

Отчет по лабораторной работе №7

дисциплина: Архитектура компьютера

Сергеев Даниил Олегович

Содержание

1	Цель лабораторной работы	4
2	Ход выполнения лабораторной работы	5
3	Ход выполнения заданий для самостоятельной работы	14
4	Вывод	21

Список иллюстраций

2.1	Создание lab7-1.asm	5
2.2	Код файла lab7-1.asm	6
2.3	Результат работы lab7-1	6
2.4	Код файла из листинга 7.2	7
2.5	Результат работы измененного файла lab7-1	7
2.6	Измененный код файла lab7-1.asm	8
2.7	Результат работы программы	8
2.8	Создание lab7-2.asm	9
2.9	Код файла из листинга 7.3	10
2.10	Результат работы файла lab7-2	11
2.11	Создание файла листинга для lab7-2.asm	11
2.12	Вид листинга из текстового редактора	12
2.13	Код из листинга 6.1	12
2.14	Результат работы lab7-2 после удаления операнда	13
2.15	Вывод листинга для файла lab7-2	13
3.1	Создаем файл zadanie1.asm	14
3.2	Результат работы программы zadanie1	17
3.3	Создаем файл zadanie2.asm	17
3.4	Результат работы программы zadanie2	20

1 Цель лабораторной работы

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

2 Ход выполнения лабораторной работы

1. Создаем каталог для лабораторной работы №7, переходим в него и создаем lab7-1.asm. В файл вводим текст программы из листинга 7.1.

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

Рис. 2.1: Создание lab7-1.asm

```
lab7-1.asm [----] 17 L: [ 1+24 25/ 25] *(363 / 363b) <EOF> [*] [X]
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
<----->call printf
....
_label2:
<----->mov eax, msg2
<----->call printf
....
_label3:
<----->mov eax, msg3
<----->call printf
....
_end:
<----->call quit
```

Рис. 2.2: Код файла lab7-1.asm

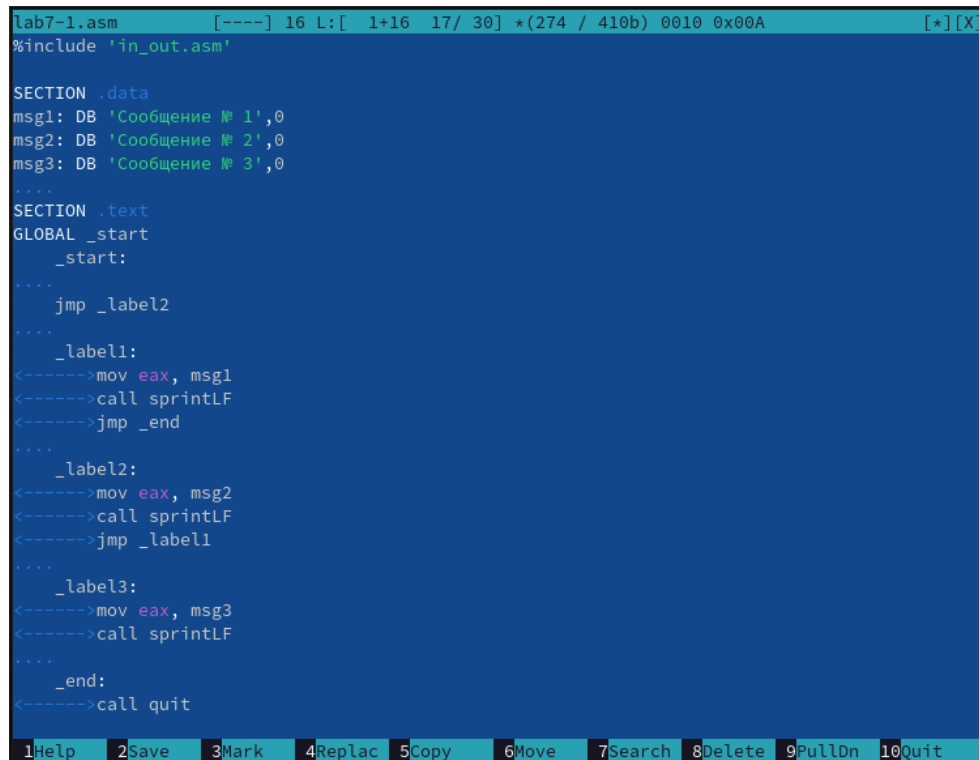
2. Создаем исполняемый файл и запускаем его.

