

# **Лабораторная работа 7**

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

Головина Мария Игоревна

# Содержание

|          |                                       |           |
|----------|---------------------------------------|-----------|
| <b>1</b> | <b>Цель работы</b>                    | <b>5</b>  |
| <b>2</b> | <b>Задание</b>                        | <b>6</b>  |
| <b>3</b> | <b>Теоретическое введение</b>         | <b>8</b>  |
| <b>4</b> | <b>Выполнение лабораторной работы</b> | <b>11</b> |
| <b>5</b> | <b>Выводы</b>                         | <b>24</b> |
|          | <b>Список литературы</b>              | <b>25</b> |

## Список иллюстраций

|      |   |    |
|------|---|----|
| 4.1  | Создание каталога и файла для выполнения лабораторной работы                        | 11 |
| 4.2  | Листинг 1 . . . . .   | 12 |
| 4.3  | Результаты работы программы из листинга 1 методического указания                    | 12 |
| 4.4  | Листинг 1 с внесенными изменениями . . . . .  | 13 |
| 4.5  | Результаты работы Листинга 1 с внесенными изменениями . . . .                       | 13 |
| 4.6  | Листинг 2 . . . . .   | 14 |
| 4.7  | Результаты работы программы из Листинга 2 методического указания                    | 15 |
| 4.8  | Листинг 3 . . . . .   | 16 |
| 4.9  | Результаты работы программы из листинга 3 методического указания                    | 16 |
| 4.10 | Создание файл листинга для программы для программы из файла<br>lab7-2.asm . . . . . | 17 |
| 4.11 | Просмотр файла lab7-2.asm с помощью текстового редактора . . .                      | 17 |
| 4.12 | Объяснение 1-й выбранной строки с листинга файла . . . . .                          | 18 |
| 4.13 | Объяснение 2-й выбранной строки с листинга файла . . . . .                          | 18 |
| 4.14 | Объяснение 3-й выбранной строки с листинга файла . . . . .                          | 18 |
| 4.15 | Создание файла без одного операнда . . . . .  | 18 |
| 4.16 | Файл листинга без одного операнда . . . . .   | 19 |
| 4.17 | Листинг самостоятельного задания №1 . . . . .                                       | 19 |
| 4.18 | Продолжение листинга самостоятельного задания №1 . . . . .                          | 20 |
| 4.19 | Результаты работы программы по самостоятельному заданию №1                          | 20 |
| 4.20 | Листинг самостоятельного задания №2 . . . . .                                       | 21 |
| 4.21 | Продолжение листинга самостоятельного задания №2 . . . . .                          | 22 |
| 4.22 | Продолжение листинга самостоятельного задания №2 . . . . .                          | 23 |
| 4.23 | Результаты работы программы . . . . .   | 23 |

# Список таблиц

3.1 Типы операндов инструкции jmp . . . . . 8

# 1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

## 2 Задание

1. Создать каталог для программ лабораторной работы № 7, перейти в него и создать файл lab7-1.asm.
2. Ввести в файл lab7-1.asm текст программы из листинга 1 методического указания. Создать исполняемый файл и запустить его. Посмотреть результаты работы. Написать вывод.
3. Изменить программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Ввести текст программы в соответствии с листингом. Создать исполняемый файл и запустить его. Посмотреть результаты работы.
4. Создать файл lab7-2.asm. Ввести в него текст из листинга 3 методического указания. Создать исполняемый файл и запустить его. Проверить его работу при разных значениях В.
5. Создать файл листинга для программы из файла lab7-2.asm. Открыть его с помощью текстового редактора и изучили. Ознакомить с содержимым файла, описать любые три строки листинга.
6. Открыть файл с программой lab7-2.asm и в инструкции с двумя операндами удалили один операнд. Выполнить трансляцию с получением файла листинга. Написать вывод по заданию.

Задание для самостоятельной работы

1. Напишите программу нахождения наименьшей из 3 целочисленных переменных  $a$ ,  $b$ ,  $c$ . Значения переменных выбрать из таблицы в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу.
2. 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 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу для значений  $x$  и  $a$ .

### 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.

Команды безусловного перехода

Безусловный переход выполняется инструкцией `jmp` (от англ. `jump` – прыжок), которая включает в себя адрес перехода, куда следует передать управление:

`jmp` адрес перехода

Адрес перехода может быть либо меткой, либо адресом области памяти, в которую предварительно помещен указатель перехода. Кроме того, в качестве операнда можно использовать имя регистра, в таком случае переход будет осуществляться по адресу, хранящемуся в этом регистре.

В табл. [3.1] приведены типы операндов инструкции `jmp`.

Таблица 3.1: Типы операндов инструкции `jmp`

| Тип                | Описание                            |
|--------------------|-------------------------------------|
| операнда           |                                     |
| <code>jmp</code>   | переход на метку <code>label</code> |
| <code>label</code> |                                     |



| Тип                      |   |
|--------------------------|---|
| операнда                 | Описание  |
| <code>jmp [label]</code> | переход по адресу в памяти, помеченному меткой <code>label</code> |
| <code>jmp eax</code>     | переход по адресу из регистра <code>eax</code>                    |

### Команды условного перехода

Как отмечалось выше, для условного перехода необходима проверка какого-либо условия. В ассемблере команды условного перехода вычисляют условие перехода анализируя флаги из регистра флагов.

