

# **Отчёт по лабораторной работе 6**

**Архитектура ЭВМ**

Шеенкова Мария

# **Содержание**

<b>1 Цель работы</b>	<b>5</b>
<b>2 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 Ответы на вопросы . . . . .	16
2.4 Задание для самостоятельной работы . . . . .	18
<b>3 Выводы</b>	<b>21</b>

# **Список иллюстраций**

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

Создаю каталог для программам лабораторной работы № 6, перехожу в него и создаю файл lab6-1.asm.

Рассмотрим примеры программ 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”). Далее к значению в регистре eax прибавляем значение регистра ebx (add eax,ebx, результат сложения запишется в регистр eax). Далее выводим результат. (рис. 2.1) (рис. 2.2)

Так как для работы функции sprintLF в регистр eax должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра eax в переменную buf1 (mov [buf1],eax), а затем запишем адрес переменной buf1 в регистр eax (mov eax,buf1) и вызовем функцию sprintLF.

The screenshot shows a terminal window with a dark blue background. At the top, it displays the command line: `mashenkova@ubuntu: ~/work/arch-pc/`. Below this, the assembly code for `lab6-1.asm` is shown:

```
/home/mashenkova/06-1.asm [-----] 0 L:[ 1+14 15/ 15] *(17)
%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 lab6-1.asm

The screenshot shows a terminal window with a dark blue background. It displays the following commands and their output:

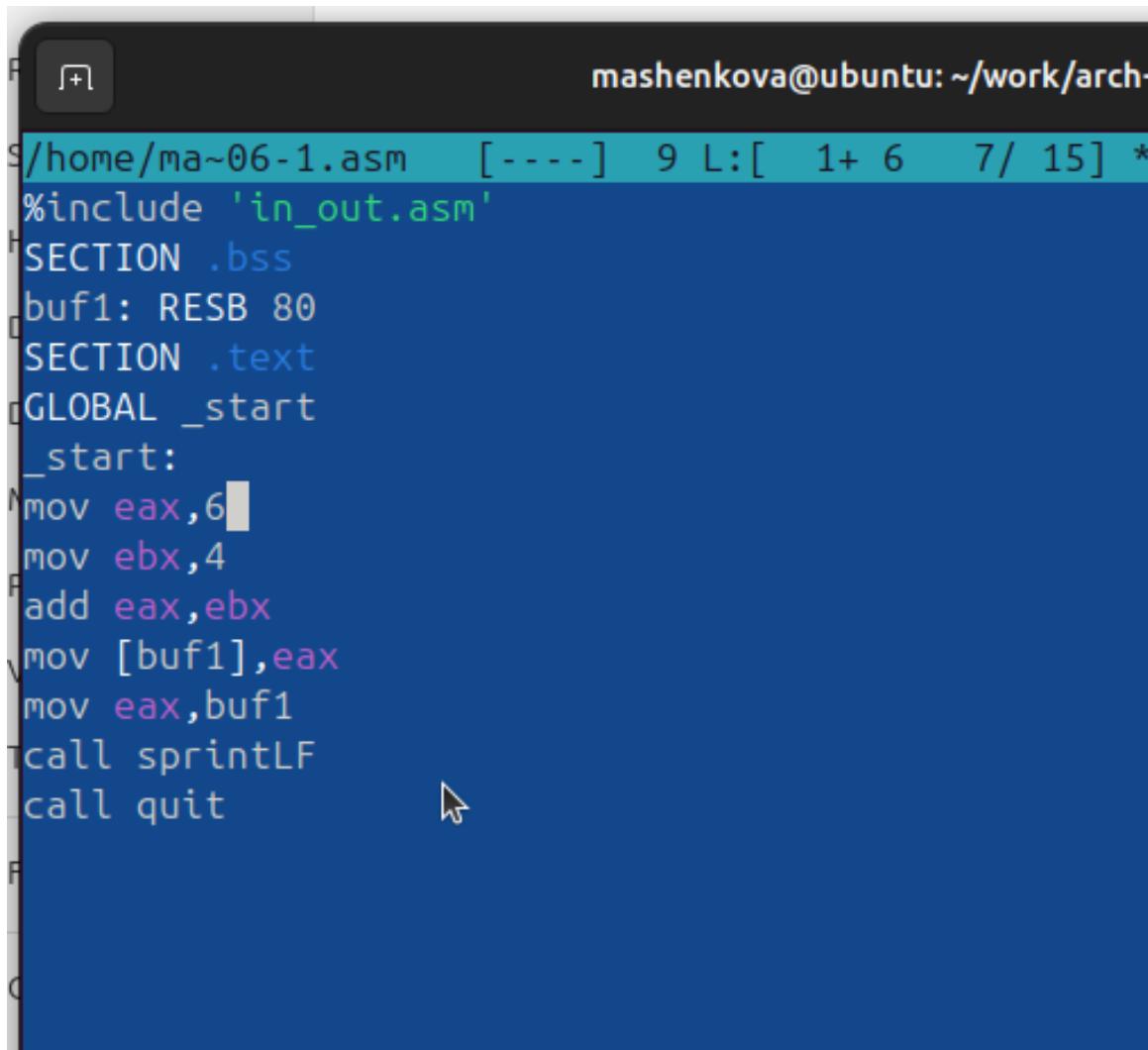
```
mashenkova@ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm
mashenkova@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o -o lab06-1
mashenkova@ubuntu:~/work/arch-pc/lab06$ ./lab06-1
j
mashenkova@ubuntu:~/work/arch-pc/lab06$
```

Рисунок 2.2: Запуск программы lab6-1.asm

В данном случае при выводе значения регистра `eax` мы ожидаем увидеть число 10. Однако результатом будет символ `j`. Это происходит потому, что код символа 6 равен 00110110 в двоичном представлении (или 54 в десятичном

представлении), а код символа 4 – 00110100 (52). Команда add eax,ebx запишет в регистр eax сумму кодов – 01101010 (106), что в свою очередь является кодом символа j.

Далее изменяю текст программы и вместо символов, запишуем в регистры числа. (рис. 2.3) (рис. 2.4)



The screenshot shows a terminal window with the following details:

- Terminal title: mashenkova@ubuntu: ~/work/arch-
- File path: /home/mashenkova/06-1.asm
- Line numbers: 9 L:[ 1+ 6 7/ 15] \*
- Code content:

```
%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 2.3: Программа lab6-1.asm

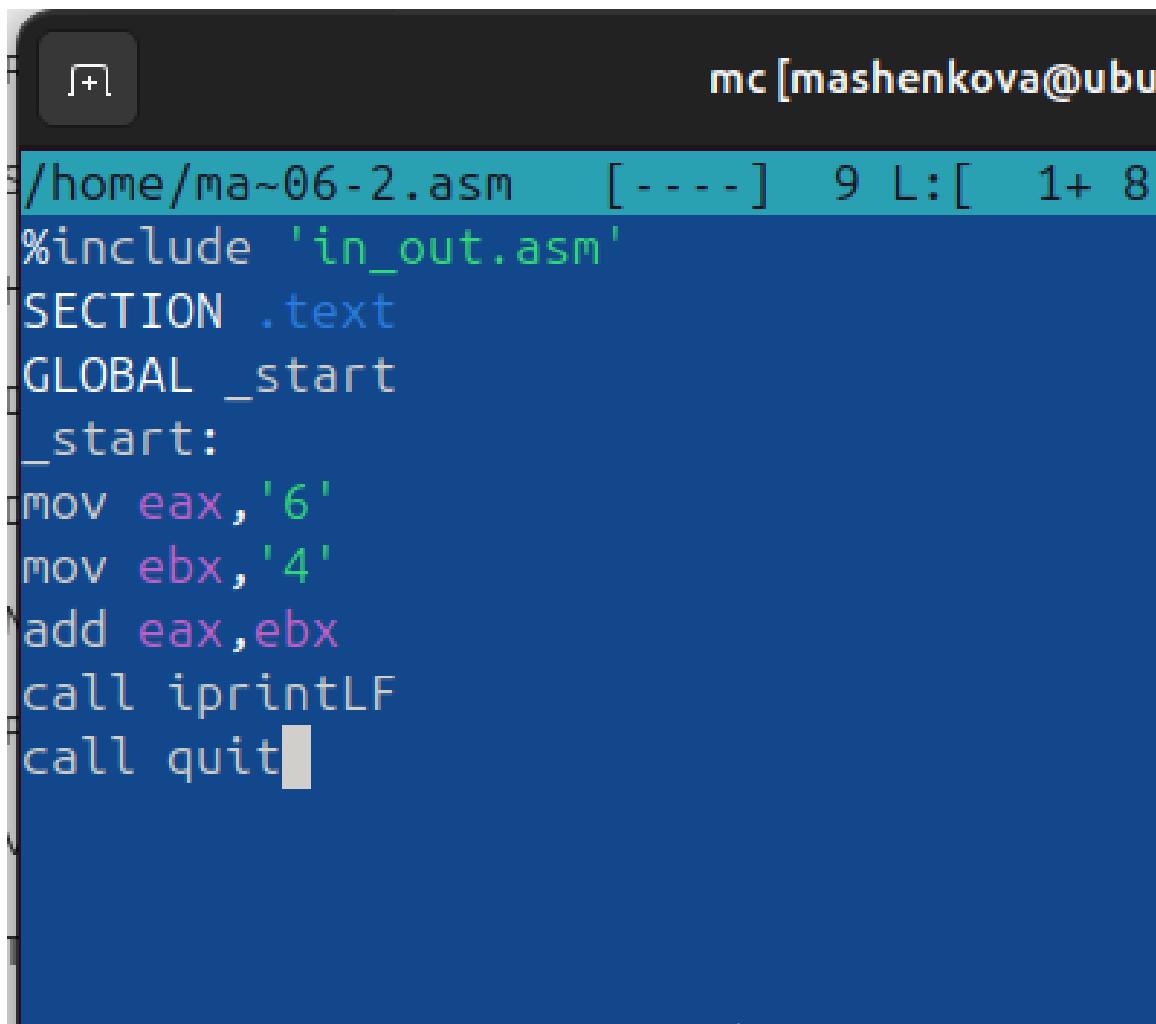
```
mashenkova@ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm
mashenkova@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o -o lab06-1
mashenkova@ubuntu:~/work/arch-pc/lab06$ ./lab06-1
```

```
mashenkova@ubuntu:~/work/arch-pc/lab06$ █
```

Рисунок 2.4: Запуск программы lab6-1.asm

Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10. Это символ конца строки (возврат каретки). В консоле он не отображается, но добавляет пустую строку.

Как отмечалось выше, для работы с числами в файле in\_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразую текст программы с использованием этих функций. (рис. 2.5) (рис. 2.6)



