

Chapter #14

Processor Structure and Function

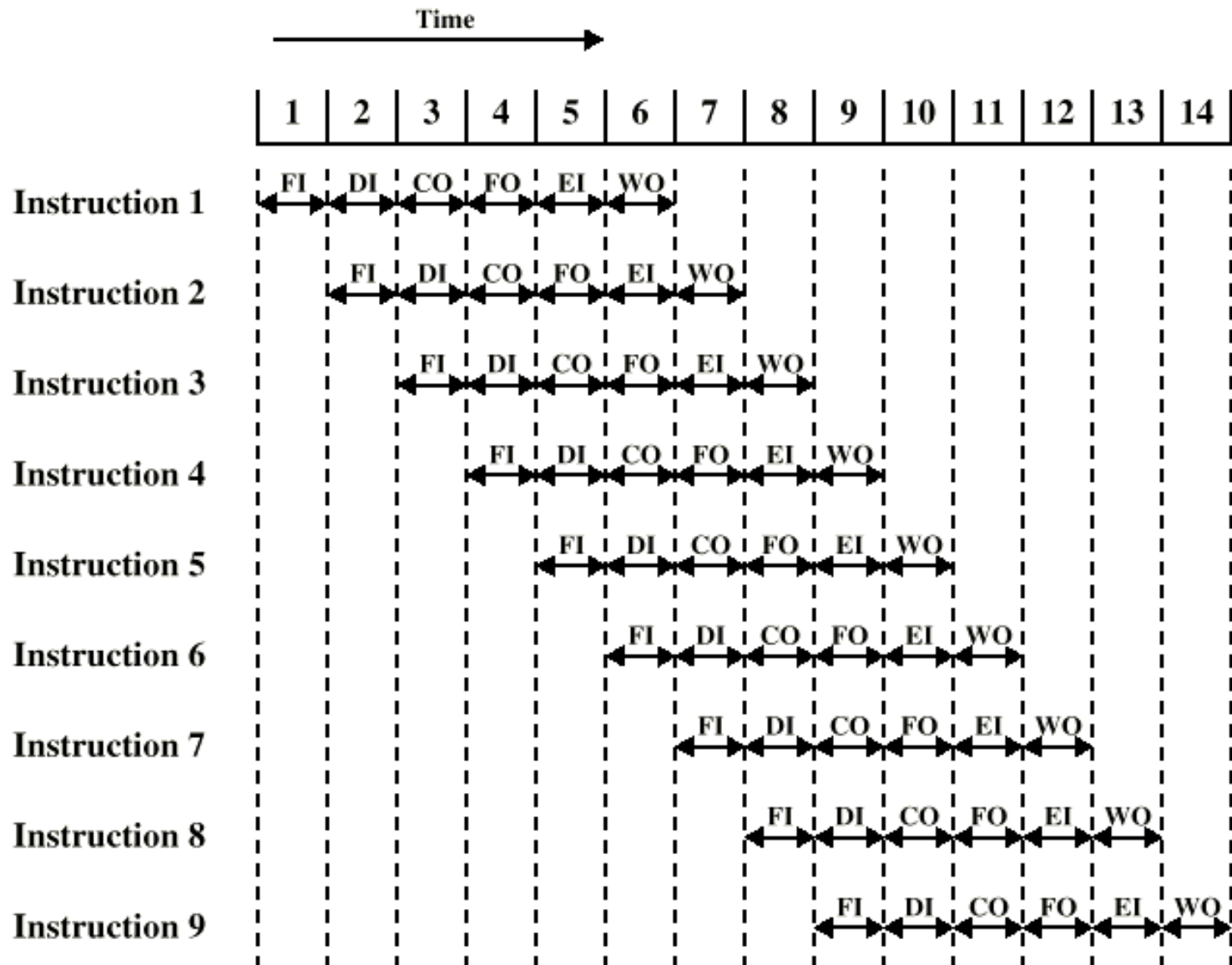
Instruction Pipelining

- Dividing instruction steps into stages
- Each step is clocked and intermediate instruction and/or register values are stored to allow processing of next instruction
- Adv:
 - Reduces idle time of units
 - Generally improves performance
- Disadv:
 - More complex architecture
 - Issues to resolve (branches, dependencies, etc.)

