

Отчёт по лабораторной работе 7

Архитектура компьютеров

Агджабекова Эся Рустамовна НПИбд-01-24

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Самостоятельное задание	15
3	Выводы	20

Список иллюстраций

2.1	Создан каталог	6
2.2	Программа lab7-1.asm	7
2.3	Запуск программы lab7-1.asm	7
2.4	Программа lab7-1.asm	8
2.5	Запуск программы lab7-1.asm	9
2.6	Программа в файле lab7-1.asm	10
2.7	Запуск программы lab7-1.asm	10
2.8	Программа в файле lab7-2.asm	12
2.9	Запуск программы lab7-2.asm	12
2.10	Файл листинга lab7-2	13
2.11	Ошибка трансляции lab7-2	14
2.12	Файл листинга с ошибкой lab7-2	15
2.13	Программа в файле prog1.asm	16
2.14	Запуск программы prog1.asm	17
2.15	Программа в файле prog2.asm	18
2.16	Запуск программы prog2.asm	19

Список таблиц

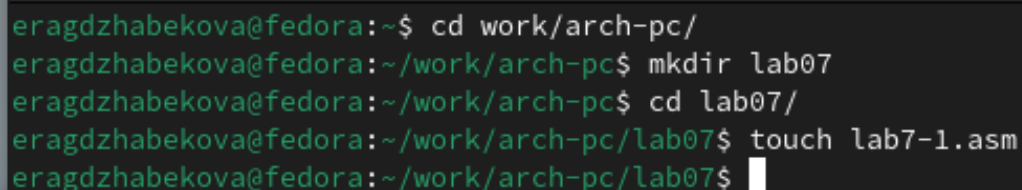
1 Цель работы

Целью работы является изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

2.1 Реализация переходов в NASM

Создала каталог для программ лабораторной работы № 7 и файл lab7-1.asm (рис. 2.1).



```
eragdzhbekova@fedora:~$ cd work/arch-pc/  
eragdzhbekova@fedora:~/work/arch-pc$ mkdir lab07  
eragdzhbekova@fedora:~/work/arch-pc$ cd lab07/  
eragdzhbekova@fedora:~/work/arch-pc/lab07$ touch lab7-1.asm  
eragdzhbekova@fedora:~/work/arch-pc/lab07$
```

