

Отчёта по лабораторной работе 8

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

Мухин Даниил Александрович НПИбд-02-22

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	23
	Список литературы	24

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	15
4.9	Файл листинга lab8-2	16
4.10	ошибка трансляции lab8-2	17
4.11	файл листинга с ошибкой lab8-2	18
4.12	Файл lab8-3.asm	19
4.13	Программа lab8-3.asm	20
4.14	Файл lab8-4.asm	21
4.15	Программа lab8-4.asm	22

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:

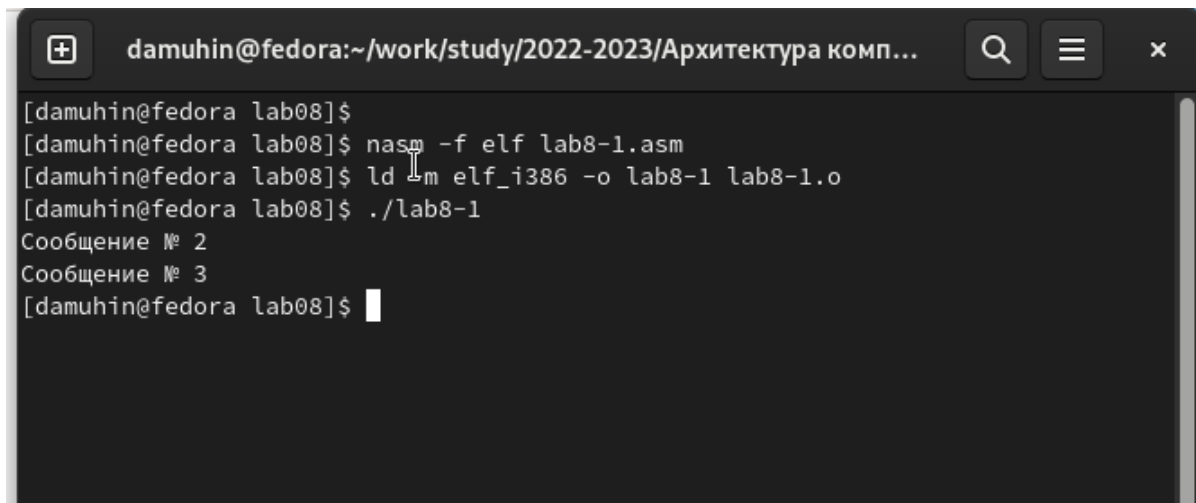
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'

_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 screenshot of a terminal window with a dark background. The window title is "damuhin@fedora:~/work/study/2022-2023/Архитектура комп...". The terminal shows the following commands and output:

```
[damuhin@fedora lab08]$  
[damuhin@fedora lab08]$ nasm -f elf lab8-1.asm  
[damuhin@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o  
[damuhin@fedora lab08]$ ./lab8-1  
Сообщение № 2  
Сообщение № 3  
[damuhin@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)

Открыть ▾  lab8-1.asm
~/work/study/2022-2023/Архитектура компьютера/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

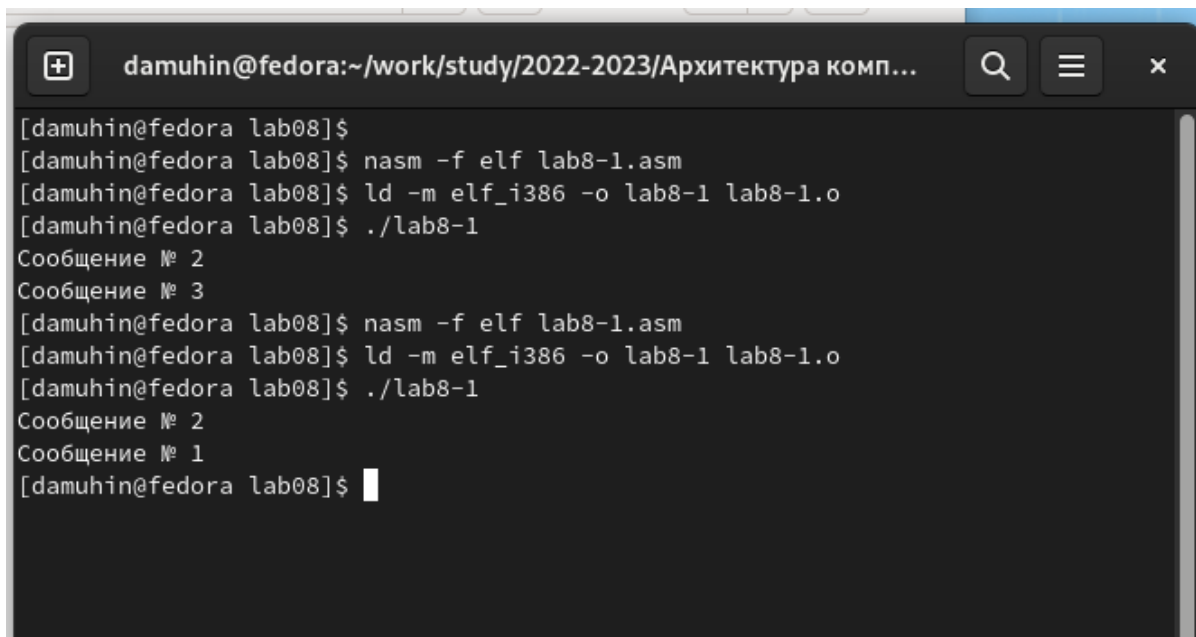
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end

_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:



```
damuhin@fedora:~/work/study/2022-2023/Архитектура комп...
[damuhin@fedora lab08]$
[damuhin@fedora lab08]$ nasm -f elf lab8-1.asm
[damuhin@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[damuhin@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[damuhin@fedora lab08]$ nasm -f elf lab8-1.asm
[damuhin@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[damuhin@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 1
[damuhin@fedora lab08]$
```

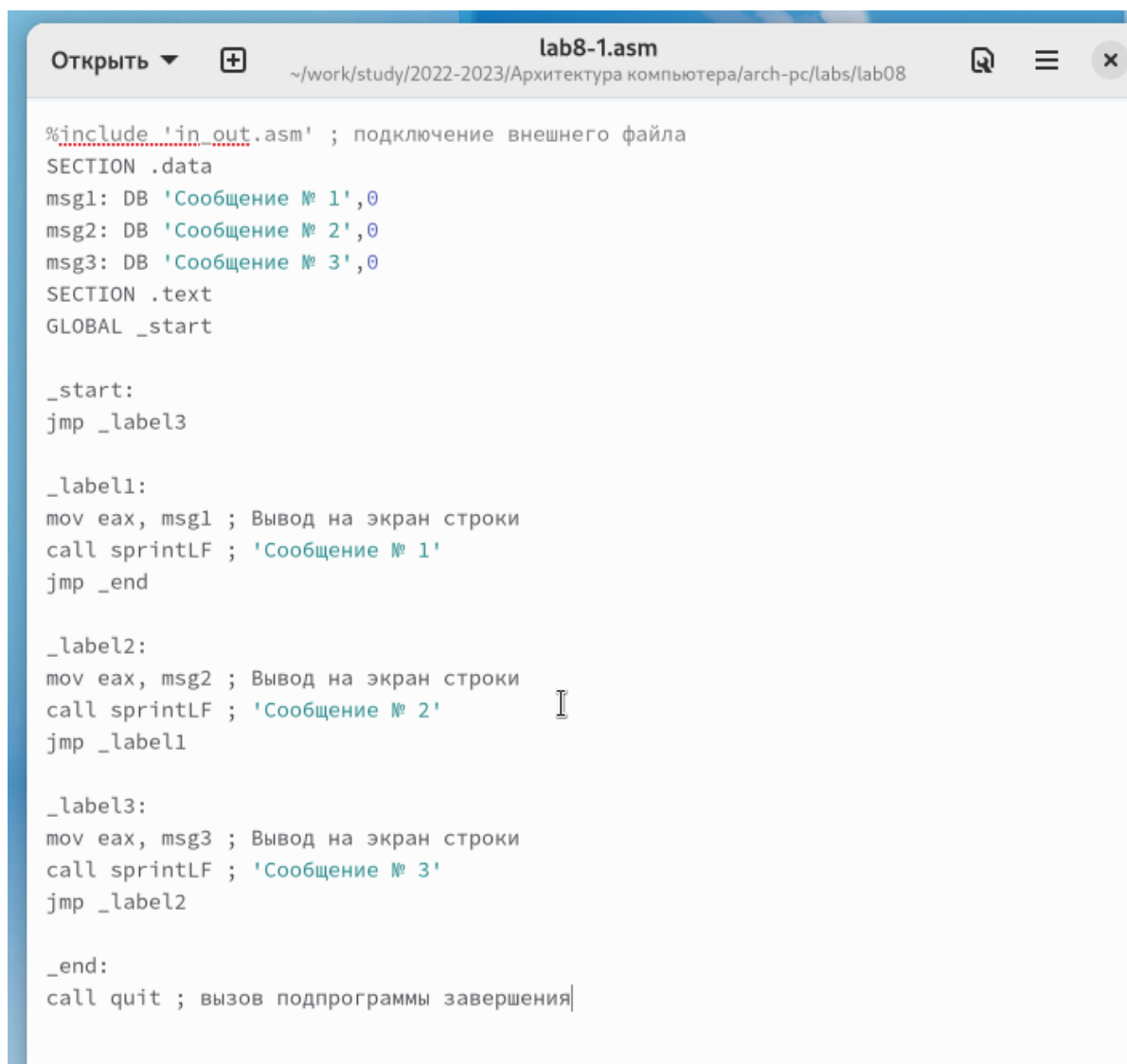
Рис. 4.4: Программа lab8-1.asm:

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

Сообщение № 3

Сообщение № 2

Сообщение № 1



