

# **Отчёт по лабораторной работе №7**

**Дисциплина: архитектура компьютера**

Самойлова Софья Дмитриевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
4.1	Реализация переходов в NASM . . . . .	7
4.2	Изучение структуры файлы листинга . . . . .	12
4.3	Задание для самостоятельной работы . . . . .	13
<b>5</b>	<b>Выводы</b>	<b>19</b>

## Список иллюстраций

4.1	Создание каталога . . . . .	7
4.2	Ввод программы . . . . .	8
4.3	Результаты работы программы . . . . .	8
4.4	Редактирование программы . . . . .	9
4.5	Работа программы . . . . .	9
4.6	Редактирование программы . . . . .	10
4.7	Результат редактирования . . . . .	10
4.8	Ввод программы . . . . .	11
4.9	Результат работы программы . . . . .	11
4.10	Открытие файла листинга . . . . .	12
4.11	Ошибка . . . . .	13
4.12	Работа программы . . . . .	15
4.13	Работа программы . . . . .	18

# 1 Цель работы

Целью лабораторной работы является изучение команд условного и безусловного переходов, приобретение навыков написания программ с использованием переходов, знакомство с назначением и структурой файла листинга.

## **2 Задание**

1. Реализация переходов в NASM
2. Изучение структуры файлы листинга
3. Задание для самостоятельной работы

### 3 Теоретическое введение

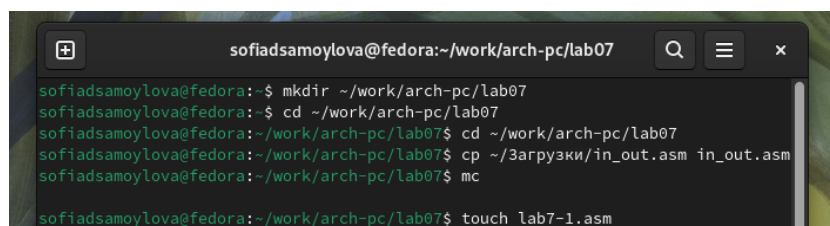
Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- *условный переход* – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.
- *безусловный переход* – выполнение передачи управления в определенную точку программы без каких-либо условий.

## 4 Выполнение лабораторной работы

### 4.1 Реализация переходов в NASM

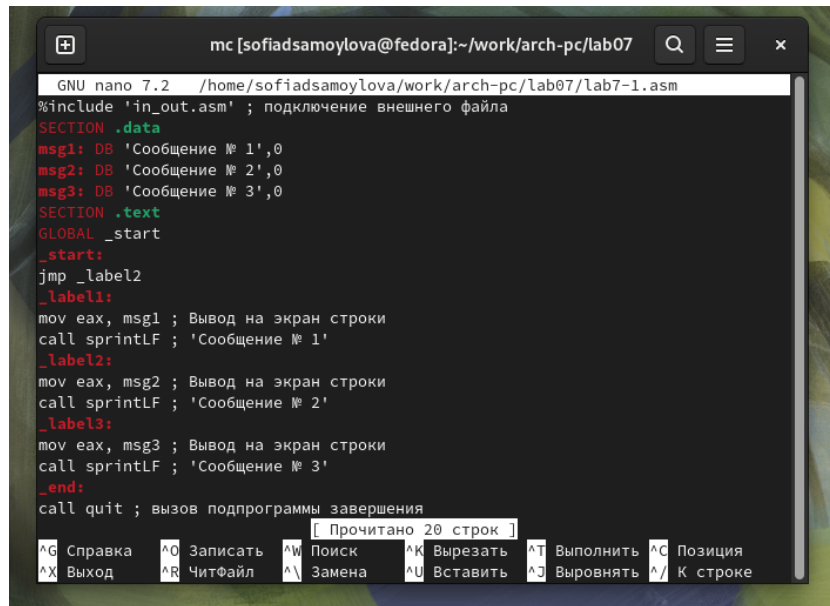
С помощью утилиты `mkdir` создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №7. Перехожу в созданный каталог с помощью утилиты `cd` и создаю файл `lab7-1.asm` (рис. 4.1). Дополнительно копирую в текущий каталог файл `in_out.asm` с помощью утилиты `cp`, т.к. он будет использоваться в других программах

