Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОССУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОННИКИ

КАФЕДРА ИНФОРМАТИКИ

Отчёт по лабораторной работе №4-6

По дисциплине «Архитектуры вычислительных систем»

По теме «Арифметические операции с числами с плавающей точкой»

Выполнил:

студент гр. 653505

Хлопцев А. А.

Проверил:

Шиманский В. В.

Минск 2018

Оглавление

[1 Цель работы 3](#_Toc510038628)

[2 Постановка задачи 4](#_Toc510038629)

[3 Краткие теоретические сведения 5](#_Toc510038630)

[3.1 Представление чисел в компьютере 5](#_Toc510038631)

[3.2 Примеры представление чисел в компьютере 6](#_Toc510038632)

[4 Операции над числами с плавающей точкой 8](#_Toc510038633)

[4.1 Операция сложения/вычитания 8](#_Toc510038634)

[4.2 Операция умножения 11](#_Toc510038635)

[4.3 Операция деления 13](#_Toc510038636)

[5 Демонстрация работы программы 16](#_Toc510038637)

[6 Вывод 18](#_Toc510038638)

[7 Приложение. Код 19](#_Toc510038639)

# Цель работы

Изучить как производятся арифметические операции над числами с плавающей точкой в языке ассемблер.

# Постановка задачи

Написать программу эмулятора АЛУ, реализующего операции сложения, вычитания, умножения и деления чисел с плавающей точкой над двумя введенными числами, с возможностью пошагового выполнения алгоритмов.

# Краткие теоретические сведения

## Представление чисел в компьютере

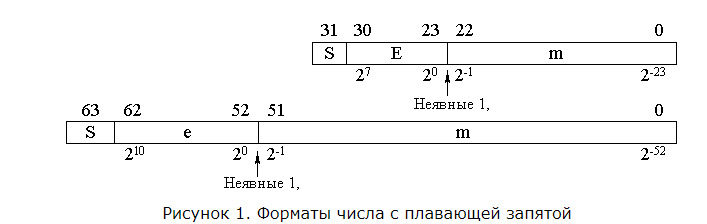
**Представление чисел в формате с плавающей запятой.** В десятичной арифметике для записи таких чисел используется алгебраическая форма. При этом число записывается в виде мантиссы, умноженной на 10 в степени, отображающей порядок числа. Например:

0,2×105;  
0,16×10-38

Для записи двоичных чисел тоже используется такая форма записи. Она позволяет работать с числами с большим диапазоном значений Эта форма записи называется запись числа с плавающей точкой. Напомним, что мантисса не может быть больше единицы и после запятой в мантиссе не может записываться ноль.

В программировании для записи таких чисел используются специальные форматы переменных. В языке СИ это такие форматы как **float** и **double**. Они описываются стандартом IEEE 754. Этих же стандартов придерживается большинство других языков программирования. При работе с числами с плавающей запятой в составе языка программирования обычно не возникает трудностей. Все преобразования форматов чисел осуществляются средствами самого языка программирования. Однако при передаче данных или при работе с микроконтроллерами часто приходится писать программу на языке программирования ассемблер и тогда может потребоваться знание внутреннего представления этих чисел.

Для записи числа в формате с плавающей запятой одинарной точности требуется тридцатидвухбитовое слово. Для записи чисел с двойной точностью требуется шестидесятичетырёхбитовое слово. Чаще всего числа хранятся в нескольких соседних ячейках памяти процессора. Форматы числа в формате с плавающей запятой одинарной точности и числа в формате с плавающей запятой удвоенной точности приведены на рисунке



На рисунке буквой S обозначен знак числа, 0 — это положительное число, 1 — отрицательное число.

Группа бит, обозначенная e предназначена для записи смещённого порядка числа. Смещение потребовалось, чтобы не вводить в двоичный код числа с плавающей запятой еще один знак. Смещённый порядок всегда является положительным числом. В двоичном коде одинарной точности **float** для записи порядка числа выделено восемь бит. Для него смещение порядка числа принято 127. Для смещённого порядка в двоичном коде числа с плавающей запятой двойной точности **double** отводится 11 бит. В нем смещение порядка числа составляет — 1023.

В десятичной мантиссе после запятой могут присутствовать цифры 1...9, а в двоичной — только 1. Поэтому для хранения единицы после двоичной запятой не выделяется отдельный бит в числе с плавающей запятой. Единица подразумевается, как и двоичная запятая. Кроме того, в формате чисел с плавающей запятой принято, что мантисса всегда больше 1. То есть диапазон значений мантиссы лежит в диапазоне от 1 до 2.

## Примеры представление чисел в компьютере

Рассмотрим несколько примеров:

1) Определить число с плавающей запятой, лежащее в четырёх соседних байтах:

11000001 01001000 00000000 00000000

- Знаковый бит, равный 1 показывает, что число отрицательное.

- Экспонента 10000010 в десятичном виде соответствует числу 130. Вычтя число 127 из 130, получим число 3.

- Теперь запишем мантиссу: 1,100 1000 0000 0000 0000 0000

- И, наконец, определим десятичное число: 1100,1b = 12,5d

2) Определить число с плавающей запятой, лежащее в четырёх соседних байтах:

11000011 00110100 00000000 00000000

- Знаковый бит, равный 1 показывает, что число отрицательное.

