

# **Отчёт по лабораторной работе 6**

**Архитектура компьютера**

Эргешов Байрам НКАбд-02-23

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Задание для самостоятельной работы . . . . .	19
<b>3</b>	<b>Выводы</b>	<b>22</b>

## Список иллюстраций

2.1	Код программы lab6-1.asm . . . . .	7
2.2	Компиляция и запуск программы lab6-1.asm . . . . .	7
2.3	Код программы lab6-1.asm . . . . .	8
2.4	Компиляция и запуск программы lab6-1.asm . . . . .	9
2.5	Код программы lab6-2.asm . . . . .	10
2.6	Компиляция и запуск программы lab6-2.asm . . . . .	10
2.7	Код программы lab6-2.asm . . . . .	11
2.8	Компиляция и запуск программы lab6-2.asm . . . . .	11
2.9	Код программы lab6-2.asm . . . . .	12
2.10	Компиляция и запуск программы lab6-2.asm . . . . .	12
2.11	Код программы lab6-3.asm . . . . .	13
2.12	Компиляция и запуск программы lab6-3.asm . . . . .	14
2.13	Код программы lab6-3.asm . . . . .	15
2.14	Компиляция и запуск программы lab6-3.asm . . . . .	15
2.15	Код программы variant.asm . . . . .	17
2.16	Компиляция и запуск программы variant.asm . . . . .	17
2.17	Код программы program.asm . . . . .	20
2.18	Компиляция и запуск программы program.asm . . . . .	21

## **Список таблиц**

# 1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

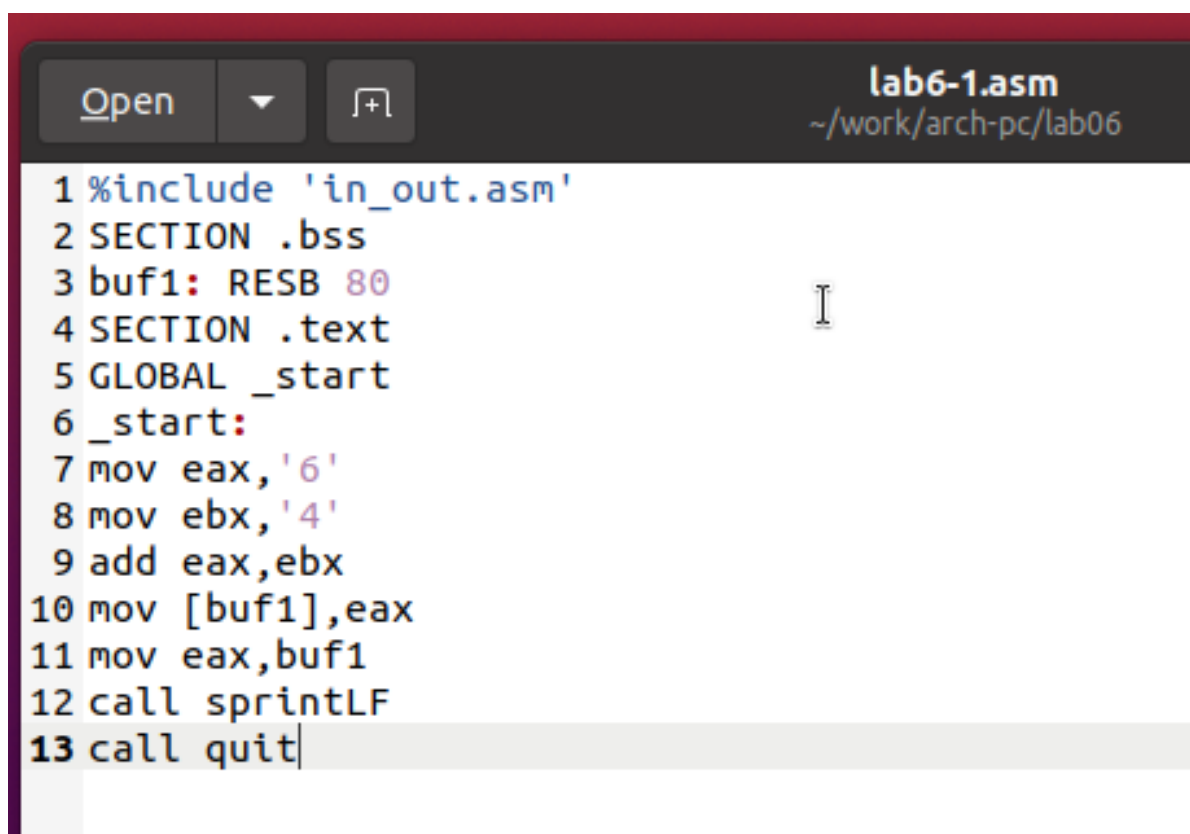
## 2 Выполнение лабораторной работы

Я создал каталог для программ лабораторной работы № 6, перешел в него и создал файл lab6-1.asm

В данной лабораторной работе мы рассмотрим примеры программ, которые выводят символьные и числовые значения. Программы будут использовать значение, записанное в регистре `eax`.

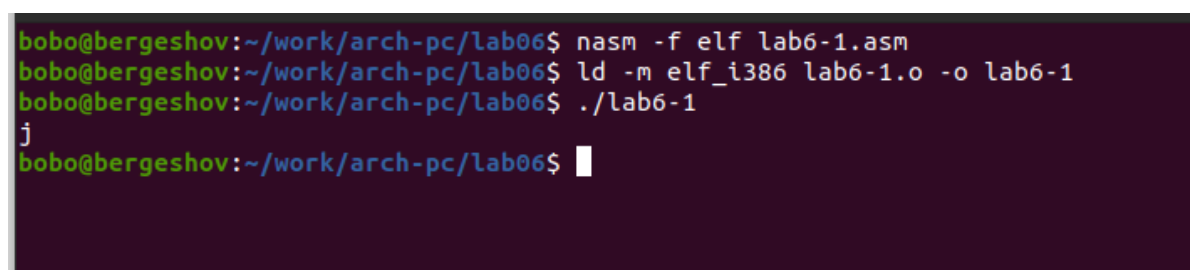
В этой программе мы записываем символ 6 в регистр `eax` (`mov eax, '6'`), и символ 4 в регистр `ebx` (`mov ebx, '4'`). Затем мы складываем значение в регистре `eax` с значением в регистре `ebx` (`add eax, ebx`), и выводим результат.

