Отчёт по лабораторной работе №7

Дисциплина: архитектура компьютеров

Симонова Виктория Игоревна

Содержание

1 Цель работы **5**

2 Задание **6**

3 Теоретическое введение **7**

4 Выполнение лабораторной работы **8**
4.1 Реализация переходов в NASM 8
4.2 Изучение структуры файла листинга 13
4.3 Задание для самостоятельной работы 15

5 Выводы **21**

Список литературы **22**

Список иллюстраций

4.1 Создаю папку и файл 8
4.2 Изменяю файл 9
4.3 Запускаю исполняемый файл 9
4.4 Выод исполняемого файла 9
4.5 Изменяю кодом из листинга 10
4.6 Вывод изменённого файла 10
4.7 Изменяю свой кодом 11
4.8 Вывод изменённого файла 11
4.9 Создаю файл 11
4.10 Вношу текст программы в файл 12
4.11 Вывод сравнения 13
4.12 Создаю файл листинга 13
4.13 Открываю файл 14
4.14 Строки файла листинга 14
4.15 Убераю операнд 15
4.16 Ошибка трансляции 15
4.17 Код программы 16
4.18 Вывод наименьшего числа 16
4.19 Программа для вычисления функции 18
4.20 Считаю выражение 19

Список таблиц

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 файла листинга.
3. Задание 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.

-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 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. -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.

4 Выполнение лабораторной работы

4.1 Реализация переходов в NASM

Создаю каталог lab07.asm для файлов лабораторной работы и файл lab7-1(рис. [4.1]).

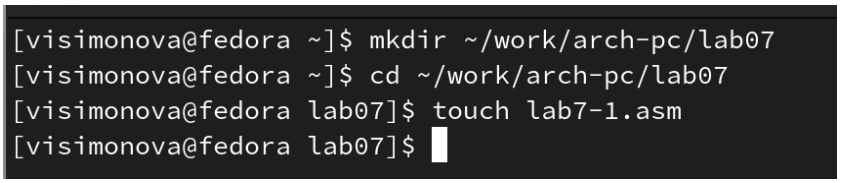

Рис. 4.1: Создаю папку и файл

Ввожу в файл lab7-1.asm пример программы с импользованием инструкции jmp (рис. [4.2]).

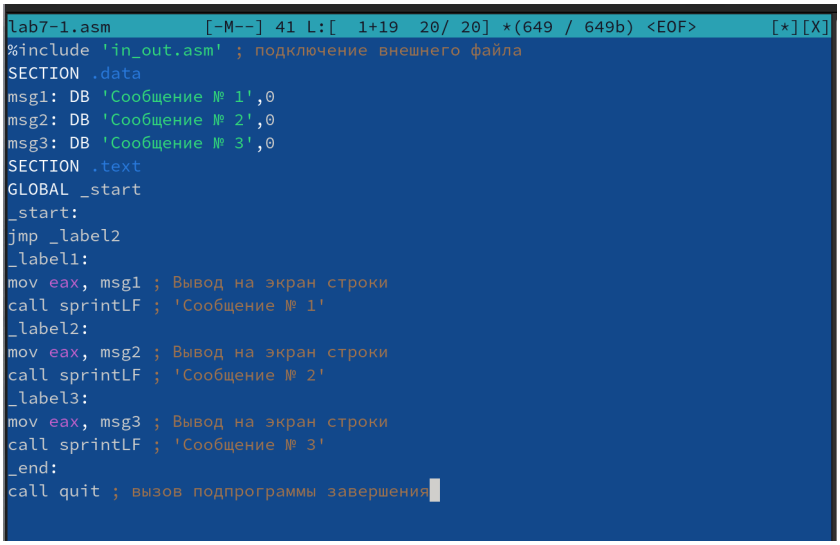

Рис. 4.2: Изменяю файл

Создаю и запускаю исполняемый файл (рис. [4.3]).

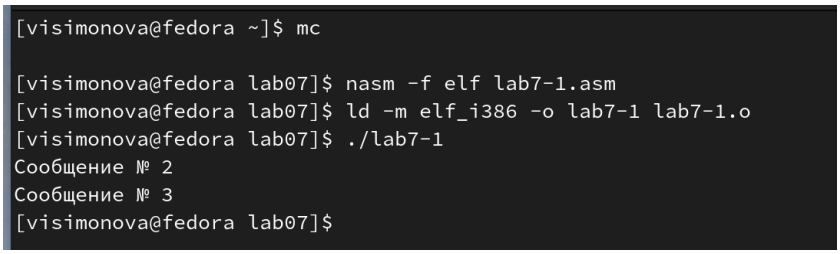

Рис. 4.3: Запускаю исполняемый файл

Таким образом, использование инструкции jmp _label2 меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки _label2, пропустив вывод первого сообщения. (рис. [4.4]).