- Экспонента 10000110 в десятичном виде соответствует числу 134. Вычтя число 127 из 134, получим число 7.

- Теперь запишем мантиссу: 1,011 0100 0000 0000 0000 0000

- И, наконец, определим десятичное число: 10110100b=180d

Для того чтобы записать ноль, в двоичном представлении числа с плавающей запятой достаточно записать в смещенный порядок число 00000000b. Значение мантиссы при этом не имеет значения. Число, в котором все байты равны 0, тоже попадает в этот диапазон значений.

Бесконечность в числе с плавающей запятой соответствует смещенному порядку 11111111b и мантиссе, равной 1,0. При этом существует минус бесконечность и плюс бесконечность (переполнение и антипереполнение), которые часто отображаются на экран монитора компьютера или дисплей микропроцессорного устройства как **+INF** и **-INF**.

Все остальные комбинации битов мантиссы числа с плавающей запятой (в том числе и все единицы) при смещенном порядке 11111111b воспринимаются языками программирования как не числа и отображаются на экран: **NaN**.

# Операции над числами с плавающей точкой

## Операция сложения/вычитания

**Алгоритм сложения / вычитания**:

1) Производится выравнивание порядков чисел:

порядок меньшего по модулю числа принимается равным порядку большего числа, а мантисса меньшего сдвигается влево на *S*-ичных разрядов, равных разности порядков.

2) Производится сложение / вычитание мантисс, и получается сумма / разность мантисс.

3) Порядок суммы / разности принимается равным порядку большего числа.

4) Полученный результат нормализуется.

**Операция сложения / вычитания** состоит из следующих **этапов**:

1) прием операндов

2) выравнивание порядков со сдвигом мантисс

3) сложение / вычитание мантисс

4) нормализация результата

1) Прием операндов:

Прием слагаемого / вычитаемого *х* в регистр *RG3*. Фиксация знака числа (из *α0*) в триггере знака Тзн1. Установка регистра *RG3* в ноль.

Прием второго слагаемого / вычитаемого *y* в *RG1*. Фиксация знака в триггере знака Тзн2. Установка регистра *RGA* в ∅.

2) Операции над порядками:

Значение порядков из разрядов α*1* ÷ α*7* регистров *RG1* и *RG3* подаются в регистры *RGC* и *RGD*.

Далее в блоке обработки порядков происходит их сравнение, по результатам которого мантисса с меньшим порядком сдвигается вправо на число разрядов, равное разности порядков. За порядок результата операции принимается больший из порядков.

**При сравнении порядков** **возможны**пять случаев их **соотношения**:

• *Px* – *Py* > *m*, где *m* – число разрядов мантиссы, за результат в этом случае принимается слагаемое *х*, так как при сдвиге мантиссы слагаемого *y* все ее разряды примут нулевое значение.

• *Py* – *Px* > *m*, за результат принимается мантисса *y*.

• *Px* – *Py* = 0, производится суммирование / вычитание мантиссы.

• *Px* – *Py* = *K1* (*K1* < *m*), мантисса *y* сдвигается на *K1* разрядов.

• *Py* – *Px* = *K2* (*K2* < *m*), мантисса *x* сдвигается на *K2* разрядов.

Процесс сдвига мантиссы заключается в следующем: в счетчик циклов из блока обработки порядков заносится число разрядов *K1*или *K2*, на которые ее надо сдвинуть. По мере сдвига мантиссы содержимое счетчика циклов уменьшается, и когда счетчик циклов становится равным нулю, сдвиг прекращается.

Полученные модули мантисс хранятся в регистрах *RG1* и *RG3* в разрядах α*8*÷α*31*, их знаки – в триггерах знака Тзн1 и Тзн2, а принятый порядок хранится в регистре счетчика *RG*Сч1.

3) Сложение / вычитание мантисс:

а) при одинаковых знаках чисел модули мантисс передаются в *RGA* и *RGB* и складываются в сумматоре.

Если в сумматоре в разряде α*7* окажется единица, то возникло переполнение разрядной сетки, поэтому сумма сдвигается на разряд вправо, а порядок увеличивается на единицу, т.е. *RG1*:= *RG*Сч1 + 1.

Если окажется, что в сумматоре в разряде α*0* находится единица, то произошло переполнение порядка, поэтому вырабатывается сигнал прерывания вычислительного процесса.

Если переполнения порядка нет, то в регистр сумматора в разряды α*1* ÷ α*7* заносится порядок из регистра счетчика *RG*Сч1, в разряд α*0* в регистр сумматора заносится знак мантиссы, а в разряды сумматора α*8* ÷ α*31* заносится мантисса суммы.

Таким образом, в регистре сумматора полностью сформирован результат.

б) при разных знаках мантисс отрицательный из них передается на входной регистр *RGA* или *RGB*, в обратном коде производится суммирование с дальнейшим под суммированием единицы.

