

# **Отчёт по лабораторной работе 6**

**Дисциплина: архитектура компьютера**

Маныев Ресулбег Алексеевич

# 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 . . . . .	15
2.3	Задание для самостоятельной работы . . . . .	21
<b>3</b>	<b>Выводы</b>	<b>24</b>

## Список иллюстраций

2.1	Код программы lab6-1.asm . . . . .	7
2.2	Компиляция и запуск программы lab6-1.asm . . . . .	8
2.3	Код программы lab6-1.asm . . . . .	9
2.4	Компиляция и запуск программы lab6-1.asm . . . . .	10
2.5	Код программы lab6-2.asm . . . . .	11
2.6	Компиляция и запуск программы lab6-2.asm . . . . .	11
2.7	Код программы lab6-2.asm . . . . .	12
2.8	Компиляция и запуск программы lab6-2.asm . . . . .	13
2.9	Код программы lab6-2.asm . . . . .	14
2.10	Компиляция и запуск программы lab6-2.asm . . . . .	14
2.11	Код программы lab6-3.asm . . . . .	16
2.12	Компиляция и запуск программы lab6-3.asm . . . . .	16
2.13	Код программы lab6-3.asm . . . . .	17
2.14	Компиляция и запуск программы lab6-3.asm . . . . .	18
2.15	Код программы variant.asm . . . . .	19
2.16	Компиляция и запуск программы variant.asm . . . . .	20
2.17	Код программы program.asm . . . . .	22
2.18	Компиляция и запуск программы program.asm . . . . .	23

## Список таблиц

# 1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

## 2 Выполнение лабораторной работы

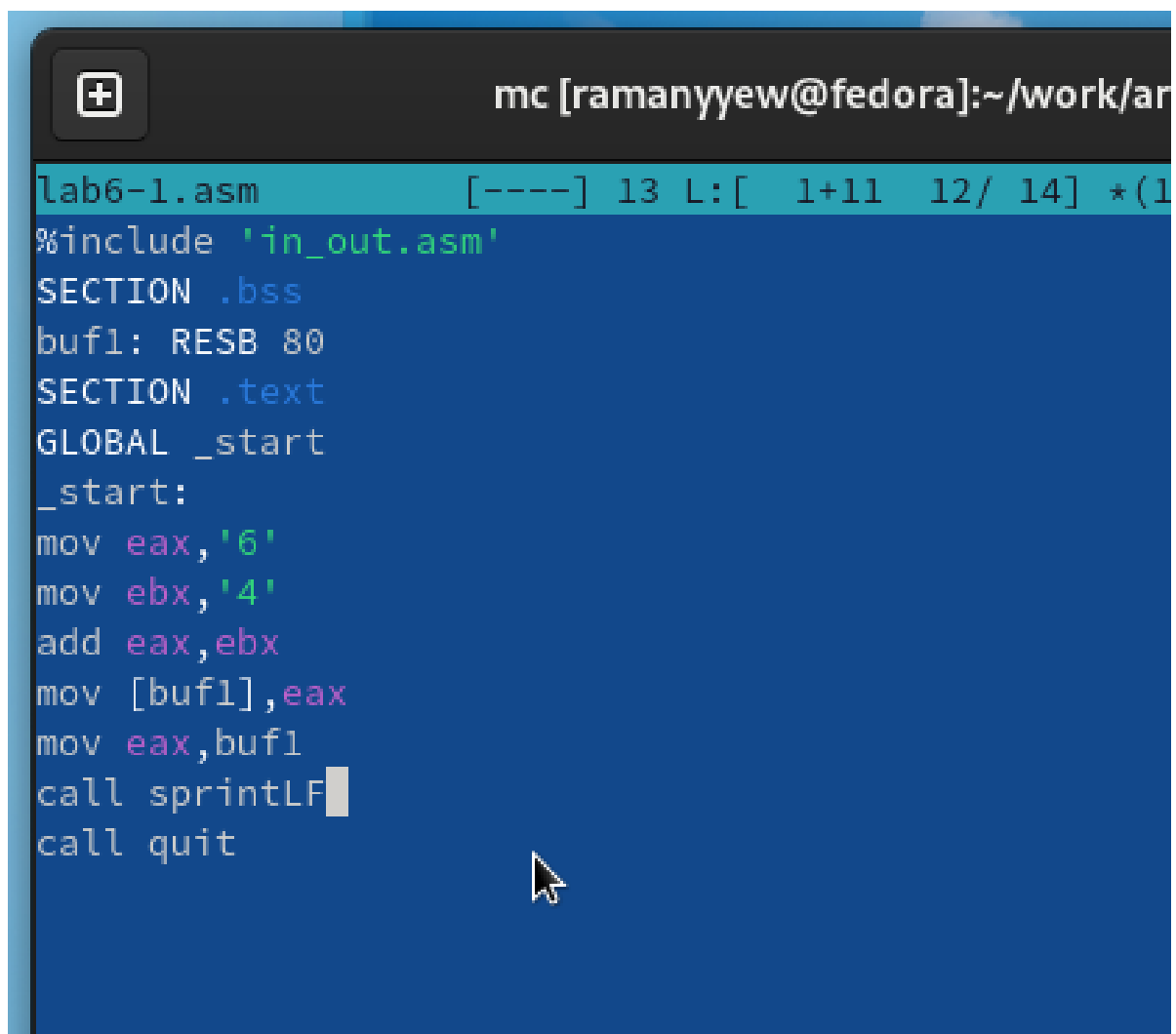
### 2.1 Символьные и численные данные в NASM