Рис. 2.1: Создан каталог

В NASM инструкция `jmp` используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. В файле lab7-1.asm разместил текст программы из листинга 7.1 (рис. 2.2).

```
mc [eragdzhbekova@fedora]:~/work/arch-pc/la...
lab7-1.asm [-----] 9 L: [ 1+24 25/ 25] *(329
#include 'in_out.asm'
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
call sprintLF

_label2:
mov eax, msg2
call sprintLF

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Рис. 2.2: Программа lab7-1.asm

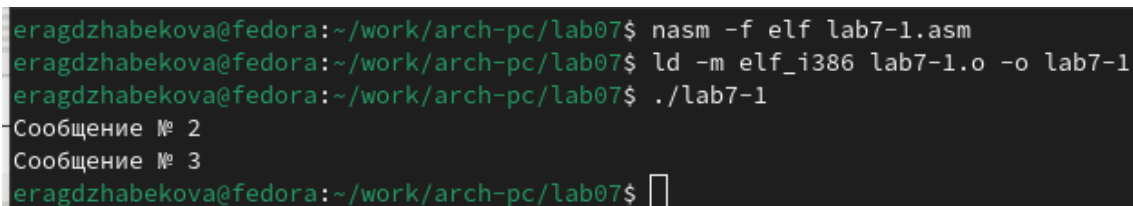
Создала исполняемый файл и запустил его (рис. 2.3).

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

Рис. 2.3: Запуск программы lab7-1.asm

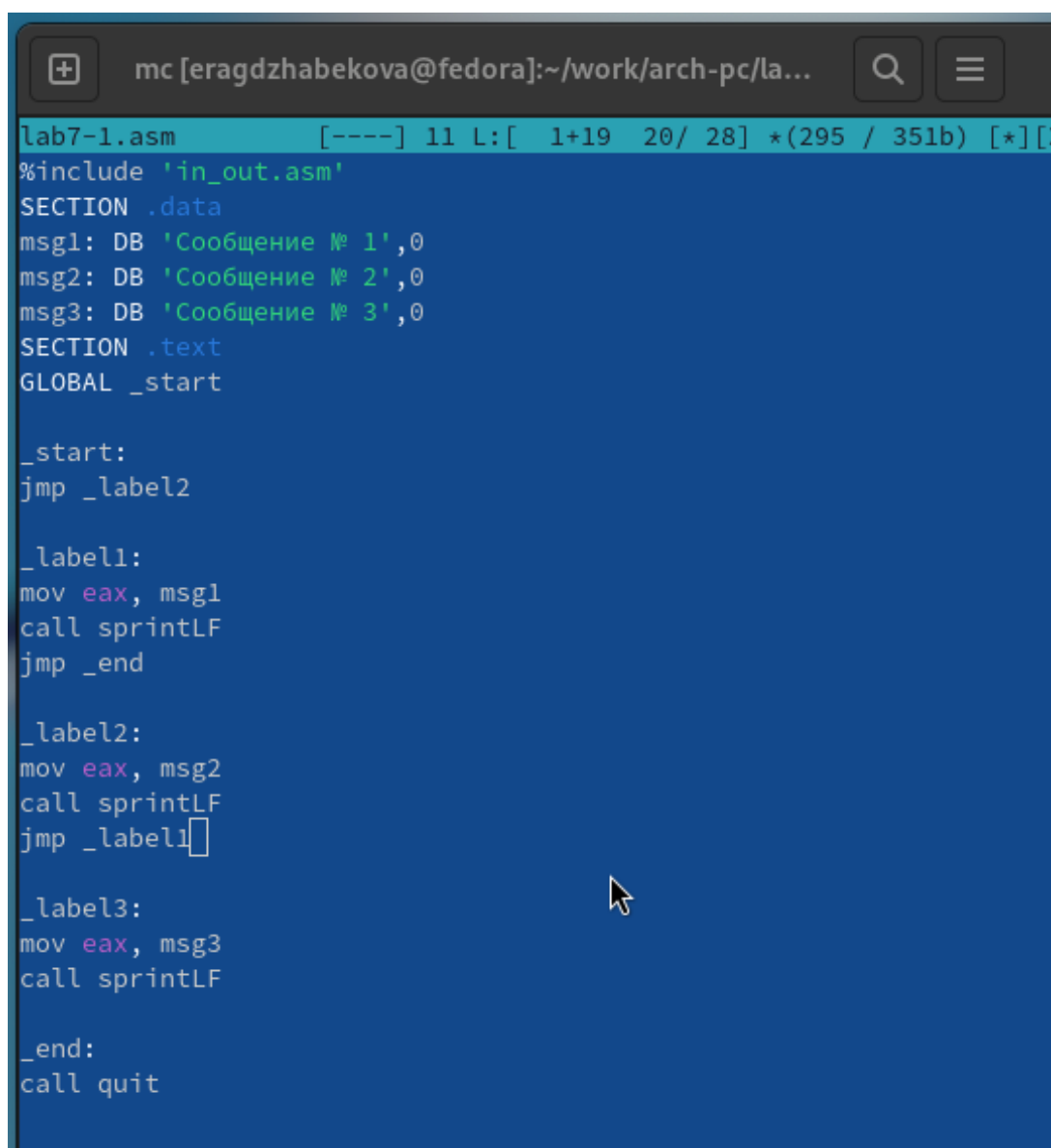
Инструкция `jmp` позволяет выполнять переходы как вперёд, так и назад. Изменил программу так, чтобы сначала выводилось сообщение № 2, затем сообщение № 1, после чего программа завершала работу. Для этого добавил в текст программы инструкцию `jmp` с меткой `_label1` после вывода сообщения № 2 (чтобы перейти к инструкции вывода сообщения № 1) и инструкцию `jmp` с меткой `_end` после вывода сообщения № 1 (для перехода к инструкции `call quit`).

Обновила текст программы согласно листингу 7.2 (рис. 2.4 и 2.5).



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

Рис. 2.4: Программа lab7-1.asm



```
lab7-1.asm [----] 11 L: [ 1+19 20/ 28] *(295 / 351b) [*] [
#include 'in_out.asm'
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
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

