ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САХАЛИНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ЛАБОРАТОРНЫЕ РАБОТЫ ПО АРХИТЕКТУРЕ КОМПЬЮТЕРА

Учебное пособие

Южно-Сахалинск
СахГУ
2016

УДК 004.2(075.32)
ББК 32.973-02я723
Л12

Печатается по решению учебно-методического совета Сахалинского государственного университета, 2014 г.

Рецензенты:

Никонов Ю. Ю., *начальник отдела автоматизированных информационных технологий ФГБУ «Сахалинское УГМС»;*
Максимов В. П., *профессор кафедры теории и методики обучения технологии и предпринимательству ФГБОУ ВПО «СахГУ», доктор педагогических наук, профессор.*

Авторский коллектив:
М. А. Смирнова, Е. Д. Уткин, О. А. Федоров,
Д. С. Богданов, М. А. Некрасов.

Л12 **Лабораторные работы по архитектуре компьютера :** учебное пособие / М. А. Смирнова, Е. Д. Уткин, О. А. Федоров [и др.]. – Южно-Сахалинск : изд-во СахГУ, 2016. – 60 с.
ISBN 978-5-88811-528-2

Целью данной работы является оказание методической помощи учащимся и преподавателям для проведения лабораторного практикума по дисциплине «Архитектура компьютера». Пособие включает описания лабораторных работ с элементами программирования на ассемблере в среде DEBUG для исследования работы виртуального режима процессоров Intel.

УДК 004.2(075.32)
ББК 32.973-02я723

ISBN 978-5-88811-528-2

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 4
РАБОТА 1. Системы счисления.....7
РАБОТА 2. Центральные и внешние устройства ПК 15
РАБОТА 3. Отладчик DEBUG как средство
для ознакомления с архитектурой INTEL 8086.....29
РАБОТА 4. Микропроцессор и память компьютера.....35
РАБОТА 5. Программные и аппаратные прерывания 48
РАБОТА 6. Изучение системы команд ассемблера.....54
ЛИТЕРАТУРА.....58

ВВЕДЕНИЕ

В целях интеграции теории и практики в вузах в последнее время получают широкое распространение комплексные лабораторные работы, проводимые на широком техническом фоне с применением разнообразной аппаратуры и компьютера в условиях, близких к реальным, в которых будет работать будущий специалист.

Лабораторная работа – существенный элемент учебного процесса в вузе, в ходе которого обучающиеся сталкиваются с самостоятельной практической деятельностью в конкретной области исследования. Как и другие виды практических занятий, лабораторные занятия являются средним звеном между углубленной теоретической работой обучающихся на лекциях и применением знаний на практике. Эти занятия удачно сочетают элементы теоретического исследования и практической работы.

Выполняя лабораторные работы, учащиеся лучше усваивают программный материал, так как многие определения и формулы, казавшиеся несколько абстрактными, становятся вполне конкретными. Происходит соприкосновение теории с практикой, что в целом содействует изучению сложных вопросов науки и становлению обучающихся как будущих специалистов.

Само значение слова «лаборатория» (от лат. «*labor*» – труд, работа, трудность, трудиться, стараться, хлопотать, преодолевать затруднения) указывает на сложившиеся понятия, связанные с применением умственных и физических усилий к изысканию ранее неизвестных путей для разрешения научных и жизненных задач [6].

Тематическое наполнение предлагаемых лабораторных работ практикума основывается на государственном образовательном стандарте по дисциплине «Архитектура компьютера», а тематическая разбивка осуществляется по логическим уровням, предложенным в книге Э. Таненбаум [16] (табл. 1.1).

Такое разделение дает понимание того, что компьютер есть сложная иерархическая структура, основанная на простых принципах (каждый последующий уровень строится из бло-

ков предыдущего уровня). Разделяя таким образом теоретический материал, выделяются ключевые моменты по каждому из уровней. Закрепление ключевых моментов и является целью каждой лабораторной работы.

Таблица 1.1

Тематическая таблица по логическим уровням

Название темы ГОС	Номер лабораторной работы
История развития компьютерной техники, поколения ЭВМ и их классификация	–
Центральные и внешние устройства ЭВМ, их характеристики	2 и 4
Кабельная и шинная системотехника	2
Микропроцессор и память компьютера	4
Система прерываний, регистры и модель доступа к памяти	3, 4 и 5
Защищенный режим работы процессора как средство реализации многозадачности	–
Принципы управления внешними устройствами персонального компьютера	5 и 6
Базовая система ввода-вывода	6
Ассемблер как машинно ориентированный язык программирования	3, 4, 5 и 6
Понятие о микропрограммировании	3, 4, 5 и 6
Современные тенденции развития архитектуры ЭВМ	–

Успех лабораторных занятий зависит от многих слагаемых: от теоретической, практической и методической подготовленности преподавателя, его организаторской работы по подготовке занятия, от состояния лабораторной базы и методического обеспечения, а также от степени готовности самих обучающихся, их активности на занятии.

Формы организации лабораторного занятия зависят, прежде всего, от числа студентов, содержания и объема программного материала, числа лабораторных работ, а также от вместимости

учебных помещений и наличия оборудования. В зависимости от этих условий в вузах применяют следующие формы лабораторных занятий: фронтальную, по циклам, индивидуальную и смешанную (комбинированную) [6].

Данный практикум ориентирован на фронтальную форму проведения занятий в компьютерном классе; непосредственно после чтения лекционного материала с последующей индивидуальной работой преподавателя с учащимися по каждой теме. Такая форма обучения более рациональна, так как вовлекает на начальном этапе сразу всех в процесс работы. И только после обсуждения и понимания хода работы выполняются индивидуальные задания.

РАБОТА 1.
СИСТЕМЫ СЧИСЛЕНИЯ

Цель работы: изучение позиционных систем счисления.
Приборы и принадлежности: компьютер.

1.1. Используемые системы счисления

Система счисления – символический метод записи чисел или способ представления чисел с помощью письменных знаков, именуемых цифрами.

Число – это некоторая абстрактная сущность для описания количества чего-либо.

Цифры – это знаки, используемые для записи чисел.

Поскольку чисел гораздо больше, чем цифр, то для записи числа обычно используется набор (комбинация) цифр.

Существует много способов записи чисел с помощью цифр. Каждый такой способ называется системой счисления. Величина числа может зависеть от порядка цифр в записи, а может и не зависеть. Это свойство определяется системой счисления и служит основанием для простейшей классификации таких систем.

Указанное основание позволяет все системы счисления разделить на три класса (группы): позиционные, непозиционные и смешанные.

Примером «чисто» непозиционной системы счисления является римская система, а смешанной – денежная система единиц.

Позиционные системы счисления – это системы счисления, в которых значение цифры напрямую зависит от ее позиции в числе. Например, число 01 обозначает единицу, 10 – десять. Позиционные системы счисления позволяют легко производить арифметические расчеты.

Представление чисел с помощью арабских цифр – самая распространенная позиционная система счисления, она называется «десятичной системой счисления». Десятичной системой она называется потому, что использует десять цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8 и 9.

Для составления машинных кодов удобно использовать не де-

сятичную, а двоичную систему счисления, содержащую только две цифры 0 и 1. Программисты для вычислений также пользуются восьмеричной и шестнадцатеричной системами счисления.

Количество цифр, используемых в системе счисления, называется ее «основанием». В десятичной системе основание равно десяти, в двоичной системе – двум, а в восьмеричной и шестнадцатеричной – соответственно, восьми и шестнадцати. В q-ичной системе счисления количество цифр равно q и используются цифры от 0 до $q - 1$.

Для работы **необходимо знать** представления десятичных чисел от нуля до 15 в системах счисления с основаниями $q = 2$, 8, 16 (см. табл. 1.2).

Таблица 1.2

Представления десятичных чисел в разных системах счисления

q = 10	q = 2	q = 8	q = 16
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	*A*
11	1011	13	*B*
12	1100	14	*C*
13	1101	15	*D*
14	1110	16	*E*
15	1111	17	F

Кроме этого, полезно знать десятичные значения чисел 2^k от $k = 0$ до $k = 10$ (см. таб. 1.3).

Таблица 1.3

Значения чисел 2^k

k	0	1	2	3	4	5	6	7	8	9	10
2^k	1	2	4	8	16	32	64	128	256	512	1024

Перевод чисел из одной системы счисления в другую

Для перевода целого числа N с q-ичным основанием в десятичное число записывают в виде многочлена, а затем вычисляют его по правилам десятичной арифметики:

$$N = a_n \cdot q^n + a_{n-1} \cdot q^{n-1} ... + a_2 \cdot q^1 + a_1 \cdot q^0.$$

Здесь a_n – это цифры числа,

q – основание системы счисления,

n – 0, 1, 2 ... .

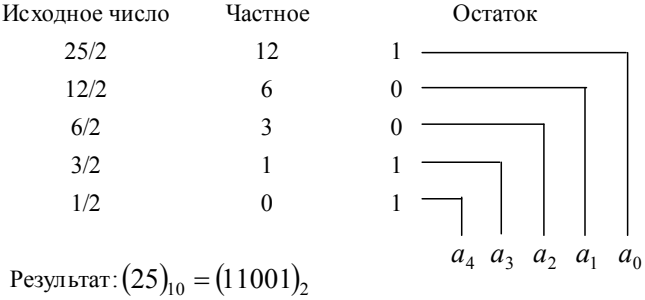
Пример:

$$(11001)_2 = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 =$$
$$= 1 \cdot 16 + 1 \cdot 8 + 0 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 = (25)_{10}$$
$$(221)_3 = 2 \cdot 3^2 + 2 \cdot 3^1 \cdot 1 \cdot 3^0 = 2 \cdot 9 + 2 \cdot 3 + 1 \cdot 1 = (25)_{10}$$
$$(221)_3 = 2 \cdot 3^2 + 2 \cdot 3^1 \cdot 1 \cdot 3^0 = 2 \cdot 9 + 2 \cdot 3 + 1 \cdot 1 = (25)_{10}$$
$$(31)_8 = 3 \cdot 8^1 + 1 \cdot 8^0 = 3 \cdot 8 + 1 \cdot 1 = (25)_{10}$$
$$(534D)_{16} = 5 \cdot 16^3 + 2 \cdot 16^2 + 4 \cdot 16^1 + 13 \cdot 16^0 =$$
$$= 20480 + 512 + 64 + 13 = (21069)_{10}$$

Примечание: при работе с различными системами счисления число записывают в скобках, а за скобками – основание системы.

Для обратного преобразования целых чисел (из десятичной системы счисления в систему с основанием q) число N делят на q и записывают остатки от деления до тех пор, пока частное от предыдущего деления не станет равным нулю.

Пример: преобразуем число 25 в двоичную систему:



Когда последнее частное стало равно нулю, записывают все остатки подряд от последнего к первому. Таким образом, получили число в двоичной системе счисления – $(11001)_2$.

