DOI: 10.25586/RNU.V9187.18.04.P.106 УДК 51.511-33:519.67

Г.Т. Мачарадзе, Е.А. Морозова

АЛГОРИТМ БЫСТРОГО ВЫЧИСЛЕНИЯ ОБРАТНОГО КВАДРАТНОГО КОРНЯ И ЕГО АКТУАЛЬНОСТЬ

В данной статье приводится обобщенный алгоритм для быстрого вычисления обратного квадратного корня, а также коэффициент ускорения алгоритма быстрого вычисления по отношению к алгоритму, использующему стандартные функции, что позволит сделать вывод о пригодности данного алгоритма на современных быстродействующих процессорах.

Ключевые слова: *быстрый обратный корень, алгоритм быстрого вычисления обратного квадратного корня, алгоритм быстрого вычисления показательных функций.*

G.T. Macharadze, E.A. Morozova

ALGORITHM OF FAST INVERSE SQUARE ROOT AND ITS RELEVANCE

In this article a generalized algorithm for rapid calculation of the inverse root is provided, and a coefficient of algorithm acceleration of rapid towards to the algorithm, which uses the standard functions is provided, too. This will enable to make a conclusion about the suitability of this algorithm on modern quick-action processors.

Keywords: *fast inverse square root, algorithm of fast calculation of inverse root, algorithm of fast calculation of exponential function.*

При разработке различных программных систем иногда встает вопрос о возможности ускорения её работы. В системах реального времени существует квант времени, в течение которого необходимо успеть обработать данные, так как по истечении этого кванта на вход системы придут новые данные. Если эта система осуществляет обработку графической информации, то большую часть времени она работает с различными поверхностями, а точнее – с их нормалями. Нормаль – это вектор единичной длины, перпендикулярный данной поверхности. При вычислении нормали используется обратный квадратный корень. Таким образом, ускорив операцию обратного квадратного корня, можно добиться ускорения в работе программной системы в целом.

В ЭВМ вещественные числа представляются в формате IEEE754, согласно которому число представляется в виде бита знака, смещенного порядка и мантиссы. Для одинарной точности представления (соответствует типу float в языке С) справедливо следующее (рис. 1).

s	E – порядок	M – мантисса
1 бит	8 бит	23 бит

Рис. 1. Формат IEEE754

Формула для перехода из IEEE754 к десятичному формату:

$$x = (-1)^s \cdot (1+m) \cdot 2^e, \quad (1)$$

где $m = \frac{M}{L}$, $e = E - B$, $L = 2^{23}$, $B = 127$ – смещение порядка.

Например, для десятичного числа 12.75 имеем:

$$12.75_{10} = 1100.11_2 = 1.10011 \cdot 2^3,$$

тогда $E = 3 + 127 = 130_{10} = 10000010_2$, а $M = 10011_2$, т.е. получим (с учетом бита знака $s = 0$): $01000001010011000000000000000000_2$.

Если интерпретировать порядок и мантиссу числа x как целые числа, то целочисленное представление числа x можно записать в виде

$$INT(x) = M + L \cdot E. \quad (2)$$

Пусть необходимо вычислить значение функции:

$$y = f(x) = x^p, \ |p| = \frac{1}{2^k}, \ k \in \mathbb{N}.$$

Прологарифмируем обе части равенства по основанию 2:

$$\log_2 y = \log_2 x^p = p \cdot \log_2 x.$$

Заменим x и y, используя выражение (1):

$$\log_2 (1+m_y) 2^{e_y} = p \cdot \log_2 (1+m_x) 2^{e_x}.$$

Упростим выражение, раскрыв логарифм произведения:

$$\log_2 (1+m_y) + e_y = p \cdot \left[\log_2 (1+m_x) + e_x \right]. \quad (3)$$

Рассмотрим функцию $g(m) = \log_2 (1+m)$. В нашем случае мантисса нормализована, т.е. справедливо неравенство: $0 \le m < 1$. На полуинтервале [0, 1) функция $g(m)$ примерно ведет себя, как линейная функция $m + \varepsilon$. Меняя параметр ε, мы можем улучшить точность приближения, $\varepsilon = 0.0450465$ дает лучшее приближение.

