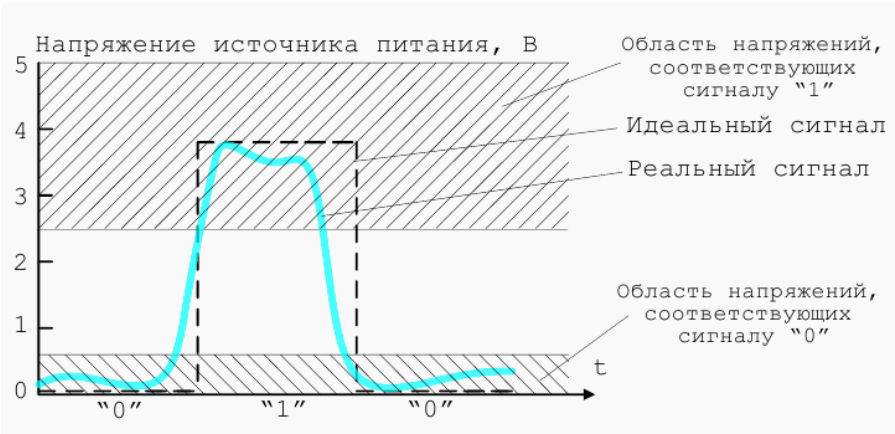
Экзаменационные вопросы по дисциплине Основы Вычислительной Техники. 2017-2018 уч. год.

1. Две формы представления информации.

Первая форма представления информации называется аналоговой или непрерывной (с помощью сходной величины – аналога). Количество значений, которые может принимать величина, представленная в такой форме бесконечно велико, даже если величина изменяется в ограниченном диапазоне. Отсюда названия – непрерывная величина и непрерывная информация. Слово непрерывность выделяет основное свойство таких величин – отсутствие разрывов, промежутков между значениями, которые может принимать аналоговая величина. Величина представляется в виде одного сигнала, пропорционального этой величине. Эта форма представления используется в аналоговых вычислительных машинах



Вторая форма представления информации называется цифровой или дискретной (с помощью набора напряжений, каждое из которых соответствует одной из цифр представляемой величины). Такие величины, принимающие не все возможные, а лишь вполне определённые значения, называются дискретными (прерывистыми). В отличие от непрерывной величины количество значений дискретной величины всегда будет конечным. Величина представляется в виде нескольких сигналов, каждый из которых соответствует одной из цифр заданной величины. Эта форма представления используется в электронных вычислительных машинах (ЭВМ).

Примером аналогового представления графической информации может служить, например, живописное полотно, цвет которого изменяется непрерывно, а дискретного - изображение, напечатанное с помощью струйного принтера и состоящее из отдельных точек разного цвета. Примером аналогового хранения звуковой информации является виниловая пластинка (звуковая дорожка изменяет свою форму непрерывно), а дискретного - аудиокомпакт-диск (звуковая дорожка которого содержит участки с различной отражающей способностью).

2. Способы представления дискретной информации. Системы счисления, используемые в вычислительной технике: двоичная, восьмеричная, десятичная, шестнадцатеричная, двоично-десятичная.

Каждое значение из набора исходных данных задачи, результатов её решения может быть представлено в ЭВМ в виде нескольких электрических сигналов, один из которых соответствует числу единиц в значении, другой – числу десятков, третий – числу сотен и т.д. Однако такое представление не является наилучшим с технических позиций. Устройство, предназначенное для обработки подобных сигналов, должно различать в каждом из них десять состояний. Значительно проще построить устройство, которое различало бы всего два состояния (его наличие или отсутствие). Это тем более целесообразно, т.к. существующие сейчас дешёвые устройства для ввода данных в ЭВМ также кодируют отдельные составляющие вводимой информации с помощью двух состояний.

Это натолкнуло создателей первых ЭВМ применение другой системы счисления при внутреннем представлении чисел в машинах: вместо привычной десятичной системы счисления была взята двоичная. 2СС также является позиционной СС, т.е. в ней значение каждой цифры зависит от позиции этой цифры в записи числа.

Существуют специальные термины, широко используемые в вычислительной технике: бит, байт и слово.

Двоичный разряд – бит

Восьмибитовая единица – байт

ЭВМ содержит большое количество ячеек памяти и регистров для хранения двоичной информации. Большинство этих ячеек имеет одинаковую длину n, т.е. они используются для хранения n бит двоичной информации. Информация, хранимая в такой ячейки, называется словом.

Удобная для использования в ЭВМ двоичная система счисления совсем неудобна для записи и чтения чисел человеком. Для сокращения трудоёмкости ручной обработки кодов чисел, команд широко применяют 8- и 16СС. В 8 СС используется 8 цифр (0-7), в 16СС – 10 цифр и 6 прописных букв (0-9, A-F). Т.к. основанием 8СС является $8=2^3$, то для перевода двоичных чисел в восьмеричные необходимо разделить двоичные числа на триады. Каждую из таких групп можно представить одной восьмеричной цифрой. Аналогичным образом осуществляется перевод двоичных чисел в шестнадцатеричные. Только в этом случае двоичное число разбивается на 4 тетрады, которые представляются одной шестнадцатеричной цифрой.

Наконец следует упомянуть о двоично-десятичной СС, которая используется в цифровых устройствах, где основная часть операций связана не с обработкой и хранением вводимой информации, а с самим её выводом на какие-либо на какие-либо индикаторы с десятичным представлением полученных результатов. В 2-10СС десятичные цифры от 0 до 9 представляют 4-разрядными двоичными комбинациями от 0000 до 1001. Две двоично-десятичные цифры составляют 1 байт (можно представлять значения от 0 до 99)

3. Представление чисел с фиксированной точкой. Прямой, обратный и дополнительный код. Формирование битовых признаков переноса, переполнения, отрицательного результата, нуля.

Целые двоичные числа без знака можно использовать для представления нуля и целых положительных чисел. При размещении таких чисел в одном 16-разрядном слове они могут изменяться от $(0000\ 0000\ 0000\ 0000)_2 = (000\ 0)_{16} = 0$ до $(1111\ 1111\ 1111\ 1111)_2 = (FFFF)_{16} = 2^{16} - 1 = 65535$. Такая запись называется прямым кодом числа.

Подобные числа (так же как и рассмотренные ниже двоичные числа со знаком) относятся к числам с фиксированной запятой, разделяющей целую и дробную части числа. В числах, используемых в базовой ЭВМ, положение запятой строго фиксировано после младшего бита слова.

Целые двоичные числа со знаком используются тогда, когда необходимо различать положительные и отрицательные числа. В современных ЭВМ для представления целых чисел со знаком используется дополнительный код, в котором старший бит формата определяет знак числа: 0 - для положительных чисел и 1 - для отрицательных чисел. При этом дополнительный код положительного числа совпадает с его прямым кодом. А для представления отрицательного числа в дополнительном коде производится инвертирование прямого кода модуля числа (получение обратного кода числа) и добавление к результату единицы. Такая же операция используется при изменении знака числа, представленного в дополнительном коде.

Использование дополнительного кода упрощает конструкцию ЭВМ, так как при сложении двух таких чисел, имеющих разные знаки, не требуется переходить к операциям вычитания меньшего (по модулю) числа из большего и присвоения результату знака большего числа. Кроме того, одной и той же схемой сумматора можно воспользоваться для выполнения операций над знаковым и беззнаковым представлением числа.

Признаком выхода за границы разрядной сетки для беззнакового представления числа является перенос в старший разряд (бит C - Carry). Например, при сложении:

$$\begin{array}{r} 1000\ 0000\ 0000\ 0000 \\ +\ 1000\ 0000\ 0000\ 0000 \\ \hline 1\ 0000\ 0000\ 0000\ 0000 \end{array}$$

В ответе должно получиться 32768+32768=65536, но т.к. разрядность слова составляет лишь 16 бит, то в нем сохраняется только часть результата, т.е. 0. Единица, возникшая вследствие переноса оказалась в несуществующем 17 разряде.

Признаком переполнения разрядной сетки для знакового представления

является бит переполнения (OVerflow). Разные знаки слагаемых, или совпадение знаков слагаемых со знаком суммы свидетельствуют о том что результат корректен. в противном случае формируется сигнал – Переполнение

Признак отрицательного результата N при знаковом представлении выставляется в случае когда в старшем разряде числа в доп. коде находится 1

Признак нулевого результата Z выставляется в случае когда все разряды числа равны 0

4. Представление символьных и строковых данных. Принципы построения кодовых таблиц ASCII, КОИ-8, ISO8859-5, Windows-1251, UTF-8, UTF-16.

ASCII

Порядковые номера	Коды символов	Что за…
0 — 31	00000000 — 00011111	Управляющие символы ДЛЯ управления процессом вывода текста на экран или печать, подача звукового сигнала, разметка текста и т.п.
32 — 127	00100000 — 01111111	Сюда входят строчные и прописные буквы латинского алфавита, десятичные цифры, знаки препинания, всевозможные скобки, коммерческие и другие символы. (Символ 32 – пробел)
128 — 255	10000000 — 11111111	Кодовая страница ДЛЯ нелатинского алфавита

КОИ-8

— восьмибитовая кодовая страница, совместимая с ASCII

Разработчики КОИ-8 поместили символы русского алфавита в верхней части кодовой таблицы таким образом, что позиции символов кириллицы соответствуют их фонетическим аналогам в английском алфавите из нижней части таблицы. Это означает, что если в тексте, написанном в КОИ-8, убрать восьмой бит каждого символа, то получится «читаемый» текст, подобный транслиту. Например, слова «Русский Текст» превратятся в «rUSSKIJ tEKST». Из-за этого символы кириллицы расположены не в алфавитном порядке.

.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F	
8.	─ 2500	│ 2502	┌ 250C	┐ 2510	└ 2514	┘ 2518	├ 251C	┤ 2524	┬ 252C	┴ 2534	┼ 253C	▀ 2580	▄ 2584	█ 2588	▌ 258C	▐ 2590
9.	░ 2591	▒ 2592	▓ 2593	⌠ 2320	■ 25A0	· 2219	√ 221A	≈ 2248	≤ 2264	≥ 2265	A0	⌡ 2321	° B0	² B2	· B7	÷ F7
A.	═ 2550	║ 2551	╒ 2552	ё 451	╓ 2553	╔ 2554	╕ 2555	╖ 2556	╗ 2557	╘ 2558	╙ 2559	╚ 255A	╛ 255B	╜ 255C	╝ 255D	╞ 255E
B.	╟ 255F	╠ 2560	╡ 2561	Ё 401	╢ 2562	╣ 2563	╤ 2564	╥ 2565	╦ 2566	╧ 2567	╨ 2568	╩ 2569	╪ 256A	╫ 256B	╬ 256C	© A9
C.	ю 44E	а 430	б 431	ц 446	д 434	е 435	ф 444	г 433	х 445	и 438	й 439	к 43A	л 43B	м 43C	н 43D	о 43E
D.	п 43F	я 44F	р 440	с 441	т 442	у 443	ж 436	в 432	ь 44C	ы 44B	з 437	ш 448	э 44D	щ 449	ч 447	ъ 44A

E.	Ю 42E	А 410	Б 411	Ц 426	Д 414	Е 415	Ф 424	Г 413	Х 425	И 418	Й 419	К 41A	Л 41B	М 41C	Н 41D	О 41E
F.	П 41F	Я 42F	Р 420	С 421	Т 422	У 423	Ж 416	В 412	Ь 42C	Ы 42B	З 417	Ш 428	Э 42D	Щ 429	Ч 427	Ъ 42A

ISO – 8859-5

- 8-битная кодовая страница из семейства кодовых страниц стандарта ISO-8859 для представления кириллицы. Нижняя часть таблицы кодировки полностью соответствует кодировке ASCII. Числа под буквами — шестнадцатеричный код буквы в Юникоде. Основной недостаток - отсутствие некоторых символов, такие как тире (—), кавычки-ёлочки («»), градус (°), поэтому в России почти не использовалась.

.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F	
8.	PAD 80	HOP 81	BPH 82	NBH 83	IND 84	NEL 85	SSA 86	ESA 87	HTS 88	HTJ 89	VTS 8A	PLD 8B	PLU 8C	RI 8D	SS2 8E	SS3 8F
9.	DCS 90	PU1 91	PU2 92	STS 93	CCH 94	MW 95	SPA 96	EPA 97	SOS 98	SGCI 99	SCI 9A	CSI 9B	ST 9C	OSC 9D	PM 9E	APC 9F
A.	 A0	Ё 401	Ђ 402	Ѓ 403	Є 404	S 405	I 406	Ї 407	J 408	Љ 409	Њ 40A	Ћ 40B	Ќ 40C	SHY AD	Ў 40E	Џ 40F
B.	А 410	Б 411	В 412	Г 413	Д 414	Е 415	Ж 416	З 417	И 418	Й 419	К 41A	Л 41B	М 41C	Н 41D	О 41E	П 41F
C.	Р 420	С 421	Т 422	У 423	Ф 424	Х 425	Ц 426	Ч 427	Ш 428	Щ 429	Ъ 42A	Ы 42B	Ь 42C	Э 42D	Ю 42E	Я 42F
D.	а 430	б 431	в 432	г 433	д 434	е 435	ж 436	з 437	и 438	й 439	к 43A	л 43B	м 43C	н 43D	о 43E	п 43F
E.	р 440	с 441	т 442	у 443	ф 444	х 445	ц 446	ч 447	ш 448	щ 449	ъ 44A	ы 44B	ь 44C	э 44D	ю 44E	я 44F
F.	№ 2116	ё 451	ђ 452	ѓ 453	є 454	s 455	і 456	ї 457	j 458	љ 459	њ 45A	ћ 45B	ќ 45C	§ A7	ў 45E	џ 45F

Windows-1251

— набор символов и кодировка, являющаяся стандартной 8-битной кодировкой для русских версий Microsoft Windows до 10-й версии. Windows-1251 как и KOI8-R выгодно отличается от других 8-битных кириллических кодировок (таких как CP866 и ISO 8859-5) наличием практически всех символов, использующихся в русской типографике для обычного текста

Минусы:

1. Строчная буква «я» имеет код 0xFF (255 в десятичной системе), что совпадает со служебным символом в некоторых других кодировках, например в CP437 это «неразрывный пробел»
2. Отсутствуют символы псевдографики, имеющиеся в CP866 и KOI8

.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F	
8.	Ђ 402	Ѓ 403	‚ 201A	ѓ 453	„ 201E	… 2026	† 2020	‡ 2021	€ 20AC	‰ 2030	Љ 409	‹ 2039	Њ 40A	Ќ 40C	Ћ 40B	Џ 40F
9.	ђ 452	‘ 2018	’ 2019	“ 201C	” 201D	• 2022	– 2013	— 2014		™ 2122	љ 459	› 203A	њ 45A	ќ 45C	ћ 45B	џ 45F
A.	 A0	Ў 40E	ў 45E	J 408	¤ A4	Ґ 490	¦ A6	§ A7	Ё 401	© A9	Є 404	« AB	¬ AC	AD	® AE	Ї 407
B.	° B0	± B1	I 406	і 456	ґ 491	µ B5	¶ B6	· B7	ё 451	№ 2116	є 454	» BB	j 458	S 405	s 455	ї 457
C.	А 410	Б 411	В 412	Г 413	Д 414	Е 415	Ж 416	З 417	И 418	Й 419	К 41A	Л 41B	М 41C	Н 41D	О 41E	П 41F

D.	Р 420	С 421	Т 422	У 423	Ф 424	Х 425	Ц 426	Ч 427	Ш 428	Щ 429	Ъ 42A	Ы 42B	Ь 42C	Э 42D	Ю 42E	Я 42F
E.	а 430	б 431	в 432	г 433	д 434	е 435	ж 436	з 437	и 438	й 439	к 43A	л 43B	м 43C	н 43D	о 43E	п 43F
F.	р 440	с 441	т 442	у 443	ф 444	х 445	ц 446	ч 447	ш 448	щ 449	ъ 44A	ы 44B	ь 44C	э 44D	ю 44E	я 44F

UTF-8

— одна из общепринятых 8-битных и стандартизированных кодировок текста, которая позволяет хранить символы Юникода, используя переменное количество байт (от 1 до 6)
Совместимость с ASCII — любые их 7-битные символы отображаются как есть, а остальные выдают пользователю мусор (шум). Поэтому в случае, если латинские буквы и простейшие знаки препинания (включая пробел) занимают существенный объём текста, UTF-8 даёт выигрыш по объёму по сравнению с UTF-16.

Кодирование UTF-8

1. Если размер символа в кодировке UTF-8 = 1 байт

Код имеет вид (0aaa aaaa), где «0» — просто ноль, остальные биты «a» — это код символа в кодировке ASCII;

2. Если размер символа в кодировке в UTF-8 > 1 байт (то есть от 2 до 6):

2.1 Первый байт содержит количество байт символа, закодированное в **единичной** системе счисления;

2 — 11
3 — 111
4 — 1111
5 — 1111 1
6 — 1111 11

2.2 «0» — бит терминатор, означающий завершение кода размера

2.3 далее идут значащие байты кода, которые имеют вид (10xx xxxx), где «10» — биты признака продолжения, а «x» — значащие биты.

В общем случае варианты представления **одного символа** в кодировке UTF-8 выглядят так:

(1 байт) 0aaa aaaa
(2 байта) 110x xxxx 10xx xxxx
(3 байта) 1110 xxxx 10xx xxxx 10xx xxxx
(4 байта) 1111 0xxx 10xx xxxx 10xx xxxx 10xx xxxx
(5 байт) 1111 10xx 10xx xxxx 10xx xxxx 10xx xxxx 10xx xxxx
(6 байт) 1111 110x 10xx xxxx 10xx xxxx 10xx xxxx 10xx xxxx 10xx xxxx

UTF-16

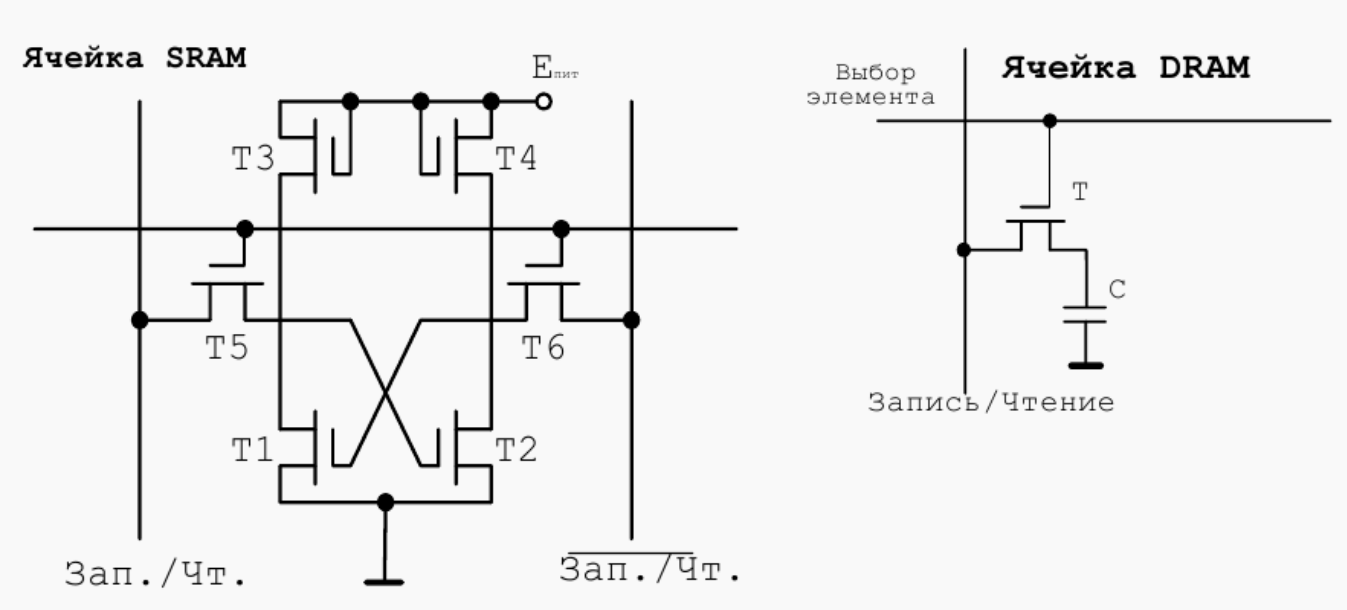
— символы кодируются двухбайтовыми словами с использованием всех возможных диапазонов значений (от 0000_{16} до $FFFF_{16}$). При этом можно кодировать символы Unicode Сдвигаем символ в 8в дипазонах $0000_{16}..D7FF_{16}$ и $E000_{16}..10FFFF_{16}$. Исключенный отсюда диапазон

$D800_{16}..DFFF_{16}$ используется как раз для кодирования так называемых суррогатных пар — символов, которые кодируются двумя 16-битными словами. Символы Unicode до $FFFF_{16}$ включительно (исключая диапазон для суррогатов) записываются как есть 16-битным словом. Символы же в диапазоне $10000_{16}..10FFFF_{16}$ (больше 16 бит) уже кодируются парой 16-битных слов. Для этого их код арифметически сдвигается до нуля (из него вычитается минимальное число 10000_{16}). В результате получится значение от нуля до FFFFF, которое занимает до 20 бит. Старшие 10 бит этого значения идут в лидирующее (первое) слово, а младшие 10 бит — в последующее (второе). При этом в обоих словах старшие 6 бит используются для обозначения суррогата. Биты с 11 по 15 имеют значения 11011_2, а 10-й бит содержит 0у лидирующего слова и 1 — у последующего.

5. Базовые элементы вычислительной техники: ячейки, регистры, шины, вентили, тактовые генераторы.

Ячейка памяти – минимальный адресуемый элемент запоминающего устройства ЭВМ. Ячейки имеют адрес (порядковый номер, число), по которому к ним могут обращаться команды процессора. Ячейки памяти состоят из элементов, которые могут находиться в одном из двух устойчивых состояний: конденсатор заряжен или разряжен, транзистор находится в проводящем или непроводящем состоянии. Одно из таких физических состояний создает высокий уровень выходного напряжения элемента памяти, а другое – низкий. Первое обычно принимается за двоичную 1, а второе – за двоичный 0. Возможно и обратное кодирование. Хотя переход от 0 к 1 и от 1 к 0 происходит не мгновенно, однако в определенные моменты времени этот сигнал достигает значений, которые воспринимаются элементами ЭВМ как 0 или 1.

Память бывает статическая (SRAM - *static random access memory*) и динамическая (DRAM – dynamic ...)



Статическая память с произвольным доступом (*SRAM, static random access memory*) — полупроводниковая оперативная память, в которой каждый двоичный разряд хранится в схеме с положительной обратной связью, позволяющей поддерживать состояние без регенерации. Тем не менее, сохранять данные без перезаписи SRAM может, только пока есть питание.

Преимущества:

- Быстрый доступ (действительно память *произвольного* доступа, доступ к любой ячейке памяти в любой момент занимает одно и то же время)
- Простая схемотехника.
- Возможны очень низкие частоты синхронизации (вплоть до полной остановки синхроимпульсов.)

Недостатки:

- Невысокая плотность записи (шесть-восемь элементов на бит, вместо двух у DRAM).
- Вследствие чего — дороговизна килобайта памяти.
- Особенность: непредсказуемое (произвольное) содержимое памяти после включения питания.

SRAM применяется в микроконтроллерах, в которых объём ОЗУ невелик (единицы килобайт), зато нужны низкое энергопотребление (за счёт отсутствия сложного контроллера динамической памяти), предсказываемое с точностью до такта время работы подпрограмм.

DRAM (*dynamic random access memory* — динамическая память с произвольным доступом) — тип компьютерной памяти, отличающийся использованием полупроводниковых материалов, энергозависимостью и возможностью доступа к данным, хранящимся в произвольных ячейках памяти.

Физически DRAM состоит из ячеек, созданных в полупроводниковом материале в виде емкости. Заряженная или разраженная емкость хранит бит данных. Каждая ячейка такой памяти имеет свойство разряжаться, поэтому их постоянно надо подзаряжать. Совокупность ячеек образует условный «прямоугольник», состоящий из определённого количества *строк* и *столбцов*. Один такой «прямоугольник» называется *страницей*, а совокупность страниц называется *банком*.

Регистр процессора – память внутри процессора, предназначенная для хранения адресов и промежуточных результатов вычислений или данных, необходимых для работы самого процессора. Регистр характеризуется единственным числом: количеством битов, которые могут в нем храниться. Операция чтения информации, хранимой в регистре, сводится к созданию копии его содержимого, оригинал же сохраняется в регистре без изменений.

Шина - электрическая цепь, соединяющая регистр с другим регистром или иным устройством ЭВМ. Шина состоит из параллельных проводов, каждый из которых предназначен для передачи соответствующего регистра. Также шина содержит несколько дополнительных проводов, используемых для передачи сигналов синхронизации и управления. Шины служат для передачи информации лишь в направлении, обозначенном стрелкой на шине. Специальные схемы позволяют в одни моменты времени передавать информацию по шине в одну сторону, а в другие – в обратном направлении, т.е. организовать двунаправленную шину.

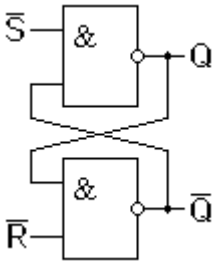
Вентильные схемы – это электронные ключевые схемы, предназначенные для управления потоком информации из регистров в шины и обратно. Такая схема содержит два входа и один выход. На один вход подается информационный сигнал (данные с регистра), а на другой (являющийся вентилем) – управляющий. Если управляющий сигнал равен 1, то данные проходят схему без препятствий, если 0 – никакая информация не пройдет через схему. Для подачи информационного сигнала на вход вентильной схемы обычно используется многопроводная шина. Для передачи выходного сигнала требуется шина с таким же количеством проводов. Если управляющий сигнал равен 1, то информационные сигналы на входной и выходной шинах совпадают.

Тактовый генератор – устройство, генерирующее электрические импульсы заданной частоты (обычно прямоугольной формы). Используется для синхронизации процессов передачи информации между устройствами ЭВМ.

6. Базовые элементы вычислительной техники: логические схемы, триггеры, регистры, счетчики, сумматоры.

