answer to labrotary work 7

Discipline: Computer Architecture

Ерфан Хосейнабади

Content

List of illustrations

List of Tables

The text file you provided contains a lab report detailing work with NASM assembly language, including conditional and unconditional jumps. I have translated the Russian sections into English, preserving all original spacing and punctuation.

Goal of the Work

Study of conditional and unconditional jump instructions. Acquisition of skills in writing programs using jumps. Familiarization with the purpose and structure of the listing file.

Assignment

1. Implementation of jumps in NASM
2. Study of the structure of listing files
3. Independent writing of programs based on the materials of the laboratory work

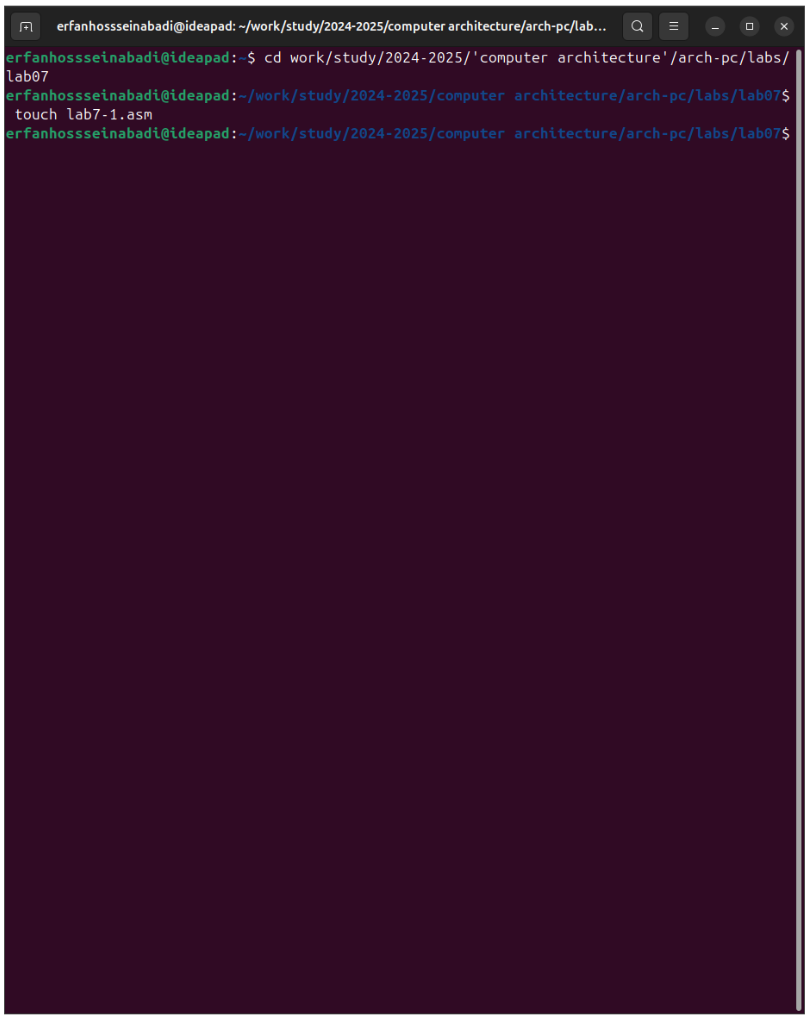
Theoretical Introduction

So-called control transfer instructions or jump instructions are used to implement branching in assembler. Two types of jumps can be distinguished: • conditional jump – execution or non-execution of a jump to a specific point in the program depending on the condition check. • unconditional jump – execution of control transfer to a specific point in the program without any conditions.

Laboratory Work Execution

Implementation of Jumps in NASM

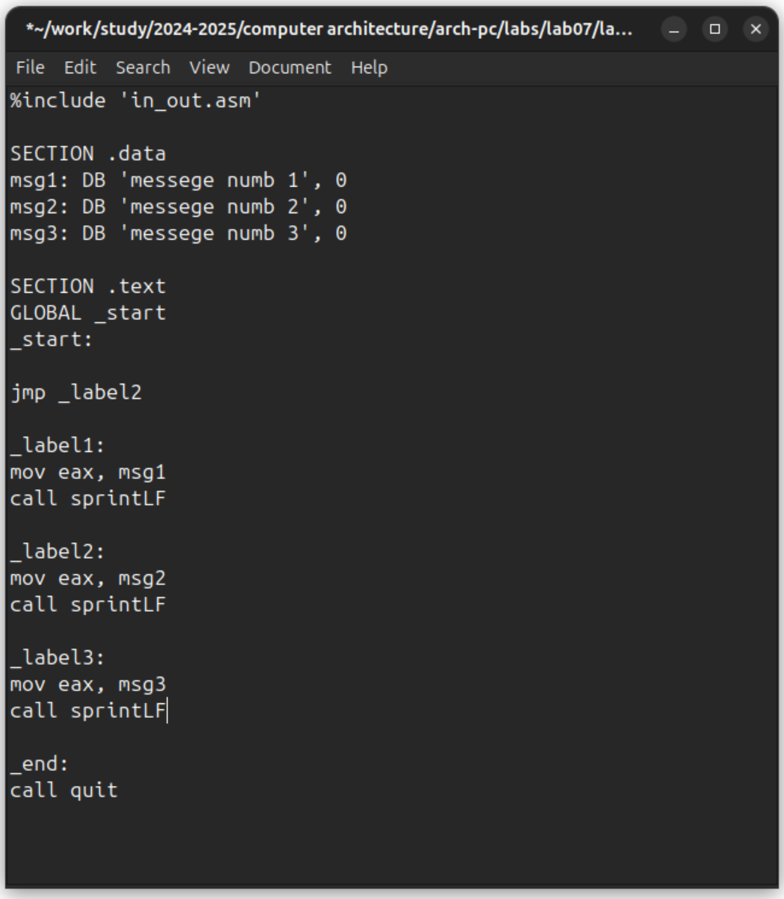
I create a file the programs of laboratory work No. 7 (Fig. -fig. 1).



```
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/lab...
erfanhossseinabadi@ideapad:~$ cd work/study/2024-2025/'computer architecture'/arch-pc/labs/
lab07
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 touch lab7-1.asm
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
```

