Отчет по лабораторной работе № 7

Дисциплина: Архитектура данных

Никулина Ксения Ильинична

Содержание

Список иллюстраций

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM

2 Задание

Освоить арифметические инструкций 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. Далее рассмотрены все существующие способы задания адреса хранения операндов способы адресации. 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

Для выполнения лабораторных работ в файле in_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Это: • iprint – вывод на экран чисел в формате ASCII, перед вызовом iprint в регистр eax необходимо записать выводимое число (mov eax,). • iprintLF – работает аналогично iprint, но при выводе на экран после числа добавляет к символ перевода строки. • atoi – функция преобразует ascii-код символа в целое число и записает результат в регистр eax, перед вызовом atoi в регистр eax необходимо записать число (mov eax,).

4 Выполнение лабораторной работы

1. Создала каталог для программам лабораторной работы №7,перешла в него и создала файл lab7-1.asm (рис. 4.1)

Рис. 4.1: Создание файла lab7-1.asm

2. Ввела в файл lab7-1.asm текст программы из листинга 7.1. (рис. 4.2)

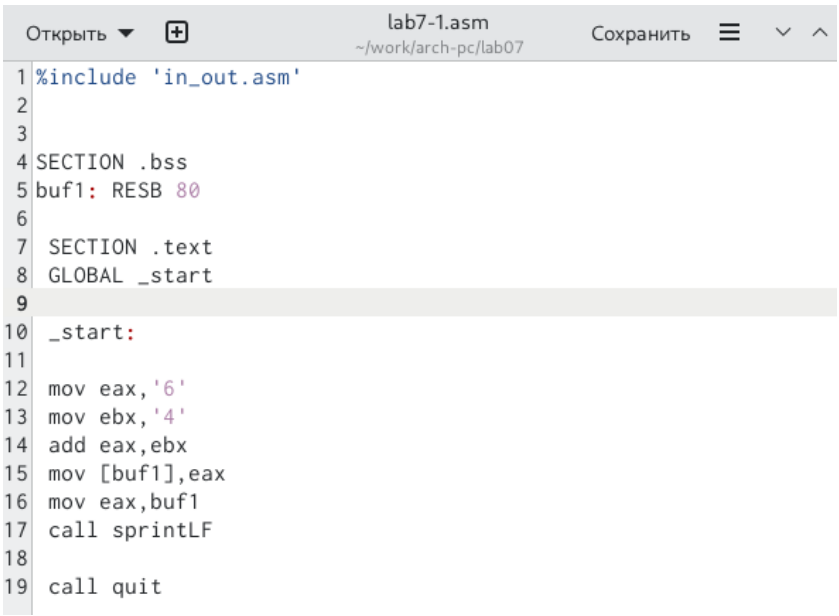

Рис. 4.2: Текст

3. Создала исполняемый файл и запустила его. (рис. 4.3)

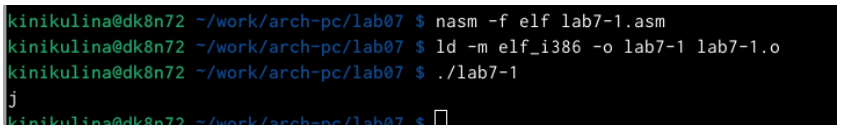

Рис. 4.3: Запуск файла

4. Далее изменила текст программы и вместо символов записал в реги- стры числа. (рис. 4.4)

