Vulkanised 2024

The 6th Vulkan Developer Conference
Sunnyvale, California | February 5-7, 2024

Vulkanised - AN(G|C)LE as an OpenCL Compute Driver

Pavan Lanka, Samsung Austin Research Center (SARC)
Co-presenter Austin Annestrand, SARC
Co-presenter Gowtham Tammana, SARC

- Acknowledgements
- Recap
- ANGLE Introduction
- Why ANGLE for OpenCL?
- Current Progress
- Next Steps

Acknowledgements

CLSPV team – Alan, Romaric

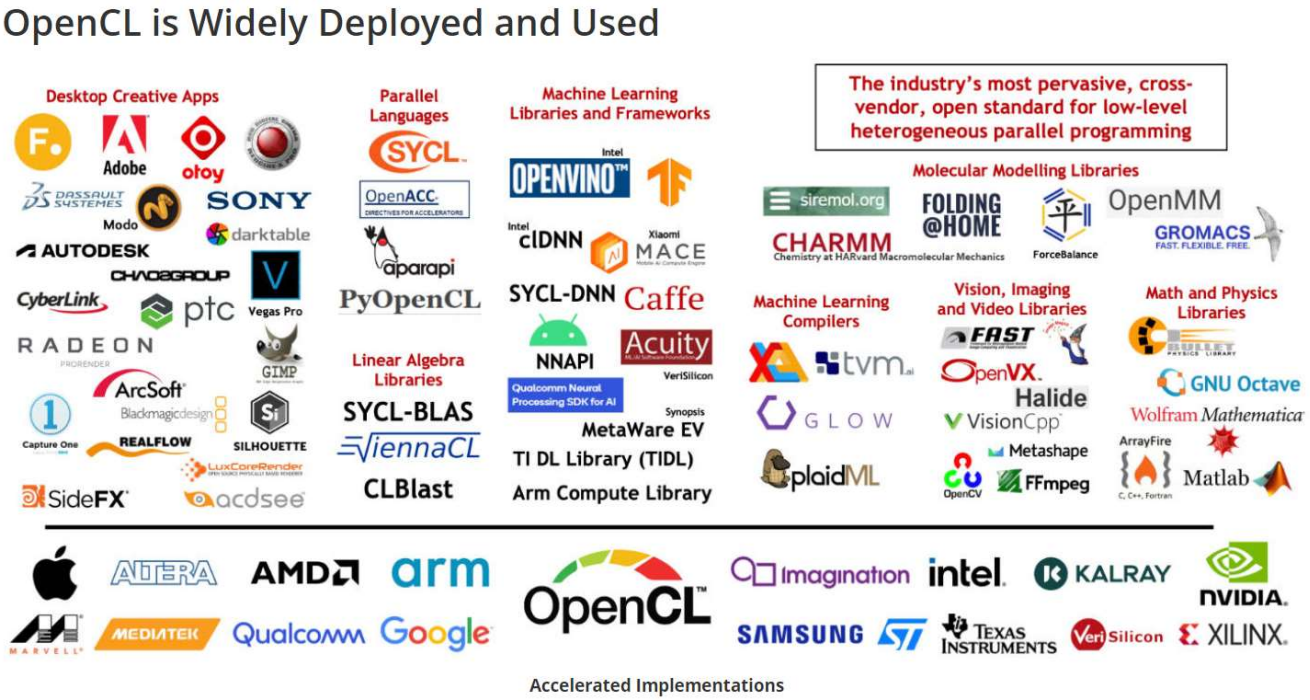
Samsung AN(G|C)LE team

Google ANGLE team

Background

- Previously at Vulkanised 2023…
- Our Story:
  - **Commitment to reduce the Total Cost of Ownership** for various APIs including OpenCL
- Vulkan as HAL

What is OpenCL?

