

**Homework Cover Page**

|  |  |
| --- | --- |
| Name: Brandon Hoang | Date: 03/20/19 |
| Course: EGCP-381 | HW #: 4 |

Grading Criteria:

|  |  |  |
| --- | --- | --- |
| **Problem #** | **Earned Points** | **Possible Points** |
| RQ4.1 |  | 2 |
| RQ4.2 |  | 2 |
| RQ4.3 |  | 2 |
| RQ4.4 |  | 2 |
| RQ4.5 |  | 2 |
| RQ4.6 |  | 2 |
| RQ4.7 |  | 2 |
| RQ4.8 |  | 2 |
| RQ4.9 |  | 2 |
| P4.4 |  | 6 |
| P4.14 |  | 2 |
| Total: | 0 | 26 |

**PLEASE UPLOAD YOUR REPORT IN TITANIUM. NO PAPER REPORTS.**

Professor Comments:

**4.1 What are the differences among sequential access, direct access, and random access?**

*In sequential, memory is arranged into records, and access must be made in a linear sequence. There is a shared read and write mechanism. Direct access also has this mechanism, but each record has a unique address relative to it's physical position. Random access has the same specific addressing of Direct, but access time is independent of prior accesses and is thus constant, where Direct access has variable access times.*

4**.2 What is the general relationship among access time, memory cost, and capacity?**

*Generally speaking, a faster access time necessitates greater memory costs, greater capacity decreases the cost of per bit and access time.*

**4.3 How does the principle of locality relate to the use of multiple memory levels?**

*Data can be organized such the percentage of access to each successively lower level of the hierarchy is significantly lower than the level above it.*

**4.4 What are the differences among direct mapping, associative mapping, and set associative mapping?**

*Direct mapping maps each block into only one cache line, and has the distinct disadvantage of being of having fixed cache locations. Associative mapping permits mapping of memory blocks to any line in cache, thereby addressing direct mapping's flaw. Set Associative compromises between the above two methods by interpreting the memory address as Tag, Set, and Word fields rather than just the Tag and Word or Tag, Line, and Word.*

**4.5 For a direct-mapped cache, a main memory address is viewed as consisting of three**

**fields. List and define the three fields.**

*Tag: Used to distinguish blocks that can fit in the line.*

*Line: Identifies line of the cache*

*Word: identifies unique word or byte in block.*

**4.6 For an associative cache, a main memory address is viewed as consisting of two fields.**

**List and define the two fields.**

*Tag: Same as a direct map, identifies a unique block of memory*

*Word: Uniquely points to a word/byte in a block.*

**4.7 For a set-associative cache, a main memory address is viewed as consisting of three**

**fields. List and define the three fields.**

*Tag: Used to identify blocks of main memory with Set bits combined.*

*Set: Identifies one of the sets*

*Word: Corresponds to a word/byte in a block*

**4.8 What is the distinction between spatial locality and temporal locality?**

Spatial locality refers to the execution tendency of a processor to access instruction sequentially. Temporal locality is the tendency for a processor to access memory locations that have been used recently.

**4.9 In general, what are the strategies for exploiting spatial locality and temporal locality?**

*Temporal locality is exploited by keeping recently used data and instructions in the cache then using cache hierarchy. Spatial locality is optimally used by using larger cache blocks, and using prefetching in cache control logic.*

***4.4 List the following values:***

***a. For the direct cache example of Figure 4.10: address length, number of addressable units, block size, number of blocks in main memory, number of lines in cache,***

***size of tag***

*Address length: s + w = (S = tag + line = 22 / w = 2) 22+ 2 = 24 bits*

*Number of addressable units: 2^s+w = 2^24 bytes*

*Block size: 2^2 = 4*

*blocks in main memory: 2^S = 2^22*

*lines in cache: 2^r (r = line) = 2^14*

*Size of tag: s-r = 22-14 = 8 bits*

*b.* ***For the associative cache example of Figure 4.12: address length, number of addressable units, block size, number of blocks in main memory, number of lines in***

***cache, size of tag***

Address length: s+w = (S = tag = 22 / w=2) 22 + 2 =24 bits

Addressable Units: 2^ s+W = 2^24 bytes

Block Size:2^2 = 4

Number of blocks in main memory: 2^s = 2^22

Number of lines in cache: undefined\*\*\*

Tag size: s = 22

***c. For the two-way set-associative cache example of Figure 4.15: address***

***length, number of addressable units, block size, number of blocks in main***

***memory, number of lines in set, number of sets, number of lines in cache, size***

***of tag***

*Address length: s + w = (tag + set = 22 / w = 2) = 24 bits*

*Addressable Units:2^s+w = 2^24 bytes*

Size of Block: 2 ^ w = 2^2 = 4

Blocks in main memory:2^s = 2^22

Lines in cache: (size of block = size of line = 2^2) k = 2^13 / 2^2 = 2^11 => K \*2^d = 2^11 \* 2^13 = 2^24

Tag size: 2^ (s-d) = 2(22-13) = 2^ 9

**4.14 Consider again Example 4.3. How does the answer change if the main memory uses a**

**block transfer capability that has a first-word access time of 30 ns and an access time**

**of 5 ns for each word thereafter?**

Write-Back

Write-Through

If the first word access time is 30 ns and subsequent access is 5 ns per word, then the average line must be must be written at least 3 times to make Write-back more efficient.