

answer to labrotary work 6

Discipline: Computer Architecture

Ерфан Хосейнабади

Content

1	Purpose of the work	5
2	Task	6
3	Theoretical Introduction	7
4	Performing the lab work	8
4.1	Symbolic and numerical data in NASM	8
4.2	Performing arithmetic operations in NASM	17
4.3	Answers to security questions	23
4.4	Assignment for independent work	24
5	Conclusions	27
6	References	28

List of illustrations

4.1	go to lab difrectory	9
4.2	Saving a new program	10
4.3	Launching the original program	11
4.4	change the program	12
4.5	running the new program	13
4.6	second program	14
4.7	Output of the second program	15
4.8	output the changed program	16
4.9	Replacing the output function in the second program	17
4.10	third program	18
4.11	Launching the third program	19
4.12	change the third program	20
4.13	run the modified program	21
4.14	Program for calculating the variant	22
4.15	Running the program to calculate the variant	23
4.16	run the program	25

List of Tables

1 Purpose of the work

The purpose of this lab is to master the arithmetic instructions of the NASM assembly language.

2 Task

1. Symbolic and numerical data in NASM
2. Performing arithmetic operations in NASM
3. Completing assignments for independent work

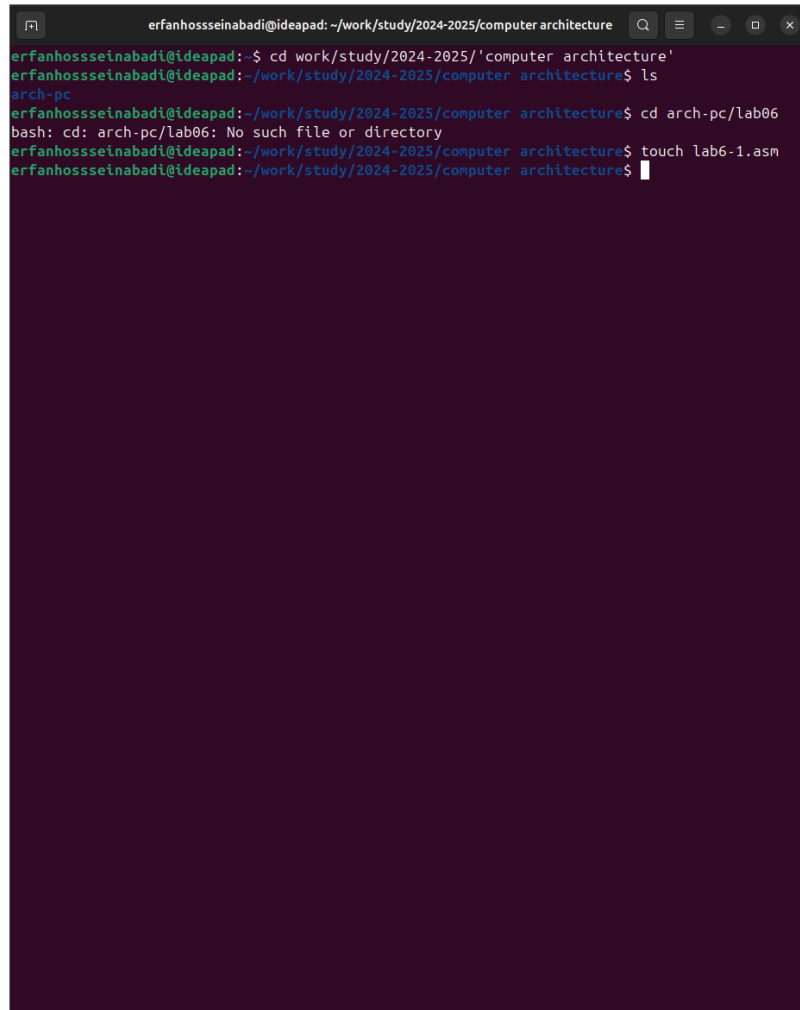
3 Theoretical Introduction

Most assembly language instructions require operands to be processed. The address of an operand provides the location where the data to be processed is stored. This can be data stored in a register or in a memory cell. - Register addressing - operands are stored in registers and the names of these registers are used in the command, for example: `mov ax, bx`. - Direct addressing - the operand value is specified directly in the command, for example: `mov ax, 2`. - Memory addressing - the operand specifies an address in memory. The command specifies a symbolic designation of the memory cell on whose contents the operation must be performed. Information is entered from the keyboard and displayed on the screen in symbolic form. This information is encoded according to the ASCII character code table. ASCII is an abbreviation for American Standard Code for Information Interchange. According to the ASCII standard, each character is encoded by one byte. There is no NASM instruction that outputs numbers (not in symbolic form). Therefore, for example, to output a number, you must first convert its digits into the ASCII codes of these digits and output these codes to the screen, not the number itself. If you output a number to the screen directly, the screen will perceive it not as a number, but as a sequence of ASCII characters - each byte of the number will be perceived as one ASCII character - and will output these characters to the screen. A similar situation occurs when entering data from the keyboard. The entered data will be characters, which will make it impossible to obtain a correct result when performing arithmetic operations on them. To solve this problem, it is necessary to convert ASCII characters to numbers and vice versa.

