

Отчёт по лабораторной работе № 6

**дисциплина: Архитектура компьютера. Арифметические
операции в NASM**

Студент: Святашова Ксения Евгеньевна

Содержание

1	Цель работы	4
2	Теоритическое введение	5
3	Выполнение лабораторной работы	7
3.1	Символьные и численные данные в NASM	7
3.2	Выполнение арифметических операций в NASM	17
3.3	Ответы на вопросы	24
4	Задания для самостоятельной работы	26
5	Вывод	30

Список иллюстраций

3.1	Каталог lab06	7
3.2	Файл lab6-1.asm	8
3.3	Ввод текста из листинга 1	9
3.4	Копирование файла in_out.asm	10
3.5	Создание файла lab5-1.asm	10
3.6	Изменение текста программы	11
3.7	Запуск исполняемого файла	12
3.8	Создание файла lab6-2.asm	12
3.9	Ввод текста из листинга 2	13
3.10	Запуск исполняемого файла	14
3.11	Изменение файла lab6-2.asm	15
3.12	Запуск исполняемого файла	15
3.13	Замена функции iprintLF на iprint	16
3.14	Запуск исполняемого файла	16
3.15	Создание файла lab6-3.asm	17
3.16	Ввод текста из листинга 3	18
3.17	Запуск исполняемого файла	19
3.18	Изменения программы	20
3.19	Запуск исполняемого файла	20
3.20	Создание файла variant.asm	21
3.21	Ввод текста из листинга 4	22
3.22	Запуск исполняемого файла	24
4.1	Создание файла lab6-4.asm	26
4.2	Программа вычисления выражение из варианта 9	27
4.3	Запуск исполняемого файла	28
4.4	Запуск исполняемого файла	29

1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

2 Теоритическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. Существует три основных способа адресации:

- **Регистровая адресация** – операнды хранятся в регистрах и в команде используются имена этих регистров, например: `mov ax,bx`.

- **Непосредственная адресация** – значение операнда задается непосредственно в команде, Например: `mov ax,2`.

- **Адресация памяти** – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

Схема команды целочисленного сложения `add` (от англ. addition - добавление) выполняет сложение двух операндов и записывает результат по адресу первого операнда. Команда `add` работает как с числами со знаком, так и без знака.

Довольно часто при написании программ встречается операция прибавления или вычитания единицы. Прибавление единицы называется инкрементом, а вычитание — декрементом.

Для команд умножения один из сомножителей указывается в команде и должен находиться в регистре или в памяти, но не может быть

непосредственным операндом. Второй сомножитель в команде явно не указывается и должен находиться в регистре EAX, AX или AL, а результат помещается в регистры EDX:EAX, DX:AX или AX.

В командах указывается только один операнд – делитель, который может быть регистром или ячейкой памяти, но не может быть непосредственным операндом. Местоположение делимого и результата для команд деления зависит от размера делителя.

Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно стандарту ASCII каждый символ кодируется одним байтом

3 Выполнение лабораторной работы

3.1 Символьные и численные данные в NASM

1. Создадим каталог для программ лабораторной работы № 6, перейдем в него и создадим файл lab6-1.asm:(рис. 3.1):

```
kesvyatashova@fedora:~$ mkdir ~/work/arch-pc/lab06
kesvyatashova@fedora:~$ cd ~/work/arch-pc/lab06
kesvyatashova@fedora:~/work/arch-pc/lab06$ touch lab6-1.asm
kesvyatashova@fedora:~/work/arch-pc/lab06$
```

Рис. 3.1: Каталог lab06

2. Введем в файл lab6-1.asm(рис. 3.2) текст программы из листинга 1(рис. 3.3).

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. 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`.

