

Отчёт по лабораторной работе 7

Архитектура компьютера

Булут Умут

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Изучение структуры файлы листинга	13
2.3	Задание для самостоятельной работы	16
3	Выводы	21

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	9
2.5	Программа в файле lab7-1.asm	10
2.6	Запуск программы lab7-1.asm	11
2.7	Программа в файле lab7-2.asm	12
2.8	Запуск программы lab7-2.asm	13
2.9	Файл листинга lab7-2	14
2.10	Ошибка трансляции lab7-2	15
2.11	Файл листинга с ошибкой lab7-2	16
2.12	Программа в файле task7-1.asm	17
2.13	Запуск программы task7-1.asm	18
2.14	Программа в файле task7-2.asm	19
2.15	Запуск программы task7-2.asm	20

Список таблиц

1 Цель работы

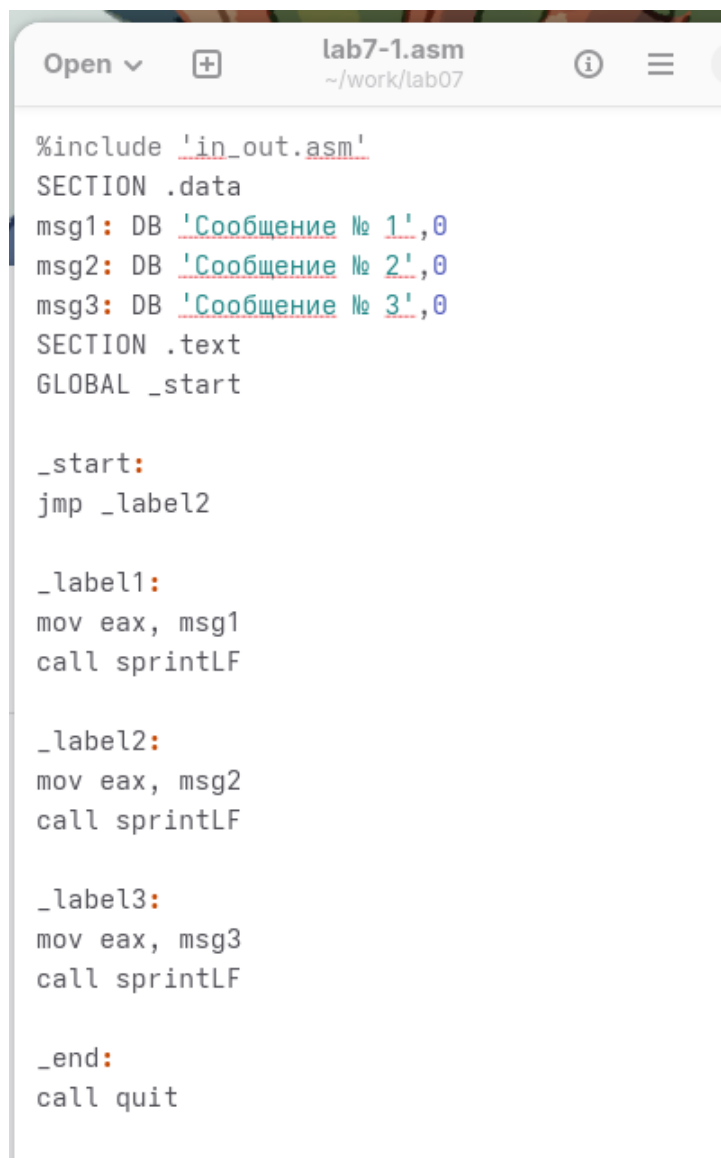
Целью работы является изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

2.1 Реализация переходов в NASM

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.



```
Open ▾  lab7-1.asm  
~/work/lab07  
  
%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 sprintfLF  
  
_label2:  
mov eax, msg2  
call sprintfLF  
  
_label3:  
mov eax, msg3  
call sprintfLF  
  
_end:  
call quit
```

