

Отчёт по лабораторной работе №6

Дисциплина: Архитектура компьютера

Зарицкая Марина Петровна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Символьные и численные данные в NASM	9
4.2	Выполнение арифметических операций в NASM	14
4.3	Выполнение заданий для самостоятельной работы	19
5	Выводы	23
	Список литературы	24

Список иллюстраций

4.1	Создание каталога и файла программы	9
4.2	Программа из листинга 6.1	10
4.3	Результат работы программы	10
4.4	Измененный текст программы	11
4.5	Результат работы измененной программы	11
4.6	Текст программы из листинга 6.2	12
4.7	Запуск второй программы	12
4.8	Измененный текст второй программы	13
4.9	Результат работы программы	13
4.10	Изменения в тексте второй программы	14
4.11	Вывод результата программы без переноса строки	14
4.12	Создание файла третьей программы	15
4.13	Текст программы из листинга 6.3	15
4.14	Результат работы третьей программы	16
4.15	Измененный текст программы	16
4.16	Вывод измененной программы	17
4.17	Создание файла	17
4.18	Текст программы из листинга 6.4	17
4.19	Результат работы программы для вычисления варианта	18
4.20	Текст программы для вычисления значения выражения	20
4.21	Результат работы программы для вычисления значения выражения	20

Список таблиц

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. 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.

- 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

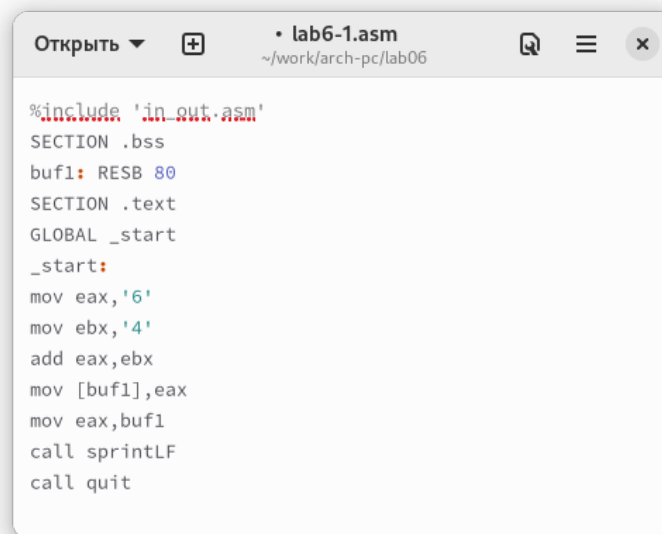
4.1 Символьные и численные данные в NASM

Создала каталог для программ лабораторной работы №6, перешла в него и создала файл lab6-1.asm (рис. 4.1).

```
[mpzarickaya@fedora lab07]$ mkdir ~/work/arch-pc/lab06  
[mpzarickaya@fedora lab07]$ cd ~/work/arch-pc/lab06  
[mpzarickaya@fedora lab06]$ touch lab6-1.asm  
[mpzarickaya@fedora lab06]$ ls  
lab6-1.asm
```

Рис. 4.1: Создание каталога и файла программы

Ввела в файл lab6-1.asm текст программы из листинга 6.1 (рис. 4.2).

