Introduction to Concurrent Programming
Software & Hardware Basics

Slides by: Ofer Givoli

Based on:
The art of multiprocessor programming
Maurice Herlihy and Nir Shavit, 2008
• Appendix A – Software Basics
• Appendix B – Hardware Basics
Lets get to know ourselves 😊

• MAMAS?
• Java?
  • C#?
• Remember threads/synchronization?
Hardware Basics
From: https://software.intel.com/en-us/articles/optimizing-applications-for-numa
From: Computer Structure 2014 slides, by Lihu Rappoport and Adi Yoaz   (modified)
From: Computer Structure 2014 slides, by Lihu Rappoport and Adi Yoaz
SMP (symmetric multiprocessing)

NUMA (Non-uniform memory access)

From: The art of multiprocessor programming, by Maurice Herlihy and Nir Shavit, 2008 (modified)
Cache Coherence

From: The art of multiprocessor programming, by Maurice Herlihy and Nir Shavit, 2008  (modified)
Spinning

NUMA

SMP

From: The art of multiprocessor programming, by Maurice Herlihy and Nir Shavit, 2008  (modified)
Multi-Core and Multi-Threaded Architectures

- Reordering reads-writes to memory
- Memory barrier
- Volatile variables in Java

From: The art of multiprocessor programming, by Maurice Herlihy and Nir Shavit, 2008 (modified)
Software Basics
Threads

A *thread* executes a single, sequential program.

subclass of `java.lang.Thread`
public class HelloWorld implements Runnable {

...

public void run() {
    System.out.println(message);
}
}

Thread thread = new Thread(new HelloWorld());
thread = new Thread{
    public void run()
    {
        System.out.println(m);
    }
};;

thread.start();

thread.join();
Monitors

Monitor model (built-in)

Lock + Wait set

Waiting -> Runnable -> Running
```java
public synchronized T deq() {
    return call[(head++) % QSIZE]
}

public synchronized enq(T x) {
    call[(tail++) % QSIZE] = x;
}
```
Yielding and Sleeping

- yield()
- sleep(t)
Thread-Local Objects

ThreadLocal\<T\>

```java
public class ThreadID {
    private static volatile int nextID = 0;
    private static class ThreadLocalID extends ThreadLocal<Integer> {
        protected synchronized Integer initialValue() {
            return nextID++;
        }
    }
    private static ThreadLocalID threadID = new ThreadLocalID();
    public static int get() {
        return threadID.get();
    }
    public static void set(int index) {
        threadID.set(index);
    }
}
```

From: The art of multiprocessor programming, by Maurice Herlihy and Nir Shavit, 2008  (modified)
Thanks!