

# **Отчёт по лабораторной работе 6**

**Архитектура компьютера**

Абдурахмонова Рухшона Бахтиёровна

# 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 . . . . .	11
2.2.1	Ответы на вопросы по программе variant.asm . . . . .	17
2.3	Выполнение заданий для самостоятельной работы . . . . .	19
<b>3</b>	<b>Выводы</b>	<b>22</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 . . . . .	8
2.5	Программа в файле lab6-2.asm . . . . .	9
2.6	Запуск программы lab6-2.asm . . . . .	9
2.7	Программа в файле lab6-2.asm . . . . .	10
2.8	Запуск программы lab6-2.asm . . . . .	10
2.9	Программа в файле lab6-2.asm . . . . .	11
2.10	Запуск программы lab6-2.asm . . . . .	11
2.11	Программа в файле lab6-3.asm . . . . .	12
2.12	Запуск программы lab6-3.asm . . . . .	13
2.13	Программа в файле lab6-3.asm . . . . .	14
2.14	Запуск программы lab6-3.asm . . . . .	14
2.15	Программа в файле variant.asm . . . . .	16
2.16	Запуск программы variant.asm . . . . .	17
2.17	Программа в файле calc.asm . . . . .	20
2.18	Запуск программы calc.asm . . . . .	21

## **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 с названием “lab6-1.asm”.

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

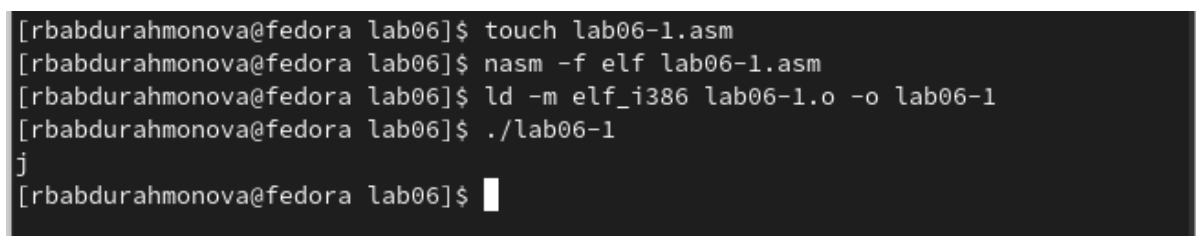
В данной программе, в регистр `eax` записан символ ‘6’, а в регистр `ebx` символ ‘4’. Затем мы прибавляем значение регистра `ebx` к значению в регистре `eax` (результат сложения будет записан в регистр `eax`). После этого мы выводим результат.

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`.



```
%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
```

Рис. 2.1: Программа в файле lab6-1.asm



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

Рис. 2.2: Запуск программы 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'.

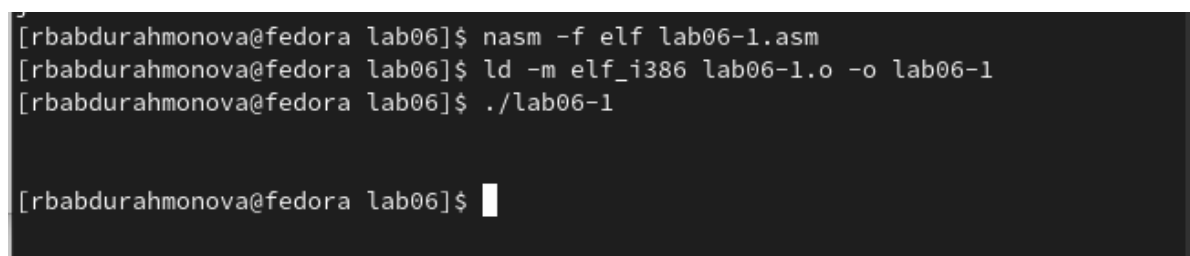
Далее был изменен текст программы и вместо символов записаны числа.



```
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 sprintLF
call quit
```

