

# **Отчёт по лабораторной работе №7**

**Дисциплина: архитектура компьютера**

Сокирка Анна Константиновна

# 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	Порядок выполнения лабораторной работы . . . . .	8
4.2	Изучение структуры файлы листинга . . . . .	10
4.3	Задания для самостоятельной работы . . . . .	11
<b>5</b>	<b>Выводы</b>	<b>14</b>
<b>6</b>	<b>Список литературы</b>	<b>15</b>

## Список иллюстраций

4.1	Создание каталога . . . . .	8
4.2	Ввод программы из листинга . . . . .	8
4.3	Изменение текста программы . . . . .	9
4.4	Изменение текста программы . . . . .	9
4.5	Создание и проверка файла . . . . .	9
4.6	Создание файла листинга . . . . .	10
4.7	Открытие файла с помощью текстового редактора . . . . .	10
4.8	Удаление операнда из программы . . . . .	11
4.9	Просмотр ошибки в файле листинга . . . . .	11
4.10	Возвращение операнда и изменение программы . . . . .	12
4.11	Проверка корректности программы . . . . .	12
4.12	Написание программы . . . . .	13
4.13	Проверка корректности программы . . . . .	13

## **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

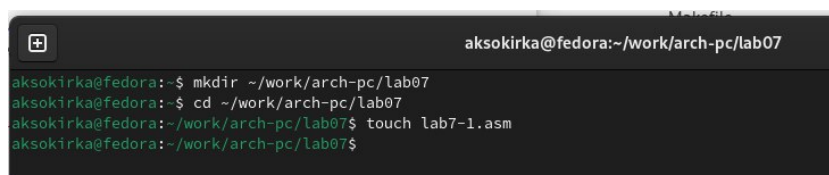
Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- 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 Порядок выполнения лабораторной работы

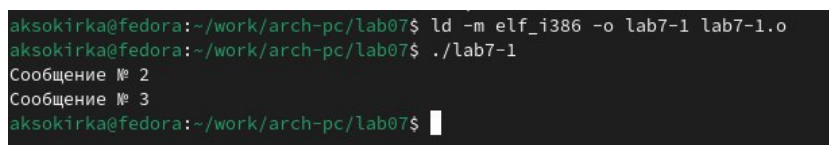
Создам каталог для программ лабораторной работы № 7, перейду в него и создам файл lab7-1.asm (рис. 4.1).



```
aksokirka@fedora:~$ mkdir ~/work/arch-pc/lab07
aksokirka@fedora:~$ cd ~/work/arch-pc/lab07
aksokirka@fedora:~/work/arch-pc/lab07$ touch lab7-1.asm
aksokirka@fedora:~/work/arch-pc/lab07$
```

Рис. 4.1: Создание каталога

Введу в файл lab7-1.asm текст программы из листинга 7.1. Создам исполняемый файл и запущу его (рис. 4.2).



```
aksokirka@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
aksokirka@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
aksokirka@fedora:~/work/arch-pc/lab07$
```

Рис. 4.2: Ввод программы из листинга

Изменяю программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавляю инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавляю инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`).



Изменяю текст программы в соответствии с листингом 7.2. Создам исполняемый файл и проверю его работу (рис. 4.3).

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

Рис. 4.3: Изменение текста программы

Изменяю текст программы добавив или изменив инструкции jmp (рис. 4.4).

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

Рис. 4.4: Изменение текста программы

Создам файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. Внимательно изучу текст программы из листинга 7.3 и введу в lab7-2.asm. Создам исполняемый файл и проверю его работу для разных значений В (рис. 4.5).

```
aksokirka@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
aksokirka@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2
ld: отсутствуют входные файлы
aksokirka@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
aksokirka@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 10
Наибольшее число: 50
aksokirka@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 30
Наибольшее число: 50
aksokirka@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 1
Наибольшее число: 50
aksokirka@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 70
Наибольшее число: 70
aksokirka@fedora:~/work/arch-pc/lab07$
```

Рис. 4.5: Создание и проверка файла

## 4.2 Изучение структуры файлы листинга

Создам файл листинга для программы из файла lab7-2.asm (рис. 4.6).

```
наибольшее число: 70
aksokirka@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
aksokirka@fedora:~/work/arch-pc/lab07$
```

Рис. 4.6: Создание файла листинга

Открою файл листинга lab7-2.lst с помощью любого текстового редактора (рис. 4.7).