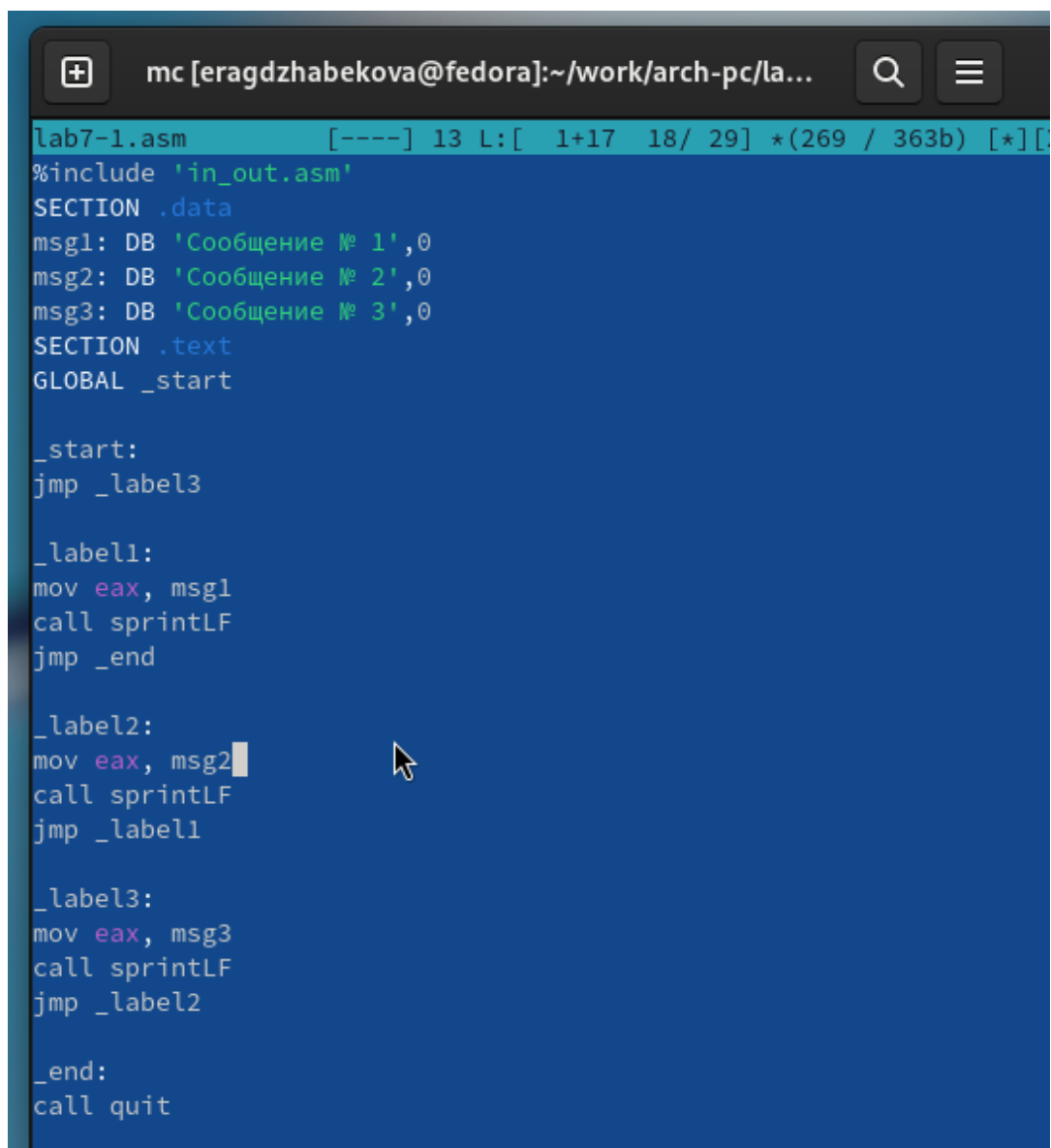
_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Рис. 2.5: Запуск программы lab7-1.asm

