

Отчет по лабораторной работе №7

Дисциплина: архитектура компьютера

Рахматова Жылдыз Талантбековна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Самостоятельная работа.	11
4	Вывод	16

Список иллюстраций

2.1	Создание файла lab8-1.asm	6
2.2	Текст в файле lab8-1.asm	6
2.3	Запуск программы lab8-1	7
2.4	Изменение текста	7
2.5	Проверка работы программы	7
2.6	Изменение текста	8
2.7	Запуск программы	8
2.8	Текст программы для сравнения чисел	9
2.9	Программа для сравнения чисел	9
2.10	Файл листинга lab8-2.lst	9
2.11	Объяснения первой строки	10
2.12	Объяснения второй строки	10
2.13	Объяснения третьей строки	10
2.14	Файл листинга без одного операнда	10
3.1	Текст программы	12
3.2	Результат работы программы	13
3.3	Текст программы	14
3.4	Проверка работы программы	15

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.

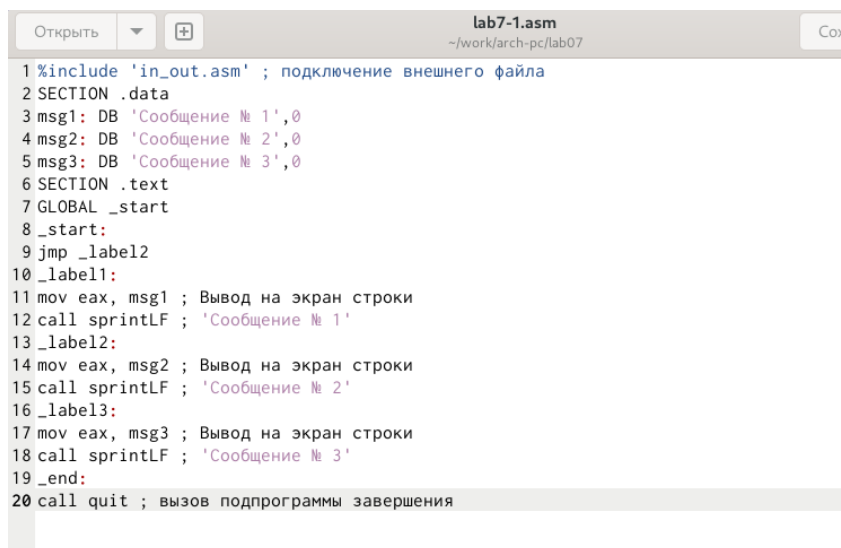
2 Выполнение лабораторной работы

- 1) Я создала каталог lab7 и внутри создал файл lab7-1.asm

```
ztrakhmatova@dk3n55 ~ $ mkdir ~/work/arch-pc/lab07
ztrakhmatova@dk3n55 ~ $ cd ~/work/arch-pc/lab07
ztrakhmatova@dk3n55 ~/work/arch-pc/lab07 $ touch lab7-1.asm
```

Рис. 2.1: Создание файла lab8-1.asm

- 2) Я ввела в файл текст программы и запустила его.



```
lab7-1.asm
~/work/arch-pc/lab07

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label2
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintf ; 'Сообщение № 1'
13 _label2:
14 mov eax, msg2 ; Вывод на экран строки
15 call sprintf ; 'Сообщение № 2'
16 _label3:
17 mov eax, msg3 ; Вывод на экран строки
18 call sprintf ; 'Сообщение № 3'
19 _end:
20 call quit ; вызов подпрограммы завершения
```

Рис. 2.2: Текст в файле lab8-1.asm

- 3) Я создала исполняемый файл и запустила его. Результат соответствовал нужному.

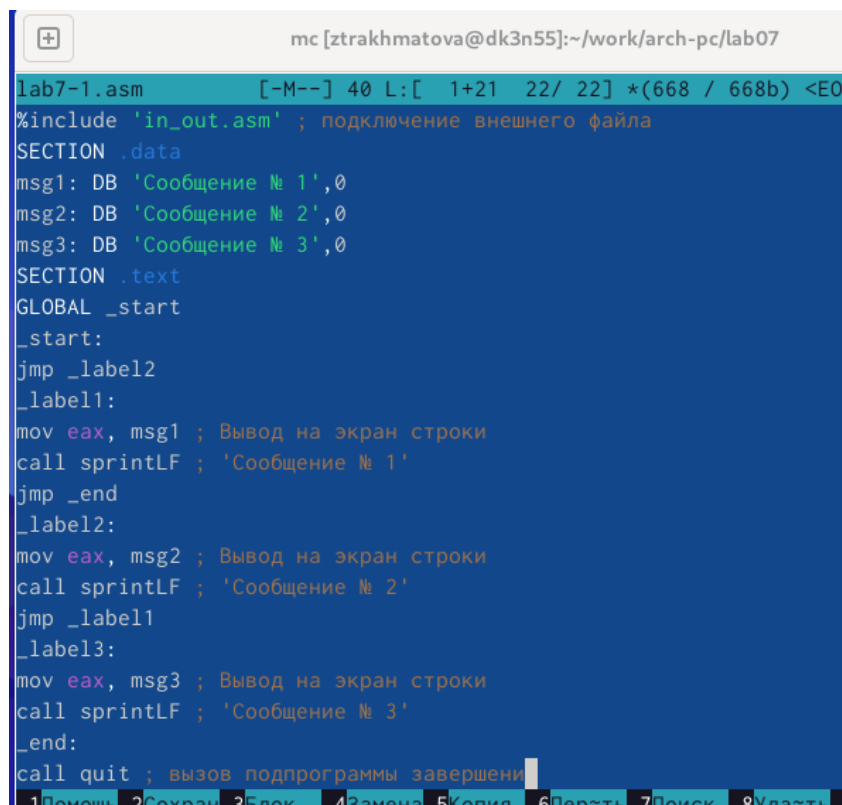
```

ztrakhmatova@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
ztrakhmatova@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
ztrakhmatova@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
ztrakhmatova@dk3n55 ~/work/arch-pc/lab07 $ █