Знак результата фиксируется в триггере знака. Если полученный результат нормализован, т.е. в разряде сумматора α*8* находится единица, то в регистр сумматора заносится: знак результата – в α*0*, порядок – в α*1* ÷ α*7*, модуль мантиссы – в разряды α*8* ÷ α*31*.

Если результат не нормализован, т.е. в α*8* находится ноль, и нет исчезновения мантиссы ([α*8* ÷ α*31*] ≠ 0), производится нормализация сдвигом мантиссы влево с одновременным уменьшением порядка.

При отрицательном переполнении порядка, т.е. если *RG*Сч1[0]=1, формируется признак исчезновения порядка, т.е. результат равен нулю.

Если нормализация происходит без исчезновения порядка, то формируется результат из кодов знака, порядка и мантиссы.

## Операция умножения



При переумножении порядки складываются, мантиссы перемножаются, произведение нормализуется и ему присваивается знак «+», если знаки сомножителей одинаковы, и знак «−» в противном случае.

Если одна из мантисс равна нулю, то произведение сразу принимается нулевым.

Если при суммировании порядков возникает переполнение порядка и его знак отрицательный, то результат умножения равен нулю, т.к. произведение окажется меньше разрядной сетки.

Если возникло переполнение порядка со знаком «+», то может оказаться, что после нормализации результата перемножения мантиссы переполнение исчезнет, поэтому в этом случае факт переполнения запоминается до окончания нормализации.

Рассмотрим случай перемножения, когда операции над порядками производятся в основном АЛУ. В такой структуре АЛУ для перемножения чисел с плавающей точкой должны быть:

• *RG1* – для приема и хранения множимого *х;*

• *RG2* и *RG2’* – для приема множителя *y* и сдвига его мантиссы в процессе перемножения сомножителей;

• *RGA* – для передачи на сумматор смещенного порядка множимого *х* для его суммирования с порядком множителя *y* и для передачи мантиссы при умножении;

• *RGB* – для передачи на сумматор порядка множителя *y* для суммирования с порядком множимого *х*и для хранения суммы частичных произведений при перемножении мантисс;

• *RG*СМ – для приема результата суммирования порядков и перемножения мантисс;

• Тзн1 и Тзн2 – для хранения знака;

• *RG*Сч1 – для хранения порядка произведения.

Упрощенная схема:

Алгоритмы сложения порядков и перемножения мантисс аналогичны алгоритмам сложения / умножения целых чисел.

Нормализация аналогична нормализации результата при выполнении операции сложения / вычитания чисел с плавающей точкой.

## Операция деления



При делении чисел с плавающей точкой мантисса частного равна частному от деления мантиссы делимого на мантиссу делителя, а порядок частного равен разности порядков.

Частное нормализуется и ему присваивается знак «+», если делимое и делитель имеют одинаковые знаки, знак «–» в противоположном случае.

Если делимое равно нулю, то в частное записывается ноль без выполнения деления.

Если при вычитании порядков образовалось переполнение со знаком «+», или если делитель равен нулю, деление не производится и формируется сигнал прерывания.

При делении нормализованных чисел может оказаться, что мантисса делимого много больше делителя, и тогда мантисса частного образуется с переполнением. Поэтому часто перед делением мантисс нарушают нормализацию делителя сдвигом на разряд влево. Тогда нарушение нормализации частного не происходит.

Деление производится, как правило, аналогично делению целых чисел. Деление отличается только тем, что делимое берется такой же длины как и делитель. Однако, учитывая то, что мантисса делимого выражается дробным числом, можно условно принять, что делимое имеет двойную длину с нулями в разрядах младшей половины числа. После сдвига влево частичных остатков освобождающиеся разряды всегда заполняются нулями, поэтому деление можно выполнять точно так же, как деление целых чисел.

На схеме АЛУ должны быть:

• *RG1* и *RG2* – для приема делимого и делителя;

• *RGА* и *RGВ* – для передачи на сумматор смещенных порядков делителя и делимого, их мантисс при делении последовательно (*RGА* и *RGВ*требуются для обрботки порядков, если отсутствует блок обработки порядков отдельно);

• *RG*Сч1 – для хранения смещенного порядка частного;

• *RG2*, *RG2’* – для хранения мантиссы частного;

• триггеры знаков делимого и делителя.

Операция деления начинается с приема операндов в *RG1* и *RG2*, фиксации знаков в триггерах знака и передачи в *RGА* и *RGВ* смещенных порядков операндов.

Далее вычисляется разность порядков путем сложения в сумматоре прямого кода смещенного порядка делимого *х* и дополнительного кода делителя *y*.

Для получения смещенной разности порядков надо было бы прибавить единицу в разряд сумматора, однако коррекция смещения выполняется по-другому:

сначала производится анализ первого и второго разрядов сумматора (СМ[0÷1]) и далее:

• если СМ [0÷1] = 00, то получен положительный непереполненный порядок *p* и в смещенный порядок преобразуется инверсией первого разряда сумматора;

• если СМ [0÷1] = 01, положительное переполнение, сигнал прерывания.

Если СМ [0÷1] = 11, получим отрицательный порядок – смещение образуется инверсией первого разряда сумматора (СМ [1] → инверсия) и аналогично нулевого разряда (СМ [0] → инверсия).

