ТОЛЬЯТТИНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

В.В. Ермаков, М.А. Пьянов

МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА И СИСТЕМЫ

Практикум по лабораторным работам

Тольятти
ТГУ
2011

Министерство образования и науки Российской Федерации
Тольяттинский государственный университет
Электротехнический факультет
Кафедра «Электрооборудование автомобилей и электромеханика»

В.В. Ермаков, М.А. Пьянов

МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА И СИСТЕМЫ

Практикум
по лабораторным работам

Тольятти
ТГУ
2011

УДК 621.3.049
ББК 32.973.26-04
Е721

Рецензенты:
д.т.н., профессор Поволжского государственного
университета сервиса *А.А. Кувшинов*;
к.т.н., доцент Тольяттинского государственного
университета *В.В. Королев*.

Е721 Ермаков, **В.В. Микропроцессорные средства и системы : практикум** по лабораторным работам / В.В. Ермаков, М.А. Пьянов. – Тольятти : ТГУ, 2011. – 72 с. : обл.

Практикум содержит краткие теоретические сведения по микропроцессорным системам, методические указания по выполнению лабораторных работ, примеры тестов промежуточного контроля знаний, задания на разработку программного обеспечения и пример выполнения такого задания.

Предназначен для студентов электротехнических специальностей вузов: 140607 «Электрооборудование автомобилей и тракторов» и 140601 «Электромеханика».

УДК 621.3.049
ББК 32.973.26-04

Рекомендовано к изданию научно-методическим советом Тольяттинского государственного университета.

Предисловие

Одной из важных задач нашего времени является подготовка высококлассных специалистов в высших учебных заведениях. Проведение лабораторных работ является одним из основных этапов подготовки специалистов. Лабораторные работы позволяют закрепить полученные теоретические знания, приобрести практические навыки, необходимые в последующей работе на производстве.

Настоящий лабораторный практикум состоит из семи лабораторных работ, охватывающих все необходимые разделы общего курса. Количество работ, включенных в практикум, может быть меньше, чем предусмотрено программой. Поэтому при разработке календарного плана учебного года и планировании лабораторного практикума необходимо учитывать работы по специализации данного курса для каждой из специальностей.

Задания на разработку программного обеспечения состоят из трех уровней различной степени сложности. Для завершающей работы – разработка программного обеспечения низкого уровня. Варианты содержат методику решения заданий на разработку программного обеспечения.

Практика проведения лабораторных работ показывает, что число членов в бригаде не должно превышать трех человек. Такой оптимальный численный состав бригады взят за основу при проведении большинства лабораторных работ. Однако при проведении лабораторных работ, в которых предусмотрено использование программного эмулятора KP580BM80/INTEL8080, при установке его на персональные компьютеры типа IBM PC необходимо оставить в бригаде двух человек, а также возможно индивидуальное выполнение комплекса.

Обучение студентов с использованием всего комплекса современных обучающих технологий позволяет повысить качество знаний студентов, их практические навыки в программировании на языках низкого уровня.

Описание всех лабораторных работ построено по единой системе. В лабораторный практикум включены краткие теоретические сведения, рекомендации по составлению отчета, контрольные вопросы и задания для самопроверки.

Практикум предназначен для студентов инженерно-технических специальностей высших учебных заведений, изучающих микропроцессорные средства и системы.

Общие методические рекомендации по выполнению и оформлению лабораторных работ

Правила выполнения работ в лаборатории

1. Лабораторные работы выполняют бригады, состоящие из двух-трех человек.
2. Студенты должны проводить работы в строгом соответствии с календарным планом.
3. Каждый член бригады должен подготовиться к выполнению работы: изучить задание, составить алгоритм решения задачи, составить программу, подготовить бланк-отчет о выполнении работы.
4. Перед началом работы со студентами необходимо проводить краткое собеседование по содержанию работы. Недостаточно подготовленные студенты к занятиям не допускаются.
5. Студенты, допущенные к работе, переходят на свое, закрепленное за ними рабочее место.
6. Включать лабораторное оборудование можно только после подробного ознакомления с его устройством, правилами работы с ним и с разрешения преподавателя.
7. При выполнении работы следует придерживаться последовательности, указанной в описании лабораторной работы.
8. По окончании работы каждый член бригады должен представить преподавателю протокол проведения работы; преподаватель проверяет полученные результаты и подписывает работу, только в этом случае работа считается законченной.
9. К следующему занятию студенты представляют полностью оформленный письменный отчет о выполненной работе.

Составление отчета

1. Отчет о работе выполняют чернилами на одной стороне отдельных листов формата А4 с полями: 3 см – слева, и по 2 см – справа, сверху и снизу; либо с применением печатного текста, шрифт Times New Roman Cyr, высота – 14, интервал полуторный.
2. На обложке отчета должны быть указаны фамилия, имя, отчество студента, шифр группы, фамилия, имя, отчество преподавателя, название вуза, название кафедры, год выполнения.

3. В отчет необходимо внести следующие данные: дату выполнения работы; номер и название работы; цель работы; краткую теоретическую часть; задание, выданное преподавателем; алгоритм решения задачи; программное решение задачи; протокол выполнения работы; выводы по лабораторной работе.
4. Алгоритм и все рисунки следует выполнять в строгом соответствии со стандартами и правилами ЕСКД.

Техника безопасности при проведении лабораторных работ

Лабораторные работы выполняются с применением электрических, электронных и компьютерных систем. При их эксплуатации необходимо строго соблюдать следующие правила техники безопасности.

1. На одной лабораторной установке должны работать не менее двух человек. Исключение составляет лабораторно-технический комплекс КР580ВМ80/Intel8080, сымитированный программным путем на персональном компьютере типа IBM PC, где возможно выполнение работ одним человеком.
2. Перед началом работы на лабораторной установке необходимо убедиться, что все выключатели установки находятся в положении «Выключено».
3. Перед началом работы путем внешнего осмотра необходимо убедиться в отсутствии внешних повреждений лабораторного оборудования, токоведущих частей и электрической изоляции. Обо всех обнаруженных неисправностях или нарушениях техники безопасности следует сообщить преподавателю или лаборанту.
4. Категорически запрещается включать стендовое оборудование, если на нем открыты защитные панели, закрывающие токоведущие части оборудования.
5. Категорически запрещается включать стендовое оборудование без разрешения преподавателя.
6. При проведении лабораторных работ на стендовом оборудовании все переключения и регулировки должен осуществлять один человек и только одной рукой. Вторая рука должна быть свободна и не касаться аппаратуры стендового оборудования.
7. Запрещается касаться металлических элементов двух различных работающих установок.

8. Студентам запрещается производить какие-либо переключения и регулировки на стенде, не предусмотренные предписанием по выполнению лабораторной работы.
9. В случае неисправности электрического оборудования лабораторных установок, а также при появлении дыма, искрения или запаха перегретой изоляции необходимо немедленно обесточить установку. Обо всех неисправностях следует сообщить преподавателю или лаборанту.
10. Во время проведения лабораторных работ запрещается отходить от лабораторных установок, включенных в сеть.
11. При поражении человека электрическим током следует немедленно обесточить установку. При потере сознания и остановке дыхания пострадавшего следует освободить от стесняющей одежды. До прибытия врача необходимо делать пострадавшему искусственное дыхание и непрямой массаж сердца.
12. Перед началом цикла лабораторных работ студент должен ознакомиться с настоящими правилами, противопожарными инструкциями и требованиями данной лаборатории.

ЛАБОРАТОРНАЯ РАБОТА 1
Системы счисления. Перевод чисел. Дополнительный код. Двоичная арифметика. Арифметика в дополнительном коде. Основные логические функции и элементы. Оперативные и постоянные запоминающие устройства

Цели работы: изучить двоичную, шестнадцатеричную системы счисления; получить практические навыки перевода чисел из одной системы в другую; изучить двоичную арифметику и представление отрицательных чисел в двоичной системе счисления; изучить основные логические функции, реализуемые микропроцессором и логические элементы; изучить статические и динамические оперативные запоминающие устройства словарной и матричной организации; изучить назначение и типы постоянных запоминающих устройств.

Краткие теоретические сведения

Обработка информации в электронных устройствах выполняется двумя методами: аналоговым или цифровым. При цифровом методе обработки информации величины представляются в цифровой форме, а сама обработка сводится к последовательности операций над числами.

Цифровой метод по сравнению с аналоговым имеет ряд преимуществ:

– возможность обеспечения любой требуемой точности обработки;
– высокую помехозащищенность;
– высокую стабильность характеристик обработки;

Цифровая аппаратура обладает:

– малой стоимостью, небольшими габаритами и массой;
– довольно низким потреблением энергии.

Вся цифровая техника работает с двоичными числами. Двоичная система счисления использует только числа 0 и 1. Эти числа называются битом. Физически в цифровых системах бит «0» представлен низким напряжением, а бит «1» – высоким уровнем напряжения.

Системы счисления

1. Десятичная система счисления (система с основанием 10) содержит 10 цифр (от 0 до 9). Кроме того, она характеризуется значением позиции числа или так называемым «весом».

Десятичное число в аналитической форме записи можно представить как

$A = a_{n-1} \cdot 10^{n-1} + a_{n-2} \cdot 10^{n-2} + ... + a_i \cdot 10^i + a_0 \cdot 10^0 + a_{-1} \cdot 10^{-1} + .. + a_{-m} \cdot 10^{-m}$;

число 1327,694 равно 1000 + 300 + 20 + 7 + 0,6 + 0,09 + 0,004.

Степень основания	10^3	10^2	10^1	10^0	10^{-1}	10^{-2}	10^{-3}
Значение позиции	1000	100	10	1	0,1	0,01	0,001
Десятичные числа	1	3	2	7	6	9	4

$1000 \times 1 + 100 \times 3 + 10 \times 2 + 1 \times 7 + 0{,}1 \times 6 + 0{,}01 \times 9 + 0{,}001 \times 4 = 1327{,}694$.

2. Двоичная система счисления (система с основанием 2) содержит 2 цифры (0 и 1). А число представляется последовательностью записи этих символов.

В аналитической записи двоичное число имеет вид:

$B = b_{n-1} \cdot 2^{n-1} + b_{n-2} \cdot 2^{n-2} + ... + b_i \cdot 2^i + b_0 \cdot 2^0 + b_{-1} \cdot 2^{-1} + . + b_{-m} \cdot 2^{-m}$.

Степень основания	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Значение позиции	128	64	32	16	8	4	2	1
Двоичное число	1	0	1	1	0	1	1	0
Десятичные числа	128 + 0 + 32 + 16 + 0 + 4 + 2 + 0 = 182							

$1011.0110_{(2)} = 182_{(10)}$.

3. Шестнадцатеричная система счисления – система с основанием 16. Длинная цепь нулей и единиц при записи двоичного числа трудна для запоминания, а процесс перевода чисел занимает много времени. Поэтому для представления двоичных чисел используют шестнадцатеричную систему счисления (систему с основанием 16), которая включает в себя 16 символов (цифры от 0 до 9 и латинские буквы A, B, C, D, E, F). Причем каждый шестнадцатеричный символ представляется единственным сочетанием четырех бит двоичной системы счисления. Двоичное число разделяется на тетрады, и каждая из них преобразуется в соответствующий шестнадцатеричный символ. Чтобы произвести обратное преобразование, каждый символ шестнадцатеричной системы заменяется четырьмя разрядами двоичной системы.

Перевод чисел 10/2/16 систем счисления

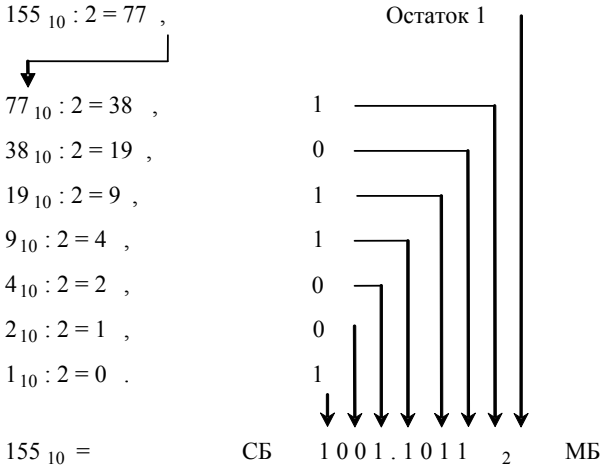
Десятичные	Шестнадцатеричные	Двоичные			
		8	4	2	1
0	0	0	0	0	0
1	1	0	0	0	1
9	9	1	0	0	1

Десятичные	Шестнадцатеричные	Двоичные			
		8	4	2	1
10	A	1	0	1	0
11	B	1	0	1	1
12	C	1	1	0	0
13	D	1	1	0	1
14	E	1	1	1	0
15	F	1	1	1	1

Степень основания	16^3	16^2	16^1	16^0
Значение позиции	4096	256	16	1
Шестнадцатиричное число	2	C	6	E
Десятичное число	4096×2=8192	256×12=3072	16×6=96	1×14=14
	8192 + 3072 + 96 + 14 = 11374			

Перевод чисел из одной системы счисления в другую

1. *Перевод чисел из десятичной в двоичную систему счисления.* Для перевода десятичного числа в двоичную систему необходимо десятичное число разделить на основание двоичной системы (т. е. на 2), остаток от деления записать (он становится младшим разрядом результата), а частное вновь делить до тех пор, пока частное от деления не станет равным нулю.



2. *Перевод чисел из десятичной в шестнадцатеричную систему счисления.* Для перевода десятичного числа в шестнадцатеричную систему необходимо десятичное число разделить на основание системы (т. е. на 16), остаток от деления записать (он становится младшим разрядом

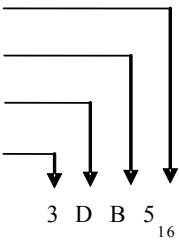
результата), а частное вновь делить до тех пор, пока частное от деления не станет равным нулю (*аналогично 10 ⇒ 2)*.

$15797_{10} : 16 = 987$, остаток $5_{10} = 5_{16}$

$987_{10} : 16 = 61$, остаток $11_{10} = B_{16}$

$61_{10} : 16 = 3$, остаток $13_{10} = D_{16}$

$3_{10} : 16 = 0$. остаток $3_{10} = 3_{16}$



$15797_{10} = 3DB5H = 0011.1101.1011.0101.$

3. Перевод чисел из двоичной в десятичную систему счисления. Для перевода двоичного числа в десятичную систему необходимо провести сложение двоичного числа с учетом веса разряда.

Пример перевода: двоичное число 11.0110 переводим в десятичную систему.

