Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информатики

Отчет по лабораторной работе №6

Эмуляция АЛУ. Операция деления

чисел с плавающей точкой

Студент: гр. 853504

Вечеринский Максим Сергеевич

Руководитель: старший преподаватель

Шиманский В.В.

Минск 2020

СОДЕРЖАНИЕ

1. Введение
2. Постановка задачи
3. Программная реализация
4. Выводы

Литература

Приложение 1 – Текст программы

1.Введение

**1.1 Представление целых чисел.**

В двоичной системе счисления числа представляются с помощью комбинации единиц и нулей, знака "минус" и знака разделяющей точки между целой и дробной частью числа. Например, десятичное число -1.312510 в двоичном виде будет выглядеть как -1001.01012. Но в компьютере мы не можем хранить и обрабатывать символы знака и разделяющей точки — для "машинного" представления чисел могут использоваться только двоичные цифры (0 и 1). Если операции выполняются только с неотрицательными числами, то формат представления очевиден. В машинном слове из 8 бит можно представить числа в интервале от 0 до 255. Пример:

|  |  |
| --- | --- |
| *00000000  =* | *0* |
| *00000000  =* | *0* |
| *00000001  =* | *1* |
| *00101001  =* | *41* |
| *10000000  =* | *128* |
| *11111111* | *255* |

В общем случае n-битовая последовательность двоичных цифр an-1an-2…a1a0 интерпретирована как целое число А, значение которого равно

**1.2. Прямой код.**

Существует несколько соглашений о едином формате представления как положительных, так и отрицательных чисел. Всех их объединяет то, что старший бит слова (с точки зрения европейца — самый левый, или бит, которому при представлении числа без знака должен быть приписан самый большой вес) является битом хранения знака или знаковым разрядом. Все последующие биты слова представляют значащие разряды числа, которые в каждом формате интерпретируются по-своему. Значение 1 в знаковом разряде интерпретируется как представление всем словом отрицательного числа.

00010010 = +18

10010010 = -18

Общее правило математически формулируется следующим образом:

Формат представления чисел в прямом коде неудобен для использования в вычислениях. Во-первых, сложение и вычитание положительных и отрицательных чисел выполняется по-разному, а потому требуется анализировать знаковые разряды операндов. Во-вторых, в прямом коде числу 0 соответствуют две кодовых комбинации:

00000000 = +010

10000000 = -010

Это также неудобно, поскольку усложняется анализ результата на равенство нулю, а такая операция в программах встречается очень часто.

Из-за этих недостатков прямой код практически не применяется при реализации в АЛУ арифметических операций над целыми числами. Вместо этого более широкое применение находит другой формат, получивший наименование дополнительного кода.

1.3. Двоичное деление

Если умножение выполняется путем многократных сдвигов и сложений, то деление, будучи операцией обратной умножению,— путем многократных сдвигов и вычитаний.

(ПРАВИЛЬНЫЕ ДРОБИ, БЕЗ ЦЕЛОГО.)

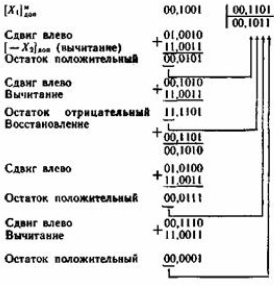
*При представлении чисел с фиксированной запятой деление возможно, если делимое по модулю меньше делителя, в противном случае произойдет переполнение разрядной сетки*.

Так же, как и при «ручном» делении, разряды частного при делении чисел на машине определяются (начиная со старшего) путем последовательного вычитания делителя из остатка, полученного от предыдущего вычитания. Однако здесь операция вычитания заменяется операцией сложения остатка с отрицательным делителем, представленным в обратном или дополнительном коде. Знак частного определяется сложением по модулю два кодов знаков делимого и делителя.

В машинах вместо сдвига делителя вправо осуществляется сдвиг остатка влево, что, по сути, ничего не изменяет.

