

Отчёт по лабораторной работе 7

Архитектура компьютера

Кази ар Рафи НКАбд-03-24

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	21

Список иллюстраций

2.1	Программа lab7-1.asm	7
2.2	Запуск программы lab7-1.asm	8
2.3	Программа lab7-1.asm	9
2.4	Запуск программы lab7-1.asm	9
2.5	Программа lab7-1.asm	10
2.6	Запуск программы lab7-1.asm	11
2.7	Программа lab7-2.asm	12
2.8	Запуск программы lab7-2.asm	13
2.9	Файл листинга lab7-2	14
2.10	Ошибка трансляции lab7-2	15
2.11	Файл листинга с ошибкой lab7-2	16
2.12	Программа lab7-3.asm	17
2.13	Запуск программы lab7-3.asm	17
2.14	Программа lab7-4.asm	19
2.15	Запуск программы lab7-4.asm	20

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

1. Создал каталог для программ лабораторной работы № 7 и файл lab7-1.asm
2. 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.



```
Открыть ▾ + lab7-1.asm  
~/work/a... pc/lab07  
%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. 2.1: Программа lab7-1.asm


Создал исполняемый файл и запустил его.

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

Рис. 2.2: Запуск программы lab7-1.asm

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

Изменил текст программы в соответствии с листингом 7.2.



```
Открыть ▾ + lab7-1.asm
~/work/a... pc/lab07

%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

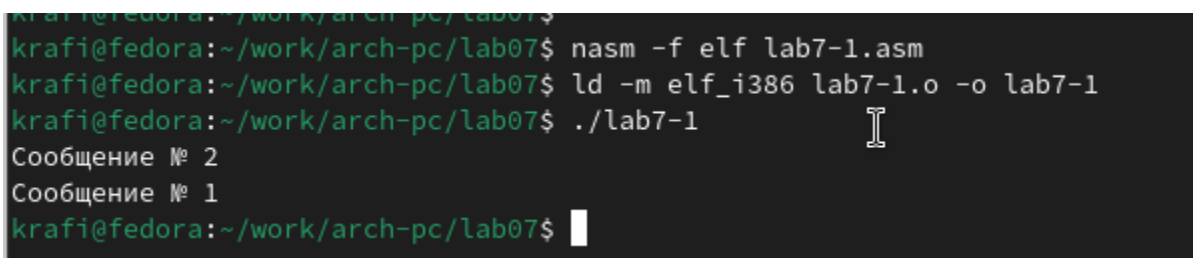
_label1:
mov eax, msg1
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Рис. 2.3: Программа lab7-1.asm



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

Рис. 2.4: Запуск программы lab7-1.asm

Изменил текст программы, изменив инструкции `jmp`, чтобы вывод программы был следующим:

Сообщение № 3

Сообщение № 2

Сообщение № 1



```
Открыть ▾ + lab7-1.asm
~/work/a... pc/lab07

%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
```

