

Отчет по лабораторной работе №8

**Программирование циклов. Обработка аргументов
командной строки.**

Пашутина Анна Алексеевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Реализация циклов в NASM	7
3.2	Обработка аргументов командной строки.	9
3.3	Задание для самостоятельной работы	12
4	Выводы	14

Список иллюстраций

3.1	Создаем каталог и файл lab8-1.asm	7
3.2	Заполняем файл в соответствии с заданием	7
3.3	Запускаем файл и проверяем его работу	8
3.4	Изменяем файл в соответствии с заданием	8
3.5	Запускаем файл и смотрим на его работу	8
3.6	Редактируем файл	9
3.7	Проверяем, сошелся ли наш вывод с данным в условии выводом . .	9
3.8	Создаем файл lab8-2.asm	9
3.9	Заполняем файл в соответствии с заданием	10
3.10	Смотрим на работу программ	10
3.11	Создаем файл lab8-3.asm	10
3.12	Заполняем файл в соответствии с заданием	11
3.13	Смотрим на работу программы	11
3.14	Изменяем файл	11
3.15	Проверяем работу файла(работает правильно)	12
3.16	Создаем файл lab8-4.asm	12
3.17	Пишем программу, которая выполнит задание	13
3.18	Смотрим на работу программы при x1=2 x2=3 x1=5(всё верно)	13
3.19	Смотрим на работу программы при x1=1 x2=6 x1=0(всё верно)	13

Список таблиц

1 Цель работы

Изучить работу циклов и обработкой аргументов командной строки.

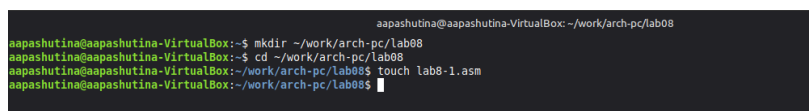
2 Задание

Написать программы с использованием циклов и обработкой аргументов командной строки.

3 Выполнение лабораторной работы

3.1 Реализация циклов в NASM

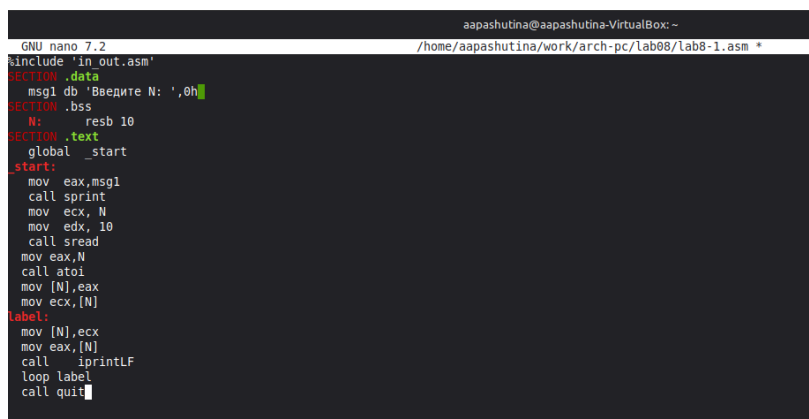
Создаем каталог для программ лабораторной работы №8 с помощью команды `mkdir`, и в нем создаем файл `lab8-1.asm` с помощью команды `touch`(рис.1).



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

Рисунок 3.1: Создаем каталог и файл `lab8-1.asm`

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



```
GNU nano 7.2 /home/aapashutina/work/arch-pc/lab08/lab8-1.asm *
$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    iprintf
    loop    label
    call    quit
```

Рисунок 3.2: Заполняем файл в соответствии с заданием

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

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

Рисунок 3.3: Запускаем файл и проверяем его работу

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

```
GNU nano 7.2 aapashutina@aapashutina-VirtualBox: -
~/home/aapashutina/work/arch-pc/lab08/lab8-1.asm *
#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:
    sub     ecx,1
    mov     [N],ecx
    mov     eax,[N]
    call    iprintLF
    loop    label
```

Рисунок 3.4: Изменяем файл в соответствии с заданием

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

```
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 8
7
5
3
1
Ошибка сегментирования (образ памяти сброшен на диск)
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08$
```

Рисунок 3.5: 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.

