

# **Отчёт по лабораторной работе 6**

**Архитектура компьютеров**

Ермишина Мария Кирилловна

# 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
2.3	Ответы на вопросы . . . . .	17
2.4	Задание для самостоятельной работы . . . . .	19
<b>3</b>	<b>Выводы</b>	<b>22</b>

## 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 . . . . .	10
2.6	Запуск программы lab6-2.asm . . . . .	10
2.7	Программа lab6-2.asm . . . . .	11
2.8	Запуск программы lab6-2.asm . . . . .	11
2.9	Запуск программы lab6-2.asm . . . . .	12
2.10	Программа lab6-3.asm . . . . .	13
2.11	Запуск программы lab6-3.asm . . . . .	13
2.12	Программа lab6-3.asm . . . . .	14
2.13	Запуск программы lab6-3.asm . . . . .	15
2.14	Программа variant.asm . . . . .	16
2.15	Запуск программы variant.asm . . . . .	17
2.16	Программа calc.asm . . . . .	20
2.17	Запуск программы calc.asm . . . . .	21

## Список таблиц

# 1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

## 2 Выполнение лабораторной работы

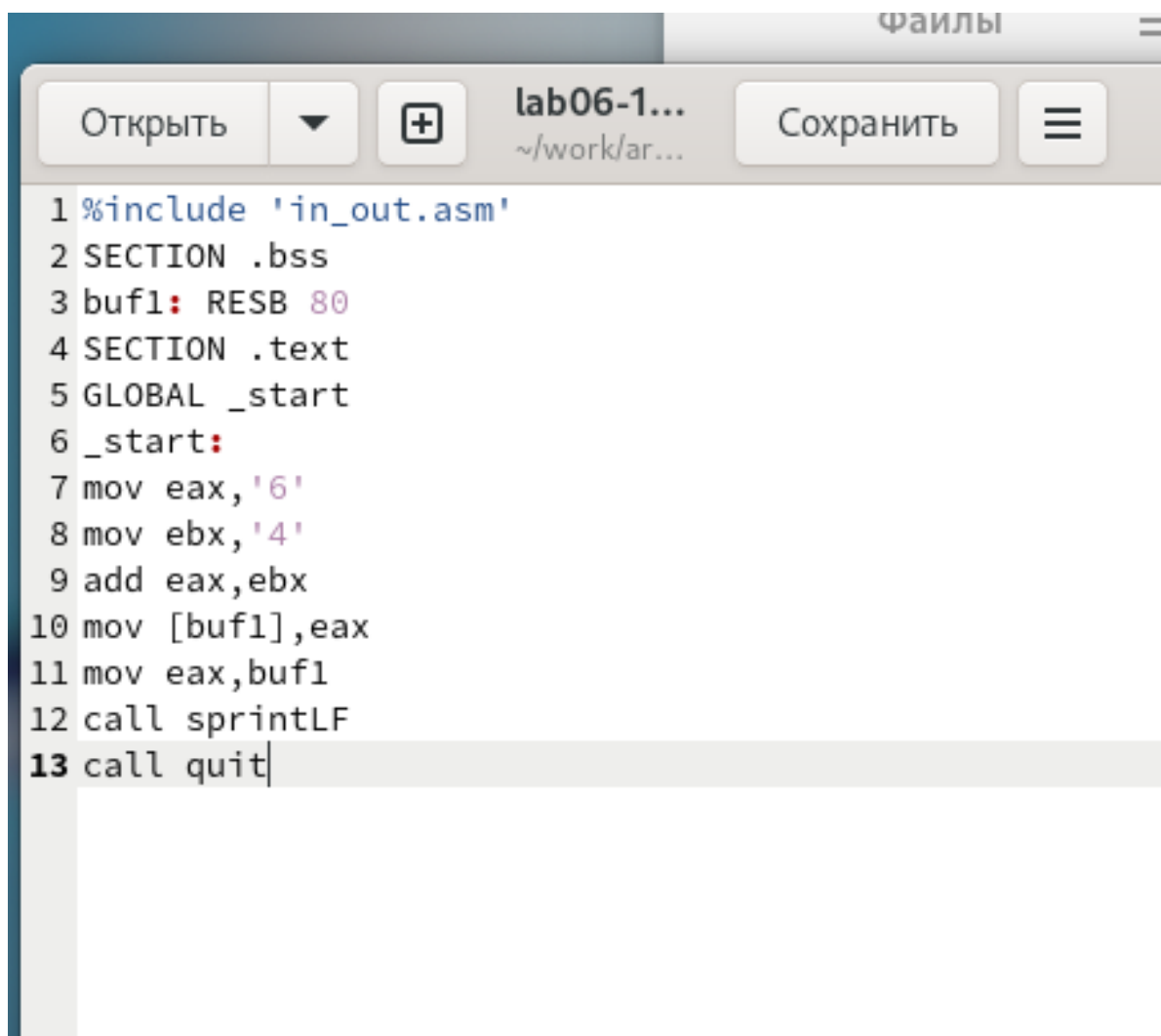
### 2.1 Символьные и численные данные в NASM

Создаю каталог для программ лабораторной работы № 6, перехожу в него и создаю файл lab6-1.asm.

