

Department of Computer Science
National Tsing Hua University
EECS403000 Computer Architecture
Spring 2023 Homework 6
 Deadline: 2023/06/11 23:59


Q1 and Q2 are simulation questions in which we will use the cache simulator in the Ripes, a lightweight RISC-V pipeline simulator, to observe the behavior of the cache.

First, follow the steps to set up the cache simulator in the Ripes.

- (1) Please load the program you wrote in HW5 into the Ripes. Also, do not forget to set the processor to be a *64-bit 5-stage processor* and the global pointer (x3/gp) to be *base+0x800*, respectively. This step is the same as what we did in HW5. You may check the setup tutorial released with HW5 if you are not sure whether you are doing it correctly.



Note 1: The correctness of your program doesn't matter for HW6. So, don't worry if you messed up the previous homework. Or, if you encountered difficulties doing Q1 and Q2 with your own program, you may directly use the provided "[KMP.elf](#)" which helps you away from any potential troubles :)

Note 2: Be careful that the directory path of your program (.elf file) must contain alphanumeric characters only.

- (2) Click the "Cache button 
- (3) Set the cache configuration of L1 data cache to be 2^3 lines, 2^1 ways, and 2^0 words/line, as the picture shows.

L1 Data Cache L1 Instr. Cache

Cache configuration:

Preset:  

2^N Lines: 3 Repl. policy: LRU
 2^N Ways: 1 Wr. hit: Write-back
 2^N Words/Line: 0 Wr. miss: Write allocate

Index	V	D	LRU	Tag	Word 0
0	0	0	1		
1	0	0	1		
2	0	0	1		
3	0	0	1		
4	0	0	1		
5	0	0	1		
6	0	0	1		
7	0	0	1		

This setting implies that our data cache would be a 2-way set associative cache with 8 sets (index 0~7) and 1 double word (i.e., 8 bytes) for each block. Thus, the cache has 16 blocks with a size of 128 bytes in total.

Also, keep in mind that we adopt LRU, write-back, and write allocate for the replacement policy and write handling. For LRU in the Ripes, the smaller the value, the more recently used the corresponding block.

- (4) Now you are ready to run some simulations to see how the cache works in real-time. To stop and check the current CPU/cache state at certain instructions, you can switch to the "Editor" panel and add some breakpoints by clicking the left blue bar.
- Besides, here are some useful tips from the Ripes' documentation for you to interact with the cache simulator and observe how it works.

Note 3: Don't get confused if you find any mismatches between the cache simulator and the memory view in Ripes. Mismatch happens because the processor models in the Ripes actually don't access the cache simulator when accessing the memory. That is, the Ripes doesn't follow the cache simulator to maintain its memory view. You may read the [docs](#) if you want to know more about the details. However, focusing on the cache simulator is good enough to observe the cache's behavior for this homework.

2. (6 points) Improve the hit rate by designing your own cache. You may adjust the associativity or the cache size. Please give some screenshots to show (i) the cache you design and (ii) the improvement of the hit rate. A brief discussion about why the cache you design makes the hit rate higher is also needed. Note that if you cannot get a better hit rate somehow, a reasonable discussion is still needed. Otherwise, you can use the given ELF example that a higher hit rate is definitely possible.

This is the end of the simulation assignment. Please continue with Q3 ~ Q6.

3. (36 points) Consider four different cache configurations below:

Cache 1	direct-mapped with one-word blocks
Cache 2	direct-mapped with four-word blocks
Cache 3	2-way set associative with two-word blocks and LRU replacement
Cache 4	Fully associative with four-word blocks and LRU replacement

Assume that each cache has a total data size of 16 32-bit words with all the blocks initially empty. The word address has 20 bits.

Please write down the calculation process in detail. Otherwise, you will get zero points.

- (a) (10 points) What are the numbers of tag, index, and block offset bits for caches, respectively? How many total bits are required for each cache, including tag and valid fields? Please fill out the table below.

