Отчёт по лабораторной работе №6

Дисциплина: Архитектура компьютера

Наговицын Арсений Владимирович

Содержание

# 1 Цель работы

Цель данной лабораторный 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

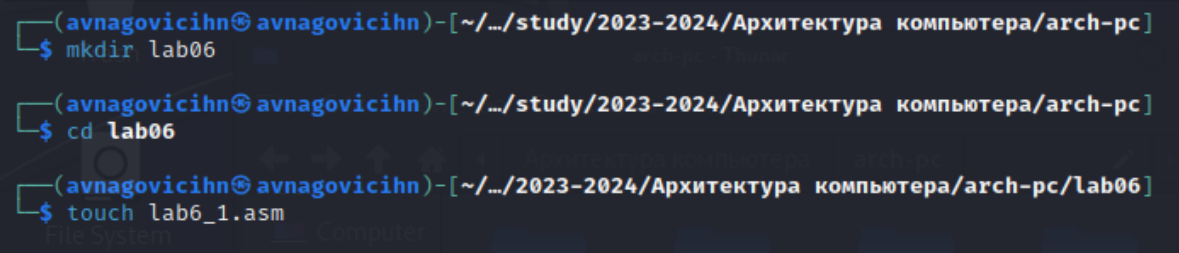
Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. - Регистровая адресация – операнды хранятся в регистрах и в команде используются имена 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.

# 4 Выполнение лабораторной работы

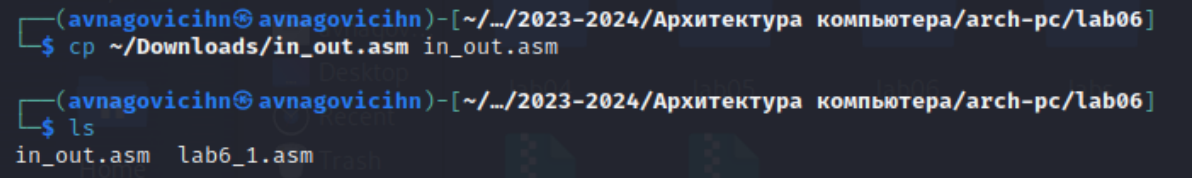
## 4.1 Выполнение арифметических операций в NASM

Создаю директорию, в которой буду выполнять лабораторную работу (рис. ??).



Создание директории и файла

Копирую в каталог файл in\_out.asm (рис. ??).



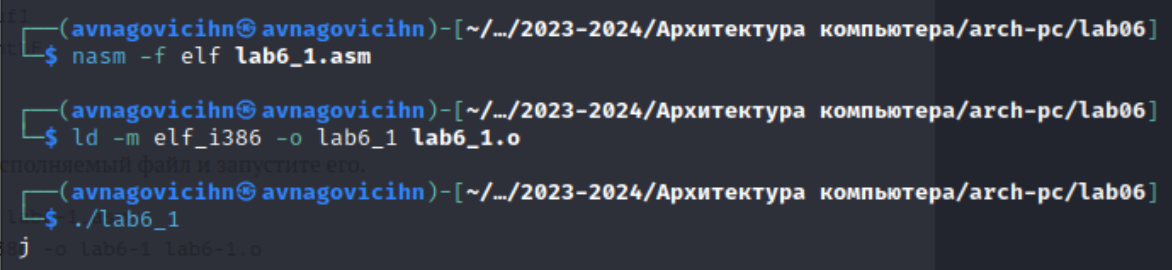
Копирование файла

Открываю созданный файл и вставляю в него программу для вывода (рис. ??).



Копирование файла

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



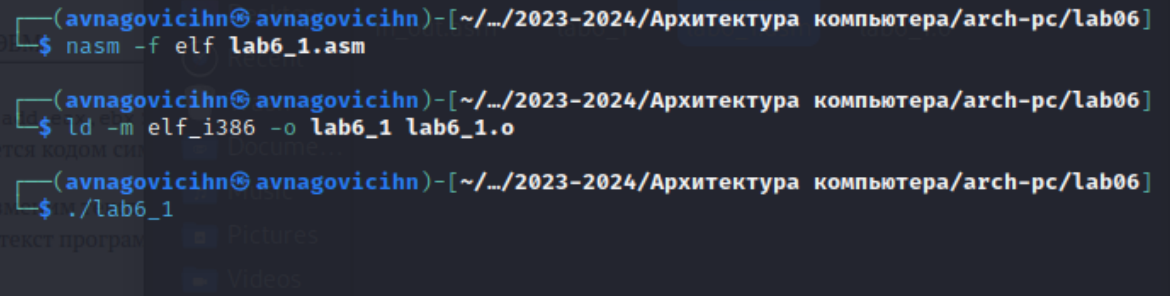
Компиляция и запуск файла

Изменяю текст программы (рис. ??).