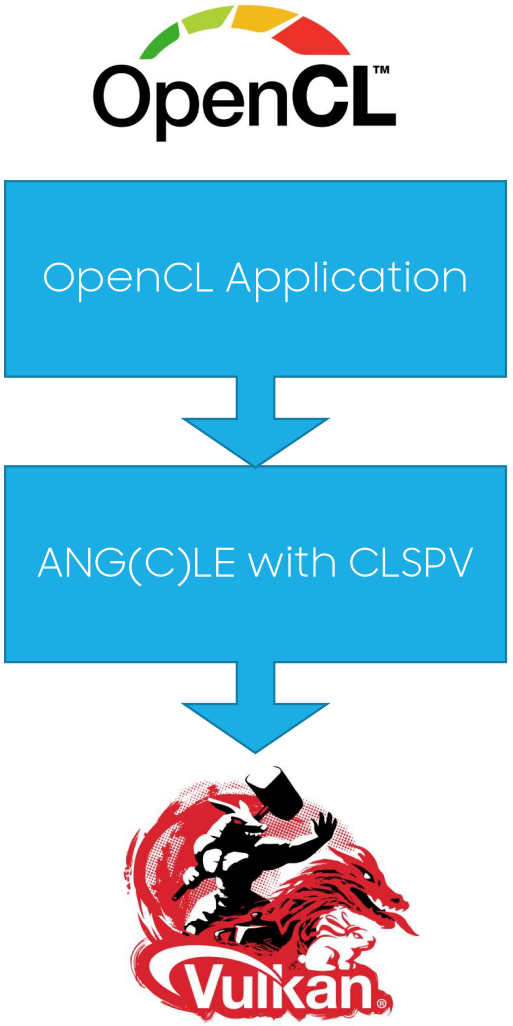


OpenCL is making a comeback thanks to ML

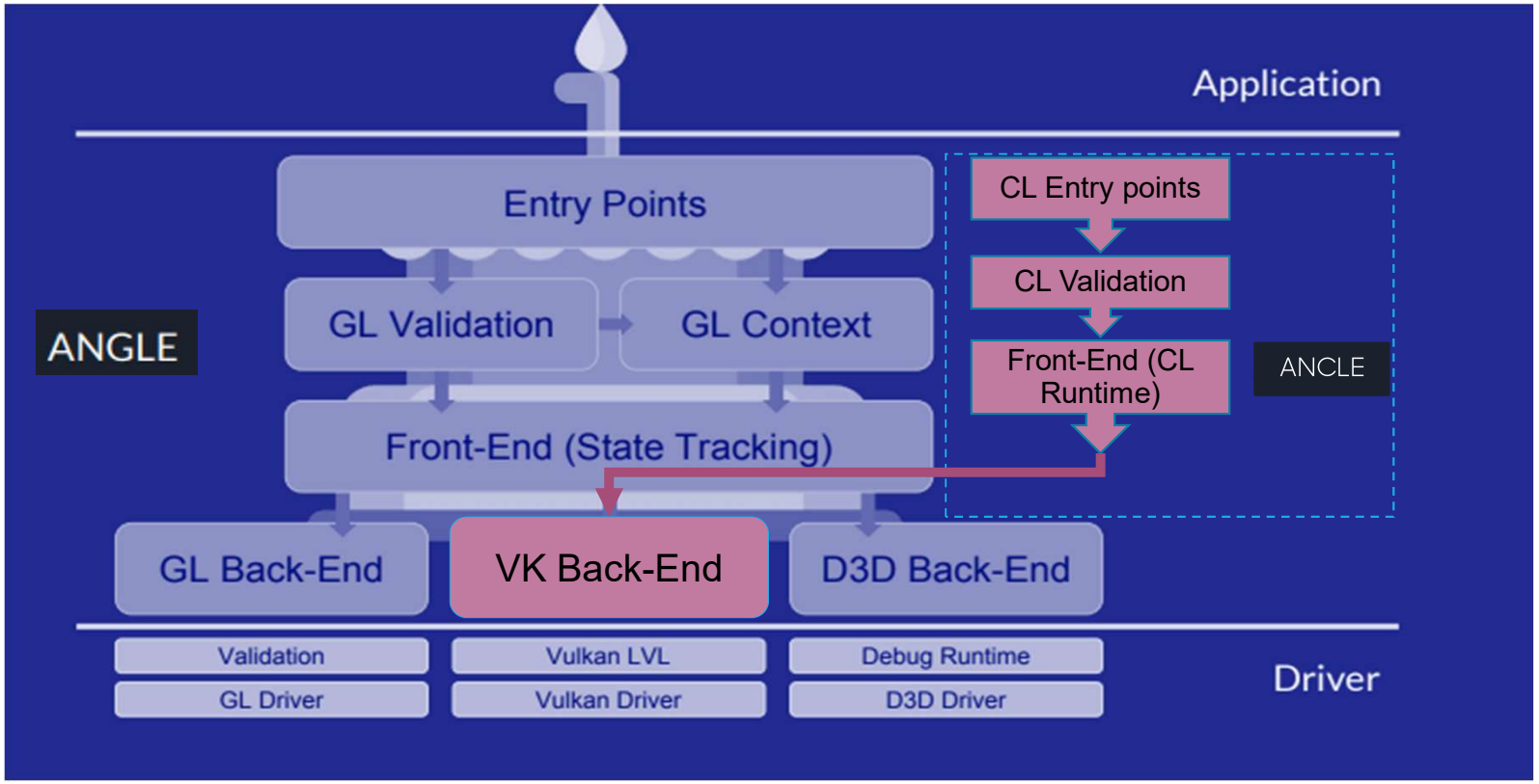
ANGLE Introduction

- Allow multiple flavors of OS to run OpenGL ES content seamlessly by translating to a native hardware-supported API.

- **NEW!**
  - Extend existing functionality to support Compute/OpenCL by **translating OpenCL to Vulkan**.