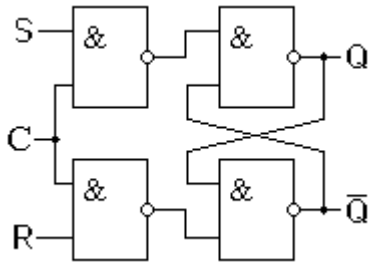
Функциональная логическая схема - совокупность логических элементов (простейшее устройство ЭВМ, выполняющее одну определённую логическую операцию над входными сигналами согласно правилам алгебры логики) и связей между ними.

Триггер — класс электронных устройств, обладающих способностью длительно находиться в одном из двух устойчивых состояний и чередовать их под воздействием внешних сигналов. Каждое состояние триггера легко распознаётся по значению выходного напряжения. Отличительной особенностью триггера как функционального устройства является свойство запоминания двоичной информации. Под памятью триггера подразумевают способность оставаться в одном из двух состояний и после прекращения действия переключающего сигнала.



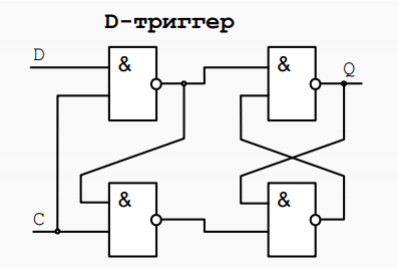
RS-триггер получил название по названию своих входов. Вход S (Set — установить англ.) позволяет устанавливать выход Q в единичное состояние. (Устанавливать означает записывать логическую единицу). Вход R (Reset — сбросить англ.) позволяет сбрасывать выход Q (Quit — выход англ.) в нулевое состояние.

R	S	Q(t)	Q(t+1)	Пояснения
0	0	0	0	Режим хранения информации (триггером) R=S=0
0	0	1	1	
0	1	0	1	Режим установки триггера в единичное состояние S=1
0	1	1	1	
1	0	0	0	Режим записи нуля в триггер R=1
1	0	1	0	
1	1	0	*	R=S=1 запрещенная комбинация
1	1	1	*	



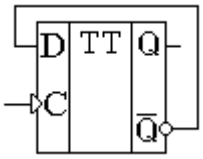
Синхронный RS-триггер. Схема, пропускающая входные сигналы только при наличии синхронизирующего сигнала. В этой таблице символ x означает, что значения логических уровней на данном входе не важны. Они не влияют на работу триггера.

C	R	S	Q(t)	Q(t+1)	Пояснения
0	x	x	0	0	Режим хранения информации
0	x	x	1	1	
1	0	0	0	0	Режим хранения информации
1	0	0	1	1	
1	0	1	0	1	Режим установки единицы S=1
1	0	1	1	1	
1	1	0	0	0	Режим записи нуля R=1
1	1	0	1	0	
1	1	1	0	*	R=S=1 запрещенная комбинация
1	1	1	1	*	



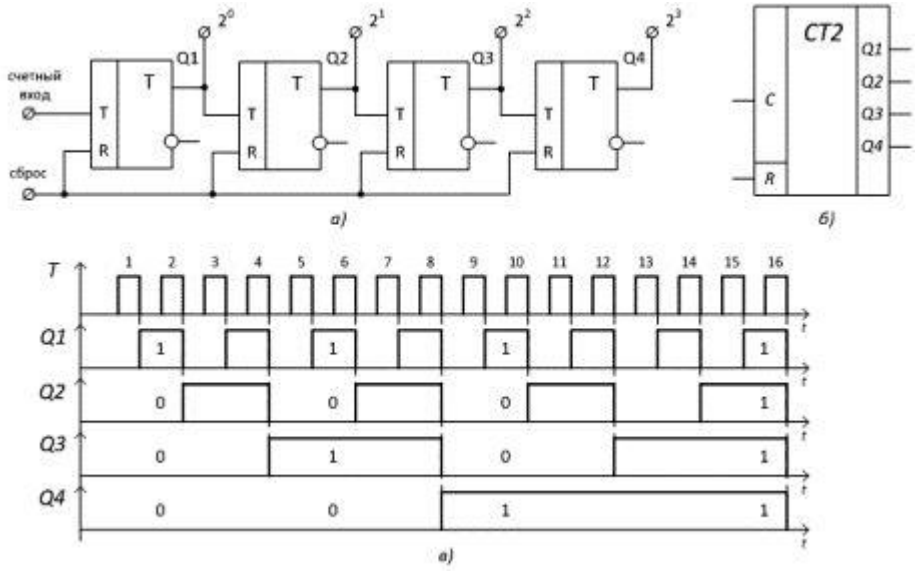
D-тригер. При записи и хранении данных один бит может принимать значение, как нуля, так и единицы. Для его передачи достаточно одного провода. сигналы установки и сброса триггера не могут появляться одновременно, поэтому можно объединить эти входы при помощи инвертора.

C	D	Q(t)	Q(t+1)	Пояснения
0	x	0	0	Режим хранения информации
0	x	1	1	
1	0	x	0	Режим записи информации
1	1	x	1	

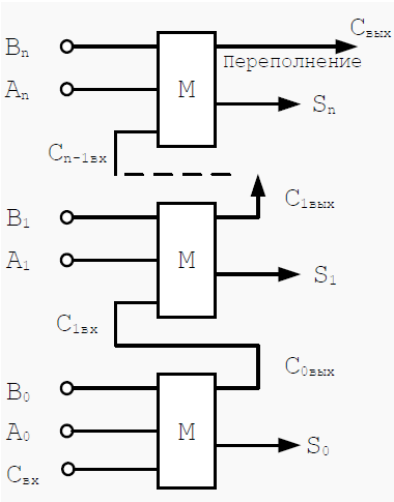


T-триггер — это счетный триггер. У данного триггера имеется только один вход. Принцип работы Т-триггера заключается в следующем. После поступления на вход Т импульса, состояние триггера меняется на прямо противоположное.

Счётчик числа импульсов — устройство, на выходах которого получается двоичный (двоично–десятичный) код, определяемый числом поступивших импульсов. Счетчики импульсов являются разновидностью регистров (счетные регистры) и строятся соответственно на триггерах и логических элементах. Основной параметр счётчика —модуль счёта —максимальное число единичных сигналов, которое может быть сосчитано счётчиком. На рисунке представлена схема четырехразрядного счетчика на Т-триггерах, соединенных последовательно. Счетные импульсы подаются на счетный вход первого триггера. Счетные входы последующих триггеров связаны с выходами предыдущих триггеров.



Сумматор —устройство, преобразующее информационные сигналы (аналоговые или цифровые) в сигнал, эквивалентный сумме этих сигналов.



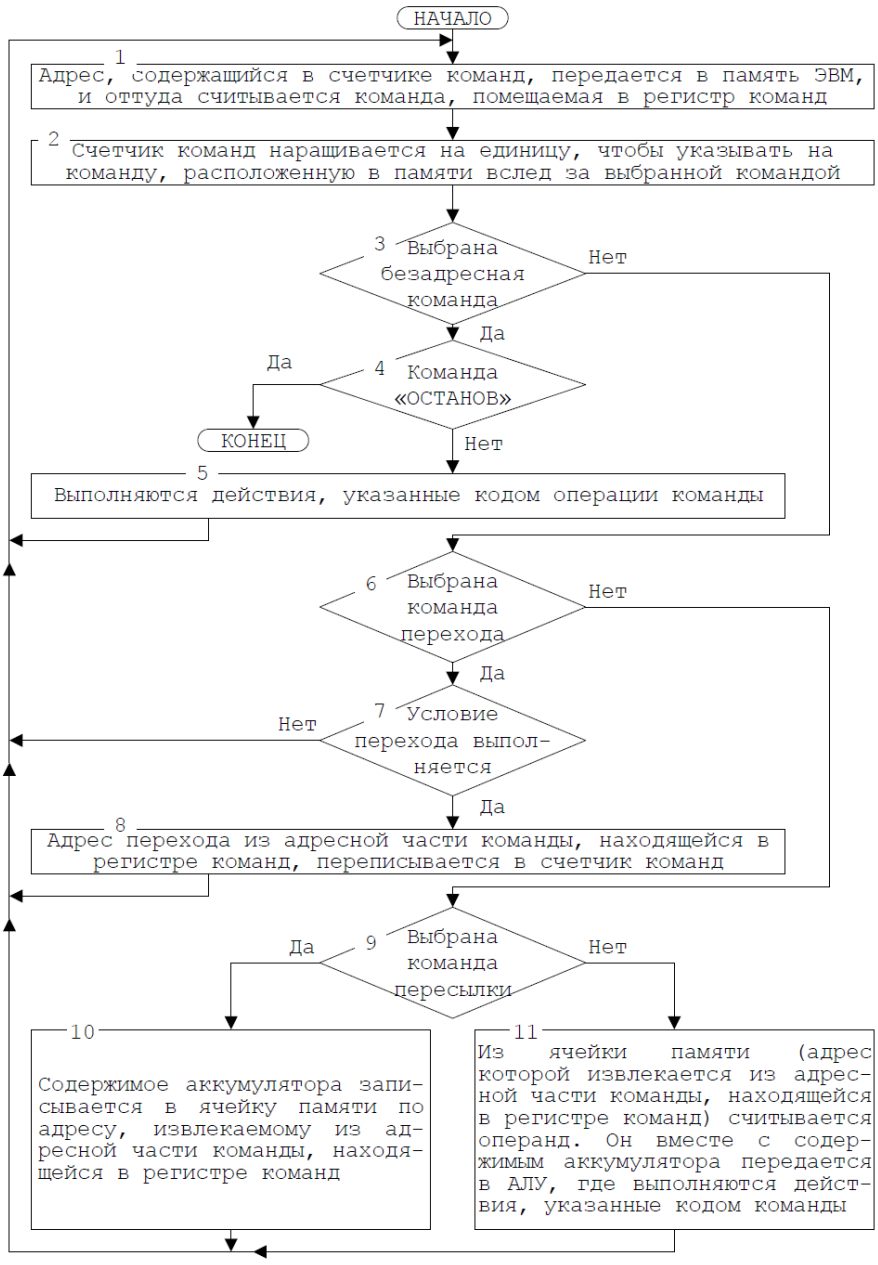
SUM				
Свх	Ai	Bi	Свых	Si
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

7. Структура и принцип функционирования ЭВМ. Порядок функционирования простого процессора.

Типичная ЭВМ состоит из процессора, памяти и устройств ввода-вывода.
«Сердцем» ЭВМ является процессор, в состав которого входят устройство управления выборкой команд из памяти и их выполнением, арифметико-логическое устройство, производящее операции над данными, регистры, осуществляющие временное хранение данных и состояний процессора, схемы для управления и связи с подсистемами памяти и ввода-вывода.
Устройство ввода обеспечивает считывание информации с определенных носителей информации и ее представление в форме электрических сигналов, воспринимаемых другими устройствами ЭВМ. Устройства вывода представляют результаты обработки информации в форме, удобной для визуального восприятия. Память ЭВМ включает устройство, обеспечивающее хранение команд и данных. Это устройство состоит из блоков одинакового размера – ячеек памяти, предназначенных для хранения одного слова информации.
Ячейка памяти состоит из элементов памяти, состояние каждого из которых соответствует одной двоичной цифре. Совокупность нулей и единиц, хранящихся в элементах одной ячейки, представляет собой содержимое этой ячейки памяти. В микро ЭВМ используются безадресные, одноадресные и реже двухадресные команды. В одноадресных командах один из операндов

выбирается из специального регистра – аккумулятора. В него же заносится и результат операции. Безадресные команды или задают какое-либо действие с устройствами ЭВМ, или используются для работы с операндами, имеющими фиксированное расположение (чаще всего с аккумулятором). В процессе работы ЭВМ последовательно выполняет набор достаточно простых операций: выборку команды, определение ее типа, исполнение команды и определение адреса следующей команды.

Порядок функционирования простого процессора



8. Операционная система Unix — ядро ОС и файловая система.

From wiki: https://ru.wikipedia.org/wiki/UNIX

UNIX — семейство переносимых, многозадачных и многопользовательских операционных систем. Идеи, заложенные в основу UNIX, оказали огромное влияние на развитие компьютерных операционных систем. В настоящее время UNIX-системы признаны одними из самых исторически важных ОС.

Основное отличие UNIX-подобных систем от других операционных систем заключается в том, что это изначально многопользовательские многозадачные системы. То есть в один и тот же момент времени сразу множество людей может выполнять множество вычислительных задач (процессов). Даже популярную во всём мире систему Microsoft Windows нельзя назвать полноценной многопользовательской системой, так как кроме как на некоторых серверных версиях, в один и тот же момент за одним компьютером с Windows может работать только один человек. В Unix может работать сразу много людей, при этом каждый из них может выполнять множество различных вычислительных процессов, которые будут использовать ресурсы именно этого компьютера.

Вторая колоссальная заслуга Unix в её мультиплатформенности. Ядро системы написано таким образом, что его легко можно приспособить практически под любой микропроцессор.

UNIX имеет и другие характерные особенности:

- использование простых текстовых файлов для настройки и управления системой;
- широкое применение утилит, запускаемых из командной строки;
- взаимодействие с пользователем посредством виртуального устройства — терминала;
- представление физических и виртуальных устройств и некоторых средств межпроцессового взаимодействия в виде файлов;
- использование конвейеров из нескольких программ, каждая из которых выполняет одну задачу.

Файловая система UNIX

From http://works.doklad.ru/view/r8f_Kyf1Whs.html

Понятие файла является одним из наиболее важных для ОС UNIX. Все файлы, с которыми могут манипулировать пользователи, располагаются в файловой системе, представляющей собой дерево, промежуточные вершины которого соответствуют каталогам, и листья - файлам и пустым каталогам. Каждый каталог и файл файловой системы имеет уникальный полный путь. Каталог, являющийся корнем файловой системы (корневой каталог) имеет путь /. Коротким или относительным путем называется путь к файлу от текущего рабочего каталога. В каждом каталоге содержатся два специальных файла-ссылки, файл "." - ссылка на текущий каталог, и ссылка ".." на родительский каталог.

inode - Index-node - описатель файла, его уникальный номер. Он содержит всю информацию о файле, за исключением имени файла, и собственно данных файла. В inod'е хранится:
тип файла, права, время модификации/создания файла и другая служебная информация под общим названием «метаданные».

9. Операционная система Unix — интерпретаторы, стандартные потоки ввода вывода, фильтры.

Командный интерпретатор – программа, предоставляющая пользователю интерфейс для общения с командной строкой; эта программа «переводит» введенные пользователем команды на понятный операционной системе язык. Интерпретатор более известен как ***оболочка*** *(англ. shell)*. Наиболее распространенными оболочками являются *sh, bash* (стандарт в Unix)*, c shell.* Пользователь может вводить команды как по отдельности, так и с помощью набора команд (скриптов). Команды могут задаваться как напрямую в командной строке, так и поступать из *стандартного ввода* или указанного файла. В качестве команд могут приниматься вызовы системных или прикладных утилит или управляющие конструкции. Кроме того, оболочка отвечает за перенаправление потоков ввода-вывода. В совокупности с набором утилит, она представляет собой операционную среду, язык программирования и средства решения как системных, так и прикладных задач, особенно по части автоматизации выполняемых последовательностей команд.
Для взаимодействия и обмена информацией с пользователем используются файлы, именуемые **стандартными потоками ввода** (для чтения из него) и **вывода** (для записи в него). Вывод на экран представляется тоже как запись в файл, а ввод – как чтение из файла. Кроме потоков ввода и вывода существует так же **стандартный поток ошибок**, на который выводится вся служебная информация, которая не должна попадать в поток вывода (сообщения об ошибке или ходе работы программы).
Стандартные потоки привязаны к *файловым дескрипторам* с номерами:
0 для ввода (stdin) 1 для вывода (stdout) 2 для ошибок (stderr).

Потоки по умолчанию связаны с терминалом (командной строкой), но их можно подключить к чему угодно – к файлам, программам или устройствам. В интерпретаторе такая операция называется *перенаправлением*. Таким образом, стандартные потоки можно перенаправлять не только в файлы, но и на вход других программ.
Для осуществления перенаправления используются следующие операции:

Команда > файл (или >>)
Выполняется команда, а вывод помещается в файл (или добавляется в конец).

Команда < файл
Файл используется в качестве источника ввода. При этом на каждый запрос ввода программы считывается 1 строка текста из файла.

Команда1 | команда2
Вывод команды1 пойдет в качестве ввода на команду2 без использования промежуточных файлов. Такая возможность называется ***конвейером.***

Команда 2> файл
Поток ошибок направляется в файл. По умолчанию этот поток выводится на стандартный вывод.

Команда 2>&1 файл (или &> или >&)
Такой синтаксис используется для объединения потоков вывода и потока ошибок для обработки их вместе.

Файл т.н. «пустое устройство» - /dev/null – перенаправление в него позволяет избавиться от ненужных сообщений об ошибке или игнорирования вывода. С помощью него также можно создавать пустые файлы, используя в качестве источника ввода. При записи в него может вместить любое количество информации, он работает в качестве «черной дыры».

10. Операционная система Unix — основные команды, права файлов и способы их задания.

Основные команды

touch файл

Создает пустой файл, а если он уже есть – обновляет время последней модификации.

mkdir каталог

Создает пустой каталог.

rm файл

Удаляет файл.

–r

Рекурсивно стирает каталоги. Если этого флага нет, файл не может быть каталогом.

rmdir каталог

Стирает только пустые каталоги.

echo

Выводит строку текста.

cat файл

Выводит содержимое файла.

pwd

Выводит имя текущего каталога.

ls файл

Выводит список файлов в каталоге или информацию о файле, если это не каталог.

–l

Длинный формат. Выводится с подробной информацией о каждом файле.

–a

Вывод вместе со скрытыми файлами.

–F

К имени файла добавляется его тип.

–R

Рекурсивно выводит подкаталоги.

cd каталог

Переходит в каталог.

cp файл1 файл2

Копирует файл в другой файл.

mv файл каталог

Перемещает файл в каталог.

ln файл1 файл2

Создает новую жесткую ссылку на файл. Жесткая ссылка может ссылаться только в пределах одного диска. Файл не будет удален, пока на него есть хоть одна жесткая ссылка.

–s

Создает символическую ссылку. Может ссылаться куда угодно. Если переместить/удалить файл, симв. ссылка будет недействительна.

head/tail файл

Выводит первые/последние 4 строки файла

–n

Первые/последние n строк.

–c

Первые/последние c байт.

wc файл

Выводит количество строк, слов и байт в файле.

–l

Только кол-во строк.

–w

Только кол-во слов.

–c

Только кол-во байт.

–m

Кол-во символов.

find выражение

Ищет файлы в иерархии каталогов по заданным параметрам.

man команда

Выводит справку по команде.

Права доступа к файлам

Для каждого файла существуют следующие категории пользователей:

u (user)

Владелец файла.

g (group)

Члены группы, владеющей файлом.

o (others)

Все остальные.

a (all)

Все категории. Не рассматривается как отдельная катеория.

Каждая из этих категорий может иметь любую комбинацию из следующих прав:

r (read)

Право на чтение файла/просмотр каталога.

w (write)

Право на запись в файл/добавление или удаление каталога.

x (execute)

Право на исполнение файла/поиск и переход в каталог.

Права представляют собой последовательность из 9 бит – по 3 бита на категорию: владелец, группа, прочие; в следующем порядке – чтение, запись, исполнение. В случае отсутствия какого-либо из прав у категории, ставится символ «-».
Вторым способом записи прав является запись этой последовательности в 8-ричной системе счисления, где праву на чтение (r) соответствует цифра 4, праву на запись (w) – цифра 2, а праву на исполнение (x) – цифра 1. Цифра 0 означает отсутствие прав. Для получения конечной цифры, нужные права суммируются. Таким образом, запись занимает всего 3 бита: по 1 биту на категорию.

Для выставления прав файлу (каталогу) используется команда chmod.
Существует 3 способа задания прав доступа:

chmod [ugoa]{+-=}[rwx] файл

Добавляет, удаляет или устанавливает выбранную комбинацию прав для выбранной комбинации категорий.

chmod число файл

Устанавливает права на основе восьмиричной записи.

chmod категория1=категория2 файл

Копирует права одной категории и присваивает их другой.

11. Состав и структура БЭВМ. Адресные пространства БЭВМ.

Базовая ЭВМ – это простая гипотетическая машина, обладающая типичными чертами многих конкретных ЭВМ. Память состоит из 2048 ячеек (16---битовых) с адресами 0, 1, …, 2046, 2047. Восемь ячеек памяти с адресами 008---00F называются индексными и их лучше использовать в циклических программах. Процессор состоит из ряда регистров (РК, А, РД, СК, РА, С), арифметико---логического устройства и устройства управления. Счетчик команд служит для организации обращений к ячейкам памяти, в которых хранятся команды программы. После исполнения любой команды СК указывает адрес ячейки памяти, содержащий следующую команду. Имеет 11 разрядов. Регистр адреса – 11---разрядный регистр, содержащий значение исполнительного адреса (адреса ячейки памяти, к которой обращается ЭВМ за командой или данными). Регистр команд – 16---разрядный регистр, который используется для хранения кода команды, непосредственно выполняемой машиной. Регистр данных используется для временного хранения 16---разрядных слов при обмене информацией между памятью и процессором. Аккумулятор – 16-разрядный регистр, являющийся одним из главных элементов процессора. Машина может одновременно выполнять арифметические и логические операции только с одним или двумя операндами. Один из операндов находится в аккумуляторе, а второй – в регистре данных. Результат помещается в А. Регистр переноса – это 1---разрядный регистр, выступающий в качестве продолжения аккумулятора и заполняющийся при переполнении А. Этот регистр используется при выполнении сдвигов. АЛУ может выполнять такие арифметические операции, как сложение и вычитание с учетом переноса, полученного в результате выполнения предыдущей операции. Кроме того, оно способно выполнять операции логического умножения, инвертирования, циклического сдвига, инкремента, etc.

12. Система команд БЭВМ, форматы команд. Машинные циклы.

ЭВМ способна понимать и выполнять точно определенный набор команд. При составлении программы пользователь ограничен этими командами. В зависимости от того, к каким блокам базовой ЭВМ обращается команда или на какие блоки она ссылается, команды можно разделить на три группы: обращения к памяти (адресные команды), обращения к регистрам (регистровые или безадресные команды), команды ввода---вывода. Команды обращения к памяти предписывают машине производить действия с содержимым ячейки памяти, адрес которой указан в адресной части команды. Безадресные команды выполняют различные действия без ссылок на ячейку памяти. Команды ввода---вывода осуществляют обмен данными между процессором и внешними

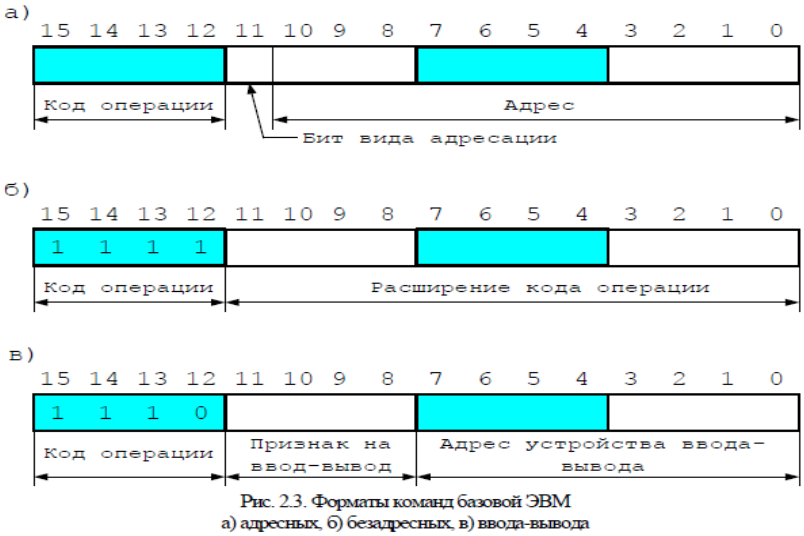


Рис. 2.3. Форматы команд базовой ЭВМ
а) адресных, б) безадресных, в) ввода-вывода

устройствами ЭВМ. Разработчики базовой ЭВМ выбрали три формата 16---битовых (однословных) команд с 4---битовым кодом операции. В командах обращения к памяти на адрес отведено 11 бит. Следовательно, можно прямо адресоваться к 2^{11}=2048 ячейкам памяти, т.е. ко всей памяти базовой ЭВМ (прямая адресация). В этом случае бит вида адресации должен содержать 0. Если же в этом бите установлена 1, о адрес, размещенный в адресной части команды, указывает на ячейку, в которой находится адрес операнда (косвенная адресация). В процессе исполнения команд устройство управления ЭВМ производит анализ и пересылку команд, отдельных ее частей (кода операции, признака адресации и адреса) или операнда из одного регистра ЭВМ в другой ее регистр, АЛУ, память или устройство ввода---вывода. Эти действия (микрооперации) протекают в определенной временной последовательности и скоординированы между собой. Для обеспечения такой последовательности в ЭВМ используется ГТИ. Для реализации одной команды требуется выполнить определенное количество микрокоманд, каждая из которых инициируется одним тактовым импульсом. Общее число тактовых импульсов, требуемых для выполнения команды, определяет время ее выполнения, называемое циклом команды. Цикл команды обычно включает один или несколько машинных циклов. Устройство управления базовой ЭВМ может находиться в 4 возможных состояниях: выборки команды, выборки адреса, исполнения, прерывания.

