

Отчет по лабораторной работе №8

Костеренко Полина

Содержание

Список иллюстраций

Список таблиц

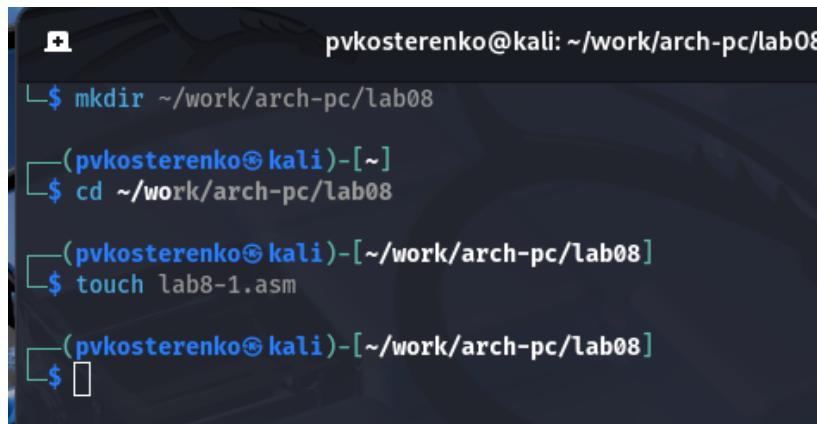
1 Цель работы

Изучить работу циклов и обработкой аргументов командной строки.

2 Выполнение лабораторной работы

2.1 Реализация циклов в NASM

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

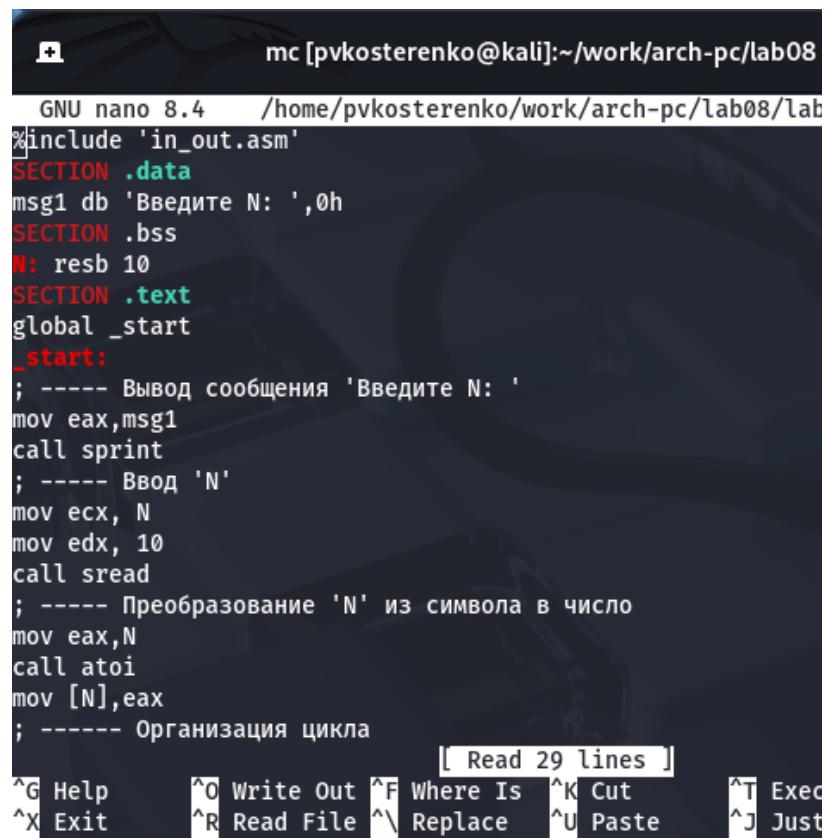


```
pvkosterenko@kali: ~/work/arch-pc/lab08
└ $ mkdir ~/work/arch-pc/lab08
└ (pvkosterenko@kali)-[~]
└ $ cd ~/work/arch-pc/lab08
└ (pvkosterenko@kali)-[~/work/arch-pc/lab08]
└ $ touch lab8-1.asm
└ (pvkosterenko@kali)-[~/work/arch-pc/lab08]
└ $ 
```

A screenshot of a terminal window titled "pvkosterenko@kali: ~/work/arch-pc/lab08". The terminal shows the following command history:
\$ mkdir ~/work/arch-pc/lab08
\$ cd ~/work/arch-pc/lab08
\$ touch lab8-1.asm
\$
The prompt at the end indicates the user has finished entering commands.

Рисунок 2.1: 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. Рисунок ??).



The screenshot shows a terminal window titled 'mc [pvkosterenko@kali]:~/work/arch-pc/lab08'. The file being edited is 'in_out.asm'. The code is as follows:

```
GNU nano 8.4      /home/pvkosterenko/work/arch-pc/lab08/lab08.asm
%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
[ Read 29 lines ]
^G Help      ^O Write Out ^F Where Is ^K Cut      ^T Exec
^X Exit      ^R Read File ^\ Replace   ^U Paste    ^J Just
```

Рисунок 2.2: Заполняем файл

Создаем исполняемый файл и запускаем его (рис. Рисунок ??).

The screenshot shows a terminal window with the following session:

```
(pkosterenko㉿kali)-[~/work/arch-pc/lab08]
$ nasm -f elf lab8-1.asm
(pkosterenko㉿kali)-[~/work/arch-pc/lab08]
$ ld -m elf_i386 -o lab8-1 lab8-1.o
(pkosterenko㉿kali)-[~/work/arch-pc/lab08]
$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
```

Рисунок 2.3: Запускаем файл и проверяем его работу

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

The screenshot shows a terminal window with a dark background. At the top, it says "pvkosterenko@kali: ~/work/arch-pc/lab08". Below that is the assembly code for lab8-1.asm:

```
GNU nano 8.4      /home/pvkosterenko/work/arch-pc/lab08/lab8-1.asm
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

```

At the bottom of the terminal window, there is a menu bar with various keyboard shortcuts:

- ^G Help
- ^O Write Out
- ^F Where Is
- ^K Cut
- ^T Execute
- ^X Exit
- ^R Read File
- ^V Replace
- ^U Paste
- ^J Justify

Рисунок 2.4: Изменяем файл

Создаем исполняемый файл и запускаем его (рис. Рисунок ??).

The screenshot shows a terminal window with a dark background. It displays the following commands and their output:

```
(pvkosterenko@kali)-[~/work/arch-pc/lab08]
$ nasm -f elf lab8-1.asm
(pvkosterenko@kali)-[~/work/arch-pc/lab08]
$ ld -m elf_i386 -o lab8-1 lab8-1.o
(pvkosterenko@kali)-[~/work/arch-pc/lab08]
$ ./lab8-1
Введите N: 10
9
7
5
3
1
```

Рисунок 2.5: 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. Рисунок ??).

```
GNU nano 8.4      /home/pvkosterenko/work/arch-pc/lab08/lab8
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 iprintf
pop ecx ; извлечение значения ecx из стека
loop label[]

[ Wrote 29 lines ]
^G Help      ^O Write Out ^F Where Is  ^K Cut      ^T Execut
^X Exit      ^R Read File ^\ Replace   ^U Paste    ^J Justifi
```

Рисунок 2.6: Редактируем файл

Создаем исполняемый файл и запускаем его (рис. Рисунок ??).

```
(pvkosterenko㉿kali)-[~/work/arch-pc/lab08]
└─$ nasm -f elf lab8-1.asm

(pvkosterenko㉿kali)-[~/work/arch-pc/lab08]
└─$ ld -m elf_i386 -o lab8-1 lab8-1.o

(pvkosterenko㉿kali)-[~/work/arch-pc/lab08]
└─$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
```

Рисунок 2.7: Проверяем, сошелся ли наш вывод с данным в условии выводом

В данном случае число проходов цикла равна числу N.

2.2 Обработка аргументов командной строки.

Создаем новый файл (рис. Рисунок ??).

```
(pvkosterenko㉿kali)-[~/work/arch-pc/lab08]
└─$ touch lab8-2.asm

(pvkosterenko㉿kali)-[~/work/arch-pc/lab08]
└─$
```

Рисунок 2.8: Создаем файл командой touch

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

The screenshot shows a terminal window titled 'pvkosterenko@kali: ~/work/arch-pc/lab08'. It displays the contents of a file named 'lab8-2.asm' using the nano editor. The code is assembly language with comments in Russian. It includes sections for .text, .start, .next, and .end, and uses various assembly instructions like pop, cmp, jz, call, and loop. A message '[Wrote 20 lines]' is visible at the bottom right of the editor window. The terminal prompt is '\$'.

```
GNU nano 8.4      /home/pvkosterenko/work/arch-pc/lab08/lab8-2.asm
%include 'in_out.asm'
SECTION .text
global _start
_start:
    pop ecx ; Извлекаем из стека в `есх` количество
    ; аргументов (первое значение в стеке)
    pop edx ; Извлекаем из стека в `edx` имя программы
    ; (второе значение в стеке)
    sub ecx, 1 ; Уменьшаем `есх` 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 sprintLF ; вызываем функцию печати
    loop next ; переход к обработке следующего
    ; аргумента (переход на метку `_next`)
_end:
    call quit[]

[Goto] [Wrote 20 lines]
^G Help      ^O Write Out ^F Where Is ^K Cut      ^T Execute
^X Exit      ^R Read File ^V Replace   ^U Paste   ^J Justify
```

Рисунок 2.9: Заполняем файл

Создаем исполняемый файл и проверяем его работу, указав аргументы
Програмой было обработано 3 аргумента.

Создаем новый файл lab8-3.asm (рис. Рисунок ??).

The screenshot shows a terminal window with a single command: '\$ touch lab8-3.asm'. The terminal prompt is '\$'.

```
(pvkosterenko㉿kali)-[~/work/arch-pc/lab08]
$ touch lab8-3.asm

(pvkosterenko㉿kali)-[~/work/arch-pc/lab08]
$
```

Рисунок 2.10: Создаем файл командой touch

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

```
GNU nano 8.4      /home/pvkosterenko/work/arch-pc/lab08/lab8-:
%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
^G Help      ^O Write Out ^F Where Is  ^K Cut      ^T Execute
^X Exit      ^R Read File ^V Replace   ^U Paste   ^J Justify
```

Рисунок 2.11: Заполняем файл

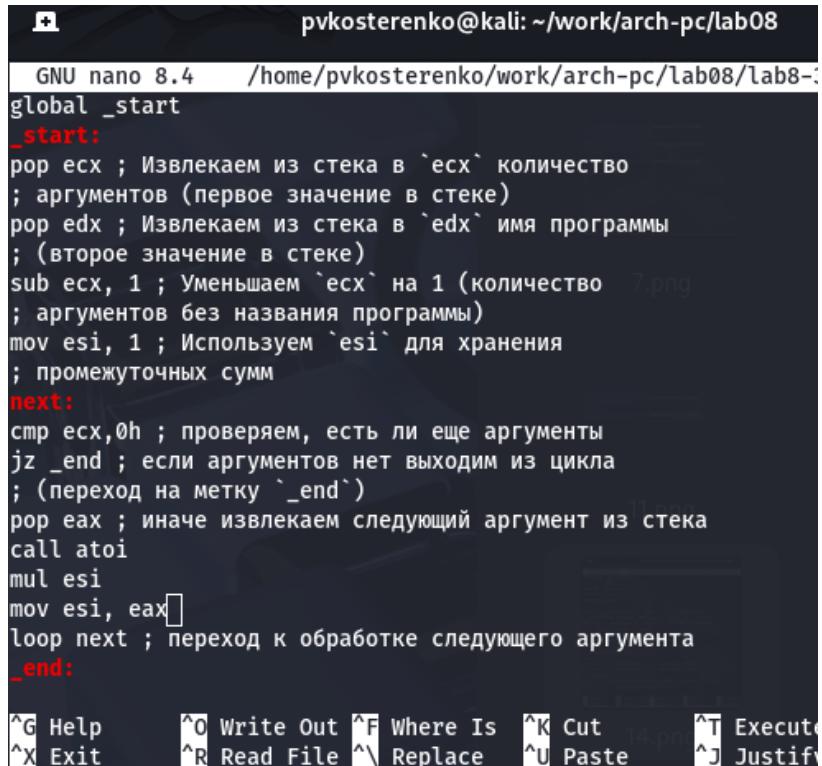
Создаём исполняемый файл и запускаем его, указав аргументы (рис. Рисунок ??).

```
(pvkosterenko㉿kali)-[~/work/arch-pc/lab08]
$ nasm -f elf lab8-3.asm
(pvkosterenko㉿kali)-[~/work/arch-pc/lab08]
$ ld -m elf_i386 -o lab8-3 lab8-3.o
(pvkosterenko㉿kali)-[~/work/arch-pc/lab08]
$ ./lab8-3
Результат: 0
(pvkosterenko㉿kali)-[~/work/arch-pc/lab08]
$ ./lab8-3 12 13 7 10 5
Результат: 47
(pvkosterenko㉿kali)-[~/work/arch-pc/lab08]
```

Рисунок 2.12: Смотрим на работу программы

Снова открываем файл для редактирования и изменяем его, чтобы вычислялось

произведение вводимых значений (рис. Рисунок ??).

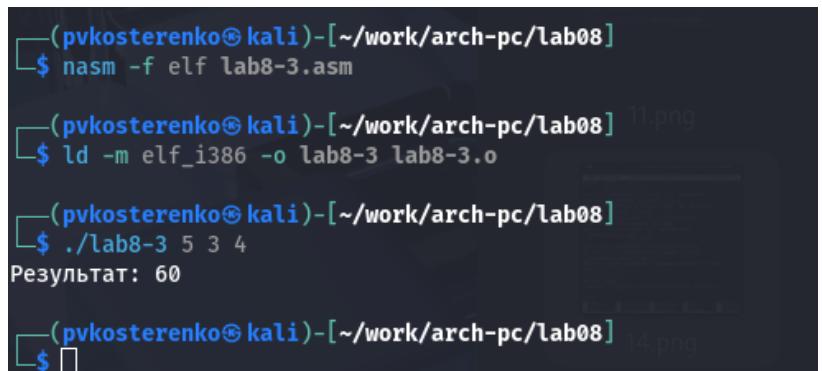


```
GNU nano 8.4      /home/pvkosterenko/work/arch-pc/lab08/lab8-3.asm
global _start
_start:
    pop ecx ; Извлекаем из стека в `ecx` количество
    ; аргументов (первое значение в стеке)
    pop edx ; Извлекаем из стека в `edx` имя программы
    ; (второе значение в стеке)
    sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
    ; аргументов без названия программы)
    mov esi, 1 ; 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
    mul esi
    mov esi, eax
loop next ; переход к обработке следующего аргумента
_end:

^G Help      ^O Write Out ^F Where Is  ^K Cut      ^T Execute
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify
```

Рисунок 2.13: Изменяем файл

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



```
(pvkosterenko㉿kali)-[~/work/arch-pc/lab08]
$ nasm -f elf lab8-3.asm
(pvkosterenko㉿kali)-[~/work/arch-pc/lab08]
$ ld -m elf_i386 -o lab8-3 lab8-3.o
(pvkosterenko㉿kali)-[~/work/arch-pc/lab08]
$ ./lab8-3 5 3 4
Результат: 60
(pvkosterenko㉿kali)-[~/work/arch-pc/lab08]
```

Рисунок 2.14: Проверяем работу файла(работает правильно)

2.3 Задание для самостоятельной работы

ВАРИАНТ-17

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

Создаем новый файл (рис. Рисунок ??).



```
(pvkosterenko㉿kali)-[~/work/arch-pc/lab08]
$ touch lab8-4.asm
```

The screenshot shows a terminal window with a dark background and light-colored text. It displays a command-line session where the user is in their home directory under the 'arch-pc' user on a Kali Linux system. The user has run the 'touch' command to create a file named 'lab8-4.asm'. The terminal window title is partially visible as '14.png'.

Рисунок 2.15: Создаем файл командой touch

Открываем его и пишем программу, которая выведет сумму значений, получившихся после решения выражения $10(x-1)$ (рис. Рисунок ??).

The screenshot shows a terminal window titled "GNU nano 8.4 /home/pvkosterenko/work/arch-pc/lab08/lab8-4". The code is an assembly program:

```
%include "in_out.asm"

SECTION .data
msg db "Результат: ",0

SECTION .bss
prm: resb 80

SECTION .text
global _start

_start:
    pop ecx
    pop edx
    sub ecx, 1
    mov esi, 10

next:
    cmp ecx, 0h
    jz _end
    pop eax
    call atoi
    sub eax, 1
    mul esi
    add [prm], eax
    loop next

_end:
    mov eax, msg
    call sprint
    mov eax, [prm]
    call iprintLF
    call quit
```

Рисунок 2.16: Пишем программу

Транслируем файл и смотрим на работу программы (рис. Рисунок ??).