Рисунок 2.1: Программа в файле lab7-1.asm

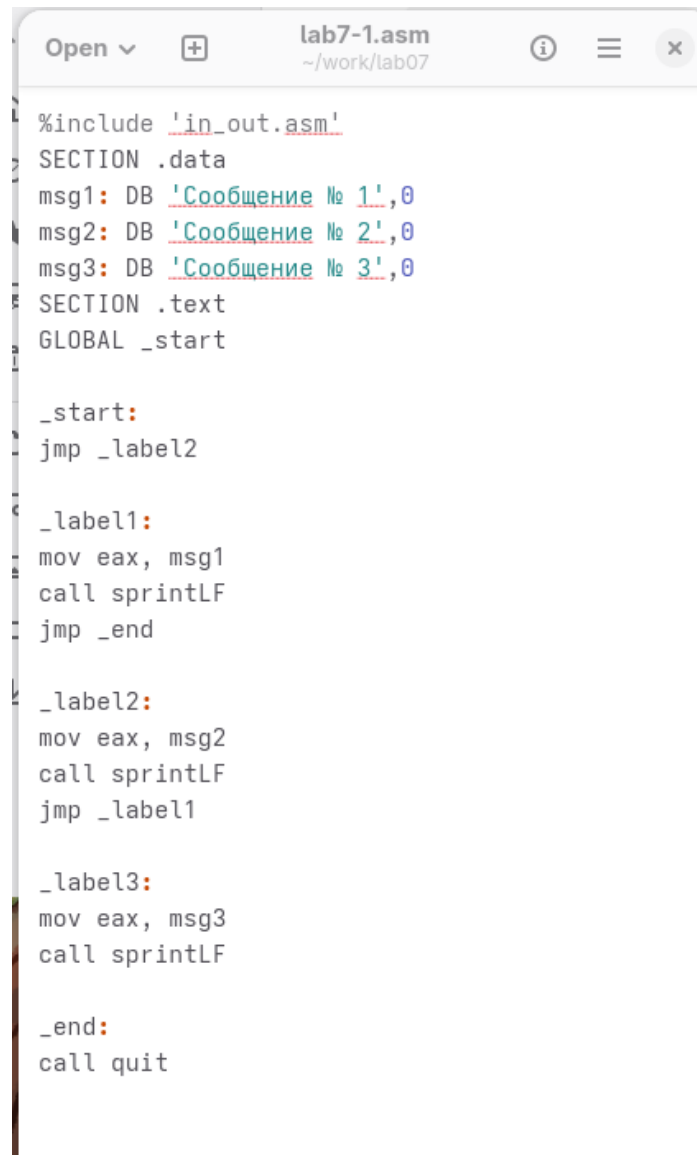
Создал исполняемый файл и запустил его.

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

Рисунок 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.



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

#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

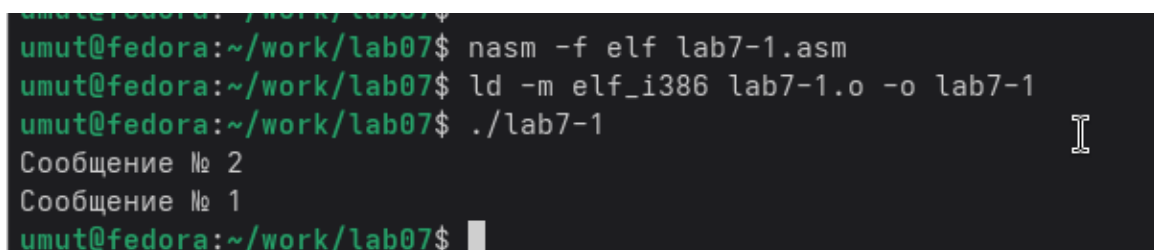
_label1:
mov eax, msg1
call sprintf
jmp _end

