

# **Отчёт по лабораторной работе 6**

**Компьютерные науки и технологии программирования. Раздел  
Архитектура компьютеров**

Фахми Джакси Гамал Адли

# Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	21

## Список иллюстраций

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

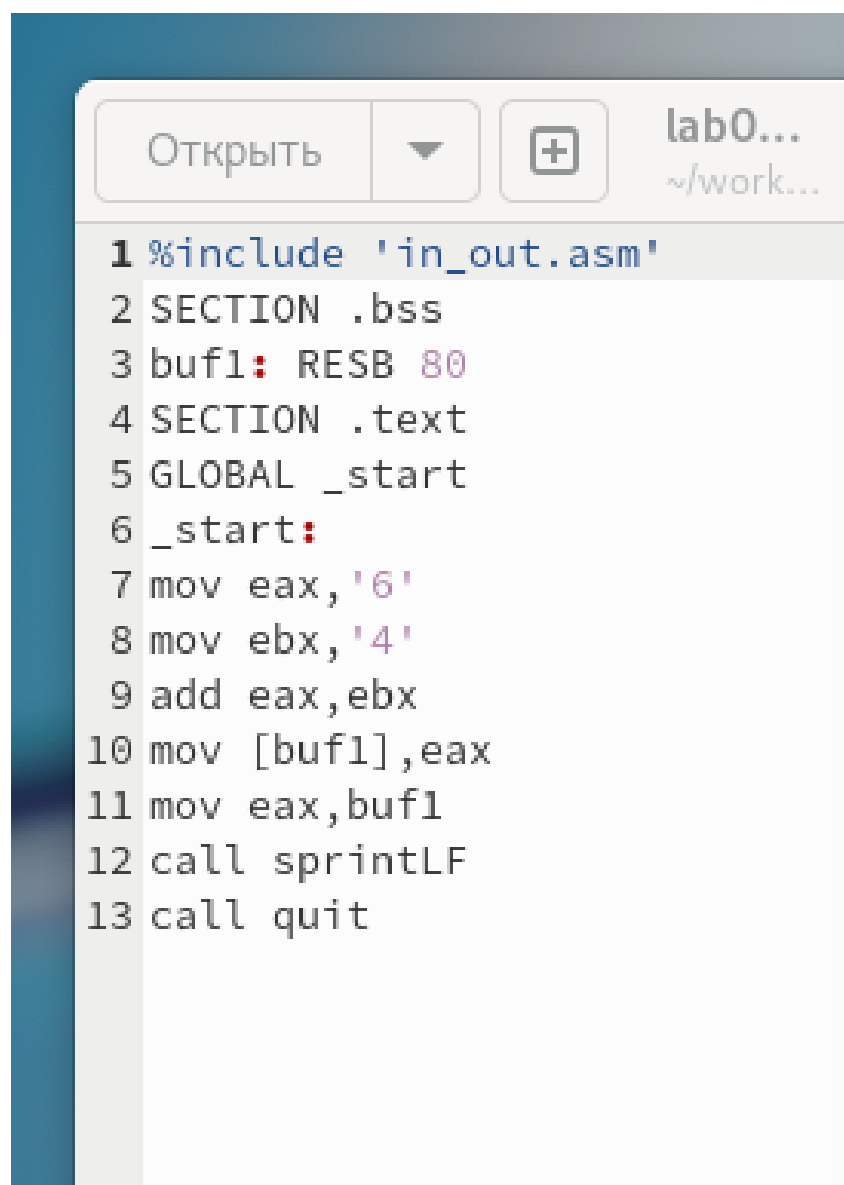
## Список таблиц

# 1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

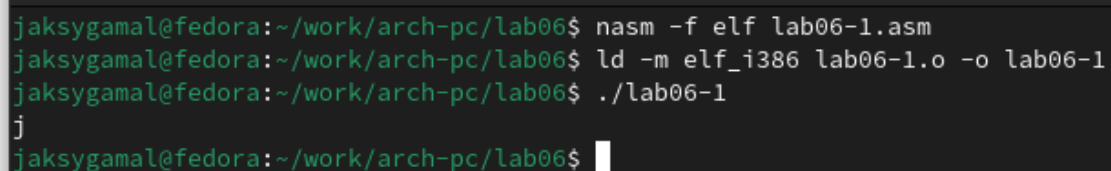
## 2 Выполнение лабораторной работы

1. Создал каталог для программ лабораторной работы № 6, перешел в него и создал файл lab6-1.asm.
2. Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения, записанные в регистр eax.



```
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 sprintf
13 call quit
```