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

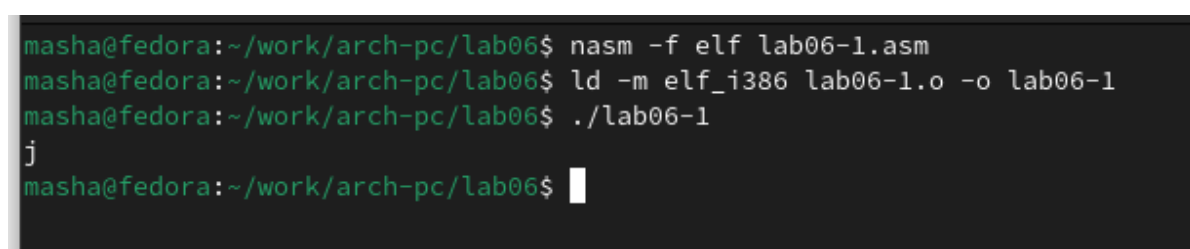
В данной программе в регистр `eax` записывается символ 6 (`mov eax,'6'`), в регистр `ebx` символ 4 (`mov ebx,'4'`). Далее к значению в регистре `eax` прибавляем значение регистра `ebx` (`add eax,ebx`, результат сложения запишется в регистр `eax`). Далее выводим результат. (рис. 2.1) (рис. 2.2)

Так как для работы функции `sprintf` в регистр `eax` должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра `eax` в переменную `buf1` (`mov [buf1],eax`), а затем запишем адрес переменной `buf1` в регистр `eax` (`mov eax,buf1`) и вызовем 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: Программа lab6-1.asm



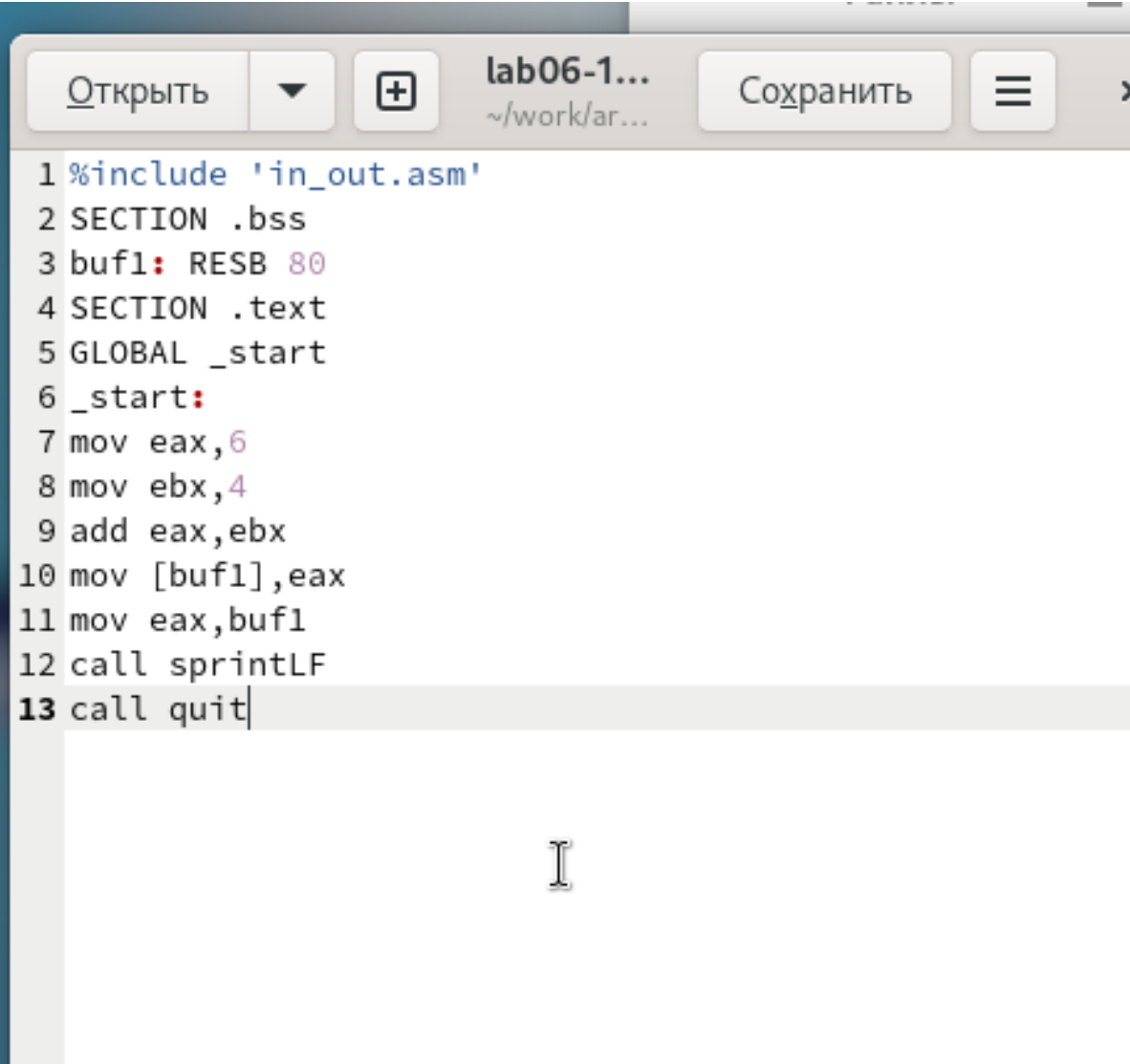
```
masha@fedora:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm
masha@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o -o lab06-1
masha@fedora:~/work/arch-pc/lab06$ ./lab06-1
j
masha@fedora:~/work/arch-pc/lab06$
```

Рис. 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`.

