

# Отчёт по лабораторной работе №6

## Архитектура компьютеров

Баштованович Анита

### Содержание

1	Цель работы .....	1
2	Задание .....	1
3	Теоретическое введение.....	Ошибка! Закладка не определена.
4	Выполнение лабораторной работы .....	Ошибка! Закладка не определена.
5	Выводы .....	Ошибка! Закладка не определена.
	Список литературы.....	Ошибка! Закладка не определена.

### 1 Цель работы

Освоить арифметические инструкции языка ассемблера NASM.

### 2 Задание

### 3. Основная часть

Создадим каталог для программ лабораторной работы №6, перейдем в него и создадим файл lab6-1.asm:(рис.1 [-@fig:001]).

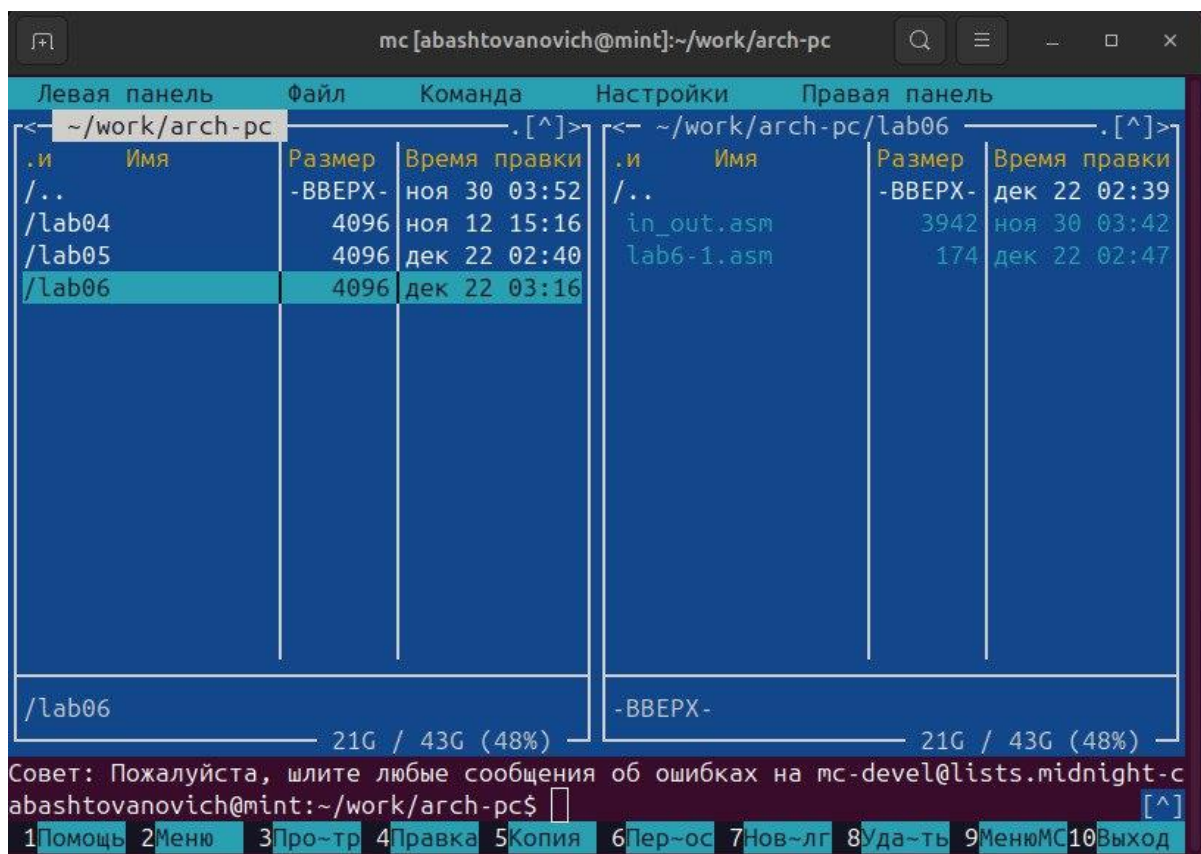
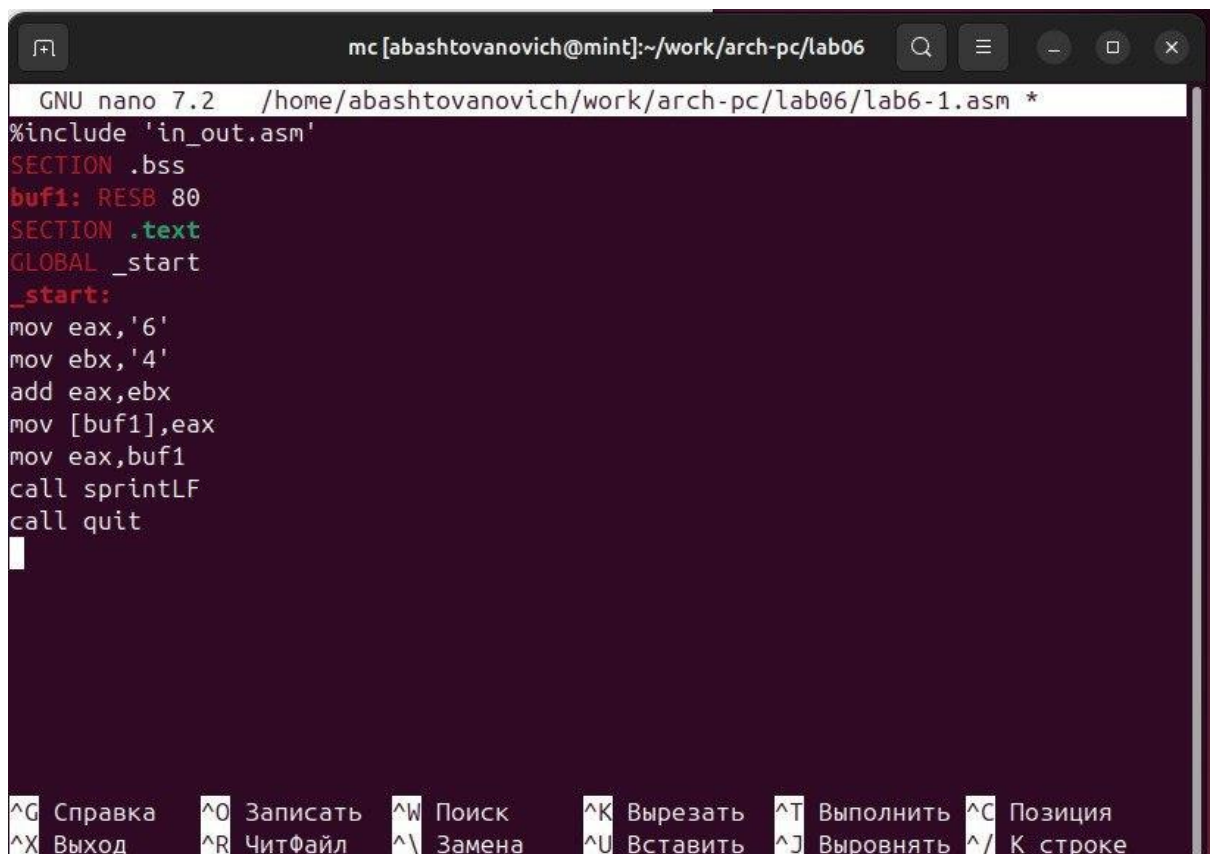


