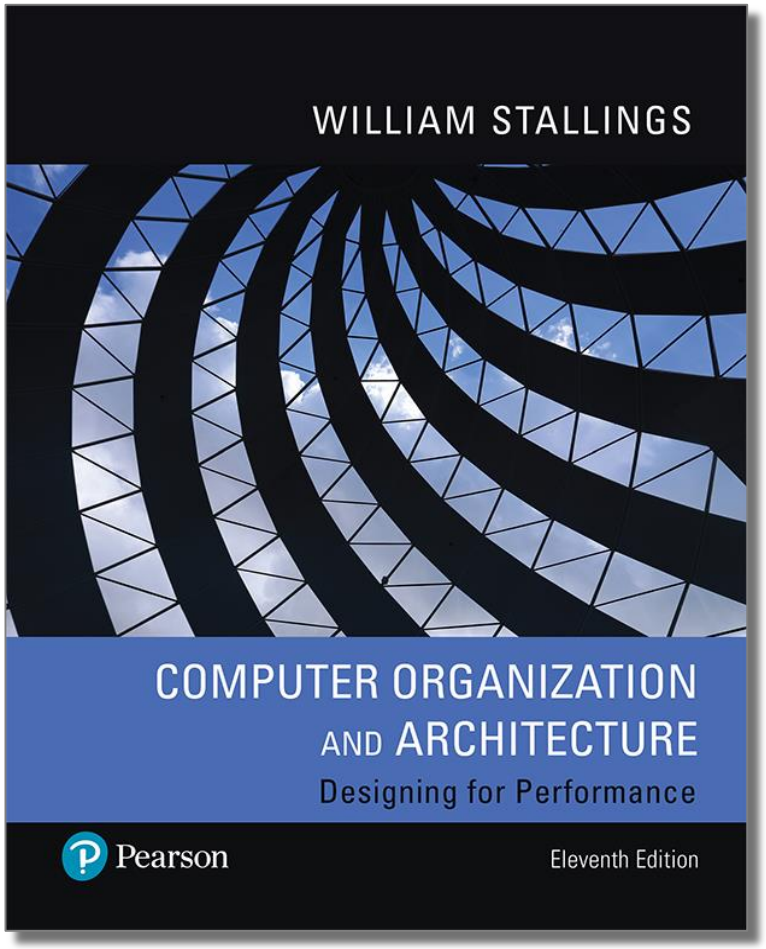
Computer Organization and Architecture

Designing for Performance

11th Edition



Chapter 16

Processor Structure and Function

Processor Organization

Processor Requirements:

- Fetch instruction
  - The processor reads an instruction from memory (register, cache, main memory)
- Interpret instruction
  - The instruction is decoded to determine what action is required
- Fetch data
  - The execution of an instruction may require reading data from memory or an I/O module
- Process data
  - The execution of an instruction may require performing some arithmetic or logical operation on data
- Write data
  - The results of an execution may require writing data to memory or an I/O module
- In order to do these things the processor needs to store some data temporarily and therefore needs a small internal memory

