

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
Факультет физико-математических и естественных наук

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7

Студент: Белакова Полина Вячеславовна

Ст.билет: 1032252589

Группа: НКАбд-01-25

МОСКВА

2025 г

Цель работы

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

Порядок выполнения лабораторной работы

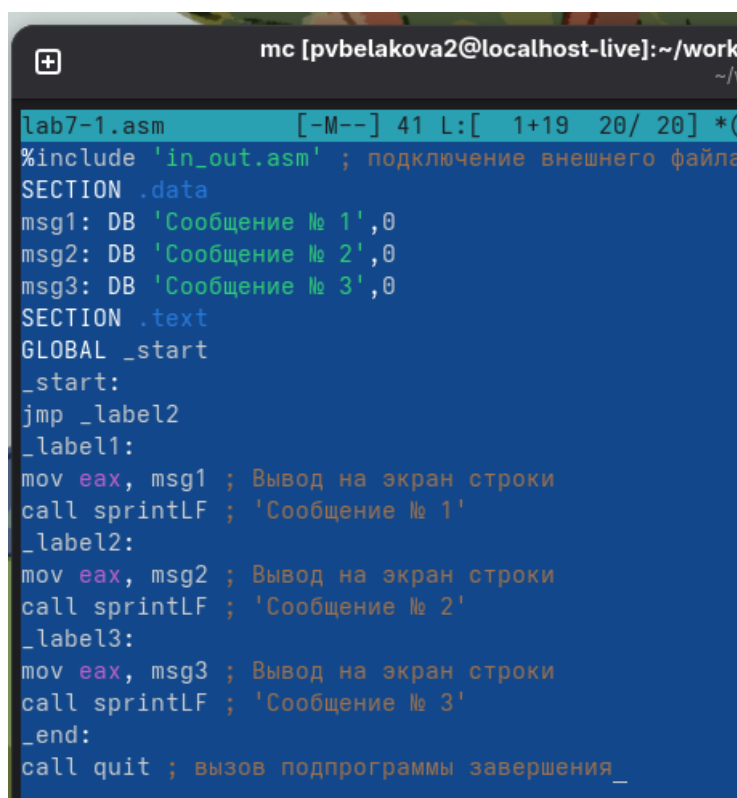
Реализация переходов в NASM

1. Создаю каталог для программ лабораторной работы № 7, перехожу в него и создаю файл lab7-1.asm (рисунок 1).

```
pvelakova2@localhost-live:~$ mkdir ~/work/arch-pc/lab07
pvelakova2@localhost-live:~$ cd ~/work/arch-pc/lab07
pvelakova2@localhost-live:~/work/arch-pc/lab07$ touch lab7-1.asm
pvelakova2@localhost-live:~/work/arch-pc/lab07$
```

Рисунок 1 - создание папки lab07, перемещение в нее и создание файла lab7-1.asm.

2. Ввожу в файл lab7-1.asm текст программы из листинга 7.1 (рисунок 2).



```
lab7-1.asm [-M--] 41 L: [ 1+19 20/ 20] *
%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 sprintf ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения_
```

Рисунок 2 - код листинга 7.1

Создаю исполняемый файл и запускаю его (рисунок 3).

