

Отчет по лабораторной работе №6

Арифметические операции в NASM

Симонова Полина Игоревна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
4.1	Символьные и численные данные в NASM	8
4.2	Выполнение арифметических операций в NASM	13
4.3	Вопросы по программе	16
4.4	Задание для самостоятельной работы	17
5	Выводы	20
6	Список литературы	21

Список иллюстраций

4.1	Создание директории	8
4.2	Создание копии файла	8
4.3	Редактирование файла	9
4.4	Запуск исполняемого файла	9
4.5	Редактирование файла	10
4.6	Запуск исполняемого файла	10
4.7	Создание файла	10
4.8	Редактирование файла	11
4.9	Запуск исполняемого файла	11
4.10	Редактирование файла	11
4.11	Запуск исполняемого файла	12
4.12	Редактирование файла	12
4.13	Запуск исполняемого файла	12
4.14	Создание файла	13
4.15	Редактирование файла	13
4.16	Запуск исполняемого файла	14
4.17	Изменение программы	14
4.18	Запуск исполняемого файла	14
4.19	Редактирование файла	15
4.20	Редактирование файла	15
4.21	Запуск исполняемого файла	15
4.22	Создание файла	17
4.23	Написание программы	17
4.24	Запуск исполняемого файла	18
4.25	Запуск исполняемого файла	18

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM

2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Задание 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.

- 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.

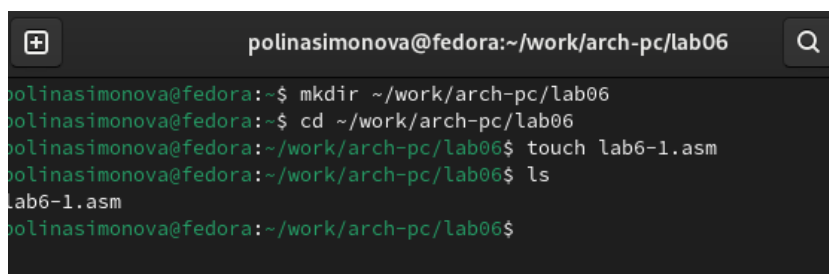
Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно 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. Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные будут представлять собой символы, что сделает невозможным получение корректного результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно.

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM

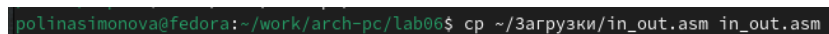
С помощью утилиты `mkdir` создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №6 (рис. 4.1). Перехожу в созданный каталог с помощью утилиты `cd`. С помощью утилиты `touch` создаю файл `lab6-1.asm` и с помощью утилиты `ls` проверяю, что файл действительно был создан (рис. 4.1).



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

Рис. 4.1: Создание директории

Копирую в текущий каталог файл `in_out.asm` с помощью утилиты `cp`, т.к. он будет использоваться в других программах (рис. 4.2).



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

Рис. 4.2: Создание копии файла

Открываю созданный файл `lab6-1.asm`, вставляю в него программу вывода значения регистра `eax` (рис. 4.3).


```

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

```

Рис. 4.3: Редактирование файла

Создаю исполняемый файл программы и запускаю его (рис. 4.4). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

```

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

```

Рис. 4.4: Запуск исполняемого файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. 4.5).

