Отчёт по лабораторной работе №7

Дисциплина: архитектура компьютеров и операционные системы

Черная София Витальевна

Содержание

# 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.

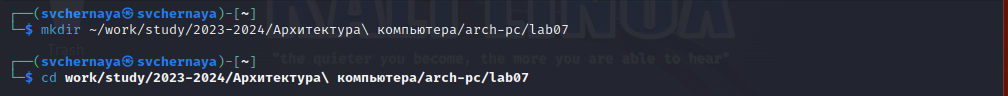
• безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

Безусловный переход выполняется инструкцией jmp. 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.

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.

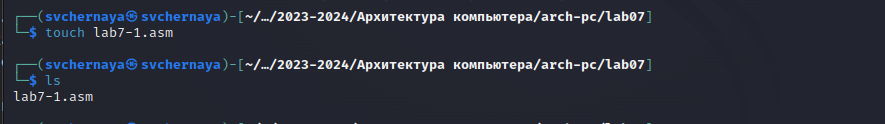
# 4 Выполнение лабораторной работы

С помощью утилиты mkdir создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №7 (рис. [??]). Перехожу в созданный каталог с помощью утилиты cd.



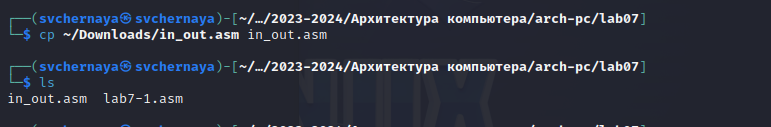
Создание директории

С помощью утилиты touch создаю файл lab7-1.asm (рис. [??]).



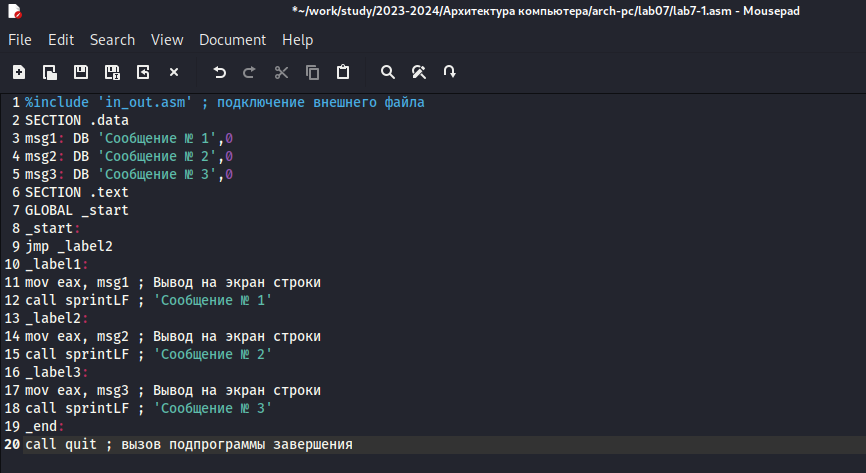
Создание файла

Копирую в текущий каталог файл in\_out.asm с помощью утилиты cp, т.к. он будет использоваться в других программах (рис. [??]).



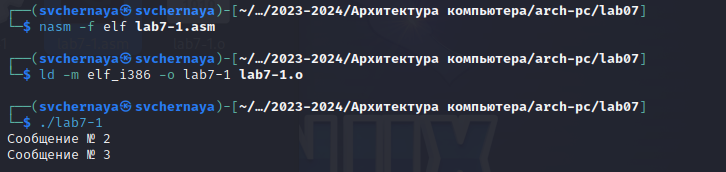
Создание копии файла

Ввожу в файл lab7-1.asm текст программы из листинга 7.1. (рис. [??]).