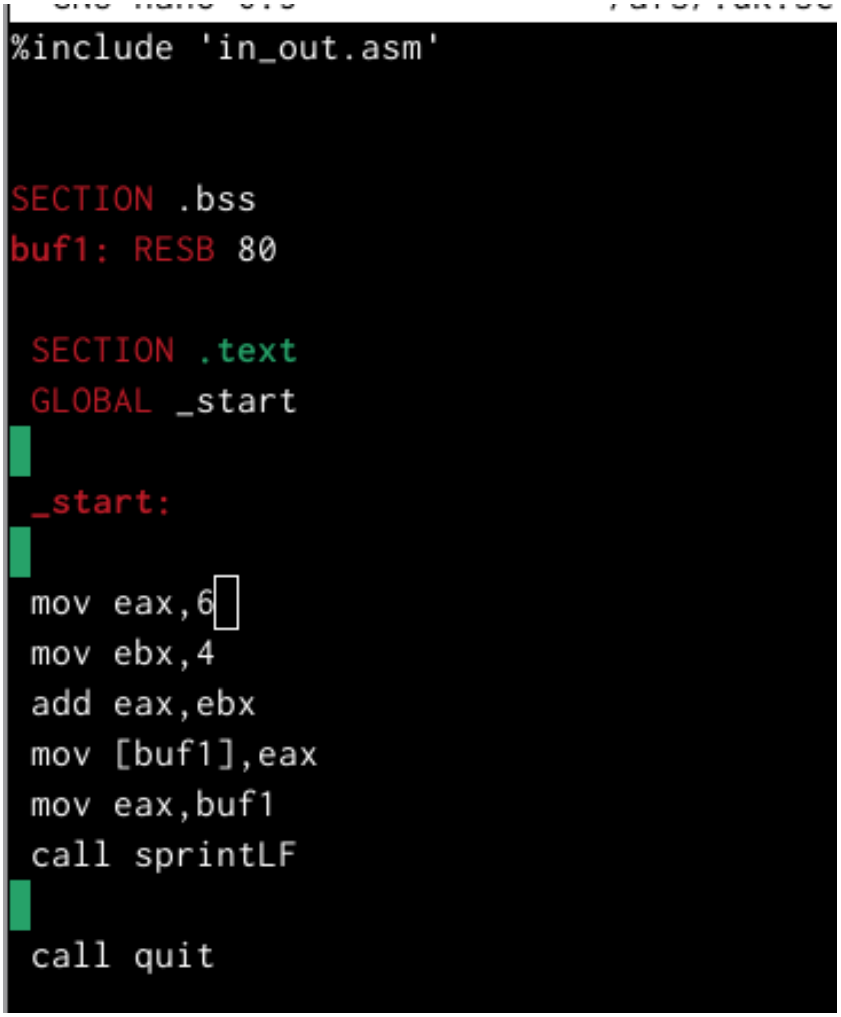

Рис. 4.4: Замена символов в тексте

5. Создала исполняемый файл и запустила его. (рис. 4.5)

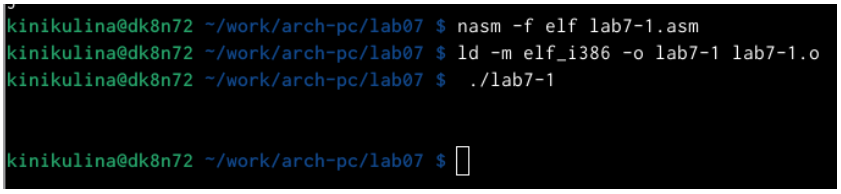

Рис. 4.5: Запуск файла

6. Создала файл lab7-2.asm (рис. 4.6)

Рис. 4.6: Создание файла lab7-2.asm

7. Ввела в него текст программы из листинга 7.2 (рис. 4.7)

Рис. 4.7: Текст

8. Создала исполняемый файл и запустила его. (рис. 4.8)

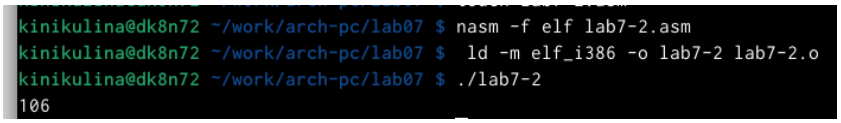

Рис. 4.8: Запуск файла

9. Аналогично предыдущему примеру изменила символы на числа. (рис. 4.9)

Рис. 4.9: Замена символов

10. Создала исполняемый файл и запустила его. (рис. 4.10)