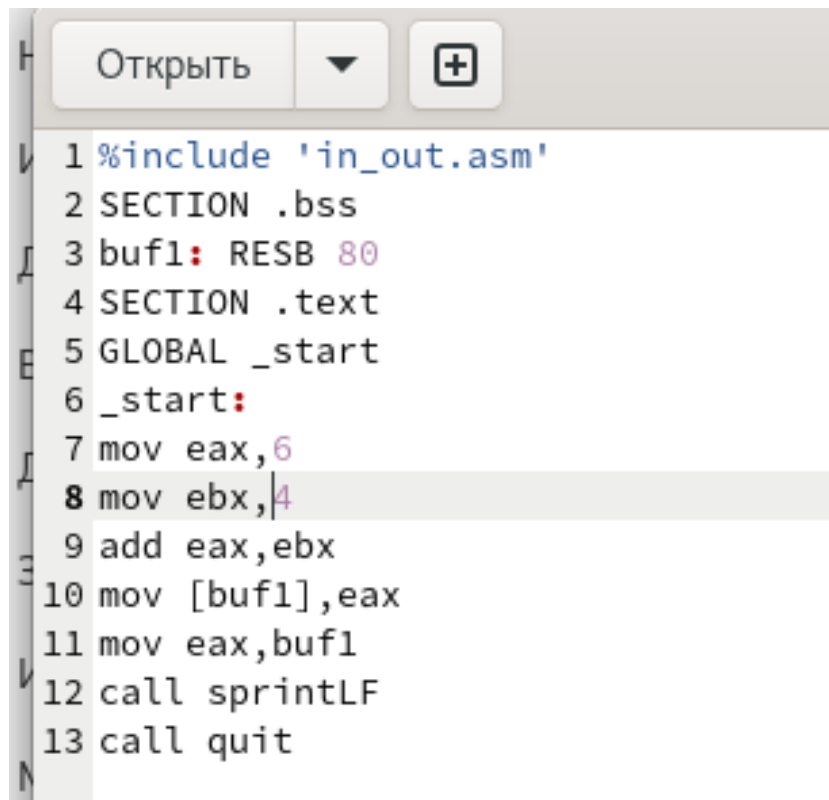


Рис. 4.5: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. 4.6). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

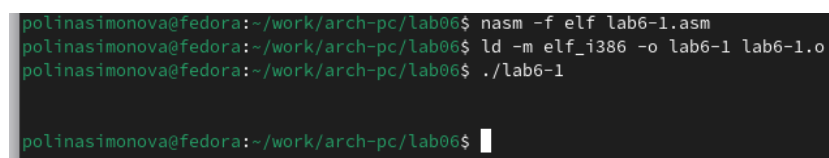


Рис. 4.6: Запуск исполняемого файла

Создаю новый файл lab6-2.asm с помощью утилиты touch и проверяю с помощью ls, что файл был создан (рис. 4.7).

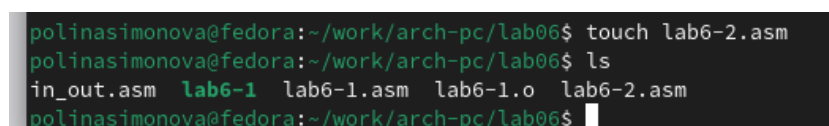


Рис. 4.7: Создание файла

Ввожу в файл текст другой программы для вывода значения регистра eax (рис. 4.7).

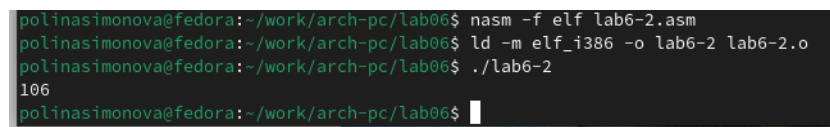


```
lab6-2.asm
~/work/arch-pc/lab06

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

Рис. 4.8: Редактирование файла

Создаю и запускаю исполняемый файл lab6-2 (рис. 4.9). Теперь вывод число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”.



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

Рис. 4.9: Запуск исполняемого файла

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 4.10).



```
lab6-2.asm
~/work/arch-pc/lab06

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

Рис. 4.10: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 4.11).. Теперь программа

складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

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

Рис. 4.11: Запуск исполняемого файла

Заменяю в тексте программы функцию `iprintLF` на `iprint` (рис. 4.12).

