МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПУТЕЙ СООБЩЕНИЯ (МИИТ)

Институт управления и информационных технологий

Кафедра «Вычислительные системы и сети»

Т.Б. ЛАРИНА

УТВЕРЖДЕНО
ционно-издательским
оветом университета

Программирование на ассемблере

Методические указания к практическим занятиям
по дисциплине «Программирование на ассемблере»
для студентов специальности
«Вычислительные машины, комплексы, системы и сети»
ИУИТ

Москва – 2005

УДК 681.3

Л-25

Ларина Т.Б. Программирование на ассемблере. Методические указания. -М.: МИИТ, 2005, - 51с.

Методические указания предназначены для проведения практических занятий и выполнения самостоятельных работ по дисциплине «Программирование на ассемблере».

Учебно-методическое издание

Ларина Татьяна Борисовна

Программирование на ассемблере

Методические указания

Подписано в печать - 09.12.05. Формат-60×84/16 Тираж - 100.

Усл.печ.л. - 3,25. Заказ - 696.

Изд.№ 305-05.

127994. Москва, ул. Образцова, 15.

Типография МИИТ

Содержание

1. Темы и содержание практических занятий

Тема 1. Представление целых чисел в памяти, регистрах процессора. Арифметика над hex-кодами

1) Запись беззнаковых кодов в двоичной и шестнадцатиричной (hex) системах. Переводы кодов: 10 ↔16↔ 2.

2) Представление беззнаковых целых чисел в hex-коде в форматах байт, слово, двойное слово. Оценка диапазона представления.

3) Представление знаковых целых чисел в hex-коде, получение дополнительного кода в форматах: байт, слово, двойное слово. Оценка диапазона представления знаковых чисел в различных форматах.

4) Переводы из 10 ↔ 16 для беззнаковых и знаковых чисел.

Упражнения:

- записать hex- представление беззнаковых чисел: 3, 127, 128, 255, 257, 32767;

- записать hex- представление знаковых чисел: -7, -127, 128, -128, -255, 1, -32768;

- сложить 1Bh и 3F1h. Результат получить в формате двойного слова. Определить десятичный эквивалент результата, рассматривая его как знаковое число;

- вычесть 1Bh и 3F1h. Результат получить в формате двойного слова;

- умножить FFFF на 16_{10} и сложить с FFFF . Результат получить в минимально необходимом для получения правильного результата формате;

- определить диапазоны десятичных чисел, которые «укладываются» в трехбайтный формат своим hex- представлением:

а) знаковые десятичные б) беззнаковые десятичные

Тема 2. Представление целых чисел в двоично-десятичном коде (BCD-код). Форматы BCD-кодов

Упражнения:

- записать беззнаковое число 63_{10} в распакованном коде (в hex);

- записать $+761_{10}$ и -761_{10} в упакованном и распакованном форматах (в hex);

- определить диапазон знаковых и беззнаковых десятичных чисел, которые «уложатся» упакованным BCD-кодом в два байта;

- записать в hex- виде BCD-коды (упакованный и распакованный варианты) следующих беззнаковых и знаковых чисел: 0, +81, 91, 763, +9998, -81.

Тема 3. Кодирование символьных данных в коде ASCII. Использование BCD-кодов для перехода от числового к символьному представлению

Упражнения:

- записать в hex- виде последовательность байтов в памяти после ввода с клавиатуры следующей последовательности символов, заканчивающейся нажатием Enter:

That's the 1-st_symbol_string !

- определить последовательность символов, которые появятся на экране, если для вывода задана следующая последовательность ASCII – кодов:

45 72 72 6F 72 20 20 21 21 0D 0A

- выписать из таблицы ASCII –кодов однобайтные hex-коды символов '0', '1', '2', . . ,'9'. Написать последовательность однобайтных hex-кодов числовых значений 0, 1, ...9. Определить разницу между символьными и числовыми кодами. Предложить алгоритм преобразования числа от 0 до 9 в код соответствующего символа.

- выписать из таблицы ASCII –кодов однобайтные hex-коды латинских символов от 'A' до 'F' и от 'a' до 'f'. Определить разницу между кодами этих прописных/ строчных букв и числовыми кодами 0A, 0B, . . . 0F. Предложить алгоритм преобразования hex-числа от A до F в код соответствующего прописного или строчного символа.

Тема 4. Регистры процессора x386 в реальном режиме. Арифметические флаги

Контрольные вопросы: каковы имена, формат и назначение каждого программно-доступных регистров процессора;

Упражнения:

- задана последовательность команд программы и исходное состояние регистров до начала ее выполнения. По результатам выполнения каждой команды написать: текущее состояние регистров и состояние арифметических флагов.

Подробно документировать получение результата команды с конкретными числовыми значениями операндов.

Исходное состояние регистров: eax= ebx= edx= esi=

Команды	Операнды (hex)	После выполнения	Флаги
add dx. bx	dx= bx=	dx=	C= , A= ,S= , Z= , P= , O=
mov esi, 4	esi=	esi=	
sub esi, edx	esi= edx=	esi= edx=	
and bx, si	bx= si=	bx= si=	
sub bl, al	bl= al=	bl= ax=	
inc edx	edx=		

Тема 5. Сегментная адресация памяти

Контрольные вопросы

1. Каковы программные составляющие адреса памяти?
2. Какие регистры задают сегментную составляющую адреса?
3. Каково максимальное значение внутрисегментного адреса (смещения)?
4. Как аппаратно вычисляется физический адрес из программных компонентов?

Упражнения:

- к какому физическому адресу приведет программный адрес 2300:0040h;
- какое значение будет в сегментном регистре CS, если кодовый сегмент был загружен в память с адреса (hex):00000, 00210, 1fc67, 1fc60;
- каковы должны быть значения в регистрах DS и CS, чтобы указываемые программные сегменты после загрузки в память разместились на расстоянии 256 байт друг от друга?
- привести несколько комбинаций программных адресов, приводящих к одному физическому адресу – 27f38.

Тема 6. Команды пересылки данных

Упражнения:

- занести в регистр число 4 в формате байта, слова, двойного слова;

- прочитать в регистр байт, слово, двойное слово из памяти, начиная с адреса DS:0006,

- поменять местами слово из регистра AX и слово в памяти с адреса ES:0006 с помощью команд XCHG и MOV;

- в начале сегмента данных размещены байты: 05, ff,1f, 01. Переместить их от начала сегмента следующим образом: 1f, 01, ff, 05

Тема 7. Арифметические команды, команды передачи управления

Контрольные вопросы

1. Чем похожи команды вычитания SUB и сравнения CMP?
2. Почему используются разные команды условного перехода при сравнении знаковых и беззнаковых чисел?
3. Каковы различия между длинным, коротким, ближним и дальним переходами?

Упражнения:

- выполнить умножение: 4 на 255, 3 на 256, 4 на 65536;
- выполнить деление: 65536 на 4, 65536 на 420, 256 на 3, 255 на 4;
- сложить 2 байта данных, находящихся в сегменте данных, начиная с адреса A1 друг за другом. Сумму в формате слова записать в сегмент данных следом за исходными байтами;
- получить разность этих байтов. Расширить до двойного слова с учетом знака и разместить в регистре;
- уменьшить на единицу содержимое байта в сегменте данных с адресом A1 и разместить в регистре число, обратное ему по знаку

Тема 8. Логические команды и команды сдвигов

Контрольные вопросы

1. Чем отличаются логические команды от арифметических?
2. Чем отличается команда логического умножения AND от команды TEST?

3. Как с помощью логических команд можно обнулить регистр или ячейку памяти?

4. В каких случаях с помощью команд логического сдвига можно выполнять операции умножения и деления?

5. Каковы особенности выполнения операций обычного (логического), арифметического и циклического сдвига?

Упражнения:

- обнулить бит 10 в регистре BX;
- установить в единицы биты 3-2 в байте памяти ds:a1;
- инвертировать старшие три бита регистра BX;
- поменять местами содержимое полубайтов (тетрад) регистра AL;
- сформировать в регистре BX код по следующему правилу:

- разряды 15-13 - единицы;
- разряды 12-8 - нули;
- разряды 7-4 - инвертированные разряды 3-0 байта памяти ds:a1;
- разряды 3-0 - из разрядов 7-4 байта памяти ds:a1.

Тема 9. Массивы данных. Организация циклов

Контрольные вопросы

1. В чем суть различных способов программной адресации памяти: прямая, косвенная, прямая с индексированием?

2. Какие регистры можно использовать для косвенной адресации и прямой с индексированием?

3. Какой командой заносится внутрисегментной адрес памяти в регистр?

4. Как выполняется команда LOOP? Какой регистр используется ею в качестве вычитающего счетчика циклов?

5. Можно ли организовать цикл без использования команды LOOP?

Упражнение:

Определить сумму отрицательных значений в массиве однобайтных чисел. Предложить варианты реализаций с использованием косвенной адресации и прямой с индексированием.

Тема 10. Вывод данных на экран

Контрольные вопросы

1. Каковы алгоритмы преобразования числовых данных в символьные коды: десятичного, шестнадцатиричного, двоичного представления?
2. Какие программные сервисы можно использовать для вывода символьной информации на экран?

Упражнение:

Организовать вывод на экран однобайтного знакового числа в десятичном виде, hex-виде, в виде двоичного кода.

Тема 11. Ввод данных с клавиатуры

Контрольные вопросы

1. С помощью каких программных сервисов можно организовать ввод с клавиатуры?
2. Какая из функций программного прерывания Int 21h более удобна? В каких случаях?
3. Каковы входные и выходные параметры этих функций?

Упражнение:

Организовать ввод с клавиатуры десяти символов:

а) используя функцию ввода одиночного символа,

б) используя функцию ввода строки символов

2. Индивидуальные задания

Задания 1-6 предполагают разработку программы на языке ассемблера. Выполненное задание должно содержать:

- описание размещения данных и их формата;
- детальный или содержательный алгоритм работы программы;
- листинг программы - файл .lst, создаваемый транслятором;
- отладочные примеры для всех ветвей алгоритма

Рассмотрим подготовку ассемблерной программы на примере простейшей задачи нахождения суммы двух однобайтных знаковых чисел, размещенных в памяти.

