

# **Отчёт по лабораторной работе 6**

**дисциплина: Архитектура компьютера**

Харламова Арина Александровна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Ответы на вопросы . . . . .	17
2.2	Задание по варианту . . . . .	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 . . . . .	10
2.6	Запуск программы lab6-2.asm . . . . .	10
2.7	Программа в файле lab6-2.asm . . . . .	11
2.8	Запуск программы lab6-2.asm . . . . .	12
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 . . . . .	17
2.16	Программа в файле work.asm . . . . .	19
2.17	Запуск программы work.asm . . . . .	20

## **Список таблиц**

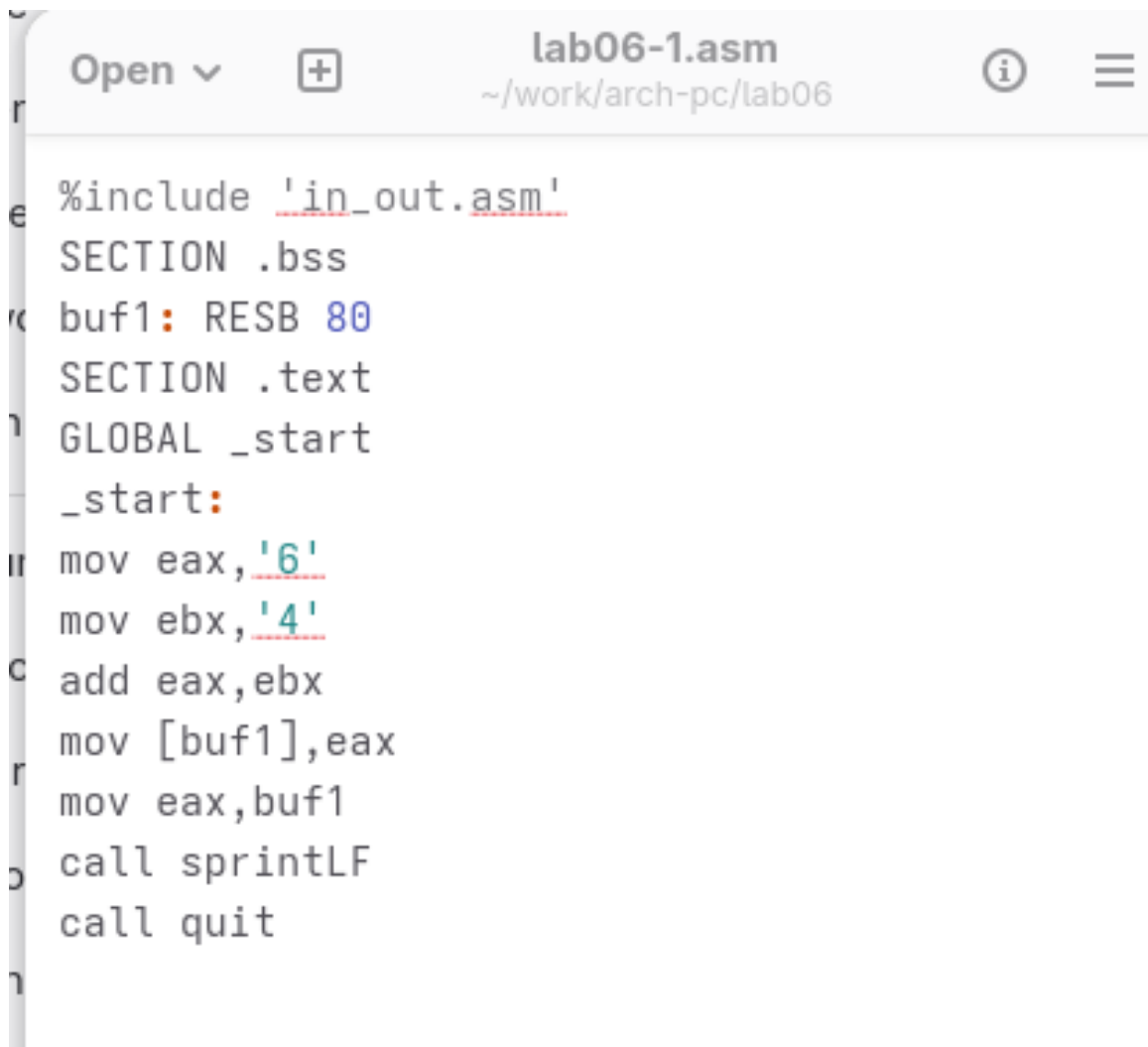
# 1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

## 2 Выполнение лабораторной работы

1. Создала каталог для программ лабораторной работы № 6, перешла в него и создала файл lab6-1.asm.
2. Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения, записанные в регистр еах.