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

Рис. 4.12: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 4.13). Вывод не изменился, 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`.

```
polinasimonova@fedora:~/work/arch-pc/lab06$ ./lab6-2
10polinasimonova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.13: Запуск исполняемого файла

4.2 Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm с помощью утилиты touch (рис. 4.14).

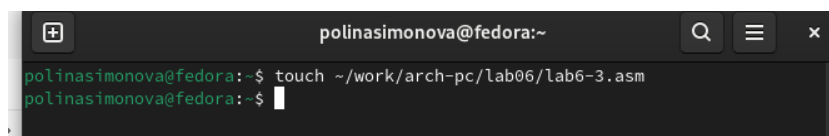


Рис. 4.14: Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$ (рис. 4.15).

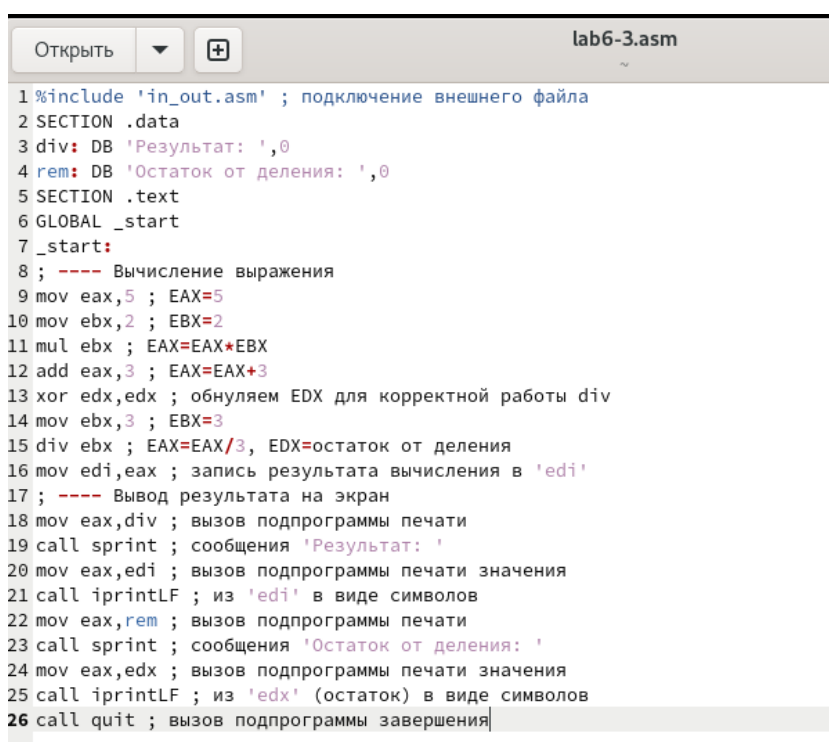


Рис. 4.15: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 4.16).

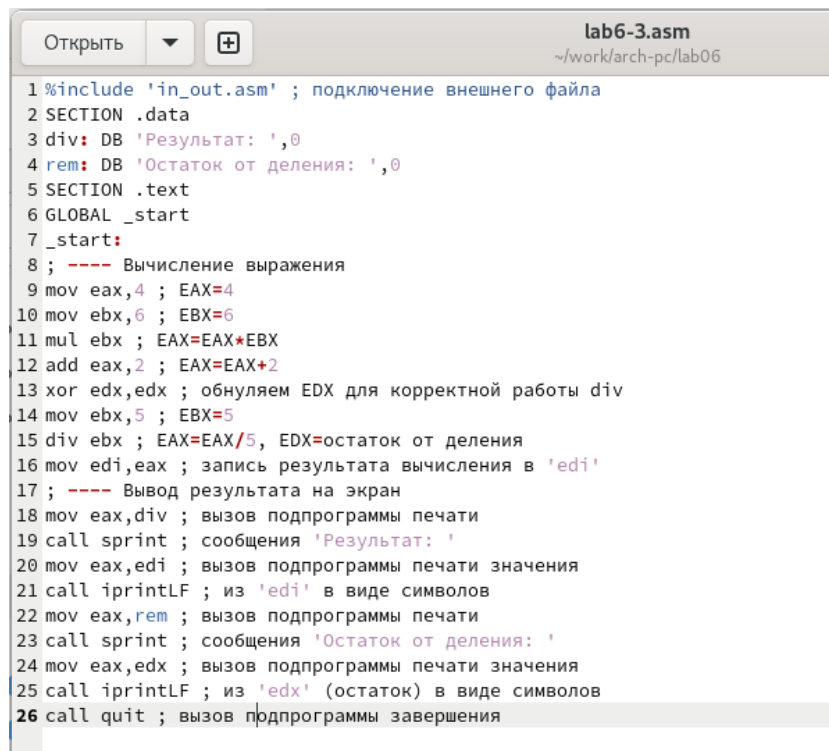
```

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

```

Рис. 4.16: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$ (рис. 4.17).



```

lab6-3.asm
~/work/arch-pc/lab06

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8 ; ---- Вычисление выражения
9 mov eax,4 ; EAX=4
10 mov ebx,6 ; EBX=6
11 mul ebx ; EAX=EAX*EBX
12 add eax,2 ; EAX=EAX+2
13 xor edx,edx ; обнуляем EDX для корректной работы div
14 mov ebx,5 ; EBX=5
15 div ebx ; EAX=EAX/5, EDX=остаток от деления
16 mov edi,eax ; запись результата вычисления в 'edi'
17 ; ---- Вывод результата на экран
18 mov eax,div ; вызов подпрограммы печати
19 call sprint ; сообщения 'Результат: '
20 mov eax,edi ; вызов подпрограммы печати значения
21 call iprintLF ; из 'edi' в виде символов
22 mov eax,rem ; вызов подпрограммы печати
23 call sprint ; сообщения 'Остаток от деления: '
24 mov eax,edx ; вызов подпрограммы печати значения
25 call iprintLF ; из 'edx' (остаток) в виде символов
26 call quit ; вызов подпрограммы завершения

```

Рис. 4.17: Изменение программы

Создаю и запускаю новый исполняемый файл (рис. 4.18). Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.

