

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

Преподаватель

Старший преподаватель

Д.В. Куртяник

Отчет

по лабораторной работе №2

по дисциплине Программирование на языках Ассемблера

на тему: «Разработка программы в среде MASM»

Работу выполнил

студент гр. 4141

В.С. Сыворотнев

Санкт-Петербург
2022

Цель лабораторной работы: изучение концепций и освоение технологии программирования 16-битных приложений на языке ассемблера для архитектуры процессоров семейства Intel. Закрепление навыков работы с отладчиком Turbo Debugger.

Вариант №18

18. Задан массив чисел размером в байт. Найти максимальный элемент массива.

Описание выбранной модели решения:

Для начала в сегменте `.Data` был инициализирован массив типа `BYTE`. Для осуществления алгоритма поиска максимального значения в массиве, необходимо рассмотреть все элементы массива, поэтому количество итераций цикла будет непосредственно равняться количеству элементов данного массива. Так как регистр `CX` отвечает за количество итераций цикла в программе, в него было записано число 5 с помощью команды `mov`. После чего в регистр `SI` был записан массив с помощью инструкции `offset`. Далее был обнулен регистр `BX` с помощью команды `mov`. Затем следует метка `@m1`, содержащая циклический алгоритм поиска максимального значения. Сначала идет инструкция `lodsb`, которая копирует 1 байт из памяти по адресу `DS:SI` в регистр `AL`. После чего происходит сравнение значений в регистрах `AL` и `BH`. Если значение в регистре `AL` больше, то далее происходит переход в метку `max` с помощью команды `jb`. В метке `max` происходит запись значения из регистра `AL` в регистр `BH`, после чего наступает следующая итерация цикла. Если же значение в регистре `AL` меньше, то команда `jb` пропускается, программа переходит к выполнению команды `jmp` на метку `@m2`, в которой выполняется команда `loop`, которая за циклирует метку `@m1` до тех пор, пока значение в регистре `CX` не станет равно 0. После окончания выполнения цикла, в регистре `BH` будет находиться значение максимального элемента заданного массива.

Код программы:

```
.8086
.MODEL small

.DATA
    array byte 1, 4, 3, 6, 15
.CODE
start:
    mov AX, @DATA
    mov DS, AX

    mov CX, 5 ; количество итераций( равно количеству элементов
массива)
    mov SI, offset array ; записываем массив в регистр SI

    mov BX, 0

@m1:
    lodsb ; копирование 1 байта из памяти по адресу DS:SI в регистр AL
    cmp AL, BH ; сравнение значений в AL и BH

    jg max ; Если AL > BH, то выполняется переход на метку тах

    jmp @@m2 ; если переход на метку тах не был осуществлен, происходит переход на
метку m2
max:
    mov BH, AL ; сохраняем максимальное значение в регистр bh
@@m2:
    loop @m1 ; переход на метку m1, до тех пор, пока значение в CX != 0 (инструкция
цикла)

    mov AH, 4ch
    int 21h

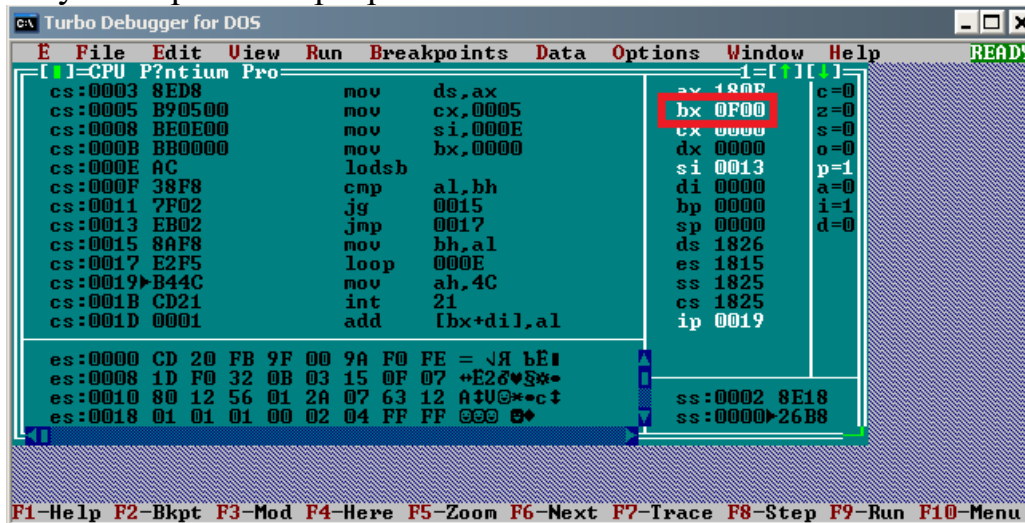
end start
```

