

Отчет лабораторной работе №7

Дисциплина: архитектура компьютера

Зайцева Ульяна Владимировна

Содержание

1	Цель работы	1
2	Задание	1
3	Теоретическое введение.....	1
4	Выполнение лабораторной работы.....	2
5	Выводы.....	9

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. Реализация переходов в NASM
2. Изучение структуры файлы листинга.
3. Задания для самостоятельной работы.

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

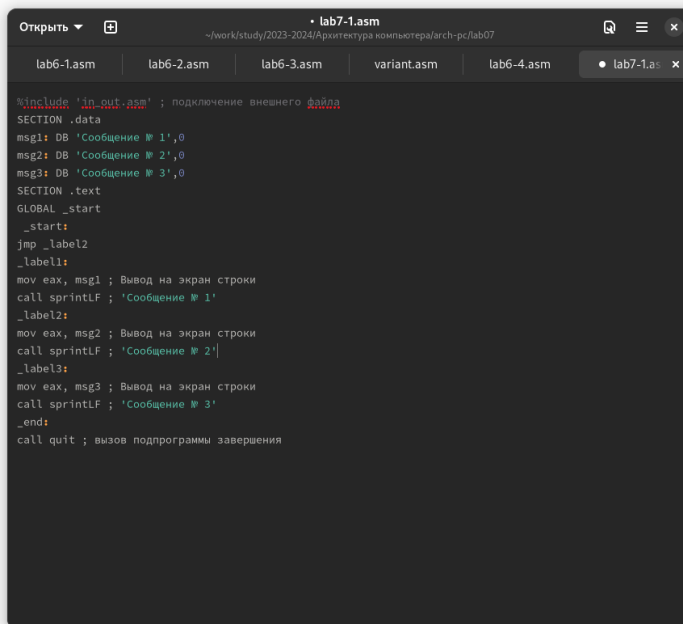
Безусловный переход выполняется инструкцией jmp (от англ. jump – прыжок), которая включает в себя адрес перехода, куда следует передать управление: jmp Адрес перехода может быть либо меткой, либо адресом области памяти, в которую предва- рительно помещен указатель перехода. Кроме того, в качестве операнда

можно использовать имя регистра, в таком случае переход будет осуществляться по адресу, хранящемуся в этом регистре

4 Выполнение лабораторной работы

1. Реализация переходов в NASM

Создаю каталог для программ лабораторной работы №7, создаю файл lab7-1.asm. Ввожу в файл текст программы из листинга 7.1. (рис. ??).



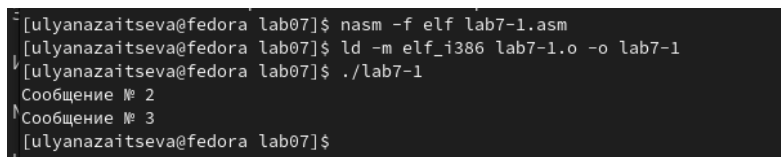
```
lab7-1.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07

lab6-1.asm lab6-2.asm lab6-3.asm variant.asm lab6-4.asm lab7-1.asm

#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
```

Ввод листинга

Создаю исполняемый файл и запускаю его.(рис. ??)

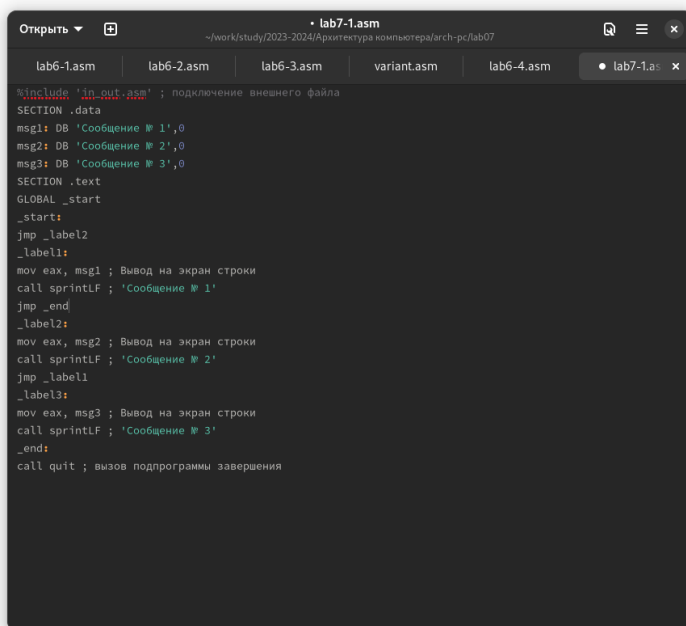


```
[ulyanazaitseva@fedora lab07]$ nasm -f elf lab7-1.asm
[ulyanazaitseva@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1
[ulyanazaitseva@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 3
[ulyanazaitseva@fedora lab07]$
```

Выполнение

Использование инструкции jmp _label2 изменяет порядок исполнения инструкций и позволяет выполнить инструкции начиная с _label2, пропустив вывод первого сообщения.

