

# **Отчёта по лабораторной работе 6**

**Архитектура компьютеров и операционные системы**

Плетьяго Кирилл НММбд-03-23

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Ответы на вопросы по программе variant.asm . . . . .	18
2.2	Самостоятельное задание . . . . .	19
<b>3</b>	<b>Выводы</b>	<b>22</b>

## Список иллюстраций

2.1	Подготовил каталог . . . . .	6
2.2	Программа в файле lab6-1.asm . . . . .	7
2.3	Запуск программы lab6-1.asm . . . . .	8
2.4	Программа в файле lab6-1.asm . . . . .	9
2.5	Запуск программы lab6-1.asm . . . . .	10
2.6	Программа в файле lab6-2.asm . . . . .	11
2.7	Запуск программы lab6-2.asm . . . . .	11
2.8	Программа в файле lab6-2.asm . . . . .	12
2.9	Запуск программы lab6-2.asm . . . . .	12
2.10	Запуск программы lab6-2.asm . . . . .	13
2.11	Программа в файле lab6-3.asm . . . . .	14
2.12	Запуск программы lab6-3.asm . . . . .	15
2.13	Программа в файле lab6-3.asm . . . . .	15
2.14	Запуск программы lab6-3.asm . . . . .	16
2.15	Программа в файле variant.asm . . . . .	17
2.16	Запуск программы variant.asm . . . . .	18
2.17	Программа в файле work.asm . . . . .	20
2.18	Запуск программы task.asm . . . . .	21

## **Список таблиц**

# 1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

## 2 Выполнение лабораторной работы

Создал каталог для программ лабораторной работы № 6, перешел в него и создал файл lab6-1.asm. (рис. [2.1])

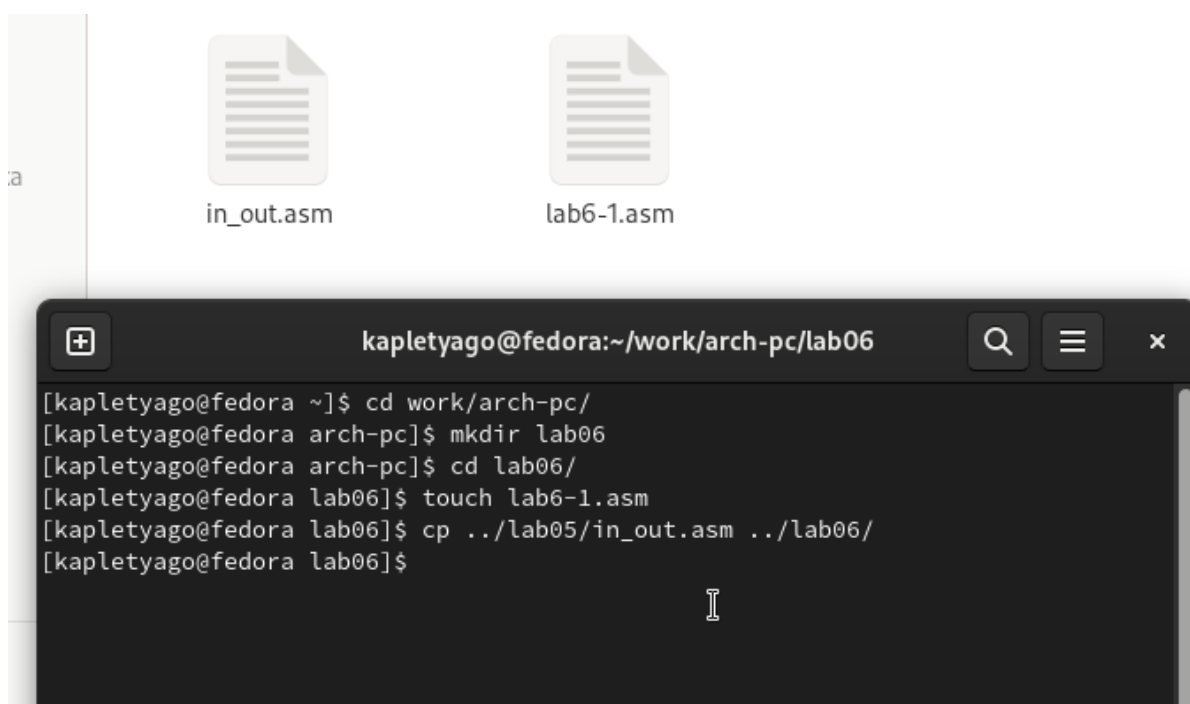
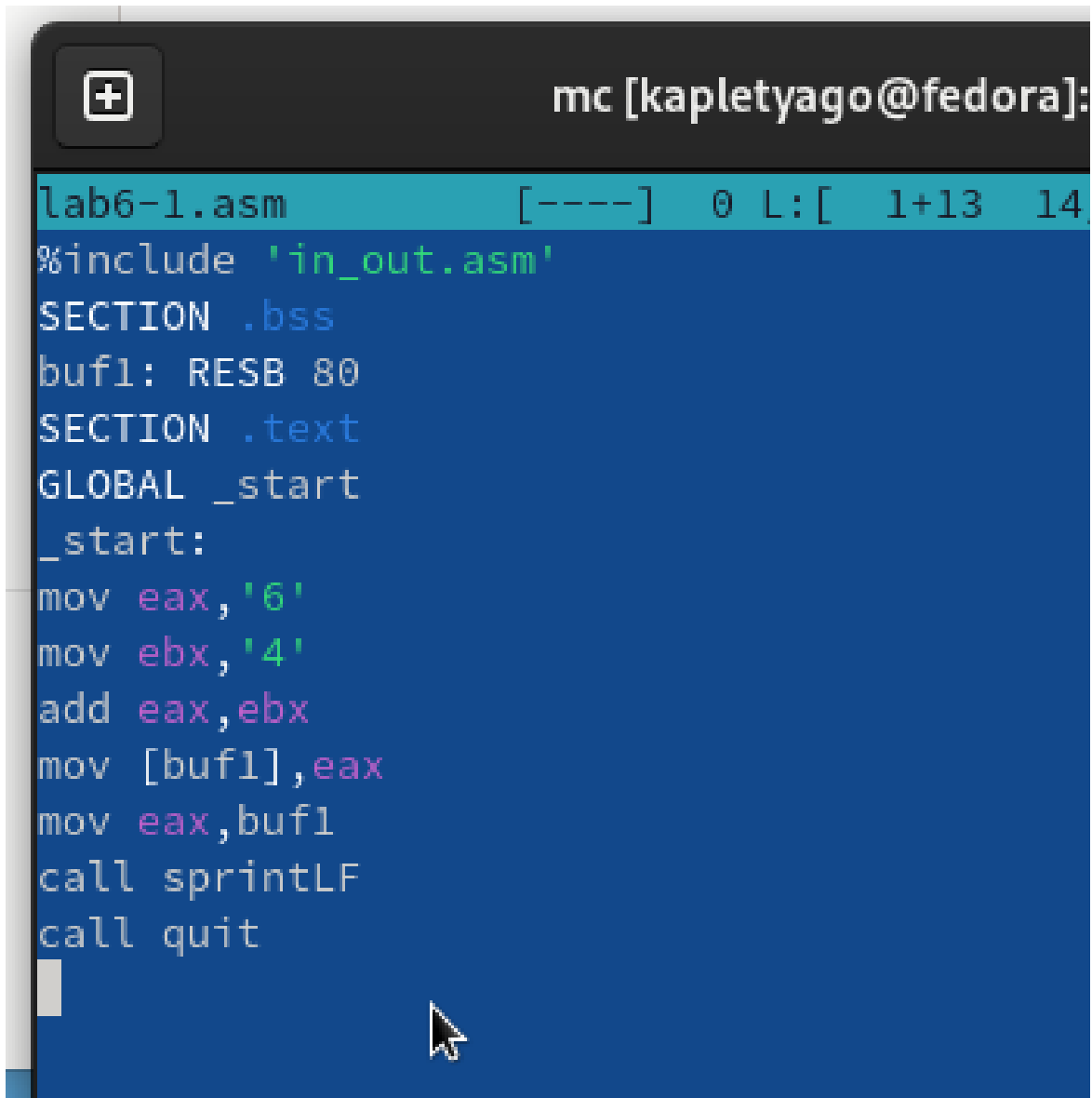