Наименование	Мнемон.	Код	Описание
Логическое умножение	**AND M**	1XXX	(M) & (A) ➔ A
Пересылка	**MOV M**	3XXX	(A) ➔M
Сложение	**ADD M**	4XXX	(M) + (A) ➔ A
Сложение с переносом	**ADC M**	5XXX	(M) + (A) + (C) ➔ A
Вычитание	**SUB M**	6XXX	(A) – (M) ➔ A
Переход, если перенос	**BCS M**	8XXX	Если (C) = 1, то M ➔ CK
Переход, если плюс	**BPL M**	9XXX	Если (N)= 0, то M ➔ CK
Переход, если минус	**BMI M**	AXXX	Если (N) = 1, то M ➔ CK
Переход, если ноль	**BEQ M**	BXXX	Если (Z) = 1, то M ➔ CK
Безусловный переход	**BR M**	CXXX	M ➔ CK
Приращение и пропуск	**ISZ M**	0XXX	M + 1 ➔ M, если (M) >= 0,то (CK) + 1 ➔ CK
Обращение к подпрограмме	**JSR M**	2XXX	(CK) ➔ M, M + 1 ➔ CK

Очистка аккумулятора	CLA	F200	0 ➔ A
Очистка рег. переноса	CLC	F300	0 ➔ C
Инверсия аккумулятора	CMA	F400	(!A) ➔ A
Инверсия рег. переноса	CMC	F500	(!C) ➔ C
Циклический сдвиг влево на 1 разряд	ROL	F600	Содержимое A и C сдвигается влево, A(15) ➔ C, C ➔ A(0)
Циклический сдвиг вправо на 1 разряд	ROR	F700	Содержимое A и C сдвигается вправо, A(0) ➔ C, C ➔ A(15)
Инкремент аккумулятора	INC	F800	(A) + 1 ➔ A
Декремент аккумулятора	DEC	F900	(A) – 1 ➔ A
Останов	HLT	F000	
Нет операции	NOP	F100	
Разрешение прерывания	EI	FA00	
Запрещение прерывания	DI	FB00	
Очистка флага	CLF B	E0XX	0 ➔ флаг устр.
Опрос флага	TSF B	E1XX	Если (флаг устр. B) = 1, то (СК) + 1 ➔ СК
Ввод	IN B	E2XX	(B) ➔ A
Вывод	OUT B	E3XX	(A) ➔ B

13. Организация вычислений в БЭВМ. Сдвиги, арифметические и логические операции. Цикл выборки команды.

Целые двоичные числа без знака можно использовать для представления нуля и целых положительных чисел. В 16-разрядном слове они могут изменяться от 0 до 65535. Это числа с фиксированной запятой. Целые двоичные числа со знаком используются, когда необходимо различать положительные и отрицательные числа. Отрицательные числа представляются в дополнительном коде. Это упрощает конструкцию ЭВМ. Сложение целых двоичных чисел со знаком и без знака выполняется в базовой ЭВМ с помощью команды ADD. По команде INC к содержимому аккумулятора прибавляется единица, а по команде DEC – единица вычитается. Если при этом возникает перенос из старшего разряда A, то в регистр переноса заносится 1, в противном случае в него заносится 0. Вычитание может выполняться путем сложения уменьшаемого и дополнительного кода вычитаемого. В базовой ЭВМ нет команд для выполнения умножения и деления (АЛУ не выполняет таких операций), поэтому произведение и частное необходимо получать программным путем. Для изменения знака числа необходимо его инвертировать, а затем прибавить единицу к младшему разряду. Побитовая обработка данных обеспечивается командами логического умножения, циклических сдвигов, а также командами инвертирования и очистки регистра переноса. Команда AND выполняет над каждым разрядом аккумулятора и содержимым ячейки булеву операцию «И». Результат выполнения команды для каждой пары битов операндов равен 1 только тогда, когда оба бита равны 1, а в остальных случаях бит результата равен 0, т.е. команда позволяет выделять или очищать определенные биты слова. Команды ROL и ROR замыкают аккумулятор и регистр переноса в кольцо и сдвигают все биты кольца влево или вправо. Сдвигами числа можно реализовать операции умножения или деления на 2 (один сдвиг), 4 (два сдвига), 8 (три сдвига) и т.д. Выборка команды: **1)** СК ---> РА, **2)** ОП(РА) ---> РД, СК + 1 ---> СК, **3)** РД ---> РК, **4)** Определение типа команды, вида адресации, **5*)** Выполнение безадресных команд и команд ввода---вывода.

14. Организация массивов данных. Косвенная адресация. Цикл выборки адреса.

Символьные данные представляются в БЭВМ в виде 8-разрядного знакового числа, которое формируется с использованием той или иной кодировки. Строковые данные формируются из массива символьных данных: т.к. аккумулятор и ячейки в памяти БЭВМ 2-х байтовые, то максимально мы можем уместить всего 2 символа в каждую ячейку памяти.

Есть два типа массивов, реализуемых в БЭВМ:
1) Массив, хранящий в себе количество обрабатываемых ячеек - выбирается индексная ячейка, содержащая адрес первого элемента массива. Дальше либо записывается число обрабатываемых ячеек в доп. коде в любую выбранную ячейку, либо первый элемент массива переводится в доп. код, пересылается в какую-нибудь ячейку, а потом мы работаем с ней через ISZ
2) Массив, содержащий в себе некоторый заданный стоп-символ

С количеством ячеек		
Адрес	Содержимое	Мнемоника / Комментарий
00A	0700	Индекс яч. с адресом
01F	Число в доп коде	Наш счетчик
030	F200	CLA
031	480A	ADD (00A) / Достаем I эл. Массива
. . .	Алгоритм по необходимой обработке	
0XX	001F	ISZ 01F
AHTUNG		
Желательно, чтобы в счетчике лежало количество СИМВОЛОВ в строке, иначе при нечетном количестве символов вы рискуете получить символ 0000 0000. И используйте соответствующий алгоритм		

Проверка ->

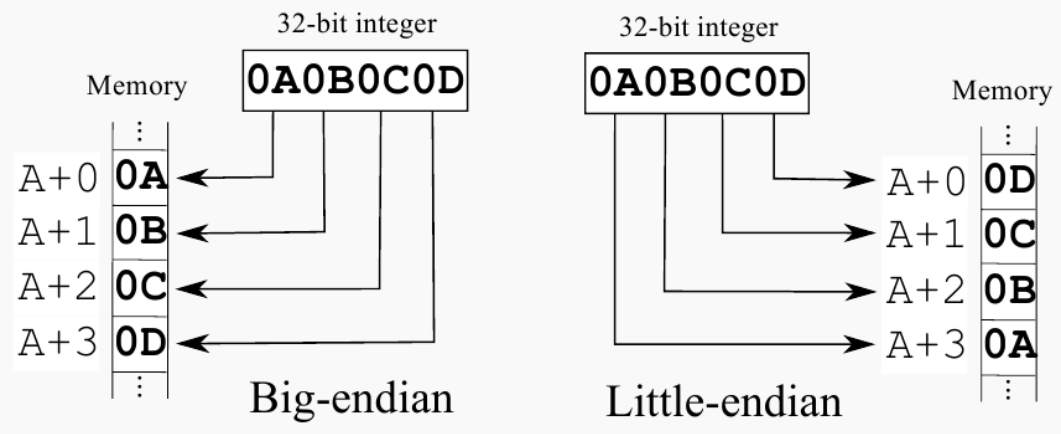
С стоп-символом		
Адрес	Содержимое	Мнемоника / Комментарий
00A	0700	Индекс яч. с адресом
01F	Код стоп-символа	То, на чем все закончится
020	Буфер	Чтобы возвращаться
021	00FF	Стиралка старших битов
030	F200	CLA
031	480A	ADD (00A) / Достаем I эл. массива
032	3020	MOV 020 / Сохраняем все
033	1021	AND 021 / Стираем старший байт
034	601F	SUB 01F / Проверяем симв.
035	BXXX	BEQ / Переход к завершающему действию программы, если 0
036	F200	CLA
037	4020	ADD 020 / Возврат, всего назад
. . .	Алгоритм по необходимой обработке	

Реализации следующие:

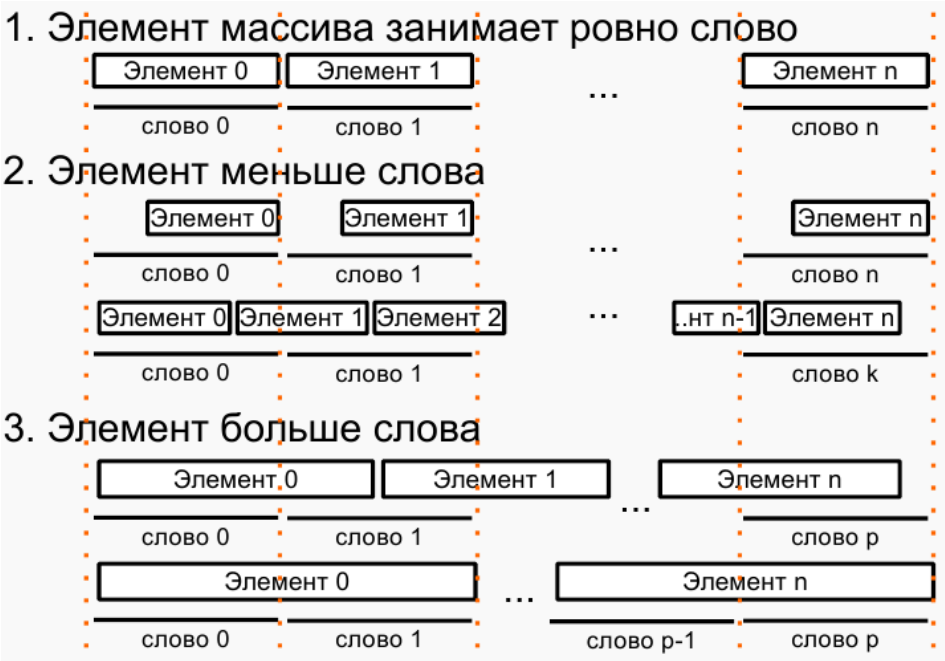
Выборка адреса следует за циклом выборки для адресных команд с косвенной адресацией:**1)** РД ---> РА, **2)** ОП(РА) ---> РД. Если косвенно адресуется одна из индексных ячеек, то данный цикл продолжается: **3)** РД + 1 ---> РД, **4)** РД ---> ОП(РА), **5)** РД – 1 ---> РД.

Little-endian, Big-endian

Little Endian – от младшего к старшему, Big Endian – наоборот.



Представление одномерных массивов



15. Управление вычислительным процессом в БЭВМ. Цикл исполнения команды ISZ.

Задача управления вычислительным процессом, т.е. требуемой последовательностью выполнения команд, решается в базовой ЭВМ при помощи команд перехода (BCS, BPL, BMI, BEQ, BR), команды «Приращение и пропуск» (ISZ) и «Останов» (HLT).

Разветвления в программах организуются с помощью команд перехода. Они не изменяют содержимое аккумулятора и регистра переноса, а лишь изменяют содержимое СК, помещая в него адрес, определяемый адресной частью команды.

BCS M (переход, если перенос) – переход к команде, расположенной по адресу M, если C=1.

BPL M (переход, если плюс) – переход к команде по адресу M, если A>=0 (в старшем бите A содержится 0).

BMI M (переход, если минус) - переход к команде по адресу M, если A<0 (в старшем бите 1).

BEQ M (переход, если ноль) - переход к команде по адресу M, если A=0 (во всех битах 0).

Если условие, указанное в этих командах, не выполняется, то переход не осуществляется, и выполняется команда, следующая за командой условного перехода.

BR M (безусловный переход) - переход к команде по адресу M при любых значениях регистров ЭВМ.

Циклические программы используются в тех случаях, когда требуется несколько раз выполнить набор одинаковых действий над некоторым изменяющимся набором данных. Для организации циклических программ часто используют команды перехода и команду ISZ, которая служит для увеличения на 1 содержимого адресуемой ячейки памяти и перехода к одному из двух путей продолжения программы в зависимости от знака этого содержимого (если содержимое меньше 0, то выполняется команда, следующая за ISZ, в противном случае, эта команда пропускается, т.е. выполняется команда через одну за ISZ). Очень ценно, что ISZ не портит содержимого A и C (увеличение ячейки производится без их участия).

Также для циклов часто используют косвенную адресацию индексных ячеек. Если произвести косвенную адресацию какой-либо из индексных ячеек, то сначала ее содержимое будет использовано в качестве адреса операнда, а затем оно автоматически увеличится на 1.

16. Подпрограммы в БЭВМ. Передача параметров. Цикл исполнения команд перехода и возврата из подпрограммы. Стек.

Достаточно часто встречаются ситуации, когда отдельные части программы должны выполнить одни и те же действия по обработке данных. В подобных случаях повторяющиеся части программы выделяют в подпрограмму. В базовой ЭВМ для этой цели используется команда JSR. При оформлении подпрограммы перед ее первой командой следует разместить ячейку, в которую будет пересылаться адрес возврата из подпрограммы. В команде обращения к подпрограмме указывается адрес именно этой ячейки, и команда JSR M выполняет следующие действия:

помещает в ячейку M адрес следующей за JSR M команды

передает управление команде, расположенной по адресу M+1

Последней командой подпрограммы должна быть команда выхода (команда BR (M)). По ней осуществляется переход к команде, адрес которой хранится в ячейке M.

СТЕК – память типа LIFO. Обмен информацией между ЭВМ и стеком всегда выполняется через верхнюю ячейку – вершину стека. Стековая память очень удобна для упрощения решения многих задач, возникающих при работе с подпрограммами, обслуживания прерываний и т.д.

В БЭВМ нет аппаратной реализации стека, но его можно смоделировать. При этом в качестве стека используют просто часть адресной памяти. LIFO обеспечивается с помощью указателя стека (SP), который содержит адрес плавающей вершины стека.

Стек предусматривает 2 операции: загрузка нового слова (уменьшается содержимое SP, и слово записывается в ячейку, на которую он указывает) и извлечение слова из стека (читается содержимое ячейки, на которую указывает SP, а потом он увеличивается).

Push: --SP, MOV

Pop: MOV, SP++

Варианты передачи данных в подпрограмму:

- Аккумулятор (Регистры Общего Назначения)
- Адресуемые ячейки памяти
- Стек – нет в БЭВМ, но можно реализовать (делать этого я, конечно, не буду)
- Регистровые окна

17. Организация ввода-вывода в вычислительных системах. Инициация обмена, передача информации и завершение обмена.

К ЭВМ можно подключать большое число разнообразных устройств ввода-вывода или внешних устройств (ВУ). Эти устройства передают в ЭВМ и получают из нее большой объем информации, который не может быть размещен только в регистрах процессора. Поэтому информация передается из ВУ в память ЭВМ и поступает на ВУ из ее памяти. При этом обмен может идти под управлением программы ЭВМ через регистры процессора (программно-управляемая передача данных) или под управлением специального внешнего устройства (контроллера прямого доступа в память), минуя процессор (передача данных при прямом доступе к памяти).

Программно-управляемый обмен осуществляется малыми порциями, при прямом доступе к памяти информация передается большими блоками. При использовании программно-управляемого обмена должна быть составлена программа, обеспечивающая пересылку данных из памяти ЭВМ в аккумулятор и далее в регистр памяти контроллера ВУ (вывод данных) или из регистра данных контроллера ВУ в аккумулятор и затем в память ЭВМ (ввод данных). В этой программе можно реализовать один из трех типов обмена: синхронный, асинхронный и по прерыванию.

К операциям ввода/вывода относятся все операции, управляющие работой ВУ и обеспечивающие передачу данных между ВУ и ОП.

Операции в/в распространяются и на внешние ЗУ и на устройства в/в и образуют один класс внешних (периферийных) устройств компьютера.

Все ВУ подключаются к интерфейсам В/В:

Каждый контроллер выполняет две основные функции:

1. Преобразует стандартную последовательность сигналов интерфейса в последовательность сигналов, обеспечивающих управление работой ВУ.

2. Контроллер управляет работой устройства с целью доступа и передачи затребованных данных.

18. Организация ввода-вывода в БЭВМ. Устройства ввода-вывода, команды.

В базовой ЭВМ реализована только программно-управляемая передача данных, т.е. обмен данными происходит только под управлением программы ЭВМ через регистры процессора.

В БЭВМ используются простейшие внешние устройства: два устройства ввода (ВУ-1, ВУ-2) и одно устройство вывода (ВУ-3). Устройства представлены 8-разрядными регистрами данных. Через регистры данных ВУ-1 и ВУ-2 информация может быть введена в базовую ЭВМ, а в регистр данных ВУ-3 принята из базовой ЭВМ.

Между ВУ и процессором установлены простейшие контроллеры, каждый из которых содержит: регистр данных (для обмена данными между ВУ и процессором), дешифратор адреса (позволяющий выделить обращение к данному ВУ среди всех обращений у устройствам ввода-вывода, подключенным к процессору), дешифратор приказов (декодирующий приказ от процессора на выполнение тех или иных операций) и регистр состояния (в котором хранится информация о готовности ВУ к обмену данными с процессором). В контроллерах простейших ВУ обычно используется однобитовые регистры состояния, которые часто называют флагом. Контроллеры ВУ связаны с процессором шинами ввода и вывода информации, шиной адреса и шиной управления, по которым передаются приказы от процессора и сведения о состоянии ВУ.

Команды ввода-вывода БЭВМ имеют одинаковый формат: 4-разрядный код операции, 4-разрядный код приказа, 8-разрядный кода выборки ВУ (адрес). Адрес позволяет связать процессор с одним из подключенных к нему ВУ (их может быть 2^8=256).

CLF B (E0xx, xx – код выборки ВУ) – устанавливает в исходное состояние (сбрасывает флаг) ВУ с кодом выборки B.

TSF B (E1xx) – помещает в регистр состояний ВУ содержимое флага ВУ с кодом выборки B. Затем процессор суммирует содержимое регистра Ф с содержимым СК.

IN B (E2xx) – пересылает в 8 младших бит A содержимое РДВУ с кодом выборки B.

OUT B (E3xx) - пересылает содержимое 8 младших бит A в РДВУ с кодом выборки B.

Когда в процессе работы ЭВМ УУ обнаруживает, что в РК и в РД находится команда с кодом операции $(1110)_2$, производится переход в режим ввод-вывод и на все контроллеры ВУ подается содержимое 12 младших разрядов РД. Дешифраторы адреса всех контроллеров декодируют это содержимое, но срабатывает тот из них, адрес которого совпадает с адресом в команде. Этот дешифратор открывает вентили для передачи информации на дешифратор приказов, который декодирует содержимое 8-11 бита РД и выдает приказ на выполнение одной из команд: открывает вентиль для передачи в ЭВМ состояния флага (TSF), обнуляет флаг (CLF) или открывает вентили, связывающие A с РД контроллера (IN или OUT).

19. Организация асинхронного обмена в БЭВМ. Пример программы.

Ввод двух символов с устройства ввода

```
ORG     005
L8:     WORD    FFF8  ; -8 количество сдвигов
RES:    WORD    ?     ; Ячейка для записи слова "ДА"
ORG     020           ; Первая команда – адрес 020
BEGIN:  CLA
SPIN1:  TSF     2     ; Ожидание ввода первого символа
        BR      SPIN1 ;
        IN      2     ; Ввод первого символа
        CLF     2     ; Очистка флага. Можно вводить дальше
RL:     ROL           ; Сдвиг первого символа
        ISZ     L8    ; старший байт аккумулятора
        BR      RL
SPIN2:  TSF     2     ; Ожидание ввода второго символа
        BR      SPIN2
        IN      2     ; Ввод второго в младшие 8 разрядов A
        CLF     2
        MOV     RES
        HLT
```

Сколько циклов команд БЭВМ будет ждать ввода второго символа?

Алгоритм программы асинхронного обмена: сначала проверяется готовность ВУ к обмену и если оно готово, дается команда на обмен (ввод или вывод). ВУ сообщает о готовности установкой в единицу флага в контроллере ВУ. При асинхронном обмене ЭВМ должна тратить время на ожидание момента готовности, а так как готовность проверяется командным путем (команда TSF), то в это время ЭВМ не может выполнять никакой другой работы по преобразованию данных.

20. Организация обмена по прерыванию программы в БЭВМ. Пример программы. Цикл прерывания.

Используется при работе с низкоскоростными ВУ или когда момент передачи данных заранее неизвестен. Обмен данными между ЭВМ и ВУ инициируется сигналом с ВУ. Для реализации данного типа обмена используется аппаратная проверка наличия внешнего прерывания, т. е. сигнала готовности по линии "Запрос прерывания".

По завершении цикла исполнения текущей команды происходит переход к циклу прерывания, который есть у всех команд кроме EI (Разр. прер.), DI (Запр. прер.) и HLT (Останов). Если в этот момент на линии «Запрос прерывания» нет сигнала о готовности ВУ или прерывания запрещены, то исполняется следующая команда. Иначе, прерывания запрещаются, в ячейку с адресом 000 заносится содержимое СК, и управление передается команде, расположенной в ячейке 001, с которой начинается программа обработки прерываний.

Программа обработки прерываний запоминает в памяти содержимое А в ячейку SAVED_A и содержимое С в ячейку SAVED_C. Т.е. минимальная информация о прерванной программе хранится в ячейках 000, SAVED_A и SAVED_C.

Производится поиск источника прерываний, и переход к последовательности действий по работе с конкретным ВУ. Затем выполняется передача данных и сброс флага готовности ВУ.

Восстановление значений A и C. Разрешение прерывания. И возврат к выполнению прерванной программы (командой BR(000)).

```
                    Обработка прерываний
  Готовность ВУ1: 2*A→РДВУ1, Готовность ВУ3: РДВУ3→ яч.29
        ORG 000
RET:    WORD ?   ; Ячейка адреса возврата
        NOP      ; Ячейка отладочной точки
                 ; останова (NOP/HLT)
        BR INT   ; Переход к обработке прерываний
        ORG 020  ; Основная программа
BEGIN:  EI       ; Установка состояния разрешения прерывания
        CLA      ; Первоначальная очистка аккумулятора
LOOP:   INC      ; Цикл для наращивания
        BR LOOP  ; содержимого аккумулятора

        ORG 029  ; Ячейка для хранения кодов,
IO3:    WORD  ?  ; поступающих с ВУ-3
```

```
RESTORE:NOP          ; отладочная точка останова (NOP/HLT)
        CLA          ; Восстановление содержимого C и A
        ADD SAVED_C
        ROR
        CLA
        CMA
        AND SAVED_A
        EI           ; Возобновление состояния
                     ; разрешения прерывания
        BR (RET)     ; Возврат из обработки прерывания
SAVED_A:WORD ?
SAVED_C:WORD ?
```

```
        ORG   030
INT:    MOV   SAVED_A  ; Сохранение содержимого A и C
        ROL
        MOV   SAVED_C
```

```
         TSF 3       ; Опрос флага ВУ-3
         BR CHECK1   ; Если сброшен → к опросу флага ВУ-1
         BR READY3   ; Переход на ввод данных из ВУ-3
CHECK1:  TSF 1       ; Опрос флага ВУ-1
         BR READY2   ; Если сброшен → к опросу флага ВУ-2
         BR READY1   ; Переход на вывод данных в ВУ-1
READY3:  CLA         ; Ввод данных из ВУ-3
         IN 3
         CLF 3       ; Сброс флага ВУ-3
         MOV IO3     ; Пересылка значения в ячейку 29
         BR RESTORE
READY1:  CLA
         ADD SAVED_A
         ROL
         OUT 1       ; Вывод на ВУ-1 8-ми
         CLF 1       ; Сброс флага ВУ-1
         BR RESTORE
READY2:  CLF 2
```

21. Понятие многоуровневой ЭВМ. Понятие и пример программы на разных уровнях.

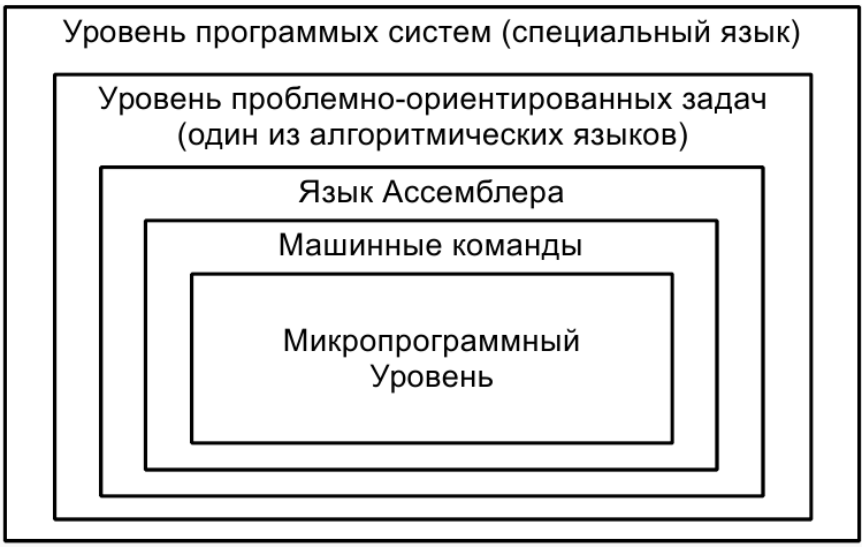
Возможность исполнения на ЭВМ программы, написанной на алгоритмическом языке, обеспечивается с помощью специальных системных программ: компиляторов и интерпретаторов. Компиляция, заключается в том, что процесс выполнения алгоритма осуществляется лишь после завершения процесса перевода исходной программы. В интерпретации же каждый оператор исходной программы заменяется программой-интерпретатором на эквивалентную последовательность машинных команд непосредственно перед исполнением. В отличие от компиляции, в интерпретации во время решения задачи машине нужны и исходная программа, и программа-интерпретатор.

Затраты на создание компиляторов (интерпретаторов) и время на процесс перевода программы в значительной мере определяются сходством компилируемого и получаемого языков. Поэтому алгоритмические языки не сразу переводят программу на язык машинных команд. Существует определенная иерархия языков программирования, в которой более сложный язык базируется на предшествующем.

Примером промежуточного языка служит язык символического кодирования команд, часто называемый языком ассемблера. Языки ассемблеров (разработанные для каждого типа ЭВМ) - это первые средства автоматизации программирования в вычислительной технике. В них допускается использование символических имен и меток. Компиляторы с таких языков называются ассемблерами. Они отводят определенные ячейки памяти для символических переменных, организуют связи между различными частями программы, что резко облегчает программирование по сравнению с программированием на уровне команд.

Человеку, работающему с ЭВМ на том или другом языке, чаще всего кажется, что язык, на котором он общается с ЭВМ, – это ее машинный язык. Следовательно, разным пользователям одной и той же ЭВМ может казаться, что они работают на разных вычислительных машинах. Отсюда появились понятия: виртуальная (кажущаяся) ЭВМ и многоуровневая ЭВМ.