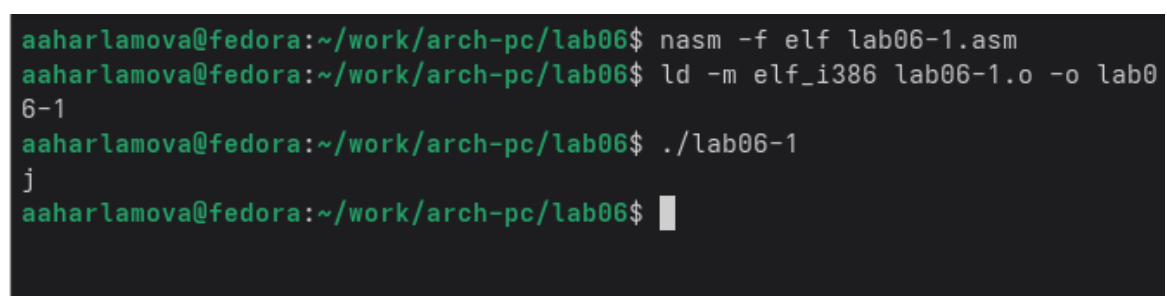
В данной программе в регистр еах записывается символ 6 (`mov еах,„6“`), в регистр ебх символ 4 (`mov ебх,„4“`). Далее к значению в регистре еах прибавляем значение регистра ебх (`add еах,ебх`, результат сложения запишется в регистр еах). Далее выводим результат. Так как для работы функции `sprintf` в регистр еах должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра еах в переменную `buf1` (`mov [buf1],еах`), а затем запишем адрес переменной `buf1` в регистр еах (`mov еах,buf1`) и вызовем функцию `sprintf`.



```
Open ▾ [ + ] lab06-1.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 sprintf
call quit
```

Рисунок 2.1: Программа в файле lab6-1.asm



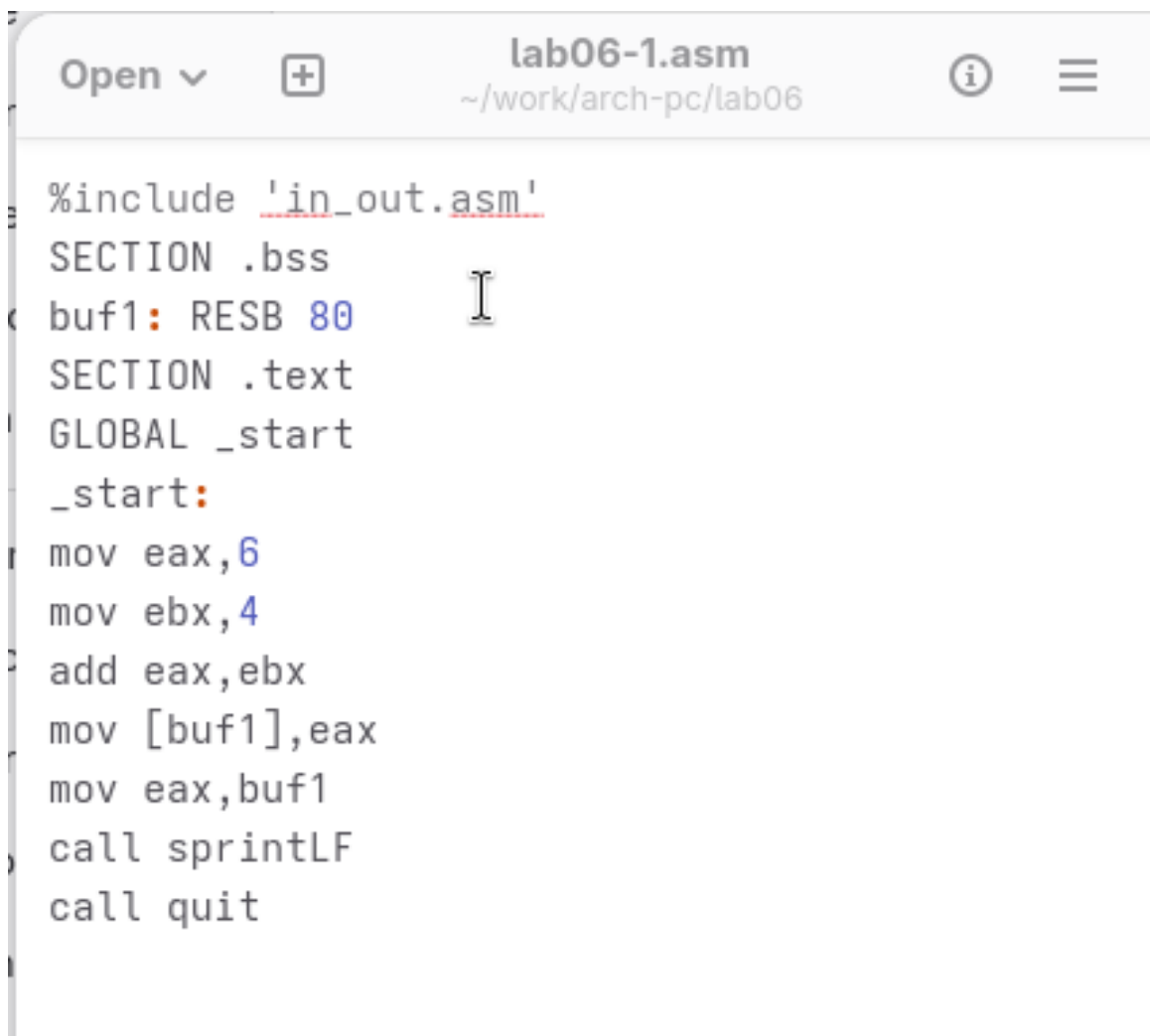
```
aaharlamova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm
aaharlamova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o -o lab06-1
aaharlamova@fedora:~/work/arch-pc/lab06$ ./lab06-1
10
aaharlamova@fedora:~/work/arch-pc/lab06$
```

