Parallel Computing

**Week 1**

**Sections [1.\*]**

* Other issues exist besides a “fast” processor
  + The most prominent, is the ability (or lack thereof) of the memory system to feed data to the processor at the required rate
  + Moore’s Law states that “circuit complexity doubles every eighteen months”

**Week 2**

**Sections [2.1,2.2,2.3,2.4]**

* Processors have long relied on pipelines for improving execution rates
  + Overlapping various stages in instruction execution
  + Enables faster execution
* To increase the speed of a single pipeline, one would break down the tasks into smaller and smaller units, thus lengthening the pipeline and increasing the overlap in execution
* In order to improve parallelism, we can increase the number of pipelines
  + During each clock cycle, multiple instructions are “piped” into the processor in parallel.
* **Superscalar Execution**
  + Ability of a processor to issue multiple instructions in the same cycle
* **True Data Dependency**
  + If the results of an instruction are not required for a subsequent instruction
  + Dependencies of this type must be resolved before simultaneous issue of instructions
    - Since the resolution id done at runtime, it must be supported in hardware
      * The complexity of this hardware can be high
    - The amount of instruction level parallelism in a program is often limited and is a function of coding technique.
      * *Minimize number of code segments that rely on other code segments*
* **Resource Dependency**
  + Occurs when two instructions require the same hardware
  + If there is no data dependency between instructions, we need to check that simultaneous instructions do not require the same resource. Thus, cannot be scheduled together
* **Branch Dependencies (Procedural Dependencies)**
  + Branch destination (i.e. if-else) is only known at time of execution, scheduling instructions *a priori* across branches may lead to errors.
  + Handled by speculatively scheduling across branches and rolling back in case of errors
  + Accurate branch prediction is critical for efficient superscalar execution.
  + \*\*Scheduling is very important in parallelism
* Processor needs the ability to issue instructions **out-of-order** to accomplish desired re-ordering.
  + Known as **dynamic instruction issue**
    - Exploits maximum instruction level parallelism
* The parallelism available in **in-order** issue of instructions can be highly limited
  + Disallows scheduling from re-arranging the instructions to make best use of pipelines
* Performance of superscalar architectures is limited by the available instruction level of parallelism
* **Vertical Waste**
  + If, during a particular cycle, no instructions are issued on the execution units
* **Horizontal Waste**
  + If, during a particular cycle, only part of the execution units are used