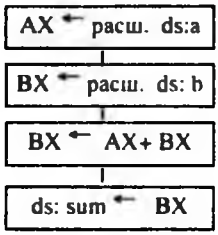
Размещение данных:

ds: a – исходное число (байт); ds: b – исходное число (байт);

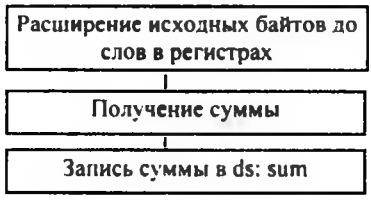
ds: sum – сумма (слово);

регистры AX, BX – для расширения исходных байтов до слов

Детальный алгоритм



Содержательный алгоритм



Листинг программы

```
Turbo Assembler    Version 2.0   05/12/05 17:54:06    Page 1   Sum.asm

1       ; Сложение двух однобайтных знаковых чисел
2                                       .386
3       0000                    dseg    segment use16
4       0000 ??                         a       db      ?
5       0001 ??                         b       db      ?
6       0002 ????                       sum     dw      ?
7       0004                    dseg    ends
8       0000                    cseg    segment use16
9                                       assume ds:dseg, cs:cseg
```

```
10      0000 B8 0000s           m1:     mov     ax, dseg
11      0003 8E D8                      mov  ds,ax
12      0005 0F BE      06 0000r        movsx    ax, ds:a
13      000A 0F BE      1E 0001r        movsx    bx, ds:b
14      000F 03 D8                      add  bx,ax
15      0011 89 1E      0002r           mov  ds:sum,bx
16      0015 B4 4C              m2:     mov  ah, 4ch
17      0017 CD 21                      int 21h
18      0019                    cseg    ends
19                                      end  m1
```

Отладочный пример

Первая часть отладочного примера содержит тестовые исходные данные в десятичном виде (если это нужно) и в hex, и ожидаемый в соответствии с алгоритмом результат.

Вторая часть – это взгляд из отладчика на занесенные исходные данные и полученный результат при исполнении программы. Отображает реальное содержимое памяти или регистров в hex.

Исходные данные		Ожидаемый результат
Ds:a = -15 (F1)	Ds:b= +127 (7F)	Ds:sum= +112 (0070)
В отладчике		
Ds:0000	Ds:0001	Ds:0002-0003
F1	7F	70 00

2.1 Размещение данных в памяти. Команды пересылки данных

Цель работы:

- уметь пользоваться директивами ассемблера для размещения данные или резервирования места для них в программе;
- понимать особенности размещения в памяти многобайтных числовых данных и символьных строк;
- научиться пользоваться командами пересылки данных

Варианты заданий

1. В сегменте данных размещены числа: -5, 128, +65531, -65535. Сумму второго и третьего чисел получить в регистре процессора. Пер-

вое число расширить до четырехбайтного формата и разместить следом за исходными данными. Третье и четвертое – поменять в памяти местами.

2. В сегменте данных размещены последовательно числа: -1, 2, +128, 128, +32769. Расширить два первых числа до формата слова и записать следом за исходными числами. Занести в регистры ES и SS произвольные числовые значения, затем сохранить их в дополнительном сегменте данных в формате двойного слова.

3. В сегменте данных размещены последовательно исходные данные: однобайтные числа 34 и 75h, +12 и –1 в формате слова, коды символов '$' и '#'. Расширить первые два числа до формата двойного слова и записать в сегмент данных следом за исходными данными. Далее записать код из первого байта кодового сегмента. Коды символов поменять в памяти местами.

4. Сегмент данных содержит следующие исходные числовые и символьные коды: -9, +128, +127, 'F', '!', '1234'. Расширить первые три числа до формата двойного слова и записать в сегмент данных следом за исходными данными. На место '!' записать символ '*', символы '12' заменить на '41'.

5. В сегменте данных записана строка символов '12345' и два числовых значения: -255 и –3. Переписать во второй сегмент данных исходную строку символов в следующем порядке: '12543' и исходные числа, преобразовав их в формат двойного слова. Получить сумму исходных числовых значений и разместить следом.

6. Сегмент данных содержит 5 байтов с одинаковым кодом 80h, числовые слова: 77h, -127, 15 и символ 'AZ' . Переписать в другой сегмент данных один из первых пяти байтов, расширенный со знаком до слова, первое числовое слово и коды символов в обратном порядке.

7. В сегменте данных размещены 5 числовых слов и символьная строка 'text'. Записать на место первого числа нули, вместо символа 't' символ '*', символы 'e' и 'x' поменять местами. Получить в регистре сумму второго и третьего числа.

8. В сегменте данных записаны 5 символов и два слова с числовыми значениями. Записать во второй сегмент данных исходную строку

символов в обратном порядке, исходные числа с обратным знаком в формате двойного слова.

9. В сегменте данных размещены числа 34 и 75h в формате байта, –1 в формате слова и символ '####'. Получить разность двух первых чисел в формате двойного слова и записать в другой сегмент данных. Последний символ '#' заменить на первое число, взятое с обратным знаком.

10. В сегменте данных размещены числа 1, 2, -3, 128, +128 и символьная строка 'abc'. Получить исходные числовые коды с обратным знаком и записать после исходных данных. Следом записать нулевое слово и исходную символьную строку в обратной последовательности.

11. В сегменте данных размещены три знаковых однобайтных числа. Расширить их до двойных слов и записать в другой сегмент данных. Следом записать сумму первых двух чисел в формате слова и их разность в формате двойного слова.

12. В сегменте данных размещена строка из четырех символов и три числа: однобайтное, двухбайтное и четырехбайтное. Преобразовать два первых числа в формат двойного слова и записать следом за исходными данными. Последние два символа в символьной строке поменять местами.

13. Сегмент данных содержит три числовых однобайтных кода и коды символов 'F123'. Преобразовать первое число в формат слова, второе – в формат двойного слова и записать следом за исходными данными. На место 'F' записать числовой ноль, символы '123' заменить на '321'.

14. В сегменте данных записана строка символов "hello" и два слова с числовыми значениями -5 и 80h. Записать во второй сегмент данных исходную строку символов в обратном порядке без первого символа и исходные числа с обратным знаком в формате слова.

15. В сегменте данных размещены последовательно числа: два – в формате байта, одно - в формате слова. Получить сумму первых чисел в формате слова и записать следом за исходными данными. Получить из третьего число, обратное по знаку и записать в другой сегмент данных.

16. Сегмент данных содержит 5 байтов с одинаковым кодом, число со знаком в формате слова и три символа. Переписать в другой сегмент данных один байт из одинаковых кодов, старший байт от числового слова и последний символ из символьной строки. Получить сумму двух первых байтов в формате слова и разместить в регистре.

17. В сегменте данных размещены 4 однобайтных знаковых числа и символьная строка 'abc'. Записать следом за ними исходные числовые коды с обратным знаком, расширенные до слова, нулевой байт и исходную символьную строку в обратной последовательности.

18. В сегменте данных размещены два числа в формате двойного слова, одно число в формате байта и символ 'X'. Записать следом за исходными данными старшие байты от первых двух чисел и символ с кодом на 1 меньше, чем 'X'.

19. В сегменте данных размещены три числа в формате слова и строка из трех символов. Взять из исходных чисел младшие байты и записать следом за исходными данными. Далее записать символы исходной строки в обратном порядке.

20. Сегмент данных содержит три числа в формате слова и два символьных кода. Преобразовать числа в формат двойного слова и записать в другой сегмент данных. На место первого символа записать код FF, на место последнего символа – старший байт первого исходного числа.

21. В сегменте данных размещены однобайтные числа: 1, -8, -128, +127. Расширить два первых числа до формата слова, получить их сумму и записать следом за исходными числами. Последние два числа расширить до формата двойного слова и записать в другой сегмент данных. Вместо них в исходный сегмент данных записать символы '12'.

22. В сегменте данных размещены два числа в формате байта, одно в формате слова и 4 символа. Расширить первые два числа до формата слова и записать в другой сегмент данных. Третье число заменить на обратное по знаку. Первый и третий символы поменять местами.

23. В сегменте данных размещены три двухбайтных кода. Расширить их до слов и записать в другой сегмент данных. Следом записать

старшие байты исходных кодов, сумму первых двух кодов в формате слова и символ '!'.

24. В сегменте данных размещены три знаковых однобайтных числа и символьная строка 'txt' . Расширить числа до двойных слов и записать в другой сегмент данных. Следом записать сумму второго и третьего числа, первое число с обратным знаком и два последних символа.

25. В сегменте данных размещены четыре однобайтных числа. Расширить два первых числа до двойных слов и записать следом за исходными числами. Получить сумму третьего и четвертого числовых значений в формате двойного слова и записать ее в другой сегмент данных. Следом записать строку символов 'text'.

26. Сегмент данных содержит три двухбайтных числовых значения и два символьных кода. Преобразовать числа в формат двойного слова и записать в другой сегмент данных. На место первого символа записать старший байт первого числового значения, на место второго – его младший. Получить сумму исходных числовых значений в формате двойного слова в регистре.

2.2 Ветвления в программах. Арифметические и логические команды. Исполняемый модуль типа .com

Цель работы:

- освоить команды передачи управления, арифметические и логические команды;
- создать исполняемый программный модуль типа .com

Варианты заданий

1. Проанализировать содержимое регистра BL:
если BL = 0 , в BL занести содержимое первого байта кодового сегмента, циклически сдвинутое влево на 3 бита;
если BL < 0 , то BL инвертировать;
если 10 <= BL < 100 , инвертировать 2 младших разряда BL
если BL => 100 , обменять содержимое полубайтов регистра BL

2. Сформировать содержимое регистра CL следующим образом:
биты 7-5 – взять из трех младших разрядов регистра DL;
биты 4-2 – из разрядов 6-4 регистра BH;
биты 1-0 - нулевые.

Если сформированный код находится в диапазоне от 5 до 20, то инвертировать его.

3. Сформировать код в регистре EBX следующим образом:
4 старших бита – из разрядов 6-3 байта данных
4 младших бита – единицы;
остальные - нулевые

Если полученное значение находится в интервале от 150 до 2000 , инвертировать содержимое регистра BX.

4. Сформировать содержимое регистра CH следующим образом:
CH ← 2 , если сумма двух байтов исходных данных меньше 127
CH ← 2 * BX, если два младших бита регистра BH нулевые
CH ← разность двух байтов исходных данных - в остальных случаях

5. Определить делимость на 3 знакового двухбайтного числа, размещенного в кодовом сегменте, используя вычитание. Если оно кратно 3, то в регистре DL инвертировать значение старшей тетрады, в противном случае - младшей.

