CSCE 5160: Solutions to Homework #1

2.2. The computation performs 8 FLOPS on 2 cache lines, i.e., 8 FLOPS in 200 ns. This corresponds to a computation rate of 40 MFLOPS.

2.3 In the best case, the vector gets cached. In this case, 8 FLOPS can be performed on 1 cache line (for the matrix). This corresponds to a peak computation rate of 80 MFLOPS (note that the matrix does not fit in the cache).

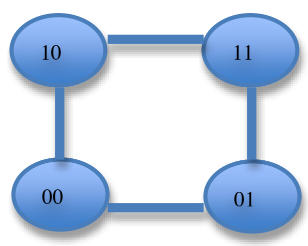
2. 12 (two different ways to solve this)

Consider a cycle $A1, A2, \ldots, Ak$ in a hypercube. As we travel from node Ai to $Ai+1$, the number of ones in the processor label (that is, the parity) must change. Since $A1 = Ak$, the number of parity changes must be
even. Therefore, there can be no cycles of odd length in a hypercube.

(Proof adapted from Y. Saad and M. H. Schultz. Topological properties of hypercubes. *IEEE Transactions on Computers*, 37:867–872, 1988

Using Induction.

For 1-dimensional cube, there are two nodes, you can only have one cycle of even length.
For 2-dimensional cube, we have 4 nodes.



Possible cycles include
Length 2: 00-01-00; 00-10-00; 10-11-10; 01-11-01, etc.
Length 4: 00-01-11-10-00, etc.

No odd length cycles are possible.

Assume that only even length cycles are possible in a k-dimensional hypercube

Now we will show that only even length cycles are possible in (k+1) dimensional hypercube.

A k+1 dimensional cube can be viewed as 2 k-dimensional cubes with corresponding nodes connected.
We can only have even cycles in each of the k-dimensional cubes.

We can extend an even cycle in one cube by visiting one or more nodes in the second. This is only possible by adding even number of links to return back to the start node in the first cube.

Consider adding a single node from the second cube to a cycle that is entirely contained in the first cube. You will need to travel one link to reach the new node and (at least) another link to return to source node of the cycle.

If we are adding more than one node from the second cube, we are actually merging two cycles of even length and adding at least 2 additional links to connect the two cycles (to cross from one subcube to the second and return). Thus the new cycle will have even number of links.

2.13 Consider a 2^d processor hypercube. By fixing k of the d bits in the processor label, we can change the remaining $d - k$ bits. There are 2^{d-k} distinct processors that have identical values at the remaining k bit positions. A p-processor hypercube has the property that every processor has log p communication links, one each to a processor whose label differs in one bit position. To prove that the 2^{d-k} processors are connected in a hypercube topology, we need to prove that each processor in a group has $d - k$ communication links going to other processors in the same group.

Since the selected d bits are fixed for each processor in the group, no communication link corresponding to these bit positions exists between processors within a group. Furthermore, since all possible combinations of the $d - k$ bits are allowed for any processor, all $d - k$ processors that differ along any of these bit positions are also in the same group. Since the processor will be connected to each of these processors, each processor within a group is connected to $d - k$ other processors. Therefore, the processors in the group are connected in a hypercube topology.