

# **answer to labrotary work 8**

**Discipline: Computer Architecture**

Ерфан Хосейнабади

# Content

<b>1</b>	<b>Goal of the Work</b>	<b>5</b>
<b>2</b>	<b>Assignment</b>	<b>6</b>
<b>3</b>	<b>Theoretical Introduction</b>	<b>7</b>
<b>4</b>	<b>Performing the Laboratory Work</b>	<b>8</b>
4.1	Implementing Loops in NASM . . . . .	8
4.2	Processing Command-Line Arguments . . . . .	15
4.3	Independent Work Assignment . . . . .	21
<b>5</b>	<b>Conclusions</b>	<b>26</b>
<b>6</b>	<b>References</b>	<b>27</b>

# List of illustrations

4.1	create file . . . . .	9
4.2	copy program from list . . . . .	10
4.3	run it . . . . .	11
4.4	change the program . . . . .	12
4.5	run the new one . . . . .	13
4.6	Adding push and pop to the program loop . . . . .	14
4.7	run the new program . . . . .	15
4.8	copy program from the list . . . . .	16
4.9	run the code . . . . .	17
4.10	Copying the program from the third listing . . . . .	18
4.11	run the third program . . . . .	19
4.12	change the program . . . . .	20
4.13	run the new program . . . . .	21
4.14	write the program for individual program . . . . .	22
4.15	run the program . . . . .	25

## List of Tables

# **1 Goal of the Work**

Acquiring skills in writing programs using loops and processing command-line arguments.

## **2 Assignment**

1. Loop implementation in NASM
2. Processing command-line arguments
3. Independent program writing based on the materials of the laboratory work

## 3 Theoretical Introduction

A stack is a data structure organized according to the LIFO principle (“Last In — First Out”). The stack is part of the processor architecture and is implemented at the hardware level. The processor has special registers (ss, bp, sp) and commands for working with the stack.

The main function of the stack is to save return addresses and pass arguments when calling procedures. In addition, memory is allocated in it for local variables, and register values can be temporarily stored.

## **4 Performing the Laboratory Work**

### **4.1 Implementing Loops in NASM**

I create a file for laboratory work No. 8 (Fig. -fig. 4.1).

