Лабораторная работа №8

Архитектура компьютеров

Тойчубекова Асель Нурлановна

Содержание

# 1 Цель работы

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

# 2 Задание

1. Изучить теоретическое введение:

* Организация стека;
* Добавление элемента в стек;
* Извлечение элемента из стека;
* Инструкции организации циклов.

1. Изучив реализацию циклов в NASM, написать программу вывода значений регистра ecx.
2. Изучив обработку аргументов командной строки, написать программу выводящую на экран аргументы командной строки.
3. Написать программу вычисляющую суммы и произведений аргументов командной строки.
4. Выполнить задание для самостоятельной работы:  
   Напишите программу, которая находит сумму значений функции F(x) для x = x1, x2, …, xn, т.е. программа должна выводить значение F(x1) + f(x2) + … + f(xn). Значения xi передаются как аргументы. Вид функции f(x) выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы No 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах x = x1, x2, …,xn

# 3 Теоретическое введение

## 3.1 Организация стека

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. Для стека существует две основные операции:push- добавление элементов в вершину стека; pop-извлечение элемента из вершины стека.

## 3.2 Добавление элемента в стек

Команда push размещает значение в стеке, т.е. помещает значение в ячейку памяти, на которую указывает регистр esp, после этого значение регистра esp увеличивается на 4. Данная команда имеет один операнд — значение, которое необходимо поместить в стек.

## 3.3 Извлечение элемента из стека

Команда pop извлекает значение из стека, т.е. извлекает значение из ячейки памяти, на которую указывает регистр esp, после этого уменьшает значение регистра esp на 4. У этой команды также один операнд, который может быть регистром или переменной в памяти. Нужно помнить, что извлечённый из стека элемент не стирается из памяти и остаётся как “мусор”, который будет перезаписан при записи нового значения в стек

## 3.4 Инструкции организации циклов

Для организации циклов существуют специальные инструкции. Для всех инструкций максимальное количество проходов задаётся в регистре ecx. Наиболее простой является ин- струкция loop. Она позволяет организовать безусловный цикл, типичная структура. Иструкция loop выполняется в два этапа. Сначала из регистра ecx вычитается единица и его значение сравнивается с нулём. Если регистр не равен нулю, то выполняется переход куказанной метке. Иначе переход не выполняется и управление передаётся команде, которая следует сразу после команды loop.

# 4 Выполнение лабораторной работы

Для начала создадим каталог для программ лабораторной работы №8. Перейдя в каталог создадим файл lab8-1.asm.(РИС.1)

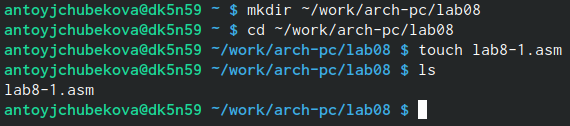


РИС.1 Создание каталога и файла

Напишем в файл lab8-1.asm программу вывода значений регистра eax, перед этим внимательно изучив ее.(РИС.2)

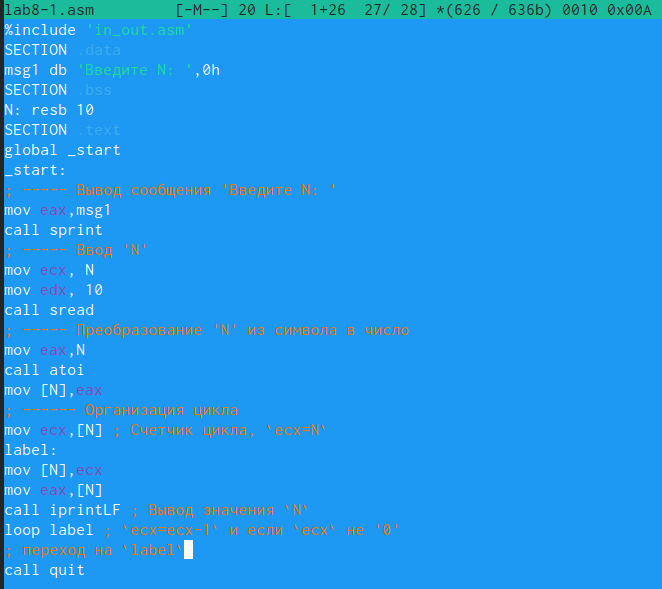


РИС.2 Редактирование файла

Создадим исполняемый файл и запустим его. Мы видим, что программа выводит значения от N до 1.(РИС.3) Данный пример показывает, что использование регистра ecx в теле цилка loop может привести к некорректной работе программы

