**Лабораторная работа № 1**

**Тема:** Арифметические действия в языке ассемблера.

**Цель:** Отработка навыков формирования операций умножений и делений на основе операций сложения и вычитания.

**Теория:**

**Флаг** – это один или несколько битов памяти, которые могут принимать двоичные значения (или комбинации значений) и характеризуют состояние какого-либо объекта. Обычно флаг может принимать одно из двух логических значений.

**CF (Carry Flag)** - Флаг переноса. Устанавливается в 1, если результат предыдущей операции не уместился в приёмнике и произошёл перенос из старшего бита или если требуется заём (при вычитании). Иначе установлен в 0.

**PF (Parity Flag)** - Флаг чётности. Устанавливается в 1, если младший байт результата предыдущей команды содержит чётное количество битов, равных 1. Если количество единиц в младшем байте нечётное, то этот флаг равен 0.

**AF (Auxiliary Carry Flag)**- Вспомогательный флаг переноса (или флаг полупереноса). Устанавливается в 1, если в результате предыдущей операции произошёл перенос (или заём) из третьего бита в четвёртый. Этот флаг используется автоматически командами двоично-десятичной коррекции.

**SF (Sign Flag)** - Флаг знака. Этот флаг всегда равен старшему биту результата.

**TF (Trap Flag)** - Флаг трассировки (или флаг ловушки). Он был предусмотрен для работы отладчиков в пошаговом выполнении, которые не используют защищённый режим. Если этот флаг установить в 1, то после выполнения каждой программной команды управление временно передаётся отладчику (вызывается прерывание 1).

**IF (Interrupt Enable Flag)** - Флаг разрешения прерываний. Если сбросить этот флаг в 0, то процессор перестанет обрабатывать прерывания от внешних устройств. Обычно его сбрасывают на короткое время для выполнения критических участков программы.

**DF (Direction Flag)** - Флаг направления. Контролирует поведение команд обработки строк. Если установлен в 1, то строки обрабатываются в сторону уменьшения адресов, если сброшен в 0, то наоборот.

**OF** **(Overflow Flag)** - Флаг переполнения. Устанавливается в 1, если результат предыдущей арифметической операции над числами со знаком выходит за допустимые для них пределы. Например, если при сложении двух положительных чисел получается число со старшим битом, равным единице, то есть отрицательное. И наоборот.

**Задание**:

Реализовать умножение двух чисел через сложение двумя способами, реализовать деление через вычисление через флаг carry (cf).

**Примечание:** Начальная загрузка регистра. Выгрузка результата mov в 1 ячейке оперативной памяти

**Код программы:**

**Умножение (вариант 1)**

1. MOV AX, 0
2. MOV BX, 0
3. MOV CX, 0
4. MOV DX, 0
5. MOV AX, 1487
6. MOV BX, 4
7. ADD DX, AX
8. INC CX
9. CMP BX, CX
10. JNZ 7

**Умножение (вариант 2)**

1. MOV AX, 0
2. MOV BX, 0
3. MOV CX, 0
4. MOV DX,0
5. MOV AX, 1487
6. MOV BX, 4
7. ADD DX, AX
8. INC CX
9. CMP BX, CX
10. JZ C
11. JMP 7
12. END

