Chapter Exercises Solution Manual

Programming Massively Parallel Processors: A Hands-on Approach
By Wen-mei W. Hwu, David B. Kirk, and Izzat El Hajj

Chapter 1: Introduction
No exercises.

Chapter 2: Heterogeneous data parallel computing

1. (C) i=blockIdx.x*blockDim.x + threadIdx.x;
2. (C) i=(blockIdx.x*blockDim.x + threadIdx.x)*2;
3. (D) i=blockIdx.x*blockDim.x*2 + threadIdx.x;
4. (C) 8192
5. (D) v * sizeof(int)
6. (D) (void **) &A_d
7. (C) cudaMemcpy(A_d, A_h, 3000, cudaMemcpyHostToDevice);
8. (C) cudaError_t err;
9.
    a. 128
    b. 200,064
    c. 1,563
    d. 200,064
    e. 200,000
10. The intern should use both the "__host__" keyword and the "__device__" keyword in declaring the functions that need to be executed on both the host and the device. The CUDA compiler will generate both versions of the functions.

Chapter 3: Multidimensional grids and data

1. *Not included*
2.

```
__global__ void matrixVectorMulKernel(float *A, float *B, float *C, int vectorLen) {
  int i = threadIdx.x + blockIdx.x*blockDim.x;
  float sum = 0.0f;
  if (i < vectorLen) {
    for (int j = 0; j < vectorLen; j++)
      sum += B[i*vectorLen + j] * C[j];
    A[i] = sum;
  }
}

void matrixVectorMul(float *h_A, float *h_B, float *h_C, int vectorLen) {
  float *d_A, *d_B, *d_C;
  int matrixSize = vectorLen*vectorLen*sizeof(float);
  int vectorSize = vectorLen*sizeof(float);
  cudaMalloc((void**)&d_A, vectorSize);
  cudaMalloc((void**)&d_B, matrixSize);
  cudaMalloc((void**)&d_C, vectorSize);

  // Initialize GPU memory contents with 0 or host data
  cudaMemset(d_A, 0, vectorSize);
  cudaMemcpy(d_B, h_B, matrixSize, cudaMemcpyHostToDevice);
  cudaMemcpy(d_C, h_C, vectorSize, cudaMemcpyHostToDevice);
```