Лабораторная работа №7

НПИбд-01-25 №1032252598

Иванова Ангелина Олеговна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Целью данной лабораторной работы является изучение команд условного и безусловного переходов, а также приобретение навыков написания программ с использованием переходов и знакомство с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

2.1 Задания лабораторной работы

2.1.1 Реализация переходов в NASM

Создали каталог для программам лабораторной работы № 7, перешли в него и создали файл lab7-1.asm

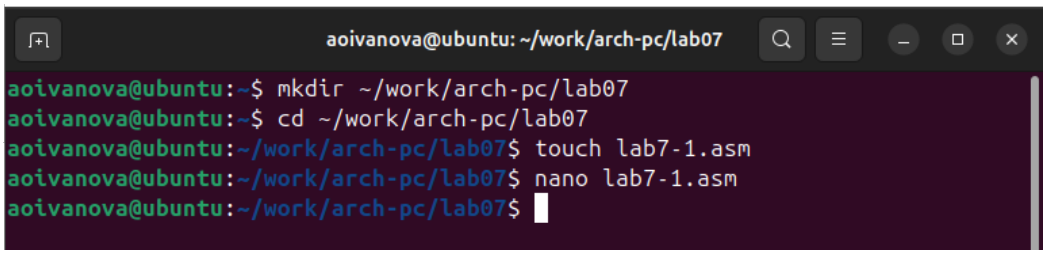

Рисунок 2.1: Создание каталога и файла

Рассмотрели пример программы с использованием инструкции jmp. Ввели в файл lab7-1.asm текст программы из первого листинга

