Parallel Programming Skills

* (15p) ***Race condition:***
  + ***(2p) What is race condition?***

This is Electronics, software or any other system behavior where the result depends on sequence or timing of uncontrollable events. We can see timing issues or chronological inaccuracies when errors occur here.

* + (5p) ***Why race condition is difficult to reproduce and debug?***

Usually programmers don’t even consider Race conditions because of how threads interact with one another. It is considered non-deterministic and is resultant on the relative timing between interfering threads. Because of this, it is not unusual to see problems disappear in debugging.

* + (8p) ***How can it be fixed? Provide an example from your Project\_A2 (see spmd2.c).***

Back then one main issue we had was the sharing of variables and states. We want each to be independent of the others and if we go back to older projects when we were working with numThreads without pragma. We shared the same resources or id in that case and then we had our issue. We fixed this condition by giving seperate id’s.

* (15p) ***Summaries the Parallel Programming Patterns section in the “Introduction to Parallel Computing\_3.pdf” (two pages) in your own words (one paragraph, no more than 150 words).***

There are multiple patterns described throughout however, each requires a different approach. For example, for strategies pattern developers determine what algorithmic strategy to use. We determine whether we want to run in parallel and distribute or how most efficiently to implement. There is another called concurrent execution pattern where the P/T control (process/thread) pattern where patterns are hidden since they are built in and units are controlled on call or runtime. They offer two types called message parsing and mutual exclusion where one uses MPI to pass messages between processes for different multicore machines and the other uses OpenMP to share memory app. Between threads.

* (12p) ***In the section “Categorizing Patterns” in the “Introduction to Parallel Computing\_3.pdf” compare the following:*** 
  + Collective synchronization (barrier) with Collective communication (reduction)
    - Collective Synchronization
      * how multiple processes will join up at a certain point
      * how multiple copies of a dataset are kept in order with one another to balance data integrity
    - Collective Communication
      * Each process performs the same communication operations
  + Master-worker with fork join
    - Master-worker
      * Master is responsible of assigning and receiving tasks to and from the worker, and workers are responsible for executing certain sub-task
      * After worker completes the task, the result is then submitted to the master, where it will be summarized
    - Fork join
      * Execution branches off (fork) in parallel at certain point in the program execution, and then merges (join) at a following point to resume sequential execution
      * Programs fork (make new) threads when needed, and when a thread finished execution, it will join with the program
* (26p) ***Dependency: Using your own words and explanation, answer the following:***
  + ***(3p) Where can we find parallelism in programming?***
    - Statement Level
      * Between program statements
    - Block/loop/routine/process level
      * Larger-grained program statements
  + (6p) What is dependency and what are its types (provide one example for each)?

Dependency is when one operation depends on an earlier operation to finish before a later operation can be executed

**Fundamental** **concurrent execution assumption :** Processes execute independent of each other

**sequential consistency :** Execution does not interfere with one another

* + (3p) ***When a statement is dependent and when it is independent (Provide two examples)?***

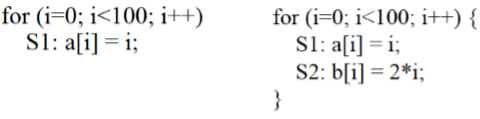
A statement is independent when the order of its execution doesn’t matter. This is like having independent states and we can set either without consequences happening. They are dependent when their execution order affects the result. This is like functions where y is dependent on x

* + (3p) ***When can two statements be executed in parallel?***

Two statements can be executed in parallel if and only if there are no dependency between the two statements.

* + (3p) ***How can dependency be removed?***

Dependence can be removed by modifying the program, via rearranging or eliminating statements.

* + (8p) ***How do we compute dependency for the following two loops and what type/s of dependency?***
    - For the first for loop (true dependence):

* + - For the second for loop (true dependence):