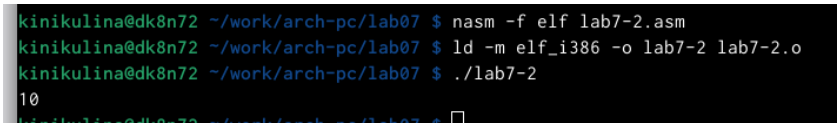

Рис. 4.10: Запуск файла

11. Заменила функцию iprintLF на iprint. (рис. 4.11)

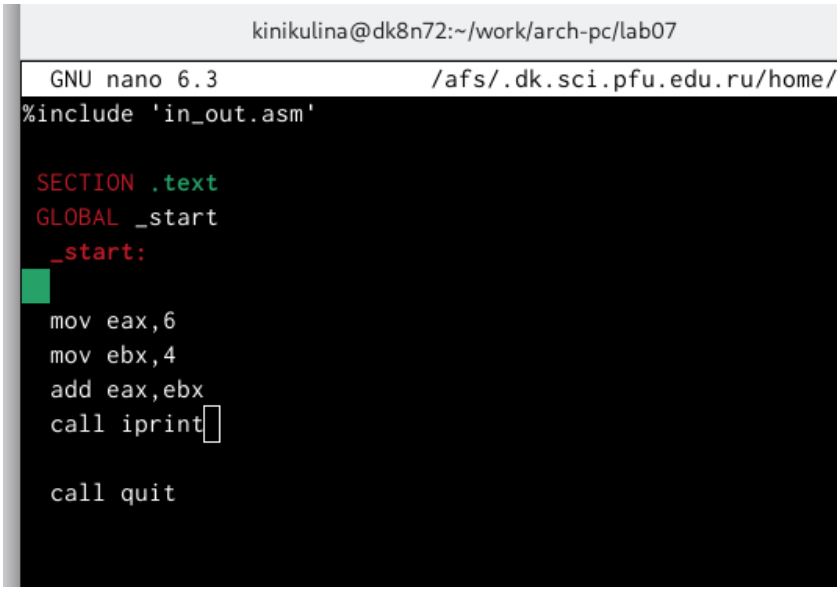

Рис. 4.11: Замена функции

12. Создала исполняемый файл и запустила его. (рис. 4.12)

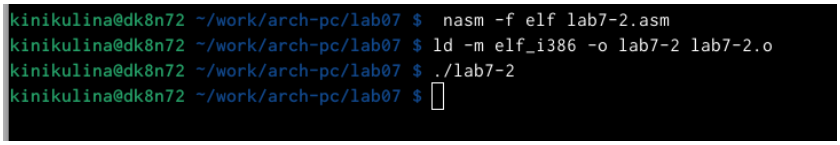

Рис. 4.12: Запуск файла

13. Создала файл lab7-3.asm в каталоге ~/work/arch-pc/lab07 (рис. 4.13)