The screenshot shows a terminal window with a dark blue background. In the top right corner, the text "mc [mashenkova@ubu" is visible. The main area of the terminal contains the following assembly code:

```
/home/ma~06-2.asm      [ ---- ]  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 2.5: Программа lab6-2.asm



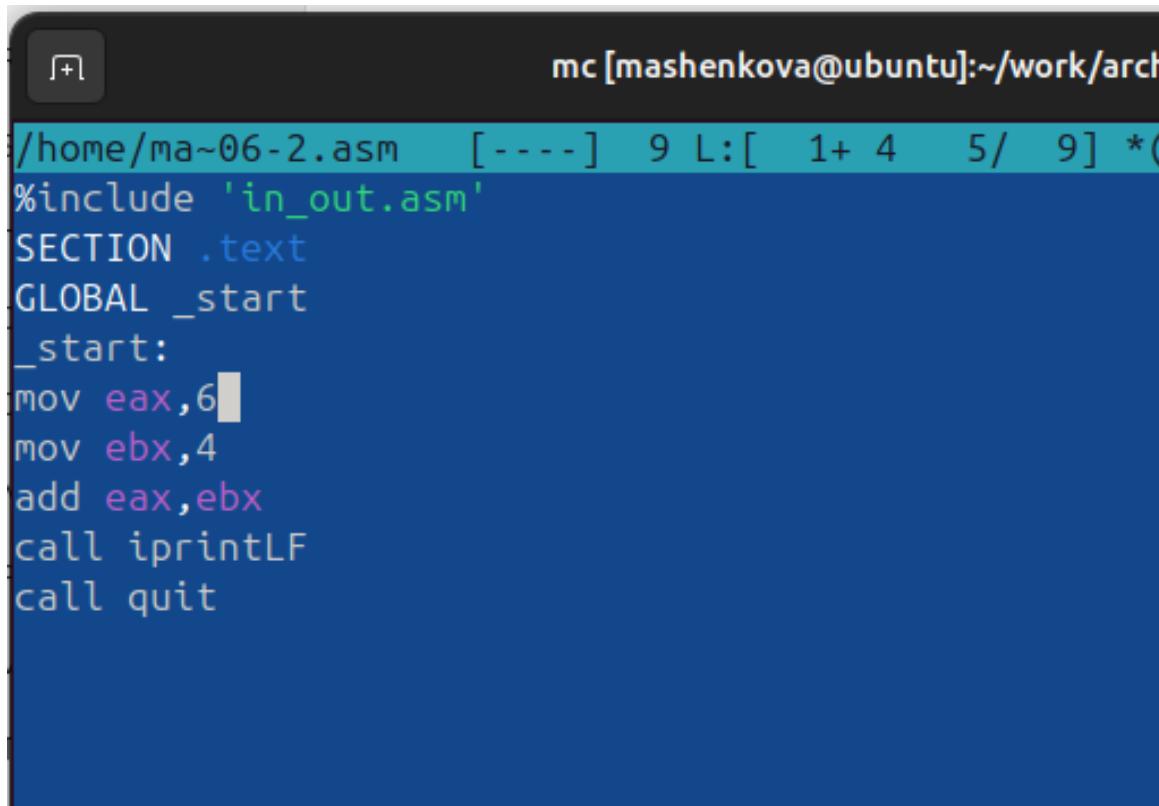
```
mashenkova@ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm
mashenkova@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-2.o -o lab06-2
mashenkova@ubuntu:~/work/arch-pc/lab06$ ./lab06-2
106
mashenkova@ubuntu:~/work/arch-pc/lab06$ █
```

Рисунок 2.6: Запуск программы lab6-2.asm

В результате работы программы мы получим число 106. В данном случае, как и в первом, команда add складывает коды символов „6“ и „4“ ( $54+52=106$ ). Однако, в отличии от прошлой программы, функция iprintLF позволяет выве-

сти число, а не символ, кодом которого является это число.

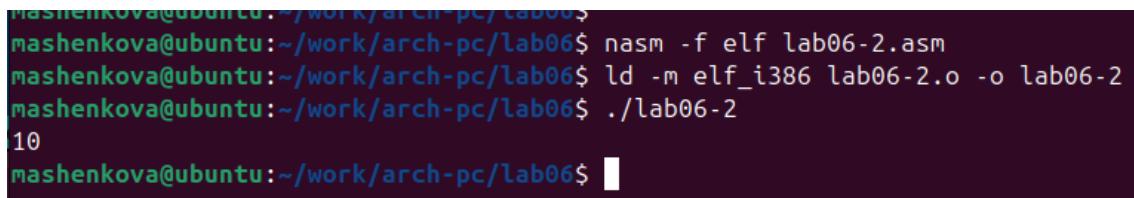
Аналогично предыдущему примеру изменим символы на числа. (рис. 2.7) (рис. 2.8)



