

Отчёта по лабораторной работе 8

**Команды безусловного и условного переходов в Nasm.
Программирование ветвлений**

Хассан Факи Абакар

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	24
	Список литературы	25

Список иллюстраций

4.1	Файл lab8-1.asm:	9
4.2	Программа lab8-1.asm:	10
4.3	Файл lab8-1.asm:	11
4.4	Программа lab8-1.asm:	12
4.5	Файл lab8-1.asm	13
4.6	Программа lab8-1.asm	14
4.7	Файл lab8-2.asm	15
4.8	Программа lab8-2.asm	16
4.9	Файл листинга lab8-2	17
4.10	ошибка трансляции lab8-2	18
4.11	файл листинга с ошибкой lab8-2	19
4.12	Файл lab8-3.asm	20
4.13	Программа lab8-3.asm	21
4.14	Файл lab8-4.asm	22
4.15	Программа lab8-4.asm	23

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 Задание

1. Изучите примеры программ.
2. Изучите файл листинга.
3. Напишите программу нахождения наименьшей из 3 целочисленных переменных a, b и c . Значения переменных выбрать из табл. 8.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу
4. 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 8.6 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 8.6.

3 Теоретическое введение

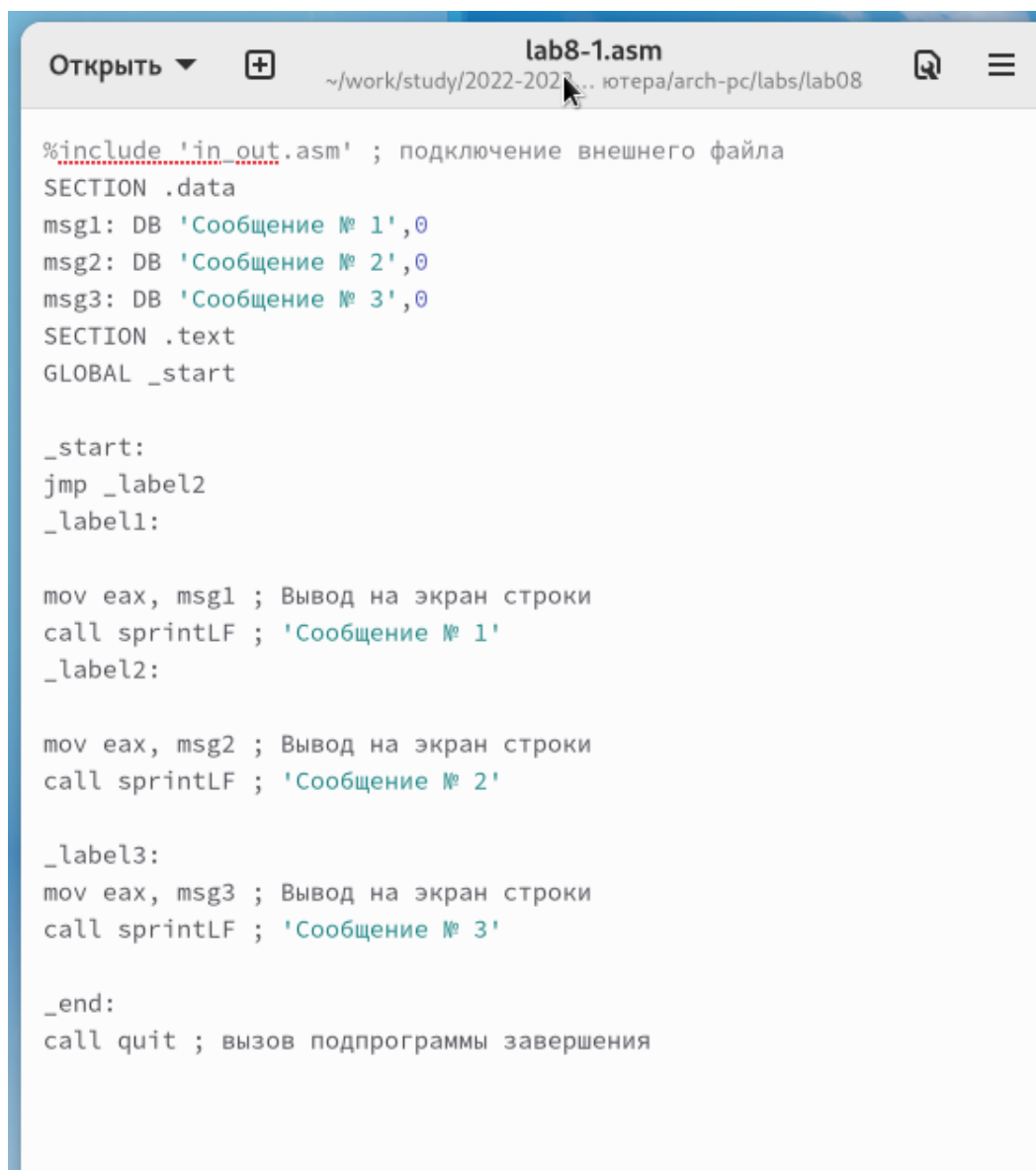
Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- 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.

Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную информацию.

4 Выполнение лабораторной работы

1. Создайте каталог для программ лабораторной работы № 8, перейдите в него и создайте файл `lab8-1.asm`
2. Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. Введите в файл `lab8-1.asm` текст программы из листинга 8.1. (рис. 4.1)



```
%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 sprintLF ; 'Сообщение № 1'
_label2:

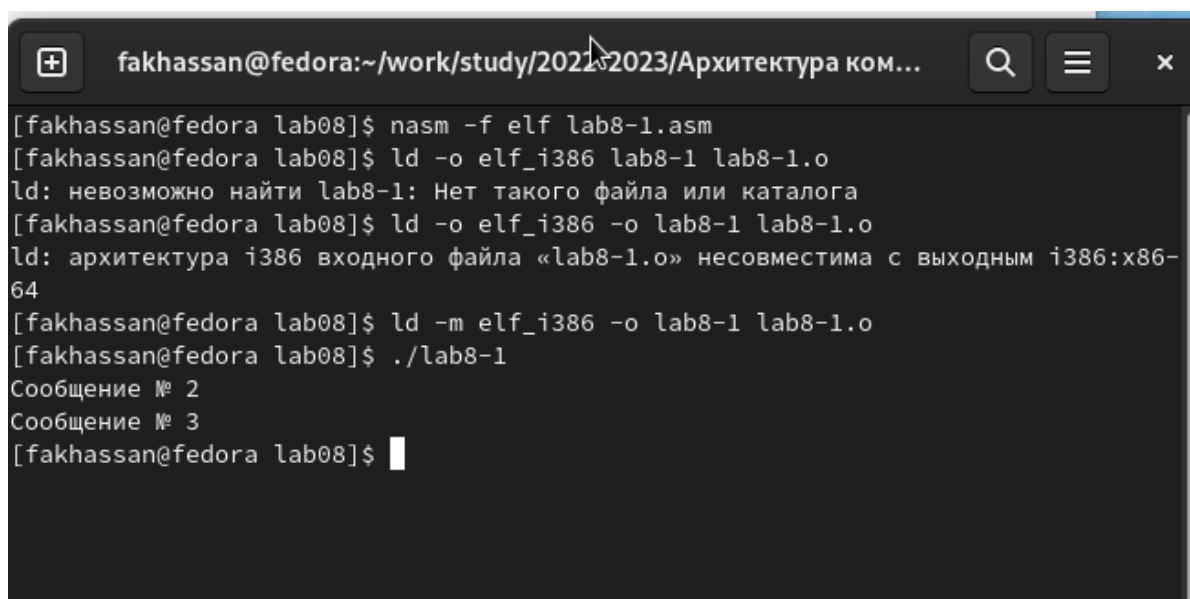
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'

_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.1: Файл lab8-1.asm:

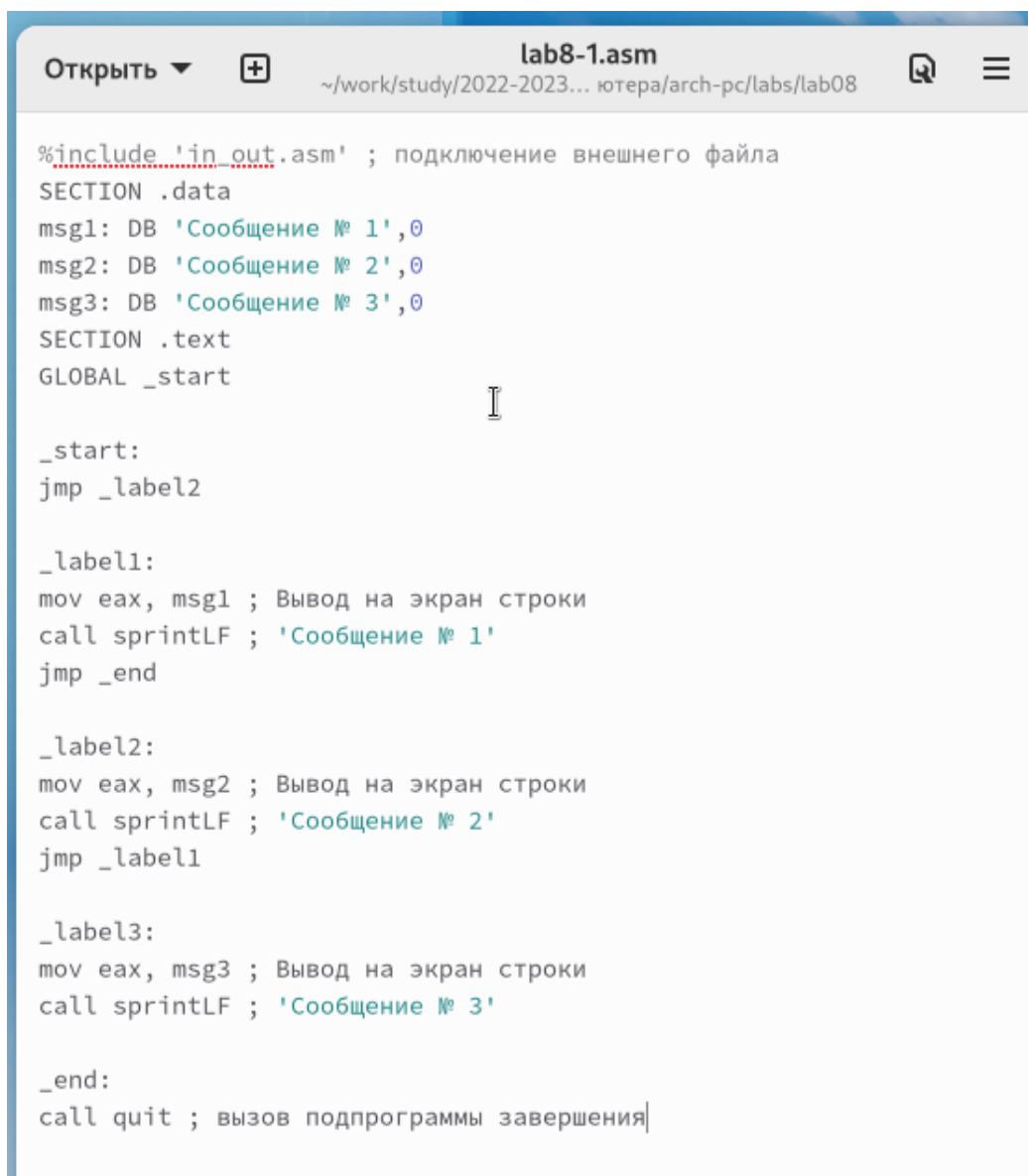
Создайте исполняемый файл и запустите его. (рис. 4.2)

A terminal window with a dark background and light text. The window title is 'fakhassan@fedora:~/work/study/2022-2023/Архитектура ком...'. The terminal shows the following commands and output:

```
[fakhassan@fedora lab08]$ nasm -f elf lab8-1.asm
[fakhassan@fedora lab08]$ ld -o elf_i386 lab8-1 lab8-1.o
ld: невозможно найти lab8-1: Нет такого файла или каталога
[fakhassan@fedora lab08]$ ld -o elf_i386 -o lab8-1 lab8-1.o
ld: архитектура i386 входного файла «lab8-1.o» несовместима с выходным i386:x86-64
[fakhassan@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[fakhassan@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[fakhassan@fedora lab08]$
```

Рис. 4.2: Программа lab8-1.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`). Измените текст программы в соответствии с листингом 8.2. (рис. 4.3, 4.4)



```
lab8-1.asm
~/work/study/2022-2023... ютеpa/arch-pc/labs/lab08

%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

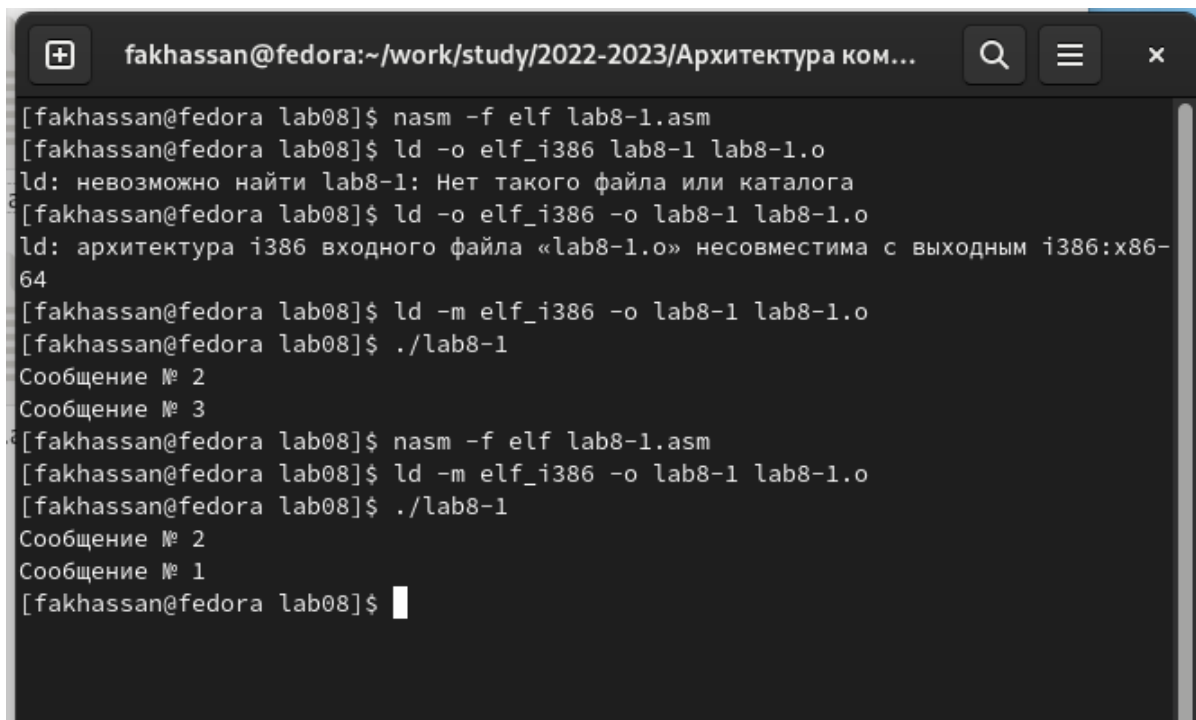
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end

