

# **Отчёт по лабораторной работе 6**

**Арифметические операции в NASM.**

Вахиш Дуаа Иссам Али Ахмед

# Содержание

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Цель работы</b>                                  | <b>5</b>  |
| <b>2</b> | <b>Выполнение лабораторной работы</b>               | <b>6</b>  |
| 2.1      | Символьные и численные данные в NASM . . . . .      | 6         |
| 2.2      | Выполнение арифметических операций в NASM . . . . . | 12        |
| 2.3      | Задание для самостоятельной работы . . . . .        | 18        |
| <b>3</b> | <b>Выводы</b>                                       | <b>21</b> |

## Список иллюстраций

|      |   |    |
|------|---|----|
| 2.1  | Программа в файле lab6-1.asm . . . . .  | 7  |
| 2.2  | Запуск программы lab6-1.asm . . . . .   | 7  |
| 2.3  | Программа в файле lab6-1.asm . . . . .  | 8  |
| 2.4  | Запуск программы lab6-1.asm . . . . .   | 9  |
| 2.5  | Программа в файле lab6-2.asm . . . . .  | 9  |
| 2.6  | Запуск программы lab6-2.asm . . . . .   | 10 |
| 2.7  | Программа в файле lab6-2.asm . . . . .  | 11 |
| 2.8  | Запуск программы lab6-2.asm . . . . .   | 11 |
| 2.9  | Запуск программы lab6-2.asm . . . . .   | 12 |
| 2.10 | Программа в файле lab6-3.asm . . . . .  | 13 |
| 2.11 | Запуск программы lab6-3.asm . . . . .   | 13 |
| 2.12 | Программа в файле lab6-3.asm . . . . .  | 14 |
| 2.13 | Запуск программы lab6-3.asm . . . . .   | 15 |
| 2.14 | Программа в файле variant.asm . . . . . | 16 |
| 2.15 | Запуск программы variant.asm . . . . .  | 16 |
| 2.16 | Программа в файле task.asm . . . . .    | 19 |
| 2.17 | Запуск программы task.asm . . . . .     | 20 |