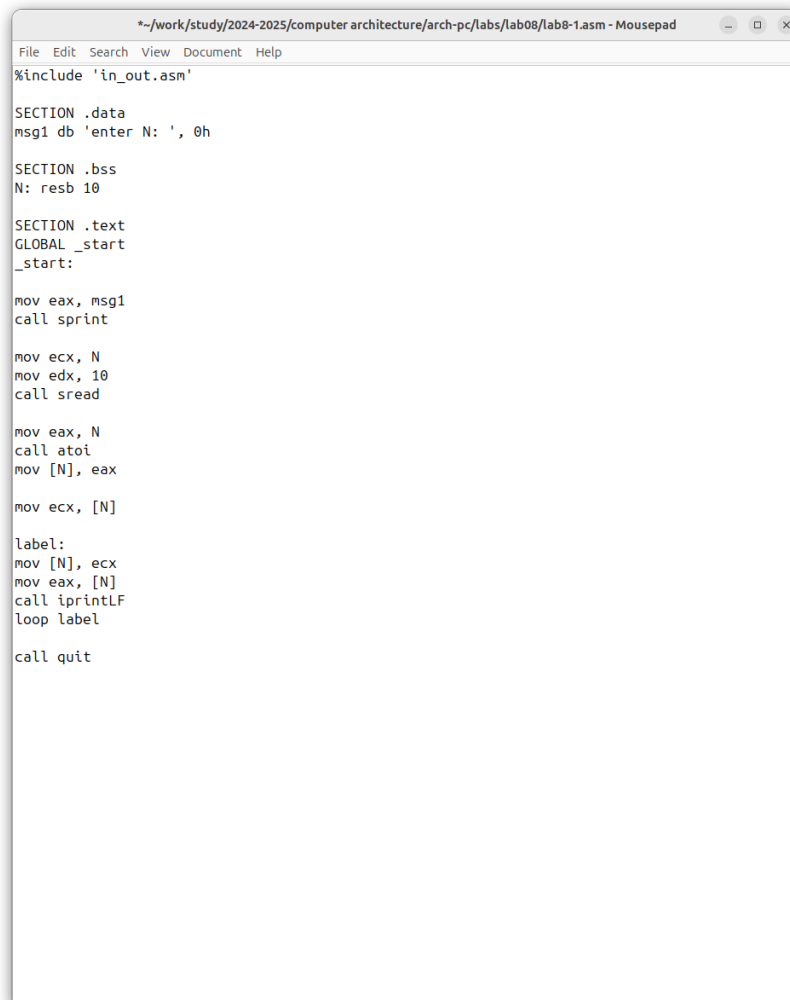


A terminal window with a dark purple background. The title bar at the top shows the user 'erfanhosseinabadi@ideapad' and the current directory path. The terminal displays two lines of text: a prompt followed by the command 'touch lab8-1.asm', and a second prompt line. The rest of the terminal is empty.

```
erfanhosseinabadi@ideapad: ~/work/study/2024-2025/computer architecture/arch-pc/lab...  
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$  
touch lab8-1.asm  
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
```

Fig. 4.1: create file

I copy the program from the listing into the created file (Fig. -fig. 4.2).

A screenshot of a text editor window titled "\*/work/study/2024-2025/computer architecture/arch-pc/labs/lab08/lab8-1.asm - Mousepad". The window contains the following assembly code:

```
File Edit Search View Document Help
%include 'in_out.asm'

SECTION .data
msg1 db 'enter 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 iprintLF
loop label

call quit
```

