

名称	说明
Add	将两个值相加并将结果推送到计算堆栈上。
Add.Ovf	将两个整数相加，执行溢出检查，并且将结果推送到计算堆栈上。
Add.Ovf.Un	将两个无符号整数值相加，执行溢出检查，并且将结果推送到计算堆栈上。
And	计算两个值的按位“与”并将结果推送到计算堆栈上。
Arglist	返回指向当前方法的参数列表的非托管指针。
Beq	如果两个值相等，则将控制转移到目标指令。
Beq.S	如果两个值相等，则将控制转移到目标指令（短格式）。
Bge	如果第一个值大于或等于第二个值，则将控制转移到目标指令。
Bge.S	如果第一个值大于或等于第二个值，则将控制转移到目标指令（短格式）。
Bge.Un	当比较无符号整数值或不可排序的浮点型值时，如果第一个值大于第二个值，则将控制转移到目标指令。
Bge.Un.S	当比较无符号整数值或不可排序的浮点型值时，如果第一个值大于第二个值，则将控制转移到目标指令（短格式）。
Bgt	如果第一个值大于第二个值，则将控制转移到目标指令。
Bgt.S	如果第一个值大于第二个值，则将控制转移到目标指令（短格式）。
Bgt.Un	当比较无符号整数值或不可排序的浮点型值时，如果第一个值大于第二个值，则将控制转移到目标指令。
Bgt.Un.S	当比较无符号整数值或不可排序的浮点型值时，如果第一个值大于第二个值，则将控制转移到目标指令（短格式）。
Ble	如果第一个值小于或等于第二个值，则将控制转移到目标指令。
Ble.S	如果第一个值小于或等于第二个值，则将控制转移到目标指令（短格式）。
Ble.Un	当比较无符号整数值或不可排序的浮点型值时，如果第一个值小于或等于第二个值，则将控制转移到目标指令。
Ble.Un.S	当比较无符号整数值或不可排序的浮点值时，如果第一个值小于或等于第二个值，则将控制权转移到目标指令。
Blt	如果第一个值小于第二个值，则将控制转移到目标指令。
Blt.S	如果第一个值小于第二个值，则将控制转移到目标指令（短格式）。
Blt.Un	当比较无符号整数值或不可排序的浮点型值时，如果第一个值小于第二个值，则将控制转移到目标指令。
Blt.Un.S	当比较无符号整数值或不可排序的浮点型值时，如果第一个值小于第二个值，则将控制转移到目标指令（短格式）。

Bne.Un	当两个无符号整数值或不可排序的浮点型值不相等时，将控制转移到目标指令。
Bne.Un.S	当两个无符号整数值或不可排序的浮点型值不相等时，将控制转移到目标指令（短格式）。
Box	将值类转换为对象引用（O 类型）。
Br	无条件地将控制转移到目标指令。
Br.S	无条件地将控制转移到目标指令（短格式）。
Break	向公共语言结构 (CLI) 发出信号以通知调试器已撞上了一个断点。
Brfalse	如果 value 为 false、空引用（Visual Basic 中的 Nothing）或零，则将控制转移到目标指令。
Brfalse.S	如果 value 为 false、空引用或零，则将控制转移到目标指令。
Brtrue	如果 value 为 true、非空或非零，则将控制转移到目标指令。
Brtrue.S	如果 value 为 true、非空或非零，则将控制转移到目标指令（短格式）。
Call	调用由传递的方法说明符指示的方法。
Calli	通过调用约定描述的参数调用在计算堆栈上指示的方法（作为指向入口点的指针）。
Callvirt	对对象调用后期绑定方法，并且将返回值推送到计算堆栈上。
Castclass	尝试将引用传递的对象转换为指定的类。
Ceq	比较两个值。如果这两个值相等，则将整数值 1 (int32) 推送到计算堆栈上；否则，将 0 (int32) 推送到计算堆栈上。
Cgt	比较两个值。如果第一个值大于第二个值，则将整数值 1 (int32) 推送到计算堆栈上；反之，将 0 (int32) 推送到计算堆栈上。
Cgt.Un	比较两个无符号的或不可排序的值。如果第一个值大于第二个值，则将整数值 1 (int32) 推送到计算堆栈上；否则，将 0 (int32) 推送到计算堆栈上。
Ckfinite	如果值不是有限数，则引发 ArithmeticException。
Clt	比较两个值。如果第一个值小于第二个值，则将整数值 1 (int32) 推送到计算堆栈上；反之，将 0 (int32) 推送到计算堆栈上。
Clt.Un	比较无符号的或不可排序的值 value1 和 value2。如果 value1 小于 value2，则将整数值 1 (int32) 推送到计算堆栈上；否则，将 0 (int32) 推送到计算堆栈上。
Constrained	约束要对其进行虚方法调用的类型。
Conv.I	将位于计算堆栈顶部的值转换为 native int。
Conv.I1	将位于计算堆栈顶部的值转换为 int8，然后将其扩展（填充）为 int32。
Conv.I2	将位于计算堆栈顶部的值转换为 int16，然后将其扩展（填充）为 int32。
Conv.I4	将位于计算堆栈顶部的值转换为 int32。

