

# **Отчет по лабораторной работе №7**

**Дисциплина: архитектура компьютера**

Горбунова Яна Сергеевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
4.1	Реализация переходов в NASM . . . . .	8
4.2	Изучение структуры файла листинга . . . . .	11
4.3	Задания для самостоятельной работы . . . . .	13
<b>5</b>	<b>Выводы</b>	<b>20</b>
	<b>Список литературы</b>	<b>21</b>

## Список иллюстраций

4.1	Создание каталога и файла для программы . . . . .	8
4.2	Сохранение программы . . . . .	8
4.3	Запуск программы . . . . .	9
4.4	Изменение программы . . . . .	9
4.5	Запуск измененной программы . . . . .	10
4.6	Проверка изменений . . . . .	10
4.7	Сохранение новой программы . . . . .	10
4.8	Проверка программы из листинга . . . . .	11
4.9	Проверка файла листинга . . . . .	11
4.10	Удаление операнда из программы . . . . .	12
4.11	Просмотр ошибки в файле листинга . . . . .	13
4.12	Первая программа самостоятельной работы . . . . .	14
4.13	Проверка работы первой программы . . . . .	16
4.14	Вторая программа самостоятельной работы . . . . .	17
4.15	Проверка работы второй программы . . . . .	19

## **Список таблиц**

# 1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

## 2 Задание

1. Реализация переходов в NASM
2. Изучение структуры файлов листинга
3. Самостоятельное написание программ по материалам лабораторной 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.

## 4 Выполнение лабораторной работы

### 4.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы №7 (рис. 4.1).

```
ysgorbunova@dk6n54 ~ $ mkdir ~/work/arch-pc/lab07
ysgorbunova@dk6n54 ~ $ cd ~/work/arch-pc/lab07
ysgorbunova@dk6n54 ~/work/arch-pc/lab07 $ touch lab7-1.asm
```

Рис. 4.1: Создание каталога и файла для программы

Копирую код из листинга в файл будущей программы. (рис. 4.2).

```
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: Сохранение программы

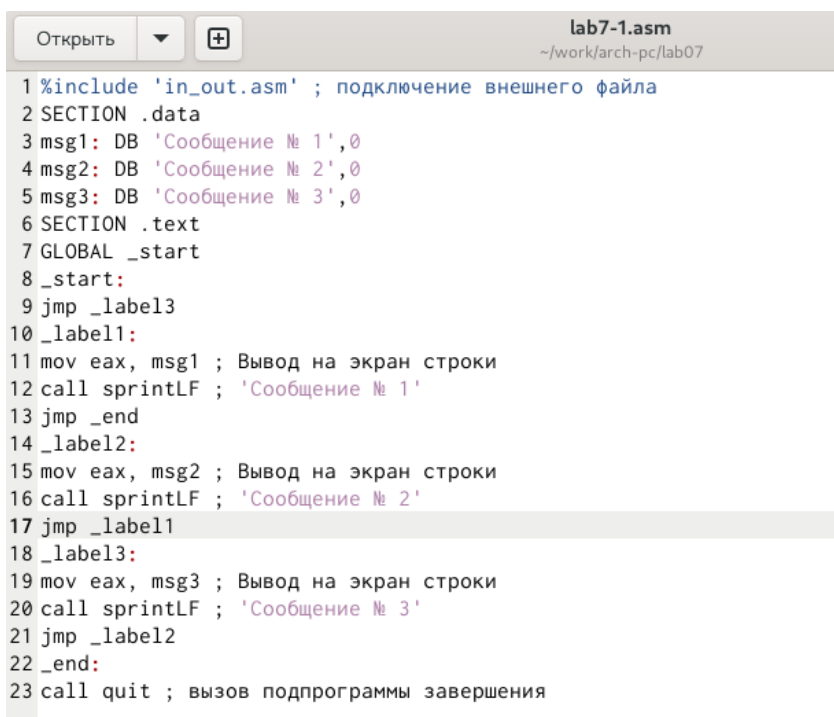


При запуске программы я убедилась в том, что безусловный переход действительно изменяет порядок выполнения инструкций (рис. 4.3).

```
ysgorbunova@edk6n54 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
ysgorbunova@edk6n54 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
ysgorbunova@edk6n54 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
ysgorbunova@edk6n54 ~/work/arch-pc/lab07 $
```

Рис. 4.3: Запуск программы

Изменяю программу таким образом, чтобы поменялся порядок выполнения функций (рис. 4.4).



```
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 _label3
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 jmp _label2
22 _end:
23 call quit ; вызов подпрограммы завершения
```

Рис. 4.4: Изменение программы

Запускаю программу и проверяю, что примененные изменения верны (рис. 4.5).

```
ysgorbunova@dk6n54 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
ysgorbunova@dk6n54 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
ysgorbunova@dk6n54 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
ysgorbunova@dk6n54 ~/work/arch-pc/lab07 $
```

Рис. 4.5: Запуск измененной программы

Теперь изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке (рис. ??).

