

Отчёт по лабораторной работе 7

дисциплина: Архитектура компьютера

Фархад Ахамд Камран

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Изучение структуры файла листинга	13
2.3	Самостоятельное задание	15
3	Выводы	20

Список иллюстраций

2.1	Создан каталог	6
2.2	Программа lab7-1.asm	7
2.3	Запуск программы lab7-1.asm	7
2.4	Программа lab7-1.asm	8
2.5	Запуск программы lab7-1.asm	9
2.6	Программа lab7-1.asm	10
2.7	Запуск программы lab7-1.asm	10
2.8	Программа lab7-2.asm	12
2.9	Запуск программы lab7-2.asm	12
2.10	Файл листинга lab7-2	13
2.11	Ошибка трансляции lab7-2	14
2.12	Файл листинга с ошибкой lab7-2	15
2.13	Программа lab7-3.asm	16
2.14	Запуск программы lab7-3.asm	16
2.15	Программа lab7-4.asm	18
2.16	Запуск программы lab7-4.asm	19

Список таблиц

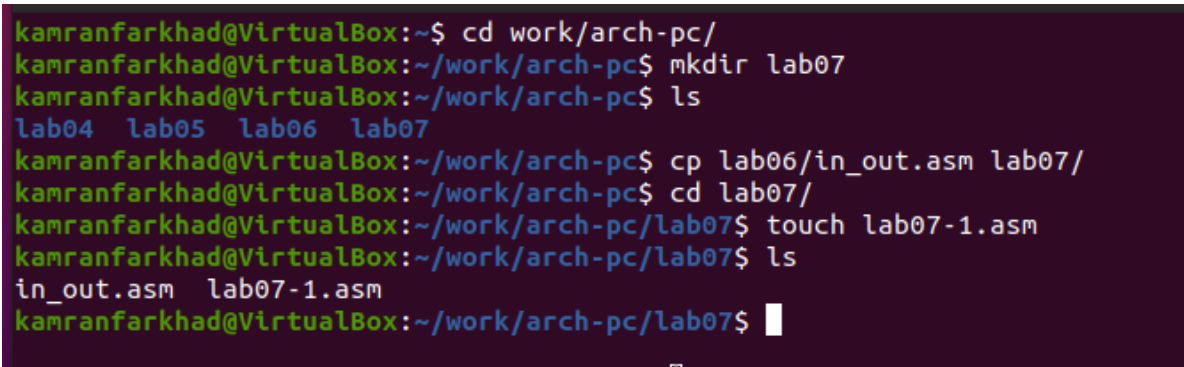
1 Цель работы

Целью работы является изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

2.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы № 7 и файл lab7-1.asm.
(рис. 2.1)



```
kamranfarkhad@VirtualBox:~$ cd work/arch-pc/  
kamranfarkhad@VirtualBox:~/work/arch-pc$ mkdir lab07  
kamranfarkhad@VirtualBox:~/work/arch-pc$ ls  
lab04  lab05  lab06  lab07  
kamranfarkhad@VirtualBox:~/work/arch-pc$ cp lab06/in_out.asm lab07/  
kamranfarkhad@VirtualBox:~/work/arch-pc$ cd lab07/  
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$ touch lab07-1.asm  
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$ ls  
in_out.asm  lab07-1.asm  
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.1: Создан каталог

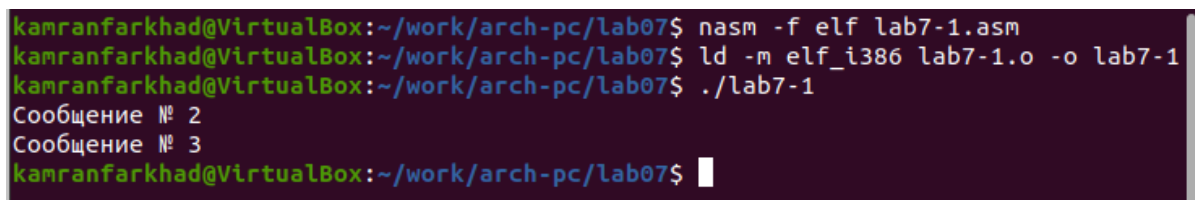
Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. Написал в файл `lab7-1.asm` текст программы из листинга 7.1. (рис. 2.2)



