

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7

дисциплина: Архитектура компьютера

Студент: Семёнов Александр Дмитриевич

Группа: НКАбд-05-25

МОСКВА

2025 г.

Содержание

1. Цель работы.

Изучить команды условного и безусловного переходов, приобрести навыки написания программ с использованием переходов познакомиться с назначением и структурой файла листинга.

2. Задание.

Изучить команды условного и безусловного переходов, приобрести навыки написания программ с использованием переходов и познакомиться с назначением и структурой файла листинга.

3. Выполнение лабораторной работы.

3.1 Реализация переходов в NASM.

Я создание и переход в каталог для лабораторной №7.

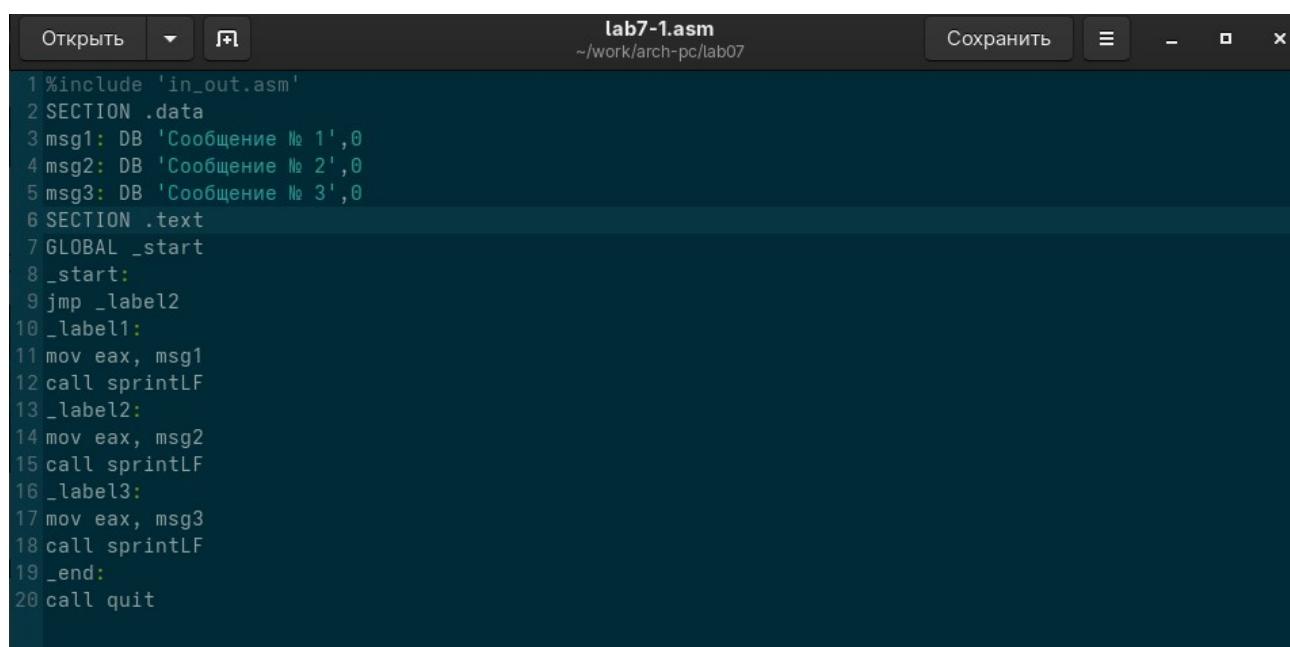
```
adsemyonov@fedora:~$ mkdir -p ~/work/arch-pc/lab07
adsemyonov@fedora:~$ cd ~/work/arch-pc/lab07
adsemyonov@fedora:~/work/arch-pc/lab07$
```

Рис. 1. Создание каталога и переход в него.

