

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

Преподаватель

Старший преподаватель

Д.В. Куртяник

Отчет

по лабораторной работе №1

по дисциплине Программирование на языках Ассемблера

на тему: «Изучение среды программирования MASM»

Работу выполнил

студент гр. 4141

В.С. Сыворотнев

Санкт-Петербург
2022

Цель лабораторной работы: ознакомиться со средой разработки и ее функциями. Изучить основные компоненты ассемблерной программы. Научиться компилировать программу. Изучить способы описания констант и идентификаторов. Установить различия между командами и директивами ассемблера. Изучить механизмы выделения памяти для переменных. Рассмотреть основные компоненты 16-битного приложения. Познакомиться с основными элементами отладчика Turbo Debugger.

Ответы на вопросы.

1. Почему в последних версиях Windows нет возможности напрямую создавать 16-битные приложения?

Windows 7 и выше не поддерживает работу с 16-битными приложениями как с устаревшим форматом; их можно запускать только с помощью DOS-эмуляторов или виртуальных машинах с предыдущими версиями Windows.

2. Сколько систем счисления доступно в MASM?

Всего в MASM доступно 4 СС: двоичная, восьмеричная, десятичная и шестнадцатеричная.

3. Перечислите директивы для описания типов переменных. Чем отличаются знаковые типы от беззнаковых?

Директивы для описания типов переменных: BYTE, SBYTE, WORD, SWORD, DWORD, SDWORD, FWORD, QWORD, TBYTE, REAL4, REAL8, REAL10. Знаковые и беззнаковые переменные различаются диапазоном возможных значений:

- $0..2^k - 1$ для беззнаковых чисел;
- $-2^{(k - 1)}..2^{(k - 1)} - 1$ для знаковых чисел,

где k – разрядность типа (в байтах).

4. Почему директивы не отображаются при дизассемблировании программы?

При компиляции директивы не отображаются в качестве команд как таковых, поэтому при дизассемблировании программы они не отображаются.

5. Как можно использовать оператор dup при создании массива?

Его можно использовать во время инициализации массива, например, так:

Mass DWORD 10 DUP (0)

В Mass будет находиться 10 нулей.

6. В чем различия между инициализированными и неинициализированными переменными? Для чего используется директива «.data?»?

Инициализированные переменные имеют в себе значение с самого начала их времени жизни, чего нельзя сказать о неинициализированных. Неинициализированные переменные рекомендуется описывать в секции «.data?», для уменьшения размера конечного файла, потому что при создании больших неинициализированных массивов будет занимать большое количество памяти.

7. Что представляет собой счетчик команд и чем он полезен в программе? Оператор счетчика команд возвращает смещение текущего оператора относительно начала сегмента. Он помогает узнать количество элементов массива.

8. Опишите различия между директивой = и директивой EQU.

Директива присваивания «=» связывает метку с целочисленным выражением. В процессе метку можно неоднократно изменять (присваивать иное значение). Директива EQU используется для назначения символического имени целочисленному выражению или произвольной текстовой строке. Главным отличием данных схожих директив является то, что директива EQU не допускает переопределения метки новым значением.

9. С какими проблемами могут столкнуться разработчики 16-битных приложений, использующие последние версии Windows? Как их решить?

Последние версии ОС Windows имеют разрядность 64 или 32 бит, и не позволяют понижаться до 16 бит. Чтобы разрабатывать 16-битные приложения, им необходимо будет воспользоваться DOS-эмуляторами или виртуальными машинами с предыдущими версиями Windows.

10. Перечислите ключевые директивы и команды консольного приложения?

Для работы с современным процессором Amd/Intel 8086 используется директива .8086; директива .model small позволяет сделать доступными для программиста идентификаторы, в которых хранится информация о физических адресах сегментов; директива .code позволяет обозначить местонахождение кода консольного приложения; для завершения процесса исполнения программы используются строки mov ah, 4ch и int 21h.

