Отчёт по лабораторной работе №9

Программирование цикла. Обработка аргументов командной строки

Мулин Иван Владимирович

Содержание

# 1 Цель работы

По результатам данной лабораторной работы необходимо научиться программировать циклы и обрабатывать аргументы командной строки в языке программирования NASM. По адресу <https://github.com/ivmulin/study_2022-2023_arch-pc> расположен репозиторий github.

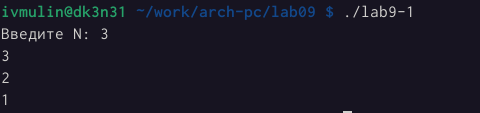
В ходе написания данного отчёта была незначительно изменена структура репозитория: написанные программы теперь хранятся в папке

.../labs/lab09/programs.

# 2 Ход работы

## 2.1 Выполнение лабораторной работы

Напишем и запустим программу lab9-1.asm, которая реализует простейший пример использования циклов

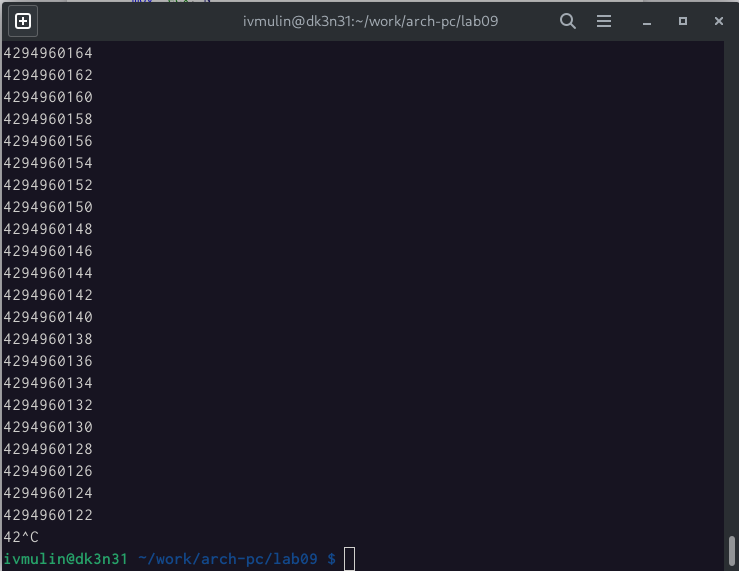


Запуск программы lab9-1.asm

При работе в цикле необходимо аккуратно обращаться со счётчиком - регистром ecx. Изменённая программа при помощи строки

sub ecx,1

изменяет поведение программы, беконечно выводя числа из памяти:



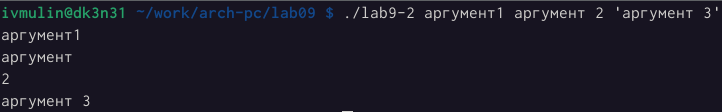
Запуск изменённой первой программы

Дело в том, что, когда указанной инструкцией из регистра вычитается единица и его значение оказывается равным нулю, команда loop также вычитает единицу, делая по итогу значение регистра ecx равным -1. Скорректируем программу так, чтобы внутри цикла можно было использовать регистр счётчика. Данная программа, как видно, работает корректно:



Работа вновь изменённой первой программы

Напишем вторую программу, обрабатывающую исключения:



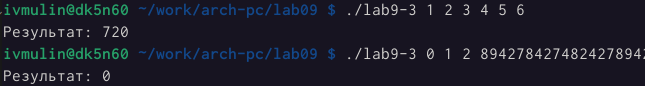
Запуск программы lab9-2.asm

Насколько видно из примера, программа вывела четыре введённых через пробел аргумента. Создадим программу lab9-3.asm такую, что она выводит сумму всех введённыз аргументов:

Сумма введённых аргументов

Сумма введённых аргументов

Изменим её так, чтобы она выводила произведение аргументов программы:

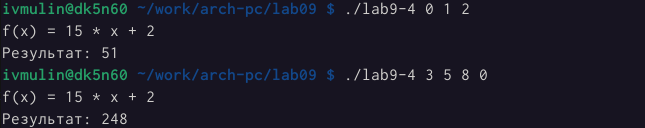


Произведение введённых аргументов

## 2.2 Выполнение заданий для самостоятельной работы

В качестве задания для самостоятельной работы нужно написать программу, вычисляющую сумму значений функции

в точках, соответствующим аргументам программы.



