第二十节：隐藏中间变量为何物？

【20.1　隐藏中间变量为何物？】

“隐藏中间变量”虽然视之不见摸之不着，但是像空气一样无处不在。它有什么规律，是什么类型，数值范围是多大，研究它有什么实用价值？这就是本节要解开之谜。

前面章节提到，两个加数相加，其结果暂时先保存在一个“隐藏中间变量”里，运算结束后才把这个“隐藏中间变量”赋值给左边的“保存变量”。这里的“隐藏中间变量”到底是 unsigned int 类型还是 unsigned long 类型?为了研究它的规律，我在 keil 自带的 C51 编译环境下，专门编写了几个测试程序来观察实际运行的结果。

“保存变量”=“加数 1”+“加数 2”；

下面分别变换“保存变量”、“加数 1”、“加数 2”这三个元素的数据类型，来观察“隐藏中间变量”背后的秘密。

（1）“unsigned int”=“unsigned char”+“unsigned char”；

```
unsigned int a;
unsigned char x1=0x12;
unsigned char y1=0xfe;
a=x1+y1;
```

运算结果：a 等于 0x0110。

分析过程：两个 char 类型的数相加其运算结果暂时保存在“隐藏中间变量”，当运算结果大于两个“加数”unsigned char 本身时，并没有发生溢出现象，unsigned int 类型的“保存变量”a 最终得到了完整的结果 0x0110。

初步结论：这种情况，“隐藏中间变量”估计为 unsigned int 类型。

（2）“unsigned long”=“unsigned int”+“unsigned char”；

```
unsigned long b;
unsigned int x2=0xfffe;
unsigned char y2=0x12;
b=x2+y2;
```

运算结果：b 等于十六进制的 0x0010。

分析过程：一个 unsigned int 类型的数与一个 unsigned char 类型的数相加，当运算结果大于其中最大加数 unsigned int 类型本身时，因为左边的“保存变量”本来就是 unsigned long 类型，所以我本来以为运算结果应该是 unsigned long 类型的 0x00010010,但是实际结果出乎我的意料,最终结果是 unsigned int 类型的 0x0010，显然发生了溢出现象。

初步结论：这种情况，“隐藏中间变量”估计为 unsigned int 类型。

（3）“unsigned long”=“常量”+“常量”；

```
unsigned long c;
c=50000+50000;
```

运算结果：c 等于 100000。

分析过程：unsigned int 的最大数据范围是 65535，而两个常量相加，其结果超过了 65535 却还能完整保存下来。

初步结论：这种右边加数都是常量的情况下，“隐藏中间变量”估计等于左边的“保存变量”类型。

（4）“unsigned long”=“unsigned int”+“常量”；

```
unsigned long d;
unsigned long e;
unsigned int x3=50000;
d=x3+30000;
e=x3+50000;
```

运算结果：d 等于 14464,e 等于 100000。

分析过程:本来以为 d 应该等于 80000 的,结果却是 14464 显然发生了溢出。而同样的条件下,e 是 100000 却没有发生溢出。

个人结论：这个现象让我突然崩溃，实在研究不下去了。这是一种很怪异的现象，为什么同样的类型，因为常量的不同，一个发生了溢出，另外一个没有发生溢出？这时的“隐藏中间变量”到底是 unsigned int 类型还是 unsigned long 类型？我无法下结论。经过上述简单的测试，我发现规律是模糊的，模糊的规律就不能成为规律。如果真要按这种思路研究下去，那真是没完没了，因为还有很多情况要研究，当超过 3 个以上加数相加，同时存在 unsigned long,unsigned int,unsigned char,以及“常量”这 4 种类型时又是什么规律？在不同的 C 编译器里又会是什么现象？即使把所有情况的规律摸清楚了又能怎么样，因为那么繁杂很容易忘记导致出错。有什么解决的办法吗？

【20.2　解决办法。】

“当遇到有争议的问题时，与其参与争议越陷越深，还不如想办法及时抽身绕开争议。”根据这个指导思想，我提出一种解决思路**“为了避免出现意想不到的溢出，在实际项目中，所有参与运算的变量都预先转化为 unsigned long 变量，再参与运算。”**

当然，也可能有人会问，如果计算结果超过了 unsigned long 最大范围时怎么办？我的回答是：首先，大多数项目的计算量都比较简单，一般情况下都不会超过 unsigned long 最大范围，但是，如果真遇到有可能超过 unsigned long 最大范围的运算项目时，那么就要用另外一种 BCD 码数组的运算算法来解决，而这个方法本节暂时不介绍，等以后再讲。

继续回到刚才的话题，“为了避免出现意想不到的溢出，在实际项目中，所有参与运算的变量都预先转化为 unsigned long 变量，再参与运算。”如何把所有的运算变量都转化为 unsigned long 变量？现在介绍一下这个方法。