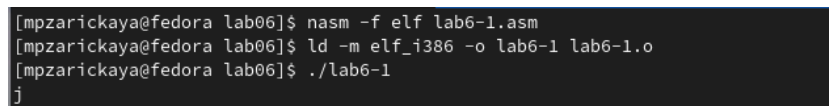


```
Открыть ▾ + • lab6-1.asm
~/work/arch-pc/lab06

%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. 4.2: Программа из листинга 6.1

Создала исполняемый файл и запустила его. Программа вывела символ j (рис. 4.3).



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

Рис. 4.3: Результат работы программы

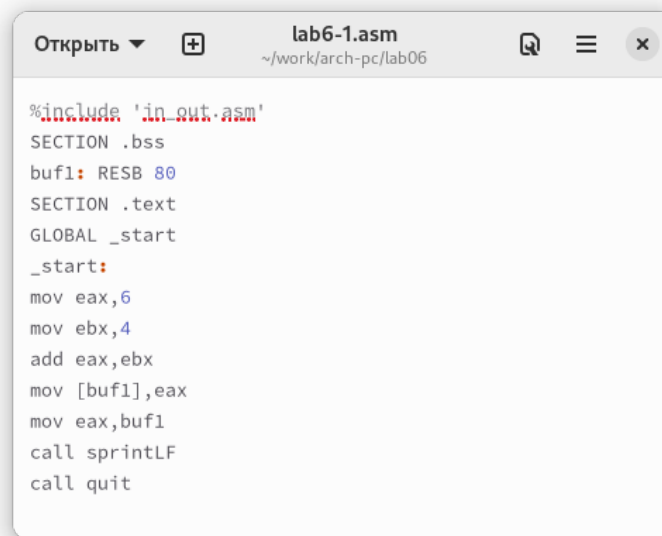
Изменила текст программы (Листинг 6.1) следующим образом: заменила строки

```
mov eax, '6'
mov ebx, '4'
```

на строки

```
mov eax, 6
mov ebx, 4
```

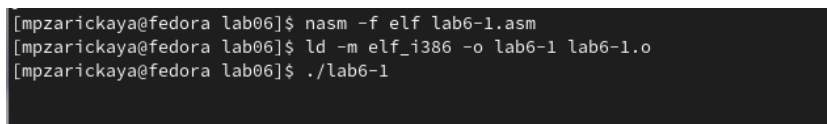
(рис. 4.4).

A screenshot of a text editor window titled 'lab6-1.asm' with a path '~/.work/arch-pc/lab06'. The code is as follows:

```
%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. 4.4: Измененный текст программы

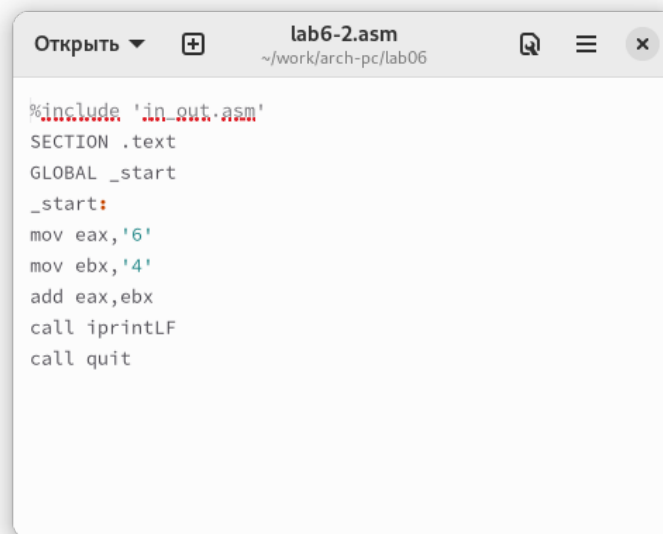
Создала исполняемый файл и запустила его. Результат программы - символ с кодом 10, это символ перевода строки и он не отображается при выводе на экран (рис. 4.5).

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

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

Рис. 4.5: Результат работы измененной программы

Затем я создала файл lab6-2.asm в каталоге ~/.work/arch-pc/lab06 и ввела в него текст программы из листинга 6.2 (рис. 4.6).

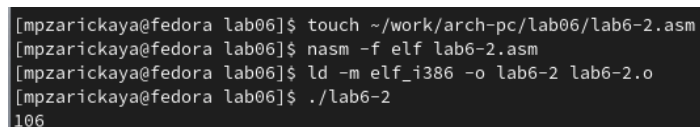


```
Открыть ▾ + lab6-2.asm
~/work/arch-pc/lab06

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

Рис. 4.6: Текст программы из листинга 6.2

Запустила созданный исполняемый файл (рис. 4.7).



