

Отчёт по лабораторной работе 7

Архитектура компьютеров

Исмаил М. А. НКАбд-03-24

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Изучение структуры файла листинга	11
2.3	Самостоятельное задание	13
3	Выводы	17

Список иллюстраций

Список таблиц

1 Цель работы

Целью работы является изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

2.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы № 7 и файл lab7-1.asm.



(рис.

)

Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. Написал в файл `lab7-1.asm` текст программы из листинга 7.1.

```

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

```

(рис.

)

Создаю исполняемый файл и запускаю его.

```

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

```

(рис.

)

Инструкция `jmp` позволяет осуществлять переходы не только вперед, но и назад. Изменяю программу таким образом, чтобы она выводила сначала «Сообщение № 2», затем «Сообщение № 1», и завершала работу. Для этого после

вывода сообщения № 2 добавляю инструкцию `jmp` с меткой `_label1` (переход к инструкциям вывода сообщения № 1), и после вывода сообщения № 1 добавляю инструкцию `jmp` с меткой `_end` (переход к инструкции `call quit`).

Изменяю текст программы в соответствии с листингом 7.2.

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

(рис.

```
mismail@VirtualBox:~/work/arch-pc/lab07$
mismail@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
mismail@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
mismail@VirtualBox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
mismail@VirtualBox:~/work/arch-pc/lab07$
```

(рис.

После изменений программа выводит следующее: Сообщение № 3 Сообщение

№ 2 Сообщение № 1

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

(рис.

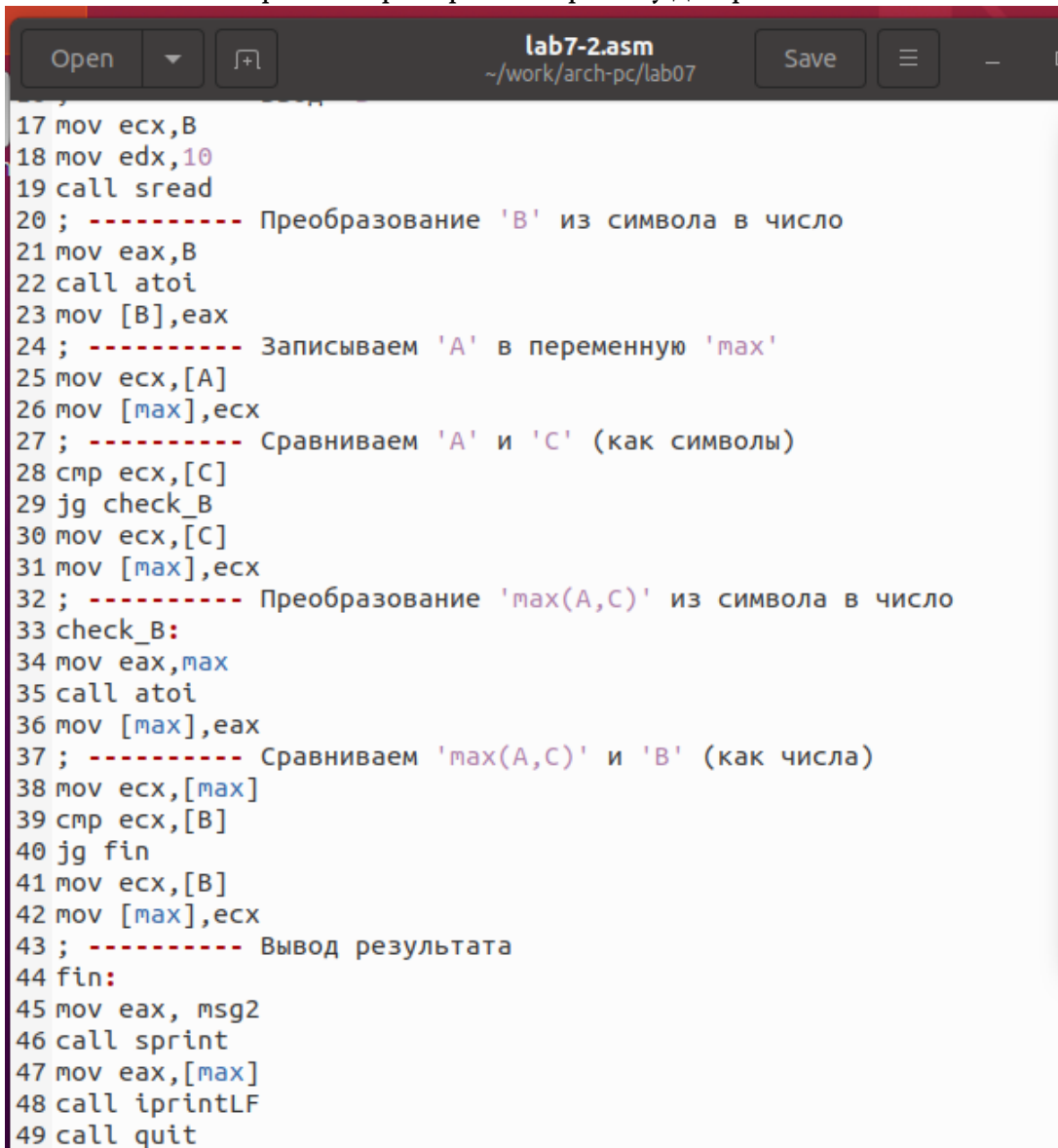
```
mismail@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
mismail@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
mismail@VirtualBox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
mismail@VirtualBox:~/work/arch-pc/lab07$
```