4 Performing the lab work

4.1 Symbolic and numerical data in NASM

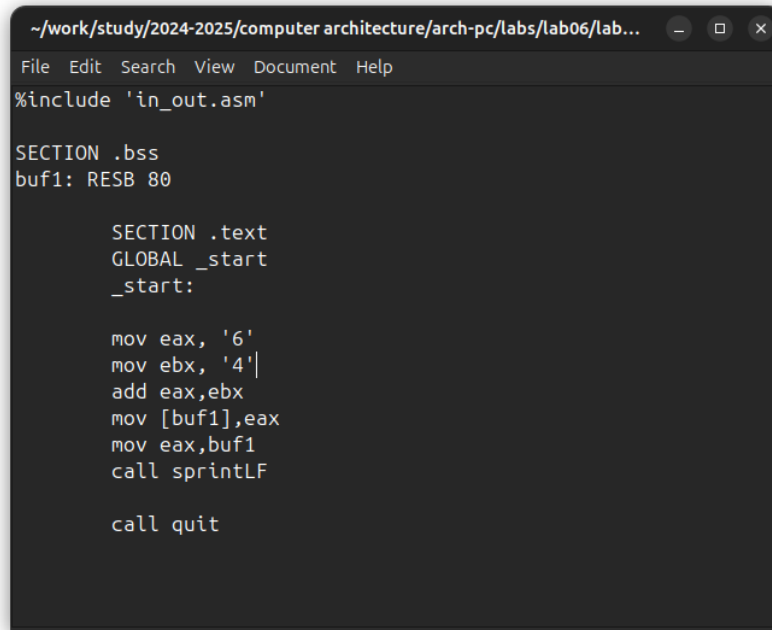
I go into lab work 6, create a file there (рис. -fig. 4.1).

A terminal window with a dark purple background and white text. The window title is 'erfanhosseinabadi@ideapad: ~/work/study/2024-2025/computer architecture'. The terminal shows the following commands and output:

```
erfanhosseinabadi@ideapad:~$ cd work/study/2024-2025/'computer architecture'  
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture$ ls  
arch-pc  
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture$ cd arch-pc/lab06  
bash: cd: arch-pc/lab06: No such file or directory  
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture$ touch lab6-1.asm  
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture$
```

Fig. 4.1: go to lab difrectory

In the created file I enter the program from the listing (рис. -fig. 4.2).

A screenshot of a text editor window with a dark background. The title bar at the top shows the file path: ~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06/lab... and standard window controls (minimize, maximize, close). The menu bar includes File, Edit, Search, View, Document, and Help. The code content is as follows:

```
%include 'in_out.asm'

SECTION .bss
buf1: RESB 80

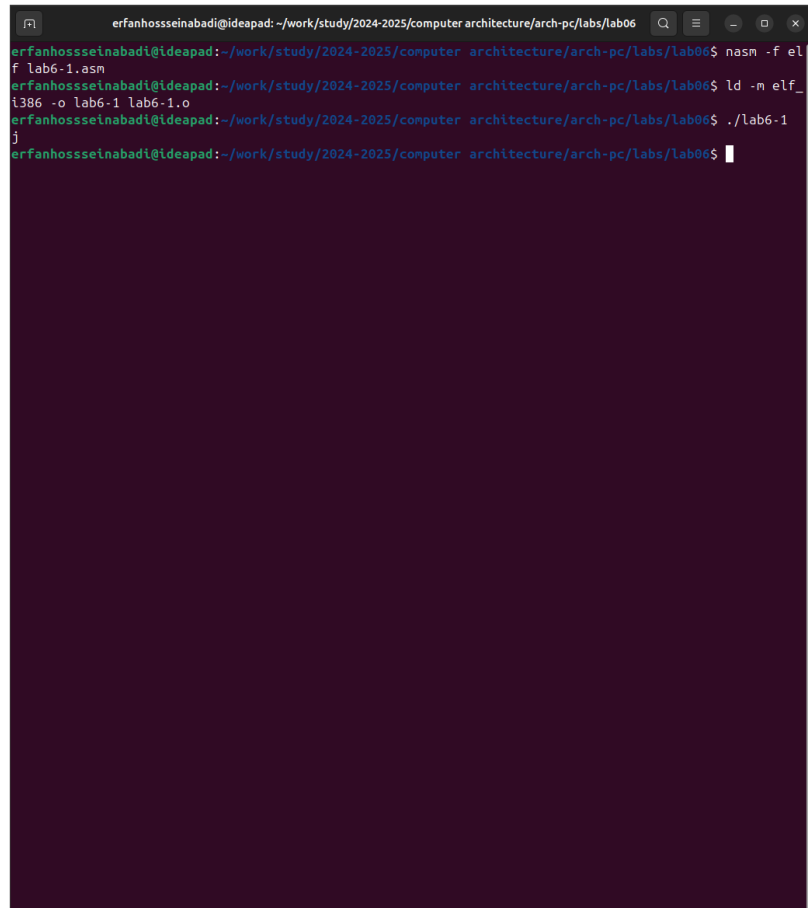
    SECTION .text
    GLOBAL _start
_start:

    mov eax, '6'
    mov ebx, '4'|
    add eax,ebx
    mov [buf1],eax
    mov eax,buf1
    call sprintLF

    call quit
```

