

# **Отчёт по лабораторной работе №6**

**Дисциплина: Архитектура компьютеров**

Карпова Анастасия Александровна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Выводы</b>	<b>18</b>
	<b>Список литературы</b>	<b>19</b>

## Список иллюстраций

4.1	Создание каталога lab06 . . . . .	8
4.2	Создание файла lab7-1.asm . . . . .	8
4.3	Копирование файла . . . . .	8
4.4	Ввод программы значения регистра еах . . . . .	9
4.5	Создание исполняемого файла и его запуск . . . . .	9
4.6	Изменение строки . . . . .	10
4.7	Создание нового исполняемого файла и его запуск . . . . .	10
4.8	Создание исполняемого файла . . . . .	11
4.9	Ввод программы значения регистра еах . . . . .	11
4.10	Создание и запуск исполняемого файла . . . . .	11
4.11	Замена . . . . .	12
4.12	Создание и запуск исполняемого файла . . . . .	12
4.13	Замена . . . . .	12
4.14	Создание и запуск исполняемого файла . . . . .	13
4.15	Создание файла . . . . .	13
4.16	Ввод программы . . . . .	13
4.17	Создание файла . . . . .	14
4.18	Изменение программы . . . . .	14
4.19	Создание и запуск исполняемого файла . . . . .	14
4.20	Создание файла . . . . .	15
4.21	Ввод программы . . . . .	15
4.22	Создание и запуск исполняемого файла . . . . .	15
4.23	Создание файла . . . . .	16
4.24	Редактирование . . . . .	17
4.25	Создание и запуск исполняемого файла . . . . .	17

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

## 2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических действий в NASM
3. Выполнение заданий для самостоятельной работы

### 3 Теоретическое 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. Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации. 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. 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. Среди инструкций NASM нет такой, которая выводит числа (не в символьном виде). Поэтому, например, чтобы вывести число, надо предварительно преобразовать его цифры в ASCII-коды этих цифр и выводить на экран эти коды, а не само число. Если же выводить число на экран непосредственно, то экран воспримет его не как число, а как последовательность ASCII-символов – каждый байт числа будет воспринят как один ASCII-символ – и выведет на экран эти символы. Для выполнения лабораторных работ в файле `in_out.asm` реализованы подпрограммы

для преобразования ASCII символов в числа и обратно. Это:

- `iprint` – вывод на экран чисел в формате ASCII, перед вызовом `iprint` в регистр `eax` необходимо записать выводимое число (`mov eax, ...`).
- `iprintLF` – работает аналогично `iprint`, но при выводе на экран после числа добавляет к символ перевода строки.
- `atoi` – функция преобразует `ascii`-код символа в целое число и записывает результат в регистр `eax`, перед вызовом `atoi` в регистр `eax` необходимо записать число (`mov eax, ...`).

## 4 Выполнение лабораторной работы

### 4.1 Символьные и численные данные в NASM

С помощью `mkdir` создаю каталог `lab06` в `~/work/arch-pc`. Перехожу в созданный каталог с помощью утилиты `cd` (рис. 4.1).

```
akarpova@Justclown:~$ cd work/arch-pc
akarpova@Justclown:~/work/arch-pc$ mkdir lab06
akarpova@Justclown:~/work/arch-pc$ ls
lab04 lab05 lab06
```

Рис. 4.1: Создание каталога `lab06`

С помощью утилиты `touch` создаю файл `lab6-1.asm` (рис. 4.2).

```
akarpova@Justclown:~/work/arch-pc$ cd lab06
akarpova@Justclown:~/work/arch-pc/lab06$ touch lab6-1.asm
akarpova@Justclown:~/work/arch-pc/lab06$ ls
lab6-1.asm
```

Рис. 4.2: Создание файла `lab6-1.asm`

Копирую в текущий каталог файл `in_out.asm` с помощью утилиты `cp` (рис. 4.3).

```
akarpova@Justclown:~/work/arch-pc/lab06$ cp ~/Downloads/in_out.asm in_out.asm
akarpova@Justclown:~/work/arch-pc/lab06$ ls
in_out.asm lab6-1.asm
```

Рис. 4.3: Копирование файла

Открываю созданный файл `lab6-1.asm`, вставляю в него программу вывода значения регистра `eax` (рис. 4.4).

