Отчёт по лабораторной работе №7

*дисциплина: Архитектура компьютера*

Максим Александрович Мишонков

Содержание

# 1 Цель работы

Целью данной лабораторной работы является освоение арифметических инструкций языка ассемблера NASM.

# 2 Задание

Освоить арифметические инструкции языка ассемблера NASM.

# 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.

Существует три основных способа адресации:

**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

# 4 Выполнение лабораторной работы

1. Создал каталог для программам лабораторной работы №7, перешёл в него и создал файл lab7-1.asm. (рис. 1)

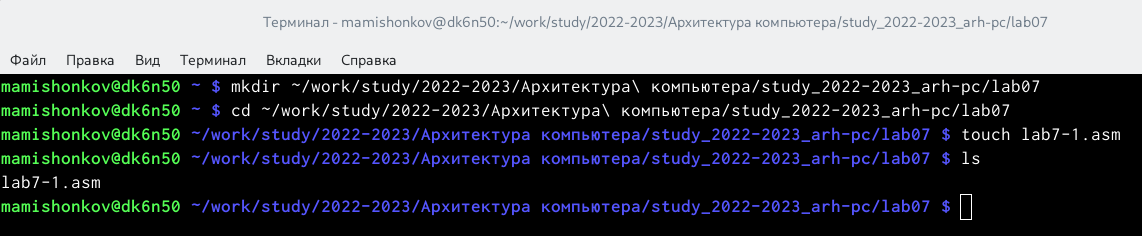


Рис. 1: Создание каталога и файла

1. Ввёл в файл lab7-1.asm текст программы из листинга 7.1. (рис. 2)

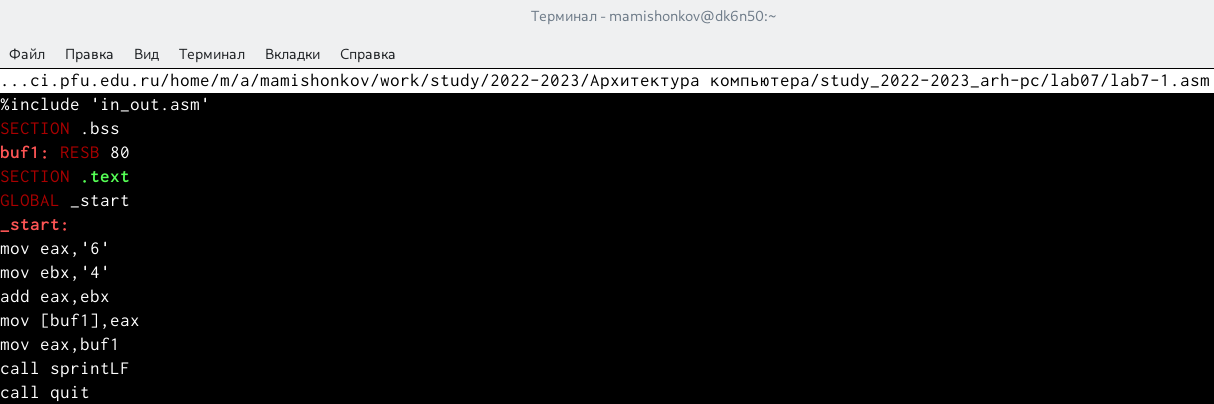


Рис. 2: Введение текста программы

1. Создал исполняемый файл и запустил его. Программа вывела символ j. (рис. 3)

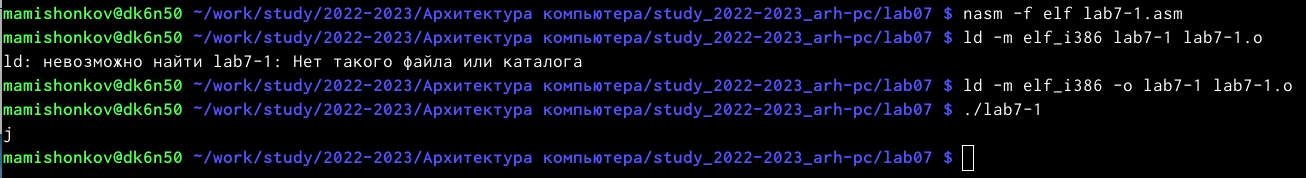


Рис. 3: Проверка работы исполняемого файла

1. Изменил текст программы, вместо символов записав в регистры числа. (рис. 4)

