Отчёт по лабораторной работе №8

Дисциплина: Архитектура компьютера

Самойлова Софья Дмитриевна

Содержание

# 1 Цель работы

Целью лабораторной работы является приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

# 2 Задание

Здесь приводится описание задания в соответствии с рекомендациями методического пособия и выданным вариантом.

# 3 Теоретическое введение

Стек — это структура данных, организованная по принципу *LIFO* (*«Last In — First Out»* или «последним пришёл — первым ушёл»). Стек является частью архитектуры процессора и реализован на аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды. Основной функцией стека является функция сохранения адресов возврата и передачи аргументов при вызове процедур. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров.

Стек имеет вершину, адрес последнего добавленного элемента, который хранится в регистре esp (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указатель стека уменьшается, а при извлечении — увеличивается. Для стека существует две основные операции:

* добавление элемента в вершину стека (push);
* извлечение элемента из вершины стека (pop).

# 4 Выполнение лабораторной работы

## 4.1 Реализация циклов в NASM

Создаю каталог для программ лабораторной работы № 8, перехожу в него и создаю файл lab8-1.asm, дополнительно копирую в текущий каталог файл in\_out.asm с помощью утилиты cp, т.к. он будет использоваться в других программах (рис. 1).

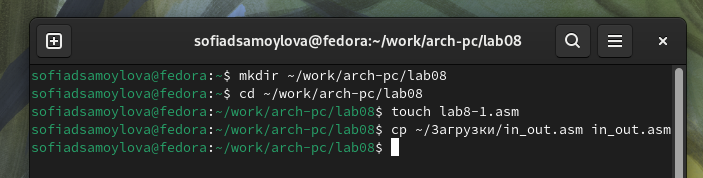


Рис. 1: Создание каталога

В качестве примера рассмотрю программу, которая выводит значение регистра ecx. Ввожу в файл lab8-1.asm текст программы (рис. 2).

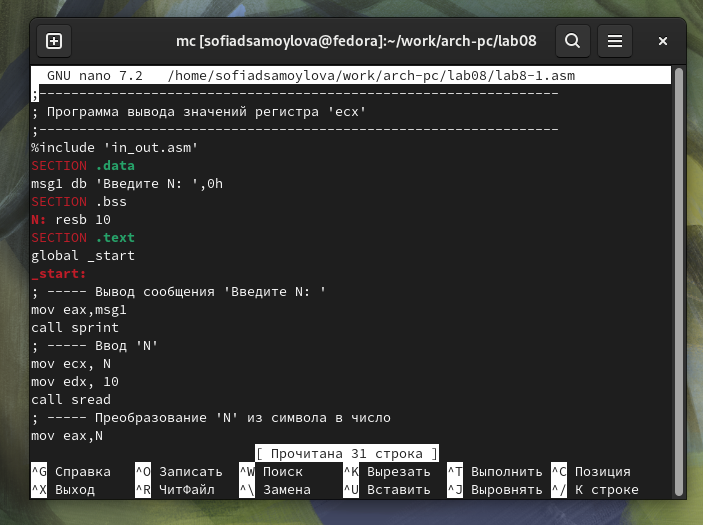


Рис. 2: Программа из листинга

Создаю исполняемый файл и проверяю его работу (рис. 3).

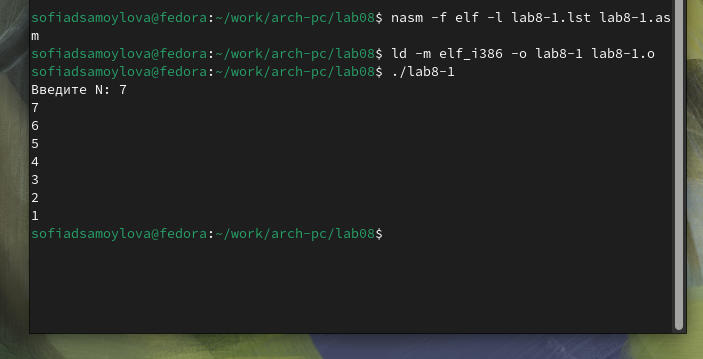


Рис. 3: Работа программы

Программа показывает работу циклов в NASM.

