内联汇编入门

语法壳子

首先内联汇编属于c语法中的一部分，按照一定格式书写，由gcc编译器编译为正确的指令执行。

内联汇编的语法格式特别简单，就是

```
asm (核心代码);
或
asm volatile(核心代码);
```

就是一条简单的c语言表达式，然后asm就是内联汇编的关键字，volatile是可选项，告诉gcc编译器不要对我后面的代码进行优化，可选项还可以是inline等其他的，不过用不上，这里也不解释了。

当然，有时候你看见的格式会有些变化

比如

```
__asm__ (核心代码);
或
__asm__ volatile(核心代码);
```

反正不要在意你学的是哪种格式的内联汇编，意思都差不多，然后到这里语法的壳子了解了，这部分你不需要做任何深入了解，暂时就当成壳子用即可，括号里面的核心代码才是真正要掌握的。

代码怎么写

这里用举例子的方法来进行讲解，不按照一般给格式来解释的方法，

首先，比如我在c语言里面定义变量x，y，z，我想让x和y的求和结果放在z中，我起码有两种实现方式

实现方式1

```
int x,y,z;
x = 1;
y = 2;
z = x + y;
```

实现方式2

```
int x,y,z;
x = 1;
y = 2;
int add(int x,int y) {
    return x+y;
}
z = add(x,y);
```

ok，到这里，你会觉得好简单，确实相当的简单，就拿方式2来说，不就是将x和y看成输入，然后在函数中完成计算，最后返回结果给变量z吗，so easy！好的，记住这个实现方式，接下来我用内联汇编实现这个，并告诉你它是怎么做到的。

实现方式3(内联汇编)

```
int x,y,z;
x = 1;
y = 2;
asm volatile (
"movl %1, %%eax;"    // 将x加载到寄存器eax中
"addl %2, %%eax;"    // 将y加到eax中
: "=a" (z)          // 输出: 将%eax输出到变量z
: "b" (x), "c" (y)   // 输入: 将x输入到%ebx，将y输入到%ecx
);
```

格式:

```
asm volatile (
一条或多条指令
: 输出部分
: 输入部分
);
```

我们按照c语言函数的定义，先看一下输入部分:

```
"b" (x), "c" (y)
```

其中，"b"，双引号中的表示寄存器，b是%ebx的缩写，"c"是%ecx的缩写，”r“是自动分配一个寄存器，

然后(x)，括号中表示的是c语言定义的变量或者表达式，

"b" (x)实际上就是%ebx = x，就是将x的值赋给寄存器%ebx，

两个语句之间用逗号分隔。

然后再看操作的指令部分，相当于c语言的函数体:

```
"movl %1, %%eax;"    // 将x加载到寄存器eax中
"addl %2, %%eax;"    // 将y加到eax中
```

这里每条指令就是比c语言的每个表达式多个双引号，当然还有其他格式，这里就不提了，去查参考资料。

指令要做的事情，这里不用多提，和汇编要做的事情一模一样，但是一些符号稍微变化一下，

$0x10表示立即数，

%%eax表示寄存器，比正常汇编里面多了个%，

然后这里1%，2%分别表示%ebx和%ecx,这是因为从输出部分到输入部分，会按顺序把出现的寄存器编号，

比如第一个出现的寄存器是输出的%eax，然后是输入部分的%ebx和%ecx，因此它们的编号分别是0，1，2.

就可以这样简略写，当然，也可以直接使用%%ebx和%%ecx。

最后再看输出部分：

```
"=a" (z)          // 输出: 将%eax输出到变量z
```

这里做的事和输入部分完全相反，输入是把内存中的变量弄到寄存器中，输出是把寄存器的变量弄到内存里，

这里就是把%eax的内容弄到变量z里面，由于前面操作部分使得%eax里面已经是x，y的和了，因此直接把这个计算结果送给变量z即可。这里的等号是约束符号，表示只写，你就不要管什么意思了，反正每次输出部分就这么写就行了。

因此，内联汇编其实差不多也是一个函数，包括操作部分，输出部分，输入部分，每个部分用冒号分割，执行的时候先执行输入部分，把定义的内存变量送到寄存器中，然后执行操作部分，最后执行输出部分，把结果送给内存变量。

应用

```
int open(const char * filename, int flag, ...)
{
    register int res;
    va_list arg;

    va_start(arg,flag);
    __asm__("int $0x80"
      :"=a" (res)
      :"0" (__NR_open),"b" (filename),"c" (flag),
      "d" (va_arg(arg,int)));
    if (res>=0)
      return res;
    errno = -res;
    return -1;
}
```

上面是操作系统提供给用户的系统调用接口，大致实现方式就是通过内联汇编进入中断，然后实现相应的功能，最后返回。比如这里把*_NR_open中断向量号给%eax，还有其他参数赋给一些寄存器，然后使用int $0x80启动中断，调用_*NR_open号中断，完成打开文件的操作后，如果成功打开，把打开的文件fd放在%eax中返回给res，返回给open()的返回结果，如果打开失败，返回-1.

这个基本就是os里面各种用户接口的实现套路，完成哈工大oslab实现自己的系统调用，也是使用这个套路。

当然，上面的内容只是把内联汇编最核心的东西讲了下，但你感觉好像我讲的内容实在是太少了，任何一个内联汇编教程都不可能写这么一点点，肯定漏了很多内容，这点确实如此，但是这个只是让你明白内联汇编什么意思，具体的东西你还得查其他教程，不过还有最快的办法，就是遇到不会的直接问chatgpt，结合前面的基本知识，就不会有多大问题了。