

# Отчёт по лабораторной работе №6

## Дисциплина: архитектура компьютера

Чашемова Гульназик

### Содержание

1	Цель работы .....	1
2	Задание .....	1
3	Теоретическое введение.....	1
4	Выполнение лабораторной работы.....	2
4.1	Символьные и численные данные в NASM.....	2
4.2	Выполнение арифметических операций в NASM.....	6
4.2.1	В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения $f(x) = (5 * 2 + 3)/3$	6
4.2.2	В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму: 8	
4.3	Ответы на вопросы по программе.....	9
5	Выполнение заданий для самостоятельной работы .....	10
6	Выводы.....	11
	Список литературы .....	11

### 1 Цель работы

Цель данной лабораторной работы - освоение арифметических инструкций языка ассемблера NASM.

### 2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Выполнение заданий для самостоятельной работы

### 3 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут

быть данные хранящиеся в регистре или в ячейке памяти. Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации. Существует три основных способа адресации:

- **Регистровая адресация** – операнды хранятся в регистрах и в команде используются имена этих регистров, например: mov ax,bx.
- **Непосредственная адресация** – значение операнда задается непосредственно в команде. Например: mov ax,2.
- **Адресация памяти** – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию. Например, определим переменную intg DD 3 – это означает, что задается область памяти размером 4 байта, адрес которой обозначен меткой intg. В таком случае, команда mov eax,[intg] копирует из памяти по адресу intg данные в регистр eax. В свою очередь команда mov [intg],eax запишет в память по адресу intg данные из регистра eax. Также рассмотрим команду mov eax,intg. В этом случае в регистр eax запишется адрес intg. Допустим, для intg выделена память начиная с ячейки с адресом 0x600144, тогда команда mov eax,intg аналогична команде mov eax,0x600144 – т.е. эта команда запишет в регистр eax число 0x600144.

## 4 Выполнение лабораторной работы

### 4.1 Символьные и численные данные в NASM

Для начала я создала каталог для программ лабораторной работы №6, потом перешла на этот каталог и создала файл lab6-1.asm (рис. 1).

```
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_арх-pc/labs$ cd lab06
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_арх-pc/labs/lab06$ touch lab6-1.asm
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_арх-pc/labs/lab06$
```

Рис. 1: Создания каталога и файла

Потом я зашла на Midnight Commander и скопировала in\_out.asm в каталог с файлом lab6-1.asm с помощью клавиши F5

