

# **Отчёт по лабораторной работе 6**

**Дисциплина архитектура компьютера**

Неустроева Ирина

# 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
2.2.1	Ответы на вопросы по программе variant.asm . . . . .	17
2.3	Выполнение заданий для самостоятельной работы . . . . .	18
<b>3</b>	<b>Выводы</b>	<b>21</b>

## Список иллюстраций

2.1	Редактирование файла lab6-1.asm . . . . .	7
2.2	Проверка кода lab6-1.asm . . . . .	7
2.3	Редактирование файла lab6-1.asm . . . . .	8
2.4	Проверка кода lab6-1.asm . . . . .	8
2.5	Редактирование файла lab6-2.asm . . . . .	9
2.6	Проверка кода lab6-2.asm . . . . .	9
2.7	Редактирование файла lab6-2.asm . . . . .	10
2.8	Проверка кода lab6-2.asm . . . . .	11
2.9	Редактирование файла lab6-2.asm . . . . .	11
2.10	Проверка кода lab6-2.asm . . . . .	12
2.11	Редактирование файла lab6-3.asm . . . . .	13
2.12	Проверка кода lab6-3.asm . . . . .	13
2.13	Редактирование файла lab6-3.asm . . . . .	14
2.14	Проверка кода lab6-3.asm . . . . .	15
2.15	Редактирование файла variant.asm . . . . .	16
2.16	Проверка кода variant.asm . . . . .	16
2.17	Редактирование файла calc.asm . . . . .	19
2.18	Проверка кода calc.asm . . . . .	20

## Список таблиц

# 1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

## 2 Выполнение лабораторной работы

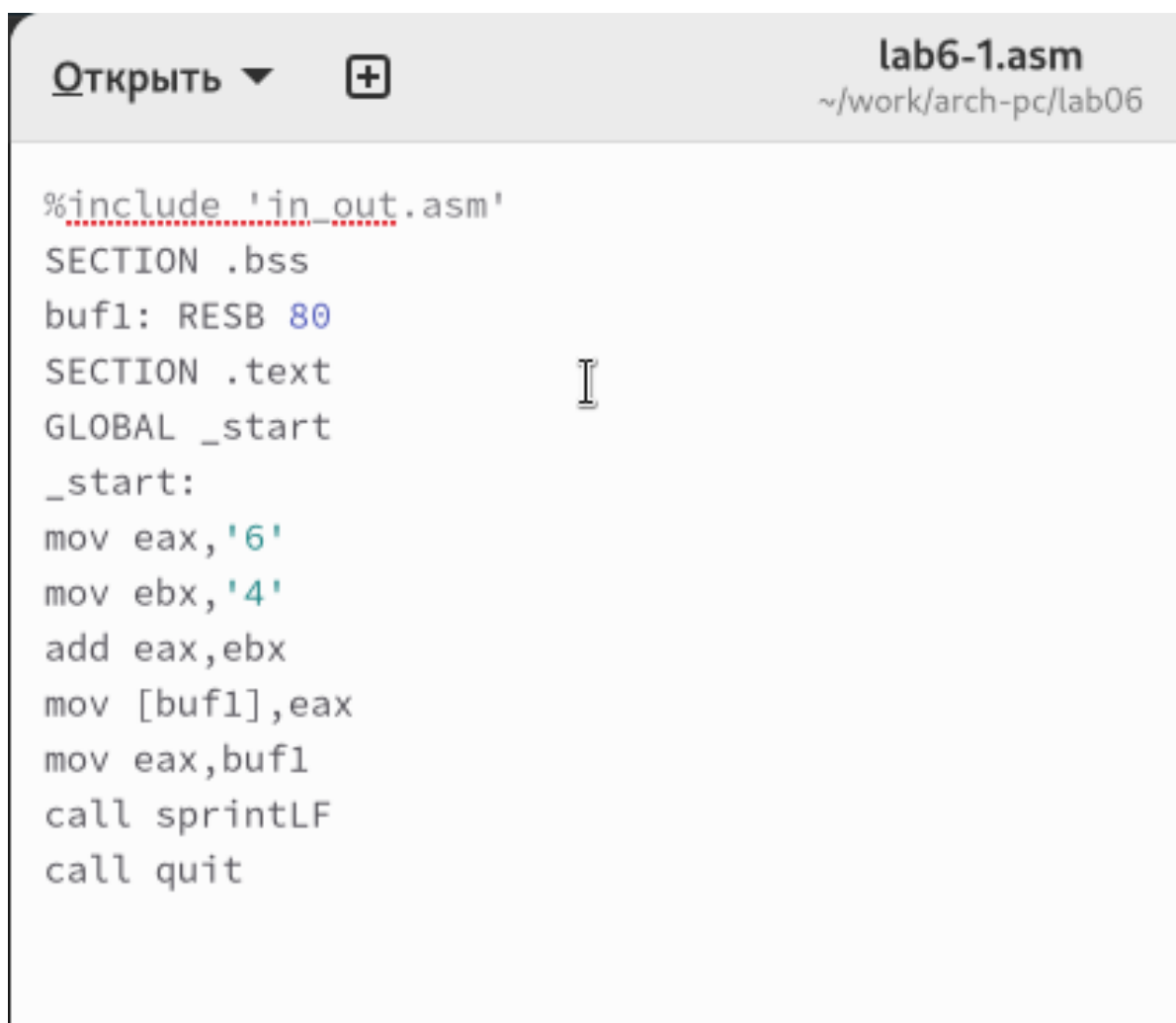
### 2.1 Символьные и численные данные в NASM

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

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

В данной программе, в регистр `eax` записан символ ‘6’, а в регистр `ebx` символ ‘4’. Затем мы прибавляем значение регистра `ebx` к значению в регистре `eax` (результат сложения будет записан в регистр `eax`). После этого мы выводим результат.

