循环语义与汇编语义

- `do-while` 语义和汇编语义一致，条件成立则转移

```
// 打印 0 - n
void imitate_dowhile(int n)
{
    int i = 0;
DOWHILE_BEGIN:
    printf("%d ", i);
    i++;
    if (i < n) {
        goto DOWHILE_BEGIN;
    }
    printf("\n");
    return;
}
```

- `while` 语义和汇编语义不一致，条件不成立则转移

```
// 打印 0 - n
void imitate_while(int n)
{
    int i = 0;
WHILE_BEGIN:
    if (i >= n) {
        goto WHILE_END;
    }
    printf("%d ", i);
    i++;
    goto WHILE_BEGIN;
WHILE_END:
    printf("\n");
    return;
}
```

- `for` 语义和汇编语义不一致，条件不成立则转移

```
// 打印 0 - n
void imitate_for(int n)
{
    int i = 0;
    goto FOR_CMP;

FOR_STEP:
    i++;

FOR_CMP:
```

```
    if (i >= n) {
        goto FOR_END;
    }
    printf("%d ", i);
    goto FOR_STEP;

FOR_END:
    printf("\n");
    return;
}
```

函数

定义

- 标准

```
类型标识符 函数名(参数列表)
{
    函数体
}

int foo(int a, double b)
{
    // TODO:
    return 0;
}
```

- 老式

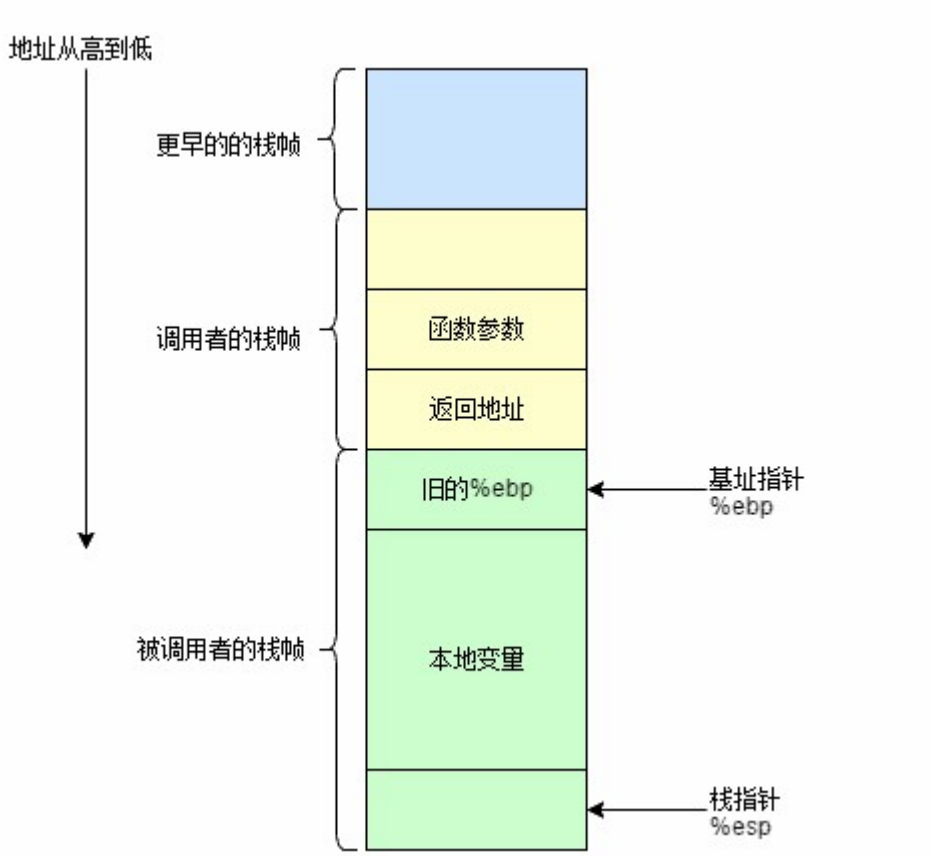
```
类型标识符 函数名(参数名列表)
参数列表
{
    函数体
}

int foo(a, b)
int a;
double b;
{
    // TODO:
    return 0;
}
```

函数调用

栈

栈（FILO），是一种后进先出的结构。在函数调用中，利用栈结构记录函数的调用关系。增长方向是由**高**到**低**

- `push`：压栈，栈顶指针减少
- `pop`：出栈，栈顶指针增加



调用过程

1. 按调用约定传递参数
    - 调用约定（调用方`caller`与被调用方`callee`）
        - **caller**与**callee**约定**传参循序**、**传参的媒介**、**清理参数的责任归属**和**返回值的传递媒介**
        - `__cdecl`
            - 传参由右向左
            - 使用栈顶传递参数
            - 由**caller**方清理参数（可支持变参函数）
        - `__stdcall`
            - 传参由右向左
            - 使用栈顶传递参数
            - 由**callee**方清理参数
        - `__fastcall`
            - 传参由右向左
            - 左边前两个基本数据类型的参数通过寄存器，其余通过栈顶
            - 由**callee**方清理参数
2. 保存返回地址
3. 保存`caller`的栈基址
    - `push ebp`

4. 设置当前栈顶为`callee`栈底（此时被调方处于空栈）
    - `mov ebp, esp`
5. 根据局部变量所需的空间抬高栈顶（为局部变量分配空间）
6. 保存寄存器（调用方寄存器中的值）
7. **Debug**选项在此时设置局部变量的初值为`0xCC`，**Release**选项此时不做操作
8. 执行函数体
9. 恢复寄存器
10. 释放局部变量空间
    - `mov esp, ebp`
11. 恢复`caller`的栈基址
    - `pop ebp`
12. 从栈顶取值作为返回地址
    - `__stdcall`与`__fastcall`此时释放参数空间
13. 流程恢复到`caller`
    - `__cdecl`释放参数空间

注：自定义函数与调用约定

一般在C环境中，调用约定默认为`__cdecl`，而且在有变参函数前使用非`__cdecl`调用约定的行为根据平台不同会有不同的处理（在VC下一般是强制转换为了`__cdecl`调用约定）
自定义函数显式指定调用约定的语法如下：

```
// 指定__cdecl调用约定
void __cdecl foo(int a, int b)
{
    // TODO:
    ...
}
```