Рис. 2.1: Подготовил каталог

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

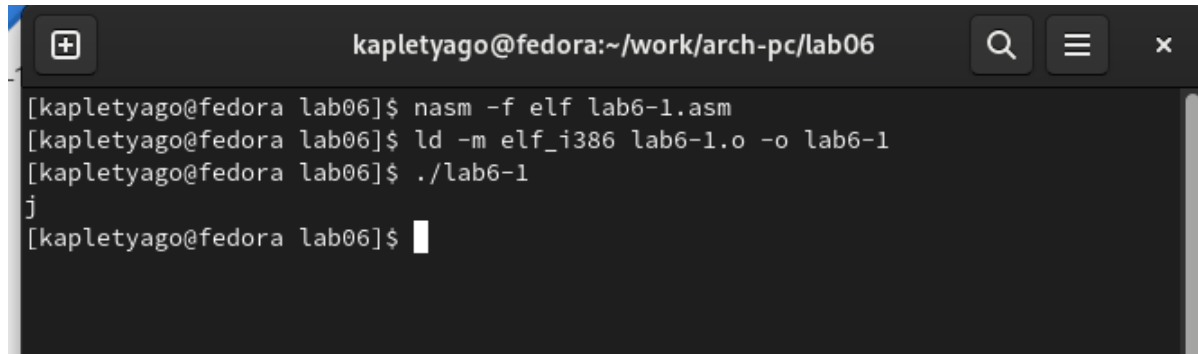


```
mc [kapletyago@fedora]:  
lab6-1.asm [----] 0 L:[ 1+13 14  
%include 'in_out.asm'  
SECTION .bss  
buf1: RESB 80  
SECTION .text  
GLOBAL _start  
_start:  
mov eax, '6'  
mov ebx, '4'  
add eax, ebx  
mov [buf1], eax  
mov eax, buf1  
call sprintLF  
call quit
```

Рис. 2.2: Программа в файле lab6-1.asm

В данной программе (рис. [2.2]) мы записываем символ '6' в регистр `eax` (`mov eax, '6'`), а символ '4' в регистр `ebx` (`mov ebx, '4'`). Затем мы добавляем значение регистра `ebx` к значению в регистре `eax` (`add eax, ebx`, результат сложения записывается в регистр `eax`). После этого мы выводим результат. Однако, для использования функции `sprintLF`, необходимо, чтобы в регистре `eax` был записан адрес, поэтому мы используем дополнительную переменную. Мы записываем

значение регистра `eax` в переменную `buf1` (`mov [buf1], eax`), а затем записываем адрес переменной `buf1` в регистр `eax` (`mov eax, buf1`) и вызываем функцию `sprintLF`.