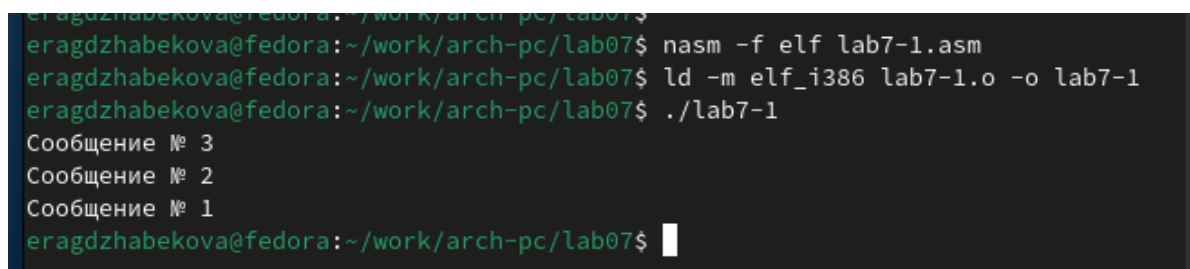
Изменила текст программы так, чтобы итоговый вывод программы выглядел следующим образом (рис. 2.6 и 2.7):

Сообщение № 3 Сообщение № 2 Сообщение № 1



```
lab7-1.asm [-----] 13 L: [ 1+17 18/ 29] *(269 / 363b) [*]  
%include 'in_out.asm'  
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  
call sprintLF  
jmp _end  
  
_label2:  
mov eax, msg2  
call sprintLF  
jmp _label1  
  
_label3:  
mov eax, msg3  
call sprintLF  
jmp _label2  
  
_end:  
call quit
```