Изменяю программу изначальную таким образом, что в теле цикла я изменяю значение регистра ecx (рис. 4).

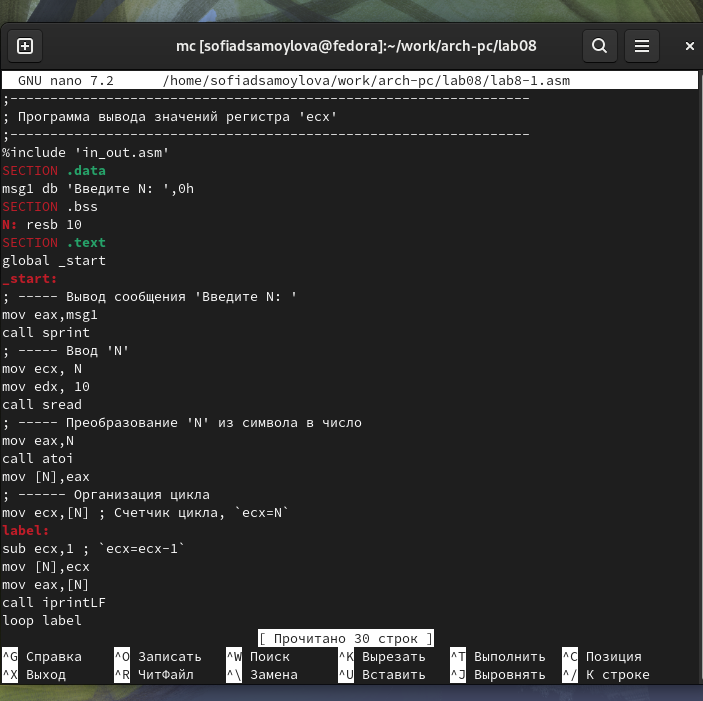


Рис. 4: Редактирование программы

Как результат цикл программы стал бесконечным, несмотря на то, что значение для N было задано 5 (рис. 5).

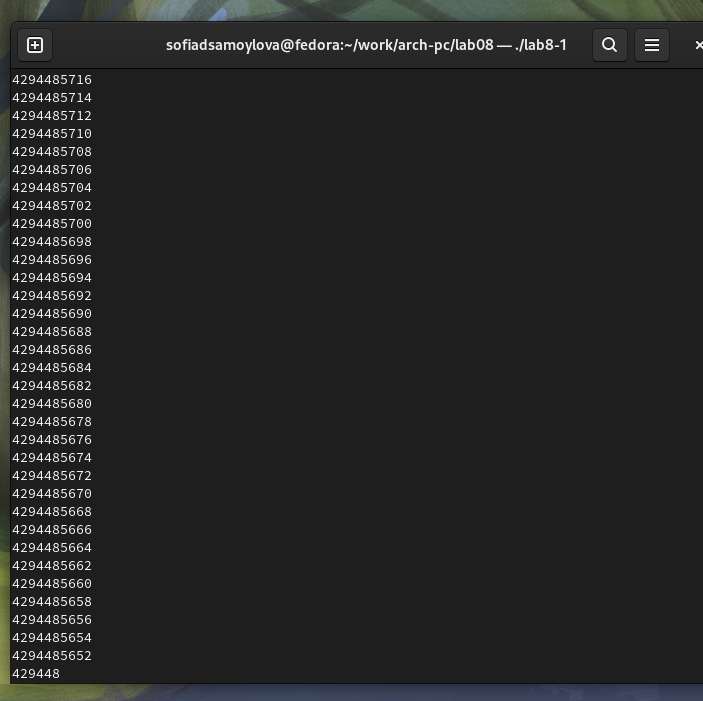


Рис. 5: Результат изменений программы

Для использования регистра ecx в цикле и сохранения корректности работы программы попробую использовать стек. Вношу изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop(рис. 6).