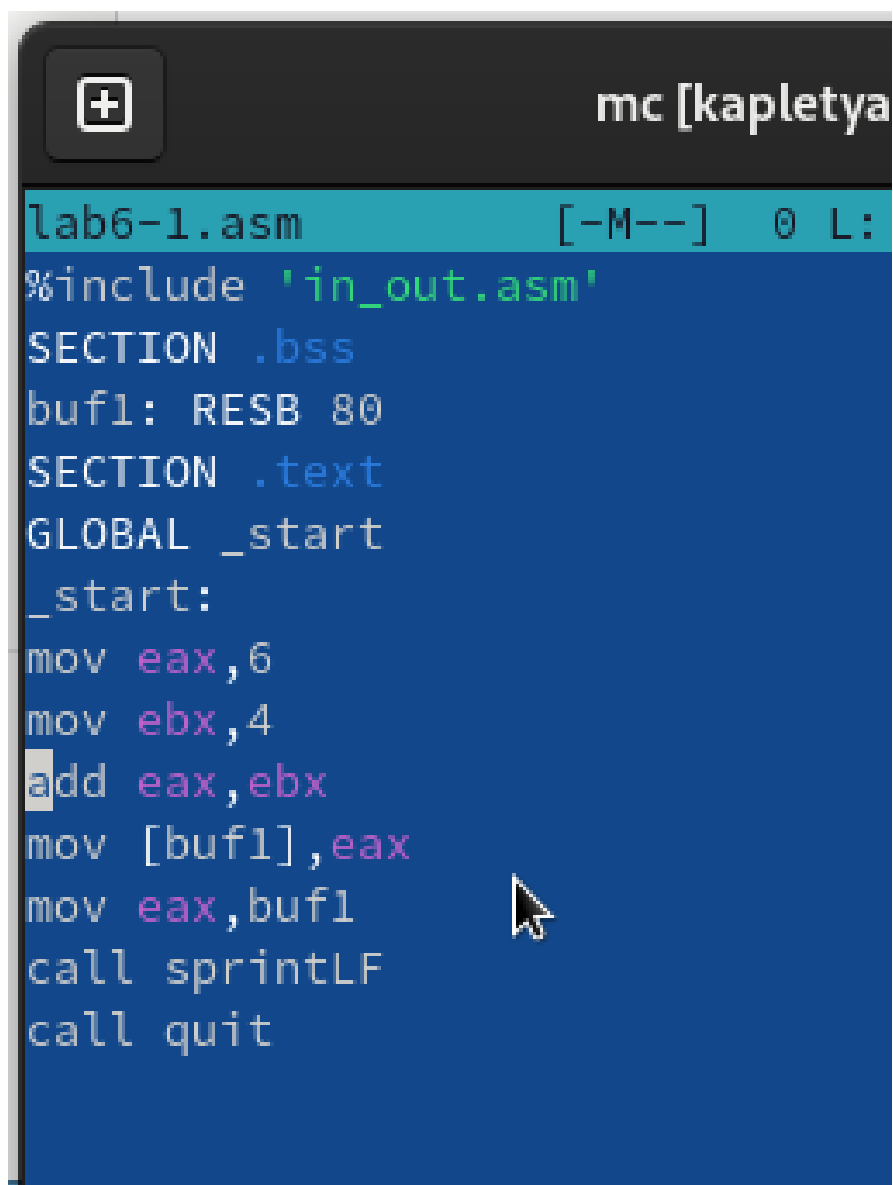
```
kapletyago@fedora:~/work/arch-pc/lab06
[kapletyago@fedora lab06]$ nasm -f elf lab6-1.asm
[kapletyago@fedora lab06]$ ld -m elf_i386 lab6-1.o -o lab6-1
[kapletyago@fedora lab06]$ ./lab6-1
j
[kapletyago@fedora lab06]$
```

Рис. 2.3: Запуск программы `lab6-1.asm`

В данном случае, когда мы ожидаем увидеть число 10 при выводе значения регистра `eax`, фактическим результатом будет символ `'j'`. Это происходит из-за того, что код символа `'б'` равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа `'4'` равен 00110100 (или 52 в десятичном представлении). Когда мы выполняем команду `add eax, ebx`, результатом будет сумма кодов - 01101010 (или 106 в десятичном представлении), который соответствует символу `'j'`. (рис. [2.3])

