

# Шаблон отчёта по лабораторной работе

---

## архитектура компьютера

мохамед Муса

## Цель работы

---

Цель этой работы - попрактиковаться в написании математических функций в ассемблерном коде и попрактиковаться в этом еще больше.

## выполнения лабораторной работы

---

- Сначала я создал файл lab8-1.asm, скопировал код из pdf и запустил его :

```
bs/lab08/report$ nasm -f elf lab8-1.asm
liveuser@localhost-live:~/work/study/2023-2024/архитектура компьютера/arch-pc/la
bs/lab08/report$ ld -m elf_i386 -o lab8-1 lab8-1.o
liveuser@localhost-live:~/work/study/2023-2024/архитектура компьютера/arch-pc/la
bs/lab08/report$ ./lab8-1
Введите N: 12 13 7 10 5
12
11
10
9
8
7
6
5
4
3
2
1
```

```

lab8-1.asm
~/work/study/2023-2024/архитектура компьютера/arch-pc/labs/lab08/report
Save
1 ;-----
2 ; Программа вывода значений регистра 'ecx'
3 ;-----
4 %include 'in_out.asm'
5 SECTION .data
6 msg1 db 'Введите N: ',0h
7 SECTION .bss
8 N: resb 10
9 SECTION .text
10 global _start
11 _start:
12 ; ---- Вывод сообщения 'Введите N: '
13 mov eax,msg1
14 call sprint
15 ; ---- Ввод 'N'
16 mov ecx, N
17 mov edx, 10
18 call sread
19 ; ---- Преобразование 'N' из символа в число
20 mov eax,N
21 call atoi
22 mov [N],eax
23 ; ----- Организация цикла
24 mov ecx,[N] ; Счетчик цикла, `ecx=N`
25 label:
26 sub ecx,1
27 mov [N],ecx
28 mov eax,[N]
29 call iprintLF ; Вывод значения `N`
30 loop label ; `ecx=ecx-1` и если `ecx` не `0`
31 ; переход на `label`
32 call quit

```

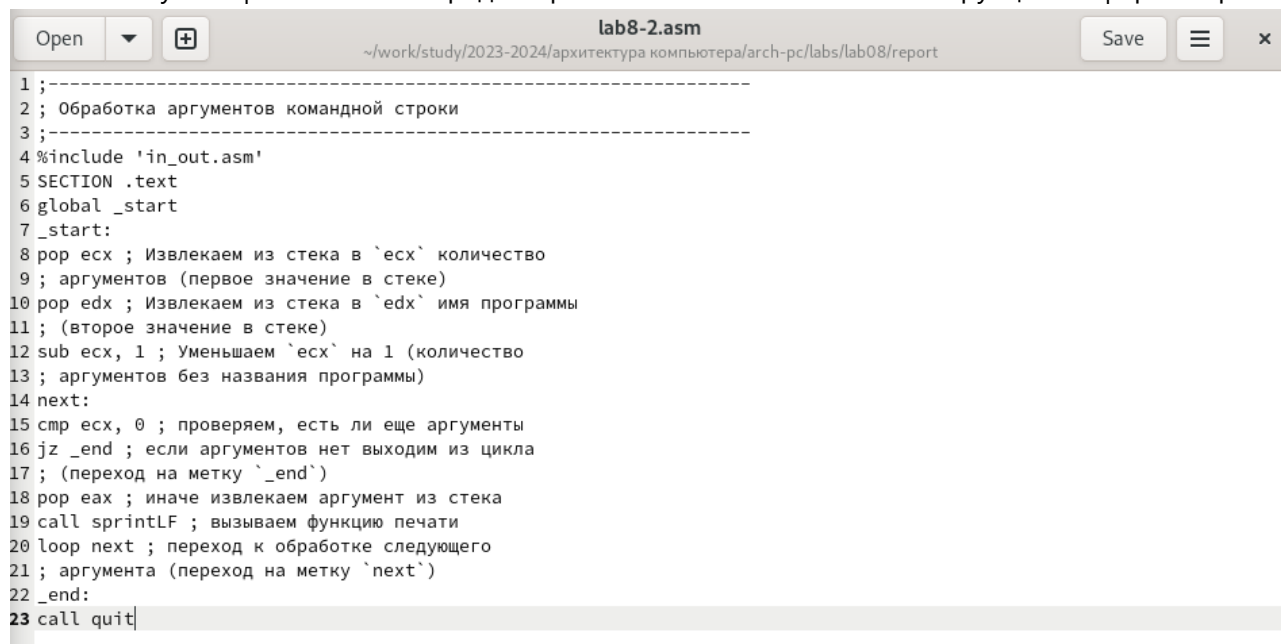
- И я внес необходимые изменения из pdf-файл в lab8-1.asm и запустил его снова :

```

bs/lab08/report$ gedit lab8-1.asm
liveuser@localhost-live:~/work/study/2023-2024/архитектура компьютера/arch-pc/la
bs/lab08/report$ nasm -f elf lab8-1.asm
liveuser@localhost-live:~/work/study/2023-2024/архитектура компьютера/arch-pc/la
bs/lab08/report$ ld -m elf_i386 -o lab8-1 lab8-1.o
liveuser@localhost-live:~/work/study/2023-2024/архитектура компьютера/arch-pc/la
bs/lab08/report$ ./lab8-1
Введите N: 12 45 64
11
9
7
5
3
1
liveuser@localhost-live:~/work/study/2023-2024/архитектура компьютера/arch-pc/la
bs/lab08/report$
liveuser@localhost-live:~/work/study/2023-2024/архитектура компьютера/arch-pc/la
bs/lab08/report$ ./lab8-1
Введите N: 5
4
3
2
1
0
liveuser@localhost-live:~/work/study/2023-2024/архитектура компьютера/arch-pc/la