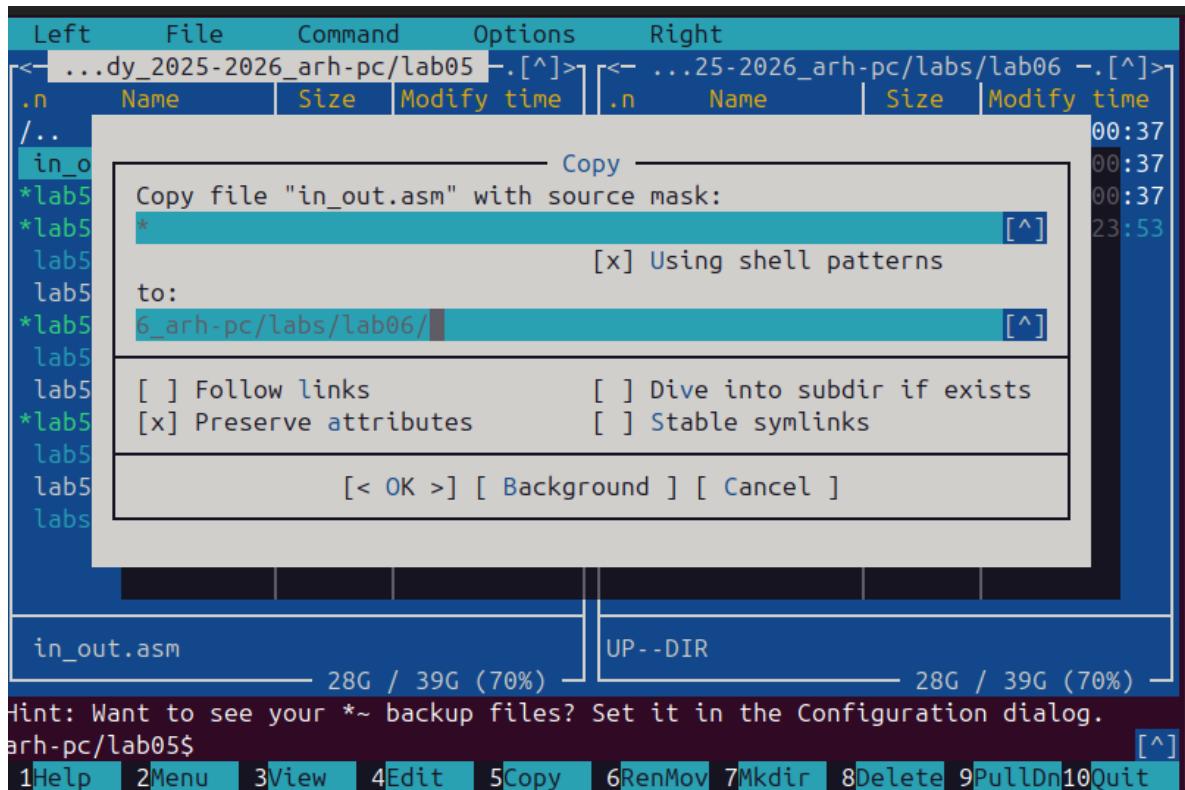


Рис. 2: Копированием файла на нужный каталог

Потом я открыла созданный файл lab6-1.asm и внес изменения в тексте файла.

```
vboxuser@gulsoltan:~/work/study/2025-20...
...epa/study_2025-2026_arh-pc/labs/lab06/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 sprintLF
    call quit
```

*Рис. 3: Изменения в тексте файла*

ПОТОМ В ТЕКСТЕ Я ИЗМЕНИЛА eax,'6' НА eax,6 А ebx,'4' НА ebx,4 .

The screenshot shows a terminal window with two parts. The top part is a code editor displaying assembly code. The bottom part is a terminal window showing the compilation and execution of the assembly code.

```
...pa компьютера/study_2025-2026_арх-pc/labs/lab06/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 sprintLF
    call quit
```

```
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab06$ nasm -f elf lab6-1.asm
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab06$ ./lab6-1

vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab06$
```

*Рис. 4: Изменения '6' и '4' на 6 и 4*

Все равно не вышло число 10 а вместо него вышло пустота. я зашла и посмотрела таблицу ASCII и там я увидела что символ числа 10 это пустота

Потом я создала новый файл в том же каталоге в котором был прошлый файл

```
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab06$ touch lab6-2.asm
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab06$
```

*Рис. 5: Создания нового файла lab6-2.asm*

После того как я создала файл я зашла на него и изменила текст файла

```
...итектура компьютера/study_2025-2026_arh-pc/labs/lab06/lab6-2.asm *
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
    mov eax,'6'
    mov ebx,'4'
    add eax,ebx
    call iprintLF
    call quit
```

Рис. 6: Изменения текста файла lab6-2.asm

Создала исполняемый файл и запустила его

```
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab06$ nasm -f elf lab6-2.asm
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab06$ ./lab6-2
106
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab06$
```

Рис. 7: Создание и запуск исполняемого файла

Потом заменила где eax '6' и eax '4' на ebx,6 и ebx,4

```
mc [vboxuser@gulsoltan]:~/work/study/2025-2026/Архитектура ...
...итектура компьютера/study_2025-2026_arh-pc/labs/lab06/lab6-2.asm
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
    mov eax,6
    mov ebx,4
    add eax,ebx
    call iprintLF
    call quit
'
```

Рис. 8: Изменения eax и ebx

После изменения я создала исполняемый файл и запустила его

```
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025  
-2026_arh-pc/labs/lab06$ nasm -f elf lab6-2.asm  
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025  
-2026_arh-pc/labs/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o  
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025  
-2026_arh-pc/labs/lab06$ ./lab6-2  
10  
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025  
-2026_arh-pc/labs/lab06$
```

Рис. 9: Создание и запуск исполняемого файла(измененного eax)