(рис.

Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, то есть переход должен осуществляться только при выполнении определенного

условия. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшее из трех целочисленных переменных: A, B и C. Значения для A и C задаются в программе, значение B вводится с клавиатуры.

Создаю исполняемый файл и проверяю его работу для различных значений



```
lab7-2.asm
~/work/arch-pc/lab07

17 mov ecx,B
18 mov edx,10
19 call sread
20 ; ----- Преобразование 'B' из символа в число
21 mov eax,B
22 call atoi
23 mov [B],eax
24 ; ----- Записываем 'A' в переменную 'max'
25 mov ecx,[A]
26 mov [max],ecx
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx,[C]
29 jg check_B
30 mov ecx,[C]
31 mov [max],ecx
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 mov eax,max
35 call atoi
36 mov [max],eax
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 mov ecx,[max]
39 cmp ecx,[B]
40 jg fin
41 mov ecx,[B]
42 mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 mov eax, msg2
46 call sprintf
47 mov eax,[max]
48 call iprintLF
49 call quit
```

(рис.)

```

mismail@VirtualBox:~/work/arch-pc/lab07$
mismail@VirtualBox:~/work/arch-pc/lab07$
mismail@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
mismail@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
mismail@VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 5
Наибольшее число: 50
mismail@VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 55
Наибольшее число: 55
mismail@VirtualBox:~/work/arch-pc/lab07$

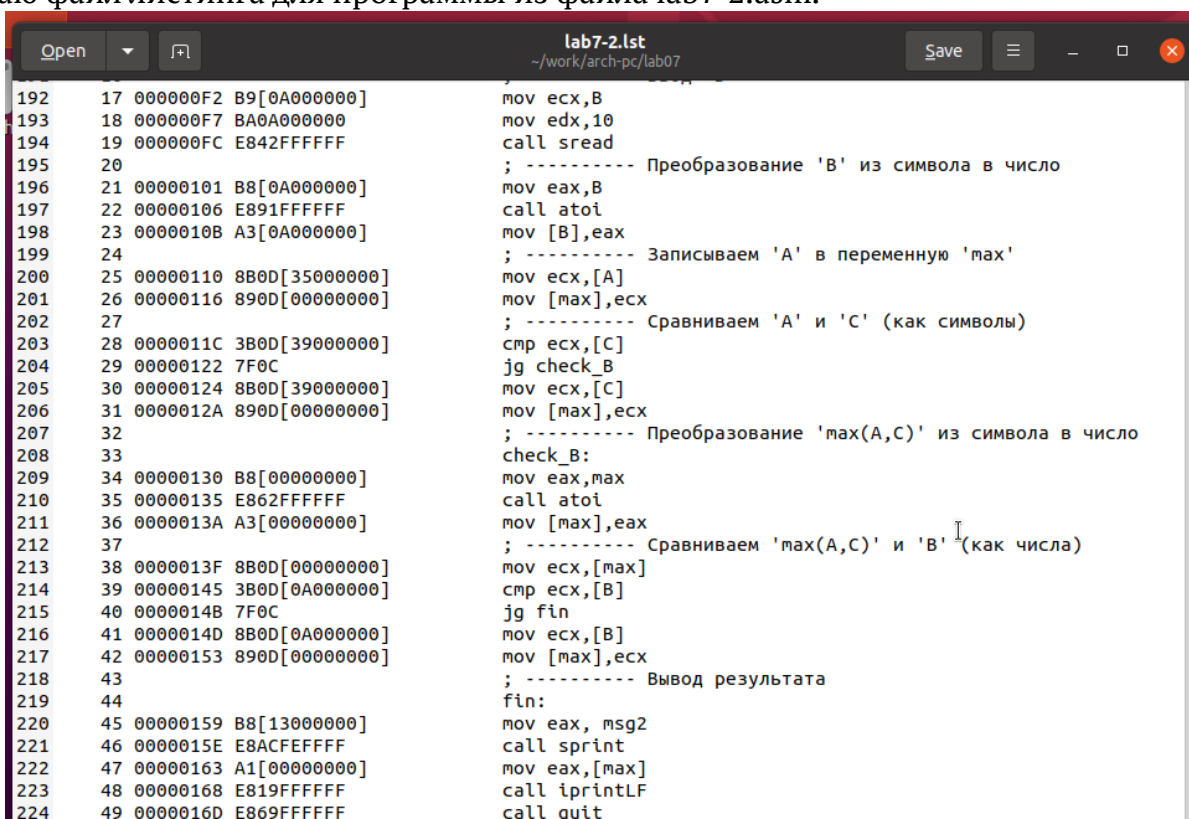
```