Fig. 1: Ccreate file

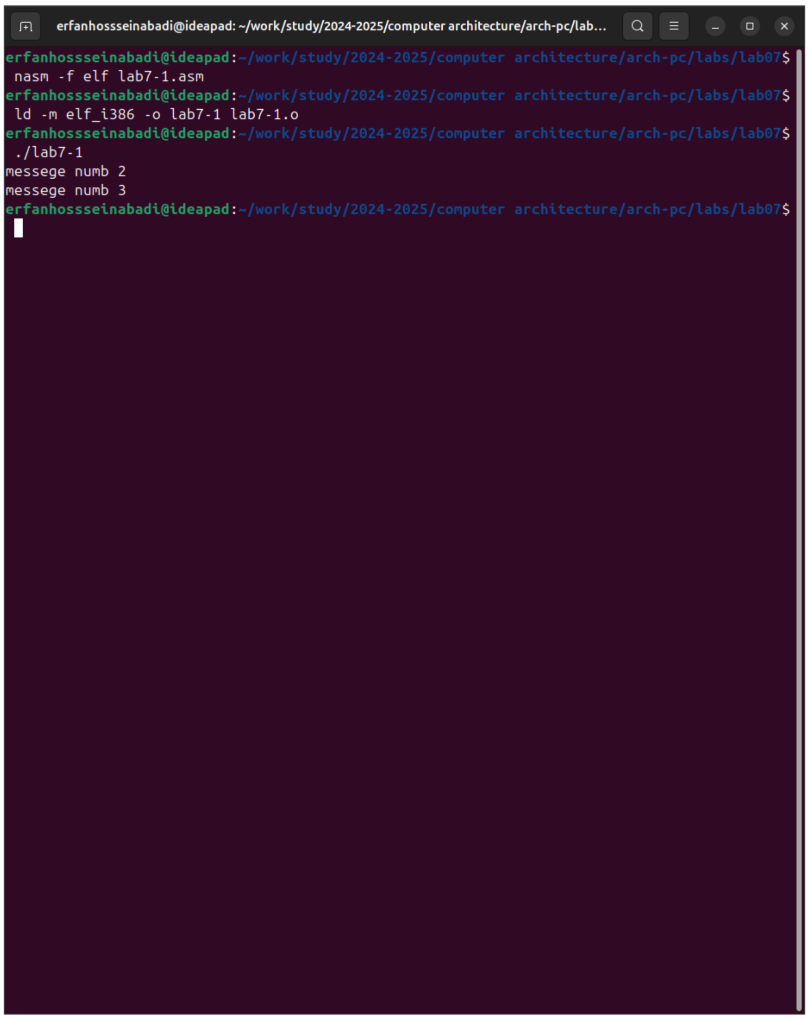
I copy the code from the listing into the file of the future program. (Fig. -fig. 2).


*~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07/la...
File Edit Search View Document Help

```
%include 'in_out.asm'

SECTION .data
msg1: DB 'messege numb 1', 0
msg2: DB 'messege numb 2', 0
msg3: DB 'messege numb 3', 0

SECTION .text
GLOBAL _start
_start:

jmp _label2

_label1:
mov eax, msg1
call sprintLF

_label2:
mov eax, msg2
call sprintLF

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Fig. 2: saving program

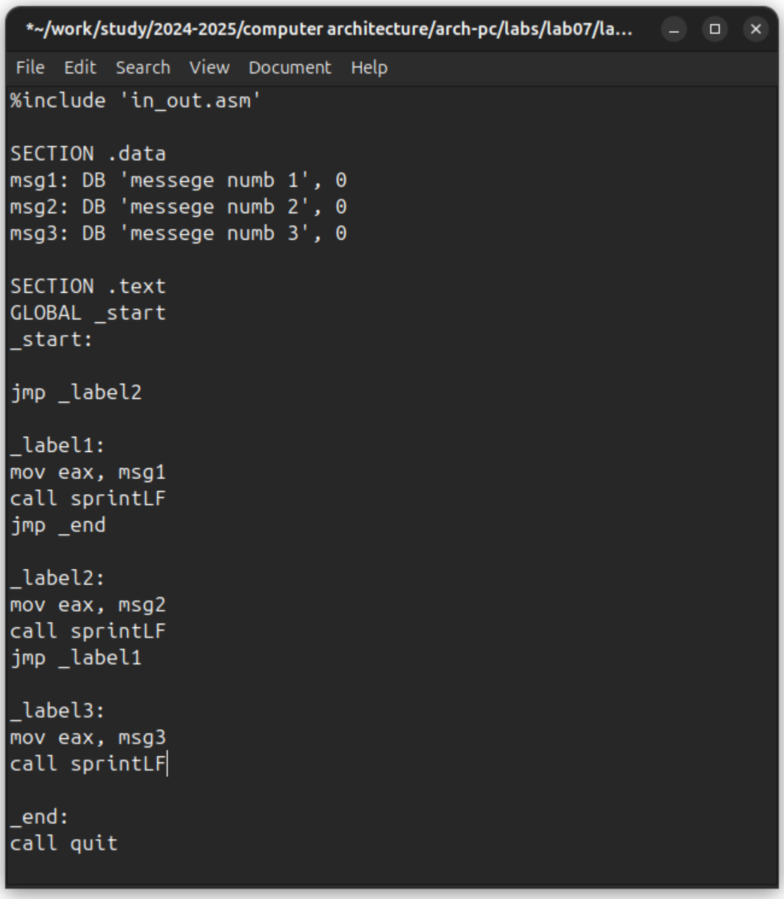
When launching the program, I made sure that the unconditional jump really changes the order of execution of instructions (Fig. -fig. 3).



```
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 nasm -f elf lab7-1.asm
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 ld -m elf_i386 -o lab7-1 lab7-1.o
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 ./lab7-1
messege numb 2
messege numb 3
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
```

Fig. 3: lunch program

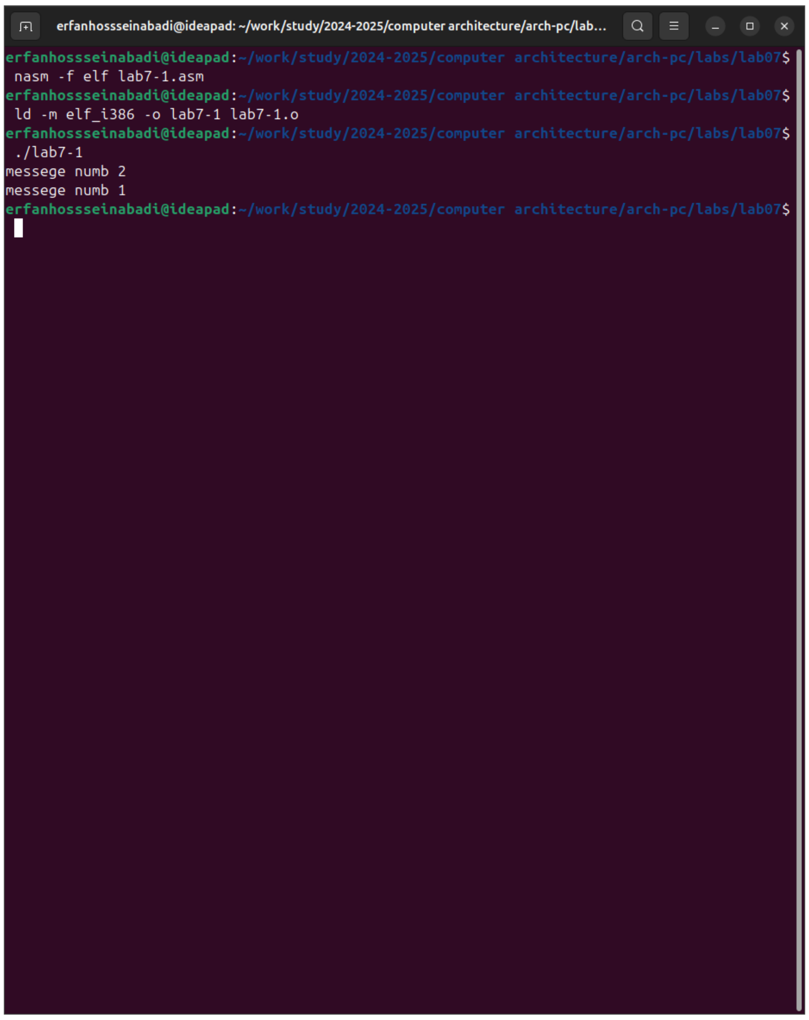
I change the program so that the order of execution of functions changes (Fig. -fig. 4).


*~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07/la...
File Edit Search View Document Help