После этих изменений я получила цифру 10. Потом я изменила iprintLF на iprint

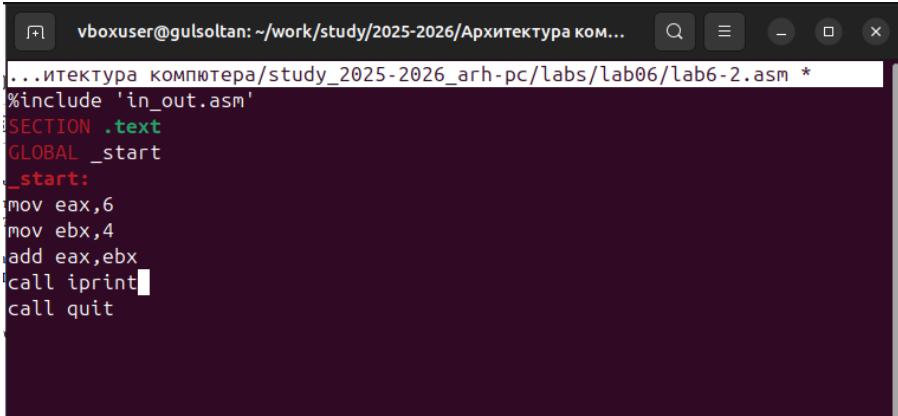


Рис. 10: Изменения iprintLF на iprint

После изменения я заново создала исполняемый файл и запустила его

```
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025  
-2026_arh-pc/labs/lab06$ nasm -f elf lab6-2.asm  
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025  
-2026_arh-pc/labs/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o  
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025  
-2026_arh-pc/labs/lab06$ ./lab6-2  
10  
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025  
-25  
-2026_arh-pc/labs/lab06$
```

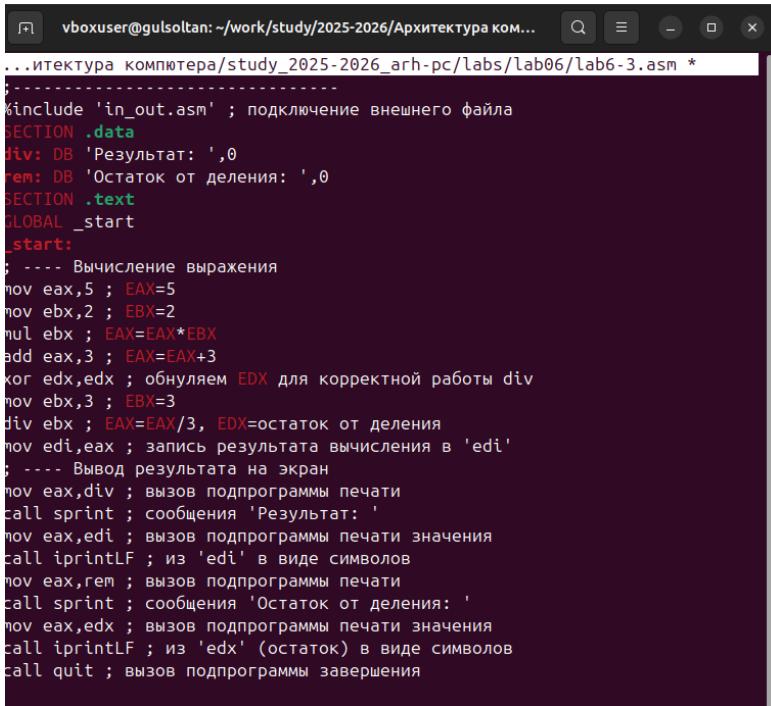
Рис. 11: Создание и запуск исполняемого файла(измененного iprintLF)

Таким образом, разница между iprintLF и iprint в NASM заключается в том, что iprintLF — это функция для печати целых чисел с переводом на новую строку, а iprint — для простой печати целых чисел без перевода на новую строку