```
mc [kamranfarkhad@VirtualBox]:~/...  
/home/ka~7-1.asm [----] 13 L:[ 1+20 21/ 26]  
%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  
  
_label2:  
mov eax, msg2  
call sprintLF  
  
_label3:  
mov eax, msg3  
call sprintLF  
  
_end:  
call quit
```

Рис. 2.2: Программа lab7-1.asm

Создаю исполняемый файл и запускаю его. (рис. 2.3)

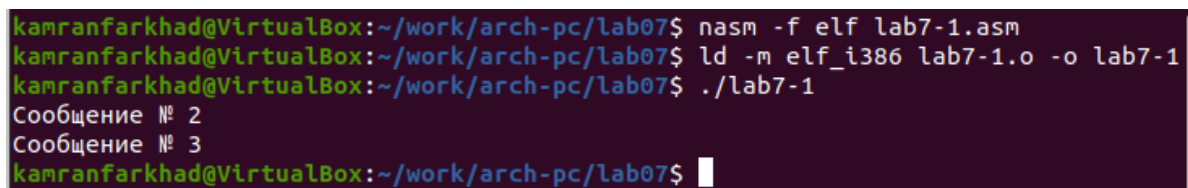


```
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm  
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1  
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$ ./lab7-1  
Сообщение № 2  
Сообщение № 3  
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.3: Запуск программы lab7-1.asm

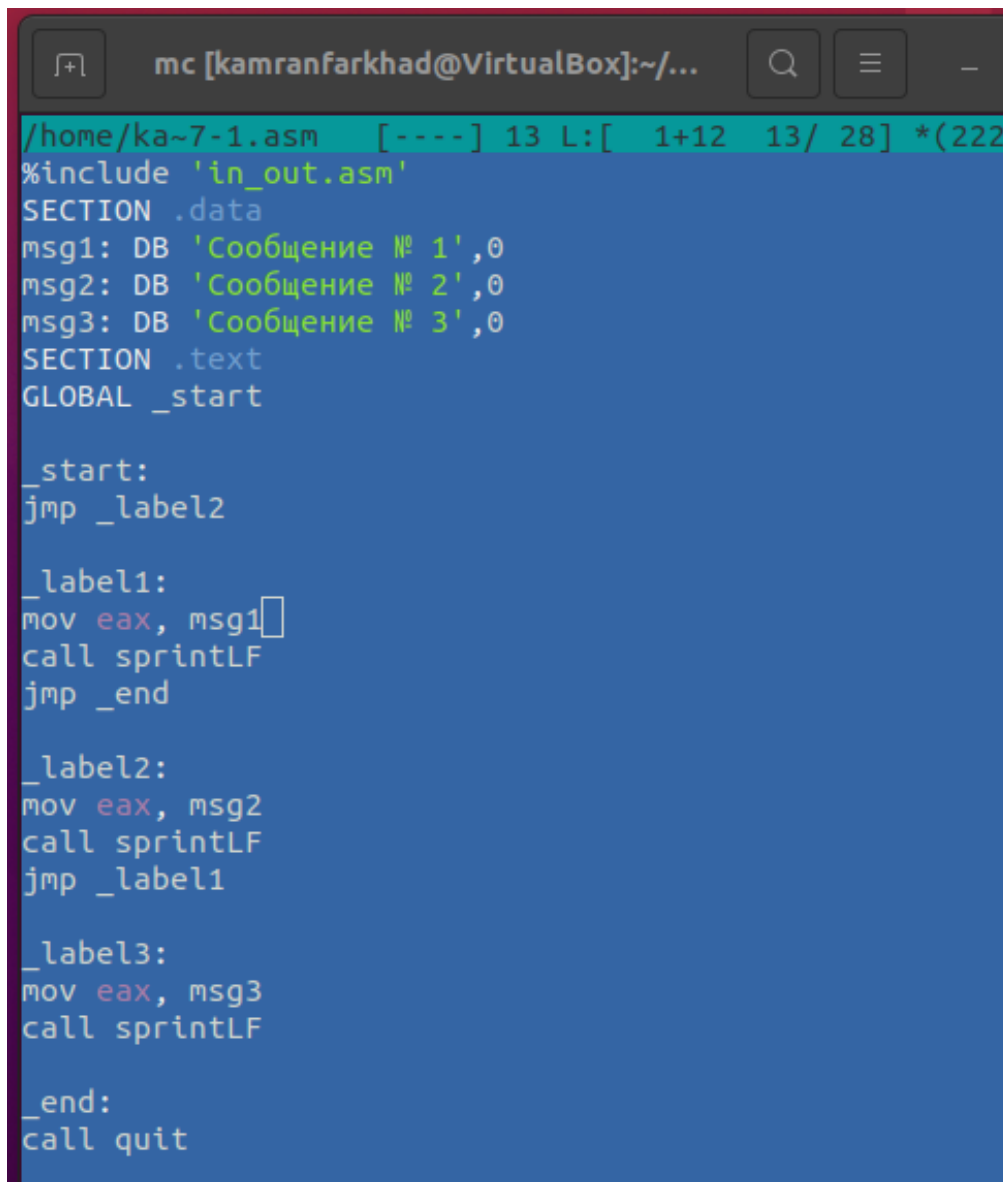
Инструкция `jmp` позволяет осуществлять переходы не только вперед, но и назад. Изменим программу так, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавляем инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавляем инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`).