```
%include 'in_out.asm'

SECTION .data
msg1: DB 'messege numb 1', 0
msg2: DB 'messege numb 2', 0
msg3: DB 'messege numb 3', 0

SECTION .text
GLOBAL _start
_start:

jmp _label2

_label1:
mov eax, msg1
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Fig. 4: change program

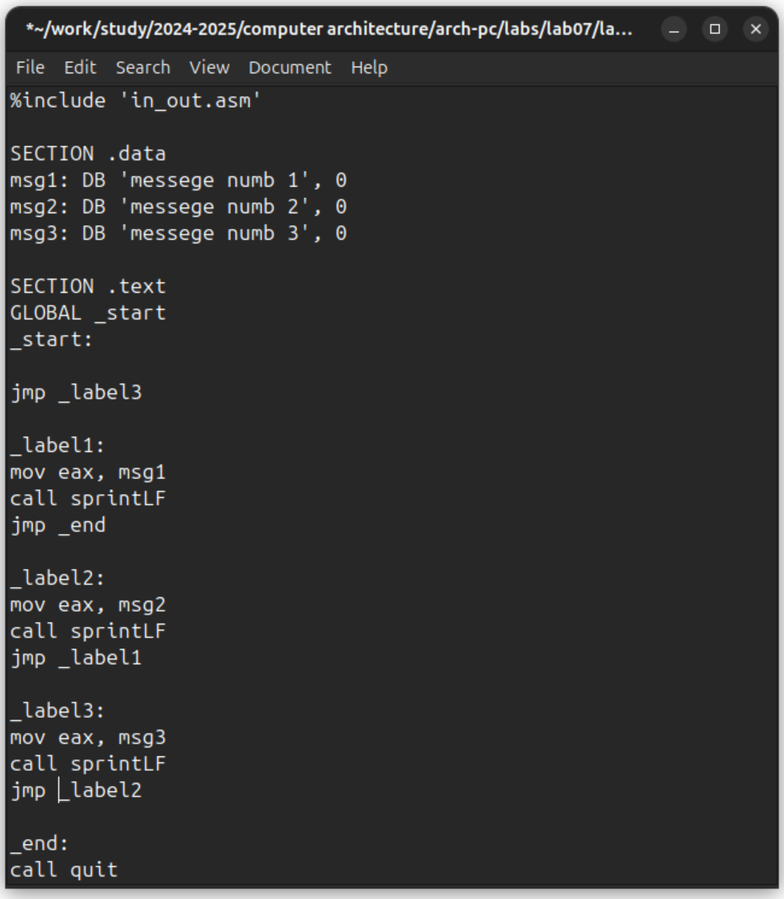
I launch the program and check that the applied changes are correct (Fig. -fig. 5).



```
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 nasm -f elf lab7-1.asm
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 ld -m elf_i386 -o lab7-1 lab7-1.o
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 ./lab7-1
messege numb 2
messege numb 1
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
```

Fig. 5: run the new program

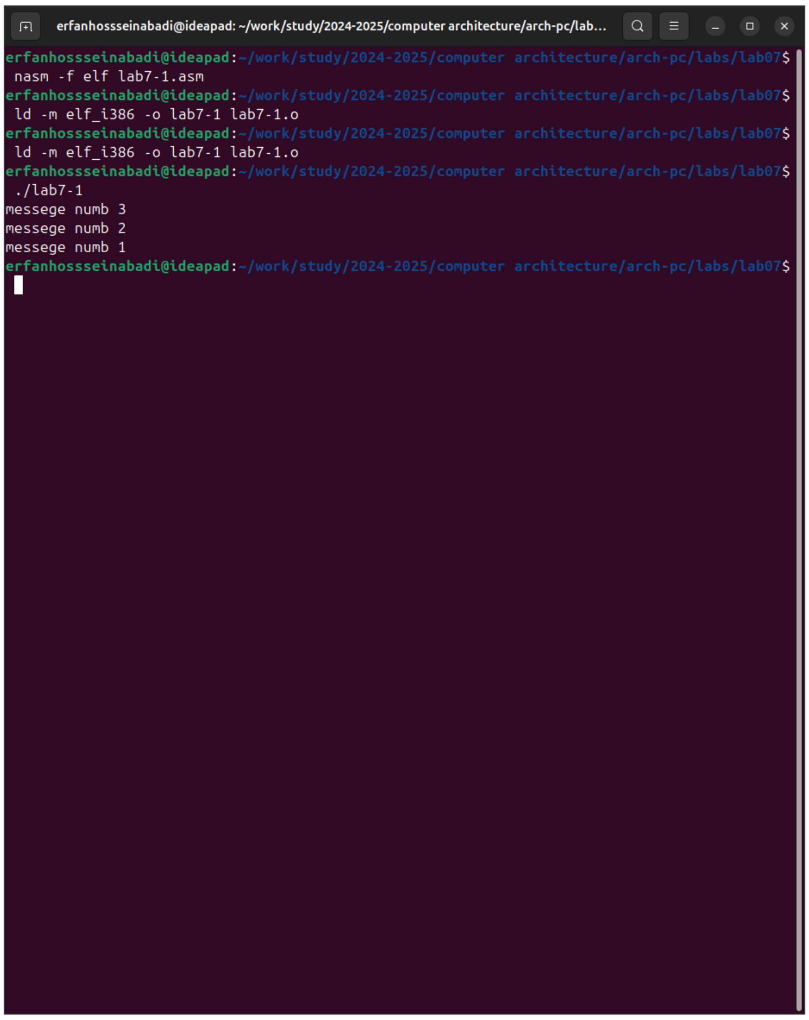
Now I change the text of the program so that all three messages are displayed in reverse order (Fig. -fig. 6).


*~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07/la...
File Edit Search View Document Help

```
%include 'in_out.asm'

SECTION .data
msg1: DB 'messege numb 1', 0
msg2: DB 'messege numb 2', 0
msg3: DB 'messege numb 3', 0

SECTION .text
GLOBAL _start
_start:

jmp _label3

_label1:
mov eax, msg1
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF
jmp _label2

_end:
call quit
```

Fig. 6: change program

The work is done correctly, the program displays messages in the order I need (Fig. -fig. 7).

