

Отчет по лабораторной работе №7

Хагуров Андрей Андреевич

Содержание

1	Цель работы	1
2	Выполнение лабораторной работы.....	1
3	Задания для самостоятельной работы	5
4	Выводы	8

1 Цель работы

Изучить основы работы команд условного и безусловного перехода в assembler.

2 Выполнение лабораторной работы

Создадим рабочую папку и рабочий файл (рис. 1)

```
aakhagurov@dk8n59 ~/work/arc-pc $ mkdir lab07
aakhagurov@dk8n59 ~/work/arc-pc $ cd lab07
aakhagurov@dk8n59 ~/work/arc-pc/lab07 $ touch lab07.asm
aakhagurov@dk8n59 ~/work/arc-pc/lab07 $ gedit lab07.asm
```

Запишем в файл код, проасSEMBЛИруем его, запустим (рис. 1)

```
aakhagurov@dk8n59 ~/work/arc-pc/lab07 $ nasm -f elf lab07-1.asm
aakhagurov@dk8n59 ~/work/arc-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
aakhagurov@dk8n59 ~/work/arc-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 3
aakhagurov@dk8n59 ~/work/arc-pc/lab07 $
```

Рис. 1: jmp работает

изменим программу так, чтобы она выводила второе, затем первое сообщение и завершала работу. Её код (рис. 2) и работа (рис. 3)

```

1 | %include 'in_out.asm'
2 |
3 | SECTION .data
4 | msg1: DB 'Сообщение No 1',0
5 | msg2: DB 'Сообщение No 2',0
6 | msg3: DB 'Сообщение No 3',0
7 |
8 | SECTION .text
9 |
10 | GLOBAL _start
11 | _start:
12 | jmp _label2
13 |
14 | _label1:
15 | mov eax,msg1
16 | call sprintLF
17 | jmp _end
18 |
19 | _label2:
20 | mov eax,msg2
21 | call sprintLF
22 | jmp _label1
23 |
24 | _label3:
25 | mov eax,msg3
26 | call sprintLF
27 |
28 | _end:
29 | call quit

```

Рис. 2: Код программы

```

aakhagurov@dk8n59 ~/work/arc-pc/lab07 $ nasm -f elf lab7-1.asm
aakhagurov@dk8n59 ~/work/arc-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
aakhagurov@dk8n59 ~/work/arc-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 1
aakhagurov@dk8n59 ~/work/arc-pc/lab07 $

```

Рис. 3: Её работа

Еще напишем программу, выводящую сообщения в обратном порядке(рис. 4 и рис. 5)

```

1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1: DB 'Сообщение No 1',0
5 msg2: DB 'Сообщение No 2',0
6 msg3: DB 'Сообщение No 3',0
7
8 SECTION .text
9
10 GLOBAL _start
11 _start:
12 jmp _label3
13
14 _label1:
15 mov eax,msg1
16 call sprintLF
17 jmp _end
18
19 _label2:
20 mov eax,msg2
21 call sprintLF
22 jmp _label1
23
24 _label3:
25 mov eax,msg3
26 call sprintLF
27 jmp _label2
28
29 _end:
30 call quit

```

Рис. 4: Код программы

```

aakhagurov@dk8n59 ~/work/arc-pc/lab07 $ nasm -f elf lab7-1.asm
aakhagurov@dk8n59 ~/work/arc-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
aakhagurov@dk8n59 ~/work/arc-pc/lab07 $ ./lab7-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
aakhagurov@dk8n59 ~/work/arc-pc/lab07 $

```

Рис. 5: Её работа

Напишем программу с условным переходом(рис. 6)

```

aakhagurov@dk8n59 ~/work/arc-pc/lab07 $ ./lab7-2
Введите B: 100
Наибольшее число: 100
aakhagurov@dk8n59 ~/work/arc-pc/lab07 $ ./lab7-2
Введите B: 1
Наибольшее число: 50
aakhagurov@dk8n59 ~/work/arc-pc/lab07 $

```

Рис. 6: Работа программы

Рассмотрим файл листинга одной из программ(рис. 7:008?)

```

1      %include 'in_out.asm'
2      ;----- slen -----
3      <1> ; Функция вычисления длины сообщения
4      <1> slen:
5      00000000 53          <1>      push     ebx
6      00000001 89C3       <1>      mov      ebx, eax
7
8      <1> nextchar:
9      00000003 803800     <1>      cmp      byte [eax], 0
10     00000006 7403       <1>      jz       finished
11     00000008 40         <1>      inc      eax
12     00000009 EBF8       <1>      jmp      nextchar
13
14     <1> finished:
15     0000000B 29D8       <1>      sub      eax, ebx
16     0000000D 5B         <1>      pop      ebx
17     0000000E C3         <1>      ret
18
19
20     <1> ;----- sprint -----
21     <1> ; Функция печати сообщения
22     <1> ; входные данные: mov eax,<message>
23     <1> sprint:
24     0000000F 52         <1>      push     edx
25     00000010 51         <1>      push     ecx
26     00000011 53         <1>      push     ebx
27     00000012 50         <1>      push     eax
28     00000013 E8E8FFFF   <1>      call    slen
29
30     00000018 89C2       <1>      mov      edx, eax
31     0000001A 58         <1>      pop      eax
32
33     0000001B 89C1       <1>      mov      ecx, eax
34     0000001D BB01000000 <1>      mov      ebx, 1