Conv.I8	将位于计算堆栈顶部的值转换为 int64。
Conv.Ovf.I	将位于计算堆栈顶部的有符号值转换为有符号 native int，并在溢出时引发 OverflowException。
Conv.Ovf.I.Un	将位于计算堆栈顶部的无符号值转换为有符号 native int，并在溢出时引发 OverflowException。
Conv.Ovf.I1	将位于计算堆栈顶部的有符号值转换为有符号 int8 并将其扩展为 int32，并在溢出时引发 OverflowException。
Conv.Ovf.I1.Un	将位于计算堆栈顶部的无符号值转换为有符号 int8 并将其扩展为 int32，并在溢出时引发 OverflowException。
Conv.Ovf.I2	将位于计算堆栈顶部的有符号值转换为有符号 int16 并将其扩展为 int32，并在溢出时引发 OverflowException。
Conv.Ovf.I2.Un	将位于计算堆栈顶部的无符号值转换为有符号 int16 并将其扩展为 int32，并在溢出时引发 OverflowException。
Conv.Ovf.I4	将位于计算堆栈顶部的有符号值转换为有符号 int32，并在溢出时引发 OverflowException。
Conv.Ovf.I4.Un	将位于计算堆栈顶部的无符号值转换为有符号 int32，并在溢出时引发 OverflowException。
Conv.Ovf.I8	将位于计算堆栈顶部的有符号值转换为有符号 int64，并在溢出时引发 OverflowException。
Conv.Ovf.I8.Un	将位于计算堆栈顶部的无符号值转换为有符号 int64，并在溢出时引发 OverflowException。
Conv.Ovf.U	将位于计算堆栈顶部的有符号值转换为 unsigned native int，并在溢出时引发 OverflowException。
Conv.Ovf.U.Un	将位于计算堆栈顶部的无符号值转换为 unsigned native int，并在溢出时引发 OverflowException。
Conv.Ovf.U1	将位于计算堆栈顶部的有符号值转换为 unsigned int8 并将其扩展为 int32，并在溢出时引发 OverflowException。
Conv.Ovf.U1.Un	将位于计算堆栈顶部的无符号值转换为 unsigned int8 并将其扩展为 int32，并在溢出时引发 OverflowException。
Conv.Ovf.U2	将位于计算堆栈顶部的有符号值转换为 unsigned int16 并将其扩展为 int32，并在溢出时引发 OverflowException。
Conv.Ovf.U2.Un	将位于计算堆栈顶部的无符号值转换为 unsigned int16 并将其扩展为 int32，并在溢出时引发 OverflowException。
Conv.Ovf.U4	将位于计算堆栈顶部的有符号值转换为 unsigned int32，并在溢出时引发 OverflowException。
Conv.Ovf.U4.Un	将位于计算堆栈顶部的无符号值转换为 unsigned int32，并在溢出时引发 OverflowException。
Conv.Ovf.U8	将位于计算堆栈顶部的有符号值转换为 unsigned int64，并在溢出时引发 OverflowException。
Conv.Ovf.U8.Un	将位于计算堆栈顶部的无符号值转换为 unsigned int64，并在溢出时引发 OverflowException。
Conv.R.Un	将位于计算堆栈顶部的无符号整数值转换为 float32。
Conv.R4	将位于计算堆栈顶部的值转换为 float32。
Conv.R8	将位于计算堆栈顶部的值转换为 float64。

