

# **Отчёт по лабораторной работе №7**

**Дисциплина: Архитектура компьютера**

**Бережной Иван Александрович**

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
3.1	Разбор основ по переходам в NASM . . . . .	7
3.2	Изучение структуры файла листинга . . . . .	12
3.3	Задание для самостоятельной работы . . . . .	14
<b>4</b>	<b>Выводы</b>	<b>22</b>
	<b>Список литературы</b>	<b>23</b>

# Список иллюстраций

3.1	Создание файла lab7-1.asm . . . . .	7
3.2	Копирование кода в файл lab7-1.asm . . . . .	8
3.3	Запуск исполняемого файла lab7-1 . . . . .	8
3.4	Изменение lab7-1.asm . . . . .	9
3.5	Запуск изменённого файла lab7-1.exe . . . . .	9
3.6	Добавление инструкции jmp_label2 . . . . .	10
3.7	Проверка работы lab7-1.asm . . . . .	10
3.8	Создание файла lab7-2.asm . . . . .	11
3.9	Проверка работы lab7-2.asm . . . . .	12
3.10	Просмотр файла листинга . . . . .	13
3.11	Удаление операнда в lab7-2.asm . . . . .	14
3.12	Поиск ошибки в файле листинга . . . . .	14
3.13	Написание программы №1 . . . . .	15
3.14	Проверка работы программы №1 . . . . .	16
3.15	Написание программы №2 . . . . .	18
3.16	Проверка работы программы №2 . . . . .	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. Разбор основ по переходам в NASM
2. Изучение структуры файла листинга
3. Задание для самостоятельной работы

## 3 Выполнение лабораторной работы

### 3.1 Разбор основ по переходам в NASM

Создадим каталог для дальнейшей работы, а также файл в нём lab7-1.asm (рис. 3.1). Скопируем в новый файл код из предложенного листинга (рис. 3.2), затем создадим и запустим исполняемый файл (рис. 3.3).

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

Рис. 3.1: Создание файла lab7-1.asm

```
mc [iaberezhnoy@fedora]:~/work/arch-pc/lab07
lab7-1.asm [-M--] 0 L:[ 1+20 21/ 21] *(650 / 650b) <EOF>
#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. 3.2: Копирование кода в файл lab7-1.asm

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

Рис. 3.3: Запуск исполняемого файла lab7-1

Немного изменим программу, добавив инструкцию `jmp_label1` после вывода сообщения №2 (рис. 3.4). Повторно создадим и запустим исполняемый файл (рис. 3.5).



```
mc [iaberezhnuy@fedora]:~/work/arch-pc/lab07
lab7-1.asm [----] 41 L:[ 1+21 22/ 22] *(670 / 670b) <EOF>
#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'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 3.4: Изменение lab7-1.asm

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

Рис. 3.5: Запуск изменённого файла lab7-1.exe

Снова изменим файл, добавив инструкцию `jmp_label2` после вывода сообщения №3 (рис. 3.6). Теперь при запуске программы видим вывод сообщений в обратном порядке (рис. 3.7).

```
mc [iaberezhnuy@fedora]:~/work/arch-pc/lab07
lab7-1.asm      [-M--] 41 L:[ 1+22 23/ 23] *(682 / 682b) <EOF>
#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 _label3
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 3.6: Добавление инструкции jmp\_label2

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

Рис. 3.7: Проверка работы lab7-1.asm

Создадим файл lab7-2.asm и скопируем в него предложенный листинг (рис. 3.8). Создадим исполняемый файл и проверим его работу для разных значений В (рис. 3.9).

```
mc [iaberezhnoy@fedora]:~/work/arch-pc/lab07
lab7-2.asm [----] 9 L: [ 1+20 21/ 49] *(442 /1743b) 0010 0x00/
%include 'in_out.asm'
section .data
msg1 db 'Введите В: ',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 В: '
mov eax,msg1
call sprint
; ----- Ввод 'В'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'В' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'В'
; ----- Записываем 'А' в переменную 'max'
mov ecx,[A] ; 'ecx = А'
mov [max],ecx ; 'max = А'
; ----- Сравниваем 'А' и 'С' (как символы)
cmp ecx,[C] ; Сравниваем 'А' и 'С'
jg check_B ; если 'А>С', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = С'
mov [max],ecx ; 'max = С'
; ----- 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 'В' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'В'
jg fin ; если 'max(A,C)>В', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = В'
mov [max],ecx
```

Рис. 3.8: Создание файла lab7-2.asm