Флаг – это бит, принимающий значение 1 («флаг установлен»), если выполнено некоторое условие, и значение 0 («флаг сброшен») в противном случае. Флаги работают независимо друг от друга, и лишь для удобства они помещены в единый регистр — регистр флагов, отражающий текущее состояние процессора. Флаги состояния (биты 0, 2, 4, 6, 7 и 11) отражают результат выполнения арифметических инструкций, таких как ADD, SUB, MUL, DIV.

### Описание инструкции `cmp`

Инструкция `cmp` является одной из инструкций, которая позволяет сравнить операнды и выставляет флаги в зависимости от результата сравнения. Инструкция `cmp` является командой сравнения двух операндов и имеет такой же формат, как и команда вычитания:

`cmp операнд_1, операнд_2`

Команда `cmp`, так же как и команда вычитания, выполняет вычитание `операнд_2-операнд_1`, но результат вычитания никуда не записывается и единственным результатом команды сравнения является формирование флагов.

### Описание команд условного перехода.

Команда условного перехода имеет вид

`j` мнемометка перехода `label`

Мнемоника перехода связана со значением анализируемых флагов или со способом формирования этих флагов.

#### Файл листинга и его структура

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.

#### Структура листинга:

номер строки — это номер строки файла листинга (нужно помнить, что номер строки в файле листинга может не соответствовать номеру строки в файле с исходным текстом программы);

адрес — это смещение машинного кода от начала текущего сегмента;

машинный код представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности. (например, инструкция `int 80h` начинается по смещению `00000020` в сегменте кода; далее идёт машинный код, в который ассемблируется инструкция, то есть инструкция `int 80h` ассемблируется в `CD80` (в шестнадцатеричном представлении); `CD80` — это инструкция на машинном языке, вызывающая прерывание ядра);

исходный текст программы — это просто строка исходной программы вместе с комментариями (некоторые строки на языке ассемблера, например, строки, содержащие только комментарии, не генерируют никакого машинного кода, и поля «смещение» и «исходный текст программы» в таких строках отсутствуют, однако номер строки им присваивается).

## 4 Выполнение лабораторной работы

1. Создали каталог для программ лабораторной работы № 7, перешли в него и создали файл lab7-1.asm (рис. 4.1 Создание каталога и файла для выполнения лабораторной работы).

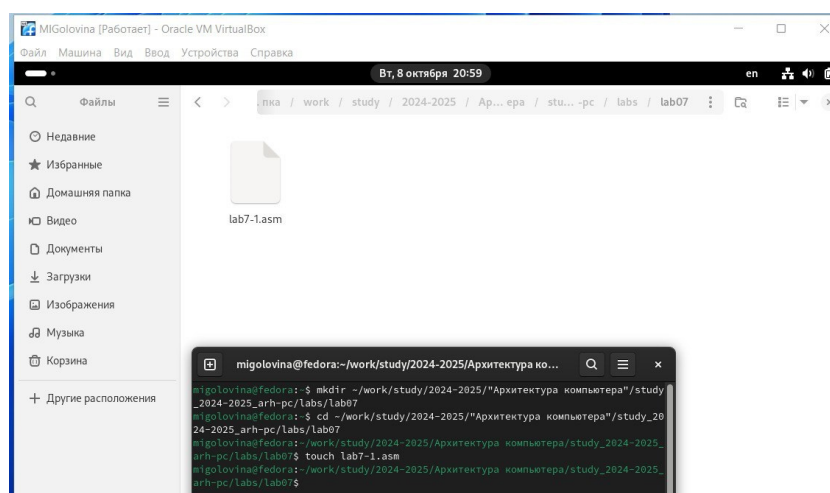
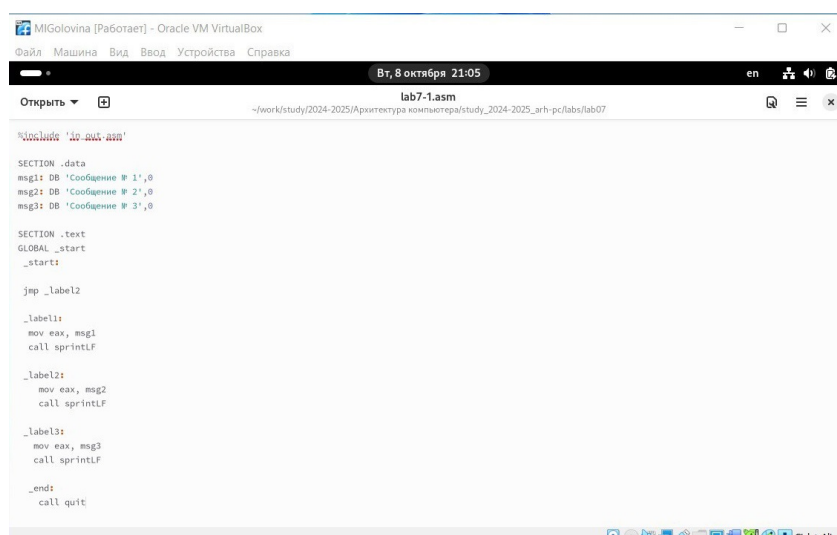


Рис. 4.1: Создание каталога и файла для выполнения лабораторной работы

2. Ввели в файл lab7-1.asm текст программы из листинга 1 (рис. 4.2 Листинг 1) методического указания. Создали исполняемый файл и запустили его. Посмотрели результаты работы (рис. 4.3 Результаты работы программы из листинга 1 методического указания).



```
%include 'io_out.asm'

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
    call sprintf

_label2:
    mov eax, msg2
    call sprintf

_label3:
    mov eax, msg3
    call sprintf

_end:
    call quit
```