第一个例子：比如上述第（2）个例子，其转换方法如下：

```
unsigned long f;
unsigned int x2=0xfffe;
unsigned char y2=0x12;
unsigned  long t;  //多增加一个 long 类型的变量，用来变换类型。
unsigned  long r;  //多增加一个 long 类型的变量，用来变换类型。
t=0;    //把变量的高位和低位全部清零。
t=x2;   //把 x2 的数值先放到一个 long 类型的变量里，让”加数”跟”保存变量”类型一致。
r=0;    //把变量的高位和低位全部清零。
r=y2;   //把 y2 的数值先放到一个 long 类型的变量里，让”加数”跟”保存变量”类型一致。
f=t+r;
```

运算结果：f 等于十六进制的 0x00010010，没有发生溢出现象。

第二个例子：比如上述第（4）个例子，其转换方法如下：

```
unsigned long g;
unsigned long h;
unsigned  int x3=50000;
unsigned  long t;  //多增加一个 long 类型的变量，用来变换类型
t=0;  //把变量的高位和低位全部清零。
t=x3;   //把 x3 的数值先放到一个 long 类型的变量里，让”加数”跟”保存变量”类型一致。
g=t+30000;
h=t+50000;
```

运算结果：g 等于 80000,h 等于 100000。都没有发生溢出。

【20.3　例程练习和分析。】

现在我们编写一个程序来验证上面讲到的例子：
程序代码如下：

```
/*---C 语言学习区域的开始。-------------------------------------------*/

void main() //主函数
{
    unsigned int a;   //第（1）个例子
    unsigned char x1=0x12;
    unsigned char y1=0xfe;

    unsigned long b;  //第（2）个例子
    unsigned int x2=0xfffe;
    unsigned char y2=0x12;

    unsigned long c;  //第（3）个例子

    unsigned long d;  //第（4）个例子
    unsigned long e;
    unsigned int x3=50000;

    unsigned long f;  //第（2）个例子改进之后

    unsigned long g;  //第（4）个例子改进之后
    unsigned long h;

    unsigned  long t;  //多增加一个 long 类型的变量，用来变换类型。
    unsigned  long r;  //多增加一个 long 类型的变量，用来变换类型。


    //第（1）个例子
    a=x1+y1;
```

```
    //第（2）个例子
    b=x2+y2;

    //第（3）个例子
    c=50000+50000;

    //第（4）个例子
    d=x3+30000;
    e=x3+50000;

    //第（2）个例子改进之后
    t=0;   //把变量的高位和低位全部清零。
    t=x2;  //把 x2 的数值先放到一个 long 类型的变量里，让”加数”跟”保存变量”类型一致。
    r=0;   //把变量的高位和低位全部清零。
    r=y2;   //把 y2 的数值先放到一个 long 类型的变量里，让”加数”跟”保存变量”类型一致。
    f=t+r;

    //第（4）个例子改进之后
    t=0;    //把变量的高位和低位全部清零。
    t=x3;   //把 x3 的数值先放到一个 long 类型的变量里，让”加数”跟”保存变量”类型一致。
    g=t+30000;
    h=t+50000;



    View(a);   //把第 1 个数 a 发送到电脑端的串口助手软件上观察。
    View(b);   //把第 2 个数 b 发送到电脑端的串口助手软件上观察。
    View(c);   //把第 3 个数 c 发送到电脑端的串口助手软件上观察。
    View(d);   //把第 4 个数 d 发送到电脑端的串口助手软件上观察。
    View(e);   //把第 5 个数 e 发送到电脑端的串口助手软件上观察。
    View(f);   //把第 6 个数 f 发送到电脑端的串口助手软件上观察。
    View(g);   //把第 7 个数 g 发送到电脑端的串口助手软件上观察。
    View(h);   //把第 8 个数 h 发送到电脑端的串口助手软件上观察。

    while(1)
    {
    }
}

/*---C 语言学习区域的结束。-----------------------------------------*/
```

在电脑串口助手软件上观察到的程序执行现象如下：

```
开始...
```

```
第 1 个数
十进制:272
十六进制:110
二进制:100010000

第 2 个数
十进制:16
十六进制:10
二进制:10000

第 3 个数
十进制:100000
十六进制:186A0
二进制:11000011010100000

第 4 个数
十进制:14464
十六进制:3880
二进制:11100010000000

第 5 个数
十进制:100000
十六进制:186A0
二进制:11000011010100000

第 6 个数
十进制:65552
十六进制:10010
二进制:10000000000010000

第 7 个数
十进制:80000
十六进制:13880
二进制:10011100010000000

第 8 个数
十进制:100000
十六进制:186A0
二进制:11000011010100000
```

分析：

通过实验结果，发现在单片机上的计算结果和我们的分析是一致的。

【20.4　如何在单片机上练习本章节 C 语言程序？】

直接复制前面章节中第十一节的模板程序，练习代码时只需要更改“C 语言学习区域”的代码就可以了，其它部分的代码不要动。编译后，把程序下载进带串口的 51 学习板，通过电脑端的串口助手软件就可以观察到不同的变量数值，详细方法请看第十一节内容。