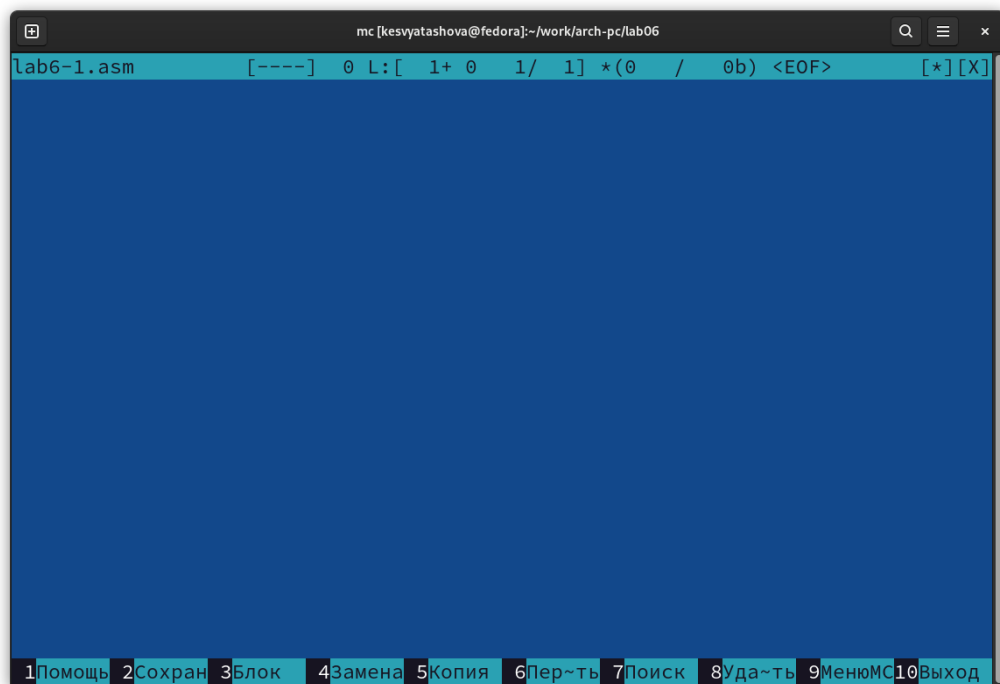
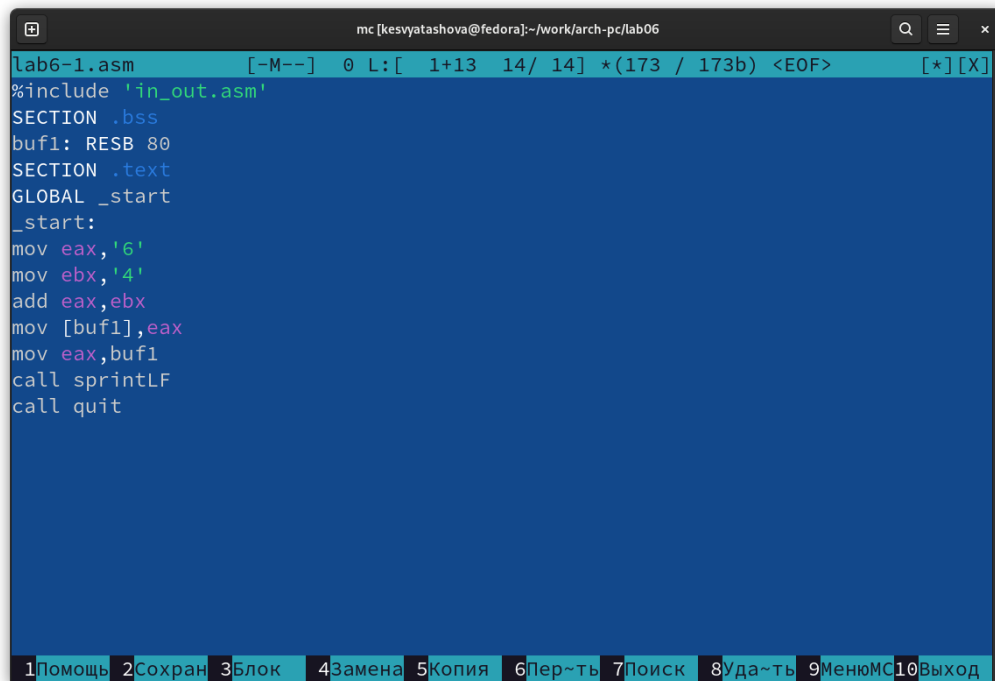


Рис. 3.2: Файл `lab6-1.asm`



```
lab6-1.asm [-M--] 0 L: [ 1+13 14/ 14] *(173 / 173b) <EOF> [*] [X]
#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
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС10Выход

Рис. 3.3: Ввод текста из листинга 1

Листинг 1. Программа вывода значения регистра eax

```
%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
```

Для корректной работы программы подключаемый файл `in_out.asm` должен лежать в том же каталоге, что и файл с текстом программы. Перед созданием исполняемого файла создайте копию файла в каталоге `~/work/arch-pc/lab06` (рис. 3.4):

