

# **Отчёт по лабораторной работе 7**

**Архитектура компьютера**

Тумуреева Галина Аркадьевна

# Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	22

## 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 . . . . .	16
2.11	Файл листинга с ошибкой lab7-2 . . . . .	17
2.12	Программа в файле task.asm . . . . .	18
2.13	Запуск программы task.asm . . . . .	19
2.14	Программа в файле task2.asm . . . . .	20
2.15	Запуск программы task2.asm . . . . .	21

## Список таблиц

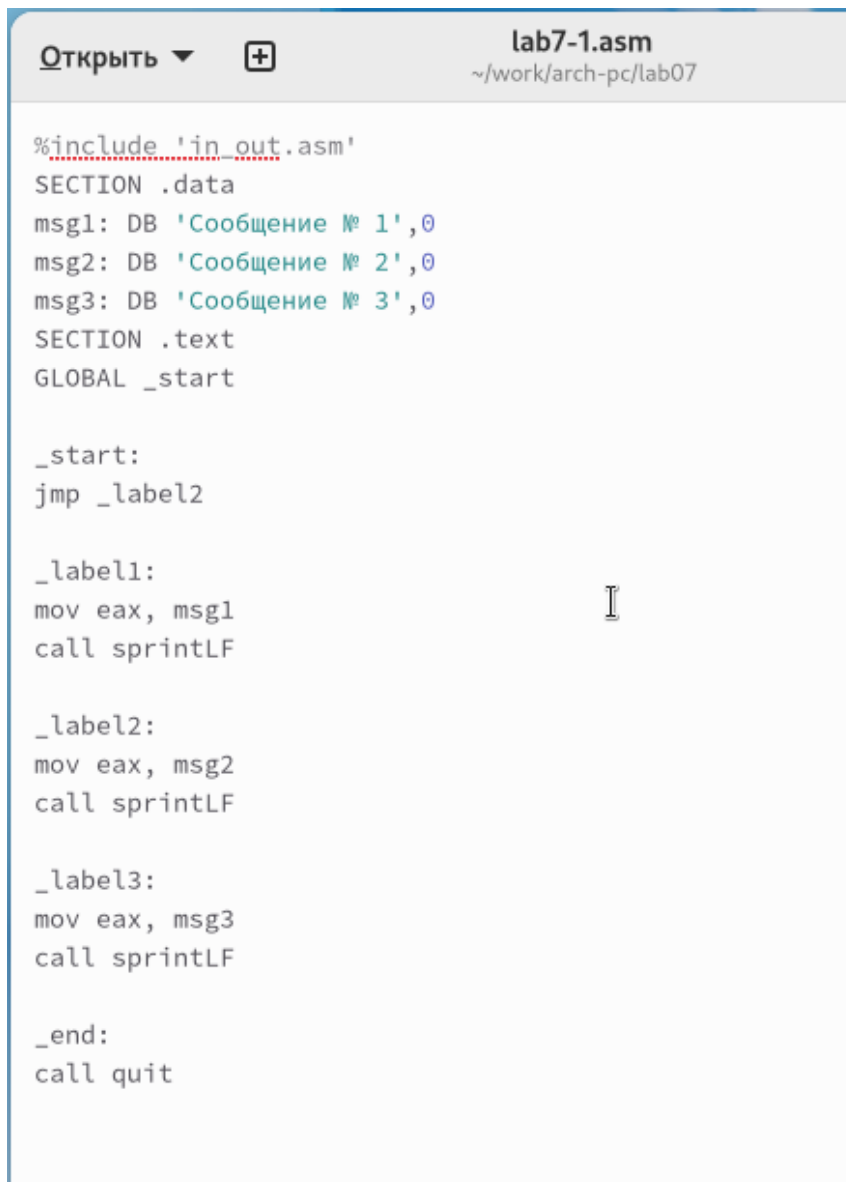