Я создал файл **lab7-1.asm** и ввел в него текст программы из листинга 7.1.

```
adsemyonov@fedora:~/work/arch-pc/lab07$ touch lab7-1.asm
adsemyonov@fedora:~/work/arch-pc/lab07$ gedit lab7-1.asm
```

Рис. 2. Создание и открытие файла.



The screenshot shows the Gedit text editor window with the file 'lab7-1.asm' open. The window title is 'lab7-1.asm' and the path is '~ / work / arch - pc / lab07'. The code in the editor is:

```
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 _start:
9 jmp _label2
10 _label1:
11 mov eax, msg1
12 call sprintLF
13 _label2:
14 mov eax, msg2
15 call sprintLF
16 _label3:
17 mov eax, msg3
18 call sprintLF
19 _end:
20 call quit
```

Рис. 3. Текст программы.

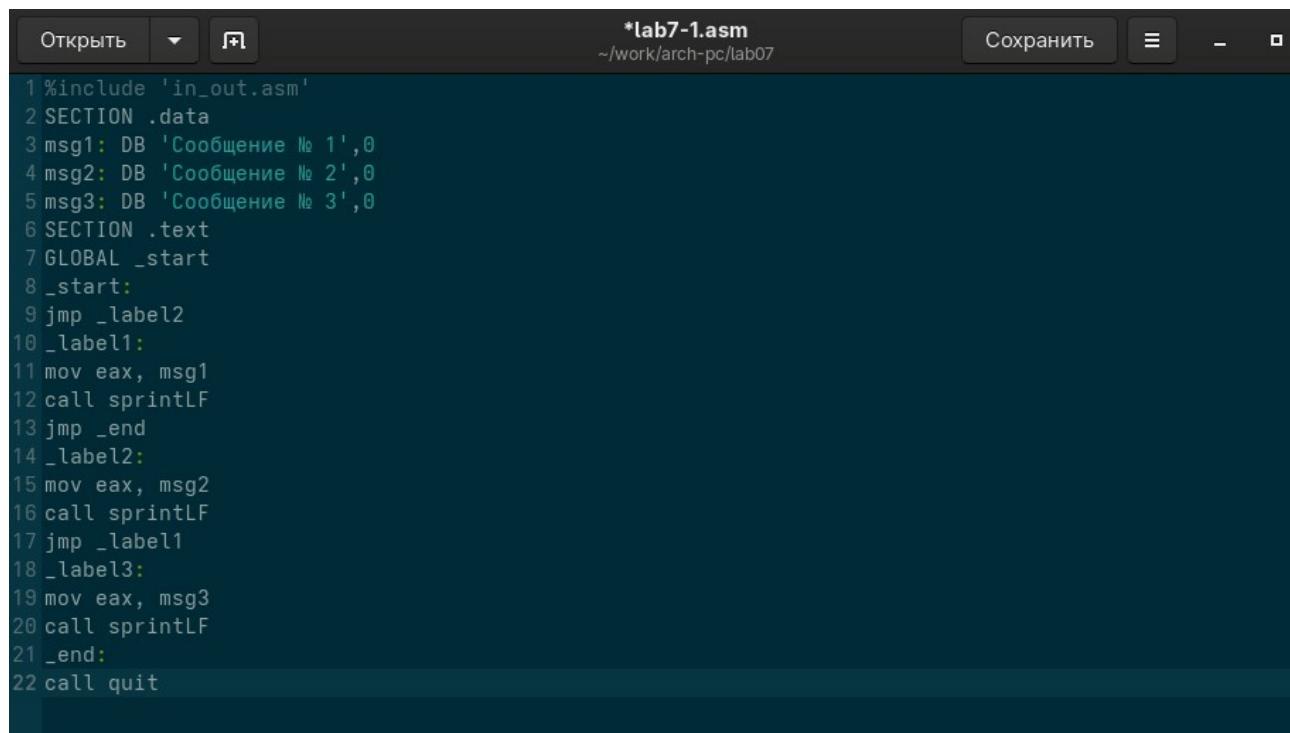
Потом я создал файл и запустил его. Файл **in_out.asm** я скопировал через **Midnight Commander**.

```

adsemyonov@fedora:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1.asm
adsemyonov@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
lab7-1.asm:3: warning: no operand for data declaration [-w+db-empty]
lab7-1.asm:4: warning: no operand for data declaration [-w+db-empty]
lab7-1.asm:5: warning: no operand for data declaration [-w+db-empty]
adsemyonov@fedora:~/work/arch-pc/lab07$ gedit lab7-1.asm
adsemyonov@fedora:~/work/arch-pc/lab07$ gedit lab7-1.asm
adsemyonov@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
adsemyonov@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1.o
ld: no input files
adsemyonov@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
adsemyonov@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
adsemyonov@fedora:~/work/arch-pc/lab07$ █

