第32卷 第23期 计 算 机 工 程 2006年12月
Vol.32 No.23 Computer Engineering December 2006
·开发研究与设计技术· 文章编号：1000—3428(2006)23—0271—02 文献标识码：A 中图分类号：TP312

32位嵌入式系统中扩展精度数学算法实现

聂胜伟，陆士强，程恩惠
(华东计算技术研究所，上海 200233)

摘 要：提出了一种32位嵌入式系统中应用的扩展精度数学算法。适用于缺乏数字协处理器硬件支持并且软件浮点运算达不到系统时间要求的系统。算法运算数据精度高、扩展性好。介绍了32位乘法、除法、开方算法以及64位加法、减法、乘法算法。
关键词：扩展精度数学算法；32位乘法算法；32位除法算法；32位开方算法；64位乘法算法

Implementation of Extended Precision Mathematical Algorithm in 32-bit Embedded System

NIE Shengwei, LU Shiqiang, CHENG Enhui
(East China Research Institute of Computing Technology, Shanghai 200233)

【Abstract】This paper presents a mathematical algorithm for extended precision in 32-bit embedded system. The algorithm can be applied in the system without the support of mathematical coprocessor and provides high precision. It provides 32-bit multiplication algorithm implementation, 64-bit multiplication algorithm implementation, 32-bit division algorithm and 32-bit square root algorithm implementation.
【Key words】Extended precision mathematical algorithm; 32-bit multiplication algorithm; 32-bit division algorithm; 32-bit square root algorithm; 64-bit multiplication algorithm

特定环境的嵌入式系统不仅要求各任务执行无误、准时，而且还要求数据结果有很高的精度。本文介绍的算法是在32位嵌入式系统中实现的。这种嵌入式环境一般有以下要求：

(1)硬实时嵌入式系统；

(2)系统缺乏数学协处理器硬件支持而且软件浮点仿真运算达不到系统的实时性要求，时间精度要求高，算法速度要求快；

(3)系统中的数据跨度大；

(4)运算结果数据精度要求高。

1 算法介绍

算法的思想是用整型数代替浮点数来进行计算。一个整型数的最大表示范围取决于所给定的字长。显然，字长越长，所能表示的数的范围越大，精度也越高。本文涉及嵌入式系统的CPU采用的是32位的处理器，一般给定的字长是32位。如果32位数据不能满足精度要求，在32位处理器上可以通过本文介绍的算法实现64位运算。

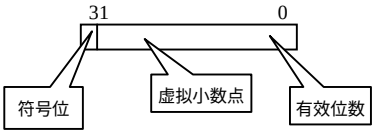

图1 32位数据格式

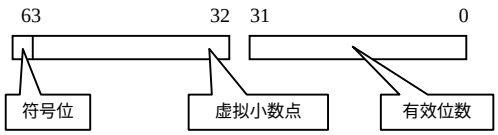

图2 64位数据格式

通过在32位或者64位数据中设定小数点的不同位置，可以表示不同大小和不同精度的小数。代替浮点算法要解决的最大问题是精度和速度的问题。采用的数据以2的补码的形式表示，数据的定标采用S表示法。定义的32位和64位数据格式如图1和图2所示。

要把浮点数转换成整型数，首先需确定小数点位置，然后乘以相应的比例因子。要确定小数点位置，需要事先利用数学仿真估计数据范围。估计的数学范围跨度越大，数据精度越低，反之亦然。

假设数据$|a|<2^x$。首先转化为绝对值小于1的数，即

$$|\frac{a}{2^x}|<1$$

由于最高位是符号位，因此对32位数据转化后的最大精度的整型数是

$$\frac{a}{2^x}\times 2^{31}=a\times 2^{31-x}$$

64位数据转化后的最大精度的整型数是

$$\frac{a}{2^x}\times 2^{63}=a\times 2^{63-x}$$

将浮点数格式化为需要的整型数的思想是将处理后绝对值小于1的浮点数循环乘2，然后保存整数位，最后得到的整数就是我们需要的整型数。在32位处理器中保存浮点数转化后的32位整型数，只要直接定义整型变量即可。而保存浮点数转化后的64位整型数，需要定义整型数组来保存。

本文主要介绍32位数据乘法、除法、开方运算以及64位加法、减法、乘法运算。更高位数的运算可以作相应的扩展。对于32位数据的加法、减法运算，可以直接用编译器提供的加、减指令进行。

作者简介：聂胜伟(1973－)，男，工程师，主研方向：嵌入式系统；陆士强，高工；程恩惠，助工
收稿日期：2006-01-04 **E-mail：**niedak@163.net

2 算法的具体实现

2.1 32 位乘法

32 位乘法的思想是：将两个 32 位的整数相乘，结果是 64 位，取高 32 位。乘法过程如图 3 所示。

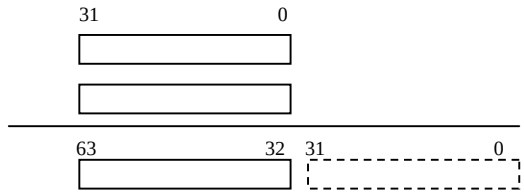

图 3 32 位乘法的思想

假设$a<2^x$,$b<2^y$，则

$$a\times2^{31-x}\times b\times2^{31-y}=a\times b\times2^{62-(x+y)}$$

乘法结果取高 32 位。由于抛掉了低 32 位，因此最后结果应该乘以 2 才能符合 32 位数据格式，即

$$a\times b\times2^{62-(x+y)-32}\times2=a\times b\times2^{30-(x+y)}\times2=a\times b\times2^{31-(x+y)}$$