Изменяю текст программы в соответствии с листингом 7.2. (рис. 2.4) (рис. 2.5)



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

Рис. 2.4: Программа lab7-1.asm

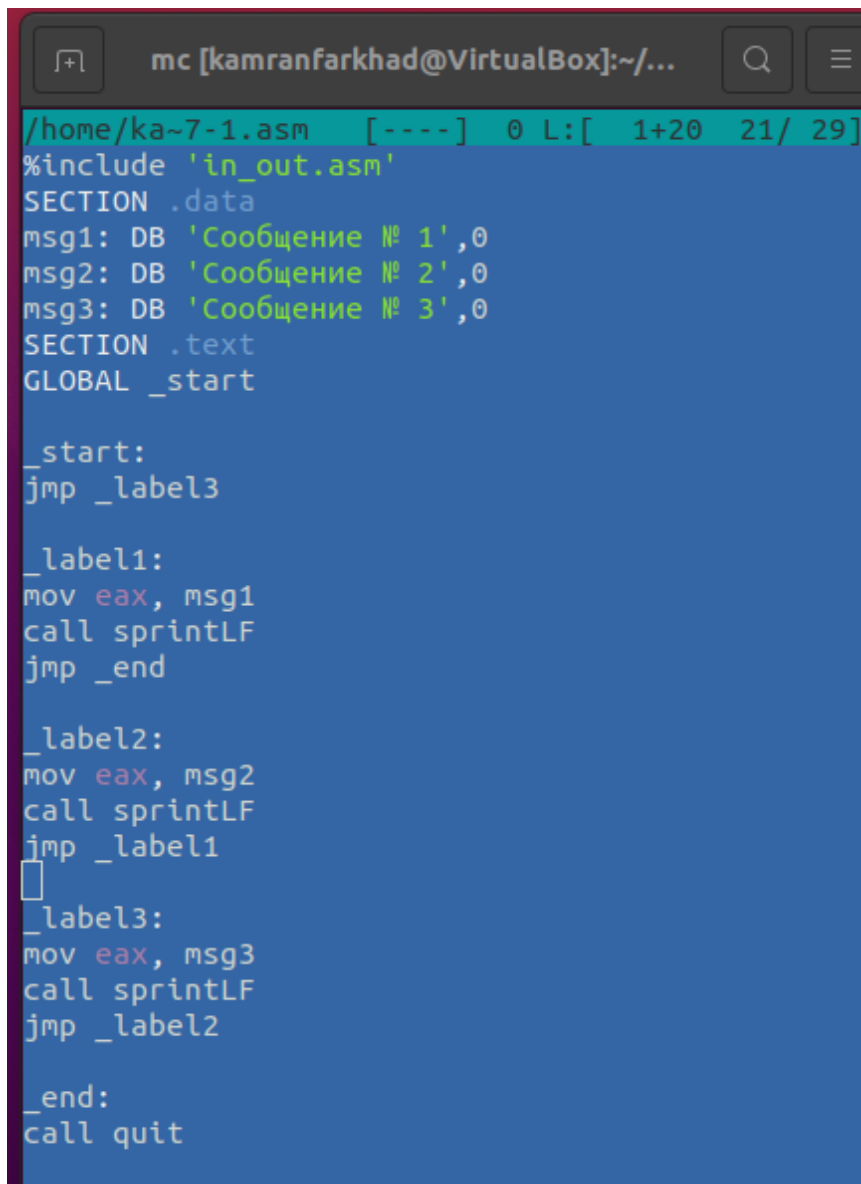


```
mc [kamranfarkhad@VirtualBox]:~/...  
/home/ka~7-1.asm [----] 13 L:[ 1+12 13/ 28] *(222  
%include 'in_out.asm'  
SECTION .data  
msg1: DB 'Сообщение № 1',0  
msg2: DB 'Сообщение № 2',0  
msg3: DB 'Сообщение № 3',0  
SECTION .text  
GLOBAL _start  
  
_start:  
jmp _label2  
  
_label1:  
mov eax, msg1  
call sprintLF  
jmp _end  
  
_label2:  
mov eax, msg2  
call sprintLF  
jmp _label1  
  
_label3:  
mov eax, msg3  
call sprintLF  
  
_end:  
call quit
```

