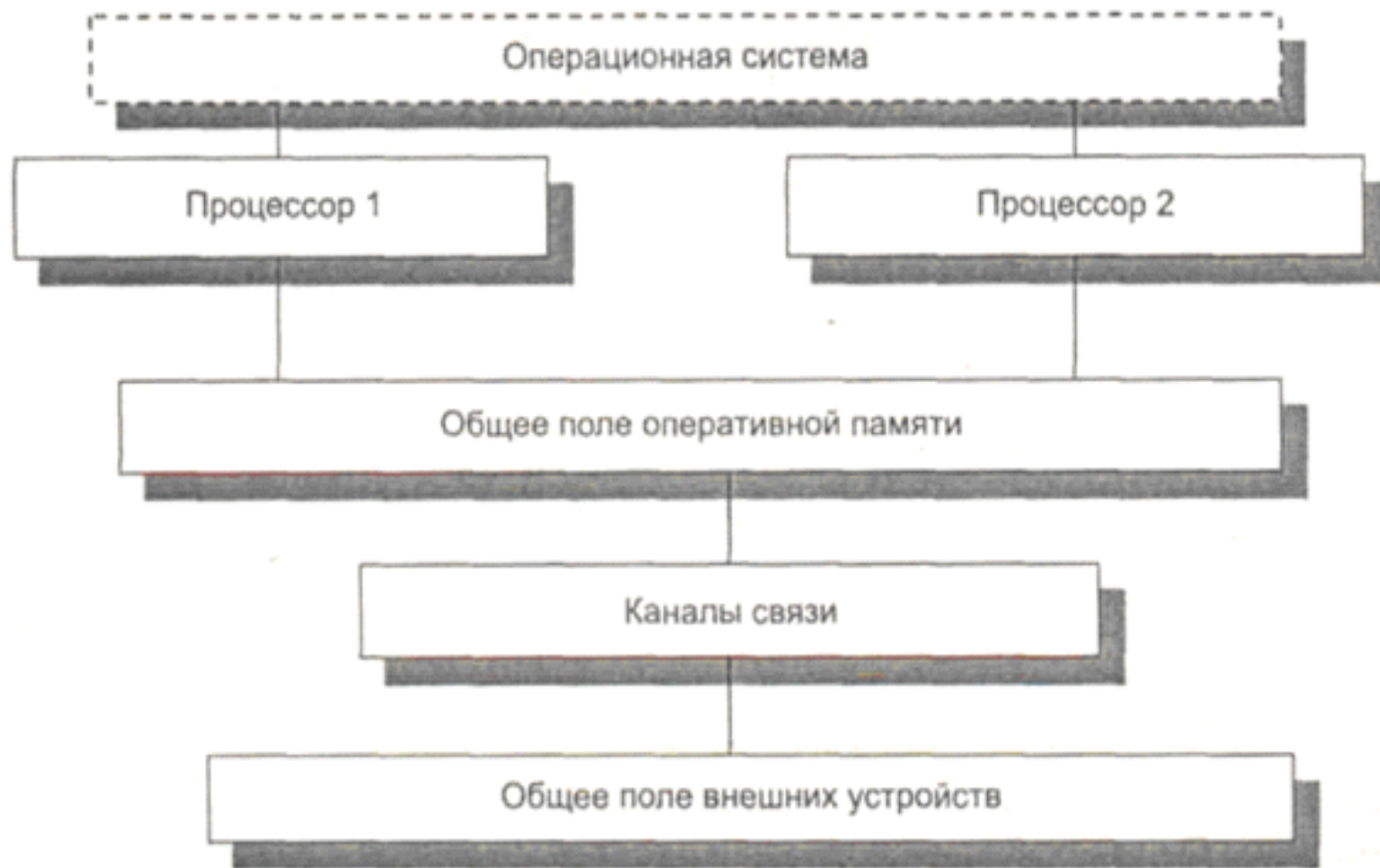
 **Многопроцессорные системы** (МПС) содержат несколько процессоров, информационно взаимодействующих между собой либо на уровне регистров процессорной памяти, либо на уровне ОП.

# Многопроцессорные системы

---


-  Общий доступ к внешней памяти и устройствам ввода-вывода обеспечивается обычно через каналы ОП.
-  Важным является и то, что многопроцессорная вычислительная система работает под управлением единой ОС, общей для всех процессоров.

# Схема взаимодействия процессоров



# Классификация ВС

---


 ***По типу ЭВМ или процессоров,***  
используемых для построения ВС,  
различают:

 **однородные системы**

 **неоднородные системы.**


# Классификация ВС

---

 **Однородная ВС** строится на базе однотипных компьютеров или процессоров. Однородные системы позволяют использовать стандартные наборы технических, программных средств, стандартные протоколы (процедуры) сопряжения устройств. Поэтому их организация значительно проще, облегчается обслуживание систем и их модернизация.


# Классификация ВС

---

 **Неоднородная ВС** включает в свой состав различные типы компьютеров или процессоров. При построении системы приходится учитывать их различные технические и функциональные характеристики, что существенно усложняет создание и обслуживание неоднородных систем.

# Классификация ВС

---

 ***По методам управления элементами ВС различают:***


 **централизованные**


 **децентрализованные**

 **со смешанным управлением.**

# Классификация ВС

---


 **В централизованных ВС** за управление отвечает главная или диспетчерская ЭВМ (процессор).

 Централизованный орган управления в системе может быть жестко фиксирован или эти функции могут передаваться другой ЭВМ (процессору), что способствует повышению надежности системы. Централизованные системы имеют более простые ОС.



# Классификация ВС

---

 **В децентрализованных системах** функции управления распределены между ее элементами. Каждая ЭВМ (процессор) системы сохраняет известную автономию, а необходимое взаимодействие между элементами устанавливается по специальным наборам сигналов.

# Классификация ВС


---



**В системах со смешанным управлением** совмещаются процедуры централизованного и децентрализованного управления. Перераспределение функций осуществляется в ходе вычислительного процесса, исходя из сложившейся ситуации.


# Классификация ВС


---


 *По принципу закрепления вычислительных функций* за отдельными ЭВМ (процессорами) различают системы с **жестким** и **плавающим закреплением функций**. В зависимости от типа ВС следует решать задачи статического или динамического размещения программных модулей и массивов данных, обеспечивая необходимую гибкость системы и надежность ее функционирования.

# Классификация ВС

---


 **По степени территориальной разобщенности вычислительных модулей ВС** делятся на системы:


 **территориально-сосредоточенные** – это когда все компоненты располагаются в непосредственной близости друг от друга;

 **распределенные** – это когда компоненты могут располагаться на значительном расстоянии, например, вычислительные сети;

# Классификация ВС


---

 **структурно-одноуровневые** – это когда имеется лишь один общий уровень обработки данных;

 **многоуровневые** (иерархические) структуры – это когда в иерархических ВС машины или процессоры распределены по разным уровням обработки информации, некоторые машины (процессоры) могут специализироваться на выполнении определенных функций.

# Классификация ВС

---

 ***По режиму работы ВС*** различают системы, работающие в **оперативном и неоперативном временных режимах**

# Вычислительные системы

## По назначению

- универсальные
- специализированные

## По типу построения

- многомашинные
- многопроцессорные

## По типу используемых ЭВМ или процессоров

- однородные
- неоднородные

## По методам управления элементами ВС

- централизованные
- децентрализованные
- со смешанным управлением

## По принципу закрепления вычислительных функций за отдельными ЭВМ (процессорами)


- с жестким закреплением функций
- с плавающим закреплением функций


## По степени территориальной разобщенности вычислительных модулей ВС

- территориально-сосредоточенные
- распределенные
- структурно-одноуровневые
- многоуровневые(иерархические) структуры

# **Структура вычислительной системы.**

---


 Структура ВС - это совокупность комплексируемых элементов и их связей. В качестве элементов ВС выступают отдельные ЭВМ и процессоры.


 Вычислительные системы имеют многоуровневую информационную организацию.



# 1-й уровень


---

 На 1 уровне системы располагаются *ЦП*, в состав которых входят *АЛУ*, *центральные устройства управления* и *внутренняя память процессоров* (иногда *сверхоперативная память СОП*).

 На этом же уровне находятся *модули ОЗУ*.


## 2-й уровень

---

 II уровень составляют *процессоры ввода-вывода* (каналы ввода - вывода), которые предназначены для выполнения операций ввода – вывода и обеспечивают все двусторонние связи между ОП и процессором, с одной стороны, и множеством различных периферийных устройств – с другой.


## 3-й уровень


---

 На III уровне находятся *интерфейс ввода – вывода* (устройство сопряжения) и *устройство управления внешними устройствами (УУВУ)*.

# 4-й уровень


---

 IV уровень составляет периферийные устройства.



 К ним относятся *внешние запоминающие устройства (ВЗУ) и устройства ввода-вывода.*

# 5-й уровень

---



 В современных вычислительных системах можно выделить V уровень, который составляют абонентские пункты, аппаратура передачи данных и каналы связи. Этот уровень необходим при использовании ВС в системах распределенной обработки данных, вычислительных центрах коллективного пользования, вычислительных сетях.

# Архитектура компьютера

-  **Архитектурой компьютера** называется его описание на некотором общем уровне, включающее описание пользовательских возможностей программирования, системы команд, системы адресации, организации памяти и т.д.
-  Архитектура определяет принципы действия, информационные связи и взаимное соединение основных логических узлов компьютера: процессора, оперативного ЗУ, внешних ЗУ и периферийных устройств



# Неймановские принципы построения ЭВМ

---

-  1. Вся информация в системе представляется и обрабатывается в двоичной системе счисления и разделяется на слова.
-  Все типы слов (данные, адреса, команды) кодируются одинаково.

# Неймановские принципы построения ЭВМ


---

-  3. Слова размещаются в ячейках памяти и идентифицируются номерами (адресами ячеек памяти).
-  4. Алгоритм решения задачи записывается в виде последовательности управляющих слов. Управляющее слово указывает на тип операции и операнды.




# Неймановские принципы построения ЭВМ

---

 Управляющее слово называется машинной командой.  
Последовательность управляющих слов называется машинной программой.

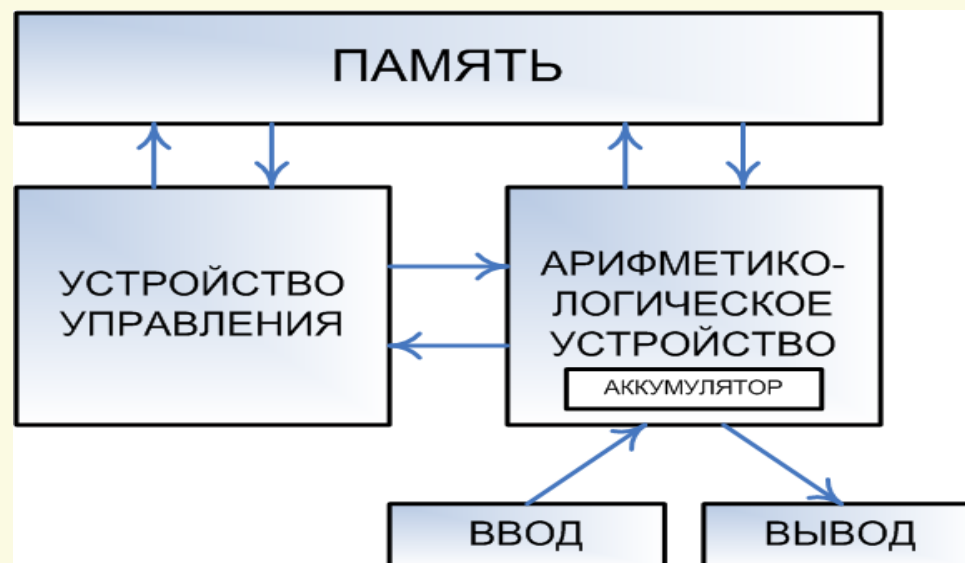
# Неймановские принципы построения ЭВМ

---

 5. Выполнение вычислений в ЭВМ сводится к последовательному выполнению машинных команд в порядке, определяемом машинной программой.




# Структура машины Фон-Неймана

- Операционная устройство
- Устройство управления
- Запоминающее устройство
- Устройство ввода-вывода







# Структура машины Фон-Неймана

---

-  Это *однопроцессорный компьютер*.
-  К этому типу архитектуры относится и архитектура персонального компьютера с *общей шиной*.
-  Все функциональные блоки здесь связаны между собой общей шиной, называемой также *системной магистралью*.



# Структурная организация современных ЭВМ

---

-  ЭВМ- инженерная система для выполнения вычислений по алгоритмам.
-  ЭВМ характеризуется составом и структурой.
-  Состав – набор устройств.
-  Структура – устройства и связи между ними.








# Структурная организация современных ЭВМ

---

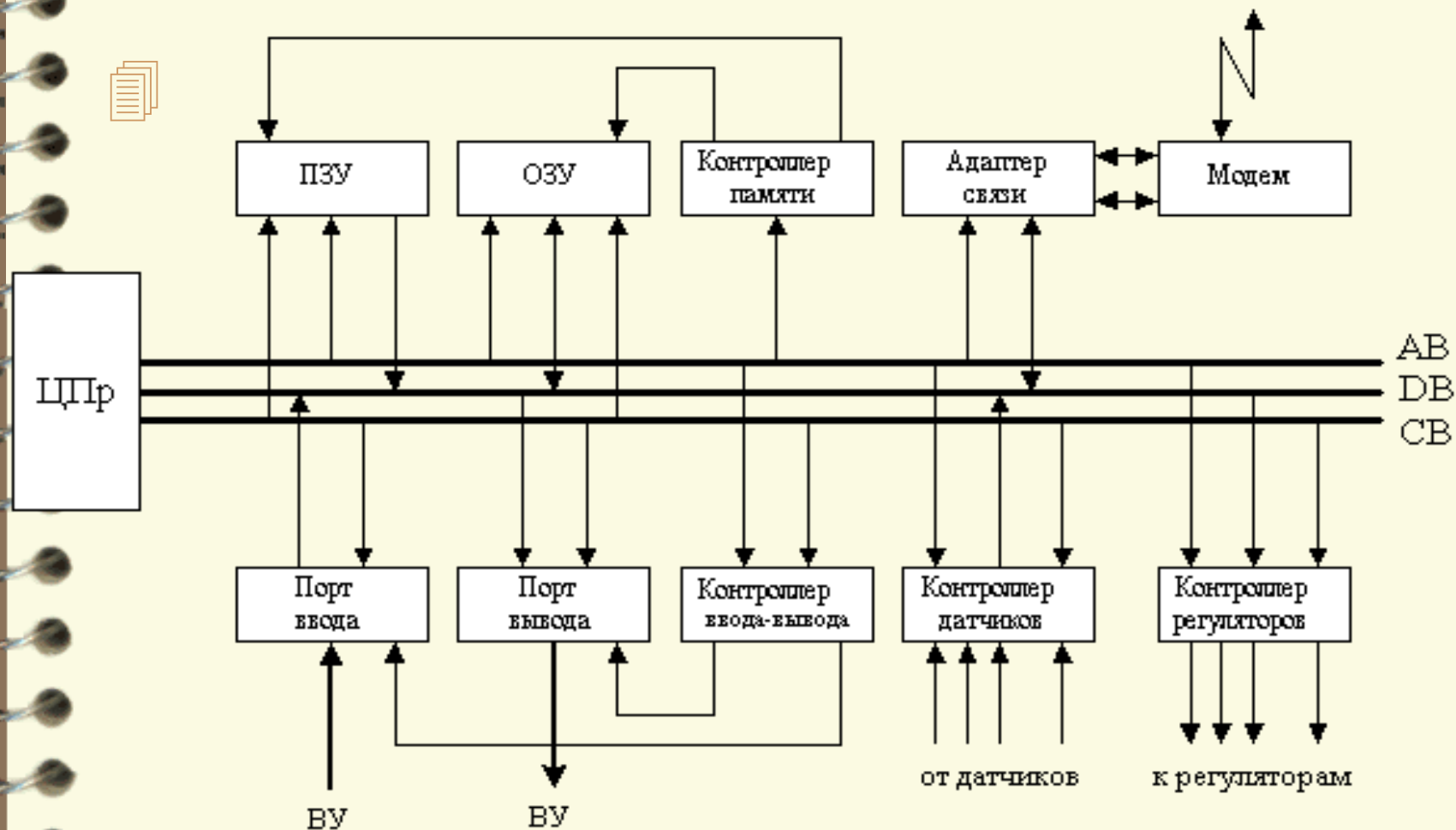
-  Совокупность связей двух взаимодействующих устройств по электрическим цепям называют интерфейсом.
-  Основными характеристиками интерфейса являются: скорость надёжность, стоимость.

# Типы интерфейсов

---

-  Последовательный
-  Параллельный
-  Параллельно-последовательный
-  Типовой шинный интерфейс:
  -  ШУ-шина управления;
  -  ША-шина адреса;
  -  ШД-шина данных

# Построение ЭВМ на основе единого интерфейса






# Основные характеристики ЭВМ

---


## 1. *Операционные ресурсы:*

 Способы представления данных; способы адресации; система машинных команд; средства контроля и диагностики; режимы работы ЭВМ (пакетный, запрос-ответ, разделение времени).

# Основные характеристики ЭВМ

---

 **2. Ёмкость памяти и организация памяти.**

 **Организация памяти определяется разрядностью хранимых в памяти слов. Память одной и той же ёмкости может иметь разную организацию.**



# Основные характеристики ЭВМ

---

- Исходя из класса решаемых задач, определяется требуемая ёмкость основной памяти  $E$
- Затем определяется длина адреса в битах  $n = \log_2 E$
- Процессор разрабатывается под заданную длину адреса памяти.

# Основные характеристики ЭВМ


---


-  Ёмкость памяти измеряется в битах, байтах и производных от них (Кбит, Кбайт, Мбит, Мбайт и т.д.)
-  В этих же единицах измеряется и ёмкость внешней памяти.

# Основные характеристики ЭВМ


---

## 3. Быстродействие.


 Основная характеристика быстродействия: число операций, выполняемых в секунду  $V$  [оп/сек];

 Длительность операции:  $T_{оп} = 1/V$  [сек]

# Основные характеристики ЭВМ

 Длительность различных операций существенно отличается (в 5-10 раз), поэтому быстродействие задаётся в векторной форме


$$V = (V_1, V_2, \dots, V_i)$$


 Для удобства сравнения быстродействия различных ЭВМ ввели скалярную величину: среднее количество операций, выполняемых в секунду:

$$V_i = 1 / \sum_{i=1}^m P_i t_i [on / c]$$

# Основные характеристики ЭВМ


$$V = 1 / \sum_{i=1}^m P_i t_i [on / c]$$


 где  $P_i$  -вероятность выполнения  $i$  – ой операции.

 Среднее быстродействие зависит от номинального быстродействия и от класса решаемых задач.

# Основные характеристики ЭВМ

---


 Для основных классов задач (экономические, научно-технические и т.д.) получены таблицы вероятностей появления различных операций и их длительностей.

 Эти таблицы получили названия смесей Гибсона, по фамилии разработавшего их статистика.




# Основные характеристики ЭВМ

## 4. Производительность.

 Производительность  $\lambda$  определяется средним числом задач, обрабатываемых ЭВМ в единицу времени.

$$T_{\text{средн}} = 1 / \lambda$$

 Среднее время решения задачи в однопрограммном режиме зависит от количества операций  $Q$  в программе,

# Основные характеристики ЭВМ

вероятностей  $P_i$ ,

номинального быстродействия  $t_i$


$$T_{\text{решен}} = Q * \sum P_i t_i$$


При мультипрограммном режиме определение времени решения задачи значительно усложняется, в общем случае:

$T_{\text{решения}} = T_{\text{задачи}} + T_{\text{ожидания ресурса}}$

# Основные характеристики ЭВМ


---

 В общем случае производительность ЭВМ определяется:

 - номинальным быстродействием,

 - классом задач,


 -организацией вычислительного процесса.


 Производительность, полученная с учётом этих параметров, называется системной производительностью.