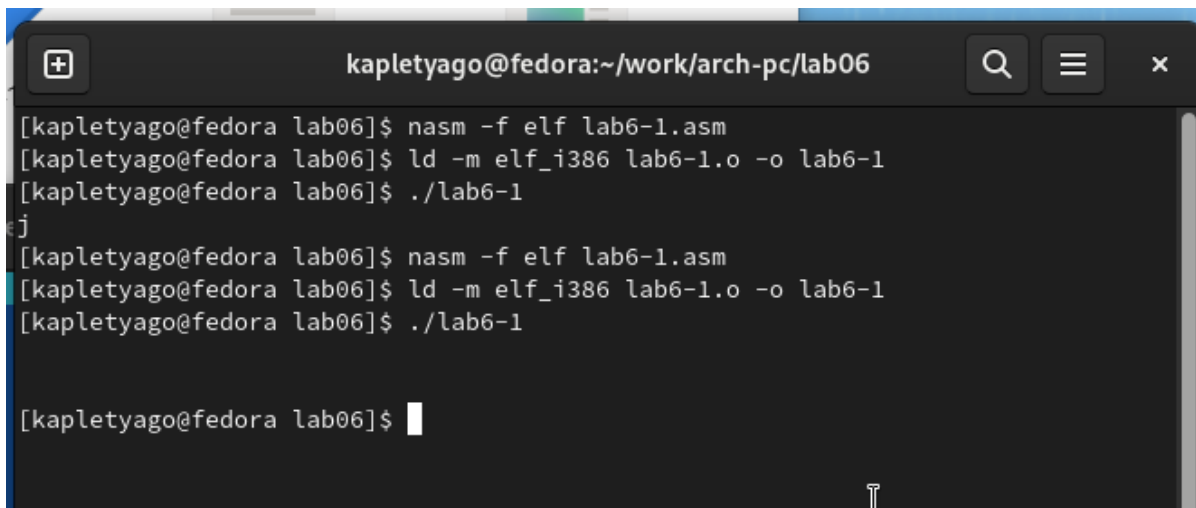
Далее изменяю текст программы и вместо символов, запишем в регистры числа. (рис. [2.4])





```
lab6-1.asm [-M--] 0 L:
#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. 2.4: Программа в файле lab6-1.asm

A terminal window titled 'kapletyago@fedora:~/work/arch-pc/lab06' with search, menu, and close icons. The terminal shows the following commands and their outputs:

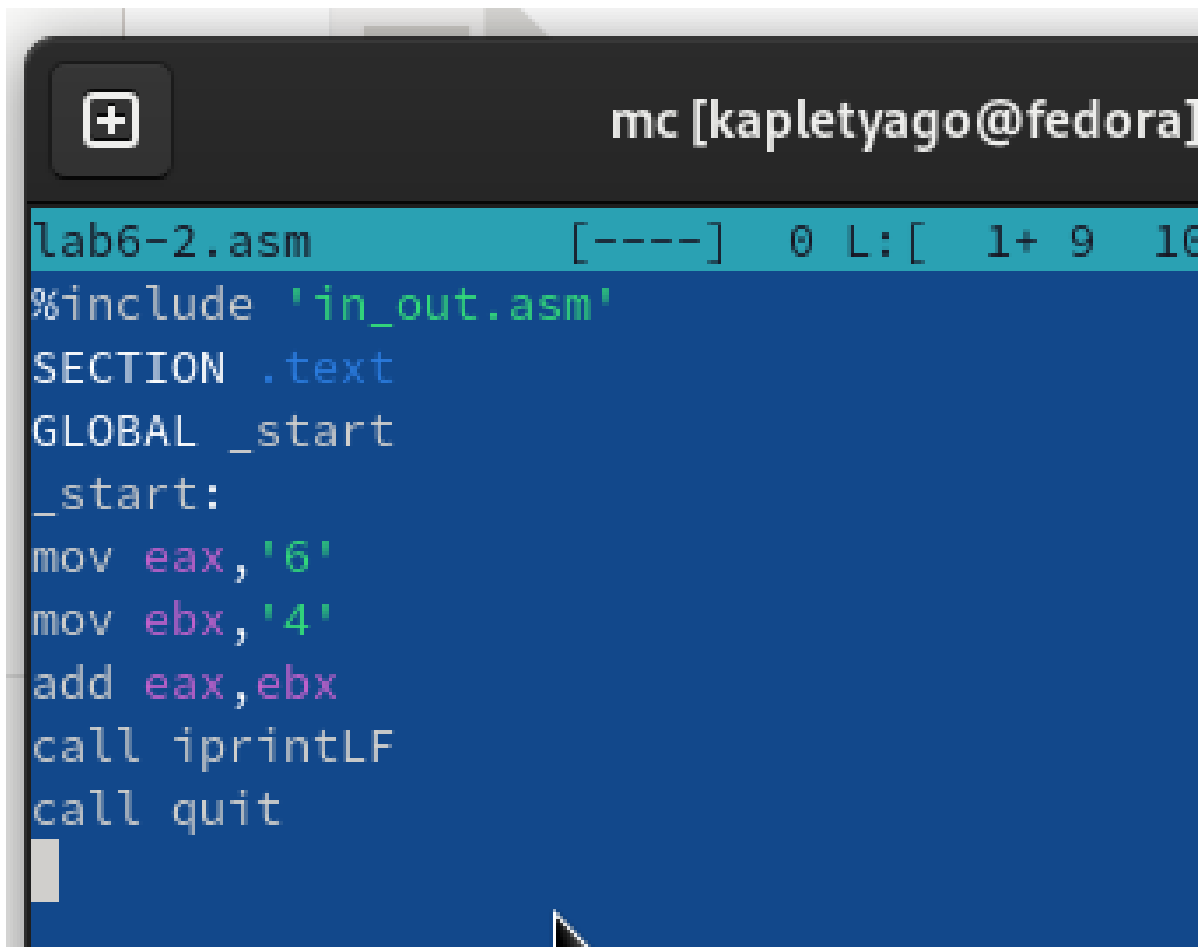
```
[kapletyago@fedora lab06]$ nasm -f elf lab6-1.asm
[kapletyago@fedora lab06]$ ld -m elf_i386 lab6-1.o -o lab6-1
[kapletyago@fedora lab06]$ ./lab6-1
j
[kapletyago@fedora lab06]$ nasm -f elf lab6-1.asm
[kapletyago@fedora lab06]$ ld -m elf_i386 lab6-1.o -o lab6-1
[kapletyago@fedora lab06]$ ./lab6-1

[kapletyago@fedora lab06]$
```

Рис. 2.5: Запуск программы lab6-1.asm

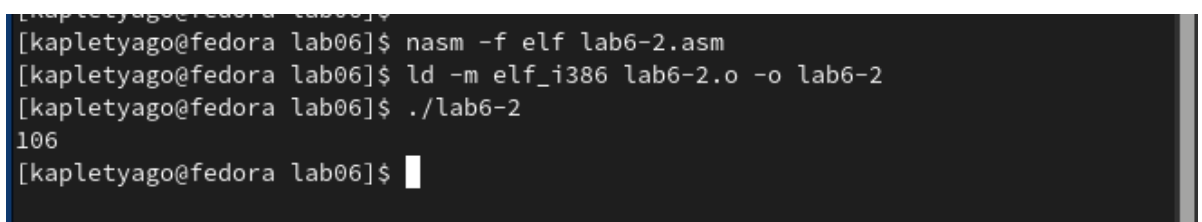
Как и в предыдущем случае, при выполнении программы мы не получим число 10. Вместо этого выводится символ с кодом 10, который представляет собой символ конца строки (возврат каретки). (рис. [2.5]) Этот символ не отображается в консоли, но он добавляет пустую строку.

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. [2.6])



```
mc [kapletyago@fedora]
lab6-2.asm [-----] 0 L: [ 1+ 9 10
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

