

CSE 321a

Computer Organization (1)

تنظيم الحاسبات (1)



3rd year, Computer Engineering
Fall 2017

Lecture #11



Dr. Hazem Ibrahim Shehata

Dept. of Computer & Systems Engineering

Credits to Dr. Ahmed Abdul-Monem Ahmed for the slides

Administrivia

- Assignment #3:
 - To be released by the end of this week.

Website: <http://hshehata.github.io/courses/zu/cse321a>

Office hours: Sunday 1:00pm-2:00pm

Chapter 14. Processor Structure and Function

Outline

- Processor organization
- Register organization
- Instruction cycle
- Instruction pipelining
 - Pipelining strategy
 - Pipeline performance
 - Pipeline hazards
 - Dealing with branches
 - Intel 80486 Pipelining
- The x86 Processor Family
- The Arm Processor

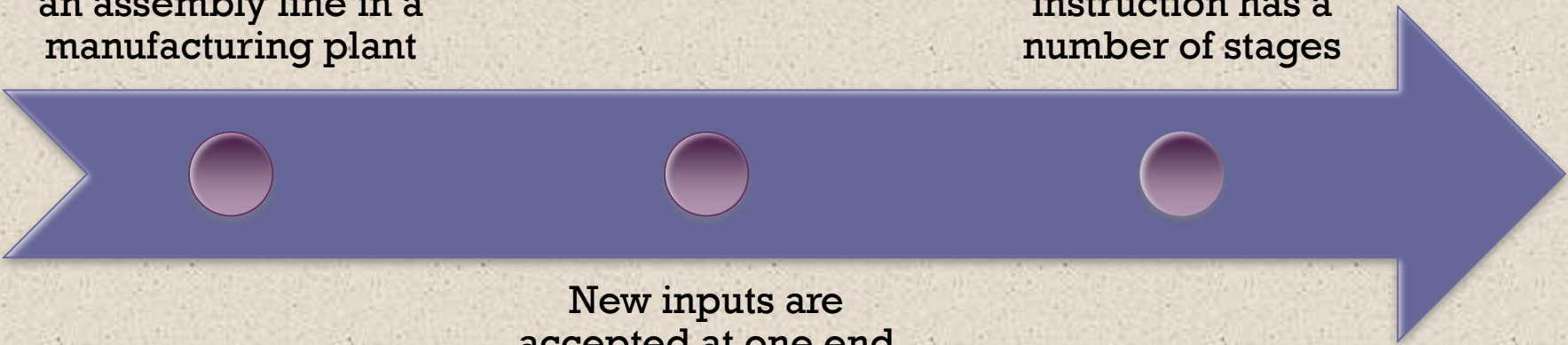
Outline

- ~~Processor organization~~
- ~~Register organization~~
- ~~Instruction cycle~~
- Instruction pipelining
 - Pipelining strategy
 - Pipeline performance
 - Pipeline hazards
 - Dealing with branches
 - ~~Intel 80486 Pipelining~~
- ~~The x86 Processor Family~~
- ~~The Arm Processor~~

Pipelining Strategy

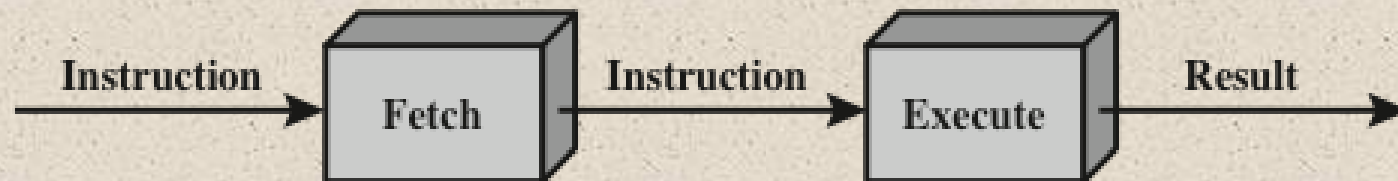
Similar to the use of
an assembly line in a
manufacturing plant

To apply this concept
to instruction
execution we must
recognize that an
instruction has a
number of stages