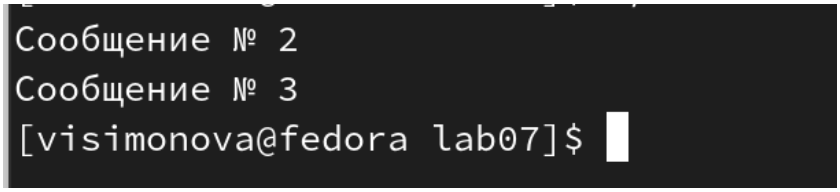

Рис. 4.4: Выод исполняемого файла

Программа выводит Сообщение № 2 ,Сообщение № 3 .Изменяю файл так,чтобы программа выводила выводила а сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу.(рис. [4.5]).

```
%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'
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'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.5: Изменяю кодом из листинга

Создаю и запускаю исполняемый файл (рис. [4.6]).

```
Сообщение № 2
Сообщение № 1
[visimonova@fedora lab07]$
```

Рис. 4.6: Вывод изменённого файла

Программа выводит Сообщение № 2 ,Сообщение № 1 .Изменяю файл так,чтобы программа выводила выводила а сначала ‘Сообщение № 3’, потом ‘Сообщение № 2’, ‘Сообщение № 1’ и завершала работу.(рис. [4.7]).