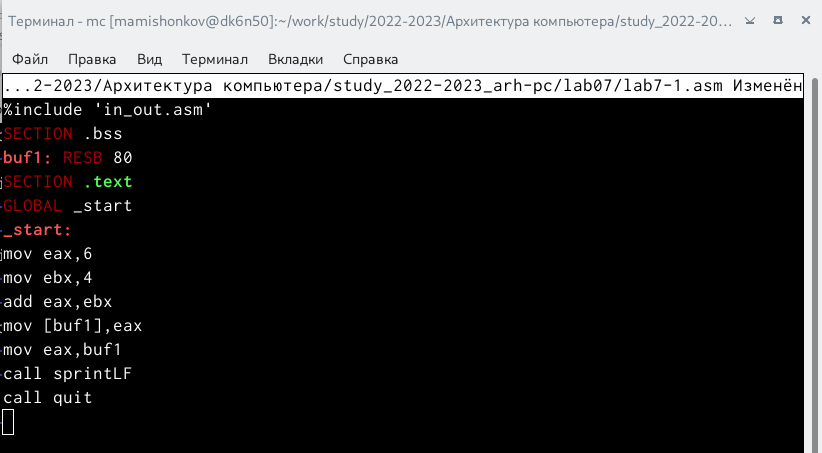


Рис. 4: Изменение текста программы

1. Создал исполняемый файл и запустил его. Программа вывела невидимый символ с кодом 10. (рис. 5)

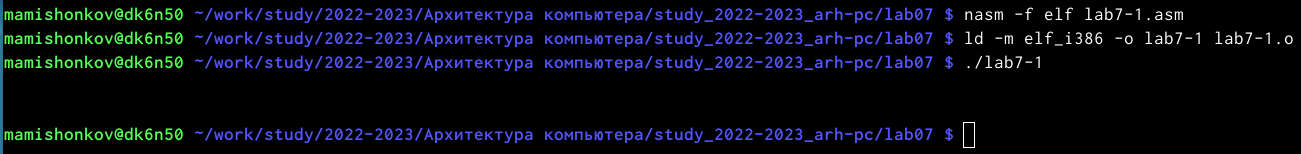


Рис. 5: Проверка работы исполняемого файла

1. Создал файл lab7-2.asm в каталоге lab07 и ввёл в него текст программы из листинга 7.2. (рис. 6, 7)

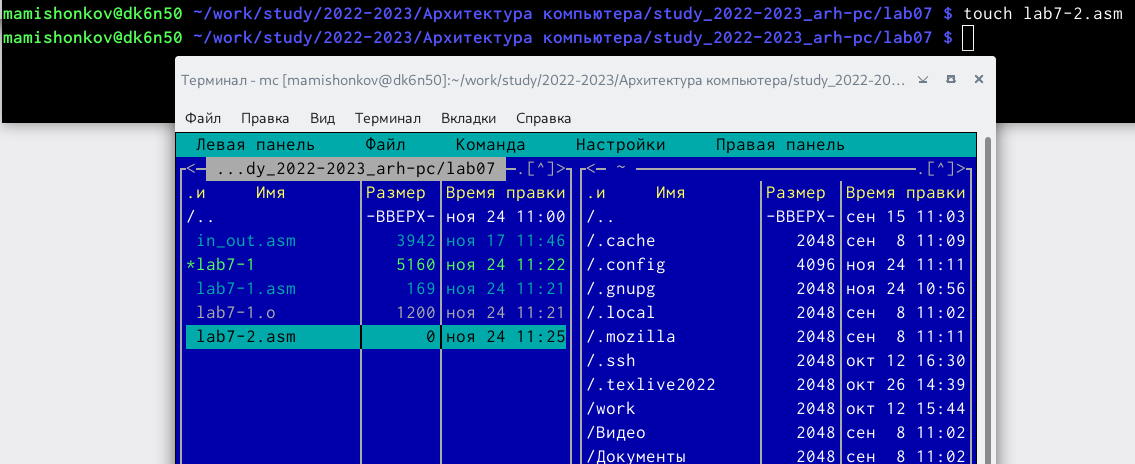


Рис. 6: Создание файла

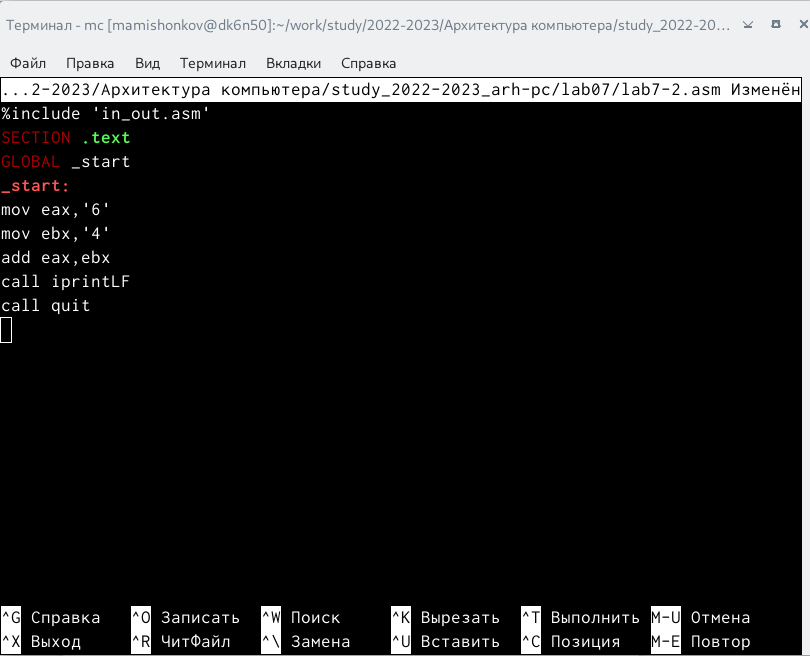


Рис. 7: Введение текста программы

1. Создал исполняемый файл и запустил его. Программа вывела число 106. (рис. 8)