Рис. 4.2: Листинг 1

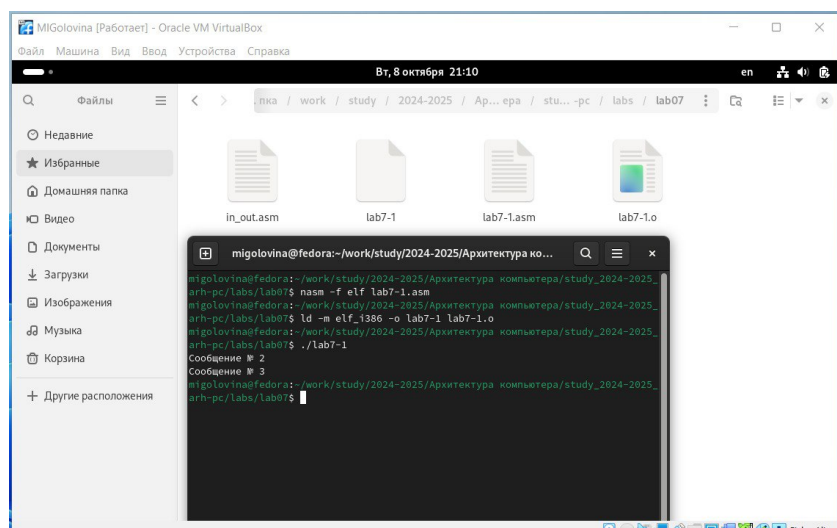
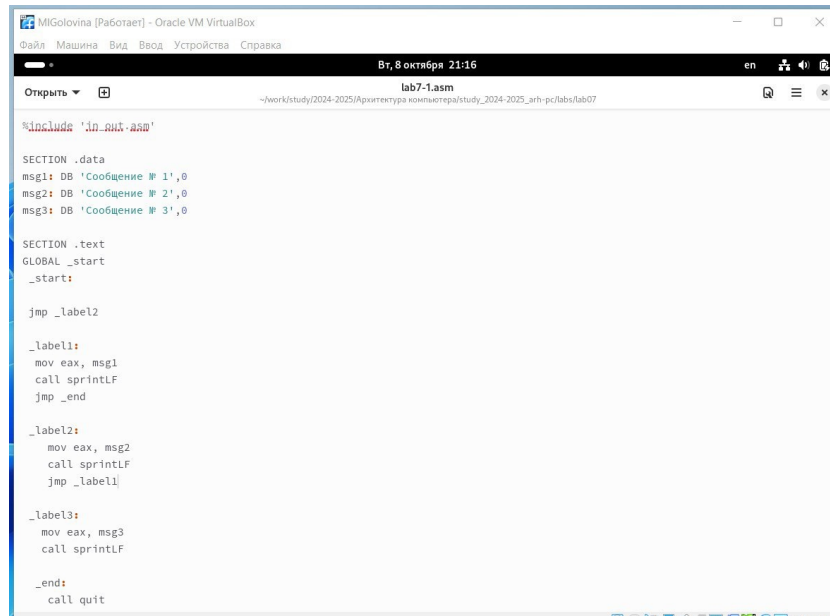


Рис. 4.3: Результаты работы программы из листинга 1 методического указания

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

3. Изменили программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст

программы после вывода сообщения № 2 добавили инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавили инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`).



```

%include "in_out.asm"

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
call printf
jmp _end

_label2:
mov eax, msg2
call printf
jmp _label1

_label3:
mov eax, msg3
call printf

_end:
call quit
  
```

Рис. 4.4: Листинг 1 с внесенными изменениями

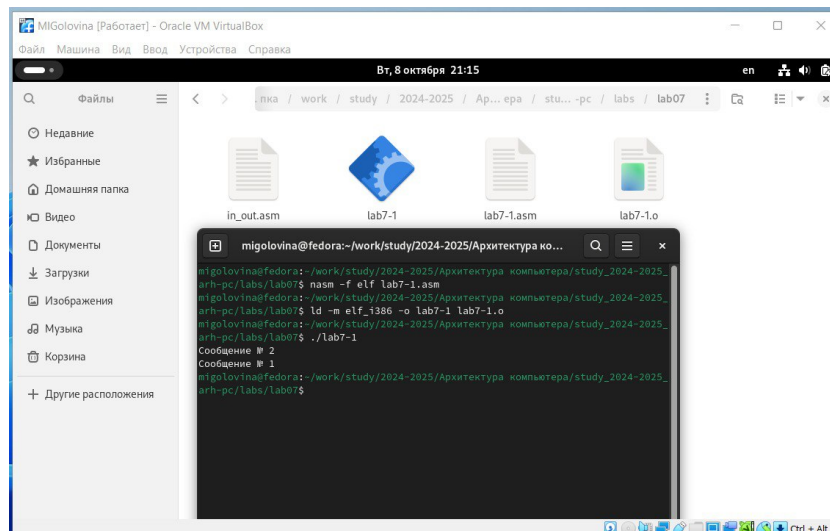
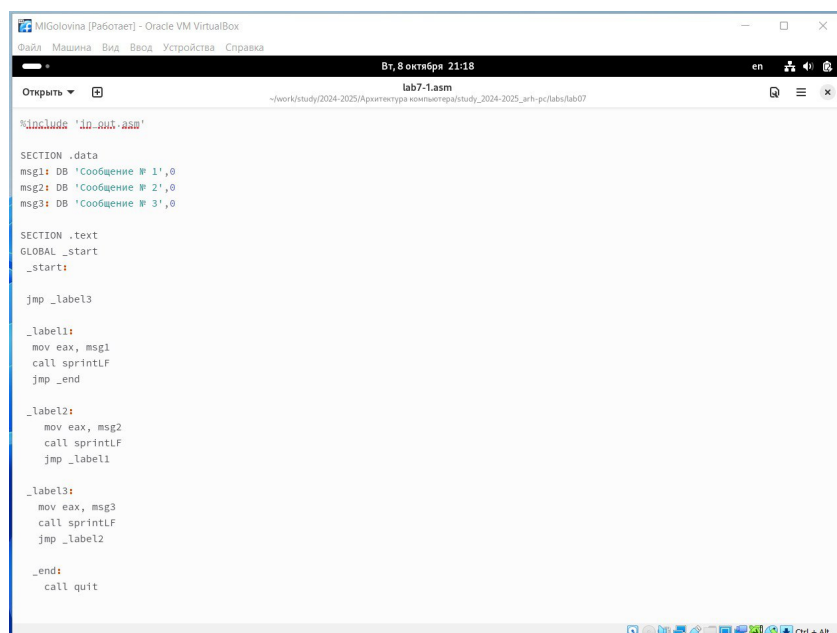