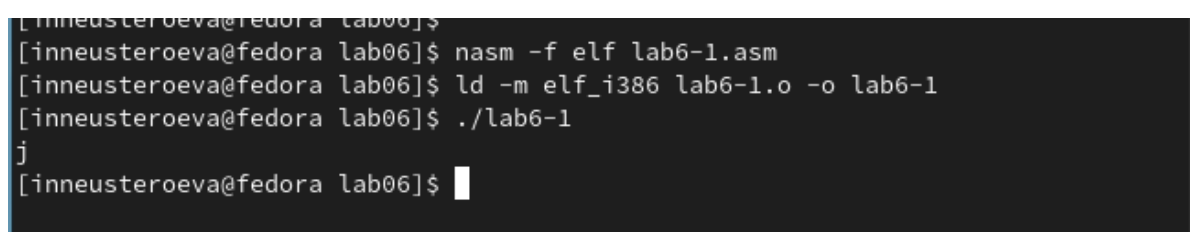
Так как для работы функции `sprintLF` в регистр `eax` должен быть записан адрес, мы используем дополнительную переменную. Мы записали значение регистра `eax` в переменную с именем “`buf1`”, а затем записали адрес переменной `buf1` в регистр `eax` и вызвали функцию `sprintLF`. (рис. [2.1]) (рис. [2.2])



```
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. 2.1: Редактирование файла lab6-1.asm



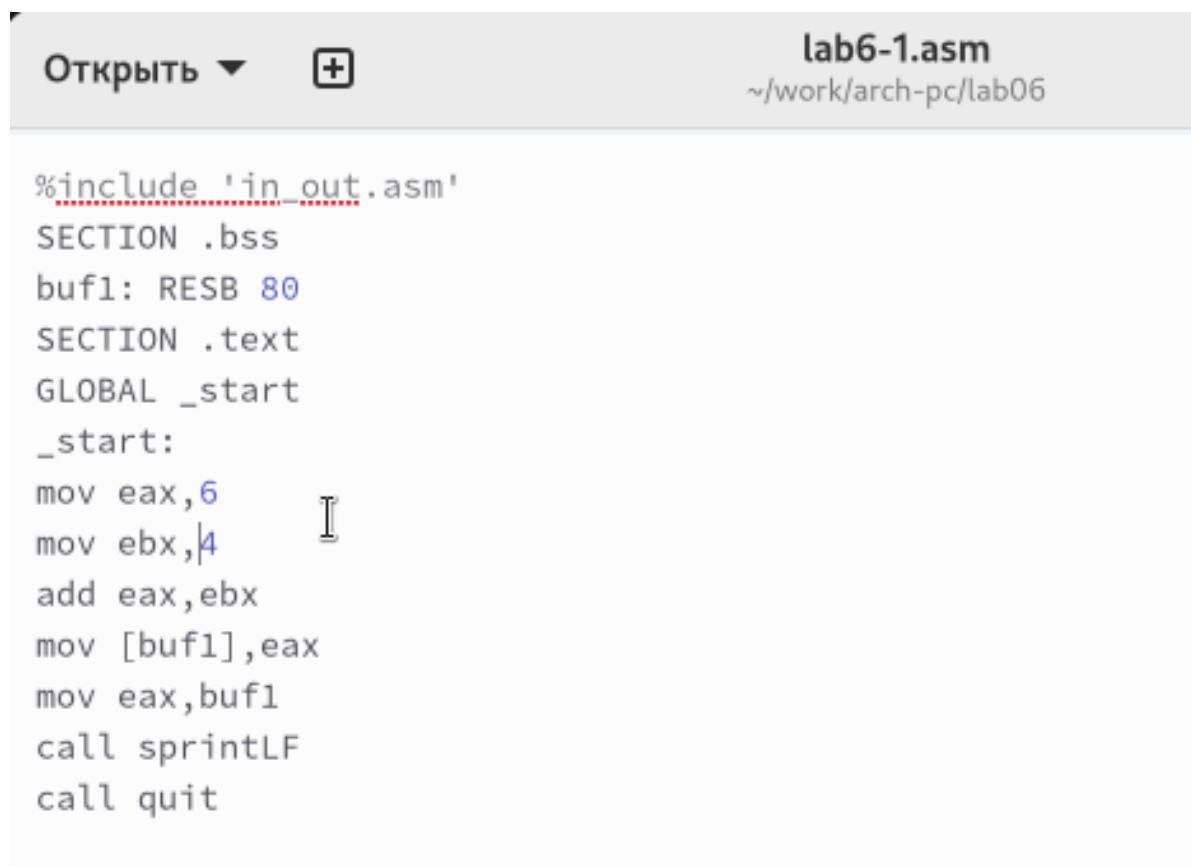
```
[inneusteroeva@fedora lab06]$
[inneusteroeva@fedora lab06]$ nasm -f elf lab6-1.asm
[inneusteroeva@fedora lab06]$ ld -m elf_i386 lab6-1.o -o lab6-1
[inneusteroeva@fedora lab06]$ ./lab6-1
j
[inneusteroeva@fedora lab06]$
```

Рис. 2.2: Проверка кода lab6-1.asm

В данном случае, при выводе значения регистра `eax`, ожидалось увидеть число 10. Однако, результатом был символ 'j'. Это произошло 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 записала в регистр eax сумму кодов – 01101010 (106), что в свою очередь является кодом символа 'j'.

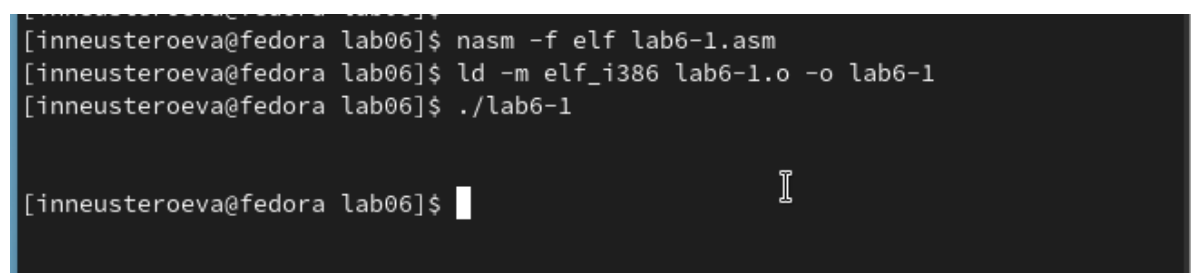
3 Далее был изменен текст программы и вместо символов записаны числа. (рис. [2.3]) (рис. [2.4])



