

# **Отчет по Лабораторной работе №7**

**дисциплина: Архитектура компьютера**

Кудинец Максим Антонович

# 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 файлы листинга . . . . .	12
<b>5</b>	<b>Задания для самостоятельной работы</b>	<b>15</b>
<b>6</b>	<b>Выводы</b>	<b>20</b>
	<b>Список литературы</b>	<b>21</b>

# Список иллюстраций

4.1	Переход в каталог и создание файла . . . . .	8
4.2	Программа с использованием инструкции jmp . . . . .	9
4.3	Исполнение программы из листинга 7.1 . . . . .	9
4.4	Исправленный текст программы lab7-1.asm . . . . .	10
4.5	Исполнение программы lab7-1 . . . . .	10
4.6	Программа из листинга 7.3 . . . . .	11
4.7	Исполнение программы из листинга 7.3 . . . . .	12
4.8	Создания файла листинга программы . . . . .	12
4.9	Содержимое файла листинга . . . . .	13
4.10	Удаление операнда . . . . .	14
4.11	Трансляция . . . . .	14
5.1	Текст программы lab7-3.asm . . . . .	16
5.2	Запуск программы . . . . .	16
5.3	Текст программы lab7-4.asm . . . . .	18
5.4	Запуск 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

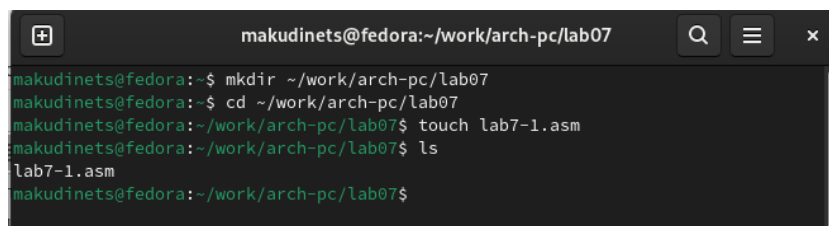
1. Напишите программу нахождения наименьшей из 3 целочисленных переменных  $a, b$  и  $c$ . Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу.
2. Напишите программу, которая для введенных 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.

### **3 Теоретическое введение**

## 4 Выполнение лабораторной работы

### 4.1 Реализация переходов в NASM

1. Создаю каталог для программ лабораторной работы №7, перехожу в него и создаю файл lab7-1.asm.



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

Рис. 4.1: Переход в каталог и создание файла

2. Ввожу в файл lab7-1.asm текст программы из листинга 7.1. Запускаю исполняемый файл.



```

%include 'in_out.asm' ; подключение внешнего файла
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 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения

```

Рис. 4.2: Программа с использованием инструкции jmp

```

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

```

Рис. 4.3: Исполнение программы из листинга 7.1

3. Изменим текст программы так, чтобы она сначала выводила “Сообщение №2”, потом “Сообщение №1” и завершала работу. Запустим исправленную программу.

```

%include 'in_out.asm' ; подключение внешнего файла
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 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'
jmp _end
Архитектура ЭВМ
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения

```

Рис. 4.4: Исправленный текст программы lab7-1.asm

```

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

```

Рис. 4.5: Исполнение программы lab7-1

4. Создадим файл lab7-2.asm. Введем в файл текст программы из листинга 7.3. Программа определяет и выводит на экран наибольшую из целочисленных переменных A, B, C. Значения для A, C задаются в программе, значение вводится с клавиатуры. Запускаю исполняемый файл.

```

%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
A dd '20'
C dd '50'
section .bss
max resb 10
B resb 10
section .text
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 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход

```

Рис. 4.6: Программа из листинга 7.3

```

makudinets@fedora:~/work/arch-pc/lab07$ touch lab7-2.asm
makudinets@fedora:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2.asm
makudinets@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
makudinets@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
makudinets@fedora:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2  lab7-2.asm  lab7-2.o
makudinets@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 20
Наибольшее число: 50
makudinets@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 60
Наибольшее число: 60
makudinets@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 0
Наибольшее число: 50
makudinets@fedora:~/work/arch-pc/lab07$

```

Рис. 4.7: Исполнение программы из листинга 7.3

## 4.2 Изучение структуры файлы листинга

5. Создаю файл листинга для программы из файла lab7-2.asm. Открываю файл листинга в любом текстовом редакторе.

```

makudinets@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
makudinets@fedora:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1.asm  lab7-2  lab7-2.lst
lab7-1      lab7-1.o    lab7-2.asm  lab7-2.o
makudinets@fedora:~/work/arch-pc/lab07$

```

Рис. 4.8: Создания файла листинга программы

```

1      %include 'in_out.asm'
1      <1> ;----- slen -----
2      <1> ; Функция вычисления длины сообщения
3      <1> slen:
4      00000000 53      <1> mov     ebx, 53
5      00000001 89C3    <1> mov     ebx, ebx
6      <1>
7      <1> nextchar:
8      00000003 803800  <1> cmp     byte [ebx], 0
9      00000006 7403    <1> jz      finished
10     00000008 40      <1> inc     eax
11     00000009 F8FA    <1> jmp     nextchar
12     <1>
13     <1> finished:
14     0000000B 72D8    <1> sub     eax, ebx
15     0000000D 5B      <1> mov     ebx, 0
16     0000000F C3      <1> ret
17     <1>
18     <1>
19     <1> ;----- print -----
20     <1> ; Функция печати сообщения
21     <1> ; входные данные: mov, ebx, <message>
22     <1> print:
23     0000000F 52      <1> mov     edi, 0
24     00000010 51      <1> mov     ecx, 0
25     00000011 53      <1> mov     ebx, 0
26     00000012 50      <1> mov     eax, 0
27     00000013 F8FAFFFF <1> call    slen
28     <1>
29     00000018 89C2    <1> mov     edi, ebx
30     0000001A 5B      <1> mov     ebx, 0
31     <1>
32     0000001B 89C1    <1> mov     ecx, ebx
33     0000001D 8B010000 <1> mov     ebx, 1
34     00000022 8B040000 <1> mov     eax, 4
35     00000027 CD80    <1> int     80h
36

