САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра
Системный Анализ и Управление

КОНСПЕКТ ЛЕКЦИЙ
«Архитектура ЭВМ и систем»

Составитель: В.Е. Баранов

Санкт-Петербург
2011

СОДЕРЖАНИЕ

Введение

Вычислительные машины позволяют решать весьма широкий круг задач, ограниченный способностью человека указывать инструкции для их решения. Подобные инструкции в форме точной и специфической последовательности операторов, подробно определяющих процедуру решения задачи, задаются программой.

На начальном этапе использования современной вычислительной системы мы имеем дело не с самой машиной, а с совокупностями правил, называемых языками программирования, на которых указываются действия, которые должна выполнять ЭВМ. При этом, сама вычислительная машина может рассматриваться как аппаратный интерпретатор некоторого конкретного языка, который называется **машинным** языком. Большинство пользователей используют языки высокого уровня (ЯВУ), которые с помощью программ-трансляторов (компиляторов или интерпретаторов) преобразуются в программы на машинном языке.

Таким образом, вычислительная машина состоит не только из вычислительной машины, представляющей собой комплекс оборудования, но и из программ, включая те, которые транслируют программы пользователей с любого из имеющихся языков в программы на машинном языке. То есть речь идет об аппаратной и программной частях ЭВМ.

Исторический обзор.

Механические средства для счета и вычисления были известны еще в далеком прошлом. Одним из древних средств подобного рода являются счеты, которые сохранились до наших дней в качестве простого практического приспособления для вычислений во многих частях света, в особенности на Востоке. (Разновидность счетов - абак - использовалась, видимо, древними египтянами и была знакома китайцам еще в Y1 в. до нашей эры). При этом, операции умножения и деления выполняются с помощью последовательностей операций сложения и вычитания.

Хотя в ранних устройствах большее внимание уделялось необходимости механизации арифметических операций, хранение промежуточных результатов было также важным. Большинство устройств, подобных счетам, хранили лишь простой текущий результат. В качестве памяти другого типа использовался любой писчий материал, например глиняные дощечки, а позднее бумага.

Первые предвестники управления последовательностями операций появились в областях далеких от вычислительной техники. Например, в ткацком станке Жаккарда (1801г.) использовались перфорированные (пробитые) карты для управления командами переплетения нитей в ткани.

Чарльз Бэббидж (1792-1871) был одним из первых, кто сформулировал идею создания универсальной вычислительной машины. Первым побудительным мотивом была невысокая надежность вычислений. Любопытство привело его к тому, что он обнаружил много ошибок в астрономомических таблицах. Бэббидж детально разработал проект универсальной вычислительной машины, но отсутствие средств не позволило ему реализовать проект.

Перепись населения в США в 1890 г. была революционной с точки зрения обработки ее результатов. Доктор Герман Холлерит применил перфорированные карты и простые машины для обработки данных переписи в 1890 г. С тех пор машины с перфорированными картами получили широкое распространение в деловой и административной сферах.

Первая треть двадцатого века ознаменовалась последовательным развитием и внедрением многих вычислительных устройств. Значительный вклад в это внес математик Алан Тьюринг, опубликовавший в 1937 году описание универсальной гипотетической вычислительной машины (схемы вычислений). Идея привлекла внимание многих ученых, но практического смысла в создание такого рода машины не было никого - она работала бы недопустимо медленно.

С 40-х годов 20-го века началось интенсивное развитие аппаратной и программной частей средств вычислительной техники (ВТ). Аппаратную часть ВТ можно условно разделить на пять поколений, связанных с техническим развитием ее компонентов.

В 1944 году Говард Айкен и группа исследователей из IBM построили электрическую вычислительную машину на релейных логических элементах. Чуть позднее в 1946 году Дж.П.Эккерт и Дж.В.Мочли разработали электронную вычислительную машину (ЭВМ) ENIAC на электронных лампах. Эти машины были предназначены для выполнения научных расчетов. Серийное производство ламповых ЭВМ ENIAC1 ассоциируют с термином "первое поколение". Примерно в это же время в СССР (1949-1951г.г.) были разработаны и реализованы под руководством академика С.А.Лебедева в АН УССР (Киев) первые ламповые ЭВМ МЭСМ (малая электронная счетная машина).

Появление транзисторов (изобретены в 1948 г.) привело к созданию ЭВМ " второго поколения. По сравнению с электронной лампой полупроводниковый транзистор является более эффективным элементом, так как: не требует нагрева источника электронов, имеет более продолжительный период жизни и высокую надежность, затраты на его производство существенно меньше. 1960 год считают годом широкого внедрения и использования ЭВМ "второго поколения", используемых в областях: общего назначения; для экономических задач; задач управления производственными процессами.

Третье и четвертое поколения технологии ЭВМ (относящиеся примерно к 1964 и 1970 г.г.) отличаются возрастающим применением методов интегрального изготовления с целью производства большей части ЭВМ в едином автоматическом непрерывном процессе без использования ручного труда. Третье поколение ЭВМ характеризуется широким применением операционных систем (ОС), а четвертое решением новых классов задач: имитационное моделирование; планирование; многопараметрическая оптимизация и т.д.

Пятое поколение ЭВМ связывают с понятием - "искусственного интеллекта". Ориентировочно можно говорить об их появлении в 1980-х годах. В этих машинах значительно увеличивается интеграция программной и аппаратной частей между собой, а также появляются возможности общения с внешней средой.

Успехи в разработке оборудования сопровождались достижениями в программировании. Все языки программирования можно разделить на: машинные,

Ассемблеры и языки высокого уровня (ЯВУ). Они реализуются, как правило, в системе, которая может быть более или менее сложной. Цель системы заключается в том, чтобы обеспечить быстрое написание и выполнение программ. При этом, затраты времени на подготовку программ прямо пропорциональны их стоимости. Достижения в области автоматического программирования и операционных систем за последние 40 лет позволили увеличить скорость написания программ в десятки-сотни раз. Это дает возможность пользователю не только писать программы быстрее, но и позволяет решать их с меньшими затратами.

Программное обеспечение вычислительной системы предназначено для упрощения процедуры подготовки пользователями задания для ЭВМ и облегчения процесса его прогона и отладки. На машинном уровне (в машинных кодах - машинном языке) ЭВМ выполняет команды, составленные из цифр. Язык Ассемблера (его разработку приписывают Г.Хопперу) представляет собой набор мнемонических обозначений, соответствующих машинному языку. Используется в основном для операций ввода/вывода информации. Он позволяет не только упростить чтение/запись и написание программ, но и создавать дополнительные языковые средства, повышающие качество пользования вычислительной машины. Эти дополнительные средства обеспечиваются операционной системой. Для обычного пользователя не имеет значения, являются ли эти дополнительные команды частью аппаратных средств или они относятся к программному обеспечению. Важно предоставить ему более мощную систему программирования высокого уровня.

Основные аппаратные средства обеспечивают выполнение последовательности команд, находящихся в памяти ЭВМ. Задача пользователя и системы заключается в том, чтобы ввести программу в память и начать ее выполнение с нужной команды. При этом, сами команды ввода должны находиться в ЭВМ.

По сравнению с машинным языком или Ассемблером языки высокого уровня значительно упрощают программирование и позволяют уменьшить количество ошибок в программах. Программа, написанная на ЯВУ для одной машины, с небольшими изменениями или вообще без изменений может быть использована на других. Однако программы на языках высокого уровня выполняются медленнее и занимают больше памяти, чем программы на машинных языках или Ассемблере.

Первым языком высокого уровня был Фортран, разработанный в 1954 году Джоном Бэкусом и примененный в 1956 году. Основное назначение - решение математических и научных задач. Существует очень большое количество ЯВУ. В данном курсе не предусматривается их разбор. Остановимся на их общих особенностях.

Языки высокого уровня являются средством, позволяющим вычислять заданную функцию или решать некоторую задачу. Команды, написанные на ЯВУ, должны быть преобразованы в машинные коды с помощью вычислительной системы, на которой реализована программа. Программа преобразования называется транслятором, а программа, написанная на ЯВУ, исходной программой. Результатом трансляции исходной программы будет программа на

машинном языке - объектная программа. Трансляторы подразделяются на компиляторы и интерпретаторы. Первые транслируют всю программу целиком (требуют больше памяти и более производительны), а вторые - команды исходной программы по одной.

Развитие элементной базы ЭВМ стимулировало разработку более эффективных систем программирования, предназначенных для решения различного рода задач. Общая тенденция создания мощных баз программного обеспечения заключается в поисках возможностей использования разнородных программных продуктов в едином целом.

1. Теоретические основы вычислительной техники
1.1. Представление информации в ЭВМ
1.1.1. Системы счислении

Система счисления (СС) – это способ представления любого числа с помощью принятого набора символов (цифр, букв, …).

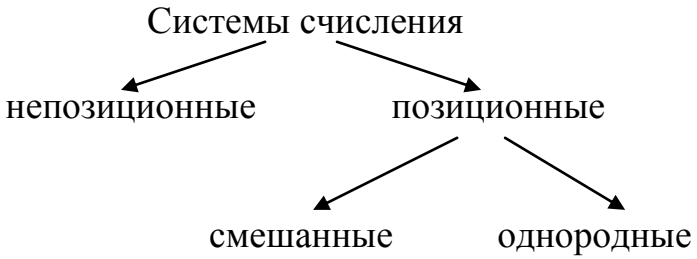
Представление чисел в различных системах счисления (СС)

В различных странах и в различное время исторически сложились и использовались следующие СС: римская; 5 (Африка); 12 (1фут=12дюймов, 12 месяцев…); 20 (ацтеки, майя); 60 (Вавилон – время, углы, …); 2 (Австралия, Полинезия); 10 (Индия → арабы → Европа).

Выбор системы счисления основывается в каждом конкретном случае на ряде критериев. Наиболее распространёнными из них являются:

- эффективность вычислений (обеспечивает простые алгоритмы);
- наглядность (определяется физическими свойствами субъекта);
- реализуемость (в данном случае – техническая).

Системы счисления классифицируются следующим образом:



Различные СС связаны между собой, например: римская и десятичная.

Римская СС	10-я СС
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

Пример. Представить число MDCCCXII в десятичной СС
1000(M) + 500(D) + 100(C) + 100(C) + 100(C) + 10(X) + 1(I) + 1(I) = 1812

Пример. Представить число 1988 в римской СС
1988 = 1000+900+80+8 = 1000+(1000-100)+50+10+10+10+5+1+1+1=
=MCMLXXXVIII

MDCCCXII – 1812

XXX – 30
XL – 40
LXXX – 80
XC – 90
CCC – 300
CD – 400
DC – 600
CM – 900

Позиционная СС (ПСС) определяется конфигурацией и местом (позицией) каждого элемента, а характеризуется: основанием (числом символов n) и алфавитом.

Представление позиционной СС полиномом

$$a_m \cdot n^m + a_{m-1} \cdot n^{m-1} + ... + a_1 \cdot n^1 + a_0 \cdot n^0 + a_{-1} \cdot n^{-1} + ... + a_{-l} \cdot n^{-l} = \sum_{i=-l}^{m} a_i \cdot n^i$$

где n – основание СС, a_i – любой символ из алфавита СС

Примеры ПСС.

СС	основание	алфавит
десятичная	10	0 1 2 3 4 5 6 7 8 9
двоичная	2	0 1
восьмеричная	8	0 1 2 3 4 5 6 7
шестнадцатеричная	16	0 1 2 3 4 5 6 7 8 9 A B C D E F

Однородной СС является система, у которой в каждом разряде одинаковое число цифр. С теоретической точки зрения оптимальной ЭВМ является та, у которой основание системы счисления – е (ближайшее число 3), а с практической используется основание кратное 2 (это определено существующей технологической базой: триггеры – 2 состояния).

Задание 1. Представить число 1539 в римской С. С.

Задание 2. Представить число MMDCXLVII в 10-ной С. С.

Задание 3. Представить число $15.394_{(10)}$ полиномом.

Задание 4. Представить число 11010100,001 полиномом.

1.1.2. Методы перевода чисел из одной позиционной системы счисления в другую.

$A_{(n)}$=>$A_{(k)}$, где A – число, n и k – основания двух С. С.

Алгоритм перевода при n<k:

- представить $A_{(n)}$ полиномом основания n,

- все коэффициенты, основание n-ичной С. С. и все степени основания полинома записать в k-ичной С. С.,

- выполнив все предусмотренные действия получить $A_{(k)}$.

Пример. Перевести число 1011,01 из двоичной С. С. в десятичную С. С.

Здесь n = 2, k = 10, т. е. n<k.

$$1011{,}01_{(2)} = 1\cdot 10^{11} + 0\cdot 10^{10} + 1\cdot 10^{1} + 1\cdot 10^{0} + 0\cdot 10^{-1} + 0\cdot 10^{-1} + 1\cdot 10^{-10} =$$

$$= 1\cdot 2^{3} + 0\cdot 2^{2} + 1\cdot 2^{1} + 1\cdot 2^{0} + 0\cdot 2^{-1} + 1\cdot 2^{-2} = 8 + 2 + 1 + 0{,}25 = 11{,}25_{(10)}$$

Задание 5. Перевести число $10001{,}011_{(2)}$ в десятичную С. С.

Задание 6. Перевести число $0{,}10011_{(2)}$ в десятичную С. С.

Алгоритм перевода при n>k. Преобразуются отдельно и по-разному целая $A_{ц(n)}$ и дробная $A_{д(n)}$ части числа $A_{(n)}$.

$A_{(n)} = A_{ц(n)}+A_{д(n)}$.

Преобразование $A_{ц(n)}$=>$A_{ц(k)}$ осуществляется последовательным делением $A_{ц(n)}$ на k, выделением остатков от деления и составлением из них числа $A_{ц(k)}$.

Пример. Перевести число 37,75 из десятичной С. С. в двоичную С. С. Здесь n=10, k=2, т. е. n>k.

$A_{(10)}=A_{ц(10)}+A_{д(10)}=37_{(10)}+0{,}75_{(10)}$. Нужно $A_{ц(10)}$=>$A_{ц(2)}$ и $A_{д(10)}$=>$A_{д(2)}$.

37	|2				
36	18	|2			
1	18	9	|2		
a_0	0	8	4	|2	
	a_1	1	4	2	|2
		a_2	0	2	1
			a_3	0	a_5
				a_4	

$$\Rightarrow A_{ц(2)} = a_5a_4a_3a_2a_1a_0 = 100101_{(2)}$$

x0,	75
	2
a_{-1}=1,	50
	2
a_{-2}=1,	00
	2
a_{-3}=0	00

=> $A_{д(2)}=0,a_{-1}a_{-2}a_{-3}\ldots=0{,}11_{(2)}$

В итоге: $37{,}75_{(10)}=100101{,}11_{(2)}$

Проверка: $1\cdot 2^{5} + 0\cdot 2^{4} + 0\cdot 2^{3} + 1\cdot 2^{2} + 0\cdot 2^{1} + 1\cdot 2^{0} + 1\cdot 2^{-1} + 1\cdot 2^{-2} = 37{,}75_{(10)}$

Задание 7. Перевести число $29{,}8125_{(10)}$ в двоичную С. С.

Задание 8. Перевести число $100{,}3_{(10)}$ в двоичную С. С.

Задание 9. Перевести число $364_{(7)}$ в пятеричную С. С. через десятичную С. С.

Перевод чисел в С. С. с основанием кратным двум.

Пусть $n=2^l$, а k=2. Тогда каждая цифра n-ичной С. С. представляется l – разрядным двоичным числом.

Пример. Перевести число $472{,}54_{(8)}$ в двоичную и шестнадцатеричную С. С.

Сначала выполним перевод в двоичную С. С.

Так как $16=2^4$, то l_2 и, следовательно,

$472{,}54_{(8)}$= 1 0011 1010 , 1011 = 1 3 A , $B_{(16)}$.

Задание 10. Перевести число $3103{,}12_{(4)}$ в двоичную С. С.

Задание 11. Перевести число $1011010101{,}11_{(2)}$ в восьмеричную С. С. и шестнадцатеричную С. С.

Задание 12. Перевести число $F0A_{(16)}$ в четвертичную С. С.

Задание 13. Перевести число $27{,}59375_{(10)}$ в шестнадцатеричную С. С.

1.1.3. Формы представления чисел.

Формы представления чисел
в вычислительной технике

с фиксированной запятой — с естественной запятой — с плавающей запятой

Представление чисел в форме с фиксированной запятой (ФЗ)

Формат представления чисел с запятой, фиксированной перед старшим разрядом:

,	знак	2^{-1}	2^{-2}	2^{-3}		$2^{-(n-2)}$	$2^{-(n-1)}$
	0	1	2	3		-2	-1

<= веса разрядов

<= номера разрядов

↑ n – разрядная сетка ЭВМ

В этом формате диапазон представляемых чисел $2^{-(n-1)} \le |A| \le 1 - 2^{-(n-1)}$

Формат представления чисел с запятой, фиксированной после младшего разряда:

знак	2^{n-2}	2^{n-3}	2^{n-4}		2^1	2^0
n-1	n-2	n-3	n-4		1	0

В этом формате диапазон представляемых чисел $1 \le |A| \le 2^{n-1} - 1$.

Представление знаков: “+” <==> 0; “-“ <==> 1.

Пример. Представить число $-0{,}0225_{(10)}$ в форме с ФЗ в 16-разрядной сетке ЭВМ.

$-0{,}0225_{(10)} = -0{,}0000010111_{(2)}$.

,	1	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0

Формат представления чисел в форме с естественной запятой.

(с произвольным расположением запятой в пределах n-разрядной сетки ЭВМ)

знак	2^{l-1}	2^{l-2}		2^1	2^0	,	2^{-1}		$2^{-(m-1)}$	2^{-m}
	целая часть числа (l разрядов)						дробная часть числа (m разрядов)			
n – разрядная сетка ЭВМ (n=l+m+1)										

Задание 14. Представить число $1141_{(10)}$ в форме с ФЗ в 16-разрядной сетке.

Задание 15. Представить число $-0{,}4_{(10)}$ в форме с ФЗ в 8-разрядной сетке.

Задание 16. Представить число $-88{,}8125_{(10)}$ в форме с естественной запятой в 16-разрядной сетке.

Представление чисел в форме с плавающей запятой (ПЗ).

Число представляется в виде: $A = \pm S^{\pm P} \cdot M$, где А – исходное число, М – мантисса (/M/<1), S^P – характеристика числа А, S – основание С. С., Р – порядок.

знак порядка					знак числа		
	разряды Р				разряды М		
l разрядов				m разрядов			

При таком представлении чисел необходимо проведение их нормализации. Число А называется нормализированным, если мантисса удовлетворяет условию $1/S \le |M| \le 1 - 2^{-(m-1)}$, т. е. если старший разряд мантиссы отличен от нуля.

Пример. Представить число $-27{,}375_{(10)}$ в форме с ПЗ в 16-разрядной сетке ЭВМ.

$-27{,}375_{(10)} = -11011{,}011_{(2)}$. Проведя нормализацию получим $-0{,}11011011 \cdot 10^{+101}$. Пусть под порядок отведено 4 разряда. Тогда содержимое разрядной сетки будет:

0	0	1	0	1	1	1	1	0	1	1	0	1	1	0	0

Задание 17. Представить число $59{,}625_{(10)}$ в форме с ПЗ в 16-разрядной сетке.

Задание 18. Представить число $-0{,}0625_{(10)}$ в форме с ПЗ в 8-разрядной сетке.

Задание 19. Представить число $-6{,}025_{(10)}$ в форме с ПЗ в 32-разрядной сетке.

Задание 20. Представить число $0{,}1E_{(16)}$ в форме с ПЗ в 16-разрядной сетке.

Представление чисел в форме с ПЗ и смещённым порядком.

Смещённый порядок $P_{см} = (\pm P) + N$. Здесь $\pm P$ - истинный порядок, а $N=2^k$, где число k – число двоичных разрядов отведённое для представления $\pm P$ (или $P_{см}$). Всегда $P_{см}>0$.

Короткий (32 разряда) формат представления чисел с ПЗ и смещённым порядком в ЕС ЭВМ

знак числа	$P_{см}$	мантисса					
0 β_1 1 3	4 β_2 7	8 α_1 11	12 α_2 15	16 α_3 19	20 α_4 23	24 α_5 27	28 α_6 31

Здесь $\beta_1, \beta_2, \alpha_1 \div \alpha_6$ - шестнадцатеричные разряды. Следовательно содержимое разрядной сетки может быть представлено числом $\beta_1\beta_2\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_6$. При таком представлении нормализация проводится для шестнадцатеричного числа. При определении смещённого порядка число N в шестнадцатеричной С. С. будет равно $40_{(16)}$.

В ЕС ЭВМ существуют также длинный (64 двоичных разряда) и расширенный (128 двоичных разряда) форматы. В них расширяется только поле отводящееся под мантиссу.

Пример. Представить число $-0{,}01_{(10)}$ в форме с ПЗ и смещённым порядком в коротком формате ЕС ЭВМ.

Переведём число в шестнадцатеричную С. С. $-0{,}01_{(10)} \approx -0{,}02815C2_{(16)}$. Проведя нормализацию этого числа получим мантиссу $(-0{,}2815C3)_{(16)}$, а порядок $(-1)_{(16)}$. Определим смещённый порядок.

$$P_{см} = N + (\pm P) = 40_{(16)} + (-1)_{(16)} = 1000000_{(2)} - 1_{(2)} = 0111111_{(2)}.$$

Расположим число в 32-х разрядной сетке

знак числа	P		M					
1	0 1 1	1 1 1 1	0 0 1 0	1 0 0 0	1 1 1 1	0 1 0 1	1 1 0 0	0 0 1 1
B		F	2	8	F	5	C	3

Следовательно в памяти ЕС ЭВМ число $-0{,}01_{(10)}$ будет представлено шестнадцатеричным словом BF28F5C3.

Задание 21. Представить число $-43{,}375_{(10)}$ в форме с ПЗ и смещённым порядком в коротком формате ЕС ЭВМ. Содержимое разрядной сетки представить 16-ричным словом.

Задание 22. Представить число $17{,}1_{(10)}$ в форме с ПЗ и смещённым порядком в длинном формате ЕС ЭВМ.

Задание 23. Представить число $-0{,}0625_{(10)}$ в форме с ПЗ и смещённым порядком в коротком формате ЕС ЭВМ. Содержимое разрядной сетки представить 16-ричным словом.

Задание 24. Содержимое 32-х разрядной сетки ЕС ЭВМ представлено словом $C3180000_{(16)}$. Определить какое десятичное число записано в разрядной сетке.

1.1.4. Представление алфавитно-цифровой информации.

Используются различные системы соответствия символов (цифр, букв, математических и служебных знаков) их двоичному представлению. Это ДКОИ – двоичный код для обработки информации (8 разрядное представление символа), КОИ-8 – код обмена информацией (8 разрядов), КОИ-7 – код обмена информацией (7 разрядов).

байт	байт		байт
символ	символ		символ

алфавитно-цифровая информация

Соответствующие таблицы кодирования символов даны в [1].

Пример. Представить запись «СМ-4» в ДКОИ.

байт	байт	байт	байт
11000011	11010100	01100000	11110100
C	M	-	4

Система представления данных состоящих только из десятичных цифр использует двоично-десятичное кодирование. Соответствие десятичных цифр их 2-10-м кодам дано в таблице 1. Не используемые в таблице 1 комбинации (1010 – 1111) применяются для представления значков чисел и зоны в соответствии с табл. 2.

Таблица 1

десятичная цифра	2-10 код
0	0000
1	0001
2	0010
…	…
9	1001

Таблица 2

	ДКОИ	КОИ-8
+	1100	1010
-	1101	1011
зона	1111	1100

Существуют два специальных формата для многоразрядных десятичных чисел:

зонный формат

зона	цифра	зона	цифра		знак	цифра
байт		байт			байт	

упакованный формат

цифра	цифра		цифра	знак
байт			байт	

В упакованном формате число байтов должно быть целым.

Пример. Представить десятичное число -5901 в зонном и упакованном форматах ДКОИ.

1111 зона	0101 5	1111 зона	1001 9	1111 зона	0000 0	1101 знак	0001 1

число в зонном формате

0000 (*)	0101 5	1001 9	0000 0	0001 1	1101 знак

число в упакованном формате
(*) – дополнение до целого числа байтов

Задание 25. Представить запись «A/B-I» в ДКОИ и КОИ-8 используя таблицы в [1].

Задание 26. Представить последовательность десятичных чисел -17+0-10 в зонном формате ДКОИ.

Задание 27. Представить последовательность десятичных чисел +10-187-4 в упакованном формате КОИ-8.

Арифметические основы вычислительной техники.

Так как двоичная С. С. и десятичная С. С. принадлежат к одному классу позиционных аддитивных С. С. с естественным порядком весов, то правила выполнения арифметических операций над числами в этих С. С. одинаковы.

Пример. Выполнить четыре арифметических операции над числами $A=1001_{(2)}$ и $B=11_{(2)}$.

```
 +1001      -1001      x1001      -1001 | 11
    11         11         11         11 | 11
  ------     ------     ------     ------
  1100       0110       1001       -011
                        1001          11
                       ------     ------
                       11011         00
```

1.2. Арифметические основы вычислительной техники.

1.2.1. Прямой, обратный и дополнительный коды чисел.

Коды целых и дробных чисел, представленных в форме с ФЗ, формируются одинаково. Вопрос о том, какое число закодировано – целое или дробное, решается по формату с которым работает ЭВМ.

Прямой код представляет собой запись самого числа с соответствующим знаком. Знак «+» кодируется 0, а знак «–» кодируется 1 и записывается крайним слева (для наглядности будем выделять знаковый разряд рамкой).

Обратный код положительного числа совпадает с его прямым кодом. Обратный код отрицательного числа формируется из прямого путём инвертирования цифр во всех разрядах за исключением знакового.

Дополнительный код положительного числа совпадает с его прямым кодом. Дополнительный код отрицательного числа формируется добавлением единицы к младшему разряду обратного кода этого числа.

Задание 28. Представить число $-0{,}111_{(2)}$ в прямом, обратном и дополнительных кодах.

Задание 29. Представить число $-10000_{(2)}$ в прямом, обратном и дополнительных кодак.

1.2.2. Сложение и вычитание двоичных чисел.

Сложение и вычитание чисел представленных в форме с ФЗ.

Сложение и вычитание чисел А и В в форме с ФЗ сводится в ЭВМ к суммированию обратных или дополнительных кодов этих чисел с учётом знаков.

При суммировании дополнительных кодов чисел А и В сумма S получается в дополнительном коде: $A_{доп}+B_{доп}=S_{доп}$.

При суммировании обратных кодов чисел А и В сумма S получается в обратном коде: $A_{обр}+B_{обр}=S_{обр}$

всегда, за исключением случая, когда B<0. В этом случае, если есть перенос из знакового разряда, нужна коррекция результата суммирования S’ путём добавления 1 к его младшему разряду, т. е. при B<0: $A_{обр}+B_{обр}=S'+1=S_{обр}$.

Для этого используется перенос из знакового разряда.

Пример. Выполнить суммирование чисел $A=\pm 9_{(10)}=\pm 1001_{(2)}$ и $B=\pm 3_{(10)}=\pm 0011_{(2)}$ в обратном и дополнительном кодах при всех возможных комбинациях знаков.

Представим результаты вычислений таблицей и в каждой графе проведём проверку результата переводом его в прямой код $S_{пр}$.

A=9 $(A\geq 0)$ B=3 $(B\geq 0)$	A=-9 $(A<0)$ B=3 $(B\geq 0)$	A=9 $(A\geq 0)$ B=-3 (B<3)	A=-9 (A<0) B=-3 (B<0)
$+A_{обр}=$ 0 1001 $B_{обр}=$ 0 0011	$+A_{обр}=$ 1 0110 $B_{обр}=$ 0 0011	$+A_{обр}=$ 0 0110 $B_{обр}=$ 1 0011	$+A_{обр}=$ 1 0110 $B_{обр}=$ 1 0011
$S_{обр}=$ 0 1100	$S_{обр}=$ 1 1001	S’= 0 0101 +1	S’= 1 0010 +1
		$S_{обр}=$ 0 0110	$S_{обр}=$ 1 0011
$S_{пр}=S_{обр}=+12_{(10)}$	$S_{пр}=$ 1 0110$=-6_{(10)}$	$S_{пр}=S_{обр}=+6_{(10)}$	$S_{пр}=$ 1 1100=-12
$+A_{доп}=$ 0 1001 $B_{доп}=$ 0 0011	$+A_{доп}=$ 1 0111 $B_{доп}=$ 0 0011	$+A_{доп}=$ 0 1001 $B_{доп}=$ 1 1101	$+A_{доп}=$ 1 0111 $B_{доп}=$ 1 1101
$S_{доп}=$ 0 1100	$S_{доп}=$ 1 1010	$S_{доп}=$ 0 0110	$S_{доп}=$ 1 0100
$S_{пр}=S_{доп}=+12_{(10)}$	$S_{пр}=$ 1 0110$=-6_{(10)}$	$S_{пр}=S_{доп}=+6_{(10)}$	$S_{пр}=$ 1 1100=-12

Задание 30. Выполнить суммирование чисел -7 и 5 в обратном и дополнительном кодах.

Задание 31. Выполнить суммирование чисел 14 и -10 в обратном и дополнительных кодах.

Задание 32. Выполнить суммирование чисел -8 и -6 в обратном и дополнительных кодах.

При выполнении операции суммирования кодов чисел, представленных в форме с ФЗ, возможно переполнение разрядной сетки ЭВМ. Существуют следующие признаки переполнения:

- одинаковые знаковые разряды слагаемых отличаются от знака результата;
- в процессе суммирования значение переносов из старшего и из знакового разрядов не совпадают.

Пример. Определить факт переполнения разрядной сетки ЭВМ при суммировании чисел $A = \pm 9_{(10)} = \pm 1001_{(2)}$ и $B = \pm 10_{(10)} = \pm 1010_{(2)}$, представленных в обратном и дополнительном кодах при всех возможных комбинациях знаков.

Переполнение возможно лишь при равенстве знаков суммируемых чисел поэтому рассмотрим лишь пары чисел A>0, B>0 и A<0, B<0.

A=+9 (A>0) B=+10 (B>0)	A=-9 (A<0) B=-10 (B<0)	A=-9 (A<0) B=-10 (B<0)
$_{+}A_{обр}=A_{доп}=$ 0 1001 $B_{обр}=B_{доп}=$ 0 1010	$_{+}A_{обр}=$ 1 0110 $B_{обр}=$ 1 0101	$_{+}A_{доп}=$ 1 0111 $B_{доп}=$ 1 0110
S= 1 0011	S'= 0 1011 +1 $S_{обр}=$ 0 1100	$S_{доп}=$ 0 1101

В обоих случаях произошло переполнение выявляемое по любому из приведённых выше признаков. Например, знак результата в каждом случае отличается от знаковых разрядов слагаемых, а значения переносов не совпадают.

Задание 33. Определить, возникнет ли переполнение разрядной сетки при суммировании чисел $7_{(10)}$ и $11_{(10)}$.

Задание 34. По какому признаку можно определить факт переполнения разрядной сетки при суммировании чисел $-13_{(10)}$ и $-8_{(10)}$.

Сложение и вычитание чисел, представленных в форме с ПЗ.

Выполнение операций сложения и вычитания двоичных чисел $A = M_A \cdot 2^{P_A}$ и $B = M_B \cdot 2^{P_B}$ в форме с ПЗ аналогично соответствующим операциям над числами в форме с ФЗ, если выполняется условие $P_A=P_B$. Если $P_A \neq P_B$, то необходимо:

- провести предварительное выравнивание порядков, что приводит к денормализации одной из мантисс;
- провести нормализацию результата операции, если мантисса результата $M_S \geq 1$ или $M_S<0{,}5$. При $M_S \geq 1$ нужна нормализация вправо, а при $M_S<0{,}5$ – нормализация влево.

Признаком необходимости нормализации вправо является переполнение поля разрядной сетки, отводимое под M_S. Это возможно лишь при суммировании чисел с одинаковыми знаками.

Признаком необходимости нормализации влево является совпадение цифр в знаковом разряде и старшем разряде мантиссы. Это возможно лишь при суммировании чисел с разными знаками, т. е. при представлении мантисс в обратном или дополнительных кодах.

Пример. Выполнить суммирование двоичных чисел $A = +0{,}1011 \cdot 10^{+100} = 11_{(10)}$ и $B = +0{,}11 \cdot 10^{+10} = 3_{(10)}$.

В данном случае $P_A \neq P_B$, следовательно проведём предварительное выравнивание порядков.

$A = +0{,}1011 \cdot 10^{+100}$, $B = +0{,}0011 \cdot 10^{+100}$.

Так как A>0 и B>0, то выполним суммирование в прямом коде (мантиссы суммируются в прямом коде)

	порядок	знак	мантисса
$_+$A	100	0	1011
B	100	0	0011
S	100	0	1110

Так как $0{,}5<M_S<1$, то нормализация результата не требуется и получим $S = +0{,}1110 \cdot 10^{+100} = 14_{(16)}$.

Пример. Выполнить суммирование двоичных чисел $A = +0{,}111 \cdot 10^{+100} = 14_{(10)}$ и $B = \pm 0{,}1011 \cdot 10^{+100} = 11_{(10)}$ с представлением мантиссы шестью разрядами.

Так как $P_A=P_B$, то выравнивание порядков не требуется. Для случая B>0 нахождение суммы S проведём используя прямой код.

	порядок	знак	мантисса
$_+$A	100	0	111000
B	100	0	101100
	100	1	100100

Так как возникло переполнение поля разрядной сетки, отведённого под мантиссу, проведём нормализацию результата вправо на один разряд, соответственно увеличив значение порядка на единицу, т. е.

100 1 → 100100
101 0 → 110010

В итоге получим результат $S = +0{,}11001 \cdot 10^{+101} = 25_{(10)}$.

Для случая B<0 нахождение суммы S проведём используя дополнительный код (мантиссы суммируются в дополнительном коде).

	порядок	знак	мантисса в доп. коде
$_+$A	100	0	111000
B	100	1	010100
S	100	0	001100

Так как возникло совпадение цифр в знаковом разряде и старшем разряде мантиссы, то проведём нормализацию результата влево на два разряда, соответственно уменьшив значение порядка на два, т. е.

S 100 0 ← 001100
S 010 0 110000

В итоге получим результат $S = +0{,}11 \cdot 10^{+10} = 3_{(10)}$.

Задание 35. Выполнить суммирование двоичных чисел $+0{,}1011\cdot 10^{+11} = +5{,}5_{(10)}$ и $+0{,}11\cdot 10^{+10} = +3_{(10)}$.

Задание 36. Выполнить суммирование двоичных чисел +1101,11 и -1000,01 в форме с ПЗ.

Задание 37. Выполнить суммирование двоичных чисел $-0{,}111\cdot 10^{+11}$ и $-0{,}1001\cdot 10^{+100}$.

1.2.3. Сложение и вычитание чисел в двоично-десятичных кодах (Д-кодах).

0	0000
1	0001
2	0010
…	….
9	1001

Соответствие десятичных цифр их двоично-десятичным кодам дано таблицей.

Такое кодирование применяется при записи десятичных чисел в зонном и упакованном формате.

Сложение чисел в Д-кодах сводится к суммированию тетрад, соответствующих их разрядам, по правилам двоичной арифметики и коррекции тетрад суммы. Для тетрад суммы, значение которых превышают 1001, необходимо организовать перенос в следующую тетраду и выполнить коррекцию. Коррекция представляет собой вычитание 1010, что соответствует сложению тетрады с 0110 без переноса (т. к. 0110 есть дополнительный код числа 1010).

Пример. Определить сумму чисел $A=753_{(10)}$ и $B=439_{(10)}$ в двоично-десятичном коде.

Представим числа A и B в Д-коде и выполним суммирование:

$_{+}$A =	0111	0101	0011
B =	0100	0011	1001
S' =	1011	1000	1100
выполним коррекцию	+0110		+0110
S = 1	0001	1001	0010

,что соответствует $1192_{(10)}$.

Вычисление чисел в Д-кодах выполняется так же, как и вычисление двоичных чисел, т. е. вычисляемое представляется в дополнительном коде, а операция вычисления сводится к сложению. Дополнительный код числа, представленного Д-кодом, формируется дополнением каждой тетрады до 1001 и добавлением к младшей тетраде единицы.

Пример. Определить разность чисел $A=753_{(10)}$ и $B=439_{(10)}$ в двоично-десятичном коде.

Представим число B в дополнительном коде

-1001	-1001	-1001	дополнение каждой тетрады до 1001
0100	0011	1001	
0101	0111	0000	добавление к младшей тетраде единицы
		+1	
$B_{доп}=$ 0101	0110	0001	

Теперь выполним операцию $S=A-B=A+B_{доп}$.

$+A$	0111	0101	0011
$B_{доп}$	0101	0110	0001
S’	1100	1011	0100
коррекция	+0110	+0110	
S	0011	0001	0100

, что соответствует $314_{(10)}$.

Задание 38. Выполнить сложение чисел $280_{(10)}$ и $157_{(10)}$ в Д-коде.

Задание 39. Определить разность чисел $1092_{(10)}$ и $956_{(10)}$ в Д-коде.

1.2.4. Умножение и деление двоичных чисел.

Ограничимся рассмотрением выполнения операций умножения и деления над двоичными числами в прямом коде с учётом их знаков. Эти операции выполняются с многократным применением операций сложения и сдвига и могут быть реализованы схемой показанной на рисунке.

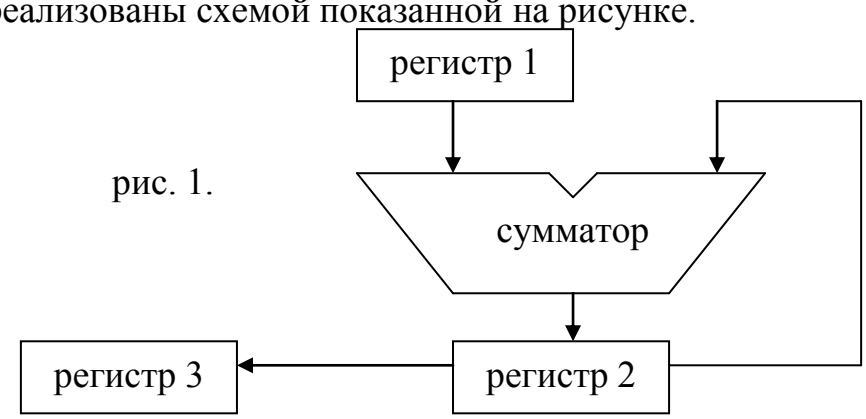


рис. 1.

Операция умножения чисел A и B выполняется в несколько тактов. Множимое A записывается в Рег.1, а множитель B в Рег.3. Рег.2 предварительно устанавливается в 0. На каждом такте анализируется содержимое старшего (левого) разряда Рег.3. Если в нём находится 1-ца, то производится общий сдвиг влево содержимого Рег.3 и Рег.2 (с переносом между ними) и суммирование содержимого регистров Рег.1 и Рег.2 с занесением результата в Рег.2. Если в старшем разряде Рег.3 находится 0, то производится лишь сдвиг влево содержимого Рег.3 и Рег.2. Число тактов равно числу разрядов множителя B. В результате произведение будет располагаться в Рег.3 и Рег.2.