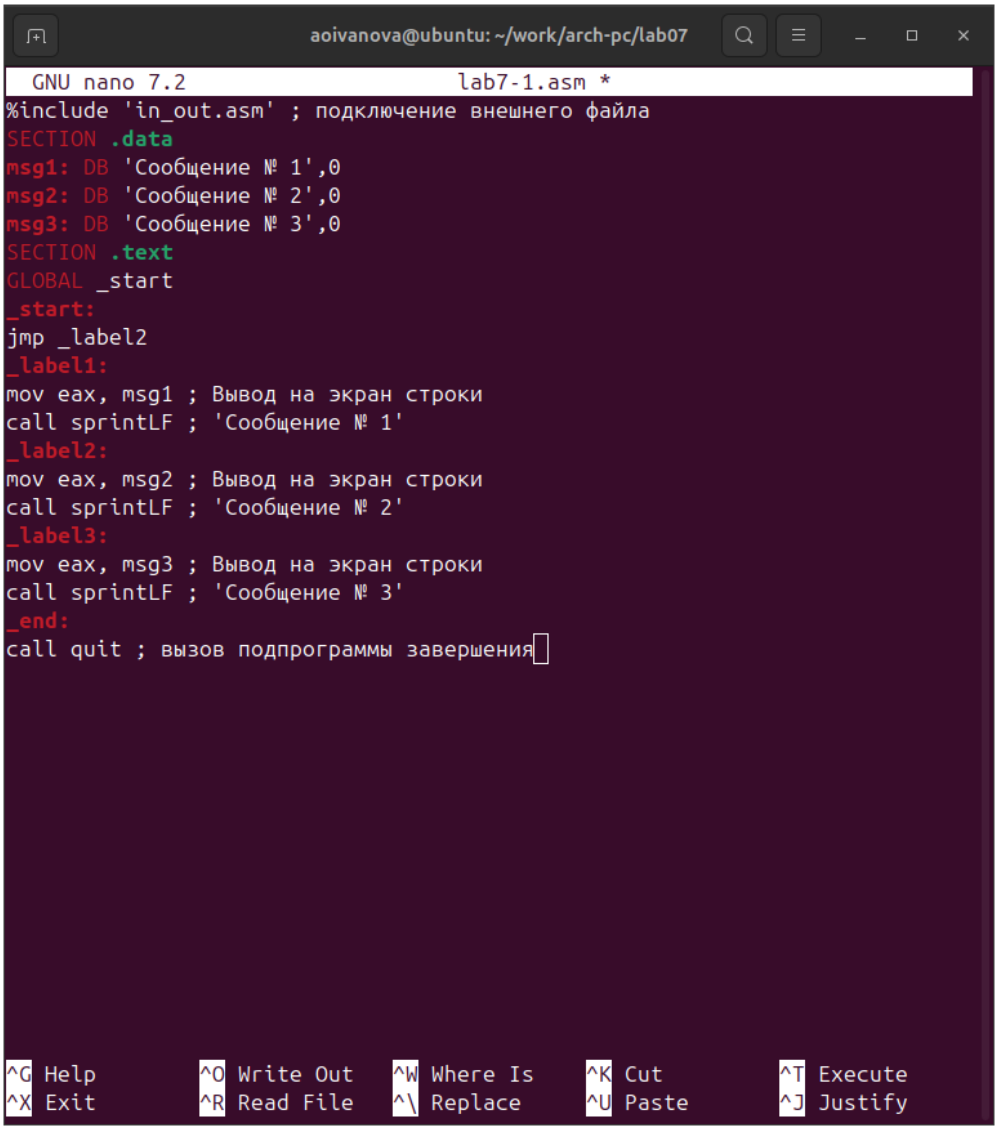

Рисунок 2.2: Измененный файл lab7-1.asm

Создали исполняемый файл и запустили его. Получили корректный результат работы данной программы

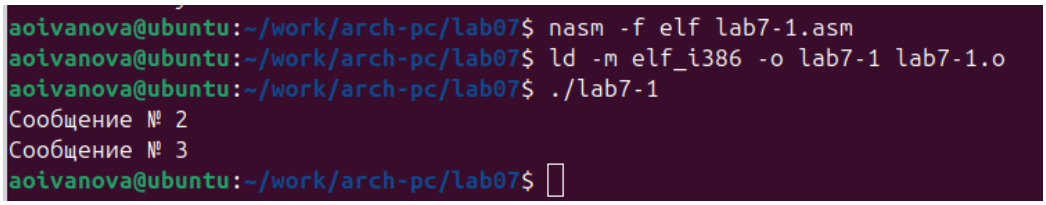

Рисунок 2.3: Создание исполняемого файла и вывод его работы

Далее изменили программу таким образом, чтобы она выводила сначала „Сообщение № 2“, потом „Сообщение № 1“ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавили инструкцию jmp с меткой _label1 (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавим инструкцию jmp с меткой _end (т.е. переход к инструкции call quit). Измените текст программы в соответствии со вторым листингом

