Лабораторная работа №9. Программирование цикла. Обработка

Архитектура ЭВМ

Осокин Георгий Иванович НММбд-02-22

Содержание

# 1 Цель работы

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

# 2 Выполнение лабораторной работы

Создадим каталог для выполнения лабораторной, перейдем в него и введем листинг 9.1 в файл

%include 'in\_out.asm'  
  
SECTION .data  
  
msg1 db 'Введите N: ',0h  
  
SECTION .bss  
  
N: resb 10  
  
SECTION .text  
  
global \_start  
  
\_start:  
; ----- Вывод сообщения 'Введите N: '  
  
 mov eax,msg1  
 call sprint  
  
; ----- Ввод 'N'  
 mov ecx, N  
 mov edx, 10  
 call sread  
  
; ----- Преобразование 'N' из символа в число  
  
 mov eax,N  
 call atoi  
 mov [N],eax  
; ------ Организация цикла  
  
 mov ecx,[N] ; Счетчик цикла, `ecx=N`  
label:  
  
 mov [N],ecx  
 mov eax,[N]  
 call iprintLF ; Вывод значения `N`  
  
 loop label ; `ecx=ecx-1` и если `ecx` не '0'  
 ; переход на `label`  
 call quit

Запустим код и введем аргумент

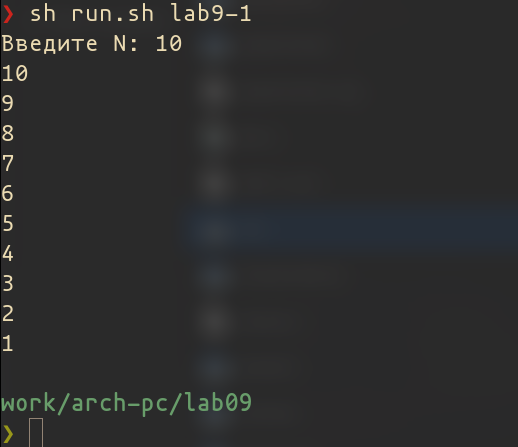


Рис. 1: Вывод программы с аргументом 10

Добавим строчку с уменьшением ecx на 1

sub ecx, 1 ; уменьшаем ECX на 1  
 mov [N],ecx  
 mov eax,[N]  
 call iprintLF ; Вывод значения `N`

Исполним программу.

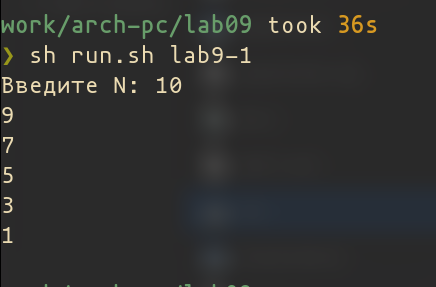


Рис. 2: Исполнение измененный программы с число 10

Обернем, данную часть кода в push, pop. И теперь посмотрим на результат.

push ecx  
 sub ecx, 1  
 mov [N],ecx  
 mov eax,[N]  
 call iprintLF ; Вывод значения `N`  
 pop ecx

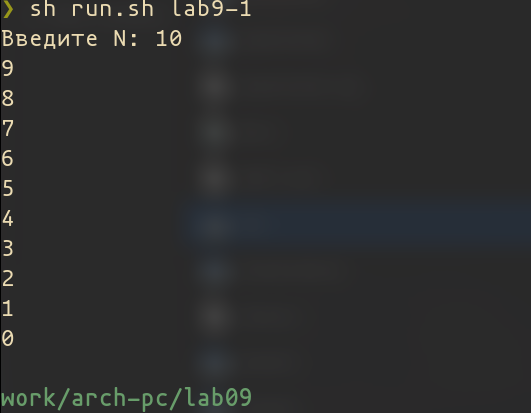


Рис. 3: Вывод измененной программы аргументом 10

## 2.1 Вывод на экран аргументов коммандной строки

Скопируем в файл lab9-2.asm листинг 9.2

%include 'in\_out.asm'  
  
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)  
next:  
 cmp ecx, 0 ; проверяем, есть ли еще аргументы  
 jz \_end ; если аргументов нет выходим из цикла  
 ; (переход на метку `\_end`)  
 pop eax ; иначе извлекаем аргумент из стека  
 call sprintLF ; вызываем функцию печати  
 loop next ; переход к обработке следующего  
 ; аргумента (переход на метку `next`)  
\_end:  
 call quit

Скомпилируем и запустим файл.