Fig. 4.2: Saving a new program

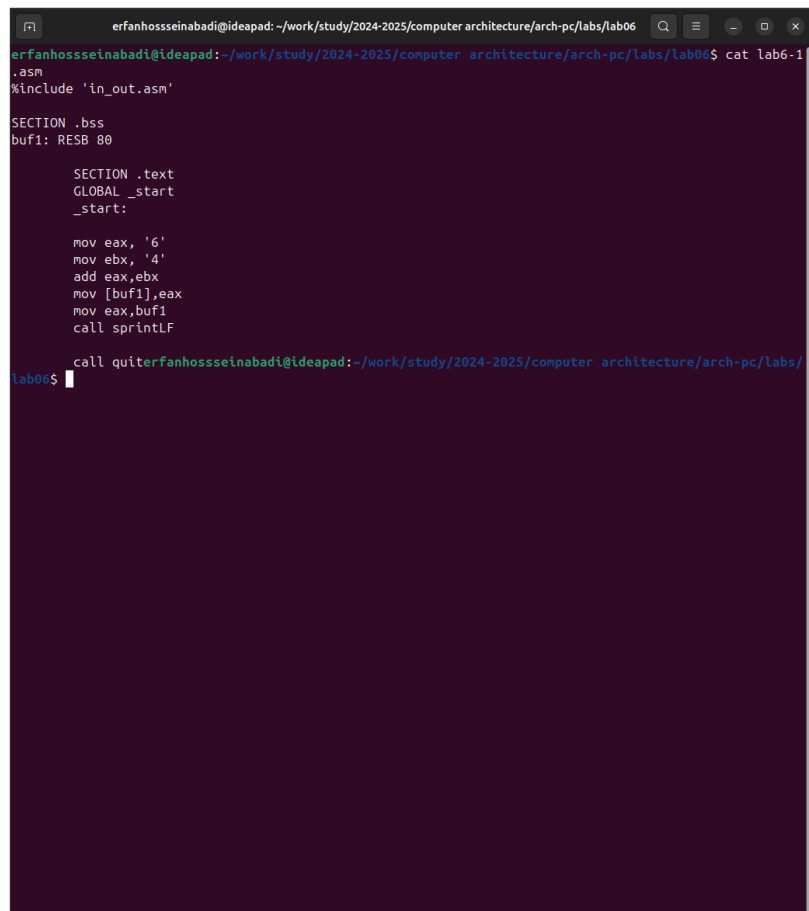
I create an executable file and run it, the program output differs from what was initially expected, because the character codes together give the character j according to the ASCII table. {#fig:003 width=70%}

A terminal window with a dark purple background. The title bar shows the user 'erfanhosseinabadi@ideapad' and the directory '~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06'. The terminal contains the following commands and output:

```
erFanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ nasm -f elf lab6-1.asm
erFanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ ld -n elf_ i386 -o lab6-1 lab6-1.o
erFanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ ./lab6-1
j
erFanhossseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$
```

Fig. 4.3: Launching the original program

I change the text of the original program by removing the quotes (рис. -fig. 4.4).

A terminal window with a dark purple background. The title bar shows the user 'erfanhosseinabadi@ideapad' and the directory '~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06'. The terminal content shows the command 'cat lab6-1.asm' being executed. The output is assembly code: '%include \'in_out.asm\'', 'SECTION .bss', 'buf1: RESB 80', 'SECTION .text', 'GLOBAL _start', '_start:', 'mov eax, \'6\'', 'mov ebx, \'4\'', 'add eax,ebx', 'mov [buf1],eax', 'mov eax,buf1', 'call sprintf', and 'call _quit'. The prompt 'lab06\$' is visible at the bottom left.

```
erfanhosseinabadi@ideapad: ~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ cat lab6-1.asm
%include 'in_out.asm'

SECTION .bss
buf1: RESB 80

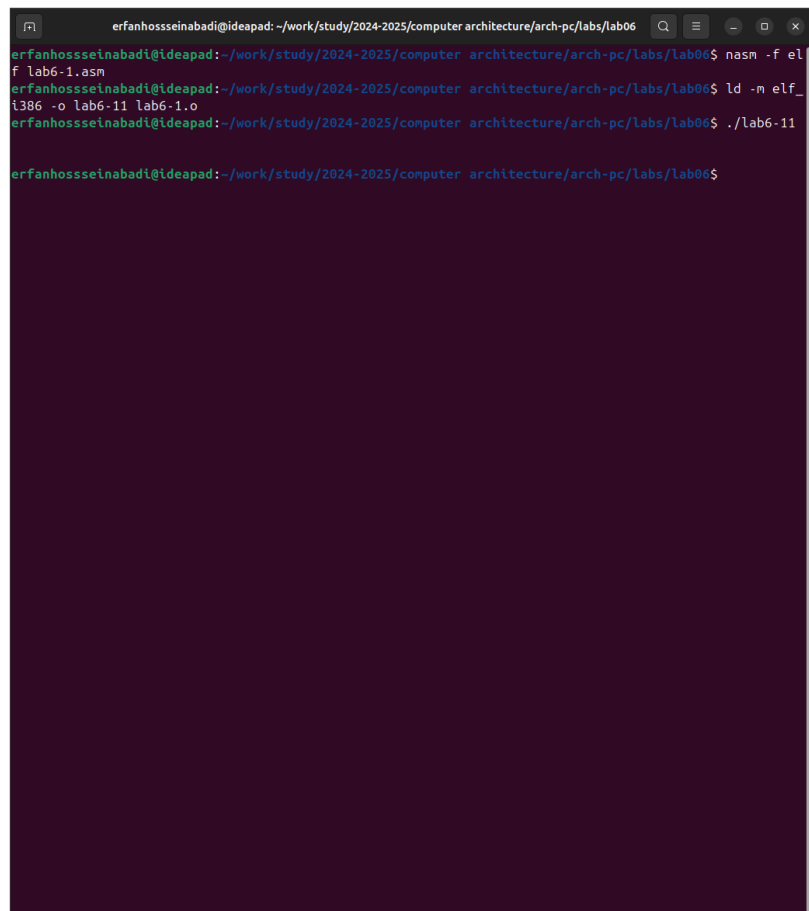
SECTION .text
GLOBAL _start
_start:

mov eax, '6'
mov ebx, '4'
add eax,ebx
mov [buf1],eax
mov eax,buf1
call sprintf

call _quit
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/
lab06$
```