# 1 Цель работы

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.



```
%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. 2.1: Программа в файле lab7-1.asm

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

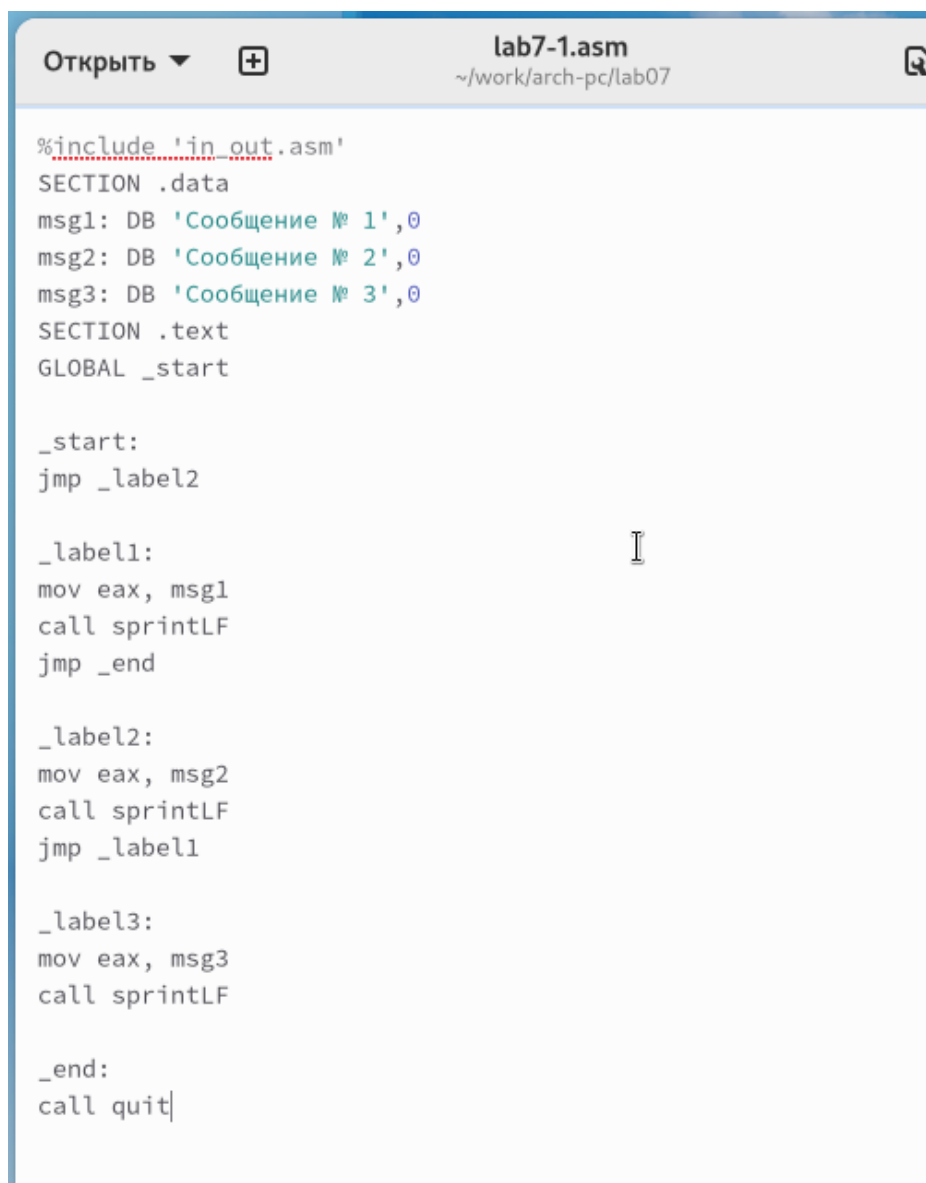
```
[gtumureeva@gtumureeva lab07]$ nasm -f elf lab7-1.asm
[gtumureeva@gtumureeva lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1
[gtumureeva@gtumureeva lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 3
[gtumureeva@gtumureeva lab07]$
```

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.