_label2:
mov eax, msg2
call sprintf
jmp _label1

_label3:
mov eax, msg3
call sprintf

_end:
call quit
```

Рисунок 2.3: Программа в файле lab7-1.asm



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

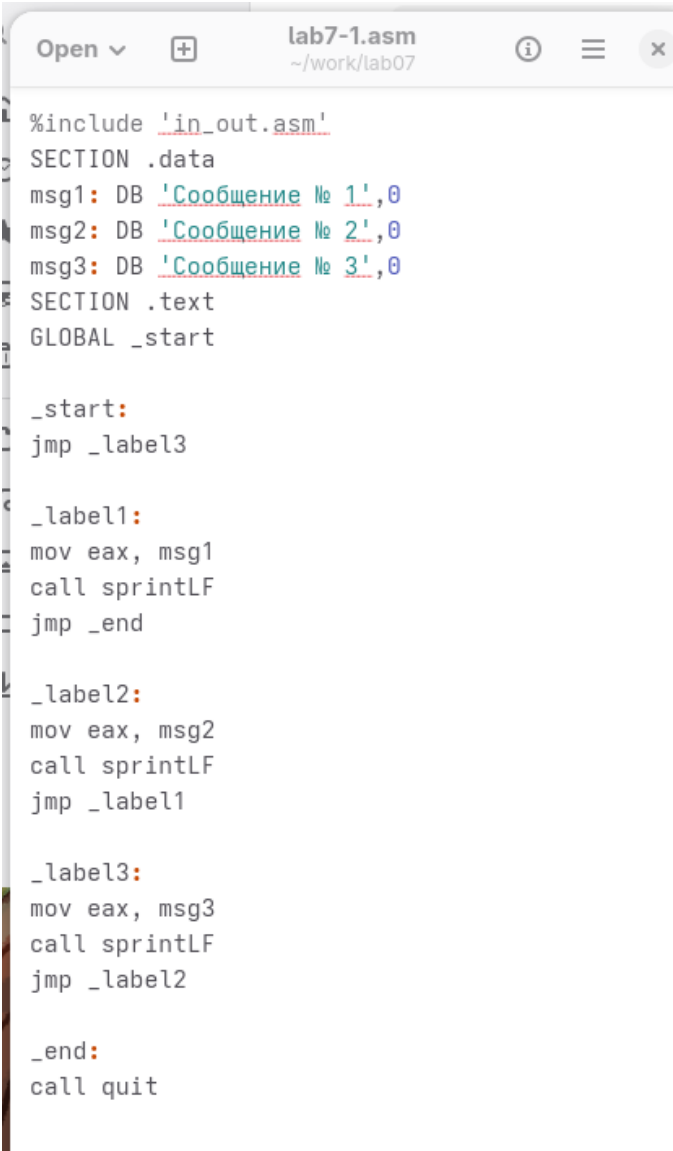
Рисунок 2.4: Запуск программы lab7-1.asm

Изменил текст программы, изменив инструкции `jmp`, чтобы вывод программы был следующим:

Сообщение № 3

Сообщение № 2

Сообщение № 1



```
%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
```

Рисунок 2.5: Программа в файле lab7-1.asm

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

Рисунок 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.



```
lab7-2.asm
~/work/lab07

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]
```

Рисунок 2.7: Программа в файле lab7-2.asm

```
umut@fedora:~/work/lab07$ nasm -f elf lab7-2.asm
umut@fedora:~/work/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
umut@fedora:~/work/lab07$ ./lab7-2
Введите B: 5
Наибольшее число: 50
umut@fedora:~/work/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
umut@fedora:~/work/lab07$
```

Рисунок 2.8: Запуск программы lab7-2.asm

2.2 Изучение структуры файлы листинга

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`

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	jb 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	jb fin
41	0000014D 8B0D[0A000000]	mov ecx,[B]
42	00000153 890D[00000000]	mov [max],ecx
43		; ----- Вывод результата
44		fin:

Рисунок 2.9: Файл листинга lab7-2

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

строка 203

- 28 - номер строки в подпрограмме
- 0000011C - адрес
- 3B0D[39000000] - машинный код
- cmp ecx,[C] - код программы - сравнивает регистр ecx и переменную C

строка 204

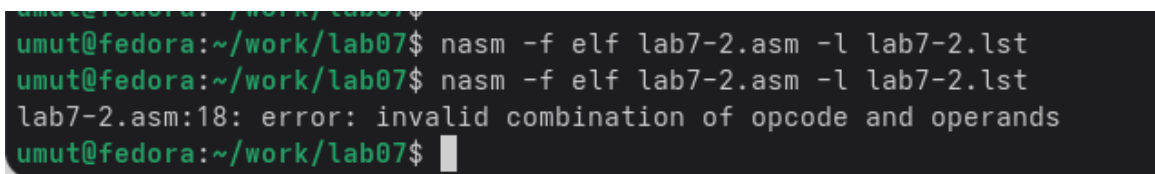
- 29 - номер строки в подпрограмме
- 00000122 - адрес

- 7F0C - машинный код
- `jb check_B` - код программы - если `>`, то переход к метке `check_B`

строка 205

- 30 - номер строки в подпрограмме
- 00000124 - адрес
- 8B0D[39000000] - машинный код
- `mov ecx,[C]` - код программы - перекладывает в регистр `ecx` значение переменной `C`

Открыл файл с программой `lab7-2.asm` и в инструкции с двумя операндами удалил один операнд. Выполнил трансляцию с получением файла листинга.



