электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 16

Моделирование целых чисел и арифметических операций над ними в троичной симметричной системе счисления с использованием длинной арифметики

***Строганов Ю.В.**, магистр*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»

Научный руководитель: Ковтушенко А.П., к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»

Консультант: Горин С.В., к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
gorin@bmstu.ru

Исследования по троичной системе счисления проводились в нашей стране ещё в 60-х годах прошлого века, в 1958 году была разработана и серийно выпускалась троичная вычислительная машина на безламповых элементах «Сетунь», а в 1969 году разработана «Сетунь-70». К сожалению, развития данное направление тогда не получило [4]. Но в 2010 году было замечено увеличение интереса к этому направлению. Моделирование чисел в троичной системе счисления является перспективным направлением для определения преимуществ троичного представления чисел по сравнению с двоичным.

Троичная система счисления (ТСС) – позиционная система счисления с целочисленным основанием, равным 3. Выделяют два вида: несимметричная и симметричная (уравновешенная). В несимметричной используются символы '0', '1', '2', а в симметричной '-1', 0', '1'. [1,2,3,4]

Троичная система счисления обладает рядом преимуществ перед двоичной:

1. большая плотность записи – основание ТСС ближе к числу e, чем основание двоичной системы счисления;
2. естественное и неизбыточное представление чисел со знаком [10];
3. ошибки округления – пропорционально не первой степени числа произведённых арифметических действий, а квадратному корню из этого числа [5];
4. округление числа производится простым отбрасыванием младших разрядов

[10];

5. погрешности округления в процессе вычислений взаимно компенсируются [10];

6. троичная система счисления включает в себя двоичную;

7. естественность – три состояния «да», «нет», «не знаю» ближе к человеческому восприятию мира;

Недостатки ТСС проявляются на этапе аппаратной реализации троичных логических элементов [7], но существуют реализации, демонстрирующие преимущества троичных логических элементов в тепловыделении и энергопотреблении [8]

Базовые элементы, используемые в троичной и двоичной системах счисления:

Бит – базовая единица измерения количества информации, равная количеству информации, содержащемуся в опыте, имеющем два равновероятных исхода, или это двоичный логарифм вероятности равновероятных событий или сумма произведений вероятности на двоичный логарифм вероятности при равновероятных событиях [9]. Байт – набор битов, количество битов в одном байте обычно равно 8, хотя в байте не всегда находилось 8 бит [4], и не всегда находится 32 бита в байте (Cray).

Дит – двойной бит. Дайт – двойной байт. В [7] рассмотрена возможность моделирования троичной логики на основе дитов.

Трит – троичная единица информации, может находится в трёх состояниях -1, 0, 1. Трайт – набор тритов, количество тритов в одном трайте не определено в стандарте, из-за отсутствия такого стандарта. В советской троичной ЭВМ «Сетунь» в трайте содержалось 6 трит [10], а изначально планировалось 4 [4], некоторые авторы высказывают предположение, что наилучшим количеством тритов будет 9, 27 или любая другая степень тройки.

Перевод в троичную систему счисления

По причине того, что троичная система счисления основана на том же позиционном принципе кодирования чисел, что и принятая в современных компьютерах двоичная система, однако вместо битов используются триты, благодаря чему положительные и отрицательные числа представляются однотипно: [6]

$$N = \sum \alpha_i \cdot 3^i = \alpha_{n-1} \cdot 3^{n-1} + \alpha_{n-2} \cdot 3^{n-2} + \ldots + \alpha_i \cdot 3^i + \ldots + \alpha_1 \cdot 3^1 + \alpha_0 \cdot 3^0$$

Для перевода из 10 СС в ТСС необходимо

1. Исходное десятичное число делится на основание ТСС (три).

2. В одну переменную записывается частное в виде целого числа, в другую – остаток.

3. Если частное не было равно нулю, то оно снова делится на три. Переменная, связанная со старым частным связывается с новым (прежнее частное теряется). Новый остаток добавляется в начало списка, где хранятся остатки.

4. П. 3 продолжает повторяться до тех пор, пока частное не станет равно нулю.

5. Остатки от деления, записанные в обратном порядке, представляют собой троичное представление заданного десятичного числа.