```
Открыть ▾  lab7-1.asm  
~/work/arch-pc/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
```

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

```
[gtumureeva@gtumureeva lab07]$ nasm -f elf lab7-1.asm
[gtumureeva@gtumureeva lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1
[gtumureeva@gtumureeva lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 1
[gtumureeva@gtumureeva lab07]$
```

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

Изменила команды `jmp` для изменения порядка вывода сообщений программой.

Сообщение № 3

Сообщение № 2

Сообщение № 1

Открыть ▾

+

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

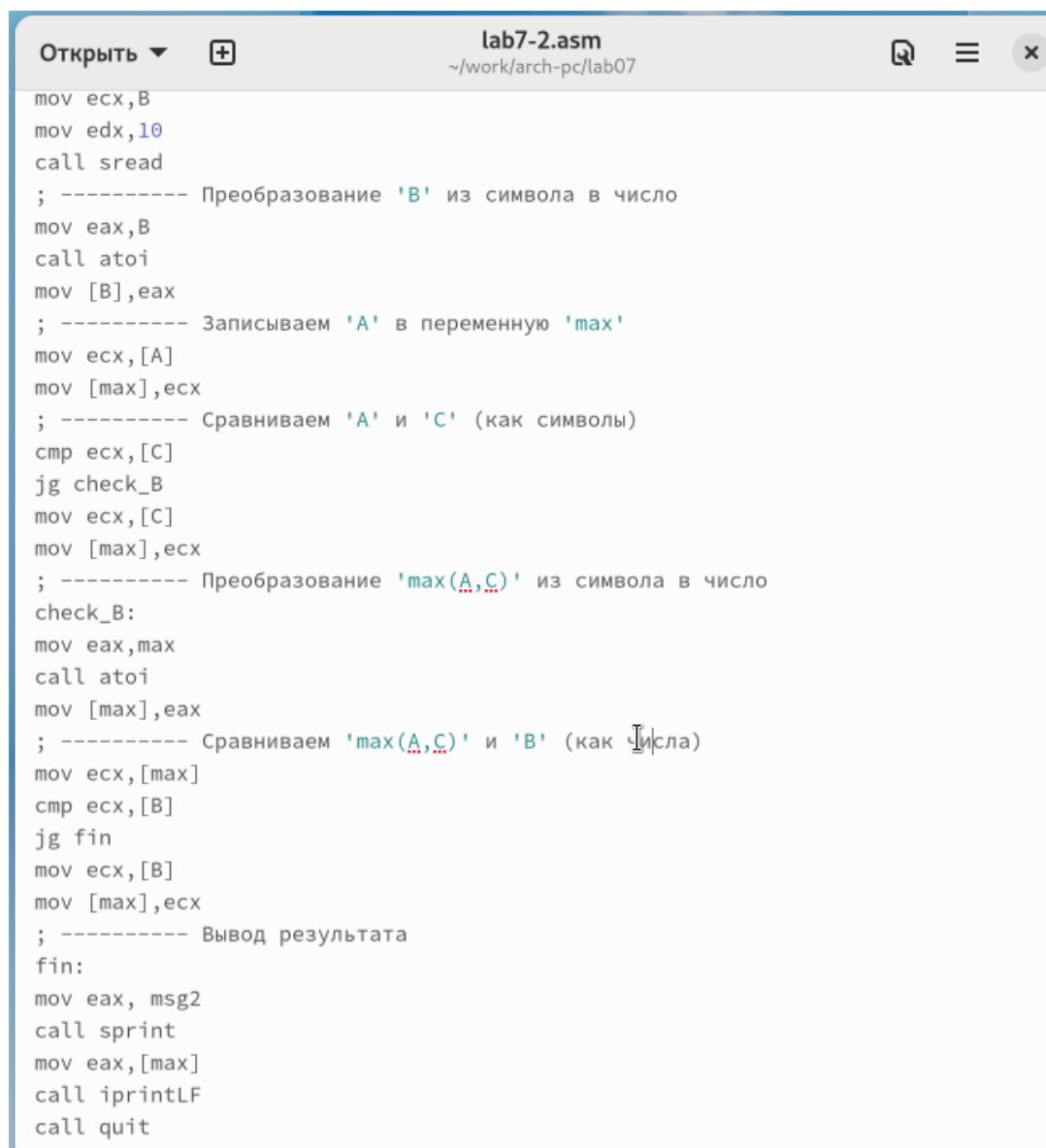
Рис. 2.5: Программа в файле lab7-1.asm