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

Рис. 2.3: Результат работы lab7-1

3. Изменим программу так, чтобы она выводила сначала “Сообщение № 2”, потом “Сообщение № 1” и завершал работу, в соответствии с

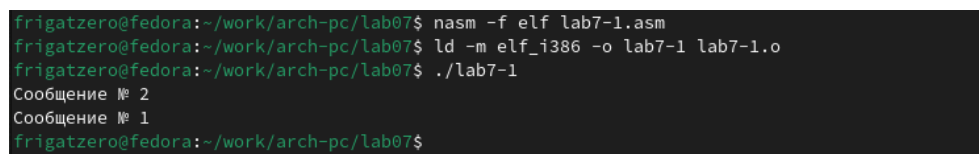
листингом 7.2. Создадим исполняемый файл и проверим его работу.



```
lab7-1.asm [----] 16 L: [ 1+16 17/ 30] *(274 / 410b) 0010 0x00A [*] [X]
#include 'in_out.asm'

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
<----->call sprintLF
<----->jmp _end
....
_label2:
<----->mov eax, msg2
<----->call sprintLF
<----->jmp _label1
....
_label3:
<----->mov eax, msg3
<----->call sprintLF
....
_end:
<----->call quit
```

Рис. 2.4: Код файла из листинга 7.2



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

Рис. 2.5: Результат работы измененного файла lab7-1

4. Изменим программу, чтобы она выводила сообщения в порядке

убывания с 3 по 1.

```
lab7-1.asm [----] 19 L: [ 2+20 22/ 31] *(335 / 423b) 0010 0x00A [*][X]

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
<----->call sprintf
<----->jmp _end
....
_label2:
<----->mov eax, msg2
<----->call sprintf
<----->jmp _label1
....
_label3:
<----->mov eax, msg3
<----->call sprintf
<----->jmp _label2
....
_end:
<----->call quit

1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn 10Quit
```

Рис. 2.6: Измененный код файла lab7-1.asm

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

Рис. 2.7: Результат работы программы

5. Создадим файл lab7-2.asm в том же каталоге, введём код из листинга 7.3. Создадим исполняемый файл и проверим его работу для

разных значений В.

```
frigatzero@fedora:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o
frigatzero@fedora:~/work/arch-pc/lab07$ touch lab7-2.asm
frigatzero@fedora:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2.asm
frigatzero@fedora:~/work/arch-pc/lab07$
```

Рис. 2.8: Создание lab7-2.asm

```
lab7-2.asm [----] 21 L:[ 1+ 0 1/ 52] *(21 / 649b) 0010 0x00A [*] [X]
#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:
    <----->
    <----->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 [max], ecx
    <----->
    <----->cmp ecx, [C]
    <----->jg check_B
    <----->mov ecx, [C]
    <----->mov [max], ecx
    <----->
    check_B:
    <----->mov eax, max
    <----->call atoi
    <----->mov [max], eax
    <----->
    <----->mov ecx, [max]
    <----->cmp ecx, [B]
    <----->jg fin
    <----->mov ecx, [B]
    1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn 10Quit
```

Рис. 2.9: Код файла из листинга 7.3

```

frigatzero@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
frigatzero@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
frigatzero@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 60
Наибольшее число: 60
frigatzero@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 40
Наибольшее число: 50
frigatzero@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 2
Наибольшее число: 50
frigatzero@fedora:~/work/arch-pc/lab07$ █

```

Рис. 2.10: Результат работы файла lab7-2

6. Создадим файл листинга для программы из файла lab7-2.asm и откроем его с помощью mcedit.

```

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

