

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 8

дисциплина: Архитектура компьютера

Студент: Сухова Арина

Группа: НПИбд-02-25

МОСКВА

2025 г.

1. Цель работы

Изучить работу циклов и обработку аргументов командной строки.

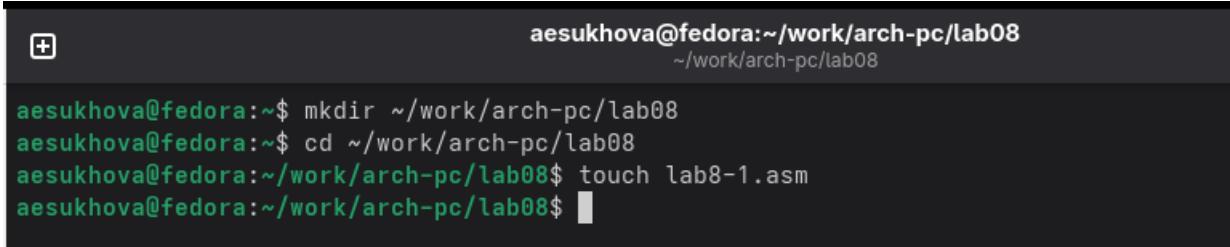
2. Задание

Написать программы с использованием циклов и обработкой аргументов командной строки.

3. Выполнение лабораторной работы

3.1 Реализация циклов в NASM

Создаем каталог для программ ЛБ8, и в нем создаем файл (рис. 1).



```
aesukhova@fedora:~/work/arch-pc/lab08
~/work/arch-pc/lab08

aesukhova@fedora:~$ mkdir ~/work/arch-pc/lab08
aesukhova@fedora:~$ cd ~/work/arch-pc/lab08
aesukhova@fedora:~/work/arch-pc/lab08$ touch lab8-1.asm
aesukhova@fedora:~/work/arch-pc/lab08$
```

Рис.1 Создаем каталог с помощью команды *mkdir* и файл с помощью команды *touch*

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 8.1 (рис. 2).

```
[+] mc [aesukhova@fedora]:~/work/arch-pc/lab08 — /usr/bin/mc -P /tmp/mc.pwd.tSgITY
~/work/arch-pc/lab08

lab8-1.asm      [-M--] 11 L:[ 1+22 23/ 23] *(348 / 348b) <
%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<><----->iprintfLF
loop label
call quit
```

Рис.2 Заполняем файл

Создаем исполняемый файл и запускаем его (рис. 3).

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

Рис.3 Запускаем файл и проверяем его работу

Снова открываем файл для редактирования и изменяем его, добавив изменение значения регистра в цикле (рис. 4).

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

Рис.4 Изменяем файл

Создаем исполняемый файл и запускаем его (рис. 5).

```
aesukhova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
aesukhova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
aesukhova@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
Ошибка сегментирования (образ памяти брошен на диск)
Чтобы исправить Windows, перейдите в раздел
Параметры.
```

Рис.5 Запускаем файл и смотрим на его работу

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).

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. 6).

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

Рис.6 Редактируем файл

Создаем исполняемый файл и запускаем его (рис. 7).

```
aesukhova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm  
aesukhova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o  
aesukhova@fedora:~/work/arch-pc/lab08$ ./lab8-1  
Введите N: 10  
9  
8  
7  
6  
5  
4  
3  
2  
1  
0  
Активация Windows  
Ошибка сегментирования (образ памяти брошен на диск)  
Чтобы активировать Windows, перейдите в раздел  
aesukhova@fedora:~/work/arch-pc/lab08$ "Параметры".
```

Рис.7 Проверяем, сошелся ли наш вывод с данным в условии выводом

В данном случае число проходов цикла равна числу N.

3.2 Обработка аргументов командной строки

Создаем новый файл (рис. 8).

```
aesukhova@fedora:~/work/arch-pc/lab08$ touch lab8-2.asm  
aesukhova@fedora:~/work/arch-pc/lab08$ Чтобы активировать Windows,  
"Параметры".
```

Рис.8 Создаем файл командой touch

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 8.2 (рис. 9).

```
[+] mc [aesukhova@fedora]:~/work/arch-pc/lab08 — /usr/bin/mc -P /tmp/mc.pwd.234MPm
~/work/arch-pc/lab08

lab8-2.asm      [-M--] 13 L:[ 1+14 15/ 15] *(191 / 191b) <
%include 'in_out.asm'
SECTION .text
global _start
_start:
    pop ecx
    pop edx
    sub ecx,1
next:
    cmp ecx,0
    jz _enx
    pop eax
    call sprintLF
    loop next
_end:
    call quit
```

Рис.9 Заполняем файл

Создаем исполняемый файл и проверяем его работу, указав аргументы (рис. 10).

```
aesukhova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
aesukhova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
aesukhova@fedora:~/work/arch-pc/lab08$ ./lab8-2
aesukhova@fedora:~/work/arch-pc/lab08$ ./lab8-2 1 2 '3'
1
2                               Активация Windows
3                               Чтобы активировать Windows, перейдите в раздел
aesukhova@fedora:~/work/arch-pc/lab08$ Параметры".
```

Рис.10 Смотрим на работу программ

Программой было обработано 3 аргумента.