Рис. 1 Создание lab6-1.asm

Рассмотрим примеры программ вывода символьных численных значений. Программы будут выводить значения записанные в регистр еах.

Введем в файл lab6-1.asm текст программы из листинга 6.1. В данной программе в регистр еах записывается символ 6 (mov еах,'6'), в регистр еbх символ 4 (mov еbх,'4').

Далее к значению в регистре еах прибавляем значение регистра еbх (add еах,еbх, результат сложения запишется в регистр еах). Далее выводим результат. Так как для работы функции sprintfLF в регистр еах должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра еах в переменную buf1 (mov [buf1],еах), а затем запишем адрес переменной buf1 в регистр еах (mov еах,buf1) и вызовем функцию sprintfLF.(рис.2 [-@fig:002]).

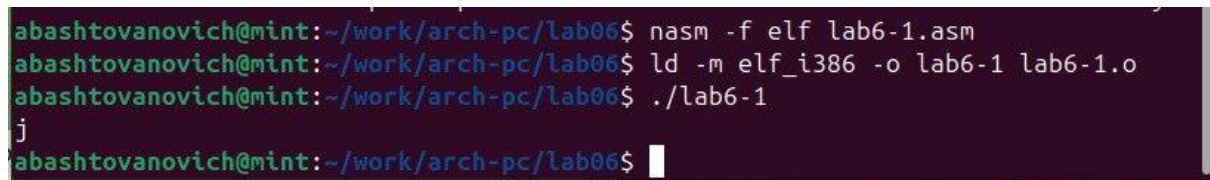


```
mc [abashtovanovich@mint]:~/work/arch-pc/lab06
GNU nano 7.2 /home/abashtovanovich/work/arch-pc/lab06/lab6-1.asm *
%include 'in_out.asm'
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
mov [buf1],eax
mov eax,buf1
call sprintf
call quit
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция  
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^/\_ К строке

Рис. 2 Текст листинга 6.1

Создадим исполняемый файл и запустим его.(рис.3 [-@fig:003]).

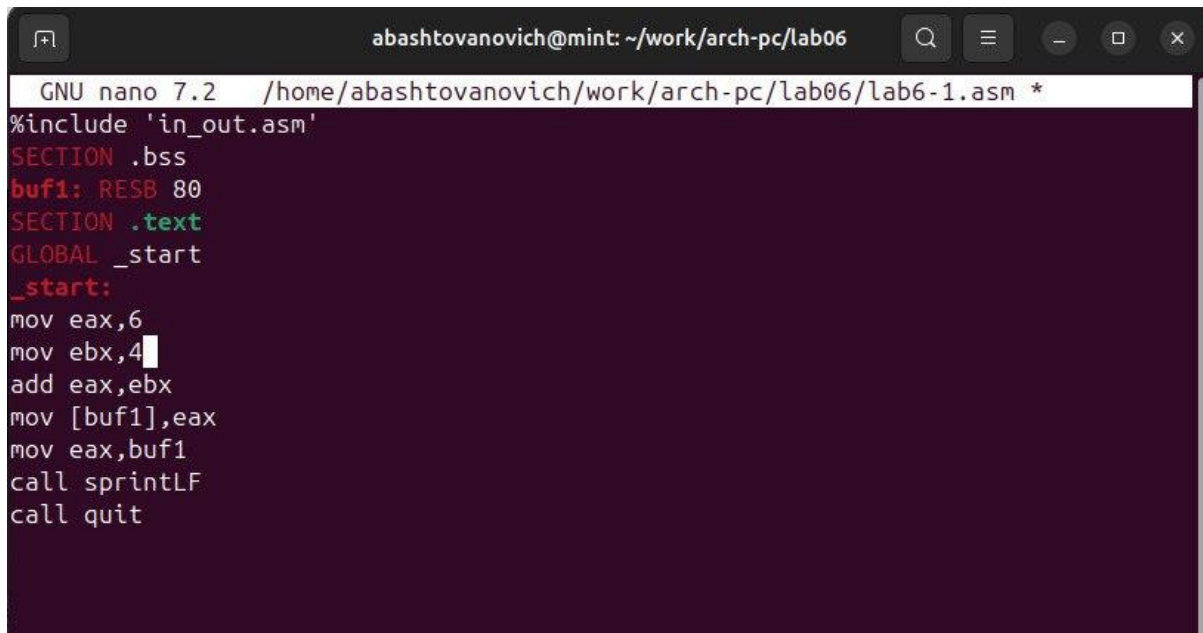


