

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

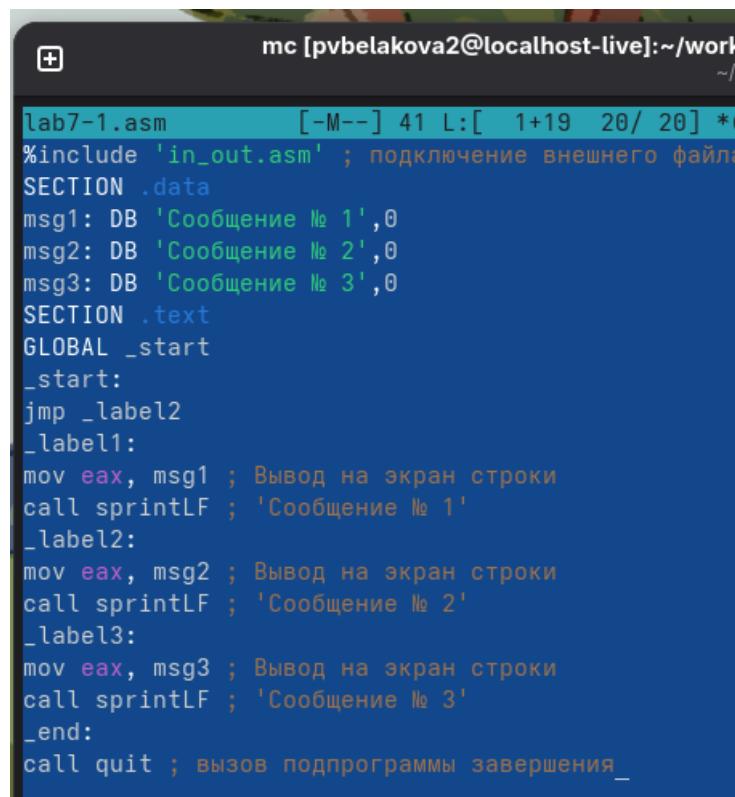
Реализация переходов в NASM

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

```
pvbelakova2@localhost-live:~$ mkdir ~/work/arch-pc/lab07
pvbelakova2@localhost-live:~$ cd ~/work/arch-pc/lab07
pvbelakova2@localhost-live:~/work/arch-pc/lab07$ touch lab7-1.asm
pvbelakova2@localhost-live:~/work/arch-pc/lab07$
```

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

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



```
mc [pvbelakova2@localhost-live]:~/work
~/work/arch-pc/lab07$ cat lab7-1.asm
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:
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения_
~/work/arch-pc/lab07$
```

