Государственное бюджетное профессиональное образовательное учреждение
«Байконурский электрорадиотехнический техникум имени М.И. Неделина»
(ГБ ПОУ «БЭРТТ»)

Методические указания по выполнению лабораторных работ

по дисциплине «Архитектура компьютерных систем»

для специальности 09.02.03. «Программирование в компьютерных системах»
(базовый уровень)

г. Байконур

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Данные методические указания по выполнению лабораторных работ составлены в соответствии с ФГОС по специальности СПО 09.02.03. «Программирование в компьютерных системах» (базовый уровень) по дисциплине «Архитектура компьютерных систем».

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся в ходе освоения дисциплины должен:

уметь:

– получать информацию о параметрах компьютерной системы;

– подключать дополнительное оборудование и настраивать связь между элементами компьютерной системы;

– производить инсталляцию и настройку программного обеспечения компьютерных систем;

знать:

– базовые понятия и основные принципы построения архитектур вычислительных систем;

– типы вычислительных систем и их архитектурные особенности;

– организацию и принцип работы основных логических блоков компьютерных систем;

– процессы обработки информации на всех уровнях компьютерных архитектур;

– основные компоненты программного обеспечения компьютерных систем;

– основные принципы управления ресурсами и организации доступа к этим ресурсам

Перечень лабораторных работ

Раздел 1	Перевод чисел из одной системы счисления в другую
Раздел 1	Выполнение операций над числами в естественной и нормальной формах
Раздел 2	Работа и особенности логических элементов ЭВМ
Раздел 2	Программирование разветвляющегося процесса
Раздел 2	Построение последовательности машинных операций для реализации простых вычислений
Раздел 2	Программирование циклов с переадресацией
Раздел 2	Изучение команд пересылки данных
Раздел 2	Изучение арифметических команд
Раздел 2	Динамическая память. Принцип работы. Обобщенная структурная схема памяти
Раздел 2	Статическая память. Применение и принцип работы
Раздел 2	Принцип работы кэш-памяти
Раздел 2	Архитектура системной платы. Внутренние интерфейсы системной платы
Раздел 2	Интерфейсы периферийных устройств
Раздел 2	Программирование арифметических и логических команд
Раздел 2	Программирование ввода-вывода
Раздел 2	Идентификация и установка процессора
Раздел 2	Изучение логических команд и команд сдвигов
Раздел 2	Изучение команд условного перехода
Раздел 2	Программирование переходов
Раздел 2	Изучение команд передачи управления

Правила выполнения лабораторных работ

Цель проведения лабораторных работ — обеспечение контроля над полученными знаниями студентов, предоставление возможности:

– студентам - проявить приобретенные в результате освоение дисциплины;

– преподавателю - оценить степень усвоения материала студентом.

Студент перед выполнением лабораторной работы должен строго выполнить весь объем домашней подготовки, указанный в описании соответствующей лабораторной работы.

Отчет о лабораторной работе должен содержать исчерпывающие систематизированные сведения о выполненной работе. Оформление отчета о лабораторной работе должно быть предусмотрено заданием или планом выполнения работы. Отчет составляется исполнителем работы и защищается в индивидуальном порядке в сроки, установленные преподавателем.

Отчет по лабораторной работе должен содержать:

– дата, наименование и цель работы;

– практическая часть – описание хода работы, алгоритм решения задач для самостоятельного выполнения;

– вывод по лабораторной работе.

Отчет составляется студентом и защищается в индивидуальном порядке. Защита проводиться в два этапа:

1. Демонстрация выполненной работы

2. Ответы на контрольные вопросы.

Лабораторная работа №1

«Перевод чисел из одной системы счисления в другую»

Цель работы: приобрести навыки преобразования чисел различных систем счисления.

Предварительная подготовка: изучить материал раздела «Представление информации в вычислительных системах» (по конспекту).

Задание 1. Перевести число из 2с/с в 10 с/с, 8 с/с, 16 с/с по вариантам:

№ варианта	P = 2	№ варианта	P = 2
1.	1001,101	**16.**	1101,0011
2.	11011,1101	**17.**	10110,001
3.	10101,001	**18.**	110110,1010
4.	10101,0111	**19.**	100110,001
5.	1101,100101	**20.**	10011,0011
6.	101101,101	**21.**	1101101,1011
7.	1100111,101	**22.**	1001001,0011
8.	10010,011	**23.**	10101,1110
9.	11100101,100	**24.**	110110,1101
10.	110101111,101	**25.**	110110,001
11.	100100,0101	**26.**	101100,1010
12.	101011,011	**27.**	110110,101
13.	110000,10111	**28.**	1011101,110
14.	101010,0110	**29.**	111011,1010
15.	1100,0111	**30.**	1111010,1101

Задание 2. Перевести число из 10 с/с в 2 с/с, 8 с/с, 16 с/с по вариантам. Для двоичной системы счисления при переводе дробной части получить 6-7 знаков после запятой. При переводе чисел в 8 с/с и 16 с/с пользоваться правилами перевода чисел из 10 с/с в любую другую; при переводе дробной части получить 4-5 знаков после запятой.

№ варианта	P = 10	№ варианта	P = 10
1.	136,15	**16.**	178,35
2.	213,127	**17.**	135,123
3.	123,64	**18.**	126,29
4.	236,18	**19.**	162,157
5.	147,82	**20.**	186,64
6.	184,38	**21.**	165,127
7.	199,32	**22.**	146,142
8.	132,96	**23.**	159,33
9.	101,56	**24.**	149,201
10.	231,38	**25.**	155,33
11.	177,853	**26.**	175,391
12.	97,456	**27.**	221,76
13.	139,69	**28.**	123,521
14.	153,238	**29.**	157,25
15.	201,33	**30.**	198,76

Задание 3. Перевести числа:

– из 8 с/с и 16 с/с в десятичную систему счисления;

– из 8 с/с в шестнадцатеричную систему счисления через двоичную;

– из 16 с/с в восьмеричную систему счисления через двоичную.

по вариантам:

№ варианта	P = 8	P = 16	№ варианта	P = 8	P = 16
1.	574,03	1A05	**16.**	147,42	84C
2.	652,42	931C	**17.**	543,35	C1F
3.	374,71	2001	**18.**	732,61	A10C
4.	431,34	FD0	**19.**	621,76	AE0
5.	106,25	84B	**20.**	452,34	ABC2
6.	227,34	7A3D	**21.**	634,15	1BC4
7.	361,17	946F	**22.**	721,62	20FF
8.	253,51	160E	**23.**	642,71	BC0
9.	327,16	18AB	**24.**	741,52	4571
10.	174,43	20F	**25.**	246,31	23DE
11.	554,24	39D1	**26.**	316,64	13CB
12.	710,36	FF0	**27.**	327,07	AC5D
13.	325,64	A0E	**28.**	561,67	DF91
14.	541,56	34F1	**29.**	723,42	E75A
15.	371,37	103D	**30.**	173,64	A01F

Задание 4. Перевести число из 10 с/с в двоично-десятичную систему счисления по вариантам:

№ варианта	P = 10	№ варианта	P = 10
1.	1026,256	**16.**	3402,869
2.	532,1205	**17.**	4710,138
3.	674,053	**18.**	8175,805
4.	358,435	**19.**	9115,052
5.	15,4723	**20.**	8520,178
6.	3905,463	**21.**	8053,1564
7.	675,035	**22.**	5316,035
8.	5302,68	**23.**	3942,578
9.	906,1273	**24.**	8065,215
10.	8145,605	**25.**	5746,243
11.	40513,104	**26.**	2705,989
12.	2386,991	**27.**	1750,989
13.	785,1001	**28.**	7435,095
14.	5273,865	**29.**	8901,237
15.	8064,193	**30.**	4650,109

Контрольные вопросы:

1. Что такое система счисления? Основание системы счисления?
2. Что такое позиционные и непозиционные системы счисления?
3. Что такое двоичная, восьмеричная, шестнадцатеричная системы счисления?
4. Правила перевода чисел из одной системы счисления в другую.

Лабораторная работа №2

«Выполнение операций над числами в естественной и нормальной формах»

Цель работы: научиться применять специальное кодирование чисел при выполнении арифметических операций.

Предварительная подготовка: изучить материал раздела «Представление информации в вычислительных системах» (по конспекту).

Теоретическая часть:

Формы представления чисел.

В вычислительных машинах применяются две формы представления двоичных чисел – естественная форма или форма с фиксированной запятой (точкой) и нормальная форма или форма с плавающей запятой (точкой).

В естественной форме положение в разрядной сетке (общее число разрядов, отведенное для представления чисел, с указанием местоположения каждого из них) запятой, отделяющей целую часть числа от дробной части, постоянно для всех чисел. Диапазон значащих чисел небольшой и при m-разрядной целой части и s-разрядной дробной части числа без учета знака составляет от P^{-s} до $P^m - P^{-s}$. Кроме того, если в результате операции получится число, выходящее за допустимый диапазон, происходит переполнение разрядной сетки, и дальнейшие вычисления теряют смысл. Следовательно, необходимо прогнозировать результаты обработки с целью соответствующего масштабирования исходных данных. По этим причинам естественная форма представления используется как вспомогательная и только для целых чисел.

В нормальной форме каждое число представляется как $N = \pm MP^{\pm R}$, где M – мантисса числа ($|M|<1$), R – порядок (целое число), P – основание системы счисления. Абсолютное значение порядка определяет число разрядов, на которое смещена запятая, отделяющая целую часть числа от дробной части, а знак порядка – направление смещения этой запятой. Таким образом, с изменением значения порядка запятая меняет своё положение, как бы "плавает" в изображении числа. Диапазон значащих чисел весьма велик и при m-разрядной мантиссе и s-разрядном порядке (без учета знаков порядка и мантиссы) составляет от $P^{-(m+P^s-1)}$ до $(1-P^{-m})\cdot P^{P^s-1}$. В связи с этим нормальная форма представления является основной в современных ЭВМ.

Прямой, обратный, дополнительный коды чисел.

В ЭВМ с целью упрощения арифметических операций применяют специальные коды для представления чисел. При помощи этих кодов:

- автоматически определяется знак результата;
- операция вычитания сводится к арифметическому сложению кодов чисел;
- упрощается операционная часть ЭВМ.

Среди арифметических операций основными являются операции сложения и вычитания, поскольку помимо самостоятельного значения, они лежат в основе операций умножения и деления, соответственно.

С целью удобства технической реализации операция вычитания заменяется операцией сложения. При этом исходные операнды (числа, участвующие в операции) должны быть представлены в обратном или в дополнительном коде.

Задание 1. Выполнить операции сложения, вычитания, умножения и деления над числами в двоичной, восьмеричной и шестнадцатеричной системе счисления по вариантам. Произвести проверку, выполнив эти действия в 10 с/с (перевести в 10 с/с исходные числа и результат каждого действия).

№ варианта	P = 2	P = 8	P = 16	№ варианта	P = 2	P = 8	P = 16
1.	x=1110011 y=1011	x=7162 y=53	x=72ABF y=B5	**16.**	x=11100111 y=1011	x=22217 y=61	x=41FFB y=53
2.	x=1000001 y=1110	x=13147 y=37	x=3CC5 y=2F	**17.**	x=10010101 y=1101	x=71157 y=77	x=ACDE8 y=B5
3.	x=10100001 y=111	x=26220 y=56	x=68E1A y=8A	**18.**	x=1110001 y=101	x=4141 y=15	x=48A04 y=7C
4.	x=1100011 y=1010	x=21407 y=61	x=8859F y=C5	**19.**	x=11011001 y=1011	x=35430 y=37	x=CEB3 y=5F
5.	x=110011 y=1001	x=63616 y=67	x=22BC3 y=23	**20.**	x=1001101 y=1010	x=7111 y=27	x=1D6D8 y=7A
6.	x=1010101 y=10001	x=10274 y=52	x=529B3 y=67	**21.**	x=1011101 y=10101	x=32574 y=34	x=1593C y=2D
7.	x=1100111 y=1101	x=14630 y=32	x=BEC62 y=D2	**22.**	x=10001011 y=10001	x=5211 y=37	x=5A858 y=A8
8.	x=11001001 y=1001	x=6522 y=37	x=50385 y=63	**23.**	x=1001110 y=100	x=25154 y=36	x=4C3C9 y=5F
9.	x=1001010 y=101	x=32414 y=34	x=45FCF y=E7	**24.**	x=11001100 y=111	x=57602 y=62	x=4EE2E y=76
10.	x=111001 y=110	x=23064 y=24	x=72D9B y=E9	**25.**	x=11001101 y=101	x=14335 y=23	x=4952A y=5E
11.	x=1100110 y=11000	x=36226 y=57	x=4AAD3 y=7D	**26.**	x=1110011 y=110	x=7771 y=57	x=CF33C y=F6
12.	x=101010 y=1001	x=10770 y=31	x=2154E y=3B	**27.**	x=10001111 y=1011	x=55322 y=62	x=5D8CC y=8C
13.	x=101001 y=100	x=10756 y=22	x=78273 y=95	**28.**	x=1011010 y=110	x=5211 y=37	x=92365 y=F1
14.	x=11000101 y=1001	x=11324 y=12	x=A89C0 y=C6	**29.**	x=1110011 y=101	x=7603 y=23	x=CA8EE y=D6
15.	x=1010101 y=10101	x=31567 y=37	x=37A50 y=38	**30.**	x=1011011 y=111	x=12324 y=76	x=55EE7 y=5F

Задание 2. Выполнить сложение двоичных чисел в обратном и дополнительном кодах по вариантам:

№ варианта	1)	2)	№ варианта	1)	2)
1.	x= -11100 y= -1011	x=1110 y= -110	**16.**	x= -100111 y=1011	x= -101011 y=111
2.	x=1000 y= -1110	x= -11100 y=1010	**17.**	x= -10101 y= -1101	x=111 y= -10101
3.	x=1010 y= -11101	x= -1110 y= -111	**18.**	x= -110001 y=101	x= -111 y= -101000
4.	x= -1100 y=1010	x= -10001 y=111	**19.**	x= -110110 y=1011	x= -11001 y=11111
5.	x=11001 y= -1001	x= -10101 y=1011	**20.**	x= -1001101 y=1010	x=111111 y= -1000000
6.	x= -1010 y=10001	x=1011 y= -11100	**21.**	x=10111 y= -101011	x=10000 y= -111111

7.	x=11001 y= -1101	x= -10111 y=110	**22.**	x=101011 y= -10001	x= -110000 y=1111111
8.	x=11001 y= -10010	x= -110011 y=1110	**23.**	x=1110 y= -111100	x=1110111 y= -11001
9.	x= -1010 y= -101	x= -1100 y= -11011	**24.**	x=110011 y= -111	x=101011 y= -10011
10.	x= -111001 y= -110	x=1111 y= -11111	**25.**	x=11001 y=-11101	x=111000 y= -1000111
11.	x= -1100110 y=11000	x= -110011 y= -11011	**26.**	x= -110011 y=1101	x=110001 y=110
12.	x= -101010 y=1001	x= -1110 y= -1010	**27.**	x=100011 y= -1011	x= -10110 y= -11001
13.	x= -101001 y=100	x=111110 y= -1111	**28.**	x=1011010 y=110	x= -11001 y= -11110
14.	x= -110001 y=1001	x= -101010 y= -1010	**29.**	x=10011 y= -10101	x= -110000 y= -111
15.	x= -1010101 y=10101	x=1100110 y= -100001	**30.**	x= -11011 y= -10111	x= -1110 y=111110

Контрольные вопросы:

1. Правила сложения, вычитания, умножения в двоичной системе счисления.
2. Прямой, обратный, дополнительный коды чисел.

Лабораторная работа №3

«Работа и особенности логических элементов ЭВМ»

Цель работы: Освоить работу логических элементов.

Предварительная подготовка: изучить материал раздела «Архитектура и принципы работы основных логических блоков вычислительных систем (ВС)» (по конспекту).

Задание:

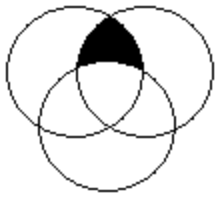
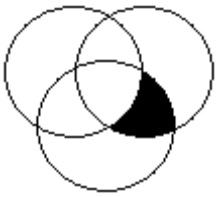
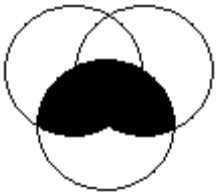
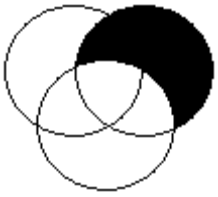
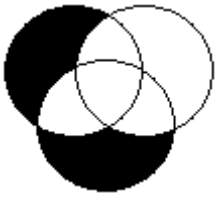
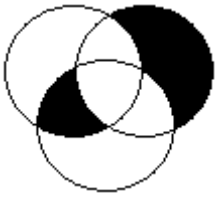
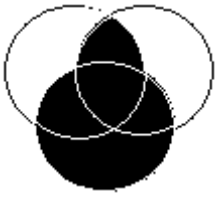
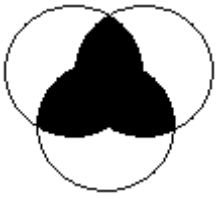
1. Изобразить выражения согласно варианту (таблица 1) на диаграммах Эйлера.
2. Описать логическим выражением диаграмму Эйлера согласно варианту (таблица 2).
3. Построить временные диаграммы работы элементов И, ИЛИ, НЕ, исключающее ИЛИ. Записать таблицы истинности.