```
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. 2.3: Редактирование файла lab6-1.asm



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

[inneusteroeva@fedora lab06]$
```

Рис. 2.4: Проверка кода lab6-1.asm



В процессе выполнения программы не получили ожидаемое число 10. Вместо этого был выведен символ с кодом 10. Это символ конца строки (возврат каретки), который в консоли не отображается, но добавляет пустую строку.

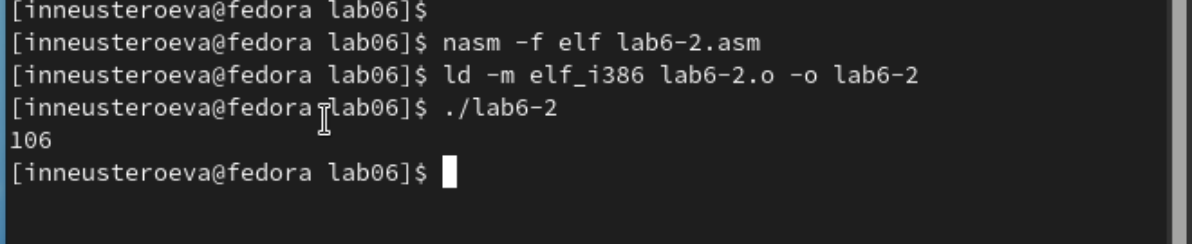
4 В файле “in\_out.asm” реализованы подпрограммы для работы с числами и преобразования символов ASCII. Был модифицирован текст программы с использованием этих функций. (рис. [2.5]) (рис. [2.6])



```
Открыть ▾ + lab6-2.asm
~/work/arch-pc/lab06

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

Рис. 2.5: Редактирование файла lab6-2.asm

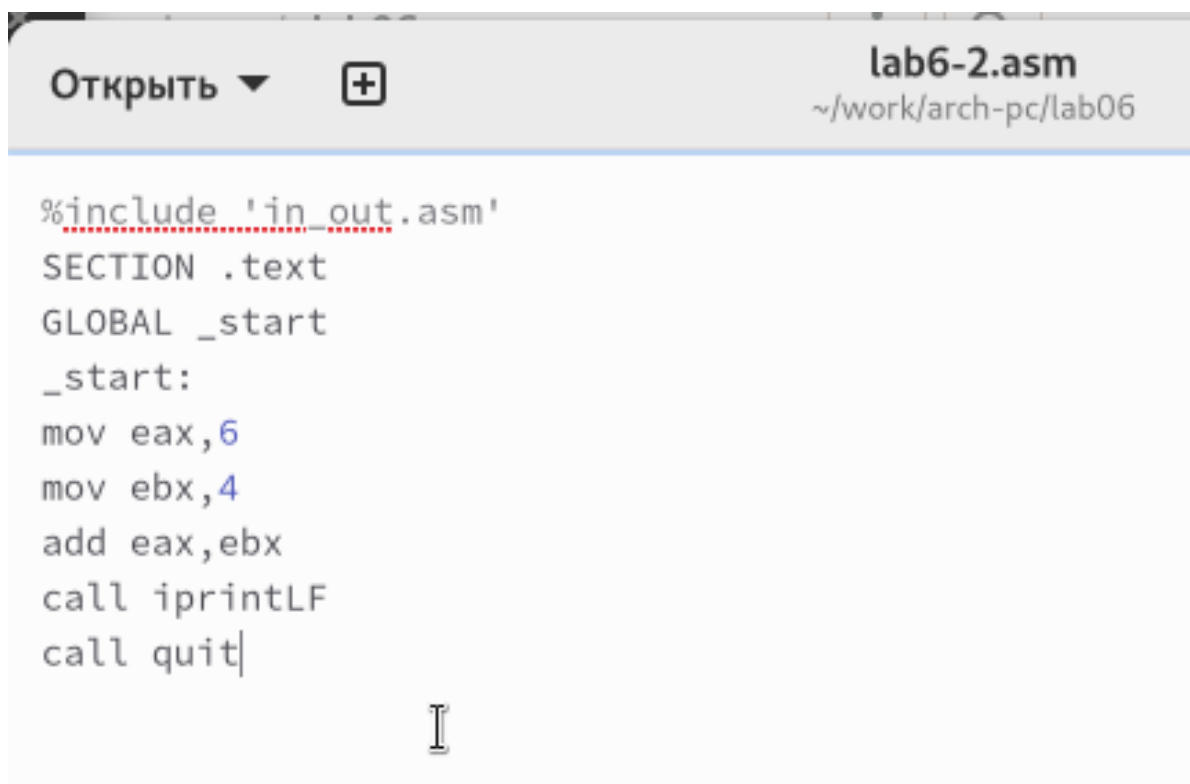


```
[inneusteroeva@fedora lab06]$
[inneusteroeva@fedora lab06]$ nasm -f elf lab6-2.asm
[inneusteroeva@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[inneusteroeva@fedora lab06]$ ./lab6-2
106
[inneusteroeva@fedora lab06]$
```