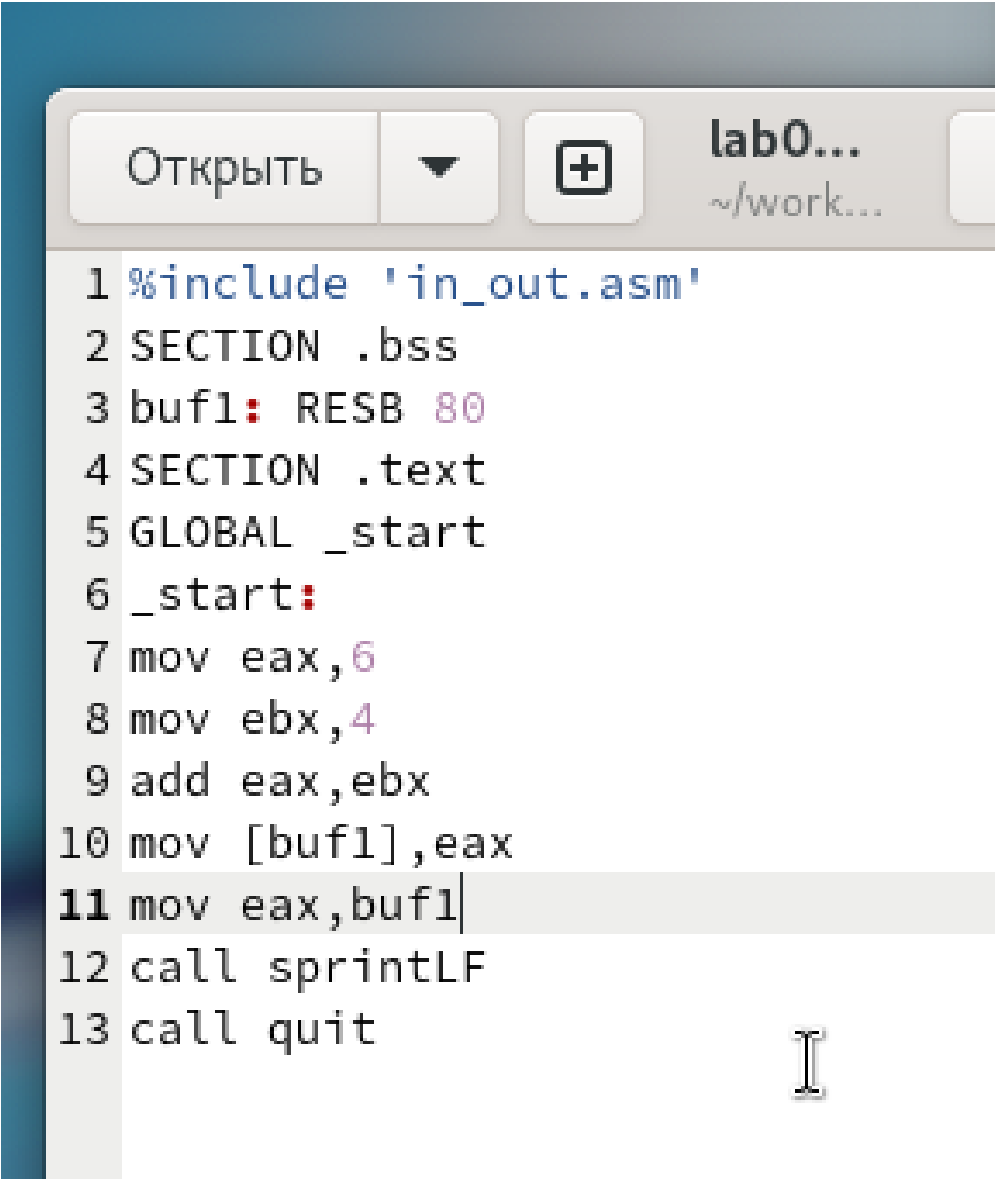
Рис. 2.1: Программа lab6-1.asm



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

Рис. 2.2: Запуск программы lab6-1.asm

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



```
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: Программа lab6-1.asm



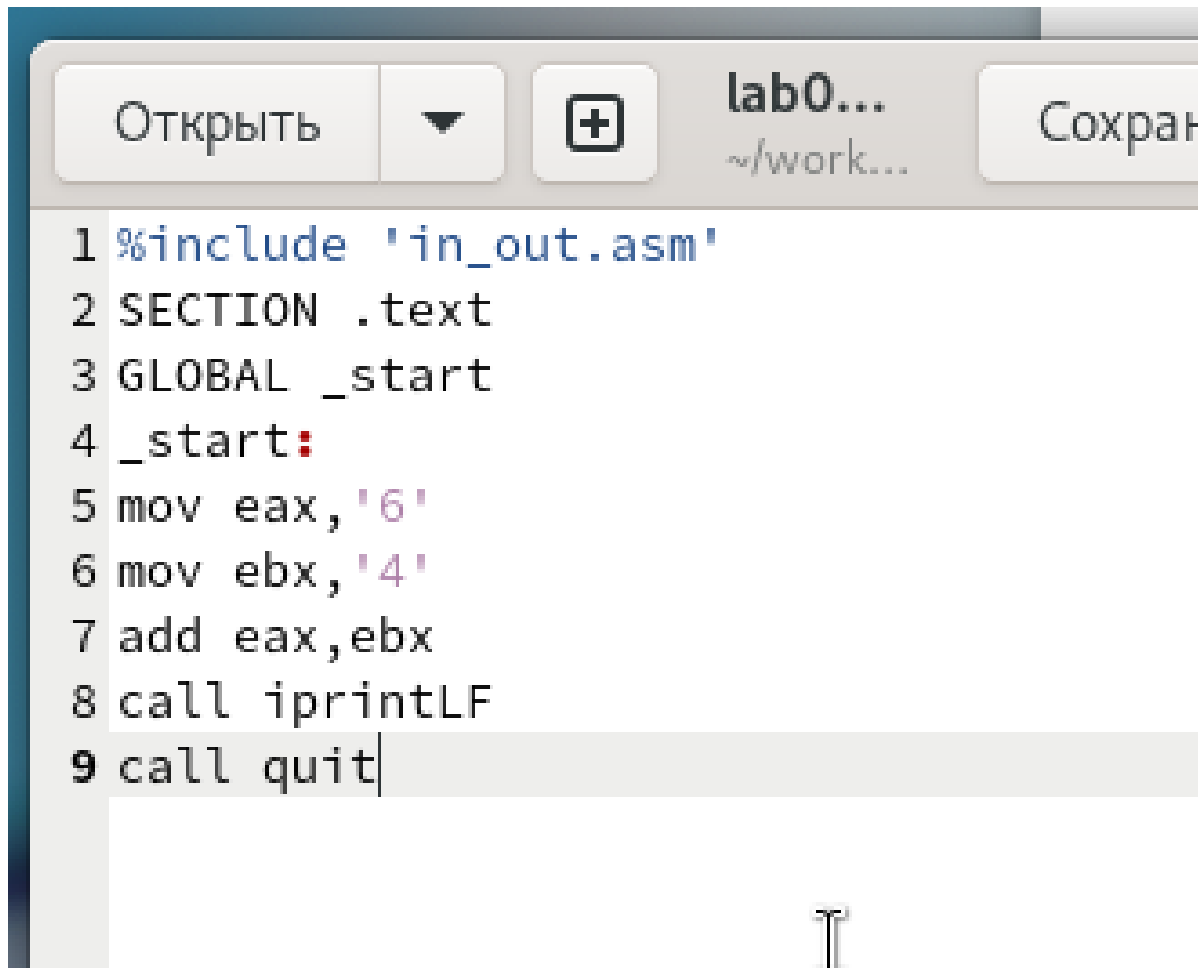
```
jaksygamal@fedora:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm
jaksygamal@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o -o lab06-1
jaksygamal@fedora:~/work/arch-pc/lab06$ ./lab06-1