Рис. 2.6: Программа в файле lab7-1.asm

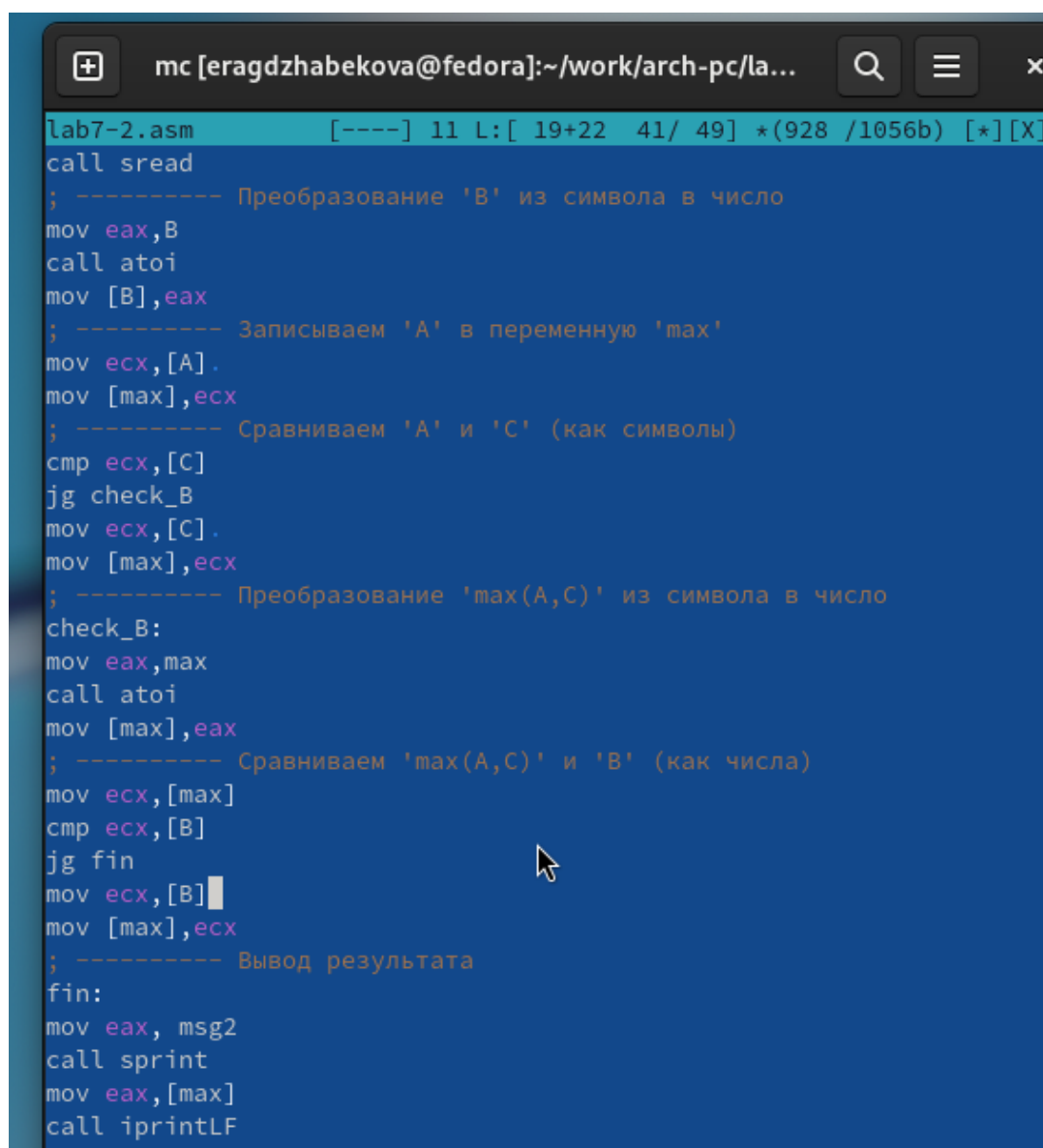


```
eragdzhbekova@fedora: ~/work/arch-pc/lab07$  
eragdzhbekova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm  
eragdzhbekova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1  
eragdzhbekova@fedora:~/work/arch-pc/lab07$ ./lab7-1  
Сообщение № 3  
Сообщение № 2  
Сообщение № 1  
eragdzhbekova@fedora:~/work/arch-pc/lab07$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

Рис. 2.7: Запуск программы lab7-1.asm

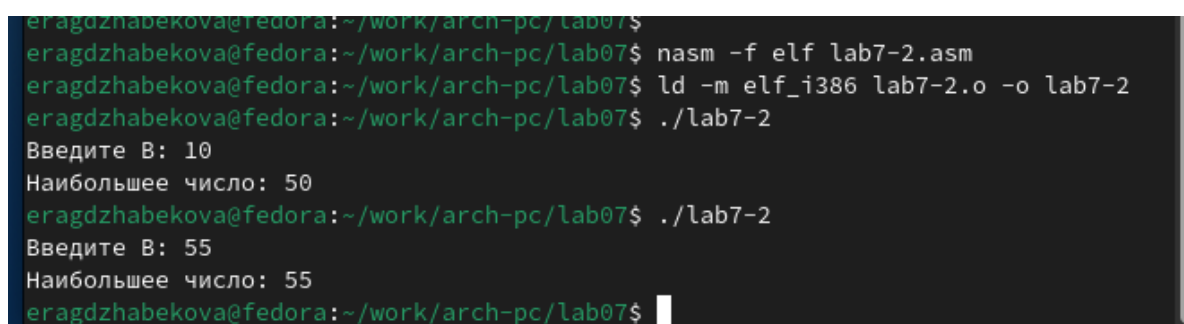
Использование инструкции `jmp` приводит к переходу в любом случае. Однако часто при написании программ требуется использовать условные переходы, когда переход должен происходить при выполнении какого-либо условия. В качестве примера рассмотрим программу, которая находит и выводит на экран наибольшую из 3 целочисленных переменных: А, В и С. Значения для А и С заданы в программе, значение В вводится с клавиатуры.

Я создала исполняемый файл и проверила его работу для разных значений В (рис. 2.8) (рис. 2.9).