```
kesvyatashova@fedora:~$ cp ~/work/arch-pc/lab05/in_out.asm ~/work/arch-pc/lab06
kesvyatashova@fedora:~$
```

Рис. 3.4: Копирование файла `in_out.asm`

Создадим исполняемый файл и запустим его (рис. 3.5):

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

Рис. 3.5: Создание файла `lab5-1.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` (в таблице ASCII).

3. Далее изменим текст программы(рис. 3.6) и вместо символов, запишем в регистры числа. Исправим текст программы (листинг 1) следующим образом:

заменим строки

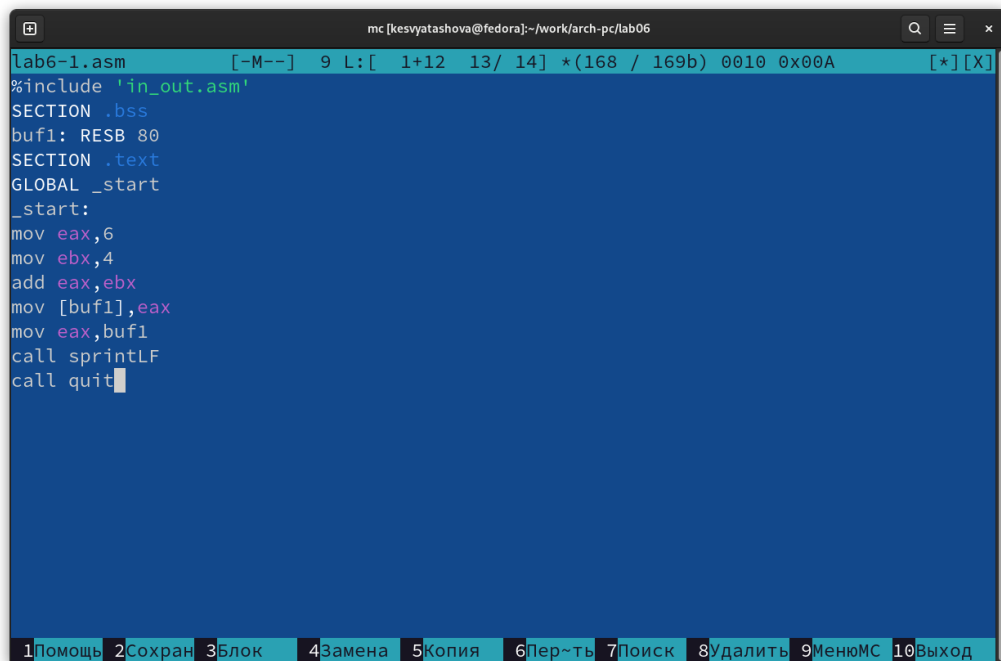
`mov eax,'6'`

`mov ebx,'4'`

на строки

`mov eax,6`

`mov ebx,4`



```
lab6-1.asm [-M--] 9 L: [ 1+12 13/ 14] *(168 / 169b) 0010 0x00A [*] [X]
#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. 3.6: Изменение текста программы

Создим исполняемый файл и запустим его(рис. 3.7):

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

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

Рис. 3.7: Запуск исполняемого файла

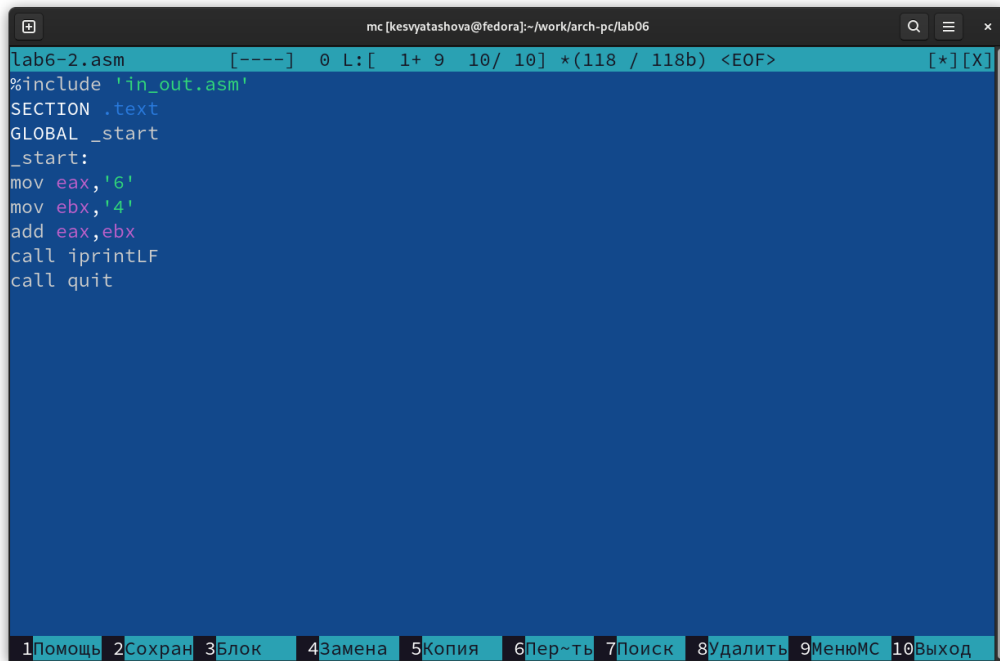
Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10. По таблице ASCII код 10 соответствует символу перевода строки, поэтому он не отображается при выводе на экран.