Отсутствие переполнения фиксируется в триггере 1 занесением в него нуля.

Если СМ [0÷1] = 10, отрицательное переполнение порядка – факт такого переполнения фиксируется в триггере 1 занесением в него единицы.

Перед делением проверяется соотношение мантисс делимого и делителя, если окажется, что мантисса делимого много больше мантиссы делителя, то происходит сдвиг делителя влево на разряд, для того, чтобы не было переполнения частного.

далее деление мантисс делимого и делителя производится так же, как деление целых чисел. После завершения деления мантисса частного находится в RG2 в разрядах с 8 по 31.

Следующий этап – анализ значений триггера 1 (в котором зафиксирован факт переполнения порядка или его отсутствия) и содержимого RGсч1 для получения порядка результата деления. Варианты сочетания возможных значений в триггере 1 и в RGсч1 в разрядах ……. фиксируется в специальной таблице, по которой производится проверка полученного порядка.

Tr 1 → ∅ ⏐ RGсч1 → ∅1: порядок положительный;

Tr 1 → ∅ ⏐ RGсч1 → ∅∅: порядок отрицательный;

Tr 1 → 1 ⏐ RGсч1 → ∅1: переполнение (∞).

# Демонстрация работы программы

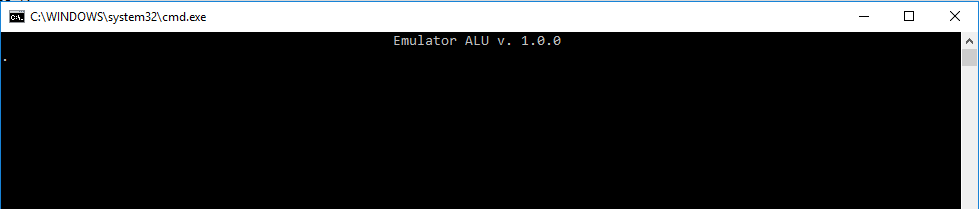


Рисунок 1 – Запуск программы

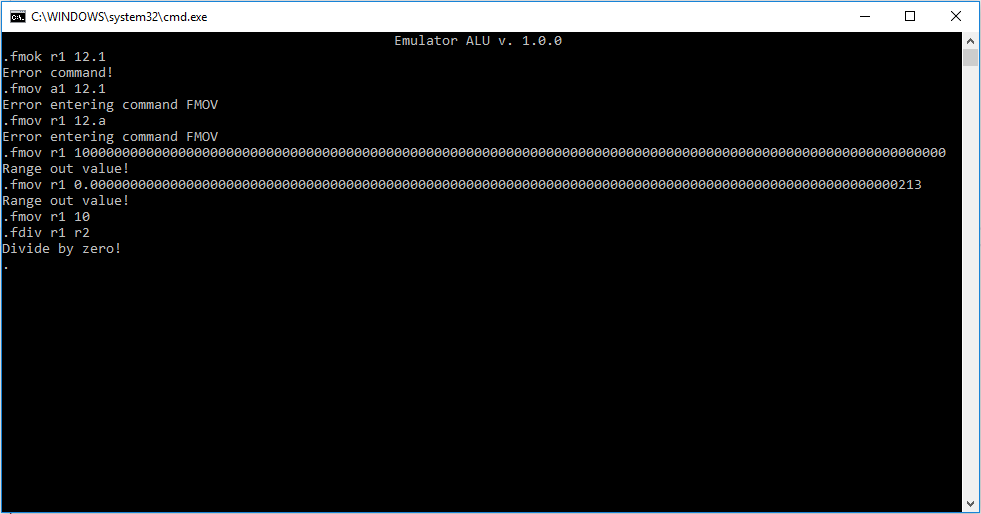
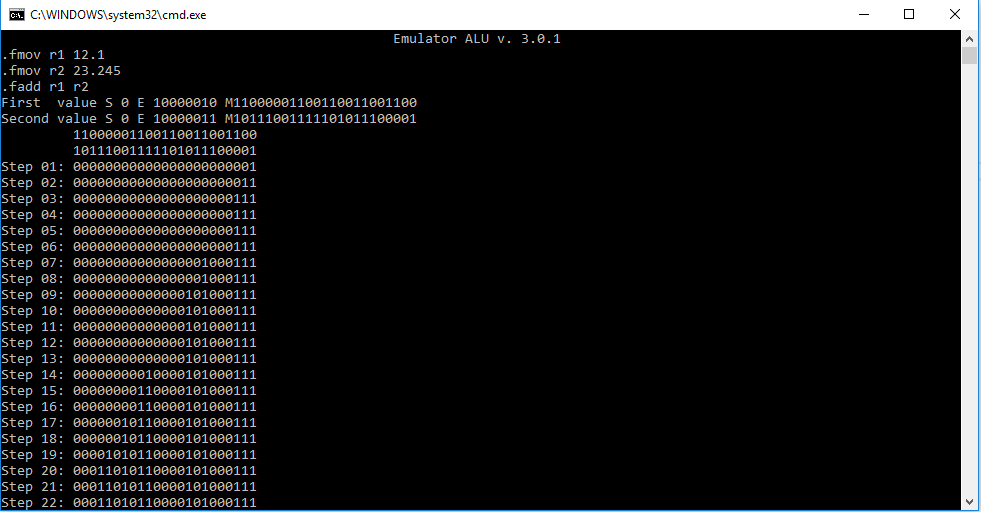
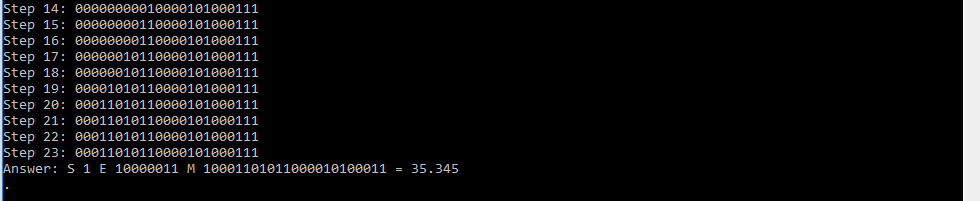
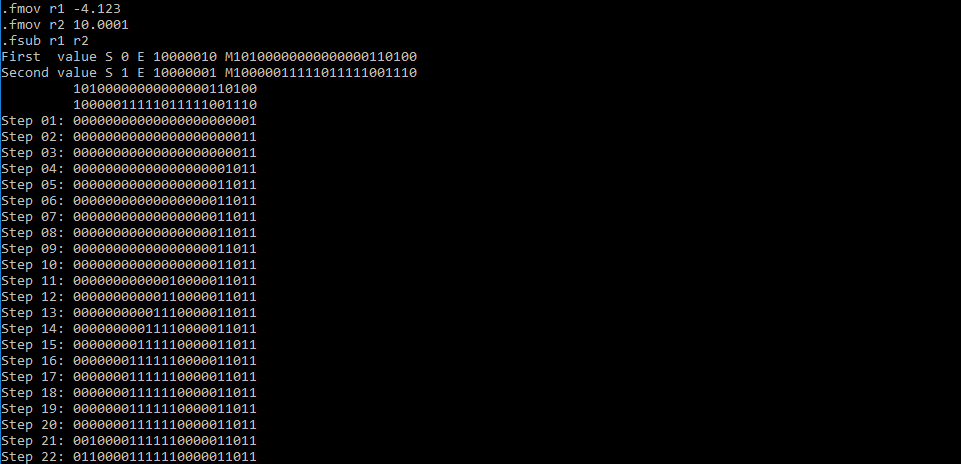
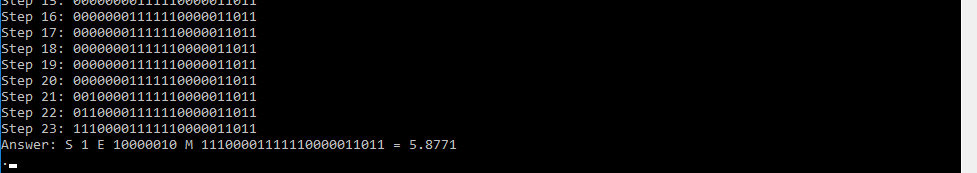