Fig. 4.4: change the program

This time the program returned an empty line, this is because the symbol 10 means a new line. (рис. -fig. 4.5).

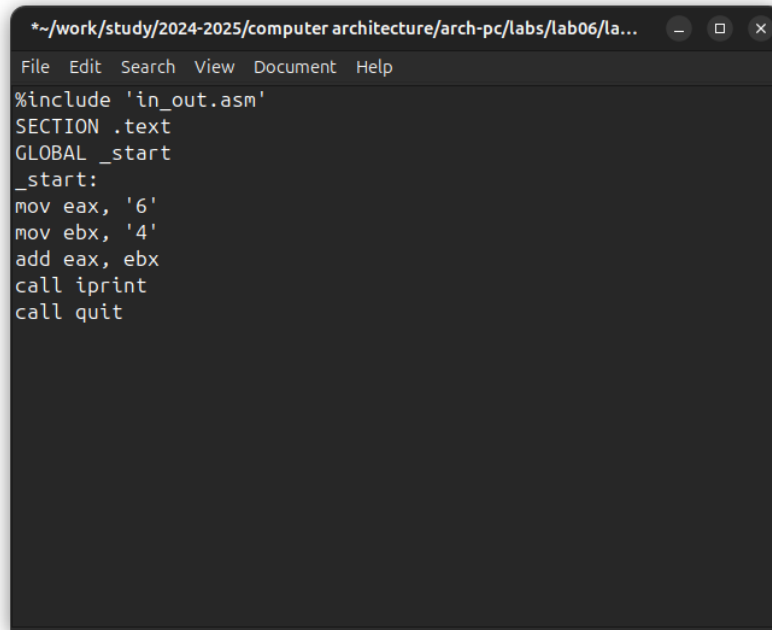
A terminal window with a dark purple background. The title bar shows the user 'erfanhosseinabadi@ideapad' and the path '~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06'. The terminal contains the following commands and output:

```
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ nasm -f elf_
f lab6-1.asm
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ ld -n elf_
i386 -o lab6-11 lab6-1.o
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ ./lab6-11

erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$
```

Fig. 4.5: running the new program

I create a new file for the future program and write the code from the listing into it (рис. -fig. 4.6).

A screenshot of a text editor window with a dark background. The title bar at the top shows the file path: `*~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06/la...`. The menu bar includes `File`, `Edit`, `Search`, `View`, `Document`, and `Help`. The code content is as follows:

```
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax, '6'
mov ebx, '4'
add eax, ebx
call iprint
call quit
```

Fig. 4.6: second program

I create an executable file and run it, now the result 106 is displayed, the program, as the first time, added up the character codes, but output the number itself, and not its symbol, thanks to replacing the output function with `iprintLF` (рис. -fig. 4.7).

```
erfanhosseinabadi@ideapad: ~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ touch lab6-2.asm
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ mousepad lab6-2.asm

(mousepad:15733): Glib-CRITICAL **: 16:15:35.687: g_strjoinv: assertion 'str_array != NULL' failed
(mousepad:15733): Glib-CRITICAL **: 16:15:35.687: g_strjoinv: assertion 'str_array != NULL' failed
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ nasm -f elf lab6-2.asm
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ ld -n elf_1386 -o lab6-2 lab6-2.o
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ ./lab6-2
106erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$
```

Fig. 4.7: Output of the second program

After removing the quotes in the program, I run it again and get the result I originally intended. (рис. -fig. 4.8).

```
erfanhosseinabadi@ideapad: ~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ mousepad l
ab6-2.asm

(mousepad:18942): Glib-CRITICAL **: 16:24:07.801: g_strjoinv: assertion 'str_array != NULL' failed
(mousepad:18942): Glib-CRITICAL **: 16:24:07.801: g_strjoinv: assertion 'str_array != NULL' failed
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ cat lab6-2
.asm
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax, 6
mov ebx, 4
add eax, ebx
call iprintLF
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ nasm -f
elf lab6-2.asm
$: command not found
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ nasm -f e
lf lab6-2.asm
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ ld -m elf_
i386 -o lab6-21 lab6-2.o
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ ./lab6-21
10
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$
```