Для использования функции `sprintf`, которая требует адрес в регистре `eax`, нам нужно использовать дополнительную переменную. Мы записываем значение регистра `eax` в переменную `buf1` (`mov [buf1], eax`), затем записываем адрес переменной `buf1` в регистр `eax` (`mov eax, buf1`), и вызываем функцию `sprintf`.



```
1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax, '6'
8 mov ebx, '4'
9 add eax, ebx
10 mov [buf1], eax
11 mov eax, buf1
12 call sprintLF
13 call quit
```

Рис. 2.1: Код программы lab6-1.asm



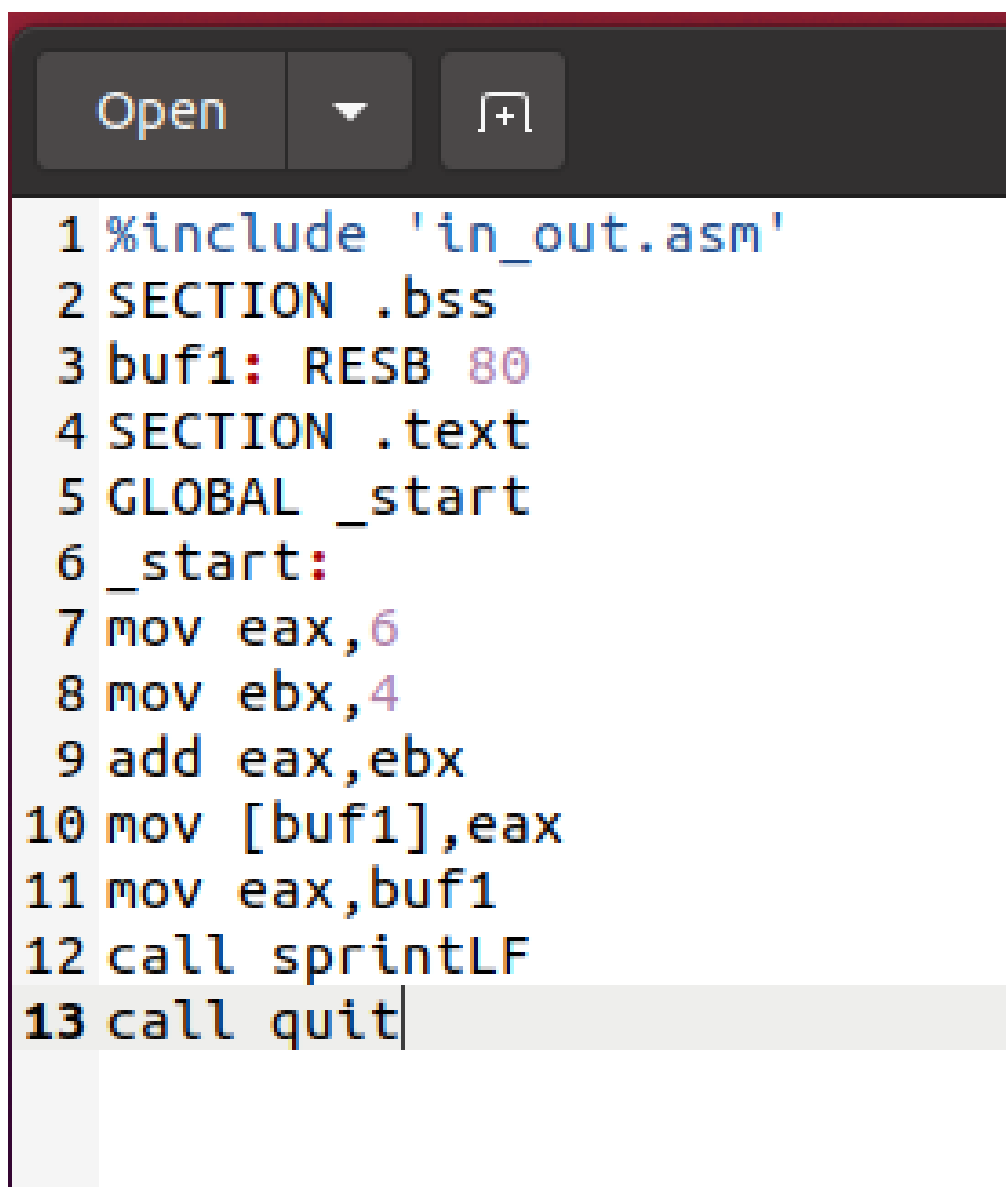
```
bobo@bergeshov:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
bobo@bergeshov:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-1.o -o lab6-1
bobo@bergeshov:~/work/arch-pc/lab06$ ./lab6-1
j
bobo@bergeshov:~/work/arch-pc/lab06$
```

Рис. 2.2: Компиляция и запуск программы lab6-1.asm

В этом случае мы ожидаем увидеть число 10 при выводе значения регистра `eax`. Однако, результатом будет символ 'j'. Это происходит потому, что код символа 6 равен 00110110 в двоичной системе (или 54 в десятичной системе), а код символа 4 равен 00110100 (или 52). Команда `add eax, ebx` записывает в регистр

eax сумму кодов – 01101010 (или 106), что в свою очередь является кодом символа 'j'.

Затем я изменил текст программы и вместо символов записал числа в регистры.



```
1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax,6
8 mov ebx,4
9 add eax,ebx
10 mov [buf1],eax
11 mov eax,buf1
12 call printf
13 call quit
```