Рисунок 2 – Виды ошибок программы 



Рисунки 3 - 4 – Операция сложения чисел с плавающей точкой



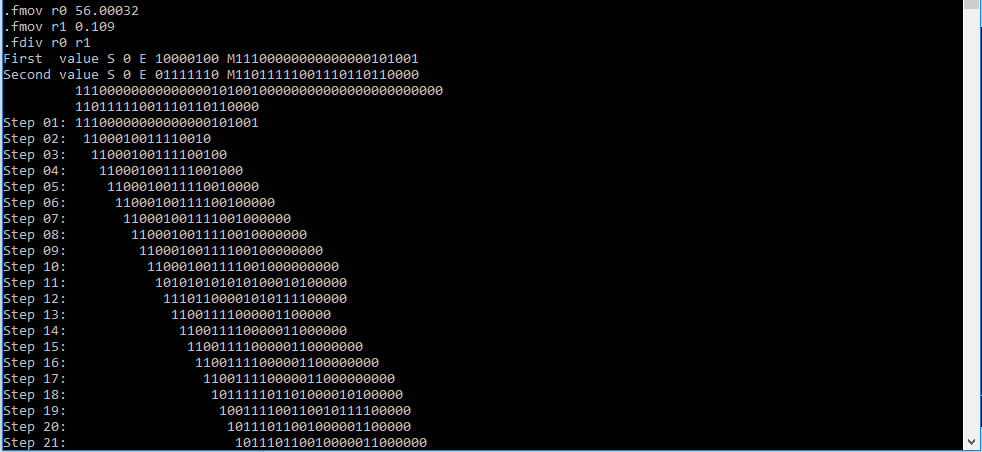


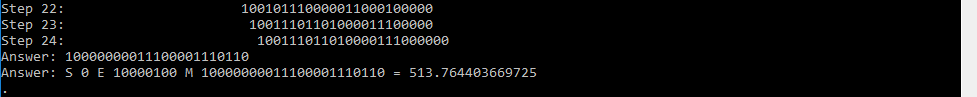
Рисунки 5 - 6 – Операция вычитание чисел с плавающей точкой





Рисунки 7 - 8 – Операция умножение чисел с плавающей точкой





Рисунки 9 - 10 – Операция деления чисел с плавающей точкой

# Вывод

В результате выполнения лабораторных работ №4-6 была реализована программа эмулятора АЛУ, реализующего операции сложения, вычитания, умножения и деления чисел с плавающей точкой над двумя введенными числами, с возможностью пошагового выполнения алгоритмов.