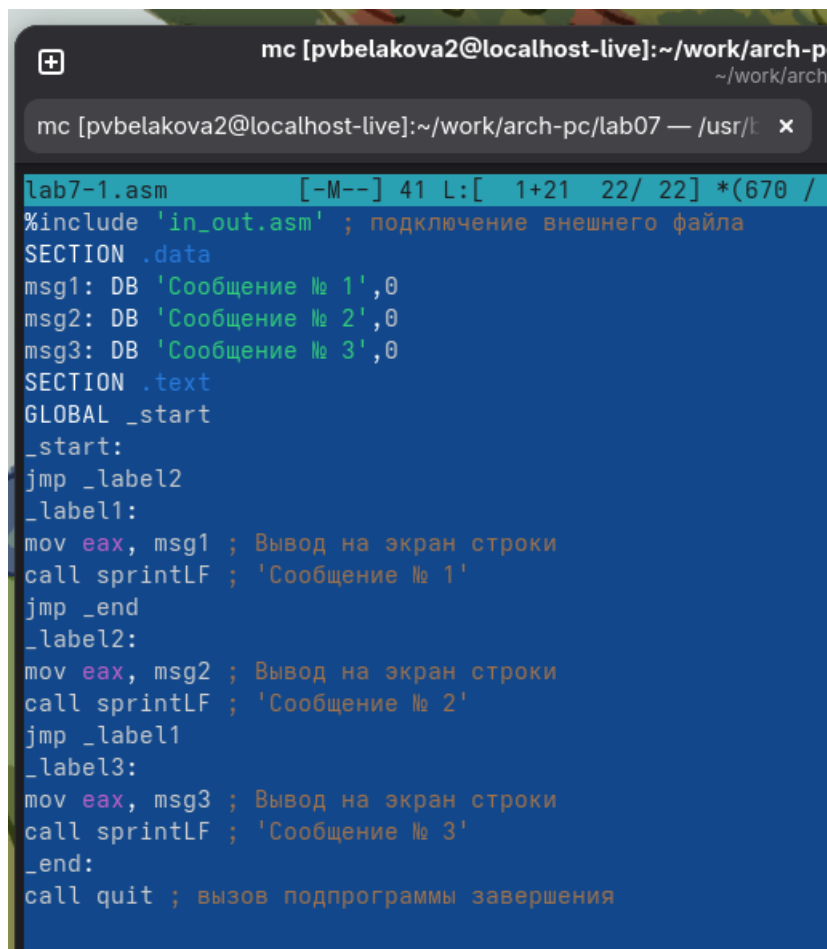
```

pvbelakova2@localhost-live:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
pvbelakova2@localhost-live:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
pvbelakova2@localhost-live:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
pvbelakova2@localhost-live:~/work/arch-pc/lab07$

```

Рисунок 3 - создание исполняемого файла lab7-1 и его запуск.

Изменяю текст программы в соответствии с листингом 7.2 (рисунок 4).



```

lab7-1.asm      [-M--] 41 L:[ 1+21 22/ 22] *(670 /
%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 sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения

```

Рисунок 4 - код листинга 7.2.

Создаю исполняемый файл и проверяю его работу (рисунок 5).

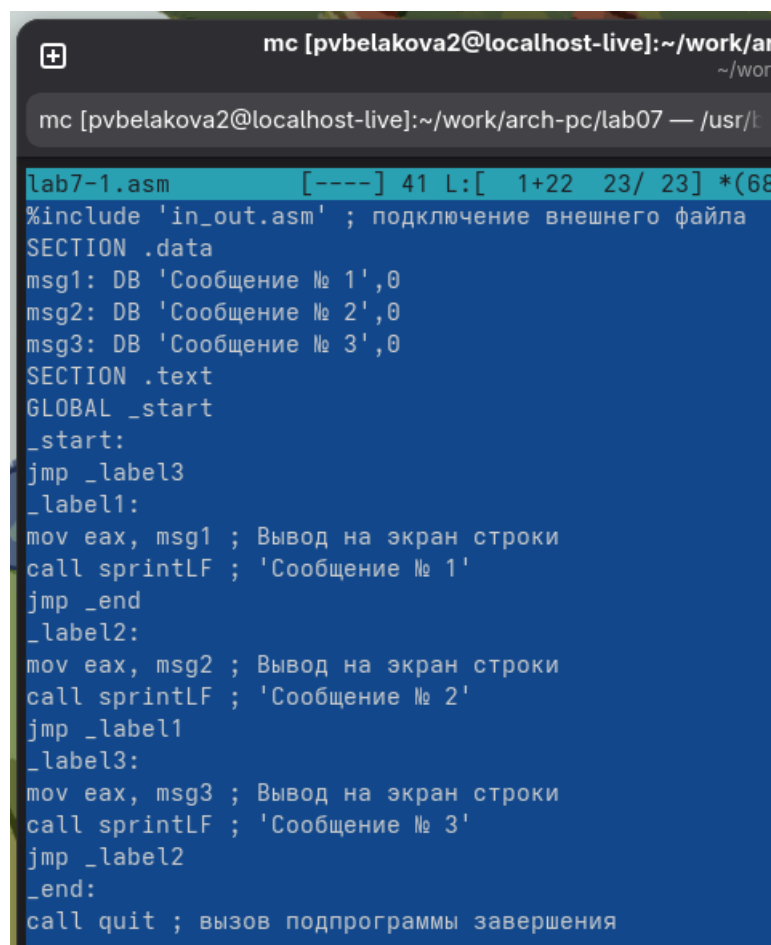
```

pvbelakova2@localhost-live:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm -o lab7-1_2.o
pvbelakova2@localhost-live:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1_2 lab7-1_2.o
pvbelakova2@localhost-live:~/work/arch-pc/lab07$ ./lab7-1_2
Сообщение № 2
Сообщение № 1
pvbelakova2@localhost-live:~/work/arch-pc/lab07$

