

Лабораторная работа №8.

Программирование цикла. Обработка аргументов командной строки.

Жукова София Викторовна

Содержание

Цель работы	5
Задание	6
Выполнение лабораторной работы	7
Выводы	16

Список иллюстраций

1	Создаем каталог и файл	7
2	Пишем код	7
3	Проверяем работу	8
4	Изменяем программу	8
5	Создаем и запускаем файл	9
6	Редактируем файл	10
7	Проверяем	10
8	Создаем файл	11
9	Перепишем листинг	11
10	Создаем исполняемый файл	11
11	Создадим файл lab8-3.asm	12
12	Введем листинг 8.3	12
13	Создаём исполняемый файл	13
14	Изменяем текст	13
15	Проверяем работу файла	13
16	Создаем файл	14
17	Заполняем файл	14
18	Смотрим на работу программы	15
19	Смотрим на работу программы при $x_1=1$ $x_2=2$ $x_3=4$	15

Список таблиц

Цель работы

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

Задание

Написать программы с использованием циклов и обработкой аргументов командной строки.

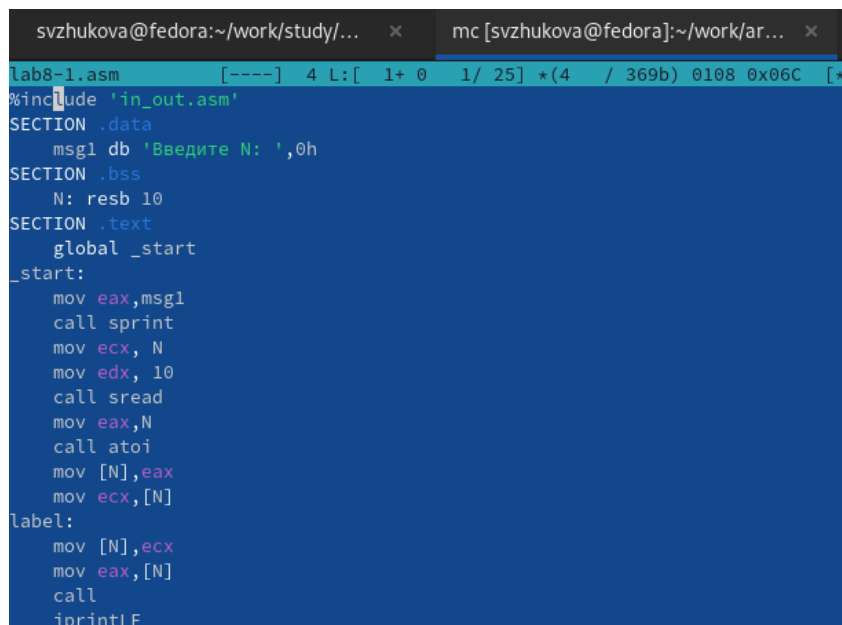
Выполнение лабораторной работы

1. Реализация циклов в NASM Создадим каталог для программ лабораторной работы № 8, перейдем в него и создадим файл lab8-1.asm: (рис. [-@fig:001]).

```
svzhukova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report$ mkdir ~/work/arch-pc/lab08
svzhukova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/report$ cd ~/work/arch-pc/lab08
svzhukova@fedora:~/work/arch-pc/lab08$ touch lab8-1.asm
svzhukova@fedora:~/work/arch-pc/lab08$
```

Рис. 1: Создаем каталог и файл

Откроем файл в Midnight Commander и введем в файл lab8-1.asm текст программы из листинга 8.1. (рис. [-@fig:002]).



```
lab8-1.asm [----] 4 L: [ 1+ 0 1/ 25] *(4 / 369b) 0108 0x06C [+
%include 'in_out.asm'
SECTION .data
    msg1 db 'Введите N: ',0h
SECTION .bss
    N: resb 10
SECTION .text
    global _start
_start:
    mov eax,msg1
    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
```

Рис. 2: Пишем код

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

```
svzhukova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
svzhukova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
svzhukova@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 15
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
svzhukova@fedora:~/work/arch-pc/lab08$
```