```
Открыть ▾ + lab8-1.asm
~/work/study/2022-2023/Архитектура компьютера/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 _label3

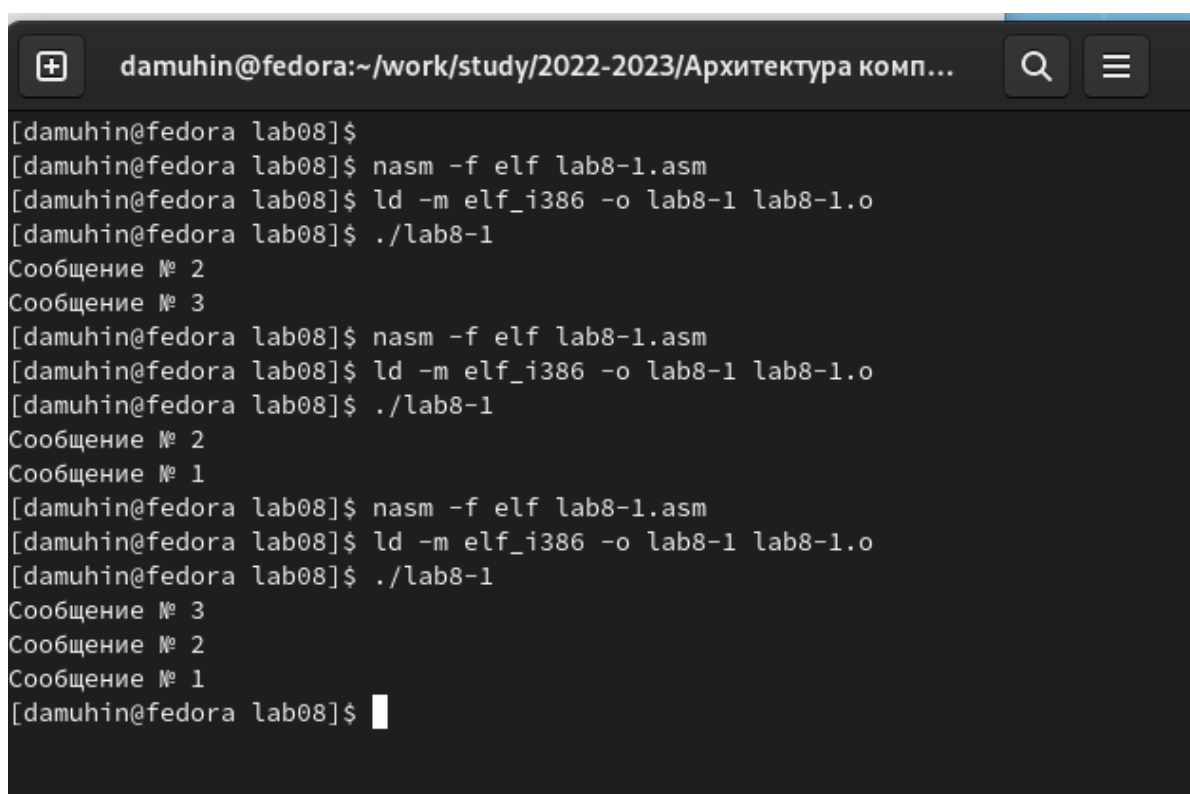
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end

