

Отчёта по лабораторной работе 8

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

Дудырев Г. А. НПИбд-01-22

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	20
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

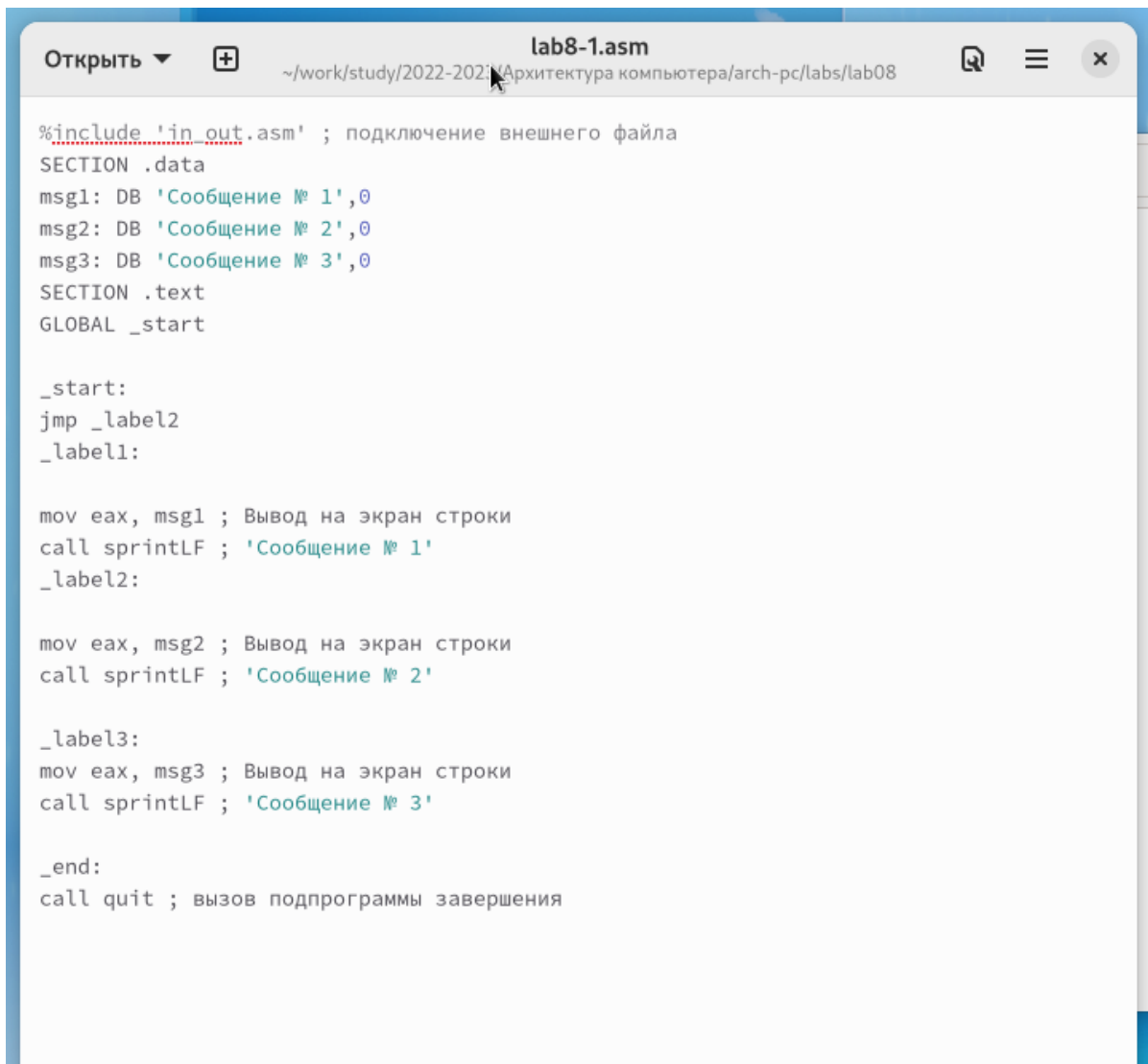
Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- 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)



```
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'
_label2:

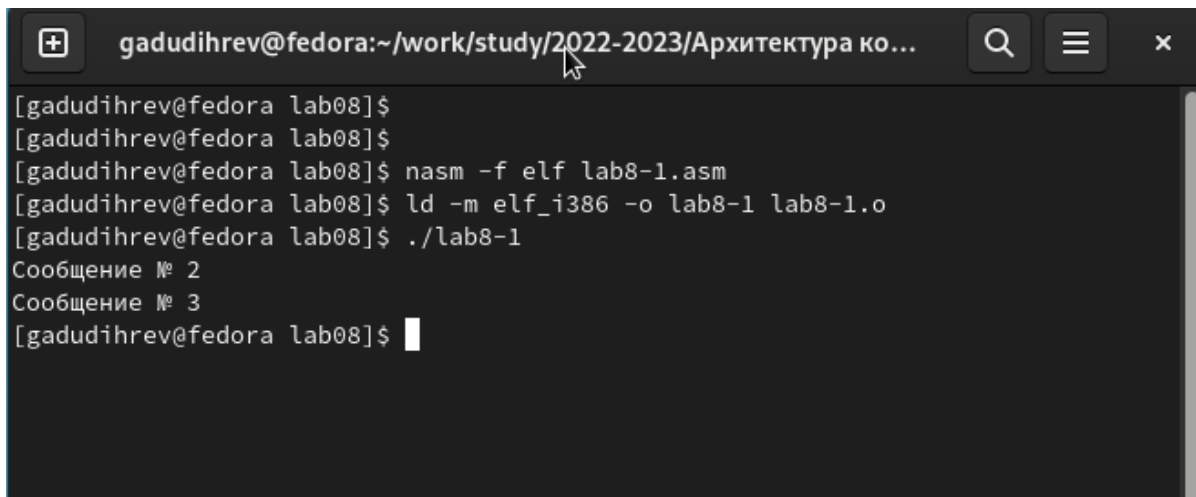
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'

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

_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.1: Файл lab8-1.asm:

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

A terminal window with a dark background. The title bar shows the user 'gadudihrev' on a 'fedora' machine, in the directory '~/work/study/2022-2023/Архитектура ко...'. The terminal contains the following text:

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

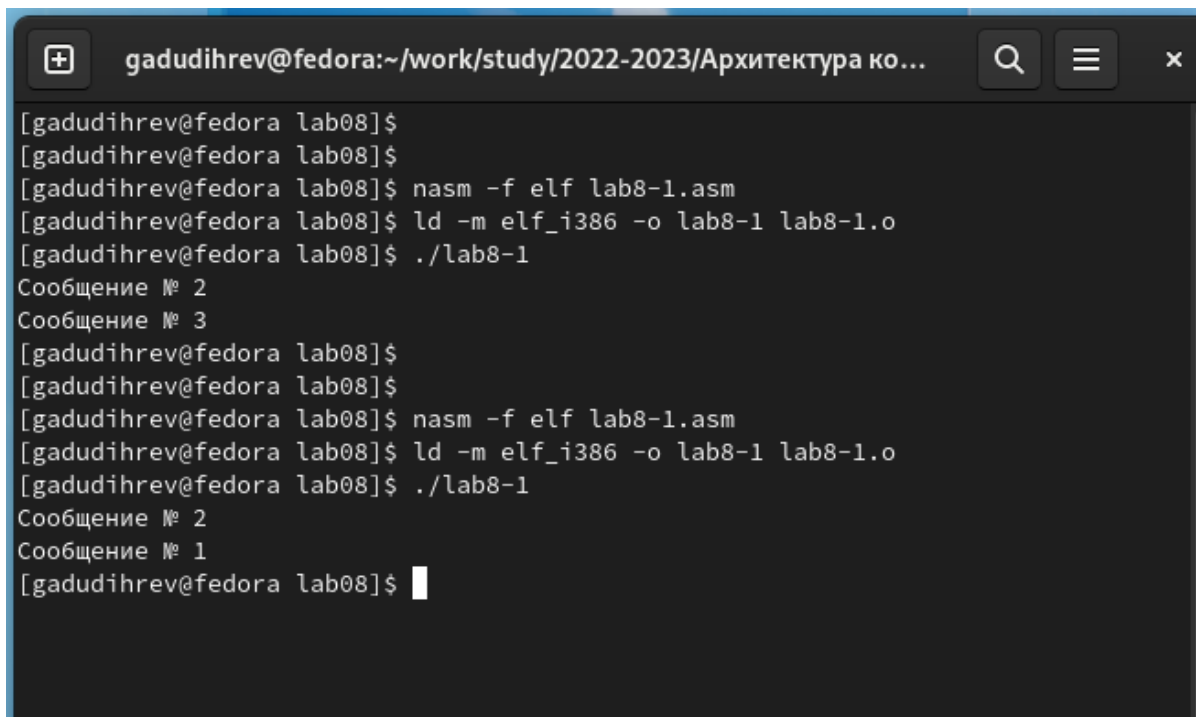
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 1'
jmp _end

_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 2'
jmp _label1

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

_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.3: Файл lab8-1.asm:

A terminal window with a dark background and light text. The title bar shows the user 'gadudihrev@fedora' and the current directory '~/work/study/2022-2023/Архитектура ко...'. The terminal contains the following commands and output:

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

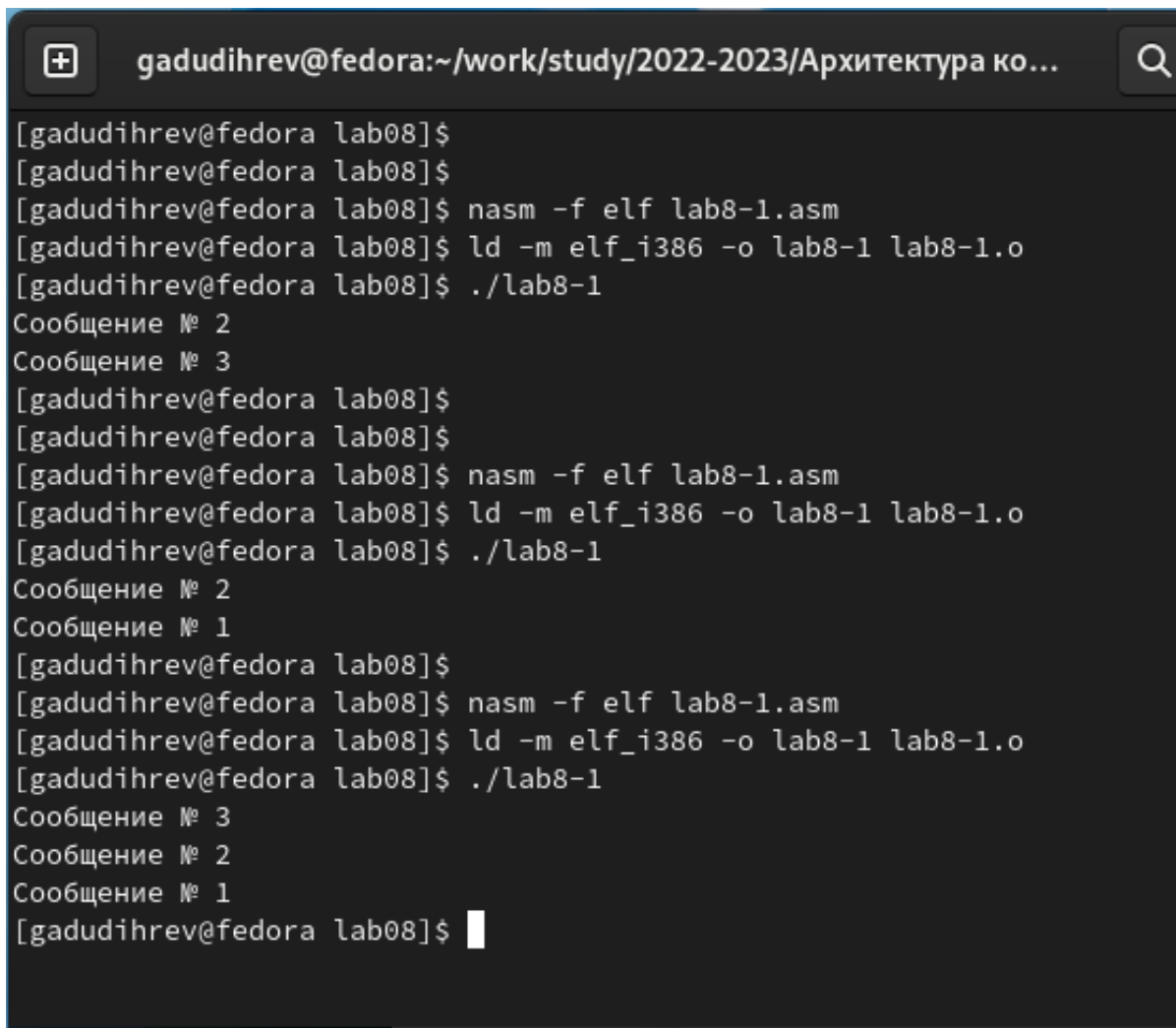
_label1:
mov eax, msg1 ; Вывод на экран строки
call printf ; 'Сообщение № 1'
jmp _end

_label2:
mov eax, msg2 ; Вывод на экран строки
call printf ; 'Сообщение № 2'
jmp _label1

_label3:
mov eax, msg3 ; Вывод на экран строки
call printf ; 'Сообщение № 3'
jmp _label2

_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.5: Файл lab8-1.asm



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