Pipelining Stages

- Varies on actual architecture
- General 6-stages:
 - Fetch instruction (FI)
 - Read next expected instruction into buffer
 - Decode instruction (DI)
 - Determine opcode and operands
 - Calculate operands (CO)
 - Determine address(es) of source operand(s)
 - Fetch operands (FO)
 - Get value from operand
 - Execute instruction (EI)
 - Perform operation
 - Write result (WO)
 - Store result from operation

Pipeline Timing Diagram



Pipeline Performance

- Parameters:
 - Number of instructions: n
 - Number of stages: k
 - Stage delay: t_m
- Performance:
 - Without pipelining: $n \times k \times t_m$
 - With pipelining: $k \times t_m + (n-1) \times t_m = (k+n-1) \times t_m$
- Speedup:
 - Speedup=(old time) / (new time) = $(n \times k) / (k+n-1)$
 - As $n \rightarrow \infty$, Speedup = k (Optimally)

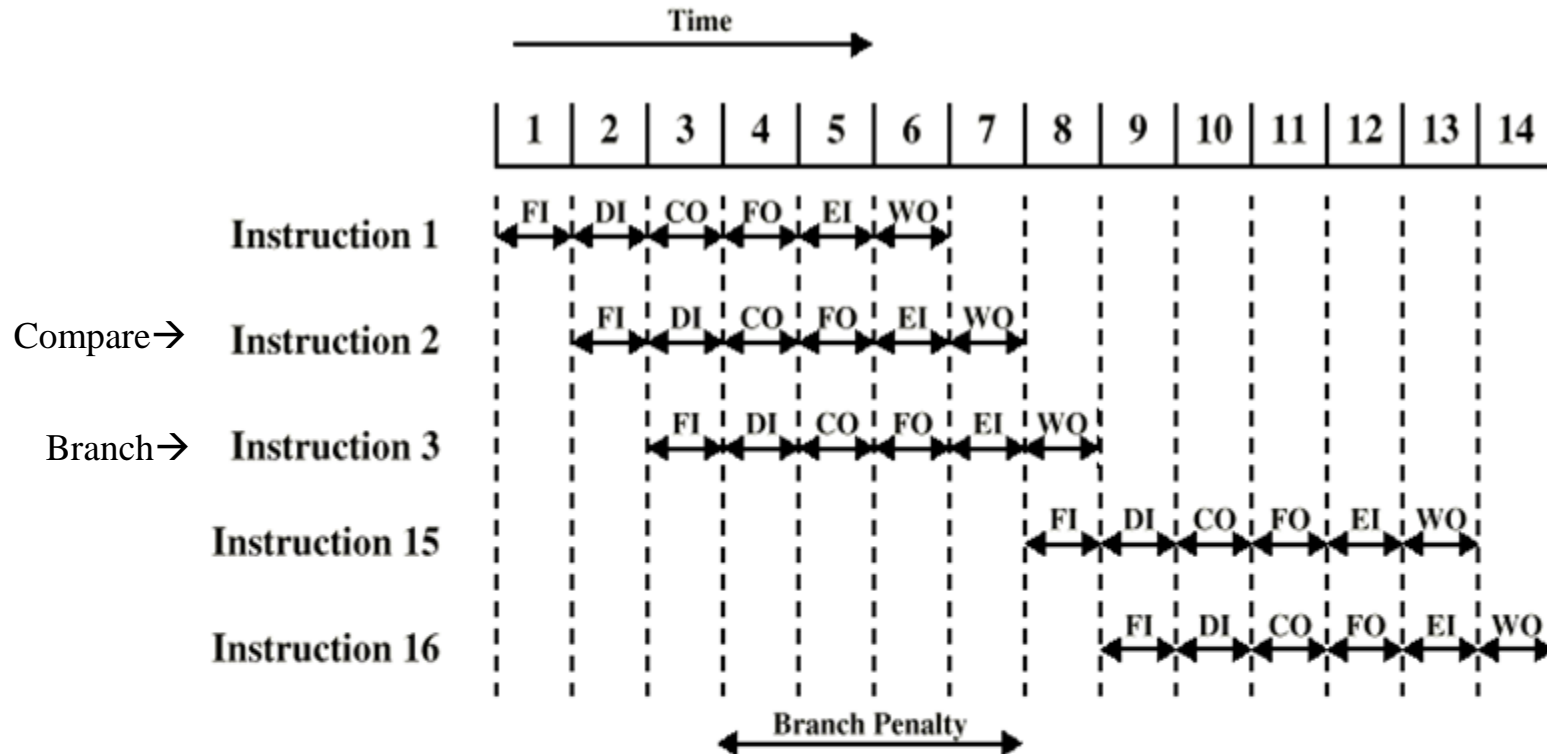
Branches

- Definition:
 - Jumps to another (usually not the next) instruction address
 - Account for about 20% of total instructions
- High-level language viewpoint:
 - If-then-else
 - Case/Switch
 - While-do/ Do-until / Repeat-until
 - For-do-loop
 - Exit / Break
- Basic types:
 - Unconditional
 - Conditional

