

Отчёт по лабораторной работе 7

Архитектура компьютера

Гуламова Е.М. НПИбд-03-23

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	23

Список иллюстраций

2.1	Программа в файле lab7-1.asm	7
2.2	Запуск программы lab7-1.asm	8
2.3	Программа в файле lab7-1.asm:	9
2.4	Запуск программы lab7-1.asm:	10
2.5	Программа в файле lab7-1.asm	11
2.6	Запуск программы lab7-1.asm	12
2.7	Программа в файле lab7-2.asm	13
2.8	Запуск программы lab7-2.asm	14
2.9	Файл листинга lab7-2	15
2.10	Ошибка трансляции lab7-2	17
2.11	Файл листинга с ошибкой lab7-2	18
2.12	Программа в файле task.asm	19
2.13	Запуск программы task.asm	20
2.14	Программа в файле task2.asm	21
2.15	Запуск программы task2.asm	22

Список таблиц

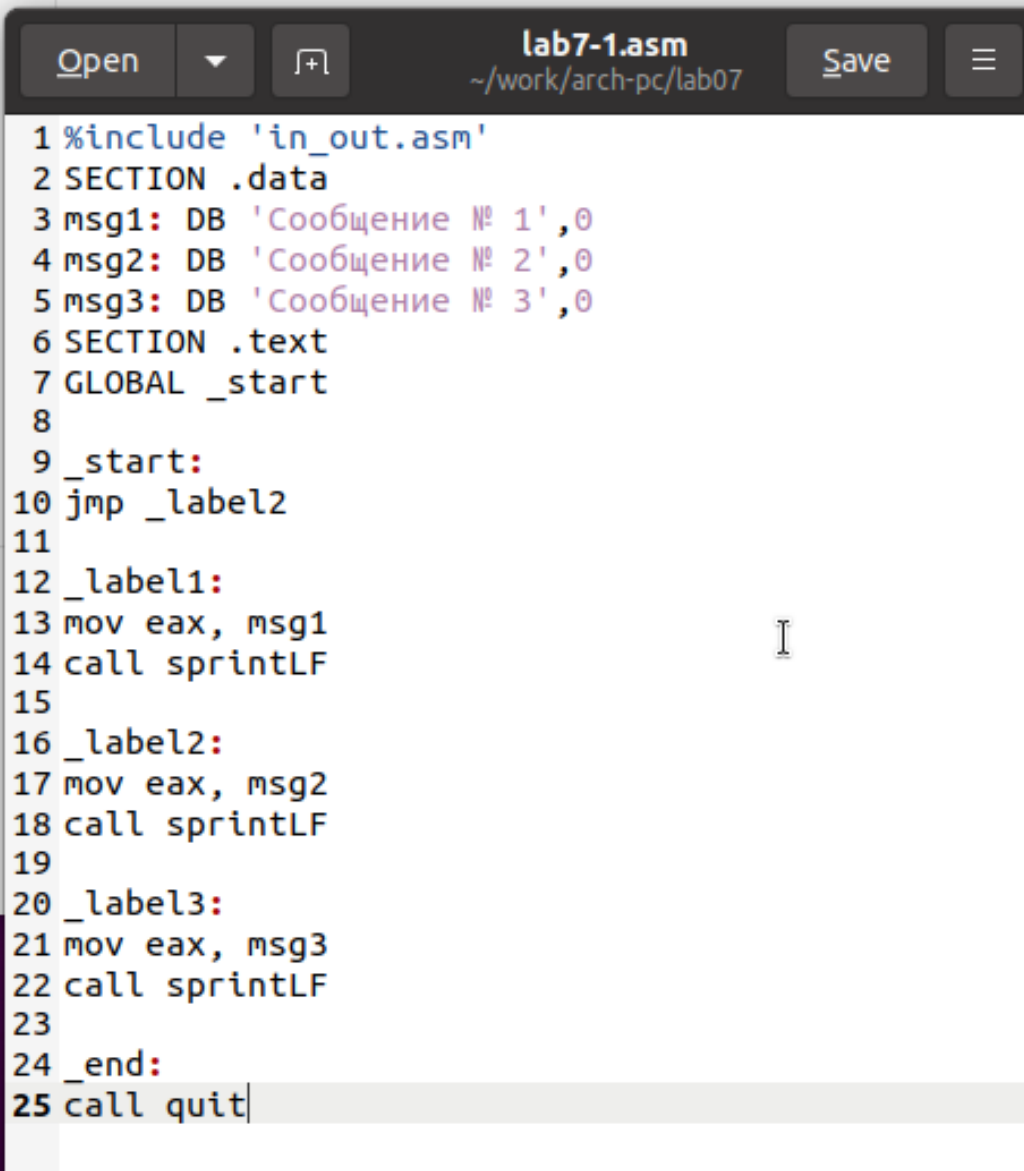
1 Цель работы

