

# **Отчёт по лабораторной работе 6**

**Архитектура компьютеров**

Ел Вакил Марьям Махмоудовна НБИбд-03-23

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
3.1	Символьные и численные данные в NASM . . . . .	7
3.2	Выполнение арифметических операций в NASM . . . . .	14
3.2.1	Ответы на вопросы . . . . .	19
3.3	Выполнение заданий для самостоятельной работы. . . . .	20
<b>4</b>	<b>Выводы</b>	<b>23</b>

## Список иллюстраций

3.1	Код программы lab6-1.asm . . . . .	8
3.2	Проверка программы lab6-1.asm . . . . .	9
3.3	Код программы lab6-1.asm . . . . .	10
3.4	Проверка программы lab6-1.asm . . . . .	11
3.5	Код программы lab6-2.asm . . . . .	12
3.6	Проверка программы lab6-2.asm . . . . .	12
3.7	Код программы lab6-2.asm . . . . .	13
3.8	Проверка программы lab6-2.asm . . . . .	14
3.9	Проверка программы lab6-2.asm . . . . .	14
3.10	Код программы lab6-3.asm . . . . .	15
3.11	Проверка программы lab6-3.asm . . . . .	15
3.12	Код программы lab6-3.asm . . . . .	16
3.13	Проверка программы lab6-3.asm . . . . .	17
3.14	Код программы variant.asm . . . . .	18
3.15	Проверка программы variant.asm . . . . .	19
3.16	Код программы calc.asm . . . . .	21
3.17	Проверка программы calc.asm . . . . .	22

## Список таблиц

# 1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

## 2 Задание

1. Изучить синтаксис арифметических операций в ассемблере
2. Разобрать примеры программ с символьными и численными данными
3. Разобрать примеры апрограмм с вычислениями
4. Изучить программы вычисления варианта и определить свой вариант
5. Выполнить самостоятельное задание по варианту

## 3 Выполнение лабораторной работы

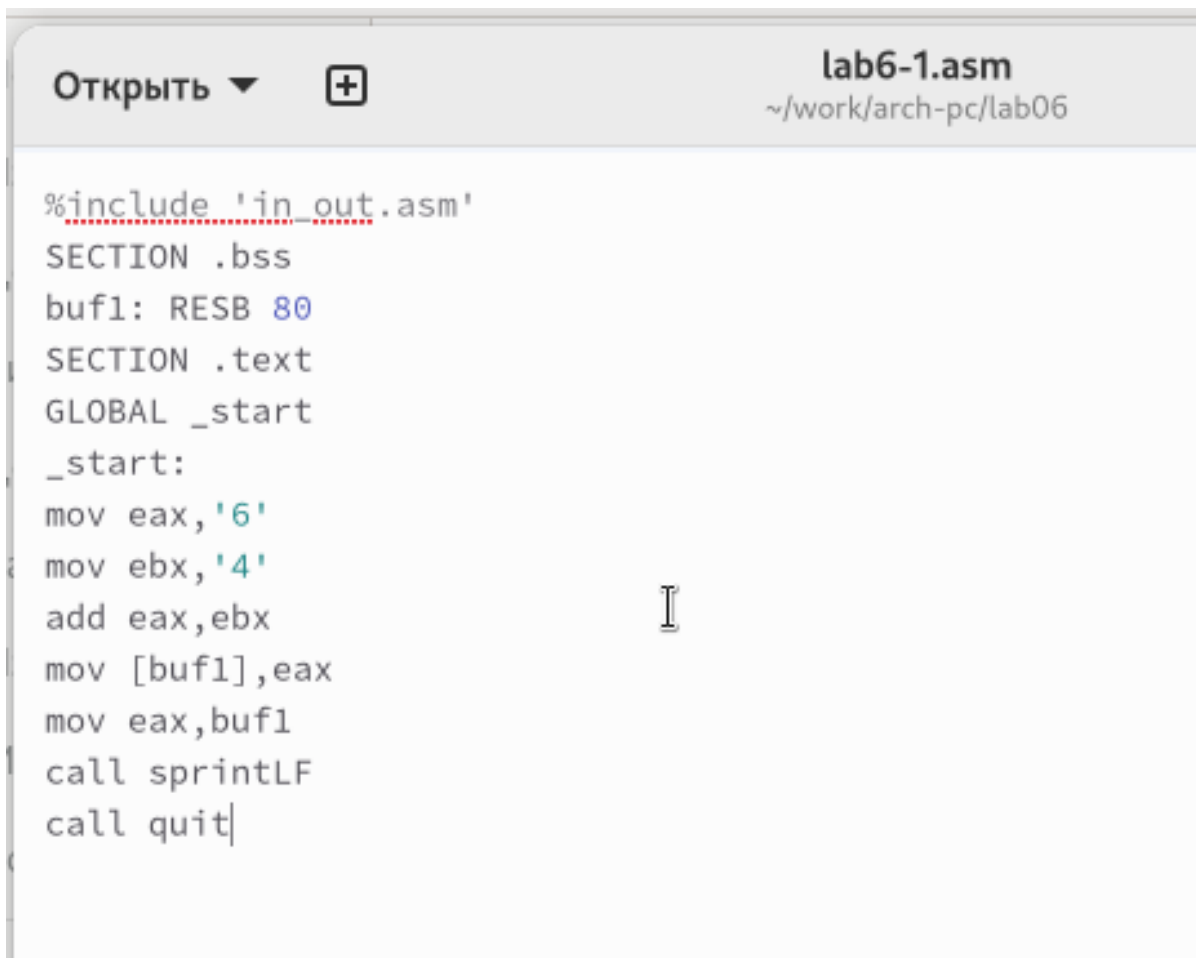
### 3.1 Символьные и численные данные в NASM

