

# **Отчёт по лабораторной работе №7**

**Дисциплина: Архитектура компьютера**

Филатов Илья Гурамович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
4.1	Реализация переходов в NASM . . . . .	8
4.2	Изучение структуры файлы листинга . . . . .	13
4.3	Задание для самостоятельной работы . . . . .	14
<b>5</b>	<b>Выводы</b>	<b>23</b>
<b>6</b>	<b>Список литературы</b>	<b>24</b>

# Список иллюстраций

4.1	Создание каталога и файла . . . . .	8
4.2	Ввод программы . . . . .	9
4.3	Копирование файла . . . . .	9
4.4	Создание и запуск исполняемого файла . . . . .	9
4.5	Изменение текста программы . . . . .	10
4.6	Создание и запуск исполняемого файла . . . . .	10
4.7	Изменение текста программы . . . . .	11
4.8	Создание и запуск исполняемого файла . . . . .	11
4.9	Создание файла . . . . .	11
4.10	Ввод текста программы . . . . .	12
4.11	Создание и запуск исполняемого файла . . . . .	12
4.12	Создание файла листинга . . . . .	13
4.13	Файл листинга . . . . .	13
4.14	Изменение текста программы . . . . .	13
4.15	Создание файла листинга . . . . .	14
4.16	Файл с ошибкой . . . . .	14
4.17	Программа нахождения наименьшей из 3 целочисленных перемен- ных . . . . .	15
4.18	Создание и запуск исполняемого файла . . . . .	18
4.19	Программа вычисления значения функции . . . . .	19
4.20	Создание и запуск исполняемого файла . . . . .	22

# 1 Цель работы

Изучить команды условного и безусловного переходов, приобрести навыки написания программ с использованием переходов, ознакомиться с назначением и структурой файла листинга.

## **2 Задание**

1. Реализация переходов в NASM
2. Изучение структуры файлы листинга
3. Задание для самостоятельной работы

### 3 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.
- безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

Безусловный переход выполняется инструкцией `jmp` (от англ. `jump` – прыжок), которая включает в себя адрес перехода, куда следует передать управление.

Адрес перехода может быть либо меткой, либо адресом области памяти, в которую предварительно помещен указатель перехода. Кроме того, в качестве операнда можно использовать имя регистра, в таком случае переход будет осуществляться по адресу, хранящемуся в этом регистре.

Флаг – это бит, принимающий значение 1 («флаг установлен»), если выполнено некоторое условие, и значение 0 («флаг сброшен») в противном случае. Флаги работают независимо друг от друга, и лишь для удобства они помещены в единый регистр – регистр флагов, отражающий текущее состояние процессора. В следующей таблице указано положение битовых флагов в регистре флагов.

Флаги состояния (биты 0, 2, 4, 6, 7 и 11) отражают результат выполнения арифметических инструкций, таких как `ADD`, `SUB`, `MUL`, `DIV`.

Инструкция `cmp` является одной из инструкций, которая позволяет сравнить операнды и выставляет флаги в зависимости от результата сравнения.

Инструкция `сmp` является командой сравнения двух операндов и имеет такой же формат, как и команда вычитания.

Команда `сmp`, так же как и команда вычитания, выполняет вычитание, но результат вычитания никуда не записывается и единственным результатом команды сравнения является формирование флагов.

Команда условного перехода имеет вид

`j(мнемоника перехода) label`

Мнемоника перехода связана со значением анализируемых флагов или со способом формирования этих флагов. В табл. 7.3. представлены команды условного перехода, которые обычно ставятся после команды сравнения `сmp`. В их мнемониках указывается тот результат сравнения, при котором надо делать переход.

Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную информацию.

Все ошибки и предупреждения, обнаруженные при ассемблировании, транслятор выводит на экран, и файл листинга не создаётся.

Структура листинга:

- номер строки — это номер строки файла листинга.
- адрес — это смещение машинного кода от начала текущего сегмента.
- машинный код представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности инструкций.
- исходный текст программы — это строка исходной программы вместе с комментариями.

## 4 Выполнение лабораторной работы

### 4.1 Реализация переходов в NASM

Открываю терминал. Создаю каталог для работы lab07 и перехожу в него. Создаю в нём файл lab7-1.asm (рис. 4.1).

