Отчёт по лабораторной работе №7

Дисциплина: архитектура компьютера

Хаджилари Гешлаг Никта

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

## 2.1 1. Команды условного перехода

## 2.2 2. Реализация переходов в NASM

## 2.3 3. Изучение структуры файлы листинга

## 2.4 4. Cамостоятельная работа

# 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.

# 4 Выполнение лабораторной работы

**1**

С помощью утилиты mkdir создаю директорию lab07, перехожу в нее и создаю файл для работы. (рис. [[1](#fig:001)]).

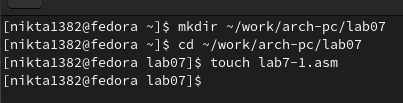


Figure 1: Создание директории

**2**

Копирую в текущий каталог файл in\_out.asm из загрузок, т.к. он будет использоваться в других программах. Открываю созданный файл lab7-1.asm, вставляю в него программу реализации безусловных переходов(рис. [**fig002?**]).

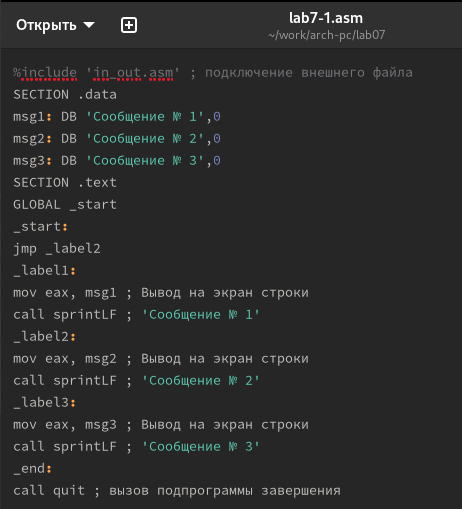


Figure 2: Создание копии файла для дальнейшей работы, редактирование файла

**3**

Создаю исполняемый файл программы и запускаю его (рис. [[3](#fig:003)]). Инструкции jmp \_label2 меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки \_label2.

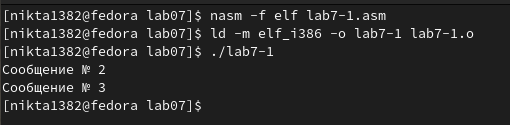


Figure 3: Запуск исполняемого файла

**4**

Изменяю текст программы, так чтобы вывод происходил в обратном порядке (рис. [[4](#fig:004)]).

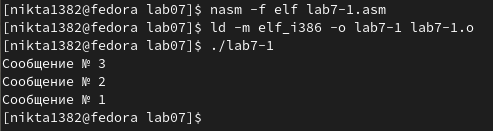


Figure 4: Редактирование программы

**5**

Создаю исполняемый файл и проверяю работу программы (рис. [[5](#fig:005)]). Программа отработало верно.

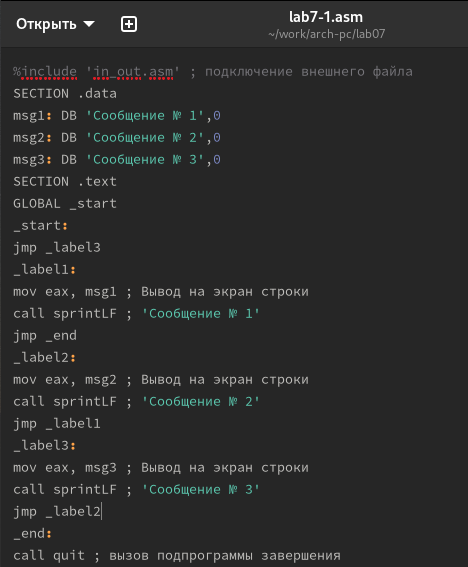


Figure 5: Создание исполняемого файла

**6**

Создаю новый файл lab7-2.asm для программы с условным оператором. (рис. [[6](#fig:006)]).

Figure 6: Создание файла

Figure 6: Создание файла

**7**

Вставляю программу, которая определяет и выводит на экран наибольшее число (рис.[[8](#fig:007)]).

