

# Contents

<b>HSA Conformance Manual</b>	<b>1</b>
Overview . . . . .	1
Building the test suite . . . . .	2
Running the test suite . . . . .	2
Command line options . . . . .	3
Interpreting results . . . . .	3
Test names . . . . .	4
Test parameters . . . . .	4
Test details . . . . .	10
Test suite internals . . . . .	25

## HSA Conformance Manual

HSA PRM (Programmer's Reference Manual) Specification revision: 1.0 Final  
(2015-03-06)

HSA PSA (Platform System Architecture) Specification revision: 1.0 Final  
(2015-01-23)

Test suite revision: 2015-05-25 (commit 281a954e86c63dd00c064d7c033e7ae46c524fe0)

## Overview

HSA Conformance test suite is used to validate an implementation of Heterogeneous System Architecture for compliance with the following specifications:

- HSA PRM (Programmer's Reference Manual) Specification;
- HSA PSA (Platform System Architecture) Specification.

The focus is on verification of ISA produced by HSAIL Finalizer and functionality of kernels on the agent. HSAIL Linking extension is used to finalize test programs while HSA Core Runtime is used to set up and invoke test kernels and to provide certain functionality for test code on the host (for some tests).

A test consists of scenario which describes test data, HSAIL program(s) in BRIG format, agent(s) code, dispatches and validation. The test is deemed passed if all scenario steps (including validation) are successful. Many basic tests for

HSAIL instruction contain just one program with one kernel, one or several input and output buffers (used to store test data and test results), one dispatch and validation of output buffer. An example of such test is a test for `add_u32` HSAIL instruction: one kernel, two input buffers and one output buffer.

Note that each test actually expands into several testcases, one for each valid combination of parameters. These parameters are described in [Test details](#). For example, a test for the `abs` instruction consists of several tests for `f32` and `f64` types.

## Building the test suite

HSA Conformance test suite uses [CMake](#). The suite also requires the following components:

- [HSAIL-Tools](#) repository, pointed to by `HSAIL-Tools-PATH` variable
- [hsa.h](#) header file, pointed to by `HSA-Runtime-Inc-PATH`
- [hsa\\_ext\\_finalize.h](#) header file, pointed to by `HSA-Runtime-Ext-PATH`

`cmake_windows.sh` and `cmake_linux.sh` scripts contain examples of command lines that can be used to run CMake on Windows and Linux.

## Running the test suite

The test suite can be invoked as “hc” program that runs on the host.

To run the suite you should have a dynamic HSA RT libraries with corresponding bitness in the path (e.g. `PATH` variable for Windows, `LD_LIBRARY_PATH` for Linux).

A simple scenario of running all the tests on Linux 64-bit may look like:

```
export LD_LIBRARY_PATH="$HSA_RT_LIB:$LD_LIBRARY_PATH"
bin/lrx64a/hc -tests /
               -exclude amdhsa.exclude
               -runner hrunner
               -verbose
               -testlog results_linux64.log
```

where `$HSA_RT_LIB` points to the HSA RT libraries, e.g.  
`/compiler/dist/Obsidian/dist/linux/debug.hsa_foundation/lib/x86_64`.

For convenience the package also contains Linux and Windows batch files, which take path to the HSA Runtime binaries and run the whole suite.

The following example demonstrates how to run these scripts on Linux. The results grouped by sub-suites will be displayed during the run along with overall statistics, per test results can be found in results\_linux64.log:

```
$ ./run.sh /compiler/dist/Obsidian/dist/linux/debug.hsa_foundation/

prm/arithmetic/intfp/abs
  Passed:    56   Failed:    0   Error:    0   NA:    0   Total:    56
prm/arithmetic/intfp/add
  Passed:   255   Failed:    0   Error:    0   NA:    0   Total:   255
...
prm/special/misc/laneid
  Passed:    24   Failed:    0   Error:    0   NA:    0   Total:    24

Testrun
  Passed: 13729   Failed:    0   Error:    0   NA:    0   Total: 13729
```

Using `-tests` option it is possible to specify a smaller set of tests, while `-exclude` allows to exclude tests (e.g. failing due to issues with the current HSA RT implementation).