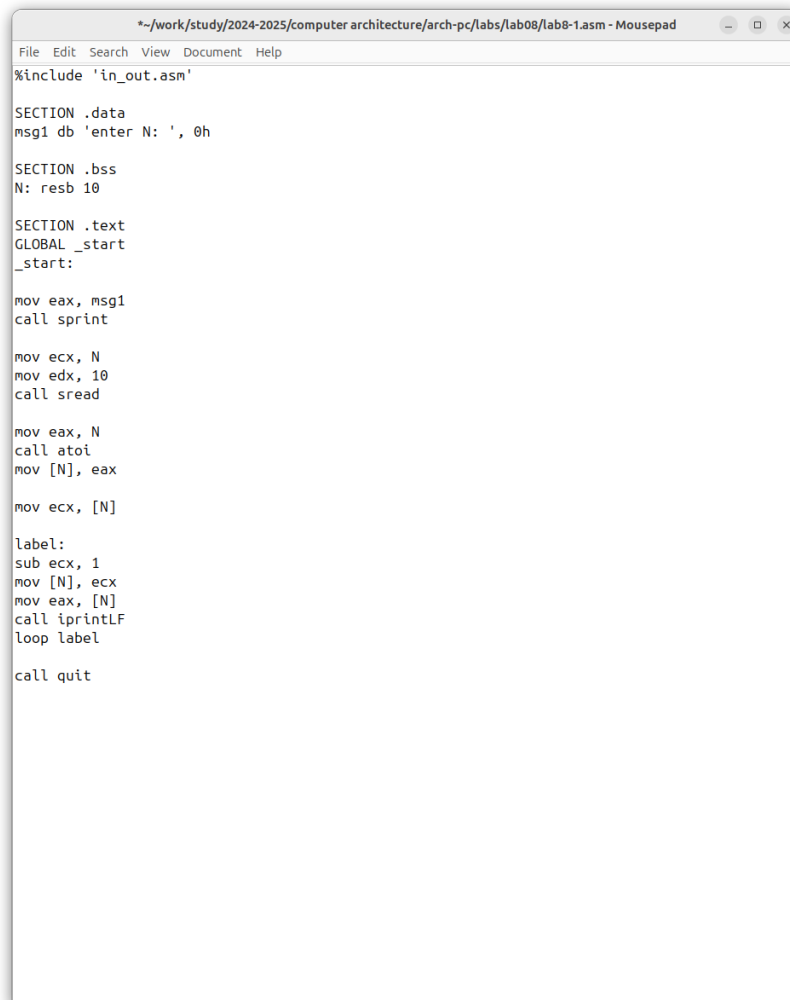
Fig. 4.2: copy program from list

I run the program; it shows the operation of loops in NASM (Fig. -fig. 4.3).

```
erfanhosseinabadi@ideapad: ~/work/study/2024-2025/computer architecture/arch-pc/lab...  
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$  
touch lab8-1.asm  
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$  
mousepad lab8-1.asm  
  
(mousepad:6616): Glib-CRITICAL **: 18:39:35.590: g_strjoinv: assertion 'str_array != NULL'  
failed  
  
(mousepad:6616): Glib-CRITICAL **: 18:39:35.590: g_strjoinv: assertion 'str_array != NULL'  
failed  
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$  
nasm -f elf lab8-1.asm  
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$  
ld -m elf_i386 -o lab8-1 lab8-1.o  
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$  
./lab8-1  
enter N: 10  
10  
9  
8  
7  
6  
5  
4  
3  
2  
1  
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$  
█
```

Fig. 4.3: run it

I replace the original program so that in the loop body I change the value of the ecx register (Fig. -fig. 4.4).



```
*~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08/lab8-1.asm - Mousepad
File Edit Search View Document Help
%include 'in_out.asm'

SECTION .data
msg1 db 'enter 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 iprintf
loop label

call quit
```

Fig. 4.4: change the program

Due to the fact that now the ecx register decreases by 2 values on each iteration, the number of iterations is halved (Fig. -fig. 4.5).

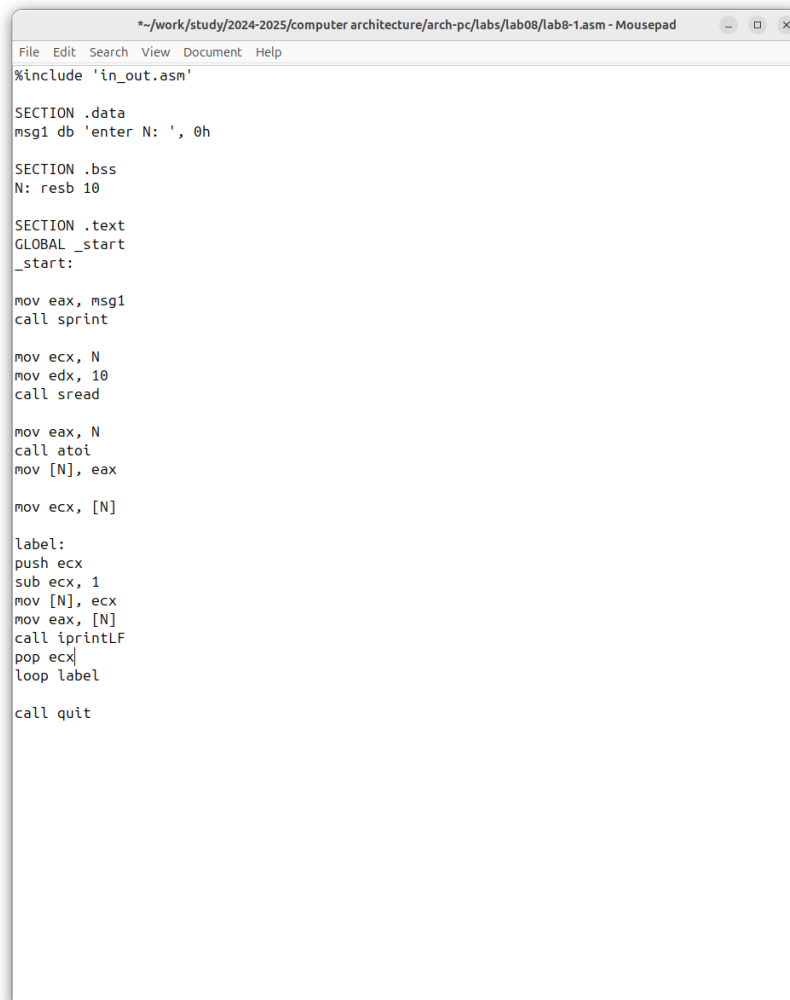
```
erfanhosseinabadi@ideapad: ~/work/study/2024-2025/computer architecture/arch-pc/lab...
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
mousepad lab8-1.asm

(mousepad:11208): GLib-CRITICAL **: 18:44:11.645: g_strjoinv: assertion 'str_array != NULL'
failed

(mousepad:11208): GLib-CRITICAL **: 18:44:11.645: g_strjoinv: assertion 'str_array != NULL'
failed
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
nasm -f elf lab8-1.asm
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
ld -m elf_i386 -o lab8-1 lab8-1.o
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
./lab8-1
enter N: 10
9
7
5
3
1
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
```