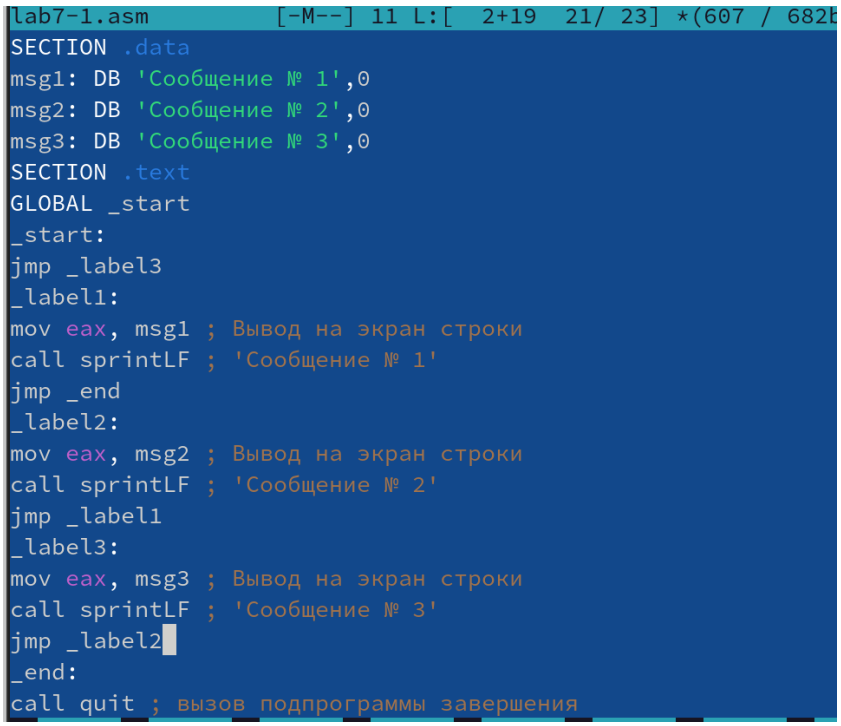

Рис. 4.7: Изменяю свой кодом

Создаю и запускаю исполняемый файл (рис. [4.8]).

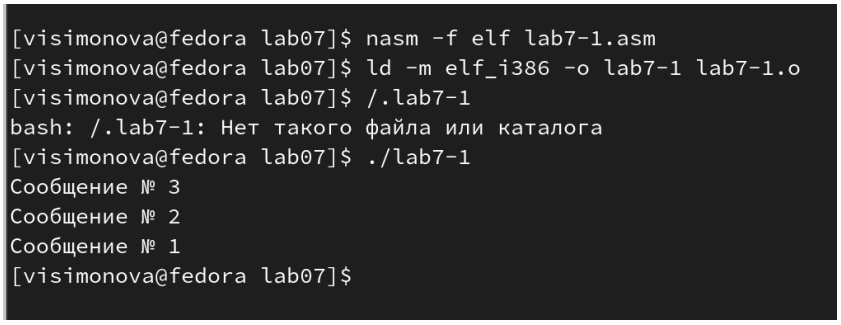

Рис. 4.8: Вывод изменённого файла

Создаю файл lab7-2.asm (рис. [4.9]).

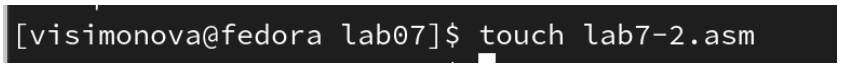

Рис. 4.9: Создаю файл

Ввожу в него код из листинга для вывода наибольшей целочисленной переменной (рис. [4.10]).

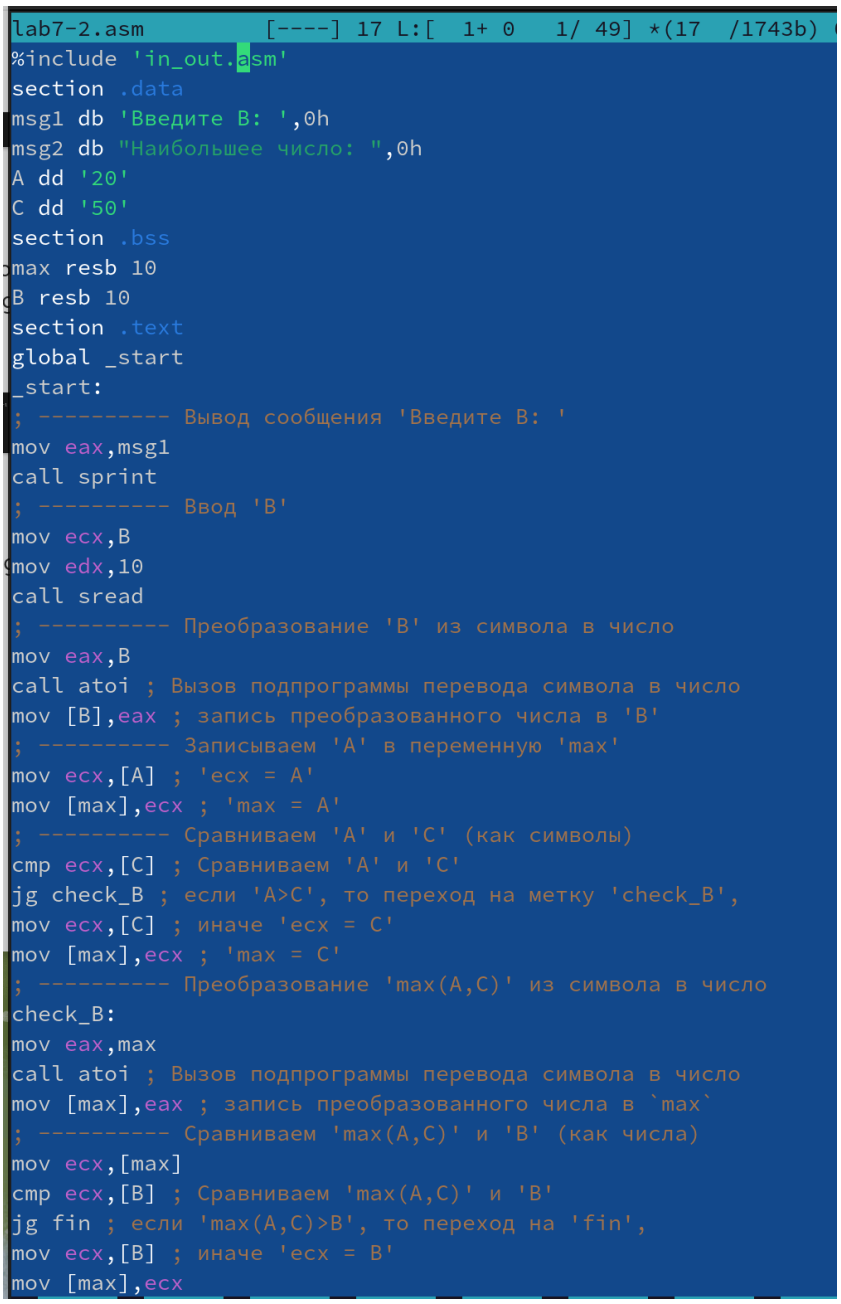

Рис. 4.10: Вношу текст программы в файл

Создаю и запускаю исполняемый файл (рис. [4.11]).

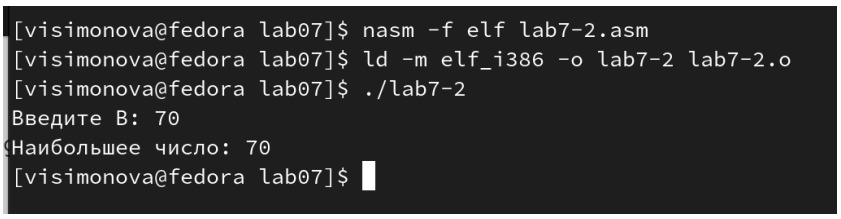

Рис. 4.11: Вывод сравнения

4.2 Изучение структуры файла листинга