```
abashtovanovich@mint:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
abashtovanovich@mint:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
abashtovanovich@mint:~/work/arch-pc/lab06$ ./lab6-1
j
abashtovanovich@mint:~/work/arch-pc/lab06$
```

Рис. 3 Исполняемый файл для 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` (см. таблицу ASCII в приложении).

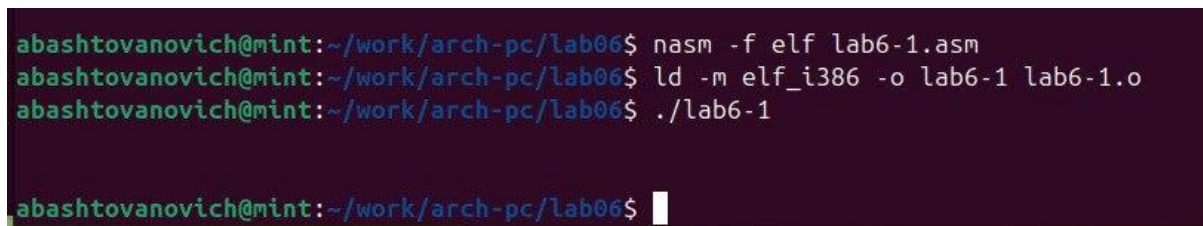
Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправим текст программы (Листинг 6.1) следующим образом: заменим строки(рис.4 [-@fig:004]).



```
GNU nano 7.2 /home/abashtovanovich/work/arch-pc/lab06/lab6-1.asm *
#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. 4 Замена строк

Создадим исполняемый файл и запустим его.(рис.5 [-@fig:005]).



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

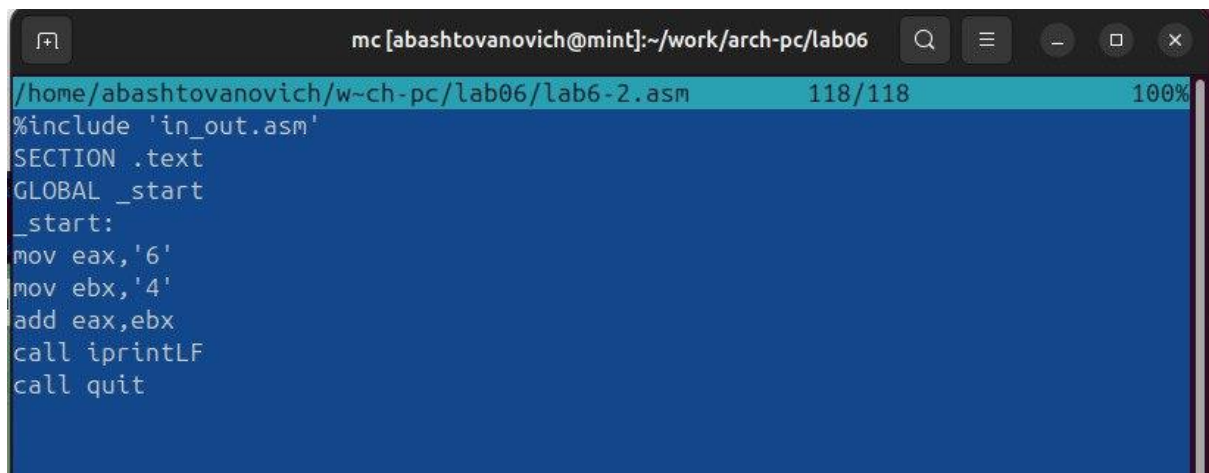
abashtovanovich@mint:~/work/arch-pc/lab06$
```

Рис. 5 Исполняемый файл с измененными строками

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. Преобразуем текст программы из Листинга 6.1 с использованием этих функций.

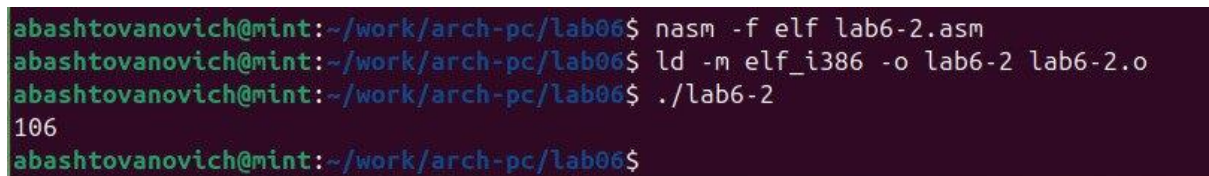
Создадим файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и введем в него текст программы из листинга 6.2.(рис.6 [-@fig:006]).

A screenshot of a code editor window titled 'mc [abashtovanovich@mint]:~/work/arch-pc/lab06'. The editor shows the file 'lab6-2.asm' at line 118 of 118, with 100% zoom. The code is as follows:

```
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

Рис. 6 Программа листинга 6.2

Создадим исполняемый файл и запустим его.(рис.7 [-@fig:007]).

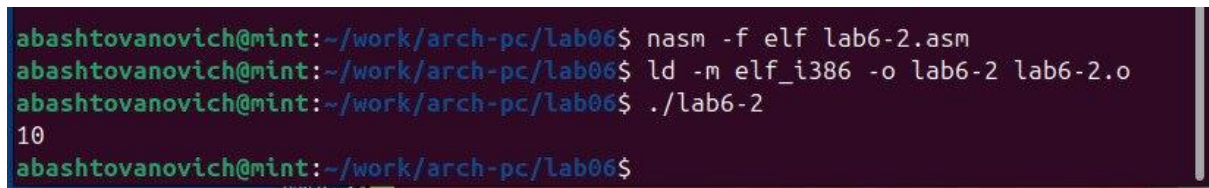
A screenshot of a terminal window showing the following commands and output:

```
abashtovanovich@mint:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
abashtovanovich@mint:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
abashtovanovich@mint:~/work/arch-pc/lab06$ ./lab6-2
106
abashtovanovich@mint:~/work/arch-pc/lab06$
```

Рис. 7 Исполняемый файл 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 программы из листинга 6.1, функция iprintLF позволяет вывести число, а не символ, кодом которого является это число.

Аналогично предыдущему примеру изменим символы на числа. Заменяем строки и создадим исполняемый файл и запустим его.(рис.8 [-@fig:008]).

A screenshot of a terminal window showing the following commands and output:

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

Рис. 8 Изменение строк и исполняемый файл

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

Заменяем функцию iprintLF на iprint. Создадим исполняемый файл и запустим его. (рис.9 [-@fig:009]).



```

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

```

Рис. 9 Замена `iprintLF` на `iprint`

Вывод отличается тем, что нет переноса строки.

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

Создадим файл `lab6-3.asm` в каталоге `~/work/arch-pc/lab06`: (рис.10 [-@fig:010]).

