Shader Assembly Language (ARB/NV) Quick Reference Guide for OpenGL®

Describes OpenGL v2.0+ Assembly Language features as specified in the specs ARB_vertex_program, ARB_fragment_program, NV_vertex_program1-4, NV_fragment_program1-4, NV_geometry_program4, and NV_gpu_program4.

OpenGL® SUPPORT FUNCTIONS

glGenProgramsARB()	or glGenProgramsNV()
glBindProgramARB()	or glBindProgramNV()
glProgramStringARB()	or glLoadProgramNV()
glDeleteProgramsARB()	or glDeleteProgramsNV()

glVertexAttrib*ARB()	- sets per-vertex attribute value
glProgramEnvParameter*ARB()	- sets global environment constants
glProgramLocalParameter*ARB()	- sets per-program constants

DATA TYPES

All ARB assembly variables are of type **float4** vectors.
ARB registers are scalar variables were only one float element may be addressed.

PARAMETER TYPES

Attrib	per-vertex attributes such as vertex normals
Local	parameters applied across a program's given shader pass.
Env	parameters are applied across all programs.

DATA TYPE QUALIFIERS

ADDRESS	variables are registers.
ATTRIB	per-vertex attributes.
PARAM	uniform properties - constants, Env or Local.
TEMP	temporary variables
ALIAS	provides alternate names for variables.
OUTPUT	designates variables that are passed back to the pipeline

VARIABLE TYPE MODIFIERS

SHORT, LONG, INT, UINT, FLOAT `e.g. INT TEMP myInteger;`

VECTOR COMPONENTS

full swizzling supported, component names may not be mixed across sets

x, y, z, w
r, g, b, a

```
e.g. ADD R0.xy, V1.zw, V2.zw;  # adds zw components of V1,V2 and
                               # stores result in xy of R0
```

MATRIX MODIFIERS

inverse
transpose
invtrans

`e.g. state.matrix.modelview.invtrans`

PROGRAM SYNTAX

Comment line:	# some comment here
Start program line:	!!ARBvp*, !!NVvp*, !!ARBfp*, !!NVfp*, !!NVgp4.0, etc…
End program line:	END

*=1.0-4.0

ARB_vertex/fragment_program INSTRUCTIONS (section 2.14.4 / 3.11.4)

Instruction	Output	Input	Description
ABS	v	v	absolute value
ADD	v	v,v	add
ARL	a	s	address register load
CMP	v	v,v,v	compare
COS	ssss	s	cosine with reduction to [-PI,PI]
DP3	ssss	v,v	3-component dot product
DP4	ssss	v,v	4-component dot product
DPH	ssss	v,v	homogeneous dot product
DST	v	v,v	distance vector
EX2	ssss	s	exponential base 2
EXP	v	s	exponential base 2 (approximate)
FLR	v	v	floor
FRC	v	v	fraction
KIL	v	v	kill fragment
LG2	ssss	s	logarithm base 2
LIT	v	v	compute light coefficients
LOG	v	s	logarithm base 2 (approximate)
LRP	v	v,v,v	linear interpolation
MAD	v	v,v,v	multiply and add
MAX	v	v,v	maximum
MIN	v	v,v	minimum
MOV	v	v	move
MUL	v	v,v	multiply
POW	ssss	s,s	exponentiate
RCP	ssss	s	reciprocal
RSQ	ssss	s	reciprocal square root
SCS	ss--	s	sine/cosine without reduction
SGE	v	v,v	set on greater than or equal
SIN	ssss	s	sine with reduction to [-PI,PI]
SLT	v	v,v	set on less than
SUB	v	v,v	subtract
SWZ	v	v	extended swizzle
TEX	v	v,u,t	texture sample
TXB	v	v,u,t	texture sample with bias
TXP	v	v,u,t	texture sample with projection
XPD	v	v,v	cross product

Key:
v = indicates a floating-point vector input or output
s = a floating-point scalar input
ssss = indicates a scalar output replicated across a 4-component result vector
ss-- = indicates two scalar outputs in the first two components
a = indicates a single address register component.
u = indicates a texture image unit identifier
t = indicates a texture target