_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
```

Рис. 4.3: Файл lab8-1.asm:

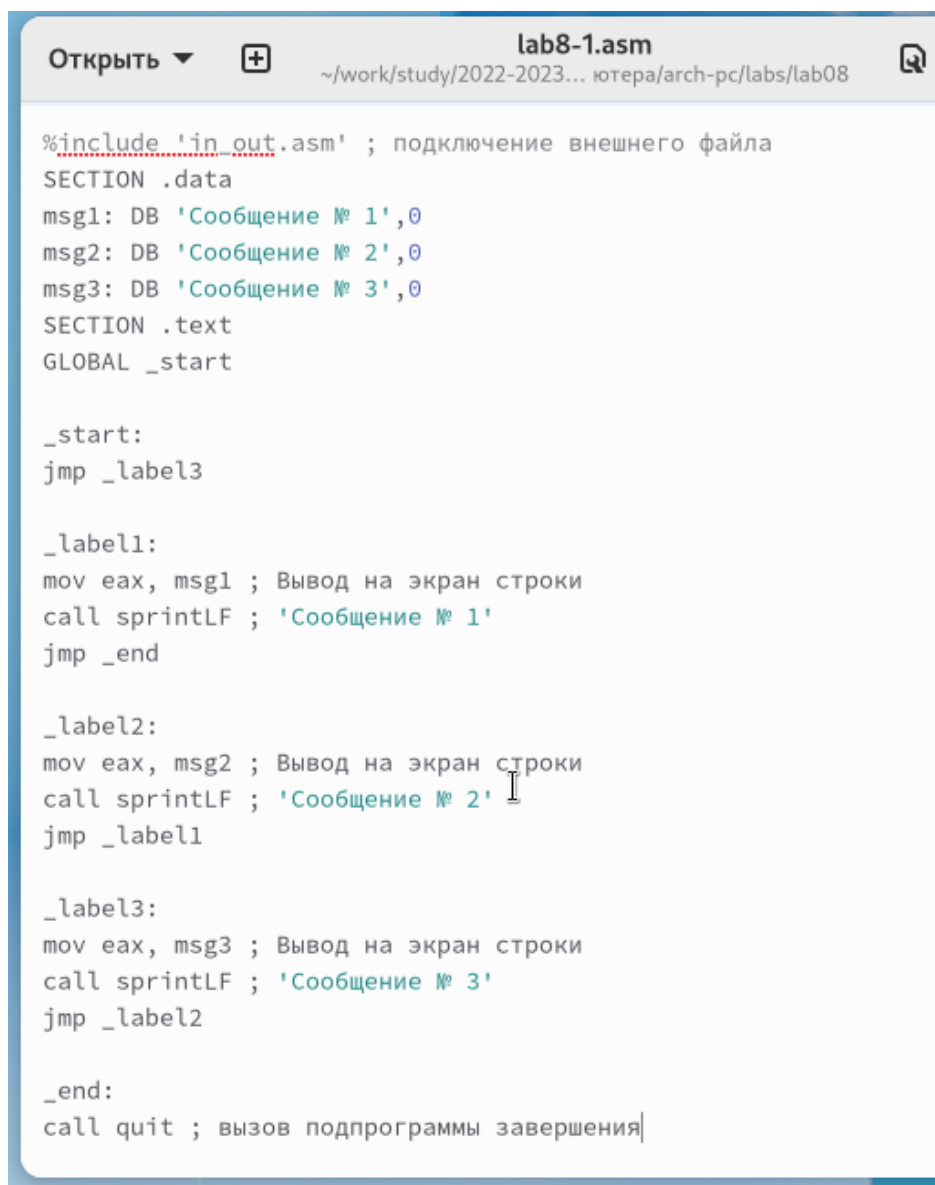


```
fakhassan@fedora:~/work/study/2022-2023/Архитектура ком...
[fakhassan@fedora lab08]$ nasm -f elf lab8-1.asm
[fakhassan@fedora lab08]$ ld -o elf_i386 lab8-1 lab8-1.o
ld: невозможно найти lab8-1: Нет такого файла или каталога
[fakhassan@fedora lab08]$ ld -o elf_i386 -o lab8-1 lab8-1.o
ld: архитектура i386 входного файла «lab8-1.o» несовместима с выходным i386:x86-64
[fakhassan@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[fakhassan@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[fakhassan@fedora lab08]$ nasm -f elf lab8-1.asm
[fakhassan@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[fakhassan@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 1
[fakhassan@fedora lab08]$
```

Рис. 4.4: Программа lab8-1.asm:

Измените текст программы добавив или изменив инструкции `jmp`, чтобы вывод программы был следующим (рис. 4.5, 4.6):

Сообщение № 3
Сообщение № 2
Сообщение № 1



