Лабораторная работа № 1

Тема: «**Отладчик DEBUG**».

Краткая аннотация: данная работа посвящена знакомству с отладчиком DEBUG, который позволяет:

- набирать небольшие фрагменты ассемблерного кода;
- выполнять трассировку фрагментов кода;
- отслеживать состояние регистров процессора и ячеек оперативной памяти в процессе трассировки;
- тестировать, корректировать, отлаживать набранные в отладчике или загруженные в отладчик фрагменты кода;
- сохранять на диске (в виде исполняемых файлов com-типа) и загружать в отладчик фрагменты кода.

Студент должен освоить макрокоманды отладчика DEBUG, научиться набирать, тестировать, трассировать небольшие фрагменты кода (в пределах 10-20 ассемблерных команд).

Приобретенный в ходе выполнения работы опыт должен позволить в последующем ориентироваться в листингах ассемблерных модулей

ЦЕЛЬ: Знакомство с макрокомандами отладчика, процедурами набора, выполнения и трассировки фрагментов ассемблерного кода.

Макрокоманды отладчика DEBUG.

Краткие сведения об отладчике DEBUG :

Debug – 16-разрядная утилита операционной системы MS DOS.

Название программы происходит от Bugs – насекомые. На программистском слэнге Bugs означает «ошибки в программе».

Используя Debug для пошаговой трассировки тестируемого кода, эти ошибки можно найти и исправить.

Процедура загрузки отладчика DEBUG :

Располагается отладчик Debug в системной папке system32 папки WINDOWS (файл debug.exe).

Запуск отладчика удобнее всего выполнять из меню Пуск/Выполнить.

Режимы работы отладчика DEBUG :

- **Основной режим работы отладчика Debug (характерная особенность – присутствие чёрточки и мигающего справа от неё курсора). Основной режим работы устанавливается автоматически после загрузки отладчика.**

В этом режиме можно набирать любые макрокоманды отладчика от A до W. Выход из этого режима означает завершение работы отладчика.

- Режим ассемблирования. Вход в режим ассемблирования из основного режима работы отладчика осуществляется набором команды A. Признак нахождения в этом режиме – наличие адреса (например, 0CC0:0100) перед знаком курсора:

```
-A 100
0CC0:0100 MOV DX,140
0CC0:0103 
```

После набора команды необходимо нажатием клавиши Enter ввести её в память (пока не нажата клавиша Enter команда в память не введена).

Если команда набрана с ошибкой и нажата клавиша Enter, эта команда в память не вводится. Вместо этого отладчик предлагает ввести её повторно, по тому же самому адресу:

```
-A 100
0CFC:0100 MOV DX,14000
                     ^ ошибка
0CFC:0100 MOV DX,1400
0CFC:0103 
```

Возвращение из режима ассемблирования в основной режим работы отладчика – нажатие клавиши Enter в пустой строке (когда не набрано ни одного символа), например, в строке 0CFC:0103.

- Режим ввода данных. Данный режим может быть реализован во время выполнения макрокоманд E и R. В этом режиме вводятся данные в ячейки оперативной памяти или в регистры процессора:

```
-E 140
0CC0:0140  00.
```

Возвращение из этого режима в основной режим работы отладчика – нажатие клавиши Enter. Продолжение работы в режиме ввода данных – нажатие клавиши пробел. Нажимая клавишу пробел, можно попасть на любую ячейку памяти и ввести в неё нужное число. Ввод значений завершается нажатием клавиши Enter.

- Режим выполнения кода. Вход в режим выполнения кода из основного режима работы отладчика осуществляется набором команды G. Как правило, время нахождения в данном режиме – считанные мгновенья. Исключение – зависание программы или выполнение огромных циклов.

Возвращение в основной режим работы отладчика осуществляется автоматически, без участия работающего с этим отладчиком человека.

- Режим ввода информации с клавиатуры во время работы в режиме выполнения кода. Завершение этого режима и продолжение выполнения кода осуществляется нажатием клавиши Enter.

- Режим трассировки выполняемого кода. В данном режиме процессор выполняет заранее заданное ограниченное количество команд. Вход в режим трассировки осуществляется набором команды T. Возвращение в основной режим работы осуществляется автоматически, без участия работающего с отладчиком человека.

- **Аварийное прекращение работы отладчика. Имеет место в случае, когда процессор пытается исполнить запрещённый в режиме эмуляции работы микропроцессора i8086 программный код. Например, пытается выполнить команды работы с портами ввода-вывода данных.**

Команды отладчика Debug:

```
-?
assemble      A [адрес]
compare       C диапазон адрес
dump          D [диапазон]
enter         E адрес [список]
fill          F диапазон список
go            G [=адрес] [адреса]
hex           H значение1 значение2
input         I порт
load          L [адрес] [диск] [превый_сектор] [число]
move          M диапазон адрес
name          N [путь] [список_аргументов]
output        O порт байт
proceed       P [=адрес] [число]
quit          Q
register      R [регистр]
search        S диапазон список
trace         T [=адрес] [значение]
unassemble    U [диапазон]
write         W [адрес] [диск] [первый_сектор] [число]
выделение памяти EMS            XA [#число_страниц]
освобождение памяти EMS         XD [дескриптор]
сопоставление страниц EMS       XM [Lстраница] [Pстраница] [дескриптор]
вывод состояния памяти EMS      XS
-
```