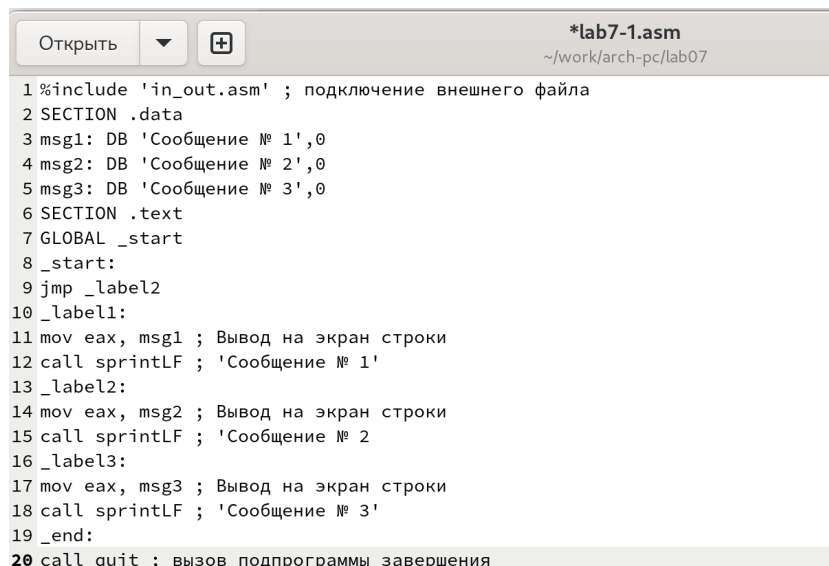
A screenshot of a terminal window with a dark background. The title bar at the top reads "igfilatov@igfilatov:~/work/arch-pc/lab07". The terminal shows the following commands and their outputs:

```
[igfilatov@igfilatov ~]$ mkdir ~/work/arch-pc/lab07
[igfilatov@igfilatov ~]$ cd ~/work/arch-pc/lab07
[igfilatov@igfilatov lab07]$ touch lab7-1.asm
[igfilatov@igfilatov lab07]$
```

Рис. 4.1: Создание каталога и файла

Открываю файл с помощью редактора gedit и ввожу текст программы из листинга 7.1 (рис. 4.2).



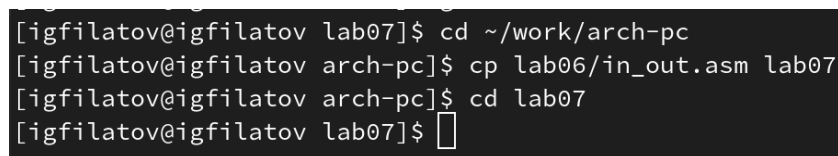


```
*lab7-1.asm
~/work/arch-pc/lab07

1 %include 'in_out.asm' ; подключение внешнего файла
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 _start:
9 jmp _label2
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintf ; 'Сообщение № 1'
13 _label2:
14 mov eax, msg2 ; Вывод на экран строки
15 call sprintf ; 'Сообщение № 2'
16 _label3:
17 mov eax, msg3 ; Вывод на экран строки
18 call sprintf ; 'Сообщение № 3'
19 _end:
20 call quit ; вызов подпрограммы завершения
```

Рис. 4.2: Ввод программы

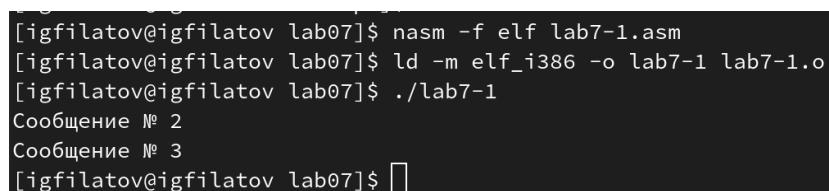
Чтобы программа, которая содержит подпрограммы из `in_out.asm`, работала корректно, копирую этот файл из каталога `lab06` в каталог `lab07` (рис. 4.3).



```
[igfilatov@igfilatov lab07]$ cd ~/work/arch-pc
[igfilatov@igfilatov arch-pc]$ cp lab06/in_out.asm lab07
[igfilatov@igfilatov arch-pc]$ cd lab07
[igfilatov@igfilatov lab07]$
```