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

Давайте разберёмся с примерами программ, которые выводят символы и числа. Эти программы будут показывать значения, которые мы занесём в регистр `eax`.

В программе, которую я сейчас рассматриваю, в регистр `eax` помещается символ '6' с помощью команды `mov eax, '6'`, а в регистр `ebx` записывается символ '4' – `mov ebx, '4'`. Затем я прибавляю значение, хранящееся в регистре `ebx`, к значению регистра `eax`, используя команду `add eax, ebx`, и результат сложения сохраняется в регистре `eax`.

Для того чтобы функция `sprintf` смогла работать правильно, в регистре `eax` должен быть адрес, поэтому мне нужно использовать дополнительную переменную. Я переношу значение из регистра `eax` в переменную `buf1` командой `mov [buf1], eax`, а потом записываю адрес переменной `buf1` обратно в регистр `eax` с помощью команды `mov eax, buf1` перед тем, как вызвать функцию `sprintf`.




```
Открыть ▾  lab6-1.asm  
~/work/arch-pc/lab06  
  
%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. 3.1: Код программы lab6-1.asm

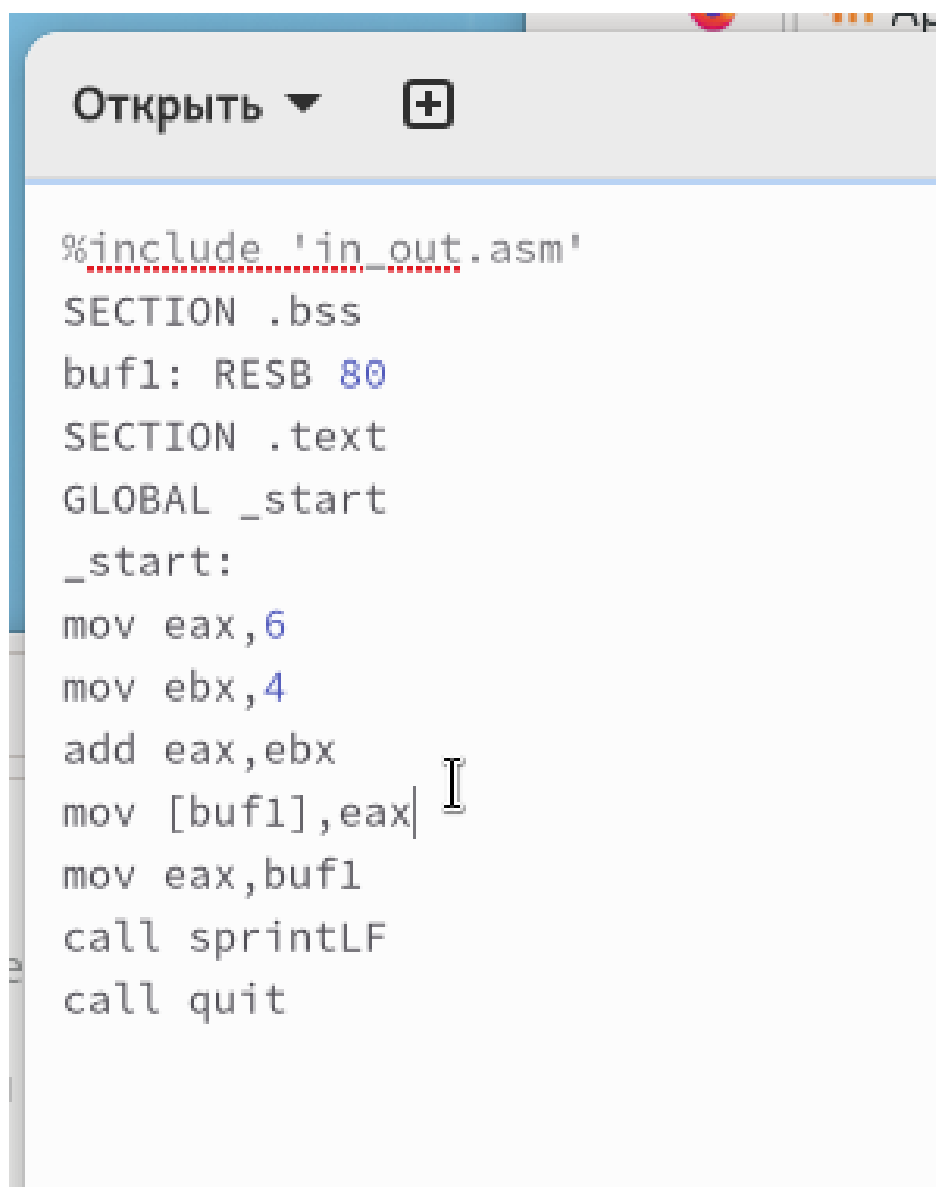
Я ожидаю увидеть на экране число 10, когда выведу значение регистра `eax`. Но вместо этого у меня отображается символ 'j'. Это происходит потому, что в двоичном коде символ '6' представлен как 00110110 (или 54 в десятичной системе), а символ '4' – как 00110100 (52 в десятичной системе). Когда я выполняю сложение командой `add eax, ebx`, в регистре `eax` оказывается сумма этих кодов – 01101010 (или 106 в десятичной системе), что соответствует коду символа 'j'.