```
lab8-2.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab08

lab8-1.asm lab8-2.asm x
mov ecx, 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
check_B:
mov eax, max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max], eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A, C)' и 'B' (как числа)
mov ecx, [max]
cmp ecx, [B] ; Сравниваем 'max(A, C)' и 'B'
jg fin ; если 'max(A, C) > B', то переход на 'fin',
mov ecx, [B] ; иначе 'ecx = B'
mov [max], ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprintf ; Вывод сообщения 'Наибольшее число: '
mov eax, [max]
call iprintLF ; Вывод 'max(A, B, C)'
call quit ; Выход
```

Рис. 4.7: Файл lab8-2.asm

```
[gadudihrev@fedora lab08]$  
[gadudihrev@fedora lab08]$  
[gadudihrev@fedora lab08]$ nasm -f elf lab8-2.asm  
[gadudihrev@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o  
[gadudihrev@fedora lab08]$ ./lab8-2  
Введите B: 1  
Наибольшее число: 50  
[gadudihrev@fedora lab08]$ ./lab8-2  
Введите B: 70  
Наибольшее число: 70  
[gadudihrev@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)


```
1                                     %include 'in_out.asm'
2                                     <1> ;----- slen -----
3                                     <1> ; Функция вычисления длины сообщения
4                                     <1> slen:
5 00000000 53                         <1> push     ebx
6 00000001 89C3                       <1> mov      ebx, eax
7                                     <1>
8                                     <1> nextchar:
9 00000003 803800                     <1> cmp      byte [eax], 0
10 00000006 7403                      <1> jz       finished
11 00000008 40                        <1> inc      eax
12 00000009 EBF8                      <1> jmp      nextchar
13                                     <1>
14                                     <1> finished:
15 0000000B 29D8                      <1> sub      eax, ebx
16 0000000D 5B                        <1> pop      ebx
17 0000000E C3                        <1> ret
18                                     <1>
19                                     <1>
20                                     <1> ;----- sprint -----
21                                     <1> ; Функция печати сообщения
22                                     <1> ; входные данные: mov eax, <message>
23                                     <1> sprint:
24 0000000F 52                        <1> push     edx
25 00000010 51                        <1> push     ecx
26 00000011 53                        <1> push     ebx
27 00000012 50                        <1> push     eax
28 00000013 E8E8FFFFFF                <1> call     slen
29                                     <1>
30 00000018 89C2                      <1> mov      edx, eax
31 0000001A 58                        <1> pop      eax
```

Рис. 4.9: Файл листинга lab8-2

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

строка 15

- 15 - номер строки
- 0000000B - адрес
- 29D8 - машинный код
- sub eax, ebx - код программы

строка 16

- 16 - номер строки
- 0000000D - адрес
- 5B - машинный код
- pop ebx- код программы

строка 17

- 17 - номер строки
- 0000000E - адрес
- C3 - машинный код
- ret - код программы

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

```
[gadudihrev@fedora lab08]$
[gadudihrev@fedora lab08]$ nasm -f elf lab8-2.asm
[gadudihrev@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[gadudihrev@fedora lab08]$ ./lab8-2
Введите B: 1
Наибольшее число: 50
[gadudihrev@fedora lab08]$ ./lab8-2
Введите B: 70
Наибольшее число: 70
[gadudihrev@fedora lab08]$ nasm -f elf lab8-2.asm -l lab8-2.lst
[gadudihrev@fedora lab08]$
[gadudihrev@fedora lab08]$
[gadudihrev@fedora lab08]$ nasm -f elf lab8-2.asm -l lab8-2.lst
lab8-2.asm:34: error: invalid combination of opcode and operands
[gadudihrev@fedora lab08]$
```

