

Лабораторная работа No7. Команды безусловного и условного переходов в Nasm. Программирование ветвлений

Ромицына Анастасия Романовна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Изучение структуры файлы листинга	10
2.2	Задание для самостоятельной работы	12
3	Выводы	16

Список иллюстраций

2.1	Создание файла	6
2.2	Ввод прораммы	6
2.3	Запуск программы	7
2.4	Изменение прграммы	7
2.5	Запуск программы	8
2.6	Редактирование файла	8
2.7	Вывод на экран	8
2.8	Создание нового файла	9
2.9	Редактирование файла	9
2.10	Запуск файла	9
2.11	Создание файла	10
2.12	Изучаем файл	10
2.13	Удаляем операндум из файла	11
2.14	Транслируем файл	11
2.15	Изучаем файл с ошибкой	12
2.16	Создаем файл командой touch	12
2.17	Пишем программу	13
2.18	Смотрим на рабботу программы(всё верно)	13
2.19	Редактирование нового файла	14
2.20	Запуск программы	15
2.21	Запуск программы	15

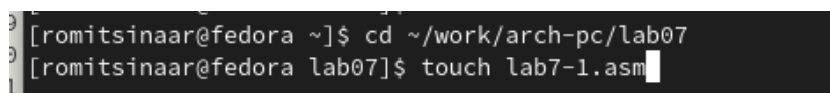
Список таблиц

1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга. # Задание Написать программы для решения задач.

2 Выполнение лабораторной работы

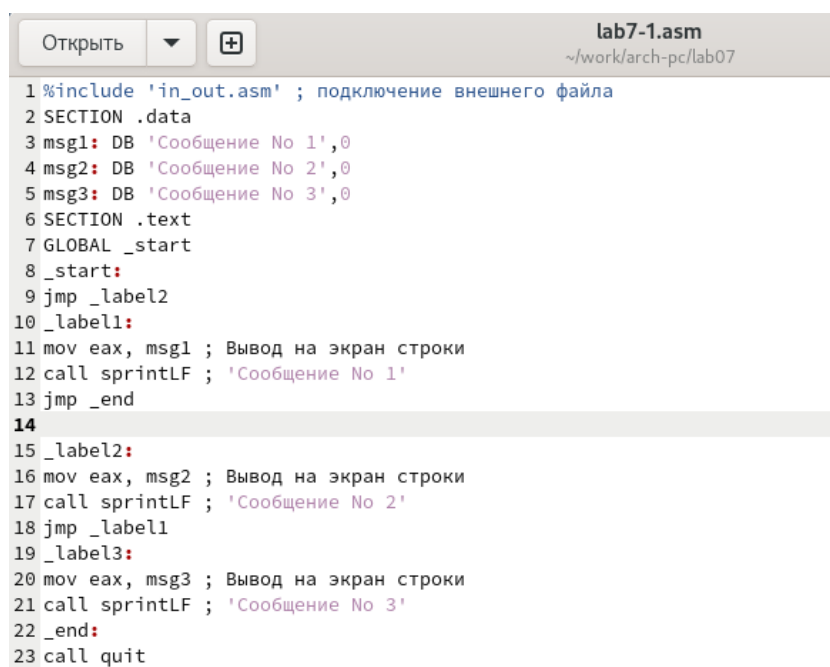
Создадим каталог для программ лабораторной работы No 7, перейдем в него и создадим файл lab7-1.asm:(рис. 2.1).



```
[romitsinaar@fedora ~]$ cd ~/work/arch-pc/lab07
[romitsinaar@fedora lab07]$ touch lab7-1.asm
```

