Midterm 1 Faux Quiz Compilation
CS378H Fall 2018

- Who was Flynn? Why is her/his taxonomy important?
- How does domain decomposition differ from functional decomposition? Give examples of each.
- Can a SIMD parallel program use functional decomposition? Why/why not?
- What is the maximum possible speedup of a 75% parallelizable program on 8 CPUs
- What is super-linear speedup? List two ways in which super-linear speedup can occur.
- What is the difference between strong and weak scaling?
- Define Safety, Liveness, Bounded Waiting, Failure Atomicity
- What is the difference between processes and threads?
- What's a fiber? When and why might fibers be a better abstraction than threads?
- What is the difference between spinning/busy-wait and blocking synchronization?
- Can you write shared memory parallel applications using single-threaded processes only?
- What criteria should you use to choose between spinlock/mutex on a multi-processor?
- Define the states of the MESI protocol. Is the E state necessary? Why or why not?
- What is bus locking?
- What is the difference between Mesa and Hoare monitors?
- Why recheck the condition on wakeup from a monitor wait?
- How can you build barriers with spinlocks?
- What is the difference between a mutex and a semaphore?
- How can you build a barrier without spinlocks or monitors?
- How are monitors and semaphores related?
- Why does pthread_cond_init accept a pthread_mutex_t parameter? Could it use a pthread_spinlock_t? Why [not]?
- Why do modern CPUs have both coherence and HW-supported RMW instructions? Why not just one or the other?
- What is priority inheritance?
- How are promises and futures related? Since there is disagreement on the nomenclature, don't worry about which is which—just describe what the different objects are and how they function.
- How does HTM resemble or differ from Load-linked Stored-Conditional?

- What are some pros and cons of HTM vs STM?
- What is Open Nesting? Closed Nesting? Flat Nesting?
- How does 2PL differ from 2PC?
- Define ACID properties: which, if any, of these properties does TM relax?
- How does HTM resemble or differ from Load-linked Stored-Conditional?
- How are promises and futures different or the same as goroutines?
- What is the difference between a goroutine and a thread?
- What is the difference between a channel and a lock?
- How is a channel different from a concurrent FIFO?
- What is the CSP model?
- What are the tradeoffs between explicit vs implicit naming in message passing?
- What are the tradeoffs between blocking vs. non-blocking send/receive in a shared memory environment? In a distributed one?
- What is a reduction? A prefix sum? Why are they hard to parallelize and what basic techniques can be used to parallelize them?
- Define flow dependence, output dependence, and anti-dependence: give an example of each. Why/how do compilers use them to detect loop-independent vs loop-carried dependences?
- What is the difference between a thread-block and a warp?
- How/Why must programmers copy data back and forth to a GPU?
- What is “shared memory” in CUDA? Describe a setting in which it might be useful.
- CUDA kernels have implicit barrier synchronization. Why is __syncthreads() necessary in light of this fact?
- How might one implement locks on a GPU?
- What ordering guarantees does a GPU provide across different hardware threads’ access to a single memory location? To two disjoint locations?
- When is it safe for one GPU thread to wait (e.g. by spinning) for another?
- What is hardware multi-threading; what problem does it solve?
- What is the difference between a vector processor and a scalar?
- Implement a parallel scan or reduction
- How are GPU workloads different from GPGPU workloads?

- How does SIMD differ from SIMT?
- List and describe some pros and cons of vector/SIMD architectures.
- GPUs historically have elided cache coherence. Why? What impact does it have on the the programmer?
- List some ways that GPUs use concurrency but not necessarily parallelism.
- Why do languages like CUDA and OpenCL have both blocks and threads?
- What does the __shared__ keyword do? How is it implemented by HW?
- CUDA kernels have implicit barrier synchronization: __syncthreads(). Is it necessary? Why or why not?
- Is traditional locking implementable on a GPU? What alternatives are there for synchronizing / avoiding races?
- How is occupancy defined (in CUDA nomenclature)?
- What is control flow divergence? How does it impact performance?
- What is a bank conflict?
- What is the difference between a thread block scheduler and a warp scheduler?
- Compare/contrast coarse/fine/simultaneous multi-threading.
- In CUDA, what is the difference between grids, blocks, and threads? What is their counterpart in OpenCL?
- What's the difference between a block scheduler (e.g. Giga-Thread Engine) and a warp scheduler?
- Modern CUDA supports UVM to eliminate the need for cudaMalloc and cudaMemcpy*. Under what conditions might you want to use or not use it and why?
- How are atomics implemented in modern GPU hardware?
- How is __shared__ memory implemented by modern GPU hardware?
- Why is __shared__ memory necessary if GPUs have an L1 cache? When will an L1 cache provide all the benefit of __shared__ memory and when will it not?
- Is cudaDeviceSynchronize still necessary after copyback if I have just one CUDA stream?