Fig. 4.8: output the changed program

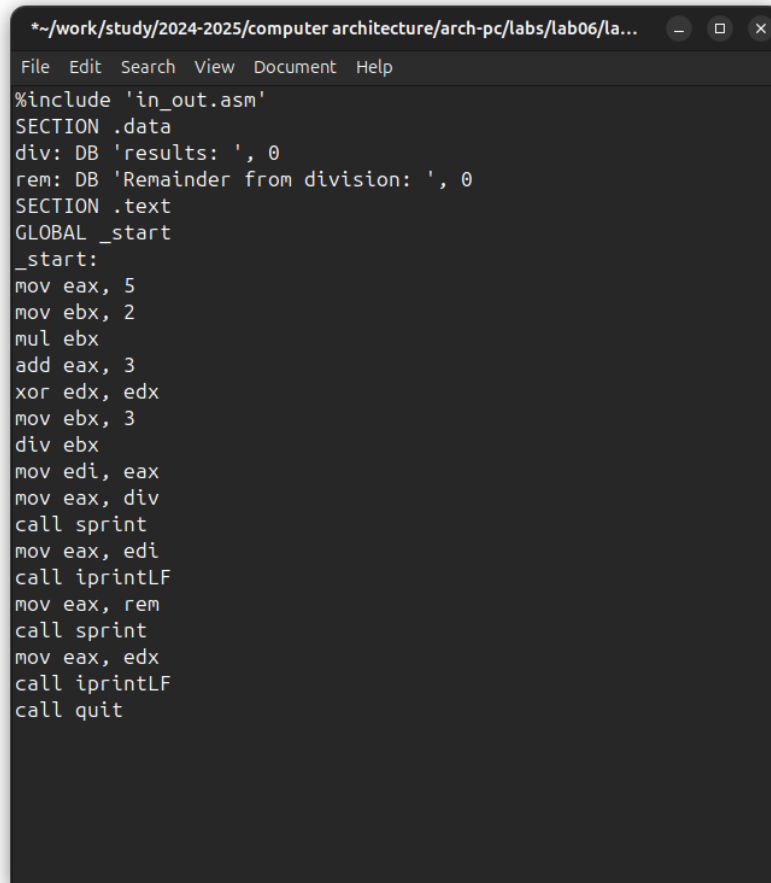
Replacing the output function with iprint gives me the same result but without the line break (рис. -fig. 4.9).


```
erfanhosseinabadi@ideapad: ~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ cat lab6-2
.asm
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax, 6
mov ebx, 4
add eax, ebx
call tprint
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ nasm -f e
lf lab6-2.asm
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ ld -m elf_
i386 -o lab6-22 lab6-2.o
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ ./lab6-22
10erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$
```

Fig. 4.9: Replacing the output function in the second program

4.2 Performing arithmetic operations in NASM

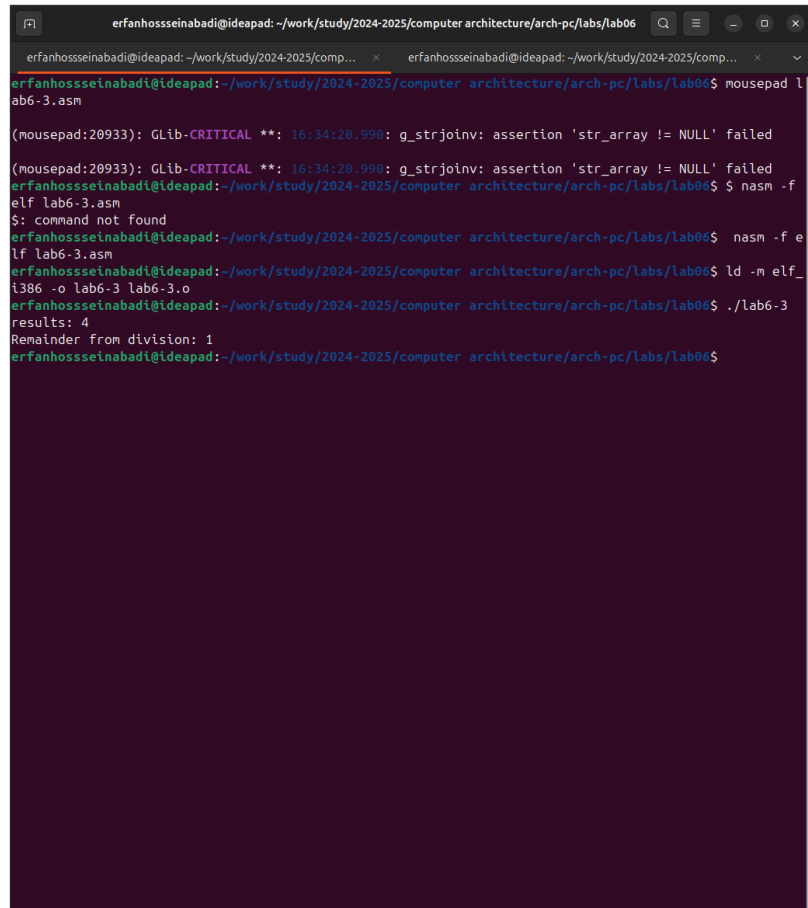
I create a new file and copy the contents of the listing into it(рис. -fig. 4.10).



```
*~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06/la...
File Edit Search View Document Help
%include 'in_out.asm'
SECTION .data
div: DB 'results: ', 0
rem: DB 'Remainder from division: ', 0
SECTION .text
GLOBAL _start
_start:
mov eax, 5
mov ebx, 2
mul ebx
add eax, 3
xor edx, edx
mov ebx, 3
div ebx
mov edi, eax
mov eax, div
call sprint
mov eax, edi
call iprintLF
mov eax, rem
call sprint
mov eax, edx
call iprintLF
call quit
```