Рис. 2.6: Программа в файле lab6-2.asm

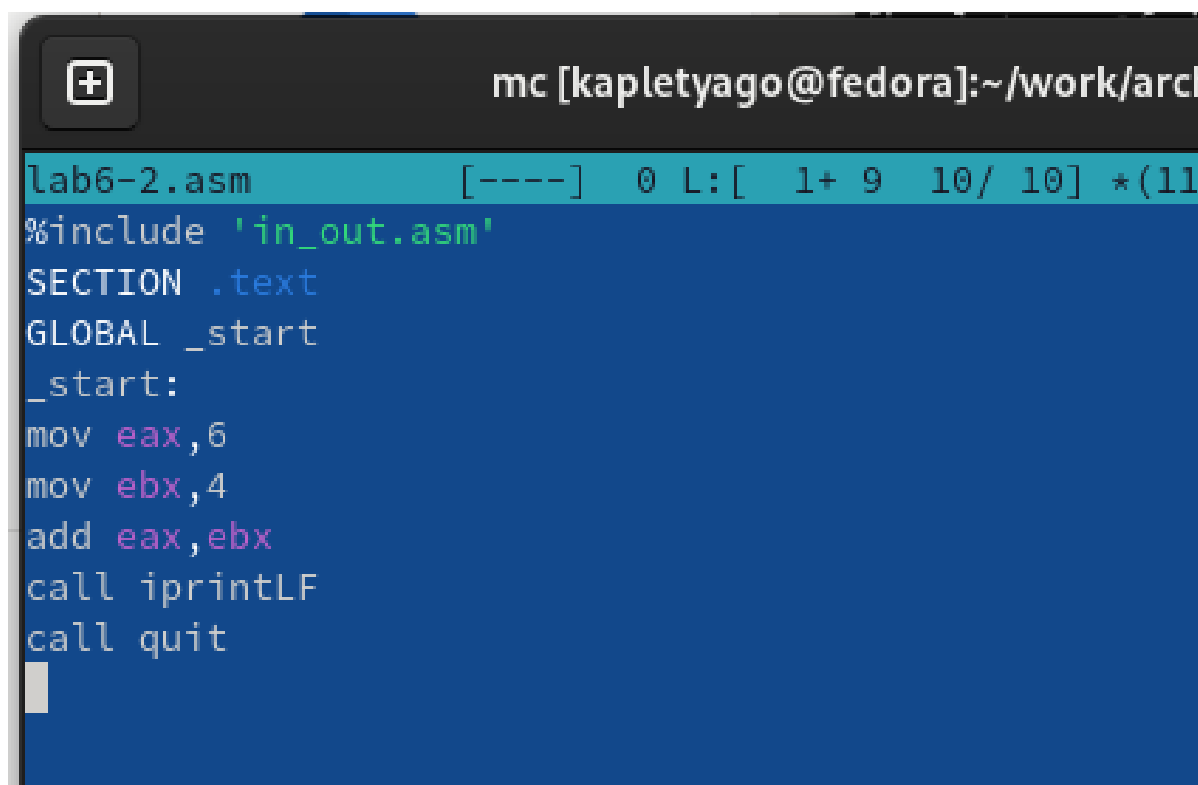


```
[kapletyago@fedora lab06]$ nasm -f elf lab6-2.asm
[kapletyago@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[kapletyago@fedora lab06]$ ./lab6-2
106
[kapletyago@fedora lab06]$
```

Рис. 2.7: Запуск программы lab6-2.asm

В результате выполнения программы мы получим число 106. (рис. [2.7]) В данном случае, как и в первом случае, команда add складывает коды 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.

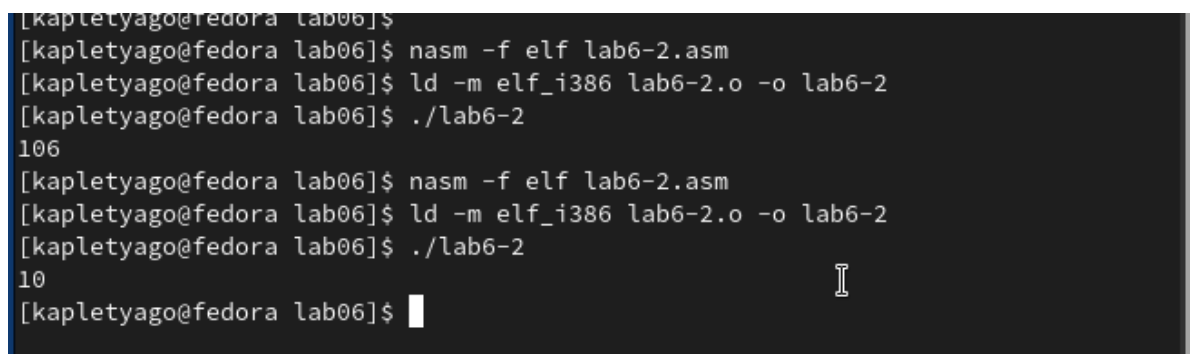
Аналогично предыдущему примеру изменим символы на числа.(рис. [2.8])



```
mc [kapletyago@fedora]:~/work/arc  
lab6-2.asm [----] 0 L:[ 1+ 9 10/ 10] *(11  
%include 'in_out.asm'  
SECTION .text  
GLOBAL _start  
_start:  
mov eax,6  
mov ebx,4  
add eax,ebx  
call iprintLF  
call quit
```

Рис. 2.8: Программа в файле lab6-2.asm

Функция iprintLF позволяет вывести число и операндами были числа (а не коды символов). Поэтому получаем число 10.(рис. [2.9])



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

Рис. 2.9: Запуск программы lab6-2.asm

Заменяю функцию iprintLF на iprint. Создал исполняемый файл и запустил его.