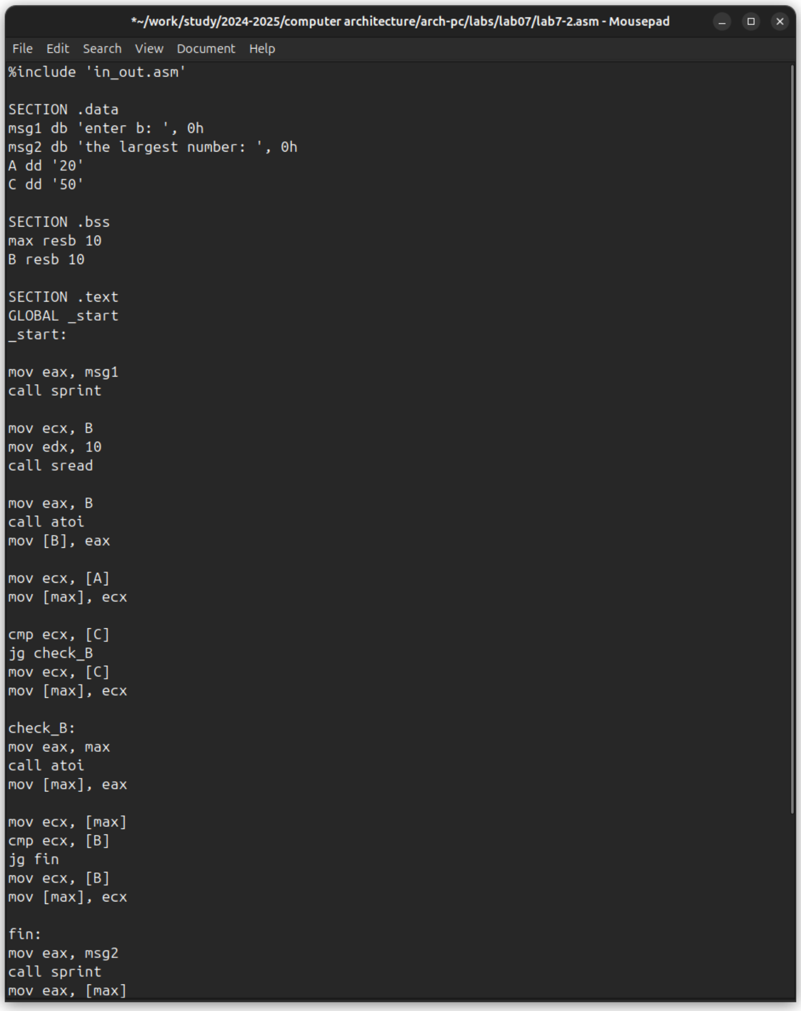


erfanhossseinabadi@ideapad: ~/work/study/2024-2025/computer architecture/arch-pc/lab...

```
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 nasm -f elf lab7-1.asm
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 ld -m elf_i386 -o lab7-1 lab7-1.o
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 ld -m elf_i386 -o lab7-1 lab7-1.o
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 ./lab7-1
messege numb 3
messege numb 2
messege numb 1
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
```

Fig. 7: checking for changes

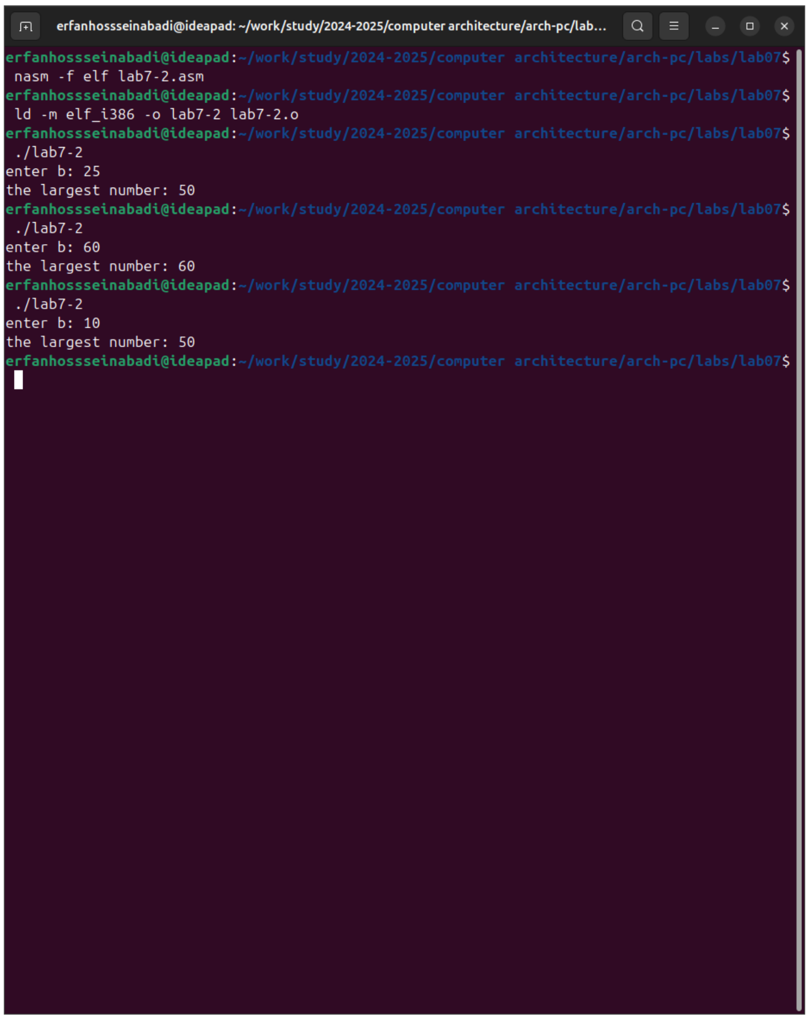
I create a new working file and paste into it the code from the following listing (Fig. -fig. 8).


*~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07/lab7-2.asm - Mousepad
File Edit Search View Document Help

```
%include 'in_out.asm'

SECTION .data
msg1 db 'enter b: ', 0h
msg2 db 'the largest number: ', 0h
A dd '20'
C dd '50'

SECTION .bss
max resb 10
B resb 10

SECTION .text
GLOBAL _start
_start:

mov eax, msg1
call sprint

mov ecx, B
mov edx, 10
call sread

mov eax, B
call atoi
mov [B], eax

mov ecx, [A]
mov [max], ecx

cmp ecx, [C]
jg check_B
mov ecx, [C]
mov [max], ecx

check_B:
mov eax, max
call atoi
mov [max], eax

mov ecx, [max]
cmp ecx, [B]
jg fin
mov ecx, [B]
mov [max], ecx

fin:
mov eax, msg2
call sprint
mov eax, [max]
```

Fig. 8: sacing new program

The program outputs the value of the variable with the maximum value, I check the operation of the program with different input data (Fig. -fig. 9).



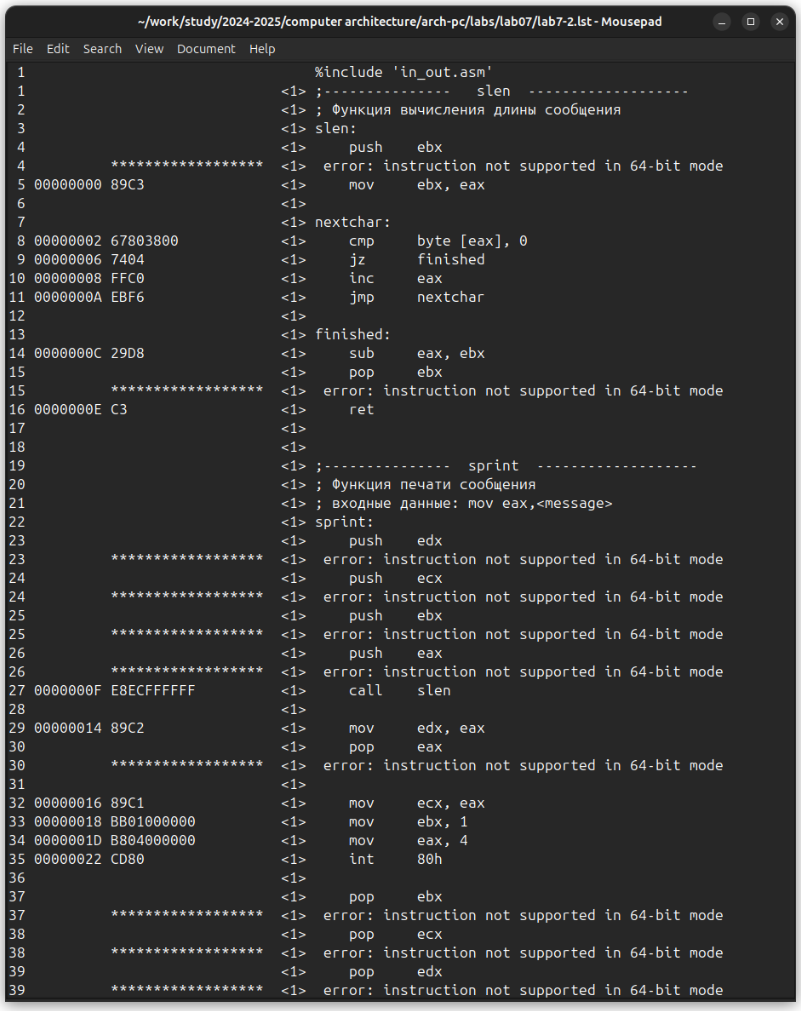
erfanhossseinabadi@ideapad: ~/work/study/2024-2025/computer architecture/arch-pc/lab...

```
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 nasm -f elf lab7-2.asm
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 ld -m elf_i386 -o lab7-2 lab7-2.o
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 ./lab7-2
enter b: 25
the largest number: 50
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 ./lab7-2
enter b: 60
the largest number: 60
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 ./lab7-2
enter b: 10
the largest number: 50
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
```

