ECE 570
High-Performance Computer Architecture
Winter 2015
Assignment #2

[25 pts]

1- Consider the following C program to perform matrix multiplication. You will need to use Visual Studio for this problem.

(a) Parallelize this code using pthreads with two threads. The code should be written so that the generation of result matric `c[][]` is subdivided across the rows, i.e., rows 0 – (N/2)-1 are allocated to Thread 0 and rows N/2 – N-1 are allocated to Thread 1. In order to run pthreads in Visual Studio, you will need to add the pthreads library and include files. See the following link on how this can be done:
http://web.cs.du.edu/~sturtevant/pthread.html

(b) Parallelize this code using OpenMP with two threads. Again, the code should be written so that the rows 0 – (N/2)-1 are allocated to Thread 0 and the rows N/2 – N-1 are allocated to Thread 1. In order to run OpenMP with Visual Studio, you need to turn on the "OpenMP Support" option. The following link shows how this can be done:
http://supercomputingblog.com/openmp/getting-started-with-openmp-on-visual-studio/

For both programs, include statements to print out when Thread 0 and 1 are created and what their thread IDs are. In addition, include statements to print out which part of the result matrix the threads are working on.

```c
#include <stdio.h>
#include <stdlib.h>

#define N 10                    /* N x N matrix */


int main(void)
{
    int i, j, k;
    double  a[N][N], b[N][N], c[N][N];

    printf("Initializing matrices...\n");

    /*** Initialize matrices ***/
    for (i = 0; i<N; i++)
    for (j = 0; j<N; j++)
        a[i][j] = i + j;
    for (i = 0; i<N; i++)
    for (j = 0; j<N; j++)
        b[i][j] = i*j;
    for (i = 0; i<N; i++)
    for (j = 0; j<N; j++)
        c[i][j] = 0;

    for (i = 0; i<N; i++)
    {
        for (j = 0; j<N; j++)
            for (k = 0; k<N; k++)
                c[i][j] += a[i][k] * b[k][j];
    }


    /*** Print results ***/
    printf("******************************************************\n");
    printf("Result Matrix:\n");
    for (i = 0; i<N; i++)
    {
        for (j = 0; j<N; j++)
            printf("%6.2f   ", c[i][j]);
        printf("\n");
    }
    printf("******************************************************\n");
    printf("Done.\n");
    getchar(); /* Keeps the command window open in Visual Studio*/
}
```

[25 pts]

2- Consider a dual-core (P1 and P2) SMP system with two-way, set-associative caches, write-back, and LRU replacement policy. Each cache has four blocks (i.e., lines) labeled 0, 1, 2, and 3. The shared main memory consists of 8 blocks labeled 0, 1, 2, …, 7. Assume the same clock drives the processors and the memory bus and the following:

- Each transaction (request/response) completes in one cycle.
- In case of simultaneous bus requests from both processors, the priority is given in a round-robin fashion, i.e., P1, P2, P1 and so on…, i.e., if the bus was last acquired by P1 then P2 will be given the priority.

For the following two asynchronous sequence of memory-access events, where boldface numbers are for writes and the remaining are for reads, trace the execution of these block accesses on the two processors using the MESI coherence protocol.