Для перевода смешанных чисел в двоичную систему счисления требуется отдельно переводить их целую часть и дробную части. В записи результата целая часть перевода отделяется от дробной запятой в соответствии с формулой:

$$N = \pm a_n \quad a_{n-1} \quad \dots \quad a_1 \quad a_0, \quad a_{-1} \quad a_{-2} \quad \dots \quad a_{-n}$$

Основные системы счисления

Двоичная система счисления. В компьютерной технике в основном используется двоичная система счисления. Такую систему очень легко реализовать в цифровой микроэлектронике, так как для нее требуется всего два устойчивых состояния (0 и 1).

Двоичная система счисления может быть непозиционной и позиционной. Реализовано это может быть присутствием какого-либо физического явления или его отсутствием. Например: есть электрический заряд или его нет, есть напряжение или нет, есть ток или нет, есть сопротивление или нет, отражает свет или нет, намагничено или не намагничено, есть отверстие или нет и т. п.

Восьмеричная система счисления – позиционная целочисленная система счисления с основанием 8. Для представления чисел в ней используются цифры от 0 до 7.

Восьмеричная система счисления часто используется в областях, связанных с цифровыми устройствами. Характеризуется легким переводом восьмеричных чисел в двоичные и обратно, путем замены восьмеричных чисел на триады двоичных.

Ранее эта система широко использовалась в программировании и компьютерной документации, однако в настоящее время почти полностью вытеснена шестнадцатеричной системой.

Для перевода двоичного числа в восьмеричное исходное число разбивают на триады влево и вправо от запятой; отсутствующие крайние цифры дополняют нулями. Затем каждую триаду записывают восьмеричной цифрой (см. табл. 1.2).

Пример: иллюстрация перевода двоичного числа в восьмеричное число:

$$N = (110011,100010)_2 = \left(\overbrace{110}^{6}\ \overbrace{011}^{3},\overbrace{100}^{4}\ \overbrace{010}^{2} \right)_2 = (63,42)_8 .$$

Шестнадцатеричная система счисления – позиционная система счисления по целочисленному основанию 16. Обычно в качестве шестнадцатеричных цифр используются десятичные цифры от 0 до 9 и латинские буквы от A до F для обозначения цифр от $(1010)_2$ до $(1111)_2$, то есть (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)$_{16}$.

Для перевода двоичного числа в шестнадцатеричное исходное число разбивают на тетрады влево и вправо от запятой; отсутствующие крайние цифры дополняют нулями. Затем каждую тетраду записывают шестнадцатеричной цифрой (см. табл. 1.2).

Пример: иллюстрация перевода двоичного числа в шестнадцатеричное число:

$$N = \left(\overbrace{0111}^{7}\ \overbrace{1010}^{A}\ \overbrace{1011}^{B},\ \overbrace{1110}^{E}\ \overbrace{1111}^{F} \right)_2 = (7AB,EF)_{16}.$$

1.2. Варианты заданий к лабораторной работе (см. табл. 1.4)

Задание 1. Перевести целые числа из десятичной системы счисления:

– в двоичную;

– в восьмеричную;

– в шестнадцатеричную.

Задание 2. *Перевести целые числа из двоичной системы счисления:*

– в восьмеричную;
– в шестнадцатеричную;
– в десятичную.

Задание 3. *Перевести целые числа из шестнадцатеричной системы счисления:*
– в двоичную;
– в восьмеричную;
– в десятичную.

Задание 4. *Сложить*:
– двоичные числа;
– восьмеричные числа;
– шестнадцатеричные числа.

Задание 5. *Найти разность:*
– двоичных чисел;
– восьмеричных чисел;
– шестнадцатеричных чисел.

Задание 6. *Вычислить значение выражения и представить в десятичной системе счисления.*

Таблица 1.4

Варианты заданий к лабораторной работе

	Вариант 1	**Вариант 2**	**Вариант 3**	**Вариант 4**	**Вариант 5**
Задание 1	a) 2515 b) 3084 c) 9042	a) 1052 b) 1387 c) 7634	a) 2042 b) 5548 c) 2372	a) 5911 b) 6321 c) 7629	a) 3988 b) 5147 c) 1123
Задание 2	a) 110101 b) 111010 c) 101111	a) 011001 b) 101010 c) 010101	a) 100110 b) 110011 c) 101111	a) 011001 b) 100001 c) 001001	a) 100010 b) 111000 c) 011111
Задание 3	a) 1F52 b) 5521 c) 1101	a)1A1B b) 2350 c) 3239	a) 5EE2 b) 2682 c) 2461	a) 7B1B b) 3458 c) 6537	a) 1C2D b) 6824 c) 8673
Задание 4	a) 1011 + + 0111 b) 573 + 325 c) F1 + E7	a) 0110 + + 1100 b) 274 + 235 c) 93 + 2C	a) 1010 + + 0101 b) 271 + 123 c) 58 + 79	a) 1101 + + 1101 b) 632++714 c) 51 + 9D	a) 1010 + + 1010 b) 521++623 c) 36 + AB

Продолжение таблицы 1.4

	Вариант 1	**Вариант 2**	**Вариант 3**	**Вариант 4**	**Вариант 5**
Задание 5	a) 1011 – – 0111 b) 573 – 325 c) F1 – E7	a) 1110 – – 1100 b) 274 – 235 c) 93 – 2C	a) 1010 – – 0101 b) 271 – 123 c) A8 – 79	a) 1101 – – 1001 b) 732 – 714 c) B1 – 9D	a) 1010 – – 1000 b) 721 –623 c) C6 – AB
Задание 6	23_8 + $A2_{16}$ * * 1001_2	$B1_{16}$ – – 1011_2 * * 117_8	51_8 * 21_{16} – – 45510	(59_{16} + + 1110_2) * * 456_8	25_8 * * 567_{16} – – 10101_2

1.3. Контрольные вопросы

1. Что называется системой счисления?
2. Какие системы счисления называются непозиционными? Почему? Приведите пример такой системы счисления и записи чисел в ней.
3. Какие системы счисления применяются в вычислительной технике: позиционные или непозиционные? Почему?
4. Как изображается число в позиционной системе счисления?
5. Что называется основанием системы счисления?
6. Как можно представить целое положительное число в позиционной системе счисления?
7. Какие системы счисления применяются в компьютере для представления информации?
8. По каким правилам выполняется сложение двух положительных целых чисел?
9. Каковы правила выполнения арифметических операций в двоичной системе счисления?
10. Для чего используется перевод чисел из одной системы счисления в другую?
11. Сформулируйте правила перевода чисел из системы счисления с основанием p в десятичную систему счисления и обратно: из десятичной системы счисления в систему счисления с основанием s. Приведите примеры.
12. Как выполнить перевод чисел из двоичной системы счисления в восьмеричную систему и обратно? Из двоичной

системы счисления в шестнадцатеричную систему и обратно? Приведите конкретные примеры.

13. По каким правилам выполняется перевод чисел из восьмеричной в шестнадцатеричную систему счисления и наоборот? Приведите примеры.

Литература:

1. Бабич, Н. П. Компьютерная схемотехника. Методы построения и проектирования / Н. П. Бабич, И. А. Жуков : учебное пособие. – Киев : МК-Прогресс, 2004. – С. 18–26.

2. Марек, Р. Ассемблер на примерах. Базовый курс / Р. Марек. – СПб. : Наука и техника, 2005. – С. 12–15.

РАБОТА 2.
ЦЕНТРАЛЬНЫЕ И ВНЕШНИЕ УСТРОЙСТВА ПК

Цель работы: изучение функционального взаимодействия центральных и внешних устройств компьютера и их характеристик.

Приборы и принадлежности: комплектующие элементы системного блока компьютера: корпус системного блока, блок питания, материнская плата, процессор, оперативная память, видеокарта, жесткий диск. Блок CD/DVD-ROM, монитор, клавиатура, мышь. Кулер, термопаста, отвертка плоская, отвертка крестовая, компьютерные болты.

2.1. Краткая теоретическая часть

Персональный компьютер (ПК) настольного исполнения в минимальной конфигурации состоит из следующих составных частей: системный блок, монитор, клавиатура, манипулятор-мышь.

2.1.1. Корпус системного блока представляет собой прямоугольный параллелепипед, изготовленный из листового металла (сталь, алюминиевый сплав), внутри которого расположены все компоненты, необходимые для функционирования ПК. Неважно, какая компания изготовила корпус компьютера, любой корпус компьютера соответствует единому стандарту, называемому **формфактором**.

Формфактор (form factor) – это стандарт технического изделия, описывающий совокупность его технических параметров, например форму, размер, положение и типы разъемов, требований к вентиляции, напряжениям и прочих параметров.

Различают два типа формфактора корпуса ПК: AT и ATX.

Формфактор AT (англ. «*Advanced Technology*» – передовая технология, класс PC на процессорах 286: технология использовалась для компьютеров, включая Pentium-1) – это первый широко использовавшийся формфактор в персональных компьютерах в 80-х годах прошлого века и в настоящее время больше не производится. Формфактор AT был создан IBM в 1984 году и пришел на смену ранее существовавшим форм-факторам PC и XT.

Формфактор ATX (от англ. «*Advanced Technology*

Extended» – передовая технология расширенная) – формфактор персональных настольных компьютеров.

В 1995 году корпорация INTEL предложила новую спецификацию корпуса – систему блока ATX (AT *eXtention*). Его появление обусловлено возросшей мощностью персональных компьютеров и наличием большого числа внешних устройств, подключаемых к ПК, а также встроенной микросхемой в материнскую плату, что повысило требования к охлаждению этих элементов. Формфактор ATX является доминирующим форматом для массово-выпускаемых компьютеров, начиная с 2001 года.

В зависимости от габаритных размеров корпусы системных блоков делятся на типы, перечень которых приведен в табл. 2.1.

Таблица 2.1

Размеры корпусов системных блоков ПК

Тип корпуса	Высота (см)	Ширина (см)	Длина (см)
Desktop	20	45	45
Slimline	7	35	45
Mini Tower	45	20	45
Midi Tower	50	20	45
Big Tower	65	20	48
Super Big Tower	75	22	48
File Server	75	35	55

Насколько это важно – знать формфактор ПК?

От правильности выбора формфактора компьютера зависит множество немаловажных параметров: например, уровень шума, температура корпуса, вентиляция, которые влияют на работу комплектующих внутренних частей системного блока: в первую очередь жесткого диска, центрального процессора, других составных частей системного блока.

Формфактор ATX определяет следующие основные характеристики ПК:

– геометрические размеры материнских плат;

– общие требования по положению разъемов и отверстий на корпусе;

– положение блока питания в корпусе;
– геометрические размеры блока питания;
– электрические характеристики блока питания и цвета проводов;
– форму и положение ряда разъемов (преимущественно питания).

Пример: цвет проводов стандарта ATX (см. табл. 2.2).

Таблица 2.2

Цвет проводов стандарта ATX

Цепь	Цвет провода	Пояснение
+5V	Красный	Основное напряжение
+12V	Желтый	Питание двигателей устройств и интерфейсных цепей
-5V	Белый	Не используется. Присутствует для соблюдения стандарта ISA Bus
-12V	Синий	Питание интерфейсных цепей
+3,3V	Оранжевый	Питание процессора
+3,3V Sence	Коричневый	Сигнал обратной связи стабилизатора +3,3V
+5VSB	Малиновый	Дежурный маломощный источник +5V
PS-ON	Зеленый	Сигнал включения источников питания
PW-OK	Серый	Сигнал «питание в норме»
GND	Черный	Общий провод относительно питающих напряжений

2.1.2. В корпусе системного блока размещаются следующие компоненты:
– материнская (системная) плата;
– блок питания;
– жесткий диск;
– дисковод гибких дисков;
– привод CD/DVD дисков;
– платы расширения (устанавливаются в слотах материнской платы, а крепятся к задней стенке корпуса системного блока).