```
[maryamelvakil@fedora lab06]$ nasm -f elf lab6-1.asm
[maryamelvakil@fedora lab06]$ ld -m elf_i386 lab6-1.o -o lab6-1
[maryamelvakil@fedora lab06]$ ./lab6-1
j
[maryamelvakil@fedora lab06]$
```

Рис. 3.2: Проверка программы lab6-1.asm

Далее изменяю текст программы и вместо символов, запишем в регистры числа.



```
%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. 3.3: Код программы lab6-1.asm

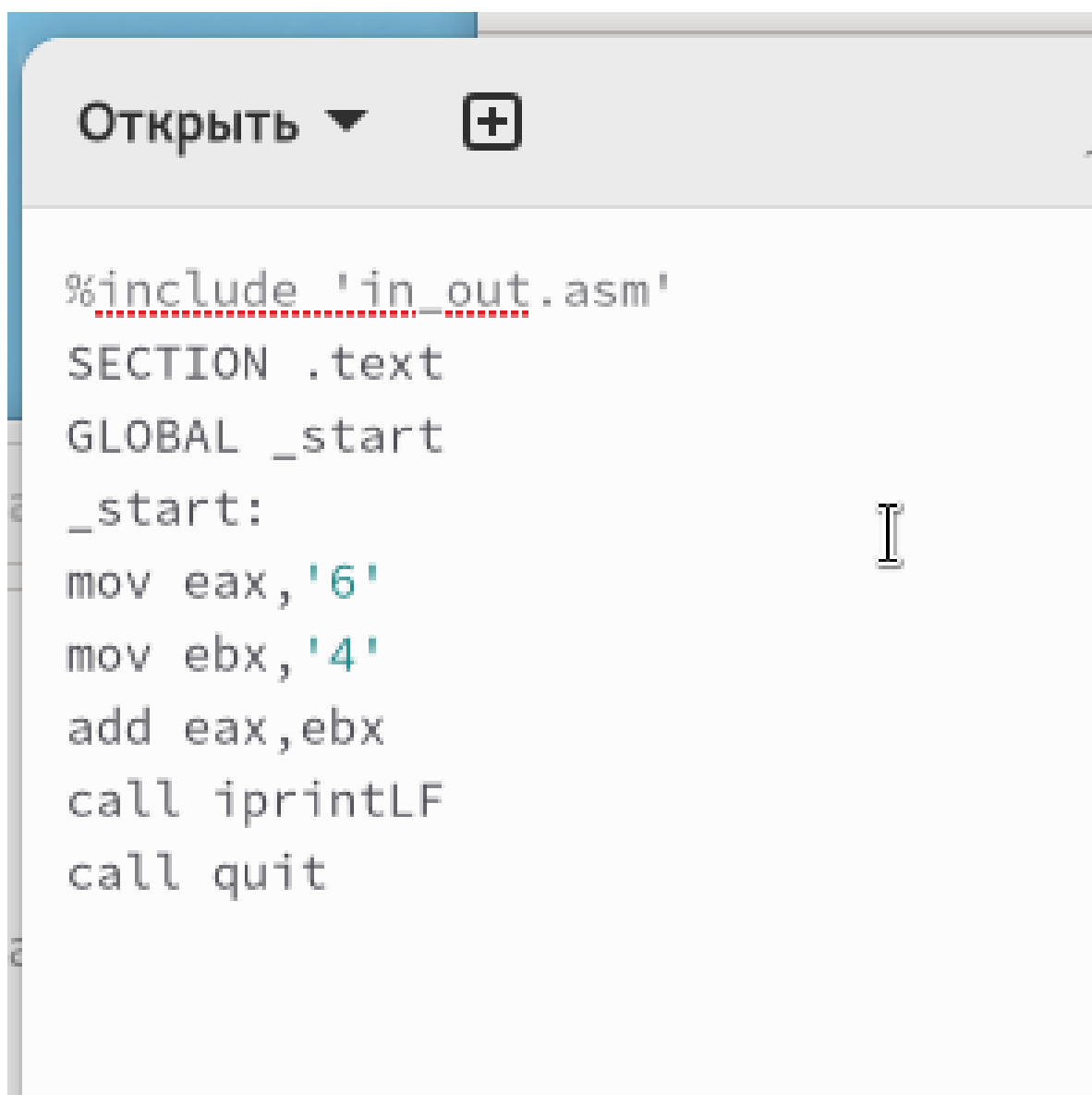
Как и раньше, при выполнении программы я не получила число 10. На этот раз на экране появился символ с кодом 10, который представляет собой символ конца строки или возврат каретки. Хотя он и не виден в консоли, он добавляет пустую строку.

```
[maryamelvakil@fedora lab06]$ nasm -f elf lab6-1.asm
[maryamelvakil@fedora lab06]$ ld -m elf_i386 lab6-1.o -o lab6-1
[maryamelvakil@fedora lab06]$ ./lab6-1

[maryamelvakil@fedora lab06]$
```

Рис. 3.4: Проверка программы lab6-1.asm

Как было сказано ранее, в файле in\_out.asm для работы с числами предусмотрены специальные подпрограммы, которые преобразуют ASCII символы в числа и наоборот. Я использовала эти функции, чтобы преобразовать текст программы.