Далее изменяю текст программы и вместо символов, запишем в регистры числа. (рис. 2.3) (рис. 2.4)



```
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. 2.3: Программа `lab6-1.asm`

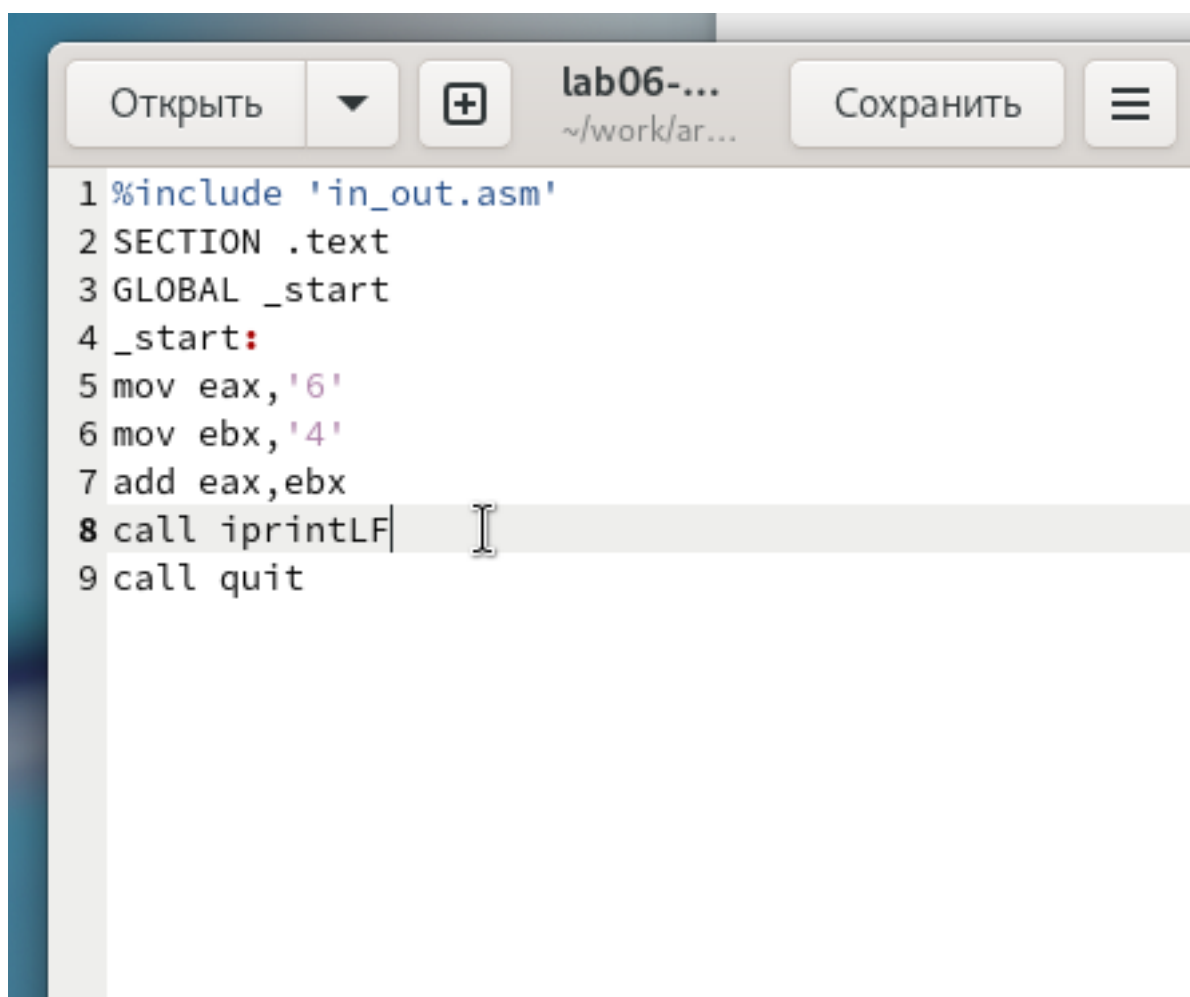