Ввод текста программы из листинга 7.1

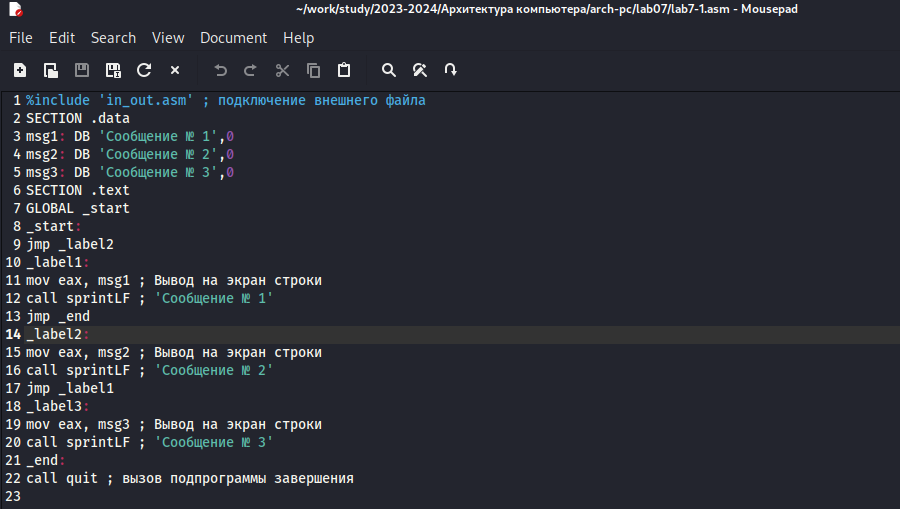
Создаю исполняемый файл и запускаю его. (рис. [??]).



Запуск программного кода

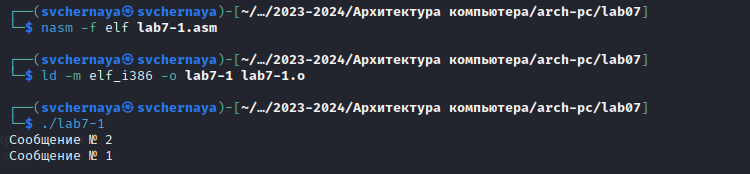
Таким образом, использование инструкции jmp \_label2 меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки \_label2, пропустив вывод первого сообщения.

Изменю программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого изменяю текст программы в соответствии с листингом 7.2. (рис. [??]).



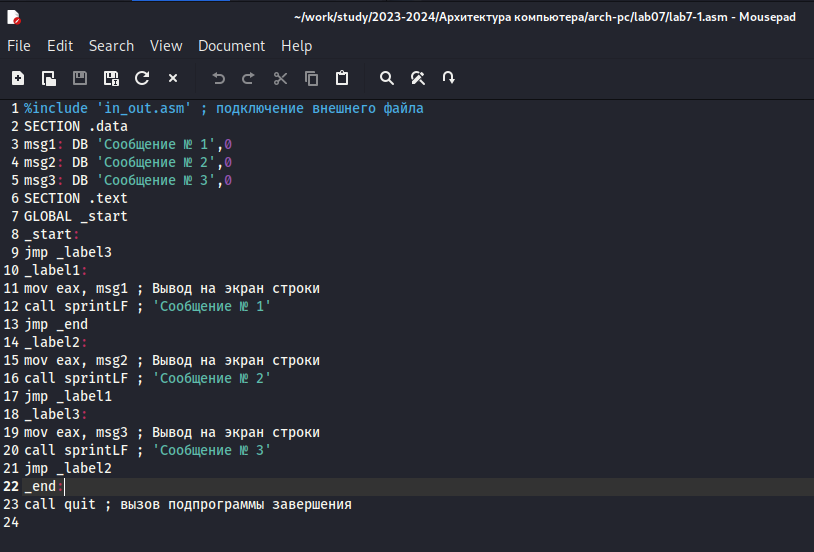
Изменение текста программы

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



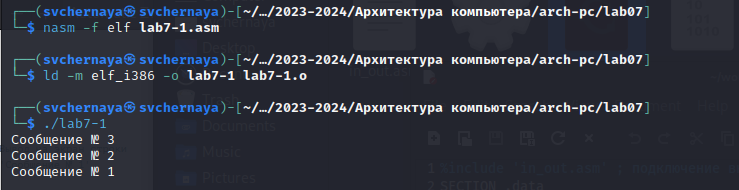
Создание исполняемого файла

Затем изменяю 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. [??]).



