COMP 222 Quiz #2 Solutions

Problem 1: General Size Parameters
Express these values as powers of 2.

- RAM: 4 GB = _4 X 1 G = 2^2 X 2^{30} = 2^{32}_
- Block/Line: 64 B = _64 X 1 = 2^6_
- Cache: 2 MB = _2 X 1 M = 2^1 X 2^{20} = 2^{21}_

Several of you wrote that 4 GB = 2^{30} X 2^{30} X 2^{30} X 2^{30} = 2^{120}. Unfortunately, multiply by 4 and raise to the 4th power are different operations. Multiplication is repeated addition, and exponentiation is repeated multiplication. It is true that 4 GB = 2^{30} + 2^{30} + 2^{30} + 2^{30} (where + means addition), but this expression doesn't really get you any closer to the answer.

Example #1: Multiply 1 GB by 4
4 X 2^{30} = 2^2 X 2^{30} = 2^{30+2} = 2^{32}
If we multiply together several terms with same base and different exponents, we add the exponents. More generally, a^b X a^c = a^{b+c}

Example #2: Raise 1GB to the 4th Power
(1 G)^4 = (2^{30})^4 = 2^{30} X 2^{30} X 2^{30} X 2^{30} = $2^{30 X 4}$ = 2^{120}
If we have a base raised to an exponent and we raise that expression to another exponent, then we multiply the exponents. More generally, (a^b)^c = $a^{b X c}$

Example #3: How big is 2^{120}?
Using the approximation 2^{10x} = 10^{3x}, 2^{120} = 2^{10X12} = 10^{3X12} = 10^{36} or 1 followed by 36 zeros, 1,000,000,000,000,000,000,000,000,000,000,000,000. According to the Wikipedia article "Names of Large Numbers," 10^{36} = 1 undecillion. More exactly, 2^{120} = 1,329,227,995,784,915,872,903,807,060,280,344,576. Many existing boards have a 32-bit or 64-bit address bus, but 120-bit bus processors are not common at the present time.

Problem 2. Refer to sizes in Problem 1.

- # of bits in a full RAM address: _32_
- # of RAM blocks: _2^{32} bytes / 2^6 bytes per block = 2^{32-6} = 2^{26} blocks_
- # of Cache lines: _2^{21} bytes / 2^6 bytes per line = 2^{21-6} = 2^{15} lines___
- RAM address breakdown: _26_ bits block index | _6_ bits byte index

Problem 3. Use sizes from Problems 1 and 2 and assume Direct Mapped Cache

- # of RAM blocks that map to each cache line: _$2^{26}/2^{15} = 2^{11}$_
- Address breakdown: _11_ bits tag | _15_ bits line index | _6_ bits byte index
- Associative search: when looking up an address tag, how many stored tags must it be compared to? _1_

Address lookup for direct mapped:

- ***CPU issues a 32-bit address = 26 bits RAM block index + 6 bits byte index.***
- ***Before fetching the data from RAM, we check to see if a copy is in cache.***
- ***The 26 bit block index is divided into 11 bit tag and 15 bit cache line index.***
- ***We use the 15 bit cache line index to select that single cache line.***
- ***We compare the address's 11 bit tag with the tag stored at the selected cache line.***
- ***If the tags match, the cache line contains the data we are looking for.***

Problem 4. Use sizes from Problems 1 and 2 and assume Fully Associative Cache

- # of RAM blocks that map to each cache line: _2^{26}_
- Address breakdown: _26_ bits tag | _0_ bits line index | _6_ bits byte index
- Associative search: when looking up an address tag, how many stored tags must it be compared to? _2^{15}_

Address lookup for fully associative:

- ***CPU issues a 32-bit address = 26 bits RAM block index + 6 bits byte index.***
- ***Before fetching the data from RAM, we check to see if a copy is in cache.***
- ***The 26 bit block index becomes a 26 bit tag with 0 bits used for cache line index. There is no preferred cache line to check, the RAM block we're looking for could be stored in any line in the entire cache.***
- ***Since there is no single cache line to check, we must compare the address's 11 bit tag with every tag in every cache line – all 2^{15} of them. This search is done associatively (in parallel) for performance reasons.***
- ***If the address tag matches any of the 2^{15} stored tags, the associative search will report that a match was detected for cache line x. The line with the matching tag will contain the data we are looking for.***

Problem 5. Use sizes from Problems 1 and 2 and assume 16-Way Set Associative Cache

- # of Cache sets: _2^{15} lines / 2^4 lines per set = 2^{11} sets_
- # of lines per Cache set: _2^4 = 16_
- # of RAM blocks that map to each cache set: _$2^{26}/2^{11} = 2^{15}$_
- Address breakdown: _15_ bits tag | _11_ bits set index | _6_ bits byte index
- Associative search: when looking up an address tag, how many stored tags must it be compared to? _2^4 = 16_

Address lookup for 16-way set associative:

- ***CPU issues a 32-bit address = 26 bits RAM block index + 6 bits byte index.***
- ***Before fetching the data from RAM, we check to see if a copy is in cache.***
- ***The 26 bit block index is divided into 15 bit tag and 11 bit cache set index.***
- ***We use the 11 bit set index to select that set. Like all sets, it contains 16 lines.***
- ***The RAM block we are looking for must be in this set, but within the set it could be in any one of the 16 lines.***
- ***We compare the 15 bit address tag with the 16 stored tags in the set. This search is done associatively (in parallel) for performance reasons.***
- ***If the address tag matches any of the stored tags in the set, the associative search will report that a tag match was detected on line x. A copy of the RAM block we are looking for is in that line with the matching tag.***