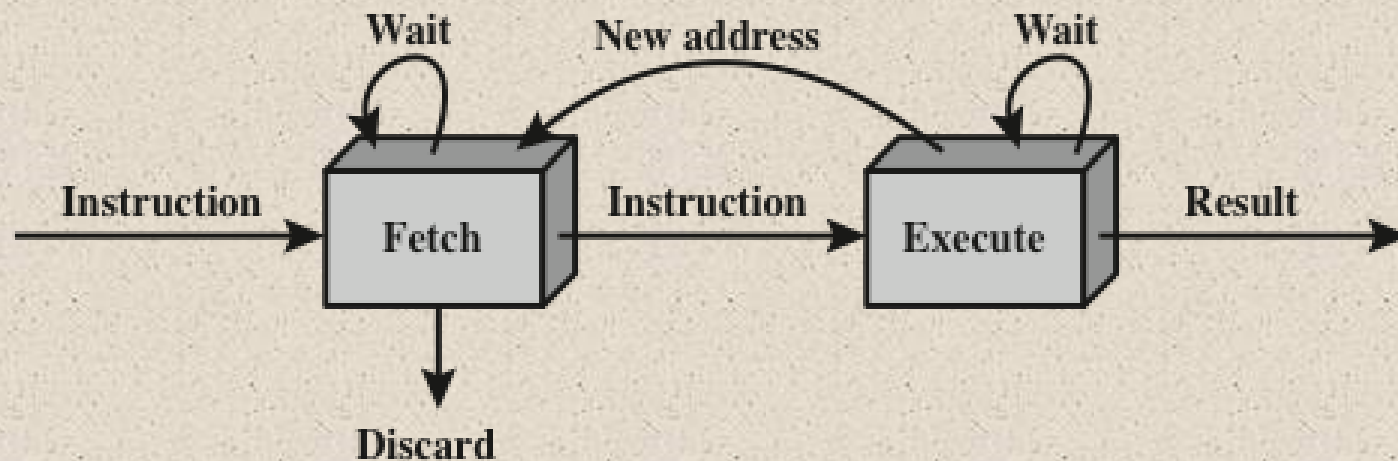


New inputs are
accepted at one end
before previously
accepted inputs
appear as outputs at
the other end

Two-Stage Instruction Pipeline



(a) Simplified view



(b) Expanded view

Figure 14.9 Two-Stage Instruction Pipeline



Additional Stages



- Fetch instruction (FI)
 - Read next expected instruction into a buffer
- Decode instruction (DI)
 - Determine the opcode and the operand specifiers
- Calculate operands (CO)
 - Calculate effective address of each source operand
 - This may involve displacement, register indirect, indirect, or other forms of address calculation
 - Read register operands
- Fetch operands (FO)
 - Fetch each operand from memory
- Execute instruction (EI)
 - Perform the indicated operation
 - Store result to destination register (if needed)
- Write operand (WO)
 - Store result in memory

