

Отчёта по лабораторной работе 8

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

Понамарев Алексей Михайлович НПИбд-02-22

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

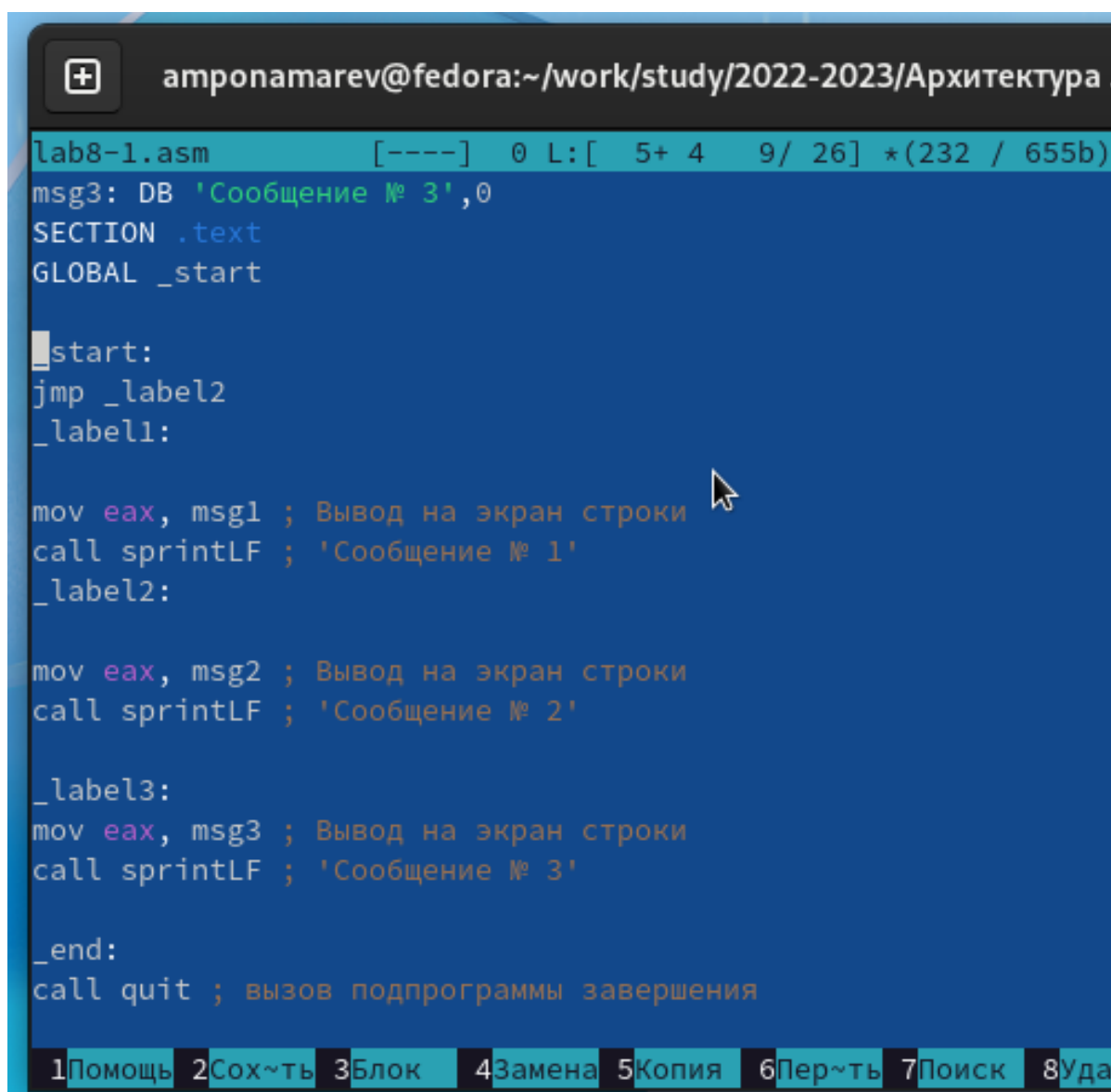
Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- 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 [----] 0 L:[ 5+ 4 9/ 26] *(232 / 655b)
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'