Код знака результата операции умножения получаем суммированием по mod2 кодов знаков сомножителей (логическая функция сложения по mod2 показана далее).

Пример. Выполнить умножение чисел $A=11_{(10)}=1011_{(2)}$ и $B=13_{(10)}=1101_{(2)}$.

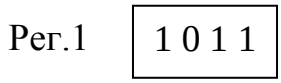
Для наглядности последующих действий предварительно умножим числа «в столбиках».

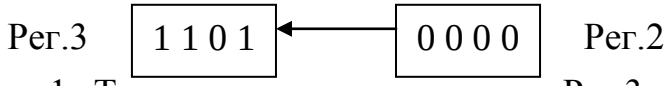
Теперь рассмотрим выполнение операции умножения схемой, представленной на рис. 1.

Операция проводится в 4 такта (число B четырёхразрядное).

$$
\begin{array}{r}
{}_{x}1011 \\
1101 \\
\hline
1011 \\
1011 \\
1011 \\
\hline
10001111
\end{array}
$$

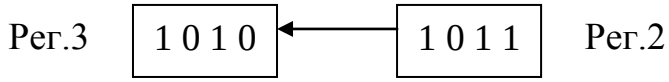
Начальное содержимое регистров:



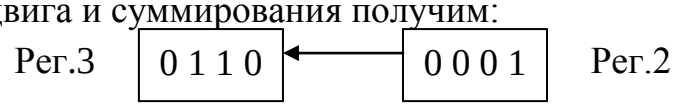


Такт 1. Так как в старшем разряде Рег.3 находится 1-ца, то производится сдвиг содержимого Рег.3 и Рег.2 и суммирование содержимого Рег.1 и Рег.2 с записью результата в Рег.2.

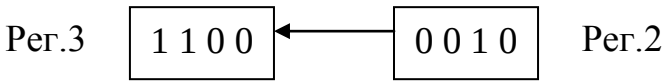
Получим:



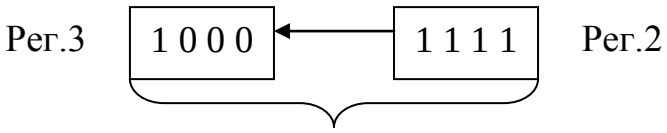
Такт 2. В старшем разряде Рег.3 опять 1-ца, следовательно, в результате сдвига и суммирования получим:



Такт 3. В старшем разряде Рег.3 находится 0, поэтому производится только сдвиг.



Такт 4. В старшем разряде Рег.3 опять 1-ца. Выполняются сдвиг и суммирование.



Проверим правильность выполнения умножения.

$10001111_{(2)}=143_{(10)}$, т. е. $\boldsymbol{13_{(10)} \cdot 11_{(10)} = 143_{(10)}}$.

Задание 40. Выполнить операцию умножения чисел $14_{(10)}$ и $-9_{(10)}$ реализуя алгоритм работы схемы рис. 1.

Задание 41. Выполнить операцию умножения чисел $-19_{(10)}$ и $-7_{(10)}$, реализуя алгоритм работы схемы рис. 1.

Операция деления чисел А и В выполняется схемой рис. 1 также за несколько тактов. Делимое А располагается одновременно в Рег.3 и Рег.2, а делитель В в Рег.1. На каждом такте сравнивается содержимое Рег.3 и Рег.1. Если число в Рег.3 больше числа в Рег.1, то производится:

- вычитание из содержимого Рег.3 содержимого Рег.1 с записью результата в Рег.3;
- общий сдвиг влево содержимого Рег.3 и Рег.2 (с переносом между ними);
- запись 1-цы в младший разряд Рег.2.

Если число в Рег.3 меньше числа в Рег.1, то производится лишь сдвиг влево содержимого Рег.3 и Рег.2. Итоговый результат операции деления будет располагаться в Рег.2.

Код знака результата операции деления получается аналогично коду знака результата операции умножения.

Пример. Выполнить деление числа $143_{(10)}=10001111_{(2)}$ на число $11_{(10)}=1011_{(2)}$.

Для наглядности последующих действий предварительно разделим числа «в столбик».

```
- 1 0 0 0 1 1 1 1 | 1 0 1 1
    1 0 1 1       | 1 1 0 1
  - 0 1 1 0 1
      1 0 1 1
    - 0 1 0 1 1
        1 0 1 1
```

Теперь рассмотрим выполнение операции деления схемой, представленной на рис.1. Начальное содержимое регистров:

Рег.1 | 1 0 1 1 |

Рег.3 | 0 0 1 0 0 0 | ← | 1 1 1 1 | Рег.2

Такт 1. Так как число в Рег.3 меньше числа в Рег.1 то производится общий сдвиг влево содержимого Рег.3 и Рег.2. Получим

Рег.3 | 0 1 0 0 0 1 | ← | 1 1 1 0 | Рег.2

Такт 2. Теперь число в Рег.3 больше числа в Рег.1, поэтому производится вычитание из содержимого Рег.3 содержимого Рег.1 с записью результата в Рег.3.

Рег.3 | 0 0 0 1 1 0 | ← | 1 1 1 0 | Рег.2

а затем выполняется общий сдвиг всего содержимого Рег.3 и Рег.2 и запись 1-цы в младший разряд Рег.2 Получим

Рег.3 | 0 0 1 1 0 1 | ← | 1 1 0 1 | Рег.2

Такт 3. Так как сейчас число в Рег.3 больше числа в Рег.1, то действия предыдущего такта повторяются. После вычитания получим

Рег.3 | 0 0 0 0 1 0 | ← | 1 1 0 1 | Рег.2

а после сдвига влево в Рег.3 и Рег.2 и записи 1-цы в младший разряд Рег.2 будем иметь

Рег.3 | 0 0 0 1 0 1 | ← | 1 0 1 1 | Рег.2

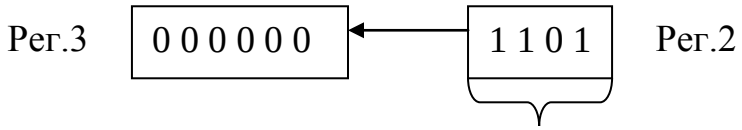
Такт 4. В этом случае число в Рег.3 стало меньше числа в Рег.1, поэтому производится только общий сдвиг влево Рег.3 и Рег.2

Рег.3 | 0 0 1 0 1 1 | ← | 0 1 1 0 | Рег.2

Такт 5. Теперь опять число в Рег.3 больше числа в Рег.1. Поэтому выполняется вычитание

Рег.3 | 0 0 0 0 0 0 | ← | 0 1 1 0 | Рег.2

сдвиг и запись 1-цы в младший разряд Рег.2



результат

Проверим правильность выполнения деления:

$1101_{(2)}=13_{(10)}$, т. е. $143_{(10)}:11_{(10)}=13_{(10)}$.

Замечание. Чтобы реализовать операцию вычитания в сумматоре (рис.1) нужно заменить её операцией сложения с числом в дополнительном коде, т. е. к содержимому Рег.3 прибавлять содержимое Рег.1 взятое в дополнительном коде. В рассмотренном примере вычиталось число 1011, следовательно нужно прибавлять число 0101. (Проверить самостоятельно).

Задание 42. Выполнить операцию деления чисел $126_{(10)}$ и $14_{(10)}$, реализуя алгоритм работы схемы рис.1.

Задание 43. Выполнить операцию деления чисел $123_{(10)}$ и $-7_{(10)}$, реализуя алгоритм работы схемы рис.1.

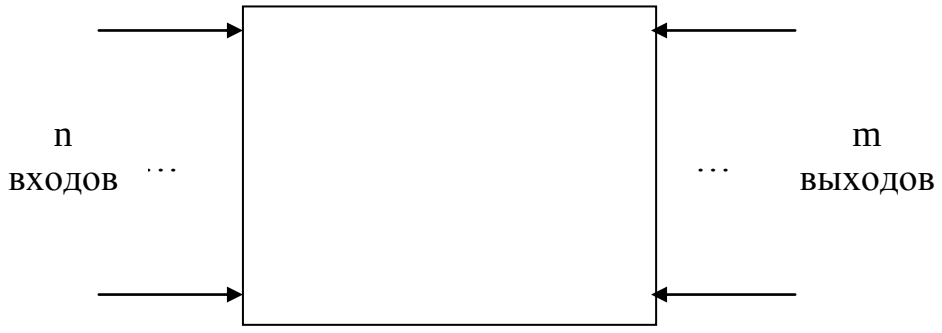
Замечание. При выполнении операции умножения и деления чисел в форме с ПЗ необходимо выполнить:

- умножение или деление мантисс исходных чисел (по алгоритмам рассмотренным выше);
- определение порядка результата (получается соответственно сложением или вычитанием порядков исходных чисел с учётом их знаков);
- нормализацию результата с соответствующей коррекцией порядка.

1.3. Логические основы вычислительной техники.

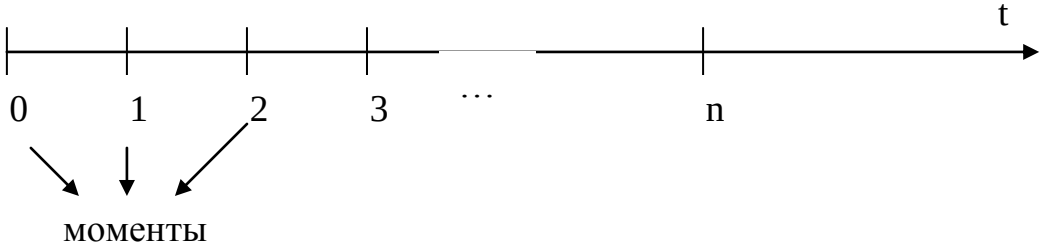
Комбинационные схемы и конечные автоматы.

Устройство преобразовывающее информацию



Входной и выходной сигналы.

В цифровой вычислительной машине (ЦВМ) время дискретно, т. е. переход от 0 в 1 за момент. Интервал между моментами – такт.



Преобразующее устройство P:

$X(t) \Rightarrow P \Rightarrow Y(t)$, где X(t)={$x_1$, x_2 … x_n} – множество входных дискретных тактиров-х сигналов, Y(t)={y_1, y_2 … y_m} – множество выходных сигналов, P – функция или функционал преобразования.

Если элементарный сигнал может быть представлен с различными состояниями:

$x_i \in \{0,1,2,...k-1\}, i = \overline{1,n}$ $y_i \in \{0,1,2,...k-1\}, i = \overline{1,m}$ то можно ввести понятие k-значной логики.

k-значной логикой называют математический аппарат, позволяющий описать функционирование P-преобразователей. k-значная логика может приведена быть к 2-ой логике, т. к. k-значную переменную можно заменить на $\log_2 k$ двоичных переменных.

Поэтому дальше 2-ая (Булева) логика.

Устройство реализующее P называется в ЦВМ автоматом.

2 вида дискретных автоматов:

1) Если совокупность выходных сигналов (выходного слова) Y(t) зависит от X(t) и не зависит от внутреннего состояния автомата, то это комбинационная схема. Y(t)=P[X(t)].

Структурная схема: $X(t) \Rightarrow P \Rightarrow Y(t)$

ЭП комбинационный автомат не содержит.

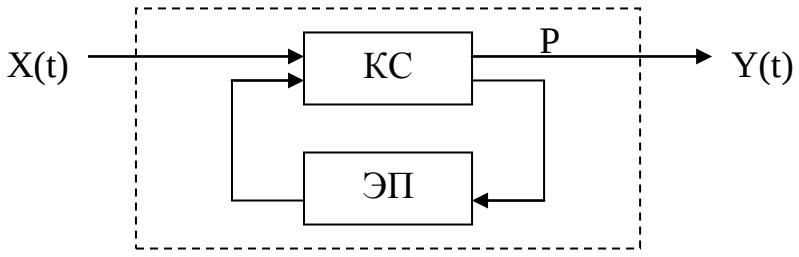
Закон функционирования комбинационного автомата определяется заданием соответствия между входным и выходным словами.

Можно задать: таблицей

аналитически (булевыми функциями)

2) Если Y(t)=P[X(t), S(t-1)], то автомат с памятью (конечный).

r+1 внутренних состояний S(t)={S_0, S_1, … S_r}. При подаче X(t) в t-ом такте автомат переходит их S(t-1) в S(t). Структурная схема



КС – комбинационная схема.

Одновременность сигналов на комбинационной схеме обеспечивается синхронизацией (тактируемостью сигнала).

Элементы теории конечных автоматов в главе 3.

1.3.1. Переключательные (булевы, логические) функции.

I. Основные понятия и определения.

Булевой функцией называется $f(x_1, x_2, \dots x_n)$, аргументы которой $x_i (i = \overline{1,n})$ и сами функции принимают значение из множества $E^2=\{0,1\}$. Они могут быть заданы таблицей. Полная таблица функции от n переменных имеет 2^n строк и n+1 столбцов. Упорядоченная таблица называется таблицей истинности. Другая форма – булевы выражения.

x_1	x_2	…	x_n	$f(x_1 \dots x_n)$
0	0	…	0	$f(0,0,\dots 0) \in \{0,1\}$
1	0	…	0	
…	…	…	…	…
1	1	…	1	$f(1,1,\dots 1) \in \{0,1\}$

Теорема: число различных переключательных функций n-аргументов равно 2^{2^n}.

Определение: Две переключательных функции называются равными, если на всех наборах аргументов они принимают одинаковые значения.

Определение: Переключательная функция f называется существенно зависимой от x_i, если выполняется неравенство: $f(x_1 \dots x_{i-1} 0 x_{i+1} \dots X_n) \neq f(x_1 \dots x_{i-1} 1 x_{i+1} \dots X_n)$. Не все переключательные f существенно зависимы.

Определение: Переключательная функция f независящая ни от одного аргумента и равная 0 (1) на всех наборах называется нулевой (единичной).

II. Элементарные логические функции.

Называются функции 1 или 2-х аргументов.

Логическая функция 1-ой переменной:

$f_1(x) \equiv 0$ - нулевая функция;

$f_2(x) = x$ - функция повторения аргумента

$f_3(x) = \overline{x}$ - функция инверсии аргумента

$f_4(x) \equiv 1$ - единичная функция

x	0	1
$f_1(x)$	0	0
$f_2(x)$	0	1
$f_3(x)$	1	0
$f_4(x)$	1	1

Логические функции 2-х переменных:

функции	аргументы x_1 0011 x_2 0101	обозначение функции	название функции и комментарии
$f_0(x_1,x_2)$	0 0 0 0	0	Нулевая
$f_1(x_1,x_2)$	0 0 0 1	$x_1 \wedge x_2$	Конъюнкция
$f_2(x_1,x_2)$	0 0 1 0	$x_1 \Delta x_2$	запрет по x_2 $(x_1 \overline{x}_2)$
$f_3(x_1,x_2)$	0 0 1 1	x_1	повторение x_1
$f_4(x_1,x_2)$	0 1 0 0	$x_2 \Delta x_1$	запрет по x_1
$f_5(x_1,x_2)$	0 1 0 1	x_2	повторение x_2
$f_6(x_1,x_2)$	0 1 1 0	$x_1 \oplus x_2$	сумма по mod2
$f_7(x_1,x_2)$	0 1 1 1	$x_1 \vee x_2$	Дизъюнкция
$f_8(x_1,x_2)$	1 0 0 0	$x_1 \downarrow x_2$	стрелка Пирса $(\overline{x_1 \vee x_2})$

$f_9(x_1,x_2)$	1 0 0 1	$x_1 \sim x_2$	эквивалентность (равнозначность)
$f_{10}(x_1,x_2)$	1 0 1 0	$\overline{x_2}$	отрицание x_2
$f_{11}(x_1,x_2)$	1 0 1 1	$x_2 \to x_1$	импликация от x_2 к x_1 $(x_1 \vee \overline{x_2})$
$f_{12}(x_1,x_2)$	1 1 0 0	$\overline{x_1}$	отрицание x_1
$f_{13}(x_1,x_2)$	1 1 0 1	$x_1 \to x_2$	импликация от x_1 к x_2 $(x_2 \vee \overline{x_1})$
$f_{14}(x_1,x_2)$	1 1 1 0	$x_1 \backslash x_2$	штрих Шеффера $(\overline{x_1 \wedge x_2})$
$f_{15}(x_1,x_2)$	1 1 1 1	1	Единичная

III. Системы элементарных логических функций.

Определение: Набор элементарных логических функций, используемый для записи любых логических выражений называется системой логических функций (СЛФ).

Определение: Набор элементарных логических функций достаточный для записи любых логических выражений называется функционально полной СЛФ (ФП СЛФ).

Определение: Если исключение любой элементарной логической функции из ФП СЛФ приводит к функциональной неполноте, то исходная ФП СЛФ называется безызбыточной. Иначе ФП СЛФ называется избыточной.

Теорема о функциональной полноте СЛФ (Постников, Яблонский).

СЛФ называется функционально полной в том и только том случае, если она включает хотя бы одну функцию, не принадлежащую к 5 важнейшим замкнутым классам:

1) – класс функций сохраняющих 0,
2) – класс функций сохраняющих 1,
3) – класс самодвойственных функций,
4) – класс монотонных функций,
5) – класс линейных функций.

Рассмотрим классы:

1) $f(0, 0, \dots 0) = 0$
2) $f(1, 1, \dots 1) = 1$ т. е. если подставить 0-е значение аргументов, то функция =0 или 1
3) $f(x_1, x_2, ..., x_n) = \overline{f(\overline{x_1}, \overline{x_2}, ..., \overline{x_n})}$
4) Пусть $x_{1i}=1$, а $x_{1j}=0$. Тогда $x_{1i}>x_{1j}$. Функция $f(x_1, x_2, \dots x_n)$ является монолитной, если для всех случаев, когда $x_{1i} \le x_{1j}; x_{2i} \le x_{2j}; ...; x_{ni} \le x_{nj}. i, j = \overline{0, 2^n - 1}$, во всех наборах выполняется неравенство $f(x_{1i}, x_{2i}, ..., x_{ni}) \le f(x_{1j}, x_{2j}, ..., x_{nj})$

Например:

x_1	$0(x_{10})$	$1(x_{11})$	$1(x_{12})$
x_2	$0(x_{20})$	$0(x_{21})$	$1(x_{22})$

видно, что $(x_{10}) \le (x_{11}) \le (x_{12})$ и $(x_{20}) \le (x_{21}) \le (x_{22})$, то выполняется: $f(x_{10}, x_{20}) \le f(x_{11}, x_{21}) \le f(x_{12}, x_{22})$

5) Если $F(x_1,x_2,...,x_n)=a_0\oplus a_1x_1\oplus a_2x_2\oplus...\oplus a_nx_x$, где $a_i\in\{0,1\},i\in\overline{0,n}$, то $f(x_1,x_2,\ldots,x_n)$ – называется линейной логической функцией

Пример:

		Сохр. «0»	Сохр. «1»	самодвой-ственные	монотонные	линейные
и	$\wedge$	+	+	-	+	-
или	$\vee$	+	+	-	+	-
инверсия	$\neg$	-	-	+	-	+
штрих Шеффера	\	-	-	-	-	-

Из таблицы на основании теоремы о ФП СЛФ:

1) $x_1\wedge x_2$ $x_1\vee x_2$ $\bar{x}$ - избыточная СЛФ, но наиболее распространённая (<u>каноническая</u> СЛФ)

2) $x_1\wedge x_2$ $\bar{x}$

3) $x_1\vee x_2$ $\bar{x}$ } безызбыточные СЛФ, есть другие безызбыточные ФПСЛФ

4) $x_1\backslash x_2$

Технический аналог булевой функции – комбинационная схема.

Вывод: для синтеза любой комбинационной схемы достаточно иметь лишь технические устройства, соответствующие ФП СЛФ.

<u>Основные законы и правила.</u>

<u>А. Преобразование выражений в булевой алгебре.</u>

№	логическое сложение	логическое умножение	название закона
1	x+0=x	$x\cdot 1=x$	-
2	x+1=1	$x\cdot 0=0$	-
3	x+x=x	$x\cdot x=x$	закон идемпотентности
4	$x+\bar{x}=1$	$x\cdot\bar{x}=0$	-
5	$\bar{\bar{x}}=x$		закон двойного отрицания
6	$x_1+x_2=x_2+x_1$	$x_1x_2=x_2x_1$	коммутативный закон
7	$x_1+x_1x_2=x_1$	$x_1(x_1+x_2)=x_1$	закон поглощения
8	$\overline{(x_1+x_2)}=\bar{x}_1\bar{x}_2$	$\overline{(x_1x_2)}=\bar{x}_1+\bar{x}_2$	закон Де Моргана (правило)
9	$(x_1+x_2)+x_3=$ $=x_1+(x_2+x_3)=$ $=x_1+x_2+x_3$	$(x_1x_2)x_3=x_1(x_2x_3)=$ $=x_1x_2x_3$	ассоциативный закон
10	$x_1+x_2x_3=$ $=(x_1+x_2)(x_1+x_3)$	$x_1(x_2+x_3)=$ $=x_1x_2+x_1x_3$	дистрибутивный закон (распределительный)

примем обозначения: $x_i^{\alpha_i}\begin{cases}x_i \text{ при } \alpha_i=1\\ \bar{x}_i \text{ при } \alpha_i=0\end{cases}$

<u>В. Теорема разложения.</u>

Любую переключательную функцию n аргументов можно представить в следующем виде:

$$f(x_1,x_2,...,x_n)=\bigvee_{(\alpha_1,\alpha_2,...,\alpha_k)} x_1^{\alpha_1}\cdot...\cdot x_k^{\alpha_k}\cdot f(\alpha_1,\alpha_2,...,\alpha_k,x_{k+1},...,x_n)$$

Следствие:

$$f(x_1,...,x_n)=\bar{x}_i f(x_1,...,x_{i-1},0,x_{i+1},...,x_n)\vee x_i f(x_1,...,x_{i-1},1,x_{i+1},...,x_n)$$

Теорема о СДНФ (совершенная дизъюнктивная нормальная форма):

Всякую переключательную функцию можно представить:

$$f(x_1,...,x_n)=\underset{f=1}{\vee}(x_1^{\alpha_1}\cdot...\cdot x_n^{\alpha_n})$$

Доказать самостоятельно.

Теорема о СКНФ (совершенная конъюнктивная нормальная форма):

$$f(x_1,...,x_n)=\underset{f=0}{\wedge}(\bar{x}_1^{\alpha_1}\vee...\vee\bar{x}_n^{\alpha_n})$$

Доказательство:

$$f(...)=\underset{f=1}{\vee}(x_1^{\alpha_1}...x_n^{\alpha_n})=\overline{\overline{\underset{f=1}{\vee}(x_1^{\alpha_1}...x_n^{\alpha_n})}}=\overline{\underset{f=1}{\wedge}\overline{x_1^{\alpha_1}...x_n^{\alpha_n}}}=\overline{\underset{f=1}{\wedge}(\bar{x}_1^{\alpha_1}\vee...\vee\bar{x}_n^{\alpha_n})}=\underset{f=0}{\wedge}(\bar{x}_1^{\alpha_1}\vee...\vee\bar{x}_n^{\alpha_n})$$

$\overline{\overline{\underset{f=1}{\vee}(x_1^{\alpha_1}...x_n^{\alpha_n})}}$ — закон_двойственности; $\overline{\underset{f=1}{\wedge}\overline{x_1^{\alpha_1}...x_n^{\alpha_n}}}$ — з_Де_Морг; $\overline{\underset{f=1}{\wedge}(\bar{x}_1^{\alpha_1}\vee...\vee\bar{x}_n^{\alpha_n})}$ — закон_Де_Моргана

ч. т. д.

Пример: Пусть переключательная функция $f(x_1,x_2,x_3)$ задана таблицей

x_1	x_2	x_3	$f(x_1,x_2,x_3)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

$$f(x_1,x_2,x_3)_{СДНФ}=\underset{где\_f=1}{из\_строк}=\bar{x}_1\bar{x}_2x_3+\bar{x}_1x_2\bar{x}_3+\bar{x}_1x_2x_3+x_1x_2x_3$$

$$\mathbf{f(x_1,x_2,x_3)_{СКНФ}=\underset{иверсий\_строк\_f=0}{из}=(x_1\vee x_2\vee x_3)\cdot(\bar{x}_1\vee x_2\vee x_3)\cdot(\bar{x}_1\vee x_2\vee\bar{x}_3)\cdot(\bar{x}_1\vee\bar{x}_2\vee x_3)}$$

таблица истинности избыточна более чем в 2 раза.

Пример. Упростить логическое выражение

$$(((\bar{x}_2\oplus 1)x_2x_3)\downarrow(x_2\vee x_2\bar{x}_3)\vee x_3\bar{x}_1)\setminus(\overline{x_2x_1}(x_3\vee\bar{x}_3))$$

используя законы и правила булевой алгебры.

$$(((\underbrace{\bar{x}_2\oplus 1}_{x_2})x_2x_3)\downarrow\underbrace{(x_2\vee x_2\bar{x}_3)}_{x_2}\vee x_3\bar{x}_1)\setminus(\overline{x_2x_1}\underbrace{(x_3\vee\bar{x}_3)}_{1})=$$

$$=((x_2x_3\downarrow x_2)\vee x_3\bar{x}_1)\setminus\overline{x_2x_1}=\underset{функции\downarrow и\setminus}{раскрываем}=\overline{\overline{\underbrace{x_2x_3\vee x_2}_{x_2}\vee x_3\bar{x}_1}}\cdot\overline{x_2x_1}=\overline{(\bar{x}_2\vee x_3\bar{x}_1)}\cdot\overline{x_2x_1}=$$

$$=\underset{Де\_Моргана}{применим\_правило}=\overline{\bar{x}_2\vee x_3\bar{x}_1}\vee\overline{x_2x_1}=\underset{применяется\_правило\_Де\_Моргана}{последовательно\_дважды}=\bar{\bar{x}}_2\cdot\overline{x_3\bar{x}_1}\vee x_2x_1=$$

$$= x_2(\overline{x_3} \vee x_1) \vee x_2 x_1 = x_2(\underbrace{\overline{x_3} \vee x_1 \vee x_1}_{x_1}) = x_2(\overline{x_3} \vee x_1)$$

Задание 44. Упростить выражение

$$\overline{\overline{((x_1 \vee x_3) \cdot \overline{x_2(\overline{x_1} \vee x_3)})}} \cdot \overline{(x_3 x_1 \vee x_1(x_2 \vee \overline{x_2}))} \cdot \overline{x_3}$$

Задание 45. Упростить выражение

$$(x_2(\overline{x_1} \oplus 1) \vee \overline{(x_1 \vee \overline{x_2})}) \cdot (x_1 \downarrow (\overline{x_2} \vee \overline{x_2} x_3)) \vee \overline{(x_1 \vee \overline{x_3} \vee \overline{(\overline{x_1} \setminus 1)})}$$

Задание 46. Привести выражение

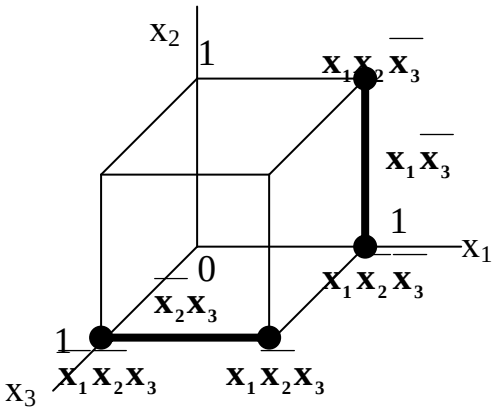
$$((\overline{x_1} x_3 x_2) \setminus \overline{(x_1 \overline{x_2} \vee \overline{x_3})}) \downarrow x_1$$ к виду содержащему лишь операции и дизъюнкции.

Минимизация выражений в булевой алгебре.

Цель – упростить техническую реализацию (комбинационную схему)

I. Геометрическая интерпретация задачи упрощения и минимизации логических функций.

x_3	x_2	x_1	$f(x_1,x_2,x_3)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0
таблица истинности			



Из _ таблицы $\rightarrow$ СДНФ

$$f = x_1 \overline{x_2}\,\overline{x_3} \vee x_1 x_2 \overline{x_3} \vee \overline{x_1} x_2 x_3 \vee x_1 \overline{x_2} x_3$$

Если 2 смежные вершины куба заменить ребром, то получим упрощение: $f = x_1 \overline{x_3} \vee \overline{x_2} x_3$ возможен алгоритм.

II. Метод Квайна-Мак-Класки.

Определения:

Рангом конъюнкции называется число различных аргументов (с инверсией или без) входящих в конъюнкцию.

Длинной ДНФ называется число попарно различимых конъюнкций в ДНФ.

Кратчайшей ДНФ называют ДНФ с наименьшей, среди всех возможных ДНФ, длинной.

Минимальной ДНФ называется ДНФ у которой наименьший среди всех возможных ДНФ ранг конъюнкции.

Сокращённой ДНФ называется ДНФ состоящая из простейших конъюнкций.

$$\underset{\text{одна}}{\text{ТИ}} \rightarrow \underset{\text{одна}}{\text{СДНФ}} \rightarrow \underset{\text{может\_быть\_много}}{\text{Сокр.ДНФ}} \rightarrow \underset{\text{тупиковая*}}{\text{ТДНФ}} \rightarrow \text{МДНФ}$$

* - отбрасываются члены сокр. ДНФ и проверяется истинность

1.3.2. Формы представления логических функций.

Задать логическую функцию можно в различных формах: а)словесно, б)таблицей истинности, в)алгебраически (формульно), г)картами Карно, д)в цифровой форме, е)в строчной форме и др.

Словесное представление логических функций.

Покажем на примере функции конъюнкции. Словесно эту функцию можно представить так: логическая функция принимает значение равное 1 лишь в том случае, когда все её аргументы равны 1.

Табличное представление логических функций.

Каждому набору аргументов ставится в соответствие значение функции и представляется таблицей. Если наборы аргументов упорядочены по возрастанию, то такая таблица называется таблицей истинности (ТИ).

Алгебраическое представление логических функций (в СДН и СКН формах).

Примем обозначение $x_i^{\alpha_i} = \begin{cases} x_i, \text{при} \alpha = 1 \\ \bar{x}_i, \text{при} \alpha = 0 \end{cases}, i = \overline{1,n}.$, где n – количество аргументов логической функции. Тогда любую логическую функцию (кроме F=0) можно представить в совершенной дизъюнктивной нормальной форме (СДНФ).

$$F(x_1,...,x_n) = \bigvee_{F=1}(x_1^{\alpha_1},...,x_n^{\alpha_n})$$

или в совершенной конъюнктивной нормальной форме (СКНФ)

$$F(x_1,...,x_n) = \bigwedge_{F=0}(\bar{x}_1^{\alpha_1},...,\bar{x}_n^{\alpha_n}).$$

Для логической функции, представленной ТИ, СДНФ строится следующим образом:

- выделяются наборы аргументов (строки ТИ) при которых функция равны 1;
- из аргументов каждого выделенного набора формируется соответствующая конъюнкция (аргументы, значения которых в наборе равны 0, входят в конъюнкцию с инверсией);
- сформированные конъюнкции соединяются знаком дизъюнкций.

При построении СКНФ порядок действий следующий:

- выделяются наборы аргументов при которых функция равна 0;
- из аргументов каждого выделенного набора формируется соответствующая дизъюнкция (аргументы, значения которых в наборе равны 1, входят в дизъюнкцию с инверсией);
- сформированные дизъюнкции соединяются знаками конъюнкций.

Пример. Представить логическую функцию $F_1(x_1,x_2,x_3)$, заданную таблицей истинности, в СДНФ и СКНФ.

x_1	x_2	x_3	F_1	F_2
0	0	0	0	1
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0

$$F_1(x_1,x_2,x_3)_{СДНФ} = \overline{x_1}\,\overline{x_2}x_3 \vee x_1\overline{x_2}\,\overline{x_3} \vee x_1\overline{x_2}x_3 \vee x_1x_2x_3$$

$$F_1(x_1,x_2,x_3)_{СКНФ} = (x_1 \vee x_2 \vee x_3)\cdot(x_1 \vee \overline{x_2} \vee x_3)\cdot(x_1 \vee \overline{x_2} \vee \overline{x_3})\cdot(\overline{x_1} \vee \overline{x_2} \vee x_3)$$

Задание 47. Логическая функция $F_2(x_1,x_2,x_3)$ задана ТИ. Представить функцию в СДНФ и СКНФ.

Задание 48. Представить логическую функцию $(x_1 \cdot \overline{x_2}) \oplus (x_2 \vee \overline{x_3})$ таблицей истинности, а также в СКНФ и СДНФ.

Представление логических функций картами Карно.

Карта Карно – это графическое представление таблицы истинности. Между строками ТИ и клетками карты Карно (КК) существует взаимно однозначное соответствие. Каждая клетка КК заполняется нулём или единицей в соответствии со значением функции, вычисленной при значениях аргументов на пересечении которых расположена данная клетка.

Например, для 2-х переменных ТИ и КК будут выглядеть следующим образом:

x_1	x_2	$F(x_1,x_2)$
0	0	F(0,0)
0	1	F(0,1)
1	0	F(1,0)
1	1	F(1,1)

x_1 \ x_2	0	1
0	F(0,0)	F(0,1)
1	F(1,0)	F(1,1)

Карты Карно для 3-х и 4-х переменных выглядят так:

КК для 3-х переменных

x_1 \ x_2x_3	00	01	11	10
0	F(0,0,0)	F(0,0,1)	F(0,1,1)	F(0,1,0)
1	F(1,0,0)	F(1,0,1)	F(1,1,1)	F(1,1,0)

КК для 4-х переменных

x_1x_2 \ x_3x_4	00	01	11	10
00	…	…	…	…
01	…	…	…	…
11	…	…	…	…
10	…	…	…	…

Следует обратить внимание на расположение наборов аргументов по координатным осям КК. Последовательность наборов определяется изменением значения только одной переменной. Поэтому расположение 00 01 10 11 является неправильным, т. к. при переходе от 01 к 10 изменяются сразу две переменные. Правильным будет расположение 00 01 11 10.

Задание 49. Представить логическую функцию $\overline{x_2}x_3 + \overline{x_1}\,\overline{x_3}$ картой Карно.

Задание 50. Представить логическую функцию $x_1\overline{x_4} \downarrow \overline{x_2}\,\overline{x_3}x_4$ картой Карно.

Цифровая форма представления логических функций.

Если аргумент x_i в таблице истинности располагать в порядке возрастания индекса i, то наборы аргументов в таблице можно сокращённо обозначать десятичными числами. Тогда СДНФ в цифровой форме будет представлять сумму чисел, соответствующих наборам аргументов на которых функция принимает значение равное единице.

СКНФ представляется произведением десятичных чисел, соответствующих наборам аргументов на которых функция принимает значение равное нулю. Но в этом случае при получении чисел, соответствующих наборам аргументов, значения аргументов нужно заменить на обратное.

Десятичные числа в цифровой форме располагаются в порядке возрастания.

Пример. Представить логическую функцию, заданную таблицей, в цифровых формах.

x_1	x_2	x_3	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

При построении $F_{СДНФ}$ в цифровой форме следует выделять в ТИ наборы аргументов 001, 010, 100, 110, 111 и представлять их десятичными числами 1, 2, 4, 6, 7. Тогда $F_{СДНФ} = \sum_0^7 (1,2,4,6,7)$.

При построении $F_{СКНФ}$ в цифровой форме выделяем наборы 000, 011, 101, заменяем их на обратные, т. е. на 111, 100, 010 и представляем их десятичными числами 7, 4, 2, располагая по возрастанию: $F_{СКНФ} = \sum_0^7 (2,4,7)$.

Возможен переход от цифровой формы к алгебраической. Для рассмотренного примера переход будет следующим:

$$\sum_0^7 (1,2,4,6,7) = \sum_0^7 (001,010,100,110,111) = \overline{x_1}\,\overline{x_2}x_3 + \overline{x_1}x_2\overline{x_3} + x_1\overline{x_2}\,\overline{x_3} + x_1x_2\overline{x_3} + x_1x_2x_3$$

$$\prod_0^7 (2,4,7) = \prod_0^7 (010,100,111) = (\overline{x_1} + x_2 + \overline{x_3}) \cdot (x_1 + \overline{x_2} + \overline{x_3}) \cdot (x_1 + x_2 + x_3)$$

При цифровых формах представления логических функций переход от СДНФ к СКНФ или наоборот выполняется по следующим правилам.

Пусть задана функция $F_{СДНФ} = \sum_0^{N-1} (\alpha_1,...,\alpha_t)$.

Перейдём к обратной функции $\bar{F}_{СДНФ} = \sum_{0}^{N-1}(\beta_1,...,\beta_k), t + k = N,$ где N – полное количество наборов аргументов функции. Обратная функция $\bar{F}_{СДНФ}$ получается путём записи в неё десятичных чисел, отсутствующих в представлении $F_{СДНФ}$. Используя $\bar{F}_{СДНФ}$ можно получить $F_{СКНФ} = \prod_{0}^{N-1}(\varphi_1,...,\varphi_k)$, где $\varphi_1 = (N-1) - \beta_k,...,\varphi_i = (N-1) - \beta_i,...,\varphi_k = (N-1) - \beta_1.$

Пример. Для функции, рассмотренной в предыдущем примере, выполнить переход от $F_{СДНФ}$ к $F_{СКНФ}$ в цифровой форме.

Задана $F_{СДНФ} = \sum_{0}^{7}(\alpha_1,\alpha_2,\alpha_3,\alpha_4,\alpha_5) = \sum_{0}^{7}(1,2,4,6,7).$

Перейдём к обратной функции: $\bar{F}_{СДНФ} = \sum_{0}^{7}(\beta_1,\beta_2,\beta_3) = \sum_{0}^{7}(0,3,5).$ Так как N=8, то $\varphi_1 = (N-1) - \beta_3 = 7-5 = 2,$ $\varphi_2 = (N-1) - \beta_2 = 7-3 = 4,$ $\varphi_3 = (N-1) - \beta_1 = 7-0 = 7.$

Следовательно, $F_{СКНФ} = \prod_{0}^{7}(2,4,7).$

Таблица 3

x_1	x_2	x_3	x_4	F_1	F_2
0	0	0	0	0	1
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	1	1
0	1	1	1	1	0
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	1	1
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	0	1

Задание 51. Представить логическую функцию F_1, заданную таблицей истинности (табл. 3), в цифровых формах.

Задание 52. Представить логическую функцию F_2, заданную таблицей истинности (табл. 3), в цифровых формах.

Задание 53. Представить $F = x_1 \cdot \overline{x_2} \cdot (x_3 \oplus 1) + \overline{x_2 \cdot x_3}$ в цифровых формах

Задание 54. Представить $F_{СДНФ} = \overline{x_1}x_2\overline{x_3}x_4 + x_1\overline{x_2}\overline{x_3}\overline{x_4} + x_1\overline{x_2}x_3x_4 + x_1x_2x_3\overline{x_4}$ в цифровой форме и выполнить переход к $F_{СКНФ}$.

Задание 55. Представить $F_{СКНФ} = (x_1 + x_2 + x_3) \cdot (x_1 + x_2 + x_3) \cdot (x_1 + x_2 + x_3)$ в цифровой форме и выполнить переход к $F_{СДНФ}$.

Строчная форма представления логических функций.

Строчное представление целесообразно при обработке логических функций на ЭВМ. При таком представлении столбец значений функции из таблицы истинности записывается горизонтально – строчкой и выделяется рамкой.