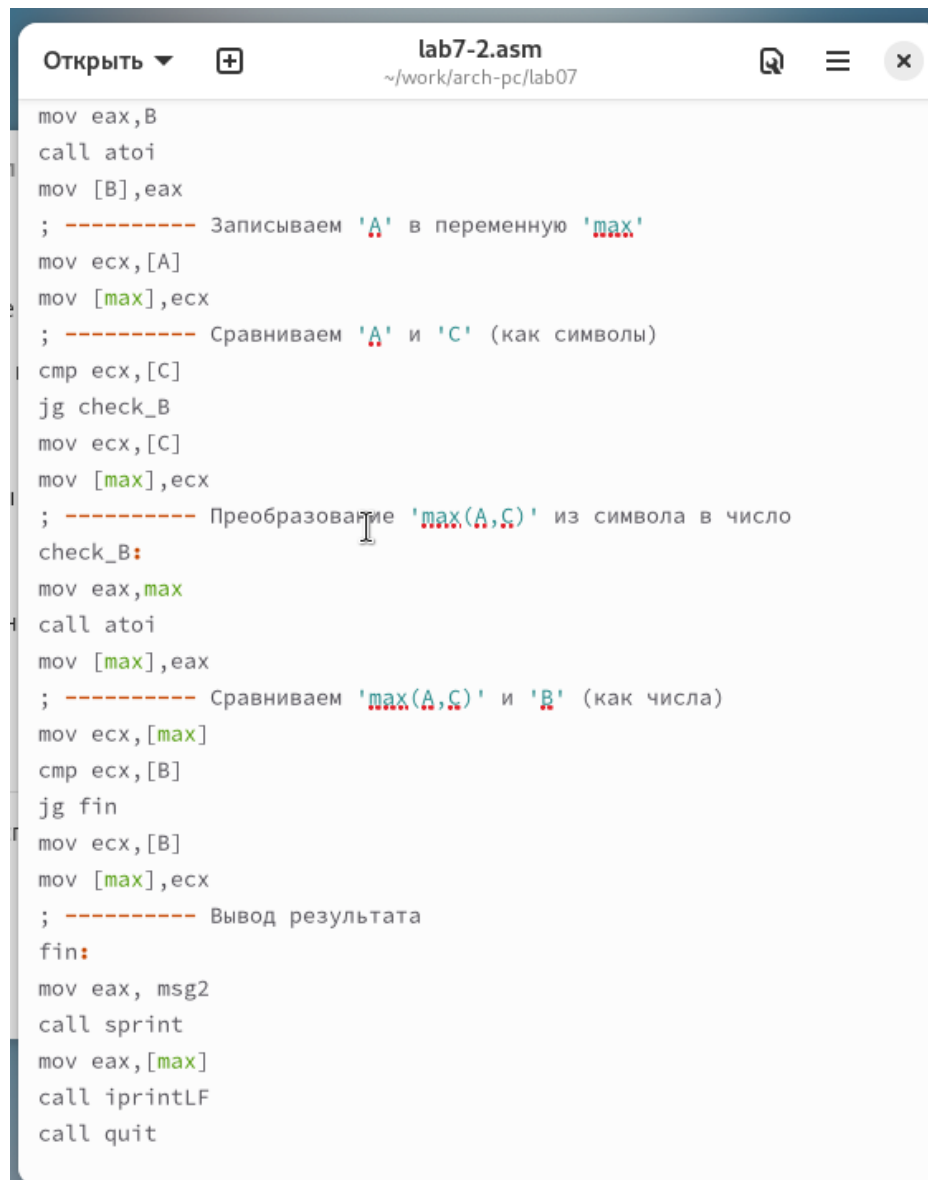
Рис. 2.5: Программа lab7-1.asm

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

Рис. 2.6: Запуск программы lab7-1.asm

3. 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: A, B и C. Значения для A и C задаются в программе, значение B вводится с клавиатуры.

Создал исполняемый файл и проверил его работу для разных значений B.



```
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. 2.7: Программа lab7-2.asm