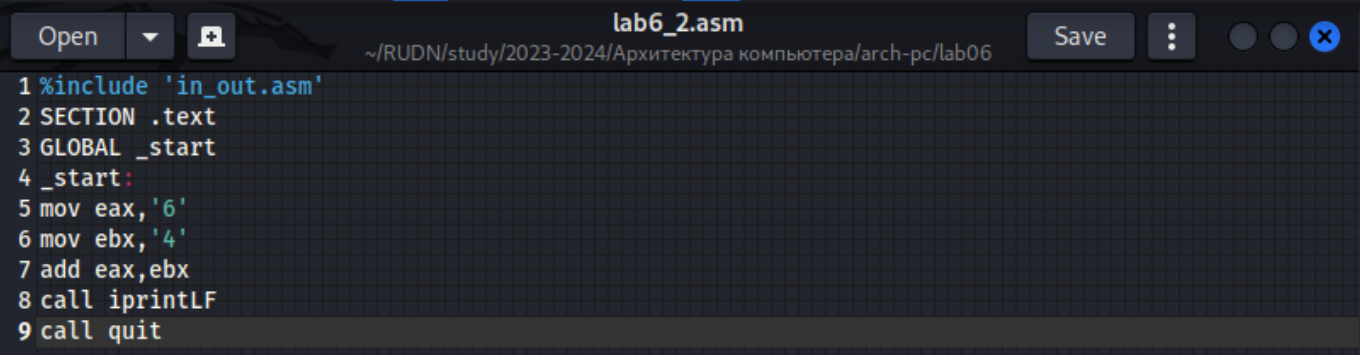
Редактирование файла

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



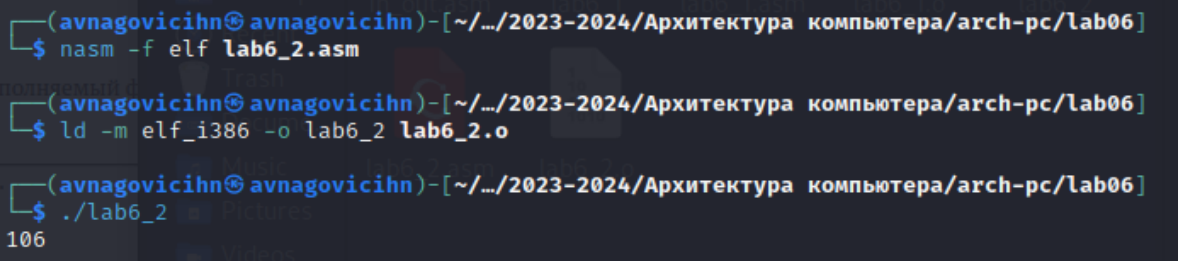
Компиляция и запуск файла

Создав файл, ввожу текст программы (рис. ??).



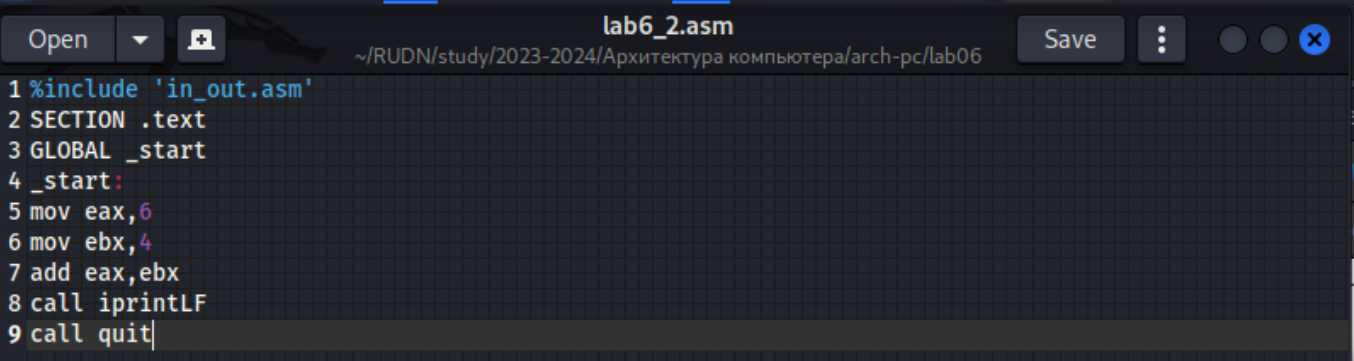
Редактирование файла

Создаю исполняемый файл и запускаю его (рис. ??). Теперь вывод 10, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”.