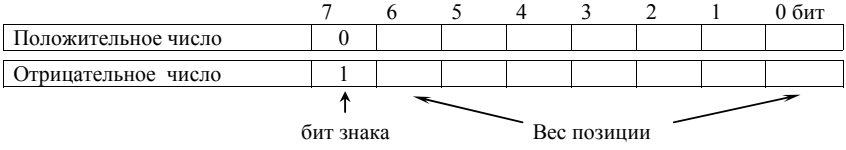
$$110110_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 54_{10}.$$

4. Перевод чисел из шестнадцатиричной в десятичную систему счисления. Для перевода шестнадцатеричного числа в десятичную систему необходимо провести сложение шестнадцатеричного числа с учетом веса разряда.

$$1A4H = 1 \cdot 16^2 + 10 \cdot 16^1 + 4 \cdot 16^0 = 256 + 160 + 4 = 420_{10}.$$

Дополнительный код

Если нужно использовать двоичные числа со знаком, то используют так называемый дополнительный код. При этом старший бит в байте отвечает за знак числа. Если число положительное, то старший бит равен нулю. Если отрицательное – единице.



Перевод в дополнительный код или из дополнительного кода

Для перевода числа в дополнительный код или из дополнительного кода необходимо получить обратный код, т. е. проинвертировать пол-

ный байт числа (заменить все нули единицами, а единицы – нулями), затем прибавить к результату единицу в младший (нулевой) бит.

Перевод в дополнительный код – *число* (-14_{10}).

1. $14_{10} = E_{10}=$ $00001110_{(2)}$.
2. Обратный код: $11110001_{(2)}$.
   $+\underline{\quad\quad 1.}$
3. Дополнительный код: 11110010.

Перевод из дополнительного кода– *число* (11110000_2).

1. Дополнительный код 11110000.
2. Обратный код: 00001111
   $+\underline{\quad\quad 1}$
3. Дополнительный код: $00010000_{(2)} = 16_{10}$,
   т. е. $11110000_2 = -16_{10}$.

Двоичная арифметика

1. Сложение положительных двоичных чисел

$$59_{10} + 42_{10} = 101_{10}.$$

1. Слагаемое: $59_{10} = 3B_{16} = 0011.1011_{(2)}$.
   $+$
2. Слагаемое: $42_{10} = 2A_{16} = 0010.1010_{(2)}$.

$$0110.0101_{(2)} = 65_{16} = 6 \cdot 16^1 + 5 \cdot 16^0 = 96_{10} + 5_{10} = 101_{10}.$$

2. Вычитание положительных двоичных чисел

$$85_{10} - 57_{10} = 28_{10}.$$

1. Уменьшаемое: $85_{10}=55_{16}= 01010101_{(2)}$.
   $-$
2. Вычитаемое: $57_{10}=39_{16}= 00111001_{(2)}$.

$$00011100_{(2)} = 1C_{16} = 1 \cdot 16^1 + 12 \cdot 16^0 = 16_{10} + 12_{10} = 28_{10}.$$

$$\begin{array}{r} \_55H \\ \underline{\quad 39H} \\ 1CH \end{array}$$

3. Вычитание двоичных чисел с использованием дополнительного кода

$$17_{10} - 9_{10} = 17 + (-9) = 8_{10.}$$

1. Число $17_{10} = 11_{16} = 0001.0001_2$.
2. Число – 9_{10}; $9_{10} = 0000.1001_2$.

Обратный код: 1111.0110

\+ 1

Дополнительный код: 1111.0111

3. Сложение: 0001.0001

\+

1111.0111

$1.0000.1000 = 08_{16} = 8_{10}$.

(переполнением пренебречь)

4. Умножение положительных двоичных чисел

$11_{10} \times 13_{10} = 143_{10}$.

1. Множимое: $11_{10} = B_{16} =$ $1011_{(2)}$.
2. Множитель: $13_{10} = D_{16} =$ $1101_{(2)}$.

1-е частичное произведение 1011

0000

1011

4-е частичное произведение 1011

$10001111_{(2)} = 8FH =$

$= 8 \cdot 16^1 + 15 \cdot 16^0 = 128_{10} + 15_{10} = 143_{10}$.

Логические операции, реализуемые ЦП

1. Операция отрицания (инверсия, «НЕ»). $Q = \bar{x}$.

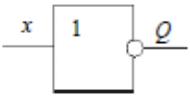


Таблица истинности

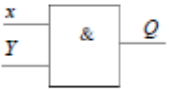
x	Q
0	1
1	0

Электрический аналог

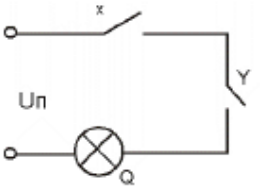
Теоремы Булевой алгебры для 1[й] переменной

$x + \bar{x} = 1$, $x + \bar{x} = 0$, $\bar{\bar{x}} = x$.

2. Операция *конъюнкция* (логическое умножение) $Q = x \wedge Y = x \cdot Y$.



X	Y	Q
0	0	0
0	1	0
1	0	0
1	1	1



«И»

Входы равноценны

Теоремы:

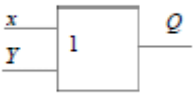
$x \cdot 0 = 0;$

$x \cdot 1 = x;$

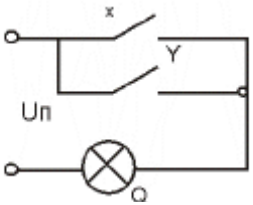
$x \cdot x = x;$

$x \cdot \bar{x} = 0.$

3. Операция *дизъюнкция* (логическое сложение) $Q = x \vee Y = x + Y$.



X	Y	Q
0	0	0
0	1	1
1	0	1
1	1	1



«ИЛИ»

Теоремы:

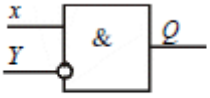
$x + 0 = x;$

$x + 1 = 1;$

$x + x = x;$

$x + \bar{x} = 1.$

4. Операцин *импликации* (запрет) $Q = x \wedge \bar{Y}$.

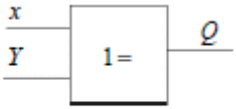


x	Y	Q
0	1	0
1	1	0
0	0	0
1	0	1

«НЕТ»

Вход Y – запрещающий.

5. Операция *неравнозначности* $Q = x\overline{Y} + \overline{x}Y$.



x	Y	Q
0	0	0
0	1	1
1	0	1
1	1	0

«Исключающее ИЛИ»
«$\sum$ по модулю 2».

6. Операция сдвига двоичного числа.

$3EH = 0011.1110 = 62_{10}$.
Сдвиг влево: $0111.1100 = 7CH = 124_{10}$.
Сдвиг вправо: $0001.1111 = 1FH = 31_{10}$.

Операция сдвига двоичного числа влево означает умножение его на два, вправо – деление на два.

Запоминающие устройства (ЗУ)

ЗУ представляет собой функционально законченный МС высокой степени интеграции (БИС), что позволяет создавать устройства памяти большой информационной ёмкости.

1 байт = 8 бит.

1 Кбайт = 2^{10} = 1024 байт.

По способу хранения информации ЗУ делятся на ПЗУ и ОЗУ.

Оперативные запоминающие устройства (ОЗУ)

Предназначены для хранения изменяемой информации. Информация в ОЗУ сохраняется и может быть модифицирована, пока на МС подаётся питание.

ОЗУ может работать в режиме записи, считывания и хранения информации. Основной составной частью микросхем ОЗУ является массив элементов памяти, объединённых в матрицу накопителя. Элемент памяти (ЭП) может хранить один бит информации (0 или 1). Каждый ЭП имеет свой адрес. Для обращения к ЭП необходимо его выбрать с помощью кода адреса, сигналы которого подводят к соответствующим выводам микросхемы. Все ОЗУ энергозависимые, т. е. при выключении питания вся информация, находящаяся в них, уничтожается.

Матрица-накопитель содержит элементы памяти, хранящие 1 бит информации.

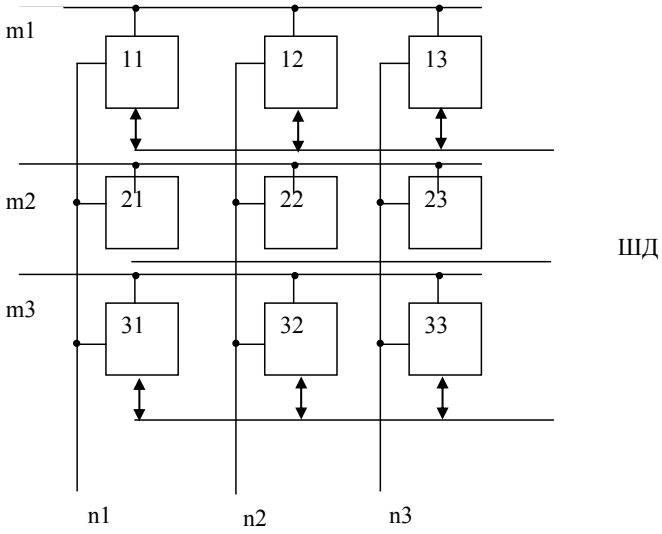


Рис. 1.1. Матрица оперативного запоминающего устройства

Для обращения к нужному элементу матрицы-накопителю необходимо подать адресную комбинацию сигналов на m_i-строку и n_i-столбец матрицы, в результате выбираем один элемент памяти.

ОЗУ в зависимости от принципа построения накопителя имеют матричную или словарную организацию.

ОЗУ матричной организации – обращение только к одной ячейке, независимо от других. Выборка производится по принципу совпадения сигналов возбуждения строки и столбца (матрица).

ОЗУ словарной организации – обращение производится сразу к запоминающим элементам нескольких разрядов по строке (обычно байту).

В зависимости от ЭП, ОЗУ разделяют на статические и динамические.

Статические ОЗУ строятся на DV-триггерах.

МС ОЗУ содержит на одном кристалле матрицу-накопитель, дешифратор адреса, шинные формирователи и схему управления.

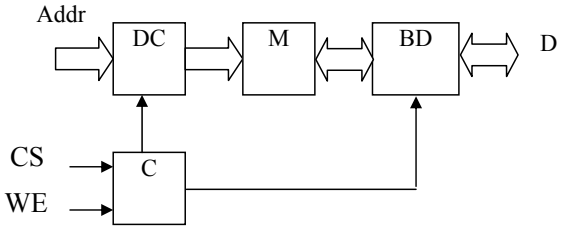


Рис. 1.2. Словарная организация ОЗУ

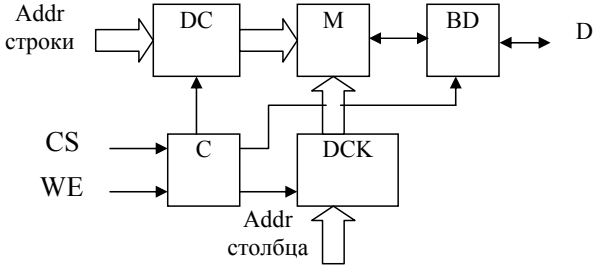


Рис. 1.3. Матричная организация ОЗУ

Для того чтобы записать или считать информацию с матрицы-накопителя, необходимо подать сигналы на схему управления:

CS – сигнал выбора кристалла (МС).

WE – сигнал «запись/считывание».

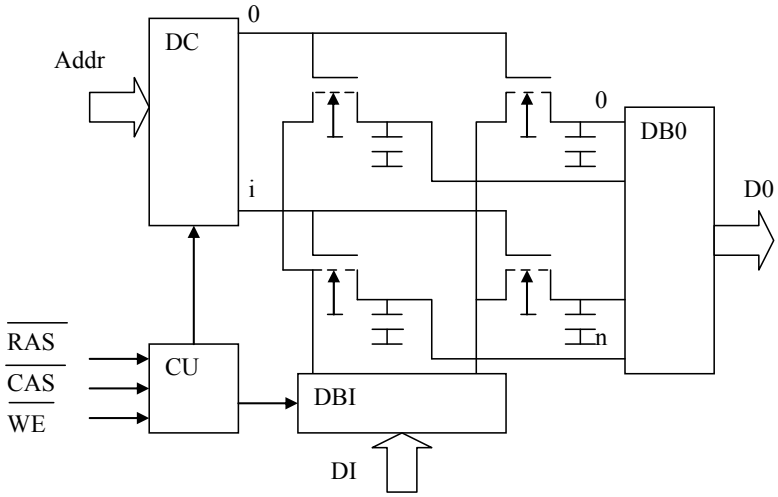
Для построения сверхбольших накопителей ОЗУ в качестве элемента запоминания используется конденсатор и МДП-ключ – такое ОЗУ называется динамическим.

В динамических ОЗУ элементы памяти выполнены на основе электрических конденсаторов, сформированных внутри полупроводникового кристалла. Такие ЭП не могут долгое время сохранять своё состояние, определяемое наличием или отсутствием электрического заряда, и поэтому нуждаются в периодическом восстановлении (регенерации). Микросхемы динамических ОЗУ отличаются от микросхем статических ОЗУ большей информационной ёмкостью, что обусловлено меньшим числом компонентов в одном ЭП и, следовательно, более плотным их размещением в полупроводниковом кристалле. Однако динамические

ОЗУ сложнее в применении, поскольку нуждаются в организации принудительной регенерации и в усложнении устройств управления.

Структура динамического ОЗУ

Регенерация динамического ОЗУ осуществляется за n-циклов обращения к строкам матрицы-накопителя за время, которое меньше периода регенерации (подать *U* на затворы МДП-ключей матрицы).



$\overline{RAS}$ – выбор строки адреса ; $\overline{GAS}$ – выбор столбца адреса

Рис. 1.4. Организация динамического ОЗУ

Отличие динамических от статических ОЗУ:

– в динамических ОЗУ отсутствует источник питания матрицы-накопителя;
– простота матрицы-накопителя;
– необходимость внешних схем регенерации.

Постоянные запоминающие устройства

Модуль памяти, в котором изменить записанную информацию снятием питания невозможно, – постоянное запоминающее устройство (ПЗУ).

ПЗУ работают в режиме считывания информации и предназначены для хранения констант и программ. Процесс занесения информации в ПЗУ – программирование.

Микросхемы ПЗУ построим также по принципу матричной структуры накопителя. Функцию ЭП в микросхемах ПЗУ выполняют перемычки в виде проводников, диодов или транзисторов между шинами строк и столбцов в накопителе. Микросхемы ПЗУ имеют словарную организацию, поэтому информация считывается байтом или полубайтом. При отключении питания информация хранится долго (десятки тысяч часов), поэтому микросхемы ПЗУ называют энергозависимыми.

В состав ПЗУ входят: дешифратор адреса, накопитель, устройство управления и буфер данных.

По способу программирования ПЗУ делятся:

1) на масочные ПЗУ (ПЗУМ), программируемые в процессе изготовления;
2) однократно программируемые у заказчика (ППЗУ);
3) репрограммируемые с электрическим стиранием и записью (ЭРПЗУ);
4) репрограммируемые с электрическим стиранием и электрической записью (РПЗУ-УФ).

1. *Масочные ПЗУ* – МС памяти, информация в которых записывается при изготовлении с фиксированным рисунком межсоединений, определяемых фотошаблоном (маской). Занесённую информацию в ПЗУМ называют прошивкой.

Микросхемы на биполярных транзисторах программируют путём формирования перемычек между строками и столбцами в тех точках матрицы, куда следует занести логическую 1, и не формируют её там, где необходимо занести 0.