jaksygamal@fedora:~/work/arch-pc/lab06$
```

Рис. 2.4: Запуск программы lab6-1.asm

Никакой символ не виден, но он есть. Это возврат каретки LF.

4. Как отмечалось выше, для работы с числами в файле in\_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразовал текст программы с использованием этих функций.



```
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
```

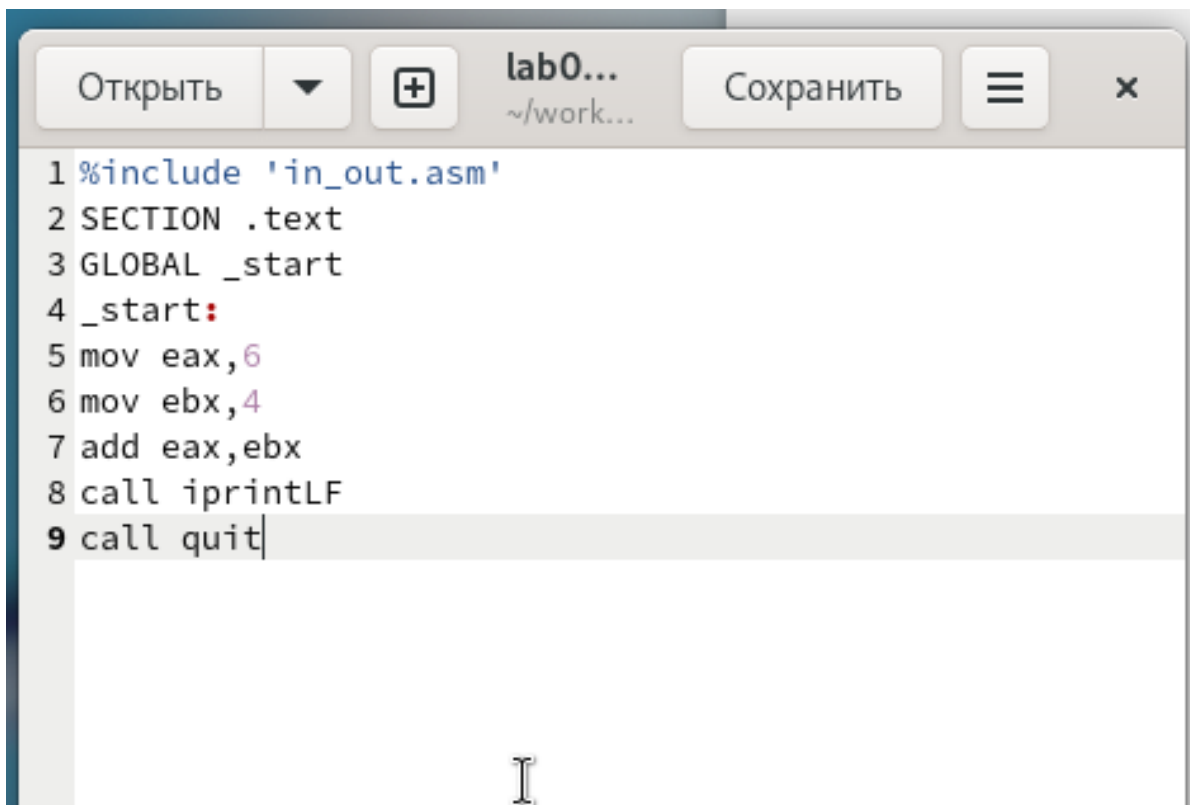
Рис. 2.5: Программа lab6-2.asm

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

Рис. 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 вывести число, а не символ, кодом которого является это число.

5. Аналогично предыдущему примеру изменим 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
```