```

Рисунок 5 - создание исполняемого файла lab7-1_2 и его запуск.

Изменяю текст программы добавив или изменив инструкции jmp (рисунок 6).



```

mc [pvbelakova2@localhost-live]:~/work/arch-pc/lab07
lab7-1.asm [----] 41 L:[ 1+22 23/ 23] *(68
%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 _label3
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения

```

Рисунок 6 - измененный код.

Создаю исполняемый файл и проверяю его работу (рисунок 7).

```
pvelakova2@localhost-live:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm -o lab7-1_3.o
pvelakova2@localhost-live:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1_3 lab7-1_3.o
pvelakova2@localhost-live:~/work/arch-pc/lab07$ ./lab7-1_3
Сообщение № 3
Сообщение № 2
Сообщение № 1
pvelakova2@localhost-live:~/work/arch-pc/lab07$
```

Рисунок 7 - создание исполняемого файла lab7-1_3 и его запуск.

3. Создаю файл lab7-2.asm (рисунок 8).

```
pvelakova2@localhost-live:~/work/arch-pc/lab07$ touch lab7-2.asm
```

Рисунок 8 - создание файла lab7-2.asm.

Ввожу в lab7-2.asm. код листинга 7.3. (рисунок 9).

```

mc [pvbelakova2@localhost-live]:
lab7-2.asm [-M--] 21 L: [ 1+ 8 1/ 50] *(21 /1773b) 8
%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
A dd '20'
C dd '50'
section .bss
max resb 10
B resb 10
Архитектура 386
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
str ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
str ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход

```

Рисунок 9 - код листинга 7.3.

Создаю исполняемый файл и проверяю его работу для разных значений B (рисунок 10).

```
pvtbelakova2@localhost-live:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
pvtbelakova2@localhost-live:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
pvtbelakova2@localhost-live:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 2
Наибольшее число: 50
pvtbelakova2@localhost-live:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 100
Наибольшее число: 100
pvtbelakova2@localhost-live:~/work/arch-pc/lab07$ ./lab7-2
Введите В: -10
Наибольшее число: 50
pvtbelakova2@localhost-live:~/work/arch-pc/lab07$ █
```

Рисунок 10 - создание исполняемого файла lab7-1_3 и его запуск с разными значениями В.

Изучение структуры файлы листинга

4. Создаю файл листинга для программы из файла lab7-2.asm (рисунок 11).

```
pvtbelakova2@localhost-live:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
```

Рисунок 11 - создание файла листинга.

Открываю файл листинга lab7-2.lst (рисунок 12).


```
mc [pvbelakova2@localhost-live]:~/work/arch-pc/lab07 — /usr/bin/mc -P /tmp/mc.pwd.nfwlv
~/work/arch-pc/lab07
mc [pvbelakova2@localhost-live]:~/work/arch-pc/lab07 — /usr/
pvbelakova2@localhost-live:~/work/a

lab7-2.lst [----] 61 L:[ 1+ 0 1/225] *(61 /14458b) 0010 0x00A
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 E8E8FFFFFF <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
39 0000002B 5A <1> pop edx
40 0000002C C3 <1> ret
41 <1>.....
```