```

в строке 9 содержится собственно номер сторки [9], адресс [00000003], машинный код [803800] и содержимое строки кода [cmp byte [eax], 0] в строке 11 содержится номер сторки [11], адресс [00000008], машинный код [40] и содержимое строки кода [inc eax] в строке 24 содержится номер сторки [24], адресс [0000000F], машинный код [52] и содержимое строки кода [push edx]

Если в коде появляется ошибка, то ее описание появится в файле листинга(рис. 7)

```

9      section .bss
10     00000000 <res Ah>    max resb 10
11     0000000A <res Ah>    B resb 10
12
13     section .text
14
15     global _start
16     _start:
17     mov eax
17     *****
18     000000E8 E822FFFF    call sprint
19
20     000000ED B9[0A000000]  mov ecx, B
21     000000F2 BA0A000000  mov edx,10
22     000000F7 E847FFFF    call sread
23
24     000000FC B8[0A000000]  mov eax, B
25     00000101 E896FFFF    call atoi
26     00000106 A3[0A000000]  mov [B], eax
27
28     0000010B 8B0D[35000000]  mov ecx, [A]

```

Рис. 7: ошибка в файле листинга

3 Задания для самостоятельной работы

(Вар 17)

программа для сравнения трех заранее известных чисел(рис. **fig:010?**) и ее работа(рис. **fig:011?**)

```

1 %include 'in_out.asm'
2 section .data
3 msg2 db "Наибольшее число: ",0h
4 A dd '26'
5 B dd '12'
6 C dd '68'
7
8 section .bss
9 max resb 10
10
11 section .text
12
13 global _start
14 _start:
15 mov eax,B
16 call atoi ; Вызов подпрограммы перевода символа в число
17 mov [B],eax ; запись преобразованного числа в 'B'
18 ; ----- Записываем 'A' в переменную 'max'
19 mov ecx,[A] ; 'ecx = A'
20 mov [max],ecx ; 'max = A'
21 ; ----- Сравниваем 'A' и 'C' (как символы)
22 cmp ecx,[C] ; Сравниваем 'A' и 'C'
23 jg check_B ; если 'A>C', то переход на метку 'check_B',
24 mov ecx,[C] ; иначе 'ecx = C'
25 mov [max],ecx ; 'max = C'
26
27 ; ----- Преобразование 'max(A,C)' из символа в число
28 check_B:
29 mov eax,max
30 call atoi ; Вызов подпрограммы перевода символа в число
31 mov [max],eax ; запись преобразованного числа в 'max'
32 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
33 mov ecx,[max]
34 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
35 jg fin ; если 'max(A,C)>B', то переход на 'fin',
36 mov ecx,[B] ; иначе 'ecx = B'
37
38 mov [max],ecx
39
40 ; ----- Вывод результата
41 fin:
42 mov eax, msg2
43 call sprint ; Вывод сообщения 'Наибольшее число: '
44 mov eax,[max]
45 call iprintLF ; Вывод 'max(A,B,C)'

```

```

aakhagurov@dk8n59 ~/work/arc-pc/lab07 $ gedit lab07-3.asm
aakhagurov@dk8n59 ~/work/arc-pc/lab07 $ nasm -f elf lab07-3.asm && ld -m elf_i386
-o lab07-3 lab07-3.o && ./lab07-3
Наибольшее число: 68
aakhagurov@dk8n59 ~/work/arc-pc/lab07 $

```

Программа для вычисления выражения в зависимости от условия на одну из вводимых переменных(рис. [fig:012?](#)) и ее работа(рис. [fig:013?](#))

```

1 %include 'in_out.asm'
2
3 SECTION .data
4 input1 db "Введите x: ",0h
5 input2 db "Введите a: ",0h
6
7 SECTION .bss
8 max resb 10
9 x resb 10
10 a resb 10
11
12 SECTION .text
13 GLOBAL _start
14
15 _start:
16 mov eax,input1
17 call sprint
18
19 mov ecx,x
20 mov edx,10
21 call sread
22
23 mov eax,x
24 call atoi
25 mov [x],eax
26
27 mov eax,input2
28 call sprint
29
30 mov ecx,a
31 mov edx,10
32 call sread
33
34 mov eax,a
35 call atoi
36 mov [a],eax
37
38 mov ebx, 8
39 cmp [a], ebx
40 jge check
41
42 mov eax, [a]
43 mov ebx, 8
44 add eax, ebx
45 call iprintLF

```

```

9aakhagurov@dk8n59 ~/work/arc-pc/lab07 $ nasm -f elf lab07-4.asm && ld -m elf_i386 -o lab07-4 lab07-4.o && ./lab07-4
Введите x: 3
Введите a: 4
12
aakhagurov@dk8n59 ~/work/arc-pc/lab07 $ nasm -f elf lab07-4.asm && ld -m elf_386 -o lab07-4 lab07-4.o && ./lab07-4
2Введите x:
Введите a: 9
18

```

4 Выводы

Были изучены основные принципы работы с условным и безусловным переходом в assembler и изучены основы чтения файлов листинга.