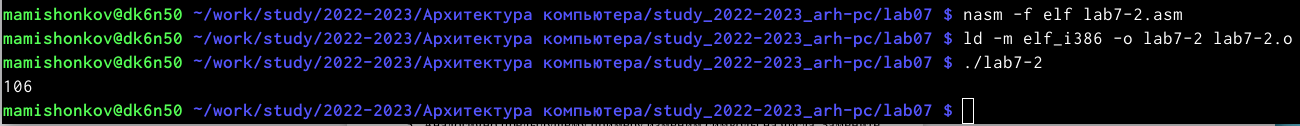


Рис. 8: Проверка работы исполняемого файла

1. Изменил текст программы, вместо символов записав в регистры числа. (рис. 9)

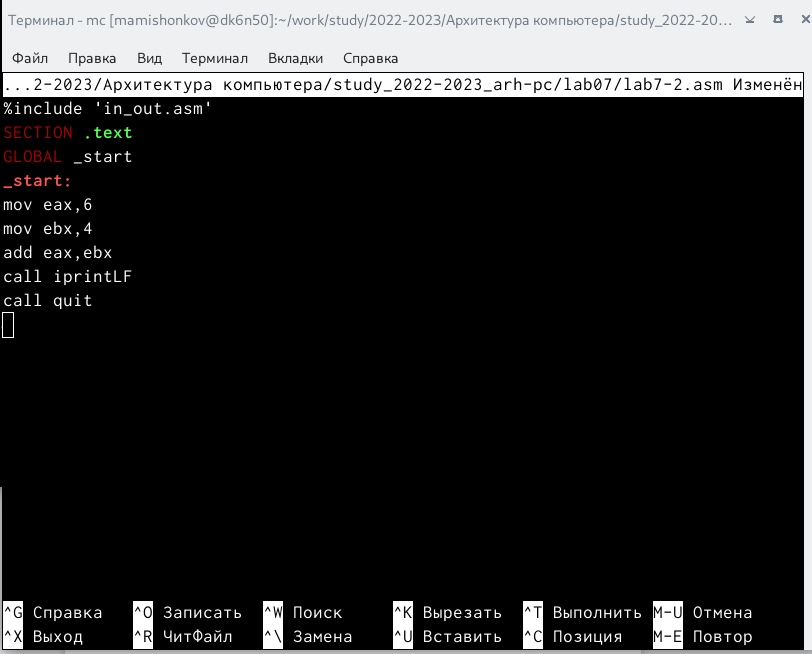


Рис. 9: Изменение текста программы

1. Создал исполняемый файл и запустил его. Программа вывела число 10. (рис. 10)

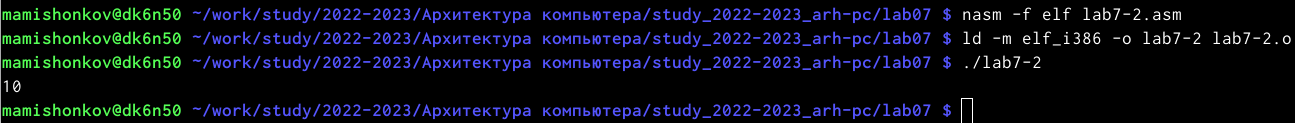


Рис. 10: Проверка работы исполняемого файла

1. Заменил в тексте программы функцию iprintLF на iprint. (рис. 11)