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

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

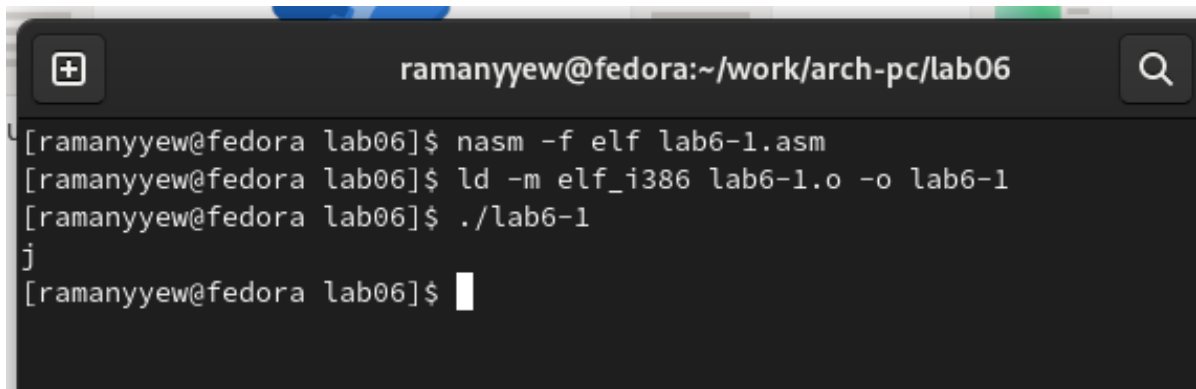
В данной программе (рис. [2.1]) в регистр `eax` записывается символ 6 (`mov eax, '6'`), в регистр `ebx` символ 4 (`mov ebx, '4'`). Далее к значению в регистре `eax` прибавляем значение регистра `ebx` (`add eax, ebx`, результат сложения запишется в регистр `eax`). Далее выводим результат. (рис. [2.2])

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`.



```
lab6-1.asm [----] 13 L: [ 1+11 12/ 14] *(1
#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: Код программы lab6-1.asm

A terminal window with a dark background. The title bar shows the user 'ramanyyew@fedora' and the directory '~/work/arch-pc/lab06'. The terminal contains the following commands and output:

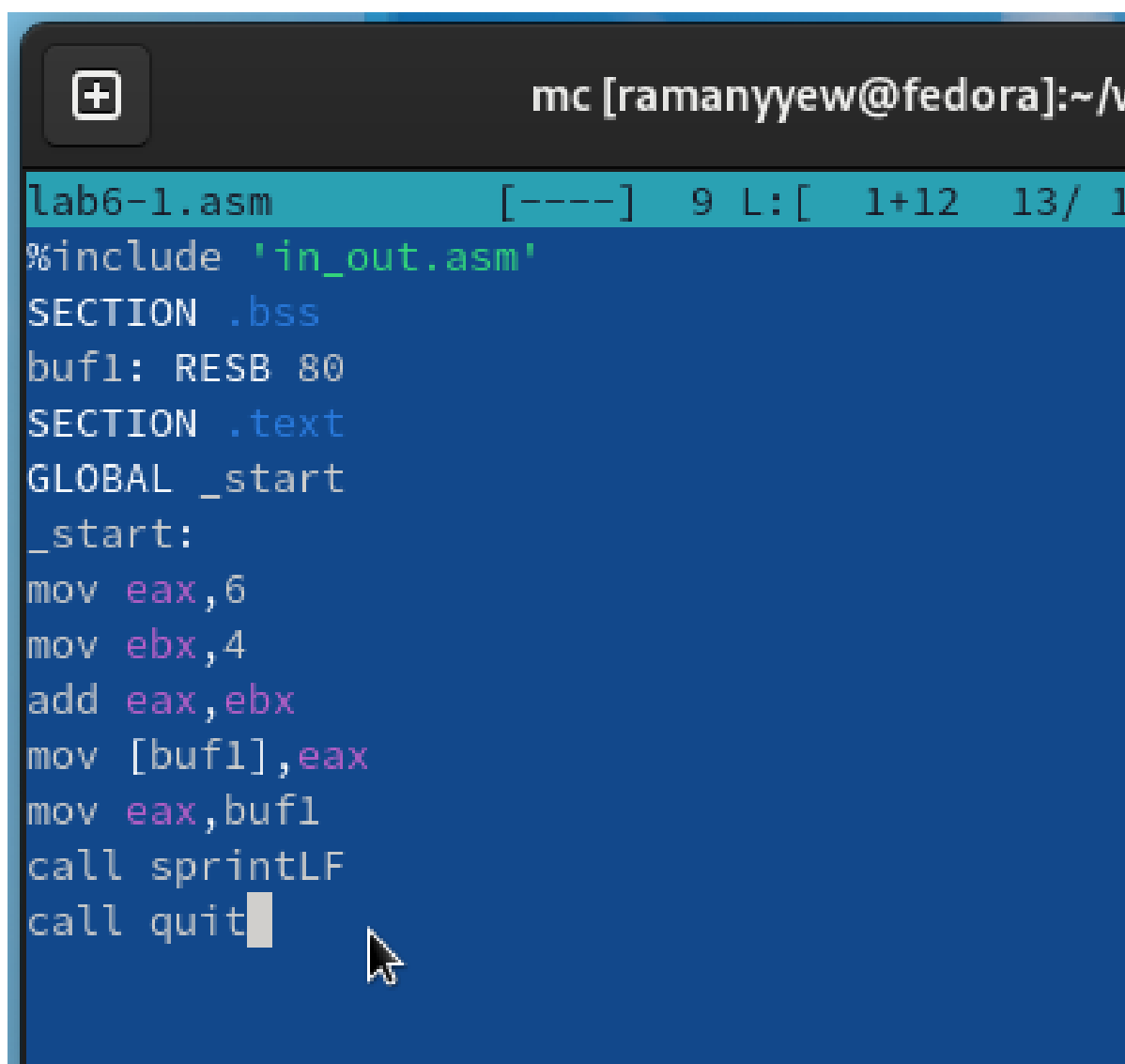
```
[ramanyyew@fedora lab06]$ nasm -f elf lab6-1.asm
[ramanyyew@fedora lab06]$ ld -m elf_i386 lab6-1.o -o lab6-1
[ramanyyew@fedora lab06]$ ./lab6-1
j
[ramanyyew@fedora lab06]$
```

Рис. 2.2: Компиляция и 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 `б` 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`.

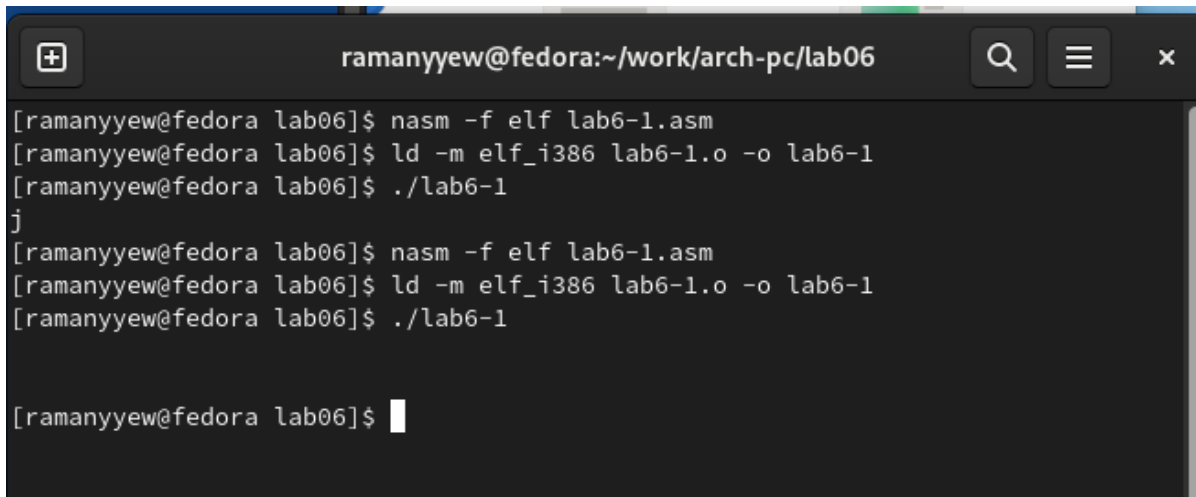
Далее изменяю текст программы и вместо символов, запишем в регистры числа. (рис. [2.3])