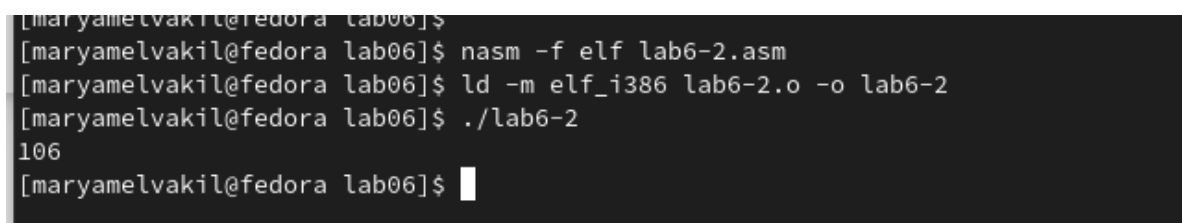
```
Открыть ▼ 
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

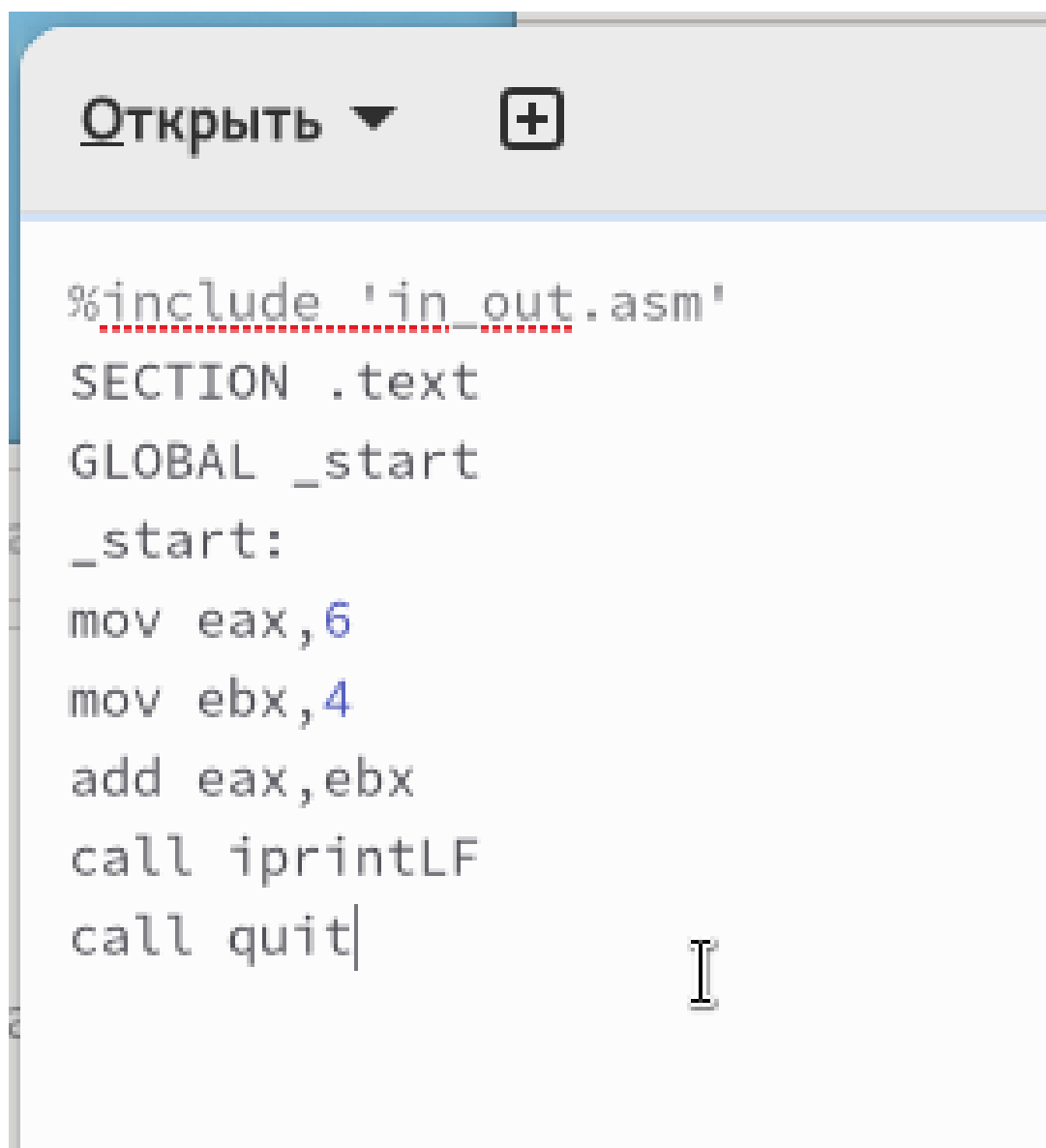
Рис. 3.5: Код программы lab6-2.asm



```
[maryameltvakil@fedora lab06]$
[maryameltvakil@fedora lab06]$ nasm -f elf lab6-2.asm
[maryameltvakil@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[maryameltvakil@fedora lab06]$ ./lab6-2
106
[maryameltvakil@fedora lab06]$
```

Рис. 3.6: Проверка программы lab6-2.asm

Так же, как и в предыдущем примере, я изменила символы на числа.



```
Открыть ▼ +  
  
%include 'in_out.asm'  
SECTION .text  
GLOBAL _start  
_start:  
mov eax,6  
mov ebx,4  
add eax,ebx  
call iprintLF  
call quit|
```

Рис. 3.7: Код программы lab6-2.asm

Благодаря функции iprintLF, которая позволяет выводить число, и тому, что операндами были именно числа, а не коды символов, я получила число 10.