Timing Diagram for Instruction Pipeline Operation

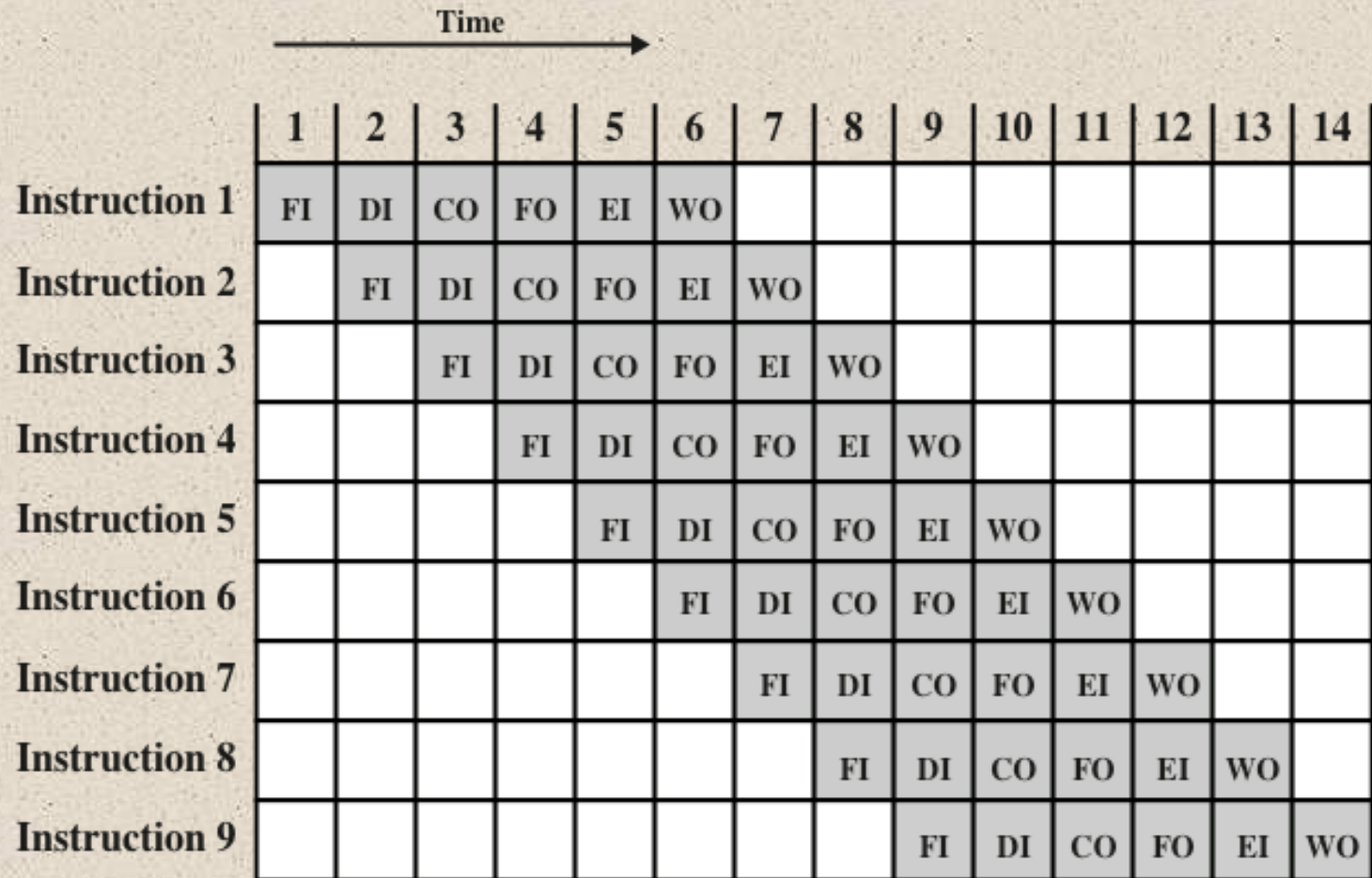


Figure 14.10 Timing Diagram for Instruction Pipeline Operation