При делении с восстановлением остатка отрицательный остаток восстанавливается суммированием с положительным делителем. Восстановленный остаток сдвигается влево на один разряд. Из сдвинутого остатка вновь вычитается делитель. По знаку полученного остатка определяется цифра очередного разряда частного. Процесс деления продолжается до получения заданного числа цифр частного, обеспечивающего необходимую точность результата.

Процесс деления начинается со сдвига делимого влево на один разряд, после чего к нему прибавляется делитель, представленный, например, в дополнительном модифицированном коде:



1) Определить знак частного суммированием по модулю два содержимых знаковых разрядов делимого и делителя.

2) Из делимого вычесть делитель. Если остаток отрицательный, перейти к пункту 3. В противном случае вычисление закончить (произошло переполнение).

3) Запомнить знак остатка.

4) Сдвинуть остаток на один разряд влево.

5) Присвоить делителю знак, обратный знаку остатка, запомненному в п. 2.

6) Сложить сдвинутый остаток и делитель (с учетом знака).

7) Присвоить цифре частного значение, противоположное коду знака остатка.

8) Повторять выполнение пунктов 3—7 до тех пор, пока не будет обеспечена требуемая точность вычисления частного.

Деление целых ненулевых n-разрядных (не считая знаковых разрядов) чисел А:В, представленных в прямом (для простоты) коде, приводит к получению целого частного С и целого остатка 0, которому присваивается знак делимого; знак частного вычисляется как сумма по модулю два операндов А и В.

Деление выполняется в следующей последовательности.

1) Делитель В сдвигается влево (нормализуется), так чтобы в старшем информационном разряде оказалась 1;подсчитывается количество сдвигов S; частное от деления может быть не более (S + 1) разрядов, не равных нулю.

2) Выполняется (S+1) цикл деления модулей |А| на IB’l где В' — нормализованное В, в результате находится(S+ 1) разряд частного, начиная со старшего из (S+ 1)младших.

3) Полученный в последнем цикле деления остаток Rs+1, если он положительный, сдвигается вправо на S разрядов; если же Rs+1 < 0 (отрицательный), то остаток восстанавливается: к нему добавляется |В'|, т. е.[Rs+1]вост = Rs+1+|B'|. После этого выполняется сдвиг вправо на S разрядов. В результате получается целый остаток от деления.



1.4. Двоичное деление чисел с плавающей точкой

Перевод выполняется за n+1 такт, где n – число двоичных разрядов, обеспечивающих точность перевода такую же, как у исходного десятичного числа. Необходимое сило разрядов может быть определено из выражения

 где k – число разрядов десятичного числа, определяющее его точность.

Чтобы определить алгоритм перевода числа из системы счисления с основанием q в систему счисления с основанием p, запишем числоAq в виде разложения по степеням 2:

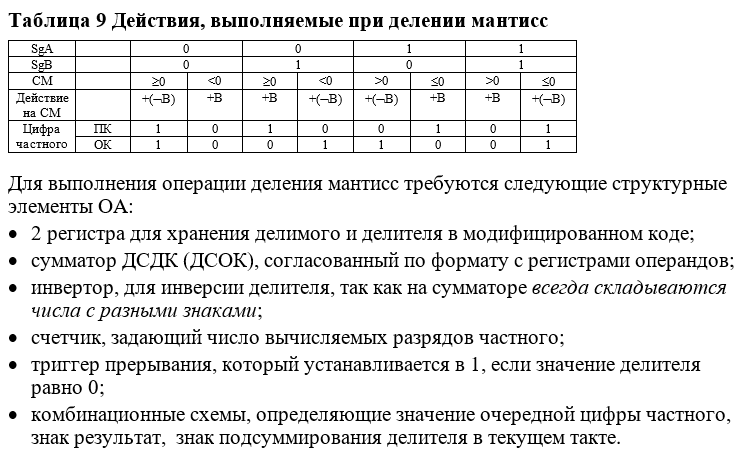
Aq=a1p-1+a2p-2+…+anp-n=p-1(a1+p-1(a2+…+p-1(an+0))…).

