

Отчёт по лабораторной работе №7

Цатурьян Лев Вячеславович НММбд-03-23

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Задание для самостоятельной работы	10
4	Задание 1	11
5	Задание 2	13
6	Выводы	16

Список иллюстраций

2.1	Создание каталога lab07, переход в него, создание в нём файла . .	5
2.2	Запуск исполняемого файла	5
2.3	Изменённый текст программы	6
2.4	Создание объектного и исполняемого файлов. Запуск программы	6
2.5	Изменённый текст программы	7
2.6	Создание объектного и исполняемого файлов. Запуск программы	7
2.7	Создание объектного и исполняемого файлов. Запуск программы	8
2.8	Создание файла листинга	8
2.9	Три строки из файла листинга	8
2.10	Строка ошибки в листинге	8
4.1	Программа, сравнивающая 3 заранее известных числа, выводящее наименьшее на экран	11
4.2	Создание объектного и исполняемого файлов. Запуск программы	12
5.1	Програма в mc	14
5.2	Проверка программы, все работает корректно	15

1 Цель работы

Получение навыков написания программ с использованием переходов. Изучение структуры файла листинга

2 Выполнение лабораторной работы

```
lvcaturjyan@dk6n52 ~ $ cd ~/work/arch-pc
lvcaturjyan@dk6n52 ~/work/arch-pc $ mkdir lab07
lvcaturjyan@dk6n52 ~/work/arch-pc $ cd lab07
lvcaturjyan@dk6n52 ~/work/arch-pc/lab07 $ touch lab7-1.asm
lvcaturjyan@dk6n52 ~/work/arch-pc/lab07 $
```

Рис. 2.1: Создание каталога lab07, переход в него, создание в нём файла

Далее я ввёл в созданный файл текст листинга 7.1, создал объектный и исполняемый файлы

```
lvcaturjyan@dk6n52 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 3
```

Рис. 2.2: Запуск исполняемого файла

Сначала выводится сообщение 2, потом 3

Далее я изменил текст программы таким образом, чтобы программа выводила сначала второе, потом первое сообщение. Для этого я добавил инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1), и инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`)

```

#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
_end:

```

Рис. 2.3: Изменённый текст программы

Теперь создадим объектный и исполняемый файлы и запустим программу

```

lvcaturjyan@dk6n52 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
lvcaturjyan@dk6n52 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
lvcaturjyan@dk6n52 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 1
lvcaturjyan@dk6n52 ~/work/arch-pc/lab07 $

```

Рис. 2.4: Создание объектного и исполняемого файлов. Запуск программы

Теперь сначала выводится второе сообщение, затем первое

Теперь нужно задать такой порядок вывода сообщений : 3,2,1 Для этого я перешел в midnight commander и изменил текст следующим образом:

```

SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label3
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
jmp _label2
_end:

```

Рис. 2.5: Изменённый текст программы

В самом начале стоит метка, отсылающая к выводу сообщения 3, после этого стоит метка на сообщение 2, после этого - на 1, и метка перехода к инструкции `call quit`

```

lvcaturjyan@dk6n52 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
lvcaturjyan@dk6n52 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
lvcaturjyan@dk6n52 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 3
Сообщение No 2
Сообщение No 1

```

Рис. 2.6: Создание объектного и исполняемого файлов. Запуск программы

Все работает корректно

Далее я создал файл `lab7-2.asm` и ввел в него текст из листинга 7.3

```

lvcaturjyan@dk6n52 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
lvcaturjyan@dk6n52 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
lvcaturjyan@dk6n52 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 2
Наибольшее число: 50
lvcaturjyan@dk6n52 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 100
Наибольшее число: 100

```

Рис. 2.7: Создание объектного и исполняемого файлов. Запуск программы

Протестировал работу программы разными числами, все работает правильно
После этого я создал файл листинга с помощью ключа -l

```

lvcaturjyan@dk6n52 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
lvcaturjyan@dk6n52 ~/work/arch-pc/lab07 $ mcedit lab7-2.lst

```

Рис. 2.8: Создание файла листинга

После этого я открыл этот файл в текстовом редакторе и изучил его структуру

```

17 000000F2 B9[0A000000]      mov ecx,B
18 000000F7 BA0A000000      mov edx,10
19 000000FC E842FFFFFF      call sread

```

Рис. 2.9: Три строки из файла листинга

Первая строка: 17 - номер строки, 000000F2 адрес, B9[0A000000] машинный код, mov ecx, B исходный текст программы
Вторая строка: 18 - номер строки, 000000F7 адрес, BA0A000000 машинный код, mov edx, 10 исходный текст программы
Третья строка: 19 - номер строки, 000000FC адрес, E842FFFFFF машинный код, call sread исходный текст программы

После этого я удалил один операнд из инструкции в файле lab7-2.asm и транслировал его в файл листинга

```

12      _start:
13      ; ----- Вывод сообщения 'Введите B: '
14      mov eax
14      ***** error: invalid combination of opcode and operands