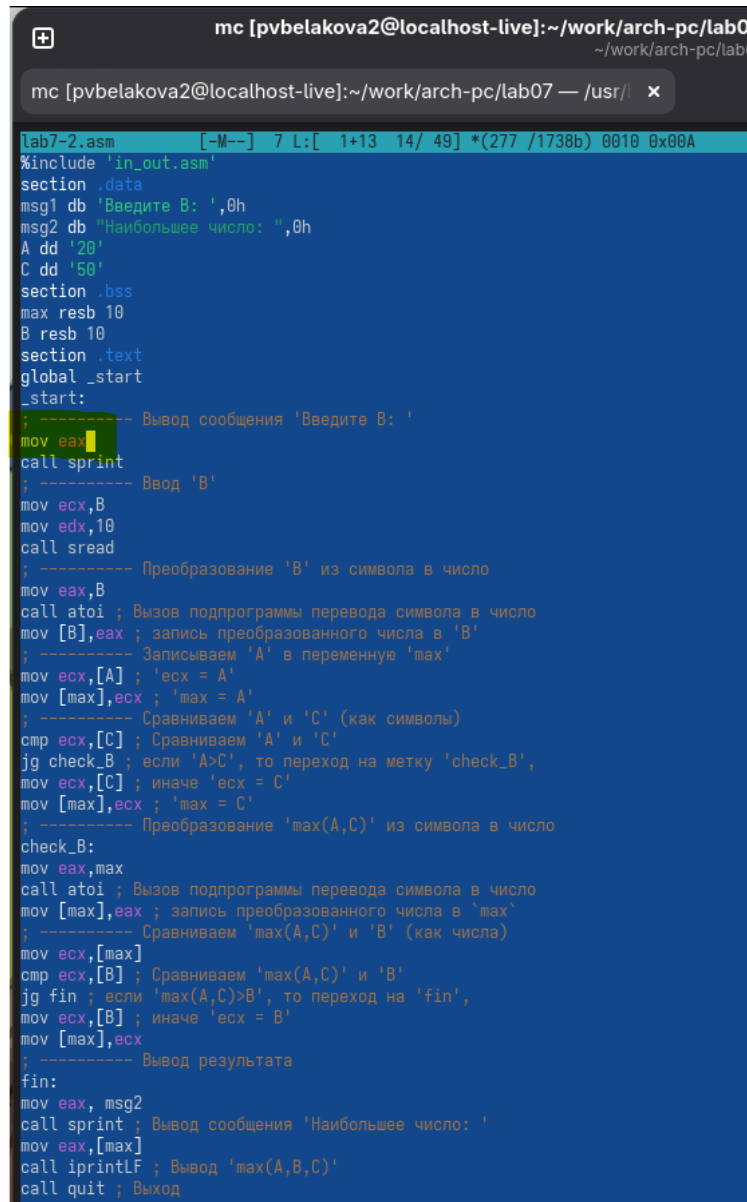
Рисунок 12 - часть кода файла листинга.

Строка 4 - первая инструкция в функции slen, сохраняет значение указанного регистра на вершине стека. Стек используется для временного хранения данных и сохранения состояния регистров.

Строка 10 - Эта инструкция находится внутри цикла nextchar в функции slen, inc увеличивает значение указанного операнда на единицу.

Строка 23 - первая инструкция в функции sprint, инструкция push сохраняет значение указанного регистра на вершине стека.

Открываю файл с программой lab7-2.asm и в инструкции mov eax, msg1 удаляю операнд msg1 (рисунок 13).



```
lab7-2.asm [-M--] 7 L: [ 1+13 14/ 49] *(277 /1738b) 0010 0x00A
#include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
A dd '20'
C dd '50'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход
```

Рисунок 13 - код файла lab7-2.asm без операнда msg1.

