COL 380 **January 28, 2016**

Homework 1

Instructor: Subodh Sharma **Due: February 3, 18:00 hrs**

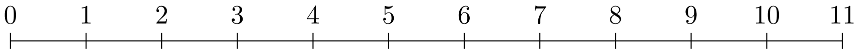
Problem 1: Pipelining

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 the multiplication of a matrix (4K × 4K) and a vector shown below. Each row of the matrix takes 16 KB of storage. What is the peak achievable performance of this multiplication using a two-loop dot product based matrix-vector product? (3 marks)

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

- Consider now the problem of multiplying two dense matrices (4K × 4K) where the matrices are laid out in a row-major fashion in memory. What is the peak achievable performance for the sought multiplication? (3 marks)

Problem 3: Caches++

We execute the following program with two threads on a shared memory machine:
$T_0 : x = 1; \mid T_1 : y = x;$
Assume initially the values of shared variable x, y is set to zero. Suppose the system uses a snoping cache coherence with a write-back mechanism.

- What will be the value assigned to y? (2 mark)
- If we shift to directory-based coherence, will the answer to the above question change? If so, then what will be the value of y? If not, then explain the rationale. (2 mark)
- How the problems found in the above two questions, if any, can be resolved? (2 marks)

Problem 4: 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 prefix sums 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)

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.