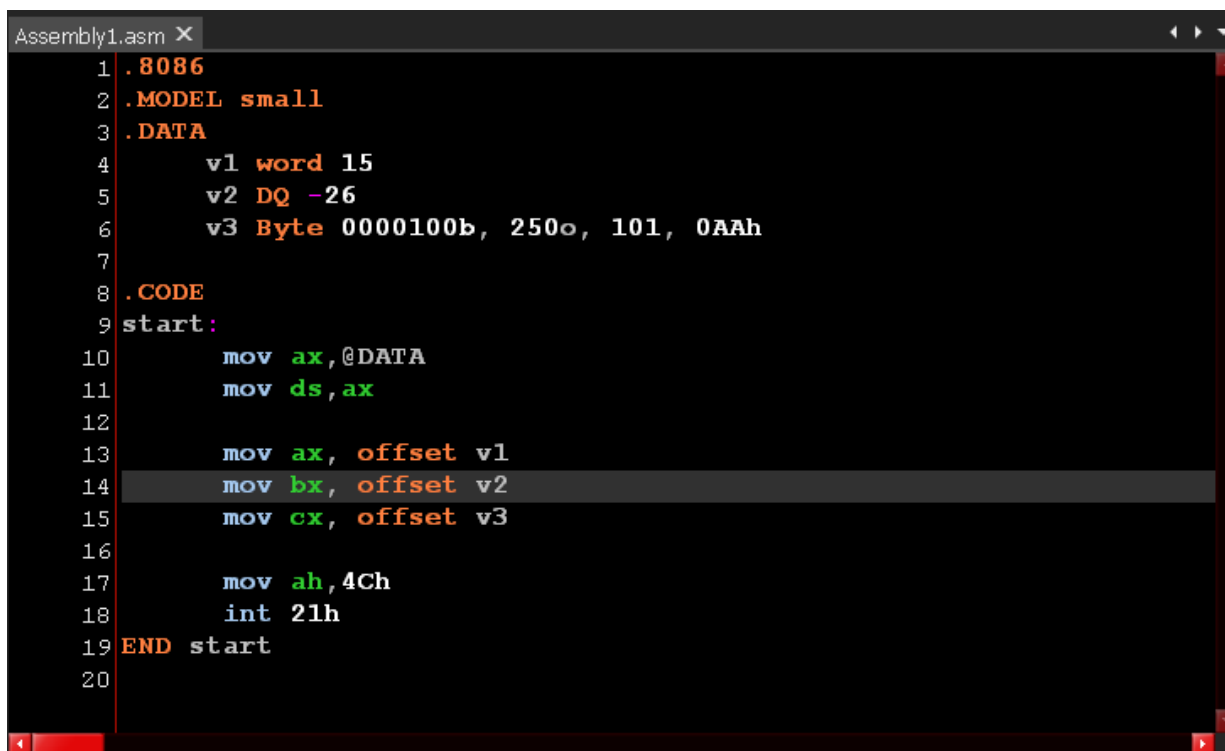
11. Какие элементы программы позволяет отслеживать отладчик Turbo Debugger?

TD позволяет отслеживать:

1. Адресное пространство программного кода. За сегмент кода отвечает регистр CS.
2. Опкоды инструкций в оперативной памяти.
3. Инструкции процессору.
4. Значение регистров в текущий момент времени.
5. Значение бит у регистра флагов.
6. Дамп памяти. Левая колонка задает адрес. Центральная выводит содержимое байт памяти (в строке по 8 байт). Правая содержит символы опкода с точки зрения ASCII представления.
7. Содержимое стека.

12. Напишите программу: опишите переменную размером в слово и переменную размером в учетверенное слово со знаком. Опишите и инициализируйте значениями массив типа байт из четырех чисел, где каждое число представлено в двоичной, восьмеричной, десятичной и шестнадцатеричной системах счисления соответственно. На снимке экрана отладчика выделите соответствующие ячейки памяти.