```
[mpzarickaya@fedora lab06]$ touch ~/work/arch-pc/lab06/lab6-2.asm
[mpzarickaya@fedora lab06]$ nasm -f elf lab6-2.asm
[mpzarickaya@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[mpzarickaya@fedora lab06]$ ./lab6-2
106
```

Рис. 4.7: Запуск второй программы

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 заменила строки программы

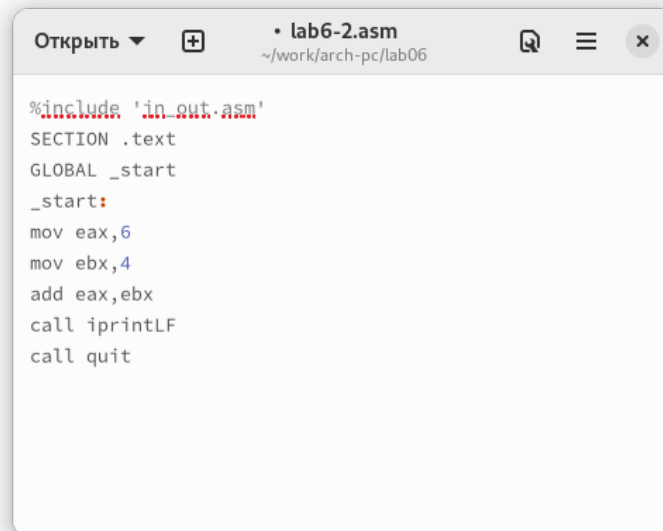
```
mov eax, '6'
mov ebx, '4'
```

на строки

```
mov eax,6
```

```
mov ebx,4
```

(рис. 4.8).



```
Открыть ▾ + • lab6-2.asm ~/work/arch-pc/lab06

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

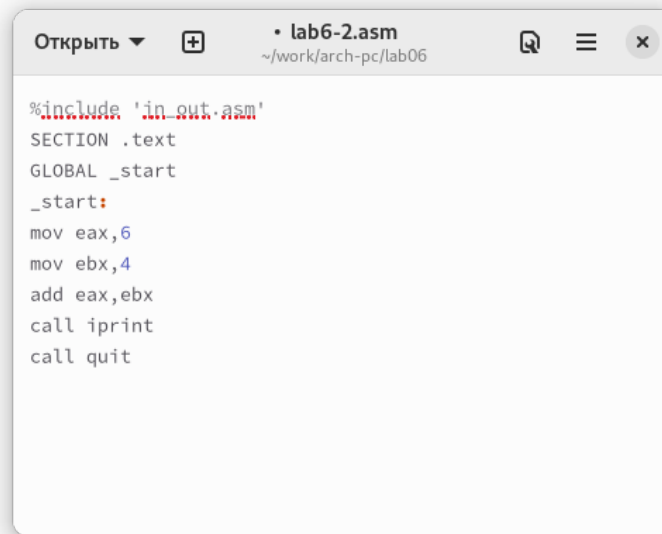
Рис. 4.8: Измененный текст второй программы

Создала и запустила новый исполняемый файл. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10 (рис. 4.9).

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

Рис. 4.9: Результат работы программы

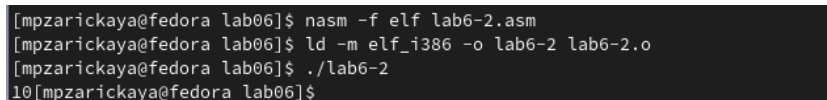
Изменила функцию iprintLF на iprint в тексте программы (рис. 4.10).



```
Открыть ▾ + • lab6-2.asm  
~/work/arch-pc/lab06  
  
%include 'in_out.asm'  
SECTION .text  
GLOBAL _start  
_start:  
mov eax,6  
mov ebx,4  
add eax,ebx  
call iprint  
call quit
```

Рис. 4.10: Изменения в тексте второй программы

Затем создала и запустила исполняемый файл. В результате работы программы так же выводится число 10, но уже без переноса строки, за который отвечает функция `iprintLF` (рис. 4.11).