Рис. 2.3: Программа в файле lab6-1.asm



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

[rbabdurahmonova@fedora lab06]$
```

Рис. 2.4: Запуск программы 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.

В файле “in\_out.asm” реализованы подпрограммы для работы с числами и преобразования символов ASCII. Был модифицирован текст программы с исполь-



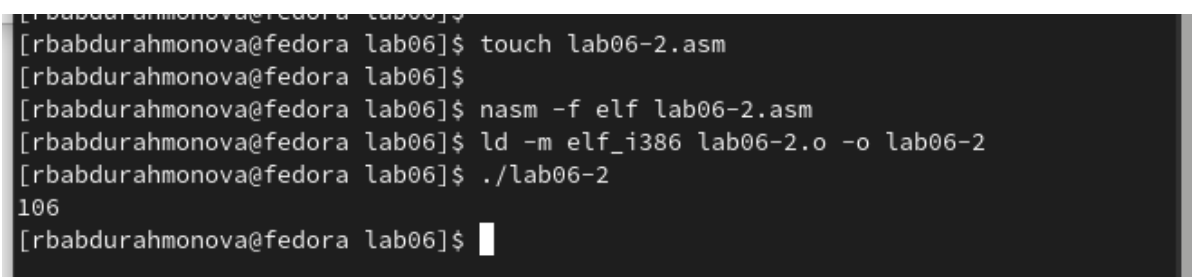
зованием этих функций.



```
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
```

Рис. 2.5: Программа в файле lab6-2.asm

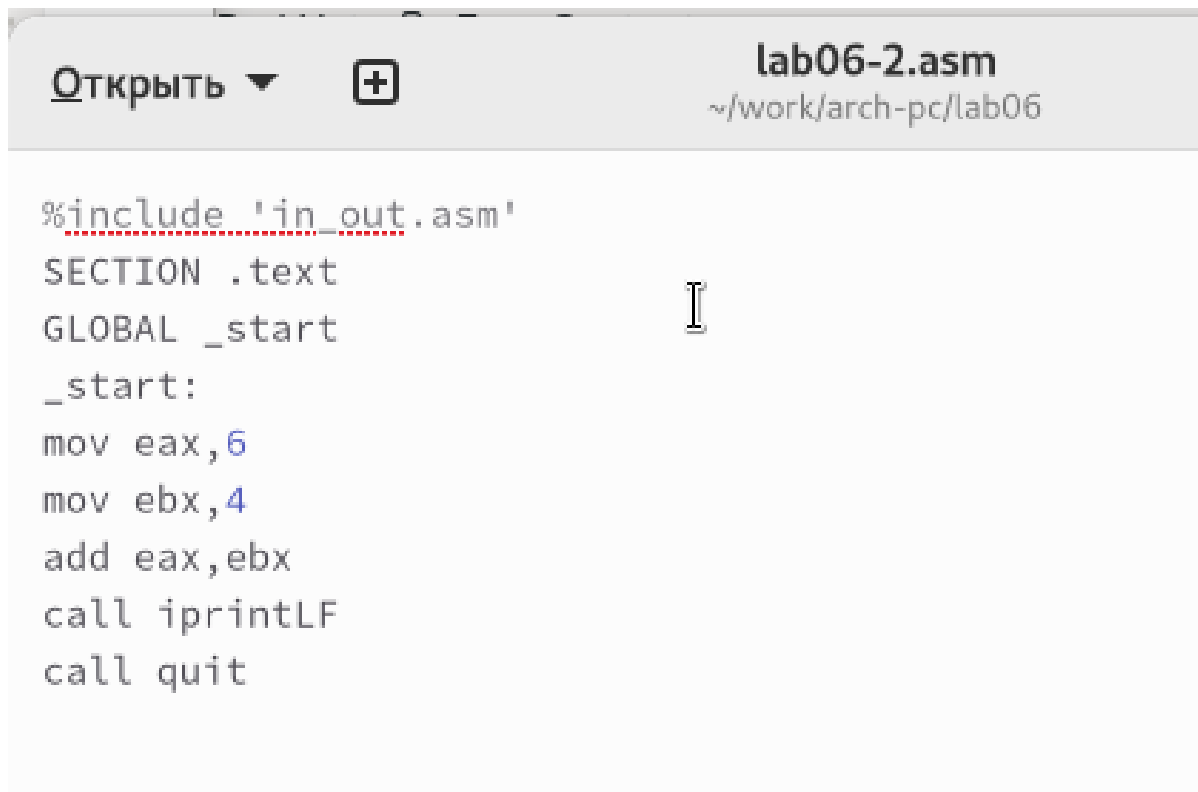