В приведённом выше окне фигурирует полный перечень команд отладчика. Ниже тот же перечень приведён в более удобном для чтения виде:

```
-?
Assemble        A  [адрес]
Compare         C  диапазон адрес
Dump            D  [диапазон]
Enter           E  адрес [список]
Fill            F  диапазон список
Go              G  [=адрес] [адреса]
Help            ?
Hex             H  значение1 значение2
Input           I  порт
Load            L  [адрес] [диск] [первый_сектор] [число]
Move            M  диапазон адрес
Name            N  [путь 1 [список_аргументов]
Output          O  порт байт
Proceed         P  [=адрес] [число]
Quit            Q
Register        R  [регистр]
Search          S  диапазон список
Trace           T  [=адрес] [значение]
Unassembled     U  [диапазон]
Write           W  [адрес] [диск] [первый_сектор] [число]
```

Параметры макрокоманд отладчика DEBUG :

1. **Значения** задаются в 16-ой системе счисления (отметки типа **hex** или **h** не ставятся).
1-байтовые значения задаются **двузначными 16-ми числами**.
2-байтовые значения задаются **четырёхзначными 16-ми числами**.
Например, **80** **(1-байтовое)**, **1A80** **(2-байтовое)**.

2. **Адреса** задаются в 16-ой системе счисления.
Например, **0100**

3. **Диапазоны** адресов задаются тремя способами:
Например, диапазон **0100..011F**:
1-ый способ: **0100 L20**

2-ой способ: **0100 011F**
3-ий способ: **0100,011F**

4. **Списки** задается набором своих элементов (строк или однобайтовых чисел). Элементы списка отделяются друг от друга запятыми или пробелами.

Например (приводятся 4 варианта задания одного и того же списка):
1-ый способ: **'AA','BB','CC'**
2-ой способ: **'AA' 'BB' 'CC'**
3-ий способ: **41,41,42,42,43,43**
4-ый способ: **41 41 42 42 43 43**

В приводимом ниже комментарии дана краткая характеристика каждой макрокоманды отладчика:

A **Вход в режим ассемблирования (набора ассемблерных команд)**
C **Сравнение содержимого ячеек двух диапазонов**
D **Вывод содержимого ячеек заданного диапазона**
E **Ввод данных в ячейки оперативной памяти**
F **Заполнение ячеек диапазона заданным рядом значений**
G **Запуск на исполнение**
? **Вызов списка команд отладчика**
H **Калькулятор: вычисление суммы и разности заданных операндов**
L **Загрузка файла**
M **Перемещение ячеек указанного диапазона**
N **Задание имени сохраняемого / загружаемого файла (до выполнения команд L и W)**
Q **Выход из отладчика**
R **Просмотр регистров / изменение содержимого указанного регистра**
S **Поиск образца в заданном диапазоне**
T **Трассировка с заданного адреса указанного количества команд**
U **Вывод списка ассемблерных команд заданного дипазона**
W **Запись на диск заданного фрагмента кода**

Макрокоманда ? отладчика DEBUG :

Выводит на экран перечень макрокоманд отладчика от A до W.

Результат её выполнения приведён выше.

Макрокоманда D отладчика DEBUG :

Выводит на экран содержимое ячеек заданного диапазона. Ниже приведены образцы выполнения команды без параметров, с одним параметром, с двумя параметрами.

Выполнение команды D:

```
C:\WINDOWS\system32\debug.exe
-D
0CC0:0100  05 01 04 BD 05 03 04 D2-05 05 04 ED 05 06 04 FC   ................
0CC0:0110  05 08 04 10 06 09 04 22-06 0A 04 34 34 00 AF 0C   ......."...44...
0CC0:0120  06 0D 04 5A 06 0E 04 6C-06 0F 04 85 06 10 04 C6   ...Z...l........
0CC0:0130  06 14 04 DA 06 16 04 ED-06 17 04 2A 07 18 04 4C   ...........*...L
0CC0:0140  07 19 04 61 07 1A 04 6D-07 1B 04 98 07 1C 04 AE   ...a...m........
0CC0:0150  07 1D 04 CF 07 1E 04 0E-08 1F 04 49 08 20 04 57   ...........I. .W
0CC0:0160  08 21 04 66 08 23 04 73-08 24 04 75 08 25 04 76   .!.f.#.s.$.u.%.v
0CC0:0170  08 27 04 94 08 28 04 93-08 29 04 92 08 2A 04 91   .'...(...)...*..
-
```

