

# **Отчёт по лабораторной работе 7**

**Архитектура компьютеров**

Ахмади Ахмад Фаисал

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Самостоятельное задание . . . . .	15
<b>3</b>	<b>Выводы</b>	<b>20</b>

## Список иллюстраций

2.1	Программа в файле lab7-1.asm . . . . .	7
2.2	Запуск программы lab7-1.asm . . . . .	7
2.3	Программа в файле lab7-1.asm . . . . .	8
2.4	Запуск программы lab7-1.asm . . . . .	9
2.5	Программа в файле lab7-1.asm . . . . .	10
2.6	Запуск программы lab7-1.asm . . . . .	10
2.7	Программа в файле lab7-2.asm . . . . .	12
2.8	Запуск программы lab7-2.asm . . . . .	12
2.9	Файл листинга lab7-2 . . . . .	13
2.10	Ошибка трансляции lab7-2 . . . . .	14
2.11	Файл листинга с ошибкой lab7-2 . . . . .	15
2.12	Программа в файле lab7-3.asm . . . . .	16
2.13	Запуск программы lab7-3.asm . . . . .	16
2.14	Программа в файле lab7-4.asm . . . . .	18
2.15	Запуск программы lab7-4.asm . . . . .	19

## **Список таблиц**

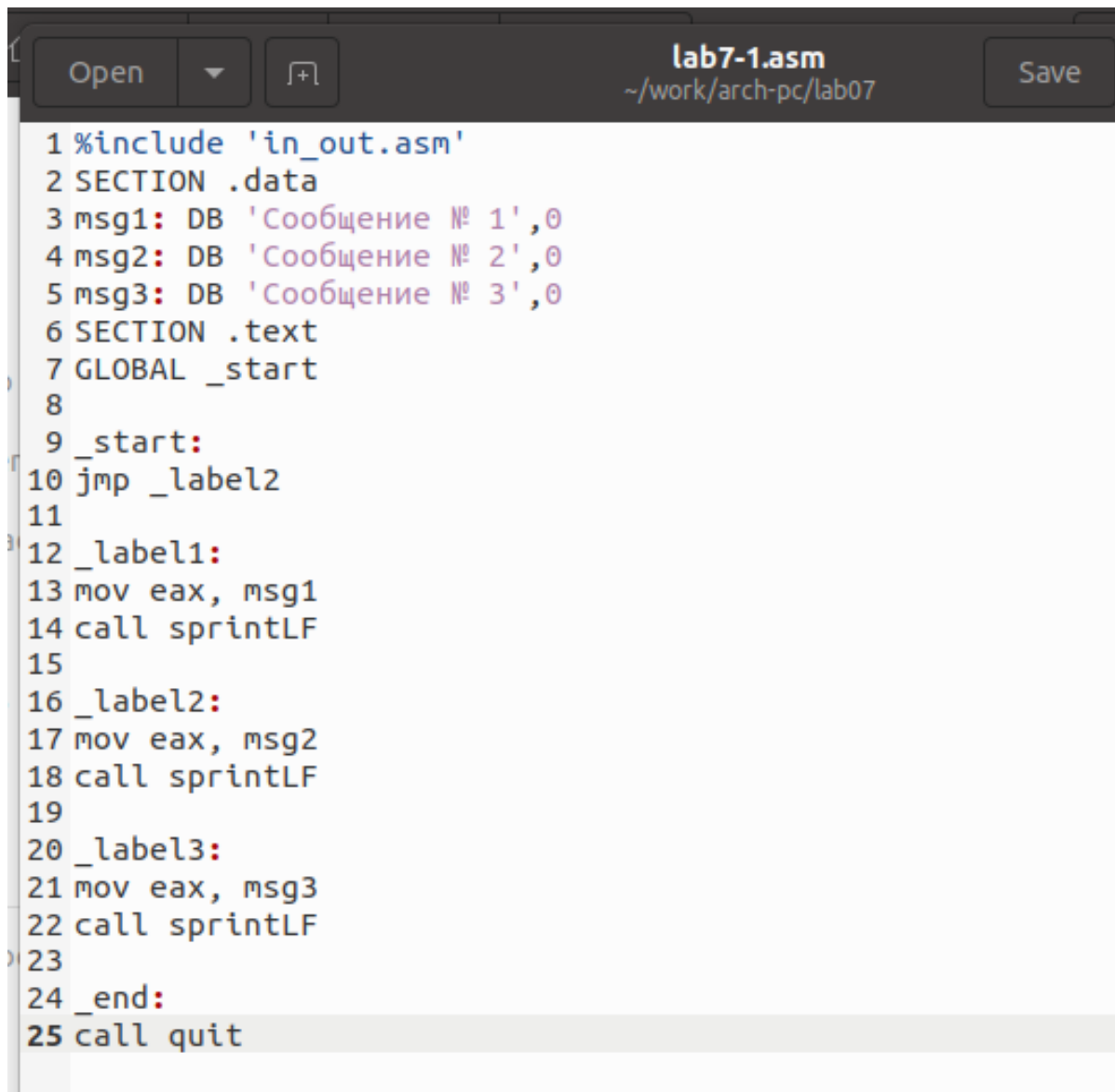
# 1 Цель работы

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

Создал каталог для программ лабораторной работы № 7 и файл lab7-1.asm.

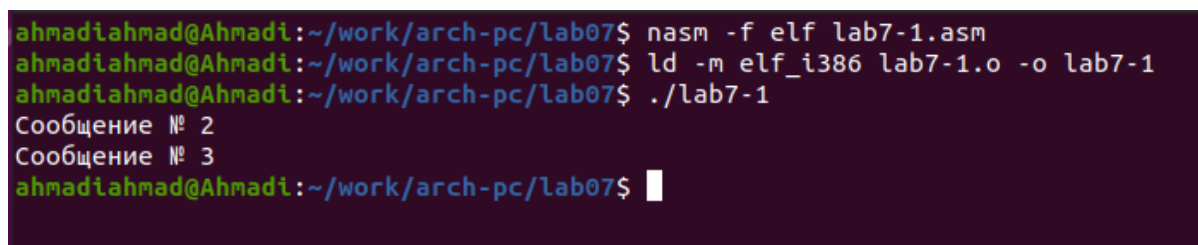
Инструкция jmp в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции jmp. Написал в файл lab7-1.asm текст программы из листинга 7.1. (рис. 2.1)



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

Рис. 2.1: Программа в файле lab7-1.asm

Создал исполняемый файл и запустил его. (рис. 2.2)

