Appendix A

Low-Level Shading Languages Reference

Paul Baker

Instructions

Table A.1 lists all of the instructions that are common to both `ARB_vertex_program` and `ARB_fragment_program`. Table A.2 lists the instructions that are only available in `ARB_vertex_program`, and Table A.3 lists the instructions that are only available in `ARB_fragment_program`.

Table A.1 Shared Instructions

Instruction	Meaning	Input	Output	Description
`ABS`	Absolute Value	(x, y, z, w)	(|x|, |y|, |z|, |w|)	Calculates the absolute value of each component of the source vector.
`ADD`	Addition	(x1, y1, z1, w1), (x2, y2, z2, w2)	(x1 + x2, y1 + y2, z1 + z2, w1 + w2)	Adds the two source vectors component-wise.
`DP3`	3-component dot product	(x1, y1, z1, w1), (x2, y2, z2, w2)	(d, d, d, d) where + y1 * y2 + z1 * z2 d = x1 * x2	Calculates a 3-component scalar (dot) product of the source vectors and writes the result to all components of the destination vector.
`DP4`	4-component dot product	(x1, y1, z1, w1), (x2, y2, z2, w2)	(d, d, d, d) where d = x1 * x2 + y1 * y2 + z1 * z2 + w1 * w2	Calculates a 4-component scalar (dot) product of the source vectors and writes the result to all components of the destination vector.

continued

DPH	Homogeneous dot product	(x1, y1, z1, w1), (x2, y2, z2, w2)	(d, d, d, d) where d = x1 * x2 + y1 * y2 + z1 * z2 + w2	Calculates a 4-component scalar (dot) product of the source vectors, with the exception that w1 is forced to 1.0. Writes the result to all components of the destination vector.
DST	Distance vector	(x1, y1, z1, w1), (x2, y2, z2, w2)	(1.0, y1 * y2, z1, w2)	When used with input vectors of the form:| (x1, y1, z1, w1) = (N/A, d^2, d^2, N/A) (x2, y2, z2, w2) = (N/A, 1/d, N/A, 1/d) Calculates: (1.0, d, d^2, 1/d)
EX2	Exponential base 2	s	(2^s, 2^s, 2^s, 2^s)	Calculates 2 raised to the power of the scalar operand and writes the result to all four components of the destination vector.
FLR	Floor	(x, y, z, w)	(floor(x), floor(y), floor(z), floor(w))	Calculates the floor of each component of the source vector. The floor of a value is the largest integer less than or equal to the value.
FRC	Fraction	(x, y, z, w)	(fraction(x), fraction (y), fraction (z), fraction (w))	Calculates the fractional part of each component of the source vector.
LG2	Logarithm base 2	s	(log2(s), log2(s), log2(s), log2(s))	Calculates the base 2 logarithm of the scalar operand and writes the result to all four components of the destination vector.
LIT	Calculate lighting coefficients	(d, s, N/A, x)	(1.0, D, S, 1.0)	d is a diffuse dot product, s is a specular dot product, x is the specular exponent, D is the value to be multiplied by the diffuse color, and S is the value to be multiplied by the specular color.
MAD	Multiply and accumulate	(x1, y1, z1, w1), (x2, y2, z2, w2), (x3, y3, z3, w3)	(x1 * x2 + x3, y1 * y2 + y3, z1 * z2 + z3, w1 * w2 + w3)	Multiplies the first two source vectors and adds the third source vector, component-wise.
MAX	Maximum	(x1, y1, z1, w1), (x2, y2, z2, w2)	(max(x1, x2), max(y1, y2), max(z1, z2), max(w1, w2))	Calculates the component-wise maximum of the two source vectors.

