

Лабораторная работа №7

**Команды безусловного и условного переходов в Nasm.
Программирование ветвлений**

Прозорова Елизавета Евгеньевна

Содержание

1	Цель работы	3
2	Выполнение лабораторной работы	4
3	Выполнение самостоятельной работы	11
4	Выводы	15

1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

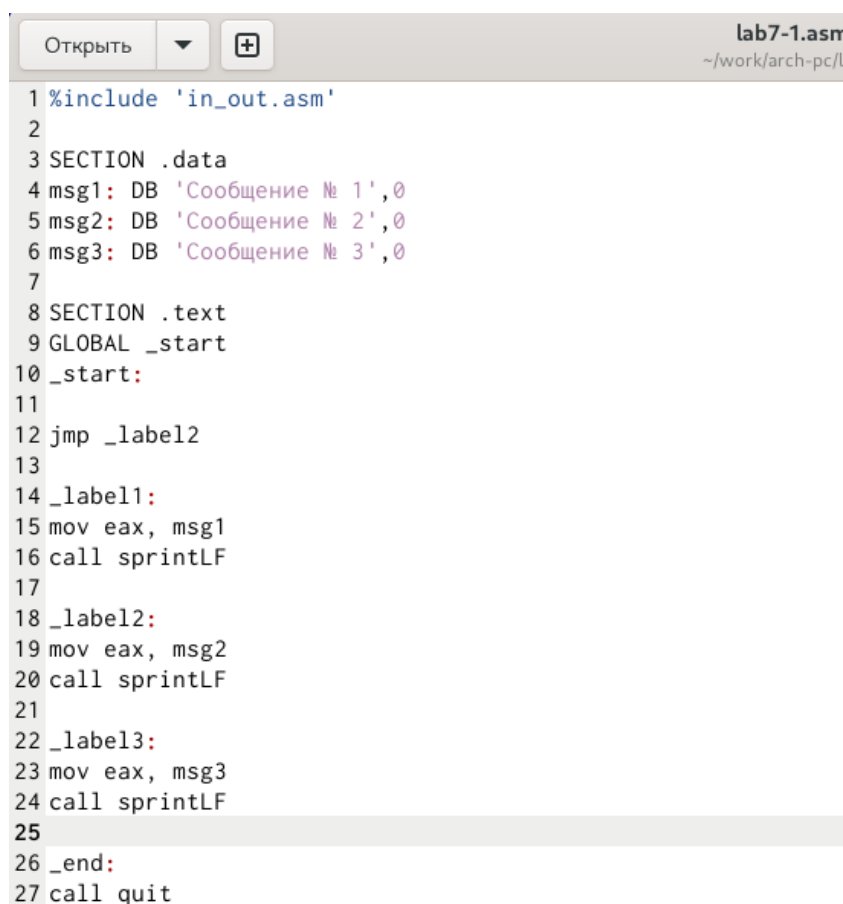
2 Выполнение лабораторной работы

1. Сначала я создала каталог для программ лабораторной работы № 7, затем перешла в него и создала файл lab7-1.asm

```
eeprozorova@dk8n60 ~ $ mkdir ~/work/arch-pc/lab07
eeprozorova@dk8n60 ~ $ cd ~/work/arch-pc/lab07
eeprozorova@dk8n60 ~/work/arch-pc/lab07 $ touch lab7-1.asm
```

Рис. 2.1: Создание каталога и файла lab7-1

2. Я ввела в файл lab7-1.asm текст программы из листинга 7.1.



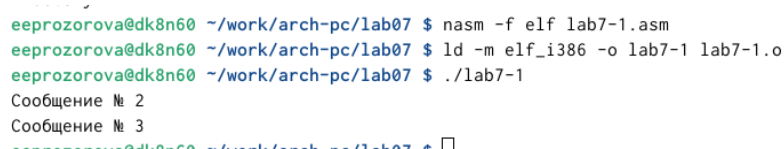
```

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

```

Рис. 2.2: Текст программы lab7-1

Я создала исполняемый файл и запустила его



```

eeprozorova@dk8n60 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
eeprozorova@dk8n60 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
eeprozorova@dk8n60 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3

```

Рис. 2.3: Создание и запуск lab7-1

Я изменила программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу в соответствии с листингом 7.2.

```

1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1: DB 'Сообщение № 1',0
5 msg2: DB 'Сообщение № 2',0
6 msg3: DB 'Сообщение № 3',0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
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. 2.4: Изменения текста

Затем я создала и проверила измененный файл.

