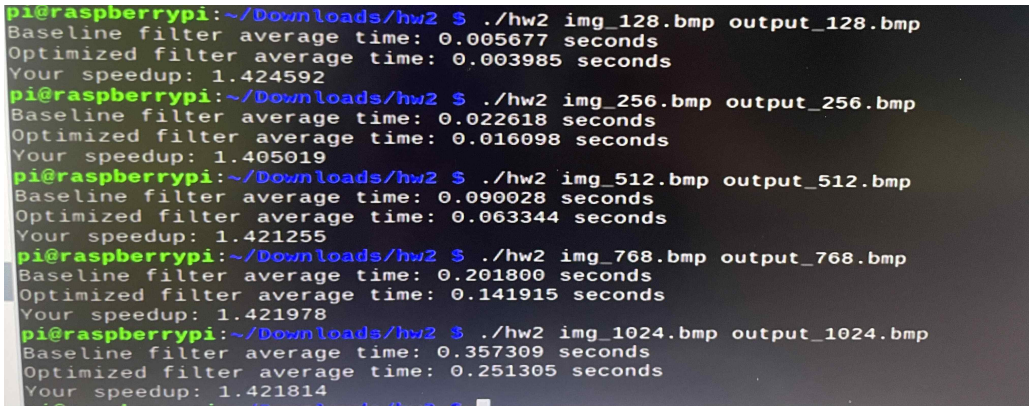
Assignment 2: Efficient Program Implementation

202111040 김지현

1. Implementation Results

The provided baseline and optimized convolution implementations were tested on various image sizes to measure performance improvements. Below are the results showing the execution times and speedups hieved.

	img_128	img_256	img_512	img_768	img_1024
speedup	1.424592	1.405019	1.421255	1.421978	1.421814



2. Optimization Approach

The optimization of the convolution filter focused on improving the computational efficiency of the baseline implementation. The primary strategies employed include loop unrolling and boundary checks optimization.

2-1 Loop Unrolling

- Strategy: The baseline code used nested loops to iterate over the 3x3 filter, which introduced significant loop overhead. By manually unrolling the loop, we reduced the number of iterations and the associated overhead.

- Implementation: In the convolution function, we replaced the nested loops with direct assignments. This approach allowed for fewer conditional checks and a reduction in loop control instructions, thereby speeding up the execution.

2-2 Boundary Checks Optimization

- Strategy: Boundary checks within the nested loops added conditional branches, which can degrade performance due to branch prediction penalties and increased instruction count.

- Implementation: By precomputing the boundary conditions outside the innermost loops, we reduced the number of conditional checks inside the loops. This strategy ensures that only necessary boundary checks are performed, minimizing the performance impact.

- The combination of boundary checks optimization with loop unrolling resulted in a significant reduction in execution time. The effect was particularly noticeable for larger images, where the overhead of boundary checks is more pronounced.