Рис. 2.1: Создание файла

Введем в файл программу из листинга 7.1 (рис. 2.2).



```
lab7-1.asm
~/work/arch-pc/lab07

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

Рис. 2.2: Ввод программы

Запускаем и смотрим ответ (рис. 2.3).

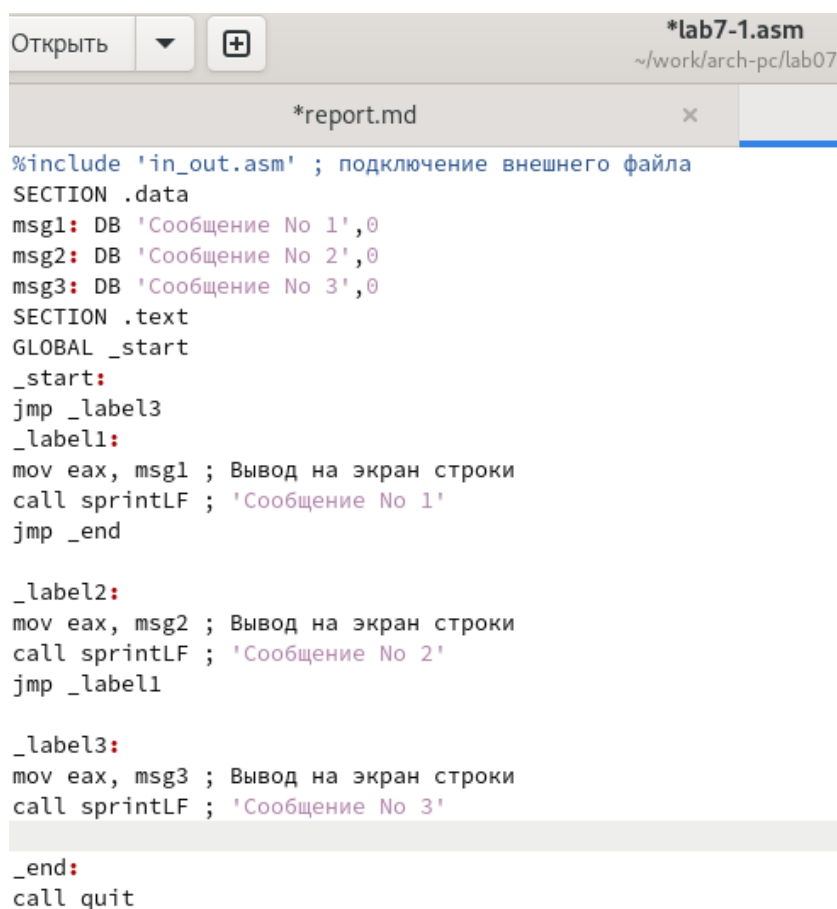
```

[romitsinaar@fedora lab07]$ nasm -f elf lab7-1.asm
[romitsinaar@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab5-7.o
ld: невозможно найти lab5-7.o: Нет такого файла или каталога
[romitsinaar@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[romitsinaar@fedora lab07]$ ./lab7-1
Сообщение No 2
Сообщение No 3
[romitsinaar@fedora lab07]$

```

Рис. 2.3: Запуск программы

Изменяем программу по примеру (рис. 2.4).



```

*lab7-1.asm
~/work/arch-pc/lab07

*report.md

#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label3
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
jmp _end

_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
jmp _label1

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'

_end:
call quit

```

Рис. 2.4: Изменение программы

Запускаем её и смотрим, что выводится на экран(рис. 2.5).

```

[romitsinaar@fedora lab07]$ gedit lab7-1.asm
[romitsinaar@fedora lab07]$ nasm -f elf lab7-1.asm
[romitsinaar@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[romitsinaar@fedora lab07]$ ./lab7-1
Сообщение No 2
Сообщение No 1
[romitsinaar@fedora lab07]$

```

Рис. 2.5: Запуск программы

Изменяем программу так, чтоб она выводила все сообщения (рис. 2.6).

```

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

```

Рис. 2.6: Редактирование файла

Запускаем программу и проверяем её, все выводится верно.(рис. 2.7).