Таким образом, можно записать:

$$\log_2 (1+m_x) \approx m_x + \varepsilon. \quad (4)$$

Преобразуем (3), используя (4):

$$e_y + m_y + \varepsilon = p \cdot (e_x + m_x + \varepsilon).$$

Перейдем от e, m к E, M по равенствам, указанным в (1):

$$E_y - B + \frac{M_y}{L} + \varepsilon = p \cdot \left(E_x - B + \frac{M_x}{L} + \varepsilon \right).$$

Умножим обе части равенства на L и перенесем все слагаемые, не связанные с y, в правую часть равенства:

$$L \cdot E_y + M_y = p \cdot (M_x + L \cdot E_x) + (1-p) \cdot (B - \varepsilon) \cdot L.$$

Используя (2), получим:

$$INT(y) = p \cdot INT(x) + (1-p) \cdot (B - \varepsilon) \cdot L = p \cdot INT(x) + \text{const}. \quad (5)$$

Значение $|p| = \frac{1}{2^k}$ выбрано не просто так. Данный выбор позволяет заменить операцию умножения на сдвиг числа вправо или влево, в зависимости от знака p. Таким образом, целочисленную интерпретацию результата можно получить, сдвинув аргумент функции на k разрядов (вправо/влево) и прибавив константу. Данное приближение дает точность порядка 4%. Повысить точность можно с помощью метода Ньютона.

Рассчитаем значение константы из выражения (5) при $p = -\frac{1}{2}$:

$$\text{const} = \left(1 + \frac{1}{2}\right) \cdot \left(127 - 0.0450465\right) \cdot 2^{23} = 1597463007_{10} = 0\text{x}5\text{F}3759\text{DF}.$$

Для уточнения результата используется метод Ньютона [2]. Формула для следующего приближения имеет следующий вид:

$$y_{n+1} = y_n - \frac{f(y_n)}{f'(y_n)}. \quad (6)$$

Применим данный метод для функции $f(x) = y = \frac{1}{\sqrt{x}}$;

$$f(y,x) = \frac{1}{y^2} - x = 0;$$

$$f'^{(y,x)} = \frac{-2}{y^3}.$$

Подставив производную в (6), получим формулу для второго приближения результата:

$$y_{n+1} = y_n - \frac{\frac{1}{y_n^2} - x}{\frac{-2}{y_n^3}} = y_n \cdot \left(1.5 - 0.5x \cdot {y_n}^2\right).$$

Данное уточнение повышает точность результата до 0.18%.

Реализация алгоритма вычисления обратного квадратного корня на языке С:

```
inline float Q_Sqrt_Inverse(float x) {
        const float half = 0.5f * x; //запоминаем половину для последующей аппроксимации
        int y = * (int *)&x; //приводим к int
        y = 0x5F3759DF - (y >> 1); //вычисляем первое приближение
        x = * (float *)&y; //приводим к float
        x = x * (1.5f - (half * x * x)); //аппроксимация метода Ньютона
        return x;
}
```

В таблице 1 приведено полученное время выполнения функции по сравнению с функциями, использующими стандартные математические функции sqrt и sqrtf.

Таблица 1

Время выполнения функций

	sqrt	**sqrtf**	**Q_Sqrt_Inverse**
Без уточнения результата	0.170480	0.140246	0.060292
С уточнением результата	0.170480	0.140246	0.083915

В таблице 2 приведены коэффициенты ускорения приведенного алгоритма по отношению к стандартным функциям sqrt и sqrtf.

Таблица 2

Коэффициенты ускорения алгоритма по отношению к стандартным функциям

	Ускорение относительно sqrt	**Ускорение относительно sqrtf**
Без уточнения результата	2.83	2.33
С уточнением результата	2.03	1.67

Резюмируя полученные данные, можно сделать вывод, что в настоящее время, даже с учетом существования блоков FPU (Floating Point Unit), которые на аппаратном уровне реализуют большинство операций с плавающей точкой, приведенный алгоритм всё еще может использоваться по назначению. При точности 0.2% можно получить ускорение данной операции минимум на 50%.

Литература

1. Fast inverse square root [Электронный ресурс]. – URL: https://en.wikipedia.org/wiki/Fast_inverse_square_root

2. *Амосов А.А., Дубинский Ю.А., Копченова Н.В*. Вычислительные методы. – СПб. : Лань, 2014.

References

1. Fast inverse square root [Электронный ресурс]. – URL: https://en.wikipedia.org/wiki/Fast_inverse_square_root

2. *Amosov, A.A., Dubinskiy, Yu.A., Kopchenova, N.V.* Vychislitel’nye metody. – SPb. : Lan’, 2014.