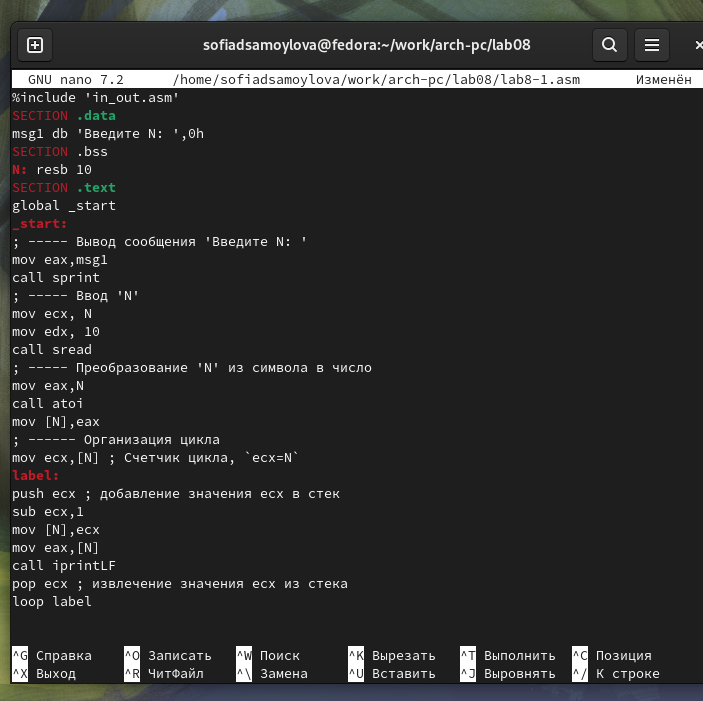


Рис. 6: Редактирование программы

Создаю исполняемый файл и проверяю его работу (рис. 7).

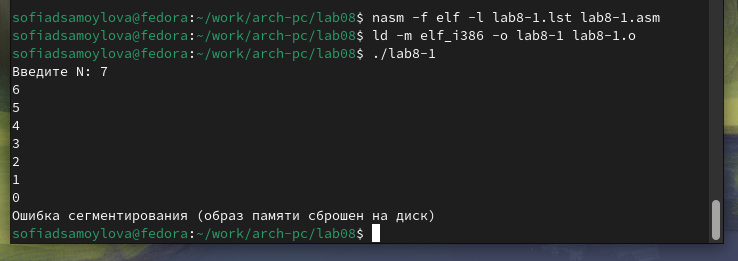


Рис. 7: Работа программы

Число проходов по циклу соответствует значению N, с учетом 0.

## 4.2 Обработка аргументов командной строки

Создаю файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввожу в него текст программы (рис. 8).

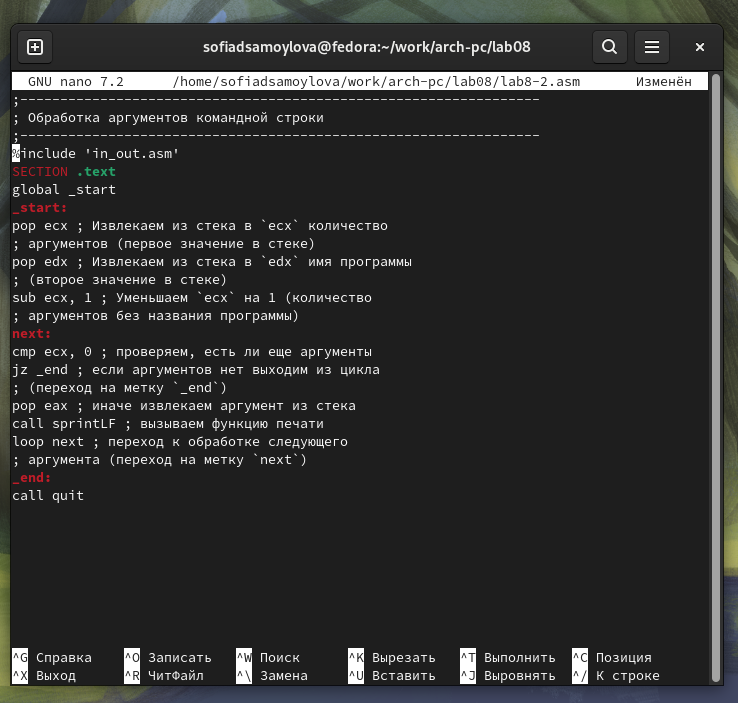


Рис. 8: Редактирование программы

Запускаю его, указав аргументы (рис. 9).

