*Copyright ©2019, the University of Minnesota.* 

| **CSci 5451, S'20** | **Homework # 1** | **Due Date: 02-12-20** |
|---|---|---|

1. Consider the problem of computing the sum of $n$ numbers, $x = a_1 + a_2 + \cdots + a_n$. For the case $n = 8$, show the Dependence Analysis Graph (DAG) of the sum operation when it is performed as
$$x = a_1 + (a_2 + (a_3 + (a_4 + (a_5 + (a_6 + (a_7 + a_8)))))).$$
Recall that the DAG is a binary tree with data at the leaves and operations (in this case '+') at all other nodes. Still for the case $n = 8$, show the same DAG for the case when the sum is done by the cascade algorithm.

   What is the shortest height that a binary tree can have, among all possible binary trees with $n$ leaves? [No proofs required]. Assuming you have enough adders, what is the smallest number of time units needed to add $n$ numbers, if a time unit corresponds to the time an adder completes one operation (and if communication times are neglected)? What algorithm achieves this mininum?

2. Given a vector $x$ with the $n$ components $x_0, x_1, \cdots, x_{n-1}$ the prefix sum (or scan-sum, or cumulative sum) of $x$ is the vector consisting of the $n$ scalars $y_i$, for $i = 0, 1, \cdots, n-1$ where $y_i = x_0 + x_1 + \cdots + x_i$.
   (a) Design an algorithm for computing prefix sums in which the following property is exploited:
$$[y_m, y_{m+1}, \cdots, y_{n-1}] = y_{m-1} + \text{PrefSum}(x_m, x_{m+1}, \cdots, x_{n-1}),$$
   where $m < n-1$ and $\text{PrefSum}(x_m, x_{m+1}, \cdots, x_{n-1})$ is the prefix sum of $x_m, x_{m+1}, \cdots, x_{n-1}$. For example if you have 4 numbers you can calculate the half prefix sums $y_0 = x_0, y_1 = x_0 + x_1$ and $z_0 = x_2, z_1 = x_2 + x_3$ and then get the whole prefix sum of $x$ by calculating $y_2 = y_1 + z_0$, $y_3 = y_1 + z_3$. Write down the algorithm in pseudo-code [or matlab code] – Indicate which loop (or loops) is (are) parallel. Show the results of the outer loop steps (sequential loop) for the case when $x = [1, 2, -1, 4, -1, 6, -1, 8]$. What will be the execution time of your algorithm in a PRAM model assuming an addition takes $\tau$ seconds? What memory access protocol will be needed (EREW? CREW? ....). How many processors will be needed? (you may assume that $n$ is a power of 2 for simplicity)

   (b) A related problem is that of determining the 'nearest one from the left': in an array of zeros and ones, determine for any $j$ the nearest index $i$ of an $x(i)$ that is equal to one, where $0 \leq i \leq j$. [If all elements $x(i)$, for $i \leq j$ are zero then set $y(j) = -1$.] For example if $x = [0, 0, 1, 1, 0]$ then the answer is $y = [-1, -1, 2, 3, 3]$. How can you adapt the prefix sum algorithm to solve this problem? Show the result of the sequential loop of your algorithm for the case when $x = [0, 1, 0, 1, 0, 1, 1, 0]$. Answer the same questions as in (a): Number of processors needed and parallel time complexity with the PRAM model.

3. The time it takes to perform a given operation (e.g., an addition) of two vectors of length $n$ on a pipelined vector computer is usually cast in the form
$$t(n) = \sigma + n/\rho$$
   where $\sigma$ is the start-up time and $\rho$ is the peak speed (number of FLoating point Operations Per Second or FLOPS).

(a) What is the peak MFLOPS (Mega FLOPS) rate of a vector computer when $\sigma = 10^{-6} sec$, $\rho = 10^8$? What is the MFLOPS rate that is actually achieved for vectors of length 1000? 100? 10?

(b) What is the number $n_{1/2}$ for which the MFLOPS rate achieved is half the peak rate (obtain a general formula for $n_{1/2}$ which depends on the parameters $\sigma$ and $\rho$)? What is $n_{1/2}$ for the example in (a)? Show that $t(n)$ can be written as

$$t(n) = \frac{n_{1/2} + n}{\rho}$$

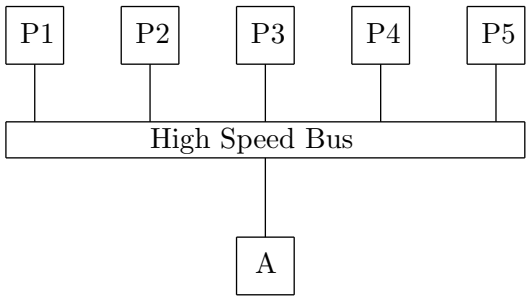
The number $n_{1/2}$ is termed the *half-performance length* or *half vector length.*

(c) Determine the peak performance rate and the half vector length for the CRAY X-MP (a machine from the early 1990s) which had a clock cycle is 8.5 ns, knowing that it took 10 cycles to get one result out ("start-up of 9 cycles").

4. A 'master' computer $A$ is connected to a number $p$ of identical processors $P_i, i = 1, ..., p$, via a high-speed bus. Consider the problem of adding $N$ numbers by having processor $A$ split the computatoin on $p$ equal pieces and have the 'workers' do the the $p$ subsums. Then $A$ will obtain the subsums and add them (see notes). It is assumed that the high-speed bus is infinitely fast but that moving data with the local bus of processor $A$ to any processor takes a time of the form

$$\beta + N/\tau$$

where $N$ is the size of the data, $\beta$ is the communication start-up time, and $\tau$ is the memory bandwidth of $A$'s bus in words/sec. All processors ($A$ and the $P_i$'s) can perform $\rho$ floating point operations per second, and the computations are synchronus (the adds start at the same time and end at the same time).




(a) Estimate the execution time for adding $N$ numbers with when $p$ processors are used. Is there a speed-ups when $\tau < \rho$?

(b) Show a typical curve of execution time versus $p$. Is there an optimal number of processors $p_*$ to use? If so what is it?

(c) Assume that $\beta = 50/\rho$ and $\tau = \rho/10$. What is the optimal number of processors when $N = 10,000$? What is the corresponding optimal time (expressed in terms of $1/\rho$ seconds)?

5. Consider a memory system with a level 1 cache of 32KB, and DRAM of 512MB with the processor operating at 1 GHz. The latency to L1 cache is one cycle and the latency to DRAM is 100 cycles. In each memory cycle, the processor fetches 4 words (cache line size is 4 words). What is peak achievable performance of a dot-product of 2 vectors? [Where necessary assume an optimal cache replacement policy]