```
umut@fedora:~/work/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
umut@fedora:~/work/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:18: error: invalid combination of opcode and operands
umut@fedora:~/work/lab07$
```

Рисунок 2.10: Ошибка трансляции `lab7-2`

15 000000ED E81DFFFFFF	call sprint
16	; ----- Ввод 'B'
17 000000F2 B9[0A000000]	mov ecx,B
18	mov edx,
18 *****	error: invalid combination of opcode and operands
19 000000F7 E847FFFFFF	call sread
20	; ----- Преобразование 'B' из символа в число
21 000000FC B8[0A000000]	mov eax,B
22 00000101 E896FFFFFF	call atoi
23 00000106 A3[0A000000]	mov [B],eax
24	; ----- Записываем 'A' в переменную 'max'
25 0000010B 8B0D[35000000]	mov ecx,[A]
26 00000111 890D[00000000]	mov [max],ecx
27	; ----- Сравниваем 'A' и 'C' (как символы)
28 00000117 3B0D[39000000]	cmp ecx,[C]
29 0000011D 7F0C	jg check_B
30 0000011F 8B0D[39000000]	mov ecx,[C]
31 00000125 890D[00000000]	mov [max],ecx
32	; ----- Преобразование 'max(A,C)' из символа в число
33	check_B:
34 0000012B 8B[00000000]	

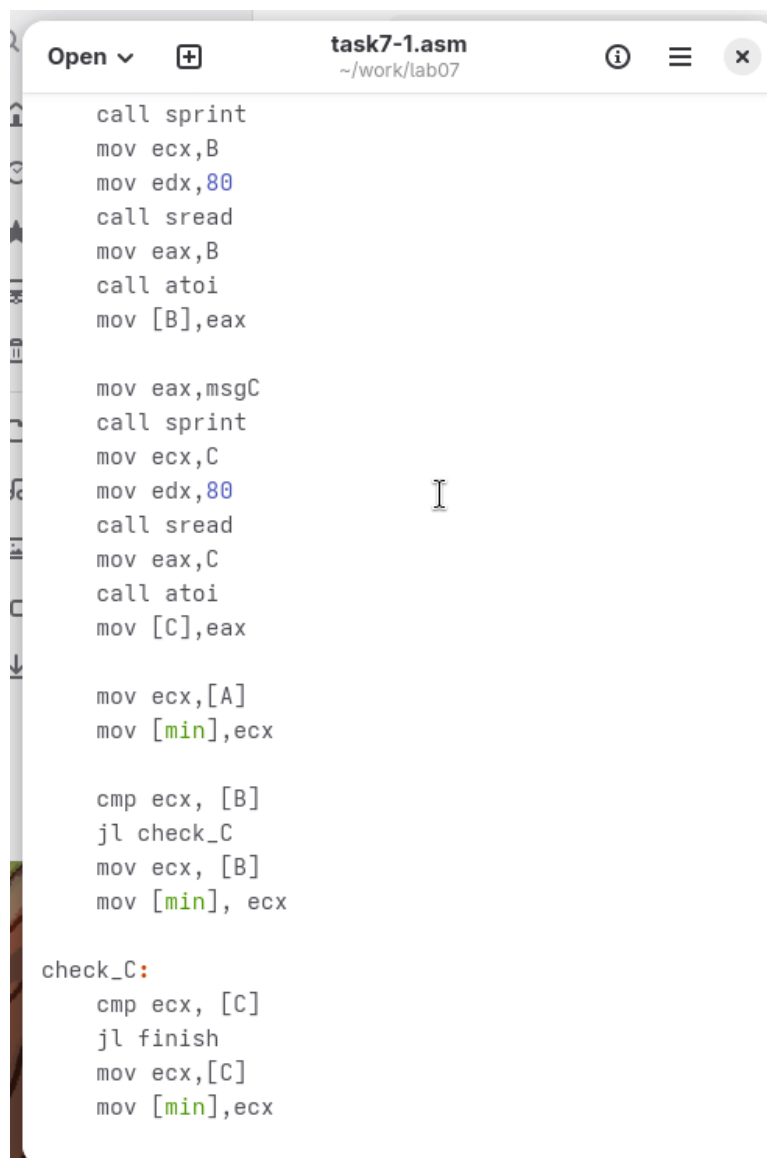
Рисунок 2.11: Файл листинга с ошибкой lab7-2

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

2.3 Задание для самостоятельной работы

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

для варианта 10 - 41,62,35