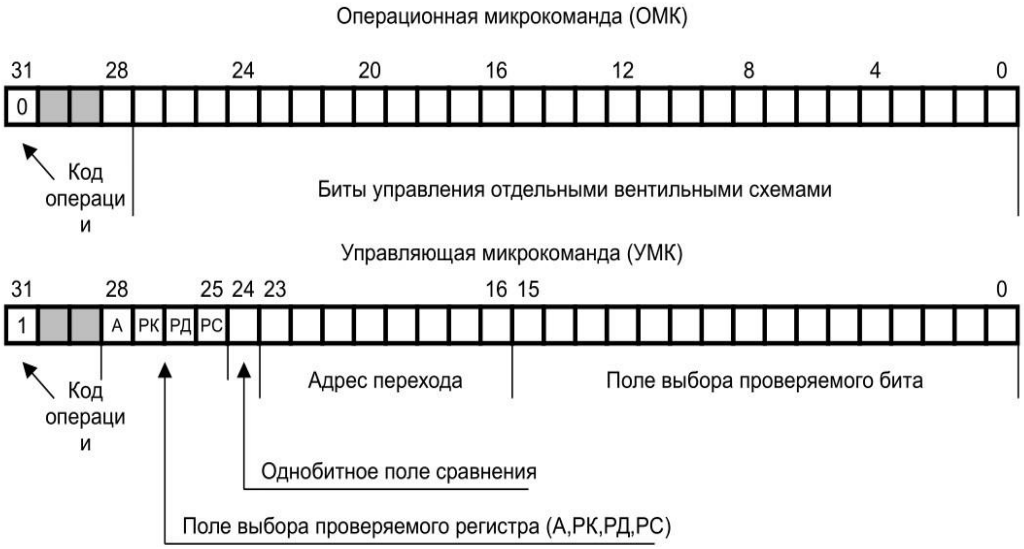
Многоуровневая ЭВМ – это вычислительная машина, имеющая средства для работы с n различными уровнями языков программирования. Нижний язык, или уровень, является наиболее простым, верхний –наиболее сложным. Такую машину можно рассматривать как n различных виртуальных машин, каждая из которых имеет свой машинный язык. Сложность аппаратурной реализации этих виртуальных машин возрастает по мере увеличения номера уровня.



22. Микропрограммный уровень БЭВМ. Форматы горизонтальных и вертикальных микрокоманд.

Если рассматривать БЭВМ как многоуровневую машину, то оказывается, что на её нижнем уровне выполняются элементарные действия (микрооперации) над словами информации. Управление порядком следования микроопераций осуществляется с помощью устройства управления БЭВМ, которое, в свою очередь, является очень простой ЭВМ. Для этой ЭВМ регистры и вентильные схемы базовой ЭВМ служат как бы устройствами ввода и вывода. Программа работы такой ЭВМ – микропрограммного устройства управления (МПУ) – называется **микропрограммой**, а ее команды, содержащие информацию об элементарных действиях, подлежащих выполнению в течение одного рабочего такта ЭВМ, – **микрокомандами**.

Микропрограмма обычно хранится в постоянном запоминающем устройстве - памяти микрокоманд, но в эмуляторе БЭВМ реализована возможность изменения микропрограммы. В каждом такте работы ЭВМ из этой памяти в регистр микрокоманд (РМК) пересылается очередная микрокоманда, т.е. микрокоманда, на которую указывает счетчик микрокоманд (СчМК), одновременно выполняющий функции регистра адреса микрокоманд.



Если из памяти выбрана **операционная микрокоманда** (ОМК), то в 31-й бит РМК, записывается 0. Этот сигнал через инвертор НЕ открывает вентильную схему ВРО и обеспечивает передачу соответствующих битов РМК (управляющих сигналов У0-У28).

Разряды РМК, содержащие 1, создают открывающий управляющий сигнал, а содержащие 0 – закрывающий. Подобная структура микрокоманд, где каждый бит используется для создания отдельного управляющего сигнала, называется **горизонтальной**.

Если из памяти выбрана **управляющая микрокоманда**, то в 31-ый бит РМК записывается 1. Этот сигнал открывает вентильную схему БР1 и тем самым создает условия для выполнения УМК. Теперь по сигналу, создаваемому каким-либо битом поля выбора проверяемого регистра (У1, У2, У4 или У5) открывается из вентильных схем В1, В2, В4 или В5 и на вентили ВВ0-ВВ15 поступает через АЛУ содержимое соответствующего регистра (РД, РК, А или РС). Одновременно на эти же вентили поступает с РМК содержимое поля выбора проверяемого бита. Так как в этом поле записана только одна 1 (на месте, соответствующем проверяемому биту), то открывается лишь один из вентилей ВВ0-ВВ15, через который на схему сравнения поступает содержимое проверяемого бита из проверяемого регистра. На другой вход этой схемы поступает содержимое однобитового поля сравнения (24-ый бит УМК), в которое при кодировании УМК записали 0 или 1.

Если проверяемый бит и бит из поля сравнения идентичны, то схема сравнения формирует единичный сигнал, который открывает вентильную схему ВА и на СчМК пересылается адрес перехода (16-24-ый биты УМК). В противном случае на СчМК передается нулевое значение, которое инициирует увеличение значения СчМК на единицу (используется схема ИЛИ-НЕ для всех битов адреса перехода).

У описываемой здесь ЭВМ имеются две памяти, два набора команд и две программы: традиционного машинного уровня и микропрограммного уровня. Однако аппаратно реализованный процессор - только один, а архитектура машины формирует микропрограммный уровень.

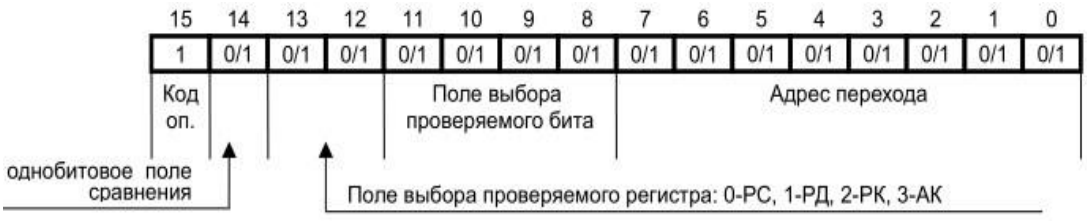
Так как в ЭМВ больших размеров могут быть сотни вентильных схем горизонтальная команда может быть неприемлемо длинной, поэтому используют кодируемые поля. Кодируемые поля, шифруют вентили, которые не могут быть открыты одновременно.

В управляющей микрокоманде сокращение её разрядов можно обеспечить за счет кодирования полей выбора проверяемого регистра и проверяемого бита в этом регистре, что позволяет сократить УМК до 16 бит. Однако при декодировании таких сжатых полей требуется использовать дешифратор. Для декодирования операционных микрокоманд ОМК1 и ОМК2 потребуется ещё 8 дешифраторов. Но экономия затрат на память микрокоманд будет превышать стоимость этих дешифраторов.

Сокращенная путем кодирования горизонтальная микрокоманда называется **вертикальной микрокомандой**.

ОМК0

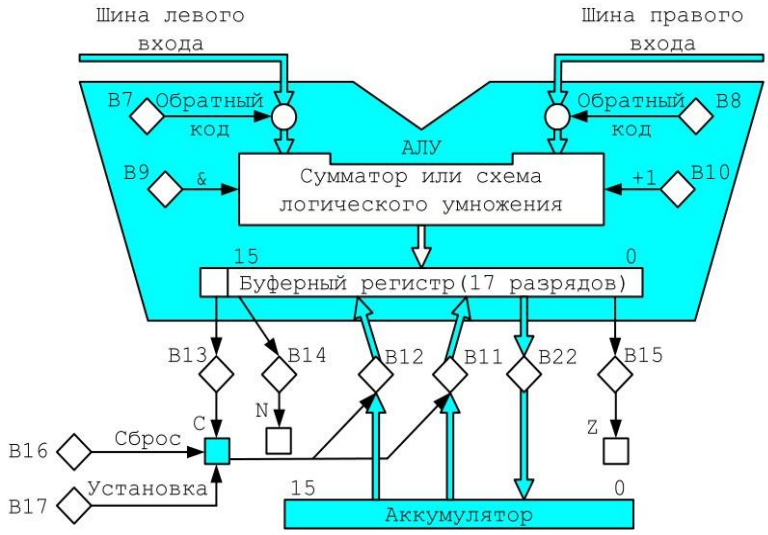
ОМК1



УМК

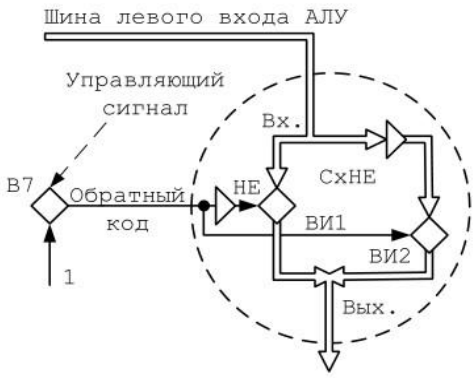
23. Работа арифметико-логического устройства

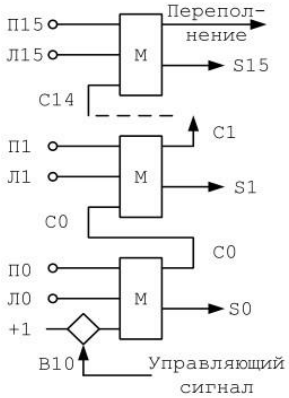
АЛУ использует А и РД в качестве операндов для получения результата, который помещается в аккумулятор. Все арифметические и логические команды БЭВМ и вспомогательные арифметические операции (например, увеличение на единицу содержимого СК) можно выполнить с помощью АЛУ.



Обратный код входных сигналов АЛУ получается по схеме, которая приведена слева снизу. В данной схеме изменение всех 0 на 1 и всех 1 на 0 осуществляется с помощью инверторов СхНЕ. При отсутствии управляющего сигнала единица не проходит на выход В7, вентиль ВИ2 закрыт, на выходе инвертора НЕ единичный сигнал, значит вентиль ВИ1 открыт. Тогда в узле Вых. – прямой код операнда (равен коду в узле Вх.). Когда на В7 подается единичный управляющий сигнал, закрывается ВИ1, открывается ВИ2 и в узел Вых. подается обратный по отношению к узлу Вх. код.

Сложение осуществляется типовой схемы сумматора, которая приведена снизу справа. С помощью управляющего сигнала, подаваемого на вентиль В10, можно обеспечить прибавление к сумме единицы: S = Л+П+1. Перенос из старшего разряда (номер 15) сигнализирует о переполнении сумматора. Этот перенос записывается в старший разряд БР(номер 16) и может быть переписан в регистр переноса С, путем подачи сигнала на вентиль В13.





Сумматор, вентильная схема и инвертор – комбинационные схемы, т.е. схемы, в которых значения выходных переменных полностью определяются значениями входных переменных в данный момент времени. Это позволяет выполнять несколько различных микроопераций при разных сочетаниях управляющих сигналов, подаваемых на вентили схемы В1-В8 и В10:

- Суммирование двух регистров (ADD - В1 и В4).

- Суммирование двух регистров и 1 (B1, B4 и B10).
- Вычитание (SUB – B1, B4, B8 и B10)
- Добавление единицы к какому-нибудь регистру (B10 и вентильная схема, устанавливающая связь сумматора с соответствующим регистром)
- Вычитание единицы из какого-нибудь регистра (B4 и BS)
- Инвертирование содержимого какого-либо регистра (CMA – B4 и B7)
- Очистка какого-либо регистра (B1-B10 закрыты, открыт нужный вентиль [для CLA открыт B22]) 8) и так далее.

Чтобы интерпретируемая программа выполняла программу, необходимо в определенной последовательности открывать и закрывать вентильные схемы. Описание того, какую схему и когда открывать, составляет программу (микропрограмму) для машины, система команд которой включает команду «Открыть вентильную схему».

Логическое умножение осуществляется с помощью вентильной схемы, на входы которой подаются коды операндов, а на выходе образуется искомый результат.

В АЛУ сумматор и схема логического умножения объединены в один блок, в котором входные шины разветвляются на оба эти устройства, а выходы устройств подключены к БР. Но так как по сигналу с B9 открыта лишь одна вентильная схема, то на БР поступит результат только с одного устройства. При отсутствии сигнала с B9 в БР записывается сумма, а иначе результат логического умножения.

Циклический сдвиг на один разряд вправо или влево производится путем подачи единичного управляющего сигнала на вентильную схему B11 или B12 соответственно, а затем на вентильные схемы B13 и B22.

Установка признаков переноса из старшего разряда сумматора, а также отрицательно или нулевого значения результата осуществляется с помощью посылки управляющих сигналов на вентили B13, B14 и B15 соотв.

Сигнал на **B13**: запись 16 разряда БР в регистр C.
Сигнал на **B14**: запись знакового 15 разряда БР в регистр N.
Сигнал на **B15**: запись 1 в регистр Z, если во всех 16 разрядах БР содержатся нули (схема ИЛИ-НЕ).
Сигнал на **B16**: запись 0 в регистр C.
Сигнал на **B17**: запись 1 в регистр C.

24. Регистр состояния БЭВМ, вентильные схемы.

При организации ветвлений в микропрограмме используется содержимое РС, являющегося объединением однобитовых признаков результата и служебных битов состояния ЭВМ
Вентильные схемы – это электронные ключевые схемы, предназначенные для управления потоком информации из регистров в шины и обратно. На один вход подается информационный сигнал, на другой – управляющий.

Вентильные схемы B1, B2, B3 предназначены, соответственно, для передачи содержимого РД, РК, СК на правый вход АЛУ. Если все эти схемы закрыты (У1=У2=У3=0), то сигнал на правом входе АЛУ соответствует коду числа 0.

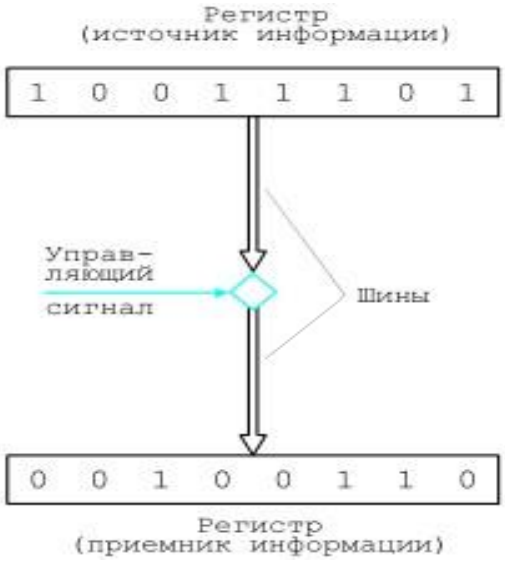
Аналогично используются вентильные схемы B4, B5, B6, позволяющие передать на левый вход АЛУ содержимое А, РС, КР или кода числа 0.

Управляющие сигналы У7-У10 перестраивают АЛУ на выполнение различных микроопераций. При У7=...У10=0 в 17-разрядный буферный регистр АЛУ (БР) записывается сумма входных сигналов АЛУ: при У7=У8=У9=0 и У10=1 к такой сумме добавляется 1; при У7=У8=У10=0 и У9=1 в БР записывается результат логического умножения входных сигналов АЛУ; при У7=1 и (или) У8=1 можно получить аналогичные результаты, но для инверсных значений одного или двух входных сигналов.

Вентильные схемы В11 и В12 позволяют записать в БР сдвинутое на один разряд вправо или влево содержимое аккумулятора. При этом "лишний" разряд БР заполняется содержимым регистра переноса С.

Регистр состояния

Разряд	Содержимое
0	Перенос (С)
1	Нуль (Z)
2	Знак (N)
3	0 - используется для организации безусловных переходов в МПУ
4	Разрешение прерывания
5	Прерывание
6	Состояния ВУ (Ф)
7	Состояние тумблера РАБОТА/ОСТАНОВ (1 - РАБОТА)
8	Программа
9	Выборка команды
10	Выборка адреса
11	Исполнение
12	Ввод-вывод



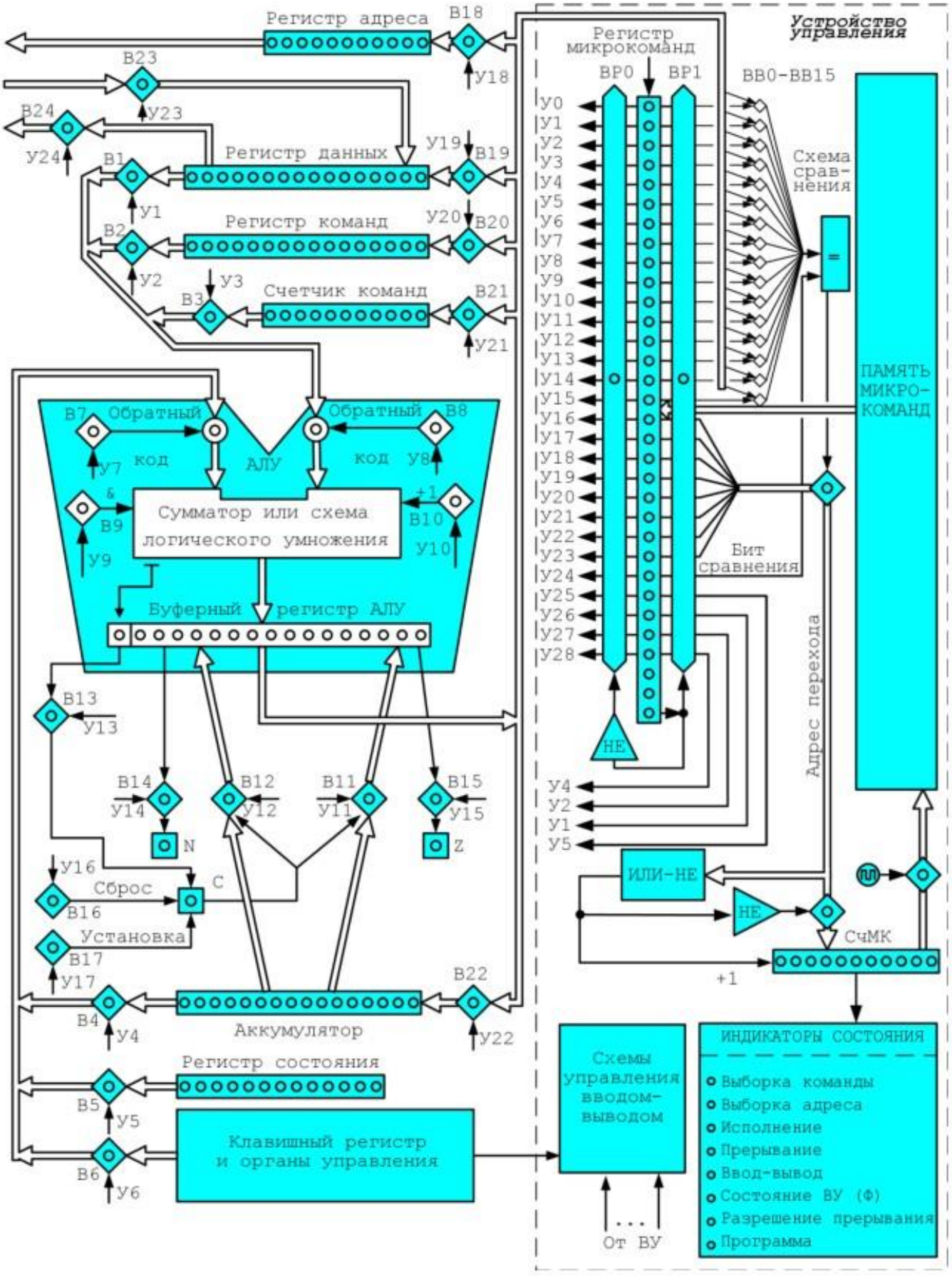
Вентильные схемы В13-В15 используются для передачи в младшие три бита регистра состояния, являющимися признаками результата С, N, Z операции, выполненной в АЛУ. В С и N копируются соответственно 16-й и 15-й бит БР. В Z записывается 1 в случае, если младшие 16 бит БР равны 0 (данная операция реализуется на аппаратном уровне схемой ИЛИ-НЕ).

Управляющие сигналы У16 и У17 позволяют установить признак С в 0 или 1 независимо от результата выполнения операции, сохраняемого в БР.

Вентильные схемы В18-В22 позволяют переписать содержимое 16 или 11 младших разрядов в РА, РД, РК, СК и А соответственно.

Вентильные схемы В23-В28 используются для организации обмена информацией между регистрами процессора и другими подсистемами ЭВМ (памятью и устройствами ввода-вывода).

Вентильная схема В0 используется в команде HLT для передачи сигнала прекращения выполнения программы и записывает 0 в 8-й бит РС.


Устройство управления
Регистр адреса
Регистр данных
Регистр команд
Счетчик команд
Регистр микрокоманд
ВР0
ВР1
ВВ0-ВВ15
Схема сравнения
ПАМЯТЬ МИКРО-КОМАНД
Обратный код
АЛУ
Сумматор или схема логического умножения
Буферный регистр АЛУ
Бит сравнения
Адрес перехода
ИЛИ-НЕ
НЕ
СчМК
Аккумулятор
Регистр состояния
Клавишный регистр и органы управления
Схемы управления вводом-выводом
От ВУ
ИНДИКАТОРЫ СОСТОЯНИЯ
Выборка команды
Выборка адреса
Исполнение
Прерывание
Ввод-вывод
Состояние ВУ (Ф)
Разрешение прерывания
Программа
Сброс
Установка

25. Микропрограммное управление вентильными схемами. Схема управления. Интерпретатор БЭВМ.

При появлении тактового импульса из памяти микрокоманд извлекается и загружается в РМК слово, на которое указывает СчМК, и к содержимому этого счетчика прибавляется единица. Если из памяти извлечена УМК, то в 31 - бите РМК содержится 1 (код операции УМК), которая открывает вентильную схему ВР1 и тем самым создает условия для исполнения УМК. Если же извлечена ОМК, то в 31---бите РМК – ноль. Этот сигнал с помощью инвертора НЕ открывает вентильную схему ВР0, и через нее на В0 - В28 передаются состояния соответствующих битов РМК. Разряды РМК, содержащие единицы, создают открывающий управляющий сигнал, а содержащие нули – закрывающий (У0---У28).

При исполнении УМК по сигналу, создаваемому каким-либо битом поля выбора проверяемого регистра, открывается одна из вентильных схем В1, В2, В4 или В5 и на вентили ВВ0-ВВ15 поступает через АЛУ содержимое соответствующего регистра. Одновременно на эти же вентили поступает с РМК содержимое поля выбора проверяемого бита. Так как в этом поле должна быть записана только одна единица, то открывается лишь один из вентилей ВВ0---ВВ15, через который на схему сравнения поступает содержимое проверяемого бита из проверяемого регистра.

На второй вход схемы поступает содержимое однобитового поля сравнения (24---й бит УМК), в которое при кодировании УМК записали цифру 0 или 1. Если проверяемый бит и бит из поля сравнения идентичны, то схема сравнения формирует единичный сигнал, который открывает вентильную схему ВА, и на СчМК пересылается адрес перехода (биты с 16 по 23). В противном случае на СчМК сохраняется адрес микрокоманды, расположенной вслед за исполняемой.

В системе команд базовой ЭВМ ряд кодов операций зарезервирован для включения новых команд. Это арифметическая команда 7xxx, команда перехода Dxxx и безадресные команды FC00, FD00, FE00 или FF00. Когда при декодировании команды выясняется, что выбрана команда 7xxx, производится передача управления к строке В0. Следовательно, часть микропрограммы, описывающая последовательность микроопераций по реализации этой команды, должна начинаться со строки В0. Сюда можно записать, например, действия по умножению или делению операндов. Часть микропрограммы, реализующая дополнительные безадресные команды, начинается от строки Е0. Первые микрокоманды этой части должны осуществить декодирование выбранной команды (определить расширение кода операции: C, D, E или F) и передать управление соответствующим микрокомандам нового куска микропрограммы. Так как память микрокоманд имеет 256 ячеек, то при составлении микропрограмм новых команд можно использовать лишь строки с номерами от AD до FF.

26. Архитектура ЭВМ. Гарвардская и фон-Неймановская архитектура. Организация обмена архитектуры ЭВМ с использованием шин.

Архитектура ЭВМ

Синяя Библия - стр. 100.

Архитектура ЭВМ – состав, характеристики и взаимосвязь устройств ЭВМ (структурная организация ЭВМ), принцип функционирования ЭВМ и ее машинный язык.

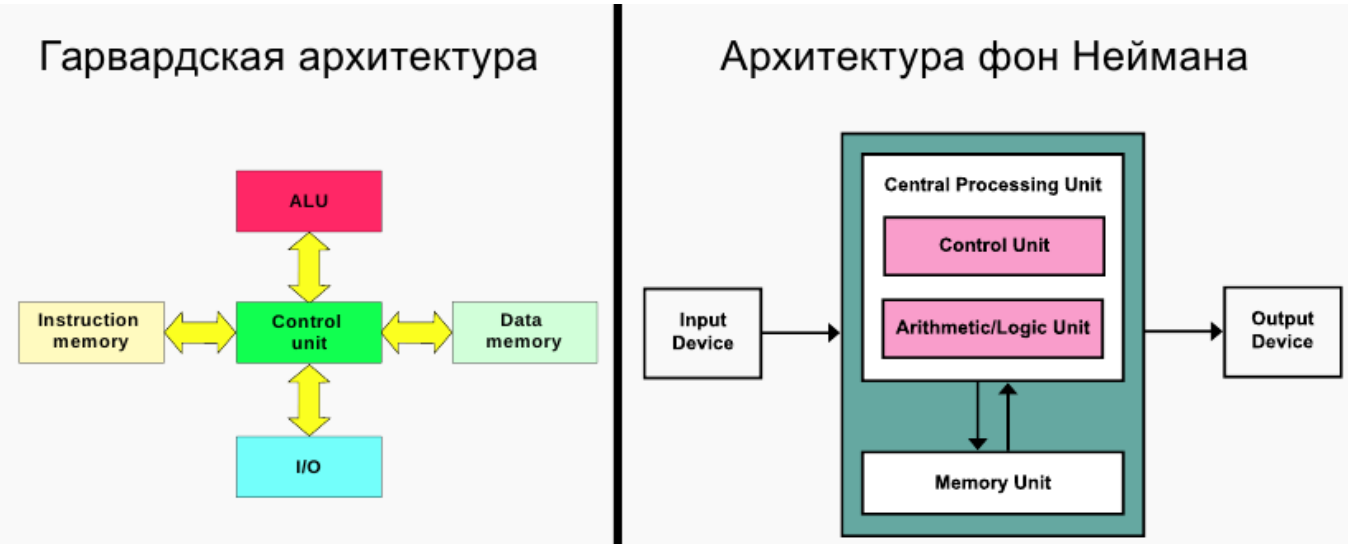
Принстонская архитектура, которая часто называется архитектурой Фон-Неймана, характеризуется использованием общей оперативной памяти для хранения программ, данных, а также для организации стека. Для обращения к этой памяти используется общая системная шина, по которой в процессор, по которой в процессор поступают и команда, и данные. Эта архитектура имеет ряд важных достоинств. Наличие общей памяти позволяет оперативно перераспределять ее объем для хранения отдельных массивов команд, данных и реализации стека в зависимости от решаемых задач. Таким образом, обеспечивается возможность более эффективного использования имеющегося объема оперативной памяти в каждом конкретном случае применения МП. Использование общей шины для передачи команд и данных значительно упрощает отладку, тестирование и текущий контроль функционирования системы, повышает ее надежность. Поэтом Принстонская архитектура в течение долгого времени доминировала в вычислительной технике.