```

Рис. 4. Создание и запуск файла.



The screenshot shows the Gedit text editor interface with the file `*lab7-1.asm` open. The file path is `~/work/arch-pc/lab07`. The assembly code contains three messages ('Сообщение № 1', 'Сообщение № 2', 'Сообщение № 3') each followed by a null byte, and a loop that prints them out.

```

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 _start:
9 jmp _label2
10 _label1:
11 mov eax, msg1
12 call sprintLF
13 jmp _end
14 _label2:
15 mov eax, msg2
16 call sprintLF
17 jmp _label1
18 _label3:
19 mov eax, msg3
20 call sprintLF
21 _end:
22 call quit

```

Рис. 5. Измененный текст программы.

Далее я изменил текст программы в файле **lab7-1.asm** в соответствие с листингом 7.2.

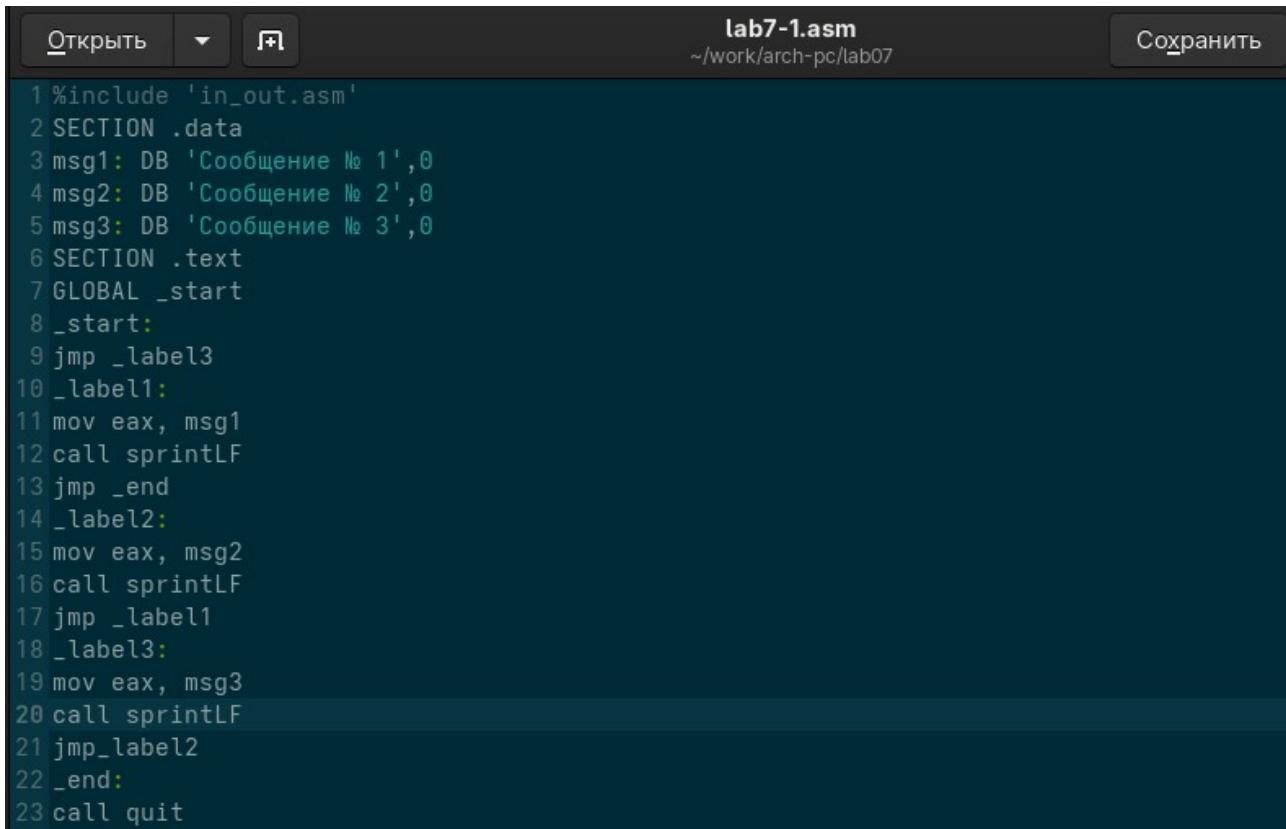
```

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

