

# **Отчёт по лабораторной работе 7**

**Архитектура компьютера**

Булут Умут

# **Содержание**

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

# **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**

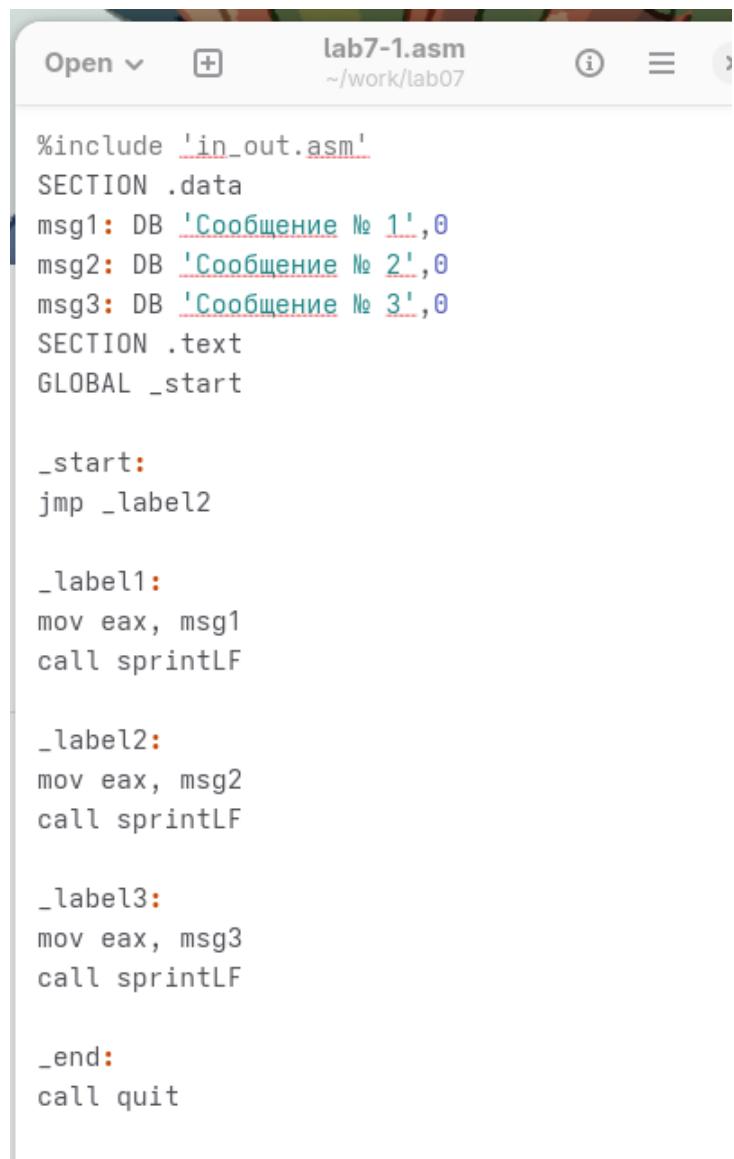
Целью работы является изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

## **2 Выполнение лабораторной работы**

### **2.1 Реализация переходов в NASM**

Создал каталог для программам 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 sprintLF

_label2:
mov eax, msg2
call sprintLF

_label3:
mov eax, msg3
call sprintLF

_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.

The screenshot shows a code editor window with the file 'lab7-1.asm' open. The code is written in assembly language using the AT&T syntax. It includes three data sections ('msg1', 'msg2', 'msg3') containing Russian strings. The program starts at the '\_start' label, prints 'msg1', then loops between '\_label1' and '\_label2', printing 'msg2' each time it loops. Finally, it prints 'msg3' once and exits at the '\_end' label.

```
%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
```

Рисунок 2.3: Программа в файле lab7-1.asm

The terminal window shows the user's actions to compile and run the assembly program:

- Compiling: `nasm -f elf lab7-1.asm`
- Linking: `ld -m elf_i386 lab7-1.o -o lab7-1`
- Running: `./lab7-1`

The output of the program is displayed in the terminal, showing the three messages sequentially.

```
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$ █
```

