Отчёт по лабораторной работе 8

Программирование цикла. Обработка аргументов командной строки.

Тукаев Тимур Ильшатович НММбд-03-23

Содержание

# 1 Цель работы

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

# 2 Выполнение лабораторной работы

1. Создал каталог для программам лабораторной работы № 8, перешел в него и создал файл lab8-1.asm
2. Написал в файл lab8-1.asm текст программы из листинга 8.1. Создал исполняемый файл и проверил его работу.

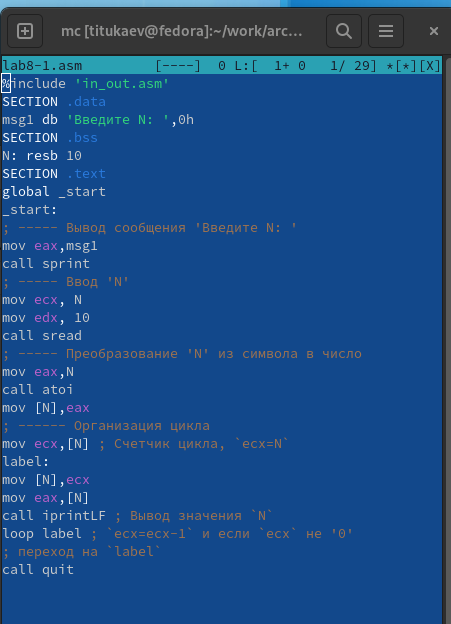


Figure 1: Программа в файле lab8-1.asm

Размещаю код программы в отчете

%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

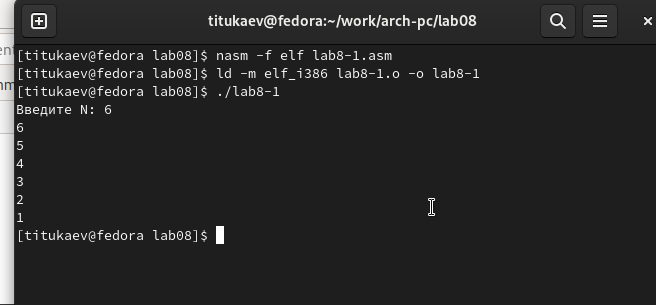


Figure 2: Запуск программы lab8-1.asm

1. Данный пример показывает, что использование регистра ecx в теле цилка loop может привести к некорректной работе программы. Изменил текст программы добавив изменение значение регистра ecx в цикле: Создайте исполняемый файл и проверьте его работу. Какие значения принимает регистр ecx в цикле? Соответствует ли число проходов цикла значению N, введенному с клавиатуры?

Программа запускает бесконечный цикл при нечетном N и выводит только нечетные числа при четном N.

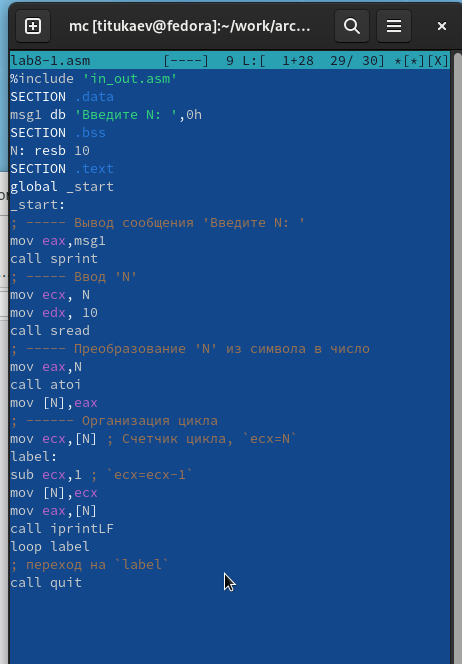


Figure 3: Программа в файле lab8-1.asm

Код программы

%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:  
sub ecx,1 ; `ecx=ecx-1`  
mov [N],ecx  
mov eax,[N]  
call iprintLF  
loop label  
; переход на `label`  
call quit

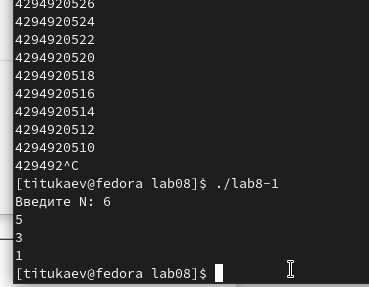


Figure 4: Запуск программы lab8-1.asm

1. Для использования регистра ecx в цикле и сохранения корректности работы программы можно использовать стек. Внеси изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop. Создал исполняемый файл и проверьте его работу. Соответствует ли в данном случае число проходов цикла значению N введенному с клавиатуры?

Программа выводит числа от N-1 до 0, число проходов цикла соответсвует N.