6. Сформировать содержимое регистра CX следующим образом:
CX ← M , если N < M
CX ← 8 * N , если N = M
CX ← N - M , если N > M где N – код из старшей тетрады регистра BH, M - код из младшей тетрады регистра BH.

7. Вычислить произведение четырех беззнаковых однобайтных кодов. Если полученный результат находится в интервале от 254 до 65534, то полученном 4-байтном произведении инвертировать разряды 31-27 , в противном случае – поменять местами тетрады в каждом его байте.

8. Определить наибольшее из трех двухбайтных знаковых чисел, размещенных в памяти. Найденное слово записать в регистр BL, затем установить в нем разряды 15-12 -в единицы , разряды 11-7 оставить без изменения, разряды 6-4 установить в нули, разряды 3-0 инвертировать.

9. Проанализировать код в слове данных, размещенном в кодовом сегменте. Если эта величина больше 18, установить в единицы разряды 7,5 и 2 младшего байта. Если равен 18 – инвертировать старший байт данных. В противном случае - увеличить эту величину в 4 раза. Результат получить в формате двойного слова в регистре EDX.

10. Проверить численное соотношение: X + Y <= Z + N, где X и Y - байты данных в кодовом сегменте, Z - содержимое регистра BH, N - содержимое регистра BL. Если соотношение выполняется, сдвинуть циклически на 2 разряда код в байте X и инвертировать биты 6-5 в байте Y. В противном случае, поменять местами младшие тетрады в BH и BL и установить в единицы разряды 2-0 в байте X.

11. Определить двухбайтную сумму трех байтов данных. Если сумма меньше 12, то поменять местами тетрады в старшем байте результата. Если сумма равна 30, сбросить в ноль разряды 3-2 результата. В остальных случаях сдвинуть арифметически значение результата на 7 бит влево.

12. Сложить содержимое трех байтов данных. Результат получить в формате двухбайтного числа. Если значение результата находится в диапазоне от 5 до 300 , записать в память младший байт результата с установленным в 0 младшим разрядом. В противном случае записать старший байт с инвертированными разрядами 7 и 4.

13. Сформировать код в регистре DX следующим образом:
DX = X + Y, если X < Y и X < 30
DX = X - Y, если X > Y
DX = 0, в остальных случаях
где X и Y - байты знаковых числовых данных в кодовом сегменте. Если полученный в DX код - отрицательное число, преобразовать его в положительное, сдвинуть циклически на 3 разряда влево и инвертировать биты 14-11.

14. Определить, содержит ли байт данных в кодовом сегменте в разрядах 6-2 двоичную комбинацию 11010. Если да, то сформировать код в регистре BH следующим образом:
BH← сумма тетрад исходного байта, если значение старшей тетрады больше младшей

BH← разность тетрад в противном случае.
Если полученный в BH код меньше 40, инвертировать его.

15. В кодовом сегменте размещены три однобайтных беззнаковых кода. Определить остаток от деления максимального значения на минимальное:
а) используя команду деления, если два старших бита делимого - единицы и два младших - нули.
б) используя команду вычитания – в противном случае

16. Сформировать код в регистре CX следующим образом:
CX ← 2 * X, если 10 <= X <= 20
CX ← X - 3, если X < 10
CX ← X^2 , если X > 20 и 2 старших бита X - нули
где X – однобайтный код в кодовом сегменте.

17. Сложить содержимое трех байтов данных из кодового сегмента. Результат в формате слова разместить в CX. Если при сложении перенос произошел один раз, то в кодовый сегмент записать младший байт результата с инвертированным младшим разрядом. Если перенос происходил дважды, то - младший байт результата с инвертированным старшим разрядом.

18. Определить наименьшее из трех знаковых двухбайтных кодов, размещенных в кодовом сегменте. Если это отрицательная величина, в полученном коде поменять местами значения старшей и младшей тетрад. В противном случае, инвертировать разряды 12, 2 и 0.

19. Занести содержимое 4-2 разрядов байта данных из кодового сегмента в разряды 7-5 другого байта данных, не искажая значений его остальных битов. Если полученный во втором байте код находится в числовом интервале от -10 до 10 (включительно), инвертировать полученное значение, иначе – установить в единицы биты 7-6 и в ноль биты 4-3 этого кода.

20. Определить, принадлежит ли двухбайтное число, размещенное в кодовом сегменте, числовому интервалу от -25 до +25 включительно. Если - да, то в регистр CH занести младший байт этого числа с инвертированными тремя младшими разрядами, иначе - старший байт с установленными в единицы тремя старшими разрядами.

21. Проверить соотношение: A + 2 < B * C, где
A - содержимое байта в кодовом сегменте
B - содержимое регистра BL с установленным в ноль старшим разрядом
C - содержимое регистра CH с инвертированными 6-4 разрядами
Если соотношение выполняется, в регистр BH записать код FFh, в противном случае - 55h.

22. Сформировать содержимое регистра CX следующим образом:
-разряды 15-12 - из 4-х младших разрядов регистра BH;
-разряды 11-8 и 1-0 - единичные;
-разряды 7-2 - из разрядов 6-1 регистра BL
Если полученное значение принадлежит интервалу от +2 до + 255, то в старший байт CX занести значение младшего байта.

23. Вычислить величину 4 * N + 3 * M, где
N - циклически сдвинутое влево на 2 бита содержимое регистра DH
M - содержимое первого байта кодового сегмента, преобразованное следующим образом: биты 6-5 и 3-2 поменять местами.
Результат в формате двухбайтного числа занести в DX.
Если полученный код в DX < 0, получить его положительный эквивалент. Если 1< DX < 275, то инвертировать биты 4-3 регистра DH.

24. Сложить содержимое регистра BX и слова данных из кодового сегмента. Если при сложении произошел вспомогательный перенос, значения в битах 6-4 результата заменить на значения битов 3-1. Если при сложении произошел перенос за пределы разрядной сетки, то биты 6-4 результата инвертировать.

25. В кодовом сегменте размещены два беззнаковых числовых байта. Определить остаток от деления большего на меньшее. Если значение остатка меньше 3, частное инвертировать. В противном случае, установить в байте частного биты 3-2 в единицы, а биты 6-5 в нули.

2.3 Массивы данных. Организация циклов

Цель работы:

- освоить методы косвенной адресации данных в памяти;
- уметь реализовать программные циклы

Варианты заданий

1. В области кодового сегмента содержится массив двухбайтных числовых данных. Величины, меньшие числа 50, переписать в обратном порядке в другую область кодового сегмента в однобайтном формате.

2. В байте данных кодового сегмента установить младшие 3 бита в единицы, разряды 6-5 в нули, остальные оставить без изменения. Полученным кодом, расширенным с учетом знака до двухбайтного формата, заполнить 20-ти байтную область в сегменте данных.

3. Каждый байт области в сегменте данных преобразовать следующим образом: если значение байта находится в интервале 50h -5fh включительно, установить старшие 4 бита в единицы, иначе - установить младшие 4 бита в нули. Получившиеся отрицательные числа занести в другую область сегмента данных в 2-х байтном формате.

4. Если количество нулевых байтов в области сегмента данных больше десяти, инвертировать эти байты. В противном случае, установить 3 младших бита каждого байта этой области в нули. Преобразовать данные к формату двухбайтных чисел с учетом знака и записать в другую область памяти.

5. Область кодового сегмента содержит двухбайтные знаковые числа. Определить наименьшее из отрицательных чисел. Если это значение больше -30, то записать его в память в формате двойного слова.

6. Область сегмента данных содержит однобайтные числа. Определить, сколько байтов содержат значения из числового диапазона от –30 до +30 (включая границы).

7. Если содержимое байтов двух областей в сегменте данных соответственно равны между собой, то в первой области коды инвертировать , а во второй – установить младшие 3 бита каждого байта в единицы.

8. Область в сегменте данных содержит массив однобайтных знаковых чисел. Занести эти данные в другую область сегмента данных в формате слов в следующей последовательности: сначала отрицательные числа, затем - положительные.

9. В области кодового сегмента размещен массив двухбайтных данных. Данные, значения которых находятся в интервале от -20 до +20, заменить на противоположные по знаку и в однобайтном формате записать в другую область.

10. В сегменте данных размещен массив числовых слов. Определить наибольшее из отрицательных чисел и записать его в кодовой сегмент в формате двойного слова.

11. Если содержимое трех байтов сегмента данных одинаковое, то этим кодом, расширенным до двухбайтного формата, заполнить массив в сегменте данных. В противном случае, эту область заполнить кодом из первого байта кодового сегмента.

12. В байте сегмента данных поменять местами содержимое битов 6-5 и 1-0. Расширить полученный код до двухбайтного формата с учетом знака и заполнить им область памяти в дополнительном сегменте.

13. Сегмент данных содержит массив числовых байтов. Определить сумму байтов, числовые значения которых находятся в диапазоне от -10 до 10. Результат в формате двойного слова разместить в сегменте данных.

14. Область сегмента данных содержит массив двухбайтных чисел. Определить сумму отрицательных чисел, расположенных до первого положительного, в формате двойного слова.

15. Определить разность двух байтов из сегмента данных. Если она отрицательна, то некоторую область сегмента данных заполнить кодом разности в формате слова, иначе - нулями.

16. Если содержимое байта в сегменте данных больше содержимого регистра BH, то область сегмента данных заполнить кодом разности этих байтов в двухбайтном формате, иначе - двухбайтной суммой.

17. Числовой массив содержит двухбайтные данные. Данные, числовое значение которых находится в диапазоне от 0 до 255 (включительно), записать в другой сегмент данных в однобайтном формате.

18. Сравнить содержимое двух байтов памяти. Если содержимое первого байта больше второго, то область в кодовом сегмента заполнить значением двухбайтной суммы этих байтов. В противном случае – значением их разности

19. Область памяти содержит массив двухбайтных знаковых чисел. Заменить данные, числовые значения которых находятся в интервале от –32 до +32, на обратные по знаку. Получившиеся после этого преобразования положительные числа переписать в другую область памяти.

20. В сегменте данных размещен массив двухбайтных чисел. Переписать в область кодового сегмента те из них, которые больше своих "соседей" в массиве, в формат двойного слова.

21. Найти наибольшее в массиве двухбайтных знаковых чисел. Если эта величина меньше +25, заполнить ею область сегмента данных, иначе - записать эту величину в кодовый сегмент.