Однако ей присущи и существенные недостатки. Основным из них является необходимость последовательной выборки команд и обрабатываемых данных по общей системной шине. При этом общая шина становится «узким местом», которое ограничивает производительность цифровой системы.

Гарвардская архитектура характеризуется физическим разделением памяти команд и памяти данных. Каждая память соединяется с процессором отдельной шиной, что позволяет одновременно с чтением-записью данных при выполнении текущей команды производить выборку и декодирование следующей команды. Благодаря такому разделению потоков команд и данных и совмещению операций их выборки реализуется более высокая производительность, чем при использовании Принстонской архитектуры.

Недостатки Гарвардской архитектуры связаны с необходимостью проведения большего числа шин, а также с фиксированным объемом памяти, выделенной для команд и данных, назначение которой не может оперативно перераспределяться в соответствии с требованиями решаемой задачи. Поэтому приходится использовать память большего объема, коэффициент использования которой при решении разнообразных задач оказывается более низким, чем в системах с Принстонской архитектурой. Однако развитие микроэлектронной технологии позволило в значительной степени преодолеть указанные недостатки, поэтому Гарвардская архитектура широко применяется во внутренней структуре современных высокопроизводительных МП, где используется отдельная кэш-память для хранения команд и данных. В то же время во внешней структуре большинства микропроцессорных систем (МПС) реализуются принципы Принстонской архитектуры.

Гаравардская архитектура. Фон-Неймановская архитектура



27. Архитектура многопроцессорных ЭВМ. Системный коммутатор. Архитектуры UMA и NUMA.

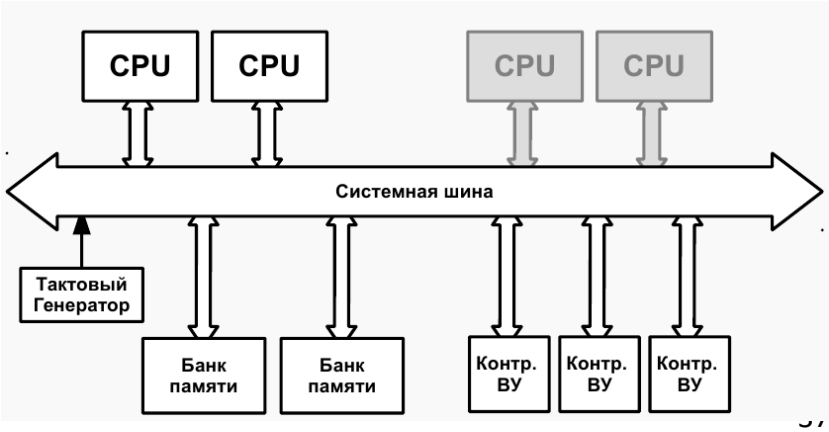
Процессоры современных микро ЭВМ (микропроцессоры) обычно изготавливают в виде одной большой интегральной схемы (БИС), имеющей до 300 тыс. элементов в кристалле. Такой микропроцессор должен соединяться системой шин с памятью и контроллерами ВУ, и эти соединения должны осуществляться через выводы БИС.

Сокращения выводов БИС добиваются за счет использования двунаправленных шин. Так, вместо шин «Чтение» и «Запись» можно иметь лишь одну шину, по которой в зависимости от значения дополнительного управляющего сигнала данные передаются либо из памяти в процессор, либо наоборот. Полученная таким образом структура называется структура с раздельными шинами.

Сходство процессов обмена информацией процессора – памяти и процессора – регистров контроллеров ВУ приводит к мысли об использовании одних и тех же выводов для связи с памятью и контроллерами ВУ. При такой структуре (с изолированными шинами) число выводов БИС уменьшается, но исчезает возможность одновременного обмена информацией между процессором, памятью и ВУ.

Дальнейшего сокращения выводов БИС микропроцессора с трехшинной структурой (ША, ШД, ШУ) можно добиться за счет объединения адресной шины и шины данных. При этом возникает так называемая мультиплексируемая шина, по которой в одни моменты времени передаются адреса, а в другие – данные.

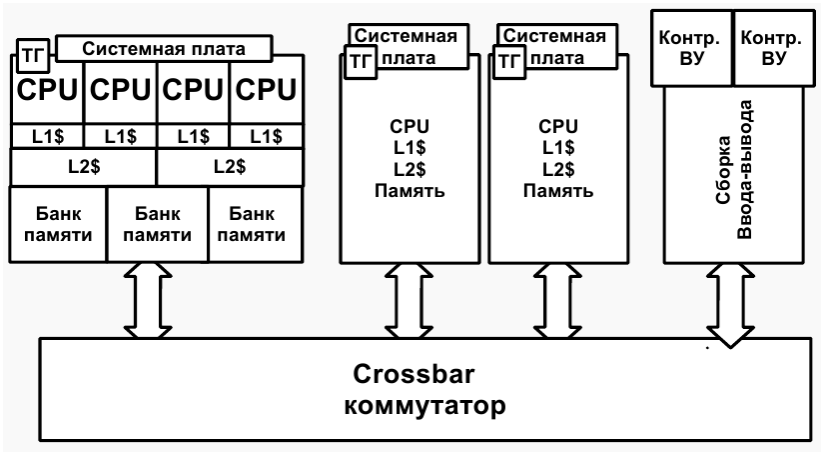
Структуры с вводом---выводом, организованным по аналогии с обращением к памяти– структуры с общими шинами. Процессор как бы обманывают, заставляя его выполнять операцию с регистром данных контроллера ВУ, тогда как процессор работает так, как если бы он имел дело с ячейкой памяти ЭВМ.



Uniform Memory Access (сокращённо UMA — «однородный доступ к памяти») — архитектура многопроцессорных компьютеров с

общей памятью. Все микропроцессоры в UMA---архитектуре используют физическую память одновременно. При этом время запроса к данным из памяти не зависит ни от того, какой именно процессор обращается к памяти, ни от того, какой именно чип памяти содержит нужные данные.

NUMA (Non---Uniform Memory Access — «неравномерный доступ к памяти» или Non---Uniform Memory Architecture — «Архитектура с неравномерной памятью») — схема реализации компьютерной памяти, используемая в мультипроцессорных системах, когда время доступа к памяти определяется её расположением по отношению к процессору.



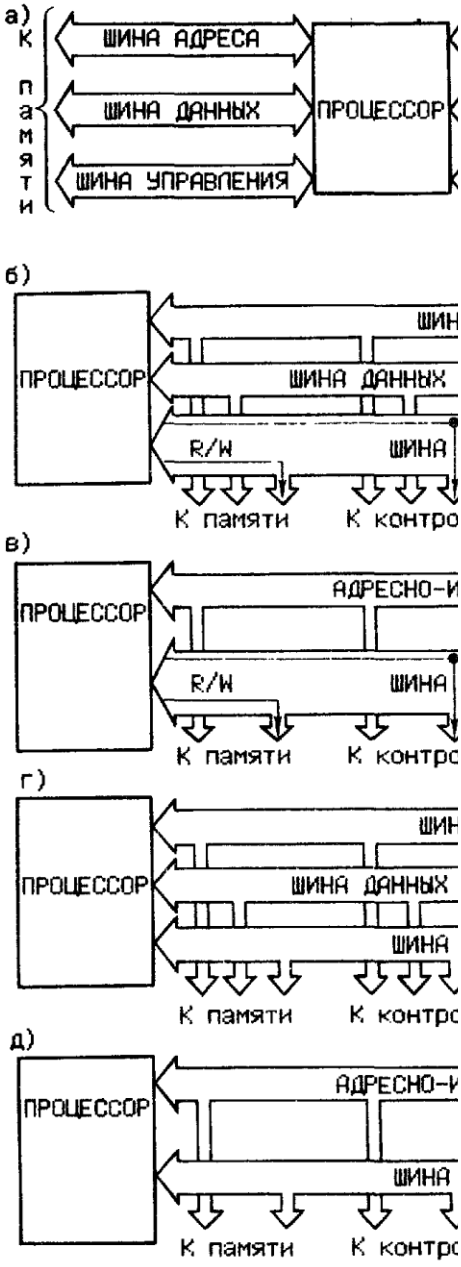


Рис. 5.2. Стру

а — с раздельными шинами, *б* — с изолиро
ми и мультиплексированием шин адресов и
шинами и мультиплексиро

28. Структура современных процессоров. Окружение процессора. CISC, RISC, VLIW.

Немного копипаста из лекций Бати и книги Цилькера:

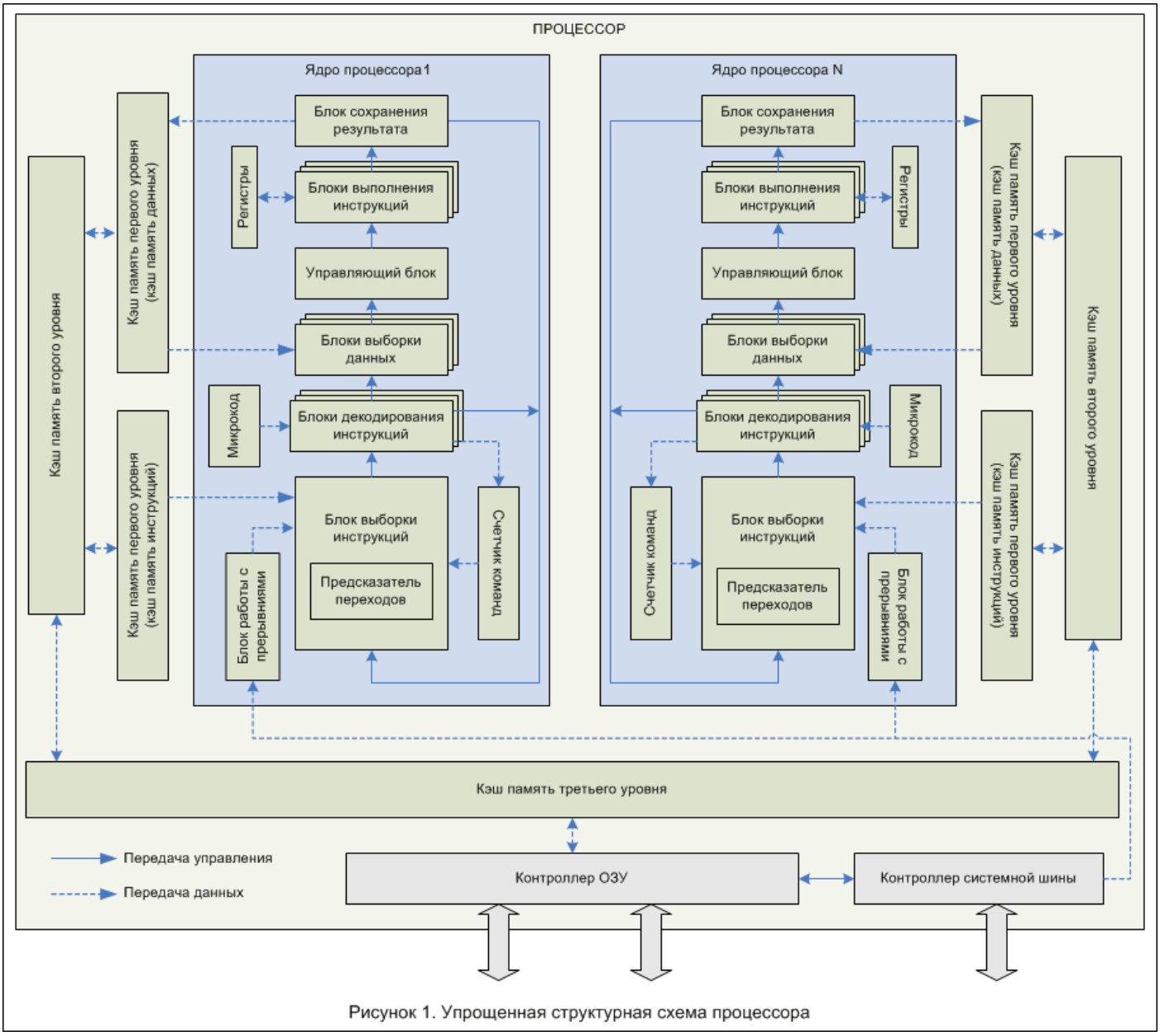


Рисунок 1. Упрощенная структурная схема процессора

Ядро процессора.

Ядро процессора – это его основная часть, содержащая все функциональные блоки и осуществляющая выполнение всех логических и арифметических операций.

http://all-ht.ru/inf/pc/cp_struct.html

Complex Instruction Set Computer

CISC - Концепция проектирования процессоров, которая характеризуется следующим набором свойств:

· большим числом различных по формату и длине команд;

· введением большого числа различных режимов адресации;

· обладает сложной кодировкой инструкции.

Процессору с архитектурой CISC приходится иметь дело с более сложными инструкциями неодинаковой длины. Выполнение одиночной CISC-инструкции может происходить быстрее, однако обрабатывать несколько таких инструкций параллельно сложнее.

Облегчение отладки программ на ассемблере влечет за собой загромождение узлами микропроцессорного блока. Для повышения быстродействия следует увеличить тактовую частоту и степень интеграции, что вызывает необходимость совершенствования технологии и, как следствие, более дорогого производства.

Для CISC-архитектуры типичны:

- наличие в процессоре сравнительно небольшого числа регистров общего назначения;
- большое количество машинных команд, некоторые из них аппаратно реализуют сложные операторы ЯВУ;
- разнообразие способов адресации операндов;
- множество форматов команд различной разрядности;
- наличие команд, где обработка совмещается С обращением к памяти.

Характеристика	CISC	RISC	VLIW
Длина команды	Варьируется	Единая	Единая
Расположение полей в команде	Варьируется	Неизменное	Неизменное
Количество регистров	Несколько (часто специализированных)	Много регистров общего назначения	Много регистров общего назначения
Доступ к памяти	Может выполняться как часть команд различных типов	Выполняется только специальными командами	Выполняется только специальными командами -

Reduced Instruction Set Computer

Процессор с сокращенным набором команд. Система команд имеет упрощенный вид. Все команды одинакового формата с простой кодировкой. Обращение к памяти происходит посредством команд загрузки и записи, остальные команды типа регистр-регистр. Команда, поступающая в CPU, уже разделена по полям и не требует дополнительной дешифрации.

Команда меньше загромождает ОЗУ, CPU дешевле. Программной совместимостью указанные архитектуры не обладают. Отладка программ на RISC более сложна. Поскольку RISC-инструкции просты, для их выполнения нужно меньше логических элементов, что в конечном итоге снижает стоимость процессора. Но большая часть программного обеспечения сегодня написана и откомпилирована специально для CISC-процессоров фирмы Intel. Для использования архитектуры RISC нынешние программы должны быть перекомпилированы, а иногда и переписаны заново.

Very Long Instructions Word

VLIW - архитектура процессоров с несколькими вычислительными устройствами. Характеризуется тем, что одна инструкция процессора содержит несколько операций, которые должны выполняться параллельно, базируется на RISC-архитектуре.

– Несколько инструкций, упакованных в одну команду
– Упаковка операций в инструкцию ложится на компилятор

Идея VLIW базируется на том, что задача эффективного планирования параллельного выполнения

нескольких команд возлагается на «разумный» компилятор. Такой компилятор вначале исследует исходную программу с целью обнаружить все команды, которые могут быть выполнены одновременно, причем так, чтобы это не приводило к возникновению конфликтов. В процессе анализа компилятор может даже частично имитировать выполнение рассматриваемой программы. На следующем этапе компилятор пытается объединить такие команды в пакеты, каждый из которых рассматривается так одна сверхдлинная команда.

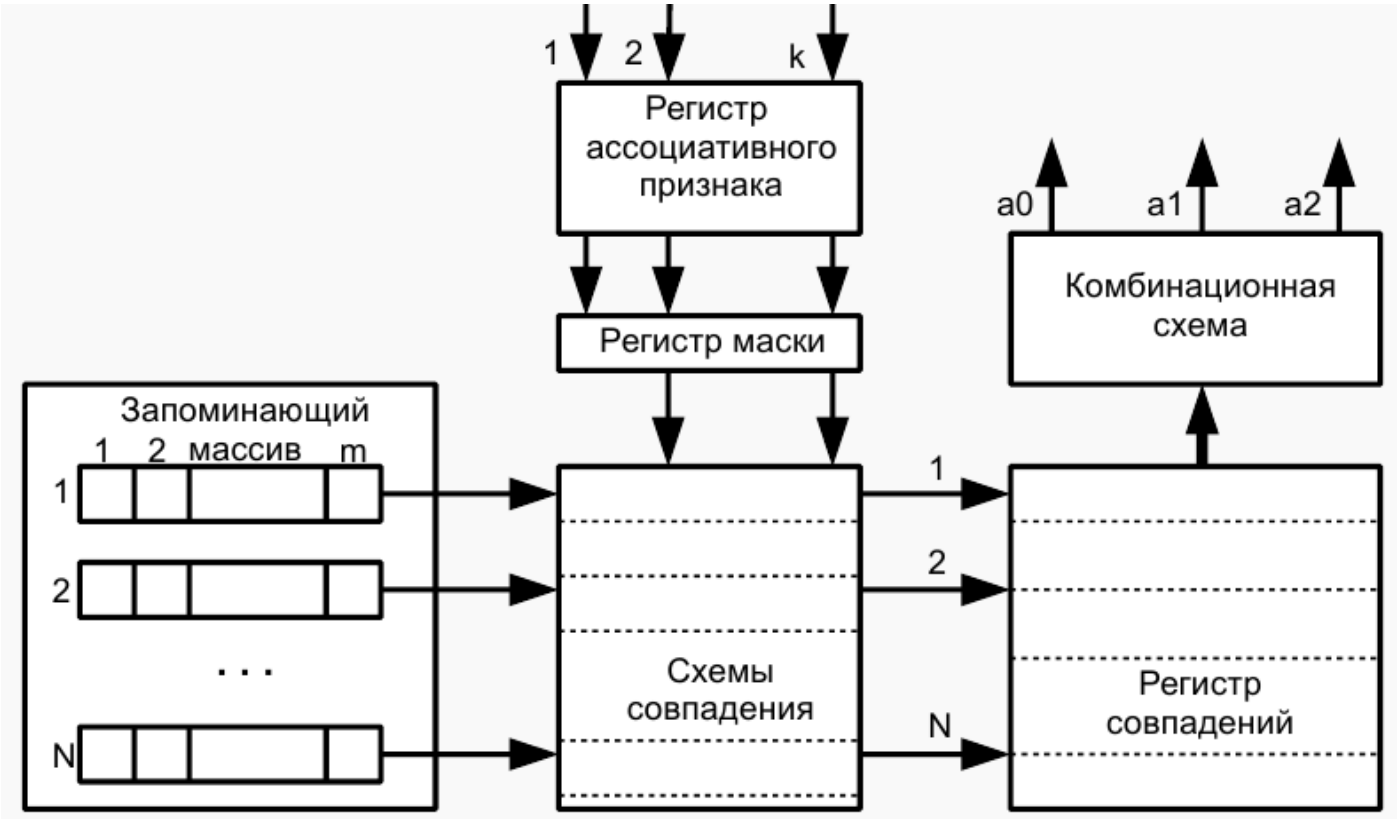
29. Адресуемая память, адресное пространство ЭВМ. Раздельные и смежные адресные пространства. Организация памяти. Ассоциативная память, кэш-память.

Адресное пространство — это набор адресов, который может быть использован процессом для обращения к памяти.

Ассоциативная память

Священная книга с. 110

Структура ассоциативной памяти



Кэш

ОЧЕНЬ МНОГО копипаста из Эндрю, стр. 81, стр. 295:

Процессоры всегда работали быстрее, чем память. Процессоры и память совершенствовались параллельно, поэтому это несоответствие сохранялось. Поскольку на микросхему можно помещать все больше и больше транзисторов, разработчики процессоров использовали эти преимущества для создания конвейеров и супер-скалярной архитектуры, что еще больше повышало скорость работы процессоров. Разработчики памяти обычно использовали новые технологии для увеличения емкости, а не скорости, что еще больше усугубляло проблему. На практике такое несоответствие в

скорости работы приводит к следующему: после того как процессор дает запрос памяти, должно пройти много циклов, прежде чем он получит слово, которое ему нужно. Чем медленнее работает память, тем дольше процессору приходится ждать, тем больше циклов должно пройти.

Есть два пути решения этой проблемы. Самый простой из них — начать считывать информацию из памяти, когда это необходимо, и при этом продолжать выполнение команд, но если какая-либо команда попытается использовать слово до того, как оно считалось из памяти, процессор должен приостанавливать работу. Чем медленнее работает память, тем чаще будет возникать такая проблема и тем больше будет проигрыш в работе. Например, если отсрочка составляет 10 циклов, весьма вероятно, что одна из 10 следующих команд попытается использовать слово, которое еще не считалось из памяти.

Другое решение проблемы — сконструировать машину, которая не приостанавливает работу, но следит, чтобы программы-компиляторы не использовали слова до того, как они считаются из памяти. Однако это не так просто осуществить на практике. Часто при выполнении команды загрузки машина не может выполнять другие действия, поэтому компилятор вынужден вставлять пустые команды, которые не производят никаких операций, но при этом занимают место в памяти. В действительности при таком подходе простаивает не аппаратное, а программное обеспечение, но снижение производительности при этом такое же.

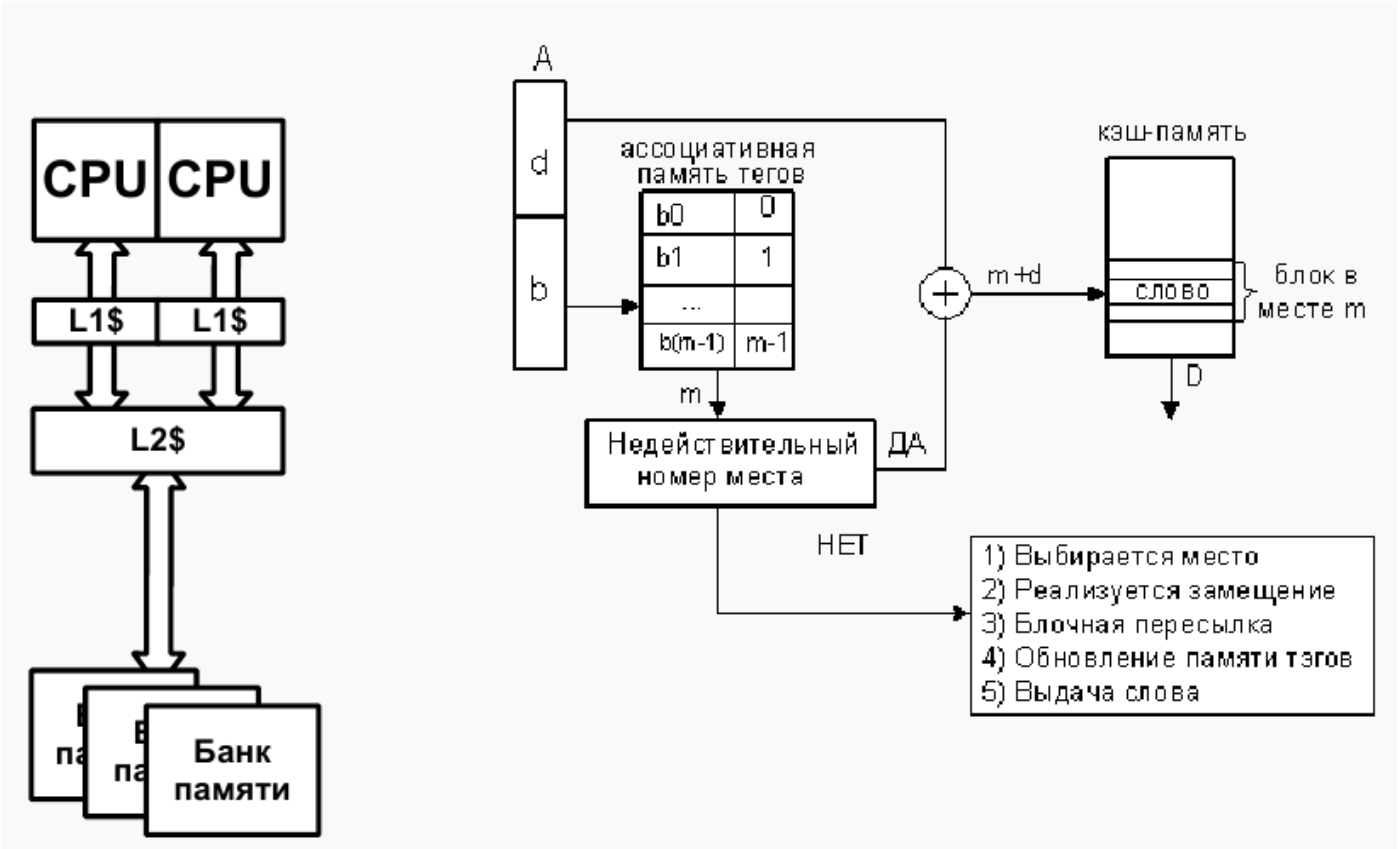
На самом деле эта проблема не технологическая, а экономическая. Инженеры знают, как построить память, которая будет работать так же быстро, как и процессор, но при этом ее приходится помещать прямо на микросхему процессора (поскольку информация через шину поступает очень медленно). Установка большой памяти на микросхему процессора делает его больше и, следовательно, дороже, и даже если бы стоимость не имела значения, все равно существуют ограничения в размерах процессора, который можно сконструировать. Таким образом, приходится выбирать между быстрой памятью небольшого размера и медленной памятью большого размера. Мы бы предпочли память большого размера с высокой скоростью работы по низкой цене. Маленькая память с высокой скоростью работы называется **кэш-памятью** (от французского слова cacher «прятать»). Основная идея кэш-памяти проста: в ней находятся слова, которые чаще всего используются. Если процессору нужно какое-нибудь слово, сначала он обращается к кэш-памяти. Только в том случае, если слова там нет, он обращается к основной памяти. Если значительная часть слов находится в кэш-памяти, среднее время доступа значительно сокращается. Таким образом, успех или неудача зависит от того, какая часть слов находится в кэш-памяти. Давно известно, что программы не обращаются к памяти наугад. Если программе нужен доступ к адресу A, то скорее всего после этого ей понадобится доступ к адресу, расположенному поблизости от A. Практически все команды обычной программы (за исключением команд перехода и вызова процедур) вызываются из последовательных участков памяти. Кроме того, большую часть времени программа тратит на циклы, когда ограниченный набор команд выполняется снова и снова. Точно так же при манипулировании матрицами программа скорее всего будет обращаться много раз к одной и той же матрице, прежде чем перейдет к чему-либо другому.