RELATIVE ADDRESSING (NV_gpu_program4)

This extension allows users to declare attribute, result, and temporary arrays such as:

```
PARAM lookup[] = { program.env[100..104] }; # declare an array of 5 variables (float4's) to program Env's

INT TEMP idx;                    # declare an integer variable for array indexing
ROUND.U idx.x, someFloat.x;      # rounds a float variable to nearest unsigned integer index
MOV R0.x, lookup[idx.x];         # index into our array and store the variables x component in R0
```

NV_gpu_program4 INSTRUCTIONS (Section 2.X.4)

Instruction	F	I	C	S	H	D	Out	In	Description
ABS	x	x	x	x	x	F	v	v	absolute value
ADD	x	x	x	x	x	F	v	v,v	add
AND	-	x	x	-	-	S	v	v,v	bitwise and
BRK	-	-	-	-	-	-	-	c	break out of loop instruction
CAL	-	-	-	-	-	-	-	c	subroutine call
CEIL	x	x	x	x	x	F	v	vf	ceiling
CMP	x	x	x	x	x	F	v	v,v,v	compare
CONT	-	-	-	-	-	-	-	c	continue with next loop interation
COS	x	-	x	x	x	F	s	s	cosine with reduction to [-PI,PI]
DIV	x	x	x	x	x	F	v	v,s	divide vector components by scalar
DP2	x	-	x	x	x	F	s	v,v	2-component dot product
DP2A	x	-	x	x	x	F	s	v,v,v	2-comp. dot product w/scalar add
DP3	x	-	x	x	x	F	s	v,v	3-component dot product
DP4	x	-	x	x	x	F	s	v,v	4-component dot product
DPH	x	-	x	x	x	F	s	v,v	homogeneous dot product
DST	x	-	x	x	x	F	v	v,v	distance vector
ELSE	-	-	-	-	-	-	-	-	start if test else block
ENDIF	-	-	-	-	-	-	-	-	end if test block
ENDREP	-	-	-	-	-	-	-	-	end of repeat block
EX2	x	-	x	x	x	F	s	s	exponential base 2
FLR	x	x	x	x	x	F	v	vf	floor
FRC	x	-	x	x	x	F	v	v	fraction
I2F	-	x	x	-	-	S	vf	v	integer to float
IF	-	-	-	-	-	-	-	c	start of if test block
KIL	x	x	-	-	x	F	-	vc	kill fragment
LG2	x	-	x	x	x	F	s	s	logarithm base 2
LIT	x	-	x	x	x	F	v	v	compute lighting coefficients
LRP	x	-	x	x	x	F	v	v,v,v	linear interpolation
MAD	x	x	x	x	x	F	v	v,v,v	multiply and add
MAX	x	x	x	x	x	F	v	v,v	maximum
MIN	x	x	x	x	x	F	v	v,v	minimum
MOD	-	x	x	-	-	S	v	v,s	modulus vector components by scalar
MOV	x	x	x	x	x	F	v	v	move
MUL	x	x	x	x	x	F	v	v,v	multiply
NOT	-	x	x	-	-	S	v	v	bitwise not
NRM	x	-	x	x	x	F	v	v	normalize 3-component vector
OR	-	x	x	-	-	S	v	v,v	bitwise or
PK2H	x	x	-	-	-	F	s	vf	pack two 16-bit floats
PK2US	x	x	-	-	-	F	s	vf	pack two floats as unsigned 16-bit
PK4B	x	x	-	-	-	F	s	vf	pack four floats as signed 8-bit
PK4UB	x	x	-	-	-	F	s	vf	pack four floats as unsigned 8-bit
POW	x	-	x	x	x	F	s	s,s	exponentiate
RCC	x	-	x	x	x	F	s	s	reciprocal (clamped)
RCP	x	-	x	x	x	F	s	s	reciprocal
REP	x	x	-	-	x	F	-	v	start of repeat block
RET	-	-	-	-	-	-	-	c	subroutine return
RFL	x	-	x	x	x	F	v	v,v	reflection vector
ROUND	x	x	x	x	x	F	v	vf	round to nearest integer
RSQ	x	-	x	x	x	F	s	s	reciprocal square root
SAD	-	x	x	-	-	S	vu	v,v,vu	sum of absolute differences
SCS	x	-	x	x	x	F	v	s	sine/cosine without reduction
SEQ	x	x	x	x	x	F	v	v,v	set on equal
SFL	x	x	x	x	x	F	v	v,v	set on false
SGE	x	x	x	x	x	F	v	v,v	set on greater than or equal
SGT	x	x	x	x	x	F	v	v,v	set on greater than

NV_gpu_program4 INSTRUCTIONS (cont.)

