Multicore Processors: Architecture & Programming

Homework Assignment # 2

[Total: 40 points]

1. [4] Suppose we have a parallel program, for a shared-memory machines, consists of two threads. What are the characteristic(s) of these two threads that make them getting better performance if executing on a single two-way hyperthreading core instead of two cores each of which is two-way hyperthreading? Assume the two threads belong to the same process.

2. [4] Having too many threads (i.e. way more than the available cores) in an application may not be a good idea. State two reasons for that.

3.[4] Having too few threads in an application is not a good idea either. State three reasons explaining why. Assume the problem size is big enough.

4. [6] Modify the following piece of code to allow iterations to be executed in parallel to gain some performance.

```
a[0] = 0;
for( i = 1; i < n; i++)
    a[i] = a[i-1] + i;
```

5. Suppose a program is divided into 7 tasks. One task at the beginning must be executed by itself (i.e. with no other tasks in parallel with it) and takes 3 ms. Also there is a task at the end of the program that must be executed by itself and takes 4 ms. Between these two tasks, there are 5 tasks that can be executed simultaneously and each of these tasks takes 16ms.

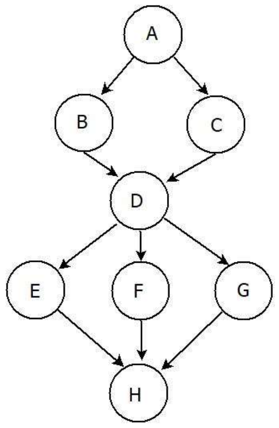
a. [2] What is maximum speed-up according to Amdahl's law? [Hint: Calculate F as a fraction of time of the sequential part to the total execution time]

b. [2] What is the largest number of cores we can use after which no speedup will be seen? Justify. [Hint: Do not use the parallelism equation here. Just try to figure out the number of cores.]

c. [2] If we use that largest number of cores you calculated in the above question, what is the speedup we get? Do not use Amdahl's law in your calculations in this question.

d. [2] Is there a difference between the speedup you calculated in c and a above? Justify.

6. Assume we have the following task flow graph where every node is a task and an arrow from a task to another means dependencies. For example, task B cannot start before task A is done.



The following table shows the time taken by each task in nano seconds.

Task	Time Taken
A	50
B	10
C	10
D	5
E	10
F	100
G	10
H	100

a. [4 points] What is the minimum number of cores needed to get the highest performance? What is the speedup given the number of cores you calculated? Do not use the parallelism ratio in this part of the problem.

b. [4 points] What is the span? What is the work?

c. [2 points] Given what you calculated in (b) above, what is the parallelism?

d. [4 points] Is the number of cores you calculated in (c) different than the number of cores you calculated in (a)? Justify.