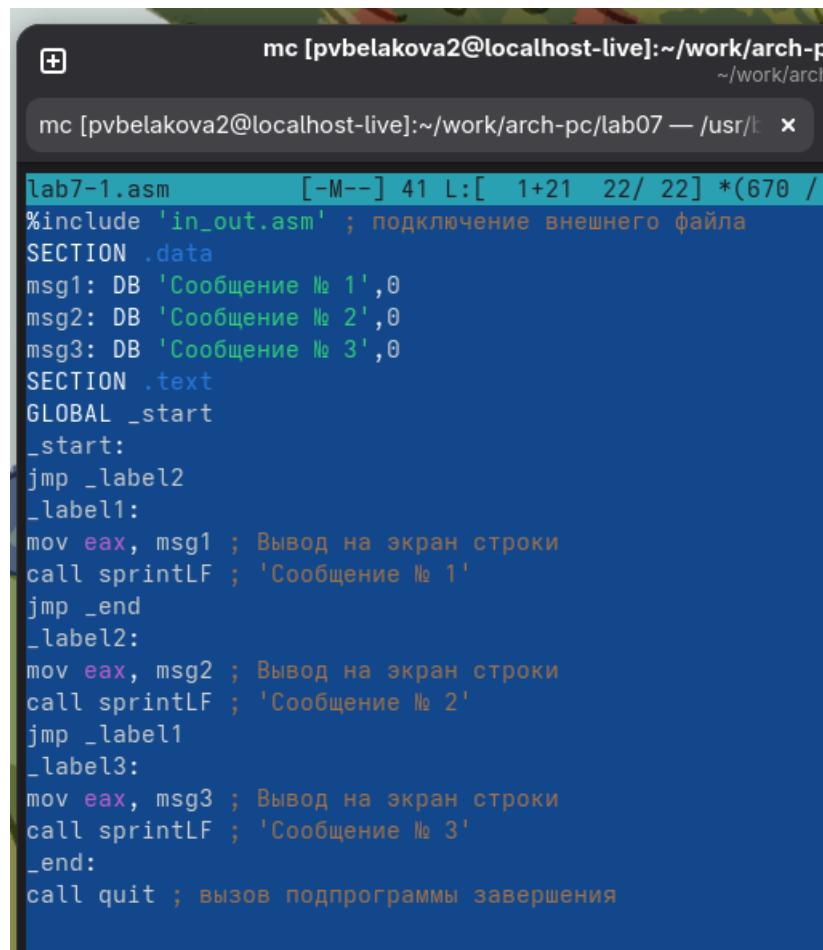
Рисунок 2 - код листинга 7.1

Создаю исполняемый файл и запускаю его (рисунок 3).

```
pbelakova2@localhost-live:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
pbelakova2@localhost-live:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
pbelakova2@localhost-live:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
pbelakova2@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).



```
mc [pbelakova2@localhost-live]:~/work/arch-pc/lab07 — /usr/b x
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 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'
_end:
call quit ; вызов подпрограммы завершения
```

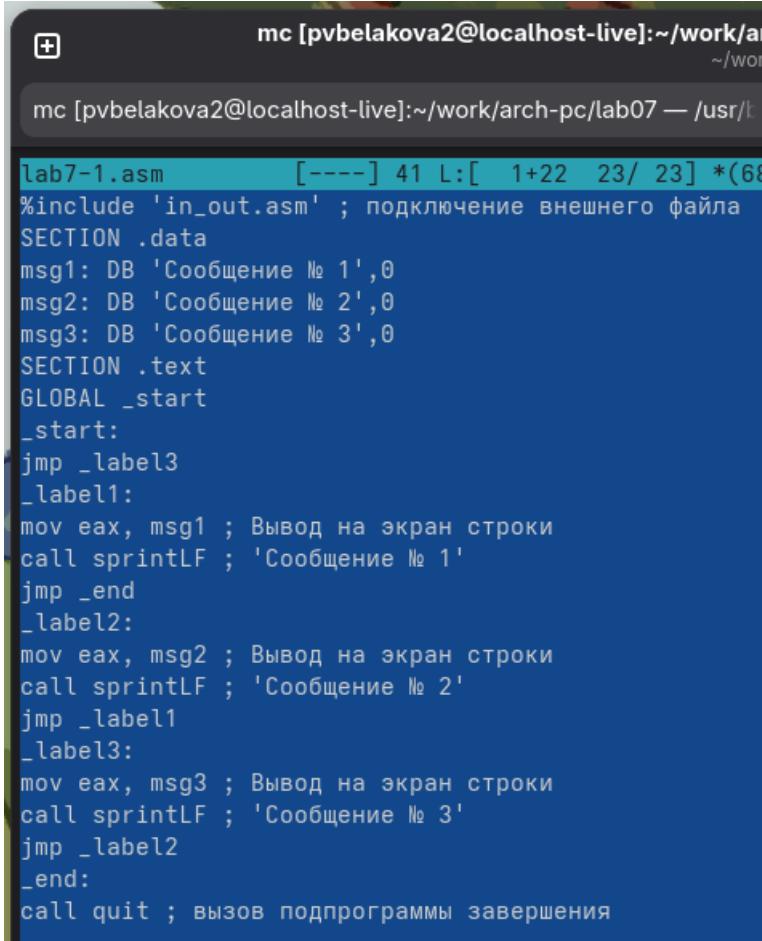
Рисунок 4 - код листинга 7.2.

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 — /usr/bin/mc
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 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 6 - измененный код.

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

```
pvbelakova2@localhost-live:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm -o lab7-1_3.o
pvbelakova2@localhost-live:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1_3 lab7-1_3.o
pvbelakova2@localhost-live:~/work/arch-pc/lab07$ ./lab7-1_3
Сообщение № 3
Сообщение № 2
Сообщение № 1
pvbelakova2@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).

```
pvbelakova2@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+ 0 1/ 50] *(21 /1773b) 0  
%include 'in_out.asm'  
section .data  
msg1 db 'Введите В: ',0h  
msg2 db "Наибольшее число: ",0h  
A dd '20'  
C dd '50'  
section .bss  
max resb 10  
B resb 10  
Architektura ZBM  
section .text  
global _start  
_start:  
; ----- Вывод сообщения 'Введите В: '  
mov eax,msg1  
call sprint  
; ----- Ввод 'В'  
mov ecx,B  
mov edx,10  
call sread  
; ----- Преобразование 'В' 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 9 - код листинга 7.3.

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