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.

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



```
lab06-1.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 sprintf
call quit
```

Рисунок 2.3: Программа в файле lab6-1.asm



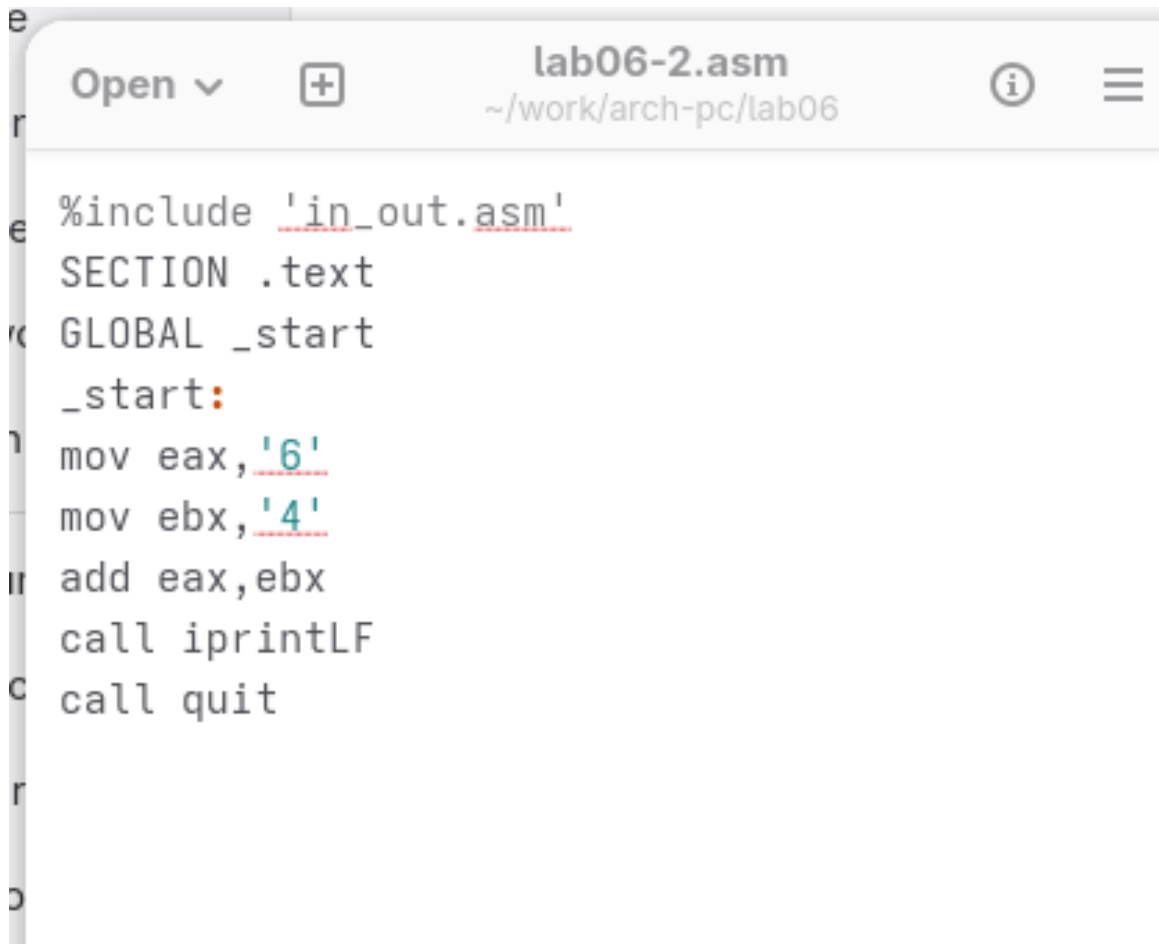
```
aaharlamova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm
aaharlamova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o -o lab06-1
aaharlamova@fedora:~/work/arch-pc/lab06$ ./lab06-1

aaharlamova@fedora:~/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.

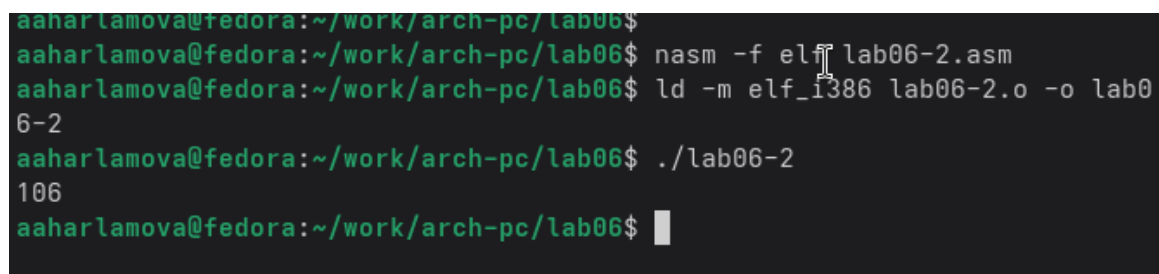
4. Как отмечалось выше, для работы с числами в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразовала текст программы с использованием этих функций.