Рис. 4.7: Открытие файла с помощью текстового редактора

Директива `%include 'in_out.asm'` в коде на ассемблере позволяет вставить код из определённого файла в другой файл. В качестве операнда она принимает имя файла, код которого будет вставляться. Команда `call quit` в ассемблере вызывает функцию выхода. Функция `atoi` в ассемблере используется для преобразования строки в целое число.

Удаляю один операнд из случайной инструкции, чтобы проверить поведение файла листинга (рис. 4.8).

```
aksokirka@fedora: ~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:34: error: invalid combination of opcode and operands
aksokirka@fedora: ~/work/arch-pc/lab07$
```

Рис. 4.8: Удаление операнда из программы

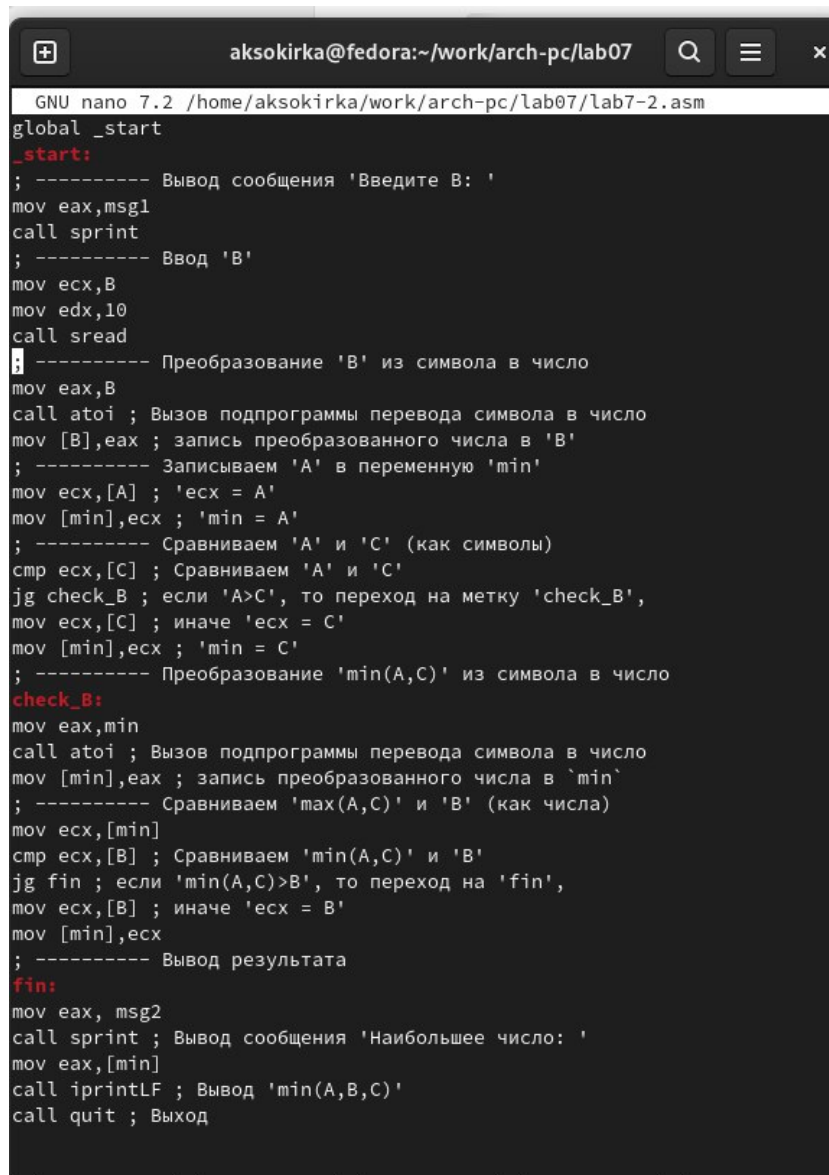
В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла. Никакие выходные файлы при этом помимо файла листинга не создаются (рис. 4.9).

```
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 mov eax,
34 ***** error: invalid combination of opcode and operands
35 00000130 E867FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
```