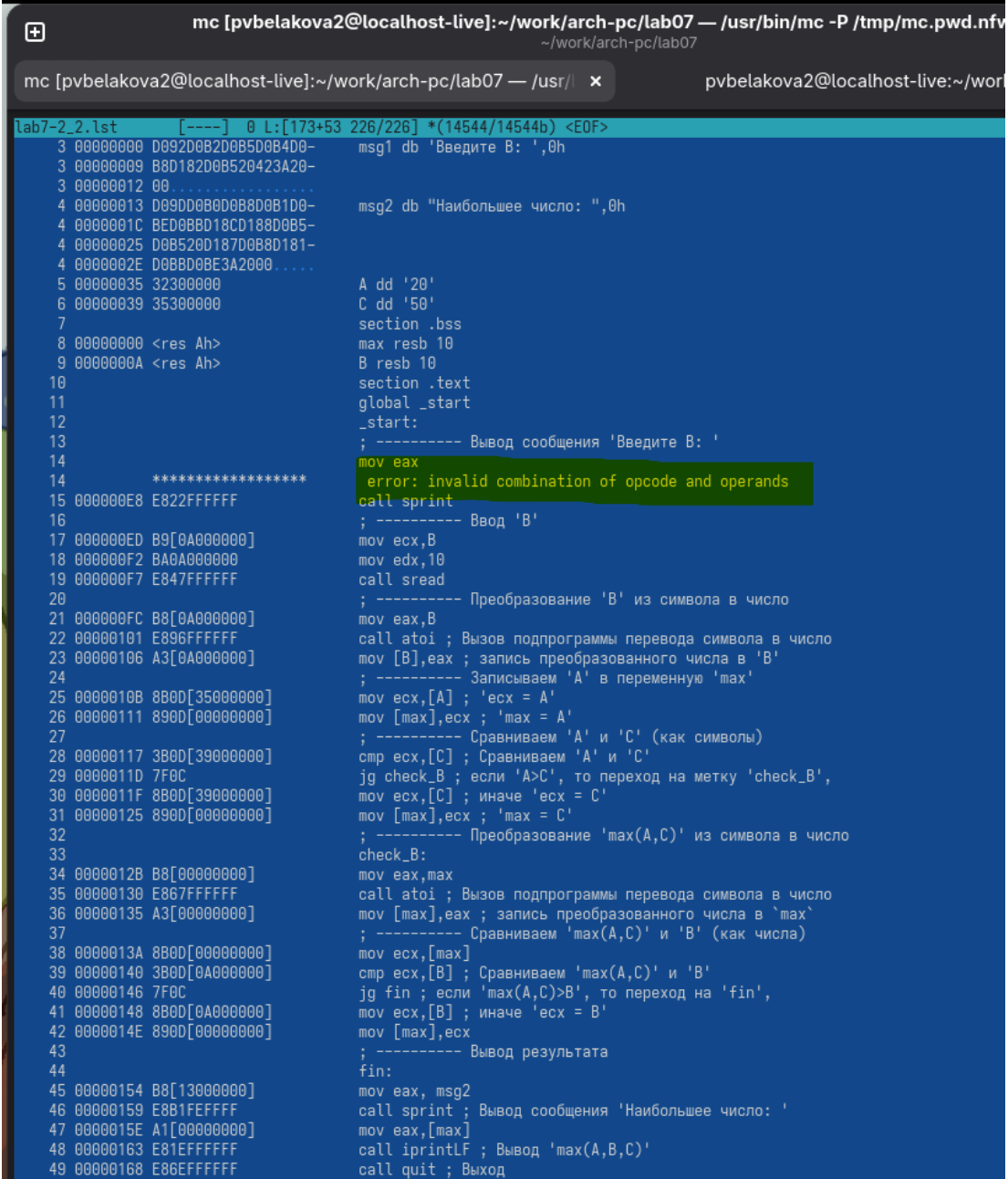
Выполняю трансляцию с получением файла листинга (рисунок 14).

```
pvbelakova2@localhost-live:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2_2.lst lab7-2.asm
lab7-2.asm:14: error: invalid combination of opcode and operands
pvbelakova2@localhost-live:~/work/arch-pc/lab07$
```

Рисунок 14 - получение файла листинга.

Какие выходные файлы создаются в этом случае? Что добавляется в листинге?

В этом случае, так как код содержит синтаксическую ошибку, создается только файл листинга lab7-2_2.lst. В листинг файле появляется указание ошибки (рисунок 15).



