Лабораторная работа №8

Архитектура компьютера

Мурашов Иван Вячеславович

Содержание

# 1 Цель работы

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

# 2 Задание

1. Реализация циклов в NASM
2. Обработка аргументов командной строки
3. Выполнение заданий для самостоятельной работы

# 3 Выполнение лабораторной работы

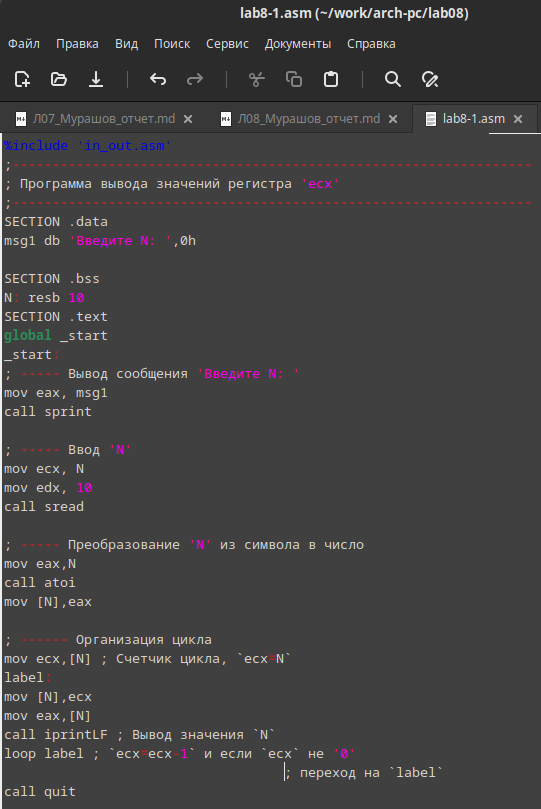
## 3.1 Реализация циклов в NASM

Создаю каталог для программам лабораторной работы №8, перехожу в него и создаю файл lab8-1.asm (рис. [??]).

Создание каталога и файла в нём

Создание каталога и файла в нём

Ввожу в файл lab8-1.asm текст программы из листинга 8.1 (рис. [??]).

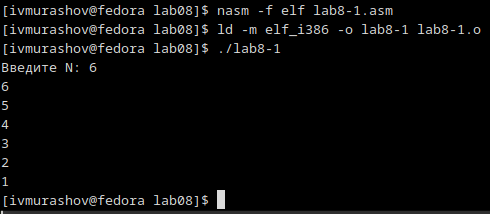


Редактирование файла

**Листинг 1. Программа вывода значений регистра ecx**

;-----------------------------------------------------------------  
; 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

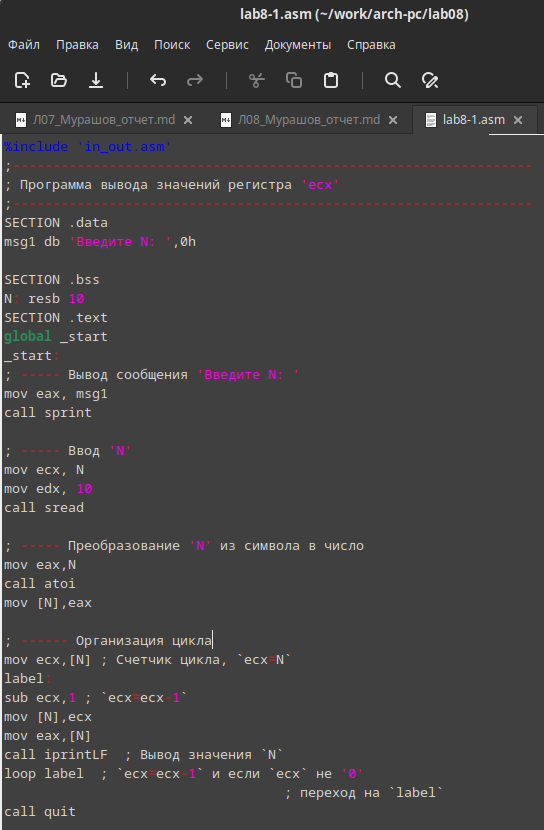
Создаю исполняемый файл и запускаю его (рис. [??]).



