Отчёт по лабораторной работе №8

Дисциплина: Архитектура компьютера

Абакумов Тимофей Александрович

Содержание

# 1 Цель работы

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

# 2 Задание

**Порядок выполнения лабораторной работы**

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

1. Для начала создадим каталог для программам лабораторной работы № 8, перейдём в него и создадим файл lab8-1.asm (рис. 1).

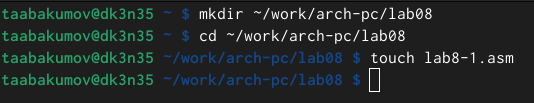


Рис. 1: Перемещение между директориями

1. Введём в файл lab8-1.asm текст программы из листинга 8.1 (рис. 2).

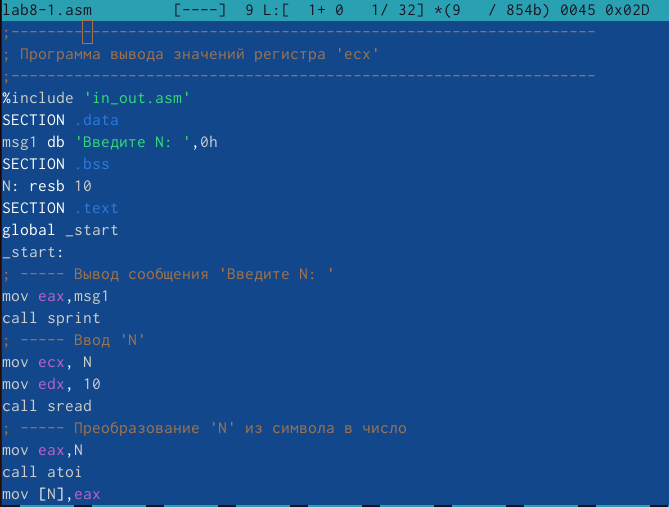


Рис. 2: Код программы

Код программы из пункта 2:

%include “in\_out.asm”

SECTION .data

msg db ‘введите N:’, 0h

SECTION .bss

N: resb 10

SECTION .text

global \_start

\_start:

mov eax,msg

call sprint

mov ecx,N

mov edx,10

call sread

mov eax,N

call atoi

mov [N],eax

mov ecx,[N]

label:

mov [N],ecx

mov eax, [N]

call iprintLF

loop label

call quit

1. Создадим исполняемый файл и проверим его работу (рис. 3).

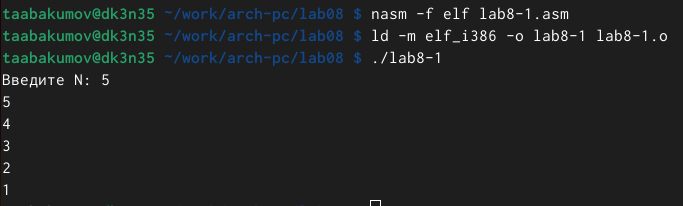


Рис. 3: Работа программы

1. Данный пример показывает, что использование регистра ecx в теле цилка loop может привести к некорректной работе программы. Изменим текст программы добавив изменение значение регистра ecx в цикл (рис. 4).

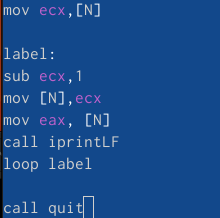


Рис. 4: Изменение программы

1. Создадим исполняемый файл и проверим его работу (рис. 5).

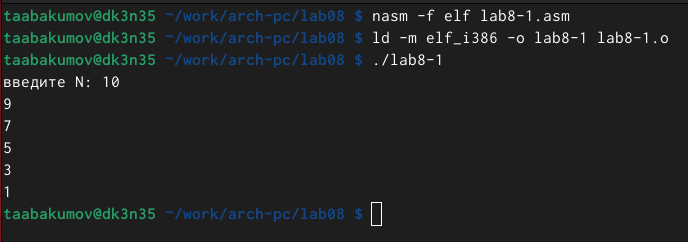


Рис. 5: Работа файла

Код программы из пункта 5:

%include “in\_out.asm”

SECTION .data

msg db ‘введите N:’, 0h

SECTION .bss

N: resb 10

SECTION .text

global \_start

\_start:

mov eax,msg

call sprint

mov ecx,N

mov edx,10

call sread

mov eax,N

call atoi

mov [N],eax

mov ecx,[N]

label:

sub ecx,1

mov [N],ecx

mov eax, [N]

call iprintLF

loop label

call quit

1. 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 и извлечения из стека) для сохранения значения счетчика цикла loop (рис. 6).