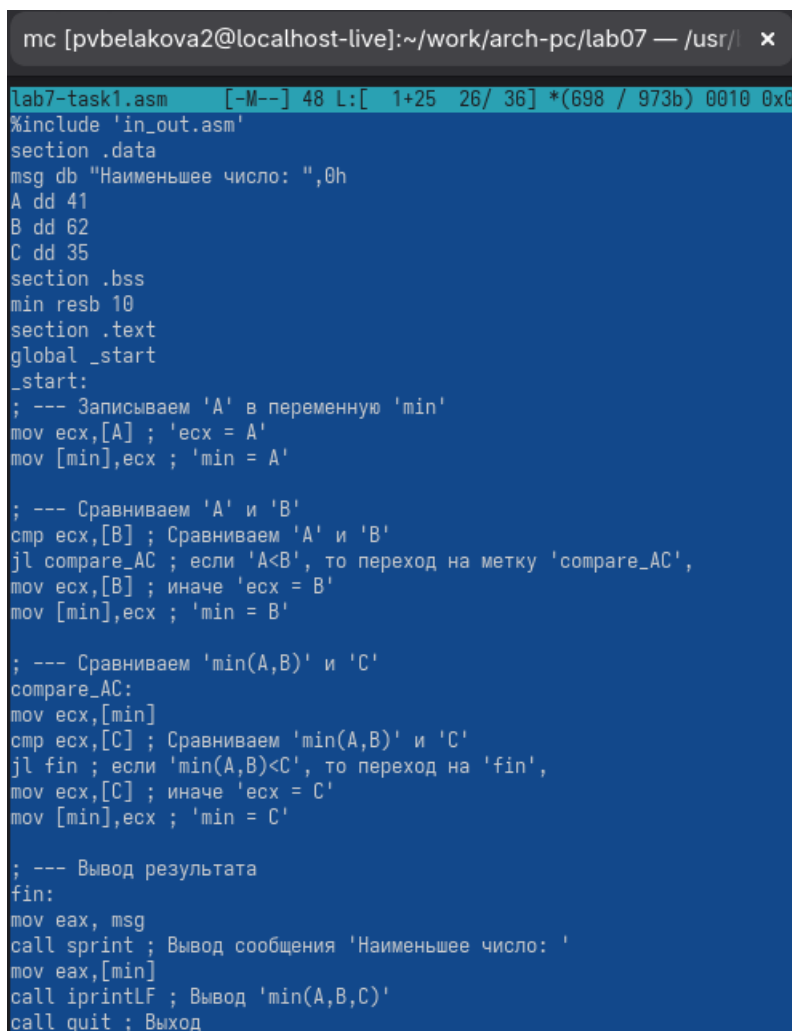
```
mc [pvbelakova2@localhost-live]:~/work/arch-pc/lab07 — /usr/bin/mc -P /tmp/mc.pwd.nfv
~/work/arch-pc/lab07
mc [pvbelakova2@localhost-live]:~/work/arch-pc/lab07 — /usr/ x pvbelakova2@localhost-live:~/wor

lab7-2_2.lst [----] 0 L:[173+53 226/226] *(14544/14544b) <EOF>
3 00000000 0092D0B2D0B5D0B4D0- msg1 db 'Введите B: ',0h
3 00000009 B8D182D0B520423A20-
3 00000012 00.....
4 00000013 009DD0B0D0B8D0B1D0- msg2 db "Наибольшее число: ",0h
4 0000001C BED0BB018CD188D0B5-
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 mov eax
14 ***** error: invalid combination of opcode and operands
15 000000E8 E827FFFFFF call sprint
16 ; ----- Ввод 'B'
17 000000ED B9[0A000000] mov ecx,B
18 000000F2 BA0A000000 mov edx,10
19 000000F7 E847FFFFFF call sread
20 ; ----- Преобразование 'B' из символа в число
21 000000FC B8[0A000000] mov eax,B
22 00000101 E896FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
23 00000106 A3[0A000000] mov [B],eax ; запись преобразованного числа в 'B'
24 ; ----- Записываем 'A' в переменную 'max'
25 0000010B 8B0D[35000000] mov ecx,[A] ; 'ecx = A'
26 00000111 890D[00000000] mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 00000117 3B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 0000011D 7F0C jg check_B ; если 'A>C', то переход на метку 'check_B',
30 0000011F 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'
31 00000125 890D[00000000] mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 0000012B B8[00000000] mov eax,max
35 00000130 E867FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
36 00000135 A3[00000000] mov [max],eax ; запись преобразованного числа в 'max'
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013A 8B0D[00000000] mov ecx,[max]
39 00000140 3B0D[0A000000] cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 00000146 7F0C jg fin ; если 'max(A,C)>B', то переход на 'fin',
41 00000148 8B0D[0A000000] mov ecx,[B] ; иначе 'ecx = B'
42 0000014E 890D[00000000] mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 00000154 B8[13000000] mov eax,msg2
46 00000159 E8B1FFFFFF call sprint ; Вывод сообщения 'Наибольшее число: '
47 0000015E A1[00000000] mov eax,[max]
48 00000163 E81EFFFFFF call iprintLF ; Вывод 'max(A,B,C)'
49 00000168 E86EFFFFFF call quit ; Выход
```

Рисунок 15 - часть листинг файла с указание ошибки.