Fig. 4.5: run the new one

I add the push and pop commands to the program (Fig. -fig. 4.6).



```
*~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08/lab8-1.asm - Mousepad
File Edit Search View Document Help
%include 'in_out.asm'

SECTION .data
msg1 db 'enter 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 iprintLF
pop ecx
loop label

call quit
```

Fig. 4.6: Adding push and pop to the program loop

Now the number of iterations matches the entered N, but there was a shift in the output numbers by -1 (Fig. -fig. 4.7).

```
erfanhosseinabadi@ideapad: ~/work/study/2024-2025/computer architecture/arch-pc/lab...
failed

(mousepad:11208): GLib-CRITICAL **: 18:44:11.645: g_strjoinv: assertion 'str_array != NULL'
failed
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
nasm -f elf lab8-1.asm
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
ld -m elf_i386 -o lab8-1 lab8-1.o
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
./lab8-1
enter N: 10
9
7
5
3
1
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
mousepad lab8-1.asm

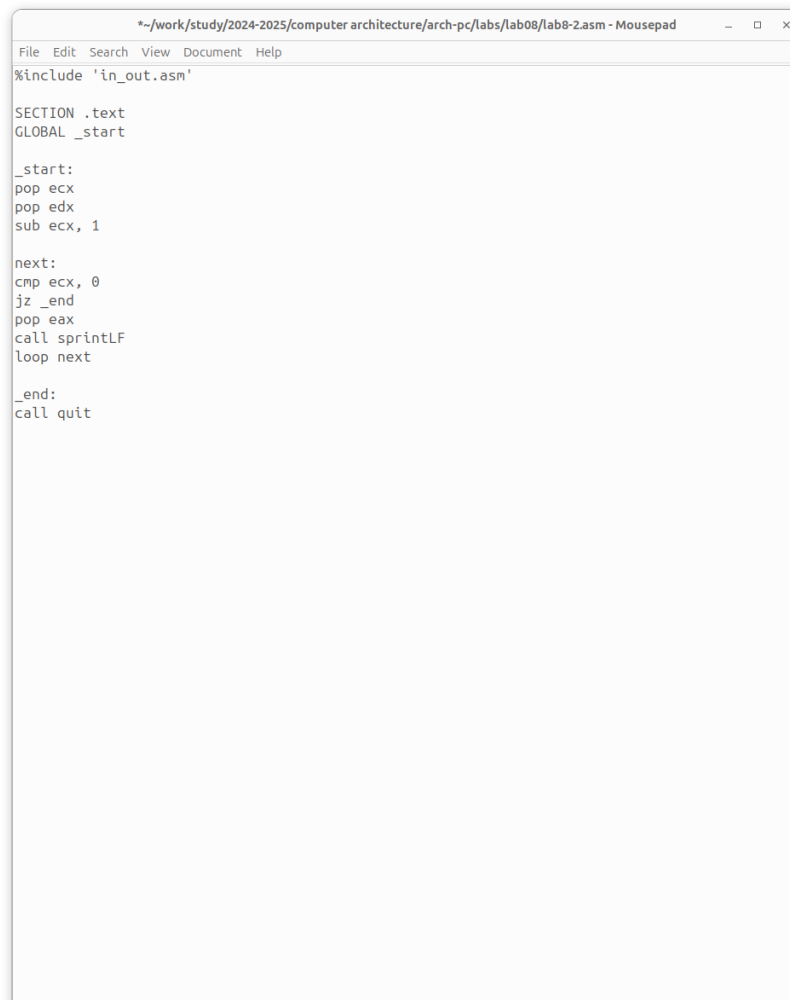
(mousepad:11696): GLib-CRITICAL **: 18:45:51.024: g_strjoinv: assertion 'str_array != NULL'
failed

(mousepad:11696): GLib-CRITICAL **: 18:45:51.024: g_strjoinv: assertion 'str_array != NULL'
failed
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
nasm -f elf lab8-1.asm
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
ld -m elf_i386 -o lab8-1 lab8-1.o
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
./lab8-1
enter N: 10
9
8
7
6
5
4
3
2
1
0
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
```