Figure 16.4
Instruction Cycle State Diagram

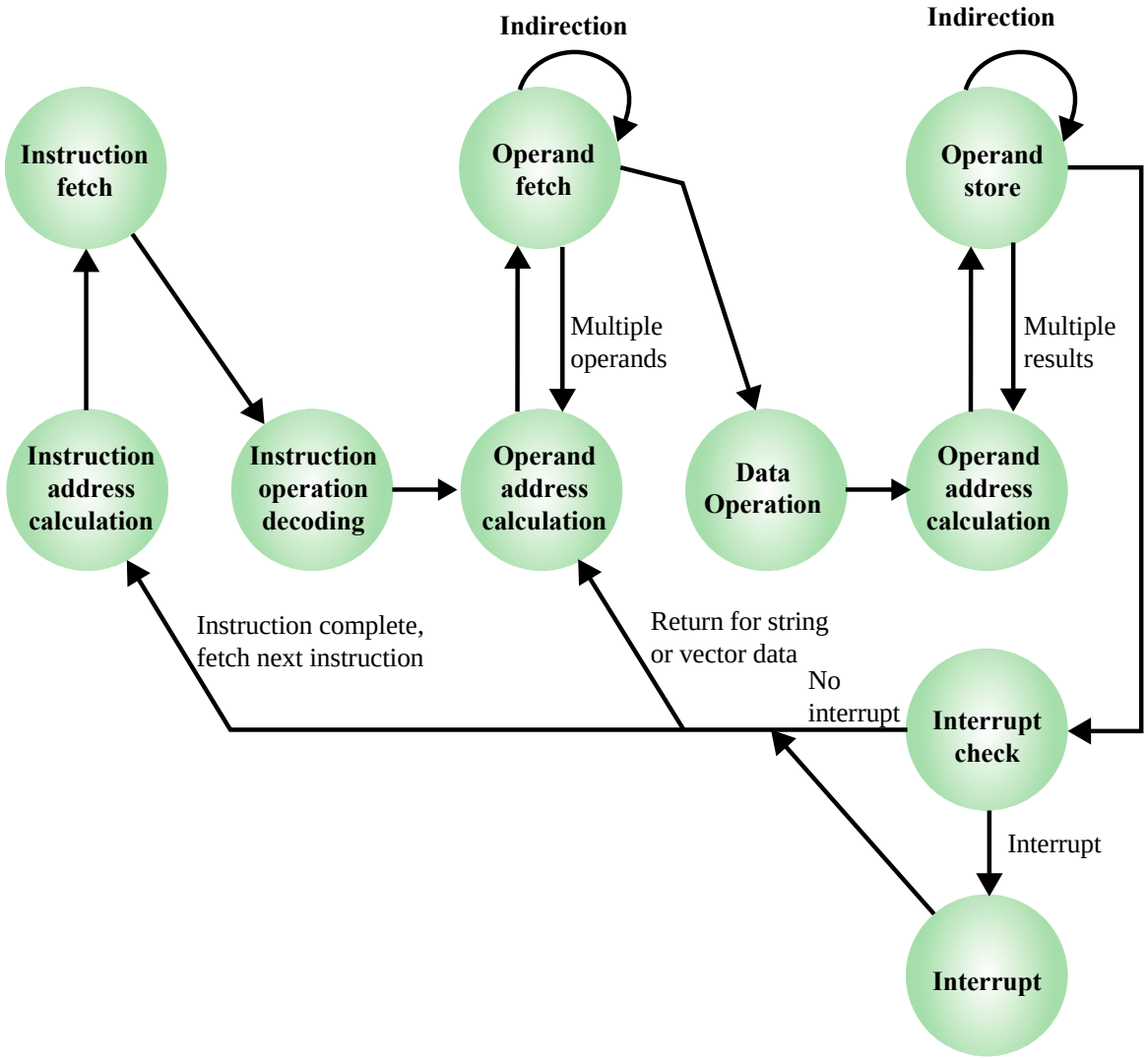
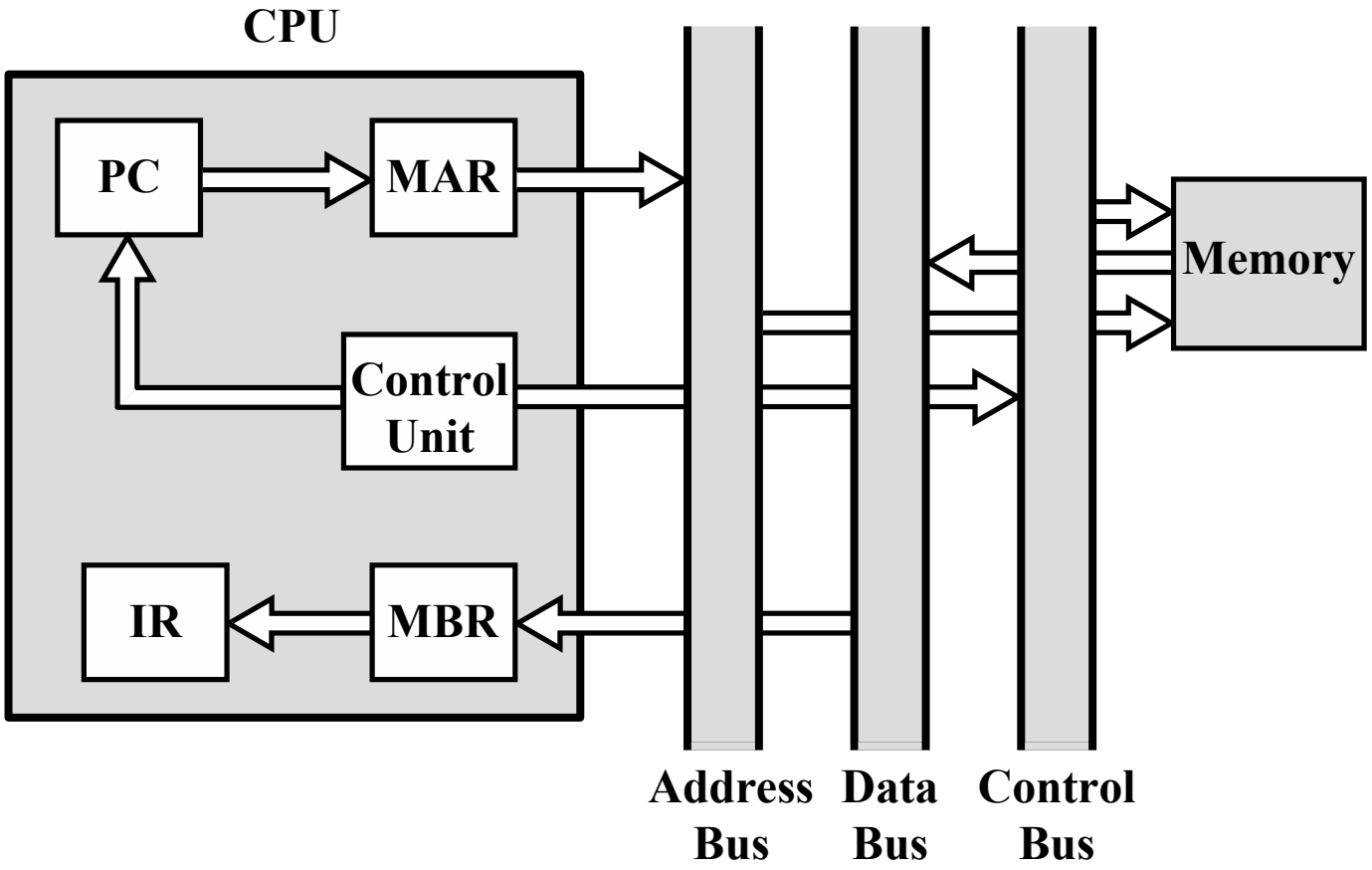