## Command line options

The “hc” supports the following command line options:

- `-tests TestSet`: prefix of test to run, e.g. `-tests /` to run all tests or `-tests prm/` to run only PRM tests;
- `-exclude File`: file containing a list of tests to be excluded from testing;
- `-verbose`: enables detailed test output in a log file;
- `-testlog File`: name for a log file, the default name is test.log;
- `-runner Runner`: a mode of test grouping. May be either `hrunner` (default) or `simple`. By default tests are grouped by category. `simple` runner may be specified to avoid tests grouping. See option `-testloglevel` which also affects grouping.
- `-testloglevel`: test grouping depth, the default is 4. See also `-runner` option;
- `-dump`: dump HSAIL and BRIG test sources for each test under corresponding folder (prm/...);
- `-results`: path to folder which will contain dumped test sources (prm/...), the default is the current folder.

## Interpreting results

*TODO*: add example of how to interpret test output (both standard and detailed).

## Test names

Every test has a unique identifier: test name. A test name consists of several words in lower case separated by slash (/). First word is either **prm** or **sysarch**; it is the name of HSA specification this test checks compliance with. An example of test name is **prm/special/dispatchpacket/dim/basic/kernel\_1\_200x1x1\_64x1x1\_ND**. The following scheme for the words is used (starting from the beginning of test name):

- **prm**: Test suite (**prm** or **sysarch**)
- **special**: Chapter in PRM or PSA specification
- **dispatchpacket**: Section in PRM or PSA specification
- **dim**: Instruction or group of instructions
- **basic**: Name of test/scenario
- **kernel\_1\_200x1x1\_64x1x1\_ND**: Name/identifier of testcase within test. For **dim/basic** test, it is the testcase for **dim** instruction in kernel body with dispatch dimension 1, grid sizes 200x1x1, workgroup sizes 64x1x1 and no control directives.

The naming scheme for testcases depends on test. For many tests, it contains text representation of parameters described in [Test details](#), separated by underscore (\_). For some tests, simple counter is used. Refer to documentation below for the list of parameters.

A prefix of test names identifies test category. For example, **prm/special/dispatchpacket/** are tests for dispatch packet operations. These categories may be used to select which tests to run (see option **-tests**).

## Test parameters

**Code Location:** location of validated code

- Used sets:
- **All**: kernel, function
- **Kernel**: just kernel

**Grid Geometry:** geometry of a dispatch

- Consists of:
  - Number of dimensions
  - Grid sizes for **x**, **y**, **z**. Unused dimensions have value 1.
  - Workgroup sizes for **x**, **y**, **z**. Unused dimensions have value 1.
- Used sets:

- All: various geometries, includes samples of partial workgroups/workitems, corner cases
- DimensionSet: representative geometry for each dimension
- OneGroupSet: one group, one dimension geometry
- DefaultPlusNGroupSet: set of geometry with 1 WorkItem, OneGroupSet, and 1 dim & N groups
- Boundary32Set: samples of geometry with sizes  $> 2^{32}$
- Boundary24Set: samples of geometry with sizes  $> 2^{24}$
- DegenerateSet: samples of degenerated geometry when a dimension is used, but has size 1
- BarrierSet: samples of one-dimensional geometries for barrier testing
- FBarrierSet: special samples of geometries for fbarrier testing
- ImageSet: special samples of geometries for testing operations with images and samples
- ImageRdSet: special samples of geometries for testing rdimimage operation
- AtomicSet: special samples of geometries for testing atomic operations
- MModelSet: special samples of geometries for testing memory model

**Control Directives: control directives in BRIG modules [PRM, chapter 13.4]**

- Control directives may be controlled with the following settings:
  - Location: location of control directives (function, kernel or module).
  - List of enabled control directives.
- A test shall use all possible combinations of enabled control directives. For example, if two directives are enabled - **requiredgridsize** and **requiredworkgroupsize**, the test shall use the following combinations:
  - [] (no control directives)
  - [ **requiredgridsize** ]
  - [ **requiredworkgroupsize** ]
  - [ **requiredgridsize**, **requiredworkgroupsize** ]
- Used sets:
  - DimensionSet: enabled directive is **requiredddim** (directives affecting dimensions)
  - GeometrySet: enabled directives are **requiredddim**, **requiredgridsize**, **requiredworkgroupsize** (directives affecting grid geometry)
  - GridSizeSet: enabled directives are **requiredddim**, **requiredgridsize** (directives affecting grid size)

- **DegenerateSet:** enabled directives are `requiredgridsize`, `requiredworkgroupsize` (directives affecting computation of operations that can be simplified for degenerate set)