# Приложение. Код

public static class SCPU

{

//static public SoReg r1 = new SoReg();

static public Hashtable reg;

static public bool MExit;

static SCPU()

{

MExit = false;

reg = new Hashtable

{

{ "r7", new SoReg("r7") },

{ "r6", new SoReg("r6") },

{ "r5", new SoReg("r5") },

{ "r4", new SoReg("r4") },

{ "r3", new SoReg("r3") },

{ "r2", new SoReg("r2") },

{ "r1", new SoReg("r1") },

{ "r0", new SoReg("r0") }

};

}

public static void FMOV(string name, string sval)

{

double val = double.Parse(sval);

if (Math.Abs(val) > 1e38 || (Math.Abs(val) < 1e-38))

{

Console.WriteLine("Range out value!");

return;

}

(reg[name] as SoReg).In(val);

}

public static void Info()

{

foreach (DictionaryEntry v in reg)

(v.Value as SoReg).Info();

}

public static void Clear()

{

foreach (DictionaryEntry v in reg)

(v.Value as SoReg).Clear();

}

public static void FADD(string fir, string sec)

{

SoReg rf = reg[fir] as SoReg;

SoReg rs = reg[sec] as SoReg;

if (rf.val < 0 && rs.val > 0)

Visual.VisualCommandFSub(1, fir, sec);

else

if (rf.val > 0 && rs.val < 0)

Visual.VisualCommandFSub(0, fir, sec);

else

Visual.VisualCommandFAdd(1,fir, sec);

double ans = rf.val + rs.val;

rf.In(ans);

}

public static void FSUB(string fir, string sec)

{

SoReg rf = reg[fir] as SoReg;

SoReg rs = reg[sec] as SoReg;

if (rf.val < rs.val)

{

string s = fir;

fir = sec;

sec = s;

}

if (rf.val < 0 && rs.val > 0)

Visual.VisualCommandFAdd(1, fir, sec);

else

if (rf.val > 0 && rs.val < 0)

Visual.VisualCommandFAdd(0, fir, sec);

else

if (rf.val < 0 && rs.val < 0)

Visual.VisualCommandFSub(1, fir, sec);

else

Visual.VisualCommandFSub(0, fir, sec);

double ans = rf.val - rs.val;

rf.In(ans);

}

public static void FMUL(string fir, string sec)

{

Visual.VisualCommandFMul(fir, sec);

SoReg rf = reg[fir] as SoReg;

SoReg rs = reg[sec] as SoReg;

double ans = rf.val \* rs.val;

rf.In(ans);

}

public static void FDIV(string fir, string sec)

{

SoReg rf = reg[fir] as SoReg;

SoReg rs = reg[sec] as SoReg;

if (Math.Abs(rs.val) <= 0.00000001)

{

Console.WriteLine("Divide by zero!");

return;

}

Visual.VisualCommandFDiv(fir, sec);

double ans = rf.val / rs.val;

rf.In(ans);

}

public class SoReg

{

public double val;

public int exp;

public byte sign;

public string or;

public string man;

string name;

public SoReg(string name)

{

sign = 0;

or = new string('0', 8);

man = new string('0', 23);

val = 0;

exp = 0;

this.name = name;

}

public void Info()

{

Console.Write(name + " S " + sign + " E ");

Console.Write(or);

Console.Write(" M ");

Console.Write(man);

Console.Write(" = " + val + '\n');

}

public void Clear()

{

sign = 0;

or = new string('0', 8);

man = new string('0', 23);

val = 0;

}

public void In(double ch)

{

Clear();

string s = ch.ToString(); ;

val = ch;

if (ch < 0)

{

sign = 1;

ch \*= -1;

}

int fir = 0;

fir = (int)Math.Truncate(ch);

exp = Convert.ToString(fir, 2).Length;

int len = exp - 1 + 127;

if (fir < 1)

{

double d = ch - fir;

double c = ch;

int x = 0;

while ((int)Math.Truncate(c) == 0)

{

c \*= 10;

x++;

}

len = -x + 127;

}

man = Convert.ToString(fir, 2);

if (fir != ch)

{

man += Help.ToBinFrac(ch - fir, 23 - man.Length);

}

man = man.TrimStart('0');

or = Convert.ToString(len, 2);

string help = new string('0', 8 - or.Length);

or = help + or;

if (man.Length < 23)

{

help = new string('0', 23 - man.Length);

man += help;

}

}

}

static class Help

{

public static string ToBinFrac(double frac, int len)

{

string str = "";

int c;

int n = 0;

while (n < len)

{

frac \*= 2;

c = Convert.ToInt32(Math.Truncate(frac));

str = String.Concat(str, Convert.ToString(c));

frac -= c;

n++;