Fig. 9: Checking the program from the listing

#Study of the Listing File Structure

I create a listing file using the -l flag of the nasm command and open it using the mousepad text editor (Fig. -fig. 10).


~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07/lab7-2.lst - Mousepad

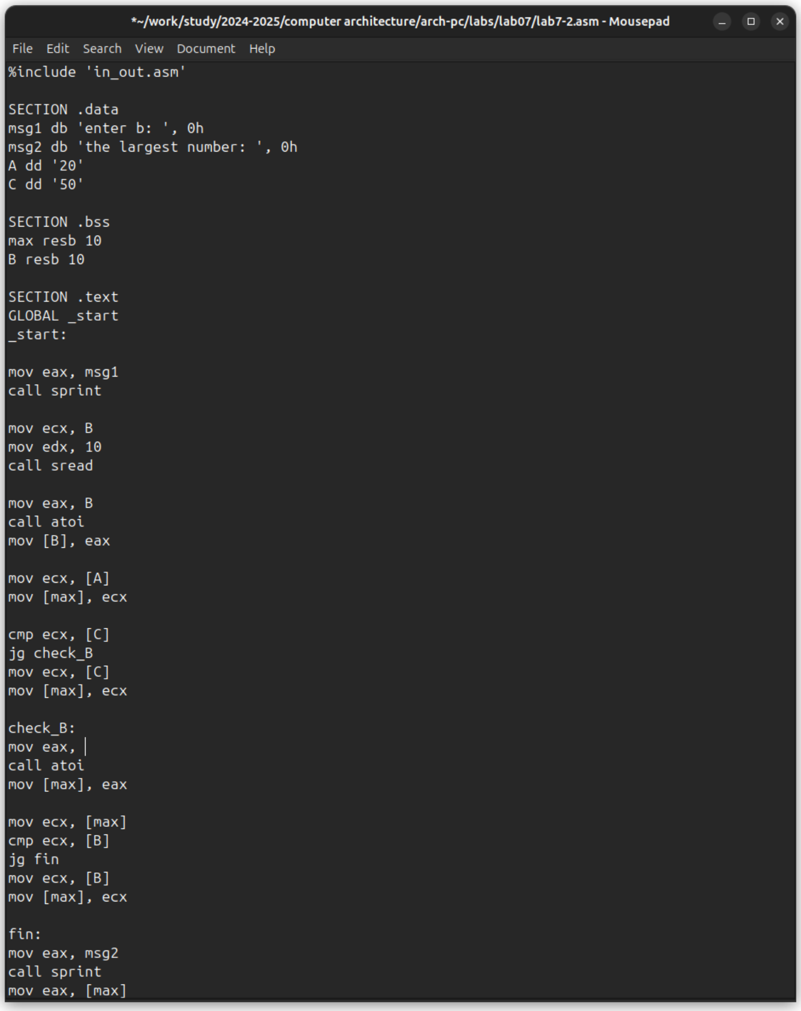
File Edit Search View Document Help

```
 1                                       %include 'in_out.asm'
 1                                   <1> ;--------------   slen  ------------------
 2                                   <1> ; Функция вычисления длины сообщения
 3                                   <1> slen:
 4                                   <1>     push    ebx
 4          ******************       <1>  error: instruction not supported in 64-bit mode
 5 00000000 89C3                     <1>     mov     ebx, eax
 6                                   <1>
 7                                   <1> nextchar:
 8 00000002 67803800                 <1>     cmp     byte [eax], 0
 9 00000006 7404                     <1>     jz      finished
10 00000008 FFC0                     <1>     inc     eax
11 0000000A EBF6                     <1>     jmp     nextchar
12                                   <1>
13                                   <1> finished:
14 0000000C 29D8                     <1>     sub     eax, ebx
15                                   <1>     pop     ebx
15          ******************       <1>  error: instruction not supported in 64-bit mode
16 0000000E C3                       <1>     ret
17                                   <1>
18                                   <1>
19                                   <1> ;--------------  sprint  ------------------
20                                   <1> ; Функция печати сообщения
21                                   <1> ; входные данные: mov eax,<message>
22                                   <1> sprint:
23                                   <1>     push    edx
23          ******************       <1>  error: instruction not supported in 64-bit mode
24                                   <1>     push    ecx
24          ******************       <1>  error: instruction not supported in 64-bit mode
25                                   <1>     push    ebx
25          ******************       <1>  error: instruction not supported in 64-bit mode
26                                   <1>     push    eax
26          ******************       <1>  error: instruction not supported in 64-bit mode
27 0000000F E8ECFFFFFF               <1>     call    slen
28                                   <1>
29 00000014 89C2                     <1>     mov     edx, eax
30                                   <1>     pop     eax
30          ******************       <1>  error: instruction not supported in 64-bit mode
31                                   <1>
32 00000016 89C1                     <1>     mov     ecx, eax
33 00000018 BB01000000               <1>     mov     ebx, 1
34 0000001D B804000000               <1>     mov     eax, 4
35 00000022 CD80                     <1>     int     80h
36                                   <1>
37                                   <1>     pop     ebx
37          ******************       <1>  error: instruction not supported in 64-bit mode
38                                   <1>     pop     ecx
38          ******************       <1>  error: instruction not supported in 64-bit mode
39                                   <1>     pop     edx
39          ******************       <1>  error: instruction not supported in 64-bit mode
```

Fig. 10: checking list file

The first value in the listing file is the line number, and it may not coincide with the line number of the original file. The second occurrence is the address, the offset of the machine code relative to the beginning of the current segment, then the machine code itself goes directly, and the line is concluded by the source text of the program with comments.