Код программы:



```
Assembly1.asm X
1  .8086
2  .MODEL small
3  .DATA
4      v1 word 15
5      v2 DQ -26
6      v3 Byte 0000100b, 250o, 101, 0AAh
7
8  .CODE
9  start:
10     mov ax, @DATA
11     mov ds, ax
12
13     mov ax, offset v1
14     mov bx, offset v2
15     mov cx, offset v3
16
17     mov ah, 4Ch
18     int 21h
19 END start
20
```

Отладка в Turbo debugger:

```

E File Edit View Run Breakpoints Data Options Window Help
[ ]=CPU P?ntium Pro=
cs:0000 B82618 mov ax,1826
cs:0003 8ED8 mov ds,ax
cs:0005 B80200 mov ax,0002
cs:0008 BB0400 mov bx,0004
cs:000B B90C00 mov cx,000C
cs:000E B44C mov ah,4C
cs:0010 CD21 int 21
cs:0012 0F00E6 verr si
cs:0015 FF db FF
cs:0016 FF db FF
cs:0017 FF db FF
cs:0018 FF db FF
cs:0019 FF db FF
cs:001A FF db FF
cs:001B FF04 inc word ptr [si]

ax 0002 c=0
bx 0004 z=0
cx 000C s=0
dx 0000 o=0
si 0000 p=0
di 0000 a=0
bp 0000 i=1
sp 0000 d=0
ds 1826
es 1815
ss 1825
cs 1825
ip 000E

ds:0000 CD 21 0F 00 E6 FF FF FF =!* ц
ds:0008 FF FF FF FF 04 A8 65 AA  иек
ds:0010 68 BB 27 E8 6D 07 59 8B  hл'wm=Ул
ds:0018 E5 07 1F 5F 5E 5D 5B 5B  x=V ^lll
ds:0020 5A 59 58 C3 50 51 52 53  ZYX PQRS

ss:0008 04BB
ss:0006 0002
ss:0004 B8D8
ss:0002 8E18
ss:0000 26B8

```

Ячейка памяти переменной v1:

```

C:\ Turbo Debugger for DOS
E File Edit View Run Breakpoints Data Options Window Help
[ ]=CPU P?ntium Pro=
cs:0000 B82618 mov ax,1826
cs:0003 8ED8 mov ds,ax
cs:0005 B80200 mov ax,0002
cs:0008 BB0400 mov bx,0004
cs:000B B90C00 mov cx,000C
cs:000E B44C mov ah,4C
cs:0010 CD21 int 21
cs:0012 0F00E6 verr si
cs:0015 FF db FF
cs:0016 FF db FF
cs:0017 FF db FF
cs:0018 FF db FF
cs:0019 FF db FF
cs:001A FF db FF
cs:001B FF04 inc word ptr [si]

ax 0002 c=0
bx 0004 z=0
cx 000C s=0
dx 0000 o=0
si 0000 p=0
di 0000 a=0
bp 0000 i=1
sp 0000 d=0
ds 1826
es 1815
ss 1825
cs 1825
ip 000E

ds:0000 CD 21 0F 00 E6 FF FF FF =!* ц
ds:0008 FF FF FF FF 04 A8 65 AA  иек
ds:0010 68 BB 27 E8 6D 07 59 8B  hл'wm=Ул
ds:0018 E5 07 1F 5F 5E 5D 5B 5B  x=V ^lll
ds:0020 5A 59 58 C3 50 51 52 53  ZYX PQRS

ss:0008 04BB
ss:0006 0002
ss:0004 B8D8
ss:0002 8E18
ss:0000 26B8

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

```

Ячейка памяти переменной v2:

Turbo Debugger for DOS window showing assembly code and registers. The memory address `ds:0008` is highlighted with a red box, showing the value `04BB`. The registers section on the right shows `ax: 0002`, `bx: 0004`, `cx: 000C`, `dx: 0000`, `si: 0000`, `di: 0000`, `bp: 0000`, `sp: 0000`, `ds: 1826`, `es: 1815`, `ss: 1825`, and `ip: 000E`.

Ячейка памяти переменной v3:

Turbo Debugger for DOS window showing assembly code and registers. The memory address `ds:0008` is highlighted with a red box, showing the value `04A8`. The registers section on the right shows `ax: 0002`, `bx: 0004`, `cx: 000C`, `dx: 0000`, `si: 0000`, `di: 0000`, `bp: 0000`, `sp: 0000`, `ds: 1826`, `es: 1815`, `ss: 1825`, and `ip: 000E`.

