Отчёт по лабораторной работе №8

дисциплина: Архитектура компьютера

Маслова Анна Павловна

Содержание

# 1 Цель работы

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

# 2 Выполнение лабораторной работы

Для начала создаём каталог для программ лабораторной работы №8, перейдём в него и создадим файл *lab8-1.asm* (рис. [1](#fig:001)).

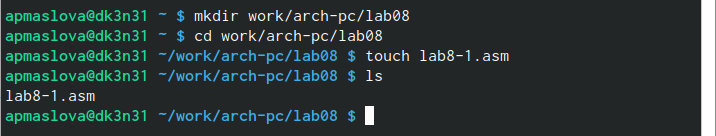


Figure 1: Создание каталога lab08 и файла lab8-1.asm

Введём в файл *lab8-1.asm* текст программы вывода значений регистра *ecx* (рис. [2](#fig:002)).

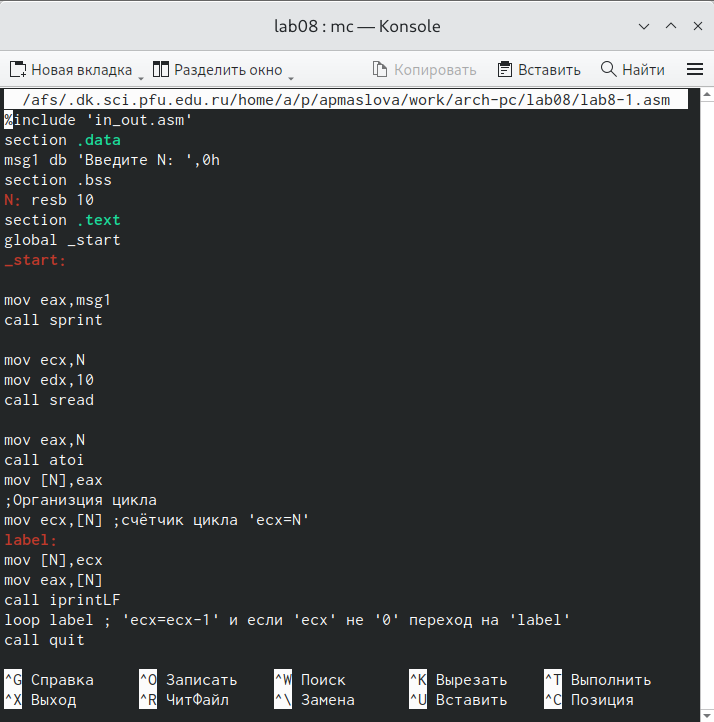


Figure 2: Текст файла lab8-1.asm

Создадим исполняемый файл и проверим его работу (рис. [3](#fig:003)).

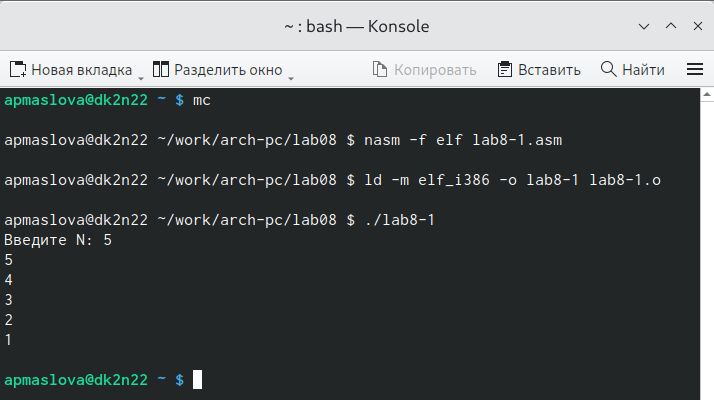


Figure 3: Создание исполняемого файла lab8-1

На экран вывелись значения от *N* до 1.

Данный пример показывает, что использование регистра *ecx* в теле цикла *loop* может привести к некорректной работе программы. Подкорректируем текст программы добавив изменение значения регистра *ecx* в цикле (рис. [4](#fig:004)).