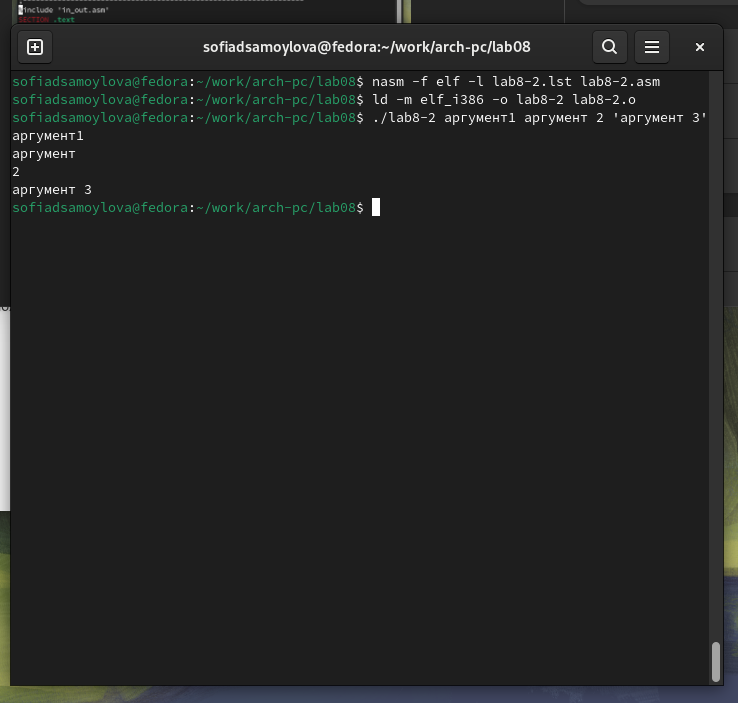


Рис. 9: Запуск программы

Все аргументы были обработаны программой.

Создаю файл lab8-3.asm в каталоге ~/work/archpc/lab08 и ввожу в него текст программы, которая выводит сумму чисел, которые передаются в программу как аргументы (рис. 10).

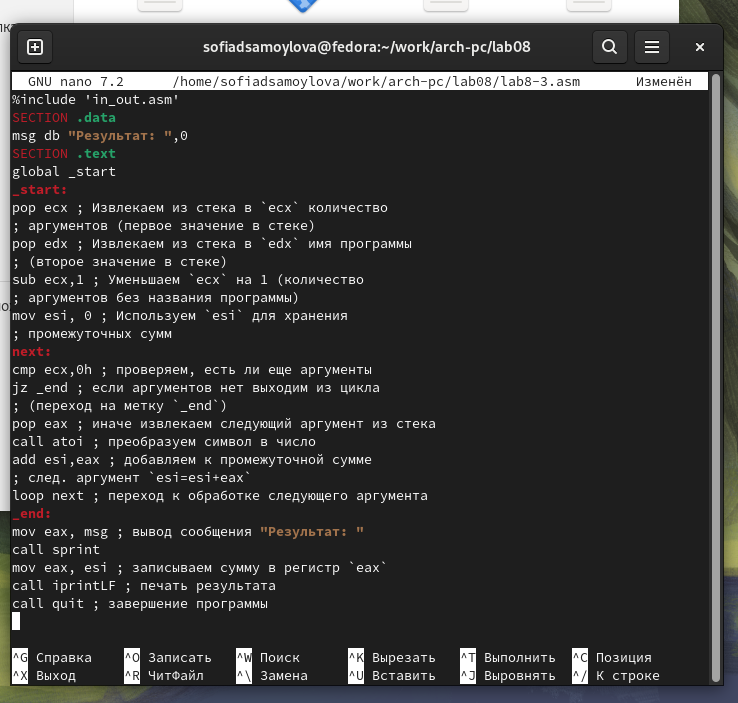


Рис. 10: Создание программы

Создаю исполняемый файл и запускаю его, указав аргументы (рис. 11).

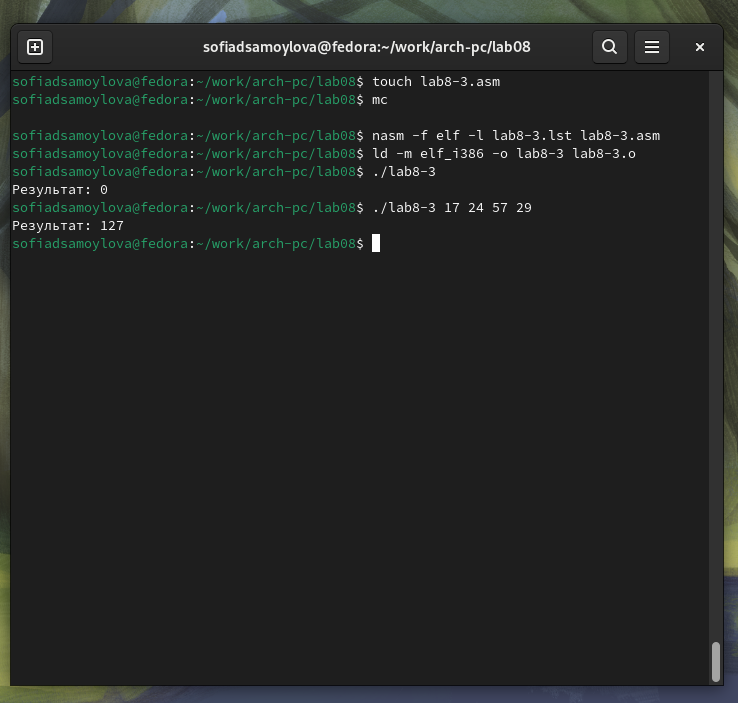


Рис. 11: Работа программы

Изменяю текст программы для вычисления произведения аргументов командной строки (рис. 12).

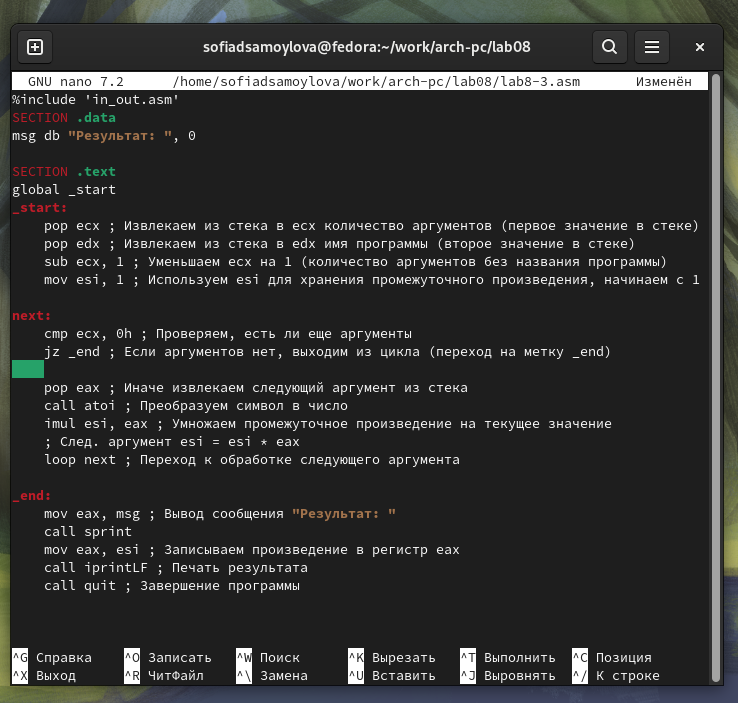


Рис. 12: Изменение программы

Проверяю работу программы (рис. 13).

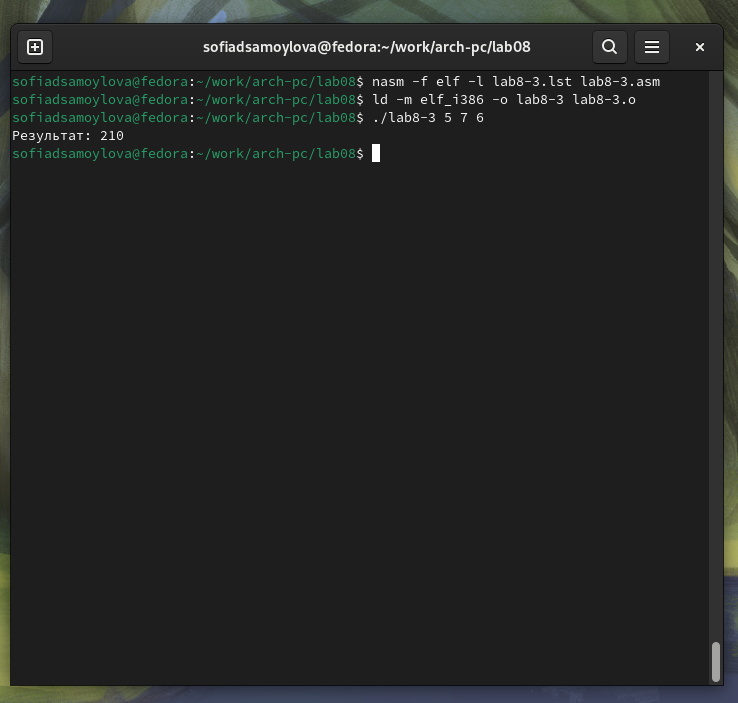


Рис. 13: Работа программы

## 4.3 Выполнение самостоятельной работы

Вариант задания номер 17. Код программы:

%include 'in\_out.asm'  
SECTION .data  
msg\_func db "Функция: f(x) = 10(x - 1)", 0  
msg\_result db "Результат: ", 0  
SECTION .text  
GLOBAL \_start  
\_start:  
 mov eax, msg\_func  
 call sprintLF  
 pop ecx ; Извлекаем количество аргументов  
 pop edx ; Извлекаем имя программы  
 sub ecx, 1 ; Уменьшаем на 1, чтобы исключить имя программы  
 mov esi, 0 ; Инициализируем сумму (esi) нулем  
  
next:  
 cmp ecx, 0h ; Проверяем, есть ли еще аргументы  
 jz \_end ; Если нет, переходим к окончанию  
  
 pop eax ; Извлекаем следующий аргумент  
 call atoi ; Преобразуем строку в число  
  
 mov ebx, eax ; Сохраняем значение x в ebx  
 sub ebx, 1 ; Вычисляем (x - 1)  
 mov eax, 10 ; Загружаем 10 в eax  
 imul eax, ebx ; Умножаем 10 на (x - 1)  
  
 add esi, eax ; Добавляем результат функции к сумме  
  
 loop next ; Переход к следующему аргументу  
\_end:  
 mov eax, msg\_result  
 call sprint ; Выводим сообщение "Результат: "  
 mov eax, esi ; Загружаем сумму в eax  
 call iprintLF ; Печатаем результат  
 call quit ; Завершаем программу

Результат работы программы (рис. 14).

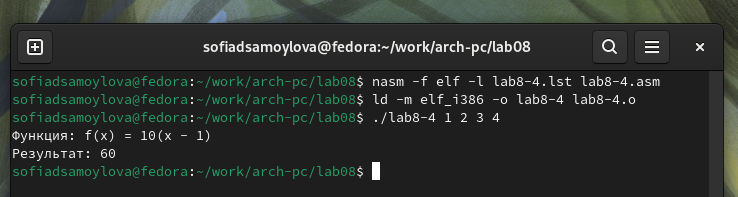


Рис. 14: Работа программы

# 5 Выводы

В результате выполнения данной лабораторной работы я приобрела навыки написания программ с использованием циклов, а также научилась обрабатывать аргументы командной строки.