```
mc [mashenkova@ubuntu]:~/work/arch-  
/home/ma~06-2.asm  [----]  9 L:[ 1+ 4 5/ 9] *(  
%include 'in_out.asm'  
SECTION .text  
GLOBAL _start  
_start:  
    mov eax,6  
    mov ebx,4  
    add eax,ebx  
    call iprintLF  
    call quit
```

Рисунок 2.7: Программа lab6-2.asm

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



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

Рисунок 2.8: Запуск программы lab6-2.asm

Заменила функцию iprintLF на iprint. Создала исполняемый файл и запустила его. Вывод отличается тем, что нет переноса строки. (рис. 2.9)

```
mashenkova@ubuntu:~/work/arch-pc/Lab06$  
mashenkova@ubuntu:~/work/arch-pc/Lab06$ nasm -f elf lab06-2.asm  
mashenkova@ubuntu:~/work/arch-pc/Lab06$ ld -m elf_i386 lab06-2.o -o lab06-2  
mashenkova@ubuntu:~/work/arch-pc/Lab06$ ./lab06-2  
10mashenkova@ubuntu:~/work/arch-pc/Lab06$  
mashenkova@ubuntu:~/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 приведем программу вычисления арифметического выражения  $f(x) = (5 * 2 + 3)/3$ . (рис. 2.10) (рис. 2.11)

The screenshot shows a terminal window titled 'mc [mashenkova@ubuntu]:~/work/arch-pc/lab06'. The code displayed is:

```
/home/ma~6-3.asm      [----]  0 L:[ 1+25 26/ 26] *(345 /
%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 lab6-3.asm

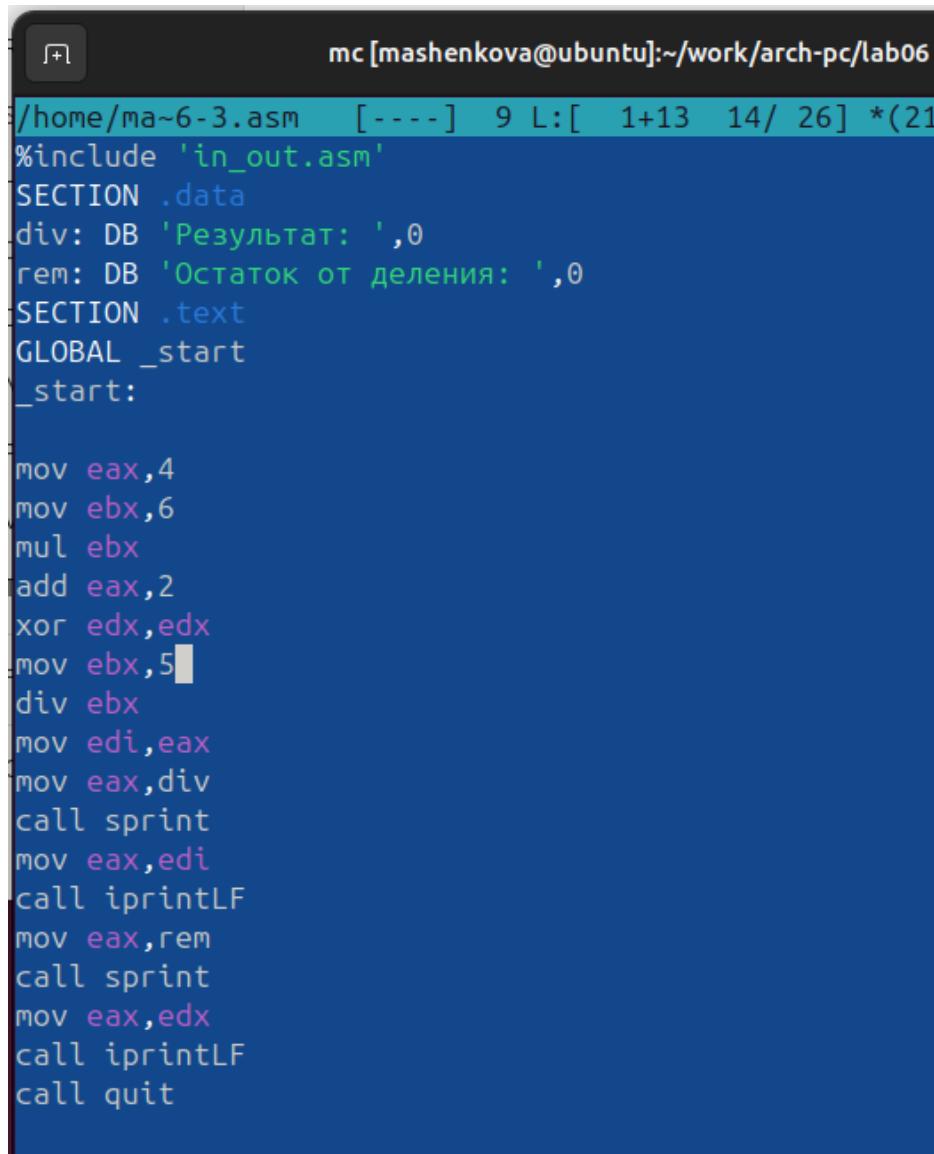
The screenshot shows a terminal window with the following command-line session:

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

Рисунок 2.11: Запуск программы lab6-3.asm

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

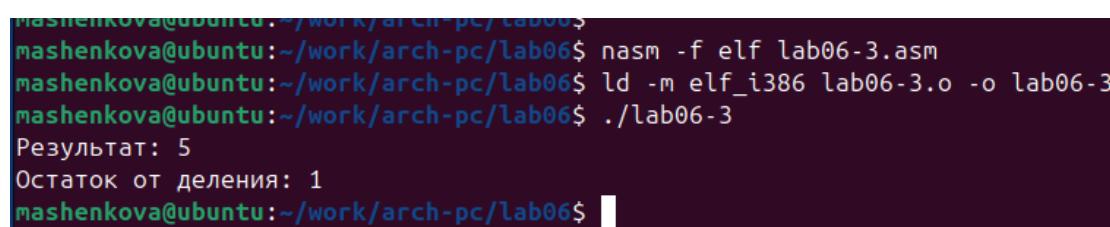
Создала исполняемый файл и проверила его работу. (рис. 2.12) (рис. 2.13)