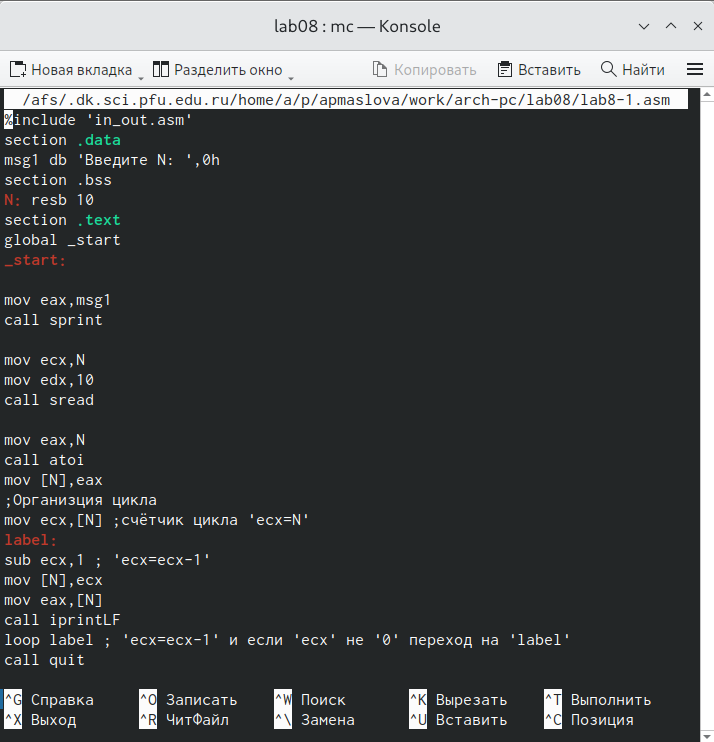


Figure 4: Исправление текста программы в файле lab8-1.asm

Создадим исполняемый файл и проверим его работу (рис. [5](#fig:005)).

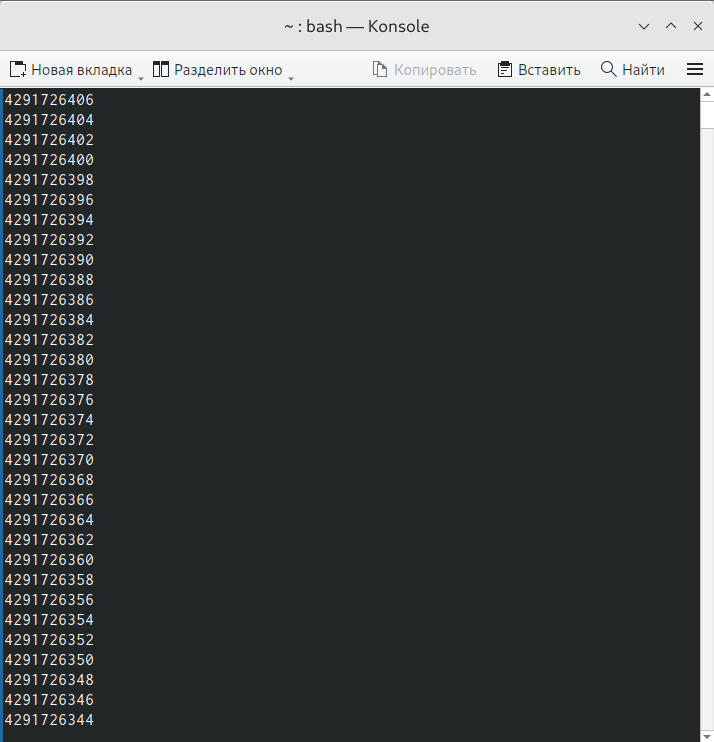


Figure 5: Запуск изменённого исполняемого файла lab8-1

Как мы видим, в этом случае программа работает долго, число проходов гораздо больше заявленного *N*. Регистр *ecx* принимает большие чётные значения, меньшие, но близкие к 5000000000.