continued

MIN	Minimum	(x1, y1, z1, w1), (x2, y2, z2, w2)	(min(x1, x2), min(y1, y2), min(z1, z2), min(w1, w2))	Calculates the component-wise minimum of the two source vectors.
MOV	Move	(x1, y1, z1, w1)	(x1, y1, z1, w1)	Copies the source vector into the destination vector.
MUL	Multiplication	(x1, y1, z1, w1), (x2, y2, z2, w2)	(x1 * x2, y1 * y2, z1 * z2, w1 * w2)	Multiplies the two source vectors component-wise.
POW	Power	s, p	(s^p, s^p, s^p, s^p)	Raises the first scalar operand to the power of the second scalar operand and writes the result to all four components of the destination vector.
RCP	Reciprocal	s	(1/s, 1/s, 1/s, 1/s)	Calculates the reciprocal of the scalar operand and writes the result to all four components of the destination vector.
RSQ	Reciprocal square root	s	(1/sqrt(s), 1/sqrt(s), 1/sqrt(s), 1/sqrt(s))	Calculates the reciprocal of the scalar operand and writes the result to all four components of the destination vector.
SGE	Set On Greater Than Or Equal	(x1, y1, z1, w1), (x2, y2, z2, w2)	(x1 >= x2 ? 1.0 : 0.0, y1 >= y2 ? 1.0 : 0.0, z1 >= z2 ? 1.0 : 0.0, w1	Compares the two source vectors component-wise, setting the corresponding component of the destination vector if the first source is greater than or equal to the second.
SLT	Set On Less Than	(x1, y1, z1, w1), (x2, y2, z2, w2)	(x1 < x2 ? 1.0 : 0.0, y1 < y2 ? 1.0 : 0.0, z1 < z2 ? 1.0 : 0.0, w1 < w2 ? 1.0 : 0.0)	Compares the two source vectors component-wise, setting the corresponding component of the destination vector if the first source is less than the second.
SUB	Subtraction	(x1, y1, z1, w1), (x2, y2, z2, w2)	(x1 –– x2, y1 –– y2, z1 –– z2, w1 –– w2)	Subtracts the two source vectors component-wise.

continued

SWZ	Extended Swizzle	(x1, y1, z1, w1), c1, c2, c3, c4	(c1, c2, c3, c4)	Reorders the components of a source vector, optionally negating selected components or replacing them with a constant. The parameters c1–c4 can be x, −x, y, −y, z, −z, w, −w, 0, 1 or −1. The corresponding component of the source vector, or the given constant, is used as the component of the destination vector.
XPD	Cross Product	(x1, y1, z1, N/A), (x2, y2, z2, N/A)	(y1 * z2 − z1 * y2, z1 * x2 − x1 * z2, x1 * y2 − y1 * x2, UNDEF)	Calculates a vector (cross) product using the first three components of the source vectors and writes the result to the first three components of the destination vector.

Table A.2 Vertex Program-Only Instructions

Instruction	Meaning	Input	Output	Description
ARL	Address Register Load	s	floor(s)	Loads an address register with a scalar.
EXP	Approximate Exponential base 2	s	(2 ^ (floor(s)), s − floor(s), 2 ^ s (approx), 1.0)	Approximates 2 raised to the power of the scalar operand and writes the result to the z component of the result. The x and y components of the result can be used to generate a more accurate approximation by calculating: result.x * f(result.y) where f takes a parameter x in [0.0, 1.0) and calculates 2^x.
LOG	Approximate Logarithm base 2	s	(floor(log2(|s|)), |s| / 2 ^ floor (log2(|s|)), log2(|s|) (approx), 1.0)	Approximates the base-2 logarithm of the scalar operand and writes the result to the z component of the result. The x and y components of the result can be used to generate a more accurate approximation by calculating: result.x * f(result.y) where f takes a parameter x in [1.0, 2.0) and calculates 2^x.

Table A.3 Fragment Program-Only Instructions

Instruction	Meaning	Input	Output	Description
CMP	Compare	(x1, y1, z1, w1), (x2, y2, z2, w2), (x3, y3, z3, w3)	(x1 < 0.0 ? x2 x3, y1 < 0.0 ? y2 : y3, z1 < 0.0 ? z2 : z3, w1 < 0.0 ? w2 : w3)	Compares each component of the first source vector with zero, and selects the corresponding component of the second or third source vector depending on the result.
COS	Cosine	s	(cos(s), cos(s), cos(s), cos(s))	Calculates the cosine of the scalar operand and writes the result to all four components of the destination vector.
KIL	Kill fragment	(x1, y1, z1, w1)	None	Prevents any further processing of this fragment if any component of the source vector is less than zero.
LRP	Linear interpolation	(x1, y1, z1, w1), (x2, y2, z2, w2), (x3, y3, z3, w3)	(x1 * x2 + (1.0 − x1) * x3, y1 * y2 + (1.0 − y1) * y3, z1 * z2 + (1.0 − z1) * z3, w1 * w2 + (1.0 v w1) * w3)	Uses the first source vector as a blend factor to linearly interpolate between the second and third source vectors, component-wise.
SCS	Sine/Cosine	s	(cos(s), sin(s), UNDEF, UNDEF)	Calculates the cosine and sine of the scalar operand and writes the results to the x and y components of the destination vector respectively. The scalar operand must lie in the range [$-\pi$, π].
SIN	Sine	s	(sin(s), sin(s), sin(s), sin(s))	Calculates the sine of the scalar operand and writes the result to all four components of the destination vector.
TEX	Texture Sample	(s, t, r, q), texture[n],	TARGET	Texture sample at (s, t, r). Samples a texture using the coordinates (s, t, r). TARGET may be 1D, 2D, 3D, RECT or CUBE.
TXB	Texture Sample with Bias	(s, t, r, q), texture[n],	TARGET	Texture sample at (s, t, r), with level-of-detail bias q. Samples a texture using the coordinates (s, t, r), using the q coordinate as a level-of-detail bias.
TXP	Texture Sample with Projection	(s, t, r, q), texture[n],	TARGET	Texture sample at (s/q, t/q, r/q). Samples a texture using the projective coordinates (s/q, t/q, r/q).