В микросхемах ПЗУ, изготовленных на МДП-транзисторах, программирование осуществляют по способу формирования МДП-транзисторов с двумя порогами отпирания – низким и высоким (эти пороги имеют разную толщину подзатворного диэлектрика).

2. *Однократно программируемые ПЗУ* (ППЗУ) – МС памяти, в которых записать информацию можно после изготовления, разрушив плавкие перемычки между эмиттерами транзисторов матрицы накопителя и шинами данных.

Операция программирования заключается в разрушении кристалла импульсами тока амплитудой 30...150 мА. Перемычки устанавливают из поликремния или силицида платины. Перемычка выполняет роль ЭП. Если она выжжена – 0, если нет – 1.

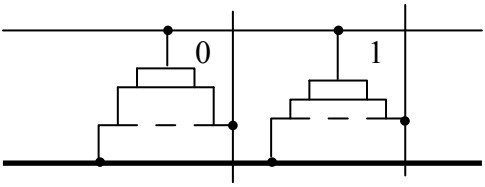


Рис. 1.5. Элемент памяти ПЗУ для МДП-структуры

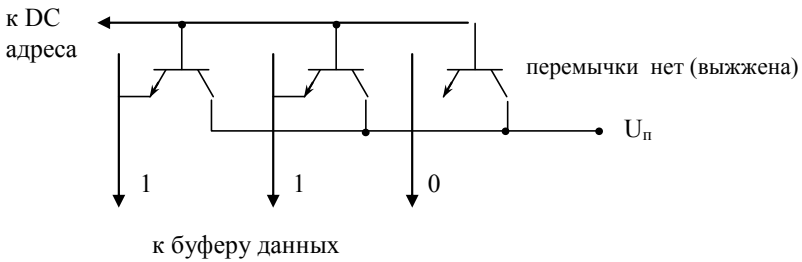


Рис. 1.6. Элемент однократно программируемого ПЗУ

3. *Репрограммируемые ПЗУ* с электрическим стиранием и записью информации (ЭРПЗУ) – МС, использующие элементы коммутации, которые можно устанавливать в одно (замкнутое) состояние избирательно, а в другое (разомкнутое) коллективно.

Основная отличительная особенность микросхем РПЗУ заключается в их способности к многократному (до 200 тысяч) перепрограммированию своим пользователем. Это качество микросхем обеспечено применением ЭП со свойствами управляемых «перемычек», функцию которых выполняют транзисторы со структурной nМОП (металл Al – нитрид кремния Si_3N_4 – окисел кремния SiO_2 – полупроводник Si).

Нитрид кремния способен долгое время сохранять электрический заряд после подачи на затвор транзистора программирующего импульса.

Программирование ПЗУ сводится сначала к коллективной установке перемычек в одно состояние, что равносильно стиранию ранее

записанной информации (логическая 1), и последующей поочерёдной установке нужных перемычек в другое состояние (логический 0).

Программирование осуществляется следующим способом: между затвором и подложкой подают напряжение +25...30 В, под действием сильного электрического поля уменьшается пороговое напряжение. Это состояние соответствует логической 1. При подаче –25...30 В увеличивается пороговое напряжение. Это состояние – логический 0.

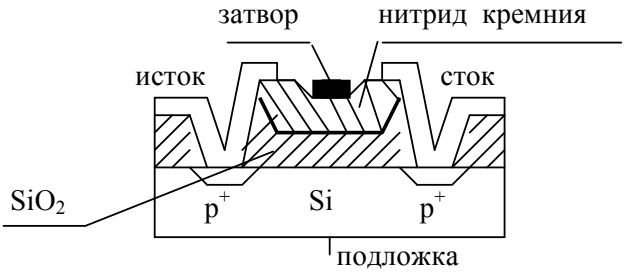


Рис. 1.7. Элемент памяти ПЗУ с электрической записью/стиранием

Время хранения данных – от 5000 до 25000 часов. Количество циклов «запись/стирание» – до 150–200 тысяч раз.

4. *ПЗУ с электрической записью и УФ-стиранием информации* (РПЗУ–УФ).

Основная отличительная особенность микросхем РПЗУ заключается в их способности к многократному (до 25 раз) перепрограммированию своим пользователем. Это свойство микросхем обеспечено применением n-МОП транзисторов с плавающим затвором.

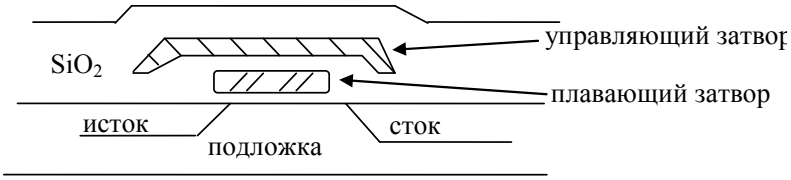


Рис. 1.8. Элемент памяти ПЗУ с УФ-стиранием

Запоминающая матрица выполнена на основе МОП-транзисторов с двумя затворами.

При записи «0» на управляющий затвор подаётся (U = 25 В) и электроны, приобретая большую энергию, уходят в окисел и образуют плавающий затвор (и данный МОП-транзистор проводит ток). При записи «1» плавающий затвор не образуется.

Стирание УФ-излучением в течение 30 мин. Квант излучения передаёт электронам плавающего затвора дополнительную энергию и они выбиваются в подложку кристалла, т. е. плавающий затвор исчезает (логическая 1).

Время хранения данных – до 250 тыс. часов. Количество циклов «запись/стирание» – до 25 раз.

Содержание отчета

Отчет должен содержать:

- цель работы;
- примеры перевода чисел из десятичной системы в шестнадцатеричную и двоичную, а также пример обратного перевода;
- пример перевода в дополнительный код и из него;
- примеры двоичной арифметики в прямом и дополнительном коде;
- таблицы истинности и уравнения логических функций;
- структурные схемы запоминающих устройств со словарной и матричной организацией накопителей;
- выводы по работе.

Вопросы и задания для самопроверки

1. Переведите десятичное число 184 в двоичную и шестнадцатеричную системы счисления.
2. Переведите шестнадцатеричное число 184H в двоичную и десятичную системы счисления.
3. Переведите десятичное число –54 в двоичную и шестнадцатеричную системы счисления с использованием дополнительного кода.
4. Переведите шестнадцатеричное число BFH в десятичную систему с использованием дополнительного кода.
5. Какие два типа ОЗУ существуют?
6. Какая длина слова ОЗУ 256 8?
7. Сколько слов данных содержит ОЗУ 1024 8?
8. Какие типы ПЗУ существуют?

ЛАБОРАТОРНАЯ РАБОТА 2
Структура однокристального центрального процессора

Цель работы – изучить структуру однокристального центрального процессора, работу и назначение его логических блоков.

Краткие теоретические сведения

К настоящему времени сложились некоторые принципы, которые формируют структурную схему центрального процессора (ЦП) и позволяют наглядно рассмотреть его работу. Использование такой схемы облегчает понимание того, как ЦП решает поставленные задачи. В дальнейшем будем пользоваться структурной схемой однокристального ЦП (рис. 2.1).

Однокристальный ЦП состоит из следующих блоков: арифметико-логического устройства (АЛУ), блока управления и синхронизации, регистров общего назначения (РОН), аккумулятора, регистра временного хранения, регистра состояния (флагов), регистра и дешифратора команд, счетчика команд, блока десятичной коррекции, служебных регистров, регистра указателя стека.

1. *Арифметико-логическое устройство*. АЛУ выполняет одну из главных функций ЦП – обработку данных. АЛУ имеет два входа, что позволяет принимать данные из главного регистра ЦП, называемого аккумулятором, и регистра временного хранения данных.

Результат выполняемых операций из АЛУ поступает по внутренней шине данных в аккумулятор. Кроме результата выполнения операции, АЛУ выдает признаки поверки результата в регистр состояния (флагов). АЛУ используется в тех случаях, когда требуется изменить или проверить значение слов данных. Перечень функций АЛУ зависит от типа ЦП. Типичными операциями, выполняемыми АЛУ большинства ЦП, являются следующие: сложение, вычитание, инверсия, сдвиг влево, сдвиг вправо, конъюнкция, дизъюнкция, исключающее «ИЛИ».

2. *Аккумулятор* является главным регистром хранения данных ЦП. Большинство арифметических и логических операций осуществляется путем использования аккумулятора и АЛУ. Любая из таких операций над двумя операндами предполагает размещение одного из них в аккумуляторе, а другого – в одном из регистров. Результат выполнения операции помещается в аккумулятор, замещая там один из операндов.

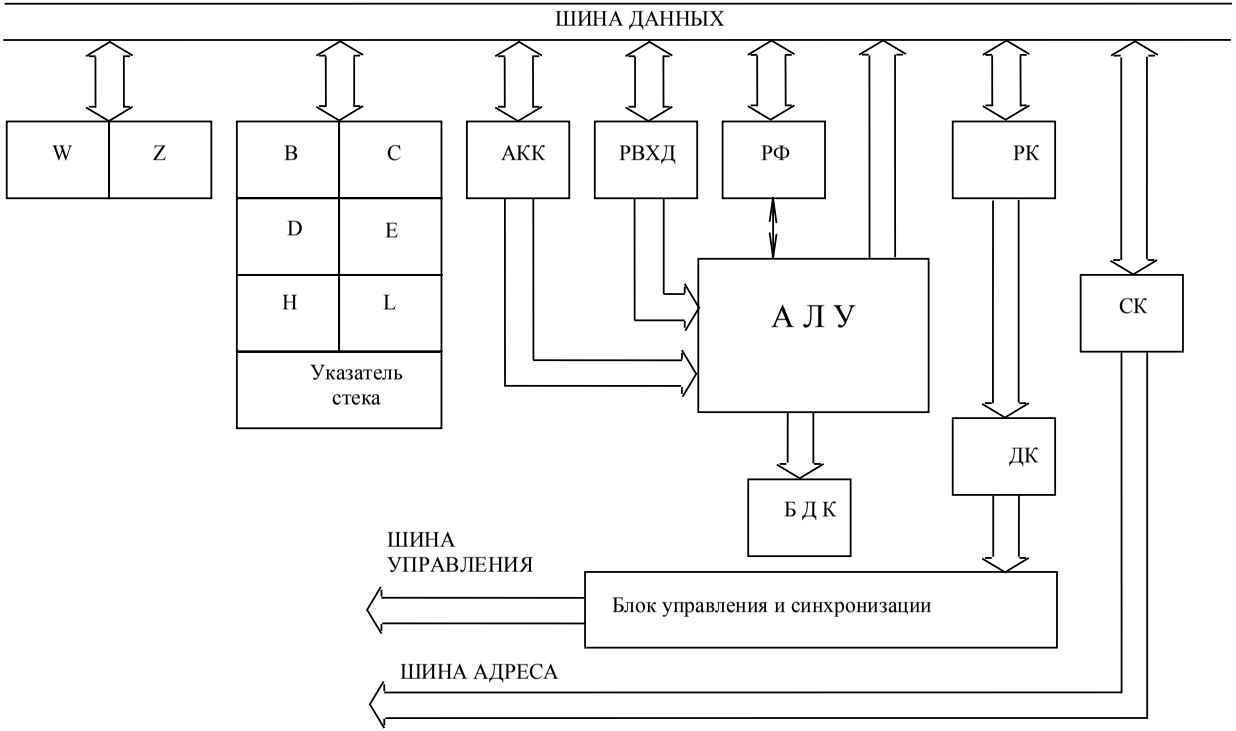


Рис. 2.1. Структурная схема типового центрального процессора

3. *Счетчик команд* – многоразрядный специальный узел ЦП, указывающий адрес следующей выполняемой команды. Программа – последовательность команд, хранимых в памяти и предназначенных для инструктирования ЦП. Для корректного выполнения программы команды должны поступать в строго определенном порядке. На счетчике команд лежит ответственность следить за тем, какая команда выполняется, а какая подлежит выполнению следующей.

Разрядность счетчика команд равна разрядности шины адреса и определяет максимальный объем адресуемой ЦП памяти. Например, при 16-разрядном счетчике команд ЦП может обратиться к 65536 ячейкам памяти.

Когда ЦП начинает работать по команде начальной установки (сброса), в счетчик команд загружается адрес, заданный проектировщиком ЦП, т. е. запуск программы происходит с этого, заранее заданного адреса.

Адрес местоположения первой команды программы выставляется на шине адреса. В результате выбирается одна из ячеек памяти и ее содержимое, через шину данных пересылается в специальный регистр ЦП, называемый регистром команд. После этого действия блок управления ЦП автоматически дает приращение содержимому счетчика команд (инкрементирует его содержимое). Следовательно, начиная с момента, когда ЦП начинает выполнять команду, только что извлеченную из памяти, счетчик команд указывает на адрес следующей выполняемой команды.

Счетчик команд может быть загружен иным содержимым при выполнении особой группы команд, когда возникает необходимость обратится к подпрограмме или выполнить часть программы, которая выпадает из последовательности команд основной программы.

4. *Регистр и дешифратор команд.* Регистр команд предназначен исключительно для хранения текущей выполняемой команды программы, причем эта функция реализуется ЦП автоматически, с началом цикла выборки/выполнения, называемого машинным циклом. Число разрядов регистра команд зависит от типа ЦП и может совпадать с числом разрядов слова данных или быть меньше его.

Дешифратор команд предназначен для декодирования команды, хранящейся в данный момент времени в регистре команд, т. е. в де-

шифраторе команд происходит распознавание команды и формирование сигналов управления для выполнения команды. Сигналы с дешифратора команд поступают на блок управления и синхронизации.

5. *Блок управления и синхронизации*. Блок управления определяет последовательность функционирования всех схем ЦП. По указанию блока управления очередная команда извлекается из регистра команд, поступая на дешифратор команд. При этом определяется, какую операцию необходимо провести с данными, а затем генерируется последовательность действий по выполнению поставленной задачи.

Блок управления и синхронизации имеет доступ к шине управления ЦП и может работать с внешними сигналами, выполняя специальные функции. К таким функциям можно отнести:

– запись или чтение областей памяти;
– запись или чтение периферийных устройств;
– запрос и подтверждение прерываний;
– запрос и подтверждение прямого доступа к памяти.

6. *Регистры общего назначения*. Все ЦП имеют несколько регистров, в которых можно временно хранить обрабатываемые данные. Эти регистры получили название регистров общего назначения (РОН). Обычно функциональные возможности этих регистров не уступают возможностям аккумулятора.

Выбор конкретного регистра для выполнения определенной операции определяется лишь тем, какой из них не занят данными и кажется наиболее удобным. В некоторых типах ЦП два РОН совместно могут выполнять функции регистра двойной длины, который называется регистровой парой. В таком регистре можно хранить данные, разрядность которых превышает разрядность слова ЦП.