(рис.

2.2 Изучение структуры файла листинга

Обычно `nasm` создает в результате ассемблирования только объектный файл. Чтобы получить файл листинга, необходимо указать ключ `-l` и задать имя файла листинга в командной строке.

Создаю файл листинга для программы из файла `lab7-2.asm`.



```

lab7-2.lst
~/work/arch-pc/lab07
Save
192 17 000000F2 B9[0A000000] mov ecx,B
193 18 000000F7 BA0A000000 mov edx,10
194 19 000000FC E842FFFFFF call sread
195 20 ; ----- Преобразование 'B' из символа в число
196 21 00000101 B8[0A000000] mov eax,B
197 22 00000106 E891FFFFFF call atoi
198 23 0000010B A3[0A000000] mov [B],eax
199 24 ; ----- Записываем 'A' в переменную 'max'
200 25 00000110 8B0D[35000000] mov ecx,[A]
201 26 00000116 890D[00000000] mov [max],ecx
202 27 ; ----- Сравниваем 'A' и 'C' (как символы)
203 28 0000011C 3B0D[39000000] cmp ecx,[C]
204 29 00000122 7F0C jg check_B
205 30 00000124 8B0D[39000000] mov ecx,[C]
206 31 0000012A 890D[00000000] mov [max],ecx
207 32 ; ----- Преобразование 'max(A,C)' из символа в число
208 33 check_B:
209 34 00000130 B8[00000000] mov eax,max
210 35 00000135 E862FFFFFF call atoi
211 36 0000013A A3[00000000] mov [max],eax
212 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
213 38 0000013F 8B0D[00000000] mov ecx,[max]
214 39 00000145 3B0D[0A000000] cmp ecx,[B]
215 40 0000014B 7F0C jg fin
216 41 0000014D 8B0D[0A000000] mov ecx,[B]
217 42 00000153 890D[00000000] mov [max],ecx
218 43 ; ----- Вывод результата
219 44 fin:
220 45 00000159 B8[13000000] mov eax,msg2
221 46 0000015E E8ACFFFFFF call sprint
222 47 00000163 A1[00000000] mov eax,[max]
223 48 00000168 E819FFFFFF call iprintLF
224 49 0000016D E869FFFFFF call quit

```

(рис.

Ознакомимся с его форматом и содержанием.

- строка 211:

- 34 - номер строки
 - 0000012E - адрес
 - B8[00000000] - машинный код
 - mov eax, max - код программы
- строка 212:
 - 35 - номер строки
 - 00000133 - адрес
 - E864FFFFFF - машинный код
 - call atoi - код программы
 - строка 213:
 - 36 - номер строки
 - 00000138 - адрес
 - A3[00000000] - машинный код
 - mov [max], eax - код программы

Открываю файл с программой lab7-2.asm и удаляю один операнд из инструкции с двумя операндами. Затем выполняю трансляцию с получением файла листинга.

```
mismail@VirtualBox:~/work/arch-pc/lab07$
mismail@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
mismail@VirtualBox:~/work/arch-pc/lab07$
mismail@VirtualBox:~/work/arch-pc/lab07$
mismail@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:21: error: invalid combination of opcode and operands
mismail@VirtualBox:~/work/arch-pc/lab07$
mismail@VirtualBox:~/work/arch-pc/lab07$
```