# Основные характеристики ЭВМ

---


## **5. Надёжность.**

 **Надёжность – это свойство ЭВМ выполнять свои функции в течение заданного времени.**


 Надёжность характеризуется средним временем наработки на отказ и зависит от элементной базы и конструкции ЭВМ.

# Основные характеристики ЭВМ


---

 Обычно при расчёте надёжности принимают, что интенсивность отказов не изменяется во времени, и что отказы взаимно независимы и каждый из них нарушает работоспособность ЭВМ.

# Основные характеристики ЭВМ

 В этом случае интенсивность отказов устройства определяется суммой интенсивностей отказов элементов, входящих в состав устройства:

$$\lambda = \sum_{i=1}^m \lambda_i n_i \quad i = 1, m$$

  $\lambda_i$  -интенсивность отказов  $i$ -го типа элементов,  $n_i$  -количество элементов  $i$ -го типа в устройстве.

# Основные характеристики ЭВМ

---

📄 На основе интенсивности отказов определяют:

📄 - среднее время наработки на отказ

$$T = 1 / \lambda$$

📄 - вероятность безотказной работы устройств в течение времени  $t$

$$P(t) = e^{-\lambda t}$$

# Основные характеристики ЭВМ

Для ремонта и восстановления требуется время, называемое временем восстановления  $T_B$

Часто эксплуатационные свойства ЭВМ характеризуют коэффициентом готовности:

$$K_z = T / (T + T_B)$$



# Основные характеристики ЭВМ

---

Для увеличения надёжности ЭВМ применяют резервирование – параллельное включение нескольких элементов, узлов, устройств и т.д.

Кратность резервирования определяется по формуле:

$$K = \lg(1 - P) / \lg(1 - P_0)$$

# Основные характеристики ЭВМ

---

Если имеется устройство с надёжностью  $P_0$ , а необходимо иметь надёжность  $P > P_0$


# Основные характеристики ЭВМ

---

## **6. Стоимость.**

 На стоимость влияют:

 - элементная база;

 - операционные ресурсы;


 - быстродействие;

 - производительность;


 - надёжность.

# Основные характеристики ЭВМ

---

 К Найт показал связь стоимости ЭВМ и её производительности, подчинению закону Гроша:

$$V = K * S^2$$

  $V$  -среднее быстродействие,

  $S$  -стоимость ЭВМ,

  $K$  - коэффициент пропорциональности.

# Основные характеристики ЭВМ

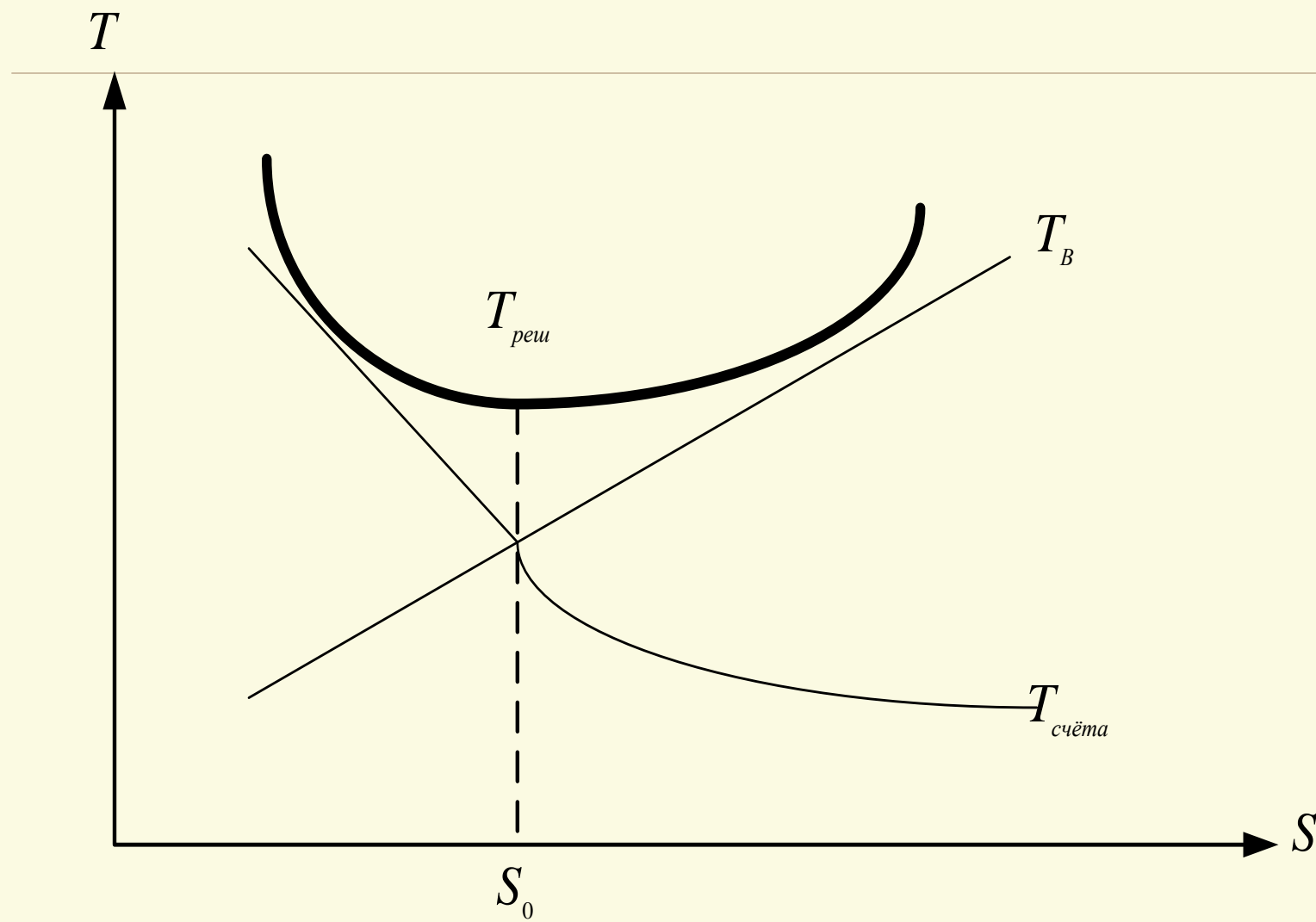
Оценим связь производительности ЭВМ со стоимостью. На производительность влияют быстродействие и надёжность.

$$T_{\text{решен}} = T_{\text{счёта}} + T_B$$

Из закона Гроша  $T_{\text{счёта}}$  обратно пропорционально квадрату стоимости. Время восстановления зависит от количества оборудования, т.е. монотонно-возрастающая функция стоимости.

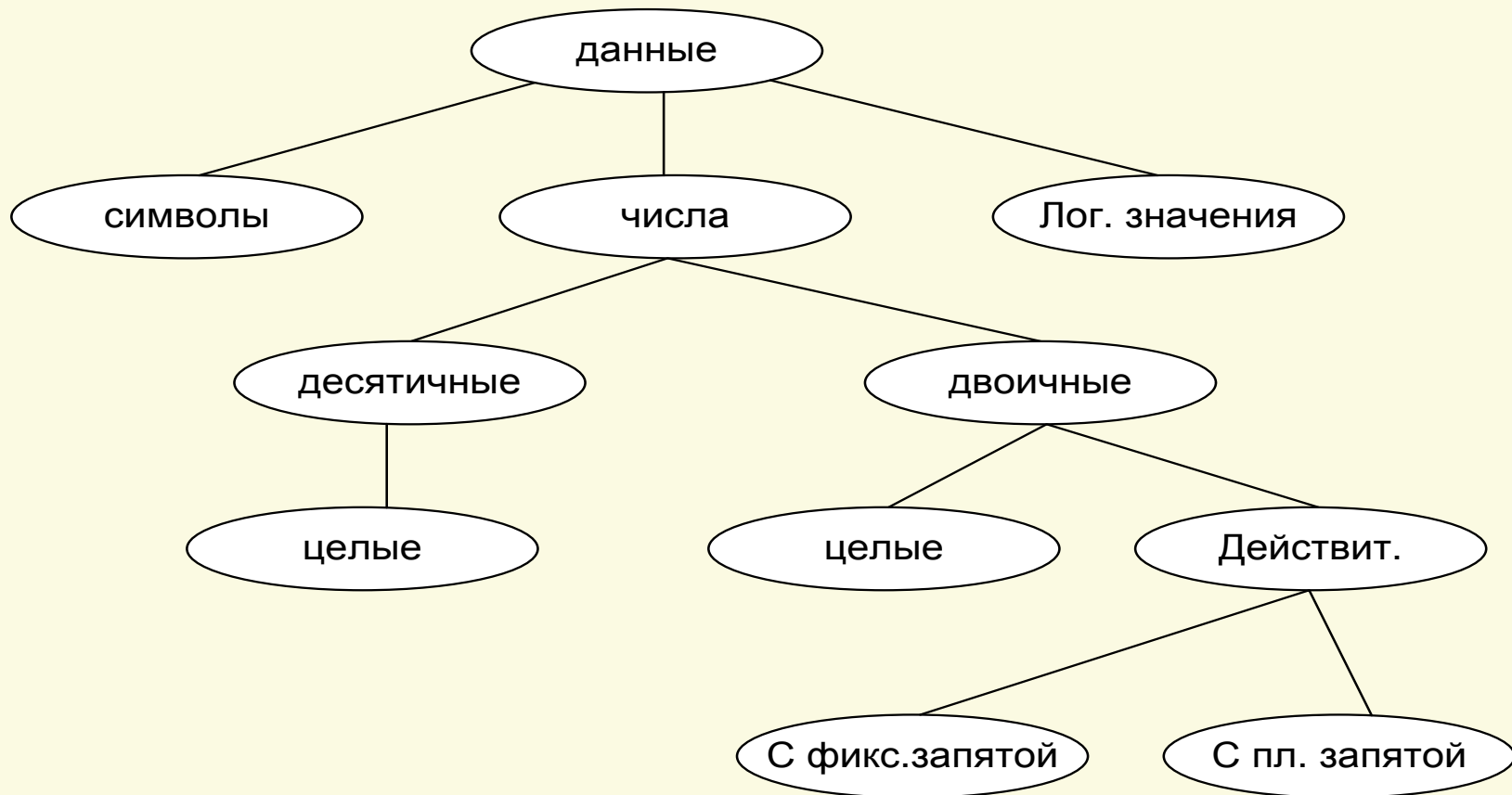
# Основные характеристики ЭВМ

- Рассмотрим графики  $T_{счёта}(S)$  и
- $T_B(S)$ . Из них следует, что функция
- $T_{решен}(S)$  носит экстремальный характер и достигает максимума при
- $S_0$  единиц оборудования.
- Кривая  $T_{счёта}(S)$  определяется эффективностью схемных решений, т.е. уровнем теории и практики проектирования ЭВМ.



# Представление данных в ЭВМ

## 1. Типы данных.






# Представление данных в ЭВМ

## 2. Целые двоичные числа

знак	Цифровые разряды
------	------------------




 Кодировка знака: 0 – «+», 1 – «-».

 Диапазон представления:

 
$$[-(2^n - 1); +(2^n - 1)]$$

# Представление данных в ЭВМ

## Особенности:

-  1. Сложение/вычитание – возможность переполнения из-за нехватки разрядов и инициирование прерывания.
-  2. При умножении результат требует  $2n$  разрядов.
-  3. Деление.  $C=A/B$ ; в качестве результата берётся целая часть частного.  $B \neq 0$

# Представление данных в ЭВМ

- Диапазон представления чисел с фиксированной запятой:

$$[-(1 - 2^{-n}); +(1 - 2^{-n})]$$

- Точность:  $2^{-n}$


- Особенность выполнения арифметических операций:

1. Сложение/вычитание: результат по модулю д.б. меньше 1.
2. Умножение: результат занимает  $2n$  разрядов, младшие  $n$  отбрасываются.

# Представление данных в ЭВМ

---

 3. Деление:

  $C = A/B; \quad B \neq 0 \quad |A/B| < 1$

# Представление данных в ЭВМ



## **4. Двоичные числа с плавающей запятой.**


Знак мантиисы	Мантиисса (дв. число с фикс. запятой)	Знак порядка	порядок
------------------	--	-----------------	---------


## 4. Числа с плавающей запятой

---


 Структура числа:

 Число с плавающей запятой состоит из:

 Знака мантиссы (указывающего на отрицательность или положительность числа)

 Мантиссы (выражающей значение числа без учёта порядка)

 Знака порядка

 Порядка (выражающего степень основания числа, на которое умножается мантисса)

Представление числа:


$$Z = \pm M * d^{\pm P},$$


где d-основание числа с пл. точкой.

$$d = 2^n, n = 1, 2, 3, \dots$$

## 4. Числа с плавающей запятой

---


 **Нормальная форма и нормализованная форма**

 *Нормальной формой* числа с плавающей запятой называется такая форма, в которой мантисса (без учёта знака) находится на полуинтервале  $[0; 1)$  ().




## 4. Числа с плавающей запятой

---

 Такая форма записи имеет недостаток: некоторые числа записываются неоднозначно (например, 0,0001 можно записать в 4 формах —  $0,0001 \cdot 100$ ,  $0,001 \cdot 10^{-1}$ ,  $0,01 \cdot 10^{-2}$ ,  $0,1 \cdot 10^{-3}$ )


## 4. Числа с плавающей запятой

---

 Нормализованная форма, в которой мантисса десятичного числа принимает значения от 1 (включительно) до 10 (не включительно), а мантисса двоичного числа принимает значения от 1 (включительно) до 2 (не включительно) (). В такой форме любое число (кроме 0) записывается единственным образом.

## 4. Числа с плавающей запятой

---

 Недостаток заключается в том, что в таком виде невозможно представить 0, поэтому представление чисел в информатике предусматривает специальный признак (бит) для числа 0.


## 4. Числа с плавающей запятой


---

📄 Так как старший разряд (целая часть числа) мантиссы двоичного числа (кроме 0) в *нормализованном* виде равен «1», то при записи мантиссы числа в ЭВМ старший разряд можно не записывать, что и используется в стандарте IEEE 754.

## 4. Числа с плавающей запятой

---

 Из-за ограничений на разрядность мантиссы и порядка возможны ситуации:

 - потеря значимости:  $M=0$ ,  $P \neq 0$  - машинный ноль;

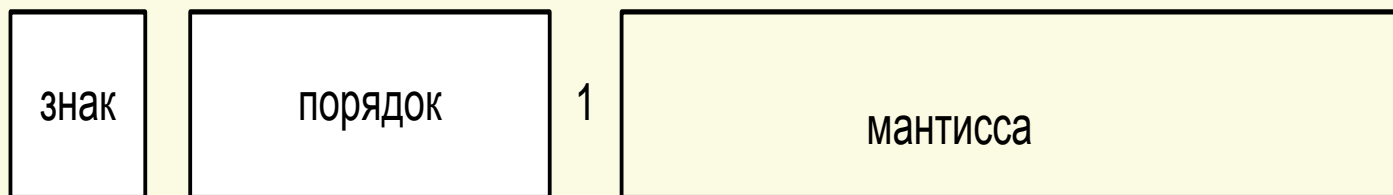
 -исчезновение порядка  $P < -2(2^n - 1)$

 -деление на 0

## 4. Числа с плавающей запятой


По принятому стандарту IEEE 754

структура представления чисел с плавающей запятой принята следующей:



# Представление данных в ЭВМ

 **5. Десятичные целые числа.**

 **Числа обрабатываются последовательно разряд за разрядом начиная с разрядов младшей тетрады.**

Двоичная  
тетрада

Двоичная  
тетрада

Двоичная  
тетрада

Двоичная  
тетрада






Двоичная  
тетрада

Бит  
знака

# Представление данных в ЭВМ

## 6. Строки символов.

1 или 2 байта на символ	1 или 2 байта на символ	1 или 2 байта на символ	1 или 2 байта на символ	1 или 2 байта на символ	1 или 2 байта на символ
-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------

-  *Таблица кодировки* — это стандарт, ставящий в соответствие каждому символу алфавита свой порядковый номер.
-  Международным стандартом для персональных компьютеров стала таблица ASCII.
-  На практике можно встретиться и с другой таблицей —
-  КОИ-8.
-  Unicode. Это 16-разрядная кодировка, т.е. в ней на каждый символ отводится 2 байта памяти. Такая кодовая таблица допускает включение до 65 536 символов.



# Представление данных в ЭВМ

---

 **7. Логические значения.**



***10000110100011***





***1110001100100***

# Машинные операции

---




## 1. Свойства машинных операций:

 Машинная операция –это действие, инициированное машинной командой и реализованное оборудованием ЭВМ.

 Множеству машинных операций соответствует множество машинных команд. Машинная команда инициирует определённую машинную операцию.






# Машинные операции

---

-  Набор машинных операций характеризуется двумя свойствами:
-  а) функциональной полнотой (лог. Функции, арифм. действия и т.д.)
-  б) эффективность, которая определяется затратами на оборудования для достижения требуемой производительности.


# Машинные операции


## 2. Классификация машинных операций.

-  а) арифметические и логические операции :  $+$ ,  $-$ ,  $*$ ,  $/$ , извл. корня, ... И, Или, Не.....;
-  б) посылочные: обмен между ОП-ЦП;
-  в) операции прерывания;
-  г) ввод-вывод;
-  д) системные

# Система команд ЭВМ


---


 По функциональному назначению в системе команд ЭВМ различают следующие группы:

 **команды передачи данных** (обмен входами между регистрами процессора, процессора и оперативной памятью, процессора и периферийными установками).

# Система команд ЭВМ


---

 **Команды обработки данных**  
(команды сложения, умножения, сдвига, сравнения).



 **Команды передачи управления**  
(команды безусловного и условного перехода).

# Система команд ЭВМ

---

 **Команды дополнительные** (типа  
RESET, TEST,-).

# Способы адресации

-  **Способ адресации** – это правило определения адреса и операнда на основе информации в адресной части команды.
-  Эффективность способа адресации влияет на временные затраты и затраты на определённый состав оборудования.






# Способы адресации

Способы формирования адресов ячеек памяти можно разделить на **абсолютные и относительные.**

Абсолютные способы формирования предполагают, что двоичный код адреса ячейки памяти может быть целиком извлечен либо из адресного поля команды, либо из какой-нибудь другой ячейки в случае косвенной адресации.


# Способы адресации


---

-  Относительные способы формирования предполагают, что двоичный код адресной ячейки памяти образуется из нескольких составляющих:
-   $B$  в код базы,  
 $I$  в код индекса,  
 $S$  в код смещения.
-  Эти составляющие используются в различных сочетаниях.

# Способы адресации

## Относительная адресация

 При относительной адресации применяется способ вычисления адреса путем суммирования кодов, составляющих адрес.


  $A = B + I + C$


$$A = B + C$$

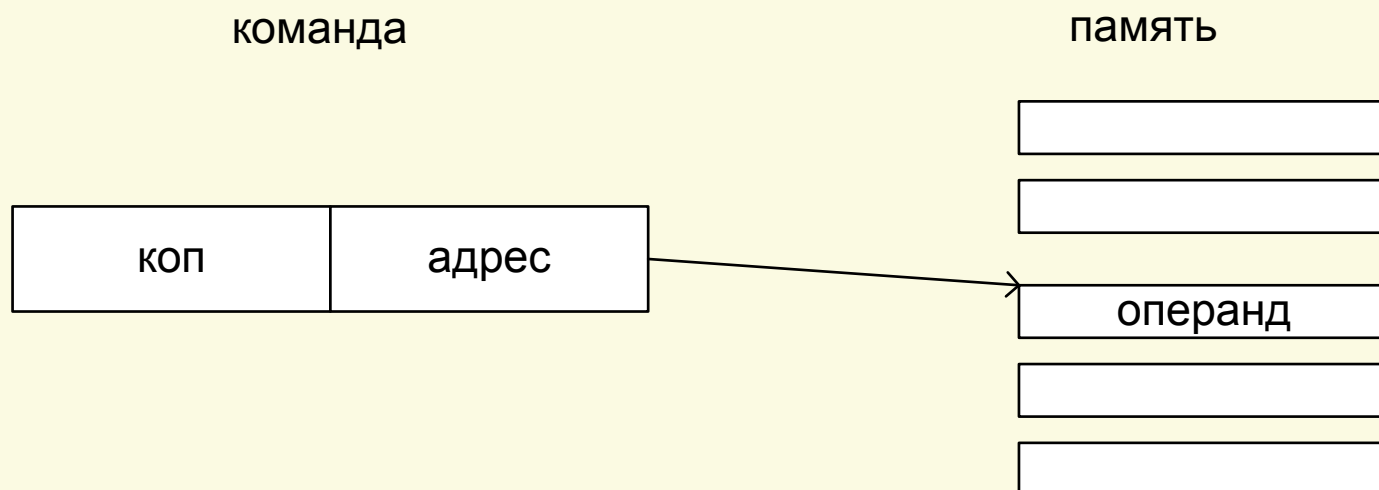
$$A = I + C$$

# Способы адресации

## 1. Прямая адресация.


 адресная часть команды содержит непосредственный (прямой) адрес

 операнда в памяти.