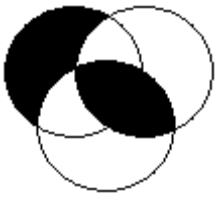
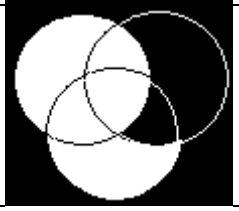
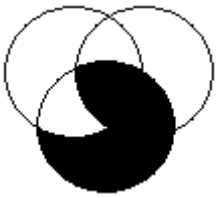
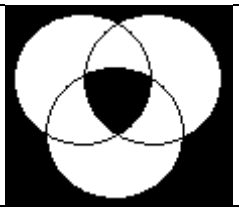
Таблица 1

Вариант	Выражения
1	A∨(B&C) A-(B∨$\overline{C}$)
2	A⊕$\overline{B-C}$ A∨$\overline{B}$∨C
3	A&$\overline{B\&C}$ A∨B-C
4	A∨B⊕C $\overline{A}$∨(B&C)
5	$\overline{A\&B\&C}$ A∨($\overline{B}$⊕$\overline{C}$)
6	A&B-C A⊕$\overline{B\vee C}$
7	A⊕B∨$\overline{C}$ A-(B&C)
8	$\overline{A\&\overline{B\vee C}}$ (A-B)∨C

Вариант	Выражения
9	A-B-C $\overline{A\vee\overline{B\&C}}$
10	A-(B&C) $\overline{\overline{A}\&B}$⊕C
11	$\overline{A}$⊕B∨C A-$\overline{B-C}$
12	A&B&C $\overline{A\vee B}$-C
13	A⊕B⊕C $\overline{A}$⊕$\overline{B-C}$
14	(A⊕B)∨$\overline{C}$ $\overline{A\&B}$⊕$\overline{C}$
15	$\overline{A}$&B-$\overline{C}$ A⊕(B∨C)

Таблица 2

Вариант	Выражения
1	
2	
3	
4	
5	
6	
7	
8	

Вариант	Выражения
9	
10	
11	
12	
13	
14	
15	

Контрольные вопросы:

1. Логические элементы И, ИЛИ, НЕ. Обозначение, таблицы истинности, назначение.
2. Логические элементы И-НЕ, ИЛИ-НЕ, И-ИЛИ-НЕ. Обозначение, таблицы истинности, назначение.

Лабораторная работа №4

«Программирование разветвляющегося процесса»

Цель работы: рассмотреть программу разветвляющегося процесса.

Предварительная подготовка: изучить материал раздела «Архитектура и принципы работы основных логических блоков вычислительных систем (ВС)» (по конспекту).

Методические указания:

Если вычислительный процесс зависит от определенных условий и реализуется по одному из нескольких заранее предусмотренных направлений, он называется разветвляющимся вычислительным процессом, а каждое из этих направлений – ветвью вычислений. Для выбора ветви вычислений в Паскале используются операторы IF и CASE.

Однако прежде, чем перейти к рассмотрению этих операторов, необходимо познакомиться с понятиями составного оператора, логическими операциями и выражениями.

Составной оператор.

Составной оператор предписывает выполнение составляющих его операторов в порядке их написания. Зарезервированные слова BEGIN и END являются операторными скобками. Формат оператора:

BEGIN {Начало составного оператора}
<Оператор 1;>
<Оператор 2;>
...
<Оператор n>
END; {Конец составного оператора}

Составной оператор используется в тех конструкциях, где по синтаксису языка должен быть только один оператор, а для решения задачи требуется более одного. В составном операторе все операторы 1, 2,..., n выполняются последовательно ДРУГ за другом.

Логические выражения.

Одним из нечисловых видов данных является тип BOOLEAN. Булевы (логические) переменные имеют только два значения: FALSE (ложь), TRUE (истина). Существует несколько форм конструирования логического выражения:

- константа, описанная в разделе CONST;
- переменная, которой можно присвоить булевы значения (например FLAG:= TRUE);
- отношение между переменными скалярных и некоторых структурированных типов.

В Паскале допускаются отношения, перечисленные в таблице 16.

Таблица 16

Отношение	Содержание	Отношение	Содержание
=	равно	o	не равно
>	больше, чем	>=	больше, чем ... или равно
<	меньше, чем	<=	меньше, чем ... или равно

Задание 1. Пусть заданы вещественные переменные A, B и логическая переменная FLAG. Требуется построить примеры простых логических выражений, содержащих отношения между A и B.

Если:

```
VAR
FLAG, FLAG1, FLAG2: BOOLEAN;
A, B: REAL;
```

тогда допустимы выражения вида:

FLAG := A <= B;

Значение TRUE 'истина' присваивается переменной FLAG, если A меньше или равно B.

FLAG 1 := A <> B;

Значение TRUE 'истина' присваивается переменной FLAG1, если A не равно B.

FLAG2 := A = B;

Значение TRUE 'истина' присваивается переменной FLAG2, если A равно B.

Помимо указанных выше отношений (таблица 16), логические выражения конструируются с помощью булевых операций, описанных в таблице 17.

Таблица 17

Математическое обозначение	Название	Обозначение в Паскале
¬	Логическое отрицание НЕ	NOT
∧	Логическая конъюнкция И	AND
∨	Логическая дизъюнкция ИЛИ	OR
⊕	Исключающее ИЛИ	XOR

Задание 2. Сформулировать логическое условие попадания точки с координатами (x, y) в область S (рисунок 2).

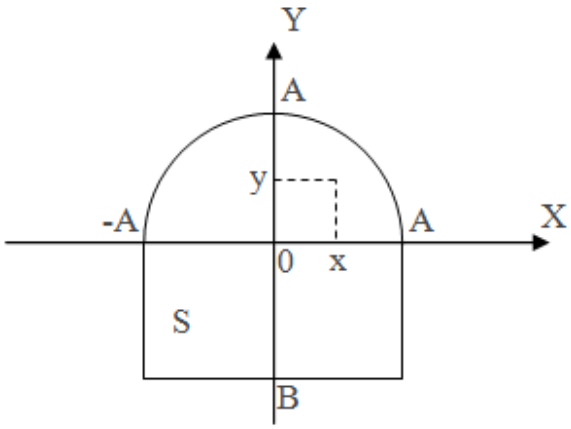


Рисунок 2

Пусть:

VAR FLAG: BOOLEAN;

Уравнение окружности, которая ограничивает область S в первом и втором квадранте системы координат XOY имеет вид:

$$Y = \sqrt{A^2 - X^2}$$

Тогда величину FLAG, которая принимает значение TRUE в том случае, когда точка с координатами (x, y) принадлежит области S, можно найти по формуле: FLAG:=(X>=-A) AND (X<=A) AND (((Y<=SQRT(A*A-X*X)) AND (Y>=0)) OR ((Y<0) AND (Y>=B)));

В языке Паскаль логическое выражение просчитывается до тех пор, пока результат не становится очевидным. После чего вычисления прекращаются. Так в нашем случае используется конъюнкция трех условий: X больше -A, X меньше A и ограничение на значение Y. Достаточно любой логической величине принять значение FALSE и остальные величины, стоящие правее в логическом выражении, уже не просчитываются, так как переменная FLAG независимо от значений оставшихся отношений будет равна FALSE. В нашем случае это удобно! Потому, что уравнение окружности определено для значений X, удовлетворяющих условию -A < X < A. Именно это условие и проверяется в двух левых отношениях, поэтому используемое логическое выражение Y<=SQRT(A* A - X*X) для расчета переменной FLAG корректно для любых значений X.

```
PROGRAM LAB4;
VAR
A, B, X, Y: REAL;
FLAG: BOOLEAN;
BEGIN
WRITELN('Введите параметры A и B');
READLN(A, B);
WRITELN('Введите координаты X и Y); READLN(X, Y);
FLAG := (X>—A) AND (X<=A) AND (((Y<=SQRT(A*A - X*X)) AND (Y>-0))
OR ((Y<0) AND (Y>~B)));
IF FLAG THEN WRITELN('Точка в области S')
ELSE WRITELN('Точка вне области S')
END.
```

В стандартном Паскале предусмотрен порядок старшинства операций в булевых выражениях: Высший - (скобки); NOT; AND; (OR, XOR); (>, =, >-, <-, <>) - низший. Однако в различных версиях языка эти требования могут и не соблюдаться, поэтому надежнее использовать скобки для уточнения последовательности вычислений.

Существуют встроенные булевы функции, наиболее известные из которых ODD(X), EOF(F), EOLN(F), описание которых приведено в таблице 10.

Логическое выражение может быть достаточно сложным и включать в себя арифметические и логические функции, например: FLAG := ODD(I*3+K) AND ((SQR(C) > SIN(D/2)) OR (A = 5));

Переменная FLAG принимает значение TRUE, если целочисленное выражение I*3 + K принимает нечетное значение и квадрат C больше, чем синус D, деленной пополам, или A равно 5. В противном случае FLAG принимает значение FALSE.

В приведенных примерах в правой части оператора присваивания расположено логическое выражение, а в левой части – логическая переменная.

Лабораторная работа №5

«Построение последовательности машинных операций для реализации простых вычислений»

Цель работы: изучить структуру и форматы машинных команд.

Предварительная подготовка: изучить материал раздела «Архитектура и принципы работы основных логических блоков вычислительных систем (ВС)» (по конспекту).

Теоретическая часть:

Машинная команда представляет собой код, определяющий операцию вычислительной машины и данные, участвующие в операции. Команда должна содержать в явной или неявной форме информацию об адресе результата операции, и об адресе следующей команды.

Машинная операция – это действия машины по преобразованию информации, выполняемые под воздействием одной команды.

Программа – последовательность команд, отображающих все действия, необходимые для решения задачи по некоторому алгоритму.

Машинный такт – период тактовой частоты работы процессора.

Машинный цикл – количество машинных тактов, требуемых для выполнения одной команды.

По характеру выполняемых операций различают следующие основные группы команд:

- арифметические операции над числами с фиксированной или плавающей точкой;
- команды двоично-десятичной арифметики;
- логические (поразрядные) операции;
- пересылка операндов;
- операции ввода-вывода;
- передача управления;
- управление работой центрального процессора.

Машинная команда состоит из операционной и адресной частей. Эти части могут состоять из нескольких полей. В общем виде машинная команда имеет следующую структуру:

Операционная часть содержит код, задающий вид операции (сложение, умножение, передача и т.д.).

Адресная часть содержит информацию об адресах операндов, результата операции и следующей команды.

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

Формат команды – это структура команды с разметкой номеров разрядов, определяющих границы отдельных полей команды.

Возможные структуры машинных команд

Четырехадресная структура содержит наиболее полную информацию о выполняемой операции, включает поле кода операции и четыре адреса для указания ячеек памяти двух операндов, ячейки результата операции, и ячейки, содержащей адрес следующей команды. Такой порядок выборки команд называется ***принудительным***. Он использовался в первых моделях вычислительных машин, имеющих небольшое число команд и очень незначительный объем ОП, поскольку длина такой команды зависит от разрядности адресов операндов и результата.

КОП	Адрес 1 операнда	Адрес 2 операнда	Адрес результата	Адрес след. команды

Трехадресная структура используется в вычислительных машинах, построенных так, что после выполнения команды по адресу K (команда занимает L ячеек памяти) выполняется команда по адресу K+L. Такой порядок выборки команд называется ***естественным***. Он нарушается только специальными командами передачи управления. При естественном порядке выборки адрес следующей команды формируется в устройстве, называемом счетчик адреса команд. В этом случае команда становится трехадресной.

КОП	Адрес 1 операнда	Адрес 2 операнда	Адрес результата

Двухадресная структура используется в вычислительных машинах, построенных так, что результат операции будет всегда помещаться в фиксированный регистр процессора, например на место первого операнда. В этом случае адрес результата может явно не указываться.

КОП	Адрес 1 операнда	Адрес 2 операнда

Одноадресная структура подразумеваемые адреса имеют результат операции и один из операндов. При этом один из операндов и результат операции размещаются в одном фиксированном регистре. Выделенный для этой цели внутренний регистр процессора получил название ***аккумулятор***. Адрес другого операнда указывается в команде.

КОП	Адрес 1 операнда

Безадресная структура фиксирует адреса обоих операндов и результата операции, например при работе со стековой памятью.

КОП

Обычно в вычислительной машине используется несколько форматов команд разной длины (чаще всего безадресные, одноадресные и двухадресные).

Контрольные вопросы:

1. Основы построения ЭВМ.
2. Внутренняя организация процессора.

Лабораторная работа №6

«Программирование циклов с переадресацией»

Цель работы: Разработать программу функции.

Предварительная подготовка: изучить материал раздела «Архитектура и принципы работы основных логических блоков вычислительных систем (ВС)» (по конспекту).

Задание:

1. Разработать программу вычисления и вывода значения функции:

$$y = \begin{cases} F_i(x), & \text{при } x \geq a, \\ F_j(x), & \text{при } x < a, \end{cases}$$

для вводимого из IR значения аргумента *x*.

2. Исходя из допустимых пределов изменения аргумента функций и значения параметра *a* для своего варианта задания выделить на числовой оси *Ox* области, в которых функция *y* вычисляется по представленной в п. 1 формуле, и недопустимые значения аргумента. На недопустимых значениях аргумента программа должна выдавать на OR максимальное отрицательное число: 199 999.

3. Ввести текст программы в окно **Текст программы,** при этом возможен набор и редактирование текста непосредственно в окне **Текст программы** или загрузка текста из файла, подготовленного в другом редакторе.

4. Ассемблировать текст программы, при необходимости исправить синтаксические ошибки.

5. Отладить программу. Для этого:

а) записать в 1R значение аргумента $x>a$ (в области допустимых значений);

б) записать в PC стартовый адрес программы;

в) проверить правильность выполнения программы (т-е. правильность результата и адреса останова) в автоматическом режиме. В случае наличия ошибки выполнить пп. 5, *г* и 5, *д;* иначе перейти к п. 5, е;

г) записать в PC стартовый адрес программы;

д) наблюдая выполнение программы в режиме Шаг, найти команду, являющуюся причиной ошибки; исправить ее; выполнить пп. 5, а — 5, в;

е) записать в IR значение аргумента $x<a$ (в области допустимых значений); выполнить пп. 5, *б* и 5, в;

ж) записать в IR недопустимое значение аргумента *x* и выполнить пп. 5, *б* и 5, *е.*

6. Для выбранного допустимого значения аргумента x наблюдать выполнение отлаженной программы в режиме Шаг и записать, содержимое регистров ЭВМ перед выполнением каждой команды.

При решении задач, связанных с обработкой массивов, возникает необходимость изменения исполнительного адреса при повторном выполнении некоторых команд. Эта задача может быть решена путем использования косвенной адресации.

Лабораторная работа №7-8

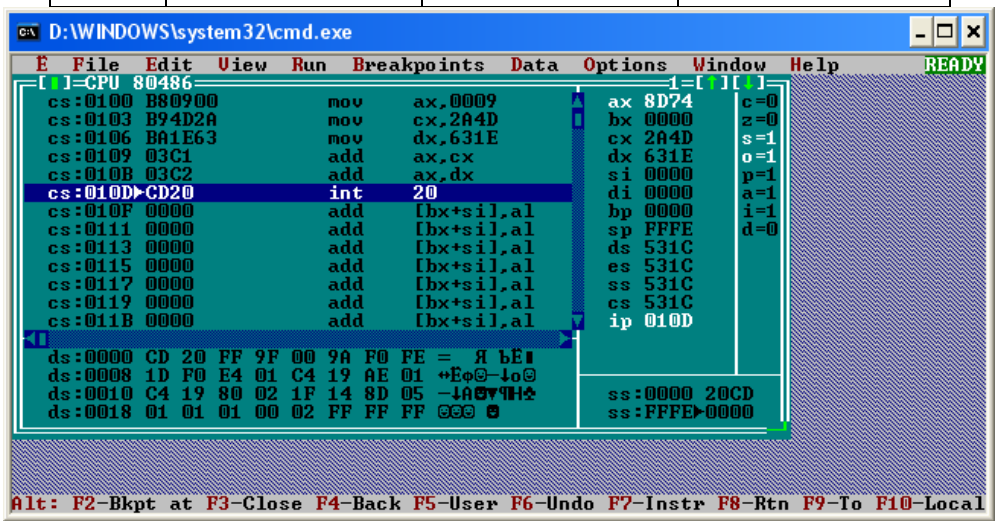
«Изучение арифметических команд и команд пересылки данных»

Цель работы: приобрести практические навыки работы с микропроцессором Intel 80x86, изучить возможности устройства Intel 80x86, практически освоить режимы его работы.

Предварительная подготовка: изучить материал раздела «Архитектура и принципы работы основных логических блоков вычислительных систем (ВС)» (по конспекту).

Методические указания:

№ п/п	1-ое слагаемое	2-ое слагаемое	3-ое слагаемое
1	0009	2A4D	631E




```
cdseg segment 'code'
      org 100h
start:

mov     ax,0009h
mov     cx,2A4Dh
mov     dx,631Eh

add     ax,cx
add     ax,dx

int     20h

cdseg ends
end start
```

Ответ:

AX= $8D74_{16}$.

Например, необходимо найти сумму ряда чисел. При этом задача состоит в том, чтобы осуществить суммирование нескольких чисел сразу. Эти числа могут представлять собой совокупности входных сигналов, находящихся под управлением системы, число изделий (или число сообщений), изготовленных (или принятых) за определенный промежуток времени.

Программа сложения ряда чисел

№ п/п	Исходный ряд чисел
1	046E,05E7,240F,3031,0820,1AF9,1F32

```
cdseg segment 'code'
     assume cs:cdseg
org 100h
start:
mov si,offset array
lodsw
 mov    cx,ax    ; count of digits
 mov    bx,si    ; addr of array
 sub    ax,ax
m1:  add    ax,[bx]
    dec    cx
    je     fin
    add    bx,0002
jmp    m1

 fin:
    int    20h
array dw 7, 046Eh,05E7h,240Fh,3031h,0820h,1AF9h,1F32h
 cdseg ends
end start
```

Ответ:
A0E0 $_{16}$.

Рассмотрим арифметику для чисел, занимающих несколько ячеек. Задача состоит в том, чтобы сложить два числа длиной более 16 бит каждое. Исходные числа располагаются в области памяти таким образом, что сначала идут младшие разряды, а затем более старшие. Полученную сумму необходимо поместить в те ячейки памяти, где хранилось первое число.