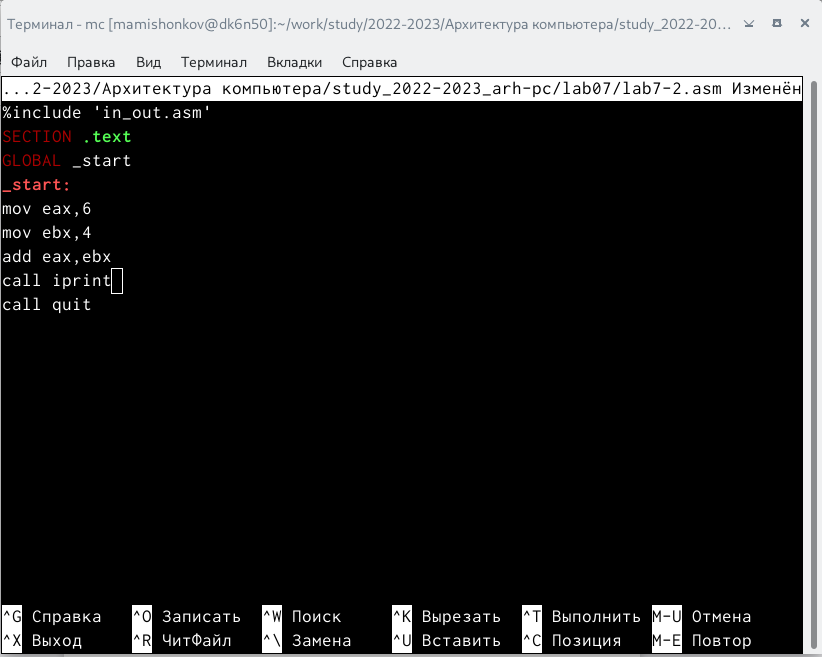


Рис. 11: Изменение текста программы

1. Создал исполняемый файл и запустил его. Результат отличается в выводе данных: при команде iprintLF мы начинаем вводить команду на следующей строке, а при использовании команды iprint мы вводим данные на той же строке. (рис. 12)

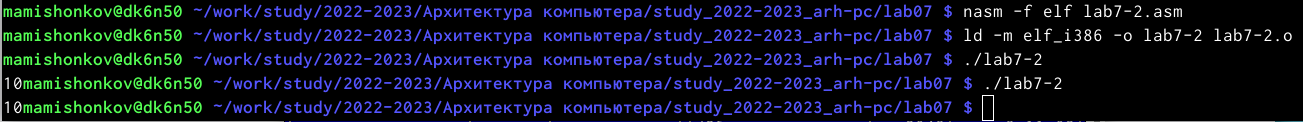


Рис. 12: Проверка работы исполняемого файла

1. Создал файл lab7-3.asm в каталоге lab07. (рис. 13)

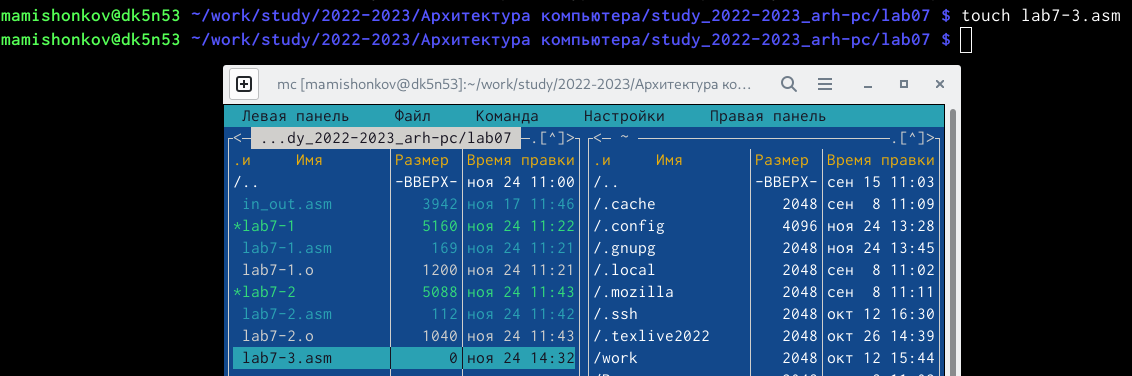


Рис. 13: Создание файла

1. Ввёл в lab7-3.asm. текст программы из листинга 7.3. (рис. 14)

