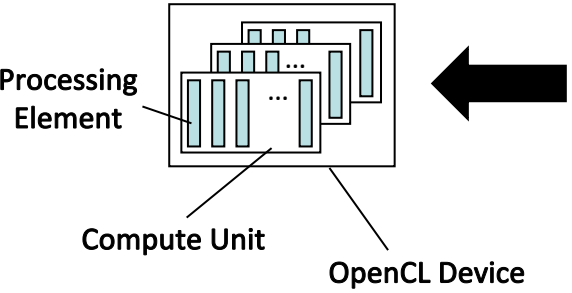
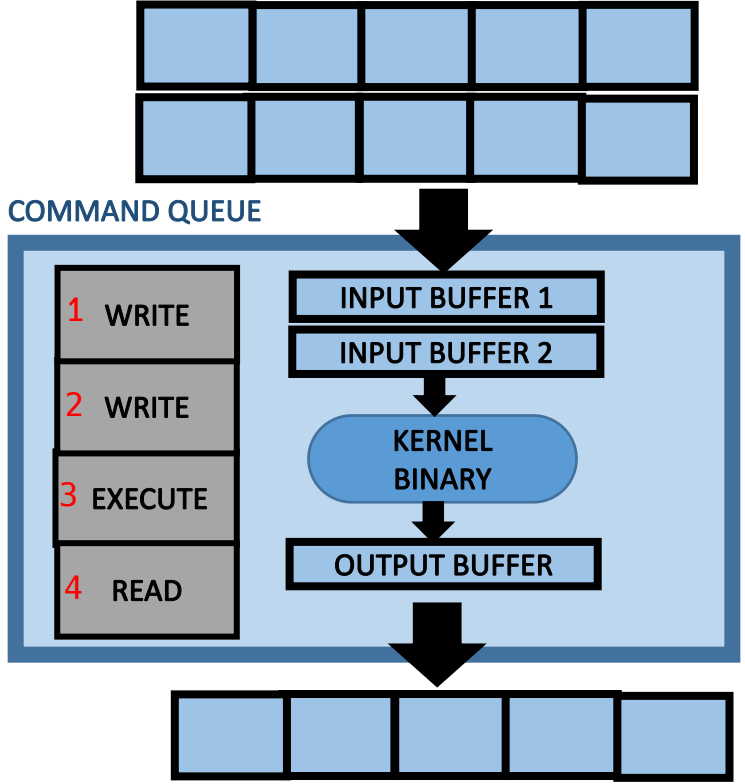
```
__kernel void vadd( __global float* input1,
__global float* input2, __global float* output,
int n)
{
  int i = get_global_id(0);
  if(i<n)
    output[i] = input1[i]+input2[i];
}
```






```
1 d_A = clCreateBuffer(clContext,CL_MEM_READ_WRITE,size,NULL,&clStatus);
2  d_B = clCreateBuffer(clContext,CL_MEM_READ_WRITE,size,NULL,&clStatus);
3  d_C = clCreateBuffer(clContext,CL_MEM_READ_WRITE,size,NULL,&clStatus);
4  /*H2D Copy */
5  clEnqueueWriteBuffer(clCommandQueue,d_A,CL_FALSE,0,size,h_A,0,NULL,&ev1);
6  clEnqueueWriteBuffer(clCommandQueue,d_B,CL_FALSE,0,size,h_B,0,&ev1,&ev2);
7  /* Setting Kernel Parameters */
8  int tx = 32;
9  size_t block[3] = {tx,1,1};
10  size_t grid[3] = {num_elements,1,1};
11  clSetKernelArg(clKernel,0,sizeof(cl_mem),(void*)&d_A);
12  clSetKernelArg(clKernel,1,sizeof(cl_mem),(void*)&d_B);
13  clSetKernelArg(clKernel,2,sizeof(cl_mem),(void*)&d_C);
14  clSetKernelArg(clKernel,4,sizeof(int),(void*)&num_elements);
15  /*Work item Processing*/
16 clEnqueueNDRangeKernel(clCommandQueue,clKernel,2,NULL,grid,block,0,&ev2,&ev3);
17  /*D2H Copy*/
18  clEnqueueReadBuffer(clCommandQueue,d_C,CL_TRUE,0,size,h_C,0,&ev3,&ev4);
19  clWaitForEvents(1,&ev4)
```