№ п/п	1-ое слагаемое	2-ое слагаемое
1	D4241C879DAB	DB893E0731C5

```
cdseg segment 'code'
     assume cs:cdseg
org 100h
start:
 mov    cx,(offset data2 -offset data1)/2
 mov    di,offset data1
 mov    si,offset data2
m:
 mov   ax,[di]
 adc   ax,[si]
 stosw
  inc si
  inc si
 dec   cx
jne    m
 int    20h

 data1    dw  D424h,1C87h,9DABh
 data2    dw  DB89h,3E07h,31C5h
```

```
cdseg ends
end start
```

Ответ:
Сумма равна 1AFA D5A8 ECF70.

Обработка массивов информации и организация циклов

Рассмотрим следующую задачу. Массив чисел расположен в области памяти, начиная с адреса NNNN, и состоит из N элементов. Необходимо переслать массив в другую область памяти, начиная с адреса DDDD. В программе для МП Intel 80x86 в качестве адресного регистра используются регистры si и di. Необходимо учитывать, что при каждом проходе программа должна изменять содержимое обоих адресных регистров.

№ п/п	Исходный массив
1	D424,1C87,9DAB,DB89,3E07,31C5,14B2,4F87,8A00

Программа:

```
cdseg segment 'code'
org 100h
start:
 mov   cx,(offset data2 - offset data1)/2
 mov   si,offset data1
 mov   di,offset data2

 repne movsw

int   20h

 data1 dw D424h,1C87h,9DABh,DB89h,3E07h,31C5h,14B2h,4F87h,8A00h
 data2 dw 0AAAAh,0AAAAh,0AAAAh,0AAAAh,0AAAAh,0AAAAh,0AAAAh,0AAAAh,0AAAAh

cdseg ends
end start
```

Массив данных был перемещен в область памяти под названием data2.

В ряде случаев возникает необходимость выбрать из массива информации данные, представляющие собой ряд максимальных, либо минимальных величин. Предположим, что необходимо написать программу для решения следующей задачи. Дан массив A1, состоящий из N однобайтовых чисел. Необходимо переписать из массива A1 в массив B1 все числа в диапазоне от H1 до H2.

№ п/п	Исходный массив	Нижний Предел	Верхний предел
1	D424,1C87,9DAB,DB89,3E07,31C5,14B2	24C3	4433

```
cdseg segment 'code'
    assume cs:cdseg
org 100h
start:
 mov   cx,(offset dest - offset source)/2
```

```
 mov    si,offset source
 mov    di,offset dest
m1:
   lodsw
   cmp ax, 24C3h     ; min
    jb m2
    cmp ax, 4433h    ; max
    ja m2
    stosw
   m2:
loop m1

int    20h

source     dw  0D424h,1C87h,9DABh,0DB89h,3E07h,31C5h,14B2h
 dest      dw (offset dest - offset source)/2 dup (0)

cdseg ends
end start
```

Ответ: 3E07,31C5

Лабораторная работа №10

«Статическая память. Применение и принцип работы»

Цель работы: Изучить применение и принцип работы статической памяти.

Предварительная подготовка: изучить материал раздела «Архитектура и принципы работы основных логических блоков вычислительных систем (ВС)» (по конспекту).

Теоретическая часть:

Основой ячейки памяти в ЗУ статического типа является триггер. В качестве базовых элементов для реализации триггера могут использоваться как биполярные транзисторы, так и полевые. Однако первые не нашли широкого применения в силу большой потребляемой мощности построенных на их основе микросхем памяти. Поэтому оптимальным является использование полевых транзисторов. На рис.1 представлен триггер на МОП-транзисторах с индуцируемым р-каналом. Для отпирания такого транзистора напряжение на его затворе относительно истока должно быть меньше нуля: $U_{зи}<0$.

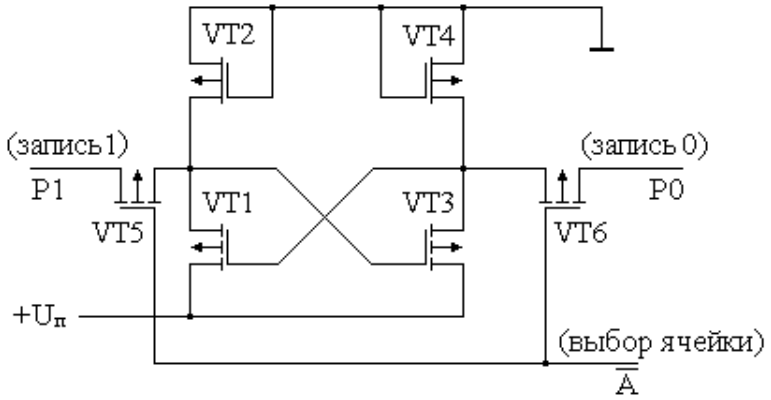


Рис.1. Принципиальная схема ячейки ОЗУ статического типа.

Пусть в исходном состоянии транзистор *VT*3 открыт, а *VT*1 закрыт (состояние хранения нуля). Транзисторы *VT*2 и *VT*4 выполняют роль резисторов, поэтому на стоке транзистора *VT*3 будет потенциал напряжения питания $+U_n$, а на стоке транзистора *VT*1 – нулевой потенциал. Транзисторы *VT*5 и *VT*6 осуществляют запись и считывание информации. В режиме хранения данных напряжения на разрядных линиях *P*0 и *P*1 равны нулю, а на линии $\overline{A}$ потенциал равен напряжению питания схемы $+U_n$. При этом напряжение стока-истока $U_{си.VT5}$ на транзисторе *VT*5 равно нулю, $U_{зи.VT5}>0$ и транзистор *VT*5 закрыт. Напряжение $U_{зи.VT6}$ транзистора *VT*6 равно нулю и он также закрыт.

Для установления триггера в единичное состояние (запись единицы) на линию $\overline{A}$ подается нулевой потенциал, а на разрядную линию *P*1 потенциал равный $+U_n$. При этом транзистор *VT*5 будет включен инверсно, т.е. истоком становится вывод, подсоединенный к разрядной линии *P*1. Напряжение затвор-исток инверсно включенного транзистора *VT*5 становится меньше нуля $U_{зи.VT5}<0$ и транзистор *VT*5 открывается. Положительный сигнал поступает на затвор транзистора *VT*3, при этом $U_{зи.VT3}$ становится равным нулю, и транзистор *VT*3 закрывается. В результате на

затвор транзистора *VT*1 поступает нулевой потенциал. $U_{зи.VT1}$ этого транзистора становится отрицательным и транзистор *VT*1 открывается, на его стоке устанавливается положительное напряжение, что соответствует единичному состоянию триггера. Напряжение на стоке *VT*3 становится равным нулю.

Для записи нуля необходимо при нулевом напряжении на линии $\bar{A}$ подать напряжение $+U_n$ на разрядную линию *P*0, при этом через открытый транзистор *VT*6 положительное напряжение, попадая на затвор транзистора *VT*1, запирает его, что приводит к открыванию транзистора *VT*3. На стоке транзистора *VT*1 установится нулевой потенциал, а на стоке транзистора *VT*3 – потенциал напряжения питания.

Для считывания информации предварительно записанной в триггер необходимо подать нулевой потенциал только на линию $\bar{A}$. При этом, если был открыт транзистор *VT*1 (единичное состояние), то отрицательным напряжением $U_{зи.VT5}$ будет открыт транзистор *VT*5 и через него высокий потенциал поступит в разрядную линию *P*1. Если триггер находится в состоянии нуля, то откроется транзистор *VT*6 и высокий потенциал поступит в разрядную линию *P*0.

На рис. 2 приведена типичная структура микросхемы ОЗУ статического типа. Информация хранится в накопителе. Накопитель представляет собой матрицу, составленную из ячеек памяти рассмотренных выше. Для поиска требуемой ячейки памяти указываются строка и столбец, соответствующие положению ячейки памяти в накопителе.

Адрес ячейки памяти в виде двоичного числа принимается по шине адреса в регистр адреса. Число разрядов адреса связано с емкостью накопителя. Число строк и столбцов накопителя выбираются равными целой степени двух. Если число строк $N_{стр}=2^{n1}$ и число столбцов $N_{ст}=2^{n2}$, то общее число ячеек памяти (емкость накопителя) $N=N_{стр}?N_{ст}=2^{n1+}n^2=2^n$, где $n=n1+n2$ - число разрядов адреса, принимаемого в регистр адреса. Например, при емкости $N=2^{10}=1024$ число разрядов адреса $n=10$. При этом выбирается $n1=n2=5$. В этом случае число строк и число столбцов накопителя равно $2^{n1}=2^{n2}=32$. Требуемая размерность матрицы накопителя 32?32.

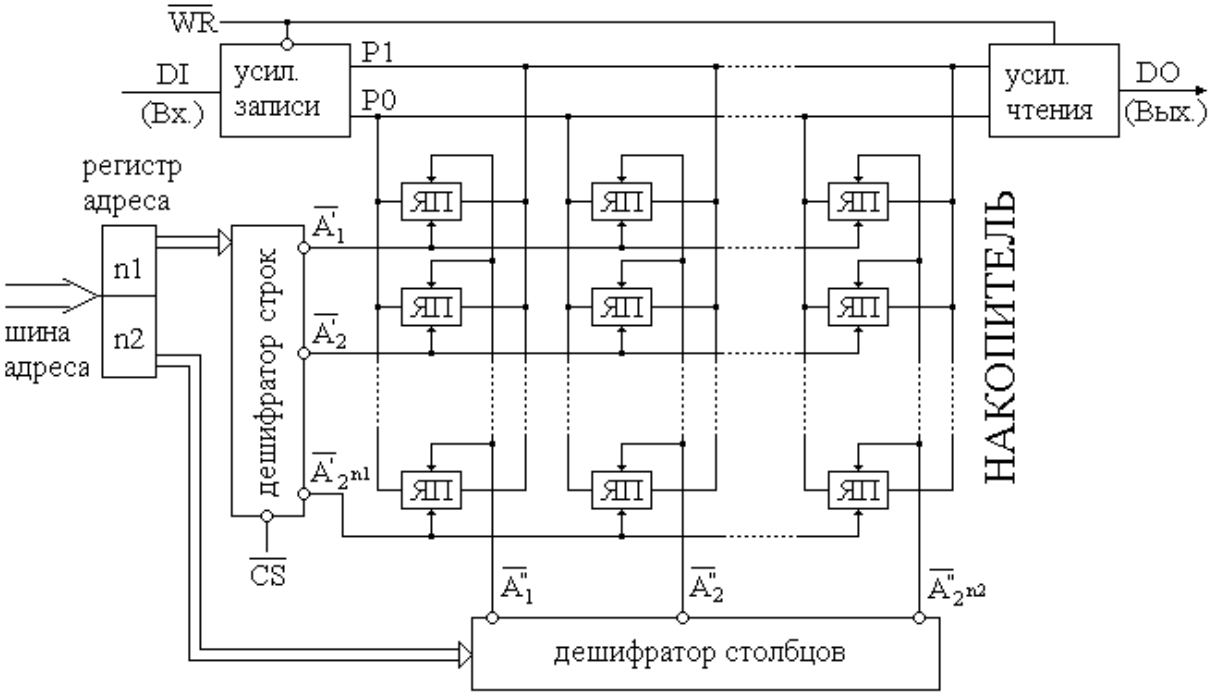


Рис. 2. Структура микросхемы ОЗУ статистического типа.

Разряды регистра адреса делятся на две группы. Одна группа в $n1$ разрядов определяет двоичный номер строки, в которой расположена ячейка памяти, другая группа в $n2$ разрядов определяет двоичный номер столбца, в котором находится ячейка памяти. Каждая группа разрядов адреса подается на соответствующий дешифратор строк или столбцов. При этом каждый из дешифраторов создает на одной из своих выходных цепей уровень логического нуля. На остальных выходах устанавливается уровень логической единицы. Ячейка памяти, оказавшаяся под воздействием двух логических нулей на соответствующих линиях $\overline{A}$ и $\overline{A'}$ одновременно, является выбранной. Этому соответствует подача логического нуля на линию $\overline{A}$ триггера ячейки памяти рассмотренной выше.

В режиме чтения содержимое ячейки памяти выдается на усилитель чтения и с него на выход микросхемы *DO*. При этом сигнал записи $\overline{WR}$ должен иметь пассивный единичный уровень. Режим записи устанавливается подачей активного нулевого уровня сигнала на вход записи $\overline{WR}$. Открывается усилитель записи и бит информации с входа данных *DI* поступает в выбранную ячейку памяти для запоминания, при этом усилитель чтения закрывается и данные на выход *DO* схемы не поступают.

Указанные процессы происходят, если на входе $\overline{CS}$ выбора микросхемы действует активный уровень логического нуля. При уровне логической единицы на этом входе на всех выходах дешифратора строк устанавливается уровень логической единицы, и ЗУ оказывается в режиме хранения. Последовательность подачи управляющих сигналов индивидуальна для каждого типа микросхемы памяти. Между тем, имеются общие закономерности. Рассмотрим последовательность подачи сигналов управления в режимах чтения и записи (рис. 3).

Первым как в режиме записи, так и в режиме чтения, на шину адреса должен выставляться адрес активизируемой ячейки памяти. Снимается адрес с шины после того, как запись в ячейку или чтение из ячейки завершено. Один из управляющих сигналов $\overline{WR}$ записи или $\overline{CS}$ выбора микросхемы или оба должны устанавливаться в активное состояние после установки адреса (интервалы времени t_1, t_2 и t_7, t_8) и сниматься до снятия адреса (интервалы времени t_3, t_4 и t_9, t_{10}). Тем самым обеспечивается высокоимпедансное состояние выводов *DO* и *DI* микросхемы, что исключает возможность ложного обмена информацией между микросхемами памяти и устройствами при смене адресов. В случае пассивного уровня сигнала $\overline{WR}$ отключается соответствующий буферный усилитель чтения или записи в каждом из своих режимах. В случае же пассивного уровня сигнала $\overline{CS}$ вырабатывается единичный уровень сигнала на линии $\overline{A}$ ячейки памяти, благодаря чему она отключается от линий $P0$ и $P1$ и хранит записанную информацию.

На рис. 3 приведены временные диаграммы работы ОЗУ в случае смены режима. Т.е. режим считывания осуществляется после режима записи и режим записи – после режима считывания. Поэтому происходит установка обоих сигналов $\overline{WR}$ и $\overline{CS}$. Обычно при нескольких режимах чтения подряд и при отсутствии обращения к микросхеме памяти сигнал $\overline{WR}$ имеет постоянное значение логической единицы. В этом случае активизация входа *DO* осуществляется только

нулевым уровнем сигнала на входе $\overline{CS}$. Первым определяется режим работы памяти, т.е. подается сигнал $\overline{WR}$. Управление выводами *DI* и *DO* осуществляется сигналом $\overline{CS}$, который подается внутри временного интервала действия сигнала $\overline{WR}$

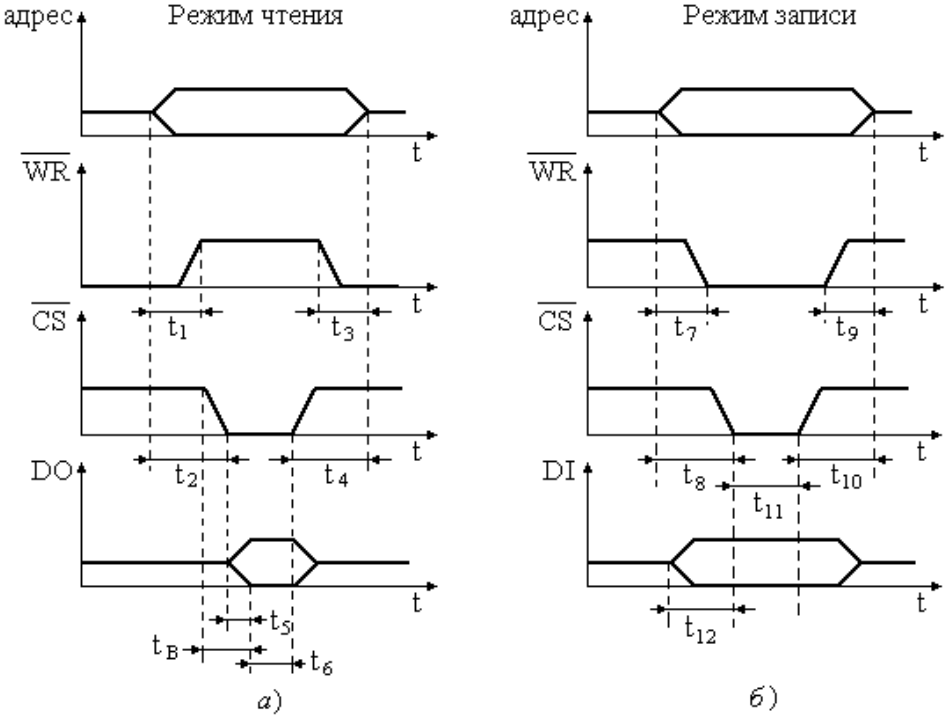


Рис. 3. Временная диаграмма работы ОЗУ статического типа.

Считывание информации из микросхемы памяти возможно только в интервал времени t_6, когда завершился процесс формирования данных на выходе *DO* (интервал времени t_5), и пока не снят сигнал $\overline{CS}$ выбора микросхемы. При этом время выборки $t_в$ характеризуется временным интервалом с момента выставления сигнала $\overline{CS}$ и до момента формирования информации на выходе *DO*. В режиме записи сигнал $\overline{CS}$ должен выставляться только тогда, когда записываемые данные готовы и поступили на вход *DI* (временной интервал t_{11}). Аналогично сами данные для записи должны быть подготовлены к моменту, когда выработается активный уровень сигнала $\overline{CS}$ (временной интервал t_{12}), и удержаны до окончания действия этого сигнала.

