РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

дисциплина: Архитектура компьютера

Студент: Валиева Карина Ринатовна

Группа: НБИбд-01-25

МОСКВА

2025 г.

Оглавление

1 Цель работы

Цель данной лабораторной работы - освоение арифметческих 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

С помощью утилиты mkdir создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №6. Перехожу в созданный каталог с помощью утилиты cd. С помощью утилиты touch создаю файл lab6-1.asm (Рис.1.)

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

Рис. 1. Создание файла

Вручную копирую в текущий каталог файл in_out.asm, т.к. он будет использоваться в других программах.(Рис.2.)

Рис. 2. Копирование файла in_out.asm

Открываю созданный файл lab6-1.asm, вставляю в него программу вывода значения регистра eax. (Рис.3.).

Рис. 3. Редактирование файла

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.)

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

Рис. 4. Запуск исполняемого файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4

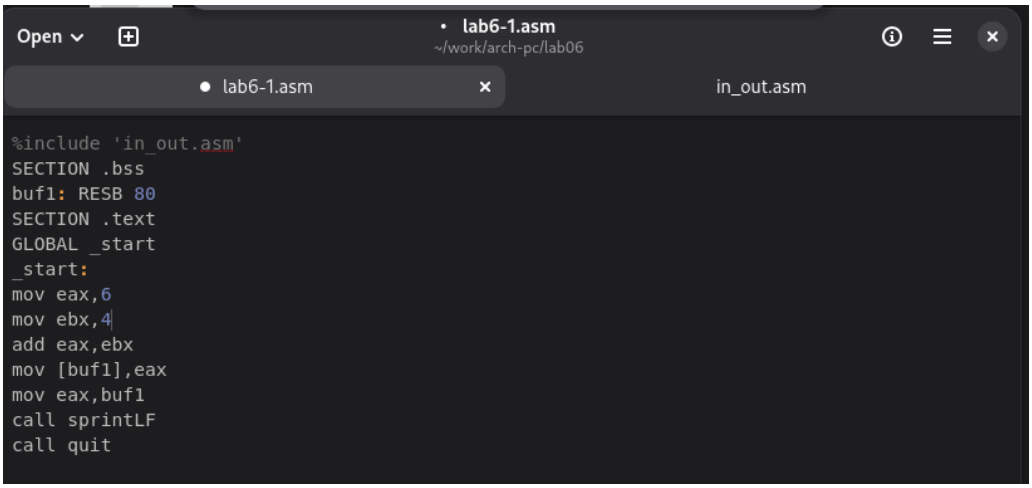

Рис. 5. Редактирование файла

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

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


krvalieva@vbox:~/work/arch-pc/lab06$
```

Рис. 6. Запуск исполняемого файла

Создаю новый файл lab6-2.asm с помощью утилиты touch и проверяю (Рис.7.)