```

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

```

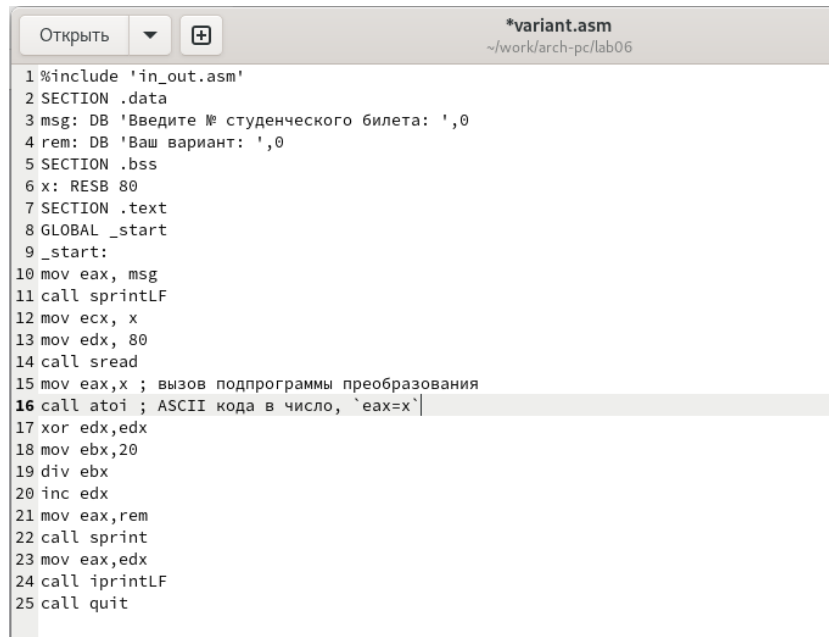
Рис. 4.18: Запуск исполняемого файла

Создаю файл variant.asm с помощью утилиты touch (рис. 4.19).

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

Рис. 4.19: Редактирование файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 4.20).



```
*variant.asm
~/work/arch-pc/lab06

1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите № студенческого билета: ',0
4 rem: DB 'Ваш вариант: ',0
5 SECTION .bss
6 x: RESB 80
7 SECTION .text
8 GLOBAL _start
9 _start:
10 mov eax, msg
11 call sprintf
12 mov ecx, x
13 mov edx, 80
14 call sread
15 mov eax, x ; вызов подпрограммы преобразования
16 call atoi ; ASCII кода в число, 'eax=x'
17 xor edx, edx
18 mov ebx, 20
19 div ebx
20 inc edx
21 mov eax, rem
22 call sprintf
23 mov eax, edx
24 call iprintLF
25 call quit
```

Рис. 4.20: Редактирование файла

Создаю и запускаю исполняемый файл (рис. 4.21). Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 7.

```
polinasimonova@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
polinasimonova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
polinasimonova@fedora:~/work/arch-pc/lab06$ ./variant
bash: ./: Это каталог
polinasimonova@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
11322246738
Ваш вариант: 7
polinasimonova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.21: Запуск исполняемого файла

4.3 Вопросы по программе

1. За вывод сообщения “Ваш вариант” 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
```

это запись в регистр edx длины вводимой строки

```
call sread
```

это 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 ; 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
```

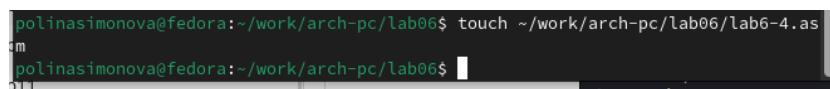
5. При выполнении инструкции div ebx остаток от деления записывается в регистр edx

6. Инструкция `inc edx` увеличивает значение регистра `edx` на 1
7. За вывод на экран результатов вычислений отвечают строки:

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

4.4 Задание для самостоятельной работы

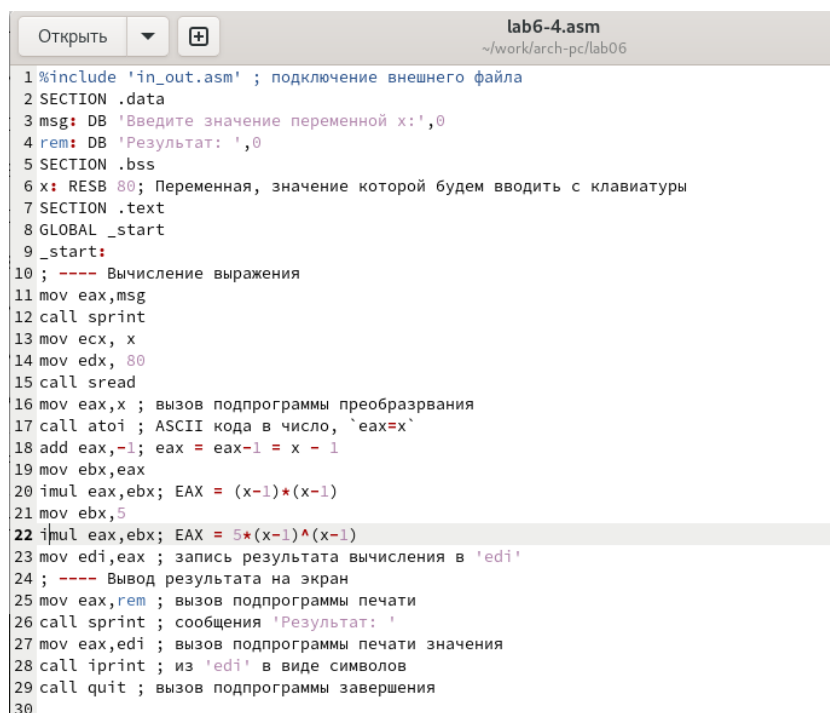
Создаю файл `lab6-4.asm` с помощью утилиты `touch` (рис. 4.22).



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

Рис. 4.22: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения $5(x - 1)^2$ (рис. 4.23). Это выражение было под вариантом 7.



```
lab6-4.asm  
~/work/arch-pc/lab06  
Открыть  
1 %include 'in_out.asm' ; подключение внешнего файла  
2 SECTION .data  
3 msg: DB 'Введите значение переменной x:',0  
4 rem: DB 'Результат: ',0  
5 SECTION .bss  
6 x: RESB 80; Переменная, значение которой будем вводить с клавиатуры  
7 SECTION .text  
8 GLOBAL _start  
9 _start:  
10 ; ---- Вычисление выражения  
11 mov eax,msg  
12 call sprint  
13 mov ecx, x  
14 mov edx, 80  
15 call sread  
16 mov eax,x ; вызов подпрограммы преобразования  
17 call atoi ; ASCII кода в число, `eax=x`  
18 add eax,-1; eax = eax-1 = x - 1  
19 mov ebx,eax  
20 imul eax,ebx; EAX = (x-1)*(x-1)  
21 mov ebx,5  
22 imul eax,ebx; EAX = 5*(x-1)*(x-1)  
23 mov edi,eax ; запись результата вычисления в 'edi'  
24 ; ---- Вывод результата на экран  
25 mov eax,rem ; вызов подпрограммы печати  
26 call sprint ; сообщения 'Результат: '  
27 mov eax,edi ; вызов подпрограммы печати значения  
28 call iprint ; из 'edi' в виде символов  
29 call quit ; вызов подпрограммы завершения  
30
```

Рис. 4.23: Написание программы

Создаю и запускаю исполняемый файл (рис. 4.24). При вводе значения 3, вывод - 20.

```
polinasimonova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
polinasimonova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
polinasimonova@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x:3
Результат: 20polinasimonova@fedora:~/work/arch-pc/lab06$ ./lab6-4
```

Рис. 4.24: Запуск исполняемого файла

Провожу еще один запуск исполняемого файла для проверки работы программы с другим значением на входе - 5 (рис. 4.25). Программа отработала верно и в ответе мы получили 80.

```
Результат: 20polinasimonova@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x:5
Результат: 80
polinasimonova@fedora:~/work/arch-pc/lab06$
```

Рис. 4.25: Запуск исполняемого файла

Листинг 4.1. Программа для вычисления значения выражения $5(x - 1)^2$.

```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg: DB 'Введите значение переменной x:',0
rem: DB 'Результат: ',0
SECTION .bss
x: RESB 80; Переменная, значение которой будем вводить с клавиатуры
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax,msg
call sprint
mov ecx, x
mov edx, 80
```

```

call sread
mov eax,x ; вызов подпрограммы преобразрвания
call atoi ; ASCII кода в число, 'eax=x'
add eax,-1; eax = eax-1 = x - 1
mov ebx,eax
imul eax,ebx; EAX = (x-1)*(x-1)
mov ebx,5
imul eax,ebx; EAX = 5*(x-1)^(x-1)
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
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

```

5 Выводы

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

6 Список литературы

Лабораторная работа 6