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

Рис. 4.11: Вывод результата программы без переноса строки

4.2 Выполнение арифметических операций в NASM

Создала файл `lab6-3.asm` в каталоге `~/work/arch-pc/lab06` (рис. 4.12).

```
[mpzarickaya@fedora lab06]$ touch ~/work/arch-pc/lab06/lab6-3.asm
[mpzarickaya@fedora lab06]$ ls
in_out.asm  lab6-1.asm  lab6-2      lab6-2.o    variant.asm
lab6-1      lab6-1.o    lab6-2.asm  lab6-3.asm
```

Рис. 4.12: Создание файла третьей программы

Ввела текст программы из листинга 6.3 в файл lab6-3.asm (рис. 4.13).

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

%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
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
```

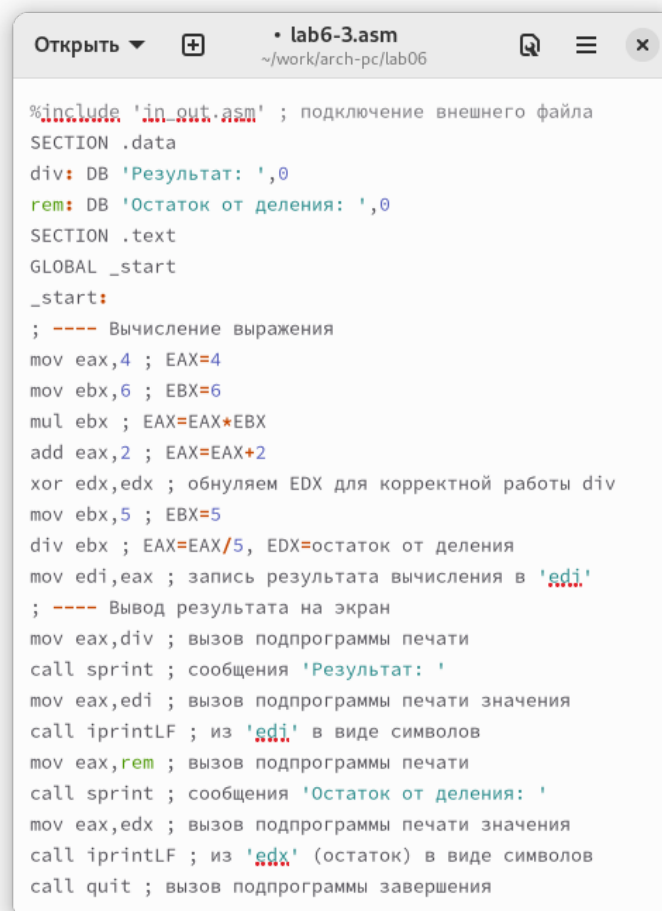
Рис. 4.13: Текст программы из листинга 6.3

Создала и запустила исполняемый файл (рис. 4.14).

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

Рис. 4.14: Результат работы третьей программы

Изменила программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$ (рис. 4.15).



```
Открыть ▾ + • lab6-3.asm ~/work/arch-pc/lab06
%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=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'
; ---- 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
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
```

Рис. 4.15: Измененный текст программы

Создала и запустила новый исполняемый файл (рис. 4.16).


```
[mpzarickaya@fedora lab06]$ nasm -f elf lab6-3.asm
[mpzarickaya@fedora lab06]$ ld -m elf_i386 -o lab6-3 lab6-3.o
[mpzarickaya@fedora lab06]$ ./lab6-3
Результат: 5
Остаток от деления: 1
```

Рис. 4.16: Вывод измененной программы

Создала файл `variant.asm` с помощью утилиты `touch` (рис. 4.17).

```
[mpzarickaya@fedora lab06]$ touch ~/work/arch-pc/lab06/variant.asm
[mpzarickaya@fedora lab06]$ ls
in_out.asm  lab6-1.asm  lab6-2      lab6-2.o  lab6-3.asm  variant.asm
lab6-1      lab6-1.o    lab6-2.asm  lab6-3    lab6-3.o
```

Рис. 4.17: Создание файла

Ввела в файл текст программы из листинга 6.4 (рис. 4.18).

```
variant.asm
~/work/arch-pc...
Сохранить

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 sprint
23 mov eax,edx
24 call iprintLF
25 call quit

Matlab  Ширина табуляции: 8  Ln 25, Col 10  INS
```

Рис. 4.18: Текст программы из листинга 6.4

Создала исполняемый файл и запустила программу. Ввела номер своего студенческого билета и получила номер своего варианта - 7 (рис. 4.19). Проверила правильность выполнения, вычислив номер варианта аналитически.

