CS 429/529
Project 1 – Cache Simulator

1. Introduction

For Project 1, you will implement a suite of cache simulations in a programming language of your choice.

Teams

You may solve this homework in teams of two students to increase discussion, participation, and engagement. If you work with another student, only one should submit your solution, including both names in all files.

Data Sets

You will run each simulation using three application traces of memory references:

- C compiler ("cc.trace")
- circuit simulator ("spice.trace")
- Tex document processing ("tex.trace")

Find the traces on Canvas.

Each line of each data set contains two integer values. The first integer specifies the type of memory reference: 0 for a data read, 1 for a data write, and 2 for an instruction read. The second integer is the hexadecimal value of the 32-bit memory address to be accessed. Note that the cache sizes used in this assignment will be appropriately small with respect to these traces so that you can see interesting behavior features.

2. Parts

1. [20] Write-through Cache Class

Implement a *set associative write-through cache class*. The total size in bytes, block size in bytes, and set associativity (i.e., blocks/set) should all be specified in the constructor. Assume that each write must access main memory sequentially and therefore counts as a cache miss. Your cache should only register a read hit on blocks previously written to or loaded from main memory via a cache miss (e.g., using a valid bit for each block). Your cache should use the *least recently used (LRU)* replacement policy within a set.

Tip: When designing this class, consider that you will also be implementing a write-back cache class.

2. [10] Write-through Instruction and Data Caches

Implement a driver that creates two instances of your write-through cache class, one to represent an *L1 instruction cache* and one to represent an *L1 data cache*. Both caches should store 1024 bytes in 32-byte blocks. Simulate each trace with an associativity of 1, 2, 4, 8, 16, and 32 for both caches. A set associativity of S is equal to the number of blocks (S=1 is direct-

mapped, and S=32 is fully associative, in our case). Assume that each cache has a *hit time* of H cycles (typically 1) and a *miss penalty* of M cycles (assume 100). Calculate the *average memory access time (AMAT)* in cycles for each run as:

$$AMAT = H + missrate * M$$

3. [20] Write-back Cache

Implement a set associative write-back cache class similar to your write-through cache class (you might find it helpful to inherit from a common parent class). Your cache should only write a block to main memory if that block has changed since the last time the block was read from or written to main memory (e.g., using a dirty bit for each block). Hint: both reads and writes can trigger a block write. Your cache should again use LRU as its replacement policy within a set.

4. [10] 2. [10] Write-back Instruction and Data Caches

Implement a driver that creates two instances of your write-back cache class, one to represent an L1 instruction cache and one to represent an L1 data cache. Both caches should store 1024 bytes in 32-byte blocks. Simulate each trace with an associativity of 1, 2, 4, 8, 16, and 32 for both caches. Use the cache hit and miss times from part 2 and again calculate the AMAT for each run. Assume that cache write misses, which do not replace a dirty block, count as cache hits because no main-memory access is required.

5. [20] Two Cache Levels

Implement a driver that creates three instances of your write-back cache class: one to represent an L1 instruction cache, one to represent an L1 data cache, and one to represent a shared L2 cache. The L1 caches should store 1024 bytes in 32-byte blocks and be 2-way associative. The L2 cache should store 16384 bytes in 128-byte blocks. Simulate each trace with an associativity of 1, 4, 16, 64, and 128 for the L2 cache. Assume that the L1 caches have a 1-cycle hit time (L1H=1) and that the L2 cache has a hit time of 10 cycles (L2H=10). The L2 miss penalty is 100-cycle (L2M=100). Calculate the AMAT for each run as:

$$AMAT = L1H + L1missrate * (L2H + L2missrate * L2M)$$

6. [20] Data Collection

Collect and analyze miss rate and AMAT data by varying the following parameters from part 5: L1 cache size, L1 block size, L1 associativity, L2 cache size, L2 block size, and L2 associativity. You may adjust the parameters of the two L1 caches independently. Create at least three plots to illustrate your findings, and write at least one paragraph analyzing each plot and a paragraph summarizing your findings.

7. Grad Students Extra Work

Implement an additional driver that simulates the L1 and L2 caches from part 5 but also simulates a fully associative victim cache that holds entries recently flushed from the L2 cache. Experiment with different sizes and access times to find configurations

that improve overall performance. Analyze your results along with your results from part 6.

8. [0] Advised Work

You might find it helpful to create traces to exercise your simulator to validate its operation. For instance, you might generate an address trace that will touch every location in the cache with one set of addresses and do it again with another different set of addresses. You should see that every block brought in by the first set of addresses gets replaced by the second set. You can write these trace generators in a program to output an address trace on standard output. You can write it to a file or pipe it into your cache simulator. (Note: This part is optional.)

3. Report

Please produce a report that contains the following:

Steps 1 and 2:

Present a table that organizes the results from the experiments described in Step 2. The table must contain:

Assoc. Level	L1I accesses	L1I misses	L1D accesses	L1D misses	L1I hit rate	L1D hit rate	AMAT
1	---	---	---	---	---	---	---
2	---	---	---	---	---	---	---
4	---	---	---	---	---	---	---
8	---	---	---	---	---	---	---
16	---	---	---	---	---	---	---
32	---	---	---	---	---	---	---

Steps 3 and 4:

Include a table that organizes the results from the experiments described in Step 4. It should contain:

Assoc. Level	L1I accesses	L1I misses	L1D accesses	L1D misses	L1I hit rate	L1D hit rate	AMAT
1	---	---	---	---	---	---	---
2	---	---	---	---	---	---	---
4	---	---	---	---	---	---	---
8	---	---	---	---	---	---	---
16	---	---	---	---	---	---	---
32	---	---	---	---	---	---	---

Step 5:

Produce a table that organizes the results from all of the experiments described. It should contain:

Assoc. y	L1I accesses	L1I misses	L1I hit rate	L1D accesses	L1D misses	L1D hit rate	L2 accesses	L2 misses	L2 hit rate	AMAT
1	---	---	---	---	---	---	---	---	---	---
4	---	---	---	---	---	---	---	---	---	---
16	---	---	---	---	---	---	---	---	---	---
64	---	---	---	---	---	---	---	---	---	---
128	---	---	---	---	---	---	---	---	---	---
32	---	---	---	---	---	---	---	---	---	---

Step 6:

It is your choice of how to vary the parameters. Try to have at least five variations; describe what they are. The three graphs should highlight interesting behaviors, if any. The paragraph you write summarizing the findings should be clear.

Step 7:

Grad Students: Describe what you did and show results as in Step 6.

Step 8:

If you do this, please briefly describe what your trace generators are doing and for what purpose to test your cache simulator.

4. Submission

You will submit your project report, code, and other files on Canvas. Please submit your report in PDF. Please package your code in a TAR or ZIP file. If you want to provide additional files, please package them in a separate TAR or ZIP file.

This file may change in response to your questions/suggestions.