Indian Institute of Technology, Roorkee
CSN-221 : Computer Architecture and Microprocessors
Course Project

Roopam Taneja

November 8, 2023

Github Link : https://github.com/RoopamTaneja/CSN-221-Comp-Arch-Cache

Configurable Cache Simulator

This project was made as part of course CSN-221 : Computer Architecture and Microprocessors in Autumn Semester 2023-24.

Associativity can be modified to configure a *N-way set-associative, direct-mapped or fully-associative cache.*

The cache simulator uses **LRU (least recently used)** replacement policy.

It uses **Write Back and Write Allocate policies** for handling stores.

It also evaluates statistics like hit rate, average memory access time, no of loads, no of stores etc.

Contents of Cache Folder

- Configurable data cache simulator `cache_sim.cpp` written in C++ which also simulates main memory for integrating with processor simulator.
- Another simulator `raw_cache_sim.cpp` written in C++ which does not handle data. It is meant for evaluating statistics for large memory traces which are beyond the scope of first simulator.

  The file `state_table.xlsx` tabulates the states of cache blocks and their transitions and the few design choices made to simplify the design.

Cache Simulator Description and Usage Guidelines

The cache simulator will take as arguments one .txt file containing your memory traces.

Commands: After compiling `cache_sim.cpp` or `raw_cache_sim.cpp`, run:

```
./<file_name>.exe <trace_file>.txt
```

Some sample traces taken from links in References have been provided in `small_traces` and `large_traces`.

- `small_traces`: Contains memory traces which can be tested on either simulator. Also contains a C++ script for generating new random traces.

- `large_traces`: Contains large memory traces which MUST be tested only on `raw_cache_sim.cpp`. Also contains two C++ scripts used for processing traces which were not in right format. The traces in the folder are already processed and can be used directly.

You can also add you own traces in .txt format.

Guidelines:

1. Cache configuration parameters must be specified only in the `params.txt` file present in Cache folder.
2. Parameters needed to be specified :
   - Cache Size (in KB)
   - Associativity (power of 2)
   - Block Size (in bytes and multiple of 4)
   - Miss Penalty (in no of cycles)
3. The memory is BYTE-ADDRESSABLE.
4. All loads and stores happen in terms of a single 32-bit word.
5. Trace Format:
   - For `cache_sim.cpp`:
     `LS ADDRESS DATAWORD`
     where LS is a 0 for a load and 1 for a store, ADDRESS is an 8-character hexadecimal number, and DATAWORD is the 8-character hexadecimal data provided only in case of stores. There is a single space between each field.
   - For `raw_cache_sim.cpp`:
     `LS ADDRESS IC`
     where LS is a 0 for a load and 1 for a store, ADDRESS is an 8-character hexadecimal number, and IC is the number of instructions (in base 10) executed between the previous memory access and this one (including the load or store instruction itself). There is a single space between each field. The instruction count information will be used to calculate execution time in cycles.
6. Remember to change the size of main_memory array in `cache_sim.cpp` according to the addresses of your trace. (Current size of 1 MB works fine with all traces in `small_traces` folder)

Calculations

- **Hit Rate**:

$$\text{Hit Rate} = \frac{\text{hits}}{\text{hits} + \text{misses}}$$

- **Miss Rate**: 1 - hit_rate
- **Cache access and operation time**: 1 cycle (common to hits & misses)
- **Miss Penalty**: Main memory access and operation time (for misses) = User specified
- **Extra penalty for memory writes for dirty evictions**: 2 cycles (assumed)

The Average Memory Access Time (AMAT) is calculated as follows:

$$\text{AMAT} = \text{hit\_rate} \times \text{hit\_time} + \text{miss\_rate} \times \text{miss\_time}$$

$$\text{AMAT} = \text{hit\_rate} \times 1 + \text{miss\_rate} \times (1 + \text{miss\_pen}) + \frac{\text{dirty\_evict} \times 2}{\text{hits} + \text{misses}}$$

$$\text{AMAT} = 1 + \text{miss\_rate} \times \text{miss\_pen} + \frac{\text{dirty\_evict} \times 2}{\text{hits} + \text{misses}}$$