Материнская плата является главной составной частью персонального компьютера. От ее качества и быстродействия

напрямую зависит стабильность и скорость работы всего компьютера.

Материнская плата задает фундаментальные параметры компьютера (тип процессора, тип памяти), определяющие уровень производительности, а также практические параметры (формфактор, количество слотов расширения, наличие интегрированных устройств), потребительские свойства и возможную сферу применения.

2.1.3. На материнской плате размещено следующее:

– сокет для установки процессора;
– основной набор микросхем материнской платы (чипсет);
– микросхема BIOS (Basic Input Output System);
– модули ОЗУ;
– слоты для установки плат расширения;
– коннекторы для подключения IDE- или SCSI-устройств и FDD;
– вспомогательные коннекторы для подключения индикаторных элементов и кнопок управления, расположенных на передней панели корпуса;
– внешние порты для клавиатуры, мыши, принтера, шины USB и др.

Размеры материнской платы определяются также формфактором. Основные их формфакторы перечислены в таблице 2.3.

Таблица 2.3

Формфакторы материнских плат

Формфактор	**Физические размеры (мм)**	**Спецификация, год**	**Примечание**
XT	(8,5 × 11)″ (216 × 279)	IBM, 1983	Архитектура IBM PC XT
AT	(12 × 11 – 13)″ (305 × 279÷330)	IBM, 1984	Архитектура IBM PC AT (Desktop/Tower)
ATX	(12 × 9,6)″ (305 × 244)	Intel, 1995	Для системных блоков типов MiniTower, FullTower

Продолжение таблицы 2.3

Форм-фактор	Физические размеры (мм)	Спецификация, год	Примечание
Mini ATX	(11,2 × 8,2)″ (284 × 208)	Intel, 1995	Для системных блоков типа Tower и компактных Desktop
Micro ATX	(9,6 × 9,6)″ (244 × 244)	Intel, 1997	Имеет меньше слотов, чем ATX

2.1.4. Непосредственно на материнскую плату устанавливаются:

– процессор;

– радиатор процессора с кулером;

– модули оперативной памяти;

– платы расширения (крепятся винтами к задней стенке системного блока).

Такие модули, как блок питания, жесткий диск, дисковод гибких дисков, привод CD/DVD дисков, размещаются в специальных отсеках корпуса системного блока и крепятся к нему винтами.

2.2. Практическая часть

Задание 1.1. **Идентификация компонентов системного блока**

Получите у преподавателя набор компонентов, предназначенных для сборки системного блока ПК. Определите формфактор корпуса ПК. Аккуратно разложите компоненты на столе, определите их назначение и заполните таблицу 2.4.

Таблица 2.4

Основные компоненты системного блока ПК

№	Наименование компонента	Модель, тип, функциональное назначение
1.	Формфактор корпуса	
2.	Материнская плата	
3.	Блок питания	
4.	Жесткий диск	
5.	Дисковод гибких дисков	

Продолжение таблицы 2.4

№	Наименование компонента	Модель, тип, функциональное назначение
6.	Привод CD/DVD дисков	
7.	Кулер	
8.	Другое...	

Задание 1.2. **Идентификация компонентов материнской платы**

Определите расположение на материнской плате основных установленных компонентов, а также компонентов, которые на нее устанавливаются, и заполните таблицу 2.5.

Таблица 2.5

Основные компоненты материнской платы ПК

№	Наименование компонента	Установленные компоненты (или требуется установить)	Модель, тип, функциональное назначение
1.	Socket для установки процессора		
2.	Процессор		
3.	Радиатор процессора с кулером		
4.	Чипсет		
5.	Модули оперативной памяти		
6.	Микросхема BIOS		
7.	Слоты для плат расширения		
8.	Платы расширения		
9.	Внешние порты для клавиатуры		
10.	Внешние порты для мыши		

Продолжение таблицы 2.5

№	Наименование компонента	Установленные компоненты (или требуется установить)	Модель, тип, функциональное назначение
11.	Внешние порты для принтера		
12.	Шины USB внутренние		
13.	Внешние порты для USB		
14.	Другое...		

Задание 2. **Установка процессора и радиатора с кулером**

Сборка компьютера начинается с установки процессора в сокет материнской платы. У сокета процессора на системной плате поднимите на 90 °С рычаг фиксации контактов. Этот рычаг может быть пластмассовым или металлическим.

Процессор поднесите к сокету и определите положение ключей-контактов, пропущенных по углам корпуса процессора и сокета. Процессор должен устанавливаться в сокет без усилий, в противном случае можно у процессора погнуть контакты.

Закройте рычаг сокета. Усилие, прикладываемое к рычагу, не должно быть слишком большим.

Для охлаждения процессора используется радиатор с закрепленным на нем кулером (вентилятором). Для повышения эффективности передачи тепла от процессора радиатору применяется термопаста (например, типа КПТ-8), которая наносится на крышку процессора перед установкой на него радиатора. На центр крышки процессора из тюбика наносится тонкий слой термопасты.

На процессор очень аккуратно наложите радиатор и прижмите его к процессору. Закрепите радиатор.

Задание 3. **Установка модулей памяти**

Перед установкой модулей памяти обязательно следует найти в документации на системную плату таблицу установки модулей по слотам.

Существуют определенные ограничения по комбинациям модулей в слотах. Обычно одиночный модуль должен быть установлен в первый слот первого банки памяти. Если чипсет, на базе которого построена системная плата, имеет двухканальную конфигурацию памяти, то для активации двухканальной памяти в разные банки памяти можно устанавливать только одинаковые по структуре модули памяти.

При установке модуль памяти вводится в направляющие слота, при этом следует проследить, чтобы прорези (ключи) в печатной плате модуля памяти и пластмассовые выступы на слоте совпадали. Защелки на слотах должны быть отведены в сторону.

Для окончательной установки модуля памяти надо сильно нажать двумя пальцами по краям модуля. При правильной установке защелки должны зафиксировать модуль в слоте, в этом случае обычно раздастся негромкий щелчок. Если фиксации не произошло, значит, модуль при установке был неправильно ориентирован или делается попытка установить неподходящий для этой системной платы модуль памяти.

Для двухканальных вариантов модули памяти устанавливаются в разные слоты (банки). Принято использовать для слотов разноцветную пластмассу, поэтому рекомендуется устанавливать модули в слоты одинакового цвета.

Задание 4. **Монтаж материнской платы**

Системная плата крепится в корпусе компьютера на металлических шестигранных резьбовых втулках.

Шестигранные втулки ввинчиваются в тех точках крепления на корпусе системного блока, где есть отверстия с резьбой. Металлические втулки используются не только для крепления системной платы, но и для создания точек заземления. Как правило, две такие точки заземления имеются у края системной платы, где находятся слоты расширения. Для эффективного заземления на системной плате вокруг крепежных отверстий (не обязательно всех) выполняется токопроводящая дорожка. В тех точках крепления, где на системной плате нет заземляющей токопроводящей дорожки, при использовании металлических втулок следует дополнительно использовать пластмассовые шайбы.

Определившись с точками крепления, установите системную плату в корпус. Если блок питания размещен над поверхностью системной платы (как в корпусах Mini Tower), то убедитесь, что радиатор процессора не упирается в блок питания (остается зазор около 10 мм). Крепить системную плату винтами следует только после проверки совпадения всех крепежных стоек с отверстиями на системной плате, возможности свободной установки блока питания и отсутствия заглушек в корпусе напротив слотов расширения, в которые предполагается установить платы расширения. Удалять все заглушки не следует, так как через неиспользуемые отверстия будет набираться пыль и появится возможность попадания посторонних предметов внутрь системного блока.

Задание 5. **Подключение органов управления**

На лицевой стороне корпуса системного блока расположены несколько кнопок управления (включение питания, сброс) и светодиодных индикаторов (питание, работа накопителей). Для их подключения на материнской плате имеется контактная панель, к которой присоединяются все разъемы, идущие от электронных элементов лицевой панели корпуса, в том числе и от динамика. Как правило, контактная панель расположена в одном из углов материнской платы.

Подключать разъемы следует в соответствии с указаниями в документации на материнскую плату. На самой материнской плате назначение групп контактов обычно маркируется краской. Для определения назначения штекеров найдите на их корпусах условные обозначения, а если они непонятны или надписей нет, то проследите, куда идет провод от того или иного разъема. При подключении светодиодных индикаторов обратите внимание на полярность подсоединения разъема. Обычно цветной проводок подключается к «+», а черный или белый – к «–» на материнской плате (разъем «+» маркируется краской).

Задание 6. **Подключение разъемов питания**

Блок питания всегда закреплен на корпусе системного блока и в процессе сборки, как правило, не снимается. Исключе-

нием могут быть случаи, когда используется малогабаритный корпус и блок питания мешает установке материнской платы.

В первую очередь следует подключить разъем кулера (вентилятора системы охлаждения процессора). Обычно разъем питания кулера представлен в виде миниатюрного трехконтактного разъема, который следует присоединить к аналогичному разъему на материнской плате.

Затем подключается основной разъем блока питания. Разъем на кабеле должен до конца пойти в ответную часть на материнской плате, а защелка разъема надежно зафиксироваться.

В ряде случаев для питания процессора используется дополнительный разъем +12 В. Необходимость такого варианта надо уточнить по документации, четырехконтактный разъем для данной цепи всегда находится около сокета процессора.

Задание 7. **Установка накопителей**

Установка накопителя на гибких магнитных дисках, жесткого диска, а также привода CD/DVD-дисков производится по одним и тем же правилам.

Прежде всего, накопители нужно установить в соответствующие их физическим размерам отсеки корпуса системного блока и надежно закрепить винтами. После этого необходимо подключить их к соответствующим интерфейсным разъемам и разъемам питания.

Для подключения жестких дисков на системной плате устанавливаются один или два 40-контактных разъема для IDE-интерфейса или несколько SATA-разъемов. Для всех современных IDE-винчестеров требуется использовать 80-жильный шлейф. При использовании 40-жильного шлейфа винчестер не сможет передавать информацию с максимально доступной ему скоростью.

При установке жесткого диска, прежде всего, необходимо с помощью перемычек, расположенных на задней стороне корпуса, задать режим его работы: Master или Slave (режим Cable используется крайне редко). Для единственного системного жесткого диска следует устанавливать режим Master. Далее к жесткому диску подключаются интерфейсный шлейф и разъем питания. Практически всегда первый провод (красный или черный) шлей-

фа находится ближе к разъему питания жесткого диска.