7. *Стек и регистр – указатель стека*. При выполнении ЦП подпрограмм, когда требуется ввести, вывести или обработать новые данные, а затем вернуться к выполнению основной программы, возникает необходимость временного сохранения данных основной программы, а также адреса возврата в основную программу. Поскольку количество РОН в ЦП ограничено и они все могут быть занятыми, то при переходе к подпрограмме данные основной программы могут быть разрушены. При возвращении к основной программе из подпрограммы, ЦП не сможет продолжить ее выполнение. Для временного сохранения

операндов счетчика команд и РОН ЦП используется специальная область ОЗУ микропроцессорной системы с особым доступом к ней, называемая стеком.

Стек или память магазинного типа организуется таким образом, что очередная запись данных или адресов всегда осуществляется в вершине (начале) стека. Местоположение стека определяется программистом.

Стек функционирует как память с последовательным доступом по типу LIFO (последний вошел – первый вышел), т. е. данные или адреса должны извлекаться из стека в порядке, обратном по сравнению с загрузкой в него. Каждой команде загрузки в стек позже будет соответствовать команда извлечения из стека в порядке, обратном загрузке.

Данные и адреса можно записать в стек, используя команды PUSH (поместить) или CALL (вызвать), а извлечь из стека командами POP (извлечь) или RET (возврат). Команды записи и извлечения из стека используются всегда совместно, однако между ними могут находится подпрограммы или фрагменты программ.

Для нахождения местоположения вершины (начала) стека в самом ЦП предусмотрен специальный регистр, называемый регистром-указателем стека. В регистре-указателе стека хранится адрес последней записи в стек. Он имеет разрядность, равную разрядности шины адреса ЦП.

Поскольку ЦП загружает в стек или извлекает из него содержимое пары регистров (для этого используются две ячейки ОЗУ), то при записи данных или адресов в стек содержимое регистра-указателя стека уменьшается на два (SP-2), а при считывании операндов из стека – увеличивается на два (SP+2).

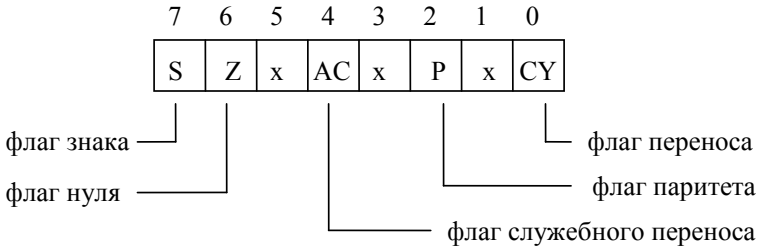
8. *Регистр состояния (флагов)*. Регистр состояния предназначен для хранения признаков поверок результата выполнения операций в АЛУ. Запоминание результатов упомянутых поверок позволяет использовать программы и подпрограммы, содержащие переходы, т. е. нарушение естественного хода выполнения команд.

При наличии в программе перехода выполнение команд начинается с некоторой новой области памяти, т. е. счетчик команд загружается новым числом. В случае условного перехода такое действие имеет место, если результаты определенных поверок совпадают с ожидаемыми значениями.

Регистр состояния представляет программисту возможность организовать работу ЦП так, чтобы при определенных условиях изменялся порядок выполнения команд программы.

Можно сказать, что ЦП принимает решение о том или ином продолжении хода вычислений в зависимости от указанных условий. Таким образом, использование содержимого разрядов регистра состояния приводит к появлению нового набора команд ЦП. Эти команды предназначены для изменения хода выполнения программы в соответствии со значением, принимаемым тем или иным разрядом регистра состояния.

Наиболее часто используются пять разрядов (флагов) регистра состояния.



(пятый, третий и первый биты регистра не используются)

Рис. 2.2. Регистр состояния (флагов)

1. Флаг знака *S* устанавливается в зависимости от состояния наиболее значимого бита (7 разряд) аккумулятора после выполнения арифметических или логических операций в АЛУ. Если число в аккумуляторе положительное, то флаг сбрасывается ($S = 0$), а если отрицательное – флаг устанавливается ($S = 1$).

2. Флаг нуля *Z* устанавливается ($Z = 1$), если результат выполнения операции, помещенный в аккумулятор, равен нулю (во всех разрядах аккумулятора нули), в противном случае – флаг нуля сброшен ($Z = 0$).

3. Флаг служебного переноса *AC* устанавливается, если произошел перенос из третьего разряда аккумулятора в четвертый ($AC = 1$), в противном случае флаг сбрасывается ($AC = 0$). Флаг служебного переноса используется при выполнении операции десятичной коррекции аккумулятора.

4. Флаг четности *P* проверяет число бит единиц в аккумуляторе. Если число единиц в аккумуляторе четное (четный паритет), то флаг четности устанавливается ($P = 1$), в противном случае – флаг четности сбрасывается.

5. Флаг переноса *CY* устанавливается в результате арифметических операций, если произошел перенос из старшего разряда аккумулятора при выполнении операции сложения (т. е. произошло переполнение аккумулятора) или при вычитании, если произошел заем в старший разряд аккумулятора (т. е. вычитаемое больше уменьшаемого). В противном случае флаг переноса сбрасывается ($CY = 0$). Кроме того, флаг переноса используется в командах сдвига данных.

Обычно в ЦП предусмотрены специальные команды для очистки или установки флагов регистра состояния.

9. *Регистры ЦП, доступные программисту*. ЦП – сложное цифровое электронное устройство, причем большая часть его элементов недоступна извне. Например, невозможно увеличить содержимое счетчика команд (PC) с помощью внешнего аппаратного сигнала ЦП. Это осуществимо только программным путем, в результате выполнения специальных команд программы.

Составляя программу для МПС, программист абстрагируется от всего многообразия элементов как ЦП, так и МПС в целом, и имеет дело лишь с системой команд и ограниченным числом программно-доступных регистров.

Эти регистры характеризуются тем, что их имена могут применятся в машинных командах, а содержимое регистров может быть использовано в командах по желанию программиста. Другие узлы ЦП не находят отражения в программах, написанных либо транслированных в машинные коды.

Следовательно, с точки зрения программиста, ЦП представляет собой совокупность программно-доступных регистров, которые каким-либо образом связаны с остальными компонентами процессора с целью выполнения команд программы.

В типовую документацию на ЦП включается структурная схема используемых программистом регистров.

Аккумулятор является главным регистром ЦП. Кроме аккумулятора программный пользователь может использовать регистр флагов.

Аккумулятор и регистр флагов вместе образуют слово – состояние процессора (PSW), причем аккумулятору соответствуют восемь старших разрядов PSW, а регистру флагов – восемь младших.

Регистры общего назначения – B, C, D, E, H, L – являются универсальными регистрами хранения данных. 16-разрядные данные или адреса можно хранить в ЦП в регистровых парах. Шесть регистров общего назначения можно использовать как три регистровые пары (регистр двойной длины): BC, DE и HL. Кроме того, регистровая пара HL может быть использована не только для хранения данных, но и в качестве адресного 16-разрядного адресного регистра при использовании программистом косвенной регистровой адресации данных.

АКК	RF
B	C
D	E
H	L
Счетчик команд PC	
Указатель стека SP	

Рис. 2.3. Регистры ЦП, доступные программисту

Кроме указанных регистров программист имеет доступ к двум специальным регистрам: счетчику команд (PC) и регистру-указателю стека (SP).

Счетчик команд (PC) – 16-разрядный регистр, указывающий адрес следующей выполняемой команды. Пользователь может программным способом изменить содержимое счетчика команд с помощью команд вызова или возврата из подпрограммы, рестарта (повторного пуска с фиксированного адреса), условных или безусловных переходов.

Регистр-указатель стека – 16-разрядный регистр, в котором находится адрес вершины стека, т. е. адрес последней записи в стек. Центральный процессор загружает или извлекает из стека содержимое пары регистров, поэтому содержимое регистра-указателя стека при записи или считывании стека изменяется на два. Данные можно поместить в стек, используя команды CALL (вызвать) или PUSH (поместить), а извлечь – командами RET (возврат) или POP (извлечь).

10. *Блок десятичной коррекции*. Блок десятичной коррекции совместно с АЛУ подвергает преобразованию операнд, находящийся в аккумуляторе, из двоичного в двоично-десятичный формат при выполнении команды DAA (десятичная коррекция аккумулятора). При выполнении этой команды работают два флага регистра состояния: флаг служебного переноса и флаг переноса. Операнд, находящийся в аккумуляторе, разбивается на две тетрады. Если младшая тетрада больше 9 или установлен флаг служебного переноса, то к этой тетраде прибавляется число 6 (т. е. содержимое аккумулятора увеличивается на 06H). Если старшая тетрада больше 9 или установлен флаг переноса, то 6 прибавляется к старшей тетраде (т. е. к содержимому аккумулятора прибавляется 60H).

11. *Регистр временного хранения данных*. Регистр временного хранения данных предназначен для хранения одного из операндов, поступающего в АЛУ при выполнении какой–либо операции. Данные в регистре сохраняются только на момент выполнения машинной команды, а после истечения этого времени – разрушаются или заменяются другим операндом.

Содержание отчета

Отчет должен содержать:

- цель работы;
- краткие теоретические сведения;
- структурную схему однокристального центрального процессора;
- алгоритм работы программы;
- протокол выполнения работы.

Вопросы и задания для самопроверки

1. Каково назначение и работа АЛУ?
2. Для чего предназначены и как функционируют регистр команд, дешифратор команд, блок управления и синхронизации?
3. Каково назначение и работа счетчика команд?
4. Что такое стек, где он расположен, для чего предназначен и какую организацию он имеет?
5. Каково назначение и работа указателя стека?
6. Охарактеризуйте регистры, доступные программисту.
7. Как устроен и для чего предназначен регистр флагов?
8. Охарактеризуйте работу каждого флага.

ЛАБОРАТОРНАЯ РАБОТА 3
Принципы построения микропроцессорных систем

Цель работы – изучить структуру и назначение функциональных узлов базовой микропроцессорной системы и их работу.

Краткие теоретические сведения

1. *Обобщенная структурная схема МПС.* Центральный процессор – центральный блок любой МПС контроля или управления, но не сама система. Отдельно МПС не может выполнить каких-либо полезных функций. Любая электронная система на основе микропроцессора, кроме него самого, должна содержать в своем составе следующие функциональные устройства:

– блок запоминающих устройств;
– устройства ввода и вывода информации;
– устройство поддержки реального времени.

В МПС все устройства реализованы на нескольких кристаллах (ЧИПах). Микропроцессорная электронная система может иметь самую разнообразную конфигурацию аппаратных средств, но в любом случае ЦП может функционировать только совместно с другими составными частями системы.

Необходимость выполнения сложных функций управления внутри самой системы приводит к необходимости использования сложных управляющих устройств, называемых контроллерами. Контроллеры выполняют функции логического анализа состояний сигналов и управления системой, облегчая работу центральному процессору.

Разберем подробнее структурную схему МПС (рис. 3.1).

Центральный процессор – программно-управляющее устройство, предназначенное для обработки цифровой информации и управления процессом этой обработки. В настоящее время принято разделять все ЦП на три основных класса:

1) однокристальные ЦП с фиксированной системой команд и разрядностью слова данных;
2) многокристальные ЦП, у которых логическая структура разбита на функционально законченные части и реализована в виде отдельных БИС;

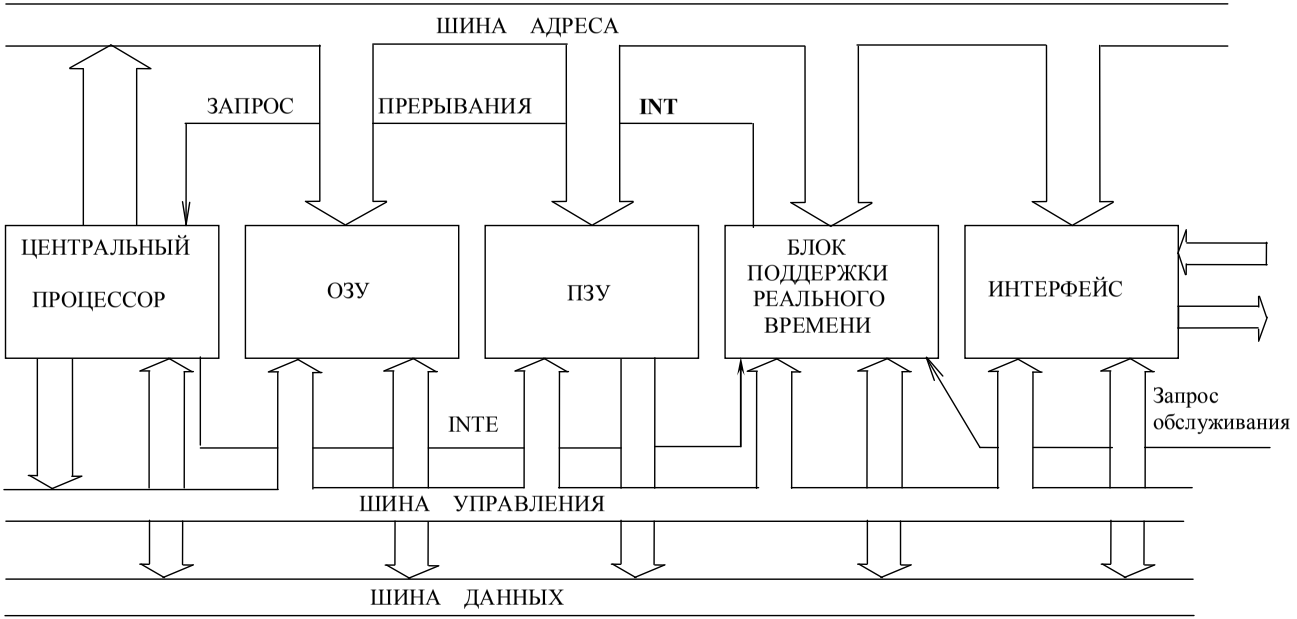


Рис. 3.1. Функциональная схема базовой микропроцессорной системы

3) секционированные ЦП на основе микропроцессорных секций (самостоятельных БИС) с микропрограммным управлением и возможностью расширения разрядности.

При выборе ЦП для конкретного применения необходимо рассмотреть его структуру, технические характеристики и программное обеспечение. Сравнение ЦП следует проводить по следующим параметрам:

– длина слова, на выбор которой влияют точность представления аналоговых сигналов, требуемая точность вычислений, разрядность параллельно принимаемых и выдаваемых данных;
– количество внутренних регистров, наличие стека и возможность многоуровневого прерывания программы;
– быстродействие ЦП;
– гибкость системы команд, возможность применения различных способов адресации данных;
– комплектность, т. е. наличие набора БИС, совместимого с ЦП;
– наличие ассемблера и средств автоматизации проектирования системы, а также отладочных устройств.

2. *Блок запоминающих устройств*. Блок запоминающих устройств МПС включает в себя постоянное запоминающее устройство (ПЗУ) и оперативное запоминающее устройство (ОЗУ). ПЗУ в МПС необходимо для хранения программ функционирования системы и некоторых констант. Хранение изменяемых данных в МПС осуществляется в ОЗУ. В последнем может находится и прикладная программа пользователя.