_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
```

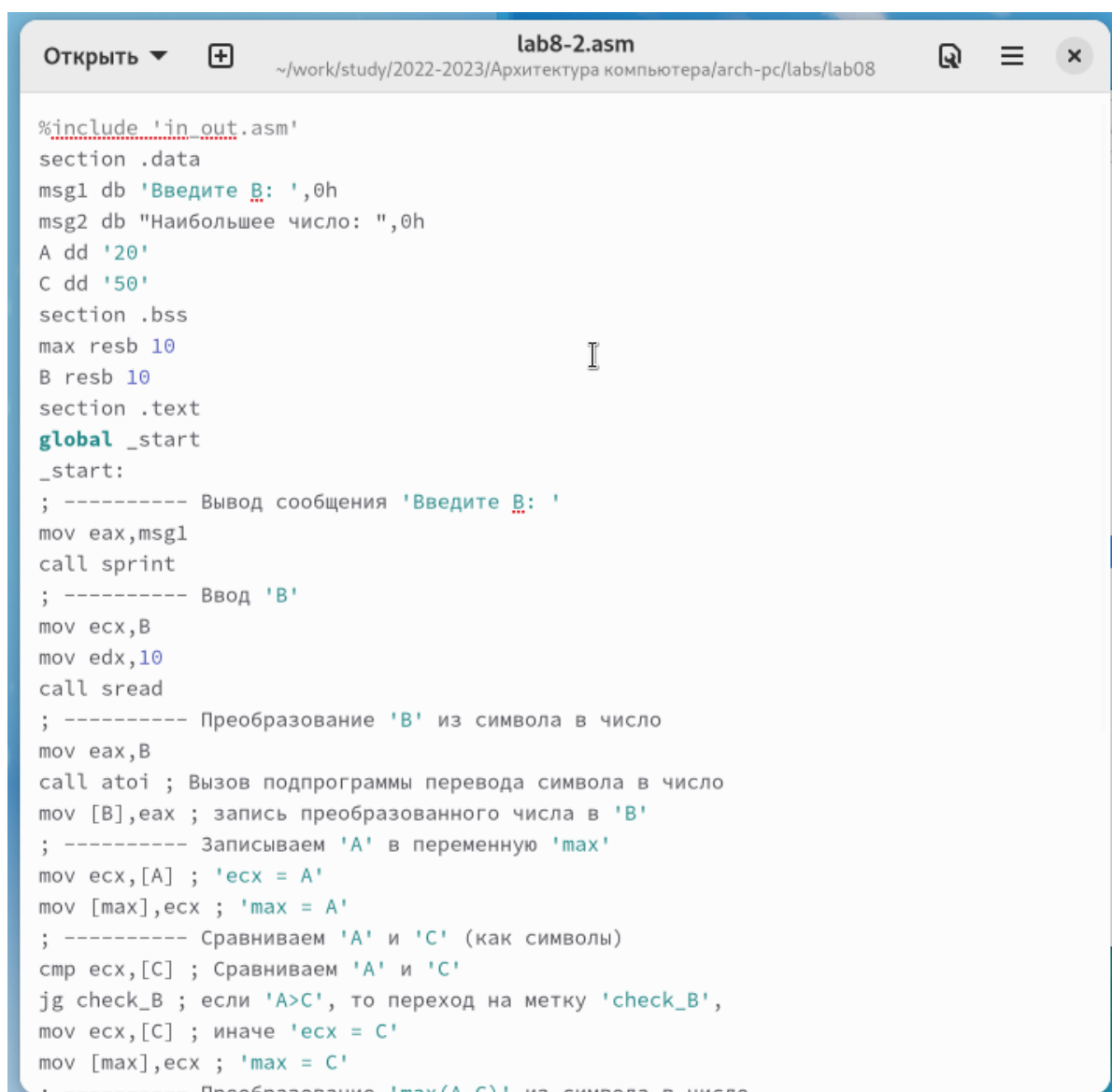
Рис. 4.5: Файл lab8-1.asm