Рис. 2.5: Запуск программы lab7-1.asm

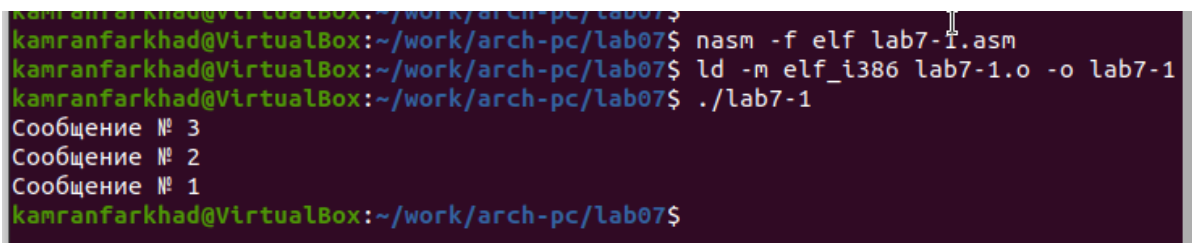
Изменил текст программы, чтобы вывод программы был следующим (рис. 2.6) (рис. 2.7):

Сообщение № 3
Сообщение № 2
Сообщение № 1



```
mc [kamranfarkhad@VirtualBox]:~/...  
/home/ka~7-1.asm [----] 0 L: [ 1+20 21/ 29]  
%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 _label3  
  
_label1:  
mov eax, msg1  
call sprintLF  
jmp _end  
  
_label2:  
mov eax, msg2  
call sprintLF  
jmp _label1  
  
_label3:  
mov eax, msg3  
call sprintLF  
jmp _label2  
  
_end:  
call quit
```