Пример. Представить функцию F_1 (таблица 3) в строчной форме.

$F_1 = \sum_{4} 0001001111110000.$

Переход от цифровой формы представления функции к строчной форме и наоборот выполняется просто. Покажем на примере.

Пример. Представить функцию, заданную цифровыми формами $\sum_{0}^{7}(1,2,4,6,7)$ и $\prod_{0}^{7}(2,4,7)$ в строчных формах.

$$F_{СДНФ} = \sum_{0}^{7}(1,2,4,6,7) = \sum_{3} 01101011$$

$$F_{СКНФ} = \prod_{0}^{7}(2,4,7) = \prod_{3} 00101001$$

$$01234567 - \text{номера\_разрядов}$$

Как видно из примера, единицы при строчной записи расположены в разрядах, соответствующих десятичным числам при цифровой записи.

Задание 56. Представить функцию F_2 (табл. 3) в строчной форме и выполнить переход к цифровой форме представления.

Задание 57. Представить функцию $F = x_1 \cdot \overline{x_2} \cdot (x_3 \oplus 1) + \overline{x_2 \cdot x_3}$ в строчной форме.

1.3.3. Минимизация логических функций.

Задача минимизации заключается в сведении логической функции к виду, позволяющему выполнить наиболее простую её техническую реализацию.

Для минимизации применяется различные методы: метод алгебраических преобразований, метод Карно, метод Морреля и другие.

Ниже рассматриваются два метода: метод Карно, который эффективен при ручной минимизации и метод Морреля, обычно используемый при минимизации функции на ЭВМ.

Минимизация методом Карно (рассмотрим три варианта: А, Б, В).

А. Минимизация в дизъюнктивно нормальной форме (получение МДНФ).

Метод минимизации логических выражений по карте Карно состоит в выделении групп клеток содержащих 1-цы и в составлении минимизированного логического выражения по этим группам. При этом каждая 1-ца в карте Карно должна войти хотя бы в одну группу.

Группой клеток называется совокупность соседних клеток, составляющих прямоугольник размерности $2^a \times 2^b$, где a и b = 0, 1, 2, … .

Каждой группе размерности $2^a \times 2^b$ соответствует конъюнкция состоящая из $n - a - b$ переменных, где n – полное число аргументов логической функции. В конъюнкцию включаются все аргументы, значения которых не изменяются в пределах соответствующей группы. Если переменная в пределах группы равна 0, то она входит в конъюнкцию с инверсией, а если 1 – без инверсии.

Составление минимизированного выражения заключается в объединении дизъюнкциями конъюнкций, соответствующих всем выделенным группам.

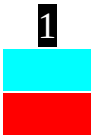
Два принципа формирования группы:

- каждая группа должна быть как можно больше, т. е. max(a+b);
- число групп должно быть как можно меньше.

Пример. Дана карта Карно для функции 4-х переменных. Получить МДНФ этой функции.

$$f(x_1,x_2,x_3,x_4)=\underset{\text{группа1}}{\bar{x}_3\bar{x}_4}\vee\underset{\text{группа2}}{x_1x_2\bar{x}_3}\vee\underset{\text{группа3}}{\bar{x}_1x_3}$$

x_1x_2 \ x_3x_4	00	01	11	10
00	1	0	1	1
01	1	0	1	1
11	**1**	**1**	0	0
10	1	0	0	0



группа 2
группа 1
группа 3

Например, конъюнкция соответствующая 1-ой группе состоит из двух аргументов, т. к. $n-a-b=4-2-0=2$, а для 2-ой группы $4-1-0=3$.

Следует обратить внимание, что группы могут пересекаться, т. е. одна и та же 1-ца может входить в разные группы.

В КК для 3-х и 4-х переменных клетки могут объединяться в так называемые разорванные группы, например:

1	0	0	1
1	0	0	1

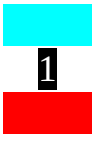
0	0	1	0
1	0	0	1
1	0	0	1
0	0	1	0

1	0	0	1
0	0	0	0
0	0	0	0
1	0	0	1

Пример. Дана карта Карно для функций 4-х переменных. Получить МДНФ этой функции.

$$f(x_1,x_2,x_3,x_4)=\underset{\text{1гр}}{\bar{x}_2\bar{x}_4}\vee\underset{\text{2гр}}{x_1\bar{x}_4}\vee\underset{\text{3гр}}{\bar{x}_1x_2\bar{x}_3x_4}$$

x_1x_2 \ x_3x_4	00	01	11	10
00	1	0	0	1
01	0	1	0	0
11	**1**	0	0	**1**
10	**1**	0	0	**1**



группа 1
группа 2
группа 3

Задание 58. Получить МДНФ для функции F_1 заданной в таблице 4.

Задание 59. Получить МДНФ для функции F_2 заданной в таблице 4.

Таблица 4

x_1	x_2	x_3	x_4	F_1	F_2	F_3	F_4	F_5	F_6
0	0	0	0	0	1	1	1	0	0
0	0	0	1	1	0	1	1	0	1
0	0	1	0	0	1	0	0	1	-
0	0	1	1	0	1	0	1	1	0
0	1	0	0	1	1	1	0	1	1
0	1	0	1	1	0	0	0	1	0
0	1	1	0	1	1	1	1	1	1
0	1	1	1	1	0	1	0	0	1
1	0	0	0	1	1	0	0	-	-
1	0	0	1	1	0	1	1	0	-
1	0	1	0	1	1	0	0	-	0
1	0	1	1	0	1	0	1	1	0
1	1	0	0	0	0	0	1	-	1
1	1	0	1	0	0	1	0	1	1
1	1	1	0	0	0	0	1	0	0
1	1	1	1	0	0	1	0	1	1

Б. Минимизация в конъюнктивной нормальной форме (получение МКНФ).

Здесь возможны 2 способа.

1. Перейти от МДНФ к МКНФ при помощи правила Де Моргана.

2. Использовать клетки КК содержащие нули.

Составление МКНФ заключается в объединении конъюнкциями дизъюнкций, соответствующих всем выделенным группам. В дизъюнкцию включаются все переменные, значения которых не изменяются в пределах группы. Причём, если переменная в пределах группы равна 0, то она входит в дизъюнкцию без инверсии, а если равна 1, то с инверсией.

Пример. Дана карта Карно для функции 4-х переменных. Получить МКНФ этой функции.

$$F(x_1, x_2, x_3, x_4) =$$
$$= (\overline{x_3} \vee \underset{1}{\overline{x_4}}) \cdot (\overline{x_1} \vee \underset{2}{\overline{x_4}}) \cdot (x_2 \vee \underset{3}{\overline{x_4}}) \cdot (x_1 \vee \underset{4}{\overline{x_2}} \vee x_4)$$

x_1x_2 \ x_3x_4	00	01	11	10
00	1	0	0	1
01	0	1	0	0
11	1	0	0	1
10	1	0	0	1

группа 1
группа 2
группа 3
группа 4

Задание 60. Получить МКНФ для функции F_4 заданной в таблице 4.

Задание 61. Получить МКНФ для функции F_3 заданной в таблице 4.

В. Минимизация неполностью определённых функций.

Неполностью определённые функции это такие функции, которые определены не на всех 2^n наборах аргументов.

На карте Карно неопределённое условие обозначается прочерком (-). Клетки с прочерком могут произвольным образом включаться в группы, как при построении МДНФ, так и при построении МКНФ. Более того, их можно вообще никуда не включать (игнорировать).

Пример. Дана карта Карно для функции 4-х переменных. Получить МДНФ и МКНФ этой функции.

$$F(x_1,x_2,x_3,x_4)_{МДНФ} = \underset{1}{x_1\overline{x_4}} \vee \underset{2}{x_2x_4}$$

$$F(x_1,x_2,x_3,x_4)_{МКНФ} = \underset{1}{(x_2 \vee \overline{x_4})} \cdot \underset{2}{(x_1 \vee x_4)}$$

x_1x_2 \ x_3x_4	00	01	11	10
00	-	0	0	-
01	0	1	-	0
11	1	-	-	-
10	1	0	0	1

группа 1
группа 2
группа 1
группа 2

Задание 62. Получить МДНФ для функции F_5 заданной в таблице 4.

Задание 63. Получить МДНФ и МКНФ для функции F_6 заданной в таблице 4.

Минимизация методом Морреля.

В основе метода лежит теорема разложения.

Теорема разложения. Любую логическую функцию от n переменных можно свести к двум функциям от (n-1) переменных, проведя разложение вида

$$F(x_1,...,x_n) = \overline{x_i} \cdot F_0(x_1,...,x_{i-1},0,x_{i+1},...,x_n) + x_i \cdot F_1(x_1,...,x_{i-1},1,x_{i+1},...,x_n)$$

Сущность метода заключается в последовательном разложении функции по переменным и в анализе остаточных функций на равенство их нулю или единице.

$$\underbrace{F(x_1,...,x_n)}_{\text{минимизируемая функция}} = \overline{x_1} \cdot \underbrace{F_0(0,x_2,...,x_n)}_{\text{остоточная функция}} + x_1 \cdot \underbrace{F_1(1,x_2,...,x_n)}_{\text{остаточная функция}}$$

Если остаточная функция равна нулю, то член, её включающий, равен нулю и исключается из разложения. Если остаточная функция равна единице, то переменная или конъюнкция переменных, связанных с ней, входит в тупиковую форму, а член из дальнейшего рассмотрения исключается. Все оставшиеся после анализа остаточные функции разлагаются по следующей переменной и снова анализируются и т. д. Минимизация заканчивается когда все остаточные функции будут равны 0 или 1.

Для получения формы, наиболее близкой к минимальной, Моррель предложил в формулу разложения добавить так называемый «терм Морреля», который добавляется всякий раз при разложении по любой переменной.

Формула разложения примет вид: $F(...) = \overline{x_1} \cdot F_0(...) + x_1F_1(...) + \underbrace{F_0(...) \cdot F_1(...)}_{\text{терм Морелля}}$

Покажем, что терм Морелля не изменит функцию:

$$F(...) = \overline{x_1} \cdot F_0(...) + x_1 \cdot F_1(...) + F_0(...) \cdot F_1(...)\underbrace{(\overline{x_1} + x_1)}_{1} =$$

$$= \overline{x_1} \cdot F_0(...) + x_1 \cdot F_1(...) + F_0(...) \cdot F_1(...) \cdot \overline{x_1} + F_0(...) \cdot F_1(...) \cdot x_1 =$$

$$= \overline{x_1} \cdot F_0(...) \cdot \underbrace{(1 + F_1(...))}_{1} + x_1 \cdot F_1(...) \cdot \underbrace{(1 + F_0(...))}_{1} = \overline{x_1} \cdot F_0(...) + x_1 \cdot F_1(...).$$

При анализе остаточных функций на каждом этапе разложения проверяются условия поглощения термом Морреля других остаточных функций.

Минимизацию методом Морреля удобно проводить над функцией представленной в строчной форме (т. к. метод реализуем на ЭВМ). Разложение функции, представленной в строчной форме, по переменной x_i означает её

разбиение на две части. В первую часть входят те наборы, на которых $x_i=0$, а во вторую – на которых $x_i=1$.

Пример. Пусть задана функция F(A,B,C,D) в строчной форме.

$F(A,B,C,D)=\sum_{A,B,C,D}0000000100010111$. Выполнить разложение функции по переменной A.

$$F(A,B,C,D)=\sum_{A,B,C,D}0000000100010111=\overline{A}\cdot\sum_{B,C,D}00000001+A\cdot\sum_{B,C,D}00010111.$$

Теперь получим терм Морреля и добавим в разложение.

получение терма Морреля поразрядным умножением

00000001
00010111
00000001

-терм Морреля

Следовательно, $F(A,B,C,D)=\overline{A}\cdot\sum_{B,C,D}00000001+A\cdot\sum_{B,C,D}00010111+\sum_{B,C,D}00000001$

Пример. Минимизировать методом Морреля функцию

$$F(A,B,C,D)=\sum_{0}^{7}(1,3,4,5,6,7).$$

$$F=\sum_{A,B,C}0101111=\overline{A}\cdot\sum_{B,C}0101+A\cdot\sum_{B,C}1111+\sum_{B,C}0101=\sum_{B,C}0101=$$

$$=\overline{B}\sum_{C}01+B\sum_{C}01+\sum_{C}01=\sum_{C}01=\overline{C}\sum 0+C\sum 1+\sum 0=C$$

Таким образом, получим F(A,B,C,D)=A+C.

Задание 64. Выполнить разложение функции F_1 (таблица 3) по переменной x_1.

Задание 65. Минимизировать функцию F_2 (таблица 3) методом Морелля.

Задание 66. Минимизировать функцию $F_{СДНФ}=\sum_{0}^{15}(2,4,5,6,8,9,10,12,14,15)$ методом Морреля.

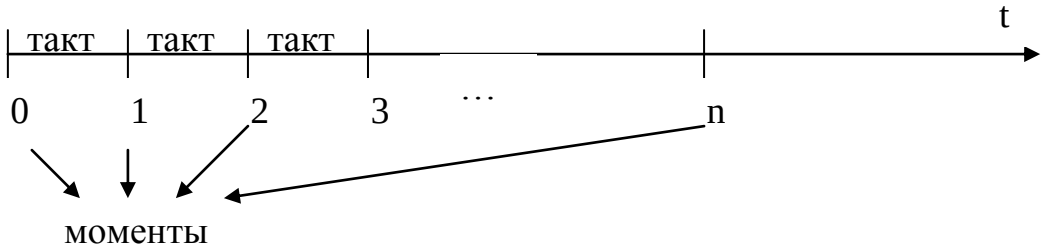
1.4. Синтез и анализ комбинационных схем.

Устройство, осуществляющее дискретное преобразование информации ЭВМ, называется автоматом. Существуют два основных вида дискретных автоматов: комбинационные автоматы (схемы) и конечные автоматы.

Введём: $X(t)=\{x_1, x_2, \ldots, x_i, \ldots, x_n\}$ – множество входных дискретных сигналов; $x_i\in\{0,1\}, i=\overline{1,n}$.

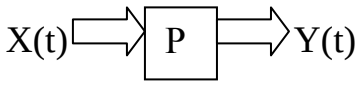
$Y(t)=\{y_1, y_2, \ldots, y_j, \ldots, y_m\}$ – множество выходных дискретных сигналов; $y_j\in\{0,1\}, j=\overline{1,m}$.

Здесь t – дискретное время, определяется моментами перехода автомата из одного состояния в другое.



Определение. Если совокупность выходных сигналов (выходного слова) Y(t) зависит только от совокупности входных сигналов (входного слова) X(t) и не зависит от внутренних состояний автомата, то такой автомат называется комбинационным автоматом (комбинационной схемой).

Структурная схема комбинационного автомата:



Здесь P – функция преобразования.

Закон функционирования комбинационной схемы определяется заданием соответствия между её входными и выходными словами Y(t)=P[X(t)] и может быть дан таблицей истинности, в аналитический форме и др.

1.4.1. Алгоритм синтеза комбинационной схемы.

1. Задать закон функционирования комбинационной схемы.
2. Провести минимизацию функции преобразования P, получить $P_{МДНФ}$ и $P_{МКНФ}$.
3. Преобразовать $P_{МДНФ}$ и $P_{МКНФ}$ в соответствии с заданным базисом логических функций (используемыми логическими элементами) и выбрать наиболее компактную форму представления P.
4. Построить функциональную схему устройства.
5. Выполнить схемотехническую коррекцию схемы с учётом характеристик и параметров используемых логических элементов:

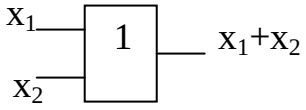
- коэффициента объединения (количество входов элемента),
- коэффициента разветвления (характеризует нагрузочную способность – максимально допустимое количество элементов подключаемых к входу),
- уровня логических 0 и 1,
- помехозащитности и др.

6. Провести тестирование полученной схемы на соответствие закону функционирования. Этот этап является уже частью анализа схемы и ставит задачей убедится в её работоспособности.

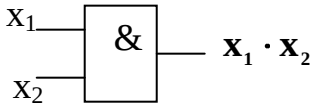
1.4.2. Функциональные схемы логических элементов.

Приведём лишь функциональные схемы, реализующие простейшие логические функции.

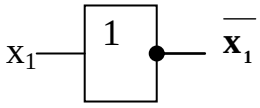
дизъюнкция («или»)



конъюнкция («и»)

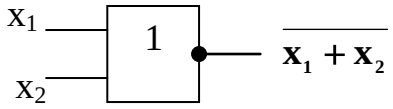


инверсия («не»)

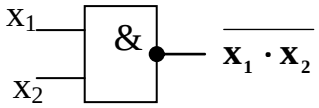


(здесь и в дальнейшем будет употребляться закрашенный кружок в обозначении инверсии вместо не закрашенного как принято)

стрелка Пирса («или-не»)

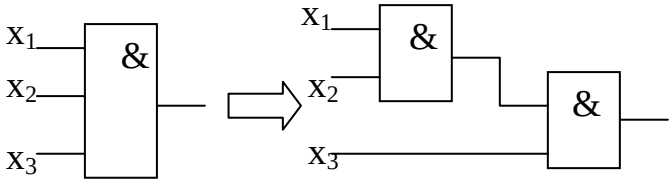


штрих Шеффера («и-не»)

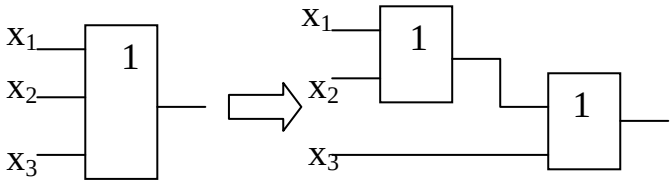


Свести 3-х и 4-х входовые логические схемы к 2-х входовым можно следующим образом:

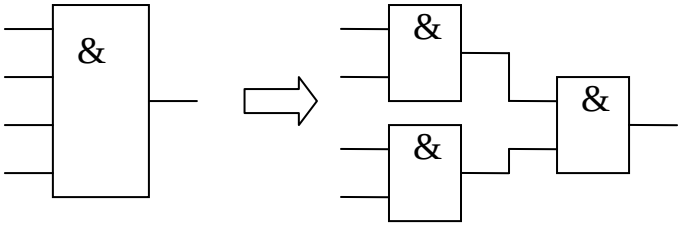
«3 и»



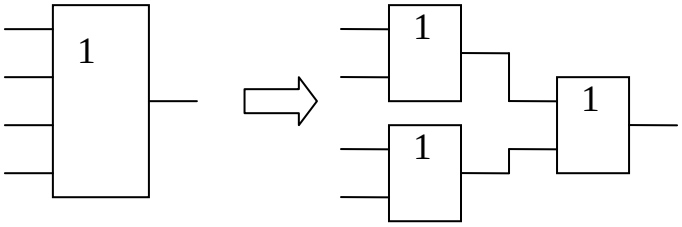
«3 или»



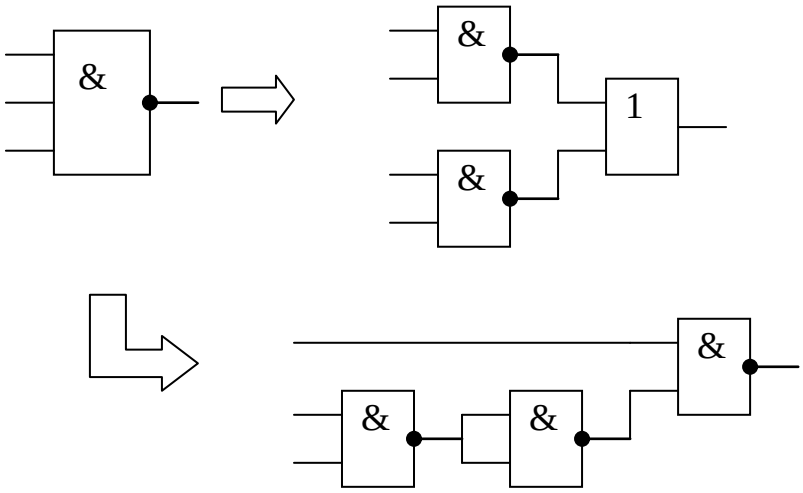
«4 и»



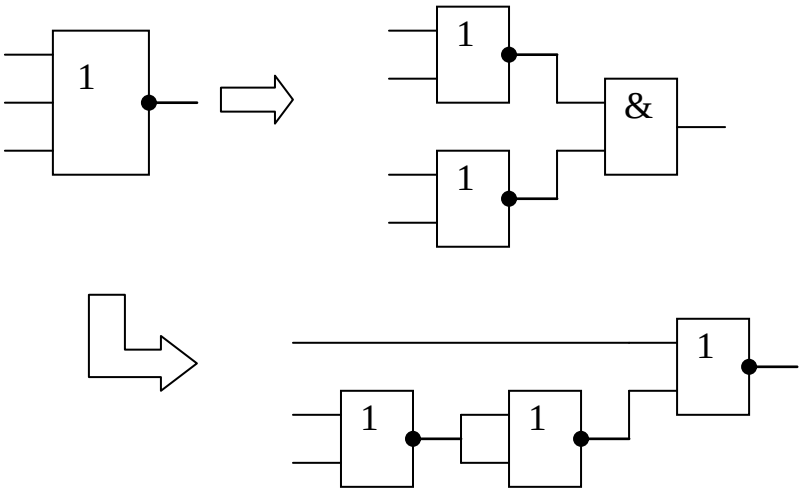
«4 или»



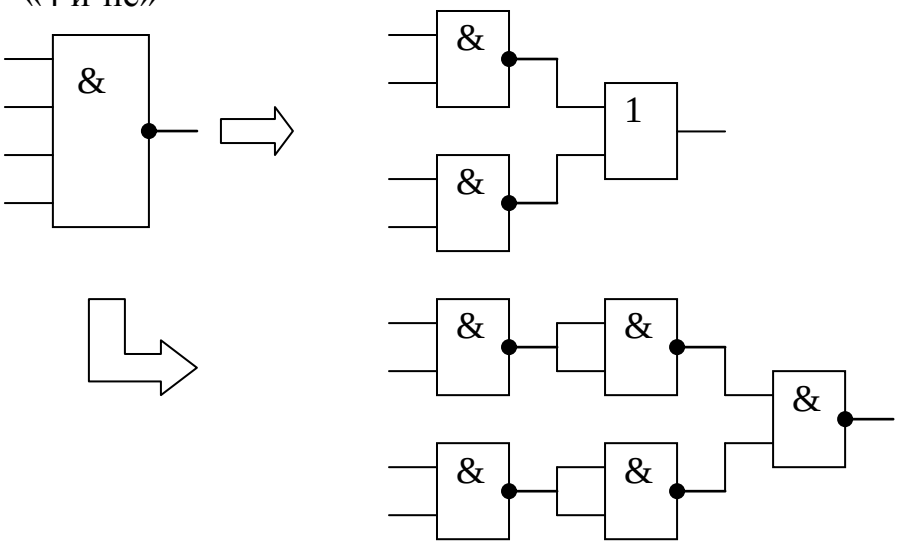
«3 и-не»



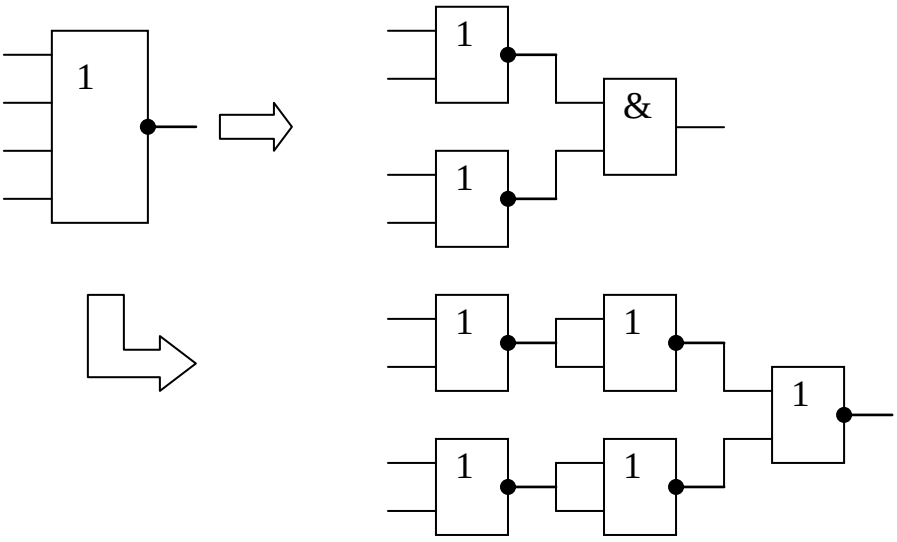
«3 или-не»



«4 и-не»



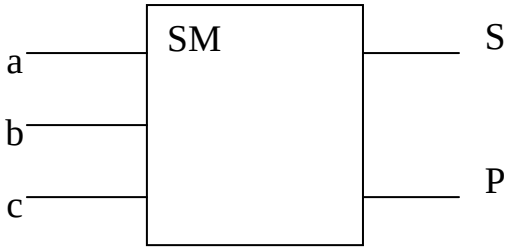
«4 или-не»



1.4.3. Пример синтеза полного одноразрядного двоичного сумматора.

Пример. Синтезировать схему полного одноразрядного двоичного сумматора используя двухвходовые логические элементы, реализующие логическую функцию «и-не».

Представим сумматор в виде:



a и b – одноразрядные двоичные слагаемые.,

c – перенос из младшего разряда,

S – сумма,

P – перенос в старший разряд.

$S=F_1(a,b,c)$, $P=F_2(a,b,c)$.

При синтезе реализуем последовательно этапы алгоритма.

1. Зададим закон функционирования сумматора в виде ТИ.

c	b	a	S	P
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

2. Проведём минимизацию функций S и P методом Карно. По ТИ составим карты Карно:

c \ ab	00	01	11	10
0	0	1	0	1
1	1	0	1	0

для S

c \ ab	00	01	11	10
0	0	0	1	0
1	0	1	1	1

для P

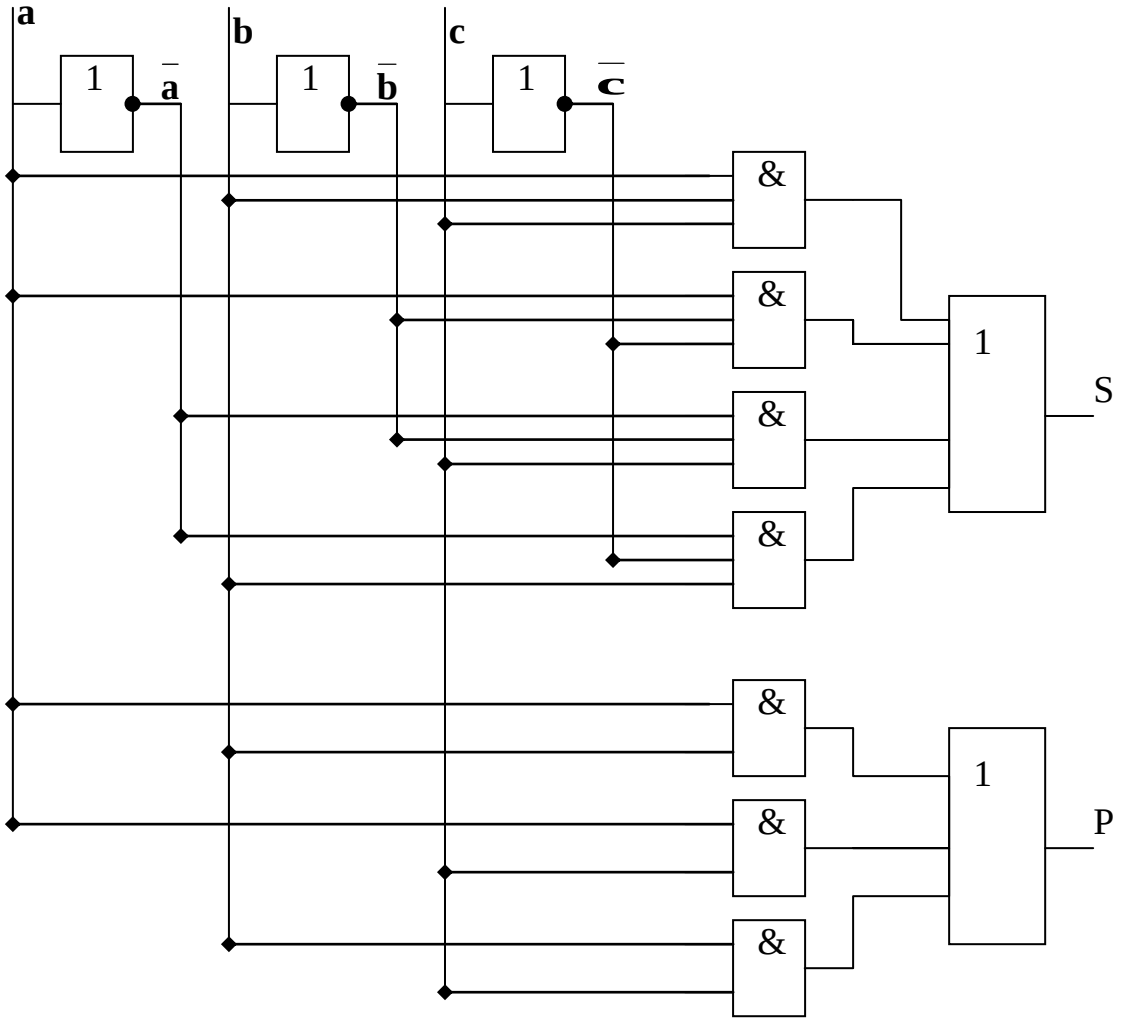
$$S_{МДНФ} = abc + a\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}b\bar{c}$$

$$S_{МКНФ} = (a+b+c)\cdot(a+\bar{b}+\bar{c})\cdot(\bar{a}+\bar{b}+c)\cdot(\bar{a}+b+\bar{c})$$

$$P_{МДНФ} = ab + ac + bc$$

$$P_{МКНФ} = (a+b)\cdot(a+c)\cdot(b+c)$$

Проверим предварительное построение схем сумматора непосредственно по выражениям $S_{МДНФ}$ и $P_{МДНФ}$ без преобразования их к заданным двухвходовым элементам «и-не».



3. Преобразуем полученные выражения в соответствии с задуманными логическими элементами «и-не».

$$S_{МДНФ} = \overline{abc + a\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}b\bar{c}} = \overline{\overline{abc} + \overline{a\bar{b}\bar{c}} + \overline{\bar{a}\bar{b}c} + \overline{\bar{a}b\bar{c}}}$$

$$S_{МКНФ} = \overline{\overline{(a+b+c)}\cdot\overline{(a+\bar{b}+\bar{c})}\cdot\overline{(\bar{a}+\bar{b}+c)}\cdot\overline{(\bar{a}+b+\bar{c})}} = \overline{(\bar{a}bc)}\cdot\overline{(a\bar{b}c)}\cdot\overline{(ab\bar{c})}\cdot\overline{(\bar{a}\bar{b}\bar{c})} =$$

$$= \overline{(\bar{a}bc)}\cdot\overline{(a\bar{b}c)}\cdot\overline{(ab\bar{c})}\cdot\overline{(\bar{a}\bar{b}\bar{c})}.$$

Для построения $S_{МДНФ}$ и $S_{МКНФ}$ на двухвходовых элементах «и-не» потребуется соответственно 20 и 21 логический элемент. Поэтому для построения части схемы, реализующей S, выберем $S_{МДНФ}$.

$$P_{МДНФ} = \overline{\overline{ab + ac + bc}} = \overline{\overline{ab}\cdot\overline{ac}\cdot\overline{bc}},$$

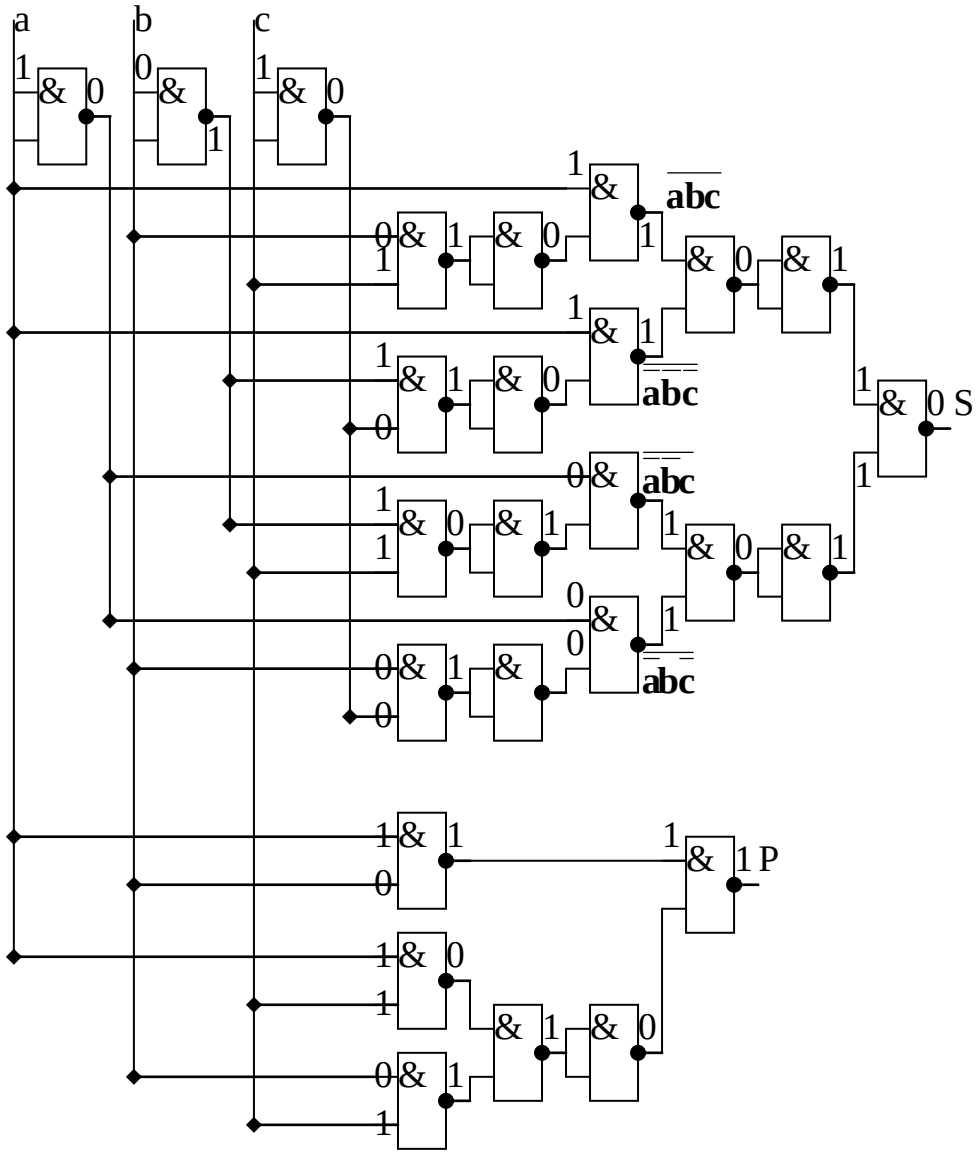
$$P_{МКНФ} = \overline{\overline{(a+b)}\cdot\overline{(a+c)}\cdot\overline{(b+c)}} = \overline{\bar{a}\bar{b}}\cdot\overline{\bar{a}\bar{c}}\cdot\overline{\bar{b}\bar{c}}.$$

Для реализации $P_{МДНФ}$ необходимо 6 двухвходовых элементов «и-не», а для $P_{МКНФ}$ – 7 (с учётом того, что $\bar{a}, \bar{b}, \bar{c}$ уже получены в $S_{МДНФ}$).

Для построения части схемы, реализующей P, выберем $P_{МДНФ}$.

4. Построим функциональную схему сумматора используя выражения, выбранные на третьем шаге алгоритма.

$$S_{МДНФ} == \overline{(\bar{a}\bar{b}c)}\cdot\overline{(\bar{a}b\bar{c})}\cdot\overline{(a\bar{b}\bar{c})}\cdot\overline{(abc)};\ P_{МДНФ} = \overline{\overline{ab}\cdot\overline{ac}\cdot\overline{bc}}.$$



В данном случае построение частей схемы, реализующих S и P проводилось независимо друг от друга.

5. На этапе схемотехнической коррекции ограничимся лишь оценкой выполнения требований по коэффициентам объединения $К_о$ и разветвления $К_р$. $К_0$ был задан равным двум, а $К_р$ для логических схем равен $5 \div 20$. Зададим $К_р$=5. Схема показывает, что для всех элементов $К_0$=2, а $К_р$<4 и, следовательно, требования по $К_о$ и $К_р$ удовлетворены.

6. Проведём тестирование схемы сумматора только на одном наборе аргументов. Пусть a=1, b=0, c=1. По ТИ этому набору соответствуют S=0 и P=1. Расположим на всех входах и выходах элементов схемы соответствующие логические значения (0 или 1) и убедимся в правильности реализации функции S и P на данном наборе аргументов (см. схему). Если выполнить аналогичные действия для всех наборов аргументов их ТИ, то тестирование будет полным.

Обратим внимание на то, что полученная схема, в соответствии с заданием, содержит только двухвходовые элементы «и-не».

Задание 67. Синтезировать комбинационную схему, закон функционирования которой задан функцией F_4 (таблица 4), используя логические элементы «и-не».

Задание 68. Синтезировать комбинационную схему полного одноразрядного сумматора на двухвходовых логических элементах «или-не» и протестировать её на наборе a=1, b=1, c=0.

Задание 69. Синтезировать комбинационную схему, реализующую функцию F_1, заданную в таблице 5. Использовать двухвходовые логические элементы «или-не». Провести тестирование схемы при x_1=0, x_2=1, x_3=1, x_4=0.

таблица 5

x_1	x_2	x_3	x_4	F_1	F_2	F_3
0	0	0	0	1	0	0
0	0	0	1	1	0	1
0	0	1	0	1	1	-
0	0	1	1	0	1	0
0	1	0	0	0	1	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	1
1	0	0	0	1	0	-
1	0	0	1	1	0	-
1	0	1	0	0	1	0
1	0	1	1	0	1	0
1	1	0	0	1	0	1
1	1	0	1	0	1	1
1	1	1	0	0	0	0
1	1	1	1	0	1	1

Задание 70. Синтезировать комбинационную схему, реализующую функции F_2 и F_5, заданные в таблицах 5 и 6. Использовать двухвходовые логические элементы «и-не». Провести тестирование схем при x_1=0, x_2=1, x_3=1, x_4=1.

таблица 6

x_1	x_2	x_3	F_1	F_2	F_3
0	0	0	0	1	0
0	0	1	1	0	1
0	1	0	1	1	-
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	0	0	-
1	1	0	1	1	0
1	1	1	1	1	1

Задание 71. Синтезировать комбинационную схему, реализующую функции F_3 и F_6, заданные в таблицах 5 и 6. Использовать логические элементы: «не» и «2и». Провести тестирование при $x_1=x_3=x_5=0$, $x_2=x_4=1$.