```
mc [mashenkova@ubuntu]:~/work/arch-pc/lab06
/home/ma~6-3.asm  [----]  9 L:[ 1+13 14/ 26] *(21
%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 lab6-3.asm

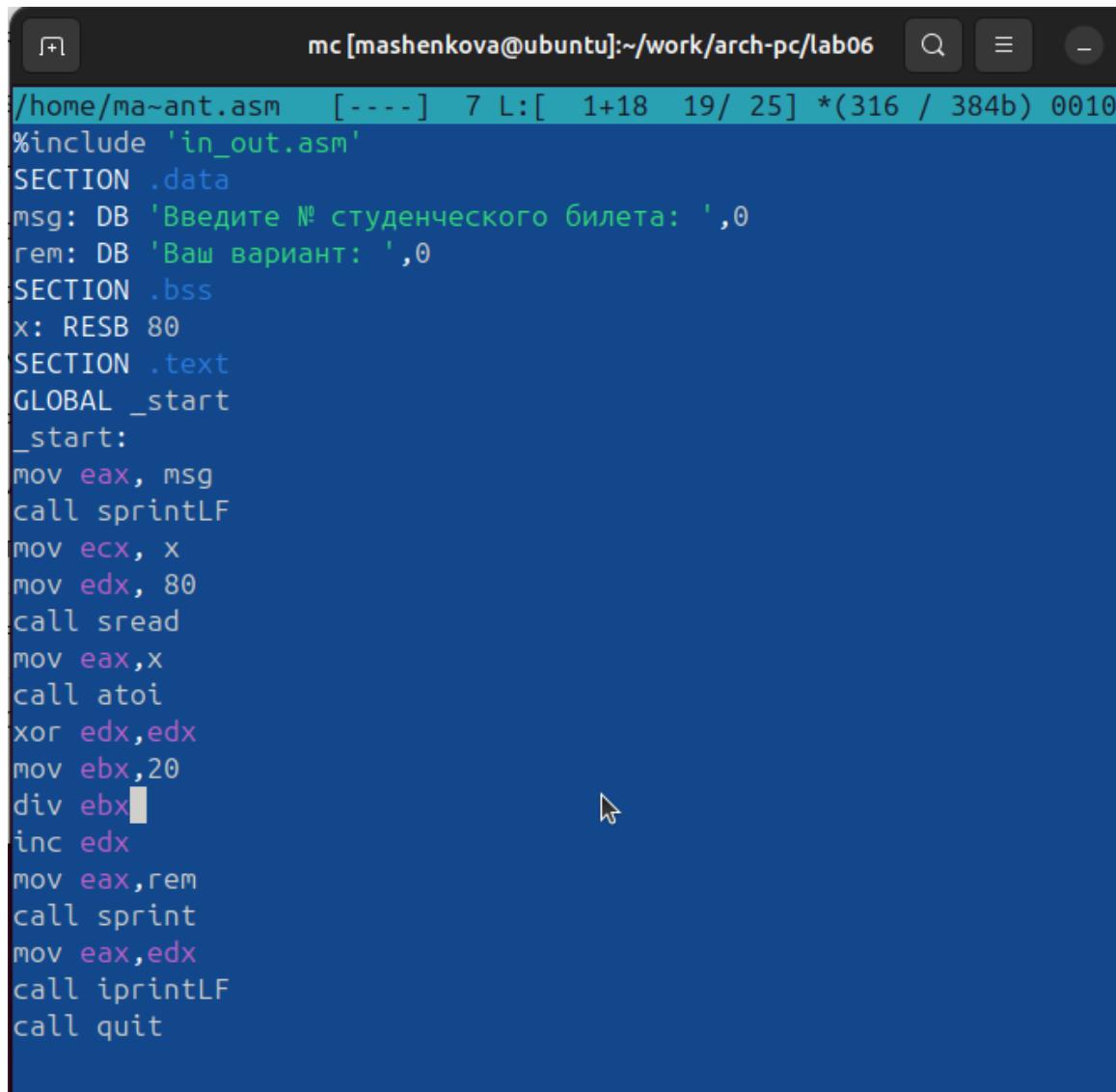


```
mashenkova@ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab06-3.asm
mashenkova@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-3.o -o lab06-3
mashenkova@ubuntu:~/work/arch-pc/lab06$ ./lab06-3
Результат: 5
Остаток от деления: 1
mashenkova@ubuntu:~/work/arch-pc/lab06$
```

Рисунок 2.13: Запуск программы lab6-3.asm

В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета. (рис. 2.14) (рис. 2.15)

В данном случае число, над которым необходимо проводить арифметические операции, вводится с клавиатуры. Как отмечалось выше ввод с клавиатуры осуществляется в символьном виде и для корректной работы арифметических операций в NASM символы необходимо преобразовать в числа. Для этого может быть использована функция atoi из файла in\_out.asm.



The screenshot shows a terminal window titled 'mc [mashenkova@ubuntu]:~/work/arch-pc/lab06'. The code is as follows:

```
/home/mashenkova/ma~ant.asm  [----]  7 L:[ 1+18 19/ 25] *(316 / 384b) 0010
%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
    call atoi
    xor edx,edx
    mov ebx,20
    div ebx
    inc edx
    mov eax,rem
    call sprint
    mov eax,edx
    call iprintLF
    call quit
```