**Segment:** memory segment [PRM, chapter 2.8]

- Used sets:
  - **Atomic:** flat, global, group
  - **HasFlatAddress:** global, group, private (segments which may be accessed via flat address)
  - **MemFence:** global, group
  - **Variable:** global, readonly, kernarg, group, private, spill, arg
  - **All:** all standard segments

**Linkage:** variable linkage [PRM, chapter 4.12]

- Used sets:
  - **All:** program, module, linkage, arg

**Operand Kind:** kind of instruction operand [PRM, chapter 4.16]

- Used sets:
  - **All:** all legal operands

**Dst Type:** type of destination operand [PRM, chapter 4.13]

- Used sets:
  - **All:** all legal types

**Src Type:** type of source operands [PRM, chapter 4.13]

- Used sets:
  - **All:** all legal types

**Ftz:** flush to zero modifier [PRM, chapter 4.19.3]

- Used sets:
  - **All:** all legal values

**Rounding: rounding modifier** [PRM, chapter 4.19.2]

- Used sets:
  - All: all legal values

**Packed Controls: controls for processing packed operands** [PRM, chapter 4.14]

- Used sets:
  - All: all legal values

**Compare Operation: operation used by cmp instruction** [PRM, chapter 5.18]

- Used sets:
  - All: all operations

**Atomic Operation: operation used by atomic instructions** [PRM, chapter 6.6]

- Used sets:
  - All: all atomic operations
  - Atomic: all atomic operations except for ld

**Align: alignment** [PRM, chapter 6.1.3]

- Used sets:
  - All: all legal values

**Const: constant memory access** [PRM, chapter 6.3.2]

- Used sets:
  - All: all legal values

**Equiv: equivalence class** [PRM, chapter 6.1.4]

- Used sets:
  - All: all legal values

### **Memory Order [PRM, chapter 6.2.1]**

- Used sets:
  - **All**: rlx, screl, scacq, scar
  - **SignalAll**: rlx, screl, scacq, scar
  - **SignalWait**: rlx, screl, scacq
  - **MemFence**: screl, scacq, scar

### **Memory Scope [PRM, chapter 6.2.2]**

- Used sets:
  - **All**: wi, wv, wg, cmp, sys
  - **Global**: wv, wg, cmp, sys
  - **Group**: wv, wg

### **Width: width modifier [PRM, chapter 6.3.2]**

- Used sets:
  - **UpToWavesizeAndAll**: 1, 2, 4, 8, 16, 32, WAVESIZE, All
  - **All**: all legal values

### **Test Data: data for testing arithmetic and memory operations**

- Used sets:
  - **Standard**:
    - \* Includes the following values:
      - regular values;
      - boundary values (e.g. -128 and 127 for s8);
      - special values (NaNs, infinities, subnormals, etc);
      - values producing regular, boundary or special values as a result.
    - \* Does not include the following values:
      - values which signal exceptions;
      - values which result in an undefined behavior.

### **Exceptions Mask: a number that specifies a list of exceptions [PRM, chapter 11.2.2]**

- Used sets:
  - **All**: all legal values



**Image Geometry:** geometries associated with images [PRM, chapter 7.1.3]

- Used sets:
  - All: all legal values

**Image Channel Order:** channel orders associated with images [PRM, chapter 7.1.4.1]

- Used sets:
  - All: all legal values

**Image Channel Type:** channel types associated with images [PRM, chapter 7.1.4.2]

- Used sets:
  - All: all legal values

**Image Array:** image array size

- Used sets:
  - All: 1, 2, 10

**Image Property:** image properties [PRM, chapter 7.5.2]

- Used sets:
  - All: all legal values

**Sampler Coord:** sampler coordinate normalization mode [PRM, chapter 7.1.8]

- Used sets:
  - All: all legal values

**Sampler Filter:** sampler filter mode [PRM, chapter 7.1.8]

- Used sets:
  - All: all legal values

**Sampler Addressing:** sampler addressing mode [PRM, chapter 7.1.8]

- Used sets:
  - All: all legal values

**Sampler Property:** sampler properties [PRM, chapter 7.5.2]

- Used sets:
  - All: all legal values

## Test details

**initializer:** Variable Initializers [PRM, chapter 4.10]

- Segment: Variable
- Grid Geometry: OneGroupSet
- Code Location: Kernel

**arithmetic:** Arithmetic operations [PRM, chapter 5]

**intfp:** Integer/floating point arithmetic operations [PRM, chapters 5.2 and 5.11]

- Instructions: abs, add, borrow, carry, div, max, min, mul, mulhi, neg, rem, sub, ceil, floor, fma, fract, rint, sqrt, trunc
- Operand Kind: All
- Dst Type: All
- Ftz: All
- Rounding: All
- Packed controls: All
- Grid Geometry: OneGroupSet
- Code Location: Kernel
- Test Data: Standard