Рис. 2.6: Проверка кода lab6-2.asm

В результате выполнения обновленной программы было выведено число 106. Здесь, как и в первом случае, команда `add` складывает коды символов '6' и '4' ( $54 + 52 = 106$ ). Но в отличие от предыдущей версии, функция `iprintLF` позволяет напечатать само число, а не символ с соответствующим кодом.

5 По аналогии с предыдущим примером, были заменены символы на числа. (рис. [2.7]) (рис. [2.8])



```
Открыть ▾ + lab6-2.asm
~/work/arch-pc/lab06

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

Рис. 2.7: Редактирование файла lab6-2.asm

Функция `iprintLF` позволяет выводить числа, и на этот раз в качестве операндов использовались именно числа, а не коды символов. В результате мы получили число 10.

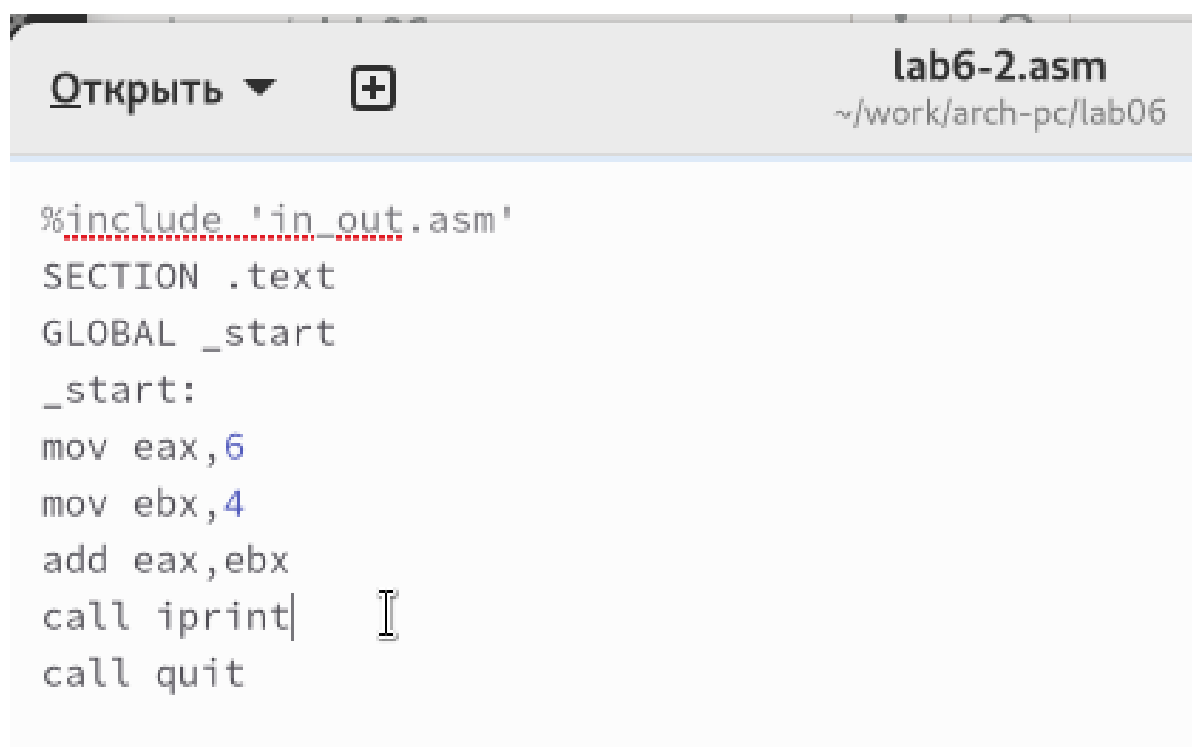
```

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

```

Рис. 2.8: Проверка кода lab6-2.asm

Далее была заменена функция `iprintLF` на `iprint`, создан исполняемый файл и запущен. Вывод теперь отличается отсутствием перехода на новую строку. (рис. [2.9]) (рис. [2.10])