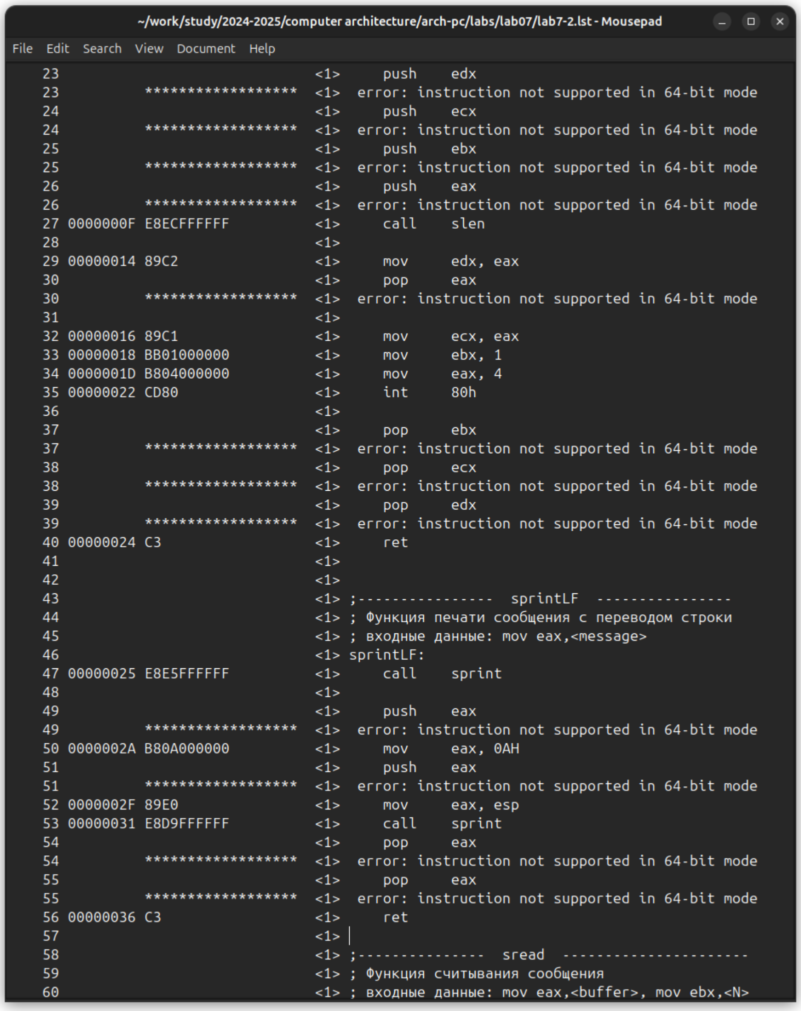
I delete one operand from a random instruction to check the behavior of the listing file in the future (Fig. -fig. 11).


*~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07/lab7-2.asm - Mousepad
File Edit Search View Document Help

```
%include 'in_out.asm'

SECTION .data
msg1 db 'enter b: ', 0h
msg2 db 'the largest number: ', 0h
A dd '20'
C dd '50'

SECTION .bss
max resb 10
B resb 10

SECTION .text
GLOBAL _start
_start:

mov eax, msg1
call sprint

mov ecx, B
mov edx, 10
call sread

mov eax, B
call atoi
mov [B], eax

mov ecx, [A]
mov [max], ecx

cmp ecx, [C]
jg check_B
mov ecx, [C]
mov [max], ecx

check_B:
mov eax, 
call atoi
mov [max], eax

mov ecx, [max]
cmp ecx, [B]
jg fin
mov ecx, [B]
mov [max], ecx

fin:
mov eax, msg2
call sprint
mov eax, [max]
```

Fig. 11: Removing an operand from a program

The new listing file shows the error that occurred when attempting to compile the file. No output files other than the listing file are created. (Fig. -fig. 12).