То, что при последовательных отсылках к памяти в течение некоторого промежутка времени используется только небольшой ее участок, называется принципом локальности. Этот принцип составляет основу всех систем кэш-памяти. Идея состоит в следующем: когда определенное слово вызывается из памяти, оно вместе с соседними словами переносится в кэш-память, что позволяет при очередном запросе быстро обращаться к следующим словам. Если слово считывается или записывается k раз, компьютеру понадобится сделать 1 обращение к медленной основной памяти и к-1 обращений к быстрой кэш-памяти. Чем больше к, тем выше общая производительность.

Одной из самых эффективных технологий одновременного увеличения пропускной способности и уменьшения времени ожидания является применение нескольких блоков кэш-памяти. Основная технология — введение отдельной кэш-памяти для команд и отдельной для данных (разделенной кэш-памяти). Такая кэш-память имеет несколько преимуществ. Во-первых, операции могут начинаться независимо в каждой кэш-памяти, что удваивает пропускную способность системы памяти. Именно по этой причине в микроархитектуре Mic-1 нам понадобились два отдельных порта памяти: особый порт для каждой кэш-памяти. Отметим, что каждая кэш-память имеет независимый доступ к основной памяти.

В настоящее время многие системы памяти гораздо сложнее этих. Между разделенной кэш-памятью и основной памятью часто помещается кэш-память второго уровня. Вообще говоря, может быть три и более уровней кэш-памяти, поскольку требуются более продвинутые системы. Прямо на микросхеме центрального процессора находится небольшая кэш-память для команд и небольшая кэш-память для данных, обычно от 16 до 64 Кбайт. Есть еще кэш-память второго уровня, которая расположена не на самой микросхеме процессора, а рядом с ним в том же блоке. Эта кэш-память обычно не является разделенной и содержит смесь данных и команд. Ее размер — от 512 Кбайт до 1 Мбайт. Кэш-память третьего уровня находится на той же плате, что и процессор, и обычно состоит из статического ОЗУ в несколько мегабайтов, которое функционирует гораздо быстрее, чем динамическое ОЗУ основной памяти. Обычно все содержимое кэш-памяти первого уровня находится в кэш-памяти второго уровня, а все содержимое кэш-памяти второго уровня находится в кэш-памяти третьего уровня.

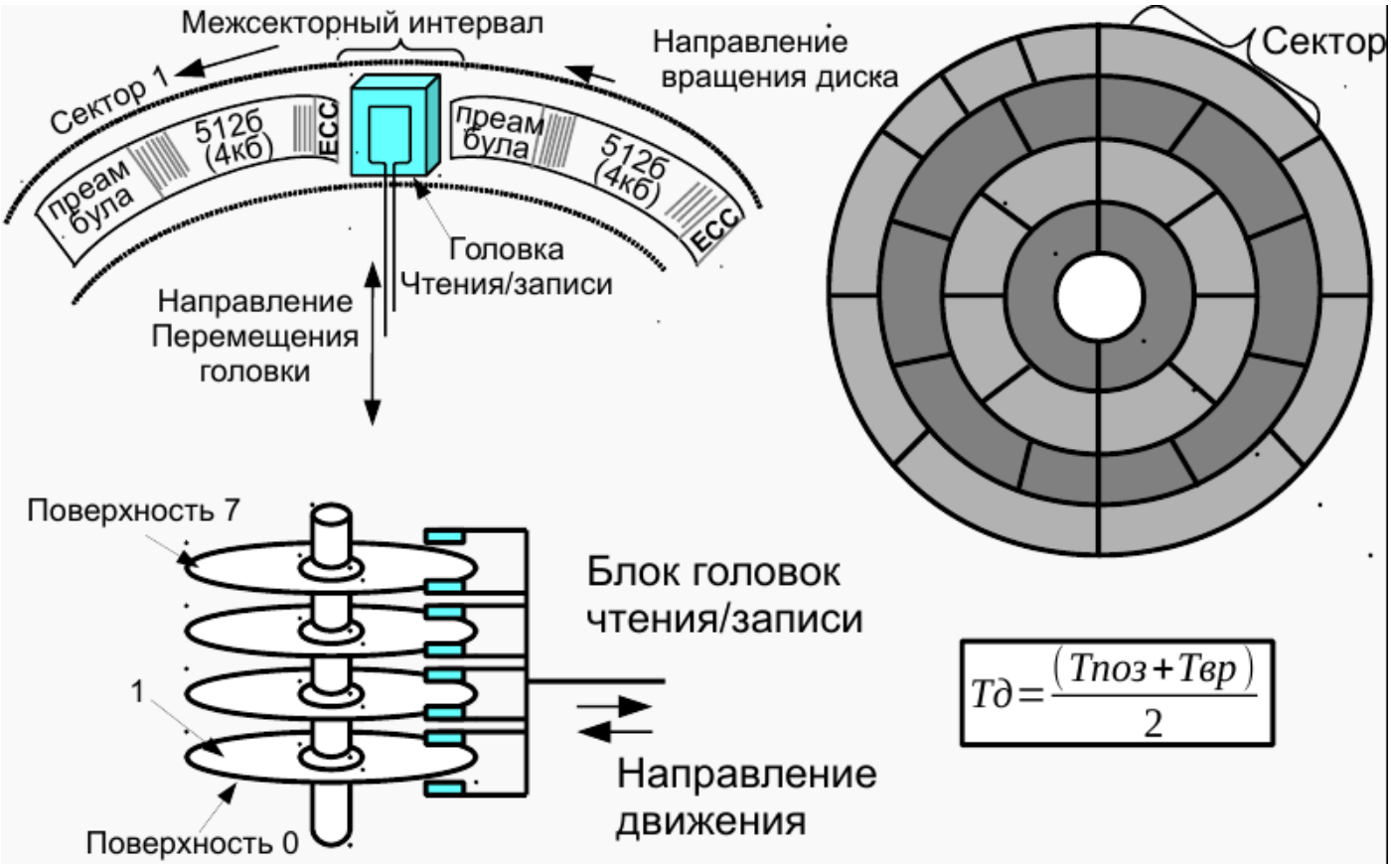
Структура кэш-памяти



30. Блочная память. Организация дисковой памяти и памяти на магнитных лентах.

Блочная структура (схема). Адресное пространство памяти разбивается на группы последовательных адресов, и каждая такая группа обеспечивается отдельным банком памяти (рис. 25). Для обращения к ОП используется 9-разрядный адрес, семь младших разрядов которого (А6-А0) поступают параллельно на все банки памяти и выбирают в каждом из них одну ячейку. Два старших разряда адреса (А8, А7) содержат номер банка. Выбор банка обеспечивается либо с помощью дешифратора номера банка памяти, либо путем мультиплексирования информации (на рис. 25 показаны оба варианта). В функциональном отношении такая ОП может рассматриваться как единое ЗУ, емкость которого равна суммарной емкости составляющих банков, а быстродействие – быстродействию отдельного банка.

Священная книга с. 154



31. Характеристики запоминающих устройств. Современные типы памяти и их характеристики. Влияние промахов кэш-памяти на производительность.

- Месторасположение – процессорные, внутренние, внешние
- Емкость – В метрических (Кило-) и двоичных (Киби-) множителях
- Единица пересылки – Слово, строка кэша, блок на диске
- Метод доступа – Произвольный (адресный), ориентированных на записи (прямой), последовательный, ассоциативный

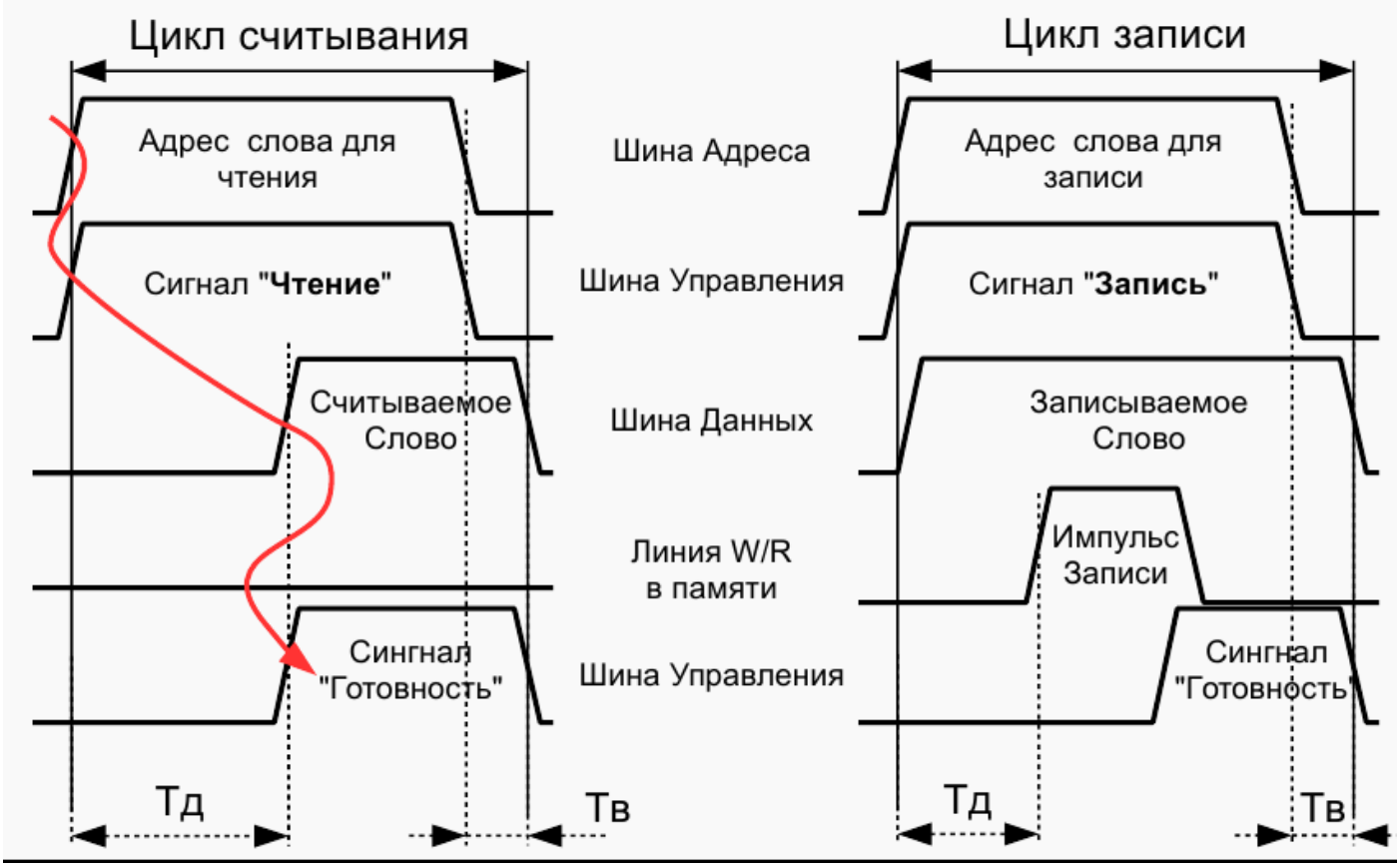
1. Быстродействие и временные соотношения
    a. Время доступа Тд
    b. Длительность цикла памяти (время обращения) Тц
    c. Время чтения и время записи
    d. Время восстановления Тв
    e. Скорость передачи информации
2. Физический тип и особенности
3. Стоимость

Запоминающие устройства имеют ряд показателей качества, характеризующих их информационные и временные свойства.

Информационная емкость памяти выражается в количестве битов, байтов или слов, состоящих из определенного числа байтов.

Время доступа – временной интервал, определяемый от момента, когда процессор выставил на адресной шине адрес требуемой ячейки и послал по шине управления приказ на чтение или запись данных, до момента осуществления связи адресуемой ячейки с шиной данных.

Время записи – интервал времени, необходимый для переписи содержимого шины данных в связанную с ней ячейку памяти.



Время восстановления необходимо для приведения памяти в исходное состояние после того, как процессор снял с шин адрес, сигнал ЧТЕНИЕ/ЗАПИСЬ и данные.

Цикл считывания и цикл записи определяются как время с момента выдачи процессором адреса требуемой ячейки памяти и сигнала на считывание или запись до того момента, когда заканчиваются все действия, связанные с выполняемой операцией, и память будет готова реализовать следующую операцию.

Стоимость 1 бита определяется отношением стоимости памяти к ее информационной емкости.

Возможность изменения информации характеризует тип памяти. Существует элементы памяти с легко изменяемыми состояниями (ОЗУ). Есть более дешевые элементы памяти, в которые единожды записывают 0 или 1 (ПЗУ, ППЗУ, СППЗУ (содержимое можно стереть, а затем вновь заполнить информацией)).

Различают 2 основных типа ЗУ: с произвольной выборкой и последовательной выборкой. В первых время доступа к заданному слову не зависит от месторасположения этого слова в памяти, а во вторых– зависит.

ППЗУ (аббревиатура от «**программируемое постоянное запоминающее устройство**»)

СППЗУ — комп. **стираемое программируемое постоянное запоминающее устройство**

(Запоминающее устройство с произвольной выборкой (сокращённо ЗУПВ; англ. Random Access Memory, RAM) — один из видов памяти компьютера, позволяющий единовременно получить доступ к любой ячейке (всегда за одно и то же время, вне зависимости от расположения) по её адресу на чтение или запись.

Запоминающее устройство с последовательной выборкой – например, магнитная летна)

В некоторых видах ЗУ происходит потеря записанной информации при отключении питающего напряжения. Такие устройства называют энергозависимыми ЗУ. В энергонезависимых ЗУ информация при отключении питания сохраняется.

Пирамида памяти

	Объем	Тд	*	Тип	Управл.
CPU	100-1000 б.	<1нс	1с	Регистр	компилятор
L1 Cache	32-128Кб	1-4нс	2с	Ассоц.	аппаратура
L2-L3 Cache	0.5-32Мб	8-20нс	19с	Ассоц.	аппаратура
Основная память	0.5Гб-4ТБ	60-200нс	50-300с	Адресная	программно
SSD	128Гб-1Тб/drive	25-250мкс	5д	Блочн.	программно
Жесткие диски	0.5Тб-4Тб/drive	5-20мс	4м	Блочн.	программно
Магнитные ленты	1-6Тб/к	1-240с	200л	Последов.	программно

Современные ЗУ

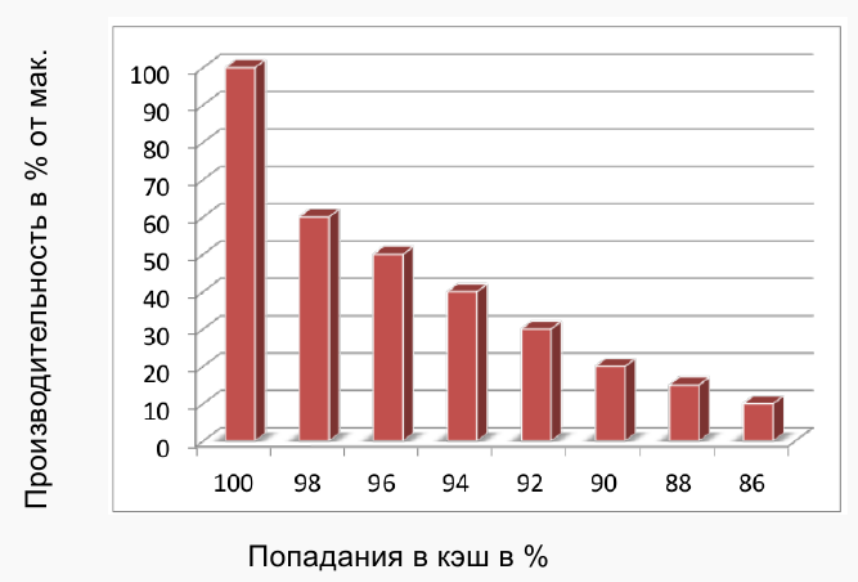
Копипаст из лекций from Batya:

- ● Burst mode — пакетный режим
- ● Double Data Rate — передача данных и по фронту и по спаду
- ● SPD — чип, содержащий идентификационную информацию
- ● Interleaving — расслоение памяти, повышает производительность
- ● DDR4-2133 8192MB PC4-17000 индекс производительности

Влияние промахов кэш-памяти

Cache Miss(промах кэша) случается, когда запрашиваемые данные отсутствуют в кэше и их нужно подгружать из основного источника.

Так, например, у любого современного процессора есть кэш, в котором он хранит недавно прочитанные блоки оперативной памяти и обращение к кэшированным данным происходит существенно быстрее, чем "перекачивание" этих же данных из оперативной памяти на кристалл процессора. Если всё-таки запрашиваются данные, которые в данный момент в кэше отстутствуют или они уже не актуальны, они вместе с другими находящимися к ним близко данными, поступают в кэш процессора.



32. Предназначение и организация виртуальной памяти.

Виртуа́льная па́мять — метод управления памятью компьютера, позволяющий выполнять программы, требующие больше оперативной памяти, чем имеется в компьютере, путём автоматического перемещения частей программы между основной памятью и вторичным хранилищем. Для выполняющейся программы данный метод полностью прозрачен и не требует дополнительных усилий со стороны программиста, однако реализация этого метода требует как аппаратной поддержки, так и поддержки со стороны операционной системы.

В системе с виртуальной памятью используемые программами адреса, называемые виртуальными адресами, транслируются в физические адреса в памяти компьютера. Трансляцию виртуальных адресов в физические выполняет аппаратное обеспечение, называемое блоком управления памятью. Для программы основная память выглядит как доступное и непрерывное адресное пространство либо как набор непрерывных сегментов, вне зависимости от наличия у компьютера соответствующего объёма оперативной памяти. Управление виртуальными адресными пространствами, соотнесение физической и виртуальной памяти, а также перемещение фрагментов памяти между основным и вторичным хранилищами выполняет операционная.

Применение виртуальной памяти позволяет:

- освободить программиста от необходимости вручную управлять загрузкой частей программы в память и согласовывать использование памяти с другими программами
- предоставлять программам больше памяти, чем физически установлено в системе
- в многозадачных системах изолировать выполняющиеся программы друг от друга путём назначения им непересекающихся адресных пространств

Страничная организация виртуальной памяти

В большинстве современных операционных систем виртуальная память организуется с помощью страничной адресации. Оперативная память делится на страницы: области памяти фиксированной длины, которые являются минимальной единицей выделяемой памяти. Исполняемый процессором пользовательский поток обращается к памяти с помощью адреса виртуальной памяти, который делится на номер страницы и смещение внутри страницы. Процессор преобразует номер виртуальной страницы в адрес соответствующей ей физической страницы при помощи буфера ассоциативной трансляции (TLB). Если ему не удалось это сделать, то требуется дозаполнение буфера путём обращения к таблице страниц, что может сделать либо сам процессор, либо операционная система. Если страница была выгружена из оперативной памяти, то операционная система подкачивает страницу с жёсткого диска в ходе обработки события. При запросе на выделение памяти операционная система

может «сбросить» на жёсткий диск страницы, к которым давно не было обращений. Критические данные (например, код запущенных и работающих программ, код и память ядра системы) обычно находятся в оперативной памяти (исключения существуют, однако они не касаются тех частей, которые отвечают за обработку аппаратных прерываний, работу с таблицей страниц и использование файла подкачки).

Сегментная организация виртуальной памяти

Механизм организации виртуальной памяти, при котором виртуальное пространство делится на части произвольного размера — сегменты. Этот механизм позволяет, к примеру, разбить данные процесса на логические блоки.[15] Для каждого сегмента, как и для страницы, могут быть назначены права доступа к нему пользователя и его процессов. При загрузке процесса часть сегментов помещается в оперативную память, а часть сегментов размещается в дисковой памяти. Во время загрузки система создает таблицу сегментов процесса, в которой для каждого сегмента указывается начальный физический адрес сегмента в оперативной памяти, размер сегмента, правила доступа, признак модификации, признак обращения к данному сегменту за последний интервал времени и некоторая другая информация. Если виртуальные адресные пространства нескольких процессов включают один и тот же сегмент, то в таблицах сегментов этих процессов делаются ссылки на один и тот же участок оперативной памяти, в который данный сегмент загружается в единственном экземпляре. Система с сегментной организацией функционирует аналогично системе со страничной организацией: время от времени происходят прерывания, связанные с отсутствием нужных сегментов в памяти, при необходимости освобождения памяти некоторые сегменты выгружаются, при каждом обращении к оперативной памяти выполняется преобразование виртуального адреса в физический. Кроме того, при обращении к памяти проверяется, разрешен ли доступ требуемого типа к данному сегменту.

Виртуальный адрес при сегментной организации памяти может быть представлен парой (g, s), где g — номер сегмента, а s — смещение в сегменте. Физический адрес получается путём сложения начального физического адреса сегмента, найденного в таблице сегментов по номеру g, и смещения s.

Недостатком данного метода распределения памяти является фрагментация на уровне сегментов и более медленное по сравнению со страничной организацией преобразование адреса.

33. Интерфейсы ввода-вывода. Контроллеры внешних устройств. Уровни стандартизации, сопряжения с системной шиной, циклы обмена. Регистры контроллера.

Интерфейс - это аппаратное и программное обеспечение (элементы соединения и вспомогательные схемы управления, их физические, электрические и логические параметры), предназначенное для сопряжения систем или частей системы (программ или устройств). Под сопряжением подразумеваются следующие функции:

- выдача и прием информации;
- управление передачей данных;
- согласование источника и приемника информации.

Контроллер - специализированный процессор, предназначенный для управления внешними устройствами: накопителями, мониторами, принтерами и т.п. Наличие контроллера освобождает центральный процессор от выполнения этих функций.

Интерфейсы ввода-вывода

Системный интерфейс МПК БИС КР580 (Microbus)

В этом интерфейсе 36 линий, разделенных на 3 шины(16-разрядная шина адреса, 8-разрядная шина данных, 12-разрядная шина управления). Шина адреса – для пересылки адреса ячейки и для передачи по 8 младшим линиям адреса ВУ. Шина данных используется в 3 режимах:

1. Для обмена байтом информации между процессором и памятью
2. Для обмена байтом информации между процессором и контроллером ВУ при выполнении команд ввода-вывода
3. Для передачи 8-разрядного слова состояния МП во внешние схемы управления работой микроЭВМ

Шина управления – для передачи 6 входных и 6 выходных сигналов. Они обеспечивают управление работой процессора и порядок использования шин адреса и данных.

Управляющие сигналы позволяют организовать 2 режима обмена информацией с контроллерами ВУ:

1. Программно-управляемый (асинхронный и по прерыванию)
2. Прямой доступ в память

Системный интерфейс «Электроника-60»(Q-bus)

Состоит из 34 линий, разделенных на 2 шины(16-разрядная мультиплексируемая шина «адрес/данные», 18-разрядная шина управления).

Шина «адрес/данные» состоит из 16 двунаправленных линий для поочередной передачи адресов и данных. Часть адресного пространства, начиная с адреса (160000)8 до адреса (177776)8 (всего 4096 адресов) выделена для адресов регистров контроллеров ВУ. Поэтому 16-разрядная шина обеспечивает прямую адресацию памяти емкостью 56 Кбайт и 4096 (4К) регистров контроллеров ВУ. Обмен данными может производиться 16-разрядными словами и байтами. Для передачи данных используются 8 линий шины.

Сигналы, передаваемые по линиям управления, обеспечивают управление:

- Передачей информации по шине «адес/данные»
- Режимами обмена с ВУ с учетом их приоритетов
- Работой микроЭВМ

Управление передачей информации – циклы обмена (ввод и вывод) между двумя устройствами. Одно из устройств управляет обменом(активное устройство, например, процессор или контроллер), второе устройство – пассивное(например, память микроЭВМ).

Сигналы, управляющие работой микроЭВМ обеспечивают начальную установку всех устройств (СБРОС), регенерацию динамической памяти (РГН), перевод процессора в останов и режим связя с пультовым терминалом (ОСТ), извещают процессор о состоянии источников питания (ПОСТН, ПИТН).

Уровни стандартизации

- логический (все алгоритмы неким образом понимаются системой)
- физический (напряжение, ток)
- конструктивный (особенности конструкции)

Сопряжение с системной шиной

(Рис. 8.1, а) При использовании для обмена с ВУ команд ввода-вывода адрес (номер) ВУ передается по шине адреса. По этой же шине передаются и адреса ячеек памяти. Информация на шине адреса имеет смысл адреса ВУ только при наличии специальных управляющих сигналов (например, «Ввод из ВУ», «Вывод в ВУ», которые инициируются соответствующими командами ввода-вывода).