Рис. 4.5: Результаты работы Листинга 1 с внесенными изменениями

Изменили текст программы в соответствии с листингом 2 (рис.4.6 Листинг 2). Создали исполняемый файл и запустили его. Посмотрели результаты работы (рис. 4.7 Результаты работы программы из Листинга 2 методического указания).



```
%include 'in.asm'

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
call sprintf
jmp _end

_label2:
mov eax, msg2
call sprintf
jmp _label1

_label3:
mov eax, msg3
call sprintf
jmp _label2

_end:
call quit
```

Рис. 4.6: Листинг 2

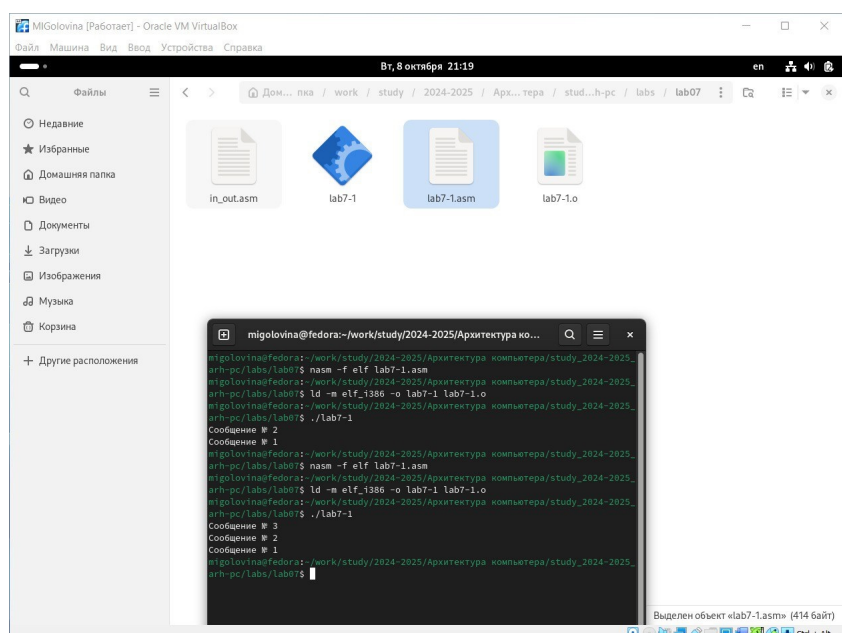
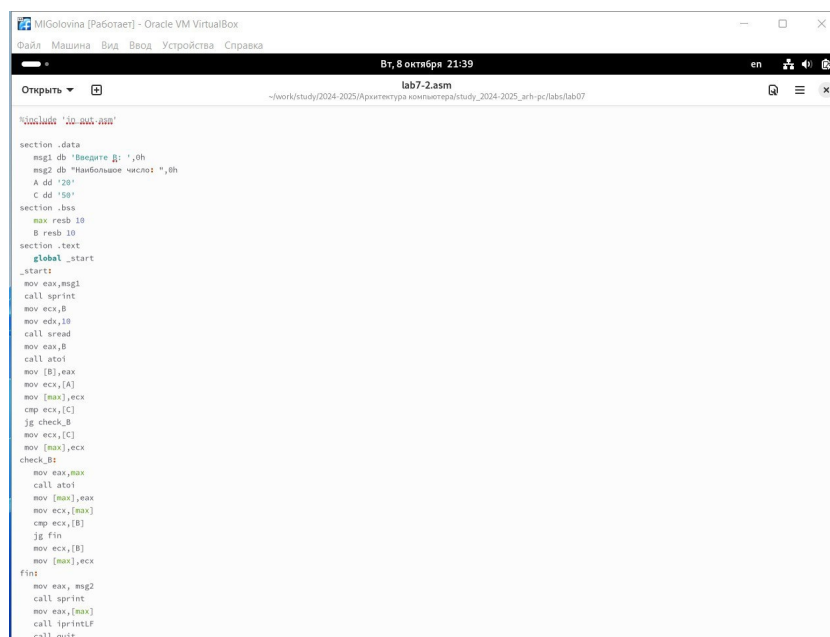