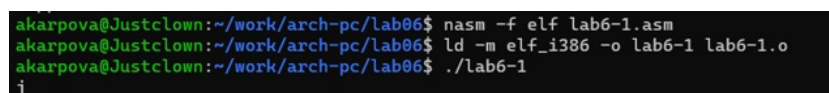




```
1 %include 'in_out.asm'
2
3 SECTION .bss
4 buf1:   RESB 80
5
6 SECTION .text
7 GLOBAL _start
8 _start:
9
10 mov eax, '6'
11 mov ebx, '4'
12 add eax, ebx
13 mov [buf1], eax
14 mov eax, buf1
15 call sprintf
16
17 call quit
```

Рис. 4.4: Ввод программы значения регистра eax

Создаю исполняемый файл программы и запускаю его. Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6. (рис. 4.5)



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

Рис. 4.5: Создание исполняемого файла и его запуск

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. 4.6)

```

1 %include 'in_out.asm'
2
3 SECTION .bss
4 buf1:    RESB 80
5
6 SECTION .text
7 GLOBAL _start
8 _start:
9
10 mov eax,6
11 mov ebx,4
12 add eax,ebx
13 mov [buf1], eax
14 mov eax,buf1
15 call sprintf
16
17 call quit|

```

Рис. 4.6: Изменение строки

Создаю новый исполняемый файл и запускаю его. Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран (рис. 4.7)

```

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

```

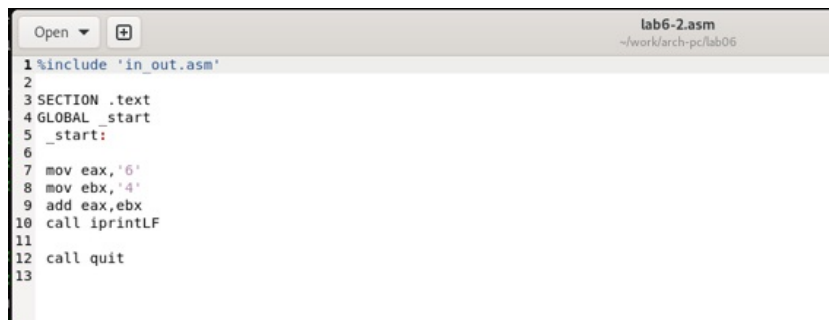
Рис. 4.7: Создание нового исполняемого файла и его запуск

С помощью touch создаю новый файл lab6-2.asm (рис. 4.8)

```
akarpova@Justclown:~/work/arch-pc/lab06$ touch lab6-2.asm
akarpova@Justclown:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1  lab6-1.asm  lab6-1.o  lab6-2.asm
```

Рис. 4.8: Создание исполняемого файла

Ввожу в файл текст другой программы для вывода значения регистра еах (рис. 4.9).



```
lab6-2.asm
~/work/arch-pc/lab06
1 %include 'in_out.asm'
2
3 SECTION .text
4 GLOBAL _start
5 _start:
6
7 mov eax, '6'
8 mov ebx, '4'
9 add eax, ebx
10 call iprintLF
11
12 call quit
13
```


Рис. 4.9: Ввод программы значения регистра еах

Создаю и запускаю исполняемый файл lab6-2. Теперь выводится число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4” (рис. 4.10)

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

Рис. 4.10: Создание и запуск исполняемого файла

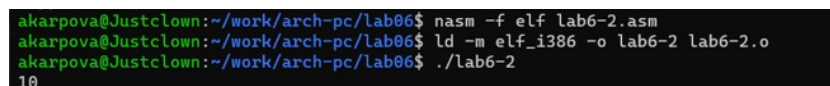
Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 4.11)



```
1 %include 'in_out.asm'
2
3 SECTION .text
4 GLOBAL _start
5 _start:
6
7 mov eax,6
8 mov ebx,4
9 add eax,ebx
10 call iprintLF
11
12 call quit
13
```

Рис. 4.11: Замена

Создаю и запускаю новый исполняемый файл. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10 (рис. 4.12)



