Discussion

- In your pair, detail all the steps required to build an OpenCL program that completes Matrix-Vector Multiplication
  - Assume the Matrix is MxN
  - Assume the Vector has N elements
  - Call the kernel vecmult

Example Design:

1. Write Kernel

Write Kernel

```
__kernel void vecmult(__global int* matrix, __global int* vector,
                      __global int* result, int m, int n)
{
    int i= get_global_id(0);

    //Complete Matrix-Vector Multiplication
    //How do you break this up for parallel?

}
```

Write Kernel

```
__kernel void vecmult(__global int* matrix, __global int* vector,
                                        __global int* result, int m, int n)
{
    int i= get_global_id(0);

    //Complete Matrix-Vector Multiplication
    //How do you break this up for parallel?
    //Loop over your row of the matrix to complete the  //
multiplication and receive your singular result
    …
    result[i] = myMultResult;
}
```

Example Design:

1. Write Kernel
2. **Platform Layer (copy paste to set up device)**

Example Design:

1. Write Kernel
2. Platform Layer (copy paste to set up device)
3. **Create then build program**

Create and Build Program

```
cl_program program = clCreateProgramWithBinary(context,
                        num_device,device_list,length,
                        const unsigned char**) binaries,
                        status,&error);
```

Create and Build Program

```
cl_program program = clCreateProgramWithBinary(context,
                         num_device,device_list,length,
                         const unsigned char**) binaries,
                         status,&error);

clError = clBuildProgram(program, 1, &device,
                                      compilerOptions,
  NULL, NULL);
```

Example Design:

1. Write Kernel
2. Platform Layer (copy paste to set up device)
3. Create then build program
4. **Create kernel from the program**

Create Kernel

```
cl_kernel kernel = clCreateKernel(program, ”vecmult”, &err);
```

Example Design:

1. Platform Layer (copy paste to set up device)
2. Create then build program
3. Create kernel from the program
4. **Allocate and transfer buffers on/to device**

Allocate and Transfer Buffers

```
//Assume host memory is already created in hostmat and
hostvec
cl_mem matrix = clCreateBuffer(context, CL_MEM_READ_WRITE,
                                m*n*sizeof(int), NULL, &err);
err = clEnqueueWriteBuffer(queue, matrix, CL_TRUE, 0,
                                 m*n*sizeof(int), hostmat, 0, NULL,
                          &writeEventMat;
```

Allocate and Transfer Buffers

```
//Assume host memory is already created in hostmat and hostvec
cl_mem matrix = clCreateBuffer(context, CL_MEM_READ_WRITE,
                               m*n*sizeof(int), NULL, &err);
err = clEnqueueWriteBuffer(queue, matrix, CL_TRUE, 0,
                                                  m*n*sizeof(int), hostmat,
0, NULL,                                       &writeEventMat;
cl_mem vector = clCreateBuffer(context, CL_MEM_READ_WRITE,
                                    n*sizeof(int), NULL, &err);
err = clEnqueueWriteBuffer(queue, vector, CL_TRUE, 0, n*sizeof(int),
                                   hostvec, 0, NULL, &writeEventVec);
```

Allocate and Transfer Buffers

```
//Assume host memory is already created in hostmat and hostvec
cl_mem matrix = clCreateBuffer(context, CL_MEM_READ_WRITE,
                               m*n*sizeof(int), NULL, &err);
err = clEnqueueWriteBuffer(queue, matrix, CL_TRUE, 0,
                                                   m*n*sizeof(int), hostmat,
0, NULL,                                        &writeEventMat;
cl_mem vector = clCreateBuffer(context, CL_MEM_READ_WRITE,
                                    n*sizeof(int), NULL, &err);
err = clEnqueueWriteBuffer(queue, vector, CL_TRUE, 0, n*sizeof(int),
                                   hostvec, 0, NULL, &writeEventVec);
cl_mem result =clCreateBuffer(context, CL_MEM_READ_WRITE,
                                           n*sizeof(int), NULL, &err);
```

Example Design:

1. Write Kernel
2. Platform Layer (copy paste to set up device)
3. Create then build program
4. Create kernel from the program
5. Allocate and transfer buffers on/to device
6. **Set up the kernel argument list**

Set Up Arguments

```
clError = clSetKernelArg(kernel, 0, sizeof(cl_mem),
                    (void *)&matrix);
clError = clSetKernelArg(kernel, 1, sizeof(cl_mem),
                    (void *)&vector);
clError = clSetKernelArg(kernel, 2, sizeof(cl_mem),
                    (void *)&result);
clError = clSetKernelArg(kernel, 3, sizeof(int), &m);

clError = clSetKernelArg(kernel, 4, sizeof(int), &n);
```

Example Design:

1. Write Kernel
2. Platform Layer (copy paste to set up device)
3. Create then build program
4. Create kernel from the program
5. Allocate and transfer buffers on/to device
6. Set up the kernel argument list
7. **Launch the kernel**

Launch Kernel

```
//globalWorkSize = ?
clEnqueueNDRangeKernel(queue, kernel, 1, NULL,
                       &globalWorkSize, NULL, 1,
                       &writeEventVec, &kernelEvent);
```

Launch Kernel

```
globalWorkSize = m;
clEnqueueNDRangeKernel(queue, kernel, 1, NULL,
                       &globalWorkSize, NULL, 1,
                       &writeEventVec, &kernelEvent);
```

Example Design:

1. Write Kernel
2. Platform Layer (copy paste to set up device)
3. Create then build program
4. Create kernel from the program
5. Allocate and transfer buffers on/to device
6. Set up the kernel argument list
7. Launch the kernel
8. **Transfer result buffer back**

Transfer Result

```
clError = clEnqueueReadBuffer(queue, result, CL_TRUE, 0,
                    m*sizeof(int), hostresult, 1, &kernelEvent,
                    NULL);
```

Discussion

- Which language would you use to solve the following problems, and why?
  - Vector Addition
  - Image Thresholding
  - State Machine Design
  - Audio Filtering