Изменение текста программы

чтобы вывод программы был следующим: (рис. [??]).



Вывод программы

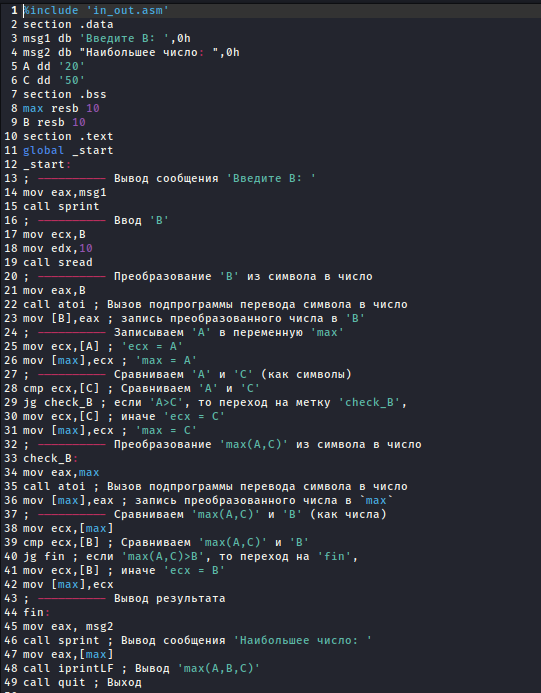
Рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A,B и C. Значения для A и C задаются в программе, значение B вводиться с клавиатуры.

Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. (рис. [??]).

Создание файла

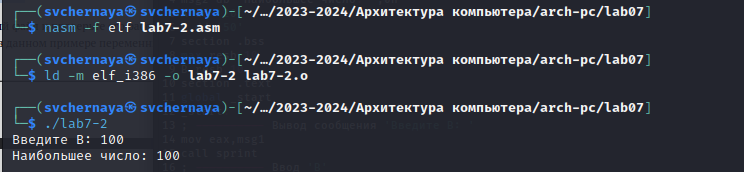
Создание файла

Текст программы из листинга 7.3 ввожу в lab7-2.asm. (рис. [??]).



Ввод текста программы из листинга 7.3

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



Проверка работы файла

Файл работает корректно.

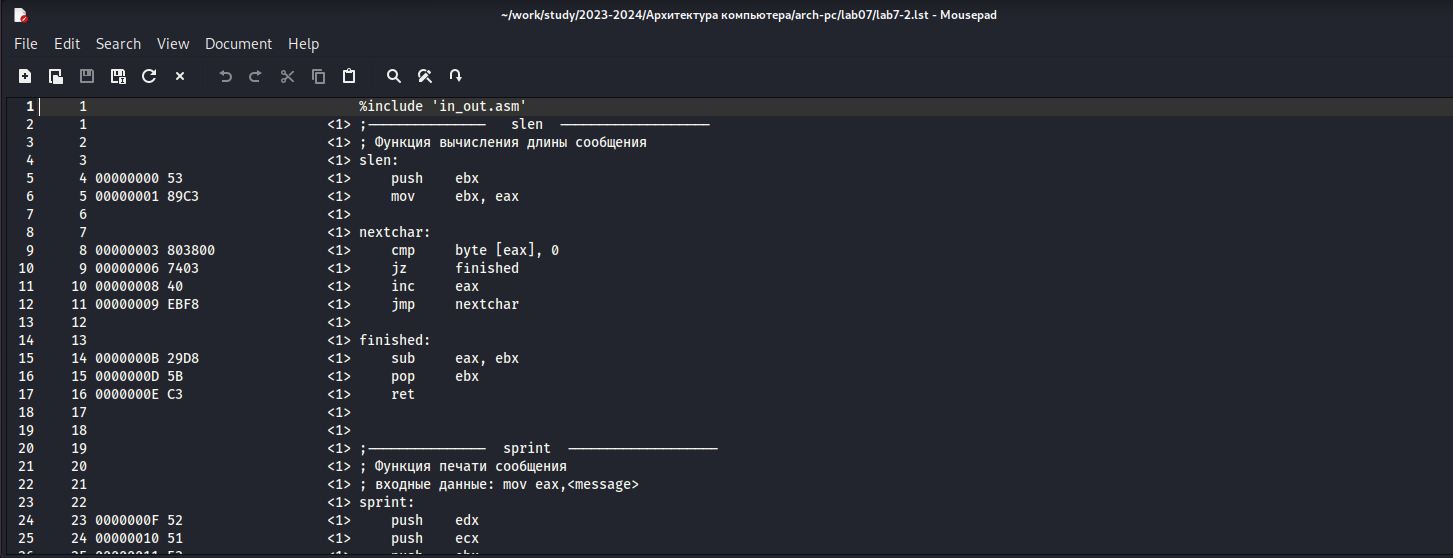
## 4.1 **Изучение структуры файлы листинга**