Fig. 4.7: run the new program

## 4.2 Processing Command-Line Arguments

I create a new file for the program and copy the code from the next listing into it (Fig. -fig. 4.8).

A screenshot of a text editor window titled "\*/work/study/2024-2025/computer architecture/arch-pc/labs/lab08/lab8-2.asm - Mousepad". The window contains assembly code for a copy program. The code includes a menu bar (File, Edit, Search, View, Document, Help) and a toolbar with icons for file operations. The code itself starts with an include directive for "in\_out.asm", followed by section and global declarations. It then defines a loop structure with labels "\_start:", "next:", and "\_end:". The loop body includes instructions to pop registers, subtract a counter, compare it to zero, jump if above, pop the next argument, call a printf function, and loop back. Finally, it calls a quit function at the end.

```
File Edit Search View Document Help
%include 'in_out.asm'

SECTION .text
GLOBAL _start

_start:
pop ecx
pop edx
sub ecx, 1

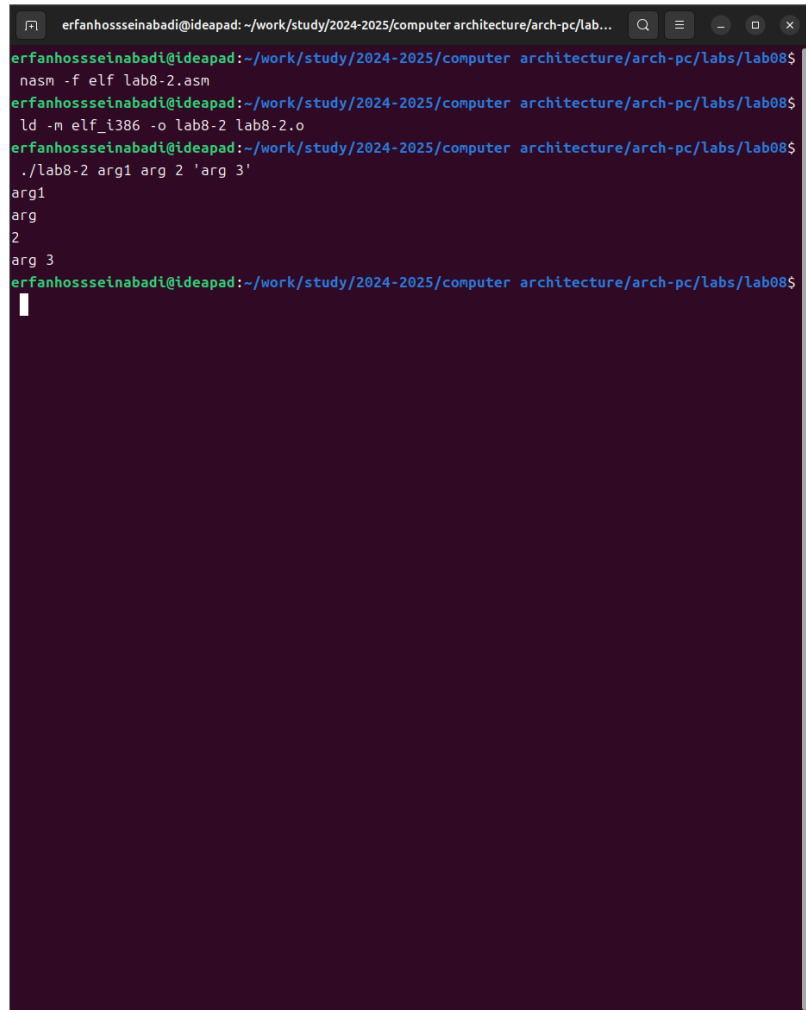
next:
cmp ecx, 0
jz _end
pop eax
call printf
loop next

_end:
call quit
```