Повторное выполнение команды D:

```
-D
0CC0:0180  08 2B 04 90 08 2C 04 8E-08 2D 04 95 08 2E 04 95   .+...,...-......
0CC0:0190  08 2F 04 94 08 30 04 93-08 31 04 98 08 32 04 9D   ./...0...1...2..
0CC0:01A0  08 33 04 A2 08 34 04 A4-08 35 04 A3 08 36 04 A7   .3...4...5...6..
0CC0:01B0  08 37 04 BC 08 38 04 C2-08 39 04 CE 08 3C 04 E8   .7...8...9...<..
0CC0:01C0  08 42 04 F5 08 43 04 FF-08 44 04 15 09 45 04 2E   .B...C...D...E..
0CC0:01D0  09 46 04 49 09 47 04 79-09 48 04 98 09 B0 04 D6   .F.I.G.y.H......
0CC0:01E0  09 14 05 D3 09 28 05 74-0A 29 05 B9 0A 3C 05 2D   .....(.t.)...<.-
0CC0:01F0  0B 3D 05 78 0B 3E 05 C3-0B 50 05 4D 0C 78 05 62   .=.x.>...P.M.x.b
-
```

Видно, что начальный адрес автоматически увеличился на 80 (80= 10*8 - система 16-ричная, то есть выведено 8 рядов по 16 значений в каждом ряду).

Смысл приведённых выше данных:

```
-D
0CFC:0100  20 E4 A0 A9 AB A0 0D 0A-38 8D A5 A4 AE E1 E2 A0   .......8.......
0CFC:0110  E2 AE E7 AD AE 20 AF A0-AC EF E2 A8 34 00 EB 0C   ..... ......4...
0CFC:0120  20 AF A5 E0 A5 AC A5 AD-AD EB E5 20 E1 E0 A5 A4   ........... ....
0CFC:0130  EB 20 28 45 6E 76 69 72-6F 6E 6D 65 6E 74 29 0D   . (Environment).
0CFC:0140  0A 1B 8E E8 A8 A1 AA A0-20 AF E0 A8 20 E1 AE A7   ........ ... ...
0CFC:0150  A4 A0 AD A8 A8 20 E4 A0-A9 AB A0 0D 0A 1B 8E E2   ..... ..........
0CFC:0160  E1 E3 E2 E1 E2 A2 E3 A5-E2 20 AF A0 AA A5 E2 AD   ......... ......
0CFC:0170  EB A9 20 E4 A0 A9 AB 0D-0A 23 0D 0A 82 E1 E2 A0   .. .......#......
-
```

Числа, содержащиеся в ячейках оперативной памяти

Колонка адресов ячеек оперативной памяти

Символы, соответствующие числам, содержащимся в ячейках оперативной памяти

Команда D может использоваться с параметром – начальным адресом диапазона:

```
-D 110
0CC0:0110  05 08 04 10 06 09 04 22-06 0A 04 34 34 00 AF 0C   ......."...44...
0CC0:0120  06 0D 04 5A 06 0E 04 6C-06 0F 04 85 06 10 04 C6   ...Z...l........
0CC0:0130  06 14 04 DA 06 16 04 ED-06 17 04 2A 07 18 04 4C   ...........*...L
0CC0:0140  07 19 04 61 07 1A 04 6D-07 1B 04 98 07 1C 04 AE   ...a...m........
0CC0:0150  07 1D 04 CF 07 1E 04 0E-08 1F 04 49 08 20 04 57   ...........I. .W
0CC0:0160  08 21 04 66 08 23 04 73-08 24 04 75 08 25 04 76   .!.f.#.s.$.u.%.v
0CC0:0170  08 27 04 94 08 28 04 93-08 29 04 92 08 2A 04 91   .'...(...)...*..
0CC0:0180  08 2B 04 90 08 2C 04 8E-08 2D 04 95 08 2E 04 95   .+...,...-......
-
```

Команда D может использоваться с двумя параметрами. Возможные варианты:

```
-D 110 L10
0CC0:0110  05 08 04 10 06 09 04 22-06 0A 04 34 34 00 AF 0C   ......."...44...
-D 110,11F
0CC0:0110  05 08 04 10 06 09 04 22-06 0A 04 34 34 00 AF 0C   ......."...44...
-D 110 11F
0CC0:0110  05 08 04 10 06 09 04 22-06 0A 04 34 34 00 AF 0C   ......."...44...
-
```

Выше приведены 3 способа задания параметров диапазона.

```
-D 110 11F
0CC0:0110  05 08 04 10 06 09 04 22-06 0A 04 34 34 00 AF 0C   ......."...44...
-
```

0110 0111 0112 0113 0114 011D 011E 011F

Выше приведено соотношение адресов и чисел, располагающихся в ячейках с этими адресами.

Макрокоманда F отладчика DEBUG :

Позволяет заполнять ячейки памяти элементами задаваемого списка значений.

```
-F 100 L60 0
-D 100 L60
0CC0:0100  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0CC0:0110  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0CC0:0120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0CC0:0130  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0CC0:0140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0CC0:0150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

В данном случае ячейки заданного диапазона заполнены нулями.

Макрокоманда E отладчика DEBUG :

Предназначена для ввода значений в ячейки оперативной памяти. Ниже приведены образцы выполнения команды с одним параметром и двумя параметрами.

Варианты выполнения команды E приведены ниже.