Создаю файл листинга для программы из файла lab7-2.asm. (рис. [??]).

Создание файла листинга

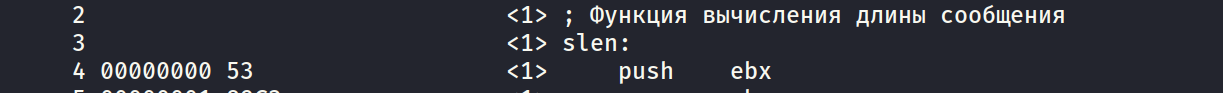
Создание файла листинга

Открываю файл листинга lab7-2.lst с помощью текстового редактора и внимательно изучаю его формат и содержимое. (рис. [??]).



Изучение файла листинга

В представленных трех строчках содержаться следующие данные: (рис. [??]).



Выбранные строки файла

“2” - номер строки кода, “; Функция вычисления длинны сообщения” - комментарий к коду, не имеет адреса и машинного кода.

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

“4” - номер строки кода, “00000000” - адрес строки, “53” - машинный код, “push ebx” - исходный текст программы, инструкция “push” помещает операнд “ebx” в стек.

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

Удаление выделенного операнда из кода

Удаление выделенного операнда из кода

Выполняю трансляцию с получением файла листинга. (рис. [??]).

Получение файла листинга

Получение файла листинга

На выходе я не получаю ни одного файла из-за ошибки:инструкция mov (единственная в коде содержит два операнда) не может работать, имея только один операнд, из-за чего нарушается работа кода.

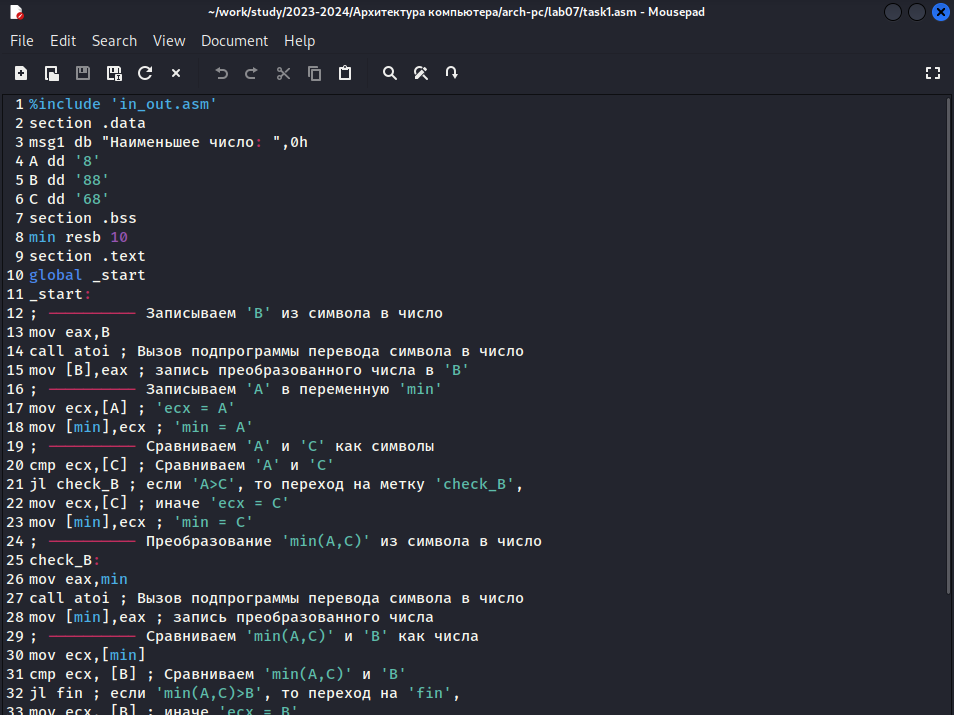
## 4.2 Выполнение заданий для самостоятельной работы