```
[gtumureeva@gtumureeva lab07]$  
[gtumureeva@gtumureeva lab07]$ nasm -f elf lab7-1.asm  
[gtumureeva@gtumureeva lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1  
[gtumureeva@gtumureeva lab07]$ ./lab7-1  
Сообщение № 3  
Сообщение № 2  
Сообщение № 1  
[gtumureeva@gtumureeva 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: A, B и C, чтобы выявить и показать на экране самое большое из них. Я заранее задала значения для A и C, а значение для B программа получает от пользователя через ввод с клавиатуры.

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



```
Открыть ▾ + lab7-2.asm ~/work/arch-pc/lab07
mov ecx,B
mov edx,10
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 quit
```

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

```
[gtumureeva@gtumureeva lab07]$ nasm -f elf lab7-2.asm
[gtumureeva@gtumureeva lab07]$ ld -m elf_i386 lab7-2.o -o lab7-2
[gtumureeva@gtumureeva lab07]$ ./lab7-2
Введите В: 20
Наибольшее число: 50
[gtumureeva@gtumureeva lab07]$ ./lab7-2
Введите В: 55
Наибольшее число: 55
[gtumureeva@gtumureeva lab07]$ ./lab7-2
Введите В: 80
Наибольшее число: 80
[gtumureeva@gtumureeva 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.

```
195 19 0000001C E842111111 call $+5
196 20 ; ----- Преобразование 'B' из символа в число
197 21 00000101 B8[0A000000] mov eax,B
198 22 00000106 E891FFFFFF call atoi
199 23 0000010B A3[0A000000] mov [B],eax
200 24 ; ----- Записываем 'A' в переменную 'max'
201 25 00000110 8B0D[35000000] mov ecx,[A]
202 26 00000116 890D[00000000] mov [max],ecx
203 27 ; ----- Сравниваем 'A' и 'C' (как символы)
204 28 0000011C 3B0D[39000000] cmp ecx,[C]
205 29 00000122 7F0C jg check_B
206 30 00000124 8B0D[39000000] mov ecx,[C]
207 31 0000012A 890D[00000000] mov [max],ecx
208 32 ; ----- Преобразование 'max(A,C)' из символа в число
209 33 check_B:
210 34 00000130 B8[00000000] mov eax,max
211 35 00000135 E862FFFFFF call atoi
212 36 0000013A A3[00000000] mov [max],eax
213 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
214 38 0000013F 8B0D[00000000] mov ecx,[max]
215 39 00000145 3B0D[0A000000] cmp ecx,[B]
216 40 0000014B 7F0C jg fin
217 41 0000014D 8B0D[0A000000] mov ecx,[B]
218 42 00000153 890D[00000000] mov [max],ecx
219 43 ; ----- Вывод результата
220 44 fin:
221 45 00000159 B8[13000000] mov eax,msg2
222 46 0000015E E8ACFFFFFF call sprint
223 47 00000163 A1[00000000] mov eax,[max]
224 48 00000168 E819FFFFFF call iprintLF
225 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.

```
[gtumureeva@gtumureeva lab07]$  
[gtumureeva@gtumureeva lab07]$ nasm -f elf lab7-2.asm -l lab7-2.lst  
[gtumureeva@gtumureeva lab07]$ nasm -f elf lab7-2.asm -l lab7-2.lst  
lab7-2.asm:34: error: invalid combination of opcode and operands  
[gtumureeva@gtumureeva lab07]$  
[gtumureeva@gtumureeva lab07]$
```