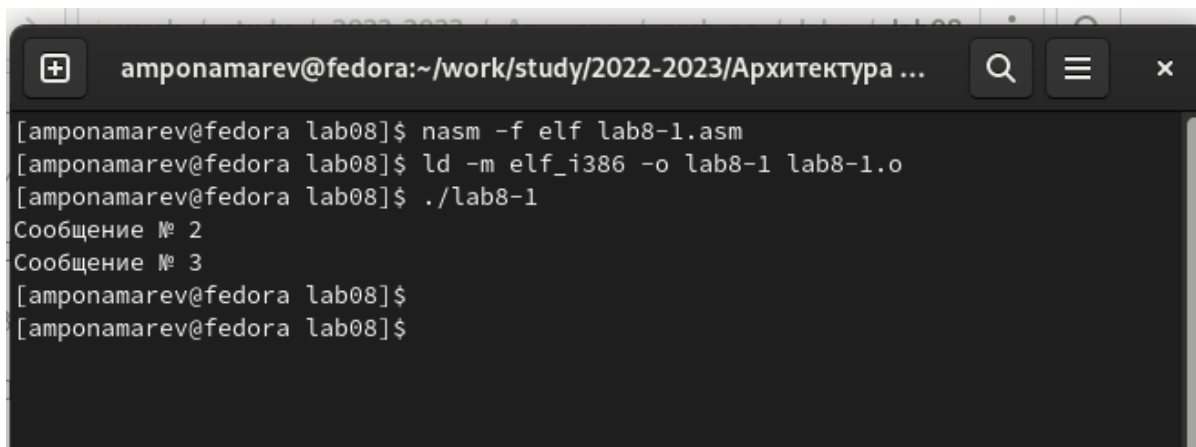
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'

_end:
call quit ; вызов подпрограммы завершения
```

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить

Рис. 4.1: Файл lab8-1.asm:

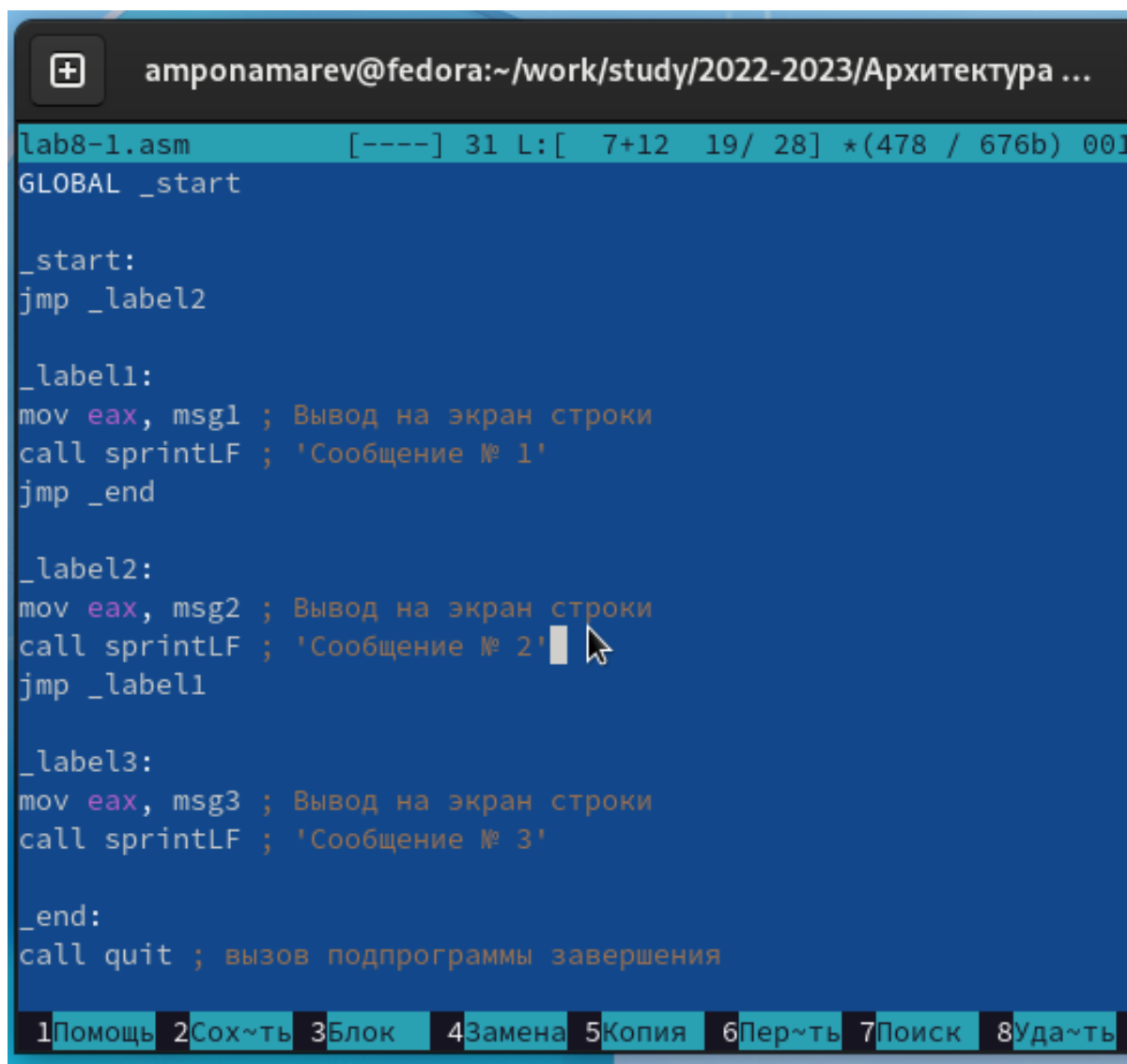
Создайте исполняемый файл и запустите его. (рис. 4.2)

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