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

Рис. 4.12: Создание и запуск исполняемого файла

Заменяю в тексте программы функцию iprintLF на iprint (рис. 4.13)



```
1 %include 'in_out.asm'
2
3 SECTION .text
4 GLOBAL _start
5 _start:
6
7 mov eax,6
8 mov ebx,4
9 add eax,ebx
10 call iprint
11
12 call quit
13
```

Рис. 4.13: Замена

Создаю и запускаю новый исполняемый файл. Вывод не изменяется, потому что символ переноса строки не отображался, когда программа исполнялась с функцией iprintLF, а iprint не добавляет к выводу символ переноса строки, в отличие от iprintLF (рис. 4.14)

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

Рис. 4.14: Создание и запуск исполняемого файла

## 4.2 Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm с помощью touch (рис. 4.15)

```
akarpova@Justclown:~/work/arch-pc/lab06$ touch lab6-3.asm
```

Рис. 4.15: Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения  $f(x) = (5 * 2 + 3)/3$  (рис. 4.16)

```
1 %include 'in_out.asm'
2
3 SECTION .data
4
5 div: DB 'Результат',0
6 rem: DB 'Остаток от деления: ',0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 ; ---- Вычисление выражения
13 mov eax,5
14 mov ebx,2
15 mul ebx
16 add eax,3
17 xor edx,edx
18 mov ebx,3
19 div ebx
20
21 mov edi,eax
22
23 ; ---- Вывод результата на экран
24
25 mov eax,div
26 call sprint
27 mov eax,edi
28 call iprintLF
29
30 mov eax,rem
31 call sprint
32 mov eax,edx
33 call iprintLF
34
35 call quit
```

Рис. 4.16: Ввод программы

Создаю исполняемый файл и запускаю его (рис. 4.17)

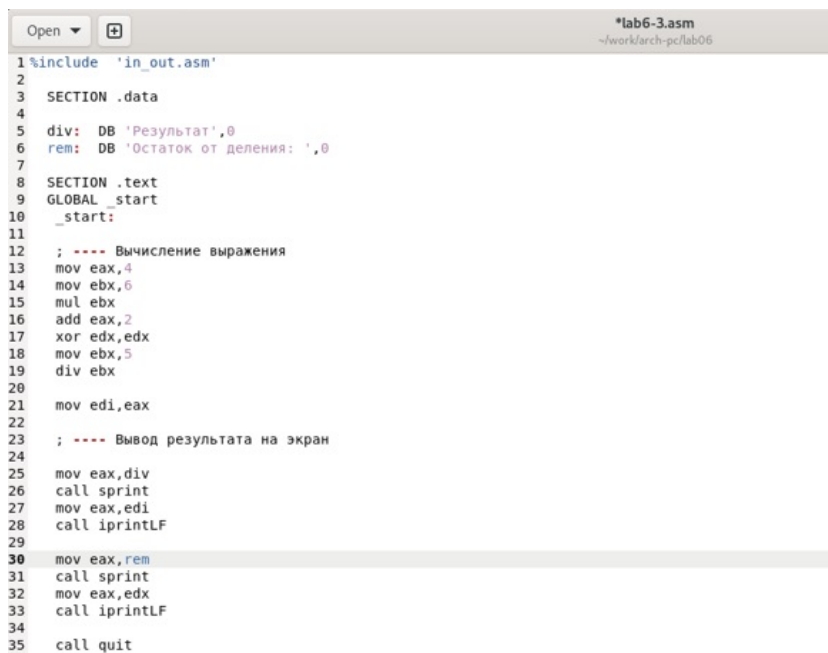
```

akarpova@JustcLown:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
akarpova@JustcLown:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
akarpova@JustcLown:~/work/arch-pc/lab06$ ./lab6-3
Результат4
Остаток от деления: 1