## **Список таблиц**

# 1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

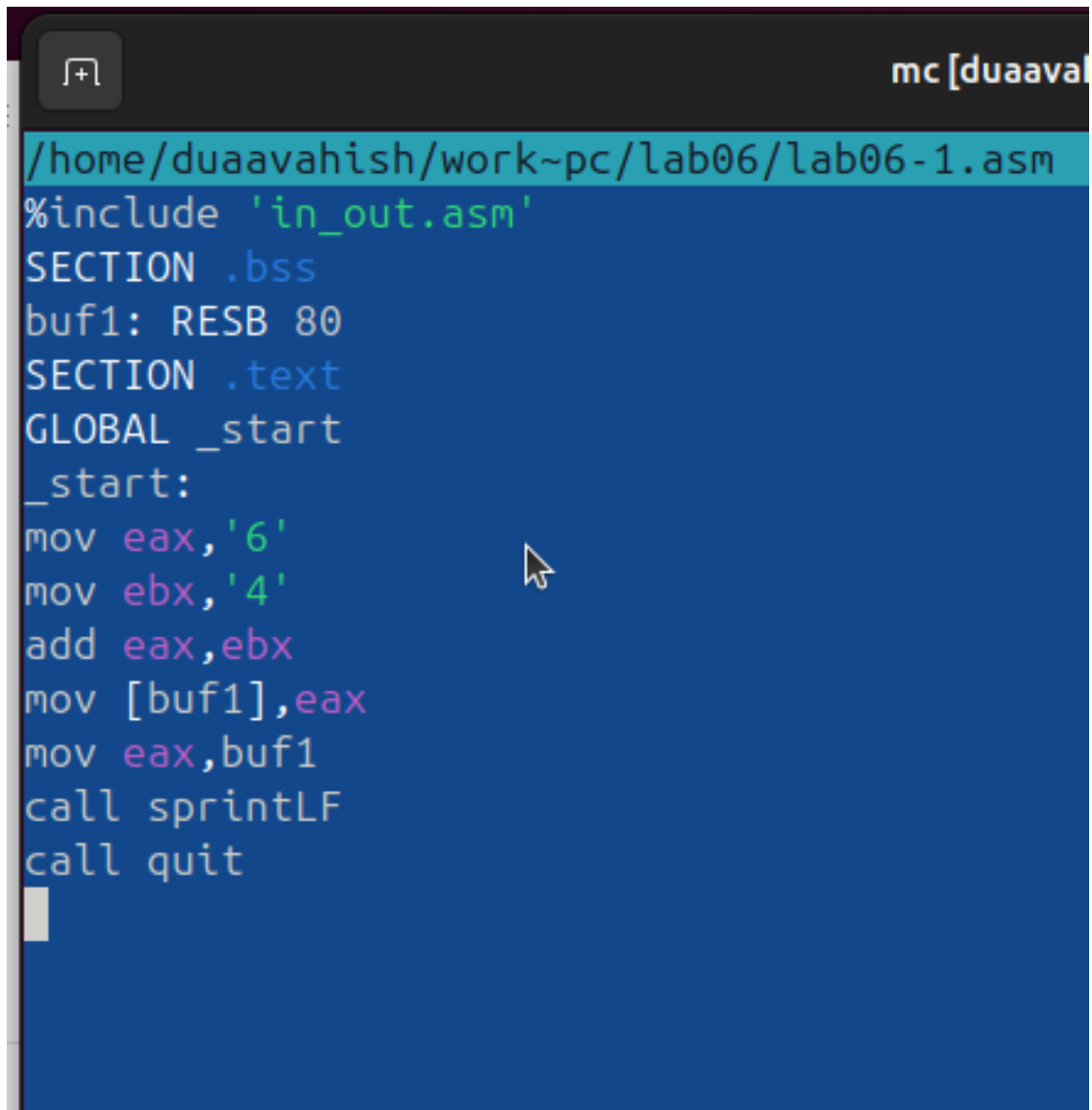
## 2 Выполнение лабораторной работы

### 2.1 Символьные и численные данные в NASM

Создала каталог для программ лабораторной работы № 6, перешла в него и создала файл lab6-1.asm.

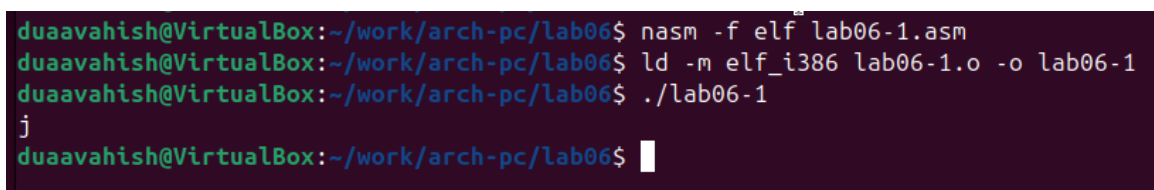
Рассмотрим примеры программ для вывода символьных и численных значений. Программы будут выводить значения, записанные в регистр `eax`.

В данной программе в регистр `eax` записывается символ 6 (`mov eax, „6“`), в регистр `ebx` — символ 4 (`mov ebx, „4“`). Затем прибавляем значение регистра `ebx` к `eax` (`add eax, ebx`, результат сложения запишется в `eax`). Далее выводим результат. Так как для работы функции `sprintf` в `eax` должен быть записан адрес, используем дополнительную переменную: записываем значение `eax` в `buf1` (`mov [buf1], eax`), затем адрес `buf1` в `eax` (`mov eax, buf1`) и вызываем `sprintf`.

A screenshot of a text editor window titled 'mc [duaava...'. The editor displays assembly code for a file named '/home/duaavahish/work~pc/lab06/lab06-1.asm'. The code includes a directive to include 'in\_out.asm', defines a .bss section with a buffer 'buf1' of 80 bytes, and a .text section. It declares the '\_start' symbol as global and defines the '\_start' function. Inside the function, it moves the character '6' into the 'eax' register, moves the character '4' into the 'ebx' register, adds 'ebx' to 'eax', moves the result into 'buf1', calls 'sprintLF', and finally calls 'quit'.

```
/home/duaavahish/work~pc/lab06/lab06-1.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
```

Рисунок 2.1: Программа в файле lab6-1.asm

A screenshot of a terminal window showing the compilation and execution of the assembly program. The user is at a prompt 'duaavahish@VirtualBox:~/work/arch-pc/lab06\$'. They run 'nasm -f elf lab06-1.asm', then 'ld -m elf\_i386 lab06-1.o -o lab06-1', and finally './lab06-1'. The program runs successfully and returns to the prompt.