Объем и тип памяти определяется прежде всего теми задачами, которые решает МПС. Чем сложнее и разностороннее функции, возлагаемые на МПС, тем больше необходим объем памяти и шире ее состав. ЦП может выбирать любое запоминающее устройство программным способом, используя для этого шину адреса. Однако местоположение областей памяти МПС определяется конструктором системы аппаратным способом.

При использовании адресной шины каждая ячейка памяти имеет свой собственный однозначный адрес. Размер адресного пространства определяется максимальной разрядностью шины адреса. Запись в память или считывание из нее происходит при наличии доступа к памяти. Почти всегда память МПС выполняется с произвольным доступом, когда данные могут быть записаны или считаны из любой ячейки памяти за время, называемое временем доступа к памяти.

В МПС может быть использовано масочное ПЗУ, однократно программируемое ПЗУ, репрограммируемое ПЗУ с электрической записью и стиранием информации, репрограммируемое ПЗУ с электрической записью и ультрафиолетовым стиранием информации.

Конкретный выбор типа ПЗУ для МПС определяется назначением системы и совместимостью по уровням логических сигналов с ЦП.

Для считывания информации из ПЗУ, кроме подачи комбинации адресных сигналов, необходимо на ПЗУ подать сигнал управления, называемый выбором кристалла CS. Если уровень сигнала выбора кристалла низкий, то происходит разблокирование выходных формирователей ПЗУ и матрица-накопитель получает доступ к выходным линиям микросхемы. Когда же приходит сигнал высокого уровня на вход CS, информационные выходы приобретают высокое (полное) сопротивление, т. е. выходы микросхемы имеют высокий импеданс. Наличие выходов с тремя состояниями (ноль, единица и высокий импеданс) позволяет параллельно связывать линии данных многих запоминающих устройств вместе и выбирать конкретное устройство посредством подачи на его вход CS сигнала выбора кристалла, имеющего низкий уровень.

В МПС может быть использовано как статическое, так и динамическое ОЗУ. При использовании последнего необходимо в МПС предусмотреть установку схем регенерации данных ОЗУ. Поскольку линии данных ОЗУ являются двунаправленными (информация может быть и записана, и считана), то каждая микросхема ОЗУ имеет дополнительную линию управления – «запись/чтение» WR/RD. Для запоминания информации в ОЗУ выбирается адрес нужной ячейки памяти, данные пересылаются по шине данных, линия управления WR/RD переводится в состояние логического нуля, после этого поступает сигнал выбора кристалла CS нужной микросхемы ОЗУ и данные запоминаются.

Состояние линии управления «запись/чтение» определяет направление потока данных. Следует отметить, что вход микросхемы ОЗУ «запись/чтение» не работает до тех пор, пока не будет подан сигнал выбора кристалла.

3. *Устройство поддержки реального времени* осуществляет согласование работы МПС с объектом управления в режиме реального времени. Для определения требований к построению системных устройств поддержки реального времени рассмотрим особенности функциониро-

вания схем прерывания работы ЦП. Процесс прерывания работы ЦП сигналами запроса обслуживания внешних устройств (различных датчиков, исполнительных механизмов и т. д.) устраняет необходимость выполнения ЦП неэффективных операций по проверке готовности внешних устройств к обмену информацией и существенно снижает затраты времени на ожидание их готовности.

Реализация механизма прерываний имеет большую важность в случае необходимости обмена информацией с большим числом асинхронно работающих внешних устройств. Все ЦП имеют специальный вход управления – запрос прерывания INT, сигнал на который поступает от внешнего устройства на прерывание выполнения основной программы и переход на подпрограмму обслуживания устройства, запросившего прерывание. Кроме того, ЦП имеет специальный выход управления – подтверждение прерывания INTE, наличие сигнала на нем показывает готовность ЦП обслужить устройство, запросившее прерывание.

Момент возникновения запроса прерывания не связан с работой ЦП по основной программе или какой-либо подпрограмме, а только с готовностью внешнего устройства. Поэтому сигналы прерывания текущей программы асинхронны относительно циклов работы ЦП.

В случае если МПС имеет только одно внешнее устройство, работающее по сигналам прерывания, то устройство поддержки реального времени включает в себя только таймер меток реального времени, задающий точные временные интервалы работы системы, а сигнал готовности внешнего устройства подается сразу на вход INT ЦП.

Получив сигнал запроса прерывания INT, ЦП завершает выполнение текущей команды основной программы и приступает к удовлетворению запроса, т. е. переходит на подпрограмму обслуживания внешнего устройства, имеющую фиксированный адрес (для INTEL 8080 – 0038H, что соответствует команде RST 7). При переходе на подпрограмму обслуживания внешнего устройства автоматически производится запись адреса возврата прерванной программы в стек (запись ведется в вершину стека). Регистр-указатель стека (находящийся внутри ЦП) уменьшает свое содержимое на два. Далее программист может сохранить контекстное состояние рабочих регистров (на момент прерывания основной программы) в стеке. При каждой записи в стек содержимое регистра-указателя стека будет уменьшено на два. После этих операций

следует «тело подпрограммы» обслуживания внешнего устройства, где данные считываются с датчика, подвергаются обработке или выводятся на исполнительное устройство и т. д. Затем программист должен восстановить контекст основной прерванной программы (в порядке, обратном записи в стек) и по команде RET (возврат из подпрограммы) в счетчик команд (расположенный внутри ЦП) из стека заносится адрес возврата из прерывания, указывающий на команду, которая должна выполнятся первой в прерванной программе.

В любой МПС, имеющей несколько внешних устройств, работающих по прерываниям, возникают сложности управления при одновременном поступлении на ЦП запросов от нескольких устройств. В этом случае устройство поддержки реального времени включает в себя не только таймер меток реального времени, но и специальный логический узел – программируемый контроллер прерываний, на входы которого подаются сигналы запроса обслуживания от внешних устройств. Сам контроллер имеет выход INT, соединяемый со входом INT ЦП, и вход INTE, соединяемый с выходом INTE ЦП.

С помощью контроллера прерываний программист может назначить начальный адрес каждой из подпрограмм обслуживания прерывания каждого внешнего устройства, выбрать метод прерывания, назначить приоритет обслуживания каждому внешнему устройству, маскировать поступившие запросы на прерывание от конкретного внешнего устройства. Все эти операции выполняются только программным способом.

В МПС могут применяться два метода прерывания: прерывание по опросу и прерывание по вектору. В первом случае осуществляется последовательный опрос внешних устройств (на предмет готовности какого-либо из них к информационному обмену), до тех пор пока не будет обнаружено то, которое запросило прерывание. При наличии сигнала прерывания от одного из устройств, ему присваивается наивысший уровень приоритета обслуживания. Другим внешним устройствам назначаются более низкие приоритеты обслуживания в соответствии с местом внешнего устройства в последовательности опроса контроллером прерывания.

В случае прерывания по вектору, программист с помощью контроллера прерывания присваивает всем внешним устройствам определен-

ные уровни приоритета обслуживания. Если в момент обслуживания внешнего устройства с низким уровнем приоритета на контроллер прерывания придет запрос от устройства с более высоким уровнем приоритета, то подпрограмма обслуживания первого устройства сама будет прервана подпрограммой второго устройства.

При поступлении запроса на обслуживание от внешнего устройства на контроллер прерывания, последний определяет его приоритет обслуживания и начальный адрес подпрограммы обслуживания.

Далее контроллер прерывания выдает на ЦП аппаратный сигнал – запрос прерывания INT. При получении этого сигнала ЦП завершает выполнение очередной команды основной программы и выдает обратно на контроллер прерывания сигнал подтверждения прерывания INTE. В момент выдачи этого сигнала сам ЦП переходит в режим чтения шины данных. При получении контроллером прерывания сигнала INTE, он на шину данных выдает трехбайтную команду вызова подпрограммы обслуживания именно того устройства, которое запросило прерывание (CALL Addr). ЦП побайтно прочитывает данную команду и переходит к выполнению этой подпрограммы, а после команды RET (возврат из подпрограммы) возвращается в основную программу.

При выполнении ЦП подпрограммы обслуживания внешнего устройства контроллер прерывания блокирует обслуживание всех запросов, имеющих одинаковый или более низкий по сравнению с обслуживаемым уровень приоритета. Более высокий уровень приоритета может прервать обслуживание, если в подпрограмме не было команды запрета прерывания. Запретить обслуживание запросов прерывания можно также использованием процедуры маскирования запросов программным способом.

4. *Устройства ввода-вывода информации в МПС.* МПС вступает во взаимодействие с внешней средой с помощью периферийных устройств. Каждое устройство ввода-вывода для подключения к МПС имеет свою схему сопряжения, называемую интерфейсом.

Интерфейс – совокупность правил, устанавливающих единые принципы обмена информацией устройств в МПС. В состав интерфейса входят аппаратные средства соединения устройств, а также программное обеспечение, осуществляющее поддержку протокола обмена данными.

По функциональному назначению интерфейсы подразделяются на: магистральные (внутримашинные, предназначенные для скоростной передачи параллельной информации внутри МПС), системные (предназначенные для последовательной связи ЭВМ и различных МПС с локальными сетями), внешние интерфейсы периферийных устройств (используемые для связи МПС с датчиками и исполнительными устройствами).

В интерфейсах используются два принципа обмена информацией: с передачей в последовательном или параллельном формате, причем последовательная передача информации является более предпочтительней (по причине упрощения линии связи), если при этом обеспечивается необходимая скорость передачи данных.

Для внешних интерфейсов периферийных устройств существуют три метода ввода/вывода, обеспечивающих взаимодействие МПС с внешними устройствами:

1) программируемый ввод–вывод информации;
2) ввод-вывод по сигналам прерывания программы;
3) ввод-вывод с прямым доступом к памяти.

Программируемый ввод-вывод осуществляется по инициативе ЦП, а обмен данными с внешним устройством происходит под управлением ЦП, который выполняет соответствующую программу. В этом методе обмена ЦП программным путем должен определить готовность внешнего устройства к выполнению операции ввода-вывода, до того как начнется программная передача информации. Внешнее устройство должно иметь аппаратные средства для представления информации о готовности к обмену. ЦП считывает эту информацию, анализирует ее и на основе анализа принимает решение о возможности информационного обмена между МПС и внешним устройством. В МПС со множеством интерфейсов рассмотренный метод ввода-вывода не эффективен, так как на чтение и анализ информации о готовности внешних устройств ЦП тратит большое чисто циклов работы, которые, по существу, будут потеряны.

При вводе-выводе по сигналам прерывания программы инициатива процедуры обмена принадлежит внешнему устройству, которое выдает на ЦП сигнал запроса прерывания. Последний, в ответ на запрос, завершает выполнение текущей команды основной программы и пере-

ходит на подпрограмму обработки прерывания с целью осуществления информационного обмена МПС с внешним устройством.

Ввод-вывод с прямым доступом к памяти производится по инициативе внешнего устройства, но пересылка информации между внешним устройством и памятью (ОЗУ) МПС производится не через ЦП, а напрямую. Для обеспечения такого метода ввода-вывода в состав МПС должно быть включено устройство, позволяющее организовать канал прямого доступа к памяти, альтернативный программной передаче данных через ЦП. Такое устройство называется «программируемый контроллер прямого доступа». При использовании этого метода значительно сокращается время обмена информацией, а сам он используется при пересылке массивов данных.

5. *Шинная структура соединений в МПС*. ЦП связан с другими узлами МПС с помощью трех шин: адресной, данных и управления.

Шина – группа проводов, обеспечивающая параллельное соединение узлов МПС. Использование шинной структуры соединений в МПС позволяет существенно сократить схему соединений внутри системы.

Адресная шина используется для идентификации областей памяти и периферийных устройств МПС. Шина адреса (ША) является однонаправленной. ЦП, являясь главным элементом системы, выставляет на ША адрес, обращаясь к любому другому элементу системы. В обратном направлении передача запрещена.

Разрядность ША определяет объем памяти, с которым способен работать ЦП. Так, при 16-разрядной ША, ЦП может обратится к $2^{16} = 65536$ ячейкам памяти.

Шина данных (ШД) соединяет параллельно все устройства МПС с процессором. По этой шине в любой определенный момент времени передаются данные в ЦП от периферийных устройств или в обратном направлении. В отличие от адресной шины ШД является двунаправленной.

Шина управления (ШУ) позволяет ЦП управлять работой других устройств в МПС. По этой шине ЦП сообщает периферийным устройствам, когда передавать данные или когда их получать с ШД. Главное отличие ШУ от ША и ШД состоит в том, что каждая ее линия выполняет свою собственную единственную функцию. К таким функциям относятся операции: запись-чтение памяти, запись-чтение периферийных

кристаллов, запрос и подтверждение прерывания, запрос и подтверждение прямого доступа к памяти.

Все шины МПС являются трехстабильными, т. е. имеющими три состояния (передача нуля, передача единицы и высокий импеданс). Шина с тремя состояниями похожа по своему назначению на групповую телефонную линию, к которой подсоединяются несколько абонентов. Для работы в каждый момент времени ЦП инициирует работу только одного источника информации. Если бы в этот момент были разблокированы сразу несколько источников, информация, передаваемая по ШД, потеряла бы всякий смысл. Как только какой-либо из источников информации разблокируется, данные с его выхода пересылаются на шину. Все остальные источники информации заблокированы. Их выходы находятся в состоянии, характеризуемом высоким импедансом (работают на холостом ходу), и поэтому они не оказывают никакого влияния на логическое состояние шины.

Содержание отчета

Отчет должен содержать:

- цель работы;
- краткие теоретические сведения;
- структурную схему базовой микропроцессорной системы;
- алгоритм работы программы;
- программное обеспечение;
- протокол выполнения работы.

Вопросы и задания для самопроверки

1. Каково назначение основных элементов базовой микропроцессорной системы?
2. Охарактеризуйте назначение и работу блока запоминающих устройств.
3. Охарактеризуйте назначение и работу блока поддержки реального времени.
4. Охарактеризуйте назначение и работу блока ввода-вывода информации в МПС.
5. Для чего предназначена и как функционирует шинная структура в микропроцессорной системе?

ЛАБОРАТОРНАЯ РАБОТА 4
Способы адресации данных однокристального микропроцессора. Состав команд передачи данных

Цель работы – изучить способы адресации данных типового центрального процессора и команды передачи данных, входящие в систему команд типового центрального процессора.

Краткие теоретические сведения

Машинным языком называется язык программирования, предложения которого выражают машинные команды. Самый веский аргумент в пользу машинного языка состоит в том, что это единственный язык, непосредственно воспринимаемый ЦП. Каждая команда программы, написанная на машинном языке, начинается с кода операции процессора (КОП), указывающего на необходимое действие. Все КОП содержат 8 бит информации. Однако команды ЦП могут быть одно-, двух- или трехбайтными.

Рис. 4.1. Форматы команд центрального процессора

Команды ЦП – двоичные числа. Но даже однобайтное двоичное число трудно запомнить. Еще труднее оперировать двоичными числами команд длиной в 2–3 байта. Для сокращения длинной записи нулей и единиц в двоичной системе можно было бы воспользоваться шестнадцатеричными числами при написании команд. Однако и в этом случае остается неразрешенной основная проблема: непонятно, что означает каждая команда, записанная в подобной форме. Данная проблема решается путем применения мнемонического обозначения – сокращенной записи названия команд (обычно это несколько начальных букв английского названия операции).