Задание для самостоятельной работы (вариант 10)

1. Программа для нахождения наименьшей из 3 целочисленных переменных a , b и c (рисунок 16).



```
mc [pvbelakova2@localhost-live]:~/work/arch-pc/lab07 — /usr/ x
lab7-task1.asm  [-M--] 48 L: [ 1+25 26/ 36] *(698 / 973b) 0010 0x0
#include 'in_out.asm'
section .data
msg db "Наименьшее число: ",0h
A dd 41
B dd 62
C dd 35
section .bss
min resb 10
section .text
global _start
_start:
; --- Записываем 'A' в переменную 'min'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'min = A'

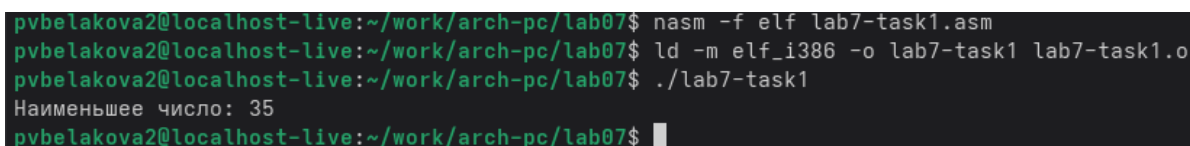
; --- Сравниваем 'A' и 'B'
cmp ecx,[B] ; Сравниваем 'A' и 'B'
jnl compare_AC ; если 'A<B', то переход на метку 'compare_AC',
mov ecx,[B] ; иначе 'ecx = B'
mov [min],ecx ; 'min = B'

; --- Сравниваем 'min(A,B)' и 'C'
compare_AC:
mov ecx,[min]
cmp ecx,[C] ; Сравниваем 'min(A,B)' и 'C'
jnl fin ; если 'min(A,B)<C', то переход на 'fin',
mov ecx,[C] ; иначе 'ecx = C'
mov [min],ecx ; 'min = C'

; --- Вывод результата
fin:
mov eax, msg
call sprint ; Вывод сообщения 'Наименьшее число: '
mov eax,[min]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход
```

Рисунок 16 - код программы для решения 10 варианта.

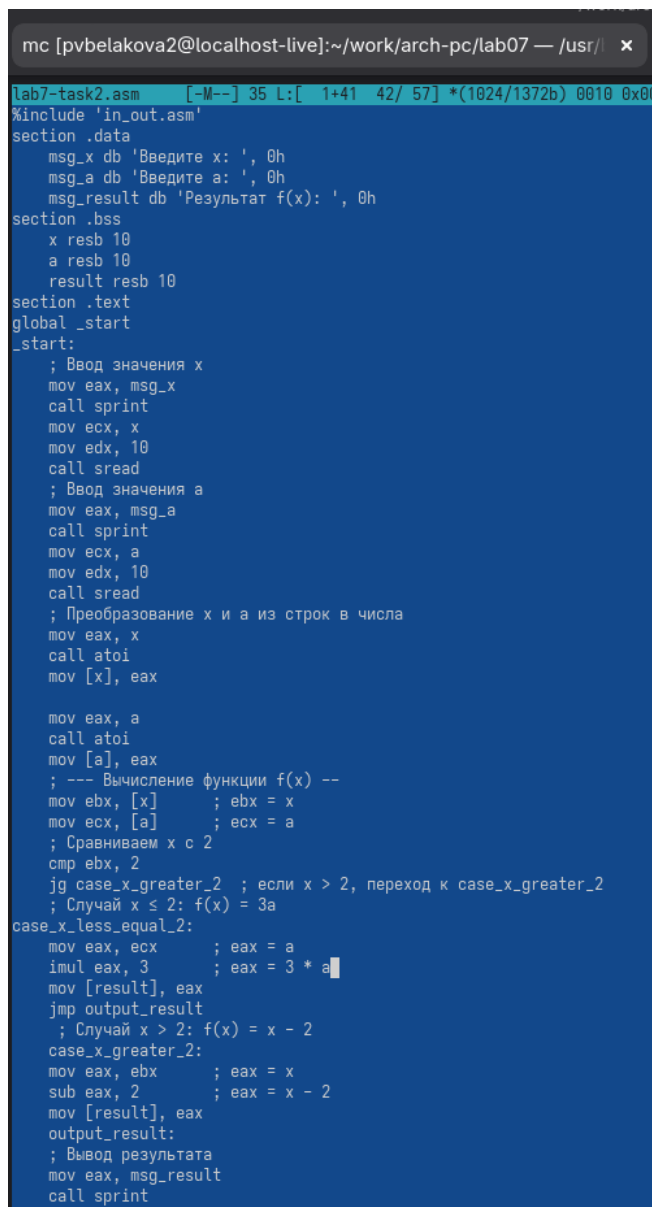
Создайте исполняемый файл и проверьте его работу (рисунок 17).



```
pvbelakova2@localhost-live:~/work/arch-pc/lab07$ nasm -f elf lab7-task1.asm
pvbelakova2@localhost-live:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-task1 lab7-task1.o
pvbelakova2@localhost-live:~/work/arch-pc/lab07$ ./lab7-task1
Наименьшее число: 35
pvbelakova2@localhost-live:~/work/arch-pc/lab07$
```