```
lab6-1.asm [-----] 9 L: [ 1+12 13/ 1
#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. 2.3: Код программы lab6-1.asm



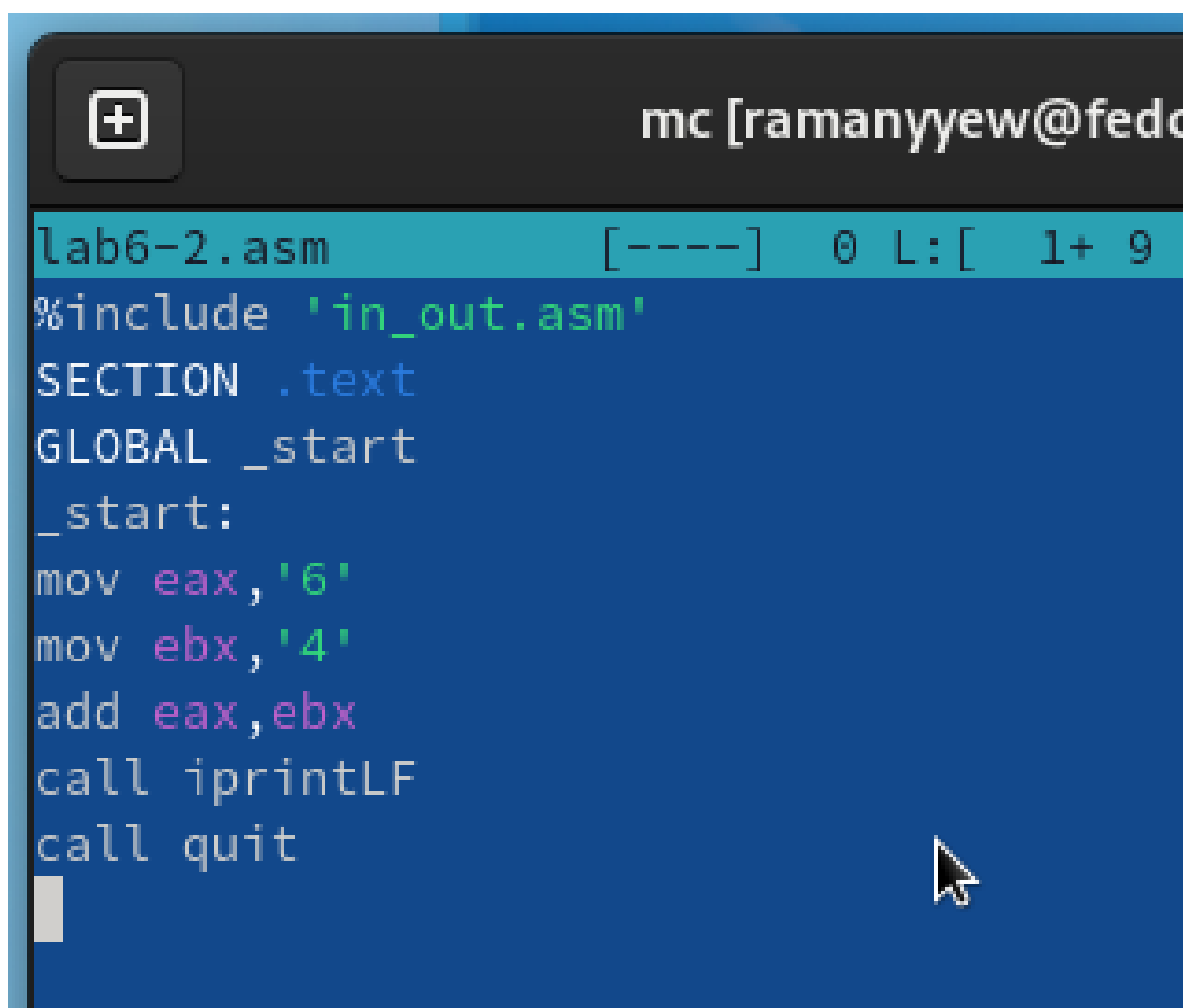
```
ramanyyew@fedora:~/work/arch-pc/lab06
[ramanyyew@fedora lab06]$ nasm -f elf lab6-1.asm
[ramanyyew@fedora lab06]$ ld -m elf_i386 lab6-1.o -o lab6-1
[ramanyyew@fedora lab06]$ ./lab6-1
j
[ramanyyew@fedora lab06]$ nasm -f elf lab6-1.asm
[ramanyyew@fedora lab06]$ ld -m elf_i386 lab6-1.o -o lab6-1
[ramanyyew@fedora lab06]$ ./lab6-1

[ramanyyew@fedora lab06]$
```