```
lab06-2.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
```

Рисунок 2.5: Программа в файле lab6-2.asm



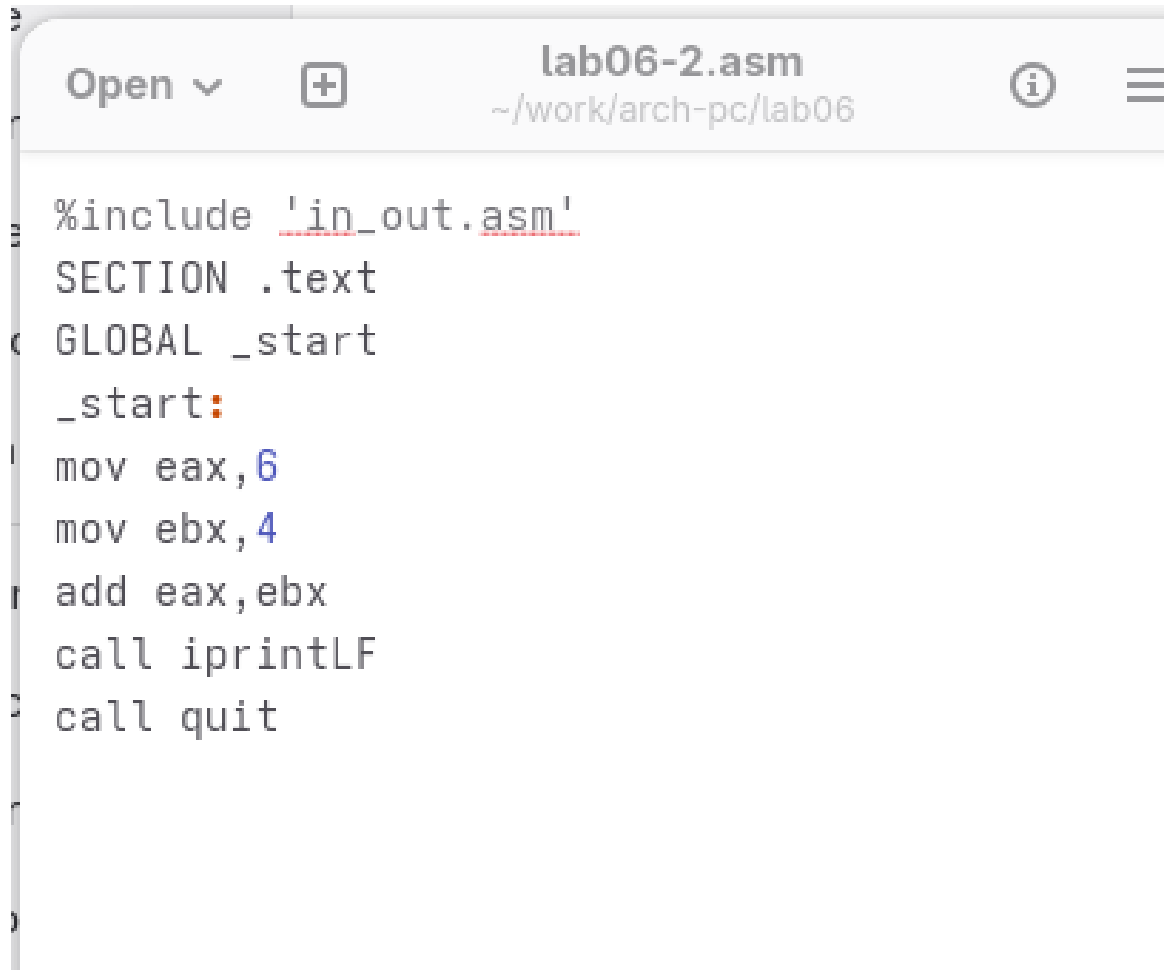
```
aaharlamova@fedora:~/work/arch-pc/lab06$
aaharlamova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm
aaharlamova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-2.o -o lab06-2
aaharlamova@fedora:~/work/arch-pc/lab06$ ./lab06-2
106
aaharlamova@fedora:~/work/arch-pc/lab06$
```

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 отличие от прошлой программы, функция `iprintLF` позволяет выве-

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

5. Аналогично предыдущему примеру изменим символы на числа.