The Effect of a Conditional Branch on Instruction Pipeline Operation

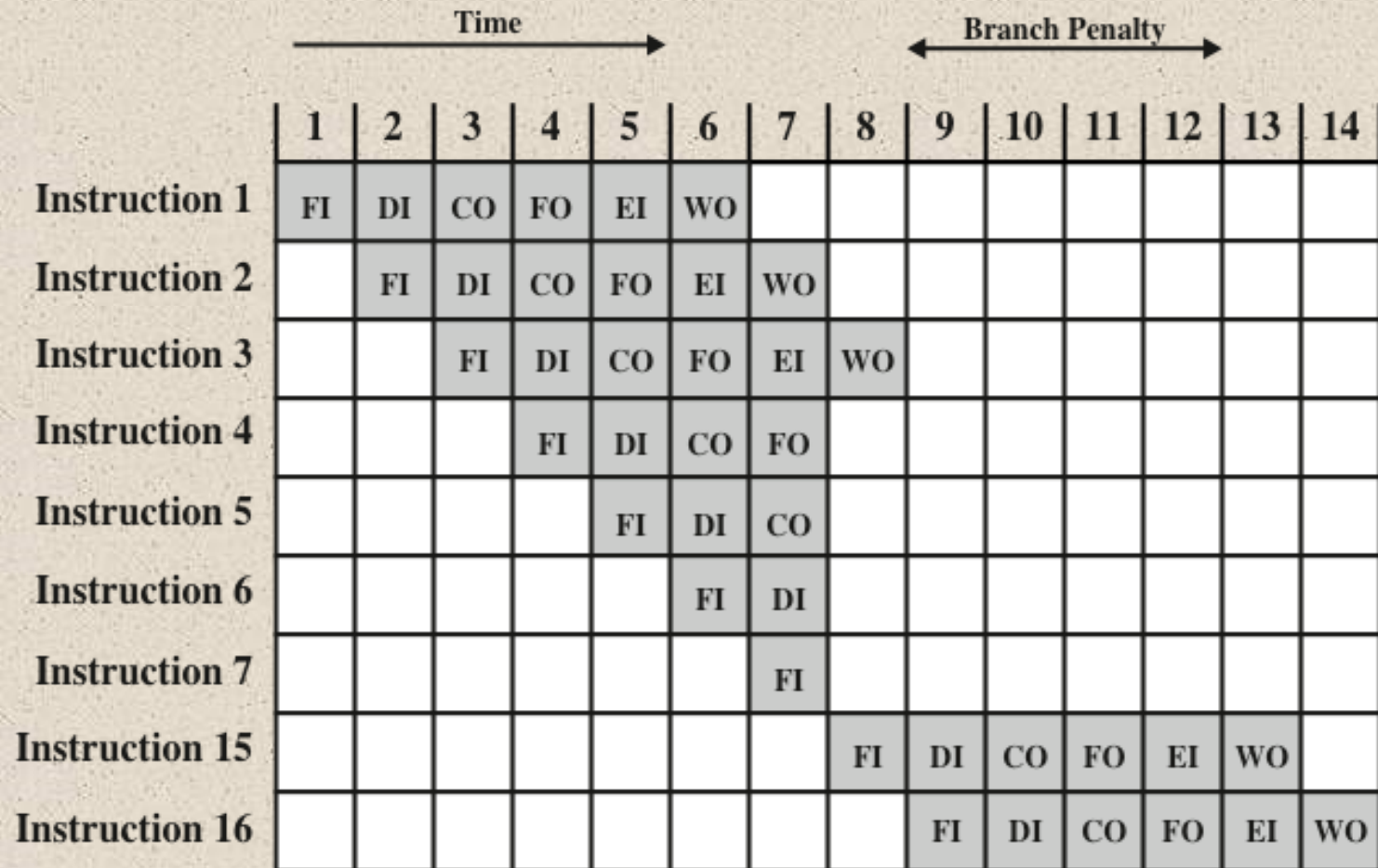


Figure 14.11 The Effect of a Conditional Branch on Instruction Pipeline Operation