*Note: some tests for operations with f16 are not implemented yet.*

*Note: some tests for rounding modes are not implemented yet.*

*Note: tests for fma with f64 type are not implemented yet.*

**intopt: Integer and floating-point optimization operations** [PRM, chapter 5.3 and 5.12]

- Instructions: mad
- Operand Kind: All
- Dst Type: All
- Ftz: All
- Rounding: All
- Grid Geometry: OneGroupSet
- Code Location: Kernel
- Test Data: Standard

**24int: 24-bit integer optimization operations** [PRM, chapter 5.4]

- Instructions: mad24, mad24hi, mul24, mul24hi
- Operand Kind: All
- Dst Type: All
- Grid Geometry: OneGroupSet
- Code Location: Kernel
- Test Data: Standard

**intshift: Integer shift operations** [PRM, chapter 5.5]

- Instructions: shl, shr
- Operand Kind: All
- Dst Type: All
- Grid Geometry: OneGroupSet
- Code Location: Kernel
- Test Data: Standard

**indbit: Individual bit operations** [PRM, chapter 5.6]

- Instructions: and, or, xor, not, popcount
- Operand Kind: All
- Dst Type: All
- Src Type: All
- Grid Geometry: OneGroupSet
- Code Location: Kernel
- Test Data: Standard

**bitstr: Bit string operations [PRM, chapter 5.7]**

- Instructions: bitextract, bitinsert, bitmask, bitrev, bitselect, firstbit, lastbit
- Operand Kind: All
- Dst Type: All
- Src Type: All
- Grid Geometry: OneGroupSet
- Code Location: Kernel
- Test Data: Standard

**copymove: Copy and move operations [PRM, chapter 5.8]**

- Instructions: combine, expand, mov
- Operand Kind: All
- Dst Type: All
- Src Type: All
- Grid Geometry: OneGroupSet
- Code Location: Kernel
- Test Data: Standard

**packed: Packed data operations [PRM, chapter 5.9]**

- Instructions: shuffle, unpacklo, unpackhi, pack, unpack
- Operand Kind: All
- Dst Type: All
- Src Type: All
- Grid Geometry: OneGroupSet
- Code Location: Kernel
- Test Data: Standard

**bitcmov: Bit conditional move operation [PRM, chapter 5.10]**

- Instructions: cmov
- Operand Kind: All
- Dst Type: All
- Grid Geometry: OneGroupSet
- Code Location: Kernel
- Test Data: Standard

*Note: some tests for rounding modes are not implemented yet.*

**fpbit: Floating-point bit operations [PRM, chapter 5.13]**

- Instructions: class, copysign
- Operand Kind: All
- Dst Type: All
- Src Type: All
- Packed controls: All
- Grid Geometry: OneGroupSet
- Code Location: Kernel
- Test Data: Standard

**nativefp: Native floating-point operations [PRM, chapter 5.14]**

- Instructions: nsin, ncos, nlog2, nexp2, nsqrt, nrsqrt, nrcp, nfma
- Operand Kind: All
- Dst Type: All
- Grid Geometry: OneGroupSet
- Code Location: Kernel
- Test Data: Standard

**multimedia: Multimedia operations [PRM, chapter 5.15]**

- Instructions: bitalign, bytealign, lerp, packcvt, unpackcvt, sad, sadhi
- Operand Kind: All
- Dst Type: All
- Src Type: All
- Grid Geometry: OneGroupSet
- Code Location: Kernel
- Test Data: Standard

**compare: Compare operation [PRM, chapter 5.18]**

- Instructions: cmp
- Operand Kind: All
- Dst Type: All
- Src Type: All
- Compare Operation: All
- Ftz: All
- Grid Geometry: OneGroupSet
- Code Location: Kernel
- Test Data: Standard

**conversion:** Conversion operation [PRM, chapter 5.19]

- Instructions: cvt
- Operand Kind: All
- Dst Type: All
- Src Type: All
- Rounding: All
- Ftz: All
- Grid Geometry: OneGroupSet
- Code Location: Kernel
- Test Data: Standard

