

Отчёт по лабораторной работе 7

дисциплина: Архитектура компьютера

Магомедов Султан Гасанович

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	16
3	Выводы	21

Список иллюстраций

2.1	Создан каталог	6
2.2	Программа lab7-1.asm	7
2.3	Запуск программы lab7-1.asm	8
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	13
2.10	Файл листинга lab7-2	14
2.11	Ошибка трансляции lab7-2	15
2.12	Файл листинга с ошибкой lab7-2	16
2.13	Программа task1.asm	17
2.14	Запуск программы task1.asm	18
2.15	Программа task2.asm	19
2.16	Запуск программы task2.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

2.1 Реализация переходов в NASM

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

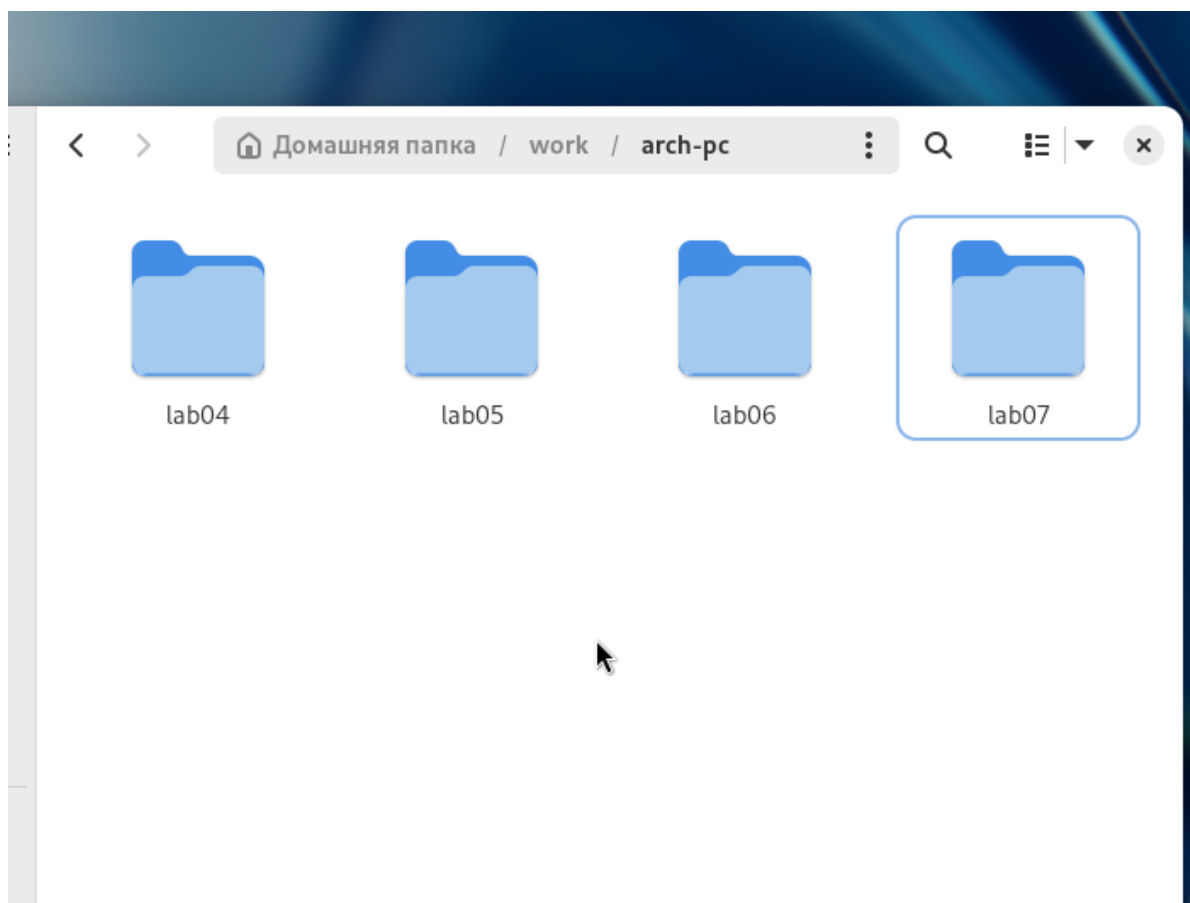


Рис. 2.1: Создан каталог

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)



```
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 sprintf

_label2:
mov eax, msg2
call sprintf

_label3:
mov eax, msg3
call sprintf

_end:
call quit
```