# Способы адресации

## 2. Непосредственная адресация

 Целочисленное значение операнда записывается в поле команды.


команда


КОП	1010
-----	------

Операнд: константа 10 в  
двоичном коде

# Способы адресации

## 3. Косвенная адресация.

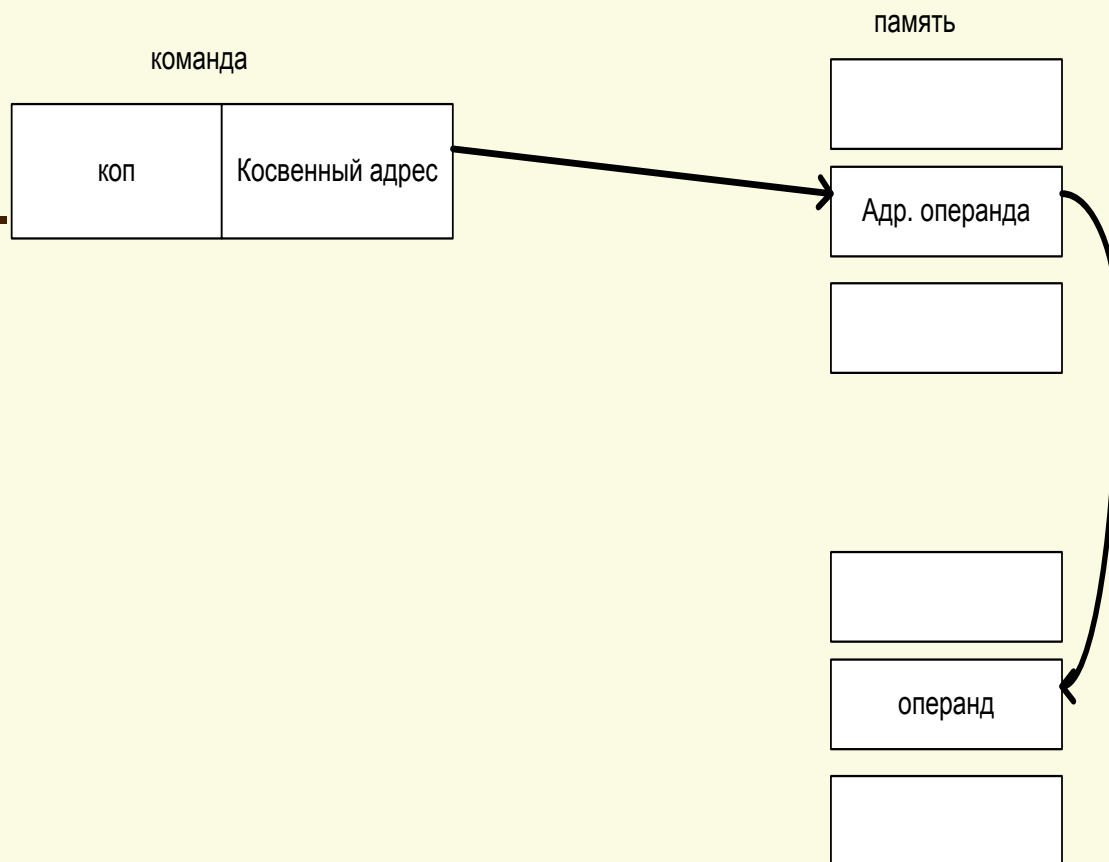
 Адресная часть команды содержит косвенный адрес;

 Адресный код команды в этом случае указывает адрес ячейки памяти, в которой находится адрес операнда или команды. Косвенная адресация широко используется в малых и микроЭВМ, имеющих короткое машинное слово, для преодоления ограничений короткого формата команды (совместно используются регистровая и косвенная адресация).

# Способы адресации




Пояснение  
косвенной  
адресации.



# Способы адресации

---

## 4. Регистровая адресация.

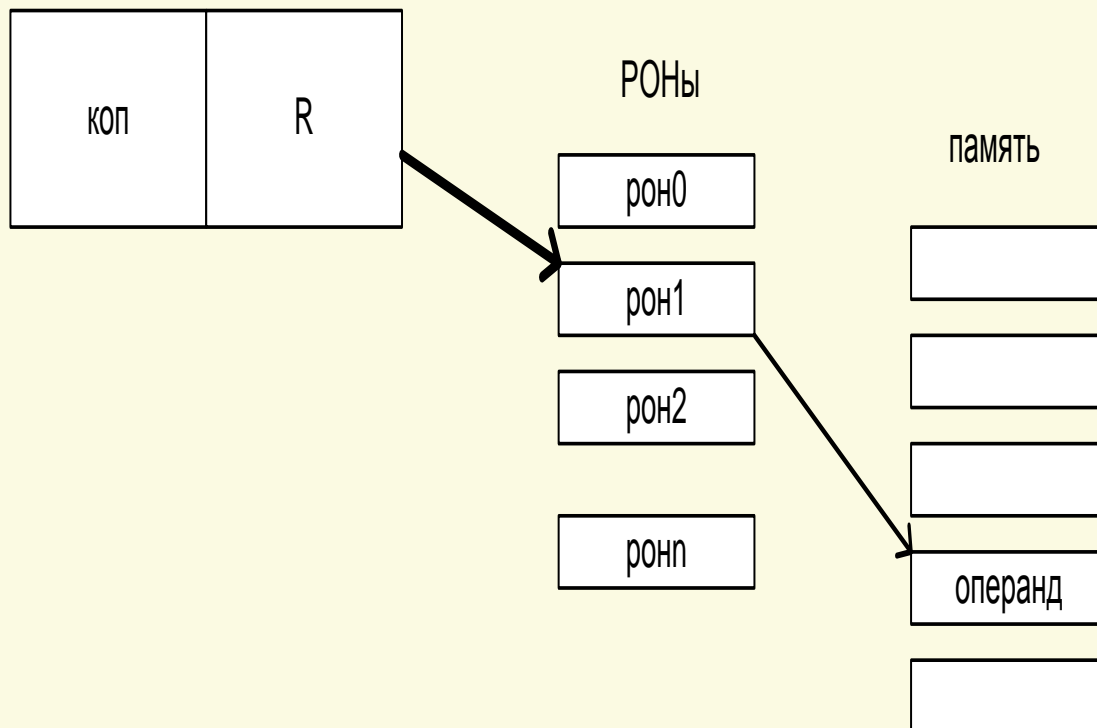
 Применяется, когда промежуточные результаты хранятся в одном из рабочих регистров центрального процессора (регистрах общего назначения (РОН)). Поскольку регистров значительно меньше чем ячеек памяти, то небольшого адресного поля может хватить для адресации.



# Способы адресации




Пояснение  
регистровой  
адресации.



# Способы адресации

---

## 5. Адресация с модификацией адресов.

 Для реализуемых на ЭВМ методов решения математических задач и обработки данных характерна цикличность вычислительных процессов, когда одни и те же процедуры выполняются над различными операндами, упорядоченно расположенными в памяти.


# Способы адресации

- ❏ Программирование циклов существенно упрощается, если после каждого выполнения цикла обеспечено автоматическое изменение в соответствующих командах их адресных частей согласно расположению в памяти обрабатываемых операндов.
- ❏ Такой процесс называется **модификацией команд**, и основан на возможности выполнения над кодами команд арифметических и логических операций.

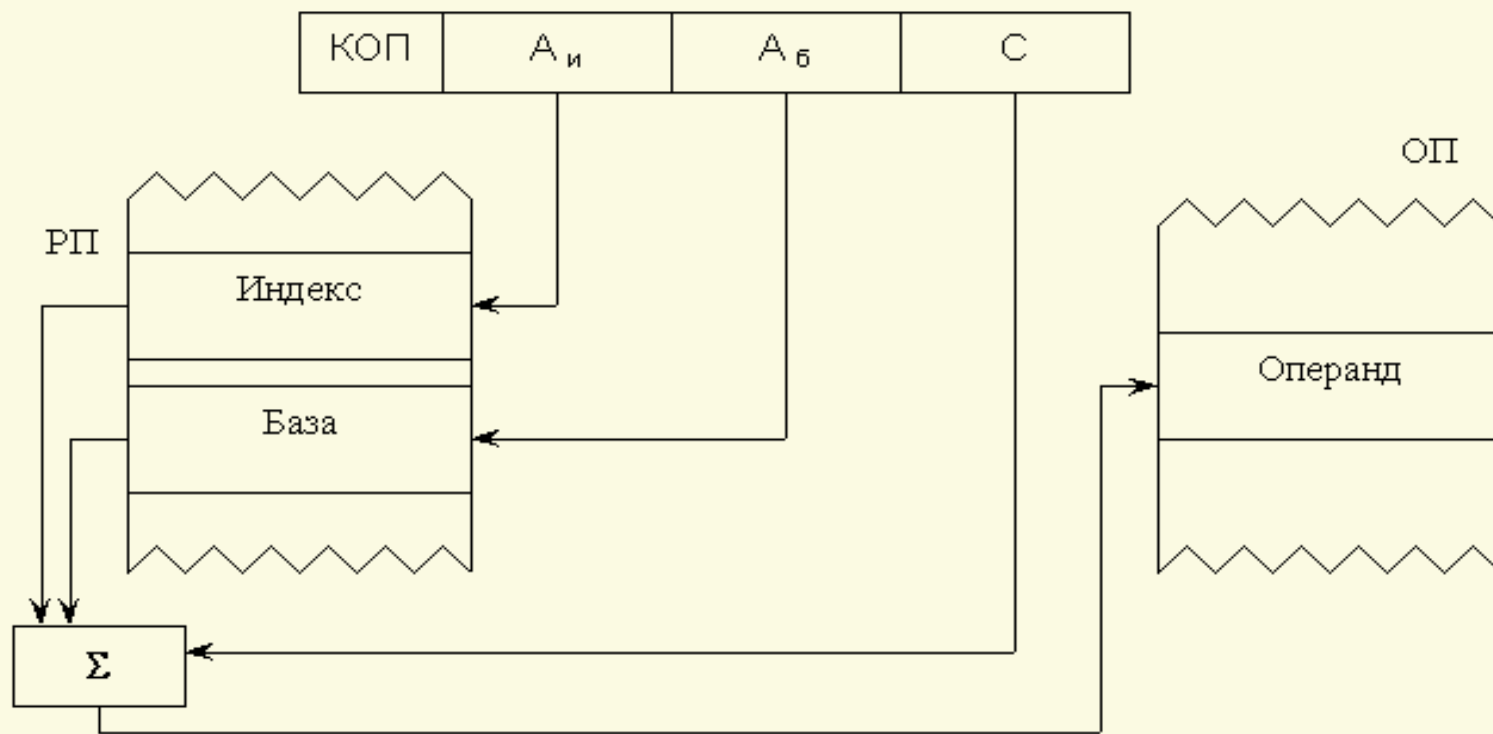
# Способы адресации

---

## Индексная адресация

 Для работы программ с массивами, требующими однотипных операций над элементами массива, удобно использовать индексную адресацию.

# Индексная адресация




# Индексная адресация

- ❏ Адрес  $i$ -того операнда в массиве определяется как сумма начального адреса массива операнда, задаваемого смещением  $S$ , и индекса  $I$ , записанного в одном из регистров регистровой памяти, называемым **индексным регистром**.
- ❏ Адрес индексного регистра задается в команде полем адреса индекса  $A_i$ .
- ❏ В каждом  $i$ -том цикле содержимое индексного регистра изменяется на постоянную величину, как правило, это 1.

# Способы адресации.

---


## **6. Прямая адресация с модификацией.**

 В полях команды содержится Рон, в котором текущий индекс, а во втором поле команды базовый (начальный) адрес.

# Способы адресации

---

## 7. Регистровая адресация с модификацией.

 Поля команды содержать два значения РОН: в первом значение индекса, второй содержит значение базового адреса.



# Способы адресации

---




## 8. Страничная адресация.

КОП	Адрес страницы	Адрес слова
-----	-------------------	----------------

# Способы адресации

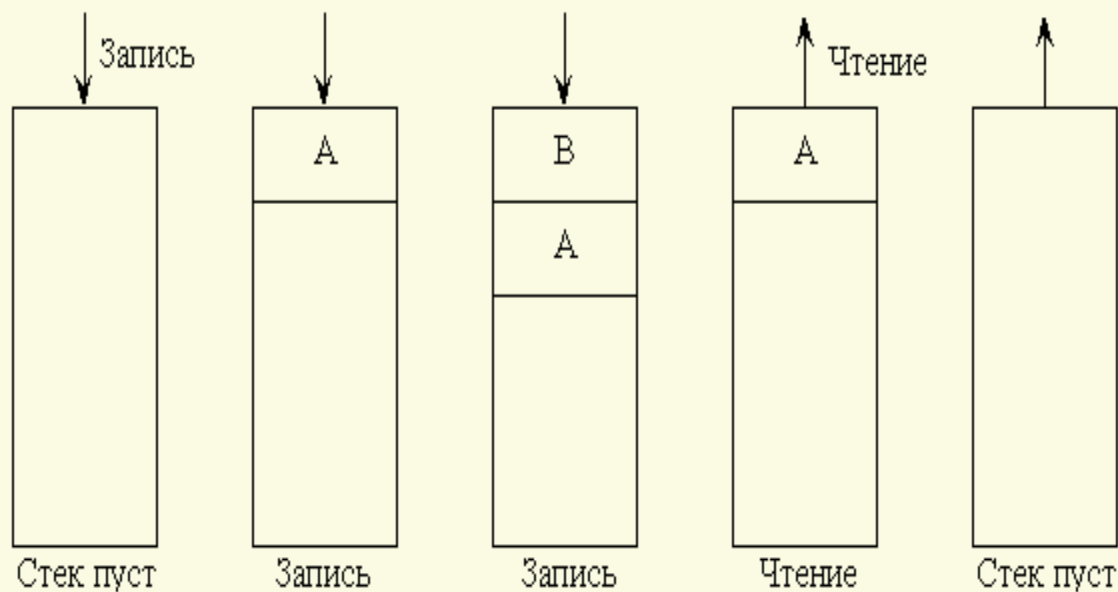
---

## 9. Стековая адресация

 Стековая память широко используется в современных ЭВМ. Хотя адрес обращения в стек отсутствует в команде, он формируется схемой управления:

# Способы адресации

## 9. Стекковая адресация



## 9. Стековая адресация


- ❏ Для чтения записи доступен только один регистр в вершина стека. Этот способ адресации используется, в частности, системой прерывания программ при вложенных вызовах подпрограмм.
- ❏ Стековая память реализуется на основе обычной памяти с использованием указателя стека и автоиндексной адресации.
- ❏ Запись в стек производится с использованием автодекрементной адресации, а чтение - с использованием автоинкрементной адресации.


# Исходные данные для проектирования ОУ

---

 F - множество операций,



 D - множество входных данных,

 R - множество выходных данных, результатов вычислений,

 ограничения на время выполнения операции.



# Исходные данные для проектирования ОУ

---

-  Задача проектирования- создание ОУ минимальной размерности и сложности.
-  Для проектирования ОУ все операции описываются в виде наборов микропрограмм.



# Исходные данные для проектирования ОУ

---

-  Формализованная микропрограмма
-  (ФМП) описывает работу ОУ безотносительно к его структуре на основе математических методов прикладной математики.

# Исходные данные для проектирования ОУ




---

-  Язык формализованного описания микропрограмм (ЯФМП) применяется для описания слов, микроопераций (МО) и логических условий (ЛУ).
-  Как правило удобнее использовать инженерную версию ЯФМП.



# ЯФМП


---

-  ЯФМП состоит из описательной части (описание слов, МО и ЛУ) и содержательной части – графа ФМП.
-  Описательный уровень позволяет описать работу ОУ на регистровом уровне.
-  Содержательный уровень отслеживает выполнение переходов по логическим условиям.

# ЯФМП


---

## 1. Описание слов:

 с ( $n1:n2$ ), где:

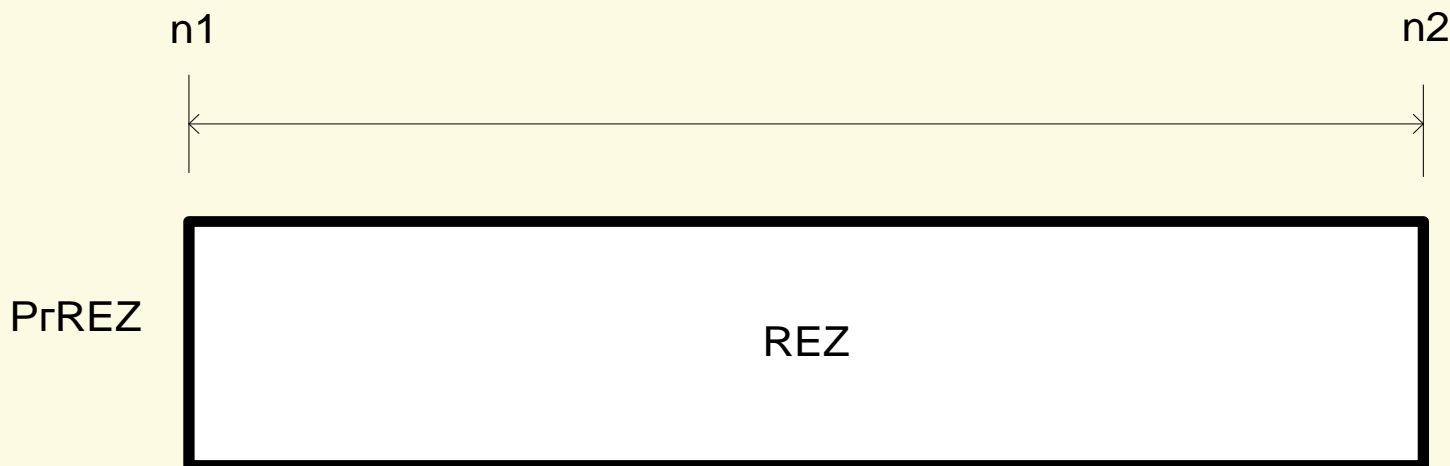
 С – идентификатор (присвоенное имя),  $n1$ -старший разряд слова,

  $n2$ -младший разряд слова.

 Каждое слово связано со своим регистром, либо другим устройством.

# ЯФМП


Например, запись СчК(3:0) обозначает четырёхразрядный счётчик с присвоенным ему идентификатором СчК, ниже изображён регистр результата PrREZ(n1:n2)





# ЯФМП


---

## 2. Описание массивов:

  $M[m2:m1](n2:n1)$  где:

  $m2$  и  $m1$  –указатели номеров старшей и младшей ячейки массива;


  $n2$  и  $n1$  - разряды слова внутри массива.