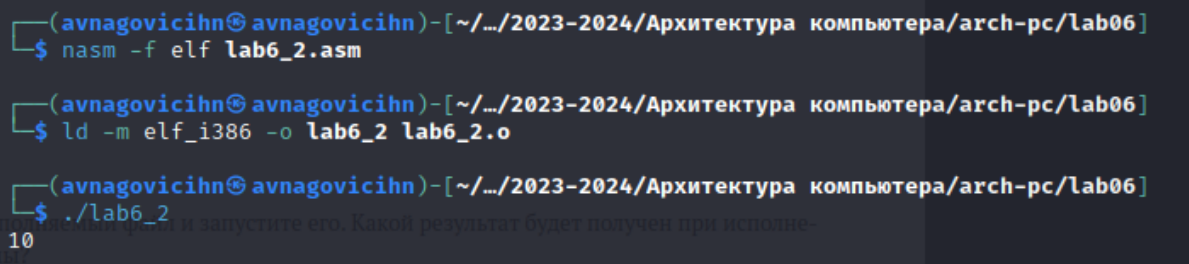
Компиляция и запуск файла

Заменяю в тексте файла символы “6” и “4” на 6 и 4 (рис. ??).



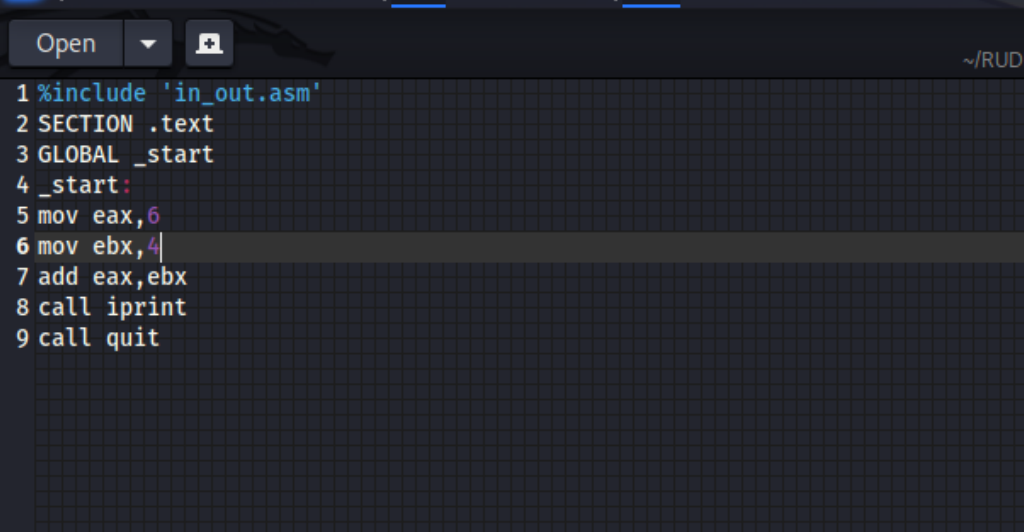
Редактирование файла

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



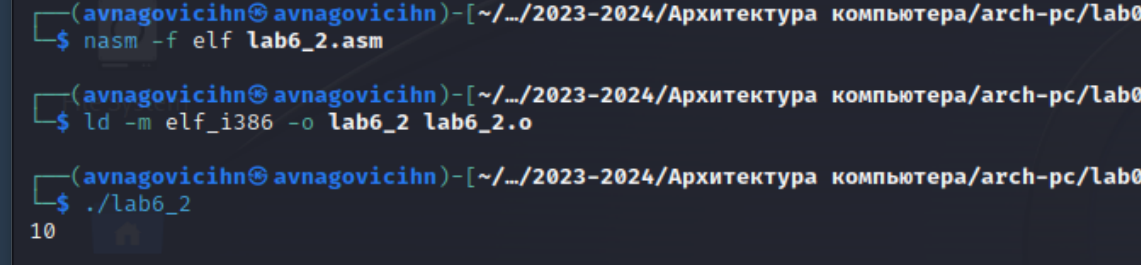
Компиляция и запуск файла

Заменяю в тексте файла функцию iprintLF на iprint (рис. ??).