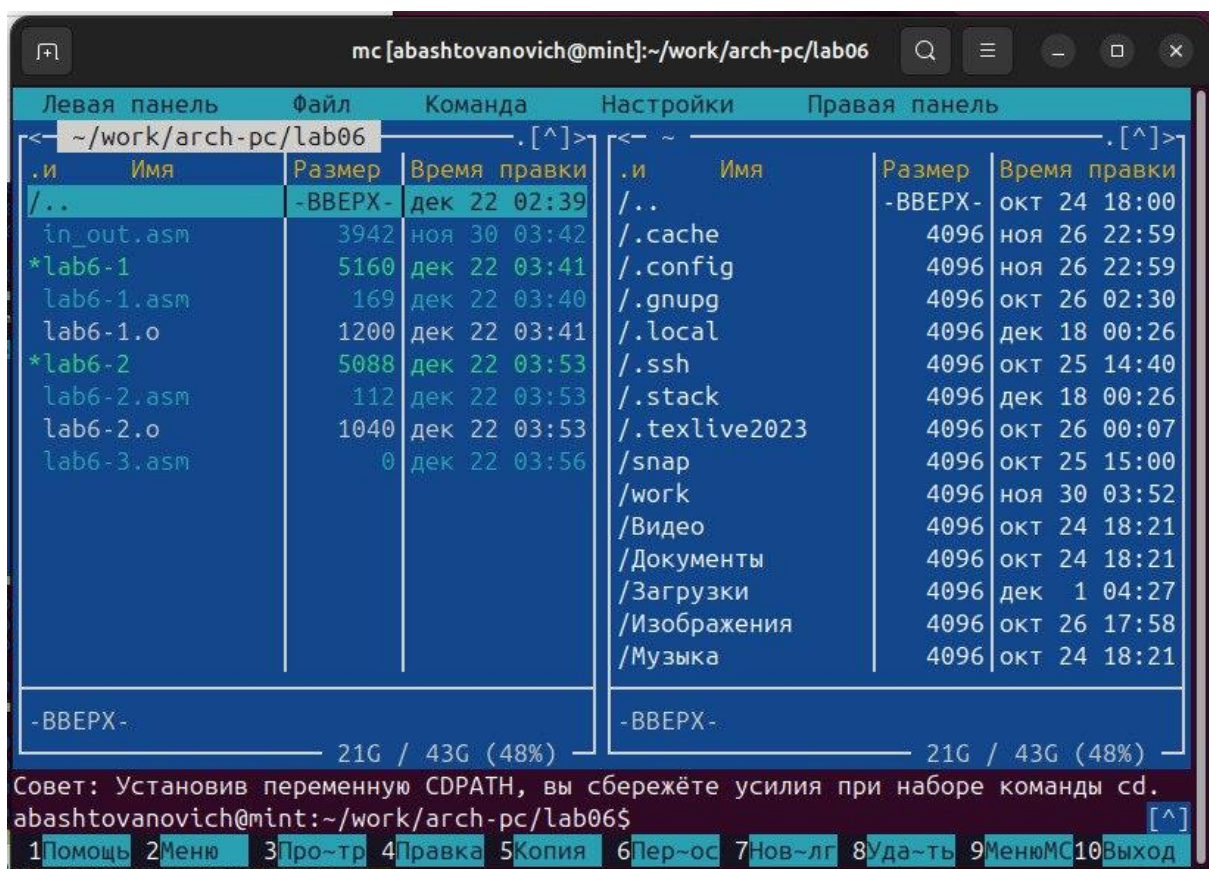
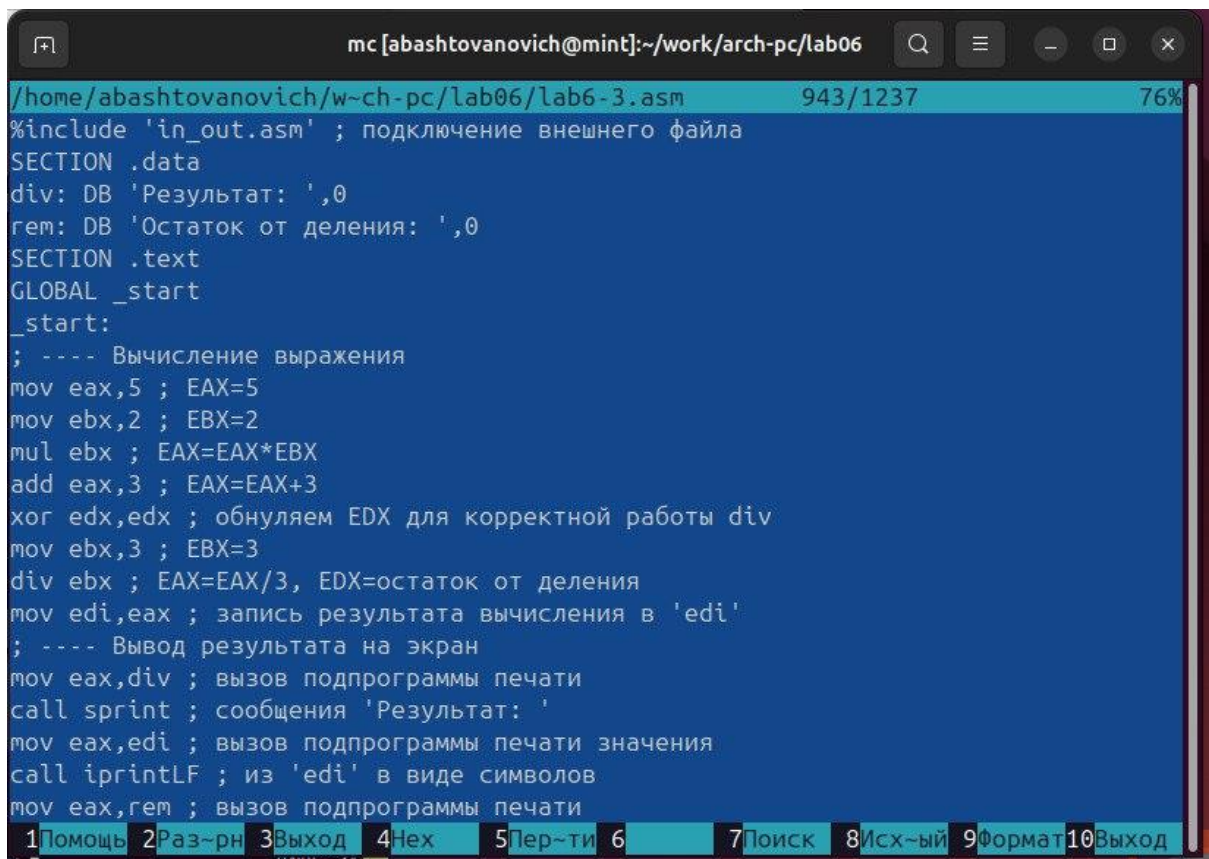


