МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»

(РУТ (МИИТ)

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

Отчет по лабораторной работе №2

«Арифметические команды»

по дисциплине

«Машинно-ориентированные языки программирования»

Выполнил: студент ТКИ-341 Козлов А. Д.

Проверили: Доцент “УиЗИ” Логинова Л.Н.

Зав. лаборатории “УиЗИ” Антонов Д. А.

Москва 2024 г.

Цель работы: изучение арифметических команд и получение навыка работы с ними.

Постановка задачи:

Занести числа в память

a= 5'678'901;
b= 6'789'012;
c= 7'890'123;
d= 8'901'234;
e= 9'012'345;
f=10'123'456.

Произвести операции над числами a,b,c,d,e,f по вариантам, заданным в таблице. Результат вывести в память и распечатать в *Memo*.

Номер варианта	Выражение
25	$\frac{b + cb - \frac{a}{4}}{ab - 1}$

Код программы:

```
#include<iostream>
#include<conio.h>
#include<iomanip>
#include<sstream>

using namespace std;
const int DataSize = 144;

string IntToHex(int n)
{
	stringstream ss;
	ss << hex << n;
	return ss.str();
}

// 1)
//     b + c*b - (a/4) = num1
//       3   1   4   2

// 2)
//     a*b - 1 = num2
//      1   2

// 3)
// num1/num2 = num3

int main()
{
	int32_t a = 5678901; // 0x56A735
	int32_t b = 6789012; // 0x679794
	int32_t c = 7890123; // 0x7864CB
	int32_t d = 8901234; // 0x87D272
	int32_t e = 9012345; // 0x898479
	int32_t f = 10123456; // 0x9A78C0

	unsigned char Memo[DataSize];
	for (int i = 0; i++; i < DataSize) {
		Memo[i] = NULL;
	}

	__asm {
		PUSHAD

		LEA EDI, Memo // Храним в регистре EDI адресс первого байта Memo

		// 1)-1
		MOV EAX, c
		MUL b
		MOV DWord Ptr [EDI], EAX // заносим в Memo младшую часть произведения
		MOV DWord Ptr [EDI + 4], EDX // заносим в Memo старшую часть
произведения

		// 1)-2
		MOV EAX, a // хочу поменять на пару регистров DX:AX, пара EDX:EAX
избыточна
		xor EDX, EDX
		MOV EBX, 4
		DIV EBX
		MOV DWord Ptr [EDI + 2 * 4], EAX // !

		// 1)-3
		MOV EAX, b // младшая часть b
		MOV EDX, 0 // старшая чать b
```

```
            MOV ECX, DWord Ptr [EDI] // младшая часть пунтка 1)-1
            MOV EBX, DWord Ptr [EDI + 4] // старшая часть пункта 1)-1

            ADD EAX, ECX
            ADC EDX, EBX

            // 1)-4
            MOV ECX, DWord Ptr [EDI + 8] // младшая часть пунтка 1)-2
            MOV EBX, 0 // старшая часть пунтка 1)-2

            SUB EAX, ECX
            SBB EDX, EBX

            MOV DWord Ptr [EDI], EAX
            MOV DWord Ptr [EDI + 4], EDX

            // 2)-1
            MOV EAX, a
            MUL b // младшая часть EAX, старшая EDX

            // 2)-2
            SUB EAX, 1
            SBB EDX, 0

            MOV EBX, EAX
            MOV ECX, EDX

            // 3)
            MOV EAX, DWord Ptr [EDI]
            MOV EDX, DWord Ptr [EDI + 4]

            DIV EBX
            MOV DWord Ptr [EDI], EAX

            POPAD
      }

      for (int i = 0; i < DataSize; i++) {
            if ((i % 16) == 0) {
                  cout << "\n" << setw(2) << i / 16 << ":";
            }
            else {
                  cout << " " << setw(2) << IntToHex(Memo[i - 1]);
            }
      }
      _getch();

      return 0;
}
```