Редактирование файла

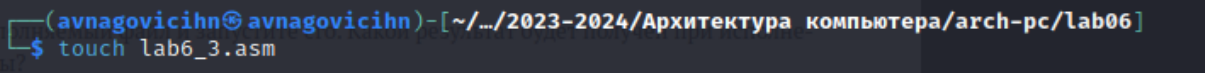
Создаю исполняемый файл и запускаю его (рис. ??). Вывод не изменился, потому что символ переноса строки не отображается, когда программа исполнялась с функцией iprintLF, а iprint не добавляет к выводу символ переноса строки.



Компиляция и запуск файла

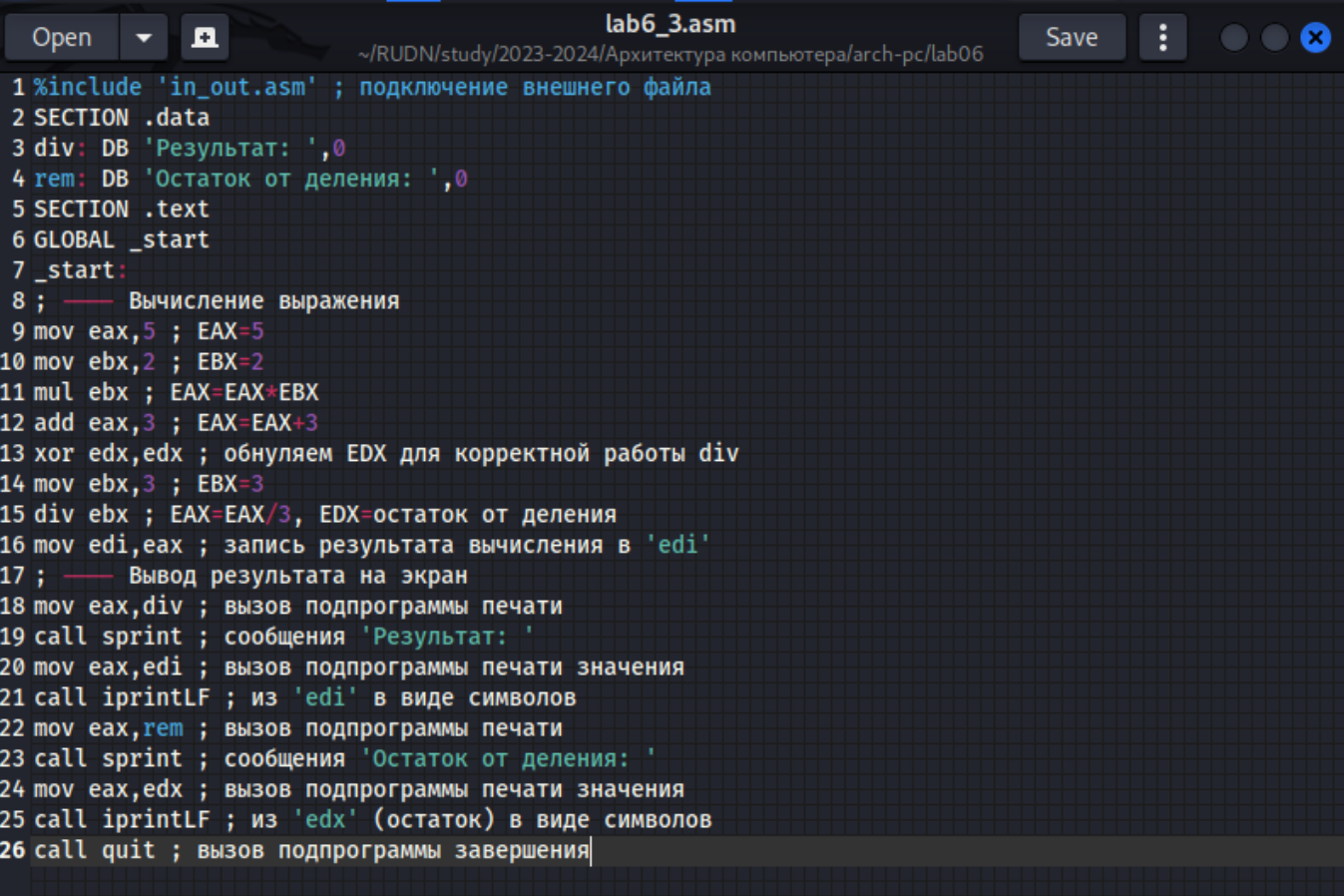
## 4.2 Выполнение арифметических операций в NASM

Создаю файл (рис. ??).



