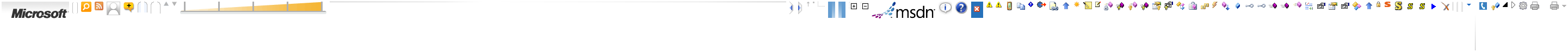
**Parameter Passing**

**Visual Studio 2010**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/zthk2dkh(v=vs.90).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/zthk2dkh(v=vs.80).aspx)

1 out of 2 rated this helpful [Rate this topic](http://msdn.microsoft.com/en-us/library/zthk2dkh.aspx#feedback)

The first four integer arguments will be passed in registers. Integer values will be passed (in order left to right) in RCX, RDX, R8, and R9. Arguments four and higher will passed onto the stack. All arguments are right justified in registers. This is done so the callee can ignore the upper bits of the register if need be and can access only the portion of the register necessary.

Floating-point and double-precision arguments are passed in XMM0 – XMM3 (up to 4) with the integer slot (RCX, RDX, R8, and R9) that would normally be used for that cardinal slot being ignored (see example) and vice versa.

[\_\_m128](http://msdn.microsoft.com/en-us/library/ayeb3ayc.aspx) types, arrays and strings are never passed by immediate value but rather a pointer will be passed to memory allocated by the caller. Structs/unions of size 8, 16, 32, or 64 bits and \_\_m64 will be passed as if they were integers of the same size. Structs/unions other than these sizes will be passed as a pointer to memory allocated by the caller. For these aggregate types passed as a pointer (including \_\_m128), the caller-allocated temporary memory will be 16-byte aligned.

Intrinsic functions that do not allocate stack space and do not call other functions can use other volatile registers to pass additional register arguments because there is a tight binding between the compiler and the intrinsic function implementation. This is a further opportunity for improving performance.

The callee has the responsibility of dumping the register parameters into their shadow space if needed.

The following table summarizes how parameters are passed:

|  |  |
| --- | --- |
| Parameter type | How passed |
| Floating point | First 4 parameters – XMM0 through XMM3. Others passed on stack. |
| Integer | First 4 parameters – RCX, RDX, R8, R9. Others passed on stack. |
| Aggregates (8, 16, 32, or 64 bits) and \_\_m64 | First 4 parameters – RCX, RDX, R8, R9. Others passed on stack. |
| Aggregates (other) | By pointer. First 4 parameters passed as pointers in RCX, RDX, R8, and R9 |
| \_\_m128 | By pointer. First 4 parameters passed as pointers in RCX, RDX, R8, and R9 |

[[http://i.msdn.microsoft.com/Hash/030c41d9079671d09a62d8e2c1db6973.gif](javascript:void(0))**Example of argument passing 1 – all integers**](javascript:void(0))

[Copy](javascript:CodeSnippet_CopyCode('CodeSnippetContainerCode_1e41303c-d60b-43d6-8347-c1388566cbd2');" \o "Copy to clipboard.)

func1(int a, int b, int c, int d, int e);

// a in RCX, b in RDX, c in R8, d in R9, e pushed on stack

[[http://i.msdn.microsoft.com/Hash/030c41d9079671d09a62d8e2c1db6973.gif](javascript:void(0))**Example of argument passing 2 – all floats**](javascript:void(0))

[Copy](javascript:CodeSnippet_CopyCode('CodeSnippetContainerCode_1c993376-9483-454f-b5e1-3431eb7feac9');)

func2(float a, double b, float c, double d, float e);

// a in XMM0, b in XMM1, c in XMM2, d in XMM3, e pushed on stack

[[http://i.msdn.microsoft.com/Hash/030c41d9079671d09a62d8e2c1db6973.gif](javascript:void(0))**Example of argument passing 3 – mixed ints and floats**](javascript:void(0))

[Copy](javascript:CodeSnippet_CopyCode('CodeSnippetContainerCode_e6962e9f-e8c4-455c-870d-b252aee79755');)

func3(int a, double b, int c, float d);

// a in RCX, b in XMM1, c in R8, d in XMM3

[[http://i.msdn.microsoft.com/Hash/030c41d9079671d09a62d8e2c1db6973.gif](javascript:void(0))**Example of argument passing 4 –\_\_m64, \_\_m128, and aggregates**](javascript:void(0))

[Copy](javascript:CodeSnippet_CopyCode('CodeSnippetContainerCode_3a775df1-9277-4088-8798-eb1d2d565dea');)

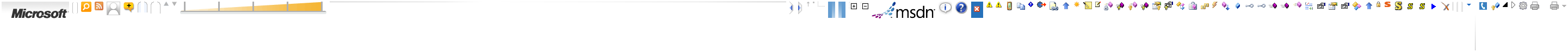
func4(\_\_m64 a, \_m128 b, struct c, float d);

// a in RCX, ptr to b in RDX, ptr to c in R8, d in XMM3

# Return Values (C++)

**Visual Studio 2010**

[Other Versions](javascript:;)



* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/7572ztz4(v=vs.90).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/7572ztz4(v=vs.80).aspx)

This topic has not yet been rated [Rate this topic](http://msdn.microsoft.com/en-us/library/7572ztz4.aspx#feedback)

Return values that can fit into 64-bits are returned through RAX (including \_\_m64 types), except for [\_\_m128](http://msdn.microsoft.com/en-us/library/ayeb3ayc.aspx), [\_\_m128i](http://msdn.microsoft.com/en-us/library/26232t5c.aspx), [\_\_m128d](http://msdn.microsoft.com/en-us/library/25537780.aspx), floats, and doubles, which are returned in XMM0. If the return value does not fit within 64 bits, then the caller assumes the responsibility of allocating and passing a pointer for the return value as the first argument. Subsequent arguments are then shifted one argument to the right. That same pointer must be returned by the callee in RAX. User defined types to be returned must be 1, 2, 4, 8, 16, 32, or 64 bits in length.