```
-E 120 '0123456789'
-D 120 L10
0CC0:0120  30 31 32 33 34 35 36 37-38 39 00 00 00 00 00 00   0123456789......
-
```

Выше приведена команда ввода в ячейки 120..129 строки '0123456789'.

```
-E 12A 24
-D 120 L10
0CC0:0120  30 31 32 33 34 35 36 37-38 39 24 00 00 00 00 00   0123456789$.....
-
```

Выше приведена команда ввода в ячейку 12A числа 24 (ASCII-код знака '$').

```
-E 130 0 1 2 3 4 5 6 7 8 9
-D 130 L10
0CC0:0130  00 01 02 03 04 05 06 07-08 09 00 00 00 00 00 00   ................
-
```

Выше приведена команда ввода в ячейки 130..139 строки чисел 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

```
-E 140
0CC0:0140  00.
```

Выше приведена команда ввода в ячейки, начиная с адреса 140, значений непосредственно с клавиатуры. В ячейке с адресом 140 находится 0. Запечатлён момент ожидания ввода какого-либо значения.

```
-E 140
0CC0:0140  00.40   00.41   00.42   00.43   00.44   00.45   00.46   00.47
-D 140 L10
0CC0:0140  40 41 42 43 44 45 46 47-00 00 00 00 00 00 00 00   @ABCDEFG........
-
```

Выше приведена команда ввода в ячейки 140..147, чисел 40, 41, 42, 43, 44, 45, 46, 47, которые представляют собою ASCII -коды символов строки '@ABCDEFG'. Ввод осуществляется непосредственно с клавиатуры.

```
-E 148 '$'
-D 140 L10
0CC0:0140  40 41 42 43 44 45 46 47-24 00 00 00 00 00 00 00   @ABCDEFG$.......
-
```

Выше приведена команда ввода в ячейку 148 символа '$'.

Макрокоманда A отладчика DEBUG :

Макрокоманда предназначена для входа в режим ассемблирования. В этом режиме осуществляется набор ассемблерных команд. Ниже приведены образцы выполнения этой команды.

```
-A 100
0CC0:0100
```

Выше: вход в режим ассемблирования состоялся, однако, пока не набрано ни одного символа.

```
-A 100
0CC0:0100 MOV DX,140
```

Выше: первая команда набрана, однако, клавиша Enter не нажата. Это означает, что команда в ячейки оперативной памяти не введена.

```
-A 100
0CC0:0100 MOV DX,140
0CC0:0103
```

Выше: первая команда набрана и клавиша Enter нажата. Введённая в память команда MOV DX,140 занимает в памяти ячейки с адресами 100, 101, 102. Отладчик ожидает ввода следующей команды с адреса 103.

```
-A 100
0CC0:0100 MOV DX,140
0CC0:0103 MOV AH,9
0CC0:0105 INT 21
0CC0:0107 INT 20
0CC0:0109
```

Выше: 4 команды набраны и введены в ячейки оперативной памяти. Отладчик ожидает либо ввода 5-ой команды, либо выхода из режима ассемблирования.

```
-A 100
0CC0:0100 MOV DX,140
0CC0:0103 MOV AH,9
0CC0:0105 INT 21
0CC0:0107 INT 20
0CC0:0109
-
```

Выше: выход из режима ассемблирования (нажатие клавиши Enter) состоялся.

Макрокоманда U отладчика DEBUG :

Макрокоманда U дисассемблирует код, расположенный в указанном диапазоне, обеспечивая вывод на экран содержимого ячеек этого диапазона, интерпретируемого как набор ассемблерных команд.