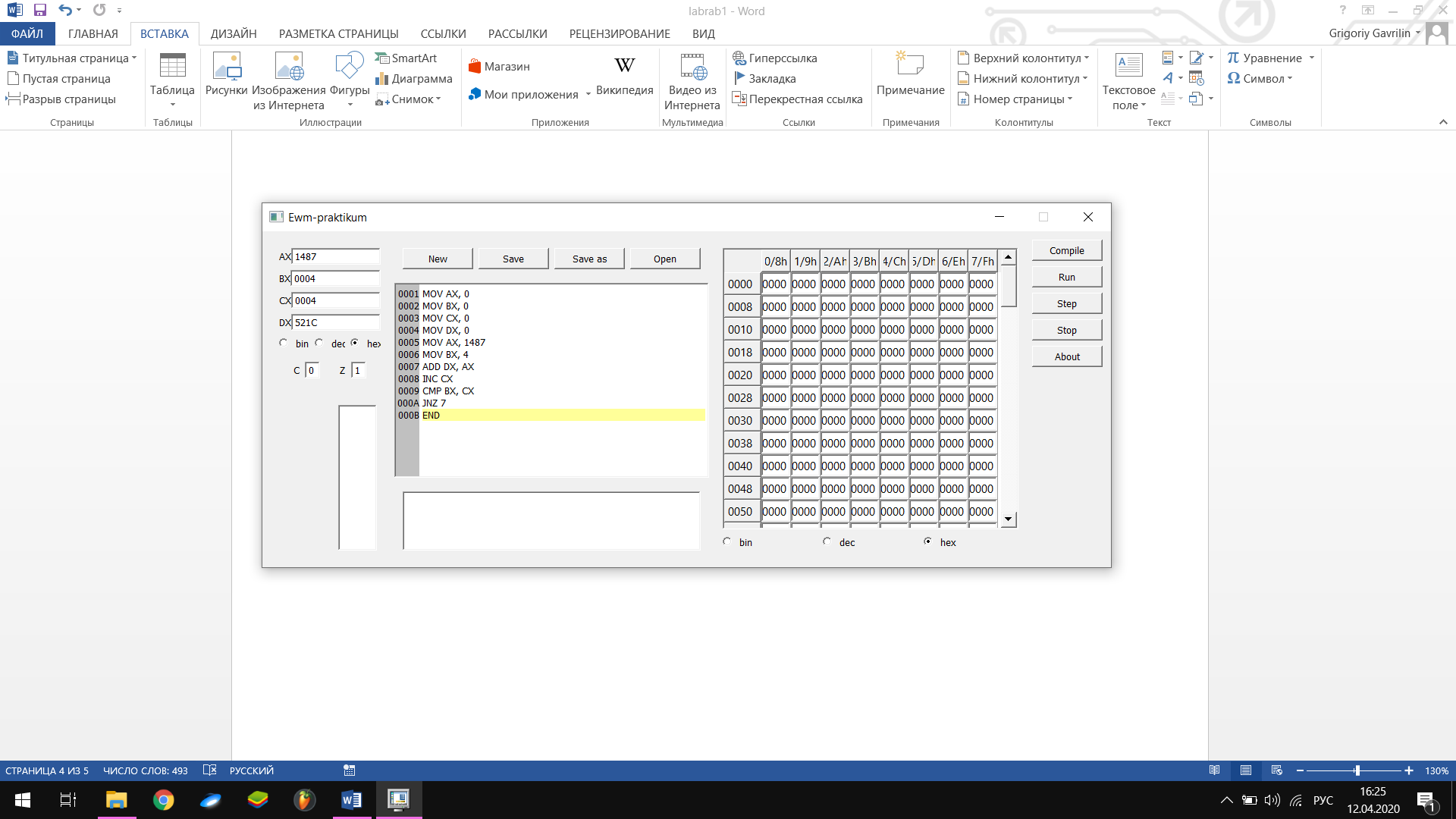