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

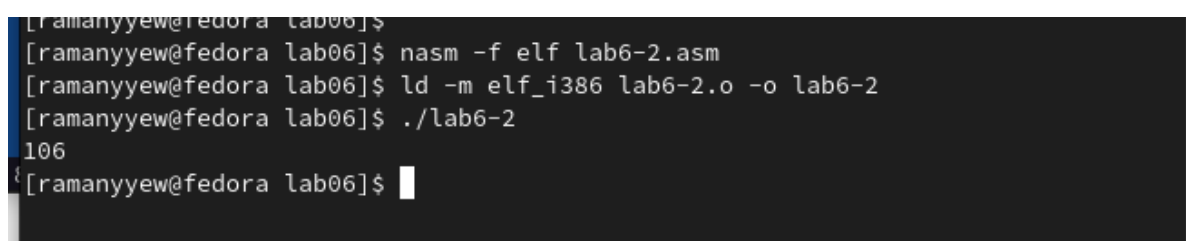
Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10 (рис. [2.4]). Это символ конца строки (возврат каретки). В консоле он не отображается, но добавляет пустую строку.

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



```
lab6-2.asm [-----] 0 L: [ 1+ 9
#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: Код программы lab6-2.asm



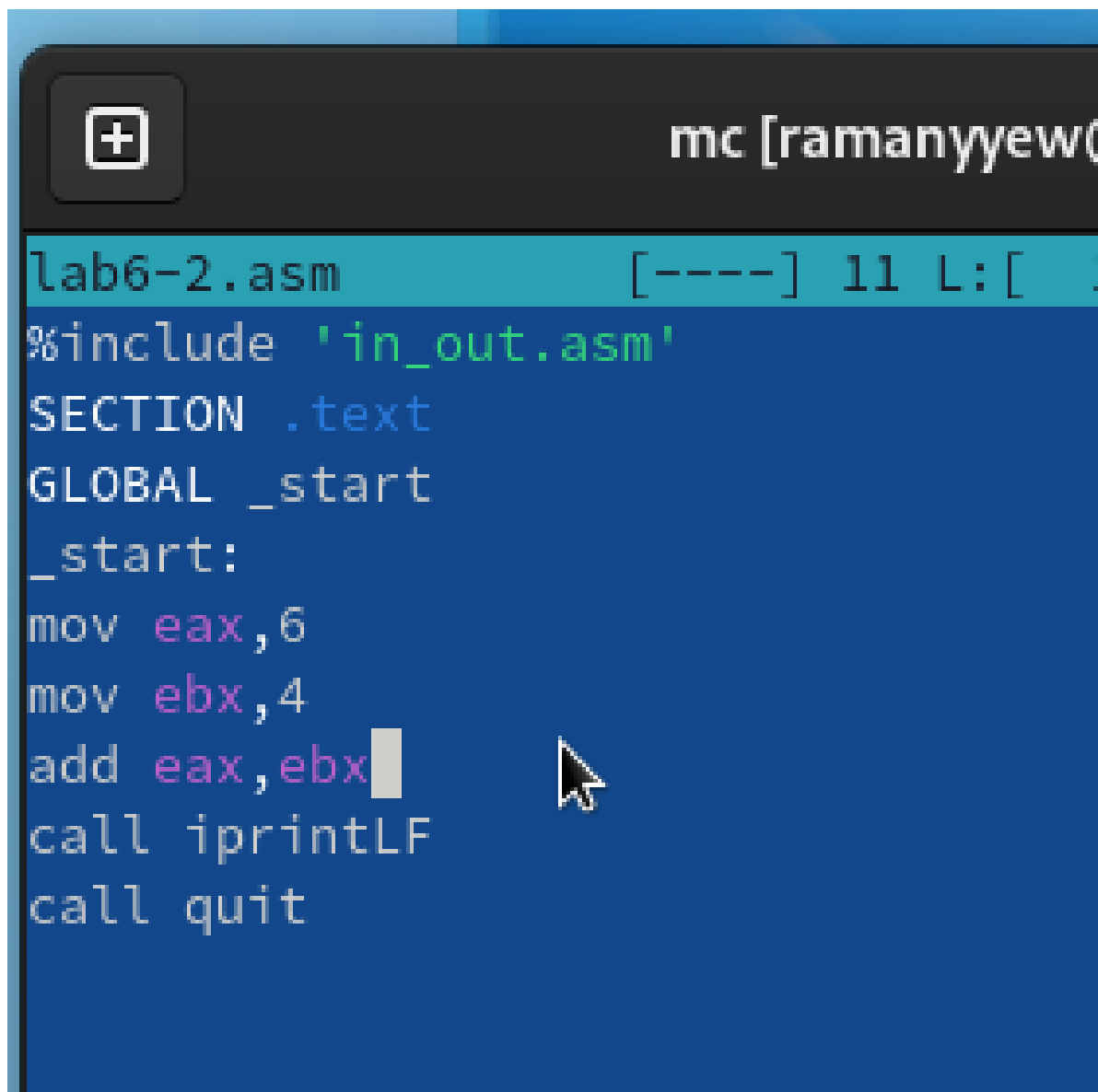
```
[ramanyyew@fedora lab06]$
[ramanyyew@fedora lab06]$ nasm -f elf lab6-2.asm
[ramanyyew@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[ramanyyew@fedora lab06]$ ./lab6-2
106
[ramanyyew@fedora lab06]$
```

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

В результате работы программы мы получим число 106.(рис. [2.6]) 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])