```
-U 100 108
0CC0:0100 BA4001        MOV     DX,0140
0CC0:0103 B409          MOV     AH,09
0CC0:0105 CD21          INT     21
0CC0:0107 CD20          INT     20
-
```

Выше: набор 4-х команд. Рассмотрим содержимое ячеек памяти, занятых кодами этих команд.

MOV DX,0140 занимает 3 ячейки памяти 100..102

- **100 – код команды MOV DX,... (число BA)**
- **101 – младший байт (40) числа 140**
- **102 – старший байт (1) числа 140**

MOV AH,09 занимает 2 ячейки памяти 103..104

- **103 – код команды MOV AH,... (число B4)**
- **104 – 1-байтовый операнд (9) команды**

INT 21 занимает 2 ячейки памяти 105..106

- **105 – код команды INT (число CD)**
- **106 – 1-байтовый операнд (21) команды**

NT 20 занимает 2 ячейки памяти 107..108

- **107 – код команды INT (число CD)**
- **108 – 1-байтовый операнд (20) команды**

Псевдокоманды отладчика DEBUG :

Псевдокоманды отладчика набираются в режиме ассемблирования (**Assemble**) отладчика. Псевдокоманды не компилируются в машинные коды.

Псевдокоманда **DB:**

позволяет заполнить текущие байты числами или строкой. Например:

0138 **DB 41** – ввод числа **41** (ASCII-код символа '**A**') в ячейку памяти (адрес смещения = **0138**)

0139 **DB 'A'** – ввод символа '**A**' в ячейку памяти (адрес смещения = **0139**)

013A **DB 42 43 44 45** – ввод ряда чисел (**42,43,44,45**) в ячейки памяти (адреса смещения = **013A, 013B, 013C, 013D**)

013E **DB 'BCDE'** – ввод строки '**BCDE**' в ячейки памяти (адреса смещения = **013E, 013F, 0140, 0141**)

Псевдокоманда **DW:**

позволяет заполнить текущие пары байтов числами или строкой. Например:

0138 **DW 4142** – ввод слова **4142** в ячейки памяти (адрес смещения = **0138**)

013A **DW 'AB'** – ввод символов '**A**' и '**B**' в ячейки памяти (адреса смещения = **013A..013B**)

013C **DW 4243 4445** – ввод ряда чисел (**4243** и **4445**) в ячейки памяти (адреса смещения = **013C** и **013E**)

0140 **DW 'BCDE'** – ввод строки '**BCDE**' в ячейки памяти (адреса смещения = **0140..0143**)

Псевдокоманда **ORG:**

позволяет осуществлять передачу управление в режиме ассемблирования. Например:

0100 **ORG 0110** – передача управления в режиме ассемблирования на адрес смещения **0110**

0110 ■ – после передачи управления адрес смещения принял значение **0110**

Макрокоманда G отладчика DEBUG :

Макрокоманда запускает на исполнение код с указанного адреса.

Рассмотрим запуск на исполнение программы ввода с клавиатуры строки.

```
-E 150 4
-E 101 50
-E 104 A
-D 150 L10
0CC0:0150  04 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-U 100 108
0CC0:0100 BA5001        MOV     DX,0150
0CC0:0103 B40A          MOV     AH,0A
0CC0:0105 CD21          INT     21
0CC0:0107 CD20          INT     20
-
```

Комментарий к приводимому выше результату выполнения макрокоманд:

E 150 4 - занесение в ячейку 150 числа 4 (ограничитель на количество вводимых с клавиатуры символов)

E 101 50 - корректировка содержимого ячейки 101 (исправление адреса на 150)

E 104 A - корректировка содержимого ячейки 104 (исправление номера функции прерывания INT 21 на 0A – ввод строки с клавиатуры)

D 150 L10 – просмотр ячеек 150..15Fдо запуска на исполнение приводимого выше кода.

U 100 108 – дисассемблирование диапазона 100..108 с целью просмотра команд.

```
-G=100
```

Команда запуска кода с адреса 100 набрана, но не запущена. Отладчик ожидает нажатия клавиши Enter.

```
-G=100

```

Программа запущена и ожидает ввода с клавиатуры строки.

```
-G=100
ABC
```

Строка 'ABC' набрана, но в память ещё не введена (не нажата клавиша Enter).

```
-G=100
ABC
Программа завершилась нормально
-
```

Работа программы завершилась нормально. Смотрим результаты:

```
-D 150 L10
0CC0:0150  04 03 41 42 43 0D 00 00-00 00 00 00 00 00 00 00   ..ABC...........
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
```

0D - метка окончания символов введённой строки

41 42 43 - ASCII-коды символов (**ABC**) введённой строки

3- Количество введённых с клавиатуры символов (3 символа)

4 - Ограничитель на количество вводимых символов (не более 3-х символов)

<u>Работа с файлами в отладчике Debug:</u>

Работа с файлами в DEBUG может начинаться с момента загрузки самого отладчика.

<u>Запуск отладчика **DEBUG:**</u>

Запуск отладчика можно осуществлять двумя способами:

1-ый способ:

Debug Запуск отладчика

2-ой способ:

Debug prim.com Запуск отладчика с последующей загрузкой файла **prim.com** с адреса смещения **0100** (при этом, в пару регистров **BX:CX** заносится размер загружаемого файла)

Загрузка и запись файлов в отладчике **DEBUG:**

Загрузка и запись файлов в отладчике осуществляется командами отладчика:

L [адрес] - команда загрузки файла с указанного адреса смещения

W [адрес] - команда записи файла с указанного адреса смещения

Перед выполнением этих команд следует выполнить два действия:

1. Командой **N** задать полное имя файла и путь к нему.
2. В пару регистров **BX:CX** занести размер загружаемого / записываемого файла.

Пример: сохранить диапазон адресов **100..11B** в файле **PRIMER.COM.**

-N C:\TEMP\PRIMER.COM – ввод имени файла **PRIMER.COM** (расположен в папке **TEMP** диска **C**)

-R – просмотр содержимого регистров процессора (интерес представляет пара регистров **BX:CX**)

-R CX – вход в режим изменения содержимого регистра **CX** (с целью занесения в регистр числа **1C** - количества сохраняемых ячеек диапазона адресов **100..11B**)

-W 100 – запись в файл **PRIMER.COM** содержимого **1C** ячеек памяти, начиная с адреса **100**

Задания к лабораторной работе:

Выполнить следующие задания:

1. **Изучить содержание лабораторной работы (в часы самоподготовки).**
2. **Повторить выполнение разобранных в работе примеров (в часы самоподготовки).**
3. **Законспектировать примеры применения команд (в часы самоподготовки).**
4. **Выполнить следующие команды и прокомментировать результаты их выполнения:**

