

# Отчёт по лабораторной работе №8

## Простейший вариант

Волчкова Елизавета Дмитриевна

## Содержание

## Цель работы

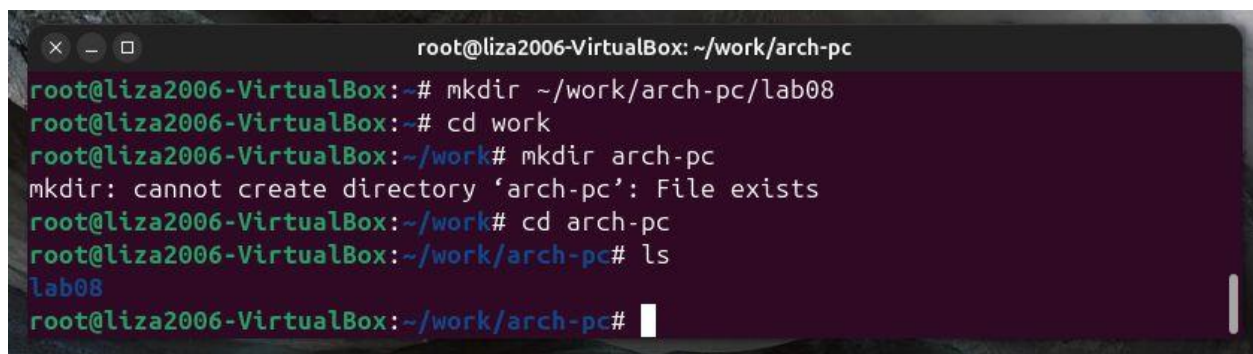
Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

## Задание

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  $x = x_1, x_2, \dots, x_n$ .

## Выполнение лабораторной работы

Создала каталог lab08



```
root@liza2006-VirtualBox: ~/work/arch-pc
root@liza2006-VirtualBox:~# mkdir ~/work/arch-pc/lab08
root@liza2006-VirtualBox:~# cd work
root@liza2006-VirtualBox:~/work# mkdir arch-pc
mkdir: cannot create directory 'arch-pc': File exists
root@liza2006-VirtualBox:~/work# cd arch-pc
root@liza2006-VirtualBox:~/work/arch-pc# ls
lab08
root@liza2006-VirtualBox:~/work/arch-pc#
```

Ввела в файл lab8-1.asm текст программы из листинга 8.1.

```
выводит значение регистра есх. Внимательно изучите текст программы (листинг 8.1).
root@liza2006-VirtualBox: ~/work/arch-pc/lab08
lab08
root@liza2006-VirtualBox:~/work/arch-pc# touch lab8-1.asm
root@liza2006-VirtualBox:~/work/arch-pc# cd lab08
root@liza2006-VirtualBox:~/work/arch-pc/lab08# ls
root@liza2006-VirtualBox:~/work/arch-pc/lab08# touch lab8-1.asm
root@liza2006-VirtualBox:~/work/arch-pc/lab08# ls
lab8-1.asm
root@liza2006-VirtualBox:~/work/arch-pc/lab08#
```

Затем создала исполняемый файл и проверила его работу.

```
//root/work/arch-pc/lab08/lab8-1.asm 637/637 100%
#include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ---- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ---- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ---- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, 'ecx=N'

label:
mov [N],ecx
mov eax,[N]
call iprintlnLF ; Вывод значения 'N'
loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
; переход на 'label'
call quit

1Help 2UnWrap 3Quit 4Hex 5Goto 6 7Search 8Raw 9Format 10Quit
```

Данный пример показывает, что использование регистра есх в теле цикла loop может привести к некорректной работе программы. Далее я изменила текст программы добавив изменение значение регистра есх в цикле.

Потом создала исполняемый файл и проверила его работу. В качестве примера рассмотрела программу, которая выводит значение регистра есх. Внимательно изучила текст программы. Создала исполняемый файл и проверила его работу. Регистр есх принимает значение, равное значению *N*, введенному с клавиатуры.

```

/home/liza2006/work/study/20~/arch-pc/labs/lab08/lab8.asm [-M--] 11 L:[ 1+13 14/ 2
%include "in_out.asm"
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, 'ecx=N'
label:
sub ecx,1 ; 'ecx=ecx-1'
mov [N],ecx
mov eax,[N]
call iprintfLF
loop label
call quit
1Help 2Save 3Mark 4Replac 5Copy 6love 7Search 8