```
mc [ramanyyew@
lab6-2.asm      [-----] 11 L: [
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov  eax,6
mov  ebx,4
add  eax,ebx
call iprintLF
call quit
```

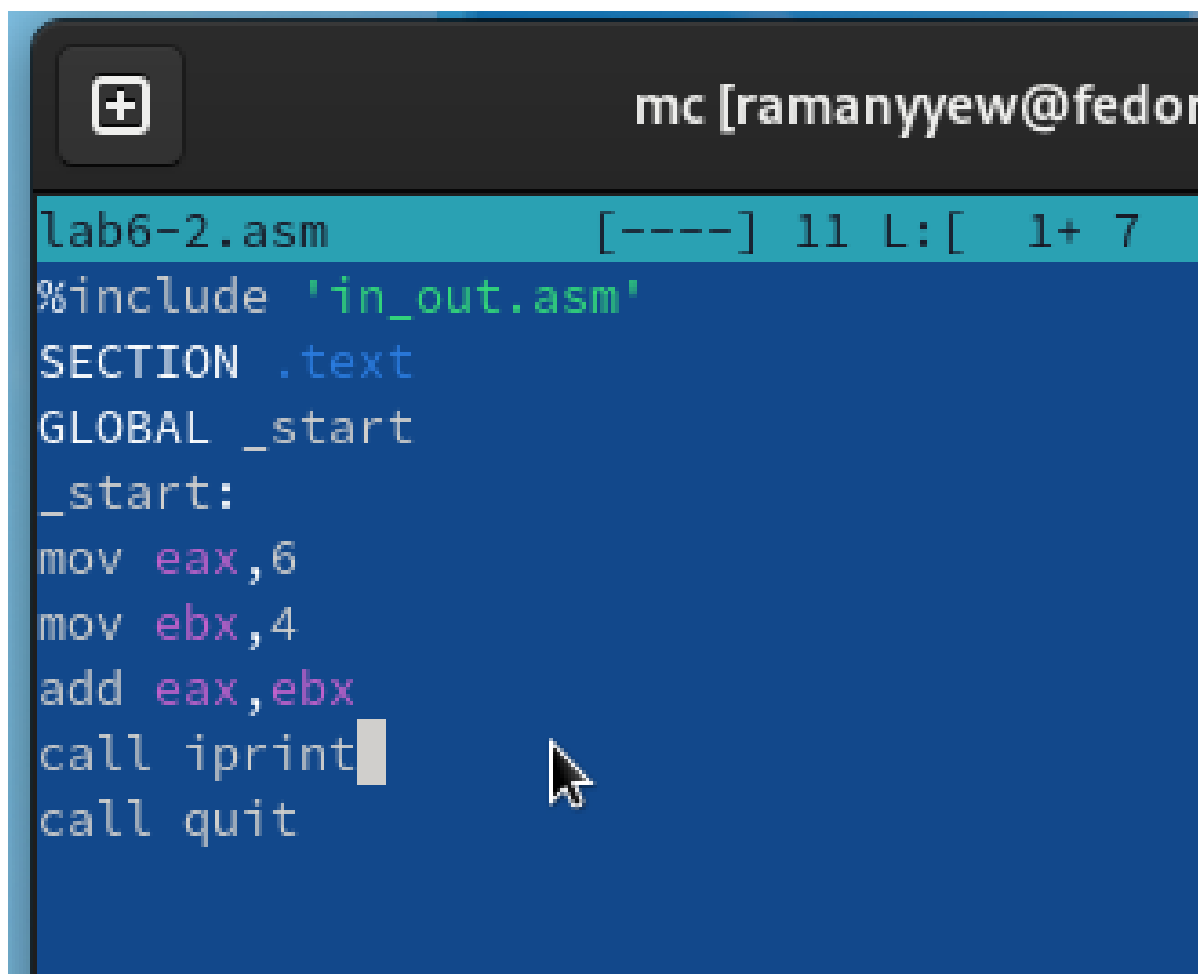
Рис. 2.7: Код программы lab6-2.asm

Функция `iprintLF` позволяет вывести число и операндами были числа (а не коды символов). Поэтому получаем число 10. (рис. [2.8])

