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	$\frac{2 * b - \frac{a}{4}}{a - 1}$

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

// 2)
//     a - 1 = num2

// 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 < DataSize; i++) {
		Memo[i] = NULL;
	}

	__asm {
		PUSHAD

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

		// 1)-1
		MOV EAX, b
		MOV EBX, 2
		MUL EBX
		MOV DWord Ptr [EDI], EAX
		MOV DWord Ptr [EDI + 4], EDX

		// 1)-2
		MOV EAX, a
		xor EDX, EDX
		MOV EBX, 4
		DIV EBX
		MOV EBX, EAX
		MOV ECX, 0

		// 1)-3
		MOV EAX, DWord Ptr [EDI]
		MOV EDX, DWord Ptr [EDI + 4]
		SUB EAX, EBX
```

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

            // 2)
            MOV EAX, a
            SUB EAX, 1
            MOV EBX, EAX

            // 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;
}
```

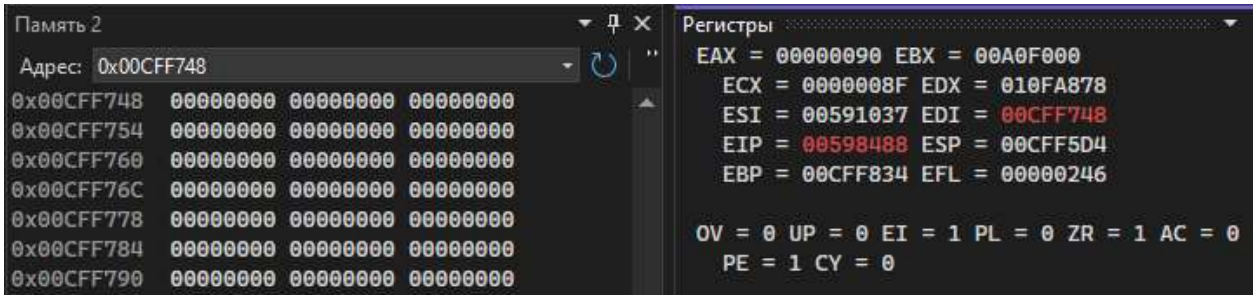
Ход выполнения:

Разобьем выражение $\frac{2*b-\frac{a}{4}}{a-1}$ 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:

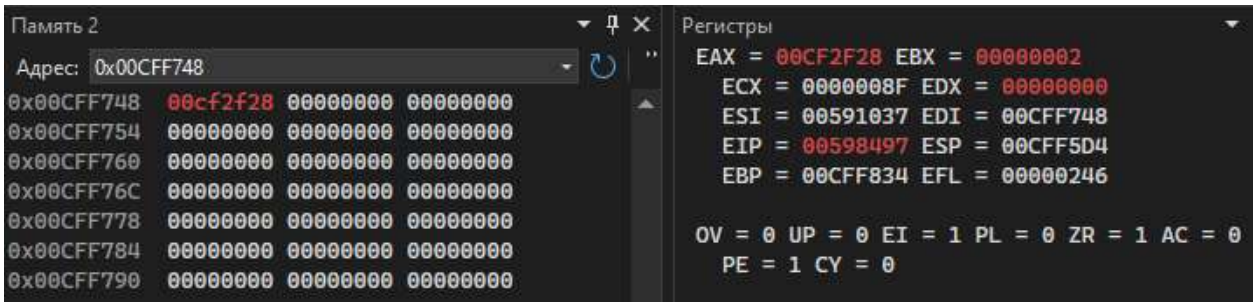
$$2 * b - a/4$$

$$1 \quad 3 \quad 2$$

Операция №1:

Записываем переменную b в регистр EAX и константу в регистр EBX, затем перемножаем с числа в регистрах EAX и EBX, результат записываем в память Memo, как два двойных слова:

```
MOV EAX, b
MOV EBX, 2
MUL EBX
MOV DWord Ptr [EDI], EAX
MOV DWord Ptr [EDI + 4], EDX
```