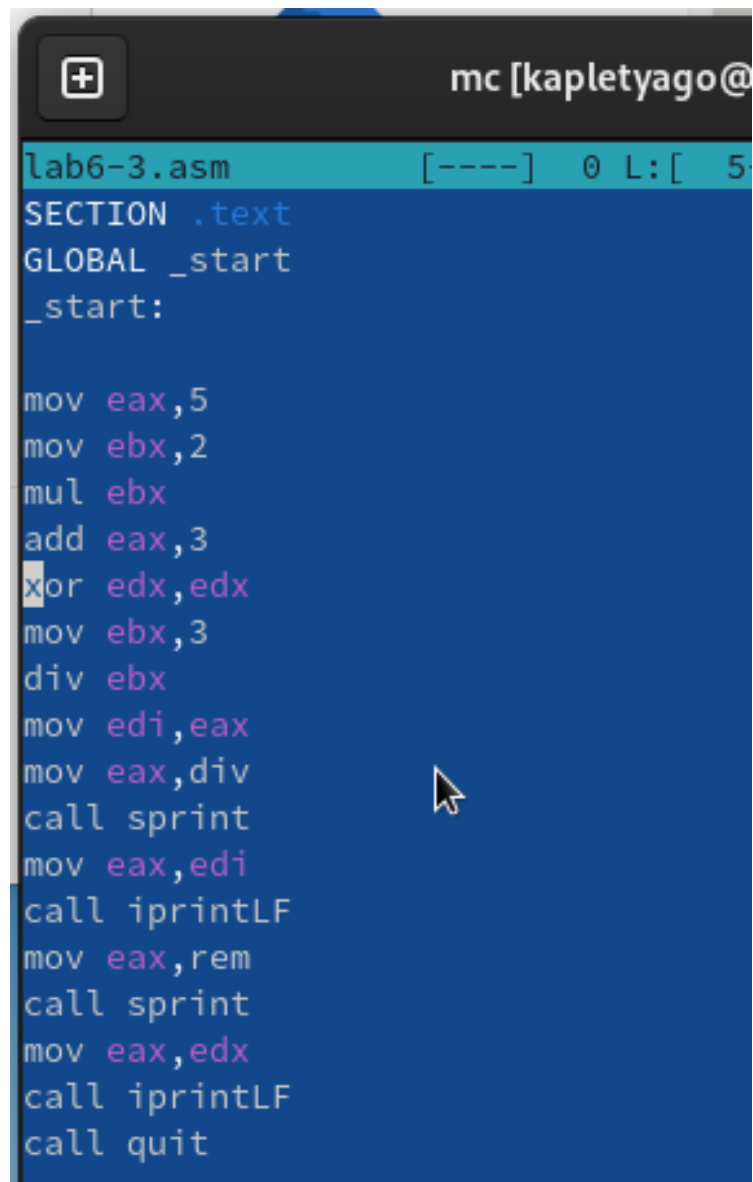
Вывод отличается тем, что нет переноса строки.(рис. [2.10])

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

Рис. 2.10: Запуск программы lab6-2.asm

В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения (рис. [2.11]) (рис. [2.12])

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



```
mc [kapletyago@
lab6-3.asm [----] 0 L: [ 5
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

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

Рис. 2.12: Запуск программы lab6-3.asm

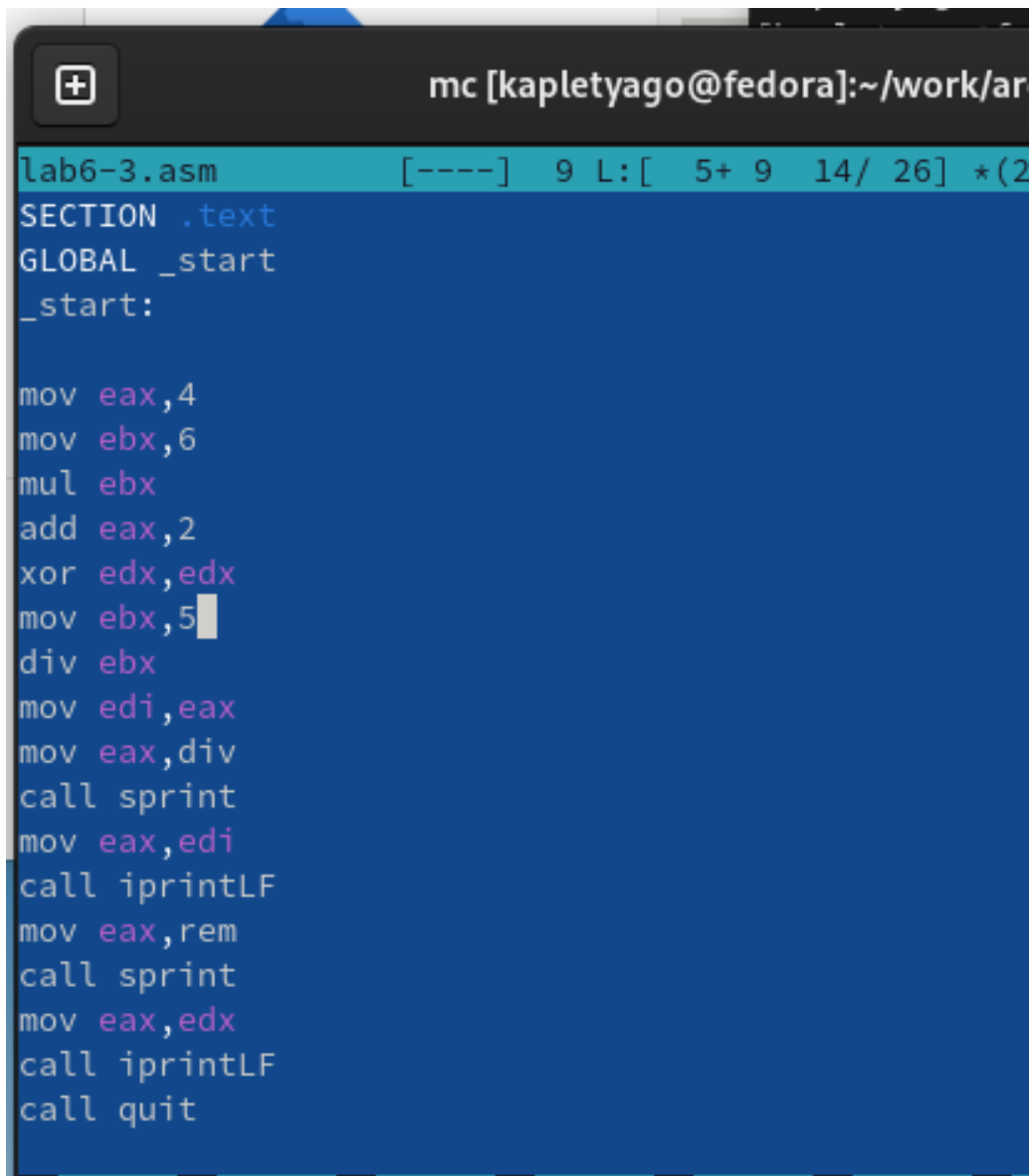
Изменил текст программы для вычисления выражения

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

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

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

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



