[MI3.04a] Advanced Programming for HPC

Threads

Labwork 4

HUYNH Vinh Nam

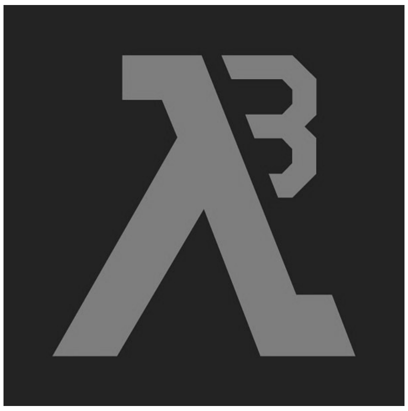
M19.ICT.007

November 2020

Original Input

(a) Original Image

Output



(b) Output

Implementation

```
__global__ void grayscale_2d(uchar3 *input, uchar3 *output, int img_width, int img_height) {
    int col = threadIdx.x + blockIdx.x * blockDim.x;
    int row = threadIdx.y + blockIdx.y * blockDim.y;
    if (col >= img_width || row >= img_height) return;
    int tid = row * img_width + col;
    output[tid].x = (unsigned char)(((int)input[tid].x + (int)input[tid].y
                                + (int)input[tid].z) / 3);
    output[tid].z = output[tid].y = output[tid].x;
}

void Labwork::labwork4_GPU() {
    // Calculate number of pixels
    int pixelCount = inputImage->width * inputImage->height;
    char *hostInput = inputImage->buffer;
    outputImage = static_cast<char *>(malloc(pixelCount * 3));

    // Allocate CUDA memory
    uchar3 *devInput;
    uchar3 *devOutput;
    cudaMalloc(&devInput, pixelCount * 3);
    cudaMalloc(&devOutput, pixelCount * 3);

    // Copy CUDA Memory from CPU to GPU
    cudaMemcpy(devInput, hostInput, pixelCount * 3, cudaMemcpyHostToDevice);

    // Processing
    dim3 blockSize = dim3(32,32);
    dim3 gridSize = dim3((int)((inputImage->width + blockSize.x - 1) / blockSize.x),
                     (int)((inputImage->height + blockSize.y - 1)/ blockSize.y));
    grayscale_2d<<<gridSize, blockSize>>>(devInput, devOutput,
                                      inputImage->width, inputImage->height);

    // Copy CUDA Memory from GPU to CPU
    cudaMemcpy(outputImage, devOutput, pixelCount * 3, cudaMemcpyDeviceToHost);
    // Cleaning
    free(hostInput);
    cudaFree(devInput);
    cudaFree(devOutput);
}
```