Fig. 4.8: copy program from the list

I compile the program and run it, specifying the arguments. The program processed the same number of arguments as were entered (Fig. -fig. 4.9).

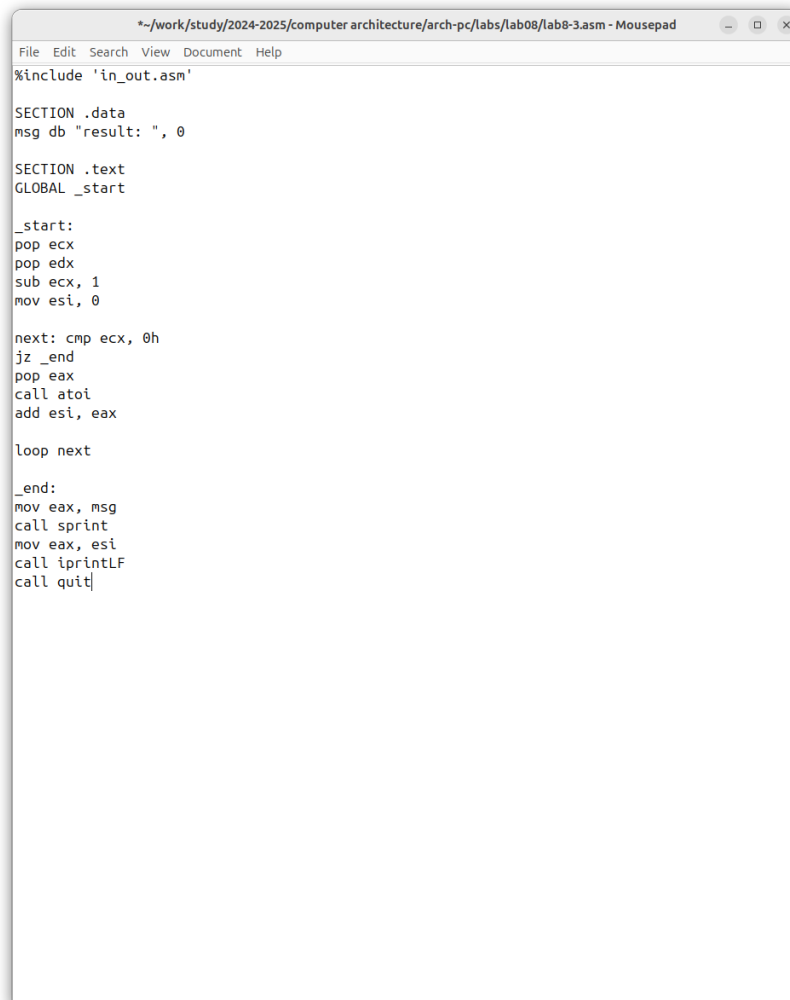




```
erfanhosseinabadi@ideapad: ~/work/study/2024-2025/computer architecture/arch-pc/labs...  
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$  
nasm -f elf lab8-2.asm  
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$  
ld -m elf_i386 -o lab8-2 lab8-2.o  
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$  
./lab8-2 arg1 arg 2 'arg 3'  
arg1  
arg  
2  
arg 3  
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
```

Fig. 4.9: run the code

I create a new file for the program and copy the code from the third listing into it (Fig. -fig. 4.10).