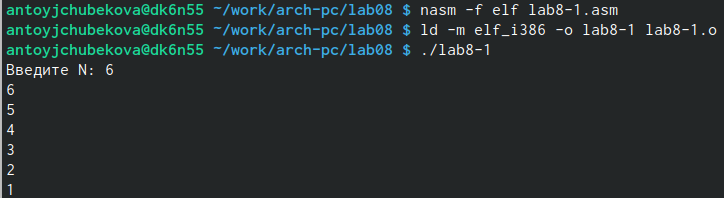


РИС.3 Запуск исполняемого файла

Чтобы исправить некорректную работу программы внесем изменения в текст программы добавив изменение значение регистра ecx в цикле.(РИС.4)

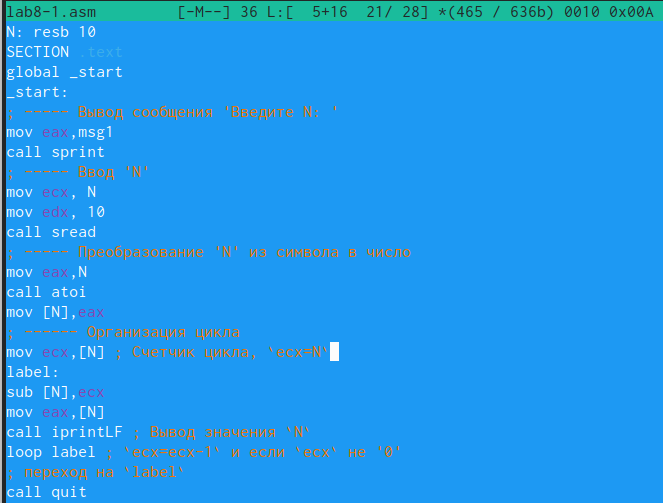


РИС.4 Редактирование файла

Создадим исполняемый файл и запустим его. Мы видим, что программа запускает бесконечный цикл при нечетном значении N (РИС.5) и выводит только нечетные числа при четном значении N.(РИС.5\_1)

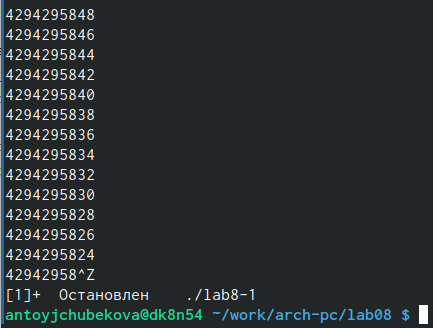


РИС.5 Запуск исполняемого файла

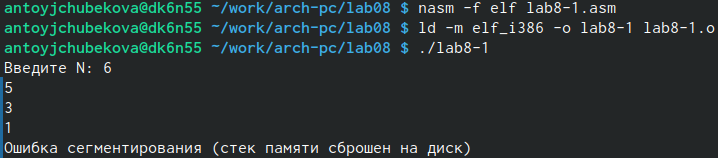


РИС.5\_1 Запус исполняемого файла

Для корректной работы программы внесем изменения в текст программы добавив push и pop для сохранения значения счетчика цикла loop.(РИС.6)