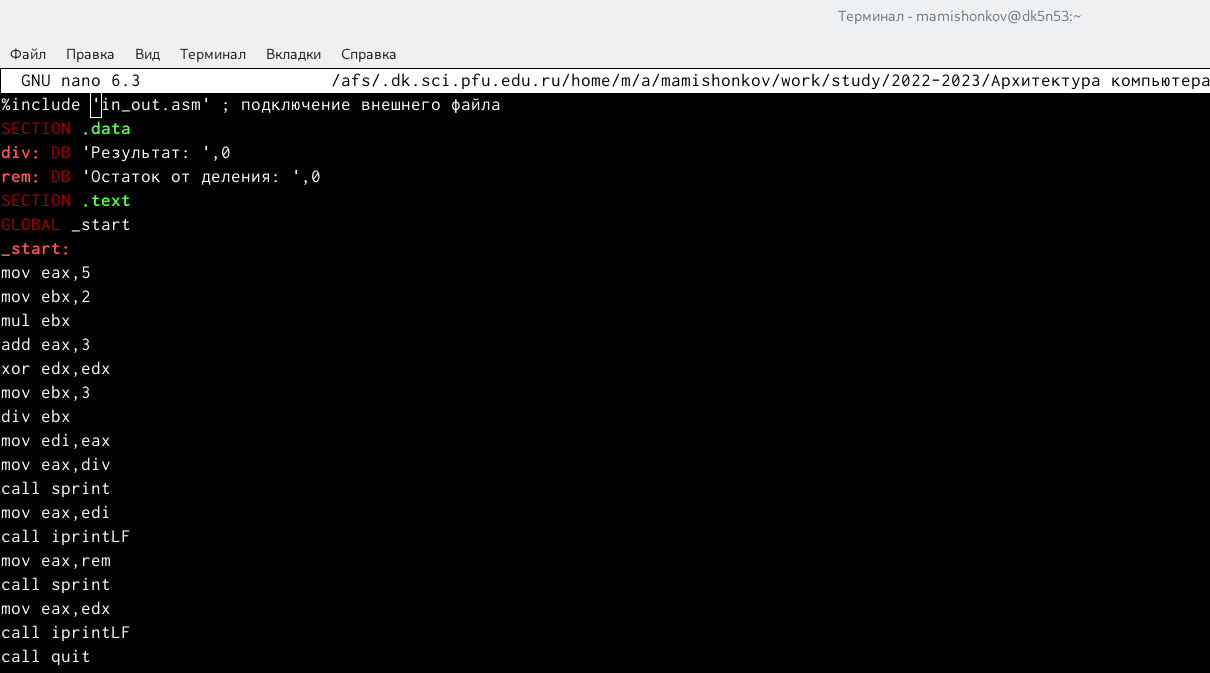


Рис. 14: Введение текста программы

1. Создал исполняемый файл и проверил его работу. (рис. 15)

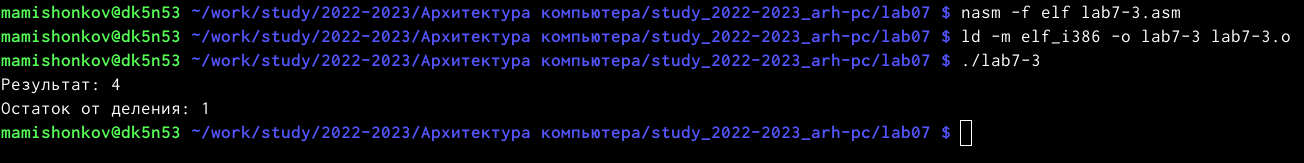


Рис. 15: Проверка работы исполняемого файла

1. Изменил текст программы для вычисления выражения данного выражения. Создал исполняемый файл и проверил его работу. (рис. 16, 17)