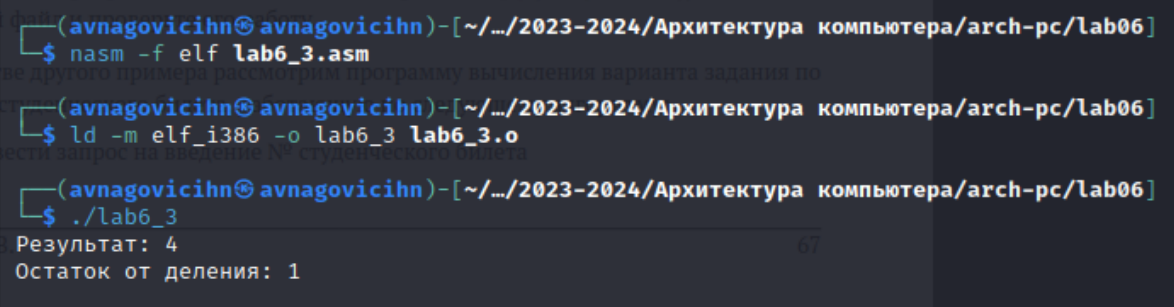
Создание файла

Ввожу текст программы в созданный файл (рис. ??).



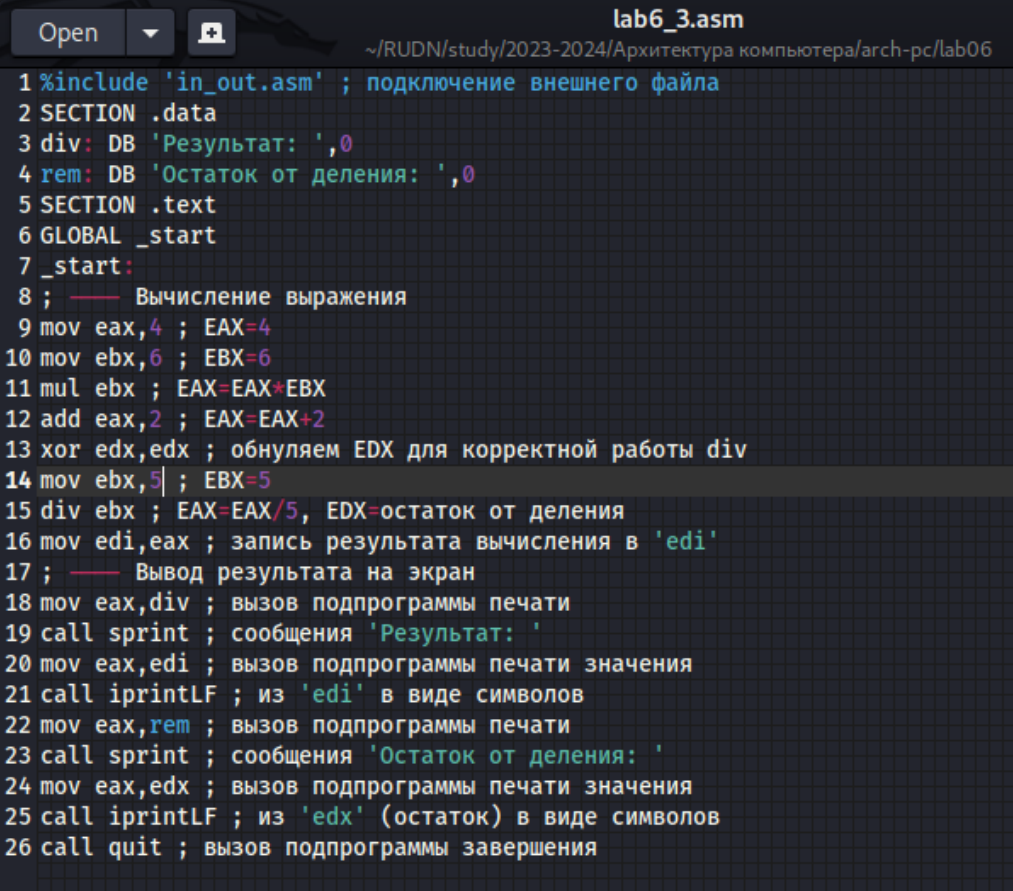
Редактирование файла

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



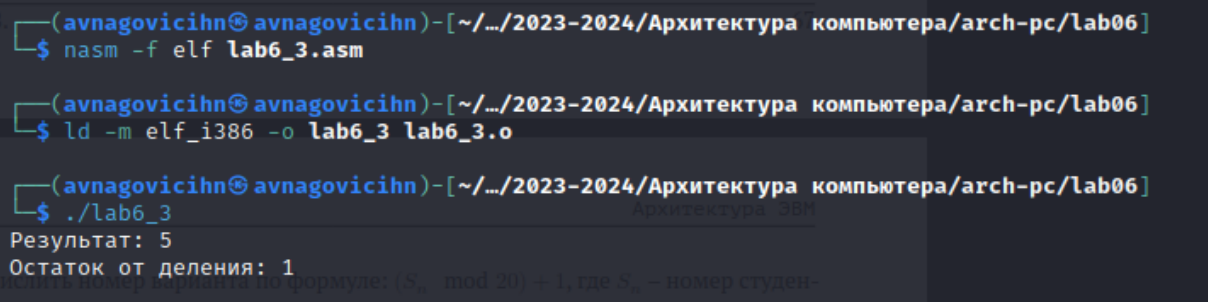
Компиляция и запуск файла