Для написания программы для конкретного типа ЦП требуется специальный машинно-зависимый язык, отражающий аппаратурные

особенности этого типа процессора, в частности, состав программно-доступных регистров ЦП. Этот язык носит название языка ассемблера. Язык ассемблера – это не какой-то один конкретный язык, а целый класс языков. Каждый ЦП имеет свой машинный язык и, следовательно, язык ассемблера, разрабатываемый изготовителем ЦП.

Написание программ на языке ассемблера обычно проводится на четырех полях: метка, мнемоника команды, операнд команды, комментарий к программе.

МЕТКА	МНЕМОНИКА	ОПЕРАНД	КОММЕНТАРИЙ
START:	MVI	A, 00H	; Очистить Акк.
LOOP:	INR	A	; A <–– A + 1
	CPI	0AH	;Сравнить A с числом 10
	JZ	STOP	;Закончить, если A=10
	JMP	LOOP	; Повторить цикл
STOP:	HLT		; Остановить ЦП

Метка в ассемблере выполняет функцию номера сроки. Для идентификации местоположения метки используется двоеточие (:).

Метка присваивается строке в том случае, когда в программе имеется несколько команд, обращающихся к данной строке, и позволяет легко идентифицировать строку, к которой требуется перейти в ходе выполнения программы.

Поле мнемоники содержит точную мнемонику, установленную разработчиком ЦП, которая указывает программе ассемблера операцию для выполнения.

Поле операнда содержит информацию о регистрах, адресах и данных, объединенных соответствующей операцией. Используя информацию трех рассмотренных полей, ассемблер может выдать соответствующий код на машинном языке.

Поле комментария не учитывается ассемблером. Его начало обозначается в программе точкой с запятой (;). Это поле в ассемблере очень важно, поскольку позволяет понять события в программе.

Способы адресации

Тип обращения (адресации) к данным принято называть способом адресации.

Способы адресации типового ЦП подразделяются на внутреннюю и внешнюю адресацию. Внутренняя адресация касается операндов, расположенных во внутренних регистрах самого ЦП. Внешняя адресация касается операндов, расположенных вне ЦП, т. е. в ячейках памяти или портах устройств ввода-вывода.

К внутренней адресации относятся неявный и регистровый способы адресации.

Неявный способ адресации – безадресный способ задания системы команд. Адрес источника и приемника команды в этом способе адресации указаны в команде неявно, т. е. они как бы встроены в команду.

К неявному способу адресации относятся только однобайтные команды ЦП, дополнительные данные для выполнения которых не нужны. Команды с неявным способом адресации отличаются наибольшей величиной быстродействия, поскольку однобайтные команды выполняются быстрее любых других. Примером команд неявной адресации могут служить команды:

CMA – инверсия содержимого аккумулятора;

NOP – отсутствие операции.

Данные команды по формату являются однобайтными и дополнительные данные для их выполнения не требуются. Все действия по выполнению команды происходят в самом ЦП.

К регистровой адресации относятся однобайтные команды ЦП, не требующие операндов вне ЦП. При этом способе адресации операнды отыскиваются во внутренних регистрах ЦП, а все действия по выполнению команды происходят в нем самом. Адрес источника и приемника данных указывается при этом в «теле» команды. Примером команд регистрового способа адресации могут служить команды:

MOV A,D – переписать содержимое регистра *D* в аккумулятор;

ADD D – сложить содержимое регистра *D* с аккумулятором.

К внешней адресации относятся: прямой, непосредственный и косвенный регистровый способы адресации.

К прямой адресации относятся команды ЦП, второй или второй и третий байт которых прямо указывает на адрес операнда. Первый байт команды предназначен для кода операции процессора, а второй и, если имеется, третий – для адреса.

Адрес в этом случае указывает ячейку памяти или порт ввода-вывода, в котором находятся подлежащие обработке данные. Примером команд прямого способа адресации могут служить команды:

OUT 80H – передать данные из ЦП в порт вывода с адресом 80Н;

LDA 1000H – загрузить аккумулятор данными из ячейки памяти с адресом 1000Н.

Прямую адресацию целесообразно использовать в том случае, когда необходимо разместить данные в любой из областей памяти либо произвести обмен данными с устройствами ввода-вывода.

К непосредственной адресации относятся двух- или трехбайтные команды, в которых операнд (данные) поступают в программную память из второго или второго и третьего байтов. В первом байте команды находится код операции процессора, а второй и третий байты отводятся под данные. Таким образом, данные, подлежащие обработке (пересылке), находятся в самом «теле» команды. Примером команд непосредственного способа адресации могут служить команды:

ADI 80H – сложить содержимое аккумулятора с числом 80Н;

LXI H, 1000H – загрузить в регистровую пару HL число 1000Н.

Команды с косвенным регистровым способом адресации – группа команд, имеющих формат, равный одному байту. В коде операции такой команды указывается регистровая пара, содержимое которой есть адрес местоположения данных в памяти. При использовании этого способа адресации работают регистровые пары BC, DE, HL.

Косвенная адресация удобна при обращении к часто используемым областям памяти и особенно в тех случаях, когда данные организованы в виде набора или же таблицы. Примером команд косвенной регистровой адресации могут служить команды:

ADD M – сложить содержимое аккумулятора с данными, находящимися в ячейке памяти с адресом, который расположен в регистровой паре HL;

STAX B – разместить данные, находящиеся в аккумуляторе, в ячейку памяти, адрес которой расположен в регистровой паре BC.

Команды передачи данных

По функциональному назначению машинные команды каждого ЦП можно разделить на следующие группы: команды передачи дан-

ных, арифметические команды, логические команды и команды сдвига данных, команды ветвления программ и связи с подпрограммами, специальные команды управления и системные команды.

Команды передачи данных служат для пересылки данных из регистра в регистр, размещения данных в памяти МПС, размещения извлеченных из памяти данных в устройствах ввода-вывода. Во всех командах передачи данных указывается адрес источника и приемника информации. Команды передачи данных следовало бы назвать командами копирования, потому что они практически осуществляют перемещение именно копии данных. Так, например, одна из этих команд перемещает данные из аккумулятора в область памяти. После выполнения данной команды и в области памяти, и в аккумуляторе находятся одни и те же данные. Очень редко в ходе выполнения команды пересылки данных разрушаются данные, находящиеся в исходном месте.

Все команды передачи данных не влияют на установку флагов в регистре состояния ЦП.

Команды передачи данных типового ЦП сведены в таблицу и записаны в общем виде. В таблице использованы следующие обозначения:

r, r1, r2 – регистр;	rp – регистровая пара;
M – ячейка памяти;	data – 8-разрядные данные;
SP – указатель стека;	data16 – 16-разрядные данные;
PC – счетчик команд;	port – адрес порта ввода-вывода;
S – флаг знака;	adr – адрес ячейки памяти;
Z – флаг нуля;	PSW – слово – состояние процессора;
P – флаг паритета;	CY – флаг переноса.

Содержание отчета

Отчет должен содержать:

- цель работы;
- записи, предназначенные для пояснения материала по: неявному, регистровому, прямому, непосредственному и косвенному регистровому способам адресации;
- таблицу команд передачи данных типового ЦП;
- решение задач на ассемблере, выданных преподавателем.

Таблица 4.1

Команды передачи данных

Мнемоника	Операция	Символика	Формат	Флаги	Адресация
MOV r2,r1	регистровая передача	r1 → r2	КОП	НЕТ	регистровая
MOV r, M	загрузка из ячейки памяти в регистр	M(HL) → r	КОП	НЕТ	косвенная регистровая
MOV M, r	загрузка из регистра в ячейку памяти	r → M(HL)	КОП	НЕТ	косвенная регистровая
MVI M, data	непосредственная передача данных	data → M(HL)	КОП data	НЕТ	непосредственная косвенная регистровая
MVI r, data	непосредственная загрузка регистра	data → r	КОП data	НЕТ	непосредственная
LXI rp, data16 (BC,DE,HL)	непосредственная загрузка регистра пары	data16 → rp	КОП, мл, ст. байт	НЕТ	непосредственная
LDA adr	прямая загрузка аккумулятора из памяти	M(adr) → A	КОП, мл, ст. байт	НЕТ	прямая
STA adr	прямое размещение данных в памяти	A → M(adr)	КОП, мл, ст. байт	НЕТ	прямая
LDAX rp (BC,DE)	косвенная загрузка аккумулятора из памяти	M(rp) → A	КОП	НЕТ	косвенная регистровая
STAX rp (BC,DE)	косвенное размещение аккумулятора	A → M(rp)	КОП	НЕТ	косвенная регистровая
LHLD adr (HL)	прямая загрузка регистра пары HL из памяти	M(adr) → L, M(adr + 1) → H	КОП, мл, ст. байт	НЕТ	прямая
SHLD adr (HL)	прямое размещение регистра пары HL в памяти	L → M(adr), H → M(adr+1)	КОП, мл, ст. байт	НЕТ	прямая

Мнемоника	Операция	Символика	Формат	Флаги	Адресация
IN port	ввод данных из порта ввода	Port (data) → A	КОП port	НЕТ	прямая
OUT port	вывод данных в порт вывода	A → port (data)	КОП port	НЕТ	прямая
XCHG	обмен регистра пар HL и DE	HL ↔ DE	КОП	НЕТ	регистровая
XTHL	обмен вершины стека с регистром парой HL	[SP] ↔ HL	КОП	НЕТ	регистровая
SPHL	передача регистра пары HL в указатель стека	HL → [(SP)]	КОП	НЕТ	регистровая
PCHL	передача регистра пары HL в счетчик команд	HL → (PC)	КОП	НЕТ	регистровая
PUSH PSW	поместить в стек аккумулятор и регистр флагов	A → ((SP)–1), rF → ((SP)–2), [(SP)–2] → [(SP)]	КОП	НЕТ	косвенная регистровая
PUSH rp (BC,DE,HL)	поместить в стек регистра пару	H → ((SP)–1), L → ((SP)–2), [(SP)–2] → [(SP)]	КОП	НЕТ	косвенная регистровая
POP PSW	извлечь из стека аккумулятор и регистр флагов	((SP)) → rF,((SP)+1) → A [(SP)+2] → [(SP)]	КОП	НЕТ	косвенная регистровая
POP rp (BC,DE,HL)	извлечь из стека регистра пару	((SP)) → rl, ((SP)+1) → rp [(SP)+2] → [(SP)]	КОП	НЕТ	косвенная регистровая

Вопросы и задания для самопроверки

1. Запишите не менее трех команд, относящихся к неявному способу адресации.
2. Запишите не менее трех команд, относящихся к регистровому способу адресации.
3. Запишите не менее трех команд, относящихся к прямому способу адресации.
4. Запишите не менее трех команд, относящихся к непосредственному способу адресации.
5. Запишите не менее трех команд, относящихся к косвенному регистровому способу адресации.
6. Поясните действия при выполнении следующих команд:
   MOV B, H;
   LXI D, 0813H;
   LDA 0813H;
   LHLD 0813H;
   OUT 64H;
   MVI M, 87H.
7. Каким образом ведется работа со стеком? Какова последовательность помещения и извлечения в/из стека? Поясните действия при выполнении команд PUSH и POP.
8. Напишите программу в следующих полях: мнемоника, операнд, комментарий. Условие: введите данные, расположенные в порте с адресом 77H, и разместите их в *rD, rE*, в ячейке памяти с адресом 0913H и в стеке.

ЛАБОРАТОРНАЯ РАБОТА 5
Состав команд арифметических действий микропроцессора

Цель работы – изучить арифметические команды, входящие в систему команд типового центрального процессора.

Краткие теоретические сведения

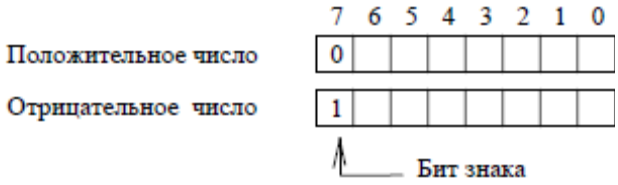
В состав команд арифметических действий ЦП входят команды: сложения, вычитания, инкремента и декремента операнда. Для сложения многобайтных чисел используют специальные арифметические команды, учитывающие флаг переноса регистра состояния. Арифметические команды оперируют с данными в памяти и регистрах. Во всех случаях, кроме указанных исключений, устанавливаются индикаторы нуля *Z*, знака *S*, четности *P*, переноса *CY* и служебного переноса *AC*.

В командах сложения (за исключением сложения регистровых пар) аккумулятор содержит одно из слагаемых. Местонахождение второго точно оговаривается в коде операции. Результат сложения помещается обратно в аккумулятор, замещая первое слагаемое.

Все операции вычитания проводят с использованием дополнительного кода (автоматически), устанавливают 1 в индикатор переноса для указания переноса при займе и сбрасывают его для указания отсутствия переноса.

Если необходимо использовать двоичные числа со знаком, то в этом случае используют дополнительный код.

При этом старший бит в байте отвечает за знак числа. Если число положительное, то старший бит равен нулю. Если отрицательное – единице.



Для перевода числа в дополнительный код или из дополнительного кода необходимо получить обратный код, т. е. проинвертировать число (заменить все нули единицами, а единицы – нулями), затем прибавить к результату единицу в младший (нулевой) бит.

Перевод в дополнительный код.

1. $(14)_{10} = (0E)_{16} =$ 0000.1110_2
2. обратный код $\rightarrow$ 1111.0001_2
3. $+1111.0001$
   0000.0001

дополнительный код $\rightarrow$ $1111.0010_2 = (F2)_{16}$

Перевод из дополнительного кода.

1. $(F0)_{16} =$ 1111.0000_2
2. обратный код $\rightarrow$ 0000.1111_2
3. $+0000.1111$
   0000.0001

$0001.0000_2 = -16_{10}$

Команды арифметических действий типового ЦП сведены в табл. 5.1.

Содержание отчета

Отчет должен содержать:

- цель работы;
- таблицу команд арифметических действий типового ЦП;
- решение задач на ассемблере, выданных преподавателем.

Вопросы и задания для самопроверки

1. Какая из команд инкремента не влияет на индикатор нуля?
2. Определите результат действия операции (для аккумулятора и регистра флагов) *DCR A*, если (A) = 00H.
3. Что произойдет с флагом переноса по команде *SUB L*, если (L) = 02H, (A) = 04H, (CY) = 1.
4. Определите результат выполнения команд:
   1) *INR C*, если (C) = 99H;
   2) *ADD D*, если (A) = 6CH, (D) = 2EH;
   3) *ADC C*, если (A) = 42H, (C) = 3DH, (CY) = 0;
   4) *SUB A*, если (A) = 3EH.