Полученное выражение есть *схема Горнера*, определяющая последовательность действий при переводе.

Умножая Aq на p, получим смешанную дробь, целая часть которой будет равна первой цифре числа в системе счисления с основанием q. Далее, умножая дробную часть получившегося числа A’q на p, будем получать в целой части получающейся смешанной дроби значения очередных цифр в системе счисления с основанием р. При этом, если q>p, то цифры числа в новой системе счисления будут сразу получаться правильными. Если же p>q, то целая часть смешанной дроби, получающейся в результате умножения q-ичной правильной дроби на основание новой системы счисления будет представлять очередную цифру р-ичного числа кодовой последовательностью в q-ичной системе счисления. Из выражения (1) следует следующий алгоритм перевода правильной дроби из кода Д1 в двоичную систему счисления, представленный в табл. 1Примем значение исходного числа равным А10=0.13710.

Число необходимых двоичных разрядов не меньше (3/lg2), т.е. больше или равно 10. Следовательно, необходимо принять число тактов равным 11 для последующего округления результата. В коде Д1 наше число будет иметь вид: А10= 0001 0011 0111. Умножение на 2 соответствует сдвигу регистра, в котором хранится число А на один разряд влево. Исходное число в коде Д1 хранится в РгА, Двоичный код записывается в РгВ последовательно разряд за разрядом, начиная со старшего разряда, путем сдвига РгВ влево. В таблице, представляющей алгоритм перевода, будем также записывать результат умножения десятичного числа на 2 в десятичной системе счисления (столбец А10), чтобы контролировать результат выполнения операций сдвига и коррекции в коде Д1. Таблица также содержит столбец СТ, в котором учитывается число уже выполненных тактов. Критерием завершения операции перевода является СТ=0.

При делении двоичных чисел мантиссы делятся, а порядки вычитаются. При делении нормализованных мантисс может случиться переполнение разрядной сетки частного, если мантисса делимого по модулю больше мантиссы делителя. В этом случае в псевдознаковый разряд частного, предшествующий старшему разряду частного, записывается единица, которая является критерием переполнения разрядной сетки. По окончании процедуры деления необходимо проверить содержимое псевдознакового разряда. При обнаружении переполнения разрядной сетки частного производится нормализация результата. Она состоит в том, что мантисса частного сдвигается на один разряд вправо, а порядок результата увеличивается на 1. В схеме операционного автомата так же, как и при умножении и делении двоичных чисел с плавающей запятой, используются специальные структурные элементы (регистры, сумматоры, комбинационные схемы), предназначенные для раздельной обработки порядков



2. Постановка задачи

2.1. Текст задания

Эмуляция АЛУ. Реализовать операцию деления чисел с плавающей точкой.

2.2. Примечание к заданию

Реализовать ввод двух чисел. Вычислить частное в двоичной системе счисления и вывести результат на экран.

3. Программная реализация

3.1. С консоли вводятся два числа в десятичной системе счисления. Затем реализуется перевод их в двоичную систему счисления. В прямом коде реализуется деление. Затем на экран выводится результат деления в двоичной системе счисления и в десятичной. Работа выполняется только для 8-ми разрядных значений.

3.2.1. Тест для «делимое = 23.12567, делитель = 2.3333»

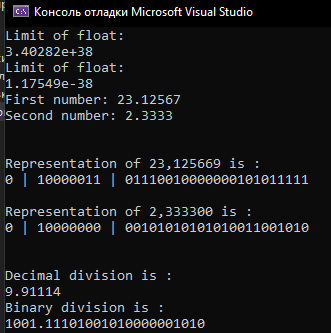


Рисунок 1. Скриншот для «делимое = 23.12567, делитель = 2.3333»

Ответ: Частное = 9.91114.

