Lab 1 Cache Simulator Report

1. Abstract

In this lab, a cache simulator is implemented. This cache simulator can simulate multiple level of caches and various cache parameters like number of blocks in each cache, associativity and hit time, as well as global cache parameters, like word size, block size and memory access time. To simulate more realistic caches, write through and write back are supported, and LRU is the replacement policy used in this simulator. After configuring caches, hit rate for each level will be given, along with AMAT of the whole memory architecture.

2. Introduction

This vCache cache simulator supports 3 levels of caches, which can be configured separately. Word size and block size are fixed for each level of cache, and write/read memory is assumed to take the same time. Parameters like number of blocks, associativity, hit time for each cache level are specified by user input.

Associativity of 1 is actually direct mapping cache, while associativity of block number is fully associative cache. Replacement policy used in vCache is LRU algorithm. An independent counter keeps records of asscessing for each block in the cache, and picks the least recently used block to be replaced by new data.

Write through and write back are supported. In write through case, for each time a write block occurred, data is also written to lower level of caches and memory. Therefore data in caches are always the same with memory's. In write back case, writing block only happens in caches. Before a block is to be replaced, check dirty bit firstly. If dirty bit is true, write data to memory before kick out this block. Write allocate is used in vCache.

Hit counter calculates hit rate and miss rate for each level, and then we can get AMAT according to each level's hit time and memory access time.

3. Experiments

3.1 Large block size to reduce miss rate

Experiments of changing block size show that, larger block size leads to lower AMAT and miss rate. It is obvious when containing more bytes in a block, the tag size and index size will shrink, so that address lines in ASCII test file has more identical index and tag bits, which means higher hit rate and better AMAT performance. Table 2 shows results of various block with fixed word size. Table 1 is other parameters in block size experiments.

Cache	Block size (words)	Hit rate	Miss rate	AMAT (ns)
L1	4	21.97%	78.03%	790.29
L1	16	66.95%	33.05%	340.53
L1	32	92.93 %	7.07%	80.72
L1	64	93.95%	6.05%	70.55
L1	128	99.95%	0.05 %	10.55

Table 1. Increasing block size to reduce miss rate

Word size (Bytes)	Write policy	Memory access time (ns)	Hit time (ns)	Test file	# of blocks	Associativity
4	Write through	1000	10	ASCII_many (51200 lines)	64	4

Table 2. General cache parameters

Word size also affects offset length in each entry. Since index size is determined by block number and associativity, word size and block size will affect tag size in a fixed length entry. Word size in this lab is in bytes, while block size is in words, therefore, actual offset (address access in bytes) should be word size * block size, in byte units, too. Table 3 shows effect of word size to cache performance.

	Word size (bytes)	Hit rate	Miss rate	AMAT (ns)
L1	1	21.97%	78.28%	790
L1	2	66.95%	33.05%	340.53
L1	3	92.93%	7.07%	80.72
L1	4	92.93%	7.07%	80.72
L1	8	93.94%	6.05%	70.55

Table 3. Effect of increasing word size

Block size (Words)	Write policy	Memory access time (ns)	Hit time (ns)	Test file	# of blocks	Associativity
32	Write through	1000	10	ASCII_many	64	4

Table 4. General cache parameters

3.2 **Large caches to reduce miss rate**

It is easy to observe that larger cache size enable more address/data cached in caches, so hit rate as well as AMAT performance will increase. Experiments with various number of blocks verify this hypothesis, as Table 5 shows.

	# of Blocks	Hit rate	Miss rate	AMAT (ns)
L1	16	54%	46%	470
L1	32	64.98%	35.02%	360.18
L1	64	92.93%	7.07%	80.72
L1	128	94.92%	5.08%	60.76
L1	256	99.91%	0.09%	10.86

Table 5. Increasing cache size to reduce miss rate

Block size (Words)	Word size (bytes)	Write policy	Memory access time (ns)	Hit time (ns)	Test file	Associativity
32	4	Write through	1000	10	ASCII_many	4

Table 6. General cache parameters

3.3 **Higher associativity to reduce miss rate**

Increasing associativity boosts cache performance by relieving conflict miss. In Table 7, for one level cache architecture, AMAT decreases from 730 ns to 10 ns when increasing associativity. Hit rate is as high as 99.91% in 8 way and 16 way configurations.

	# of blocks	Associativity	Hit time (ns)	Hit rate	Miss rate	AMAT (ns)
L1	64	1	10	28%	72%	730
L1	64	2	10	85.94%	14.06%	150.59
L1	64	4	10	92.93%	7.07%	80.72
L1	64	8	10	89.93%	10.07%	110.66
L1	64	16	10	99.91%	0.09%	10.86

L1	64	32	10	99.91%	0.09%	10.86
L1	64	64	10	99.85%	0.15%	11.48

Table 7. Increasing associativity (write back)

3.4 **Multilevel caches to reduce miss penalty**

Testing with the given sample file and extended test file, cache simulator shows that AMAT is reduced heavily by adding cache levels. The following table is data for various numbers of caches. ASCII_many is an extended version of given sample test file, containing 51200 address lines. Write policy is write back in this experiment.

# of caches	Cache	# of blocks	Associativity	Hit time (ns)	Hit rate	Miss rate	AMAT (ns)
1	L1	16	2	10	53.998%	46%	470.02
2	L1	16	2	10	53.998%	46%	315.15
	L2	32	4	20	35.67%	64.33%	
3	L1	16	2	10	53.998%	46%	76.17
	L2	32	4	20	35.67%	64.33%	
	L3	64	8	30	83.75%	16.25%	

Table 8. Configuration of each cache level with write back

Word size (Bytes)	Block size (words)	Memory access time (ns)	Test file
4	32	1000	ASCII_many

Table 9. Global parameters

3.5 **Write back VS write through**

If write policy is write-through, all the others are kept the same as section 3.4, AMAT for each experiments are:

# of caches	Cache	# of blocks	Associativity	Hit time (ns)	Hit rate	Miss rate	AMAT (ns)
1	L1	16	2	10	53.998%	46%	470.02

2	L1	16	2	10	53.998%	46%	369.38
	L2	32	4	20	23.88%	76.12%	
3	L1	16	2	10	53.998%	46%	130.37
	L2	32	4	20	23.88%	76.12%	
	L3	64	8	30	71.25%	28.75%	

Table 10. Configuration of each cache level with write through

Compared Table 8 with Table 10, write back has better AMAT performance than write through. The reason is write back keeps more data in caches, since newly written data does not have to be written to lower caches until replacement happens.

4. **Conclusion**

There are two main aspects to address from what I've learned from this projects.

Firstly, I get to know cache memory hierarchy better than before. Important design choices in improving cache performance, as the textbook introduced, are verified one by one in cache simulator. Experiment results also verify the correctness of this cache simulator.

Secondly, hands on work of writing cache simulator asks for many details like implementing LRU algorithm, write through and write back policy. At the very beginning, I wrote a simple 3 level cache with random replacement algorithm and write through policy, which can only handle limited length of HEX address. Then write-back policy is added, with cache performance improvement. At last, LRU is implemented, by using usage counter in each block.

Lesson I learned from this project is trying from simple to complex would benefit development progress. Discussing with classmates and hands on work are both very helpful for understanding textbook content.