Рис. 2.10: Ошибка трансляции lab7-2



```
lab7-2.asm
197 21 00000101 B8[0A000000] mov eax,B
198 22 00000106 E891FFFFFF call atoi
199 23 0000010B A3[0A000000] mov [B],eax
200 24 ; ----- Записываем 'A' в переменную 'max'
201 25 00000110 8B0D[35000000] mov ecx,[A]
202 26 00000116 890D[00000000] mov [max],ecx
203 27 ; ----- Сравниваем 'A' и 'C' (как символы)
204 28 0000011C 3B0D[39000000] cmp ecx,[C]
205 29 00000122 7F0C jg check_B
206 30 00000124 8B0D[39000000] mov ecx,[C]
207 31 0000012A 890D[00000000] mov [max],ecx
208 32 ; ----- Преобразование 'max(A,C)' из символа в число
209 33 check_B:
210 34 mov eax,
211 34 ***** error: invalid combination of opcode and operands
212 35 00000130 E867FFFFFF call atoi
213 36 00000135 A3[00000000] mov [max],eax
214 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
215 38 0000013A 8B0D[00000000] mov ecx,[max]
216 39 00000140 3B0D[0A000000] cmp ecx,[B]
217 40 00000146 7F0C jg fin
218 41 00000148 8B0D[0A000000] mov ecx,[B]
219 42 0000014E 890D[00000000] mov [max],ecx
220 43 ; ----- Вывод результата
221 44 fin:
222 45 00000154 B8[13000000] mov eax,msg2
223 46 00000159 E8B1FFFFFF call sprint
224 47 0000015E A1[00000000] mov eax,[max]
225 48 00000163 E81EFFFFFF call iprintLF
226 49 00000168 E86EFFFFFF 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 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 13 - 84,32,77