 Например:  $[255:0](n2:n1)$

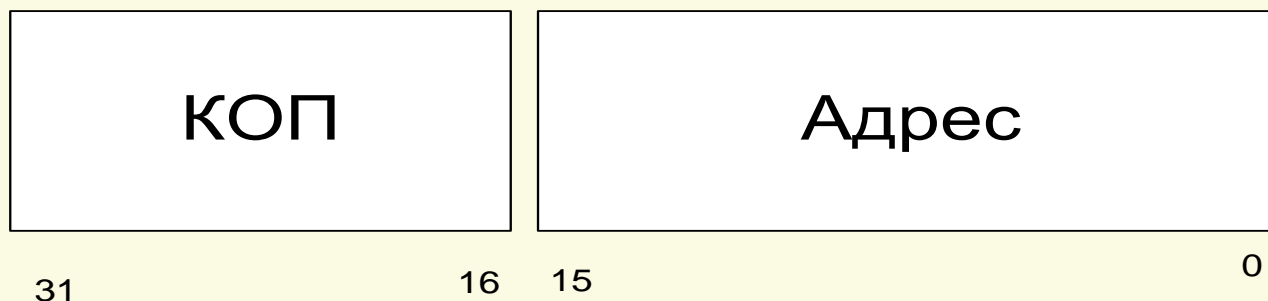
# ЯФМП

## 3. Поля

 Например: Рг А(31:0); Рг А(15:0)

 Адр(15:0) = РгА(15:0) КОП(15:0)=РгА(31:16)


 Полям можно присваивать собственные имена.




# ЯФМП

---


## 4. Типы слов.


 Каждое слово характеризуется определённым типом.

 Нашли применения следующие типы слов:

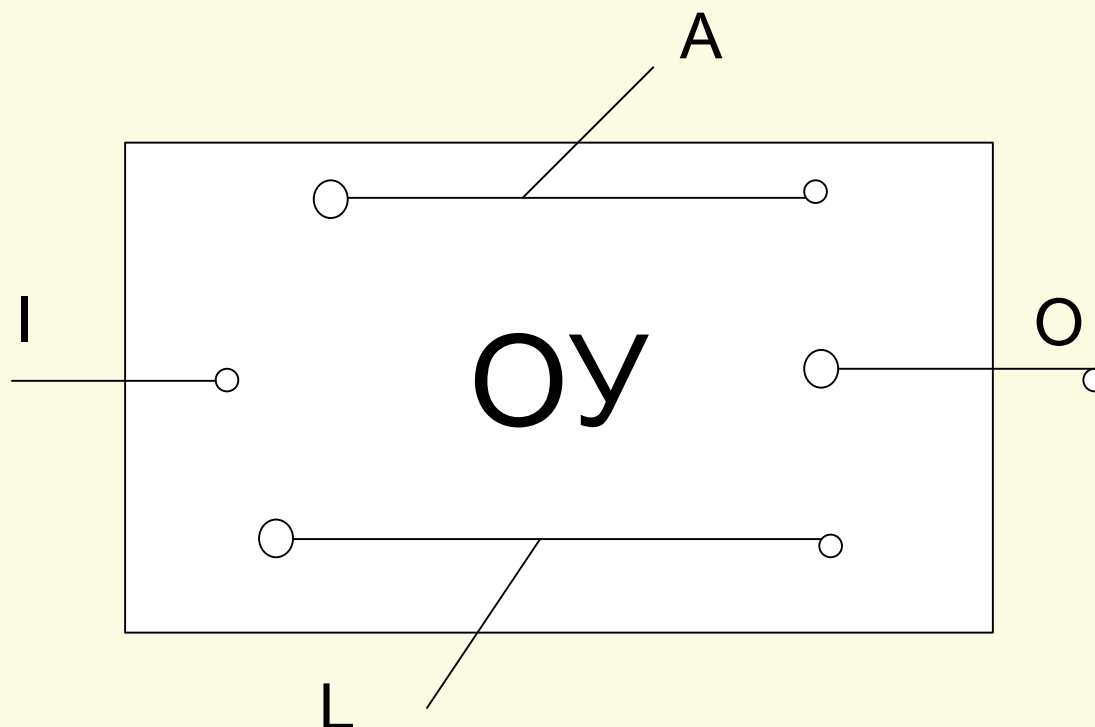
 - входные (I);

 -внутренние (L);

 - вспомогательные (A)(промежуточные)  
(действуют на 1 такт);

 - выходные (O).

# ЯФМП. 4. Типы слов.



# ЯФМП. 4. Типы слов.



Все слова, используемые в микропрограмме должны быть описаны в следующей таблице:


Наименование и формат слова	Поля	Соответствующий регистр
A(7:0)	Знак=A(0:0)	РгА
B(7:0)	Знак=B(0:0)	РгВ
R(7:0)	Знак=R(0:0)	РгR
Сч(3:0)		РгСч
C(7:0)	Знак=C(0:0)	РгC




# ЯФМП

---





## 5. Двоичные выражения.

 Описывают преобразования, выполняемые микрооперациями.

 Двоичные выражения (ДВ) состоят из элементарных ДВ, соединённых знаками двоичных операций.









## 5. Двоичные выражения.

 В качестве ДВ используют:

-  1. Константы (двоичные, восьмеричные, шестнадцатеричные);
-  2. Слова, используемые только со своими идентификаторами;
-  3. Поля;
-  4. Элементы массивов  $M[31:0](15:0)$

# ЯФМП

## 6. Двоичные операции.

-  Инверсия старшинство: 1
-  Конкатенация старшинство: 2
-  Конъюнкция старшинство: 3
-  Дизъюнкция старшинство: 4
-  Сложение по mod 2 старш: 4
-  Арифм. сложение старш: 5
-  Циклич. Сложение старш: 5
-  Вычитание старшинство: 5



# Микрооперации

- Синтаксис записи микрооперации:
- $\langle A \rangle := \langle B \rangle$  оператор присваивания.
- А м.быть словом, полем, элементом массива.
- В – двоичное выражение.
- Микрооперация (МО) выполняется за один такт. В начале такта вычисляется двоичное выражение, в конце такта выполняется присваивание.


# Микрооперации

## Типовые микрооперации. Классификация.

 1. Установка значения  $A := \text{const}$




 2. Инвертирование  $A := !A$

 3. Передача  $A := B$

 Микрооперация передаёт информацию из одного регистра в другой.

# Микрооперации


---

-  Оператор присваивания в синтаксической записи:
-   $\langle \text{левая часть} \rangle := \langle \text{двоичное выражение} \rangle$
-  Вся микрооперация выполняется за один машинный такт.

# Микрооперации

---

 4. Сдвиговые микрооперации.


 При сдвиге указывается направление сдвига, на сколько разрядов осуществляется сдвиг и какими значениями заполняются освободившиеся разряды.

# Микрооперации


---

 Синтаксис записи МО сдвига:

 RK – сдвиг на k разрядов вправо;

 LK – сдвиг на k разрядов влево.

 Например:  $A := R1(1.A)$ ,  $A := L1(A.0)$

 В первом случае заполнение единицами, во втором нулями.



# Микрооперации

---

 Типы сдвигов:

 - **логический;**



 - **циклический;**

 - **арифметический.**

# Микрооперации





---

## Правила выполнения сдвигов:

-  1. При **логическом** сдвиге освобождаемые разряды заполняются нулями.
-  2. При **циклическом** сдвиге освобождённые разряды заполняются выдвигаемыми разрядами.






# Микрооперации

---

-  При арифметическом сдвиге выполняются следующие правила:
-  - при сдвиге влево освобождаемые разряды **заполняются нулями**;
-  - при сдвиге вправо освобождаемые разряды **заполняются значением бита знака**;
-  - разряд знака **не сдвигается**, сдвигается только числовая часть числа.

# Микрооперации


## 5. Микрооперации счёта.


-  Используются в том числе и для описания работы счётчиков.
-   $A := A + 1$  – инкремент;
-   $A := A - 1$  – декремент
-  Микрооперации арифметического сложения и вычитания.
-  При сложении операнды выравниваются по младшим разрядам с заполнением нулями старших лишних разрядов.

# Микрооперации

---

## **Совместимость микроопераций.**

 Совместимыми называются микрооперации, которые выполняются в одном такте.

 Один машинный такт может содержать несколько микроопераций.

 Совместимые МО подразделяются на :

 А) функциональные;

 Б) структурные.



# Микрооперации

---

- 📄 Функциональная совместимость определяется алгоритмом. Две МО будут такими, если они присваивают значения разным словам.
- 📄  $A := A + B$  и  $A := C - D$  не совместимые операции

# Микрооперации


---

-  Структурная совместимость ограничивается аппаратурно.
-  Две МО считаются структурно совместимыми, если они выполняются на разных аппаратных средствах.


# Микрооперации

---

## **Логические условия.**

 Логические условия представляют из себя булеву функцию. В качестве первичных булевых функций выступают одноразрядные слова, поля, отношения.


 Отношение – конструкция вида:

  $C1 * C2$ , где  $*$  - операция отношения: больше, меньше, не равно.




## Содержательный граф функциональной микропрограммы.

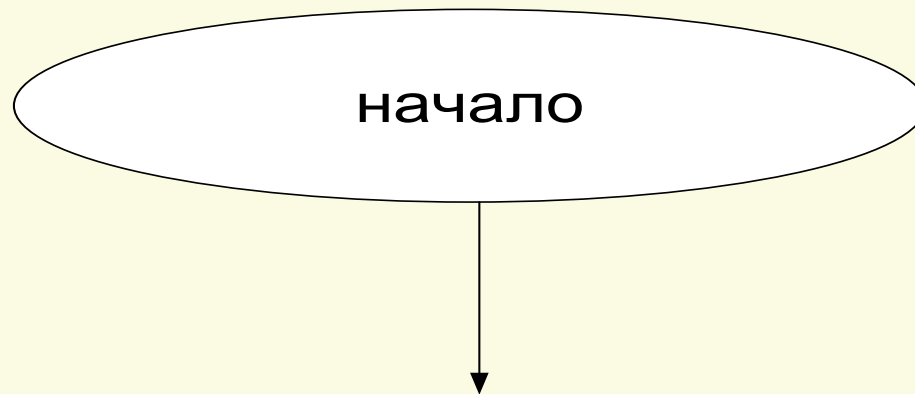
---

 Для записи графа используются 4 типа вершин и дуги, связывающие эти вершины. С их помощью описывается микропрограмма.

# Содержательный граф функциональной микропрограммы.


## Типы вершин графа .

-  1. Вершина «начало». Определяет начало микропрограммы, не имеет входов, и имеет единственный выход.



# Содержательный граф функциональной микропрограммы.


## 2. Функциональная вершина.

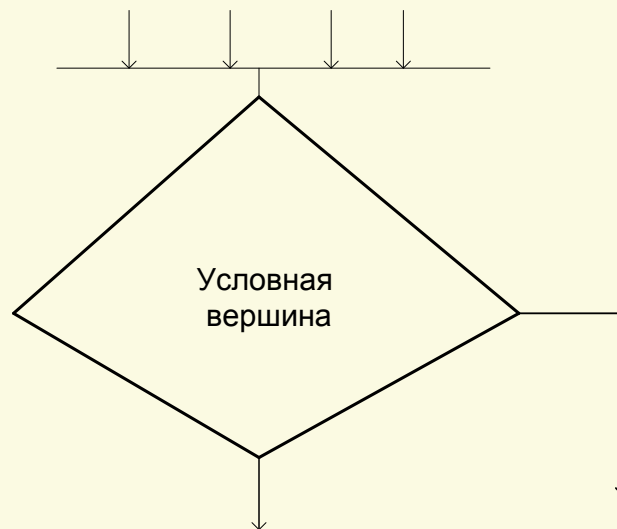
-  Используется для указания совместных МО, имеет произвольное количество входов и один выход.



# Содержательный граф функциональной микропрограммы.

## 3. Условная вершина.

-  Используется для описания разветвлений в МК, может иметь произвольное число входов и один единственный выход.



# Содержательный граф функциональной микропрограммы.




## 4. Конечная вершина.

 Может иметь произвольное число входов и не иметь выхода.






# Содержательный граф функциональной микропрограммы.

---

-  Граф должен быть корректным, то есть не должен допускать зависание микропрограммы.
-  Правила построения графа микропрограммы.
-  1. Граф должен иметь только одну начальную и одну конечную вершину.

# Содержательный граф функциональной микропрограммы.

---



-  2. В каждую вершину, кроме начальной, должна входить хотя бы одна дуга.
-  3. Из каждого выхода каждой вершины должна исходить одна и только одна дуга.
-  4. При любом наборе исходных данных должен существовать путь из начальной вершины в конечную.

# Этапы разработки ФМП

---

 Рассмотрим данные этапы на примере операции умножения.




 Исходные данные:

-  1. умножение производится над операндами одинаковой длины, целые числа со знаками;
-  2. умножение осуществляется в прямых кодах над модулями аргументов операции;



# Этапы разработки ФМП

---

-  3. Произведение занимает двойную длину слова аргументов.
-  4. Знак результата умножения определяется как арифметическая сумма битов знаков (сложение по модулю 2).
-  5. Пример умножения в столбик:

# Этапы разработки ФМП



1011



\*1101



1011



1011




1011




10001111



# Этапы разработки ФМП

---

 Словесное описание алгоритма умножения.




 1. Обнулить регистр С (хранит промежуточные суммы и и будет содержать старшие разряды результата).

# Этапы разработки ФМП

-  2. Множимое располагается в регистре А. Множитель в регистре В. Знаковые разряды устанавливаются в нулевые значения. (умножение производится над модулями сомножителей).
-  3. Если младший разряд множителя ( $B(0)$ ) равен 1, то производится сложение множимого и содержимого регистра С, при нулевом значении сложение не производится.






# Этапы разработки ФМП

---

-  4. Производится сдвиг вправо регистров С и В, при этом сдвигаемый младший разряд регистра С переносится в старший освобождаемый разряд регистра В.
-  Повторяются п.3 и п.4. столько раз, сколько разрядов содержит множитель.
-  5. Знак результата определяется как сумма по модулю 2 старших знаковых разрядов аргументов А и В и записывается в старший разряд регистра С.




# Этапы разработки ФМП

## Описание слов.

 Тип слова	Формат	Поля	Примечание
 I	A(4:0)	 $ЗнA=(4)$ $mA=A(3:0)$	множи- мое
 I	B(4:0)	 $ЗнB=(4)$ $мB=B(3:0)$	множи- тель





# Этапы разработки ФМП

## Описание слов.

 Тип слова	Формат	Поля	Примечание
 О	C(4:0)	ЗнС=(4) мС=C(3:0)	старшая часть рез.
 L	Сч(2:0)		счётчик

# Логическое проектирование операционного автомата (ОА)


## Структурный базис ОА

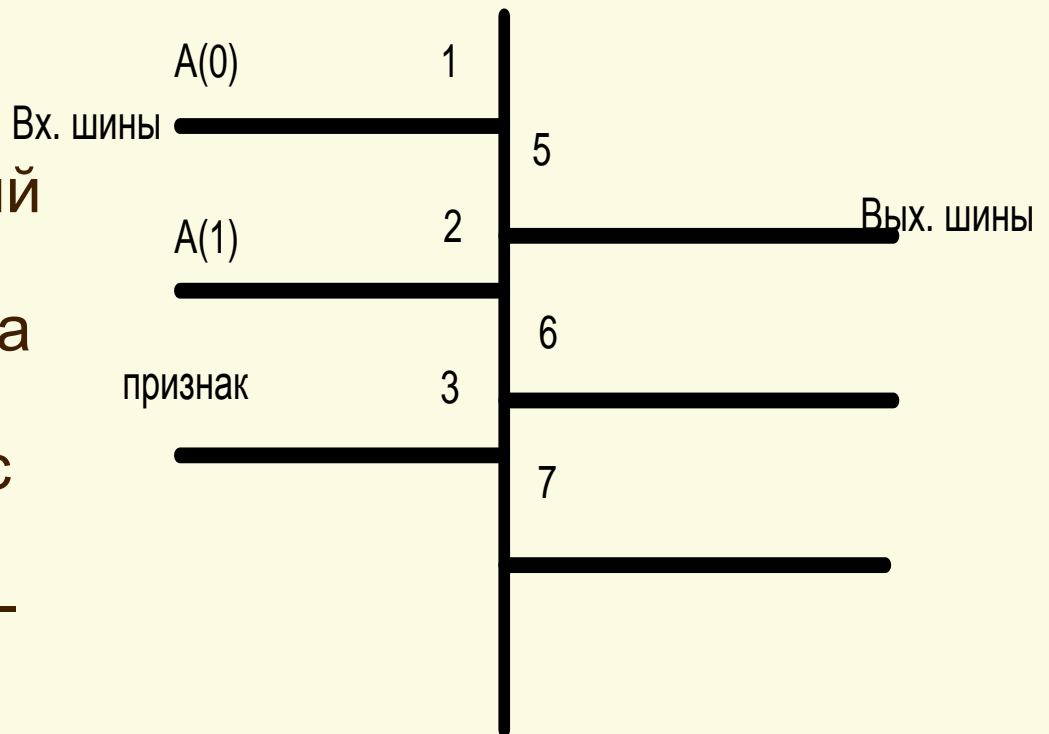
-  Структурный базис ОА - это набор элементов , из которых построен ОА:
-  а) триггеры, регистры, счётчики, и др.,
  -  б) комбинационные схемы;
  -  в) шины.



# Шины. Архитектуры шин.



## УГО шины.

 Каждая шина должна иметь свой собственный уникальный идентификатор, а также входные и выходные цепи с собственными идентификаторами.




# Шины. Архитектуры шин.

---

-  Шины могут изгибаться, разветвляться, пересекаться.
-  По шинам можно передавать информацию от одного источника к нескольким приёмникам.  
(управляемое демультиплексирование).

# Шины. Архитектуры шин.


---

 По шинам можно передавать информацию от многих источников к одному приёмнику. (управляемое мультиплексирование).

# Порядок проектирования ОА

---





## 1. Описание ОА.

 В общем случае ОА может выполнять несколько ФМП(МП)

# Порядок проектирования ОА

 МП	слова	МО	ЛУ
 МП1	A(15.0)	A:=B	A=0
	B(15.0)	A:=A+B	
	-----		
 МП2	A(7:0)	A:=B	B=1
	B(7:0)	A:=A-B	
	-----		

# Порядок проектирования ОА


 МП	слова	МО	ЛУ
 МПЗ	A(15.0)	A:=B	A=0
	B(15.0)	A:=A+B	B=1
		A:=A-B	D=1

# Методика синтеза канонической структуры ОА

---

 Исходные данные:

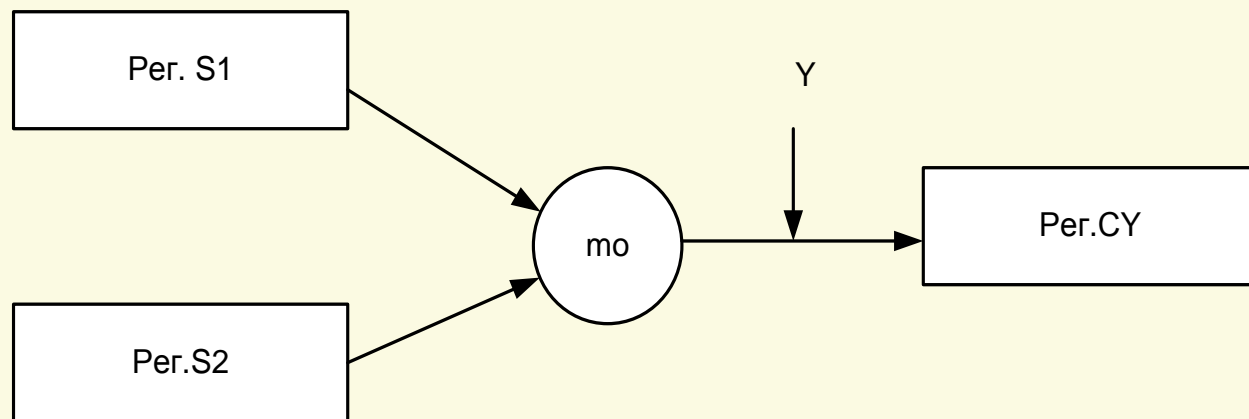
  $S$ ;  $Y$ ;  $X$ ; структурный базис.

 1) Выделение регистров под входные слова и определение разрядов и регистра под выходное слово.