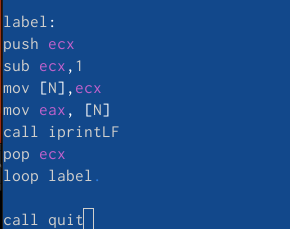


Рис. 6: Изменение программы

Код программы из пункта 6:

%include “in\_out.asm”

SECTION .data

msg db ‘введите N:’, 0h

SECTION .bss

N: resb 10

SECTION .text

global \_start

\_start:

mov eax,msg

call sprint

mov ecx,N

mov edx,10

call sread

mov eax,N

call atoi

mov [N],eax

mov ecx,[N]

label:

push ecx

sub ecx,1

mov [N],ecx

mov eax,[N]

call iprintLF

pop ecx

loop label

1. Создадим исполняемый файл и проверим его работу (рис. 7).

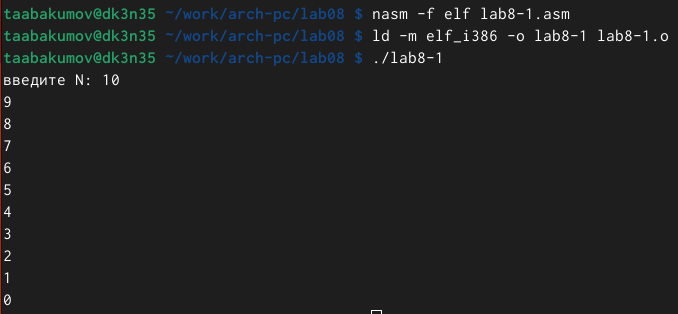


Рис. 7: Работа файла

## 3.2 Обработка аргументов командной строки

1. Создадим файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введём в него текст программы из листинга 8.2 (рис. 8).

Рис. 8: Создание файла

Рис. 8: Создание файла

1. Введём в него текст программы из листинга 8.2 (рис. 9).

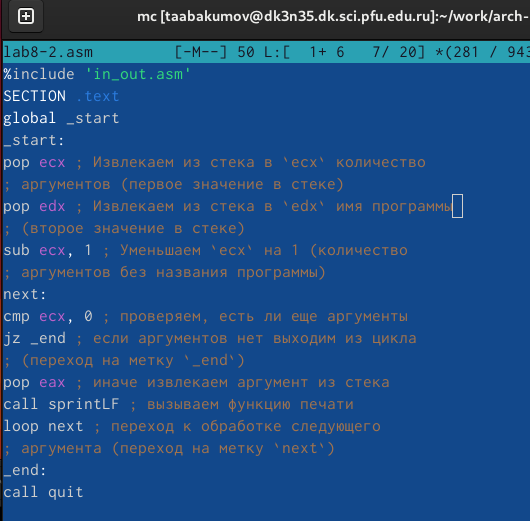


Рис. 9: Код программы

Код программы из пункта 9:

%include ‘in\_out.asm’

SECTION .text

global \_start

\_start:

pop ecx

pop edx

sub ecx, 1

next:

cmp ecx, 0

jz \_end

pop eax

call sprintLF

loop next

\_end:

call quit

1. Создадим исполняемый файл и запустим его, указав аргументы (рис. 10).

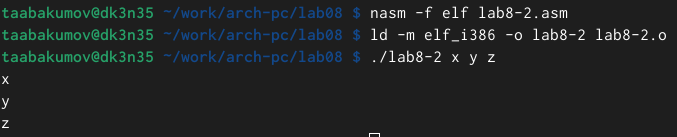


Рис. 10: Работа файла

1. Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы. Создадим файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 (рис. 11).

Рис. 11: Создание файла

Рис. 11: Создание файла

1. Введём в него текст программы из листинга 8.3 (рис. 12).

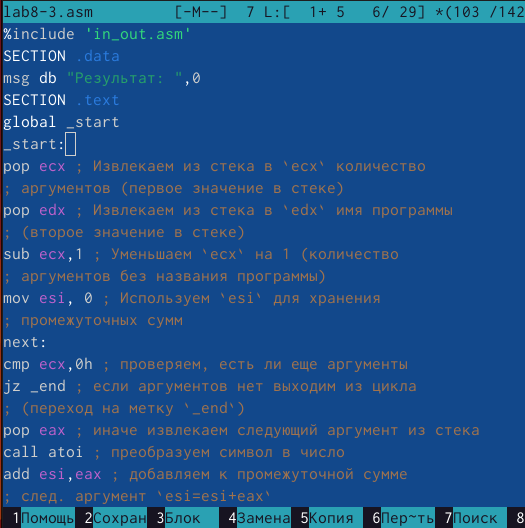


Рис. 12: Код программы

Код программы из пункта 12:

%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

next:

cmp ecx,0h

jz \_end

pop eax

call atoi

add esi,eax

loop next

\_end:

mov eax, msg

call sprint

mov eax, esi

call iprintLF

call quit

1. Создадим исполняемый файл и запустим его, указав аргументы (рис. 13).