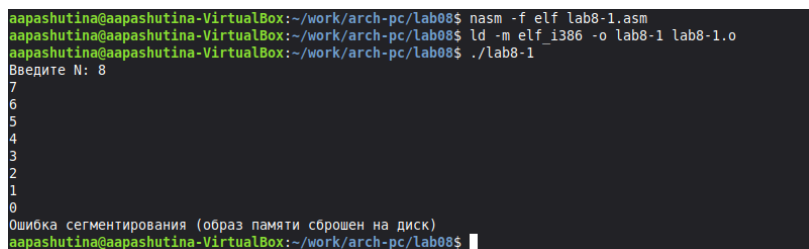
Снова открываем файл для редактирования и изменяем его, чтобы все корректно работало (рис.6).



```
GNU nano 7.2 /home/aapashutina/work/arch-pc/lab08/lab8-1.asm *
#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:
push ecx
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintlnf
pop ecx
loop label
```

Рисунок 3.6: Редактируем файл

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



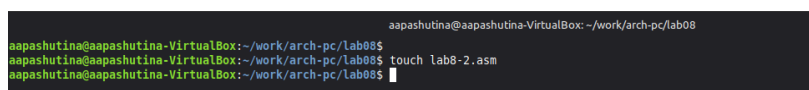
```
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 8
7
6
5
4
3
2
1
0
Ошибка сегментирования (образ памяти сброшен на диск)
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08$
```

Рисунок 3.7: 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.

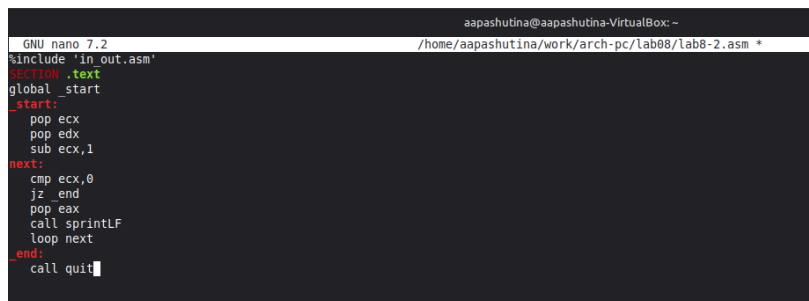
Создаем новый файл lab8-2.asm с помощью команды touch(рис.8).



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

Рисунок 3.8: Создаем файл lab8-2.asm

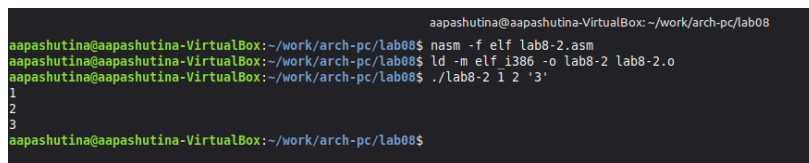
Открываем файл в Midnight Commander с помощью команды `mc` и заполняем его в соответствии с листингом 8.2 (рис.9).



```
aapashutina@aapashutina-VirtualBox: ~
GNU nano 7.2 /home/aapashutina/work/arch-pc/lab08/lab8-2.asm *
#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 sprintf
    loop next
end:
    call quit
```

Рисунок 3.9: Заполняем файл в соответствии с заданием

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



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

Рисунок 3.10: Смотрим на работу программ

Программой было обработано 3 аргумента.

Создаем новый файл `lab8-3.asm` с помощью команды `touch`(рис.11).



