Лабораторная работа №8

Программирование цикла. Обработка аргументов командной строки.

Бочаров Андрей

Содержание

# 1 Цель работы

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

# 2 Выполнение лабораторной работы

Создал и першел в каталог для 8 лабораторной работы и командой touch создал файл lab7-8.asm (рис. 1).

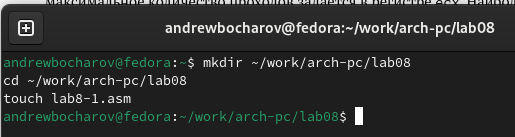


Рис. 1: Каталог lab08

Переписал код из листинга 8.1 (рис. 2).

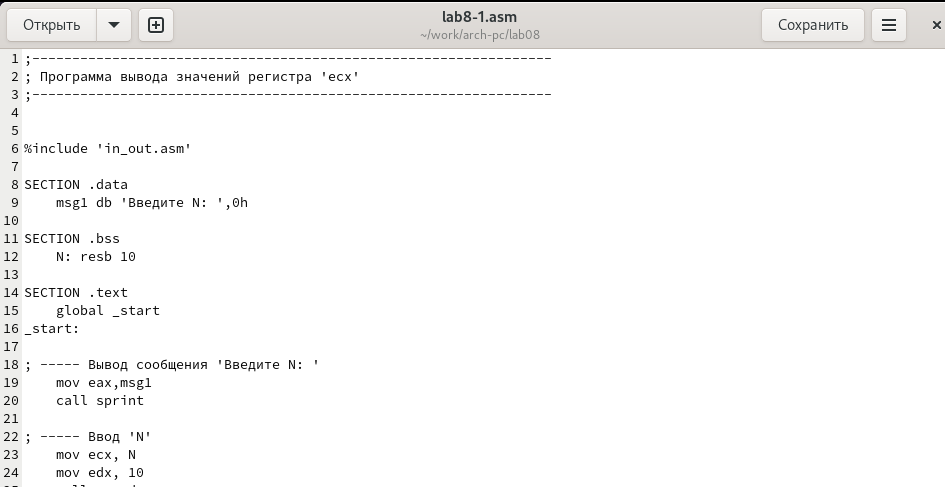


Рис. 2: Листинг кода

Листинг кода 8.1:

;-----------------------------------------------------------------  
; Программа вывода значений регистра 'ecx'  
;-----------------------------------------------------------------  
  
  
%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

Создал исполняемый файл и запустил его. (рис. 3).

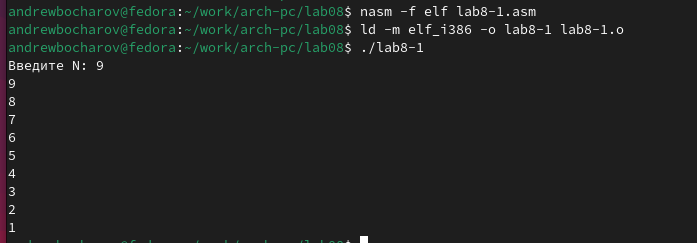


Рис. 3: Работает

Переписал код в файле с листинга 7.2 (рис. 4).

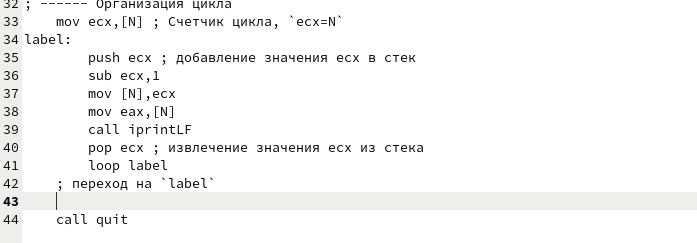


Рис. 4: Обновленный код

Листинг кода 8.1(Обновленный):

;-----------------------------------------------------------------  
; Программа вывода значений регистра 'ecx'  
;-----------------------------------------------------------------  
  
  
%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:  
 push ecx ; добавление значения ecx в стек  
 sub ecx,1  
 mov [N],ecx  
 mov eax,[N]  
 call iprintLF  
 pop ecx ; извлечение значения ecx из стека  
 loop label  
 ; переход на `label`  
   
 call quit

Создал исполняемый файл и запустил его. (рис. 5).

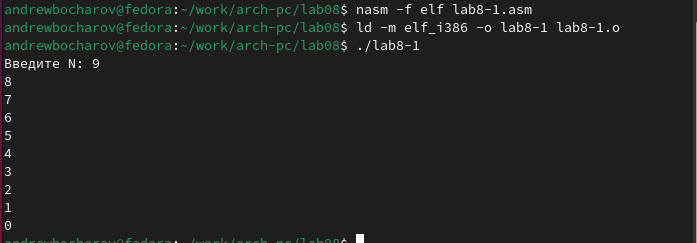


Рис. 5: Результат работы файла

Создал новый файл и переписал в него код с листинга 8.2 (рис. 6).