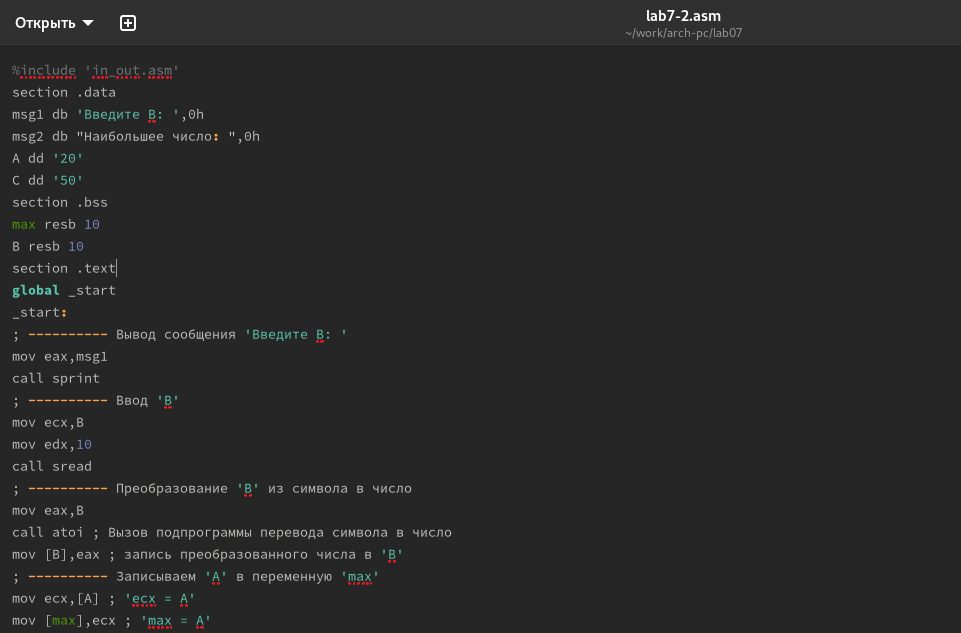


Figure 7: Вставляю текст в файл

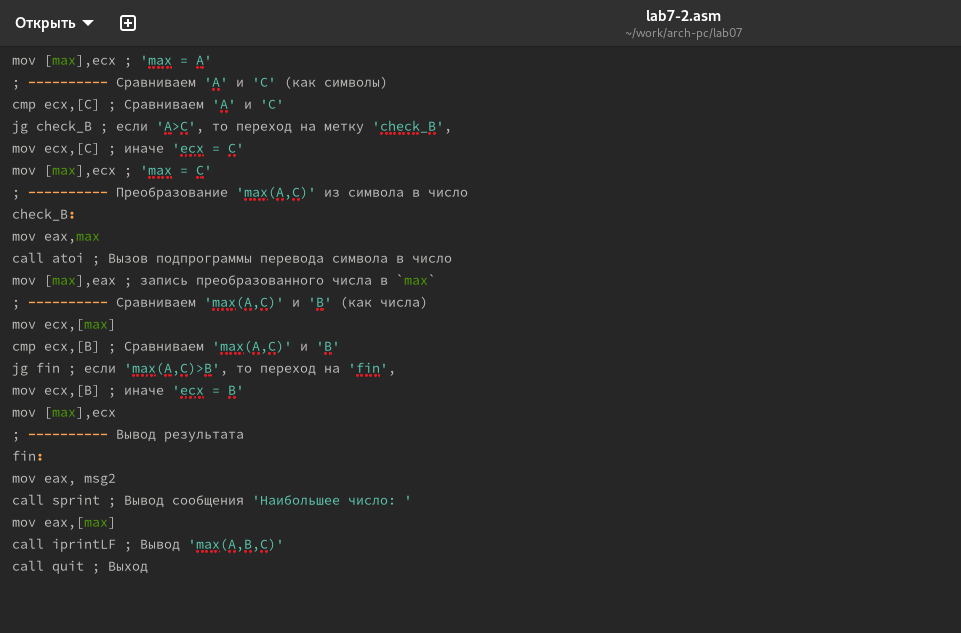


Figure 8: Вставляю текст в файл

**8**

Создаю и запускаю новый исполняемый файл, проверяю работу программы (рис. [[10](#fig:008)]).