Рис. 2.3: Код программы lab6-1.asm



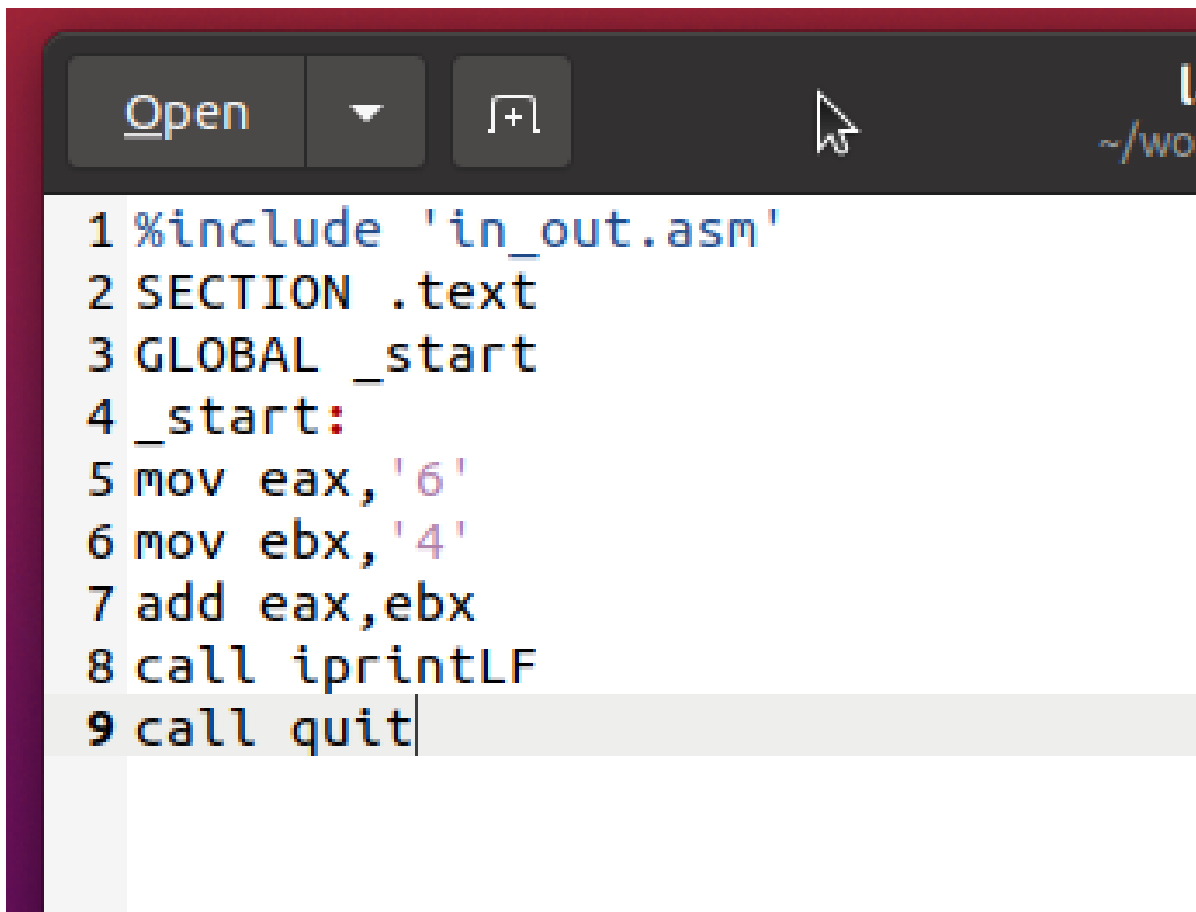
```
bobo@bergeshov:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
bobo@bergeshov:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-1.o -o lab6-1
bobo@bergeshov:~/work/arch-pc/lab06$ ./lab6-1

bobo@bergeshov:~/work/arch-pc/lab06$ █
```

Рис. 2.4: Компиляция и запуск программы lab6-1.asm

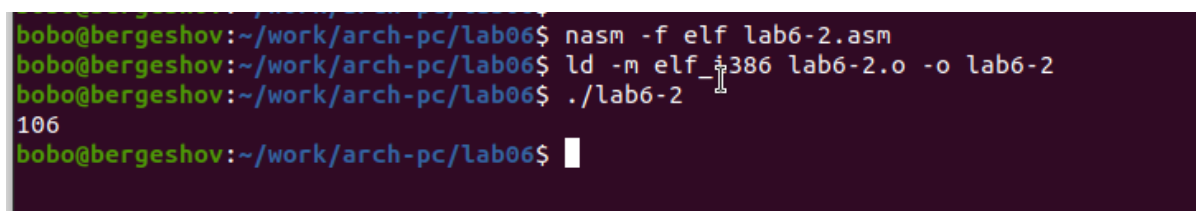
Как и в предыдущем случае, при выполнении программы мы не получим число 10. В данном случае будет выведен символ с кодом 10. Это символ конца строки (возврат каретки). В консоли он не отображается, но добавляет пустую строку.

Как уже упоминалось, в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Я изменил текст программы, используя эти функции.



```
1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax, '6'
6 mov ebx, '4'
7 add eax, ebx
8 call iprintLF
9 call quit
```

Рис. 2.5: Код программы lab6-2.asm

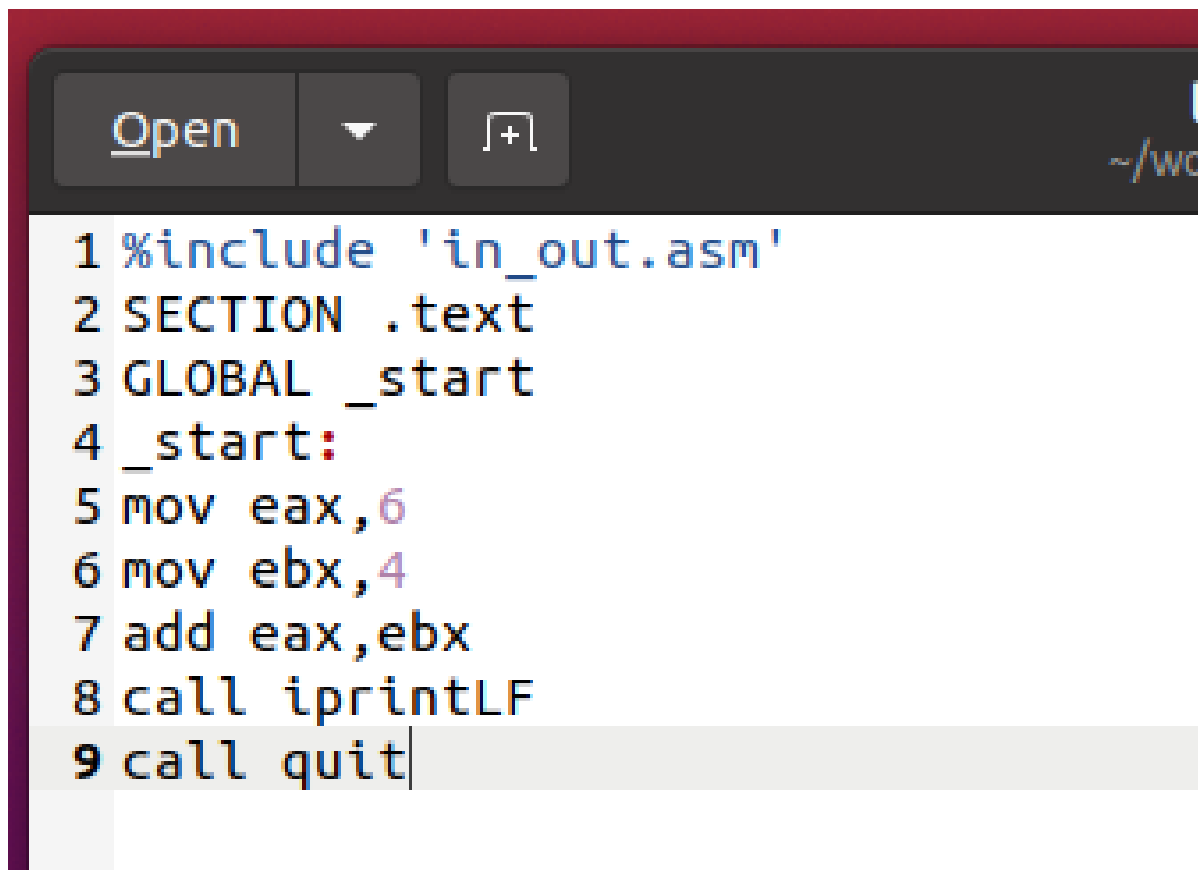


```
bobo@bergeshov:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
bobo@bergeshov:~/work/arch-pc/lab06$ ld -m elf_386 lab6-2.o -o lab6-2
bobo@bergeshov:~/work/arch-pc/lab06$ ./lab6-2
106
bobo@bergeshov:~/work/arch-pc/lab06$
```

