Cache Simulator by
Rishit Jakharia, 2022CS11621

Introduction

You must provide a Makefile such that

- make clean removes all object files and executables
- make compiles and links the program, producing an executable called cacheSim

Cache Configuration

The cache simulator is configured with the following cache design parameters which are given as command-line arguments (see below):

- number of sets in the cache (a positive power-of-2)
- number of blocks in each set (a positive power-of-2)
- number of bytes in each block (a positive power-of-2, at least 4)
- write-allocate or no-write-allocate
- write-through or write-back
- lru (least-recently-used) or fifo evictions

Note that certain combinations of these design parameters account for direct-mapped, set associative, and fully associative caches:

- a cache with n sets of 1 block each is direct-mapped
- a cache with n sets of m blocks each is m-way set-associative
- a cache with 1 set of n blocks is fully associative

Assumptions

The cache simulator assumes that loads/stores from/to the cache take one processor cycle; loads/stores from/to memory take 100 processor cycles for each 4-byte quantity that is transferred.

Results for “write-strategy”, “write-policy” and “replacement-policy” testing

On gcc.trace data

	write-allocate	no-write-allocate	
write-back	91,38,709	1,47,23,611	**LRU**
	1,05,64,141	1,51,95,841	**FIFO**
write-through	8,15,08,541	8,06,38,411	**LRU**
	8,17,13,341	8,06,79,841	**FIFO**

On swim.trace data

	write-allocate	no-write-allocate	
write-back	91,82,780	1,05,28,087	**LRU**
	1,00,80,768	1,05,42,442	**FIFO**
write-through	3,40,43,568	3,37,25,687	**LRU**
	3,44,87,568	3,37,33,642	**FIFO**

Results for associativity

(Used LRU for the below table, with gcc.trace)

	Fully Assoc.	Set-Assoc	Direct	
write-back	1,61,90,469	91,38,709	6,61,69,901	**write-alloc**
	1,68,94,087	1,47,23,611	4,04,25,892	**no-write-alloc**
write-through	8,71,30,141	8,15,08,541	12,36,63,341	**write-alloc**
	8,19,38,087	8,17,13,341	9,09,01,492	**no-write-alloc**

Clearly Set-Associative with LRU, write-back and write-through seems to be the best

For the best Set-Assoc. Configuration

(number of sets (Column 1), blocks per set (Row 1), Block size in bytes (last Column))

	2	4	8	16	
128	14138563	7304419	6883713	6677353	4
	19006521	9687601	9173081	8831533	16
	21828485	12870997	12024213	11154917	64
512	9079778	6449048	6311810	6207311	4
	10352525	8707801	8191289	7599729	16
	12261845	11430933	10381541	10209141	64
2048	6296997	6105410	5878287	5681267	4
	8198957	7881865	6985653	6917665	16
	9911109	9877477	9709237	9707621	64

Finer Adjustments

	8	16	32	64	
1024	6124132	5930912	5744852	5637734	4
	7571553	7046487	6637349	6632047	8
	7553265	7140181	7014469	7014469	16

Conclusion

1. **Number of sets** peaks at 1024 after which the cycles increase
2. **Blocks Per set** increase till 32 bit then have no significant effect with 64

Best Configuration is

32-way Set Associative with 1024 Sets, write-back, write-allocate, LRU

Screenshots