CS352: Computer Architecture

Spring 1998

HOMEWORK 3 - SOLUTIONS

1. In a two-level memory hierarchy, let $t_1 = 10^{-7}$s and $t_2 = 10^{-2}$s. If t_a denotes the average access time of the memory hierarchy, and if we define the access efficiency to be equal to t_1/t_a, then what must the hit ratio H be in order for the access efficiency to be at least 80% of the maximum value?

   Solution: For a 2-level hierarchy, the average access time is given by:

   $$t_a = t_1 + (1 - H) * t_2$$

   where H is the hit ratio, t_1 is the hit time, and t_2 s the miss penalty. Note that when on a cache miss, the amount of time it takes to service the memory access request is $(t_1 + t_2)$, and hence, the miss penalty is t_2.

   Hence, we want to find H such that the access efficiency $t_1/t_a \geq 0.8$. That is,

   $$t_1/t_a \geq 0.8 \Rightarrow t_1 \geq 0.8 * (t_1 + (1 - H) * t_2)$$

   $$\Rightarrow 0.2t_1 \geq 0.8 * (1 - H) * t_2$$

   $$\Rightarrow H \geq 1 - \frac{t_1}{4 * t_2}$$

   Substituting the values for t_1 and t_2, you can get:

   $$H \geq 0.9999975$$

2. In an n-level memory hierarchy, let H_i denote the hit ratio associated with the memory M_i at level i. If $t_1, t_2, ..., t_n$ denote the access times of the n levels of the memory hierarchy, what is the average access time a memory request?

   Solution: For a 2-level hierarchy, the average access time for a memory request is:

   $$t_a = t_1 + (1 - H_1) * t_2$$

   For a 3-level hierarchy, this equation generalizes to:

   $$t_a = t_1 + (1 - H_1) * \{t_2 + (1 - H_2) * t_3\}$$

   Note that $t_2 + (1 - H_2) * t_3$ is the average time to access data from the 2nd and the 3rd level of the memory hierarchy.

   In general, for a n-level hierarchy, we get:

   $$t_a = t_1 + (1 - H_1) * \{t_2 + (1 - H_2) * \{t_3 + (1 - H_3) * ....\{t_{n-1} + (1 - H_{n-1}) * t_n\}\}\}...\}$$

   $$\Rightarrow t_a = t_1 + \sum_{i=2}^{n} \prod_{j=1}^{n-1} (1 - H_j) * t_i$$

3. Consider a 3-level memory hierarchy as shown below:

Level i	Access time t_i time t_i (sec)	Access Probability p_i	Block transfer time b_i (sec)
M_1	10^{-6}	0.999900	0.001
M_2	10^{-5}	0.000099	0.1
M_3	10^{-3}	0.000001	-

Here p_i denotes the fraction of memory access requests that are satisfied at level M_i of memory. When a miss occurs at level M_i, a block swap takes place between memory levels M_i and M_{i+1}; the average time for this block swap is b_i.

(a) Calculate the average access time t_a for the memory hierarchy.
Solution: The average access time for a 3-level hierarchy is:

$$t_a = t_1 + (1 - H_1) * MissPenalty_1$$

where

$$MissPenalty_1 = (t_2 + b_1) + (1 - H_2) * MissPenalty_2$$

and

$$MissPenalty_2 = (t_3 + b_2)$$

Also, $H_1 = 0.9999$ and $H_2 = 0.000099/0.000100 = 0.99$. Using these values and the above equations, we get $MissPenalty_2 = 0.101$, $MissPenalty_1 = 0.00202$, and hence

$$t_a = 10^{-6} + 10^{-4} * 0.00202 = 1.202 * 10^{-6}$$

(b) It is desired to make $t_a \leq 1.15 \times 10^{-6}$s. This speedup is to be achieved by replacing M_3 by a faster memory that reduces b_2 to a new value b_2'. What should b_2' be?
Solution: If you substitute $t_a = 1.15 * 10^{-6}$ in the above equations, then given the values of t_1, t_2, t_3, b_1, H_1, and H_2, you can calculate the value of b_2'. The Solution is $b_2' = 0.048$.

4. Consider a system containing a 128-byte cache. Support that set-associative mapping is used in the cache, and that there are four sets each containing four cache blocks. The physical address is 32 bits, and the smallest addressable unit is a byte.

(a) Show how the memory address is used for cache addressing?
Solution: Since the cache contains 4 sets, each with 4 blocks, the total number of blocks in the cache is 16. Also, since the total cache size is 128 bytes, each cache block contains 128/16 = 8 bytes. Hence, of the 32 bit address, you will need to use the least significant 3 bits to identify a byte within a selected block.
Of the remaining 29 bits in the address, you will need to use the least significant 2 bits to identify one of the 4 sets in the cache. The remaining 27 bits constitute the tag.

