COL 380	January 26, 2018
Homework 1	
Instructor: Subodh Sharma	**Due: February 2, 23:55 hrs**

NOTE: Answer submissions must be made either in word document or pdfs. No hand-written assignments would be accepted. All assignment submissions will be checked for plagiarism.

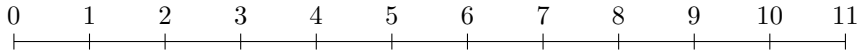
Problem 1: Pipelining

Consider the execution of the following code:

```
float x[1000], y[1000], z[1000];
...
for(i = 0; i < 1000; i++)
   z[i] = x[i] + y[i];
```

In each iteration of the loop, we are adding two floats which require 7 operations (Fetch operands, Compare exponents, Normalize, Add, Normalize result, Round result, Store result). Assuming "Fetch operands" and "Store result" take 2ns each and every other operation takes 1 ns to complete, answer the following:

- How long does it take to execute the loop in an unpipelined processor? (2 marks)
- How long does it take to execute the loop in a 2-way pipelined processor? Show the pipeline state after each cycle for 11 cycles. The format is shown below. (4 marks)



For each subproblem, show your working.

Problem 2: Caches

Consider a memory system with L1 cache of 32 KB and DRAM of 1 GB with the processor operating at 1GHz. The latency to L1 cache is 1 ns and that to DRAM is 100 ns. In each fetch cycle, 4 memory words (a cache line) are transfered from DRAM to CPU.

- Consider now the problem of multiplying two dense matrices (4K $\times$ 4K) as shown below; the matrices are laid out in a row-major fashion in memory. What is the peak achievable performance for the sought multiplication? (4 marks)

```
for(i = 0; i < SIZE; i++)
  for(j = 0; j < SIZE; j++)
    for(k = 0; k < SIZE; k++)
      z[i][j]  += x[i][k] * y[k][j];
```

Problem 3: Mutual Exclusion & Deadlocks

Barrier is a synchronization construct where a set of processes synchronizes globally i.e. each process in the set arrives at the barrier and waits for all others to arrive and then all processes leave the barrier. Let the number of processes in the set be three and S be a binary variable (also called a *binary semaphore* used to implement lock (resp. unlock) functions L (resp. UL). Consider the following **C** implementation of a barrier with line numbers shown on left.

```
1  void barrier (void) {
2   L(S);
3     process_arrived ++;
4   UL(S);
5   while (process_arrived !=3);
6   L(S);
7     process_left++;
8     if (process_left==3) {
9       process_arrived = 0;
10      process_left = 0;
11    }
12  UL(S);
13 }
```

The variables process arrived and process left are shared among all processes and are initialized to zero. In a concurrent program all the three processes call the barrier function when they need to synchronize globally. The above implementation of barrier is incorrect. Which one of the following is true? (2 marks)

1. The barrier implementation is wrong due to the use of binary variable S.
2. The barrier implementation may lead to a deadlock if two barrier in invocations are used in immediate succession.
3. Lines 6 to 10 need **not** be inside a critical section.
4. The barrier implementation is correct if there are only two processes instead of three.

How would one rectify the problem? (4 marks)

Problem 4: Mutual Exclusion & Deadlocks

Consider following algorithm for realizing mutual exclusion between two concurrent processes i and j.

```
1 while(true) {
2   flag [i] = true;
3   turn = j;
4   while ( P ) ;
5    /* enter critical section, perform actions, exit critical section */
6   flag [i] = false;
7 }
```

For the program to guarantee mutual exclusion, the predicate P in the while loop should be (3 marks):

1. flag[j] = true and turn = i
2. flag[j] = true and turn = j
3. flag[i] = true and turn = j
4. flag[i] = true and turn = i

Problem 5: Performance

- Assume $T_{parallel} = \frac{T_{serial}}{p} + T_{overhead}$ where p is the number of cores. Show that keeping p fixed, if we were to increase the input problem size, then *efficiency* will increase. All suitable assumptions must be clearly specified. (3 marks)
- Suppose $T_{serial} = n$ and $T_{parallel} = \frac{n}{p} + log_2 p$. Show whether or not the program is scalable. (4 marks)
- Design a cost-optimal version of the parallel-sum algorithm for n numbers on p processing cores such that $p < n$. Supposing that addition of two numbers takes 1 unit of time while communication among cores takes 20 units of time, derive expressions for $T_{parallel}$, S, E, cost, and the isoefficiency function. (6 marks)