嵌入式C语言之-

栈溢出的危害

讲师：叶大鹏

助力你成为优秀的电子工程师！

程序能正确执行吗?

```
void Sum(void)
{
        volatile int32_t nums[254];
        nums[253] = 50;
        volatile int32_t x = 100, y = 200;
        volatile int32_t sum;
        sum = x + y;
}
int32_t g_test = 100;
int main(void)
{
        Sum();
        printf("g_test = %d.\n", g_test);
        return 0;
}
```

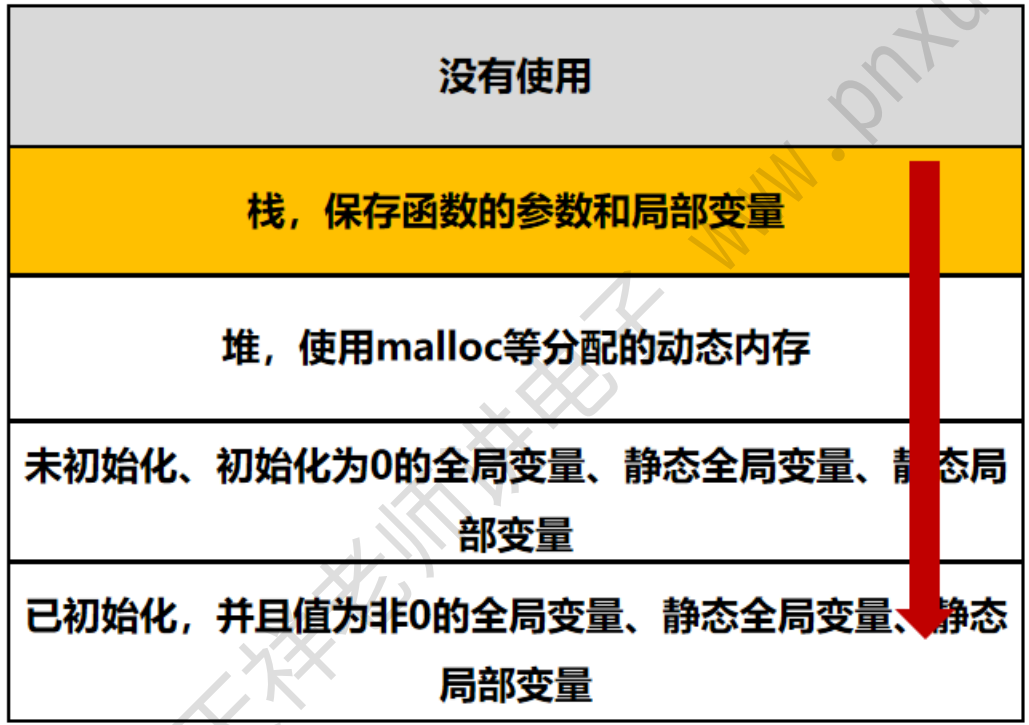
程序能正确执行吗?

```
void Sum(void)
{
        volatile int32_t nums[254];
        nums[253] = 50;
        volatile int32_t x = 100, y = 200;
        volatile int32_t sum;
        sum = x + y;
}
int32_t g_test = 100;
int main(void)
{
        Sum();
        printf("g_test = %d.\n", g_test);
        return 0;
}
```

```
g_test = 300.
```

栈溢出

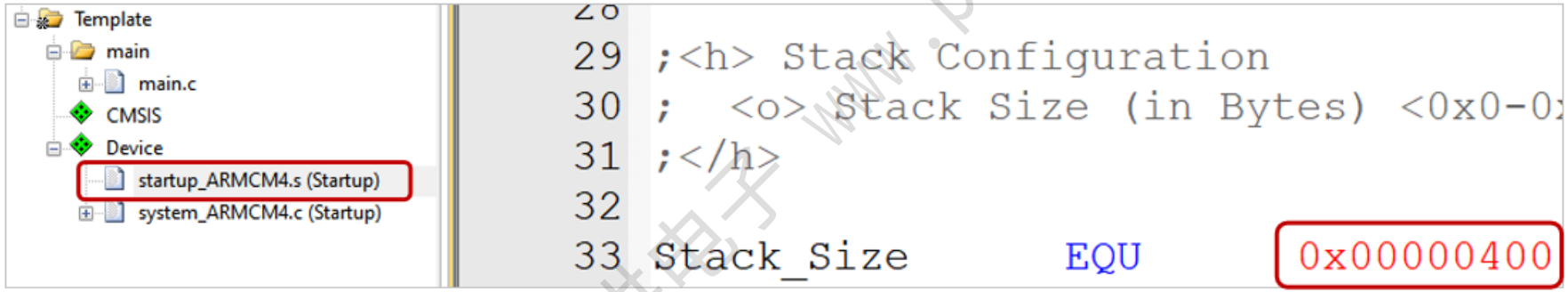
- 如果函数实际使用的栈空间超过栈最大值，就会发生栈溢出错误，导致程序异常。



PN学堂

栈空间最大值是多少?