```

После внесла изменения в текст программы, добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop. Создала исполняемый файл и проверила его работу. Соответствует ли в данном случае число проходов цикла значению  $N$  введенному с клавиатуры? - нет! Циклов было больше 100.

```

mc [root@liza2006-VirtualBox]:~/work/arch-pc/lab08
4294824886
4294824884
4294824882
4294824880
4294824878
4294824876
4294824874
4294824872
4294824870
4294824868
4294824866
4294824864
4294824862
4294824860
4294824858
4294824856
4294824854
4294824852
4294824850
4294824848
4294824846
4294824844
4294824842
4

```

```

/home/liza2006/work/study/20~/arch-pc/labs/lab08/lab8.asm [-M--] 11 L:[ 1+13 14/ 2
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, 'ecx=N'
label:
sub ecx,1 ; 'ecx=ecx-1'
mov [N],ecx
mov eax,[N]
call iprintLF
loop label
call quit

```

Для использования регистра ecx в цикле и сохранения корректности работы программы можно использовать стек. Внесите изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop:

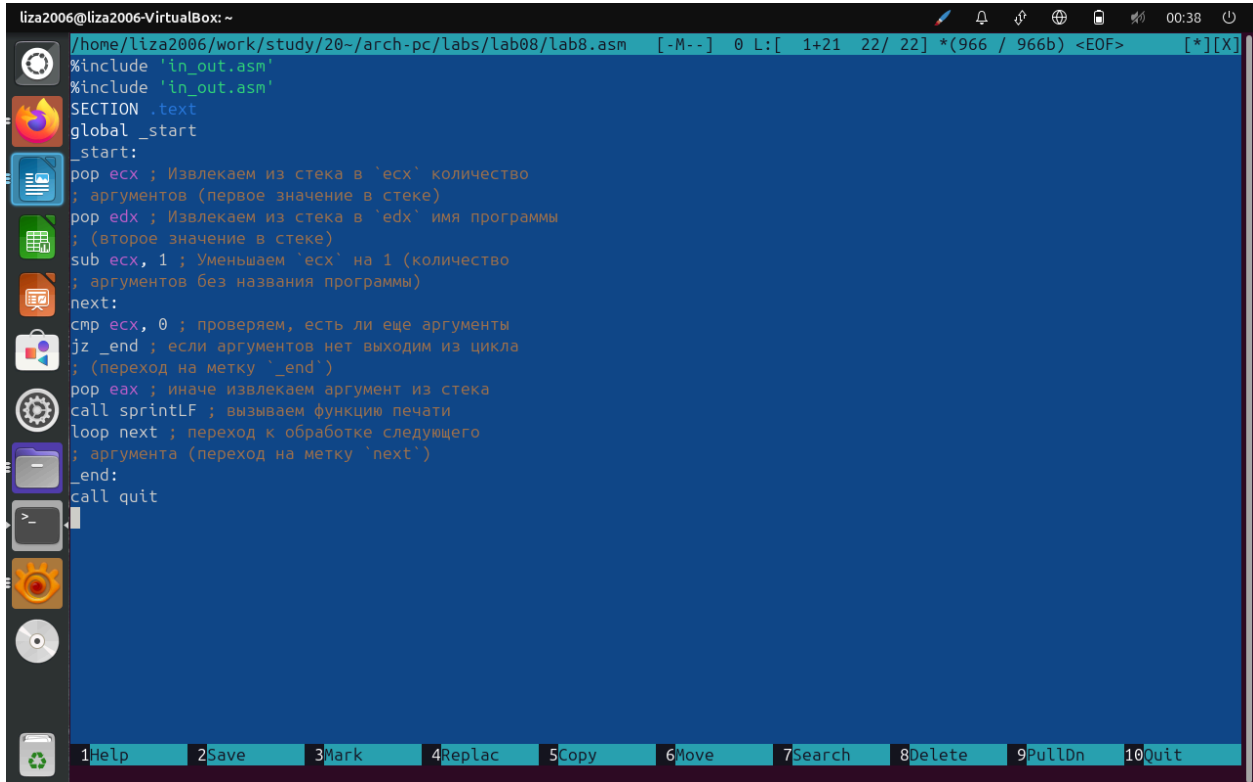
```

/home/liza2006/work/study/20~/arch-pc/labs/lab08/lab8.asm [-M
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, 'ecx=N'
label:
push ecx ; добавление значения ecx в стек
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx ; извлечение значения ecx из стека
loop label

call quit

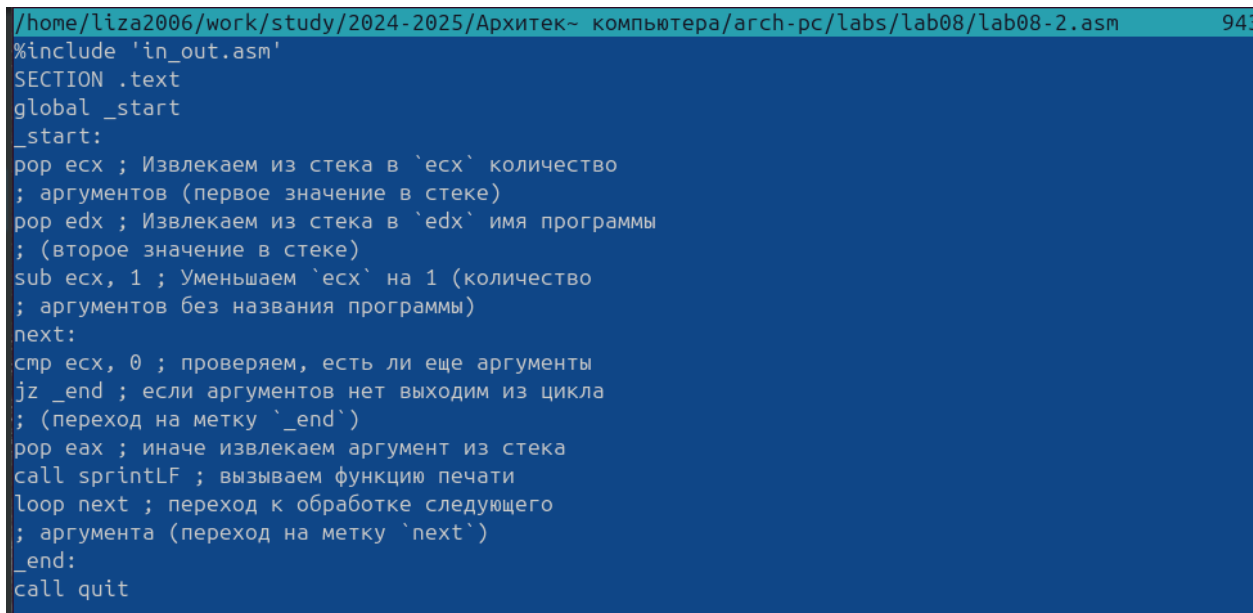
```