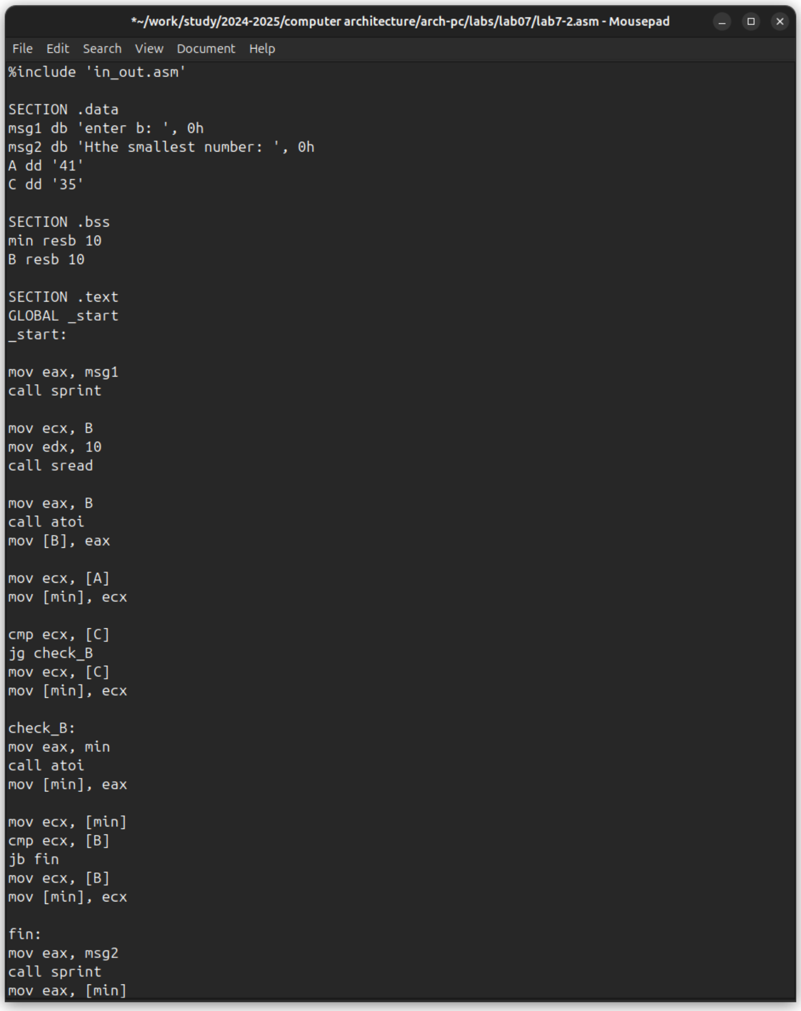

~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07/lab7-2.lst - Mousepad
File Edit Search View Document Help

```
    23                                  <1>     push    edx
    23          ******************      <1>  error: instruction not supported in 64-bit mode
    24                                  <1>     push    ecx
    24          ******************      <1>  error: instruction not supported in 64-bit mode
    25                                  <1>     push    ebx
    25          ******************      <1>  error: instruction not supported in 64-bit mode
    26                                  <1>     push    eax
    26          ******************      <1>  error: instruction not supported in 64-bit mode
    27 0000000F E8ECFFFFFF              <1>     call    slen
    28                                  <1>
    29 00000014 89C2                    <1>     mov     edx, eax
    30                                  <1>     pop     eax
    30          ******************      <1>  error: instruction not supported in 64-bit mode
    31                                  <1>
    32 00000016 89C1                    <1>     mov     ecx, eax
    33 00000018 BB01000000              <1>     mov     ebx, 1
    34 0000001D B804000000              <1>     mov     eax, 4
    35 00000022 CD80                    <1>     int     80h
    36                                  <1>
    37                                  <1>     pop     ebx
    37          ******************      <1>  error: instruction not supported in 64-bit mode
    38                                  <1>     pop     ecx
    38          ******************      <1>  error: instruction not supported in 64-bit mode
    39                                  <1>     pop     edx
    39          ******************      <1>  error: instruction not supported in 64-bit mode
    40 00000024 C3                      <1>     ret
    41                                  <1>
    42                                  <1>
    43                                  <1> ;---------------  sprintLF  ---------------
    44                                  <1> ; Функция печати сообщения с переводом строки
    45                                  <1> ; входные данные: mov eax,<message>
    46                                  <1> sprintLF:
    47 00000025 E8E5FFFFFF              <1>     call    sprint
    48                                  <1>
    49                                  <1>     push    eax
    49          ******************      <1>  error: instruction not supported in 64-bit mode
    50 0000002A B80A000000              <1>     mov     eax, 0AH
    51                                  <1>     push    eax
    51          ******************      <1>  error: instruction not supported in 64-bit mode
    52 0000002F 89E0                    <1>     mov     eax, esp
    53 00000031 E8D9FFFFFF              <1>     call    sprint
    54                                  <1>     pop     eax
    54          ******************      <1>  error: instruction not supported in 64-bit mode
    55                                  <1>     pop     eax
    55          ******************      <1>  error: instruction not supported in 64-bit mode
    56 00000036 C3                      <1>     ret
    57                                  <1>
    58                                  <1> ;--------------  sread  ---------------------
    59                                  <1> ; Функция считывания сообщения
    60                                  <1> ; входные данные: mov eax,<buffer>, mov ebx,<N>
```

Fig. 12: View error in listing file

Tasks for Independent Work

I sincerely do not understand what option I should have received during the 7th laboratory work, so I will use my option - the ninth - from the previous laboratory work. I return the operand to the function in the program and change it so that it outputs the variable with the smallest value (Fig. -fig. 13).



```
%include 'in_out.asm'

SECTION .data
msg1 db 'enter b: ', 0h
msg2 db 'Hthe smallest number: ', 0h
A dd '41'
C dd '35'

SECTION .bss
min resb 10
B resb 10

SECTION .text
GLOBAL _start
_start:

mov eax, msg1
call sprint

mov ecx, B
mov edx, 10
call sread

mov eax, B
call atoi
mov [B], eax

mov ecx, [A]
mov [min], ecx

cmp ecx, [C]
jg check_B
mov ecx, [C]
mov [min], ecx

check_B:
mov eax, min
call atoi
mov [min], eax

mov ecx, [min]
cmp ecx, [B]
jb fin
mov ecx, [B]
mov [min], ecx

fin:
mov eax, msg2
call sprint
mov eax, [min]
```

Fig. 13: First independent work program

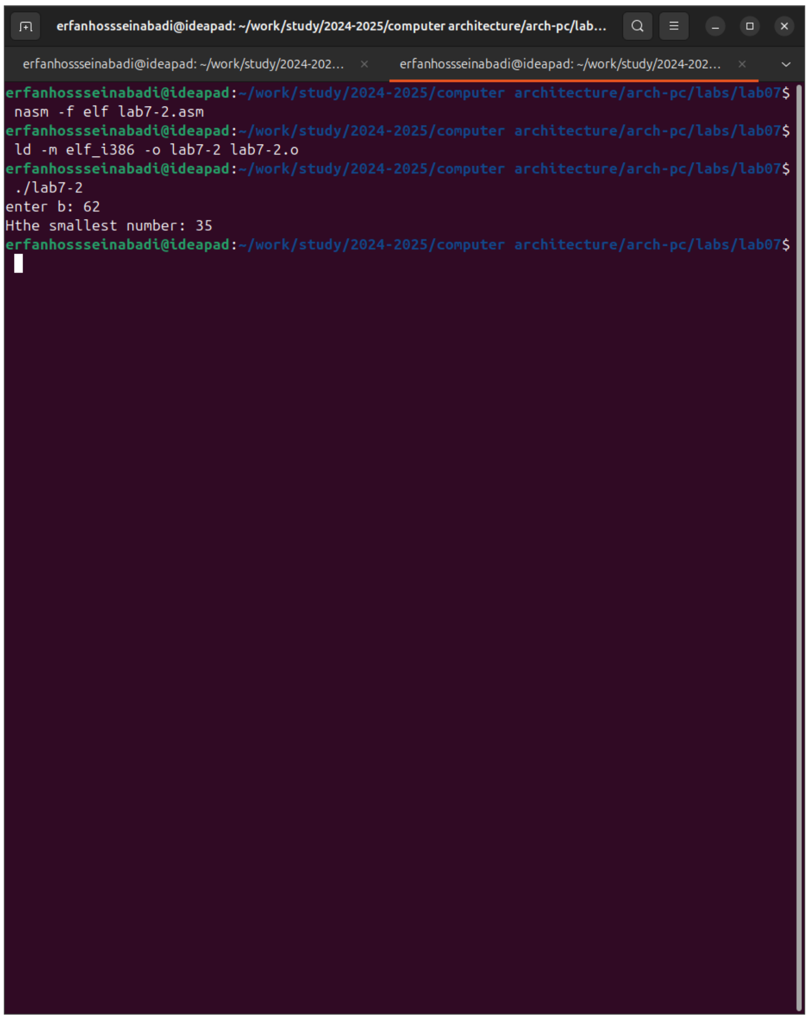
Code of the first program:

```
%%include 'in_out.asm'


SECTION .data
msg1 db 'enter b: ', 0h
msg2 db 'Hthe smallest number: ', 0h
A dd '41'
C dd '35'
```

```
SECTION .bss
min resb 10
B resb 10

SECTION .text
GLOBAL _start
_start:

mov eax, msg1
call sprint

mov ecx, B
mov edx, 10
call sread

mov eax, B
call atoi
mov [B], eax

mov ecx, [A]
mov [min], ecx

cmp ecx, [C]
jg check_B
mov ecx, [C]
mov [min], ecx

check_B:
mov eax, min
```

```
call atoi
mov [min], eax

mov ecx, [min]
cmp ecx, [B]
jb fin
mov ecx, [B]
mov [min], ecx

fin:
mov eax, msg2
call sprint
mov eax, [min]
call iprintLF
call quit
```