```

eeprozorova@dk8n60 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
eeprozorova@dk8n60 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
eeprozorova@dk8n60 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 1

```

Рис. 2.5: Создание и запуск lab7-1

Теперь изменим текст программы изменив инструкции `jmp`, чтобы при выводе программы была такая последовательность сообщений: №3, №2, №1ю

```

8 SECTION .text
9 GLOBAL _start
10 _start:
11
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. 2.6: Измененный текст программы

Затем я создала и проверила измененный файл.

```

eeprozorova@dk8n60 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
eeprozorova@dk8n60 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
eeprozorova@dk8n60 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1

```

Рис. 2.7: Проверка программы

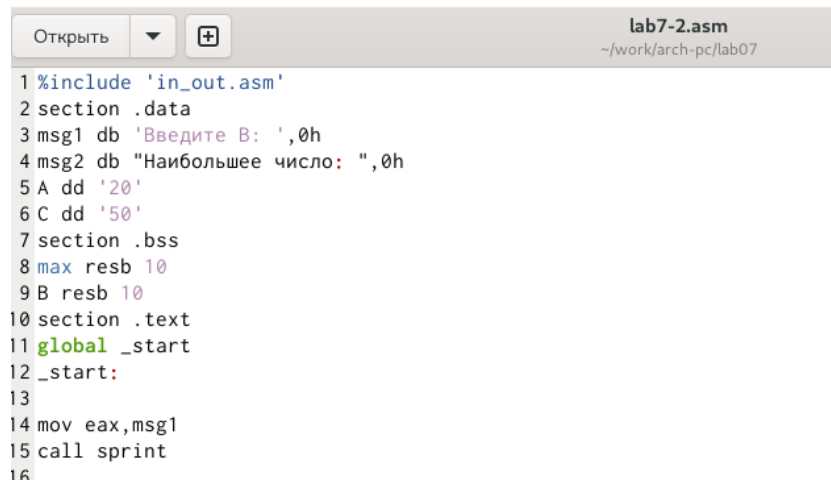
3. Я создала файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 и ввела в него текст программы из листинга 7.3.

```

eeprozorova@dk8n60 ~/work/arch-pc/lab07 $ touch lab7-2.asm

```

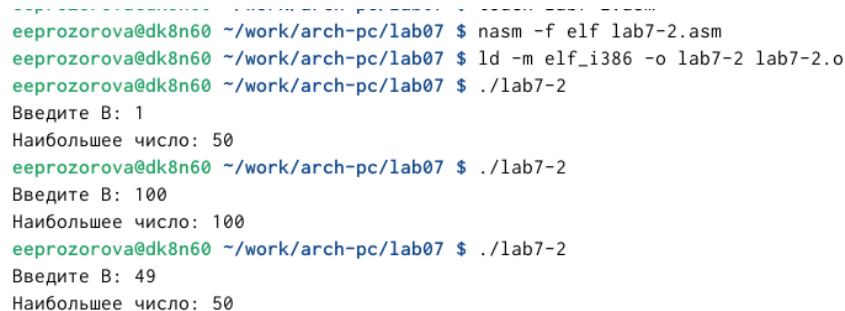
Рис. 2.8: Создание lab7-2



```
1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите B: ',0h
4 msg2 db "Наибольшее число: ",0h
5 A dd '20'
6 C dd '50'
7 section .bss
8 max resb 10
9 B resb 10
10 section .text
11 global _start
12 _start:
13
14 mov eax,msg1
15 call sprint
16
```

Рис. 2.9: Текст программы lab7-2.asm

Затем я создала и проверила работу файла для 1, 100, 49.



```
eeprozorova@dk8n60 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
eeprozorova@dk8n60 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
eeprozorova@dk8n60 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 1
Наибольшее число: 50
eeprozorova@dk8n60 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 100
Наибольшее число: 100
eeprozorova@dk8n60 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 49
Наибольшее число: 50
```

Рис. 2.10: Проверка работы файла

4. Создала файл листинга для программы из файла lab7-2.as



```
eeprozorova@dk8n60 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
```

Рис. 2.11: Создание файла листинга

Затем я открыла файл с помощью mcedit.


```

lab7-2.lst      [----]  0 L: 1+ 0 1/225] *(0 /13280b) 0032 0x020 [*][X
1               %include 'in_out.asm'
1               <1> ;----- slen -----
2               <1> ; Функция вычисления длины сообщения
3               <1> slen:.....
4 00000000 53    <1>      push    ebx.....
5 00000001 89C3  <1>      mov     ebx, eax.....
6               <1>.....
7               <1> nextchar:.....
8 00000003 803800 <1>      cmp     byte [eax], 0...
9 00000006 7403   <1>      jz      finished.....
10 00000008 40    <1>      inc     eax.....
11 00000009 EBF8  <1>      jmp     nextchar.....
12               <1>.....
13               <1> finished:
14 0000000B 29D8  <1>      sub     eax, ebx
15 0000000D 5B    <1>      pop     ebx.....
16 0000000E C3    <1>      ret.....
17               <1>.
18               <1>.