*Note: some tests for operations with f16 are not implemented yet.*

*Note: some tests for rounding modes are not implemented yet.*

**address:** Address operations [PRM, chapters 5 and 11]

**null:** Verify result of address operation for null address obtained with `nullptr` [PRM, chapter 11.4]

- Instructions: stof, ftos, segmentp
- Segment: HasFlatAddress
- Grid Geometry: OneGroupSet
- Code Location: Kernel

**identity:** Verify that converted address accesses the same location as address before conversion [PRM, chapter 5.17]

- Instructions: stof, ftos
- Segment: HasFlatAddress
- SegmentStore: use segment store/flat load and not vice versa
- nonull: use nonull in the instruction
- Grid Geometry: OneGroupSet
- Code Location: Kernel

**variable:** Verify result of segmentp operation for flat address pointing to a variable [PRM, chapter 5.16]

- Instructions: segmentp
- Segment: HasFlatAddress
- nonull: use nonull in the instruction
- Grid Geometry: OneGroupSet
- Code Location: Kernel

**lda/alignment:** Verify that result of lda operation is divisible by alignment [PRM, chapter 5.8]

- Instructions: lda
- Segment: HasFlatAddress
- nonull: use nonull in the instruction
- Grid Geometry: OneGroupSet
- Code Location: Kernel

**memory:** Memory operations [PRM, chapter 6]

**memory/ordinary:** Ordinary memory operations [PRM, chapters 6.3 and 6.4]

- Instructions: ld, st
- Operand Kind: All
- Dst Type: All
- Segment: All
- Align: All
- Const: All
- Equiv: All
- Width: All
- Grid Geometry: OneGroupSet
- Code Location: Kernel
- Test Data: Standard

*Note: tests for operations with b128 and opaque vectors are not implemented yet.*

**memory/atomic:** Atomic and atomicnoret memory operations [PRM, chapter 6.6 and 6.7]

- Instructions: atomic, atomicnoret
- Operand Kind: All
- Dst Type: All
- Atomic operation: All
- Segment: Atomic
- Memory Order: All
- Memory Scope: All
- Equiv: 0
- Grid Geometry: OneGroupSet
- Code Location: Kernel
- Test Data: Standard

**memory/atomicity: Test atomicity of Atomic and atomicnoret operations [PRM, chapter 6.5]**

- Instructions: atomic, atomicnoret
- Operand Kind: All
- Dst Type: All
- Atomic operation: Atomic
- Segment: Atomic
- Memory Order: All
- Memory Scope: All
- Equiv: 0
- Grid Geometry: AtomicSet
- Code Location: Kernel
- Test Data: Standard

**memory/signal: Notification operations [PRM, chapter 6.8]**

- Instructions: signal, signalnoret
- Memory Order: SignalAll, SignalWait
- Grid Geometry: OneGroupSet
- Code Location: Kernel

**memfence: Memory fence operation [PRM, chapter 6.9]**

- Instructions: memfence
- Grid Geometry: OneGroupSet
- Code Location: Kernel
- Segment: Memfence
- Memory Order: Memfence
- Memory Scope: Global, Group

**Image operations [PRM, chapter 7]**

**initializer/image: Image Creation and Image Handles [PRM, chapter 7.1.7]** *Tests are to be implemented.*

**initializer/sampler: Sampler Creation and Sampler Handles [PRM, chapter 7.1.8]**

- Grid Geometry: OneGroupSet
- Code Location: All



- Sampler Coord: All
- Sampler Filter: All
- Sampler Addressing: All

**image\_rd: Read Image Instruction [PRM, chapter 7.2]**

- Instructions: rdimage
- Grid Geometry: ImageRdSet
- Code Location: All
- Image Geometry: All
- Image Channel Order: All
- Image Channel Type: All
- Image Array: All
- Sampler Coord: All
- Sampler Filter: All
- Sampler Addressing: All
- Equiv: 0

**image\_ld: Load Image Instruction [PRM, chapter 7.3]**

- Instructions: ldimage
- Grid Geometry: ImageSet
- Code Location: All
- Image Geometry: All
- Image Channel Order: All
- Image Channel Type: All
- Image Array: All
- Equiv: 0