```
[ramanyyew@fedora lab06]$ nasm -f elf lab6-2.asm
[ramanyyew@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[ramanyyew@fedora lab06]$ ./lab6-2
106
[ramanyyew@fedora lab06]$ nasm -f elf lab6-2.asm
[ramanyyew@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[ramanyyew@fedora lab06]$ ./lab6-2
10
[ramanyyew@fedora lab06]$
```

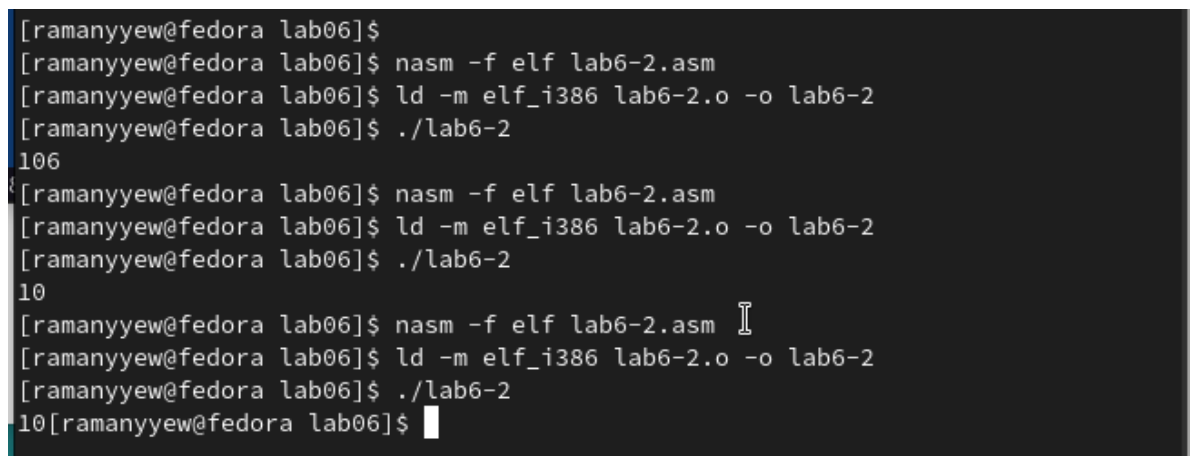
Рис. 2.8: Компиляция и запуск программы lab6-2.asm

Заменял функцию `iprintLF` на `iprint`. Создал исполняемый файл и запустил его. Вывод отличается тем, что нет переноса строки. (рис. [2.9])



```
lab6-2.asm [----] 11 L: [ 1+ 7
#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: Код программы lab6-2.asm



```
[ramanyyew@fedora lab06]$
[ramanyyew@fedora lab06]$ nasm -f elf lab6-2.asm
[ramanyyew@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[ramanyyew@fedora lab06]$ ./lab6-2
106
[ramanyyew@fedora lab06]$ nasm -f elf lab6-2.asm
[ramanyyew@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[ramanyyew@fedora lab06]$ ./lab6-2
10
[ramanyyew@fedora lab06]$ nasm -f elf lab6-2.asm
[ramanyyew@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[ramanyyew@fedora lab06]$ ./lab6-2
10[ramanyyew@fedora lab06]$
```