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

1. Я создала папку для программы, которую буду использовать в лабораторной работе номер семь, и подготовила файл lab7-1.asm для написания кода.
2. В NASM команда `jmp` позволяет выполнять безусловные переходы. Давайте посмотрим на пример программы, где эта команда применяется.

Я ввела текст программы в файл lab7-1.asm, следуя примеру из листинга 7.1.



```
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 sprintf
15
16 _label2:
17 mov eax, msg2
18 call sprintf
19
20 _label3:
21 mov eax, msg3
22 call sprintf
23
24 _end:
25 call quit
```

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

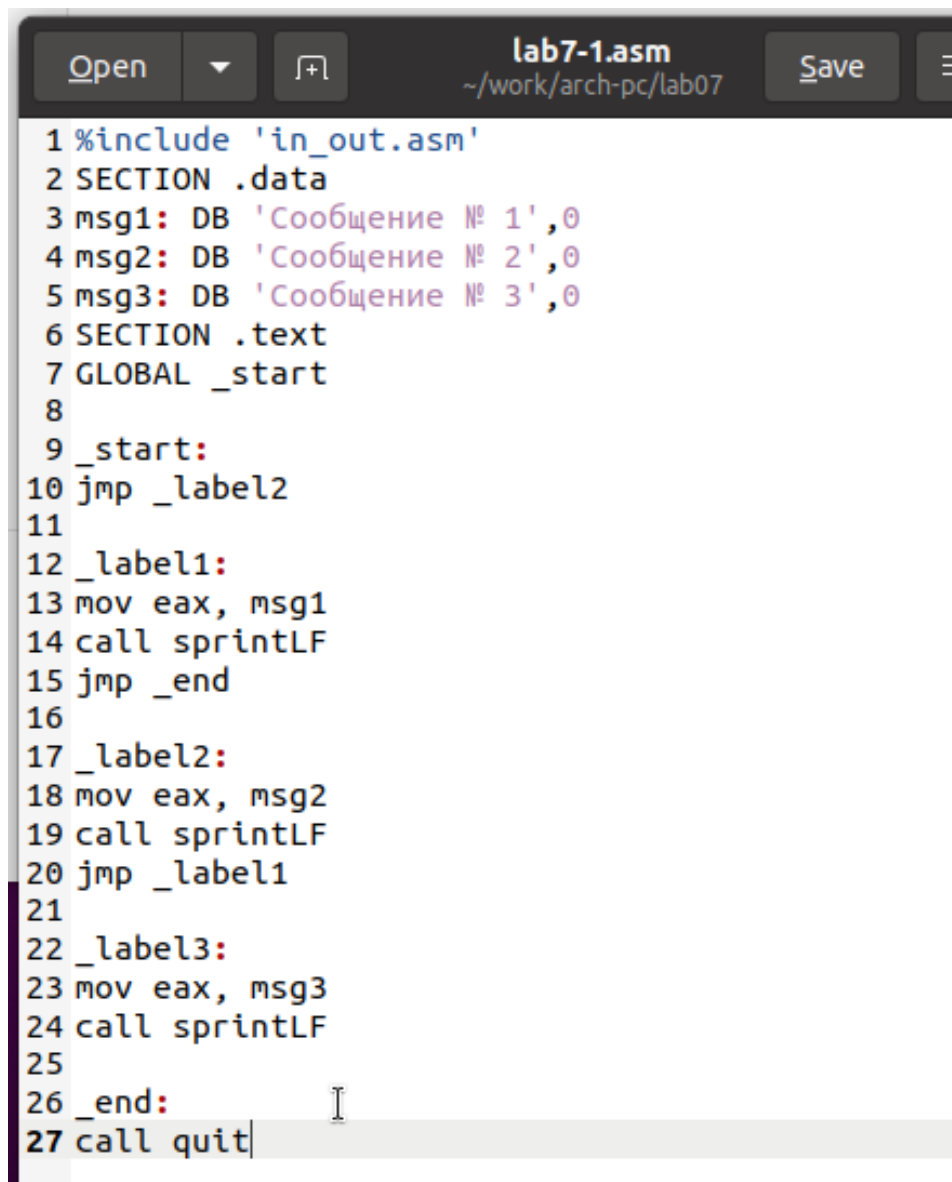
Затем я скомпилировала эту программу, создав исполняемый файл, и успешно запустила его.

