Assignment 4
Due: Tuesday, December 3, 2013

- **Show all your work clearly and legibly for all your answers.**
- You are required to answer all 8 questions.
- The assignment (written or printed) must be stapled before handing in.
- You may hand in your assignment
  - to the instructor on due date by end of the class or
  - deposit it on hand-in lockers on 4th floor of the E2 building besides the elevator by 430 pm on the due date
- The numbers under [ ] indicate the marks allocated to each question.

1. [5] Describe the general characteristics of a program that would exhibit very little temporal and spatial locality with regard to data accesses.

Solution Q1: [5]

a. Little temporal locality for data means accessing variables only once.

b. Little spatial locality for data means no marching through arrays; data is scattered.

2. [10] Consider a series of word address references 1, 4, 8, 5, 20, 17, 19, 56, 9, 11, 4, 43, 5, 6, 9, 17. Assuming a direct-mapped cache with **16 one-word blocks** that are initially empty

a. label each reference in the list as a *hit* or a *miss* and fill the respective row for Tag and Index in Table 1

b. show the final contents of the cache by filling in the Address column in Table 2

Solution Q2:

The following information is good for questions 2-6.
Translating the given word address into binary could be done in three ways.

1.

a. Translate the given word address into binary; the lowest "m" bits will be used to select one of the 2^m words; next (left) lower "n" bits will determine the index and rest will be tag. That means, you can shift the lowest m bits to the right (in other words ignore lower m bits for these questions) and remaining bits you are left with are for index and tag.

b. Multiply the word address by 4 to get the byte address and translate the resulting value into binary. Then allow 2 bits (lowest bits) for byte-offset and

next (left) "m" bits for the word (one among 2^m words) selector and next (left) "n" bits for the index and rest for the tag.

Note: converting the word address to byte address basically adds two more bits, which are meant for byte-offset, that are already taken care of in method (1a) above and hence no need to allot bits for byte-offset

2. Block number=(block address) mod (number of blocks)

= (word address/number of words) mod (number of blocks)

For example,

=(56/1) mod (16) (for a 16 1-word cache block)
= 56 mod 16
= 8 (and quotient =3)
Therefore, tag=3 (quotient) and index =8 or
tag=11 and index=1000
The address in binary is 111000

Another example: For word address 135 in a 8 four-word cache block
Block number = (135/4) mod (8) (for a 8 four-word cache blocks)
=33 mod (8)
= 1 (and quotient =4) (tag=100; index =001)
Therefore, the word address in binary is 100 001 11 (the least significant 2 bits is the binary for the reminder in the division operation 135/4)

Solution Q2

a) Table 1: Hit/Miss table:

Reference	Hit or Miss	Tag	Block #	index
1	Miss	00	1	0001
4	Miss	00	4	0100
8	Miss	00	8	1000
5	Miss	00	5	0101
20	Miss	01	4	0100
17	Miss	01	1	0001
19	Miss	01	3	0011
56	Miss	11	8	1000
9	Miss	00	9	1001
11	Miss	00	11	1011
4	Miss	00	4	0100
43	Miss	10	11	1011
5	Hit	00	5	0101
6	Miss	00	6	0110
9	Hit	00	9	1001
17	Hit	01	1	0001

b) Final contents of the cache Table 2

Block Number	Address
0	
1	17
2	
3	19
4	4
5	5
6	6
7	
8	56
9	9
10	
11	43
12	
13	
14	
15	

3. [10] Consider a series of word address references 11, 22, 33, 44, 55, 66, 77, 32, 45, 54, 67, 76. Assuming a direct-mapped cache with **8 two-word blocks** that are initially empty, label each reference in the list as a *hit* or a *miss* and fill the respective row for Binary address, Tag and Index in Table 3

Solution Q3: To get the word index shift right the binary address by 1 bit

Table 3

Reference	Hit or Miss	Address in Binary	Tag	index
11	Miss	0001011	000	101
22	Miss	0010110	001	011
33	Miss	0100001	010	000
44	Miss	0101100	010	110
55	Miss	0110111	011	011
66	Miss	1000010	100	001
77	Miss	1001101	100	110
32	Hit	0100000	010	000
45	Miss	0101101	010	110
54	Hit	0110110	011	011
67	Hit	1000011	100	001
76	Miss	1001100	100	110