Рис. 4.3: Копирование файла

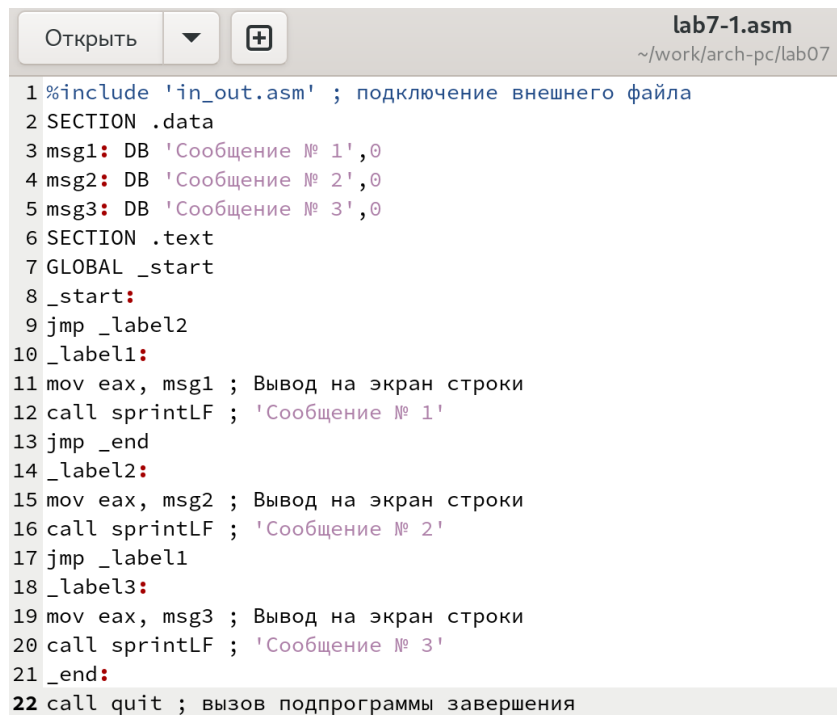
Создаю исполняемый файл и запускаю его (рис. 4.4).



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

Рис. 4.4: Создание и запуск исполняемого файла

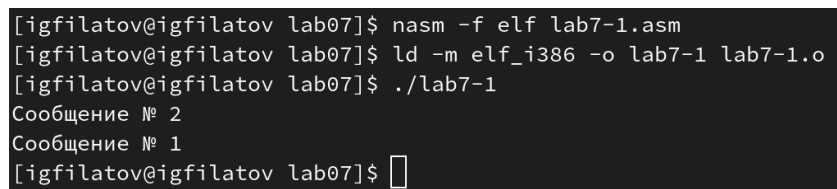
Меняю текст программы в соответствии с листингом 7.2. Программа начинает с метки `_label2`, потом переходит к метке `_label1`, а потом сразу к концу программы, пропуская `_label3` (рис. 4.5).



```
1 %include 'in_out.asm' ; подключение внешнего файла
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 _start:
9 jmp _label2
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintf ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintf ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintf ; 'Сообщение № 3'
21 _end:
22 call quit ; вызов подпрограммы завершения
```

Рис. 4.5: Изменение текста программы

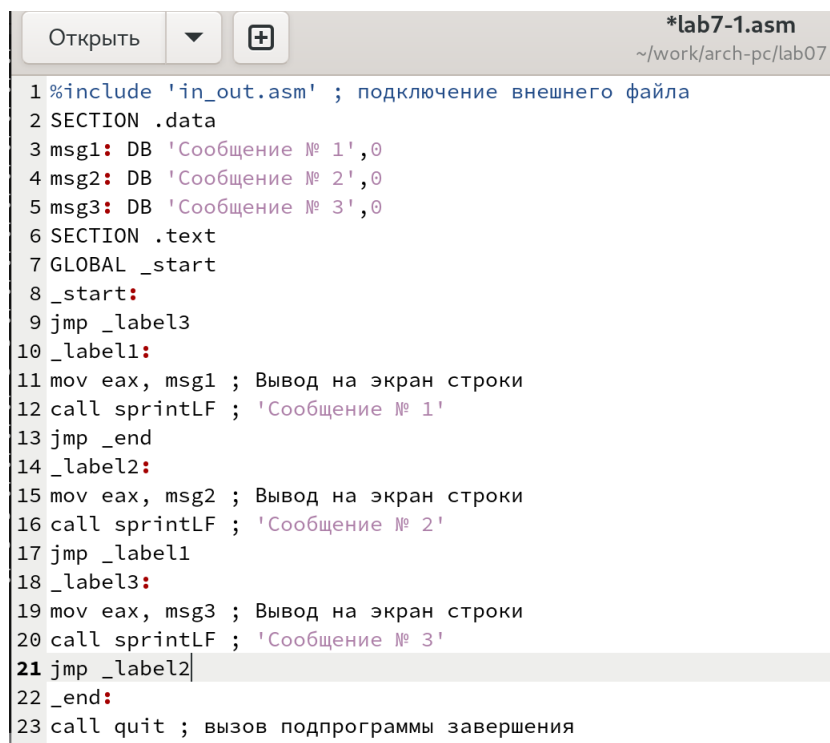
Создаю исполняемый файл и запускаю его (рис. 4.6).