```

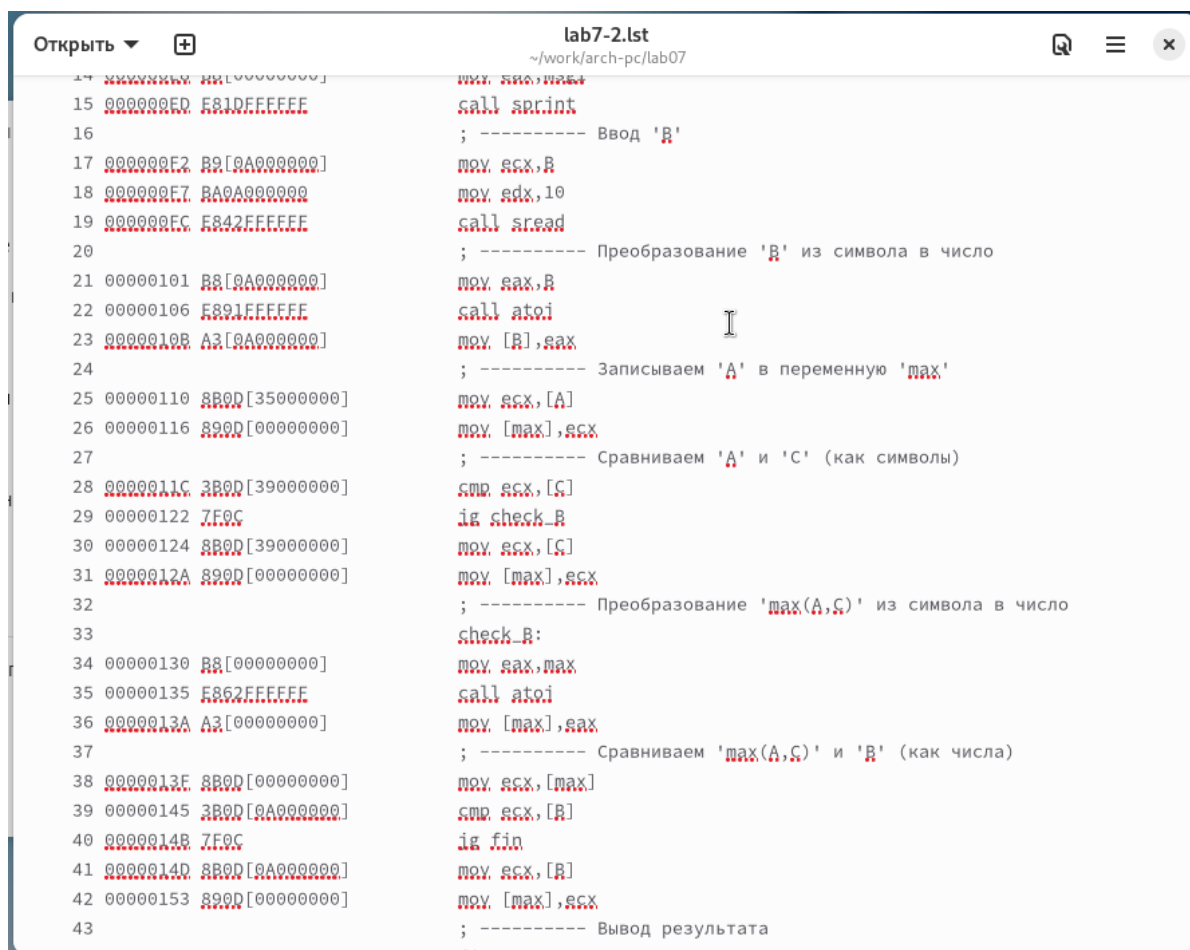
krafi@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
krafi@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
krafi@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 30
Наибольшее число: 50
krafi@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 60
Наибольшее число: 60
krafi@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 80
Наибольшее число: 80
krafi@fedora:~/work/arch-pc/lab07$

```

Рис. 2.8: Запуск программы lab7-2.asm

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

Создал файл листинга для программы из файла lab7-2.asm



```
14 00000000 00000000 mov eax,0
15 0000000D E810FFFFFF 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 sax,B
22 00000106 F801FFFFFF 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 sax,max
35 00000135 F862FFFFFF 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 00000159 00000000
```

Рис. 2.9: Файл листинга lab7-2

Внимательно ознакомился с его форматом и содержимым. Подробно объясню содержимое трёх строк файла листинга по выбору.

строка 34

- 34 - номер строки
- 00000130 - адрес
- B8[00000000] - машинный код
- mov eax,max - код программы

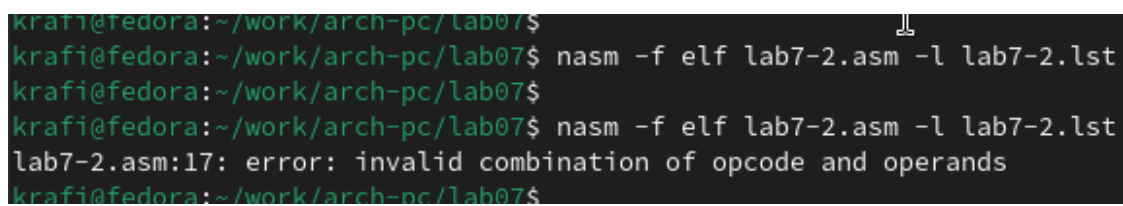
строка 35

- 35 - номер строки
- 00000135 - адрес
- E862FFFFFF - машинный код
- call atoi- код программы

строка 36

- 36 - номер строки
- 0000013A - адрес
- A3[00000000] - машинный код
- mov [max],eax - код программы

Открыл файл с программой lab7-2.asm и в инструкции с двумя операндами удалил один операнд. Выполнил трансляцию с получением файла листинга.