ANGLE Introduction Cont'd



Why add OpenCL to ANGLE?

OpenCL is a favored high-level (front-end) language!

- Easier to write than VK (SPIR-V)

No need for a native implementation stack

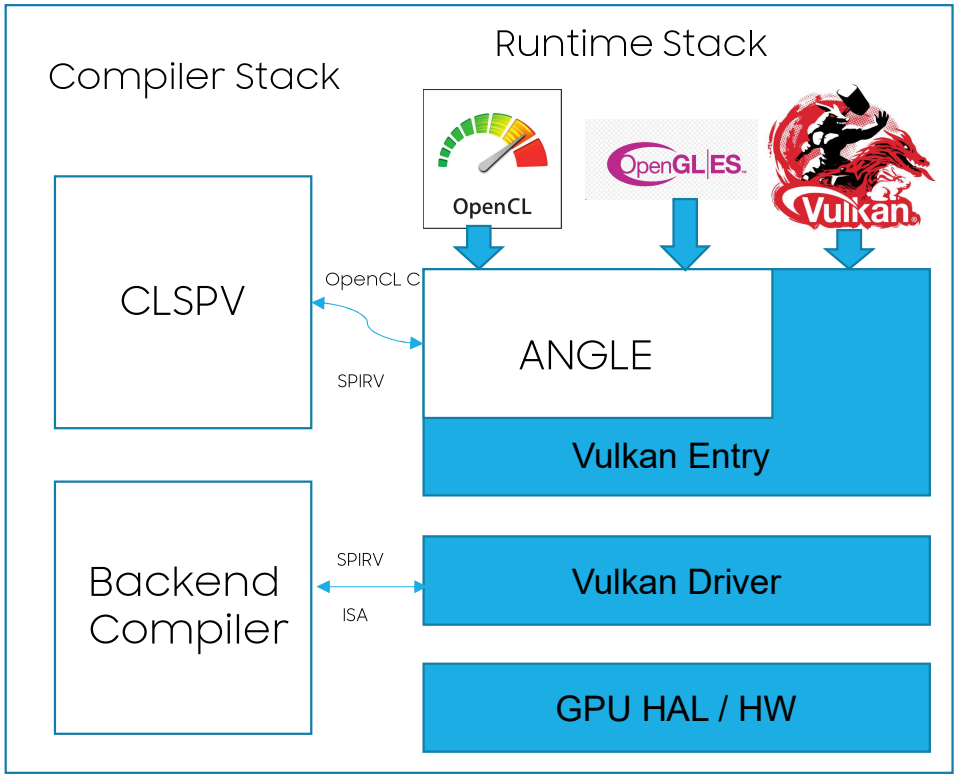
- Compute pipeline (less state, etc.), kernels compiled offline/optimized, low added overhead