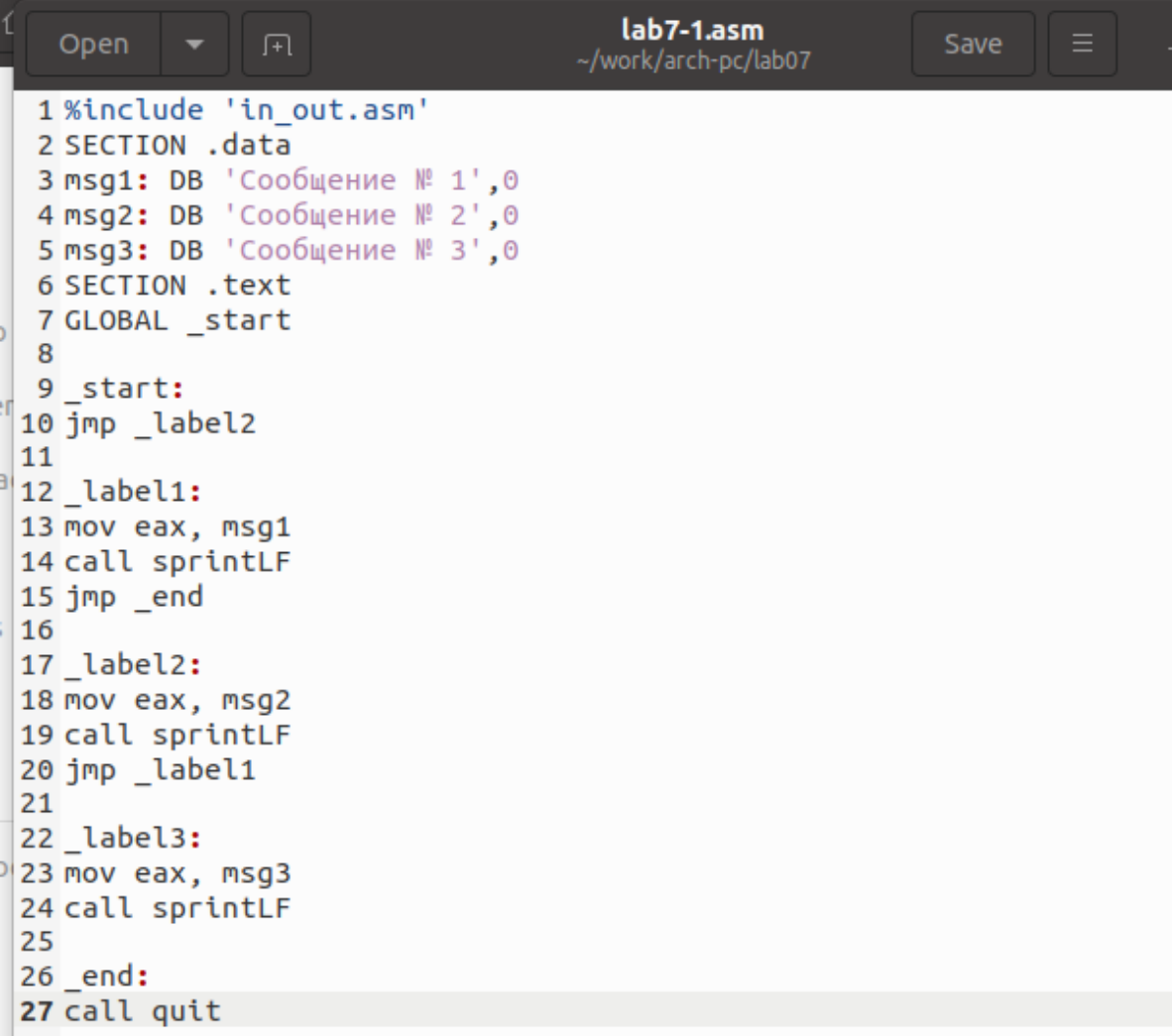


```
ahmadi@ahmad:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
ahmadi@ahmad:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
ahmadi@ahmad:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
ahmadi@ahmad:~/work/arch-pc/lab07$
```

Рис. 2.2: Запуск программы lab7-1.asm

Инструкция `jmp` позволяет осуществлять переходы не только вперед но и назад. Изменим программу таким образом, чтобы она выводила сначала 'Сообщение № 2', потом 'Сообщение № 1' и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим 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`).

Изменил текст программы в соответствии с листингом 7.2. (рис. 2.3) (рис. 2.4)



```
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
9 _start:
10 jmp _label2
11
12 _label1:
13 mov eax, msg1
14 call sprintLF
15 jmp _end
16
17 _label2:
18 mov eax, msg2
19 call sprintLF
20 jmp _label1
21
22 _label3:
23 mov eax, msg3
24 call sprintLF
25
26 _end:
27 call quit
```

Рис. 2.3: Программа в файле lab7-1.asm



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

Рис. 2.4: Запуск программы lab7-1.asm

Изменил текст программы, изменив инструкции jmp, чтобы вывод программы был следующим (рис. 2.5) (рис. 2.6):

Сообщение № 3

Сообщение № 2

Сообщение № 1

```
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
9 _start:
10 jmp _label3
11
12 _label1:
13 mov eax, msg1
14 call sprintf
15 jmp _end
16
17 _label2:
18 mov eax, msg2
19 call sprintf
20 jmp _label1
21
22 _label3:
23 mov eax, msg3
24 call sprintf
25 jmp _label2
26
27 _end:
28 call quit
```