Parameter Vectors

The OpenGL state variables listed in Table A.4 are available to vertex programs and fragment programs as parameter vectors. Unless otherwise indicated, the variables are available to both types of program.

Table A.4 Parameter Vectors

Vector	Description	Values
state.material.ambient	Front ambient material color	(r, g, b, a)
state.material.diffuse	Front diffuse material color	(r, g, b, a)
state.material.specular	Front specular material color	(r, g, b, a)
state.material.emission	Front emissive material color	(r, g, b, a)
state.material.shininess	Front material shininess	(s, 0, 0, 1)
state.material.front.ambient	Front ambient material color	(r, g, b, a)
state.material.front.diffuse	Front diffuse material color	(r, g, b, a)
state.material.front.specular	Front specular material color	(r, g, b, a)
state.material.front.emission	Front emissive material color	(r, g, b, a)
state.material.front.shininess	Front material shininess	(s, 0, 0, 1)
state.material.back.ambient	Back ambient material color	(r, g, b, a)
state.material.back.diffuse	Back diffuse material color	(r, g, b, a)
state.material.back.specular	Back specular material color	(r, g, b, a)
state.material.back.emission	Back emissive material color	(r, g, b, a)
state.material.back.shininess	Back material shininess	(s, 0, 0, 1)
state.light[n].ambient	Light n ambient color	(r, g, b, a)
state.light[n].diffuse	Light n diffuse color	(r, g, b, a)
state.light[n].specular	Light n specular color	(r, g, b, a)
state.light[n].position	Light n position	(x, y, z, w)
state.light[n].attenuation	Light n attenuation coefficients and spot light exponent	(a, b, c, e)
state.light[n].spot.direction	Light n spot direction and cutoff angle cosine	(x, y, z, c)
state.light[n].half	Light n infinite light/infinite viewer half angle	(x, y, z, 1)
state.lightmodel.ambient	Light model ambient color	(r, g, b, a)
state.lightmodel.scenecolor	Light model front scene color	(r, g, b, a)
state.lightmodel.front.scenecolor	Light model front scene color	(r, g, b, a)

continued

state.lightmodel.back.scenecolor	Light model back scene color	(r, g, b, a)
state.lightprod[n].ambient	Light n ambient color * front material ambient color	(r, g, b, a)
state.lightprod[n].diffuse	Light n diffuse color * front material diffuse color	(r, g, b, a)
state.lightprod[n].specular	Light n specular color * front material specular color	(r, g, b, a)
state.lightprod[n].front.ambient	Light n ambient color * front material ambient color	(r, g, b, a)
state.lightprod[n].front.diffuse	Light n diffuse color * front material diffuse color	(r, g, b, a)
state.lightprod[n].front.specular	Light n specular color * front material specular color	(r, g, b, a)
state.lightprod[n].back.ambient	Light n ambient color * back material ambient color	(r, g, b, a)
state.lightprod[n].back diffuse	Light n diffuse color * back material diffuse color	(r, g, b, a)
state.lightprod[n].back specular	Light n specular color * back material specular color	(r, g, b, a)
state.texgen[n].eye.s	TexGen eye linear plane coefficients, unit n, s coordinate	(a, b, c, d) *
state.texgen[n].eye.t	TexGen eye linear plane coefficients, unit n, t coordinate	(a, b, c, d) *
state.texgen[n].eye.r	TexGen eye linear plane coefficients, unit n, r coordinate	(a, b, c, d) *
state.texgen[n].eye.q	TexGen eye linear plane coefficients, unit n, q coordinate	(a, b, c, d) *
state.texgen[n].object.s	TexGen object linear plane coefficients, unit n, s coordinate	(a, b, c, d) *
state.texgen[n].object.t	TexGen object linear plane coefficients, unit n, t coordinate	(a, b, c, d) *
state.texgen[n].object.r	TexGen object linear plane coefficients, unit n, r coordinate	(a, b, c, d) *
state.texgen[n].object.q	TexGen object linear plane coefficients, unit n, q coordinate	(a, b, c, d) *
state.texenv[n].color	Texture environment color, unit n	(r, g, b, a) †
state.fog.color	Fog color	(r, g, b, a)