22. Область сегмента данных содержит однобайтные знаковые числа. Переписать их в другой массив в двухбайтном формате в следующей последовательности: сначала положительные, затем отрицательные.

23. Определить частное от деления двух байтов из сегмента данных. Если остаток от деления меньше 3, одну область сегмента данных заполнить однобайтным значением частного, другую – значением остатка. Если целочисленное деление невозможно – двухбайтной разностью этих чисел.

24. Найти наибольшее значение в массиве двухбайтных чисел. Заменить все отрицательные на полученное значение.

25. Массив в сегменте данных содержит однобайтные коды. Квадраты этих значений записать в другую область памяти в обратной последовательности.

2.4 Преобразование формы представления данных. Ввод данных с клавиатуры.

Цель работы:

- освоить алгоритмы преобразования символьных данных в числовые;
- освоить функции системного сервиса прерывания Int 21h для организации ввода данных с клавиатуры.

Варианты заданий

1. Введенное с клавиатуры в десятичном виде двухбайтное знаковое число разместить в сегменте данных в числовом виде.

2. С клавиатуры вводится в десятичном виде беззнаковое число в диапазоне от 0 до 255. Получить его упакованный BCD-код.

3. С клавиатуры вводится число в hex-виде (8 символов). Получить его числовое представление.

4. С клавиатуры вводится число в hex-виде (4 символа). Получить его упакованный BCD-код.

5. С клавиатуры вводится число в символьном двоичном виде (8 символов). Получить его числовой код.

6. С клавиатуры вводится беззнаковое число в hex-виде (4 символа). Получить его числовое значение в распакованном BCD-коде.

7. С клавиатуры вводится код знакового числа в двоичном виде (16 символов). Получить модуль его числа.

8. С клавиатуры вводится беззнаковое число в hex- виде (4 символа). Определить сумму тетрад его числового кода.

9. С клавиатуры вводится десятичное знаковое число, модуль которого не превышает 127. Получить его числовой эквивалент.

10. С клавиатуры вводится число в hex-виде (4 символа). Получить числовой код с обратным знаком.

11. С клавиатуры вводится десятичное беззнаковое число в диапазоне от 0 до 999. Получить сумму цифр, его составляющих.

12. С клавиатуры вводится два десятичных беззнаковых числа, каждое в пределах до 255. Определить их числовую сумму.

13. С клавиатуры вводится два беззнаковых числа: одно в десятичном виде (не более 255), другое – в двоичном виде (8 символов). Получить сумму этих чисел .

14. Заполнить массив в сегменте данных числовыми кодами, вводимыми с клавиатуры в шестнадцатиричном виде (каждый код 8-ми символьный).

15. Одно знаковое число вводится с клавиатуры в десятичном виде. Другое - в шестнадцатиричном виде (4 символа). Определить, одинаковые ли это числа.

16. Определить сумму цифр шестнадцатиричного кода, введенного с клавиатуры (8 символов).

17. Заполнить массив в сегменте данных числовыми кодами, вводимыми с клавиатуры в десятичном виде (каждое число в диапазоне от –128 до +127).

18. Ввести с клавиатуры текстовую строку. Если она содержит символы десятичных цифр (0 – 9), определить сумму этих цифр.

19. Два беззнаковых числа вводятся с клавиатуры в шестнадцатиричном виде - по 4 символа каждое. Определить их числовую сумму.

20. Символьная строка вводится с клавиатуры. Переписать в отдельную область памяти символы десятичных цифр, в другую область - символы букв. Преобразовать символы десятичных цифр в числа.

21. Одно знаковое число вводится с клавиатуры в десятичном виде (в диапазоне от –128 до +127), другое – в двоичном виде. Определить, какое из них больше.

22. С клавиатуры вводится знаковое десятичное число. Получить его упакованный BCD-код.

23. С клавиатуры вводится число в шестнадцатиричном виде (4 символа). Получить распакованный BCD-код числа.

24. С клавиатуры вводится последовательность десятичных чисел, каждое не более двух символов. Создать массив из их числовых значений.

25. Одно знаковое число вводится с клавиатуры в десятичном виде (5 байтов). Другое – в шестнадцатиричном виде (4 байта). Определить, равны ли они по модулю.

26. Число вводится с клавиатуры в двоичном виде (16 байтов). Определить его числовое значение и количество единичных разрядов числа.

2.5 Вывод данных на экран. Использование процедур

Цель работы:

- освоить алгоритмы преобразования числовых данных в символьные;
- освоить функции системного сервиса для организации вывода данных на экран;
- использовать в программе процедуры

Варианты заданий

1. Показать на экране в десятичном виде по одному в строке содержимое 10-ти знаковых числовых байтов из сегмента данных.

2. Показать на экране содержимое 10-ти числовых байтов из сегмента данных в шестнадцатиричном виде, разделяя коды символом пробела

3. Показать на экране содержимое 10-ти знаковых числовых байтов в двоичном виде по 2 числа, разделенных пробелами, в строке.

4. Показать содержимое первых 15 байтов кодового сегмента в шестнадцатиричных кодах, разделяя байты пробелом.

5. Вывести на экран в десятичном виде содержимое массива однобайтных знаковых упакованных ВСД-кодов.

6. Вывести на экран в шестнадцатиричном виде через запятую двухбайтные беззнаковые числа (10 слов) из сегмента данных.

7. Вывести на экран (по одному в строке) в двоичном виде двухбайтные коды из массива данных в памяти.

8. Вывести на экран в десятичном виде массив трехбайтных знаковых распакованных ВСД-чисел.

9. Показать на экране массив двухбайтных знаковых чисел в десятичном виде.

10. Определить сумму элементов массива однобайтных знаковых чисел и показать ее на экране в hex-виде.

11. Вывести на экран десять однобайтных беззнаковых кодов из массива в сегменте данных в десятичном виде, каждое - в отдельной строке.

12. Вывести на экран сумму элементов двухбайтного массива данных в десятичном виде.

13. Вывести на экран сумму содержимого регистров BX и CX в двоичном виде.

14. Показать на экране все отрицательные числа массива однобайтных чисел в hex-виде по одному в строке.

15. Показать на экране в десятичном виде сумму отрицательных элементов массива двухбайтных чисел

16. Показать на экране в обратной последовательности в десятичном виде массив двухбайтных чисел, размещенный в сегменте данных.

17. Определить количество нулевых байтов в массиве однобайтных кодов и показать на экране. В следующей строке экрана показать внутрисегментные адреса этих байтов в hex-виде.

18. Определить количество положительных чисел в массиве двухбайтных кодов и показать его на экране в десятичном виде.

19. Показать на экране содержимое массива однобайтных кодов в hex- виде, разделяя их пробелами. Показать внутрисегментные адреса первого и последнего байта массива.

20. Показать на экране текущее значение всех сегментных регистров в hex- виде. Дать сообщение о количестве используемых в программе сегментов (сегментные регистры будут иметь разные значения).

21. В массиве двухбайтных знаковых чисел найти отрицательные значения, переписать их в другую область и показать на экране в hex-виде через пробел.

22. В массиве двухбайтных кодов заменить нулевые значения на значение максимального элемента. Исходный и полученный массивы показать в отдельных строках экрана в hex-виде, разделяя коды символом подчеркивания.

23. В массиве однобайтных чисел заменить все значения на обратные по знаку. Исходный и преобразованный массивы показать на экране в десятичном виде с соответствующими заголовками.

24. Массив однобайтных кодов показать на экране в обратной последовательности в двоичном виде, по одному коду в строке.

25. Из массива двухбайтных чисел выбрать наименьшее из положительных и показать его на экране в десятичном и в hex-виде с соответствующими сообщениями.

2.6 Работа с файлами

Цель работы: освоить функции системного сервиса для работы с файлами данных.

При работе с исходным файлом предусмотреть приглашение для ввода пути к нему с клавиатуры. Обязательно выполнять проверку существования файла с соответствующим сообщением .

Варианты заданий

1. Прочитать из файла с произвольным именем строки, начинающиеся с символа А, вывести на экран и записать в новый файл.

2. Прочитать из файла с произвольным именем последнюю строку, вывести на экран, начиная с 5-й строки экрана, и добавить в конец другого файла.

3. Ввести с клавиатуры строку символов произвольной длины и записать ее, как вторую строку, в файл с произвольным именем.

4. Переписать из произвольного файла в новый файл все строки, заканчивающиеся символом "!"

5. В произвольном файле определить количество "пустых" строк (содержащих только управляющие коды Возврат каретки и Перевод строки). Результат вывести на экран с текстовым заголовком.

6. Определить количество символов в последней строке произвольного текстового файла. Результат и саму строку показать на экране. Записать строку в новый файл.

7. Найти в произвольном текстовом файле самую длинную строку. Записать ее в новый файл и показать на экране.

8. Из произвольно заданного файла считать первую и последнюю строки, объединить в одну , разбить пополам и записать две строки в новый файл. Все строки показать на экране.

9. Из произвольного файла выбрать все цифровые символы и записать последовательно в новый файл, вставляя управляющие коды Возврат каретки и Перевод строки после каждых 10 символов.

10. В произвольном файле поменять местами первую и последнюю строки. Найденные строки показать на экране.

11. Из произвольного файла считать последнюю строку и записать в обратной последовательности в новый файл. Строку показать на экране.

12. Из произвольного текстового файла прочитать 20 последовательных символов, начиная с 10-го байта в файле. Показать строку на экране и записать в новый файл по 4 символа в строке.

13. Из произвольного файла прочитать первые 5 символов из каждых последовательных 20 символов и записать, как отдельные строки, в новый файл.

14. Дополнить каждую строку произвольного файла символами "***"и записать в новый файл.

15. Из каждой строки произвольного файла удалить первые 3 символа и сохранить изменения в новом файле.

16. В начале каждой строки произвольного текстового файла добавить символы, введенные с клавиатуры. Преобразованные строки сохранить в новом файле.

17. Создать новый файл, записать в него строки , вводимые с клавиатуры. Ограничить длину вводимых строк 80-ю символами.

18. Считать из произвольного файла последние 6 символов из каждых 20 последовательных символов. Записать в новый файл как отдельные строки.

19. В произвольном файле определить самую короткую строку. Записать ее в новый файл и показать на экране.

20. В произвольном файле определить количество строк, содержащих M символов. Значение M вводится с клавиатуры. Найденные строки показать на экране.