Six Stage Instruction Pipeline

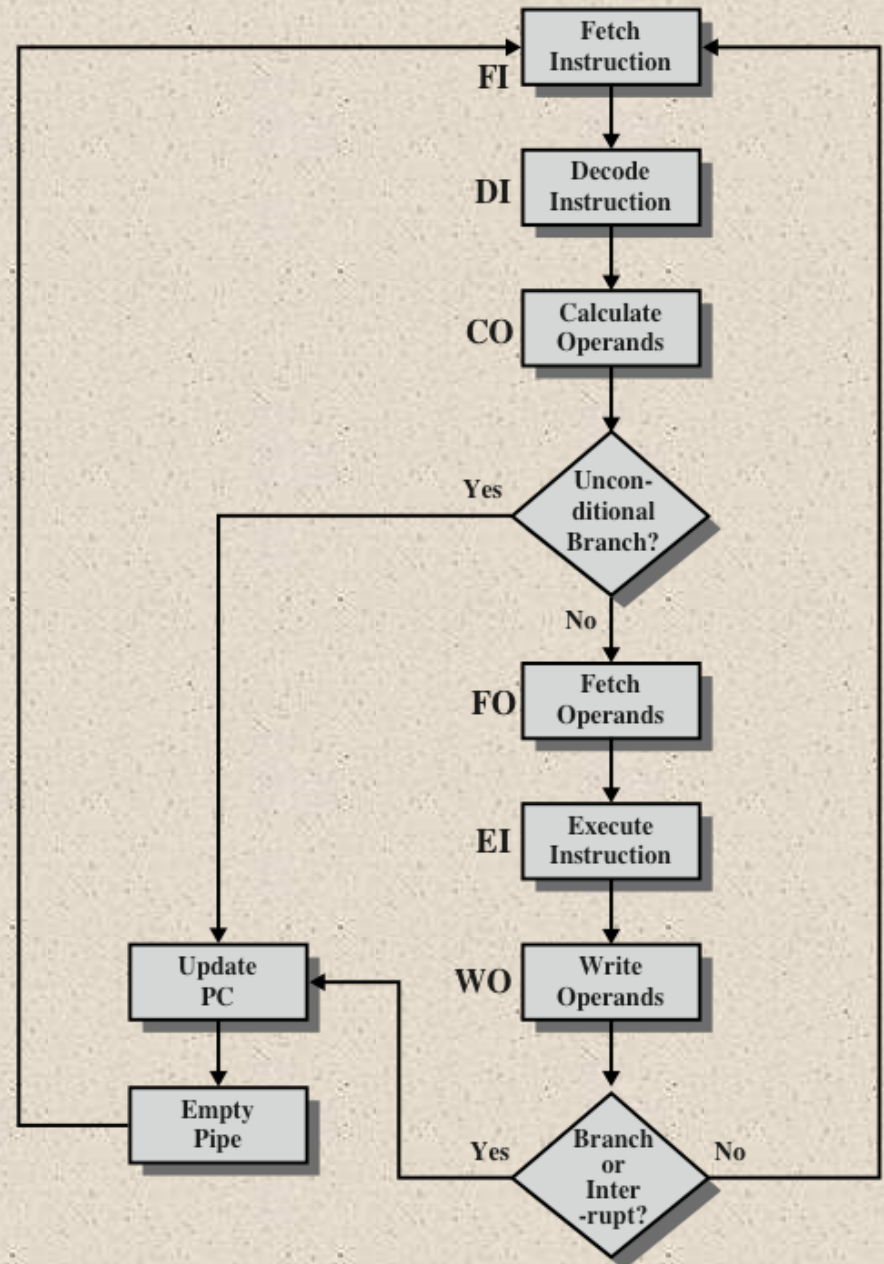


Figure 14.12 Six-Stage Instruction Pipeline



Speedup Factors with Instruction Pipelining

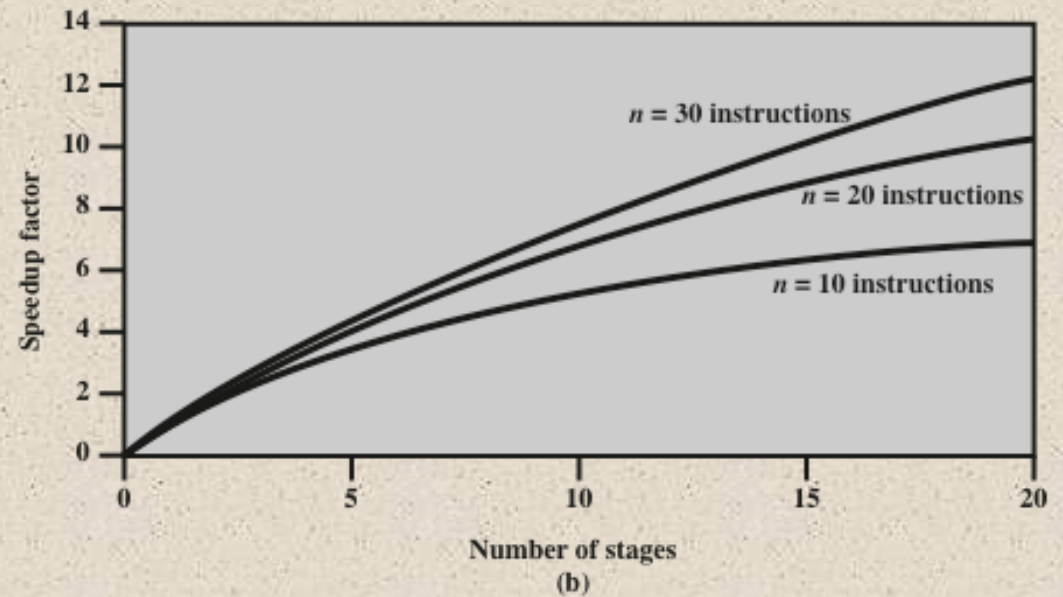
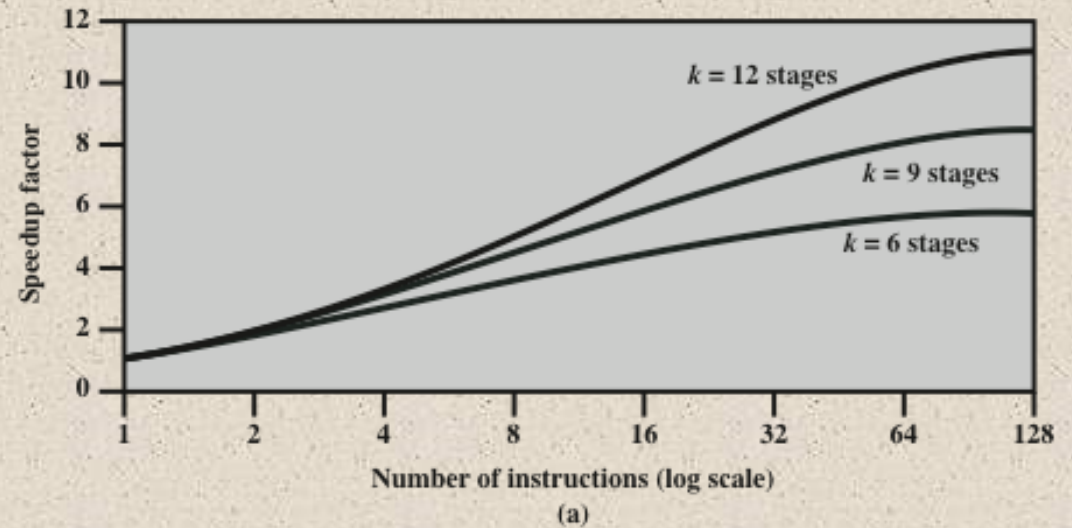