```

- Я также запустил файл lab8-2 и отредактировал его в соответствии с инструкцией в формате pdf :



```

1 ;-----
2 ; Обработка аргументов командной строки
3 ;-----
4 %include 'in_out.asm'
5 SECTION .text
6 global _start
7 _start:
8 pop ecx ; Извлекаем из стека в `ecx` количество
9 ; аргументов (первое значение в стеке)
10 pop edx ; Извлекаем из стека в `edx` имя программы
11 ; (второе значение в стеке)
12 sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
13 ; аргументов без названия программы)
14 next:
15 cmp ecx, 0 ; проверяем, есть ли еще аргументы
16 jz _end ; если аргументов нет выходим из цикла
17 ; (переход на метку `_end`)
18 pop eax ; иначе извлекаем аргумент из стека
19 call sprintf ; вызываем функцию печати
20 loop next ; переход к обработке следующего
21 ; аргумента (переход на метку `next`)
22 _end:
23 call quit

```

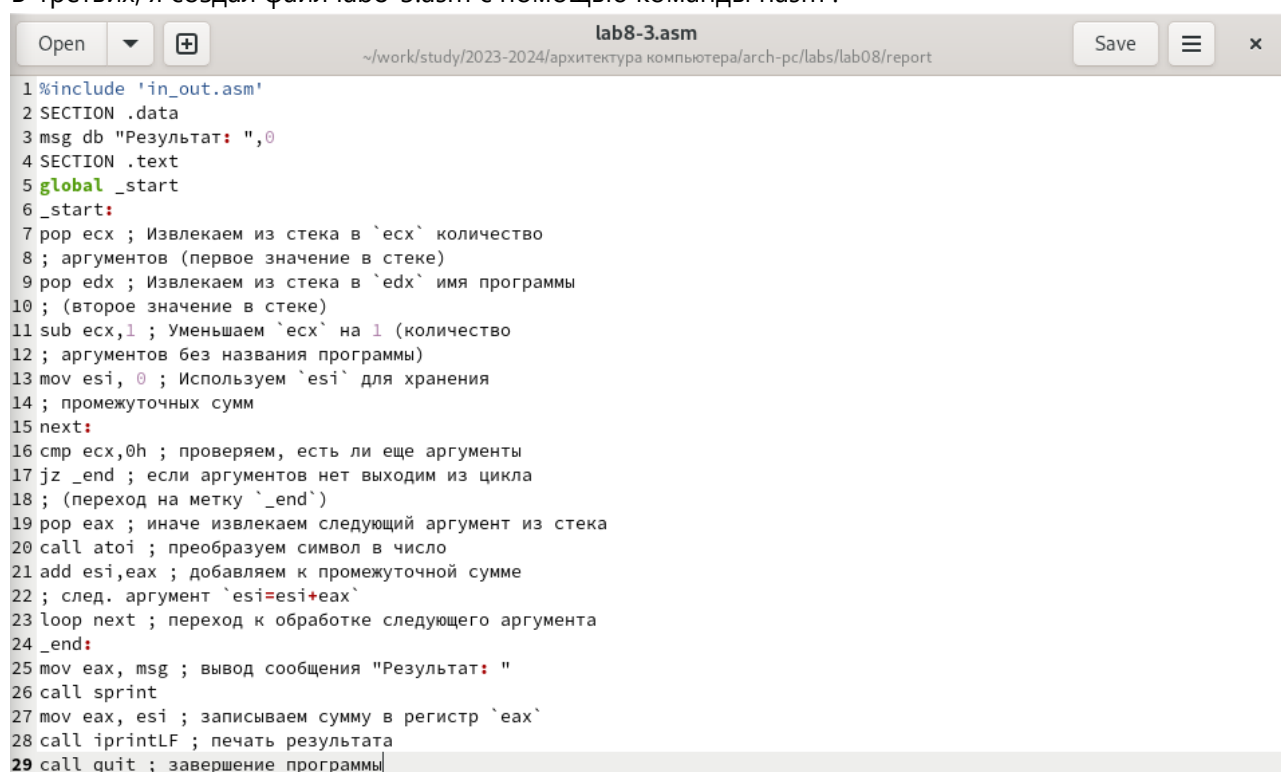
{#fig:001 width=70%}

```

bs/lab08/report$ gedit lab8-2.asm
liveuser@localhost-live:~/work/study/2023-2024/архитектура компьютера/arch-pc/la
bs/lab08/report$ nasm -f elf lab8-2.asm
liveuser@localhost-live:~/work/study/2023-2024/архитектура компьютера/arch-pc/la
bs/lab08/report$ ld -m elf_i386 -o lab8-2 lab8-2.o
liveuser@localhost-live:~/work/study/2023-2024/архитектура компьютера/arch-pc/la
bs/lab08/report$ ./lab8-2
liveuser@localhost-live:~/work/study/2023-2024/архитектура компьютера/arch-pc/la
bs/lab08/report$ ./lab8-2 2 4 6
2
4
6