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



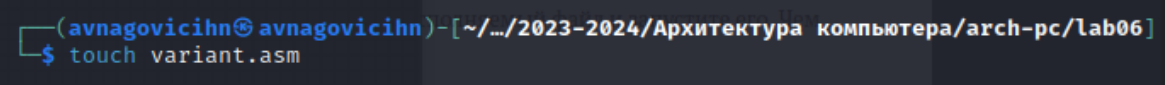
Редактирование файла

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



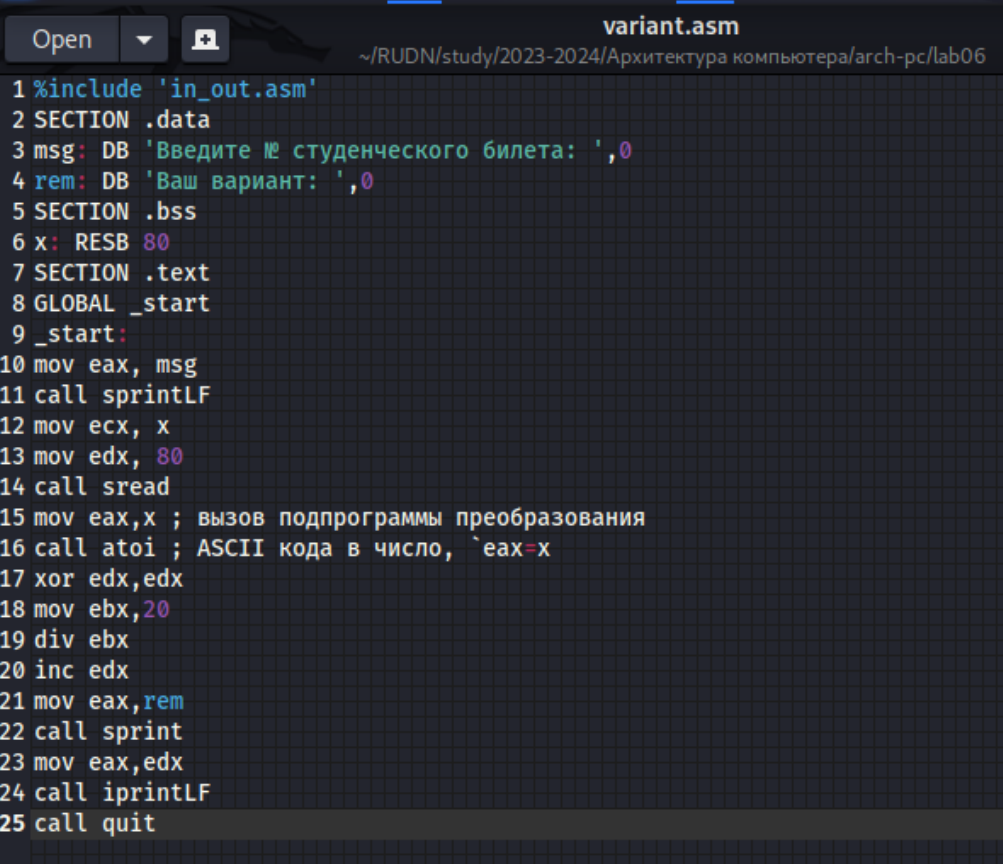
Компиляция и запуск файла

Создаю файл variant.asm (рис. ??).