Системные платы, поддерживающие интерфейс SATA, имеют дополнительные разъемы для SATA-интерфейса. К каждому такому разъему подключается только одно устройство с помощью кабеля SATA. Для подключения питания используется переходник, который одним концом присоединяется к разъему питания жесткого диска SATA, а другим – к разъему на кабеле блока питания. *Винчестеры с интерфейсом SATA джамперов для выбора типа устройства не имеют*.

Привод CD/DVD-дисков подключается точно так же, как и жесткий диск. Но для повышения производительности компьютера желательно подключать привод CD/DVD-дисков на второй канал (разъем) IDE-интерфейса. Для привода CD-RW лучше устанавливать режим Master. При установке двух оптических приводов, например CD-RW и DVD, их разрешается подключить на один шлейф. Один привод должен работать в режиме Master, а второй – в Slave.

На приводе CD/DVD-дисков имеются дополнительные разъемы, предназначенные для подключения аналогового аудиокабеля к звуковой плате. Этот кабель можно не устанавливать, если не предполагается прослушивать через компьютер диски типа аудио-CD.

Задание 8. Установка плат расширения

В зависимости от типа материнской платы на ней могут быть установлены слоты различных интерфейсов: ISA, PCI, AGP, PCI Express. Поэтому перед установкой на материнскую плату нужно внимательно рассмотреть платы расширения и определить, в какие слоты они могут быть установлены. После того как соответствие плат расширения и слотов установлено, можно переходить к сборке.

При установке платы расширения следует аккуратно ввести ножевой разъем платы в слот, а потом с небольшим усилием вставить разъем в слот до упора. Плату расширения необходимо зафиксировать, иначе в процессе эксплуатации компьютера печатная плата рано или поздно вылезет из слота. Крепится плата с помощью винта, притягивающего заднюю металлическую планку платы расширения к корпусу системного блока.

Для надежной фиксации видеоконтроллеров в слотах расширения AGP и PCI Express (формата 16x) устанавливают специальные держатели с защелкой. В этом случае при установке видеокарты в слот нужно сначала отогнуть фиксатор в сторону, установить плату, а затем отпустить фиксатор. При этом можно услышать характерный щелчок, свидетельствующий о правильной установке и закреплении платы.

Задание 9. **Проверка работоспособности компьютера**

После сборки системного блоки, не закрывая корпуса компьютера, подключите к нему клавиатуру, монитор и мышь.

Остальные внешние устройства для первого включения компьютера подключать не следует.

При включении питания следует обратить внимание на звуковые сигналы, которые издает внутренний динамик, и на светодиоды, расположенные на системном блоке и клавиатуре.

На системном блоке должен засветиться светодиод, который индицирует переход компьютера в рабочее состояние. Одновременно должны заработать все вентиляторы системного блока: в блоке питания, кулеры процессора, видеокарты и чипсета (если они там есть). На некоторых моделях системных плат вентилятор процессора включается не сразу, а спустя несколько секунд. Если вентиляторы не заработали, то снова проверьте правильность сборки системного блока. Обратите внимание на корректность подключения разъемов питания и кнопки «Сеть» (Power SW).

Если после включения питания внутренний динамик издает длительный или многократные сигналы, то по таблице звуковых сигналов BIOS (см. руководство к материнской плате) следует определить неисправный узел. У версий BIOS, разработанных различными фирмами, таблицы сигналов довольно сильно отличаются. Определив неисправность, следует устранить ее путем исправления допущенных в ходе сборки ошибок или замены неисправных узлов.

2.3. Содержание отчета

Перечень и краткое описание компонентов системного блока.

Перечень ошибок и неисправностей, обнаруженных при сборке и проверке работоспособности системного блока.

Письменные ответы на контрольные вопросы.

2.4. Контрольные вопросы

1. Какие основные блоки и платы входят в состав персонального компьютера?
2. Какова стандартная мощность блока питания? Какие напряжения выдает блок питания? В чем основные причины выхода из строя блока питания?
3. Какие типы корпусов применяются в ПК?
4. Какие основные устройства находятся на материнской плате?
5. Для чего служит постоянное запоминающее устройство?
6. Какими основными параметрами характеризуются накопители на жестких дисках?
7. Какие современные типы интерфейсов применяются для НЖМД?
8. Что такое шина на системной плате? Какие типы шин вы знаете?
9. Для чего нужна кэш-память? На каких микросхемах она реализуется?
10. Какие типы модулей ОЗУ применяются в современных ПК?
11. Каковы основные параметры CD-ROM-приводов?
12. Каковы основные характеристики процессоров?
13. Какие виды накопителей на лазерных дисках вы знаете?
14. Какие виды дисковых накопителей вы знаете?
15. Что такое порт? Какие вы знаете типы портов?
16. Структура жесткого диска.
17. Какие виды памяти вы знаете?
18. Что такое BIOS? Основные функции BIOS.
19. Какие основные характеристики системной платы вы можете назвать?
20. Какое устройство служит для вывода графической информации?

Литература:

1. Богданов, Д. С. Архитектура компьютера / Д. С. Богданов. – URL : http://asm-pc.ru

2. Гук, М. Ю. Аппаратные средства IBM PC : энциклопедия / М. Ю. Гук. – 3-е изд. – СПб. : Питер, 2006. – 1072 с.

3. Докторов, А. Е. Архитектура ЭВМ. Методические указания к лабораторным работам / А. Е. Докторов, Е. А. Докторова. – УлГТУ, 2008. – 32 с.

4. Коваль, А. С. Архитектура ЭВМ и систем : учебно-методическое пособие для вузов / А. С. Коваль, А. В. Сычев. – Воронеж, 2007. – 87 с.

5. Манохин, Ю. П. Архитектура ЭВМ и систем : учебно-методический комплекс / Ю. П. Манохин, Ю. В. Чекмарев. – М. : РГТУ, 2010. – 45 с.

6. Поворознюк, А. И. Архитектура компьютеров. Архитектура микропроцессорного ядра и системных устройств : учебное пособие / А. И. Поворознюк. – Ч. 1. – Харьков : Торнадо, 2004. – 355 с.

7. Таненбаум, Э. Архитектура компьютера / Э. Таненбаум. – 5-е изд. – СПб. : Питер, 2007. – 844 с.

8. Толстобров, А. П. Архитектура ЭВМ : учебное пособие / А. П. Толстобров. – Воронеж, 2004. – 95 с.

9. Хамахер, К. Организация ЭВМ / К. Хамахер, З. Вранешич, С. Заки. – 5-е изд. – СПб. : Питер, 2003. – 848 с.

РАБОТА 3.
ОТЛАДЧИК DEBUG КАК СРЕДСТВО ДЛЯ ОЗНАКОМЛЕНИЯ С АРХИТЕКТУРОЙ INTEL 8086

Цель: научиться использовать программу DEBUG для исследования работы виртуального режима процессора Intel.

Технические средства: персональный компьютер, оснащенный операционной системой DOS или WINDOWS, отладчик двоичного кода DEBUG.

3.1. Краткая теоретическая часть

Изучая работу процессора персонального компьютера, удобно использовать программу DEBUG, встроенную в командную оболочку операционной системы WINDOWS. Эта программа представляет собой шестнадцатиразрядный отладчик кода программ.

Запуск программы DEBUG

Запустить программу DEBUG можно из командной строки или непосредственно из папки, в которой она находится. Чтобы запустить программу из командной строки, выберите команду из меню ПУСК – ВЫПОЛНИТЬ или нажмите комбинацию клавиш WIN + R. В открывшемся окне напечатайте слово «debug» и нажмите клавишу ENTER или щелкните кнопку ОК.

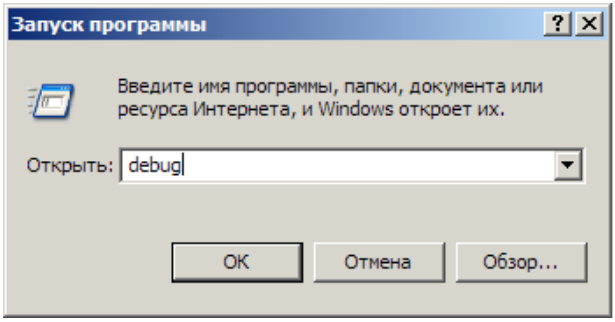


Рис. 1. *Запуск программы Debug*

В качестве приглашения выступает знак «минус».

Использование программы DEBUG

Теперь можно вводить команды программы, например, для вывода справки нужно ввести знак вопроса «?» и нажать клавишу «Enter».

Команды можно вводить как в верхнем, так и в нижнем регистре. Все числовые значения являются шестнадцатеричными-

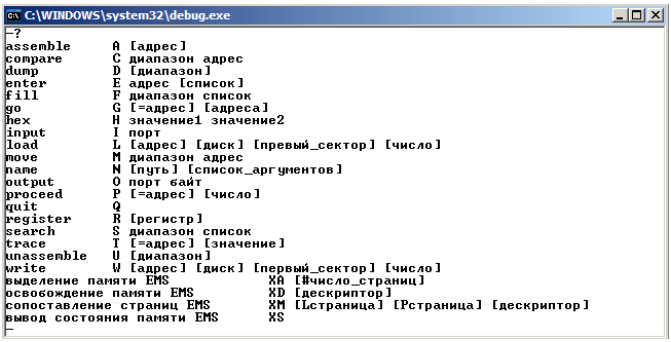


Рис. 2. *Вывод справки*

Чтобы вывести дамп памяти с адреса 0B2B:0100 до адреса 0B2B:0200 требуется ввести команду «D 100 200».

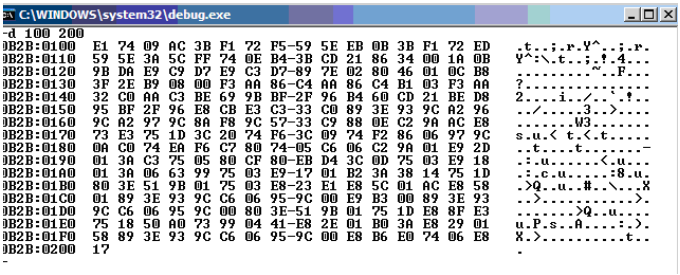


Рис. 3. *Приветствие командной строки отладчика Debug*

Для запуска программы на выполнение есть несколько путей:

1. Запустить программу командой «Go». В этом случае в командный интерпретатор удастся вернуться только после завершения всей программы.

2. Использовать команду «Trace». Она позволит выполнять последовательно каждую ассемблерную команду.

3. Использовать команду «Proceed». Так же, как и «Trace», выполняет по одной инструкции, но выполнение инструкций CALL, LOOP, INT или повторяемой строковой инструкции с префиксами REPnn происходит как выполнение одной команды.

Например, для выполнения одной инструкции, находящейся в памяти по адресу XXXX:0100, следует изменить значение регистра IP на 100 командой «R IP» и выполнить команду «T».