4. [10] Consider a series of word address references 1, 134, 212, 1, 135,213,162, 161, 2, 44, 41, 221. Assuming a direct-mapped cache with **Four 4-word** blocks that are initially empty, label each reference in the list as a *hit* or a *miss* and fill the respective rows for Binary address, Tag and Index in Table 4

Solution Q4: To get the word index, right shift the binary address by 2 bits

Table 4:

Reference	Hit or Miss	Address in Binary	Tag	index
1	Miss	00000001	0000	00
134	Miss	10000110	1000	01
212	Miss	11010100	1101	01
1	Hit	00000001	0000	00
135	Miss	10000111	1000	01
213	Miss	11010101	1101	01
162	Miss	10100010	1010	00
161	Hit	10100001	1010	00

2	Miss	00000010	0000	00
44	Miss	00101100	0010	11
41	Miss	00101001	0010	10
221	Miss	11011101	1101	11

5. **[10]** C1 is a direct-mapped Cache with **16 one-word blocks**. C2 is another direct-mapped Cache with **4 four-word blocks**. Assume that the miss penalty for C1 is 8 clock cycles and the miss penalty for C2 is 11 clock cycles. Also assume that the Caches are initially empty.

a. [3] Find the length of a shortest reference string of word addresses for which C2 has a lower miss rate but spends more cycles on Cache misses than C1. Explain your answer

b. [7] For the reference string 0, 4, 8, 11 find the miss cycles in both caches by creating Table 5 below to show your work.

Solution Q5:

a) The first occurrence of higher penalty and lower miss rate happens when there are 4 word addresses in the string. That is, if all references miss in C1, the miss cycles is 4*8=32 and with only 3 misses in C2, the miss cycles would be 3*11=33; 32<33 and hence the shortest string has four addresses.

b) Miss cycle in Cache 1 = 4*8=32 and Miss cycle in Cache 2 = 3*11 =33

Table 5

Reference	Hit or Miss		Address in Binary	Cache 1		Cache 2	
	Cache C1	Cache C2		Tag	Index	Tag	Index
0	Miss	Miss	00000	0	0000	0	00
4	Miss	Miss	00100	0	0100	0	01
8	Miss	Miss	01000	0	1000	0	10
11	Miss	Hit	01011	0	1011	0	10

6. [10] Using the series of references given in problem 2, show the hits and misses and final cache contents for a two-way set-associative cache with one-word blocks and a *total size* of 16 words. Assume LRU replacement.

Solution Q6:

Table 6

Block Number	Element #1 Address	Element #2 Address
0	56	8
1	17	9
2		
3	43	11
4	4	20
5	5	
6	6	
7		

Table 7

Reference	Hit or Miss	Reference	Hit or Miss
1	Miss	9	Miss
4	Miss	11	Miss
8	Miss	4	Hit
5	Miss	43	Miss
20	Miss	5	Hit
17	Miss	6	Miss
19	Miss	9	Hit
56	Miss	17	Hit

7. [5] Consider a virtual memory system with the following properties:

40-bit virtual byte address
16-KB pages
36-bit physical byte address

What is the total size of the page table for each process on this machine, assuming that the valid, protection, dirty, and use bits take a total of 4 bits and that all the virtual pages are in use? (Assume that disk addresses are not stored in the page table.)

Solution Q7:

The total size is equal to the no. of entries * the size of each entry.
→ Each page is 16 KB, and thus, 14 bits of the virtual and physical address will be used as a page offset.
→ The remaining 40-14=26 bits of the virtual address constitute the virtual page number and there are thus 2^{26} entries in the page table, one for each virtual page number.
→ Each entry requires 36-14=22 bits to store the physical page number and an additional 4 bits for the valid, protection dirty and use bits.
This gives us a total size of 2^{26} * 26 bits = 208MB.

8. [10] If all misses are classified into one of three categories -- compulsory, capacity, or conflict

a. [4] Which misses are likely to be reduced when a program is rewritten so as to require less memory?
b. [2] How about if the clock rate of the machine that the program running on is increased?

c. [4] How about if the associativity of the existing cache is increased?

Solution Q8:

a) Rewriting the program will likely reduce compulsory as well as capacity misses! (Conflict might still happen).
b) Increasing the clock rate will have no change.
c) Increasing the associativity should reduce conflict.