4. 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` (рис. 3.8) и введем в него текст программы из листинга 2 (рис. 3.9):

```
kesvyatashova@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-2.asm
kesvyatashova@fedora:~/work/arch-pc/lab06$
```

Рис. 3.8: Создание файла `lab6-2.asm`



```
lab6-2.asm [----] 0 L: [ 1+ 9 10/ 10] *(118 / 118b) <EOF> [*] [X]
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Удалить 9МенюМС 10Выход

Рис. 3.9: Ввод текста из листинга 2

Листинг 2. Программа вывода значения регистра `eax`

```
%include 'in_out.asm'
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
mov eax,'6'
```

```
mov ebx,'4'
```

```
add eax,ebx
```

```
call iprintLF
```

```
call quit
```

Создадим исполняемый файл и запустим его(рис. 3.10):

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

Рис. 3.10: Запуск исполняемого файла

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 1, функция `iprintLF` позволяет вывести число, а не символ, кодом которого является это число.

5. Аналогично предыдущему примеру изменим символы на числа(рис. 3.11).

Заменяем строки

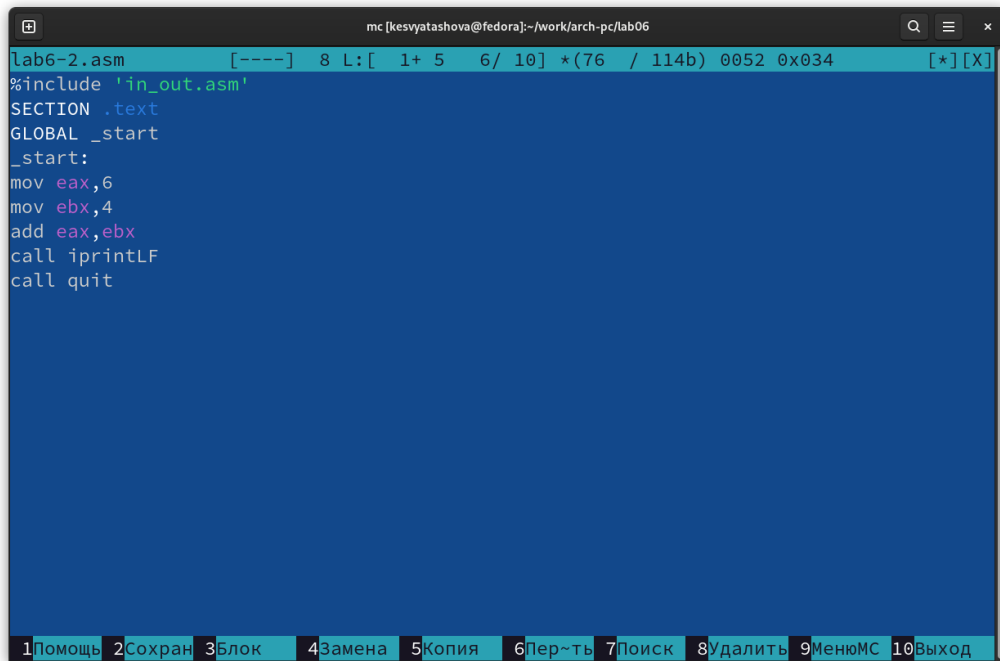
```
mov eax,'6'
```

```
mov ebx,'4'
```

на строки

```
mov eax,6
```

```
mov ebx,4
```

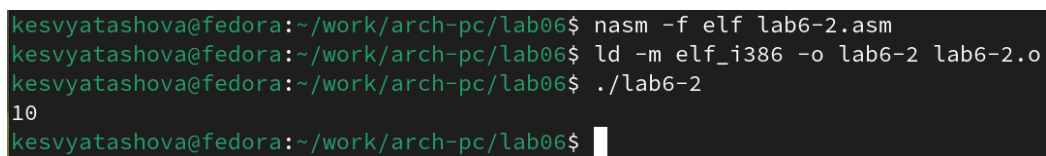


```
lab6-2.asm [----] 8 L: [ 1+ 5 6/ 10] *(76 / 114b) 0052 0x034 [*] [X]
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Удалить 9МенюМС 10Выход