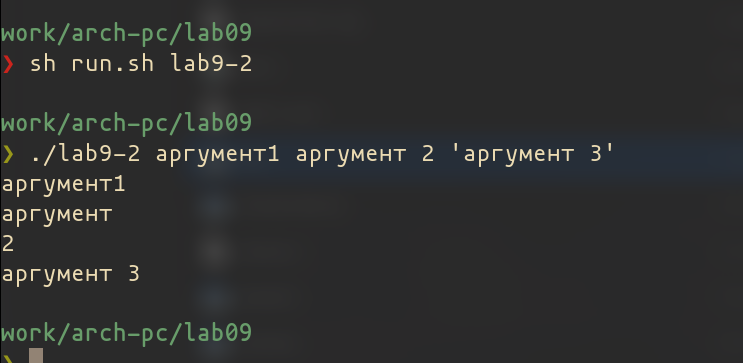


Рис. 4: Запуск lab9-2

Программа вывела 4 аргумента и вывела соответствующие строки.

## 2.2 Программа выводящая сумму аргументов

Скопируем программу из листинга **9.3** в 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 (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  
 add esi,eax ; добавляем к промежуточной сумме  
 ; след. аргумент `esi=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 ; завершение программы

Запустим программу с аргументами 12 13 7 10 5

(12 + 13 + 7 + 10 + 5 = 47)

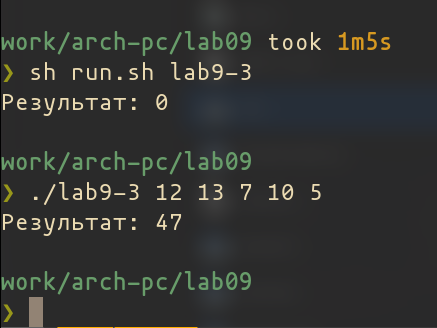


Рис. 5: Исполнение lab9-3

### 2.2.1 Умножение аргументов коммандной строки

Скопируем содержимое lab9-3.asm в lab9-3.mult.asm.

Изменим программу так, что бы она не складывала, а изменяла аргументы коммандной строки.

%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  
  
 push ecx  
 push eax  
  
 mov ecx, eax  
 mov eax, esi ; eax = res  
 mul ecx ; eax = eax \* ecx  
  
 mov esi, eax ; res = eax   
  
 pop eax  
 pop ecx  
  
 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 ; завершение программы

Запустим программу и введем аргументы 1 2 3 4 5 5 6 7 8

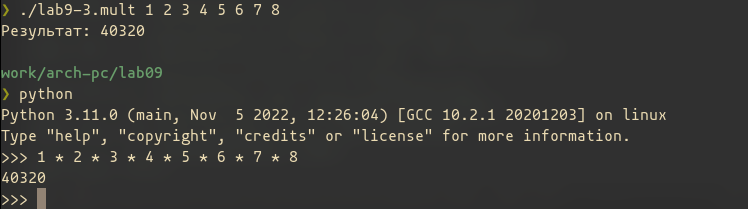


Рис. 6: Вывод программы lab9-3.mult

С помощью интерпретатора python убедимся, что результат вычеслений верный.

# 3 Задание для самостоятельной работы.

Так как наш вариант **18**, напишем программу которая будет вычеслять сумму функций от значений аргументов, введенных в коммандной строке.

%include 'in\_out.asm'  
%include 'my-function.asm'  
  
SECTION .data  
func db "Функция: f(x) = 5x + 17: ",0  
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, 0 ; Используем `esi` для хранения  
 ; промежуточных сумм  
 mov eax, func  
  
 call sprintLF  
  
  
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  
  
  
 call magic\_function  
  
 add esi,eax ; добавляем к промежуточной сумме  
  
 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 ; завершение программы

Мы подключили файл my-function.asm его содержимое - это наша функция.