Рисунок 17 - создание исполняемого файла lab7-task и его запуск.

2. Программа, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений (рисунок 18).



```
mc [pvbelakova2@localhost-live]:~/work/arch-pc/lab07 — /usr/ x
lab7-task2.asm [-M--] 35 L: [ 1+41 42/ 57] *(1024/1372b) 0010 0x00
#include 'in_out.asm'
section .data
    msg_x db 'Введите x: ', 0h
    msg_a db 'Введите a: ', 0h
    msg_result db 'Результат f(x): ', 0h
section .bss
    x resb 10
    a resb 10
    result resb 10
section .text
global _start
_start:
    ; Ввод значения x
    mov eax, msg_x
    call sprint
    mov ecx, x
    mov edx, 10
    call sread
    ; Ввод значения a
    mov eax, msg_a
    call sprint
    mov ecx, a
    mov edx, 10
    call sread
    ; Преобразование x и a из строк в числа
    mov eax, x
    call atoi
    mov [x], eax

    mov eax, a
    call atoi
    mov [a], eax
    ; --- Вычисление функции f(x) ---
    mov ebx, [x] ; ebx = x
    mov ecx, [a] ; ecx = a
    ; Сравниваем x с 2
    cmp ebx, 2
    jg case_x_greater_2 ; если x > 2, переход к case_x_greater_2
    ; Случай x ≤ 2: f(x) = 3a
case_x_less_equal_2:
    mov eax, ecx ; eax = a
    imul eax, 3 ; eax = 3 * a
    mov [result], eax
    jmp output_result
    ; Случай x > 2: f(x) = x - 2
case_x_greater_2:
    mov eax, ebx ; eax = x
    sub eax, 2 ; eax = x - 2
    mov [result], eax
output_result:
    ; Вывод результата
    mov eax, msg_result
    call sprint
```

Рисунок 18 - часть кода программы.

Создаю исполняемый файл и проверяю его работу (рисунок 19).

```
pvtbelakova2@localhost-live:~/work/arch-pc/lab07$ nasm -f elf lab7-task2.asm
pvtbelakova2@localhost-live:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-task2 lab7-task2.o
pvtbelakova2@localhost-live:~/work/arch-pc/lab07$ ./lab7-task2
Введите x: 3
Введите a: 0
Результат f(x): 1
pvtbelakova2@localhost-live:~/work/arch-pc/lab07$ ./lab7-task2
Введите x: 1
Введите a: 2
Результат f(x): 6
pvtbelakova2@localhost-live:~/work/arch-pc/lab07$ █
```

Рисунок 19 - создание исполняемого файла lab7-task2 и его запуск со значениями из таблицы.

Выводы

В ходе лабораторной работы были изучены команды условного и безусловного переходов. Приобретены навыки написания программ с использованием переходов. Изучено с назначение и структура файла листинга.