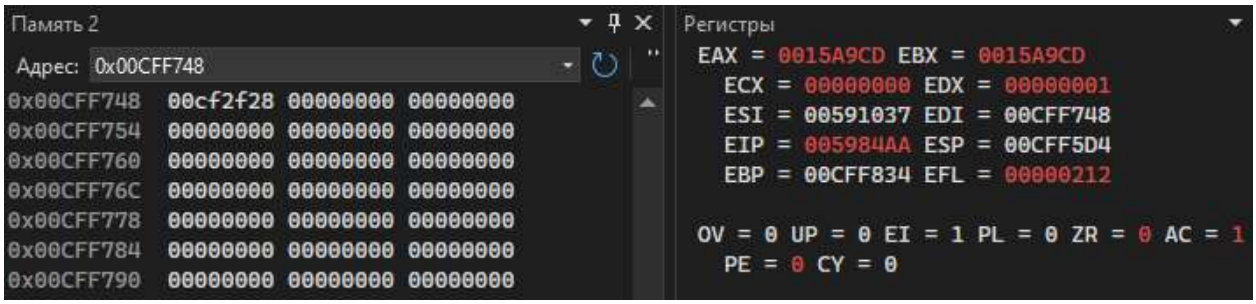


Проверяем: $2 * b = 2 * 6\,789\,012_{10} = 13\,578\,024_{10} = CF2F28_{16}$, такой же результат мы видим в регистре EAX.

Операция №2

Записываем переменную a в регистр EAX, делим на константу и записываем в регистр EBX полученный результат из регистра EAX и инициализируем нулем регистр ECX для дальнейших вычислений:

```
MOV EAX, a
xor EDX, EDX
MOV EBX, 4
DIV EBX
MOV EBX, EAX
MOV ECX, 0
```

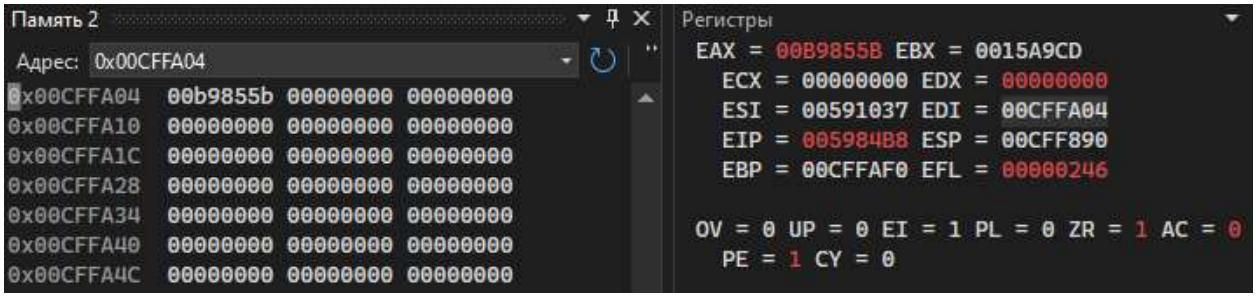


Проверяем: $a/4 = {5\ 678\ 901_{10}}/4 = 1\ 419\ 725_{10}\ (\text{ост.}\ 1) = 15A9CD_{16}$, результат совпадает со значением в регистре EBX.

Операция №3

Записываем в регистры EAX и EDX младшую и старшую часть составной результата из пункта 1 соответственно, затем вычитаем из него число, подготовленное в предыдущем пункте (EBX младшая часть, ECX старшая). Сохраняем результат в память Memo:

```
MOV EAX, DWord Ptr [EDI]
MOV EDX, DWord Ptr [EDI + 4]
SUB EAX, EBX
SBB EDX, ECX
MOV DWord Ptr [EDI], EAX
MOV DWord Ptr [EDI + 4], EDX
```



Проверяем: $13\ 578\ 024_{10} - 1\ 419\ 725_{10} = 12\ 158\ 299_{10} = B9855B_{16}$, тот же результат мы видим в регистре EAX.