```
lab7-2.asm [----] 11 L: [ 19+22 41/ 49] *(928 /1056b) [*][X]
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi
mov [B],eax
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A]
mov [max],ecx
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [max],ecx
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi
mov [max],eax
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B]
jg fin
mov ecx,[B]
mov [max],ecx
; ----- Вывод результата
fin:
mov eax,msg2
call sprint
mov eax,[max]
call iprintLF
call _exit
```

Рис. 2.8: Программа в файле lab7-2.asm

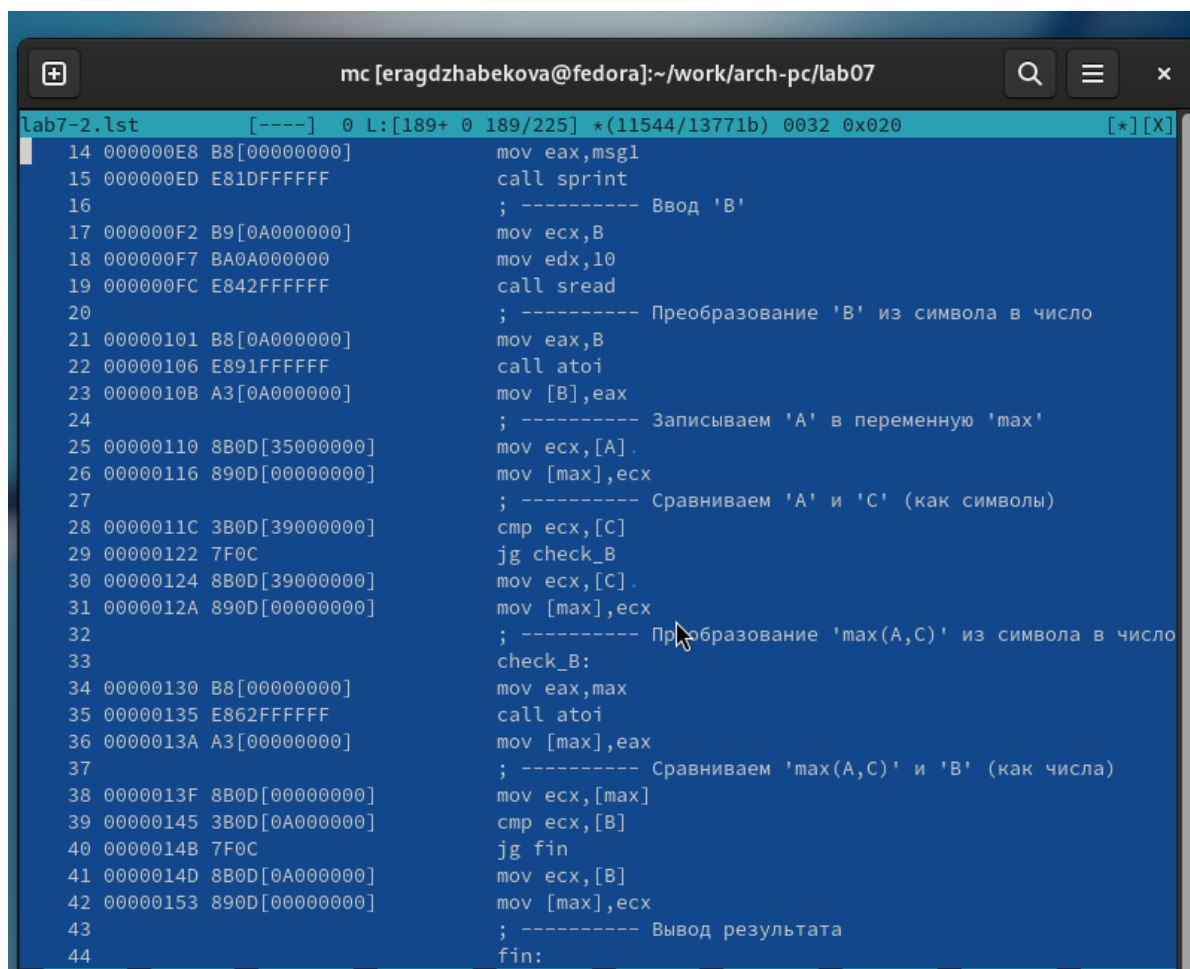


```
eragdzhbekova@fedora:~/work/arch-pc/lab07$
eragdzhbekova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
eragdzhbekova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
eragdzhbekova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50
eragdzhbekova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 55
Наибольшее число: 55
eragdzhbekova@fedora:~/work/arch-pc/lab07$
```

Рис. 2.9: Запуск программы lab7-2.asm

Обычно `nasm` создает в результате ассемблирования только объектный файл. Чтобы получить файл листинга, нужно указать ключ `-l` и задать имя файла листинга в командной строке.

Я создала файл листинга для программы из файла `lab7-2.asm` (рис. 2.10)