```
mc [kapletyago@fedora]:~/work/ar
lab6-3.asm [----] 9 L:[ 5+ 9 14/ 26] *(2
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
```

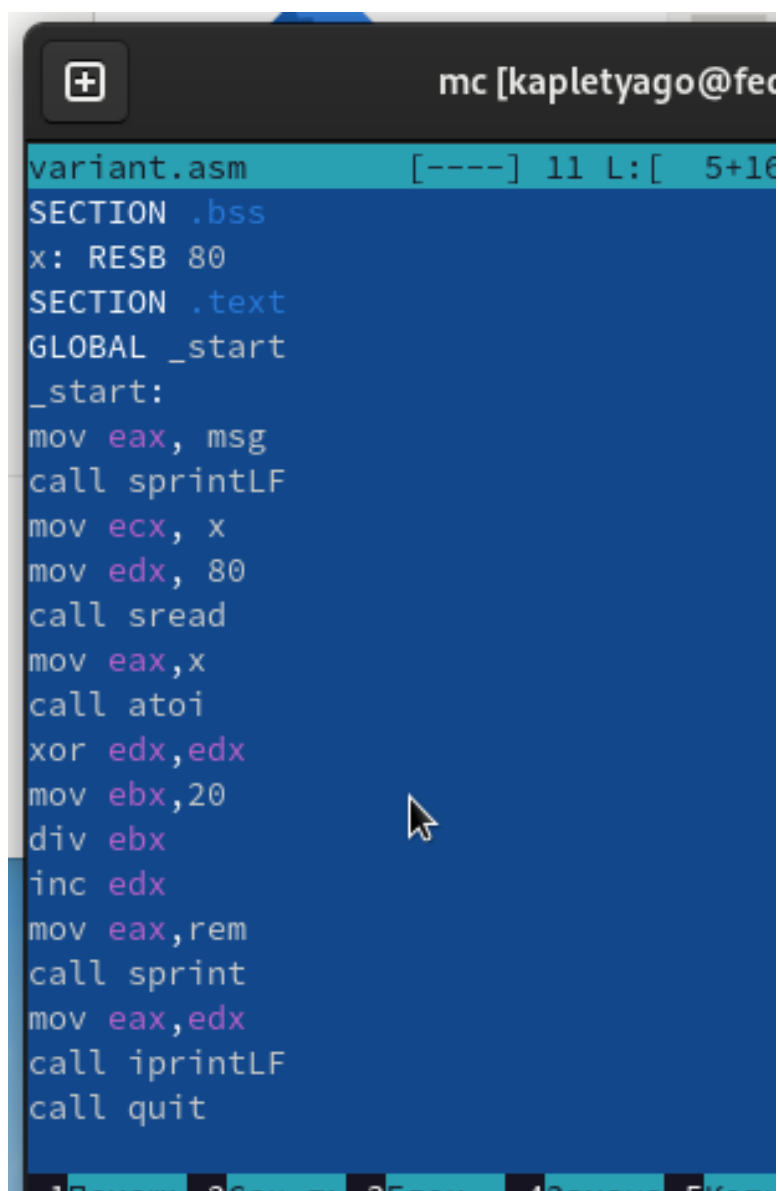
Рис. 2.14: Запуск программы lab6-3.asm

В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета. (рис. [2.15]) (рис. [2.16])

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.



```
variant.asm [----] 11 L: [ 5+16
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

```

[kapletyago@fedora lab06]$
[kapletyago@fedora lab06]$
[kapletyago@fedora lab06]$ nasm -f elf variant.asm
[kapletyago@fedora lab06]$ ld -m elf_i386 variant.o -o variant
[kapletyago@fedora lab06]$ ./variant
Введите № студенческого билета:
1132236107
Ваш вариант: 8
[kapletyago@fedora lab06]$
[kapletyago@fedora lab06]$

```

Рис. 2.16: Запуск программы variant.asm

## 2.1 Ответы на вопросы по программе variant.asm

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

Строка “mov eax, rem” перекладывает в регистр значение переменной с фразой “Ваш вариант:”

Строка “call sprint” вызывает подпрограмму вывода строки

2. Для чего используются следующие инструкции?

Инструкция “nasm” используется для компиляции кода на языке ассемблера NASM

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?