**image\_st: Store Image Instruction [PRM, chapter 7.4]**

- Instructions: stimage
- Grid Geometry: ImageSet
- Code Location: All
- Image Geometry: All
- Image Channel Order: All
- Image Channel Type: All
- Image Array: All
- Equiv: 0

**image\_query: Query Image Instruction [PRM, chapter 7.5]**

- Instructions: queryimage
- Grid Geometry: ImageSet
- Code Location: All
- Image Geometry: All
- Image Channel Order: All
- Image Channel Type: All
- Image Array: All
- Image Property: All

**image\_query\_sampler: Query Sampler Instruction [PRM, chapter 7.5]**

- Instructions: queriesampler
- Grid Geometry: ImageSet
- Code Location: All
- Sampler Coord: All
- Sampler Filter: All
- Sampler Addressing: All
- Sampler Property: All

**imagefence: Image Fence Instruction [PRM, chapter 7.5]** *Tests are to be implemented.*

**Branch operations [PRM, chapter 8]**

**basic/br: Basic unconditional jump, verify expected result by setting value of HSAIL register [PRM, chapter 8]**

- Instructions: br
- Code Location: All
- Grid Geometry: OneGroupSet

**basic: Basic conditional jump, switch conditional jump [PRM, chapter 8]**

- Instructions: cbr, sbr
- Grid Geometry: DefaultPlusNGroupSet
- Code Location: Kernel
- Width: UpToWavesizeAndAll
- Operand Kind: All

**nested:** Nested control flow (if-then-else) [PRM, chapter 8]

- Instructions: cbr, br
- Grid Geometry: DefaultPlusNGroupSet
- Code Location: Kernel
- Width: UpToWavesizeAndAll
- Operand Kind: All

**sand, sor:** Short-circuit control flow [PRM, chapter 8]

- Instructions: cbr
- Grid Geometry: DefaultPlusNGroupSet
- Code Location: Kernel
- Width: UpToWavesizeAndAll
- Operand Kind: All

**Parallel Synchronization and communication operations** [PRM, chapter 9]

**barrier:** Barrier operations [PRM, chapter 9.1]

- Instructions: barrier
- Grid Geometry: BarrierSet
- Code Location: Kernel
- Segment: Atomic
- Memory Scope: All
- Memory Order: All

*Tests for **wavebarrier** are to be implemented.*

**Fine-grain barrier operations** [PRM, chapter 9.2]

- Instructions: initfbar, joinfbar, waitfbar, arrivefbar, leavefbar, releasefbar, ldf
- Grid Geometry: FBarrierSet
- Code Location: Kernel

**crosslane:** Cross-lane operations [PRM, chapter 9.4]

- Instructions: activelaneid, activelaneid, activelanemask
- Grid Geometry: OneGroupSet
- Code Location: All

*Tests for **activelanepermute** are to be implemented.*

**Function operations** [PRM, chapter 10]

**Direct call operation** [PRM, chapter 10.6]

**arguments:** Verify passing argument/returning result of given type  
[PRM, chapter 10.2]

- Types: all BRIG types
- Grid Geometry: OneGroupSet
- Code Location: All

*More tests are to be implemented.*

**special: Special operations** [PRM, chapter 11]

**dispatchpacket:** Dispatch packet operations [PRM, chapter 11.1]

**basic: Verify result of dispatch packet operation**

- Instructions: currentworkgroupsize, dim, gridgroups, gridsize, workgroupid, workgroupsize, workitemabsid, workitemflatabsid, workitemflatid, workitemid
- Code Location: All
- Grid Geometry: DimensionSet for dim, All for others
- Control Directives: DimensionSet for dim, GeometrySet for others
- Type: u32/u64 for gridsize, workitemflatabsid, workitemflatid, u32 for others

**boundary32: Verify result of dispatch packet operation for workitems with workitemflatabsid around  $2^{32}$  boundary**

- Instructions: gridsize, workitemflatabsid, workitemflatid
- Code Location: All
- Grid Geometry: Boundary32Set
- Control Directives: GridSizeSet
- Type: u32/u64

**boundary24: Verify result of dispatch packet operation for possible mul24 finalizer optimizations around  $2^{24}$  boundary**

- Instructions: workitemabsid, workitemflatabsid, workitemflatid

- Code Location: All
- Grid Geometry: Boundary24Set
- Control Directives: Boundary24Set
- Type: u32/u64 for workitemflatabsid, workitemflatid, u32 for other instructions