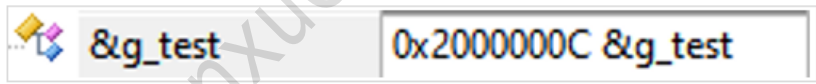
- 对于ARM 32位单片机，栈空间最大值是在startup_ARMCM4.s文件里定义的，0x00000400对应十进制是1024，代表栈空间最大值是1024个字节。

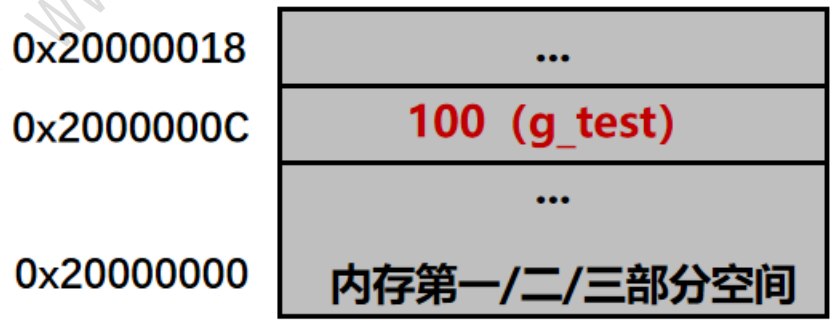


问题分析

- **全局变量g_test分配的内存在第一部分，地址为0x2000000C：**

```
int32_t g_test = 100;
int main(void)
{
    Sum();
    printf("g_test = %d.\n", g_test);
    return 0;
}
```



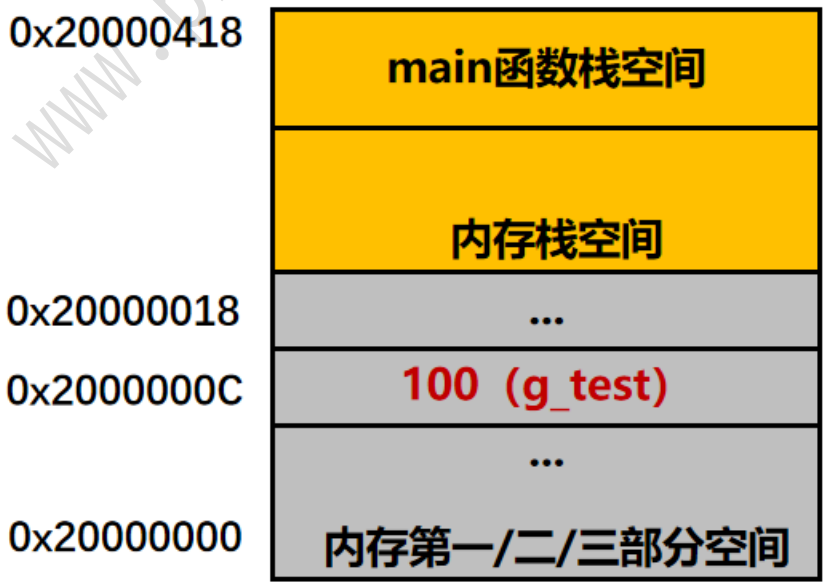


问题分析

- 进入main函数的内存分布，0x20000418这个起始地址是根据程序中一二三部分占用的内存大小加上设定的栈最大值计算得来的：

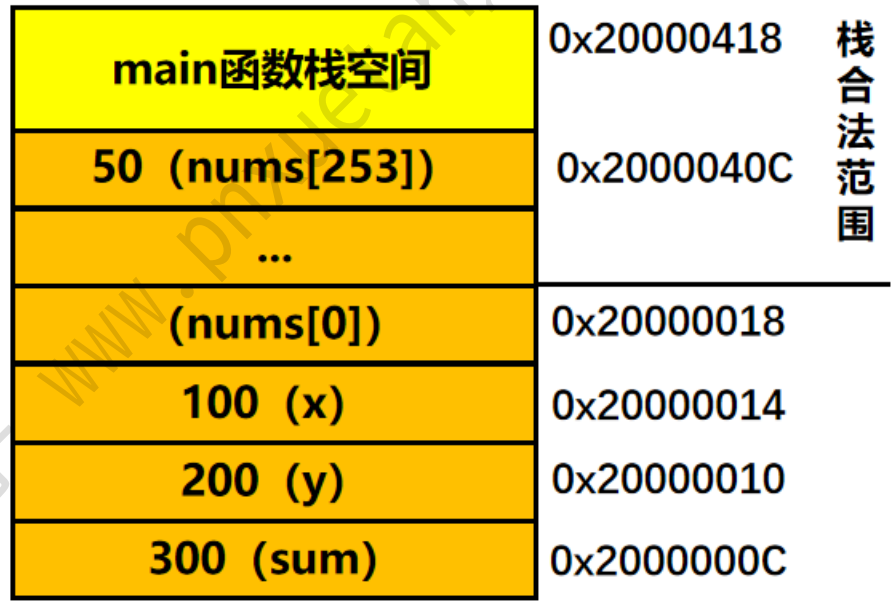
```
int32_t g_test = 100;
int main(void)
{
    Sum();
    printf("g_test = %d.\n", g_test);
    return 0;
}
```