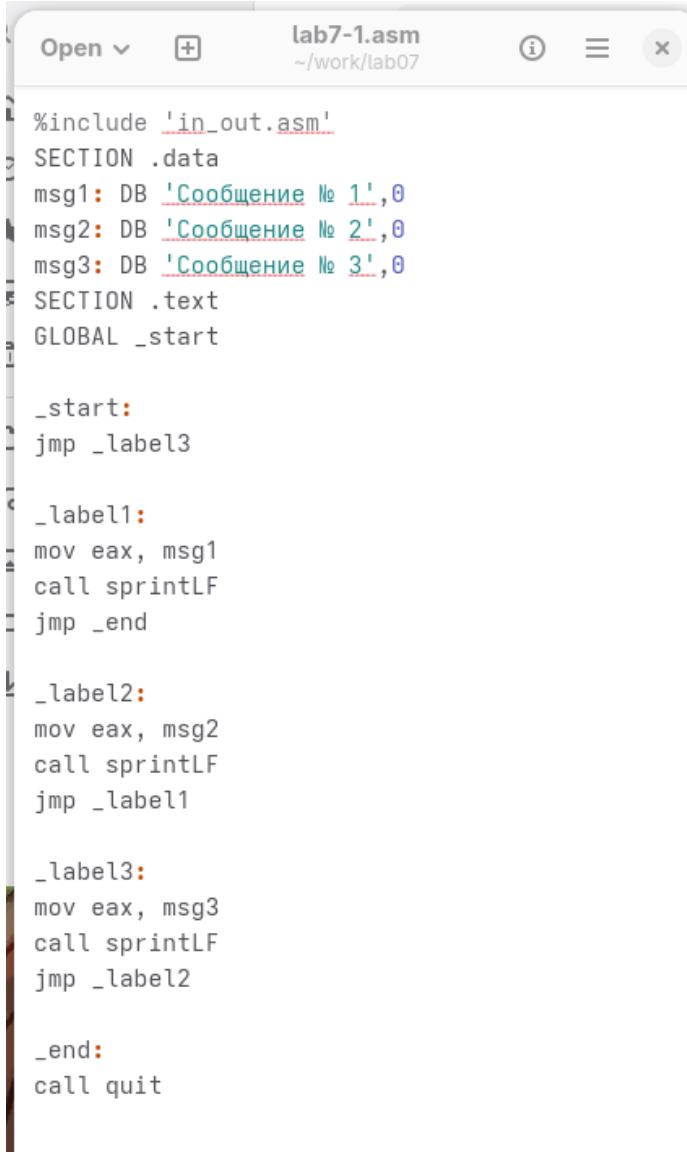
Рисунок 2.4: Запуск программы lab7-1.asm

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

Сообщение № 3

Сообщение № 2

Сообщение № 1



```
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 _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: A, B и C. Значения для A и C задаются в программе, значение B вводиться с клавиатуры.

Создал исполняемый файл и проверил его работу для разных значений B.

The screenshot shows a code editor window titled "lab7-2.asm" with the path "~/work/lab07". The assembly code is as follows:

```
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 В: 5
Наибольшее число: 50
umut@fedora:~/work/lab07$ ./lab7-2
Введите В: 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]
18 000000F7 BA0A000000
19 000000FC E842FFFFFF
20 ; ----- Преобразование 'B' из символа в число
21 00000101 B8[0A000000]
22 00000106 E891FFFFFF
23 0000010B A3[0A000000]
24 ; ----- Записываем 'A' в переменную 'max'
25 00000110 8B0D[35000000]
26 00000116 890D[00000000]
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 0000011C 3B0D[39000000]
29 00000122 7F0C
30 00000124 8B0D[39000000]
31 0000012A 890D[00000000]
32 ; ----- Преобразование 'max(A,C)' из символа в число
33
34 00000130 B8[00000000]
35 00000135 E862FFFFFF
36 0000013A A3[00000000]
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013F 8B0D[00000000]
39 00000145 3B0D[0A000000]
40 0000014B 7F0C
41 0000014D 8B0D[0A000000]
42 00000153 890D[00000000]
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