Микросхемы ОЗУ допускают наращивание емкости памяти как путем наращивания количества хранимых слов, так и путем наращивания разрядности этих слов. Наиболее простым в аппаратной реализации является второй способ – *наращивание разрядности хранимых слов*. Рассмотрим структуру построения памяти 1к?8 бит или 1024?8 бит. Хранимые слова в такой памяти будут восьмиразрядными, а адреса – десятиразрядными ($1024=2^{10}$). Для подобной организации необходимо параллельно к шине адреса подключить восемь микросхем ОЗУ 1к?1 (рис. 4). Толстой сплошной линией на электрических схемах принято изображать шины. Цифра или иной символ рядом с проводником указывает имя этого проводника в шине. Очевидно, что каждый проводник в шине должен иметь свое уникальное имя. Таким образом, осуществляется электрическое объединение всех одноименных выводов устройств, подключаемых к шине.

На все микросхемы $D1 - D8$ подается один и тот же адрес. Входы $\overline{WR}$ и $\overline{CS}$ микросхем объединяются. Каждая микросхема хранит свой разряд слова. Запись производится во все микросхемы одновременно. Точно также и чтение производится из всех микросхем одновременно. Очевидно, что организация такой памяти позволяет хранить 1024 байт информации.

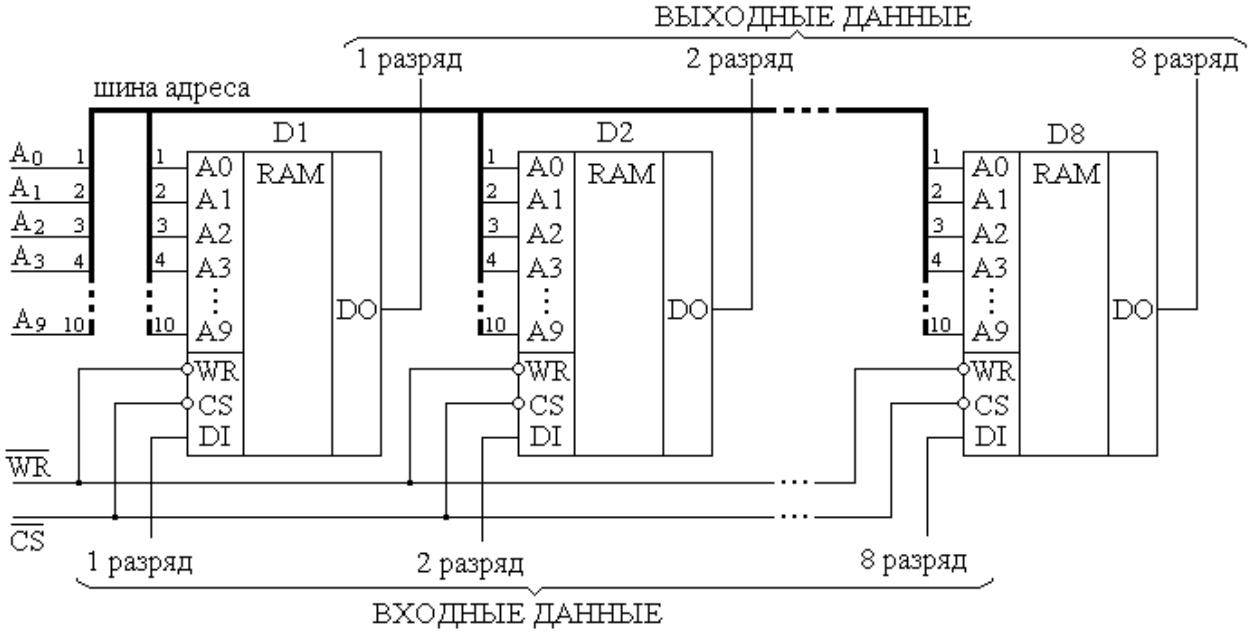


Рис. 6.4. Структура ОЗУ при наращивании разрядности хранимых слов.

Другой вариант организации структуры ОЗУ изображен на рис.5, который позволяет увеличивать объем памяти путем *наращивания количества хранимых слов*. Десять младших разрядов $A_0 - A_9$ адреса в рассматриваемой структуре также одновременно подаются на все восемь микросхем $D1 - D8$. При этом все входы *DI* микросхем объединены в один общий вход и все выходы *DO* объединены в один общий выход. Отсюда следует, что в определенный момент времени должна активизироваться только одна из восьми микросхем. Для этих целей используются три дополнительных адресных разряда $A_{10} - A_{12}$, которые подаются на дешифратор $D9$ выбора микросхемы памяти. С выхода дешифратора сигналы активизации подаются на раздельные входы $\overline{CS}$ каждой микросхемы памяти. Поскольку входы $\overline{CS}$ микросхем инверсные, то дешифратор также должен иметь инверсные выходы $Y_0 - Y_7$. Таким образом, емкость подобной структуры определяется как 8к?1=8?1024 бит или 1024 байт. Полученная емкость аналогична емкости структуры, изображенной на рис. 4, при этом для адресации к ней требуется большее количество адресных линий в шине адреса. Структура с наращиванием количества хранимых слов обладает двумя недостатками. В качестве первого можно отметить более сложную аппаратную реализацию, заключающуюся в введении дополнительных дешифраторов. Второй недостаток обусловлен меньшей производительностью памяти, поскольку обмен информацией осуществляется по одной паре выводов *DI* и *DO* вместо восьми пар выводов структуры с наращиванием разрядности хранимых слов. Однако структура, изображенная на рис. 6.5 имеет и преимущество, которое заключается в том, что она может использоваться в тех случаях, когда разрядность шины адреса превышает количество адресных входов отдельных микросхем.

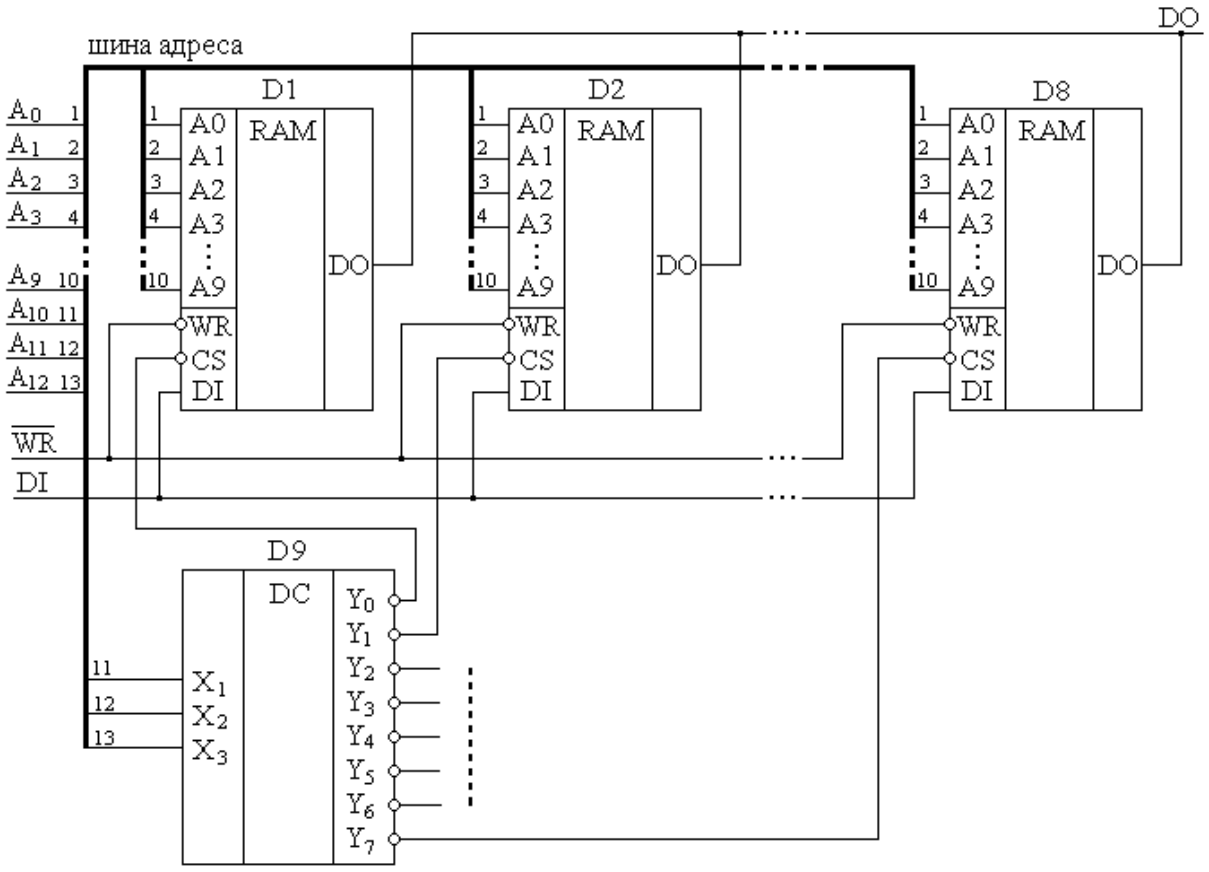


Рис. 6.5. Структура ОЗУ при наращивании количества хранимых слов.

На практике часто используется комбинированная структура, объединяющая наращивание как разрядности, так и количества хранимых слов. В этом случае формируется некоторое количество однотипных групп микросхем, объединенных в структуру с наращиванием разрядности слов. Далее эти группы объединяются в единую структуру с наращиванием количества хранимых слов. Разрядность слов комбинированной структуры определяется разрядностью слова одной группы микросхем, включенных по схеме наращивания разрядности.

Контрольные вопросы:

1. Организация работы памяти компьютера.
2. Статическая структура памяти.

Лабораторная работа №9

«Динамическая память. Принцип работы. Обобщенная структурная схема памяти»

Цель работы: Изучить применение и принцип работы динамической памяти.

Предварительная подготовка: изучить материал раздела «Архитектура и принципы работы основных логических блоков вычислительных систем (ВС)» (по конспекту).

Теоретическая часть:

Как уже отмечалось, информация в ячейке *динамического ОЗУ* представлена в виде наличия или отсутствия заряда на конденсаторе. Схема ячейки памяти *ЯП* динамического ЗУ на одном МОП–транзисторе с индуцируемым р-каналом представлена на рис. 1 (выделена пунктирной линией). На схеме также показаны общие элементы для *n*-ячеек одного столбца. Главное достоинство этой схемы - малая занимаемая площадь. Накопительный конденсатор *С*1 имеет МДП-структуру и изготавливается в едином технологическом цикле. Величина его емкости составляет сотые доли пикоФарад. Конденсатор *С*1 хранит информационный заряд. Транзистор *VT*1 выполняет роль переключателя, передающего заряд конденсатора в разрядную шину данных *ШД* при считывании, либо заряжающего конденсатор при записи. В режиме хранения на адресной линии $\overline{A_1}$ должен присутствовать потенциал логической единицы, под действием которого транзистор *VT*1 будет закрыт ($U_{зиVT1}$?0) и конденсатор *С*1 отключен от шины данных *ШД*. Включение конденсатора в шину данных осуществляется логическим нулем на линии $\overline{A_1}$. При этом на транзистор *VT*1 подается напряжение $U_{зи.VT1}$<0, что приводит к его открыванию.

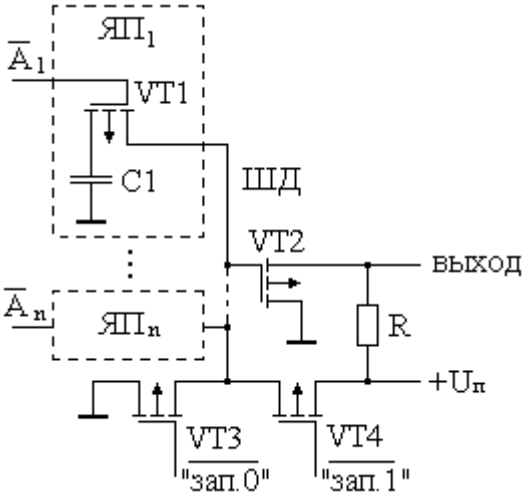


Рис. 1 Принципиальная схема ячейки ОЗУ динамического типа с элементами записи и усилителя считывания.

Поскольку шина данных Ш*Д* объединяет все ячейки памяти данного столбца, то она характеризуется большой длиной и ее собственная емкость имеет существенное значение. Поэтому при открывании транзистора *VT*1 потенциал шины данных изменяется незначительно. Чтобы установившийся потенциал на *ШД* однозначно идентифицировать с уровнем напряжения логического нуля или логической единицы, используется усилитель на базе транзистора *VT*2 и резистора

R. Непосредственно перед считыванием емкость шины данных подзаряжают подключением ее к источнику питания через транзистор *VT*4. Делается это для фиксации потенциала шины данных. При считывании информации происходит перераспределение заряда конденсатора и заряда шины данных, в результате чего информация, хранимая на конденсаторе *C*1, разрушается. Поэтому в цикле считывания необходимо произвести восстановление (регенерацию) заряда конденсатора. Для этих целей, а также для записи в ячейку памяти новых значений, используются транзисторы *VT*3 и *VT*4, которые подключают шину данных либо к источнику питания, либо к нулевому общему потенциалу. Для записи в ячейку памяти логической единицы необходимо открыть транзистор VT4 нулевым значением управляющего сигнала «$\overline{зап\_1}$» и подключить к шине данных источник питания. Для записи логического нуля необходимо нулевым потенциалом на входе «$\overline{зап\_0}$» открыть транзистор VT3. Одновременная подача логических нулей на входы «$\overline{зап\_1}$» и «$\overline{зап\_0}$» не допускается, так как это вызовет короткое замыкание источника питания на общий провод заземления.

На рис. 2 показан пример структуры микросхемы динамического ОЗУ емкостью 64кбит. Данные в этой микросхеме памяти представлены как 64к отдельных бит, т.е. формат памяти 64к?1. Ввод и вывод осуществляется раздельно, для чего предусмотрена пара выводов *DI* (вход) и *DO* (выход). Для ввода адреса имеется восемь контактов *A*0 — *A*7. Адресация к 64к ячейкам памяти осуществляется шестнадцатиразрядными адресами A_0 — A_{15}. Причем сначала на входы *A*0-*A*7 подаются восемь младших разрядов $A_0 - A_7$ адреса, а затем – восемь старших разрядов $A_8 - A_{15}$. Восемь младших разрядов адреса фиксируются в регистре адреса строки подачей сигнала $\overline{RAS}$ (сигнал выборки строки). Восемь старших разрядов адреса фиксируются в регистре адреса столбца подачей сигнала $\overline{CAS}$ (сигнал выборки столбца). Такой режим передачи кода адреса называется мультиплексированным по времени. Мультиплексирование позволяет сократить количество выводов микросхемы. Ячейки памяти расположены в виде матрицы из 128 строк и 512 столбцов. Дешифратором строк вырабатывается адресный сигнал выборки $\overline{A_i}$ ячеек памяти *i*-ой строки, т.е. выбирается одна из 128 строк. Обращение к строке вызывает подключение 512 ячеек памяти через соответствующие разрядные шины данных *ШД* этой строки к усилителям считывания (по одному на столбец). При этом автоматически происходит подзаряд запоминающих конденсаторов всех ячеек памяти выбранной строки до исходного уровня за счет передачи усиленного сигнала по цепи обратной связи. Этот процесс называется *регенерацией памяти*. Дешифратор столбцов выбирает один из 512 усилителей считывания. Бит, выбранный в режиме считывания, выдается на линию *DO*. Если одновременно с сигналом $\overline{CAS}$ при предварительно установленном сигнале $\overline{RAS}$ действует сигнал записи $\overline{WR}$, то бит с входа *DI* будет записан в выбранную ячейку памяти, при этом выход *DO* микросхемы остается в отключенном состоянии в течение всего цикла записи.

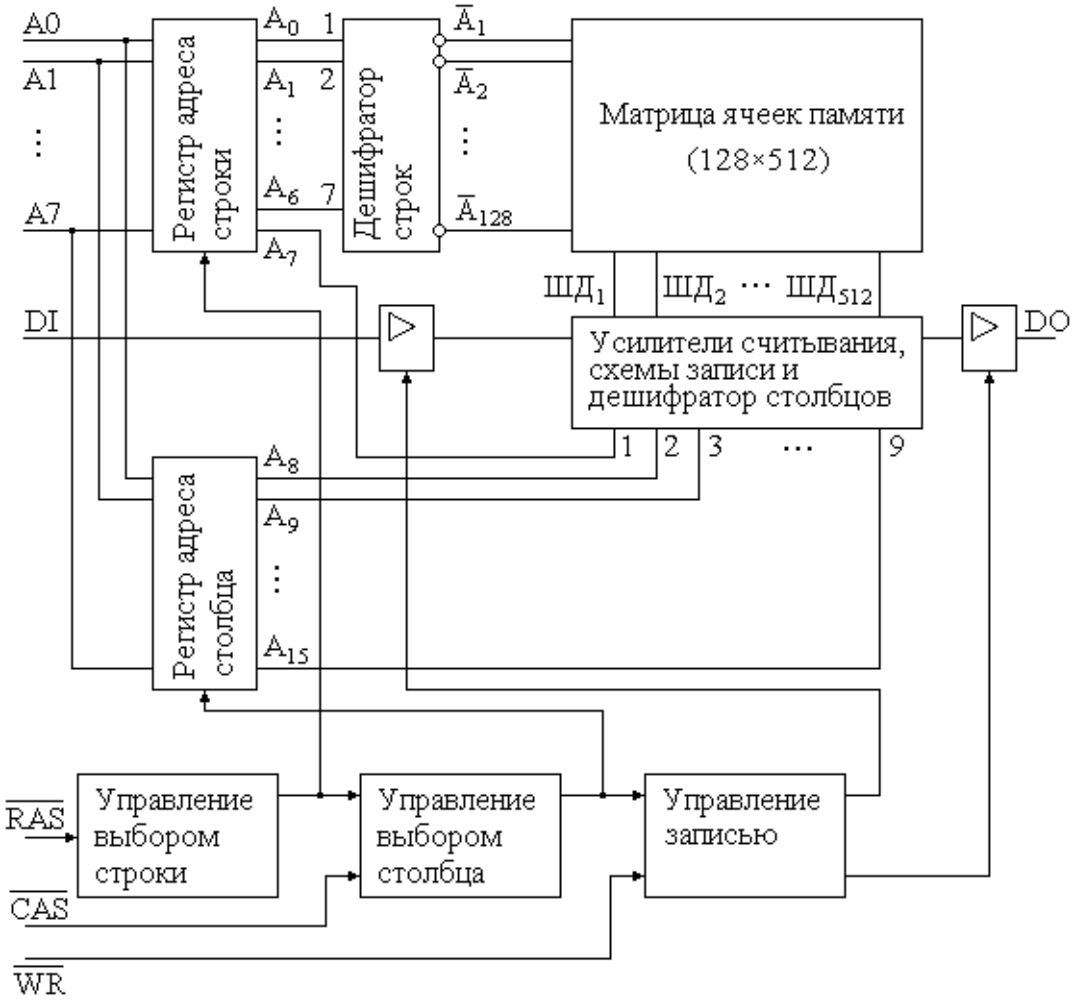


Рис. 2 Структура микросхемы ОЗУ динамического типа.

