Project: Performance on GPU's

- Log in in one of the Computer Science Department computers with GPUs (follow **handout instructions**) and determine the properties of the GPU in the computer you access.

- Consider the dot product between two vectors using the CUDA code for the kernel presented **in the notes.**

- Write a main code for multiplying two vectors. This can been accomplished by using the global_void_dot kernel **in the notes**

  1. Define N= 33*1024, threadsPerBlock=256, blocksPerGrid= min(32, (N+threadsPerBlock-1)/threadsPerBlock)
  2. define the kernel
  3. start the main code:
     (a) Define a, b, c , partial_c (c is a scalar)
     (b) Define dev_a, dev_b, dev_partial_c‘
     (c) allocate memory on the CPU side for the 3 vectors in (a)
     (d) use cudaMalloc for allocating memory on the device for the three vectors in b
     (e) Initialize vectors a and b
     (f) use cudaMemcpy for copying a, b to dev_a and dev_b respectively.
     (g) Launch the Kernel
     (h) use cudaMemcpy to copy dev_partial_c into partial_c
     (i) add the values of vector partial_c into c and print its value.

- **To do later:** compare the performances of the CPU code with the CUDA code. The comparisons should be in terms of speed up and efficiency. To measure the performance of GPU codes we use a CUDA event API such as shown below:

  ```
  cudaEvent_t start, stop;
  cudaEventCreate(&start);
  cudaEventCreate(&stop);
  cudaEventRecord(start, 0);
  //
  ```

```
// ... do some work on GPU ...
//
cudaEventRecord(stop, 0);
cudaEventSynchronize(stop);
float elapsedTime;
cudaEventElapsedTime(&elapsedTime, start, stop);
printf("elapsed time = %g msec\n", elapsedTime);
```

After using the timers, you can destroy them:

```
cudaEventDestroy(start);
cudaEventDestroy(stop);
```