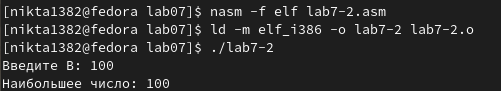


Figure 9: Запуск исполняемого файла

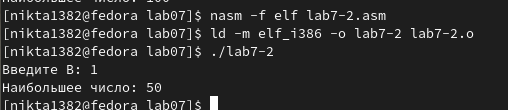


Figure 10: Запуск исполняемого файла

**9**

Открываю файл листинга с помощью редактора mcedit. Расмотрим 9-11 строки: (рис. [[11](#fig:009)]).

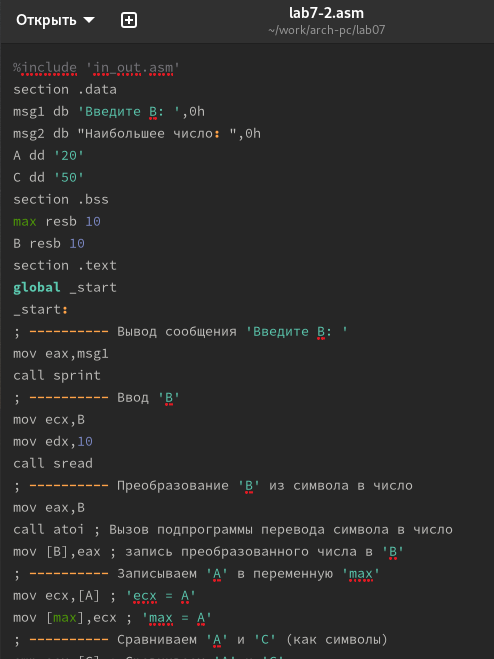


Figure 11: Файл листинга

9 строка:

* Перые цифры [9] - это номер строки файла листинга.
* Cледующие цифры [00000006] адрес — это смещение машинного кода от начала текущего сегмента, состоит из 8 чисел.
* следующие числа [7403] - это машинный код, который представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности, поэтоу и появляются буквы латынского алфавита.
* следющее [jz finished] - исходный текст программы, которая просто состоит из строкк исходной программы вместе с комментариями.

10 строка:

* Перые цифры [10] - это номер строки файла листинга.
* Cледующие цифры [00000008] адрес — это смещение машинного кода от начала текущего сегмента, состоит из 8 чисел.
* следующие числа [40] - это машинный код, который представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности, поэтоу и появляются буквы латынского алфавита.
* следющее [inc eax] - исходный текст программы, которая просто состоит из строкк исходной программы вместе с комментариями

11 строка:

* Перые цифры [11] - это номер строки файла листинга.
* Cледующие цифры [00000009] адрес — это смещение машинного кода от начала текущего сегмента, состоит из 8 чисел.
* следующие числа [EBF8] - это машинный код, который представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности, поэтоу и появляются буквы латынского алфавита.
* следющее [jmp nextchar] - исходный текст программы, которая просто состоит из строкк исходной программы вместе с комментариями

**10**

Открываю файл листинга с помощью редактора mcedit и замечаю, что в файле листинга появляется ошибка. (рис. [[12](#fig:010)]).

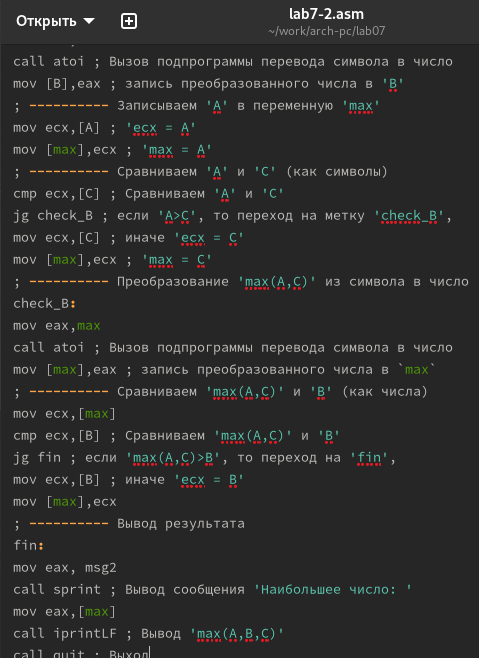


Figure 12: Файл листинга

Отсюда можно сделать вывод, что, если в коде появляется ошибка, то ее описание появится в файле листинга

**11**

Создал файл листинга для программы из файла lab7-2.asm (рис. [[13](#fig:011)]).