Рис. 2.6: Программа lab7-1.asm

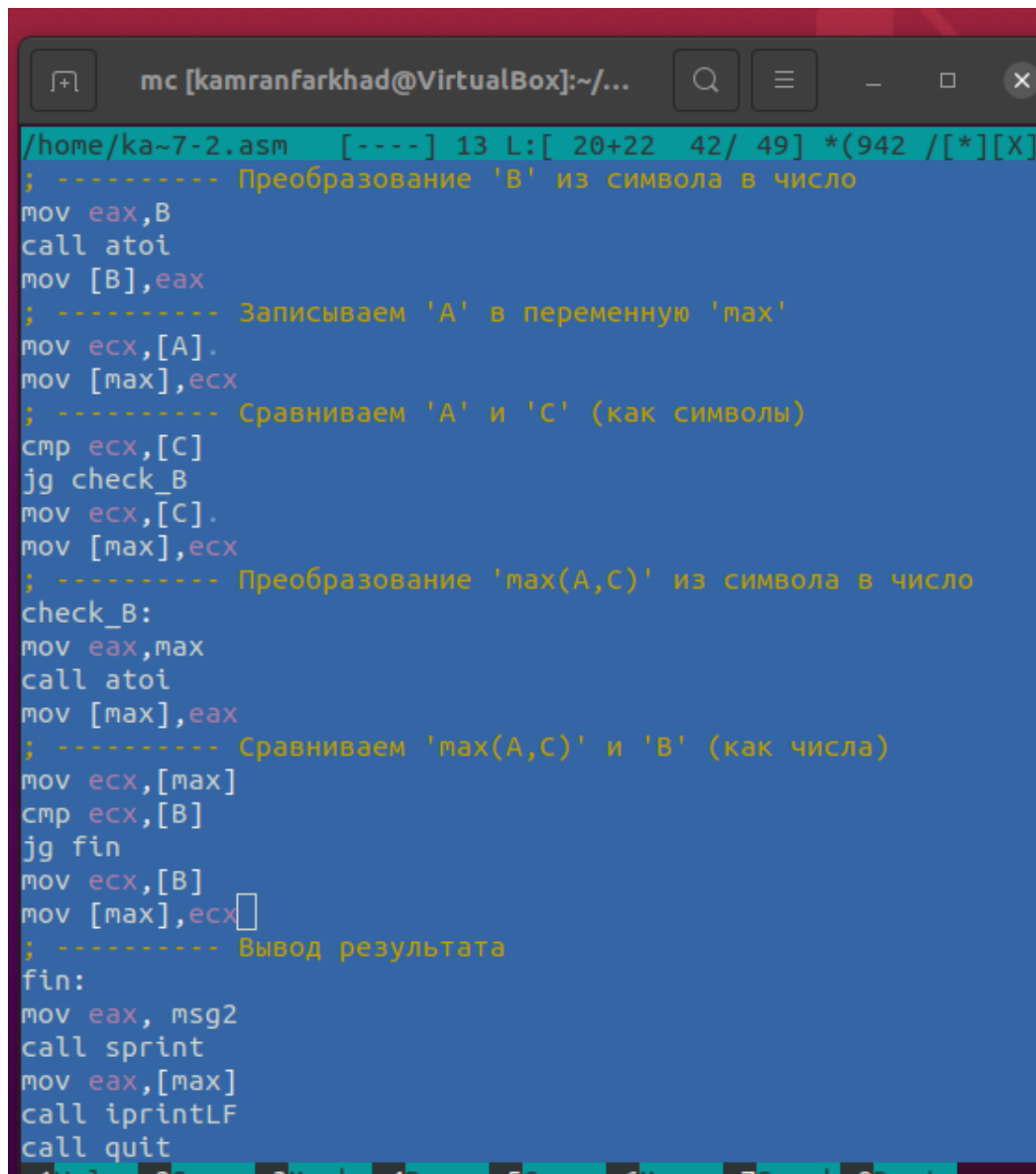


```
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$  
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm  
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1  
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$ ./lab7-1  
Сообщение № 3  
Сообщение № 2  
Сообщение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

Рис. 2.7: Запуск программы lab7-1.asm

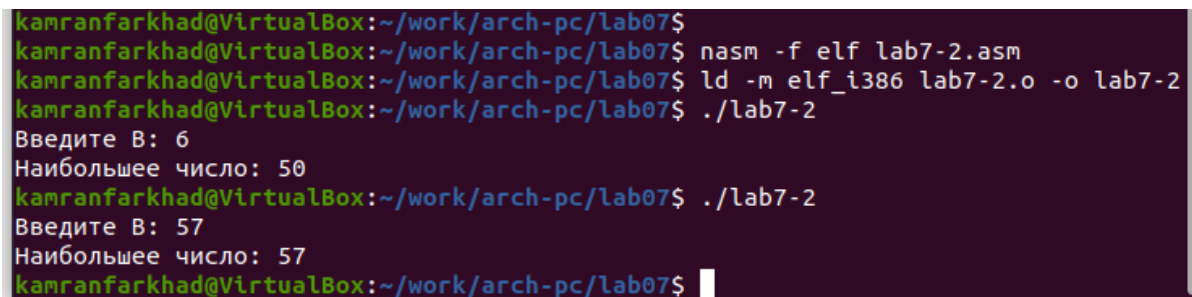
Использование инструкции `jmp` приводит к переходу в любом случае. Однако часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А, В и С. Значения для А и С задаются в программе, значение В вводится с клавиатуры.

Создаю исполняемый файл и проверяю его работу для разных значений В (рис. 2.8) (рис. 2.9).