```
lab7-2.lst      [-----]  0 L:[189+ 0 189/225] *(11544/13771b) 0032 0x020  [*][X]
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
23 0000010B A3[0A000000]      mov [B],eax
24                               ; ----- Записываем 'A' в переменную 'max'
25 00000110 8B0D[35000000]      mov ecx,[A].
26 00000116 890D[00000000]      mov [max],ecx
27                               ; ----- Сравниваем 'A' и 'C' (как символы)
28 0000011C 3B0D[39000000]      cmp ecx,[C]
29 00000122 7F0C              jg check_B
30 00000124 8B0D[39000000]      mov ecx,[C].
31 0000012A 890D[00000000]      mov [max],ecx
32                               ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 00000130 B8[00000000]      mov eax,max
35 00000135 E862FFFFFF      call atoi
36 0000013A A3[00000000]      mov [max],eax
37                               ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013F 8B0D[00000000]      mov ecx,[max]
39 00000145 3B0D[0A000000]      cmp ecx,[B]
40 0000014B 7F0C              jg fin
41 0000014D 8B0D[0A000000]      mov ecx,[B]
42 00000153 890D[00000000]      mov [max],ecx
43                               ; ----- Вывод результата
44 fin:
```

Рис. 2.10: Файл листинга lab7-2

Внимательно ознакомилась с его форматом и содержимым. Подробно объяснила содержимое трёх строк файла листинга.

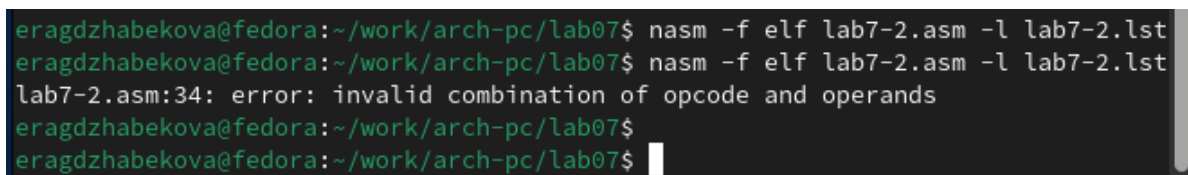
строка 189: - 14 — номер строки в подпрограмме - 000000E8 — адрес - B8[00000000] — машинный код - `mov eax,msg1` — код программы — перекладывает `msg1` в `eax`

строка 190: - 15 — номер строки в подпрограмме - 000000ED — адрес -

E81DFFFFFF — машинный код - call sprint — код программы — вызов подпрограммы печати

строка 192: - 17 — номер строки в подпрограмме - 000000F2 — адрес - B9[0A000000] — машинный код - mov ecx,B — код программы — перекладывает B в ecx

Я открыла файл с программой lab7-2.asm и в инструкции с двумя операндами удалила один операнд. Выполнила трансляцию с получением файла листинга. (рис. 2.11) (рис. 2.12)



```
eragdzhbekova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
eragdzhbekova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:34: error: invalid combination of opcode and operands
eragdzhbekova@fedora:~/work/arch-pc/lab07$ 
eragdzhbekova@fedora:~/work/arch-pc/lab07$
```

Рис. 2.11: Ошибка трансляции lab7-2

```
lab7-2.lst      [----]  8 L:[194+13 207/226] *(12682/13859b) 0032 0x020  [*][X]
19 000000FC E842FFFFFF      call sread
20                          ; ----- Преобразование 'B' из символа в число
21 00000101 B8[0A000000]      mov eax,B
22 00000106 E891FFFFFF      call atoi
23 0000010B A3[0A000000]      mov [B],eax
24                          ; ----- Записываем 'A' в переменную 'max'
25 00000110 8B0D[35000000]      mov ecx,[A]
26 00000116 890D[00000000]      mov [max],ecx
27                          ; ----- Сравниваем 'A' и 'C' (как символы)
28 0000011C 3B0D[39000000]      cmp ecx,[C]
29 00000122 7F0C              jg check_B
30 00000124 8B0D[39000000]      mov ecx,[C]
31 0000012A 890D[00000000]      mov [max],ecx
32                          ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 mov eax,
34 ***** error: invalid combination of opcode and operands
35 00000130 E867FFFFFF      call atoi
36 00000135 A3[00000000]      mov [max],eax
37                          ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013A 8B0D[00000000]      mov ecx,[max]
39 00000140 3B0D[0A000000]      cmp ecx,[B]
40 00000146 7F0C              jg fin
41 00000148 8B0D[0A000000]      mov ecx,[B]
42 0000014E 890D[00000000]      mov [max],ecx
43                          ; ----- Вывод результата
44 fin:
45 00000154 B8[13000000]      mov eax, msg2
46 00000159 E8B1FEFFFF      call sprint
47 0000015E A1[00000000]      mov eax,[max]
48 00000163 5B15555555      call _write+15
```

Рис. 2.12: Файл листинга с ошибкой lab7-2

Объектный файл не смог создаться из-за ошибки. Однако я получила листинг, в котором выделено место ошибки.

2.2 Самостоятельное задание

Напиши программу нахождения наименьшей из 3 целочисленных переменных а, b и с. Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создаю исполняемый файл и проверяю его работу (рис. 2.13) (рис. 2.14).

для варианта 15 — 32,6,54