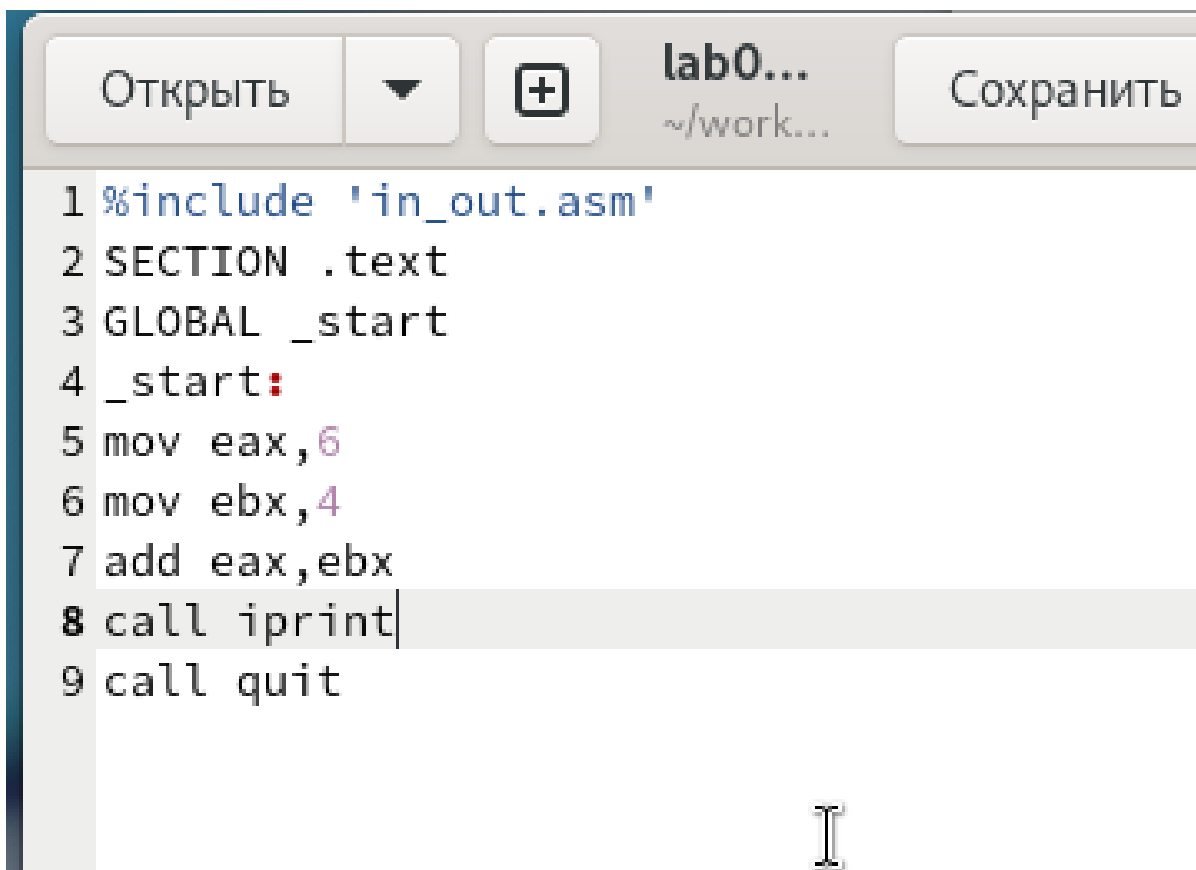
Рис. 2.7: Программа lab6-2.asm

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

```
jaksygama1@fedora:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm
jaksygama1@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-2.o -o lab06-2
jaksygama1@fedora:~/work/arch-pc/lab06$ ./lab06-2
10
jaksygama1@fedora:~/work/arch-pc/lab06$
```

Рис. 2.8: Запуск программы lab6-2.asm

Заменял функцию `iprintLF` на `iprint`. 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: Программа lab6-2.asm