Instruction	F	I	C	S	H	D	Out	In	Description
SHL	-	x	x	-	-	S	v	v,s	shift left
SHR	-	x	x	-	-	S	v	v,s	shift right
SIN	x	-	x	x	x	F	s	s	sine with reduction to [-PI,PI]
SLE	x	x	x	x	x	F	v	v,v	set on less than or equal
SLT	x	x	x	x	x	F	v	v,v	set on less than
SNE	x	x	x	x	x	F	v	v,v	set on not equal
SSG	x	-	x	x	x	F	v	v	set sign
STR	x	x	x	x	x	F	v	v,v	set on true
SUB	x	x	x	x	x	F	v	v,v	subtract
SWZ	x	-	x	x	x	F	v	v	extended swizzle
TEX	x	x	x	x	-	F	v	vf	texture sample
TRUNC	x	x	x	x	x	F	v	vf	truncate (round toward zero)
TXB	x	x	x	x	-	F	v	vf	texture sample with bias
TXD	x	x	x	x	-	F	v	vf,vf,vf	texture sample w/partials
TXF	x	x	x	x	-	F	v	vs	texel fetch
TXL	x	x	x	x	-	F	v	vf	texture sample w/LOD
TXP	x	x	x	x	-	F	v	vf	texture sample w/projection
TXQ	-	-	-	-	-	S	vs	vs	texture info query
UP2H	x	x	x	x	-	F	vf	s	unpack two 16-bit floats
UP2US	x	x	x	x	-	F	vf	s	unpack two unsigned 16-bit ints
UP4B	x	x	x	x	-	F	vf	s	unpack four signed 8-bit ints
UP4UB	x	x	x	x	-	F	vf	s	unpack four unsigned 8-bit ints
X2D	x	-	x	x	x	F	v	v,v,v	2D coordinate transformation
XOR	-	x	x	-	-	S	v	v,v	exclusive or
XPD	x	-	x	x	x	F	v	v,v	cross product

Modifiers Key:
F = floating-point data type modifiers (e.g., "ADD.F" component-wise addition of floating-point)
I = signed and unsigned integer data type modifiers (e.g., "ADD.S", "ADD.U")
C = condition code update modifiers (CC, CC0, CC1, see **Conditional Code Tests** section below)
S = clamping (saturation) modifiers (SAT [0,1], SSAT [-1,1])
H = half-precision float data type suffix
D = default data type modifier (F, U, or S)

Input/Output Key:
v: 4-component vector (data type is inherited from operation)
vf: 4-component vector (data type is always floating-point)
vs: 4-component vector (data type is always signed integer)
vu: 4-component vector (data type is always unsigned integer)
s: scalar (replicated if written to vector destination; type inherited from operation)
c: condition code test result (e.g., "EQ", "GT1.x")
vc: 4-component vector or condition code test

Conditional Code Tests: There are two condition codes -- **CC0** (or CC) and **CC1** -- each of which is a four-component vector. The condition codes are set based on the result of an instruction that specifies a condition code update modifier. Examples,

```
ADD.S.CC  R0, R1, R2;   # add signed integers R1 and R2, update  CC0 based on the result, write the final value to R0
MOV R0 (GT1.x), R1;     # move R1 to R0 only if the x component of CC1 indicates a result of ">0"

MOV.U.CC R0, R1;        # move R1 to R0, set condition code
IF GE.x;                # execute only if x component of CC0 is ">=0"
   ...
ENDIF;
```