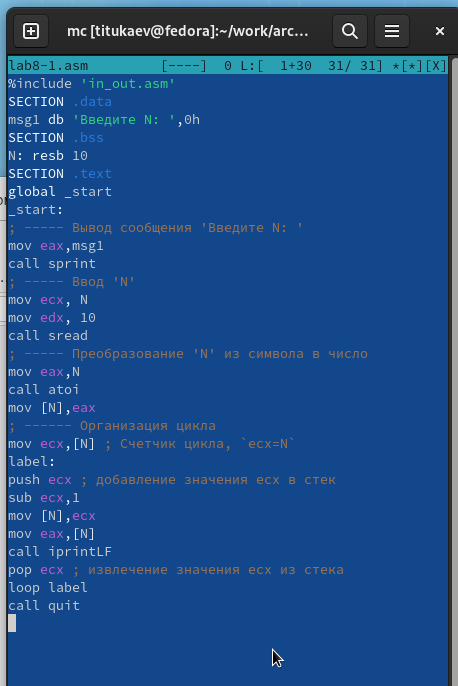


Figure 5: Программа в файле lab8-1.asm

Код программы

%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  
call quit

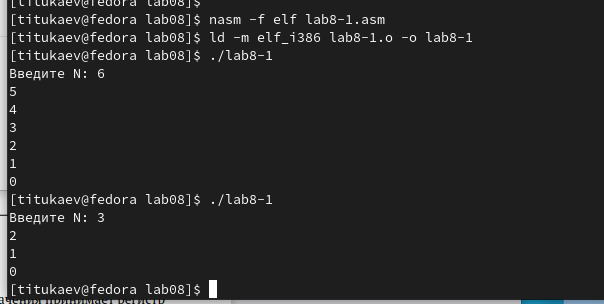


Figure 6: Запуск программы lab8-1.asm

1. Создал файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввел в него текст программы из листинга 8.2. Создал исполняемый файл и запустил его, указав аргументы. Сколько аргументов было обработано программой?

Программа обработала 5 аргументов.

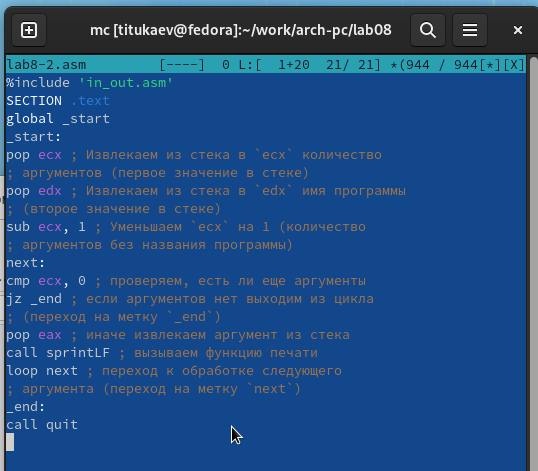


Figure 7: Программа в файле lab8-2.asm

Код программы

%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

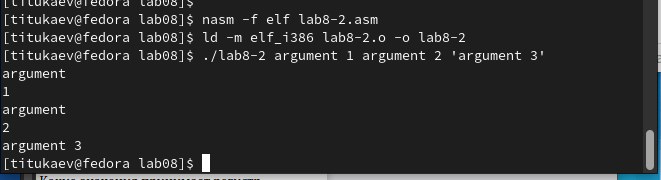


Figure 8: Запуск программы lab8-2.asm

1. Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы.

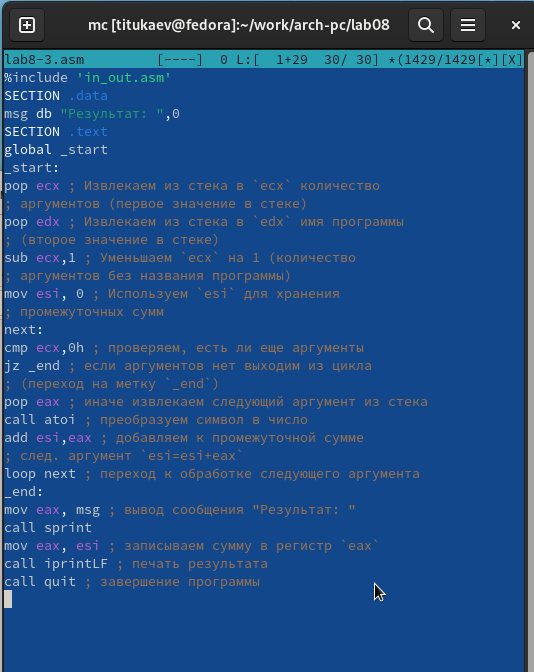


Figure 9: Программа в файле lab8-3.asm

Также размещаю код программы в отчете.

%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

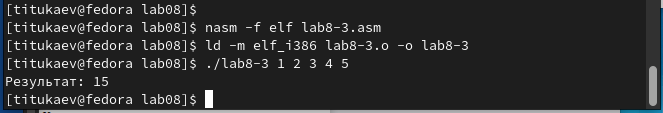


Figure 10: Запуск программы lab8-3.asm

1. Изменл текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.