Для использования регистра *ecx* в цикле и сохранения корректности работы программы можно использовать стек. Внесём изменения в текст программы добавив команды *push* и *pop* (добавления в стек *push* и извлечения из стека *pop*) для сохранения значения счетчика цикла *loop* (рис. [6](#fig:006)).

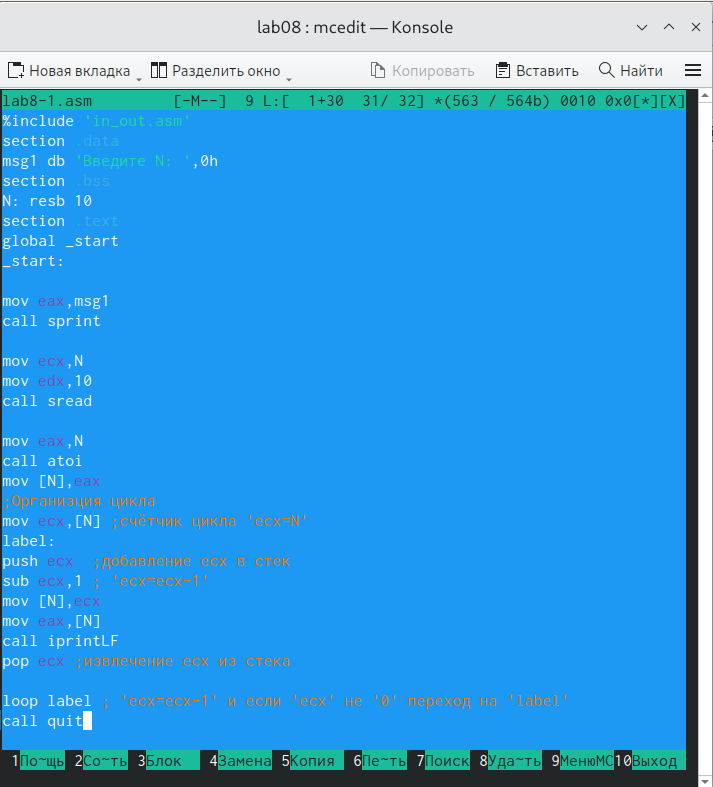


Figure 6: Добавление команд push и pop в текст файла lab8-1.asm

Создадим исполняемый файл и проверим его работу (рис. [7](#fig:007)).

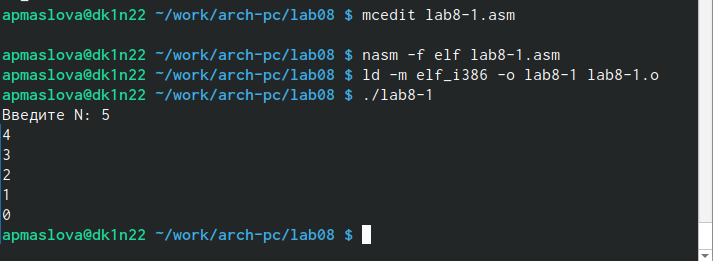


Figure 7: Запуск файла lab8-1 с командами push и pop

В данном случае число проходов цикла соответствует введённому числу *N*. На экран вывелись числа от *N-1* до 0.

Далее познакомимся с другой программой. Создадим файл *lab8-2.asm* в каталоге ~/work/arch-pc/lab08 и введём в него текст программы, которая выводит на экран аргументы командной строки (рис. [8](#fig:008)).