Таблица 5.1

Команды арифметических операций

Мнемоника	Операция	Символика	Формат	Флаги	Адресация
ADD r	сложение регистра с аккумулятором	A + r → A	S, Z, AC, P, CY	КОП	регистровая
ADD M	сложение ячейки памяти с аккумулятором	A + M(HL) → A	S, Z, AC, P, CY	КОП	косвенная регистровая
ADI data	непосредственное сложение	A + data → A	S, Z, AC, P, CY	КОП data	непосредственная
ADC r	сложение регистра с аккумулятором с переносом	A + r + CY →A	S, Z, AC, P, CY	КОП	регистровая
ADC M	сложение ячейки памяти с аккумулятором с переносом	A + M(HL) + CY → A	S, Z, AC, P, CY	КОП	косвенная регистровая
ACI data	непосредственное сложение с переносом	A + data + CY → A	S, Z, AC, P, CY	КОП data	непосредственная
INR r	инкрементирование содержимого регистра	r + 1 → r	все, кроме CY	КОП	регистровая
INR M	инкрементирование содержимого ячейки памяти	M(HL) + 1 → M(HL)	все, кроме CY	КОП	косвенная регистровая
INX rp (BC,DE,HL)	инкрементирование содержимого регистровой пары	rp + 1 → rp	никаких	КОП	регистровая
DAD rp (BC,DE,HL)	сложение регистровых пар	HL + rp → HL	только CY	КОП	регистровая
SUB r	вычитание регистра из аккумулятора	A – r → A	S, Z, AC, P, CY	КОП	регистровая
SUB M	вычитание ячейки памяти из аккумулятора	A – M(HL) → A	S, Z, AC, P, CY	КОП	косвенная регистровая
SUI data	непосредственное вычитание	A – data → A	S, Z, AC, P, CY	КОП data	непосредственная

Мнемоника	Операция	Символика	Формат	Флаги	Адресация
SBB r	вычитание регистра из аккумулятора с переносом	A – r – CY → A	S, Z, AC, P, CY	КОП	регистровая
SBB M	вычитание ячейки памяти из аккумулятора с переносом	A – M(HL) – CY → A	S, Z, AC, P, CY	КОП	косвенная регистровая
SBB data	непосредственное вычитание с переносом	A – data – CY → A	S, Z, AC, P, CY	КОП data	непосредственная
DCR r	декрементирование содержимого регистра	r – 1 → r	все, кроме CY	КОП	регистровая
DCR M	декрементирование содержимого ячейки памяти	M(HL) – 1 → M(HL)	все, кроме CY	КОП	косвенная регистровая
DCX rp (BC,DE,HL)	декрементирование содержимого регистровой пары	rp – 1 → rp	никаких	КОП	регистровая
DAA	десятичная коррекция аккумулятора	Число в аккумуляторе разбивается на 2 тетрады. Если мл. тетрада больше 9 или установлен флаг AC, к аккум. прибавляется 06H. Если ст. тетрада больше 9 или установлен флаг CY, к аккум. прибавляется 60H	S,Z,AC,P,CY	КОП	специальная

ЛАБОРАТОРНАЯ РАБОТА 6
Состав команд логических операций типового микропроцессора

Цель работы – изучить логические команды, входящие в систему команд типового центрального процессора.

Краткие теоретические сведения

Задачей команд логических действий является выполнение логических операций с обрабатываемыми данными. Типовой ЦП содержит в составе команд логические операции: И, ИЛИ, Исключающее ИЛИ, Отрицание и Сдвиг.

Все логические операции выполняются побитно. Логические операции выполняются всегда над содержимым аккумулятора и другим байтом из регистра или из памяти. Результат операции загружается в аккумулятор.

К логическим операциям относят команды сдвига содержимого аккумулятора с учетом или без учета флага переноса *CY* регистра состояния. Напомним, что сдвиг числа на один бит вправо/влево эквивалентен делению/умножению числа на два.

Используя одну или несколько команд сдвига, можно тестировать весь заданный байт побитно. Тест паритета является другим приложением использования команд сдвига. Паритет двоичного числа определяется числом содержащихся в нем единиц: четный паритет – общее четное число единиц; нечетный паритет – общее нечетное число единиц.

Таким образом, команды логических операций используются для манипуляции с переменными по законам Булевой алгебры. Они могут быть использованы для тестирования и сравнения бит.

Команды логических операций типового ЦП сведены в табл. 6.1.

Содержание отчета

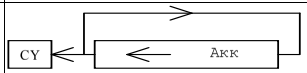
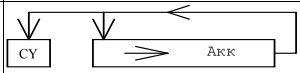
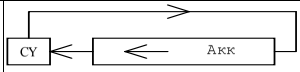
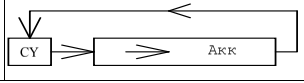
Отчет должен содержать:

- цель работы;
- таблицу команд логических операций типового ЦП;
- решение задач на ассемблере, выданных преподавателем.

Таблица 6.1

Команды логических операций

Мнемоника	Операция	Символика	Формат	Флаги	Адресация
ANA r	логическая операция «И» между аккумулятором и регистром	A ∧ r → A	ВСЕ, но AC = 1, CY = 0	КОП	регистровая
ANA M	логическая операция «И» между аккумулятором и ячейкой памяти	A ∧ M(HL) → A	ВСЕ, но AC = 1, CY = 0	КОП	косвенная регистровая
ANI data	непосредственная логическая операция «И» между аккумулятором и данными	A ∧ data → A	ВСЕ, но AC = 0, CY = 0	КОП data	непосредственная
ORA r	логическая операция «ИЛИ» между аккумулятором и регистром	A ∨ r → A	ВСЕ, но AC = 0, CY = 0	КОП	регистровая
ORA M	логическая операция «ИЛИ» между аккумулятором и ячейкой памяти	A ∨ M(HL) → A	ВСЕ, но AC = 0, CY = 0	КОП	косвенная регистровая
ORI data	непосредственная логическая операция «ИЛИ» между аккумулятором и данными	A ∨ data → A	ВСЕ, но AC = 0, CY = 0	КОП data	непосредственная
XRA r	логическая операция «Искл. ИЛИ» между аккумулятором и регистром	A ∨ r → A	ВСЕ, но AC = 0, CY = 0	КОП	регистровая
XRA M	логическая операция «Искл. ИЛИ» между аккумулятором и ячейкой памяти	A ∨ M(HL) → A	ВСЕ, но AC = 0, CY = 0	КОП	косвенная регистровая

Мнемоника	Операция	Символика	Формат	Флаги	Адресация
XRA data	непосредственная логическая операция «Искл. ИЛИ»	A ∨ data → A	ВСЕ, но AC = 0, CY = 0	КОП data	непосредственная
CMP r	сравнение содержимого аккумулятора и регистра	(A) – (r), A – не меняется; если A = r, то Z = 1; если A < r, то CY = 1	Z, CY	КОП	регистровая
CMP M	сравнение аккумулятора и содержимое ячейки памяти	(A) – (r), A – не меняется; если A = M, то Z = 1; если A < M, то CY = 1	Z, CY	КОП	косвенная регистровая
CMA	инверсия аккумулятора	A → A	НЕТ	КОП	неявная
CMC	инверсия бита переноса	CY → CY	CY	КОП	неявная
STC	установка бита переноса	(1) → CY	CY	КОП	неявная
RLC	сдвиг влево		CY	КОП	неявная
RRC	сдвиг вправо		CY	КОП	неявная
RAL	циклический сдвиг влево		CY	КОП	неявная
RAR	циклический сдвиг влево		CY	КОП	неявная

Вопросы и задания для самопроверки

1. Какая из команд устанавливает флаг служебного переноса?
2. Определите результат действия операции (для аккумулятора и регистра флагов) RLC, если (A) = 80H.
3. Что произойдет с флагом переноса и нуля по команде *CMP E*, если (A) = 02H, (E) = 14H.
4. Определите результат выполнения команд:
   1) *ANA C*, если (C) = 0EH, (A) = FCH;
   2) *ORA D*, если (A) = 33H, (D) = 0FH;
   3) *RLC*, если (A) = F2H;
   4) *RAR*, если (A) = 6AH, (CY) = 1.

ЛАБОРАТОРНАЯ РАБОТА 7
Состав команд операций ветвления и вызова подпрограмм

Цель работы – изучить команды условного и безусловного переходов, вызова и возврата из подпрограмм центрального процессора.

Краткие теоретические сведения

Команды условного и безусловного переходов позволяют изменять последовательность выполнения программ ЦП. Согласно безусловному способу последовательность выполнения программы подвергается изменению всякий раз, когда в программе содержится команда JMP Addr. Команда безусловного перехода является командой непосредственной адресации, используемой для изменения текущего адреса в счетчике команд ЦП. Второй и третий байт команды (адрес) загружается непосредственно в счетчик команд ЦП. Команды условных переходов определяют непосредственную загрузку нового адреса (второй и третий байт команды) в счетчик команд ЦП, если будут выполнены специальные условия. В противном случае счетчик команд будет инкрементирован нормально (новый адрес не будет в него загружен). В результате выполнения команд сравнения, арифметической и логической обработки данных устанавливаются флаги в регистре состояния ЦП. Команды условных переходов проверяют состояние одного из флагов регистра состояния для определения последующего хода выполнения программы в зависимости от результатов проверки. Команды переходов называют командами ветвления, позволяющими организовать программные циклы и разветвления.

Команда вызова подпрограмм CALL Addr и возврата из подпрограммы RET всегда используются парами. Команда CALL используется в программе для перехода к подпрограмме и сочетает функции операций загрузки в стек и перехода. В начале она загружает в стек содержимое счетчика команд (адрес возврата в программу). Затем счетчик команд загружается новым адресом (второй и третий байт команды) для выполнения перехода в подпрограмму (стартовый адрес подпрограммы). При встрече с командой RET содержимое вершины стека (адрес возврата в программу) загружается в счетчик команд. Таким образом, наличие стека в МПС и указателя стека в ЦП делает возможным возврат в основную программу после выполнения подпрограммы. Коман-

ды вызова подпрограмм дают возможность сократить объем программного обеспечения повторного использования подпрограмм.

Служебные команды и команды управления

DI – запрет прерываний. Внешние прерывания, поступающие на ЦП по входу INT, не разрешаются. Флаги и регистры ЦП не меняются.

EI – разрешение прерываний. Включается система прерываний ЦП по входу INT. Флаги и регистры ЦП не меняются.

HLT – остановка центрального процессора. ЦП останавливается и не выполняет никаких операций. Флаги и регистры ЦП не меняются.

NOP – отсутствие операции. ЦП выполняет холостую операцию. Флаги и регистры ЦП не меняются. Время выполнения команды – 4 периода тактовой частоты работы ЦП.

Содержание отчета

Отчет должен содержать:

- цель работы;
- таблицу команд операций ветвления и вызова подпрограмм (табл. 7.1);
- решение задач на ассемблере, выданных преподавателем.

Вопросы и задания для самопроверки

1. Куда передается 2 байт команды ***JMP***?
2. Сдвиньте содержимое аккумулятора на переменное число битов, которое определяется содержимым регистра ***D***.
3. Напишите программу, осуществляющую подсчет числа единиц в байте, находящемся в ***rB***.
4. Установите в нуль область памяти (ОЗУ), начиная с ячейки *Addr1* и заканчивая ячейкой *Addr2*.

Таблица 7.1

Команды перехода и ветвления

Мнемоника	Операция	Символика	Формат	Флаги	Адресация
JMP adr	безусловный переход	adr → (PC)	НЕТ	КОП мл., сб. байт	непосредственная
JZ adr	перейти по адресу, если ноль	если Z= 1, то adr → (PC)	Z	КОП мл., сб. байт	непосредственная
JNZ adr	перейти по адресу, если не ноль	если Z= 0, то adr → (PC)	Z	КОП мл., сб. байт	непосредственная
JC adr	перейти по адресу, если перенос	если CY= 1, то adr → (PC)	CY	КОП мл., сб. байт	непосредственная
JNC adr	перейти по адресу, если нет переноса	если CY= 0, то adr → (PC)	CY	КОП мл., сб. байт	непосредственная
JP adr	перейти по адресу, если число положительное	если S= 0, то adr → (PC)	S	КОП мл., сб. байт	непосредственная
JM adr	перейти по адресу, если число отрицательное	если S= 1, то adr → (PC)	S	КОП мл., сб. байт	непосредственная
JPO adr	перейти по адресу, если паритет нечетный	если P= 0, то adr → (PC)	P	КОП мл., сб. байт	непосредственная
JPE adr	перейти по адресу, если паритет четный	если P= 1, то adr → (PC)	P	КОП мл., сб. байт	непосредственная
CALL adr	вызов подпрограммы	(PC) → (SP), adr → (PC) [(SP)–2] → [(SP)]	НЕТ	КОП мл., сб. байт	непосредственная, косвенная, регистровая
CNZ adr	вызов подпрограммы, если не ноль	если Z= 0, то (PC) → (SP), adr → (PC), [(SP)–2] → (SP)	Z	КОП мл., сб. байт	непосредственная, косвенная, регистровая
CZ adr	вызов подпрограммы, если ноль	если Z= 1, то (PC) → (SP), adr → (PC), [(SP)–2] → (SP)	Z	КОП мл., сб. байт	непосредственная, косвенная, регистровая

Мнемоника	Операция	Символика	Формат	Флаги	Адресация
CC adr	вызов подпрограммы, если перенос	если CY=1, то (PC) → (SP), adr → (PC), [(SP)–2] → (SP)	CY	КОП мл., сб. байт	непосредственная, косвенная, регистровая
CM adr	вызов подпрограммы, если число отрицательное	если S= 0, то (PC) → (SP), adr → (PC),[(SP)–2] → (SP)	S	КОП мл., сб. байт	непосредственная, косвенная, регистровая
CPO adr	вызов подпрограммы, если паритет нечетный	если P= 0, то (PC) → (SP), adr → (PC),[(SP)–2] → (SP)	P	КОП мл., сб. байт	непосредственная, косвенная, регистровая
CPE adr	вызов подпрограммы, если паритет четный	если P= 1, то (PC) → (SP), adr → (PC),[(SP)–2] → (SP)	P	КОП мл., сб. байт	непосредственная, косвенная, регистровая
RET	возврат из подпрограммы	(SP) → (PC), [(SP)+2] → (SP)	НЕТ	КОП	косвенная регистровая
RZ	возврат из подпрограммы, если ноль	если Z= 1, то (SP) → (PC), [(SP)+2] → (SP)	Z	КОП	косвенная регистровая
RNZ	возврат из подпрограммы, если не ноль	если Z= 0, то (SP) → (PC), [(SP)+2] → [(SP)]	Z	КОП	косвенная регистровая
RC	возврат из подпрограммы, если перенос	если CY= 1, то (SP) → (PC), [(SP)+2] → (SP)	CY	КОП	косвенная регистровая
RNC	возврат из подпрограммы, если нет переноса	если CY= 0, то (SP) → (PC), [(SP)+2] → (SP)	CY	КОП	косвенная регистровая
RP	возврат из подпрограммы, если число положительное	если S= 0, то (SP) → (PC), [(SP)+2] → (SP)	S	КОП	косвенная регистровая
RM	возврат из подпрограммы, если число отрицательное	если S= 1, то (SP) → (PC), [(SP)+2] → (SP)	S	КОП	косвенная регистровая
RPO	возврат из подпрограммы если паритет нечетный	если P= 0, то (SP) → (PC), [(SP)+2] → (SP)	P	КОП	косвенная регистровая
RPE	возврат из подпрограммы, если паритет четный	если P= 1, то (SP) → (PC), [(SP)+2] → (SP)	P	КОП	косвенная регистровая

Примеры тестов промежуточного контроля знаний

1. Большинство людей в своей практической деятельности использует десятичную систему счисления. Какую систему счисления использует ЭВМ?
а) десятичную;
б) шестнадцатеричную;
в) двоичную;
г) восьмеричную;
д) двоично-десятичную.