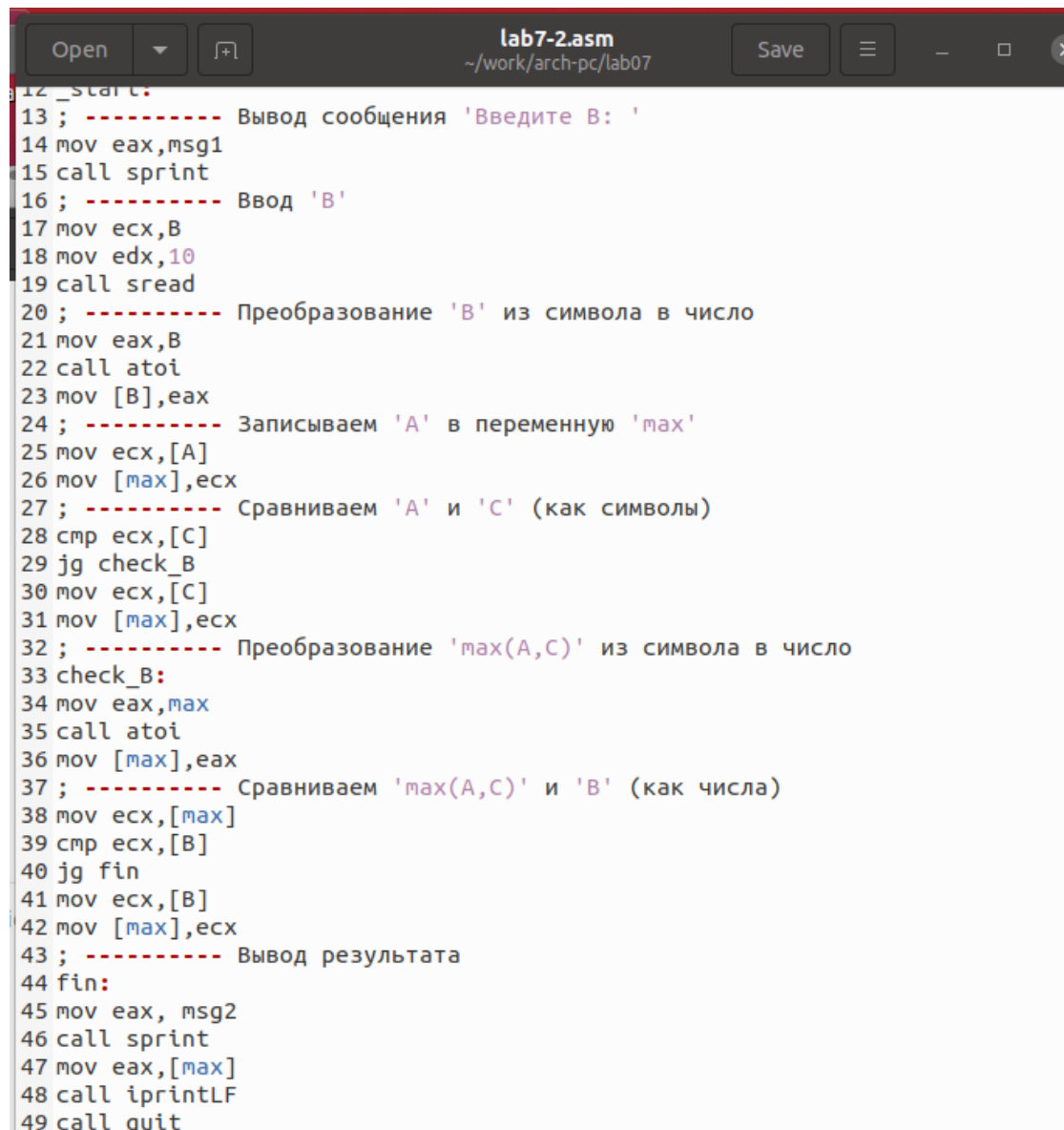
Рис. 2.5: Программа в файле lab7-1.asm

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

Рис. 2.6: Запуск программы lab7-1.asm

Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А, В и С. Значения для А и С задаются в программе, значение В вводится с клавиатуры.