```
krvalieva@vbox:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-2.asm
krvalieva@vbox:~/work/arch-pc/lab06$ ls
in_out.asm  lab5-2.asm  lab6-1  lab6-1.asm  lab6-1.o  lab6-2.asm
krvalieva@vbox:~/work/arch-pc/lab06$
```

Рис. 7. Создание файла

Ввожу в файл текст другойпрограммы для вывода значения регистра eax. (Рис.8.)

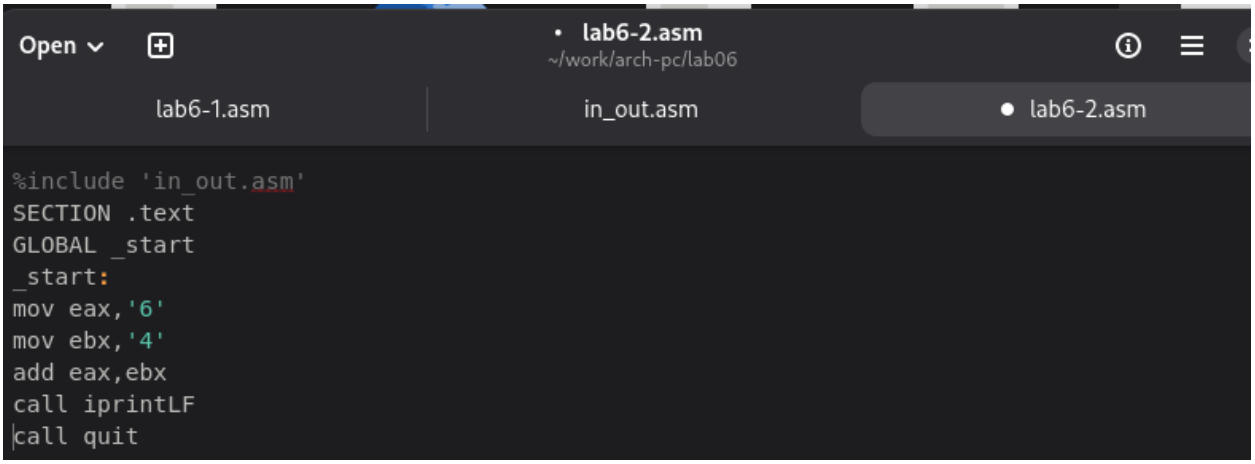

Рис. 8. Редактирование файла

Создаю и запускаю исполняемый файл lab6-2. 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”. (Рис.9.)

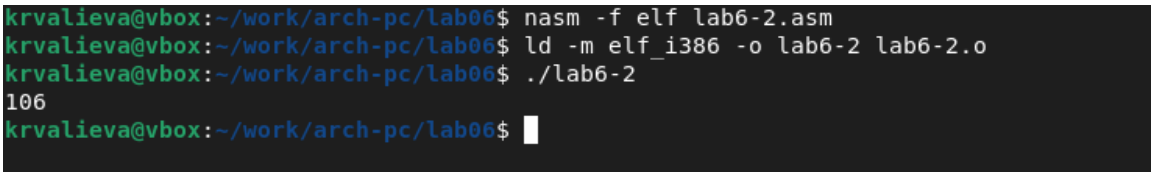

Рис. 9. Запуск исполняемого файла

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4. (Рис.10.)

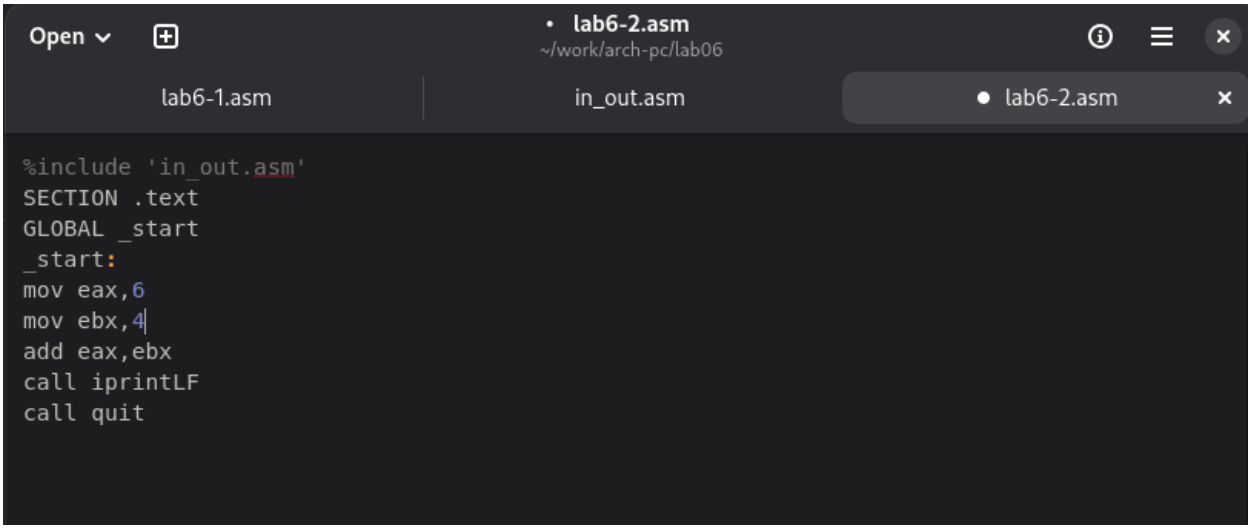

Рис. 10. Редактирование файла

Создаю и запускаю новый исполняемый файл. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, 11 поэтому вывод 10. (Рис.11.)

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

Рис. 11. Запуск исполняемого файла

Заменяю в тексте программы функцию iprintLF на iprint. (Рис.12.)

Open • lab6-2.asm ~/work/arch-pc/lab06

lab6-1.asm | in_out.asm | • lab6-2.asm

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

Рис. 12. Редактирование файла

Создаю и запускаю новый исполняемый файл. 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. (Рис.13.)

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

Рис. 13. Запуск исполняемого файла

4.2 Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm с помощью утилиты touch. (Рис.14.)

```
10krvalieva@vbox:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-3.asm
krvalieva@vbox:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1  lab6-1.asm  lab6-1.o  lab6-2  lab6-2.asm  lab6-2.o  lab6-3.asm
krvalieva@vbox:~/work/arch-pc/lab06$
```

Рис. 14. Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения f(x) = (5 * 2 + 3)/3. (Рис.15.)

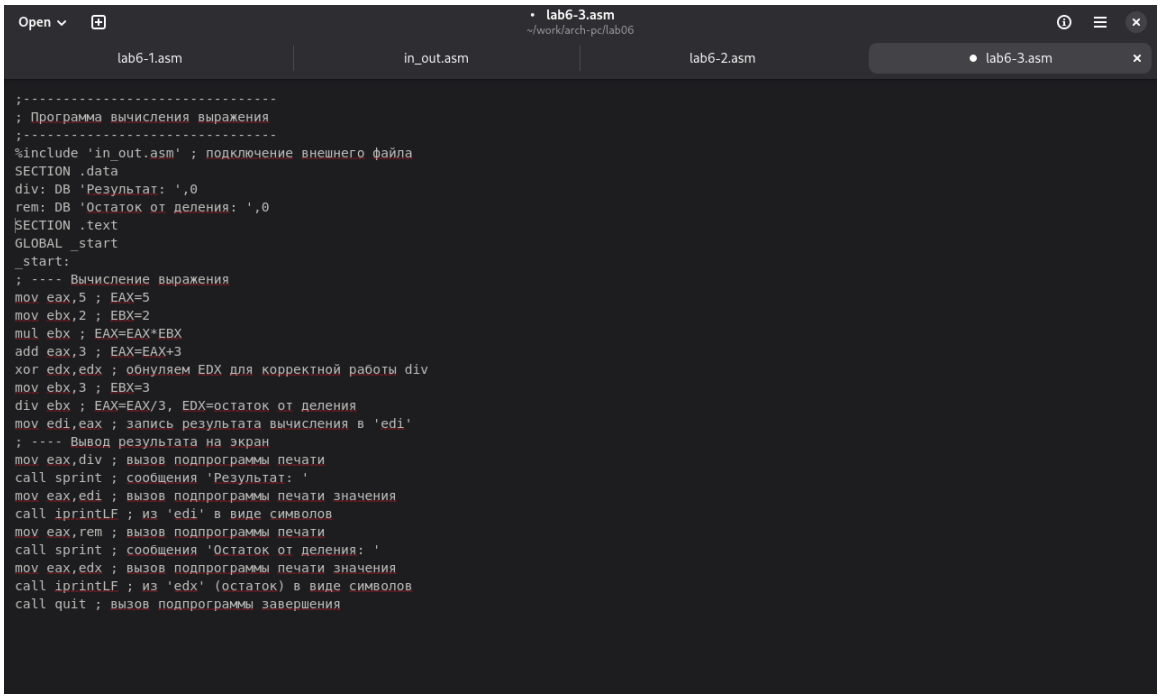

Рис. 15. Редактирование файла

Создаю исполняемый файл и запускаю его. (Рис.16.)

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

Рис. 16. Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения f(x) = (4 * 6 + 2)/5. (Рис.17.)

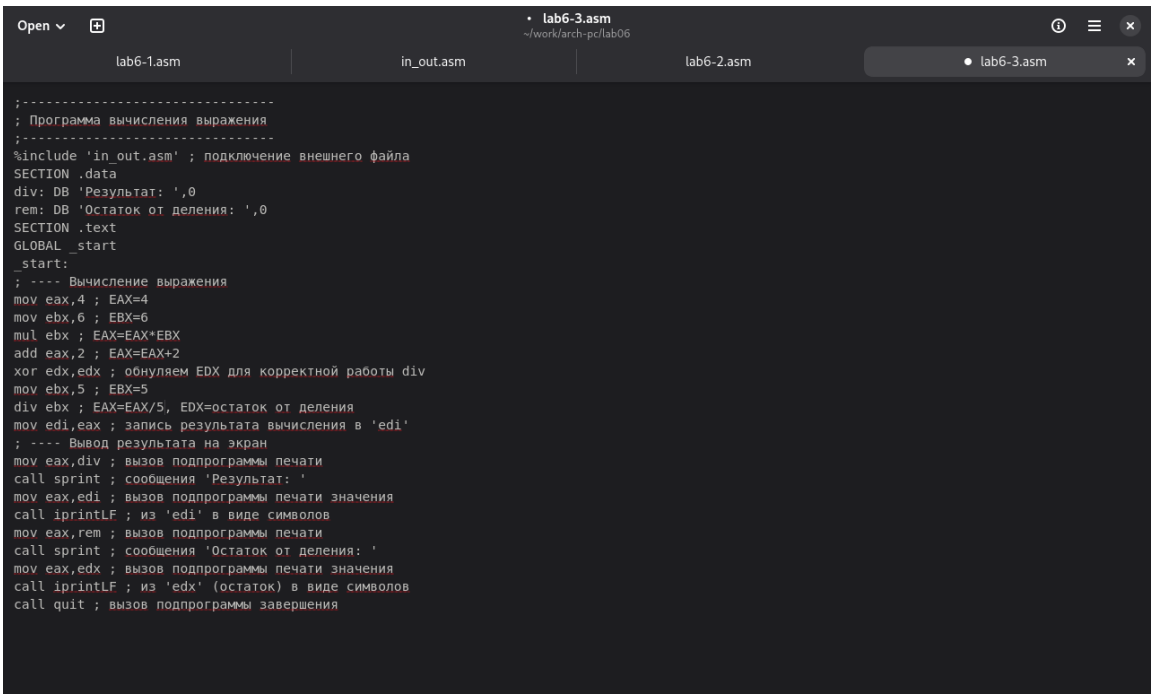

Рис. 17. Изменение программы

Создаю и запускаю новый исполняемый файл. (Рис.18.)

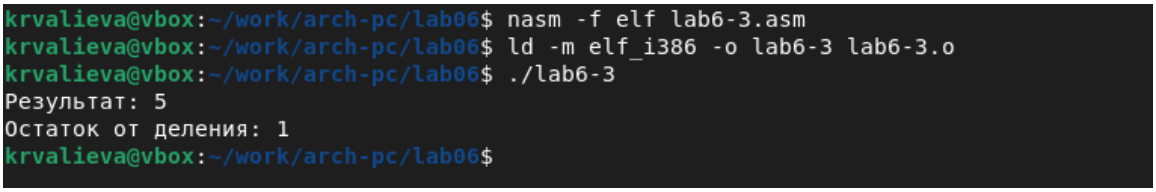

Рис. 18. Запуск исполняемого файла

Создаю файл variant.asm с помощью утилиты touch. (Рис.19.)

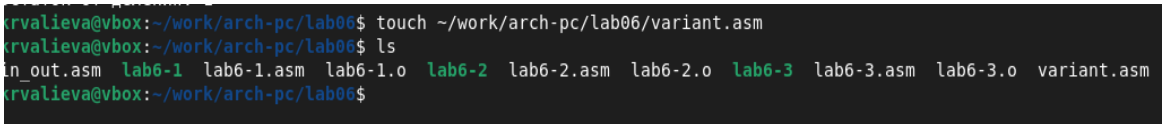

Рис. 19. Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета. (Рис.20.)

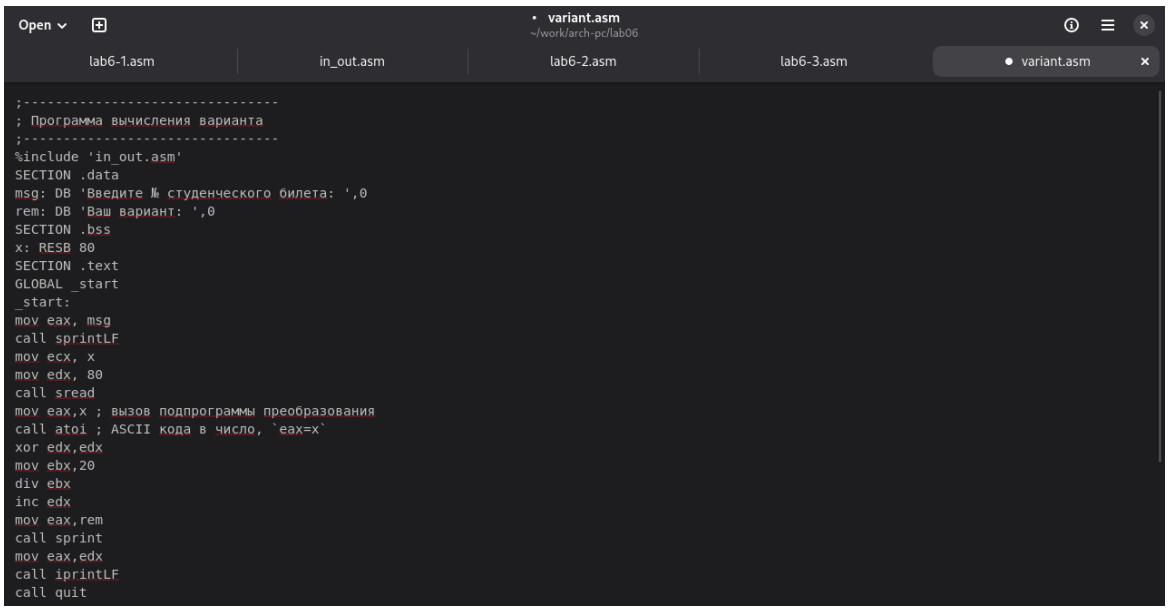

Рис. 20. Редактирование файла

Создаю и запускаю исполняемый файл. Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант – 15. (Рис.21.)

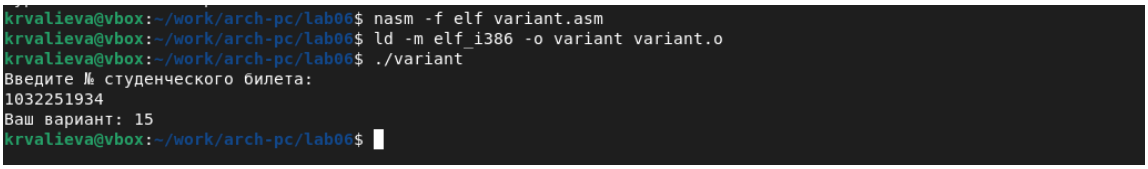

Рис. 21. Запуск исполняемого файла

4.2.1 Ответы на 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 - 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

Создаю файл lab6-4.asm с помощью утилиты touch. (Рис.22.)

Рис. 22. Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения $(5 + x)^2 - 3$ и подставляю значения $x1 = 5$ и $x2 = 1$. Это выражение было под вариантом 15. (Рис.23.)

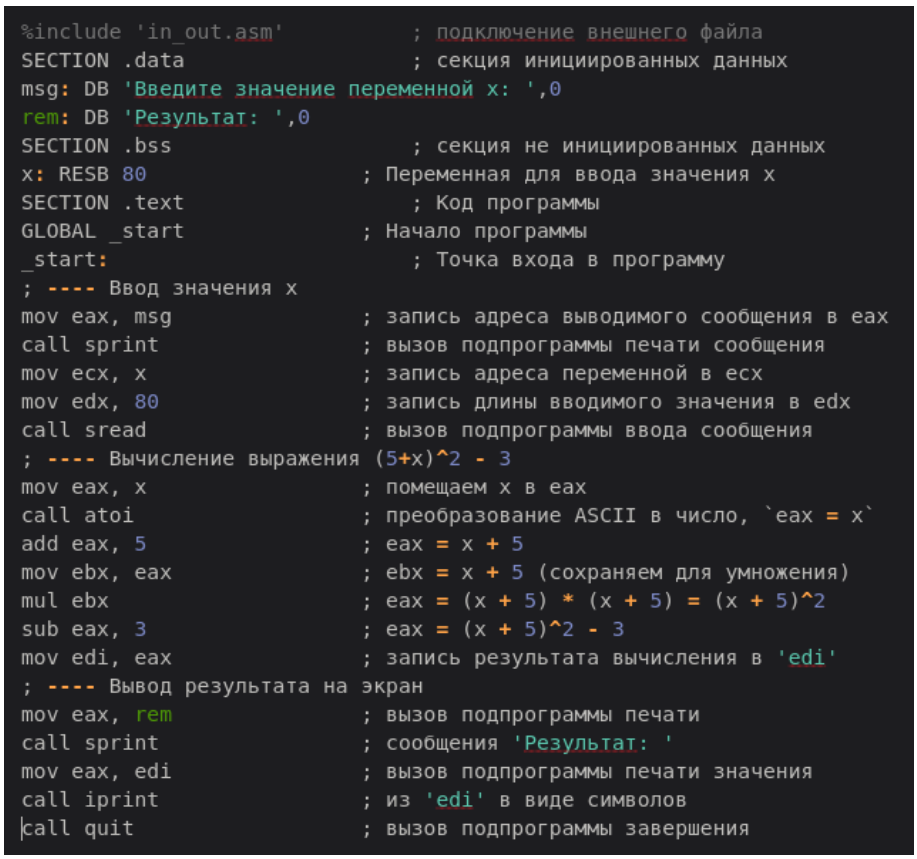

Рис. 23. Написание программы

Создаю и запускаю исполняемый файл. (Рис.24.)

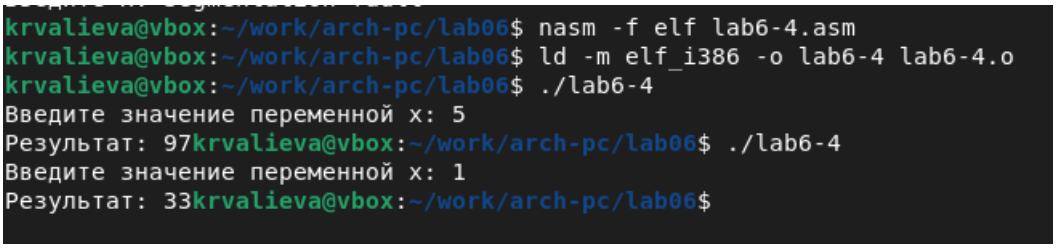

Рис. 24. Запуск исполняемого файла

Листинг 4.1. Программа для вычисления значения выражения $(5 + x)^2 - 3$

```
%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 для ввода значения x
SECTION .text                  ; Код программы
GLOBAL _start               ; Начало программы
_start:                        ; Точка входа в программу
; ---- Ввод значения x
```

```
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
; ---- Вычисление выражения (5+x)^2 - 3
mov eax, x                  ; помещаем x в eax
call atoi                   ; преобразование ASCII в число, `eax = x`
add eax, 5                  ; eax = x + 5
mov ebx, eax                ; ebx = x + 5 (сохраняем для умножения)
mul ebx                     ; eax = (x + 5) * (x + 5) = (x + 5)^2
sub eax, 3                  ; eax = (x + 5)^2 - 3
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.