Создайю файл листинга для программы из файла lab7-2.asm с помощью команды “nasm -f elf -l lab7-2.lst lab7-2.asm” (рис. [4.12]).

Рис. 4.12: Создаю файл листинга

Откройте файл листинга lab7-2.lst с помощью текстового редактора (рис. [4.13]).

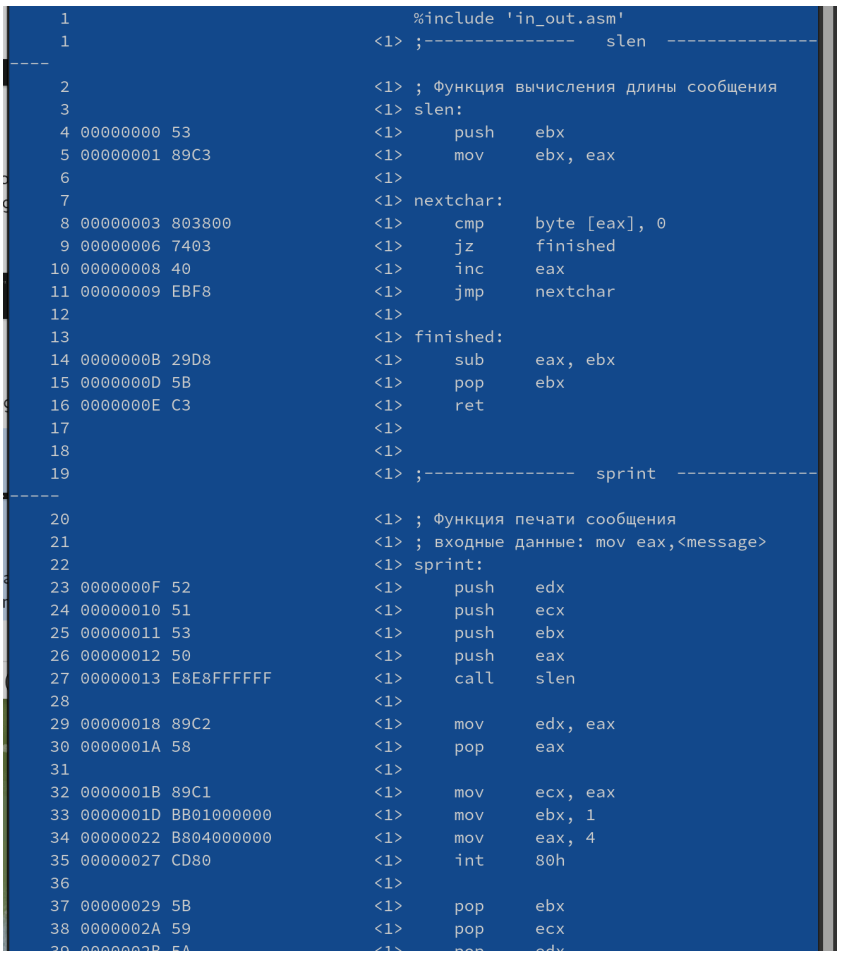

Рис. 4.13: Открываю файл

В данных трёх содержится следующее: (рис. [4.14]).

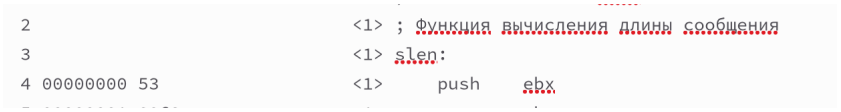

Рис. 4.14: Строки файла листинга

“2”,“3”,“4” -номера строчек.Строчки “2”,“3” не содержат адреса и машинного кода, т.к. в строке “2” записан коментарий,а в “3” название функции (в строке “4” 00000000-адрес строки, 53 машинный код).

Открываю файл с программой lab7-2.asm и в любой инструкции с двумя опе-

рандами удаляю один операнд. Выполняю трансляцию с получением файла листинга:(рис. [4.15]).