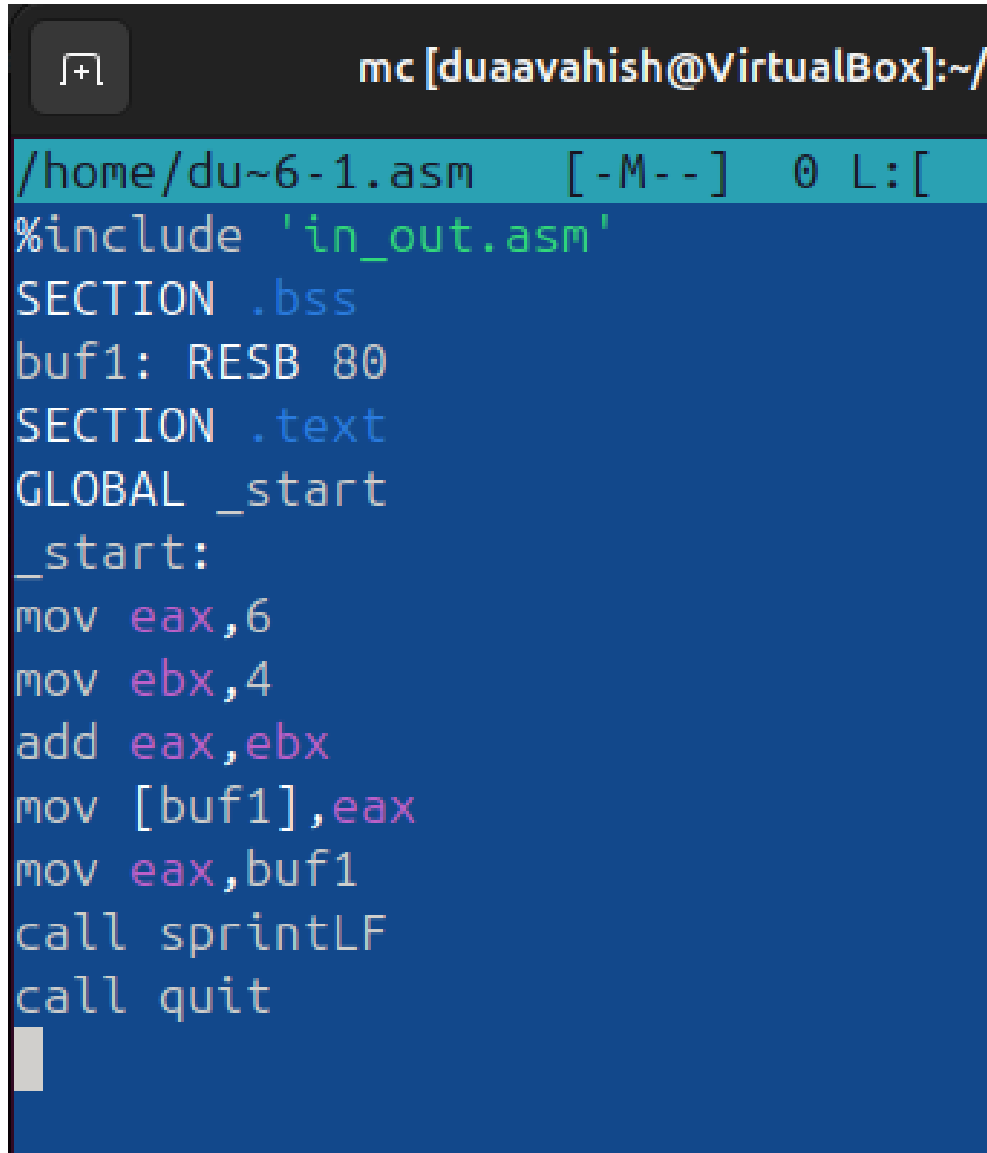
```
duaavahish@VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm
duaavahish@VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o -o lab06-1
duaavahish@VirtualBox:~/work/arch-pc/lab06$ ./lab06-1
j
duaavahish@VirtualBox:~/work/arch-pc/lab06$
```

Рисунок 2.2: Запуск программы lab6-1.asm

В данном случае при выводе значения регистра `eax` ожидаем увидеть чис-

ло 10, но результатом будет символ j. Это происходит из-за того, что код символа 6 равен 00110110 (или 54 в десятичном представлении), а символа 4 — 00110100 (52). Команда `add eax, ebx` записывает в `eax` сумму кодов — 01101010 (106), что соответствует коду символа j.