Improved Ecosystem

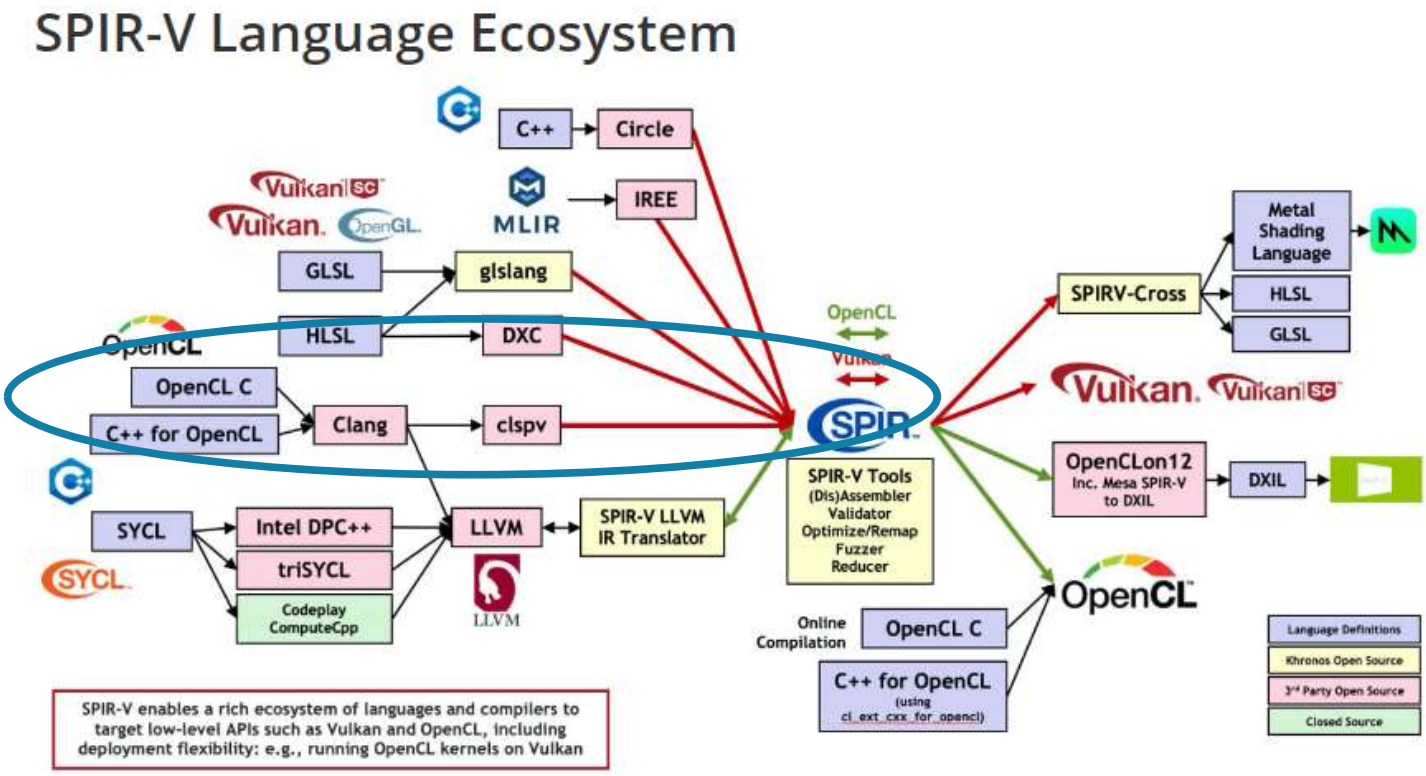
- Non-fragmented mobile driver story
- GL/CL Inter-op
- Flexible and Open Source

How are we doing this?