```
-G= 100

-D 140 L40

-A 118

-U 100,11B

-E 140 41 42
```

```
-E 140 'ABCD'

-E 140

-F 140 L40 0
```

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА:

1. Абель П. Язык ассемблера для IBM PC и программирования. - М.: Высшая школа, 1992.- 447 с.

2. Лямин Л.В. Макроассемблер MASM. - М.: Радио и связь, 1994.- 320 с.

3. Нортон П., Соухэ Д. Язык ассемблера для IBM PC. – М.: "Компьютер", Финансы и статистика, 1992. – 352 с.

4. Сван Т. Освоение Turbo Assembler. – Киев: "Диалектика", 1996.

5. Юров В. Assembler: учебный курс. – СПб: Питер, 1998.

6. Юров В., Хорошенко С. Assembler : учебный курс.-Санкт-Петербург: Питер, 1999. -672

7. Юров В. Assembler: Специальный справочник. – СПб: Питер, 2000.

8. Юров В. ASSEMBLER : практикум. (с дискетой).-Санкт-Петербург: Питер, 2002. -400

9. Юров В. Assembler: практикум – СПб: Питер, 2003.- 400 с.

10. Пирогов В.Ю. Ассемблер на примерах. - СПб.: БХВ-Петербург, 2005. - 416 с.

11. Пирогов В.Ю. Ассемблер для Windows. - 3-е изд. - СПб.: БХВ-Петербург, 2005. - 864 с.

12. Шнайдер О. Язык ассемблера для персонального компьютера фирмы IBM. Пер. с англ. под ред. Е.К.Масловского.-Москва: Мир, 1988. -405С.

13. Юров В.И. Assembler. Учебник для вузов. 2-ое изд. – СПб: Питер, 2005. - 637 с.

14. Использование Turbo Assembler при разработке программ. – Киев: "Диалектика", 1994. - 288 с.

Лабораторная работа № 2

Тема: «**Микропроцессор i8086: простейшие команды**».

Краткая аннотация: данная работа посвящена знакомству с простейшими командами микропроцессора, которые позволяют решать широкий круг несложных задач, как-то:

- копировать содержимое диапазона ячеек оперативной памяти в целевой диапазон;
- заполнять диапазон ячеек оперативной памяти заданным набором числовых значений;
- выполнять поиск и замену образцов в заданном диапазоне ячеек оперативной памяти;
- выполнять подсчёт количества образцов в заданном диапазоне ячеек оперативной памяти;
- выполнять поиск максимального и минимального значений в заданном диапазоне ячеек оперативной памяти;
- выполнять суммирование содержимого ячеек заданного диапазона ячеек оперативной памяти.

Студент должен освоить основные команды процессора, научиться применять их для составления небольших фрагментов кода (в пределах 10-20 ассемблерных команд).

Приобретенный в ходе выполнения работы опыт должен позволить в последующем ориентироваться в листингах ассемблерных модулей

ЦЕЛЬ: Знакомство с командами микропроцессора i8086, процедурой использования команд для составления небольших фрагментов кода.

Команды микропроцессора i8086.

Способы адресации команд: непосредственная, прямая и косвенная **(на примере команды MOV):**

Адрес начала команды	Команда, записанная на языке машинных кодов	Та же самая команда, записанная на языке ассемблер	Комментарий, поясняющий, как выполняется команда
100	B8 34 12	MOV AX,1234	**НЕПОСРЕДСТВЕННАЯ АДРЕСАЦИЯ:** Занесение 2-х байтового числа **1234** (**12** – старший байт числа, **34** – младший байт числа) в 2-х байтовый регистр **AX**
103	A3 20 01	MOV [0120],AX	**ПРЯМАЯ АДРЕСАЦИЯ:** Копирование 2-х байтового числа, находящегося в **AX**, в однобайтовые ячейки оперативной памяти **0120** и **0121** (в **0120** копируется младший байт **AX**, в **0121** копируется старший байт **AX**)
106	8B 1E 20 01	MOV BX,[0120]	**ПРЯМАЯ АДРЕСАЦИЯ:** Копирование содержимого ячеек оперативной памяти **0120** и **0121**, в 2-х байтовый регистр **BX** (в младший байт **BX**, регистр **BL**, копируется