Рис. 3: Проверяем работу

Изменим текст программы добавив изменение значение регистра ecx в цикле:
(рис. [-@fig:004]).

```
label:
    sub ecx,1
    mov [N],ecx
    mov eax,[N]
    call iprintLF
```

Рис. 4: Изменяем программу

Создаем исполняемый файл и запускаем его (рис. [-@fig:005]).


```

svzhukova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
svzhukova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
svzhukova@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
Segmentation fault (core dumped)
svzhukova@fedora:~/work/arch-pc/lab08$

```

Рис. 5: Создаем и запускаем файл

Регистр `ecx` принимает значения 9,7,5,3,1 (на вход подается число 10, в цикле `label` данный регистр уменьшается на 2 командой `sub` и `loop` и выводятся только нечетные числа). Число проходов цикла не соответствует числу `N`, так как уменьшается на 2.

Снова открываем файл для редактирования и изменяем его, чтобы все корректно работало (рис. [-@fig:006]).

```

label:
    push ecx
    sub ecx,1
    mov [N],ecx
    mov eax,[N]
    call iprintLF
    pop ecx
    . . .
    loop label

```

Рис. 6: Редактируем файл

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

```

svzhukova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
svzhukova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
svzhukova@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
Segmentation fault (core dumped)
svzhukova@fedora:~/work/arch-pc/lab08$

```

Рис. 7: Проверяем

Число проходов цикла соответствует значению N введенному с клавиатуры.

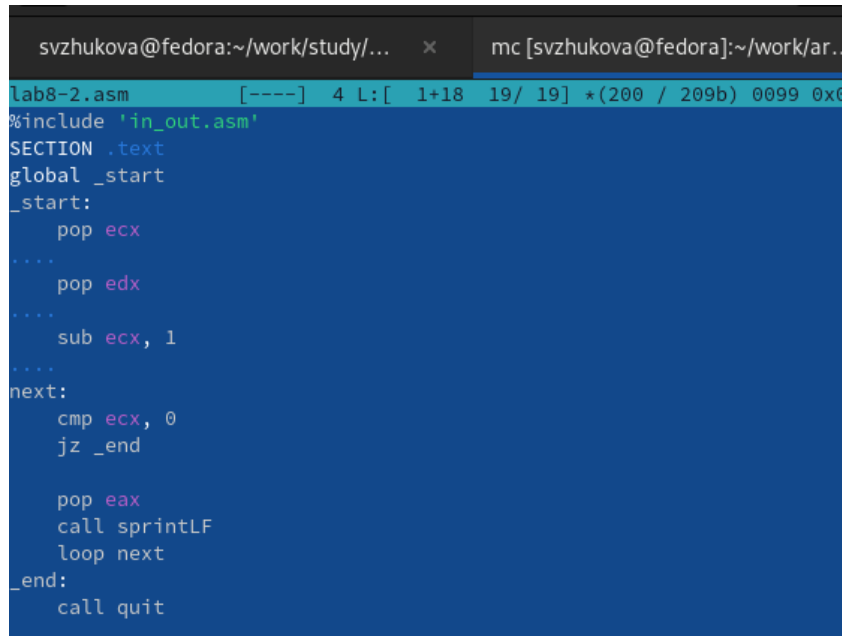
2. Обработка аргументов командной строки

Создаем файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 (рис. [-@fig:008]).

```
svzhukova@fedora:~/work/arch-pc/lab08$ touch lab8-2.asm
svzhukova@fedora:~/work/arch-pc/lab08$
```

Рис. 8: Создаем файл

Введем в него текст программы из листинга 8.2. (рис. [-@fig:009])



```
lab8-2.asm [----] 4 L: [ 1+18 19/ 19] *(200 / 209b) 0099 0xc
%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
```

Рис. 9: Перепишем листинг

Создадим исполняемый файл и запустим его, указав аргументы: (рис. [-@fig:010]).

```
svzhukova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
svzhukova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
svzhukova@fedora:~/work/arch-pc/lab08$ ./lab8-2 1 2 '3'
1
2
3
svzhukova@fedora:~/work/arch-pc/lab08$
```