```
[mpzarickaya@fedora lab06]$ nasm -f elf variant.asm
[mpzarickaya@fedora lab06]$ ld -m elf_i386 -o variant variant.o
[mpzarickaya@fedora lab06]$ ./variant
Введите № студенческого билета:
1132236026
Ваш вариант: 7
```

Рис. 4.19: Результат работы программы для вычисления варианта

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` - 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 ; 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
```

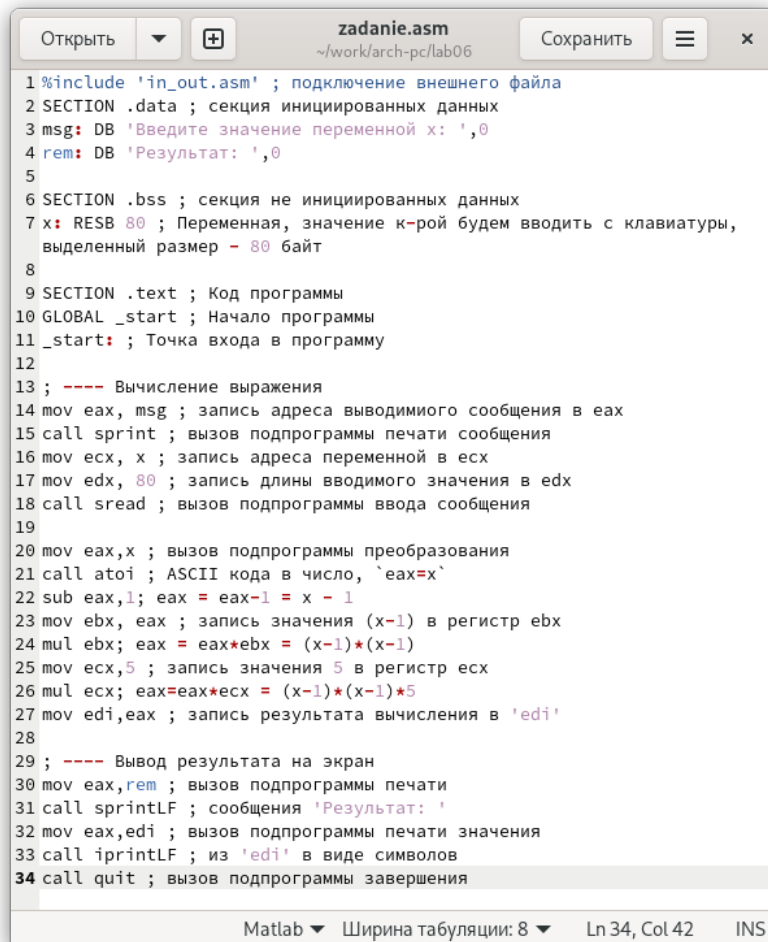
5. При выполнении инструкции `div ebx` остаток от деления записывается в регистр `edx`.