Таким образом получим:

$$79_{10} = [2221]_3 = 2 \cdot 3^3 + 2 \cdot 3^1 + 2 \cdot 3^2 + 1 \cdot 3^0 = 54 + 18 + 6 + 1 = 79_{10}$$

Разряды числа должны совпадать с порядковым номером в списке, что будет удобно для реализации арифметических операций над целыми числами в ТСС, потому п.5 не нужен, и число будет представляться так:

$$79_{10} = [1222]_3 = 1 \cdot 3^0 + 2 \cdot 3^1 + 2 \cdot 3^2 + 2 \cdot 3^3 = 1 + 6 + 18 + 54 = 79_{10}$$

Для перевода из несимметричной ТСС в симметричную ТСС необходимо провести небольшую замену:

$$2 \cdot 3^i = 1 \cdot 3^{i+1} - 1 \cdot 3^i$$

Таким образом, для стандартного представления получим:

$$79_{10} = [100\bar{1}1]_3 = 1 \cdot 3^4 + 0 \cdot 3^3 + 0 \cdot 3^2 + \bar{1} \cdot 3^1 + 1 \cdot 3^0 = 81 + 0 + 0 - 3 + 1 = 79_{10}$$

В предлагаемой реализации:

$$79_{10} = [1\bar{1}001]_3 = 1 \cdot 3^0 - 1 \cdot 3^1 + 0 \cdot 3^2 + 0 \cdot 3^3 + 1 \cdot 3^4 = 1 - 3 + 0 + 0 + 81 = 79_{10}$$

Для перевода из 10 СС в симметричную ТСС необходимо осуществить переводы из 10 СС в ТСС и из несимметричной в симметричную.

Моделирование

Для моделирования целых чисел в троичной симметричной системы счисления в качестве базового элемента будет использован трит, реализуемый типом char.

```
class long3simDigit
{
public:
        long3simDigit(void);
        long3simDigit(int x);
        ~long3simDigit(void);

private:
```

```
        int length;
        char* digit;

        char here(char el);
        char there(char el);
public:
        void setLength(int len);
        void setDigit(int el);
        void neg();
        void inc();
        void dec();
        void moveRight(int len);

        void balancing(long3simDigit &x, long3simDigit &y);
        long3simDigit sum(long3simDigit x,long3simDigit y);
        long3simDigit substruct(long3simDigit x,long3simDigit y);

        long3simDigit multiplication(long3simDigit x,long3simDigit y);
        long3simDigit multiplication2(long3simDigit x,long3simDigit y);

        long3simDigit division(long3simDigit x,long3simDigit y);
}
```

После перевода числа из 10 СС в троичную несимметричную требуется определить какое из трёх состояний будет записано в текущий разряд (функция *here*)

```
char here(char el)
{
        char res = el;
        switch(el)
        {
                case 2: res = -1; break;
                case 3: res = 0; break;
                case -2: res = 1; break;
                case -3: res = 0; break;
                default: res = el; break;
        }
        return res;
}
```

и какое дополнение необходимо передать в следующий разряд (функция *there*).

```
char there(char el)
{
        char res = 0;
        switch(el)
        {
                case 2: res = 1; break;
                case 3: res = 1; break;
                case -2: res = -1; break;
                case -3: res = -1; break;
```

```
            default: res = 0; break;
        }
        return res;
}
```

Определить количество разрядов необходимое для числа в троичной системе счисления при переводе из десятичной системы счисления можно по формуле:

$$len = [\log_3(целое\,положительное\,число)]$$

Для симметричной троичной системы счисления потребуется на 1 разряд больше.

```
void setDigit(int el)
{
        char base = 3;
        int i = el;
        int len = (int)ceil( log((double)(abs(i))) / log(3))+1;
        setLength(len);
        char xx=0;
        for(int j=0; j<length; j++)
        {
                char xxx = i%abs(base);
                char xx1 = xxx+xx;
                digit[j] =here(xx1);
                xx = there(xx1);
                i = i/abs(base);
        }
}
```

Операция сравнения

Операции сравнения необходимы для реализации практически всех математических операций: сравнение двух чисел, сравнение с нулём. Сравнение двух чисел при реализации разделяется на сравнение чисел с одинаковой длиной и с разной длиной.