2.2 32 位除法

在 32 位除法中如果是大数除以小数，则直接用编译器提供的除法指令；如果是小数除以大数，则用移位相减的方法。下面介绍的算法是将数据 a 直接除以 b，并且|a|<|b|。如果不符合上述条件，可以将数据进行相应的格式化。算法如下：

```
保护寄存器；
结果 c 赋值为 0；
IF b>30 000 000h THEN
    防溢出处理；
ENDIF
计数器赋值 30；
LOOP
    IF a-b>0 THEN
        a=a-b；
        a 左移一位；
        c 与 1 相或；
    ELSE
        a 左移一位；
    ENDIF
    c 左移一位；
    计数器减 1；
EXIT WHEN  计数器等于 0
END LOOP
防溢出处理；
结果符号处理；
c 乘以 2；
恢复寄存器；
```

因为要对 b 进行防止溢出处理，所以只循环 30 次。因为循环了 30 次，所以应该对结果乘以 2 进行补偿。此算法的特点决定了最后结果乘以 2 不会溢出。

2.3 32 位开方运算

开方运算一般采用泰勒公式展开来近似计算，

$$\sqrt{a}=\sqrt{1+x}$$

$$x=a-1=1+\frac{x}{2}-\frac{x^2}{8}+\frac{x^3}{16}-\frac{5x^4}{128}+\frac{7x^5}{256}=1+\frac{x}{2}-0.5\left(\frac{x}{2}\right)^2+0.5\left(\frac{x}{2}\right)^3-0.625\left(\frac{x}{2}\right)^4+0.875\left(\frac{x}{2}\right)^5$$

其中 $0.5\le x<1$。

如果在实际应用中选择的初值与理论值接近，也可以采用牛顿迭代法完成。对于给定的正数 a，平方根 $\sqrt{a}$ 的迭代公式是

$$X_n=X_{n-1}-\frac{X_{n-1}^2-a}{2X_{n-1}}$$

由上述迭代公式所产生的序列$\{X_n\}$必定收敛于根 $\sqrt{a}$。

算法的关键是选取一个合适的迭代中止条件。如果选取$|x_{n-1}|=|x_n|$，那么条件过于苛刻。经过实验证明迭代几步后已经达到较高的精度值，其它大部分迭代是在理论值左右震荡，震荡值是 ±1。选取上述苛刻条件会耗费大量的运算时间。在实际应用中一般选取最小的迭代中止条件$|x_{n-1}-x_n|<2$。这样节省了软件运行时间，从而提高了软件的可靠性。算法如下所示：

```
定义变量 oldx；
oldx=x；
计数器赋初值 1；
DO WHILE  计数器小于最大迭代步数
    x=x-(x-a)/2*x；
    IF |oldx-x|<2 THEN
        x 即为 a 的平方根；
    ELSE
        oldx=x；
        计数器加 1；
    ENDIF
ENDDO
```

2.4 64 位加法和减法

64 位加法的思想是：首先低 32 位相加，然后高 32 位相加，如果低 32 位相加有进位，则高 32 位相加时加上进位。Intel 80X86 指令系统直接采用 add 和 adc 指令实现。

64 位减法的思想是：首先低 32 位相减，然后高 32 位相减，如果低 32 位相减有借位，则高 32 位相减时减去借位。Intel 80X86 指令系统直接采用 sub 和 sbb 指令实现。

2.5 64 位乘法

64 位乘法是将 2 个 64 位的数据相乘，结果是 128 位，结果根据需要取高 64 位或者整个 128 位。假设 2 个 64 位数据 a、b，其中 a 是由 2 个 32 位数据 a0 和 a1 组成，b 是由 b0 和 b1 组成。结果用 ab 表示，可以根据需要选取 ab3 和 ab2 高 64 位或者选取 ab3、ab2、ab1 以及 ab0 全部 128 位。乘法过程如图 4 所示。

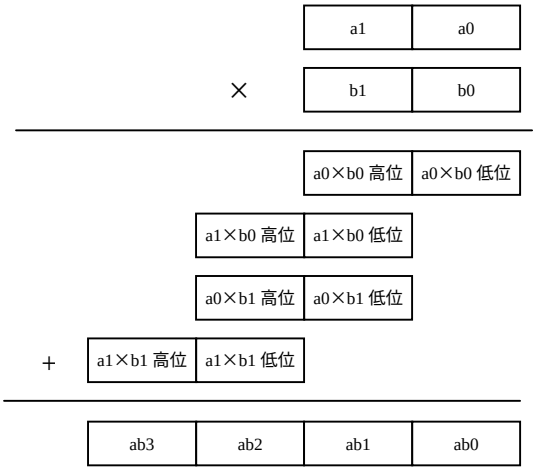

图 4 64 位数据乘法

（下转封三）