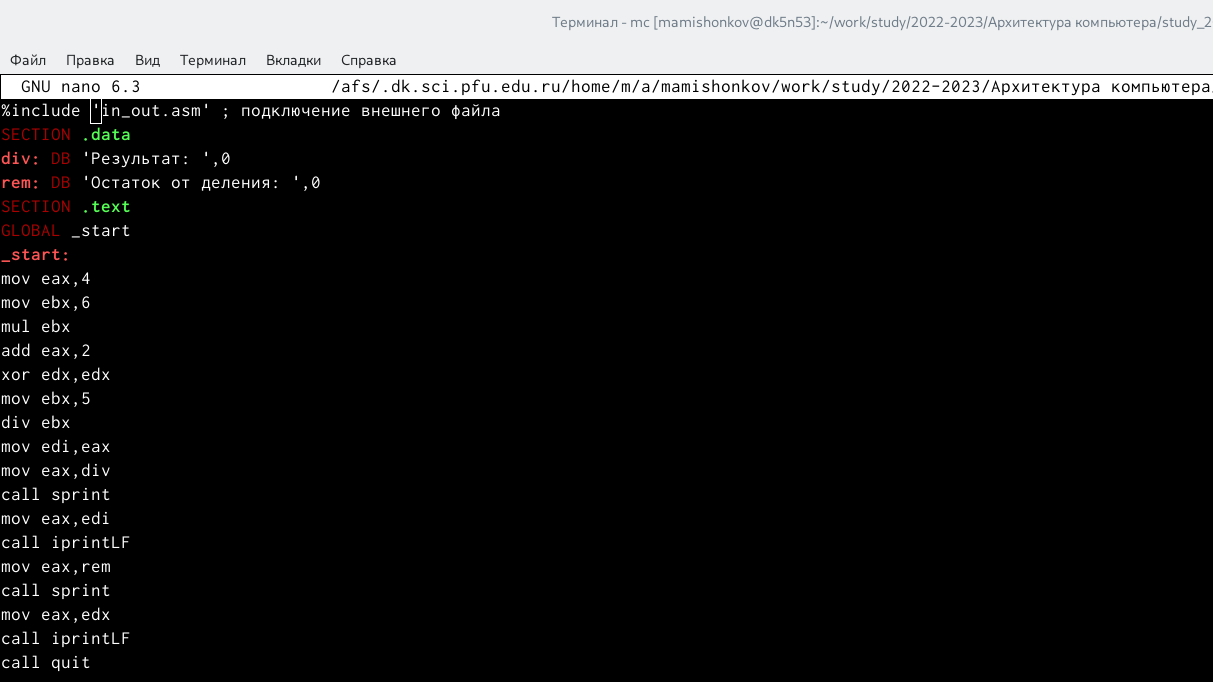


Рис. 16: Изменение текста программы

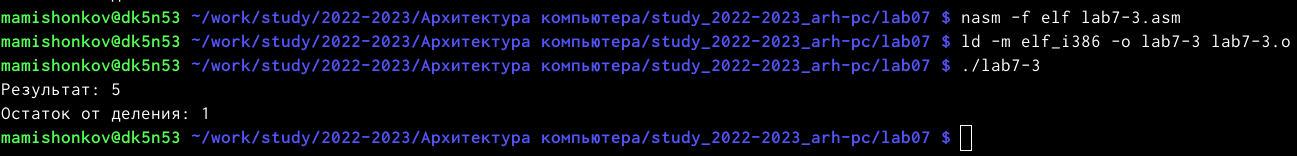


Рис. 17: Проверка работы исполняемого файла

1. Создал файл variant.asm в каталоге lab07 и ввёл в него текст программы из листинга 7.4. (рис. 18, 19)