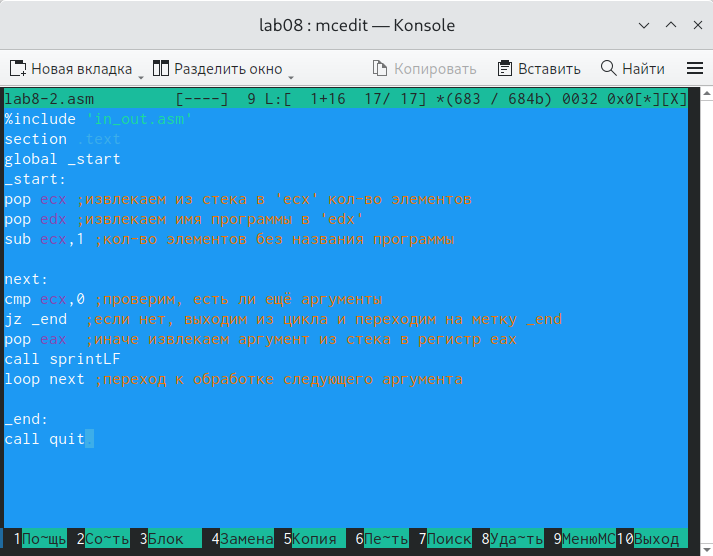


Figure 8: Текст программы в файле lab8-2.asm

Создаём исполняемый файл и запустим его (рис. [9](#fig:009)), указав следующие аргументы:

apmaslova@dk1n22:~$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'

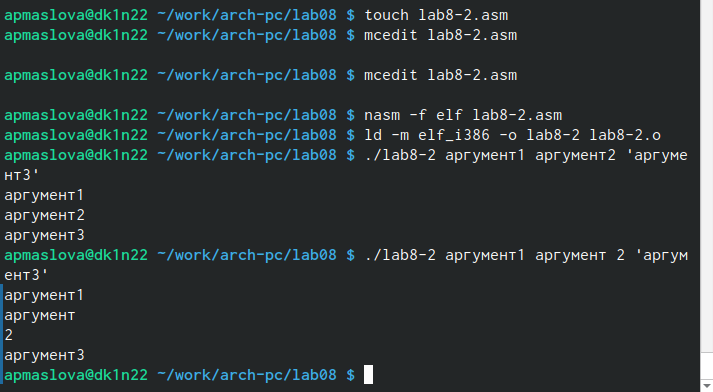


Figure 9: Создание и запуск исполняемого файла lab8-2

Программа обработала 4 аргумента, т.к. аргумент 2 был прочитан как аргумент и 2, т.е. разделён на два разных аргумента, которые поочередно были выведены на экран.

Рассмотрим следующую программу. Создадим файл *lab8-3.asm* в каталоге ~/work/arch-pc/lab08 и введём в него текст программы, которая выводит сумму чисел, которые передаются в программу как аргументы (рис. [10](#fig:010)).

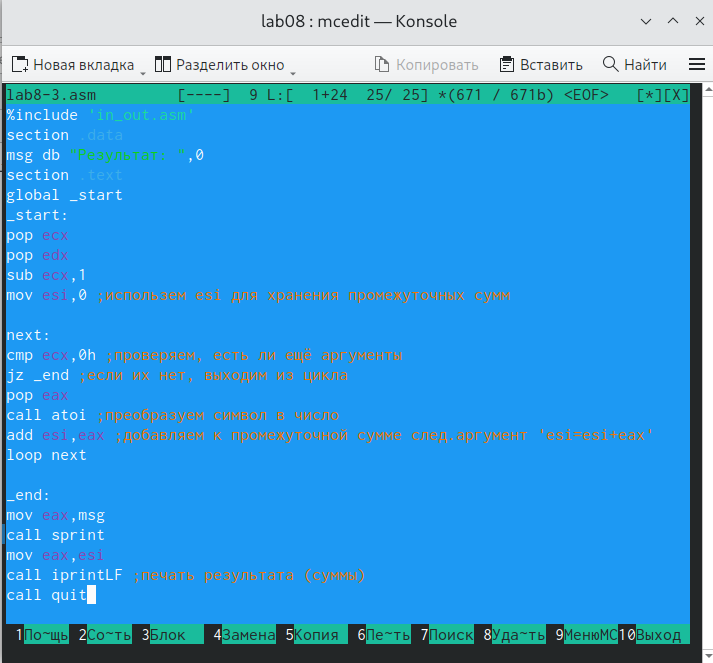


Figure 10: Текст программы в файле lab8-3.asm

Создадим исполняемый файл и запустим его, указав аргументы (рис. [11](#fig:011)).