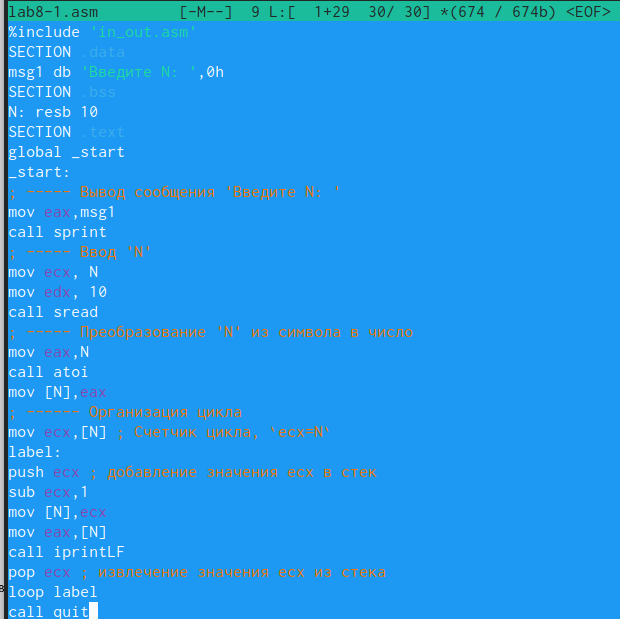


РИС.6 Редактирование файла

Создадим исполняемый файл и запустим его. Мы видим, что программа выводит числа от N-1 до 1, где количество проходов цикла соответствует значению N.(РИС.7)

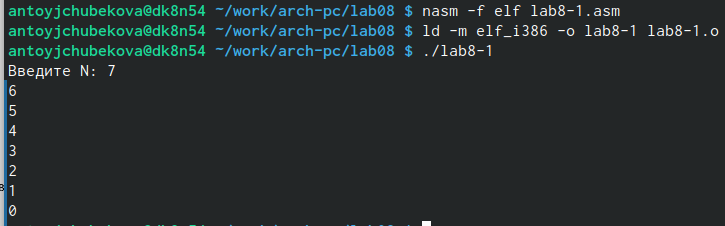


РИС.7 Запуск исполняемого файла

Создадим файл lab8-2.asm.(РИС.8) Внимательно изучив, запишем в этот файл программу выводящую на экран аргументы командной строки.(РИС.9)

РИС.8 Создание файла

РИС.8 Создание файла

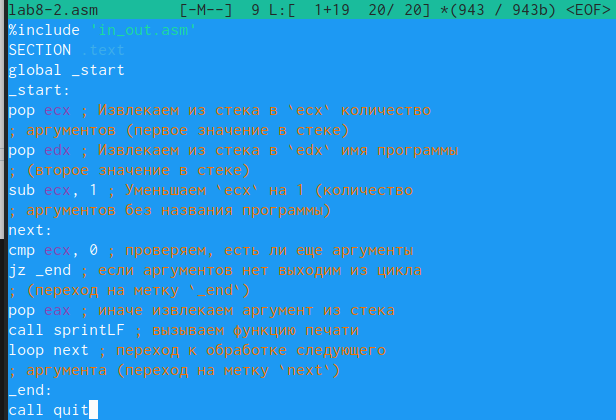


РИС.9 Редактирование файла

Создадим исполняемый файл и запустим его. Программа обработала 4 агрумента, так как аргумент 2 не был взят в кавычки, в отличии от агрумента 3, поэтому из за пробела программа берет 2 как отдельный аргумент.(РИС.10)

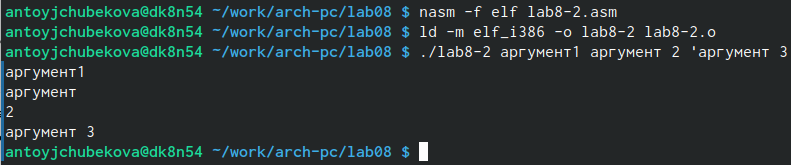


РИС.10 Запуск исполняемого файла

Создадим файл lab8-3.asm.(РИС.11) Внимательно изучив, запишем в него программу вычесления суммы аргументов командной строки.(РИС.11)