```

Рис. 2.11: Создание файла листинга для lab7-2.asm

```

lab7-2.lst      [----] 57 L:[157+32 189/228] *(11546/13484b) 0010 0x00A      [*][X]
156                                     <1> .restore:
157 000000D6 5E                       <1>     pop     esi...
158 000000D7 5A                       <1>     pop     edx...
159 000000D8 59                       <1>     pop     ecx...
160 000000D9 5B                       <1>     pop     ebx...
161 000000DA C3                       <1>     ret
162                                     <1>
163                                     <1>
164                                     <1> ;----- quit -----
165                                     <1> ; Функция завершения программы
166                                     <1> quit:
167 000000DB BB00000000                <1>     mov     ebx, 0.....
168 000000E0 B801000000                <1>     mov     eax, 1.....
169 000000E5 CD80                      <1>     int     80h
170 000000E7 C3                       <1>     ret
2.....
3                                     SECTION .data
4 00000000 D092D0B2D0B5D0B4D0-        msg1 db 'Введите B: ',0h
5 00000009 B8D182D0B520423A20-
6 00000012 00.....
7 00000013 D09DD0B0D0B8D0B1D0-        msg2 db 'Наибольшее число: ',0h
8 0000001C BED0BBD18CD188D0B5-
9 00000025 D0B520D187D0B8D181-
10 0000002E D0BBD0BE3A2000.....
11 00000035 32300000                  A dd '20'
12 00000039 35300000                  C dd '50'
13.....
14                                     SECTION .bss
15 00000000 <res Ah>                  max resb 10
16 0000000A <res Ah>                  B resb 10
17.....
18                                     SECTION .text
19 GLOBAL _start
20 _start:
21.....
22 000000E8 B8[00000000]              <----->mov eax, msg1
23 000000ED E81DFFFFFF                <----->call sprint
24.....
25 000000F2 B9[0A000000]              <----->mov ecx, B
26 000000F7 BA0A000000                <----->mov edx, 10
27 000000FC E842FFFFFF                <----->call sread
28.....
29 00000101 B8[0A000000]              <----->mov eax, B
30 00000106 E891FFFFFF                <----->call atoi

```

Рис. 2.12: Вид листинга из текстового редактора

7. Для объяснения выберем три строки из файла листинга.

```

20 000000F2 B9[0A000000]              <----->mov ecx, B
21 000000F7 BA0A000000                <----->mov edx, 10
22 000000FC E842FFFFFF                <----->call sread

```

Рис. 2.13: Код из листинга 6.1

- 1.Номер строки (20, 21, 22) - это номер строки файла листинга.
 - 2.Адрес (000000F2/F7/FC) - это смещение машинного кода от начала текущего сегмента.
 - 3.Машинный код (B9[0A000000], BA0A000000, E842FFFFFF) - это исходная строка представленная в виде 16-тиричной последовательности.
 - 4.Исходный текст программы (mov ecx, B; mov edx, 10; call sread) - это строка исходной программы.
8. В инструкции mov edx, 10 удалим правый операнд и выполним трансляцию с получением файла листинга. После трансляции, выходит ошибка в файле lab7-2.asm. В выходных файлах мы получили только листинг, объектный файл не создался. В самом же листинге продублировалась строка 21, в которую записалась ошибка.

```
frigatzero@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:21: error: invalid combination of opcode and operands
frigatzero@fedora:~/work/arch-pc/lab07$
```

Рис. 2.14: Результат работы lab7-2 после удаления операнда

```
20 000000F2 B9[0A000000] <----->mov ecx, B
21 <----->mov edx, .
21 ***** error: invalid combination of opcode and operands
22 000000F7 E847FFFFFF <----->call sread
```

Рис. 2.15: Вывод листинга для файла lab7-2

3 Ход выполнения заданий для самостоятельной работы

1. Напишем программу для нахождения наименьшей из 3 целочисленных переменных a, b и c. Выберем значения переменных из №18, табл. 7.5. Создадим исполняемый файл и проверим его работу

```
frigatzero@fedora:~/work/arch-pc/lab07$ touch zadanie1.asm
frigatzero@fedora:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2  lab7-2.asm  lab7-2.lst  zadanie1.asm
frigatzero@fedora:~/work/arch-pc/lab07$
```

Рис. 3.1: Создаем файл zadanie1.asm

Листинг 3.1 Программа для нахождения наименьшей переменной из a, b и c.

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
fnum db 'Первое число (a): ',0
```

```
snum db 'Второе число (b): ',0
tnum db 'Третье число (c): ',0
minn db 'Наименьшее число: ',0
```

```
a dd 83
b dd 73
c dd 30
```

```
SECTION .bss
```

```
min resb 10
```

```
SECTION .text
```

```
GLOBAL _start
```

```
    _start:
```

```
    ; Выводим значение a
```

```
    mov eax, fnum
```

```
    call sprint
```

```
    mov eax, [a]
```

```
    call iprintLF
```

```
    ; Выводим значение b
```

```
    mov eax, snum
```

```
    call sprint
```

```
    mov eax, [b]
```

```
    call iprintLF
```

```
    ; Выводим значение c
```

```
    mov eax, tnum
```

```

call sprint
mov eax, [c]
call iprintLF
; -----
; Сравниваем a и b
; -----
mov ecx, [a]    ; ecx = a
mov [min], ecx ; min = ecx(a)
cmp ecx, [b]    ; a < b
jl check
mov ecx, [b]    ; ecx = b
mov [min], ecx ; min = ecx(b)
check:
; -----
; Сравниваем ecx и c
; -----
cmp ecx, [c]    ; ecx < c
jl fin
mov ecx, [c]    ; ecx = c
mov [min], ecx ; min = ecx(c)
fin:
mov eax, minn
call sprint
mov eax, [min]
call iprintLF
call quit

```