Figure 16.5
Data Flow, Fetch Cycle



MBR = Memory buffer register
MAR = Memory address register
IR = Instruction register
PC = Program counter

Figure 16.6
Data Flow, Indirect Cycle

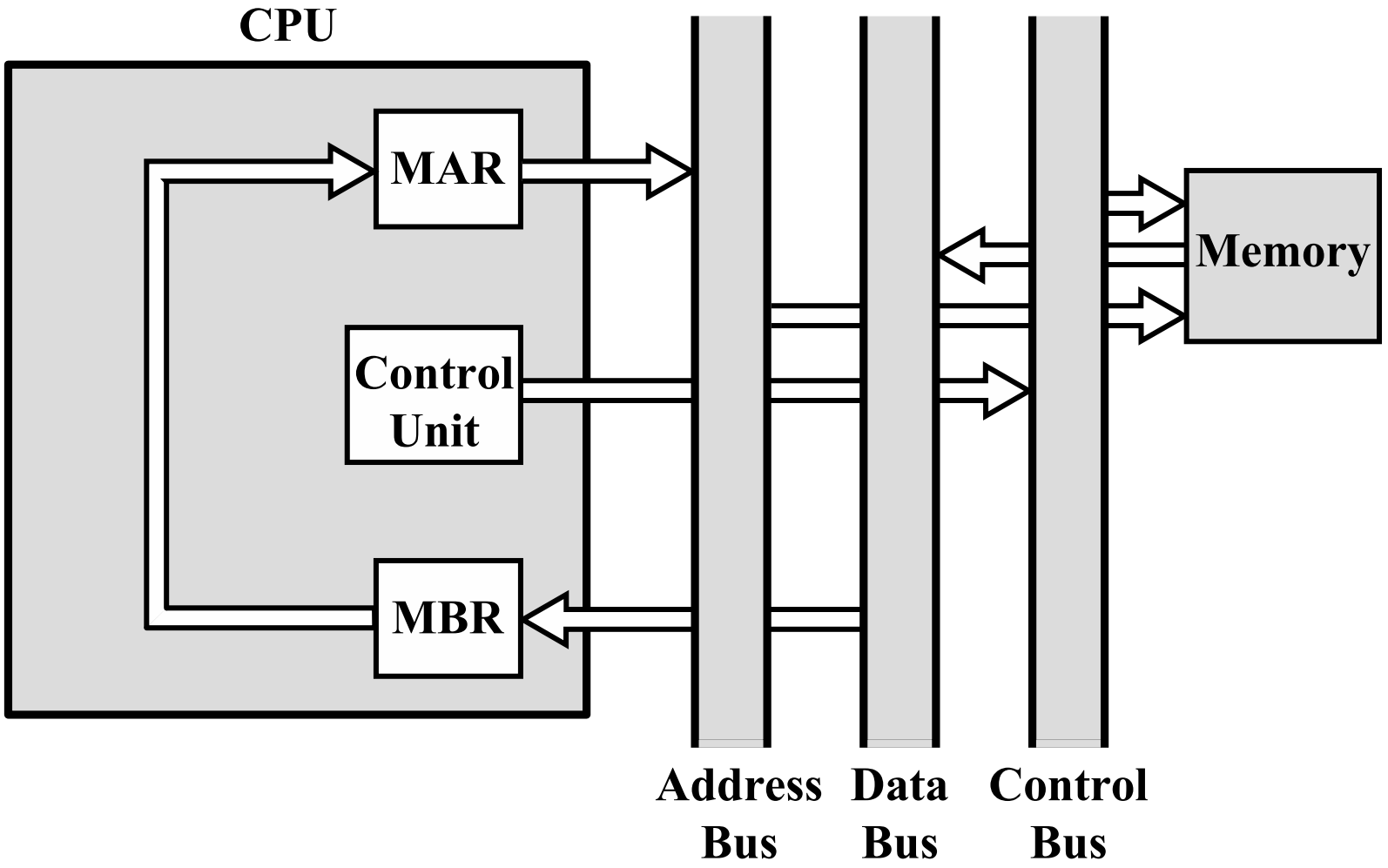
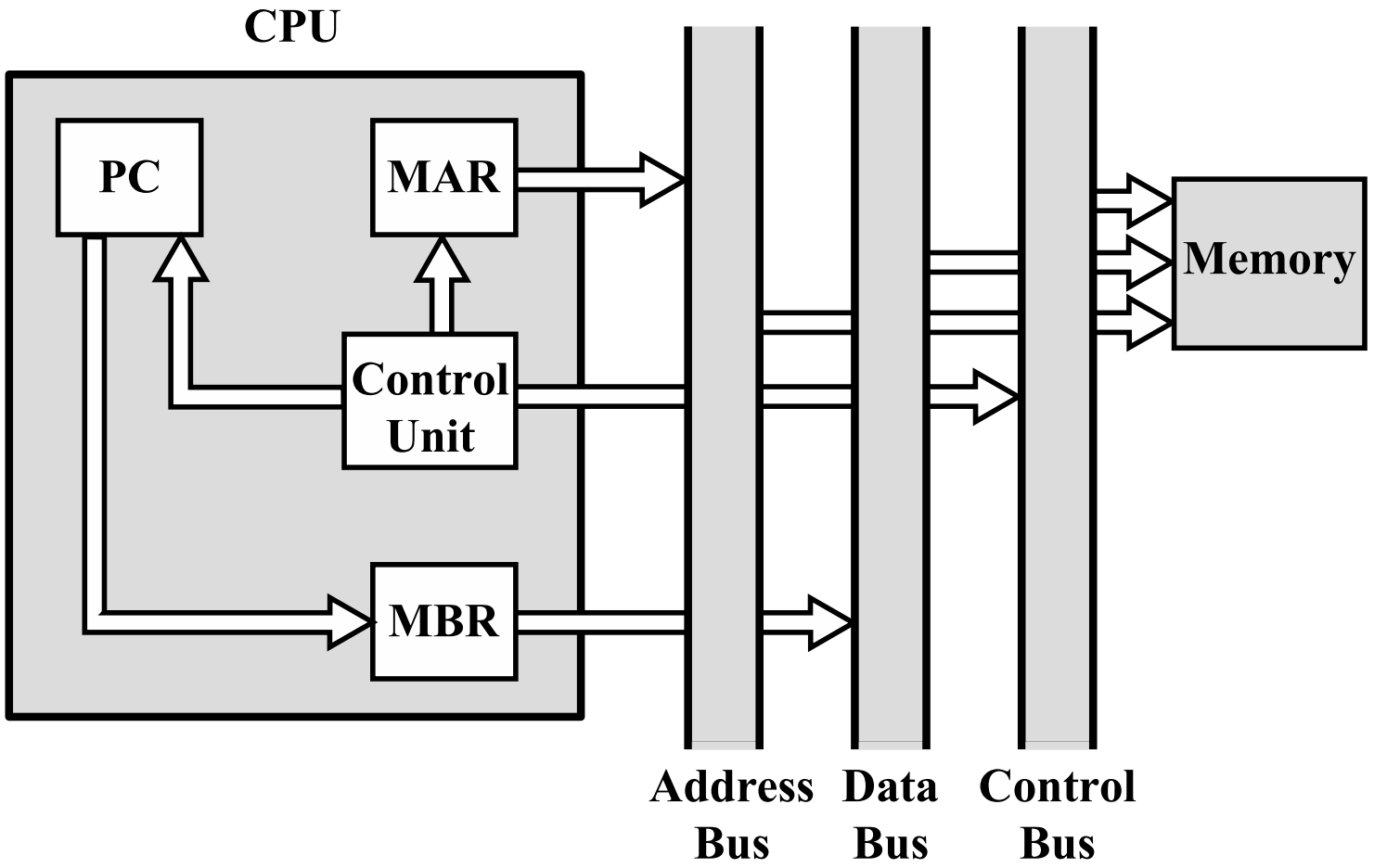
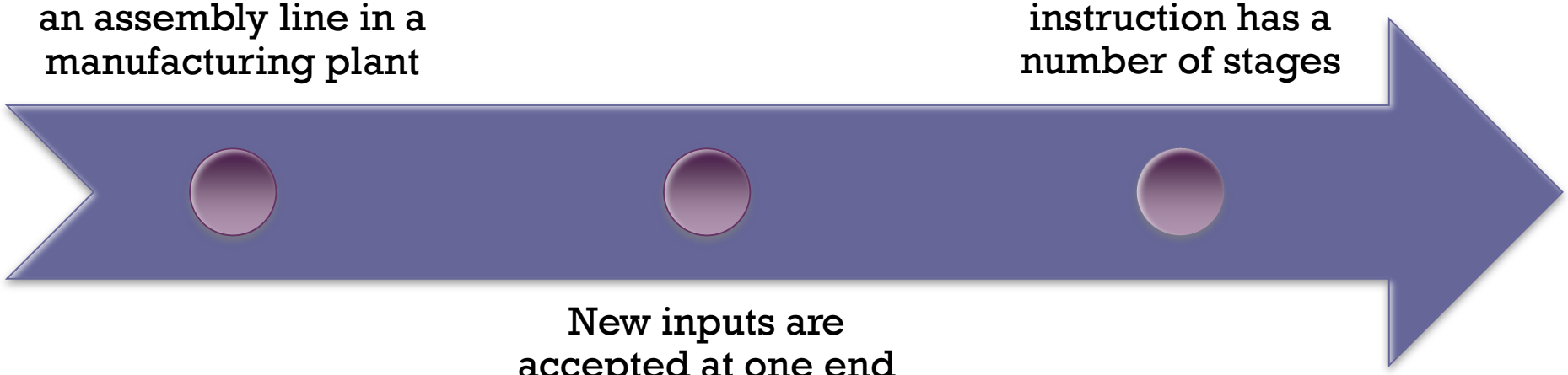


