第23卷第4期
2006年12月

广东工业大学学报
Journal of Guangdong University of Technology

Vol. 23 No. 4
December 2006

单片机C语言编程中多位乘法运算问题探讨

严克剑，张　森，黄先伟

（广东工业大学 自动化学院，广东 广州 510090）

摘要：本文指出了人们运用C51实现多字节乘法运算时可能忽视的一个问题. 通过分析该问题出现的原因，提出了中间变量法、分步运算法和强制数据类型转换法等3种解决方法，实验表明这些方法是可行的.

关键词：C51；单片机；乘法运算

中图分类号：TP332.3　　**文献标识码**：A　　**文章编号**：1007-7162(2006)04-0023-04

C语言既具有一般高级语言的特点，又能直接对计算机的硬件进行操作[1]. Keil C51是德国Keil Software公司出品的51系列兼容单片机C语言软件开发系统. 与汇编相比，C语言在功能、结构性、可读性、可维护性上有明显的优势，因而易学易用. Keil C51继承了C语言对数据有很强的表达能力的优点，具有丰富的运算符，在算术运算和逻辑运算上更体现了汇编不可比拟的优点. 由于C51语言具有强大的数据处理能力和数学运算库函数，当涉及到复杂的数学运算，使用C51语言往往会比较方便. 在一般情况下，由C51编译生成的代码不论长度还是程序运行速度均能适应程序要求. 利用C51开发单片机系统，不但可以使编程工作量大为减少，而且使软件维护、修改亦变得非常方便[2].

1　问题提出

在Keil C51中，bit和unsigned char型变量是机器语言直接支持的，因此两个unsigned char类型的数据相乘恰好与汇编指令中的"MUL AB"相符. 使用Keil C51做乘法运算编程，往往只需要利用乘法表达式来编译，只要编译通过Keil C51就会自动地寻找寄存器把运行结果存放在里面. 因此，在运用C51来完成乘法运算的时候，编程员通常就是编写一个乘法表达式和把乘积结果赋值给一个存储单元就可以了. 而如果乘法运算中的乘数不是unsigned char型变量，就算是简单的一个乘法表达式也需要调用C51中的库函数；如果乘数是大于255的常量，C51自动地把这个乘数以相应的类型储存起来，由于unsigned char是8位的，大于255的常量和其他类型的变量一样也需要调用库函数. 因此，如果在这些情况下还仅仅是用乘法表达式来编译的话，得到的结果就可能与实际结果不相等. 笔者在编程的时候曾经遇上上述情况.

例如以下有一段程序，运行之后所得结果gg的值不等于9 * 10000所得的结果.

```
#include < AT89X55. H >
main( )
```

收稿日期：2005-12-12

项目基金：广东省科技计划项目(2003C102022)

作者简介：严克剑(1982-)，男，硕士研究生，主要研究方向为控制网络集成.

```
{
 unsigned charaa;
unsigned long gg;
aa = 9;
gg = (aa * 10000);}
```

本例的实际运算结果应该是90000,但是通过以上程序得到的结果却是24464. 通过 Keil C51 单步运算结果还是24464,与实际结果相差65536,转换为16进制数刚好是10000H,显然这是一个溢出问题. 然而通过 Keil C51 的单步运算可以知道溢出标志 OV 为零,因此不能运用溢出标志判断,以得到真正的结果.

2 问题的分析

Keil C51 编译器在进行优化处理时,总是企图用工作寄存器来存放局部变量的. 在执行 aa =9 语句时,根据 C51 的存储规则,可以得到 R7 =9. 执行 aa * 10000 时,Keil C51 要求算术运算的数据类型一样的,这里 10000 是两个字节的 unsigned int 方式储存的,这样 aa 的储存必需扩展成为两个字节的 unsigned int 方式. 因此,Keil C51 用两个寄存器存储 aa =9,先存低字节,后存高字节,分别为 R7 =09H、R6 =00H,储存 10000 的寄存器 R5、R4 分别是 10H、27H. 这样的话,执行 aa * 10000 就变成了两个双字节数相乘,根据 C51 的运算规则,生成代码得到的反汇编语言代码如下:

```
MOV   A,R7
MOV   B,R5
MUL   AB
MOV   R0,B
XCH   A,R7
MOV   B,R4
MUL   AB
ADD   A,R0
MOV   R6,B
XCH   A,R6
MOV   B,R5
MUL   AB
ADD   A,R6
MOV   R6,A
RET
```

由此可知,代码少运算了一次,也就是原来存放在 R6 与 R4 里面的数据没有进行乘法运算,而且没有把第三次运算(原来存放在 R5 与 R6 的数据相乘)得到的结果的高8位储存起来,运算得到的结果的低8位存放在 R7,次低8位存放在 R6. 因此,运算的结果不正确.