```

frigatzero@fedora:~/work/arch-pc/lab07$ nasm -f elf zadanie1.asm
frigatzero@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o zadanie1 zadanie1.o
frigatzero@fedora:~/work/arch-pc/lab07$ ./zadanie1
Первое число (a): 83
Второе число (b): 73
Третье число (c): 30
Наименьшее число: 30
frigatzero@fedora:~/work/arch-pc/lab07$

```

Рис. 3.2: Результат работы программы zadanie1

2. Напишем программу, которая для введенных с клавиатуры значений x и a вычислит значение заданной функции $f(x)$ и выведет результат вычислений. Вид функции возьмем под №18 из табл. 7.6. Проверим её работу с значениями $x=1, a=2$ и $x=2, a=1$.

```

frigatzero@fedora:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1.asm lab7-2 lab7-2.lst zadanie1.asm
lab7-1 lab7-1.o lab7-2.asm zadanie1 zadanie1.o
frigatzero@fedora:~/work/arch-pc/lab07$ touch zadanie2.asm
frigatzero@fedora:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1.asm lab7-2 lab7-2.lst zadanie1.asm zadanie2.asm
lab7-1 lab7-1.o lab7-2.asm zadanie1 zadanie1.o
frigatzero@fedora:~/work/arch-pc/lab07$

```

Рис. 3.3: Создаем файл zadanie2.asm

Листинг 3.2 Программа для вычисления выражения $f(x)$.

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
func db 'Задана функция №18',0
```

```
fu_1 db 'f(x) = a^2,      a != 1',0
```

```
fu_2 db 'f(x) = 10 + x,  a == 1',0
```

```
inp_x db 'Введите значение x: ',0
inp_a db 'Введите значение a: ',0
result db 'Результат: ',0
```

```
SECTION .bss
```

```
x resb 4
a resb 4
f resb 4
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
;-[Вывод сообщений func, fu_1, fu_2]
mov eax, func
call sprintf
mov eax, fu_1
call sprintf
mov eax, fu_2
call sprintf
;-[Ввод значения x]
mov eax, inp_x
call sprintf

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

```

;-[Преобразование символа x в число]
mov eax, x
call atoi
mov [x], eax
;-[Ввод значения a]
mov eax, inp_a
call sprint

mov ecx, a
mov edx, 10
call sread
;-[Преобразование символа a в число]
mov eax, a
call atoi
mov [a], eax
; =====
; Вычисление f(x)
; =====
mov ecx, [a]      ; ecx = a
cmp ecx, 1        ; Сравниваем ecx и 1
je if_equal       ; если ecx = 1, то переходим к if_equal
mov eax, [a]      ; иначе записываем f = a^2
mov ebx, [a]
mul ebx           ; eax = a^2


mov [f], eax      ; f = eax
jmp fin          ; переходим к fin

```

```

if_equal:
mov ecx, [x]      ; ecx = x
add ecx, 10       ; ecx = x + 10
mov [f], ecx      ; f = ecx
;-[Вывод результата]
fin:
mov eax, result
call sprint
mov eax, [f]
call iprintLF
call quit

```



```

frigatzero@fedora:~/work/arch-pc/lab07$ nasm -f elf zadanie2.asm
frigatzero@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o zadanie2 zadanie2.o
frigatzero@fedora:~/work/arch-pc/lab07$ ./zadanie2
Задана функция №18
f(x) = a^2,    a != 1
f(x) = 10 + x, a == 1
Введите значение x: 1
Введите значение a: 2
Результат: 4
frigatzero@fedora:~/work/arch-pc/lab07$ ./zadanie2
Задана функция №18
f(x) = a^2,    a != 1
f(x) = 10 + x, a == 1
Введите значение x: 2
Введите значение a: 1
Результат: 12
frigatzero@fedora:~/work/arch-pc/lab07$

```

Рис. 3.4: Результат работы программы zadanie2

4 Вывод

После выполнения заданий лабораторной работы и заданий для самостоятельной работы я приобрел навыки написания программ с использованием условных и безусловных переходов, ознакомился с назначением и структурой файла листинга.