Отличием троичной системы счисления от двоичной является использование вместо битов тритов — единиц информации, которые могут находится в одном из трёх состояний 1, 0 или третье состояние, для симметричной системы счисления это третье состояние равно -1, для несимметричной – 2. Преимущество заключается в том, что для оператора, например, сравнения двух чисел, сразу определяется одно из трёх состояний: больше, равно и меньше [1].

Сравнение чисел одинаковой длины [1]

Потритовое сравнение, разрядов двух чисел, сравнение начинается с наибольшего разряда, если текущий разряд первого числа больше текущего разряда второго числа, то

первое число больше, если наоборот, то первое меньше, если же значения текущих разрядов равны, то надо сравнивать разряды младше, если же все разряды совпадают, то эти числа равны.

Сравнение чисел разной длины [1]

Необходимо сравнить наибольшие разряды, если они оба положительные, то больше то, у которого больше длина, если оба отрицательные — наоборот, если разных знаков, то больше то, у которого значение разряда больше.

Арифметические операции

1. Взятие обратного знака.

Из-за особенностей представления чисел в симметричной троичной системе счисления взятие обратного знака числа производится поразрядным умножением на -1.

```
void neg()
{
	for(int i=0; i <length; i++) {digit[i] *= -1;}
}
```

2. Инкремент и декремент.

Эти операции близки, потому объединим их в общий *параграф*. При выбранном нами представлении числа, необходимо увеличить/уменьшить на единицу значение младшего разряда, после чего применить к текущему функцию here и there, и применять их до тех пор, пока значение there не станет равным 0.

```
void inc()
{
	char xx=0+1;
	for(int i=0; i<length;i++)
	{
		char xx1 = xx+here(digit[i]);
		digit[i] =here(xx1);
		xx = there(xx1);
		if(i==length-1 && xx!=0)
		{
			setLength(length+1);
		}
	}
}
void dec()
{
	char xx=0-1;
	for(int i=0; i<length;i++)
```

```
	{
		char xx1 = xx+here(digit[i]);
		digit[i] =here(xx1);
		xx = there(xx1);
		if(xx == 0)
			break;
	}
}
```

3. Сложение и вычитание.

Данные операции близки, потому объединим их в общий *параграф*. Для упрощения работы с этими операциями необходимо дополнить их до одинаковых длин нулями, после чего в каждом разряде выполняется действие сложения/вычитания с добавлением к результату добавочного значения функции there от предыдущего сложения, для самого младшего разряда добавочное значение there равно 0. Эти действия повторяются, пока не будет достигнут последний разряд и добавочное значение there не будет равно 0.

```
	long3simDigit sum(long3simDigit x,long3simDigit y)
{
	balancing(x,y);
	long3simDigit res = long3simDigit(x.digit, x.length);
	char xx = 0;
	for(int i=0; i<x.length;i++)
	{
		char xc = xx+y.digit[i]+x.digit[i];
		char xx1 = here(xc);
		res.digit[i] =xx1;
		xx = there(xc);
		if(i==length-1 && xx!=0)
		{
			res.setLength(length+1);
		}
	}
	return res;
}
long3simDigit substruct(long3simDigit x,long3simDigit y)
{
	balancing(x,y);
	long3simDigit res = long3simDigit(x.digit, x.length);
	char xx = 0;
	for(int i=0; i<x.length;i++)
	{
		char xd = x.digit[i];
		char yd = y.digit[i];
		char xc = xd-yd+xx;
		char xx1 = here(xc);
```

```
            res.digit[i]=xx1;
            xx = there(xc);
        }
        return res;
}
```

4. Умножение

Умножение двух целых чисел реализуется несколькими способами — используя сложение и как оригинальная операция. Для реализации *умножения на основе сложения* необходимо использовать функции декремента и сравнения, результирующее число приравнивается к первому числу и к результирующему числу прибавляется первое число, одновременно с этим уменьшается на 1 второе число, как только второе число станет равным 0 в результирующем числе будет результат умножения.

```
long3simDigit multiplication2(long3simDigit x,long3simDigit y)
{
        long3simDigit res = long3simDigit(x.digit, x.length);
        res.setLength(x.length+y.length);
        long3simDigit rr = long3simDigit(y.digit, y.length);
        bool bb = false;
        if(rr.compare0()<0)
        {
            rr.neg();
            bb = true;
        }
        while(rr.compare0()!=0)
        {
            res.sum(x);
            rr.dec();
        }
        if(bb)
            res.neg();
        rr.~long3simDigit();
        return res;
}
```