Рис. 4.10: ошибка трансляции lab8-2

```

lab8-2.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab08

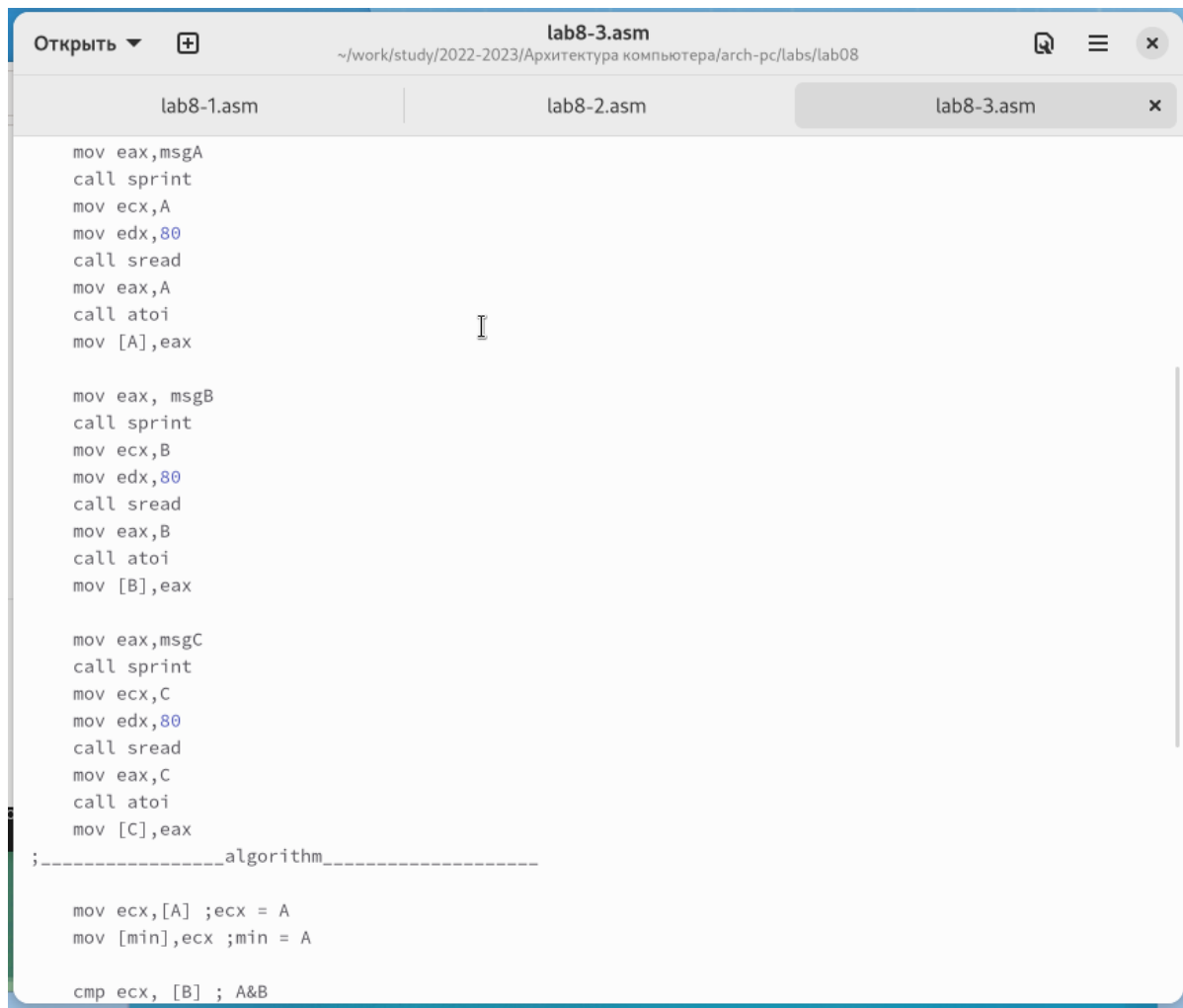
lab8-1.asm | lab8-2.asm | lab8-2.lst x
19 0000000C 00411111
20 ; ----- Преобразование 'B' из символа в число
21 00000101 B8[0A000000] mov eax,B
22 00000106 E891FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
23 0000010B A3[0A000000] mov [B],eax ; запись преобразованного числа в 'B'
24 ; ----- 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 0000011C 3B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 00000122 7F0C jg check_B ; если 'A>C', то переход на метку 'check_B',
30 00000124 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'
31 0000012A 890D[00000000] mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 mov ecx,
error: invalid combination of opcode and operands
35 00000130 E867FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
36 00000135 A3[00000000] mov [max],eax ; запись преобразованного числа в 'max'
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013A 8B0D[00000000] mov ecx,[max]
39 00000140 3B0D[0A000000] cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 00000146 7F0C jg fin ; если 'max(A,C)>B', то переход на 'fin',
41 00000148 8B0D[0A000000] mov ecx,[B] ; иначе 'ecx = B'
42 0000014E 890D[00000000] mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 00000154 B8[13000000] mov eax,msg2
46 00000159 E8B1FFFFFF call sprintf ; Вывод сообщения 'Наибольшее число: '
47 0000015E A1[00000000] mov eax,[max]
48 00000163 E81EFFFFFF call iprintLF ; Вывод 'max(A,B,C)'
49 00000168 E86EFFFFFF call quit ; Выход