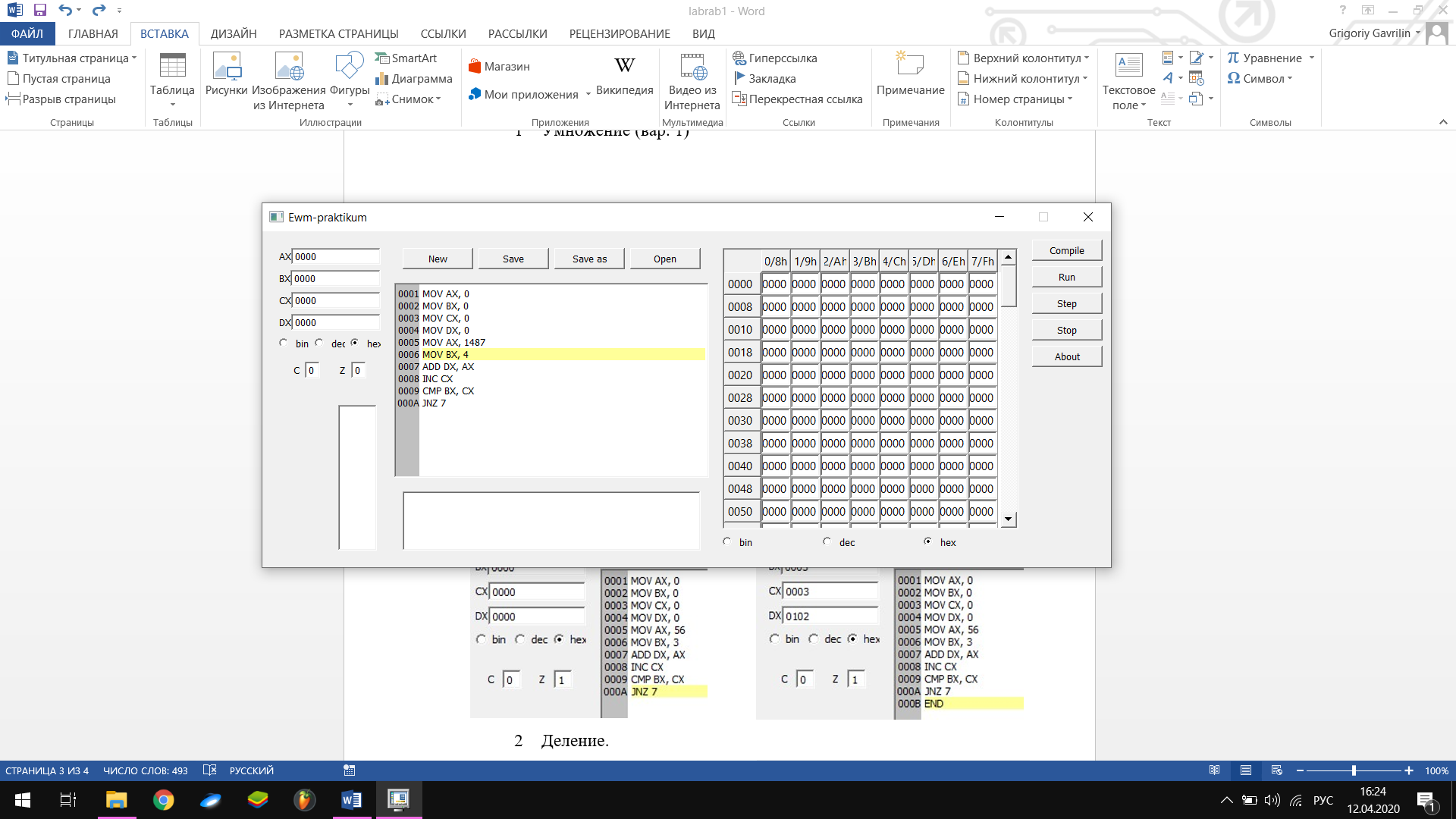
**Деление**