Рис. 4.7: Результаты работы программы из Листинга 2 методического указания

4. Создали файл lab7-2.asm. Ввели в него текст из листинга 3 методического указания (рис. 4.8 Листинг 3). Создали исполняемый файл и запустили его. Проверили его работу при разных значениях В (рис. 4.9 Результаты работы программы из листинга 3 методического указания).



```
include 'io.inc.asm'

section .data
msg1 db 'Введите B: ',0h
msg2 db 'Наибольшее число: ',0h
A dd '28'
C dd '50'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:
mov eax,msg1
call sprintf
mov ecx,B
mov edx,10
call sread
mov eax,B
call atoi
mov [B],eax
mov ecx,[A]
mov [max],ecx
cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [max],ecx
check_B:
mov eax,max
call atoi
mov [max],eax
mov ecx,[max]
cmp ecx,[B]
jg fin
mov ecx,[B]
mov [max],ecx
fin:
mov eax,msg2
call sprintf
mov ecx,[max]
call printf
call quit
```

Рис. 4.8: Листинг 3

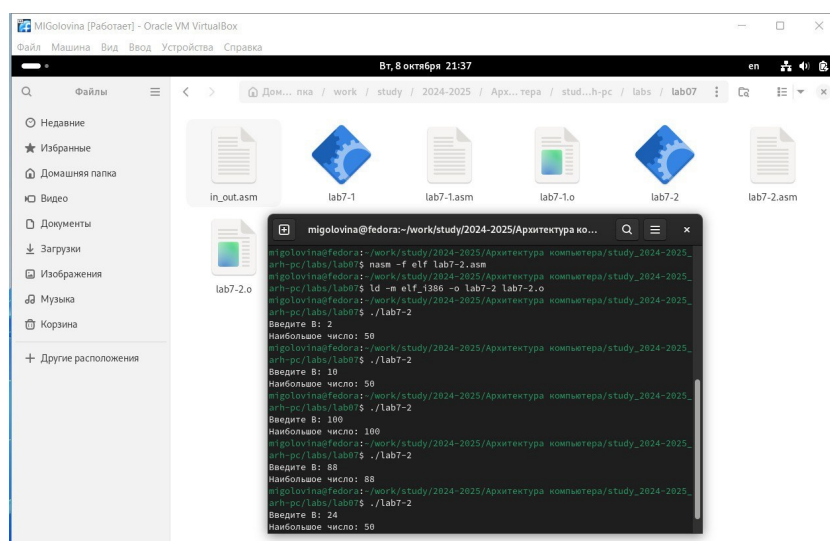


Рис. 4.9: Результаты работы программы из листинга 3 методического указания

5. Создали файл листинга для программы из файла lab7-2.asm (рис. 4.10 Создание файл листинга для программы для программы из файла lab7-2.asm). Открыли его с помощью текстового редактора (рис. 4.11 Просмотр файла lab7-2.asm с помощью текстового редактора) и изучили.



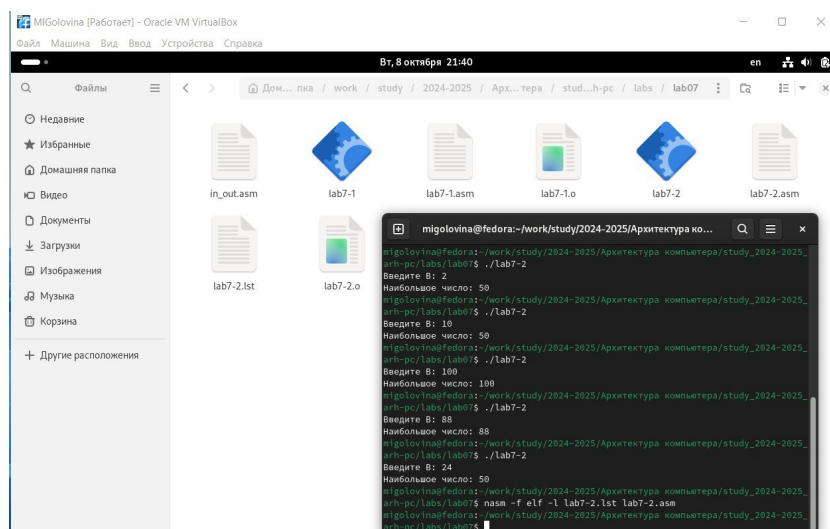


Рис. 4.10: Создание файл листинга для программы для программы из файла lab7-2.asm

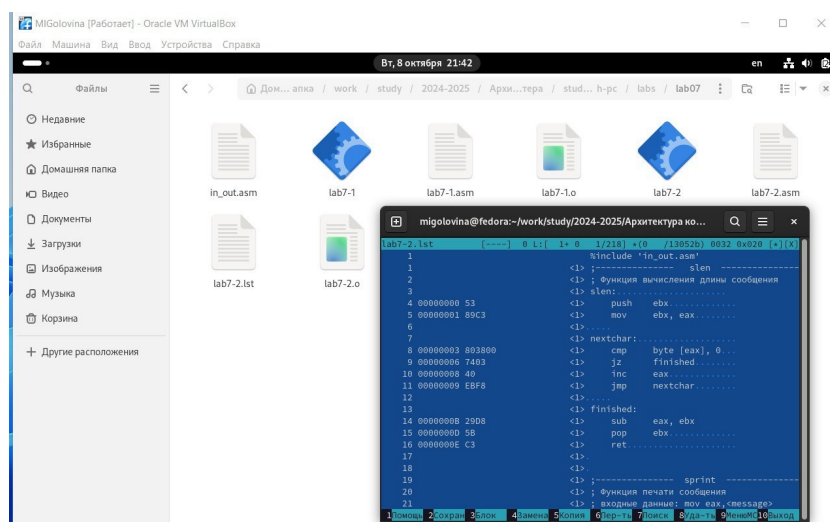


Рис. 4.11: Просмотр файла lab7-2.asm с помощью текстового редактора

Ознакомьтесь с форматом и содержимым файла:

Эта строка находится на 23 месте, ее адрес “00000101”, Машинный код- B8[0A000000], а mov eax,B - исходный текст программы, означающий, что в регистр eax мы вносим значения переменной B (рис.4.12 Объяснение 1-й выбранной строки с листинга файла).

```
23 00000101 B8[0A000000] mov eax,B
```

Рис. 4.12: Объяснение 1-й выбранной строки с листинга файла

Эта строка находится на 37 месте, ее адрес “0000012F”, Машинный код - E868FFFFFF, а call atoi - исходный текст программы, означающий, что символ лежащий в строке выше переводится в число (рис.4.13 Объяснение 2-й выбранной строки с листинга файла).

```
37 0000012F E868FFFFFF call atoi
```

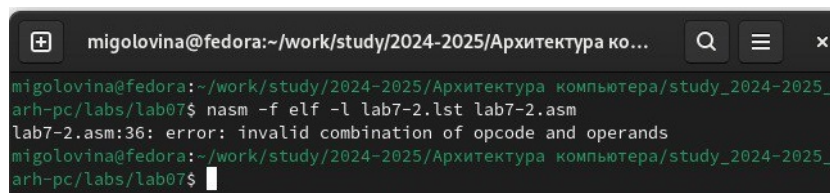
Рис. 4.13: Объяснение 2-й выбранной строки с листинга файла

Эта строка находится на 48 месте, ее адрес “0000015D”, Машинный код - A1[00000000], а mov eax,[max] - исходный текст программы, означающий что число хранившееся в переменной max записывается в регистр eax (рис. 4.14 Объяснение 3-й выбранной строки с листинга файла).

```
48 0000015D A1[00000000] mov eax,[max]
```

Рис. 4.14: Объяснение 3-й выбранной строки с листинга файла

6. Открыли файл с программой lab7-2.asm и в инструкции с двумя операндами удалили один операнд. Выполнили трансляцию с получением файла листинга (рис. 4.15 Создание файла без одного операнда).



```
migolovina@fedora:~/work/study/2024-2025/Архитектура ко...
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_
arh-pc/labs/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:36: error: invalid combination of opcode and operands
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_
arh-pc/labs/lab07$
```

Рис. 4.15: Создание файла без одного операнда

В файле листинга показывает, где именно ошибка и с чем она связана (рис. 4.16 Файл листинга без одного операнда).

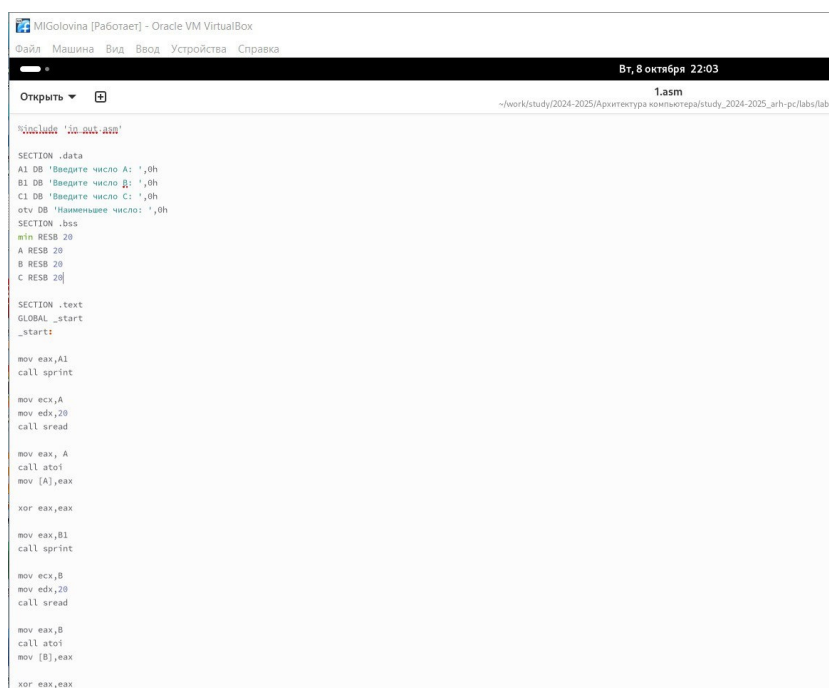
```
36                                     mov eax
36      *****
                                     error: invalid combination of opcode and operands
```

Рис. 4.16: Файл листинга без одного операнда

### Задание для самостоятельной работы

1. Напишите программу нахождения наименьшей из 3 целочисленных переменных a, b, c. Значения переменных выбрать из таблицы в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу.