Рис. 10 Создание `lab6-3.asm`

Внимательно изучим текст программы из листинга 6.3 и введем в `lab6-3.asm`. (рис.11 [-@fig:011]).



```
mc [abashtovanovich@mint]:~/work/arch-pc/lab06 943/1237 76%
/home/abashtovanovich/w-ch-pc/lab06/lab6-3.asm
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax,5 ; EAX=5
mov ebx,2 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,3 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,3 ; EBX=3
div ebx ; EAX=EAX/3, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов
mov eax,rem ; вызов подпрограммы печати
1Помощь 2Раз~рн 3Выход 4Нех 5Пер~ти 6 7Поиск 8Исх~ый 9Формат10Выход
```

Рис. 11 Текст листинга 6.3

Создадим исполняемый файл и запустим его.(рис.12 [-@fig:012]).



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

Рис. 12 Исполняемый файл листинга 6.3

Изменим текст программы для вычисления выражения  $f(x) = (4 * 6 + 2)/5$ .(рис.13 [-@fig:013]).

```
mc [abashtovanovich@mint]:~/work/arch-pc/lab06
/home/abashtovanovich/w-ch-pc/lab06/lab6-3.asm 943/1237 76%
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax,4 ; EAX=5
mov ebx,6 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+3
xog edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; EBX=3
div ebx ; EAX=EAX/3, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов
mov eax,rem ; вызов подпрограммы печати
1Помощь 2Раз-рн 3Выход 4Нех 5Пер-ти 6 7Поиск 8Исх-ый 9Формат10Выход
```

Рис. 13 Программа для  $f(x) = (4 * 6 + 2)/5$

Создадим исполняемый файл и запустим его.(рис.14 [-@fig:014]).

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

Рис. 14 Проверка

В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму:

- вывести запрос на введение № студенческого билета
- вычислить номер варианта по формуле:  $(Sn \bmod 20) + 1$ , где  $Sn$  – номер студенческого билета (В данном случае  $a \bmod b$  – это остаток от деления  $a$  на  $b$ ).
- вывести на экран номер варианта.

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`.

Создадим файл `variant.asm` в каталоге `~/work/arch-pc/lab06`:(рис.15 [-@fig:015]).