```

Рис. 4.17: Создание файла

Изменяю программу так, чтобы она вычисляла значение выражения  $f(x) = (4 * 6 + 2)/5$  (рис. 4.18)



```

1 %include 'in_out.asm'
2
3 SECTION .data
4
5 div: DB 'Результат',0
6 rem: DB 'Остаток от деления: ',0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 ; ---- Вычисление выражения
13 mov eax,4
14 mov ebx,6
15 mul ebx
16 add eax,2
17 xor edx,edx
18 mov ebx,5
19 div ebx
20
21 mov edi,eax
22
23 ; ---- Вывод результата на экран
24
25 mov eax,div
26 call sprint
27 mov eax,edi
28 call iprintLF
29
30 mov eax,rem
31 call sprint
32 mov eax,edx
33 call iprintLF
34
35 call quit

```

Рис. 4.18: Изменение программы

Создаю и запускаю новый исполняемый файл (рис. 4.19)

```

akarpova@JustcLown:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
akarpova@JustcLown:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
akarpova@JustcLown:~/work/arch-pc/lab06$ ./lab6-3
Результат5
Остаток от деления: 1

```

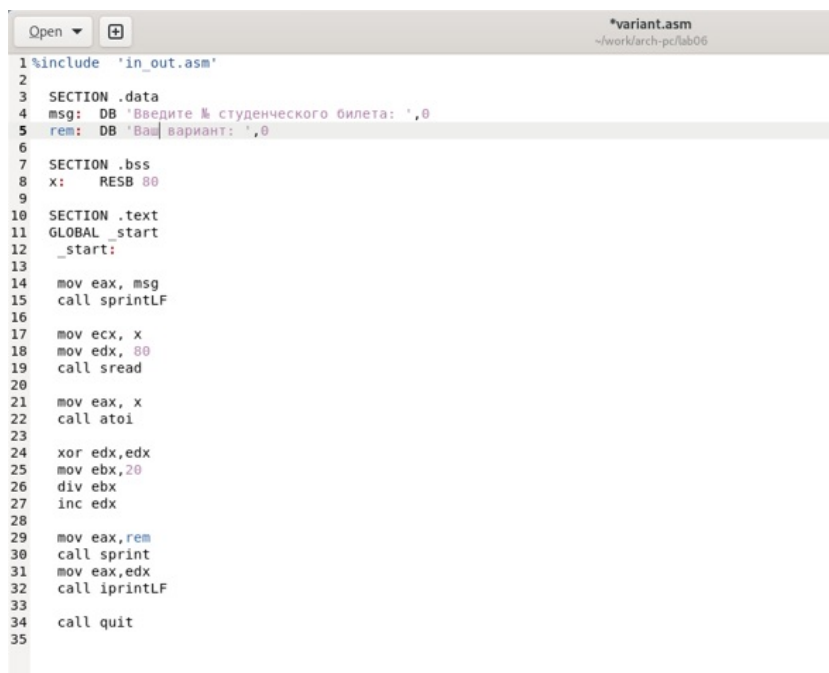
Рис. 4.19: Создание и запуск исполняемого файла

Создаю файл variant.asm с помощью утилиты touch (рис. 4.20)

```
akarpova@Justclown:~/work/arch-pc/lab06$ touch variant.asm
```

Рис. 4.20: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 4.21).



```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg: DB 'Введите № студенческого билета: ',0
5 rem: DB 'Ваш вариант: ',0
6
7 SECTION .bss
8 x: RESB 80
9
10 SECTION .text
11 GLOBAL _start
12 _start:
13
14 mov eax, msg
15 call sprintf
16
17 mov ecx, x
18 mov edx, 80
19 call sread
20
21 mov eax, x
22 call atoi
23
24 xor edx, edx
25 mov ebx, 20
26 div ebx
27 inc edx
28
29 mov eax, rem
30 call sprintf
31 mov eax, edx
32 call iprintf
33
34 call quit
35
```

Рис. 4.21: Ввод программы

Создаю и запускаю исполняемый файл. Ввожу номер своего студ. билета программа вывела, что мой вариант - 15 (рис. ??)

```
akarpova@Justclown:~/work/arch-pc/lab06$ nasm -f elf variant.asm
akarpova@Justclown:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
akarpova@Justclown:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132231934
Ваш вариант: 15
```