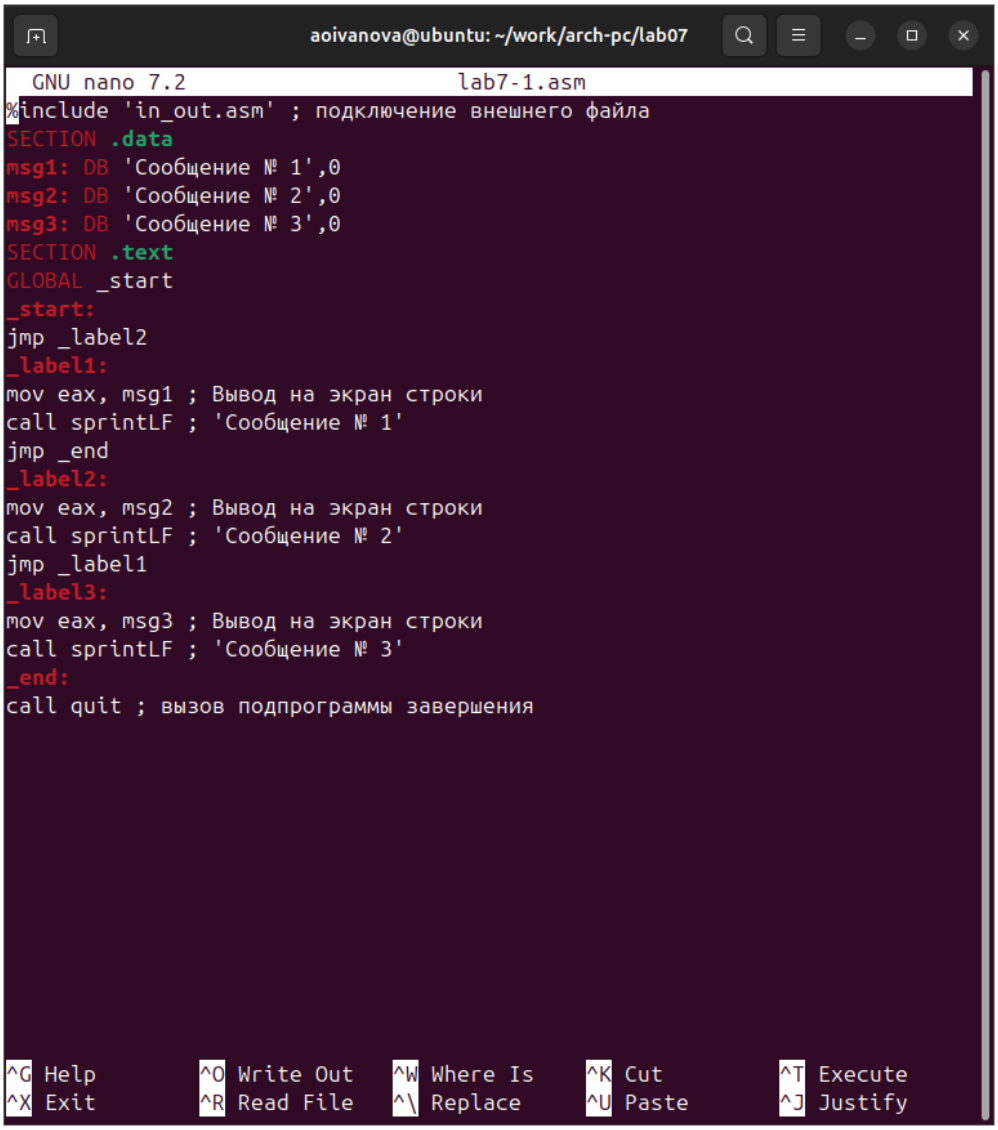

Рисунок 2.4: Измененный текст программы

Создали исполняемый файл и проверили его работу

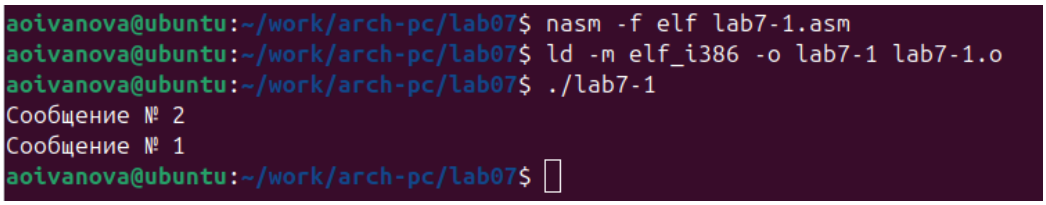

Рисунок 2.5: Создание исполняемого файла и вывод его работы

Далее изменили текст программы добавив или изменив инструкции jmp, чтобы вывод программы был следующим: Сообщение № 3 Сообщение № 2 Сообщение № 1

Листинг 1:

```
%include „in_out.asm“ ; подключение внешнего файла
SECTION .data
msg1: DB „Сообщение № 1“,0
msg2: DB „Сообщение № 2“,0
msg3: DB „Сообщение № 3“,0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintLF ; „Сообщение № 1“
jmp _end _label2: mov eax, msg2 ; Вывод на экран строки
call sprintLF ; „Сообщение № 2“
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; „Сообщение № 3“
jmp _label2
_end:
call quit ; вызов подпрограммы завершения
```