```

[romitsinaar@fedora lab07]$ gedit lab7-1.asm
[romitsinaar@fedora lab07]$ nasm -f elf lab7-1.asm
[romitsinaar@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[romitsinaar@fedora lab07]$ ./lab7-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
[romitsinaar@fedora lab07]$

```

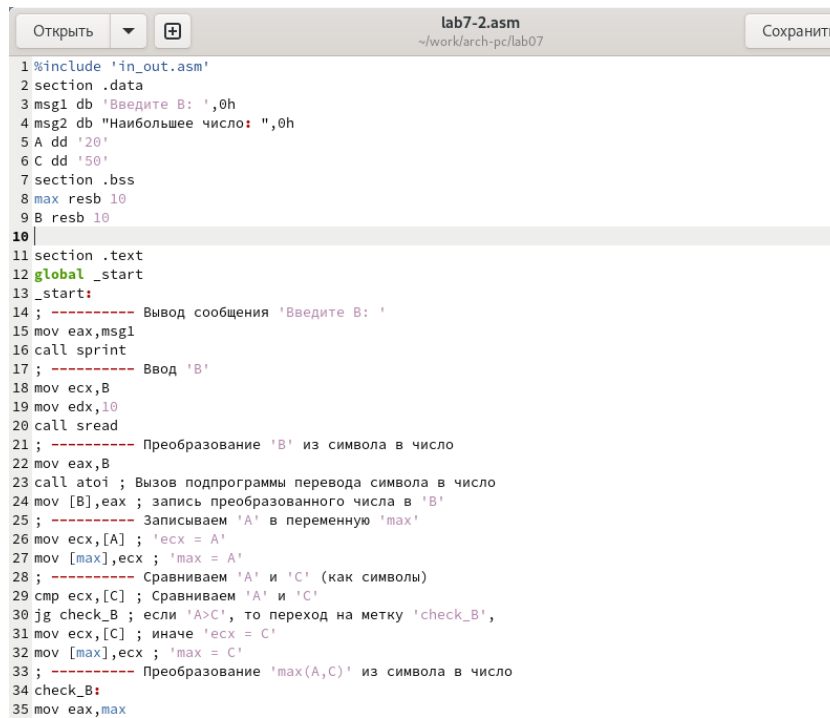
Рис. 2.7: Вывод на экран

Создаем новый файл.(рис. 2.8).

```
[romitsinaar@fedora lab07]$ touch lab7-2.asm
```

Рис. 2.8: Создание нового файла

Изцчаем и вписываем в него программу из листинга.(рис. 2.9).



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

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
11 section .text
12 global _start
13 _start:
14 ; ----- Вывод сообщения 'Введите B: '
15 mov eax,msg1
16 call sprint
17 ; ----- Ввод 'B'
18 mov ecx,B
19 mov edx,10
20 call sread
21 ; ----- Преобразование 'B' из символа в число
22 mov eax,B
23 call atoi ; Вызов подпрограммы перевода символа в число
24 mov [B],eax ; запись преобразованного числа в 'B'
25 ; ----- Записываем 'A' в переменную 'max'
26 mov ecx,[A] ; 'ecx = A'
27 mov [max],ecx ; 'max = A'
28 ; ----- Сравниваем 'A' и 'C' (как символы)
29 cmp ecx,[C] ; Сравниваем 'A' и 'C'
30 jg check_B ; если 'A>C', то переход на метку 'check_B',
31 mov ecx,[C] ; иначе 'ecx = C'
32 mov [max],ecx ; 'max = C'
33 ; ----- Преобразование 'max(A,C)' из символа в число
34 check_B:
35 mov eax,max
```

Рис. 2.9: Редактирование файла

Запускаем новый файл. Проверяем его работу на любом числе(рис. 2.10).

```
[romitsinaar@fedora lab07]$ nasm -f elf lab7-2.asm
[romitsinaar@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[romitsinaar@fedora lab07]$ ./lab7-2
Введите B: 4
Наибольшее число: 50
[romitsinaar@fedora lab07]$
```