```

krafi@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
krafi@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:17: error: invalid combination of opcode and operands
krafi@fedora:~/work/arch-pc/lab07$

```

Рис. 2.10: Ошибка трансляции lab7-2

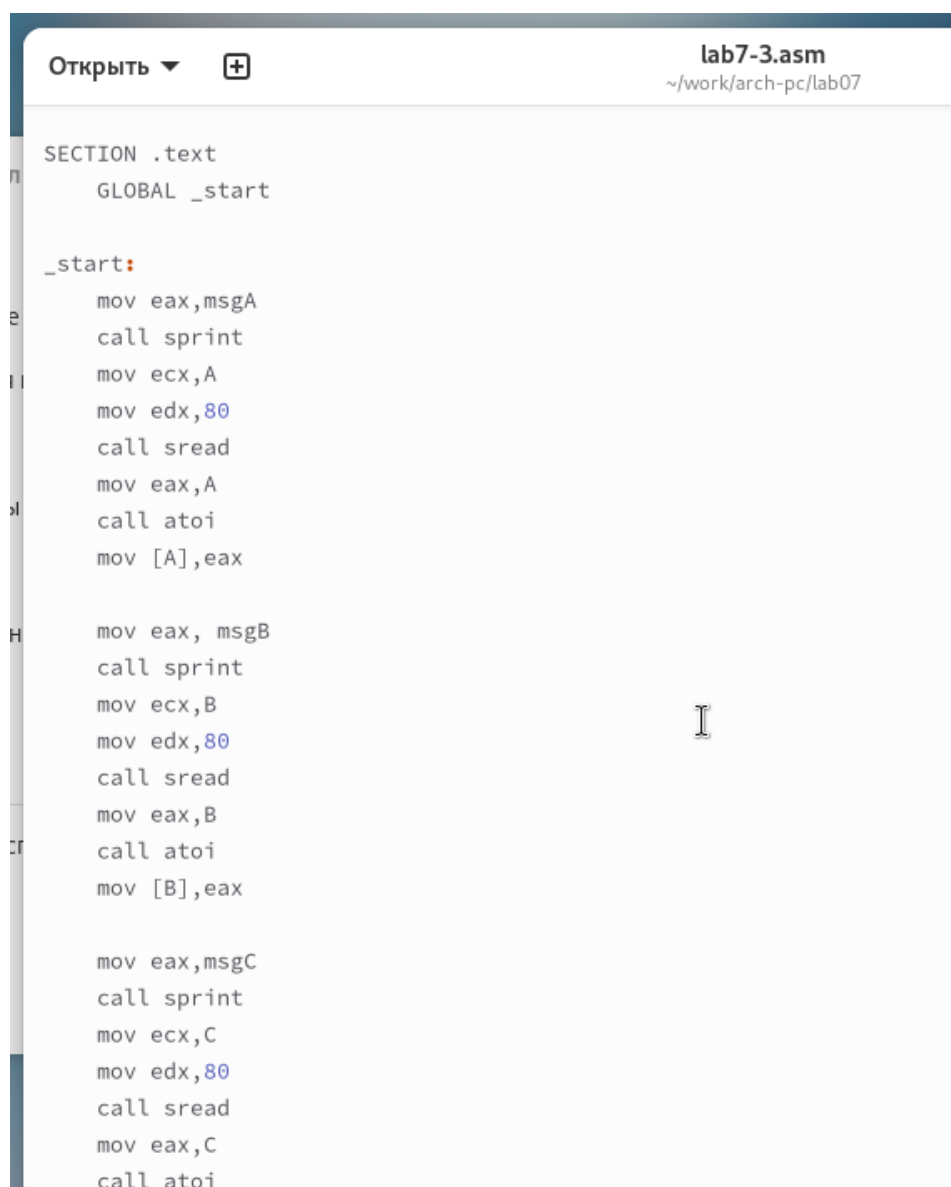
```
10      section .text
11      global _start
12      _start:
13          ; ----- Вывод сообщения 'Введите B: '
14      000000E8 B8[00000000]    mov eax, msg1
15      000000ED E810FFFFFF    call sprint
16          ; ----- Ввод 'B'
17      000000F2 B8[00000000]    mov ecx,
18      000000F7 F847FFFFFF    error: invalid combination of opcode and operands
19      000000FC B8[00000000]    mov edx, 10
20      00000101 F896FFFFFF    call sread
21          ; ----- Преобразование 'B' из символа в число
22      00000106 A3[00000000]    mov eax, B
23      0000010B 8B0D[35000000]    call atoi
24      00000110 890D[00000000]    mov [B], eax
25          ; ----- Записываем 'A' в переменную 'max'
26      00000117 8B0D[39000000]    mov ecx, [A]
27      0000011D 7F0C          mov [max], ecx
28          ; ----- Сравниваем 'A' и 'C' (как символы)
29      00000125 8B0D[39000000]    cmp ecx, [C]
30      0000012B 7F0C          jg check_B
31      00000130 8B0D[39000000]    mov ecx, [C]
32      00000137 890D[00000000]    mov [max], ecx
33          ; ----- Преобразование 'max(A,C)' из символа в число
34      0000013D 8B0D[39000000]    check_B:
35      00000142 8B0D[39000000]    mov eax, max
36      00000149 F867FFFFFF    call atoi
37      0000014E A3[00000000]    mov [max], eax
```

Рис. 2.11: Файл листинга с ошибкой lab7-2

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

5. 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

для варианта 13 - 84,32,77



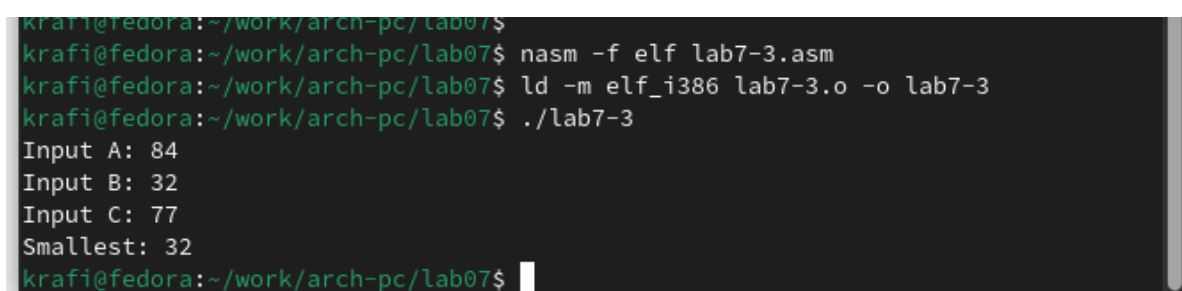
```
SECTION .text
    GLOBAL _start