A screenshot of a text editor window titled '\*~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08/lab8-3.asm - Mousepad'. The window contains assembly code for a program that reads numbers from the command line and prints their sum. The code includes a data section for a message, a text section for the main logic, and a loop to process multiple arguments.

```
File Edit Search View Document Help
%include 'in_out.asm'

SECTION .data
msg db "result: ", 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
call sprint
mov eax, esi
call iprintLF
call quit
```

Fig. 4.10: Copying the program from the third listing

I compile the program and run it, specifying some numbers as arguments; the program adds them (Fig. -fig. 4.11).

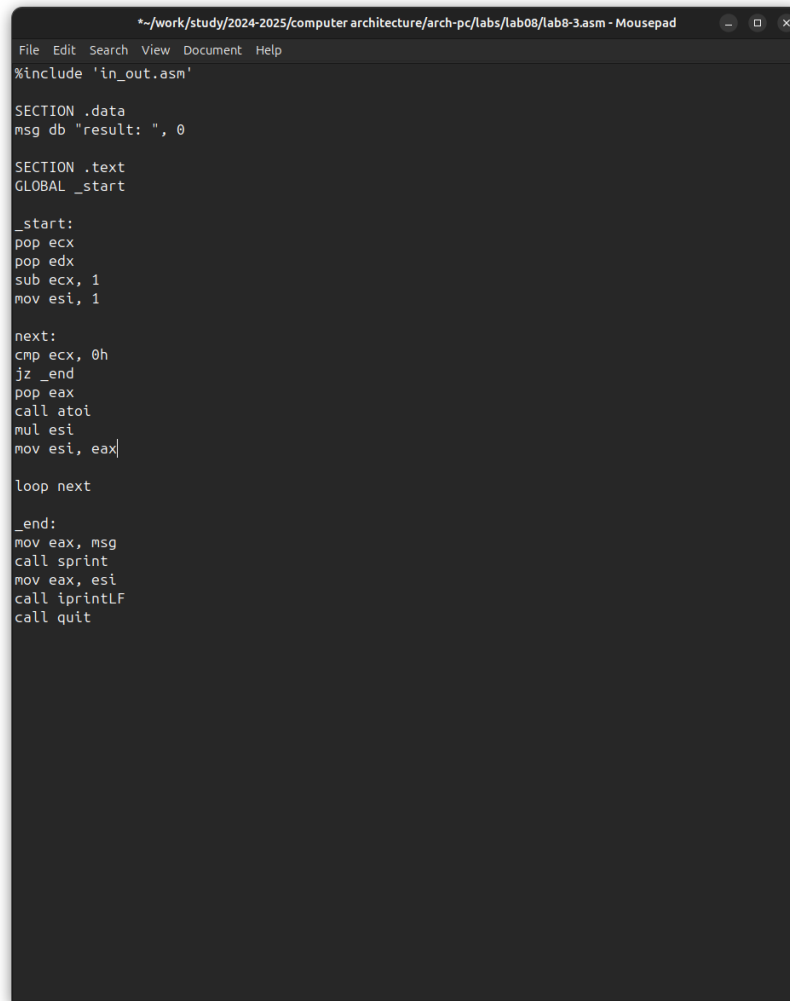
```
erfanhosseinabadi@ideapad: ~/work/study/2024-2025/computer architecture/arch-pc/lab...
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
mousepad lab8-3.asm

(mousepad:13137): GLib-CRITICAL **: 18:50:38.020: g_strjoinv: assertion 'str_array != NULL'
failed

(mousepad:13137): GLib-CRITICAL **: 18:50:38.020: g_strjoinv: assertion 'str_array != NULL'
failed
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
nasm -f elf lab8-3.asm
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
ld -m elf_i386 -o lab8-3 lab8-3.o
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
./lab8-3 12 13 7 10 5
result: 47
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
```

Fig. 4.11: run the third program

I change the program's behavior so that it multiplies the specified arguments instead of adding them (Fig. -fig. 4.12).



```
*~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08/lab8-3.asm - Mousepad
File Edit Search View Document Help
%include 'in_out.asm'

SECTION .data
msg db "result: ", 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 sprint
mov eax, esi
call iprintLF
call quit
```