```
[maryamelvakil@fedora lab06]$
[maryamelvakil@fedora lab06]$ nasm -f elf lab6-2.asm
[maryamelvakil@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[maryamelvakil@fedora lab06]$ ./lab6-2
10
[maryamelvakil@fedora lab06]$
```

Рис. 3.8: Проверка программы lab6-2.asm

Я заменила функцию `iprintLF` на `iprint`, создала исполняемый файл и запустила его. Результат отличался тем, что в выводе не было переноса строки.

```
[maryamelvakil@fedora lab06]$
[maryamelvakil@fedora lab06]$ nasm -f elf lab6-2.asm
[maryamelvakil@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[maryamelvakil@fedora lab06]$ ./lab6-2
10[maryamelvakil@fedora lab06]$
[maryamelvakil@fedora lab06]$
```

Рис. 3.9: Проверка программы lab6-2.asm

## 3.2 Выполнение арифметических операций в NASM

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

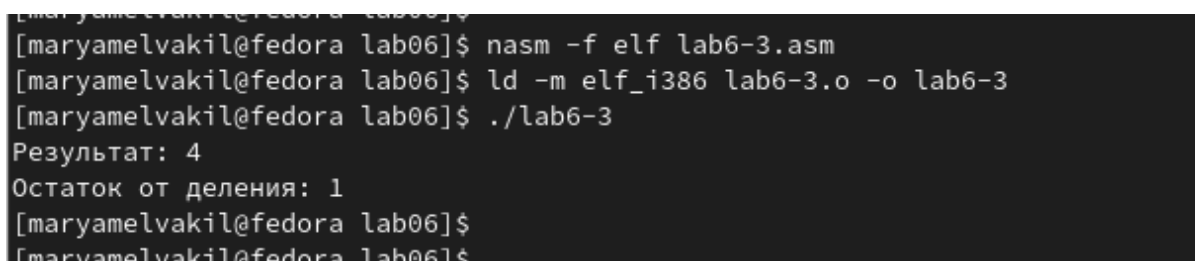


```
Открыть ▾ + lab6-3.asm
~/work/arch-pc/lab06

%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
```

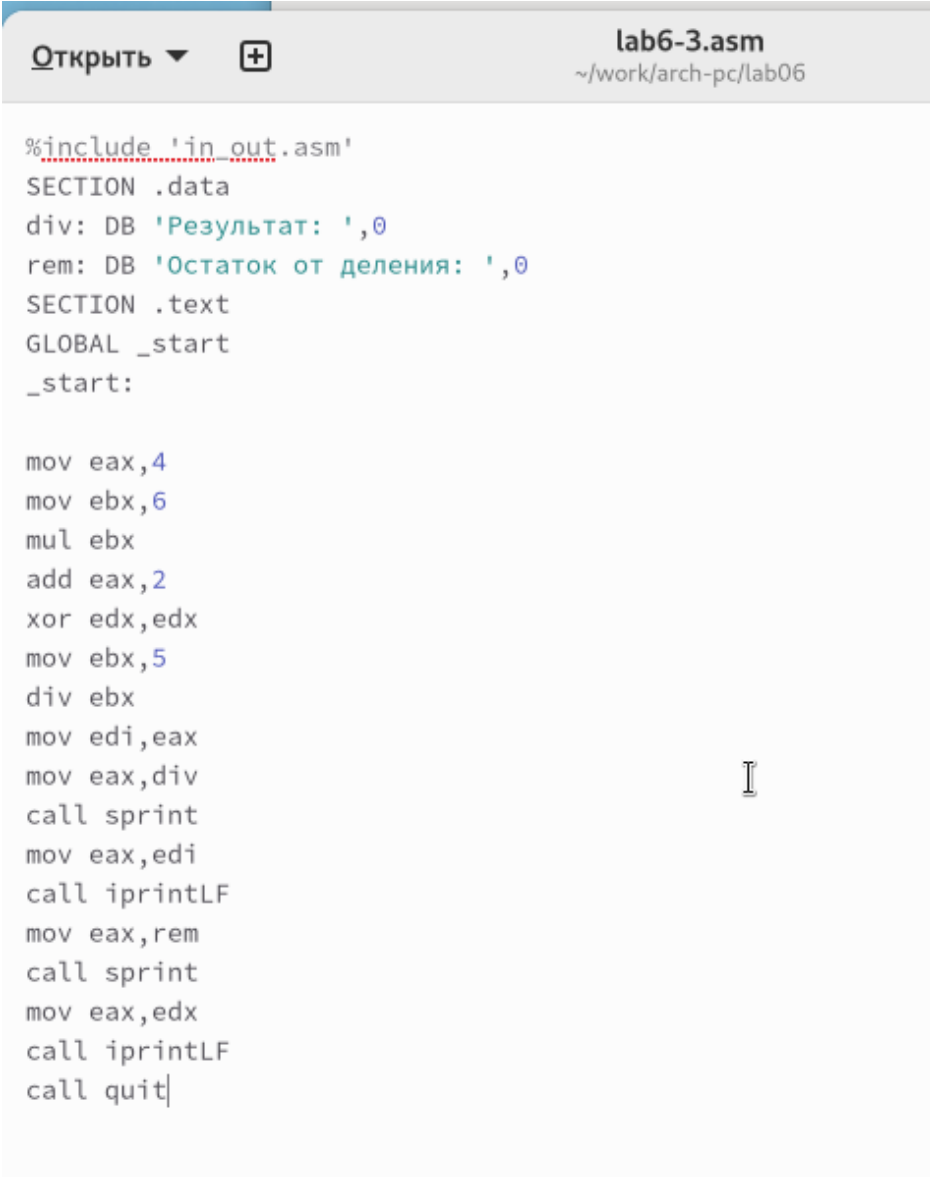
Рис. 3.10: Код программы lab6-3.asm