```
damuhin@fedora:~/work/study/2022-2023/Архитектура комп...
[damuhin@fedora lab08]$
[damuhin@fedora lab08]$ nasm -f elf lab8-1.asm
[damuhin@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[damuhin@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[damuhin@fedora lab08]$ nasm -f elf lab8-1.asm
[damuhin@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[damuhin@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 1
[damuhin@fedora lab08]$ nasm -f elf lab8-1.asm
[damuhin@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[damuhin@fedora lab08]$ ./lab8-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[damuhin@fedora lab08]$
```

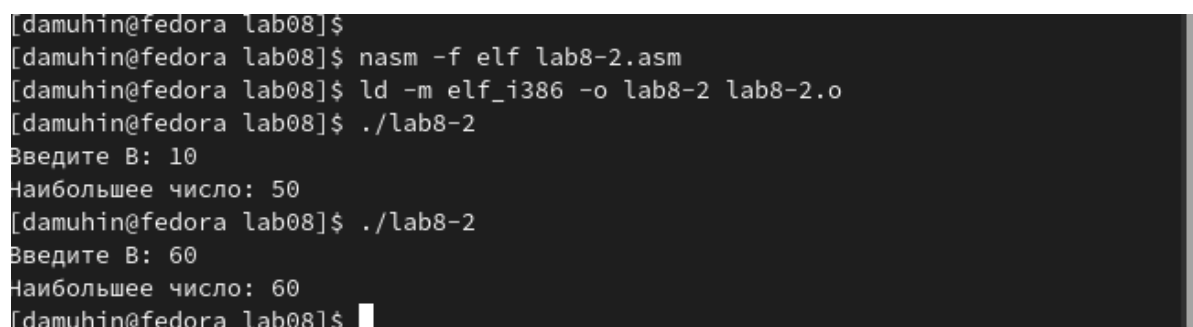
Рис. 4.6: Программа lab8-1.asm

- 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',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
```