Рис. 2.2: Программа `lab7-1.asm`

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

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

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

Рис. 2.4: Программа lab7-1.asm



```
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. 2.5: 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

```

%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

```

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

```
Открыть ▾ + lab7-2.asm
~\work\a... pc\lab07
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

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

```
26 00000116 890D[00000000]    mov [max],ecx
27                               ; ----- Сравниваем 'A' и 'C' (как символы)
28 0000011C 3B0D[39000000]    cmp ecx,[C]
29 00000122 7F0C             ig 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 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             ig fin
41 0000014D 8B0D[0A000000]    mov ecx,[B]
42 00000153 890D[00000000]    mov [max],ecx
43                               ; ----- Вывод результата
44                               fin:
45 00000159 B8[13000000]    mov eax, msg2
46 0000015E F8ACFFFFFF      call sprint
47 00000163 A1[00000000]    mov eax,[max]
48 00000168 F819FFFFFF      call iprintlf
49 0000016D F869FFFFFF      call quit
```

Рис. 2.10: Файл листинга lab7-2

Ознакомимся с его форматом и содержимым.

строка 211

- 34 - номер строки
- 0000012E - адрес
- B8[00000000] - машинный код
- mov eax,max - код программы

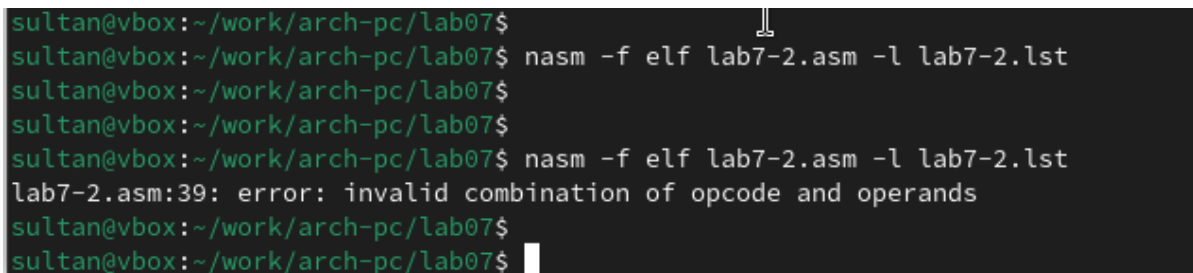
строка 212

- 35 - номер строки
- 00000133 - адрес
- E864FFFFFF - машинный код
- call atoi - код программы

строка 213

- 36 - номер строки
- 00000138 - адрес
- A3[00000000] - машинный код
- mov [max],eax - 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)



```
sultan@vbox:~/work/arch-pc/lab07$
sultan@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
sultan@vbox:~/work/arch-pc/lab07$
sultan@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:39: error: invalid combination of opcode and operands
sultan@vbox:~/work/arch-pc/lab07$
sultan@vbox:~/work/arch-pc/lab07$
```

Рис. 2.11: Ошибка трансляции lab7-2

```
lab7-2.lst
~/work/arch-pc/lab07

29 00000122 7F0C      ig 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 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                                     cmp ecx,
39                                     ***** error: invalid combination of opcode and operands
40 00000145 7F0C      ig fin
41 00000147 8B0D[0A000000]  mov ecx,[B]
42 0000014D 890D[00000000]  mov [max],ecx
43                                     ; ----- Вывод результата
44                                     fin:
45 00000153 B8[13000000]    mov eax,msg2
46 00000158 F8B2FFFFFF      call sprintf
47 0000015D A1[00000000]    mov eax,[max]
48 00000162 F81FFFFFFF      call iprintf
49 00000167 F86FFFFFFF      call quit
```