Рис. 3.11: Изменение файла lab6-2.asm

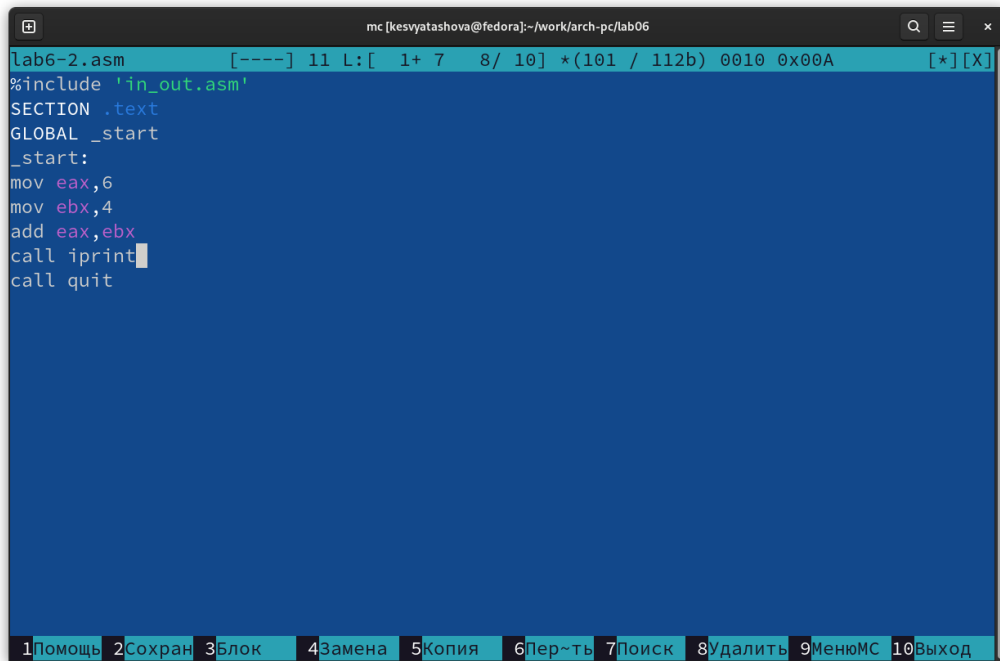
Создадим исполняемый файл и запустим его(рис. 3.12). Программа вывела результат 10, потому что теперь программа складывает не коды, которые соответствуют символам в системе ASCII, а сами числа.



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

Рис. 3.12: Запуск исполняемого файла

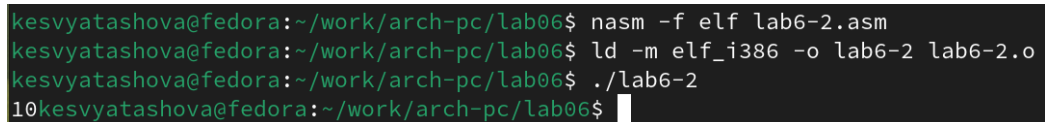
Заменяем функцию iprintLF на iprint(рис. 3.13):



```
lab6-2.asm [----] 11 L: [ 1+ 7 8/ 10] *(101 / 112b) 0010 0x00A [*] [X]
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit
```

Рис. 3.13: Замена функции iprintLF на iprint

Создадим исполняемый файл и запустим его(рис. 3.14):



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

Рис. 3.14: Запуск исполняемого файла

iprint не добавляет к выводу символ переноса строки, в отличие от iprintLF.

3.2 Выполнение арифметических операций в NASM

6. 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

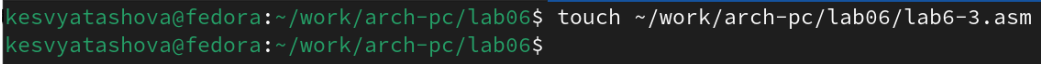
A terminal window with a dark background and green text. The prompt is 'kesvyatashova@fedora:~/work/arch-pc/lab06\$'. The command 'touch ~/work/arch-pc/lab06/lab6-3.asm' is entered and executed. The next line shows the prompt again: 'kesvyatashova@fedora:~/work/arch-pc/lab06\$'.