21. Из каждой строки произвольного текстового файла удалить все символы "!" . Сохранить измененные строки в новом файле.

22. В четвертой строке произвольного файла заменить заглавные латинские символы на прописные. Строку показать на экране. Изменения сохранить в новом файле.

23. Создать новый файл ,записав в него из произвольного текстового файла строки длиной менее 4-х символов.

24. Во второй строке произвольного текстового файла изменить последовательность символов на обратную. Исходную и измененную строку показать на экране.

25. Заменить первую строку символов в произвольном файле на такое же количество символов, введенных с клавиатуры. Количество символов для ввода запросить после определения длины строки. Исходную и измененную строки показать на экране.

2.7 Трансляция символических команд в машинный код

Цель работы: транслировать вручную заданные символические команды (базовая система команд процессоров Intel) в машинный код. Дать письменные пояснения.

Пример выполнения задания:

Команда	Выполнение команды
ADD AX,[SI+25]	AX<-- AX + [DS:SI+25]

Формат команды

cop dw	mod reg r/m	disp

Машинный код: 00000011 01000100 00011001 (034419h)

cop=00000 - код операции

d=1 - первым операндом является регистр

w=1 - операция выполняется над словами

mod=01 - в команде есть адресация памяти и длина поля disp - байт.

reg=000 - код регистра AX

r/m=100 - код способа адресации памяти [SI+disp]

disp=00011001 - величина в поле disp (- 25=00011001b)

Варианты заданий

1. ADD AH, 80h
   OR [DI-19], DI
   MOV BP, [BP+SI+0EC28h]

```
    JMP ADR          ; адрес команды JMP- 0100h,  ADR –  0200h
    TEST AX,BX
    PUSH CS
    ADC AH, ES:[BX+4]
    MOV word ptr[SI], 1

2.  OR [BX+DI+19h], DI
    MOV DX, CS:[BX+25]
    ADD AH, BL
    MOV AL, [BP+2]
    PUSH DS
    SUB word ptrDS:[30h], 4
    JMP short N1     ; адрес команды JMP- 0020h,  N1 -0038h
    RCL byte ptr[DI-1],1

3.  LOOP M1          ;адрес команды LOOP - 0100h,  M1 -00F1h
    CMP byte ptr[SI],1
    JC M2            ;адрес команды JC- 0020h, M2 - 00F1h
    MOV BX, DS:[1]
    MOV DS,BX
    PUSH ES
    SHR word ptr[BP-2],CL
    DEC DX

4.  MOV SI, 81
    CALL  M3     ; адрес команды CALL - 0070h,  M3 –  0100h
    JB M1        : адрес команды JB - 1F00h,  M1 - 1EF0h
    ADD byte ptr DS:[2Fh], 23
    MOV ES:[BP+14h], DX
    OR BL,AL
    XOR CH, [SI+1]
    INC BP

5.  MOV ES, BX
    ADD BX, [BP-2]
    LOOP M2      ; адрес команды LOOP – 020Dh,  M2 –0200h
    CMP AL, 2Eh
    DEC BP
    MOV ES:[SI+14h], DX
    AND word ptr[BX+SI], 5
    ADD [BX+1],AX

6.  LOOP  T1     ; адрес команды LOOP- 0070h,  T1 - 0050h
    ADC BX,CX
    SUB byte ptr[BP-17], 8
```

```
MOV ES:[68h]. DX
JMP short M5     ; адрес команды JMP - 0100h.   M5 – 0180h
POP AX
CALL M4  : адрес команды CALL – 0100h.  M4 - 0200h
RCR AL.CL
```

7.
```
MOV ES:[16h]. AH
ADC DX. 0B9h
NEG byte ptr[SI-5]
POP DS
SUB CX.AX
JNE M1      :  адрес команды JNE - 0200h.  M1 - 0220h
JMP word ptr[DI-5]
RET     ; внутрисегментный
```

8.
```
MOV SI. DS:[64h]
SAL SI,CL
LEA DI.[SI+0B5h]
MOV word ptr ES:[BP-14h]. 3
INC DX
CMP SI. DX
MOV SI.10
JNZ M4      : адрес команды JNZ - 0020h.  M4 - 003Fh
```

9.
```
OR AL,CL
SHL BX. 1
JMP M5      : адрес команды JMP - 02F0h.  M5 - 0100h
JMP dword ptr[BX-3]
NOT byte ptrSS:[2]
ADD ES:[DI],BP
MOV word ptr[BX+0B7h]. 5
MOV AL,27
```

10.
```
PUSH BP
XOR CH. DS:[2E34h]
SUB AX,CX
ADD CS:[DI+42h]. CH
DEC SI
AND word ptr[BX+SI+13h]. 17
JMP short M1    :адрес команды JMP - 0020h.  M1 - 007Fh
TEST AX. 2
```

11.
```
JMP word ptr[BP-4]
AND DL, [BX+SI]
```

```
ADD CS:[BX+SI+6], DX
MOV AX, ES
MOV word ptrDS:[4], 16
JNC short M1        ; адрес команды JNC - 10F0h,  M1 - 1040h
PUSH SI
NOT byte ptr[SI-6]
```

12.
```
SHR AX, CL
OR SI, -16
AND AL, DS:[0FEh]
CMP word ptr SS:[SI-4], 0B2h
TEST AL, 5
PUSH SI
JNZ M2              : адрес команды JNZ - 0100h,   M2 - 01f0h
MOV DS, AX
```

13.
```
PUSH DI
CMP DI, -1
ADD ES:[DI-1], BX
SUB DI, BX
SHR byte ptr[BP], 1
MOV word ptr DS:[2F00h], 2
LOOP M3             ; адрес команды LOOP - 0100h,  M3 - 0050h
JMP short M1        : адрес команды JMP - 1000h,  M1 -10F0h
```

14.
```
ADD BX, DS:[0Ah]
PUSH [SI-5]
MOV BP,CS:[BX+DI+8]
CMP AX,1000h
SHL word ptr[SI],CL
JNE M1              : адрес команды JNE - 0300h.  M1 -03F0h
SUB CX, 5
MOV CX,DI
```

15.
```
SUB AX, AX
PUSH [BP-5]
LEA DX, DS:[2D00h]
MOV DX, 2F0h
SHR AX, CL
OR SI, -16
AND DX, ES:[BX+SI-4]
LOOP M1       : адрес команды LOOP - 0200h.  M1 - 0180h
```

16.
```
TEST AX, 5
MOV CX,10Fh
MOV DX, CS:[184Fh]
```

```
        SBB word ptr[SI-7], 17
        XOR DX, DX
        ADD AX, [BX+SI]
        INT 21h
        JMP M4      ; адрес команды JMP - 0030h, значение M4 - 0016h

17.     MOV BX,2f00h
        LEA DX, [BX+SI]
        MOV [SI+0Ah], CL
        SUB AL, ES:[2Fh]
        JP  M1          ; адрес команды JP - 0200h.значение M1 -01F0h
        CMP CL, '!'
        ROR word ptr[SI],1
        CMP BX,CX

18.     MOV AH, 3Ch
        XOR CX,CX
        TEST DI, [BX+SI]
        MOV BX,CS:[BP-5]
        SHR AX, CL
        OR word ptr[SI+5], -7
        JMP short M1  : адрес команды JMP - 0200h. M1 - 020Fh
        SUB DX, 4

19.     DEC SI
        AND [BX+SI+13h], DH
        JNC M1      ; адрес команды JNC - 0200h.   M1 - 01D0h
        PUSH DI
        AND byte ptr ES:[BX+SI], 25
        NOT word ptr[BX+20h]
        MOV AX, ES
        MOV DS:[400h], AX

20.     OR BL, AL
        SHL word ptr[BX], 1
        ADD word ptr[SI+BX+1], 4
        INC byte ptr[BP]
        PUSH DS
        MOV CS:[2F0h], AL
        JC M2       : адрес команды JC - 0300h, значение M2 - 02F0h
        CMP AL, 5Ch

21.     PUSH [BP+1]
        LOOP M2     : адрес команды LOOP - 0100h,  M2 - 00E0h
        XOR CH, [DS:2E34h]
```

```
        INC BP
        ADD SS:[SI-4], CX
        CMP BX, CX
        CMP AX, 0B2h
        SHR DX, CL
```

22.
```
        ROL AX, 1
        OR SI, -8
        AND byte ptr[SI], 0FEh
        CMP CS:[300h], BX
        TEST AX, 5
        CALL M4      ; адрес команды CALL - 0200h,  M4 - 0250h
        JMP short M4 ; адрес команды JMP - 0200h,  M4 - 0230h
        MOV DX, 184Fh
```

23.
```
        PUSH [DI+30h]
        CMP DI, -1
        ADD ES:[DI], BX
        MOV DS:[60h], AX
        RCR DX, CL
        LOOP M5     ; адрес команды LOOP - 150h,  M5 - 100h
        AND [BX+DI+7], CH
        MOV BX, DS
```

24.
```
        PUSH BP
        CALL M1      ; адрес команды CALL - 1000h,  M1 – 10A0h
        XOR CH, ES:[2E3h]
        SUB BP, BX
        ADD word ptr[SI+49], -25
        JMP DX
        MOV DX, 18h
        INT 20h
```

25.
```
        ADD AH, 8
        OR  SS:[BX+DI+20], DI
        LEA DX, [BX+SI]
        MOV SP, [DS:28h]
        JMP dword ptr[SI]
        POP DS
        ADC byte ptr[BP+46], 8
        SUB AX,DX
```

3. Система команд процессора 80386 в реальном режиме

Команды пересылки данных позволяют пересылать данные между регистрами, памятью, портами ввода/вывода. В эту группу входят также команды преобразования форматов, операции со стеком.

Команды двоичной арифметики выполняют арифметические действия со знаковыми и беззнаковыми целыми числами в форматах байта, слова и двойного слова.

Команды десятичной коррекции дополняют команды двоичной арифметике. Они позволяют оперировать с распакованными или упакованными двоично-десятичными числами. После использования обычных арифметических операций над этими данными требуются применение команд десятичной коррекции.

Команды логических операций выполняют операции булевой алгебры над байтами, словами или двойными словами.

Сдвиги выполняются над регистром или операндом в памяти. Число бит, на которое производится сдвиг, берется из непосредственного операнда или регистра *CL*. Выталкиваемые биты попадают во флаг CF.