Рис. 2.6: Компиляция и запуск программы lab6-2.asm

В результате выполнения программы мы получим число 106. В данном случае, как и в первом примере, команда `add` складывает значения символов '6' и '4' ( $54+52=106$ ). Однако, в отличие от предыдущей программы, функция `iprintLF` позволяет вывести число, а не символ, кодом которого является это число.

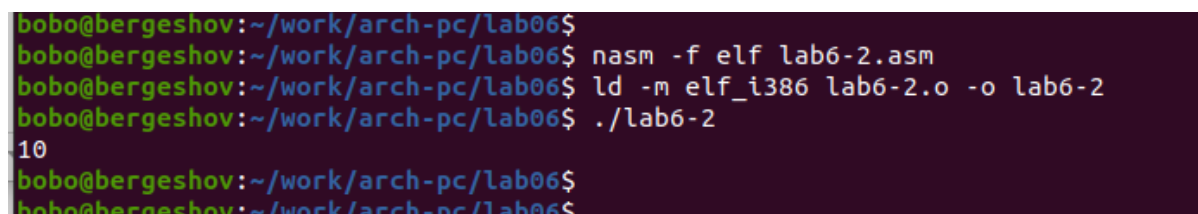
Аналогично предыдущему примеру, мы заменили символы на числа.



```
1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax,6
6 mov ebx,4
7 add eax,ebx
8 call iprintLF
9 call quit
```

Рис. 2.7: Код программы lab6-2.asm

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

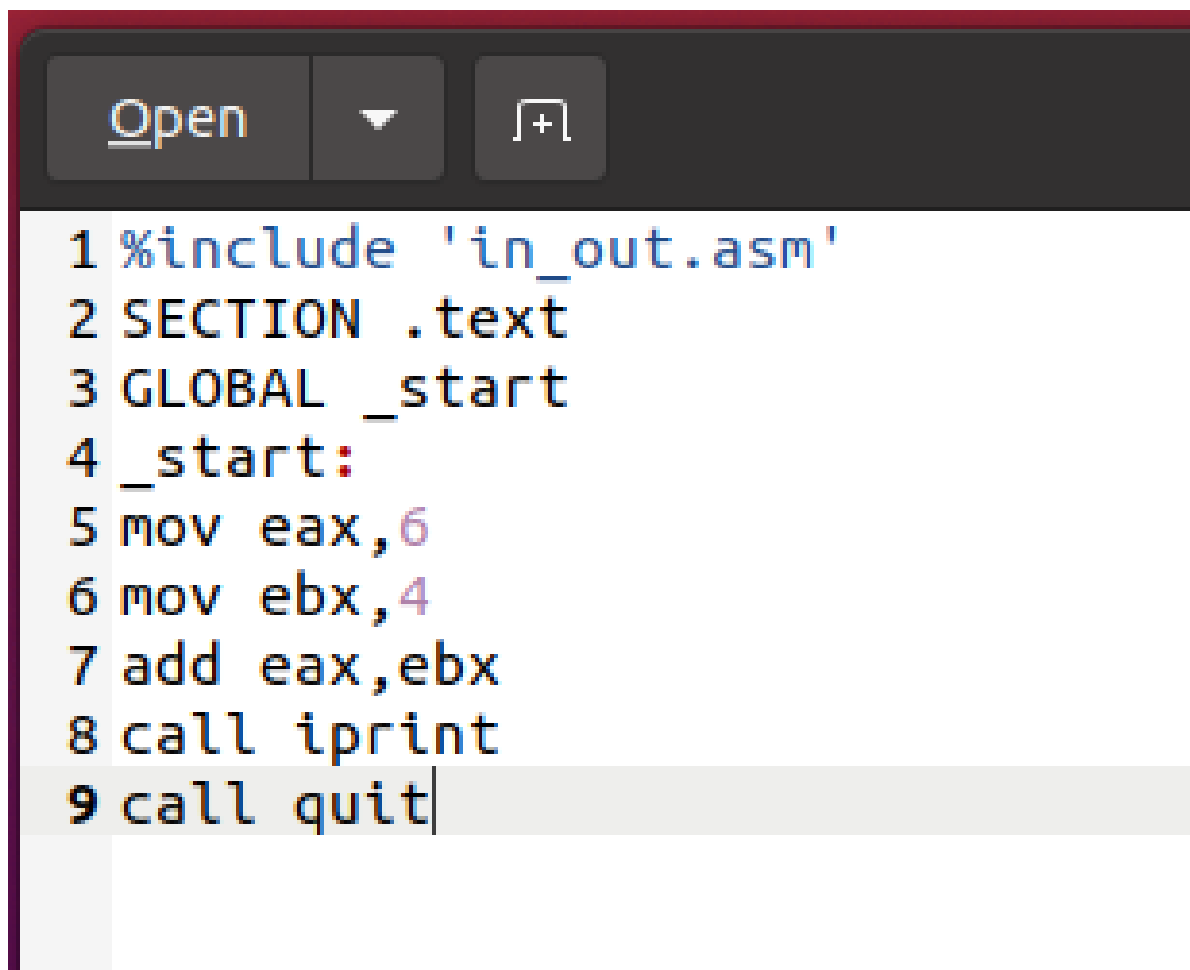


```
bobo@bergeshov:~/work/arch-pc/lab06$
bobo@bergeshov:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
bobo@bergeshov:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
bobo@bergeshov:~/work/arch-pc/lab06$ ./lab6-2
10
bobo@bergeshov:~/work/arch-pc/lab06$
bobo@bergeshov:~/work/arch-pc/lab06$
```