(Рис. 8.1, б) При реализации обмена с ВУ по аналогии с обращениями к памяти нет необходимости в спец. сигналах, указывающих, что на шине адреса находится адрес ВУ. Для этих адресов отведена

часть адресного пространства, в контроллерах организована селекция адресов ВУ. Но остается необходимость передавать в ВУ приказ на ввод/вывод информации. Для этого есть линии управляющей шины «Чтение» и «Запись». Они обеспечивают обмен информацией микропроцессора с модулями памяти.

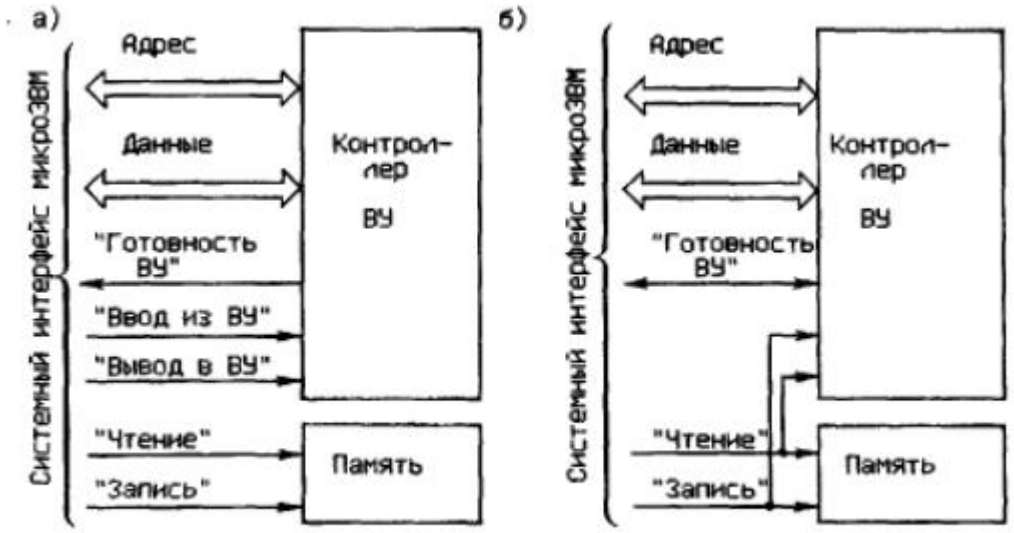


Рис. 8.1. Простейшее сопряжение контроллера ВУ с системным интерфейсом

Циклы обмена

Операция вывод: микропроцессор выставляет на линиях адресной шины адрес (номер) ВУ, на линиях шины данных – значения разрядов выводимого слова данных и единичным сигналом по линии «Вывод в ВУ» указывает тип операции. Адресуемый контроллер ВУ принимает данные, пересылает их в ВУ и единичным сигналом по линии «Готовность ВУ» сообщает процессору, что данные приняты в ВУ и можно снять информацию с шин адреса и данных, а также сигнал «Вывод в ВУ».

Операция ввод: микропроцессор выставляет на линиях адресной шины адрес (номер) ВУ, и единичным сигналом на линии «Ввод из ВУ» указывает тип операции. По сигналу «Ввод из ВУ» контроллер адресуемого ВУ считывает слово данных из ВУ, выставляет на линиях шины данных значения разрядов считанного слова и единичным сигналом на линии «Готовность ВУ» сообщает об этом процессору. Приняв данные, процессор снимает сигналы с шины адреса и линии «Ввод из ВУ».

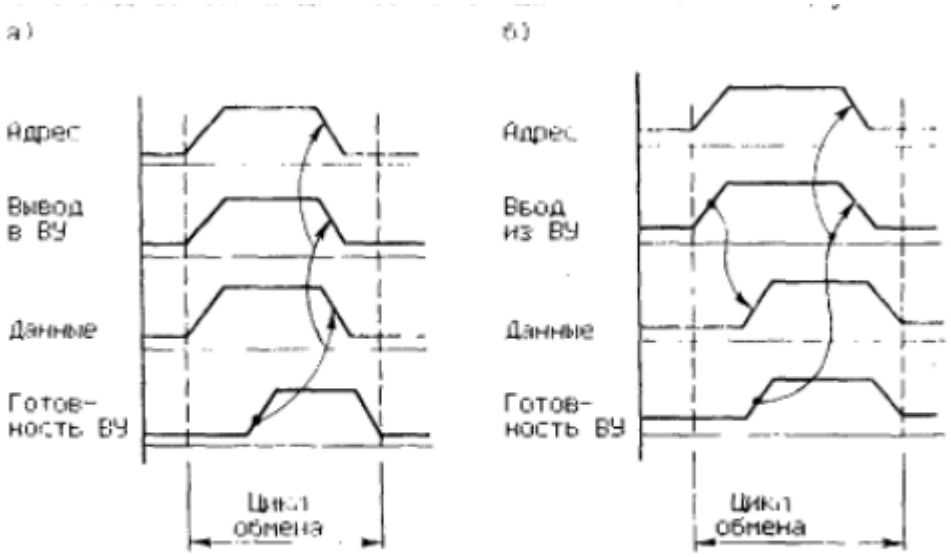


Рис. 8.2. Временные диаграммы операций для простейшего набора управляющих сигналов:
а - ВЫВОД; *б* ВВОД

Контроллеры внешних устройств

Подключение любого внешнего устройства к микроЭВМ осуществляется через контроллер ВУ. Способы организации контроллеров определяются двумя факторами:

1. Форматами данных и режимами работы конкретных ВУ (существование самых разнообразных контроллеров от простейших до очень сложных)
2. Типом системного интерфейса микроЭВМ (определяет способ организации электронных схем контроллеров ВУ, обеспечивающих связь с шинами интерфейса, в первую очередь - схем распознавания адресов ВУ)

Для разных типов микроЭВМ разработаны контроллеры, обеспечивающие:

- Связь с ВУ по стандартному параллельному (ИРПР) каналу передачи данных;
- Связь с ВУ по стандартному последовательному (ИРПС) каналу передачи данных;
- Преобразование информации из аналоговой в цифровую с заданной точностью;
- Преобразование информации из дискретной формы представления в аналоговую в заданных диапазонах изменения аналоговых величин.

Существуют также программируемые контроллеры, режимы работы которых устанавливаются специальными командами микроЭВМ или определяются программами обмена с ВУ. Такие контроллеры необходимо настраивать на конкретный режим обмена (синхронный, асинхронный, с сигналами прерывания или без них). Настройка контроллеров производится программным путем с помощью спец. команд.

Уровни сопряжения ВУ с ЭВМ

Контроллеры ВУ сопрягаются с процессором и памятью через системный интерфейс микроЭВМ

Контроллеры посредством шин связи с ВУ сопрягаются с соответствующими внешними устройствами микроЭВМ

Регистры контроллера

Рассмотрим типичные структуры контроллеров ВУ:

В основе контроллера несколько регистров (для временного хранения передаваемой информации). Каждый регистр имеет свой адрес (такие регистры называют портами ввода-вывода).

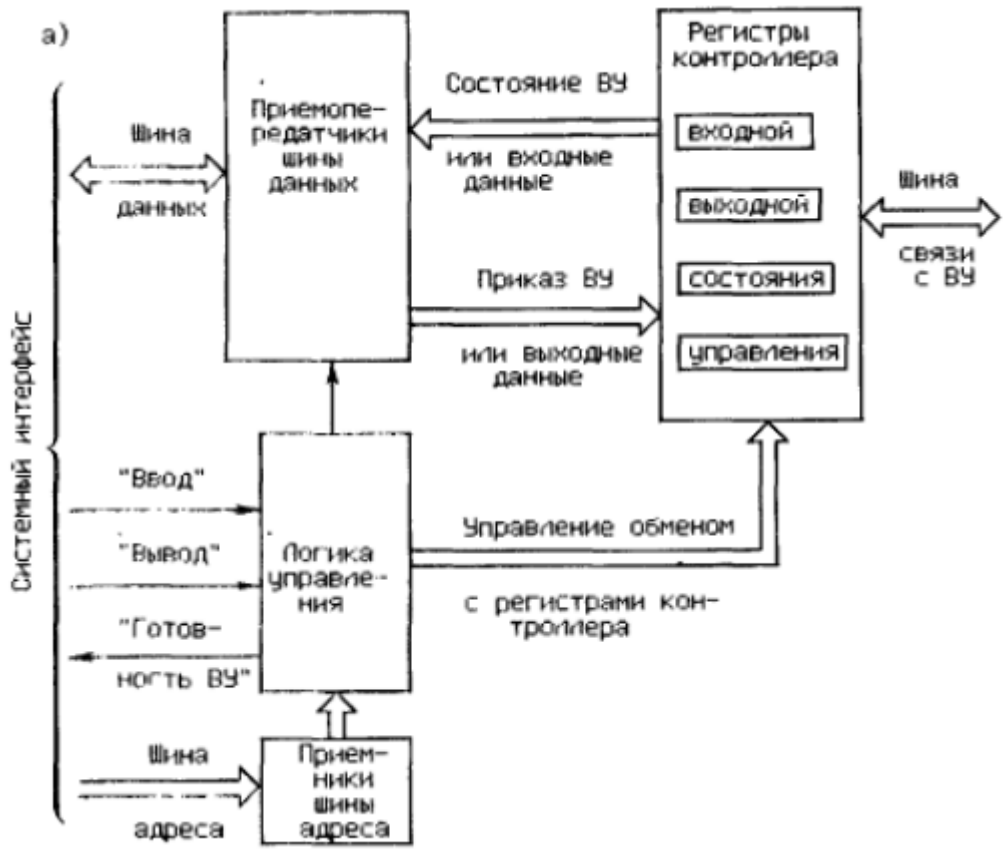
Регистр входных данных работает в режиме чтения, регистр выходных данных в режиме записи.

Регистр состояния в режиме чтения и содержит информацию о состоянии ВУ (включено/выключено, готово/не готово к обмену данными).

Регистр управления в режиме записи, служит для приема из микроЭВМ приказов для ВУ.

Логика управления выполняет селекцию адресов регистров контроллера, прием, обработку и формирование управляющего сигнала системного интерфейса, обеспечивая обмен информацией между регистрами контроллера и шиной данных системного интерфейса.

Приемопередатчики шин адреса и данных – для физического подключения электронных схем контроллера к шинам системного интерфейса.



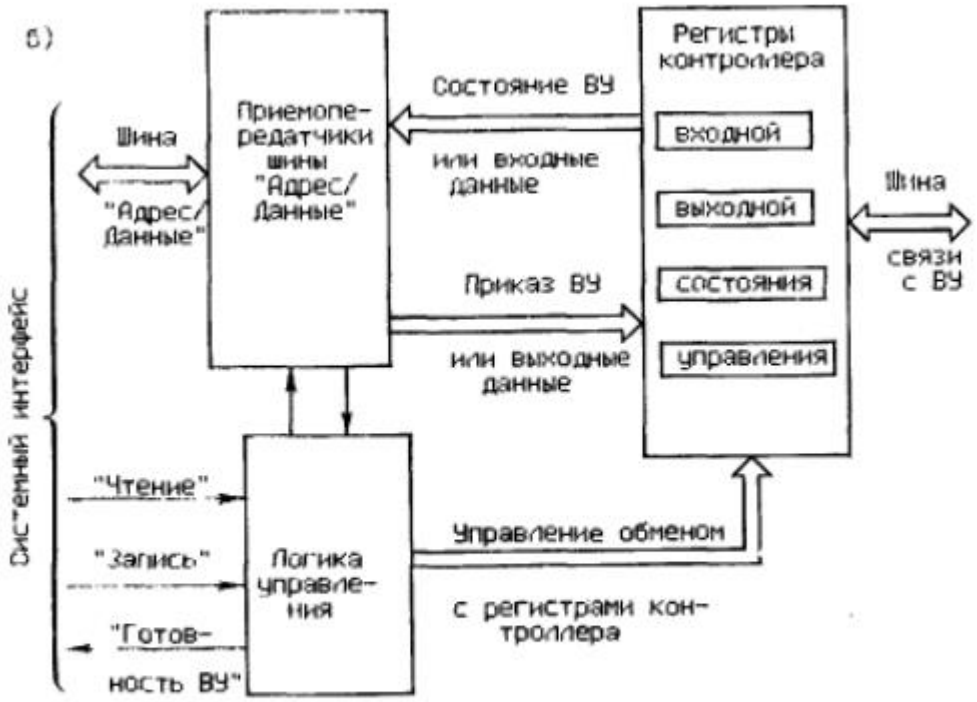


Рис 8.4. Блок-схема типичного контроллера ВУ для системного интерфейса: *а* с раздельными шинами адреса и данных; *б* с мультиплексированной шиной «Адрес/Данные»

Различия в структурах этих контроллеров в логике управления (по-разному организованы прием и селекция адресов) и способе подключения к шинам системного интерфейса.

34. Параллельная передача данных. Контроллеры параллельной передачи и приема.

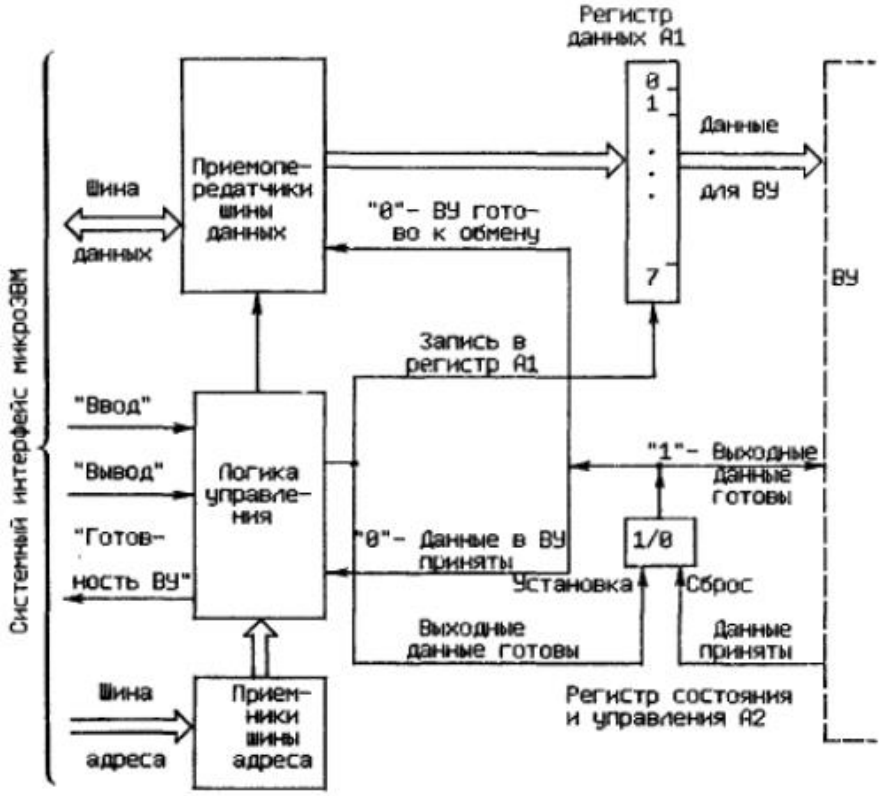


Рис. 8.6. Простой параллельный интерфейс (контроллер) вывода

Параллельная передача данных в ВУ под управлением программы асинхронного обмена:

1. Процессор микроЭВМ проверяет готовность ВУ к приему данных
2. Если ВУ готово к приему данных (логический 0 в регистре А2), то данные передаются из шины данных системного интерфейса в регистр данных А1 контроллера и далее в ВУ. Иначе повторяется пункт 1.

В шине связи с ВУ используются 2 управляющих сигнала. Для формирования управляющего сигнала «Выходные данные готовы» и приема из ВУ управ. сигнала «Данные приняты» в контроллере используется одноразрядный адресуемый регистр состояния и управления А2. Одновременно с записью очередного байта данных из шины данных сист. интерфейса в адресуемый регистр данных контроллера А1 в регистр состояния и управления записывается логическая единица (формируется управляющий сигнал «Выходные данные готовы»).

ВУ, приняв байт данных, управ. сигналом «Данные приняты» обнуляет регистр состояния. При этом формируется:

1. Управ. сигнал сист. интерфейса «Готовность ВУ»
2. Признак готовности ВУ к обмену, передаваемый в процессор по одной из линий шины данных

В логике управления – селекция адресов регистров контроллера, прием и формирование управ. сигналов, формирование сигнала «Готовность ВУ».

Для сопряжения регистров контроллера с шинами адреса и данных сист. интерфейса используются приемники шины адреса и приемопередатчики шины данных.

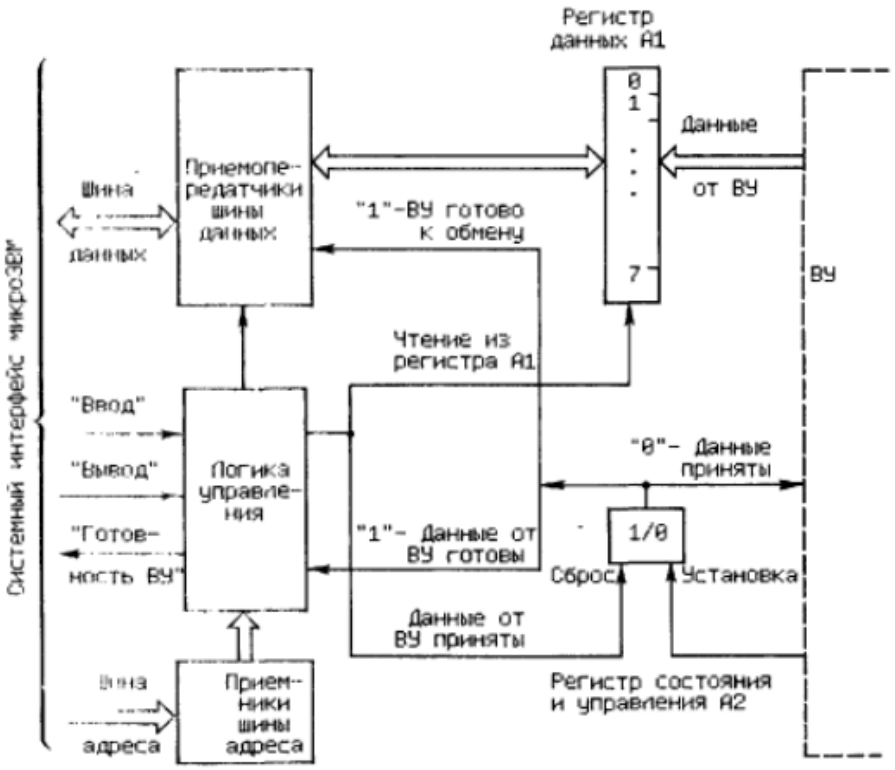


Рис. 8.7. Простой параллельный интерфейс (контроллер) ввода

Алгоритм асинхронного ввода:

1. Процессор проверяет наличие данных в регистре данных контроллера А1
2. Если данные готовы (логическая 1 в регистре А2), то они передаются из регистра данных А1 в шину данных системного интерфейса и далее в регистр процессора или ячейку памяти микроЭВМ. Иначе повторяется пункт 1.

Для формирования управляющего сигнала «Данные приняты» и приема из ВУ управ. сигнала «Данные от ВУ готовы» в контроллере используется одноразрядный адресуемый регистр состояния и управления А2.

ВУ записывает в регистр данных контроллера А1 очередной байт данных и управ. сигналом «Данные от ВУ готовы» устанавливает в единицу регистр состояния и управления А2. При этом формируется:

1. Управ. сигнал сист. интерфейса «Готовность ВУ»
2. Признак готовности ВУ к обмену, передаваемый в процессор по одной из линий шины данных

Так контроллер извещает процессор о готовности данных в регистре А1. Процессор читает байт данных из регистра данных контроллера и обнуляет регистр состояния и управления А2. При этом формируется управляющий сигнал «Данные приняты».

В логике управления – селекция адресов регистров контроллера, прием и формирование управ. сигналов, формирование сигнала «Готовность ВУ».

Для сопряжения регистров контроллера с шинами адреса и данных сист. интерфейса используются приемники шины адреса и приемопередатчики шины данных.

35. Синхронные последовательные интерфейсы. Контроллеры последовательной передачи и приема.

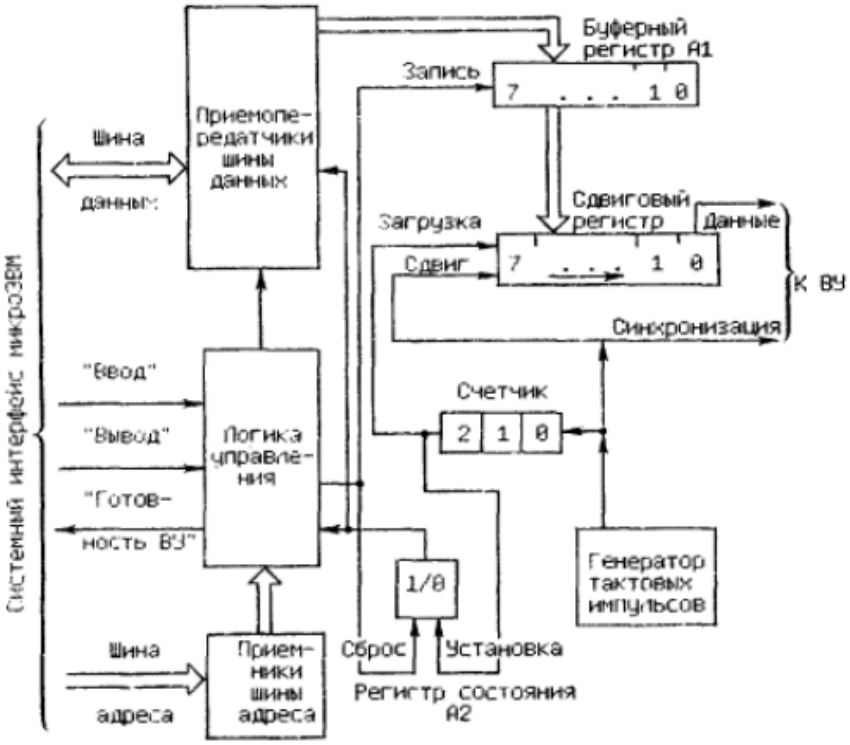


Рис. 8.8. Контроллер последовательной синхронной передачи

8-ми разрядный буферный регистр контроллера А1 - для временного хранения байта данных до его загрузки в сдвиговый регистр. Запись байта данных в буферный регистр происходит при наличии 1 в регистре состояния А2. Содержимое этого регистра передается в процессор по одной из линий шины данных и используется для формирования управ. сигнала «Готовность ВУ». При записи очередного байта в регистр А1 обнуляется регистр А2.

В сдвиговом регистре происходит преобразование данных из параллельного формата в последовательный и передача их в линию связи. По очередному тактовому импульсу содержимое сдвигового регистра сдвигается на 1 разряд вправо и в линию связи «Данные» выдается значение очередного разряда. Одновременно со сдвигом по линии «Синхронизация» передается тактовый импульс.

Количество переданных в линию тактовых сигналов (переданных бит) 01подсчитывается счетчиком тактовых импульсов. Как только его содержимое равно 7(передано 8 бит информации) формируется управляющий сигнал «Загрузка» и происходит запись в сдвиговый регистр очередного байта из буферного. Устанавливается в 1 регистр состояния. Следующим тактовым импульсом счетчик будет сброшен в 0 и начнется очередной цикл выдачи 8 бит из сдвигового регистра в линию связи.

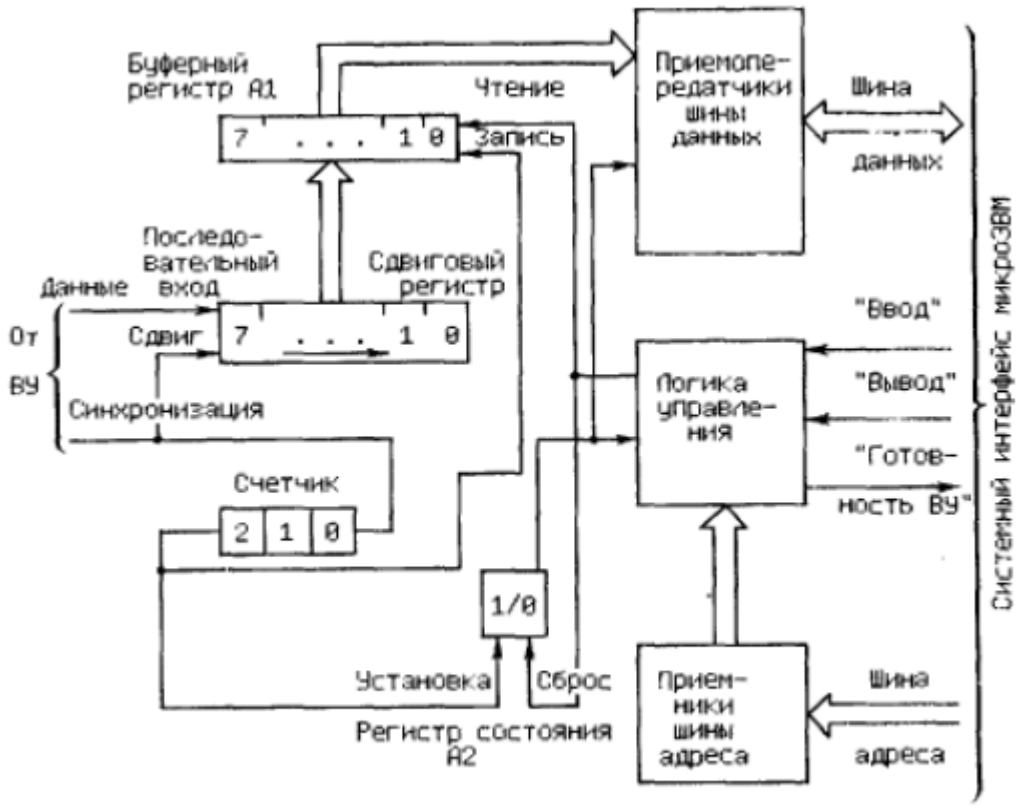


Рис. 8.9. Контроллер последовательного синхронного приема

Буферный регистр контроллера А1 - для временного хранения байта , поступившего из сдвигового регистра. Чтение байта данных из буферного регистра происходит при наличии 1 в регистре состояния А2.