}

return str;

}

}

}

public static void VisualCommandFAdd(byte sign, string a1, string a2)

{

SoReg fir = SCPU.reg[a1] as SoReg;

SoReg sec = SCPU.reg[a2] as SoReg;

string f = fir.man;

string s = sec.man;

Console.WriteLine("First value S {0} E {1} M{2}", fir.sign, fir.or, fir.man);

Console.WriteLine("Second value S {0} E {1} M{2}", sec.sign, sec.or, sec.man);

int max = Math.Max(fir.exp, sec.exp);

if (fir.exp != max)

{

f = new string('0', max - fir.exp) + f;

f = f.Substring(0, 23);

}

else

{

s = new string('0', max - sec.exp) + s;

s = s.Substring(0, 23);

}

max--;

StringBuilder ans = new StringBuilder(new string('0', 23));

Console.WriteLine("{0, 32}", fir.man);

Console.WriteLine("{0, 32}", sec.man);

int over = 0;

int i = 23;

int step = 0;

bool exit = SCPU.MExit;

while (true)

{

i--;

if (i < 0) break;

if (!exit)

if (Console.ReadKey(true).Key == ConsoleKey.Escape)

exit = true;

if (f[i] == '0' && s[i] == '0')

{

if (over == 0)

{

ans[i] = '0';

over = 0;

}

else

{

ans[i] = '1';

over = 0;

}

}

if (f[i] == '0' && s[i] == '1')

{

if (over == 0)

{

ans[i] = '1';

over = 0;

}

else

{

ans[i] = '0';

over = 1;

}

}

if (f[i] == '1' && s[i] == '0')

{

if (over == 0)

{

ans[i] = '1';

over = 0;

}

else

{

ans[i] = '0';

over = 1;

}

}

if (f[i] == '1' && s[i] == '1')

{

if (over == 0)

{

ans[i] = '0';

over = 1;

}

else

{

ans[i] = '1';

over = 1;

}

}

Console.WriteLine("Step {0}: {1}", (++step).ToString("D2"), ans);

}

if (over == 1)

{

ans.Insert(0, "1");

//max++;

}

string Ans = ans.ToString();

int len = Ans.Length;

Ans = Ans.TrimStart('0');

max += len - Ans.Length;

//if (max == 0)

// max = 1;

if (Ans.Length > 22)

Ans = Ans.Substring(0, 23);

Console.WriteLine("Answer: S {0} E {1} M {2} = {3}", sign, Convert.ToString(max + 127, 2), Ans + new string('0', 23 - Ans.Length), fir.val + sec.val);

}

public static void VisualCommandFSub(byte sign, string a1, string a2)

{

SoReg fir = SCPU.reg[a1] as SoReg;

SoReg sec = SCPU.reg[a2] as SoReg;

string f = fir.man;

string s = sec.man;

Console.WriteLine("First value S {0} E {1} M{2}", fir.sign, fir.or, fir.man);

Console.WriteLine("Second value S {0} E {1} M{2}", sec.sign, sec.or, sec.man);

int max = Math.Max(fir.exp, sec.exp);

if (fir.exp != max)

{

f = new string('0', max - fir.exp) + f;

f = f.Substring(0, 23);

}

else

{

s = new string('0', max - sec.exp) + s;

s = s.Substring(0, 23);

}

max--;

if (fir.val < 0 && sec.val < 0)

max++;

StringBuilder ans = new StringBuilder(new string('0', 23));

int over = 0;

int i = 23;

int step = 0;

bool exit = SCPU.MExit;

Console.WriteLine("{0, 32}", fir.man);

Console.WriteLine("{0, 32}", sec.man);

while (true)

{

i--;

if (i < 0) break;

if (!exit)

if (Console.ReadKey(true).Key == ConsoleKey.Escape)

exit = true;

if (f[i] == '1' && s[i] == '1')

{

if (over == 0)

{

ans[i] = '0';

over = 0;

}

else

{

ans[i] = '1';

over = 1;

}

}

else

if (f[i] == '0' && s[i] == '0')

{

if (over == 0)

{

ans[i] = '0';

over = 0;

}

else

{

ans[i] = '1';

over = 1;

}

}

else

if (f[i] == '1' && s[i] == '0')

{

if (over == 0)

{

ans[i] = '1';

over = 0;

}

else

{

ans[i] = '0';

over = 0;

}

}

else

if (f[i] == '0' && s[i] == '1')

{

if (over == 0)

{

ans[i] = '1';

over = 1;

}

else

{

ans[i] = '0';

over = 1;

}

}

Console.WriteLine("Step {0}: {1}", (++step).ToString("D2"), ans);

}

if (over == 1)

{

max--;

}

string Ans = ans.ToString();

int len = Ans.Length;

Ans = Ans.TrimStart('0');

max -= len - Ans.Length;

//if (max == 0)

// max = 1;

if (Ans.Length > 22)

Ans = Ans.Substring(0, 23);

Console.WriteLine("Answer: S {0} E {1} M {2} = {3}", sign, Convert.ToString(max + 127, 2), Ans + new string('0', 23 - Ans.Length), fir.val - sec.val);