Рис. 2.10: Запуск файла

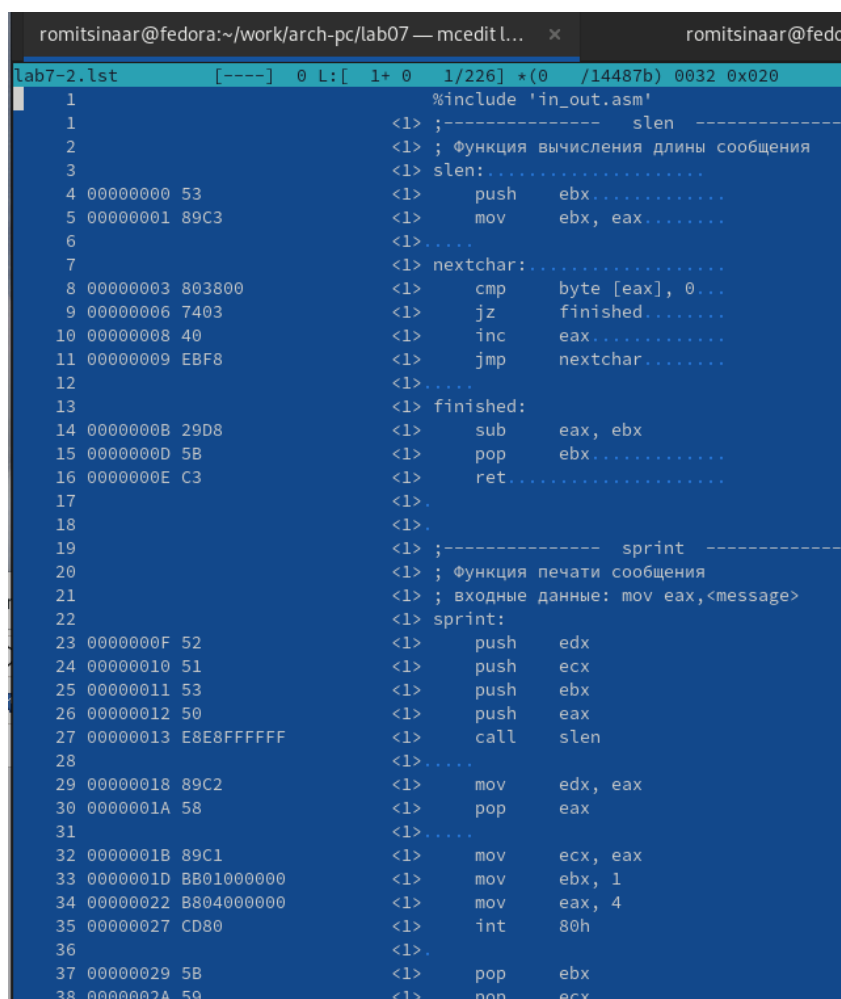
2.1 Изучение структуры файлы листинга

Создадим файл листинга для программы из файла lab7-2.asm (рис. 2.11).

```
[romitsinaar@fedora lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
```

Рис. 2.11: Создание файла

Открываем файл листинга с помощью команды `mcedit` и изучаем его (рис. 2.12).



```
lab7-2.lst  [----]  0 L:[ 1+ 0 1/226] *(0 /14487b) 0032 0x020
1          %include 'in_out.asm'
2          <1> ;----- slen -----
3          <1> ; Функция вычисления длины сообщения
4          <1> slen:.....
5          00000000 53          <1>   push    ebx.....
6          00000001 89C3       <1>   mov     ebx, eax.....
7          <1>.....
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         <1>.....
14         <1> finished:
15         0000000B 29D8       <1>   sub     eax, ebx
16         0000000D 5B         <1>   pop     ebx.....
17         0000000E C3         <1>   ret.....
18         <1>.....
19         <1> ;----- sprint -----
20         <1> ; Функция печати сообщения
21         <1> ; входные данные: mov eax,<message>
22         <1> sprint:
23         0000000F 52         <1>   push    edx
24         00000010 51         <1>   push    ecx
25         00000011 53         <1>   push    ebx
26         00000012 50         <1>   push    eax
27         00000013 E8E8FFFF   <1>   call    slen
28         <1>.....
29         00000018 89C2       <1>   mov     edx, eax
30         0000001A 58         <1>   pop     eax
31         <1>.....
32         0000001B 89C1       <1>   mov     ecx, eax
33         0000001D BB01000000 <1>   mov     ebx, 1
34         00000022 B804000000 <1>   mov     eax, 4
35         00000027 CD80       <1>   int     80h
36         <1>.....
37         00000029 5B         <1>   pop     ebx
38         0000002A 59         <1>   pop     ecx
```