13. Определить, сколько байт памяти занимают следующие переменные:

M DWORD 45	4 байта
N WORD 23, 0FFh	4 байта
Arr BYTE "Hello, world!"	13 байт
Mass BYTE 100 DUP (23, "Assembler", 0, 45h)	1400 байт

14. Какие ошибки допущены при описании следующих переменн

Text WORD "Hello, world"	Недопустимо длинная строка
N BYTE 260	Число превышает допустимое значение (255)
M DWORD 24, FFh, 45h	Шестнадцатеричное число не может начинаться с буквы (правильно: 0FFh)
K BYTE -5	Беззнаковый нельзя проинициализировать знаковой константой
Arr WORD DUP (76, 34)	Неверный синтаксис

15. Объяснить, почему в следующем блоке кода метка kol неверно возвращает число элементов массива:

Mass DWORD 34, 567, -230, 0, 123, 2015

kol = (\$ - Mass) / 2

Значение количества элементов массива возвращается неверно, так как тип DWORD содержит 4 байта, мы должны делить разность на размер типа, т.е. на 4, а не на 2.

Верное написание:

kol = (\$ - Mass) / 4

16. Напишите программу: опишите переменную типа WORD, две переменные типа DWORD и переменную типа BYTE. Поместите данные переменные в регистровую память. С помощью отладчика найти эти переменные и указать их адреса. Отметить на снимке экрана соответствующие команды и ячейки памяти.

Код программы:

```
Assembly1.asm X
2  .MODEL small
3  .DATA
4      v1 word 0AAAAh
5      v2 dword 0CBAFh
6      v3 dword 0AFAFAFAh
7      v4 byte 10h
8
9  .CODE
10 start:
11      mov ax, @DATA
12      mov ds, ax
13
14      mov ax, offset v1
15      mov bx, offset v2
16      mov cx, offset v3
17      mov dx, offset v4
18
19      mov ah, 4Ch
20      int 21h
21
22 END start
```


Окно отладчика:

The screenshot shows the Turbo Debugger for DOS interface. The main window displays assembly code for a program. The code is as follows:

```

cs:0000 B82618 mov ax,1826
cs:0003 8ED8 mov ds,ax
cs:0005 B80600 mov ax,0006
cs:0008 B80800 mov bx,0008
cs:000B B90C00 mov cx,000C
cs:000E BA1000 mov dx,0010
cs:0011 B44C mov ah,4C
cs:0013 CD21 int 21
cs:0015 00AAAAAF add [bp+si-5056],ch
cs:0019 CB retf
cs:001A 0000 add [bx+si],al
cs:001C FA cli
cs:001D FA cli
cs:001E FA cli
cs:001F 0A10 or dl,[bx+si]
  
```

On the right side, the register values are displayed:

```

ax 0006 c=0
bx 0008 z=0
cx 000C s=0
dx 0010 o=0
si 0000 p=0
di 0000 a=0
bp 0000 i=1
sp 0000 d=0
ds 1826
es 1815
ss 1825
cs 1825
ip 0011
  
```

At the bottom, the memory dump shows the current state of memory:

```

ds:0000 00 04 4C CD 21 00 AA AA |L=? KK
ds:0008 AF CB 00 00 FA FA FA 0A |...
ds:0010 10 BB 27 E8 6D 07 59 8B |...
ds:0018 E5 07 1F 5F 5E 5D 5B 5B |...
ds:0020 5A 59 58 C3 50 51 52 53 |ZYX PQRS
  
```

The status bar at the bottom shows the following function key shortcuts:

```

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu
  
```

Вывод: я познакомился со средой разработки и ее функциями. Изучил основные компоненты ассемблерной программы. Научился компилировать программу. Изучил способы описания констант и идентификаторов. Установил различия между командами и директивами ассемблера. Изучил механизмы выделения памяти для переменных. Рассмотрел основные компоненты 16-битного приложения. Познакомился с основными элементами отладчика Turbo Debugger.