```
pvbelakova2@localhost-live:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
pvbelakova2@localhost-live:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
pvbelakova2@localhost-live:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 2
Наибольшее число: 50
pvbelakova2@localhost-live:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 100
Наибольшее число: 100
pvbelakova2@localhost-live:~/work/arch-pc/lab07$ ./lab7-2
Введите В: -10
Наибольшее число: 50
pvbelakova2@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).

```
pvbelakova2@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).

The screenshot shows a terminal window with the following details:

- Terminal title: mc [pvbelakova2@localhost-live]:~/work/arch-pc/lab07 — /usr/bin/mc -P /tmp/mc.pwd.nfwlv
- Terminal command: mc [pvbelakova2@localhost-live]:~/work/arch-pc/lab07 — /usr/| x
- Terminal prompt: pvbelakova2@localhost-live:~/work/a

The main content is a listing of assembly code from file `lab7-2.lst`. The code is annotated with comments and labels. Some lines are numbered (1, 2, 3, etc.). Key annotations include:

- `%include 'in_out.asm'` at line 1.
- `slen` label at line 2.
- `; Функция вычисления длины сообщения` comment at line 2.
- `slen:` label at line 3.
- `push ebx` at line 4.
- `mov ebx, eax` at line 5.
- `nextchar:` label at line 7.
- `cmp byte [eax], 0` at line 8.
- `jz finished` at line 9.
- `inc eax` at line 10.
- `jmp nextchar` at line 11.
- `finished:` label at line 13.
- `sub eax, ebx` at line 14.
- `pop ebx` at line 15.
- `ret` at line 16.
- `sprint` label at line 19.