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

Рис. 4.6: Создание и запуск исполняемого файла

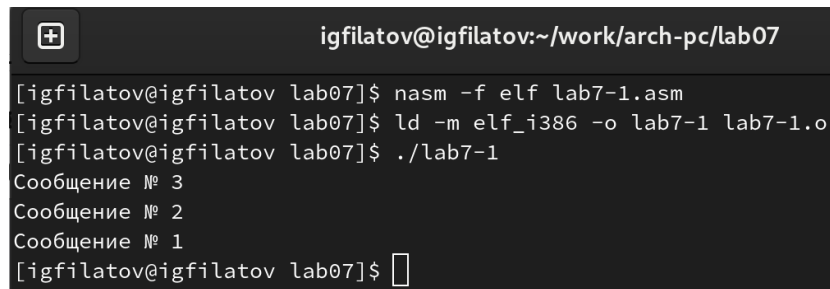
Меняю текст программы в соответствии с заданием. От начала программы перехожу к метке `_label3`, от неё — к `_label2`, от `_label2` — к `_label1`, а от `_label1` — к концу (рис. 4.7).



```
1 %include 'in_out.asm' ; подключение внешнего файла
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 _start:
9 jmp _label3
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintLF ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintLF ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintLF ; 'Сообщение № 3'
21 jmp _label2
22 _end:
23 call quit ; вызов подпрограммы завершения
```

Рис. 4.7: Изменение текста программы

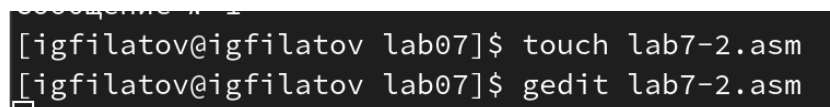
Создаю исполняемый файл и проверяю его работу (рис. 4.8).



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

Рис. 4.8: Создание и запуск исполняемого файла

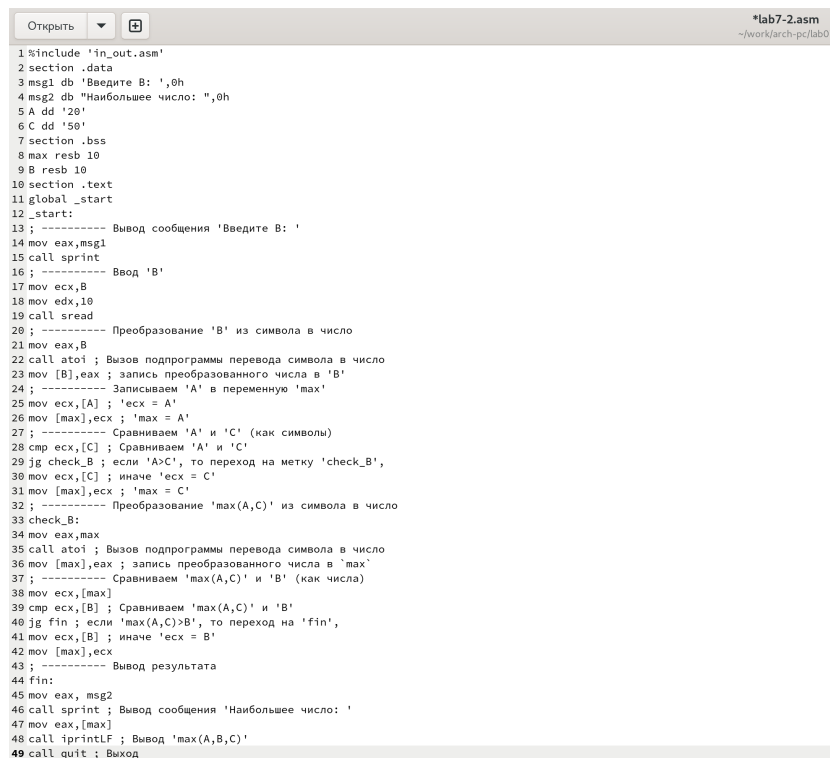
Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 и открываю его с помощью редактора gedit (рис. 4.9).



```
igfilatov@igfilatov:~/work/arch-pc/lab07
[igfilatov@igfilatov lab07]$ touch lab7-2.asm
[igfilatov@igfilatov lab07]$ gedit lab7-2.asm
```