- Use open-source CLSPV compiler
  - (CL → SPIRV) translation
- Integrate CLSPV into ANGLE code base
- Implement CL Runtime in ANGLE
  - Leverage existing ANGLE infrastructure and SW Architecture



CLSPV / SPIR-V Language Ecosystem



The SPIR-V ecosystem includes a rich variety of language front-ends, tools and run-times

Mapping OpenCL to Vulkan

Almost a 1-1 mapping for most pieces

- e.g. Devices, Buffers, Images, Samplers, etc.

Pieces that are different still have similar API paradigms

- clCreateCommandQueue → VKCommandBuffer
- clCreateKernel → VKCreateComputePipeLines

Command processing

- clEnqueue* → vkCmd*

Current Progress

- Target: OpenCL 3.0 Full Profile
  - Not Immediately Supported: SVM, Device Enqueue

- CLSPV has been validated for OpenCL 3.0: ~95% CTS passing.

- Started ANCLE development efforts ~6 months ago, and we are already ~50% CTS passing OpenCL 3.0.

Expected Performance*

OpenCL Information	
Platform Vendor	Samsung Electronics Co., Ltd.
Platform Name	Samsung Mobile GPU Platform
Device Vendor	Samsung Electronics Co., Ltd.
Device Name	Samsung Xclipse 920
Board Name	Samsung Xclipse 920
Compute Units	3
Maximum Frequency	1306 MHz
Device Memory	4.00 GB

OpenCL Performance

OpenCL Score	8454
Background Blur	4242 (17.6 images/sec)
Face Detection	3139 (10.2 images/sec)
Horizon Detection	10624 (330.6 Mpixels/sec)
Edge Detection	12063 (447.5 Mpixels/sec)
Gaussian Blur	5962 (259.8 Mpixels/sec)
Feature Matching	2863 (112.9 Mpixels/sec)
Stereo Matching	31736 (30.2 Gpixels/sec)
Particle Physics	28215 (1241.8 FPS)

Vulkan Information	
Device Name	Samsung Xclipse 920

Vulkan Performance

Vulkan Score	10219
Background Blur	4657 (19.3 images/sec)
Face Detection	3029 (9.89 images/sec)
Horizon Detection	12709 (395.5 Mpixels/sec)
Edge Detection	12315 (456.6 Mpixels/sec)
Gaussian Blur	10053 (436.0 Mpixels/sec)
Feature Matching	3524 (138.9 Mpixels/sec)
Stereo Matching	41291 (39.3 Gpixels/sec)
Particle Physics	36828 (1620.8 FPS)

Next steps

- 100% Conformance
- Performance/Profiling!
- Productize for upcoming mobile phones!
- Benefit from the Open Source Community
    - Portable OpenCL via a Vulkan HAL reality

Make OpenCL a first-class citizen in Android by relying on Vulkan as its Native Driver