```
lab06-2.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
```

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.

```

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

```

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

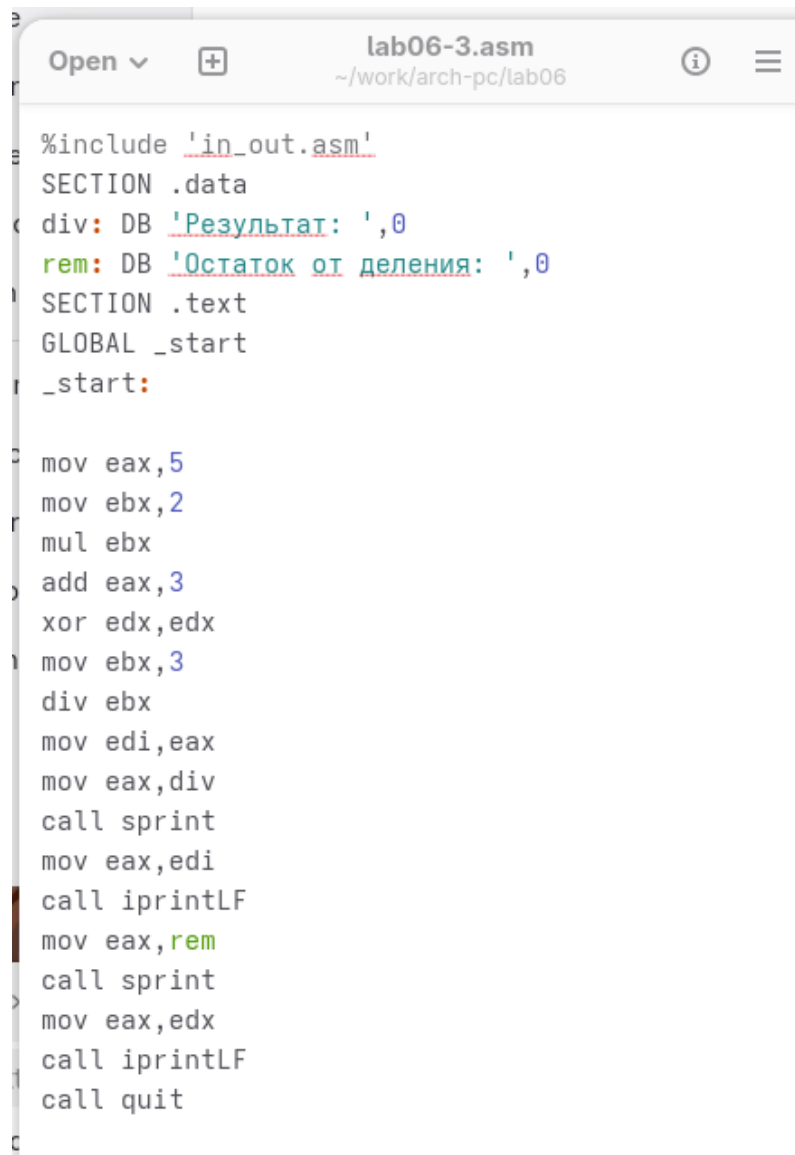
```

Рисунок 2.9: Запуск программы lab6-2.asm

- В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения

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

.