Assuming CPI = 1 for non-load store instructions:

- **Overall CPI**:

$$\text{CPI} = \frac{\text{Total Cycles}}{\text{Instr Count}}$$

- **Memory CPI**: Overall CPI − 1

Comparative Study

go_ld_trace

Memory accesses = 1500000

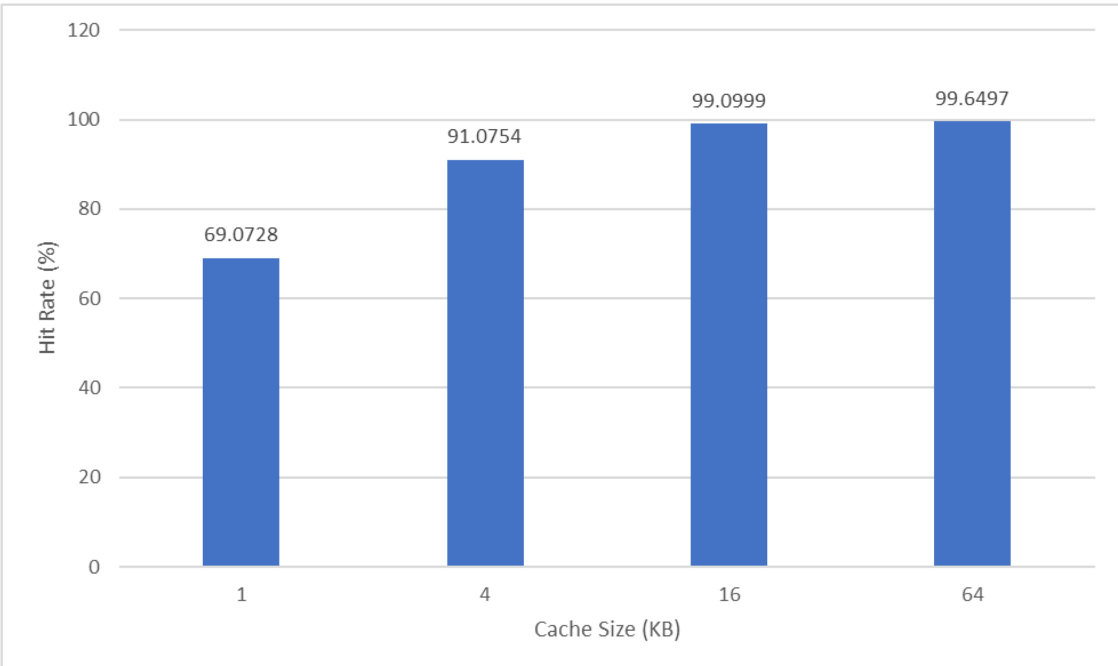
Loads = 1500000

Stores = 0 (so no dirty evictions)

Miss penalty = 30 cycles

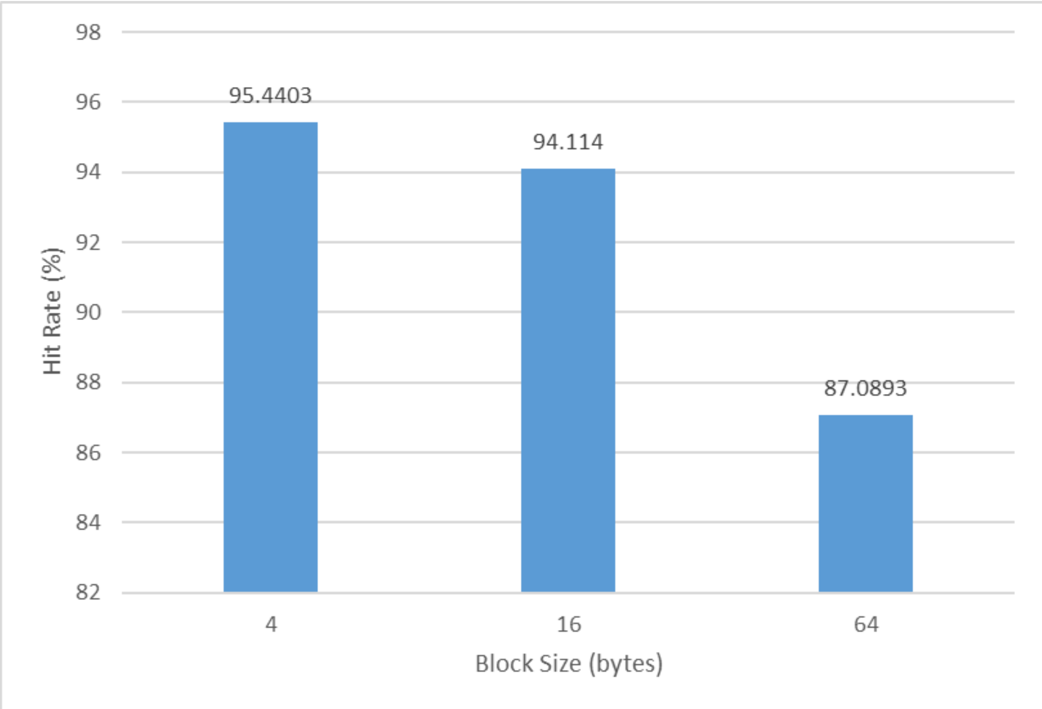
1. Block Size = 32 bytes and Associativity = 4

