Assignment 5 - Optimization

Computer Architecture

Instructions

- Work individually.
- Due Date: January 18th, 23:59.
- Submit using the “submit” system.
- Ensure your code compiles with no warnings using the -O0 flag on BIU servers.
- Add your ID and full name at the top of every file in a comment. For example:
- */* 123456789 Israel I s r a e l i */*
- Do **not** use AI tools like ChatGPT. Doing so is considered cheating. (Except for section 2B, where it is required)

Part 1 (90 Points)

You are provided with vowel_counting_original.c, which contains a baseline, unoptimized implementation for counting vowels, digits, and calculating Pi-related metrics.

1. **Rewrite or optimize vowel_counting_original.c.**
2. **Ensure your output remains exactly the same.** The optimized code must return the same results as the unoptimized version; any deviation in results will result in a failing grade regardless of speed.
3. **Benchmark performance** before and after your optimization to demonstrate the speedup.

Files Provided

1 **main.c** - This is the entry point of the application. It reads the input buffer size and the buffer content itself from stdin, invokes the counting functions, and prints the final statistics. main.c is provided for local testing only and **must not be modified or submitted.**

2 **vowel_counting_original.c** - This file contains the baseline, unoptimized "naive" implementation of the character counting and Pi-matching logic. You should use this to verify your results and measure your performance speedup.

3 **vowel_counting.c** - This is the file where you will write your optimized code. It is intended to replace the original implementation while maintaining exactly the same output.

4 **create-buffer.py** - A Python script used to generate random text buffers of a specified size for testing purposes.

5 **run_stats.sh** – A Bash script designed to automate the benchmarking process. It compiles both the original and optimized versions, runs them against an input file, checks for correctness, and calculates the execution time and speedup.

6 **input.txt** – The default text file containing the data to be processed by the programs during testing and benchmarking.

Grading

- **60 points** are awarded if the program runs the given input in **under 3 seconds** on the university server (Planet).
- **10 points** additional points are awarded if it runs in **under 2.5 seconds**.
- **10 points** additional points are awarded if it runs in **under 2 seconds**.
- The remaining **10 points** is based on **performance relative to peers**. Code that is correct but runs slowly will receive a low grade.

Rule and Constraints

- Only code compiled with the -O0 flag is valid.
- Only allowed to use material on subjects taught in the course up to the release of the homework (no using SIMD, Multithreading, vector, or gpu usage).
- You may use the Fork command since that has been tought in class. (it's actually a very good fit for this assignment)
- Small compiler level commands like register int x, is also allowed.
- **Code that returns different answer will get a failing grade regardless of speed.**

Using the Shell Script

To compare both versions (Default): Ensure you have an input file named input.txt in the directory, as well as the rest of the .c files, then run:

- chmod +x run_stats.sh (One time to give the file execution permissions)
- To run only your optimized version
    - . /run_stats.sh optimized
- To compare the optimized version to the original
    - ./run_stats.sh both

Part 2(10 Points)

Part 2A:

As a first step in optimization for job interviews your assignment will be to go to neetcode.io/roadmap. Once there, select a question, and code it as optimized as possible.

The purpose of this assignment is to take the first step into optimizations for job interview questions, and to see what you can do. Feel free to answer the question in any language.

Submit the question you chose, with a link to neetcode, and your written asnwer in a file called interview_prep.txt

Part 2B: Vibe Coding

For Part 2B, solve a different problem using only AI assistance. You should not manually write the solution logic yourself. Briefly describe which AI tool you used. Feel free to use a service like Copilot which is integrated into VS Code.

If you'd like for your own fun play around with vibe coding, maybe try to build a visualization tool for the given question, let the AI lead the way.

Note: Submissions for Part 2 will be graded generously. The primary goal is engagement and learning, not perfection. We recommend taking this seriously to learn about leetcode for job interviews.

Submission Checklist

Submit these files to the submit

- vowel_counting.c (after your optimizations)
- interview_prep.txt (submit both questions and solutions in the same file)