```
[maryamelvakil@fedora lab06]$ nasm -f elf lab6-3.asm
[maryamelvakil@fedora lab06]$ ld -m elf_i386 lab6-3.o -o lab6-3
[maryamelvakil@fedora lab06]$ ./lab6-3
Результат: 4
Остаток от деления: 1
[maryamelvakil@fedora lab06]$
[maryamelvakil@fedora lab06]$
```

Рис. 3.11: Проверка программы lab6-3.asm

Изменила текст программы для вычисления выражения  $f(x) = (4 * 6 + 2)/5$ .  
Создала исполняемый файл и проверила его работу.



```
lab6-3.asm
~/work/arch-pc/lab06

%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
```

Рис. 3.12: Код программы lab6-3.asm



```
[maryam@fedora lab06]$ nasm -f elf lab6-3.asm
[maryam@fedora lab06]$ ld -m elf_i386 lab6-3.o -o lab6-3
[maryam@fedora lab06]$ ./lab6-3
Результат: 5
Остаток от деления: 1
[maryam@fedora lab06]$
```

Рис. 3.13: Проверка программы lab6-3.asm

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

В этом случае, число, с которым предстоит работать, мы вводим с помощью клавиатуры. Ранее я уже упоминала, что ввод осуществляется в символьном формате, и чтобы арифметические операции выполнялись правильно в NASM, эти символы нужно конвертировать в числовой формат. Сделать это можно с помощью функции `atoi`, которая находится в файле `in_out.asm`.

Открыть ▾ 

variant.asm  
~/work/arch-pc/lab06

```
%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
```

Рис. 3.14: Код программы variant.asm