- `; Функция печати сообщения` comment at line 20.
- `; входные данные: mov eax,<message>` comment at line 21.
- `sprint:` label at line 22.
- `push edx` at line 23.
- `push ecx` at line 24.
- `push ebx` at line 25.
- `push eax` at line 26.
- `call slen` at line 27.
- `mov edx, eax` at line 29.
- `pop eax` at line 30.
- `....` at line 31.
- `mov ecx, eax` at line 32.
- `mov ebx, 1` at line 33.
- `mov eax, 4` at line 34.
- `int 80h` at line 35.
- `....` at line 36.
- `pop ebx` at line 37.
- `pop ecx` at line 38.
- `pop edx` at line 39.
- `ret` at line 40.

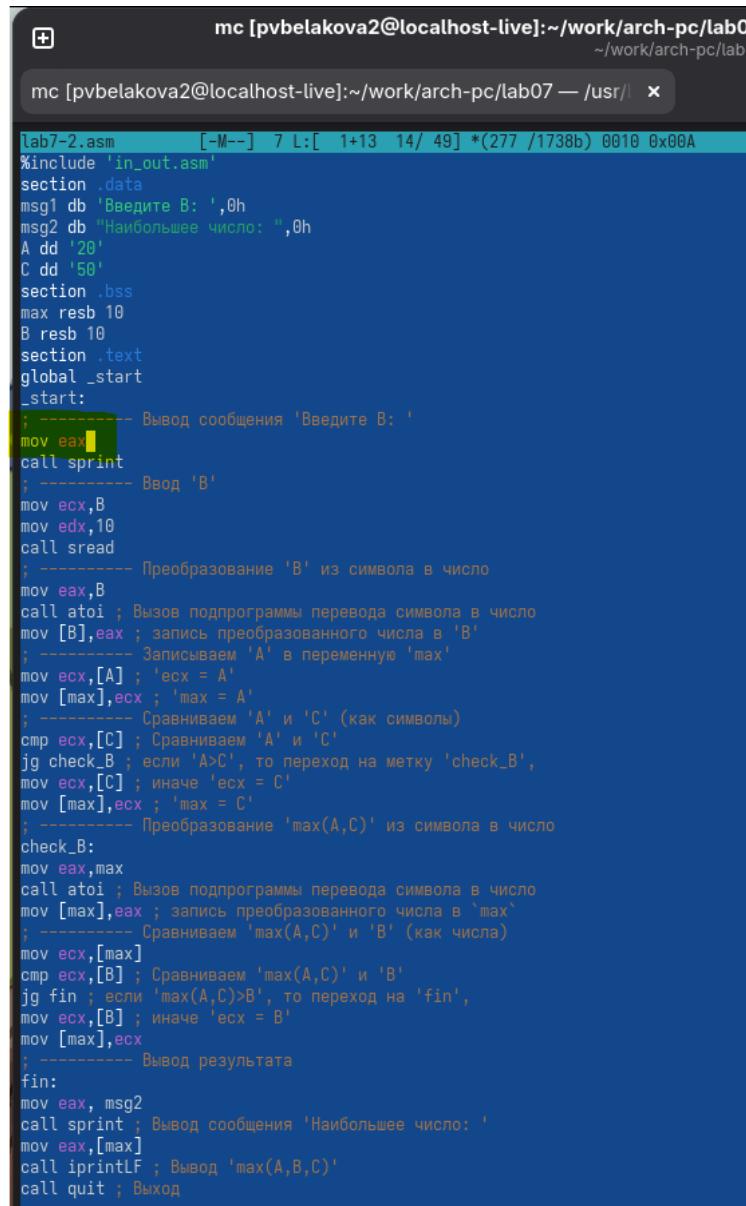
Рисунок 12 - часть кода файла листинга.

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 operand 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 В: ',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 В: '
mov eax,0
call sprint
; ----- Ввод 'В'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'В' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'В'
; ----- Записываем 'А' в переменную 'max'
mov ecx,[A] ; 'ecx = А'
mov [max],ecx ; 'max = А'
; ----- Сравниваем 'А' и 'С' (как символы)
cmp ecx,[C] ; Сравниваем 'А' и 'С'
jg check_B ; если 'А>С', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = С'
mov [max],ecx ; 'max = С'
; ----- 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 'В' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'В'
jg fin ; если 'max(A,C)>В', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = В'
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.

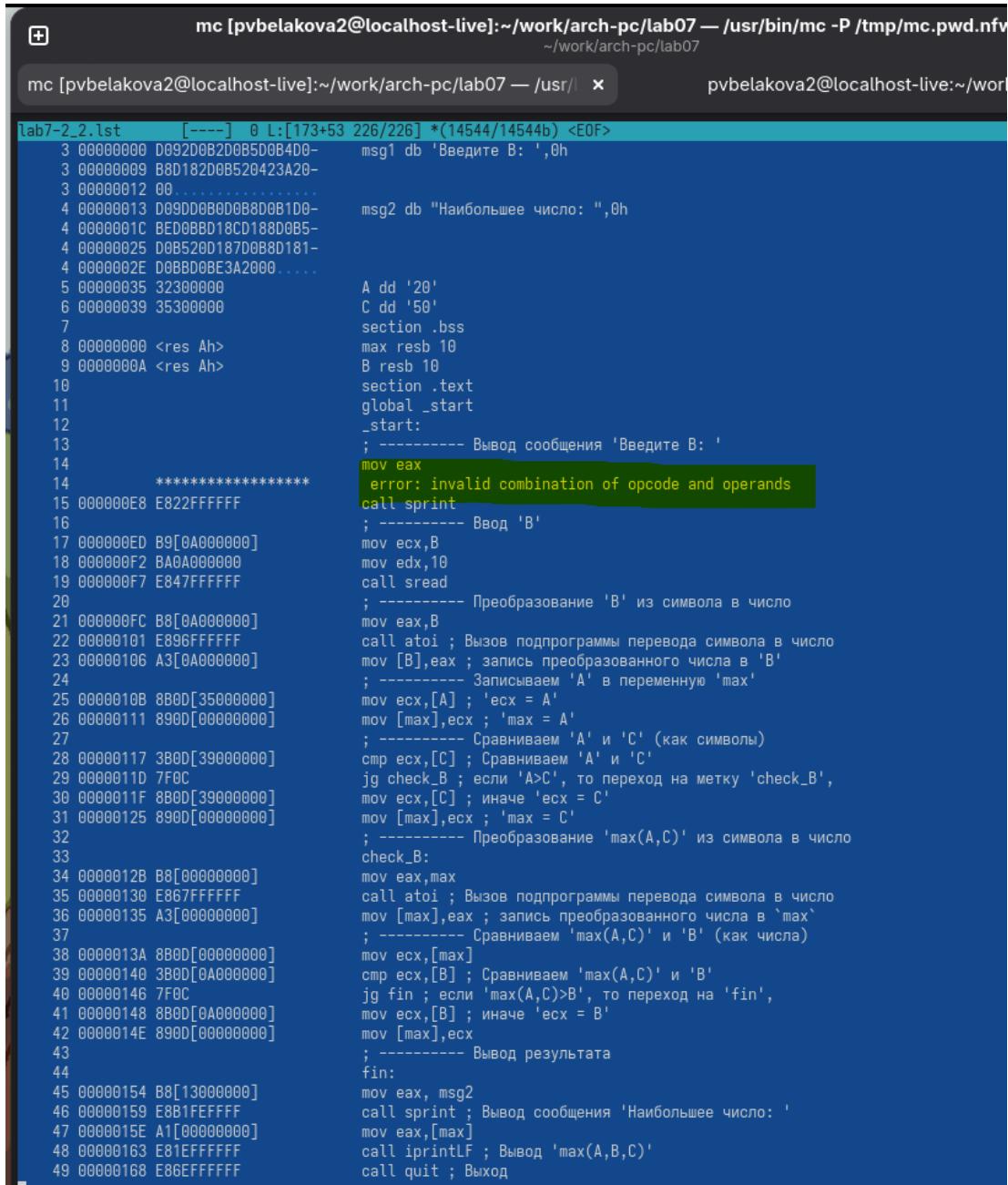
Выполняю трансляцию с получением файла листинга (рисунок 14).