```
[rbabdurahmonova@fedora lab06]$ touch lab06-2.asm
[rbabdurahmonova@fedora lab06]$
[rbabdurahmonova@fedora lab06]$ nasm -f elf lab06-2.asm
[rbabdurahmonova@fedora lab06]$ ld -m elf_i386 lab06-2.o -o lab06-2
[rbabdurahmonova@fedora lab06]$ ./lab06-2
106
[rbabdurahmonova@fedora lab06]$
```

Рис. 2.6: Запуск программы 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.

По аналогии с предыдущим примером, были заменены символы на числа.

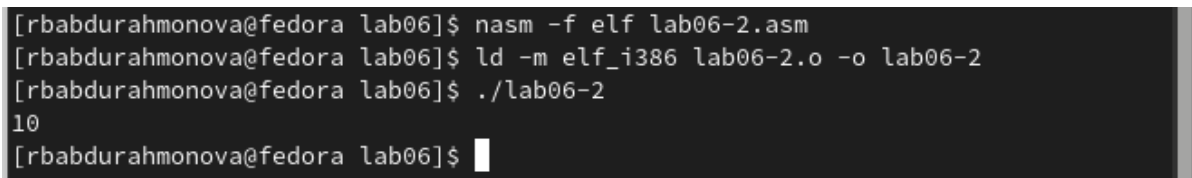


```
lab06-2.asm
~/work/arch-pc/lab06

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

Рис. 2.7: Программа в файле lab6-2.asm

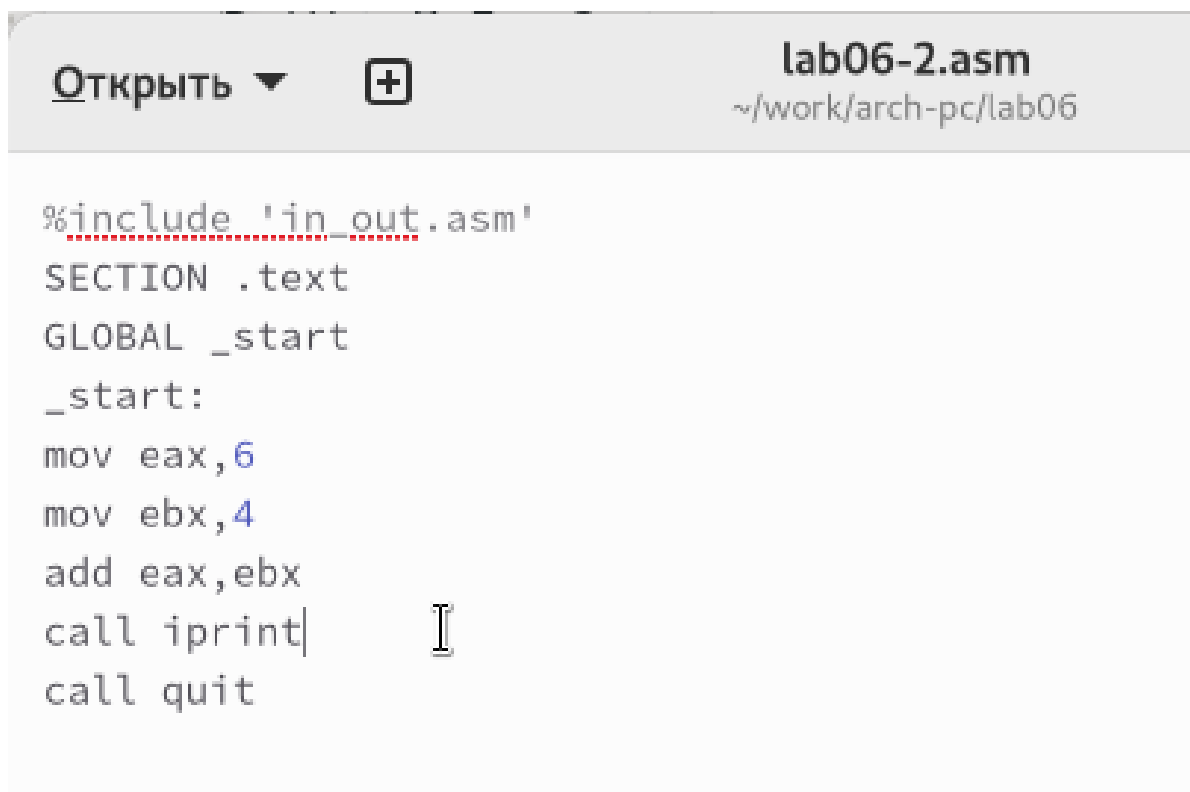
Функция iprintLF позволяет выводить числа, и на этот раз в качестве операндов использовались именно числа, а не коды символов. В результате мы получили число 10.