```
[maryamelvakil@fedora lab06]$
[maryamelvakil@fedora lab06]$ nasm -f elf variant.asm
[maryamelvakil@fedora lab06]$ ld -m elf_i386 variant.o -o variant
[maryamelvakil@fedora lab06]$ ./variant
Введите № студенческого билета:
1032230325
Ваш вариант: 6
[maryamelvakil@fedora lab06]$
```

Рис. 3.15: Проверка программы variant.asm

### 3.2.1 Ответы на вопросы

1. Какие строки листинга отвечают за вывод на экран сообщения ‘Ваш вариант:’?

Команда “mov eax, rem” загружает в регистр eax строку с текстом “Ваш вариант:”.

Команда “call sprint” инициирует вывод строки на экран.

2. Для чего используются следующие инструкции?

Команда “mov ecx, x” копирует значение из переменной x в регистр ecx.

Команда “mov edx, 80” помещает число 80 в регистр edx.

Команда “call sread” активирует функцию для ввода данных студенческого билета с клавиатуры.

3. Для чего используется инструкция “call atoi”?

Команда “call atoi” преобразует введенные символы в целое число.

4. Какие строки листинга отвечают за вычисления варианта?

Команда “xor edx, edx” очищает регистр edx.

Команда “mov ebx, 20” помещает число 20 в регистр ebx.

Команда “div ebx” выполняет деление номера студенческого на 20.

Команда “inc edx” прибавляет единицу к значению в регистре edx.

При этом выполняется деление номера студенческого билета на 20, а остаток от деления, хранящийся в регистре edx, увеличивается на 1.

5. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”?