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

Рисунок 3.11: Создаем файл `lab8-3.asm`

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

```
aapashutina@aapashutina-VirtualBox: ~
GNU nano 7.2 /home/aapashutina/work/arch-pc/lab08/lab8-3.asm
#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,ecx
    loop next
_end:
    mov eax,msg
    call sprintf
    mov eax,esi
    call iprintLF
    call quit
```

Рисунок 3.12: Заполняем файл в соответствии с заданием

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

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

Рисунок 3.13: Смотрим на работу программы

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

```
mc [aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08
GNU nano 7.2 /home/aapashutina/work/arch-pc/lab08/lab8-3.asm
#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 sprintf
    mov eax,esi
    call iprintLF
    call quit
```

Рисунок 3.14: Изменяем файл

Создаём исполняемый файл и запускаем его, указав аргументы (рис.15).

```

aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 2 4 5
Результат: 40
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08$

```

Рисунок 3.15: Проверяем работу файла(работает правильно)

3.3 Задание для самостоятельной работы

В лабораторной работе №6 программа высчитала мой номер варианта, который равен 3.

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 $f(x_1) + f(x_2) + \dots + f(x_n)$. 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 № 6. Создайте исполняемый файл и проверьте его работу на нескольких наборах $x = x_1, x_2, \dots, x_n$.

Создаем новый файл lab8-4.asm с помощью команды touch(рис.16).

```

aapashutina@aapashutina-VirtualBox: ~/work/arch-pc/lab08
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08$ touch lab8-4.asm
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08$

```

Рисунок 3.16: Создаем файл lab8-4.asm

Открываем его и пишем программу, которая выведет сумму значений, получившихся после решения выражения $(10^x - 5)$ (рис.17).

```

mc [aapashutina@aapashutina-VirtualBox] ~/work/arch-pc/lab08
GNU nano 2.2
Action: test
global _start

_start:
; Получаем количество аргументов
pop ecx ; argc
dec ecx ; пропускаем имя программы
jz exit ; если нет аргументов - выходим
mov ebx, 0 ; обнуляем сумму (здесь будет итоговая сумма)

process_args:
; Получаем следующий аргумент
pop esi ; адрес строки аргумента

; Преобразуем строку в число
xor eax, eax ; обнуляем eax (здесь будет текущее число)
xor ebx, ebx ; обнуляем ebx

convert_loop:
mov dl, byte [esi] ; берем очередной символ
test dl, dl ; проверяем на конец строки
jz calculated
cmp dl, '0'
jb next_arg
cmp dl, '9'
ja next_arg

sub dl, '0' ; преобразуем символ в цифру
imul eax, 10 ; умножаем текущее значение на 10
add ebx, eax ; добавляем новую цифру
inc esi ; следующий символ
jmp convert_loop

calculated:
; Вычисляем 18x - 5
imul ebx, 18 ; умножаем на 18 (18x)
sub ebx, 5 ; вычитаем 5 (18x - 5)
add ebx, eax ; добавляем в общую сумму

next_arg:
; Следующий аргумент
dec esi
jnz process_args

; Выводим результат
mov eax, ebx ; результат в eax
call print_number

exit:
; Завершение программы
mov eax, 1 ; sys_exit
xor ebx, ebx ; код возврата 0
int 0x80

; Функция для вывода числа
print_number:
mov ecx, 10 ; делитель
mov edi, buffer + 10 ; конец буфера
mov byte [edi], 0 ; нулевой терминатор
dec edi

mov esi, eax ; сохраняем число
test esi, esi ; проверяем знак

```

Рисунок 3.17: Пишем программу, которая выполнит задание

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

```

aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08$ ./lab8-4 2 3 5
Результат: 40
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08$

```

Рисунок 3.18: Смотрим на работу программы при x1=2 x2=3 x1=5(всё верно)

Транслируем файл и смотрим на работу программы (рис.19).

```

aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08$ ./lab8-4 1 6 0
Результат: 60
aapashutina@aapashutina-VirtualBox:~/work/arch-pc/lab08$

```

Рисунок 3.19: Смотрим на работу программы при x1=1 x2=6 x1=0(всё верно)

4 Выводы

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