```

lab6-2.asm
~/work/arch-pc/lab06

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

```

Рис. 2.9: Редактирование файла lab6-2.asm

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

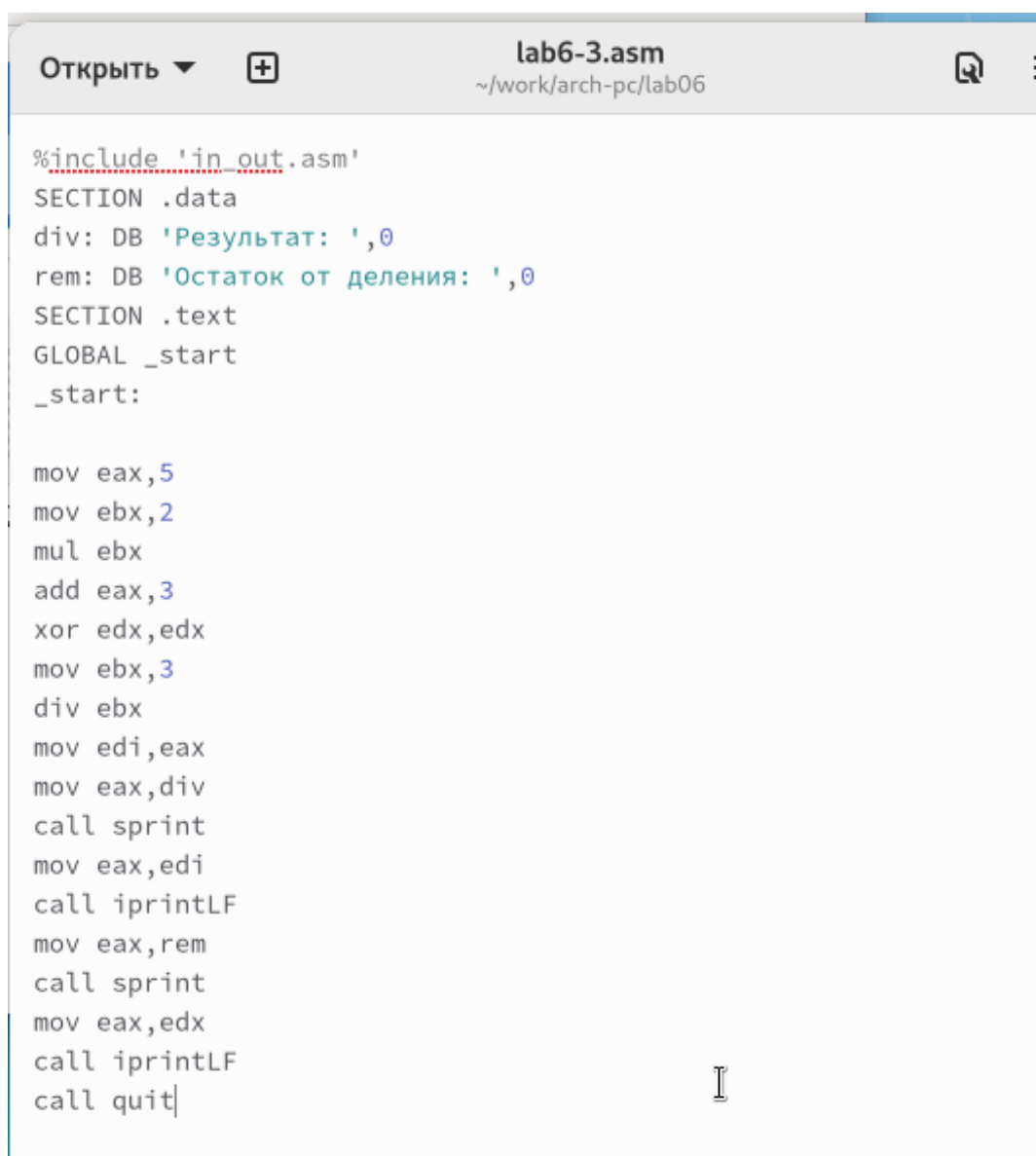
Рис. 2.10: Проверка кода lab6-2.asm

## 2.2 Выполнение арифметических операций в NASM

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

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

. (рис. [2.11]) (рис. [2.12])

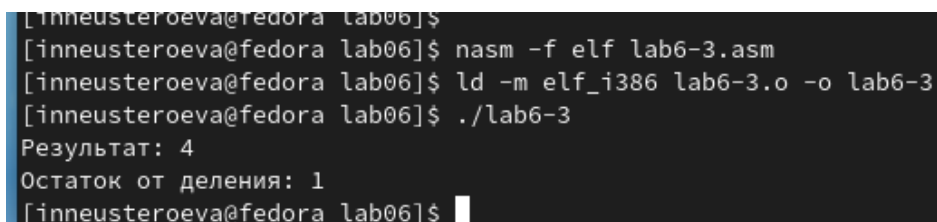


```
Открыть ▾ + lab6-3.asm ~/work/arch-pc/lab06

%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. 2.11: Редактирование файла lab6-3.asm