A screenshot of a terminal window with a dark background. The window title is "sofiadsamoylova@fedora:~/work/arch-pc/lab07". The terminal shows a series of commands and their outputs: 

```
sofiadsamoylova@fedora:~$ mkdir ~/work/arch-pc/lab07
sofiadsamoylova@fedora:~$ cd ~/work/arch-pc/lab07
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ cd ~/work/arch-pc/lab07
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ cp ~/Загрузки/in_out.asm in_out.asm
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ mc
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ touch lab7-1.asm
```

Рис. 4.1: Создание каталога

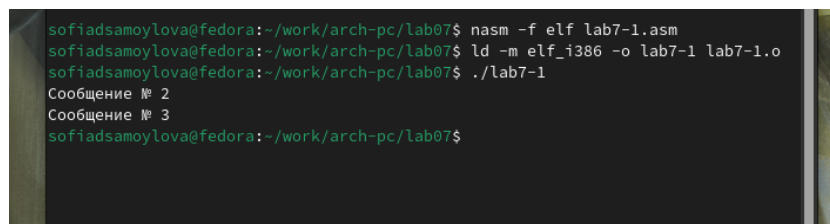
Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрю пример программы с использованием инструкции `jmp`. Ввожу в файл `lab7-1.asm` текст программы (рис. 4.2).



```
mc [sofiadsamoylova@fedora]:~/work/arch-pc/lab07
GNU nano 7.2 /home/sofiadsamoylova/work/arch-pc/lab07/lab7-1.asm
#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'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
[Прочитано 20 строк]
```

Рис. 4.2: Ввод программы

Создаю исполняемый файл и запускаю его. Результат работы данной программы следующий (рис. 4.3).



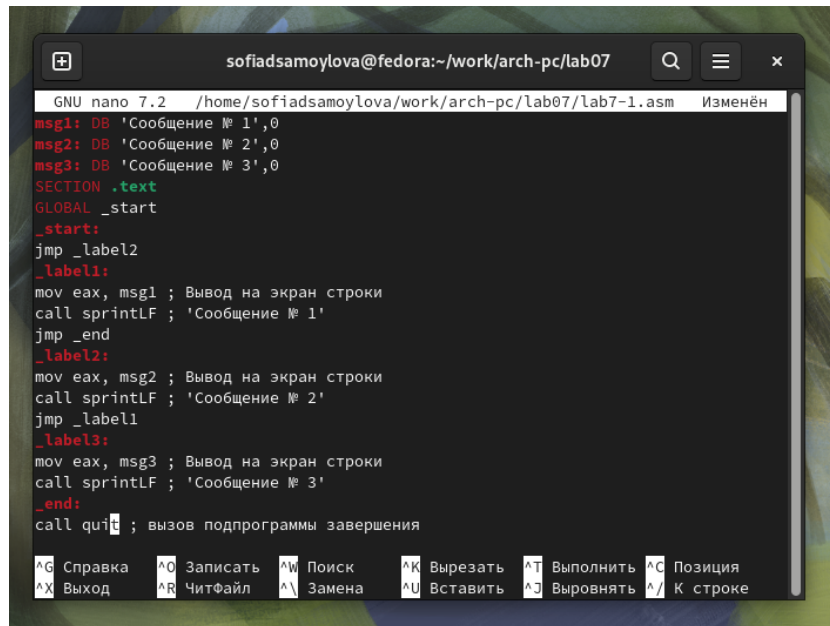
```
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
sofiadsamoylova@fedora:~/work/arch-pc/lab07$
```

Рис. 4.3: Результаты работы программы

Таким образом, использование инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`, пропустив вывод первого сообщения.

Изменяю 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`) (рис. 4.4).