Figure 13: asm -f elf -l lab7-2.lst lab7-2.asm

Figure 13: asm -f elf -l lab7-2.lst lab7-2.asm

**12**

Открыл файл листинга lab7-2.lst с помощью любого текстового редактора, например gedit:{#fig:012 width=70%}

Figure 14: gedit lab7-2.lst

Figure 14: gedit lab7-2.lst

**13**

Открыл файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалить один операнд. Выполните трансляцию с получением файла листинга:{#fig:013 width=70%}

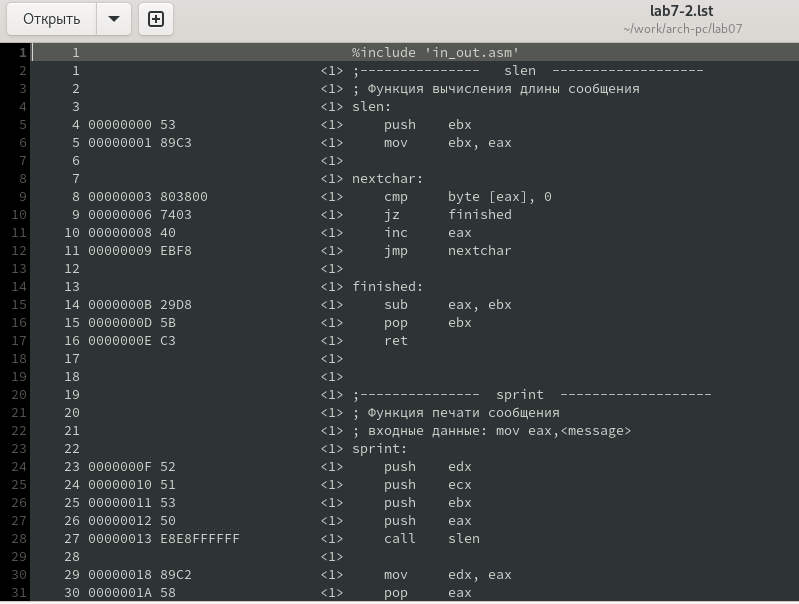


Figure 15:

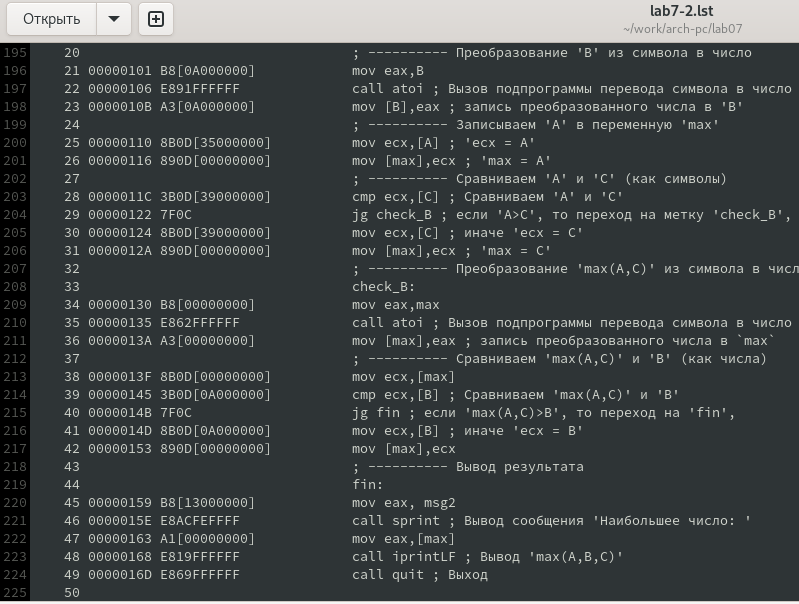


Figure 16:

# 5 Самостоятельная работа

**1**

Создаю файл lab7.asm с помощью утилиты touch и запускаю редактора gedit (рис. [[17](#fig:014)]).

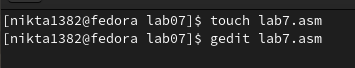


Figure 17: Создание запуск файла

**2**

Ввожу в созданный файл текст программы для вычисления наименьшего из 3 чисел. Числа беру, учитывая свой вариант из прошлой лабораторной работы. 20 вариант (рис. [[18](#fig:015)]).