Рис. 3.15: Создание файла lab6-3.asm

Введем в lab6-3.asm текст программы из листинга 3(рис. 3.16):

```

lab6-3.asm [-M--] 50 L: [ 1+ 0 1/ 24] *(74 /1134b) 0010 0x00A [*] [X]
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
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'
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintfLF ; из 'edi' в виде символов
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintfLF ; из 'edx' (остаток) в виде символов
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Удалить 9МенюМС 10Выход

```

Рис. 3.16: Ввод текста из листинга 3

Листинг 3. Программа вычисления выражения $f(x) = (5 * 2 + 3)/3$

;----- ; Программа вычисления выражения ;-----

```
%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,edx ; вызов подпрограммы печати
call sprint ; сообщения 'Остаток от деления:'
mov eax,edx ; вызов подпрограммы печати значения
call iprintLF ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения
Создадим исполняемый файл и запустим его(рис. 3.17):

```

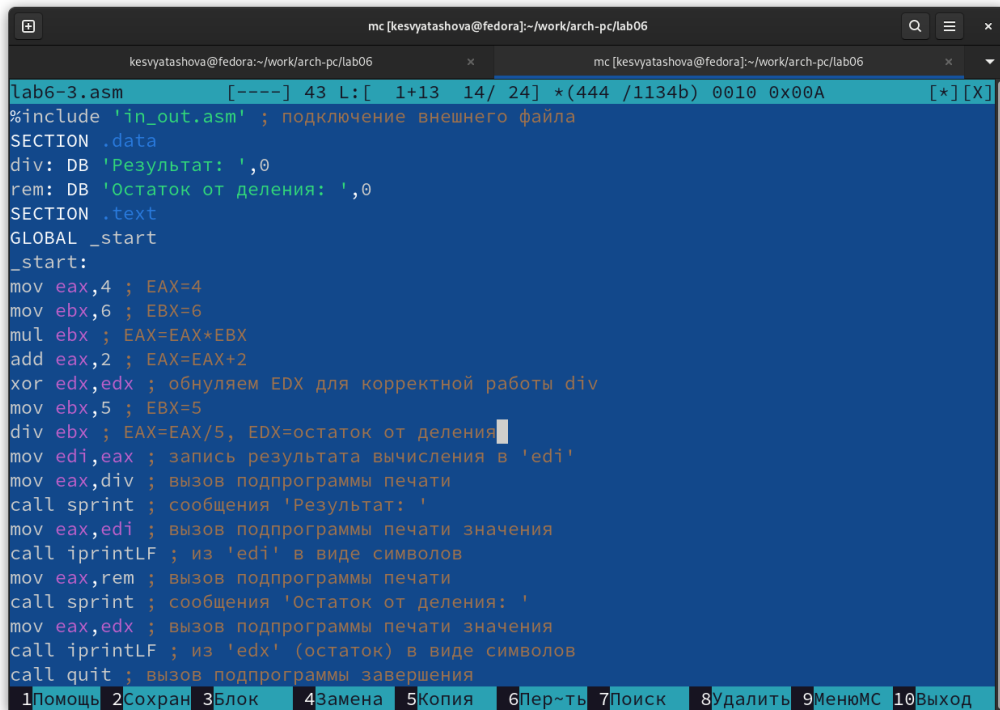
```

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

```

Рис. 3.17: Запуск исполняемого файла

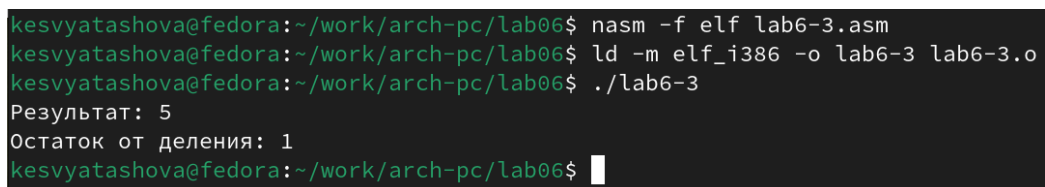
Изменим текст программы для вычисления выражения $f(x) = (4 * 6 + 2)/5$ (рис. 3.18):



```
lab6-3.asm  [----] 43 L: [ 1+13 14/ 24] *(444 /1134b) 0010 0x00A  [*] [X]
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
mov eax,4 ; EAX=4
mov ebx,6 ; EBX=6
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+2
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; EBX=5
div ebx ; EAX=EAX/5, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
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
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintLF ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Удалить 9МенюМС 10Выход
```

Рис. 3.18: Изменения программы

Создадим исполняемый файл и проверим его работу(рис. 3.19):