Создаем новый файл lab8-3.asm (рис. 11).

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

Рис.11 Создаем файл командой touch

Открываем файл и заполняем его в соответствии с листингом 8.3 (рис. 12).

```
[+] mc [aesukhova@fedora]:~/work/arch-pc/lab08 — /usr/bin/mc -P /tmp/mc.pwd.KK7s0g
~/work/arch-pc/lab08

lab8-3.asm      [-M--] 13 L:[ 1+22 23/ 23] *(331 / 331b)
%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
```

Рис.12 Заполняем файл

Создаём исполняемый файл и запускаем его, указав аргументы (рис. 13).

```
aesukhova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
aesukhova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
aesukhova@fedora:~/work/arch-pc/lab08$ lab8-3
Результат: 47                                         Чтобы активировать Windows, перейдите в раздел
                                                        "Параметры".
aesukhova@fedora:~/work/arch-pc/lab08$
```

Рис.13 Смотрим на работу программы

Снова открываем файл для редактирования и изменяем его, чтобы вычислялось произведение вводимых значений (рис. 14).

```
mc [aesukhova@fedora]:~/work/arch-pc/lab08 — /usr/bin/mc -P /tmp/mc.pwd.zp3ZLZ ~/work/arch-pc/lab08
lab8-3.asm      [-M--] 5 L:[ 1+10 11/ 25] *(161 / 347b) 0010[*][X]
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
    pop ecx
    pop edx
    sub ecx,1
    mov esi,1
next:
    cmp ecx,0h
    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
```

Рис.14 Изменяем файл

Создаём исполняемый файл и запускаем его, указав аргументы (рис. 14).

```
aesukhova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
aesukhova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
aesukhova@fedora:~/work/arch-pc/lab08$ ./lab8-3 5 6 7 8 9
Результат: 60
Чтобы активировать Windows, перейдите в раздел параметры.
aesukhova@fedora:~/work/arch-pc/lab08$
```

Рис.14 Проверяем работу файла(работает правильно)

3.3 Задание для самостоятельной работы

ВАРИАНТ-5

- Напишите программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, т.е. программа должна выводить значение $f(x_1) + f(x_2) + \dots + f(x_n)$.
Значения x_i передаются как аргументы. Вид функции $f(x)$ выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении

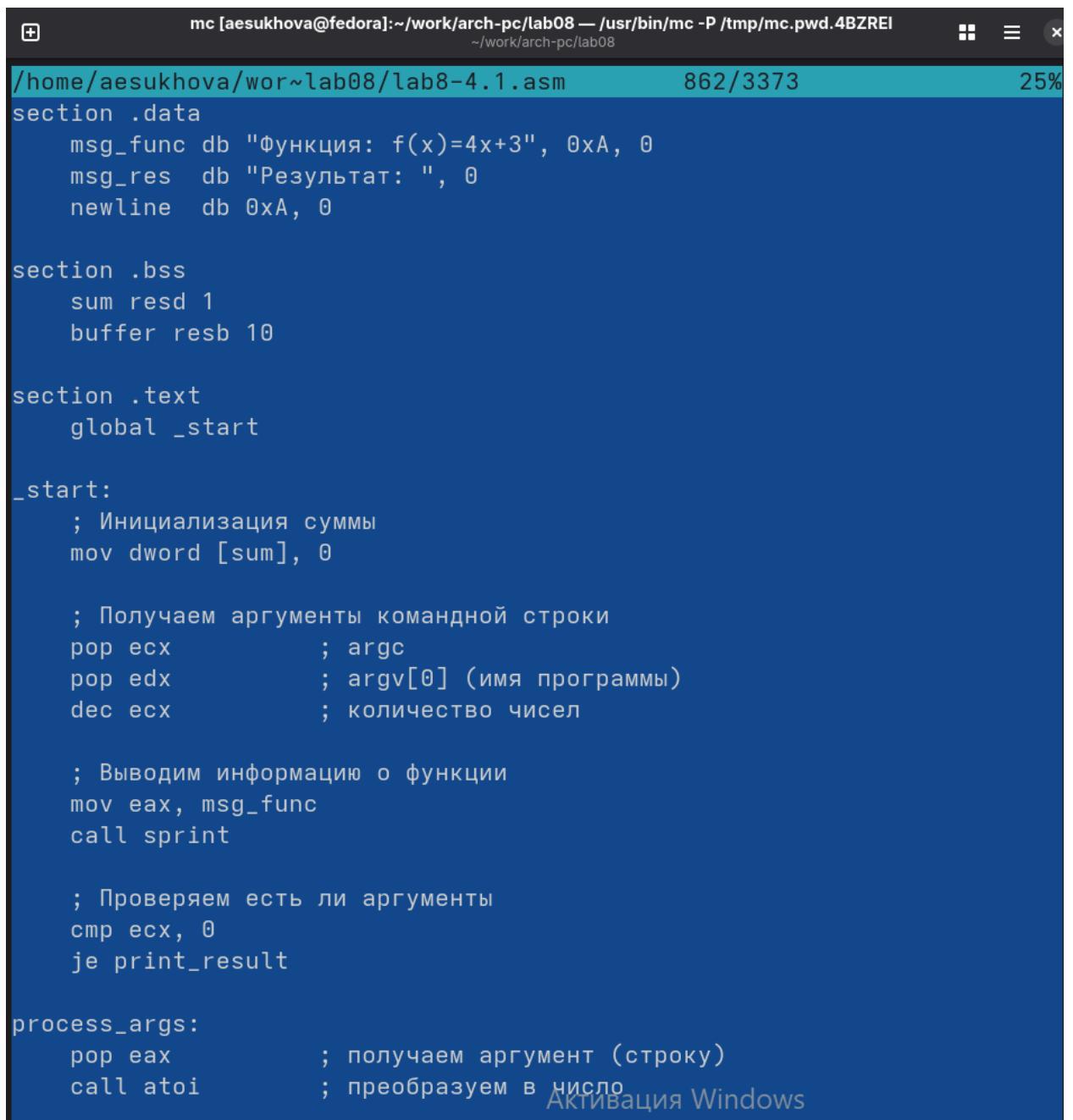
лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах $x = x_1, x_2, \dots, x_n$.