Меняю программу, чтобы она выводила сначала “Сообщение № 2”, потом “Сообщение № 1” и завершала работу. Изменяю текст программы в соответствии с листингом 7.2.(рис. ??).



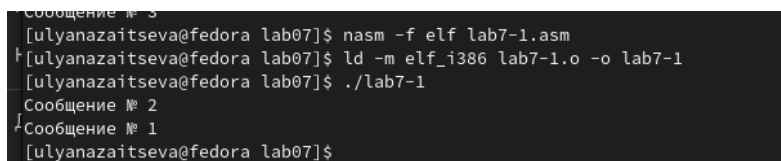
```
lab7-1.asm
~/.work/study/2023-2024/Архитектура компьютера/arch-pc/lab07

lab6-1.asm lab6-2.asm lab6-3.asm variant.asm lab6-4.asm lab7-1.asm x

%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
```

Изменяю текст программы

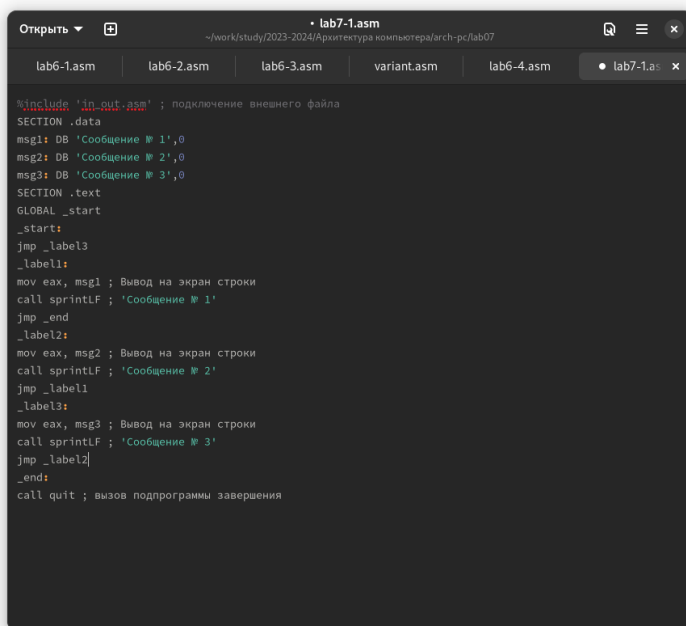
Создаю исполняемый файл и проверяю его работу.(рис. ??)



```
Сообщение № 3
[ulyanazaitseva@fedora lab07]$ nasm -f elf lab7-1.asm
[ulyanazaitseva@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1
[ulyanazaitseva@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 1
[ulyanazaitseva@fedora lab07]$
```

Проверка работы

После меняю текст программы, добавив в начале программы `jmp _label3`, `jmp _label2` в конце метки `jmp _label3`, `jmp _label1` добавляю в конце метки `jmp _label2`, и добавляю `jmp _end` в конце метки `jmp _label1`, (рис. ??).



```
lab7-1.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07

lab6-1.asm lab6-2.asm lab6-3.asm variant.asm lab6-4.asm lab7-1.asm x

#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
```