Рис. 4.9: Просмотр ошибки в файле листинга

## 4.3 Задания для самостоятельной работы

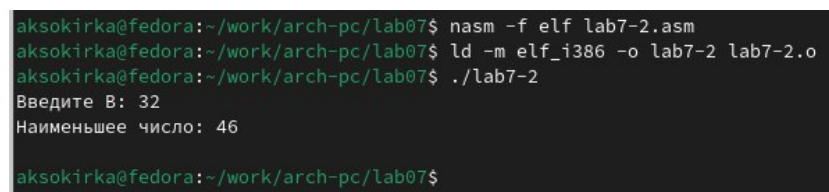
Буду использовать свой вариант - девятнадцатый - из предыдущей лабораторной работы, так как новый вариант в лабораторной №7 я не поняла, как получить. Возвращаю операнд к функции в программе и изменяю ее так, чтобы она выводила переменную с наименьшим значением (рис. 4.10).



```
aksokirka@fedora:~/work/arch-pc/lab07
GNU nano 7.2 /home/aksokirka/work/arch-pc/lab07/lab7-2.asm
global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
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'
; ----- 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 'A' и 'C'
jg 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 'max(A,C)' и 'B' (как числа)
mov ecx,[min]
cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
jg fin ; если 'min(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [min],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[min]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход
```

Рис. 4.10: Возвращение операнда и изменение программы

Проверяю корректность написания программы (рис. 4.11).



```
aksokirka@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
aksokirka@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
aksokirka@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 32
Наименьшее число: 46
aksokirka@fedora:~/work/arch-pc/lab07$
```

Рис. 4.11: Проверка корректности программы

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатуры переменных а и х (рис. 4.12).

```
GNU nano 7.2 /home/aksokirka/work/arch-pc/lab07/lab7-4.asm
#include 'in_out.asm'
SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
rem: DB 'Результат: ', 0
SECTION .bss
x: RESB 80
a: RESB 80
SECTION .text
GLOBAL _start
_start:
; Ввод значения x
mov eax, msg_x
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov edi, eax ; сохраняем x в edi

; Ввод значения a
mov eax, msg_a
call sprint
mov ecx, a
mov edx, 80
call sread
mov eax, a
call atoi
mov esi, eax ; сохраняем a в esi

; Проверка условия и вычисление результата
cmp edi, esi
jle set_x ; если x <= a, переходим к присвоению результата x
add edi, esi ; иначе вычисляем a + x
jmp output_result

set_x:
mov edi, edi ; результат равен x

output_result:
mov eax, rem
call sprint
mov eax, edi
call iprintfLF
call quit
```

Рис. 4.12: Написание программы

Проверяю корректность написания программы (рис. 4.13).

```
aksokirka@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
aksokirka@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
aksokirka@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 4
Введите значение переменной a: 5
Результат: 4
aksokirka@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 3
Введите значение переменной a: 2
Результат: 5
aksokirka@fedora:~/work/arch-pc/lab07$
```

Рис. 4.13: Проверка корректности программы

## **5 Выводы**

При выполнении лабораторной работы я изучила команды условных и безусловных переходов, а также приобрела навыки написания программ с использованием переходов, познакомилась с назначением и структурой файлов листинга.

## 6 Список литературы

[https://esystem.rudn.ru/pluginfile.php/2089087/mod\\_resource/content/0/Лабораторная%20работа%20N°7.%20Команды%20безусловного%20и%20условного%20переходов%20в%20Nasm.%20Программирование%20ветвлений..pdf](https://esystem.rudn.ru/pluginfile.php/2089087/mod_resource/content/0/Лабораторная%20работа%20N°7.%20Команды%20безусловного%20и%20условного%20переходов%20в%20Nasm.%20Программирование%20ветвлений..pdf)