Figure 14.14 Speedup Factors with Instruction Pipelining

Pipeline Hazards

Occur when the pipeline, or some portion of the pipeline, must stall because conditions do not permit continued execution

There are three types of hazards:

- Resource
- Data
- Control

Also referred to as a *pipeline bubble*





Resource Hazards

A resource hazard occurs when two or more instructions that are already in the pipeline need the same resource

The result is that the instructions must be executed in serial rather than parallel for a portion of the pipeline

A resource hazard is sometimes referred to as a *structural hazard*

	Clock cycle								
	1	2	3	4	5	6	7	8	9
Instrucion	I1	FI	DI	FO	EI	WO			
	I2		FI	DI	FO	EI	WO		
	I3			FI	DI	FO	EI	WO	
	I4				FI	DI	FO	EI	WO

(a) Five-stage pipeline, ideal case

		Clock cycle								
		1	2	3	4	5	6	7	8	9
Instrucion	I1	FI	DI	FO	EI	WO				
	I2		FI	DI	FO	EI	WO			
	I3			Idle	FI	DI	FO	EI	WO	
	I4					FI	DI	FO	EI	WO

(b) I1 source operand in memory

Figure 14.15 Example of Resource Hazard

		Clock cycle								
		1	2	3	4	5	6	7	8	9
ADD EAX, EBX		FI	DI	FO	EI	WO				
SUB ECX, EAX			FI	DI		FO	EI	WO		
I3				FI		DI	FO	EI	WO	
I4						FI	DI	FO	EI	WO

RAW

Hazard

Figure 14.16 Example of Data Hazard

+

Data Hazards

A data hazard occurs when there is a conflict in the access of an operand location



Types of Data Hazard



- Read after write (RAW), or true dependency
 - An instruction modifies a register or memory location
 - Succeeding instruction reads data in memory or register location
 - Hazard occurs if the read takes place before write operation is complete
- Write after read (WAR), or antidependency
 - An instruction reads a register or memory location
 - Succeeding instruction writes to the location
 - Hazard occurs if the write operation completes before the read operation takes place
- Write after write (WAW), or output dependency
 - Two instructions both write to the same location
 - Hazard occurs if the write operations take place in the reverse order of the intended sequence



Control Hazard



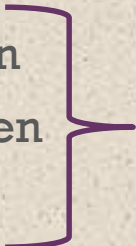

- Also known as a *branch hazard*
- Occurs when the pipeline makes the wrong decision on a branch prediction
- Brings instructions into the pipeline that must subsequently be discarded
- Dealing with Branches:
 - ~~Multiple streams~~
 - ~~Prefetch branch target~~
 - ~~Loop buffer~~
 - Branch prediction
 - ~~Delayed branch~~



Branch Prediction



- Various techniques can be used to predict whether a branch will be taken:

- | | | |
|---|--|--|
| <ul style="list-style-type: none">1. Predict never taken2. Predict always taken3. Predict by opcode |  | <ul style="list-style-type: none">■ These approaches are static■ They do not depend on the execution history up to the time of the conditional branch instruction |
| <ul style="list-style-type: none">1. Taken/not taken switch2. Branch history table |  | <ul style="list-style-type: none">■ These approaches are dynamic■ They depend on the execution history |