```
masha@fedora:~/work/arch-pc/lab06$  
masha@fedora:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm  
masha@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o -o lab06-1  
masha@fedora:~/work/arch-pc/lab06$ ./lab06-1  
  
masha@fedora:~/work/arch-pc/lab06$ █
```

Рис. 2.4: Запуск программы lab6-1.asm

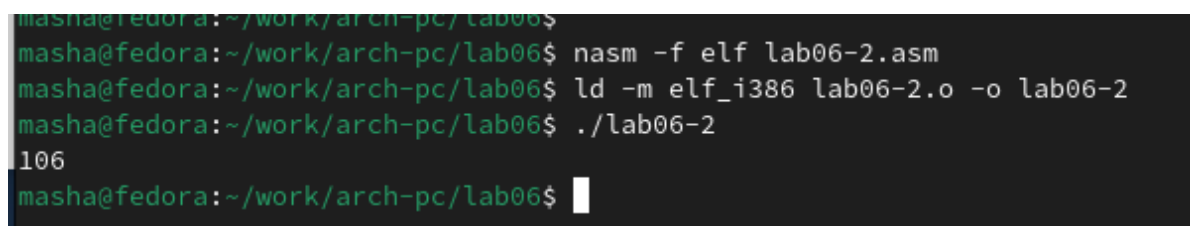
Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10. Это символ конца строки (возврат каретки). В консоле он не отображается, но добавляет пустую строку.

Как отмечалось выше, для работы с числами в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразую текст программы с использованием этих функций. (рис. 2.5) (рис. 2.6)



```
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

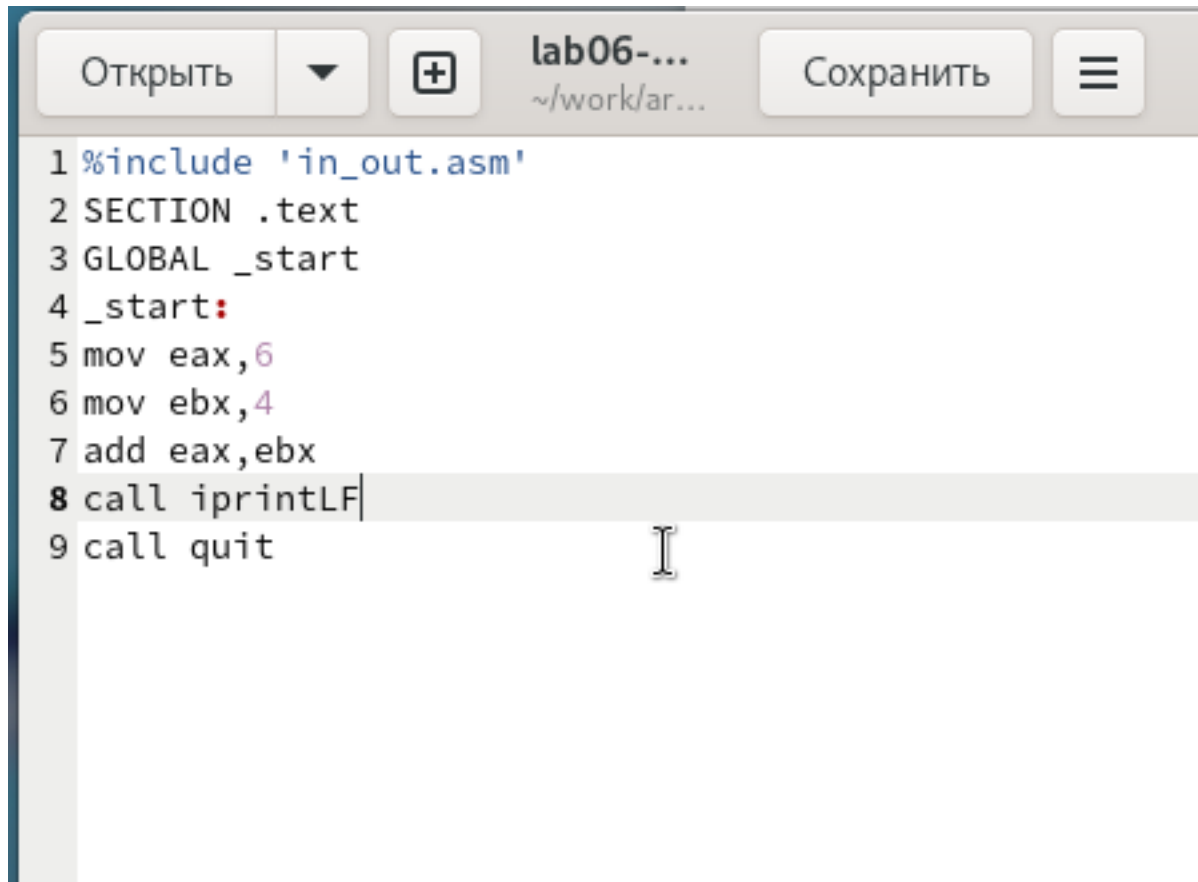


```
masha@fedora:~/work/arch-pc/lab06$
masha@fedora:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm
masha@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-2.o -o lab06-2
masha@fedora:~/work/arch-pc/lab06$ ./lab06-2
106
masha@fedora:~/work/arch-pc/lab06$
```

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.