```
call sprint
mov ecx,B
mov edx,80
call sread
mov eax,B
call atoi
mov [B],eax

mov eax,msgC
call sprint
mov ecx,C
mov edx,80
call sread
mov eax,C
call atoi
mov [C],eax

mov ecx,[A]
mov [min],ecx

cmp ecx, [B]
jl check_C
mov ecx, [B]
mov [min], ecx

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

Рисунок 2.12: Программа в файле task7-1.asm

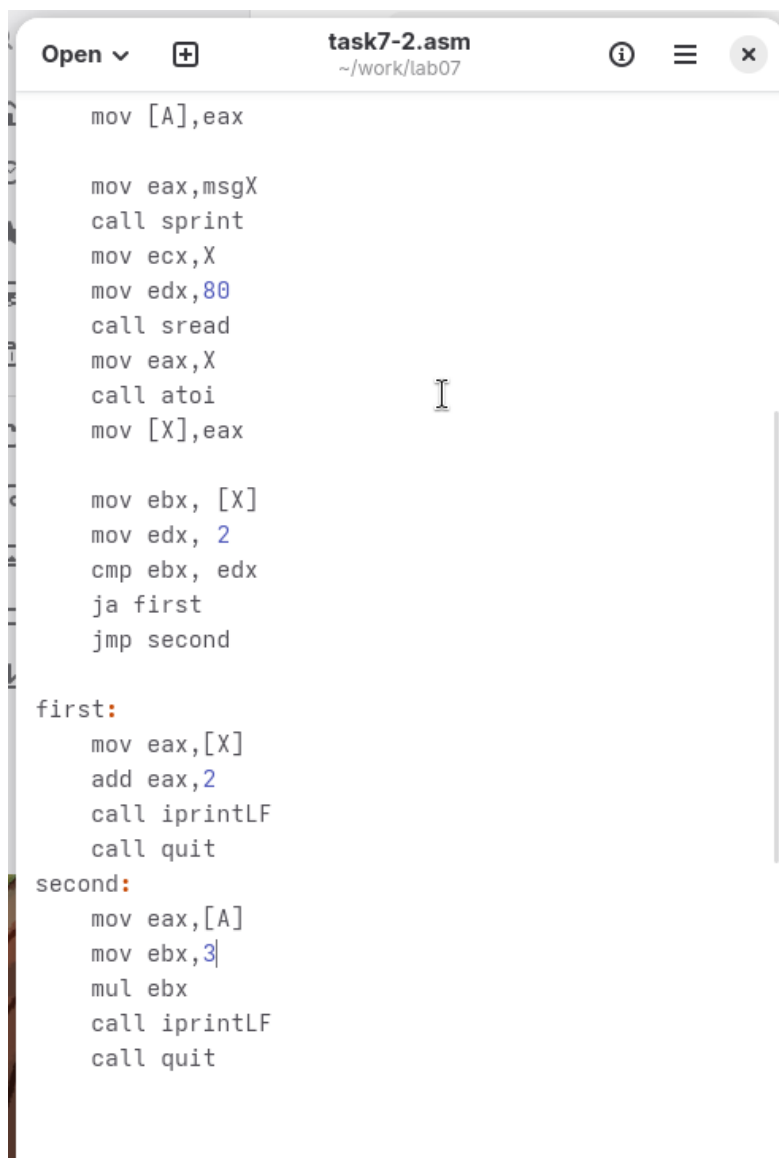
```
umut@fedora:~/work/lab07$  
umut@fedora:~/work/lab07$ nasm -f elf task7-1.asm  
umut@fedora:~/work/lab07$ ld -m elf_i386 task7-1.o -o task7-1  
umut@fedora:~/work/lab07$ ./task7-1  
Input A: 41  
Input B: 62  
Input C: 35  
Smallest: 35  
umut@fedora:~/work/lab07$
```

Рисунок 2.13: Запуск программы task7-1.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.

для варианта 10

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



```
task7-2.asm
~/work/lab07

mov [A],eax

mov eax,msgX
call sprint
mov ecx,X
mov edx,80
call sread
mov eax,X
call atoi
mov [X],eax

mov ebx, [X]
mov edx, 2
cmp ebx, edx
ja first
jmp second

first:
mov eax,[X]
add eax,2
call iprintLF
call quit

second:
mov eax,[A]
mov ebx,3
mul ebx
call iprintLF
call quit
```

Рисунок 2.14: Программа в файле task7-2.asm

```
umut@fedora:~/work/lab07$  
umut@fedora:~/work/lab07$ nasm -f elf task7-2.asm  
umut@fedora:~/work/lab07$ ld -m elf_i386 task7-2.o -o task7-2  
umut@fedora:~/work/lab07$ ./task7-2  
Input A: 3  
Input X: 0  
9  
umut@fedora:~/work/lab07$ ./task7-2  
Input A: 0  
Input X: 3  
5  
umut@fedora:~/work/lab07$ ./task7-2  
Input A: 2  
Input X: 1  
6  
umut@fedora:~/work/lab07$
```

Рисунок 2.15: Запуск программы task7-2.asm

3 Выводы

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