**degenerate: Verify result of dispatch packet operation for used dimension that dispatched with size 1**

- Instructions: currentworkgroupsize, gridgroups, gridsize, workgroupid, workgroupsize, workitemabsid, workitemflatid, workitemid
- Code Location: All
- Grid Geometry: DegenerateSet
- Control Directives: DegenerateSet
- Type: u32/u64 for workitemflatabsid, workitemflatid, u32 for other instructions

**packetid/basic: Compare result of packetid operation with value on the host**

- Instructions: packetid
- Code Location: All
- Grid Geometry: Boundary32Set
- Control Directives: None

**packetcompletionsig/basic: Compare result of packetcompletionsig operation with value on the host**

- Instructions: packetcompletionsig
- Code Location: All
- Grid Geometry: Boundary32Set
- Control Directives: None

**Exception operations [PRM, chapter 11.2]**

- Grid Geometry: OneGroupSet
- Code Location: All or Kernel
- Exceptions mask: All (where applicable)

**usermodequeue: User mode queue operations [PRM, chapter 11.3]**

**basic:** Verify result of queueid, queueptr, ldk operation for a dispatch of a kernel

- Instructions: queueid, queueptr, ldk
- Grid Geometry: OneGroupSet
- Code Location: All

**basicindex:** Verify result of queue index operation on a user mode queue

- Instructions: ldqueuereadindex, ldqueuwriteindex, addqueuwriteindex, casqueuwriteindex, stqueuereadindex, stqueuwriteindex
- Code Location: All
- Grid Geometry: OneGroupSet
- User mode queue type: separate, customized for each test

**misc:** Miscellaneous operations [PRM, chapter 11.4]

**kernargbaseptr/identity:** Verify accessing memory at kernargbaseptr address

- Grid Geometry: OneGroupSet
- Code Location: All

**kernargbaseptr/alignment:** Verify alignment of kernargbaseptr depending on alignment of kernel arguments

- Grid Geometry: OneGroupSet
- Code Location: All

**nop:** Verify kernel with nop instruction

- Grid Geometry: OneGroupSet
- Code Location: All

**cuid/lessmax:** Verify that result of cuid operation is always less than maxcuid

- Code Locations: All
- Grid Geometry: All

**cuid/identity:** Verify that result of cuid operation is same across workgroup

- Code Locations: All
- Grid Geometry: All

**maxcuid/identity:** Verify that result of maxcuid operation is same across grid

- Code Locations: All
- Grid Geometry: All

**clock/monotonic:** Verify that result of clock operation increases monotonically

- Grid Geometry: OneGroupSet
- Code Location: All or Kernel

**waveid/lessmax:** Verify that result of waveid operation is always less than maxwaveid

- Code Locations: All
- Grid Geometry: All

**waveid/identity:** Verify that result of waveid operation is same across wavefront

- Code Locations: All
- Grid Geometry: All

**maxwaveid/identity:** Verify that result of maxwaveid operation is same across grid

- Code Locations: All
- Grid Geometry: All

**laneid/lessmax:** Verify that result of laneid operation is always less than wavesize

- Code Locations: All
- Grid Geometry: All

**laneid/sequence:** Verify that result of laneid operation corresponds to work-items assigned to lanes in work-item flattened absolute ID order

- Code Locations: All
- Grid Geometry: All

**Exceptions:** [PRM, chapter 12]

*Feature not implemented/not testable.*

**Directives:** [PRM, chapter 13]

- Grid Geometry: OneGroupSet
- Code Location: All or Kernel
- Exceptions mask: All (where applicable)

**Version:** [PRM, chapter 14]

*Tests to be implemented.*

**Libraries:** [PRM, chapter 15]

- Grid Geometry: OneGroupSet
- Code Location: All
- Segment: Variable
- Linkage: All

**Profiles:** [PRM, chapter 16]

*Configuration/tests to be implemented.*

**Limits:** [appendix A]

- Grid Geometry: All
- Code Location: Kernel



### **Memory Model: Memory Consistency Model [PSA, chapter 3]**

- Instructions: atomic, atomicnoret, ld, st
- Operand Kind: All
- Dst Type: All
- Atomic operation: All
- Segment: Atomic
- Memory Order: All
- Memory Scope: All
- Equiv: 0
- Grid Geometry: MModelSet
- Code Location: Kernel

### **Test suite internals**

*Information to be added when the suite is finalized.*