Рис. 4.9: Создание файла

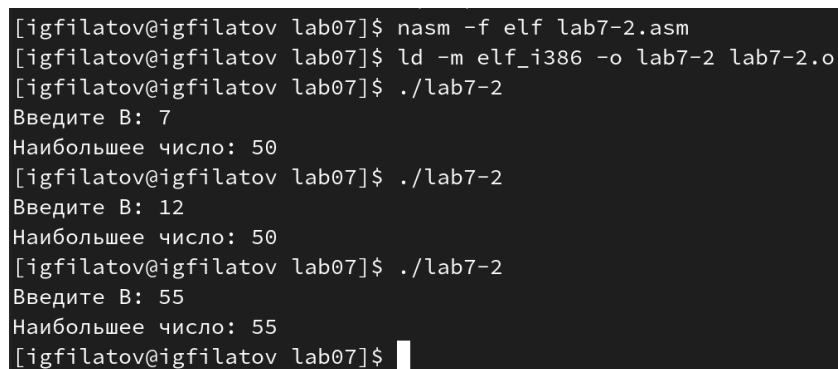
Изучаю текст программы из листинга 7.3 и ввожу его в lab7-2.asm (рис. 4.10).



```
Открыть ▼ + *lab7-2.asm
~/work/arch-pc/lab07
1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите B: ',0h
4 msg2 db "Наибольшее число: ",0h
5 A dd '20'
6 C dd '50'
7 section .bss
8 max resb 10
9 B resb 10
10 section .text
11 global _start
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 ; Вызов подпрограммы перевода символа в число
23 mov [B],eax ; запись преобразованного числа в 'B'
24 ; ----- Записываем 'A' в переменную 'max'
25 mov ecx,[A] ; 'ecx = A'
26 mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 jg check_B ; если 'A>C', то переход на метку 'check_B',
30 mov ecx,[C] ; иначе 'ecx = C'
31 mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 mov eax,max
35 call atoi ; Вызов подпрограммы перевода символа в число
36 mov [max],eax ; запись преобразованного числа в 'max'
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 mov ecx,[max]
39 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 jg fin ; если 'max(A,C)>B', то переход на 'fin',
41 mov ecx,[B] ; иначе 'ecx = B'
42 mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 mov eax, msg2
46 call sprint ; Вывод сообщения 'Наибольшее число: '
47 mov eax,[max]
48 call printf ; Вывод 'max(A,B,C)'
49 call quit ; Выход
```

Рис. 4.10: Ввод текста программы

Создаю и запускаю исполняемый файл. Проверяю его работу, вводя разные значения переменной B (рис. 4.11).

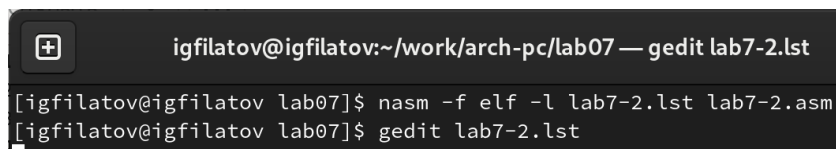


```
[igfilatov@igfilatov lab07]$ nasm -f elf lab7-2.asm
[igfilatov@igfilatov lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[igfilatov@igfilatov lab07]$ ./lab7-2
Введите B: 7
Наибольшее число: 50
[igfilatov@igfilatov lab07]$ ./lab7-2
Введите B: 12
Наибольшее число: 50
[igfilatov@igfilatov lab07]$ ./lab7-2
Введите B: 55
Наибольшее число: 55
[igfilatov@igfilatov lab07]$
```

Рис. 4.11: Создание и запуск исполняемого файла

## 4.2 Изучение структуры файлы листинга

Создаю файл листинга для программы из файла lab7-2.asm и открываю его с помощью редактора gedit (рис. 4.12).



```
igfilatov@igfilatov:~/work/arch-pc/lab07 — gedit lab7-2.lst
[igfilatov@igfilatov lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
[igfilatov@igfilatov lab07]$ gedit lab7-2.lst
```