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

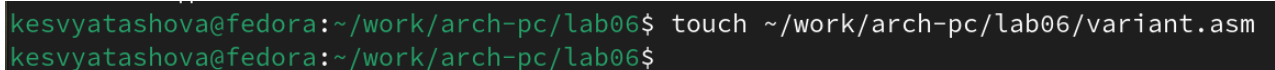
Рис. 3.19: Запуск исполняемого файла

7. 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: $(S_n \bmod 20) + 1$, где S_n – 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. 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. 3.20):



```
kesvyatashova@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm
kesvyatashova@fedora:~/work/arch-pc/lab06$
```

Рис. 3.20: Создание файла `variant.asm`

Введем в файл программу из листинга 4(рис. 3.21):

```
variant.asm [----] 9 L:[ 1+16 17/ 25] *(402 / 491b) 0100 0x064 [*] [X]
#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 sprintLF
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
mov eax, rem
call sprint
mov eax, edx
call iprintLF
call quit
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Удалить 9МенюМС 10Выход
```

Рис. 3.21: Ввод текста из листинга 4

Листинг 4. Программа вычисления вычисления варианта задания по номеру студенческого билета

```
;-----; Программа вычисления варианта ;-----
#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 sprintLF
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
mov eax, rem
call sprint
mov eax, edx
call iprintLF
call quit
Создадим исполняемый файл и запустим его(рис. 3.22):

```

```
kesvyatashova@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
kesvyatashova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
kesvyatashova@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246768
Ваш вариант: 9
kesvyatashova@fedora:~/work/arch-pc/lab06$
```

Рис. 3.22: Запуск исполняемого файла

Мой вариант - 9.

3.3 Ответы на вопросы

1. За вывод сообщения “Ваш вариант” в листинге 4 отвечают строки кода:

```
mov eax,rem
call sprint
```

2. Инструкция `mov ecx, x` используется, чтобы положить адрес вводимой строки `x` в регистр `ecx`; `mov edx, 80` - запись в регистр `edx` длины вводимой строки; `call sread` - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры.
3. Инструкция “`call atoi`” используется для вызова подпрограммы из внешнего файла, которая преобразует `ascii`-код символа в целое число и записывает результат в регистр `eax`.

4. За вычисления варианта в листинге 4 отвечают строки:

```
xor edx,edx ; обнуление edx для корректной работы div
```



```
mov ebx,20 ; ebx = 20
```

```
div ebx ; eax = eax/20, edx - остаток от деления
```

```
inc edx ; edx = edx + 1
```

5. При выполнении инструкции `div ebx` остаток от деления записывается в регистр `edx`.
6. Инструкция `inc edx` увеличивает значение регистра `edx` на 1.
7. За вывод на экран результатов вычислений из листинга 4 отвечают строки:

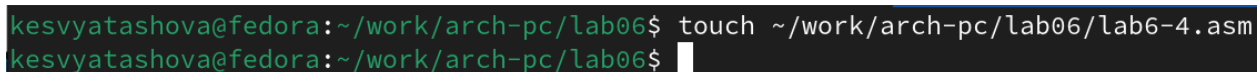
```
mov eax,edx
```

```
call iprintLF
```

4 Задания для самостоятельной работы

Напишем программу вычисления выражения $10 + (31x - 5)$ (выражение под вариантом 9). Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x , вычислять заданное выражение в зависимости от введенного x , выводить результат вычислений.