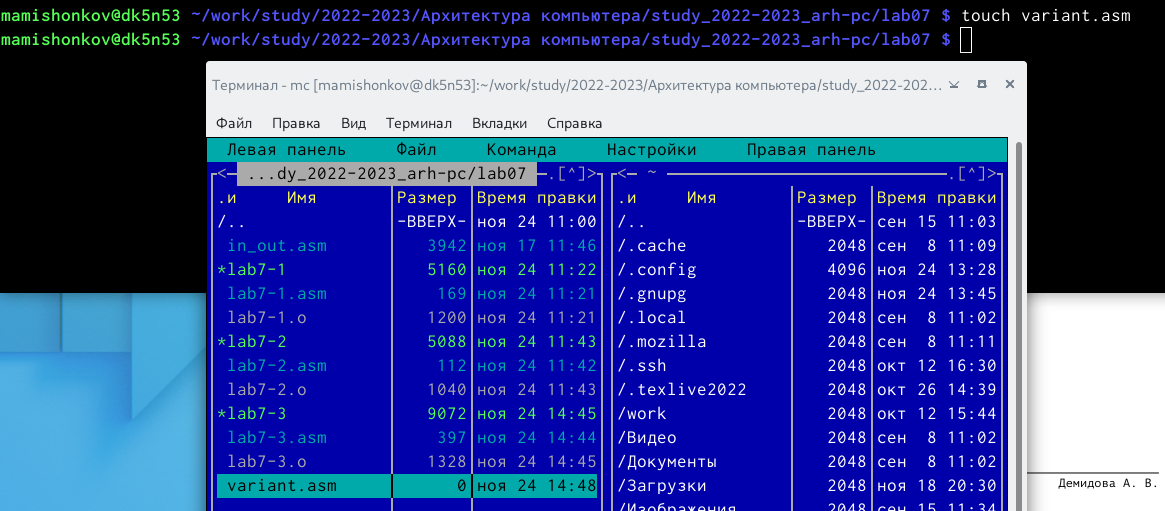


Рис. 18: Создание файла

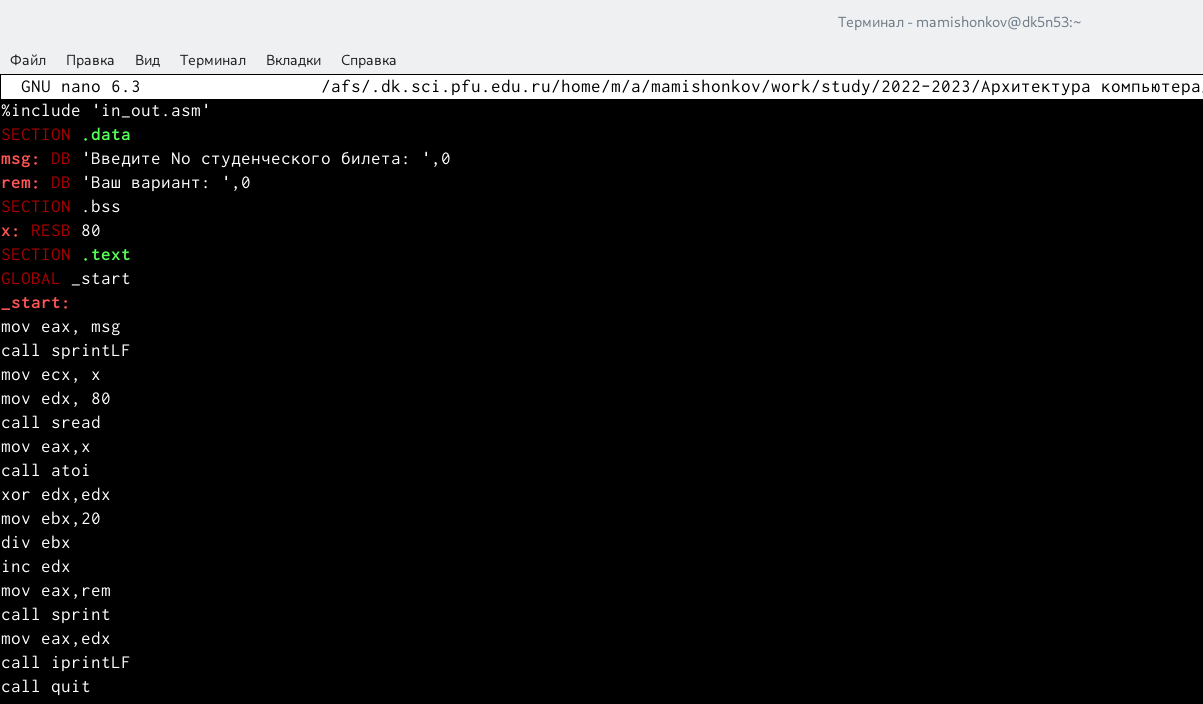


Рис. 19: Введение текста программы

1. Проверив работу исполняемого файла, я ввёл номер моего студенческого билета. Программа вывела ответ 14. Я проверил это аналитически, ответ совпал. (рис. 20)

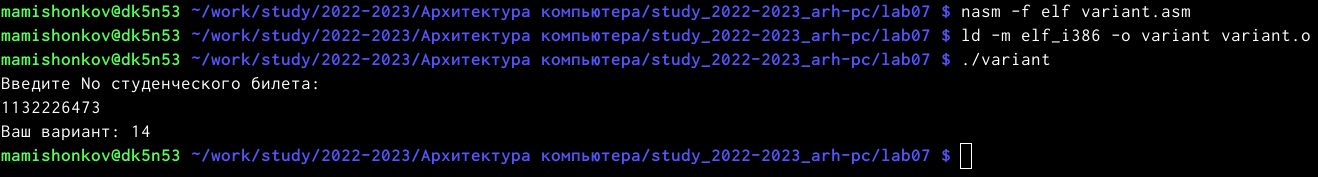


Рис. 20: Проверка работы исполняемого файла

**Вопросы**

1. Какие строки из листинга 7.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’?

Строки mov eax, rem call sprint.

1. Для чего используются инструкции nasm “mov ecx, x”, “mov edx, 80”, “call sread”?

Инструкция “mov ecx, x” записывает адресы выводимого сообщения в ‘EAX’ Инструкция “mov edx, 80” записывает длину вводимого сообщения в ‘EBX’ Инструкция “call sread” выполняет вызов программы ввода сообщений.

1. Для чего используется инструкция “call atoi”?

Эта инструкция используется для преобразования кода переменной ASCII в число.

1. Какие строки из листинга 7.4 отвечают за вычисления варианта?

Строки xor edx, edx mov ebx, 20 div ebx, inc edx.

1. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”?

В регистр ebx.

1. Для чего используется инструкция “inc edx”?

Эта инструкция используется для увеличения значения edx на единицу.

1. Какие строки из листинга 7.4 отвечают за вывод на экран результата вычислений?

Строки mov eax, edx call iprintLF.

**Самостоятельная работа**

1. Создал файл, в котором буду писать программу. (рис. 21)

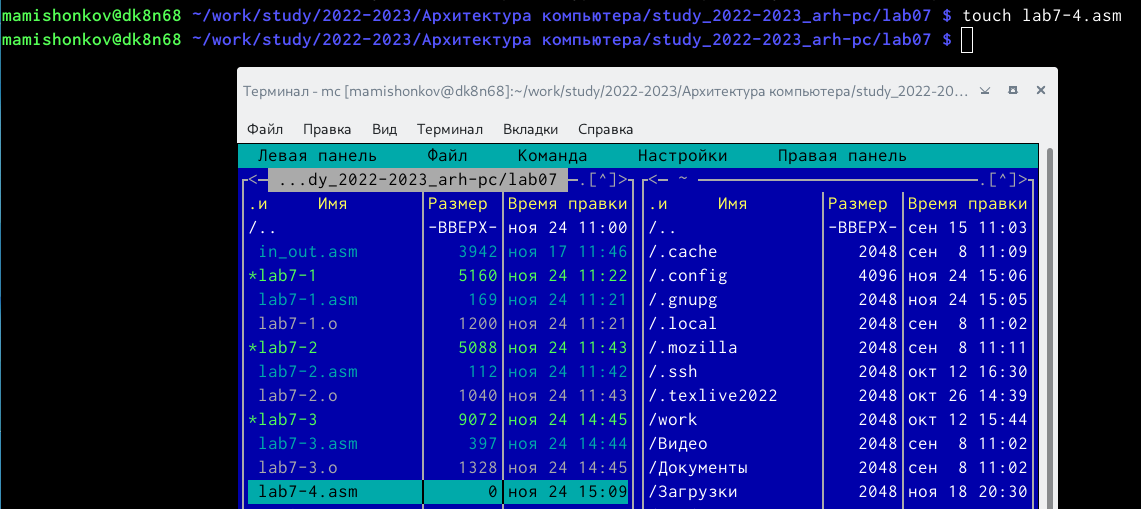


Рис. 21: Создание файла

1. Написал программу, которая будет решать выражение и выводить ответ при разных значениях переменной x. Так как в пункте 17 программа выдала 14 вариант, программу я писал для выражения из списка, соотвествующее варианту 14. (рис. 22,, 23)

