CS151B/EE116C - Homework #9, Due 6/6/2018 (NO LATE HOMEWORKS)

The final exam will be held in 5249 Boelter Hall on Thursday, June 14, 8:00am - 11:00am.

- □ During the exam you will be able to use: the course textbook (**5th Edition**), hard copy printout of pp. 318-340 from chapter 5 of the 3rd edition, the class notes posted on the class website, and a **simple calculator**. **No other material or devices can be used**. It will be **assumed** that you will have this material with you.
  Do not forget to bring the required material!
- □ You cannot use any material on which you have written your own notes. However, it is fine if all you have done is highlight or underline parts of otherwise permitted material.
- □ The exam will be comprehensive — it will cover all the material from the beginning of the quarter. You are ‘‘responsible’’ for all the reading assignments, the class notes, as well as for everything covered in lectures and discussion sections.

Reading Assignment:

- ‘‘Designing the Memory System to Support Caches’’ — pp. 471-473 from the 4th Edition of the Patterson and Hennessy book. Available in ‘‘Supplementary Readings’’ on the class web site.
- Virtual memory — Chapter 5: pp. 427-444, 452-461
- Multiprocessors and cache coherence — Chapter 6: pp. 519-520, Chapter 5: pp. 466-470.
- A clearer perspective on *consistency* and *coherence* —
  Daniel J. Sorin, Mark D. Hill, and David A. Wood, *A Primer on Memory Consistency and Cache Coherence*, Morgan & Claypool Publishers (2011).
  Read only pp. 1-2 (pages 17 and 18 of the PDF file).
  This online book is available in the ‘‘Supplementary Readings’’ part of the class website.

Problems:

(1) You are a member of the design team of a new computer system. Your task is to nail down the last details regarding the implementation of the cache.

Some decisions cannot be changed:

- The system is designed for programs where 65% of memory accesses are reads and 35% of memory accesses are writes.
- The block size is 16 bytes.
- The width of the bus between the cache and main memory is 32 bits.
- The cache access time is 6ns.
- The main memory access time is 65ns.
- The write policy is write-back.

The remaining question has to do with the handling of a write-miss. Specifically, on a write-miss there are two options: (I) The main memory is updated but the accessed block is **not** copied to the cache (*write-around*). (II) The block is moved to the cache and is updated there (*write-allocate*).

If option (I) is chosen, the hit rate will be 90% and, on average, 70% of the blocks in the cache will be dirty. If option (II) is chosen, the hit rate will be 95% and, on average, 75% of the blocks in the cache will be dirty.

In order to choose between options (I) and (II), **you must compute** the effective access time for each case and pick the design that will result in a lower access time.

Explain any assumption you need to make. Explain briefly the computations you make, i.e., what is the justification for the way you are computing the effective access time. Check your calculations carefully — a correct decision requires correct computations.

(2) 5.11.4 in the book.
Note: Each one of the applications utilizes half of its maximum possible virtual address space.

(3) 5.11.5 in the book.
Note: The first level table has 256 entries. Each one of the applications utilizes half of its maximum possible virtual address space.

(4) A virtual memory has a page size of 4096 (2^{12}) words. There are 64 pages of virtual address space but only 5 (five) page frames in real memory. The page table is stored in a special memory module and does **not** take up space in real memory. The program issues the following sequence of addresses (shown here in hex):

```
2f012, d120, 550, d58b, 7194, 30000, 7052, 5550, 2fa02, 744, 7276
```

Before the program begins execution, the real memory is "empty."

A) Assume that FIFO replacement is used. Circle the references that will cause a page fault.

```
2f012, d120, 550, d58b, 7194, 30000, 7052, 5550, 2fa02, 744, 7276
```

B) Assume that LRU replacement is used. Circle the references that will cause a page fault.

```
2f012, d120, 550, d58b, 7194, 30000, 7052, 5550, 2fa02, 744, 7276
```

(5) A computer system has the following characteristics:

- The memory is byte addressable.
- The disk address where a virtual page is stored is the virtual page number.

• Processor:	60% of memory accesses are reads The processor produces 30 bit addresses.
• Real memory:	Size: 128 Mbytes. Access time: 110 nano-seconds. The width of the bus to memory is 64 bits. Page size is 16K bytes.
• Cache:	The cache is accessed using physical addresses. Size: 512K bytes. Access time: 9 nano-seconds. Cache block size is 64 bytes. Organization: eight-way set-associative. Cache hit rate: 94%. Write policy: write-back, write-around
• TLB:	Size: 256 entries. Access time: 5 nano-seconds. Organization: direct mapped. TLB hit rate: 98%.