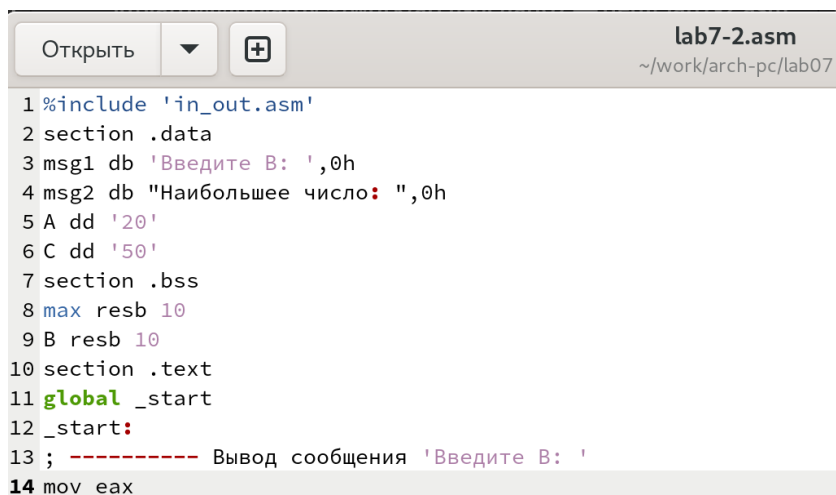
Рис. 4.12: Создание файла листинга

Выбираю три строки из файла листинга. Первый столбец — номера строк (32, 33 и 34), второй — смещение машинного кода от начала текущего сегмента в виде шестнадцатеричной последовательности, третий — машинный код, в который ассемблируется инструкция, последние два — исходный текст программы (рис. 4.13).

32	0000001B 89C1	<1>	mov	ecx, eax
33	0000001D BB01000000	<1>	mov	ebx, 1
34	00000022 B804000000	<1>	mov	eax, 4

Рис. 4.13: Файл листинга

Открываю файл с программой lab7-2.asm и в строке с инструкцией mov убираю второй операнд (рис. 4.14).



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

1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите B: ',0h
4 msg2 db "Наибольшее число: ",0h
5 A dd '20'
6 C dd '50'
7 section .bss
8 max resb 10
9 B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B: '
14 mov eax
```

Рис. 4.14: Изменение текста программы

Создаю файл листинга. Терминал предупреждает об ошибке. Открываю файл в редакторе gedit (рис. 4.15).

```
[igfilatov@igfilatov lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:14: error: invalid combination of opcode and operands
[igfilatov@igfilatov lab07]$
```

Рис. 4.15: Создание файла листинга

Файл листинга не создался, вместо него — текст программы, к которому добавился текст той же ошибки, что вывел терминал (рис. 4.16).

```
13                                     ; ----- Вывод сообщения 'Введите B: '
14                                     mov eax,
14          *****                  error: invalid combination of opcode and
operands
15 000000E8 E822FFFFFF               call sprint
16                                     ; ----- Ввод 'B'
```

Рис. 4.16: Файл с ошибкой

## 4.3 Задание для самостоятельной работы

Создаю файл lab7-3.asm, открываю его и пишу программу нахождения наименьшей из 3 целочисленных переменных (рис. 4.17).