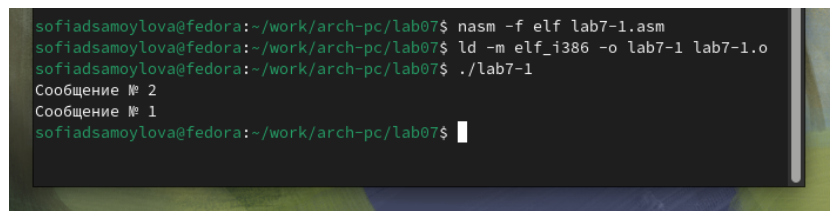


```
GNU nano 7.2 /home/sofiadsamoylova/work/arch-pc/lab07/lab7-1.asm Изменён
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
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

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^_/ К строке
```

Рис. 4.4: Редактирование программы

Создаю исполняемый файл и проверяю его работу (рис. 4.5).

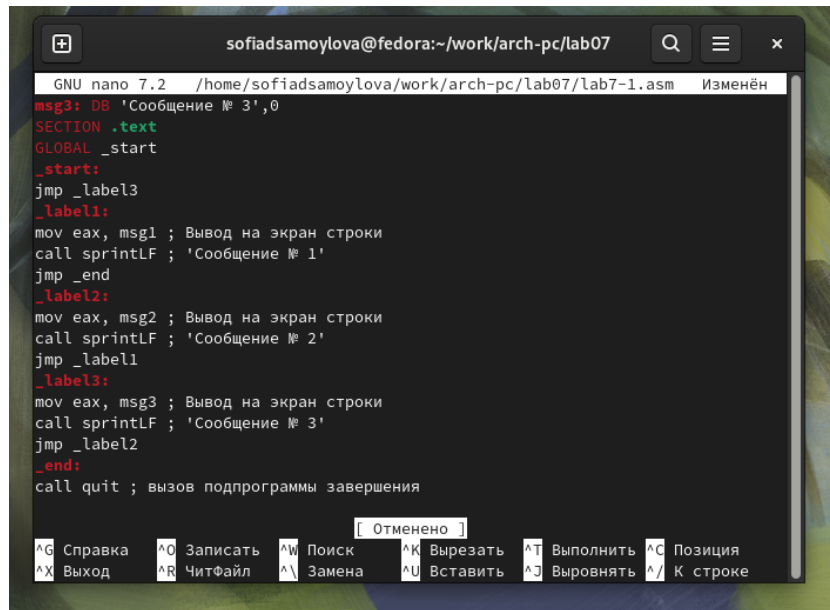


```
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
Сообщение № 3
sofiadsamoylova@fedora:~/work/arch-pc/lab07$
```

Рис. 4.5: Работа программы

Изменяю текст программы, корректируя инструкции `jmp`, чтобы вывод программы был следующим (рис. 4.6):

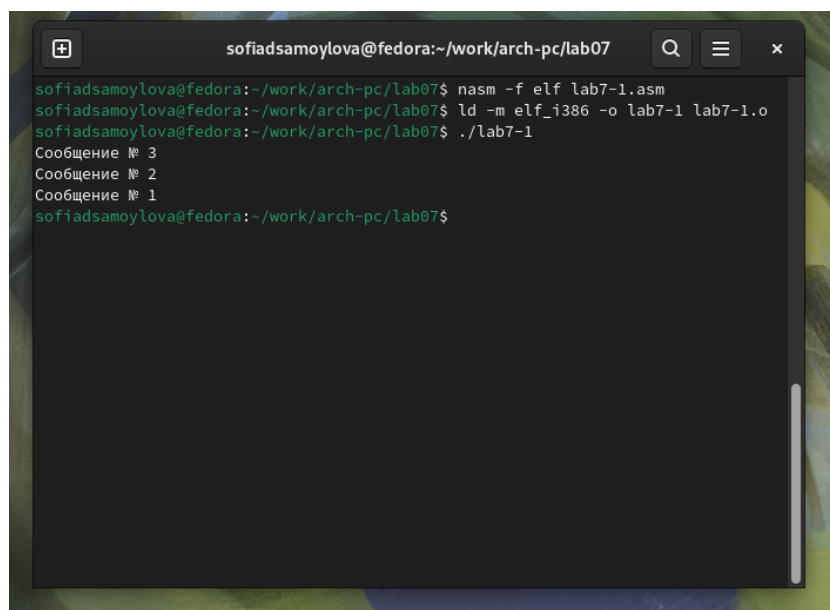
```
user@dk4n31:~$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
user@dk4n31:~$
```



```
GNU nano 7.2 /home/sofiadsamoylova/work/arch-pc/lab07/lab7-1.asm Изменён
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label3
_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. 4.6: Редактирование программы

Создаю исполняемый файл и проверяю его работу (рис. 4.7).



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

Рис. 4.7: Результат редактирования

Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07и ввожу код программы в lab7-2.asm (рис. 4.8).