При реализации операции *умножения, как оригинальной операции* необходимо учитывать смещение произведения на одно число, как в алгоритме «умножение столбиком», с учётом его порядкового номера [1].

```
long3simDigit multiplication(long3simDigit x,long3simDigit y)
{
        long3simDigit res = long3simDigit();
        res.setLength(x.length+y.length);
        for(int i=0; i<x.length; i++)
        {
            long3simDigit rr = long3simDigit(y.digit, y.length);
```

```
                rr.setLength(rr.length+1);
                for(int j=0; j<rr.length; j++)
                {
                        char xc = rr.digit[j]*x.digit[i];
                        rr.digit[j] = xc;
                }
                rr.moveRight(i);
                res.sum(rr);
                rr.~long3simDigit();
        }
        return res;
}
void moveRight(int len)
{
        long3simDigit res = long3simDigit();
        res.setLength(length+len);
        for(int i=len;i<res.length;i++)
        { res.digit[i]=digit[i-len]; }
        setLength(res.length);
        memcpy(digit, res.digit, sizeof(char)*length);
        res.~long3simDigit();
}
```

5. Деление.

Деление двух целых чисел можно реализовывать на основе вычитания, увеличивая результат, пока число не поменяет знак, как только число меняет знак необходимо однократно увеличить значение результата на единицу. Иными способами деление реализуется с в [1,3] и [11].

Заключение

Описанное представление позволяет работать с большими целыми числами в троичной симметричной системе счисления, наибольшее число, которое можно записать в текущей реализации $3^{2147483647}+3^{2147483646}+...3^1+3^0$, реализация алгоритма перевода ограничивает наибольшее переводимое число значением 2147483647, чтобы преодолеть этот барьер необходимо воспользоваться десятичной длинной арифметикой. Рассмотренный способ моделирования целого числа позволяет упростить реализацию арифметических операций над целыми числами (возможно, упростит распараллеливание каждой операции). Для операций сравнения преимуществ с [1] не даёт.

Список литературы

1. Рамиль Альварес Х. Алгоритмы троичной арифметики. М.: Фонд «Новое

тысячелетие», 2012. 20 с.

2. Рамиль Альварес Х. Троичный алгоритм извлечения квадратного корня // Программные системы и инструменты. Тематический сборник. 2010. № 11. С. 98-106.
3. Рамиль Альварес Х. Деление целых чисел в троичной симметричной системе // Программные системы и инструменты. Тематический сборник. 2011. № 12. С. 228-234.
4. Виртуальный компьютерный музей, Сетунь. Режим доступа: http://www.computer-museum.ru/histussr/12.htm (дата обращения 01.12.2014).
5. Кушнеров А. Троичная цифровая техника. Ретроспектива и современность. Университет им.Бен-Гурион. Беар-Шева. Израиль. Режим доступа: http://314159.ru/kushnerov/kushnerov1.pdf (дата обращения 01.12.2014).
6. Брусенцов Н.П., Рамиль Альварес Хосе. Троичные ЭВМ «Сетунь» и «Сетунь-70», Режим доступа: http://www.computer-museum.ru/histussr/setun_b.htm (дата обращения 01.12.2014).
7. Никитин А.В. Математика счетной логики. Режим доступа: http://www.trinitas.ru/rus/doc/0016/001b/00161278.htm (дата обращения 01.12.2014).
8. Ившин П., Леготин С., МурашёвВ. Базовые троичные логические элементы. Снижение энергопотребления // Электроника: НТБ. РИЦ «Техносфера» Электрон. журн. №4. 2010. Режим доступа: http://www.electronics.ru/journal/article/72 (дата обращения 01.12.2014).
9. Shannon C.E. A Mathematical Theory of Communication. Available at: http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf, accessed 01.12.2014.
10. Сидоров С.А., Владимирова Ю.С., Троичная виртуальная машина // Программные системы и инструменты. Тематический сборник № 12. 2011. С. 46-55.
11. Кнут Д. Искусство программирования в 4т. Т. 2. Получисленные алгоритмы. 3-е изд. М.: «Вильямс», 2007. 832 с.