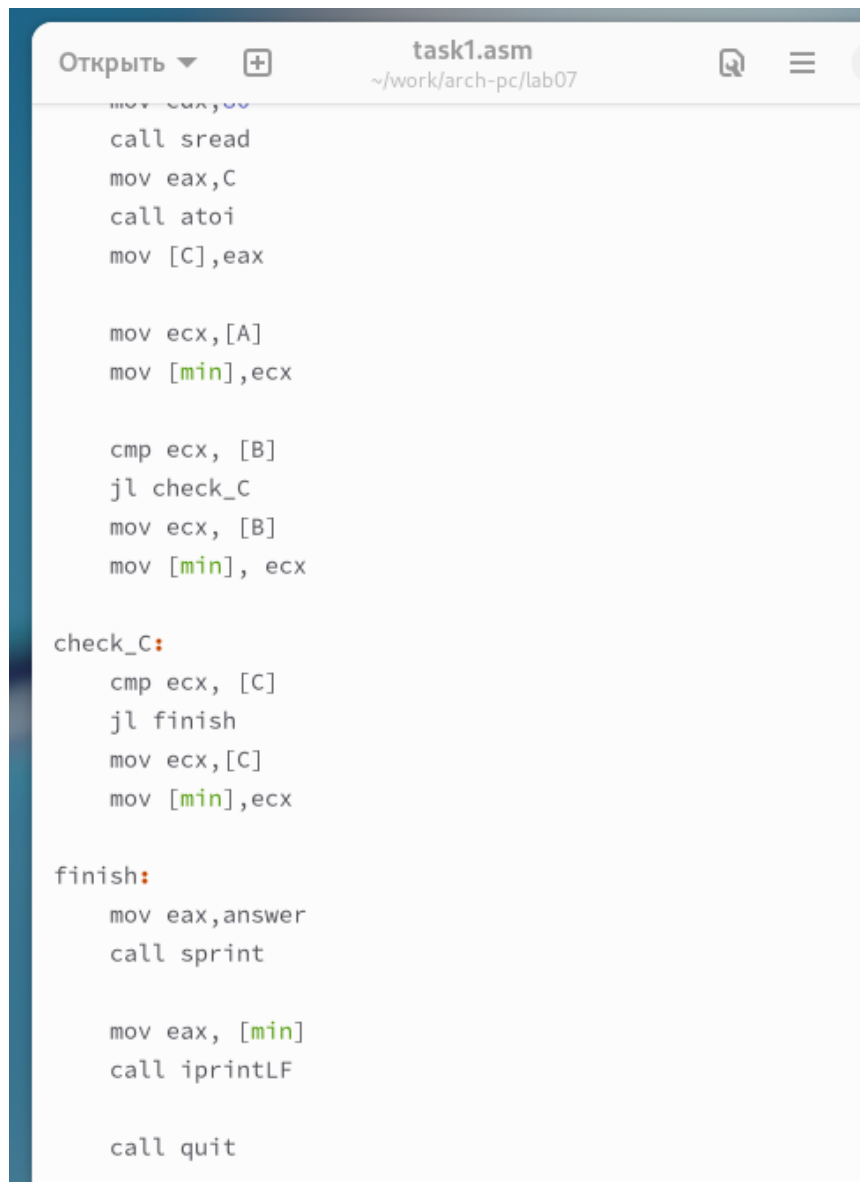
Рис. 2.12: Файл листинга с ошибкой lab7-2

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 13 - 84,32,77



```
Открыть + task1.asm ~/work/arch-pc/lab07
mov ecx, 0
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 sprint

    mov eax, [min]
    call iprintLF

    call quit
```

Рис. 2.13: Программа task1.asm

```

sultan@vbox:~/work/arch-pc/lab07$
sultan@vbox:~/work/arch-pc/lab07$ nasm -f elf task1.asm
sultan@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 task1.o -o task1
sultan@vbox:~/work/arch-pc/lab07$ ./task1
Input A: 84
Input B: 32
Input C: 77
Smallest: 32
sultan@vbox:~/work/arch-pc/lab07$