continued

state.fog.params	Fog density, linear start and end, and 1/(end-start)	(d, s, e, r)
state.depth.range	Depth range near, far and (far-near)	(n, f, d, 1) †
state.clip[n].plane	Clip plane n coefficients	(a, b, c, d) *
state.point.size	Point size, minimum and maximum size clamps and fade threshold	(s, n, x, f) *
state.point.attenuation	Point-size attenuation coefficients	(a, b, c, 1) *
state.matrix.modelview	Modelview matrix ‡	
state.matrix.modelview[n]	Modelview matrix n ‡	
state.matrix.projection	Projection matrix ‡	
state.matrix.mvp	Modelview-projection matrix ‡	
state.matrix.texture[n]	Texture matrix n ‡	
state.matrix.palette[n]	Modelview palette matrix n ‡	
state.matrix.program[n]	Generic program matrix n ‡	

* Only available in ARB_vertex_program

† Only available in ARB_fragment_program

‡ The matrix parameters fill four of the standard 4-component vectors by default, each holding one row of the 4×4 matrix. A contiguous subset of the rows can be selected by appending ".row[x..y]" to the state variable name. You can also append ".inverse", ".transpose", or ".invtrans" to the matrix parameters in order to receive the inverse, transpose, or inverse transpose, respectively, of the requested matrix. For example, in order to have rows 1 and 2 of the inverse transpose modelview matrix placed in mvit[0] and mvit[1] respectively, the following command is used:

```
PARAM mvit[] = {state.matrix.modelview.invtrans.row[1..2] };
```

ARB_vertex_program Attribute Vectors

Table A.5 lists the vertex attributes that are available.

Table A.5 Vertex Program Attribute Vectors

Vector	Description	Values
vertex.position	Object space position	(x, y, z, w)
vertex.normal	Normal	(x, y, z, 1)
vertex.color	Primary color	(r, g, b, a)
vertex.color.primary	Primary color	(r, g, b, a)
vertex.color.secondary	Secondary color	(r, g, b, a)
vertex.texcoord	Texture coordinate set 0	(s, t, r, q)
vertex.texcoord[n]	Texture coordinate set n	(s, t, r, q)
vertex.fogcoord	Fog coordinate	(f, 0, 0, 1)
vertex.weight	Vertex weights 0–3	(w, w, w, w)
vertex.weight[n]	Vertex weights n–n+3	(w, w, w, w)
vertex.matrixindex	Vertex matrix indices 0–3	(w, w, w, w)
vertex.matrixindex[n]	Vertex matrix indices n–n+3	(w, w, w, w)
vertex.attrib[n]	Vertex generic attribute n	(x, y, z, w)

ARB_vertex_program Result Vectors

The results that may be written by a vertex program are shown in Table A.6.

Table A.6 Vertex Program Result Vectors

Vector	Description	Values
result.position	Clip-space position	(x, y, z, w)
result.color	Front primary color	(r, g, b, a)
result.color.primary	Front primary color	(r, g, b, a)
result.color.secondary	Front secondary color	(r, g, b, a)
result.color.front	Front primary color	(r, g, b, a)
result.color.front.primary	Front primary color	(r, g, b, a)
result.color.front.secondary	Front secondary color	(r, g, b, a)
result.color.back	Back primary color	(r, g, b, a)
result.color.back.primary	Back primary color	(r, g, b, a)
result.color.back.secondary	Back secondary color	(r, g, b, a)
result.texcoord	Texture coordinate set 0	(s, t, r, q)
result.texcoord[n]	Texture coordinate set n	(s, t, r, q)
result.fogcoord	Fog coordinate	(f, N/A, N/A, N/A)
result.pointsize	Point size	(s, N/A, N/A, N/A)

`ARB_fragment_program` Attribute Vectors

The available attributes of a fragment are listed in Table A.7.

Table A.7 Fragment Program Attribute Vectors

Vector	Description	Values
fragment.position	Window position	(x, y, z, 1/w)
fragment.color	Primary color	(r, g, b, a)
fragment.color.primary	Primary color	(r, g, b, a)
fragment.color.secondary	Secondary color	(r, g, b, a)
fragment.texcoord	Texture coordinate set 0	(s, t, r, q)
fragment.texcoord[n]	Texture coordinate set n	(s, t, r, q)
fragment.fogcoord	Fog coordinate	(f, 0, 0, 1)

`ARB_fragment_program` Result Vectors

Table A.8 lists the results which may be written by a fragment program.

Table A.8 Fragment Program Result Vectors

Vector	Description	Values
result.color	Final color	(r, g, b, a)
result.color[n]	Final color (when using multiple render buffers)	(r, g, b, a)
result.depth	Final depth	(N/A, N/A, d, N/A)