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

Рис. 2.2: Запуск программы lab7-1.asm

Команда `jmp` не ограничивается только прямыми переходами; она также позволяет переходить назад. Я изменила программу так, чтобы она сначала выводила “Сообщение № 2”, затем “Сообщение № 1” и после этого завершала свою работу. Для этого я добавила в код программы после вывода “Сообщение № 2” команду `jmp` с меткой `_label1`, которая переводит выполнение к коду, выводящему “Сообщение № 1”. После вывода “Сообщение № 1” я вставила ещё одну команду `jmp`, на этот раз с меткой `_end`, чтобы перейти к завершающей части программы с вызовом функции `quit`.

Изменила текст программы в соответствии с листингом 7.2.



```
lab7-1.asm
~/work/arch-pc/lab07

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 sprintfLF
15 jmp _end
16
17 _label2:
18 mov eax, msg2
19 call sprintfLF
20 jmp _label1
21
22 _label3:
23 mov eax, msg3
24 call sprintfLF
25
26 _end:
27 call quit
```

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

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

Рис. 2.4: Запуск программы lab7-1.asm:

Изменила команды jmp для изменения порядка вывода сообщений программой.

Сообщение № 3

Сообщение № 2

Сообщение № 1

```
lab7-1.asm
~/work/arch-pc/lab07

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

```

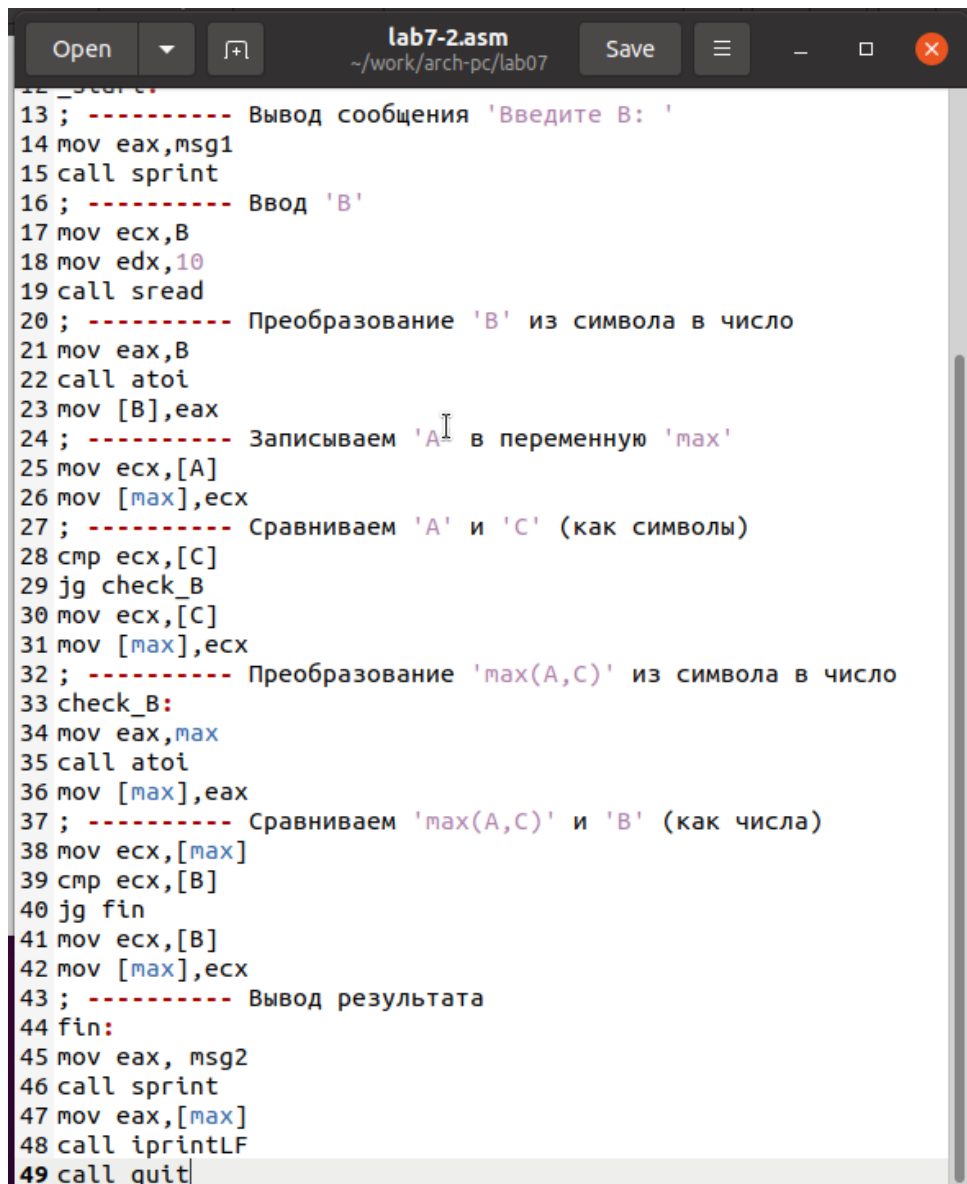
gulamova@ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
gulamova@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
gulamova@ubuntu:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
gulamova@ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
gulamova@ubuntu:~/work/arch-pc/lab07$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

```

Рис. 2.6: Запуск программы lab7-1.asm

3. Команда `jmp` всегда заставляет программу перейти к указанной точке. Но иногда мне нужно сделать так, чтобы переход выполнялся только при определённых условиях. Например, я написала программу, которая сравнивает три целых числа: А, В и С, чтобы выявить и показать на экране самое большое из них. Я заранее задала значения для А и С, а значение для В программа получает от пользователя через ввод с клавиатуры.

Я собрала исполняемый файл и проверила, как он работает, вводя различные числа для В.