```
mc [kamranfarkhad@VirtualBox]:~/...
/home/ka~7-2.asm [----] 13 L: [ 20+22 42/ 49] *(942 /[*])[X]
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi
mov [B],eax
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A].
mov [max],ecx
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C]
jg check_B
mov ecx,[C].
mov [max],ecx
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi
mov [max],eax
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B]
jg fin
mov ecx,[B]
mov [max],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint
mov eax,[max]
call iprintLF
call quit
```

Рис. 2.8: Программа lab7-2.asm



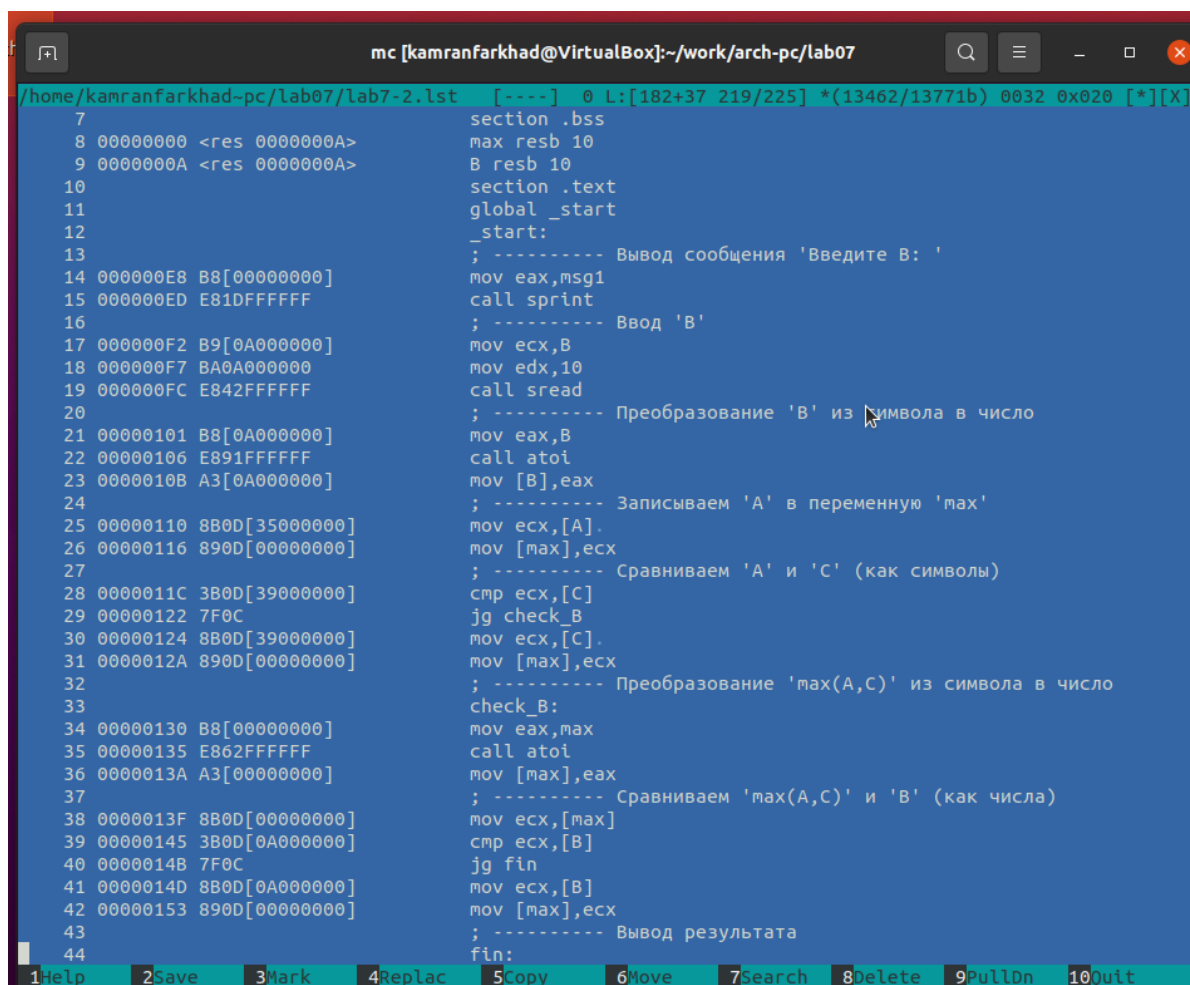
```
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 6
Наибольшее число: 50
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 57
Наибольшее число: 57
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.9: Запуск программы lab7-2.asm