- 7FOC - машинный код
- jg 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 operand. 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
16
17 000000F2 B9[0A000000]
18
18      ****
19 000000F7 E847FFFFFF
20
21 000000FC B8[0A000000]
22 00000101 E896FFFFFF
23 00000106 A3[0A000000]
24
25 0000010B 8B0D[35000000]
26 00000111 890D[00000000]
27
28 00000117 3B0D[39000000]
29 0000011D 7F0C
30 0000011F 8B0D[39000000]
31 00000125 890D[00000000]
32
33
34 00000129 8B0D[00000000]

```

```

-----+
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,
error: invalid combination of opcode and operands
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:

```

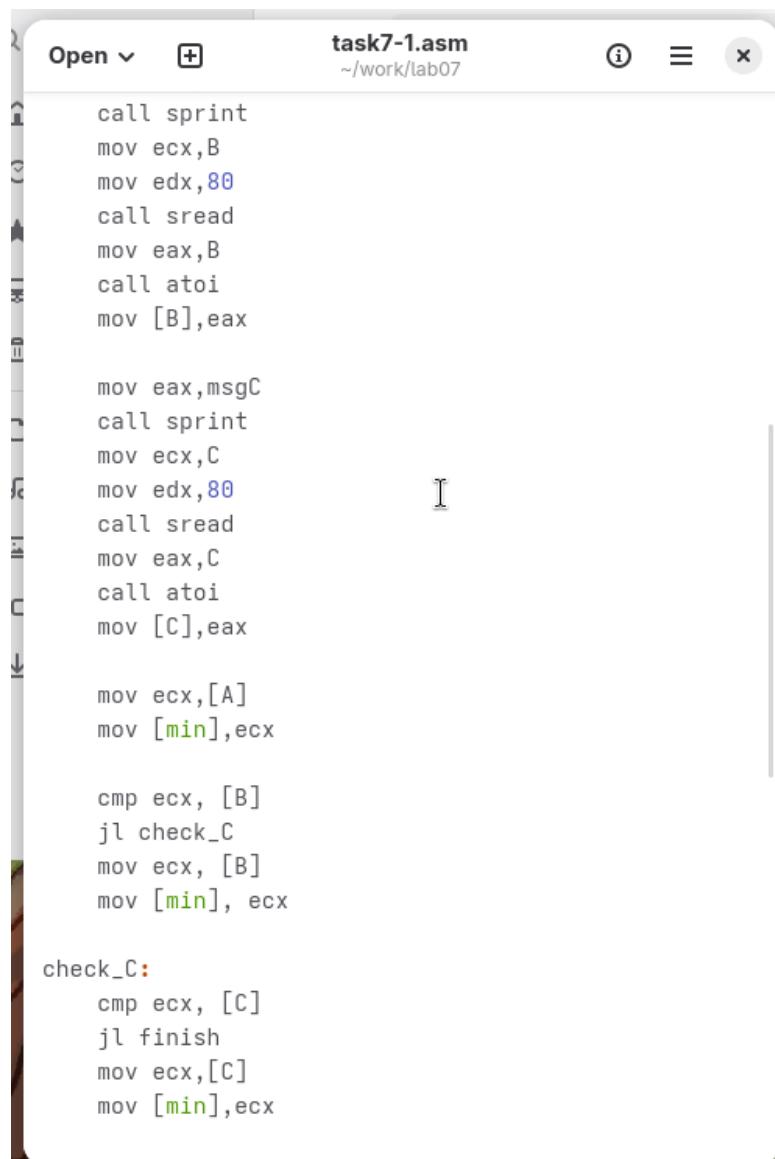
Рисунок 2.11: Файл листинга с ошибкой lab7-2

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



The screenshot shows a text editor window titled "task7-1.asm" with the path "~/work/lab07". The code is written in assembly language:

```
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$ 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

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

The screenshot shows a terminal window titled "task7-2.asm" with the path "~/work/lab07". The code is written in assembly language:

```
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$ 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.