```
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

```
gulamova@ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
gulamova@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
gulamova@ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 10
Наибольшее число: 50
gulamova@ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 30
Наибольшее число: 50
gulamova@ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 50
Наибольшее число: 50
gulamova@ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 60
Наибольшее число: 60
gulamova@ubuntu:~/work/arch-pc/lab07$ █
```

Рис. 2.8: Запуск программы lab7-2.asm

4. Обычно при работе с `nasm` получается только объектный файл после ассемблирования. Но на этот раз мне нужно было создать файл листинга, что я сделала, используя ключ `-l` и указав имя нужного файла прямо в командной строке.

Я подготовила файл листинга для своей программы, находящейся в файле `lab7-2.asm`, и внимательно изучила его структуру и содержимое. Подробно расскажу о трёх строках из этого файла.

```

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[00000000]    mov [max],eax
212 37                          ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
213 38 0000013F 8B0D[00000000]   mov ecx,[max]
214 39 00000145 3B0D[0A000000]   cmp ecx,[B]
215 40 0000014B 7F0C             jg fin
216 41 0000014D 8B0D[0A000000]   mov ecx,[B]
217 42 00000153 890D[00000000]   mov [max],ecx
218 43                          ; ----- Вывод результата
219 44                          fin:
220 45 00000159 B8[13000000]    mov eax,msg2
221 46 0000015E E8ACFFFFFF      call sprint
222 47 00000163 A1[00000000]    mov eax,[max]
223 48 00000168 E819FFFFFF      call iprintLF
224 49 0000016D E869FFFFFF      call quit

```

Рис. 2.9: Файл листинга lab7-2

строка 211

- 34 - номер строки
- 0000012E - адрес
- B8[00000000] - машинный код
- mov eax,max - код программы