Остаток от деления помещается в регистр edx.

6. Для чего используется инструкция “inc edx”?

Команда “inc edx” увеличивает на единицу значение в регистре edx, что необходимо для расчёта варианта по формуле.

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

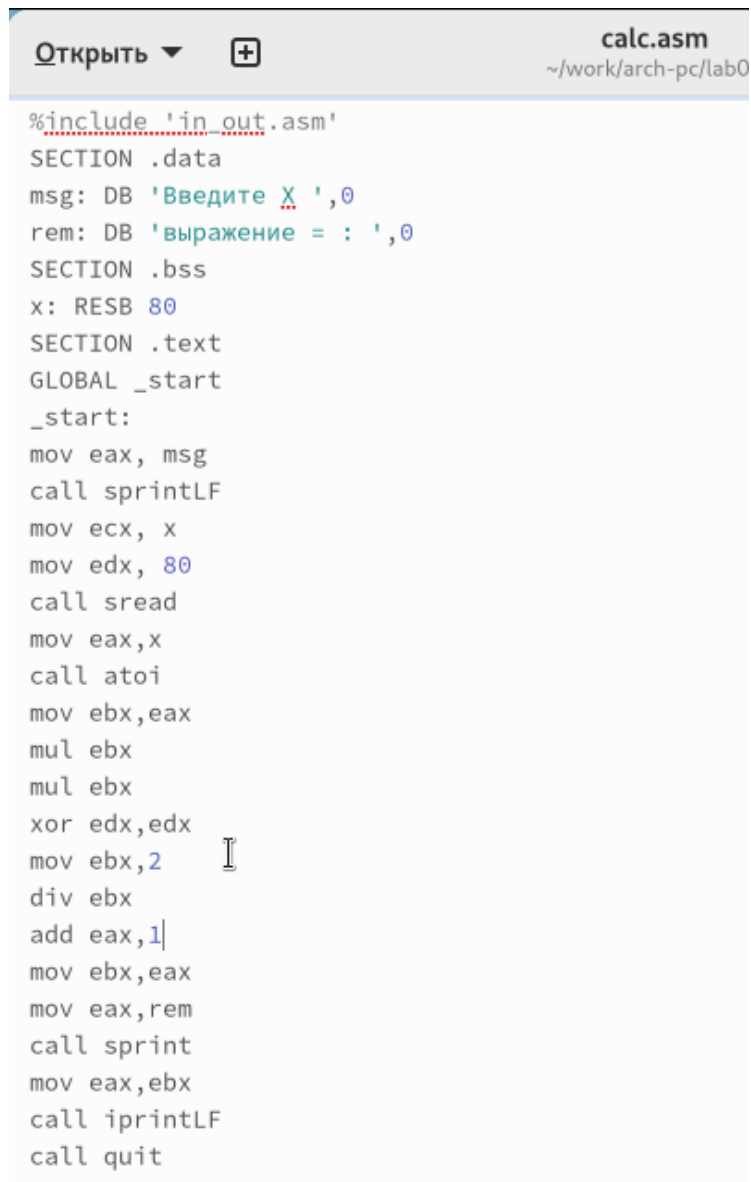
Команда “mov eax, edx” переносит результат вычислений в регистр eax.

Команда “call iprintLF” запускает функцию, которая выводит результат на экран.

### 3.3 Выполнение заданий для самостоятельной работы.

Написать программу вычисления выражения  $y = f(x)$ . Код программы 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  $8 - x^3/2 + 1$  для  $x = 2, x = 5$



```
%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
mov ebx, eax
mul ebx
mul ebx
xor edx, edx
mov ebx, 2
div ebx
add eax, 1
mov ebx, eax
mov eax, rem
call sprint
mov eax, ebx
call iprintLF
call quit
```

Рис. 3.16: Код программы calc.asm

При  $x = 2$  получается 5.

При  $x = 5$  получается 63.5. (программа выводит 63, так как деление целочисленное)

```
[maryamelvakil@fedora lab06]$ nasm -f elf calc.asm
[maryamelvakil@fedora lab06]$ ld -m elf_i386 calc.o -o calc
[maryamelvakil@fedora lab06]$ ./calc
Введите X
2
выражение = : 5
[maryamelvakil@fedora lab06]$ ./calc
Введите X
5
выражение = : 63
[maryamelvakil@fedora lab06]$
```

Рис. 3.17: Проверка программы calc.asm

Код программы считает верно.

## **4 Выводы**

Изучили работу с арифметическими операциями.