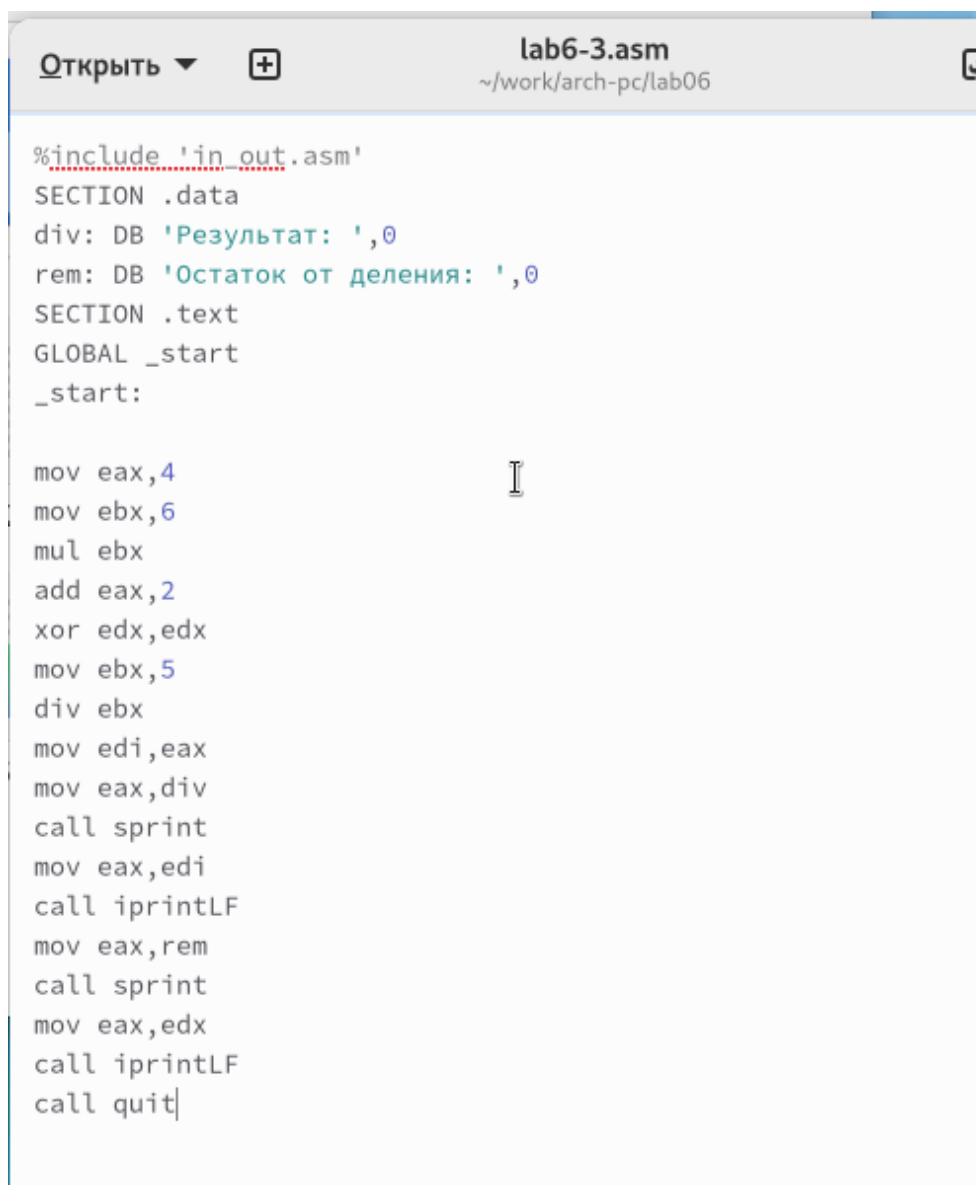
```
[inneusteroeva@fedora lab06]$
[inneusteroeva@fedora lab06]$ nasm -f elf lab6-3.asm
[inneusteroeva@fedora lab06]$ ld -m elf_i386 lab6-3.o -o lab6-3
[inneusteroeva@fedora lab06]$ ./lab6-3
Результат: 4
Остаток от деления: 1
[inneusteroeva@fedora lab06]$
```

Рис. 2.12: Проверка кода lab6-3.asm

Изменила текст программы для вычисления выражения

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

. Создала исполняемый файл и проверила его работу. (рис. [2.13]) (рис. [2.14])



```
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,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
```