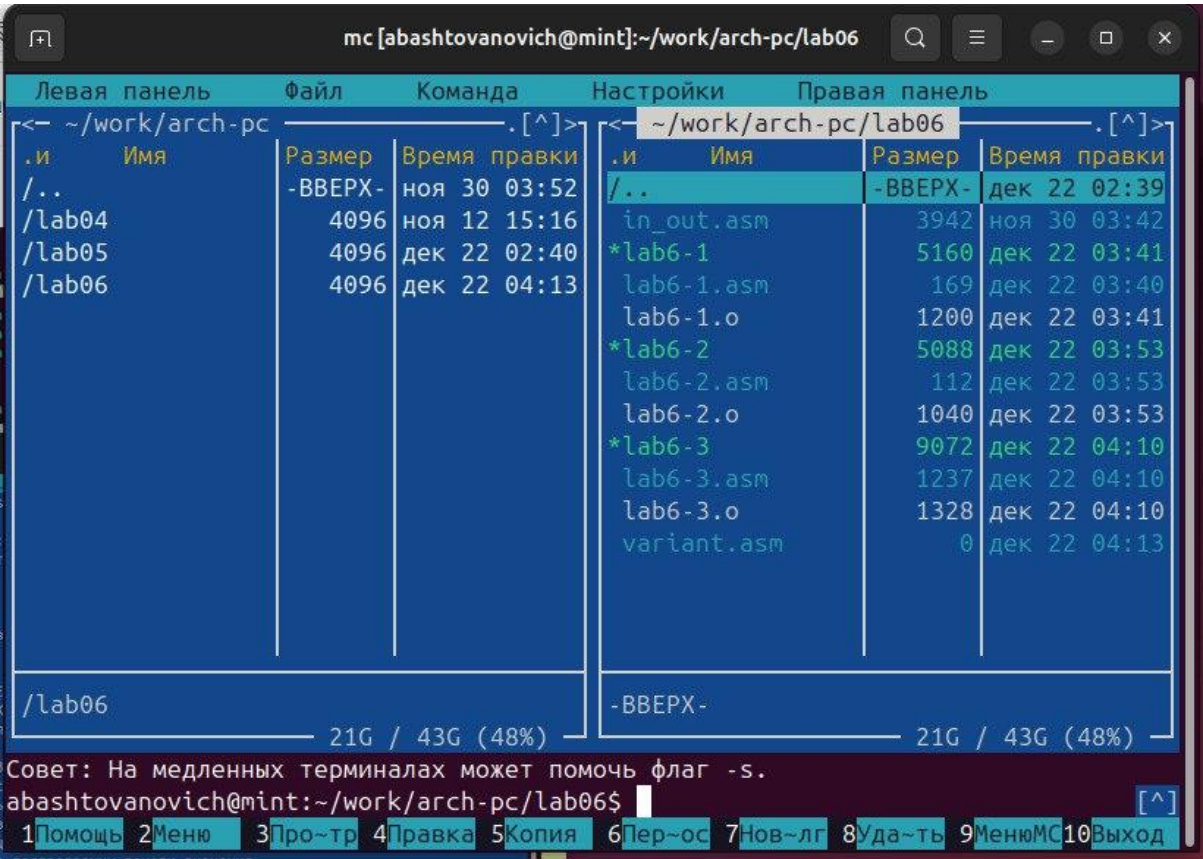
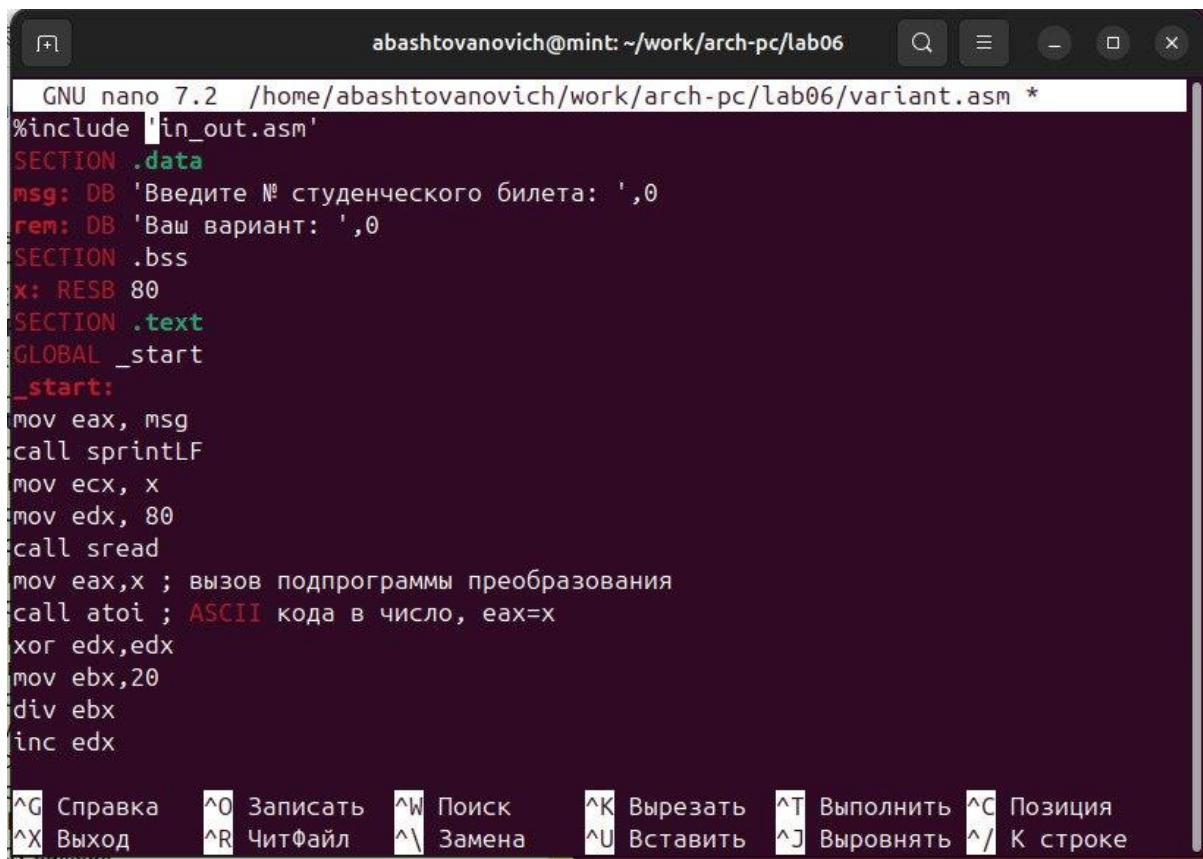


Рис. 15 Создание `variant.asm`

Внимательно изучим текст программы из листинга 6.4 и введем в файл `variant.asm`.(рис.16 [-@fig:016]).



```
GNU nano 7.2 /home/abashtovanovich/work/arch-pc/lab06/variant.asm *
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите № студенческого билета: ',0
rem: DB 'Ваш вариант: ',0
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 ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, eax=x
xor edx, edx
mov ebx, 20
div ebx
inc edx
```

^G Справка   ^O Записать   ^W Поиск   ^K Вырезать   ^T Выполнить   ^C Позиция  
^X Выход   ^R ЧитФайл   ^\ Замена   ^U Вставить   ^J Выводить   ^/ К строке

Рис. 16 Программа листинга 6.4

Создадим исполняемый файл и запустим его. (рис.17 [-@fig:017]).



```
abashtovanovich@mint:~/work/arch-pc/lab06$ nasm -f elf variant.asm
abashtovanovich@mint:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
abashtovanovich@mint:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032245372
Ваш вариант: 13
abashtovanovich@mint:~/work/arch-pc/lab06$
```

Рис. 17 Проверка листинга 6.4

## Ответы на вопросы

1. Какие строки листинга отвечают за вывод на экран сообщения 'Ваш вариант:'?
  - Инструкция "mov eax, rem" 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.