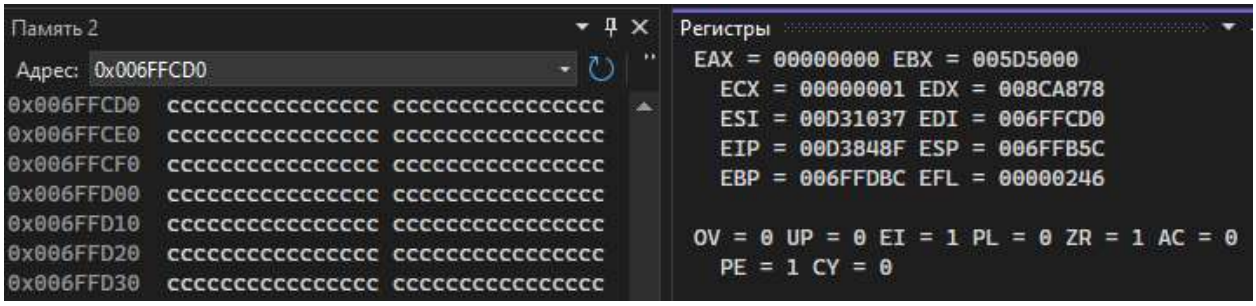
Ход выполнения:

Разобьем выражение $\frac{b+cb-\frac{a}{4}}{ab-1}$ на пункты по приоритету выполнения математических операций.

Пункт 0:

Получаем адрес первого байта Memo:

```
LEA EDI, Memo
```



Пункт 1:

Вычислим числитель дроби, для этого определим порядок выполнения операций:

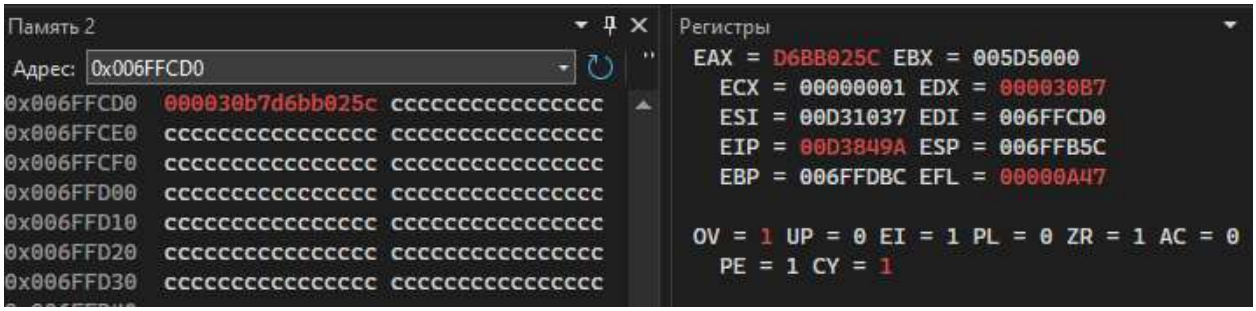
$$b + c * b - \frac{a}{4}$$

$$3 \quad 1 \quad 4 \; 2$$

Операция №1:

Записываем переменную *c* в регистр EAX и перемножаем с числом *b*, результат записываем в память Memo, как два двойных слова:

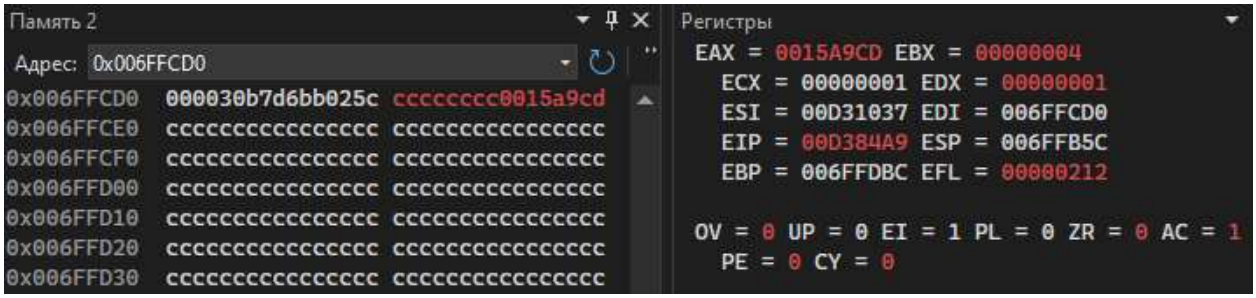
```
MOV EAX, c
MUL b
MOV DWord Ptr [EDI], EAX // заносим в Memo младшую часть произведения
MOV DWord Ptr [EDI + 4], EDX // заносим в Memo старшую часть произведения
```



Операция №2

Записываем переменную *a* в регистр EAX и делим на константу:

```
MOV EAX, a
xor EDX, EDX
MOV EBX, 4
DIV EBX
MOV DWord Ptr [EDI + 8], EAX
```