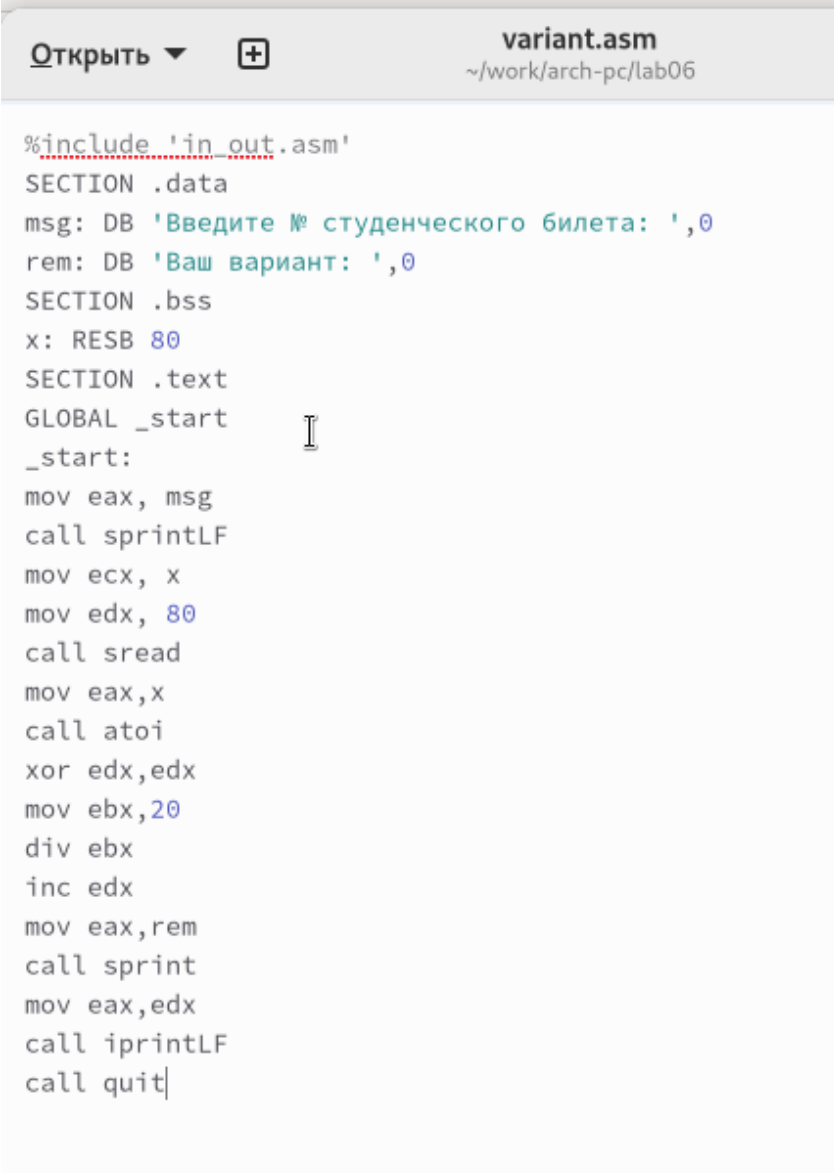
Рис. 2.13: Редактирование файла lab6-3.asm

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

Рис. 2.14: Проверка кода lab6-3.asm

7 В качестве еще одного примера рассмотрим программу для вычисления варианта задания на основе номера студенческого билета.

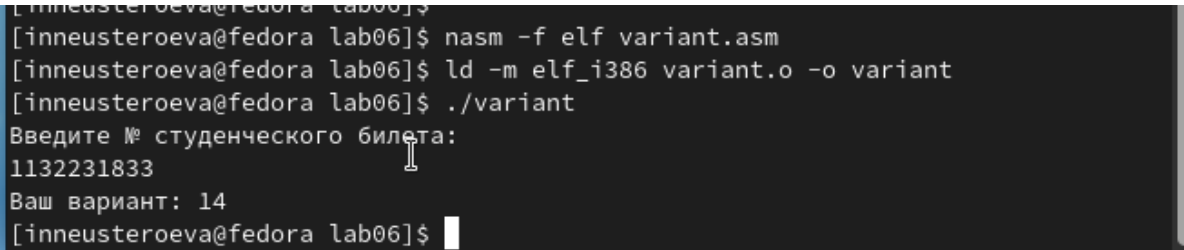
В этом случае число, над которым нужно выполнять арифметические операции, вводится с клавиатуры. Как уже отмечалось ранее, ввод с клавиатуры осуществляется в символьном виде. Для корректной работы арифметических операций в NASM эти символы необходимо преобразовать в числовой формат. С этой целью можно использовать функцию `atoi` из файла `in_out.asm`. Она конвертирует строку символов в эквивалентное десятичное число. (рис. [2.15]) (рис. [2.16])



```
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. 2.15: Редактирование файла variant.asm