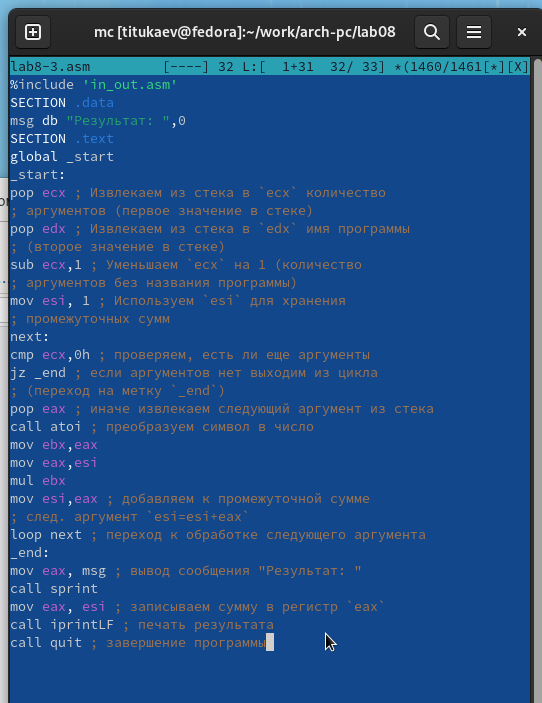


Figure 11: Программа в файле lab8-3.asm

Код программы

%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  
mov ebx,eax  
mov eax,esi  
mul ebx  
mov 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

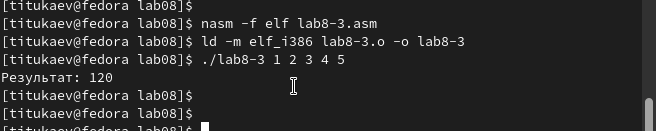


Figure 12: Запуск программы lab8-3.asm

1. 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 x передаются как аргументы. Вид функции f(x) выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах x.

для варивнта 5

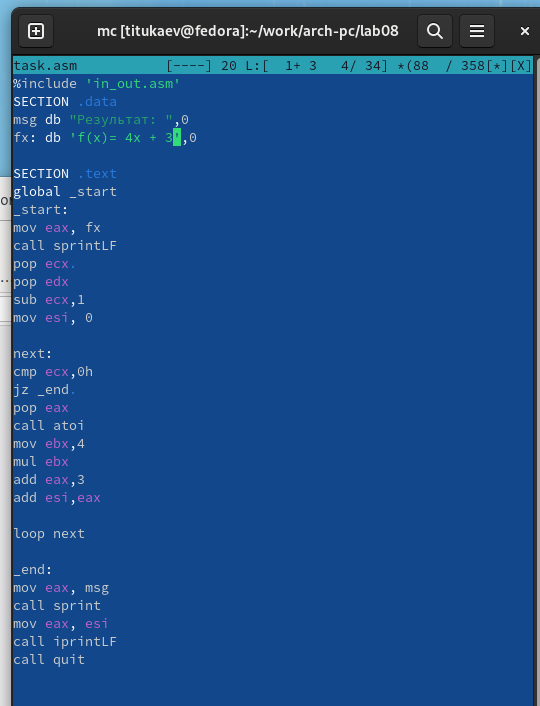


Figure 13: Программа в файле task.asm

Также размещаю код программы в отчете.

%include 'in\_out.asm'  
SECTION .data  
msg db "Результат: ",0  
fx: db 'f(x)= 4x + 3',0  
  
SECTION .text  
global \_start  
\_start:  
mov eax, fx  
call sprintLF  
pop ecx   
pop edx  
sub ecx,1  
mov esi, 0  
  
next:  
cmp ecx,0h  
jz \_end   
pop eax  
call atoi  
mov ebx,4  
mul ebx  
add eax,3  
add esi,eax  
  
loop next  
  
\_end:  
mov eax, msg  
call sprint  
mov eax, esi  
call iprintLF  
call quit

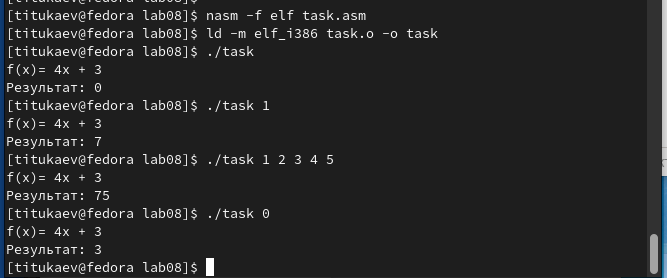


Figure 14: Запуск программы task.asm

# 3 Выводы

Освоили работы со стеком, циклом и аргументами на ассемблере nasm.