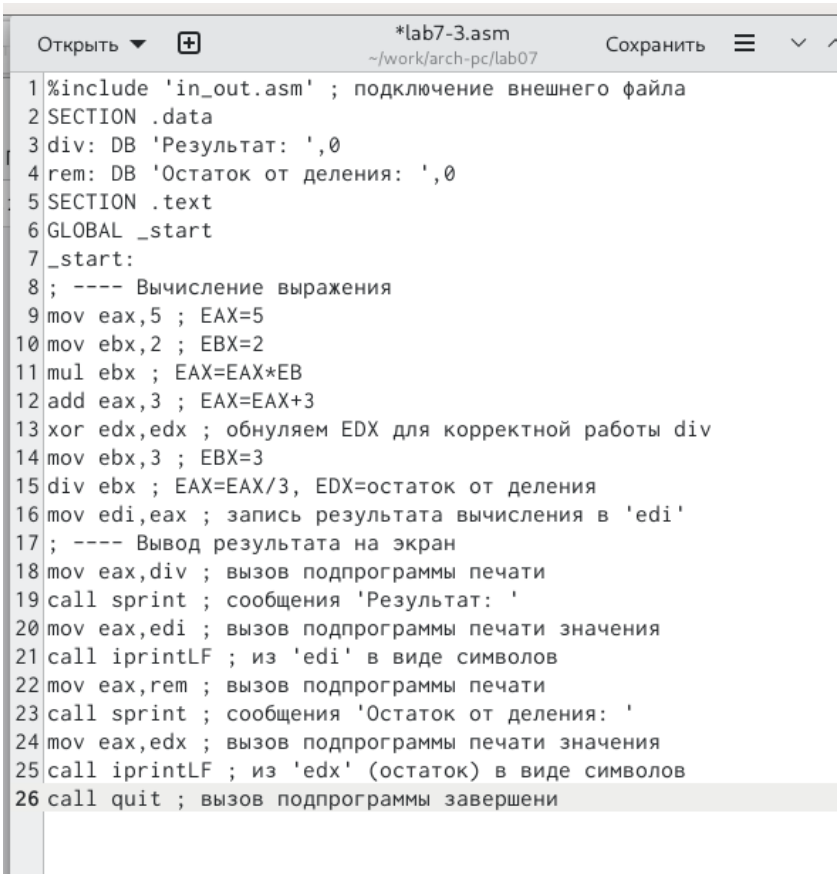

Рис. 4.13: Создание файла lab7-3

14. Создала исполняемый файл и запустила его. (рис. 4.14)

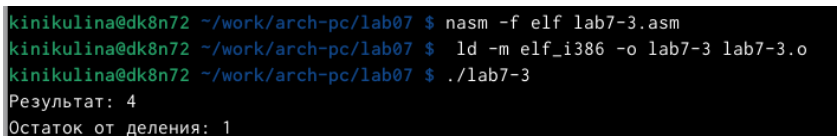

Рис. 4.14: Запуск файла

15. Изменила текст программы для вычисления выражения ⊠(⊠) = (4 ⊠ 6 + 2)/5 (рис. 4.15)