Рис. 2.12: Изучаем файл

Строка 33: 0000001D-адрес в сегменте кода, BB01000000-машинный код, `mov`

ebx,1-присвоение переменной ecx значения 1.

Строка 34: 00000022-адрес в сегменте кода, B804000000-машинный код, mov eax,4-присвоение переменной eax значения 4.

Строка 35 00000027-адрес в сегменте кода, CD80-машинный код, int 80h-вызов ядра.

Открываем файл и удаляем один операндум (рис. 2.13).

```
14 ; ----- Вывод сообщения 'Введите B: '  
15 mov eax,msg1  
16 call sprint  
17 ; ----- Ввод 'B'  
18 mov ecx,B  
19 mov edx|  
20 call sread  
21 ; ----- Преобразование 'B' из символа в число  
22 mov eax,B  
23 call atoi ; Вызов подпрограммы перевода символа в число  
24 mov [B],eax ; запись преобразованного числа в 'B'  
25 ; ----- Записываем 'A' в переменную 'max'
```

Рис. 2.13: Удаляем операндум из файла

Транслируем с получением файла листинга (рис. 2.14).

```
lab7-2.asm:19: error: invalid combination of opcode and operands  
[romitsinaar@fedora lab07]$ ls  
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.lst  
[romitsinaar@fedora lab07]$
```

Рис. 2.14: Транслируем файл

При трансляции файла, выдается ошибка, но создаются исполнительный файл lab7-2 и lab7-2.lst

Снова открываем файл листинга и изучаем его (рис. 2.15).

```

lab7-2.lst      [----]  0 L:  1+ 0  1/227] *(0  /14575b) 0032 0x020
1               %include 'in_out.asm'
2               <1> ;----- slen -----
3               <1> ; Функция вычисления длины сообщения
4               <1> slen:.....
5 00000000 53    <1>   push    ebx.....
6 00000001 89C3  <1>   mov     ebx, eax.....
7               <1>.....
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               <1>.....
14               <1> finished:
15 0000000B 29D8  <1>   sub     eax, ebx
16 0000000D 5B    <1>   pop     ebx.....
17 0000000E C3    <1>   ret.....
18               <1>.....
19               <1> ;----- sprint -----
20               <1> ; Функция печати сообщения
21               <1> ; входные данные: mov eax,<message>
22               <1> sprint:
23 0000000F 52    <1>   push    edx
24 00000010 51    <1>   push    ecx
25 00000011 53    <1>   push    ebx
26 00000012 50    <1>   push    eax
27 00000013 E8E8FFFF <1>   call    slen
28               <1>.....
29 00000018 89C2  <1>   mov     edx, eax
30 0000001A 58    <1>   pop     eax
31               <1>.....
32 0000001B 89C1  <1>   mov     ecx, eax
33 0000001D B801000000 <1>   mov     ebx, 1
34 00000022 B804000000 <1>   mov     eax, 4
35 00000027 CD80  <1>   int     80h
36               <1>.....
37 00000029 5B    <1>   pop     ebx
38 0000002A 59    <1>   pop     ecx