Далее изменяю текст программы и вместо символов записываю в регистры числа.



```
mc [duaavahish@VirtualBox]:~/
/home/du~6-1.asm  [-M - -]  0  L:[
#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 sprintfLF
call quit
```

Рисунок 2.3: Программа в файле lab6-1.asm



```

duaavahish@VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm
duaavahish@VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o -o lab06-1
duaavahish@VirtualBox:~/work/arch-pc/lab06$ ./lab06-1

duaavahish@VirtualBox:~/work/arch-pc/lab06$

```

Рисунок 2.4: Запуск программы lab6-1.asm

При исполнении программы в этом случае тоже не получаем число 10. Вместо этого выводится символ с кодом 10, который является символом конца строки и добавляет пустую строку в консоли.

Для работы с числами в файле `in_out.asm` предусмотрены подпрограммы для преобразования ASCII символов в числа и обратно. Изменяю текст программы, используя эти функции.

```

mc [duaavahish@VirtualBox]:~/work/arch-pc/l
/home/duaavahish/work/arch-pc/lab06-2.asm [ - - - - ] 9 L: [ 1+ 8 9/
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax, '6'
mov ebx, '4'
add eax, ebx
call iprintLF
call quit

```

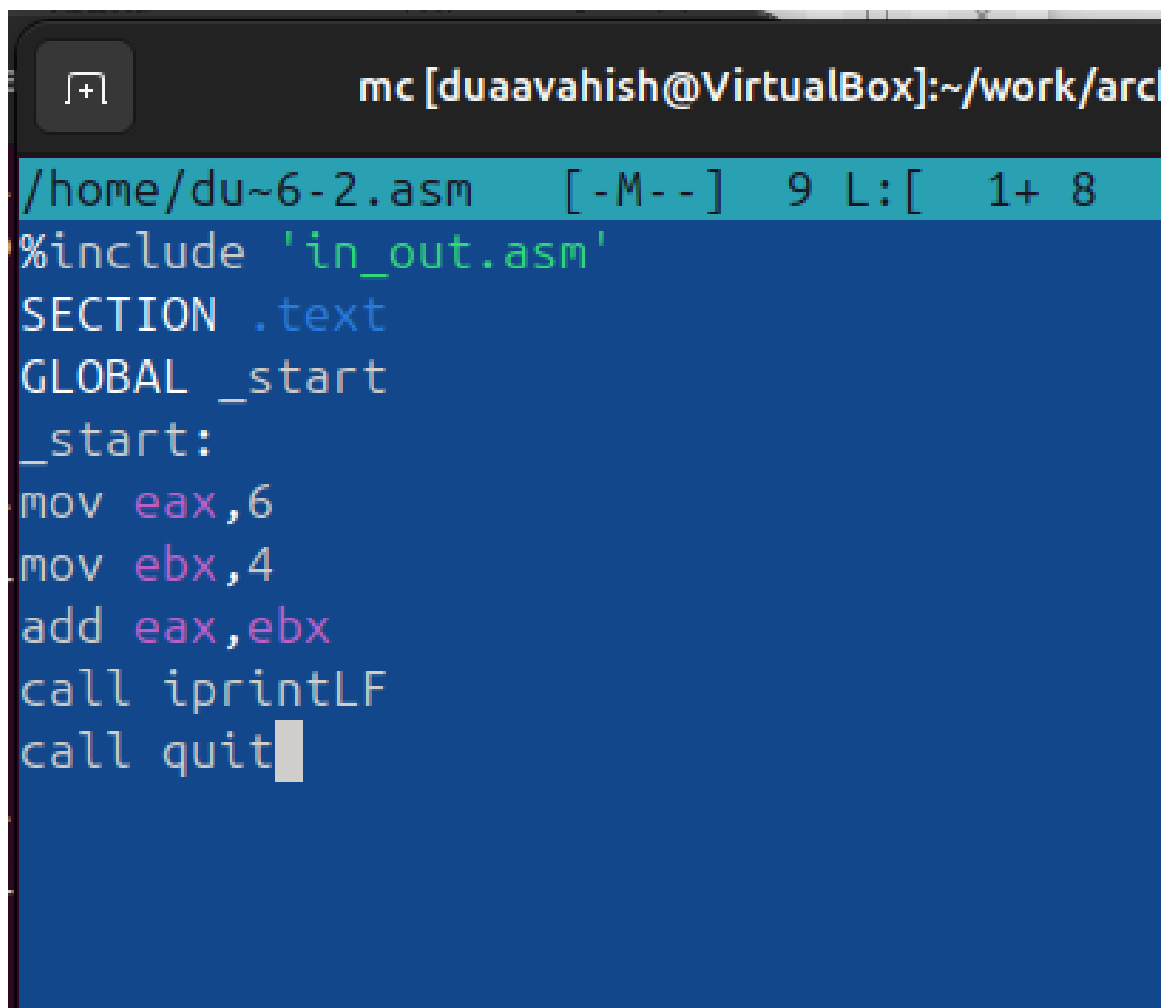
Рисунок 2.5: Программа в файле lab6-2.asm

```
duaavahish@VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm
duaavahish@VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-2.o -o lab06-2
duaavahish@VirtualBox:~/work/arch-pc/lab06$ ./lab06-2
106
duaavahish@VirtualBox:~/work/arch-pc/lab06$
```

