Lecture 22
July 2

Hazards

- Hazard solution 1: do nothing
- Data Hazard solution 2: stall
- Data Hazard solution 3: data forwarding
- Data Hazard solution 4: out of order execution
    - Consider these instructions:
        - 1. R0$\leftarrow$10
        - 2. R1$\leftarrow$R0+4
        - 3. R2$\leftarrow$20
    - if we rearrange tem n this order, there is no hazard
        - 1. R0$\leftarrow$10
        - 3. R2$\leftarrow$20
        - 2. R1$\leftarrow$R0+4
    - we can do this since instruction 3 doesn't depend on value from 1 or 2
    - in order to implement this the control unit must be able to:
        - detect a hazard
        - immediately fetch the next instruction if there is
        - decide if it can be executed out of order
        - start the execution
        - go back to the previous instruction in the next cycle
- Control hazard solution 1: Branch hazard stall
    - If we come to a conditional branch, don't put anything into the pipeline until we decide the branch
        - I.e. If we fetch a conditional branch, stall until it finishes

BRZ	IF	DOF	EX	WB			
		bubble	bubble	bubble	bubble		
			bubble	bubble	bubble	bubble	
next inst				IF	DOF	EX	WB

- Control hazard solution 2: Branch prediction
    - Problem: we don't know what the conditional branch will do with the PC
        - Solution: guess
    - The next instructions on the predicted path will be put into the pipe.
    - If the prediction was wrong, stop them!!!
        - Don't let them write back, RW, MW = 0
        - And STALL#$%#$%
    - If the branch is predicted correctly, there is no time penalty
    - The guess could be a default (always branch/ never branch) or suggested by the programmer (usually would always branch)

- Instruction Set Architectures
    - The instruction set architecture (ISA) is full description of the machine language, timing, I/O, ETC.
    - Each instruction has several “fields” typically:
        - Opcode—specifies the operation
        - Address(es)—locations of the operands
        - Mode—how should the address be interpreted?
- Register set
    - All registers visible to the programmer
        - I.e. not IR pipeline registers private reg…
    - includes general and special purpose, registers
    - special purpose
        - used for a particular reason
        - e.g. Stack pointer, processor status register
    - general purpose
        - used by the programmer for any purpose
            - load from memory intermediate calculation, etc
        - can be any number of these
            - could be 0, 1 or more
            - modern process: 16,3264