3.2.2. Тест для «делимое = -3342134.2334, делитель = 23.455555»

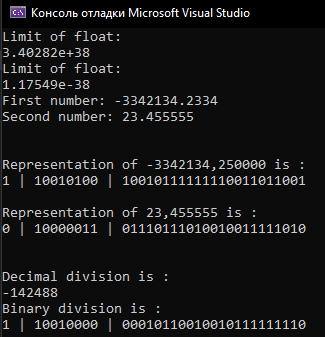


Рисунок 2. Скриншот для «делимое = -3342134.2334,

делитель = 23.455555»

Ответ: Частное = -142488.

3.2.3. Тест для «делимое = -12666432234.444, делитель = -199.33234»

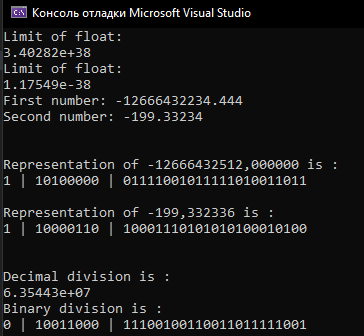


Рисунок 3. Скриншот для «делимое = -12666432234.444,

делитель = -199.33234»

Ответ: Частное = 9.91114.

3.2.4. Тест для «делимое = 3.1122, делитель = 0»

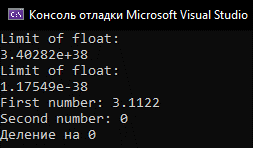


Рисунок 1. Скриншот для «делимое = 3.1122, делитель = 0»

Ответ: Деление на 0.

3.2.5. Тест для

«делимое = 123444444444444444444444444421342233231.33213,

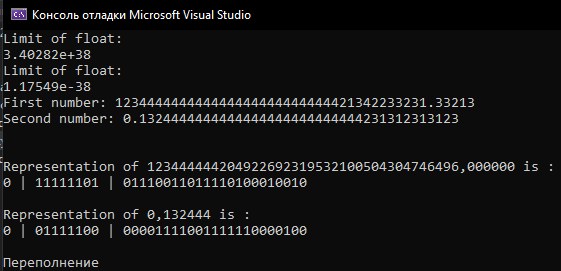
делитель = 0.1324444444444444444444444444231312313123»

Рисунок 5. Скриншот для

«делимое = 123444444444444444444444444421342233231.33213,

делитель = 0.1324444444444444444444444444231312313123»

Ответ: Переполнение.

 3.2.6. Тест для «делимое = 0.0000000000000000000000000000001, делитель = 10000000000000000000000000000.00000000001»

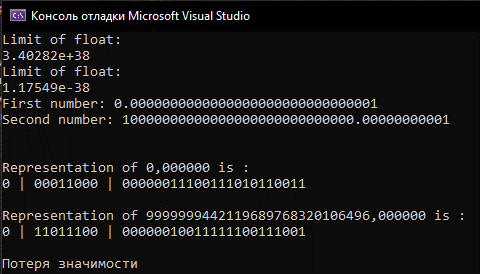


Рисунок 6. Скриншот для

«делимое = 0.0000000000000000000000000000001,

делитель = 10000000000000000000000000000.00000000001»

Ответ: Потеря значимости.

3.2.7. Резюме.

В своих примерах я старался рассмотреть максимально разнообразный набор случаев. Были приведены примеры:

* Частного двух положительных чисел
* Частного положительного и отрицательного чисел
* Частного двух отрицательных чисел
* Частное, выходящее за допустимые границы.

4.Выводы

В ходе выполнения лабораторной работы ознакомился с понятиями мантиссы, экспоненты, и стандарта IEEE 754.

Так же рассмотрел работу арифметико-логического устройства (АЛУ), рассмотрел различные нюансы использования АЛУ. Для реализации использовал язык С++ , а так же компилятор Visual Studio 2019.

Литература

1. Волорова Н. А. Лабораторный практикум по курсу «Архитектура вычислительных систем» для студентов специальности «Информатика» /985-444-487-2 – Мн.: БГУИР, 2003. — 32 с.: ил.