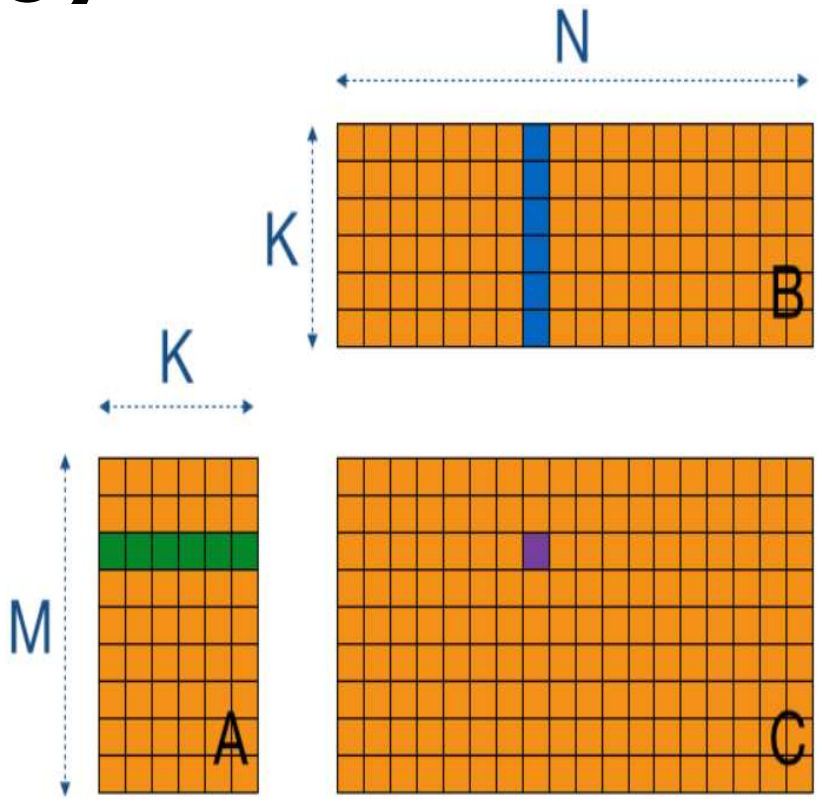
Backup

Q&A

Thank you for your time

Example (SGEMM -CPU)

```
for (int m=0; m<M; m++) {
        for (int n=0; n<N; n++) {
                float acc = 0.0f;
                for (int k=0; k<K; k++) {
                        acc += A[k*M + m]
* B[n*K + k];
                }
                C[n*M + m] = acc;
        }
}
```



Example (SGEMM -OpenCL)

```
kernel = clCreateKernel(program, "myGEMM1", &err)
err = clSetKernelArg(kernel, 0, sizeof(int), (void*)&M);
err = clSetKernelArg(kernel, 1, sizeof(int), (void*)&N);
err = clSetKernelArg(kernel, 2, sizeof(int), (void*)&K);
err = clSetKernelArg(kernel, 3, sizeof(cl_mem), (void*)&A);
err = clSetKernelArg(kernel, 4, sizeof(cl_mem), (void*)&B);
err = clSetKernelArg(kernel, 5, sizeof(cl_mem), (void*)&C);
const int TS = 32;
const size_t local[2] = { TS, TS };
const size_t global[2] = { M, N };
err = clEnqueueNDRangeKernel(queue, kernel, 2, NULL,
global, local, 0, NULL, &event);
err = clWaitForEvents(1, &event);
```

```
// First naive implementation
__kernel void myGEMM1(const int M, const int N, const int K,
                      const __global float* A,
                      const __global float* B,
                      __global float* C) {

  // Thread identifiers
  const int globalRow = get_global_id(0); // Row ID of C (0..M)
  const int globalCol = get_global_id(1); // Col ID of C (0..N)

  // Compute a single element (loop over K)
  float acc = 0.0f;
  for (int k=0; k<K; k++) {
    acc += A[k*M + globalRow] * B[globalCol*K + k];
  }

  // Store the result
C[globalCol*M + globalRow] = acc;
}
```

API Mappings

OpenCL API call	VK API Call
clCreateCommandQueue	vkCreateCommandPool
clCreateProgram/Kernel	vkCreateComputePipelines
clCreateBuffer	vkCreateBuffer + vkCreateDeviceMemory
clCreateImage	vkCreateImage + vkCreateDeviceMemory
clCreateSampler	vkCreateSampler
clEnqueue*	vkCmd*
clFinish	vkQueueWaitIdle
clGetProfilingInfo(START,END)	vkQueryPool*/vkCmdWriteTimeStamp