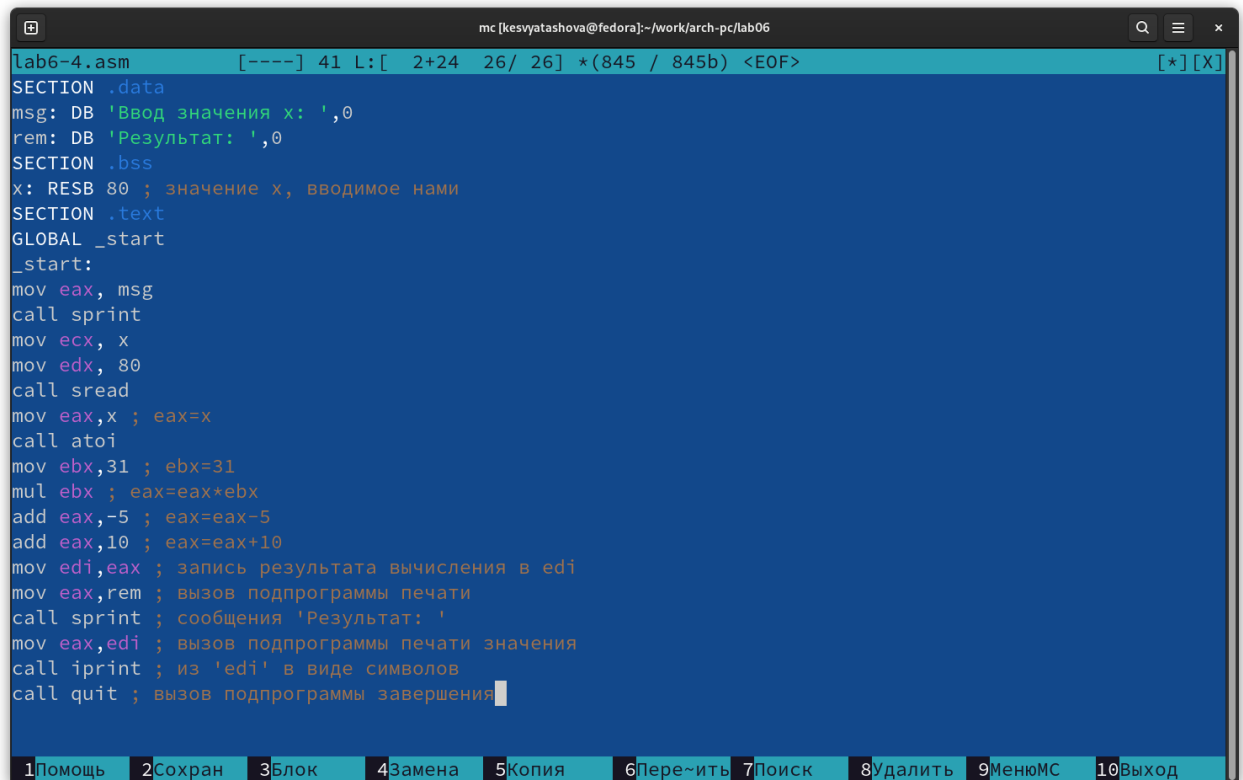
Создадим файл `lab6-4.asm` в каталоге `~/work/arch-pc/lab06` (рис. 4.1):



```
kesvyatashova@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-4.asm
kesvyatashova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.1: Создание файла `lab6-4.asm`

Открываем созданный файл для редактирования, вводим в него текст программы для вычисления значения выражения $10 + (31x - 5)$ (рис. 4.2):



```
lab6-4.asm [----] 41 L: [ 2+24 26/ 26] *(845 / 845b) <EOF> [*] [X]
SECTION .data
msg: DB 'Ввод значения x: ',0
rem: DB 'Результат: ',0
SECTION .bss
x: RESB 80 ; значение x, вводимое нами
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x ; eax=x
call atoi
mov ebx, 31 ; ebx=31
mul ebx ; eax=eax*ebx
add eax, -5 ; eax=eax-5
add eax, 10 ; eax=eax+10
mov edi, eax ; запись результата вычисления в edi
mov eax, rem ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax, edi ; вызов подпрограммы печати значения
call iprint ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пере-ить 7Поиск 8Удалить 9МенюМС 10Выход

Рис. 4.2: Программа вычисления выражение из варианта 9

Текст программы:

%include 'in_out.asm' ; подключение внешнего файла

SECTION .data

msg: DB 'Ввод значения x:',0

rem: DB 'Результат:',0

SECTION .bss

x: RESB 80 ; значение x, вводимое нами

SECTION .text

GLOBAL _start

_start:

```

mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax,x ; eax=x
call atoi
mov ebx,31 ; ebx=31
mul ebx ; eax=eax*ebx
add eax,-5 ; eax=eax-5
add eax,10 ; eax=eax+10
mov edi,eax ; запись результата вычисления в edi
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Результат:'
mov eax,edi ; вызов подпрограммы печати значения
call iprint ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения
Создадим исполняемый файл и запустим его для x=3(рис. 4.3):

```

```

kesvyatashova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
kesvyatashova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
kesvyatashova@fedora:~/work/arch-pc/lab06$ ./lab6-4
Ввод значения x: 3
Результат: 98kesvyatashova@fedora:~/work/arch-pc/lab06$

```

Рис. 4.3: Запуск исполняемого файла

Проведем еще один запуск файла для x=1 для проверки(рис. 4.4):

```
kesvyatashova@fedora:~/work/arch-pc/lab06$ ./lab6-4  
Ввод значения x: 1  
Результат: 36kesvyatashova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.4: Запуск исполняемого файла

Все работает верно.

5 Вывод

В результате выполнения работы я освоила арифметические инструкции языка ассемблера NASM.