```

1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите A: ',0h
4 msg2 db 'Введите B: ',0h
5 msg3 db 'Введите C: ',0h
6 msg4 db "Наименьшее число: ",0h
7 section .bss
8 min resb 10
9 A resb 10
10 B resb 10
11 C resb 10
12 section .text
13 global _start
14 _start:
15 ; ----- Вывод сообщения 'Введите A: '
16 mov eax,msg1
17 call sprint
18 ; ----- Ввод 'A'
19 mov ecx,A
20 mov edx,10
21 call sread
22 ; ----- Вывод сообщения 'Введите B: '
23 mov eax,msg2
24 call sprint
25 ; ----- Ввод 'B'
26 mov ecx,B
27 mov edx,10
28 call sread
29 ; ----- Преобразование 'B' из символа в число
30 mov eax,B
31 call atoi ; Вызов подпрограммы перевода символа в число
32 mov [B],eax ; запись преобразованного числа в 'B'
33 ; ----- Вывод сообщения 'Введите C: '
34 mov eax,msg3
35 call sprint
36 ; ----- Ввод 'C'
37 mov ecx,C
38 mov edx,10
39 call sread
40 ; ----- Записываем 'A' в переменную 'min'
41 mov ecx,[A] ; 'ecx = A'
42 mov [min],ecx ; 'min = A'
43 ; ----- Сравниваем 'A' и 'C' (как символы)
44 cmp ecx,[C] ; Сравниваем 'C' и 'A'
45 jl check_B ; если 'A<C', то переход на метку 'check_B',
46 mov ecx,[C] ; иначе 'ecx = C'
47 mov [min],ecx ; 'min = C'
48 ; ----- Преобразование 'min(A,C)' из символа в число
49 check_B:
50 mov eax,min
51 call atoi ; Вызов подпрограммы перевода символа в число
52 mov [min],eax ; запись преобразованного числа в 'min'
53 ; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
54 mov ecx,[min]
55 cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
56 jl fin ; если 'min(A,C)<B', то переход на 'fin',
57 mov ecx,[B] ; иначе 'ecx = B'
58 mov [min],ecx
59 ; ----- Вывод результата
60 fin:

```

Рис. 4.17: Программа нахождения наименьшей из 3 целочисленных переменных

Текст программы:

```
%include 'in_out.asm'

section .data
msg1 db 'Введите A:',0h
msg2 db 'Введите B:',0h
msg3 db 'Введите C:',0h
msg4 db "Наименьшее число:",0h

section .bss
min resb 10
A resb 10
B resb 10
C resb 10

section .text
global _start
_start:
; ----- Вывод сообщения 'Введите A:'
mov eax,msg1
call sprint
; ----- Ввод 'A'
mov ecx,A
mov edx,10
call sread
; ----- Вывод сообщения 'Введите B:'
mov eax,msg2
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
```



```

; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Вывод сообщения 'Введите C:'
mov eax,msg3
call sprint
; ----- Ввод 'C'
mov ecx,C
mov edx,10
call sread
; ----- Записываем 'A' в переменную 'min'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'min = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'C' и 'A'
jl check_B ; если 'A<C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [min],ecx ; 'min = C'
; ----- Преобразование 'min(A,C)' из символа в число
check_B:
mov eax,min
call atoi ; Вызов подпрограммы перевода символа в число
mov [min],eax ; запись преобразованного числа в min
; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
mov ecx,[min]
cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
jl fin ; если 'min(A,C)<B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'

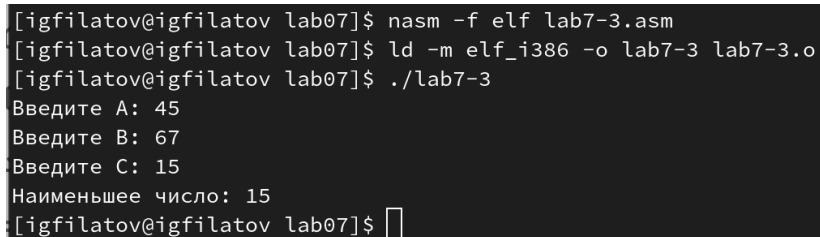
```

```

mov [min],ecx
; ---- Вывод результата
fin:
mov eax, msg4
call sprint ; Вывод сообщения 'Наименьшее число:'
mov eax,[min]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход

```

Создаю и запускаю исполняемый файл, работу которого проверяю значениями из 7-ого варианта (рис. 4.18).



```

[igfilatov@igfilatov lab07]$ nasm -f elf lab7-3.asm
[igfilatov@igfilatov lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[igfilatov@igfilatov lab07]$ ./lab7-3
Введите A: 45
Введите B: 67
Введите C: 15
Наименьшее число: 15
[igfilatov@igfilatov lab07]$ 

```

Рис. 4.18: Создание и запуск исполняемого файла

Создаю файл lab7-4.asm, открываю его и пишу программу которая для введенных с клавиатуры значений вычисляет значение функции из варианта 7 (рис. 4.19).