Таблица 3.1

Команды отладчика DEBUG

Команда	Описание
A (Assemble)	Транслирование команд ассемблера в машинный код; адрес по умолчанию – CS:0100h. A [<адрес_начала_кода>]
C (Compare)	Сравнение содержимого двух областей памяти; по умолчанию используется DS. В команде указывается либо длина участков, либо диапазон адресов. C <начальный_адрес_1> L<длина> <начальный_адрес_2> C <начальный_адрес_1> <конечный_адрес_1> <начальный_адрес_2>
D (Display/ Dump)	Вывод содержимого области памяти в шестнадцатеричном и ASCII-форматах. По умолчанию используется DS; можно указывать длину или диапазон. D [<начальный_адрес> [L<длина>]] D [начальный_адрес конечный_адрес]
E (Enter)	Ввод в память данных или инструкции машинного кода; по умолчанию используется DS. E [<адрес> [<инструкции/данные>]]

Продолжение таблицы 3.1

Команда	Описание
F (Fill)	Заполнение области памяти данными из списка; по умолчанию используется DS. Использовать можно как длину, так и диапазон. F <начальный_адрес_1> L<длина> ‘<данные>’ F <начальный_адрес> <конечный_адрес> ‘<данные>’
G (Go)	Выполнение отлаженной программы на машинном языке до указанной точки останова; по умолчанию используется CS. При этом убедитесь, что IP содержит корректный адрес. G [=<начальный_адрес>] <адрес_останова> [<адрес_останова> ...]
H (Hexadecimal)	Вычисление суммы и разности двух шестнадцатеричных величин. H <величина_1> <величина_2>
I (Input)	Считывание и вывод одного байта из порта. I <адрес_порта>
L (Load)	Загрузка файла или данных из секторов диска в память; по умолчанию – CS:100h. Файл можно указать с помощью команды N или аргумента при запуске debug.exe. L [<адрес_в_памяти_для_загрузки>] L [<адрес_в_памяти_для_загрузки> [<номер_диска> <начальный_сектор> <количество_секторов>]]
M (Move)	Копирование содержимого ячеек памяти; по умолчанию используется DS. Можно указывать как длину, так и диапазон. M <начальный_адрес> L<длина> <адрес_назначения> M <начальный_адрес> <конечный_адрес> <адрес_назначения>
N (Name)	Указание имени файла для команд L и W. N <имя_файла>
O (Output)	Отсылка байта в порт. O <адрес_порта> <байт>

Окончание таблицы 3.1

Команда	Описание
P (Proceed)	Выполнение инструкций CALL, LOOP, INT или повторяемой строковой инструкции с префиксами REPnn, переходя к следующей инструкции. P [=<адрес_начала>] [<количество_инструкций>]
Q (Quit)	Завершение работы debug.exe
R (Register)	Вывод содержимого регистров и следующей инструкции. R <имя_регистра>
S (Search)	Поиск в памяти символов из списка; по умолчанию используется DS. Можно указывать как длину, так и диапазон. S <начальный_адрес> L<длина> ‘<данные>’ S <начальный_адрес> <конечный_адрес> ‘<данные>’
T (Trace)	Пошаговое выполнение программы. Как и в команде P, по умолчанию используется пара CS:IP. Замечу, что для выполнения прерываний лучше пользоваться командой P. T [=<адрес_начала>] [<количество_выполняемых_команд>]
U (Unas-semble)	Дизассемблирование машинного кода; по умолчанию используется пара CS:IP. К сожалению, debug.exe некорректно дизассемблирует специфические команды процессоров 80286+, хотя они все равно выполняются корректно. U [<начальный_адрес>] U [<начальный_адрес конечный_адрес>]
W (Write)	Запись файла из debug.exe; необходимо обязательно задать имя файла командой N, если он не был загружен. А программы записываются только в виде файлов .COM! Число байт записываемой информации должно содержаться в регистре CX. W [<адрес> [<номер_диска> <начальный_сектор> <количество_секторов>]]

3.2. Практическая часть

Задание 1. *При помощи отладчика Debug заполнить таблицу 3.2.*

Таблица 3.2

Отчет выполнения работы с программой DEBUG

1. Вывести на экран содержимое регистров	
2. Вывести на экран 139_{10}, байт памяти, начиная с 100H	
3. Присвоить имя программе	
4. Записать на диск программу, состоящую из кода, находящегося по адресу $2B3_{16}$ размером 134_{10} байт, и данных, находящихся по адресу 340_{16} размером 200_{10} байт	

3.3. Контрольные вопросы

1. Используя отладчик DEBUG в режиме виртуальной адресации, можно ли повредить операционную систему?

2. Содержимое каких регистров можно узнать при помощи отладчика DEBUG?

РАБОТА 4.
МИКРОПРОЦЕССОР И ПАМЯТЬ КОМПЬЮТЕРА

Цель: изучить устройство процессора и оперативной памяти персонального компьютера.

Оборудование: макет материнской платы, процессоры, модули оперативной памяти.

4.1. Краткая теоретическая часть

В 1945 году математик Джон фон Нейман описал основы конструкции компьютера. Согласно принципам фон Неймана, компьютер должен иметь следующие устройства.

Арифметико-логическое устройство – для непосредственного осуществления вычислений и логических операций.

Устройство управления – для организации процесса управления программ.

Запоминающее устройство (память) – для хранения программ и информации.

Внешние устройства – для ввода и вывода информации.

Схема устройства современных компьютеров несколько отличается от классической схемы. В частности, арифметико-логическое устройство и устройство управления, как правило, объединены в центральный процессор (ЦПУ). На схеме приведены основные части современного компьютера (рис. 4).

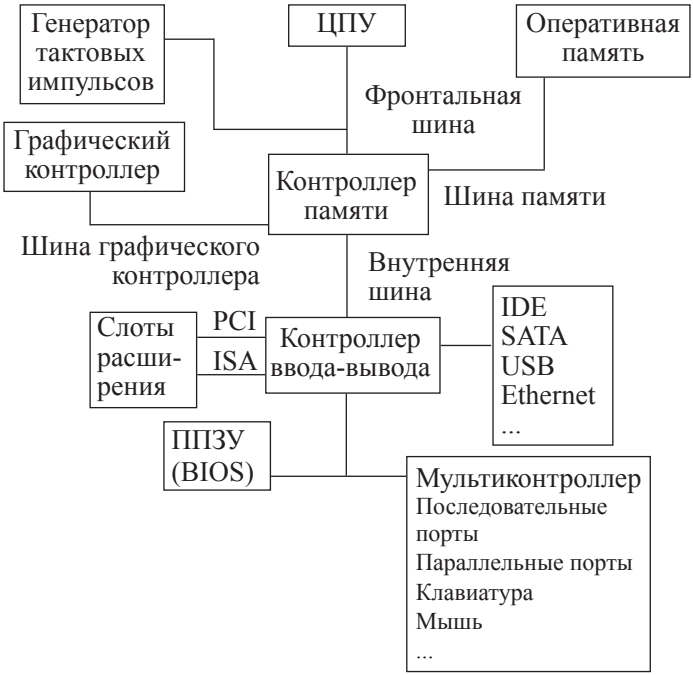


Рис. 4. *Структура материнской платы персонального компьютера*

Процессор. Процессор (рис. 5) делится на два логических устройства: устройство исполнения (УИ) и шинный интерфейс (ШИ). УИ ответственно за выполнение инструкций, а ШИ – за доставку УИ данных и инструкций для обработки. УИ содержит арифметико-логическое устройство (АЛУ, английская аббревиатура – ALU), устройство управления (УУ) и регистры. Это позволяет выполнять арифметические и логические операции.

Регистры процессора позволяют выполнять арифметические операции, адресацию памяти и управлять исполнением инструкций. Программа обращается к регистрам по имени, например, CS, DS, SS. Биты в регистре нумеруются справа налево, начиная с 0:

...15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

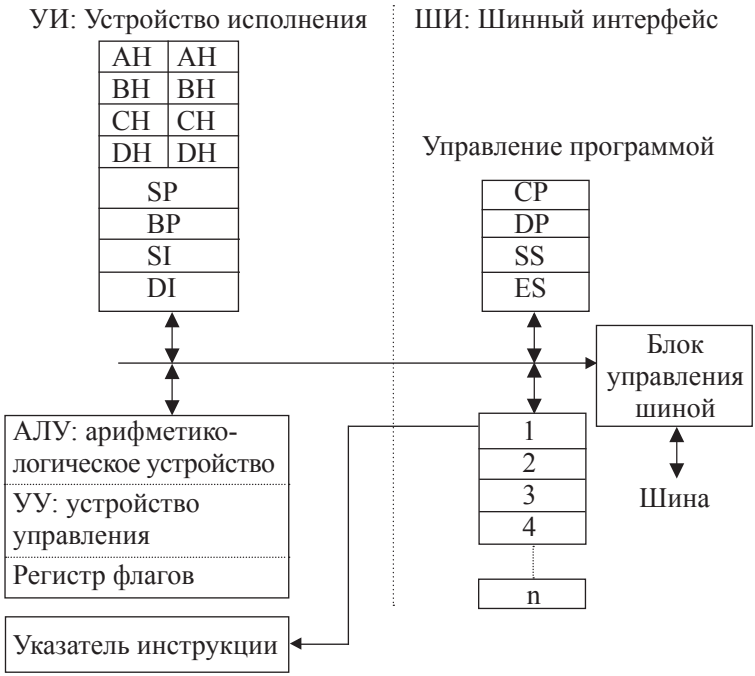


Рис. 5. *Устройство процессора Intel 8086*

Сегментный регистр адресует данные в области памяти, называемой текущим сегментом. Процессоры фирмы Intel, применяемые в IBM PC – совместимых компьютерах, обладают различными возможностями адресации.

Регистр CS. Имеет начальный адрес сегмента, содержащего код программы. Сумма содержимого этого регистра со значением смещения в указателе инструкций (instruction pointer, IP) указывает адрес инструкции, которая должна быть передана для исполнения. Для обычных целей программирования прямо ссылаться на этот регистр не нужно.

Регистр DS. Содержит начальный адрес сегмента данных программы. Инструкции используют этот адрес для поиска данных. Этот адрес, сложенный со смещением, заданным в инструкции, дает ссылку на определенную ячейку памяти, содержащую требуемые данные.

Регистр SS. Позволяет реализовывать в памяти стек, который программа использует для временного хранения адресов и данных. Регистр SS хранит начальный адрес сегмента стека программы. Этот адрес в сумме со смещением, хранимым в указателе стека (stack pointer, SP), указывает на текущее слово в стеке. Для обычных целей программирования прямо ссылаться на этот регистр не нужно.

Регистр ES. Используется некоторыми строковыми (работающими с символьными данными) операциями для управления адресацией памяти. В этом контексте сегментный регистр ES связан с регистром индекса (index, DI). Если программа требует использования сегмента ES, вы должны проинициализировать регистр соответствующим значением начального адреса.

Регистры указателей – это 32-разрядные регистры EIP, ESP, EBP; их младшие 16 разрядов – соответственно IP, SP, BP.

Указатель инструкции *(instruction pointer, IP)*. 16-разрядный регистр IP содержит смещение адреса следующей инструкции, подлежащей исполнению. IP связан с CS (как CS:IP), то есть регистр IP указывает текущую инструкцию внутри текущего кодового сегмента. Обычно к IP не обращаются непосредственно в программах, но его значение можно изменять, например, в программе DEBUG с целью тестирования выполнения программы. В процессоре 80386 впервые появился расширенный указатель инструкции – 32-разрядный регистр EIP.