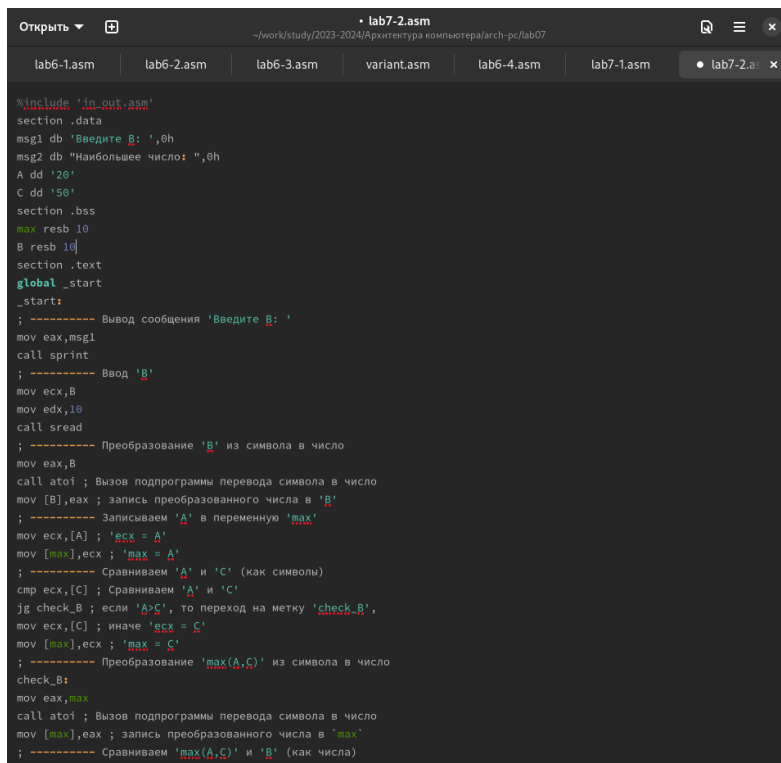
Изменяю текст

Создаю исполняемый файл и проверяю его работу.(рис. ??)

```
[ulyanazaitseva@fedora lab07]$ nasm -f elf lab7-1.asm
[ulyanazaitseva@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1
[ulyanazaitseva@fedora lab07]$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[ulyanazaitseva@fedora lab07]$
```

Вывод

Создаю файл lab7-2.asm. Ввожу текст программы из листинга 7.3 (рис. ??)



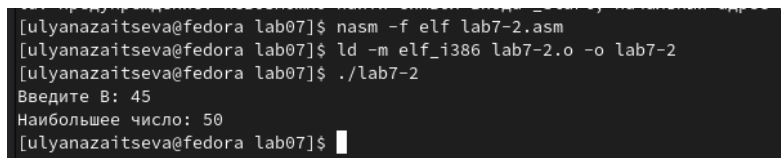
```
lab7-2.asm
~\work\study\2023-2024\Архитектура компьютера\arch-pc\lab07

lab6-1.asm lab6-2.asm lab6-3.asm variant.asm lab6-4.asm lab7-1.asm lab7-2.a x

%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
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)
```

Ввод текста листинга

Создаю исполняемый файл и проверьте его работу. (рис. ??)

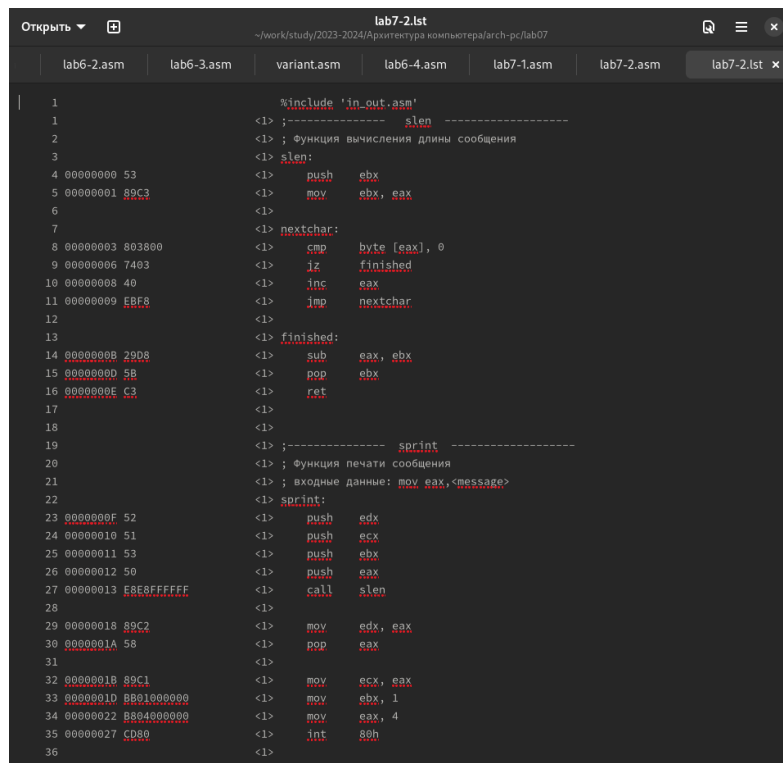


```
[ulyanazaitseva@fedora lab07]$ nasm -f elf lab7-2.asm
[ulyanazaitseva@fedora lab07]$ ld -m elf_i386 lab7-2.o -o lab7-2
[ulyanazaitseva@fedora lab07]$ ./lab7-2
Введите B: 45
Наибольшее число: 50
[ulyanazaitseva@fedora lab07]$
```

Проверка работы

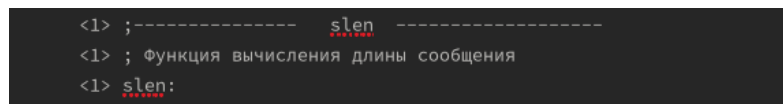
2. Изучение структуры файлы листинга.