(рис.

```

184 9 0000000A <res 0000000A> B resb 10
185 10 section .text
186 11 global _start
187 12 _start:
188 13 ; ----- Вывод сообщения 'Введите B: '
189 14 000000E8 B8[00000000] mov eax,msg1
190 15 000000ED E81DFFFFFF call sprint
191 16 ; ----- Ввод 'B'
192 17 000000F2 B9[0A000000] mov ecx,B
193 18 000000F7 BA0A000000 mov edx,10
194 19 000000FC E842FFFFFF call sread
195 20 ; ----- Преобразование 'B' из символа в число
196 21 mov eax,
197 21 *****
198 22 00000101 E896FFFFFF call atoi
199 23 00000106 A3[0A000000] mov [B],eax
200 24 ; ----- Записываем 'A' в переменную 'max'
201 25 0000010B 8B0D[35000000] mov ecx,[A]
202 26 00000111 890D[00000000] mov [max],ecx
203 27 ; ----- Сравниваем 'A' и 'C' (как символы)
204 28 00000117 3B0D[39000000] cmp ecx,[C]
205 29 0000011D 7F0C jg check_B
206 30 0000011F 8B0D[39000000] mov ecx,[C]
207 31 00000125 890D[00000000] mov [max],ecx
208 32 ; ----- Преобразование 'max(A,C)' из символа в число
209 33 check_B:
210 34 0000012B B8[00000000] mov eax,max
211 35 00000130 E867FFFFFF call atoi
212 36 00000135 A3[00000000] mov [max],eax
213 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
214 38 0000013A 8B0D[00000000] mov ecx,[max]
215 39 00000140 3B0D[0A000000] cmp ecx,[B]
216 40 00000146 7F0C ja fin

```

(рис. Объектный файл не смог создаться из-за ошибки, но файл листинга с выделенным местом ошибки был получен.)

2.3 Самостоятельное задание

Напишите программу нахождения наименьшей из трех целочисленных переменных a, b и c. Значения переменных выбрать из таблицы 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создаю исполняемый файл и проверяю его работу

```
36
37     mov eax,msgC
38     call sprintf
39     mov ecx,C
40     mov edx,80
41     call sread
42     mov eax,C
43     call atoi
44     mov [C],eax
45
46     mov ecx,[A]
47     mov [min],ecx
48
49     cmp ecx, [B]
50     jl check_C
51     mov ecx, [B]
52     mov [min], ecx
53
54 check_C:
55     cmp ecx, [C]
56     jl finish
57     mov ecx,[C]
58     mov [min],ecx
59
60 finish:
61     mov eax,answer
62     call sprintf
63
64     mov eax, [min]
65     call iprintLF
66
67     call quit
68
69
```

(рис. 6)