Branch Prediction Flow Chart

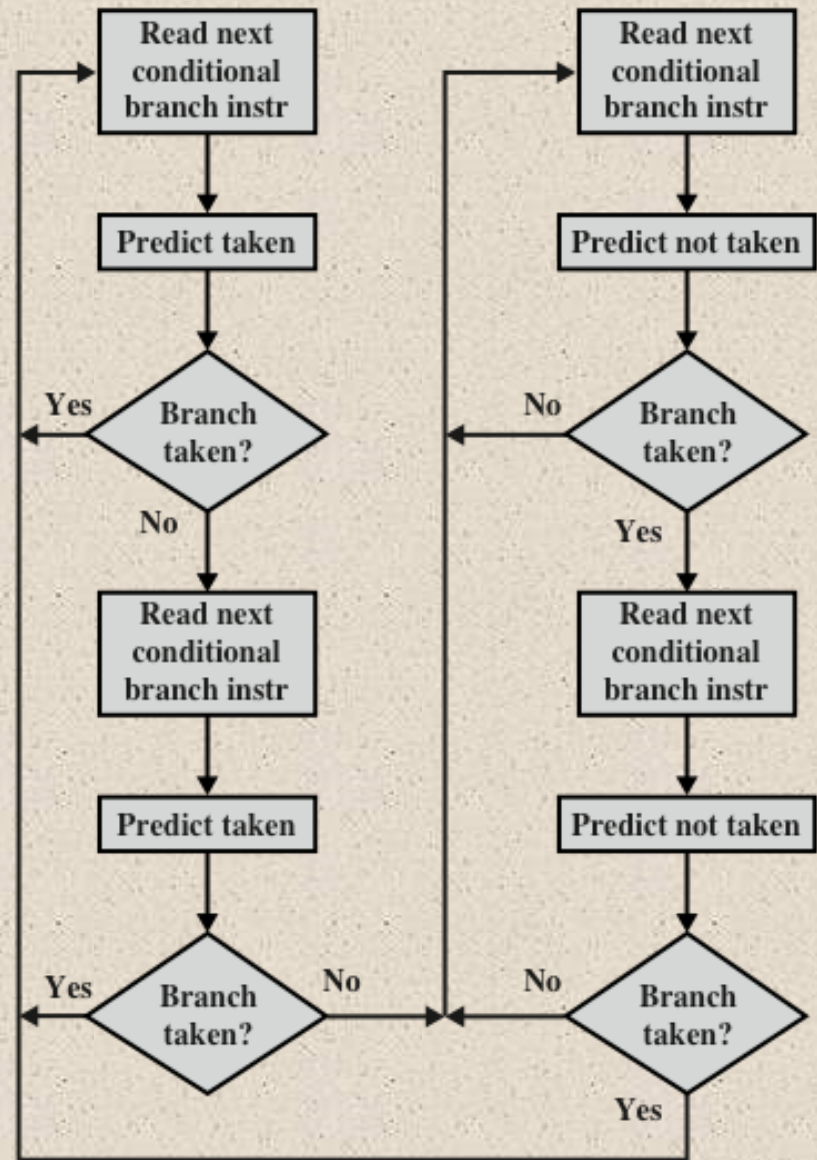


Figure 14.18 Branch Prediction Flow Chart

Branch Prediction State Diagram

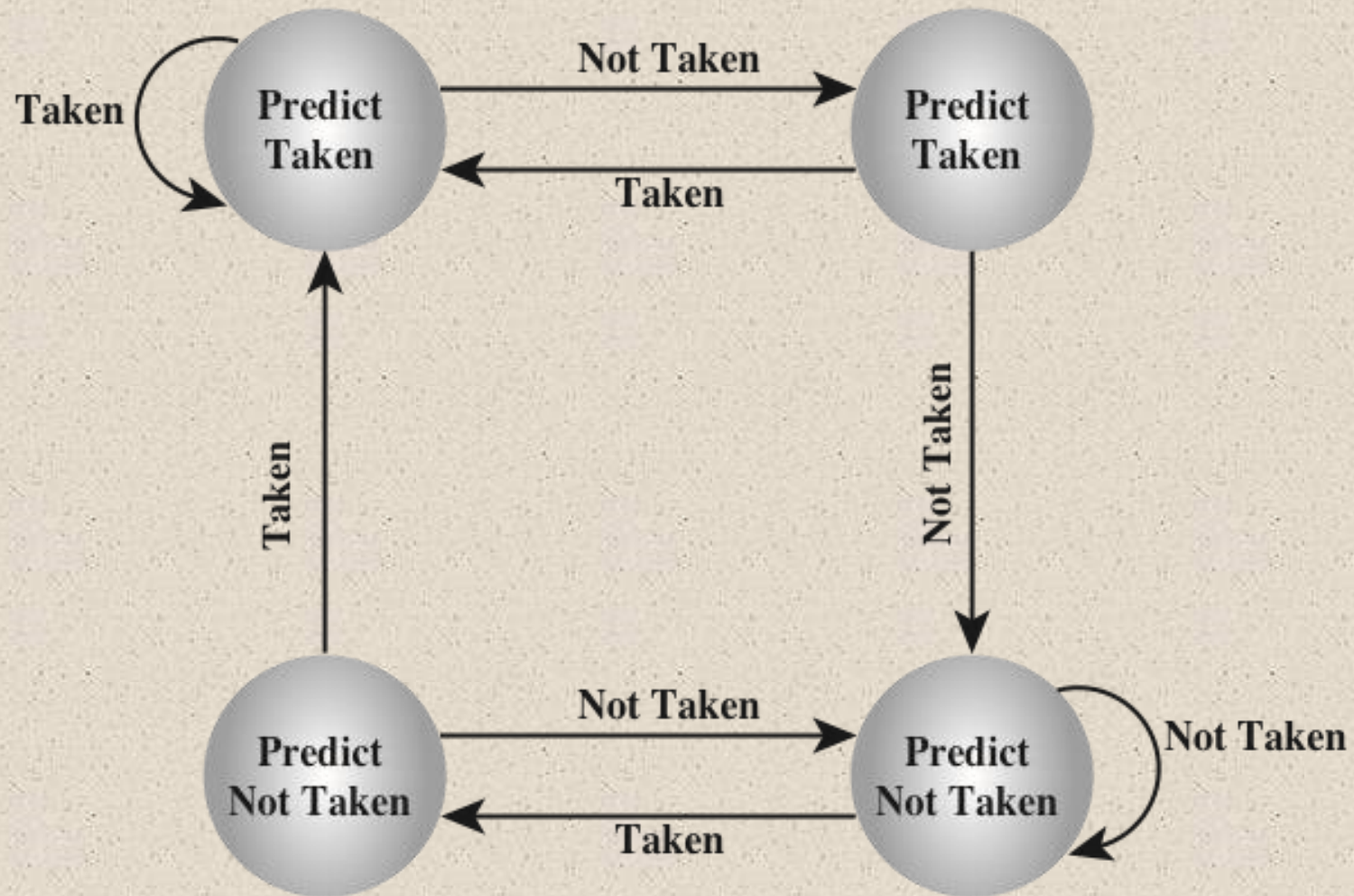
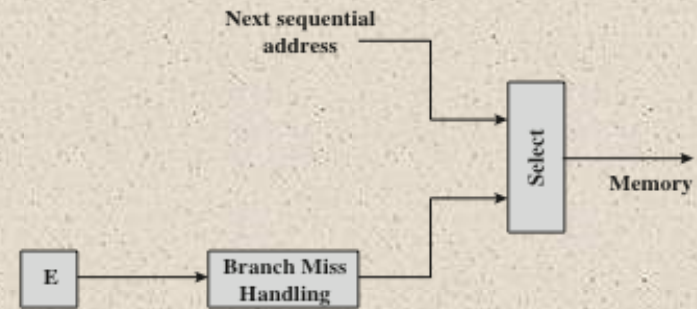