Снимки экрана с анализом работы программы:

Исходный массив:

```
.DATA
    array byte 1, 4, 3, 6, 15
.CODE
```

Результат работы программы:



Turbo Debugger for DOS

File Edit View Run Breakpoints Data Options Window Help

1=CPU P?ntium Pro

Address	Disassembly	Comment
cs:0003	8ED8	mov ds,ax
cs:0005	B90500	mov cx,0005
cs:0008	BE0E00	mov si,000E
cs:000B	BB0000	mov bx,0000
cs:000E	AC	lodsb
cs:000F	38F8	cmp al,bh
cs:0011	7F02	jg 0015
cs:0013	EB02	jmp 0017
cs:0015	8AF8	mov bh,al
cs:0017	E2F5	loop 000E
cs:0019	B44C	mov ah,4C
cs:001B	CD21	int 21
cs:001D	0001	add [bx+di],al

Registers:

- ax: 180F
- bx: 0F00**
- cx: 0005
- dx: 0000
- si: 0013
- di: 0000
- bp: 0000
- sp: 0000
- ds: 1826
- es: 1815
- ss: 1825
- cs: 1825
- ip: 0019

Stack:

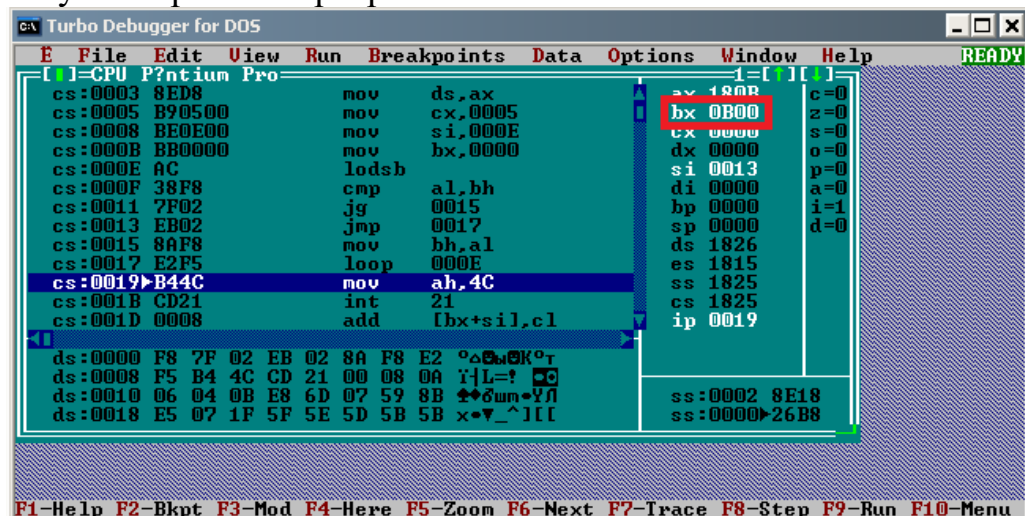
- ss:0002 8E18
- ss:0000 26B8

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

Исходный массив:

```
.DATA
    array byte 8, 10, 6, 4, 11
.CODE
```

Результат работы программы:



Turbo Debugger for DOS

File Edit View Run Breakpoints Data Options Window Help

1=CPU P?ntium Pro

Address	Disassembly	Comment
cs:0003	8ED8	mov ds,ax
cs:0005	B90500	mov cx,0005
cs:0008	BE0E00	mov si,000E
cs:000B	BB0000	mov bx,0000
cs:000E	AC	lodsb
cs:000F	38F8	cmp al,bh
cs:0011	7F02	jg 0015
cs:0013	EB02	jmp 0017
cs:0015	8AF8	mov bh,al
cs:0017	E2F5	loop 000E
cs:0019	B44C	mov ah,4C
cs:001B	CD21	int 21
cs:001D	0008	add [bx+si],cl