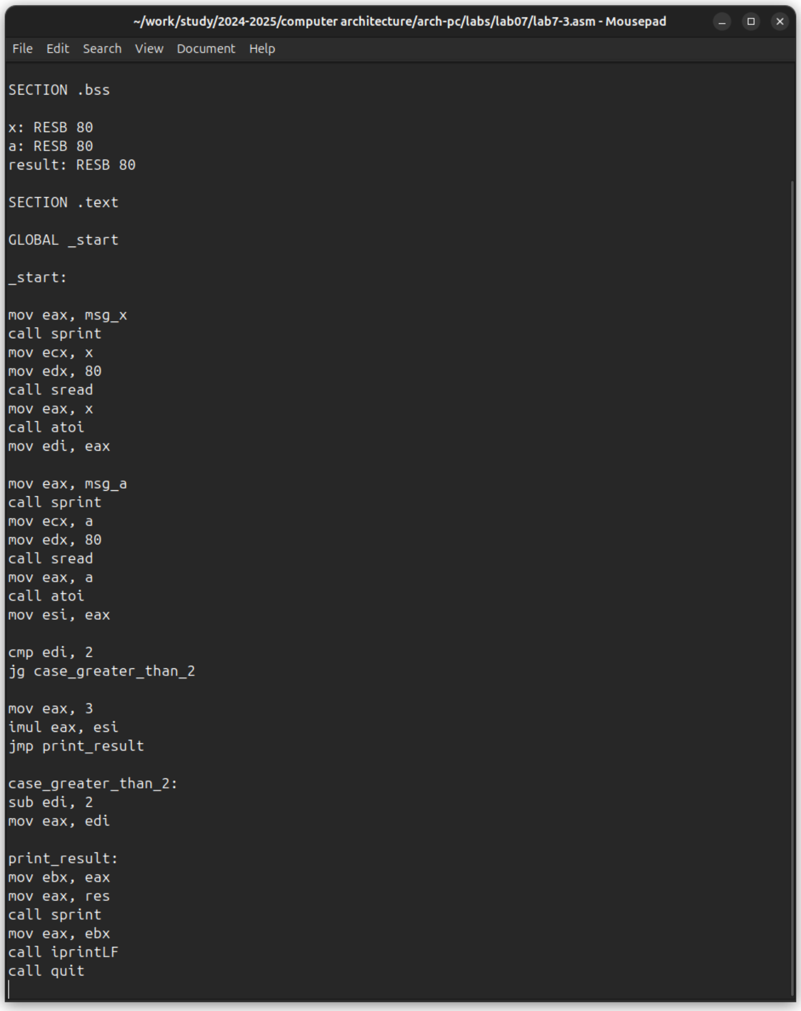
I check the correctness of writing the first program (Fig. -fig. 14).



```
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 nasm -f elf lab7-2.asm
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 ld -m elf_i386 -o lab7-2 lab7-2.o
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 ./lab7-2
enter b: 62
Hthe smallest number: 35
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
```

Fig. 14: check the first task

I write a program that will calculate the value of a given function according to my option for variables a and x entered from the keyboard (Fig. -fig. 15).


~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07/lab7-3.asm - Mousepad
File Edit Search View Document Help

```
SECTION .bss

x: RESB 80
a: RESB 80
result: RESB 80

SECTION .text

GLOBAL _start

_start:

mov eax, msg_x
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov edi, eax

mov eax, msg_a
call sprint
mov ecx, a
mov edx, 80
call sread
mov eax, a
call atoi
mov esi, eax

cmp edi, 2
jg case_greater_than_2

mov eax, 3
imul eax, esi
jmp print_result

case_greater_than_2:
sub edi, 2
mov eax, edi

print_result:
mov ebx, eax
mov eax, res
call sprint
mov eax, ebx
call iprintLF
call quit
```

Fig. 15: The second independent work program

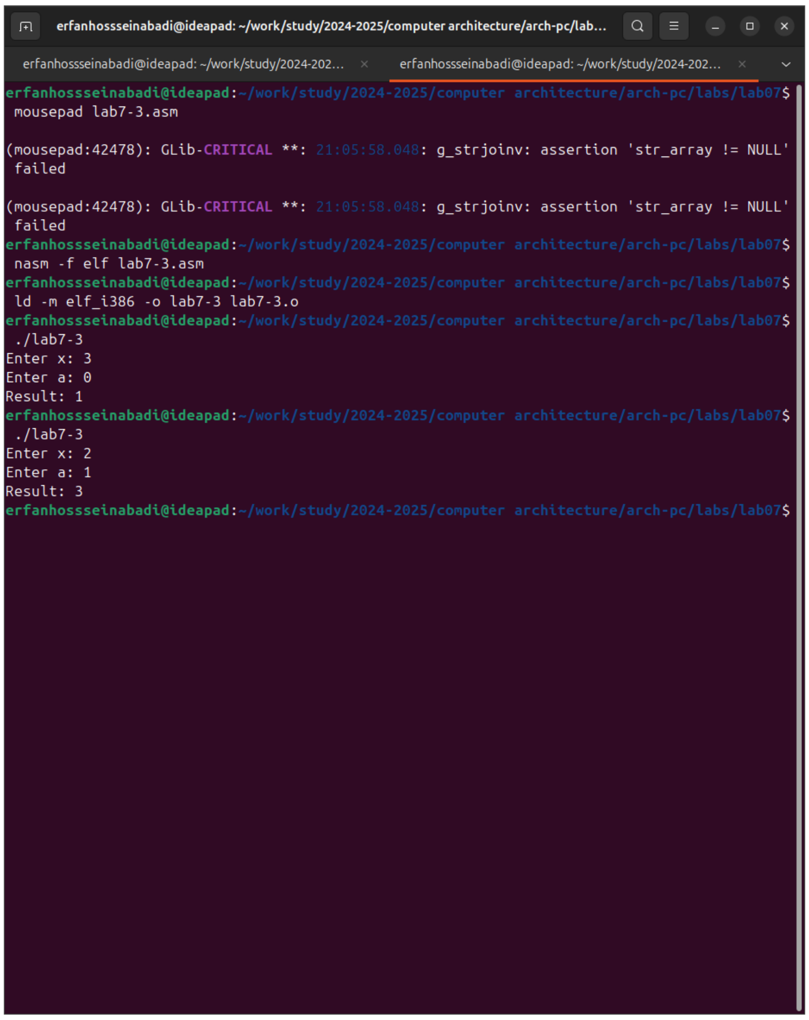
Code of the second program:

```
%include 'in_out.asm'

SECTION .data

msg_x: DB 'Enter x: ', 0
msg_a: DB 'Enter a: ', 0
res: DB 'Result: ', 0
```

```
SECTION .bss

x: RESB 80
a: RESB 80
result: RESB 80

SECTION .text

GLOBAL _start

_start:

mov eax, msg_x
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov edi, eax

mov eax, msg_a
call sprint
mov ecx, a
mov edx, 80
call sread
mov eax, a
call atoi
mov esi, eax
```

```
cmp edi, 2
jg case_greater_than_2

mov eax, 3
imul eax, esi
jmp print_result

case_greater_than_2:
sub edi, 2
mov eax, edi

print_result:
mov ebx, eax
mov eax, res
call sprint
mov eax, ebx
call iprintLF
call quit
```

I translate and link the file, run and check the operation of the program for various values of a and x (Fig. -fig. 16).



```
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 mousepad lab7-3.asm

(mousepad:42478): GLib-CRITICAL **: 21:05:58.048: g_strjoinv: assertion 'str_array != NULL'
 failed

(mousepad:42478): GLib-CRITICAL **: 21:05:58.048: g_strjoinv: assertion 'str_array != NULL'
 failed
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 nasm -f elf lab7-3.asm
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 ld -m elf_i386 -o lab7-3 lab7-3.o
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 ./lab7-3
Enter x: 3
Enter a: 0
Result: 1
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
 ./lab7-3
Enter x: 2
Enter a: 1
Result: 3
erfanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab07$
```

Fig. 16: check the second program

Conclusions

During the laboratory work, I studied the commands of conditional and unconditional jumps, and also acquired skills in writing programs using jumps, got acquainted with the purpose and structure of listing files.

References

1. Course at RUDN University
2. Laboratory work No. 7
3. Programming in NASM assembler language by Stolyarov A. V.

Note that some image file names ([image/1.png], etc.) are included, but as images were not supplied, they remain as placeholders. Also note that the links provided in the references section are in Russian, but I have given them English titles to reflect the content. I have made every effort to maintain the integrity of your original formatting.