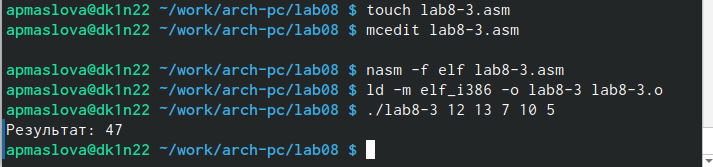


Figure 11: Создание и запуск исполняемого файла lab8-3

Программа работает корректно: выводит на экран сумму аргументов командной строки.

А теперь изменим текст этой программы для вычисления произведения аргументов командной строки. Введём в файл *lab8-3.asm* текст из рис. [12](#fig:012) :

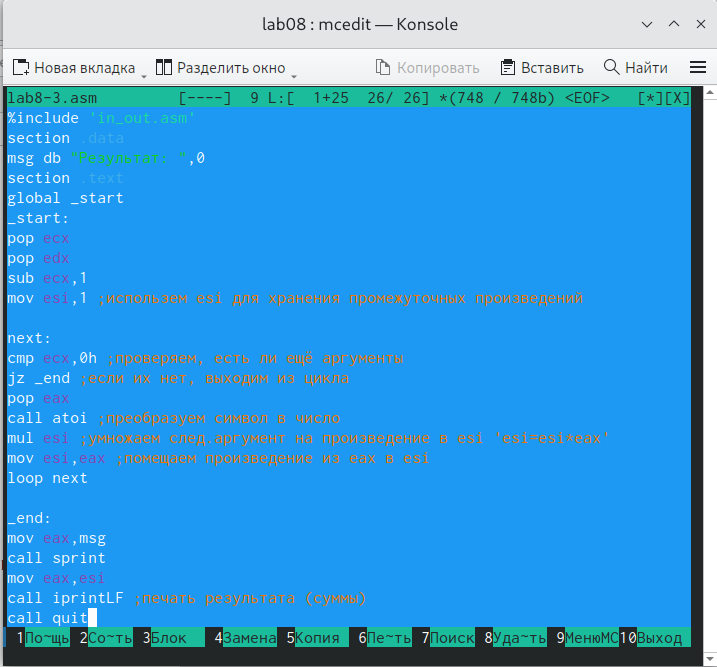


Figure 12: Изменённый текст программы в файле lab8-3.asm

Создадим исполняемый файл и запустим его, указав аргументы (рис. [13](#fig:013)).

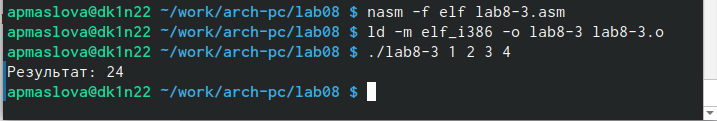


Figure 13: Создание и запуск изменённого исполняемого файла lab8-3

Как мы видим, программа верно вычисляет произведение аргументов командной строки.

# 3 Выполнение заданий для самостоятельной работы

Напишем программу, которая находит сумму значений функции для . То есть программа должна выводить значения .

Варианту №15 соответствует следующая функция:

В том же каталоге создадим файл *func.asm* и внесём в него текст программы из листинга 8.4 (рис. [14](#fig:014)).

