虚拟变量在C语言编程中的妙用

赵仁健　熊　平　(四川大学)

摘要:本文讨论了在 *Turbo C* 中使用虚拟变量直接指代硬件寄存器的方法和注意事项。探讨了使用虚拟变量在某些数据处理中的技巧,以及如何使用虚拟变量以提高程序运行速度。

1.概述

Turbo C 提供了一种功能:可以使用虚拟变量(或称"伪变量")直接进入硬件寄存器。例如,当用__DX,__SS,在 Turbo C 中虽是一个变量,却实际上对应了 DX,SS 寄存器。这属于 C 语言编程的高级技术,一般在使用 BIOS 或 DOS 系统功能时须用到它。

但是,由于虚拟变量使我们深入到计算机内部各寄存器的值,使程序员得以与计算机的本质打交道,故巧妙地使用它,可以使很多编程难关迎刃而解。

使用虚拟变量功能,再加上一些与内存地址打交道的功能,例如用"*"可知道任何数据的内存地址,用 Movedata 等语句可移动或拷贝内存、显示存储器(VRAM)中的内容,从而使程序员可以完成很多用高级语言难以完成的特殊功能。可以指定对寄存器、内存作操作,从本质上控制程序的流向和运行状态,使 C 语言象汇编语言一样深入到计算机内部,进行低级编程。

2.使用虚拟变量的方法和注意事项

使用虚拟变量一般采用如下方法:

(1)先不用虚拟变量,用真实变量将程序调试通过,达到预期编程目的;

(2)再试着将已调通的程序中的真实变量逐步用虚拟变量代替,使运行结果仍保持不变。

虽然使用虚拟变量有很多便利之处,但由于它直接对寄存器操作,故要特别小心。要做到以下两点:

(1)虚拟变量的使用不要扰乱 C 语言主程序的正常运行。首先,段地址寄存器和专用寄存器,多数是不容许编程者随意更改其内容的。除非为了达到某些十分特别的高级编程效果,一般只能读出其值来使用。例如,__CS、__SS 和__BP 等虚拟变量,它们对应的寄存器值一般是 C 源程序编译成的执行文件具体执行时分配的,而其值在程序运行过程中从头到尾都是由计算机自己"规划的",不容许随意写入新内容改变其值。而对 CS,DS 等值的读出,有时能帮助程序员完成某种特殊的高级功能。其次,C 语言时时要用到的通用寄存器,这些寄存器不是从头到尾作"控制"程序的状态用的,例如__CX、__AX 等,对它们也要慎重使用。C 语言的很多语句都要用到它们,在用完后的间隙中,程序员才可以将它们作虚拟变量使用。

(2),选择合适的虚拟变量存储有用值,避免其值被 C 语言其它操作改变。C 编译后产生的代码频繁地使用寄存器。在调用函数或进行运算等操作时,会使寄存器内的值改变。故某个虚拟变量用来存储有用值后,不会保存很久。最好是在使用寄存器之前才向它赋值;而寄存器取得所需值后立即用伪变量读出。例如,__AX 用作累加器,__DX 用作 16 位乘法的高位字等。如下例,__AX 之值已不再是 9:

```
int m=1;
```

```
__AX=9;
m*=2;
printf("%d",__AX);
```

一般地说,__BX、__DX、__SI、__DI 等虚拟变量使用起来安全系数大。但使用它们时,一定要视具体情况而定,使它们不要扰乱 C 语言程序的正常运行、同时用它们保存的有用数据不被 C 的其它操作所改变。

3.使用虚拟变量提高运行速度

由于使用虚拟变量,类似于"寄存器变量"的使用,相当于直接使用硬件寄存器。故我们可以巧妙地使用这些虚拟变量,在某些情况下大大提高速度。当某个数据要频繁地被使用时,我们把它存在虚拟变量中,比用真实变量快得多。例如:C 中某个真实变量"Value"是整型,它放在内存中 8000:0000 处。如果使用这个变量时,计算机必须先找到 8000:0000 内存地址,取出 Value 之值放入某一寄存器,作操作后,再找到 8000:0000 地址放回该数之值。而如果改用__DX 存储 Value 之值,其值则仅在寄存器内传送,速度大大提高。

在 C 中,使用特别频繁的数据或变量,可以用虚拟变量存储。如循环变量、循环终止的比较变量,循环中用到的变量。看下面几行语句:

```
int j=0;
for (i=1,i<1000;i++)
                j+=a[i];
```

改为如下程序:

```
__DX=0;
for(__CX=1;__CX<1000;__CX++)
          __DX+=a[__CX];
```

速度可提高很多倍。

4.用字符型变量存储数据时虚拟变量的使用

在很多大数据量的处理中,计算机内存空间需要尽量地节约,以免不够使用。笔者在编制一信息处理应用程序时就遇到了一个难题:从计算机接口采集进来的大批整型数需要存入内存中参与各种运算、处理,但如果每一个数用一个短整型变量来表示的话,内存容量不够,开不出这么大空间的一个整型数组。

虽然可以将采集数组写入数据文件,在程序处理数据时分段取了数据,分多次处理,然后衔接在一起。但这样给编程造成了很多麻烦,特别是要多次用到这些原始数据时。例如,对原始数据剔除平均值,我们必须分多次读入原始数据的一、二、三…各段,求和后求得平均值,再分多次将原始数据读入减去平均值。

好在我们的原始数据变动范围不大,由于是 8 个 bit 的 A/D 板采样,数据范围为 0～255 或−128～127。这促使我们想办法在内存中仅用 8 个 bit,即一个字符型变量来表示一个原始数据。

但字符型变量存储的原始数据,如果直接放在数学表达式中参与运算时,会发生错误;在屏幕显示或打印字符型变量对应的数据内容时,也不正确;当我们需要频繁使用其中一小段原始数据而不在乎内存容量时,我们也需要将这些原始数据对应的字符型变量转换为整型变量才方便。故而,我们要寻求字符型变量存储的数据与整型数的相互顺利转换。在此过程中,使用虚拟变量则很方便。当原始数据均为正值时,设 f[i]是字符型变量,它存储我们要用的数据。当用它作运算时,作如下操作:

```
__DL=f[i];
__DH=0;
```

则可用虚拟变量__DX 代表 f[i]参与任何运算或操作。

当某一个整数 m,我们知道这范围不大,欲将它存入一个字符型变量时,作如下操作:

```
DX=m;
f[i]=__DL;
```

而此时,如果需要表示−128～127 之间的整数,还可以将每一个数统一加 128,使之变到 0～255 范围内。具体要用数据时,取出每一数后减去 128,即得到本来数据。

上面讨论的仅仅是一个特例,即当有大批整数,每个整数仅用 8 个 bit 就足够存储时,利用虚拟变量作转换的方法。依此类推,如果我们有大批数据,而每一个数据仅用 2 个 bit、4 个 bit 或 12 个 bit 等就足够存储内容时,为了节约存储空间,我们必须将内存各字节拆散开来存储这些内容。又由于 C 语言中"位段"功能不能定义为数组形式,故不能存储大量这类数据。利用虚拟变量作中介转换,要快得多,也方便得多。

参考文献:

*[1]*赵仁健,熊平"一种全屏幕动画的 C 语言编程"《微型机与应用》*1995.7*

*[2]*王洪《实用 *Turbo C* 教程》西北核技术研究所

*[3]*林学焦译《*Turbo C* 用户手册》中科院希望电脑公司