Cache Size (KB)	No of Hits	Hit Rate (%)	AMAT (cycles)
1	1036092	69.0728	10.2782
4	1366131	91.0754	3.67738
16	1486499	99.0999	1.27002
64	1494746	99.6497	1.10508



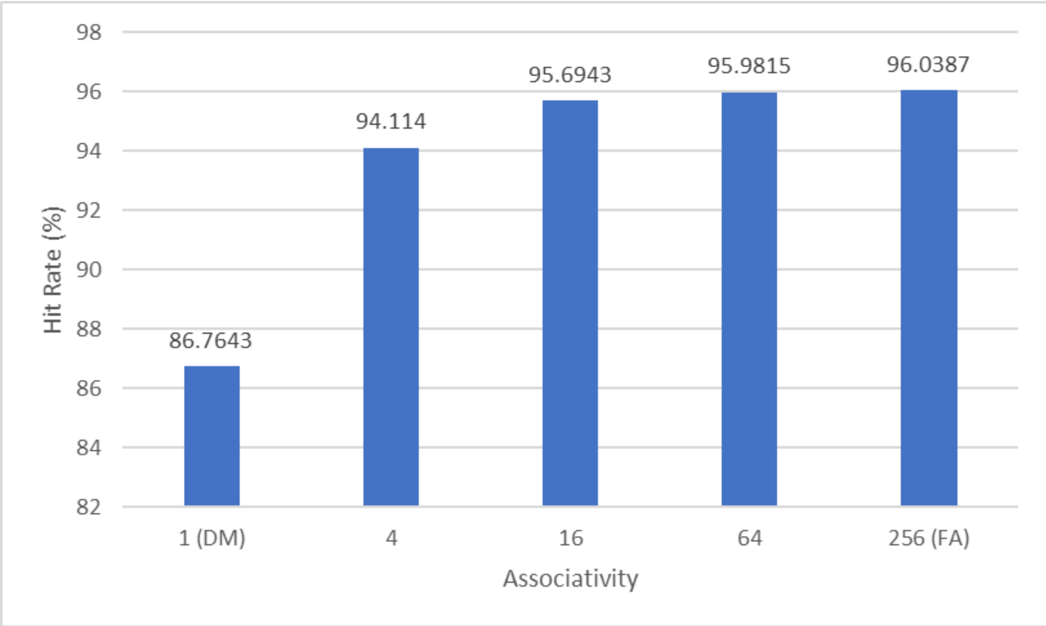
2. Cache Size = 4 KB and Associativity = 4

Block Size (bytes)	No of Hits	Hit Rate (%)	AMAT (cycles)
4	1431605	95.4403	2.3679
16	1411710	94.114	2.7658
64	1306340	87.0893	4.8732