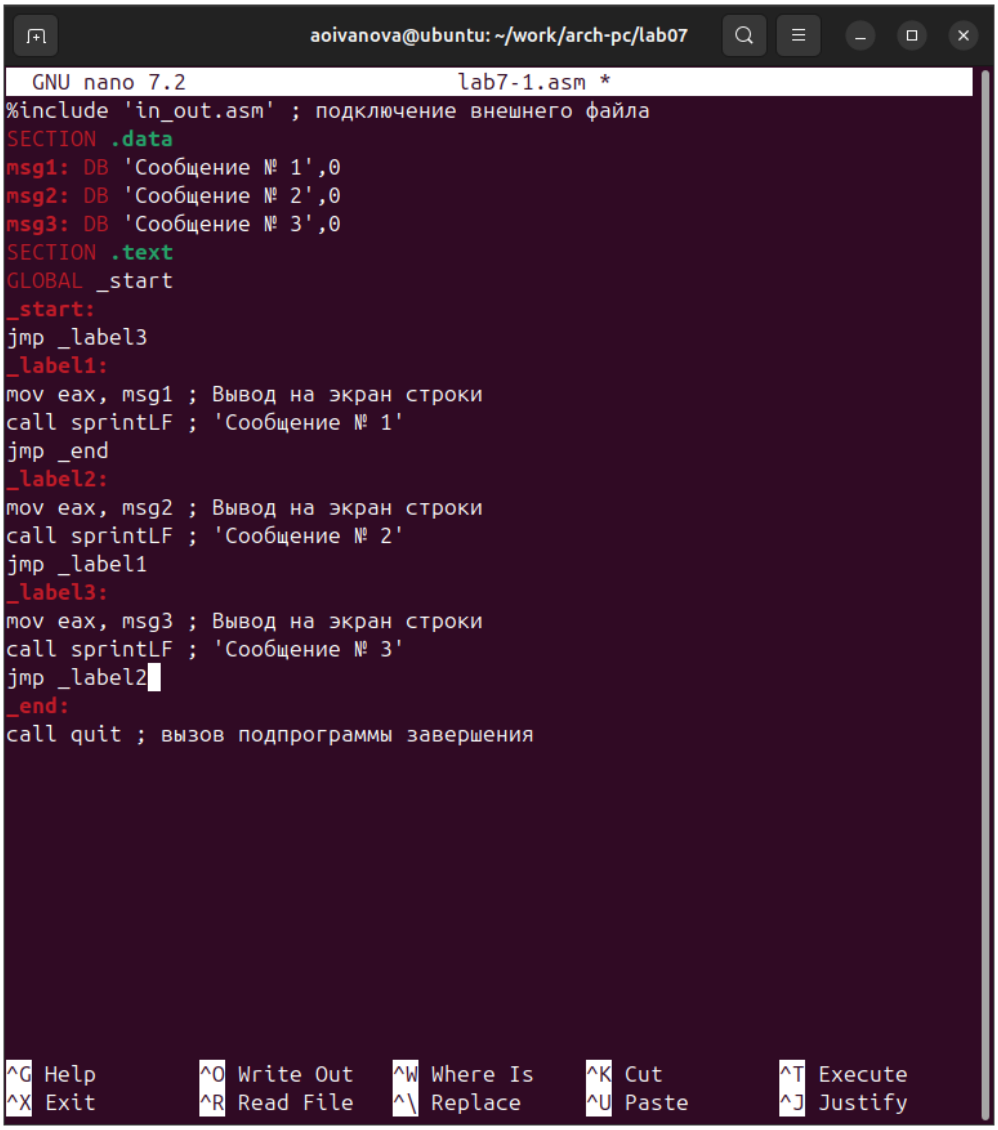

Рисунок 2.6: Измененный текст программы

Создали исполняемый файл и проверили его работу

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

Рисунок 2.7: Создание исполняемого файла и вывод его работы

Создали файл lab7-2.asm в каталоге ~/work/arch-pc/lab06 и ввели в него текст программы из второго листинга

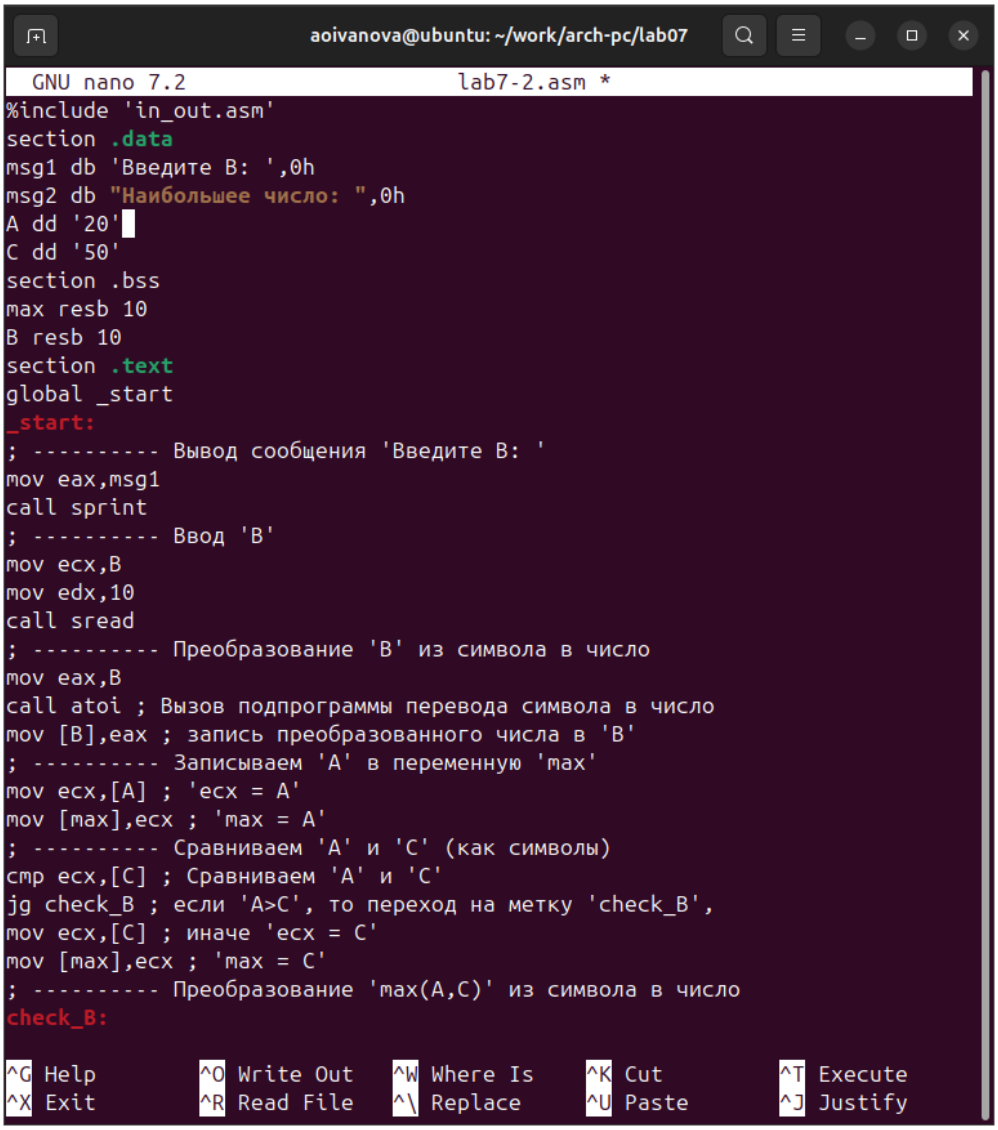

Рисунок 2.8: Создание файла lab7-2.asm

Создали исполняемый файл и проверили его работу для разных значений B.

```
aoivanova@ubuntu:~/work/arch-pc/lab07$ nano lab7-2.asm
aoivanova@ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
aoivanova@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
aoivanova@ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 3
Наибольшее число: 50
aoivanova@ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 8
Наибольшее число: 50
aoivanova@ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 1
Наибольшее число: 50
aoivanova@ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 128
Наибольшее число: 128
```

Рисунок 2.9: Создание исполняемого файла и вывод его работы

2.1.2 Изучение структуры файлы листинга

Создали файл листинга для программы из файла lab7-2.asm

```
aoivanova@ubuntu:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
```

Рисунок 2.10: Создание листинга

Открыли файл листинга lab7-2.lst с помощью текстового редактора mcedit.

```
aoivanova@ubuntu: ~/work/arch-pc/lab07
/home/aoivanova/work/arch-pc/lab07/lab7-2.lst   [----]  0 L:[177+ 0 177/225] *(10921/14458b) 0032 0x020                [*][X]
    4 0000001C BED0BBD18CD188D0B5-
    4 00000025 D0B520D187D0B8D181-
    4 0000002E D0BBD0BE3A2000.....
    5 00000035 32300000                A dd '20'
    6 00000039 35300000                C dd '50'
    7                                  section .bss
    8 00000000 <res Ah>                max resb 10
    9 0000000A <res Ah>                B resb 10
   10                                  section .text
   11                                  global _start
   12                                  _start:
   13                                  ; ---------- Вывод сообщения 'Введите B: '