Создаем новый файл (рис. 15).

```
aesukhova@fedora:~/work/arch-pc/lab08$ touch lab8-4.1.asm  
aesukhova@fedora:~/work/arch-pc/lab08$ mc
```

Рис.15 Создаем файл командой touch

Открываем его и пишем программу, которая выведет сумму значений, получившихся после решения выражения $3(10+x)$ (рис. 16).



```
mc [aesukhova@fedora]:~/work/arch-pc/lab08 — /usr/bin/mc -P /tmp/mc.pwd.4BZREI  
~/work/arch-pc/lab08  
  
/home/aesukhova/wor~lab08/lab8-4.1.asm      862/3373    25%  
section .data  
    msg_func db "Функция: f(x)=4x+3", 0xA, 0  
    msg_res  db "Результат: ", 0  
    newline   db 0xA, 0  
  
section .bss  
    sum resd 1  
    buffer resb 10  
  
section .text  
    global _start  
  
_start:  
    ; Инициализация суммы  
    mov dword [sum], 0  
  
    ; Получаем аргументы командной строки  
    pop ecx          ; argc  
    pop edx          ; argv[0] (имя программы)  
    dec ecx          ; количество чисел  
  
    ; Выводим информацию о функции  
    mov eax, msg_func  
    call sprint  
  
    ; Проверяем есть ли аргументы  
    cmp ecx, 0  
    je print_result  
  
process_args:  
    pop eax          ; получаем аргумент (строку)  
    call atoi         ; преобразуем в число
```

Рис.16 Пишем программу

Транслируем файл и смотрим на работу программы (рис. 17).

```
aesukhova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.1.asm
aesukhova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4.1 lab8-4
.1.o
aesukhova@fedora:~/work/arch-pc/lab08$ ./lab8-4.1 0 5 10
Функция: f(x)=4x+3
Результат: 69
aesukhova@fedora:~/work/arch-pc/lab08$
```

Активация Windows

Чтобы активировать Windows, перейдите в раздел
"Параметры".

Рис.17 Смотрим на работу программы при $x1=0$ $x2=5$ $x3=10$ (всё верно)

Транслируем файл и смотрим на работу программы (рис. 18).

```
aesukhova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.1.asm
aesukhova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4.1 lab8-4
.1.o
aesukhova@fedora:~/work/arch-pc/lab08$ ./lab8-4.1 2 4 6 8
Функция: f(x)=4x+3
Результат: 92
aesukhova@fedora:~/work/arch-pc/lab08$
```

Активация Windows

Чтобы активировать Windows, перейдите в раздел

Рис.18 Смотрим на работу программы при $x1=2$ $x2=4$ $x3=6$ $x4=8$ (всё верно)

4. Выводы

Мы научились решать программы с использованием циклов и обработкой аргументов командной строки.