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: Юсупова Амина Руслановна

Группа: НКАбд-06-25

МОСКВА

2025 г.

Содержание

1 Цель работы

Цель данной лабораторной 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

Создаю каталог lab06 для программ лабораторной работы №6 (с помощью команды mkdir) и перехожу в него. Создаю там файл lab6-1.asm(с помощью команды touch (рис. 1).

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

Рисунок 1: Создание нового каталога

В созданном файле lab6-1.asm ввожу программу из листинга (рис. 2).

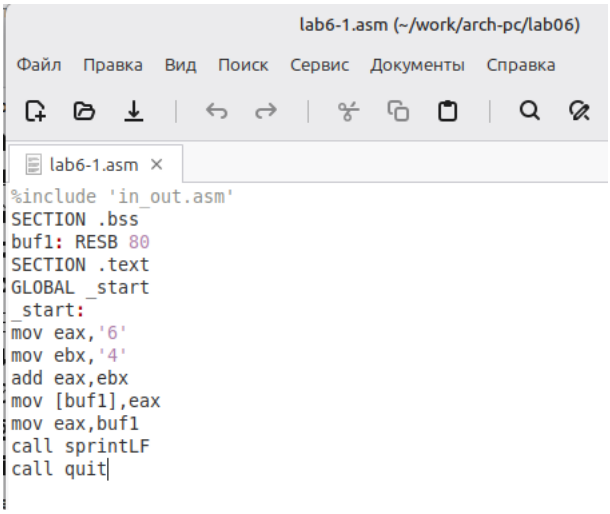

Рисунок 2: Ввод программы из листинга

Исходный код lab6-1.asm прикрепляю:

```
%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
```

Перед созданием исполняемого файла создала копию файла in_out.asm в каталоге ~/work/arch-pc/lab06 (рис. 3).

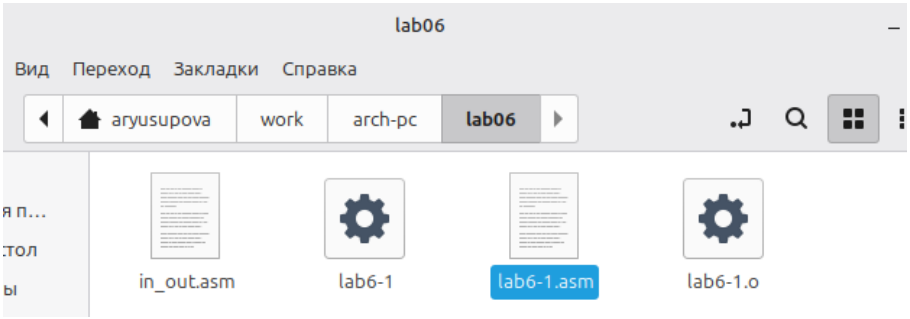

Рисунок 3: Копия файла in_out.asm в каталоге lab06

Создаю исполняемый файл и запускаю его, вывод программы отличается от предполагаемого изначально, ибо коды символов в сумме дают символ j по таблице ASCII. {#fig:003 width=70%}(рис. 4).

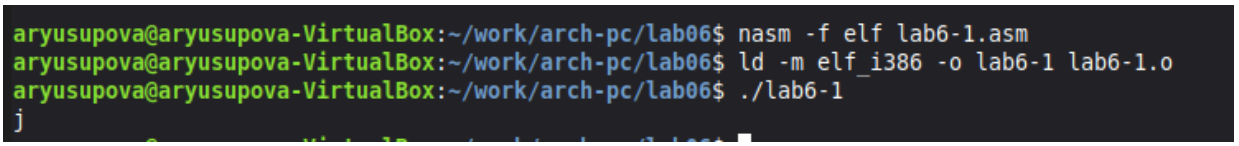

Рисунок 4: Создание и запуск файла lab6-1.asm

Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправьте текст программы следующим образом: замените строки mov eax,'6' mov ebx,'4' на строки mov eax,6 mov ebx,4. То есть уберём кавычки (рис. 5).

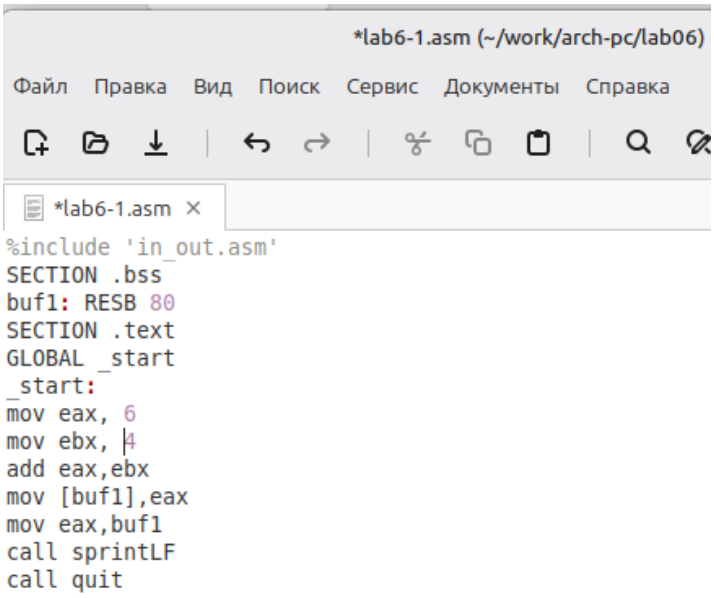

Рисунок 5: Изменение в исходном коде файла lab6-1.asm

Изменённый код lab6-1.asm прикрепляю:

%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
```

Создаём и запускаем обновленный файл. На этот раз программа выдала пустую строчку, это связано с тем, что символ 10 означает переход на новую строку (рис. 6).

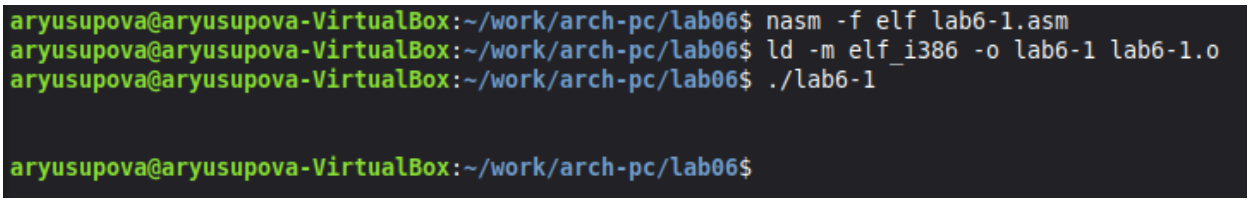

Рисунок 6: Запуск изменённой программы

Создаю файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 (рис. 7) и ввожу в него текст программы из листинга (рис. 8).

Рисунок 7: Создание файла lab6-2.asm в необходимом каталоге

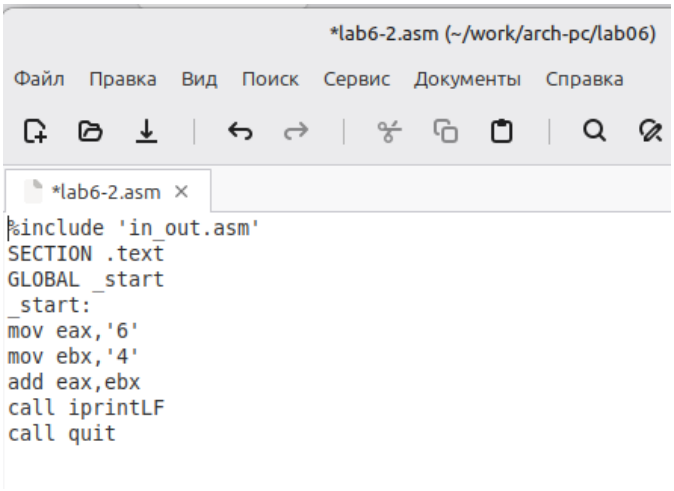

Рисунок 8: Ввод кода из листинга в файл lab6-2.asm

Прикладываю исходный код из файла lab6-2.asm:

%include 'in_out.asm'

SECTION .text

GLOBAL _start

_start:

mov eax,'6'

mov ebx,'4'

add eax,ebx

call iprintLF

call quit

Создаю исполняемый файл и запускаю его, теперь отображается результат 106, программа, как и в первый раз, сложила коды символов, но вывела само число, а не его символ, благодаря замене функции вывода на iprintLF (рис. 9).

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

Рисунок 9: Вывод второй программы файла lab6-2.asm

Аналогично предыдущему примеру изменим символы на числа. Замените строки mov eax,'6' mov ebx,'4' на строки mov eax,6 mov ebx,4. То есть уберём

кавычки (рис. 10).

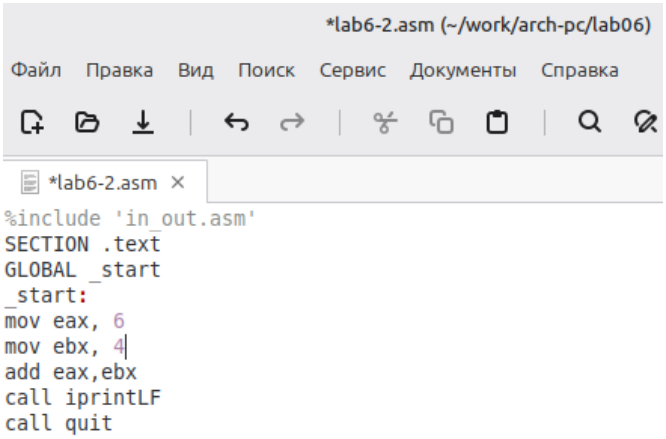

Рисунок 10: Изменённый код файла lab6-2.asm

Создаю исполняемый файл и запускаю его (рис. 11).

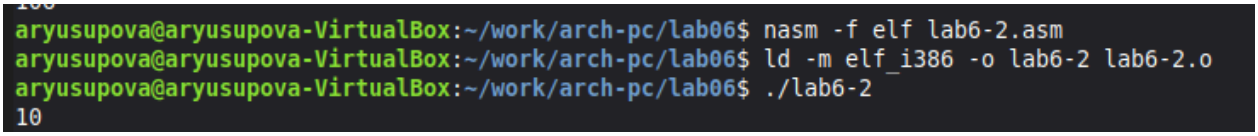

Рисунок 11: Вывод изменённой программы 2

В изменённом коде файла lab6-2.asm заменяю iprintLF на iprint (рис. 12). Создаю исполняемый файл и запускаю его (рис. 13).

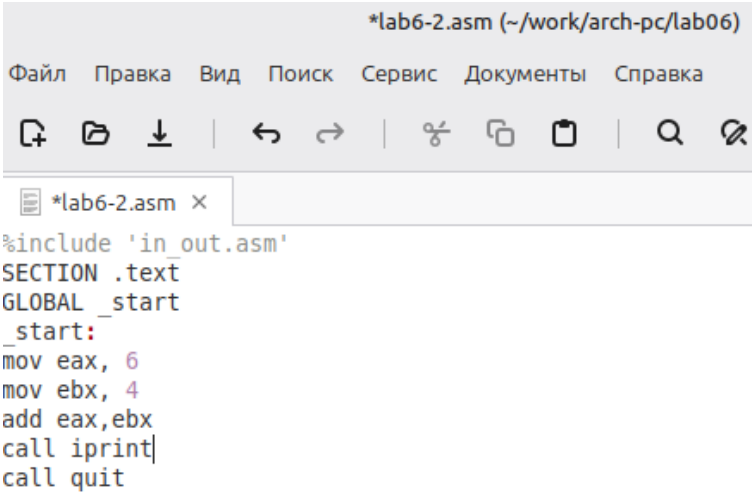

Рисунок 12: Замена функции вывода во второй программе

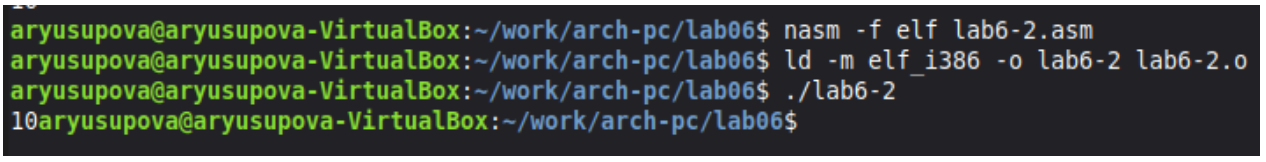

Рисунок 13: Вывод программы lab6-2.asm с заменой функции

Вывод функций iprintLF и iprint отличается тем, что в первом случае после

вывода значения идёт переход на следующую строку, а во втором случае перехода нет, и терминал продолжает работать на строке вывода значения.

4.2 Выполнение арифметических операций в NASM

Создаю новый файл lab6-3.asm (рис. 14) и копирую в него содержимое листинга (рис. 15).

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

Рисунок 14: Создание файла lab6-3.asm

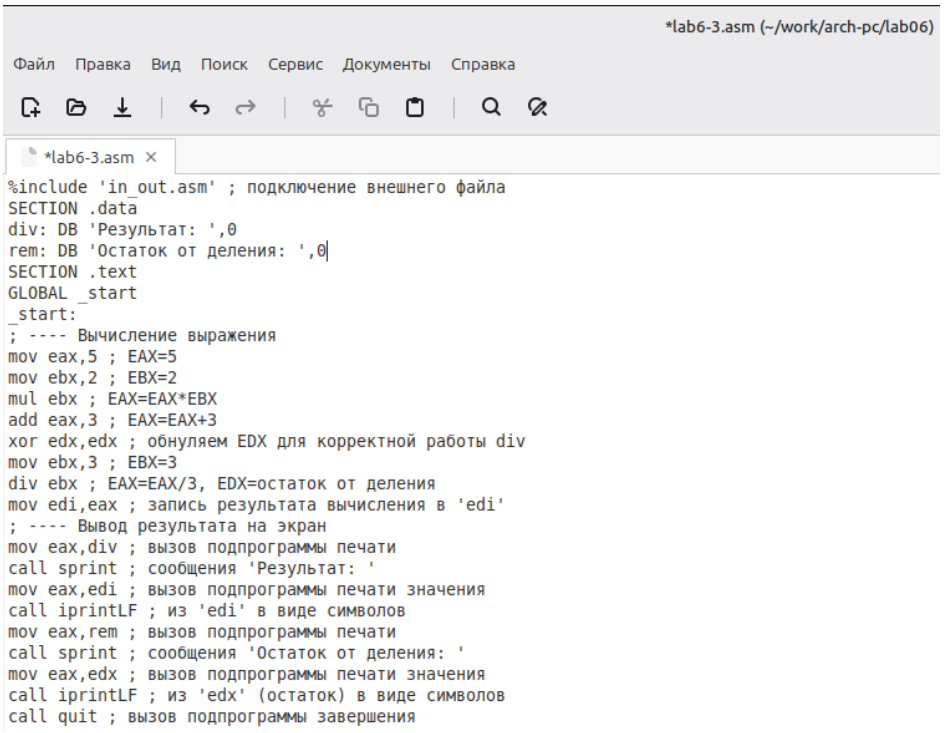

Рисунок 15: Код третьей программы

Создаю исполняемый файл и запускаю. Программа выполняет арифметические вычисления, на вывод идет результирующее выражения и его остаток от деления (рис. 16).

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

Рисунок 16: Вывод программы lab6-3.asm

Изменяю текст программы lab6-3.asm для вычисления выражения f(x)=(4*6+2)/5 (рис. 17).

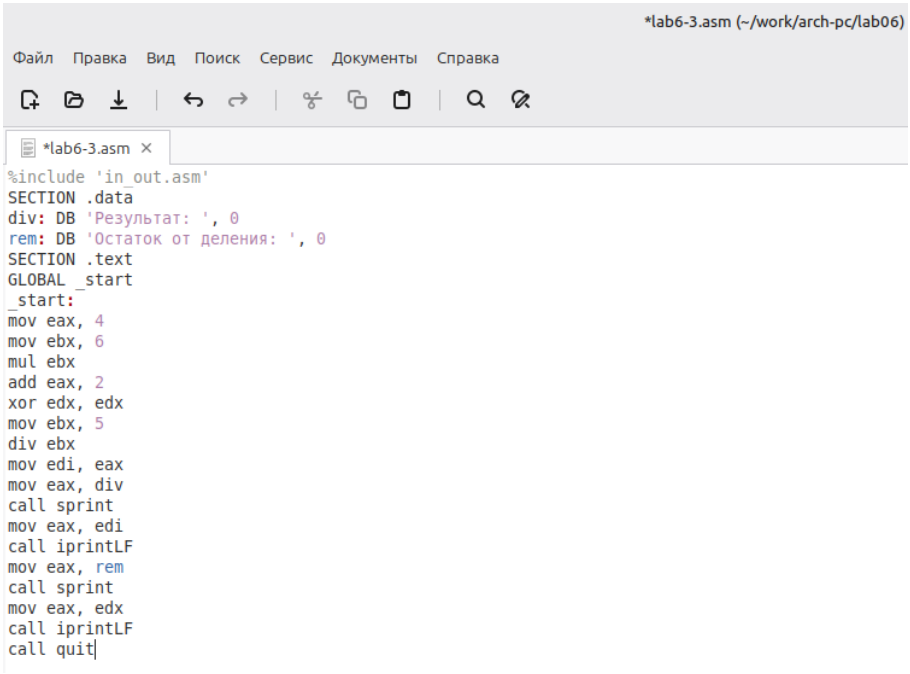

Рисунок 17: Изменение программы lab6-3.asm для другой функции

Запускаю программу и вижу правильный результат (рис. 18).

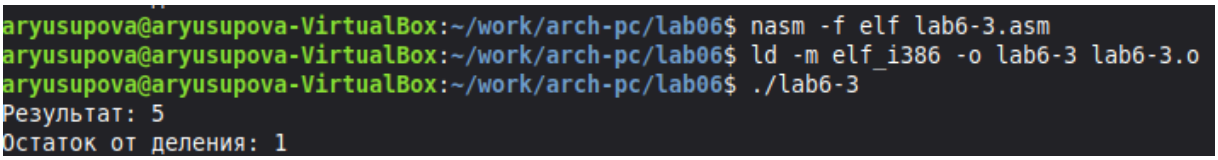

Рисунок 18: Запуск изменённой третьей программы

Создаю новый файл variant.asm (рис. 19) и помещаю текст из листинга (рис. 20).

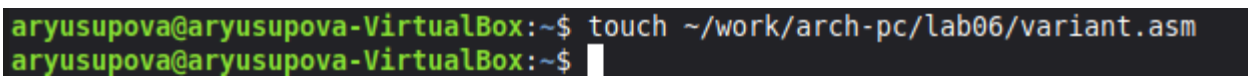

Рисунок 19: Создание файла variant.asm

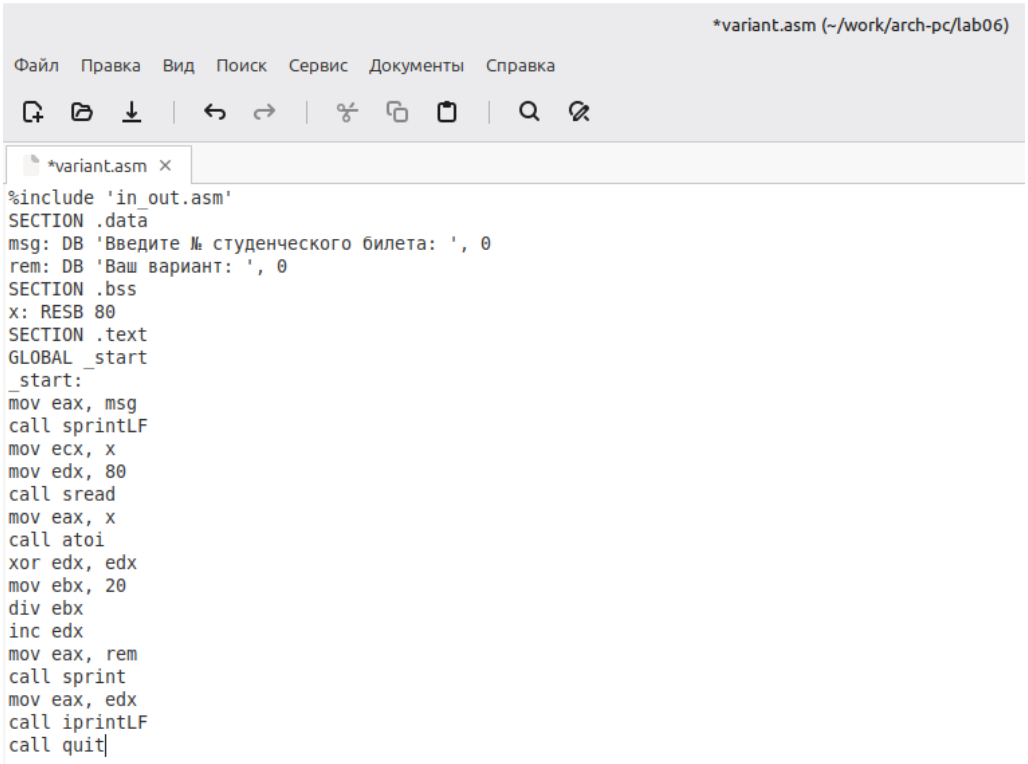

Рисунок 20: Ввод кода из листинга

Запустив программу и указав свой номер студенческого билета, я получила свой вариант для дальнейшей работы. (рис. 21).

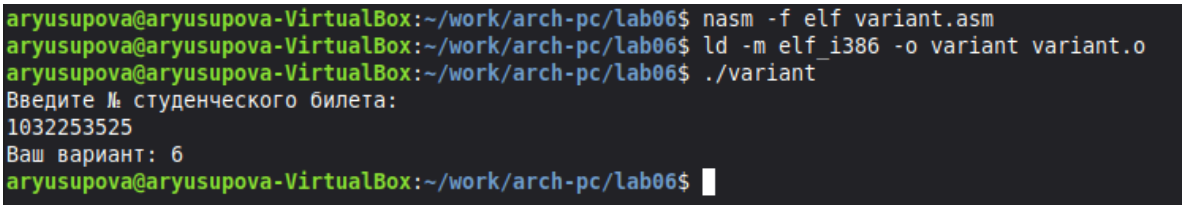

Рисунок 21: Запуск программы для подсчёта вариантов

4.3 Ответы на контрольные вопросы

4. 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
```

4.4 Задание для самостоятельной работы

Создаю файл lab6-4.asm для выполнения задания (написания кода) (рис.22).

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

Рисунок 22: Создание файла lab6-4.asm

В соответсвии с выбранным вариантом, я реализую программу для подсчета функции

3) $f(x) = (2+x)^2$(рис.23).

Проверяю корректность выполнения программы, подставляя x1=2 и x2=8. Посчитав аналитически заметила, что f(x1)=16 ; f(x2)=100. Это сходится с результатом программы (рис.24).

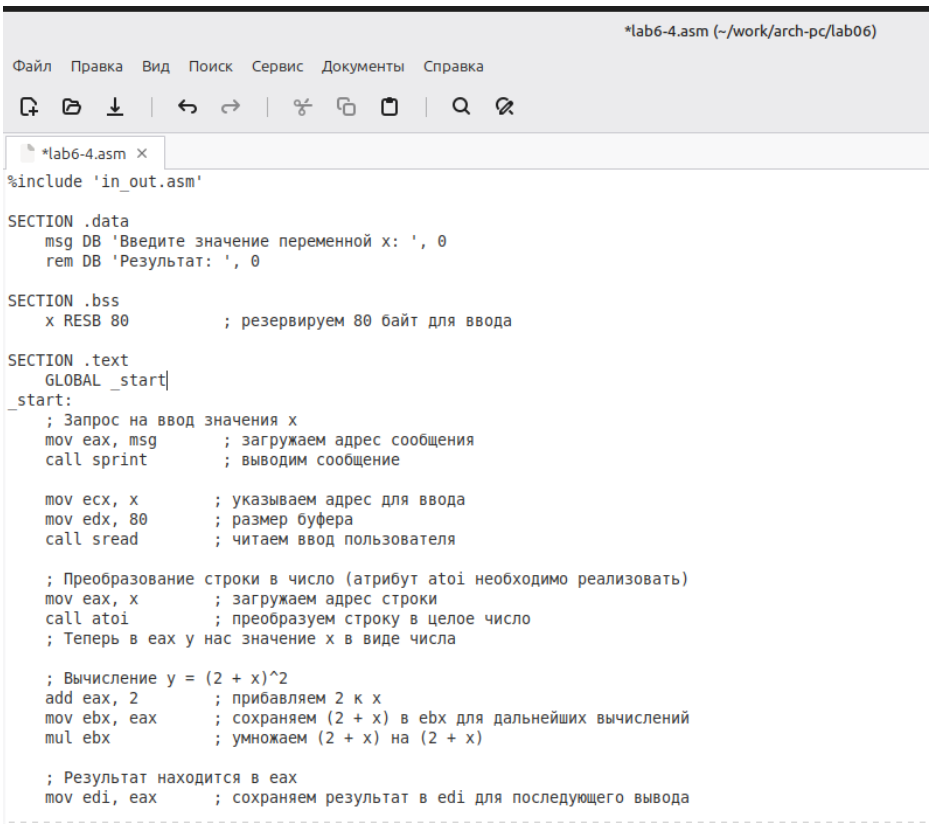

Рисунок 23: Код собственной программы на вычисления функции

Код прикладываю:

```
%include 'in_out.asm'

SECTION .data
   msg DB 'Введите значение переменной x: ', 0
   rem DB 'Результат: ', 0

SECTION .bss
   x RESB 80          ; резервируем 80 байт для ввода

SECTION .text
   GLOBAL _start

_start:
   ; Запрос на ввод значения x
   mov eax, msg       ; загружаем адрес сообщения
   call sprint        ; выводим сообщение
```

```
mov ecx, x        ; указываем адрес для ввода
mov edx, 80       ; размер буфера
call sread        ; читаем ввод пользователя

; Преобразование строки в число (атрибут atoi необходимо реализовать)
mov eax, x        ; загружаем адрес строки
call atoi         ; преобразуем строку в целое число
; Теперь в eax у нас значение x в виде числа

; Вычисление y = (2 + x)^2
add eax, 2        ; прибавляем 2 к x
mov ebx, eax      ; сохраняем (2 + x) в ebx для дальнейших вычислений
mul ebx           ; умножаем (2 + x) на (2 + x)

; Результат находится в eax
mov edi, eax      ; сохраняем результат в edi для последующего вывода

; Вывод результата
mov eax, rem      ; загружаем адрес строки результата
call sprint       ; выводим сообщение о результате
mov eax, edi      ; загружаем результат в eax
call iprint       ; выводим результат

; Завершение программы
mov eax, 1        ; системный вызов для выхода
xor ebx, ebx      ; код возврата 0
int 0x80          ; вызов ядра
```

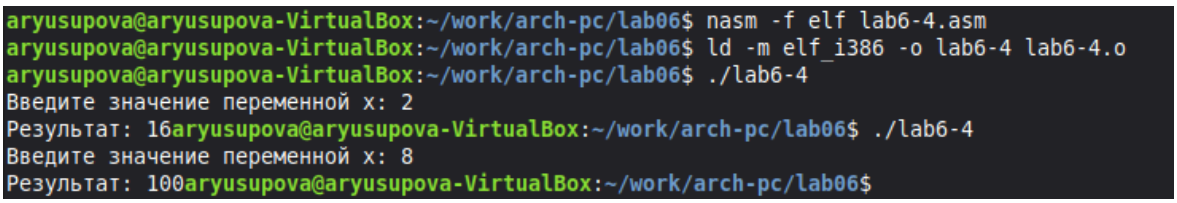

Рисунок 24: Запуск и проверка программы

5 Выводы

В ходе выполнения данной лабораторной работы я изучила арифметические инструкции языка ассемблера NASM.

6 Список литературы

1. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm_unix.

2. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).

3. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).

4. https://esystem.rudn.ru/mod/resource/view.php?id=1030554