3 解决方法

针对上述所分析的问题,笔者提出了几种可行的解决方法. 以下所列举的3种是已经通过

实验验证的解决方法.

1) 引入中间变量法

在运算的时候加入一个中间变量可以解决这个问题.要求这个中间变量数据类型必需能够存储相乘之后得到的结果,上面的程序可以改为:

```
#include < AT89X55. H >
main( )
  {
    unsigned char aa;
    unsigned long gg,bb;
    aa =9;
    bb = 10000;
    gg = ( aa * bb) ;
}
```

由程序可以看出,这里 aa 是 char 而 bb 是 long 类型.根据 C51 的运算规则,aa 先变成与 long 一样的数据类型,运行的结果也是 long 类型的,C51 能够自动存储 9 * 10000 这个数据,因此,结果不会出错.

2) 分步运算法

编程人员也可以采取分步运算的方法,把一个 16 位的数据分成两个数据的乘积,而这两个数据都是 8 位数据,结果储存在可以存储相乘之后得到的结果的储存单元中.以上程序可以改为:

```
#include < AT89X55. H >
main( )
  {
    unsigned char aa;
    unsigned long gg;
    aa =9;
    gg = ( aa * 100) ;
    gg * =100;
}
```

这个程序先把 10000 分成两个 8 位数的乘积 100 * 100,然后逐个相乘.由于 aa 与 100 都是 unsigned char 数据类型,直接调用汇编语言的"MUL AB"指令,再把结果送到存储单元为 long 类型的 gg,在运行 gg * =100 语句的时候,C51 先以 long 类型储存数据 100,然后调用库函数,得到的结果为 long 类型,可以存储 9 * 10000 这个数,最后运算结果正确.

3) 强制类型转换法

C51 中,圆括号"( )"可以作为强制类型转换运算符,顾名思义,它的作用就是将表达式或者标量的类型强制转换成为所制定的类型.C51 中,有两种数据类型转换方式,一种是隐式转换,另外一种是显式转换.

隐式转换有以下规则:

① 把所有的 char 类型的操作数转换成为 int 类型.

② 当强制类型转换运算符连接两个操作数时,如果有一个是 float 类型的,另外一个也转换

成float类型;如果有一个是long类型的,另外的一个要转换成long类型;如果有一个是unsigned类型的,另外一个也转换成为unsigned.

③ 如果强制类型转换运算符连接的两个数据是对变量的赋值,则仅将赋值号右边的表达式类型转换成为赋值号左边的类型.

隐式转换式在对程序进行编译时由编译器自动处理的,本例中应用隐式转化方式不能得到正确的结果.运用显式类型转换方式,通过强制类型转换运算符把aa强制转换成为可以储存运算结果的类型.

```
main( )
  {
    unsigned char aa;
    unsigned long gg;
    aa =9;
    gg = ( ( unsigned long) aa) * 10000;
}
```

程序中,首先把aa转换成为可以储存9 * 10000的数据类型,得到的结果是正确的.

对比3种方法,引入中间变量法可以用在内部寄存器足够使用的时候,如果程序对内部寄存器的使用本来就紧张,则不推荐使用.分步运算法必须要清楚乘数的值,多字节的乘数值的大小完全清楚或者是已知的常数时,推荐使用.使用强制数据类型法时,需要注意的是把少字节类型的数据转换成多字节.

4 结束语

Keil C51生成目标代码的效率非常高,很多语句生成的汇编代码很紧凑,而且容易理解.乘法运算时出现以上这种情况,并不能掩盖其不可比拟的优点.如果在使用Keil C51作算术运算时,注意一下溢出问题,将会更好地发挥Keil C51运算能力强的优点.

参考文献:

[1]徐爱钧,彭秀华.单机机高级语言C51Windows环境编程与应用[M].北京:电子工业出版社,2001.
[2]胡士强,张小英,王改名,等.利用单片系统应用软件[J].工业科技,1999,16(2):63-67.

Study of Multiplicative Prog Ramming Problem in C51

YAN Ke-jian, ZHANG Miao, HUANG Xian-wei
(Faculty of Automation, Guangdong University of Technology, Guangzhou 510090, China)

Abstract: This text points out a problem that people may neglect in the use of C51 to operate multiplication of multi-bytes. It analyzes the reason why this prodlem appears. Then it puts forward three methods to resolve it, and these methods are proved to be feasible by experiments.

Key words: C51; single chip; muitiplication operation