```
%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 _label3

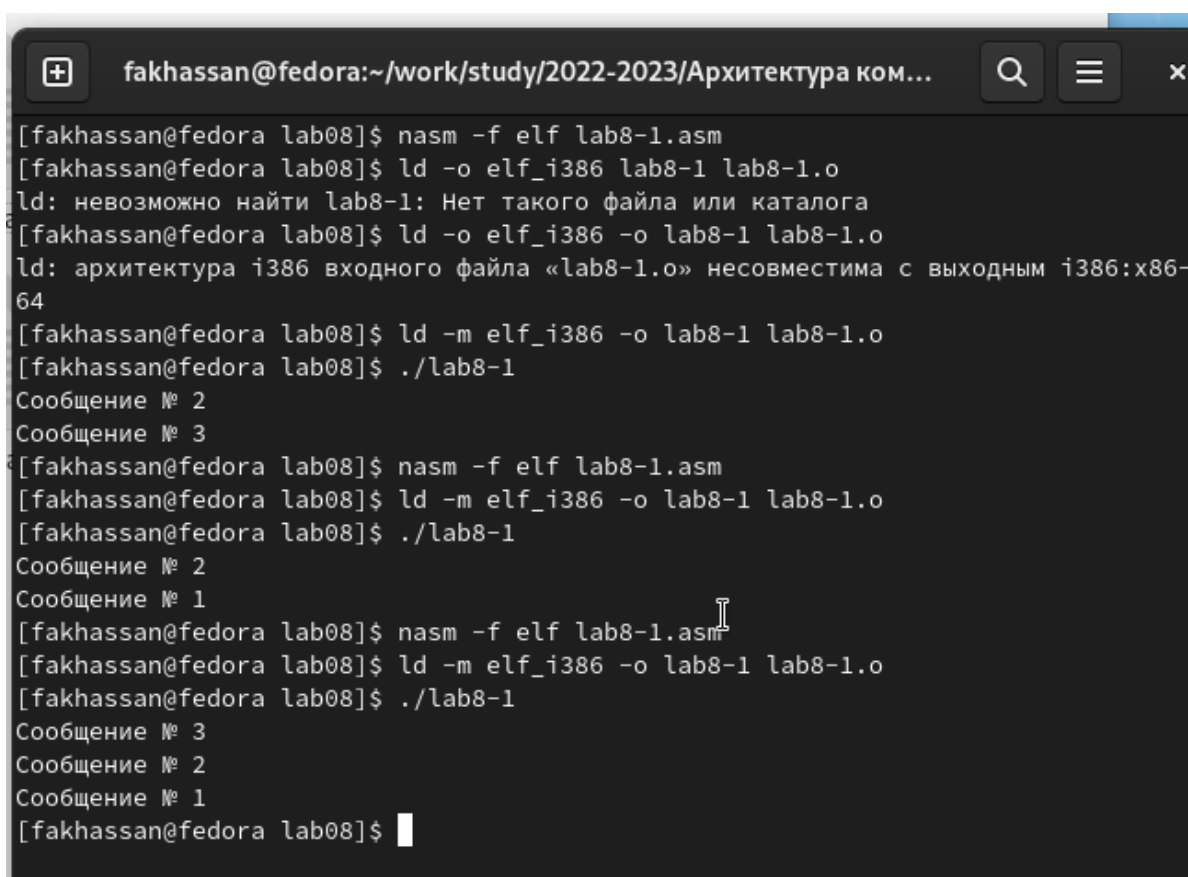
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end