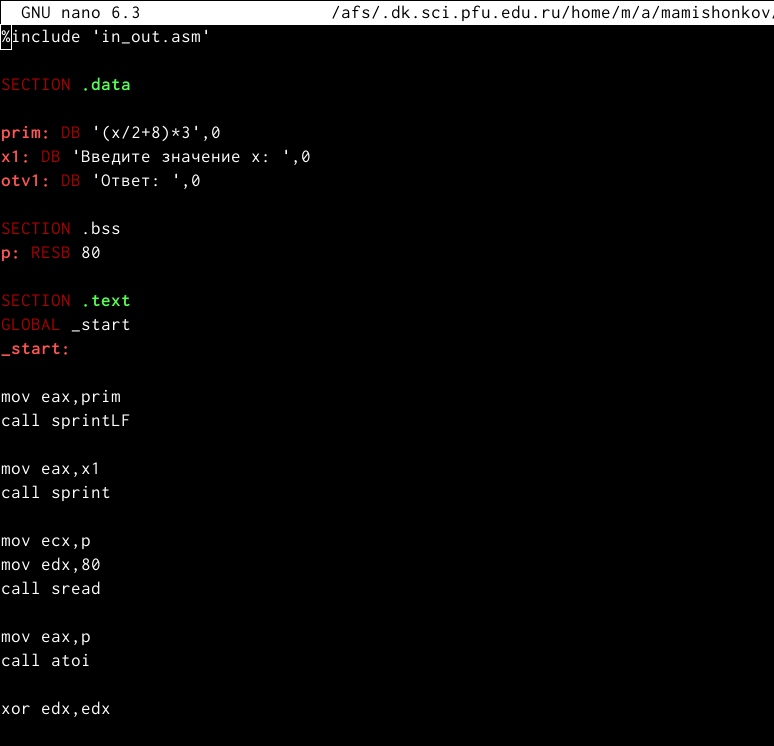


Рис. 22: Написание текста программы

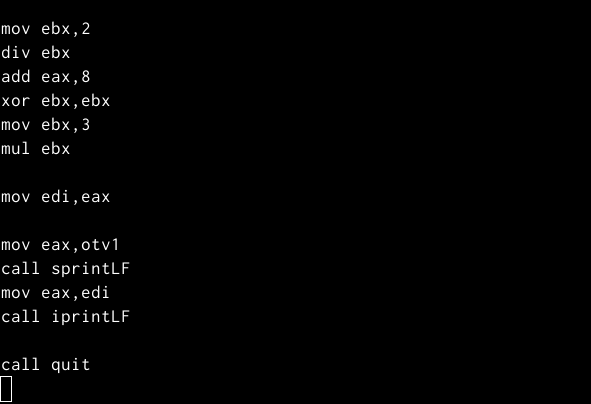


Рис. 23: Написание текста программы

1. Проверил результат работы этой программы. Стоит отметить, что так как при выполнении деления программа в качестве результата использует целую часть, то в первом случае дробь 1/2 округляется до 0, следовательно, был получен ответ 24, а не 25. (рис. 24)

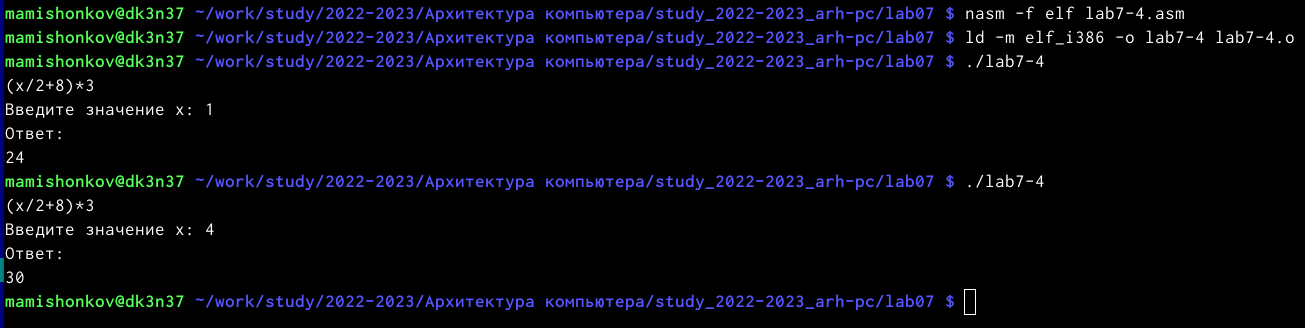


Рис. 24: Проверка работы программы

# 5 Выводы

В ходе выполнения данной лаборатопрной работы я освоил арифметические инструкции языка ассемблера NASM.