Трансляция, компоновка и запуск файлов

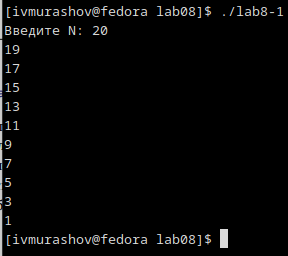
Данная программа выводит все числа от N до 1 включительно. Изменяю текст программы, добавив изменение значения регистра ecx в цикле: (рис. [??]).

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



Редактирование файла

Создаю исполняемый файл и запускаю его. При вводе чётного числа программа выводит все числа от N (не включая) до 1 с интервалом в 2 - то есть все нечётные числа, так как мы дважды вычитаем 1 из значения регистра ecx в ходе реализации цикла (рис. [??]).

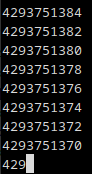


Трансляция, компоновка и запуск файлов

А при вводе нечётного числа программа выводит бесконечную последовательность значений поскольку значение регистра ecx переваливает за 0 (рис. [??]), (рис. [??]).

Трансляция, компоновка и запуск файлов

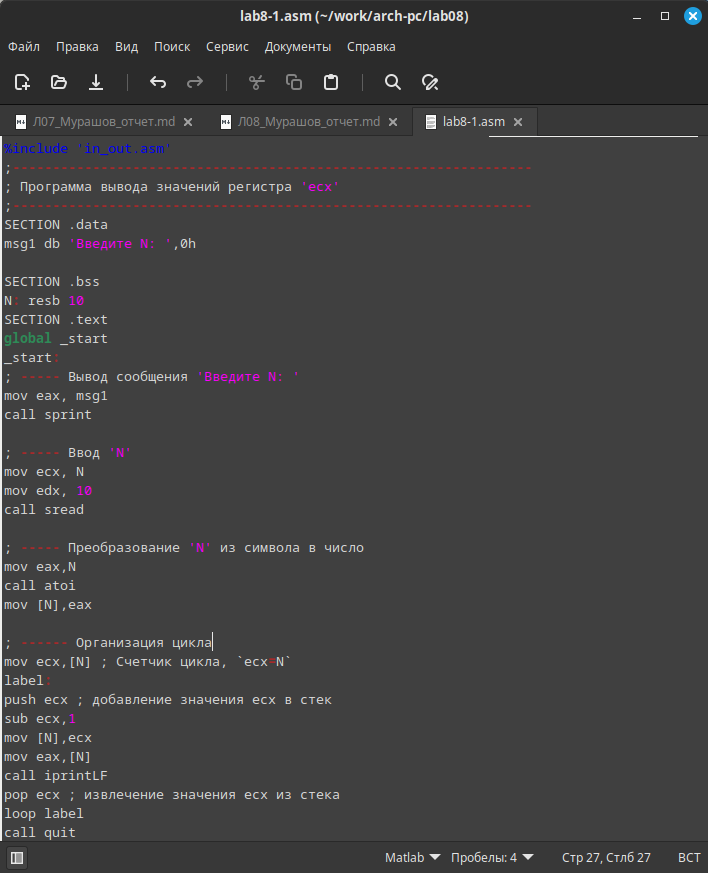
Трансляция, компоновка и запуск файлов



Трансляция, компоновка и запуск файлов

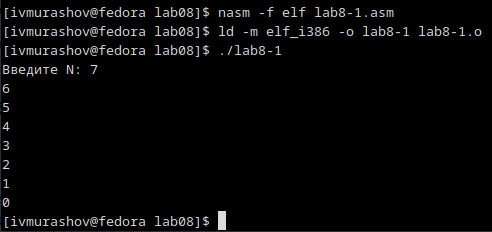
Вношу изменения в текст программы, добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop:

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



Редактирование файла

Создаю исполняемый файл и запускаю его (рис. [??]).



Трансляция, компоновка и запуск файлов

В данном случае число проходов цицкла соответствует значению N. Программа выводит значения от (N-1) до 0 включительно.

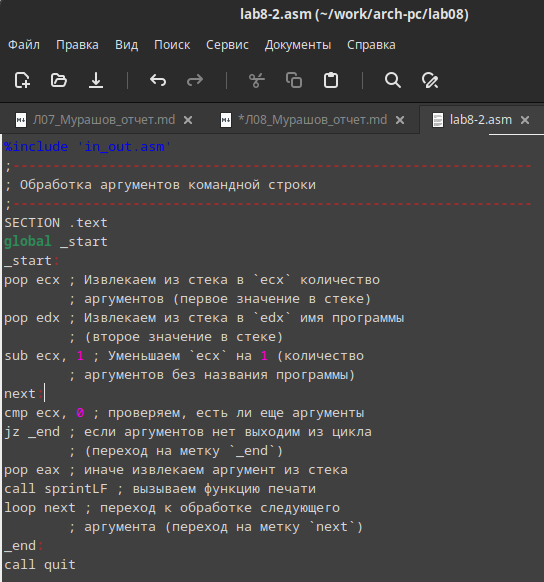
## 3.2 Обработка аргументов командной строки

Создаю файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 (рис. [??]).