```

Рис. 2.10: Строка ошибки в листинге

На месте убранного операнда возникла строка ошибки, новых файлов не создалось

3 Задание для самостоятельной работы

4 Задание 1

Мой вариант из прошлой лабораторной работы - 7

```
/afs/.dk.sci.pfu.edu.ru/home/l/v/lvcaturjyan/work/arch-pc/lab07/lab7-3.asm
#include 'in_out.asm'
section .data
msg2 db "Наименьшее число: ",0h
a dd '45'
b dd '67'
i dd '15'
section .bss
min resb 10
section .text
global _start
_start:

mov eax, b
call atoi
mov [b], eax
mov ecx, [a]
mov [min], ecx
cmp ecx, [i]
jl check_b
mov ecx, [i]
mov [min], ecx

check_b:
mov eax, min
call atoi
mov [min], eax

mov ecx, [min]
cmp ecx, [b]
jl fin
mov ecx, [b]
mov [min], ecx

fin:
mov eax, msg2
call sprint
mov eax,[min]
call iprintLF
call quit
```

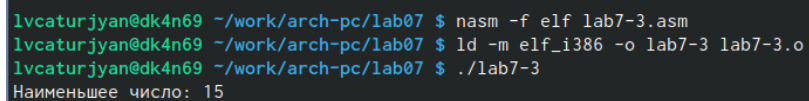
Рис. 4.1: Программа, сравнивающая 3 заранее известных числа, выводящее наименьшее на экран

Её листинг:

```

%include 'in_out.asm' section .data msg2 db "Наименьшее число:",0h a dd '45' b
dd '67' i dd '15' section .bss min resb 10 section .text global _start _start:
    mov eax, b call atoi mov [b], eax mov ecx, [a] mov [min], ecx cmp ecx, [i] jl check_b
mov ecx, [i] mov [min], ecx
check_b: mov eax, min call atoi mov [min], eax
mov ecx, [min] cmp ecx, [b] jl fin mov ecx, [b] mov [min], ecx
fin: mov eax, msg2 call sprint mov eax,[min] call iprintLF call quit

```



```

lvcaturjyan@dk4n69 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
lvcaturjyan@dk4n69 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
lvcaturjyan@dk4n69 ~/work/arch-pc/lab07 $ ./lab7-3
Наименьшее число: 15

```

Рис. 4.2: Создание объектного и исполняемого файлов. Запуск программы

Программа работает корректно даже если переставить значения переменных

5 Задание 2

Моя программа сравнивает значения двух вводимых пользователем чисел, после чего 1) Если числа совпадают, первое число умножается на 6 и выводится на экран 2) Если числа не совпадают, то на экран выводится их сумма

Листинг этой программы: %include 'in_out.asm' SECTION .data msg: DB 'Введите x:',0 aff: DB 'Введите a:',0 SECTION .bss x: RESB 80 a: RESB 80 SECTION .text GLOBAL _start _start:

```
    mov eax, msg call sprint
    mov ecx, x mov edx, 80 call sread
    mov eax, x call atoi mov [x], eax
    mov eax, aff call sprint
    mov ecx, a mov edx, 80 call sread
    mov eax, a call atoi mov [a], eax
    mov ebx, [x] cmp [a], ebx
    je check mov eax, [a] mov ebx, [x] add eax, ebx call iprintLF
    jmp fin
check: mov eax, [a] mov ebx, 6 mul ebx call iprintLF fin: call quit
```

```

/afs/.dk.sci.pfu.edu.ru/home/l/v/lvcaturjyan/work/arch-pc/lab07/lab7-4.asm
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
aff: DB 'Введите a: ',0
SECTION .bss
x: RESB 80
a: RESB 80
SECTION .text
GLOBAL _start
_start:

mov eax, msg
call sprint

mov ecx, x
mov edx, 80
call sread

mov eax, x
call atoi
mov [x], eax

mov eax, aff
call sprint

mov ecx, a
mov edx, 80
call sread

mov eax, a
call atoi
mov [a], eax

mov ebx, [x]
cmp [a], ebx

je check
mov eax, [a]
mov ebx, [x]
add eax, ebx
call iprintLF
jmp fin

check:
mov eax, [a]
mov ebx, 6
mul ebx
call iprintLF
fin:
call quit

```

Рис. 5.1: Програма в мс

```
lvcaturjyan@dk4n69 ~/work/arch-pc/lab07 $ ./lab7-4
Введите x: 1
Введите a: 2
3
lvcaturjyan@dk4n69 ~/work/arch-pc/lab07 $ ./lab7-4
Введите x: 3
Введите a: 3
18
lvcaturjyan@dk4n69 ~/work/arch-pc/lab07 $
```

Рис. 5.2: Проверка программы, все работает корректно

6 Выводы

Я получил навыки написания программ с использованием переходов. Ознакомился со структурой файла листинга