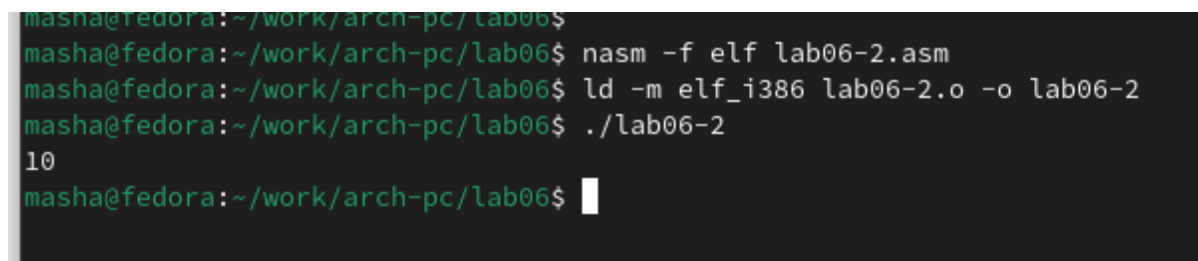
Аналогично предыдущему примеру изменим символы на числа. (рис. 2.7) (рис. 2.8)



```
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.



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

Рис. 2.8: Запуск программы lab6-2.asm

Заменяла функцию iprintLF на iprint. Создала исполняемый файл и запустила

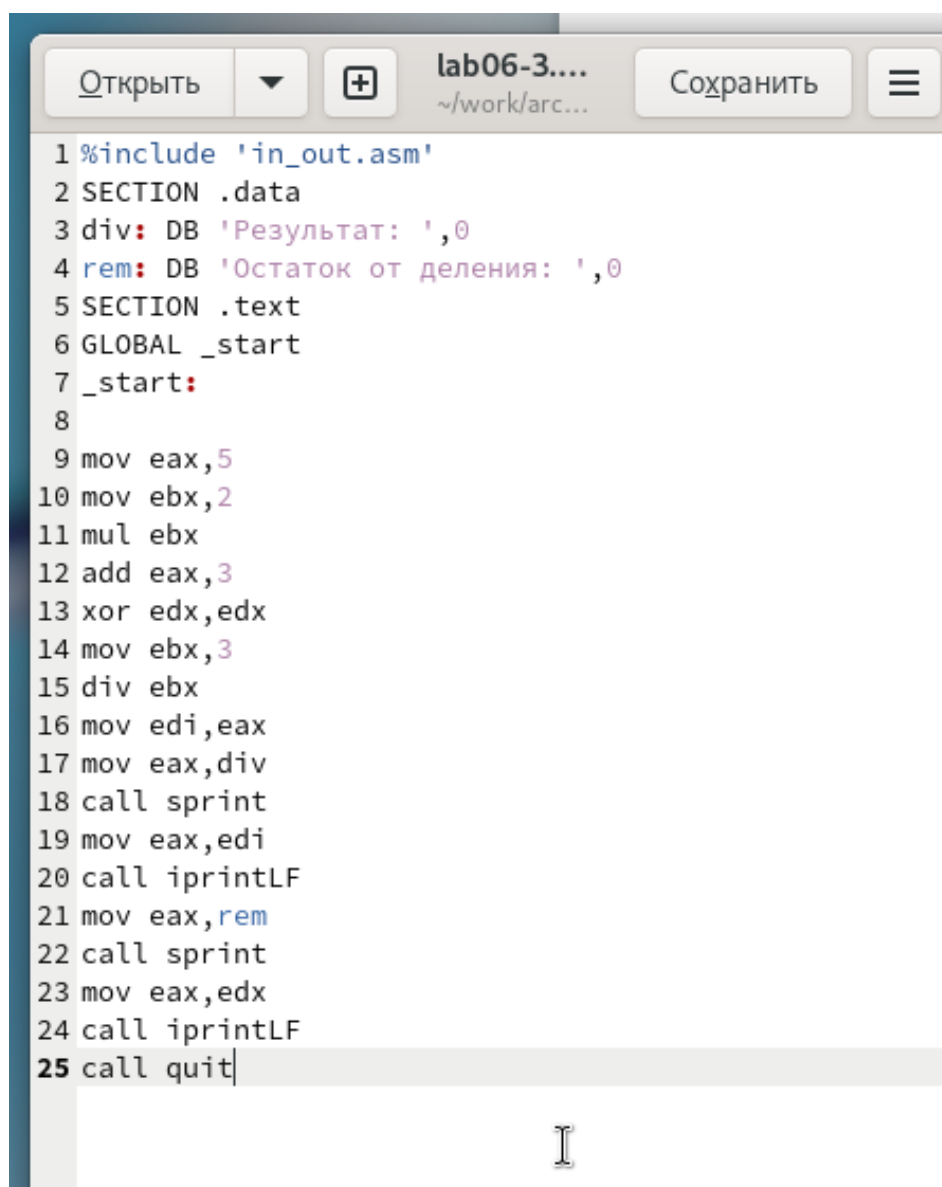
его. Вывод отличается тем, что нет переноса строки. (рис. 2.9)

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

Рис. 2.9: Запуск программы lab6-2.asm

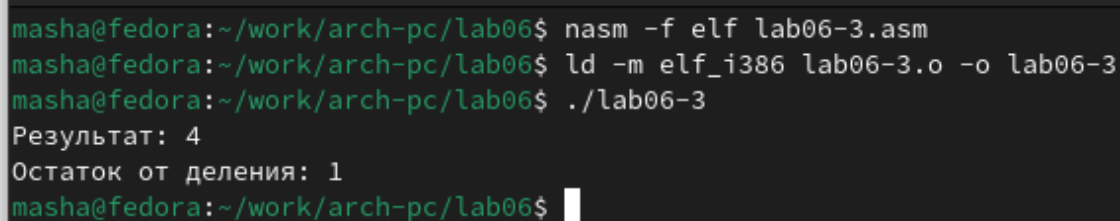
## 2.2 Выполнение арифметических операций в NASM

В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения  $f(x) = (5 * 2 + 3) / 3$ . (рис. 2.10) (рис. 2.11)