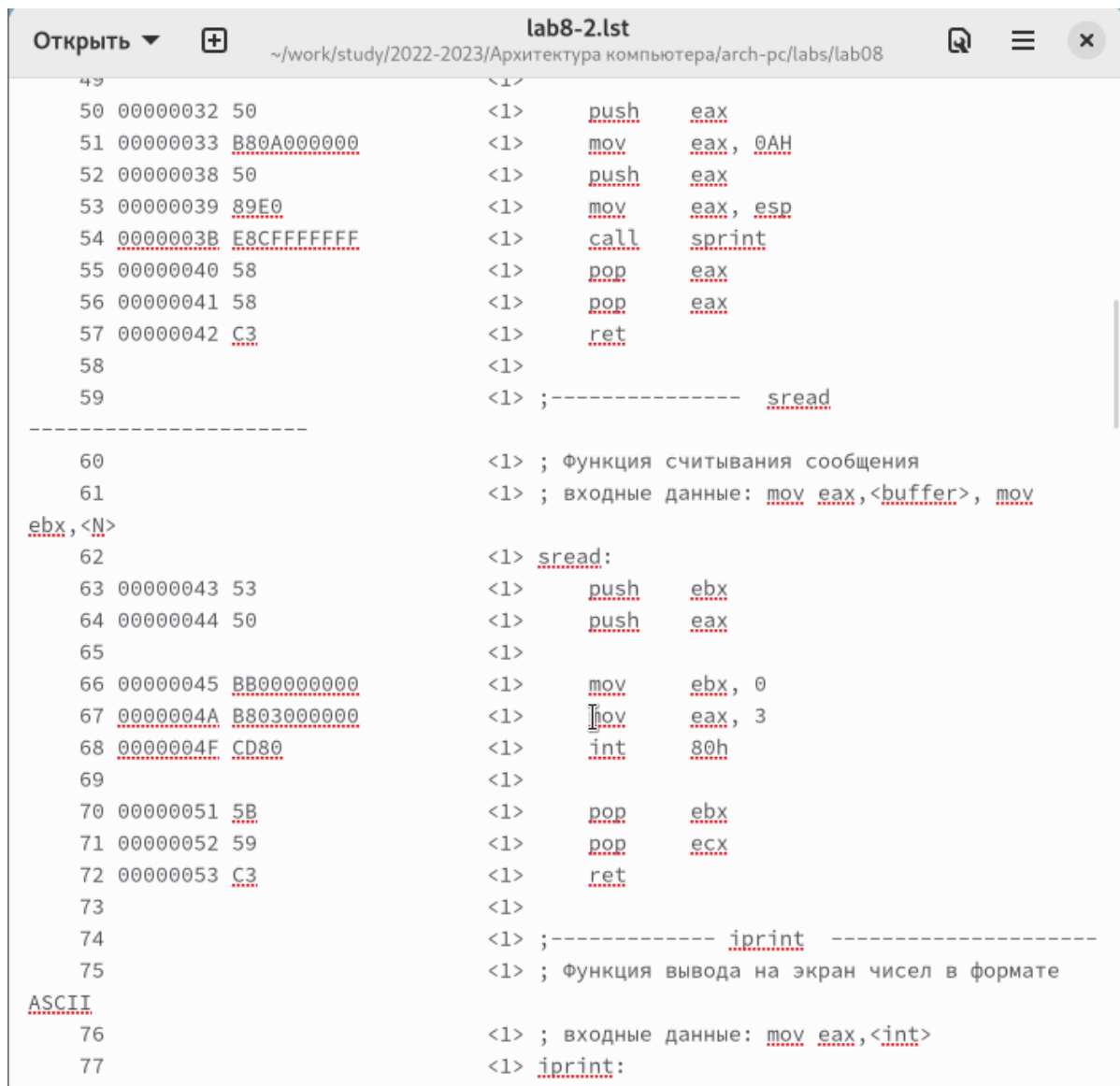
Рис. 4.7: Файл lab8-2.asm



```
[damuhin@fedora lab08]$
[damuhin@fedora lab08]$ nasm -f elf lab8-2.asm
[damuhin@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[damuhin@fedora lab08]$ ./lab8-2
Введите B: 10
Наибольшее число: 50
[damuhin@fedora lab08]$ ./lab8-2
Введите B: 60
Наибольшее число: 60
[damuhin@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)