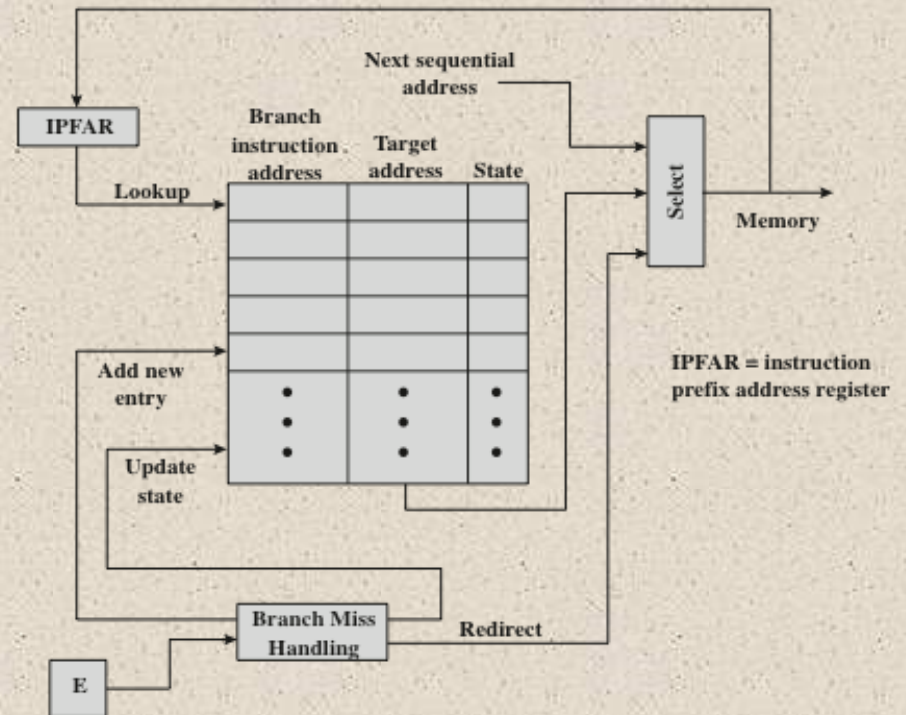
Figure 14.19 Branch Prediction State Diagram



Dealing With Branches



(a) Predict never taken strategy



(b) Branch history table strategy

Figure 14.20 Dealing with Branches

Reading Material

- Stallings, Chapter 14:
 - Pages 495-504
 - Pages 506-510