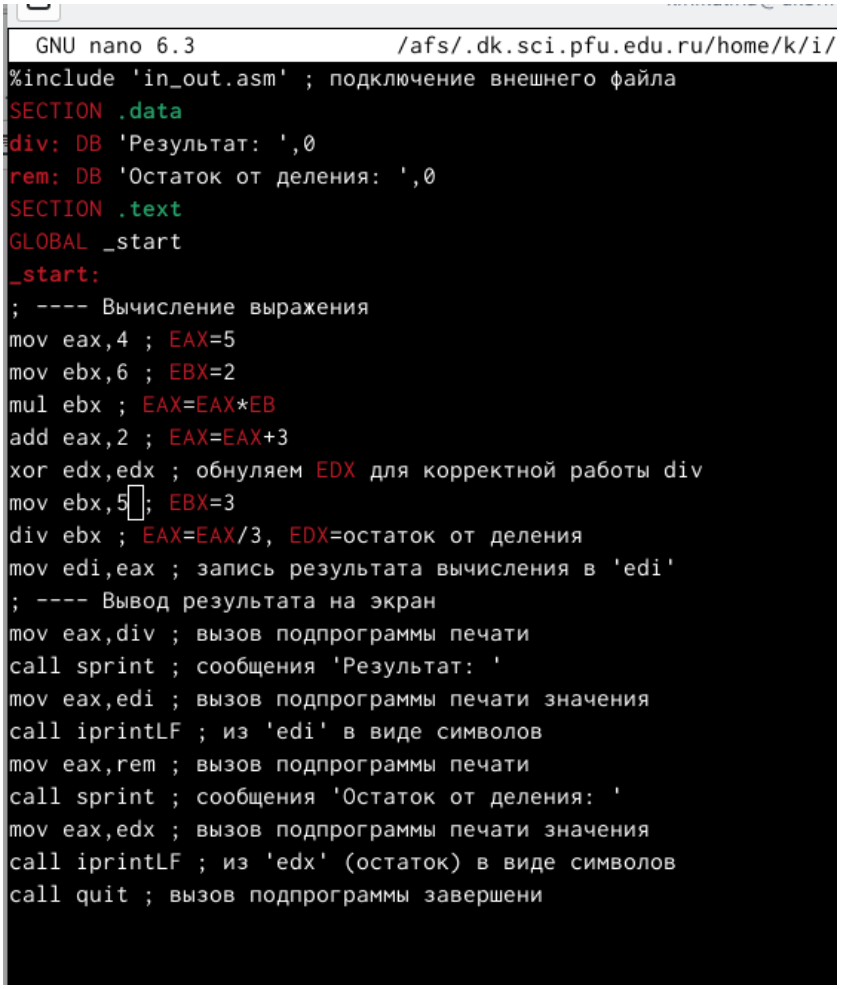

Рис. 4.15: Замена текста

16. Создала исполняемый файл и запустила его. (рис. 4.16)

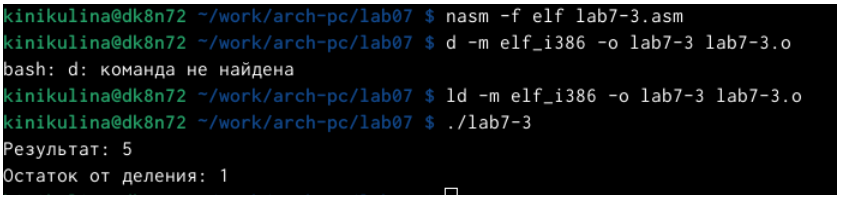

Рис. 4.16: Запуск файла

17. Создала файл variant.asm в каталоге ~/work/arch-pc/lab07 (рис. 4.17)

Рис. 4.17: Создание файла

18. Ввела в файле текст (рис. 4.18)

