CSCI 4717 – Memory Hierarchy and Cache Quiz

General Quiz Information

- This quiz is to be performed and submitted using D2L.
- This quiz is to be completed as an ***individual, not as a team.***
- Since I will not be present when you take the test, be sure to keep a list of all assumptions you have made and e-mail them to me (tarnoff@etsu.edu) ***immediately*** after completing the test. This is not a requirement. Only do this if you think one or more of your answers requires additional explanation.

1. Of the following types of memory access methods, for which can you accurately predict the amount of time it will take to receive the data after requesting it? Check all that apply.

   a.) Sequential b.) Direct c.) Random d.) Associative

   The access time for sequential depends on how far away the data is from the current head position (e.g., tape). The access time for direct depends both on how far away the head is from the track and how far away the head is from the sector (e.g., hard drive). Both of these are ***not predictable***.

   Random access uses a decoder to instantly access the location regardless of the current state of the memory (e.g., SRAM). Associative uses a standard "compare circuit" which uniquely and quickly identifies the location of the data (e.g., cache). Both of these are consistent and predictable.

2. True or false: By having the cache act as a bridge between main memory and the CPU, the time it takes to retrieve a block of data from main memory is longer than it would be if the cache did not exist.

   If the cache is functioning as a bridge, all accesses to main memory must be processed through it first. This "processing" adds delay to the accessing of main memory and hence is slower than it would have been without the cache. This is only for accesses to main memory. The speed up realized through the addition of a cache more than makes up for the extra time it takes to access main memory only when needed.

3. True or false: Typically, a split cache divides both the L1 and L2 caches into an instruction-only cache and a data-only cache.

   Okay, so this was a bit of a trick question. ONLY L1 caches are divided into instruction and data caches. A single L2 cache should feed both L1 caches. See the block diagrams of both the Pentium and PowerPC to see how the L1 instruction cache feeds the instruction unit while the L1 data cache feeds the data units.

4. Assume a memory access to main memory on a cache "miss" takes 30 ns and a memory access to the cache on a cache "hit" takes 3 ns. If 80% of the processor's memory requests result in a cache "hit", what is the average memory access time?

   a.) 8.4 nS b.) 33 nS c.) 24.6 nS d.) 27.0 nS e.) 2.4 nS f.) 3.0 nS

   Since we know that not all of the memory accesses are going to the cache, the average access time must be greater than the cache access time of 3 ns. This eliminates answers 'e' and 'f'. Similarly, not all accesses are going to the main memory, so the average must be less than 30 ns eliminating 'b' as an answer. For the exact answer, you need to see that 80% of the time, the access will be 3 ns while the rest of the time (20%) the access time will be 30 ns. You should be able to see then that the average is going to be closer to 3 ns than 30 ns meaning the answer is a. You could also see this with the following equation.

   $(0.8 \times 3 \text{ ns}) + (0.2 \times 30 \text{ ns}) = 2.4 \text{ ns} + 6 \text{ ns}$
   $= 8.4 \text{ ns}$

The following cache represents a 2-way set associative cache, i.e., there are two lines per set. Notice that the set ID values start at 01101101_2 and increment every other row. This is meant to imply that you are looking at a group of lines/sets toward the middle of the cache and not the entire cache. There are 14 bits for the tag, 8 bits for the set id, and 2 bits for the word id. Answer the following 3 questions based on this cache.