Рисунок 2.6: Запуск программы lab6-2.asm

В результате работы программы получаем число 106. Здесь, как и в первом примере, команда `add` складывает коды символов 6 и 4 ( $54 + 52 = 106$ ). В отличие от прошлой программы, функция `iprintLF` позволяет вывести число, а не символ.

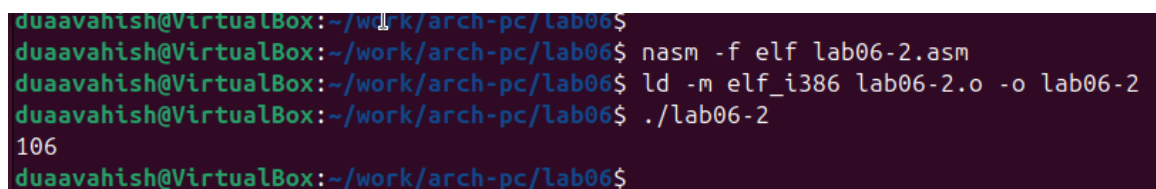
Аналогично предыдущему примеру заменяю символы на числа.



```
mc [duaavahish@VirtualBox]:~/work/arch-  
/home/duaavahish/lab06-2.asm [-M--] 9 L:[ 1+ 8  
%include 'in_out.asm'  
SECTION .text  
GLOBAL _start  
_start:  
mov eax,6  
mov ebx,4  
add eax,ebx  
call iprintLF  
call quit
```

Рисунок 2.7: Программа в файле lab6-2.asm

Функция `iprintLF` позволяет вывести число, и операндами были числа (а не коды символов), поэтому получаем число 10.



```
duaavahish@VirtualBox:~/work/arch-pc/lab06$  
duaavahish@VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm  
duaavahish@VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-2.o -o lab06-2  
duaavahish@VirtualBox:~/work/arch-pc/lab06$ ./lab06-2  
106  
duaavahish@VirtualBox:~/work/arch-pc/lab06$
```

Рисунок 2.8: Запуск программы lab6-2.asm

Заменяю функцию `iprintLF` на `iprint`, создаю исполняемый файл и запускаю его. Вывод отличается тем, что нет переноса строки.

```
duaavahish@VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm
duaavahish@VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-2.o -o lab06-2
duaavahish@VirtualBox:~/work/arch-pc/lab06$ ./lab06-2
10duaavahish@VirtualBox:~/work/arch-pc/lab06$
duaavahish@VirtualBox:~/work/arch-pc/lab06$
```

Рисунок 2.9: Запуск программы lab6-2.asm

## 2.2 Выполнение арифметических операций в NASM

В качестве примера выполнения арифметических операций в NASM рассмотрим программу для вычисления выражения

$$f(x) = (5 * 2 + 3)/3$$

.

```
mc [duaavahish@VirtualBox]:~/work/arch-pc/la
/home/du~6-3.asm [----] 7 L:[ 1+14 15/ 2
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',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
```