```
pvelakova2@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
pvelakova2@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).



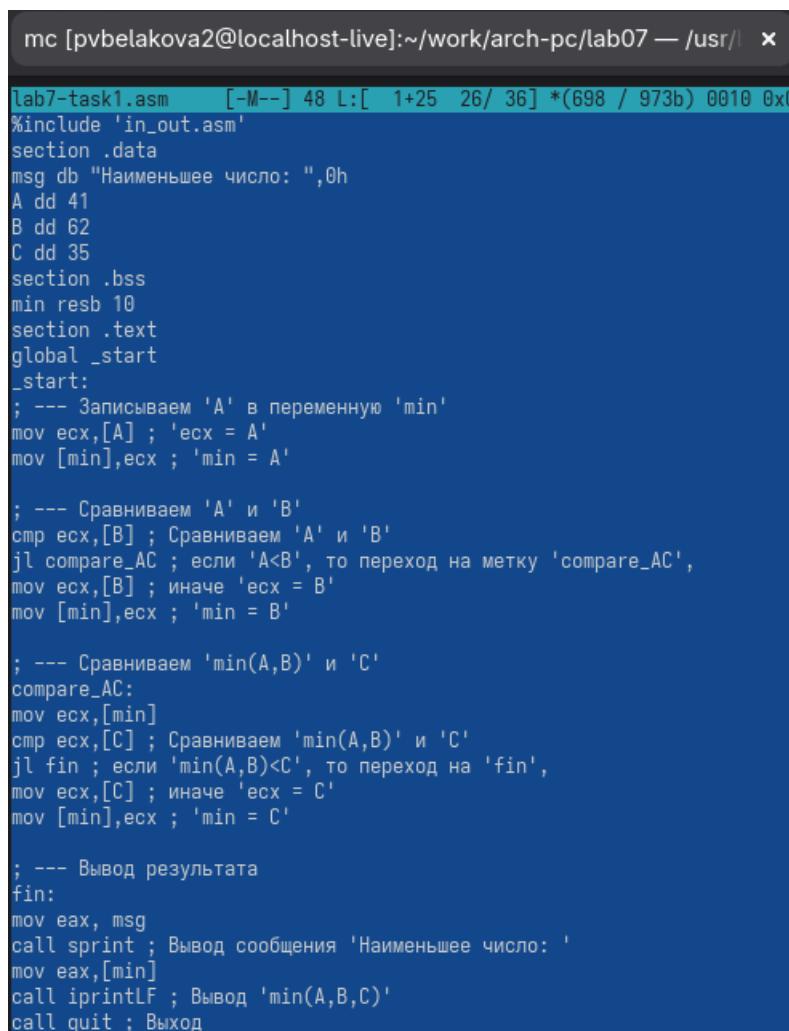
The screenshot shows a terminal window with the following details:

- Title bar: mc [pvbelakova2@localhost-live]:~/work/arch-pc/lab07 — /usr/bin/mc -P /tmp/mc.pwd.nfv
- Subtitle bar: ~ /work/arch-pc/lab07
- Text area:
 - File: lab7-2_2.lst
 - Line 14: `mov eax`
 - Line 15: `error: invalid combination of opcode and operands` (highlighted in green)
 - Line 16: `call sprint`
 - Line 17: `; ----- Ввод 'B'`
 - Line 18: `mov ecx,B`
 - Line 19: `mov edx,10`
 - Line 20: `call sread`
 - Line 21: `; ----- Преобразование 'B' из символа в число`
 - Line 22: `mov eax,B`
 - Line 23: `call atoi ; Вызов подпрограммы перевода символа в число`
 - Line 24: `mov [B],eax ; запись преобразованного числа в 'B'`
 - Line 25: `; ----- Записываем 'A' в переменную 'max'`
 - Line 26: `mov ecx,[A] ; 'ecx = A'`
 - Line 27: `mov [max],ecx ; 'max = A'`
 - Line 28: `; ----- Сравниваем 'A' и 'C' (как символы)`
 - Line 29: `cmp ecx,[C] ; Сравниваем 'A' и 'C'`
 - Line 30: `jg check_B ; если 'A>C', то переход на метку 'check_B',`
 - Line 31: `mov ecx,[C] ; иначе 'ecx = C'`
 - Line 32: `mov [max],ecx ; 'max = C'`
 - Line 33: `; ----- Преобразование 'max(A,C)' из символа в число`
 - Line 34: `check_B:`
 - Line 35: `mov eax,max`
 - Line 36: `call atoi ; Вызов подпрограммы перевода символа в число`
 - Line 37: `mov [max],eax ; запись преобразованного числа в 'max'`
 - Line 38: `; ----- Сравниваем 'max(A,C)' и 'B' (как числа)`
 - Line 39: `mov ecx,[max]`
 - Line 40: `cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'`
 - Line 41: `jg fin ; если 'max(A,C)>B', то переход на 'fin',`
 - Line 42: `mov ecx,[B] ; иначе 'ecx = B'`
 - Line 43: `mov [max],ecx`
 - Line 44: `; ----- Вывод результата`
 - Line 45: `fin:`
 - Line 46: `mov eax, msg2`
 - Line 47: `call sprint ; Вывод сообщения 'Наибольшее число: '`
 - Line 48: `mov eax,[max]`
 - Line 49: `call iprintLF ; Вывод 'max(A,B,C)'`
 - Line 50: `call quit ; Выход`

Рисунок 15 - часть листинг файла с указанием 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/i 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'

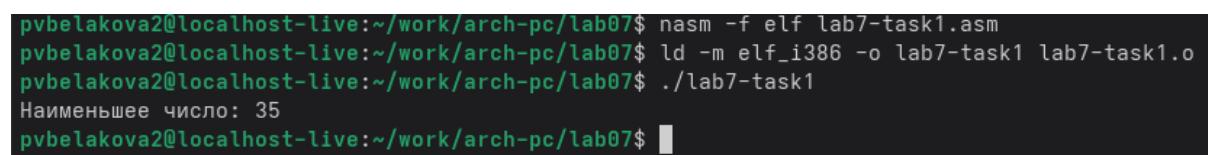
; --- Сравниваем 'A' и 'B'
cmp ecx,[B] ; Сравниваем 'A' и 'B'
jl compare_AC ; если 'A<B', то переход на метку 'compare_AC',
mov ecx,[B] ; иначе 'ecx = B'
mov [min],ecx ; 'min = B'

; --- Сравниваем 'min(A,B)' и 'C'
compare_AC:
mov ecx,[min]
cmp ecx,[C] ; Сравниваем 'min(A,B)' и 'C'
jl 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).

```
lab7-task2.asm      [-M--] 35 L:[ 1+41 42/ 57 ] *(1024/1372b) 0010 0x0E
%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).

```
pvbelakova2@localhost-live:~/work/arch-pc/lab07$ nasm -f elf lab7-task2.asm
pvbelakova2@localhost-live:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-task2 lab7-task2.o
pvbelakova2@localhost-live:~/work/arch-pc/lab07$ ./lab7-task2
Введите x: 3
Введите a: 0
Результат f(x): 1
pvbelakova2@localhost-live:~/work/arch-pc/lab07$ ./lab7-task2
Введите x: 1
Введите a: 2
Результат f(x): 6
pvbelakova2@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.