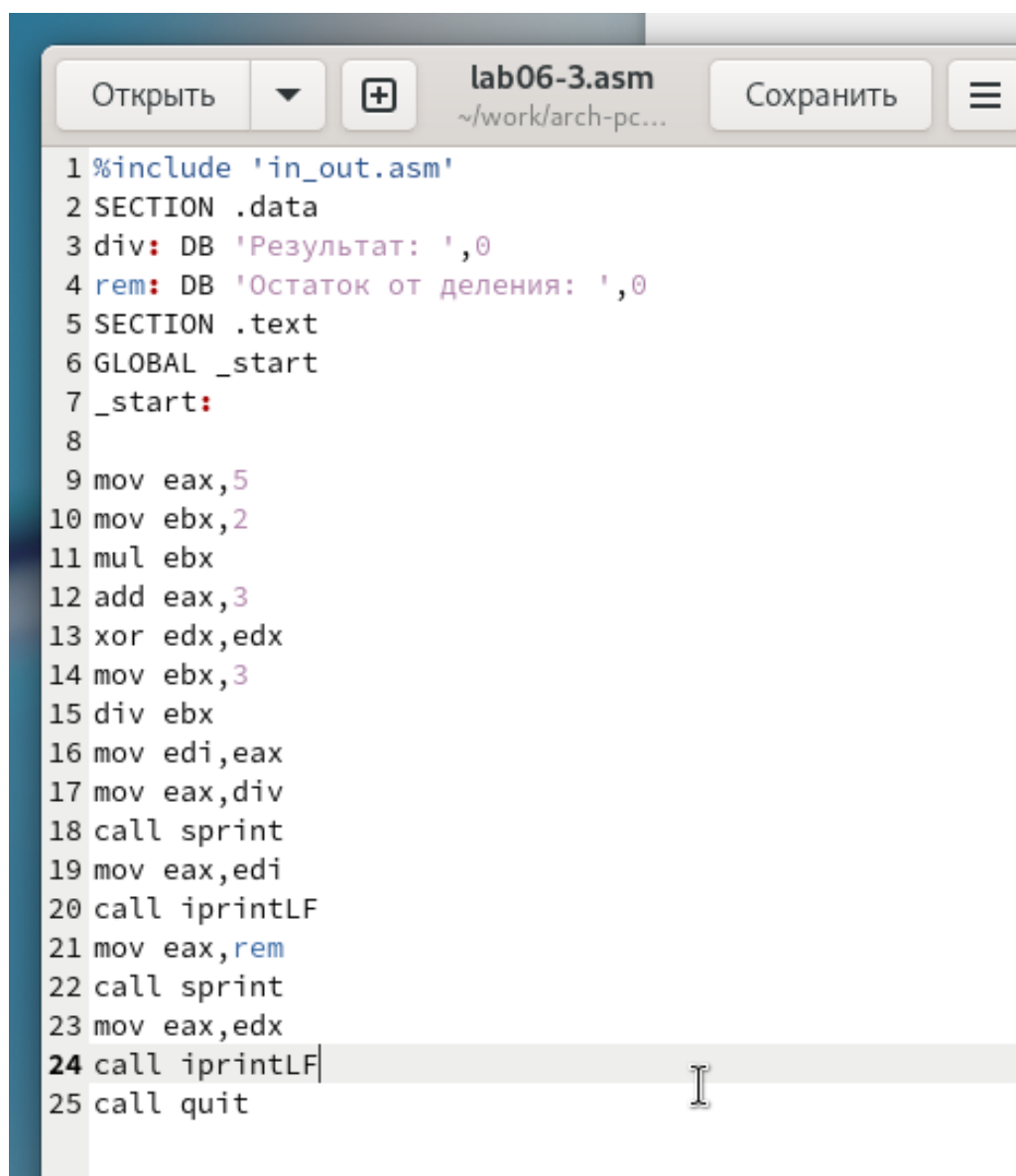
```
jaksygama1@fedora:~/work/arch-pc/lab06$  
jaksygama1@fedora:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm  
jaksygama1@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-2.o -o lab06-2  
jaksygama1@fedora:~/work/arch-pc/lab06$ ./lab06-2  
10jaksygama1@fedora:~/work/arch-pc/lab06$  
jaksygama1@fedora:~/work/arch-pc/lab06$
```