```

Рис. 4.9: Содержимое файла листинга

6. Объясняю три строчки из файла листинга: 23 00000106 E891FFFFFF call atoi - Вызов подпрограммы перевода символа в число; 23 - номер строки, 00000106 - адрес, E891FFFFFF - машинный код; 41 0000014B 7F0C jg fin - переход на label 'fin', если 'max(A,C)>B'; 41 - номер строки, 0000014B - адрес, 7F0C - машинный код; 50 0000016D E869FFFFFF call quit - Выход из программы; 50 - номер строки; 0000016D - адрес; E869FFFFFF - машинный код.
7. Открываю файл с программой lab7-2.asm и в одной из инструкций с двумя операндами удаляю один операнд. Транслирую файл с текстом программы с получением файла листинга. Я не получаю выходных файлов, программа выдает ошибку, так как в данной операции должны присутствовать два операнда, а не один.

```
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ехх ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
```

Рис. 4.10: Удаление операнда

```
makudinets@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:28: error: invalid combination of opcode and operands
makudinets@fedora:~/work/arch-pc/lab07$
```

Рис. 4.11: Трансляция

## **5 Задания для самостоятельной работы**

1. Создаю файл lab7-3.asm и ввожу в него текст программы для нахождения наименьшей из трех целочисленных переменных a, b, c. Мой вариант 10. Программа работает корректно.

```

%include 'in_out.asm'
section .data
msg1 db "Наименьшее число: ",0h
A dd 41
B dd 62
C dd 35
section .bss
min resd 1
section .text
global _start
_start:
; ----- Записываем 'A' в переменную 'min'
mov eax, [A]
mov [min], eax

; ----- Сравниваем 'A' и 'C' (как числа)
cmp eax, [C]
jl check_B
mov eax, [C]
mov [min], eax

; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
check_B:
cmp eax, [B]
jl fin
mov eax, [B]
mov [min], eax

; ----- Вывод результата
fin:
mov eax, msg1
call sprint
mov eax, [min]
call iprintLF
call quit

```

Рис. 5.1: Текст программы lab7-3.asm

```

makudinets@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
makudinets@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-3.o -o lab7-3
makudinets@fedora:~/work/arch-pc/lab07$ ./lab7-3
Наименьшее число: 35
makudinets@fedora:~/work/arch-pc/lab07$

```

Рис. 5.2: Запуск программы

2. Создаю файл lab7-4.asm и ввожу в него текст программы, которая для введен-



ных с клавиатуры значений  $x$  и  $a$  вычисляет значение заданной функции и выводит результат вычислений. Мой вариант - 10.

```

%include 'in_out.asm'
section .data
msg1 db 'Введите x: ', 0h
msg2 db 'Введите a: ', 0h
msg3 db 'Результат: ', 0h
section .bss
x resd 1
a resd 1
res resd 1
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите x: '
mov eax, msg1
call sprint

; ----- Ввод 'x'
mov ecx, x
mov edx, 11
call sread
; ----- Преобразование 'x' из символа в число
mov eax, x
call atoi
mov [x], eax
; ----- Вывод сообщения 'Введите a: '
mov eax, msg2
call sprint
; ----- Ввод 'a'
mov ecx, a
mov edx, 11
call sread
; ----- Преобразование 'a' из символа в число
mov eax, a
call atoi
mov [a], eax
; ----- Сравниваем 'x' и '2'
mov eax, [x] ; Загружаем значение 'x' в EAX
cmp eax, 2 ; Сравниваем EAX с 2
jg do_first ; если 'x>2', то переход на метку 'do_first',
jle do_second ; иначе переход на метку 'do_second'

do_first:
mov eax, [x] ; Загружаем значение 'x' в EAX
sub eax, 2 ; Вычитаем 2 из EAX
mov [res], eax ; Переносим результат в ECX
jmp fin

do_second:
mov eax, [a] ; Загружаем значение 'a' в EAX
mov edx, 0
mul dword [a] ; Умножаем EAX на 3
mov [res], eax ; Переносим результат в ECX
jmp fin

; ----- Вывод результата
fin:
mov eax, msg3
call sprint ; Вывод сообщения 'Результат: '
mov eax, [res] ; Переносим результат из ECX в EAX
call iprintLF ; Вывод 'x'
call quit ; Выход

```

Рис. 5.3: Текст программы lab7-4.asm

```
makudinets@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
lab7-4.asm:51: error: invalid combination of opcode and operands
makudinets@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
makudinets@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-4.o -o lab7-4
makudinets@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 3
Введите a: 0
Результат: 1
makudinets@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 1
Введите a: 2
Результат: 4
makudinets@fedora:~/work/arch-pc/lab07$
```

Рис. 5.4: Запуск программы

## **6 Выводы**

В результате выполнения лабораторной работы я изучил команды условного и безусловного переходов, а так же приобрёл навыки написания программ с использованием переходов. Познакомился с назначением и структурой файла листинга.

## **Список литературы**