Рисунок 2.14: Программа variant.asm

```
mashenkova@ubuntu:~/work/arch-pc/lab06$  
mashenkova@ubuntu:~/work/arch-pc/lab06$ nasm -f elf variant.asm  
mashenkova@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 variant.o -o variant  
mashenkova@ubuntu:~/work/arch-pc/lab06$ ./variant  
Введите № студенческого билета:  
1132250398  
Ваш вариант: 19  
mashenkova@ubuntu:~/work/arch-pc/lab06$ █
```

Рисунок 2.15: Запуск программы variant.asm

## 2.3 Ответы на вопросы

1. Какие строки листинга отвечают за вывод на экран сообщения „Ваш вариант:“?
  - Инструкция «`mov eax, rem`» перекладывает значение переменной с фразой „Ваш вариант:“ в регистр eax.
  - Инструкция «`call sprint`» вызывает подпрограмму для вывода строки.
2. Для чего используется следующие инструкции?
  - Инструкция «`mov esx, x`» используется для перемещения значения переменной x в регистр esx.
  - Инструкция «`mov edx, 80`» используется для перемещения значения 80 в регистр edx.
  - Инструкция «`call sread`» вызывает подпрограмму для считывания значения студенческого билета из консоли
3. Для чего используется инструкция «`call atoi`»?
  - Инструкция «`call atoi`» используется для преобразования введенных символов в числовой формат.

4. Какие строки листинга отвечают за вычисления варианта?
- Инструкция «xor edx, edx» обнуляет регистр edx.
  - Инструкция «mov ebx, 20» записывает значение 20 в регистр ebx.
  - Инструкция «div ebx» выполняет деление номера студенческого билета на 20.
  - Инструкция «inc edx» увеличивает значение регистра edx на 1.
- 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»?
- Остаток от деления записывается в регистр edx.
6. Для чего используется инструкция «inc edx»?
- Инструкция «inc edx» используется для увеличения значения в регистре edx на 1, согласно формуле вычисления варианта.
7. Какие строки листинга отвечают за вывод на экран результата вычислений?
- Инструкция «mov eax, edx» перекладывает результат вычислений в регистр eax.
  - Инструкция «call iprintLF» вызывает подпрограмму для вывода значения на экран.

## **2.4 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. (рис. 2.16) (рис. 2.17)

Получили вариант 19 -  $(x/3 + 5) * 7$  для  $x = 3, x = 9$

The screenshot shows a terminal window titled 'mc [mashenkova@ubuntu]:~/work/arch-pc/lab06'. The window displays assembly code for a program named 'calc.asm'. The code includes sections for data and bss, and defines a start label. It uses system calls for input and output, and performs arithmetic operations like multiplication and division. A cursor is visible in the terminal window.

```
/home/ma~alc.asm [----] 0 L:[ 1+22 23/ 30] *(305 / 378b) 0109 [*  
%include 'in_out.asm'  
SECTION .data  
msg: DB 'Введите X ',0  
rem: DB 'выражение = : ',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  
    call atoi  
    xor edx, edx  
    mov ebx, 3  
    div ebx  
    add eax, 5  
    mov ebx, 7  
    mul ebx  
    mov ebx,eax  
    mov eax,rem  
    call sprint  
    mov eax,ebx  
    call iprintLF  
    call quit
```

Рисунок 2.16: Программа calc.asm

При  $x = 3$  получается 42.

При  $x = 9$  получается 56.

```
mashenkova@ubuntu:~/work/arch-pc/lab06$ 
mashenkova@ubuntu:~/work/arch-pc/lab06$ nasm -f elf calc.asm
mashenkova@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 calc.o -o calc
mashenkova@ubuntu:~/work/arch-pc/lab06$ ./calc
Введите X
3
выражение = : 42
mashenkova@ubuntu:~/work/arch-pc/lab06$ ./calc
Введите X
9
выражение = : 56
mashenkova@ubuntu:~/work/arch-pc/lab06$ █
```

Рисунок 2.17: Запуск программы calc.asm

Программа считает верно.

## **3 Выводы**

Изучили работу с арифметическими операциями.