3. Cache Size = 4 KB and Block Size = 16 bytes

Associativity	No of Hits	Hit Rate (%)	AMAT (cycles)
1 (DM)	1301464	86.7643	4.97072
4	1411710	94.114	2.7658
16	1435415	95.6943	2.2917
64	1439722	95.9815	2.20556
256 (FA)	1440581	96.0387	2.18838



mcf_trace

Memory accesses = 6943857

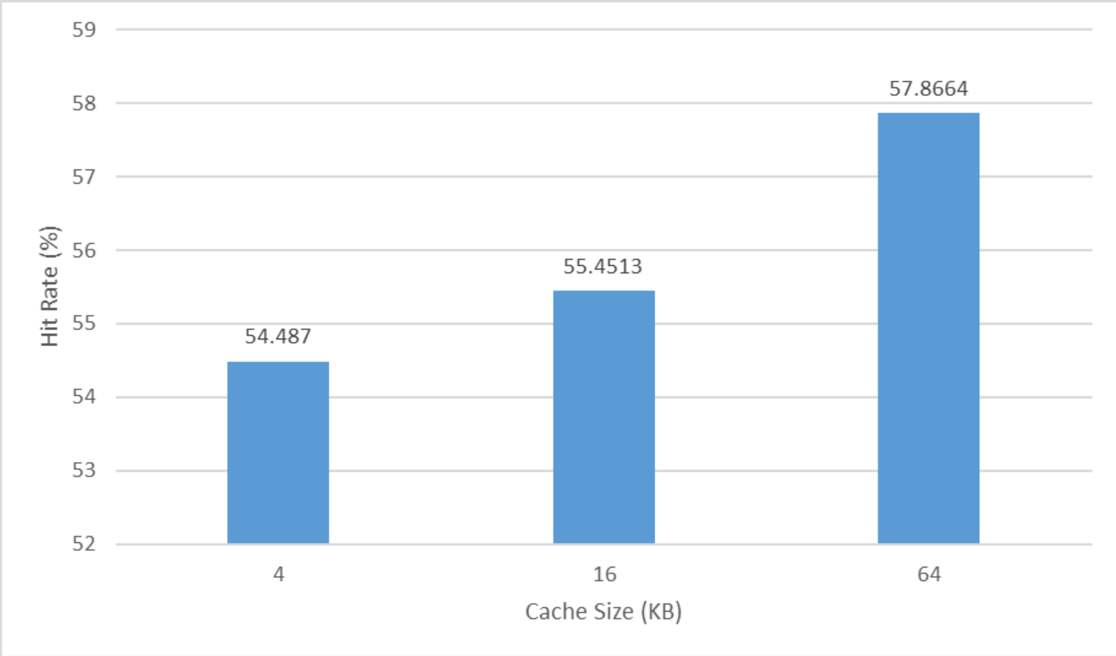
Loads = 5589472

Stores = 1354385

Miss penalty = 30 cycles

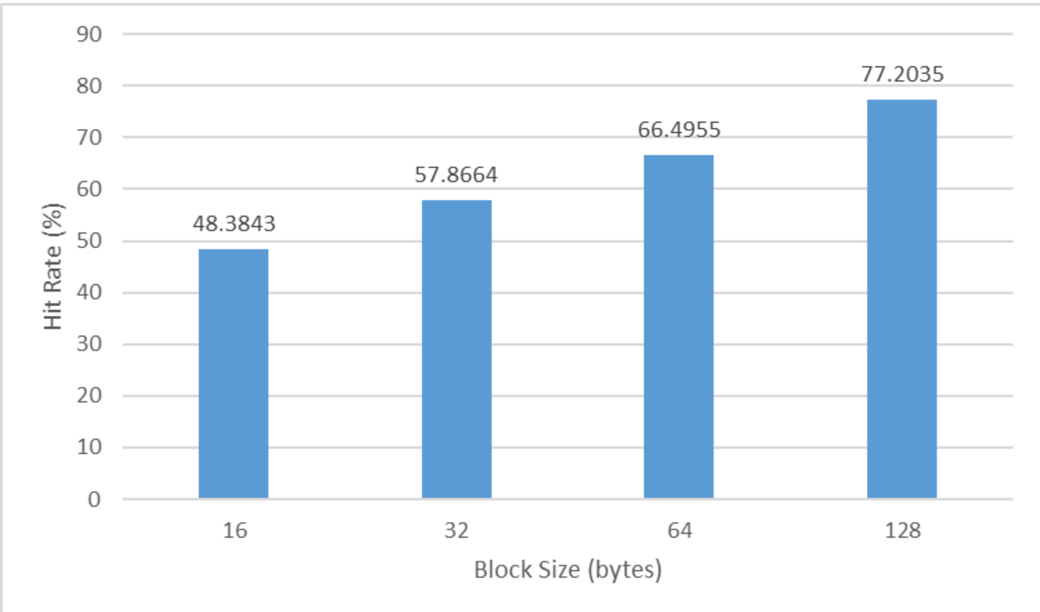
1. Block Size = 32 bytes and Associativity = 4