строка 212

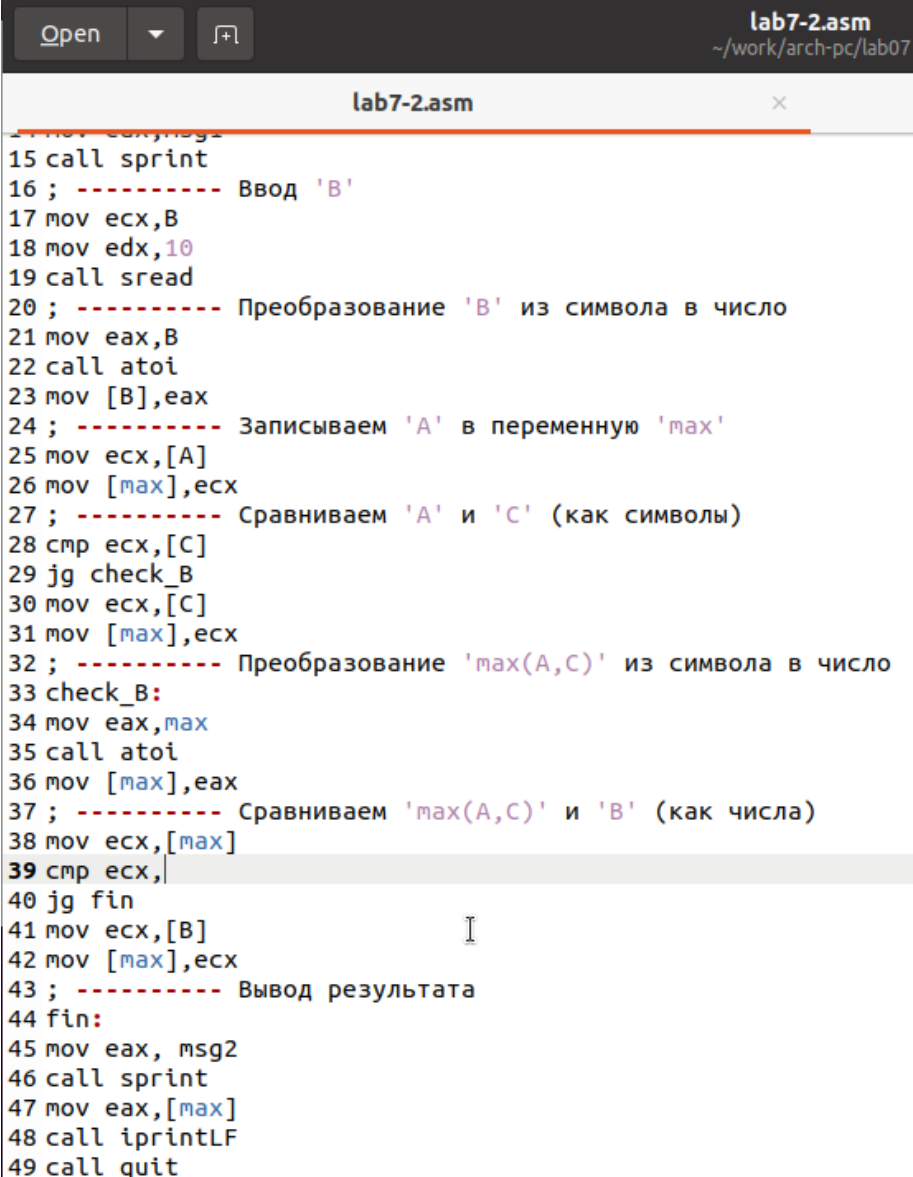
- 35 - номер строки
- 00000133 - адрес
- E864FFFFFF - машинный код

- `call atoi` - код программы

строка 213

- 36 - номер строки
- 00000138 - адрес
- `A3[00000000]` - машинный код
- `mov [max],eax` - код программы

Затем я открыла исходный файл программы `lab7-2.asm` и в одной из инструкций, где было два операнда, удалила один из них. После этого я попыталась снова ассемблировать программу, чтобы получить файл листинг.



```
lab7-2.asm
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. 2.10: Ошибка трансляции lab7-2

```

191 16 ; ----- Ввод 'В'
192 17 000000F2 B9[0A000000] mov ecx,B
193 18 000000F7 BA0A000000 mov edx,10
194 19 000000FC E842FFFFFF call sread
195 20 ; ----- Преобразование 'В' 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 'А' в переменную 'max'
200 25 00000110 8B0D[35000000] mov ecx,[A]
201 26 00000116 890D[00000000] mov [max],ecx
202 27 ; ----- Сравниваем 'А' и 'С' (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[00000000] mov [max],eax
212 37 ; ----- Сравниваем 'max(A,C)' и 'В' (как числа)
213 38 0000013F 8B0D[00000000] mov ecx,[max]
214 39 cmp ecx,
215 39 *****
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
222 46 00000158 E8B2FEFFFF call sprint
223 47 0000015D A1[00000000] mov eax,[max]
224 48 00000162 E81FFFFFFF call iprintLF
225 49 00000167 E86FFFFFFF call quit