```
ysgorbunova@dk6n54 ~/work/arch-pc/lab07 $ touch lab7-2.asm
ysgorbunova@dk6n54 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
ysgorbunova@dk6n54 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
ysgorbunova@dk6n54 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 5
Наибольшее число: 50
```

Работа выполнена

корректно, программа в нужном мне порядке выводит сообщения (рис. 4.6).

### Проверка изменений

Рис. 4.6: Проверка изменений

Создаю новый рабочий файл и вставляю в него код из следующего листинга (рис. 4.7).

```
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' ld -m elf_i386 -o lab7-1 lab7-1.o
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
```

Рис. 4.7: Сохранение новой программы

Программа выводит значение переменной с максимальным значением, проверяю работу программы с разными входными данными (рис. 4.8).

```
ysgorbunova@dk6n54 ~/work/arch-pc/lab07 $ touch lab7-2.asm
ysgorbunova@dk6n54 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
ysgorbunova@dk6n54 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
ysgorbunova@dk6n54 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 5
Наибольшее число: 50
```

Рис. 4.8: Проверка программы из листинга

## 4.2 Изучение структуры файла листинга

Создаю файл листинга с помощью флага -l команды nasm и открываю его с помощью текстового редактора mousepad (рис. 4.9).

```
ysgorbunova@dk6n54 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
ysgorbunova@dk6n54 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
ysgorbunova@dk6n54 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
ysgorbunova@dk6n54 ~/work/arch-pc/lab07 $ █
```

Рис. 4.9: Проверка файла листинга

Первое значение в файле листинга - номер строки, и он может вовсе не совпадать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем непосредственно идет сам машинный код, а заключает строку исходный текст программы с комментариями.

Удаляю один операнд из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем (рис. 4.10).

```
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
```

Matlab ▾ Ширина табуляции: 8 ▾ Ln 20, Col 42 INS

Рис. 4.10: Удаление операнда из программы

В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла. Никакие выходные файлы при этом помимо файла листинга не создаются. (рис. 4.11).

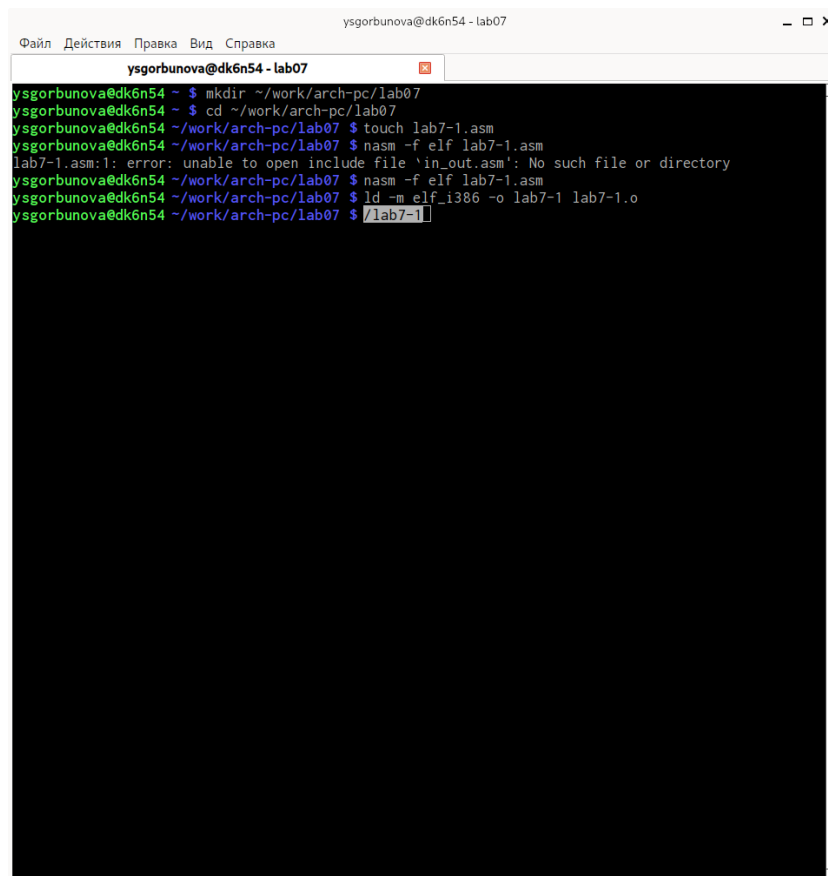
A screenshot of a terminal window titled 'ysgorbunova@dk6n54 - lab07'. The window shows a series of commands and their outputs. The commands are: 'mkdir ~/work/arch-pc/lab07', 'cd ~/work/arch-pc/lab07', 'touch lab7-1.asm', 'nasm -f elf lab7-1.asm', 'nasm -f elf lab7-1.asm', and 'ld -m elf\_i386 -o lab7-1 lab7-1.o'. The output for the first 'nasm' command is 'lab7-1.asm:1: error: unable to open include file `in\_out.asm`: No such file or directory'. The terminal window has a menu bar with 'Файл', 'Действия', 'Правка', 'Вид', and 'Справка'. The status bar at the bottom shows 'ysgorbunova@dk6n54 - lab07'.

Рис. 4.11: Просмотр ошибки в файле листинга

## 4.3 Задания для самостоятельной работы

В лабораторной работе №6 у меня был 17 вариант, буду использовать его. (рис. 4.12).