(b) To what cache blocks can the address $000010AF_{16}$ can be assigned?
Solution: Since $A = 1010_2$ and $F = 1111_2$, the least significant 5-bits of the address $000010AF_{16}$ are 01111. Since the last 3 bits are used to identify the byte being accesses and the first two bits identify the set, address $000010AF_{16}$ can be assigned to set "01".

(c) If the addresses $000010AF_{16}$ and $FFFF7Axy_{16}$ can be simultaneously assigned to the same cache set, what values can the address digits x and y have?

Solution: As we have shown above, to be assigned to the same cache set, the least significant 4^{th} and 5^{th} bit of the address must be identical. For address $000010AF_{16}$, the least significant 4^{th} and 5^{th} bits are 01. Hence, x must be such that its last bit is 0, and y must be such that its first bit is 1. Hence, x can be one of 0, 2, 4, 6, 8, A, C, or E; while y can be 8, 9, A, B, C, D, E, or F.

5. Consider a fully associative cache of size 2 blocks. Consider a program that accesses memory blocks in the following order:

a b a c a b d b a c d

(a) What is the number of cache misses that will result if the cache uses FIFO replacement policy?

Solution: For a fully associative cache of size 2 blocks, assuming the FIFO replacement policy, here is what the access sequence will look like.

$$\widehat{a}, \widehat{b}, a, \widehat{c}, \widehat{a}, \widehat{b}, \widehat{d}, b, \widehat{a}, \widehat{c}, \widehat{d}$$

where $\widehat{x}$ denotes a cache miss that occurs while accessing location x. Hence, with FIFO, the access sequence will see 9 misses.

(b) How does this change if the LRU policy is used?

Solution: With LRU policy, the access sequence will look like the following:

$$\widehat{a}, \widehat{b}, a, \widehat{c}, a, \widehat{b}, \widehat{d}, b, \widehat{a}, \widehat{c}, \widehat{d}$$

Hence, LRU will see 8 misses.

(c) How will both of these results change if the cache size is increased to 3 blocks?

Solution: With FIFO, the access sequence is:

$$\widehat{a}, \widehat{b}, a, \widehat{c}, a, b, \widehat{d}, b, \widehat{a}, c, d$$

Hence, there will be 5 misses. Note that the access to d causes a miss, and hence the replacement of a from the cache. Hence, the subsequent access to a causes a miss.

With the LRU policy, the access sequence is:

$$\widehat{a}, \widehat{b}, a, \widehat{c}, a, b, \widehat{d}, b, a, \widehat{c}, \widehat{d}$$

Here, the total number of misses are 6. Note also that in this case, when access to d causes a miss, c is replaced from the cache (since it is the least recently used memory location). Consequently, a subsequent access to c causes a miss and the replacement of d, which at that time is the least recently used memory location. Hence, the final access to d also causes a miss.

(d) Intuitively, both FIFO and LRU would seem to be better than a *most recently used (MRU)* replacement scheme. How does MRU perform on the access trace shown above assuming a cache of size 2 blocks?

Solution: Assuming a cache of size 2 blocks and the MRU replacement policy, the access pattern will be as follows:

$$\widehat{a}, \widehat{b}, a, \widehat{c}, \widehat{a}, b, \widehat{d}, \widehat{b}, a, \widehat{c}, \widehat{d}$$

That is, there will be 8 misses, which is smaller than FIFO, and the same as LRU.

6. Consider the following code fragment:

1. a ← b + c;
2. b ← a + d;
3. a ← e + f;
4. x ← a + b;
5. y ← c + d;
6. c ← e + f;

Consider a system that employs Tomasulo's algorithm to issue and execute requests. Assume that the system fetches/decodes at most one instruction in each cycle. Let the system contain a 2-stage integer ALU (i.e., the execution stage of an integer ALU operation takes 2 clock cycles). What is the order in which requests will complete? Show your work. What is the total number of clock cycles required to execute these instructions?

Solution: If we can represent the state of the system using 4 states: fetch, reservation station (RS), ALU-execute-1, and ALU-execute-2, then the following table represents the execution sequence:

Clock Cycle	FETCH	RS	ALU-Execute-1	ALU-Execute-2
1	1			
2	2		1	
3	3	2		1
4	4	3	2	
5	5	4	3	2
6	6	4	5	3
7		6	4	5
8			6	4
9				6

Hence, the instructions will finish in the order 1, 2, 3, 5, 4, and 6.