На рис.3 представлены временные диаграммы, поясняющие работу динамического ОЗУ. В режиме считывания (рис.3, *а*) на адресные входы микросхемы подаются восемь младших разрядов A_0 – A_7 адреса, после чего вырабатывается сигнал $\overline{RAS}$, при этом производится выбор строки матрицы в соответствии с поступившим адресом. У всех ячеек памяти выбранной строки регенерируется заряд конденсаторов. Далее производится подача на адресные входы микросхемы восьми старших разрядов адреса, после чего вырабатывается сигнал $\overline{CAS}$. Этим сигналом выбирается нужная ячейка памяти из выбранной строки и считанный бит информации поступает на выход микросхемы *DO*. В режиме считывания промежуток времени между подачей сигнала $\overline{RAS}$ и появлением данных на выходе *DO* называется временем выборки $t_в$.

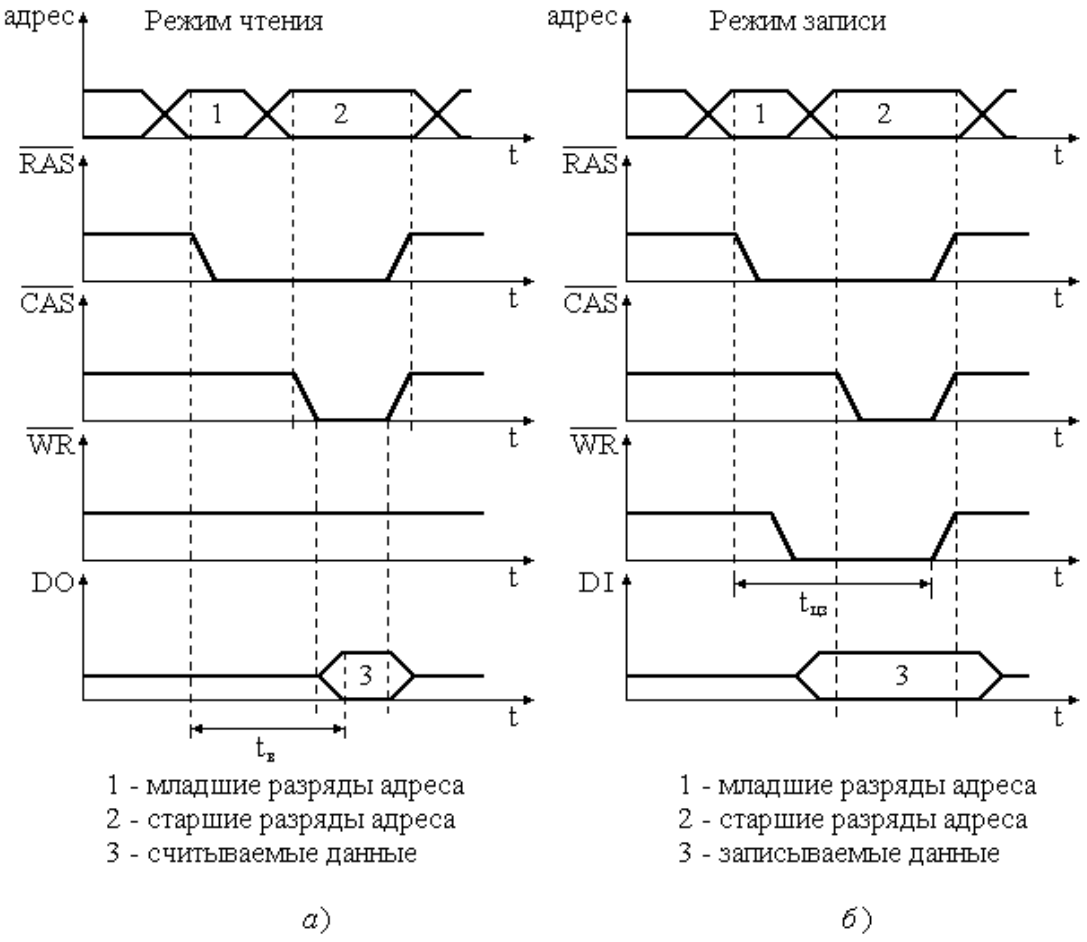


Рис. 3 Временная диаграмма работы ОЗУ динамического типа.

В режиме записи (рис. 6.8,*б*) за время цикла записи $t_{цз}$ принимается интервал времени между появлением сигнала $\overline{RAS}$ и окончанием сигнала $\overline{WR}$. В момент появления сигнала $\overline{CAS}$ записываемые данные уже должны поступать на вход *DI*. Сигнал $\overline{WR}$ обычно вырабатывается раньше сигнала $\overline{CAS}$.

Для каждого типа микросхем динамических ОЗУ в справочниках приводятся временные параметры, регламентирующие длительность управляющих сигналов, подаваемых на микросхему, а также порядок их взаимного следования.

Заряд конденсатора динамического ОЗУ со временем уменьшается вследствие утечки, поэтому для сохранения содержимого памяти процесс регенерации каждой ячейки памяти должен производится через определенное время. Следовательно, для предотвращения разряда запоминающих конденсаторов необходимо обращаться к каждой строке матрицы через определенное время. При обычном режиме работы ОЗУ это условие не соблюдается, так как обращение к одним ячейкам происходит часто, а к другим очень редко. Поэтому необходим специальный блок, ответственный за регенерацию памяти. Этот блок должен при отсутствии обращений к ОЗУ со стороны внешних устройств циклически формировать на адресных входах *A*0-*A*6 значения всех возможных адресов, сопровождая каждый из них управляющим сигналом $\overline{RAS}$, т.е. производить циклическое обращение ко всем 128 строкам матрицы ячеек памяти. Регенерацию необходимо проводить и в те моменты времени, когда ОЗУ используется устройствами, приостанавливая на время регенерации взаимодействие ОЗУ с этими устройствами, т.е. путем перевода этих устройств в режим ожидания.

Из изложенного выше следует, что использование динамического ОЗУ требует довольно сложной схемы управления. Если учесть, что обращение к ОЗУ со стороны устройств, с которыми оно работает, и обращение со стороны схемы регенерации не зависят друг от друга, следовательно, могут возникать одновременно, то необходима схема, обеспечивающая упорядоченность этих обращений. Для этих целей существуют схемы, управляющие работой динамических ОЗУ. Это так называемые контроллеры динамического ОЗУ, реализованные на одном кристалле. Их использование позволяет значительно упростить построение памяти на динамических ОЗУ.

Лидером в производстве микросхем динамического ОЗУ на сегодняшний день является фирма Samsung. Емкость одной микросхемы DRAM достигает значения 128 Мбайт и более. Кроме того, этой фирмой предлагается ряд передовых идей по обеспечению наибольшего быстродействия. Например, операции чтения и записи выполняются дважды за один такт – по переднему и заднему фронтам тактового импульса. Фирмой Mitsubishi предложена концепция встраивания в микросхемы динамической памяти статической кэш-памяти небольшого объема (Cashed DRAM), в которой хранятся наиболее часто запрашиваемые данные.

Контрольные вопросы:

3. Организация работы памяти компьютера.
4. Динамическая структура памяти.

Лабораторная работа №11

«Принцип работы кэш-памяти»

Цель работы: Изучить применение и принцип работы кэш-памяти.

Предварительная подготовка: изучить материал раздела «Архитектура и принципы работы основных логических блоков вычислительных систем (ВС)» (по конспекту).

Теоретическая часть:

Виды и принципы работы кэш-памяти

Одним из важнейших устройств компьютера является память, или запоминающее устройство (ОЗУ). Однако под это определение попадает как собственно память, так и внешние запоминающие устройства (типа накопителей на жестких и гибких дисках, магнитной ленты, CD-ROM), которые лучше отнести к устройствам ввода/вывода информации. Таким образом под компьютерной памятью в дальнейшем будет пониматься только "внутренняя память компьютера: ОЗУ, ПЗУ, кэш память и флэш-память".

Однако ОЗУ работает намного медленнее процессора. Другое дело - кэш. Очень быстрая и очень дорогая память. Но именно из-за дороговизны в компьютерной системе объем кэш составляет всего 10-15% от объема обычной ОЗУ. А также используются специальные алгоритмы работы материнской платы и процессора, обеспечивающие максимальную производительность при наличии встроенной кэш-памяти.

1. Общие понятие о кэшировании и кэш-памяти

Принцип работы кэш-памяти заключается в следующем. Процессор редко использует весь объем ОЗУ практически одновременно. Скакать из одного угла памяти в другой, периодически пошвыриваясь по всему ее объему – это не лучший способ использования ресурсов компьютера. Зачастую все обращения процессора к памяти сосредоточены в небольшой области (как показывает статистика – 5-10% от общего объема). Если данные из этой области как либо аппаратно скопировать в кэш, а затем постоянно сверять кэш и ОЗУ на предмет целостности данных, то можно обеспечить режим работы, при котором процессор будет обращаться только к кэш-памяти, тратя на это значительно меньше ресурсов и времени, чем обычно.

Естественно, что весь объем ОЗУ скопировать в кэш нет возможности – такой объем кэш-памяти по цене сравнивается со стоимостью компьютера, а смысла уменьшать объем ОЗУ тоже нет. Было решено реализовать алгоритм работы процессора, кэш-памяти и ОЗУ аппаратно, чтобы не тратить ресурсы процессора.

Принцип заключается в следующем: когда процессор обращается к определенной ячейке памяти, сегмент памяти определенного объема (этот объем называется объемом страницы кэш) копируется в кэш полностью. Если процессор дальше не совершит глобальный скачек на другой, далекий от текущего, адрес памяти, то дальнейшая работа процессора будет происходить напрямую с кэш, минуя ОЗУ, а контроллер кэш-памяти в промежутках, когда процессор занят вычислениями (либо параллельно с работой процессора) будет восстанавливать верные данные в ОЗУ либо в кэш (в случае наличия устройств, напрямую работающих с памятью). Естественно, чем больше будет страниц и чем больше будет их объем – тем выше будет скорость работы процессора.

Кэш-память можно использовать там, где существует проблема быстродействия, но есть возможность упорядочить данные. К таким применениям относят:

• аппаратное кэширование жестких дисков (кэш-память устанавливается непосредственно на жестком диске либо на специальном контроллере);

• программное кэширование CD-ROM, а также прочих устройств хранения информации (программно – при помощи операционной системы, аппаратно – на самом устройстве либо на контроллере).

И не только: сегодня зачастую даже самое простейшее устройство обладает своей памятью, работающей быстрее, чем само устройство. К таким относят принтеры, сканеры, модемы и т.д.

2. Внутренний кэш процессора

С кэшированием связаны новые функции процессоров, биты регистров и внешние сигналы.

Процессоры 486 и Pentium имеют внутренний кэш первого уровня, в Pentium Pro и Pentium II имеется и вторичный кэш. Процессоры могут иметь как единый кэш инструкций и данных, так и общий. Выделенный кэш инструкций обычно используется только для чтения. Для внутреннего кэша обычно используется наборно-ассоциативная архитектура.

Строки в кэш-памяти выделяются только при чтении, политика записи первых процессоров 486 – только Write Through (сквозная запись) – полностью программно-прозрачная. Более поздние модификации 486-го и все старшие процессоры позволяют переключаться на политику Write Back (обратная запись).

Работу внутренней кэш-памяти характеризуют следующие процессы: обслуживание запросов процессора на обращение к памяти, выделение и замещение строк для кэширования областей физической памяти, обеспечение согласованности данных внутреннего кэша и оперативной памяти, управление кэшированием.

Любой внутренний запрос процессора на обращение к памяти направляется на внутренний кэш. Теги четырех строк набора, который обслуживает данный адрес, сравниваются со старшими битами запрошенного физического адреса. Если адресуемая область представлена в строке кэш-памяти (случая попадания –cache hit), запрос на чтение обслуживается только кэш-памятью, не выходя на внешнюю шину. Запрос на запись модифицирует данную строку, и в зависимости от политики записи либо сразу выходит на внешнюю шину (при сквозной записи), либо несколько позже (при использовании алгоритма обратной записи).

В случае промаха (Cache Miss) запрос на запись направляется только на внешнюю шину, а запрос на чтение обслуживается сложнее. Если этот зарос относится к кэшируемой области памяти, выполняется цикл заполнения целой строки кэша – все 16 байт (32 для Pentium) читаются из оперативной памяти и помещаются в одну из строк кэша, обслуживающего данный адрес. Если затребованные данные не укладываются в одной строке, заполняется и соседняя. Заполнение строки процессор старается выполнить самым быстрым способом – пакетным циклом с 32-битными передачами (64-битными для Pentium и старше).

Внутренний запрос процессора на данные удовлетворяется сразу, как только затребованные данные считываются из ОЗУ – заполнение строки до конца может происходить параллельно с обработкой полученных данных. Если в наборе, который обслуживает данный адрес памяти, имеется свободная строка (с нулевым битом достоверности), заполнена будет она и для нее установится бит достоверности. Если свободных строк в наборе нет, будет замещена строка, к которой дольше всех не было обращений. Выбор строки для замещения выполняется на основе анализа бит LRU (Least Recently Used) по алгоритму “псевдо-LRU”. Эти биты (по три на каждый из наборов)

модифицируются при каждом обращении к строке данного набора (кэш-попадании или замещении).

Выделение и замещение строк выполнятся только кэш-промахов чтения, при промахах записи заполнение строк не производится. Если затребованная область памяти присутствует в строке внутреннего кэша, то он обслужит этот запрос. Управлять кэшированием можно только на этапе заполнения строк; кроме того, существует возможность их аннулирования – объявления недостоверными и очистка всей кэш-памяти.

Очистка внутренней кэш-памяти при сквозной записи (обнуление бит достоверности всех строк) осуществляется внешним сигналом FLUSH# за один такт системной шины (и, конечно же, по сигналу RESET). Кроме того, имеются инструкции аннулирования INVD и WBINVD. Инструкция INVD аннулирует строки внутреннего кэша без выгрузки модифицированных строк, поэтому ее неосторожное использование при включенной политике обратной записи может привести к нарушению целостности данных в иерархической памяти. Инструкция WBINVD предварительно выгружает модифицированные строки в основную память (при сквозной записи ее действие совпадает с INVD). При обратной записи очистка кэша подразумевает и выгрузку всех модифицированных строк в основную память. Для этого, естественно, может потребоваться и значительное число тактов системной шины, необходимых для проведения всех операций записи.

Аннулирование строк выполняется внешними схемами – оно необходимо в системах, у которых в оперативную память запись может производить не только один процессор, а и другие контроллеры шины – процессор или периферийные контроллеры. В этом случае требуются специальные средства для поддержания согласованности данных во всех ступенях памяти – в первичной и вторичной кэш-памяти и динамического ОЗУ. Если внешний (по отношению к рассматриваемому процессору) контроллер выполняет запись в память, процессору должен быть подан сигнал AHOLD. По этому сигналу процессор немедленно отдает управление шиной адреса A[31:4], на которой внешним контроллером устанавливается адрес памяти, сопровождаемый стробом EADS#. Если адресованная память присутствует в первичном кэше, процессор аннулирует строку – сбрасывает бит достоверности этой строки (она освобождается). Аннулирование строки процессор выполняет в любом состоянии.

Управление заполнением кэша возможно и на аппаратном и на программном уровнях. Процессор позволяет кэшировать любую область физической памяти, но внешние схемы могут запрещать процессору кэшировать определенные области памяти. Это делается по различным причинам, зачастую связанным с определенными условиями создания компьютерной системы.

3. Внешний кэш процессора

В отличие от внутренней кэш-памяти, внешняя больше напоминает обычную память. Однако алгоритм работы с ней практически такой же.

Внешняя кэш-память состоит из памяти данных, построенная на микросхемах SRAM, и контроллера кэша. В кэш-памяти хранится информация, копируемая из основной оперативной памяти. Каждый раз при обращении микропроцессора к памяти контроллер кэш-памяти проверяет наличие данных в кэше. Если эти данные в кэше есть (“попадание”), то микропроцессор получает данные из кэша. Если этих данных нет (“промах”), выполняется обычный цикл обращения к оперативной памяти DRAM.

Основным фактором, определяющим вероятность попадания, является емкость кэш-памяти. Как правило, при объеме кэша в 2 Кбайта вероятность попадания составляет от 50 до 60%. Поскольку размер кэш-памяти на современных компьютерах превышает 256

Кбайт, то вероятность попадания будет выше 90% (для компьютеров с объемом памяти ~ 256 Мбайт.)

Особенность контроллера кэш-памяти – обеспечение возможности параллельной работы микропроцессора с кэш-памятью и периферийных устройств с оперативной памятью в режиме прямого доступа. При записи данных по адресам, находящихся в кэше, контроллер ликвидирует копии этих блоков в кэше. Всю работу по синхронизации данных в DRAM и кэше берет на себя этот контроллер.

На многих материнских платах можно выбирать между одноуровневой или многоуровневой системами организации памяти. По умолчанию устанавливается ражим многоуровневой памяти. Если Вы установите режим одноуровневой памяти, то кэш-память SRAM просто добавляется к адресному пространству основной оперативной памяти. Одноуровневую память лучше использовать, когда внутренний кэш процессора по объему превосходит емкость кэш-памяти на материнской плате.

