Question 1
For a 256x256 image with 1000 iterations, it took 434.627 ms on a T4 GPU and 74.5637 ms on a CPU. ~5.82 times slower.
The CPU is significantly faster since it operates at a higher clock speed (3-4 Ghz) than the GPU (1 Ghz). The CPU also benefits from ILP (since its superscalar), which is abundant in the Mandelbrot code. GPU cores, on the other hand, aren't superscalar.

Question 2
Cy is initialized in the same way as in scalar code.
To initialize cx, I first shift [0, 1, … 15] by the current j value (which is a multiple of 16).
Then I multiply by scale = 2.5/float(img_size) and add scale = -2.0.

To handle differences in loop iterations between pixels in the same vector, I used a mask vector.
Using _mm512_cmp_ps_mask, the mask bit in the mask vector is set to 1 if the condition x2 + y2 ≤ 4 is satisfied and set to 0 otherwise.
Then, using _mm512_mask_add_epi32, the ith value in the iters vector is incremented if the corresponding mask bit is a 1 and kept at the same value if the mask bit is a 0.

Performance:
CPU_scalar: 69.3092 ms
CPU_vector: 4.33957 ms

The CPU vector version is 15.97x faster.

Question 3
Performance:
GPU_scalar: 431.453 ms
GPU_vector: 20.0079 ms

The GPU vector version is 21.56x faster.

The GPU, internally does some masking so that if a threads value doesn't change if another thread is running for more iterations. Unlike the CPU version, the masking is invisible to the user.