```
[amponamarev@fedora lab08]$ nasm -f elf lab8-1.asm
[amponamarev@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[amponamarev@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[amponamarev@fedora lab08]$
[amponamarev@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      [----] 31 L:[ 7+12 19/ 28] *(478 / 676b) 001
GLOBAL _start

_start:
jmp _label2

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

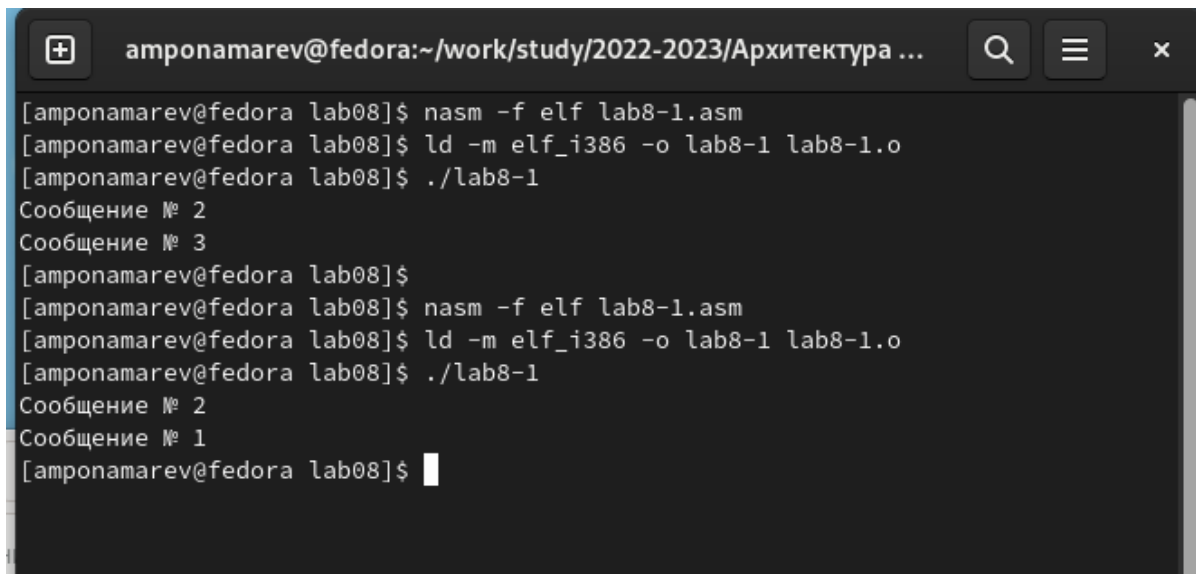
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1

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

_end:
call quit ; вызов подпрограммы завершения

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Переместить 7Поиск 8Удалить
```

Рис. 4.3: Файл lab8-1.asm:

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

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

```
amponamarev@fedora:~/work/study/2022-2023/Архитектура ...
lab8-1.asm [----] 8 L:[ 8+15 23/ 29] *(509 / 688b) 00