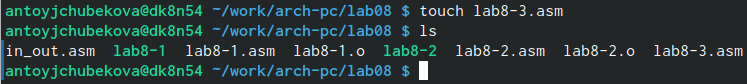


РИС.11 Создание файла

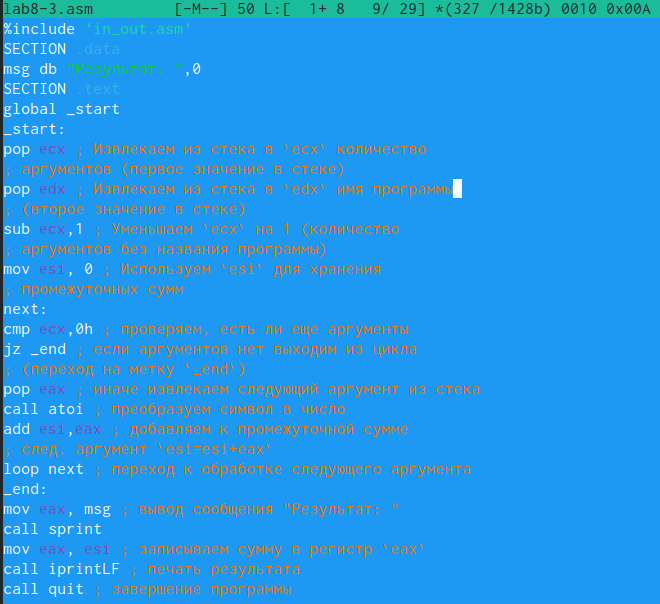


РИС.11\_1 Редактирование файла

Создадим исполняемый файл и запустим его. Проверим работу программы на значениях (12 13 7 10 5) и на значениях(7 4 6 9 10).Мы видим, что все работает правильно.(РИС.12)

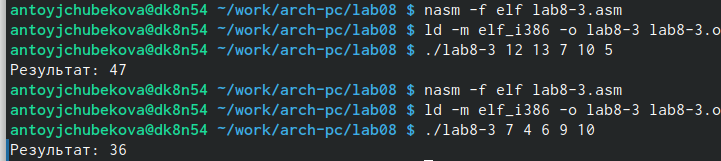


РИС.12 Запуск исполняемого файла

Изменим текст программы так, чтобы программа вычисляля произведения аргументов командной строки(РИС.13) Сама программа выглядит следующим образом:

%include ‘in\_out.asm’  
SECTION .data  
msg db “Результат:”,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, 1 ; 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  
mul esi  
mov esi,eax  
loop next ; переход к обработке следующего аргумента  
\_end: mov eax, msg ; вывод сообщения “Результат:”  
call sprint mov eax, esi ; записываем сумму в регистр eax  
call iprintLF ; печать результата  
call quit.

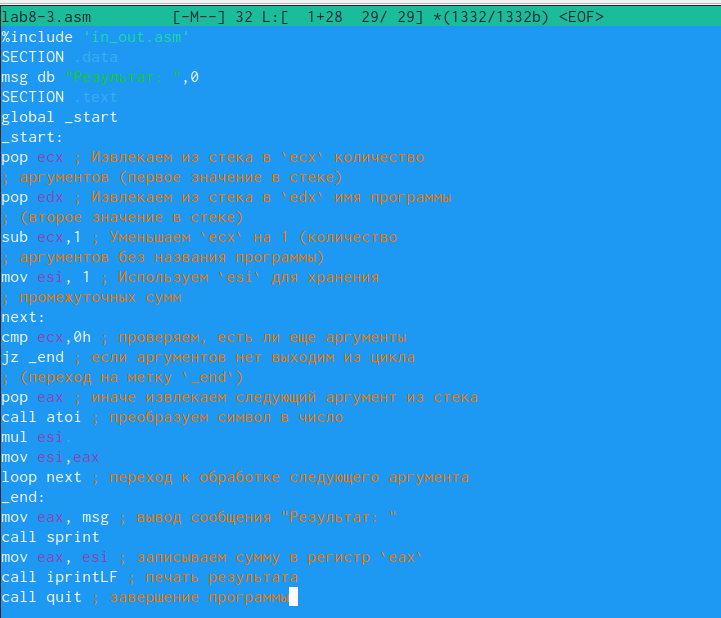


РИС.13 Редактирование файла

Создадим исполняемый файл и запустим его. Проверим работу программы на значениях (1 2 3 4 5). Мы видим, что все правильно работает(РИС.14)