Fig. 4.10: third program

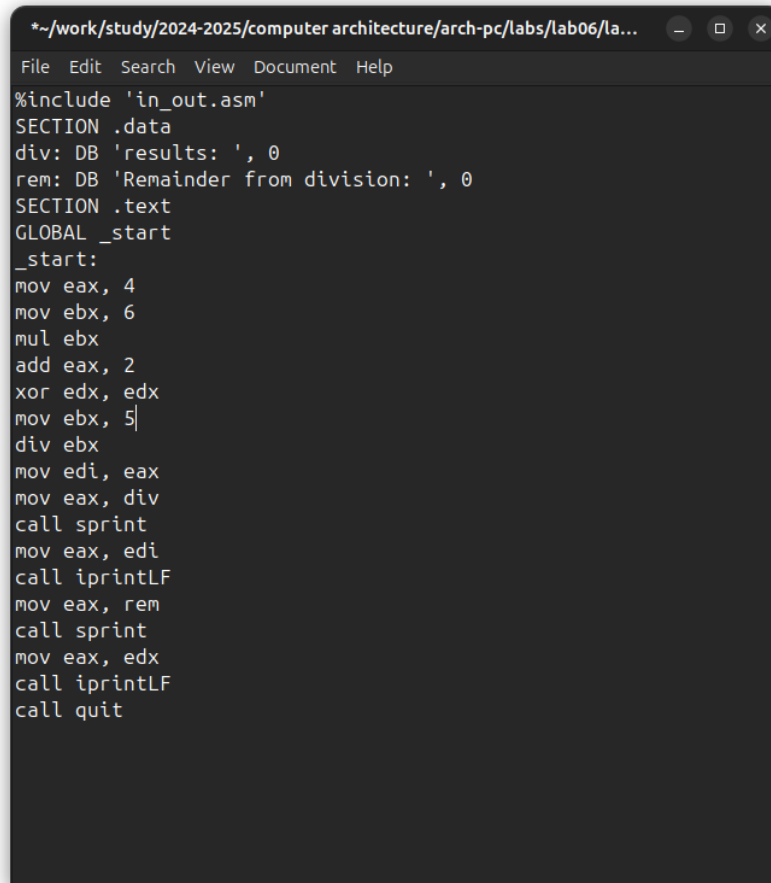
The program performs arithmetic calculations, the resulting expression and its remainder from division are output (рис. -fig. 4.11).



```
erfanhosseinabadi@ideapad: ~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ mousepad lab6-3.asm
(mousepad:20933): Glib-CRITICAL **: 16:34:20.998: g_strjoinv: assertion 'str_array != NULL' failed
(mousepad:20933): Glib-CRITICAL **: 16:34:20.998: g_strjoinv: assertion 'str_array != NULL' failed
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ nasm -f elf lab6-3.asm
$: command not found
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ nasm -f elf lab6-3.asm
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ ./lab6-3
results: 4
Remainder from division: 1
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$
```

Fig. 4.11: Launching the third program

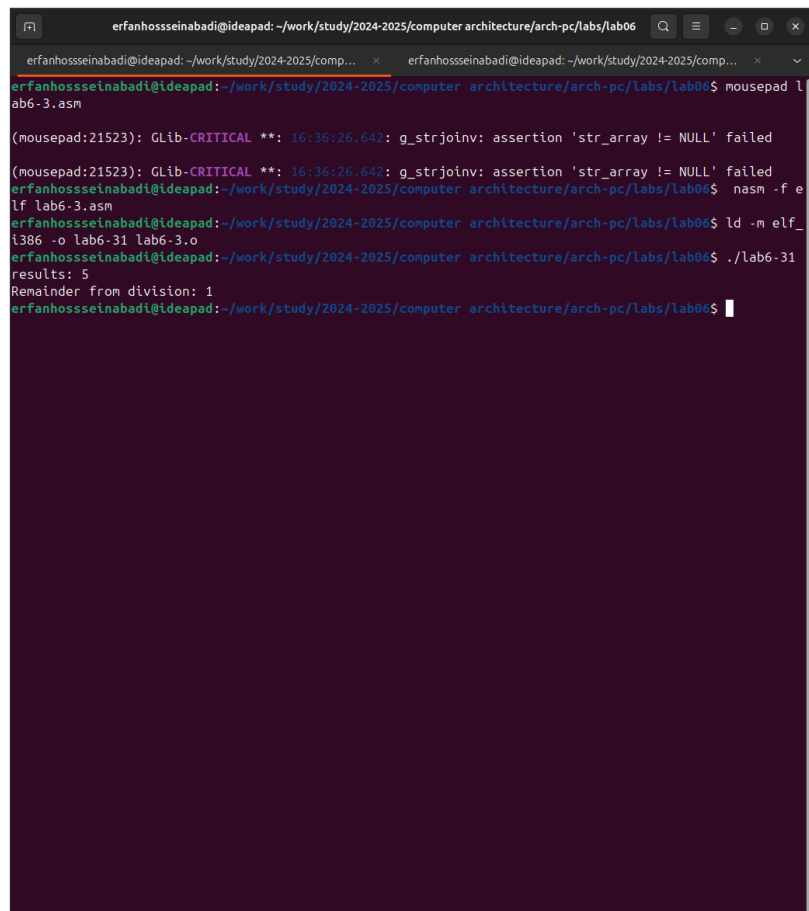
Replacing variables in the program for the expression $f(x) = (4*6+2)/5$ (рис. -fig. 4.12).



```
*~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06/la...
File Edit Search View Document Help
%include 'in_out.asm'
SECTION .data
div: DB 'results: ', 0
rem: DB 'Remainder from division: ', 0
SECTION .text
GLOBAL _start
_start:
mov eax, 4
mov ebx, 6
mul ebx
add eax, 2
xor edx, edx
mov ebx, 5
div ebx
mov edi, eax
mov eax, div
call sprint
mov eax, edi
call iprintLF
mov eax, rem
call sprint
mov eax, edx
call iprintLF
call quit
```