Рис. 4.15: Убераю операнд

В этом случае происходит ошибка трансляции (рис. [4.16]).

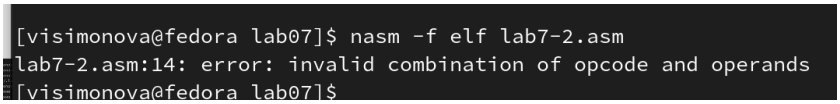

Рис. 4.16: Ошибка трансляции

Я не получаю ни одного файла,так как mov должка содержать два операнда.

4.3 Задание для самостоятельной работы

Задание 1 Создаю файл zadanie1.asm и ввожу в него код для сравнения трёх переменных и вывода наименьшего из них.(Значения переменных выбираю в соответствии с вариантом 13, полученным мной ранее)(рис. [4.17]).

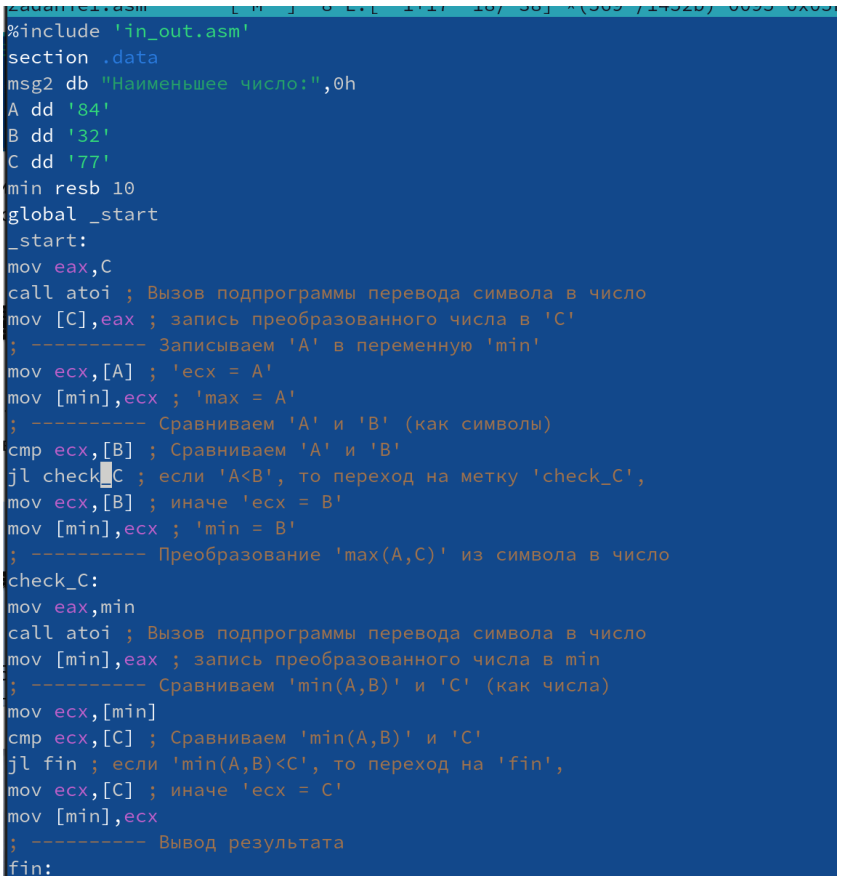

Рис. 4.17: Код программы

Создаю и запускаю исполняемый файл,программа вывела наименьшее число из трёх(из 83,32,77 вывела 32) (рис. [4.18]).

Рис. 4.18: Вывод наименьшего числа

Код к заданию 1:

```
%include 'in_out.asm'
section .data
msg2 db "Наименьшее число:",0h
```

```
A dd '84'
B dd '32'
C dd '77'
section .bss
min resb 10
section .text
global _start
_start:
mov eax,C
call atoi ; Вызов подпрограммы перевода символа в число
mov [C],eax ; запись преобразованного числа в 'C'
; ---------- Записываем 'A' в переменную 'min'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'max = A'
; ---------- Сравниваем 'A' и 'B' (как символы)
cmp ecx,[B] ; Сравниваем 'A' и 'B'
jl check_C ; если 'A<B', то переход на метку 'check_C',
mov ecx,[B] ; иначе 'ecx = B'
mov [min],ecx ; 'min = B'
; ---------- Преобразование 'max(A,C)' из символа в число
check_C:
mov eax,min
call atoi ; Вызов подпрограммы перевода символа в число
mov [min],eax ; запись преобразованного числа в min
; ---------- Сравниваем 'min(A,B)' и 'C' (как числа)
mov ecx,[min]
cmp ecx,[C] ; Сравниваем 'min(A,B)' и 'C'
jl fin ; если 'min(A,B)<C', то переход на 'fin',
mov ecx,[C] ; иначе 'ecx = C'
```

```
mov [min],ecx
; ---------- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наименьшее число: '
mov eax,[min]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход
```

Задание 2

Создаю файл zadanie2.asm и ввожу в него код для программы, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции f(x)(вариант 13) и выводит результат вычислений.(рис. [4.19]).

```
%include 'in_out.asm'
section .data
msg1 db "Введите x:",0h
msg2 db "Введите a:",0h
msg3 db "Ответ:",0h
section .bss
x resb 10
a resb 10
section .text
global _start
_start:
mov eax,msg1
call sprint
mov ecx,x
mov edx,10
call sread
mov eax,x
call atoi ; Вызов подпрограммы перевода символа в число
mov [x],eax ; запись преобразованного числа в 'x'

mov eax,msg2
call sprint
```

Рис. 4.19: Программа для вычисления функции

Создаю и запускаю исполняемый файл, в обоих вариантах введённого числа a программа работает корректно и вычисляет значения для двух соответствующих случаев функции. (рис. [4.20]).

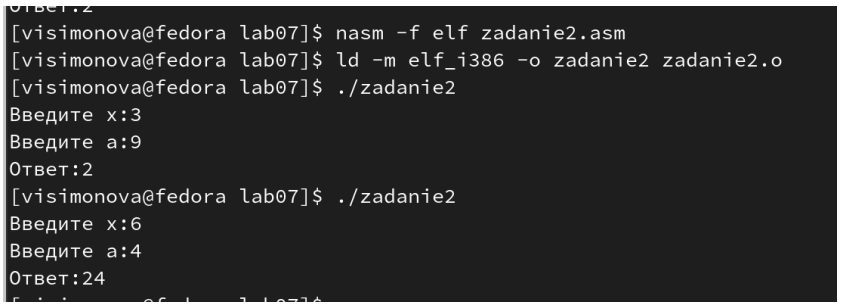

Рис. 4.20: Считаю выражение

Код к заданию 2:

```
%include 'in_out.asm'
section .data
msg1 db "Введите x:",0h
msg2 db "Введите a:",0h
msg3 db "Ответ:",0h
section .bss
x resb 10
a resb 10
section .text
global _start
_start:
mov eax,msg1
call sprint
mov ecx,x
mov edx,10
call sread
mov eax,x
call atoi ; Вызов подпрограммы перевода символа в число
mov [x],eax ; запись преобразованного числа в 'x'
```

```
mov eax,msg2
call sprint
mov ecx,a
mov edx,10
call sread
; ---------- Преобразование 'a' из символа в число
mov eax,a
call atoi ; Вызов подпрограммы перевода символа в число
mov [a],eax ; запись преобразованного числа в 'a'

mov eax,[a]
cmp eax,7
jl fin
jmp fin1
fin:
mov eax, msg3
call sprint
mov eax,[a]
mov ebx,[x]
mul ebx
call iprintLF
call quit ; Выход
fin1:
mov eax,msg3
call sprint
mov eax,[a]
add eax,-7
call iprintLF
call quit ; Выход
```

5 Выводы

Изучила команды условного и безусловного перехода, научилась писать программы с импользованием переходов . Познакомилась со структурой и назначением файла листинга.

Список литературы

Лабораторная работа №7