Рис. 2.10: Запуск программы lab6-2.asm

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

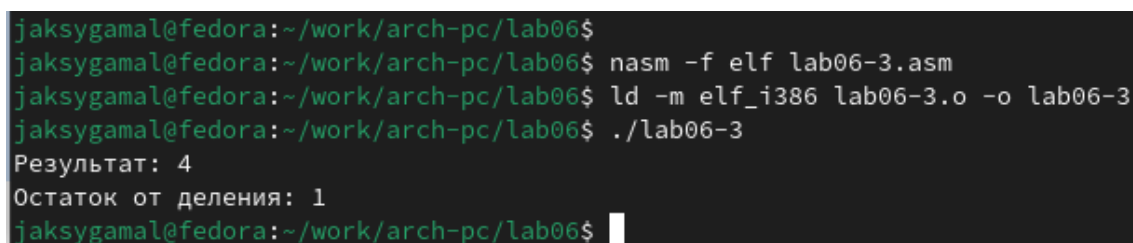
$$f(x) = (5 * 2 + 3) / 3$$

.



```
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: Программа lab6-3.asm



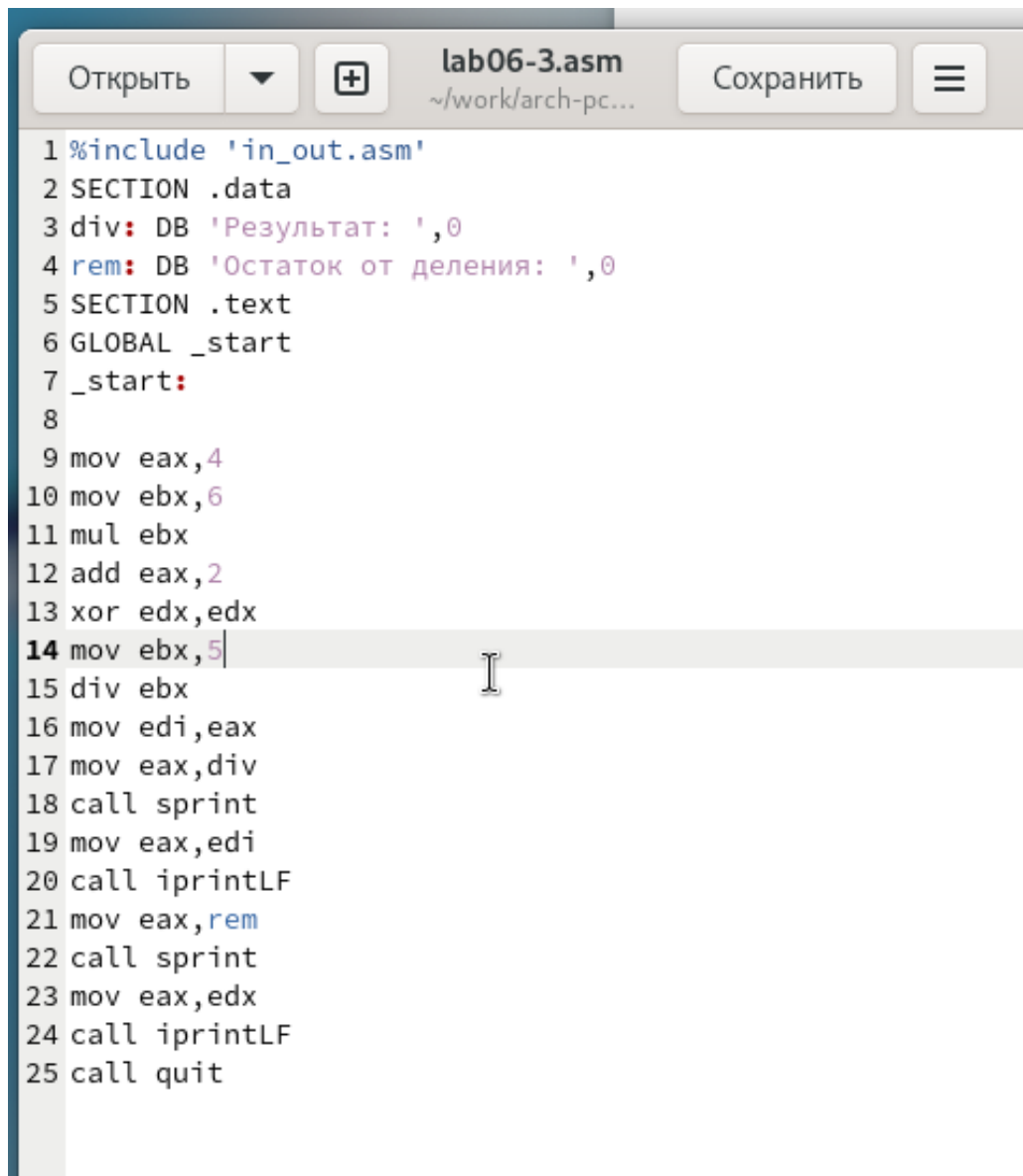
```
jaksygamal@fedora:~/work/arch-pc/lab06$
jaksygamal@fedora:~/work/arch-pc/lab06$ nasm -f elf lab06-3.asm
jaksygamal@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-3.o -o lab06-3
jaksygamal@fedora:~/work/arch-pc/lab06$ ./lab06-3
Результат: 4
Остаток от деления: 1
jaksygamal@fedora:~/work/arch-pc/lab06$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

Рис. 2.12: Запуск программы lab6-3.asm

Изменил текст программы для вычисления выражения

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

. Создал исполняемый файл и проверил его работу.