При сдвигах влево и простом сдвиге вправо освобождающиеся биты заполняются нулями. При арифметическом сдвиге вправо освобождающиеся разряды заполняются значением знакового бита. При циклических сдвигах выталкиваемые биты попадают и во флаг CF, и в освобождающиеся позиции. В сдвигах могут участвовать и два операнда (в командах *SHLD* и *SHRD).*

Команды битовых операций позволяют копировать бит в CF, устанавливать значение указанного бита, искать установленный бит. Битовые операции выполняются над 16- или 32-битным словом памяти или регистром. Команды *BSF*, *BSR* и *BT* не изменяют значения слова; *BTC*, *BTR* и *BTS* изменяют указанный бит слова в регистре или памяти.

Команды передачи управления включают в себя: безусловные и условные переходы, вызовы процедур и команды прерываний.

Безусловный переход (JMP) может быть внутрисегментным (ближний или коротким) и межсегментным (дальним). Адрес перехода может указываться прямо в команде, или при косвенной адресации находится в регистре или памяти. *Короткий* переход (short) передает управление на адрес, удаленный от текущего в пределах -128...+127 байт, *ближний* (near) — в пределах сегмента. При

дальнем (far) переходе адрес перехода (прямой или косвенный) задает новые значения для CS и IP.

Условные переходы в командах процессоров 8086 и 80286 возможны только короткие (8-разрядное смещение), в процессорах, начиная с 386+, допускается переход в пределах 16-разрядного смещения. Условные переходы выполняются по состоянию флагов и/или содержимому регистра *CX*.

Команды циклов LOOP соединяют в себе уменьшение на 1 регистра *CX* и условный переход.

Команда вызова процедуры CALL (внутрисегментный или межсегментный) передает управление по адресу, сохраняя в стеке адрес возврата, то есть адрес следующей за CALL команды. При внутрисегментном вызове сохраняется IP, при межсегментном - CS: IP. *По команде возврата RET* сохраненный адрес восстановится.

Строковые команды выполняются с операндами в памяти, адресуемыми регистрами DS:SI для источника и ES:DI для приемника. Операции могут использоваться с префиксами условного или безусловного повтора. После каждой пересылки или сравнения индексные регистры (*SI, DI* или оба) участвующих операндов автоматически инкрементируются или декрементируются на количество байт, участвующих в операции (1,2 или 4). Направление модификации определяется флагом *DF*. *DF=0* — инкремент адреса, *DF=1* — декремент.

Операции с флагами изменяют значения отдельных флагов, сохраняют их значение в стеке или восстанавливают из стека.

Команды загрузки регистров-указателей позволяют загружать адреса памяти в регистры-указатели адреса.

В описании системы команд приняты следующие обозначения:

- *reg* — регистр;
- *mem* — ячейка памяти;
- *i* — непосредственный операнд;
- *r/m* — регистр или ячейка памяти;
- *r/m/i* — регистр, ячейка памяти или непосредственный операнд;
- *r8, r16, r32* — 8-, 16- , 32-битный регистр;
- *m8, m16, m32* — 8-, 16- , 32- битная ячейка памяти;
- *i8, i16, i32*— 8-,16- , 32- битный непосредственный операнд;
- *ptr16:16* -- указатель адреса: 16-битный сегментный регистр и 16-битное смещение;
- *m16:16* — ячейка памяти, содержащая указатель адреса;

• *rel8 и rel16*— смещение короткое *rel8* (от +127 байт до –128 байт) и смещение прямое *rel16* (от 0 до 65535 байт) в кодовом сегменте относительно адреса следующей команды.

Комментарии 286+ или 386+ в описании команды означают, что команда появилась в процессорах, начиная с модели 80286 или 80386. Отсутствие комментария означает, что это команда базовой системы команд - процессора 8086.

Для некоторых команд через косую черту указаны мнемоники-синонимы.

Операнд-приемник в команде всегда стоит первым, в некоторых случаях он может одновременно являться и источником.

В двухоперандных командах недопустимы оба операнда типа «память -память».

Команды пересылки данных

MOV r/m, r/m/i	Пересылка (копирование) данных из второго операнда в первый. Недопустимые сочетания операндов: - сегм.регистр, сегм.регистр - сегм.регистр, *i* (непосредственный) - *CS, r/m/i*
MOVSX r16, r/m8 **MOVSX** r32. r/m8 **MOVSX** r32,r/m16	Пересылка с расширением знаковым разрядом (386+). Расширяет знак 8-разрядной величины до 16-ти или 32-разрядной, 16-разрядной – до 32-разрядной
MOVZX r16, r/m8 **MOVZX** r32, r/m8 **MOVZX** r32,r/ml6	Пересылка с расширением нулевым разрядом (386+). Расширяет 8-разрядную величину до 16-ти или 32-разрядной, 16-ти разрядную – до 32-разрядной
CBW **CWD** **CWDE** **CDQ**	Преобразование байта из *AL* в слово *AX* (расширением знакового бита) Преобразование слова из *AX* в двойное слово *DX:AX* Преобразование слова из *AX* в двойное слово в *EAX* Преобразование двойного слова из *EAX* в *EDX:EAX*
XCHG r/m, r	Обмен содержимым между двумя операндами
LEA r16,m	Занесение внутрисегментного адреса *m* в регистр *r16*
XLAT/XLATB	Чтение в *AL* байта памяти с адреса *ES:[BX+AL]*
POP r/m16 **POP** r/m32 **POP** sreg	Извлечение слова /двойного слова данных из стека в регистр, память или сегментный регистр (кроме CS). После извлечения значение *SP* увеличится на 2 или 4

POPA	Извлечение данных из стека, сохраненных ранее командой PUSHA, в 16-разрядные регистры *DI, SI, BP, BX, DX, CX, AX* (286+)
POPAD	Чтение данных из стека, сохраненных командой PUSHAD, в 32-разрядные регистры *EDI, ESI, EBP, EBX, EDX, ECX, EAX* (386+)
PUSH r/ml6 **PUSH** r/m32 **PUSH** i16 **PUSH** i32	Запись из регистра/памяти в стек: - слова - двойного слова (386+) - непосредственного операнда-слова (286+) - непосредственного операнда-двойного слова (386+) Перед записью *SP* уменьшится на 2 или 4
PUSHA	Запись в стек регистров *AX, CX, DX, BX, SP* (исходное значение), *BP, SI, DI* (286+). Перед каждой записью *SP* уменьшается на 2
PUSHAD	Запись в стек регистров *EAX, ECX, EDX, EBX, ESP* (исх.значение), *EBP, ESI, EDI* (386+). Перед каждой записью *SP* уменьшается на 4
IN AL, i8 **IN AX**, i8 **IN EAX**, i8	Чтение из порта ввода/вывода с номером i8 байта, слова, двойного слова. Следующие байты считаются из смежных по номеру портов
IN AL, DX **IN AX, DX** **IN EAX, DX**	Чтение из порта ввода/вывода с номером, находящимся в *DX* (байт, слов, двойное слово). Следующие байты считаются из смежных по номеру портов
OUT i8, **AL** **OUT** i8,**AX** **OUT** i8,**EAX**	Запись в порт с номером i8 из *AL, AX* или *EAX* (байт, слово, двойное слово). Последующие байты запишутся в смежные по номеру порты
OUT DX, AL **OUT DX, AX** **OUT DX, EAX**	Запись в порт ввода/вывода с номером, находящимся в *DX* (байт, слово, двойное слово). Последующие байты запишутся в смежные по номеру порты.

Команды двоичной арифметики

ADC r/m, r/m/i	Сложение двух операндов с учетом переноса от предыдущей операции. Непосредственный операнд расширяется до формата операндов знаковым разрядом. r/m := r/m/i + r/m + CF Устанавливаются арифм.флаги
ADD r/m, r/m/i	Сложение двух операндов. Непосредственный операнд расширяется до формата операндов знаковым разрядом. r/m := r/m + r/m/i Устанавливаются арифм.флаги
INC r/m	Инкремент операнда (сложение с 1) r/m= r/m+l Устанавливаются флаги, кроме CF
SBB r/m, r/m/i	Вычитание двух операндов с заемом. Непосредственный операнд расширяется до формата операндов знако-

	вым разрядом. r/m= r/m - r/m/i – CF Устанавливаются арифм.флаги
SUB r/m, r/m/i	Вычитание двух операндов. Непосредственный операнд расширяется до формата операндов знаком. r/m= r/m - r/m/i Устанавливаются арифм.флаги
DEC r/m	Декремент операнда (вычитание 1) r/m := r/m - 1 Устанавливаются флаги, кроме CF
NEG r/m	Изменение знака операнда (вычитание из нуля): r/m= 0 - r/m
CMP r/m/, r/m/i	Сравнение двух операндов. Выполняется как вычитание без сохранения результата, устанавливаются арифметические флаги
MUL r/m8 r/ml6 r/m32	Умножение беззнаковое: *AX= AL* * r/m8 *DX:AX=AX* * r/ml6 *EDX:EAX=EAX* * r/m32 Если старшая половина результата не равна нулю, флаги CF=OF=1, иначе 0
IMUL r/m8 r/ml6 r/m32 rl6, r/m16 r32, r/m32 rl6, r/ml6,i8 r32,r/m32,i8 r16,i8 r32,i8 rl6,r/ml6,i16 r32,r/m32,i32 rl6,i16 r32,i32	Умножение знаковых чисел: *AX= AL * r/m8* *DX:AX= AX* * r/ml6 *EDX:EAX= EAX* * r/m32 (386+) rl6= rl6 * r/m16 (286+) r32= r32 * r/m32 (386+) r16= r/m16 * i8 (286+) r32= r/m32 * i8 (386+) rl6= rl6 * i8 (286+) r32= r32 * i8 (386+) rl6= r/ml6 * i16 (286+) r32= r/m32 * i32 (386+) rl6= rl6 * i16 (286+) r32= r32 * i32 (386+) Непосредственный операнд расширяется знаком до формата сомножителя. Если произведение превышает формат сомножителей, то CF=PF=1, иначе 0
DIV r/m	Деление беззнаковое: *AX* на r/m8 - частное помещается в *AL*, остаток - в *AH*; *DX:AX* на r/m16 - частное в *AX*, остаток - в DX; *EDX:EAX* на r/m32 - частное в *EAX*, остаток в EDX;

IDIV r/m8 **IDIV** r/m16 **IDIV** r/m32	Деление знаковых чисел: *AX* на r/m8 - частное в *AL*, остаток в *AH*; *DX:AX* на r/m16 - частное в *AX*, остаток в DX; *EDX:EAX* на r/m32 - частное в EAX, остаток в EDX;

Команды логических операций

AND r/m, r/m/i	Логическое И. Сброс в 0 тех бит в r/m, которые являются нулевыми в r/m/i
TEST r/m, r/m/i	Логическое И без записи результата, установка флагов
OR r/m, r/m/i	Логическое ИЛИ. Установка в 1 тех бит в r/m, которые являются единичными в r/m/i
XOR r/m, r/m/i	Исключающее ИЛИ. Переключение в обратное состояние тех бит в r/m, которые являются единичными в r/m/i
NOT r/m	Инверсия. Переключение всех бит в обратное состояние.

