PROBLEM SET 1

Zhiqiang Xie 77892769

Problem 1:

- One person will cost $(n-1)*t_c$ to add n numbers.

1. All eight people sat in a circle.
    - $t=(\frac{1}{8}n-1)*t_c+3*t_c+4*t_w=(\frac{n}{8}+2)t_c+4t_w$
2. The eight people are sitting in two rows of four people each.
    - $t=(\frac{1}{8}n-1)t_c+3*t_c+3*t_w=(\frac{n}{8}+2)t_c+3t_w$

Actually, the calculation above assumes that a person can't do the summation while passing number or receiving. In such condition, the time cost $(\frac{n}{8}-1)t_c+\log(8)t_c=(\frac{n}{8}+2)t_c$ of summation is required. We could seat these 8 persons as a 3-d cube, which will achieve $3t_w$ cost for passing data.

Problem 2

Processor1 (SIMD)	**Processor2 (SIMD)**	**Processor1 (MIMD)**	**Processor2 (MIMD)**	**Step**
Compare(52,12)	Compare(24,64)	Compare(52,12)	Compare(24,64)	1
Compare(52,12)	Compare(24,64)	Compare(52,12)	Compare(24,64)	2
Subtract(52,12)		Subtract(52,12)	Subtract(24,64)	
	Subtract(24,64)	Compare(40,12)	Compare(24,40)	4
Compare(40,12)	Compare(24,40)	Compare(40,12)	Compare(24,40)	5
Compare(40,12)	Compare(24,40)	Subtract(40,12)	Subtract(24,40)	6
Subtract(40,12)		Compare(28,12)	Compare(24,16)	7
	Subtract(24,40)	Compare(28,12)	Compare(24,16)	8
Compare(28,12)	Compare(24,16)	Subtract(28,12)	Subtract(24,16)	9
Compare(28,12)	Compare(24,16)	Compare(16,12)	Compare(8,16)	10
Subtract(28,12)	Subtract(24,16)	Compare(16,12)	Compare(8,16)	11
Compare(16,12)	Compare(8,16)	Subtract(16,12)	Subtract(8,16)	12
Compare(16,12)	Compare(8,16)	Compare(4,12)	Compare(8,8)	13
Subtract(16,12)		Compare(4,12)		14
	Subtract(8,16)	Subtract(4,12)		15
Compare(4,12)	Compare(8,8)	Compare(4,8)		16
Compare(4,12)		Compare(4,8)		17
Subtract(4,12)		Subtract(4,8)		18
Compare(4,8)		Compare(4,4)		**19**
Compare(4,8)		**END**		20
Subtract(4,8)				21
Compare(4,4)				**22**
END				

Here we should see, MIMD takes less steps in this kind of general use.

Problem 3

- In a hypercube, we could represent every node in a binary number, where every two adjacent nodes differ in only one bit. For nodes u and v with distance of h, they differ in h bits. Here are some insights:

    - Every move in the shortest path will set one different bit to the same as destination.
    - From a combination view, the number of different path $n = h * (h-1) * ... * 1 = h!$
- For a set of path in which no edge appears in more than one path, we know that we have only h kind of moves to take as the first step, which will be the bottleneck of the amount of paths in this set.

Problem 4

For routing a path between two processors:

- Determine the digits of ID which differ to the ID of destination processor.
- For the i_{th} layer of Omega network, set the switch to be crossed if the i digit of processor ID differs to the destination, or set it to be pass through.

It's easy to see that every layer of Omega network helps to change the corresponding bit of ID, using methods like XOR could be helpful to route a message between two processors.

Problem 5

1. It takes 11 units of time to find the solution.
2. It could find the solution in 4 units of time.
    - The speedup $S_p = \frac{11}{4} = 2.75$
    - It is anomaly since it exceeds 2 (the number of processors we deploy). However, the original search task is different in the parallel case. Since every time we do a search, the probability in parallel case is two times larger as sequential. It's asymptotically equal in expectation but may be anomaly in small scale.
3. Since the p processor are identical, we can simulate them in $O(p)$ time by one time slot to one time slot mapping. Actually, the time cost constant for one mapping can't be less than 1 , so we can't achieve a super-linear absolute speedup. However, the concepts described here are independent to the phenomenon observed above. Besides, some practical issue like cache is not discussed here either.

Problem 6

1.

Processor1	**Processor2**
Initialization task	
Task1	Task6
Task2	Task7
Task3	Task8
Task4	Task9
Task5	Task10
Finalization task	

The time speedup $S_p = \frac{12}{7}$.

2. Based on Amdahl's Law, the max speedup $S_p = \frac{12}{2+10/p}$, where $10/p$ here can't be less than 1 since it's a minimum time unit of processor. Therefore $S_p = 4$.
3. We need at least 10 processors here to achieve the speedup.

Processor \ Time	**1**	**2**	**3**
Processor1	Initialization task	Task1	
Processor2		Task2	
Processor3		Task3	
Processor4		Task4	
Processor5		Task5	
Processor6		Task6	
Processor7		Task7	
Processor8		Task8	
Processor9		Task9	
Processor10		Task10	Finalization task

Problem 7

1. $S_p = \frac{1}{f+(1-f)/p}$, where $p = 10$ and $S_p = 9$:
   - Thus $f \leq \frac{1}{81}$
2. $S_p = \frac{f*T_1+p(1-f)T_1}{T_1}$, where $T_1 = 225$ and $f * T_1 = 9, p = 16$.
   - Thus $S_p = 15.4$.
3. $S_p = \frac{30+270*p}{30+270} = \frac{1+9p}{10}$
   - Thus $S_p(20) = 18.1$, $S_p(30) = 27.1$, $S_p(40) = 36.1$

Problem 8

1. For the case of $n = p$:
   - Execution time $= \log p + \log p = 2 \log p$
   - Speedup $= \frac{p-1}{2 \log p}$
   - Cost $= 2 * p \log p$

   It's not cost-optimal.
2. For the case of $n > p$:
   - Execution time $= \frac{n}{p} - 1 + 2 \log p$
   - Cost $= n - p + 2 * p \log p$

   If $p \log p \in O(n)$, the algorithm could be cost-optimal. And it's usually satisfied in practical situation.