```

Рис. 2.3: Запуск программы lab8-1

4)Я изменила текст программы чтобы выводился нужный ответ и создала исполняемый файл.



```

lab7-1.asm  [-M--] 40 L:[ 1+21 22/ 22] *(668 / 668b) <EO
#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
1 Помощь 2 Сохран 3 Блок 4 Замена 5 Копия 6 Вставить 7 Поиск 8 Отмена

```

Рис. 2.4: Изменение текста

```

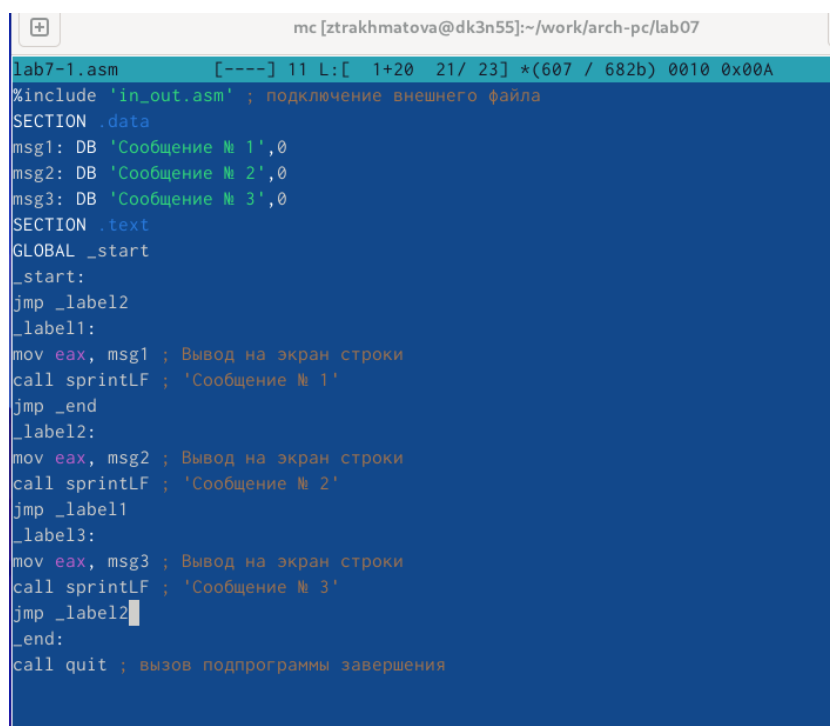
ztrakhmatova@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
ztrakhmatova@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
ztrakhmatova@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 1
ztrakhmatova@dk3n55 ~/work/arch-pc/lab07 $ █

```

Рис. 2.5: Проверка работы программы

5)Я изменила текст программы чтобы сначала выводило сообщение 3,затем 2,

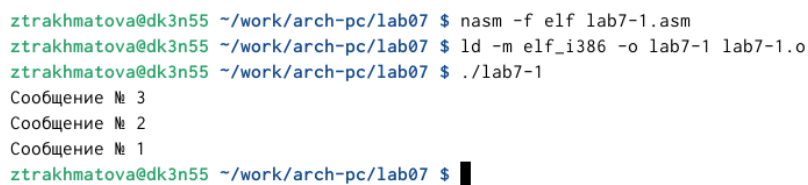
затем 1.



```
lab7-1.asm      [----] 11 L:[ 1+20 21/ 23] *(607 / 682b) 0010 0x00A
#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'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 2.6: Изменение текста

6) Запустила программу и проверила ее работу.



```
ztrakhmatova@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
ztrakhmatova@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
ztrakhmatova@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
ztrakhmatova@dk3n55 ~/work/arch-pc/lab07 $
```

Рис. 2.7: Запуск программы

7) Я создала файл lab7-2.asm и написала текст программы.