A) Assume that the page fault rate is 0.1% and the average disk access time is 6 milli-seconds. Do you now have all the information you need to compute the overall average effective access time as seen from the processor? If not, specify what additional information is needed. (Note, in this part you are not asked to actually compute an effective access time).

B) Assume that the page fault rate is 0. Compute the effective access time to the memory system. Clearly specify any assumptions you make.

(6) Consider a byte-addressable virtual memory system with the following properties:

- 64 bit virtual addresses
- 8 KiB pages
- 34 bit physical addresses
- A sixteen-way set associative TLB that holds 1024 page-table entries.
- The page table is organized as a **four** level table.

Compute the **total** amount of storage required for the TLB. Your answer should be the total number of **bits** required for **all** the storage needed for the TLB.
State every assumption you make and explain the calculation.

(7) Consider a shared-memory multiprocessor, where each processor has a local cache. The cache write policy is write-back, write-allocate. Each cache block contains two words. Elements X[0] and X[1] of array X are in the same block. Initially, X[0]=X[1]=0. Then, the two processors simultaneously execute the following C code:

P1	P2
`X[0]++ ; print X[1] ;`	`X[1] += 2 ; print X[0] ;`

A) There are several possible results from the execution of this code. Assume that cache coherence is maintained. List **all** the possible values that may be printed by this program. Your answer must be a list of **pairs**, where the first element of each pair is the value printed by P1 and the second element of each pair is the value printed by P2. Explain your answers.

B) Assume that cache coherence is not maintained. List at least one possible value of the cache block that is an outcome of executing of the code above and is not on the list that you produced in your answer to part A. As with part A, the answer here is at least one pair. Explain your answer.

C) Assume that cache coherence is maintained using a simple write-invalidate snoopy protocol. Before execution begins, all the block frames in the caches of both processors are invalid. What is the minimum number of cache misses, in both caches, that will occur during the execution of the program. Explain your answer.

D) Repeat part C but now determine the maximum number of cache misses, in both caches, that will occur during the execution of the program. Explain your answer.

Practice problems: You do not need to hand in a solution to the problems below.

(8) Page tables require large amounts of memory (as described in the Elaboration on page 436), even if most of the entries are invalid. As described on page 436 and in the class notes pp. 11.13-11.14, one solution is to use a hierarchy of page tables. One way to arrange such a scheme is to have the second-level page tables occupy exactly one page of memory. Assuming a 32-bit virtual address space with 4 KB pages and 4 bytes per page table entry, how many bytes will each program need to use to store the first-level page table (which must always be in memory)? Provide a figure similar to Figures 5.27 (page 433) that shows how the mapping of virtual to physical addresses is done.

(9) Consider a byte-addressable virtual memory system with the following properties:

- 42 bit virtual addresses
- 8 KB pages
- 30 bit physical addresses
- An four-way set associative TLB with a total of 256 entries.

A) The page table is organized as a two level table. The first level table contains 32K entries. Compute the size, in bits, of a single second level table. Take into account all the information that must be stored in the page table. Explain your answer and specify any assumptions you make.

B) Show a diagram of the TLB which will clearly indicate how it is accessed starting from the virtual address produced by the processor. Compute the total size, in bits, of the TLB. Take into account all the information that must be stored in the TLB. Explain your answer and specify any assumptions you make.

(10) Consider a shared-memory multiprocessor, where each processor has a local cache. The cache write policy is write-through, write-allocate. Each cache block contains two words. Elements X[0] and X[1] of array X are in the same block. Similarly, elements X[2] and X[3] are in the same block and elements X[4] and X[5] are in the

same block. Initially, X[0]=X[1]=X[2]=X[3]=X[4]=X[5]=0. Then, the two processors simultaneously execute the following C code:

P1	P2
`X[2] = X[0]+3 ;`	`X[3] = X[2]+5 ;`
`X[2] = X[4] ;`	`X[1] = 7 ;`
`X[1]++ ;`	`X[4] = 20 ;`

A) There are several possible results from the execution of this code. Assume that cache coherence is maintained. List at least three possible values of the three cache blocks containing array elements X[0] through X[5] Your answer must be a list of **6-tuples**, where the first element of each 6-tuple is the value of X[0], the second is the value of X[1], etc. Explain your answers.

B) Assume that cache coherence is not maintained. List at least one possible value of the three cache blocks that is an outcome of executing of the code above and is not possible if coherence is maintained. As with part A, the answer here is at least one 6-tuple. Explain your answer.

C) Assume that cache coherence is maintained using a simple write-invalidate snoopy protocol. Before execution begins, all the block frames in the caches of both processors are invalid. What is the minimum number of cache misses, in both caches, that will occur during the execution of the program. Explain your answer.

D) Repeat part C but now determine the maximum number of cache misses, in both caches, that will occur during the execution of the program. Explain your answer.