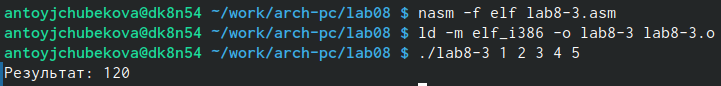


РИС.14 Запуск исполняемого файла

## 4.1 Задание для самостоятельной работы

Напишем программу которая находит сумму значений функции f(x)=7(x+1)(вариант 14) для x=x1,x2,…xn,то есть программа должна выводиь F(x1) + f(x2) + … + f(xn)(РИС.15) Сама программа выглядит следующим обрпзом:

%include ‘in\_out.asm’

SECTION .data  
msg db “Результат:”,0

SECTION .text  
global \_start  
\_start:

pop ecx

pop edx

sub ecx,1

mov esi, 0

next:  
cmp ecx,0h  
jz \_end

pop eax  
call atoi  inc eax  
mov ebx,7  
mul ebx  
add esi,eax  
loop next  
\_end:

mov eax, msg  
call sprint  
mov eax, esi  call iprintLF  call quit

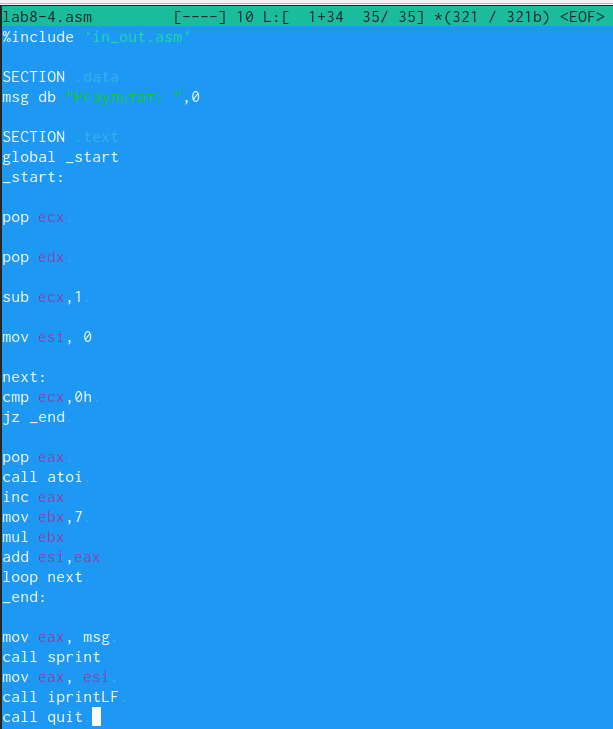


РИС.15 Редактирование файла

Создадим исполняемый файл и запустим его. Проверим корректность работы программы введя значения (1 2 3);(6 7 8);(4 3). Используя какулятор и проверив правильный ответ мы видим, что программа работает правильно.(РИС.16)

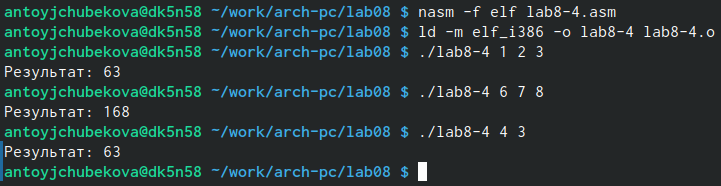


РИС.16 Запуск исполняемого файла

# 5 Выводы

В ходе выполнения лабораторной работы №8 я приобрела навыки написания программ с использованием циклов и обработкой аргументов командной строки. Применяя полученные навыки написала программу, которая наодит значение сумму значений функции f(x) для различных x.

# Список литературы

-https://esystem.rudn.ru/course/view.php?id=4975.