Сумма значений функции

Программа, очевидно, работает исправно.

# 3 Листинги написанных программ

1. lab9-1.asm

%include 'in\_out.asm'  
  
section .data  
 msg1 db 'Введите N: ',0h  
  
section .bss  
 N: resb 10  
  
section .text  
global \_start  
  
\_start:  
 mov eax,msg1  
 call sprint  
   
 mov ecx, N  
 mov edx, 10  
 call sread  
   
 mov eax,N  
 call atoi  
 mov [N],eax  
  
 mov ecx,[N]  
label:  
 push ecx  
 sub ecx,1  
 mov [N],ecx  
 mov eax,[N]  
 call iprintLF  
 pop ecx  
 loop label  
  
  
 call quit

1. lab9-2.asm

%include 'in\_out.asm'  
  
section .text  
 global \_start  
  
\_start:  
 pop ecx ; Извлекаем из стека в `ecx` количество аргументов  
 pop edx ; Извлекаем из стека в `edx` имя программы  
 sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество аргументов без названия программы)  
  
next:  
 cmp ecx, 0 ; проверяем, есть ли еще аргументы  
 jz \_end ; если аргументов нет, выходим из цикла  
  
 pop eax  
 call sprintLF  
 loop next  
  
\_end:  
 call quit

1. lab9-3.asm

%include 'in\_out.asm'  
  
section .data  
 msg db "Результат: ",0  
  
section .text  
 global \_start  
  
\_start:  
 pop ecx ; Извлекаем из стека в `ecx` количество аргументов  
 pop edx ; Извлекаем из стека в `edx` имя программы  
 sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество аргументов без названия программы)  
 mov esi, 1 ; используем 'esi' для хранения промежуточных произведений   
  
next:  
 cmp ecx, 0 ; проверяем, есть ли еще аргументы  
 jz \_end ; если аргументов нет, выходим из цикла  
  
 pop eax  
 call atoi  
 mul esi ; eax \*= eax  
 mov esi, eax  
  
 loop next  
  
\_end:  
 mov eax, msg  
 call sprint  
  
 mov eax, esi  
 call iprintLF  
  
 call quit

1. lab9-4.asm

%include 'in\_out.asm'  
  
; f(x) = 15x + 2  
  
section .data  
 msg db "Результат: ", 0  
 fun db "f(x) = 15 \* x + 2", 10  
  
section .bss  
 result resb 10  
  
section .text  
 global \_start  
  
\_start:  
 mov eax, fun  
 call sprint  
  
 pop ecx ; Извлекаем из стека в `ecx` количество аргументов  
 pop edx ; Извлекаем из стека в `edx` имя программы  
 sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество аргументов без названия программы)  
 mov esi, 0 ; используем 'esi' для хранения промежуточных сумм  
  
 mov eax, 0  
 mov [result], eax  
  
extractArguments:  
 cmp ecx, 0 ; проверяем, есть ли еще аргументы  
 jz \_end ; если аргументов нет, выходим из цикла  
  
 pop eax  
 call atoi  
  
 mov ebx, 15  
 mul ebx  
 add eax, 2  
 add [result], eax  
  
 loop extractArguments  
  
\_end:  
 mov eax, msg  
 call sprint  
  
 mov eax, [result]  
 call iprintLF  
  
 call quit

# 4 Заключение

При выполнении лабораторной работы № 9 цель, поставленная в начале данного отчета, была достигнута, оттого что были изучены программирование циклов и обработка аргументов командной строки.