API translation Example

```
246  CL CreateBuffer: context = 0xb400007a441ba8b0, flags = 1, size = 4096, host_ptr = 0x0000000000000000, errcode_ret = 0x0000007fd33e5fd4
247  vkCreateBuffer
248  vkGetBufferMemoryRequirements
249  vkGetBufferMemoryRequirements
250  vkDestroyBuffer
251  vkCreateBuffer
252  vkGetBufferMemoryRequirements
253  vkGetBufferMemoryRequirements
254  vkAllocateMemory
255  vkBindBufferMemory
256  vkMapMemory
```

```
281  CL CreateKernel: program = 0xb400007a541ab330, kernel_name = 0x0000005906cb81cc, errcode_ret = 0x0000007fd33e5fd4
282  vkCreateDescriptorSetLayout
283  vkCreatePipelineLayout
284  vkCreateDescriptorPool
285  vkAllocateDescriptorSets
```

```
290  CL EnqueueNDRangeKernel: command_queue = 0xb400007a541aaf10, kernel = 0xb400007a541ad380, work_dim = 1, global_work_offset = 0x0000000000000000, global_work_size = 0x0000007fd33e5fe0, local_wor
291  vkUpdateDescriptorSets
292  vkCreatePipelineCache
293  vkDestroyPipelineCache
294  vkCreateShaderModule
295  vkCreateComputePipelines
```

API translation Example Contd..

```
307  CL Finish: command_queue = 0xb400007a541aaf10
308  vkBeginCommandBuffer
309  vkCmdBindDescriptorSets
310  vkCmdPushConstants
311  vkCmdBindPipeline
312  vkCmdDispatch
313  vkCmdPipelineBarrier
314  vkCmdCopyBuffer
315  vkEndCommandBuffer
316  vkCreateFence
317  vkQueueSubmit
318  vkGetFenceStatus
319  vkGetFenceStatus
320  vkWaitForFences
321  vkGetFenceStatus
322  vkUnmapMemory
323  vkDestroyBuffer
324  vkResetCommandBuffer
325  vkFreeMemory
```

Kernel CL → SPIRV Binding

```
__kernel void math_kernel16( __global float16* out,
__global float16* in )
{
  size_t i = get_global_id(0);
  out[i] = acosh( in[i] );
}
```

http://htmlpreview.github.io/?https://github.com/KhronosGroup/SPIRV-Registry/blob/master/nonsemantic/NonSemantic.ClspvReflection.html

```
%2174 = OpExtInstImport "NonSemantic.ClspvReflection.2"
..
OpEntryPoint GLCompute %545 "math_kernel16" %gl_GlobalInvocationID
%2175 = OpString "math_kernel16"
 %2177 = OpString "out"
 %2180 = OpString "in"
....
    %uint_0 = OpConstant %uint 0
%_ptr_StorageBuffer_float = OpTypePointer StorageBuffer %float
    %uint_1 = OpConstant %uint 1
    %uint_2 = OpConstant %uint 2
    %uint_3 = OpConstant %uint 3
// Kernel Argument Info (GetKernelArgInfo CL API)

    %2176 = OpExtInst %void %2174 Kernel %545 %2175
     %2178 = OpExtInst %void %2174 ArgumentInfo %2177
     %2179 = OpExtInst %void %2174 ArgumentStorageBuffer %2176 %uint_0 %uint_0 %uint_0 %2178
     %2181 = OpExtInst %void %2174 ArgumentInfo %2180
     %2182 = OpExtInst %void %2174 ArgumentStorageBuffer %2176 %uint_1 %uint_0 %uint_1 %2181
     %2183 = OpExtInst %void %2174 SpecConstantWorkgroupSize %uint_0 %uint_1 %uint_2
```