			содержимое ячейки **0120**, в старший байт **BX**, регистр **BH**, копируется содержимое ячейки **0121**)
10A	BD 38 01	MOV BP,0138	**НЕПОСРЕДСТВЕННАЯ АДРЕСАЦИЯ:** Занесение 2-х байтового числа **0138** в 2-х байтовый регистр **BP**
10D	88 46 00	MOV [BP+00],AL	**КОСВЕННАЯ АДРЕСАЦИЯ:** Копирование однобайтового числа, находящегося в **AL**, в однобайтовую ячейку оперативной памяти, адрес которой содержится в регистре **BP**, т.е. в ячейку с вычисляемым адресом **0138** (**0138**+**00**=**0138**)
110	89 46 08	MOV [BP+07],AX	**КОСВЕННАЯ АДРЕСАЦИЯ:** Копирование 2-х байтового числа, находящегося в AX, в ячейки оперативной памяти, на адреса которых указывает регистр BP: младший байт регистра AX, именуемый AL, копируется в ячейку с вычисляемым адресом **013F** (**0138** + **07** + **0** = **013F**) старший байт регистра **AX**, именуемый **AH**, копируется в ячейку с вычисляемым адресом **0140** (**0138** + **07** + **1** = **0140**)
113	CD 20	INT 20	Команда завершения работы программы

Команда ADD:

Адрес начала команды	Команда, записанная на языке машинных кодов	Та же самая команда, записанная на языке ассемблер	Комментарий, поясняющий, как выполняется команда
100	05 34 12	ADD AX,1234	**НЕПОСРЕДСТВЕННАЯ АДРЕСАЦИЯ:** Сложение 2-х байтового числа **1234** (**12** – старший байт числа, **34** – младший байт числа) и содержимого 2-х байтового регистра **AX.** Результат размещается в регистре **AX.**
103	01 06 20 01	ADD [0120],AX	**ПРЯМАЯ АДРЕСАЦИЯ:** Сложение 2-х байтового числа, находящегося в **AX**, с 2-х байтовым числом, расположенном в ячейках оперативной памяти **0120** и **0121** (с содержимым ячейки **0120** складывается содержимое младшего байта **AX**, а с содержимым ячейки **0121** складывается содержимое старшего байта **AX**). **Результат размещается в ячейках 0120 и 0121.**

107	03 1E 20 01	ADD BX,[0120]	**ПРЯМАЯ АДРЕСАЦИЯ:** Сложение содержимого ячеек оперативной памяти **0120** и **0121** **с** содержимым 2-х байтового регистра **BX** (к младшему байту регистра **BX**, регистру **BL**, добавляется содержимое ячейки **0120**, а к содержимому старшего байта регистра **BX**, регистра **BH**, добавляется содержимое ячейки **0121**). **Результат размещается в регистре BX.**
10B	BD 38 01	MOV BP,0138	**НЕПОСРЕДСТВЕННАЯ АДРЕСАЦИЯ:** Занесение 2-х байтового числа **0138** в 2-х байтовый регистр **BP**
10E	00 46 20	ADD [BP+20],AL	**КОСВЕННАЯ АДРЕСАЦИЯ:** Сложение однобайтового числа, находящегося в **AL**, с содержимым 1-байтовой ячейки оперативной памяти, адрес которой определяется как сумма содержимого регистра **BP** и смещения **20**, т.е. с содержимым ячейки с вычисляемым адресом **0158** (**0138**+**20**=**0158**). **Результат размещается в ячейке 0158.**
111	01 46 E0	ADD [BP-20],AX	**КОСВЕННАЯ АДРЕСАЦИЯ:** Сложение 2-байтового числа, находящегося в **AX** с содержимым ячеек оперативной памяти, адреса которых вычисляются как **BP**-**20**: младший байт регистра **AX**, именуемый **AL**, складывается с содержимым ячейки, вычисляемый адрес которой определяется как следующая сумма: **0118** (**0138** - **20** +**0** = **0118**) старший байт регистра **AX**, именуемый **AH**, складывается с содержимым ячейки с вычисляемым адресом **0119** (**0138** - **20** + **1** = **0119**) **Результат размещается в ячейках 0118 и 0119.**
114	CD 20	INT 20	Команда завершения работы программы

Трассировка представленного выше кода :

Позволяет проследить как выполняется операция сложения.