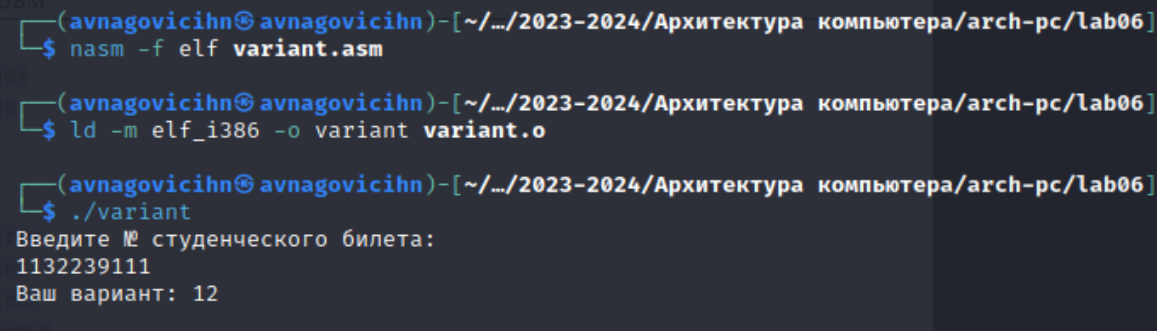
Создание файла

Ввожу текст программы в созданный файл (рис. ??).



Редактирование файла

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



Компиляция и запуск файла

### 4.2.1 Ответы на вопросы по программе

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

mov eax,rem call sprint

1. 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
2. call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax
3. За вычисления варианта отвечают строки:

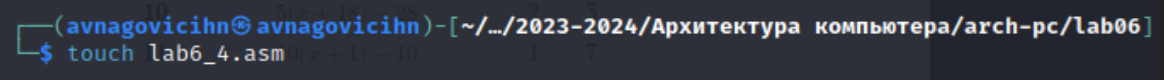
xor edx,edx mov ebx,2 div ebx inc edx

1. При выполнении инструкции div ebx остаток от деления записывается в регистр edx
2. inc edx увеличивает значение регистра edx на 1
3. За вывод на экран результатов вычислений отвечают строки:

mov eax,edx call iprintLF

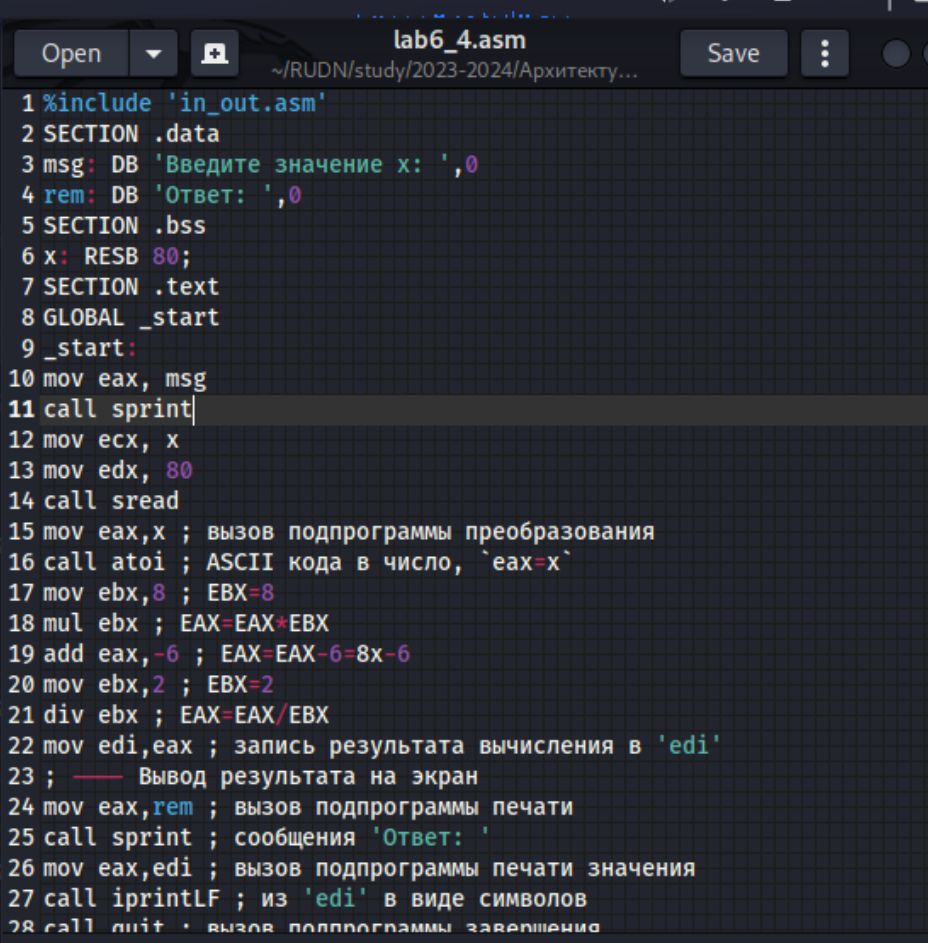
## 4.3 Выполнение заданий для самостоятельной работы.