```

Рис. 2.12: Текст файла

Я подробно изучила содержимое файла и выбрала следующие строки для объяснения.

```

13.....
14 000000E8 B8[00000000]      mov eax,msg1

```

Рис. 2.13: 14 строка

1)

Эта строка находится на 14 месте, 000000E8 - ее адрес, B8[00000000] - машинный код. mov eax,msg1 - это исходный текст программы, означающий что в регистр eax вносится значение msg1, в нашем случае это строка со словами “Введите В:”

```

29 00000122 7F0C      jg check_B.

```

Рис. 2.14: 29 строка

2)

Эта строка находится на 29 месте, 00000122 - ее адрес, 7F0C - машинный код. `jpg check_B` - это исходный текст программы, означающий что нужно сделать переход на строку `check-B` если первое число больше второго. Сравнение и определение этих чисел происходят в предыдущих командах.

3) `39 00000145 3B0D[0A000000] cmp ecx,[B].`

Эта строка находится на 39 месте, 00000145 - ее адрес, `3B0D[0A000000]` - машинный код. `cmp ecx,[B]` - это исходный текст программы, означающий что происходит сравнение регистра `ecx` с значением `B`.

Затем я снова открыла с программой `lab7-2.asm` и удалила один операнд в инструкции с двумя операндами.

```

16
17 mov ecx,
18 mov edx,10

```

Рис. 2.15: Строка в которой я удалила операнд

Затем я выполнила трансляцию с получением файла листинга. В файле произошли изменения, которые указывали что на строке с удаленным операндом присутствует ошибка.

```

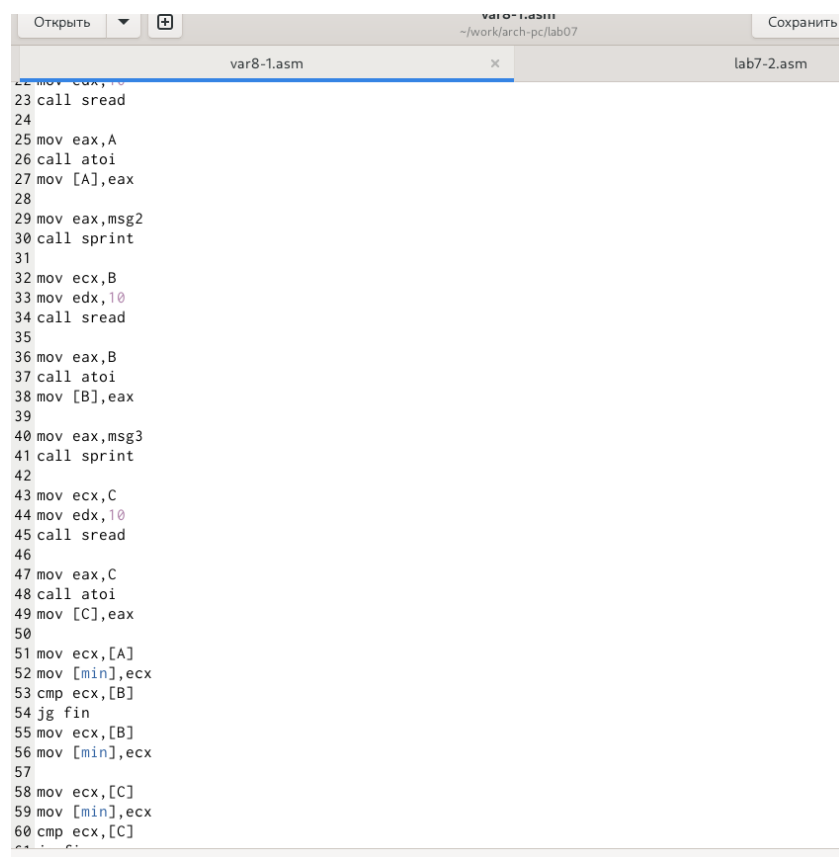
16 .....
17 ..... mov ecx,
17 ..... ***** error: invalid combination of opcode and operands
18 000000F2 BA0A000000 mov edx,10
19 000000F7 E847FFFFFF call sread
20 .....