R10	0x00000780
R11	0x00000780
R12	0x00000000
R13 (SP)	0x20000418
R14 (LR)	0x00000453
R15 (PC)	0x00000738



问题分析

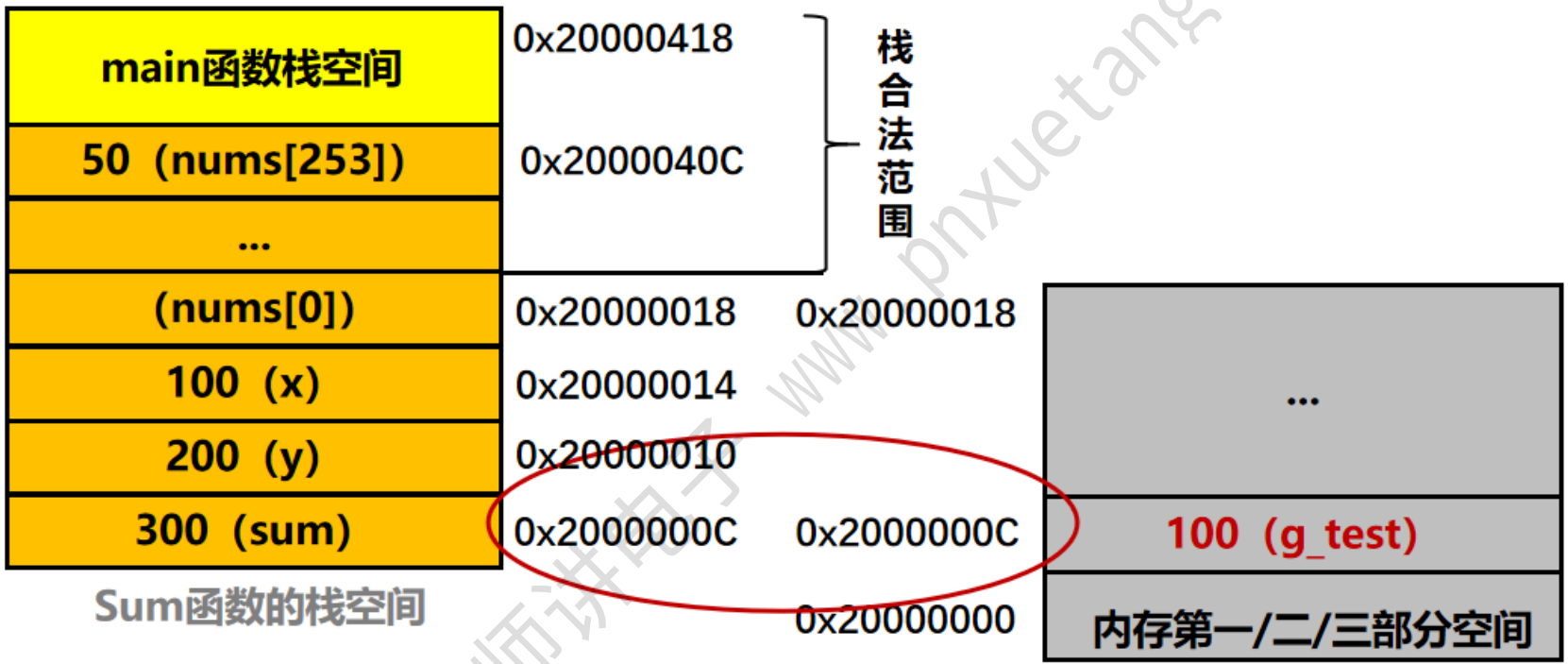
```
void Sum(void)
{
    volatile int32_t nums[254];
    nums[253] = 50;
    volatile int32_t x = 100, y = 200;
    volatile int32_t sum;
    sum = x + y;
}
```



Sum函数的栈空间

- 进入Sum函数的内存分布，在Sum函数中分配的栈空间为1028字节（0x404），再加上mian函数的，已经超过设定的0x400的栈最大值，导致sum变量使用的内存地址穿透到了内存第一部分。

问题分析



- sum变量数值300，使用的内存地址穿透到了内存第一部分，覆盖了g_test的数据100。

如何避免栈溢出?

- **减少栈空间需求:**

1. 不要定义占用内存较多的局部变量，可以将此类变量修改成指针，从堆空间分配内存;
2. 函数参数中不要传递大型数据结构，应该使用引用或指针作为函数参数，传递地址;
3. 减少函数调用层次，慎用递归函数，例如A->B->C->A环式调用。

- **增加栈最大值**

1. 在保证硬件内存空间够用的前提下，可以适当调整栈最大值。

THANK YOU!