	Address format			Cache size
Cache Type	Tag (bits)	Index (bits)	Block offset (bits)	Total size (bits)
Cache 1				
Cache 2				
Cache 3				
Cache 4				

- (b) (26 points) Given a series of word-address references:

16, 17, 18, 19, 20, 48, 49, 17, 48, 49, 17, 5, 6, and 7.

- (18 points) For the caches 2, 3 and 4, label each reference in the list as a hit or a miss. If the reference is a miss, show what kind of miss occurred (e.g., compulsory, conflict or capacity).
- (8 points) For the caches 2 and 3, show the final cache contents (the word addresses in each block or set) after all references. Please show the block/set number and write down the content of each block.

4. (12 points) Consider the (7,4) Hamming code. Each codeword contains 7 bits, i.e., p1, p2, d1, p4, d2, d3, d4. Answer part (a) for single error correcting (SEC) code. For parts (b) and (c), we add an additional parity bit, so that the code can detect two errors. One additional equation holds for this need: $p1 \oplus p2 \oplus d1 \oplus p4 \oplus d2 \oplus d3 \oplus d4 \oplus p8 = 0$. Answer parts (b) and (c) with single error correcting, double error detecting (SECDDED) code.

Please write down the calculation process in detail. Otherwise, you will get zero points.

- (a) (4 points) Let A = 1101100 be a 7-bit data. Show how to find and correct the single-bit error in A.
- (b) (4 points) Let B = 10110101 be an 8-bit data. Show how to find and correct/detect errors in B.
- (c) (4 points) Let C = 10001011 be an 8-bit data. Show how to find and correct/detect errors in C.

5. (10 points) Consider three processors with different cache configurations:

- Cache 1: Direct-mapped with four-word blocks
- Cache 2: Two-way set associative with two-word blocks
- Cache 3: Direct-mapped with one-word blocks

Assume the instruction miss rate is 4% and the data miss rate is 3%. Suppose a program with 70,000 instructions is executed on these processors, and one-third of the instructions contain a data reference. Assume the cache miss penalty is (5 + block size in words) and the CPI of this workload was measured to be 1.7 for the processor with Cache3.

Please write down the calculation process in detail. Otherwise, you will get zero points.

- (a) (6 points) Calculate the miss penalty and the number of cycles spent on cache misses for the processor with Cache 1 and the processor with Cache 2, respectively.
- (b) (4 points) Calculate the CPI for the processor with Cache 1 and the processor with Cache 2, respectively.

6. (18 points) Consider a 32-bit virtual address size with 64KB pages. And a 30-bit byte addressing physical memory with a physically indexed data cache of 64KB cache size and 256-Byte blocks. Assume the data cache and TLB are initially empty and the page table is shown below.

(VPN: virtual page number, PPN: physical page number)

VPN	Valid	PPN	VPN	Valid	PPN
0x0642	1	0x284e	0x682d	0	-
0x1a53	0	-	0x7aa6	1	0x3551
0x2b74	1	0x14ac	0x8e89	1	0x154c
0x361a	0	-	0x954a	1	0x3b94
0x48bb	1	0x0c84	0x9a84	0	-
0x56c3	1	0x251d	0xcd61	1	0x0795
0x6542	1	0x0ae6	0xd21e	1	0x141c

The CPU includes a fully associative TLB of 2 entries with the LRU replacement policy. The CPU data cache is direct-mapped. Data reads from the following virtual addresses are performed: 0x954a16c2, 0x6542c746, 0x954a1647, 0x6542c412, 0x2b74c4d3, 0x6542c46a, 0x954a16dd, 0x6542c417, 0x2b74c723

For every address, please write down if it is a hit in the cache, a hit in the TLB or a miss.

Virtual address	Physical address	Cache Hit/Miss	TLB Hit/Miss
0x954a16c2			
...			