mask rule	test name	condition	mask rule (cont.)	test name	condition
EQ, EQ0, EQ1	equal	!SF && ZF	LEG, LEG0, LEG1	less, equal, or greater	!SF || !ZF
GE, GE0, GE1	greater than or equal	!(SF ^ OF)	CF, CF0, CF1	carry flag	CF
GT, GT0, GT1	greater than	(!SF ^ OF) && !ZF	NCF, NCF0, NCF1	no carry flag	!CF
LE, LE0, LE1	less than or equal	SF ^ (ZF || OF)	OF, OF0, OF1	overflow flag	OF
LT, LT0, LT1	less than	(SF && !ZF) ^ OF	NOF, NOF0, NOF1	no overflow flag	!OF
NE, NE0, NE1	not equal	SF || !ZF	SF, SF0, SF1	sign flag	SF
FL, FL0, FL1	false	always false	NSF, NSF0, NSF1	no sign flag	!SF
TR, TR0, TR1	true	always true	AB, AB0, AB1	above	CF && !ZF
NAN, NAN0, NAN1	not a number	SF && ZF	BLE, BLE0, BLE1	below or equal	!CF || ZF

VERTEX SHADER CONSTRUCTS

Input Variables *access = RO*

vertex.position	(x,y,z,w)	object coordinates
vertex.weight	(w,w,w,w)	vertex weights 0-3
vertex.weight[n]	(w,w,w,w)	vertex weights n-n+3
vertex.normal	(x,y,z,1)	normal
vertex.color	(r,g,b,a)	primary color
vertex.color.primary	(r,g,b,a)	primary color
vertex.color.secondary	(r,g,b,a)	secondary color
vertex.fogcoord	(f,0,0,1)	fog coordinate
vertex.texcoord	(s,t,r,q)	texture coordinate, unit 0
vertex.texcoord[n]	(s,t,r,q)	texture coordinate, unit n
vertex.matrixindex	(i,i,i,i)	vertex matrix indices 0-3
vertex.matrixindex[n]	(i,i,i,i)	vertex matrix indices n-n+3
vertex.attrib[n]	(x,y,z,w)	generic vertex attribute n

Output Variables *access = WO*

result.position	(x,y,z,w)	position in clip coordinates
result.color	(r,g,b,a)	front-facing primary color
result.color.primary	(r,g,b,a)	front-facing primary color
result.color.secondary	(r,g,b,a)	front-facing secondary color
result.color.front	(r,g,b,a)	front-facing primary color
result.color.front.primary	(r,g,b,a)	front-facing primary color
result.color.front.secondary	(r,g,b,a)	front-facing secondary color
result.color.back	(r,g,b,a)	back-facing primary color[1]
result.color.back.primary	(r,g,b,a)	back-facing primary color
result.color.back.secondary	(r,g,b,a)	back-facing secondary color
result.fogcoord	(f,*,*,*)	fog coordinate
result.pointsize	(s,*,*,*)	point size[2]
result.texcoord	(s,t,r,q)	texture coordinate, unit 0
result.texcoord[n]	(s,t,r,q)	texture coordinate, unit n

[1] enable GL_VERTEX_PROGRAM_TWO_SIDE, [2] enable GL_VERTEX_PROGRAM_POINT_SIZE

FRAGMENT SHADER CONSTRUCTS

Input Variables *access = RO*

fragment.color	(r,g,b,a)	primary color
fragment.color.primary	(r,g,b,a)	primary color
fragment.color.secondary	(r,g,b,a)	secondary color
fragment.texcoord	(s,t,r,q)	texture coordinate, unit 0
fragment.texcoord[n]	(s,t,r,q)	texture coordinate, unit n
fragment.fogcoord	(f,0,0,1)	fog distance/coordinate
fragment.position	(x,y,z,1/w)	window pixel position
fragment.clip[n]	(c,-,-,-)	interpolated clip distance n
fragment.attrib[n]	(x,y,z,w)	generic interpolant n
fragment.clip[n..o]	(c,-,-,-)	clip distances n thru o
fragment.facing	(f,-,-,-)	fragment facing
primitive.id	(id,-,-,-)	primitive number