```

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 14 - 81, 22, 72



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

lab8-1.asm | lab8-2.asm | lab8-3.asm x

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

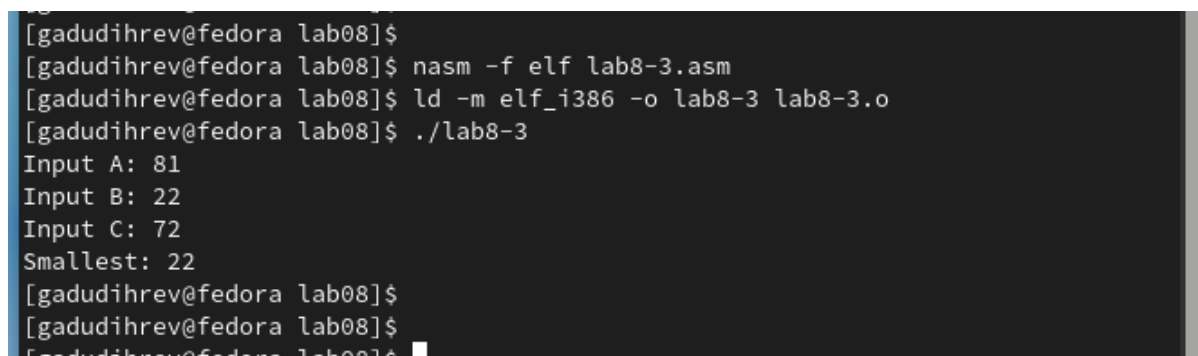
mov eax,msgC
call sprint
mov ecx,C
mov edx,80
call sread
mov eax,C
call atoi
mov [C],eax

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

mov ecx,[A] ;ecx = A
mov [min],ecx ;min = A

cmp ecx, [B] ; A&B
```

Рис. 4.12: Файл lab8-3.asm



```
[gadudihrev@fedora lab08]$
[gadudihrev@fedora lab08]$ nasm -f elf lab8-3.asm
[gadudihrev@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[gadudihrev@fedora lab08]$ ./lab8-3
Input A: 81
Input B: 22
Input C: 72
Smallest: 22
[gadudihrev@fedora lab08]$
[gadudihrev@fedora lab08]$
[gadudihrev@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

$$\begin{cases} 3a + 1, & x < a \\ 3x + 1, & x \geq a \end{cases}$$

Открыть ▾ 

lab8-4.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab08

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

first:
mov eax,[A]
mov ebx,3
mul ebx
add eax,1
call iprintLF
call quit
second:
mov eax,[X]
mov ebx,3
mul ebx
add eax,1
call iprintLF
call quit
```

Рис. 4.14: Файл lab8-4.asm

```
[gadudihrev@fedora lab08]$  
[gadudihrev@fedora lab08]$ nasm -f elf lab8-4.asm  
[gadudihrev@fedora lab08]$ ld -m elf_i386 -o lab8-4 lab8-4.o  
[gadudihrev@fedora lab08]$ ./lab8-4  
Input A: 3  
Input X: 2  
10  
[gadudihrev@fedora lab08]$ ./lab8-4  
Input A: 4  
Input X: 2  
13  
[gadudihrev@fedora lab08]$ ./lab8-4  
Input A: 2  
Input X: 4  
13  
[gadudihrev@fedora lab08]$  
[gadudihrev@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