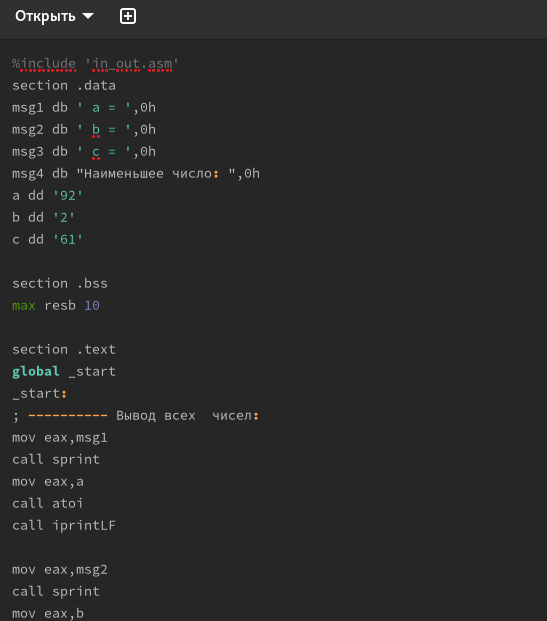


Figure 18: Редактирование файла

**3**

Создаю исполняемый файл и запускаю его (рис. [[19](#fig:016)]).

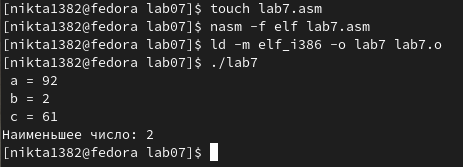


Figure 19: Запуск исполняемого файла

**Текст программы**

%include 'in\_out.asm'  
section .data  
msg1 db ' а = ',0h  
msg2 db ' b = ',0h  
msg3 db ' c = ',0h  
msg4 db "Наименьшее число: ",0h  
a dd '92'  
b dd '2'  
c dd '61'  
  
section .bss  
max resb 10  
  
section .text  
global \_start  
\_start:  
; ---------- Вывод всех чисел:   
mov eax,msg1  
call sprint  
mov eax,a  
call atoi  
call iprintLF  
  
mov eax,msg2  
call sprint  
mov eax,b  
call atoi  
call iprintLF  
  
mov eax,msg3  
call sprint  
mov eax,c  
call atoi  
call iprintLF  
  
;-------------сравнивание чисел  
mov eax,b  
call atoi ;перевод символа в число  
mov [b],eax ; запись преобразованного числа в b  
;------------ запись b в переменную мах  
mov ecx,[a] ;  
mov [max],ecx ;  
;------------сравнивание чисел a c  
cmp ecx,[c]; if a>c  
jl check\_b ; то перход на метку  
mov ecx,[c] ;  
mov [max],ecx ;  
;-------метка check\_b  
check\_b:  
mov eax,max ;  
call atoi  
mov [max],eax ;  
;------------  
mov ecx,[max] ;  
cmp ecx,[b] ;  
jl check\_c ;  
mov ecx,[b] ;  
mov [max],ecx ;  
;-------------  
check\_c:  
mov eax,msg4 ;  
call sprint ;   
mov eax,[max];  
call iprintLF ;  
call quit

**4**

Создаю новый файл lab7-3 для написания программы второго задания. (рис. [[20](#fig:017)]).

Figure 20: создание файла

Figure 20: создание файла

**5**

Ввожу в него программу, в которую ввожу значения 20 x и a, и которая выводит значения функции. Функцию беру из таблицы в соответствии со своим вариантом (рис. [[21](#fig:018)]).