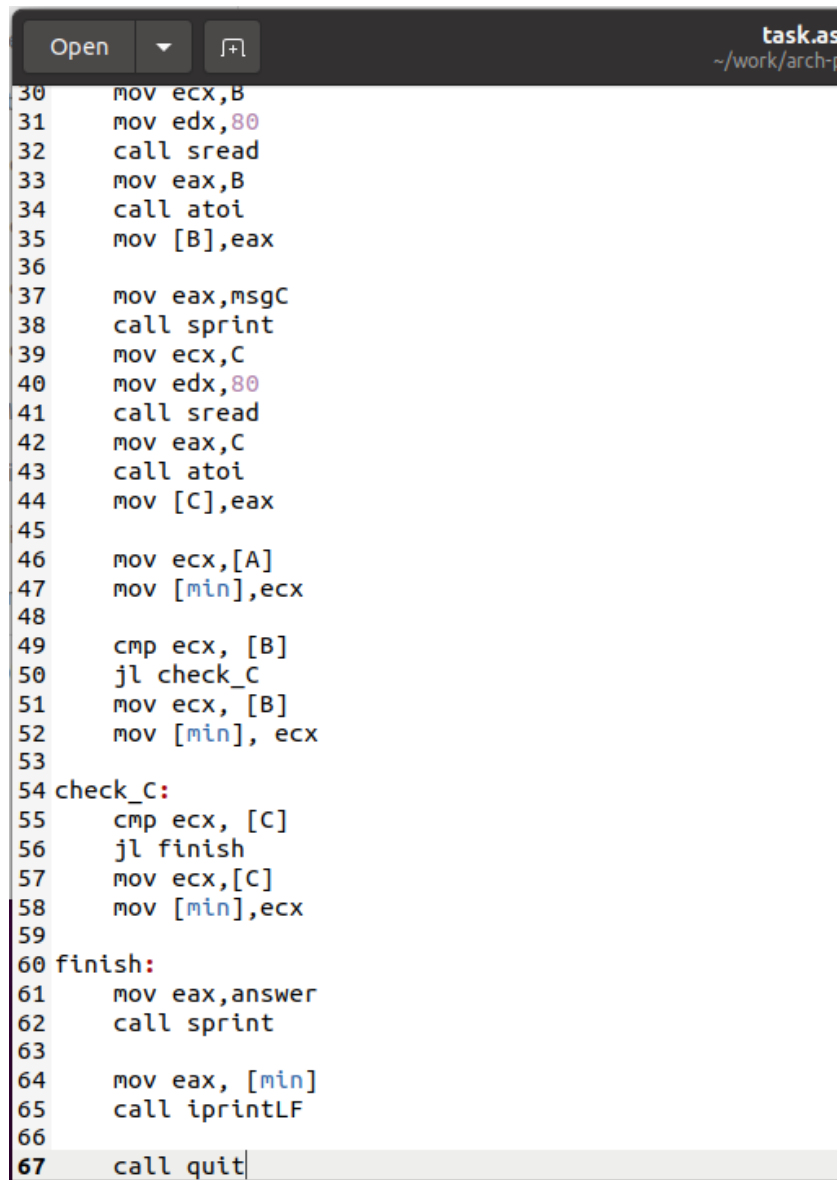
```

Рис. 2.11: Файл листинга с ошибкой lab7-2

Из-за внесённой мной ошибки объектный файл создать не удалось, однако я всё равно получила файл листинга, в котором чётко было указано, где произошла ошибка.

5. Напишите 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. Создайте исполняемый файл и проверьте его работу

для варианта 10 - 41,62,35



```
task.asm
~/work/arch-p

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
42  mov eax,C
43  call atoi
44  mov [C],eax
45
46  mov ecx,[A]
47  mov [min],ecx
48
49  cmp ecx, [B]
50  jl check_C
51  mov ecx, [B]
52  mov [min], ecx
53
54 check_C:
55  cmp ecx, [C]
56  jl finish
57  mov ecx,[C]
58  mov [min],ecx
59
60 finish:
61  mov eax,answer
62  call sprint
63
64  mov eax, [min]
65  call iprintLF
66
67  call quit
```

Рис. 2.12: Программа в файле task.asm