```
mc [ztrakhmatova@dk3n55]:~/work/arch-pc/lab07
lab7-1.asm [-M--] 17 L:[ 1+ 0 1/ 49] *(17 /1743b) 009
#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
```

Рис. 2.8: Текст программы для сравнения чисел

8) Я ввела два разных числа чтобы проверить как работает программа.

```
ztrakhmatova@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
ztrakhmatova@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
ztrakhmatova@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Введите B: 65
Наибольшее число: 65
ztrakhmatova@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Введите B: 43
Наибольшее число: 50
ztrakhmatova@dk3n55 ~/work/arch-pc/lab07 $ █
```

Рис. 2.9: Программа для сравнения чисел

9) Я создала файл листинга lab7-2.lst и открыла его.

```
ztrakhmatova@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
ztrakhmatova@dk3n55 ~/work/arch-pc/lab07 $ mcedit lab7-2.lst
ztrakhmatova@dk3n55 ~/work/arch-pc/lab07 $ █
```

Рис. 2.10: Файл листинга lab8-2.lst

- 10) Проанализировав файл, я поняла как он работает и какие значения выводит.
- 11) Эта строка находится на 24 месте, ее адрес "00000101", Машинный код - B8 [0A000000], а `mov eax,B` - исходный текст программы, означающий что в регистр `eax` мы вносим значения переменной `B`.

```

24 00000101 B8[0A000000]          mov eax,B

```

Рис. 2.11: Объяснения первой строки

- 2) Эта строка находится на 38 месте, ее адрес "00000134", Машинный код - E863FFFFFF, а `call atoi` - исходный текст программы, означающий что символ лежащий в строке выше переводится в число.

```

38 00000134 E863FFFFFF          call atoi

```

Рис. 2.12: Объяснения второй строки

- 3) Эта строка находится на 50 месте, ее адрес "00000162", Машинный код - A1[00000000], а `mov eax,[max]` - исходный текст программы, означающий что число хранившееся в переменной `max` записывается в регистр `eax`.

```

50 00000162 A1[00000000]          mov eax,[max]

```

Рис. 2.13: Объяснения третьей строки

- 11) В строке `mov eax,max` я убрала `max` и попробовал создать файл. Выдало ошибку, так как для программы нужно два операнда.
- 12) В файле листинга показывает где именно ошибка и с чем она связана.

```

37                                     mov eax
37 ***** error: invalid combination of opcode and operands

```

Рис. 2.14: Файл листинга без одного операнда

3 Самостоятельная работа.

- 1) Я написала программу для нахождения меньшего из трех чисел. Для большего удобства я сделала ввод чисел с клавиатуры. У меня первый вариант поэтому числа были :17,23,45. Программа вывела меньшее из этих чисел.

```

#include 'in_out.asm'

SECTION .data
A1 DB 'Введите число A: ',0h
B1 DB 'Введите число B: ',0h
C1 DB 'Введите число C: ',0h
otv DB 'Наименьшее число: ',0h
SECTION .bss
min RESB 20
A RESB 20
B RESB 20
C RESB 20

SECTION .text
GLOBAL _start
_start:

mov eax,A1
call sprint

mov ecx,A
mov edx,20
call sread

mov eax, A
call atoi
mov [A],eax

xor eax,eax

mov eax,B1
call sprint

mov ecx,B
mov edx,20
call sread

mov eax,B
call atoi
mov [B],eax

xor eax,eax

mov ecx, [A]

```

Рис. 3.1: Текст программы

```
Введите число А: 17  
Введите число В: 23  
Введите число С: 45  
Наименьшее число: 17
```

Рис. 3.2: Результат работы программы

- 2) Я написала программу, чтобы она вычисляла выражение при введенных Х и А. Для большего удобства, выражение которое будет вычисляться я вывожу в начале работы программы. Так как у меня 1 вариант, то программа написана для 1 варианта.

```

#include 'in_out.asm'

SECTION .data
prim1 DB '2a-x ,x<a' ,0
prim2 DB '8, x=>a',0
X1 DB 'Введите значение X:',0
A1 DB 'Введите значение a:',0
otv DB 'Ответ: ',0

SECTION .bss
X RESB 20
A RESB 20
F RESB 20
SECTION .text
GLOBAL _start
_start:

mov eax,prim1
call sprintf
mov eax,prim2
call sprintf

mov eax,X1
call sprintf

mov ecx,X
mov edx,10
call sread

mov eax,X
call atoi
mov [X],eax

mov eax,A1
call sprintf

mov ecx,A
mov edx,10
call sread

mov eax,A
call atoi
mov [A],eax

```

Рис. 3.3: Текст программы

```
2a-x ,x<a
8, x=>a
Введите значение X:1
Введите значение a:2
Ответ: 3
[aalushin@fedora lab08]$ ./3
2a-x ,x<a
8, x=>a
Введите значение X:2
Введите значение a:1
Ответ: 8
```

Рис. 3.4: Проверка работы программы

4 Вывод

Я изучила команды условного и безусловного перехода. Приобрел навыки написания программ с переходами.