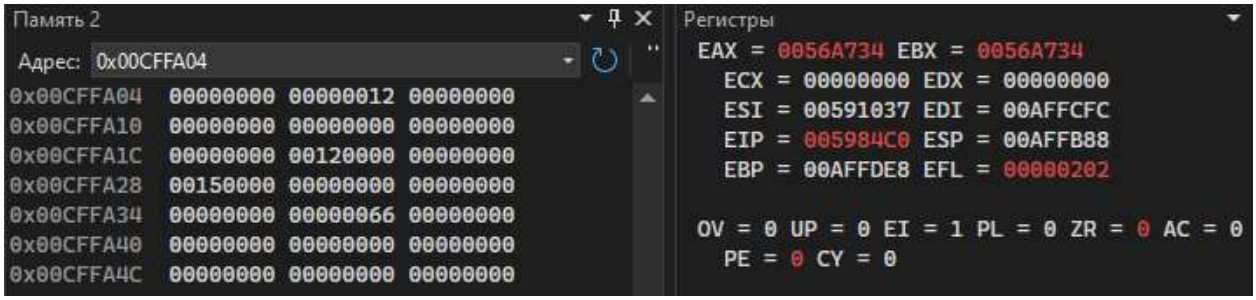
Пункт 2:

Вычислим знаменатель дроби, для этого определим порядок выполнения операций:

$$a - 1$$

Записываем переменную a в регистр EAX и перемножаем с числом, затем перемещаем результат в EBX для дальнейших вычислений:

```
MOV EAX, a
SUB EAX, 1
MOV EBX, EAX
```

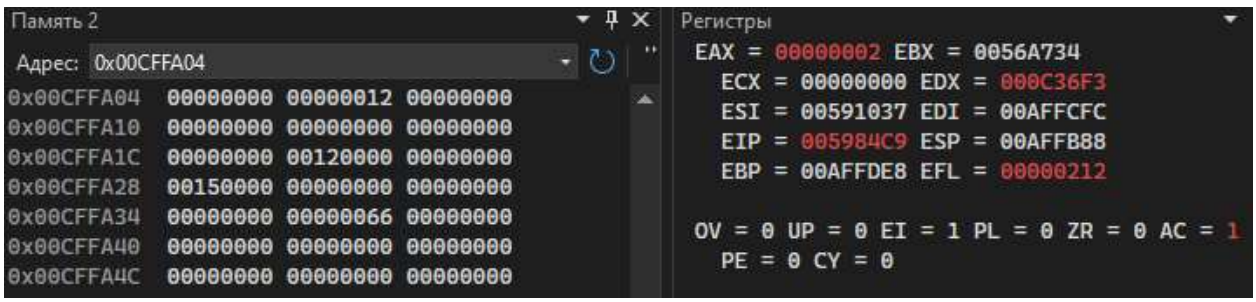


Проверяем: $5\ 678\ 901_{10} - 1 = 5\ 678\ 900_{10} = 56A734_{16}$, как видим это число находится в регистре EBX.

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
```



Проверим: возьмем числитель дроби, полученный после выполнения пункта 1 ($num1 = 12\ 158\ 299_{10} = B9855B_{16}$) и знаменатель дроби, который хранится в регистре EBX после пункта 2 ($num2 = 5\ 678\ 900_{10} = 56A734_{16}$).

Разделим числитель на знаменатель, получим:

$$num1/_{num2} = {}^{12\ 158\ 299_{10}}/_{5\ 678\ 900_{10}} = 2_{10}\ (\text{ост.}\ 800\ 499) = 2_{16}$$

В регистре EAX находится результат деления, сохраняем его в Memo как двойное слово.

Полный пример:

$$\frac{2 * b - \frac{a}{4}}{a - 1} = \frac{2 * 6\,789\,012_{10} - \frac{5\,678\,901_{10}}{4}}{5\,678\,901_{10} - 1} = \frac{13\,578\,024_{10} - 1\,419\,725_{10}}{5\,678\,900_{10}} = \frac{12\,158\,299_{10}}{5\,678\,900_{10}} = 2_{10}$$

$$\frac{2 * b - \frac{a}{4}}{a - 1} = \frac{2 * 679794_{16} - \frac{56A735_{16}}{4}}{56A735_{16} - 1} = \frac{CF2F28_{16} - 15A9CD_{16}}{56A734_{16}} = \frac{B9855B_{16}}{56A734_{16}} = 2_{16}$$

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.