При выполнении лабораторной работы №6 у меня получился вариант 11, соответственно для варианта №11 значение переменных: a=21, b=28, c=34.



```
%include 'in_out.asm'

SECTION .data
A1 DB 'Введите число A: ',0h
B1 DB 'Введите число B: ',0h
C1 DB 'Введите число C: ',0h
str DB 'Наименьшее число: ',0h

SECTION .bss
min RESB 20
A RESB 20
B RESB 20
C RESB 20

SECTION .text
GLOBAL _start
_start:

mov eax,A1
call sprint

mov ecx,A
mov edx,20
call sread

mov eax,A
call atoi
mov [A],eax

xor eax,eax

mov eax,B1
call sprint

mov ecx,B
mov edx,20
call sread

mov eax,B
call atoi
mov [B],eax

xor eax,eax
```

Рис. 4.17: Листинг самостоятельного задания №1

```

mov ecx, [A]
mov [min],ecx
mov ecx,[min]

cmp ecx,[B]
jl check_C
mov ecx,[B]
mov [min],ecx

check_C:
mov eax,C1
call sprint

mov ecx,C
mov edx,[B]
call read

mov eax,C
call atoi
mov [C],eax

xor eax,eax

mov ecx,[min]
cmp ecx,[C]
jl fin
mov ecx,[C]
mov [min],ecx

fin:
mov eax,0x0
call sprint
mov eax,[min]
call iprintf
call quit

```

Рис. 4.18: Продолжение листинга самостоятельного задания №1

```

migllovina@fedora:~/work/study/2024-2025/Архитектура ко...
%incT
+
migllovina@fedora:~/work/study/2024-2025/Архитектура ко...
touch 1.asm
migllovina@fedora:~/work/study/2024-2025/Архитектура ко...
nasm -f elf 1.asm
migllovina@fedora:~/work/study/2024-2025/Архитектура ко...
ld -m elf_i386 -o 1 1.o
migllovina@fedora:~/work/study/2024-2025/Архитектура ко...
./1
min: Введите число A: 21
A RES: Введите число B: 28
B RES: Введите число C: 34
C RES: Наименьшее число: 21
migllovina@fedora:~/work/study/2024-2025/Архитектура ко...
arh-pc/labs/lab07$

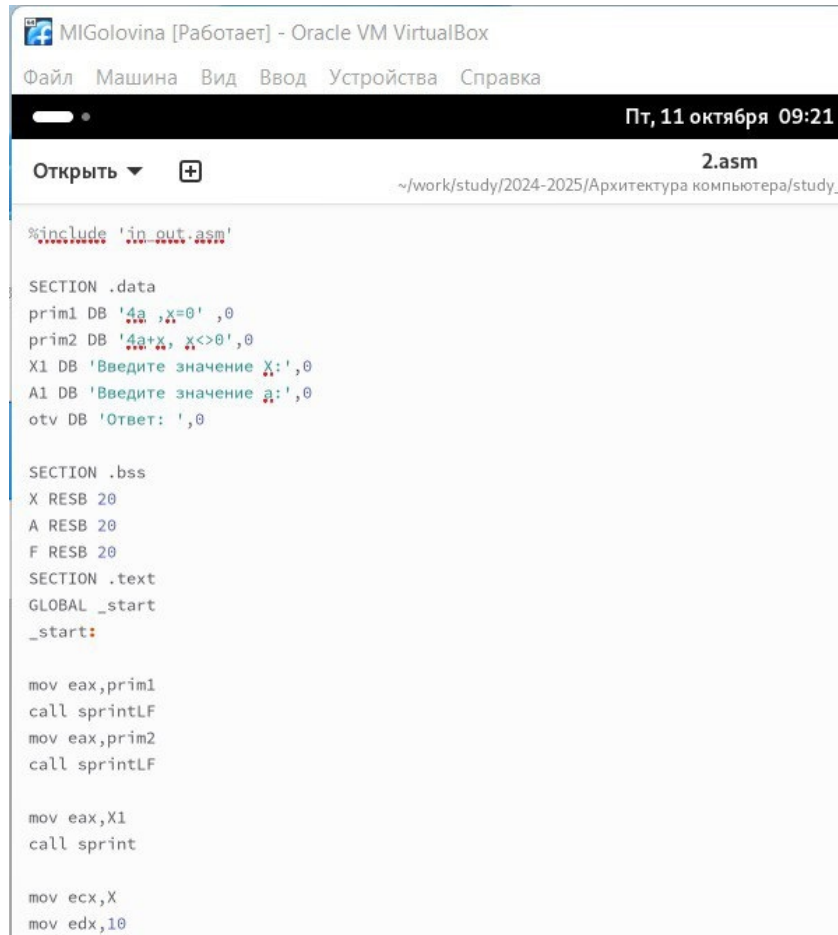
```

Рис. 4.19: Результаты работы программы по самостоятельному заданию №1

2. 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 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы №

6. Создайте исполняемый файл и проверьте его работу для значений  $x$  и  $a$ .