Контрольные вопросы:

1. Кэш-память. Виды кэш-памяти.
2. Назначение кэш-памяти.
3. Принцип работы.

Лабораторная работа №12

«Архитектура системной платы. Внутренние интерфейсы системной платы»

Цель работы: Изучение архитектуры системной платы. Изучение внутренних интерфейсов системной платы.

Предварительная подготовка: изучить материал раздела «Архитектура и принципы работы основных логических блоков вычислительных систем (ВС)» (по конспекту).

Задание 1: Идентифицируйте внутренние интерфейсы системной платы.

Задание 2: Дайте сравнительную характеристику внутренних интерфейсов целевой системной платы.

Задание 3: Рассмотреть представленную материнскую плату и указать основные компоненты, а также их назначение.

Методические указания:

Системная плата – печатная плата, соединяющая все узлы компьютера в одно устройство. Кроме термина "системная плата", используется название "материнская плата".

Интерфейс – это совокупность средств сопряжения и связи, обеспечивающая эффективное взаимодействие систем или их частей. В интерфейсе обычно предусмотрены вопросы сопряжения на механическом (число проводов, элементы связи, типы соединений, разъемы, номера контактов и т.п.) и логическом (сигналы, их длительность, полярность, частоты и амплитуда, протоколы взаимодействия) уровнях.

Внутренний интерфейс – это система связи и сопряжения узлов и блоков компьютера между собой. Представляет собой совокупность электрических линий связи, схем сопряжения с компонентами компьютера, протоколов (алгоритмов) передачи и преобразования сигналов.

В современных компьютерах в качестве системного интерфейса обычно используется системная шина.

Шина (bus) – это совокупность линий связи, по которым информация передается одновременно. Под основной или системной шиной понимается шина между процессором и подсистемой памяти. Шины характеризуются разрядностью и частотой.

Разрядность или ширина шины (bus width) – количество линий связи в шине, т.е. количество битов, которое может быть передано по шине одновременно.

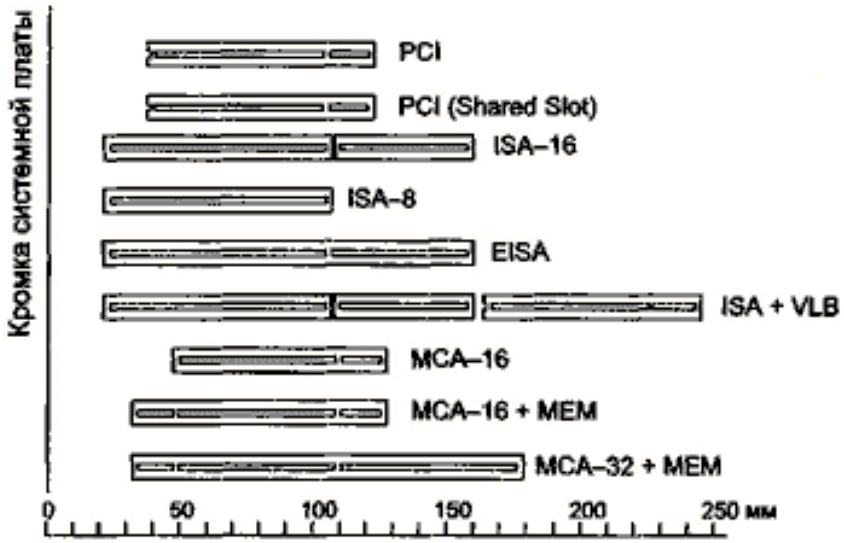
Тактовая частота шины (bus frequency) – частота с которой передаются последовательные биты информации по линиям связи.

В качестве системной шины в ПК могут использоваться шины расширений и локальные шины.

Шины расширений – шины общего назначения, позволяющие подключать большое количество самых разнообразных устройств.

Локальные шины специализируются на обслуживании небольшого количества устройств определенного класса.

ШИНЫ РАСШИРЕНИЙ



Шина расширения ISA (Industry Standard Architecture) – основная шина на устаревших материнских платах. Служит для подключения витдеокарт, модемов, звуковых карт и т.д. Конструктивно представляет собой разъем состоящий из двух частей – 62-контактного и примыкающего к нему 36-контактного сегментов. Допускает подключение до 6 устройств. Пропускная способность шины до 16 Мбайт/с. Рабочая частота до 8 МГц. Представлена в двух версиях PC/XT и PC/AT.

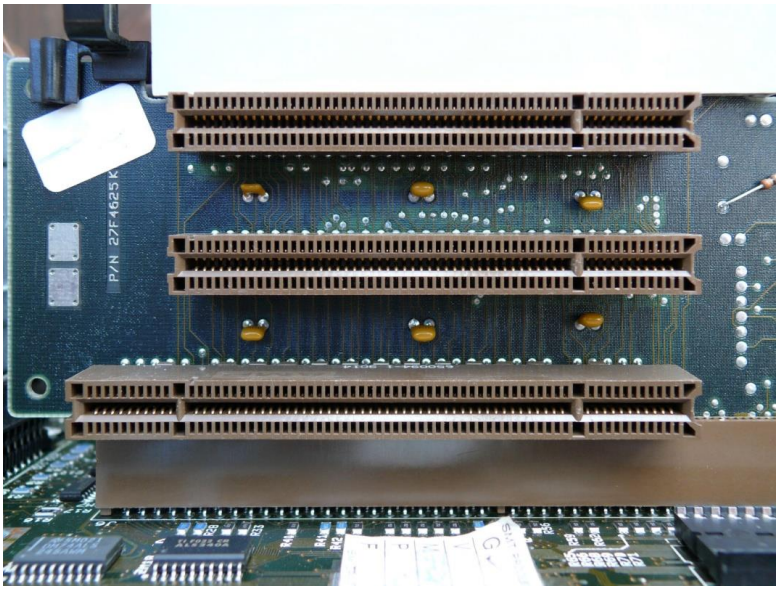
Шина PC/XT - 8-разрядная шина данных и 20-разрядная шина адреса, имеет 4 линии аппаратных прерываний и 4 канала для прямого доступа в память. Шина адреса ограничивает адресное пространство микропроцессора величиной 1 Мбайт. Тактовая частота 4,77 МГц.

Шина PC/AT – 16-разрядная шина данных и 24-разрядная шина адреса. Имеет 7 линий для аппаратных прерываний и 4 канала прямого доступа в память. Тактовая частота до 16 МГц. Адресное пространство шины до 16 Мбайт.

Шина EISA (Extended Industry Standard Architecture) – 32-разрядная адресная шина данных и 32-разрядная шина адреса. Адресное пространство шины 4 Гбайт. Тактовая частота 8-33 МГц. Пропускная способность 33 Мбайт/с. Теоретически может подключаться до 15 устройств. Шина поддерживает многопроцессорную архитектуру. Шина весьма дорога и применяется в скоростных ПК, сетевых серверах и рабочих станциях. Внешне слоты шины имеют такой же вид, как и ISA, и в них могут вставляться платы ISA, но в глубине разъема находятся дополнительные ряды контактов EISA, а платы EISA имеют более высокую ножевую часть разъема с дополнительными рядами контактов.

Шина MCA (MicroChannel Architecture) - микроканальная архитектура - была введена в пику конкурентам фирмой IBM для своих компьютеров PS/2 начиная с модели 50 в 1987 году. Обеспечивает быстрый обмен данными между отдельными

устройствами, в частности с оперативной памятью. Шина MCA абсолютно несовместима с ISA/EISA и другими адаптерами. Состав управляющих сигналов, протокол и архитектура ориентированы на асинхронное функционирование шины и процессора, что снимает проблемы согласования скоростей процессора и периферийных устройств. Адаптеры MCA широко используют Bus-Mastering, все запросы идут через устройство CACP (Central Arbitration Control Point). Архитектура позволяет эффективно и автоматически конфигурировать все устройства программным путем (в MCA PS/2 нет ни одного переключателя). При всей прогрессивности архитектуры (относительно ISA) шина MCA не пользуется популярностью из-за узости круга производителей MCA-устройств и полной их несовместимости с массовыми ISA-системами. Однако MCA еще находит применение в мощных файл-серверах, где требуется обеспечение высоконадежного производительного ввода-вывода.



ЛОКАЛЬНЫЕ ШИНЫ.

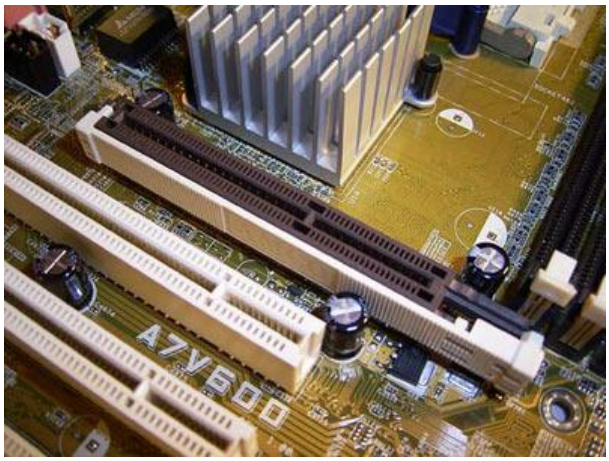
В настоящее время существует три основных стандарта универсальных локальных шин: VLB, PCI, AGP.

Шина VLB (VL-bus, VESA Local Bus) является расширением внутренней шины микропроцессора для связи с видеоадаптером или жестким диском, платами мкльтимедиа, сетевым адаптером. Разрядность шины для данных 32 бита, для адреса 30 бит.

Шина не адаптирована для процессоров класса Pentium. Имеется жесткая зависимость от тактовой частоты процессора. Шина позволяет только 4 устройства, при этом отсутствует арбитраж шины, т.е. могут быть конфликты между подключаемыми устройствами.

Шина PCI (Peripheral Component Interconnect, соединение внешних компонентов) – с помощью этого интерфейса к материнской плате подключаются видеокарты, звуковые карты, модемы, контроллеры SCSI и др. устройства. Конструктивно разъем шины на системной плате состоит из двух следующих подряд секций по 64 контакта (каждая со своим ключом). Разрядность шины – 32 разряда данных и 32 разряда адреса с возможностью расширения до 64 бит. Тактовая частота шины 33 МГц. Допускает подключение до 10 устройств. При наличии шины PCI шины расширения подключаются не непосредственно к микропроцессору, а к самой шине PCI. Благодаря такому решению шина является независимой от процессора и может работать параллельно с шиной процессора, не обращаясь к ней с запросами. Таким образом, загрузка шины процессора существенно снижается.

Шина AGP (Accelerated Graphics Port – ускоренный графический порт) – интерфейс для подключения видеоадаптера к отдельной магистрали, имеющей выход непосредственно на системную память. Шина может работать с частотой до 133 МГц и обеспечивает высочайшую скорость передачи графических данных. По сравнению с шиной PCI, в шине AGP устранена мультиплексность линий адреса и данных (в PCI для удешевления конструкции адрес и данные передаются по одним и тем же линиям) и усилена конвейеризация операций чтения-записи, что позволяет устранить влияние задержек в модулях памяти на скорость выполнения этих операций.



К шинам подключаются электронные платы (контроллеры). Каждый контроллер может быть подключен только к той шине на которую он рассчитан. Поэтому разъемы разных шин сделаны разными.

Контрольные вопросы:

1. Интерфейс. Типы интерфейсов.
2. Шина. Характеристики шин.

Лабораторная работа №13

«Интерфейсы периферийных устройств»

Цель работы: Изучение интерфейсов периферийных устройств. Изучение особенностей работы параллельных и последовательных портов

Предварительная подготовка: изучить материал раздела «Архитектура и принципы работы основных логических блоков вычислительных систем (ВС)» (по конспекту).

Задание 1.

Дать сравнительную характеристику периферийных устройств целевого компьютера. Определить их достоинства и недостатки.

Задание 2.

Указать название разъёмов и для каких устройств они применяются.

Задание 3.

Определить внешние интерфейсы целевого компьютера. Подключить к целевому компьютеру принтер, монитор, сканер, мышь, клавиатуру, колонки.

Методические указания:

Периферийные шины используются в основном для внешних запоминающих устройств.

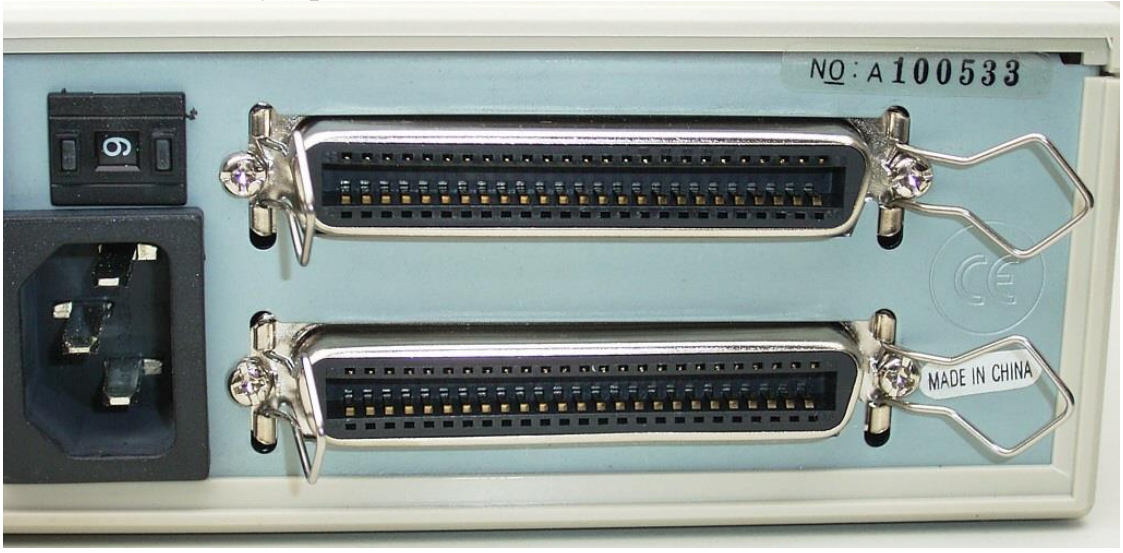
Интерфейс IDE (Integrated Drive Electronics) – интерфейс устройств со встроенным контроллером. Поддерживает несколько способов обмена. Первый способ производит обмен данными через регистры процессора под его непосредственным управлением. Следствием этого является высокая загрузка процессора при операциях ввода/вывода. Вторым способом является использование режима прямого доступа к памяти, при котором контроллер интерфейса IDE и контроллер прямого доступа к памяти материнской платы пересылают данные между диском и оперативной памятью, не загружая центральный процессор. В целях развития возможностей интерфейса IDE была предложена его расширенная спецификация EIDE (синонимы ATA, ATA-2). Она поддерживает накопители емкостью свыше 504 Мбайт, поддерживает несколько накопителей IDE и позволяет подключать к одному контроллеру до четырех устройств, а также поддерживает периферийные устройства, отличные от жестких дисков. Расширение спецификации IDE для поддержки иных типов накопителей с интерфейсом IDE называют также ATAPI.

SATA (Serial ATA) – последовательный интерфейс обмена данными с накопителями информации. Для подключения используется 8-pin разъем. SATA является развитием параллельного интерфейса ATA (IDE), который после появления SATA был переименован в PATA (Parallel ATA).Стандарт SATA (SATA150) обеспечивал пропускную способность равную 150 МБ/с (1,2 Гбит/с).

SATA 2 (SATA300). Стандарт SATA 2 увеличивал пропускную способность в двое, до 300 МБ/с (2,4 Гбит/с), и позволяет работать на частоте 3 ГГц. Стандартны SATA и SATA 2 совместимы между собой, однако для некоторых моделей необходимо вручную устанавливать режимы, переставляя джамперы.

SATA 3, хотя про требованию спецификаций правильно называть **SATA 6Gb/s**. Этот стандарт в двое увеличил скорость передачи данных до 6 Гбит/с (600 МБ/с). Также к положительным нововведениям относится функция программного управления NCQ и команды для непрерывной передачи данных для процесса с высоким приоритетом.

Интерфейс SCSI (Small Computer System Interface) - является стандартным интерфейсом для подключения приводов компакт-дисков, звуковых плат и внешних устройств массовой памяти. Спецификацией SCSI предусматривается параллельная передача данных по 8, 16 или 32 линиям данных. Структура SCSI, по существу, является магистральной, хотя устройства включаются в нее по принципу последовательной цепочки. Каждое SCSI-устройство имеет два разъема – один входной, а другой выходной. Все устройства объединяются в последовательную цепочку, один конец которой подключается к контроллеру интерфейса. Все устройства работают независимо и могут обмениваться данными как с компьютером, так и друг с другом. К шине SCSI можно подключить до 8 устройств, включая основной контроллер SCSI (хост-адаптер). Контроллер SCSI является, по сути, самостоятельным процессором и имеет свою собственную BIOS. К шине Wide SCSI подключается до 15 устройств.



Порт (персонального) компьютера предназначен для обмена информацией между устройствами, подключенными к шине внутри компьютера и внешним устройством.

Для связи с периферийными устройствами к шине компьютера подключены одна или несколько микросхем контроллера ввода-вывода.

Последовательный порт стандарта RS-232-C. Является стандартом для соединения ЭВМ с различными последовательными внешними устройствами. В операционных системах каждому порту RS-232 присваивается логическое имя COM1-COM4.

Параллельный порт используется для одновременной передачи 8 битов информации. В компьютерах этот порт используется главным образом для подключения принтера, графопостроителей и других устройств. Параллельные порты обозначаются LPT1-LPT4.

Интерфейс USB (Universal Serial Bus) – универсальная последовательная шина призвана заменить устаревшие последовательный (COM-порт) и параллельный (LTP-порт) порты. Шина USB допускает подключение новых устройств без выключения компьютера. Шина сама определяет, что именно подключили к компьютеру, какой драйвер и ресурсы понадобятся устройству, после чего выделяет их без вмешательства пользователя. Шина USB позволяет подключить до 127 устройств.