Рис. 2.8: Компиляция и запуск программы lab6-2.asm

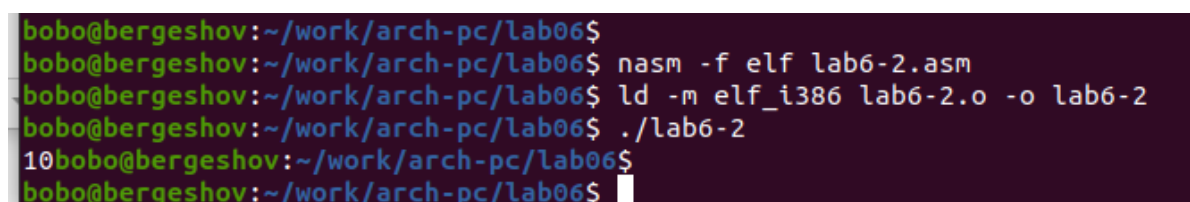
Функция iprintLF позволяет вывести число, и в этом случае операнды являются числами (а не кодами символов). Поэтому мы получаем число 10.

Мы заменили функцию `iprintLF` на `iprint`. Создали исполняемый файл и запустили его. Вывод отличается тем, что нет переноса строки.



```
1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax,6
6 mov ebx,4
7 add eax,ebx
8 call iprint
9 call quit
```

Рис. 2.9: Код программы lab6-2.asm



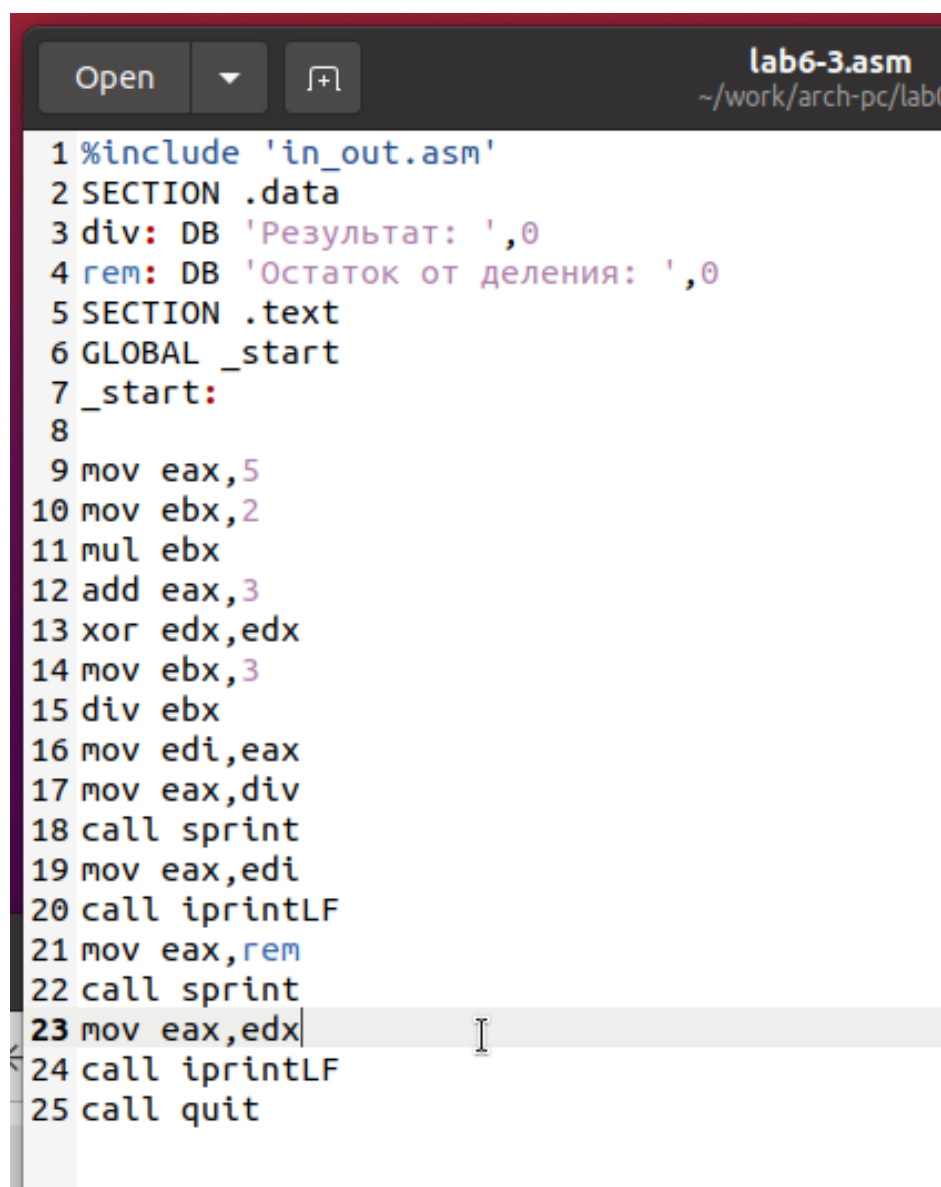
```
bobo@bergeshov:~/work/arch-pc/lab06$
bobo@bergeshov:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
bobo@bergeshov:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
bobo@bergeshov:~/work/arch-pc/lab06$ ./lab6-2
10bobo@bergeshov:~/work/arch-pc/lab06$
bobo@bergeshov:~/work/arch-pc/lab06$
```