_start:
jmp _label3

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

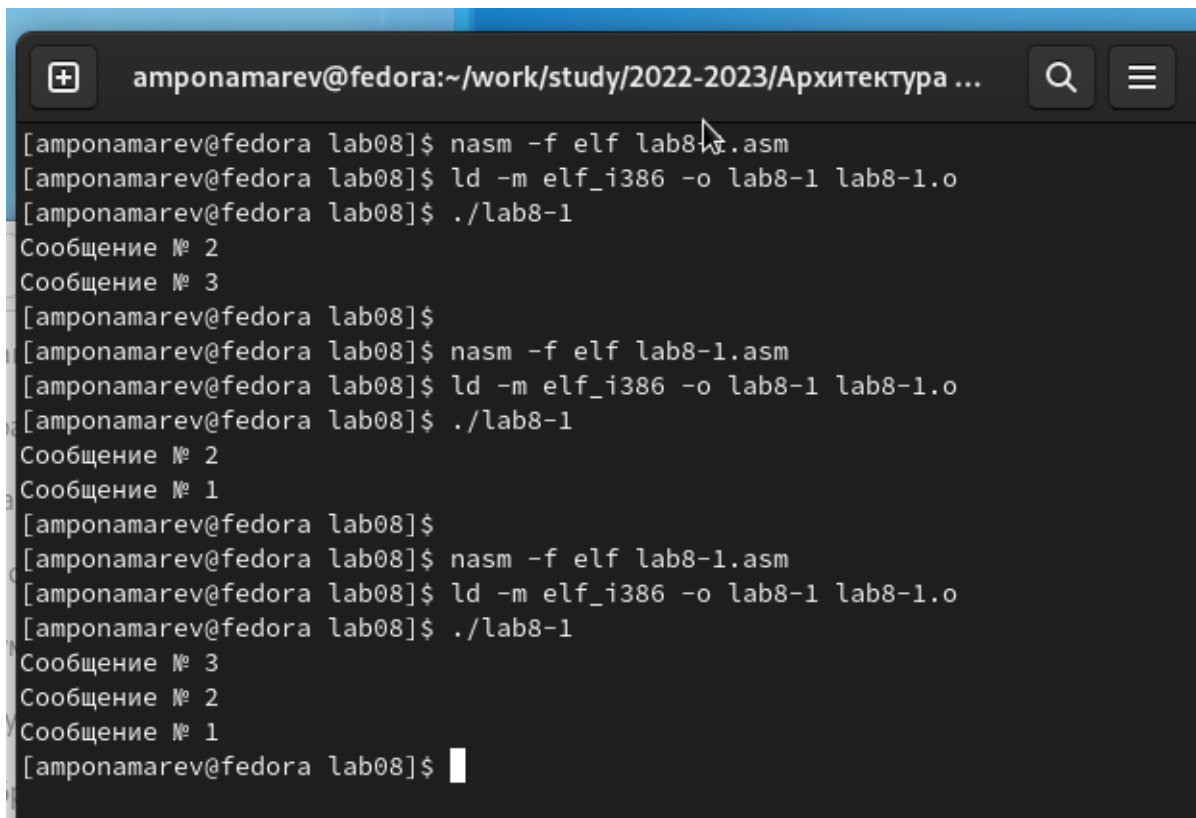
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'
jmp _label1

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

_end:
call quit ; вызов подпрограммы завершения

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить
```

Рис. 4.5: Файл lab8-1.asm



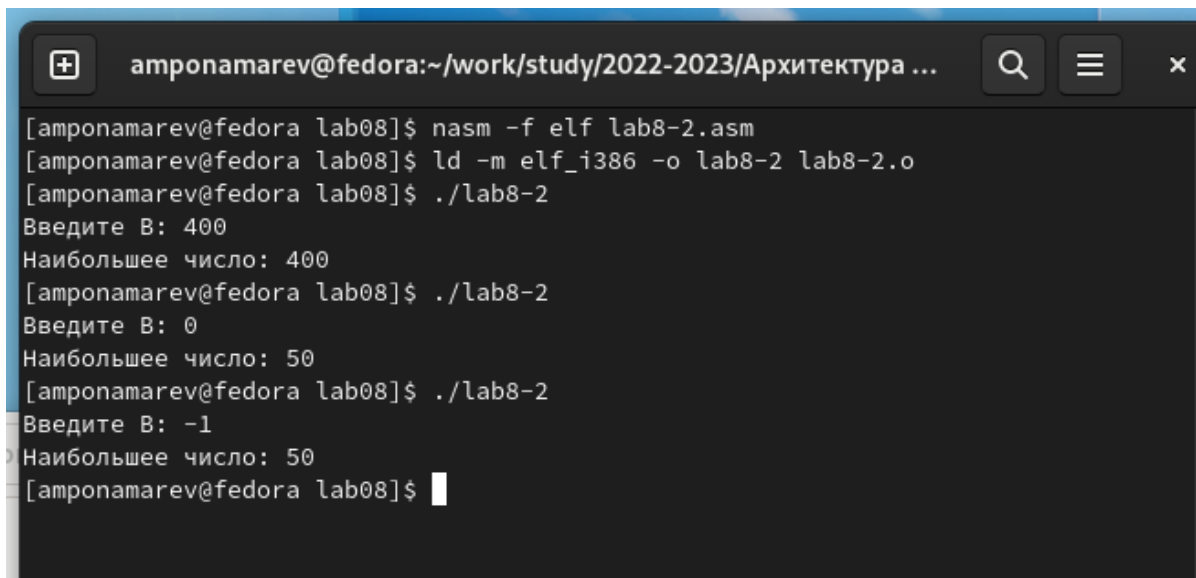
```
amponamarev@fedora:~/work/study/2022-2023/Архитектура ...  
[amponamarev@fedora lab08]$ nasm -f elf lab8-1.asm  
[amponamarev@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o  
[amponamarev@fedora lab08]$ ./lab8-1  
Сообщение № 2  
Сообщение № 3  
[amponamarev@fedora lab08]$  
[amponamarev@fedora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

```
amponamarev@fedora:~/work/study/2022-2023/Архитектура ...
lab8-2.asm      [----]  0 L:[ 25+ 0  25/ 51] *(683 /1745b) 0109
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 sprint ; Вывод сообщения 'Наибольшее число: '
1Помощь 2Сох~ть 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9М
```

Рис. 4.7: Файл lab8-2.asm

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

```
[ampronamarev@fedora lab08]$ nasm -f elf lab8-2.asm
[ampronamarev@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[ampronamarev@fedora lab08]$ ./lab8-2
Введите B: 400
Наибольшее число: 400
[ampronamarev@fedora lab08]$ ./lab8-2
Введите B: 0
Наибольшее число: 50
[ampronamarev@fedora lab08]$ ./lab8-2
Введите B: -1
Наибольшее число: 50
[ampronamarev@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)


```
lab8-2.lst [----] 0 L: [ 19+ 0 19/226] *(1185/14499b) 0032 0x020 [*][X]
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
32 <1> .....
33 0000001B 89C1 <1> mov ecx, eax
34 0000001D BB01000000 <1> mov ebx, 1
35 00000022 B804000000 <1> mov eax, 4
36 00000027 CD80 <1> int 80h
37 <1> .
38 00000029 5B <1> pop ebx
39 0000002A 59 <1> pop ecx
40 0000002B 5A <1> pop edx
1 Помощь 2 Сох~ть 3 Блок 4 Замена 5 Копия 6 Пер~ть 7 Поиск 8 Уда~ть 9 МенюМС 10 Выход
```

Рис. 4.9: Файл листинга lab8-2

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

строка 33

- 33 - номер строки
- 0000001B - адрес
- 89C1 - машинный код
- mov ecx, eax - код программы

строка 34

- 34 - номер строки

- 0000001D - адрес
- BB01000000 - машинный код
- mov ebx, 1 - код программы

строка 35

- 35 - номер строки
- 00000022 - адрес
- B804000000 - машинный код
- mov eax, 4 - 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: 30
[amponamarev@fedora lab08]$ nasm -f elf lab8-2.asm -l lab8-2.lst
[amponamarev@fedora lab08]$
[amponamarev@fedora lab08]$
[amponamarev@fedora lab08]$ nasm -f elf lab8-2.asm -l lab8-2.lst
lab8-2.asm:17: error: invalid combination of opcode and operands
[amponamarev@fedora lab08]$