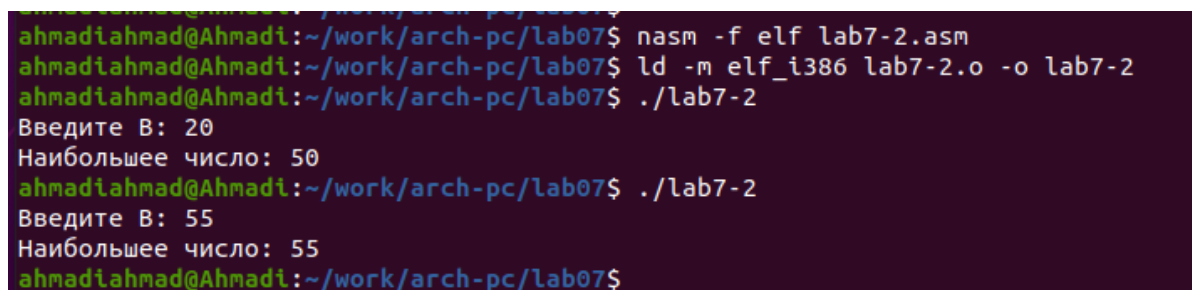
Создал исполняемый файл и проверил его работу для разных значений В (рис. 2.7) (рис. 2.8).



```
lab7-2.asm
~/work/arch-pc/lab07
Open Save

12 _start:
13 ; ----- Вывод сообщения 'Введите B: '
14 mov eax,msg1
15 call sprint
16 ; ----- Ввод 'B'
17 mov ecx,B
18 mov edx,10
19 call sread
20 ; ----- Преобразование 'B' из символа в число
21 mov eax,B
22 call atoi
23 mov [B],eax
24 ; ----- Записываем 'A' в переменную 'max'
25 mov ecx,[A]
26 mov [max],ecx
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx,[C]
29 jg check_B
30 mov ecx,[C]
31 mov [max],ecx
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 mov eax,max
35 call atoi
36 mov [max],eax
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 mov ecx,[max]
39 cmp ecx,[B]
40 jg fin
41 mov ecx,[B]
42 mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 mov eax, msg2
46 call sprint
47 mov eax,[max]
48 call iprintLF
49 call quit
```

Рис. 2.7: Программа в файле lab7-2.asm



```
ahmadiyahmad@Ahmadi:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
ahmadiyahmad@Ahmadi:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
ahmadiyahmad@Ahmadi:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 20
Наибольшее число: 50
ahmadiyahmad@Ahmadi:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 55
Наибольшее число: 55
ahmadiyahmad@Ahmadi:~/work/arch-pc/lab07$
```

Рис. 2.8: Запуск программы lab7-2.asm

Обычно `nasm` создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ `-l` и задав имя файла листинга в командной строке.

Создал файл листинга для программы из файла `lab7-2.asm` (рис. 2.9)

183	8	00000000	<res 0000000A>	max resb 10
184	9	0000000A	<res 0000000A>	B resb 10
185	10			section .text
186	11			global _start
187	12			_start:
188	13			; ----- Вывод сообщения 'Введите B: '
189	14	000000E8	B8[00000000]	mov eax,msg1
190	15	000000ED	E81DFFFFFF	call sprint
191	16			; ----- Ввод 'B'
192	17	000000F2	B9[0A000000]	mov ecx,B
193	18	000000F7	BA0A000000	mov edx,10
194	19	000000FC	E842FFFFFF	call sread
195	20			; ----- Преобразование 'B' из символа в число
196	21	00000101	B8[0A000000]	mov eax,B
197	22	00000106	E891FFFFFF	call atoi
198	23	0000010B	A3[0A000000]	mov [B],eax
199	24			; ----- Записываем 'A' в переменную 'max'
200	25	00000110	8B0D[35000000]	mov ecx,[A]
201	26	00000116	890D[00000000]	mov [max],ecx
202	27			; ----- Сравниваем 'A' и 'C' (как символы)
203	28	0000011C	3B0D[39000000]	cmp ecx,[C]
204	29	00000122	7F0C	jg check_B
205	30	00000124	8B0D[39000000]	mov ecx,[C]
206	31	0000012A	890D[00000000]	mov [max],ecx
207	32			; ----- Преобразование 'max(A,C)' из символа в число
208	33			check_B:
209	34	00000130	B8[00000000]	mov eax,max
210	35	00000135	E862FFFFFF	call atoi
211	36	0000013A	A3[0A000000]	mov [max],eax

Рис. 2.9: Файл листинга `lab7-2`

Внимательно ознакомился с его форматом и содержимым. Подробно объясню содержимое трёх строк файла листинга по выбору.