Рис. 2.10: Компиляция и запуск программы lab6-2.asm

В качестве примера выполнения арифметических операций в NASM рассмот-

рели программу вычисления арифметического выражения

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



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

1 %include 'in_out.asm'
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8
9 mov eax,5
10 mov ebx,2
11 mul ebx
12 add eax,3
13 xor edx,edx
14 mov ebx,3
15 div ebx
16 mov edi,eax
17 mov eax,div
18 call sprint
19 mov eax,edi
20 call iprintLF
21 mov eax,rem
22 call sprint
23 mov eax,edx
24 call iprintLF
25 call quit
```

Рис. 2.11: Код программы lab6-3.asm

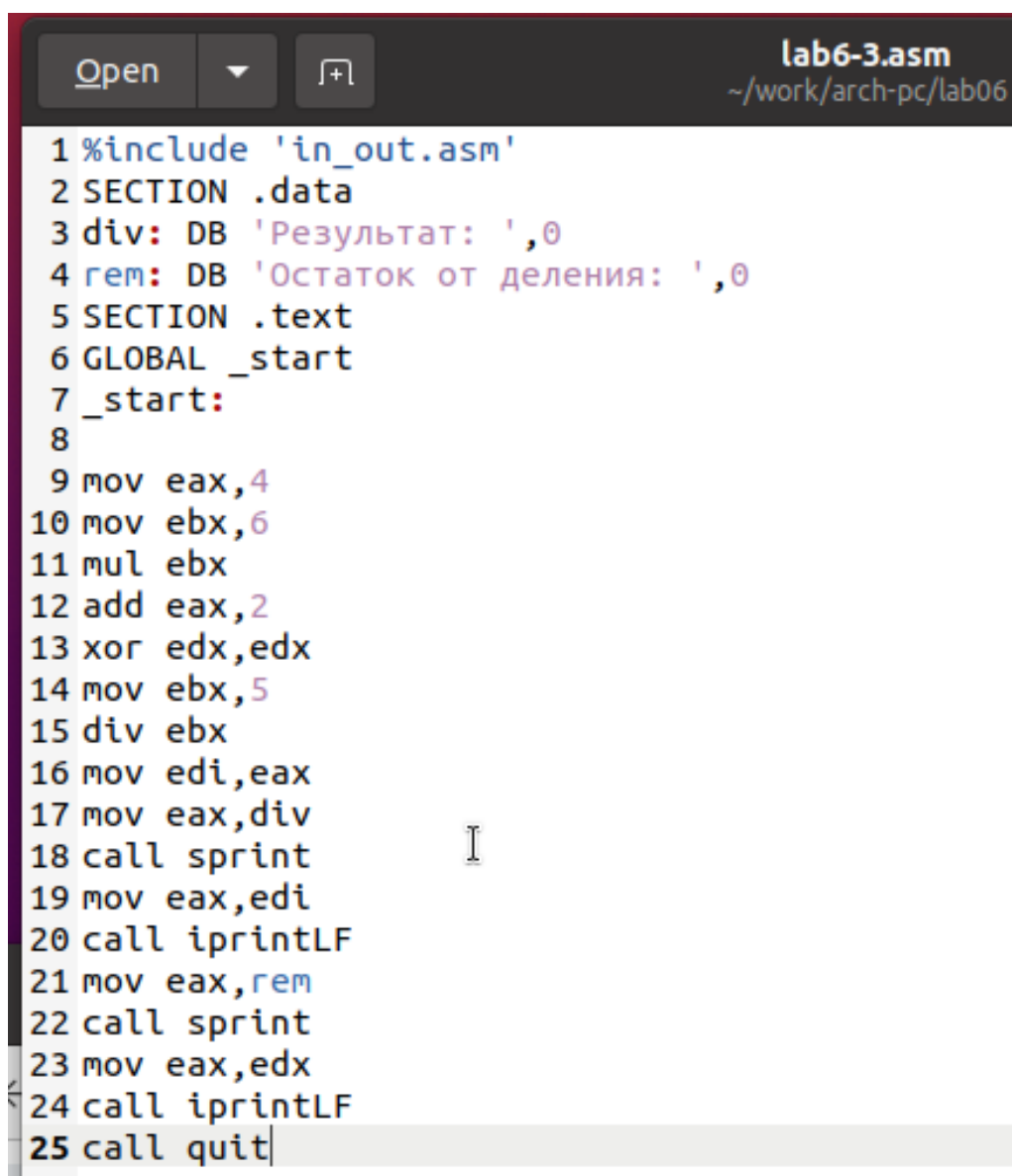
```
bobo@bergeshov:~/work/arch-pc/lab06$  
bobo@bergeshov:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm  
bobo@bergeshov:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-3.o -o lab6-3  
bobo@bergeshov:~/work/arch-pc/lab06$ ./lab6-3  
Результат: 4  
Остаток от деления: 1  
bobo@bergeshov:~/work/arch-pc/lab06$  
bobo@bergeshov:~/work/arch-pc/lab06$
```

Рис. 2.12: Компиляция и запуск программы lab6-3.asm

Z изменил текст программы для вычисления выражения

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

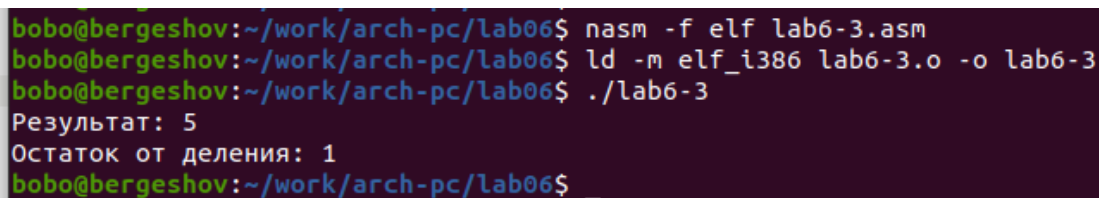
. Создал исполняемый файл и проверил его работу.



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

1 %include 'in_out.asm'
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8
9 mov eax,4
10 mov ebx,6
11 mul ebx
12 add eax,2
13 xor edx,edx
14 mov ebx,5
15 div ebx
16 mov edi,eax
17 mov eax,div
18 call sprint
19 mov eax,edi
20 call iprintLF
21 mov eax,rem
22 call sprint
23 mov eax,edx
24 call iprintLF
25 call quit
```