Данные, поступающие из линии связи в последовательном коде преобразуются в параллельный с помощью сдвигового регистра. Линия «Данные» подключается в контроллере к последовательному входу сдвигового регистра, а линия «Синхронизация» - на управ. вход «Сдвиг» и на вход счетчика тактовых импульсов. По тактовому импульсу по линии «Синхронизация» производится сдвиг содержимого сдвигового регистра на 1 разряд влево и запись очередного бита данных из линии «Данные» в младший разряд этого регистра. Одновременно увеличивается на 1 счетчик тактовых импульсов. Как только он становится равным 7 формируется управ. сигнал «Запись» и происходит запись в буферный регистр байта из сдвигового. Устанавливается в 1 регистр состояния.

При передаче байта данных из буферного регистра в шину данных регистр состояния обнуляется (т.е. в сдвиговый регистр принимается очередной байт информации).

36. Асинхронный обмен. Принципы деления частоты, формат кадра.

Асинхронный обмен

При реализации *асинхронного обмена* интервал между командами передачи данных задается самим внешним устройством. Контроллеры этих устройств снабжаются регистром состояния, который информирует ЭВМ о готовности устройства к обмену информацией.

Обмен происходит по готовности ВУ, из этого вытекают следующие:

Преимущества:

1. Не нужно знать время выполнения операции на ВУ;
2. ВУ всегда успеет выполнить обработку данных перед началом следующей операции обмена.

Недостаток:

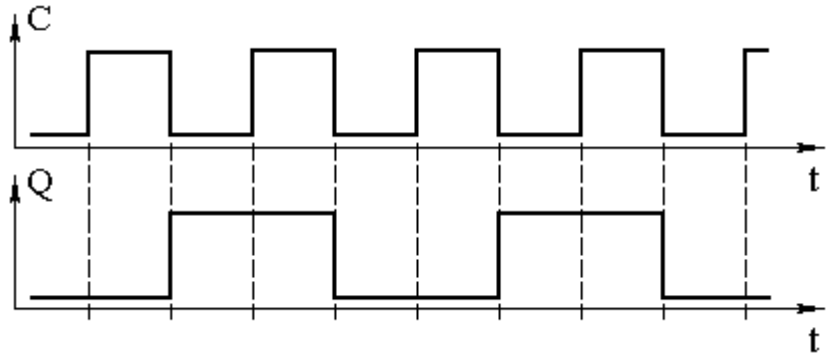
1. ЭВМ не выполняет никаких полезных действий во время ожидания момента готовности ВУ.

Принципы деления частоты

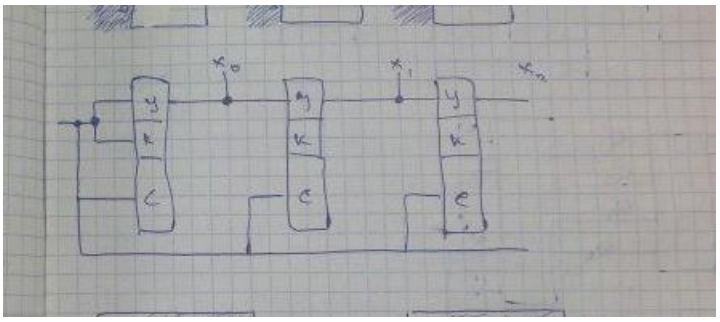
При асинхронной передаче данных, со временем может происходить рассинхронизация генераторов тактов передатчика и приёмника, в результате чего данные могут быть переданы с искажениями, или не быть переданы вовсе.

Одним из способов решения этой проблемы является деление тактовой частоты генераторов.

Принцип его заключается в том, что исходная тактовая частота делится с некоторым коэффициентом, чаще всего кратным степени двойки. Таким образом увеличивается область совпадения фаз и время передачи данных, что увеличивает точность передачи.



Триггер деления частоты (Обязательно знать)

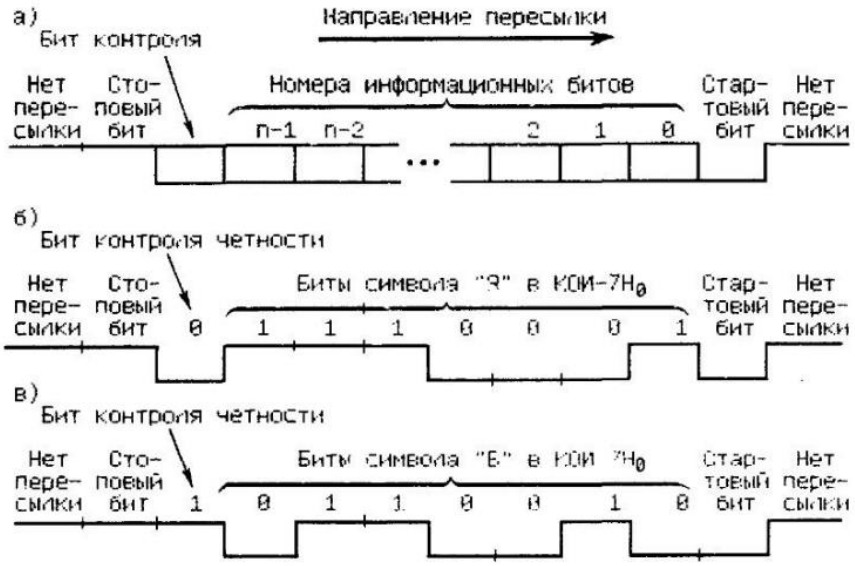


Тут 3 триггера, каждый делит частоту на 2. Соответственно, если замкнуть выходы на этом рисунке, частота будет поделена на 8

Очень часто необходимо использовать триггер для деления частоты входной последовательности импульсов на два, т. е. производить переключение триггера в новое состояние каждым входным импульсом (фронтом или спадом). Такой триггер называют счетным, или Т-триггером

Формат кадра

Формат кадра – количество бит в посылкепри передачи байта полезной информации



37. Контроллер передачи асинхронного последовательные интерфейса.

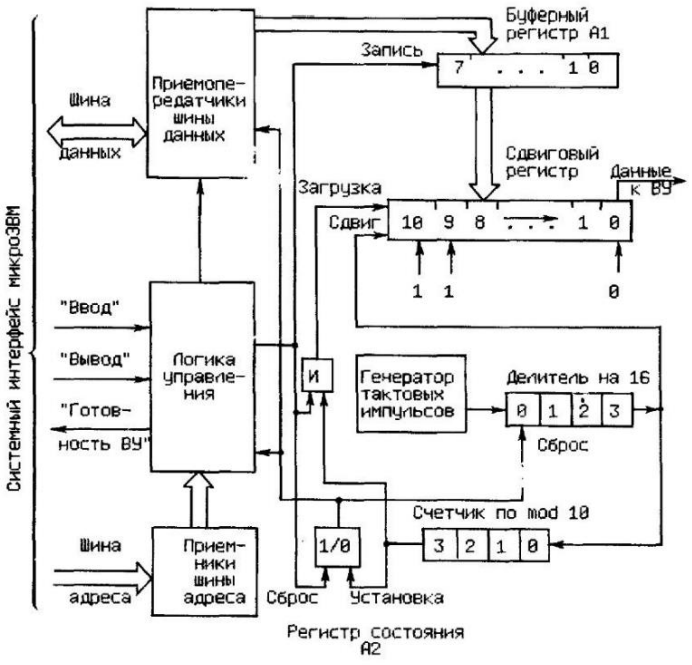


Рис. 8.10. Контроллер последовательной асинхронной передачи

На приемной и передающей стороне находятся настроенные на одну частоту, но физически разные генераторы, следовательно, отсутствует общая синхронизация.

На рис. 8.10 представлен контроллер для передачи данных с двумя стоповыми битами

Процесс передачи

1. После передачи предыдущих байтов данных в Регистр Состояния А2 записывается 1, что информирует процессор о готовности контроллера к приему следующего байта данных и передаче его по линии связи в ВУ. Он же запрещает формирование импульсов со схемы выработки импульсов сдвига – делителя частоты тактового генератора на 16 (счетчик по mod 16). Счетчик импульсов сдвига (счетчик по mod 10) находится в нулевом состоянии, и его единичный выходной сигнал поступает на вентиль И, подготавливая цепь выработки сигнала загрузки сдвигового регистра.

2. Процессор, выполняя команду «Вывод», выставляет передаваемый байт на шине данных и формирует управляющий сигнал системного интерфейса «Вывод».
3. По сигналу «Вывод» в контроллере происходит запись передаваемого байта в буферный регистр А1, сброс регистра состояния А2 и формирование на вентиле И сигнала «Загрузка».
4. Передаваемый бит переписывается в разряды 1..8 сдвигового регистра, в 0 разряд записывается нуль – стартовый бит, а в разряды 9 и 10 единицы – стоповые биты.
5. Снимается сигнал «Сброс» с делителя частоты, он начинает накапливать импульсы генератора тактовой частоты и в момент приема шестнадцатого тактового импульса срабатывает импульс сдвига (так реализовано деление частоты).
6. На шине «Данные» поддерживается 0 (значение стартового бита) до тех пор, пока не будет выработан первый импульс сдвига (время передачи 1 бита). Импульс сдвига изменит состояние счетчика импульсов сдвига и перепишет в нулевой разряд сдвигового регистра первый информационный бит передаваемого байта данных. Значение этого бита будет поддерживаться на линии «Данные» до следующего импульса сдвига.
7. Аналогично передаются остальные информационные биты, первый стоповый бит, и, наконец второй стоповый бит, при передаче которого счетчик импульсов сдвига снова установится в нулевое состояние. Это приведет к записи 1 в регистр состояния А2. Единичный сигнал с выхода регистра А2 запретит формирование импульсов сдвига, и информирует о готовности к приему нового байта данных.
8. После завершения передачи очередного кадра (стартового бита, информационного бита и 2х-стоповых битов), на линии передачи данных поддерживается значение второго стопового бита – единицы

38. Контроллер приема асинхронного последовательные интерфейса.

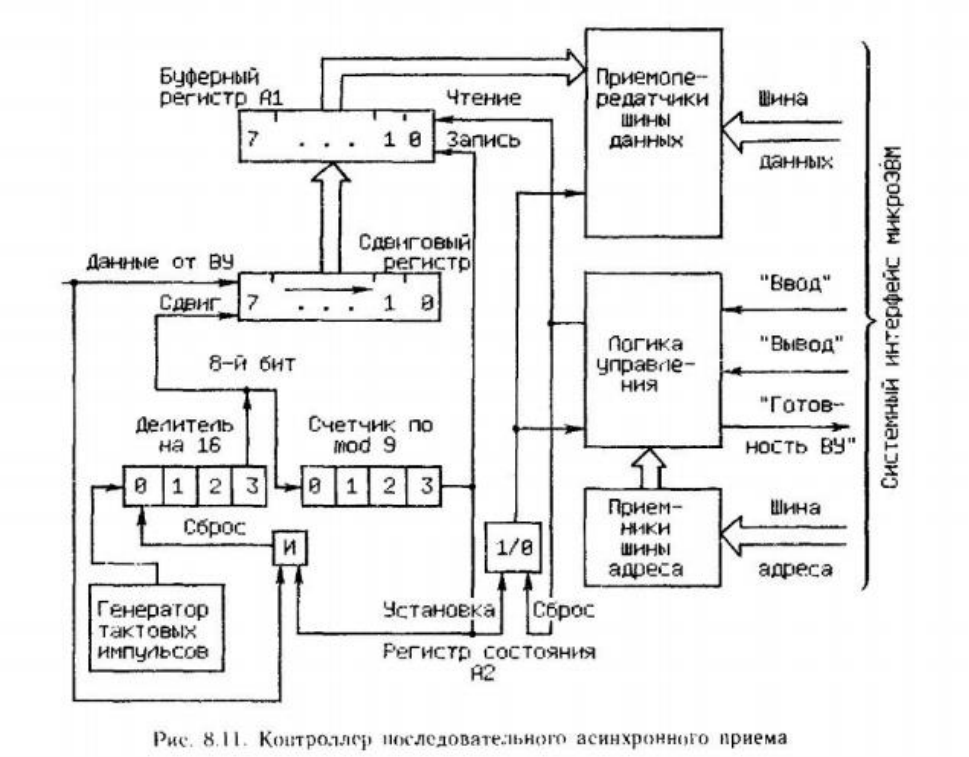


Рис. 8.11. Контроллер последовательного асинхронного приема

Процесс передачи

1. На линии «Данные» находится единица, что запрещает работу делителя частоты генератора тактовых импульсов.

2. При обнаружении нулевого сигнала на линии «Данные» (смена стопового бита на стартовый), снимается сигнал «Сброс» с делителя частоты, он начинает накапливать импульсы генератора тактовой частоты.
3. Когда на счетчике накопится значение 8 (половина времени передачи бита), он выдаст сигнал, поступающий на входы сдвигового регистра и счетчика импульсов сдвига. (Таким образом уменьшается вероятность искажения данных.)
4. Последующие сдвиги происходят после прохождения 16-ти тактовых импульсов.
5. При приеме в сдвиговый регистр 9-го бита кадра (8-го инф. Бита), из него выдвинется стартовый бит, и, следовательно в сдвиговом регистра будет размещен информационный байт. В этот момент счетчик импульсов сдвига придет в нулевое состояние и на его выходе будет выработан единичный сигнал, по которому:
    a. Содержимое сдвигового регистра будет переписано в БР
    b. В РС А2 запишется 1 и он будет информировать процессор об окончании приема очередного байта
    c. Вентиль И подготовится к вы работке сигнала «Сброс»(он сформируется после прихода первого стопового бита).
    d. Получив сигнал готовности (1 в РС А2), процессор выполнит команду «Ввод». При этом вырабатывается сигнал системного интерфейса «Ввод», по которому производится пересылка принятого байта данных из БР в процессор (сигнал «Чтение») и сброс РС А2

39. Организация прерываний в ЭВМ. Вектора прерываний, контроллеры прерываний.

Обмен с прерыванием программы отличается от асинхронного программно-управляемого обмена тем, что переход к выполнению команд, физически реализующих обмен данными, осуществляется с помощью спец. аппаратных средств.

Команды обмена данными в этом случае выделяют в отдельный программный модуль – программу обработки прерывания (ПОП).

Аппаратные средства обработки прерывания в процессоре приостанавливают выполнение основной программы и передают управление ПОП.

При этом действия, выполняемые процессором аналогичны действиями при обращении к подпрограмме, различие лишь в том, что переход к ПОП инициируется управляющим сигналом от ВУ, называющимся «Требование прерывания» или «Запрос прерывания».

Это позволяет организовать обмен в произвольные моменты времени и исчезает необходимость в организации программных циклов ожидания готовности ВУ.

ПОП может использовать некоторые регистры процессора в ходе своей работы, но поскольку ПОП не должна оказывать влияние на работоспособность основной программы, появляется необходимость сохранения содержимого этих регистров.

Обычно задачи сохранения содержимого СК и РС возлагается на аппаратные средства. Содержимое других же регистров сохраняется непосредственно в ПОП. Однако, чем большее количество регистров необходимо сохранить, тем большее время реакции ЭВМ на сигнал прерывания, и наоборот, поэтому для повышения производительности предпочтительно реализовывать сохранение информации о прерванной программе аппаратными средствами.

Формирование сигналов прерываний происходит в контроллерах ВУ.

В простейших случаях в качестве сигнала прерывания может использоваться сигнал «Готовность ВУ», однако в таком случае невозможно управление (т.е. разрешение или запрет) отдельными ВУ, следовательно значительно усложняется организация обмена данными с несколькими ВУ.

Для решения этой проблемы РС контроллера ВУ дополняют еще одним разрядом «Разрешение прерывание», запись в который производится программным путем по одной из линий шин данных системного интерфейса. Сигнал «Требование прерывания» формируется с помощью схемы совпадения только при наличии единиц в разрядах «Готовность ВУ» и «Разрешение прерывания» регистра состояния контроллера.

Аналогично решается проблема управления прерывания в микроЭВМ в целом – в РС добавляется бит «Разрешение прерывания», который устанавливается программным путем.

В микроЭВМ обычно используется одноуровневая система прерываний, т.е. сигналы «Требование прерывания» от всех ВУ поступают на один вход процессора, поэтому возникает проблема идентификации ВУ, запросившего прерывание и реализация очередности обслуживания, при нескольких таких сигналах.

Существует 2 основных способа идентификации ВУ:

1. Программный опрос регистров состояния контроллеров всех ВУ.
2. Использование векторов прерывания

Программный опрос готовности

В этом случае существует одна ПОП для всех ВУ.

Структура

1. В конце цикла исполнения проверяется наличие требования прерывания от ВУ. Если оно есть и прерывание в процессоре разрешено, то управление передается ПОП.
2. Сохраняется содержимое регистров, используемых в ПОП.
3. Последовательно опрашиваются флаги всех ВУ, пока не будет обнаружено ВУ, запросившее прерывание, после этого происходит обработка прерывания для него.
4. После опроса флагов всех ВУ и восстановления содержимого регистров процессора, ПОП завершается.

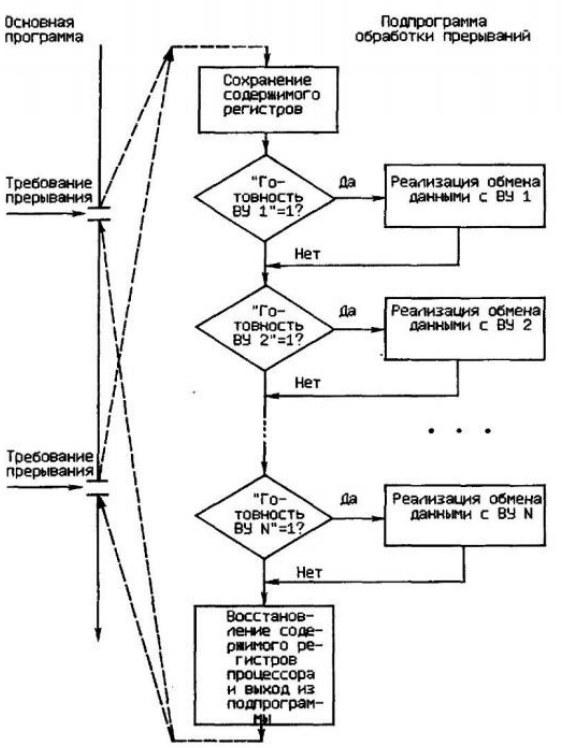


Рис. 8.13. Структура подпрограммы обработки прерывания и ее связь с основной программой

Недостаток:

1. Существенно увеличивается время обработки тех ВУ, которые опрашиваются последними.

Использование векторов прерываний

В этом случае существует своя ПОП для каждого ВУ.

Вектор прерывания – адрес ячейки основной памяти микроЭВМ в который хранится либо первая команда ПОП этого ВУ, либо адрес начала такой ПОП.

Структура

2. В конце цикла исполнения проверяется наличие требования прерывания от ВУ.
3. Если он есть и прерывание в процессоре разрешено, то ВУ с наивысшим приоритетом, отправляется сигнал «Предоставление прерывания (вх.)», и если ВУ не требовало обслуживания, его контроллер формирует сигнал «Предоставление прерывания (вых.)», который отправляется на следующее по приоритету ВУ. Иначе контроллер ВУ выдает в шину адреса вектор прерывания.
4. Происходит выполнение ПОП этого ВУ.
5. После выполнения ПОП, контроллер ВУ формирует сигнал «Предоставление прерывания (вых.)», который

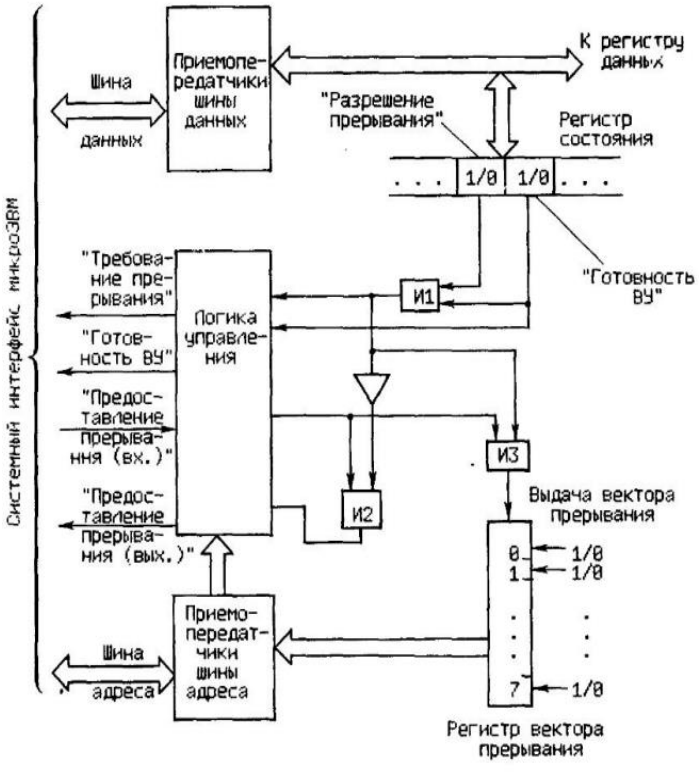


Рис. 8.14. Формирование вектора прерывания в контроллере ВУ

отправляется на следующее по приоритету ВУ.

Сигнал «Предоставление прерывания (вых.)» формируется с помощью схемы И2 и используется для организации последовательного опроса готовности ВУ, т. е. реализации требуемых приоритетов ВУ.

40. Организация прямого доступа к памяти. Контроллер ПДП.

Прямой доступ к памяти (англ. *direct memory access*, *DMA*) — режим обмена данными между устройствами компьютера или же между устройством и основной памятью, в котором центральный процессор (ЦП) не участвует. Так как данные не пересылаются в ЦП и обратно, скорость передачи увеличивается.

Этот тип обмена реализуется полностью аппаратно и управляется контроллером ПДП.

Организация ПДП

Особенности ПДП

1. Возможность начальной загрузки программ в основную память микроЭВМ из устройства ввода.
2. Обеспечивает возможность использования в микроЭВМ быстродействующих внешних запоминающих у-в.

Для экономии ресурсов контроллер ПДП не имеет свои ресурсы, а подключается к шинам данных (ШД) и адреса (ША) системного интерфейса ЭВМ, что делает невозможным одновременное использование шин контроллером ПДП и процессором.

Эта проблема решается двумя способами:

1. Захват цикла
    a. Простой захват цикла.
    Передача происходит в те машинные циклы, в которых процессор не обменивается данными с памятью. Пометка таких циклов выполняется либо с помощью спец. указывающего цикла, либо такие циклы выбираются с помощью спец. селектирующих схем, что усложняет конструкцию процессоров.
    При таком способе организации обмена ПДП не снижает производительности микроЭВМ, но обмен возможно только в случайные моменты времени одиночными байтами или словами.

    b. Захват цикла с принудительным отключением процессора от шин системного интерфейса.
    Для его реализации такого режима ПДП системный интерфейс (СИ) дополняется двумя линиями для передачи управляющих сигналов «Требование ПДП» (ТПДП) и «Предоставление ПДП» (ППДП).

1. ТПДП формируется контроллером ПДП.

2. Получив сигнал ТПДП, процессор приостанавливает выполнение очередной команды, не дожидаясь её завершения, выдает в системный интерфейс ППДП и отключается от шин СИ
3. Контроллер ПДП получает управления над шинами СИ и осуществляет обмен одним байтом или словом данных с памятью микроЭВМ.
4. Контроллер ПДП возвращает управление СИ процессору.

2. Блокировка процессора
   Отличается от «Захвата цикла» тем, что управление СИ передается контроллеру ПДП не на время обмена одним байтом, а на время обмена блоком данных.

Контроллер ПДП ввода данных из ВУ в режиме «Захват цикла»

1. Процессор загружает в СК контроллера количество принимаемых байтов, а в РА контроллера начальный адрес области памяти для вводимых данных.
2. Байты данных из ВУ поступают в РД контроллера, при этом каждый байт сопровождается управляющим сигналом из ВУ «Ввод данных», который обеспечивает запись байта в РД контроллера. По этому же сигналу при ненулевом состоянии СК контроллер формирует сигнал ТПДП.
3. По ответному сигналу процессора ППДП контроллер выставляет на ША и ЩД содержимое своих РА и РД.
4. Формируя приказ «Вывод», контроллер ПДП обеспечивает запись байта данных из своего регистра данных в память микроЭВМ.
5. По тому же сигналу ППДП содержимое СК декрементируется, а содержимое РА обновляется. Как только СК станет равным нулю, контроллер прекратит формирование сигналов ТПДП

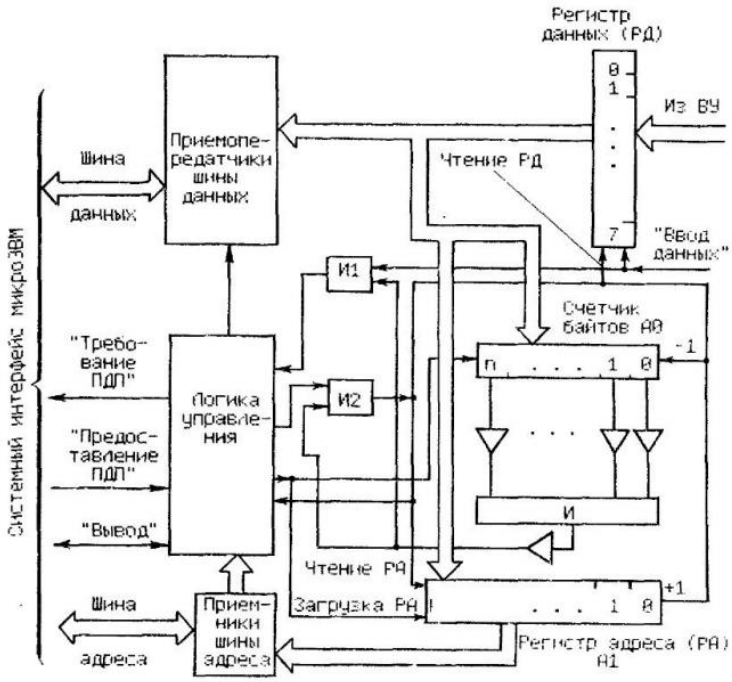


Рис. 8. 16. Контроллер ПДП для ввода данных из ВУ в режиме «Захват цикла»