```
Открыть + lab8-2.lst ~\work\study\2022-2023\Архитектура компьютера\arch-pc\labs\lab08
49
50 00000032 50 <1> push eax
51 00000033 B80A000000 <1> mov eax, 0AH
52 00000038 50 <1> push eax
53 00000039 89F0 <1> mov eax, esp
54 0000003B E8CFFFFFFF <1> call sprint
55 00000040 58 <1> pop eax
56 00000041 58 <1> pop eax
57 00000042 C3 <1> ret
58 <1>
59 <1> ;----- sread
-----
60 <1> ; Функция считывания сообщения
61 <1> ; входные данные: mov eax,<buffer>, mov
ebx,<N>
62 <1> sread:
63 00000043 53 <1> push ebx
64 00000044 50 <1> push eax
65 <1>
66 00000045 BB00000000 <1> mov ebx, 0
67 0000004A B803000000 <1> mov eax, 3
68 0000004F CD80 <1> int 80h
69 <1>
70 00000051 5B <1> pop ebx
71 00000052 59 <1> pop ecx
72 00000053 C3 <1> ret
73 <1>
74 <1> ;----- iprint -----
75 <1> ; Функция вывода на экран чисел в формате
ASCII
76 <1> ; входные данные: mov eax,<int>
77 <1> iprint:
-----
```

Рис. 4.9: Файл листинга lab8-2

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

строка 66

- 66- номер строки
- 00000045 - адрес
- BB00000000 - машинный код
- mov ebx, 0 - код программы

строка 67

- 67 - номер строки
- 0000004A - адрес
- B803000000 - машинный код
- mov eax, 3- код программы

строка 68

- 68 - номер строки
- 0000004F - адрес
- CD80 - машинный код
- int 80h- 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)

```
[damuhin@fedora lab08]$  
[damuhin@fedora lab08]$ nasm -f elf lab8-2.asm -l lab8-2.lst  
[damuhin@fedora lab08]$  
[damuhin@fedora lab08]$  
[damuhin@fedora lab08]$  
[damuhin@fedora lab08]$ nasm -f elf lab8-2.asm -l lab8-2.lst  
lab8-2.asm:28: error: invalid combination of opcode and operands  
[damuhin@fedora lab08]$
```

Рис. 4.10: ошибка трансляции lab8-2

```

Открыть  + lab8-2.lst ~\work\study\2022-2023\Архитектура компьютера\arch-pc\labs\lab08
10 , ----- 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 eax,B
22 00000106 E891FFFFFF call atoi ; Вызов подпрограммы перевода
символа в число
23 0000010B A3[0A000000] mov [B],eax ; запись преобразованного числа
в 'B'
24 ; ----- Записываем 'A' в переменную
'max'
25 00000110 8B0D[35000000] mov ecx,[A] ; 'ecx = A'
26 00000116 890D[00000000] mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'A' и 'C' (как
символы)
28 cmp ecx, ; Сравниваем 'A' и 'C'
28 *****
error: invalid combination of opcode and
operands
29 0000011C 7F0C jg check_B ; если 'A>C', то переход на метку
'check_B',
30 0000011E 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'
31 00000124 890D[00000000] mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' из
символа в число
33 check_B:
34 0000012A B8[00000000] mov eax,max
35 0000012F E868FFFFFF call atoi ; Вызов подпрограммы перевода
символа в число
36 00000134 A3[00000000] mov [max],eax ; запись преобразованного
числа в 'max'
37 ; ----- Сравниваем 'max(A,C)' и

```

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 9 - 24,98,15

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

SECTION .text
    GLOBAL _start

_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

```

[damuhin@fedora lab08]$
[damuhin@fedora lab08]$ nasm -f elf lab8-3.asm
[damuhin@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[damuhin@fedora lab08]$ ./lab8-3
Input A: 24
Input B: 98
Input C: 15
Smallest: 15
[damuhin@fedora lab08]$
[damuhin@fedora lab08]$
[damuhin@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 9

$$\begin{cases} a + x, x \leq a \\ a, x > a \end{cases}$$

```
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]
mov edx, [A]
cmp ebx, edx
jbe first
jmp second

first:
mov eax,[X]
mov ebx,[A]
add eax,ebx
call iprintLF
call quit
second:
mov eax, [A]
call iprintLF
call quit
```

Рис. 4.14: Файл lab8-4.asm

```
[damuhin@fedora lab08]$  
[damuhin@fedora lab08]$ nasm -f elf lab8-4.asm  
[damuhin@fedora lab08]$ ld -m elf_i386 -o lab8-4 lab8-4.o  
[damuhin@fedora lab08]$ ./lab8-4  
Input A: 7  
Input X: 5  
12  
[damuhin@fedora lab08]$  
[damuhin@fedora lab08]$  
[damuhin@fedora lab08]$ ./lab8-4  
Input A: 4  
Input X: 6  
4  
[damuhin@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