```

Рис. 6. Создание и запуск файла.

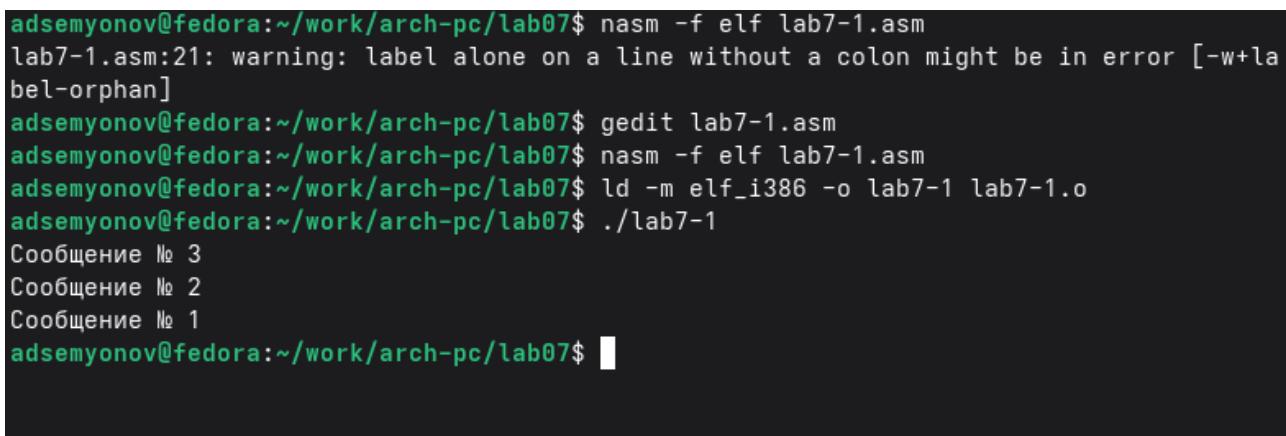
Потом я опять изменил текст программы и запустил ее.



The screenshot shows a text editor window with the title "lab7-1.asm" and the path "~/work/arch-pc/lab07". The code in the editor is as follows:

```
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 _start:
9 jmp _label3
10 _label1:
11 mov eax, msg1
12 call sprintLF
13 jmp _end
14 _label2:
15 mov eax, msg2
16 call sprintLF
17 jmp _label1
18 _label3:
19 mov eax, msg3
20 call sprintLF
21 jmp _label2
22 _end:
23 call quit
```