```
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
```

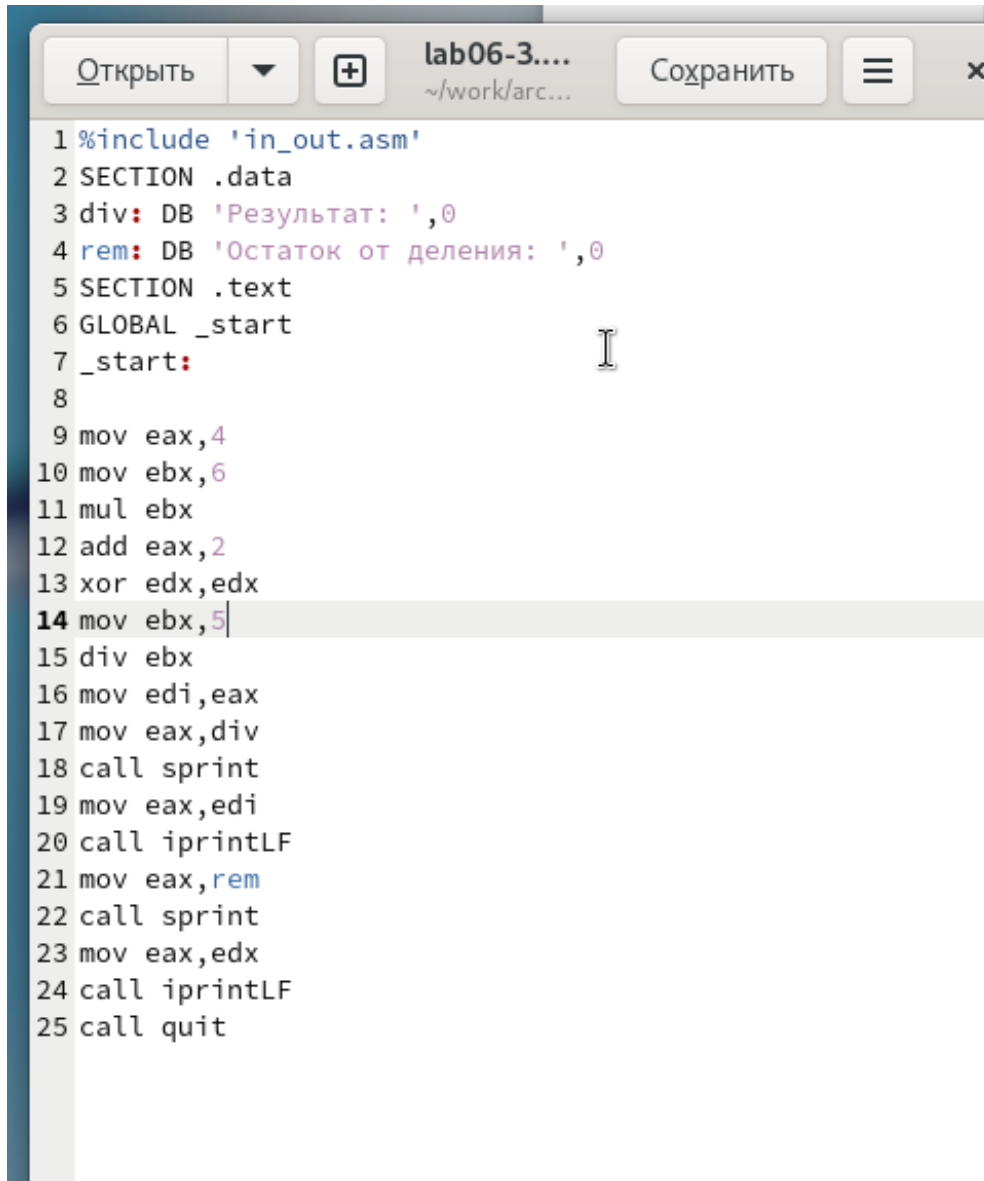
Рис. 2.10: Программа lab6-3.asm



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

Рис. 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. (рис. 2.12) (рис. 2.13)