2. Tanenbaum A. S. «Structured computer organization» /Vrije Universiteit Amsterdam, The Netherlands, 2013. — 810 c.:ил.

ПРИЛОЖЕНИЕ 1

Текст программы

#include <iostream>

#include <string>

#include <conio.h>

#include <stdio.h>

#include <cstdlib>

#include <math.h>

#include<limits>

using namespace std;

double n, bin, f1 = 0.0, f2, f3 = 0.1;

#include <stdio.h>

void printBinary(int n, int i)

{

int k;

for (k = i - 1; k >= 0; k--) {

if ((n >> k) & 1)

printf("1");

else

printf("0");

}

}

typedef union {

float f;

struct

{

unsigned int mantissa : 23;

unsigned int exponent : 8;

unsigned int sign : 1;

} raw;

} myfloat;

int\* MakeArr(int\* array, myfloat var);

int\* Division(int\* arr1, int\* arr2, int\* arr3);

void printIEEE(myfloat var)

{

printf("%d | ", var.raw.sign);

printBinary(var.raw.exponent, 8);

printf(" | ");

printBinary(var.raw.mantissa, 23);

printf("\n");

}

std::string decimalToBinary(double num, int k\_prec)

{

std::string binary = "";

int Integral = num;

double fractional = num - Integral;

while (Integral)

{

int rem = Integral % 2;

binary.push\_back(rem + '0');

Integral /= 2;

}

reverse(binary.begin(), binary.end());

binary.push\_back('.');

while (k\_prec--)

{

fractional \*= 2;

int fract\_bit = fractional;

if (fract\_bit == 1)

{

fractional -= fract\_bit;

binary.push\_back(1 + '0');

}

else

binary.push\_back(0 + '0');

}

return binary;

}

int main()

{

setlocale(LC\_ALL, "Russian");

myfloat var, var2, div;

float float\_max = fabs(numeric\_limits<float>::max());

float float\_min = fabs(numeric\_limits<float>::min());

float num, num2;

cout << "Limit of float: " << endl << float\_max << endl;

cout << "Limit of float: " << endl << float\_min << endl;

cout << "First number: ";

cin >> num;

cout << "Second number: ";

cin >> num2;

if (num2 == 0)

{

printf("Деление на 0\n");

return 0;

}

var.f = num;

var2.f = num2;

div.f = num / num2;

printf("\n\nRepresentation of %f is : \n", var.f);

printIEEE(var);

printf("\nRepresentation of %f is : \n", var2.f);

printIEEE(var2);

int\* arr1 = new int[32];

arr1 = MakeArr(arr1, var);

int\* arr2 = new int[32];

arr2 = MakeArr(arr2, var2);

int\* arr3 = new int[32];

arr3 = Division(arr1, arr2, arr3);

if (fabs(num / num2) >= float\_max)

{

cout << "\nПереполнение";

return 0;

}

else if (fabs(num / num2) <= pow(0.1, 25))

{

cout << "\nПотеря значимости";

return 0;

}

else

{

cout << "\n\nDecimal division is : \n" << div.f;

cout << "\nBinary division is : \n";

printIEEE(div);

}

return 0;

}

int\* MakeArr(int\* num, myfloat var) {

num[0] = var.raw.sign;

for (int j = 0; j < 32; j++)

num[j] = 0;

int k;

int n = var.raw.exponent;

int i = 8;

for (k = i - 1; k >= 0; k--) {

if ((n >> k) & 1)

num[8 - k] = 1;

else

num[8 - k] = 0;

}

n = var.raw.mantissa;

i = 23;

int z = 9;

for (k = i - 1; k >= 0; k--) {

if ((n >> k) & 1)

num[z] = 1;

else

num[z] = 0;

z++;

}

return num;

}

int\* Division(int\* arr1, int\* arr2, int\* sum) {

for (int i = 31; i >= 0; i--)

{

sum[i] = arr1[i] - arr2[i];

}

return sum;

}