CSE 240A - Principles of Computer Architecture - Spring 2014
Assignment 3

1. Consider the following code, and assume that long integers are 8 bytes and that ThreadA() and ThreadB() are run on different processors. You notice that when ThreadA() runs alone, it can update A 1 billion times per second. When ThreadB() runs alone, it increments B 1 billion times per second. However, when they run simultaneously on different processors, they each only manage to increment their variable 100 million times per second. Why? How would you fix it?

```
long int A;
long int B;

void ThreadA() {while (1) {A++;}}
void ThreadB() {while (1) {B++;}}
```

2. Let's say we have a CMP that has two cores with one execution unit each that do not share resources, and an SMT that has one core with 2 execution units and shared resources (such as instruction queue, reorder buffer, L1 cache). Both the CMP and the SMT have the same amount of resources.
    a. Describe code for which the SMT would outperform the CMP and explain why.
    b. Describe code for which the CMP would do better than the SMT and explain why.

3. Consider that you have the following information about your system:
    • Accessing the TLB: 3ns
    • Indexing the cache to access its data portion: 4ns
    • Indexing the tag array of the data cache: 3.5ns
    • Tag comparisons: 2ns
    • Multiplexing the output data: 1.5ns

    For the following configurations, calculate the amount of time it takes to get data from the cache on a load hit, along with a brief explanation of your calculation. Assume that all other parts are insignificant to the access time of the cache.
    a. A physically-indexed, virtually-tagged cache
    b. A virtually-indexed, virtually-tagged cache
    c. A virtually-indexed, physically-tagged cache
    d. A physically-indexed, physically-tagged cache

4. Describe one way that we can exploit each of the three types of parallelism along with one challenge in doing so.
    a. Instruction level parallelism
    b. Data level parallelism

c. Thread level parallelism

5. Assume a machine with a typical MIPS 5-stage pipeline that uses branch prediction without branch delay slots and has a branch mispredict penalty of 3 cycles. 1 in every 5 instructions is a branch for a certain program, of which 80% are predicted correctly by our branch predictor. Given this:
    a. How many cycles would it take to execute 'n' instructions?
    b. Imagine we had a Pentium 4 instead which has a 20-stage pipeline because of which the branch mispredict penalty is now a staggering 19 cycles. What should your branch prediction rate be to have the same performance as the MIPS machine we saw in (a).

6. The size of the branch predictor can affect the amount of aliasing (two branches mapping to the same entry in a branch prediction table) between branches. While this sort of aliasing usually results in negative interference, it can sometimes result in positive interference.
    a. Describe a branch T-NT (taken not-taken) pattern for two branches for which aliasing will result in negative interference.
    b. Describe a branch T-NT (taken not-taken) pattern for two branches for which aliasing will result in positive interference.
    c. Why is it that aliasing usually results in negative interference?

    Some key terms here:
    Aliasing - The mapping of two or more branches to the same entry in a branch predictor data structure.
    Negative interference - The branch prediction rate of either or both aliased branches reduces from what they would've been without aliasing.
    Positive interference - The branch prediction rate of either or both aliased branches increases from what they would've been without aliasing.