

# The *COMPLETE* HLSL Reference

All the information developers need from shader assembly instruction to a complete HLSL intrinsic function overview covering up to shader model 3.0. Presented in a quick and easy to access format!

Sebastien St-Laurent

The logo for Paradoxal Press is located at the bottom of the page. It features the word "Paradoxal" in a blue serif font and "Press" in a red serif font below it. A circular graphic element with three white dots is positioned between the two words. The entire logo is set against a light blue rounded rectangular background.

Paradoxal  
Press

# Vertex and Pixel Shader Assembly Language

Although the development of assembly level shaders is rare today because of the HLSL language, a proper understanding of the underlying assembly instruction sets is core to the development and debugging of efficient shaders.

Both the vertex and pixel shader assembly languages are composed of setup and assembly instructions. Throughout the following pages you will find tables containing a summary of both types of instructions along with a description of their function, parameter information, performance indications and support of various shader versions.

Before starting, I have to make a note in regards to some of the flow control instructions. For example, the *if\_comp* instruction defines flow control based on a comparison where the *\_comp* component will be replaced by a comparison operation. The table below summarizes all the operations which can be used within flow control:

## Flow Control Comparison Modes

Mode	Description
<i>_gt</i>	Greater
<i>_lt</i>	Lesser
<i>_ge</i>	Greater or equal
<i>_le</i>	Lesser or equal
<i>_eq</i>	Equal
<i>_ne</i>	Not equal

In addition, the following tables contain some indication of the support of each instruction on every vertex and pixel shader version using a color coded system. The table below summarizes the coding:

Color	Description
Green	Requires less than 4 instructions slots.
Blue	Requires between 4 and 7 instruction slots.
Yellow	Required 8 or more instruction slots.
Red	Unsupported.

I must also make a final note in regards to the performance values indicated over the next pages. The quoted values are based on the DirectX specifications and indicate the worst case performance that must be ensured by the rendering hardware. This means that certain hardware implementations may execute certain instructions faster than prescribed by the specifications.

# Vertex Shader Assembly

Name	Description	1.1	2.0	2.x	3.0
<i>dcl_usage</i>	Declare input registers.				
<i>def</i>	Define constants.				
<i>defb</i>	Define boolean constants.				
<i>defi</i>	Define integer constants.				
<i>label</i>	Define flow control labels.				
<i>vs</i>	Retrieve the shader version.				

Name	1.1	2.0	2.x	3.0
	Description			
	Additional Parameter Info			
<i>abs dst, src</i>		1	1	1
	Absolute value of the <i>src</i> .			
<i>add dst, src0, src1</i>	1	1	1	1
	Add <i>src0</i> and <i>src1</i> together.			
<i>break</i>			1	1
	Break out of a <i>loop</i> or <i>rep</i> block.			
<i>break_comp src0, src1</i>			3	3
	Conditionally break out of a <i>loop</i> or <i>rep</i> block based on a scalar conditional.			
	_comp = comparison mode. src0, src1 = source registers.			
<i>breakp [!]p0.[x y z w]</i>			3	3
	Break out of a <i>loop</i> or <i>rep</i> block based on a predicate.			
	[!] = optional <i>NOT</i> operator. p0 = is a predicate register. {x y z w} = required replicate swizzle.			
<i>call l#</i>		2	2	2
	Call a subroutine defined by a label.			
	l# = label to subroutine.			
<i>callnz_bool l#, [!]b#</i>		2	3	3
	Call a subroutine if register is nonzero.			
	l# = label to call. [!] = optional <i>NOT</i> operator. b# = boolean register.			
<i>callnz_pred l#, [!]p0.[x y z w]</i>			3	3
	Call a subroutine based on a predicate.			
	l# = label to call. [!] = optional <i>NOT</i> operator. p0 = predicate register. {x y z w} = required replicate swizzle.			
<i>crs dst, src0, src1</i>		2	2	2
	Cross product of <i>src0</i> and <i>src1</i> .			
<i>dp3 dst, src0, src1</i>	1	1	1	1
	Three-component dot product.			
<i>dp4 dst, src0, src1</i>	1	1	1	1
	Four-component dot product.			