System programing hw2 202011230 홍서형

Introduction

This report details the optimizations implemented in the `filter_optimized()` function for the system programming homework assignment. The goal was to enhance the performance of the baseline convolution filter by leveraging various optimization techniques learned throughout the course. Key areas of focus included memory access patterns, cache utilization, and instruction efficiency. The optimized function successfully produces the same output as the baseline while achieving better performance metrics.

Optimizations Implemented

1. Function Inlining:
    - Description: Eliminated function calls within the convolution process by incorporating the functionality directly into the loop.
    - Benefit: Reduces overhead associated with function calls such as stack manipulation and jump instructions, resulting in fewer executed instructions and improved instruction cache usage.
2. Eliminating Unnecessary `malloc`:
    - Description: Removed any dynamic memory allocation (`malloc`), which is not used in this implementation.
    - Benefit: Reduces overhead associated with memory allocation and deallocation, leading to faster execution times.
3. Sharing Common Subexpressions:
    - Description: Calculated common expressions once and reused the results instead of recalculating them multiple times.
    - Benefit: Minimizes redundant calculations, reducing the overall number of executed instructions.
4. Eliminating Unused Headers:
    - Description: Removed unnecessary header files to streamline the code.
    - Benefit: Simplifies the codebase and may improve compilation time and binary size.
5. Cache-Friendly Memory Access:
    - Description: Optimized the order of loops and memory access patterns to enhance spatial locality and cache usage.
    - Benefit: Improves cache hit rates by accessing contiguous memory locations, reducing cache misses and memory latency.
6. Loop Unrolling:
    - Description: Manually unrolled loops to decrease the overhead of loop control instructions.

- **Benefit:** Reduces the number of loop control instructions and increases instruction-level parallelism, potentially leading to better performance.