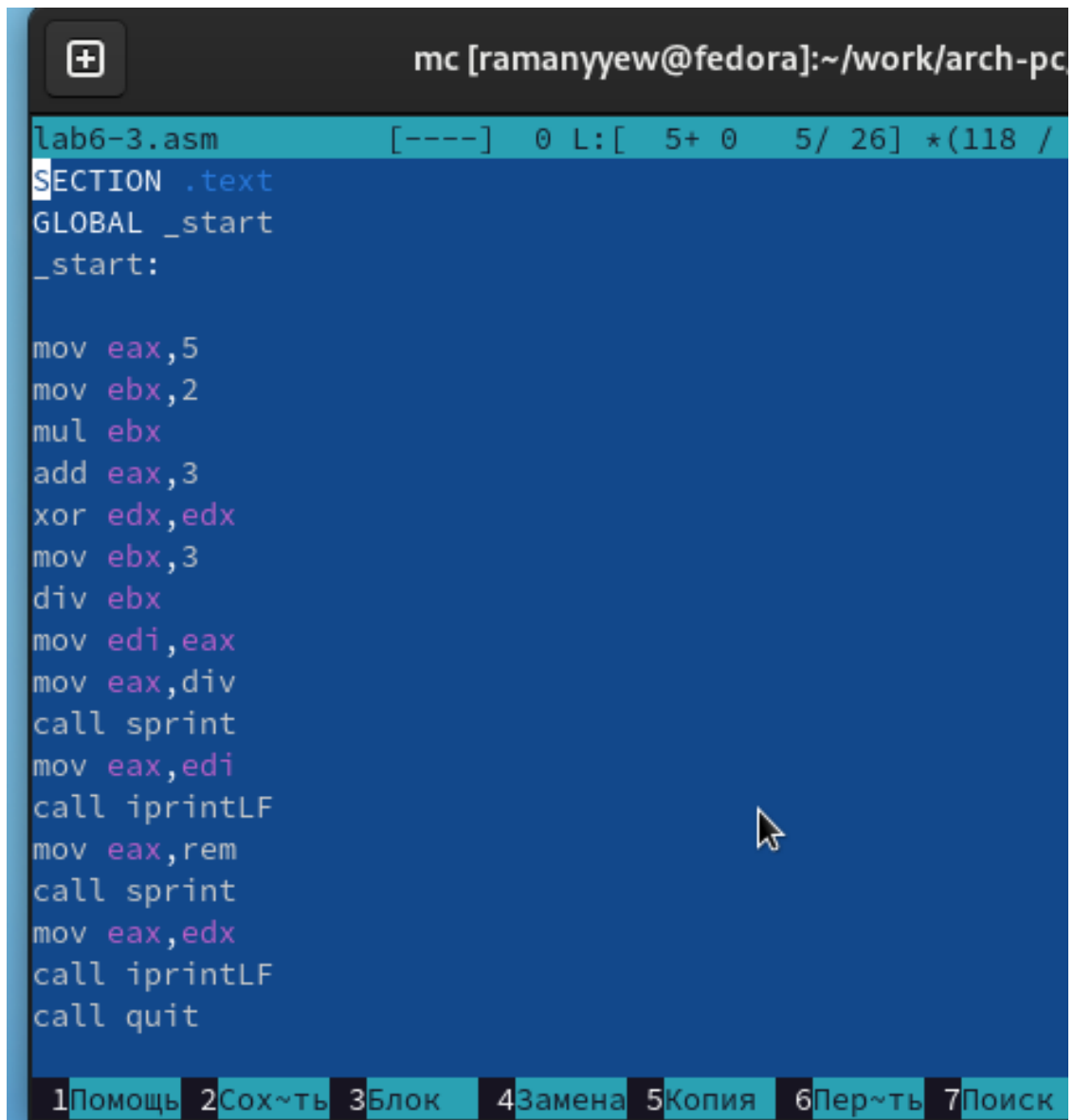
Рис. 2.10: Компиляция и запуск программы lab6-2.asm

## 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. [2.11]) (рис. [2.12])



```
mc [ramanyyew@fedora]:~/work/arch-pc
lab6-3.asm [----] 0 L: [ 5+ 0 5/ 26] *(118 /
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
```

1Помощь 2Сох~ть 3Блок 4Замена 5Копия 6Пер~ть 7Поиск

Рис. 2.11: Код программы lab6-3.asm



```
[ramanyyew@fedora lab06]$
[ramanyyew@fedora lab06]$ nasm -f elf lab6-3.asm
[ramanyyew@fedora lab06]$ ld -m elf_i386 lab6-3.o -o lab6-3
[ramanyyew@fedora lab06]$ ./lab6-3
Результат: 4
Остаток от деления: 1
[ramanyyew@fedora lab06]$
```

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



Изменил текст программы для вычисления выражения

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

. Создал исполняемый файл и проверил его работу. (рис. [2.13]) (рис. [2.14])



```
mc [ramanyyew@fedora]
lab6-3.asm [----] 9 L: [ 5+ 9
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

1Помощь 2Сохранить 3Блок 4Замена 5Копия
```

Рис. 2.13: Код программы lab6-3.asm

```
[ramanyyew@fedora lab06]$  
[ramanyyew@fedora lab06]$ nasm -f elf lab6-3.asm  
[ramanyyew@fedora lab06]$ ld -m elf_i386 lab6-3.o -o lab6-3  
[ramanyyew@fedora lab06]$ ./lab6-3  
Результат: 4  
Остаток от деления: 1  
[ramanyyew@fedora lab06]$ nasm -f elf lab6-3.asm  
[ramanyyew@fedora lab06]$ ld -m elf_i386 lab6-3.o -o lab6-3  
[ramanyyew@fedora lab06]$ ./lab6-3  
Результат: 5  
Остаток от деления: 1  
[ramanyyew@fedora lab06]$
```

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

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`. (рис. [2.15]) (рис. [2.16])



```
mc [ramanyyew@fedora]:~/...  
variant.asm [----] 7 L: [ 5+15 20/ :  
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
```