В следующем примере CS содержит 39B4[0]H. IP содержит 514H. Чтобы определить следующую подлежащую исполнению инструкцию, процессор суммирует адрес CS со смещением в IP.

Адрес следующей выполняемой инструкции

CS	39B40H
Смещение в IP	+ 514H
Адрес следующей выполняемой инструкции	3A054H

При выполнении каждой инструкции процессор меняет значение смещения в регистре IP, и этот регистр постоянно указывает на следующую подлежащую исполнению инструкцию.

Указатель стека *(stack pointer, SP)*. Регистры SP (*stack pointer* – указатель стека) и BP (*base pointer* – базовый указатель), связанные регистром SS, и позволяют системе работать с данными в стековом сегменте. Процессор автоматически работает с этими регистрами.

16-разрядный регистр SP хранит значение смещения, которое в сочетании с регистром SS (SS:SP) указывает на текущее слово в стеке. В процессоре 80386 введен в оборот 32-разрядный расширенный регистр указателя стека – ESP.

В следующем примере регистр SS содержит адрес сегмента 4BB3[0]H, а регистр SP – смещение 412H. Для вычисления адреса текущего слова в стеке процессор суммирует значения регистров SS и SP.

Адрес вершины стека

SS	4BB30H
Смещение в SP	+ 412H
Адрес текущего слова в стеке	4BF42H

Базовый указатель *(base pointer, BP)*. 16-разрядный регистр BP обрабатывает такие ссылочные параметры, как данные и адреса, передаваемые программой через стек. Процессор сочетает адрес в SS со смещением в BP. BP может также сочетаться с DI и SI как базовый регистр специальной адресации. В процессоре 80386 впервые появился расширенный 32-разрядный регистр базового указателя – EBP.

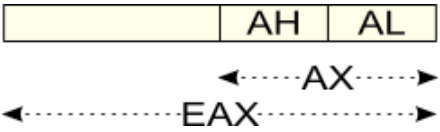
Регистры общего назначения. 32-разрядные регистры общего назначения – это EAX, EBX, ECX и EDX; их младшие 16 разрядов обозначаются соответственно AX, BX, CX и DX. Каждый из последних, в свою очередь, состоит из двух байтов, например, AX – из старшего AH и младшего AL. Это свойство позволяет обращаться к разным частям регистра для обработки байтов, слов и двойных слов.

Следующие ниже инструкции на языке ассемблера иллюстрируют запись нулей в регистры AX, BH и ECX:

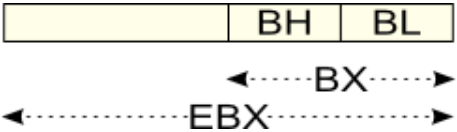
```
MOV AX, 00
MOV BH, 00
```

MOV ECX, 00

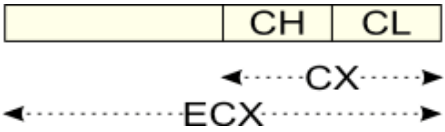
Регистр AX – основной аккумулятор, используется в арифметических операциях и операциях ввода/вывода. Например, инструкции сложения и умножения предполагают использование регистра AX. Кроме того, некоторые инструкции преобразуются в более эффективный машинный код, если они используют именно регистр AX. 8-разрядные AH и AL являются левой и правой частями AX, а сам AX – это правая часть регистра EAX.



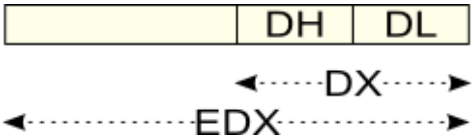
Регистр BX называют базовым регистром, поскольку это единственный регистр общего назначения, который можно использовать в качестве указателя (индекса) для расширения адресации. Другое частое применение этого регистра – для вычислений. BX может также быть использован в сочетании с DI или SI как базовый регистр для специальной адресации. Восьмиразрядные BH и BL являются левой и правой частями BX, и сам BX – это правая часть регистра EBX.



Регистр CX используется как регистр-счетчик. Он может содержать число повторений цикла в программе или величину, на которую нужно сдвинуть биты влево или вправо. CX можно также использовать для различных вычислений. Восьмиразрядные CH и CL являются левой и правой частями CX, и сам CX – это правая часть регистра ECX.



Регистр DX – регистр данных. Он используется в некоторых операциях ввода/вывода, а операции умножения и деления больших чисел используют AX и DX в паре. Восьмиразрядные DH и DL являются левой и правой частями DX, и сам DX – это правая часть регистра EDX.



Регистры общего назначения доступны для сложения и вычитания 8-,16- и 32-разрядных двоичных чисел:

```
MOV EAX, 337 ; поместить число 337 в EAX (двойное слово)
ADD CX, AL ; прибавить AX к CX (слово)
SUB BL, AL ; отнять AL от BL (байт)
```

Регистры индекса – это 32-разрядные ESI и EDI. Их младшие слова (16-разрядные регистры) – соответственно, SI и DI. Эти регистры доступны для индексированной адресации и используются для достижения определенных целей при сложении и вычитании.

Регистр SI. 16-разрядный регистр индекса источника (source index register) используется в некоторых строковых (символьных) операциях. В этом случае SI связан с регистром DS. Процессор 80386 впервые представил расширенный 32-разрядный регистр ESI.

Регистр DI. 16-разрядный регистр индекса назначения (destination index register) также используется в строковых операциях. В этом случае DI связан с регистром ES. Процессор 80386 впервые представил расширенный 32-разрядный регистр EDI.

Регистр флагов. Биты 32-разрядного регистра Eflags указывают на состояние различных процессов и компонентов системы. Младшие 16 разрядов регистра Eflags – это, собственно, регистр Flags. Девять из его шестнадцати разрядов указывают на текущее состояние компьютера и результаты выполнения инструкций. Многие инструкции, например, арифметические

и сравнения, изменяют состояние этих битов (флагов), и последующие инструкции могут их проверять для определения дальнейших действий.

Далее кратко описаны эти флаги.

OF (*overflow,* переполнение). Указывает на переполнение старшего разряда после арифметической операции.

DF (*direction*, направление). Определяет правое или левое направление движения при сравнении или перемещении строковых (символьных) данных.

IF (*interrupt*, прерывание). Указывает, будут ли внешние прерывания, например, ввод с клавиатуры, обработаны или проигнорированы.

TF (*trap,* остановка). Разрешает работу процессора в пошаговом режиме. Программы-отладчики, такие, как DEBUG, устанавливают этот флаг для того, чтобы вы могли детально отслеживать изменения содержимого регистров и других параметров после каждой выполненной инструкции.

SF (*sign,* знак). Содержит знак результата последней арифметической операции (0 – положительный, 1 – отрицательный).

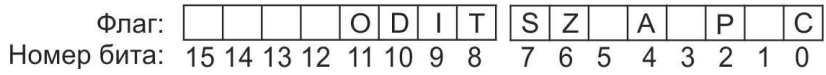
ZF (*zero,* ноль). Указывает результат последней арифметической операции (0 – ненулевой результат, 1 – нулевой результат).

AF (*auxiliary carry,* вспомогательный перенос). Содержит бит, переносимый из третьего бита в четвертый бит в специализированных арифметических операциях.

PF (*parity,* четность). Указывает на число единичных битов в результате последней операции. Четное число битов устанавливает положительную четность, нечетное – отрицательную.

CF (*carry,* перенос). Содержит перенос из старшего разряда после последней арифметической операции. Содержит также значение последнего бита после операции сдвига и циклического сдвига (вращения).

Флаги в регистре Flags расположены в порядке, показанном ниже:



Флаги OF, SF, ZF и CF наиболее важны для программиро-

вания на ассемблере арифметических операций и операций сравнения, а DF – для строковых операций. Некоторые другие флаги используются для внутренних нужд, в основном связанных с защищенным режимом работы.

Память. Два типа внутренней памяти PC – это постоянная память (ПЗУ) и оперативная память (ОЗУ). Последняя также называется памятью с произвольным доступом. Байты в памяти нумеруются последовательно, начиная с 0, и каждый байт имеет свой уникальный адрес.

На рисунке 6 показана структура внутренней памяти PC с процессором 8086 в качестве простого примера. Из первого мегабайта памяти 640 килобайт – основная оперативная память, большая часть которой доступна для непосредственного использования.

Таблица 4.1

Структура ПЗУ персонального компьютера

Начало (десятичное)	Адрес (шестнадцатеричный)	Объем и назначение
960K	F0000	64 K – основное ПЗУ системы
768 K	C0000	192 K – расширенная область (ПЗУ)
640 K	A0000	128 K – видеопамять (ОЗУ)
0	00000	640 K (ОЗУ) – область BIOS. Таблица векторов прерываний

ПЗУ состоит из специальных чипов, информация из которых, будучи однажды записана, в дальнейшем только считывается. В силу этого свойства записанные в таких чипах данные и инструкции не могут быть изменены (современные чипы ППЗУ – перезаписываемые, то есть могут быть изменены с помощью специальной программы). ПЗУ BIOS (Basic Input/Output System – основная система ввода/вывода) начинается с адреса 768 K и отвечает за устройства ввода/вывода, например,

контроллер винчестера (жесткого диска). ПЗУ, начинающаяся с адреса 960 K, отвечает за основные функции компьютера, например, самопроверку при включении, рисование точек при выводе графики и загрузку операционной системы с диска. Когда вы включаете компьютер, ПЗУ управляет различными проверками и загрузкой специальных данных системы с диска в ОЗУ.

ОЗУ. Программиста в основном интересует оперативная память (ОЗУ – оперативное запоминающее устройство). В нее можно записывать и из нее можно считывать информацию. Память ОЗУ доступна в качестве «рабочего пространства» для временного хранения данных и выполнения программ. Когда вы включаете компьютер, часть операционной системы загружается с диска в ОЗУ. Выполняемая программа находится в памяти и обычно выводит информацию на экран, принтер или диск. После завершения программы операционная система может загрузить на ее место другую программу.

Выключение компьютера приводит к потере данных в ОЗУ, но не влияет на ПЗУ. Поэтому, если вы редактировали документ, вы должны сохранять его на диске перед выключением компьютера.

Адресация данных в памяти. В зависимости от модели процессор может обращаться к одному или более байтам памяти. Рассмотрим десятичное число 1315. Шестнадцатеричное представление этого числа 0529H занимает два байта или одно слово в памяти. Слово состоит из старшего (более значащего) байта 05 и младшего (менее значащего) байта 29. Процессор хранит данные в памяти в инверсном порядке (reverse-byte sequence), то есть младший байт – в ячейке с меньшим адресом, а старший байт – в ячейке с большим адресом. Например, процессор передает значение 0529H из регистра в память по адресам 04A26H и 04A27H так:

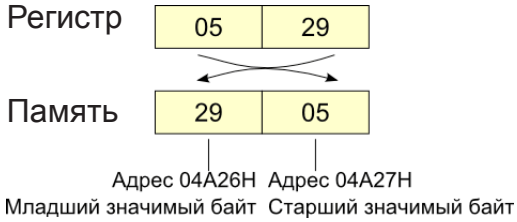


Рис. 6. *Передача данных из регистра в память*

Процессор подразумевает, что данные в памяти хранятся именно в инверсном порядке, и обрабатывает их соответствующим образом. Когда процессор извлекает данные из памяти, он переставляет байты, помещая их в регистр в виде hex 05 29. Хотя этот процесс полностью автоматизирован, вы должны помнить о нем при написании и отладке программ на ассемблере. Для записи числа в память компьютера предусмотрена ассемблерная команда *mov*, в качестве одного из аргументов которой в квадратных скобках записывается адрес в памяти, по которому нужно произвести запись или чтение, причем в память невозможно записать непосредственное число, можно записать только содержимое какого-либо регистра. Также для того, чтобы поместить данные в память, можно использовать команду *push*, единственным аргументом которой является регистр, который необходимо сохранить в памяти. Данные сохраняются по адресу, записанному в регистре SP – указатель стека. Для чтения данных из стека предусмотрена команда *pop*, аргументом которой является регистр, в который необходимо записать значение из памяти.