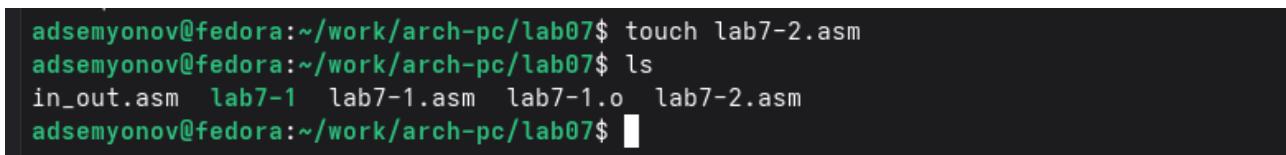
Рис. 7. Изменение программы во второй раз.



```
adsemyonov@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
lab7-1.asm:21: warning: label alone on a line without a colon might be in error [-w+label-orphan]
adsemyonov@fedora:~/work/arch-pc/lab07$ gedit lab7-1.asm
adsemyonov@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
adsemyonov@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
adsemyonov@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
adsemyonov@fedora:~/work/arch-pc/lab07$
```

Рис. 8. Создание и запуск файла.

Я создал файл **lab7-2.asm**.



```
adsemyonov@fedora:~/work/arch-pc/lab07$ touch lab7-2.asm
adsemyonov@fedora:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2.asm
adsemyonov@fedora:~/work/arch-pc/lab07$
```

Рис. 9. Создание файла.

Потом я ввел текст программы из листинга 7.3 и проверил работу программы для разных значений В.

The screenshot shows a text editor window with the following details:

- File menu: Открыть (Open), Сохранить (Save).
- Title bar: lab7-2.asm, ~/work/arch-pc/lab07.
- Code area (Assembly language):

```
1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите В: ',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 mov eax,msg1
14 call sprint
15 mov ecx,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 [max],ecx
23 cmp ecx,[C]
24 jg check_B
25 mov ecx,[C]
26 mov [max],ecx
27 check_B:
28 mov eax,max
29 call atoi
30 mov [max],eax
31 mov ecx,[max]
32 cmp ecx,[B]
33 jg fin
34 mov ecx,[B]
35 mov [max],ecx
36 fin:
37 mov eax, msg2
38 call sprint
39 mov eax,[max]
40 call iprintLF
41 call quit
```

Рис. 10. Текст программы.

```
adsemyonov@fedora:~/work/arch-pc/lab07$ gedit lab7-2.asm
adsemyonov@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
adsemyonov@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
adsemyonov@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 6
Наибольшее число: 50
adsemyonov@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 29
Наибольшее число: 50
adsemyonov@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 100
Наибольшее число: 100
adsemyonov@fedora:~/work/arch-pc/lab07$
```

Рис. 11. Создание и запуск файла.

3.2 Изучение структуры файла листинга.

Я создал файл листинга для программы из файла **lab7-2.asm** и открыл его с помощью текстового редактора **mcedit**.

```
Наибольшее число: 100
adsemyonov@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
adsemyonov@fedora:~/work/arch-pc/lab07$ mcedit lab7-2.lst

adsemyonov@fedora:~/work/arch-pc/lab07$
```

Рис. 12. Создание файла листинга.

С помощью **ls -la** я сравнил время когда были созданы исполняемый файл и файл листинга. Как и ожидалось, исполняемый файл старый, потому новый не был создан из-за ошибки, а файл листинга создался только что.

```
adsemyonov@fedora:~/work/arch-pc/lab07$ ls -la lab7-2 lab7-2.lst
-rwxr-xr-x. 1 adsemyonov adsemyonov 5144 ноя 23 16:12 lab7-2
-rw-r--r--. 1 adsemyonov adsemyonov 13037 ноя 23 16:30 lab7-2.lst
```

Рис. 13. Просмотр.

```
~ /work/arch-pc/lab07  
lab7-2.lst      [---]  0 L:[ 1+ 0  1/217] *(0  /12951b) 0032 0x020  
1          %include 'in_out.asm'  
1          ;-----  slen -----  
2          ; Функция вычисления длины сообщения  
3          slen:  
4 00000000 53          push    ebx.....  
5 00000001 89C3          mov     ebx, eax.....  
6          .....  
7          nextchar:  
8 00000003 803800          cmp     byte [eax], 0...  
9 00000006 7403          jz      finished.....  
10 00000008 40          inc     eax.....  
11 00000009 EBF8          jmp     nextchar.....  
12          .....  
13          finished:  
14 0000000B 29D8          sub     eax, ebx  
15 0000000D 5B          pop     ebx.....  
16 0000000E C3          ret.....  
17          .....  
18          .....  
19          ;-----  sprint -----  
20          ; Функция печати сообщения  
21          ; входные данные: mov eax,<message>  
22          sprint:  
23 0000000F 52          push    edx  
24 00000010 51          push    ecx  
25 00000011 53          push    ebx  
26 00000012 50          push    eax  
27 00000013 E8E8FFFFFF          call    slen  
28          .....  
29 00000018 89C2          mov     edx, eax  
30 0000001A 58          pop     eax  
31          .....  
32 0000001B 89C1          mov     ecx, eax  
33 0000001D BB01000000          mov     ebx, 1  
34 00000022 B804000000          mov     eax, 4  
35 00000027 CD80          int    80h  
36          .....  
37 00000029 5B          pop     ebx  
38 0000002A 59          pop     ecx  
39 0000002B 5A          pop     edx  
40 0000002C C3          ret.....  
41          .....  
42          .....  
43          ;-----  sprintLF -----  
44          ; Функция печати сообщения с переводом строки  
45          ; входные данные: mov eax,<message>
```

1 Помощь 2 Сохран 3 Блок 4 Замена 5 Копия 6 Перенить 7 Поиск 8 Удалить 9 М

Рис. 13. Содержимое листинга.

Строка 8: cmp byte [eax], 0

Что делает: Сравнивает байт (символ) по адресу в регистре EAX с нулем.
Объяснение:

- byte - операция с одним байтом
 - [eax] - обращение к памяти по адресу в регистре EAX
 - 0 - нулевой байт (символ конца строки в C-style строках)
- Назначение: Проверяет, не достигнут ли конец строки (нулевой терминатор)

Строка 27: call slen

Что делает: Вызывает функцию slen (string length)

Объяснение:

- call - команда вызова подпрограммы
 - slen - имя функции вычисления длины строки
 - Адрес FFFFFFFEB - это смещение -21 в дополнительном коде (переход назад)
- Назначение: Вычисляет длину строки для последующего вывода

Строка 35: int 80h

Что делает: Вызывает прерывание ядра Linux

Объяснение:

- int - команда вызова программного прерывания
- 80h - номер прерывания (128 в десятичной системе)
- Перед этим в регистрах устанавливаются параметры:
 - eax=4 - номер системного вызова (sys_write)
 - ebx=1 - файловый дескриптор (stdout)
 - ecx - указатель на строку
 - edx - длина строки

Назначение: Вывод строки на экран через системный вызов Linux

После этого я открыла файл **lab7-2.asm** и удалила один operand.

```
21 mov ecx|  
Rис. 14. До удаления было: mov ecx,[A].
```

Потом я выполнил трансляцию с получением файла листинга.

```
adsemyonov@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm  
lab7-2.asm:21: error: invalid combination of opcode and operands  
adsemyonov@fedora:~/work/arch-pc/lab07$ |  
Rис. 15. Выполнение трансляции.
```

Использовал команду **ls**, чтобы проверить какие файлы у меня есть.

```
adsemyonov@fedora:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2  lab7-2.asm  lab7-2.lst
adsemyonov@fedora:~/work/arch-pc/lab07$
```

Рис. 16. Проверка файлов.

```
*****          mov ecx
*****          error: invalid combination of opcode and operands
0D[00000000]    mov [max],ecx.
```

Рис. 17. Сообщение об ошибке.