```
[iaberezhnoy@fedora lab07]$ ./lab7-2
Введите В: 89
Наибольшее число: 89
[iaberezhnoy@fedora lab07]$ ./lab7-2
Введите В: 23
Наибольшее число: 50
[iaberezhnoy@fedora lab07]$ ./lab7-2
Введите В: 50
Наибольшее число: 50
[iaberezhnoy@fedora lab07]$
```

Рис. 3.9: Проверка работы lab7-2.asm

## 3.2 Изучение структуры файла листинга

Создадим файл листинга для программы из файла lab7-2.asm командой `nasm -f elf -l lab7-2.lst lab7-2.asm` и откроем его (рис. 3.10). Разберём несколько строк из этого файла:

1. Строка 11: `jmp nextchar` позволяет безусловно перейти к метке `nextchar`;
2. Строка 29: `mov edx, eax` содержит инструкцию `mov`, а значит значение в регистре `eax` копируется в регистр `edx`;
3. Строка 30: `pop eax` перемещает последнее значение в стеке в регистр `eax`.

```
iaberezhnuy@fedora:~/work/arch-pc/lab07 — mcedit lab7-2.lst
lab7-2.lst [----] 47 L: [ 1+ 0 1/225] *(47 /14458b) 0101 0x065
1      %include 'in_out.asm'
2      <1> ;----- slen -----
3      <1> ; Функция вычисления длины сообщения
4      <1> slen:.....
5      <1>   push    ebx.....
6      <1>   mov     ebx, eax.....
7      <1>   nextchar:.....
8      <1>   cmp     byte [eax], 0...
9      <1>   jz      finished.....
10     <1>   inc     eax.....
11     <1>   jmp     nextchar.....
12     <1>   finished:
13     <1>   sub     eax, ebx
14     <1>   pop     ebx.....
15     <1>   ret.....
16     <1>
17     <1>
18     <1>
19     <1> ;----- sprint -----
20     <1> ; Функция печати сообщения
21     <1> ; входные данные: mov eax,<message>
22     <1> sprint:
23     <1>   push    edx
24     <1>   push    ecx
25     <1>   push    ebx
26     <1>   push    eax
27     <1>   call    slen
28     <1>   .....
29     <1>   mov     edx, eax
30     <1>   pop     eax
31     <1>   .....
32     <1>   mov     ecx, eax
33     <1>   mov     ebx, 1
34     <1>   mov     eax, 4
35     <1>   int     80h
36     <1>   .....
37     <1>   pop     ebx
38     <1>   pop     ecx
39     <1>   pop     edx
40     <1>   ret
41     <1>
```

Рис. 3.10: Просмотр файла листинга

Теперь откроем файл с программой lab7-2.asm и в блоке “Сравниваем ‘max(A,C)’ и ‘B’ (как числа)” удалим второй операнд в строке `str ecx,[B]` (рис. 3.11). Выполним трансляцию с получением файла листинга и посмотрим, что в нём добавилось (рис. 3.12). Видим, что в строке 39 появилась запись ошибки, которая как раз и указывает на то, что мы ввели некорректное число операндов к инструкции.

```

; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx ; Сравниваем 'max(A,C)' и 'B'
error: invalid combination of opcode and operands
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx

```

Рис. 3.11: Удаление операнда в lab7-2.asm

```

37                                     ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013F 8B0D[00000000]          mov ecx,[max]
39                                     cmp ecx ; Сравниваем 'max(A,C)' и 'B'
39                                     error: invalid combination of opcode and operands
40 00000145 7F0C                    jg fin ; если 'max(A,C)>B', то переход на 'fin',
41 00000147 8B0D[0A000000]          mov ecx,[B] ; иначе 'ecx = B'
42 0000014D 890D[00000000]          mov [max],ecx

```

Рис. 3.12: Поиск ошибки в файле листинга

### 3.3 Задание для самостоятельной работы

Создадим файл lab7-3.asm и напишем в нём программу нахождения наименьшей из 3 целочисленных переменных a,b и c в соответствии с вариантом №2, полученным в предыдущей лабораторной работе (рис. 3.13). Создадим исполняемый файл и проверим его работу (рис. 3.14). Программа работает корректно.