```

GNU nano 7.2 /home/sofiadsamoylova/work/arch-pc/lab07/lab7-2.asm
#include 'in_out.asm'
section .data
msg1 db 'Введите В: ',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 В: '
mov eax,msg1
call sprint
; ----- Ввод 'В'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'В' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'В'
; ----- Записываем 'А' в переменную 'max'
mov ecx,[A] ; 'ecx = А'
mov [max],ecx ; 'max = А'
; ----- Сравниваем 'А' и 'С' (как символы)
cmp ecx,[C] ; Сравниваем 'А' и 'С'
jg check_B ; если 'А>С', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = С'

```

Рис. 4.8: Ввод программы

Создаю исполняемый файл и проверяю его работу для разных значений В (рис. 4.9).

```

sofiadsamoylova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 7
Наибольшее число: 50
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 26
Наибольшее число: 50
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 67
Наибольшее число: 67
sofiadsamoylova@fedora:~/work/arch-pc/lab07$

```

Рис. 4.9: Результат работы программы

В данном примере переменные А и С сравниваются как символы, а переменная В и максимум из А и С как числа (для этого используется функция atoi преобразо-

вания символа в число). Это сделано для демонстрации того, как сравниваются данные.

## 4.2 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` и открываю его с помощью `mcedit`(рис. 4.10).

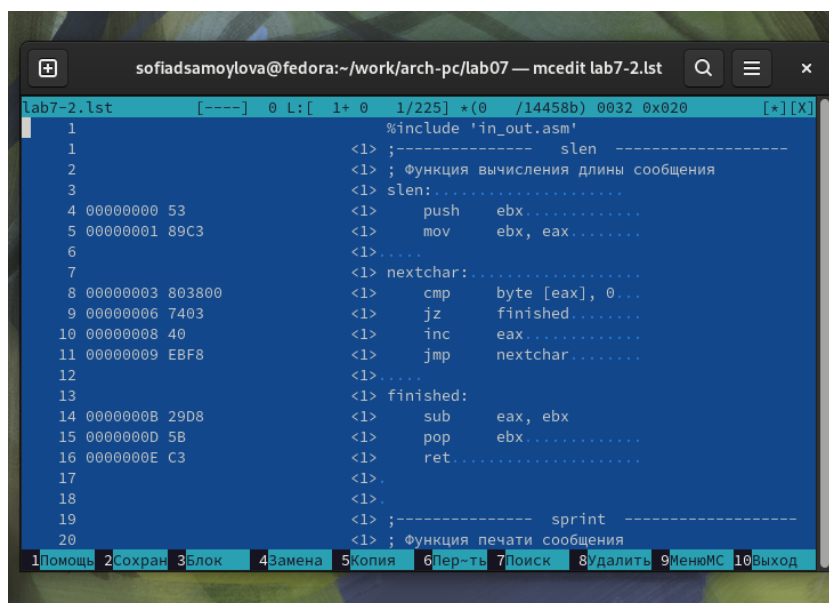


Рис. 4.10: Открытие файла листинга

Первое значение в файле листинга - номер строки, и он может вовсе не совпадать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем непосредственно идет сам машинный код, а заключает строку исходный текст программы с комментариями.

Попробую изменить следующую строку:

`cmp ebx, [C] ; Сравниваем 'A' и 'C'`

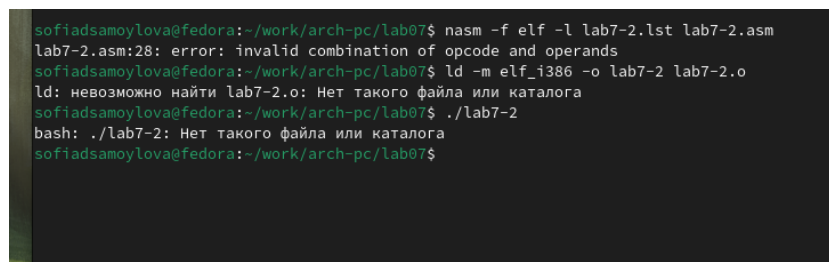
Удалив один из операндов, изменяю её на:

`cmp ехх ; Сравниваем 'А' и 'С'`

Последствия изменений

1. Синтаксические ошибки: Удаление одного из операндов в инструкции сравнения `cmp` приведет к синтаксической ошибке, так как команда `cmp` требует два операнда (регистры или память).
2. Выходные файлы:
  - Если вы попытаетесь собрать программу с этой ошибкой, компилятор (или ассемблер) не сможет создать исполняемый файл, так как код будет содержать ошибку.
  - В случае успешной сборки (если исправить ошибку), будут созданы выходные файлы, такие как `.o` (объектный файл) и исполняемый файл (например, `a.out` или с другим именем, если вы укажете его).

Соответственно, не получится создать и скомпилировать файлы (рис. 4.11).