```

Рис. 2.15: Изучаем файл с ошибкой

2.2 Задание для самостоятельной работы

ВАРИАНТ-13

1

Создаем новый файл (рис. 2.16).

```

[romitsinaar@fedora lab07]$ touch lab7-3.asm
[romitsinaar@fedora lab07]$

```

Рис. 2.16: Создаем файл командой touch

Открываем его и пишем программу, которая выберет наименьшее число из трех(2 числа уже в программе, 3е вводится из консоли) (рис. 2.17).

```

1 %include 'in_out.asm'
2 section .data
3 msg1 db '',0h
4 msg2 db '',0h
5 A dd '84'
6 C dd '32'
7 section .bss
8 min resb 10
9 B resb 10
10 section .text
11 global _start
12 _start:
13 mov eax, msg1
14 call sprint
15 mov edx, B
16 mov edx, 10
17 call sread
18 mov eax, B
19 call atoi
20 mov [B], eax
21 mov ecx, [A]
22 mov [min], ecx
23 cmp ecx, [C]
24 jl check_B
25 mov ecx, [C]
26 mov [min], ecx
27 check_B:
28 mov eax, min
29 call atoi
30 mov [min], eax
31 mov ecx, [min]
32 cmp ecx, [B]
33 jl fin
34 mov ecx, [B]
35 mov [min], ecx
36 fin:
37 mov eax, msg2
38 call sprint
39 mov eax, [min]

```

Рис. 2.17: Пишем программу

Запускаем и смотрим на работу программы (рис. 2.18).

```

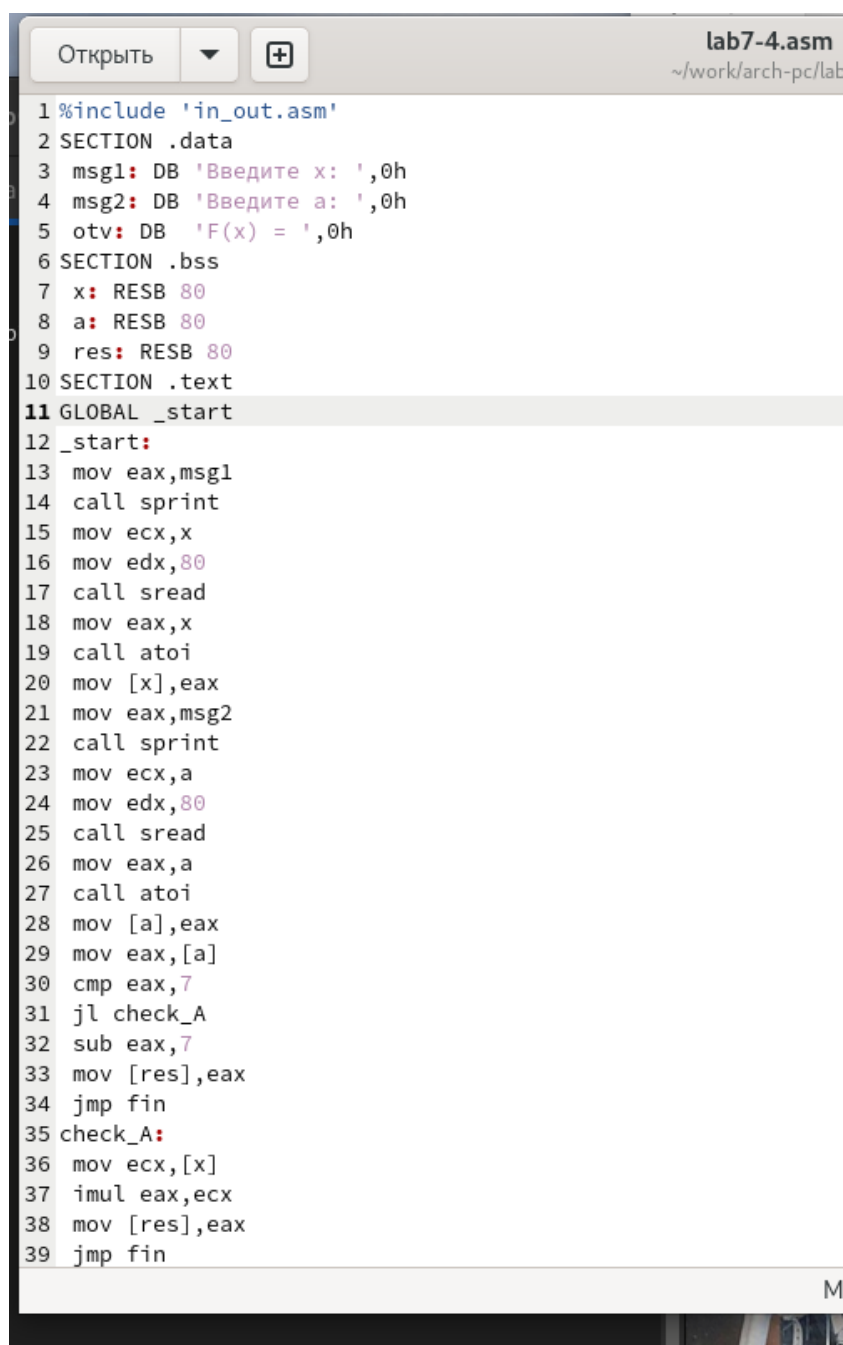
[romitsinaar@fedora lab07]$ gedit lab7-3.asm
[romitsinaar@fedora lab07]$ nasm -f elf lab7-3.asm
[romitsinaar@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[romitsinaar@fedora lab07]$ ./lab7-3
Введите число: 77
наименьшее: 32
[romitsinaar@fedora lab07]$

