# 汇编语言调用约定

# Assembly Calling Conventions

## x86 (IA-32)

采用微软stdcall调用约定：

Microsoft stdcall calling conventions are used:

所有参数（包括浮点数）**从右到左**压栈（右边的参数先入栈，占据高位地址；左边的参数后入栈，占据低位地址），如\_cdecl调用约定。大小不为1、2、4或8字节的参数（结构体）必须通过引用（指针）传递。

All parameters (including floating point numbers) are pushed onto the stack in **right-to-left** order (the arguments on the right are pushed onto the stack first and occupy high addresses; the arguments on the left are pushed onto the stack later and occupy low addresses), as in the \_cdecl calling conventions. Any argument that is not 1, 2, 4, or 8 bytes (structures), must be passed by reference (pointer).

**被调用者**清理栈，不支持可变数量的参数，如pascal调用约定。要传递可变数量的参数的话，需要在栈上传递指向包含它们的结构体的指针。这种方法虽然不支持可变数量的参数，但可以节省调用函数时恢复栈指针的代码所占的空间。

The **callee** is responsible for cleaning up the stack, as in the pascal calling conventions. Therefore pushing a various number of arguments onto the stack is not supported. If a various number of arguments are needed to pass to the function, they should be passed by a pointer to a structure that contains them. Although this method does not support a various number of arguments, it saves space for restoring the stack pointer after calling functions.

EAX、ECX和EDX是易失性寄存器，被调用者使用这3个寄存器前不需要先备份它们的值。其他寄存器则是非易失性的，被调用者使用它们前需要备份它们的值，并在返回前恢复它们的值。

EAX, ECX and EDX are volatile registers, which means that the callee does not need to save their values before using these 3 registers. Other registers are non-volatile so the callee must save their values before using them and restore their values before returning.

大小为1、2或4字节的整数返回值放在EAX（AX或AL）寄存器中。浮点数返回值放在x87 FPU的st(0)（MMX0）中。大小不为1、2或4字节的返回值（结构体），将其指针放在EAX中。

The integer return value of 1, 2 or 4 bytes size should be placed in EAX (AX or AL). The floating-point return value should be placed in x86 FPU stack st(0) (MMX0). The return value that is not 1, 2 or 4 bytes should be returned by their reference (pointer) in EAX.

涉及浮点数操作的函数被调用前，调用者必须确保x87 FPU浮点栈为空。函数被调用后，被调用者必须确保除st(0)外的x87 FPU浮点栈为空。

Before calling the function that may perform floating point operations, the caller must ensure that the x87 FPU stack is empty. The callee that performs floating point operations mush ensure that the x87 FPU stack is empty except st(0) before it returns.

## x64 (Intel 64 and AMD 64)

采用微软x64调用约定：

Microsoft x64 calling conventions are used:

前4个参数（从左到右）通过RCX、RDX、R8、R9（整数或指针）或XMM0、XMM1、XMM2、XMM3传递。第5个及更右面的参数从右到左依次压栈。请参考：<http://msdn.microsoft.com/zh-cn/library/zthk2dkh.aspx>

The first 4 arguments are passed (in order left to right) in registers RCX, RDX, R8 and R9 (integers or pointers) or in XMM0, XMM1, XMM2 and XMM3 (floating point numbers). The 5th and its right arguments are pushed onto the stack from right to left. Please see <http://msdn.microsoft.com/en-us/library/zthk2dkh.aspx>

尽管前4个参数通过寄存器传递。调用者要在栈上为它们留出4×8=32字节的空间，供被调用者在需要时保存4个参数寄存器的值。请参考：<http://msdn.microsoft.com/zh-cn/library/ew5tede7.aspx>

Although the first 4 arguments are passed in registers, the caller should reserve 4×8=32 bytes in the stack for the callee to save the values of the 4 registers that contain the arguments. Please see <http://msdn.microsoft.com/en-us/library/ew5tede7.aspx>

**调用者**负责清理栈，支持可变数量的参数，如\_cdecl调用约定。

The **caller** is responsible for cleaning up the stack, as in the \_cdecl calling conventions.

RAX、RCX、RDX、R8、R9、R10和R11是易失性寄存器，被调用者使用这7个寄存器前不需要先备份它们的值。其他寄存器则是非易失性的，被调用者使用它们前需要备份它们的值，并在返回前恢复它们的值。

RAX, RCX, RDX, R8, R9, R10 and R11 are volatile registers, which means that the callee does not need to save their values before using these 3 registers. Other registers are non-volatile so the callee must save their values before using them and restore their values before returning.

大小为1、2、4或8字节的整数返回值放在RAX（EAX、AX或AL）寄存器中。浮点数返回值放在XMM0中。大小不为1、2或4字节的返回值（结构体），将其指针放在RAX中。

The integer return value of 1, 2, 4 or 8 bytes size should be placed in RAX (EAX, AX or AL). The floating-point return value should be placed in XMM0. The return value that is not 1, 2, 4 or 8 bytes should be returned by their reference (pointer) in RAX.