# Методика синтеза канонической структуры ОА

2) Каждой микрооперации вида:

$$S_{\alpha} := \varphi_m(S_1, \dots, S_k)$$



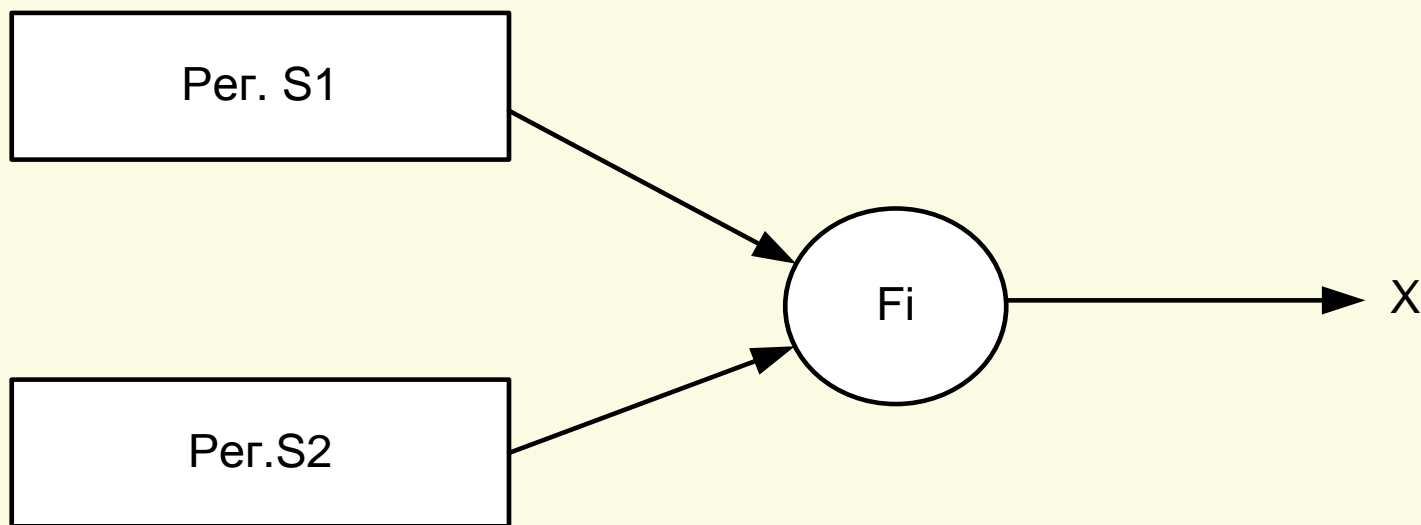


# Методика синтеза канонической структуры ОА



3). Каждому логическому условию (ЛУ) вида:

$$X_l = \varphi_l(S_1, \dots, S_k)$$



# Синтез ОА для блока умножения

---

 Составление алгоритма умножения

# Синтез ОА для блока умножения


---

 Описание блока умножения:

 Составление таблицы:



---

 Слово	МО	У	ЛУ	Х
---	----	---	----	---



---


# Синтез канонической структуры блока умножения


---

 Рисунок структуры блока

# Порядок работы ОА


---





 В ОА в каждом машинном такте выполняется одна или несколько МО, причём сама МО с синтаксической точки зрения представляет собой оператор присваивания:

 *<адрес>:=<двоичное выражение>*

# Порядок работы ОА

---


 ОА работает по тактам. В такте происходит:


-  1) формирование управляющих сигналов  $Y_1, \dots, Y_n$ ;
-  2) вычисление значения двоичного выражения (МО);
-  3) сохраняется результат в регистре;
-  4) вычисляются логические условия и определяются значения условных переменных  $X_1, X_2, \dots, X_n$ .

# Порядок работы ОА

---




 Длительность машинного такта:

  $T_{\text{MT}} > t_y + t_{\text{MO}} + t_{\text{ЛУ}}$

 В конце такта запись в регистр производится по синхроимпульсу тактового генератора.

# Характеристики ОА

---


-  1. Производительность (количество МО за такт).
-  2. Быстродействие (длительность такта).
-  3. Затраты оборудования.



# Структурная организация ОА.



---

 ***Классификация структур автоматов:***

 ***1. I –автоматы.*** Их производительность такая же как и у автоматов с канонической структурой. Особенностью является отсутствие избыточности и как следствие меньшие аппаратные затраты.


# Структурная организация ОА.


---

-  2. М –автоматы: в каждом машинном такте может выполняться только одна МО, следовательно производительность = 1 (очень мала), но минимальны аппаратурные затраты.
-  3. IM – автоматы с промежуточными характеристиками: производительность  $>1$ , но имеются структурные ограничения на совместимость МО.

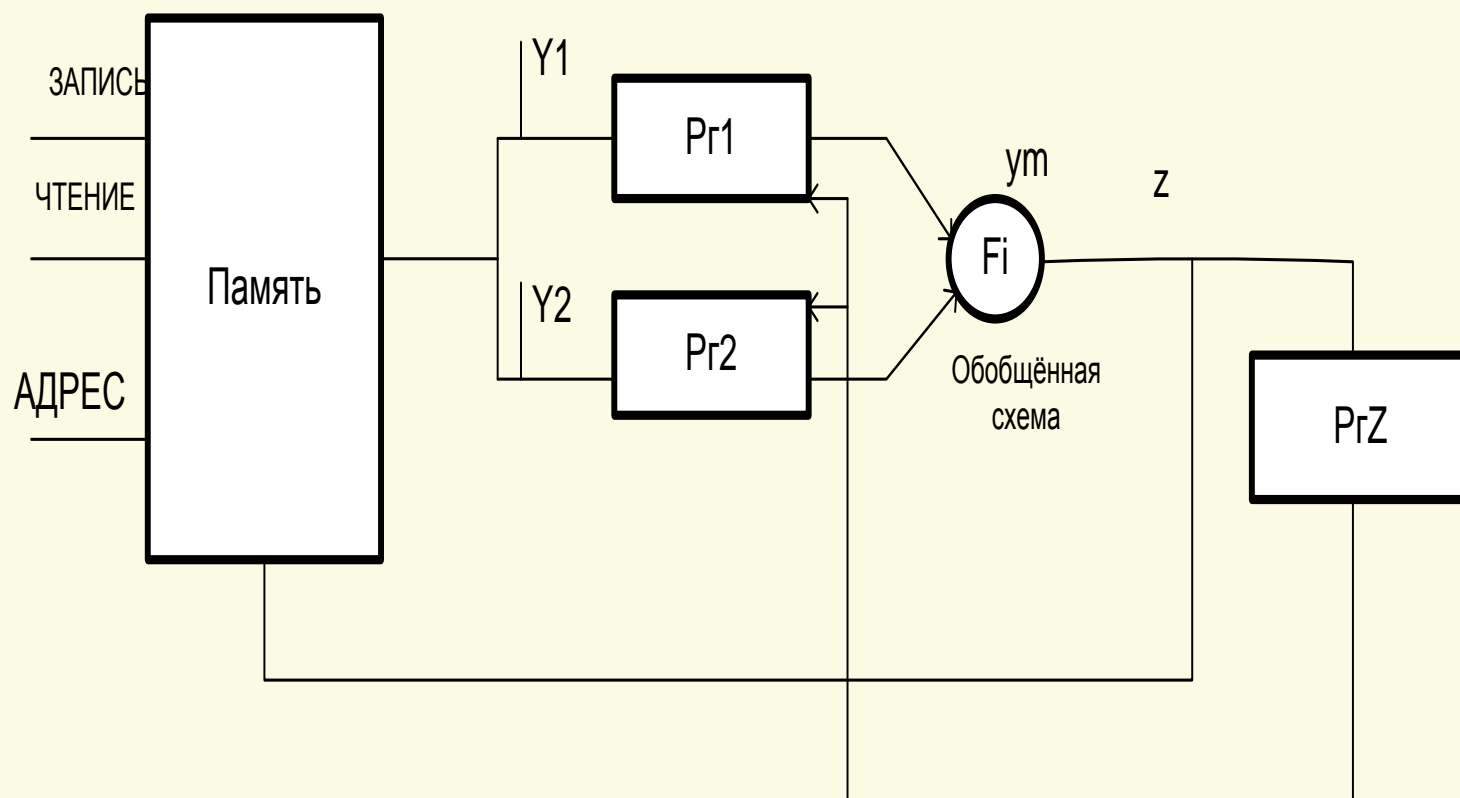
# Структурная организация ОА.

---

 4. S - автоматы: используются, когда надо обрабатывать большое количество слов. (например, каналы ввода/вывода).

 В этом случае для хранения используются не регистры, а память.

# Структурная организация ОА.



# Построение ОЭ на основе регистра

---

 Возможные операции на регистре:

 1).  $C := A$

 2).  $C := 0$

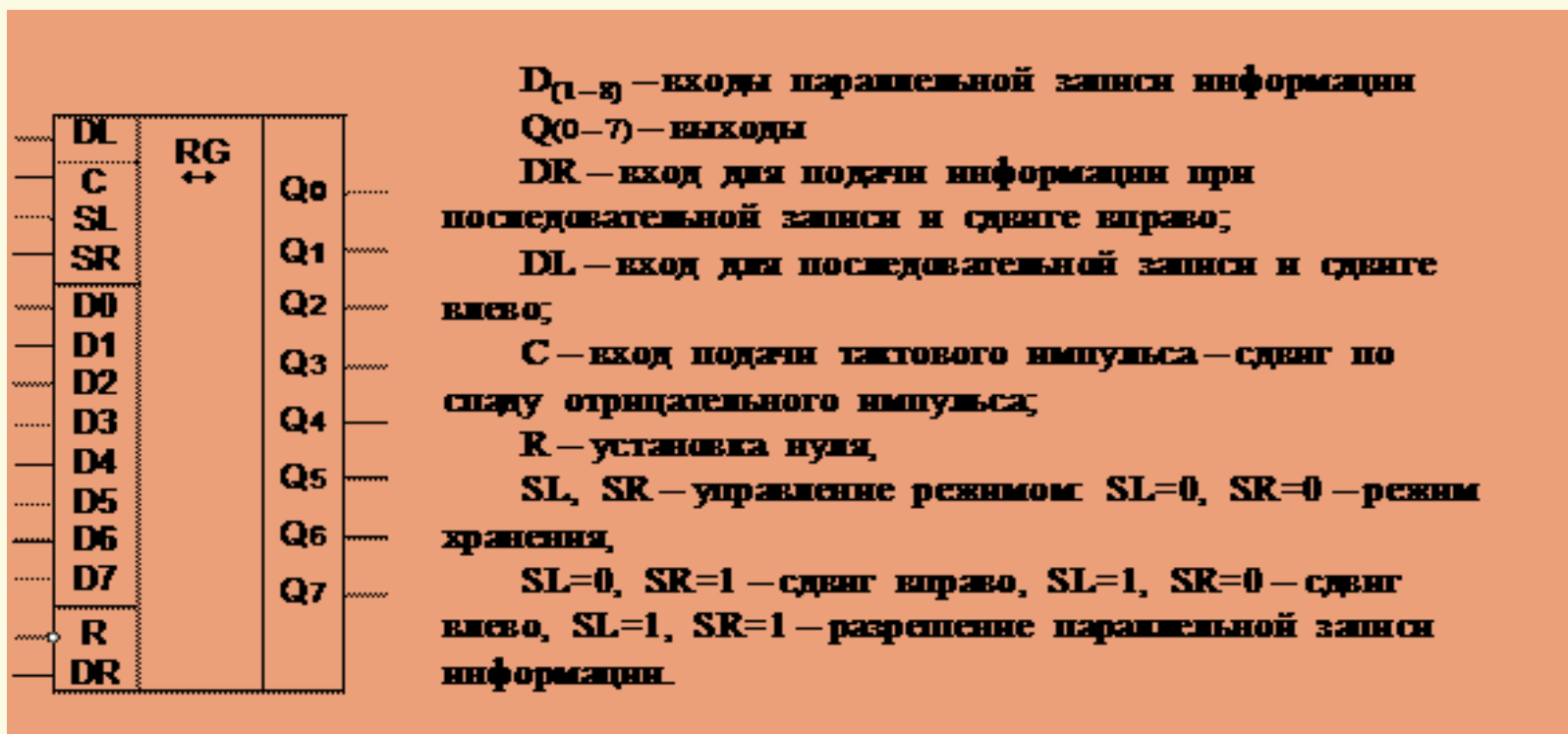
 3).  $C := R1(1.C)$

 4).  $C := R1(0.C)$

 5).  $C := L1(C.0)$

# Построение ОЭ на основе регистра

Берём за основу универсальный регистр



# Таблица функций регистра

Режимы	S0	S1	R	C	Микроопе- рация
Хранение	0	0	0	*	$F := F$
Сдв. Влево	0	1	0	1	$F := L1(F.DL)$
Сдв. вправо	1	0	0	1	$F := R1(DR.F)$
Запись	1	1	0	1	$F := D$
Сброс	*	*	1	*	$F := 0$


# Таблица описания работы ОЭ


Y	MO	S0	S1	R	DL	DR
Y1	C: = A	1	1	0	*	*
Y2	C: = 0	*	*	1	*	*
Y3	C: = R1(1.C)	1	0	0	*	1
Y4	C: = R1(0.C)	1	0	0	*	0
Y5	C: = L1(C.0)	0	1	0	0	*




# Синтезируем комбинационную схему

---

  $S_0 = y_1 + y_3 + y_4$

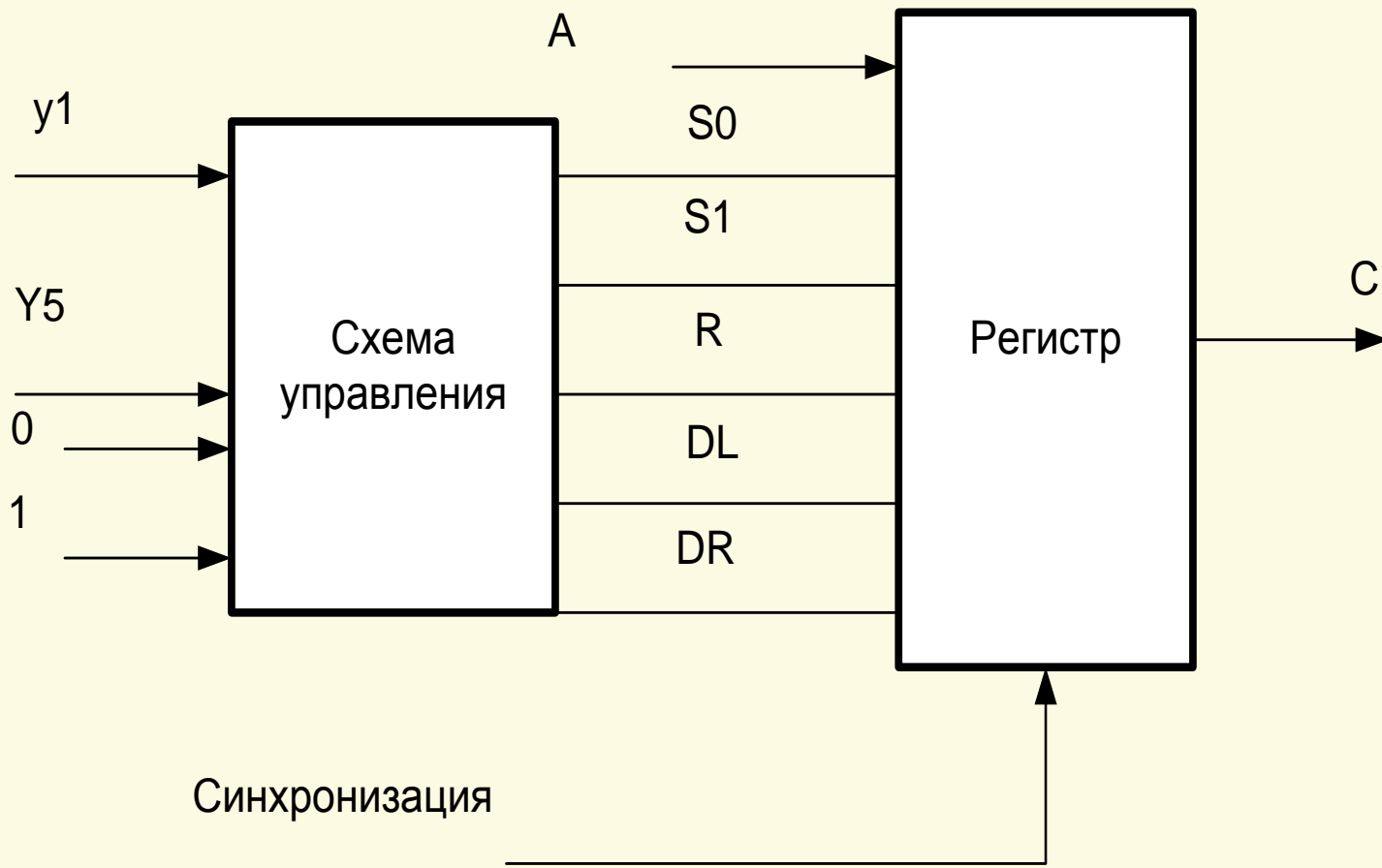
  $S_1 = y_1 + y_5$

  $R = y_2$

  $DL = 0$

  $DR = y_3$

# Синтезированная структура ОЭ



# Синтез операционного элемента на основе счётчика

## Режимы счётчика

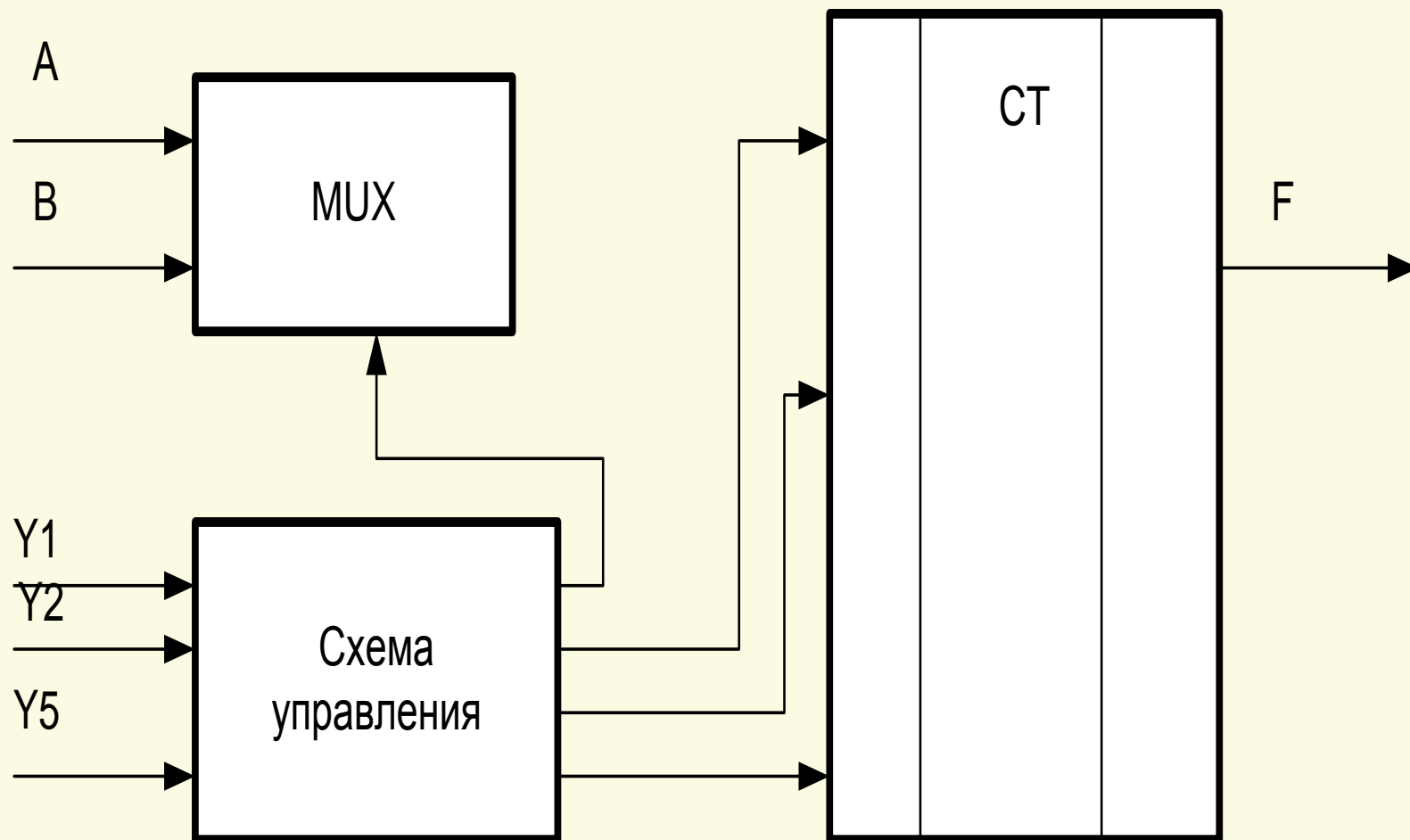
Режимы	S0	S1	R	C	Микрооперация
Запись	1	1	0	1	$F := D$
Сложение	1	0	0	1	$F := F + 1$
Вычитание	0	1	0	1	$F := F - 1$
Хранение	0	0	0	*	$F := F$
Сброс	*	*	1	*	$F := 0$

# Синтез операционного элемента на основе счётчика

 Таблица микроопераций

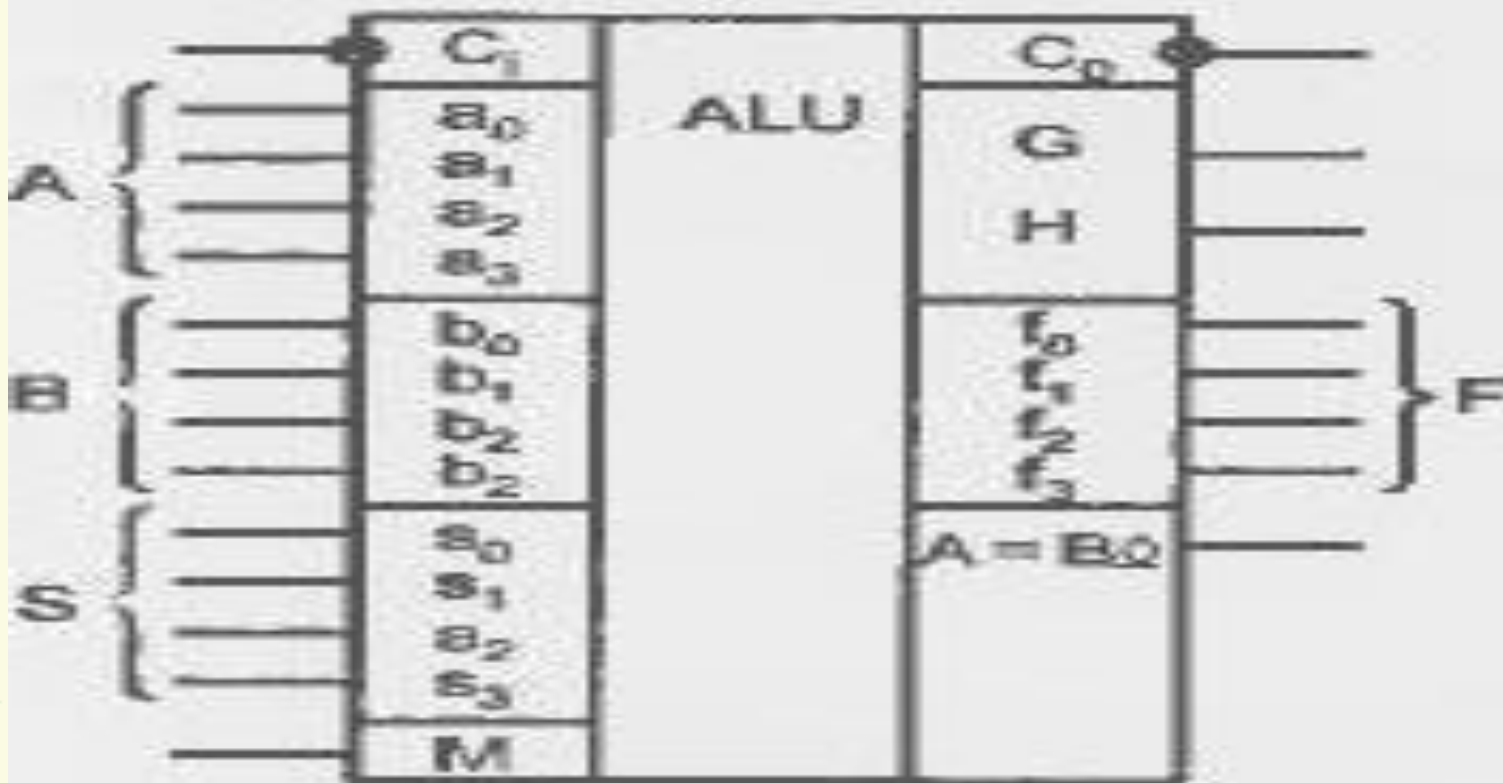
Y	Микр. операция	S0	S1	R	+1	Sm	C
Y1	F:=A	1	1	0	*	0	1
Y2	F:=B	1	1	0	*	1	1
Y3	C:=C+1	1	0	0	1	*	1
Y4	C:=C-1	0	1	0	1	*	1
Y5	C:=0	*	*	1	*	*	*

# Синтез операционного элемента на основе счётчика



# Синтез операционного элемента на основе АЛУ

📄 Интерфейс микросхемы АЛУ



# Операции АЛУ



Перечень выполняемых АЛУ операций дан в след.таблице. Для краткости двоичные числа  $s_3s_2s_1s_0$  представлены их десятичными эквивалентами. Под утолщенными обозначениями 1 и 0 следует понимать наборы 1111 и 0000, входной перенос поступает в младший разряд слова, т. е. равен 0000<sub>Si</sub>. При арифметических операциях учитываются межразрядные переносы.

# Операции АЛУ

S	Логические функции (M = 1)	Арифметико-логические функции (M = 0)
0	$A$	$A + C_i$
1	$A \vee B$	$A \vee B + C_i$
2	$AB$	$A \vee B + C_i$
3	0	$1 + C_i$
4	$AB$	$A + AB + C_i$
5	$B$	$A \vee B + AB + C_i$
6	$A \oplus B$	$A + B + C_i$
7	$AB$	$AB + 1 + C_i$
8	$A \vee B$	$A + AB + C_i$
9	$A \oplus B$	$A + B + C_i$
10	$B$	$A \vee B + AB + C_i$
11	$AB$	$AB + 1 + C_i$
12	1	$A + A + C_i$
13	$A \vee B$	$A \vee B + A + C_i$
14	$A \vee B$	$A \vee B + A + C_i$
15	$A$	$A + 1 + C_i$



# Соединение нескольких АЛУ

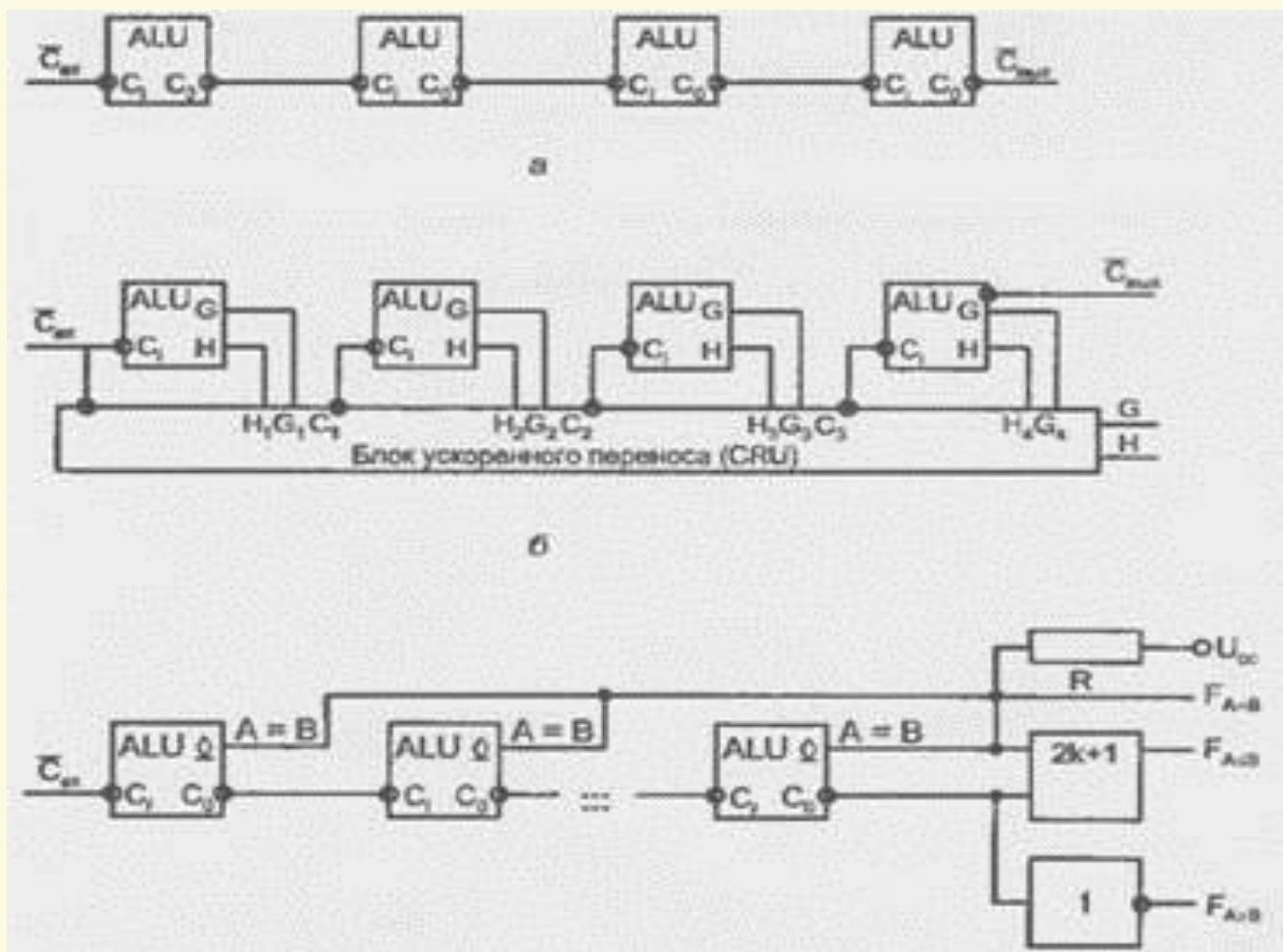


При операциях над словами большой размерности АЛУ соединяются друг с другом с организацией последовательных или параллельных переносов.



В последнем случае совместно с АЛУ применяют микросхемы — блоки ускоренного переноса, получающие от отдельных АЛУ функции генерации и прозрачности, а также входной перенос и вырабатывающие сигналы переноса.


# Соединение нескольких АЛУ





# Синтез операционного элемента на основе АЛУ


---


 Перечень микроопераций:


  $F=A+B$

  $F=A-B$

  $F=A+1$

  $F=A-1$

  $F=A \& B$

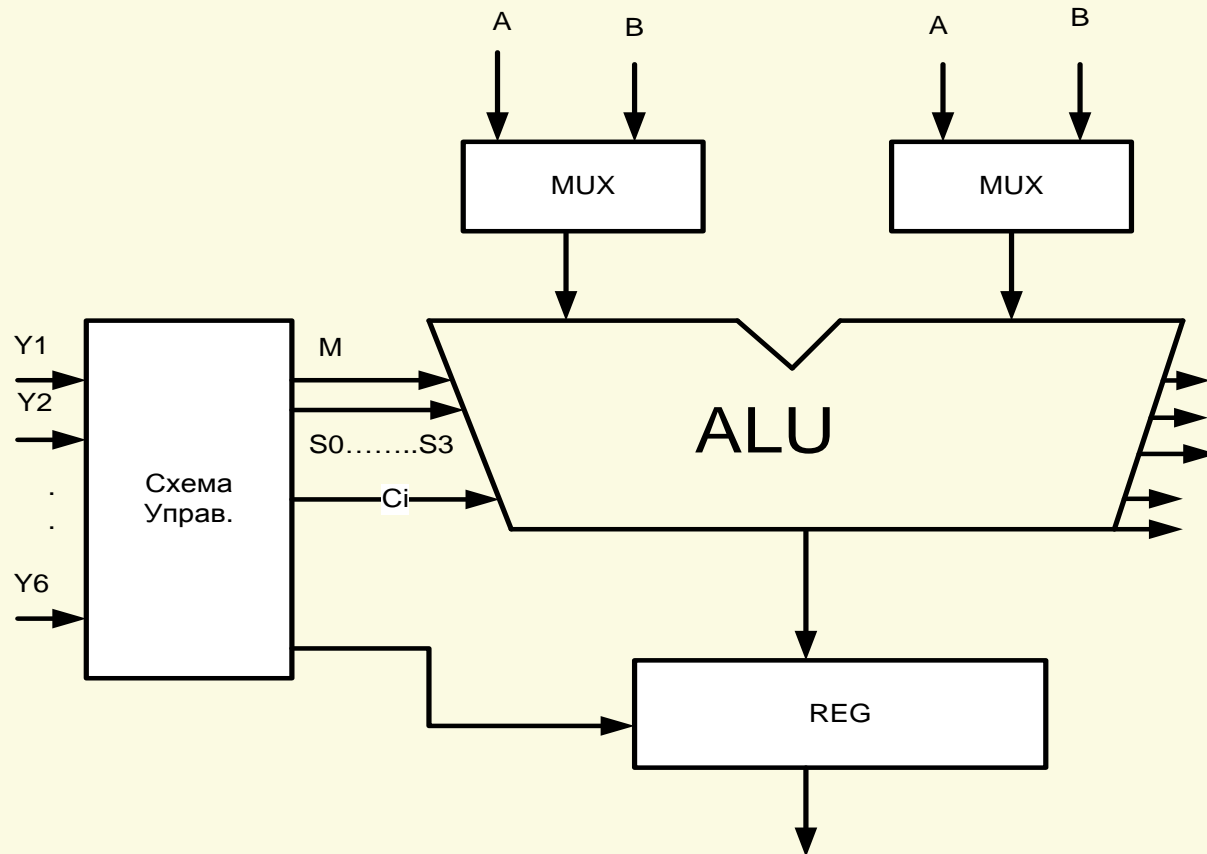
  $F=A \wedge B$

# Синтез операционного элемента на основе АЛУ

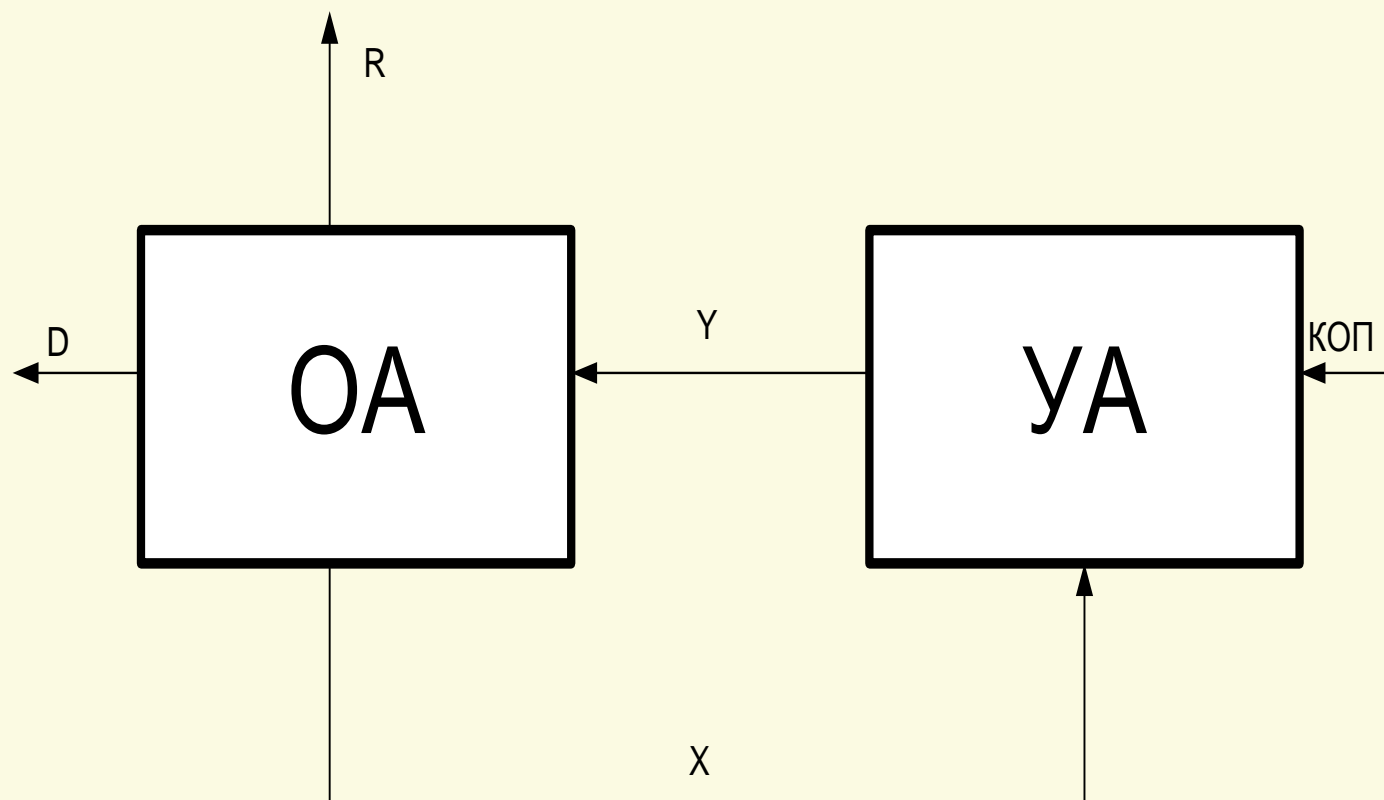
Таблица микроопераций для синтеза ОЭ

Y	Микро-операция	M	S0	S1	S2	S3	Ci	
Y1	$F=A+B$	0	0	0	0	1	0	
Y2	$F=A-B$	0	0	0	1	0	0	
Y3	$F=A+1$	0	0	0	0	0	1	
Y4	$F=A-1$	0						
Y5	$F=A \& B$	1						
Y6	$F=A \wedge B$	1						

# Синтез операционного элемента на основе АЛУ




# Управляющий автомат




# Типы цифровых автоматов


---

 Два типа автоматов:

 А) Цифровой автомат Мили;

 Б) Цифровой автомат Мура.

# Определение автомата Мили

 **Конечным детерминированным автоматом типа Мили** называется совокупность пяти объектов



,



где  $S$ ,  $X$  и  $Y$  — конечные непустые множества, а  $\delta$  и  $\lambda$  — отображения вида:



и



со связью элементов множеств  $S$ ,  $X$  и  $Y$  в абстрактном времени  $T = \{0, 1, 2, \dots\}$  уравнениями:






# Определение автомата Мили

- ❏ (Отображения и получили названия, соответственно функции переходов и функции выходов автомата  $A$ ).
- ❏ Особенностью **автомата Мили** является то, что функция выходов является двухаргументной и символ в выходном канале  $y(t)$  обнаруживается только при наличии символа во входном канале  $x(t)$ . Функциональная схема не отличается от схемы абстрактного автомата.

# Цифровой автомат Мура

 Зависимость выходного сигнала *только от состояния* представлена в автоматах типа Мура (англ. *Moore machine*).

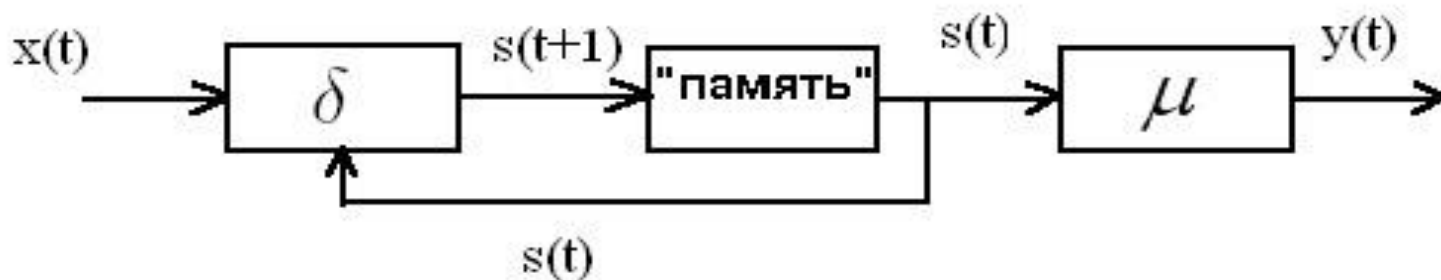
В **автомате Мура** функция выходов определяет значение выходного символа только по одному аргументу — состоянию автомата. Эту функцию называют также функцией меток, так как она каждому состоянию автомата ставит метку на выходе.

# Цифровой автомат Мура

- Конечным детерминированным автоматом типа Мура называется совокупность пяти объектов:  $A = (S, X, Y, \delta, \mu)$ ,
- где  $S$ ,  $X$ ,  $Y$  и  $\delta$  — соответствуют определению автомата типа Мили, а  $\mu$  является отображением вида:  $\mu : S \rightarrow Y$ ,
- с зависимостью состояний и выходных сигналов во времени уравнением:
  - $y(t) = \mu(s(t)), t \in T$

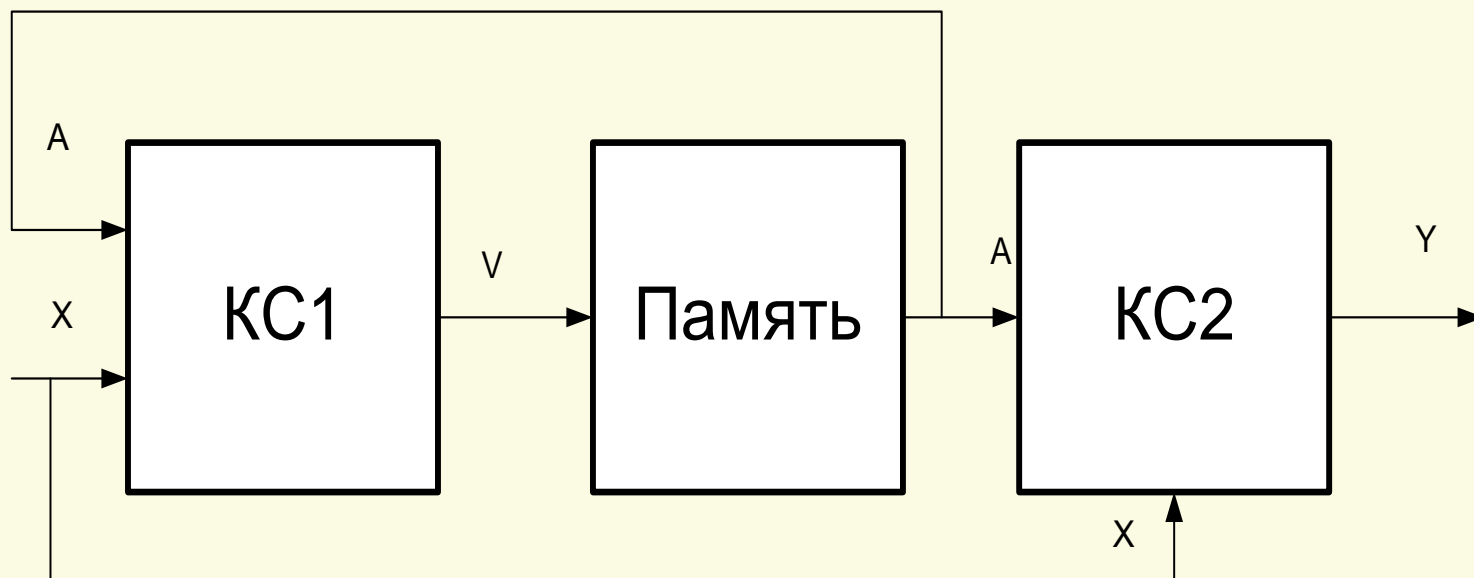
# Цифровой автомат Мура

Особенностью автомата Мура является то, что символ  $y(t)$  в выходном канале существует все время, пока автомат находится в состоянии  $s(t)$ .






# Структурная организация управляющих автоматов.

Построение цифрового автомата по схеме Мура.




# Построение цифрового автомата по схеме Мура.

---

-  КС1 управляет памятью;
-  КС2 формирует набор выходных сигналов;
-  V-сигнал возбуждения, управляющий памятью.

# Построение цифрового автомата по схеме Мура.

 Исходные данные для автомата Мура:

 ФМП хранит всю информацию.

$$Y = \{y_1, y_2, \dots, y_n\}$$

$$X = \{x_1, x_2, \dots, x_n\}$$

$$A = \{a_1, a_2, \dots, a_{n-1}\}$$

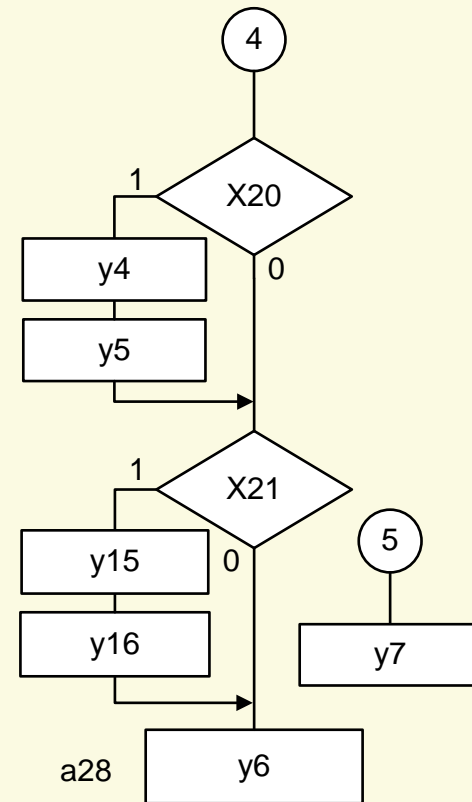
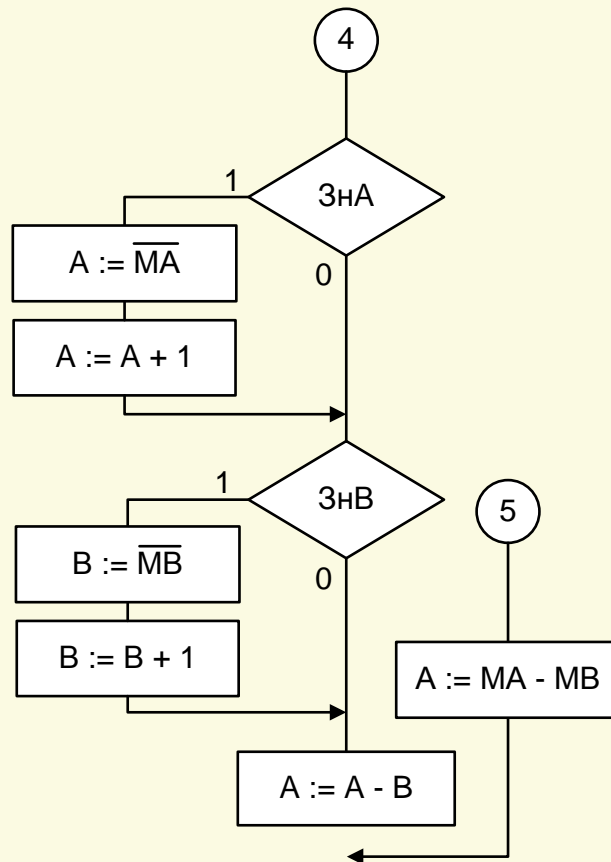
# Построение цифрового автомата по схеме Мура

---

- 📄 Формализуем МП с помощью закодированного графа, в котором каждая микрооперация (МО) заполняется соответственно  $Y$ ,
- 📄 логические условия - осведомительным сигналом  $X$ .






## Переход от содержательного графа к закодированному графу



# Этапы синтеза УА

---

-  1. Построение таблицы микроопераций и логических условий.
-  2. Построение закодированного графа ФМП путём замены каждой МО управляющим сигналом  $Y$ , каждого ЛУ осведомительным сигналом  $X$ .
-  3. На закодированном графе пометить вершины индексами  $a_0, a_1 \dots a_{n-1}$ .



# Этапы синтеза УА

---

- 4. Построение графа автомата МУРА:
- Каждому состоянию поставить в соответствие вершину графа;
- Каждому переходу поставить в соответствие дугу графа.


# Этапы синтеза УА

---

-  5. Построение списка переходов в табличной форме. В таблице каждая дуга соответствует строке таблицы с указанием условия,  $Y$ , и состояния
-  (а нач. и а конеч.)

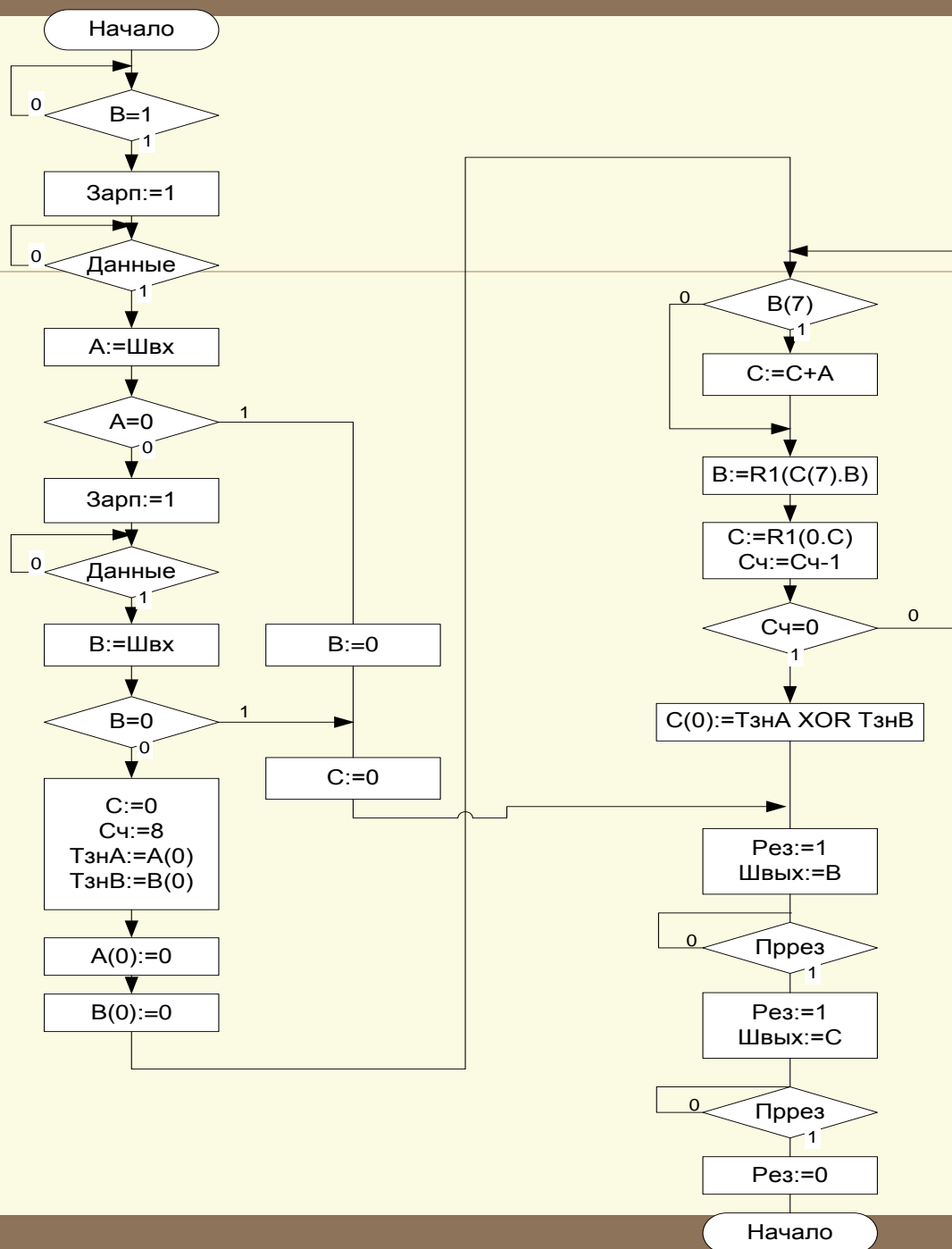
# Организация памяти цифрового автомата

---

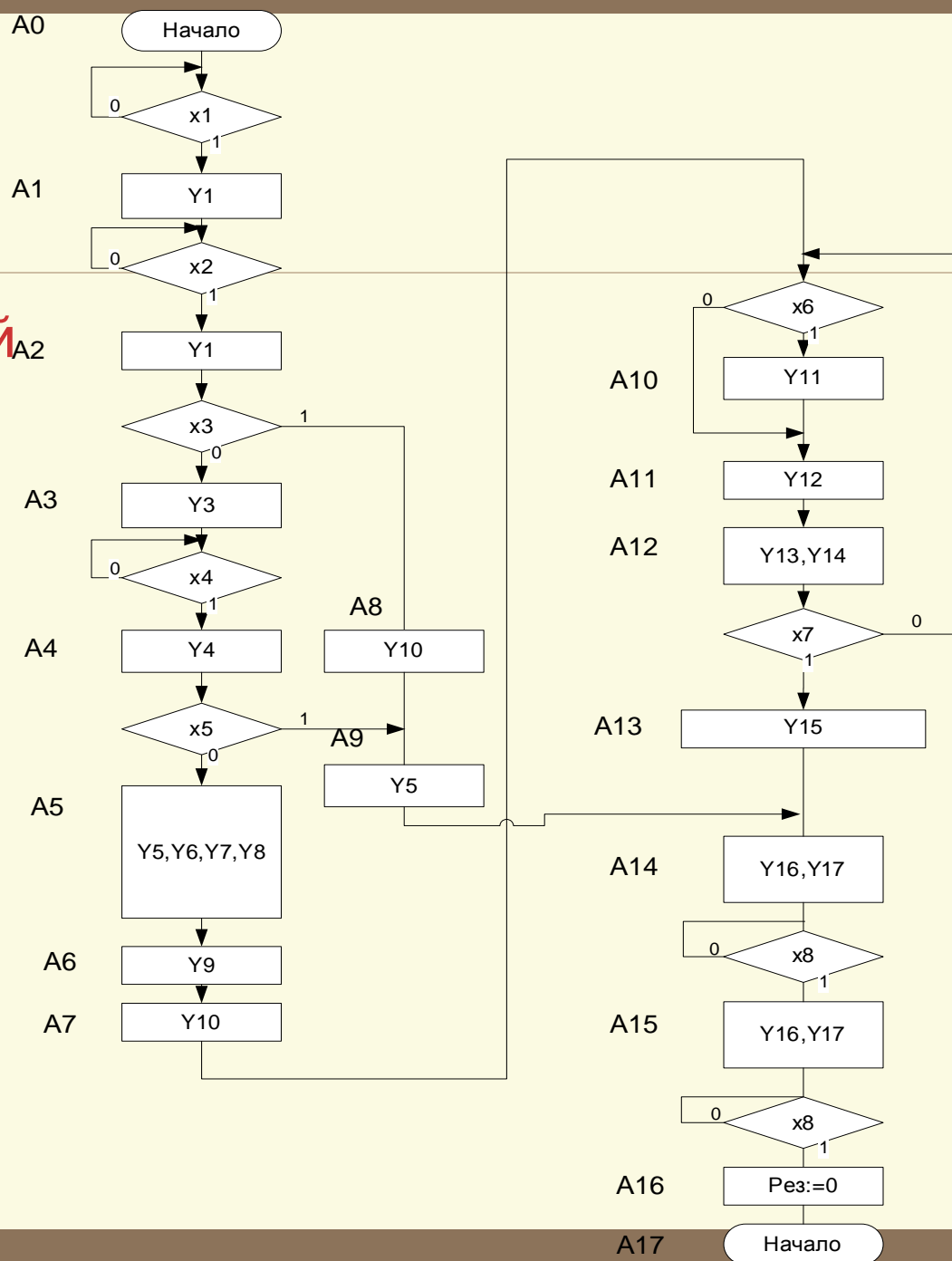
 Память ЦА строится на основе регистра. Разрядность регистра определяется формулой, где  $P$  - количество состояний ЦА (количество вершин графа):

$$N = \lceil \log P \rceil$$

# Пример синтеза УА для умножения



## Закодированный граф ФМП



## Составление списка (таблицы) переходов

№	Исх. состояние	Код исходного состояния	следующее состояние	Код след. состояния	X	Y	Сигнал возбужде- ния
1	a0	00000	a0	00000	!x1	-	-
2	a0	00000	a1	00001	x1	-	D0
3	a1	00001	a1	00001	!x2	Y1	D0
4	a1	00001	a2	00010	x2	Y1	D1
5	a2	00010	a3	00011	!x3	Y1	D1,D0
6	a2	00010	a8	01000	x3	Y1	D3
7	a3	00011	a3	00011	!x4	Y3	D2,D1
8	a3	00011	a4	00100	x4	Y3	D2



## Составление списка (таблицы) переходов

№	Исх. состояние	Код исходного состояния	следующее состояние	Код след. состояния	X	Y	Сигнал возбужде- ния
9	a4	00100	a5	00101	!x5	y4	D2,D0
10	a4	00100	a9	01001	x5	y4	D3,D0
11	a5	00101	a6	00110	-	Y5,y6 ,y7,y8	D2,D1
12	a6	00110	a7	00111	-	y9	D2,D1,D0
13	a8	01000	a9	01001	-	y10	D3,D0
14	a7	00111	a10	01010	x6	y10	D3,D1
15	a7	00111	a11	01011	!x6	y10	D3,D1,D0
16	a9	01001	a14	01111	-	y5	D3,D2,D1, D0

## Составление списка (таблицы) переходов


№	Исх. состояние	Код исходного состояния	следующее состояние	Код след. состояния	X	Y	Сигнал возбуждения
17	a10	01010	a11	01011	-	y11	D3,D1,D0
18	a11	01011	a12	01100	-	y12	D3,D2
19	a12	01100	a13	01101	x7	Y13, y14	D3,D2,D0
20	a12	01100	a10	01010	!x7, x6	Y13, y14	D3,D1
21	a12	01100	a11	01011	!x7, !x6	Y13, y14	D3,D1,D0
22	a13	01001	a14	01110	-	y15	D3,D2,D1
23	a14	01111	a14	01110	!x8	Y16, y17	D3,D2,D1
24	a14	01111	a15	01111	x8	Y16, y17	D3,D2,D1,D0

## Составление списка (таблицы) переходов

№	Исх. состояние	Код исходного состояния	следующее состояние	Код след. состояния	X	Y	Сигнал возбужде- ния
25	a15	01111	a15	01111	!x8	Y16, y17	D3,D2,D1 ,D0
26	a15	01111	a16	10000	x8	Y16, y17	D4
27	a16	10000	a0	00000	-	y18	-

# Функции сигналов возбуждения

---

  $D0 = a0 * x1 + a1 * !x2 + a2 * !x3 + a4 * !x5 + a4 * x5 + a7 * !x6 + a15 * !x8$

  $D1 =$


  $D2 =$  и т.д.