Conv.U	将位于计算堆栈顶部的值转换为 unsigned native int，然后将其扩展为 native int。
Conv.U1	将位于计算堆栈顶部的值转换为 unsigned int8，然后将其扩展为 int32。
Conv.U2	将位于计算堆栈顶部的值转换为 unsigned int16，然后将其扩展为 int32。
Conv.U4	将位于计算堆栈顶部的值转换为 unsigned int32，然后将其扩展为 int32。
Conv.U8	将位于计算堆栈顶部的值转换为 unsigned int64，然后将其扩展为 int64。
Cpblk	将指定数目的字节从源地址复制到目标地址。
Cpobj	将位于对象（&、* 或 native int 类型）地址的值类型复制到目标对象（&、* 或 native int 类型）的地址。
Div	将两个值相除并将结果作为浮点（F 类型）或商（int32 类型）推送到计算堆栈上。
Div.Un	两个无符号整数值相除并将结果（int32）推送到计算堆栈上。
Dup	复制计算堆栈上当前最顶端的值，然后将副本推送到计算堆栈上。
Endfilter	将控制从异常的 filter 子句转移回公共语言结构 (CLI) 异常处理程序。
Endfinally	将控制从异常块的 fault 或 finally 子句转移回公共语言结构 (CLI) 异常处理程序。
Initblk	将位于特定地址的内存的指定块初始化为给定大小和初始值。
Initobj	将位于指定地址的值类型的每个字段初始化为空引用或适当的基元类型的 0。
Isinst	测试对象引用（O 类型）是否为特定类的实例。
Jmp	退出当前方法并跳至指定方法。
Ldarg	将参数（由指定索引值引用）加载到堆栈上。
Ldarg.0	将索引为 0 的参数加载到计算堆栈上。
Ldarg.1	将索引为 1 的参数加载到计算堆栈上。
Ldarg.2	将索引为 2 的参数加载到计算堆栈上。
Ldarg.3	将索引为 3 的参数加载到计算堆栈上。
Ldarg.S	将参数（由指定的短格式索引引用）加载到计算堆栈上。
Ldarga	将参数地址加载到计算堆栈上。
Ldarga.S	以短格式将参数地址加载到计算堆栈上。
Ldc.I4	将所提供的 int32 类型的值作为 int32 推送到计算堆栈上。

Ldc.I4.0	将整数值 0 作为 int32 推送到计算堆栈上。
Ldc.I4.1	将整数值 1 作为 int32 推送到计算堆栈上。
Ldc.I4.2	将整数值 2 作为 int32 推送到计算堆栈上。
Ldc.I4.3	将整数值 3 作为 int32 推送到计算堆栈上。
Ldc.I4.4	将整数值 4 作为 int32 推送到计算堆栈上。
Ldc.I4.5	将整数值 5 作为 int32 推送到计算堆栈上。
Ldc.I4.6	将整数值 6 作为 int32 推送到计算堆栈上。
Ldc.I4.7	将整数值 7 作为 int32 推送到计算堆栈上。
Ldc.I4.8	将整数值 8 作为 int32 推送到计算堆栈上。
Ldc.I4.M1	将整数值 -1 作为 int32 推送到计算堆栈上。
Ldc.I4.S	将提供的 int8 值作为 int32 推送到计算堆栈上（短格式）。
Ldc.I8	将所提供的 int64 类型的值作为 int64 推送到计算堆栈上。
Ldc.R4	将所提供的 float32 类型的值作为 F (float) 类型推送到计算堆栈上。
Ldc.R8	将所提供的 float64 类型的值作为 F (float) 类型推送到计算堆栈上。
Ldelem	按照指令中指定的类型，将指定数组索引中的元素加载到计算堆栈的顶部。
Ldelem.I	将位于指定数组索引处的 native int 类型的元素作为 native int 加载到计算堆栈的顶部。
Ldelem.I1	将位于指定数组索引处的 int8 类型的元素作为 int32 加载到计算堆栈的顶部。
Ldelem.I2	将位于指定数组索引处的 int16 类型的元素作为 int32 加载到计算堆栈的顶部。
Ldelem.I4	将位于指定数组索引处的 int32 类型的元素作为 int32 加载到计算堆栈的顶部。
Ldelem.I8	将位于指定数组索引处的 int64 类型的元素作为 int64 加载到计算堆栈的顶部。
Ldelem.R4	将位于指定数组索引处的 float32 类型的元素作为 F 类型（浮点型）加载到计算堆栈的顶部。
Ldelem.R8	将位于指定数组索引处的 float64 类型的元素作为 F 类型（浮点型）加载到计算堆栈的顶部。
Ldelem.Ref	将位于指定数组索引处的包含对象引用的元素作为 O 类型（对象引用）加载到计算堆栈的顶部。
Ldelem.U1	将位于指定数组索引处的 unsigned int8 类型的元素作为 int32 加载到计算堆栈的顶部。
Ldelem.U2	将位于指定数组索引处的 unsigned int16 类型的元素作为 int32 加载到计算堆栈的顶部。