## 4.2 Выполнение арифметических операций в NASM

### 4.2.1 В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения $f(x) = (5 * 2 + 3)/3$ .

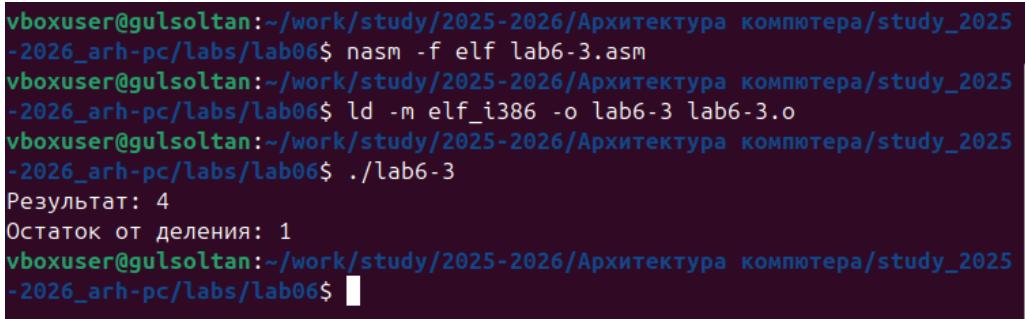
Для начала я создала файл lab6-3.asm с помощью touch потом внес изменения в текст файла



```
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab06/lab6-3.asm *  
...итектура компьютера/study_2025-2026_arh-pc/labs/lab06/lab6-3.asm *  
-----  
;include 'in_out.asm' ; подключение внешнего файла  
SECTION .data  
div: DB 'Результат: ',0  
rem: DB 'Остаток от деления: ',0  
SECTION .text  
GLOBAL _start  
.start:  
; ---- Вычисление выражения  
mov eax,5 ; EAX=5  
mov ebx,2 ; EBX=2  
mul ebx ; EAX=EAX*EBX  
add eax,3 ; EAX=EAX+3  
xor edx,edx ; обнуляем EDX для корректной работы div  
mov ebx,3 ; EBX=3  
div ebx ; EAX=EAX/3, EDX=остаток от деления  
mov edi,eax ; запись результата вычисления в 'edi'  
; ---- Вывод результата на экран  
mov eax,div ; вызов подпрограммы печати  
call sprint ; сообщения 'Результат: '  
mov eax,edi ; вызов подпрограммы печати значения  
call iprintLF ; из 'edi' в виде символов  
mov eax,rem ; вызов подпрограммы печати  
call sprint ; сообщения 'Остаток от деления: '  
mov eax,edx ; вызов подпрограммы печати значения  
call iprintLF ; из 'edx' (остаток) в виде символов  
call quit ; вызов подпрограммы завершения
```

Рис. 12: Создание файла и изменения текста lab6-3.asm

Потом создала исполняемый файл lab6-3.asm и запустила его



```
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab06$ nasm -f elf lab6-3.asm  
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o  
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab06$ ./lab6-3  
Результат: 4  
Остаток от деления: 1  
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab06$
```

Рис. 13: Создание исполняемого файла lab6-3.asm и запуск файла

Потом изменила текст программы для вычисления выражения  $f(x) = (4 * 6 + 2)/5$

```
...epa/study_2025-2026_arh-pc/labs/lab06/lab6-3.asm *
;-----
;%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax,4 ; EAX=4
mov ebx,6 ; EBX=6
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+2
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; EBX=5
div ebx ; EAX=EAX/5, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintLF ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения
]

```

Рис. 14: Изменения текста lab6-3.asm

Потом создала исполняемый файл lab6-3.asm и запустила его

```
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера$ nasm -f elf lab6-3.asm
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера$ ld -m elf_i386 -o lab6-3 lab6-3.o
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера$ ./lab6-3
Результат: 5
Остаток от деления: 1
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера$
```

Рис. 15: Создание исполняемого файла lab6-3.asm и запуск файла (с изменением)

#### 4.2.2 В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму:

Для начала я создала файл variant.asm с помощью команды touch и внес в него изменения

The screenshot shows a terminal window with the following content:

```
vboxuser@gulsoltan:~/work/study/2025-2026_arh-pc/labs/lab06/variant.asm *  
-----  
; Программа вычисления варианта  
-----  
%include 'in_out.asm'  
SECTION .data  
msg: DB 'Введите № студенческого билета: ',0  
rem: DB 'Ваш вариант: ',0  
SECTION .bss  
x: RESB 80  
SECTION .text  
GLOBAL _start  
_start:  
    mov eax, msg  
    call sprintLF  
    mov ecx, x  
    mov edx, 80  
    call sread  
    mov eax,x ; вызов подпрограммы преобразования  
    call atoi ; ASCII кода в число, `eax=x'  
    xor edx,edx  
    mov ebx,20  
    div ebx  
    inc edx  
    mov eax,rem  
    call sprint  
    mov eax,edx  
    call iprintLF  
    call quit
```

At the bottom of the terminal window, there is a menu bar with the following options:

- ^G Help
- ^O Write Out
- ^W Where Is
- ^K Cut
- ^T Execute
- ^X Exit
- ^R Read File
- ^V Replace
- ^U Paste
- ^J Justify

Рис. 16: Создание файла и изменения текста variant.asm

Потом создала исполняемый файл запустилаа его

The screenshot shows a terminal window with the following content:

```
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab06$ nasm -f elf variant.asm  
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab06$ ld -m elf_i386 -o variant variant.o  
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab06$ ./variant  
Введите № студенческого билета:  
1032255901  
Ваш вариант: 2  
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab06$
```

Рис. 17: Создание исполняемого файла lab6-3.asm и запуск файла

#### 4.3 Ответы на вопросы по программе

1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’? Ответ: За вывод сообщения “Ваш вариант” отвечают строки кода: `mov eax,rem` `call sprint`

2. Для чего используется следующие инструкции? `mov ecx, x` `mov edx, 80` `call sread`  
 Ответ: Инструкция `mov ecx, x` используется, чтобы положить адрес вводимой строки `x` в регистр `ecx` `mov edx, 80` - запись в регистр `edx` длины вводимой строки `call sread` - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры
3. Для чего используется инструкция “`call atoi`”? Ответ: Инструкция «`call atoi`» используется для преобразования строки в целое число.
4. Какие строки листинга 6.4 отвечают за вычисления варианта? Ответ: `xor edx,edx` ; обнуление `edx` для корректной работы `div mov ebx,20 ; ebx = 20 div ebx ; eax = eax/20, edx - остаток от деления inc edx ; edx = edx + 1`
5. В какой регистр записывается остаток от деления при выполнении инструкции “`div ebx`”? Ответ: При выполнении инструкции `div ebx` остаток от деления записывается в регистр `edx`
6. Для чего используется инструкция “`inc edx`”? Ответ: Инструкция `inc edx` увеличивает значение регистра `edx` на 1
7. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?  
 Ответ: За вывод на экран результатов вычислений отвечают строки: `mov eax,edx` `call iprintLF`

## 5 Выполнение заданий для самостоятельной работы

Для начала я создалаа файл lab6-4.asm с помощью touch. И открыла файл для редактирования, ввела туда текст программы для вычисления  $(11+x)^*2-6$

```
..026/Архитектура компьютера/study_2025-2026_арх-pc/labs/lab06/lab6-4.asm *
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; секция инициализированных данных
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0
SECTION .bss ; секция не инициализированных данных
x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры, выделенный р
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
; ---- Вычисление выражения
mov eax, msg ; запись адреса выводимого сообщения в eax
call sprint ; вызов подпрограммы печати сообщения
mov ecx, x ; запись адреса переменной в ecx
mov edx, 80 ; запись длины вводимого значения в edx
call sread ; вызов подпрограммы ввода сообщения
mov eax,x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
add eax,11; eax = eax+11 = x + 11
mov ebx,2 ; запись значения 2 в регистр ebx
mul ebx; EAX=EAX*EBX = (x+11)*2
add eax,-6; eax = eax-6 = (x+11)*2-6
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения
```

**^G Help      ^O Write Out    ^W Where Is    ^K Cut      ^T Execute    ^C Location  
 ^X Exit      ^R Read File    ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line**

Рис. 18: Создание файла и изменения текста lab6-4.asm

Потом создала исполняемый файл и запустила его. И ввела туда цифру 1

```
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_a  
rh-pc/labs/lab06$ nasm -f elf lab6-4.asm  
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_a  
rh-pc/labs/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o  
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_a  
rh-pc/labs/lab06$ ./lab6-4  
Введите значение переменной x: 5  
Результат: 26  
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_a
```

Рис. 19: Создание исполняемого файла *lab6-4.asm* и запуск файла

Еще раз запустила файл но в этот раз ввел цифру 9 и по алгоритму отработала верно и дал верный ответ

```
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_a  
rh-pc/labs/lab06$ ./lab6-4  
Введите значение переменной x: 25  
Результат: 66  
vboxuser@gulsoltan:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_a  
rh-pc/labs/lab06$
```

Рис. 20: Запуск исполняемого файла *lab6-4.asm*

## 6 Выводы

При выполнении данной лабораторной работы я освоил арифметические инструкции языка ассемблера NASM.

## Список литературы

1. Лабораторная работа №7
2. Таблица ASCII