   14 000000E8 B8[00000000]            mov eax,msg1
   15 000000ED E81DFFFFFF              call sprint
   16                                  ; ---------- Ввод 'B'
   17 000000F2 B9[0A000000]            mov ecx,B
   18 000000F7 BA0A000000              mov edx,10
   19 000000FC E842FFFFFF              call sread
   20                                  ; ---------- Преобразование 'B' из символа в число
   21 00000101 B8[0A000000]            mov eax,B
   22 00000106 E891FFFFFF              call atoi ; Вызов подпрограммы перевода символа в число
   23 0000010B A3[0A000000]            mov [B],eax ; запись преобразованного числа в 'B'
   24                                  ; ---------- Записываем 'A' в переменную 'max'
   25 00000110 8B0D[35000000]          mov ecx,[A] ; 'ecx = A'
   26 00000116 890D[00000000]          mov [max],ecx ; 'max = A'
   27                                  ; ---------- Сравниваем 'A' и 'C' (как символы)
   28 0000011C 3B0D[39000000]          cmp ecx,[C] ; Сравниваем 'A' и 'C'
   29 00000122 7F0C                    jg check_B ; если 'A>C', то переход на метку 'check_B',
   30 00000124 8B0D[39000000]          mov ecx,[C] ; иначе 'ecx = C'
   31 0000012A 890D[00000000]          mov [max],ecx ; 'max = C'
   32                                  ; ---------- Преобразование 'max(A,C)' из символа в число
   33                                  check_B:
   34 00000130 B8[00000000]            mov eax,max
   35 00000135 E862FFFFFF              call atoi ; Вызов подпрограммы перевода символа в число
   36 0000013A A3[00000000]            mov [max],eax ; запись преобразованного числа в `max`
1Help  2Save  3Mark  4Replac  5Copy  6Move  7Search  8Delete  9PullDn  10Quit
```

Рисунок 2.11: Открытый листинг

Внимательно ознакомились с его форматом и содержимым. Опишем строки под номерами 14, 15 и 17.