Создаю файл task1.asm с помощью утилиты touch (рис. [??]).

Создание файла

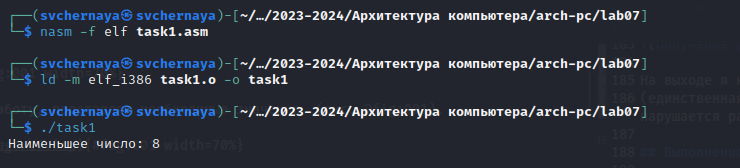
Создание файла

Пишу программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбираю из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Мой вариант под номером 4, поэтому мои значения - 8, 88 и 68. (рис. [??]).



Написание программы

Создаю исполняемый файл и проверяю его работу, подставляя необходимые значение. (рис. [??]).



Запуск файла и проверка его работы

Программа работает корректно.

Код программы:

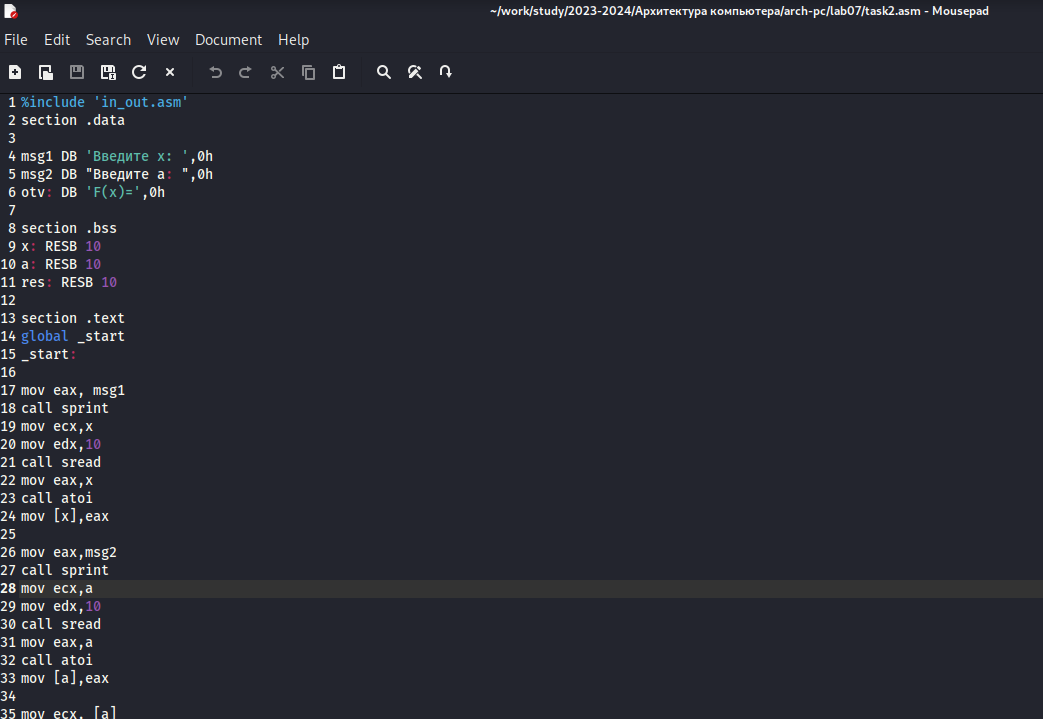
%include 'in\_out.asm'  
section .data  
msg1 db "Наименьшее число: ",0h  
A dd '8'  
B dd '88'  
C dd '68'  
section .bss  
min resb 10  
section .text  
global \_start  
\_start:  
; ---------- Записываем 'B' из символа в число  
mov eax,B  
call atoi ; Вызов подпрограммы перевода символа в число  
mov [B],eax ; запись преобразованного числа в 'B'  
; ---------- Записываем 'A' в переменную 'min'  
mov ecx,[A] ; 'ecx = A'  
mov [min],ecx ; 'min = A'  
; ---------- Сравниваем 'A' и 'C' как символы  
cmp ecx,[C] ; Сравниваем 'A' и 'C'  
jl check\_B ; если 'A>C', то переход на метку 'check\_B',  
mov ecx,[C] ; иначе 'ecx = C'  
mov [min],ecx ; 'min = C'  
; ---------- Преобразование 'min(A,C)' из символа в число  
check\_B:  
mov eax,min  
call atoi ; Вызов подпрограммы перевода символа в число  
mov [min],eax ; запись преобразованного числа   
; ---------- Сравниваем 'min(A,C)' и 'B' как числа  
mov ecx,[min]  
cmp ecx, [B] ; Сравниваем 'min(A,C)' и 'B'  
jl fin ; если 'min(A,C)>B', то переход на 'fin',  
mov ecx, [B] ; иначе 'ecx = B'  
mov [min],ecx  
; ---------- Вывод результата  
fin:  
mov eax, msg1  
call sprint ; Вывод сообщения 'Наименьшее число: '  
mov eax,[min]  
call iprintLF ; Вывод 'min(A,B,C)'  
call quit ; Выход