Рисунок 2.10: Программа в файле lab6-3.asm

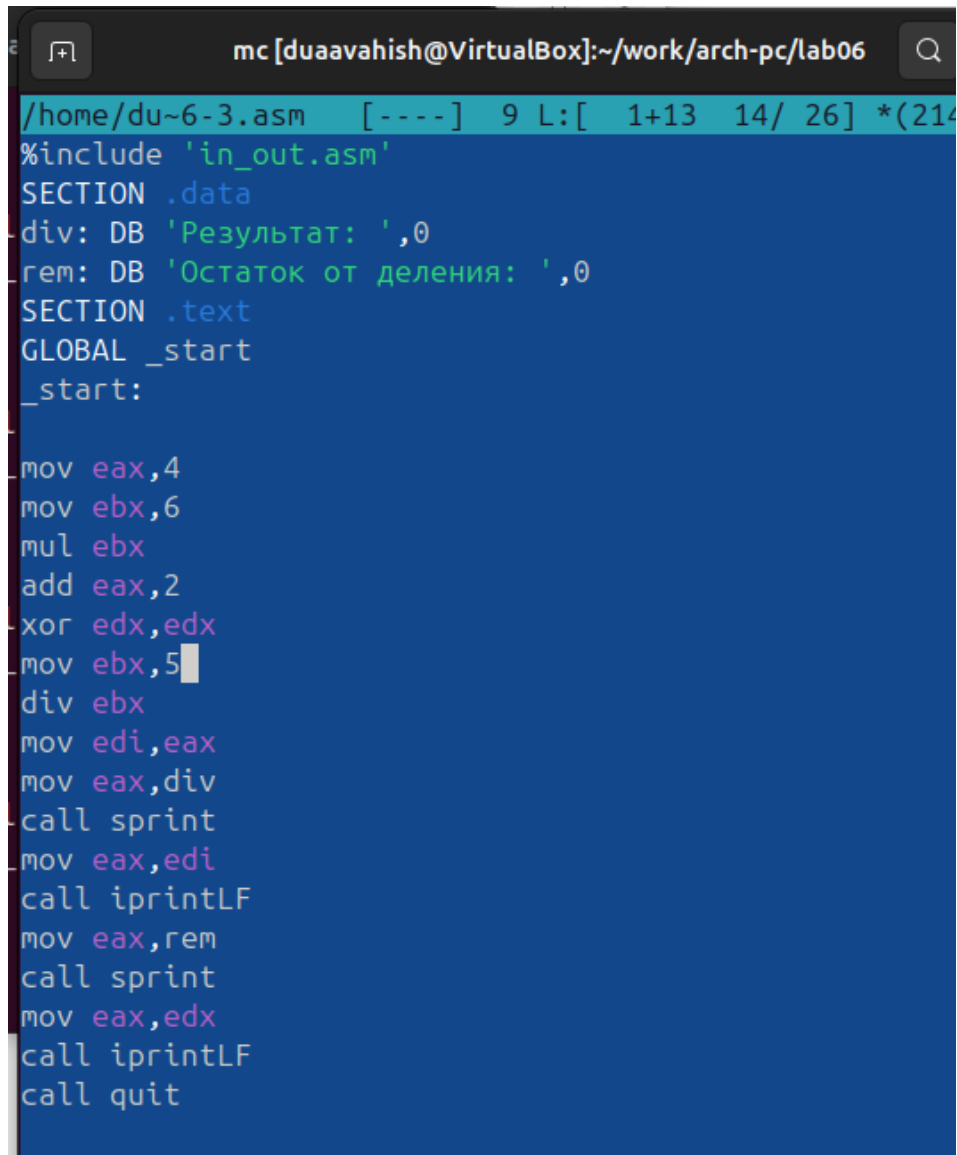
```
duaavahish@VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab06-3.asm
duaavahish@VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-3.o -o lab06-3
duaavahish@VirtualBox:~/work/arch-pc/lab06$ ./lab06-3
Результат: 4
Остаток от деления: 1
duaavahish@VirtualBox:~/work/arch-pc/lab06$
```

Рисунок 2.11: Запуск программы lab6-3.asm

Изменяю текст программы для вычисления выражения

$$f(x) = (4 * 6 + 2)/5$$

. Создаю исполняемый файл и проверяю его работу.



```
mc [duaavahish@VirtualBox]:~/work/arch-pc/lab06
/home/du~6-3.asm [----] 9 L:[ 1+13 14/ 26] *(214
#include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',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
```

Рисунок 2.12: Программа в файле lab6-3.asm

```
duaavahish@VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab06-3.asm
duaavahish@VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-3.o -o lab06-3
duaavahish@VirtualBox:~/work/arch-pc/lab06$ ./lab06-3
Результат: 5
Остаток от деления: 1
duaavahish@VirtualBox:~/work/arch-pc/lab06$
```

Рисунок 2.13: Запуск программы lab6-3.asm

В качестве другого примера рассмотрим программу для вычисления варианта задания по номеру студенческого билета.