P1: 0, **0**, 0, **1**, 1, 4, 3, 3, 5, **5**, 5
P2: 2, **2**, 0, 0, **7**, 5, 5, 5, **7**, 7, 0

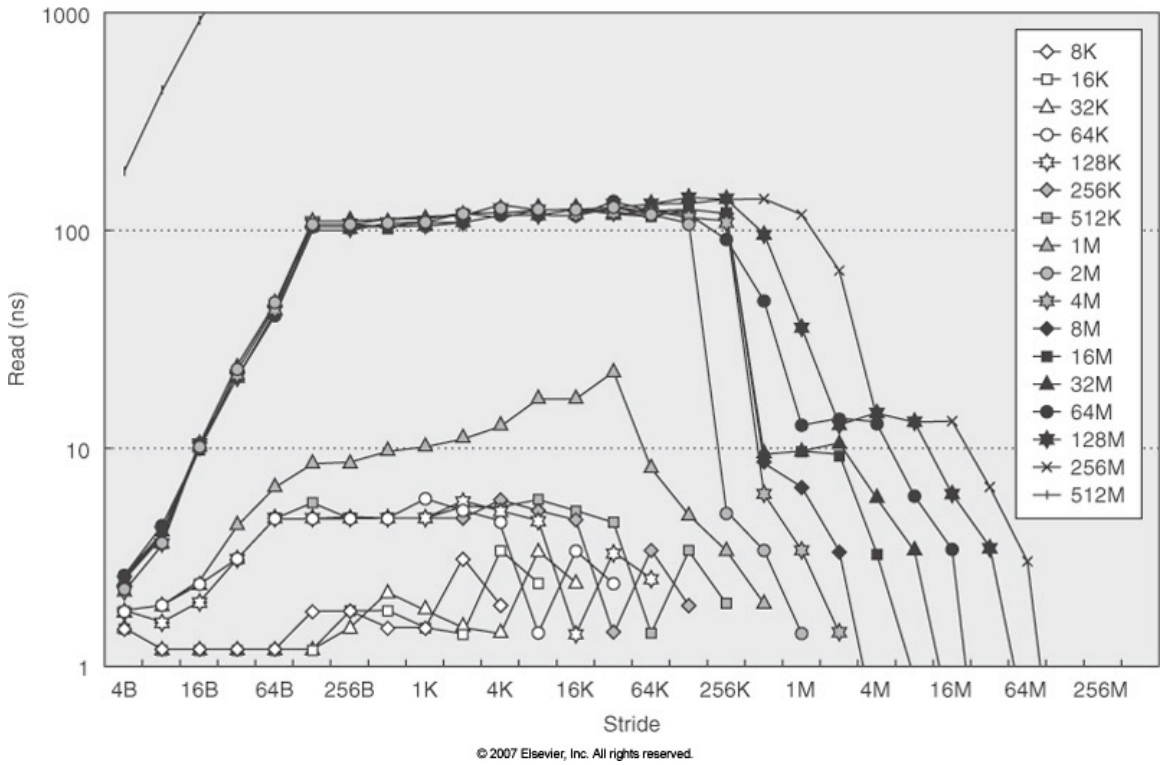
Clearly indicate all the operations performed in each cycle in the "Comments" column; i.e., processor request (`PrRd`-hit/miss or `PrWr`-hit/miss), bus request (`BusRd`(`S` or `S'`), `BusRdX`, `BusUpgr`), and whether a cache (`flush`) or main memory responds (MM responds) to complete the transaction. The first bus transaction is shown below.

T	P1	Set 0		Set 1		h	B	P2	Set 0		Set 1		h	Comments
		0	1	2	3				0	1	2	3		
1	0	E (0)					P1	2						P1 `PrRd`-miss, `BusRd(S')`, MM responds P2 `PrRd`-miss, P2 waits
2	**0**													
3	0													
4	**1**													
5	**1**													
6	1													
7	4													
8	3													
9	3													
10	3													
11	5													
12	**5**													
13	5													

T indicates the cycle. For P1 and P2 columns, b indicates when a request (read/write) is first issued for block b. 0-3 columns indicate the state of the block b (and cache block number indicated in parenthesis). B column indicates which processor has access to the bus and h column indicates whether there was a cache hit.

[25 pts]

3- Consider the graph shown below, which is the result of running a program that reads memory locations using different block (8 Kbytes - 512 Mbytes) and stride (4 bytes – 256 Mbytes) sizes on a hypothetical system. This allows for estimating latencies of different levels of the memory hierarchy.
   (a) How many levels of caches are there?
   (b) What are the overall size and block size of the L1-cache?
   (c) What are the overall size and block size of the L2-cache, if there is one?
   (d) What are the miss penalties for the L1-cache and (if present) L2-cache?
   (e) What is the time for a page fault to secondary memory (i.e., hard disk)?



[25 pts]

4- Suppose the base CPI with a perfect memory system is 1.5, i.e., CPU_{exec}=1.5. Determine which cache configuration is the best by computing the CPU_{time} for the three caches below:
   (a) 16-Kbyte direct-mapped unified cache using write-back and has a miss rate of 0.029.
   (b) 16-Kbyte 2-way, set-associative unified cache using write-back and has a miss rate of 0.022.
   (c) 32-Kbyte direct-mapped unified cache using write back and has a miss rate of 0.02.

   Assume the following:
   - The memory system has a unified cache (for instruction and data) and main memory (there is no L2 cache).
   - 64-bit wide main memory has an access latency of 100 clock cycles, the bandwidth between main memory and cache is 4 bytes per clock cycle and that 50% of the blocks in the cache are dirty.
   - There are 64 bytes per block and 25% of the instructions are load and store operations.
   - There is no write buffer, thus there is latency involved in writing back the evicted cache blocks to the main memory.
   - $CCT_{16K,\ 2\text{-way}} = 1.10 \times CCT_{16K,\ 1\text{-way}}$
   - $CCT_{32K,\ 1\text{-way}} = 1.30 \times CCT_{16K,\ 1\text{-way}}$