Output Variables *access = WO*

result.color	(r,g,b,a)	color
result.color[n]	(r,g,b,a)	color output n
result.depth	(*,*,d,*)	depth coordinate

GEOMETRY SHADER CONSTRUCTS

Input Variables *access = RO*

vertex[m].position	(x,y,z,w)	clip coordinates
vertex[m].color	(r,g,b,a)	front primary color
vertex[m].color.primary	(r,g,b,a)	front primary color
vertex[m].color.secondary	(r,g,b,a)	front secondary color
vertex[m].color.front	(r,g,b,a)	front primary color
vertex[m].color.front.primary	(r,g,b,a)	front primary color
vertex[m].color.front.secondary	(r,g,b,a)	front secondary color
vertex[m].color.back	(r,g,b,a)	back primary color
vertex[m].color.back.primary	(r,g,b,a)	back primary color
vertex[m].color.back.secondary	(r,g,b,a)	back secondary color
vertex[m].fogcoord	(f,-,-,-)	fog coordinate
vertex[m].pointsize	(s,-,-,-)	point size
vertex[m].texcoord	(s,t,r,q)	texture coordinate, unit 0
vertex[m].texcoord[n]	(s,t,r,q)	texture coordinate, unit n
vertex[m].attrib[n]	(x,y,z,w)	generic interpolant n
vertex[m].clip[n]	(d,-,-,-)	clip plane distance
vertex[m].texcoord[n..o]	(s,t,r,q)	array of texture coordinates
vertex[m].attrib[n..o]	(x,y,z,w)	array of generic interpolants
vertex[m].clip[n..o]	(d,-,-,-)	array of clip distances
vertex[m].id	(id,-,-,-)	vertex id
primitive.id	(id,-,-,-)	primitive number

Output Variables *access = WO*

result.position	(x,y,z,w)	position in clip coordinates
result.color	(r,g,b,a)	front-facing primary color
result.color.primary	(r,g,b,a)	front-facing primary color
result.color.secondary	(r,g,b,a)	front-facing secondary color
result.color.front	(r,g,b,a)	front-facing primary color
result.color.front.primary	(r,g,b,a)	front-facing primary color
result.color.front.secondary	(r,g,b,a)	front-facing secondary color
result.color.back	(r,g,b,a)	back-facing primary color
result.color.back.primary	(r,g,b,a)	back-facing primary color
result.color.back.secondary	(r,g,b,a)	back-facing secondary color
result.fogcoord	(f,*,*,*)	fog coordinate
result.pointsize	(s,*,*,*)	point size
result.texcoord	(s,t,r,q)	texture coordinate, unit 0
result.texcoord[n]	(s,t,r,q)	texture coordinate, unit n
result.attrib[n]	(x,y,z,w)	generic interpolant n
result.clip[n]	(d,*,*,*)	clip plane distance
result.texcoord[n..o]	(s,t,r,q)	texture coordinates n thru o
result.attrib[n..o]	(x,y,z,w)	generic interpolants n thru o
result.clip[n..o]	(d,*,*,*)	clip distances n thru o
result.primid	(id,*,*,*)	primitive id
result.layer	(l,*,*,*)	layer for cube/array/3D FBOs

CONFIG SETTINGS

```
PRIMITIVE_IN   - { POINTS, LINES, LINES_ADJACENCY, TRIANGLES, TRIANGLES_ADJACENCY }
PRIMITIVE_OUT  - { POINTS, LINE_STRIP, TRIANGLE_STRIP }
VERTICES_OUT # - number of vertices geometry shader can emit
```

FUNCTIONS

```
EMIT           - emit vertex (result variables may become undefined after EMIT)
ENDPRIM        - end of primitive
```

BUILT-IN PROPERTY BINDINGS

Program Env/Local Bindings

program.env[a]	(x,y,z,w)	program environment parameter a
program.local[a]	(x,y,z,w)	program local parameter a
program.env[a..b]	(x,y,z,w)	program environment parameters a through b
program.local[a..b]	(x,y,z,w)	program local parameters a through b