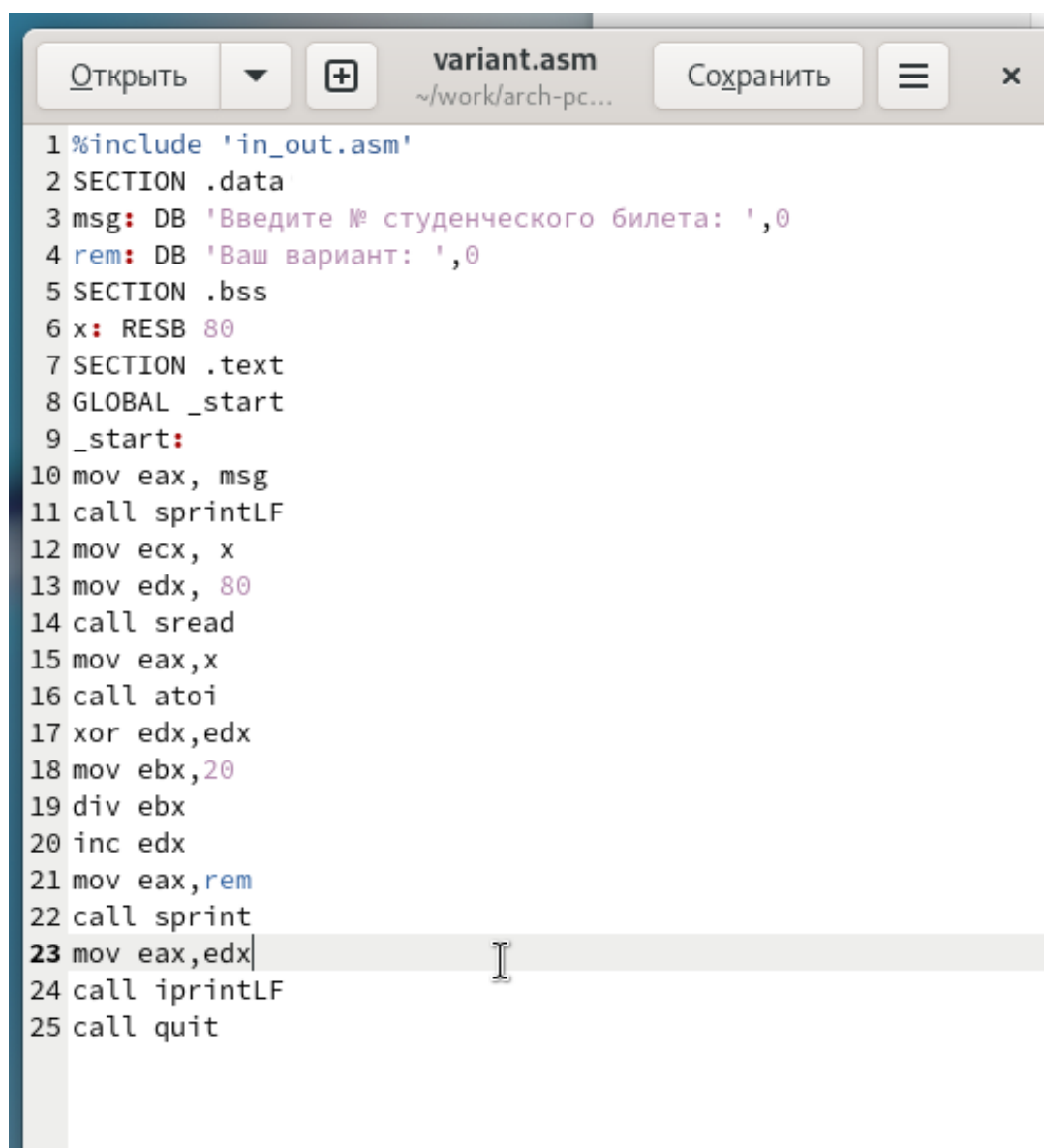
```
1 %include 'in_out.asm'
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8
9 mov eax,4
10 mov ebx,6
11 mul ebx
12 add eax,2
13 xor edx,edx
14 mov ebx,5
15 div ebx
16 mov edi,eax
17 mov eax,div
18 call sprint
19 mov eax,edi
20 call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

Рис. 2.13: Программа lab6-3.asm

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

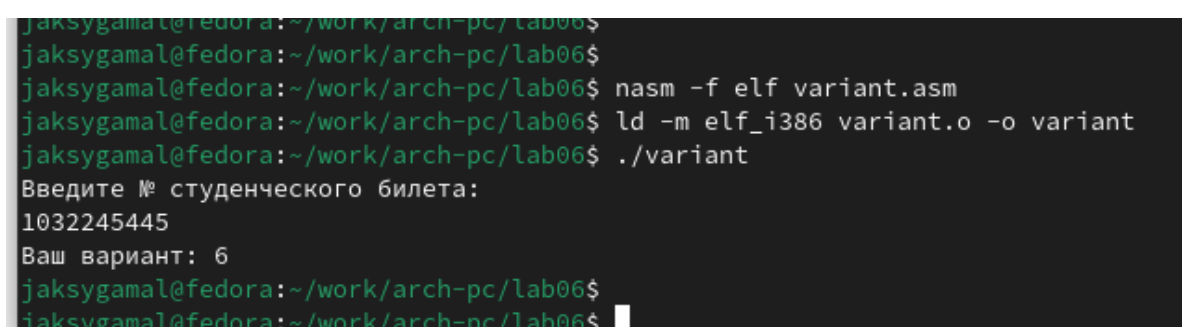
Рис. 2.14: Запуск программы lab6-3.asm

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



```
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: Программа variant.asm



```
jaksygamal@fedora:~/work/arch-pc/lab06$
jaksygamal@fedora:~/work/arch-pc/lab06$
jaksygamal@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
jaksygamal@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 variant.o -o variant
jaksygamal@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032245445
Ваш вариант: 6
jaksygamal@fedora:~/work/arch-pc/lab06$
jaksygamal@fedora:~/work/arch-pc/lab06$
```

Рис. 2.16: Запуск программы variant.asm



## ответы на вопросы

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

В строке `mov eax,ret` значение переменной с фразой 'Ваш вариант:' перекладывается в регистр `eax`.

Строка `call sprint` вызывает подпрограмму для вывода строки.

2. Для чего используются следующие инструкции?

```
mov ecx, x mov edx, 80 call sread
```

`mov ecx, x` - перемещает значение переменной `X` в регистр `ecx`.

`mov edx, 80` - устанавливает значение 80 в регистр `edx`.

`call sread` - вызывает подпрограмму для чтения значения с консоли.

3. Для чего используется инструкция "call atoi"?

Эта инструкция вызывает подпрограмму, которая преобразует введенные символы в числовой формат.

4. Какие строки листинга отвечают за вычисления варианта?

```
xor edx,edx
```

 - обнуляет регистр `edx`.

```
mov ebx,20
```

 - устанавливает значение 20 в регистр `ebx`.

```
div ebx
```

 - производит деление номера студенческого билета на 20.

```
inc edx
```

 - увеличивает значение регистра `edx` на 1.

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

Остаток от деления записывается в регистр `edx`.

6. Для чего используется инструкция "inc edx"?

Инструкция `inc edx` увеличивает значение 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?

`mov eax,edx` - результат вычислений перекладывается в регистр `eax`.

`call iprintLF` - вызывается подпрограмма для вывода результата на экран.