```

Рис. 2.16: Строка в которой я удалила операнд

3 Выполнение самостоятельной работы

1. Я создала файл var-1, в котором написала программу нахождения наименьшей из 3 целочисленных переменных `a`, `b` и `c`. В моей программе значения переменных вводятся вручную с клавиатуры. Мой вариант - 8.



```
22 mov ecx, 10
23 call sread
24
25 mov eax, A
26 call atoi
27 mov [A], eax
28
29 mov eax, msg2
30 call sprint
31
32 mov ecx, B
33 mov edx, 10
34 call sread
35
36 mov eax, B
37 call atoi
38 mov [B], eax
39
40 mov eax, msg3
41 call sprint
42
43 mov ecx, C
44 mov edx, 10
45 call sread
46
47 mov eax, C
48 call atoi
49 mov [C], eax
50
51 mov ecx, [A]
52 mov [min], ecx
53 cmp ecx, [B]
54 jg fin
55 mov ecx, [B]
56 mov [min], ecx
57
58 mov ecx, [C]
59 mov [min], ecx
60 cmp ecx, [C]
61 jg fin
62 mov [min], ecx
63
64 mov eax, msg4
65 call sprint
66
67 mov ecx, 10
68 call sread
69
70 mov eax, msg5
71 call sprint
72
73 mov ecx, 10
74 call sread
75
76 mov eax, msg6
77 call sprint
78
79 mov ecx, 10
80 call sread
81
82 mov eax, msg7
83 call sprint
84
85 mov ecx, 10
86 call sread
87
88 mov eax, msg8
89 call sprint
90
91 mov ecx, 10
92 call sread
93
94 mov eax, msg9
95 call sprint
96
97 mov ecx, 10
98 call sread
99
100 mov eax, msg10
101 call sprint
```

Рис. 3.1: Текст программы

Затем я проверила работу команды. Мой вариант - 8, поэтому я ввела значения 52,33,40.

```

eeprozorova@dk8n60 ~/work/arch-pc/lab07 $ nasm -f elf var8-1.asm
eeprozorova@dk8n60 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o var8-1 var8-1.o
eeprozorova@dk8n60 ~/work/arch-pc/lab07 $ ./var8-1
Введите A: 52
Введите B: 33
Введите C: 40
Наименьшее число: 33

```

Рис. 3.2: Проверка работы программы

2. Я создала файл var8-2, в который написала программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений.

$$\mathbf{8} \quad \begin{cases} 3a, & a < 3 \\ x + 1, & a \geq 3 \\ a + x, & x < a \end{cases} \quad (1;4) \quad (1;2)$$

Рис. 3.3: Функция для моего варианта

```
20 mov edx,10
21 call sread
22
23 mov eax,a
24 call atoi
25 mov [a],eax
26
27 mov eax,msg2
28 call sprint
29
30 mov ecx,x
31 mov edx,10
32 call sread
33
34 mov eax,x
35 call atoi
36 mov [x],eax
37
38 mov eax,[a]
39 cmp eax, 3
40 jml less_1
41 jge more_2
42
43 less_1:
44 mov eax,[a]
45 mov ebx,3
46 mul ebx
47 mov edi,eax
48 jmp end_
49
50 more_2:
51 mov eax,[x]
52 add eax,1
53 mov edi,eax
54
55 end_:
56 mov eax,rem
57 call sprint
58 mov eax,edi
59 call iprintLF
60
61 call quit
```

Рис. 3.4: Текст программы для моей функции

Затем я создала и проверила работу файла. Ввела такие значения (\square_1, \square_1)=(1;4), (\square_2, \square_2)=(1;2)

```
eeprozorova@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf var8-2.asm
eeprozorova@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o var8-2 var8-2.o
eeprozorova@dk3n55 ~/work/arch-pc/lab07 $ ./var8-2
Введите a: 4
Введите x: 1
2
eeprozorova@dk3n55 ~/work/arch-pc/lab07 $ ./var8-2
Введите a: 2
Введите x: 1
6
```

Рис. 3.5: Проверка работы файла

4 Выводы

В ходе лабораторной работы мной были изучены команды условного и безусловного переходов, навыки написания программ с их использованием. А также я познакомилась с назначением и структурой файла листинга.