Создаю файл листинга для программы из файла lab7-2.asm. Открываю файл листинга lab7-2.lst с помощью текстового редактора.(рис. ??)



Изучение содержимого

Рассмотрим 3 строки(рис. ??)



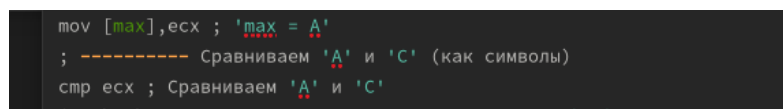
Рассматриваемые строки

“; —slen—” - комментарий к коду с названием последующей функции

“; Функция вычисления длины сообщения” - комментарий к коду, не имеющий адреса и машинного кода.

“slen” - название функции, не имеет адреса и машинного кода.

Открываю файл с программой lab7-2.asm и в выбранной мной инструкции с двумя операндами удаляю выделенный операнд.(рис. ??)



Изменение текста листинга

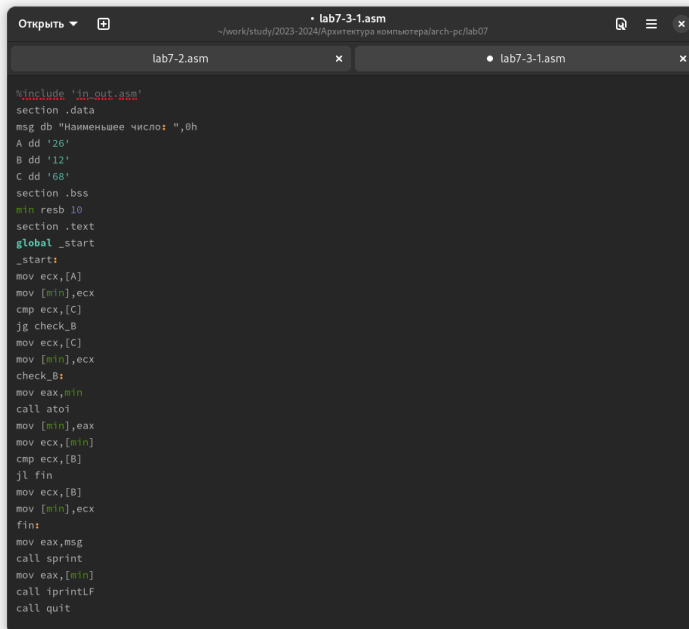
На выходе получаю ошибку.Инструкция mov не может работать, имея только один операнд.(рис. ??)

```
[ulyanazaitseva@fedora ~]$ cd ~/work/study/2023-2024/Архитектура комп
[ulyanazaitseva@fedora lab07]$ nasm -f elf lab7-2.asm
lab7-2.asm:28: error: invalid combination of opcode and operands
[ulyanazaitseva@fedora lab07]$
```

Ошибка

3. Задания для самостоятельной работы.

Пишу программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Мой вариант - 17, поэтому мои значения - 26, 12 и 68 (рис. ??)



```
lab7-3-1.asm
~work/study/2023-2024/Архитектура компьютера/arch-pc/lab07

lab7-2.asm x lab7-3-1.asm x

%include "in_out.asm"
section .data
msg db "Наименьшее число: ",0h
A dd 26
B dd 12
C dd 68
section .bss
min resb 10
section .text
global _start
_start:
mov ecx,[A]
mov [min],ecx
cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [min],ecx
check_B:
mov ecx,[min]
cmp ecx,[B]
jl fin
mov ecx,[B]
mov [min],ecx
fin:
mov eax,msg
call sprint
mov eax,[min]
call iprintLF
call quit
```

Текст программы

```
%include 'in_out.asm' section .data msg db "Наименьшее число:",0h A dd 26 B dd 12 C dd
68 section .bss min resb 10 section .text global _start _start: mov ecx,[A] mov [min],ecx cmp
ecx,[C] jl check_B mov ecx,[C] mov [min],ecx check_B: mov ecx,[min] cmp ecx,[B] jl fin mov
ecx,[B] mov [min],ecx fin: mov eax,msg call sprint mov eax,[min] call iprintLF call quit
```