При выполнении лабораторной работы №6 у меня получился вариант 11, соответственно для варианта №11 была написана программа.



```
%include 'in_out.asm'

SECTION .data
prim1 DB '4a * x = 0', 0
prim2 DB '4a + x, x < 0', 0
X1 DB 'Введите значение x:', 0
A1 DB 'Введите значение a:', 0
otv DB 'Ответ: ', 0

SECTION .bss
X RESB 20
A RESB 20
F RESB 20
SECTION .text
GLOBAL _start
_start:

mov eax, prim1
call sprintf
mov eax, prim2
call sprintf

mov eax, X1
call sprintf

mov ecx, X
mov edx, 10
```

Рис. 4.20: Листинг самостоятельного задания №2

```
call sread

mov eax,X
call atoi
mov [X],eax

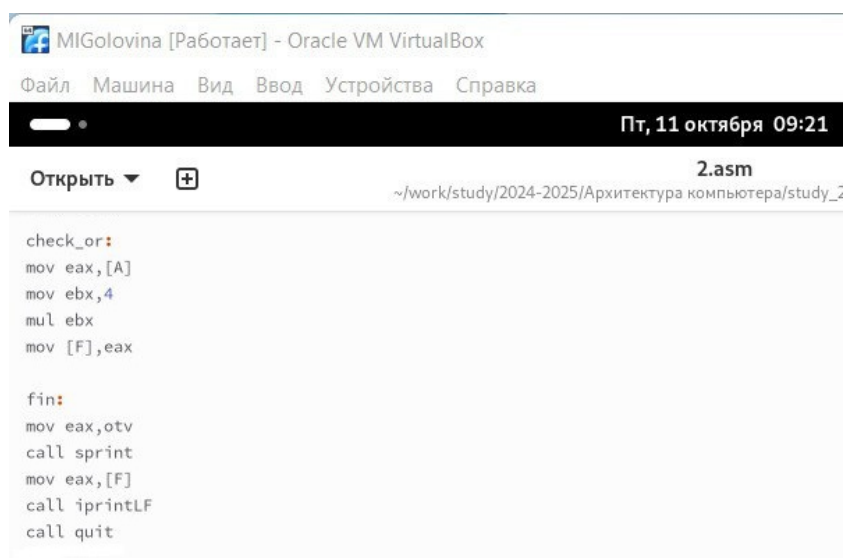
mov eax,A1
call sprint

mov ecx,A
mov edx,10
call sread

mov eax,A
call atoi
mov [A],eax

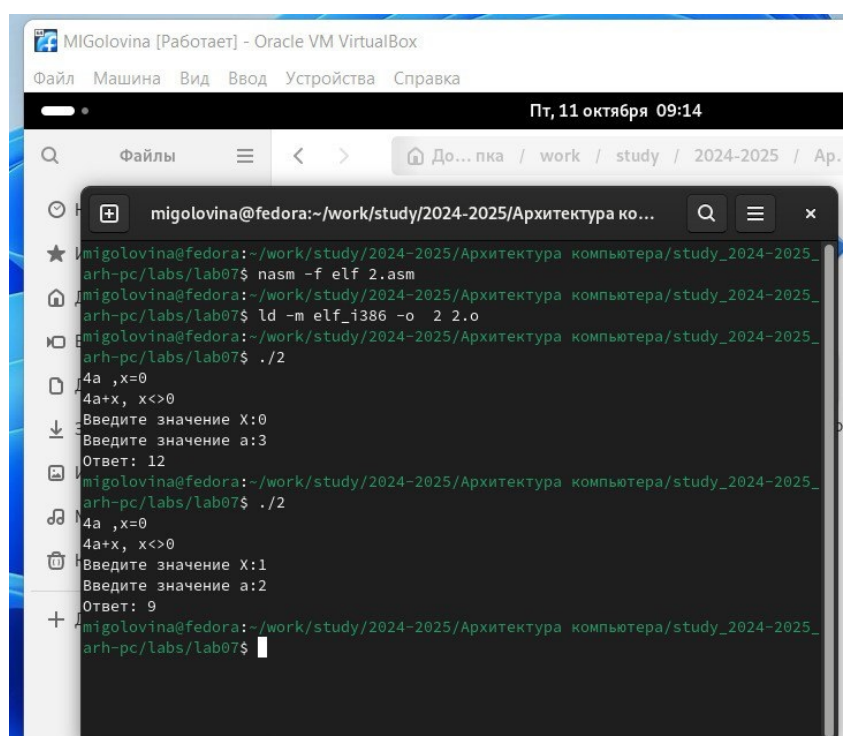
mov ecx,[X]
mov [F],ecx
|
mov eax,[A]
mov ebx,4
mul ebx
mov ecx, [X]
add eax,ecx
mov [F],eax
jmp fin
```

Рис. 4.21: Продолжение листинга самостоятельного задания №2



```
check_or:  
mov eax,[A]  
mov ebx,4  
mul ebx  
mov [F],eax  
  
fin:  
mov eax,otv  
call sprint  
mov eax,[F]  
call iprintLF  
call quit
```

Рис. 4.22: Продолжение листинга самостоятельного задания №2



```
migolovina@fedora:~/work/study/2024-2025/Архитектура ко...  
★ migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_  
arh-pc/labs/lab07$ nasm -f elf 2.asm  
★ migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_  
arh-pc/labs/lab07$ ld -m elf_i386 -o 2 2.o  
★ migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_  
arh-pc/labs/lab07$ ./2  
4a ,x=0  
4a+x, x<>0  
Введите значение X:0  
Введите значение a:3  
Ответ: 12  
★ migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_  
arh-pc/labs/lab07$ ./2  
4a ,x=0  
4a+x, x<>0  
Введите значение X:1  
Введите значение a:2  
Ответ: 9  
★ migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_  
arh-pc/labs/lab07$
```

Рис. 4.23: Результаты работы программы

## 5 Выводы

Изучили команды условного и безусловного переходов. Приобрели навыки написания программ с использованием переходов. Познакомились с назначением и структурой файла листинга.



## Список литературы

1. GDB: The GNU Project Debugger. — URL:<https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: [http://www.stolyarov.info/books/asm\\_unix](http://www.stolyarov.info/books/asm_unix).

15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).