Рис. 2.13: Код программы lab6-3.asm



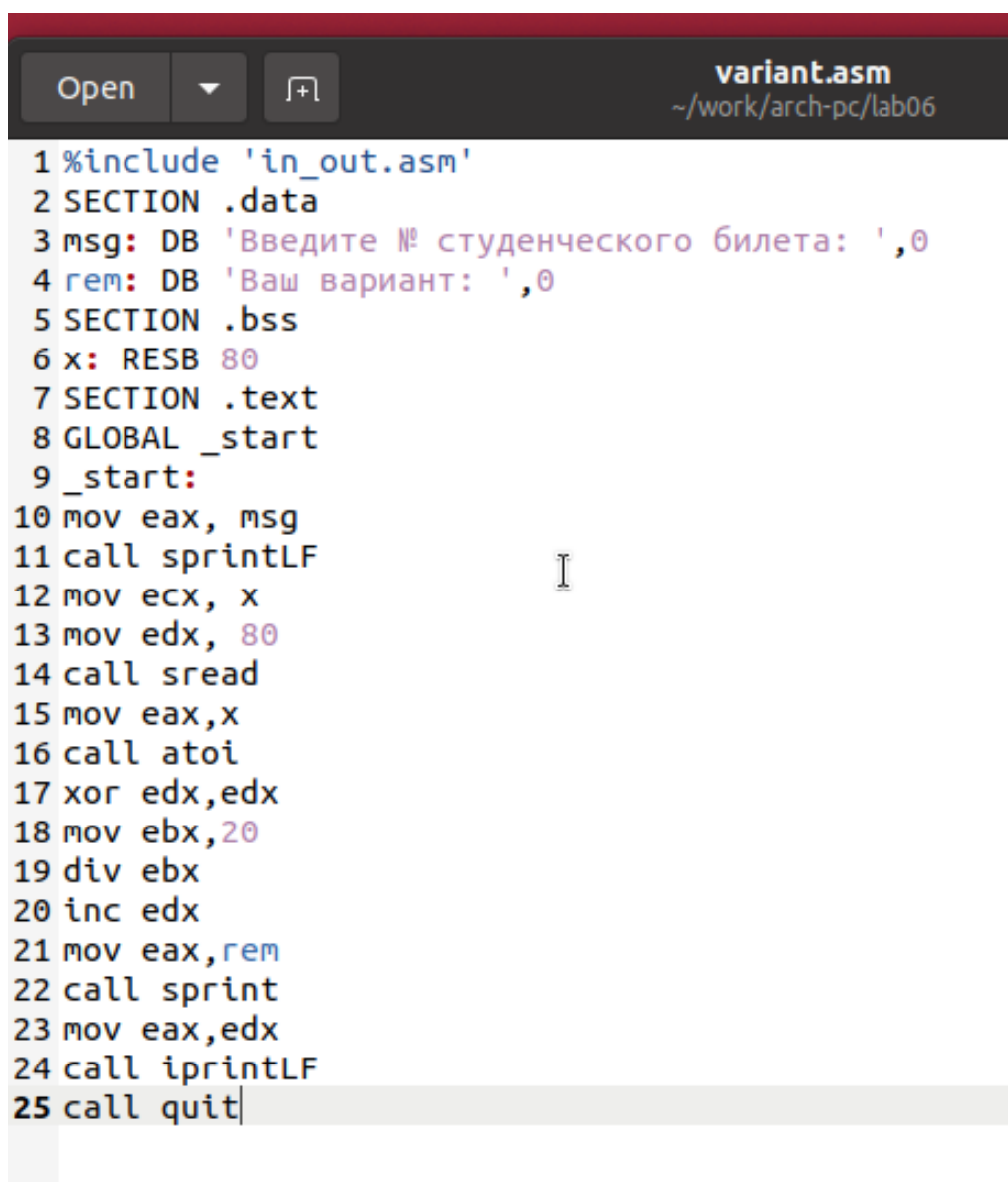
```
bobo@bergeshov:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
bobo@bergeshov:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-3.o -o lab6-3
bobo@bergeshov:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
bobo@bergeshov:~/work/arch-pc/lab06$
```

Рис. 2.14: Компиляция и запуск программы lab6-3.asm

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

В данном случае число, над которым необходимо выполнить арифметические операции, вводится с клавиатуры. Как уже упоминалось ранее, ввод с клавиатуры осуществляется в символьном виде, и для корректной работы арифметических операций в NASM символы необходимо преобразовать в числа. Для этого мы можем использовать функцию `atoi` из файла `in_out.asm`.

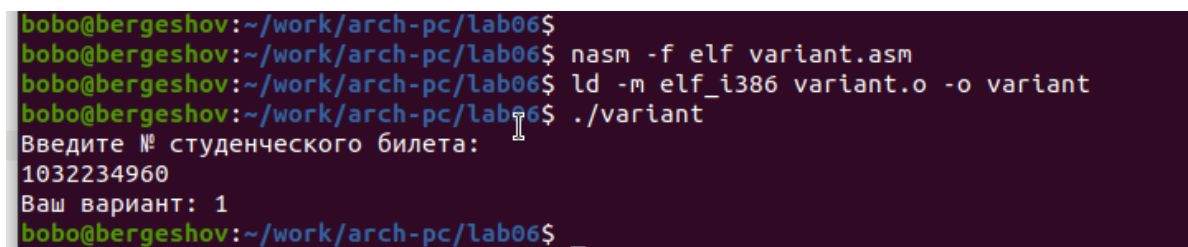




```
variant.asm
~/work/arch-pc/lab06

1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите № студенческого билета: ',0
4 rem: DB 'Ваш вариант: ',0
5 SECTION .bss
6 x: RESB 80
7 SECTION .text
8 GLOBAL _start
9 _start:
10 mov eax, msg
11 call sprintf
12 mov ecx, x
13 mov edx, 80
14 call sread
15 mov eax, x
16 call atoi
17 xor edx, edx
18 mov ebx, 20
19 div ebx
20 inc edx
21 mov eax, rem
22 call sprintf
23 mov eax, edx
24 call iprintLF
25 call quit
```

Рис. 2.15: Код программы variant.asm



```
bobo@bergeshov:~/work/arch-pc/lab06$
bobo@bergeshov:~/work/arch-pc/lab06$ nasm -f elf variant.asm
bobo@bergeshov:~/work/arch-pc/lab06$ ld -m elf_i386 variant.o -o variant
bobo@bergeshov:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032234960
Ваш вариант: 1
bobo@bergeshov:~/work/arch-pc/lab06$
```

Рис. 2.16: Компиляция и запуск программы variant.asm

## ответы на вопросы

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

*Строка `mov eax, ret` перекладывает значение переменной с фразой “Ваш вариант:” в регистр `eax`. Строка `call sprint` вызывает подпрограмму для вывода строки на экран.*

2. Для чего используются следующие инструкции?