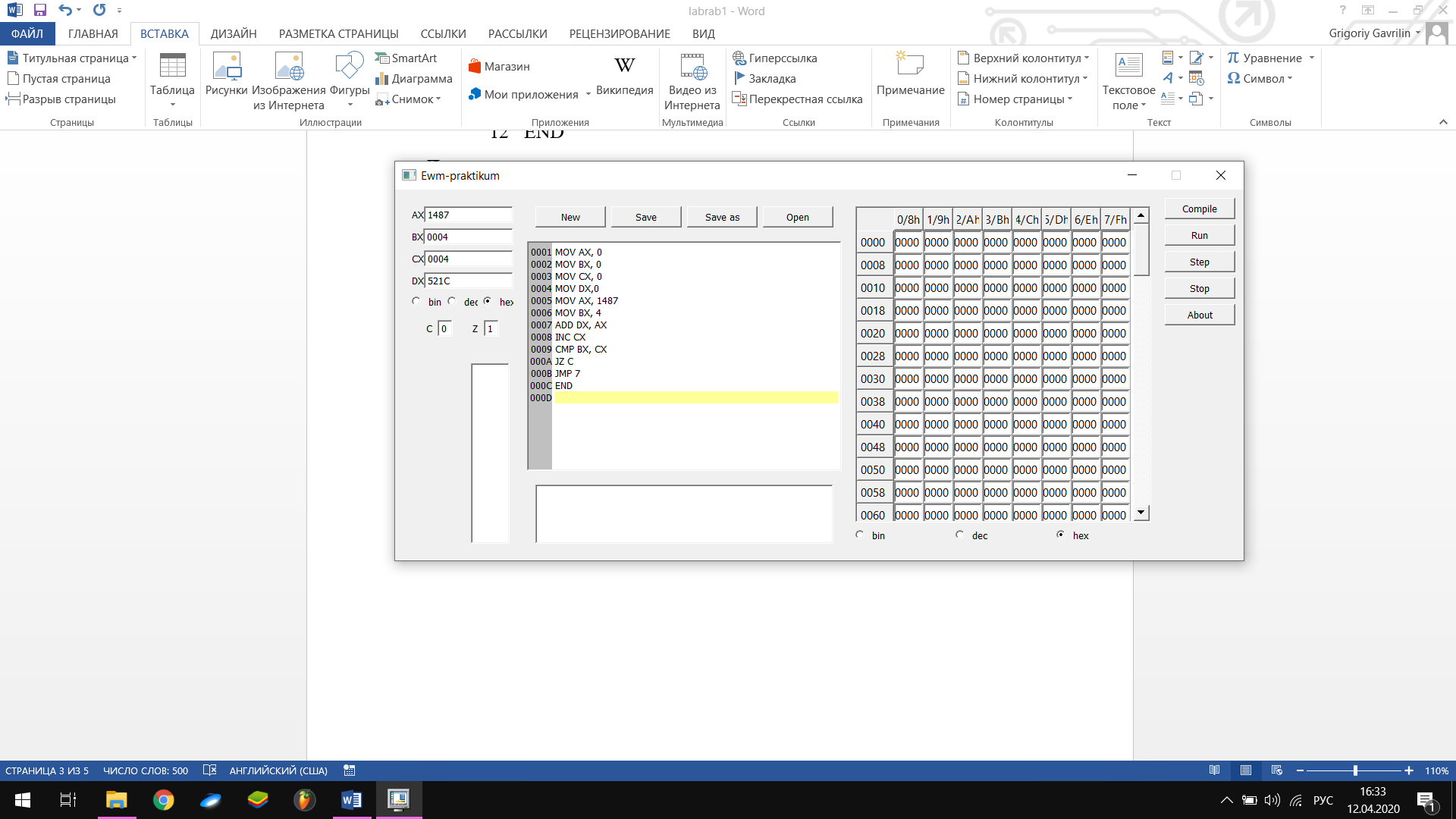
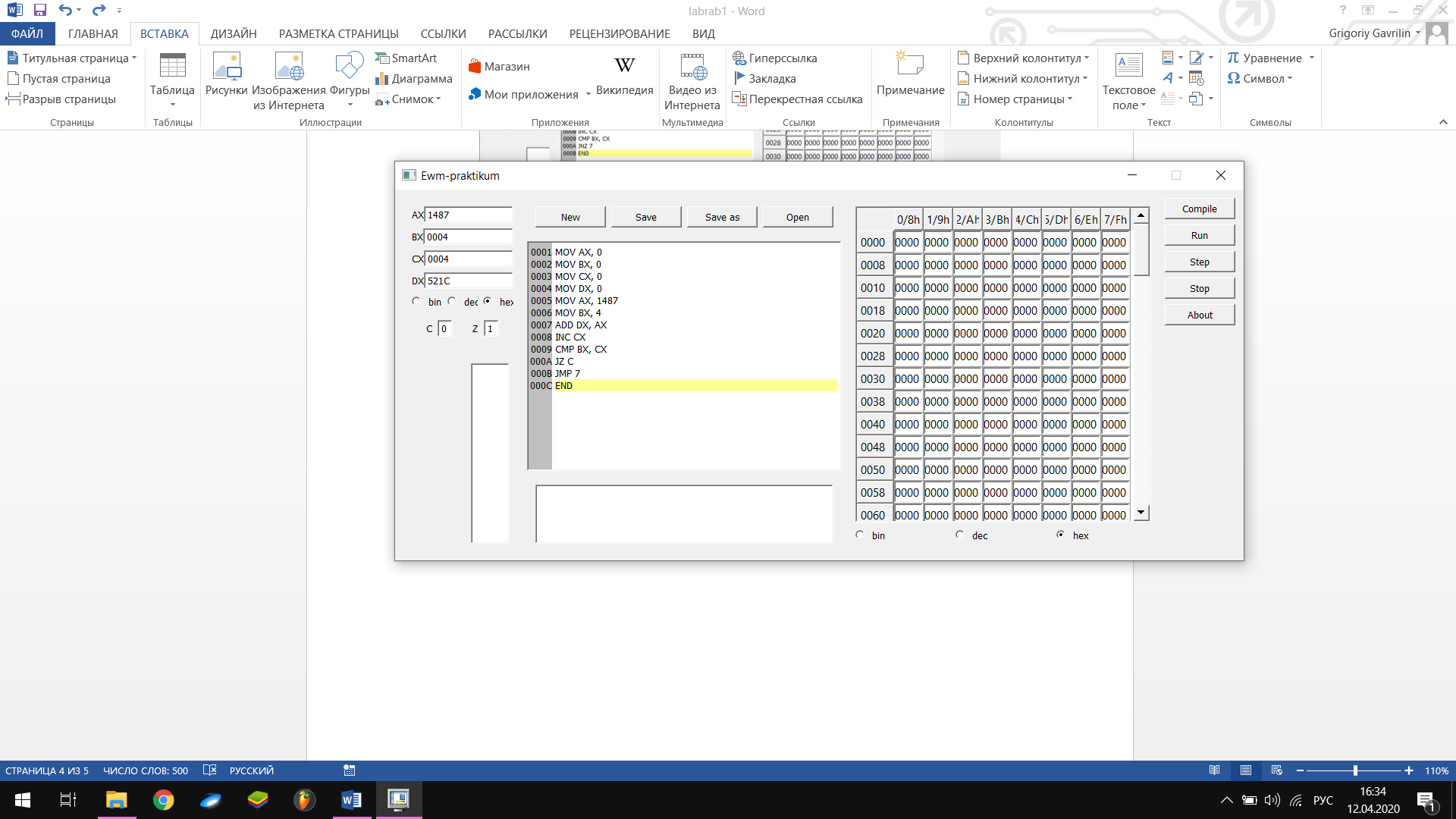
1. MOV AX, 0
2. MOV BX, 0
3. MOV CX, 0
4. MOV DX, 0
5. MOV AX,54
6. MOV DX,9
7. SUB AX,DX
8. INC CX
9. CMP AX,DX
10. JNC 7
11. END