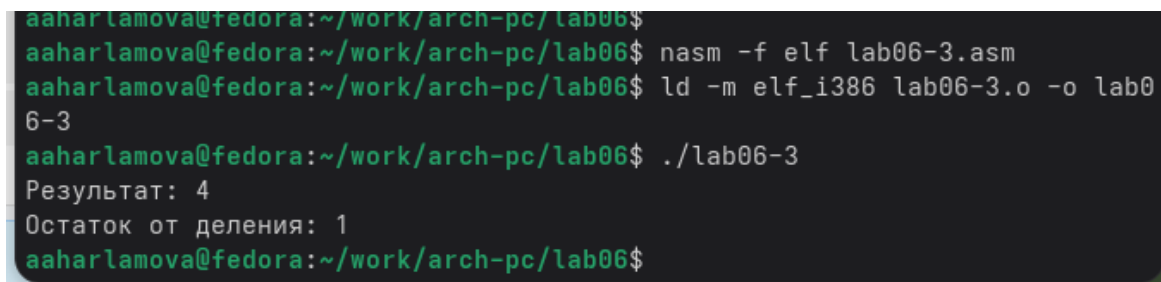


```
Open ▾ + lab06-3.asm ~/work/arch-pc/lab06

%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
```

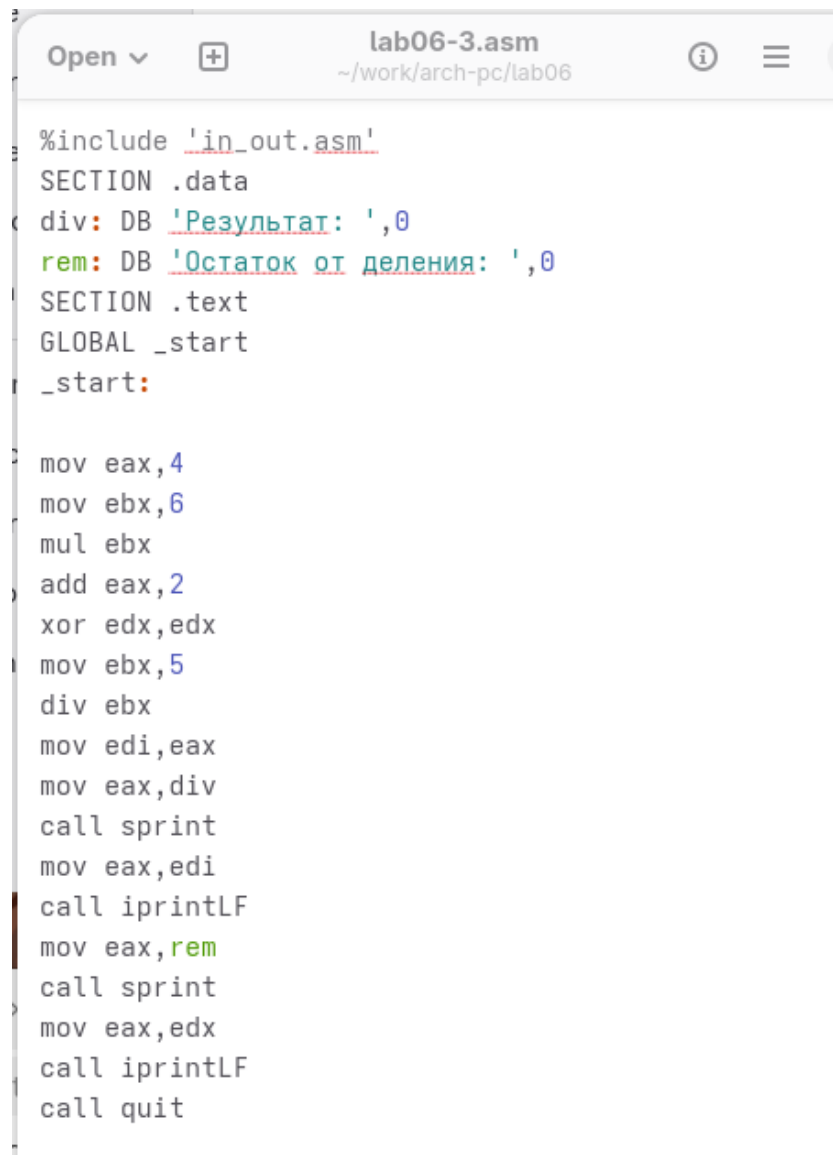
Рисунок 2.10: Программа в файле lab6-3.asm