Registers:

- ax: 180F
- bx: 0B00**
- cx: 0005
- dx: 0000
- si: 0013
- di: 0000
- bp: 0000
- sp: 0000
- ds: 1826
- es: 1815
- ss: 1825
- cs: 1825
- ip: 0019

Stack:

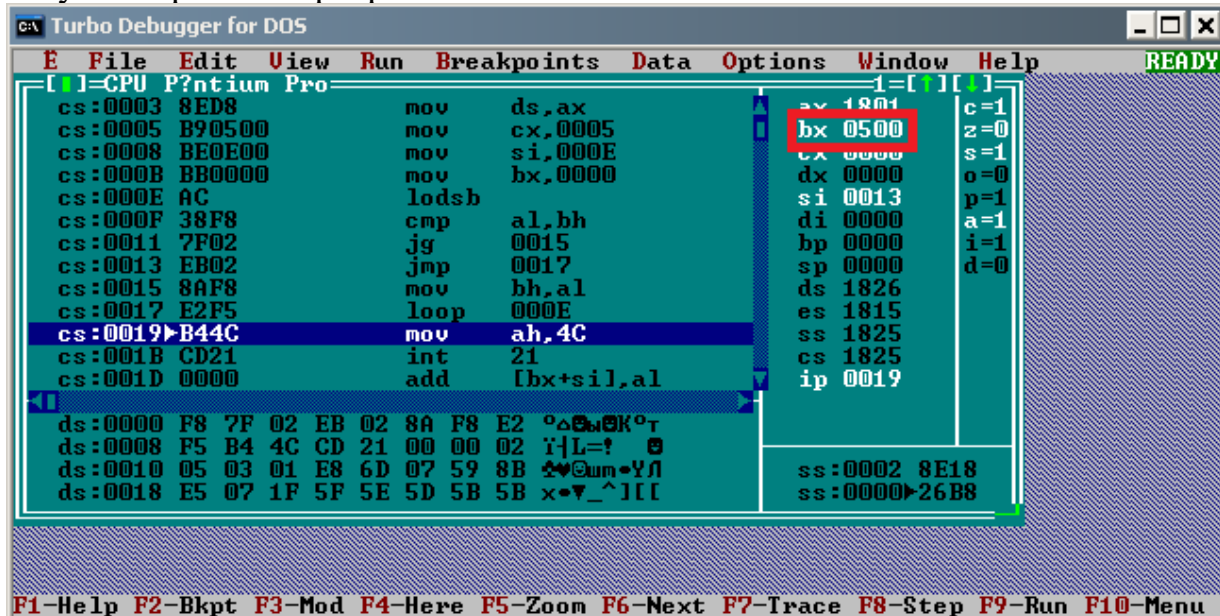
- ss:0002 8E18
- ss:0000 26B8

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

Исходный массив:

```
.DATA
    array byte 0, 2, 5, 3, 1
.CODE
```

Результат работы программы:



The screenshot shows the Turbo Debugger for DOS interface. The main window displays assembly code with the following instructions highlighted:

```
cs:0003 8ED8      mov     ds,ax
cs:0005 B90500     mov     cx,0005
cs:0008 BE0E00     mov     si,000E
cs:000B BB0000     mov     bx,0000
cs:000E AC        lodsb
cs:000F 38F8      cmp     al,bh
cs:0011 7F02      jg      0015
cs:0013 EB02      jmp     0017
cs:0015 8AF8      mov     bh,al
cs:0017 E2F5      loop    000E
cs:0019 B44C      mov     ah,4C
cs:001B CD21      int     21
cs:001D 0000      add     [bx+si],al
```

The register window on the right shows the following values:

ax	1804
bx	0500
cx	0000
dx	0000
si	0013
di	0000
bp	0000
sp	0000
ds	1826
es	1815
ss	1825
cs	1825
ip	0019

The status bar at the bottom shows the following keyboard shortcuts:

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

Вывод: я изучил концепции и освоил технологии программирования 16-битных приложений на языке ассемблера для архитектуры процессоров семейства Intel. Закрепил навыки работы с отладчиком Turbo Debugger.