Fig. 4.12: change the program

The program now actually multiplies the input numbers (Fig. -fig. 4.13).

```
erfanhosseinabadi@ideapad: ~/work/study/2024-2025/computer architecture/arch-pc/lab...
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
mousepad lab8-3.asm

(mousepad:13137): GLib-CRITICAL **: 18:50:38.020: g_strjoinv: assertion 'str_array != NULL'
failed

(mousepad:13137): GLib-CRITICAL **: 18:50:38.020: g_strjoinv: assertion 'str_array != NULL'
failed
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
nasm -f elf lab8-3.asm
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
ld -m elf_i386 -o lab8-3 lab8-3.o
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
./lab8-3 12 13 7 10 5
result: 47
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
mousepad lab8-3.asm

(mousepad:15700): GLib-CRITICAL **: 18:56:34.574: g_strjoinv: assertion 'str_array != NULL'
failed

(mousepad:15700): GLib-CRITICAL **: 18:56:34.574: g_strjoinv: assertion 'str_array != NULL'
failed
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
nasm -f elf lab8-3.asm
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
ld -m elf_i386 -o lab8-3 lab8-3.o
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
./lab8-3 111 1 6
result: 666
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$
```

Fig. 4.13: run the new program

## 4.3 Independent Work Assignment

I write a program that will find the sum of the values for the function  $f(x) = 5(2+x)$ , which matches my ninth variant (Fig. -fig. 4.14).

```
*~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08/lab8-4.asm - Mousepad
File Edit Search View Document Help
%include 'in_out.asm'

SECTION .data
msg_func db "function: f(x) = 5 *( 2 + x )", 0
msg_result db "results: ", 0

SECTION .text
GLOBAL _start
_start:
mov eax, msg_func
call sprintf
pop ecx
pop edx
sub ecx, 1
mov esi, 0
next:
cmp ecx, 0h
jz _end
pop eax
call atoi
add eax, 2
mov ebx, 5
mul ebx
add esi, eax
loop next
_end:
mov eax, msg_result
call sprintf
```

Fig. 4.14: write the program for individual program

Program code:

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg_func db "Функция: f(x) = 5 *( 2 + x )", 0
```

```
msg_result db "Результат: ", 0
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
mov eax, msg_func
```

```
call sprintf
```

```
pop ecx
```

```
pop edx
```

```
sub ecx, 1
```

```
mov esi, 0
```

```
next:
```

```
cmp ecx, 0h
```

```
jz _end
```

```
pop eax
```

```
call atoi ; Convert input string to integer in EAX
```

```

add eax, 2      ;  $f(x)$  starts with 2, so add 2 to  $x$ 
mov ebx, 5      ; Prepare to multiply by 5
mul ebx         ;  $EAX = EAX * 5$ 

add esi, eax    ; Accumulate the results

loop next

_end:

mov eax, msg_result

call sprint

mov eax, esi

call iprintLF

call quit

```

I check the program's operation, specifying several numbers as arguments (Fig. 4.15).



```
erfanhosseinabadi@ideapad: ~/work/study/2024-2025/computer architecture/arch-pc/lab...  
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$  
mousepad lab8-4.asm  
  
(mousepad:16348): GLib-CRITICAL **: 18:58:45.298: g_strjoinv: assertion 'str_array != NULL'  
failed  
  
(mousepad:16348): GLib-CRITICAL **: 18:58:45.299: g_strjoinv: assertion 'str_array != NULL'  
failed  
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$  
nasm -f elf lab8-4.asm  
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$  
ld -m elf_i386 -o lab8-4 lab8-4.o  
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$  
./lab8-3 1 2 3  
result: 6  
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab08$  
█
```

Fig. 4.15: run the program

## 5 Conclusions

As a result of this laboratory work, I acquired skills in writing programs using loops and also learned how to process command-line arguments.

## 6 References

1. Course on TUIS
2. Laboratory Work No. 8
3. Programming in NASM Assembler Language, Stolyarov A. V.