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



```
lab06-3.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
```

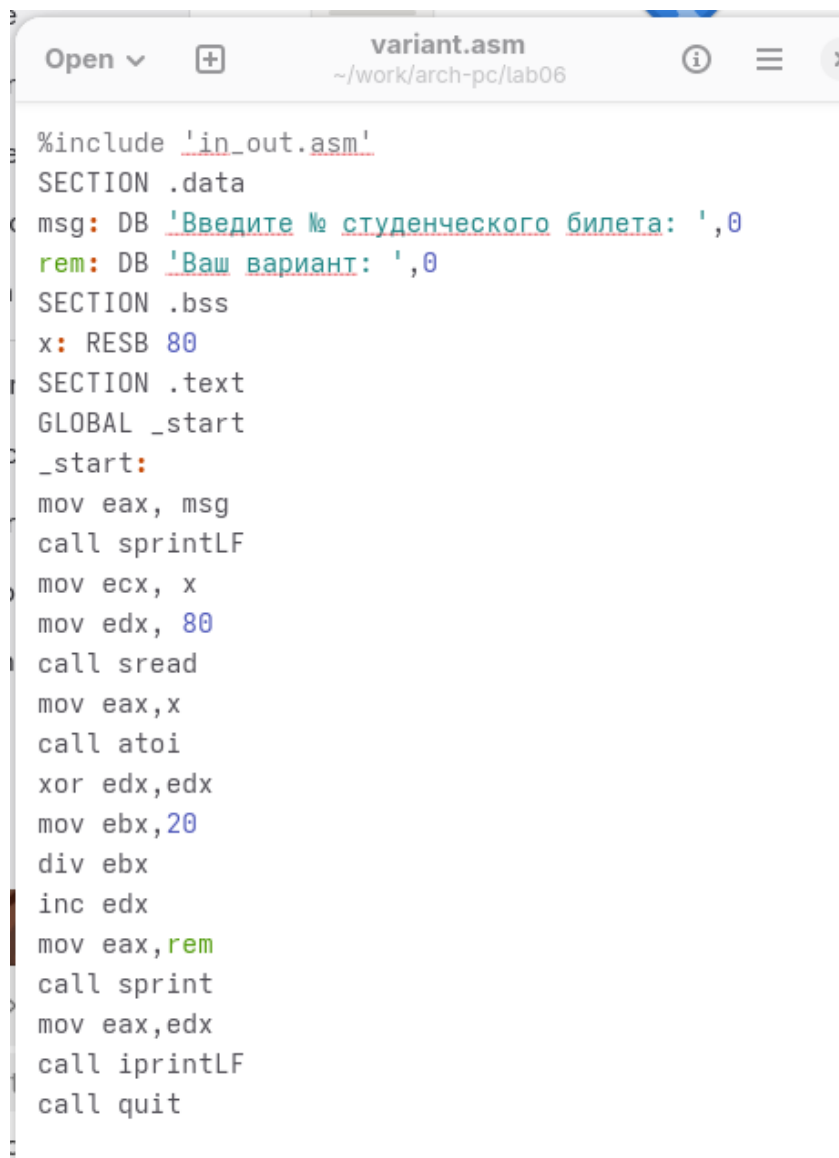
Рисунок 2.12: Программа в файле lab6-3.asm

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

Рисунок 2.13: Запуск программы lab6-3.asm

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

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`.



```
%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 iprintLF
call quit
```

Рисунок 2.14: Программа в файле variant.asm



```

aaharlamova@fedora:~/work/arch-pc/lab06$
aaharlamova@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
aaharlamova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 variant.o -o variant
aaharlamova@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132250421
Ваш вариант: 2
aaharlamova@fedora:~/work/arch-pc/lab06$

```

Рисунок 2.15: Запуск программы variant.asm

## 2.1 Ответы на вопросы

1. Какие строки листинга отвечают за вывод на экран 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 следующие 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