Компьютерная шина (от англ. «*computer bus, bidirectional universal switch*» – двунаправленный универсальный коммутатор) – в архитектуре компьютера подсистема, которая передает данные между функциональными блоками компьютера. Обычно шина управляется драйвером. В отличие от связи «точка–точка» к шине можно подключить несколько устройств по одному набору проводников. Каждая шина определяет свой набор коннекторов для физического подключения устройств, карт и кабелей.

Ранние компьютерные шины представляли собой параллельные электрические шины с несколькими подключениями, но сейчас данный термин используется для любых физических механизмов, предоставляющих такую же логическую функциональность, как параллельные компьютерные шины. Современные компьютерные шины используют как параллельные, так и последовательные соединения и могут иметь параллельные (multidrop) и цепные (daisy chain) топологии. В случае

USB и некоторых других шин могут также использоваться хабы (концентраторы).

Шина адреса – компьютерная шина, используемая центральным процессором или устройствами, способными инициировать сеансы DMA, для указания физического адреса слова ОЗУ (или начала блока слов), к которому устройство может обратиться для проведения операции чтения или записи.

Основной характеристикой шины адреса является ее ширина в битах. Ширина шины адреса определяет объем адресуемой памяти. Например, если ширина адресной шины составляет 20 бит и размер слова памяти равен одному байту (минимальный адресуемый объем данных), то объем памяти, который можно адресовать, составляет 220 = 1 048 576 байтов (1 МБайт) как в IBM PC/XT.

С точки зрения архитектуры микропроцессорной системы, если не применять мультиплексирование, каждый бит в адресе определяется одним проводником (линией) в магистрали, по которой передается адрес.

Если рассматривать структурную схему микро-ЭВМ, то адресная шина активизирует работу всех внешних устройств по команде, которая поступает с микропроцессора.

Шина данных – шина, предназначенная для передачи информации. В компьютерной технике принято различать выводы устройств по назначению: одни для передачи информации (например, в виде сигналов низкого или высокого уровня), другие для сообщения всем устройствам (шина адреса) – кому эти данные предназначены.

На материнской плате шина может также состоять из множества параллельно идущих через всех потребителей данных проводников (например, в архитектуре IBM PC).

Основной характеристикой шины данных является ее ширина в битах. Ширина шины данных определяет количество информации, которое можно передать за один такт.

4.2. Практическая часть

Задание 1. Запустите программу DEBUG и убедитесь, что регистр IP содержит значение 0100.

Задание 2. С помощью команды A отладчика составьте программу, помещающую нули во все регистры общего назначения. В конце программы введите «int 20», что значит конец программы.

Задание 3. С помощью команды R поместите в регистры общего назначения случайные ненулевые числа.

Задание 4. Выполните пошагово написанную программу, анализируя каждое действие. Следите за изменяемыми регистрами.

Задание 5. Измените программу так, чтобы к текущему значению регистров общего назначения прибавлялось число $F000_{16}$ и записывалось в память, а только после этого регистры обнулялись. Следите за переполнением при сложении чисел. Запись в память производить при помощи ассемблерной команды *push*.

Задание 6. Измените программу так, чтобы к текущему значению регистров общего назначения прибавлялось число $F000_{16}$ и записывалось в память, а только после этого регистры обнулялись. Следите за переполнением при сложении чисел. Запись в память производить при помощи ассемблерной команды *mov*.

4.3. Форма отчета

Таблица 4.2

Форма отчета выполняемых действий

Команда	Выполняемое действие	Изменяемые регистры при выполнении команды

РАБОТА 5.
ПРОГРАММНЫЕ И АППАРАТНЫЕ ПРЕРЫВАНИЯ

Цель: научиться использовать программные и аппаратные прерывания.

5.1. Краткая теоретическая часть

Прерывание (англ. «*interrupt*») – сигнал, сообщающий процессору о наступлении какого-либо события. При этом выполнение текущей последовательности команд приостанавливается и управление передается обработчику прерывания, который реагирует на событие и обслуживает его, после чего возвращает управление в прерванный код.

В зависимости от источника возникновения сигнала прерывания делятся на:

– *асинхронные, или внешние* (аппаратные), – события, которые исходят от внешних источников (например, периферийных устройств) и могут произойти в любой произвольный момент: сигнал от таймера, сетевой карты или дискового накопителя, нажатие клавиш клавиатуры, движение мыши. Факт возникновения в системе такого прерывания трактуется как запрос на прерывание (англ. «*Interrupt request, IRQ*»);

– *синхронные, или внутренние,* – события в самом процессоре как результат нарушения каких-то условий при исполнении машинного кода: деление на ноль или переполнение, обращение к недопустимым адресам или недопустимый код операции;

– *программные* (частный случай внутреннего прерывания) – инициируются исполнением специальной инструкции в коде программы. Программные прерывания, как правило, используются для обращения к функциям встроенного программного обеспечения (firmware), драйверов и операционной системы.

Термин «ловушка» (англ. «*trap*») иногда используется как синоним термина «прерывание» или «внутреннее прерыва-

ние». Как правило, словоупотребление устанавливается в документации производителя конкретной архитектуры процессора.

В зависимости от возможности запрета внешние прерывания делятся на:

– *маскируемые* – прерывания, которые можно запрещать установкой соответствующих битов в регистре маскирования прерываний (в x86-процессорах – сбросом флага IF в регистре флагов);

– *немаскируемые* (англ. «*Non maskable interrupt, NMI*») – обрабатываются всегда, независимо от запретов на другие прерывания. К примеру, такое прерывание может быть вызвано сбоем в микросхеме памяти.

Обработчики прерываний обычно пишутся таким образом, чтобы время их обработки было как можно меньшим, поскольку во время их работы могут не обрабатываться другие прерывания, а если их будет много (особенно от одного источника), то они могут теряться.

Приоритезация. До окончания обработки прерывания обычно устанавливается запрет на обработку этого типа прерывания, чтобы процессор не входил в цикл обработки одного прерывания. Приоритезация означает, что все источники прерываний делятся на классы, и каждому классу назначается свой уровень приоритета запроса на прерывание. Приоритеты могут обслуживаться как относительные и абсолютные.

Относительное обслуживание прерываний означает, что если во время обработки прерывания поступает более приоритетное прерывание, то это прерывание будет обработано только после завершения текущей процедуры обработки прерывания.

Абсолютное обслуживание прерываний означает, что если во время обработки прерывания поступает более приоритетное прерывание, то текущая процедура обработки прерывания вытесняется, и процессор начинает выполнять обработку

вновь поступившего более приоритетного прерывания. После завершения этой процедуры процессор возвращается к выполнению вытесненной процедуры обработки прерывания.

Таблица прерываний. Вектор прерывания – закрепленный за устройством номер, который идентифицирует соответствующий обработчик прерываний. Векторы прерываний объединяются в таблицу векторов прерываний, содержащую адреса обработчиков прерываний. Местоположение таблицы зависит от типа и режима работы процессора.

Программное прерывание. Программное прерывание – синхронное прерывание, которое может осуществить программа с помощью специальной инструкции.

В процессорах архитектуры x86 для явного вызова синхронного прерывания имеется инструкция Int, аргументом которой является номер прерывания (от 0 до 255). В IBM PC-совместимых компьютерах обработку некоторых прерываний осуществляют подпрограммы BIOS, хранящиеся в ПЗУ, и это служит интерфейсом для доступа к сервису, предоставляемому BIOS. Также обслуживание прерываний могут взять на себя BIOS карт расширений (например, сетевых или видеокарт), операционная система и даже обычные (прикладные) программы, которые постоянно находятся в памяти во время работы других программ (так называемые резидентные программы). В отличие от реального режима в защищенном режиме x86-процессоров обычные программы не могут обслуживать прерывания, эта функция доступна только системному коду (операционной системе).

MS-DOS использует для взаимодействия со своими модулями и прикладными программами прерывания с номерами от 20h до 3Fh (числа даны в шестнадцатеричной системе счисления, как это принято при программировании на языке ассемблера x86). Например, доступ к основному множеству функций MS-DOS осуществляется исполнением инструкции Int 21h (при этом номер функции и ее аргумен-

ты передаются в регистрах). Это распределение номеров прерываний не закреплено аппаратно, и другие программы могут устанавливать свои обработчики прерываний вместо или поверх уже имеющихся обработчиков, установленных MS-DOS или другими программами, что, как правило, используется для изменения функциональности или расширения списка системных функций. Также этой возможностью пользуются вирусы.

Функция 09H прерывания 21H для вывода на экран требует определения строки, подлежащей выводу, в области данных и завершения этой строки символом доллара ($) или 24H, который функция воспринимает как требование завершить вывод. Недостаток этого метода в том, что вы не сможете вывести на экран этот символ.

Пример использования функции:

```
mov ah, 09h       ;Запросить вывод на экран
mov dx, 1234h ;Загрузить адрес, по которому находится строка
int 21h           ;Вызвать обработчик прерывания
```

Функция 01H прерывания 21H для ввода с клавиатуры не требует определения каких-либо параметров. На выходе функции в регистр AL заносится ASCII код введенного символа.

Пример использования функции:

```
mov ah, 01h       ;Запросить ввод с клавиатуры
int 21h           ;Вызвать обработчик прерывания
```

Инструкция сравнения значений регистров CMP

Инструкция сравнивает в двоичном виде содержимое двух полей данных. CMP вычитает значение операнда 2 из операнда 1 и устанавливает/сбрасывает флаги в зависимости от результата, но не сохраняет сам результат. Оба операнда должны

быть байтами, словами или двойными словами. CMP можно применять для сравнения регистров, элементов данных в памяти или непосредственных значений с регистром, или сравнений регистров, или непосредственных значений с памятью. Результаты сравнения могут быть следующими:

Таблица 5.1

Результат выполнения команды CMP

Значения флагов:	CF	SF	ZF
операнд 1 < операнд 2	1	1	0
операнд 1 = операнд 2	0	0	1
операнд 1 > операнд 2	0	0	0

Инструкции условного перехода JG и JL

Инструкции используют значения флагов CF, SF и ZF, которые изменяются инструкцией CMP. Таким образом, можно сначала сравнить два числа, а затем сделать условный переход в другую точку программы по результатам сравнения. Если значение первого операнда инструкции CMP меньше второго, то инструкция JL ds:адрес приведет к переходу по этому адресу, а если значение первого операнда инструкции CMP больше второго, то к переходу приведет инструкция JG ds:адрес.

