对于基于ARM的RISC处理器，GNU C编译器提供了在C代码中内嵌汇编的功能。这种非常酷的特性提供了C代码没有的功能，比如手动优化软件关键部分的代码、使用相关的处理器指令。

这里设想了读者是熟练编写ARM汇编程序读者，因为该片文档不是ARM汇编手册。同样也不是C语言手册。

这篇文档假设使用的是GCC 4 的版本，但是对于早期的版本也有效。

GCC asm 声明让我们以一个简单的例子开始。就像C中的声明一样，下面的声明代码可能出现在你的代码中。

```
/* NOP 例子 */asm("mov r0,r0");
```

该语句的作用是将r0移动到r0中。换句话讲他并不干任何事。典型的就是NOP指令，作用就是短时的延时。

请接着阅读和学习这篇文档，因为该声明并不像你想象的和其他的C语句一样。内嵌汇编使用汇编指令就像在纯汇编程序中使用的方法一样。可以在一个asm声明中写多个汇编指令。但是为了增加程序的可读性，最好将每一个汇编指令单独放一行。

```
asm("mov r0, r0\n\t""mov r0, r0\n\t""mov r0, r0\n\t""mov r0, r0");
```

换行符和制表符的使用可以使得指令列表看起来变得美观。你第一次看起来可能有点怪异，但是当C编译器编译C语句的是候，它就是按照上面（换行和制表）生成汇编的。到目前为止，汇编指令和你写的纯汇编程序中的代码没什么区别。但是对比其它的C声明，asm的常量和寄存器的处理是不一样的。通用的内嵌汇编模版是这样的。

```
asm(code : output operand list : input operand list : clobber list);
```

汇编和C语句这间的联系是通过上面asm声明中可选的output operand list和input operand list。Clobber list后面再讲。

下面是将C语言的一个整型变量传递给汇编，逻辑左移一位后在传递给C语言的另外一个整型变量。

```
/* Rotating bits example */asm("mov %[result], %[value], ror #1" : [result] "=r" (y) : [value] "r" (x));
```

每一个asm语句被冒号（:）分成了四个部分。

- 汇编指令放在第一部分中的“”中间。