_start:
    mov eax,msgA
    call sprintf
    mov ecx,A
    mov edx,80
    call sread
    mov eax,A
    call atoi
    mov [A],eax

    mov eax, msgB
    call sprintf
    mov ecx,B
    mov edx,80
    call sread
    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
```

Рис. 2.12: Программа lab7-3.asm



```
krafi@fedora:~/work/arch-pc/lab07$
krafi@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
krafi@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-3.o -o lab7-3
krafi@fedora:~/work/arch-pc/lab07$ ./lab7-3
Input A: 84
Input B: 32
Input C: 77
Smallest: 32
krafi@fedora:~/work/arch-pc/lab07$
```

Рис. 2.13: Запуск программы lab7-3.asm

6. 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.

для варианта 13

$$\begin{cases} a - 7, a \geq 7 \\ ax, a < 7 \end{cases}$$

```
Открыть ▾  lab7-4.asm  
~/work/arch-pc/lab07  
  
mov eax, msgA  
call sprint  
mov ecx, X  
mov edx, 80  
call sread  
mov eax, X  
call atoi  
mov [X], eax  
  
mov ebx, [A]  
mov edx, 7  
cmp ebx, edx  
jge first  
jmp second  
  
first:  
    mov eax, [A]  
    sub eax, 7  
    call iprintLF  
    call quit  
  
second:  
    mov eax, [X]  
    mov ebx, [A]  
    mul ebx  
    call iprintLF  
    call quit
```

Рис. 2.14: Программа lab7-4.asm

```
krafi@fedora:~/work/arch-pc/lab07$  
krafi@fedora:~/work/arch-pc/lab07$  
krafi@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm  
krafi@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-4.o -o lab7-4  
krafi@fedora:~/work/arch-pc/lab07$ ./lab7-4  
Input A: 9  
Input X: 3  
2  
krafi@fedora:~/work/arch-pc/lab07$ ./lab7-4  
Input A: 4  
Input X: 6  
24  
krafi@fedora:~/work/arch-pc/lab07$
```

Рис. 2.15: Запуск программы lab7-4.asm

3 Выводы

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