Ldelem.U4	将位于指定数组索引处的 unsigned int32 类型的元素作为 int32 加载到计算堆栈的顶部。
Ldelema	将位于指定数组索引的数组元素的地址作为 & 类型（托管指针）加载到计算堆栈的顶部。
Ldfld	查找对象中其引用当前位于计算堆栈的字段的价值。
Ldflda	查找对象中其引用当前位于计算堆栈的字段的地址。
Ldftn	将指向实现特定方法的本机代码的非托管指针（native int 类型）推送到计算堆栈上。
Ldind.I	将 native int 类型的值作为 native int 间接加载到计算堆栈上。
Ldind.I1	将 int8 类型的值作为 int32 间接加载到计算堆栈上。
Ldind.I2	将 int16 类型的值作为 int32 间接加载到计算堆栈上。
Ldind.I4	将 int32 类型的值作为 int32 间接加载到计算堆栈上。
Ldind.I8	将 int64 类型的值作为 int64 间接加载到计算堆栈上。
Ldind.R4	将 float32 类型的值作为 F (float) 类型间接加载到计算堆栈上。
Ldind.R8	将 float64 类型的值作为 F (float) 类型间接加载到计算堆栈上。
Ldind.Ref	将对象引用作为 O（对象引用）类型间接加载到计算堆栈上。
Ldind.U1	将 unsigned int8 类型的值作为 int32 间接加载到计算堆栈上。
Ldind.U2	将 unsigned int16 类型的值作为 int32 间接加载到计算堆栈上。
Ldind.U4	将 unsigned int32 类型的值作为 int32 间接加载到计算堆栈上。
Ldlen	将从零开始的、一维数组的元素的数目推送到计算堆栈上。
Ldloc	将指定索引处的局部变量加载到计算堆栈上。
Ldloc.0	将索引 0 处的局部变量加载到计算堆栈上。
Ldloc.1	将索引 1 处的局部变量加载到计算堆栈上。
Ldloc.2	将索引 2 处的局部变量加载到计算堆栈上。
Ldloc.3	将索引 3 处的局部变量加载到计算堆栈上。
Ldloc.S	将特定索引处的局部变量加载到计算堆栈上（短格式）。
Ldloca	将位于特定索引处的局部变量的地址加载到计算堆栈上。

Ldloc.S	将位于特定索引处的局部变量的地址加载到计算堆栈上（短格式）。
Ldnull	将空引用（O 类型）推送到计算堆栈上。
Ldobj	将地址指向的值类型对象复制到计算堆栈的顶部。
Ldsfld	将静态字段的值推送到计算堆栈上。
Ldsflda	将静态字段的地址推送到计算堆栈上。
Ldstr	推送对元数据中存储的字符串的新对象引用。
Ldtoken	将元数据标记转换为其运行时表示形式，并将其推送到计算堆栈上。
Ldvirtftn	将指向实现与指定对象关联的特定虚方法的本机代码的非托管指针（native int 类型）推送到计算堆栈上。
Leave	退出受保护的代码区域，无条件将控制转移到特定目标指令。
Leave.S	退出受保护的代码区域，无条件将控制转移到目标指令（缩写形式）。
Localloc	从本地动态内存池分配特定数目的字节并将第一个分配的字节的地址（瞬态指针，* 类型）推送到计算堆栈
Mkrefany	将对特定类型实例的类型化引用推送到计算堆栈上。
Mul	将两个值相乘并将结果推送到计算堆栈上。
Mul.Ovf	将两个整数值相乘，执行溢出检查，并将结果推送到计算堆栈上。
Mul.Ovf.Un	将两个无符号整数值相乘，执行溢出检查，并将结果推送到计算堆栈上。
Neg	对一个值执行求反并将结果推送到计算堆栈上。
Newarr	将对新的从零开始的一维数组（其元素属于特定类型）的对象引用推送到计算堆栈上。
Newobj	创建一个值类型的新对象或新实例，并将对象引用（O 类型）推送到计算堆栈上。
Nop	如果修补操作码，则填充空间。尽管可能消耗处理周期，但未执行任何有意义的操作。
Not	计算堆栈顶部整数值的按位求补并将结果作为相同的类型推送到计算堆栈上。
Or	计算位于堆栈顶部的两个整数值的按位求补并将结果推送到计算堆栈上。
Pop	移除当前位于计算堆栈顶部的值。
Prefix1	基础结构。此指令为保留指令。
Prefix2	基础结构。此指令为保留指令。
Prefix3	基础结构。此指令为保留指令。