Строка `“xor edx, edx”` обнуляет регистр `edx`

Строка `“mov ebx, 20”` записывает значение 20 в регистр `ebx`

Строка `“div ebx”` выполняет деление номера студенческого билета на 20

Строка `“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 в регистре `edx` на 1, в соответствии с формулой вычисления варианта

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

## 2.2 Самостоятельное задание

Написать программу вычисления выражения  $y = f(x)$ . Программа должна вывести 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.

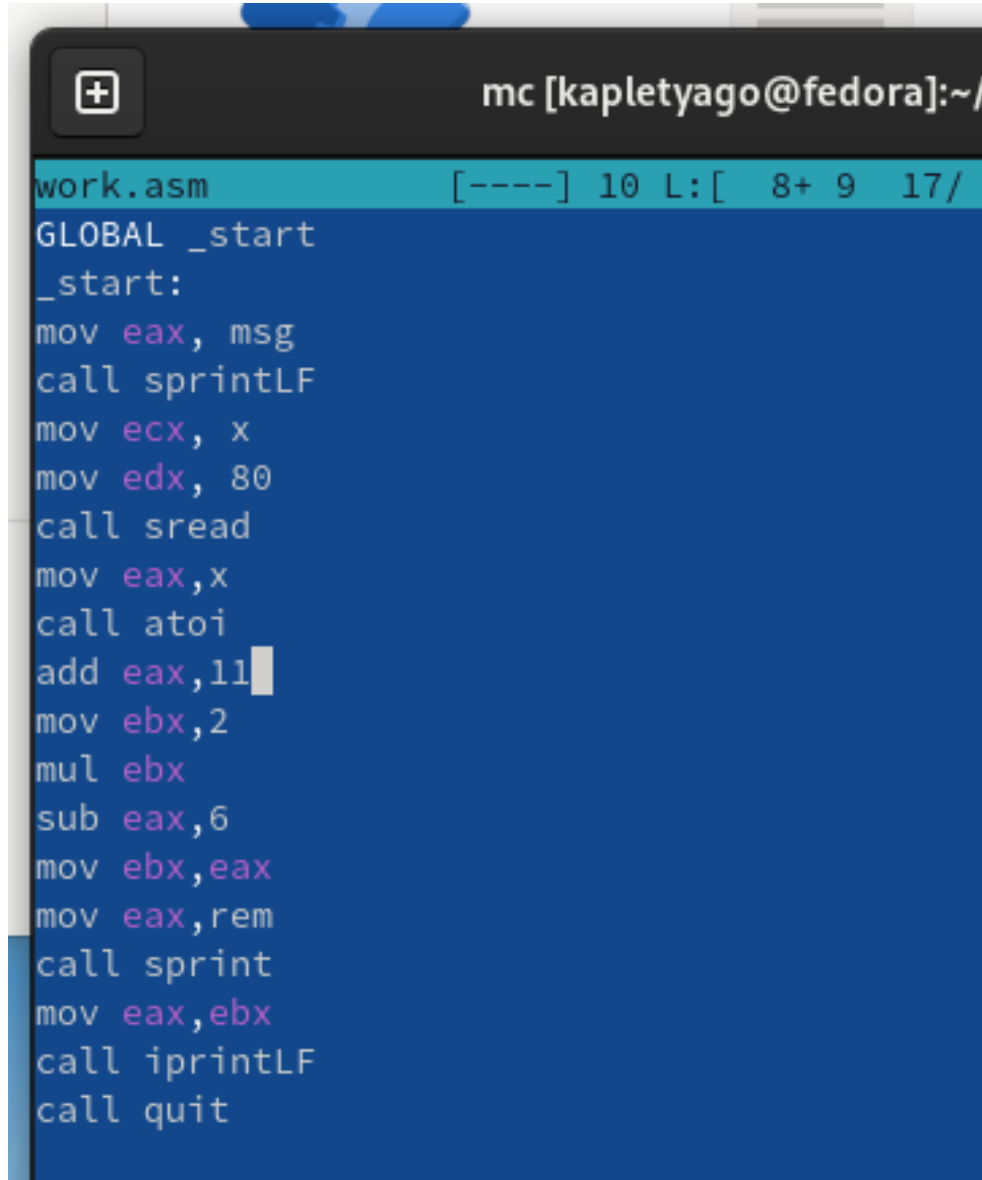
Получили вариант 8 -

$$(11 + x) * 2 - 6$$

для

$$x_1 = 1, x_2 = 9$$

(рис. [2.17]) (рис. [2.18])

A screenshot of a terminal window with a dark background. The title bar shows a plus icon and the text 'mc [kapletyago@fedora]:~/'. The terminal content shows the assembly code in 'work.asm'. The code includes a global \_start, a call to sprintf, a read system call, an atoi conversion, an addition, a multiplication, a subtraction, and calls to sprintf, iprintf, and quit. The cursor is at the end of the 'add eax, 11' line.

```
work.asm [----] 10 L:[ 8+ 9 17/  
GLOBAL _start  
_start:  
mov eax, msg  
call sprintf  
mov ecx, x  
mov edx, 80  
call sread  
mov eax, x  
call atoi  
add eax, 11  
mov ebx, 2  
mul ebx  
sub eax, 6  
mov ebx, eax  
mov eax, rem  
call sprintf  
mov eax, ebx  
call iprintf  
call quit
```

Рис. 2.17: Программа в файле work.asm

```
[kapletyago@fedora lab06]$  
[kapletyago@fedora lab06]$ touch work.asm  
[kapletyago@fedora lab06]$ nasm -f elf work.asm  
[kapletyago@fedora lab06]$ ld -m elf_i386 work.o -o work  
[kapletyago@fedora lab06]$ ./work  
Введите X  
1  
выражение = : 18  
[kapletyago@fedora lab06]$ ./work  
Введите X  
9  
выражение = : 34  
[kapletyago@fedora lab06]$
```

Рис. 2.18: Запуск программы task.asm

## **3 Выводы**

Изучили работу с арифметическими операциями.