```

1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1 db 'Введите B: ', 0h
5 msg2 db 'Наименьшее число: ', 0h
6 A dd '26'
7 C dd '68'
8
9 SECTION .bss
10 min resb 10
11 B resb 10
12
13 SECTION .text
14 GLOBAL _start
15 _start:
16
17 mov eax, msg1
18 call sprint
19
20 mov ecx, B
21 mov edx, 10
22 call sread
23
24 mov eax, B
25 call atoi
26 mov [B], eax./lab7-2
27
28 mov ecx, [A]
29 mov [min], ecx
30
31 cmp ecx, [C]
32 je check_B

```

Рис. 4.12: Первая программа самостоятельной работы

Код первой программы:

```

%include 'in_out.asm'

SECTION .data
msg1 db 'Введите B: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '26'
C dd '68'

SECTION .bss
min resb 10
B resb 10

SECTION .text
GLOBAL _start
_start:

```

```
mov eax, msg1
call sprint
```

```
mov ecx, B
mov edx, 10
call sread
```

```
mov eax, B
call atoi
mov [B], eax
```

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

```
cmp ecx, [C]
jg check_B
mov ecx, [C]
mov [min], ecx
```

```
check_B:
mov eax, min
call atoi
mov [min], eax
```

```
mov ecx, [min]
cmp ecx, [B]
jb fin
mov ecx, [B]
```

```
mov [min], ecx
```

```
fin:
```

```
mov eax, msg2
```

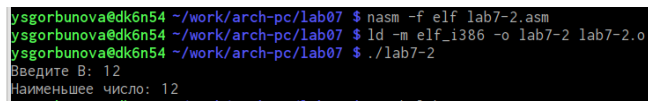
```
call sprint
```

```
mov eax, [min]
```

```
call iprintLF
```

```
call quit
```

Проверяю корректность написания первой программы (рис. 4.13).



```
ysgorbunova@dk6n54 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
ysgorbunova@dk6n54 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
ysgorbunova@dk6n54 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 12
Наименьшее число: 12
```

Рис. 4.13: Проверка работы первой программы

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатуры переменных *a* и *x* (рис. 4.14).



```

1 %include 'in_out.asm'
2 SECTION .data
3 msg_x: DB 'Введите значение переменной x: ', 0
4 msg_a: DB 'Введите значение переменной a: ', 0
5 res: DB 'Результат: ', 0
6 SECTION .bss
7 x: RESB 80
8 a: RESB 80
9 SECTION .text
10 GLOBAL _start
11 _start:
12 mov eax, msg_x
13 call sprint
14 mov ecx, x
15 mov edx, 80
16 call sread
17 mov eax, x
18 call atoi
19 mov edi, eax
20
21 mov eax, msg_a
22 call sprint
23 mov ecx, a
24 mov edx, 80
25 call sread
26 mov eax, a
27 call atoi
28 mov esi, eax
29
30 cmp edi, 8
31 jl add_values
32 mov eax, edi
33 imul eax, esi
34 jmp print_result
35
36 add_values:
37 mov eax, edi
38 add eax, 8
39
40 print_result:
41 mov edi, eax
42 mov eax, res
43 call sprint
44 mov eax, edi
45 call iprintLF

```

Matlab ▾ Ширина табуляции:

Рис. 4.14: Вторая программа самостоятельной работы

Код второй программы:

```

#include 'in_out.asm'

SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
res: DB 'Результат: ', 0

SECTION .bss
x: RESB 80
a: RESB 80

```

```

SECTION .text
GLOBAL _start
_start:
mov eax, msg_x
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov edi, eax

mov eax, msg_a
call sprint
mov ecx, a
mov edx, 80
call sread
mov eax, a
call atoi
mov esi, eax

cmp edi, 8
jl add_values
mov eax, edi
imul eax, esi
jmp print_result

add_values:
mov eax, edi

```

```
add eax, 8
```

```
print_result:
```

```
mov edi, eax
```

```
mov eax, res
```

```
call sprint
```

```
mov eax, edi
```

```
call iprintLF
```

```
call quit
```

Транслирую и компоную файл, запускаю и проверяю работу программы для различных значений а и х (рис. 4.15).

```
ysgorbunova@dk3n58 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
ysgorbunova@dk3n58 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
ysgorbunova@dk3n58 ~/work/arch-pc/lab07 $ ./lab7-3
Введите значение переменной х: 3
Введите значение переменной а: 4
Результат: 11
ysgorbunova@dk3n58 ~/work/arch-pc/lab07 $ ./lab7-3
Введите значение переменной х: 2
Введите значение переменной а: 9
Результат: 10
```

Рис. 4.15: Проверка работы второй программы

## **5 Выводы**

При выполнении лабораторной работы я изучил команды условных и безусловных переходов, а также приобрел навыки написания программ с использованием переходов, познакомился с назначением и структурой файлов листинга.

# Список литературы

1. Курс на ТУИС
2. Лабораторная работа №7
3. Программирование на языке ассемблера NASM Столяров А. В.