```
-u 100 113
0CC0:0100 053412        ADD     AX,1234
0CC0:0103 01062001      ADD     [0120],AX
0CC0:0107 031E2001      ADD     BX,[0120]
0CC0:010B BD3801        MOV     BP,0138
0CC0:010E 004620        ADD     [BP+20],AL
0CC0:0111 0146E0        ADD     [BP-20],AX
-d 118 150
0CC0:0110                          00 00 00 00 00 00 00 00          ........
0CC0:0120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0CC0:0130  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0CC0:0140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0CC0:0150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0CC0:0160  00 00 00 00 00 00 00 00                          ........
-t=100 3

AX=1234  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CC0  ES=0CC0  SS=0CC0  CS=0CC0  IP=0103   NV UP EI PL NZ NA PO NC
0CC0:0103 01062001      ADD     [0120],AX                       DS:0120=0000

AX=1234  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CC0  ES=0CC0  SS=0CC0  CS=0CC0  IP=0107   NV UP EI PL NZ NA PO NC
0CC0:0107 031E2001      ADD     BX,[0120]                       DS:0120=1234

AX=1234  BX=1234  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0CC0  ES=0CC0  SS=0CC0  CS=0CC0  IP=010B   NV UP EI PL NZ NA PO NC
0CC0:010B BD3801        MOV     BP,0138
-D 120 110
0CC0:0120  34 12 00 00 00 00 00 00-00 00 00 00 00 00 00 00  4...............
-
```

Первая команда ADD AX,1234 приводит к тому, что AX= 0+1234= 1234

Вторая команда ADD [0120], AX приводит к результату: [0120]= [0120] + AX = 0 + 1234 = 1234 ([0120]= 34 [0121]= 12)

Третья команда ADD BX,[0120] приводит к результату: BX= BX + 1234= 1234

Трассировка второй группы команд даёт следующие результаты:

```
-t=10B 3

AX=1234  BX=1234  CX=0000  DX=0000  SP=FFEE  BP=0138  SI=0000  DI=0000
DS=0CC0  ES=0CC0  SS=0CC0  CS=0CC0  IP=010E   NV UP EI PL NZ NA PO NC
0CC0:010E 004620        ADD     [BP+20],AL                      SS:0158=00

AX=1234  BX=1234  CX=0000  DX=0000  SP=FFEE  BP=0138  SI=0000  DI=0000
DS=0CC0  ES=0CC0  SS=0CC0  CS=0CC0  IP=0111   NV UP EI PL NZ NA PO NC
0CC0:0111 0146E0        ADD     [BP-20],AX                      SS:0118=0000

AX=1234  BX=1234  CX=0000  DX=0000  SP=FFEE  BP=0138  SI=0000  DI=0000
DS=0CC0  ES=0CC0  SS=0CC0  CS=0CC0  IP=0114   NV UP EI PL NZ NA PO NC
0CC0:0114 06            PUSH    ES
-d 118 15f
0CC0:0110                          34 12 00 00 00 00 00 00          4.......
0CC0:0120  34 12 00 00 00 00 00 00-00 00 00 00 00 00 00 00  4...............
0CC0:0130  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0CC0:0140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0CC0:0150  00 00 00 00 00 00 00 00-34 00 00 00 00 00 00 00  ........4.......
-
```

Первая команда MOV BP,138 приводит к результату: BP= 0138

Вторая команда ADD [BP+20],AL приводит к результату: адрес= 0138 + 20 = 0158. Далее, [0158]= [0158] + AL = 0 + 34= 34

Третья команда ADD [BP-20],AX приводит к результату: адрес= 0138 - 20 = 0118. Далее, [0118]= [0118] + AL = 0 + 34= 34 и [0119]= [0119] + AH = 0 + 12= 12

Задания к лабораторной работе:

<u>Выполнить следующие задания</u>:

1. **Изучить содержание лабораторной работы (в часы самоподготовки).**
2. **Повторить выполнение разобранных в работе примеров (в часы самоподготовки).**
3. **Законспектировать примеры применения команд (в часы самоподготовки).**
4. **Выполнить следующие задания и прокомментировать результаты их выполнения:**

```
MOV BP,0140

MOV AX,2030

MOV [BP-20],AL

MOV [BP-21],AX

ADD BX,[BP-20]

ADD [BP-21],BX
```

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА:

1. Абель П. Язык ассемблера для IBM PC и программирования. - М.: Высшая школа, 1992.- 447 с.

2. Лямин Л.В. Макроассемблер MASM. - М.: Радио и связь, 1994.- 320 с.

3. Нортон П., Соухэ Д. Язык ассемблера для IBM PC. – М.: "Компьютер", Финансы и статистика, 1992. – 352 с.

4. Сван Т. Освоение Turbo Assembler. – Киев: "Диалектика", 1996.

5. Юров В. Assembler: учебный курс. – СПб: Питер, 1998.

6. Юров В., Хорошенко С. Assembler : учебный курс.-Санкт-Петербург: Питер, 1999. -672

7. Юров В. Assembler: Специальный справочник. – СПб: Питер, 2000.

8. Юров В. ASSEMBLER : практикум. (с дискетой).-Санкт-Петербург: Питер, 2002. -400

9. Юров В. Assembler: практикум – СПб: Питер, 2003.- 400 с.

10. Пирогов В.Ю. Ассемблер на примерах. - СПб.: БХВ-Петербург, 2005. - 416 с.

11. Пирогов В.Ю. Ассемблер для Windows. - 3-е изд. - СПб.: БХВ-Петербург, 2005. - 864 с.

12. Шнайдер О. Язык ассемблера для персонального компьютера фирмы IBM. Пер. с англ. под ред. Е.К.Масловского.-Москва: Мир, 1988. -405С.

13. Юров В.И. Assembler. Учебник для вузов. 2-ое изд. – СПб: Питер, 2005. - 637 с.

14. Использование Turbo Assembler при разработке программ. – Киев: "Диалектика", 1994. - 288 с.