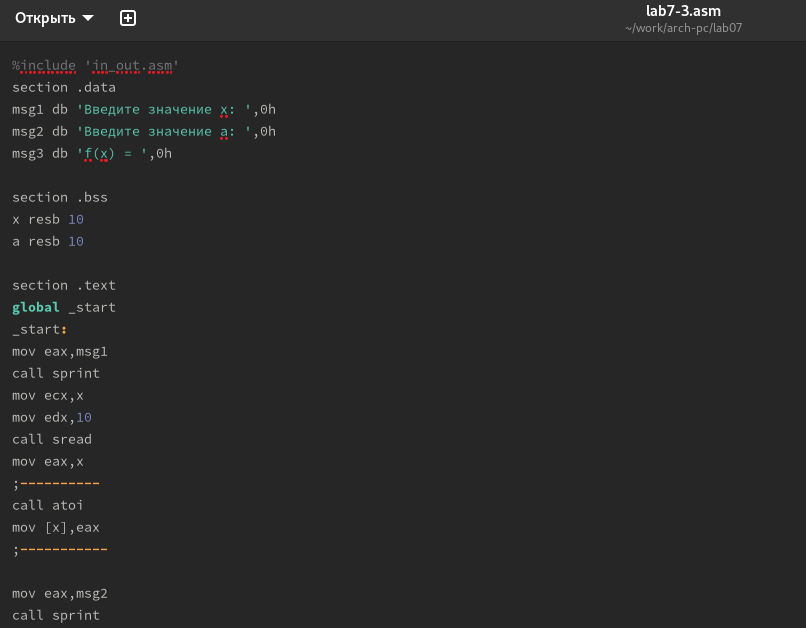


Figure 21: ввод программы в файл

**6**

Создаю испольняемый файл и проверяю её выполнение при x=1, a=2

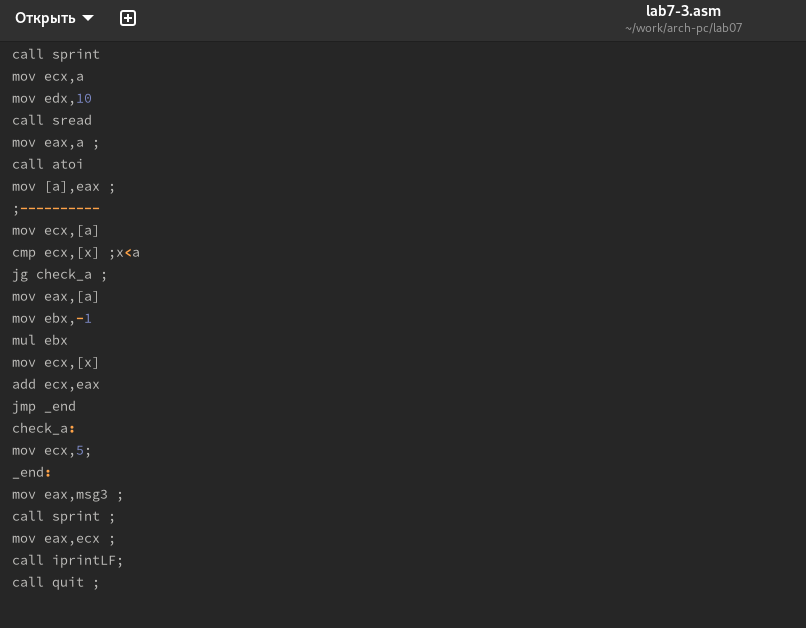


Figure 22: Создание исполняемого файла

**7**

Повторный раз запускаю программу и проверяю ее выполнение при x=2 и a=1

Программа отработала верно!

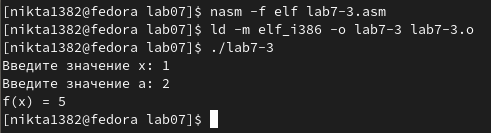


Figure 23: запуск исполняемого файла

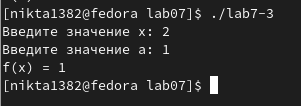


Figure 24: запуск исполняемого файла

**Текст программы**

%include 'in\_out.asm'  
section .data  
msg1 db 'Введите значение x: ',0h  
msg2 db 'Введите значение a: ',0h  
msg3 db 'f(x) = ',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  
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,[x] ;x<a  
jg check\_a ;  
mov eax,[a]  
mov ebx,-1  
mul ebx  
mov ecx,[x]  
add ecx,eax  
jmp \_end  
check\_a:  
mov ecx,5;  
\_end:  
mov eax,msg3 ;  
call sprint ;  
mov eax,ecx ;  
call iprintLF;  
call quit ;

# 6 Выводы

При выполнении данной лабораторной работы я освоил инструкции условного и безусловного вывода и ознакомился с структурой файла листинга.

::

{#refs} :::