Рис. 2.15: Код программы variant.asm

```
[ramanyyew@fedora lab06]$  
[ramanyyew@fedora lab06]$ nasm -f elf variant.asm  
[ramanyyew@fedora lab06]$ ld -m elf_i386 variant.o -o variant  
[ramanyyew@fedora lab06]$ ./variant  
Введите № студенческого билета:  
1032234170  
Ваш вариант: 11  
[ramanyyew@fedora lab06]$  
[ramanyyew@fedora 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:’?

Переменная с фразой “Ваш вариант:” перекладывается в регистр `eax` с помощью строки `mov eax, rem`.

Для вызова подпрограммы вывода строки используется строка `call sprint`.

2. Для чего используются следующие инструкции?

- `mov ecx, x` - перекладывает регистр `ecx` в переменную
- `mov edx, 80` - устанавливает значение 80 в регистр `edx`.
- `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?

- `xor edx, edx` - обнуляет регистр `edx`.
- `mov ebx, 20` - устанавливает значение 20 в регистр `ebx`.

- `div ebx` - выполняет 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` 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.

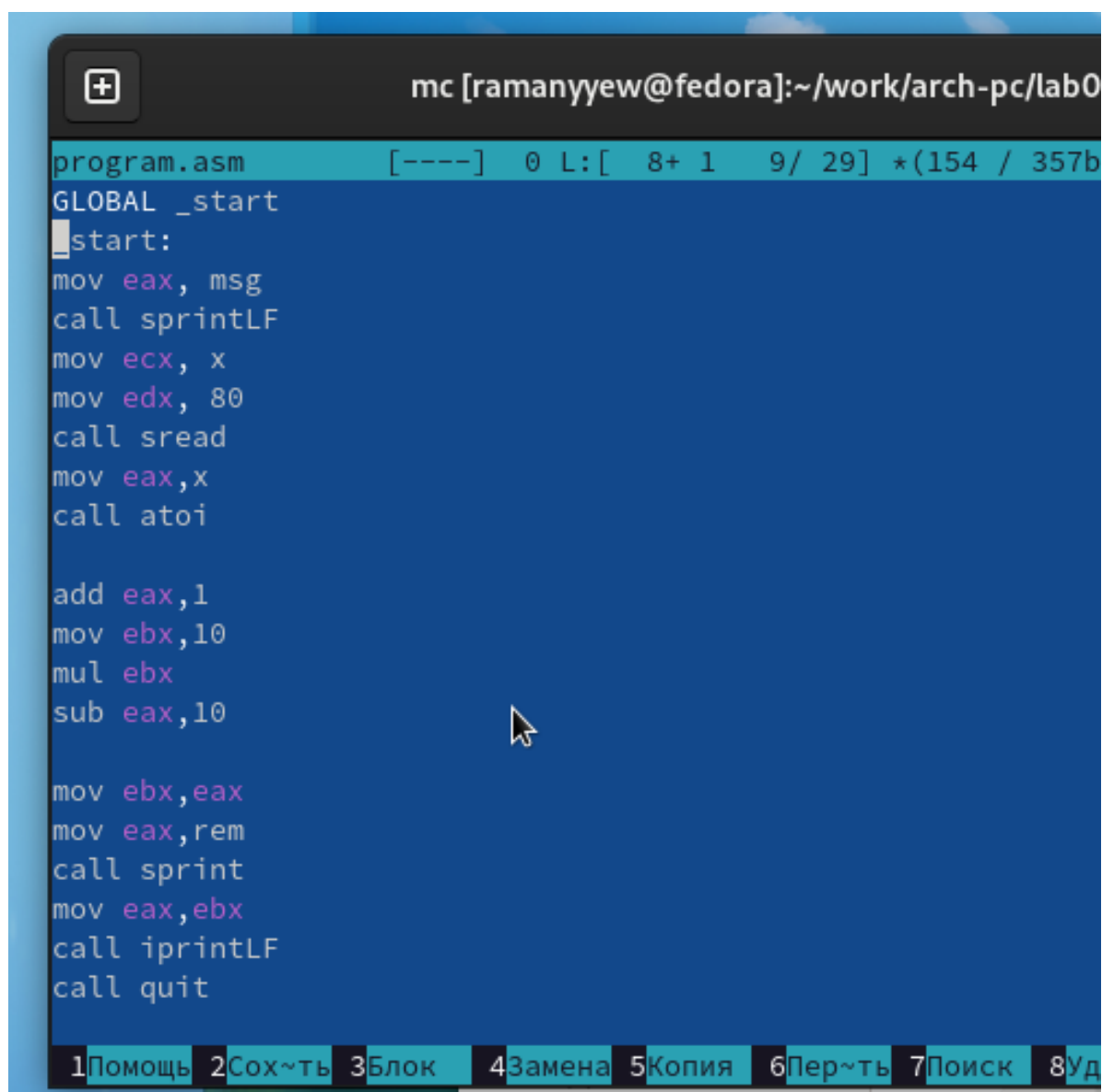
7. Какие строки листинга отвечают за вывод на экран результата вычислений?

- Результат вычислений перекладывается в регистр `eax` с помощью строки `mov eax, edx`.
- Для вызова подпрограммы вывода используется строка `call iprintLF`.

## 2.3 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. (рис. [2.17]) (рис. [2.18]) Создайте исполняемый файл и проверьте его работу для значений  $x_1$  и  $x_2$  из 6.3.

Вариант 11 -  $10(x + 1) - 10$  для  $x=1, x=7$



```
mc [ramanyyew@fedora]:~/work/arch-pc/lab0
program.asm [----] 0 L:[ 8+ 1 9/ 29] *(154 / 357b
GLOBAL _start
_start:
mov eax, msg
call sprintLF
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi

add eax, 1
mov ebx, 10
mul ebx
sub eax, 10

mov ebx, eax
mov eax, rem
call sprint
mov eax, ebx
call iprintLF
call quit
```

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить

Рис. 2.17: Код программы program.asm

при  $x=1$   $f(x) = 10$

при  $x=7$   $f(x) = 70$

```
[ramanyyew@fedora lab06]$  
[ramanyyew@fedora lab06]$ nasm -f elf program.asm  
[ramanyyew@fedora lab06]$ ld -m elf_i386 program.o -o program  
[ramanyyew@fedora lab06]$ ./program  
Введите X  
1  
выражение = : 10  
[ramanyyew@fedora lab06]$ ./program  
Введите X  
7  
выражение = : 70  
[ramanyyew@fedora lab06]$
```

Рис. 2.18: Компиляция и запуск программы program.asm

Программа считает верно.

## **3 Выводы**

Изучили работу с арифметическими операциями.