```

Рис. 4.10: ошибка трансляции lab8-2

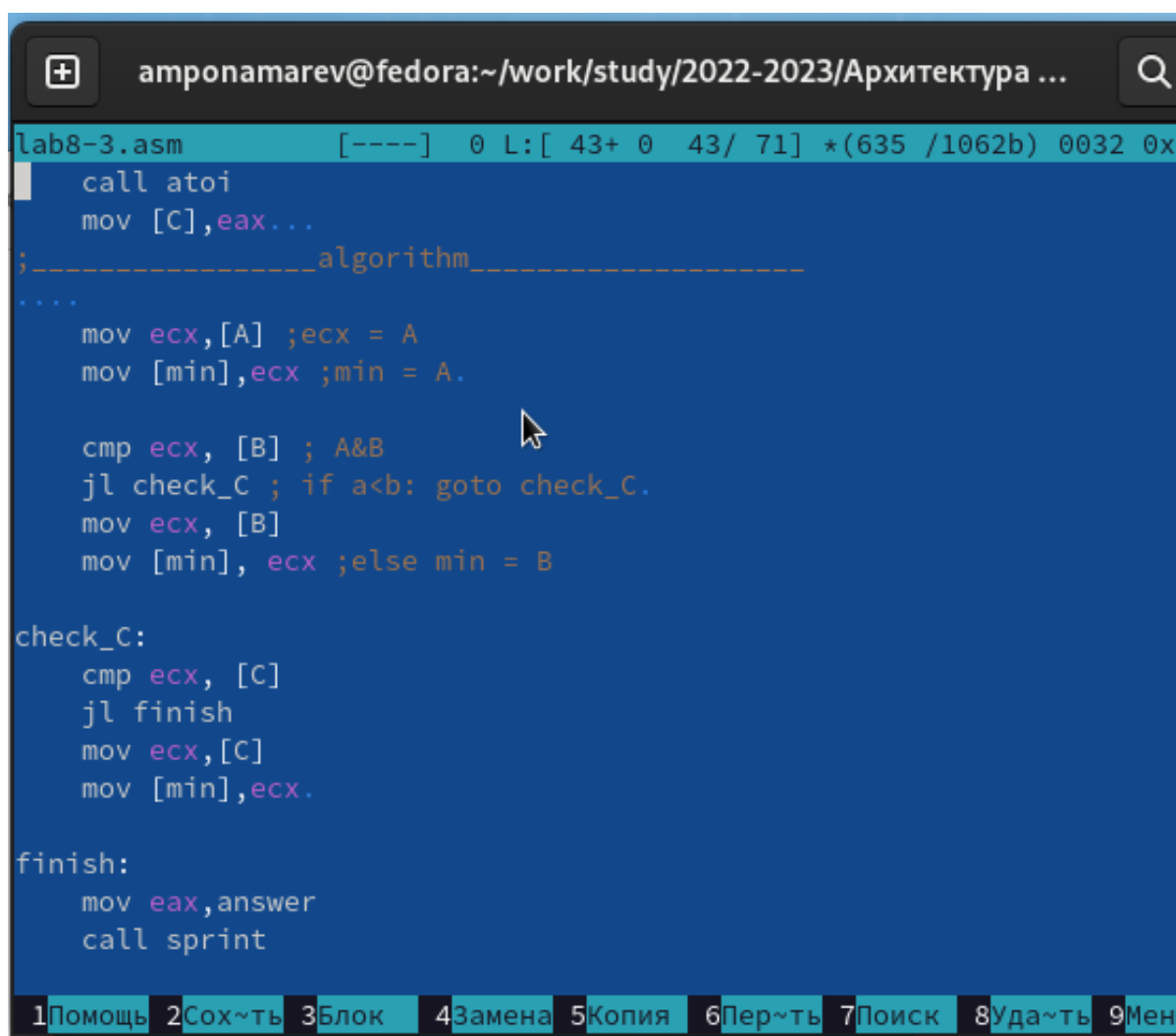
```
lab8-2.lst [----] 24 L:[185+ 0 185/227] *(11308/14588b) 0032 0x020[*][X]
10                                     section .text
11                                     global _start
12                                     _start:
13                                     ; ----- Вывод сообщения 'Введите B:
14 000000E8 B8[00000000]               mov eax,msg1
15 000000ED E81DFFFFFF               call sprint
16                                     ; ----- Ввод 'B'
17                                     mov ecx
17                                     ***** error: invalid combination of opcode and
18 000000F2 BA0A000000               mov edx,10
19 000000F7 E847FFFFFF               call sread
20                                     ; ----- Преобразование 'B' из символа
21 000000FC B8[0A000000]               mov eax,B
22 00000101 E896FFFFFF               call atoi ; Вызов подпрограммы перевода
23 00000106 A3[0A000000]               mov [B],eax ; запись преобразованного числа
24                                     ; ----- Записываем 'A' в переменную
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 m
30 0000011F 8B0D[39000000]            mov ecx,[C] ; иначе 'ecx = C'

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9МенюМС10Выход
```