```
task.asm
~/.work/arch-pc/lab07

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

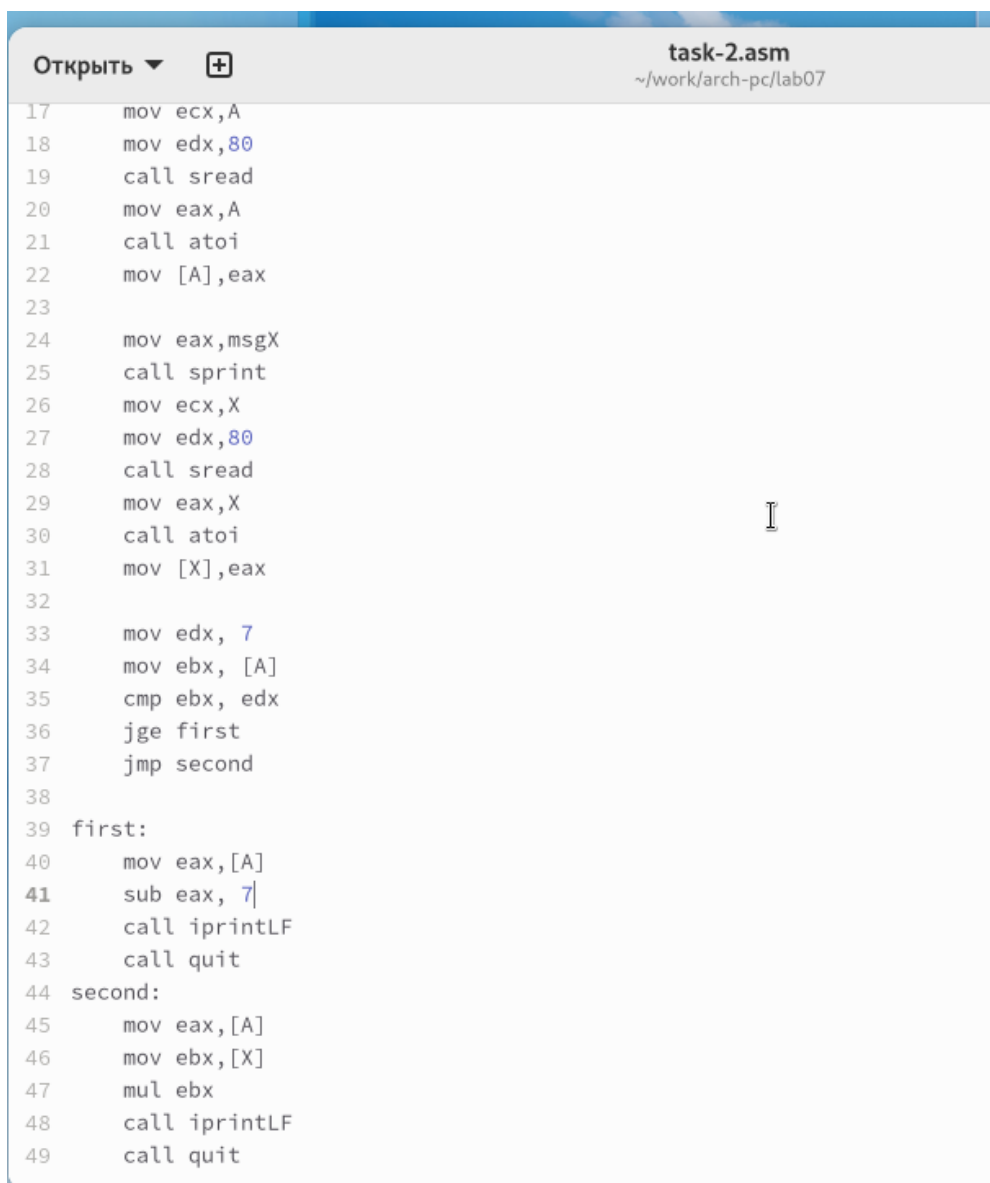
```
[gtumureeva@gtumureeva lab07]$ nasm -f elf task.asm
[gtumureeva@gtumureeva lab07]$ ld -m elf_i386 task.o -o task
[gtumureeva@gtumureeva lab07]$ ./task
Input A: 84
Input B: 32
Input C: 77
Smallest: 32
[gtumureeva@gtumureeva lab07]$
```

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 13

$$\begin{cases} a - 7, a \geq 7 \\ ax, a < 7 \end{cases}$$



```
task-2.asm
~/work/arch-pc/lab07

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, 7
34    mov ebx, [A]
35    cmp ebx, edx
36    jge first
37    jmp second
38
39 first:
40    mov eax,[A]
41    sub eax, 7
42    call iprintLF
43    call quit
44 second:
45    mov eax,[A]
46    mov ebx,[X]
47    mul ebx
48    call iprintLF
49    call quit
```

Рис. 2.14: Программа в файле task2.asm

```
[gtumureeva@gtumureeva lab07]$ nasm -f elf task-2.asm
[gtumureeva@gtumureeva lab07]$ ld -m elf_i386 task-2.o -o task-2
[gtumureeva@gtumureeva lab07]$ ./task-
bash: ./task-: Нет такого файла или каталога
[gtumureeva@gtumureeva lab07]$ ./task-2
Input A: 9
Input X: 3
2
[gtumureeva@gtumureeva lab07]$ ./task-2
Input A: 4
Input X: 6
24
[gtumureeva@gtumureeva lab07]$
```

Рис. 2.15: Запуск программы task2.asm

## 3 Выводы

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