Fig. 4.12: change the third program

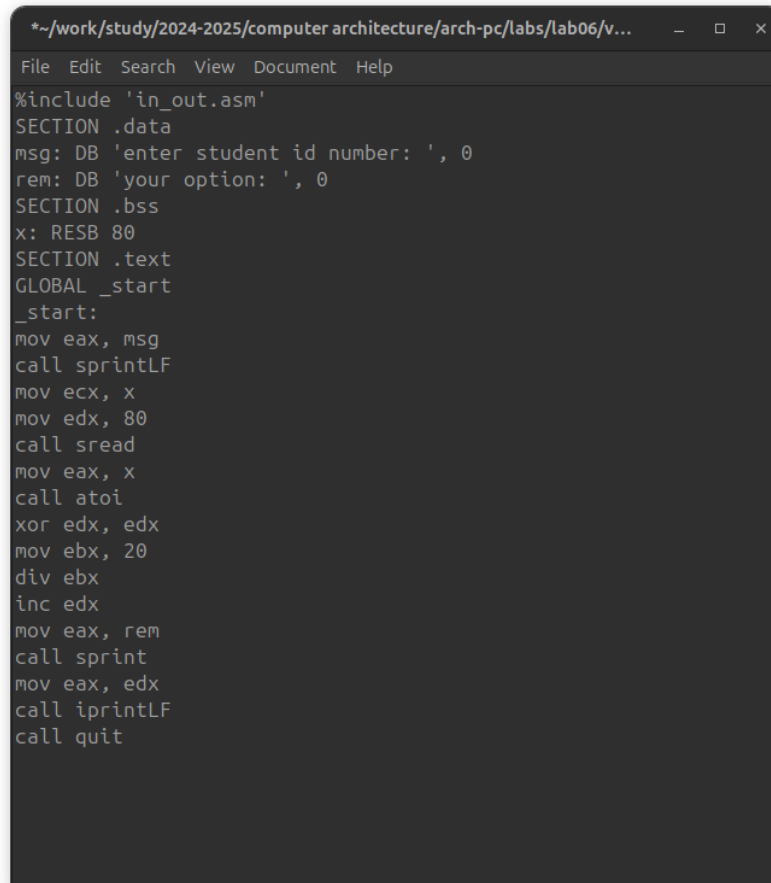
Running the program gives the correct result (рис. -fig. 4.13).



```
erfanhosseinabadi@ideapad: ~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ mousepad lab6-3.asm
(mousepad:21523): Glib-CRITICAL **: 16:36:26.642: g_strjoinv: assertion 'str_array != NULL' failed
(mousepad:21523): Glib-CRITICAL **: 16:36:26.642: g_strjoinv: assertion 'str_array != NULL' failed
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ nasm -f elf64 lab6-3.asm
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ ld -n elf_386 -o lab6-31 lab6-3.o
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ ./lab6-31
results: 5
Remainder from division: 1
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$
```

Fig. 4.13: run the modified program

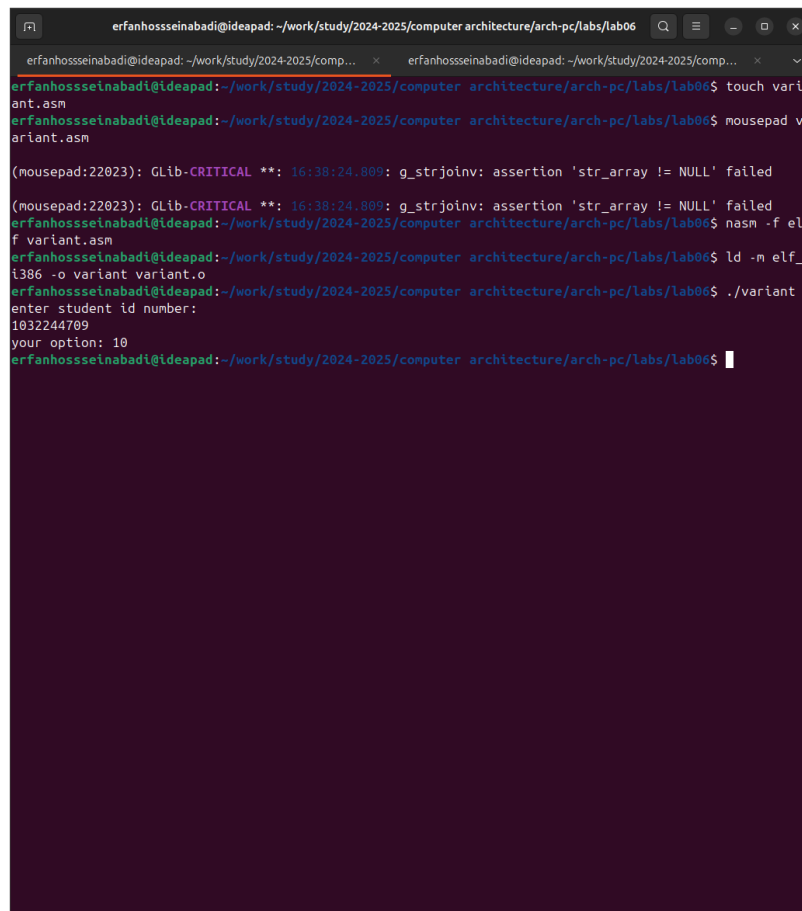
I create a new file and place the text from the listing (рис. -fig. 4.14).