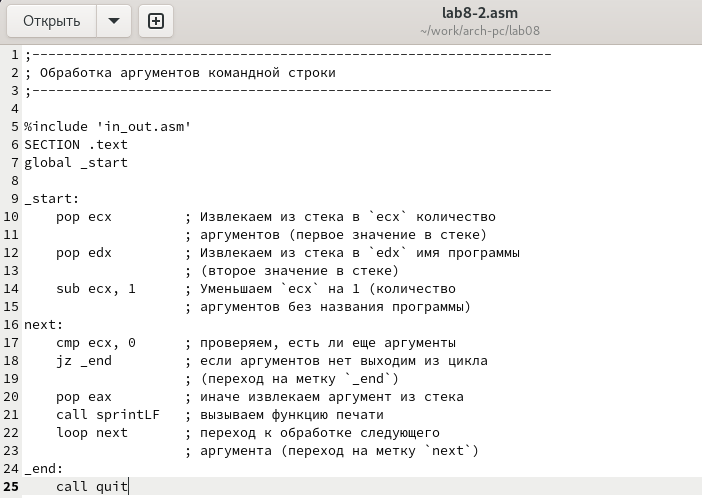


Рис. 6: Листинг кода

Листинг кода 8.2:

;-----------------------------------------------------------------  
; Обработка аргументов командной строки  
;-----------------------------------------------------------------  
  
%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

Создал исполняемый файл и запустил его. (рис. 7).

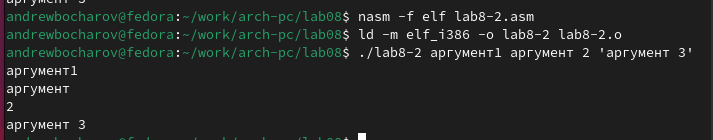


Рис. 7: Работает

Создал новый файл и переписал в него код с листинга 8.3 (рис. 8).

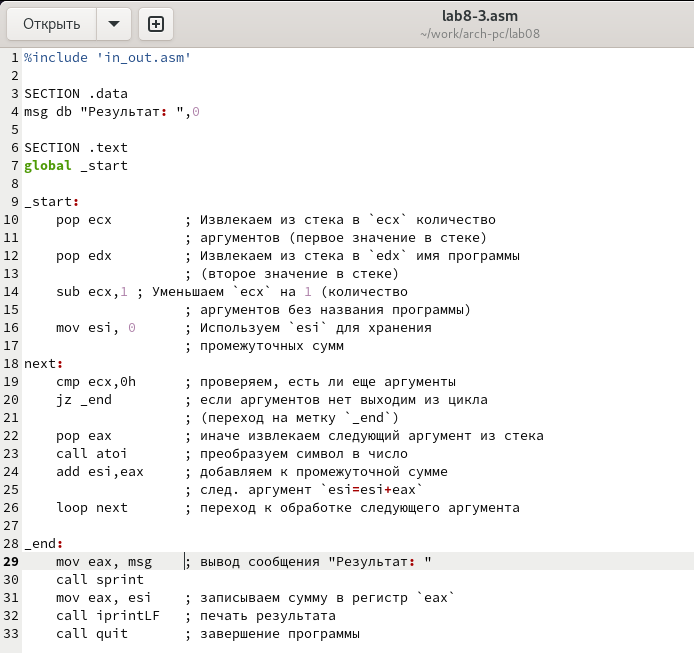


Рис. 8: Листинг кода

Листинг кода 8.3:

%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

Создал исполняемый файл и запустил его. Проверил с случайными введенными числыми. (рис. 9).

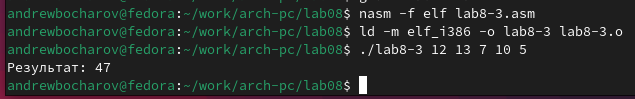


Рис. 9: Результат работы

# 3 Выполнение самостоятельной работы

Создал новый файл для самостоятельной работы, написал код, который выполняет задание соответсвтенно варианту 13 (рис. 10).

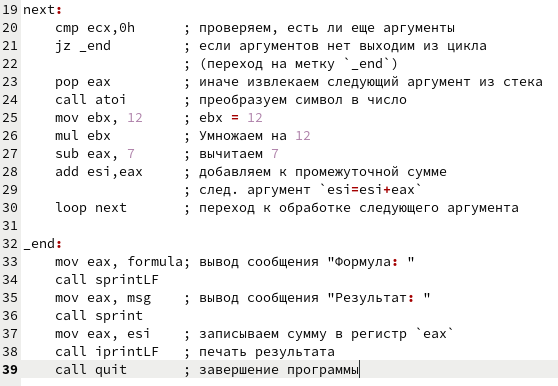


Рис. 10: Листинг кода

Запустил код, и проверил с различными аргументами командной строки (рис. 11).

Листинг кода самостоятельной работы:

%include 'in\_out.asm'  
  
SECTION .data  
msg db "Результат: ",0  
formula db "Формула: f(x)=12x-7",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  
 mov ebx, 12 ; ebx = 12  
 mul ebx ; Умножаем на 12  
 sub eax, 7 ; вычитаем 7  
 add esi,eax ; добавляем к промежуточной сумме  
 ; след. аргумент `esi=esi+eax`  
 loop next ; переход к обработке следующего аргумента  
   
\_end:  
 mov eax, formula; вывод сообщения "Формула: "  
 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

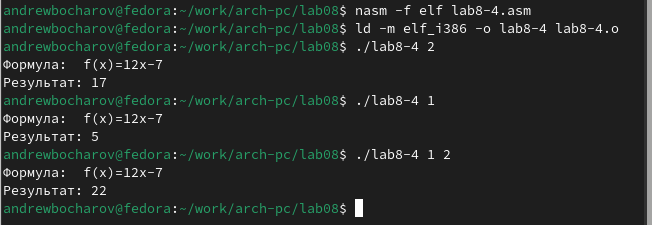


Рис. 11: Листинг кода

# 4 Выводы

Выполнив данную лабараторную работу, я обрел навыки написания программ с использованием циклов и обработкой аргументов командной строки.