}

public static void VisualCommandFMul(string a1, string a2)

{

SoReg fir = SCPU.reg[a1] as SoReg;

SoReg sec = SCPU.reg[a2] as SoReg;

string first = fir.man;

string second = sec.man;

int exp = fir.exp + sec.exp - 2;

Console.WriteLine("First value S {0} E {1} M{2}", fir.sign, fir.or, fir.man);

Console.WriteLine("Second value S {0} E {1} M{2}", sec.sign, sec.or, sec.man);

//Console.WriteLine(Convert.ToString(exp + 127, 2));

Console.WriteLine("{0, 55}", fir.man);

Console.WriteLine("{0, 55}", sec.man);

string help = new string('0', first.Length + second.Length);

StringBuilder ans = new StringBuilder(help);

int len = ans.Length;

int i = 23;

int step = 0;

bool exit = SCPU.MExit;

while (true)

{

i--;

if (i < 0) break;

if (!exit)

if (Console.ReadKey(true).Key == ConsoleKey.Escape)

exit = true;

help = "";

for (int j = 0; j < first.Length; j++)

if (first[j] == '0' || second[i] == '0')

help += '0';

else

help += '1';

int k = first.Length - 1;

int over = 0, j1 = 0;

for (int j = len - 1; k > -1; j--, k--)

{

int prval = ((ans[j] - '0') + over + (help[k] - '0'));

ans[j] = (char)(prval % 2 + '0');

over = prval / 2;

j1 = j;

}

if (over > 0)

ans[j1 - 1] = '1';

Console.WriteLine("step {0}| {1}", (++step).ToString("D2"), help.PadLeft(len--));

}

Console.WriteLine(new string('-', ans.Length + 9));

Console.WriteLine("{0, -7}| {1}", "answer", ans);

string Ans = ans.ToString();

Ans = Ans.TrimStart('0');

if (Ans.Length > 22)

Ans = Ans.Substring(0, 23);

int v = 0;

if (fir.val \* sec.val < 0)

v = 1;

Console.WriteLine("Answer: S {0} E {1} M {2} = {3}", v, Convert.ToString(exp + 127 + 1, 2), Ans + new string('0', 23 - Ans.Length), fir.val \* sec.val);

}

public static void VisualCommandFDiv(string a1, string a2)

{

SoReg fir = SCPU.reg[a1] as SoReg;

SoReg sec = SCPU.reg[a2] as SoReg;

string f = fir.man;

string s = sec.man;

int exp = fir.exp - sec.exp;

Console.WriteLine("First value S {0} E {1} M{2}", fir.sign, fir.or, fir.man);

Console.WriteLine("Second value S {0} E {1} M{2}", sec.sign, sec.or, sec.man);

//Console.WriteLine(Convert.ToString(exp + 127, 2));

//Console.WriteLine("{0, 55}", fir.man);

//Console.WriteLine("{0, 55}", sec.man);

int i = 0;

f += new string('0', 23);

bool exit = SCPU.MExit;

int step = 0;

string help = "";

string ans = "";

int u = 0;

Console.WriteLine(f.PadLeft(f.Length + 9));

Console.WriteLine(s.PadLeft(s.Length + 9));

for (; u < 23; ++u)

help += f[u];

while (true)

{

Console.WriteLine("Step {0}: {1}{2}",(++step).ToString("D2"), new string(' ', i++), help);

if (u >= f.Length)

break;

if (!exit)

if (Console.ReadKey(true).Key == ConsoleKey.Escape)

exit = true;

if (Checker.Equals(help, s))

{

ans += "1";

int over = 0;

StringBuilder A = new StringBuilder(new string('0', help.Length));

s = new String('0', help.Length - s.Length) + s;

for (int j = help.Length - 1; j > -1; --j)

{

if (help[j] == '1' && s[j] == '1')

{

if (over == 0)

{

A[j] = '0';

over = 0;

}

else

{

A[j] = '1';

over = 1;

}

}

else

if (help[j] == '0' && s[j] == '0')

{

if (over == 0)

{

A[j] = '0';

over = 0;

}

else

{

A[j] = '1';

over = 1;

}

}

else

if (help[j] == '1' && s[j] == '0')

{

if (over == 0)

{

A[j] = '1';

over = 0;

}

else

{

A[j] = '0';

over = 0;

}

}

else

if (help[j] == '0' && s[j] == '1')

{

if (over == 0)

{

A[j] = '1';

over = 1;

}

else

{

A[j] = '0';

over = 1;

}

}

}

help = A.ToString();

help = help.TrimStart('0');

s = s.TrimStart('0');

help += f[u++];

}

else

{

ans += "0";

help += f[u++];

}

}

Console.WriteLine("Answer: " + ans);

ans = ans.TrimStart('0');

if (ans.Length > 22)

ans = ans.Substring(0, 23);

int v = 0;

if (fir.val / sec.val < 0)

v = 1;

Console.WriteLine("Answer: S {0} E {1} M {2} = {3}", v, Convert.ToString(exp + 127, 2), ans + new string('0', 23 - ans.Length), fir.val / sec.val);

}

}

}