Vector Addition Example

Vector Addition Example

```
__global__void VecAdd(float* A, float* B,
        float* C, int N) {
  int i = blockDim.x * blockIdx.x + threadI
dx.x;
  if (i < N)
    C[i] = A[i] + B[i];
}

int main() {
  …
  float*   h_A   =   (float*)malloc(size);
  float*   h_B   =   (float*)malloc(size);
  float*    h_C   =   (float*)malloc(size);
```

```
float* d_A;
cudaMalloc(&d_A, size);
float* d_B;
cudaMalloc(&d_B, size);
float* d_C;
cudaMalloc(&d_C, size);

// Copy vectors from host memory to
// device memory
cudaMemcpy(d_A, h_A, size,
           cudaMemcpyHostToDevice);
cudaMemcpy(d_B, h_B, size,
           cudaMemcpyHostToDevice);
```

Vector Addition Example

```
  // Invoke kernel
  int threadsPerBlock = 256;
  int blocksPerGrid = N/threadsPerBlock;
  VecAdd<<<blocksPerGrid, threadsPerBlock>>>(d_A, d_B, d_C, N);
```

```
// Copy result from device memory to
// host memory  cudaMemcpy(h_C,
  d_C, size,
                cudaMemcpyDeviceToHost);
  …
  cudaFree(d_A);
  cudaFree(d_B);
  cudaFree(d_C);
…
}
```

Typical CUDA Program Flow

1. Load data into CPU memory
   - `fread/rand`
2. Copy data from CPU to GPU memory
   - `cudaMemcpy(..., cudaMemcpyHostToDevice)`
3. Call GPU kernel
   - `yourkernel<<<x, y>>>(...)`
4. Copy results from GPU to CPU memory.
   - `cudaMemcpy(..., cudaMemcpyDeviceToHost)`
5. Use results on CPU

CUDA Extensions for C/C++

- Kernel launch
  - Calling functions on GPU
- Memory management
  - GPU memory allocation, copying data to/from GPU
- Declaration qualifiers
  - `__device_ , _ shared,_ local, _ global_ ,_ host___`
- Special instructions
  - Barriers, fences, etc.
- Keywords
  - `threadIdx, blockIdx, blockDim`