Рис. 10: Создаем исполняемый файл

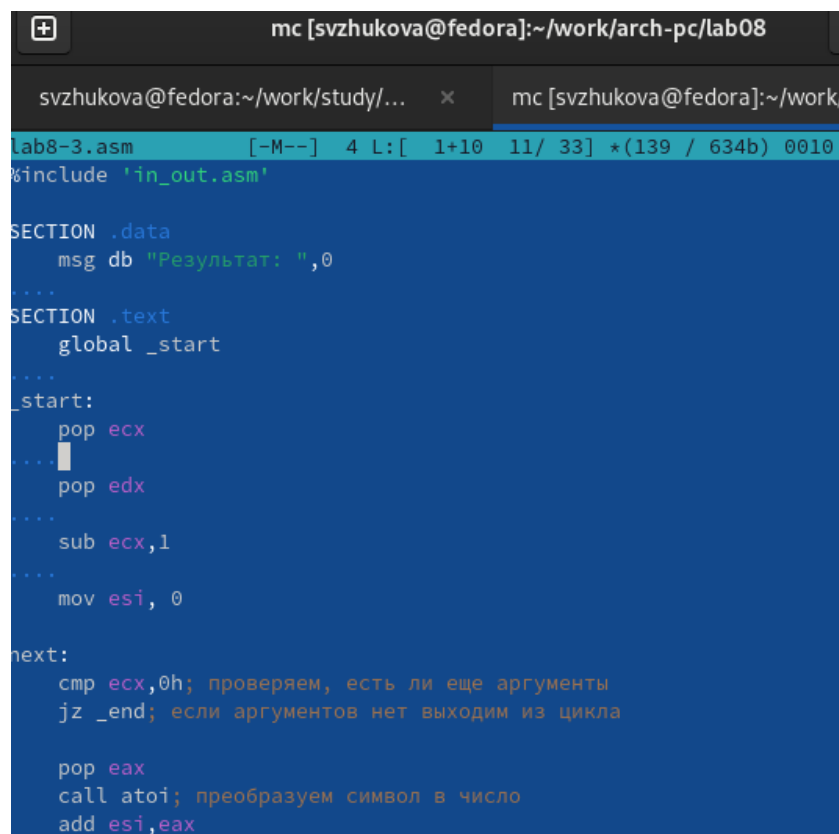
Программой было обработано 3 аргумента.

Создадим файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 (рис. [-@fig:011]).

```
svzhukova@fedora:~/work/arch-pc/lab08$ touch lab8-3.asm
svzhukova@fedora:~/work/arch-pc/lab08$
```

Рис. 11: Создадим файл lab8-3.asm

Введем в него текст программы из листинга 8.3. (рис. [-@fig:012]).



```
mc [svzhukova@fedora]:~/work/arch-pc/lab08
lab8-3.asm [-M--] 4 L: [ 1+10 11/ 33] *(139 / 634b) 0010
#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; проверяем, есть ли еще аргументы
    jz _end; если аргументов нет выходим из цикла

    pop eax
    call atoi; преобразуем символ в число
    add esi,eax
```

Рис. 12: Введем листин 8.3

Создаём исполняемый файл и запускаем его, указав аргументы (рис. [-@fig:013]).

```
svzhukova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
svzhukova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
svzhukova@fedora:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
svzhukova@fedora:~/work/arch-pc/lab08$
```

Рис. 13: Создаём исполняемый файл

Изменим текст программы из листинга 8.3 для вычисления произведения аргументов командной строки. (рис. [-@fig:014]).

```
    mov esi, 0
next:
    cmp ecx, 0h; проверяем, есть ли еще аргументы
    jz _end; если аргументов нет выходим из цикла

    pop eax
    call atoi; преобразуем символ в число
    mul esi
    mov esi, eax
    ....
    loop next
; переход к обработке следующего аргумента
_end:
    mov eax, msg
```

Рис. 14: Изменяем текст

Создаём исполняемый файл и запускаем его, указав аргументы (рис. [-@fig:015]).

```
svzhukova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
svzhukova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
svzhukova@fedora:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 0
svzhukova@fedora:~/work/arch-pc/lab08$
```