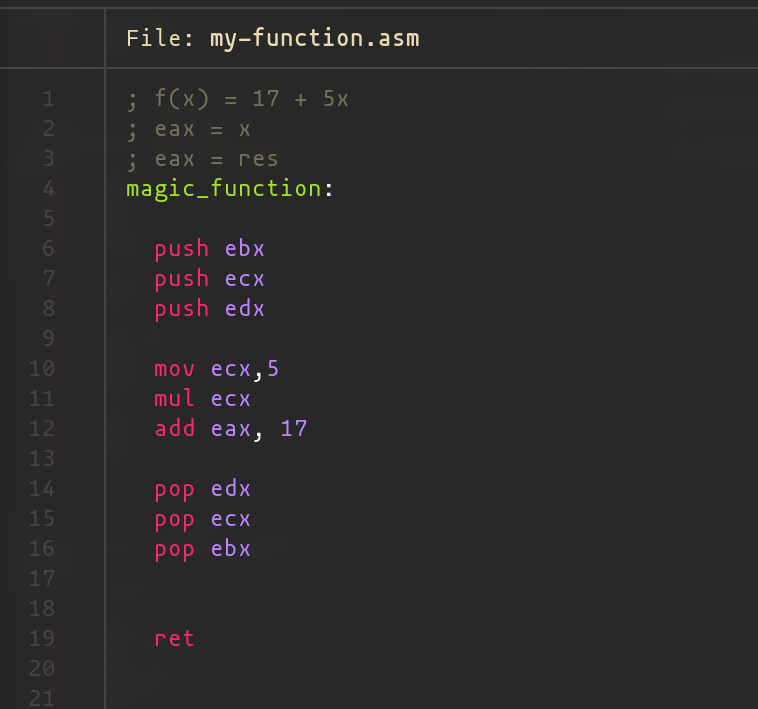


Рис. 7: Содержимое файла my-funciton.asm

Скомпилируем и запустим программу с аргументами 1 2 3 4.

Убедимся в правильности вычислений с помощьюу интерпретаторая python.

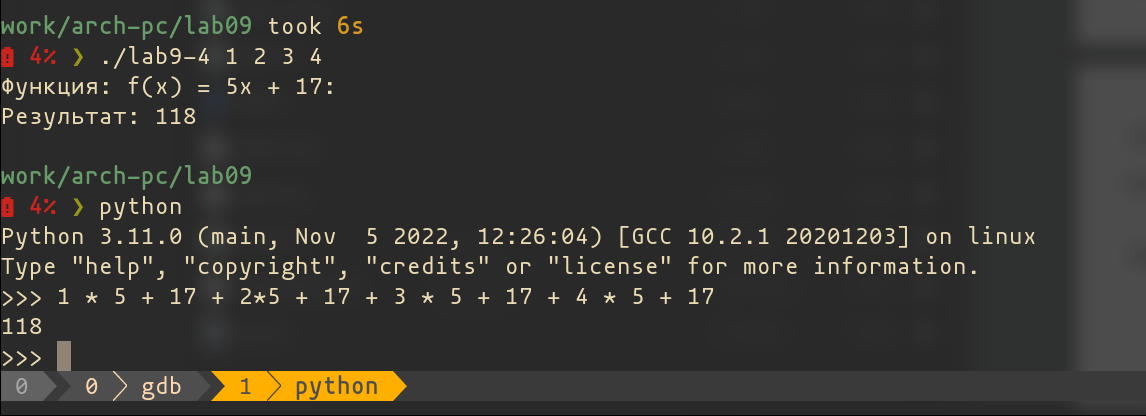


Рис. 8: Вывод программы lab9-4

# 4 Выводы

Мы приобрели навыки написание программ с использованием циклов и обработткой аргументов коммандной строки. Мы написали программы, которые умножают аргументы друг на друга, складывают и складывают значения функции от этих аргументов.