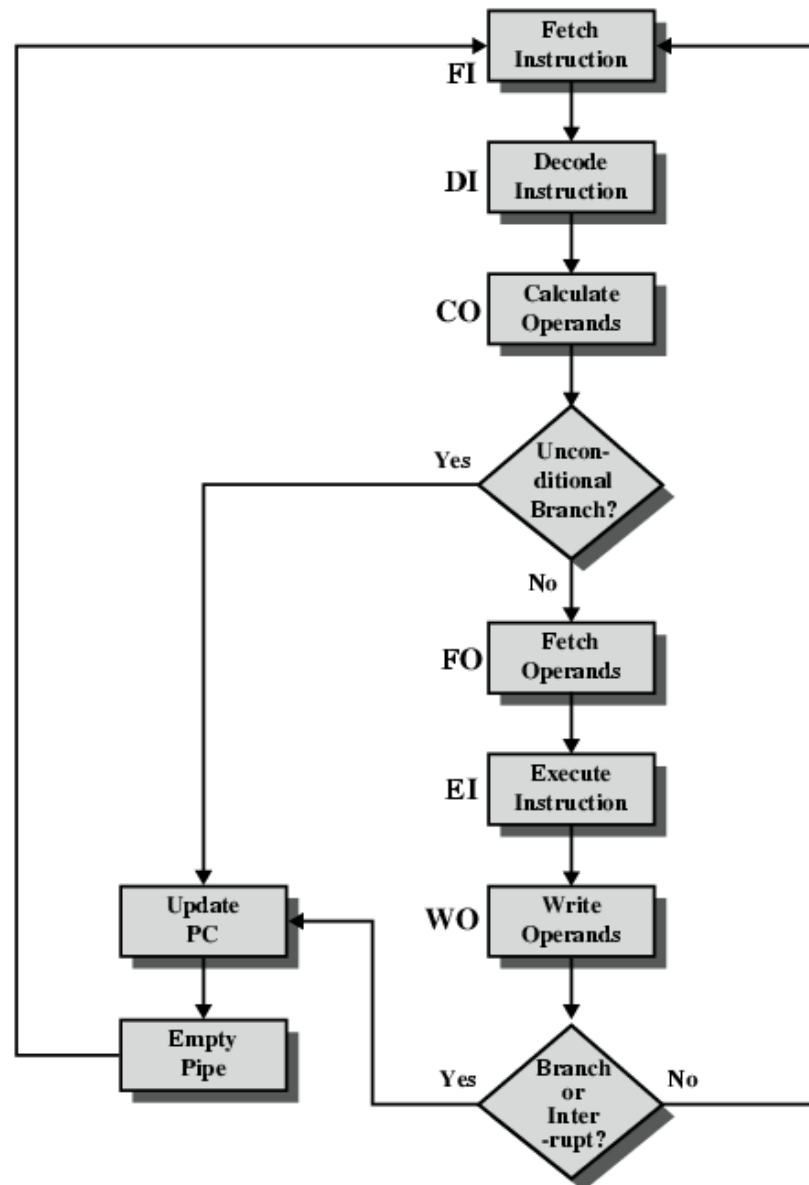
Effect of Branches on Pipeline

- Problem:
 - Unconditional branch:
 - Address label to jump to not known at proper time
(i.e. next instruction is fetched before jump instruction's operand is calculated)
 - Conditional branch:
 - Condition for jumping not known at proper time
 - (i.e. next several instructions are fetched before jump instruction's condition is executed)
- Solution:
 - Pipelining stalling
 - Branch prediction
 - Branch delaying

