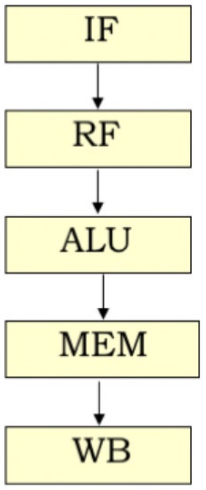
Ch 15

Pipelined Implementation

- Divide datapath in multiple pipeline stages to reduce t_{CK}
  - Each instruction executes over multiple cycles
  - Consecutive instructions are overlapped to keep CPI ≈ 1.0
- We'll study the classic 5-stage pipeline:



Instruction Fetch stage: Maintains PC, fetches instruction and passes it to

Register File stage: Reads source operands from register file, passes them to

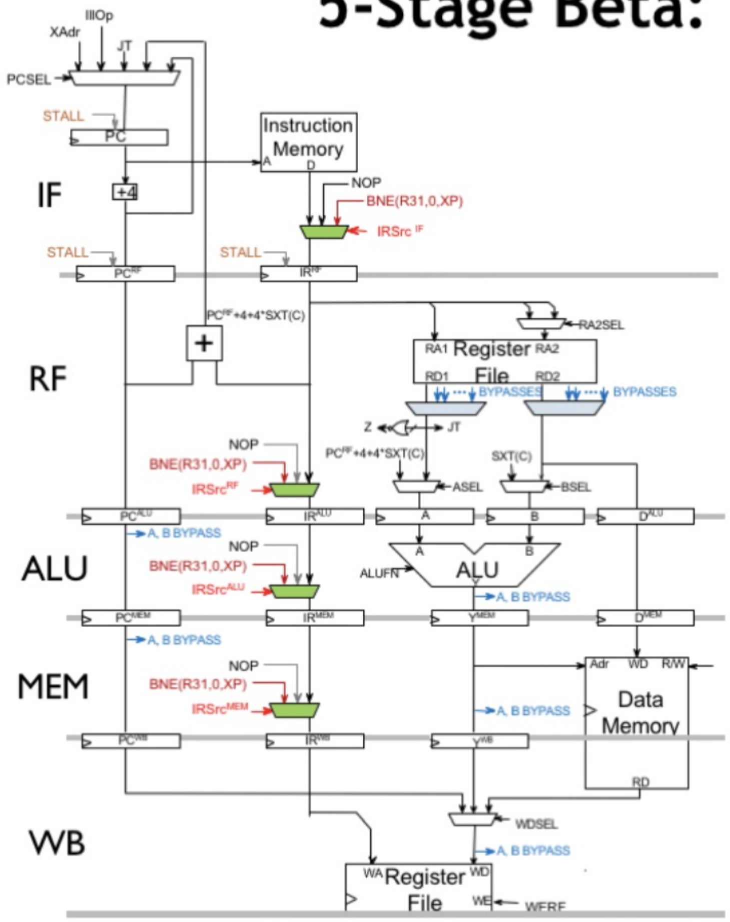
ALU stage: Performs indicated operation in ALU, passes result to

Memory stage: If it's a LD, use ALU result as an address, pass mem data (or ALU result if not LD) to

Write-Back stage: writes result back into register file.

$t_{CLK} = \max\{t_{IFETCH},\ t_{RF},\ t_{ALU},\ t_{MEM},\ t_{WB}\}$

5-Stage Beta: Final Version



- Data hazards:
  - Stall IF and RF (STALL=1 + IRSrc^{RF}=NOP)
  - Bypass
- Control hazards: Speculate PC+4 and
  - JMP or taken branch in RF,
    - IRSrc^{IF}=NOP
    - PCSEL → JT/branch target
  - If exception at stage X
    - IRSrc^{X}=BNE
    - Previous IRSrc^{X}=NOP
    - PCSEL → XAdr or IllOp
  - If interrupt
    - IRSrc^{IF}=BNE
    - PCSEL → XAdr

Reminder: Resolving Hazards

- Strategy 1: Stall. Wait for the result to be available by freezing earlier pipeline stages

- Strategy 2: Bypass. Route data to the earlier pipeline stage as soon as it is calculated

- Strategy 3: Speculate
  - Guess a value and continue executing anyway
  - When actual value is available, two cases
    - Guessed correctly → do nothing
    - Guessed incorrectly → kill & restart with correct value