Material Bindings

state.material.ambient	(r,g,b,a)	front ambient material color
state.material.diffuse	(r,g,b,a)	front diffuse material color
state.material.specular	(r,g,b,a)	front specular material color
state.material.emission	(r,g,b,a)	front emissive material color
state.material.shininess	(s,0,0,1)	front material shininess
state.material.front.ambient	(r,g,b,a)	front ambient material color
state.material.front.diffuse	(r,g,b,a)	front diffuse material color
state.material.front.specular	(r,g,b,a)	front specular material color
state.material.front.emission	(r,g,b,a)	front emissive material color
state.material.front.shininess	(s,0,0,1)	front material shininess
state.material.back.ambient	(r,g,b,a)	back ambient material color
state.material.back.diffuse	(r,g,b,a)	back diffuse material color
state.material.back.specular	(r,g,b,a)	back specular material color
state.material.back.emission	(r,g,b,a)	back emissive material color
state.material.back.shininess	(s,0,0,1)	back material shininess

Light Bindings

state.light[n].ambient	(r,g,b,a)	light n ambient color
state.light[n].diffuse	(r,g,b,a)	light n diffuse color
state.light[n].specular	(r,g,b,a)	light n specular color
state.light[n].position	(x,y,z,w)	light n position
state.light[n].attenuation	(a,b,c,e)	light n attenuation constants and spot light exponent
state.light[n].spot.direction	(x,y,z,c)	light n spot direction and cutoff angle cosine
state.light[n].half	(x,y,z,1)	light n infinite half-angle
state.lightmodel.ambient	(r,g,b,a)	light model ambient color
state.lightmodel.scenecolor	(r,g,b,a)	light model front scene color
state.lightmodel.front.scenecolor	(r,g,b,a)	light model front scene color
state.lightmodel.back.scenecolor	(r,g,b,a)	light model back scene color
state.lightprod[n].ambient	(r,g,b,a)	light n / front material ambient color product
state.lightprod[n].diffuse	(r,g,b,a)	light n / front material diffuse color product
state.lightprod[n].specular	(r,g,b,a)	light n / front material specular color product
state.lightprod[n].front.ambient	(r,g,b,a)	light n / front material ambient color product
state.lightprod[n].front.diffuse	(r,g,b,a)	light n / front material diffuse color product
state.lightprod[n].front.specular	(r,g,b,a)	light n / front material specular color product
state.lightprod[n].back.ambient	(r,g,b,a)	light n / back material ambient color product
state.lightprod[n].back.diffuse	(r,g,b,a)	light n / back material diffuse color product
state.lightprod[n].back.specular	(r,g,b,a)	light n / back material specular color product

Texture Bindings

state.texgen[n].eye.s	(a,b,c,d)	TexGen eye linear plane coefficients, s coord, unit n
state.texgen[n].eye.t	(a,b,c,d)	TexGen eye linear plane coefficients, t coord, unit n
state.texgen[n].eye.r	(a,b,c,d)	TexGen eye linear plane coefficients, r coord, unit n
state.texgen[n].eye.q	(a,b,c,d)	TexGen eye linear plane coefficients, q coord, unit n
state.texgen[n].object.s	(a,b,c,d)	TexGen object linear plane coefficients, s coord, unit n
state.texgen[n].object.t	(a,b,c,d)	TexGen object linear plane coefficients, t coord, unit n
state.texgen[n].object.r	(a,b,c,d)	TexGen object linear plane coefficients, r coord, unit n
state.texgen[n].object.q	(a,b,c,d)	TexGen object linear plane coefficients, q coord, unit n

Texture Enviroment Bindings

state.texenv[n].color	(r,g,b,a)	texture environment n color

Fog Bindings

state.fog.color	(r,g,b,a)	RGB fog color
state.fog.params	(d,s,e,r)	fog density, linear start and end, and 1/(end-start)

Clip Plane Bindings

state.clip[n].plane	(a,b,c,d)	clip plane n coefficients

Point Bindings

state.point.size	(s,n,x,f)	point size, min and max size clamps, and fade threshold
state.point.attenuation	(a,b,c,1)	point size attenuation consts

Depth Bindings

state.depth.range	(n,f,d,1)	Depth range near, far, and (far-near)

Matrix Bindings

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]	program matrix n