```

- В-третьих, я создал файл lab8-3.asm с помощью команды nasm :



```

1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx ; Извлекаем из стека в `ecx` количество
8 ; аргументов (первое значение в стеке)
9 pop edx ; Извлекаем из стека в `edx` имя программы
10 ; (второе значение в стеке)
11 sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
12 ; аргументов без названия программы)
13 mov esi, 0 ; Используем `esi` для хранения
14 ; промежуточных сумм
15 next:
16 cmp ecx,0h ; проверяем, есть ли еще аргументы
17 jz _end ; если аргументов нет выходим из цикла
18 ; (переход на метку `_end`)
19 pop eax ; иначе извлекаем следующий аргумент из стека
20 call atoi ; преобразуем символ в число
21 add esi,eax ; добавляем к промежуточной сумме
22 ; след. аргумент `esi=esi+eax`
23 loop next ; переход к обработке следующего аргумента
24 _end:
25 mov eax, msg ; вывод сообщения "Результат: "
26 call sprint
27 mov eax, esi ; записываем сумму в регистр `eax`
28 call iprintLF ; печать результата
29 call quit ; завершение программы

```

```
bs/lab08/report$ touch lab8-3.asm
liveuser@localhost-live:~/work/study/2023-2024/архитектура компьютера/arch-pc/la
bs/lab08/report$ gedit lab8-3.asm
liveuser@localhost-live:~/work/study/2023-2024/архитектура компьютера/arch-pc/la
bs/lab08/report$ nasm -f elf lab8-3.asm
liveuser@localhost-live:~/work/study/2023-2024/архитектура компьютера/arch-pc/la
bs/lab08/report$ ld -m elf_i386 -o lab8-3 lab8-3.o
liveuser@localhost-live:~/work/study/2023-2024/архитектура компьютера/arch-pc/la
bs/lab08/report$ ./lab8-3 12 13 7 10 5
Результат: 47
```

## Выполнения заданий для самостоятельной работы:

- я написал программу и запустил ее :

```
bs/lab08/report$ gedit program1.asm
liveuser@localhost-live:~/work/study/2023-2024/архитектура компьютера/arch-pc/la
bs/lab08/report$ nasm -f elf program1.asm
liveuser@localhost-live:~/work/study/2023-2024/архитектура компьютера/arch-pc/la
bs/lab08/report$ ld -m elf_i386 -o program1 program1.o
liveuser@localhost-live:~/work/study/2023-2024/архитектура компьютера/arch-pc/la
bs/lab08/report$ ./program1 12 13 7 10
f(x)= 15x + 2
Результат: 638
```



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 fx: db 'f(x)= 15x + 2',0
5
6 SECTION .text
7 global _start
8 _start:
9 mov eax, fx
10 call sprintf
11 pop ecx
12 pop edx
13 sub ecx,1
14 mov esi, 0
15
16 next:
17 cmp ecx,0h
18 jz _end
19 pop eax
20 call atoi
21 mov ebx,15
22 mul ebx
23 add eax,2
24 add esi,eax
25
26 loop next
27
28 _end:
29 mov eax, msg
30 call sprintf
31 mov eax, esi
32 call iprintLF
33 call quit
```

## Выводы

Мы больше изучали и практиковали ассемблер, а также научились манипулировать значениями с помощью ассемблера.