```

Рис. 2.14: Запуск программы task1.asm

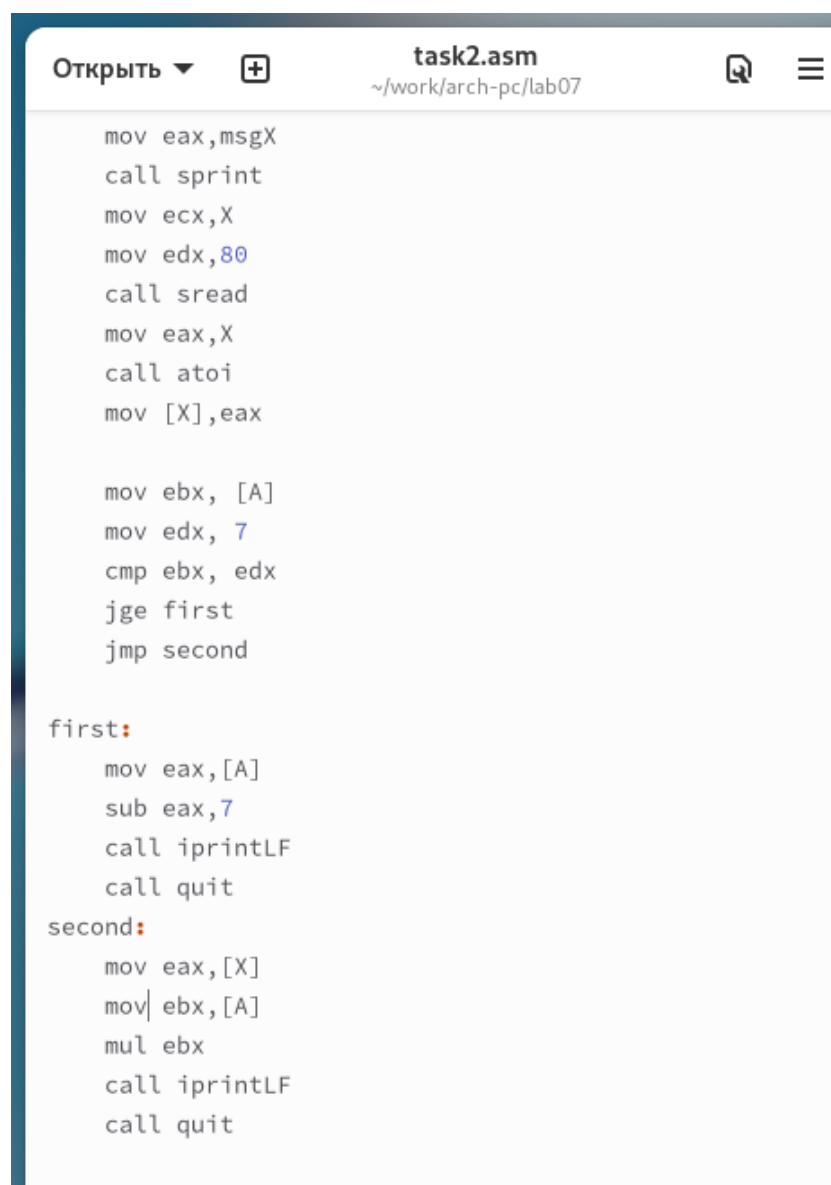
Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений X и a из 7.6. (рис. 2.15) (рис. 2.16)

для варианта 13

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

При $x = 3, a = 9$ получается 2.

При $x = 6, a = 4$ получается 24.



```
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, 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. 2.15: Программа task2.asm

```
sultan@vbox:~/work/arch-pc/lab07$ nasm -f elf task2.asm
sultan@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 task2.o -o task2
sultan@vbox:~/work/arch-pc/lab07$ ./task2
Input A: 9
Input X: 3
2
sultan@vbox:~/work/arch-pc/lab07$ ./task2
Input A: 4
Input X: 6
24
sultan@vbox:~/work/arch-pc/lab07$
```

Рис. 2.16: Запуск программы task2.asm

3 Выводы

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