Создание файла

Создание файла

Ввожу в файл lab8-2.asm текст программы из листинга 8.2 (рис. [??]).

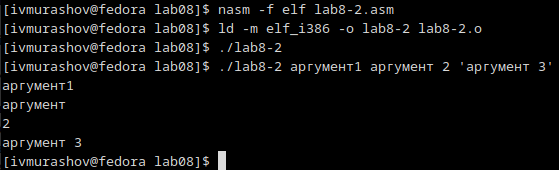


Редактирование файла

**Листинг 2. Программа выводящая на экран аргументы командной строки**

;-----------------------------------------------------------------  
; 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

Создаю исполняемый файл и запускаю его (рис. [??]).



Трансляция, компоновка и запуск файлов

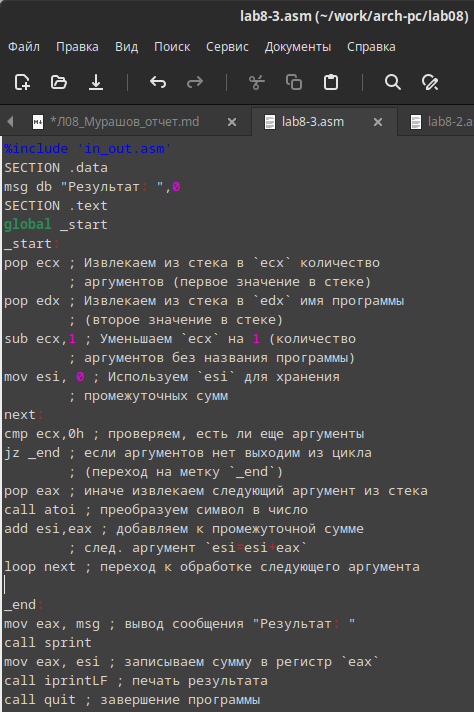
Программой был обработан 1 аргумент (выражение ‘аргумент 2’ воспринимается как 2 отдельных аргумента).

Создаю файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 (рис. [??]).

Создание файла

Создание файла

Ввожу в файл lab8-3.asm текст программы из листинга 8.3 (рис. [??]).

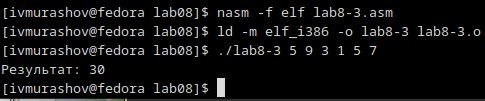


Редактирование файла

**Листинг 3. Программа вычисления суммы аргументов командной 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, 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

Создаю исполняемый файл и запускаю его (рис. [??]).



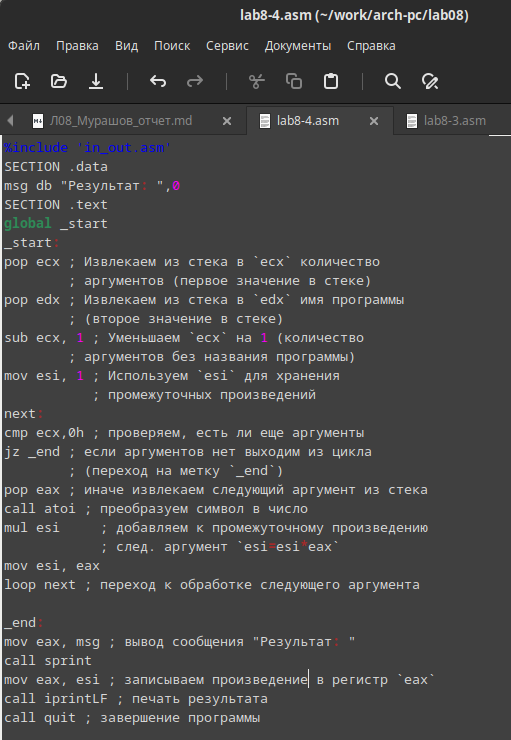
Трансляция, компоновка и запуск файлов

Создаю файл lab8-4.asm в каталоге ~/work/arch-pc/lab08 (рис. [??]).