4. Задания для самостоятельной работы.

Задание №1.

Я создал файл **lab7-3.asm** для выполнения первого задания.

```
adsemyonov@fedora:~/work/arch-pc/lab07$ touch lab7-3.asm  
adsemyonov@fedora:~/work/arch-pc/lab07$ █
```

Рис. 18. Создание файла.

Я написал программу по нахождению наименьшей из 3 целочисленных переменных a,b и с. Значения я взял из таблицы в соответствии с полученным мной 8 вариантом, и потом я запустил ее.

```
1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Наименьшее число: ',0h
4 A db '52',0
5 B db '33',0
6 C db '40',0
7 section .bss
8 resb 10
9 section .text
10 global _start
11 _start:
12 mov eax,A
13 call atoi
14 mov [min],eax
15 mov eax,B
16 call atoi
17 mov ebx,eax
18 mov eax,[min]
19 cmp eax,ebx
20 jl check_C
21 mov [min],ebx
22 check_C:
23 mov eax,C
24 call atoi
25 mov ebx,eax
26 mov eax,[min]
27 cmp eax,ebx
28 jl fin
29 mov [min],ebx
30 fin:
31 mov eax,msg1
32 call sprint
33 mov eax,[min]
34 call iprintLF
35 call quit
```

Рис. 19. Текст программы.

```
adsemyonov@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
adsemyonov@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
adsemyonov@fedora:~/work/arch-pc/lab07$ ./lab7-3
Наименьшее число: 33
adsemyonov@fedora:~/work/arch-pc/lab07$
```

Рис. 20. Создание и запуск файла.

Задание №2.

Я создал файл **lab7-4.asm** для выполнения второго задания.

```
adsemyonov@fedora:~/work/arch-pc/lab07$ touch lab7-4.asm  
adsemyonov@fedora:~/work/arch-pc/lab07$
```

Рис. 21. Создание файла.

Я написал программу, вычисляющую значение заданной функции для введенных x и a в соответствии с 8 вариантом и запустил программу.

Открыть ▼ lab/-4.asm
~/work/arch-pc/lab07

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg.x: DB 'Введите x: ',0
4 msg.a: DB 'Введите a: ',0
5 msg.res: DB 'Результат: ',0
6 SECTION .bss
7 x: RESB 10
8 a: RESB 10
9 SECTION .text
10 GLOBAL _start
11 _start:
12 mov eax, msg.x
13 call sprint
14 mov ecx, x
15 mov ebx, 10
16 call sread
17 mov eax, x
18 call atoi
19 mov [x], eax
20 mov eax, msg.a
21 call sprint
22 mov ecx, a
23 mov ebx, 10
24 call sread
25 mov eax, a
26 call atoi
27 mov [a], eax
28 mov ebx, [a]
29 cmp ebx, 3
30 jl case1
31 mov eax, [x]
32 add eax, 1
33 jmp result
34 case1:
35 mov eax, [a]
36 mov ebx, 3
37 mul ebx
38 result:
39 mov ebx, eax
40 mov eax, msg.res
41 call sprint
42 mov eax, ebx
43 call iprintLF
44 call quit
```

Рис. 22. Текст программы.

```
adsemyonov@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
adsemyonov@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
adsemyonov@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 1
Введите a: 4
Результат: 2
adsemyonov@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 1
Введите a: 2
Результат: 6
adsemyonov@fedora:~/work/arch-pc/lab07$ █
```

Рис. 23. Выполнение программы.

5. Выводы.

Я изучил команды условного и безусловного переходов. Я приобрел навыки написания программ с использованием переходов и познакомился с назначением и структурой файла листинга.

Список литературы.

1. <https://esystem.rudn.ru/course/view.php?id=112>