

# Problem-2 (Assignment-1): Gem5 MergeSort Cache Analysis

## Overview

This assignment uses gem5 to simulate two merge sort variants on RISC-V: a simple in-memory version on a 10MB random integer file and a complex chunked version processing 2MB chunks (sorted individually, then merged from 1MB streams). Students run these via provided scripts (simple-riscv\_mergesort\_chunked.py for chunked, simple-riscv\_mergesort\_simple.py for simple), analyze baseline cache performance present in the scripts, and then try out different L1/L2 size and associativity configurations.

## Learning Objectives

By completing this assignment, you will:

1. Install and configure gem5 simulator with RISC-V cross-compiler
2. Understand cache locality differences between simple vs chunked merge sort
3. Analyze gem5 statistics (sim\_ticks, IPC, cache misses, hit rates)
4. Optimize cache configurations for memory-intensive sorting workloads
5. Interpret performance trade-offs between cache size, associativity, and complexity

## Installation Guide

### Step 1: Install Dependencies

- sudo apt update
- sudo apt install dos2unix build-essential git m4 scons zlib1g zlib1g-dev libprotobuf-dev protobuf-compiler libprotoc-dev libgoogle-perftools-dev python3-dev libboost-all-dev pkg-config python3-pydot libpng-dev libcapstone-dev libhdf5-dev

### Step 2: Increase Linux Swap (for gem5 compilation)

- swapon -show
- sudo fallocate -l 20G /swapfile2
- sudo chmod 600 /swapfile2
- sudo mkswap /swapfile2
- sudo swapon /swapfile2

### Step 3: Gem5 Installation

- git clone <https://github.com/gem5/gem5>
- cd gem5
- pip install -r requirements.txt
- pip install -r optional-requirements.txt
- # Native Linux (fast):
  - o scons build/ALL/gem5.opt -j \$(nproc)
- # WSL (slow, use 1 job to avoid OOM):
  - o scons build/ALL/gem5.opt -j 1

### Step 4: RISC-V Cross-Compiler

- sudo apt install gcc-riscv64-linux-gnu binutils-riscv64-linux-gnu

### Step 5: Compile C Program to RISC-V Binary

- riscv64-linux-gnu-gcc -static -o <binary\_name> <c\_program.c> -march=rv64imafdc -mabi=lp64d

#### ISA Explanation:

-march=rv64imafdc: 64-bit base + I/M/A/F/D/C extensions  
-mabi=lp64d: 64-bit pointers + double-precision FP

### Step 6: Run Simulator (from gem5 root)

- build/ALL/gem5.opt --stats-file=<stats\_name>.txt <folder>/<config\_script>.py

#### Notes:

- Edit binary path in config .py file
- Stats file generates in g5out/ folder

# Assignment Tasks

## Part 1: Baseline Comparison

**Objective:** Compare default cache performance between merge sort variants

**Default Cache Config:**

- L1I: 32KiB, 8-way
- L1D: 64KiB, 8-way
- L2: 512KiB, 16-way

**Tasks:**

1. Run both merge sorts with default caches
2. Extract key stats from sim\_\*.txt:
  - sim\_ticks (total cycles)
  - IPC (instructions per cycle)
  - L1D/L2 demand\_accesses, misses, miss\_rate, etc.
3. Create comparison table

**Deliverables:**

- Baseline stats table for both variants
- Analysis (200-300 words): Why does chunked sorting show better locality despite complexity? Find out the parameters which show better result in chunked sorting and why.

## Part 2: Cache Optimization Sweep

**Objective:** Find optimal cache configs for both workloads

**Parameters to Test:** (or instead of us giving specific data, let the students find out by testing)

L1 Sizes: 32KiB, 64KiB, 128KiB

L1 Assoc: 4-way, 8-way, 16-way

L2 Sizes: 256KiB, 512KiB, 1024KiB

L2 Assoc: 4-way, 8-way, 16-way

**Tasks:**

1. Modify cache params in both scripts
2. Run both merge sorts with all configurations
3. Tabulate key metrics:
  - IPC
  - L1D/L2 miss\_rate
  - L1D/L2 access time
  - sim\_ticks
  - etc. (whatever other parameters you feel is important and why)

**Deliverables:**

- Results table.
- Plots (minimum 4): for example
  - L2 miss rate vs L2 size
  - IPC vs L1D size
  - L1D hit rate vs associativity
  - Simple vs Chunked comparison
- Top 3 configs ranked by IPC for each workload
- Analysis: Which configs work best for each sort? Why?

## File Organization

- gem5
  - o <folder>
    - simple-riscv\_mergesort\_chunked.py
    - simple-riscv\_mergesort\_simple.py
    - mergesort\_simple.c
    - mergesort\_chunked.c
  - o g5out
    - <stats\_name>.txt
  - o Random\_numbers.bin

## Submission Checklist

**Note: Everything in a single zipped directory ("problem\_2\_assignment\_1\_soln.zip") to be uploaded on Moodle (one submission per group).**

1. Part 1: Baseline comparison table + analysis (200-300 words)
2. Part 2: Full sweep results table, 4+ plots, top configs
3. All stats files organized