1) 14 (номер строки); 000000E8 (адрес, начинается по смещению 000000E8 в сегменте кода); B8[00000000] (машинный код); mov eax,msg1 (исходный текст программы, в котором мы перемещаем адрес метки msg1 в регистр eax. msg1 - адрес строки, которую мы хотим вывести)

2) 15 (номер строки); 000000ED (адрес, начинается по смещению 000000ED в сегменте кода); E81DFFFFFF (машинный код); call sprint (исходный текст программы, в котором мы вызываем подпрограмму печати сообщения)

3) 17 (номер строки); 000000F2 (адрес, начинается по смещению 000000F2 в сегменте кода); B9[0A000000] (машинный код); mov ecx,B (исходный текст программы, в котором мы записываем адрес введённой переменной в ecx)

Открыли файл с программой lab7-2.asm и удалили в инструкции mov ecx,B операнд B

```
; ---------- Ввод 'B'
mov ecx
mov edx,10
call sread
```

Рисунок 2.12: Удаление в инструкции mov ecx,B операнд B

Выполнили трансляцию с получением файла листинга. Создался исполнительный файл и файл листинга («lab7-2» и «lab7-2.lst»), но при создании в терминале вывело ошибку

```
aoivanova@ubuntu:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:17: error: invalid combination of opcode and operands
aoivanova@ubuntu:~/work/arch-pc/lab07$
```

Рисунок 2.13: Ошибка в терминале

Также ошибка отобразилась в листинге

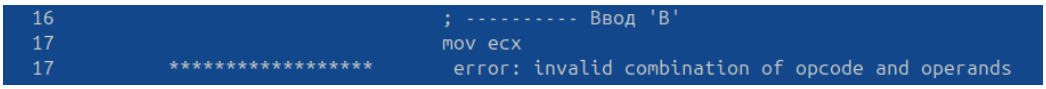

Рисунок 2.14: Ошибка в листинге

2.2 Задание для самостоятельной работы

1. Написали программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбpakb из таблицы в соответствии с 19 вариантом, то есть a=46, b=32, c=74

Листинг 2

```
%include „in_out.asm“
section .data
a dd 46
b dd 32
c dd 74
msg db «Наименьшее число:»,0h
section .bss
min resb 10
section .text
global _start
_start:
mov eax, [a]
mov ebx, [b]
mov ecx, [c]
cmp eax, ebx
jl check_c
mov eax, ebx
check_c:
```

cmp eax, ecx

jl save_result

mov eax, ecx

save_result:

mov [min], eax

mov eax, msg

call sprint

mov eax, [min]

call iprintLF

call quit

```
aoivanova@ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
aoivanova@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
aoivanova@ubuntu:~/work/arch-pc/lab07$ ./lab7-3
Наименьшее число: 32
```

Рисунок 2.15: Работа созданного нами файла

Как можно заметить, программа корректно считает наименьшее число

2. Написали программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции f(x) и выводит результат вычислений. 19 вариант : если x>a f(x)=a+x, если x<=a f(x)=x

Листинг 3

%include „in_out.asm“

section .data

msgx db „Введите x =“,0h

msga db „Введите a =“,0h

msgresult db „Результат f(x) =“,0h

section .bss

x resb 10

```
a resb 10
result resb 10
    section .text
global _start
    _start:
mov eax, msgx
call sprint
mov ecx, x
mov edx, 10
call sread mov eax, x
call atoi
mov [x], eax
mov eax, msga
call sprint
mov ecx, a
mov edx, 10
call sread mov eax, a
call atoi
mov [a], eax
mov ebx, [x]
mov ecx, [a]
cmp ebx, ecx
jg greater_case
mov [result], ebx
jmp output_result
    greater_case:
add ecx, ebx
mov [result], ecx
    output_result:
```

mov eax, msgresult

call sprint

mov eax, [result]