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

Открыл файл с программой lab7-2.asm и в инструкции с двумя операндами удалил один операнд. Выполнил трансляцию с получением файла листинга. (рис. 2.10) (рис. 2.11)

```
ahmadiyahmad@Ahmadi:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
ahmadiyahmad@Ahmadi:~/work/arch-pc/lab07$
ahmadiyahmad@Ahmadi:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:30: error: invalid combination of opcode and operands
ahmadiyahmad@Ahmadi:~/work/arch-pc/lab07$
```

Рис. 2.10: Ошибка трансляции lab7-2

197	22	00000106	E891FFFFFF	call atoi
198	23	0000010B	A3[0A000000]	mov [B],eax
199	24			; ----- Записываем 'A' в переменную 'max'
200	25	00000110	8B0D[35000000]	mov ecx,[A]
201	26	00000116	890D[00000000]	mov [max],ecx
202	27			; ----- Сравниваем 'A' и 'C' (как символы)
203	28	0000011C	3B0D[39000000]	cmp ecx,[C]
204	29	00000122	7F06	jg check_B
205	30			mov ecx,
206	30		*****	error: invalid combination of opcode and operands
207	31	00000124	890D[00000000]	mov [max],ecx
208	32			; ----- Преобразование 'max(A,C)' из символа в число
209	33			check_B:
210	34	0000012A	B8[00000000]	mov eax,max
211	35	0000012F	E868FFFFFF	call atoi
212	36	00000134	A3[00000000]	mov [max],eax
213	37			; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
214	38	00000139	8B0D[00000000]	mov ecx,[max]
215	39	0000013F	3B0D[0A000000]	cmp ecx,[B]
216	40	00000145	7F0C	jg fin
217	41	00000147	8B0D[0A000000]	mov ecx,[B]
218	42	0000014D	890D[00000000]	mov [max],ecx
219	43			; ----- Вывод результата
220	44			fin:
221	45	00000153	B8[13000000]	mov eax, msg2

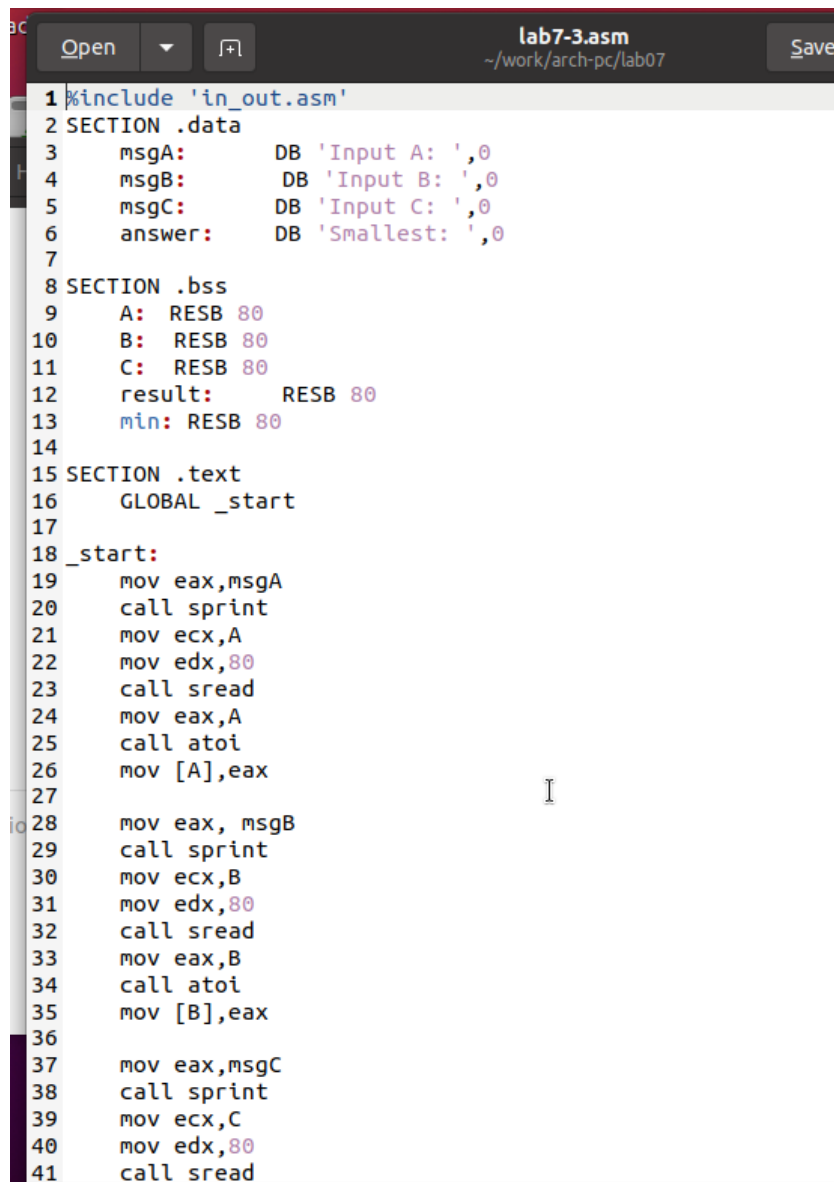
Рис. 2.11: Файл листинга с ошибкой lab7-2