Cache Size (KB)	Hits	Dirty Evictions	Hit Rate (%)	AMAT (cycles)
4	3783501	1061484	54.487	14.9596
16	3850460	1043876	55.4513	14.6653
64	4018157	999959	57.8664	13.9281



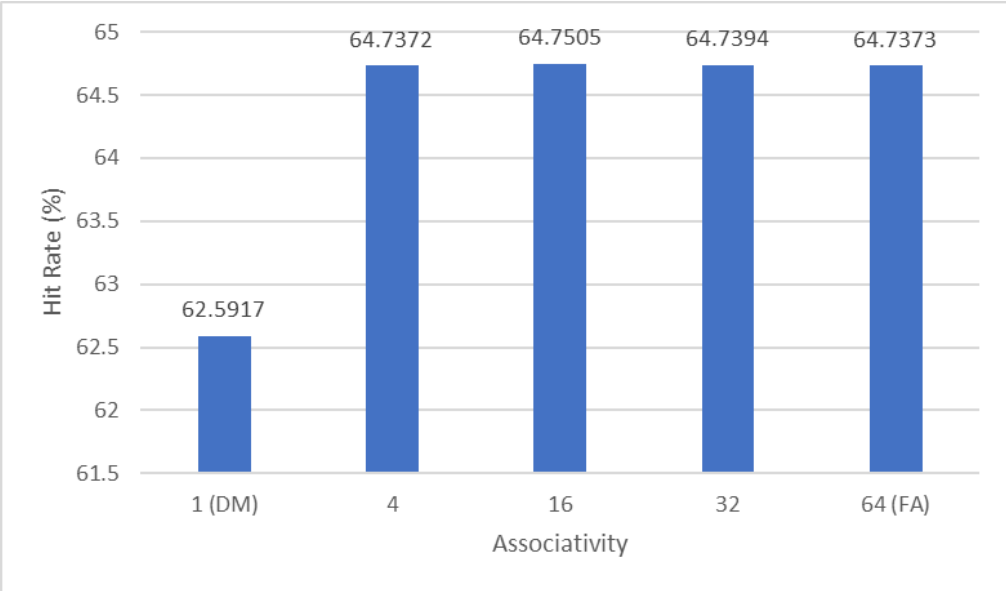
2. Cache Size = 64 KB and Associativity = 4

Block Size (bytes)	Hits	Dirty Evictions	Hit Rate (%)	AMAT (cycles)
16	3359735	999732	48.3843	16.7727
32	4018157	999959	57.8664	13.9281
64	4617352	1006328	66.4955	11.3412
128	5360904	940748	77.2035	8.10989



3. Cache Size = 4 KB and Block Size = 64 bytes

Associativity	Hits	Dirty Evictions	Hit Rate (%)	AMAT (cycles)
1 (DM)	4346281	1065500	62.5917	12.5294
4	4495260	1043702	64.7372	11.8794
16	4496183	1043955	64.7505	11.8755
32	4495408	1044430	64.7394	11.879
64 (FA)	4495266	1044598	64.7373	11.8797



References

1. https://occs.oberlin.edu/~ctaylor/classes/210SP13/cache.html
2. https://www.cs.utexas.edu/users/mckinley/352/homework/project.html
3. http://www.cs.uccs.edu/~xzhou/teaching/CS4520/Projects/Cache/Cache_Simulator.htm