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

2 \* x - a, если a ≠ 0

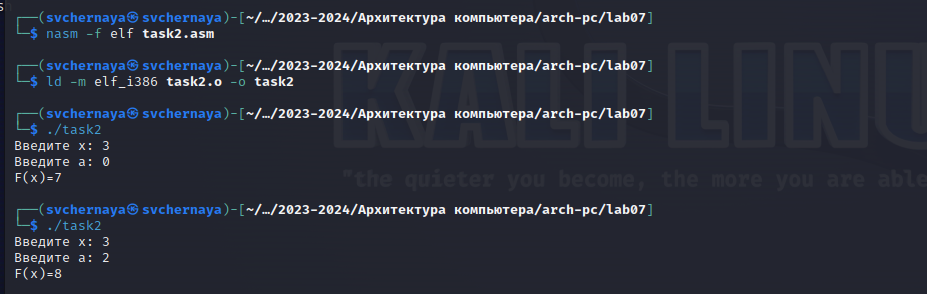
2 \* x + 1, если a = 0

(рис. [??]).



Написание программы

Создаю исполняемый файл и проверяю его работу для значений х и а соответственно: (3;0), (3;2). (рис. [??]).



Запуск файла и проверка его работы

Программа работает корректно.

Код программы:

%include 'in\_out.asm'  
section .data  
  
msg1 DB 'Введите x: ',0h  
msg2 DB "Введите a: ",0h  
otv: DB 'F(x)=',0h  
  
section .bss  
x: RESB 10  
a: RESB 10  
res: 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  
mov [x],eax  
  
mov eax,msg2  
call sprint  
mov ecx,a  
mov edx,10  
call sread  
mov eax,a  
call atoi  
mov [a],eax  
  
mov ecx, [a]  
cmp ecx, 0  
je x\_is\_0  
mov eax, [x]  
mov ebx,2  
mul ebx  
add eax, ecx  
jmp calc\_res  
x\_is\_0:  
mov ebx,2  
mov eax, [x]  
mul ebx  
inc eax  
calc\_res:  
mov [res],eax  
fin:  
mov eax,otv  
call sprint  
mov eax,[res]  
call iprintLF  
call quit

# 5 Выводы

По итогам данной лабораторной работы я изучила команды условного и безусловного переходов, приобрела навыки написания программ с использованием переходов и ознакомилась с назначением и структурой файла листинга, что поможет мне при выполнении последующих лабораторных работ.

# Список литературы