2.2 Изучение структуры файла листинга

Обычно `nasm` создает в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ `-l` и задав имя файла листинга в командной строке.

Создаю файл листинга для программы из файла `lab7-2.asm` (рис. 2.10)



```
/home/kamranfarkhad-pc/lab07/lab7-2.lst [----] 0 L:[182+37 219/225] *(13462/13771b) 0032 0x020 [*][X]
7      section .bss
8      00000000 <res 0000000A>      max resb 10
9      0000000A <res 0000000A>      B resb 10
10     section .text
11     global _start
12     _start:
13     ; ----- Вывод сообщения 'Введите B: '
14     000000E8 B8[00000000]      mov eax,msg1
15     000000ED E81DFFFFFF      call sprint
16     ; ----- Ввод 'B'
17     000000F2 B9[0A000000]      mov ecx,B
18     000000F7 BA0A000000      mov edx,10
19     000000FC E842FFFFFF      call sread
20     ; ----- Преобразование 'B' из символа в число
21     00000101 B8[0A000000]      mov eax,B
22     00000106 E891FFFFFF      call atoi
23     0000010B A3[0A000000]      mov [B],eax
24     ; ----- Записываем 'A' в переменную 'max'
25     00000110 8B0D[35000000]      mov ecx,[A]
26     00000116 890D[00000000]      mov [max],ecx
27     ; ----- Сравниваем 'A' и 'C' (как символы)
28     0000011C 3B0D[39000000]      cmp ecx,[C]
29     00000122 7F0C      jg check_B
30     00000124 8B0D[39000000]      mov ecx,[C]
31     0000012A 890D[00000000]      mov [max],ecx
32     ; ----- Преобразование 'max(A,C)' из символа в число
33     check_B:
34     00000130 B8[00000000]      mov eax,max
35     00000135 E862FFFFFF      call atoi
36     0000013A A3[00000000]      mov [max],eax
37     ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38     0000013F 8B0D[00000000]      mov ecx,[max]
39     00000145 3B0D[0A000000]      cmp ecx,[B]
40     0000014B 7F0C      jg fin
41     0000014D 8B0D[0A000000]      mov ecx,[B]
42     00000153 890D[00000000]      mov [max],ecx
43     ; ----- Вывод результата
44     fin:
```

Рис. 2.10: Файл листинга lab7-2

Ознакомимся с его форматом и содержимым.

строка 203

- 28 - номер строки
- 0000011C - адрес

- 3B0D[39000000] - машинный код
- str esx,[C] - код программы

строка 204

- 29 - номер строки
- 00000122 - адрес
- 7F0C - машинный код
- jg check_B - код программы

строка 205

- 30 - номер строки
- 00000124 - адрес
- 8B0D[39000000] - машинный код
- mov esx,[C] - код программы

Открываю файл с программой lab7-2.asm и в инструкции с двумя операндами удаляю один операнд. Выполняю трансляцию с получением файла листинга. (рис. 2.11) (рис. 2.12)