```

gulamova@ubuntu:~/work/arch-pc/lab07$ nasm -f elf task.asm
gulamova@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 task.o -o task
gulamova@ubuntu:~/work/arch-pc/lab07$ ./task
Input A: 41
Input B: 62
Input C: 35
Smallest: 35
gulamova@ubuntu:~/work/arch-pc/lab07$

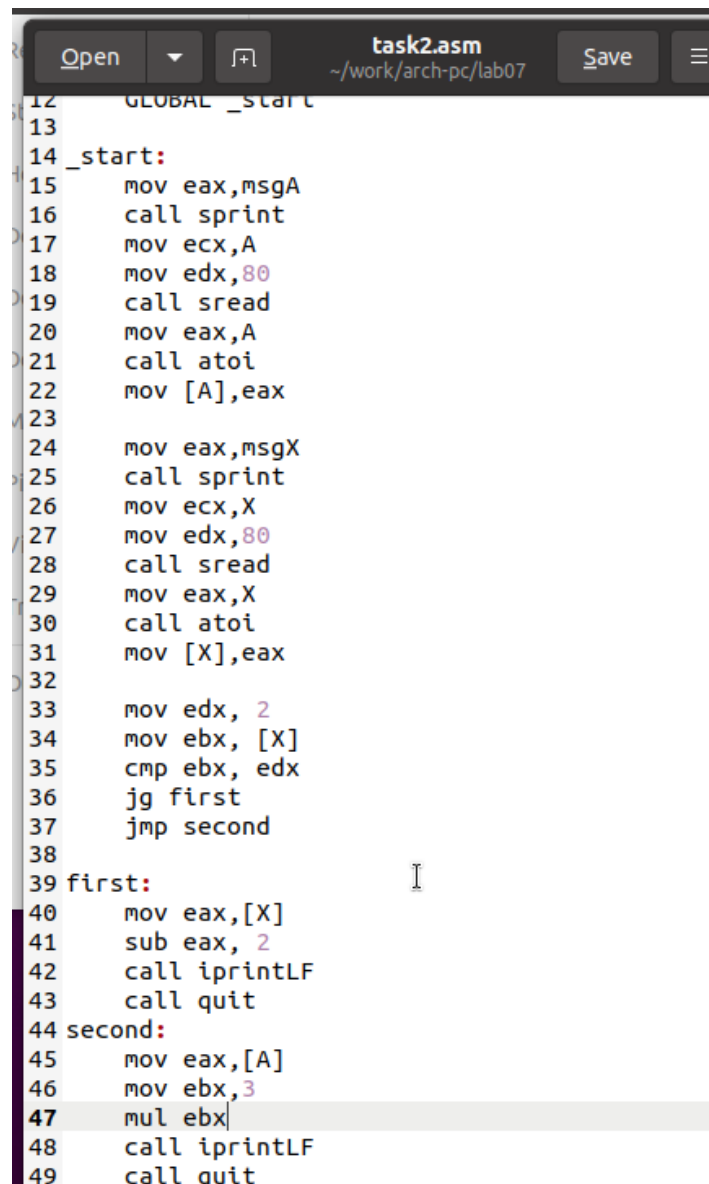
```

Рис. 2.13: Запуск программы task.asm

6. 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.

для варианта 10

$$\begin{cases} x - 2, & x > 2 \\ 3a, & x \leq 2 \end{cases}$$



```
task2.asm
~/work/arch-pc/lab07
Open Save

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 edx, 2
34     mov ebx, [X]
35     cmp ebx, edx
36     jg first
37     jmp second
38
39 first:
40     mov eax,[X]
41     sub eax, 2
42     call iprintLF
43     call quit
44 second:
45     mov eax,[A]
46     mov ebx,3
47     mul ebx
48     call iprintLF
49     call quit
```

Рис. 2.14: Программа в файле task2.asm

```
gulamova@ubuntu:~/work/arch-pc/lab07$ nasm -f elf task2.asm
gulamova@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 task2.o -o task2
gulamova@ubuntu:~/work/arch-pc/lab07$ ./task2
Input A: 0
Input X: 3
1
gulamova@ubuntu:~/work/arch-pc/lab07$ ./task2
Input A: 2
Input X: 1
6
gulamova@ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.15: Запуск программы task2.asm

3 Выводы

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