1.4.4. Односторонние булевы уравнения

Рассмотрим метод решения односторонних булевых уравнений с одним и двумя неизвестными. Уравнения с одним неизвестным:

$$F_1(x_1,...,x_n)\cdot P(x_1,...,x_n)+\overline{F_1}(x_1,...,x_n)\cdot \overline{P}(x_1,...,x_n)=F_2(x_1,...,x_n).$$

Уравнения с двумя неизвестными:

$$F_1(x_1,...,x_n)\cdot Q(x_1,...,x_n)+\overline{F_1}(x_1,...,x_n)\cdot R(x_1,...,x_n)=F_2(x_1,...,x_n),$$

где $F_1(\ldots)$ и $F_2(\ldots)$ – известные (заданные) функции, а P(…), Q(…), R(…) – неизвестные функции.

Решение этих уравнений будем искать методом подбора значений P, Q, R на всех наборах F_1 и F_2. Таких наборов 4. Составим таблицу решений.

$$F_1\cdot P+\overline{F_1}\cdot\overline{P}=F_2;\ F_1\cdot Q+\overline{F_1}\cdot R=F_2.$$

таб. решений

F_1	F_2	P	Q	R
0	0	1	-	0
0	1	0	-	1
1	0	0	0	-
1	1	1	1	-

Для получения решений в виде $P=\varphi(x_1,...,x_n)$ для уравнений с одним неизвестным и $Q=\varphi_1(x_1,...,x_n)$, $R=\varphi_2(x_1,...,x_n)$ для уравнения с двумя неизвестными, необходимо сначала перейти от формы представления исходных функций F_1 и F_2 к форме представления функций P, Q, R (для этого используется таблица решений), а затем выполнить минимизацию этих функций.

Пример. Решить булевы уравнения $F_1\cdot P+\overline{F_1}\cdot\overline{P}=F_2$ и $F_1\cdot Q+\overline{F_1}\cdot R=F_2$ при F_1 и F_2 заданных в строчной форме: $F_1=\sum_4 0110110000111001$, $F_2=\sum_4 1001011001101001$. Выполнить переход от F_1 и F_2 к P, Q, R. Используем таблицу решений односторонних булевых уравнений.

$$F_1=\sum_4 0110110000111001$$

$$F_2=\sum_4 1001011001101001$$

$$P=\sum_4 0000010110101111$$

$$Q=\sum_4 -00-01----101--1$$

$$R=\sum_4 1--1--1001---00-$$

Перейдём от представления P, Q и R к строчной форме к представлению картами Карно и выполним минимизацию:

карта Карно для P

X_1x_2 \ x_3x_4	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	1	1	1	1
10	1	0	0	1

карта Карно для Q

x_1x_2 \ x_3x_4	00	01	11	10
00	-	0	-	0
01	0	1	-	-
11	1	-	1	-
10	-	-	0	1

Карта Карно для R

x_1x_2 \ x_3x_4	00	01	11	10
00	1	-	1	-
01	-	-	0	1
11	-	0	-	0
10	0	1	-	-

Проведя минимизацию P, Q и R получим:

$P_{МДНФ} = x_2x_4 + x_1\overline{x_4}$, $Q_{МДНФ} = x_2x_4 + x_1\overline{x_4}$, $R_{МДНФ} = \overline{x_2}x_4 + \overline{x_1}\,\overline{x_4}$.

Задание 72. Решить односторонние булевы уравнения с одним и двумя неизвестными, если известные функции F_4 и F_5 заданы ТИ (таблица 6).

Задание 73. Решить односторонние булевы уравнения с одним и двумя неизвестными, если известные функции F_1 и F_2 заданы ТИ (таблица 5).

1.4.5. Методы синтеза комбинационных схем с многими выходами.

Послу раздельной минимизации булевых функций, описывающих многовыходную схему, применяются различные методы совместной обработки функций.

Рассмотрим два из них:

- метод учёта повторяющихся членов,
- метод последовательного наращивания.

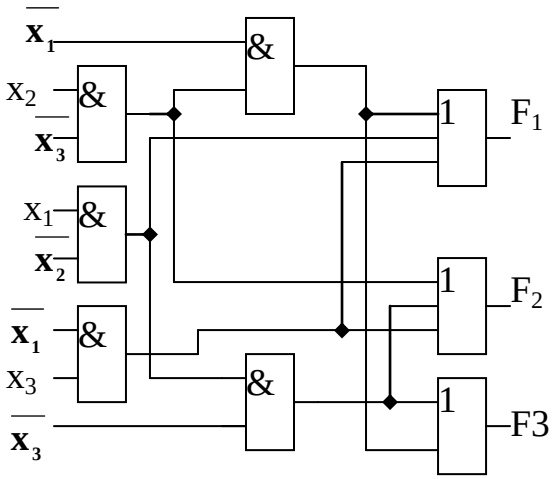
Метод учёта повторяющихся членов.

В данном методе находятся общие для формул нескольких функций члены(функции, отдельные дизъюнкции, конъюнкции и их части). При построении общей комбинационной схемы, реализующей несколько логических функций, фрагменты, соответствующие общим (повторяющимся) членам, формируются только один раз и используются для получения любой функции, куда они входят.

Пример. Синтезировать комбинационную схему, реализующую функции:

$F_1 = x_1\overline{x_2} + \overline{x_1}x_2\overline{x_3} + \overline{x_1}x_3$, $F_2 = x_1\overline{x_2}\,\overline{x_3} + \overline{x_1}x_3 + x_2\overline{x_3}$, $F_3 = \overline{x_1}x_2\overline{x_3} + x_1\overline{x_2}\,\overline{x_3}$.

Общими членами в этих выражениях являются: $x_1\overline{x_2}, \overline{x_1}x_2, x_2\overline{x_3}, x_1\overline{x_3}, x_1\overline{x_2}\,\overline{x_3}, \overline{x_1}x_2\overline{x_3}$.



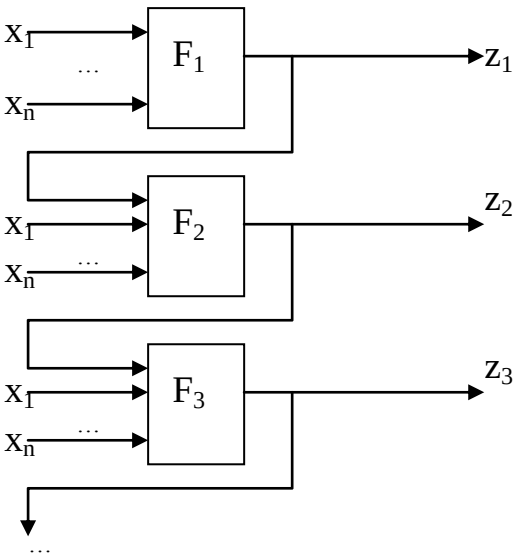
Задание 74. Синтезировать комбинационную схему, реализующую функции:
$F_1 = AB\overline{C} + \overline{AB}C + BC, F_2 = \overline{A}B + A\overline{C} + A\overline{B}C, F_3 = \overline{A}\overline{B}\overline{C} + A\overline{B}C.$

Метод последовательного наращивания.

Логическая схема строится сначала для какого-либо одного выхода, затем для другого, третьего и т. д.

При построении схемы последующего выхода, выходные сигналы уже построенной схемы используются на ряду с общими выходными сигналами схемы. Следовательно, функция этого выхода может быть представлена в виде функции от (n+1) переменной (n – входных сигналов плюс выход предыдущей схемы).

$$z_i = F_i(x_1,...,x_n) = F_i'(\underbrace{F_{(i-1)}, x_1,...,x_n}_{(n+1)})$$



Реализуя данный метод нужно получить специальный блок, преобразующий функцию F_{i-1} в функцию F_i. Для этого проведём разложение F_i по новой перемноженной F_i': $F_i'(F_{(i-1)}, x_1,...,x_n) = F_{(i-1)}F'_{1_i} + \overline{F_{(i-1)}}F'_{0_i} = F_1(x_1,...,x_n)$.

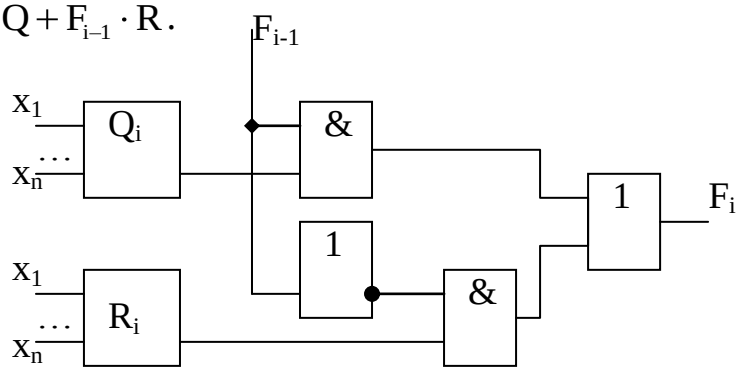
Таким образом, получаем обычное булево уравнение с двумя неизвестными $F_{(i-1)} \cdot Q_i + \overline{F_{(i-1)}} \cdot R_i = F_i$. Следовательно, при построении многовыходной схемы методом последовательного наращивания необходимо найти решение (n-1) булева уравнения и определить все Q_i и R_i.

Рекомендации. За исходную функцию F_1 можно брать любую из заданных, но, как правило, лучшие результаты получаются, если взять ту, у которой проще минимальная форма.

Если одна из исходных функций не доопределена, а вторая определена полностью, то неопределённые значения группируются так, чтобы это было наиболее выгодно при минимализации Q_i и R_i.

Структурная схема, реализующая функцию F'$_i$ может быть представлена следующим образом:

$$F_i = F_{i-1} \cdot Q + \overline{F_{i-1}} \cdot R.$$



Пример. Синтезировать комбинационную схему полного одноразрядного двоичного сумматора методом последовательного наращивания.

$P = AB + AC + BC$, $S = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC$. В качестве F_1 выбираем P, как функцию с простой минимальной формой. $S = P \cdot Q + \overline{P} \cdot R$.

AB

C	00	01	11	10
0	0	0	1	0
1	0	1	1	1

для P

AB

C	00	01	11	10
0	0	1	0	1
1	1	0	1	0

для S

AB

C	00	01	11	10
0	-	-	0	-
1	-	0	1	0

для Q

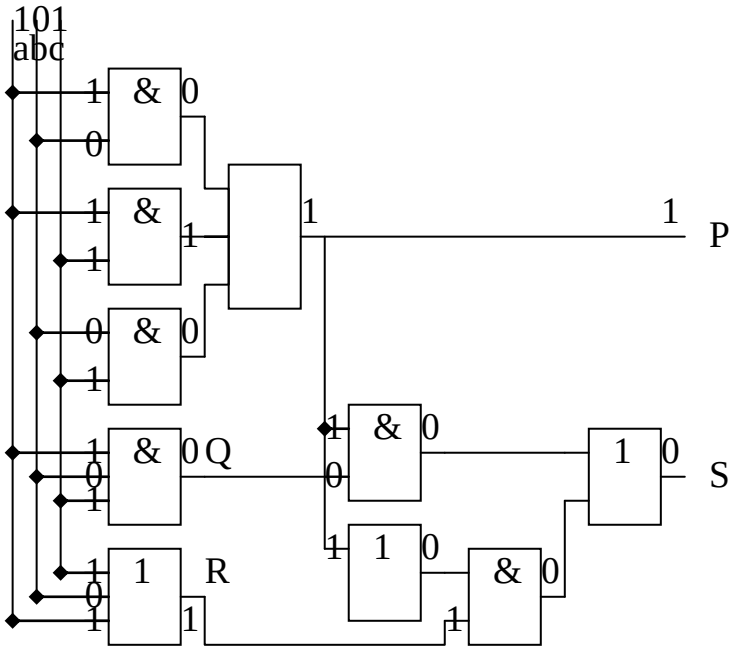
AB

C	00	01	11	10
0	0	1	-	1
1	1	-	-	-

для R

$Q = ABC, R = A + B + C.$

Проведём тестирование построенной схемы на наборе аргументов A=1, B=0, C=1. Этому набору должны соответствовать значения S=0 и P=1. Проставленные логические значения на входах и выходах элементов схемы показывают работоспособность схемы (на данном наборе).



При синтезе многовыходных схем методом последовательного наращивания удаётся, обычно, построить схему с использованием меньшего количества логических элементов, чем при синтезе с независимым построением частей схемы, реализующих функции выходов. Но при этом получается некоторый проигрыш в быстродействии схемы.

Задание 75. Реализовать рассмотренную схему только на двухвходовых элементах «и-не» и, проведя сравнение полученной реализации со схемой, определить выигрыш (проигрыш) в количестве элементов схемы и её быстродействии.

Задание 76. Используя условия задания 70 синтезировать двухвыходную комбинационную схему методом последовательного наращивания.

Задание 77 Используя условия задания 71 синтезировать двухвыходную комбинационную схему методом последовательного наращивания.

1.5. Синтез и анализ конечных автоматов.

Определение. Если совокупность выходных сигналов (выходного слова) Y(t) зависит, как от совокупности входных сигналов (входного слова) X(t), так и от внутренних состояний S(t-1), то такой преобразователь информации называется конечным автоматом (автоматом с памятью).

Для описания функционирования конечного автомата (КА) задаются:

$X(t)=\{x_1,...,x_i,...,x_n\}$ - множество входных дискретных сигналов, $x_i \in \{0,1\}, i=\overline{1,n}$;

$Y(t)=\{y_1,\ldots,y_j,\ldots,y_m\}$ – множество выходных дискретных сигналов, $y_j \in \{0,1\}, j=\overline{1,m}$;

$S(t)=\{s_1,\ldots,s_k,\ldots s_r\}$ – множество внутренних дискретных состояний, $s_k \in \{0,1\}, k=\overline{1,r}$;

s_0 – начальное состояние автомата;

FP – функция переходов, позволяющая определить новое внутреннее состояние автомата, если известно предыдущее внутреннее состояние и состояние входных сигналов в данных момент времени S(t+1)=FP[S(t),X(t)];

FV – функция выходов, позволяющая определить состояние выходных сигналов автомата, если известно внутреннее состояние и состояние входных сигналов Y(t)=FV[S(t),X(t)].

Таким образом, функционирования КА можно представить так:

$$\begin{cases} S(t+1)=FP[S(t),X(t)], \\ Y(t)=FV[S(t),X(t)], \text{ где } t=0,1,2\ldots \\ S(t=0)=s_0 \end{cases}$$

Автомат Мили

или так:

$$\begin{cases} S(t+1)=FP[S(t),X(t)], \\ Y(t)=FV[S(t)], \text{ где } t=0,1,2\ldots \\ S(t=0)=s_0 \end{cases}$$

Автомат Мура

Конечные автоматы делятся на 2 части:

Абстрактную – описание полноты FV, X, Y, S автомата, работоспособности

Структурную – отвечает на вопрос как построить КА.

При описании работы конечного автомата используют таблицы состояний или графы (диаграммы состояний).

таблицы состояний:

таблица переходов

	s_0	s_1	…	s_r
x_1	s_3	s_0	…	…
x_2	s_1	s_3	…	…
…	…	…	…	…
x_n	…	…	…	…

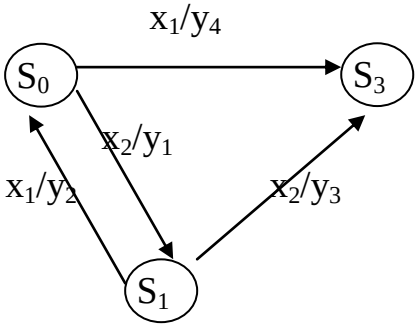
таблица выходов

	s_0	s_1	…	s_r
x_1	y_4	y_2	…	…
x_2	y_1	y_3	…	…
…	…	…	…	…
x_n	…	…	…	…

совмещённая таблица

	$s_0(t)$	$s_1(t)$	…
$x_1(t)$	$s_3(t+1) / y_4(t)$	$s_0(t+1) / y_2(t)$	…
$x_2(t)$	$s_1(t+1) / y_1(t)$	$s_3(t+1) / y_3(t)$	…
…	…	…	…

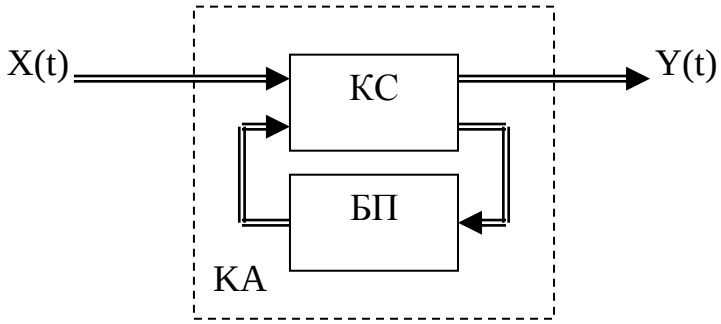
(фрагмент)



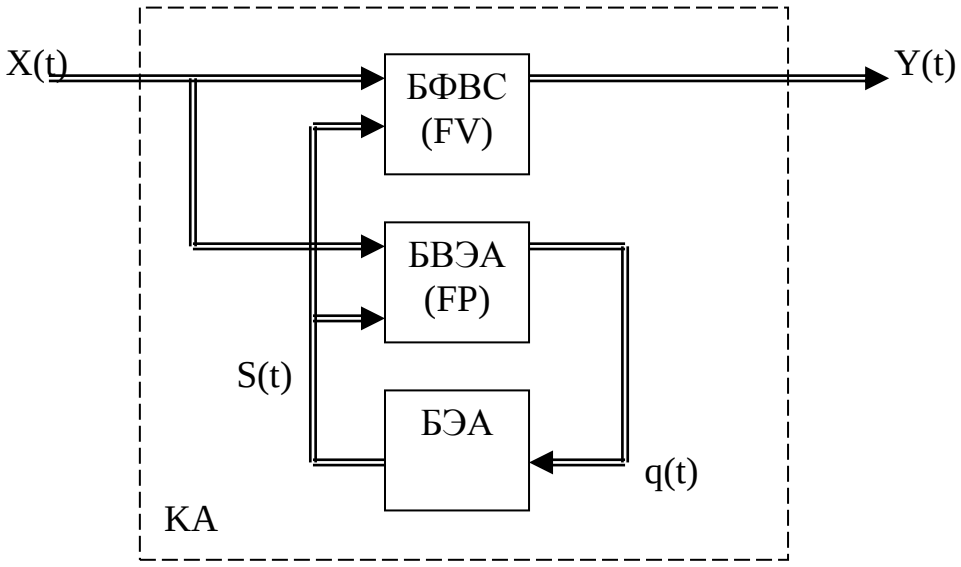
Задание КА графом (соответствует содержимому таблиц состояний).

Различают два основных типа КА: автомат Мили м автомат Мура. Различие между ними заключается в том, что в автомате Мура состояние выходных сигналов не зависит от состояния входных сигналов, а определяется лишь внутренним состоянием, т. е. Y(t)=FV[S(t)].

Структурная схема КА.



или несколько подробнее



КС – комбинационная схема,
БП – блок памяти,

БФВС – блок формирования выходных сигналов (комбинационная схема, реализующая FV),

БВЭА – блок возбуждения элементарных автоматов (комбинационная схема, реализующая FP),

БЭА – блок элементарных автоматов (память КА),

q – функция возбуждения.

1.5.1. Элементарные конечные автоматы и их техническая реализация.

Функция памяти КА реализуется на элементарных автоматах. Элементарный автомат (ЭА) – это автомат Мура, имеющий два и только два различных состояния, имеющий 1 или 2 входа, при подаче сигналов на которые возможен переход из одного состояния в другое.

Для построения памяти КА используются, в основном, четыре типа ЭА: два типа одновходовых и два двухвходовых.

Одновходовые ЭА:

$Q_1(t+1)=q(t)$ – ЭА D-типа,

$Q_2(t+1)=q(t)\oplus Q(t)$ - ЭА T-типа.

q(t)	0 0 1 1
Q(t)	0 1 0 1
$Q_1(t+1)$	0 0 1 1
$Q_2(t+1)$	0 1 1 0

Двухвходовые ЭА:

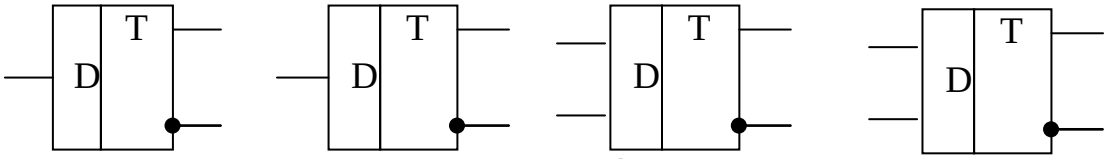
$Q_3(t+1)=\overline{q_0(t)}\cdot Q(t)+q_1(t)$ - ЭА RS-типа,

$Q_4(t+1)=\overline{q_0(t)}\cdot Q(t)+q_1(t)\cdot\overline{Q(t)}$ - ЭА JK-типа.

$q_0(t)$	0 0 0 0 1 1 1 1
$q_1(t)$	0 0 1 1 0 0 1 1
Q(t)	0 1 0 1 0 1 0 1
$Q_3(t+1)$	0 1 1 1 0 0 - -
$Q_4(t+1)$	0 1 1 1 0 0 1 0

Техническая реализация ЭА:

Автомат D-типа Автомат T-типа Автомат RS-типа Автомат JK-типа



D - триггер T - триггер RS - триггер JK - триггер

$q_1(t)=q_S(t)$, $q_1(t)=q_J(t)$, $q_0(t)=q_R(t)$, $q_0(t)=q_K(t)$.

1.5.2. Алгоритм структурного синтеза конечного автомата.

1. Задать закон функционирования конечного автомата.

2. Определить n – требуемое количество ЭА. $n=\lceil \log_2(r+1) \rceil$, где r+1 – количество внутренних состояний ЭА, $\lceil \ldots \rceil$ - ближайшее большее целое.

3. Выбрать тип ЭА по таблице. Для этого предварительно построить таблицу для требуемой функции возбуждения.

4. Построить функциональную схему БЭА.

$Q(t) \to Q(t+1)$	тип ЭА					
	D	T	RS		JK	
	$q_D(t)$	$q_T(t)$	$q_S(t)$	$q_R(t)$	$q_J(t)$	$q_K(t)$
$0 \to 0$	0	0	0	-	0	-
$0 \to 1$	1	1	1	0	1	-
$1 \to 0$	0	1	0	1	-	1
$1 \to 1$	1	0	-	0	-	0

5. Провести синтез комбинационной схемы БВЭА.

6. Провести синтез комбинационной схемы БФВС.

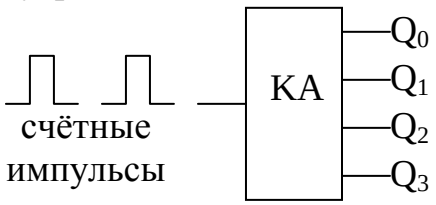
7. Составить из БЭА, БВЭА и БФВС схему конечного автомата.

8. Провести тестирование полученной схемы на соответствие закону функционирования.

1.5.3. Пример синтеза конечного автомата – двоично-десятичного счётчика.

Пример. Синтезировать схему двоично-десятичного счётчика.

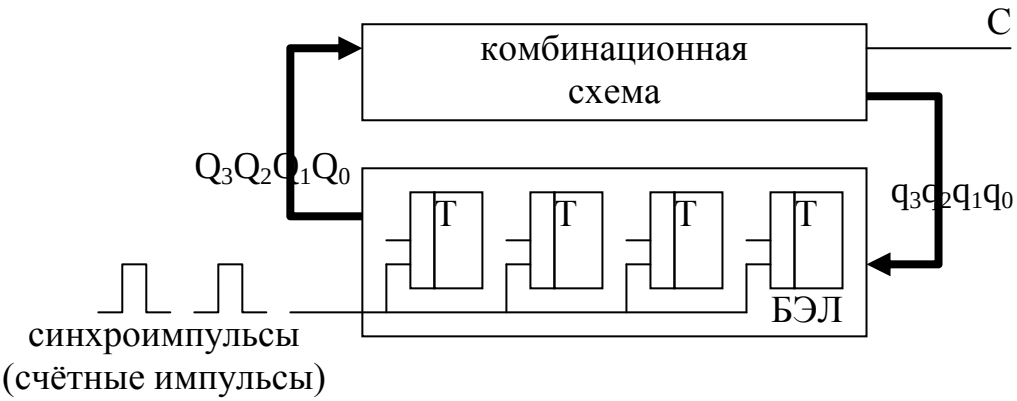
Зададим закон функционирования двоично-десятичного счётчика. Этот КА имеет 10 внутренних состояний:



	0	1	2	3	4	5	6	7	8	9
Q_0	0	1	0	1	0	1	0	1	0	1
Q_1	0	0	1	1	0	0	1	1	0	0
Q_2	0	0	0	0	1	1	1	1	0	0
Q_3	0	0	0	0	0	0	0	0	1	1

Так как $n=\lceil \log_2 10 \rceil=4$, то для запоминания всех 10 состояний КА потребуется четыре ЭА.

Выберем Тим ЭА. Пусть это будет ЭА Т-типа. Тогда структурная схема синтезируемого КА будет иметь вид:



Выполним синтез БВЭА. Построим либо совмещённую таблицу состояний, либо диаграмму состояний:

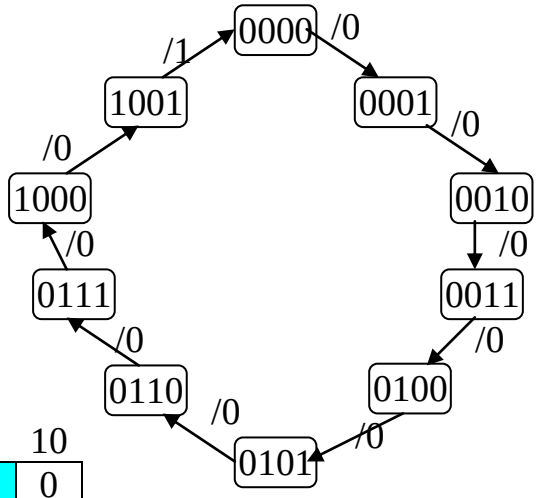
Таблица состояний

текущее состояние				следующее состояние				выходы комбинационной схемы					
Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0	q_3	q_2	q_1	q_0	/	c
0	0	0	0	0	0	0	1	0	0	0	1	/	0
0	0	0	1	0	0	1	0	0	0	1	1	/	0
0	0	1	0	0	0	1	1	0	0	0	1	/	0
0	0	1	1	0	1	0	0	0	1	1	1	/	0
0	1	0	0	0	1	0	1	0	0	0	1	/	0
0	1	0	1	0	1	1	0	0	0	1	1	/	0
0	1	1	0	0	1	1	1	0	0	0	1	/	0
0	1	1	1	1	0	0	0	1	1	1	1	/	0
1	0	0	0	1	0	0	1	0	0	0	1	/	0
1	0	0	1	0	0	0	0	1	0	0	1	/	1

Над дробью в диаграмме ничего нет, т. к. входные сигналы отсутствуют в данной схеме. Под дробью значение выхода С.

Построим карты Карно для выходных сигналов БВЭА (т. е. для сигналов q_0, q_1, q_2, q_3).

Диаграмма состояний



КК для q_3

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	-	-	-	-
10	0	1	-	-

КК для q_2

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	-	-	-	-
10	0	0	-	-

КК для q_1

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	-	-	-	-
10	0	0	-	-

КК для q_0

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	-	-	-	-
10	1	1	-	-

$$q_3 = Q_3Q_0 \vee Q_2Q_1Q_0, q_2 = Q_1Q_0, q_1 = \overline{Q_3}Q_0, q_0 = 1.$$

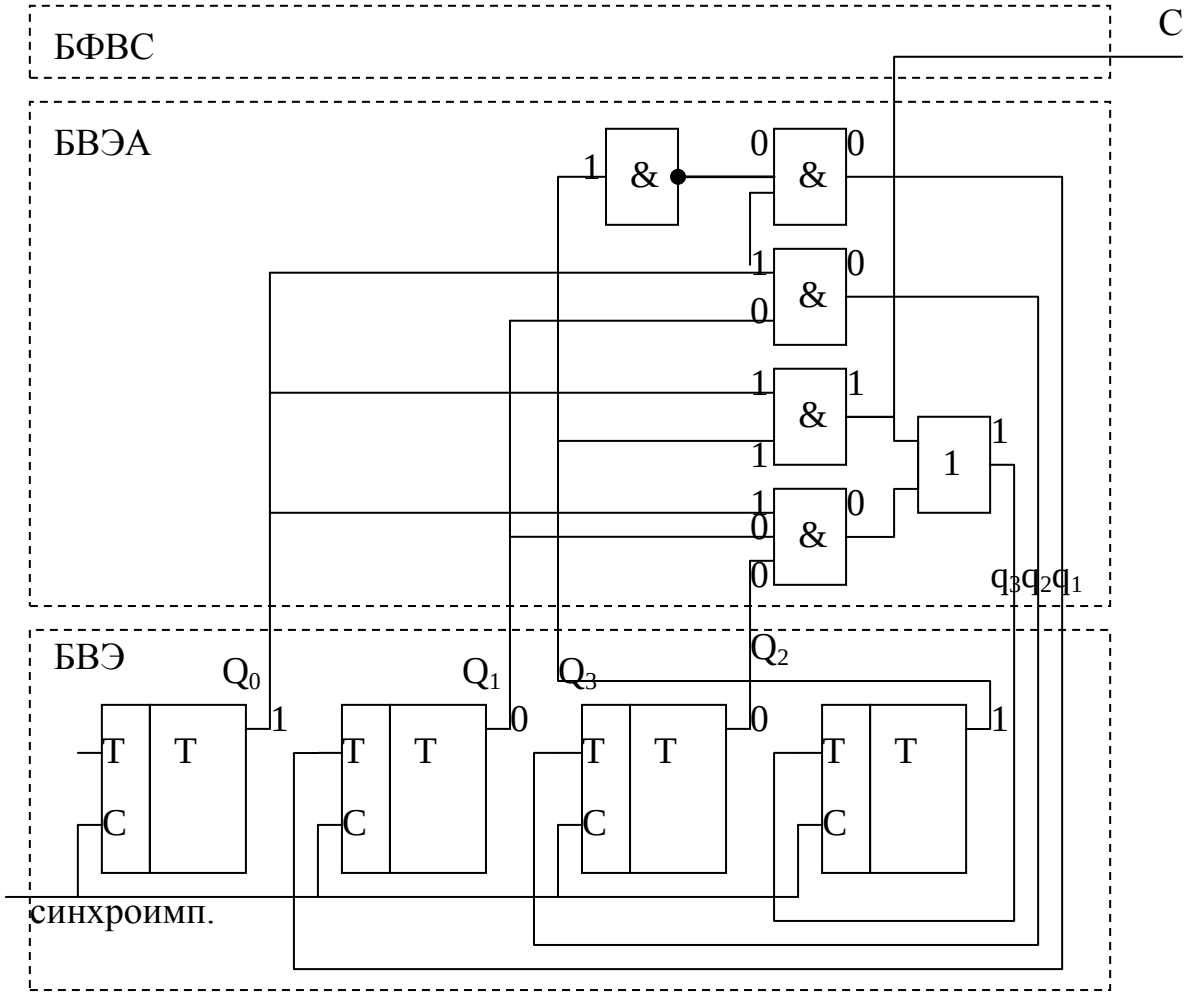
Проведём синтез БФВС. Построим карту Карно для выходных сигналов комбинационной схемы БФВС. (т. е. для сигнала С).

КК для С

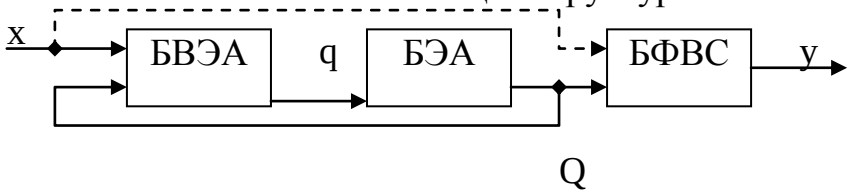
Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	-	-	-	-
10	0	1	-	-

$C=Q_3Q_0$.

Составим из БЭА, БВЭА и БФВС схему конечного автомата:



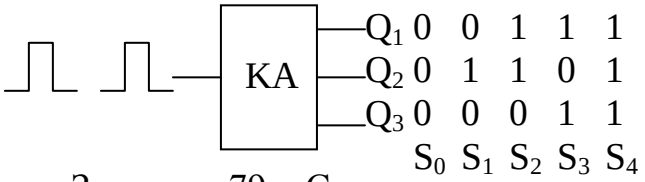
Вспомним соответствие с общей структурой КА:



------ - цепь в данном случае отсутствует

Проверим работоспособность схемы на примере перехода из состояния 9 в 0 и формирования С. В этом состоянии Q_3=1, Q_2=0, Q_1=0, Q_0=1 (см. 1 и 0 в схеме). При этом сформируются сигналы возбуждения: q_3=1, q_2=0, q_1=0, q_0=1 и, следовательно, следующим состоянием КА будет: $Q_3=Q_2=Q_1=Q_0=0$ и С=1, что соответствует правильной работе двоично-десятичного счётчика.

Задание 78. Синтезировать схему КА, который, при подаче на вход счётных импульсов, переходит последовательно в состояния $S_0 \div S_4$ в соответствии с рисунком.



Задание 79. Синтезировать схему КА, который на входные счётные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 2, 4, 5, 7, 9. В качестве ЭА выбрать Т-триггеры.

Задание 80. Синтезировать схему КА в соответствии с заданием №79, но с использованием в качестве ЭА RS-триггеров.

2. Организация ЭВМ.

2.1. Типовые структурные элементы цифровой техники.

2.1.1. Основные характеристики и классификация цифровых элементов вычислительной техники.

Типовыми структурными элементами называются наименьшие функциональные части из которых состоят устройства цифровой техники.

Это:

- элементы реализующие логические функции,
- элементы преобразующие информацию,
- элементы запоминающие информацию,
- вспомогательные элементы (генерирующие, формирующие и усиливающие сигналы).

По способу представления информации структурные элементы делятся на:

- импульсные («0» - отсутствие импульса напряжения, «1» - наличие импульса напряжения)
- потенциальные («0» - один уровень напряжения, «1» - другой уровень напряжения)

Условные обозначения схем.

0123456 – номера элементов обозначения

0 – буква «К» или « »

1+2 – обозначение серии

3 – буква, указывающая на функциональный класс

4 – буква, указывающая на группу данного функционального класса

5 – число, указывающее номер разработки данной микросхемы в серии

6 – буква, цветная точка или др. – маркировка по разбросу параметров, предельным эксплутационным режимам и другим признакам, вызванным отклонением технологического процесса.

Микросхема дана в технических условиях на серию.

К – широкое потребление

серия – ряд функционально различных микросхем объединённых по технологии, параметрам, конструктивному оформлению.

Обозначение серии:

1 элемент – цифра, указывающая на технологический признак

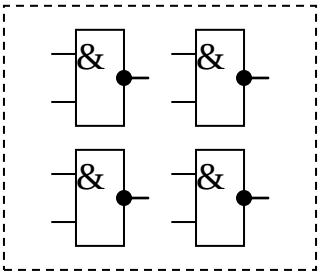
1, 5, 7 – полупроводниковые микросхемы

2, 4, 6 – гибридные микросхемы

3 – плёночные микросхемы

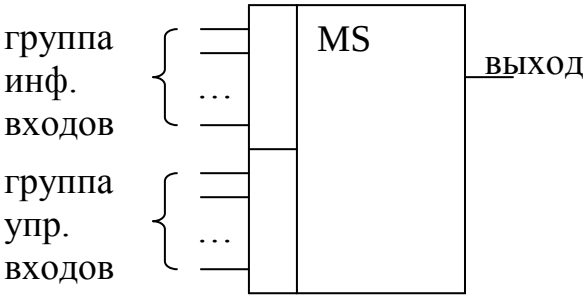
например:

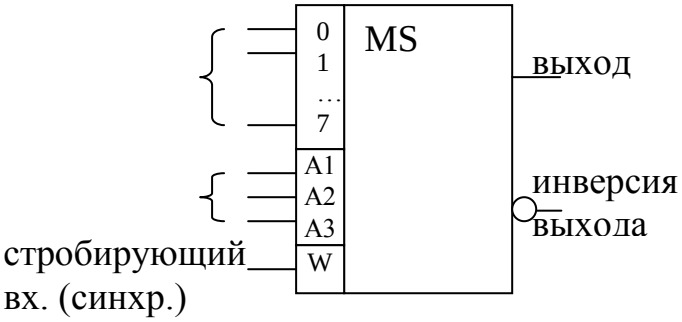
К155ЛА3



2.1.2. Мультиплексоры и дешифраторы.

I. Мультиплексор – схема, передающая сигналы с одной из нескольких входных линий в выходную линию



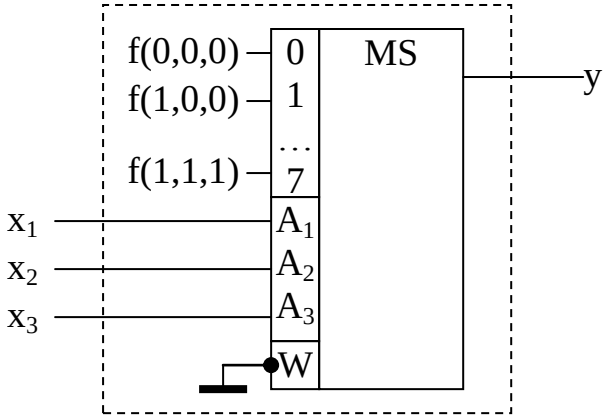


При подаче на управляющие входы двоичного кода и на вход W нуля, к выходу подключается только тот вход, номер которого в десятичном изображении совпадает со значением двоичного кода на управляющих входах. Информация на других информационных входах не влияет.

Реализация логических функций на мультиплексорах. (любых КС).