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

Рис. 2.11: Ошибка трансляции lab7-2

```
/home/kamranfarkhad-pc/lab07/lab7-2.lst  [----] 51 L:[181+18 199/226] *(12228/13861b) 0010 0x00A [*][X]
6 00000039 35300000 C dd '50'
7 section .bss
8 00000000 <res 0000000A> max resb 10
9 0000000A <res 0000000A> B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B: '
14 000000E8 B8[00000000] mov eax,msg1
15 000000ED E81DFFFFFF call sprint
16 ; ----- Ввод 'B'
17 000000F2 B9[0A000000] mov ecx,B
18 000000F7 BA0A000000 mov edx,10
19 000000FC E842FFFFFF call sread
20 ; ----- Преобразование 'B' из символа в число
21 mov eax,
21 *****
22 00000101 E896FFFFFF call atoi
23 00000106 A3[0A000000] mov [B],eax
24 ; ----- Записываем 'A' в переменную 'max'
25 0000010B 8B0D[35000000] mov ecx,[A]
26 00000111 890D[00000000] mov [max],ecx
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 00000117 3B0D[39000000] cmp ecx,[C]
29 0000011D 7F0C jg check_B
30 0000011F 8B0D[39000000] mov ecx,[C]
31 00000125 890D[00000000] mov [max],ecx
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 0000012B B8[00000000] mov eax,max
35 00000130 E867FFFFFF call atoi
36 00000135 A3[00000000] mov [max],eax
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013A 8B0D[00000000] mov ecx,[max]
39 00000140 3B0D[0A000000] cmp ecx,[B]
40 00000146 7F0C jg fin
41 00000148 8B0D[0A000000] mov ecx,[B]
42 0000014E 890D[00000000] mov [max],ecx
```

Рис. 2.12: Файл листинга с ошибкой lab7-2

Объектный файл не смог создаться из-за ошибки. Но получился листинг, где выделено место ошибки.

2.3 Самостоятельное задание

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 № 6. Создайте исполняемый файл и проверьте его работу (рис. 2.13) (рис. 2.14)

для варианта 4 - 8,88,68

```

/home/ka~7-3.asm [---] 17 L: [ 33+32 65/ 70] *(933 [*][X]
    mov eax,B
    call atoi
    mov [B],eax

    mov eax,msgC
    call sprintf
    mov ecx,C
    mov edx,80
    call sread.
    mov eax,C
    call atoi
    mov [C],eax...
....
    mov ecx,[A]
    mov [min],ecx

    cmp ecx, [B]
    jl check_C
    mov ecx, [B]
    mov [min], ecx

check_C:
    cmp ecx, [C]
    jl finish
    mov ecx,[C]
    mov [min],ecx.

finish:
    mov eax,answer
    call sprintf

    mov eax, [min]
    call iprintLF
    call quit

```

Рис. 2.13: Программа lab7-3.asm

```

kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-3.o -o lab7-3
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$ ./lab7-3
Input A: 8
Input B: 88
Input C: 68
Smallest: 8
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$

```

Рис. 2.14: Запуск программы lab7-3.asm

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. (рис. 2.15) (рис. 2.16)

для варианта 4

$$\begin{cases} 2x + a, a \neq 0 \\ 2x + 1, a = 0 \end{cases}$$

При $x = 3, a = 0$ получается 7.

При $x = 3, a = 2$ получается 8.

```

/home/ka~7-4.asm [----] 13 L:[ 16+36 52/ 54] *(725 [*]
    call sprint
    mov ecx,A
    mov edx,80
    call sread
    mov eax,A
    call atoi.
    mov [A],eax

    mov eax,msgX
    call sprint
    mov ecx,X
    mov edx,80
    call sread.
    mov eax,X
    call atoi
    mov [X],eax...

    mov ebx, [A]
    mov edx, 0
    cmp ebx, edx
    jne first
    jmp second

first:
    mov eax,[X]
    mov ebx,2
    mul ebx
add    eax,[A]
    call iprintLF.
    call quit
second:
    mov eax,[X]
    mov ebx,2
    mul ebx
    add eax,1
    call iprintLF.
    call quit

```

Рис. 2.15: Программа lab7-4.asm

```
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$  
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm  
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-4.o -o lab7-4  
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$ ./lab7-4  
Input A: 0  
Input X: 3  
7  
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$ ./lab7-4  
Input A: 2  
Input X: 3  
8  
kamranfarkhad@VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.16: Запуск программы lab7-4.asm

3 Выводы

Изучили команды условного и безусловного переходов, познакомились с фалом листинга.