Команды сдвигов

RCL r/m,1 r/m,CL r/m, i8	Циклический сдвиг влево через бит переноса CF: - на один бит, - на *CL* бит, - на i8-бит (286+)
RCR r/m,1 r/m,CL r/m, i8	Циклический сдвиг вправо через бит переноса CF: - на один бит, - на *CL* бит, - на i8-бит (286+)
ROL r/m,1 r/m,CL r/m,i8	Циклический сдвиг влево: - на один бит, - на *CL* бит, - на i8 бит (286+)
ROR r/m,1 r/m,CL r/m,i8	Циклический сдвиг вправо: - на один бит, - на *CL* бит, - на i8 бит (286+)
SAL r/m,1 r/m,CL r/m,i8	Сдвиг арифметический влево: - на один бит, - на *CL* бит, - на i8 бит (286+). Освобождающиеся разряды заполняются 0.
SAR r/m,1 r/m,CL r/m,i8	Сдвиг арифметический вправо: - на один бит, - на *CL* бит, - на i8 бит (286+) .Освобождающиеся разряды заполняются знаковым битом.
SHL r/m,1 r/m,CL r/m,i8	Сдвиг простой влево: - на один бит, - на *CL* бит, - на i8 бит (286+) . Аналогичен SAL по действию и коду.

SHR r/m,1 r/m,CL r/m,i8	Сдвиг простой вправо: - на один бит, - на *CL* бит, - на i8 бит (286+) . Освобождающиеся разряды заполняются 0.
SHLD r/ml6,rl6,i8 r/m32,r16,i8 r/ml6,r16,CL r/m32,r32,CL	Сдвиг влево с двойной точностью: r/ml6 или r/m32 на i8 бит, вставка данных из r16 или r32 в освободившиеся позиции (386+). Вариант с CL, который является счетчиком сдвигом, аналогичен
SHRD r/ml6,rl6,i8 r/m32,rl6,i8 r/ml6,rl6,CL r/m32,r32,CL	Сдвиг вправо с двойной точностью: r/ml6 или r/m32 на i8 бит, вставка данных из r16 или r32 в освободившиеся позиции (386+). Вариант с CL, который является счетчиком сдвигом, аналогичен

Команды битовых операций (386+)

BSF reg, reg/mem	Сканирование битов вперед. В reg загружается номер (беззнаковое смещение относительно бита 0) самого младшего единичного бита reg/mem. Если в reg/mem только нули, устанавливается флаг *ZF*
BSR reg, reg/mem	Сканирование битов назад. В reg загружается номер (беззнаковое смещение относительно бита 0) самого старшего единичного бита reg/mem. Если в reg/mem только нули, устанавливается флаг *ZF*
BT reg/mem,i8 reg /mem, reg	Тестирование бита. В *CF* копируется бит с номером i8 или reg из reg/mem (386+)
BTC reg/mem, i8 reg /mem, reg	Тестирование и инверсия бита. В *CF* копируется бит с номером i8 или reg из reg/mem, затем исходный бит инвертируется
BTR reg/mem,i8 reg /mem, reg	Тестирование и сброс бита. В *CF* копируется бит с номером i8 или reg из reg/mem, затем исходный бит сбрасывается.
BTS reg /mem, i8 reg /mem, reg	Тестирование и установка бита. В *CF* копируется бит с номером i8 или reg из reg/mem, затем исходный бит устанавливается.

Команды десятичной коррекции

AAA	Десятичная коррекция *AL* после сложения двух однобайтных распакованных BCD-чисел командой *ADD*. В случае

	переноса инкрементируется регистр *AH* и устанавливаются флаги CF и AF, иначе AF=CF=0
AAD	Десятичная коррекция распакованного двузначного BCD-делимого в *AX* перед его делением командой DIV на однобайтный распакованный делитель. Выполняется: *AX←AL+10 *AH*, затем *AH←0*
AAM	Десятичная коррекция *AX* после умножения двух однобайтных распакованных BCD-чисел командой *MUL*
AAS	Десятичная коррекция *AL* после вычитания двух распакованных BCD-чисел командой *SUB*. В случае заема декрементируется регистр *AH* и устанавливаются флаги CF и AF, иначе AF=CF=0
DAA	Десятичная коррекция *AL* после сложения двух однобайтных упакованных BCD-чисел командой *ADD*. В случае десятичного переноса AF=CF=1
DAS	Десятичная коррекция *AL* после вычитания двух упакованных BCD-чисел командой *SUB*. В случае десятичного заема AF=CF=1

Команды передачи управления

JMP rel8 rel16 r16/m16 ptr16:16 m16:16	*Безусловный переход:* - внутрисегментный короткий прямой (относительно следующей команды); - внутрисегментный прямой (относительно следующей команды); - внутрисегментный косвенный по адресу в r16/m16; - межсегментный прямой по указателю сегмента и 16-разрядному смещению; - межсегментный косвенный по адресу в m16:16;
Условные переходы по состоянию флагов и регистра CX/ECX (смещение rel16 только для 386+)	
JC rel8/16	Переход, если есть перенос (CF=1)
JE/JZ rel8/16	Переход, если равно нулю (ZF=1)
JNC rel8/16	Переход, если нет переноса (CF=0)
JNE/JNZ rel8/16	Переход, если не равно (ZF=0)
JNP/JPO rel8/16	Переход, если нечетный паритет (PF=0)
JP/JPE rel8/16	Переход, если четный паритет (PF=1)

JCXZ rel8/16	Переход. если CX=0;
JECXZ rel8/16	Переход. если ECX=0 (386+)
Условные переходы после сравнения беззнаковых величин (смещение rel16 только для 386+)	
JA/JNBE rel8/16	Переход. если выше ((CF или ZF)=0)
JAE/JNB rel8/16	Переход. если не ниже (CF=0)
JB/JNAE rel8/16	Переход. если ниже (CF=1)
JBE/JNA rel8/16	Переход. если не выше (CF или ZF)=1
Условные переходы после сравнения знаковых величин (смещение rel16 только для 386+)	
JG/JNLE rel8/16	Переход. если больше (SF=(OF и ZF))
JGE/JNL rel8/16	Переход. если больше или равно (SF=OF)
JL/JNGE rel8/16	Переход. если меньше (ZF≠OF)
JLE/JNG rel8/16	Переход. если меньше или равно (SF≠OF или ZF=0)
JNO rel8/16	Переход. если нет переполнения (OF=0)
JNS rel8/16	Переход. если неотрицательно (SF=0)
JO rel8/16	Переход. если переполнение (OF=1)
JS rel8/16	Переход. если отрицательно (SF=1)
Команды организации циклов (в качестве вычитающего счетчика циклов используется регистр CX)	
LOOP rel8	*CX=CX-1* и переход. если *CX≠0*
LOOPE/LOOPZ rel8	*CX=CX-1* и переход. если *CX≠0* и ZF=1
LOOPNE/LOOPNZ rel8	*CX=CX-1* и переход. если CX≠0 и ZF=0
Команды вызова процедур и прерываний	
BOUND reg,mem	Проверка индекса массива (reg) на выход за границы, заданные двумя смежными словами/двойными словами в памяти: если (reg<DS:[mem]) или (reg>DS:[mem+2/4]), выполняется прерывание INT 5 (286+)

CALL rel16 r16/m16 ptr16:16 m16:16	Вызов процедуры: - внутрисегментный прямой (относительно следующей команды); - внутрисегментный косвенный по адресу в r16/m16 - межсегментный прямой по указателю сегмента и 16-разрядному смещению; - межсегментный косвенный по адресу в m16:16
INT 3	Выполнение программного прерывания 3 – прерывание трассировки (однобайтный код команды)
INT i8	Выполнение программного прерывания номер i8
INTO	Выполнение программного прерывания 4,если OF=1
IRET **IRETD**	Возврат из прерывания при 16-битных операндах: то же при 32-битных операндах (разные мнемоники для одного кода)
RET i16	Возврат из процедуры (внутрисегментной или межсегментной) с коррекцией значения указателя стека после возврата — *SP=SP+i16*

Строковые команды

CMPSB	Сравнение строк байт, адресуемых *DS:SI* и *ES:DI* с записью результата сравнения в регистр флагов
CMPSW	Сравнение строк слов, адресуемых *DS:SI* и *ES:DI* с записью результата сравнения в регистр флагов
CMPSD	Сравнение строк двойных слов, адресуемых *DS:SI* и *ES:DI* с записью результата сравнения в регистр флагов (386+)
INSB	Чтение байта из порта, указанного в *DX*, в *ES:DI (286+)*
INSW	Чтение слова из порта, указанного в *DX*, в *ES:DI* (286+)
INSD	Чтение двойного слова из порта, указанного в *DX*, в *ES:DI* (386+)
LODSB	Чтение байта из *DS:SI* в *AL*
LODSW	Чтение слова из *DS:SI* в *AX*
LODSD	Чтение двойного слова из *DS:SI* в *EAX* (386+)
MOVSB	Пересылка байта из *DS:SI* в *ES:DI*
MOVSW	Пересылка слова из *DS:SI* в *ES:DI*

MOVSD	Пересылка двойного слова из *DS:SI* в *ES:DI* (386+)
OUTSB	Запись байта из *DS:SI* в порт, указанный в DX (286+)
OUTSW	Запись слова из *DS:SI* в порт, указанный в DX (286+)
OUTSD	Запись двойного слова из *DS:SI* в порт, указанный в DX (386+)
SCASB	Сканирование строки байтов. Сравнение *AL* с байтом из строки *DS:SI*. Установка флага по результату сравнения
SCASW	Сканирование строки слов. Сравнение *AX* со словом из строки *DS:SI*. Установка флага по результату сравнения
SCASD	Сканирование строки двойных слов. Сравнение *EAX* со словом из строки *DS:SI*. Установка флага
STOSB	Запись байта из *AL* в *ES:DI*
STOSW	Запись слова из *AX* в *ES:DI*
STOSD	Запись двойного слова из EAX в *ES:DI* (386+)
REP	Префикс повтора строковых операций до обнуления *CX*. *CX* декрементируется на каждом повторе
REPE/REPZ	Префикс повтора строковых операций *REP*, пока ZF=1
REPNE/REPNZ	Префикс повтора строковых операций *REP*, пока ZF=0