2. Преобразованное в двоичный код десятичное число 45 – это:
а) число 1010.1010;
б) число 0100.0101;
в) число 0010.1101;
г) число 0101.1010;
д) число 0011.0011.

3. Какое из представленных двоичных чисел является результатом произведения двоичных чисел 1111 и 1110?
а) число 01010101;
б) число 11010010;
в) число 11001100;
г) число 1110.1110;
д) число 11000100.

4. Какая шина в микропроцессорной системе является однонаправленной?
а) шина адреса;
б) шина данных;
в) шина управления;
г) а и б;
д) б и в.

5. К какому количеству ячеек памяти можно получить доступ, используя 16 адресных линий?
а) 16384;
б) 65536;
в) 32768;

г) 131072;
д) 262144.

6. Как называется регистр состояний АЛУ?
а) РОН;
б) регистр временного хранения данных;
в) служебный регистр;
г) регистр флагов;
д) аккумулятор.

7. В какой регистр микропроцессора помещается КОП в начале выполнения первой команды?
а) регистр временного хранения данных;
б) служебный регистр;
в) регистр команд;
г) регистры общего назначения;
д) аккумулятор.

8. Какая адресация используется в команде XTHL?
а) регистровая;
б) косвенно регистровая;
в) непосредственная;
г) прямая;
д) неявная.

9. Как записывается команда sta addr?
а) КОП, байт;
б) КОП, мл. байт, ст. байт;
в) КОП, ст. байт, мл. байт;
г) КОП;
д) мл. байт, ст. байт, КОП.

10. Какая адресация используется в команде add r?
а) регистровая;
б) косвенно регистровая;
в) непосредственная;
г) прямая;
д) неявная.

11. Какой командой осуществляется циклический сдвиг вправо?

а) rlc;

б) rrc;

в) ral;

г) rar;

д) ani data.

12. Какой командой осуществляется вызов подпрограммы при переносе?

а) call addr;

б) cz addr;

в) cnz addr;

г) cc addr;

д) cm addr.

Задания на разработку программного обеспечения

Уровень А

1. Разработать программу деления двухбайтного числа на 2^n, где n – степень числа, записанная в регистр *B*. Начальный адрес числа записан в регистровую пару *DE* (само число расположено в памяти).

2. Разработать программу умножения двухбайтного числа на 2^n, где n – степень числа, записанная в регистр *B*. Начальный адрес числа записан в регистровую пару *DE* (само число расположено в памяти).

3. Разработать программу подсчета контрольной суммы массива по модулю 256. Адрес начала массива лежит в регистровой паре *BC*. Длина массива в регистровой паре *DE*. Результат положить в аккумулятор.

4. Преобразовать формат данных, находящихся в аккумуляторе.

Исходный формат	7	6	5	4	3	2	1	0
Конечный формат	4	5	$\overline{6}$	7	$\overline{1}$	3	0	$\overline{2}$

5. Разработать подпрограмму задержки длительностью 0,2 сек. Принять, что время выполнения одного машинного такта микропроцессора – 500 нсек.

6. Разработать программу умножения двух однобайтных чисел без знака. Первый множитель в регистре *E*, второй – в регистре *C*. Произведение необходимо разместить в памяти, адрес младшего байта произведения лежит в регистровой паре *HL*.

7. Разработать программу сложения двух четырехбайтных чисел. Начальный адрес младшего байта первого числа лежит в памяти по адресу, расположенному в регистровой паре *DE*. Числа лежат друг за другом. Результат положить в память, адрес младшего байта результата – в регистровой паре *HL*.

8. Разработать программу деления двухбайтного числа, расположенного в регистровой паре *BC*, на однобайтное число, расположенное в регистре *D*. Результат положить в память, адрес младшего байта результата – в регистровой паре *HL*.

Уровень В

9. Разработать программу сложения двоичных чисел (байтов) со знаком. В регистровой паре *HL* – начальный адрес массива байтов, в регистре *B* – длина массива. Результат сложения положить в память, адрес младшего байта результата – в регистровой паре *DE*.

10. Разработать программу перевода двоично-десятичного числа в двоичное число. Исходное число и результат перевода в регистре *A*. Принять, что двоично-десятичное число не больше 99.

11. Разработать программу нахождения среднего арифметического из пяти однобайтных чисел. Числа лежат в регистрах *B*, *C*, *D*, *E*, *H*. Результат положить в аккумулятор.

12. Разработать программу нахождения среднего арифметического массива. Адрес начала массива лежит в регистровой паре *BC*. Длина массива в регистре *D*. Результат поместить в аккумулятор.

13. Разработать программу нахождения наибольшего из 10 чисел. Числа лежат в памяти последовательно. Адрес первого числа лежит в регистровой паре *BC*. Адрес большего числа положить в регистровую пару *HL*.

14. Разработать программу нахождения наибольшего числа из массива. Адрес начала массива лежит в регистровой паре *BC*. Длина массива в регистре *D*. Адрес большего числа положить в регистровую пару *HL*.

15. Разработать программу нахождения наименьшего из 10 чисел. Числа лежат в памяти последовательно. Адрес первого числа лежит в регистровой паре *BC*. Адрес наименьшего числа положить в регистровую пару *HL*.

16. Разработать программу нахождения наименьшего числа из массива. Адрес начала массива лежит в регистровой паре *BC*. Длина массива в регистре *D*. Адрес наименьшего числа положить в регистровую пару *HL*.

17. Разработать программу сортировки массива по возрастанию. Адрес начала массива лежит в регистровой паре *BC*. Длина массива – в регистре *D*.

18. Разработать программу сортировки массива по убыванию. Адрес начала массива лежит в регистровой паре *BC*. Длина массива – в регистре *D*.

19. Разработать программу сравнения двух четырехбайтных чисел. Начальный адрес младшего байта первого числа лежит в памяти по адресу, расположенному в регистровой паре *DE*. Числа лежат друг за другом. Если первое число больше второго, то в аккумулятор положить число FFH, иначе 00H. Если числа равны – то 55H.

20. Разработать программу записи возрастающей арифметической прогрессии из 5 чисел в память. Адрес начала массива лежит в регистровой паре *HL*. Первый член лежит в регистре *B*. Шаг лежит в регистре *C*.

21. Разработать программу записи убывающей арифметической прогрессии из 5 чисел в память. Адрес начала массива лежит в регистровой паре *HL*. Первый член лежит в регистре *B*. Шаг лежит в регистре *C*.

Уровень С

22. Разработать программу умножения двух двухбайтных чисел со знаком. Первый множитель в регистровой паре *DE*, второй – в регистровой паре *BC*. Произведение необходимо разместить в памяти, адрес младшего байта произведения лежит в регистровой паре *HL*.

23. Разработать программу деления двух двухбайтных чисел со знаком. Первое число – в регистровой паре *DE*, второе – в регистровой паре *BC*. Частное – в памяти, адрес младшего байта частного – в регистровой паре *HL*.

24. Разработать программу нахождения среднего арифметического для трёх двухбайтных чисел. В регистровой паре *BC* – первое число, в регистровой паре *DE* – второе, в регистровой паре *HL* – третье. Результат (среднее арифметическое) поместить в регистровой паре *HL*.

25. Разработать программу нахождения среднего арифметического массива. Адрес начала массива лежит в регистровой паре *BC*. Длина массива в регистровой паре *DE*. Результат поместить в аккумулятор.

26. Разработать программу сортировки массива по возрастанию. Адрес начала массива лежит в регистровой паре *BC*. Длина массива – в регистровой паре *DE*.

27. Разработать программу сортировки массива по убыванию. Адрес начала массива лежит в регистровой паре *BC*. Длина массива – в регистровой паре *DE*.

28. Разработать программу сортировки пяти 32-разрядных чисел по возрастанию. Начальный адрес массива пяти чисел в регистровой паре *HL* (числа располагаются последовательно, начиная с младших байтов). Результат поместить в память, начальный адрес минимального числа – в регистровую пару *HL*.

29. Разработать программу сортировки пяти 32-разрядных чисел по убыванию. Начальный адрес массива пяти чисел в регистровой паре *DE* (числа располагаются последовательно, начиная с младших байтов). Результат поместить в память, начальный адрес максимального числа – в регистровую пару *HL*.

30. Разработать программу записи геометрической прогрессии из 5 чисел в память. Адрес начала массива лежит в регистровой паре *HL*. Первый член лежит в регистре *B*. Знаменатель прогрессии лежит в регистре *C*.

Пример оформления разработанного программного обеспечения

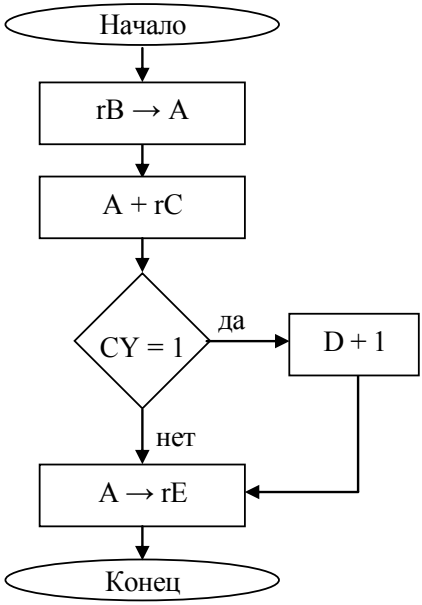
Задание

Разработать программу сложения двух однобайтных чисел. Первое число в rB. Второе число в rC. Результат необходимо положить в rp DE.

Алгоритм

Очищаем rp DE. Загружаем первое число из rB в аккумулятор. Складываем аккумулятор с rC. Если был перенос, то увеличиваем rD на 1. Загружаем результат из аккумулятора в rE.

Блок-схема алгоритма работы



Программа

```
START:    LXI DE, 00H    ;0 → rp DE
          MOV A, B       ; rB → A
          ADD C          ; A + rC → A
          JNC M1         ; переход на метку M1, если CY = 0
          INR D          ; rD + 1 → rD
M1:       MOV E, A       ; A → rE
STOP:     HLT
```

Выводы

В данной работе была создана программа сложения двух однобайтных чисел. Выполняя данную работу, я получил первоначальные навыки по построению программы на языке ассемблера, научился составлять и описывать алгоритм.

Библиографический список

Основная литература

1. Агунов, М.В. Микропроцессоры в вопросах и ответах : учеб. пособие для вузов по электротехн. спец. / М.В. Агунов ; науч. ред. В.В. Ивашин. – Тольятти : ТолПИ, 2000. – 82 с.
2. Алексеенко, А.Г. Проектирование радиоэлектронной аппаратуры на микропроцессорах: программирование, типовые решения, методы отладки / А.Г. Алексеенко, А.А. Галицын, А.Д. Иванников. – М. : Радио и связь, 1984. – 270 с.
3. Арсеньев, Ю.Н. Проектирование систем логического управления на микропроцессорных средствах : учеб. пособие для вузов / Ю.Н. Арсеньев, В.М. Журавлев. – М. : Высш. шк., 1991. – 319 с.
4. Корячко, В.П. Микропроцессоры и микро ЭВМ в радиоэлектронных средствах : учеб. для вузов / В.П. Корячко. – М. : Высш. шк., 1990. – 407 с.
5. Ломака, М.В. Микропроцессорное управление приводами / М.В. Ломака, И.В. Медведев. – М. : Машиностроение, 1990. – 96 с.
6. Микропроцессоры : учеб. для втузов : в 3 кн. / П.В. Нестеров [и др.] ; под ред. Л.Н. Преснухина. – М. : Высш. шк., 1986. – Кн. 1. – 495 с.
7. Микропроцессоры : учеб. пособие : в 5 кн. / В.А. Шахнов. – М. : Высш. шк., 1988. – Кн. 1. – 175 с.
8. Рафикузаман, М. Микропроцессоры и машинное проектирование микропроцессорных систем : в 2 кн. : [пер. с англ.] / М. Рафикузаман. – М. : Мир, 1988. – Кн. 1. – 311 с.
9. Токхайм, Р. Микропроцессоры: курс и упражнения : учеб. пособие : [пер. с англ.] / Р. Токхайм. – М. : Энергоатомиздат, 1988. – 336 с.
10. Холленд, Р.Ч. Микропроцессоры и операционные системы : крат. справ. пособие : [пер. с англ.] / Р.Ч. Холленд. – М. : Энергоатомиздат, 1991. – 191 с.
11. Черемных, С.В. От микропроцессоров к персональным ЭВМ / С.В. Черемных, А.В. Гиглавый, Ю.Е. Поляк. – М. : Радио и связь, 1988. – 287 с.
12. Шевкопляс, Б.В. Микропроцессорные структуры: инж. решения : справочник / Б.В. Шевкопляс. – 2-е изд., перераб. и доп. – М. : Радио и связь, 1990. – 511 с.

13. Шилейко, А.В. Микропроцессоры / А.В. Шилейко, Т.И. Шилейко. – М. : Радио и связь, 1986. – 112 с.

Дополнительная литература

14. Майоров, В.Г. Практический курс программирования микропроцессорных систем / В.Г. Майоров, А.И. Гаврилов. – М. : Машиностроение, 1989. – 272 с.
15. Калабеков, Б.А. Микропроцессоры и их применение в системах передачи и обработки сигналов / Б.А. Калабеков. – М. : Радио и связь, 1988. – 368 с.
16. Евстифеев, А.В. Микроконтроллеры AVR семейства Classic фирмы ATMEL / А.В. Евстифеев. – 3-е изд., стер. – М. : Додэка-XXI, 2006. – 288 с.

СОДЕРЖАНИЕ

Учебное издание

Ермаков Виктор Васильевич
Пьянов Михаил Александрович

МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА И СИСТЕМЫ

Практикум
по лабораторным работам

Редактор *Е.Ю. Жданова*
Технический редактор *З.М. Малявина*
Компьютерная верстка: *Л.В. Сызганцева*
Дизайн обложки: *Г.В. Карасева*

Подписано в печать 17.06.2011. Формат 60×84/16.
Печать оперативная. Усл. п. л. 4,18.
Тираж 110 экз. Заказ № 1-85-10.

Тольяттинский государственный университет
445667, г. Тольятти, ул. Белорусская, 14