```

1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите x: ',0h
4 msg2 db 'Введите a: ',0h
5 msg3 db "Значение: ",0h
6 section .bss
7 x resb 10
8 a resb 10
9 answer resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите x: '
14 mov eax,msg1
15 call sprint
16 ; ----- Ввод 'x'
17 mov ecx,x
18 mov edx,10
19 call sread
20 ; ----- Преобразование 'x' из символа в число
21 mov eax,x
22 call atoi ; Вызов подпрограммы перевода символа в число
23 mov [x],eax ; запись преобразованного числа в 'x'
24 ; ----- Вывод сообщения 'Введите a: '
25 mov eax,msg2
26 call sprint
27 ; ----- Ввод 'a'
28 mov ecx,a
29 mov edx,10
30 call sread
31 ; ----- Преобразование 'a' из символа в число
32 mov eax,a
33 call atoi ; Вызов подпрограммы перевода символа в число
34 mov [a],eax ; запись преобразованного числа в 'a'
35 ; ----- Записываем 'x' в регистр 'ecx'
36 mov ecx,[x] ; 'ecx = A'
37 ; ----- Сравниваем 'x' и 'a' (как числа)
38 cmp ecx,[a] ; Сравниваем 'x' и 'a'
39 je next ; если 'x=a', то переход на метку 'next',
40 mov eax,[a] ; иначе 'eax = a'
41 mov ebx,[x] ; 'ebx = x'
42 add eax,ebx ; 'eax=eax+ebx=a+x'
43 jp fin ; 'Переход к концу программы'
44 ; ----- Если 'x=a', то
45 next:
46 mov eax,6 ; иначе 'eax = 6'
47 mov ebx,[a] ; 'ebx = a'
48 mul ebx ; 'eax=eax*ebx=6a'
49 ; ----- Вывод результата
50 fin:
51 mov [answer],eax
52 mov eax, msg3
53 call sprint ; Вывод сообщения 'Наименьшее число: '
54 mov eax,[answer]
55 call iprintLF ; Вывод 'min(A,B,C)'
56 call quit ; Выход

```

Рис. 4.19: Программа вычисления значения функции

Текст программы:

```
%include 'in_out.asm'

section .data
msg1 db 'Введите x:',0h
msg2 db 'Введите a:',0h
msg3 db "Значение:",0h

section .bss
x resb 10
a resb 10
answer resb 10

section .text
global _start
_start:
; ----- Вывод сообщения 'Введите x:'
mov eax,msg1
call sprint
; ----- Ввод 'x'
mov ecx,x
mov edx,10
call sread
; ----- Преобразование 'x' из символа в число
mov eax,x
call atoi ; Вызов подпрограммы перевода символа в число
mov [x],eax ; запись преобразованного числа в 'x'
; ----- Вывод сообщения 'Введите a:'
mov eax,msg2
call sprint
; ----- Ввод 'a'
mov ecx,a
```

```

mov edx,10
call sread
; ----- Преобразование 'а' из символа в число
mov eax,a
call atoi ; Вызов подпрограммы перевода символа в число
mov [a],eax ; запись преобразованного числа в 'а'
; ----- Записываем 'х' в регистр 'есх'
mov esx,[x] ; 'есх = А'
; ----- Сравниваем 'х' и 'а' (как числа)
cmp esx,[a] ; Сравниваем 'х' и 'а'
je next ; если 'х=а', то переход на метку 'next',
mov eax,[a] ; иначе 'eax = а'
mov ebx,[x] ; 'ebx = х'
add eax,ebx ; 'eax=eax+ebx=a+х'
jr fin ; 'Переход к концу программы'
; ----- Если 'х=а', то
next:
mov eax,6 ; иначе 'eax = 6'
mov ebx,[a] ; 'ebx = а'
mul ebx ; 'eax=eax*ebx=6а'
; ----- Вывод результата
fin:
mov [answer],eax
mov eax, msg3
call sprint ; Вывод сообщения 'Наименьшее число:'
mov eax,[answer]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход

```

Создаю и запускаю исполняемый файл, работу которого проверяю предложенными значениями (рис. 4.20).

```
[igfilatov@igfilatov lab07]$ nasm -f elf lab7-4.asm
[igfilatov@igfilatov lab07]$ ld -m elf_i386 -o lab7-4 lab7-4.o
[igfilatov@igfilatov lab07]$ ./lab7-4
Введите x: 1
Введите a: 1
Значение: 6
[igfilatov@igfilatov lab07]$ ./lab7-4
Введите x: 2
Введите a: 1
Значение: 3
[igfilatov@igfilatov lab07]$
```

Рис. 4.20: Создание и запуск исполняемого файла

## 5 Выводы

Я изучил команды условного и безусловного переходов, приобрел навыки написания программ с использованием переходов и ознакомился с назначением и структурой файла листинга.

## **6 Список литературы**

1. Архитектура ЭВМ