Создаю файл lab6\_4.asm (рис. ??).



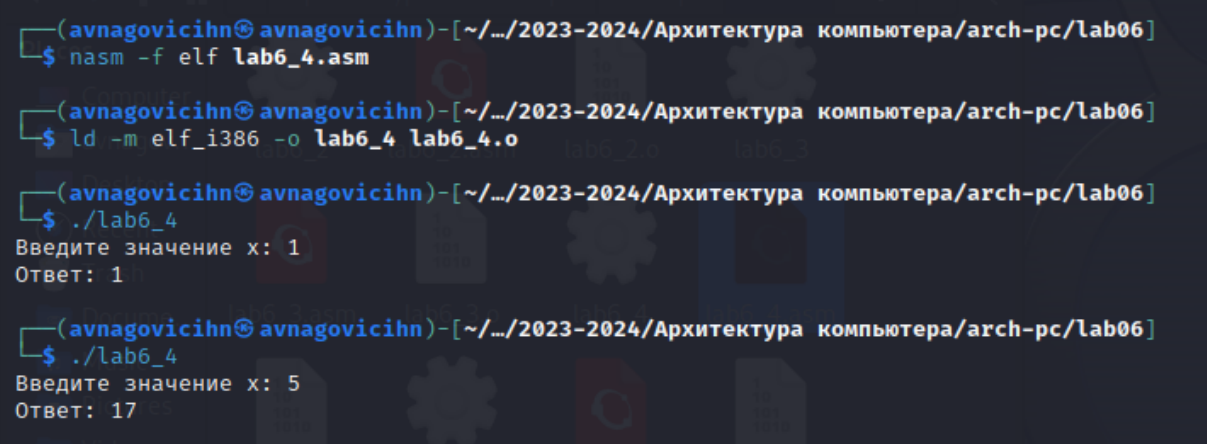
Создание файла

Ввожу текст программы для вычислений выражения f=(8x-6)/2 (рис. ??).



Редактирование файла

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



Компиляция и запуск файла

**Листинг 4.1. Программа для вычисления значения выражения (8 x - 6) / 2 .**

%include 'in\_out.asm'  
SECTION .data  
msg: DB 'Введите значение x: ',0  
rem: DB 'Ответ: ',0  
SECTION .bss  
x: RESB 80;  
SECTION .text  
GLOBAL \_start  
\_start:  
mov eax, msg  
call sprint  
mov ecx, x  
mov edx, 80  
call sread  
mov eax,x ; вызов подпрограммы преобразования  
call atoi ; ASCII кода в число, `eax=x`  
mov ebx,8 ; EBX=8  
mul ebx ; EAX=EAX\*EBX  
add eax,-6 ; EAX=EAX-6=8x-6  
mov ebx,2 ; EBX=2  
div ebx ; EAX=EAX/EBX  
mov edi,eax ; запись результата вычисления в 'edi'  
; ---- Вывод результата на экран  
mov eax,rem ; вызов подпрограммы печати  
call sprint ; сообщения 'Ответ: '  
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 освоил арифметические инструкции языка ассемблера NASM.