**Ход работы**

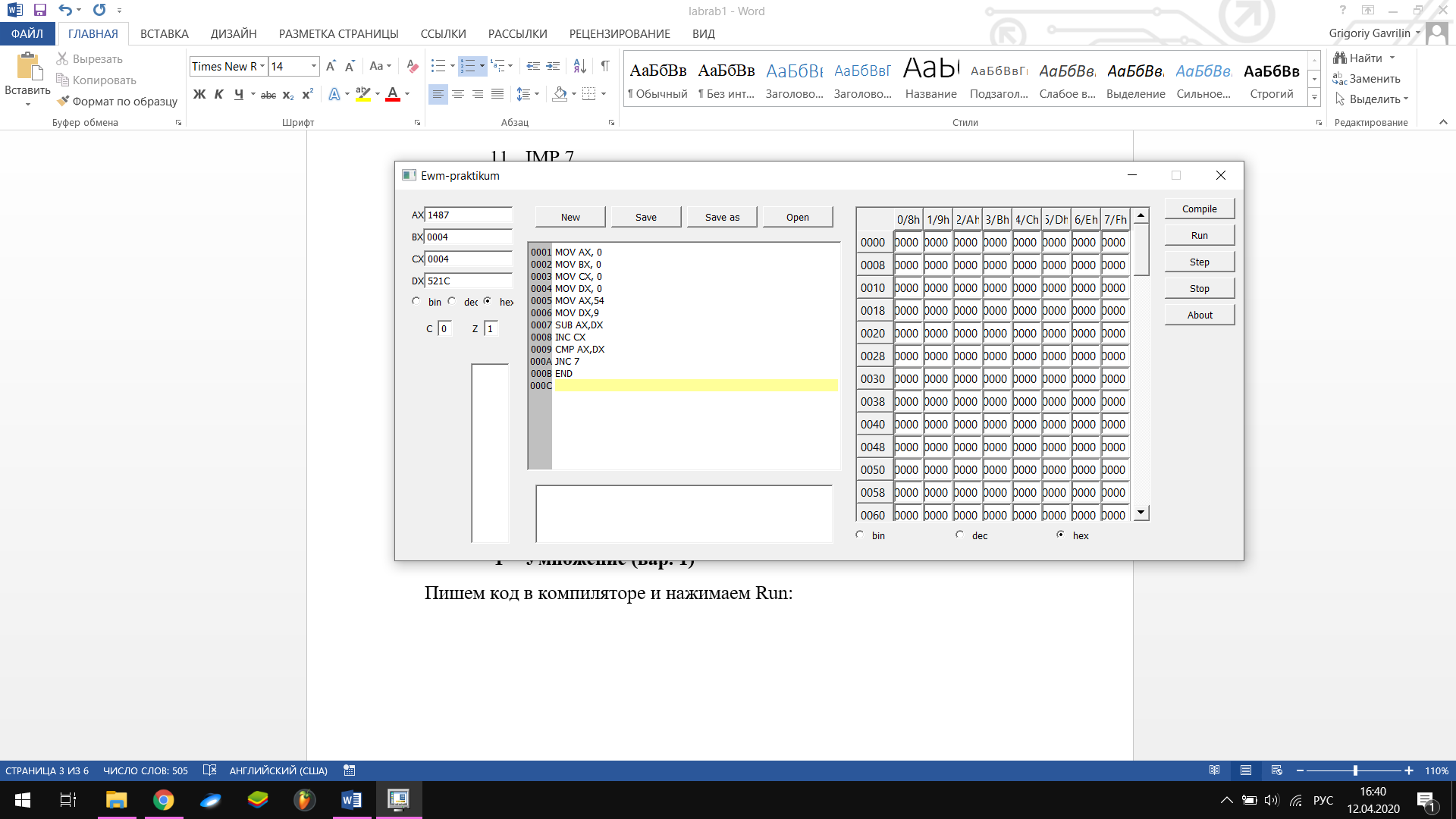
1. **Умножение (вар. 1)**

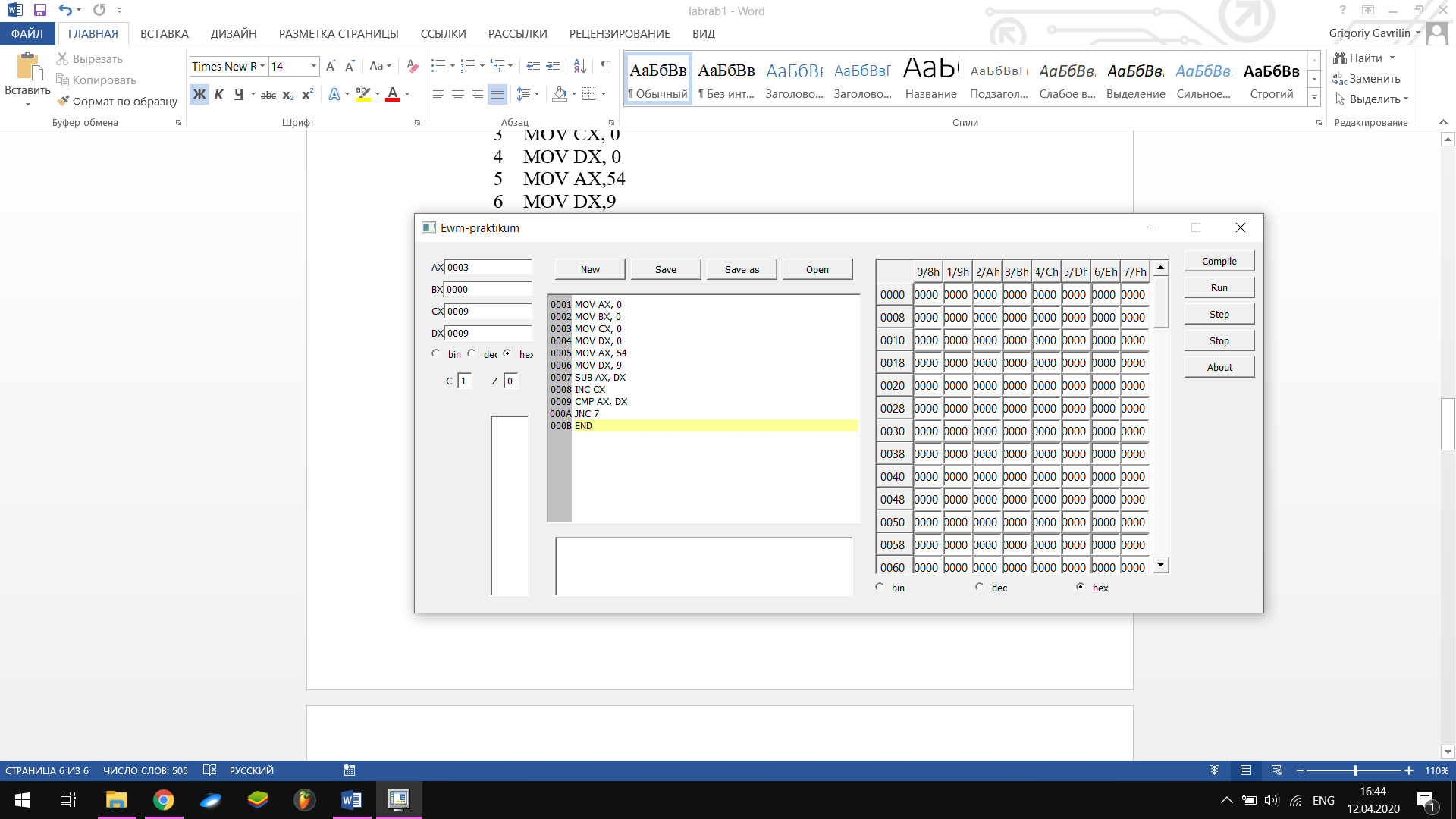
Пишем код в компиляторе и нажимаем Run:



**Умножение (вар. 2)**   Результаты в обоих случаях совпадают.

1. **Деление**





**Программа производит вычисления правильно**

**Вывод:**

Навыки формирования операций умножений и делений на основе операций сложения и вычитания были успешно мной отработаны.