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

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

4.3 Выполнение заданий для самостоятельной работы

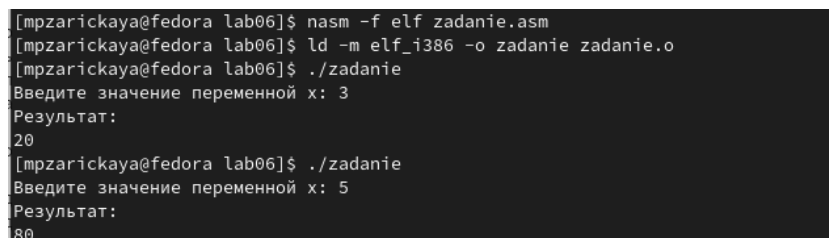
Создала файл `zadanie.asm`. Открыла его для редактирования, ввела в него текст программы для вычисления значения выражения $5(x - 1)^2$ в соответствии со своим вариантом (рис. 4.20).



```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data ; секция инициализированных данных
3 msg: DB 'Введите значение переменной x: ',0
4 rem: DB 'Результат: ',0
5
6 SECTION .bss ; секция не инициализированных данных
7 x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры,
   выделенный размер - 80 байт
8
9 SECTION .text ; Код программы
10 GLOBAL _start ; Начало программы
11 _start: ; Точка входа в программу
12
13 ; ---- Вычисление выражения
14 mov eax, msg ; запись адреса выводимого сообщения в eax
15 call sprint ; вызов подпрограммы печати сообщения
16 mov ecx, x ; запись адреса переменной в ecx
17 mov edx, 80 ; запись длины вводимого значения в edx
18 call sread ; вызов подпрограммы ввода сообщения
19
20 mov eax,x ; вызов подпрограммы преобразования
21 call atoi ; ASCII кода в число, `eax=x`
22 sub eax,1; eax = eax-1 = x - 1
23 mov ebx, eax ; запись значения (x-1) в регистр ebx
24 mul ebx; eax = eax*ebx = (x-1)*(x-1)
25 mov ecx,5 ; запись значения 5 в регистр ecx
26 mul ecx; eax=eax*ecx = (x-1)*(x-1)*5
27 mov edi,eax ; запись результата вычисления в 'edi'
28
29 ; ---- Вывод результата на экран
30 mov eax,rem ; вызов подпрограммы печати
31 call sprintLF ; сообщения 'Результат: '
32 mov eax,edi ; вызов подпрограммы печати значения
33 call iprintLF ; из 'edi' в виде символов
34 call quit ; вызов подпрограммы завершения
```

Рис. 4.20: Текст программы для вычисления значения выражения

Создала и запустила исполняемый файл. При вводе значения 3 на входе вывод программы - 20. При вводе значения 5 программа выводит результат 80. Программа работает верно (рис. 4.21).



```
[mpzarickaya@fedora lab06]$ nasm -f elf zadanie.asm
[mpzarickaya@fedora lab06]$ ld -m elf_i386 -o zadanie zadanie.o
[mpzarickaya@fedora lab06]$ ./zadanie
Введите значение переменной x: 3
Результат:
20
[mpzarickaya@fedora lab06]$ ./zadanie
Введите значение переменной x: 5
Результат:
80
```

Рис. 4.21: Результат работы программы для вычисления значения выражения

Листинг программы для вычисления значения выражения $5(x - 1)^2$ (вариант 7)

```
%include 'in_out.asm' ; подключение внешнего файла
```

```
SECTION .data ; секция инициированных данных
```

```
msg: DB 'Введите значение переменной x: ',0
```

```
rem: DB 'Результат: ',0
```

```
SECTION .bss ; секция не инициированных данных
```

```
x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры, выделенный ра
```

```
SECTION .text ; Код программы
```

```
GLOBAL _start ; Начало программы
```

```
_start: ; Точка входа в программу
```

```
; ---- Вычисление выражения
```

```
mov eax, msg ; запись адреса выводимого сообщения в eax
```

```
call sprint ; вызов подпрограммы печати сообщения
```

```
mov ecx, x ; запись адреса переменной в ecx
```

```
mov edx, 80 ; запись длины вводимого значения в edx
```

```
call sread ; вызов подпрограммы ввода сообщения
```

```
mov eax,x ; вызов подпрограммы преобразования
```

```
call atoi ; ASCII кода в число, `eax=x`
```

```
sub eax,1; eax = eax-1 = x - 1
```

```
mov ebx, eax ; запись значения (x-1) в регистр ebx
```

```
mul ebx; eax = eax*ebx = (x-1)*(x-1)
```

```
mov ecx,5 ; запись значения 5 в регистр ecx
```

```
mul ecx; eax=eax*ecx = (x-1)*(x-1)*5
```

```
mov edi,eax ; запись результата вычисления в 'edi'
```

```
; ---- Вывод результата на экран
mov eax,rem ; вызов подпрограммы печати
call sprintLF ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения
```

5 Выводы

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

Список литературы

(<https://esystem.rudn.ru/mod/resource/view.php?id=1030554>)