IEEE 1394 (Institute of Electrical and Electronic Engineers 1394 – стандарт Института инженеров по электротехнике и электронику 1394) - последовательный интерфейс, предназначенный для подключения внутренних компонентов и внешних устройств. Цифровой последовательный интерфейс **IEEE 1394** характеризуется высокой надежностью и качеством передачи данных, его протокол поддерживает гарантированную передачу критичной по времени информации, обеспечивая прохождение видео- и аудиосигналов в реальном масштабе времени без заметных искажений. При помощи шины **IEEE 1394** можно подключить до 63 устройств и практически в любой конфигурации, чем она выгодно отличается от трудноконфигурируемых шин SCSI. Этот интерфейс используется для подключения жестких дисков, дисководов CD-ROM и DVD-ROM, а также высокоскоростных внешних устройств, таких как видеокамеры, видеомагнитофоны и т.д.

Контрольные вопросы:

1. Перечислите интерфейсы накопителей и дайте их краткую характеристику.
2. Дайте сравнительную характеристику интерфейса IDE
3. Дайте сравнительную характеристику шины SCSI

Лабораторная работа №14

«Программирование арифметических и логических команд»

Цель работы: Написать программу для выполнения арифметических и логических действий над числами.

Предварительная подготовка: изучить материал раздела «Архитектура и принципы работы основных логических блоков вычислительных систем (ВС)» (по конспекту).

Задание:

1) Выполнить действия над числами в соответствии с вариантом задания ручным счетом. При необходимости осуществить перевод чисел из десятичной в шестнадцатеричную систему счисления.

2) Написать программу для вычисления результата машинным счетом. Ввод чисел в регистры организовать программно. Предоставить листинг программы и соответствующий ему объектный код. Рассчитанное значение выражения сверить с данными ручного расчета.

Вариант	Действия	Начальные значения
1	C←(C+L+H) v E	C=1FH, L=10, H=25H, E=69
2	B←(B-C+D) v E	B=105, C=1BH, D=3EH, E=17
3	E←(B+L)+(D&E)	B=3AH, L=2BH, D=2DH, E=88
4	C←(B v H)+(2*L-C)	B=44, L=53, H=3CH, C=51H
5	H←(E&C)+(L⊕H)	E=4AH, L=4DH, H=62, C=21H
6	L←B⊕(C-H+L)	B=73H, L=22, H=34, C=89H
7	D←(B&D)+(C v D)*2	B=35H, D=45, C=4DH
8	B←(B+H)+(D⊕E)	B=93, H=2CH, D=3FH, E=110
9	E←(C-L)⊕(D v E)	C=5FH, L=3CH, D=15, E=240
10	C←(B&C)&(L-D)	B=7, L=101, D=21H, C=51H
11	H←(C&H+L)-(D⊕H)	D=1AH, L=6BH, H=30, C=28H
12	L←C&(D v L+B)	D=2AH, L=16, B=20, C=30H
13	D←(L+D)-(C v B)	B=74, L=37, D=3FH, C=4CH
14	B←(L⊕C+E) &B	B=2CH, L=3DH, C=28, E=59
15	E←(B⊕C) &(D v E)	B=2CH, C=35, D=3DH, E=70

Теоретические сведения:

Возможности арифметических команд ограничиваются только операциями сложения и вычитания. Умножение, деление и более сложные арифметические операции можно организовать, составив соответствующие подпрограммы на основе имеющихся в распоряжении команд. Кроме того, возможности АЛУ позволяют одной команде оперировать лишь с однобайтными (и немного с двухбайтными) числами. Реализовать операции с многобайтными числами можно программным путем, понимая и используя признаки результатов однобайтных операций, которые устанавливаются командами этой группы в регистре признаков. К арифметическим командам относятся также команды сравнения, поскольку сравнение производится

путем вычитания и установки признаков Z и CY, способных отразить равенство сравниваемых чисел или указать на большее из них.

Логические команды предоставляют возможности непосредственно выполнить следующие операции с однобайтными числами: И (конъюнкция), ИЛИ (дизъюнкция), исключающее ИЛИ (сложение по модулю два), НЕ (инверсия).

Схема выполнения практически всех арифметических и логических

операций может быть представлена следующим образом:

(A) (A) <op> <2-й операнд>

где <op> – символ операции: +, —, & и т.д. Второй операнд может хранится в регистре процессора, в ячейке памяти или быть в составе самой команды. Первый операнд (или единственный операнд для операций с одним операндом) всегда хранится в аккумуляторе. Результат операции отправляется командой в аккумулятор, а признаки этого результата устанавливаются в регистре признаков.

Команды сложения

Команды сложения позволяют выполнять операции сложения однобайтных или двухбайтных операндов.

Команды сложения однобайтных операндов различаются по методам адресации второго операнда и по участию или неучастию в операции бита переноса CY. Общим для них является то, что первое слагаемое и результат хранятся в аккумуляторе. Команды сложения двухбайтных операндов работают с операндами только в регистровых парах: первое слагаемое и результат все такие команды хранят в регистровой HL (регистровая пара HL выступает в роли аккумулятора для однобайтных операций), второе слагаемое можно определять в любой из регистровых трех пар процессора. Двоичное сложение выполняется в соответствии с правилами двоичной арифметики.

Таблица:

Правила двоичного сложения

0 + 0 = 0

0 + 1 = 1

1 + 1 = 0 + перенос 1 в следующий разряд

1 + 1 + 1 = 1 + перенос 1 в следующий разряд

Рассмотрим команды сложения однобайтных чисел.

Команды типа ADD r, ADC r обеспечивают выбор второго операнда регистровым методом. Выполняемое командой ADD r действие:

(A)← (A) + (r)

Эти команды предполагают, что исходные операнды будут предварительно записаны в аккумулятор и в регистр r.

Команды вычитания

Команды вычитания позволяют выполняют операции только с однобайтными операндами. По схеме выполнения и способам определения 2-го операнда, а также по участию или неучастию в операциях бита CY эти команды аналогичны командам однобайтного сложения. Команды выполняют вычитание по правилам двоичного вычитания из двоичной арифметики.

Правила двоичного вычитания

0 – 0 = 0

1 – 0 = 1

1 – 1 = 0

0 – 1 = 1 – заем 1 из следующего разряда

Команды вычитания SUB r и SBB r определяют второй операнд (вычитаемое) регистровым способом. Команды SUB r выполняется по схеме:

(A) ← (A) – (r)

Эти команды предполагают, что исходные операнды будут предварительно записаны в аккумулятор (уменьшаемое) и в регистр r (вычитаемое).

Команды логических операций

Команды логических операций обеспечивают логические операции над соответствующими (имеющими одинаковый номер) битами однобайтных операндов. Влияния битов друг на друга типа переноса или заёма, имевшие место при арифметических операциях, в логических операциях отсутствуют. Поэтому, за счет влияния логических команд на все биты признаков, биты CY и AC после этих команд всегда будут сброшены в 0.

Основные логические команды обеспечивают логические операции –

И (конъюнкция), – ИЛИ (дизъюнкция), – исключающее ИЛИ(сложение по модулю два) в соответствии с правилами логики . Эти команды выполняются по общей схеме арифметически-логических команд: первый операнд и результат операции всегда хранятся в аккумуляторе, а второй операнд может быть выбран модификациями команд: либо в любом из регистров процессора; либо в ячейке памяти M, хранящей адрес в HL; либо непосредственно в составе самой команды.

Команда CMA обеспечивает логическую операцию НЕ над операндом из аккумулятора. Выполняется инвертирование всех битов и результат остаётся в аккумуляторе. Если например, до выполнения команды в аккумуляторе было число 10110101, то после команды CMA там будет число 01001010.

Команды типа ANA r, ORA r, XRA r обеспечивают операции —И, ИЛИ, исключающее ИЛИ соответственно, притом, что второй операнд адресуется регистровым способом через выбор регистра r из регистров A, B, C, D, E, H, L.

Команды типа ANA M, ORA M, XRA M обеспечивают операции И, ИЛИ, исключающее ИЛИ соответственно, притом, что второй операнд адресуется косвенно-регистровым способом через адрес подготовленный предварительно в регистровой паре HL.

Команды типа ANI d8, ORI d8, XRI d8 обеспечивают операции —И, ИЛИ, исключающее ИЛИ соответственно, притом, что второй операнд адресуется прямым способом то есть является вторым байтом d8 в составе самой команды. Команды операций И используется когда требуется, например, оценить состояние какого-то одного из битов в составе байта.

Контрольные вопросы:

1. Какие арифметические операции может выполнять МП?
2. Каковы правила сложения двоичных чисел?
3. Каковы правила вычитания двоичных чисел?
4. Назовите основные логические операции.

Лабораторная работа №15

Программирование ввода - вывода

Цель работы: Освоить команды программирования ввода-вывода и управления процессором.

Предварительная подготовка: изучить материал раздела «Архитектура и принципы работы основных логических блоков вычислительных систем (ВС)» (по конспекту).

Задание:

1. Написать программу для выполнения вывода данных на индикаторы.
2. Запустив программу открыть средства стенда и зафиксировать результат.

Вариант	Задание
1	Зажечь цифру «1» на 1, 3, 5 индикаторах.
2	Зажечь цифру «2» на 2, 3, 4 индикаторах.
3	Зажечь цифру «3» на 3, 6 индикаторах.
4	Зажечь цифру «4» на 1, 4, 6 индикаторах.
5	Зажечь цифру «5» на 1, 2, 5 индикаторах.
6	Зажечь цифру «6» на 1, 3, 6 индикаторах.
7	Зажечь цифру «7» на 1, 2, 3 индикаторах.
8	Зажечь цифру «8» на 2, 5 индикаторах.
9	Зажечь цифру «9» на 1, 5 индикаторах.
10	Зажечь цифру «0» на 3, 4 индикаторах.
11	Зажечь букву «A» на 1, 4 индикаторах.
12	Зажечь букву «b» на 3, 4, 6 индикаторах.
13	Зажечь букву «C» на 2, 4, 6 индикаторах.
14	Зажечь букву «d» на 2, 5, 6 индикаторах.
15	Зажечь букву «h» на 1, 2 индикаторах.

Указания к заданию:

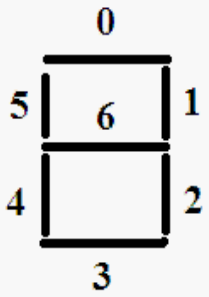
1) Зациклить программу, таким образом, чтобы получить возможность последовательного ввода в ЭВМ кодов символов.

2) Для доступа к дисплею и клавиатуре используются следующие порты ЭВМ:

Регистр чтения клавиатуры - 06;
Регистр сегментов индикаторов - 06;
Регистр сканирования клавиатуры и индикаторов - 07.

Для отображения информации на дисплее предварительно необходимо выбрать один из индикаторов, записью кода выбора в порт 7. Код выбора представляет собой унитарный код, содержащий единственную 1. Соответствующий ей номер позиции отвечает за выбор индикатора. Например код - 01 (1 в нулевом разряде - соответствует самому правому индикатору), код 20 - (1 в шестом разряде - соответствует самому левому, шестому индикатору).

После выбора индикатора для отображения числа необходимо записать его код в регистр сегментов индикаторов. Отображаемые сегменты определяется битами, установленными в 1. Номера битов и соответствующие им сегменты индикаторов отображены на рис.



Например, в результате выполнения команд

```
mvi a, 01
out 07
mvi a, 7f
out 06
```

отобразится цифра 8 в правом знакоместе дисплея.

Лабораторная работа № 16

«Идентификация и установка процессора»

Цель работы: Изучение характеристик процессора. Уметь идентифицировать и устанавливать процессоры.

Предварительная подготовка: изучить материал раздела «Архитектура и принципы работы основных логических блоков вычислительных систем (ВС)» (по конспекту).

Задание 1. Выберите процессор, подходящий для установки на целевой системной плате. Установите процессор на целевую системную плату

Задание 2. Идентифицируйте процессор целевого компьютера. Назовите его основные характеристики.

Задание 3:

1) Составить таблицу поколений ЭВМ по шаблону таблицы 1.
2) Составить таблицу характеристик процессоров Intel и AMD по шаблону таблицы 2.

Таблица 1

Показатель	Поколения ЭВМ				
	Первое	Второе	Третье	Четвертое	Пятое
Элементная база процессора					
Элементная база ОЗУ					
Максимальная емкость ОЗУ, байт					
Максимальное быстродействие процессора (оп/с)					
Языки программирования					
Средства связи пользователя с ЭВМ					

Таблица 2

Тип процессора	Поколение	Год выпуска	Разряд-ность шины данных	Разрядность шины адреса	Первичная кэш-память, Кбайт		Тактовая частота шины, Мгц	Тактовая частота процессора, Мгц	Количество транзисторов, млн	Размер минимальной структуры, мкм
					Команды	Данные				
4004	1	1971	4	12	Нет		0,75	0,75	0,0023	-
8088										
8086										
80286										
80386DX										
80386SX										
80486DX										
80486SX										
80486DX2										
80486DX4										
Pentium										
P-MMX										
Pentium Pro										
Pentium II										
Pentium II Celeron										
Pentium Xeon										
Pentium III										
AMD Athlon										
Pentium 4										
AMD Athlon 64										

Контрольные вопросы:

1. Перечислите функции процессора
2. Прокомментируйте основные параметры процессора
3. Какие современные типы процессоров вы знаете?

Лабораторная работа №17-20

Тема: Изучение команд микропроцессора

Цель работы: Освоить команды программирования микропроцессора

Теоретические сведения

Написание программ для микропроцессорной системы - важнейший и часто наиболее трудоемкий этап разработки такой системы. Для разработки программного обеспечения существуют всевозможные программные средства. Чаще всего применяются языки программирования высокого уровня, такие как Паскаль и Си. Самые компактные и быстрые программы и подпрограммы создаются на языке Ассемблер, представляющем собой символьную запись цифровых кодов машинного языка, кодов команд микропроцессора.

Основная функция любого микропроцессора - выполнение команд. Система команд, выполняемых процессором, представляет собой нечто подобное таблице истинности логических элементов или таблице режимов работы более сложных логических микросхем. Она определяет логику работы микропроцессора и его реакцию на те или иные комбинации внешних событий. Знание системы команд и языка Ассемблер позволяет в несколько раз повысить эффективность некоторых наиболее важных частей программного обеспечения любой микропроцессорной системы - от микроконтроллера до персонального компьютера. Язык машинных кодов понятен процессору, но неудобен для работы программисту, поскольку команды трудно запоминаются и читаются. Гораздо удобнее символьное представление команд, когда символы команды напоминают о ее действиях, а двухбайтные операнды и адреса в командах пишутся в нормальном виде (начиная слева, от старших байт). Такая форма представления команд используется в языке Ассемблер. Символы, представляющие команды, называются мнемокодами. Характерной особенностью языка Ассемблер, определяющей его принадлежность в языкам низкого уровня, является то, что каждой команде Ассемблера соответствует одна и только одна машинная команда.

Рассмотрим основные типы команд, имеющиеся у большинства микропроцессоров, и особенности их применения.

Каждая команда, выбираемая (читаемая) из памяти микропроцессором, определяет алгоритм его поведения на ближайшие несколько тактов. Код команды говорит о том, какую операцию предстоит выполнить микропроцессору и с какими операндами (то есть кодами данных), где взять исходную информацию для выполнения команды и куда поместить результат (если необходимо). Код команды может занимать от одного до нескольких байт, причем микропроцессор узнает о том, сколько байт команды ему надо читать из первого прочитанного им байта или слова. В микропроцессоре код команды расшифровывается и преобразуется в набор микроопераций, выполняемых отдельными узлами микропроцессора

Пример:

Адрес Число Мнемокод

0800 78 MOV A,B

MOV – сокращение от move (переместить). После пробела указывают-ся операнды: сначала – операнд-приемник (регистр A), а затем – операнд-источник (регистр B). Операнды разделяются друг от друга запятой.

Система команд микропроцессора КР580ИК80А – i8080

Таблица 1– Коды регистров и пар регистров, используемые в командах МП

Регистры				Пары регистров			
Код	Имя (r)	Код	Имя (r)	Код (RP)	Имя пары (rp)	Регистры пары	
						старший	младший
000	B	100	H	00	B	B	C
001	C	101	L	01	D	D	E
010	D	110	M (память)	10	H	H	L
011	E	111	A (аккумулятор)	11	PSW	A	PSW

Список команд. Команды МП КР580ИК80А приведены в табл. 3-6. Трехбайтовые поля адресации источника и приемника информации кодируются в машинных командах символами SSS и DDD соответственно.

В мнемонических изображениях двухадресных команд приемник указывается на первом месте, а источник — на втором. В описаниях команд для обозначения содержимого регистра или ячейки памяти используется запись вида: (rl), (r), (H), (M) и т. п.