```
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
```

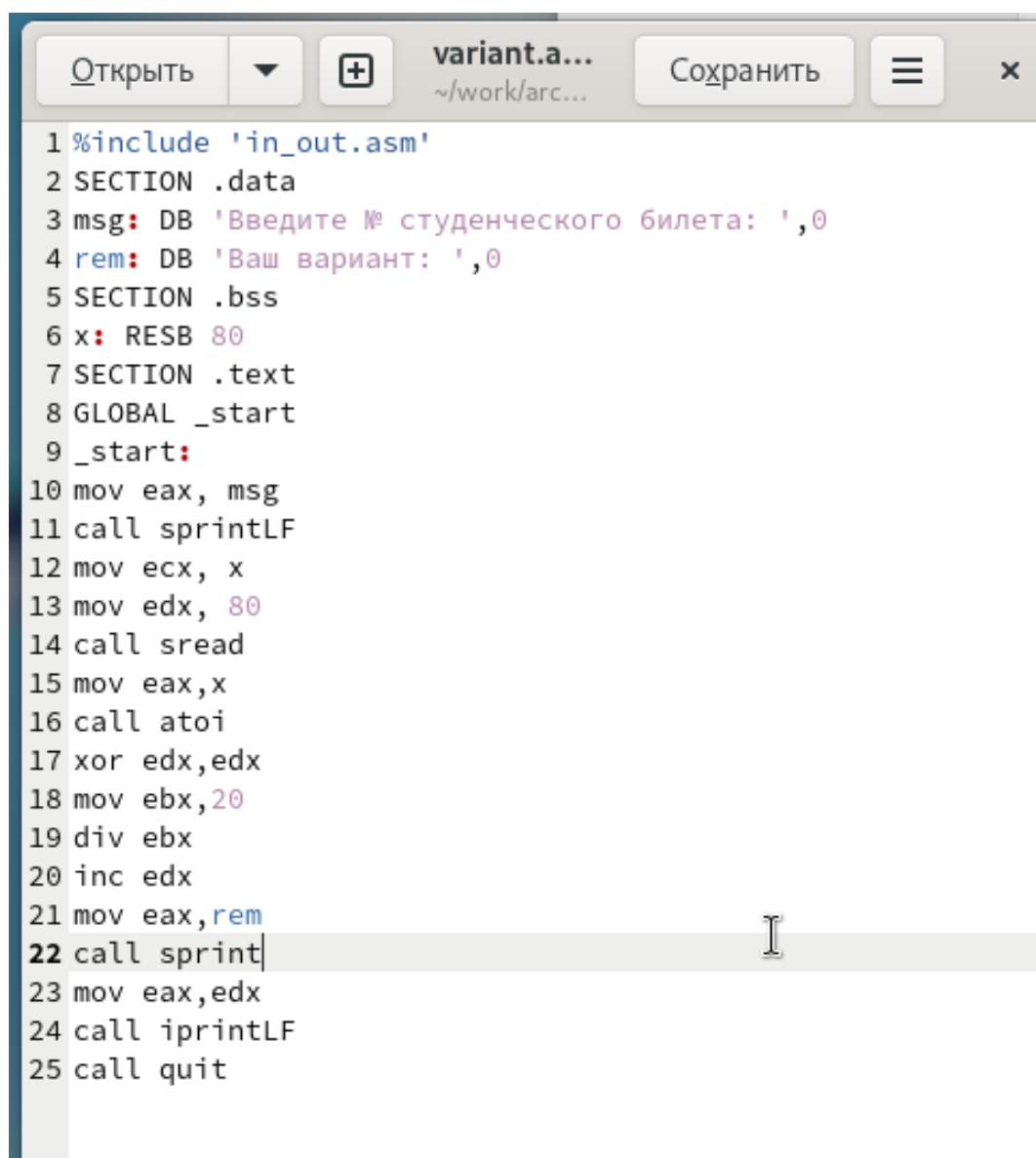
Рис. 2.12: Программа lab6-3.asm

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

Рис. 2.13: Запуск программы lab6-3.asm

В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета. (рис. 2.14) (рис. 2.15)

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`.



```
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 sprintLF
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 sprint
23 mov eax, edx
24 call iprintLF
25 call quit
```

Рис. 2.14: Программа variant.asm