Число, над которым проводятся арифметические операции, вводится с клавиатуры. Ввод с клавиатуры осуществляется в символьном виде, и для корректной работы арифметических операций в NASM символы необходимо преобразовать в числа. Для этого используется функция `atoi` из файла `in_out.asm`.

```
mc [duaavahish@VirtualBox]:~/work/arch-pc/lab06
/home/duaavahish/variant.asm [----] 2 L:[ 1+18 19/ 26] *(3
#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 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 iprintfLF
call quit
```

Рисунок 2.14: Программа в файле variant.asm

```
duaavahish@VirtualBox:~/work/arch-pc/lab06$ nasm -f elf variant.asm
duaavahish@VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 variant.o -o variant
duaavahish@VirtualBox:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132259465
Ваш вариант: 6
duaavahish@VirtualBox:~/work/arch-pc/lab06$
```

Рисунок 2.15: Запуск программы variant.asm



### 2.2.1 Ответы на вопросы

1. Какие строки листинга отвечают за вывод сообщения „Ваш вариант“?

- `mov eax, mem` – перекладывает в регистр значение переменной с фразой „Ваш вариант“.
- `call sprint` – вызов подпрограммы вывода строки.

2. Для чего используются инструкции?

```
mov ecx, x
mov edx, 80
call sread
```

- Считывает значение студбилета в переменную X из консоли

3. Для чего используется инструкция «call atoi»?

- Эта подпрограмма переводит введенные символы в числовой формат.

4. Какие строки листинга отвечают за вычисления варианта?

```
xor edx, edx
mov ebx, 20
div ebx
inc edx
```

- Здесь происходит деление номера студ билета на 20. В регистре `edx` хранится остаток, к нему прибавляется 1.

5. В какой регистр записывается остаток от деления при выполнении инструкции «div ebx»?

- регистр `edx`

6. Для чего используется инструкция «inc edx»?

- по формуле вычисления варианта нужно прибавить единицу

7. Какие строки листинга отвечают за вывод на экран результата вычислений?

- mov eax,edx – результат перекладывается в регистр eax
- call iprintLF – вызов подпрограммы вывода

## 2.3 Задание для самостоятельной работы

Написать программу вычисления выражения  $y = f(x)$ . Программа должна выводить выражение для вычисления, выводить запрос на ввод значения  $x$ , вычислять заданное выражение в зависимости от введенного  $x$ , выводить результат вычислений. Вид функции  $f(x)$  выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создайте исполняемый файл и проверьте его работу для значений  $x_1$  и  $x_2$  из 6.3.

Получили вариант 6 -

$$\frac{x^3}{2} + 1$$

для

$$x_1 = 2, x_2 = 5$$

```
/home/du~task.asm [----] 0 L:[
start:
mov eax, msg
call sprintf
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi

mov ebx, eax
mul ebx
mul ebx
mov ebx, 2
xor edx, edx
div ebx
add eax, 1

mov ebx, eax
mov eax, rem
call sprintf
mov eax, ebx
call iprintLF
```

Рисунок 2.16: Программа в файле task.asm

Если подставить 2 получается

$$\frac{2^3}{2} + 1 = 5$$

Если подставить 5 получается

$$\frac{5^3}{2} + 1 = 63$$

```
duaavahish@VirtualBox:~/work/arch-pc/lab06$  
duaavahish@VirtualBox:~/work/arch-pc/lab06$ nasm -f elf task.asm  
duaavahish@VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 task.o -o task  
duaavahish@VirtualBox:~/work/arch-pc/lab06$ ./task  
Введите X  
2  
выражение = : 5  
duaavahish@VirtualBox:~/work/arch-pc/lab06$ ./task  
Введите X  
5  
выражение = : 63  
duaavahish@VirtualBox:~/work/arch-pc/lab06$
```

Рисунок 2.17: Запуск программы task.asm

Программа считает верно.

## **3 Выводы**

Изучили работу с арифметическими операциями.