РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук Кафедра прикладной информатики и теории вероятностей

Архипов Олег Константинович

Содержание

# 1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

# 2 Выполнение лабораторной работы

## 2.1 Реализация циклов в NASM

Создаю каталог для программам лабораторной работы № 8, перехожу в него и создаю файл lab8-1.asm (рис. [1](#fig:001)).

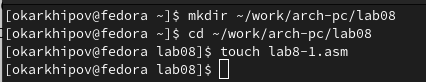


Figure 1: Папка и файл ЛР

Изучаю программу, которая выводит значение регистра ecx, являющегося счетчиком для инструкции loop (рис. [2](#fig:002)).

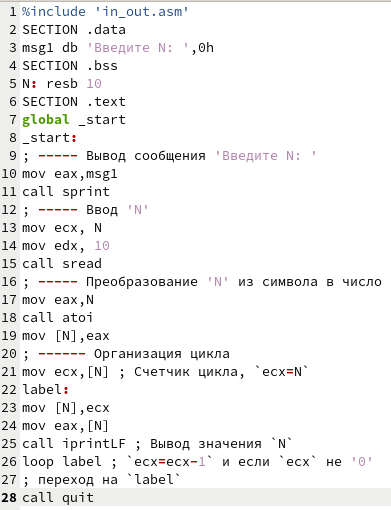


Figure 2: Программа вывода значений регистра ‘ecx’

Создаю исполняемый файл и проверяю его работу для значения 10, однако программа работает не так, как задумывалось: она должна выводить значения от 9 до 0 (рис. [3](#fig:003)).

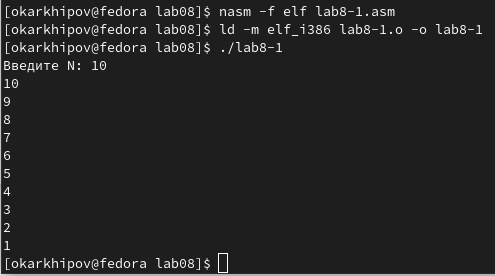


Figure 3: Работа программы lab8-1

Изменяю текст программы, добавляя изменение значения регистра ecx в цикле (рис. [4](#fig:004)).

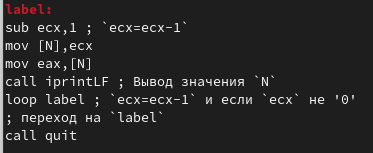


Figure 4: Изменение 1 в файле lab8-1.asm

Еще раз проверяю работу программы, снова получаю неверный результат, т.к. теперь программа за 1 проход цикла дважды уменьшает значение ‘ecx’ на 1. Т.е. получаю первое значение 9, а далее каждый раз вычитается 2: 7, 5, 3, 1 (рис. [5](#fig:005)).

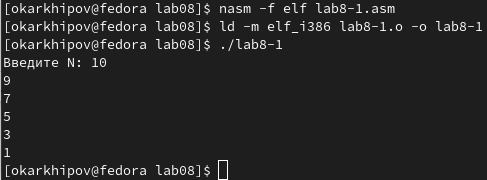


Figure 5: Работа измененной первый раз программы lab8-1

Для сохранения корректности работы программы с использованием регистра ecx в цикле нужен стек: ввожу команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop (рис. [6](#fig:006)).

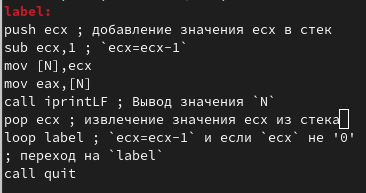


Figure 6: Изменение 2 в файле lab8-1.asm

Проверяю работу программы, на этот раз результат корректен (рис. [7](#fig:007)).

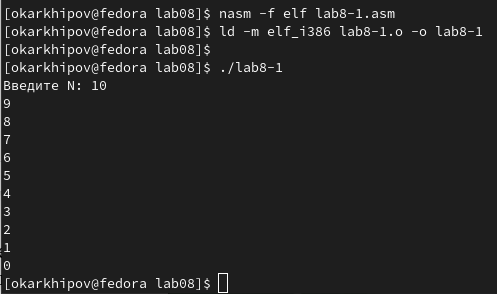


Figure 7: Работа измененной второй раз программы lab8-1

## 2.2 Обработка аргументов командной строки

Создаю файл lab8-2.asm (рис. [8](#fig:008)).

Figure 8: Файл lab8-2.asm

Figure 8: Файл lab8-2.asm

Ввожу текст программы обработки аргументов командной строки (рис. [9](#fig:009)).

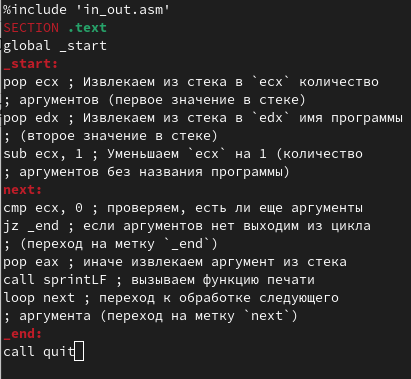


Figure 9: Текст программы lab8-2

Создаю исполняемый файл и запускаю его с аргументами 9, 5, ‘3’, т.е. программа обработала 3 аргумента, а также имя программы, итого 4 (рис. [10](#fig:010)).