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

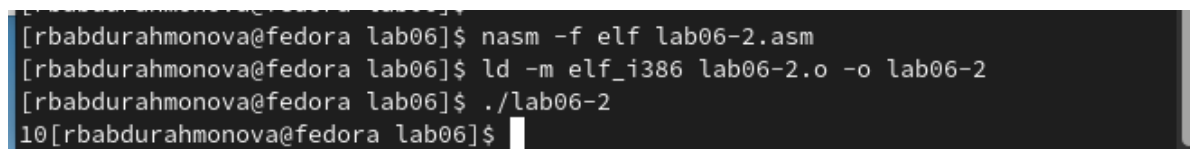
Рис. 2.8: Запуск программы lab6-2.asm

Далее была заменена функция iprintLF на iprint, создан исполняемый файл и запущен. Вывод теперь отличается отсутствием перехода на новую строку.



```
Открыть ▾ + 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 iprint  
call quit
```

Рис. 2.9: Программа в файле lab6-2.asm



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

Рис. 2.10: Запуск программы lab6-2.asm

## 2.2 Выполнение арифметических операций в NASM

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

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



```
%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: Программа в файле lab6-3.asm

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

Рис. 2.12: Запуск программы lab6-3.asm

Изменила текст программы для вычисления выражения

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

. Создала исполняемый файл и проверила его работу.

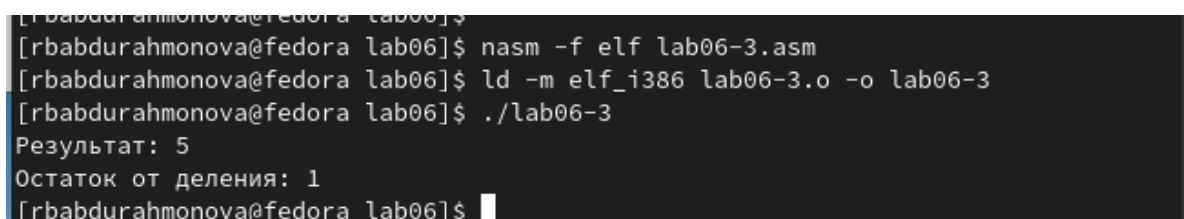


```
Открыть ▾ + lab06-3.asm
~/work/arch-pc/lab06

%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
```

Рис. 2.13: Программа в файле lab6-3.asm




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

Рис. 2.14: Запуск программы lab6-3.asm

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.



```
%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: Программа в файле variant.asm



```
[rbabdurahmonova@fedora lab06]$ touch variant.asm
[rbabdurahmonova@fedora lab06]$ nasm -f elf variant.asm
[rbabdurahmonova@fedora lab06]$ ld -m elf_i386 variant.o -o variant
[rbabdurahmonova@fedora lab06]$ ./variant
Введите № студенческого билета:
1032235942
Ваш вариант: 3
[rbabdurahmonova@fedora lab06]$
```

Рис. 2.16: Запуск программы variant.asm

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

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

Ответ: Строки, отвечающие за вывод сообщения “Ваш вариант:”, - это строки, где происходит перемещение фразы в регистр eax с помощью инструкции mov eax, rem, а затем вызов подпрограммы вывода строки с помощью инструкции call sprint.

2. Для чего используются следующие инструкции?

Ответ:

- mov esx, x: Инструкция mov esx, x используется для сохранения значения регистра esx в переменной x.
- mov edx, 80: Инструкция mov edx, 80 используется для присваивания значения 80 регистру edx.
- call sread: Инструкция call sread используется для вызова подпрограммы, которая считывает данные из консоли.

3. Для чего используется инструкция “call atoi”?

Ответ: Инструкция `call atoi` используется для преобразования введенных символов в числовой формат.

4. Какие строки листинга отвечают за вычисления варианта?

Ответ: Строки, отвечающие за вычисление варианта, включают следующие инструкции:

- `xor edx, edx`: Инструкция `xor edx, edx` используется для обнуления регистра `edx`.
- `mov ebx, 20`: Инструкция `mov ebx, 20` используется для присваивания значения 20 регистру `ebx`.
- `div ebx`: Инструкция `div ebx` используется для деления номера студента на 20.
- `inc edx`: Инструкция `inc edx` используется для увеличения значения регистра `edx` на 1.

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

Ответ: При выполнении инструкции `div ebx` остаток от деления записывается в регистр `edx`.

6. Для чего используется инструкция “`inc edx`”?

Ответ: Инструкция `inc edx` используется для увеличения значения регистра `edx` на 1, что необходимо для вычисления варианта по формуле.