```
masha@fedora:~/work/arch-pc/lab06$  
masha@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm  
masha@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 variant.o -o variant  
masha@fedora:~/work/arch-pc/lab06$ ./variant  
Введите № студенческого билета:  
1132230166  
Ваш вариант: 7  
masha@fedora:~/work/arch-pc/lab06$
```

Рис. 2.15: Запуск программы variant.asm

## 2.3 Ответы на вопросы

1. Какие строки листинга отвечают за вывод на экран сообщения ‘Ваш вариант:’?
  - Инструкция “mov eax, mem” перекладывает значение переменной с фразой ‘Ваш вариант:’ в регистр eax.
  - Инструкция “call sprint” вызывает подпрограмму для вывода строки.
2. Для чего используются следующие инструкции?
  - Инструкция “mov ecx, x” используется для перемещения значения переменной x в регистр ecx.
  - Инструкция “mov edx, 80” используется для перемещения значения 80 в регистр edx.
  - Инструкция “call sread” вызывает подпрограмму для считывания значения студенческого билета 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?

- Инструкция “xor edx, edx” обнуляет регистр edx.
- Инструкция “mov ebx, 20” записывает значение 20 в регистр ebx.
- Инструкция “div ebx” выполняет деление номера студенческого билета на 20.
- Инструкция “inc edx” увеличивает значение регистра edx на 1.

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”?

- Остаток от деления записывается в регистр edx.

6. Для чего используется инструкция “inc edx”?

- Инструкция “inc edx” используется для увеличения значения в регистре edx на 1, согласно формуле вычисления варианта.

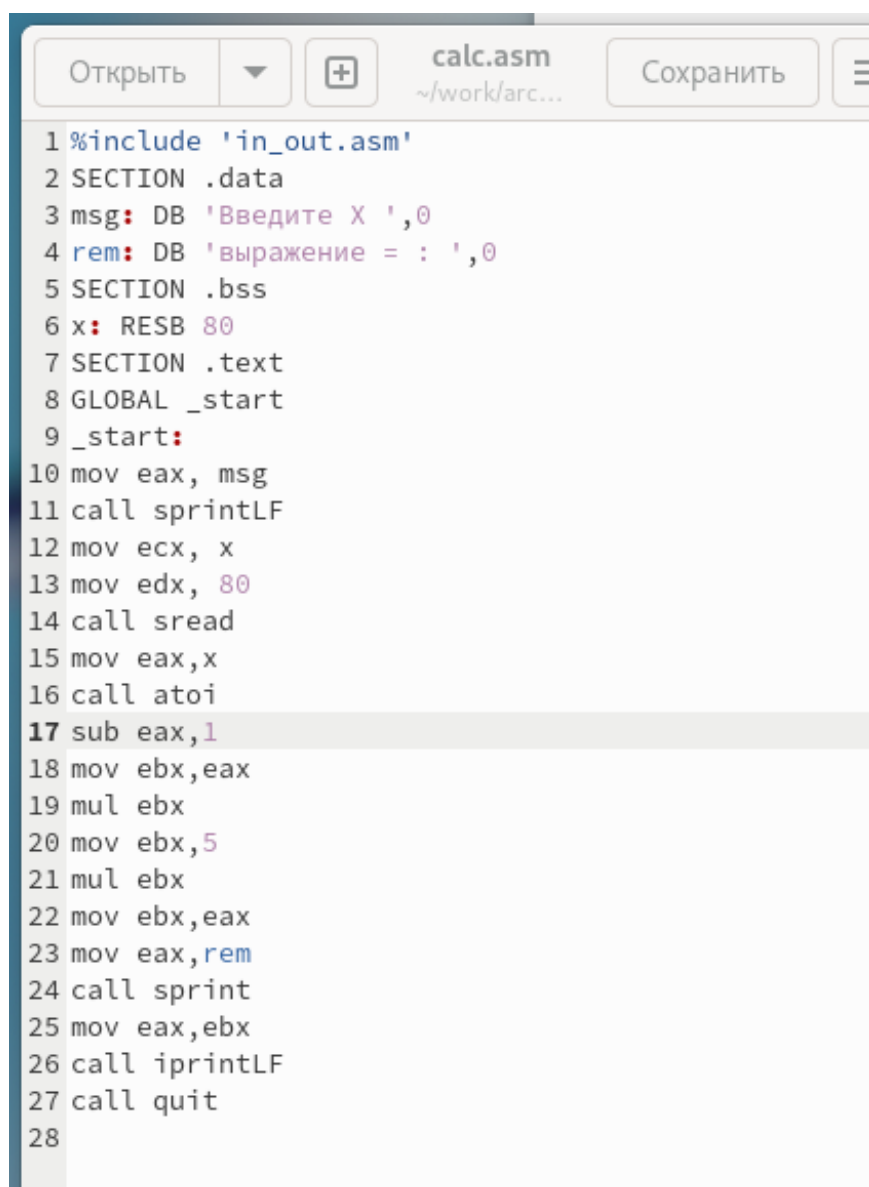
7. Какие строки листинга отвечают за вывод на экран результата вычислений?

- Инструкция “mov eax, edx” перекладывает результат вычислений в регистр eax.
- Инструкция “call iprintLF” вызывает подпрограмму для вывода значения на экран.

## 2.4 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. (рис. 2.16) (рис. 2.17)

Получили вариант  $7 - 5(x - 1)^2$  для  $x = 3, x = 5$



```
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 sprintf
12 mov ecx, x
13 mov edx, 80
14 call sread
15 mov eax, x
16 call atoi
17 sub eax, 1
18 mov ebx, eax
19 mul ebx
20 mov ebx, 5
21 mul ebx
22 mov ebx, eax
23 mov eax, rem
24 call sprintf
25 mov eax, ebx
26 call iprintLF
27 call quit
28
```

Рис. 2.16: Программа calc.asm

При  $x = 3$  получается 20.

При  $x = 5$  получается 80.

```
masha@fedora:~/work/arch-pc/lab06$  
masha@fedora:~/work/arch-pc/lab06$ nasm -f elf calc.asm  
masha@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 calc.o -o calc  
masha@fedora:~/work/arch-pc/lab06$ ./calc  
Введите X  
3  
выражение = : 20  
masha@fedora:~/work/arch-pc/lab06$ ./calc  
Введите X  
5  
выражение = : 80  
masha@fedora:~/work/arch-pc/lab06$
```

Рис. 2.17: Запуск программы calc.asm

Программа считает верно.

## **3 Выводы**

Изучили работу с арифметическими операциями.