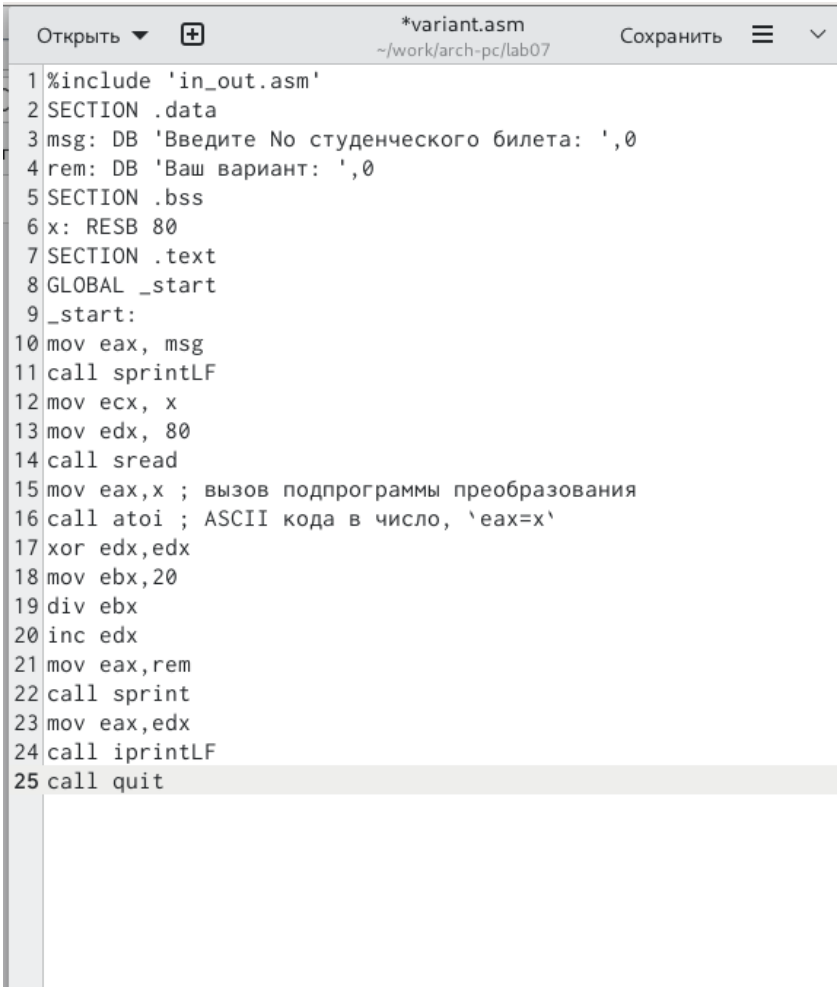

Рис. 4.18: Текст

19. Создала исполняемый файл и запустила его. (рис. 4.19)

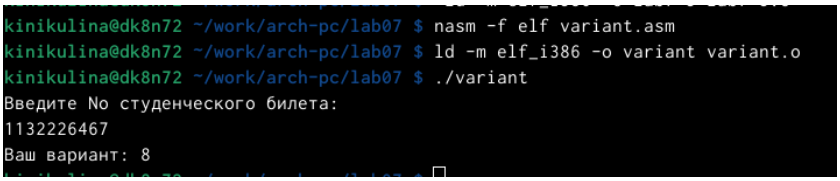

Рис. 4.19: Запуск файла

5 Самостоятельная работа

1. Создала файл для самостотельной работы (рис. 5.1)

Рис. 5.1: Создание файла

2. Ввела программу, которая будет решать уравнение (11 + ⊠) ⊠ 2 − 6 (рис. 5.2)

