Lecture 8

Probably best to ignore laundry metaphors
Pipelining == throughput, not decrease in execution time for an instruction

1. All instructions are the same length
2. Instruction formats and symmetry (6-5-5-5-5-6 vs. 6-5-5-16 vs. 6-26)
3. Memory accesses only for loads and stores
4. Memory alignment

Simplicity favors regularity, make the common case fast

IF
ID – Decode and register fetch, also sign extend the constant
EX – Execution, address computation, or (branch completion?)
MEM – Memory access or R-type completion
WB – Memory read completion

Structural hazard – Single memory vs. I-cache and D-cache

Data hazard - (forwarding, bypassing), add/sub = forwarding (Fig. 6.5 page 377)
lw/sub, we can do forwarding (fig 6.6), but still need a bubble a.k.a pipeline stall a.k.a nop "load-use"
Example on 378-379

Control hazard, can try to resolve branch in stage 2, can then stall or predict (static/dynamic)

Bubble is not good enough, may need to flush the pipeline
*Assume Branch Not Taken
*Reduce Delay of Branches
    -doing compare in ID stage creates new structural and data hazards, need new hardware
    -May need stalls
*Dynamic branch prediction

-Branch prediction buffer vs. branch target buffer.

-2-bit (page 422)

-Correlating and tournament

*Branch delay slot is an alternative to branch target buffer (pg. 424, b and c special)