Проверяю его работу(рис. ??)

```
[ulyanazaitseva@fedora lab07]$ nasm -f elf lab7-3-1.asm
[ulyanazaitseva@fedora lab07]$ ld -m elf_i386 lab7-3-1.o -o lab7-3-1
[ulyanazaitseva@fedora lab07]$ ./lab7-3-1
Наименьшее число: 12
[ulyanazaitseva@fedora lab07]$
```

Проверка работы программы

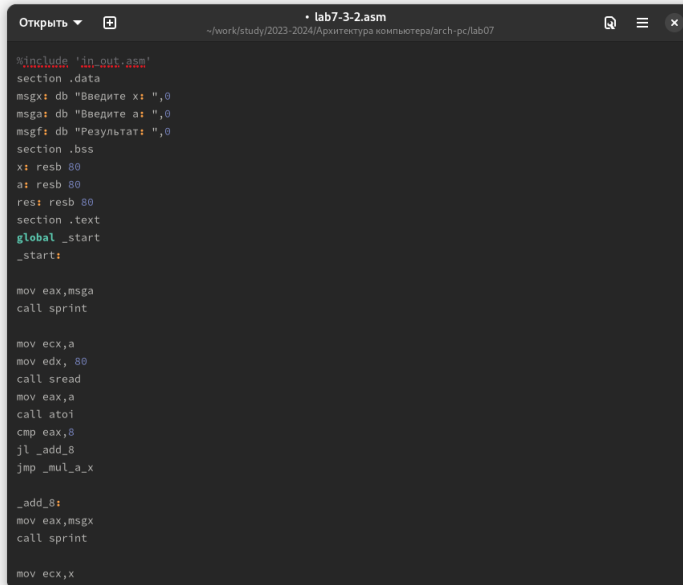
Программа работает исправно!

Пишу программу, которая для введенных с клавиатуры значений x и a вычисляет значение и выводит результат вычислений заданной для моего варианта функции $f(x)$:

$a + 8$, если $a < 8$

$a * x$, если $a \geq 8$

(рис. ??).



```
%include 'in_out.asm'
section .data
msgx: db "Введите x:",0
msga: db "Введите a:",0
msgf: db "Результат:",0
section .bss
x: resb 80
a: resb 80
res: resb 80
section .text
global _start
_start:

mov eax,msga
call sprint

mov ecx,a
mov edx, 80
call sread
mov eax,a
call atoi
cmp eax,8
jl _add_8
jmp _mul_a_x

_add_8:
mov eax,msgx
call sprint

mov ecx,x
mov edx, 80
call sread
mov eax,x
call atoi
add eax, 8
jmp _end

_mul_a_x:
mov edx, a
mov eax, x
mul edx

_end:
mov ecx, eax
mov eax, msgf
call sprint
mov eax, ecx
call iprintLF
call quit
```

Текст программы

```
%include 'in_out.asm' section .data msgx: db "Введите x:",0 msga: db "Введите a:",0 msgf: db "Результат:",0 section .bss x: resb 80 a: resb 80 res: resb 80 section .text global _start _start:
```

```
mov eax,msga call sprint
```

```
mov ecx,a mov edx, 80 call sread mov eax,a call atoi cmp eax,8 jl _add_8 jmp _mul_a_x
```

```
_add_8: mov eax,msgx call sprint
```

```
mov ecx,x mov edx, 80 call sread mov eax,x call atoi
```

```
add eax, 8 jmp _end
```

```
_mul_a_x: mov edx, a mov eax, x mul edx
```

```
_end: mov ecx, eax mov eax, msgf call sprint mov eax, ecx call iprintLF call quit
```

Проверяю его работу(рис. ??)


```
[ulyanazaitseva@fedora lab07]$ cd ..  
[ulyanazaitseva@fedora arch-pc]$ cd ~/work/study/2023-2024/'Архитектура компьюте  
ра'/arch-pc/lab07  
[ulyanazaitseva@fedora lab07]$ nasm -f elf lab7-3-2.asm  
[ulyanazaitseva@fedora lab07]$ ld -m elf_i386 -o lab7-3-2 lab7-3-2.o  
[ulyanazaitseva@fedora lab07]$ ./lab7-3-2  
Введите a: 4  
Введите x: 3  
Результат: 11  
[ulyanazaitseva@fedora lab07]$
```

Проверка работы

Ответ верный.

5 Выводы

Во время выполнения лабораторной работы я изучила команды условного и безусловного переходов, приобрела навыки написания программ с использованием переходов, познакомилась с назначением и структурой файла листинга.