Рис. 4.22: Создание и запуск исполняемого файла

#### 4.2.1 Ответы на вопросы по программе

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

```
mov eax,edx 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. `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 отвечают строки:

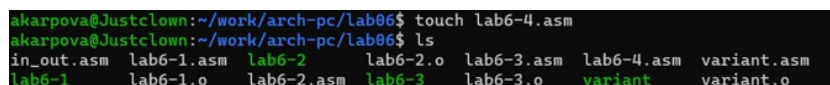
```
xor edx,edx mov ebx,20 ; ebx = 20 div ebx inc edx
```

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 отвечают след. строки:

```
mov eax,edx call iprintLF
```

#### 4.3 Выполнение заданий для самостоятельной работы

Создаю файл `lab6-4.asm` с помощью `touch` (рис. 4.23)

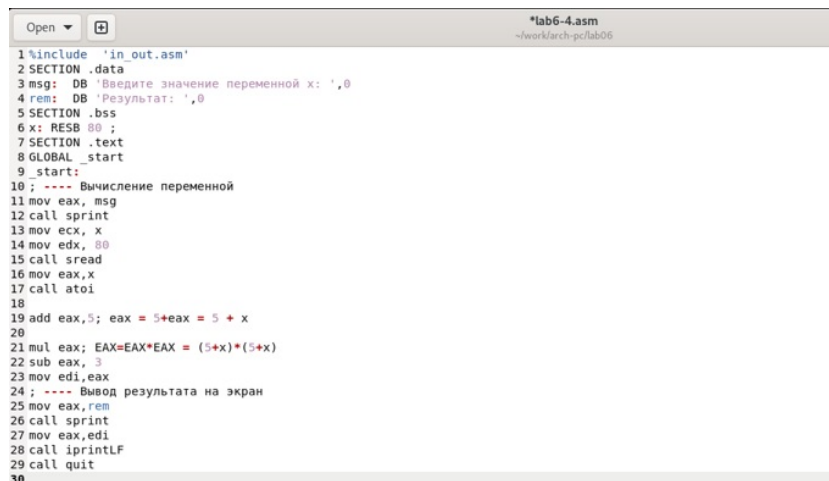


```
akarpova@Justclown:~/work/arch-pc/lab06$ touch lab6-4.asm
akarpova@Justclown:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1.asm  lab6-2      lab6-2.o  lab6-3.asm  lab6-4.asm  variant.asm
lab6-1      lab6-1.o    lab6-2.asm  lab6-3    lab6-3.o    variant     variant.o
```

Рис. 4.23: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения  $(5+x)^2-3$  (рис. 4.24)

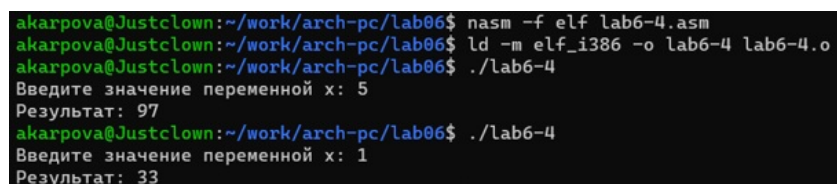




```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите значение переменной x: ',0
4 rem: DB 'Результат: ',0
5 SECTION .bss
6 x: RESB 80 ;
7 SECTION .text
8 GLOBAL _start
9 _start:
10 ; ---- Вычисление переменной
11 mov eax, msg
12 call sprint
13 mov ecx, x
14 mov edx, 80
15 call sread
16 mov eax, x
17 call atoi
18
19 add eax, 5; eax = 5+eax = 5 + x
20
21 mul eax; EAX=EAX*EAX = (5+x)*(5+x)
22 sub eax, 3
23 mov edi, eax
24 ; ---- Вывод результата на экран
25 mov eax, rem
26 call sprint
27 mov eax, edi
28 call iprintLF
29 call quit
30
```

Рис. 4.24: Редактирование

Создаю и запускаю исполняемый файл (рис. 4.25). При вводе значения 5, вывод - 97; 1 - 33.



```
akarpova@Justclown:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
akarpova@Justclown:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
akarpova@Justclown:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 5
Результат: 97
akarpova@Justclown:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 1
Результат: 33
```

Рис. 4.25: Создание и запуск исполняемого файла

## **5 Выводы**

При выполнении данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM

# **Список литературы**

## **1. Архитектура ЭВМ**