UOK - Apr 2025 - CTEC41022: Software and Hardware Optimization

Practical Group Assignment

Title: Code Optimization and Profiling for Performance-Critical Systems

Weight: 40% of Continuous Assessment
Group Size: 5 Students
Submission Deadline: **Wednesday, August 14, 2025**
Deliverables:

- Max **10-page report** (including introduction, implementation, results, discussion, and references)
- All **code files** (.cpp or .c)
- **Screenshots** of testing, profiling, tool usage
- Contribution table indicating each student's individual work

Assignment Overview

You are required to **design, implement, and optimize** a **C++ application** simulating a real-world computation task. The application must undergo **manual and compiler optimization**, and be **profiled using tools** to assess and enhance performance. This exercise will help you apply theoretical knowledge to practical problem-solving while understanding hardware-aware and software-level optimizations.

Project Task Options (Choose One)

Each group must select **one** of the following task themes:

1. **Matrix Multiplication and Optimization**
    - Implement standard matrix multiplication
    - Optimize using loop unrolling, memory layout improvements, and compiler flags
    - Profile using gprof, perf, and valgrind
2. **Image Convolution Filter (3x3 kernel)**
    - Implement grayscale image filtering using a convolution matrix
    - Optimize using data locality and SIMD (if possible)
    - Profile memory usage and CPU hotspots
3. **Sorting Large Dataset (1M+ integers)**

    - Implement merge sort and quicksort
    - Compare naive vs. optimized versions (inline functions, cache-aware techniques)
    - Profile CPU cycles and memory leaks

4. **Simulation of a Sensor Data Logger**
    - Simulate periodic sensor input (time-stamped values) and log to file
    - Apply file I/O optimization, use memory pooling or buffers
    - Identify and fix any memory inefficiencies

What You Must Do

1. **Research** the theory behind your chosen task and identify optimization targets (e.g., CPU bottlenecks, memory leaks, instruction inefficiencies).
2. **Implement** a base version of the program in **C or C++**.
3. **Optimize** it using:
    - Compiler-level optimization flags (e.g., -O2, -O3, -march=native)
    - Manual code tuning (e.g., loop unrolling, function inlining)
    - Architecture-specific insights if applicable (e.g., RISC-oriented tricks)
4. **Analyze performance** using at least 3 tools:
    - gprof or perf for profiling
    - valgrind for memory leak detection
    - Any other (optional) tool like time, cachegrind, objdump, etc.
5. **Prepare a report** with:
    - Problem description & objective
    - Optimizations applied and their rationale
    - Tool outputs with screenshots
    - Before vs. After performance comparison (execution time, memory use, etc.)
    - Reflections, difficulties, limitations, and further improvement ideas
6. **Document individual contributions** clearly in a contribution table.

Assessment Criteria (40%)

Criterion	**Marks**	**Description**
Understanding of Optimization Concepts (ILO1, ILO3)	8	Depth of explanation, clarity of code changes

Application of Manual Optimization Techniques (ILO3, ILO4)	8	Evidence of meaningful code optimization
Hardware Considerations and Awareness (ILO2)	6	Discussion of architecture or memory access effects
Usage of Profiling Tools (ILO5)	8	Proper application and analysis using tools
Quality of Report & Research (All ILOs)	6	Structure, clarity, citations, screen captures, analysis
Individual Contribution Table	4	Honest and fair declaration of each member's role

Report Format (Max 10 Pages)

1. Title Page (Group Members + IDs + Contribution Table)
2. Introduction & Problem Statement
3. Implementation Overview
4. Optimization Techniques Used
5. Tool-Based Profiling and Analysis
6. Performance Comparison (before/after)
7. Conclusion & Recommendations
8. References (IEEE or APA)
9. Appendix: Screenshots (excluded from page count)

Suggested Tools

- Compiler: g++, clang++
- Profilers: gprof, perf, valgrind, time
- Editor: VSCode, QtCreator, or terminal-based editor
- Linux (Ubuntu 20+ or WSL for Windows users)

Submission Instructions

- Upload a **ZIP file** to Moodle with:
    - Final PDF report
    - Source code files
    - Screenshots folder
- Name your ZIP file: GroupX_OptimizationAssignment.zip (e.g., Group5_OptimizationAssignment.zip)