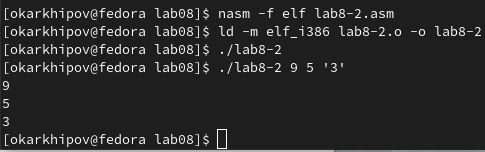


Figure 10: Работа программы lab8-2

Создаю файл lab8-3.asm (рис. [11](#fig:011)).

Figure 11: Файл lab8-3.asm

Figure 11: Файл lab8-3.asm

Пишу код программы которая выводит сумму чисел, которые передаются в программу как аргументы (рис. [12](#fig:012)).

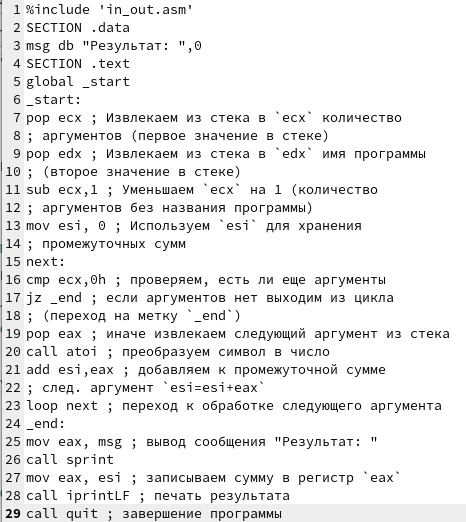


Figure 12: Код lab8-3

Далее создаю исполняемый файл с названием main и аргументами ‘12 13 7 10 5’ и запускаю его (рис. [13](#fig:013)).

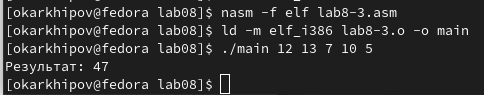


Figure 13: Работа программы lab8-3

Меняю текст программы для вычисления произведения аргументов командной строки (рис. [14](#fig:014)). Для этого вместо

mov esi, 0

пишу

mov esi, 1

т.к. нейтральный элемент по умножению - единица, а также вместо

add esi,eax

пишу

mul esi  
mov esi, eax

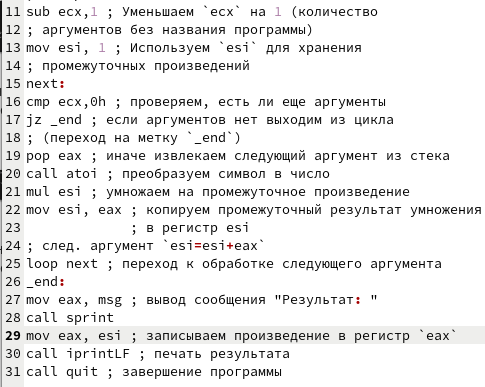


Figure 14: Измененный код lab8-3

После исполнения обновленной программы получаю 54600 (рис. [15](#fig:015)).

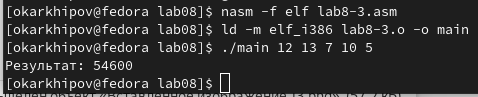


Figure 15: Работа измененной программы lab8-3

# 3 Задание для самостоятельной работы

Создаю файл sol8.asm (рис. [16](#fig:016)).

Figure 16: Файл sol8.asm

Figure 16: Файл sol8.asm

Беру тот же вариант, что был в двух предыдущих ЛР - 4. Код программы приведен в листинге ниже.

**Листинг программы для вычисления суммы значений функции**

%include 'in\_out.asm'  
SECTION .data  
msg db "Результат: ",0  
f db "Функция: f(x)=2(x-1)",0  
SECTION .text  
global \_start  
\_start:  
pop ecx ; Извлекаем из стека в `ecx` количество  
 ; аргументов (первое значение в стеке)  
pop edx ; Извлекаем из стека в `edx` имя программы  
 ; (второе значение в стеке)  
sub ecx,1 ; Уменьшаем `ecx` на 1 (количество  
 ; аргументов без названия программы)  
mov esi,0 ; Используем `esi` для хранения  
 ; промежуточных сумм  
  
next:  
cmp ecx,0h ; проверяем, есть ли еще аргументы  
jz \_end ; если аргументов нет выходим из цикла  
 ; (переход на метку `\_end`)  
pop eax ; иначе извлекаем следующий аргумент из стека  
call atoi ; преобразуем символ в число  
sub eax, 1 ; eax=eax-1  
mov edi, 2 ; edi=2  
mul edi ; eax=eax\*edi  
add esi, eax ; прибавляем к промежуточной сумме  
loop next ; переход к обработке следующего аргумента  
\_end:  
mov eax, f ; вывод сообщения "Функция: f(x)=2(x-1)"  
call sprintLF  
mov eax,msg ; вывод сообщения "Результат: "   
call sprint  
mov eax, esi ; записываем сумму в регистр `eax`  
call iprintLF ; печать результата  
call quit ; завершение программы

Проверяю работу программы с разными значениями x (рис. [17](#fig:017)).

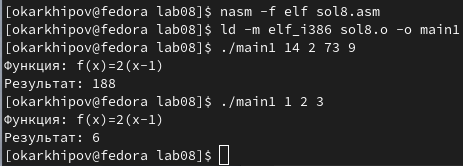


Figure 17: Результаты работы sol8

# 4 Выводы

Была освоена работа со стеком, циклами и обработка аргументов командной строки, а также методы написания программ при помощи данных инструментов.