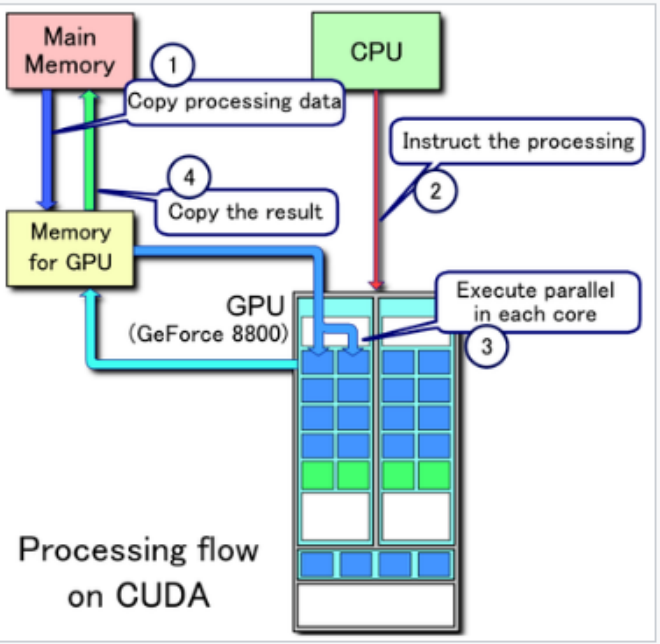
GPU Software – CUDA



Example of CUDA processing flow

1. Copy data from main memory to GPU memory
2. CPU initiates the GPU compute kernel
3. GPU's CUDA cores execute the kernel in parallel
4. Copy the resulting data from GPU memory to main memory

```cpp
#include<stdio.h>
#include<stdlib.h>

// CUDA kernel: compute vector sum z_d = x_d + y_d using a grid of threads on GPU
__global__ void vecAddKernel(float *x_d, float *y_d, float *z_d, unsigned int n){
    unsigned int i = blockDim.x * blockIdx.x + threadIdx.x;
    if(i < n){
        z_d[i] = x_d[i]+y_d[i];
    }
}

int main(int argc, char** argv){
    unsigned int n = 1024;

    // Allocate host memory for arrays x_h, y_h, and z_h; and initialize arrays x_h and y_h.
    float* x_h = (float*) malloc(sizeof(float)*n);
    for(unsigned int i=0; i<n; i++) x_h[i] = (float)rand()/(float)(RAND_MAX);
    float* y_h = (float*) malloc(sizeof(float)*n);
    for(unsigned int i=0; i<n; i++) y_h[i] = (float)rand()/(float)(RAND_MAX);
    float* z_h = (float*) calloc(n, sizeof(float));

    // (1) Allocate device memory for arrays x_d, y_d, and z_d.
    float *x_d, *y_d, *z_d;
    cudaMalloc((void**)&x_d, sizeof(float)*n);
    cudaMalloc((void**)&y_d, sizeof(float)*n);
    cudaMalloc((void**)&z_d, sizeof(float)*n);

    // (2) Copy arrays x_h and z_h to device memory x_d and y_d, respectively.
    cudaMemcpy(x_d, x_h, sizeof(float)*n, cudaMemcpyHostToDevice);
    cudaMemcpy(y_d, y_h, sizeof(float)*n, cudaMemcpyHostToDevice);

    // (3) Call kernel to launch a grid of threads to perform the vector addition on GPU.
    vecAddKernel<<<ceil(n/256.0), 256>>>(x_d,y_d,z_d,n);

    // (4) Copy the result data from the device memory of array z_d to the host memory of array z_h.
    cudaMemcpy(z_h, z_d, sizeof(float)*n, cudaMemcpyDeviceToHost);

    // (5) Free device memory of arrays x_d, y_d, and z_d
    cudaFree(x_d);
    cudaFree(y_d);
    cudaFree(z_d);

    // Free host memory of arrays x_h, y_h, and z_h
    free(x_h);
    free(y_h);
    free(z_h);

    return 0;
}
```

Define a CUDA Kernel Function

Call the CUDA Kernel