Problem-1 (Assigment-1): Cache Hierarchy Optimization using gem5

Overview

This assignment requires you to investigate how different cache configurations affect processor performance using gem5 simulation. You will run a memory-intensive benchmark (matrix multiplication) with various cache parameters and analyze the results.

Learning Objectives

By completing this assignment, you will:

1. Understand cache hierarchy design and its impact on performance

2. Learn to use gem5 simulator for architecture exploration

3. Analyze performance metrics (execution time, hit rates, misses)

4. Make data-driven design decisions based on simulation results

5. Understand trade-offs between cache size, associativity, and cost

Assignment Tasks

Part 1: Environment Setup (10 minutes)

Objective: Get familiar with the assignment tools

Tasks:

1. Verify gem5 RISCV build: `build/RISCV/gem5.opt --version`

2. Write the gem5 configuration script: `configs/cache_config.py`

3. Test single run with default cache configuration:

```
./build/RISCV/gem5.opt configs/cache_config.py \
  --l1i_size=16kB --l1d_size=16kB --l2_size=256kB \
  --l1_assoc=4 --l2_assoc=8 \
  --binary=<path_to_compiled_binary>
```

Deliverables:

- Cache configuration script

- Screenshot of successful test run

- Output log showing cache statistics

Part 2: Single Parameter Sweep (30 minutes)

Objective: Understand impact of one cache parameter

Choose ONE parameter to investigate:

- L1D cache size (vary: 16kB, 32kB, 64kB)

- L2 cache size (vary: 128kB, 256kB, 512kB, 1MB)

- L1 associativity (vary: 2, 4, 8)

- L2 associativity (vary: 4, 8, 16)

Tasks:

1. Create a custom sweep script (`custom_sweep.py`) for your chosen parameter

2. Run simulations with different values while keeping other parameters constant

3. Collect execution time (ticks) and cache hit rates

4. Create a table and plot showing the relationship

Sample Default Configuration:

- L1I: 16kB, assoc=4

- L1D: 16kB, assoc=4

- L2: 256kB, assoc=8

Deliverables:

- `custom_sweep.py` script

- Results table (CSV or JSON format)

- Plot: Parameter value vs Execution Time

- Plot: Parameter value vs Hit Rate

- Brief analysis (200-300 words):

  - What trend did you observe?

  - Why does this parameter have this effect?

  - Where does performance saturate?

Part 3: Multi-Parameter Analysis (40 minutes)

Objective: Understand parameter interactions

Tasks:

1. Run the full parameter sweep:

```
python scripts/cache_sweep.py \
    --gem5=../../build/RISCV/gem5.opt \
    --config=configs/cache_config.py \
    --binary=<path_to_binary> \
    --output=full_sweep_results
```

2. Analyze results using:

```
python scripts/analyze_sweep.py \
    full_sweep_results/results.json \
    --output=analysis_plots
```

3. Examine the generated plots and statistics

Deliverables:

- `results.json` from complete sweep
- Analysis plots (at minimum):
  - L1D size impact
  - L2 size impact
  - Associativity impact
- Summary statistics (mean, min, max for key metrics)
- Top 3 configurations ranked by:
  - Lowest execution time
  - Highest L1D hit rate
  - Highest L2 hit rate

Part 4: Design Analysis & Recommendations (30 minutes)

Objective: Make data-driven microarchitecture design decisions

Tasks:

1. Compare your single-parameter results with full sweep results
2. Identify the "Pareto-optimal" configurations:
   - Configs where you can't improve one metric without hurting another

3. Answer the following questions:

a) Performance Bottlenecks (5 points)

- Is execution time dominated by L1D misses, L2 misses, or memory stalls?
- What percentage of memory requests reach main memory?

b) Cache Efficiency (5 points)

- Which cache level has the best hit rate? Why?
- Is L2 size or associativity more important?

c) Cost-Benefit Trade-off (5 points)

- What's the smallest L1D+L2 configuration that achieves 90% of peak performance?
- How much performance do you lose by using direct-mapped caches (assoc=1)?

d) Design Recommendations (5 points)

- Recommend an optimal configuration for:
  - **Power-constrained system** (minimize cache size)
  - **High-performance system** (maximize performance)
  - **Balanced system** (best performance/cost ratio)
- Justify your recommendations with data

Deliverables:

- Answers to all 4 questions (500-800 words total)
- Graphs comparing Pareto-optimal vs suboptimal configs
- Design recommendation table with justification

File Organization

```
assignment_cache_optimization/
├── benchmarks/
│   └── matrix_multiply.c
├── configs/
│   └── cache_config.py
├── scripts/
│   ├── cache_sweep.py
```

| └── analyze_sweep.py

├── results/

| ├── single_param_sweep/

| ├── full_sweep_results/

| └── analysis_plots/

└── ASSIGNMENT-1.docx (this file)

Compilation

To compile the matrix multiply benchmark for RISCV:

```# Requires RISCV cross-compiler

riscv64-unknown-linux-gnu-gcc -O2 -static benchmarks/matrix_multiply.c -o matrix_multiply

```

Optional Enhancements

- Test with different matrix sizes (64x64, 256x256)

- Vary memory bandwidth

- Compare different CPU types (TimingSimpleCPU vs O3CPU)

- Add power estimation metrics

Submission Checklist

Note: Everything in a single zipped directory ("problem_1_assignment_1_soln.zip") to be uploaded on Moodle (one submission per group).

- [ ] Part 1: Test run output and screenshots

- [ ] Part 2: Custom sweep script, results table, 2 plots, 200-300 word analysis

- [ ] Part 3: Full sweep results.json, 3+ analysis plots, statistics summary

- [ ] Part 4: Answers to all 4 design questions (500-800 words), recommendation table

- [ ] README with instructions to reproduce results

- [ ] All code properly commented and documented

Additional Resources

- [gem5 Documentation](https://www.gem5.org/documentation/)