A screenshot of a text editor window with a dark background. The title bar shows the file path: *~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06/v... The menu bar includes File, Edit, Search, View, Document, and Help. The code is written in assembly language and includes sections for data, bss, and text. It defines a message, a memory location 'x', and a routine '_start' that reads input, converts it to an integer, and prints the result.

```
*~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06/v...
File Edit Search View Document Help
%include 'in_out.asm'
SECTION .data
msg: DB 'enter student id number: ', 0
rem: DB 'your option: ', 0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintf
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
xor edx, edx
mov ebx, 20
div ebx
inc edx
mov eax, rem
call sprintf
mov eax, edx
call iprintLF
call quit
```

Fig. 4.14: Program for calculating the variant

After running the program and entering my student ID number, I received my version for further work. (рис. -fig. 4.15).

A terminal window with a dark purple background. The prompt is 'erfanhosseinabadi@ideapad: ~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06'. The user enters 'touch variant.asm'. The prompt changes to '(mousepad:22023): Glib-CRITICAL **: 16:38:24.809: g_strjoinv: assertion 'str_array != NULL' failed'. The user enters 'nasm -f elf variant.asm'. The prompt changes to 'ld -n elf_1386 -o variant variant.o'. The user enters './variant'. The program outputs 'enter student id number:' followed by '1032244709' and 'your option: 10'. The prompt returns to the shell.

```
erfanhosseinabadi@ideapad: ~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ touch variant.asm
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ nasm -f elf variant.asm
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ ld -n elf_1386 -o variant variant.o
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ ./variant
enter student id number:
1032244709
your option: 10
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$
```

Fig. 4.15: Running the program to calculate the variant

4.3 Answers to security questions

1. The following lines of code are responsible for displaying the message “Your option”:

```
mov eax,rem
call sprint
```

2. The instruction `mov ecx, x` is used to put the address of the input string `x` into the `ecx` register `mov edx, 80` - writes the length of the input string into the `edx` register `call sread` - calls a subroutine from an external file that provides input of a message from the keyboard.

3. call atoi is used to call a subroutine from an external file that converts the ascii code of a character into an integer and writes the result into the eax register.
4. The lines responsible for calculating the variant are:

```
xor edx,edx ; reset edx for correct work div
mov ebx,20 ; ebx = 20
div ebx ; eax = eax/20, edx - remainder from division
inc edx ; edx = edx + 1
```

5. When the div ebx instruction is executed, the remainder of the division is written to the edx register.
6. The inc edx instruction increases the value of the edx register by 1.
7. The following lines are responsible for displaying the results of calculations on the screen:

```
mov eax,edx
call iprintLF
```

4.4 Assignment for independent work

In accordance with the selected option, I will implement a program for calculating the function $f(x) = 10 + (31x - 5)$, checking on several variables shows the correct execution of the program (рис. -fig. 4.16).


```
erfanhosseinabadi@ideapad: ~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ cat lab6-4.asm
%include 'in_out.asm'
SECTION .data
msg: DB 'enter the value of x: ',0
rem: DB 'result: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov ebx, 31
mul ebx
sub eax, 5
add eax, 10
mov edi, eax
mov eax, rem
call sprint
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ nasm -f elf lab6-4.asm
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ ld -n elf_i386 -o lab6-4 lab6-4.o
erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ ./lab6-4
enter the value of x: 3
result: 98erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$ ./lab6-4
enter the value of x: 1
result: 36erfanhosseinabadi@ideapad:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab06$
```

Fig. 4.16: run the program

I am attaching the code of my program:

```
%include 'in_out.asm'

SECTION .data
msg: DB 'enter a number for x: ',0
rem: DB 'result: ',0

SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
```

```
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov ebx, 31
mul ebx
sub eax, 5
add eax, 10
mov edi, eax
mov eax, rem
call sprint
mov eax, edi
call iprint
```

5 Conclusions

During this lab I mastered the arithmetic instructions of the NASM assembly language.

6 References

1. Пример выполнения лабораторной работы
2. Курс на ТУИС
3. Лабораторная работа №6
4. Программирование на языке ассемблера NASM Столяров А. В.