## 2.2 Задание по варианту

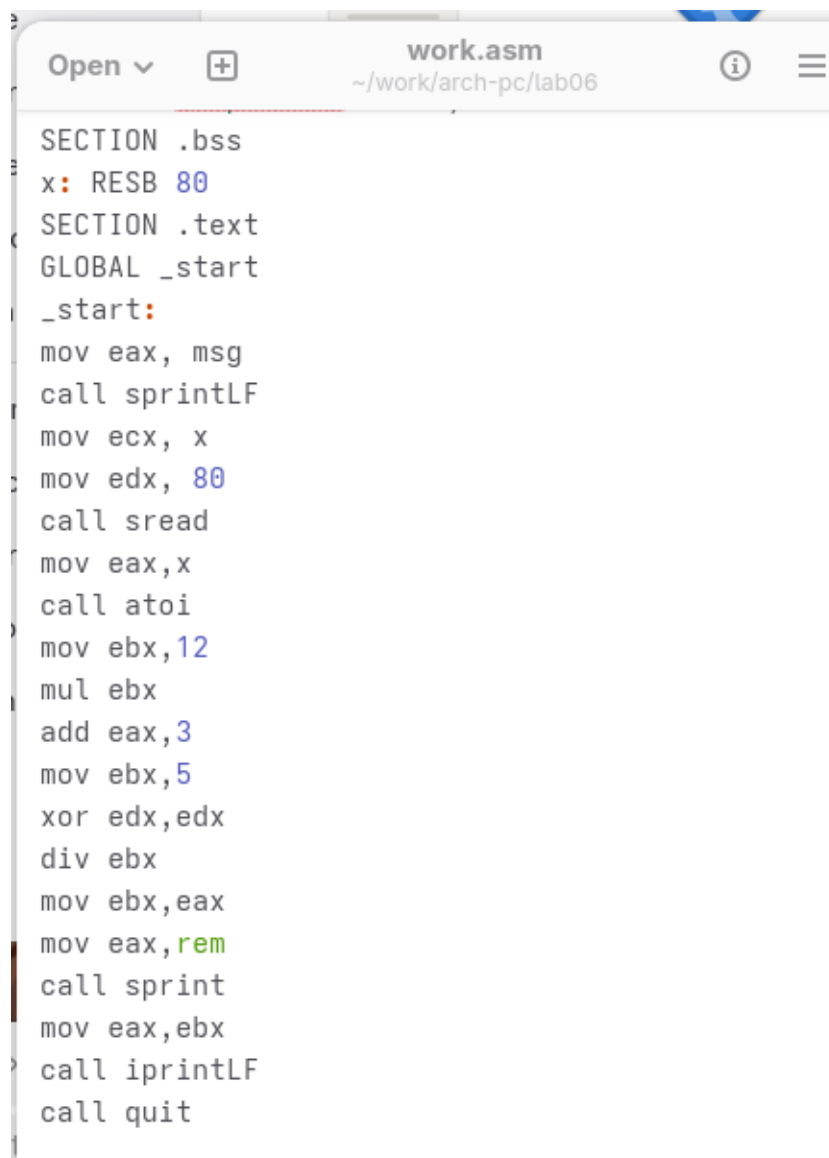
Написать программу вычисления выражения  $y = f(x)$ . Программа должна выводить выражение для вычисления, выводить запрос на ввод значения  $x$ , вычислять заданное выражение в зависимости от введенного  $x$ , выводить результат вычислений. Вид функции  $f(x)$  выбрать из таблицы 6.3 вариантов заданий 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 2 -

$$(12x + 3)/5$$

для

$$x_1 = 1, x_2 = 6$$

A screenshot of a code editor window titled "work.asm" with a subtitle "~/work/arch-pc/lab06". The editor contains assembly code for a program. The code starts with a section declaration for ".bss" and reserves 80 bytes for variable "x". It then moves to the ".text" section and declares the global "\_start" symbol. The program begins at "\_start:" by moving "msg" to "eax", calling "sprintLF", moving "x" to "ecx", and "80" to "edx", then calling "sread". It continues by moving "x" to "eax", calling "atoi", moving "12" to "ebx", multiplying "ebx" by "eax", adding "3" to "eax", moving "5" to "ebx", XORing "edx" with itself, dividing "eax" by "ebx", moving the result to "ebx", moving "rem" to "eax", calling "sprint", moving "ebx" to "eax", calling "iprintLF", and finally calling "quit".

```
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, 12
mul ebx
add eax, 3
mov ebx, 5
xor edx, edx
div ebx
mov ebx, eax
mov eax, rem
call sprint
mov eax, ebx
call iprintLF
call quit
```

Рисунок 2.16: Программа в файле work.asm

```
aaharlamova@fedora:~/work/arch-pc/lab06$  
aaharlamova@fedora:~/work/arch-pc/lab06$ nasm -f elf work.asm  
aaharlamova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 work.o -o work  
aaharlamova@fedora:~/work/arch-pc/lab06$ ./work  
Введите X  
1  
выражение = : 3  
aaharlamova@fedora:~/work/arch-pc/lab06$ ./work  
Введите X  
6  
выражение = : 15  
aaharlamova@fedora:~/work/arch-pc/lab06$
```

Рисунок 2.17: Запуск программы work.asm

## **3 Выводы**

Изучила работу с арифметическими операциями.