Tag (binary values)	Set ID (binary values)	Word within block			
		00	01	10	11
10100001001001	01101101	00_{16}	61_{16}	$C2_{16}$	23_{16}
11100001100100	01101101	10_{16}	71_{16}	$D2_{16}$	33_{16}
11001011010110	01101110	20_{16}	81_{16}	$E2_{16}$	43_{16}
11100101101011	01101110	30_{16}	91_{16}	$F2_{16}$	53_{16}
11110110110100	01101111	40_{16}	$A1_{16}$	02_{16}	63_{16}
10100111010101	01101111	50_{16}	$B1_{16}$	12_{16}	73_{16}
10101010111110	01110000	84_{16}	$E5_{16}$	46_{16}	$A7_{16}$
10101010010011	01110000	94_{16}	$F5_{16}$	56_{16}	$B7_{16}$
01110001001000	01110001	$A4_{16}$	$A5_{16}$	66_{16}	$C7_{16}$
00001101101101	01110001	$B4_{16}$	15_{16}	76_{16}	$D7_{16}$
01011010010010	01110010	$C4_{16}$	25_{16}	86_{16}	$E7_{16}$
10101111001011	01110010	$D4_{16}$	35_{16}	96_{16}	$F7_{16}$

5. A copy of the data from memory address 7121C5 (hex) is contained in the portion of the cache shown above. Enter the value that was retrieved from that address in the space below as a two-digit hexadecimal number with no base identification, e.g., 0x4F (hexadecimal 4F) should be entered as 4F. _____

   First, convert 7121C5 to binary. A simple binary conversion using MS Calculator should do this just fine.

   $$7121C5_{16} = 011100010010000111000101_2$$

   The last two bits, 01, identify the word position within the block. 01 means that it is in the second column of data.

   The next eight bits, 01110001, should identify the set. 01110001 identifies the set consisting of the third and fourth rows from the bottom.

   The fourth row from the bottom has the matching tag of 0111000100100. Therefore, the data is in the second column of the fourth row from the bottom, i.e., **A5**.

6. How many lines are contained in this cache?

   a.) 8 b.) 256 c.) 512 d.) 1024 e.) 16K f.) Cannot be determined

   Since there are eight bits identifying the set, then there are $2^8 = 256$ sets. Since there are two lines per set, then there are $2 \times 256 = 512$ lines.

7. How many blocks are contained in the memory space (not the cache, but the memory) of the cache system defined above? (Due to D2L constraints, the notation used to represent 2 raised to the power of n is 2^n)

   a.) 2^24 b.) 2^22 c.) 2^14 d.) 2^8 e.) 2^2 f.) Cannot be determined

   Remember that a block is a division of memory. The number of words in a block is identified by the number of bits set aside for the word id. Since there are 24 bits in the address and two of them are for the word id, then 22 bits are left for the block id. This means that memory has 2^{22} = 4 Meg blocks.

8. Which is the fastest cache mapping function?

   a.) Direct mapping b.) Set associative mapping c.) Fully associative mapping

9. Which cache mapping function is ***least*** likely to thrash, i.e., it has the lowest chance of two blocks contending with each other to be stored in the same line?

   a.) Direct mapping b.) Set associative mapping c.) Fully associative mapping

10. Which cache mapping function does not require a replacement algorithm?

    a.) Direct mapping b.) Set associative mapping c.) Fully associative mapping

11. True or false: The number of lines contained in a set associative cache can be calculated from the number of bits in the memory address, the number of bits assigned to the tag, the number of bits assigned to the word id (identifying the number of words per block), and the number of bits assigned to the set id (identifying the number of sets.)

    This is false. The number of bits in the memory address, the number of bits assigned to the tag, the number of bits assigned to the word id, and the number of bits assigned to the set id can only tell you how many sets there are. You need to know the number of lines per set in order to calculate the number of lines.

12. Which cache write mechanism creates more bus traffic?

    a.) Write through b.) Write back

13. Which cache write mechanism allows an updated memory location in the cache to remain out of date in memory until the block containing the updated memory location is replaced in the cache?

    a.) Write through b.) Write back c.) Both d.) Neither

14. Which cache write mechanism best supports bus watching for use with multi-processor systems?

    a.) Write through b.) Write back

15. Which cache write mechanism best supports hardware transparency for use with multi-processor systems?

    a.) Write through b.) Write back c.) It makes no difference

    I accepted either answer.