Для того чтобы использовать аргументы в программе, их просто нужно извлечь из стека. Обработку аргументов нужно проводить в цикле. В качестве примера рассмотрела программу, которая выводит на экран аргументы командной строки. Изучила текст программы:



```
liza2006@liza2006-VirtualBox: ~  
/home/liza2006/work/study/20~/arch-pc/labs/lab08/lab8.asm [-M--] 0 L: [ 1+21 22/ 22] *(966 / 966b) <EOF> [*][X]  
%include 'in_out.asm'  
%include 'in_out.asm'  
SECTION .text  
global _start  
_start:  
    pop ecx ; Извлекаем из стека в 'ecx' количество  
            ; аргументов (первое значение в стеке)  
    pop edx ; Извлекаем из стека в 'edx' имя программы  
            ; (второе значение в стеке)  
    sub ecx, 1 ; Уменьшаем 'ecx' на 1 (количество  
            ; аргументов без названия программы)  
next:  
    cmp ecx, 0 ; проверяем, есть ли еще аргументы  
    jz _end ; если аргументов нет выходим из цикла  
            ; (переход на метку '_end')  
    pop eax ; иначе извлекаем аргумент из стека  
    call sprintf ; вызываем функцию печати  
    loop next ; переход к обработке следующего  
            ; аргумента (переход на метку 'next')  
_end:  
    call quit
```

Создала файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввела в него текст программы из листинга 8.2.



```
/home/liza2006/work/study/2024-2025/Архитек~ компьютера/arch-pc/labs/lab08/lab08-2.asm 94:  
%include 'in_out.asm'  
SECTION .text  
global _start  
_start:  
    pop ecx ; Извлекаем из стека в 'ecx' количество  
            ; аргументов (первое значение в стеке)  
    pop edx ; Извлекаем из стека в 'edx' имя программы  
            ; (второе значение в стеке)  
    sub ecx, 1 ; Уменьшаем 'ecx' на 1 (количество  
            ; аргументов без названия программы)  
next:  
    cmp ecx, 0 ; проверяем, есть ли еще аргументы  
    jz _end ; если аргументов нет выходим из цикла  
            ; (переход на метку '_end')  
    pop eax ; иначе извлекаем аргумент из стека  
    call sprintf ; вызываем функцию печати  
    loop next ; переход к обработке следующего  
            ; аргумента (переход на метку 'next')  
_end:  
    call quit
```

Создала исполняемый файл и запустила его, указав аргументы:

```
liza2006@liza2006-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ nasm -f elf32 lab08-2.asm  
-o obj.o  
liza2006@liza2006-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ld -m elf_i386 -o lab8 ob  
j.o  
liza2006@liza2006-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ./lab8  
liza2006@liza2006-VirtualBox:~$ ./lab8 10 20 '30'  
bash: ./lab8: No such file or directory  
liza2006@liza2006-VirtualBox:~$ mc  
liza2006@liza2006-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ./lab8 10 20 '30'  
10  
20  
30
```

Сколько аргументов было обработано программой? -Было обработано 3!

Рассмотрела еще один пример программы, которая выводит сумму чисел, которые передаются в программу как аргументы. Создала файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 и ввела в него текст программы из листинга 8.3

```
/home/liza2006/work/study/20~ch-pc/labs/lab08/lab08-3.asm [-M--] 32 L:  
%include 'in_out.asm'  
SECTION .data  
msg db "Результат: ",0  
SECTION .text  
global _start  
_start:  
    pop ecx ; Извлекаем из стека в `ecx` количество  
            ; аргументов (первое значение в стеке)  
    pop edx ; Извлекаем из стека в `edx` имя программы  
            ; (второе значение в стеке)  
    sub ecx,1 ; Уменьшаем `ecx` на 1 (количество  
            ; аргументов без названия программы)  
    mov esi, 0 ; Используем `esi` для хранения  
            ; промежуточных сумм  
next:  
    cmp ecx,0h ; проверяем, есть ли еще аргументы  
    jz _end ; если аргументов нет выходим из цикла  
            ; (переход на метку `_end`)  
    pop eax ; иначе извлекаем следующий аргумент из стека  
    call atoi ; преобразуем символ в число  
    add esi,eax ; добавляем к промежуточной сумме  
            ; след. аргумент `esi=esi+eax`  
    loop next ; переход к обработке следующего аргумента  
_end:  
    mov eax, msg ; вывод сообщения "Результат: "  
    call sprint  
    mov eax, esi ; записываем сумму в регистр `eax`  
    call iprintLF ; печать результата  
    call quit ; завершение программы
```

Создала исполняемый файл и запустила его, указав аргументы. Пример результата работы программы:

```
liza2006@liza2006-VirtualBox:~$ mc
liza2006@liza2006-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ touch lab08-3.asm
liza2006@liza2006-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ nasm -f elf32 lab08-3.asm -o obj.o
liza2006@liza2006-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ld -m elf_i386 -o lab8-3 obj.o
liza2006@liza2006-VirtualBox:~$ mc
liza2006@liza2006-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
```

Затем изменила текст программы из листинга 8.3 для вычисления произведения аргументов командной строки

Написала 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  $f(x)$  выбрала из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7 и создав исполняемый файл проверила его работу на нескольких наборах  $x = x_1, x_2, \dots, x_n$

```
/root/work/arch-pc/lab08/lab8-dz.asm [----] 0 L: [ 63+26 89/ 94] *(3242/3410b) 0010 0x00A [*][X]
.convert_to_string:
    dec edi                ; Move backwards in the buffer
    xor edx, edx           ; Clear edx (used for division)
    mov ebx, 10            ; Divisor
    div ebx                ; Divide eax by 10
    add dl, '0'            ; Convert remainder to ASCII
    mov [edi], dl          ; Store character in the buffer
    test eax, eax          ; Check if quotient is zero
    jnz .convert_to_string ; If not zero, continue converting

    ; Calculate the length of the string
    mov edx, sum_res       ; Start of the result string
    sub edx, edi           ; Length of the string

    ; Print the result
    mov ecx, edi           ; Pointer to the start of the result string
    mov ebx, 1             ; File descriptor (stdout)
    mov eax, 4             ; Syscall number for sys_write
    int 0x80              ; Call kernel

    ; Print a new line
    mov edx, 1             ; Length of the new line
    mov ecx, new_line      ; Pointer to new line
    mov eax, 4             ; Syscall number for sys_write
    int 0x80              ; Call kernel

    ; Exit the program
    mov eax, 1             ; Syscall number for exit
    xor ebx, ebx           ; Return 0
    int 0x80              ; Call kernel

1 Help    2 Save    3 Mark    4 Replac    5 Copy    6 Move    7 Search    8 Delete    9 PullDn    10 Quit
```

## Выводы

Целью работы было приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки. Прodelав данные задания я усвоила материал и теперь имею представления о том, как правильно использовать циклы и командные 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, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ- Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: [http://www.stolyarov.info/books/asm\\_unix](http://www.stolyarov.info/books/asm_unix).
15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).