Таблица 2 – Коды условий, используемые в командах условных переходов

Код (CCC)	Мнемо-ника (cc)	Условие	Код (CCC)	Мнемо-ника (cc)	Условие
000	NZ	Не нуль (Z=0)	001	Z	Нуль (Z = l)
010	NC	Нет переноса (C = 0)	011	C	Перенос (C = 1)
100	PO	Нечетность (P = 0)	101	PE	Четность (P = 1)
110	P	Плюс (S = 0)	111	M	Минус (S = l)

Таблица 3 – Список команд передачи данных

Мнемоника	Код	Длина	Тактов	Флажки CYZMPAC	Функция
MOV R1,R2	01DDDSSS	1	4(5)	-----	R1←R2
MOV R,M	01DDD110	1	7	-----	R← (HL)
MOV M,R	01110SSS	1	7	-----	(HL)←R
MVI R,data8	00DDD110	2	7	-----	R←data8
MVI M,data8	36	2	10	-----	M←data8
LDA addr	3A	3	13	-----	A←(addr)
STA addr	32	3	13	-----	(addr) ←A
LDAX B	0A	1	7	-----	A←(BC)
LDAX D	1A	1	7	-----	A←(DE)
STAX B	02	1	7	-----	(BC) ←A
STAX D	12	1	7	-----	(DE) ←A
LXI B,data16	01	3	10	-----	BC←data16
LXI D,data16	11	3	10	-----	DE←data16
LXI H,data16	21	3	10	-----	HL←data16
LXI SP,data16	31	3	10	-----	SP←data16
LHLD addr	2A	3	16	-----	HL←(addr)
SHLD addr	22	3	16	-----	(addr) ←HL
SPHL	F9	1	5	-----	SP←HL
PUSH B	C5	1	12(11)	-----	-(SP) ←BC
PUSH D	D5	1	12(11)	-----	-(SP) ←DE
PUSH H	E5	1	12(11)	-----	-(SP) ←HL
PUSH PSW	F5	1	12(11)	-----	-(SP) ←PSW
POP B	C1	1	10	-----	BC←(SP)+
POP D	D1	1	10	-----	DE←(SP)+
POP H	E1	1	10	-----	HL←(SP)+
POP PSW	F1	1	10	+++++	PSW←(SP)+
XCHG	EB	1	10(4)	-----	DE↔HL
XTHL	E3	1	16(18)	-----	(SP) ↔HL

Таблица 4 – Арифметические команды МП

Мнемоника	Код	Длина	Тактов	Флажки CY Z M P AC	Функция
ADD R	10000SSS	1	4	+++++	A←A+R
ADC R	10001SSS	1	4	+++++	A←A+R+CY
SUB R	10010SSS	1	4	+++++	A←A–R
SBB R	10011SSS	1	4	+++++	A←A–R–CY
ADD M	86	1	7	+++++	A←A+(HL)
ADC M	8E	1	7	+++++	A←A+(HL)+CY
SUB M	96	1	7	+++++	A←A–(HL)
SBB M	9E	1	7	+++++	A←A–(HL)–CY
ADI data8	C6	2	7	+++++	A←A+data8
ACI data8	CE	2	7	+++++	A←A+data8+CY
SUI data8	D6	2	7	+++++	A←A–data8
SBI data8	DE	2	7	+++++	A←A–data8–CY
INR R	00DDD100	1	4(5)	- ++++	R←R+1
DCR R	00DDD101	1	4(5)	- ++++	R←R–1
INR M	34	1	10	- ++++	(HL) ←(HL)+1
DCR M	35	1	10	- ++++	(HL) ←(HL)–1
DAA	27	1	4	+++++	A←2/10 коррекция A
DAD B	09	1	10	+- - - -	HL←HL+BC
DAD D	19	1	10	+- - - -	HL←HL+DE
DAD H	29	1	10	+- - - -	HL←HL+HL
DAD SP	39	1	10	+- - - -	HL←HL+SP
INX B	03	1	6	- - - - -	BC←BC+1
INX D	13	1	6	- - - - -	DE←DE+1
INX H	23	1	6	- - - - -	HL←HL+1
INX SP	33	1	6	- - - - -	SP←SP+1
DCX B	0B	1	6	- - - - -	BC←BC–1
DCX D	1B	1	6	- - - - -	DE←DE–1
DCX H	2B	1	6	- - - - -	HL←HL–1
DCX SP	3B	1	6	- - - - -	SP←SP–1

Таблица 5 – Логические команды МП

Мнемоника	Код	Длина	Тактов	Флажки CYZMPAC	Функция
ANA R	10100SSS	1	4	0+++0	A←A AND R
XRA R	10101SSS	1	4	0+++0	A←A XOR R
ORA R	10110SSS	1	4	0+++0	A←A OR R
CMP R	10111SSS	1	4	+++++	A – R
ANA M	A6	1	7	0+++0	A←A AND (HL)
XRA M	AE	1	7	0+++0	A←A XOR (HL)
ORA M	B6	1	7	0+++0	A←A OR (HL)
CMP M	BE	1	7	+++++	A – (HL)
ANI data8	E6	2	7	0+++0	A←A AND data8
XRI data8	EE	2	7	0+++0	A←A AND data8

ORI data8	F6	2	7	0+++0	A←A AND data8
CPI data8	FE	2	7	+++++	A←A AND data8
RLC	07	1	4	+- - - -	$A_7 \leftarrow A_6 \leftarrow \ldots A_0 \leftarrow A_7$
RRC	0F	1	4	+- - - -	$A_0 \leftarrow A_1 \leftarrow \ldots A_7 \leftarrow A_0$
RAL	17	1	4	+- - - -	$A_7 \leftarrow A_6 \leftarrow \ldots A_0 \leftarrow CY \leftarrow A_7$
RAR	1F	1	4	+- - - -	$A_0 \leftarrow A_1 \leftarrow \ldots A_7 \leftarrow CY \leftarrow A_0$
CMA	2E	1	4	- - - - -	A←NOT A
CMC	3F	1	4	+- - - -	CY←NOT CY
STC	37	1	4	1- - - -	CY←1

Таблица 6 – Команды ВВ и управления процессором

Мнемоника	Код	Длина	Тактов	Флажки CYZMPAC	Функция
IN port	DB	2	10	- - - - -	A←(port)
OUT port	D3	2	10	- - - - -	(port) ←A
RST n	11NNN111	1	11	- - - - -	-(SP) ←PC←8 x n, n=0-7
EI	FB	1	4	- - - - -	Разрешение прерываний
DI	F3	1	4	- - - - -	Запрет прерываний
RIM	20	1	4	- - - - -	Чтение маски прерывания
SIM	30	1	4	- - - - -	Запрет маски прерывания
HLT	76	1	5(7)	- - - - -	Останов
NOP	00	1	4	- - - - -	Нет операции

Описание симулятора МП КР580ВМ80

Данная программа предназначена для изучения функционирования МП КР580ВМ80 (импортный аналог), программирования на языке Ассемблер.
Интерфейс программы приведен на рис. 1. В нем можно выделить следующие блоки:
- блок состояния памяти;
- блок состояния регистров;
- блок битов состояний;
- рабочая область;
- указатель стека;
- порты ввода и вывода
- панель инструментов.

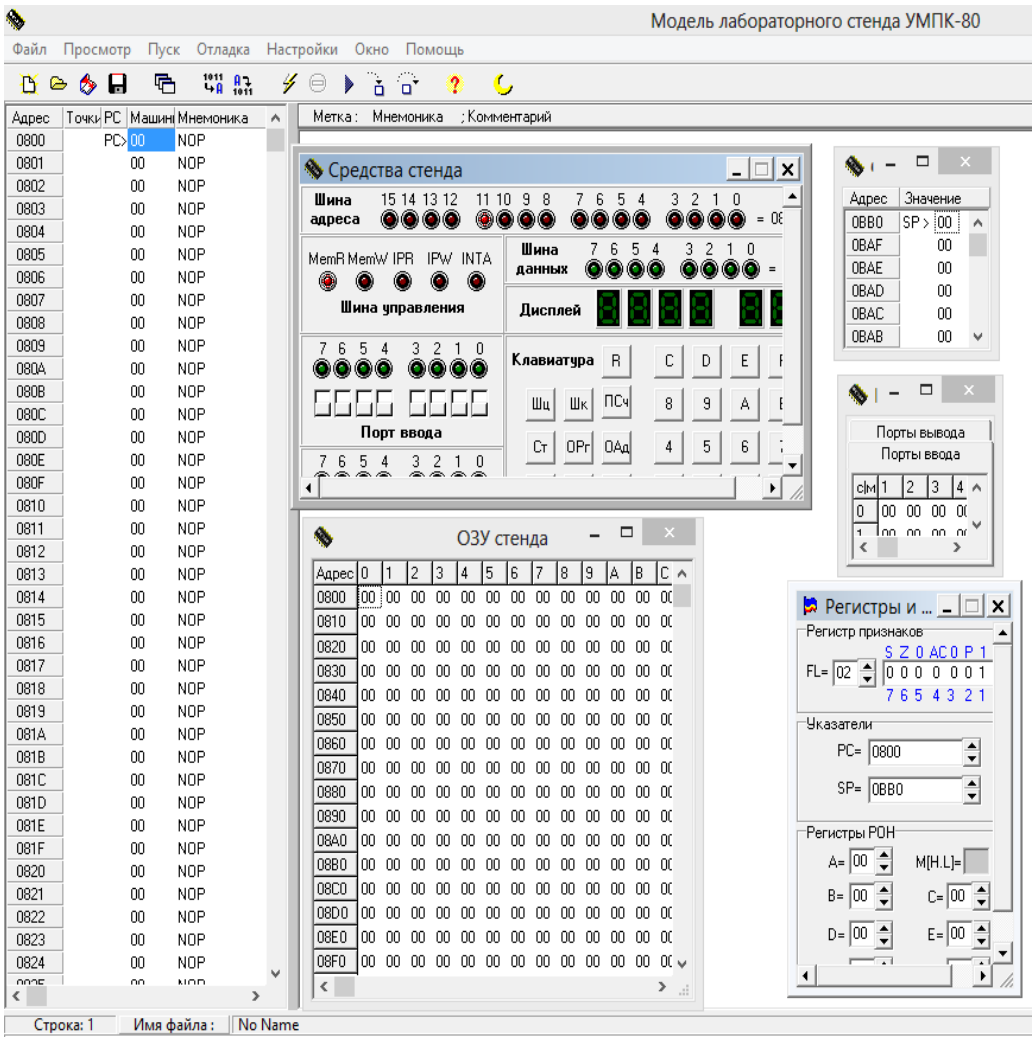


Рис. 1. Интерфейс симулятора микропроцессора КР580ВМ80 (I8080)

Программа может выполнять следующие функции:
- ввод информации и вызов директив с помощью клавиатуры;
- отображение вводимой информации в шестнадцатеричном коде на дисплее симулятора;
- запуск, выполнение и отладка программ пользователя.

Для ввода программы пользователю необходимо:
- в меню «Файл» выбрать «Новый» (рис. 2).

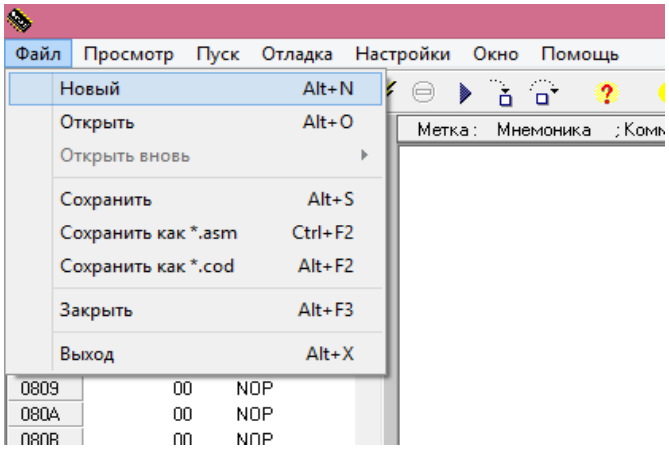


Рис. 2. Вид меню «Файл»

Теперь симулятор готов к записи программы пользователя. В блоке памяти по каждому адресу в исходном состоянии записываются пустые комманды NOP (рис. 3).

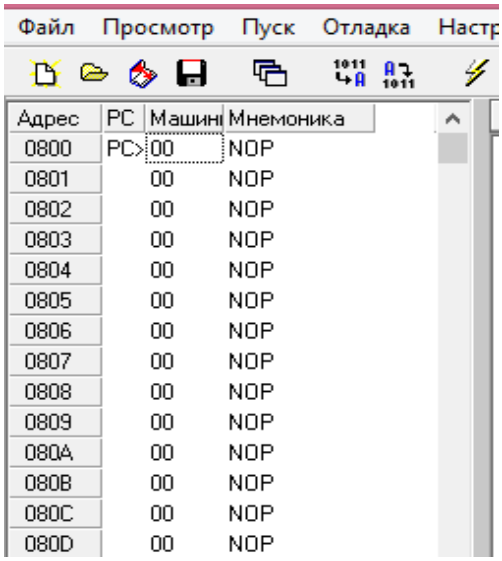


Рис. 3. Окно отображения состояния памяти

Блок памяти содержит три поля:
- поле адресов ячеек памяти;
- поле мнемокодов команд;
- шестнадцатеричный (машинный) код команды.

Новую команду можно ввести следующим образом:
- указать в области памяти на нужный адрес;
- перейти в окно ввода команд;
- записать команду, вводя его с клавиатуры компьютера (рис. 4).

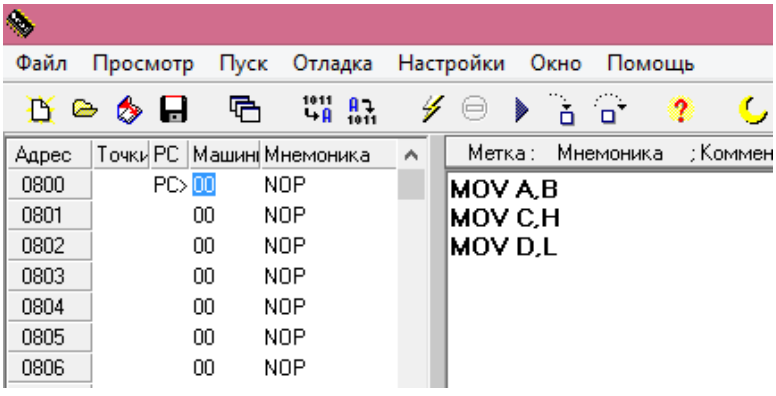


Рис. 4. Рабочее поле

После занесения программы в память, необходимо выполнить ассемблирование. Ассемблирование – процесс трансляции программы с языка ассемблера в машинный код. Ассемблирование однозначный и обратимый процесс.
В языке ассемблера каждой мнемонике соответствует одна машинная инструкция.

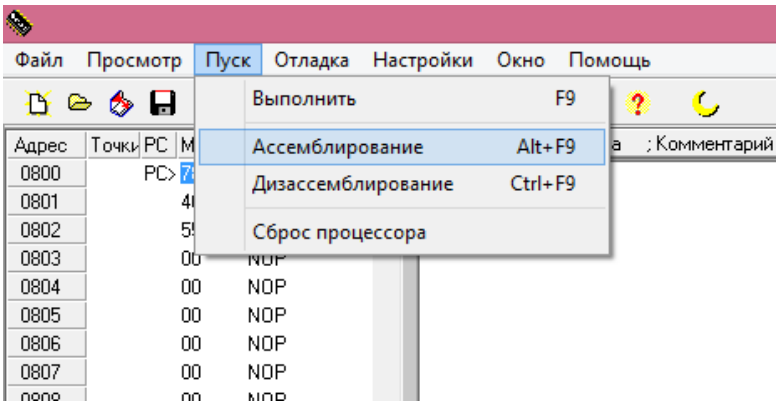


Рис. 5. Ассемблирование

После ассемблирования можно приступать к ее отладке и запуску. Сначала устанавливаем указатель памяти на начальный адрес. Пользователь может вызвать пошаговый режим выполнения программы, нажимая на клавиатуре кнопку F7 (одно нажатие – выполнение одной команды), либо воспользовавшись специальной кнопкой на панели инструментов  , либо выполнив следующее: Отладка→Шаг команды (рис. 6).

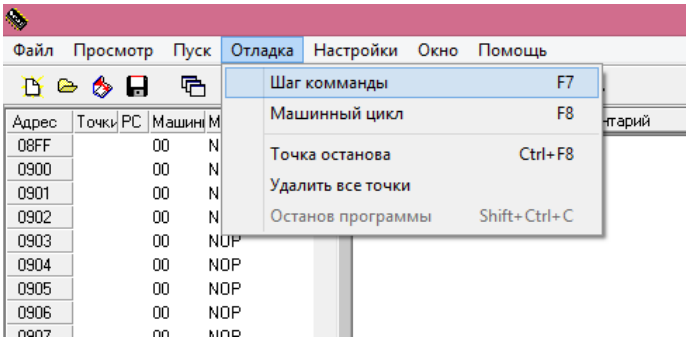


Рис. 6. Пошаговое выполнение программы

Так же запустить программу можно сразу, воспользовавшись специальной кнопкой на панели инструментов  .
Проследить выполнение и увидеть результаты работы программы можно, воспользовавшись окнами отображения регистров и флагов, состоянием портов ввода-вывода (см. рис. 7,8).

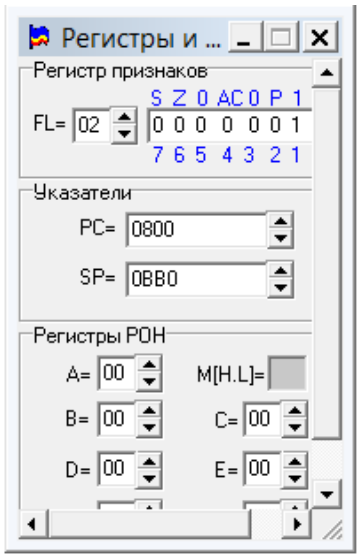


Рис. 7. Отображение состояния регистров

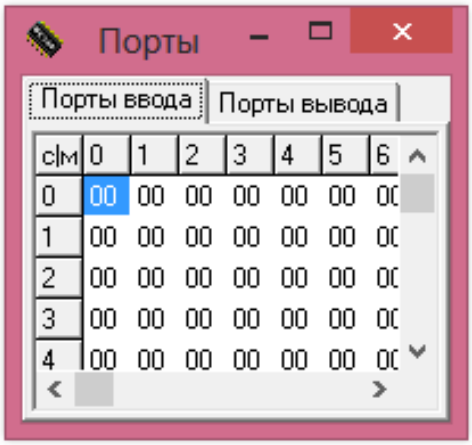


Рис. 8. Состояние портов ввода-вывода

В окне интерфейса также имеется специально выделенная область, в которую можно вносить различные записи о выполняемой программе на данном симуляторе: условие задания, комментарии.
Полученные результаты можно сохранить в удобном для пользователя виде, выбрав в меню «Файл» соответствующий формат файла: текстовый или файл памяти.
Выход из программы осуществляется нажатием комбинации клавиш Alt+X.

Контрольные вопросы:

1. Какие программные средства используются при написании программ для микропроцессорной системы?
2. Какая форма представления команд используется?
3. Какими группами представлена система команд микропроцессора (привести примеры по каждой группе команд)?
4. Что такое командный цикл, машинный цикл?