Рис. 15: Проверяем работу файла

4. Задание для самостоятельной работы

ВАРИАНТ 7

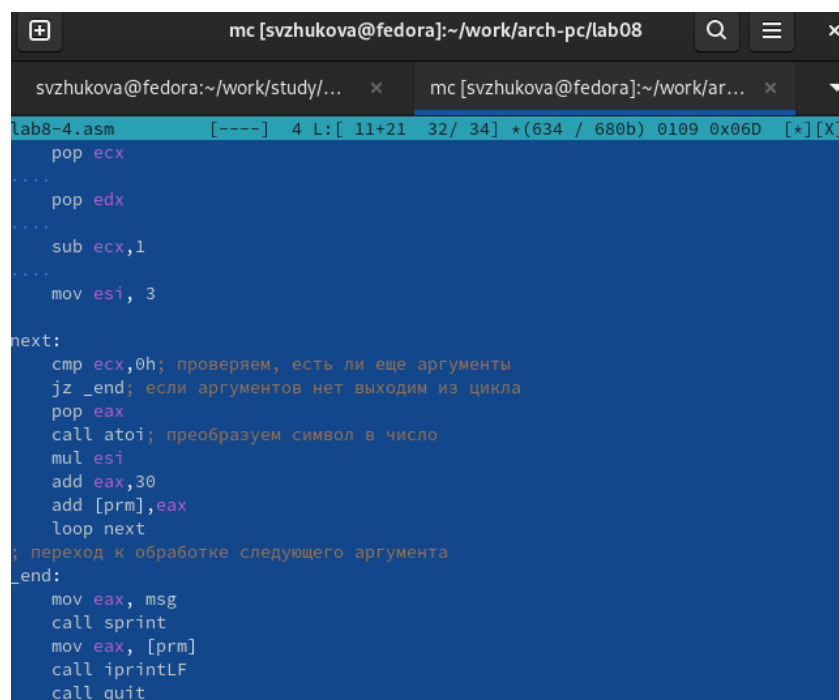
1. Напишите программу, которая находит сумму значений функции $F(X)$ для $x=x_1, x_2 \dots, x_n$, т.е. программа должна выводить значение $f(x_1) + f(x_1) + \dots + f(x_n)$. Создадим исполняемый файл и проверим его работу на нескольких наборах $x=x_1, x_2 \dots, x_n$

Создаем файл lab8-4.asm (рис. [-@fig:016]).

```
svzhukova@fedora:~/work/arch-pc/lab08$ touch lab8-4.asm
svzhukova@fedora:~/work/arch-pc/lab08$
```

Рис. 16: Создаем файл

Открываем файл и пишем программу, которая выведет сумму значений, полученных после решения выражения $3(x+2)$ (рис. [-@fig:017]).



```
lab8-4.asm  [----]  4  L: [ 11+21  32/ 34]  *(634 / 680b) 0109 0x06D  [*] [X]
    pop ecx
    ....
    pop edx
    ....
    sub ecx,1
    ....
    mov esi, 3
next:
    cmp ecx,0h; проверяем, есть ли еще аргументы
    jz _end; если аргументов нет выходим из цикла
    pop eax
    call atoi; преобразуем символ в число
    mul esi
    add eax,30
    add [prm],eax
    loop next
; переход к обработке следующего аргумента
_end:
    mov eax, msg
    call sprint
    mov eax, [prm]
    call iprintLF
    call quit
```

Рис. 17: Заполняем файл

Транслируем файл и смотрим на работу программы (рис. [-@fig:018]).

```
svzhukova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
svzhukova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
svzhukova@fedora:~/work/arch-pc/lab08$ ./lab8-4 1 2 3
Результат: 108
svzhukova@fedora:~/work/arch-pc/lab08$
```

Рис. 18: Смотрим на рабботу программы

Транслируем файл и смотрим на работу программы (рис. [-@fig:019]).

```
svzhukova@fedora:~/work/arch-pc/lab08$ ./lab8-4 1 2 4
Результат: 111
svzhukova@fedora:~/work/arch-pc/lab08$
```

Рис. 19: Смотрим на рабботу программы при $x_1=1$ $x_2=2$ $x_3=4$

Выводы

Мы приобрели навыки написания программ с использованием циклов и обработкой аргументов командной строки.