Пусть задана функция трёх переменных на мультиплексорах. Составим ТИ.

x_1	x_2	x_3	y
0	0	0	f(0,0,0)
1	0	0	f(1,0,0)
…	…	…	…
1	1	1	f(1,1,1)

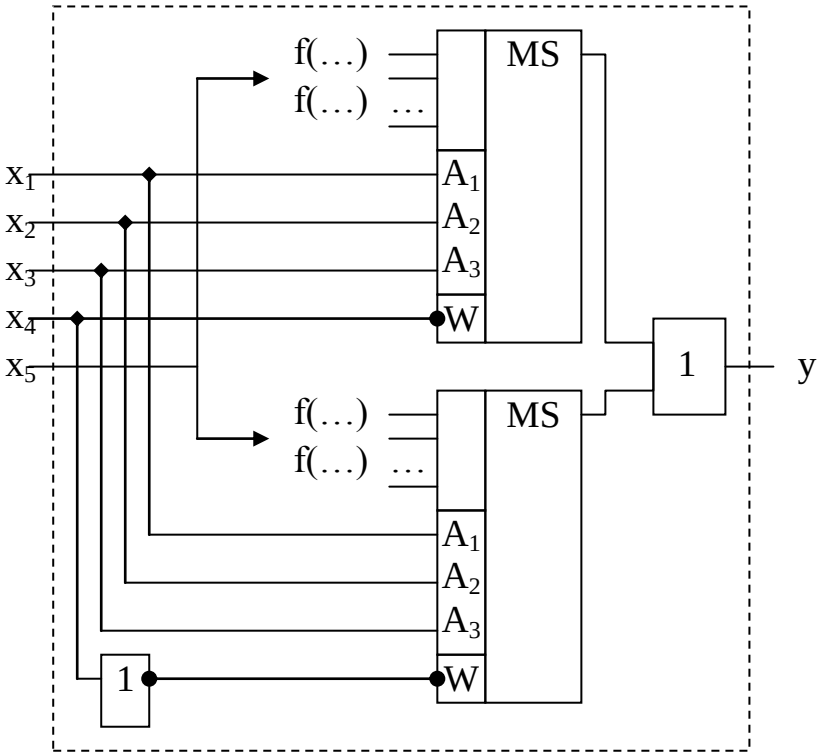


Для функции 4-х переменных:

x_1	x_2	x_3	$f(x_4)$
0	0	0	$f(0,0,0,x_4)$

Для функции 5-ти переменных:

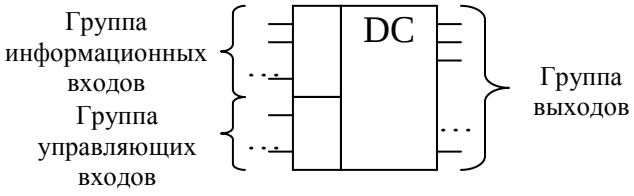
x_1	x_2	x_3	x_4	$y(x_5)$
0	0	0	0	$f(0,0,0,0,x_5)$
0	0	0	1	$f(0,0,0,1,x_5)$



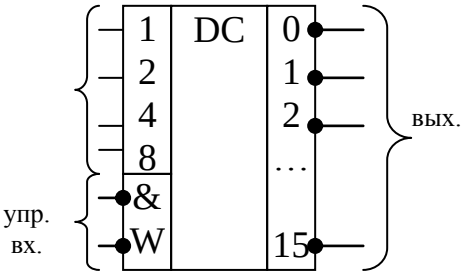
II. Дешифратор – КС преобразующая код, подаваемый на входы, в сигнал на одном из выходов.

n – входов, 2^n – выходов

функциональное обозначение



К155ИД3



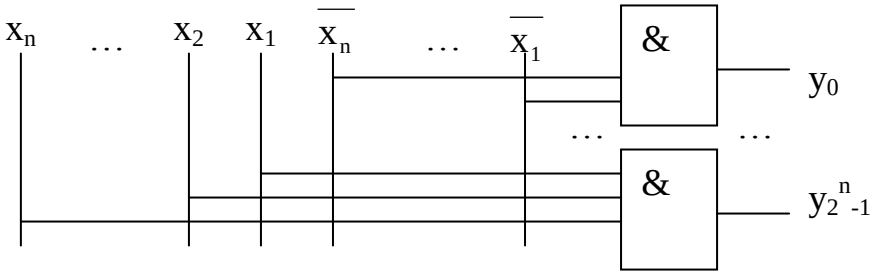
$$y_0 = \overline{x_n}\,\overline{x_{n-1}}\ldots\overline{x_2}\,\overline{x_1}$$

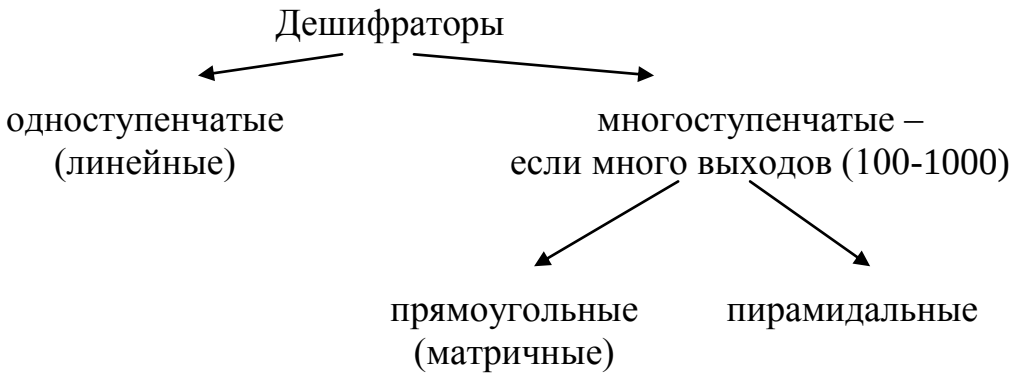
$$y_1 = \overline{x_n}\,\overline{x_{n-1}}\ldots\overline{x_2}\,x_1$$

.....

$$y_{2^n-1} = x_n x_{n-1}\ldots x_2 x_1$$

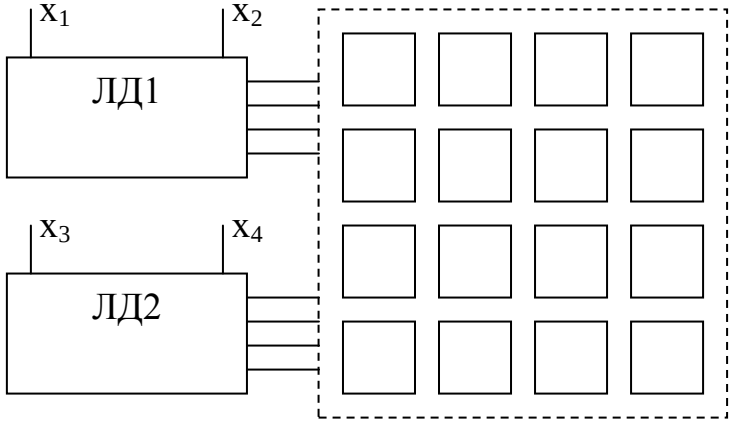
Следовательно:



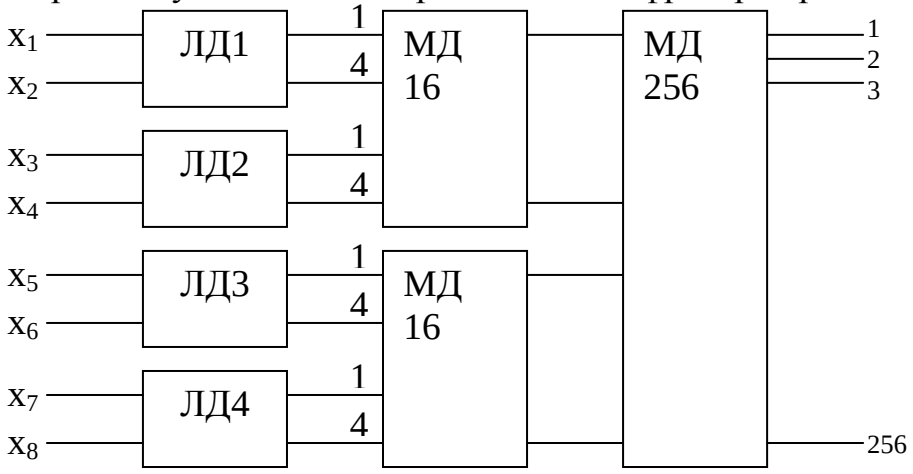


Прямоугольные дешифраторы:
1 ступень – 2 линейных дешифратора
2 ступень – матричная схема

Синтез матричного дешифратора при n=4, n^4=16 выходов.

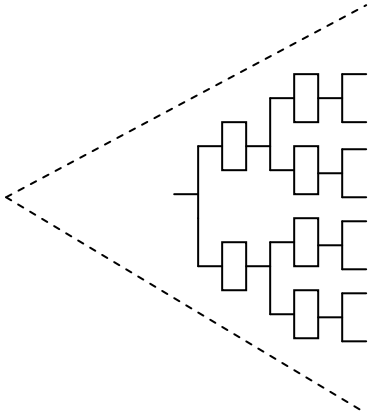


Пример 3-х ступенчатого матричного дешифратора при n=8:



плюс – меньше аппаратуры
минус – быстродействие

Пирамидальные дешифраторы:
элемент i-ой ступени нагружен
только на 2 элемента i+1-ой ступени
Промежуточное значение между
линейным и матричным.



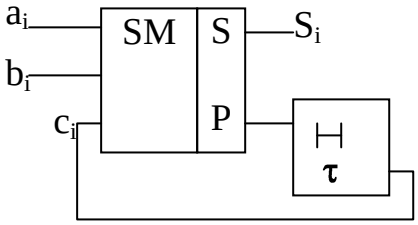
2.1.3. Сумматоры.

- КС для выполнения арифметических и логических операций над числами

Замечание: операция сложения в ЭВМ сложнее операции суммирования, т. к. учитывает: знаки чисел, выравниваются порядки, проводится нормализация и др.

одноразрядные или многоразрядные.

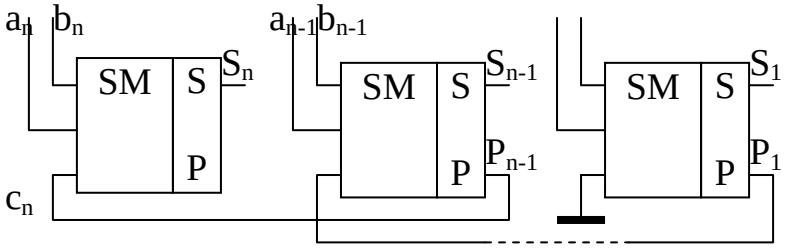
суммирование: последовательное, параллельное, параллельно последовательное (по группам).



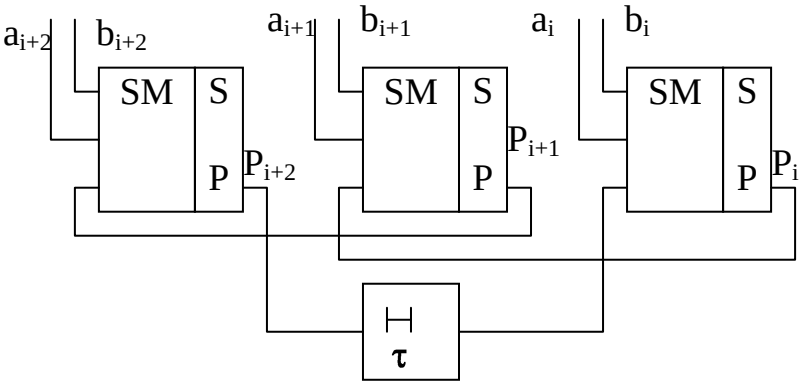
I. Сложение многоразрядных чисел.

Последовательный сумматор:
слова A и B поступают с младших разрядов
τ - элемент задержки на 1 такт

Параллельный сумматор:



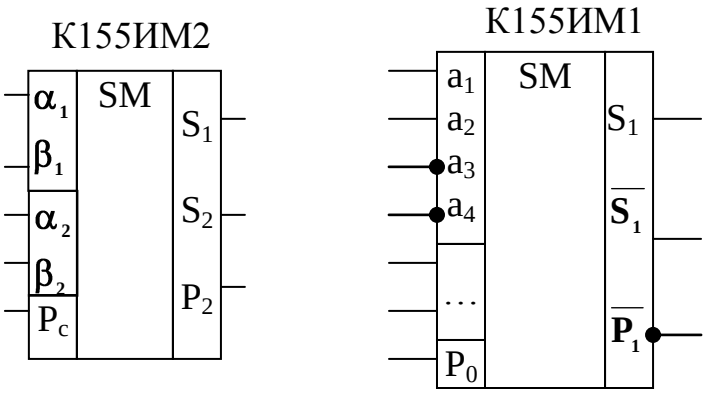
Параллельно последовательный сумматор:



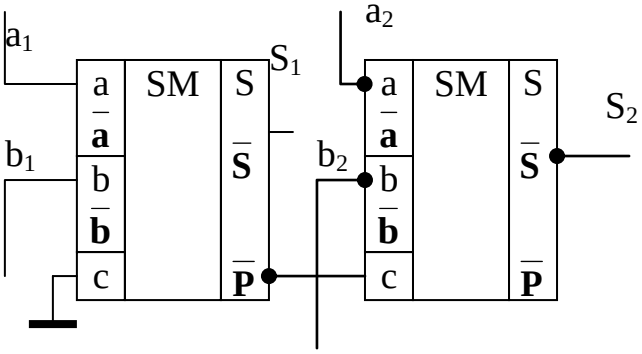
Пример:

в 155 серии

полный 2-х разрядный параллельный сумматор



(реализация многоразрядного сумматора на ИМ1)

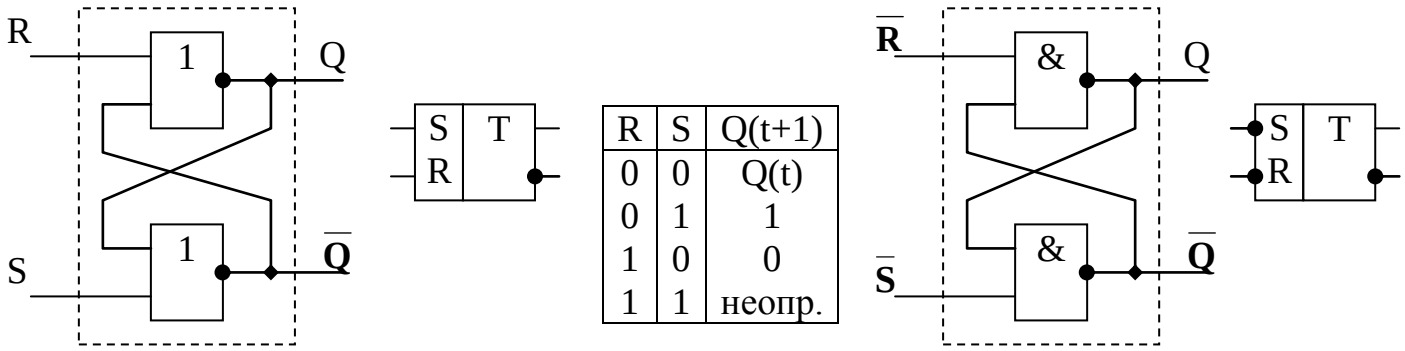


2.1.4. Триггеры.

Это КА с двумя устойчивыми состояниями. Хранит 1 бит

- регенеративная схема (собственно триггер)
- схема управления

Триггер RS-типа

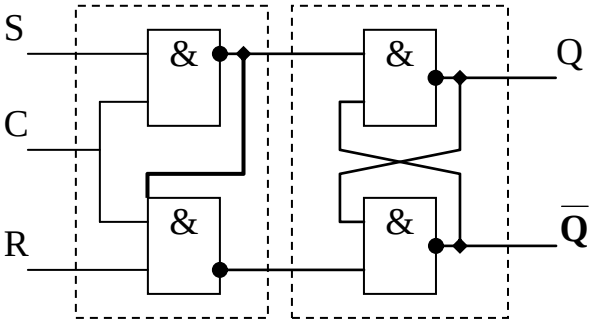


R	S	Q(t+1)
0	0	Q(t)
0	1	1
1	0	0
1	1	неопр.

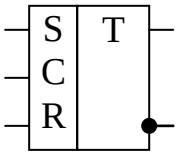
CRS-триггер

(синхронизируемый RS-триггер).

S и R – информационные входы, C – вход синхронизации (clock – времязадающий)



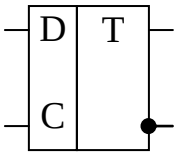
C	R	S	Q(t+1)
1	0	0	Q(t)
	0	1	1
	1	0	0
	1	1	запрещ.
0	Q(t)		



D-триггер (“delay” – задержка)

Исключается возможность одновременной подачи 2-х единиц на S и R входы.

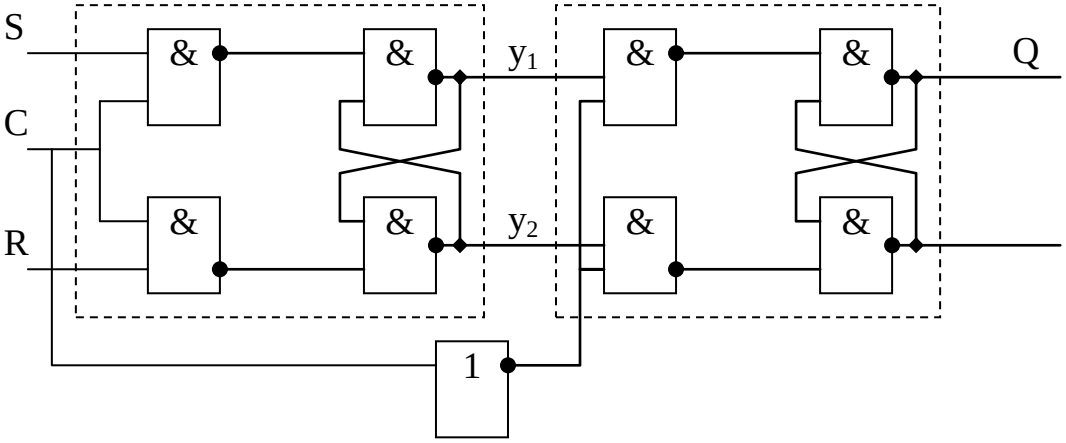
Нет запрещённых состояний.

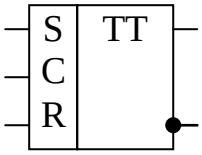


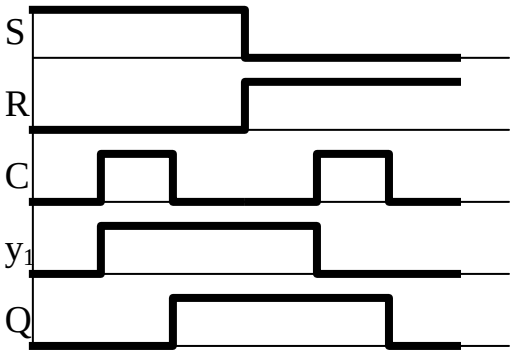
C	D	Q(t+1)
1	0	0
1	1	1
0	Q(t)	

<u>Двухтактный RS-триггер.</u>

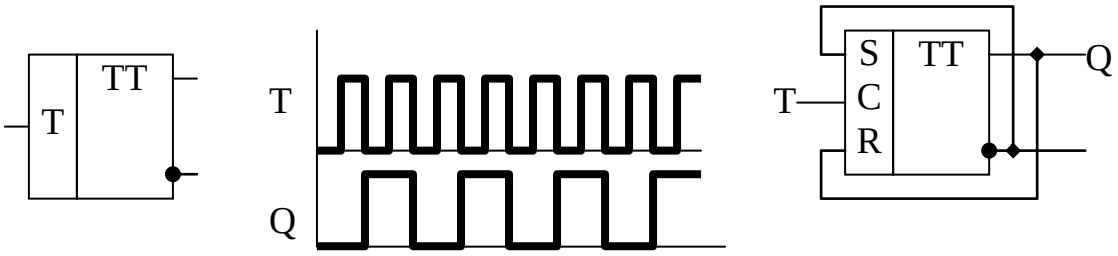
триггер состоит из двух RS-триггеров и инвертора.





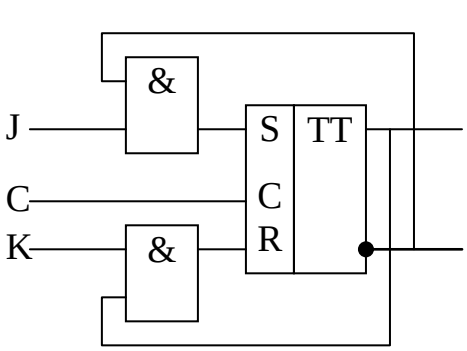


Т-триггер (счётный)

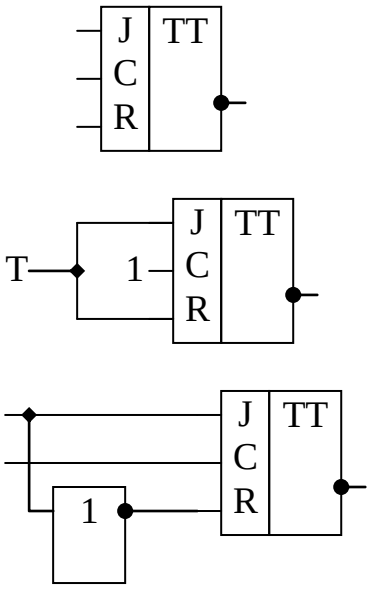


JK-триггер

Тоже на базе двухтактного (универсален)



C	J	K	Q(t+1)
1	0	0	Q(t)
	0	1	0
	1	0	1
	1	1	$\overline{Q(t)}$
0	Q(t)		



2.1.5. Регистры и счётчики.

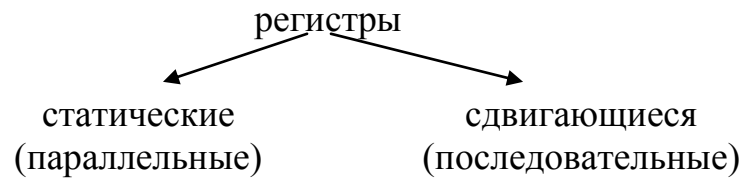
I. Регистры.

КА предназначенный для записи, хранения и считывания слов.

Кроме того можно:

- сдвиг слова влево или вправо на требуемое число разрядов;
- преобразование прямого кода в обратный (и наоборот);
- преобразование параллельного кода в последовательный (и наоборот);
- выполнение поразрядных логических операций (конъюнкций, дизъюнкций, сложение mod2 и др.)

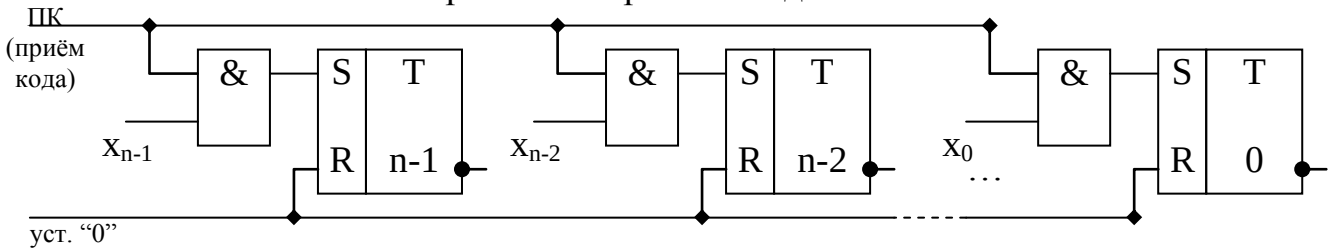
Основная функция – запоминание.

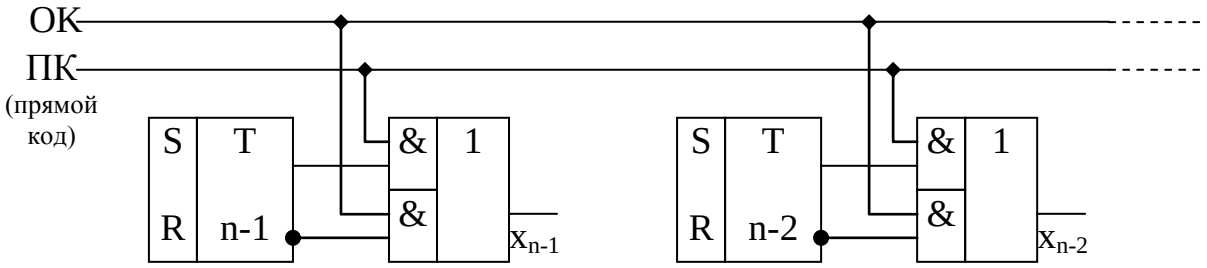


А. Статический регистр.

n-разрядное слово

Считывание слова в прямом и обратном кодах:

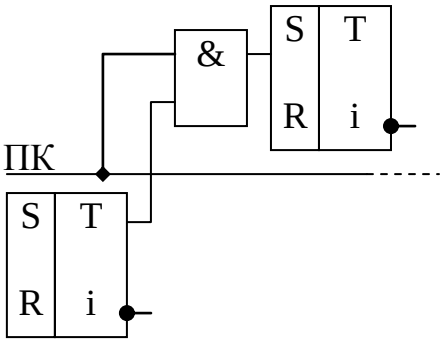




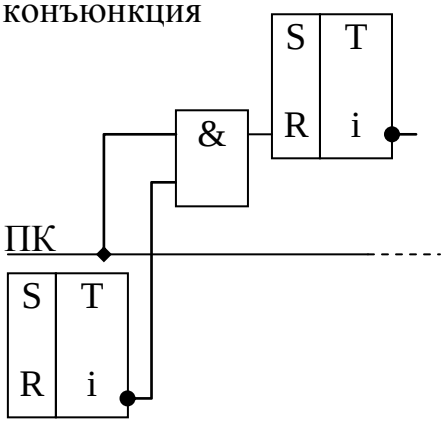
Поразрядные логические операции:

ПК – приём кода

дизъюнкция



конъюнкция



сложение по mod2

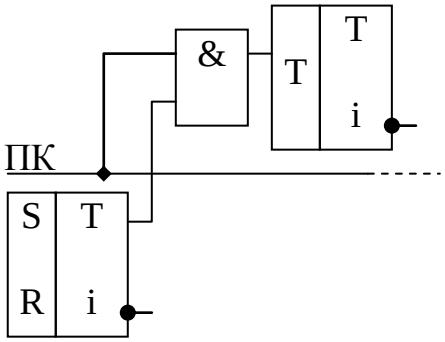
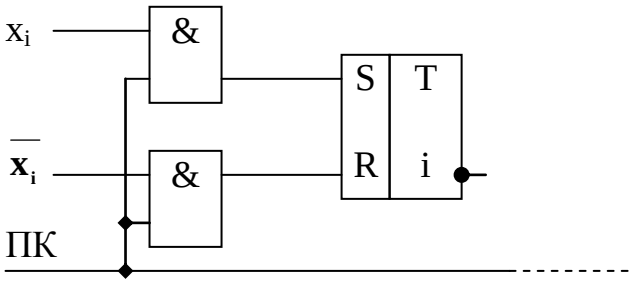
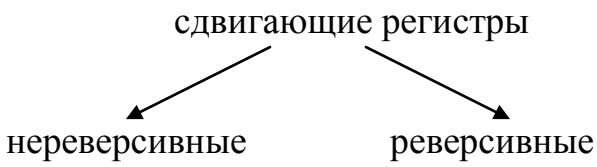


Схема регистра без предварительной установки в «0». Требует парафазных входов:

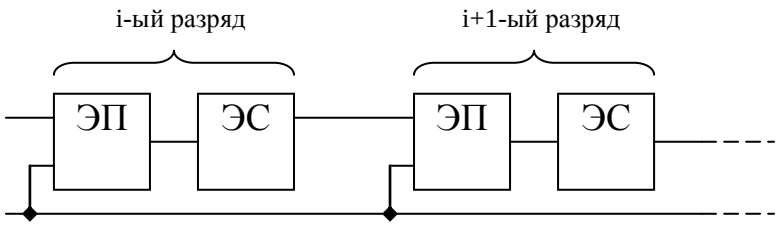


Б. Сдвигающие регистры.

Все разряды соединены последовательно. Нужны сигналы сдвига.



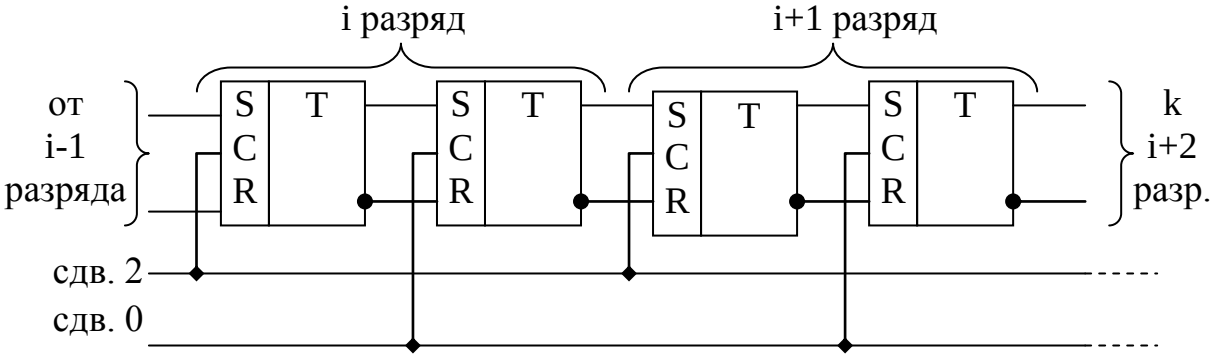
Идея:



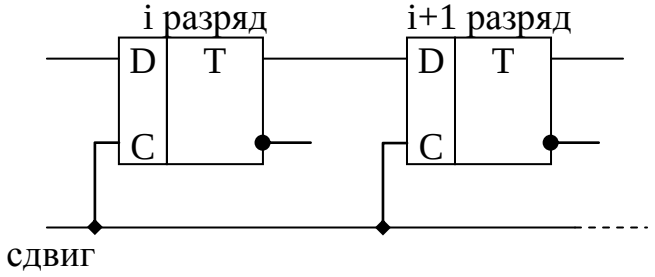
ЭП – элементы памяти

ЭС – элементы связи (задержки)

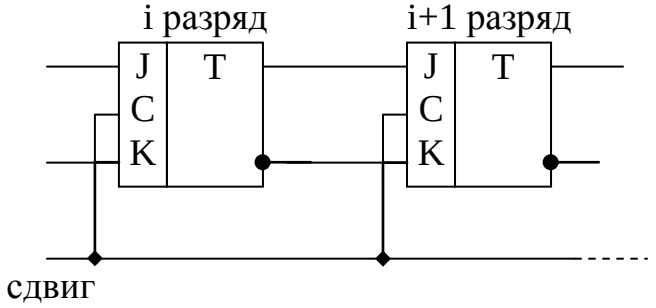
Сдвиг регистра на CRS-триггерах:
парафазный сдвиг 2 такта



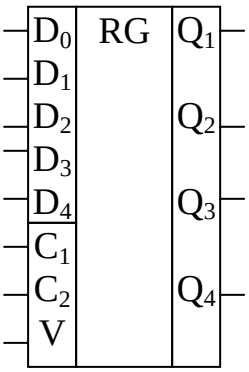
Сдвиг регистра на D-триггерах:

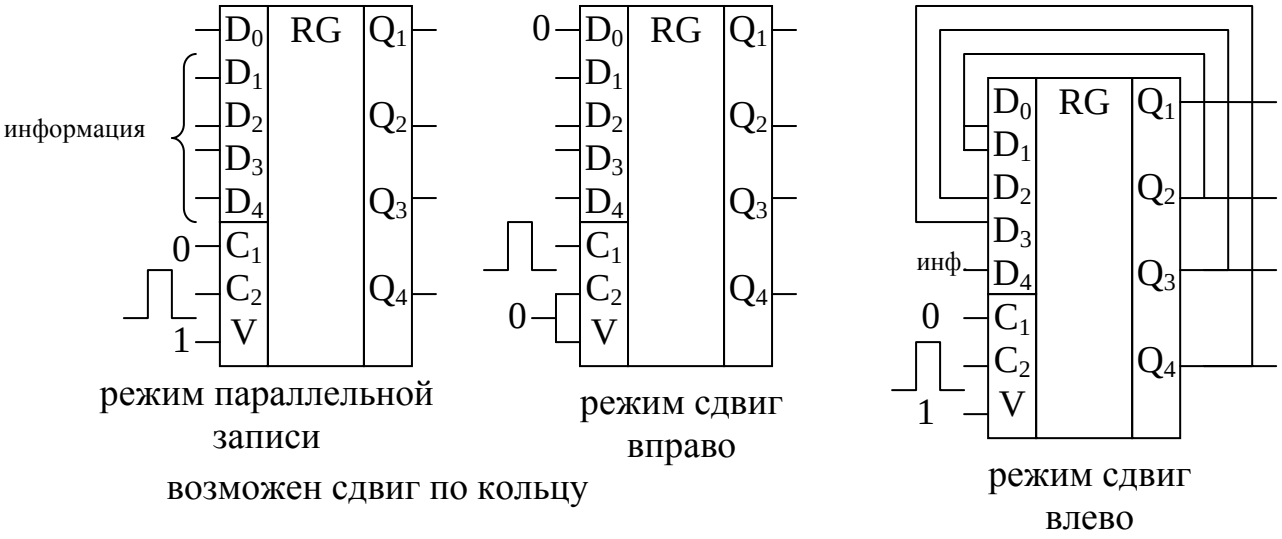


Сдвиг регистра на JK-триггерах:



Однотактные схемы
К155ИР1
D_0 – вход последовательной записи
D_1-D_4 – запись параллельным кодом
C_1 и C_2 – входы сдвига вправо и влево
V – вход управления





режим параллельной записи

режим сдвиг вправо

режим сдвиг влево

возможен сдвиг по кольцу

II. Счётчики.

КА выполняющий функцию подсчёта количества импульсов за некоторое время, а также формирования и запоминания двоичного кода, соответствующего этому количеству.

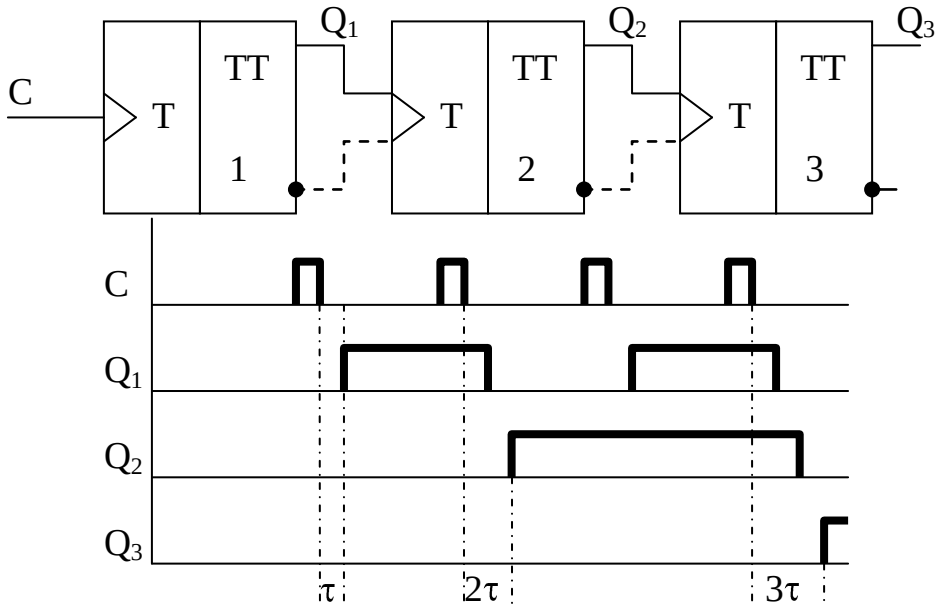
Иначе это: преобразователь числа импульсов в двоичный код.

счётчики

простые — реверсивные (как + так и -)

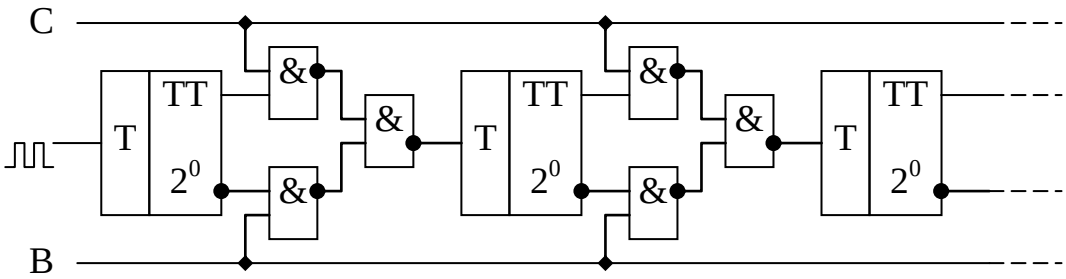
перенос может быть организован:
- последовательным (рассмотрим)
- сквозным (самостоятельно)
- групповым (самостоятельно)

А. Простые счётчики с последовательным переносом:



_____ - сложение; - - - - - - вычитание; $t_{установки\ max} = n \cdot \tau$

Б. Реверсивные счётчики с последовательным переносом.



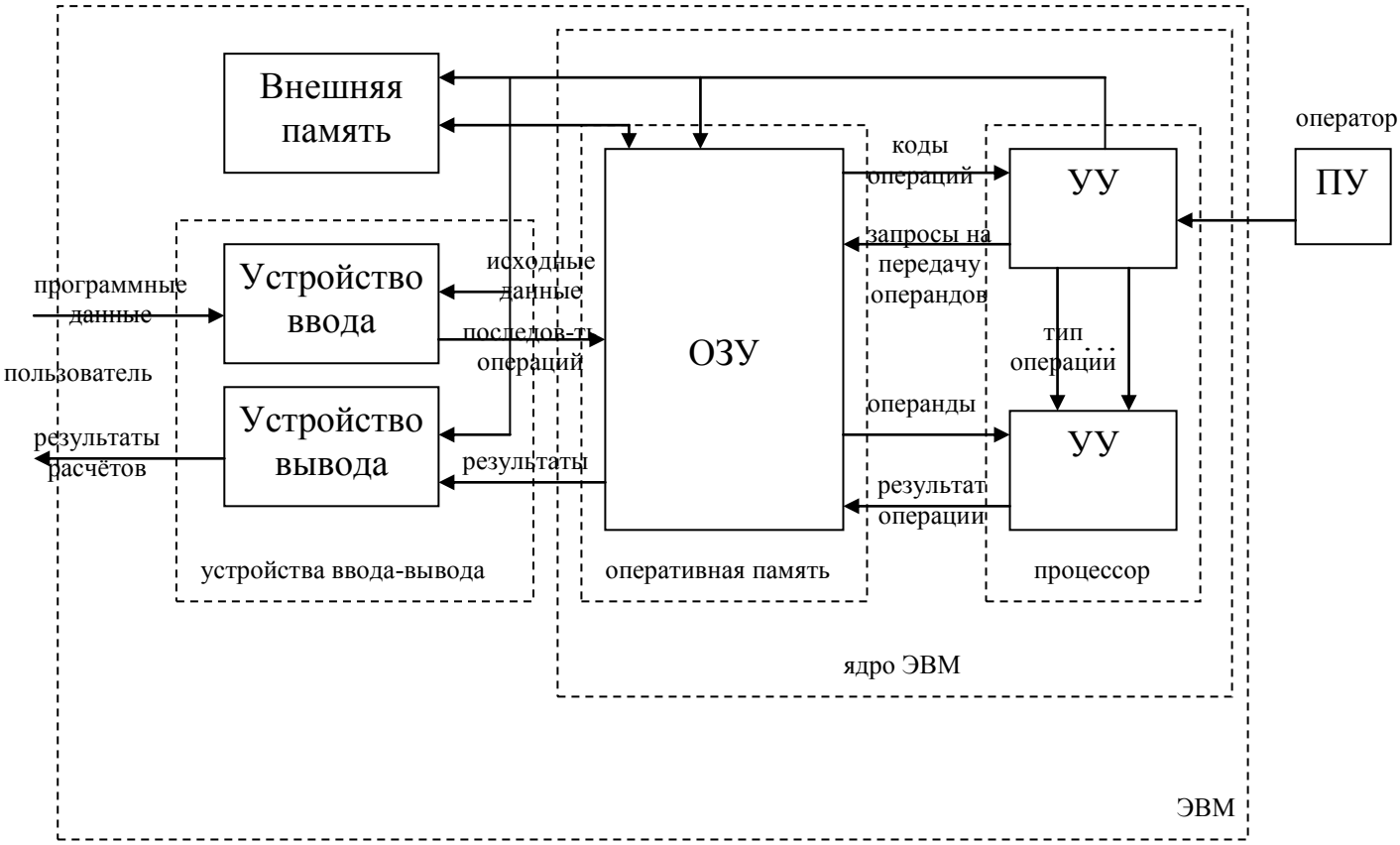
Сложение. Вычитание. С и В – постоянные потенциалы.
Пример: Было 3, вычитаем 4, В подаём 1, подаём ⎍⎍⎍. Итог -1.
одновременная подача
«1» на вход & и R – уст. в «0»
R_0 – вход общего сброса
С – вход синхронизации параллельной записи

Режимы К155ИЕ7:
параллельной записи: информация на D_1-D_4; R=C=0; +1=-1=1
Режим прямого счёта: +1=⎍; -1=C=1; R=0
Режим обратного счёта: -1=⎍; +1=C=1; R=0

2.2. Структура и принципы организации ЭВМ.

2.2.1. Принципы действия и основные характеристики ЭВМ.

Принцип действия рассмотрим по общей структуре:



ЭВМ содержит следующие основные устройства:

АЛУ – производит арифметические и логические преобразования над операндами

УУ – производит автоматически управление вычислительным процессом посредством сигналов управления, посылаемых всем устройствам ЭВМ

ОП – (ОЗУ) – производит запись, хранение и выдачу информации, непосредственно участвующей в вычислительном процессе

УВвода – производит считывание программ и исходных данных с носителей информации.

УВывода – производит выдачу результатов путём отображения её на АЦПУ или экране дисплея

ПУ – позволяет оператору проводить пуск, останов, тестирование ЭВМ и, если надо, вмешиваться в вычислительный процесс

ПРОЦЕССОР=АЛУ+УУ; ЯДРО=ПРОЦЕССОР+ОЗУ

Основные характеристики ЭВМ.

1) Производительность (быстродействие). Оценивается по смесям команд.

«Смесь Гибсона» - (для оценки производительности решения научно-технических задач). Включает: сложение, вычитание, умножение, деление (с фиксированной и плавающей запятыми), логические операции, команды передачи управления, операции с различными коэффициентами.

Производительность:

$P = \sum_{i=1}^{n} k_i \Big/ \sum_{i=1}^{n} k_i t_i$ k – коэффициент, t – время, n – число команд в смеси.

2) Эффективность.

$Э = P \Big/ (S_{ЭВМ} + S_{Экспл.})$, где $S_{ЭВМ}$ – стоимость ЭВМ, $S_{Экспл.}$ – стоимость эксплуатации (помещения, энергии, обслуживания)

3) Объём оперативной памяти.

Оценивается в Кбайтах или Мбайтах. 1Кбайт=1024байта

4) Разрядность машинного слова.

5) Надёжность.

По-разному. Например коэффициент готовности:

$K_Г = T \Big/ T + T_В$, где Т – время наработки на отказ, $T_В$ – время наработки на восстановление

6) Дополнительные характеристики.

Габариты. Вес. Энергопотребление. Климатические условия. Стоимость. Программное обеспечение и др.

2.2.2. Устройства памяти.

I. Основные понятия и определения.

Память ЭВМ – совокупность устройств для записи, хранения и выдачи информации

Информация – некоторая совокупность сведений, данных, знаний.

По Шеннону – мера неопределённости

$H = -\sum_{k=1}^{n} p_k \log_2 p_k$, где H – энтропия (мера неопределённости), n – количество состояний, p_k – вероятность k-ого состояния

Измерение информации (количеством) $I=H_{апр.}-H_{апост.}$ есть мера уменьшения неопределённости.

$H_{априорная}$ – информация до опыта, $H_{апостариорная}$ – информация после опыта.

Пример:

Бросок монеты.

$n=2$; $p_1=p_2=0,5$

$H_{апр.}=(-0,5\log_2 0,5)+(-0,5\log_2 0,5)=1$

$H_{апост.}=0$ (после броска).

Следовательно: I=1-0=1. Это бит.

Единица информации это неопределённость любого процесса имеющая 2 равновероятных исхода.

Характеристика памяти.

а) ёмкость памяти

б) удельная ёмкость – отношение ёмкости ЗУ к его физическому объёму

в) быстродействие ЗУ – характеризуется выполнением основных операций: записи и считывания информации.

$t^{обр.}_{счит.}$ - время обращения при считывании

$t^{обр.}_{зап.}$ - время обращения при записи

$t_{обр.}=\max(t^{обр.}_{счит.}, t^{обр.}_{зап.})$.

$t^{обр.}_{счит.}=t_{поиска\ (доступа)}+t_{считывания}+t_{записи}$.

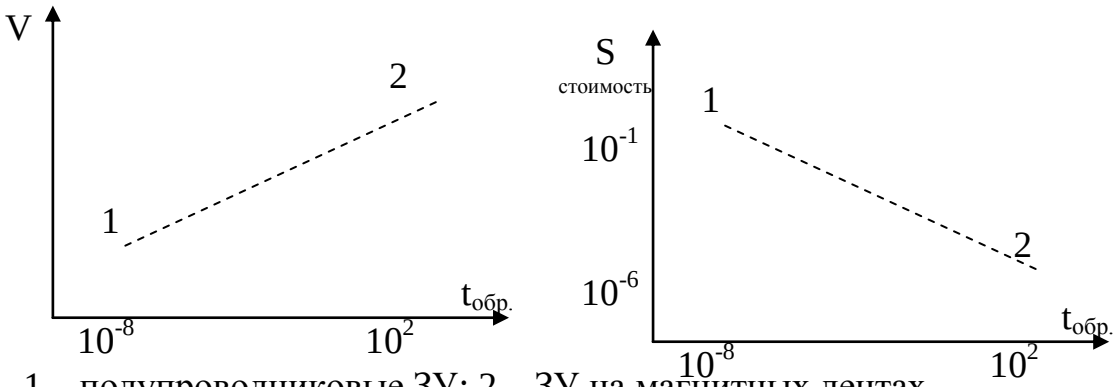
г) стоимость хранения 1-цы информации (бита)

в диапазоне 10^{-1}-10^{-6} руб./бит

Замечание:

увеличение ёмкости ЗУ => уменьшение быстродействия

увеличение времени обращения => уменьшение стоимости бита.



1 – полупроводниковые ЗУ; 2 – ЗУ на магнитных лентах

II. Классификация ЗУ.

а) по физическому принципу:

- электронные (полупроводниковые)
- магнитные
- магнитооптические
- оптические

б) по способу организации доступа к информации:

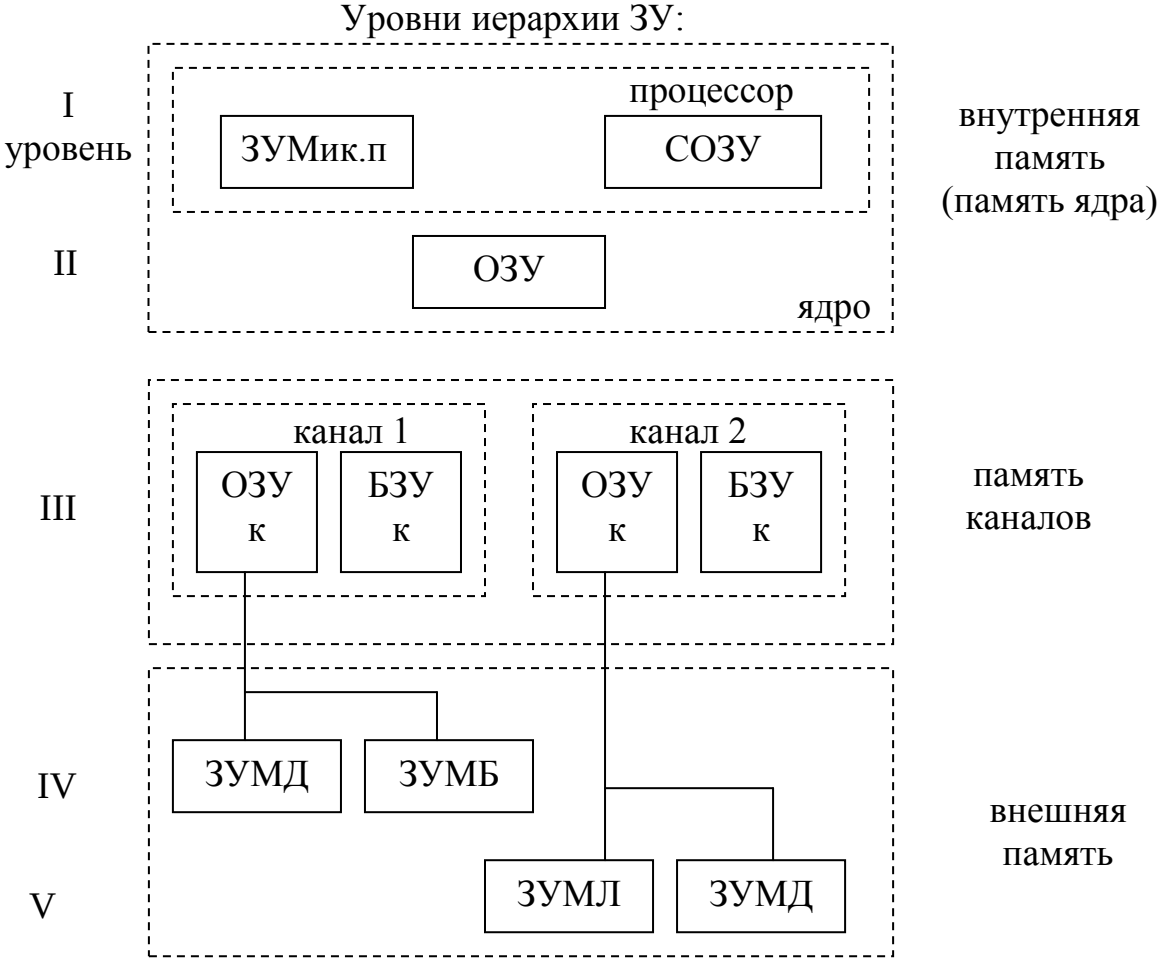
- с произвольным доступом (время не зависит от места расположения информации в ЗУ) Цикл обращения 10нСек - 1-2мсек
- с прямым доступом (циклическим) (магнитный барабан, диск). Цикл обращения 10мсек - 0,1сек
- с последовательным доступом (магнитная лента). Цикл до нескольких секунд

в) по реализуемым операциям обращения

- с произвольным обращением (и запись и считывание)
- с обращением только при считывании (записи нет)

г) по функциональному назначению в ЭВМ

Уровни иерархии ЗУ:



I

ЗУМик. п – ЗУ микропрограмм

СОЗУ – сверхоперативное запоминающее устройство

II

ОЗУ – оперативное ЗУ

III
Собственная память каналов обмена (оперативное ЗУ канала + буферное ЗУ)
IV
Внешнее ЗУ (диски, барабаны)
хранят основную информацию вне ядра
V
внешнее ЗУ (диски, ленты)
хранят архивные данные

III. Адресная организация ЗУ и их структуры.

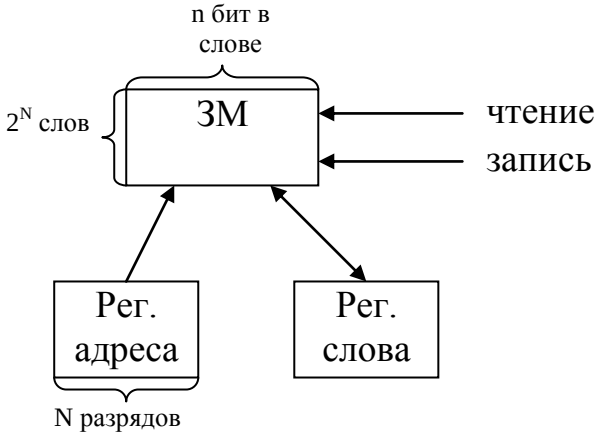
Способ организации памяти зависит от методов размещения и поиска информации в запоминающем массиве (ЗМ). Запоминающий массив – совокупность однотипных запоминающих элементов.

Структура адресной организации ЗУ с произвольным доступом:

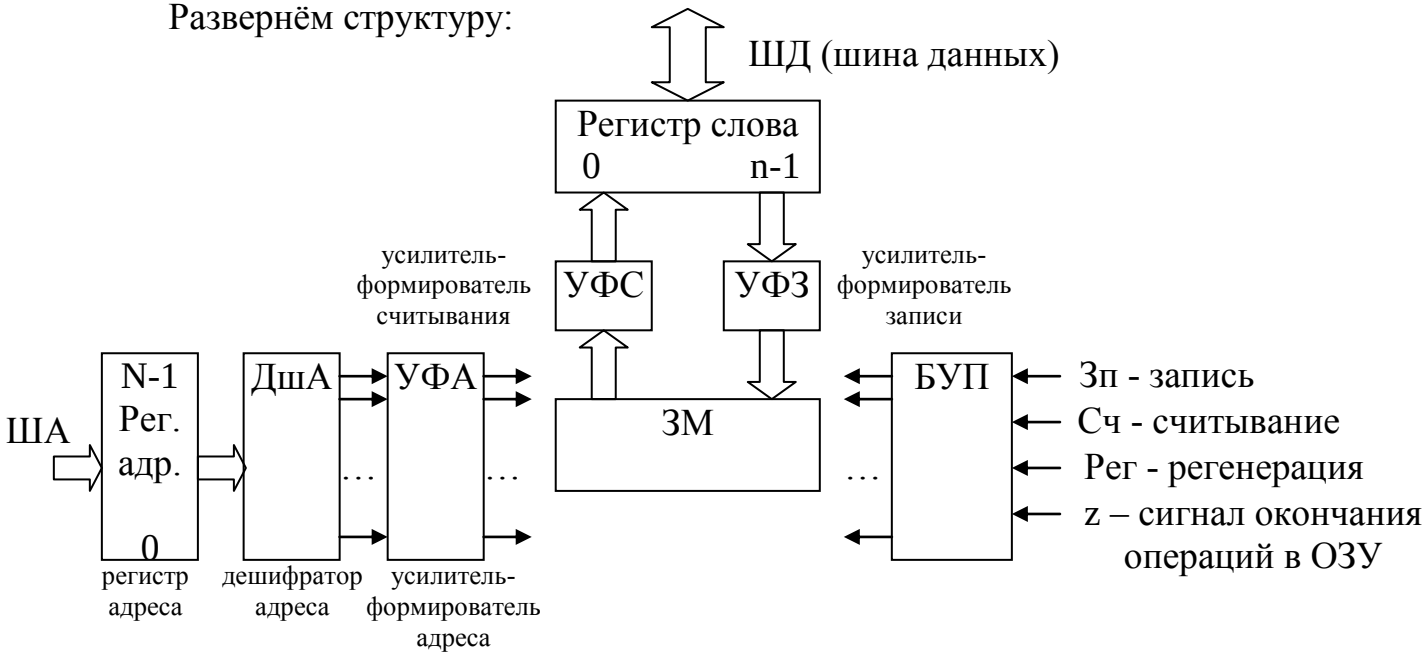
Каждая ячейка памяти в ЗМ имеет свой адрес.

Поиск слова осуществляется по адресу ячейки.

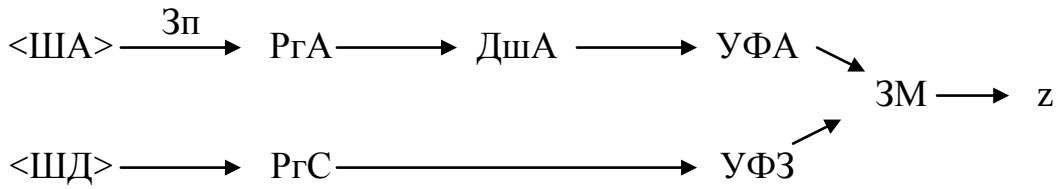
На принципах адресной организации основано ОЗУ. Адрес ячейки поступает на регистр адреса, т. к. в нём n разрядов то возможна адресация 2^N слов ЗМ. В зависимости от подачи сигнала на «чтение» или «запись» в регистр слова поступит считанное слово, или из регистра поступит в ЗМ.



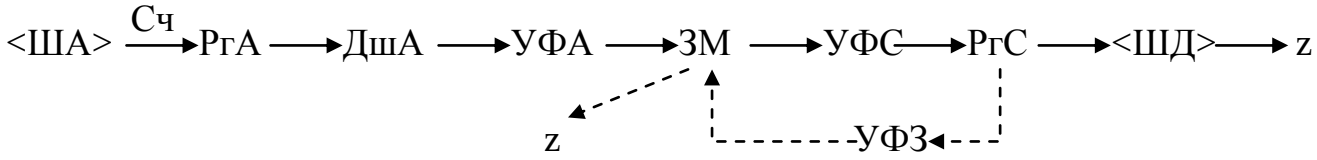
Развернём структуру:



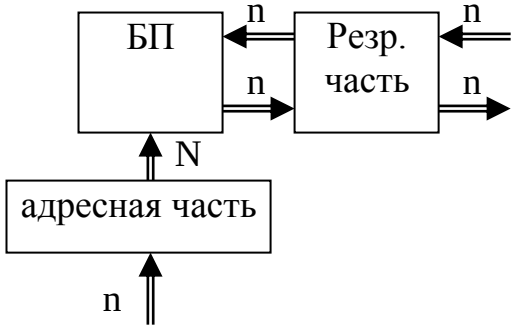
Операция записи:



Операция считывания:

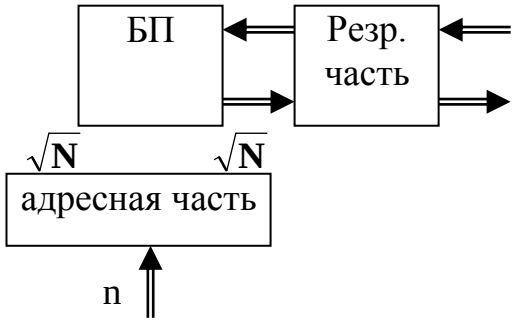


в зависимости от структуры пространственного размещения ячеек памяти в ЗМ, различают адресные ОЗУ с 2D; 3D; 2,5D; 1D организацией ЗМ. (D – размерность)



2D-организация ЗМ:

Это двумерная организация
(по одному направлению идёт адрес,
а по другому – слово)
БП – блок памяти
n – разрядный адрес на входе
$N=2^n$
плюс – высокое быстродействие,
помехоустойчивость: $\frac{\text{сигнал}}{\text{помеха}}=\frac{5}{1}$
минус – сложность адресной части



3D-организация ЗМ:

Количество выходов из адресной части $=2\sqrt{N}$

Выбор ячейки памяти осуществляется совпадением сигналов по двум шинам. Это существенно упрощает дешифратор адреса, но будут полувыбранные ячейки. Опасность ложного считывания (записи) это исключают:

- строгим стробированием сигналов (УФА выдаёт очень короткие импульсы)
- высокой синхронизацией сигналов с УФА и УФС (или УФЗ)
- жёсткой регенерацией.

плюс – проще адресная часть
минус – низкая помехоустойчивость =2

2,5D – организация ЗМ:

является промежуточной. Адрес разбивается на группы, а внутри группы 3D – организация.

1D – организация ЗМ:
Используется для постоянных ЗУ.
Обозначение ОЗУ на схемах:

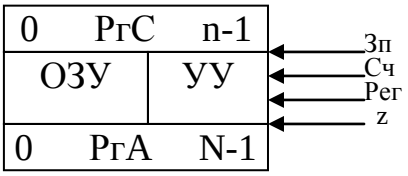
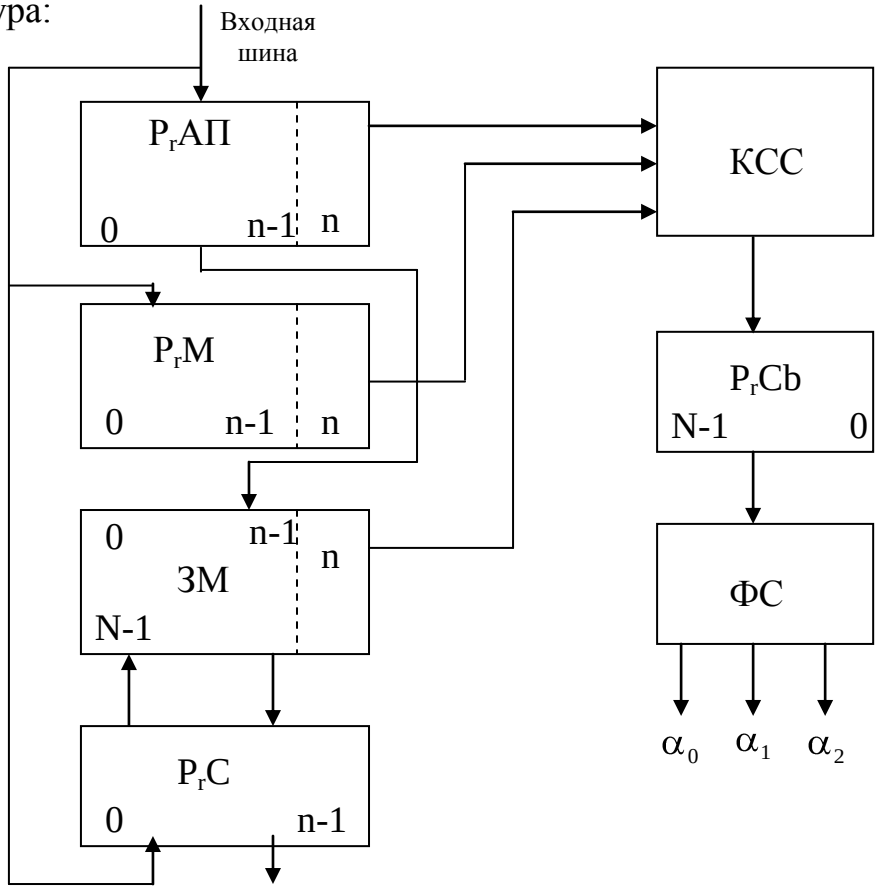


Таблица рекордов

	$t_{выб}$ nc	V бит	бит/см^3	Энергопотребление при хранении
Биполярный	40	10^5	200	+
МОП-струк	250/100	10^6	300	+
Ферритовый сердечники	250	10^8	20	-

IV. Ассоциативные ЗУ.
Поиск по ассоциативному признаку.
структура:



$Р_г$АП – регистр ассоциативного признака
$Р_г$М – регистр маски
КСС – комбинационная схема сравнения
$Р_г$Сb – регистр совпадения
ФС – формирователь сигналов
$Р_г$С – регистр слова

В ЗМ хранятся N (n+1)-разрядных слов. n-ый разряд указывает на занятость ячейки (0 – свободна; 1 – записано слово).

Принцип работы:

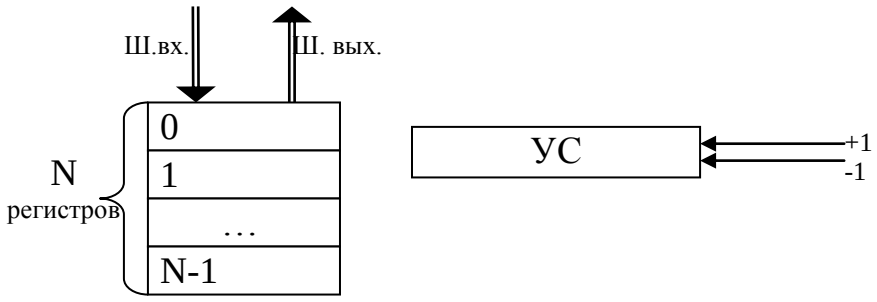
По входной шине поступает в P_rАП ассоциативный запрос (в $0 \div n-1$ разряды), а в P_rМ – код маски поиска (в $0 \div n-1$). n-ый разряд устанавливается в 0. Ассоциативный поиск проводится по совокупности разрядов P_rАП, которым соответствуют 1-цы в разрядах P_rМ (немаскированные разряды). Для слов, у которых цифры в разрядах совпали с незамаскированными разрядами P_rАП, КСС устанавливает 1 в соответствующие разряды P_rCb и 0 в остальные разряды.

Таким образом позиции 1-ц в P_rCb соответствуют адресам слов в ЗМ удовлетворяющих ассоциативному поиску. Комбинационная схема ФС формирует из слова в P_rCb сигналы $\alpha_0, \alpha_1, \alpha_2$.

V. Стековая и магазинная память.

Стековая память – безадресная.

Стек – совокупность N связанных поразрядно регистров, образующих массив информации.



УС – указатель стека

Запись и считывание информации осуществляется по принципу: первым пришёл – последним обслужен (LIFO).

Запись только в верхний регистр. При этом все слова вместе с этим сдвигаются на 1 вниз.

Считывание в стеке только из верхнего (0-го) регистра.

УС – счётчик, хранящий количество слов, записанных в стеке или переполнение.

Более сложная организация стека – связанные списки.

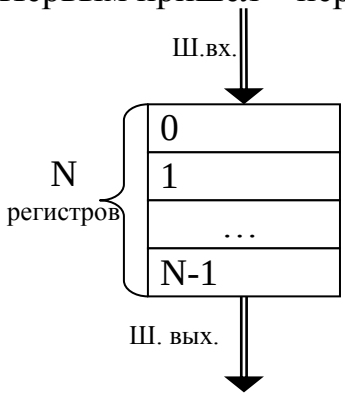
При записи слова занимаются 2 регистра:

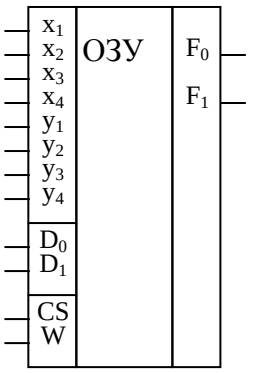
1 для слова

1 для № регистра
для следующего слова.

Магазинная память.

Первым пришёл – первым обслужен (FIFO).





Примеры ЗУ.

Схемы ЗУ – Р.

РА – ассоциативные ЗУ

РВ – постоянные ЗУ

РУ или РМ – ОЗУ

РР – перепрограммируемые ЗУ

первая буква (Р) – функциональный класс, вторая (А, В, У или М, Р) – группа.

$V_{\text{памяти}} = 16 \times 1\text{бит}\,(16\text{ одноразрядных слов})$

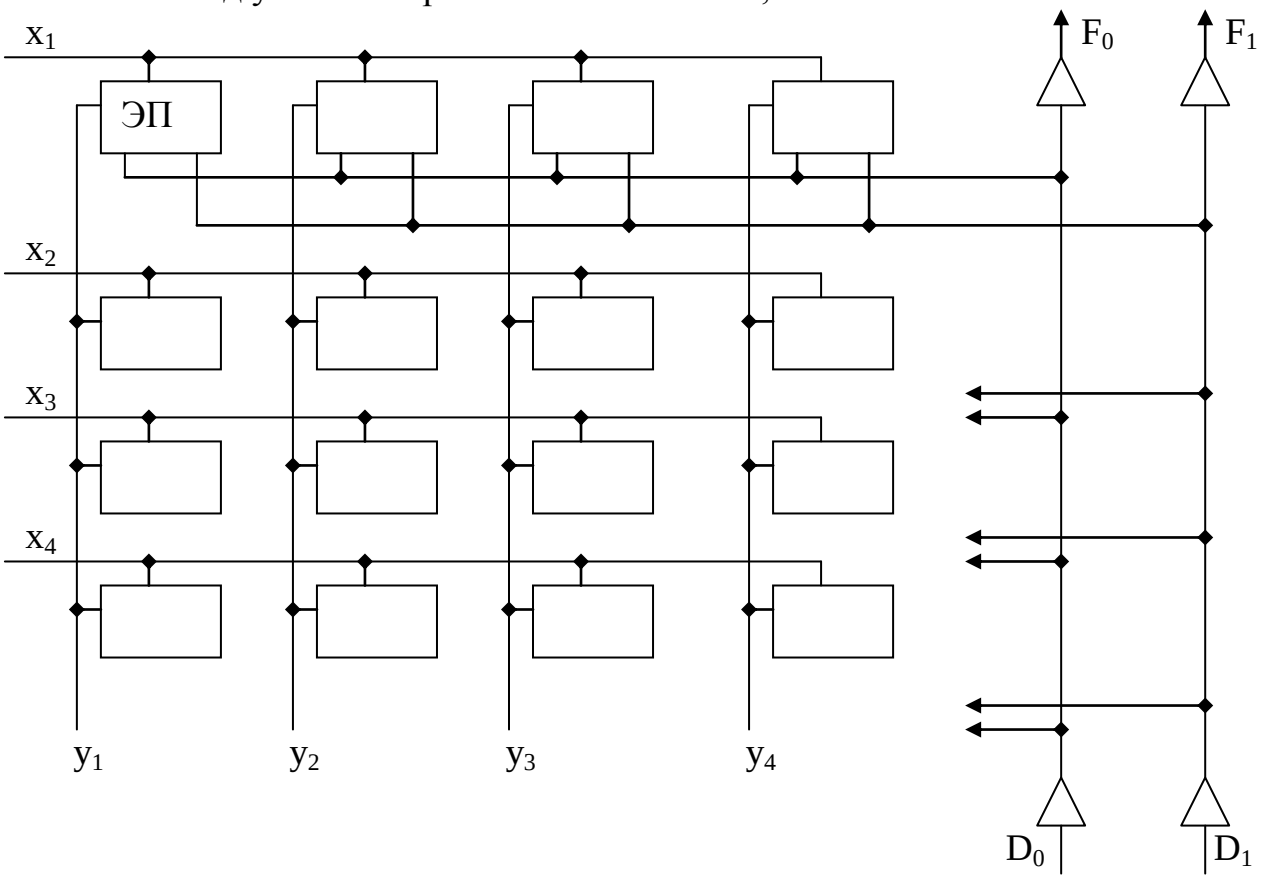
Запись информации D_0, D_1 в парафазном виде

Считывание F_0, F_1 в парафазном виде

x_1-x_4 и y_1-y_4 шины выборки элемента памяти (адресные шины)

CS – вход выбора микросхемы (1)

W – вход установки режима: «0» - чтение; «1» - запись.



2.3. Процессоры: архитектура и принципы организации.

Процессор – функциональный блок ЭВМ для арифметической и логической обработки информации на основе программного управления.

Состав: АЛУ, УУ, СОЗУ, Брег и др.

I. Система команд ЭВМ.

Рассмотрим 3 части:

А. Классификация команд (операций).

Б. Структура и форматы команд.

В. Способы адресации.

А. Классификация команд(операций).

в современном ЕС ЭВМ около 200 команд.

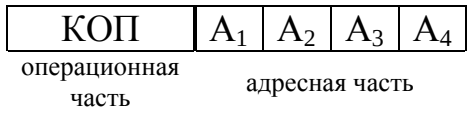
- арифметические операции: сложение, вычитание, умножение, деление, изменение знака, перенос, заем и др.
- логические операции: $\neg \vee \wedge \oplus$
- операции сдвигов: арифметический, циклический, логический
- пересылочные операции:
  - регистр – регистр (внутри процессора)
  - регистр – память (между процессором и ОЗУ)
  - память – память (внутри ОЗУ)
- операции управления: условные, безусловные переходы, вызов подпрограмм, возврат из подпрограмм, программные прерывания и др.

Б. Структура и форматы команд.

Команда – двоичный код, определяющий операцию выполняемую процессором и данные (операнды), участвующие в операции.

Операнды указываются адресами ячеек памяти, где они содержатся.

Команда в общем виде:



A_1 – адрес 1-ого оператора

A_2 – адрес 2-ого оператора

A_3 – адрес результата

A_4 – адрес следующей команды

Структура команды: - определяется составом, назначением и расположением полей в команде.

Пример.

В ЕС ЭВМ около 200 команд и до 16Кбайт адресуемой памяти, что приводит к длине 4-х адресной команды:

$$n_{ком} = n_{коп} + 4 \cdot n_A \geq \log_2 200 + 4 \cdot \log_2 (16 \cdot 1024 \cdot 8) = 8 + 4 \cdot 17 = 76 \text{ дв.разр. (бита)}$$

Проблема: длина команды

Задача: уменьшить длину

Для упрощения аппаратуры процессора и повышения быстродействия нужно ускорить команды до машинного слова (полуслова). Поэтому применяют более простые структуры команд:

КОП	A_1	A_2	A_3		КОП	A_1	A_2		КОП	A_1		КОП	- безадресная команда

Сейчас в основном 2-х и одноадресные команды. В микро ЭВМ и микропроцессорах – даже безадресные (стеки)

2-х адресная команда: $A_1:=A_1*A_2$, где * - знак операции.

1-но адресная команда:

отдельный регистр процессора - аккумулятор

$A_{кк}:=A_{кк}$*операнд.

В. Способы адресации.

Решение задачи уменьшения длины команд породило различные способы адресации информации.

1. прямая адресация

а) абсолютная прямая адресация б) неявная адресация

2. косвенная адресация

а) абсолютная косвенная адресация б) регистровая косвенная адресация

3. относительная адресация

а) адресация по базе б) индексная адресация

4. непосредственная адресация.

Рассмотрим основные способы адресации на примере одноадресной структуры команд.

Введём обозначения:

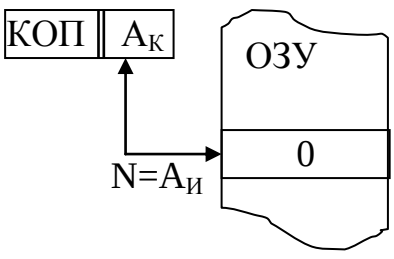
О – операнд (число над которым выполняется операция)

$A_К$ – адресный код (адрес операнда)

$A_И$ – исполнительный адрес (номер ячейки памяти к которой происходит фактическое обращение).

1. Прямая адресация:

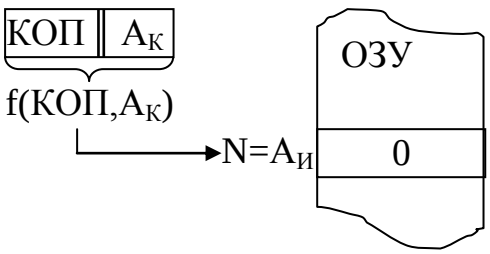
содержимое адресной части прямо указывает на адрес операнда:



абсолютная прямая адресация: $A_К=A_И$

неявная адресация: $A_И=f(КОП, A_К)$, т. е. $A_И$ определяется специальным образом из информации, содержащейся как в КОП, так и в $A_К$.

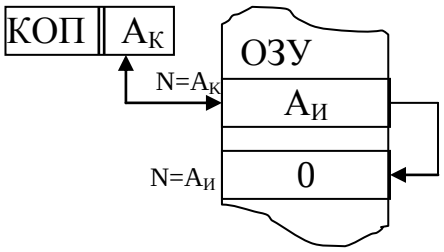
2. Косвенная адресация:



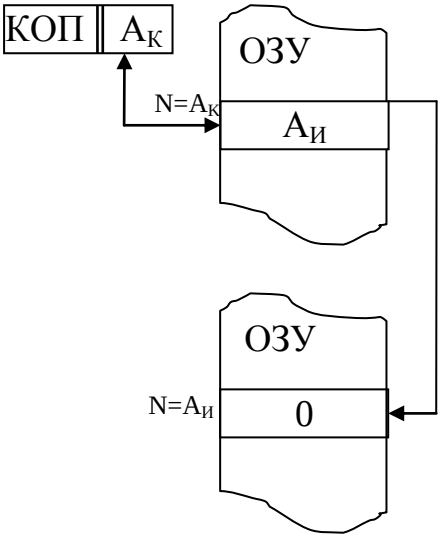
$A_К$ указывает адрес ячейки, в которой находится $A_И$, т. е. адрес операнда.

Это в малых и микро-ЭВМ с коротким словом.

Абсолютная косвенная адресация



Регистровая косвенная адресация



Используется в малых и микро ЭВМ.

3. Относительная адресация.

$A_И$ определяется смещением некоторого заданного числа на значение, определяемое адресной частью команды.

Адресация по базе:

$A_И=A_К+A_Б$, где $A_Б$ – некоторое число, называемое базовым адресом и хранящееся в специальном базовом регистре процессора.

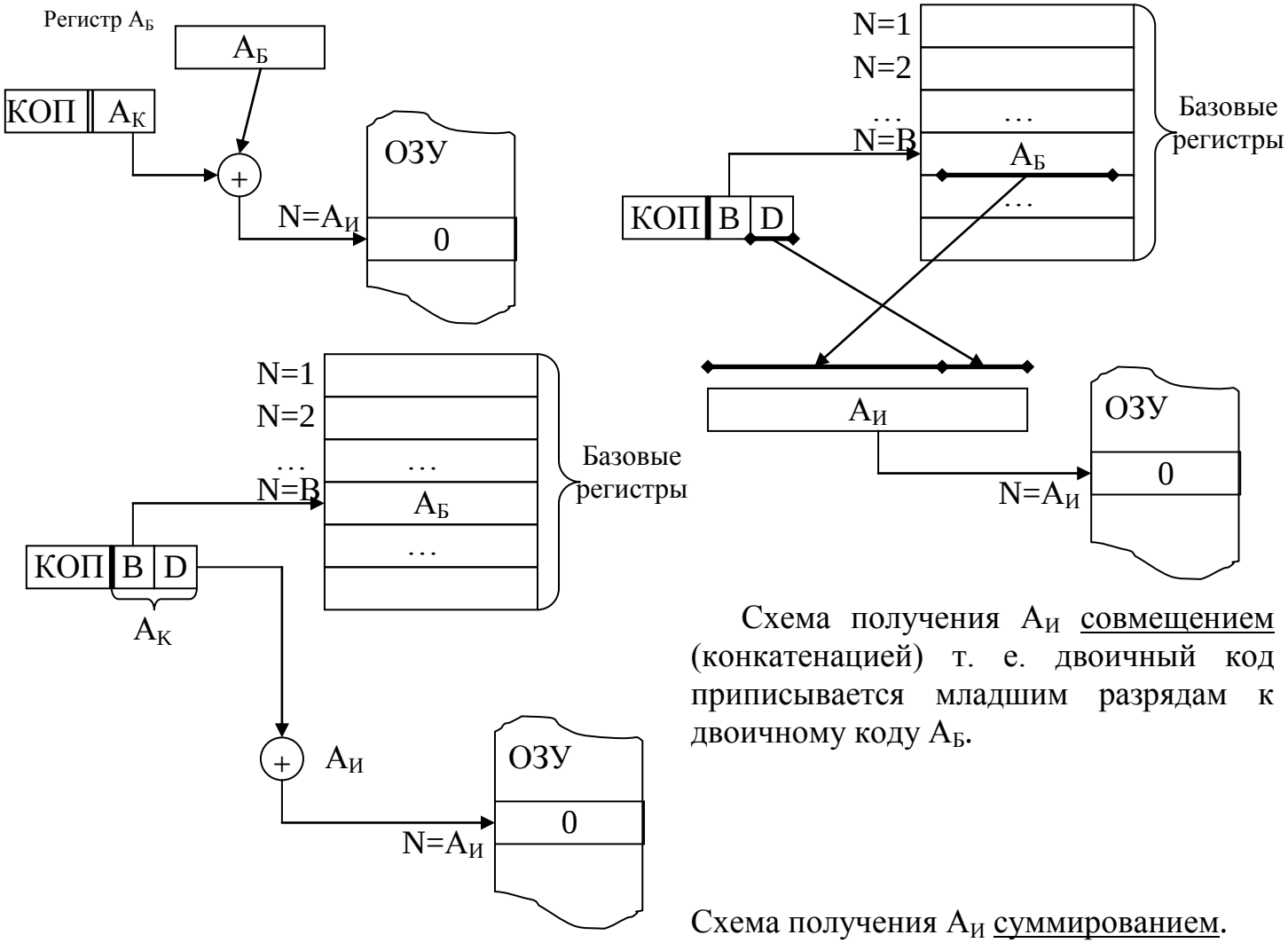
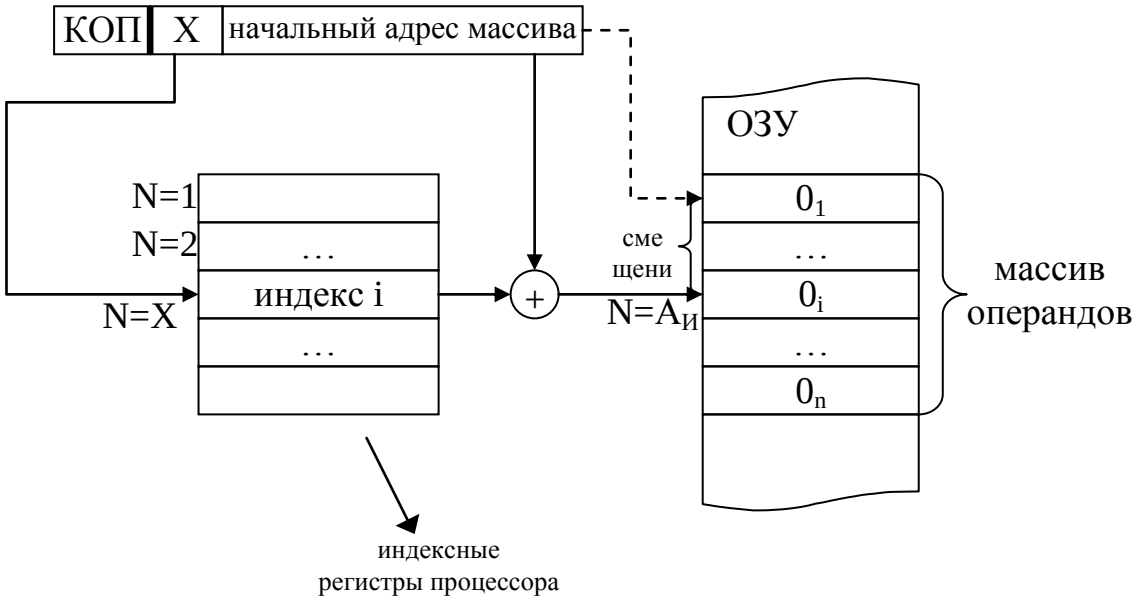


Схема получения $A_И$ совмещением (конкатенацией) т. е. двоичный код приписывается младшим разрядам к двоичному коду $A_Б$.