_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
```

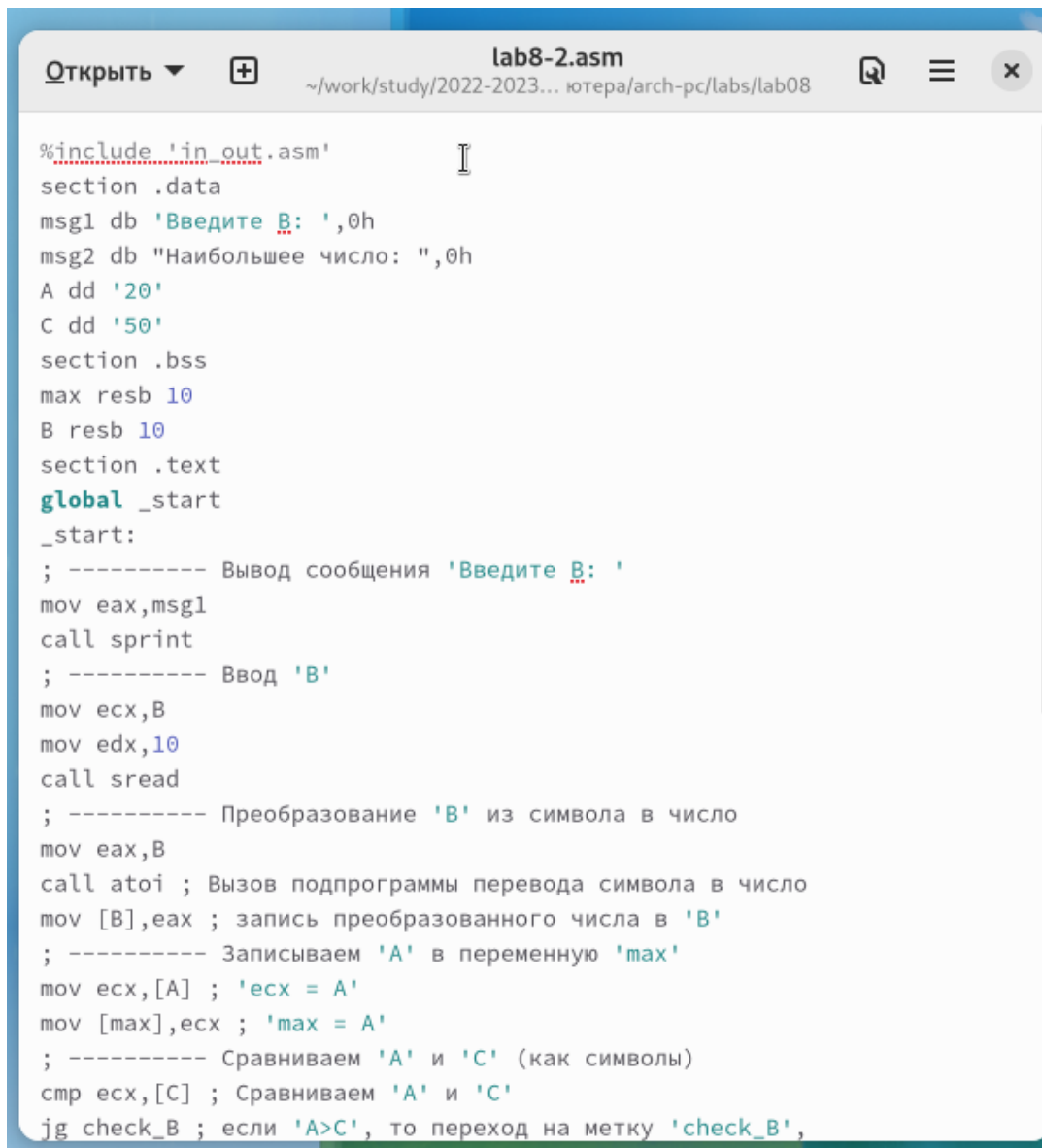
Рис. 4.5: Файл lab8-1.asm



```
fakhassan@fedora:~/work/study/2022-2023/Архитектура ком...
[fakhassan@fedora lab08]$ nasm -f elf lab8-1.asm
[fakhassan@fedora lab08]$ ld -o elf_i386 lab8-1 lab8-1.o
ld: невозможно найти lab8-1: Нет такого файла или каталога
[fakhassan@fedora lab08]$ ld -o elf_i386 -o lab8-1 lab8-1.o
ld: архитектура i386 входного файла «lab8-1.o» несовместима с выходным i386:x86-64
[fakhassan@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[fakhassan@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[fakhassan@fedora lab08]$ nasm -f elf lab8-1.asm
[fakhassan@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[fakhassan@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 1
[fakhassan@fedora lab08]$ nasm -f elf lab8-1.asm
[fakhassan@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[fakhassan@fedora lab08]$ ./lab8-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[fakhassan@fedora lab08]$
```

Рис. 4.6: Программа lab8-1.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. Создайте исполняемый файл и проверьте его работу для разных значений В. (рис. 4.7, 4.8)



```
%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',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 B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
```

Рис. 4.7: Файл lab8-2.asm

```
Сообщение № 1
[fakhassan@fedora lab08]$
[fakhassan@fedora lab08]$
[fakhassan@fedora lab08]$ nasm -f elf lab8-2.asm
[fakhassan@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[fakhassan@fedora lab08]$ ./lab8-2
Введите B: 100
Наибольшее число: 100
[fakhassan@fedora lab08]$ ./lab8-2
Введите B: 10
Наибольшее число: 50
[fakhassan@fedora lab08]$
```

Рис. 4.8: Программа lab8-2.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. Создайте файл листинга для программы из файла `lab8-2.asm` (рис. 4.9)