Объектный файл не смог создаться из-за ошибки. Но получился листинг, где выделено место ошибки.

## 2.1 Самостоятельное задание

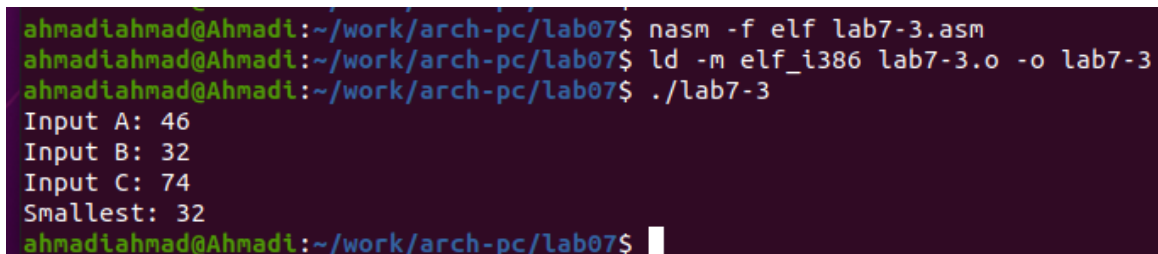
Напишите программу нахождения наименьшей из 3 целочисленных переменных a,b и c. Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу (рис. 2.12) (рис. 2.13)

для варианта 19 - 46,32,74



```
1 %include 'in_out.asm'
2 SECTION .data
3     msgA:      DB 'Input A: ',0
4     msgB:      DB 'Input B: ',0
5     msgC:      DB 'Input C: ',0
6     answer:    DB 'Smallest: ',0
7
8 SECTION .bss
9     A:  RESB 80
10    B:  RESB 80
11    C:  RESB 80
12    result:  RESB 80
13    min:  RESB 80
14
15 SECTION .text
16     GLOBAL _start
17
18 _start:
19     mov eax,msgA
20     call sprint
21     mov ecx,A
22     mov edx,80
23     call sread
24     mov eax,A
25     call atoi
26     mov [A],eax
27
28     mov eax, msgB
29     call sprint
30     mov ecx,B
31     mov edx,80
32     call sread
33     mov eax,B
34     call atoi
35     mov [B],eax
36
37     mov eax,msgC
38     call sprint
39     mov ecx,C
40     mov edx,80
41     call sread
```

Рис. 2.12: Программа в файле lab7-3.asm



```
ahmadiyahmad@Ahmadi:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
ahmadiyahmad@Ahmadi:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-3.o -o lab7-3
ahmadiyahmad@Ahmadi:~/work/arch-pc/lab07$ ./lab7-3
Input A: 46
Input B: 32
Input C: 74
Smallest: 32
ahmadiyahmad@Ahmadi:~/work/arch-pc/lab07$
```

