Computer Architecture

Final Exam

Spring 2013

2013/06/13
03:30 P.M. – 04:45 P.M.

Total: 140 points

Read Me First

0. 답은 한글로 적어도 좋습니다**.**

1. If you cheat on exam, you will get an "F" for the course.

2. Some questions have several subproblems. Make sure that you put all the answers for the subproblems in one place on your answer sheet.

Visit the course homepage for the final exam scores and grades. The exact schedule will be announced on the homepage.

Problem 1. [20]

One of the key design principles of designing high-performance systems is *to make the common case fast*. Give two cases where this principle was successfully applied. Give your explanation based on the following framework:

1. The definition of the common case
2. How to improve the performance for the common case
3. Impact of the proposed improvement on non-common cases

Problem 2. [40]

(a) [5] When a processor is based on 21-bit addresses, if you are designing a 64-KB 4-way set-associative cache, how many bits out of the 21-bit address are allocated to the cache index field? Assume that the block size is 32 bytes.

(b) [10] Suppose a computer's address size is k bits (using byte addressing), the cache size is S bytes, the block size is B bytes, and the cache is A-way set associative. Assume that B is a power of 2, so $B = 2^b$. Express the number of bits needed to implement the cache in terms of S, B, A, b and k.

(c) [10] We are going to run the following C program fragment using a data cache U of a unknown configuration with 32-byte cache blocks:

```
#define SIZE (32 * 1024)
int A[SIZE], B[SIZE], C[SIZE];
for (i = 0; i <SIZE; i++)
        A[i] = B[i] + C[i];
```

Assume that `int` maps to a 4-byte data quantity, and the arrays `A`, `B`, and `C` are located at the address `0x66340000`, `0x66440000`, and `0x66540000`, respectively. If you would like to maximize the number of data cache misses for this program, what should be the configuration of the cache U? For the cache U, how many cache misses occur while executing this program fragment? Assume that we are using 32-bit addresses.

(d) [5] What is the main difference in design considerations between a primary cache and a second-level cache?

(e) [10] Cache associativity usually improves the miss ratio, but not always. Give a short series of address references for which a cache with a higher set associativity experiences more misses than a cache with a lower set associativity.

Problem 3. [30]

(a) [10] Explain how a Branch History Table works for resolving control hazards.

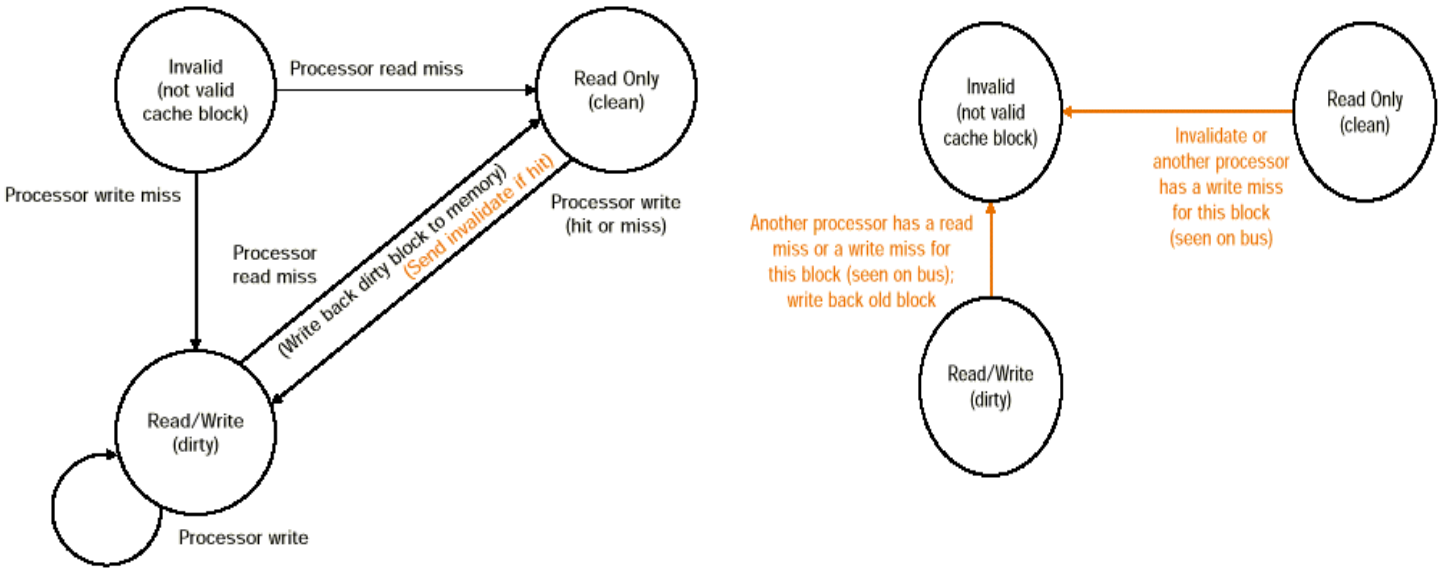
(b) [5/5] Argue if the following statement is true or false.

A. [5] A Branch Target Buffer can be used in the fetch stage.

B. [5] A Branch History Table can be used in the fetch stage.

(c) [10] Some pipelines employ multiple branch predictors, which can increase the design complexity. Explain why the design complexity increases when multiple branch predictors are used.

Problem 4. [20]



(a) [10] The above diagrams show state transitions for a write-invalidate cache coherence protocol based on a write-back policy. Assuming that you are using the above protocol with three processors, P0, P1, and P2, fill in the state for a cache block X in each of processors for the memory operations listed below. Use the same table for your answer.

Operations	P0	P1	P2
Initially	Read Only	Read Only	Read Only
(1) P2: loads X			
(2) P0: loads X			
(3) P1: stores X			
(4) P2: stores X			
(5) P1: loads X			

(b) [10] The above cache coherence protocol has a scaling problem with synchronization operations (e.g., lock/unlock). Explain what a scaling problem is in this context and give an example scenario using the multiprocessor assumed in (a).

Problem 5. [10]

Depending on whether collisions are allowed or not, we can classify bus arbitration techniques into two categories. Give one example technique from each category and explain how it works.

Problem 6. [20]

(a) [5] What is the main role of the TLB below?

(b) [5] What is the main role of the Ref bit below?

(c) [10] Explain what happens when a page fault exception is raised.