```
lab8-1.lst
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab08

lab8-2.asm
lab8-1.lst

127 <1> .atoi:
128 0000009C 53 <1> push ebx
129 0000009D 51 <1> push ecx
130 0000009E 52 <1> push edx
131 0000009F 56 <1> push esi
132 000000A0 89C6 <1> mov esi, eax
133 000000A2 B800000000 <1> mov eax, 0
134 000000A7 B900000000 <1> mov ecx, 0
135 <1>
136 <1> .multiplyLoop:
137 000000AC 31DB <1> xor ebx, ebx
138 000000AE 8A1C0E <1> mov bl, [esi+ecx]
139 000000B1 80FB30 <1> cmp bl, 48
140 000000B4 7C14 <1> jl .finished
141 000000B6 80FB39 <1> cmp bl, 57
142 000000B9 7F0F <1> jg .finished
143 <1>
144 000000BB 80EB30 <1> sub bl, 48
145 000000BE 01D8 <1> add eax, ebx
146 000000C0 BB0A000000 <1> mov ebx, 10
147 000000C5 F7E3 <1> mul ebx
148 000000C7 41 <1> inc ecx
149 000000C8 EBE2 <1> jmp .multiplyLoop
150 <1>
151 <1> .finished:
152 000000CA 83F900 <1> cmp ecx, 0
153 000000CD 7407 <1> je .restore
154 000000CF 8B0A000000 <1> mov ebx, 10
```

Рис. 4.9: Файл листинга lab8-2

Внимательно ознакомиться с его форматом и содержимым. Подробно объяснить содержимое трёх строк файла листинга по выбору.

строка 144

- 144 - номер строки
- 000000BB - адрес
- 80EB30 - машинный код
- sub bl, 48 - код программы

строка 145

- 145 - номер строки
- 000000BE - адрес
- 01D8 - машинный код
- add eax, ebx - код программы

строка 146

- 146 - номер строки
- 000000C0 - адрес
- BB0A000000 - машинный код
- mov ebx, 10 - код программы

Откройте файл с программой lab8-2.asm и в любой инструкции с двумя операндами удалить один операнд. Выполните трансляцию с получением файла листинга (рис. 4.10,4.11)

```
Наибольшее число: 50
[fakhassan@fedora lab08]$ nasm -f elf lab8-2.asm -l lab8-1.lst
[fakhassan@fedora lab08]$
[fakhassan@fedora lab08]$
[fakhassan@fedora lab08]$ nasm -f elf lab8-2.asm -l lab8-2.lst
lab8-2.asm:14: error: invalid combination of opcode and operands
[fakhassan@fedora lab08]$
[fakhassan@fedora lab08]$
[fakhassan@fedora lab08]$
```

Рис. 4.10: ошибка трансляции lab8-2

```

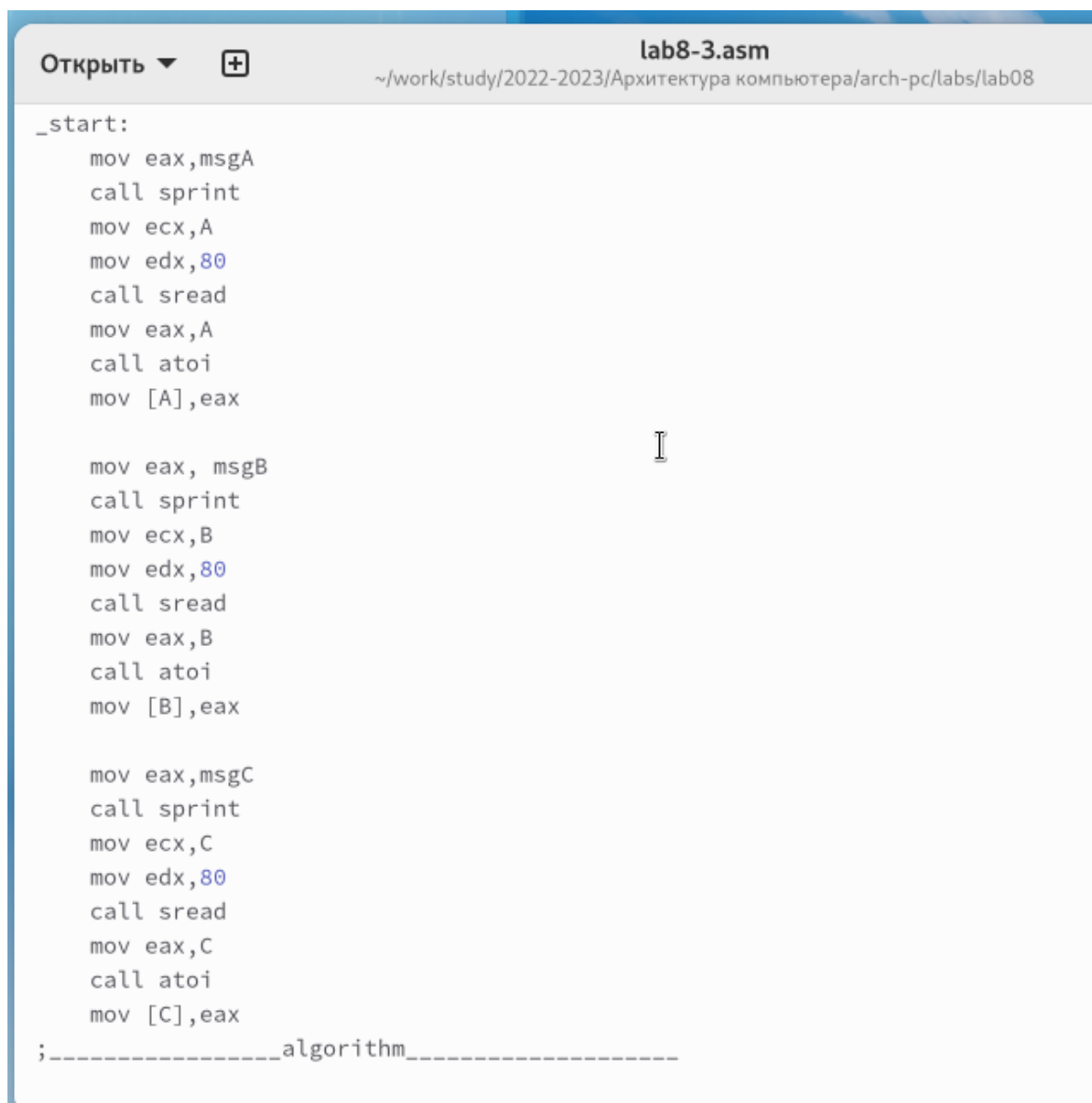
lab8-2.asm                                     lab8-2.lst
9 0000000A <res Ah>                             B resb 10
10                                         section .text
11                                         global _start
12                                         _start:
13                                         ; ----- Вывод сообщения 'Введите B: '
14                                         mov eax,
14 *****                                     error: invalid combination of opcode and operands
15 000000E8 E822FFFFFF                           call sprint
16                                         ; ----- Ввод 'B'
17 000000ED B9[0A000000]                         mov ecx,B
18 000000F2 BA0A000000                         mov edx,10
19 000000F7 E847FFFFFF                           call sread
20                                         ; ----- Преобразование 'B' из символа в число
21 000000FC B8[0A000000]                         mov eax,B
22 00000101 E896FFFFFF                           call atoi ; Вызов подпрограммы перевода символа в число
23 00000106 A3[0A000000]                         mov [B],eax ; запись преобразованного числа в 'B'
24                                         ; ----- Записываем 'A' в переменную 'max'
25 0000010B 8B0D[35000000]                       mov ecx,[A] ; 'ecx = A'
26 00000111 890D[00000000]                       mov [max],ecx ; 'max = A'
27                                         ; ----- Сравниваем 'A' и 'C' (как символы)
28 00000117 3B0D[39000000]                       cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 0000011D 7F0C                                   jg check_B ; если 'A>C', то переход на метку 'check_B',
30 0000011F 8B0D[39000000]                       mov ecx,[C] ; иначе 'ecx = C'
31 00000125 890D[00000000]                       mov [max],ecx ; 'max = C'
32                                         ; ----- Преобразование 'max(A,C)' из символа в число
33                                         check_B:
34 0000012B B8[00000000]                         mov eax,max

```

Рис. 4.11: файл листинга с ошибкой lab8-2

5. Напишите программу нахождения наименьшей из 3 целочисленных переменных a,b и c. Значения переменных выбрать из табл. 8.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу (рис. 4.12,4.13)

для варианта 10 - 41,62,35