call iprintLF

call quit

Создали исполняемый файл и проверили его работу для значений x и a

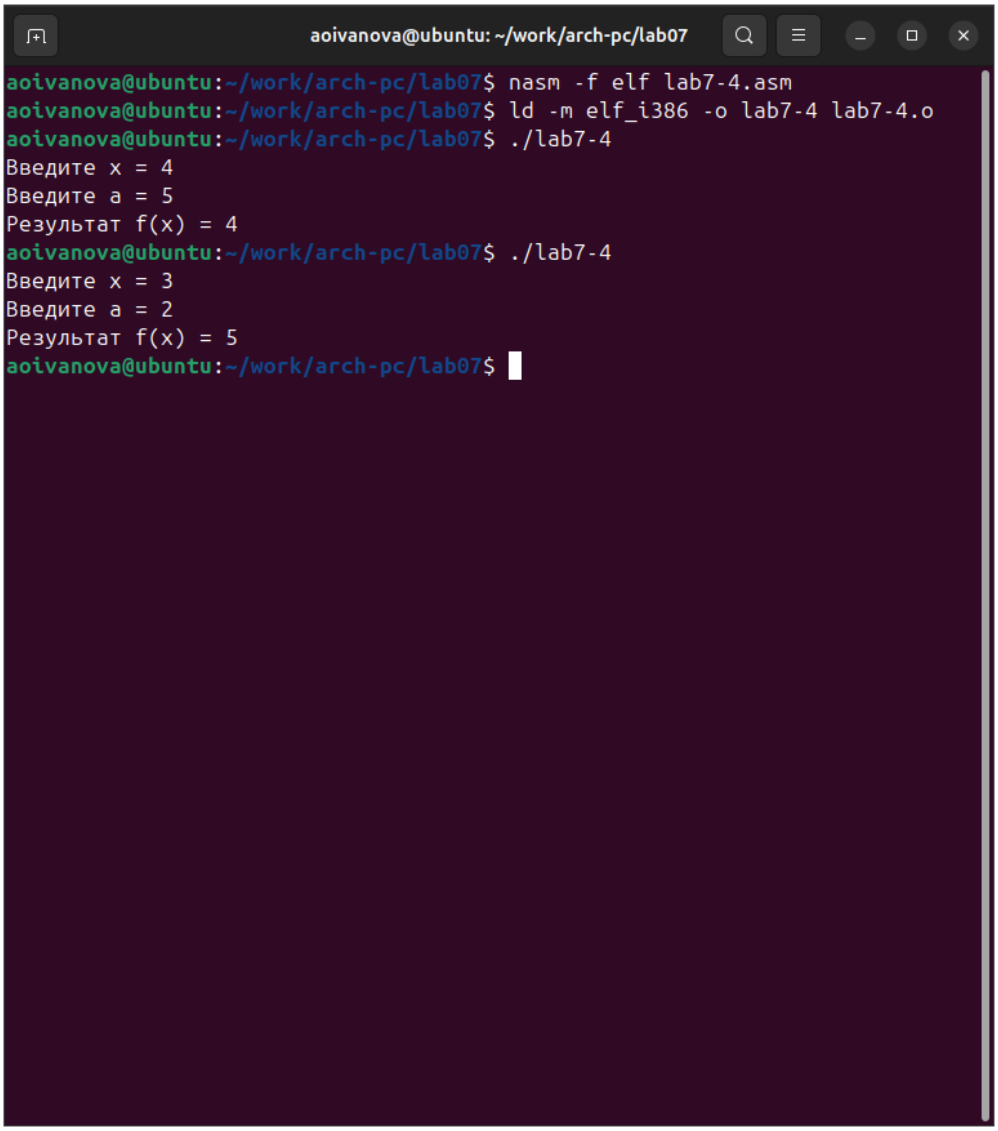

Рисунок 2.16: Работа созданного нами файла

3 Выводы

Изучили команды условного и безусловного переходов, а также приобрели навыки написания программ с использованием переходов и познакомились с назначением и структурой файла листинга.