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 8 - 52 33 40



```
lab8-3.asm  [-----]  0  L: [ 43+ 0  43/ 71]  *(635 /1062b) 0032 0x
call atoi
mov [C],eax...
;-----algorithm-----
....
mov ecx,[A] ;ecx = A
mov [min],ecx ;min = A.

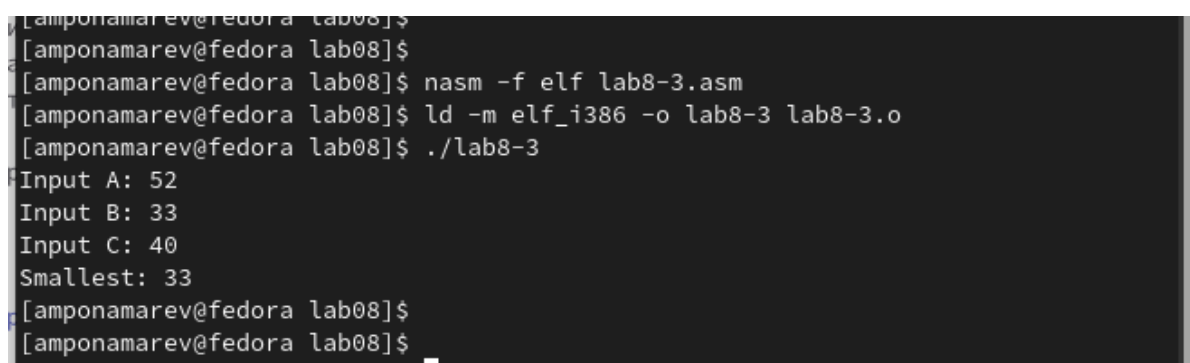
cmp ecx, [B] ; A&B
jnl check_C ; if a<b: goto check_C.
mov ecx, [B]
mov [min], ecx ;else min = B

check_C:
cmp ecx, [C]
jnl finish
mov ecx,[C]
mov [min],ecx.

finish:
mov eax,answer
call sprint
```

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9Меню

Рис. 4.12: Файл lab8-3.asm



```
[amponamarev@fedora lab08]$
[amponamarev@fedora lab08]$
[amponamarev@fedora lab08]$ nasm -f elf lab8-3.asm
[amponamarev@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[amponamarev@fedora lab08]$ ./lab8-3
Input A: 52
Input B: 33
Input C: 40
Smallest: 33
[amponamarev@fedora lab08]$
[amponamarev@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 8

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



```
amponamarev@fedora:~/work/study/2022-2023/Apx
lab8-4.asm [----] 13 L: [ 30+17 47/ 52] *(684
    call atoi
    mov [X],eax...
;-----algorithm-----

    mov ebx, [A]
    cmp ebx, 3
    jb first
    jmp second

first:
    mov eax, [A]
    mov ebx, 3
    mul ebx
    call iprintLF.
    call quit

second:
    mov eax, [X]
    add eax, 1
    call iprintLF.
    call quit
```

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск

Рис. 4.14: Файл lab8-4.asm

```
[amponamarev@fedora lab08]$  
[amponamarev@fedora lab08]$  
[amponamarev@fedora lab08]$ nasm -f elf lab8-4.asm  
[amponamarev@fedora lab08]$ ld -m elf_i386 -o lab8-4 lab8-4.o  
[amponamarev@fedora lab08]$ ./lab8-4  
Input A: 4  
Input X: 11  
12  
[amponamarev@fedora lab08]$ ./lab8-4  
Input A: 4  
Input X: 1  
2  
[amponamarev@fedora lab08]$ ./lab8-4  
Input A: 2  
Input X: 1  
6  
[amponamarev@fedora lab08]$  
[amponamarev@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