*Инструкция `mov ecx, x` перекладывает значение регистра `ecx` в переменную `x`. Инструкция `mov edx, 80` устанавливает значение 80 в регистр `edx`. Инструкция `call sread` вызывает подпрограмму для считывания значения 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?

*Строки листинга, отвечающие за вычисления варианта, включают: `xor edx, edx` (обнуление регистра `edx`), `mov ebx, 20` (установка значения 20 в регистр `ebx`), `div ebx` (деление номера студенческого билета на 20), `inc edx` (увеличение значения регистра `edx` на 1).*

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

*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 регистра `edx` на 1. В данном случае она используется для выполнения формулы вычисления варианта, где требуется добавить 1 к остатку от деления.*

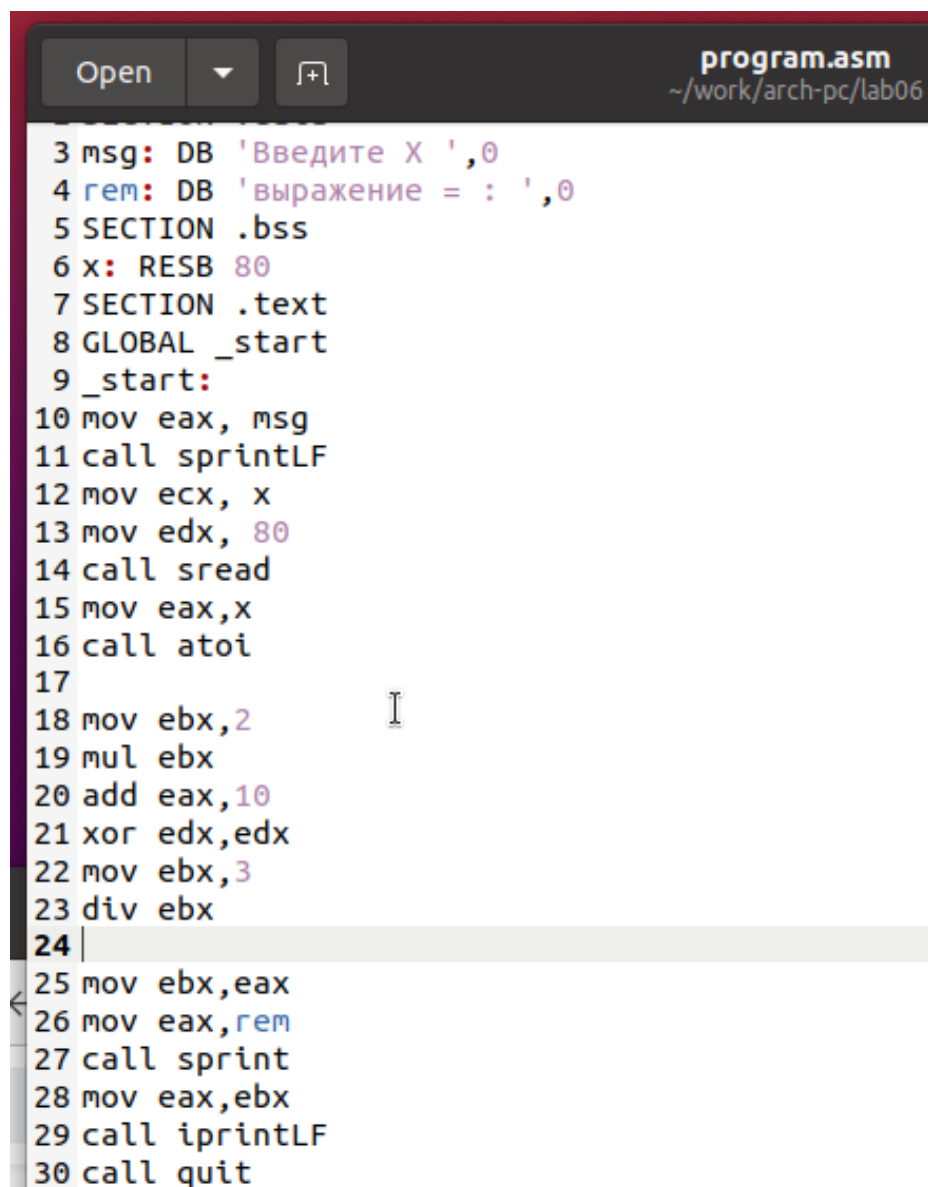
7. Какие строки листинга отвечают за вывод на экран результата вычислений?

*Строка `mov eax, edx` перекладывает результат вычислений в регистр `eax`. Строка `call iprintLF` вызывает подпрограмму для вывода значения на экран.*

## 2.1 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.

Вариант 1 -  $(10 + 2x)/3$  для  $x = 1, x = 10$



```
3 msg: DB 'Введите X ',0
4 rem: DB 'выражение = : ',0
5 SECTION .bss
6 x: RESB 80
7 SECTION .text
8 GLOBAL _start
9 _start:
10 mov eax, msg
11 call sprintf
12 mov ecx, x
13 mov edx, 80
14 call sread
15 mov eax, x
16 call atoi
17
18 mov ebx, 2
19 mul ebx
20 add eax, 10
21 xor edx, edx
22 mov ebx, 3
23 div ebx
24
25 mov ebx, eax
26 mov eax, rem
27 call sprintf
28 mov eax, ebx
29 call iprintLF
30 call quit
```

Рис. 2.17: Код программы program.asm

при  $x = 1$   $f(x) = 4$

при  $x = 10$   $f(x) = 10$

```
bobo@bergeshov:~/work/arch-pc/lab06$  
bobo@bergeshov:~/work/arch-pc/lab06$ nasm -f elf program.asm  
bobo@bergeshov:~/work/arch-pc/lab06$ ld -m elf_i386 program.o -o program  
bobo@bergeshov:~/work/arch-pc/lab06$ ./program  
Введите X  
1  
выражение = : 4  
bobo@bergeshov:~/work/arch-pc/lab06$ ./program  
Введите X  
10  
выражение = : 10  
bobo@bergeshov:~/work/arch-pc/lab06$
```

Рис. 2.18: Компиляция и запуск программы program.asm

Программа считает верно.

## **3 Выводы**

Изучили работу с арифметическими операциями.