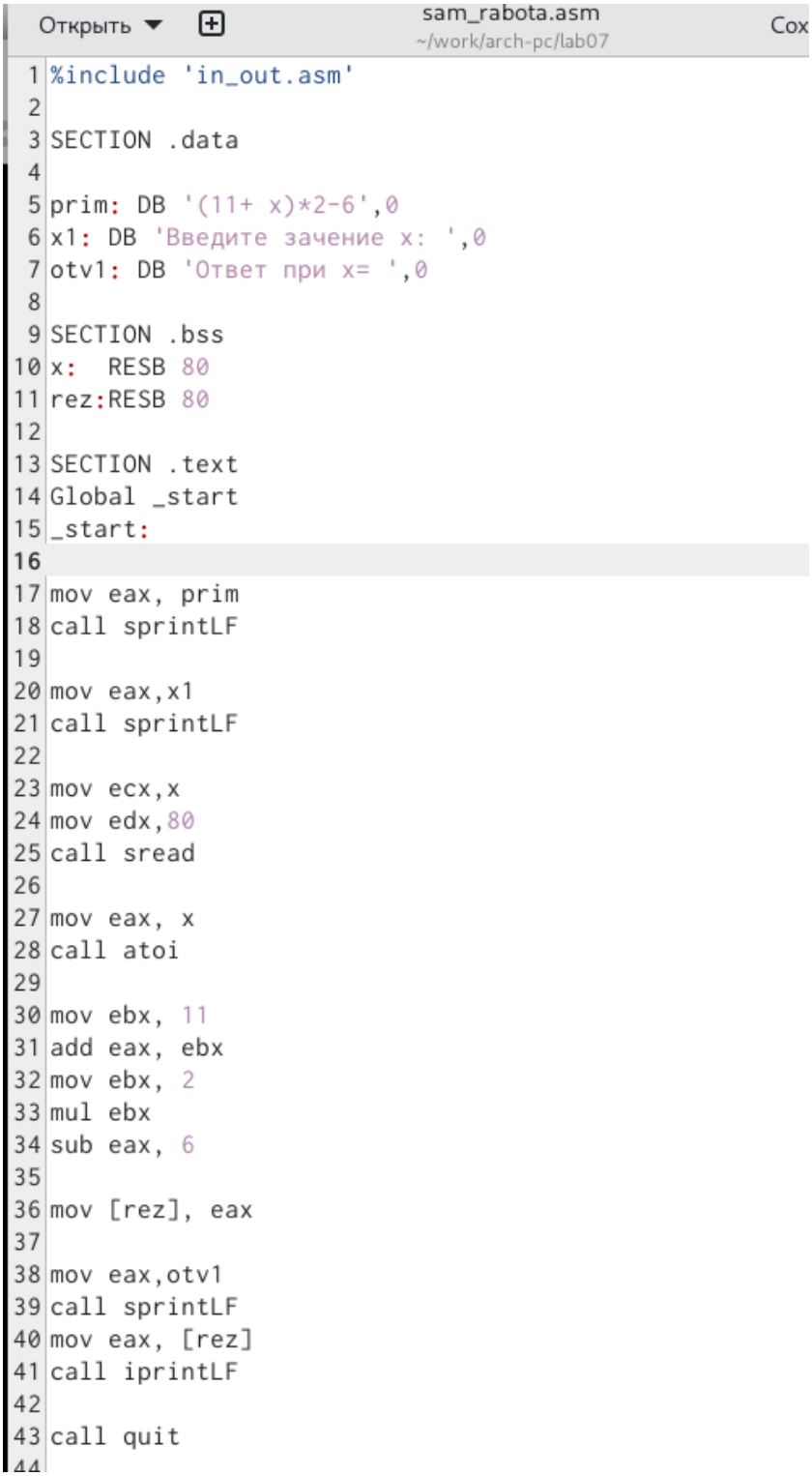

Рис. 5.2: Текст

3. Создала исполняемый файл и запустила его. (рис. 5.3, рис. 5.4)

Рис. 5.3: Ответ при x = 1

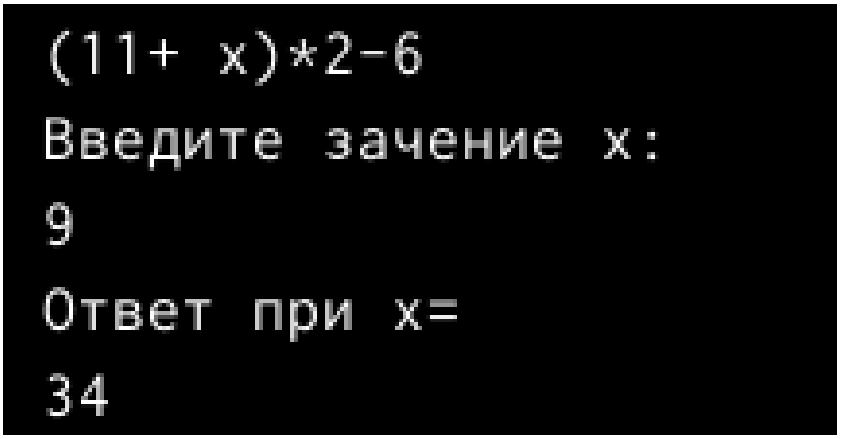

Рис. 5.4: Ответ при x = 9

6 Ответы на впросы

1. Какие строки листинга 7.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’? mov eax,msg call sprintLF
2. Для чего используется следующие инструкции? nasm mov ecx, x mov edx, 80 call sread Эти инструкции используются для ввода переменной X с клавиатуры и сохранения введенных данных.
3. Для чего используется инструкция “call atoi”? Эта инструкция используется для преобразования Кода переменной ASCII в число.
4. Какие строки листинга 7.4 отвечают за вычисления варианта? mov ebx,20 div ebx inc edx
5. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”? В регистре ebx.
6. Для чего используется инструкция “inc edx”? Для увеличения значения edx на 1.
7. Какие строки листинга 7.4 отвечают за вывод на экран результата вычислений? mov eax,edx call iprintLF

7 Выводы

Я освоила арифметические инструкции языка ассемблера NASM