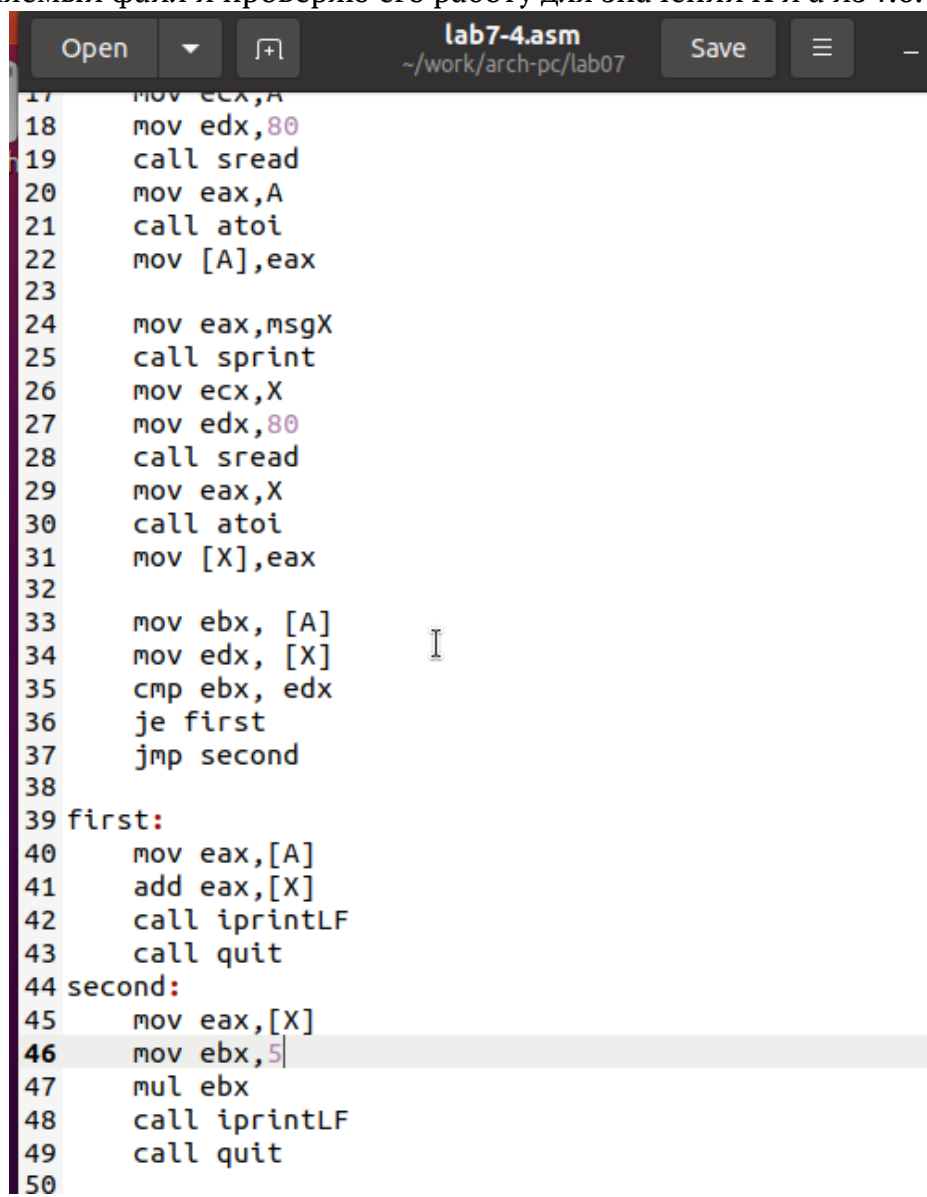
```
mismail@VirtualBox:~/work/arch-pc/lab07$
mismail@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
mismail@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-3.o -o lab7-3
mismail@VirtualBox:~/work/arch-pc/lab07$ ./lab7-3
Input A: 79
Input B: 83
Input C: 41
Smallest: 41
mismail@VirtualBox:~/work/arch-pc/lab07$
```

(рис. 7)

Для варианта 6 - 79,83,41

Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений.

Вид функции $f(x)$ выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создаю исполняемый файл и проверяю его работу для значений X и a из 7.6.



```

lab7-4.asm
~/work/arch-pc/lab07
Open Save

17 mov ecx,A
18 mov edx,80
19 call sread
20 mov eax,A
21 call atoi
22 mov [A],eax
23
24 mov eax,msgX
25 call sprintf
26 mov ecx,X
27 mov edx,80
28 call sread
29 mov eax,X
30 call atoi
31 mov [X],eax
32
33 mov ebx, [A]
34 mov edx, [X]
35 cmp ebx, edx
36 je first
37 jmp second
38
39 first:
40 mov eax,[A]
41 add eax,[X]
42 call iprintLF
43 call quit
44 second:
45 mov eax,[X]
46 mov ebx,5
47 mul ebx
48 call iprintLF
49 call quit
50

```

(рис.

)

```
mismail@VirtualBox:~/work/arch-pc/lab07$  
mismail@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm  
mismail@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-4.o -o lab7-4  
mismail@VirtualBox:~/work/arch-pc/lab07$ ./lab7-4  
Input A: 2  
Input X: 2  
4  
mismail@VirtualBox:~/work/arch-pc/lab07$ ./lab7-4  
Input A: 1  
Input X: 2  
10  
mismail@VirtualBox:~/work/arch-pc/lab07$
```

(рис.

Для варианта 6:

$$\begin{cases} x + a, & x = a \\ 5x, & x \neq a \end{cases}$$

При (x = 2, a = 2) получается 4

При (x = 2, a = 1) получается 10

3 Выводы

Изучили команды условного и безусловного переходов, познакомились с фалом листинга.