```
[inneusteroeva@fedora lab06]$ nasm -f elf variant.asm
[inneusteroeva@fedora lab06]$ ld -m elf_i386 variant.o -o variant
[inneusteroeva@fedora lab06]$ ./variant
Введите № студенческого билета:
1132231833
Ваш вариант: 14
[inneusteroeva@fedora lab06]$
```

Рис. 2.16: Проверка кода variant.asm



### 2.2.1 Ответы на вопросы по программе `variant.asm`

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

За отображение фразы “Ваш вариант:” отвечают те строки кода, где содержится команда `mov eax, hex` для передачи строки в регистр `eax` и последующий вызов процедуры `sprint` через `call sprint`.

2. Для чего используются следующие инструкции?

- `mov ecx, x`: Здесь значение регистра `ecx` копируется в переменную `x`.
- `mov edx, 80`: Эта команда присваивает регистру `edx` число 80.
- `call sread`: С помощью этой команды происходит вызов функции чтения данных из стандартного ввода.

3. Для чего используется инструкция “`call atoi`”?

Использование “`call atoi`” предназначено для конвертации строковых данных в целочисленное значение.

4. Какие строки листинга отвечают за вычисления варианта?

Расчет варианта осуществляется в строках с инструкциями:

- `xor edx, edx`: Обнуляет регистр `edx`.
- `mov ebx, 20`: Задает регистру `ebx` значение 20.
- `div ebx`: Выполняет деление с сохранением остатка.
- `inc edx`: Увеличивает значение регистра `edx`, что требуется для получения конечного варианта.

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

Остаток от деления при выполнении “div ebx” записывается в регистр edx.

6. Для чего используется инструкция “inc edx”?

Команда “inc edx” применяется для увеличения содержимого регистра edx, что используется в расчетах варианта.

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

Вывод результатов вычислений на экран осуществляется строками с командами:

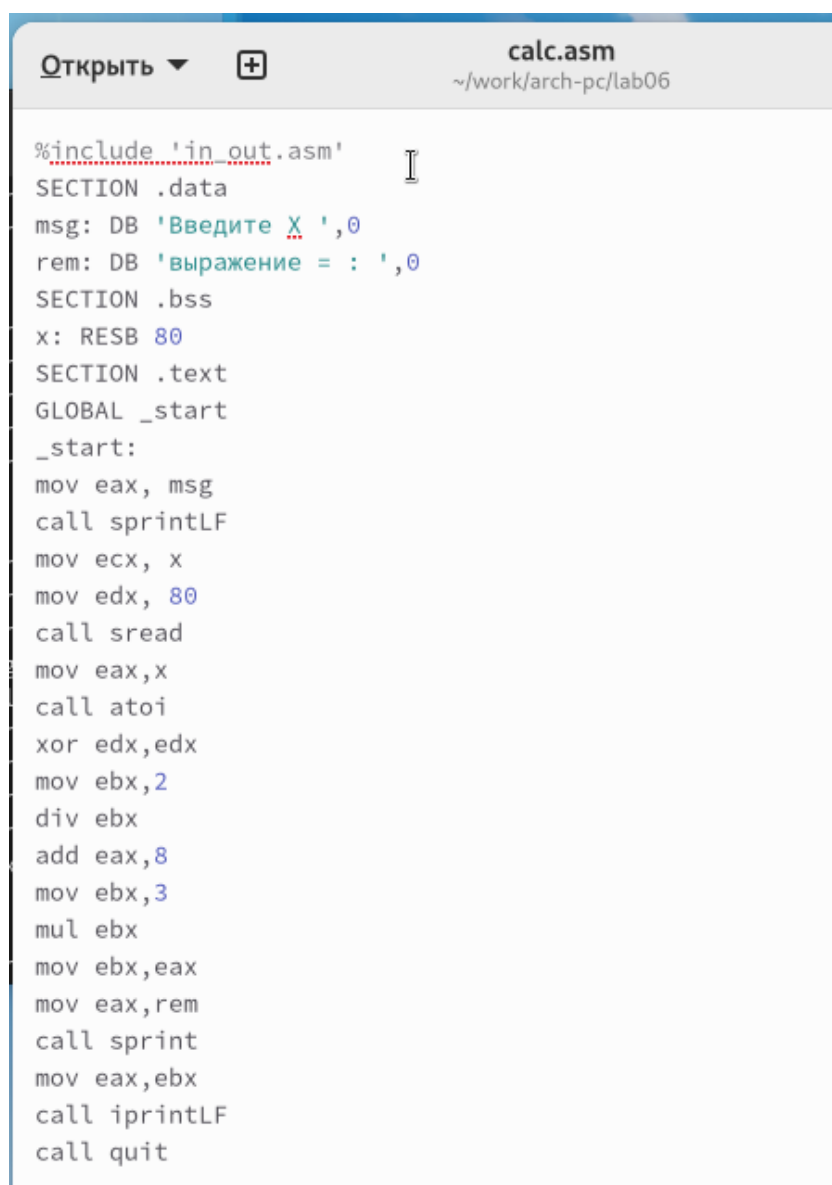
- mov eax, edx: Помещает результат вычисления в регистр eax.
- call iprintLF: Вызывает функцию для отображения результата на экране.

## 2.3 Выполнение заданий для самостоятельной работы

1 Написала программу для вычисления выражения  $y = f(x)$ . Программа выводит выражение для вычисления, запрашивает ввод значения  $x$ , вычисляет заданное выражение в зависимости от введенного  $x$  и выводит результат вычислений. Для выбора вида функции  $f(x)$  использовала таблицу 6.3 вариантов заданий, в соответствии с номером, полученным при выполнении лабораторной работы.

Создала исполняемый файл и проверила его работу для значений  $x_1$  и  $x_2$  из таблицы 6.3. (рис. [2.17]) (рис. [2.18])

Вариант 14 -  $(x/2 + 8) * 3$  для  $x = 1, x = 4$



```
%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,2
div ebx
add eax,8
mov ebx,3
mul ebx
mov ebx,eax
mov eax,rem
call sprint
mov eax,ebx
call iprintLF
call quit
```

Рис. 2.17: Редактирование файла calc.asm

```
[inneusteroeva@fedora lab06]$  
[inneusteroeva@fedora lab06]$ nasm -f elf calc.asm  
[inneusteroeva@fedora lab06]$ ld -m elf_i386 calc.o -o calc  
[inneusteroeva@fedora lab06]$ ./calc  
Введите X  
1  
выражение = : 24  
[inneusteroeva@fedora lab06]$ ./calc  
Введите X  
4  
выражение = : 30  
[inneusteroeva@fedora lab06]$
```

Рис. 2.18: Проверка кода calc.asm

Программа считает верно.

## **3 Выводы**

Изучили работу с арифметическими операциями.