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

Ответ: Строки, отвечающие за вывод на экран результата вычислений, включают следующие инструкции:

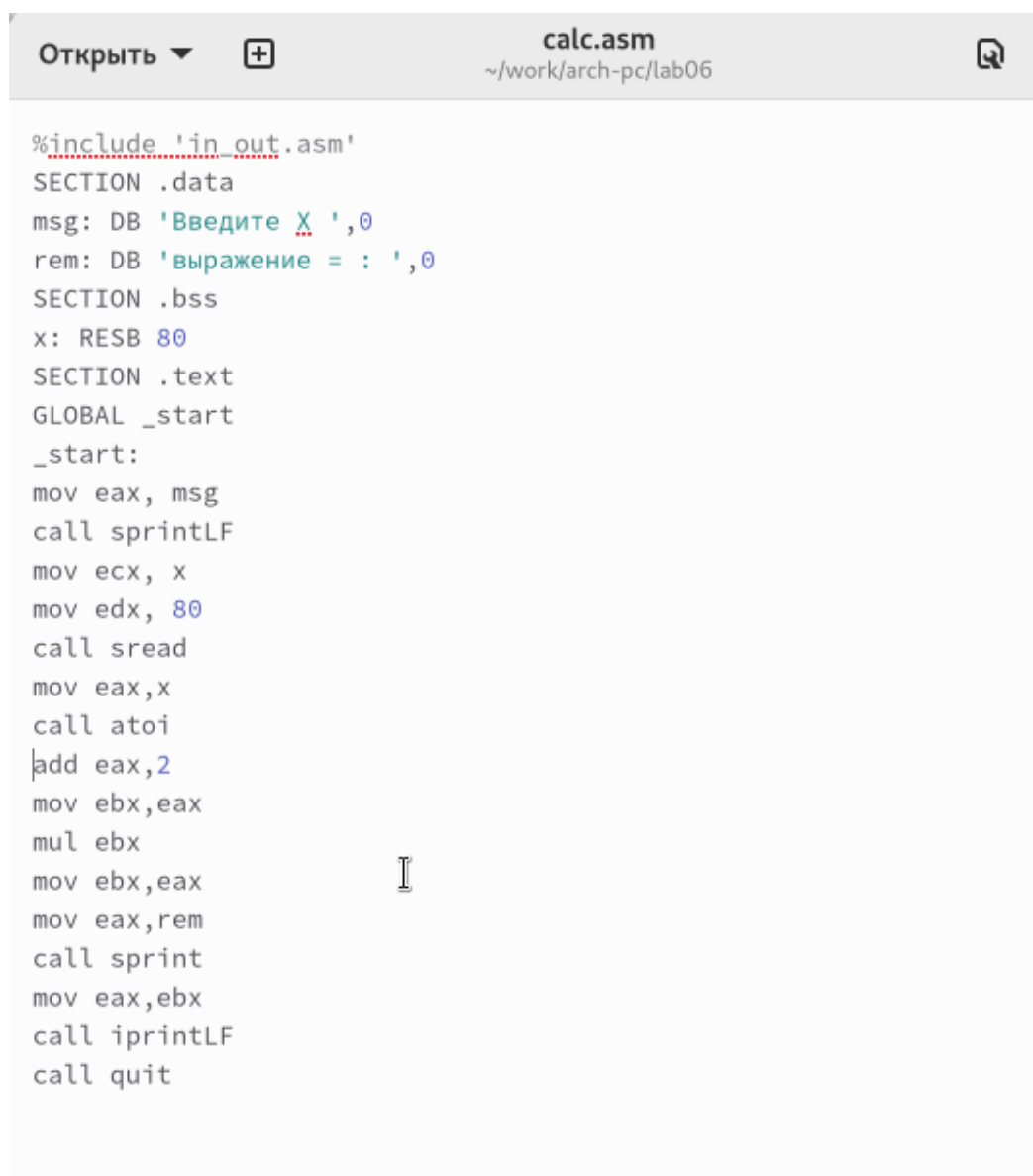
- `mov eax, edx`: Инструкция `mov eax, edx` используется для помещения результата в регистр `eax`.
- `call iprintLF`: Инструкция `call iprintLF` используется для вызова подпрограммы вывода результата.

## 2.3 Выполнение заданий для самостоятельной работы

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.

Вариант 3 -  $(2 + x)^2$  для  $x = 2, x = 8$



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

Рис. 2.17: Программа в файле calc.asm

```
[rbabdurahmonova@fedora lab06]$  
[rbabdurahmonova@fedora lab06]$ nasm -f elf calc.asm  
[rbabdurahmonova@fedora lab06]$ ld -m elf_i386 calc.o -o calc  
[rbabdurahmonova@fedora lab06]$ ./calc  
Введите X  
2  
выражение = : 16  
[rbabdurahmonova@fedora lab06]$ ./calc  
Введите X  
8  
выражение = : 100  
[rbabdurahmonova@fedora lab06]$
```

Рис. 2.18: Запуск программы calc.asm

Программа считает верно.

## **3 Выводы**

Изучили работу с арифметическими операциями.