Команды работы с флагами

CLC	Сброс флага переноса (CF=0)
STC	Установка флага переноса (CF=1)
CMC	Инверсия флага переноса (CF=1-CF)
CLD	Сброс флага направления сканирования строк (DF=0)
STD	Установка флага направления сканирования строк (DF=1)
CLI	Запрет маскируемых аппаратных прерываний (IF=0)
STI	Разрешение маскируемых аппаратных прерываний (IF=1)
LAHF	Чтение младшего байта регистра флагов (SF.ZF.AF.PF.CF) в *AH*
SAHF	Запись в младший байт регистра флагов байта из *AH*

SALC	Если CF=1, занести в AL код FF, иначе - 00
POPF	Чтение данных из стека в 16-разрядный регистр флагов
POPFD	Чтение данных из стека в 32-разрядный регистр флагов
PUSHF	Запись в стек 16-разрядного регистра флагов
PUSHFD	Запись в стек 32-разрядного регистра флагов

Команды загрузки указателей сегментов

LDS r16,m16:16	Загрузка указателя сегмента и смещения в регистры DS и r16 из адреса памяти DS:r16
LES r16,m16:16	Загрузка указателя сегмента и смещения в регистры ES и r16 из адреса памяти ES:r16
LFS r16,m16:16	Загрузка указателя сегмента и смещения в регистры FS и r16 из адреса памяти FS:r16
LGS r16,m16:16	Загрузка указателя сегмента и смещения в регистры GS и r16 из адреса памяти GS:r16
LSS r16,m16:16	Загрузка указателя сегмента и смещения в регистры SS и r16 из адреса памяти SS:r16

Системные команды

HLT	Останов процессора
LOCK	Префикс блокировки шины на время выполнения команды
NOP	Нет операции
ENTER i16,i8	Динамическое выделение стека. i16- количество байт, i8- уровень вложенности программ (0 – без вложенности)
LEAVE	Освобождение выделенного стека (обратное *ENTER)*

4. Таблица кодов операций базовых команд

Команда/префикс	Байт кода операции	Вторичный КОП	Устанавливаемые флаги
Команды пересылки данных			
MOV r/m,reg (reg,r/m)	100010dw		
MOV ax/al, metka	1010000w		
MOV metka,ax/al	1010001w		
MOV r/m,data	1100011w	000	
MOV reg,data	1011wreg		
MOV sr,r/m (r/m,sr)	100011d0		
XCHG r/m,reg (reg,r/m)	1000011w		
XCHG ax,reg	10010reg		
LEA reg,m	10001101		
LDS reg,m	11000101		
LES reg,m	11000100		
XLAT	11010111		
LAHF	10011111		
SAHF	10011110		
IN ax/al,port	1110010w		
IN ax/al,DX	1110110w		
OUT port,ax/al	1110011w		
OUT DX,ax/al	1110111w		
PUSH reg	01010reg		
PUSH sr	000sr110		
PUSH r/m	11111111	110	
PUSHF	10011100		
POP reg	01011reg		
POP sr	000sr111		
POP r/m	10001111	000	
POPF	10011101		
Команды преобразования данных			
ADD reg,r/m (r/m,reg)	000000dw		O,S,Z,A,P,C
ADD r/m,data	100000sw	000	O,S,Z,A,P,C
ADD ax/al,data	0000010w		O,S,Z,A,P,C
ADC reg,r/m (r/m,reg)	0001001w		O,S,Z,A,P,C
ADC r/m,data	100000sw	010	O,S,Z,A,P,C
ADC ax/al,data	0001010w		O,S,Z,A,P,C
INC r/m	1111111w	000	O,S,Z,A,P
INC reg16	01000reg		O,S,Z,A,P
SUB reg,r/m (r/m,reg)	001010dw		O,S,Z,A,P,C
SUB r/m,data	100000sw	101	O,S,Z,A,P,C
SUB ax/al,data	0010110w		O,S,Z,A,P,C

SBB reg.r/m (r/m.reg)	000110dw		O,S,Z,A,P,C
SBB r/m.data	100000sw	011	O,S,Z,A,P,C
SBB ax/al,data	0001110w		O,S,Z,A,P,C
DEC r/m	1111111w	001	O,S,Z,A,P
DEC reg16	01001reg		O,S,Z,A,P
CMP reg.r/m (r/m,reg)	001110dw		O,S,Z,A,P,C
CMP r/m.data	100000sw	111	O,S,Z,A,P,C
CMP ax/al.data	0011110w		O,S,Z,A,P,C
NEG r/m	1111011w	011	O,S,Z,A,P,C
MUL r/m	1111011w	100	O,S
IMUL r/m	1111011w	101	O,S
DIV r/m	1111011w	110	
IDIV r/m	1111011w	111	
CBW	10011000		
CWD	10011001		
DAA	00100111		S,Z,A,P,C
DAS	00101111		S,Z,A,P,C
AAA	00110111		A,C
AAS	00111111		A,C
AAM	11010100		S,Z,P
AAD	11010101		S,Z,P
AND reg.r/m (r/m,reg)	001000dw		O,S,Z,P,C
AND r/m.data	100000sw	100	O,S,Z,P,C
AND ax/al.data	0010010w		O,S,Z,P,C
OR reg.r/m (r/m.reg)	000010dw		O,S,Z,P,C
OR r/m.data	100000sw	001	O,S,Z,P,C
OR ax/al.data	0000110w		O,S,Z,P,C
XOR reg.r/m (r/m.reg)	001100dw		O,S,Z,P,C
XOR r/m.data	1000000w	110	O,S,Z,P,C
XOR ax/al,data	0011010w		O,S,Z,P,C
TEST r/m.reg	1000010w		O,S,Z,P,C
TEST r/m.data	1111011w	000	O,S,Z,P,C
TEST ax/al.data	1010100w		O,S,Z,P,C
NOT r/m	1111011w	010	
RCL r/m, 1/CL	110100cw	010	O,C
RCR r/m, 1/CL	110100cw	011	O,C
ROL r/m, 1/CL	110100cw	000	O,C
ROR r/m, 1/CL	110100cw	001	O,C
SHL/SAL r/m, 1/CL	110100cw	100	O,C
SHR r/m, 1/CL	110100cw	101	O,C
SAR r/m, 1/CL	110100cw	111	O,C
Строковые команды			
MOVS/MOVSB/MOVSW	1010010w		

CMPS/CMPSB/CMPSW	1010011w		O,S.Z,A,P,C
LODS/LODSB/LODSW	1010110w		
STOS/STOSB/STOSW	1010101w		
SCAS/SCASB/SCASW	1010111w		O,S,Z,A,P,C
REP	11110011		
REPE/REPZ	11110011		
REPNE/REPNZ	11110010		
Команды передачи управления			
JMP short ptr metka	11101011		
JMP near ptr metka	11101001		
JMP far ptr metka	11101010		
JMP word ptr r/m	11111111	100	
JMP dword ptr m	11111111	101	
JE/JZ metka	01110100		
JNE/JNZ	01110101		
JL/JNGE	01111100		
JLE/JNG	01111110		
JGE/JNL	01111101		
JG/JNLE	01111111		
JB/JNAE/JC	01110010		
JBE/JNA	01110110		
JAE/JNB/JNC	01110011		
JA/JNBE	01110111		
JS	01111000		
JNS	01111001		
JO	01110000		
JNO	01110001		
JP/JPE	01111010		
JNP/JPO	01111011		
JCXZ	11100011		
LOOP metka	11100010		
LOOPE/LOOPZ	11100001		
LOOPNE/LOOPNZ	11100000		
CALL near ptr metka	11101000		
CALL far ptr metka	10011010		
CALL word ptr r/m	11111111	010	
CALL dword ptr m	11111111	011	
RET (внутрисегмент-ный)	11000011		
RET (межсегментный)	11001011		
RET data (внутрисегм.)	11000010		
RET data (межсегмент.)	11001010		
INT n	11001101		I.T

INT 3	11001100		I,T
INTO	11001110		I,T
IRET	11001111		
Команды управления состоянием процессора			
CLD	11111100		D
STD	11111101		D
CLI	11111010		I
STI	11111011		I
CLC	11111000		C
STC	11111001		C
CMC	11110101		C
ESC r/m	11011xxx		
WAIT	10011011		
HLT	11110100		
NOP	10010000		
LOCK	11110000		
Байт-префикс сегмента	001sr110		

5. Таблица ASCII кодов символов

Табл. 1. Кодировки латинских и математических символов

Младшая hex-цифра кода Ascii	Старшая hex-цифра кода Ascii					
	2	3	4	5	6	7
0		0	@	P	`	p
1	!	1	A	Q	a	q
2	"	2	B	R	b	r
3	#	3	C	S	c	s
4	$	4	D	T	d	t
5	%	5	E	U	e	u
6	&	6	F	V	f	v
7	'	7	G	W	g	w
8	(	8	H	X	h	x
9	)	9	I	Y	i	y
A	*	:	J	Z	j	z
B	+	;	K	[	k	{
C	,	<	L	\	l	|
D	-	=	M	]	m	}
E	.	>	N	^	n	~
F	/	?	O	_	o	

Табл. 2. Кодировки символов русского алфавита

Младшая hex-цифра кода Ascii	Старшая hex-цифра кода Ascii				
	8	9	A	E	F
0	А	Р	а	р	Ё
1	Б	С	б	с	е
2	В	Т	в	т	
3	Г	У	г	у	
4	Д	Ф	д	ф	
5	Е	Х	е	х	
6	Ж	Ц	ж	ц	
7	З	Ч	з	ч	
8	й	Ш	й	ш	
9	И	Щ	и	щ	
A	К	Ъ	к	ъ	
B	Л	Ы	л	ы	
C	М	Ь	м	ь	№
D	Н	Э	н	э	
E	О	Ю	о	ю	
F	П	Я	п	я	