```

mc [iaberezhnuy@fedora]:~/work/arch-pc/lab07
lab7-3.asm [----] 11 L: [ 1+13 14/ 40] *(205 /1334b) 0010 0x00A
%include 'in_out.asm'
section .data
msg2 db "Наибольшее число: ",0h
A dd '82'
B dd '59'
C dd '61'
section .bss
max resb 10
section .text
global _start
_start:
mov eax,B
call atoi
mov [B],eax
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = 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 [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. 3.13: Написание программы №1

```

[iaberezhnuy@fedora lab07]$ nasm -f elf lab7-3.asm
[iaberezhnuy@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[iaberezhnuy@fedora lab07]$ ./lab7-3
Наибольшее число: 82
[iaberezhnuy@fedora lab07]$ 

```

Рис. 3.14: Проверка работы программы №1

### Листинг 7.1. Программа нахождения наибольшей из 3-ёх переменных

```

#include 'in_out.asm'

section .data
msg2 db "Наибольшее число: ",0h
A dd '82'
B dd '59'
C dd '61'

section .bss
max resb 10

section .text
global _start
_start:
mov eax,B
call atoi
mov [B],eax
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = 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 [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:

```

Создадим файл lab7-4.asm и напишем в нём программу, которая для введённых с клавиатуры значений x и a вычисляет значение заданной функции варианта 2 (рис. 3.15). Создадим исполняемый файл и проверим его работу (рис. 3.16). Программа работает корректно.

```
mc [iaberezhnoy@fedora]:~/work/arch-pc/lab07
lab7-4.asm  [----]  0  L:[ 15+41 56/ 56] *(861 / 861b) <EOF>

_start:
; Ввод x
mov eax, msg_x
call sprint
mov ecx, x
mov edx, 10
call sread
mov eax, x
call atoi
mov [x], eax

; Ввод a
mov eax, msg_a
call sprint
mov ecx, a
mov edx, 10
call sread
mov eax, a
call atoi
mov [a], eax

; Вычисление f(x)
mov eax, [x]
mov ebx, [a]
cmp eax, ebx
jl less_than_a
sub eax, 1
jmp print_result

less_than_a:
sub ebx, 1
mov eax, ebx

print_result:
mov [result], eax
mov eax, result_msg
call sprintLF
mov eax, [result]
call iprintLF
call quit
```

Рис. 3.15: Написание программы №2

```

[iaberezhnoy@fedora lab07]$ nasm -f elf lab7-4.asm
[iaberezhnoy@fedora lab07]$ ld -m elf_i386 -o lab7-4 lab7-4.o
[iaberezhnoy@fedora lab07]$ ./lab7-4
Введите x: 5
Введите a: 7
Результат:
6
[iaberezhnoy@fedora lab07]$ ./lab7-4
Введите x: 6
Введите a: 4
Результат:
5
[iaberezhnoy@fedora lab07]$ █

```

Рис. 3.16: Проверка работы программы №2

## Листинг 7.2. Программа вычисления значения заданной функции

```

#include 'in_out.asm'

section .data
    msg_x db 'Введите x: ', 0
    msg_a db 'Введите a: ', 0
    result_msg db 'Результат: ', 0

section .bss
    x resb 10
    a resb 10
    result resb 10

section .text
global _start

```

```

_start:
    ; Ввод x
    mov eax, msg_x
    call sprint
    mov ecx, x
    mov edx, 10
    call sread
    mov eax, x
    call atoi
    mov [x], eax

    ; Ввод a
    mov eax, msg_a
    call sprint
    mov ecx, a
    mov edx, 10
    call sread
    mov eax, a
    call atoi
    mov [a], eax

    ; Вычисление f(x)
    mov eax, [x]
    mov ebx, [a]
    cmp eax, ebx
    jl less_than_a
    sub eax, 1

```

```
    jmp print_result
```

```
less_than_a:
```

```
    sub ebx, 1
```

```
    mov eax, ebx
```

```
print_result:
```

```
    mov [result], eax
```

```
    mov eax, result_msg
```

```
    call sprintf
```

```
    mov eax, [result]
```

```
    call iprintf
```

```
    call quit
```

## 4 Выводы

В ходе лабораторной работы мы изучили команды условного и безусловного переходов, приобрели навыки написания программ с использованием переходов, ознакомились с назначением и структурой файла листинга.

# Список литературы

::: Архитектура ЭВМ