

Отчёт по лабораторной работе 7

Архитектура компьютера

Цыкунова Екатерина Михайловна НКАбд-05-24

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Изучение структуры файлы 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 lab7-task1.asm	17
2.14	Запуск программы lab7-task1.asm	17
2.15	Программа lab7-task2.asm	19
2.16	Запуск программы lab7-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)

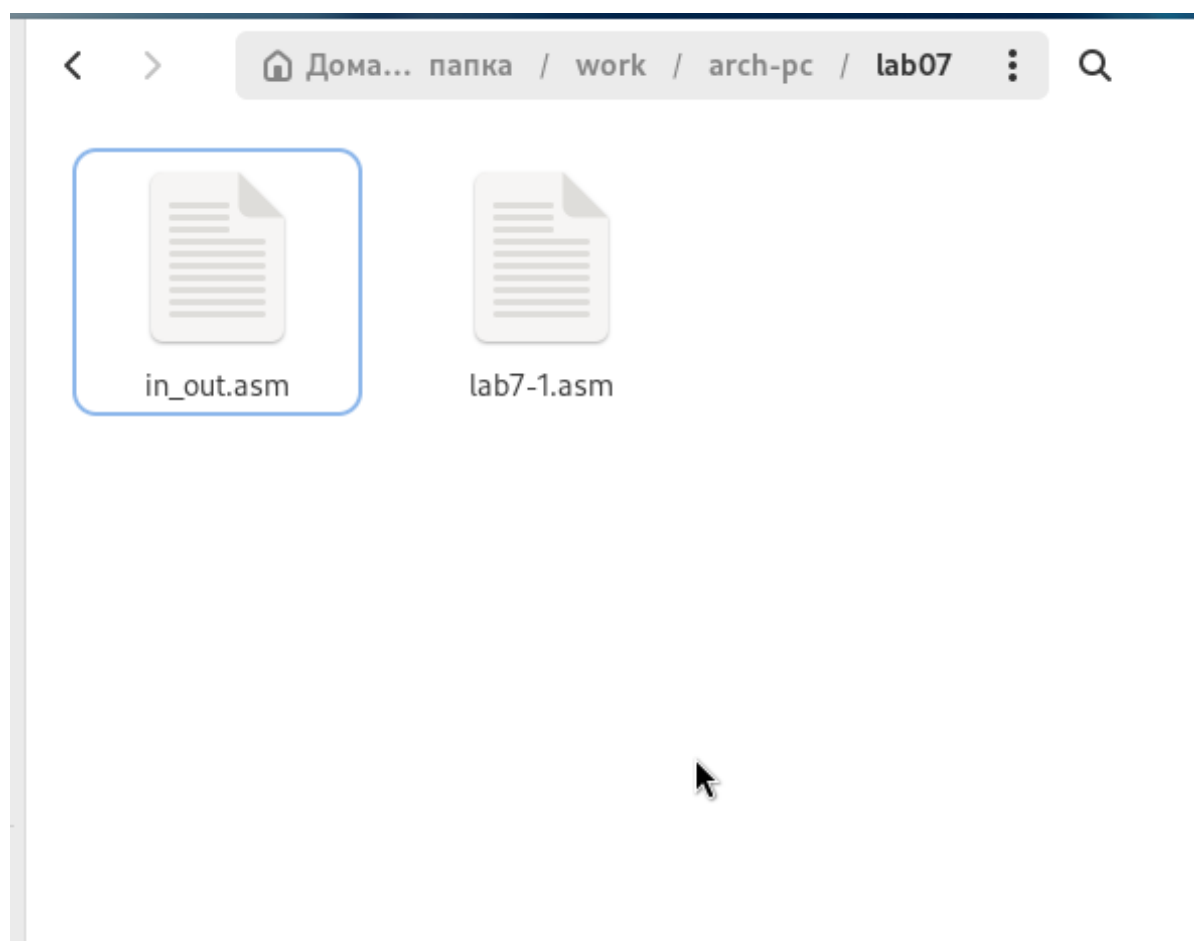


Рис. 2.1: Создан каталог

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/arch-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. 2.2: Программа `lab7-1.asm`

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

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

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

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

Рис. 2.4: Программа lab7-1.asm



```
Открыть ▾ + lab7-1.asm
~/work/arch-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
```

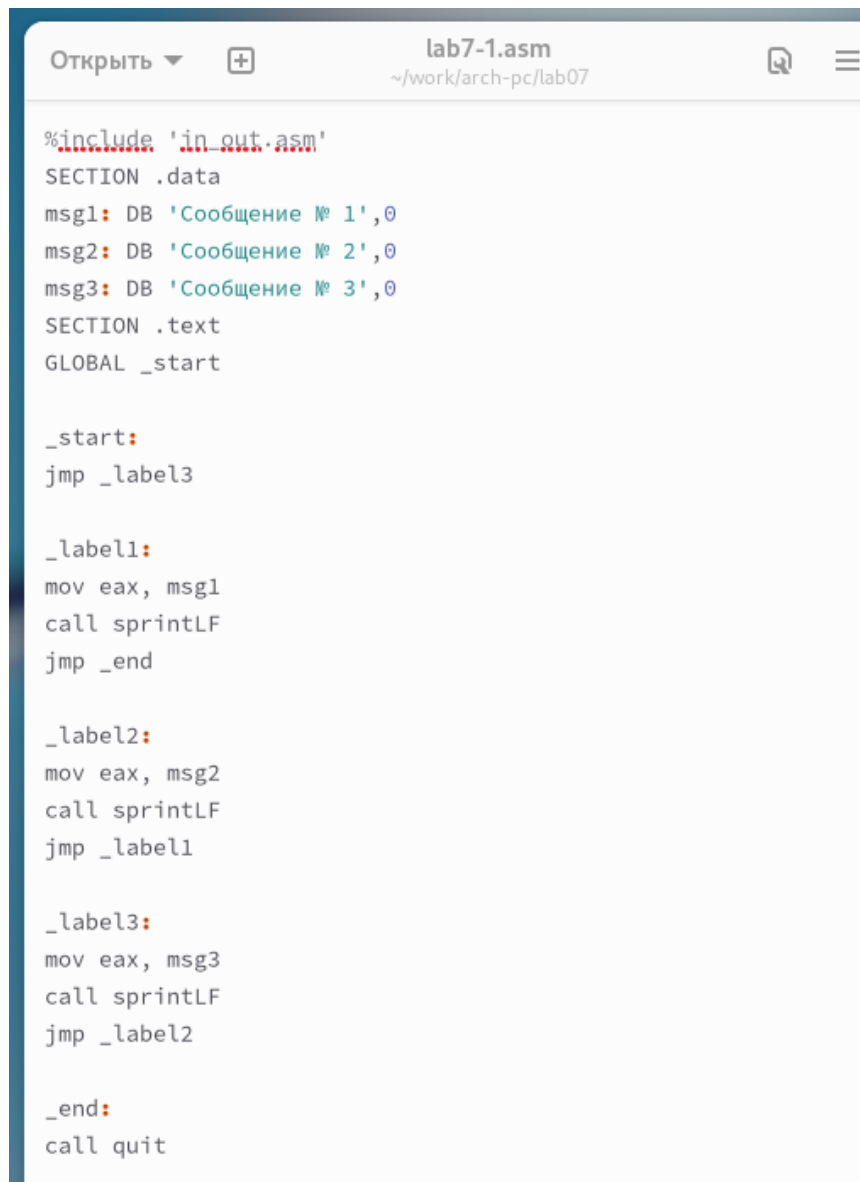
Рис. 2.5: Запуск программы lab7-1.asm

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

Сообщение № 3

Сообщение № 2

Сообщение № 1



```
Открыть + lab7-1.asm
~/.work/arch-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

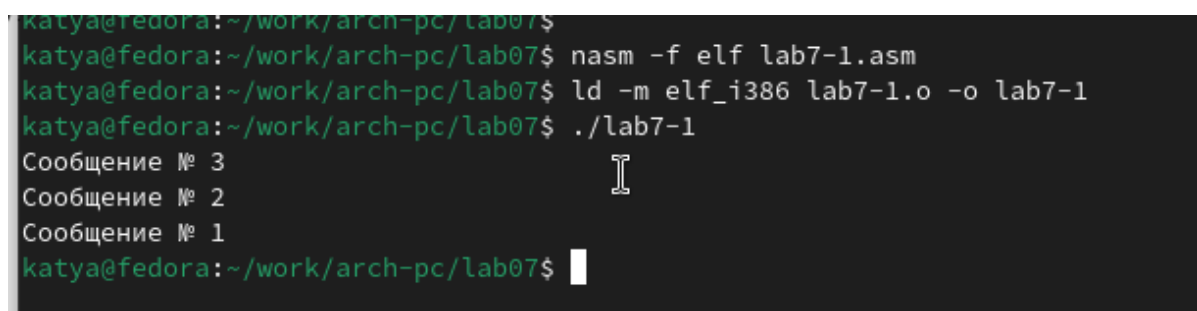
_label1:
mov eax, msg1
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF
jmp _label2

_end:
call quit
```

Рис. 2.6: Программа lab7-1.asm

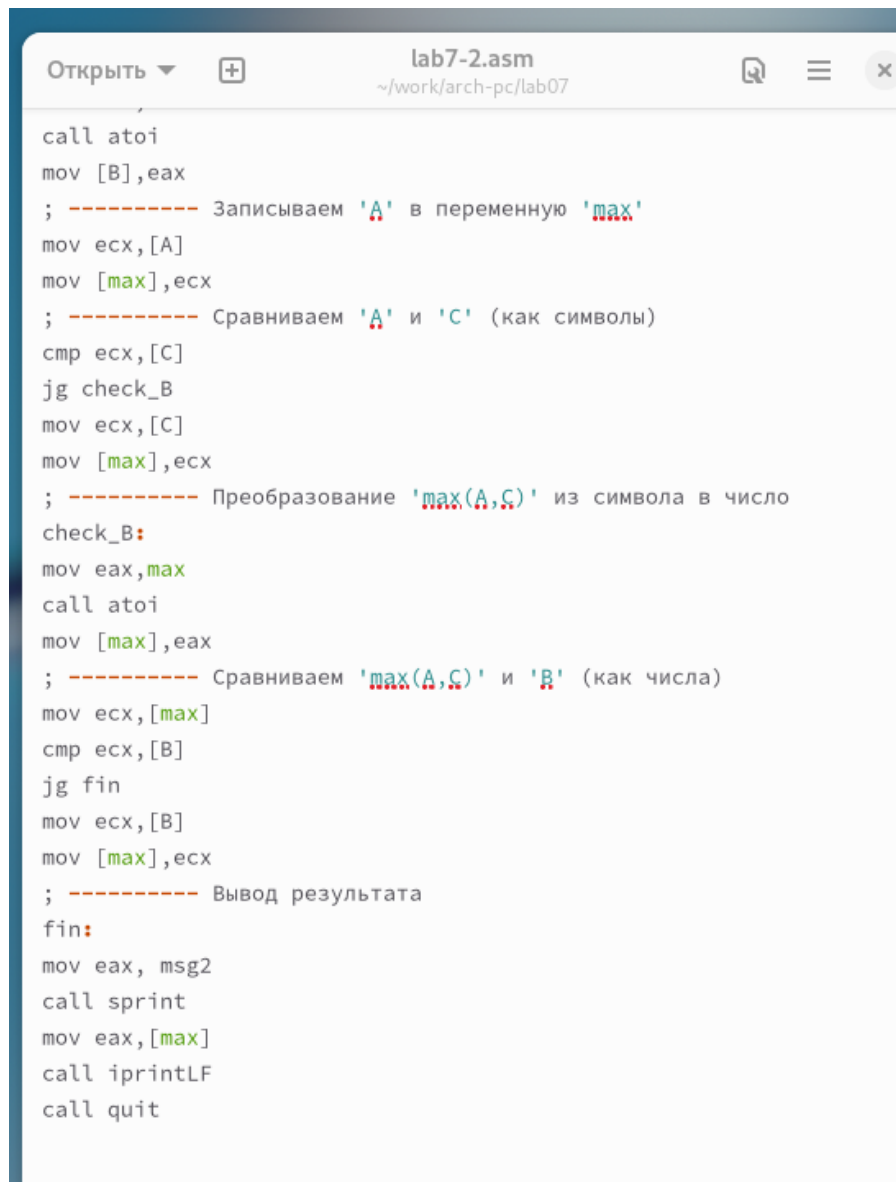


```
katya@fedora:~/work/arch-pc/lab07$
katya@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
katya@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
katya@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
katya@fedora:~/work/arch-pc/lab07$
```

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.

Создала исполняемый файл и проверила его работу для разных значений В (рис. 2.8) (рис. 2.9).



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

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

```
katya@fedora:~/work/arch-pc/lab07$  
katya@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm  
katya@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2  
katya@fedora:~/work/arch-pc/lab07$ ./lab7-2  
Введите В: 5  
Наибольшее число: 50  
katya@fedora:~/work/arch-pc/lab07$ ./lab7-2  
Введите В: 55  
Наибольшее число: 55  
katya@fedora:~/work/arch-pc/lab07$
```

Рис. 2.9: Запуск программы lab7-2.asm

2.2 Изучение структуры файлы листинга

Обычно nasm создаёт 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)

```
lab7-2.asm                                lab7-2.lst
21 00000101 B8[00000000]                   mov eax,0
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:
45 00000159 B8[13000000]                     mov eax,msg2
46 0000015E E8ACFFFFFF                       call sprint
47 00000163 A1[00000000]                     mov eax,[max]
48 00000168 E819FFFFFF                       call iprintf
49 0000016D E869FFFFFF                       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

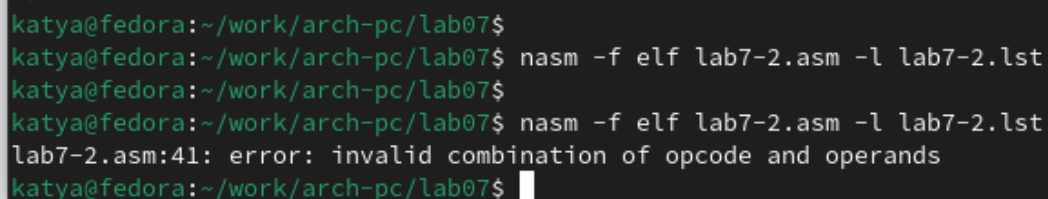
строка 212

- 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

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



```
katya@fedora:~/work/arch-pc/lab07$  
katya@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst  
katya@fedora:~/work/arch-pc/lab07$  
katya@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst  
lab7-2.asm:41: error: invalid combination of opcode and operands  
katya@fedora:~/work/arch-pc/lab07$
```

Рис. 2.11: Ошибка трансляции lab7-2

```
Открыть  + lab7-2.lst ~\work\arch-pc\lab07
21 00000101 B8[0A000000] mov eax,B
22 00000106 F891FFFFFF call atoi
23 0000010B A2[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 F862FFFFFF call atoi
36 0000013A A2[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 7F06 jg fin
41 mov ecx,
41 ***** error: invalid combination of opcode and operands
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
```

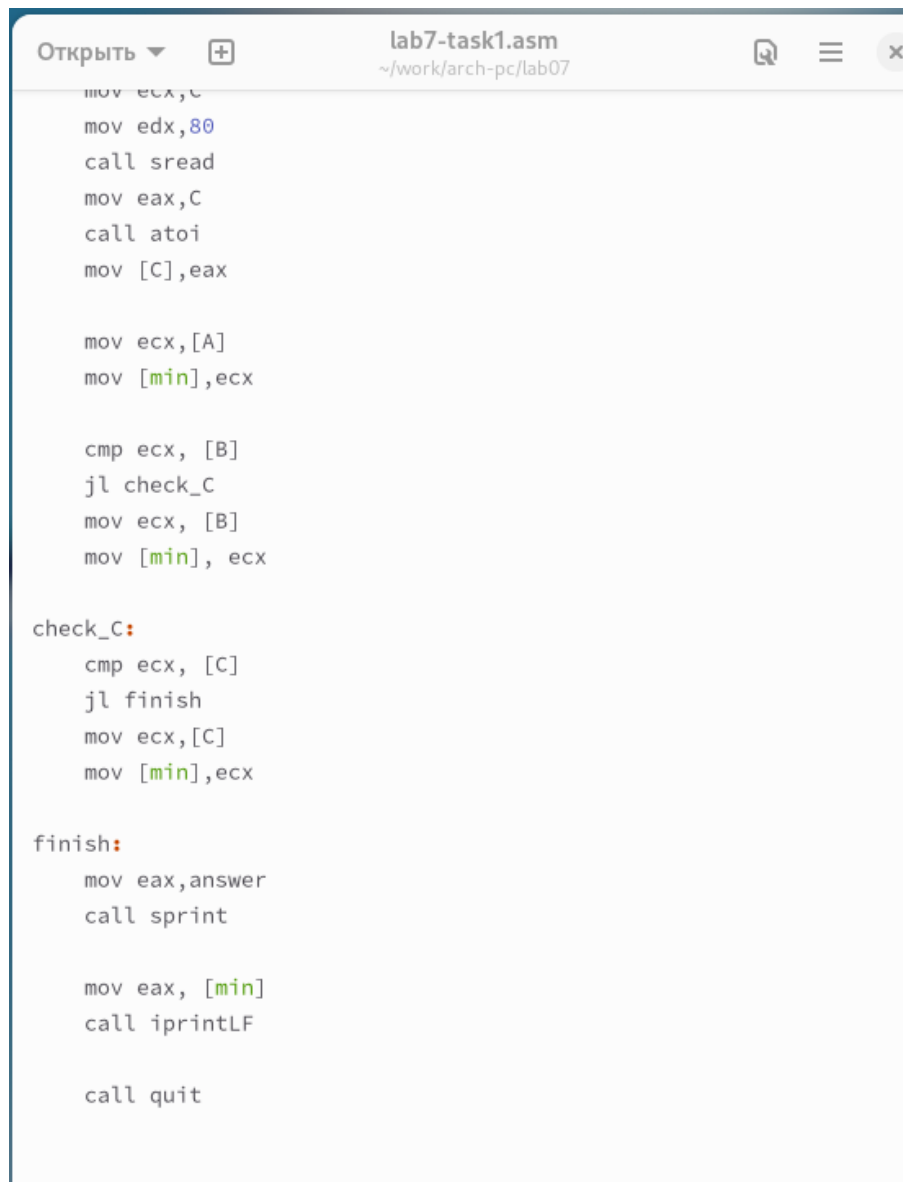
Рис. 2.12: Файл листинга с ошибкой lab7-2

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 ▾ [icon] lab7-task1.asm
~/work/arch-pc/lab07

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

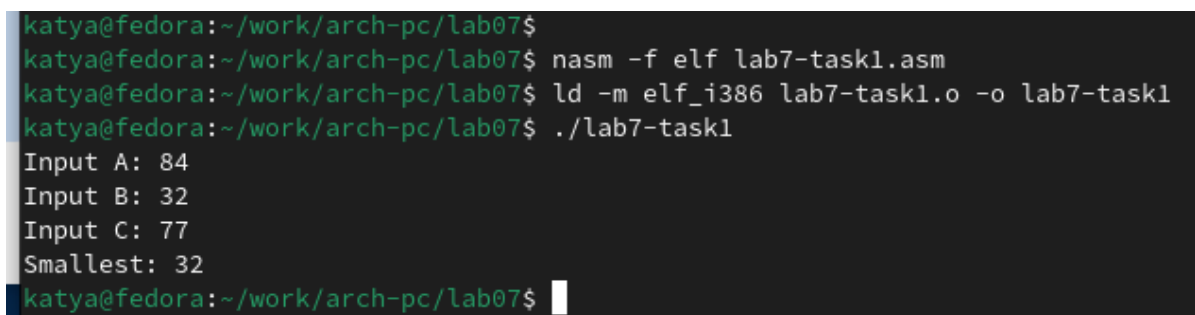
check_C:
    cmp ecx, [C]
    jl finish
    mov ecx,[C]
    mov [min],ecx

finish:
    mov eax,answer
    call sprint

    mov eax, [min]
    call iprintLF

    call quit
```

Рис. 2.13: Программа lab7-task1.asm



```
katya@fedora:~/work/arch-pc/lab07$
katya@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-task1.asm
katya@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-task1.o -o lab7-task1
katya@fedora:~/work/arch-pc/lab07$ ./lab7-task1
Input A: 84
Input B: 32
Input C: 77
Smallest: 32
katya@fedora:~/work/arch-pc/lab07$
```

Рис. 2.14: Запуск программы lab7-task1.asm

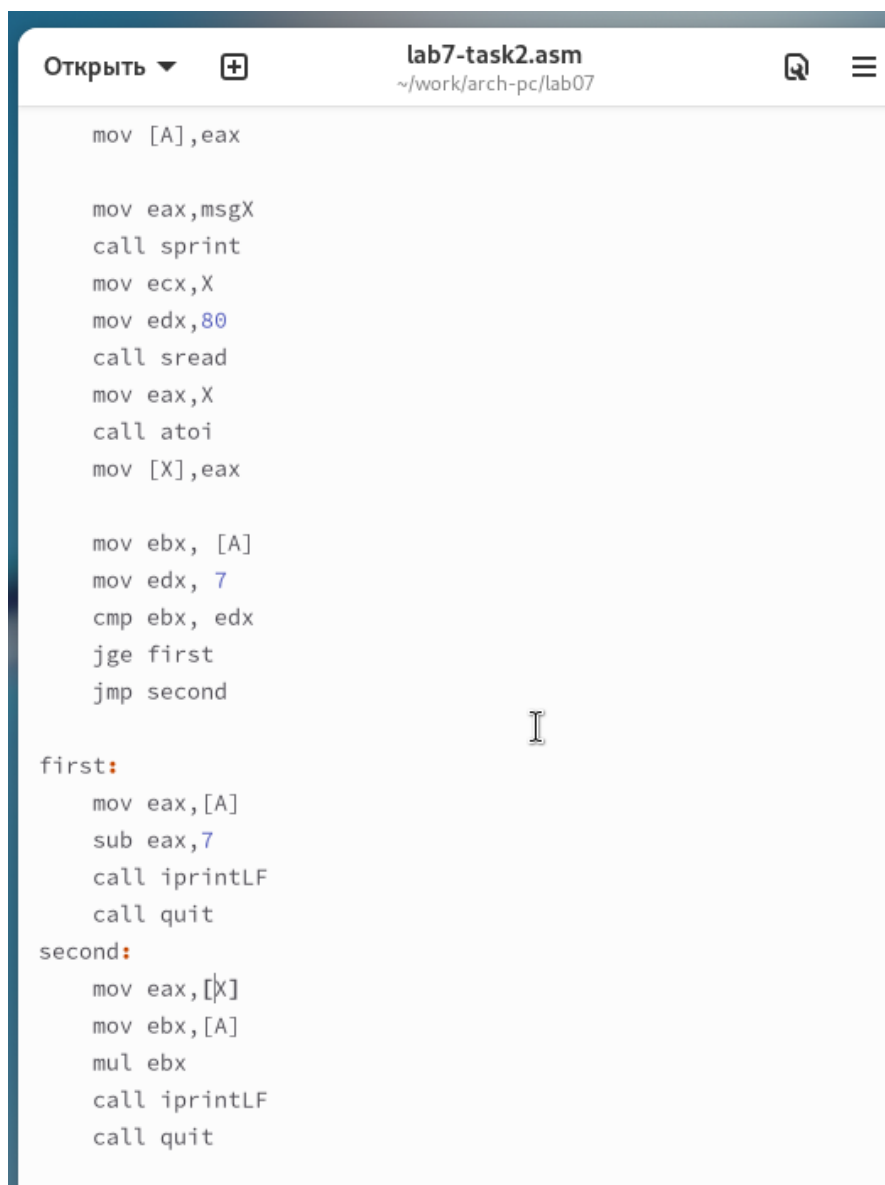
Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый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



```
Открыть ▾ + lab7-task2.asm
~/work/arch-pc/lab07

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, 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 lab7-task2.asm

```
katya@fedora:~/work/arch-pc/lab07$  
katya@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-task2.asm  
katya@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-task2.o -o lab7-task2  
katya@fedora:~/work/arch-pc/lab07$ ./lab7-task2  
Input A: 9  
Input X: 3  
2  
katya@fedora:~/work/arch-pc/lab07$ ./lab7-task2  
Input A: 4  
Input X: 6  
24  
katya@fedora:~/work/arch-pc/lab07$
```

Рис. 2.16: Запуск программы lab7-task2.asm

3 Выводы

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