Prefix4	基础结构。此指令为保留指令。
Prefix5	基础结构。此指令为保留指令。
Prefix6	基础结构。此指令为保留指令。
Prefix7	基础结构。此指令为保留指令。
Prefixref	基础结构。此指令为保留指令。
Readonly	指定后面的数组地址操作在运行时不执行类型检查，并且返回可变性受限的托管指针。
Refanytype	检索嵌入在类型化引用内的类型标记。
Refanyval	检索嵌入在类型化引用内的地址（& 类型）。
Rem	将两个值相除并将余数推送到计算堆栈上。
Rem.Un	将两个无符号值相除并将余数推送到计算堆栈上。
Ret	从当前方法返回，并将返回值（如果存在）从调用方的计算堆栈推送到被调用方的计算堆栈上。
Rethrow	再次引发当前异常。
Shl	将整数值左移（用零填充）指定的位数，并将结果推送到计算堆栈上。
Shr	将整数值右移（保留符号）指定的位数，并将结果推送到计算堆栈上。
Shr.Un	将无符号整数值右移（用零填充）指定的位数，并将结果推送到计算堆栈上。
Sizeof	将提供的值类型的大小（以字节为单位）推送到计算堆栈上。
Starg	将位于计算堆栈顶部的值存储到位于指定索引的参数槽中。
Starg.S	将位于计算堆栈顶部的值存储在参数槽中的指定索引处（短格式）。
Stelem	用计算堆栈中的值替换给定索引处的数组元素，其类型在指令中指定。
Stelem.I	用计算堆栈上的 native int 值替换给定索引处的数组元素。
Stelem.I1	用计算堆栈上的 int8 值替换给定索引处的数组元素。
Stelem.I2	用计算堆栈上的 int16 值替换给定索引处的数组元素。
Stelem.I4	用计算堆栈上的 int32 值替换给定索引处的数组元素。
Stelem.I8	用计算堆栈上的 int64 值替换给定索引处的数组元素。
Stelem.R4	用计算堆栈上的 float32 值替换给定索引处的数组元素。



Stelem.R8	用计算堆栈上的 float64 值替换给定索引处的数组元素。
Stelem.Ref	用计算堆栈上的对象 ref 值（O 类型）替换给定索引处的数组元素。
Stfld	用新值替换在对象引用或指针的字段中存储的值。
Stind.I	在所提供的地址存储 native int 类型的值。
Stind.I1	在所提供的地址存储 int8 类型的值。
Stind.I2	在所提供的地址存储 int16 类型的值。
Stind.I4	在所提供的地址存储 int32 类型的值。
Stind.I8	在所提供的地址存储 int64 类型的值。
Stind.R4	在所提供的地址存储 float32 类型的值。
Stind.R8	在所提供的地址存储 float64 类型的值。
Stind.Ref	存储所提供地址处的对象引用值。
Stloc	从计算堆栈的顶部弹出当前值并将其存储到指定索引处的局部变量列表中。
Stloc.0	从计算堆栈的顶部弹出当前值并将其存储到索引 0 处的局部变量列表中。
Stloc.1	从计算堆栈的顶部弹出当前值并将其存储到索引 1 处的局部变量列表中。
Stloc.2	从计算堆栈的顶部弹出当前值并将其存储到索引 2 处的局部变量列表中。
Stloc.3	从计算堆栈的顶部弹出当前值并将其存储到索引 3 处的局部变量列表中。
Stloc.S	从计算堆栈的顶部弹出当前值并将其存储在局部变量列表中的 index 处（短格式）。
Stobj	将指定类型的值从计算堆栈复制到所提供的内存地址中。
Stsfld	用来自计算堆栈的值替换静态字段的值。
Sub	从其他值中减去一个值并将结果推送到计算堆栈上。
Sub.Ovf	从另一值中减去一个整数值，执行溢出检查，并且将结果推送到计算堆栈上。
Sub.Ovf.Un	从另一值中减去一个无符号整数值，执行溢出检查，并且将结果推送到计算堆栈上。
Switch	实现跳转表。
Tailcall	执行后缀的方法调用指令，以便在执行实际调用指令前移除当前方法的堆栈帧。

Throw	引发当前位于计算堆栈上的异常对象。
Unaligned	指示当前位于计算堆栈上的地址可能没有与紧接的 ldind、stind、ldfld、stfld、ldobj、stobj、initblk 或
Unbox	将值类型的已装箱的表示形式转换为其未装箱的形式。
Unbox.Any	将指令中指定类型的已装箱的表示形式转换成未装箱形式。
Volatile	指定当前位于计算堆栈顶部的地址可以是易失的，并且读取该位置的结果不能被缓存，或者对该地址的多个
Xor	计算位于计算堆栈顶部的两个值的按位异或，并且将结果推送到计算堆栈上。
<div><div></div></div>	