Схема получения $A_И$ суммированием.

Индексная адресация: если операнды являются переменными с индексами, т. е. элементами массивов. Используется индексные регистры, расположенные в процессоре.

X – дополнительное поле указывающее номер индексного регистра.



4. Непосредственная адресация:

В команде содержится не адрес операнда A_K, а сам операнд. Используется это при хранении различного рода констант.

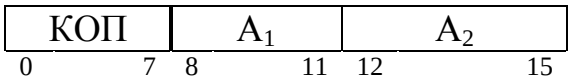
Примеры:

Команды ЕС ЭВМ:

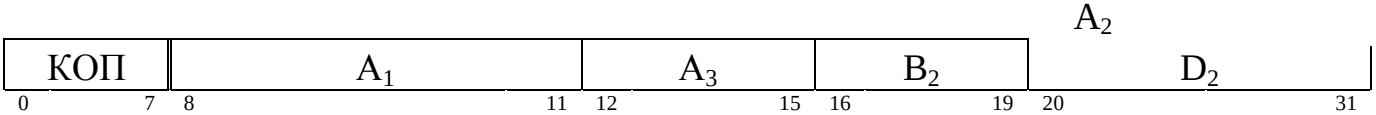
Формат RR – регистр-регистр

Формат RS – регистр-память

½ слова



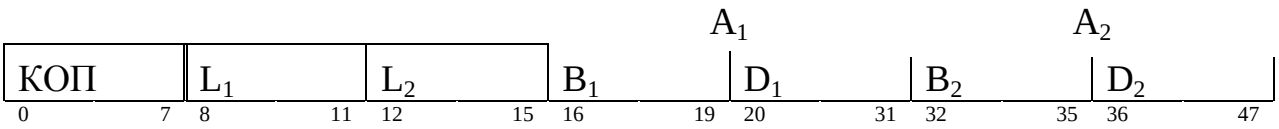
3-х адресная команда 1 слово



Формат SS – память-память

2-х адресная команда

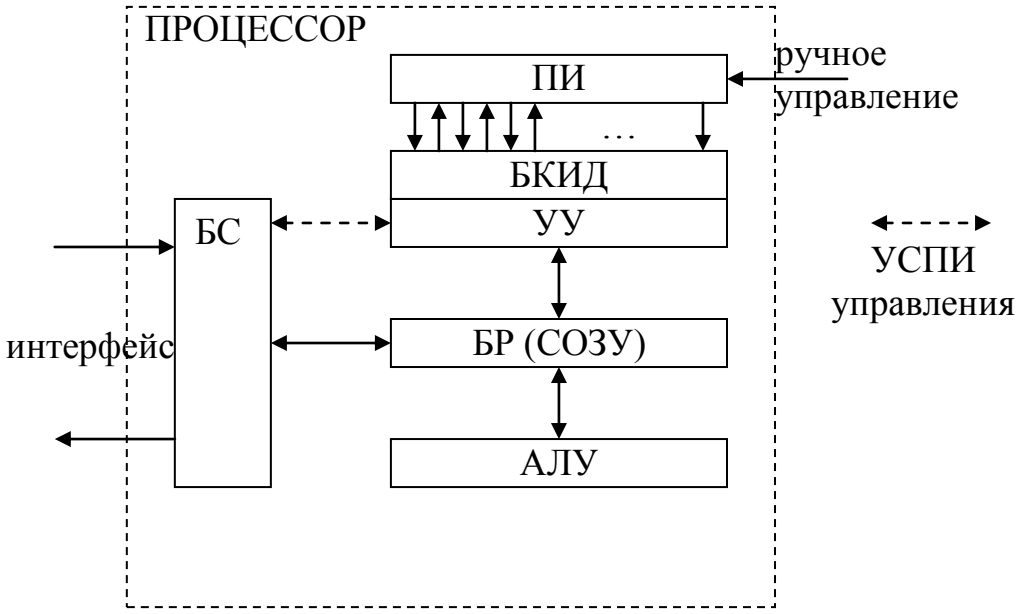
1 ½ слова



L_1 и L_2 – шестнадцатиричные числа указывающие на длину (1 до 16 байт) первого и второго операндов соответственно. Применяется для операций над операндами переменной длины. Всего в ЕС ЭВМ 6 основных форматов команд + их модификации.

II. Структура процессора и алгоритм его работы.

Блок-схема:



БР – блок регистров (местная память процессора)

БС – блок сопряжения с интерфейсом

БКИД – блок контроля и диагностики

ПИ – пульт индикации.

Регистры БР могут жёстко не фиксироваться, а назначаться из 8-32х РОН (регистров общего назначения).

БР состоит из:

программно-доступных регистров:

- аккумулятор
- базовые регистры
- индексные регистры
- указатель стеков
- счётчик команд и др.

программно-недоступных регистров:

- это рабочие регистры использующиеся в процессе выполнения одной команды:

РК – регистр команды

РА – регистр адресов

РС – регистр слов (операндов) и др.

БС – организует обмен информацией между процессором и ОЗУ, а также связь процессору с периферийными устройствами.

Интерфейс процессора включает:
шины управления
шина адреса
шина данных
шина задания режима ввода
шина задания режима вывода
шина синхронизации

БС выполняет только 2 операции:
ВВОД СЛОВА
ВЫВОД СЛОВА.

При вводе БС последовательно устанавливает:

а) код режима ввода на шину управления

б) адрес слова в ОЗУ (или периферийного устройств) на шину адреса

в) выдача сигнала синхронизации по шине синхронизации.

При вводе слова из процессора последовательно устанавливаются:

а) код режима вывода на шину управления

б) адрес слова в ОЗУ на шину адреса

в) данные на шину данных

г) выдача сигнала синхронизации по шине синхронизации

БКИД – служит для обнаружения сбоев и отказов в аппаратуре процессора; восстановление после сбоя и поиск мест неисправности при отказе.

ПИ – обеспечивает индикацию основных РОНов и базовых точек процессора. Обеспечивает режим ручного управления.

Алгоритм работы процессора

Комментарии

1. Адрес команды хранится в счётчике команд. При определении адреса следующей команды к текущему содержимому счётчика прибавляется длина предыдущей команды.

2. Адрес команды посылается из счётчика команд в РА, а БС выполняет операцию ввода слова через интерфейс.

Введённое слово поступает на РК. Если длина команды >1 слова, то на основании анализа 1-го слова вводится остальная часть команды.

3. Определяется группа команды и её адресность

4. ----------

5. По вычисленным адресам производится ввод операндов из ОЗУ через интерфейс на регистры слов БР

6. Подготавливается последовательность действий АЛУ по выполнению данной команды

7. Операнды подаются в АЛУ. Работой АЛУ управляют сигналы с шага

8. Результат с выхода АЛУ даётся на место операнда приёмника в соответствующий регистр БР

8. Результат выводится из процессора. Интерфейс обеспечивает режим вывода.

III. АЛУ.

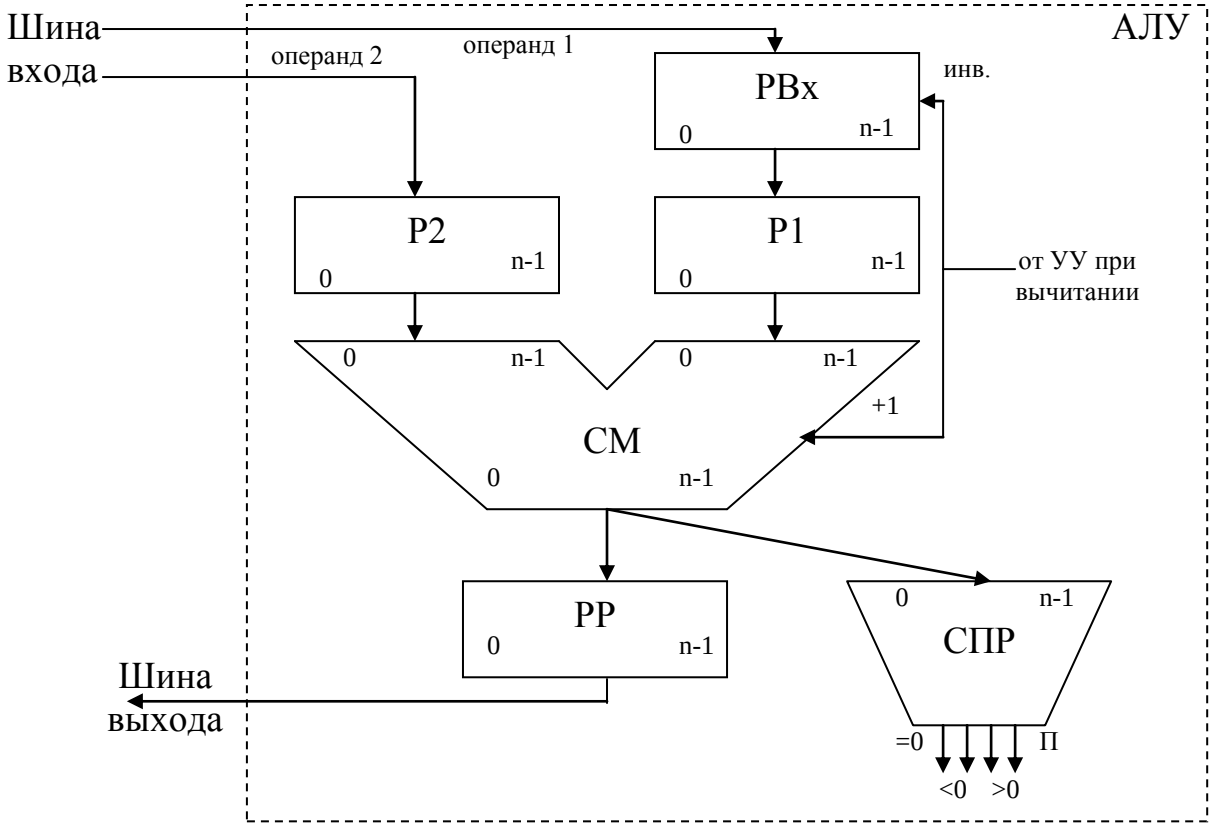
В общем случае АЛУ обеспечивает выполнение всех операций арифметико-логической группы.

Структура АЛУ зависит от формы представления данных (операндов), т. е. существуют:

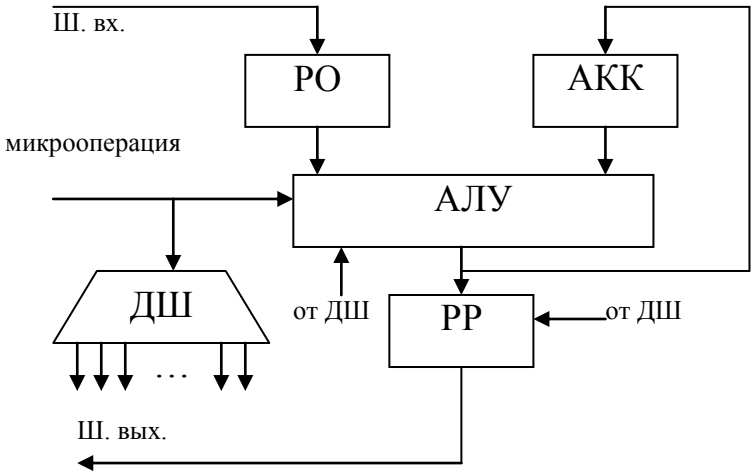
- АЛУ с фиксированной запятой
- АЛУ с плавающей запятой
- десятичные АЛУ (десятичная арифметика) и др.

В универсальных ЭВМ существуют многофункциональные АЛУ.

Ограничимся рассмотрением структуры и принципов работы АЛУ для положительных и отрицательных чисел с плавающей запятой.



РВх – входной регистр
Р1 и Р2 – регистры операндов
СМ – параллельный комбинационный сумматор
РР – регистр результата
СПР – КС формирования признака результата:
=0 – нулевой результат
>0; <0 – формирование знака результата
П – переполнение (поступает в УУ, где формируется код прерывания)

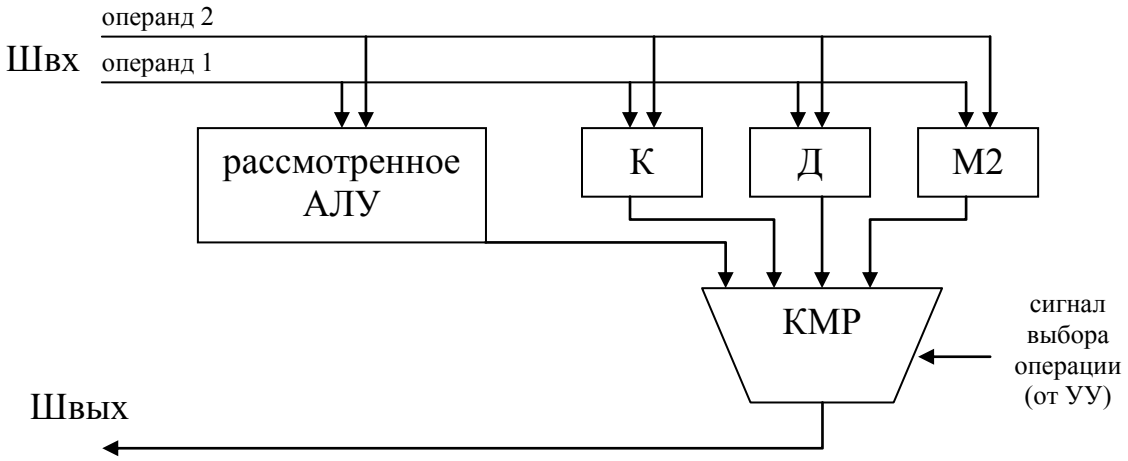


Выполнение операций умножения и деления в АЛУ возможно при реализации соответствующих микропрограмм, которые выполняются последовательностью микроопераций по схеме:

РО – регистр операнда
АКК – аккумулятор
ДШ – дешифратор микрооперации
РР – регистр результата

АЛУ обладает магистральной (подключается к шине входа и шине выхода). Такая структура позволяет обеспечить модульный принцип организации многофункционального АЛУ.

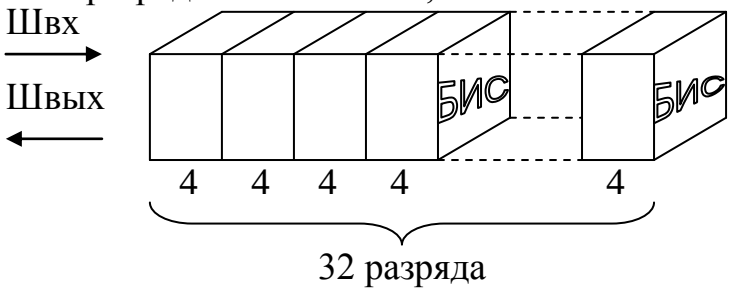
Например, дополним структуру АЛУ блоками, реализующими основные логические операции:



К, Д, М2 – комбинационные схемы, реализующие поразрядные операции конъюнкции, дизъюнкции и mod2 соответственно.

КМР – схема коммутатора результата.

В настоящее время АЛУ могут выполняться в виде одной или нескольких БИС. Возможен секционный принцип построения многоразрядного АЛУ из 2х, 4х, 8-ми разрядных БИС АЛУ, т. е.

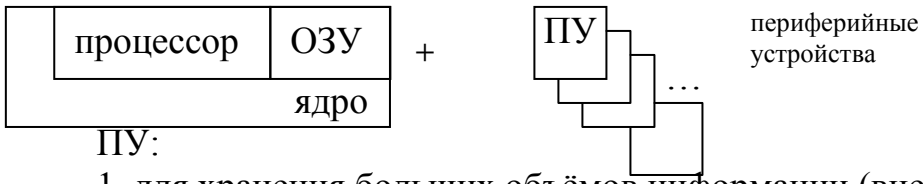


При таком построении АЛУ следует предусмотреть организацию цепей межсекционных переносов и заемов.

2.4. Системы ввода-вывода информации.

I. Модульная организация ЭВМ и интерфейсы.

вспомним состав ЭВМ



ПУ:

1. для хранения больших объёмов информации (внешние ЗУ)
2. для ввода-вывода (устройства ввода-вывода)

Современные ЭВМ – модульный принцип организации, т. е. ЭВМ из набора блоков: Процессор, Оперативная Память, Периферийные Устройства и др.

Модуль – законченный функционально и конструктивно элемент.

Преимущества модульной организации:

- гибкость структуры (наращивание или усечение)
- оперативный ремонт (замена блоков)
- надёжность (поблочное резервирование).

Связь модулей (устройств ЭВМ) друг с другом осуществляется с помощью сопряжений называется в вычислительной технике интерфейсом.

Передача информации из периферийного устройства в ядро ЭВМ называется операцией ввода.

Передача информации из ядра ЭВМ в периферийное устройство называется операцией вывода.

Характеристики ввода-вывода влияют на производительность и эффективность ЭВМ.

Интерфейс – совокупность шин, сигналов, элементов схем и алгоритмов, предназначенная для обмена информацией между устройствами.

Шины:

- информационные (передача команд, адресов и данных)
- идентификации типа информации (передаваемой по информационным шинам)
- управляющие (синхронизация, инициирование и завершение передачи).

Характеристики интерфейса:

- информационная шина (количество бит передаваемых параллельно)
- скорость обмена информацией
- максимальное расстояние
- связность:
  - односвязный И
  - многосвязный И (обмен по нескольким независимым путям)

II. Способы организации и основные структуры систем ввода-вывода.

Существует два основных способа организации и передачи данных между оперативной памятью и периферийными устройствами:

1. программно-управляемая передача (обмен программным путём)
2. прямой доступ к памяти (обмен при помощи аппаратных средств)

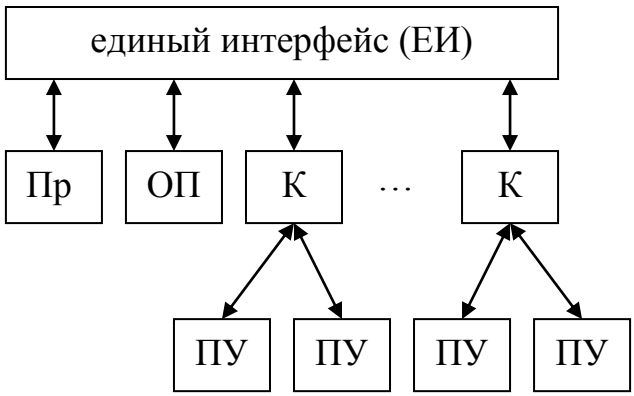
1. Передачей данных управляет процессор, отвлекаясь на время этой операции от решения задачи. Использование процессора неэффективно. Скорость обмена ниже возможностей ЗУ на дисках и бобинах.

2. Автономное от процессора установление связи и передача данных между ОП и ПУ. Устройство управляющее доступом к памяти – контроллер.

Основные структуры систем ввода-вывода.

А) Структура с одним общим интерфейсом (общая шина).

Модули соединены по схеме:



Контроллер – устройство управляющее ПУ.

Функции:

- синхронизация работы ПУ с другими устройствами
- буферизация информации (временное запоминание информации на время обмена)

- преобразование сигналов интерфейса в последовательность сигналов, обеспечивающих работу конкретного ПУ.

Каждому ПУ необходим свой контроллер.

Правила обмена информацией при едином интерфейсе.

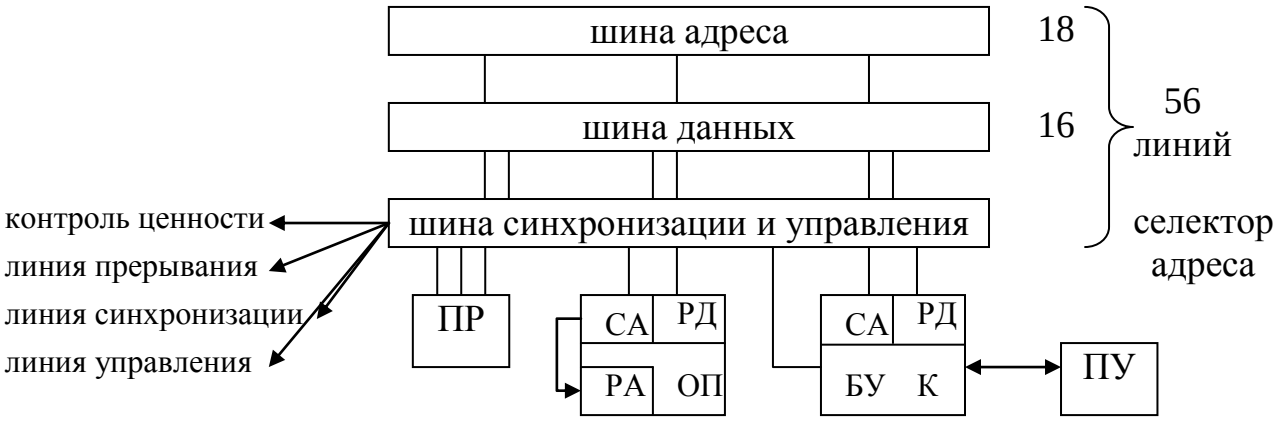
1) информация передаётся словами, равными ширине интерфейса
2) в каждый момент обмен только между одной парой устройств – источником и приёмником информации
3) прямой обмен между ПУ и ПУ невозможен
4) конфликт при одновременной необходимости передать между несколькими устройствами решается на основе приоритетов

Более высокий приоритет у устройств с более высоким быстродействием

ЕИ эффективен в малых и микро ЭВМ с коротким словом (1-2 байта), небольшим количеством ПУ, умеренной производительностью.

Например в СМ-4

организация «общей шины»



БУ – блок управления

По второму правилу обмена всегда обменивается лишь пара устройств активное и пассивное.

Процесс обмена.

Активное устройство инициирует обмен:

- захватывает шины интерфейса
- передаёт адрес пассивного устройства и сопровождает его синхроимпульсом
- одновременно передаёт код операции задающей направление обмена
- при выводе передаёт данные

Пассивное устройство:

- распознаёт свой адрес (селектор адреса)
- анализирует код операции и подключает буф. регистр к шинам данных (при вводе) или подключает буф. регистр через усилители к шинам данных (при вводе)

Буф. регистр рассчитан лишь на 1 слово.

По окончании синхроимпульса активное и пассивное устройства отключаются от шин интерфейса. Обмен закончен.

Прямой обмен между ПУ и ПУ невозможен.

Два типа пар:

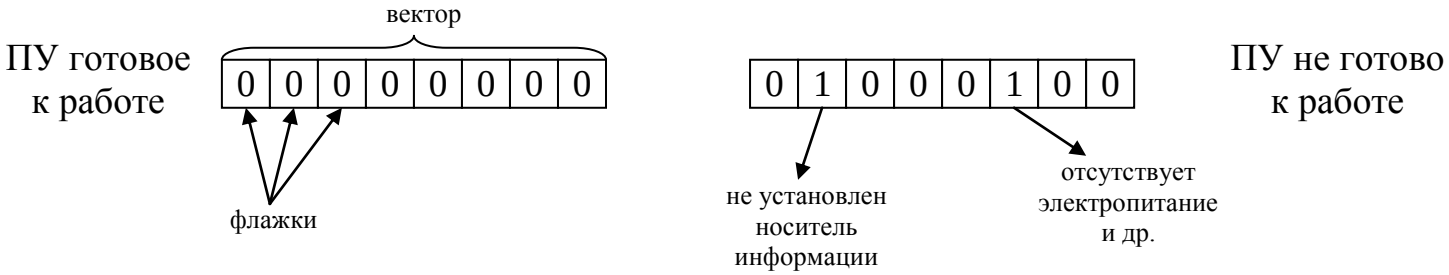
ПР* и ПУ

ОП и ПУ*

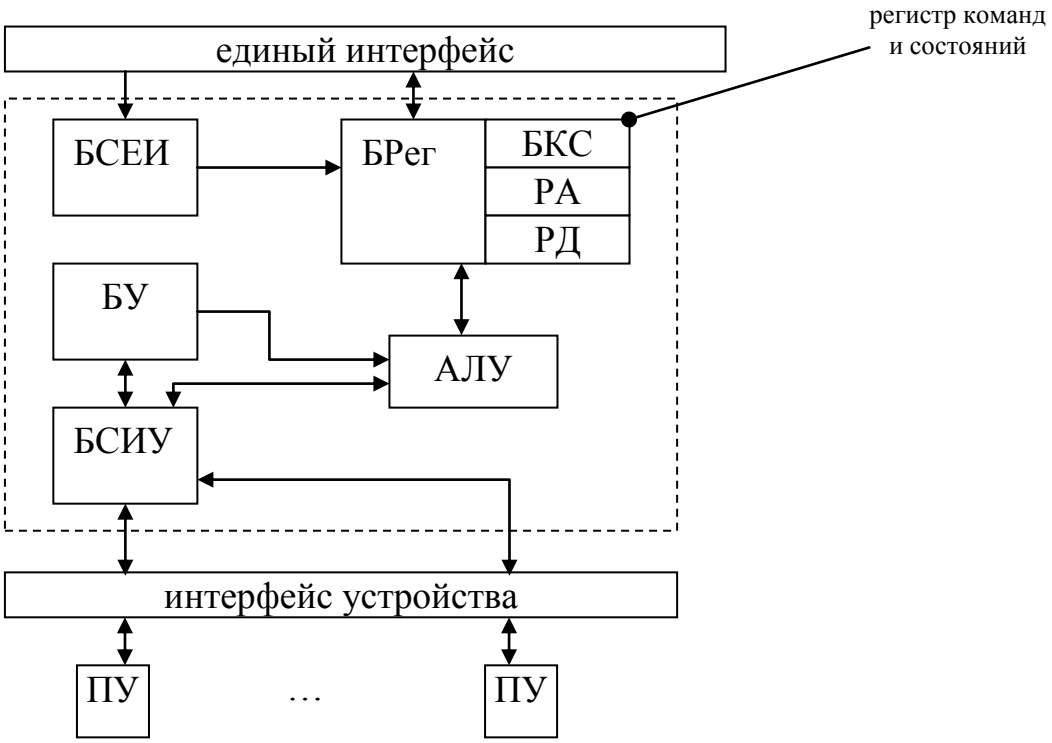
* - активное устройство

Что же делает контроллер?

Он анализирует состояние ПУ по двоичному <u>вектору состояния</u>.



<u>Структура контроллера ПУ</u>



БСЕИ – блок сопряжения с интерфейсом

БСИУ – блок сопряжения с интерфейсом устройств все регистры БРег доступны

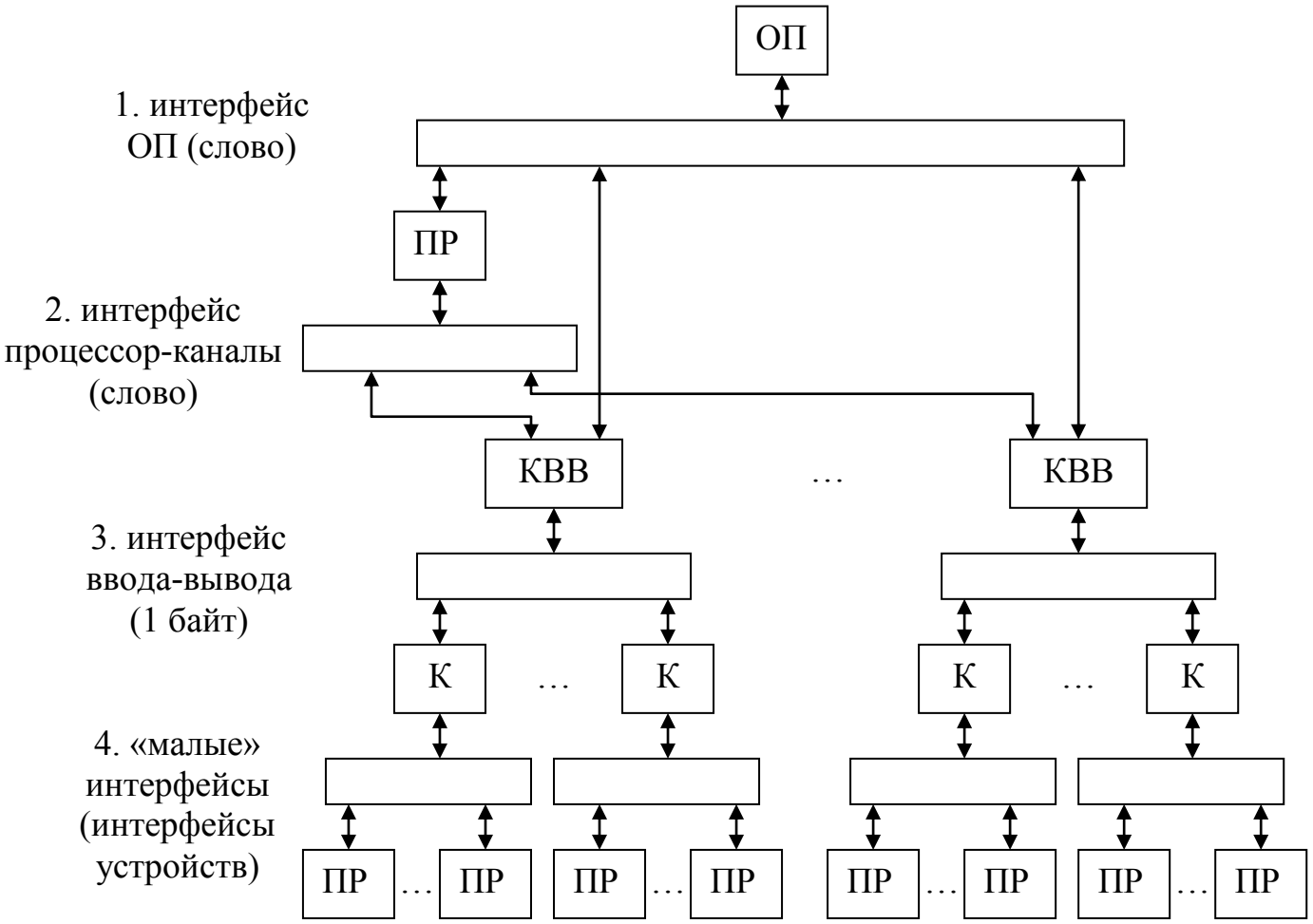
БСЕИ содержит:

- селектор адреса устройства (схемы совпадения (СС)), если это адрес одного из регистров БР то СС выдаёт 1.
- дешифратор кода операции обмена
- формирователь синхроимпульсов

В простых контроллерах БУ и АЛУ отсутствуют.

Б. <u>Иерархическая структура построения ЭВМ из модулей.</u>

(в ЭВМ общего назначения (ЕС ЭВМ))



ККВ – программно-управляемый специальный процессор ввода-вывода.

Обменом руководит специальная программа: супервизор ввода-вывода.

плюс – основной процессор разгружен от управления вводом-выводом (благодаря КВВ)

минус – нет однородности в структуре потоков и формах передаваемых данных.

1. и 2. быстродействующие интерфейсы (слово обмен или двойное слово)

3. информация представляется 1 или 2 байтами. Унифицированы.

4. не унифицированы

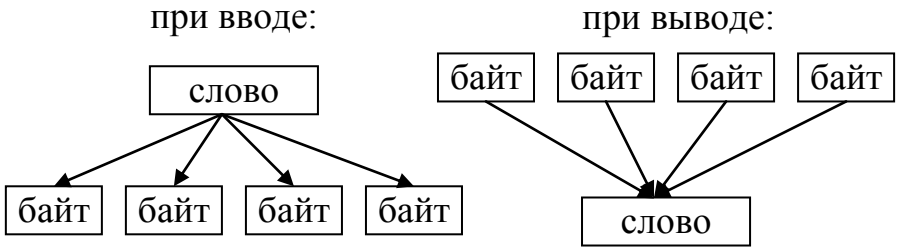
Процедура обмена.

1. Процессор получив команду ввода-вывода передаёт её в канал (КВВ)

2. КВВ из фиксированной ячейки памяти канала выбирает начальный адрес начальной программы ввода-вывода и выполняет её. Каждая команда программы задаёт параметры одной операции обмена:

- обмена
- начальный адрес СЗУ
- число

3. КВВ выполняя команды инициирует работу ПУ и последовательно читает или записывает слова информации, обращаясь в ПО.



плюс – при большом числе ПУ экономиться оборудование за счёт централизации в КВВ сложных функций по обслуживанию ПУ

минус – процессор не так оперативно реагирует на различные ситуации во внешней среде.

Скорость выдачи данных из ОП – $(1 \div 10) \cdot 10^6 \frac{\text{байт}}{\text{сек}}$

Скорость выдачи данных из ПУ –

быстродействующие (диски, барабаны) $(1 \div 10) \cdot 10^6 \frac{\text{байт}}{\text{сек}}$

медленнодействующие (перо, АЦПУ) до $(1 \div 2) \cdot$ тысячи$\frac{\text{байт}}{\text{сек}}$

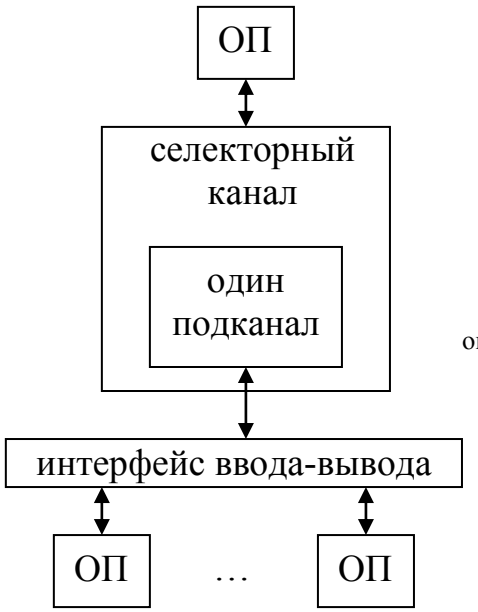
В. Основные типы и структуры КВВ.

В зависимости от соотношения быстродействия ОП и ПУ в КВВ реализуют два режима работы:

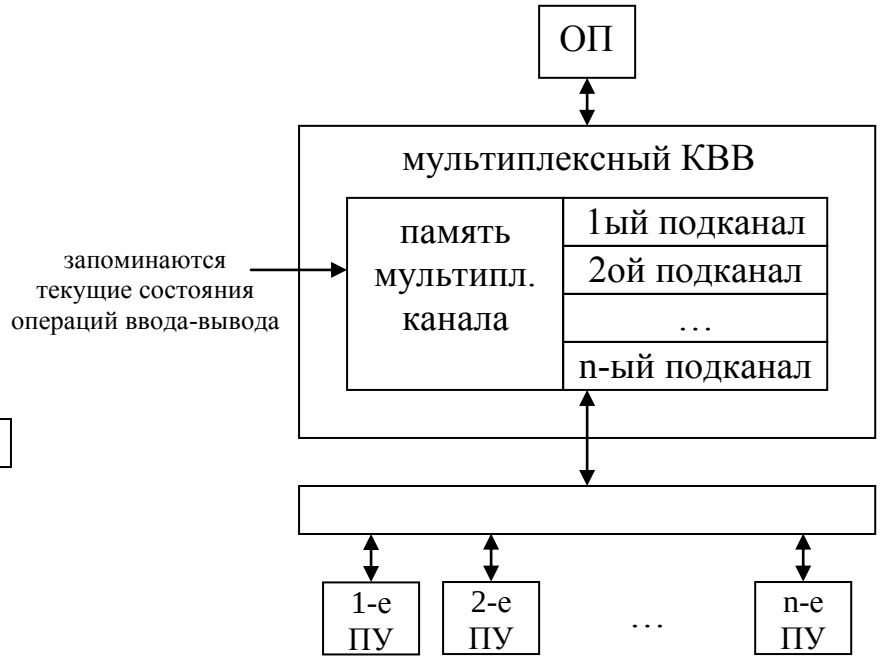
монопольный – ПУ монополизируется канал на всё время передачи данных

разделения времени (мультиплексирования) – каждое из ПУ связывается с каждым на короткие сеансы связи. Передача порций по 1 или нескольким

селекторный КВВ



мультиплексный канал КВВ



Подробно структуры КВВ в Каган Б. М. «ЭВМ и системы»
Характеристики КВВ ЭВМ ЗВ-1045:
до 2х мультиплексных каналов (40-120 кбайт/сек)
до 5ти селекторных каналов (500-1500 кбайт/сек)
Общая суммарная пропускная способность всех КВВ 5Мбайт/сек

3. Организация систем.
3.1. Принципы построения оптимальных систем.