Пример:

```
mov bx,0001h        ;заносим в bx значение 1
mov cx,0002h        ;заносим в cx значение 2
cmp cx,bx           ;сравниваем значения из cx и dx
jg ds:150h          ;если значение в cx больше, чем значение в bx, то переходим к адресу ds:150h, иначе переходим к следующей инструкции
jl ds:180h          ;если значение в cx меньше, чем значение в bx, то переходим к адресу ds:150h
```

5.2. Практическая часть

Задание. В отладчике DEBUG написать программу, позво-

ляющую вводить с клавиатуры одноразрядные числа, сравнивать их и выводить наибольшее из них. Сохранить программу на диск.

5.3. Форма отчета

Записать команды отладчика DEBUG и код программы в порядке выполнения.

РАБОТА 6.
ИЗУЧЕНИЕ СИСТЕМЫ КОМАНД АССЕМБЛЕРА

Цель: изучить основные виды команд ассемблера и научиться применять их при написании простых приложений с помощью отладчика DEBUG.

Технические средства: персональный компьютер, оснащенный операционной системой DOS или WINDOWS, отладчик двоичного кода Debug.

6.1. Краткая теоретическая часть

Команды языка ассемблера один к одному соответствуют командам процессора, фактически они представляют собой более удобную для человека символьную форму записи (мнемокод) команд и их аргументов. При этом одной команде языка ассемблера может соответствовать несколько вариантов команд процессора.

Кроме того, язык ассемблера позволяет использовать символические метки вместо адресов ячеек памяти, которые при ассемблировании заменяются на автоматически рассчитываемые абсолютные или относительные адреса, а также так называемые директивы (команды, не переводящиеся в процессорные инструкции, а выполняемые самим ассемблером).

Директивы ассемблера позволяют, в частности, включать блоки данных, задать ассемблирование фрагмента программы по условию, задать значения меток, использовать макроопределения с параметрами.

Каждая модель (или семейство) процессоров имеет свой набор команд и соответствующий ему язык ассемблера. Наиболее популярные синтаксисы – Intel-синтаксис и AT&T-синтаксис.

Существуют компьютеры, реализующие в качестве машинного язык программирования высокого уровня (Forth, Lisp, Эль-76), фактически в них он является языком ассемблера.

Одними из самых важных и используемых команд ассемблера являются:

– **mov** приемник, источник – команда пересылки данных. Копирует содержимое источника в приемник, источник не изменяется. *Например:* mov ax, 1 присваивает регистру ax значение 1. Команда mov ax, word ptr eax – записывает в ax слово, лежащее по адресу eax. Байт по адресу eax записывается в младшую половину ax (в al), а байт по адресу eax+1 записывается в ah (по закону Intel). Но не обязательно, записывая команды, использовать такую сложную адресацию. Например, если у нас есть переменая "y" типа longint, то при помощи следующей команды ей можно присвоить значение 10000: mov y, 10000. Можно также записать mov dword ptr y, 10000, показывая, что "y" – 32-разрядная переменная. Если записать mov dword ptr [y+10], 5000, то, начиная с адреса [y+10], в память будет записано 32-битное число 5000. Можете не бояться писать подобные вещи (прибавление константы к адресу), так как все эти выражения вычисляются на стадии компиляции и на скорость программы не влияют. Пусть, например, "y" находится по адресу 34567. Если мы запишем [y+10], то компилятор просто поймет это как [32577], так как он знает адрес переменной "y". Операнды команды mov могут быть как регистрами, так и переменными, но одновременно оба операнда не могут быть переменными;

– **xchg** операнд1, операнд2 – обменивает операнды. Например, если al=45, ah=37, то после выполнения xchg al, ah будет al=37, ah=45;

– **add** приемник, источник – выполняет сложение приемника и источника, результат заносится в приемник. Источник не изменяется. Но зато меняются флаги:

– **adc** приемник, источник – выполняет сложение приемника, источника и флага CF. Обычно эта команда используется для сложения чисел повышенной точности.

Пусть, например, у нас имеются два 64-битных числа: первое в edx:eax (младшее двойное слово в eax, старшее двойное слово в edx), второе – в ebx:ecx. Тогда после выполнения команд: add eax, ecx; adc edx, ebx; в паре регистров edx:eax будет находиться сумма этих 64-битных чисел. Когда были 16-разрядные процессоры, подобным образом складывали 32-бит-

ные числа (считая, что каждое состоит из двух 16-битных). Сейчас это не имеет смысла. Более того, процессору все равно, какие числа складывать – 16-разрядные или 32-разрядные. Скорость одинаковая. Поэтому сложение 32-битных чисел напрямую быстрее, чем с разбиением и сложением 16-битных (в два раза). Но иногда это может понадобиться (например, Турбо-Паскаль не понимает 32-битные регистры). Этим и объясняется, что 16-битные программы медленнее, чем 32-битные;

– **sub** приемник, источник – вычитает источник из приемника, результат заносит в приемник;

– **sbb** приемник, источник – вычитает из приемника значение источника, затем вычитает значение CF. Ее можно использовать для вычитания 64-битных слов;

– **inc** приемник – то же самое, что и add приемник, 1;

– **c** приемник – то же самое, что и sub приемник, 1;

– **p** операнд1, операнд2 – по сути, вычитание операнда2 из операнда1, только операнды не меняются (команда меняет только флаги). С помощью этой команды обычно выполняются условные переходы (самый очевидный способ);

– **d|or** приемник, источник – выполняет логическое побитовое И|ИЛИ над приемником и источником и помещает результат в приемник. Часто используется для выборочного обнуления|объединичивания отдельных битов. Например, команда and al, 00001111b обнулит старшие четыре бита регистра al, а младшие не изменит;

– **r** приемник, источник – логическое исключающее ИЛИ. Выполняет побитное логическое исключающее ИЛИ над приемником и источником, результат заносится в приемник. Часто используется для обнуления регистров. *Например*, xor ax, ax обнуляет регистр ax и делает это быстрее, чем mov ax, 0. Этой командой следует пользоваться для обнуления регистров. Можете не бояться экзотичности этой команды. Она будет эффективно работать на любом Intel-совместимом компьютере. Эта команда официально поддерживается Intel как команда обнуления регистра;

– **st** операнд1, операнд2 – по сути, выполняет команду and над операндами, но операнды не меняет, а меняет только флаги аналогично and;

– **t** приемник – каждый бит приемника, равный нулю, устанавливается в 1, и каждый бит, равный 1, устанавливается в 0. Флаги не меняются.

6.2. Практическая часть

Задание. Написать программу, использующую все виды команд процессора:

1) пересылка данных;
2) арифметические операции;
3) логические операции;
4) обработка блоков данных;
5) команды передачи управления;
6) команды условного перехода;
7) управление состоянием процессора.

6.3. Форма отчета

Письменное представление кода программы, комментарий к каждой команде ассемблера и отладчика DEBUG.

Литература

1. Абель, П. Ассемблер. Язык и программирование для IBM / П. Абель. – 5-е изд. – СПб. : КОРОНА-Век, 2009. – 734 с.

2. Бабич, Н. П. Компьютерная схемотехника. Методы построения и проектирования : учебное пособие / Н. П. Бабич. – Киев : МК-Прогресс, 2004. – 576 с.

3. Богданов, Д. С. Архитектура компьютера / Д. С. Богданов. – URL : http://asm-pc.ru

4. Гук, М. Ю. Аппаратные средства IBM PC : энциклопедия / М. Ю. Гук. – 3-е изд. – СПб. : Питер, 2006. – 1072 с.

5. Докторов, А. Е. Архитектура ЭВМ. Методические указания к лабораторным работам / А. Е. Докторов, Е. А. Докторова. – УлГТУ, 2008. – 32 с.

6. Коваль, А. С. Архитектура ЭВМ и систем : учебно-методическое пособие для вузов / А. С. Коваль, А. В. Сычев. – Воронеж, 2007. – 87 с.

7. Лабораторный практикум как разновидность практического занятия. – URL : http://www.profile-edu.ru/laboratornyj-praktikum-kak-raznovidnost-prakticheskogo-zanyatiya-page-1.html

8. Магда, Ю. С. Ассемблер для процессоров Intel Pentium / Ю. С. Магда. – СПб. : Питер, 2006. – 410 с.

9. Манохин, Ю. П. Архитектура ЭВМ и систем : учебно-методический комплекс / Ю. П. Манохин, Ю. В. Чекмарев. – М. : РГТУ, 2010. – 45 с.

10. Марек, Р. Ассемблер на примерах. Базовый курс. – СПб. : Наука и техника, 2005. – 240 с.

11. Описание простейших команд ассемблера. – URL : http://students.uni-vologda.ac.ru/pages/it10/2.php

12. Поворознюк, А. И. Архитектура компьютеров. Архитектура микропроцессорного ядра и системных устройств : учебное пособие / А. И. Поворознюк. – Ч. 1. – Харьков : Торнадо, 2004. – 355 с.

13. Программирование на ассемблере под различные операционные системы. – URL : http://wasm.ru

14. Руководство по самостоятельной сборке компьютера в картинках. – URL : http://overcomp.ru/sborka.html

15. Структура программы на ассемблере. – URL : http://cs.mipt.ru/docs/comp/ rus/programming/languages/asm/yurov/guide/text/structur.htm

16. Таненбаум, Э. Архитектура компьютера / Э. Таненбаум. – 5-е изд. – СПб. : Питер, 2007. – 844 с.

17. Толстобров, А. П. Архитектура ЭВМ : учебное пособие / А. П. Толстобров. – Воронеж, 2004. – 95 с.

18. Хамахер, К. Организация ЭВМ / К. Хамахер, З. Вранешич, С. Заки. – 5-е изд. – СПб. : Питер, 2003. – 848 с.

19. Юров, В. И. Ассемблер : учебник для вузов / В. И. Юров. – СПб. : Питер, 2003. – 637 с.

Учебное издание

СМИРНОВА Марина Александровна,
УТКИН Евгений Дмитриевич,
ФЕДОРОВ Олег Анатольевич,
БОГДАНОВ Дмитрий Сергеевич,
НЕКРАСОВ Максим Александрович

**ЛАБОРАТОРНЫЕ РАБОТЫ
ПО АРХИТЕКТУРЕ КОМПЬЮТЕРА**

Учебное пособие

Корректор В. А. Яковлева.
Верстка Г. С. Лосева.

Подписано в печать 21.03.2016. Бумага «Mondi».
Гарнитура «Times New Roman». Формат 60x84 1/16.
Тираж 500 (1-й завод 1–100 экз.). Объем 3,75 п. л. Заказ № 654-14.

Сахалинский государственный университет
693008, Южно-Сахалинск, ул. Ленина, 290, каб. 32.
Тел. (4242) 45-23-16, факс (4242) 45-23-17.
E-mail: polygraph@sakhgu.ru,
izdatelstvo@sakhgu.ru