# Технология подготовки записей булевых функций для программирования ПЛМ


<i><b>X1</b></i>	<i><b>X2</b></i>	<i><b>X3</b></i>	<i><b>Y</b></i>
<i><b>0</b></i>	<i><b>0</b></i>	<i><b>0</b></i>	<i><b>0</b></i>
<i><b>0</b></i>	<i><b>0</b></i>	<i><b>1</b></i>	<i><b>1</b></i>
<i><b>0</b></i>	<i><b>1</b></i>	<i><b>0</b></i>	<i><b>1</b></i>
<i><b>0</b></i>	<i><b>1</b></i>	<i><b>1</b></i>	<i><b>1</b></i>
<i><b>1</b></i>	<i><b>0</b></i>	<i><b>0</b></i>	<i><b>0</b></i>
<i><b>1</b></i>	<i><b>0</b></i>	<i><b>1</b></i>	<i><b>0</b></i>
<i><b>1</b></i>	<i><b>1</b></i>	<i><b>0</b></i>	<i><b>1</b></i>
<i><b>1</b></i>	<i><b>1</b></i>	<i><b>1</b></i>	<i><b>1</b></i>

## Технология подготовки записей булевых функций для программирования ПЛМ

---

  $Y = !x_1 !x_2 x_3 + (!x_1 x_2 !x_3 + !x_1 x_2 x_3) +$

  $+ (x_1 x_2 !x_3 + x_1 x_2 x_3)$

  $Y = !x_1 !x_2 x_3 + !x_1 x_2 + x_1 x_2$

  $Y = !x_1 !x_2 x_3 + x_2$

# Технология подготовки записей булевых функций для программирования ПЛМ




Таблица  
сократилась:

x1	x2	x3	Y
0	0	1	1
*	1	*	1

## Технология подготовки записей булевых функций для программирования ПЛМ

---

 Данный подход можно распространить на систему булевых функций:


$$Y_i = f_i(x_1, x_2, \dots, x_5)$$

$$i = 1, \dots, 4$$



## Технология подготовки записей булевых функций для программирования ПЛМ

---

 Пусть заданы четыре булевых функции от 5-ти переменных.

## Технология подготовки записей булевых функций для программирования ПЛМ

---


x1	x2	x3	x4	x5		Y1	Y2	Y3	Y4
0	0	0	0	0		1	0	1	1
0	0	0	1	*		0	1	0	1
0	0	*	0	1		0	1	1	1
1	*	1	*	*		0	0	0	1


## Технология подготовки записей булевых функций для программирования ПЛМ


x1	x2	x3	x4	x5	Z	Y1	Y2	Y3	Y4
0	0	0	0	0	Z1	1	0	1	1
0	0	0	1	*	Z2	0	1	0	1
0	0	*	0	1	Z3	0	1	1	1
1	*	1	*	*	Z4	0	0	0	1


# Технология подготовки записей булевых функций для программирования ПЛМ


---

  $Y1=Z1;$

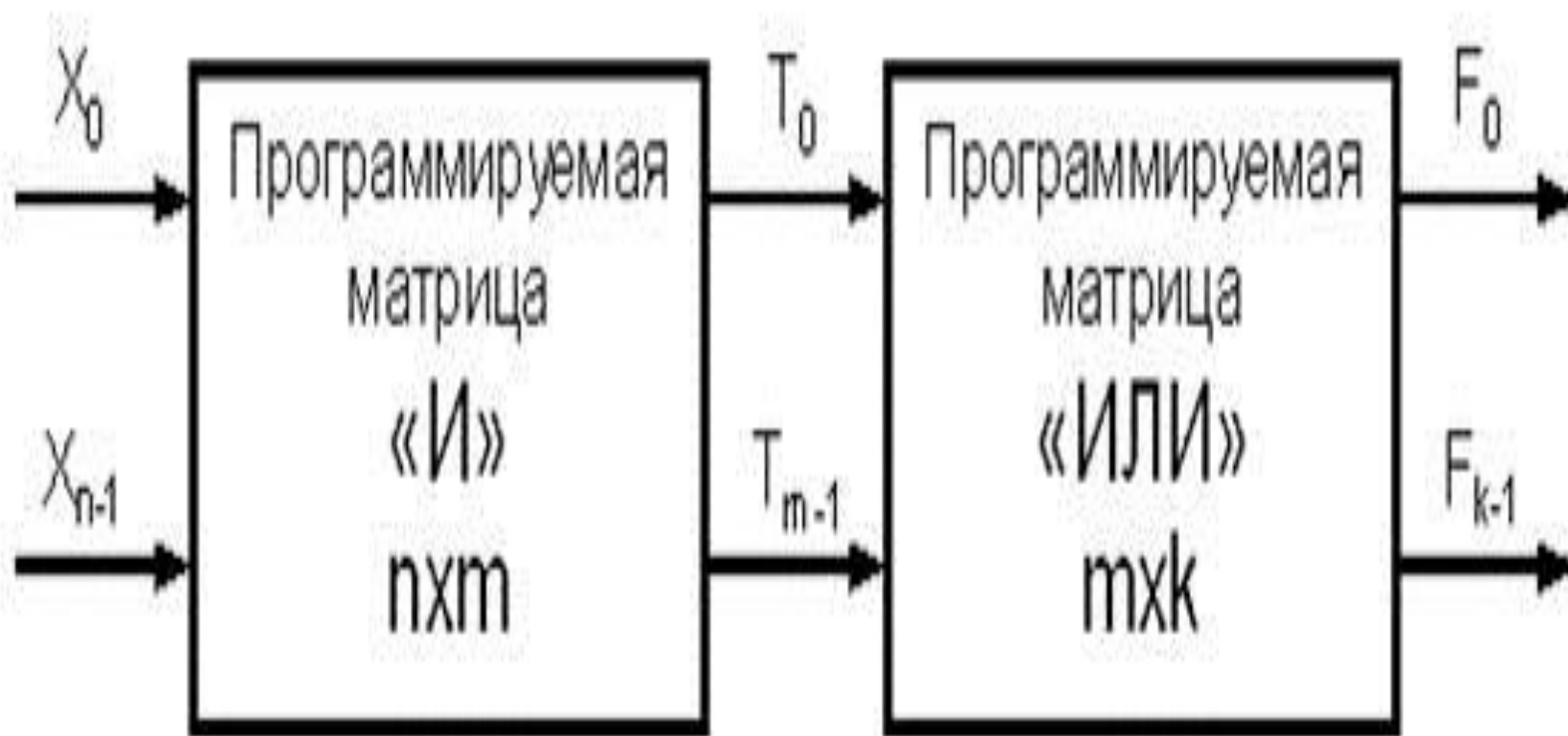
  $Y2=Z2+Z3;$

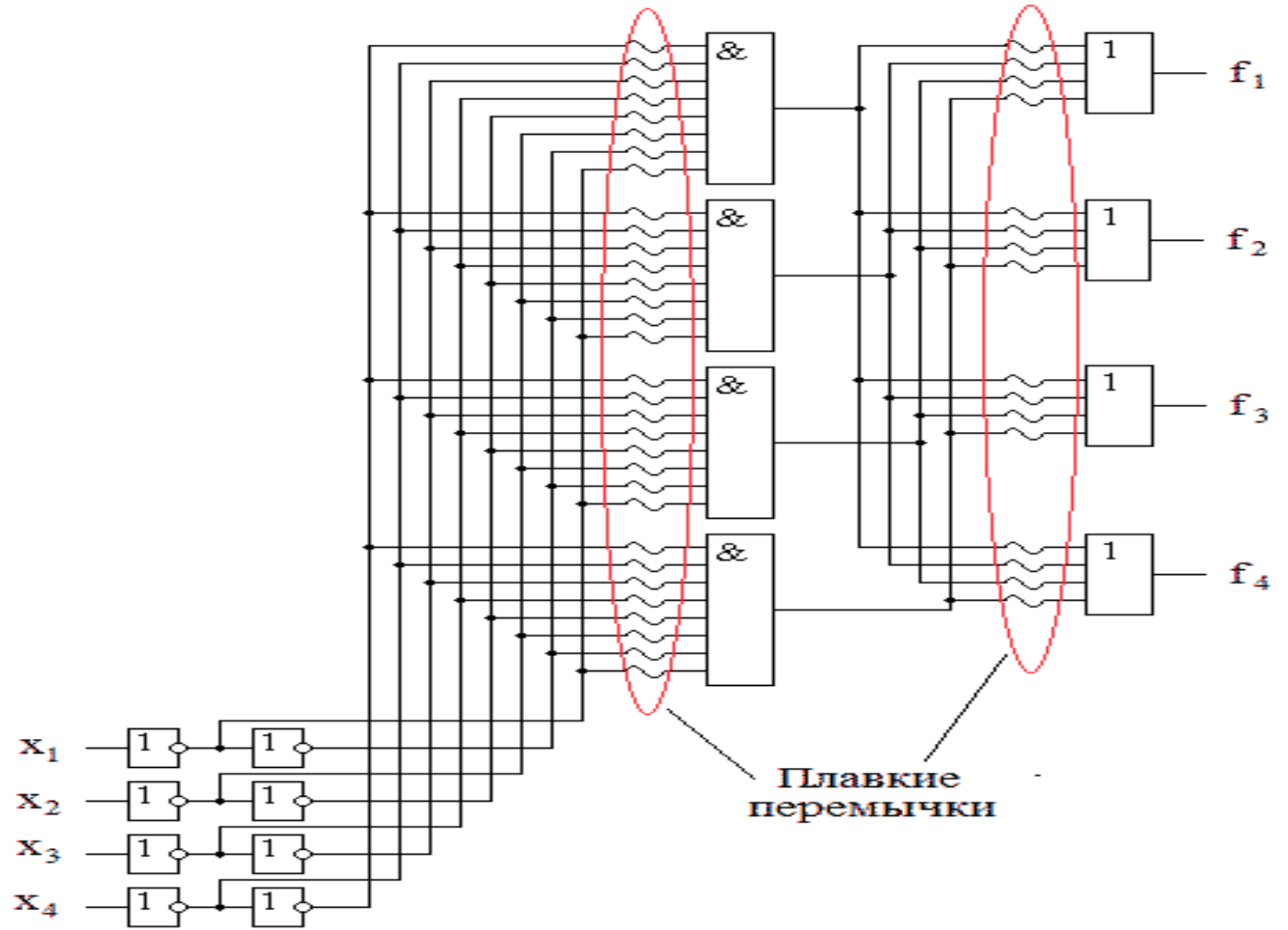
  $Y3=Z1+Z3;$

  $Y4=Z1+Z2+Z3+Z4$

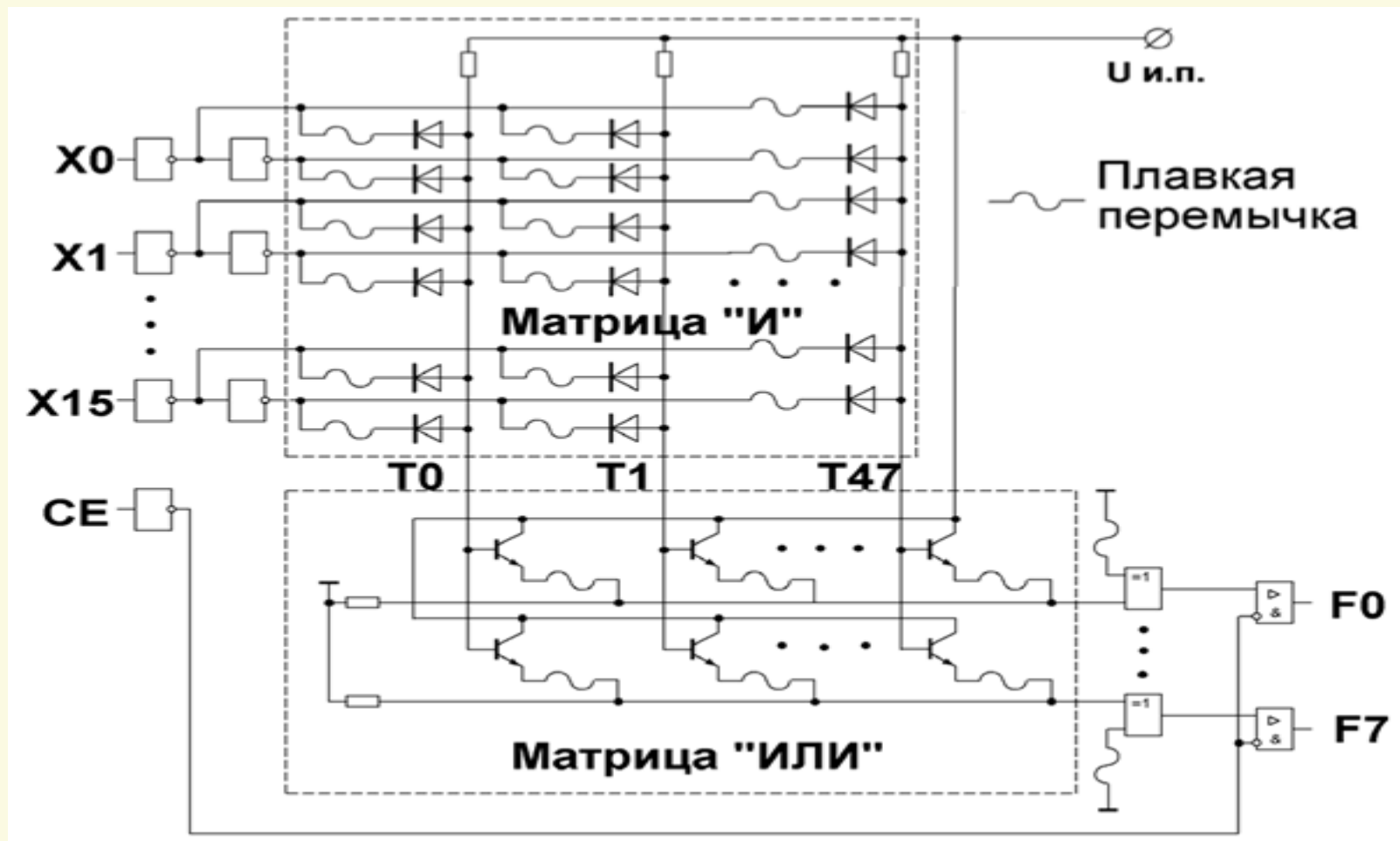
 ПЗУ реализуют произвольную логическую функцию, заданную в виде таблицы истинности, а ПЛМ – минимизированную логическую функцию

# Структура ПЛМ



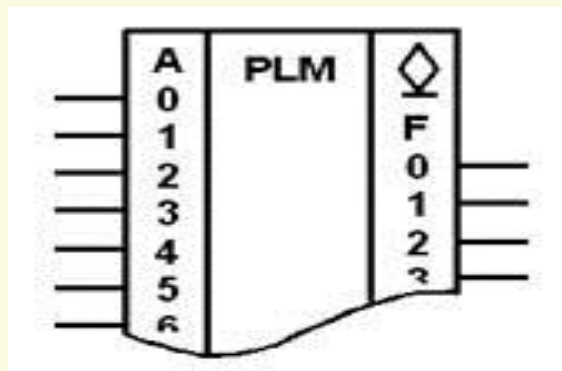


# Электрическая схема ПЛМ



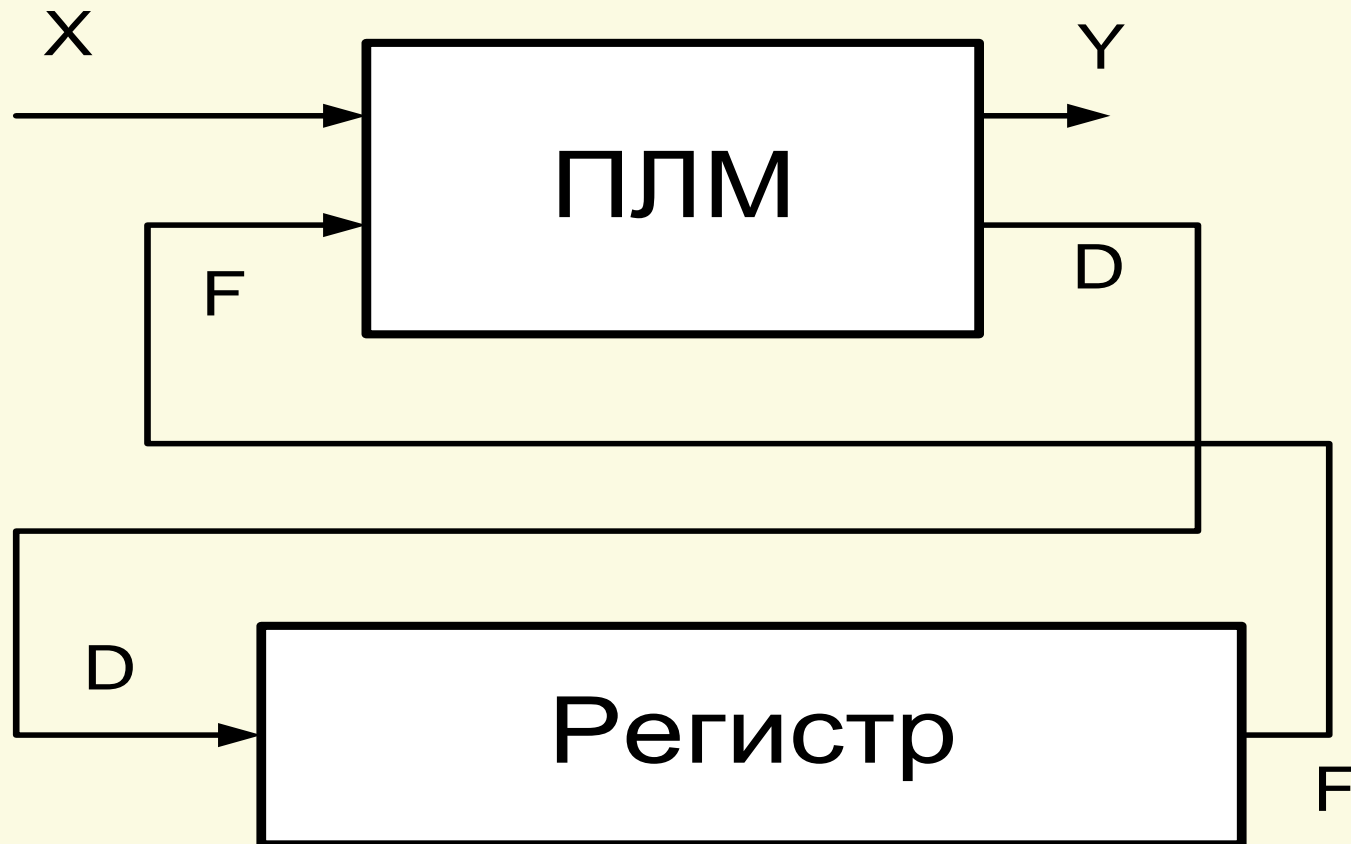
# УГО ПЛМ

Для того чтобы отличать ПЛМ от ПЗУ при изображении принципиальных электрических схем, в среднем поле условного графического обозначения пишется PLM





# Построение УА на ПЛМ



# Определение набора ПЛМ

- ПЛМ имеют 16 входов, 8 выходов и от 48 до 68 конъюнкций.
- Для определения необходимого количества (**Q**) ПЛМ, для реализации
- управляющего автомата, имеющего **K** разрядов регистра и **m** – число управляющих сигналов  $u$ , формируемых в автомате используется формула:


# Формула для определения необходимого количества ПЛМ

---

 Количество ПЛМ  $Q$ :

$$Q = \lceil (K + m) / 8 \rceil$$




# Порядок подготовки таблиц для программирования ПЛМ

 Распределить управляющие сигналы и сигналы возбуждения между всеми ПЛМ (сигналы возбуждения D1-Dn подавать на входы регистра состояний. Каждый из этих сигналов может быть закреплён только за одной ПЛМ.

# Подготовка таблиц ПЛМ

- Выполнить (виртуальное) программирование ПЛМ. Для каждой ПЛМ составить таблицу соединений.
- В каждой таблице указать входы, выходы и строки.
- На входы  $F1 - Fk$  всех ПЛМ подключить выходы регистра состояний (старшие слева- младшие справа). На остальные входы ПЛМ подключить осведомительные сигналы (условий), используемые в данной ПЛМ. Неиспользуемые входы ПЛМ не указывать. Число входов не должно превышать 16. Число выходов в каждой ПЛМ не должно превышать 8.

# Подготовка таблиц ПЛМ

-  В каждой строке входов прямое значение переменной кодировать единицей, инверсное – нулём, а безразличное звёздочкой.
-  На выходах единицей обозначать необходимость использовать данную конъюнкцию (строку) в булевой функции, описывающей соответствующую выходную переменную (управляющий сигнал  $Y$  или сигнал возбуждения  $D$ ).
-  Число строк в каждой матрице не должно превышать 68.