Любую систему (аппаратную или программную) можно представить в виде системы уравнений, параметры которой описывают ее функционирование (1).

$$\begin{cases} F_{optBT}=f_1\{x_1, x_2,\ldots, S_1,N,\ldots\} \\ F_{optBнУстр.}=f_2\{x_1, x_2,\ldots, S_2,N,\ldots\} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ F_{opt\,n}=f_n\{ x_1, x_2,\ldots, S_n,N,\ldots\} \end{cases} \quad (1)$$

F_{opt} – отдельные функции системы. Если речь идет об аппаратно-программном комплексе, то могут рассматриваться, например, функционалы, описывающие характеристики средств вычислительной техники (ВТ), внешних устройств (ВнУстр) и т.д. Структура такого рода системы представлена на рис. 2.

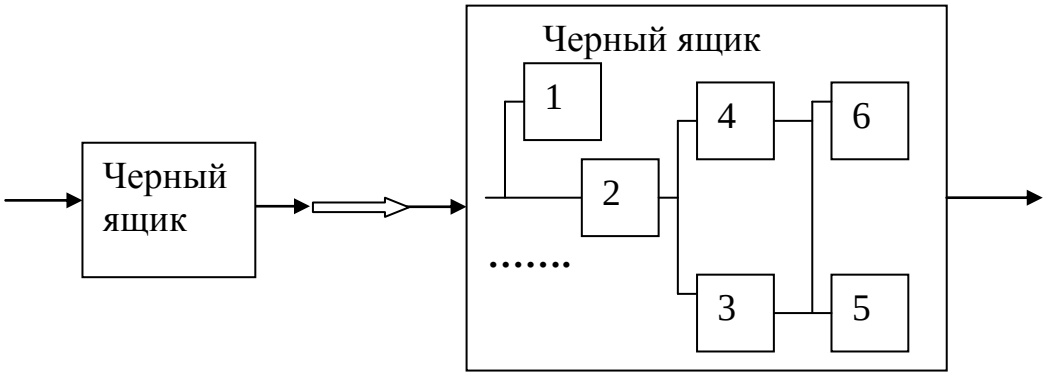


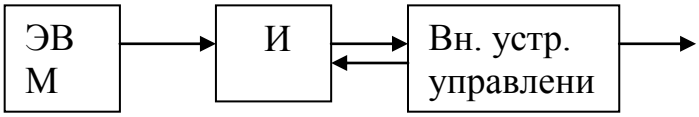
рис.2

Подсистемы 1, 2 и т.д. реализуются таким образом, чтобы число связей между элементами было минимальным, а количество выполняемых функций каждым элементом максимально.

Решение системы уравнений (1), в общем виде, заключается в поиске ее частных производных. В общем случае, число параметров такого рода систем стремится к бесконечности и число частных производных, соответственно, тоже. Например, для средств вычислительной техники параметры могут быть следующего типа: быстродействие (x_1), объем оперативной памяти (x_2), объем постоянного запоминающего устройства (x…),…,характер изготовления ЭВМ (промышленного или бытового типа), потребляемая электроэнергия, параметры надежности, стоимость, и т.д. Естественно, что какие-то из этих параметров носят первостепенный характер, а какие-то – второстепенный. Все зависит от конкретно решаемой задачи.

С этой точки зрения, основной задачей разработчиков и пользователей при построении и использовании оптимальной системы является задача поиска квазиоптимальной системы. То есть необходимо минимизировать число функций ($F_i \leq P$) и параметров ($x_1 \leq n$) системы уравнений (1), что приведет к практически возможному решению данной системы. При этом необходимо осознавать, что любые вводимые ограничения приводят к тем или иным искажениям конечного результата.

В зависимости от конкретно решаемой задачи можно использовать универсальные или специализированные средства. Универсальные средства, по сравнению со специализированными, уменьшают стоимость системы, позволяют решать более широкий круг задач, облегчают перенастройку, увеличивают время решаемой задачи и т.д. Все вышесказанное в равной мере относится, как к аппаратным, так и программным комплексам. Остановимся более подробно на аппаратных системах.
Они могут быть реализованы следующим образом:
1. С встроенными устройствами.
Пример: сверлильный станок с числовым программным управлением. Его функции ограничены типоразмерами обрабатываемой детали, набором сменного оборудования, качественными и количественными характеристиками обрабатываемого изделия и т.д.
То есть, такая система хороша для специализированных задач, но если задача меняется, то все приходится менять.
2. С внешними устройствами управления.



На рисунке представлена система, которая позволяет осуществлять взаимодействие одной ЭВМ, через интерфейс (И) и внешнее устройство управление, с одним внешним объектом (О). Если в ЭВМ присутствует более одного свободного порта, то число подключаемых объектов может быть увеличено.
Такая система позволяет использовать специализированные и универсальные средства, но ресурсы машины, при этом, полностью не реализуются. Объясняется это тем, что быстродействие ЭВМ, как правило, во много раз превосходит возможности канала ввода/вывода информации (КВВИ), а компьютер в оперативном режиме решает задачи О, то есть неэффективно тратит свои ресурсы.
С точки зрения быстродействия такая система близка к системе 1-го типа.

3. На базе промышленных контроллеров (ПК).
Простейшая система такого типа представлена ниже на рисунке. В ее состав входят ЭВМ, ПК, состоящий из контроллера и магистрали крейта, и внешних устройств (У), подключаемых к посадочным местам магистрали.

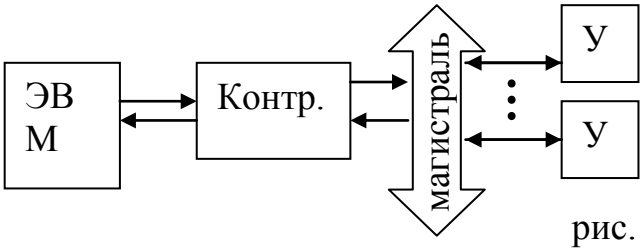


рис. 3

В такой системе, по сравнению с предыдущими, увеличивается число абонентов, но уменьшается скорость обмена данными (оперативность принятия решения). Количество внешних устройств, подключаемых к ЭВМ, определяется числом посадочных мест магистрали. Если внешний объект очень сложен и требует большого числа посадочных мест, то на базе системы рис.3 проектируются многомашинные и/или многоконтроллерные системы. При этом, обязательным условием работоспособности системы является обеспечение реального масштаба времени функционирования объекта.

Несмотря на внешнюю простоту структуры рис.3, она является основополагающей для всех ведущих фирм мира, работающих в области автоматизации производства, диагностики, контроля качества продукции и т.п.

Самым «узким» местом систем, связанных с обменом информацией, является канал приемо-передачи информации (рис. 4). Определяется это тем, что ЭВМ может обрабатывать и вырабатывать огромное количество информации. Внешний объект или процесс, в свою очередь, может создавать или поглощать еще больший объем, а канал приемо-передачи сродни водопроводной трубе – диаметр трубы определяет возможное количество передаваемой воды.

рис. 4

Необходимо также обратить внимание на то, что при решении задач систем автоматизированного управления производством необходимо реализовывать следующие аппаратные и программные функции:

1. Сбор информации
2. Хранение, обработка, архивирование и предоставление информации и т.д.
3. Управление технологическими процессами

Управление должно осуществляться в реальном масштабе времени, то есть в темпе протекания эксперимента.

Желание создать оптимальную систему привело разработчиков и пользователей к выработке основных принципов построения систем автоматического производства. Некоторые из них приведены ниже:

1. Блочно-модульный принцип.

2. Открытость архитектуры (возможность увеличения количества модулей и функций).
3. Унифицирование и агрегатирование.
4. Наличие самоконтроля на работоспособность.

Первый принцип позволяет осуществлять оперативную замену вышедшего из строя модуля и строить системы по типу игрушечного конструктора: ставится задача, имеется набор стандартных модулей и в заданном пространстве связей и объема осуществляется попытка получения конечного результата при известном входном воздействии (классическая задача «черного ящика»).
Открытость архитектуры – сродни желанию покупателя на рынке: побольше, получше и подешевле. Мы хотим иметь такую систему, чтобы любое изменение ее архитектуры и связей, по желанию пользователя, гарантировало бы ее работоспособность и обеспечивало бы желаемый результат. Реализация этого принципа на практике осуществляется путем ограничений сверху для количества модулей и функций. Например, число модулей не более 24-х или 48-и и т.д. – такое ограничение может быть введено для модулей крейта КАМАК, где в одном крейте число посадочных мест – 24.
Очень важным для практики является третий принцип. В мире существует очень большое количество фирм выпускающих однотипную аппаратуру (программное обеспечение). И если каждая фирма будет пользоваться своими стандартами на их выпуск (что происходило в нашей стране лет 10-15 назад и более), то пользователь не может безболезненно осуществлять замену отдельных элементов одной фирмы на другую. А предпосылок для этого очень много: банкротство фирм, невозможность использования отдельных элементов в конкретных климатических условиях и т.д. Этими причинами обусловлено то, что подавляющее число стран мира старается придерживаться общепринятых мировых стандартов. Рассмотрим несколько примеров. В качестве одной из составляющих для принятия решения о работоспособности турбоагрегата электрической станции используется информация о его вибрационном состоянии. Для сбора первичной информации могут быть использованы датчики (Д) перемещения. Их установка на турбоагрегат – дело трудоемкое и дорогостоящее, поэтому типоразмеры и электрические параметры этих датчиков у различных фирм конкурентов одинаковые. Если вдруг появится какая-то фирма, выпускающая аналогичные датчики с другими характеристиками, то ей понадобиться очень много средств, чтобы убедить владельцев электрических станций оснащать турбоагрегаты именно ее продукцией. Для согласоваия сигналов после датчиков по уровню, виду и качеству с последующей аппаратурой используют вторичные преобразователи (ВП). Их входные и выходные сигналы унифицируют таким образом, чтобы имелась возможность их использования независимо от типа датчика (температурный, давления, перемещения,…) – лишь бы выходной сигнал Д соответствовал входному ВП. И т.д.
Современные системы (и аппаратные и программные) необходимо обеспечивать самоконтролем на работоспособность. Обусловлено это тем, что пользователь очень часто не в состоянии оценить достоверность получаемой информации. Что в

свою очередь может привести к непоправимым, а иногда и катастрофическим последствиям. На рис.5 приведен пример структуры с самоконтролем для датчиковой аппаратуры. В качестве сигнала самоконтроля используется постоянная состовляющая (при этом считается, что измеряется чисто переменный сигнал), Если на выходе Д отсутствует постоянная составляющая, то переменная составляющая является шумом и канал не работоспособен.

Система с самоконтролем.

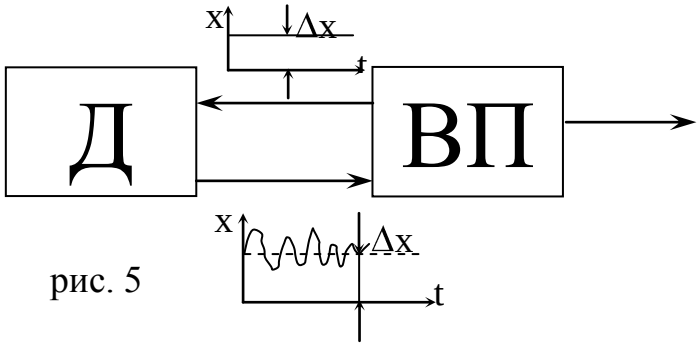


рис. 5

3.2. Реализация технических систем на базе контроллеров

Контроллеры – это устройства, позволяющие увеличить число подключаемых абонентов к ЭВМ и имеющие возможности предварительной обработки данных. Контроллеры можно разбить на больших класса: для научных исследований (КНИ) и промышленные контроллеры (ПК). Их названия отражают качество изготовления, а различия такие же как для бытовой и промышленной аппаратуры.
Все контроллеры изготавливаются в соответствии со стандартами на них разработанными. Для КНИ наибольшее распространение получила аппаратура CAMAC, а для ПК – Евростандарт.
Для всех контроллеров общим является следующее:

1. Они имеют три функциональные части: крейт (где размещаются функциональные модули); блок питания (может использоваться один на несколько контроллеров); интерфейс, осуществляющий связь с ЭВМ (называемый контроллером крейта – КК);
2. Возможность строить многоконтроллерные системы;
3. Основными функциями модулей является: сбор; хранение; диагностика; частичная обработка информации; выдача управляющих сигналов и т.д.;
4. Возможность организации межмодульных связей.

В крейте имеется определенное число (в зависимости от стандарта) посадочных мест, в которых размещаются модули. Модули могут занимать одно, два, три (как правило, не более) посадочных мест. Для интерфейса отводятся жестко закрепленные места. В зависимости от типа контроллера крейта они могут соединяться с ЭВМ той или иной архитектуры. Размеры модуля имеют стандартные габариты и любой из модулей может быть установлен в любое посадочное место. Промышленные контроллеры используются там где требования по влияющим факторам особые (повышенная влажность, вибрация, радиоактивность и т.д.).

На рис. 6 представлена структурная схема использования контроллеров в системах автоматизации производства. Где ШУ – это штатные устройства ЭВМ, такие как: процессор, память, дисплей с интерфейсом и т.д. В зависимости от типов контроллеров схема будет видоизменяться.

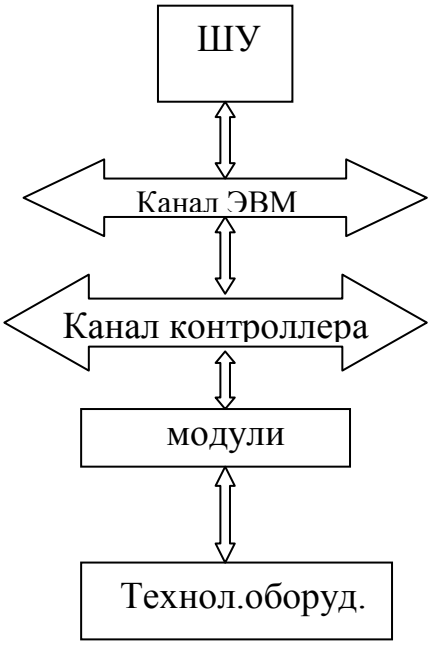
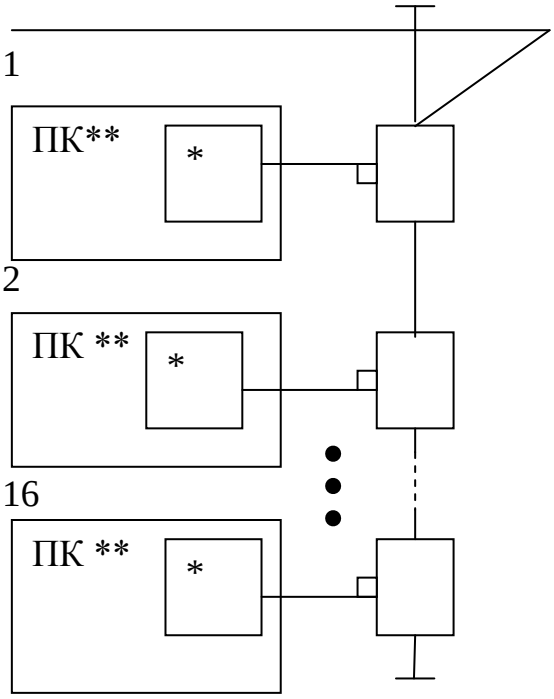


рис. 6

Группа ПК, объединенная последовательно магистралью данных.
Магистраль данных



Группа ПК объединенная 4-мя магистралями данных:

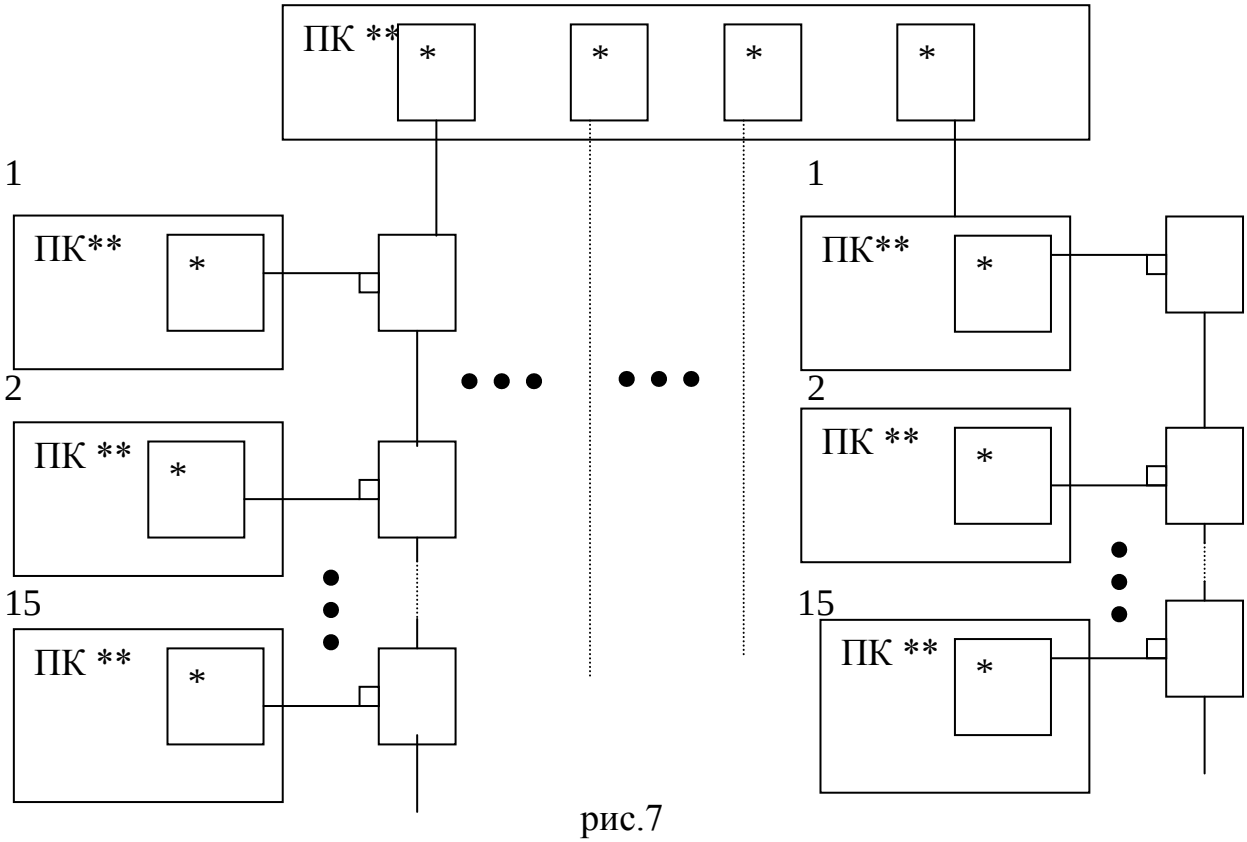


рис.7

Отечественным аналогом Евростандарта являются контроллеры Микродат, которые позволяет объединять ПК в группы, строить иерархические системы, локальные сети (рис 6 и рис. 7). Где:

*-модуль последовательного ввода/вывода.

**-ПК128 или ПК248.

Объединения ПК, изображённые на рис. 6 и рис. 7 могут охватывать территорию до 8 км.

Если существует необходимость охватить большую территорию, то используются устройство связи объектом (УСО), схема связи которых представлена далее на рисунке.

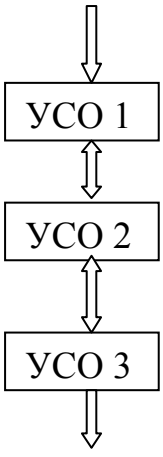


рис. 8

Таким соединением можно охватить территорию с неограниченной площадью, т.к. происходит ретрансляция. Недостатком является то, что при неограниченном росте время выполнения будет неограниченно и при выходе одного из устройств

из строя будет нарушена связь. Следует помнить, что для промышленных систем основным фактором является обеспечения функционирования системы в реальном масштабе времени.

Аппаратура для научных исследований.

В нашей стране разрабатывались и изготовлялись системы, в основном, трех стандартов: CAMAC (или КАМАК), ВЕКТОР, МЭК-685. Последний стандарт ориентирован на приборостроение и мы на нем останавливаться не будем. ВЕКТОР является модификацией CAMACa. В настоящее время практически не используется. Основной причиной этого является то, что модули, изготавливаемые в других странах, не совместимы с CAMACом.

КАМАК представляет собой систему электронных модулей, предназначенных для построения цифровых измерительных установок, управляемых от ЭВМ. Первоначально разрабатывался для управления работой синхрофазотрона. В дальнейшем использовался более широко. В частности, в Санкт-Петербурге на его базе был реализован диагностический комплекс для исследования технических характеристик тракторов «Кировец». Общий вид КАМАК представлен на рис. 9.

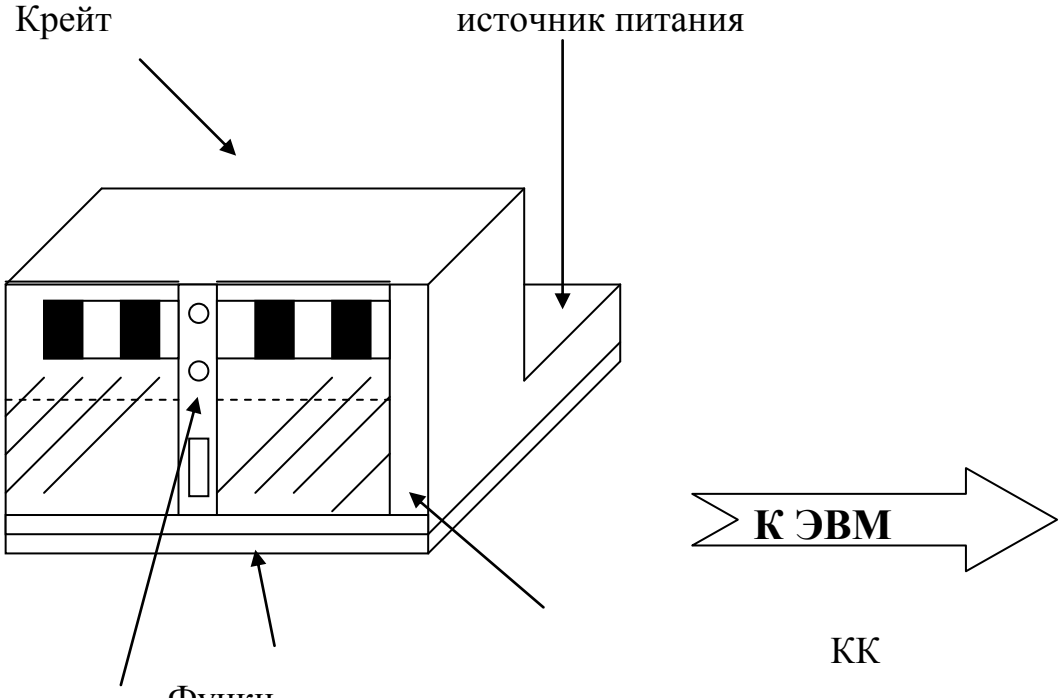


рис. 9

КАМАК удачно объединяет в себе, с одной стороны, богатый набор электронных функциональных модулей самого разнообразного назначения (усилители, счетчики, таймеры, аналого-цифровые преобразователи, запоминающие устройства и т.д.), а с другой стороны – средства связи всей этой аппаратуры с ЭВМ, для чего предусмотрен специальный управляющий модуль –контроллер КАМАК. Характерным для системы КАМАК является наличие унифицированного канала передачи данных (магистрали) между отдельными модулями и контроллером. И модули, и контроллер имеют выход на магистраль, по линиям

которой происходит обмен рабочей и служебной информацией, а также питание модулей: контроллер кроме того связан с ЭВМ (рис 10). Всеми процессами на магистрали управляет (по командам от ЭВМ) контроллер, однако если в модуле возникла ситуация, требующая вмешательства ЭВМ, модуль может послать в контроллер запрос на обслуживание и инициировать тем самым конкретную программу обработки.

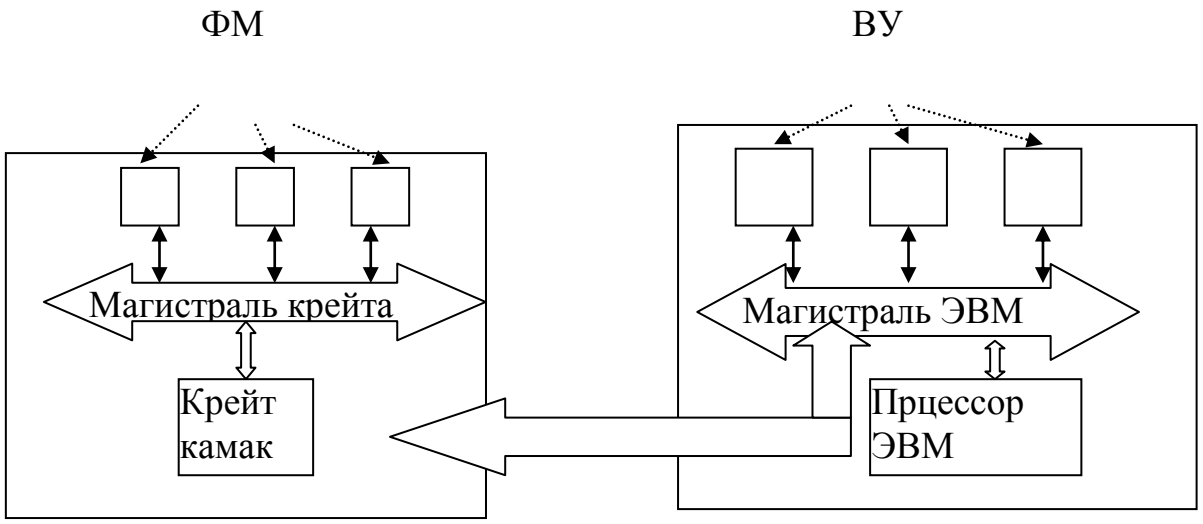


рис. 10

Стандартизация модулей по конструкции, способу подсоединения к магистрали, характеристикам электрического питания, параметрам входных и выходных сигналов позволяет быстро собирать и модернизировать экспериментальные установки, комплектуя их требуемыми модулями, а единая система команд существенно облегчает разработку алгоритмов управления системой. При этом компоновка любой системы сводится по существу к составлению программы взаимодействия ЭВМ с модулями (порядок опроса состояния модулей, записи и съема информации и т.д.), технические же вопросы согласования модулей друг с другом или с контроллером отпадают ввиду стандартизации системы.

Конструктивной основой системы КАМАК является специальный каркас – крейт, содержащий 25 станций – направляющих, по которым в крейт вдвигаются модули. В зависимости от сложности модуль может иметь единичную ширину 17,2 мм и занимать одно место в крейте либо ширину, кратную указанной. Контроллер обычно занимает два крайних правых места. Таким образом, в крейте может размещаться до 23 различных модулей. Каждый модуль имеет стандартный 86-контактный разъем для подсоединения к магистрали. Разводка линий магистрали по контактам разъемов всех станций (кроме крайней правой, принадлежащей контроллеру) выполнена единообразно, что позволяет устанавливать любые модули на любые места крейта.

На рис. 11 приведено схематическое изображение магистрали крейта. Большая часть линий магистрали – параллельные линии, соединяющие одноименные контакты всех разъемов; сигналы, передаваемые по этим линиям, доступны всем модулям.

Рис.11

Рабочая информация в системе КАМАК передается 24-разрядным двоичным параллельным кодом, для чего служат 24 линии чтения R (передача из модулей в контролер) и 24 линии записи W (передача данных из контролера в модули). Поскольку в каждом модуле могут размещаться несколько функциональных узлов (например, несколько счетчиков) и, кроме того, еще имеются многочисленные обслуживающие схемы, для адресации к элементам модуля служат 4 линии субадреса A, по которым номер узла в модуле или его субадрес передается также двоичным параллельным кодом. Всего, таким образом, в каждом модуле может использоваться до $2^4 = 16$ субадресов.

В процессе обращения контроллера к модулю может быть задано выполнение различных операций – чтение или запись информации, опрос состояния регистра и т.д. Для передачи кода операции предусмотрены 5 линий функций F, что дает возможность использовать до 32 различных функций. Значения функций стандартизированы, например, функция F(2) никогда не используется для записи информации в регистр, а только для чтения его содержимого с последующим его сбросом. Назначение же регистров и характер содержащейся в них информации зависят от функционального назначения и конкретной схемы модуля.

Группа параллельных линий отводится для управления и передачи служебных сигналов. Сюда относятся линии (и соответственно сигналы) Z, C, I, B, Q, S1, S2. Некоторые из этих сигналов (B, S1, S2) генерируются контроллером или модулями автоматически в процессе обмена информацией по магистрали, на них нельзя воздействовать программным образом; другие сигналы устанавливаются, снимаются либо контролируются программно, и их назначение необходимо понимать для правильного составления программ управления.

Например, сигнал C (Сброс) вызывает сброс регистров модулей крейта.

Сигнал X (Команда принята) вырабатывается модулем всякий раз при получении им «законной» команды, которую данный модуль в состоянии выполнить. Нулевое значение сигнала X = 0 указывает на наличие неисправности (например,

отсутствие адресуемого модуля) или серьезной ошибки в программе обслуживания (в модуль послана команда, которую он не может выполнить).
Две группы линий (N и L) служат для установления связи контроллера с определенными модулями. В отличие от остальных линий магистрали линии N и L имеют радиальный характер; каждый модуль связан с контроллером индивидуальной парой линий N и L. Когда контроллер генерирует команду обращения к какому-то модулю, он устанавливает соответствующую функцию КАМАК на линиях F, требуемый адрес на линиях A и возбуждает линию N, соответствующую адресуемому модулю. Сигналы F и A поступают во все модули. Однако воспринимает их только тот модуль, который подсоединен к возбужденной линии N, т.е. модуль, установленный на станции с номером N.
Если в модуле создалась ситуация, требующая вмешательства ЭВМ (АЦП преобразовал входной сигнал в код, счетчик зарегистрировал заданное число импульсов и т.д.), модуль может послать в контроллер запрос на обслуживание, установив логическую 1 на линии. Обычно возбуждение линии L (L-запрос) приводит к прерыванию текущей программы и переходу на программу обработки прерывания от данного модуля. Поскольку от каждого модуля в контроллер идет индивидуальная линия L, контроллер, получив запрос, может определить, из какого именно модуля он пришел.
Как уже отмечалось, каждая команда, с которой контроллер обращается к какому-то модулю, состоит из трех элементов: функции F, субадреса A и номера адресуемого модуля N. Управление аппаратурой КАМАК и заключается в выполнении последовательности команд NAF (команд КАМАК), соответствующей заданному алгоритму функционирования установки. Требуемая последовательность команд NAF записывается в виде машинной программы.
Как и в любой программе реального времени, последовательность выполняемых операций не является фиксированной, а определяется ходом эксперимента. Например, получив от модуля АЦП запрос на обслуживание, ЭВМ выполняет программу приема из модуля и записи в оперативную память подготовленного модулем кода входного сигнала. Если, однако, при этом выясняется, что общий объем накопленной информации достиг заданного значения, ЭВМ выполняет программу выключения модуля АЦП – блокировки его входа, запрещение генерации им L- запросов и т.д.

Система сбора данных и диагностики на базе ПК.

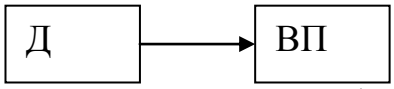
Рассмотрим систему сбора данных и диагностики на примере выпускаемой фирмой «Филипс», которая специализируется на изготовление и разработке блоков, модулей, устройств и систем сбора данных и диагностике, сопрягаемых между собой и применяемой в условиях промышленной эксплуатации объектов. Для примера отметим, что фирма «Брюль и Къер» производит аналогичную продукцию, но в конструктиве пригодном для использования в научных исследованиях.
Фирма «Филипс» для своих блоков, модулей и т.д. использует стандарт Евромеханика. Аналогичные стандарты существуют и у нас в стране: БУК – бло

унифицированный комбинационный, БУК Б. БУК БМ (предприятия общепрома); Рябина (на предприятиях средмаша). Из контроллеров применяемых в атоматизации производства наиболее близок (по габаритам) к стандарту Евромеханика контроллер микроДАТ.

Система сбора данных и диагностики состоит из множества параллельных каналов и следующих уровней преобразования информации (каждый из которых функционально закончен и может быть использован без вышестоящего уровня): датчиков (первичных преобразователей); вторичных преобразователей; каналов ввода- вывода; ЭВМ низшего звена; ЭВМ верхнего уровня.

Вся система построена по блочно–модульному принципу, все элементы унифицированы и каждый может использоваться автономно.

Далее на рисунке представлена простейшая измерительная систем (называемая монитором), которая состоит из датчика и вторичного преобразователя



В конструктивном исполнении фирма «Филипс» отдает предпочтение раздельному изготовлению датчика и вторичного преобразователя (хотя работы ведутся и в противоположном направлении). Раздельное изготовление датчика и ВП позволяет уменьшить число проводов в кабеле и вероятность сбоев при передаче информации. В зависимости от уровня сигнала, вырабатываемого датчиком, возможно удаление его от вторичного преобразователя на то или иное расстояние. Для разных типов датчиков (перемещения, индуктивных, тензорезестивных, емкостных и т.д.) это расстояние колеблется от единиц до сотен метров.

Поскольку все элементы унифицированы: датчик – законченное изделие со стандартным креплением и унифицированными сигналами. Датчик (Д) и вторичный преобразователь (ВП) соединены между собой стандартным кабелем, имеющим унифицированные характеристики.

ВП имеет средства представления информации и унифицированные сигналы входа и выхода.

Вторичный преобразователь представляет, из себя законченный, конструктивный модуль со своими средствами визуального контроля текущей информации, структурная схема которого для одного канала представлена на рис. 12.

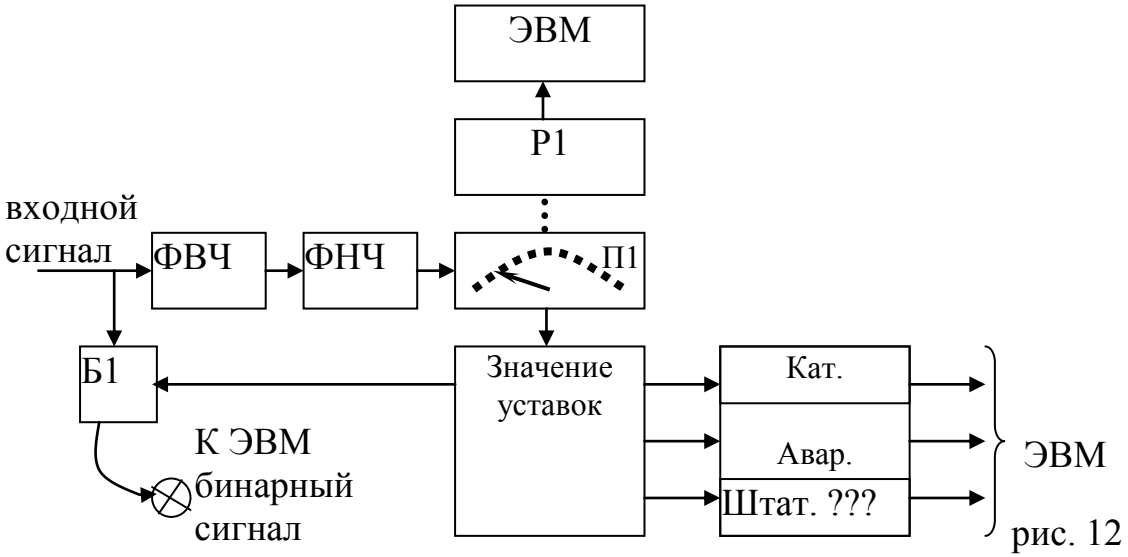


рис. 12

Основные задачи решаемые монитором:
- фильтрация входного сигнала (ФВЧ и ФНЧ);
- сравнение входного сигнала с уставкой, выставляемой потенциометрами (диапазон выставляется переключателем – П1);
- выработка бинарного сигнала (Б1) неисправности датчика с одновременной индикацией (зажженная лампочка) на передней панели ВП;
- выработка диагностирующих сообщений.

При использовании датчика перемещений такой монитор может быть использован в системах вибродиагностики турбоагрегатов. При этом в реальных системах число такого рода унифицированных каналов достигает – 1000. Например, в аппаратуре сбора данных и диагностики PR3000.

Если используются другие типы датчиков, то вторичный преобразователь перенастраивается путем переналадки диапазона и изменения значений уставок.

При подключении, к вышеуказанной структуре, ЭВМ меняется ее назначение:
- измерение;
- масштабирование и линеаризация входных сигналов;
- контроль;
- представление информации на экране дисплея;
- хранение информации в памяти;
- создание отчетов по хранимым данным.

Если требуется создать многоконтроллерную систему (охватить значительную территорию и большое число абонентов), то фирма «Филипс» предлагает использовать для этих целей архитектуру «Кольцо» (см. рис. 13). Интерфейс RS422 позволяет разносить устройства на расстояние до 500 м, а общее их число доводить до 16.

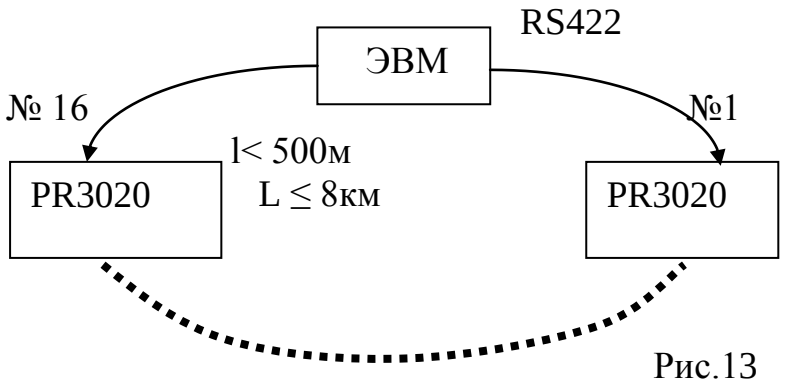


Рис.13

Отметим, что кроме технического обеспечения существует также программное обеспечение решающее различные задачи (слежение за динамикой сигнала).

СПИСОК ЛИТЕРАТУРЫ

1. Учебное пособие: «Архитектура ЭВМ и систем». В.Е.Баранов, Ю.В.Сотсков. СПб.: Изд-во Политехн. ун-та, 2007. 120

2. Букреев И. Н. и др. Микроэлектронные схемы цифро-вых устройств. Изд. 2-е и доп. М., «Сов. Радио», 1975. 368с.

3. Вуд А. Микропроцессоры в вопросах и ответах. М.: энергоатомиздат, 1985. 184с.

4. Каган Б. М. Электронные вычислительные машины и системы. М.: Энергоатомиздат, 1985. 522с.

5. Карлащук В.И. Электронная лаборатория на IBM PC. 2-е изд. – Москва, 2001. – 726 с.;ил.

6. Компьютеры: Справочное руководство. В 3-х т. Т.1. Под ред. Г. Хелмса М.: Мир, 1986. 416с.

7. Сергеев Н. П., Вашкевич Н. П. Основы вычислитель-ной техники. М.: Высш. шк., 1988. 311с.

8. Танебаум Э. Архитектура компьютера. 4-е изд. – СПб;Питер, 2003. – 704 с.: ил.