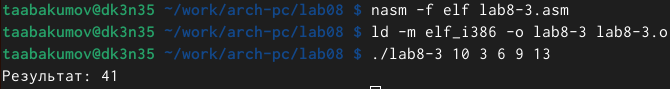


Рис. 13: Работа файла

1. Изменим текст программы из листинга 8.3 для вычисления произведения аргументов командной строки (рис. 14).

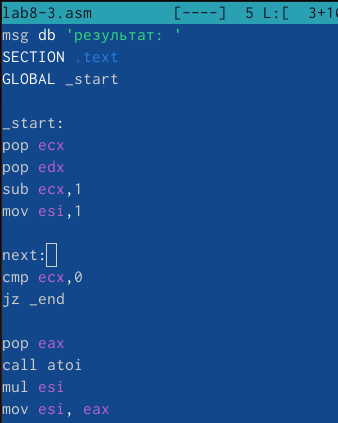


Рис. 14: Изменение программы

Код программы из пункта 14:

%include “in\_out.asm”

SECTION .data

msg db ‘результат:’

SECTION .text

GLOBAL \_start

\_start:

pop ecx

pop edx

sub ecx,1

mov esi,1

next:

cmp ecx,0

jz \_end

pop eax

call atoi

mul esi

mov esi, eax

loop next

\_end:

mov eax, msg

call sprint

mov eax, esi

call iprintLF

call quit

1. Создадим исполняемый файл и запустим его, указав аргументы (рис. 15).

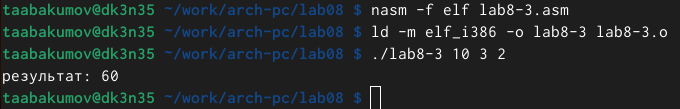


Рис. 15: Работа файла

## 3.3 Заданий для самостоятельной работы

1. Для начала создадим файл lab8-4.asm в каталоге ~/work/arch-pc/lab08 (рис. 16).

Рис. 16: Создание файла

Рис. 16: Создание файла

1. Напишем 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 xi передаются как аргументы. Вид функции f(x) выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Мой вариант - 9 (рис. 17).

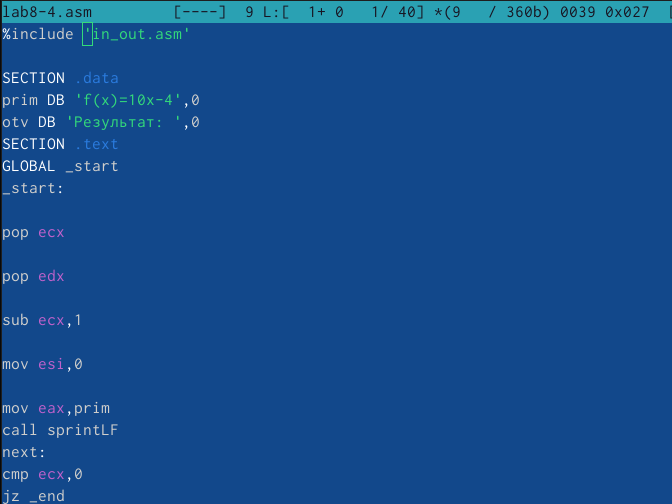


Рис. 17: Код программы

Код программы из пункта 17:

%include ‘in\_out.asm’

SECTION .data

prim DB ‘f(x)=10x-4’,0

otv DB ‘Результат:’,0

SECTION .text

GLOBAL \_start

\_start:

pop ecx

pop edx

sub ecx,1

mov esi,0

mov eax,prim

call sprintLF

next:

cmp ecx,0

jz \_end

mov ebx,10

pop eax

call atoi

mul ebx

add eax,-4

add esi,eax

loop next

\_end:

mov eax,otv

call sprint

mov eax,esi

call iprintLF

call quit

1. Создадим исполняемый файл и проверим его работу на нескольких наборах x = x1, x2, …, x (рис. 18).

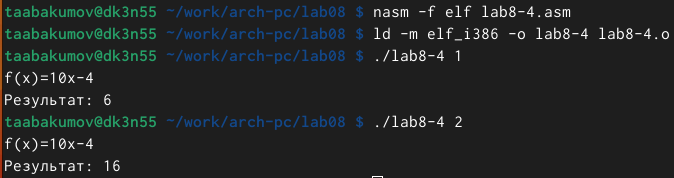


Рис. 18: Код программы

# 4 Выводы

Я приобрел навыки написания программы с использованием цикла.