```
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:28: error: invalid combination of opcode and operands
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
ld: невозможно найти lab7-2.o: Нет такого файла или каталога
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ ./lab7-2
bash: ./lab7-2: Нет такого файла или каталога
sofiadsamoylova@fedora:~/work/arch-pc/lab07$
```

Рис. 4.11: Ошибка

## 4.3 Задание для самостоятельной работы

Вариант предыдущей лабораторной работы 17.

*Задание 1* Код программы:

```

%include 'in_out.asm'

section .data
msg1 db 'Введите B: ', 0
msg2 db 'Наименьшее число: ', 0
A dd 26          ; Значение A
B dd 12          ; Значение B
C dd 68          ; Значение C

section .bss
min resd 1       ; Переменная для хранения наименьшего значения

section .text
global _start
_start:
    ; ----- Начинаем с предположения, что A - наименьшее
    mov eax, [A]
    mov [min], eax    ; min = A

    ; ----- Сравниваем min и B
    mov eax, [B]
    cmp eax, [min]
    jnl update_min    ; Если B < min, обновляем min

    ; ----- Сравниваем min и C
    mov eax, [C]
    cmp eax, [min]
    jnl update_min    ; Если C < min, обновляем min

```

```
jmp print_result ; Переход к выводу результата
```

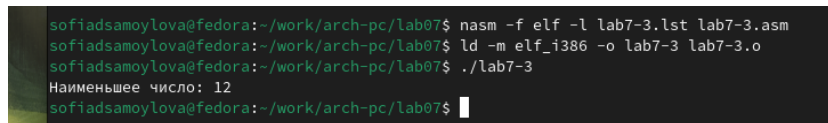
update\_min:

```
mov eax, [B] ; Если один из предыдущих сравнений был истинным, обновляем min
mov [min], eax
```

print\_result:

```
; ----- Вывод результата
mov eax, msg2
call sprint ; Вывод сообщения 'Наименьшее число: '
mov eax, [min]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход
```

Результат работы программы (рис. 4.12).



```
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-3.lst lab7-3.asm
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ ./lab7-3
Наименьшее число: 12
sofiadsamoylova@fedora:~/work/arch-pc/lab07$
```

Рис. 4.12: Работа программы

*Задание 2* Код программы:

```
%include 'in_out.asm'
```

```
section .data
```

```
msg1 db 'Введите x: ', 0
```

```
msg2 db 'Введите a: ', 0
```

```
msg3 db 'Результат f(x): ', 0
```

```
A dd 20 ; Значение A, не используется в данной задаче
```

```
C dd 50 ; Значение C, не используется в данной задаче
```

```

section .bss
result resb 10      ; Переменная для хранения результата
x resb 10           ; Переменная для хранения введенного x
a resb 10           ; Переменная для хранения введенного a

section .text
global _start
_start:

    ; ----- Вывод сообщения 'Введите x: '
    mov eax, msg1
    call sprint

    ; ----- Ввод 'x'
    mov ecx, x
    mov edx, 10
    call sread

    ; ----- Преобразование 'x' из символа в число
    mov eax, x
    call atoi        ; Вызов подпрограммы перевода символа в число
    mov [x], eax     ; Запись преобразованного числа в 'x'

    ; ----- Вывод сообщения 'Введите a: '
    mov eax, msg2
    call sprint

    ; ----- Ввод 'a'
    mov ecx, a
    mov edx, 10

```



```

call sread

; ----- Преобразование 'a' из символа в число
mov eax, a
call atoi          ; Вызов подпрограммы перевода символа в число
mov [a], eax       ; Запись преобразованного числа в 'a'

; ----- Сравниваем 'a' с 8
mov eax, [a]
cmp eax, 8         ; Сравниваем a с 8
jnl case_a_less_8  ; Если a < 8, переходим к case_a_less_8

; Если a >= 8, вычисляем a * x
mov eax, [a]
mov ebx, [x]       ; Загружаем значение x в ebx
imul eax, ebx      ; Умножаем a на x (eax = a * x)
jmp store_result   ; Переход к сохранению результата

case_a_less_8:
; Если a < 8, вычисляем a + 8
mov eax, [a]
add eax, 8         ; eax = a + 8

store_result:
mov [result], eax  ; Сохраняем результат в переменной result

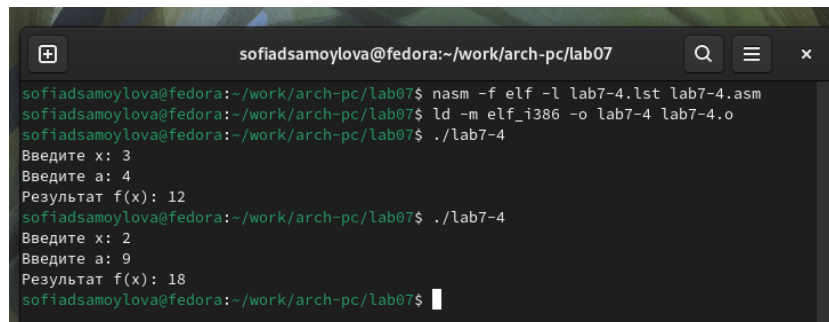
; ----- Вывод результата
mov eax, msg3
call sprint        ; Вывод сообщения 'Результат f(x): '

```

```
mov eax, [result]
call iprintLF      ; Вывод результата f(x)

call quit         ; Выход
```

Результат работы программы (рис. 4.13).



```
sofiadsamoylova@fedora:~/work/arch-pc/lab07
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-4.lst lab7-4.asm
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 3
Введите a: 4
Результат f(x): 12
sofiadsamoylova@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 2
Введите a: 9
Результат f(x): 18
sofiadsamoylova@fedora:~/work/arch-pc/lab07$
```

Рис. 4.13: Работа программы

## **5 Выводы**

При выполнении лабораторной работы я изучила команды условных и безусловных переходов, а также приобрела навыки написания программ с использованием переходов, познакомилась с назначением и структурой файлов листинга.