```
mc [eragdzhabekova@fedora]:~/work/arch-
prog1.asm [-----] 0 L: [ 41+ 0 41/ 71] *(605 /1062b) 003
call sread.
mov eax,C
call atoi
mov [C],eax...
;-----algorithm-----
....
mov ecx,[A] ;ecx = A
mov [min],ecx ;min = A.

cmp ecx, [B] ; A&B
jl check_C ; if a<b: goto check_C.
mov ecx, [B]
mov [min], ecx ;else min = B

check_C:
cmp ecx, [C]
jl finish
mov ecx,[C]
mov [min],ecx.

finish:
mov eax,answer
call sprint

mov eax, [min]
call iprintLF

call quit
```

Рис. 2.13: Программа в файле prog1.asm


```

eragdzhbekova@fedora:~/work/arch-pc/lab07$
eragdzhbekova@fedora:~/work/arch-pc/lab07$ nasm -f elf prog1.asm
eragdzhbekova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 prog1.o -o prog1
eragdzhbekova@fedora:~/work/arch-pc/lab07$ ./prog1
Input A: 32
Input B: 6
Input C: 54
Smallest: 6
eragdzhbekova@fedora:~/work/arch-pc/lab07$
eragdzhbekova@fedora:~/work/arch-pc/lab07$

```

Рис. 2.14: Запуск программы prog1.asm

Теперь пишу программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выбираю из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создаю исполняемый файл и проверяю его работу для значений X и a из 7.6. (рис. 2.15) (рис. 2.16).

для варианта 15:

$$\begin{cases} a + 10, x < a \\ x + 10, x \geq a \end{cases}$$

Если подставить $x = 2, a = 3$ получается $3 + 10 = 13$.

Если подставить $x = 4, a = 2$ получается $4 + 10 = 14$.

```
mc [eragdzhbekova@fedora]:~/work/arch-pc/lab07
prog2.asm [----] 14 L: [ 22+25 47/ 51] *(693 / 728b) 0010 0x00A
    mov [A],eax

    mov eax,msgX
    call sprint
    mov ecx,X
    mov edx,80
    call sread.
    mov eax,X
    call atoi
    mov [X],eax...
;-----_algorithm-----

    mov ebx, [X]
    mov edx, [A]
    cmp ebx, edx
    jb first
    jmp second

first:
    mov eax,[A]
    add eax,10
    call iprintLF.
    call quit
second:
    mov eax,[X]
    add eax,10
    call iprintLF.
    call quit
```

Рис. 2.15: Программа в файле prog2.asm

```
eragdzhbekova@fedora:~/work/arch-pc/lab07$  
eragdzhbekova@fedora:~/work/arch-pc/lab07$ nasm -f elf prog2.asm  
eragdzhbekova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 prog2.o -o prog2  
eragdzhbekova@fedora:~/work/arch-pc/lab07$ ./prog2  
Input A: 3  
Input X: 2  
13  
eragdzhbekova@fedora:~/work/arch-pc/lab07$ ./prog2  
Input A: 2  
Input X: 4  
14  
eragdzhbekova@fedora:~/work/arch-pc/lab07$
```

Рис. 2.16: Запуск программы prog2.asm

3 Выводы

Изучили команды условного и безусловного переходов, познакомились с фалом листинга.