Pipeline Stalling w/ Conditional Branch



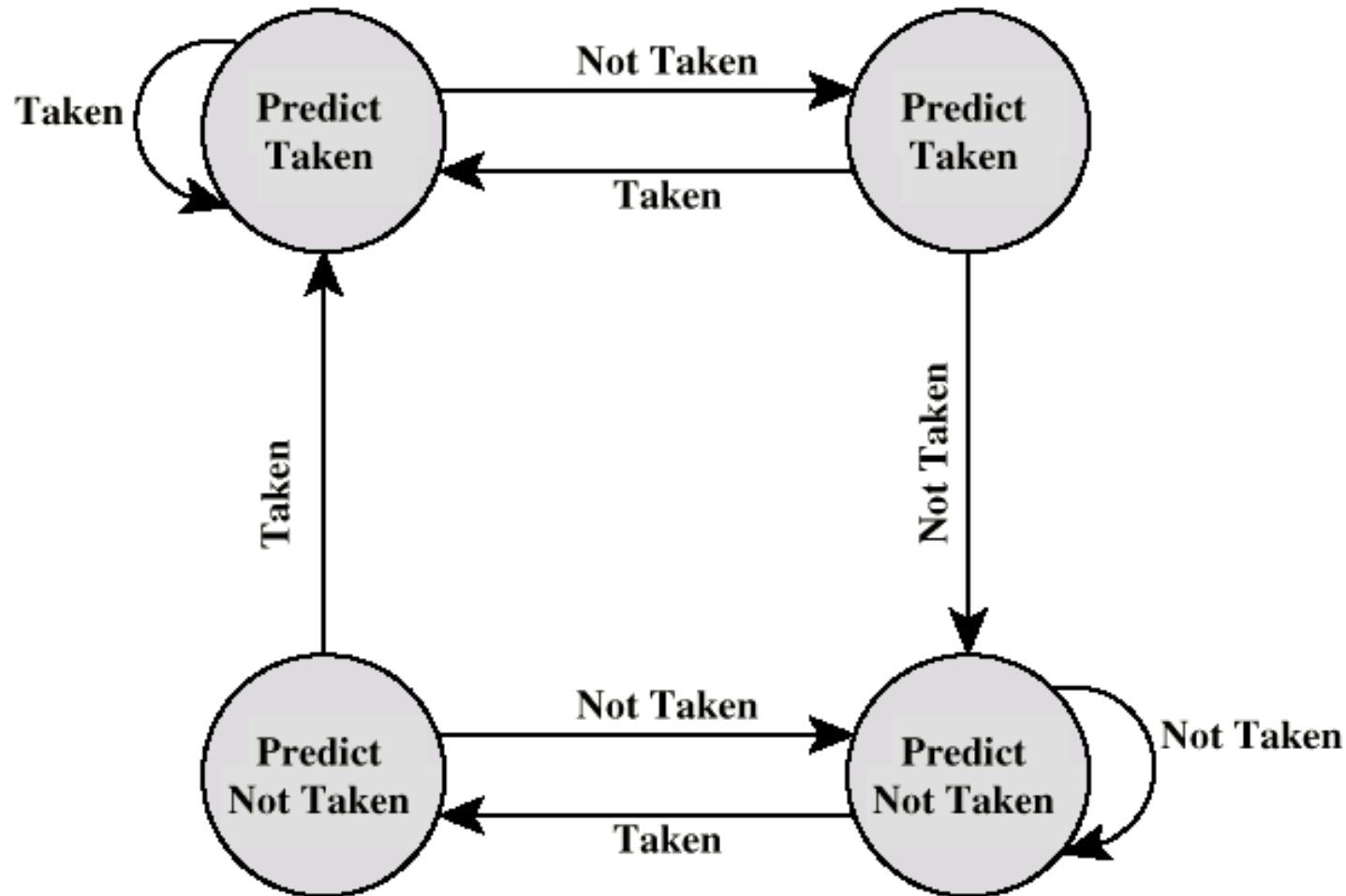
Six Stage Instruction Pipeline



Branch Prediction

- Predict never/always taken
 - Assume jump won't/will happen, always fetch next/target instruction
- Predict by Opcode
 - Some instructions more likely to be jump than others
- Predict by direction
 - Forward branches less likely to happen than backward branches
- Taken/Not taken switch
 - Based on previous history
- Problem:
 - Prediction can be wrong
- Solution:
 - Recovery from misprediction via scratch registers to restore values

2nd Chance Branch Prediction



Branch Delaying

- Definition:
 - Delaying branch until necessary (cannot be delayed any longer)
- Advantage:
 - Other later independent instructions may be issued/completed earlier
- Disadvantage:
 - More complex to implement
 - May result in same overall performance