```

Рис. 2.18: Смотрим на работу программы(всё верно)

2

Создаем и редактируем новый файл, пишем в него нужную программу.(рис. 2.19).



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg1: DB 'Введите x: ',0h
4 msg2: DB 'Введите a: ',0h
5 otv: DB 'F(x) = ',0h
6 SECTION .bss
7 x: RESB 80
8 a: RESB 80
9 res: RESB 80
10 SECTION .text
11 GLOBAL _start
12 _start:
13 mov eax,msg1
14 call sprint
15 mov ecx,x
16 mov edx,80
17 call sread
18 mov eax,x
19 call atoi
20 mov [x],eax
21 mov eax,msg2
22 call sprint
23 mov ecx,a
24 mov edx,80
25 call sread
26 mov eax,a
27 call atoi
28 mov [a],eax
29 mov eax,[a]
30 cmp eax,7
31 jl check_A
32 sub eax,7
33 mov [res],eax
34 jmp fin
35 check_A:
36 mov ecx,[x]
37 imul eax,ecx
38 mov [res],eax
39 jmp fin
```

Рис. 2.19: Редактирование нового файла

Запускаем программу и вписываем значения из 1 примера(все верно) (рис. 2.20).

```
[romitsinaar@fedora lab07]$ gedit lab7-4.asm
[romitsinaar@fedora lab07]$ nasm -f elf lab7-4.asm
[romitsinaar@fedora lab07]$ ld -m elf_i386 -o lab7-4 lab7-4.o
[romitsinaar@fedora lab07]$ ./lab7-4
Введите x: 3
Введите a: 9
F(x) = 2
[romitsinaar@fedora lab07]$
```

Рис. 2.20: Запуск программы

Запускаем программу и вписываем значения из 2 примера(все верно) (рис. 2.21).

```
[romitsinaar@fedora lab07]$ ./lab7-4
Введите x: 6
Введите a: 4
F(x) = 24
[romitsinaar@fedora lab07]$
```

Рис. 2.21: Запуск программы

3 Выводы

Мы изучили команды условного и безусловного переходов. Приобрели навыки написания программ с использованием переходов. Познакомились с назначением и структурой файла листинга.