Q5

We use OpenACC to run Pi series computing on GPU. The reason we use OpenACC instead of CUDA is that OpenACC code is easier to write. We use both single-precision and double-precision values to calculate Pi. The solution we use is that we calculate each item of the series using GPU parallelly and then sum up all the items together to calculate the Pi.

```
    start = CLOCK();

#pragma acc kernels
    for (i = 0; i < iteration; i++) {
        int s = 1;
        if (i % 2 == 1) s = -s;

        h_a[i] = (float) s * 4 / (i * 2 + 1);
    }

    float sum = 0;
    for (i = 0; i < iteration; i++)
        sum += h_a[i];

    finish = CLOCK();
```

Here is the result.

Iteration	Single-precision	Double-precision
16	3.079154	3.079153
32	3.110350	3.110350
64	3.125968	3.125969
128	3.133780	3.133780

256	3.137687	3.137686
512	3.139640	3.139640
1024	3.140616	3.140616
2048	3.141102	3.141104
4096	3.141352	3.141349
8192	3.141475	3.141471

```
[zhang.yam@c2169 Question5]$ ./Q5 16
Iteration: 16; Pi: 3.079153
Time for the loop = 2128.75 milliseconds
[zhang.yam@c2169 Question5]$ ./Q5 32
Iteration: 32; Pi: 3.110350
Time for the loop = 2123.68 milliseconds
[zhang.yam@c2169 Question5]$ ./Q5 64
Iteration: 64; Pi: 3.125969
Time for the loop = 2128.66 milliseconds
[zhang.yam@c2169 Question5]$ ./Q5 128
Iteration: 128; Pi: 3.133780
Time for the loop = 2128.35 milliseconds
[zhang.yam@c2169 Question5]$ ./Q5 256
Iteration: 256; Pi: 3.137686
Time for the loop = 2135.27 milliseconds
[zhang.yam@c2169 Question5]$ ./Q5 512
Iteration: 512; Pi: 3.139640
Time for the loop = 2128.83 milliseconds
[zhang.yam@c2169 Question5]$ ./Q5 1024
Iteration: 1024; Pi: 3.140616
Time for the loop = 2130.91 milliseconds
[zhang.yam@c2169 Question5]$ ./Q5 2048
Iteration: 2048; Pi: 3.141104
Time for the loop = 2130.74 milliseconds
[zhang.yam@c2169 Question5]$ ./Q5 4096
Iteration: 4096; Pi: 3.141349
Time for the loop = 2138.24 milliseconds
[zhang.yam@c2169 Question5]$ ./Q5 8192
Iteration: 8192; Pi: 3.141471
Time for the loop = 2126.32 milliseconds
```

```
[zhang.yam@c2169 Question5]$ ./Q5 16
Iteration: 16; Pi: 3.079154
Time for the loop = 2144.86 milliseconds
[zhang.yam@c2169 Question5]$ ./Q5 32
Iteration: 32; Pi: 3.110350
Time for the loop = 2129.67 milliseconds
[zhang.yam@c2169 Question5]$ ./Q5 64
Iteration: 64; Pi: 3.125968
Time for the loop = 2125.20 milliseconds
[zhang.yam@c2169 Question5]$ ./Q5 128
Iteration: 128; Pi: 3.133780
Time for the loop = 2127.46 milliseconds
[zhang.yam@c2169 Question5]$ ./Q5 256
Iteration: 256; Pi: 3.137687
Time for the loop = 2139.65 milliseconds
[zhang.yam@c2169 Question5]$ ./Q5 512
Iteration: 512; Pi: 3.139640
Time for the loop = 2121.19 milliseconds
[zhang.yam@c2169 Question5]$ ./Q5 1024
Iteration: 1024; Pi: 3.140616
Time for the loop = 2130.39 milliseconds
[zhang.yam@c2169 Question5]$ ./Q5 2048
Iteration: 2048; Pi: 3.141102
Time for the loop = 2132.09 milliseconds
[zhang.yam@c2169 Question5]$ ./Q5 4096
Iteration: 4096; Pi: 3.141352
Time for the loop = 2125.02 milliseconds
[zhang.yam@c2169 Question5]$ ./Q5 8192
Iteration: 8192; Pi: 3.141475
Time for the loop = 2132.51 milliseconds
```

We could tell that the single-precision data even converges faster than double-precision but the running time is stable when we even increase the iteration thanks to the parallelism of the GPU.