Создание файла

Создание файла

Изменяю текст программы из листинга 8.3 для вычисления произведения аргументов командной строки (рис. [??]).

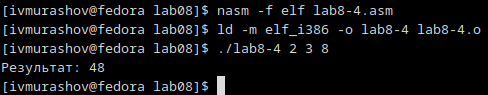


Редактирование файла

**Листинг 4. Программа вычисления произведения аргументов командной 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 произведений  
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 ; добавляем к промежуточному произведению  
 ; след. аргумент `esi=esi\*eax`  
mov esi, eax   
loop next ; переход к обработке следующего аргумента  
  
\_end:  
mov eax, msg ; вывод сообщения "Результат: "  
call sprint  
mov eax, esi ; записываем произведение в регистр `eax`  
call iprintLF ; печать результата  
call quit ; завершение программы

Создаю исполняемый файл и запускаю его, вводя несколько аргументов. При самостоятельном подсчёте полученные результаты совпали (рис. [??]).



Трансляция, компоновка и запуск файлов

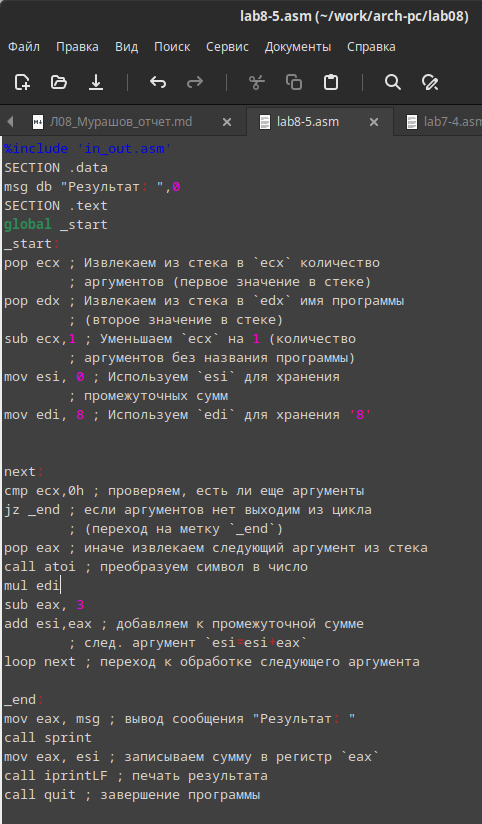
## 3.3 Выполнение заданий для самостоятельной работы

Создаю файл lab8-5.asm в каталоге ~/work/arch-pc/lab08 (рис. [??]).

Создание файла

Создание файла

Пишу программу, которая находит сумму значений функции f(x)=8x-3 (19 вариант) для x=x1, x2, x3,…xN, которые передаются как аргументы (рис. [??]).

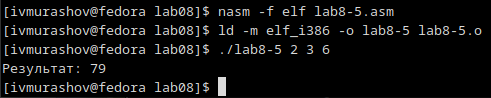


Редактирование файла

**Листинг 5. Программа вычисления суммы функций для аргументов командной строки**

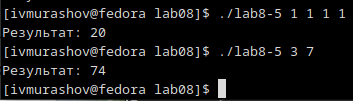
%include 'in\_out.asm'  
SECTION .data  
msg db "Результат: ",0  
SECTION .text  
global \_start  
\_start:  
pop ecx ; Извлекаем из стека в `ecx` количество  
 ; аргументов (первое значение в стеке)  
pop edx ; Извлекаем из стека в `edx` имя программы  
 ; (второе значение в стеке)  
sub ecx,1 ; Уменьшаем `ecx` на 1 (количество  
 ; аргументов без названия программы)  
mov esi, 0 ; Используем `esi` для хранения  
 ; промежуточных сумм  
mov edi, 8 ; Используем `edi` для хранения '8'  
  
  
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 edi  
sub eax, 3  
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

Создаю исполняемый файл и запускаю его, вводя следующий набор аргументов: 2, 3, 6. При самостоятельном подсчёте полученные результаты совпали (рис. [??]).



Трансляция, компоновка и запуск файлов

Проверяю работу программы на других наборах аргументов. При самостоятельном подсчёте полученные результаты также совпали (рис. [??]).



Трансляция, компоновка и запуск файлов

# 4 Выводы

В ходе выполнения данной лабораторной работы я приобрёл навыки написания программ с использованием циклов и обработкой аргументов командной строки.