Рис. 2.13: Запуск программы lab7-3.asm



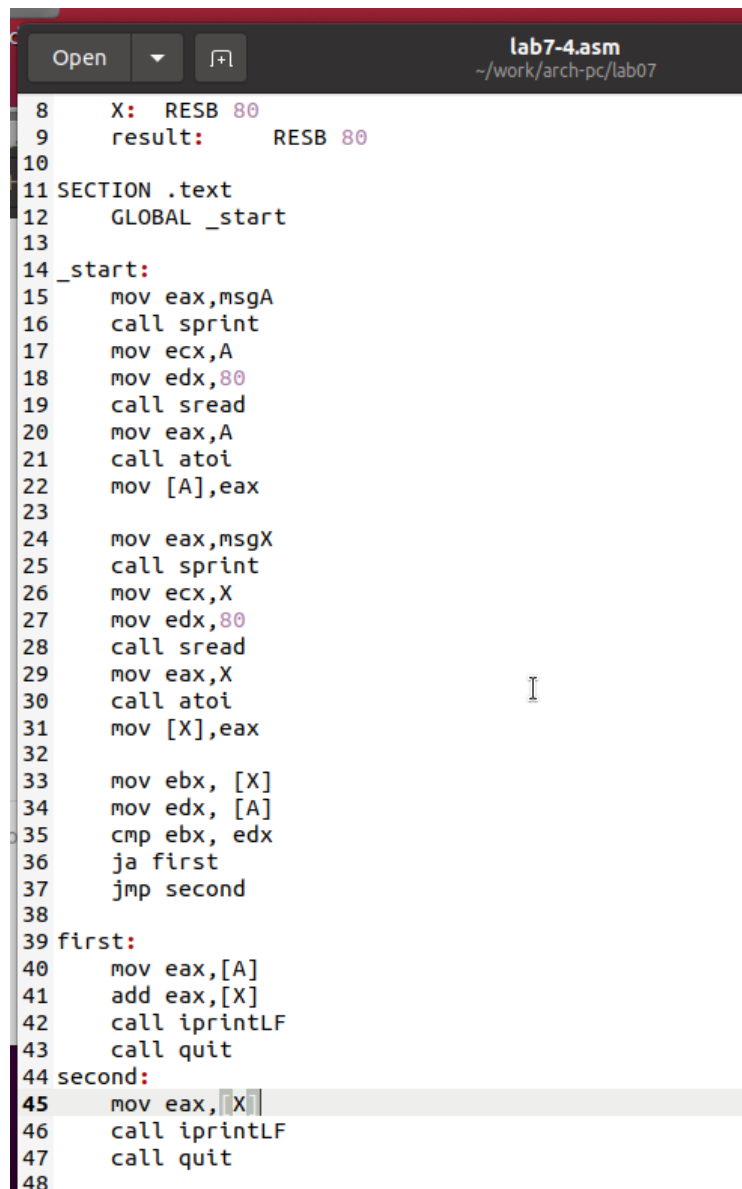
Напишите программу, которая для введенных с клавиатуры значений  $x$  и  $a$  вычисляет значение заданной функции  $f(x)$  и выводит результат вычислений. Вид функции  $f(x)$  выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений  $X$  и  $a$  из 7.6. (рис. 2.14) (рис. 2.15)

для варианта 19

$$\begin{cases} a + x, x > a \\ x, x \leq a \end{cases}$$

Если подставить  $x = 4, a = 5$  получается 4.

Если подставить  $x = 3, a = 2$  получается 5.



```
8      X:  RESB 80
9      result:  RESB 80
10
11 SECTION .text
12     GLOBAL _start
13
14 _start:
15     mov eax,msgA
16     call sprint
17     mov ecx,A
18     mov edx,80
19     call sread
20     mov eax,A
21     call atoi
22     mov [A],eax
23
24     mov eax,msgX
25     call sprint
26     mov ecx,X
27     mov edx,80
28     call sread
29     mov eax,X
30     call atoi
31     mov [X],eax
32
33     mov ebx, [X]
34     mov edx, [A]
35     cmp ebx, edx
36     ja first
37     jmp second
38
39 first:
40     mov eax,[A]
41     add eax,[X]
42     call iprintLF
43     call quit
44 second:
45     mov eax,[X]
46     call iprintLF
47     call quit
48
```

Рис. 2.14: Программа в файле lab7-4.asm

```
ahmadiahmad@Ahmadi:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
ahmadiahmad@Ahmadi:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-4.o -o lab7-4
ahmadiahmad@Ahmadi:~/work/arch-pc/lab07$ ./lab7-4
Input A: 5
Input X: 4
4
ahmadiahmad@Ahmadi:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-4.o -o lab7-4
ahmadiahmad@Ahmadi:~/work/arch-pc/lab07$ 2
2: command not found
ahmadiahmad@Ahmadi:~/work/arch-pc/lab07$ ./lab7-4
Input A: 2
Input X: 3
5
ahmadiahmad@Ahmadi:~/work/arch-pc/lab07$
```

Рис. 2.15: Запуск программы lab7-4.asm

## 3 Выводы

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