**Листинг 8.4. Программа, вычисления суммы значений функции**

%include 'in\_out.asm'  
  
section .data  
msg db "Результат: ",0  
  
section .text  
global \_start  
\_start:  
  
pop ecx  
pop edx  
sub ecx,1  
mov esi,0 ;используем esi для хранения промежуточных сумм f(x)  
  
;6x+13  
next:  
cmp ecx,0h ;проверяем, есть ли ещё аргументы  
jz \_end ;если их нет, выходим из цикла  
pop eax  
call atoi ;преобразуем символ в число  
mov ebx,6  
mul ebx ;умножаем след.аргумент на 6 'eax=eax\*6'  
add eax,13 ;eax=eax+13  
add esi,eax ;помещаем значение f(x) из eax в esi  
loop next  
  
\_end:  
mov eax,msg  
call sprint  
mov eax,esi  
call iprintLF ;печать результата (суммы)  
  
call quit

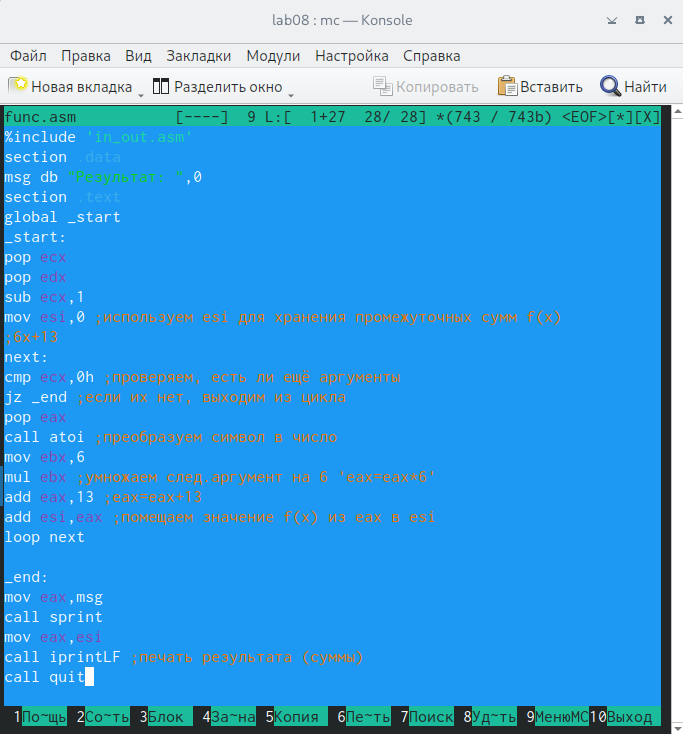


Figure 14: Текст программы в файле func.asm

Создадим исполняемый файл и несколько раз запустим его, указав аргументы (рис. [15](#fig:015)).

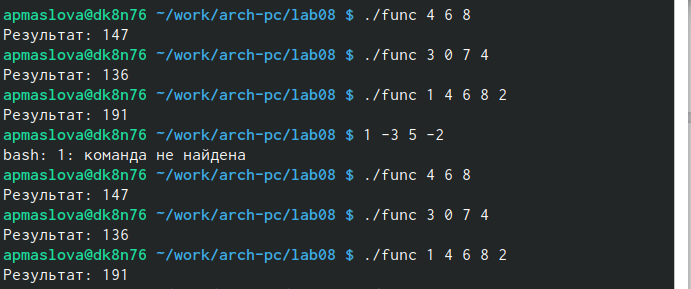


Figure 15: Создание и запуск исполняемого файла func.asm

Как мы видим, программа работает корректно и верно считает сумму значений функции.

# 4 Выводы

Мы научились писать программы использованием циклов и обработкой аргументов командной строки.

# Список литературы

1. GDB: The GNU Project Debugger. — URL: https://www.gnu.org/software/gdb/.
2. GNU Bash Manual. — 2016. — URL: https://www.gnu.org/software/bash/manual/.
3. Midnight Commander Development Center. — 2021. — URL: https://midnight-commander. org/.
4. NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O’Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: http://www.amazon.com/Learning- bash-Shell-Programming-Nutshell/dp/0596009658.
6. Robbins A. Bash Pocket Reference. — O’Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс,