

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

**ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №8**

Дисциплина: Архитектура компьютеров

Студент: Болдырева Дельгир

Группа: НКАбд-01-25

Москва

2025 г.

Оглавление

| | |
|--|----|
| 1 Цель работы | 3 |
| 2 Задание | 4 |
| 3 Теоретическое введение | 5 |
| 4 Выполнение лабораторной работы | 6 |
| 4.1 Реализация циклов в NASM..... | 6 |
| 4.2 Обработка аргументов командной строки..... | 9 |
| 5 Задания для самостоятельной работы | 13 |
| 6 Выводы | 14 |
| Список литературы | 15 |

1 Цель работы

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

2 Задания

1. Реализация циклом в NASM
2. Обработка аргументов командной строки
3. Самостоятельное написание программы по материалам лабораторной Работы

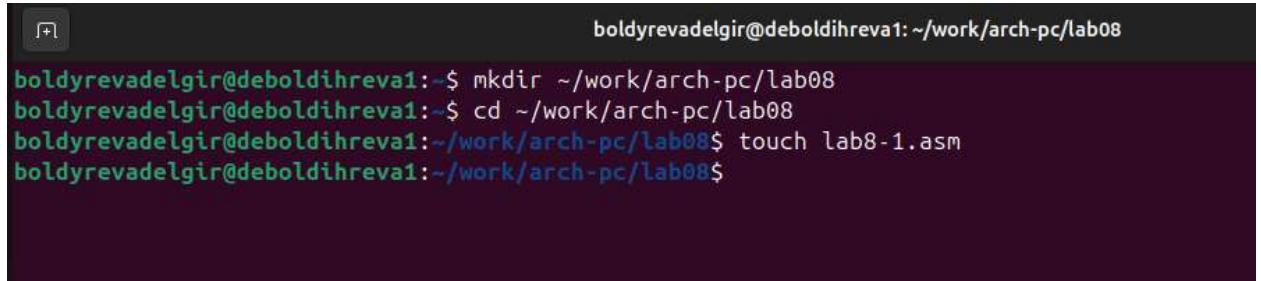
3 Теоретическое введение

Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл—первым ушёл»). Стек является частью архитектуры процессора и реализована аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды. Основной функцией стека является функция сохранения адресов возврата и передачи аргументов при вызове процедур. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров.

4 Выполнение лабораторной работы

4.1 Реализация циклов в NASM

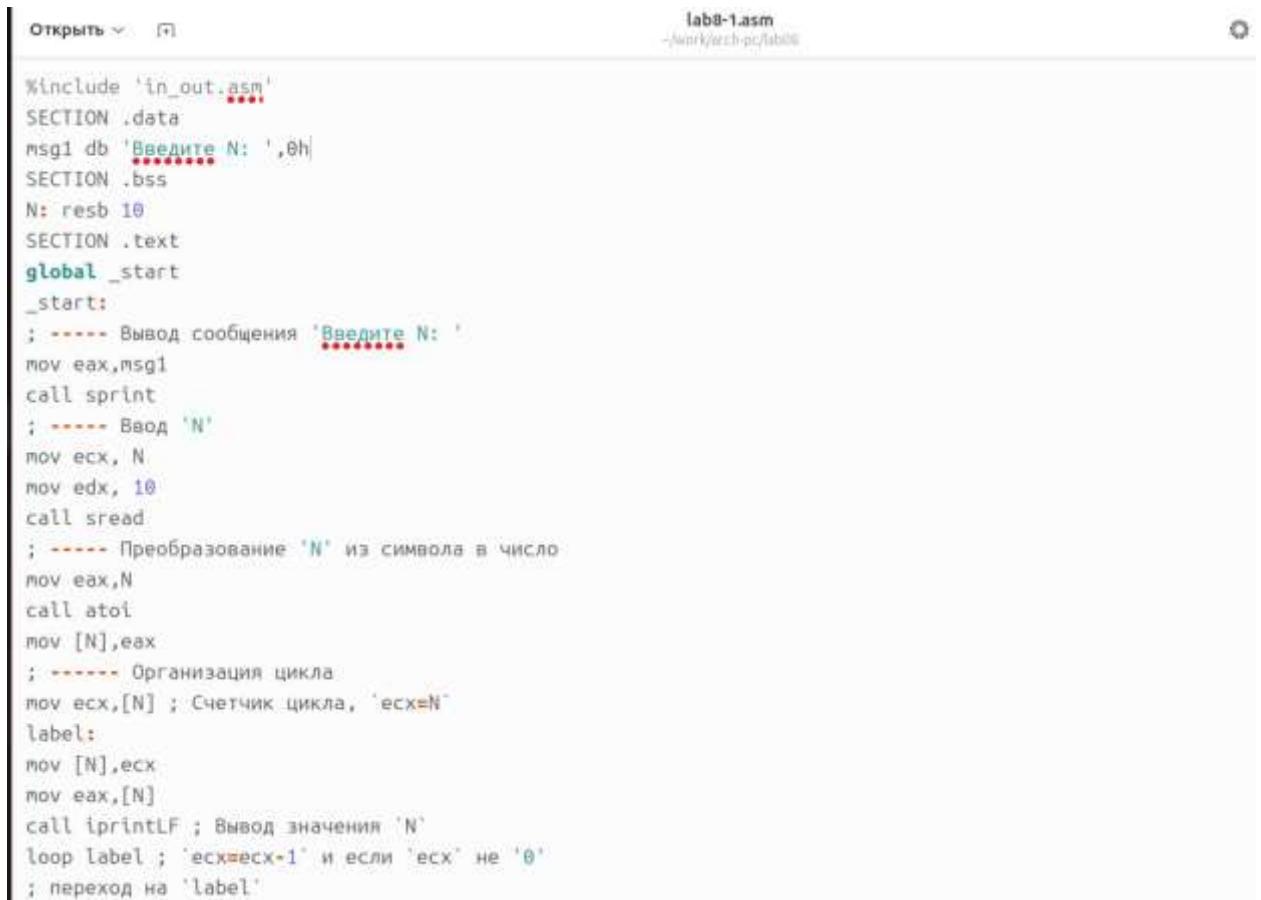
Создадим каталог для программ лабораторной работы №8, войдем в него и создадим файл lab8-1.asm (рис.4.1.1)



```
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab08
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab08$ mkdir ~./work/arch-pc/lab08
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab08$ cd ~./work/arch-pc/lab08
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab08$ touch lab8-1.asm
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab08$
```

Рис.4.1

Введем текст программы из листинга 8.1 в файл lab8-1.asm, сохраним и проверим как работает (рис.4.1.2-4.1.3)



```
lab8-1.asm
~/work/arch-pc/lab08

%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 iprintLF ; Вывод значения 'N'
loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
; переход на 'label'
```

Рис.4.1.2

```
boldyrevadelgir@deboldihrevali:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
boldyrevadelgir@deboldihrevali:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
boldyrevadelgir@deboldihrevali:~/work/arch-pc/lab08$ ./lab8-1
Введите N:
```

Рис.4.1.3

Проверим работу программы, введем N=10 (рис.4.1.4)

```
boldyrevadelgir@deboldihrevali:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
boldyrevadelgir@deboldihrevali:~/work/arch-pc/lab08$
```

Рис.4.1.4

Изменим текст программы добавив изменение значения регистра ecx в цикле(рис.4.1.5)



The screenshot shows a text editor window with the file 'lab8-1.asm' open. The code is as follows:

```
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 ; `ecx=ecx-1`
    mov [N],ecx
    mov eax,[N]
    call iprintLF
    loop label

    call quit
```

Рис.4.1.5

Запустим программу и проверим ее работу (рис.4.1.6)

```
boldyrevadelgir@deboldihrevali:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
boldyrevadelgir@deboldihrevali:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
boldyrevadelgir@deboldihrevali:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
```

Рис.4.1.6

Введем N=10 и посмотрим на результат (рис.4.1.7)

```
boldyrevadelgir@deboldihrevali:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
boldyrevadelgir@deboldihrevali:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
boldyrevadelgir@deboldihrevali:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
boldyrevadelgir@deboldihrevali:~/work/arch-pc/lab08$
```

Рис.4.1.7

Количество итераций уменьшается вдвое ввиду того, что регистр ecx на каждой итерации стал меньше на 2 значения

Внесем изменения в текст программы, добавив команды push и pop для сохранения значения счетчика цикла loop (рис.4.1.8)



The screenshot shows a text editor window with the assembly code for lab8-1.asm. The code is as follows:

```
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 iprintf
    pop ecx
loop label

    call quit
```

Рис.4.1.8

Сохраним программу и проверим. Введем N=10 , посмотрим на результат (рис.4.1.9-4.1.10)

```
boldyrevadelgir@deboldihreval:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
boldyrevadelgir@deboldihreval:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
boldyrevadelgir@deboldihreval:~/work/arch-pc/lab08$ ./lab8-1
Ведите N: 10
```

Рис.4.1.9

```
Ведите N: 10
9
8
7
6
5
4
3
2
1
0
boldyrevadelgir@deboldihreval:~/work/arch-pc/lab08$
```

Рис.4.1.10

Произошло смещение выводимых чисел на-1, но теперь количество итераций совпадает введенному N

4.2 Обработка аргументов командной строки

Создадим файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введем в него текст программы из листинга (рис.4.2.1-4.2.2)

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

Рис.4.2.1

The screenshot shows a terminal window with the title bar "lab8-2.asm" and the path "/work/arch-oc/lab08". There are two tabs: "lab8-1.asm" and "lab8-2.asm", with "lab8-2.asm" being the active tab. The code in the editor is:

```
%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.4.2.2

Проверим как работает программа и что она выводит (рис.4.2.3-4.2.4)

```
boldyrevadelgir@deboldihreval:~/work/arch-oc/lab08$ nasm -f elf lab8-2.asm
boldyrevadelgir@deboldihreval:~/work/arch-oc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
boldyrevadelgir@deboldihreval:~/work/arch-oc/lab08$ ./lab8-2 аргумент1 аргумент2 'аргумент3'
```

Рис.4.2.3

```
boldyrevadelgir@deboldihreval:~/work/arch-oc/lab08$ nasm -f elf lab8-2.asm
boldyrevadelgir@deboldihreval:~/work/arch-oc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
boldyrevadelgir@deboldihreval:~/work/arch-oc/lab08$ ./lab8-2 аргумент1 аргумент2 'аргумент3'
аргумент1
аргумент2
аргумент3
boldyrevadelgir@deboldihreval:~/work/arch-oc/lab08$
```

Рис.4.2.4

Программа обработала то же количество аргументов, что и было введено

Создадим файл lab8-3.asm в каталоге ~/work/arch pc/lab08 и введем в него текст программы из листинга(рис.4.2.5-4.2.6)

```
boldyrevadelgir@deboldihreval:~/work/arch-oc/lab08$ touch lab8-3.asm
boldyrevadelgir@deboldihreval:~/work/arch-oc/lab08$
```

Рис.4.2.5

The screenshot shows a code editor window with three tabs. The active tab is 'lab8-3.asm' containing the following assembly code:

```
%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
```

Рис.4.2.6

Запустим файл и проверим что он выводит(рис.4.2.7-4.2.8)

```
boldyrevadelgir@deb0ldihrevai:~/work/arch-pc/lab8$ nasm -f elf lab8-3.asm
boldyrevadelgir@deb0ldihrevai:~/work/arch-pc/lab8$ ld -m elf_i386 -o lab8-3 lab8-3.o
boldyrevadelgir@deb0ldihrevai:~/work/arch-pc/lab8$ ./lab8-3
```

Рис.4.2.7

```
boldyrevadelgir@deb0ldihrevai:~/work/arch-pc/lab8$ ./lab8-3 16 5 14 7 11
Результат: 53
boldyrevadelgir@deb0ldihrevai:~/work/arch-pc/lab8$
```

Рис.4.2.8

Изменим текст программы из листинга для вычисления произведения аргументов командной строки и проверим результат который он выводит(рис.4.2.9-4.2.10)

```
%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 atol
    imul esi, eax

    dec ecx
    jmp next

.
```

Рис.4.2.9

```
boldyrevadelgir@deboldihrevali:~/work/arch-pc/lab8$ nasm -f elf lab8-3.asm
boldyrevadelgir@deboldihrevali:~/work/arch-pc/lab8$ ld -m elf_i386 -o lab8-3 lab8-3.o
boldyrevadelgir@deboldihrevali:~/work/arch-pc/lab8$ ./lab8-3 16 5 14 7 11
Результат: 86240
boldyrevadelgir@deboldihrevali:~/work/arch-pc/lab8$
```

Рис.4.2.10

5 Задания для самостоятельной работы

Напишем программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, т.е. программа должна выводить значение $f(x_1) + f(x_2) + \dots + f(x_n)$.

Выражение для $f(x) = 10x - 5$. Создадим файл lab8-4.asm (рис.5.1-5.2)

```
boldyrevadelgir@deboldihreval:~/work/arch-pc/lab00$ touch lab8-4.asm
boldyrevadelgir@deboldihreval:~/work/arch-pc/lab00$
```

Рис.5.1


```
Открыть ▾
```

```
lab0-1.asm lab0-2.asm lab0-3.asm lab8-4.asm
```

```
%include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x)=10x-5", 0
msg_res db "Результат: ", 0

SECTION .bss
x resd 1

SECTION .text
global _start

_start:
pop ecx          ; argc
pop edx          ; argv[0] (имя программы)
sub ecx, 1       ; количество аргументов
mov ebx, 0        ; накопитель суммы

next:
cmp ecx, 0
jz end

pop eax          ; берём очередной аргумент
call atoi         ; преобразуем в число
mov [x], eax      ; сохраним x
```

Рис.5.2

Запустим программу и посмотрим результат (рис.5.3)

```
boldyrevadelgir@deboldihreval:~/work/arch-pc/lab00$ touch lab8-4.asm
boldyrevadelgir@deboldihreval:~/work/arch-pc/lab00$ nasm -f elf lab8-4.asm
boldyrevadelgir@deboldihreval:~/work/arch-pc/lab00$ ld -m elf_i386 -o lab8-4 lab8-4.o
boldyrevadelgir@deboldihreval:~/work/arch-pc/lab00$ ./lab8-4 1 2 3 4
Функция: f(x)=10x-5
Результат: 88
boldyrevadelgir@deboldihreval:~/work/arch-pc/lab00$
```

Рис.5.3

6 Выводы

Я приобрела навыки написания программ с использованием циклов и обработкой аргументов командной строки.

Список литературы

1. GDB:TheGNUProjectDebugger.—URL:<https://www.gnu.org/software/gdb/>.
2. GNUBashManual.—2016.—URL:<https://www.gnu.org/software/bash/manual/>.
3. Midnight CommanderDevelopment Center.—2021.—URL: <https://midnightcommander.org/>.
4. NASMAssemblyLanguageTutorials.—2021.—URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 c. —(In a Nutshell). —ISBN 0596009658.—URL: <http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658>.
6. RobbinsA. Bash Pocket Reference.—O'Reilly Media,2016.—156 c.—ISBN 978-1491941591.