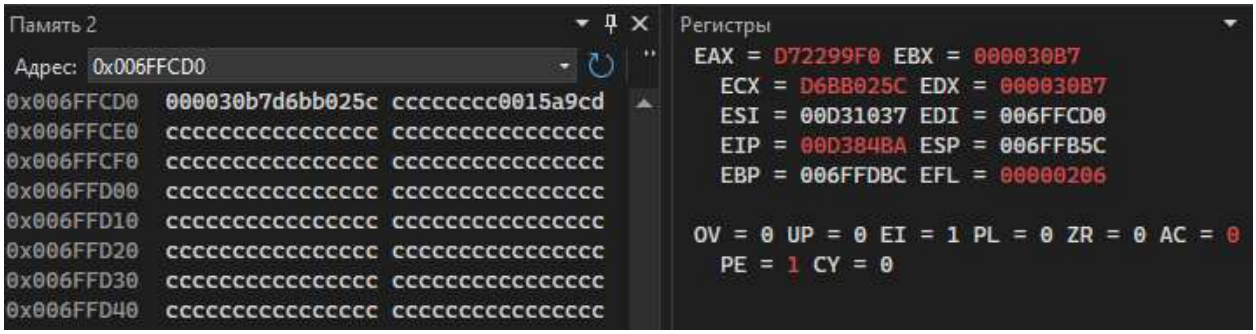


Операция №3

Записываем полученную после операции №1 результат в два регистра ECX и EBX, а переменную *b* записываем в регистр EAX и нулем в EDX (т.к. число *b* 32-битное). Затем складываем эти два числа:

```
MOV EAX, b // младшая часть b
MOV EDX, 0 // старшая чать b
MOV ECX, DWord Ptr [EDI] // младшая часть пунтка 1)-1
MOV EBX, DWord Ptr [EDI + 4] // старшая часть пункта 1)-1

ADD EAX, ECX
ADC EDX, EBX
```



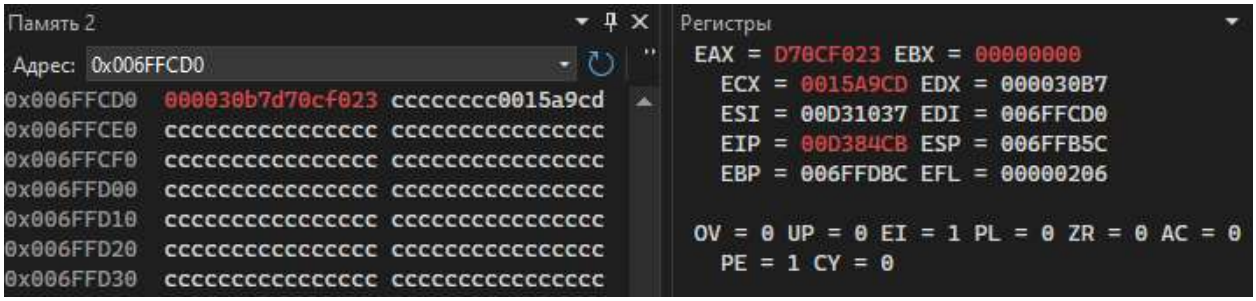
Операция №4

Записываем полученную после операции №2 результат в два регистра ECX и EBX, и вычитаем его из числа, которое осталось в регистрах EAX и EDX после предыдущего пункта:

```
MOV ECX, DWord Ptr [EDI + 8] // младшая часть пунтка 1)-2
MOV EBX, 0 // старшая часть пунтка 1)-2

SUB EAX, ECX
SBB EDX, EBX
```

```
MOV DWord Ptr [EDI], EAX
MOV DWord Ptr [EDI + 4], EDX
```



Пункт 2:

Вычислим знаменатель дроби, для этого определим порядок выполнения операций:

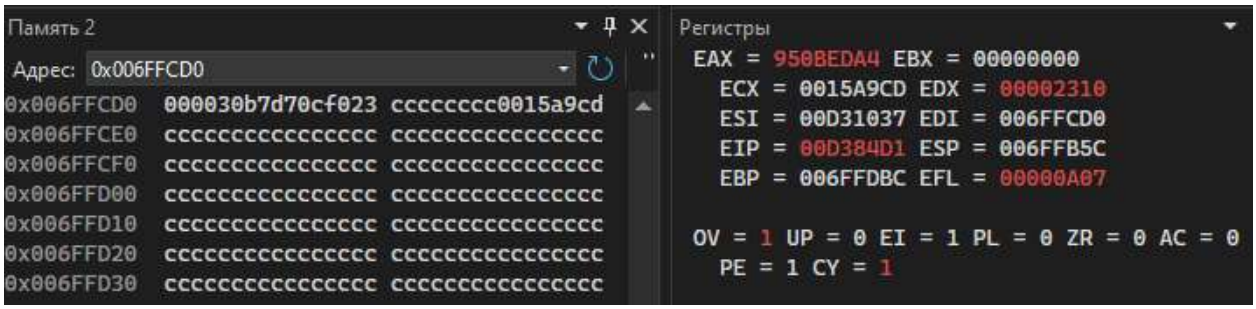
$$a * b - 1$$

$$1 \quad 2$$

Операция №1:

Записываем переменную a в регистр EAX и перемножаем с числом:

```
MOV EAX, a
MUL b // младшая часть EAX, старшая EDX
```

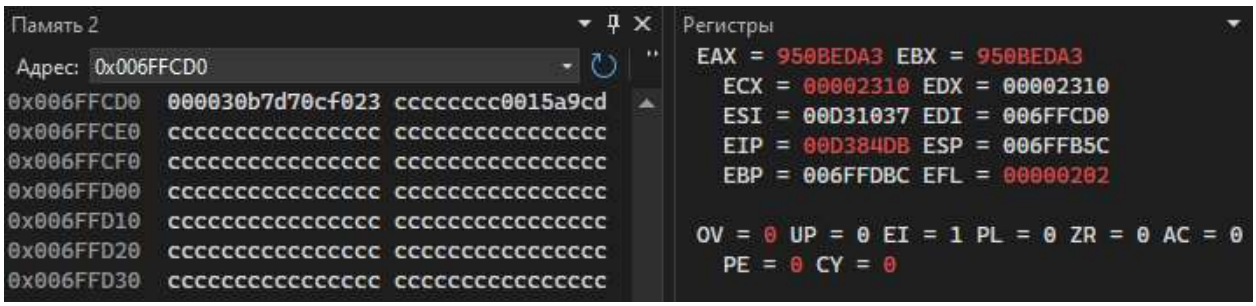


Операция №2

Вычитаем константу из переменной, полученной в результате прошлой операции и сохраняем результат в регистрах EBX (младшая часть) и ECX (старшая часть):

```
SUB EAX, 1
SBB EDX, 0

MOV EBX, EAX
MOV ECX, EDX
```



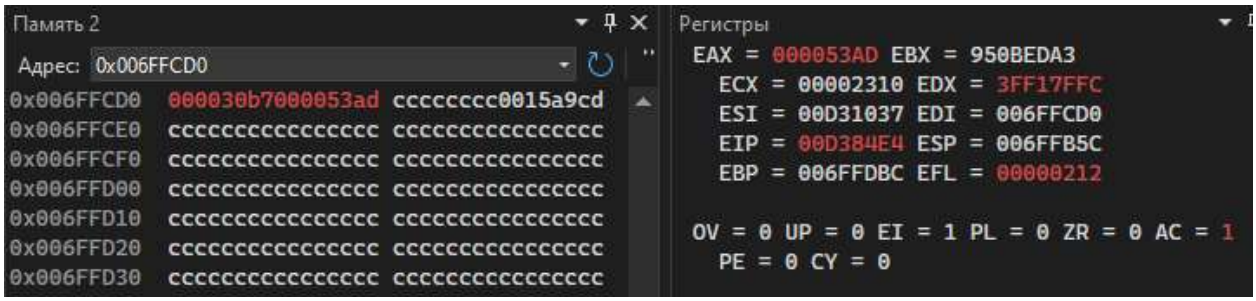
Пункт 3:

Вычислим всю дробь целиком, поделив числитель на знаменатель:

```
MOV EAX, DWord Ptr [EDI]
MOV EDX, DWord Ptr [EDI + 4]

DIV EBX
MOV DWord Ptr [EDI], EAX
```

Результат сохраняем в Memo как двойное слово.



Вывод:

1. В ходе выполнения лабораторной работы были изучены основные арифметические команды языка ассемблера, такие как ADD, SUB, MUL, DIV.
2. Было продемонстрированно вычисление большой дроби с использованием этих команд, а также работа с числами большей разрядности, чем 32-bit.
3. Выяснили, что при работе с большими числами требуется учитывать размер данных и ограничения регистров процессора.
4. Показали, как при вычислении большого числа команды математических операций могу действовать вместе (например, ADD и ADC для учета знака числа и переноса битов из младшей части в старшую и наоборот.

Заключение

Выполнение лабораторной работы расширило знание команд языка ассемблера. Мы убедились, что правильное применение этих инструкций требует учета особенностей работы с регистрами. Полученные навыки будут необходимы при разработке любых вычислительных программ.