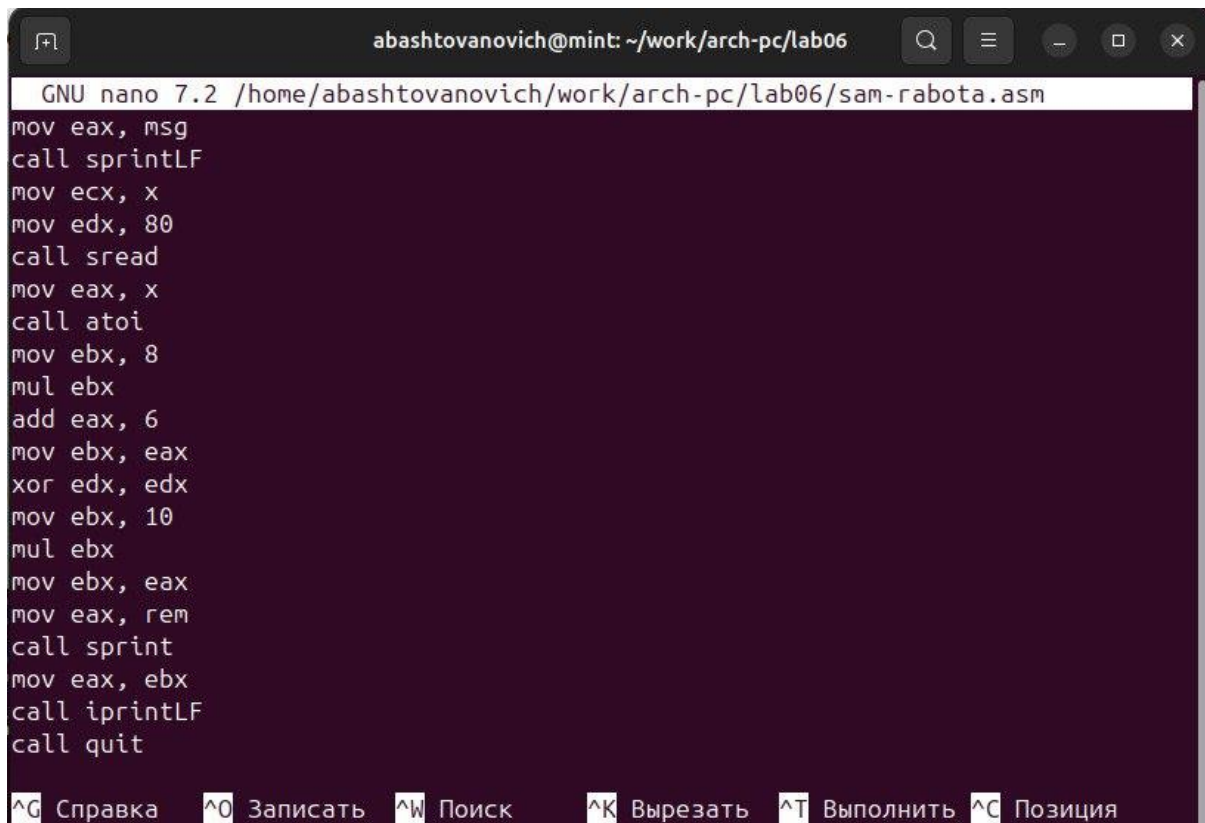
7. Какие строки листинга отвечают за вывод на экран результата вычислений?

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

## Задание для самостоятельной работы

Напишем 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.

Мой вид функции ВАРИАНТ 13 -  $f(x) = (8x + 6) \cdot 10$  проверить для  $x_1 = 1$  и  $x_2 = 4$ . (рис.18 [-@fig:018]).



```
GNU nano 7.2 /home/abashtovanovich/work/arch-pc/lab06/sam-rabota.asm
mov eax, msg
call sprintf
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov ebx, 8
mul ebx
add eax, 6
mov ebx, eax
xor edx, edx
mov ebx, 10
mul ebx
mov ebx, eax
mov eax, ret
call sprintf
mov eax, ebx
call iprintLF
call quit

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
```

Рис. 18 Программа для 13 выражения

Создадим исполняемый файл и запустим его для значения  $x_1 = 1$ . (рис.19 [-@fig:019]).



```
abashtovanovich@mint:~/work/arch-pc/lab06$ nasm -f elf sam-rabota.asm
abashtovanovich@mint:~/work/arch-pc/lab06$ ld -m elf_i386 -o sam-rabota sam-rabota.o
abashtovanovich@mint:~/work/arch-pc/lab06$ ./sam-rabota
Введите значение для x:
1
Выражение равно: 140
```

Рис. 19 Исполняемый файл для  $x_1$

Создадим исполняемый файл и запустим его для значения  $x_2 = 4$ . (рис.20 [-@fig:020]).



```
abashtovanovich@mint:~/work/arch-pc/lab06$ nasm -f elf sam-rabota.asm
abashtovanovich@mint:~/work/arch-pc/lab06$ ld -m elf_i386 -o sam-rabota sam-rabota.o
abashtovanovich@mint:~/work/arch-pc/lab06$ ./sam-rabota
Введите значение для x:
4
Выражение равно: 380
```

*Рис. 20 Исполняемый файл для x2*

## 4. Выводы

Освоены арифметические инструкции языка ассемблера NASM.