Quiz 07

Problem 1

Consider a 2MB L2 cache with a 32-byte cache line. If the virtual address space is 64-bit, and the physical address space is 44-bit, compute the size of a single cache line tag if the cache is direct mapped. What if the cache is 2-way set associative?

Problem 2

Is the following a good idea?

```
const n = 0x100000;

double shared[n];

main() {
        t0 = CreateThread(Compute, 0);
        t1 = CreateThread(Compute, 1);
        JoinThread(t0);
        JoinThread(t1);
        cout << shared << endl;
}

Void Compute(int my_id) {
        for(i = my_id; i < n; i+= 2)
                shared[i] = Gamma(32.0, i);
}
```

If it is a good idea, explain why. If not, clean it up.

Problem 3

You have been hired to evaluate a design for a cache coherence protocol. The designer theorized that if caches are made to be write-through, then the memory copy of any data item will be always up to date. Other caches that want to access the data can easily get it from main memory in this case. This can simplify the cache coherence protocol. To read an item that is not in the cache, it is simply fetched from main memory. Any updates are made directly to the main memory. The performance degradation due to using write through caches can be justified by the simplicity obtained by eliminating the complex cache coherence protocol. What do you think?

Problem 4

```
#include <iostream>
#include <vector>
#include <thread>
int f()
{
    static int i = 0;
    synchronized { // begin synchronized block
        std::cout << i << " -> ";
        ++i;       // each call to f() obtains a unique value of i
        std::cout << i << '\n';
        return i; // end synchronized block
    }
}
int main()
{
    std::vector<std::thread> v(10);
    for(auto& t: v)
        t = std::thread([]{ for(int n = 0; n < 10; ++n) f(); });
    for(auto& t: v)
        t.join();
}
```

In the code above, each of 10 threads produces 10 unique numbers concurrently. Can you identify any case of false sharing?