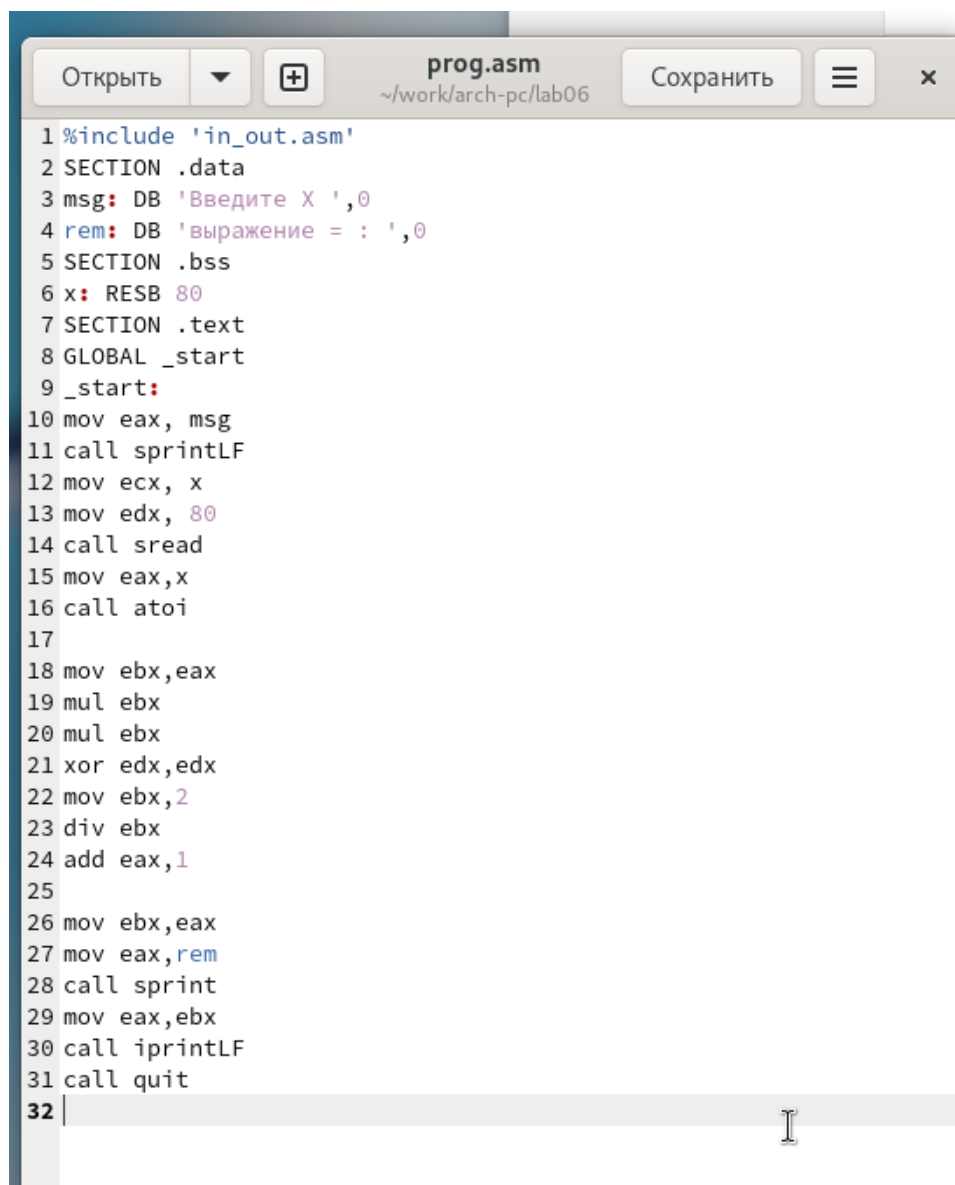
8. Написать программу вычисления выражения  $y = f(x)$ . Программа должна выводить выражение для вычисления, выводить запрос на ввод значения  $x$ , вычислять заданное выражение в зависимости от введенного  $x$ , выводить результат вычислений. Вид функции  $f(x)$  выбрать из таблицы 6.3 вариантов заданий 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 6 -

$$x^3/2 + 1$$

для

$$x = 2, x = 5$$



```
1 %include 'in_out.asm'
2 SECTION .data
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 sprintLF
12 mov ecx, x
13 mov edx, 80
14 call sread
15 mov eax, x
16 call atoi
17
18 mov ebx, eax
19 mul ebx
20 mul ebx
21 xor edx, edx
22 mov ebx, 2
23 div ebx
24 add eax, 1
25
26 mov ebx, eax
27 mov eax, rem
28 call sprint
29 mov eax, ebx
30 call iprintLF
31 call quit
32 |
```

Рис. 2.17: Программа prog.asm

```
jaksygamal@fedora:~/work/arch-pc/lab06$  
jaksygamal@fedora:~/work/arch-pc/lab06$ nasm -f elf prog.asm  
jaksygamal@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 prog.o -o prog  
jaksygamal@fedora:~/work/arch-pc/lab06$ ./prog  
Введите X  
2  
выражение = : 5  
jaksygamal@fedora:~/work/arch-pc/lab06$ ./prog  
Введите X  
5  
выражение = : 63  
jaksygamal@fedora:~/work/arch-pc/lab06$
```

Рис. 2.18: Запуск программы prog.asm

## **3 Выводы**

Изучили работу с арифметическими операциями.