```
Открыть ▾ + lab8-3.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab08

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

    mov eax, msgB
    call sprint
    mov ecx,B
    mov edx,80
    call sread
    mov eax,B
    call atoi
    mov [B],eax

    mov eax,msgC
    call sprint
    mov ecx,C
    mov edx,80
    call sread
    mov eax,C
    call atoi
    mov [C],eax

;-----algorithm-----
```

Рис. 4.12: Файл lab8-3.asm

```
[fakhassan@fedora lab08]$  
[fakhassan@fedora lab08]$  
[fakhassan@fedora lab08]$ nasm -f elf lab8-3.asm  
[fakhassan@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o  
[fakhassan@fedora lab08]$ ./lab8-3  
Input A: 41  
Input B: 62  
Input C: 35  
Smallest: 35  
[fakhassan@fedora lab08]$
```

Рис. 4.13: Программа lab8-3.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 8.6 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 8.6. (рис. 4.14,4.15)

для варианта 10

$$\begin{cases} x - 2, x > 2 \\ 3a, x \leq 2 \end{cases}$$

```
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

;-----algorithm-----

mov ebx, [X]
cmp ebx, 2
ja first
jmp second

first:
mov eax,[X]
sub eax,2
call iprintLF
call quit
second:
mov eax, [A]
mov ebx,3
```

Рис. 4.14: Файл lab8-4.asm

```
[fakhassan@fedora lab08]$  
[fakhassan@fedora lab08]$  
[fakhassan@fedora lab08]$ nasm -f elf lab8-4.asm  
[fakhassan@fedora lab08]$ ld -m elf_i386 -o lab8-4 lab8-4.o  
[fakhassan@fedora lab08]$ ./lab8-4  
Input A: 0  
Input X: 3  
1  
[fakhassan@fedora lab08]$ ./lab8-4  
Input A: 2  
Input X: 1  
6  
[fakhassan@fedora lab08]$  
[fakhassan@fedora lab08]$
```

Рис. 4.15: Программа lab8-4.asm

5 Выводы

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

Список литературы

1. Расширенный ассемблер: NASM
2. MASM, TASM, FASM, NASM под Windows и Linux