CSCE 614 Midterm Review Sheet

Chapter 1

- Performance measurement: latency and throughput. Latency = program execution time. Throughput = the amount of work done at a given time.

  $\uparrow$ latency $\Rightarrow \uparrow$ throughput, **BUT $\uparrow$ throughput $\not\Rightarrow \uparrow$ latency**

  Easier to improve on throughput but not latency
- Dynamic energy E = $\frac{1}{2} \times C \times V^2$
- $P \propto (C \times V^2 \times f)$, where P is dynamic power, C is capacitive load, V is voltage, and f is frequency switched.

  Static power is difficult to control; more power consumption statically nowadays
- Availability = $\frac{MTTF}{MTBF} = \frac{MTTF}{MTTF+MTTR}$
- Failure rate = $\frac{1}{MTTF}$, FIT = $\frac{10^9}{MTTF}$
- CPU time = IC × CPI × CCT

  Where IC = Instruction Counts, CPI = Cycles per Instruction (ideally 1 - WE have influence upon), CCT = Clock Cycle Time
- If clock frequency is given, CCT would be $\frac{1}{frequency}$.
- MIPS (Million Instructions Per Second) = $\frac{frequency}{CPI} = \frac{IC}{10^6 \times execTime}$
- Speedup = $\frac{ExecTime_{old}}{ExecTime_{new}} = \frac{Performance_{new}}{Performance_{old}}$
- Amdahl's Law: speedup is bounded by the area that one could not change. Max speedup = $\frac{original}{unchanged\ part}$. Overall speedup = $\frac{1}{(1-Fraction_{enhanced}+\frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$ (focus on common cases)

Pipelining

- 5 stages of MIPS architecture: IF, ID, EX, MEM, and WB.

  Add forwarding units between each stage (IF/ID, ID/EX, etc.)
- Three types of hazards: structural hazards, data hazards, and control hazards.
  - Structural hazards: attempt to use the same hardware to do two different things at once. (Need more hardware resources)
  - Data hazards: Instruction depends on result of prior instruction still in the pipeline. Read After Write (RAW), Write After Read (WAR), Write After Write (WAW). Need forwarding or compiler scheduling.

    RAW: aka "Dependence". True data dependence that needs attentions.

    WAR: aka "Antidependence". Can't happen in MIPS 5 stage pipeline.

    WAW: aka "Output dependence". Can't happen in MIPS 5 stage pipeline.
  - Control hazards: Caused by delay between the fetching of instructions and decisions about changes in control flow (branches and jumps).
- Speedup = $\frac{Pipeline\ depth}{1 + Pipeline\ Stall\ CPI}$
- Control hazards: four Branch Hazard Alternatives.
  - Stall.
  - Predict not taken. No penalty if prediction is correct
  - Predict taken. In MIPS **still incurs 1 cycle branch penalty** due to branch target address not calculated in time.
  - Delayed branch. Define branch to take place **AFTER** a following instruction. Still have 1 slot delay. (Can have predictions at this phase. If wrong, will need to redo and incur 1 cycle delay)
- Problems with pipelining: exception (an unusual event happens to an instruction during its execution) and interrupt (hardware signal to switch the processor to a new instruction stream) must happen between 2 instructions, or information will be gone.

Chapter 3

- ILP at software level (achieved by compilers):
  - Pipeline scheduling: Identify stalls between instructions; move instructions that do not have true data dependency on other instructions to fill in the gap
  - Loop Unrolling: unroll a number of times to eliminate stalls between instructions (only unroll the body code). !! Should attempt to do **delayed branch** which fills a stall cycle with an instruction that needs to be executed regardless.
  - Software pipelining: reorganizes loops so that each iteration is made from instructions chosen from different iterations of the original loop. (Need to include the maintenance code at the end)
- Branch prediction:
  - Basic 2 bit predictor
  - Correlating branch predictor (m,n): Branch History Register (BHR) has length m, 2^m Pattern History Tables (PHT), each predictor in PHT has length n. Use last several digits of the branch address as index; update BHR and PHT after each outcome is available (shift BHR to the left by 1, update PHT entries based on automatons)
  - Local predictor: Multiple n-bit predictors for each branch, identified by branch address, updated by automaton
  - Tournament predictor: combination of correlating branch predictor and local predictor, choose the best one among each (remember to update all 3 after each branch outcome)
- Dynamic scheduling: out-of-order execution, in-order issue. ILP at hardware level.
  - Tomasulo's algorithm: eliminate WAW and WAR data hazards through the use of reservation stations. See slides for example.
  - Tomasulo with speculation: same idea, but added a commit phase. Results are saved in Reorder Buffer (ROB). **SD won't write to the memory until the commit stage; no need to wait for branch result before executing the next loop**
  - May mix in multiple issuing.
- Branch Target Buffer (BTB): happened during IF stage. Stores PC the same way as caches. If the branch address is in the buffer, then fetch the predicted PC. Need to check: 1) if branch PC is in the buffer (hit/miss); 2) if prediction is correct (prediction accuracy).
  - Good: No penalty if prediction is correct
  - Bad: Need to flush and redo fetch if prediction is wrong