Figure 16.7
Data Flow, Interrupt Cycle



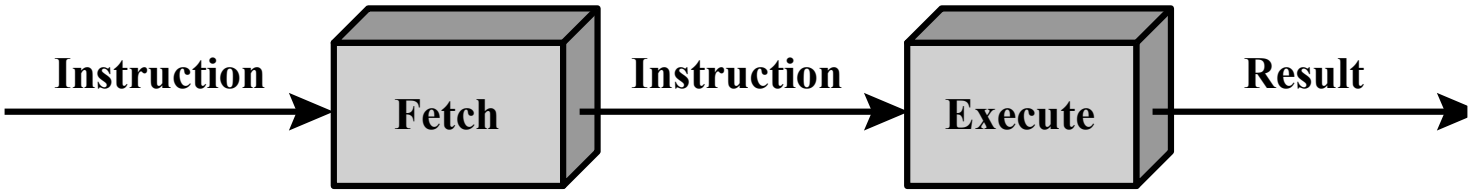
Pipelining Strategy



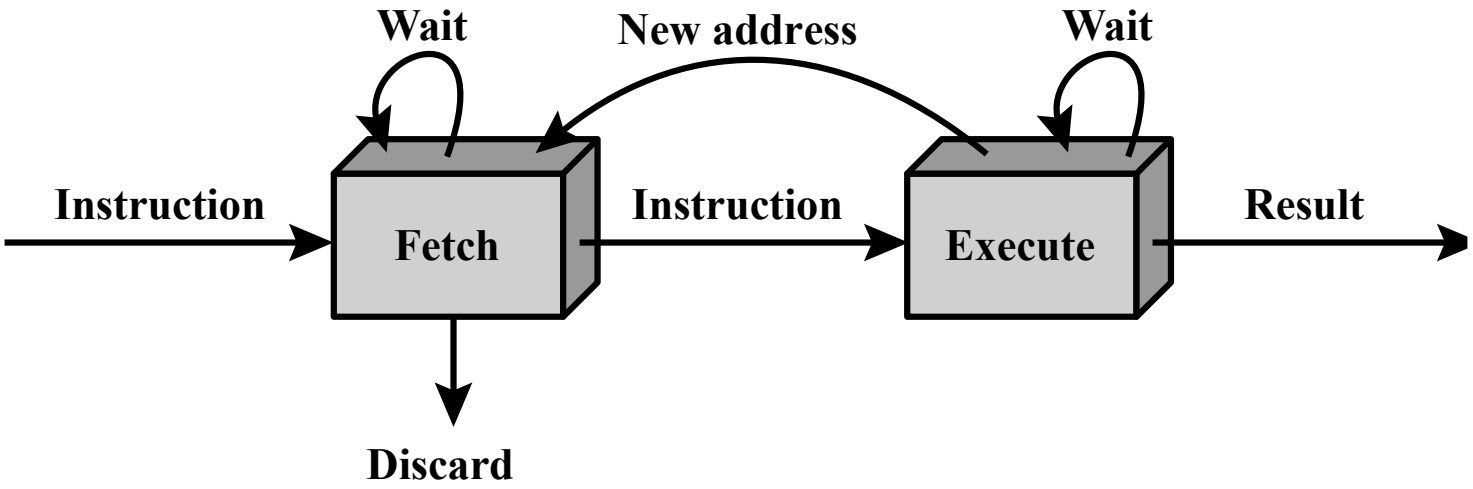
What is Pipelining

- Instruction pipelining is similar to the use of an assembly line in a manufacturing plant.
- An assembly line takes advantage of the fact that a product goes through various stages of production. By laying the production process out in an assembly line, products at various stages can be worked on simultaneously.
- To apply this concept to instruction execution, we must recognize that, in fact, an instruction has a number of stages.

Figure 16.8
Two-Stage Instruction Pipeline



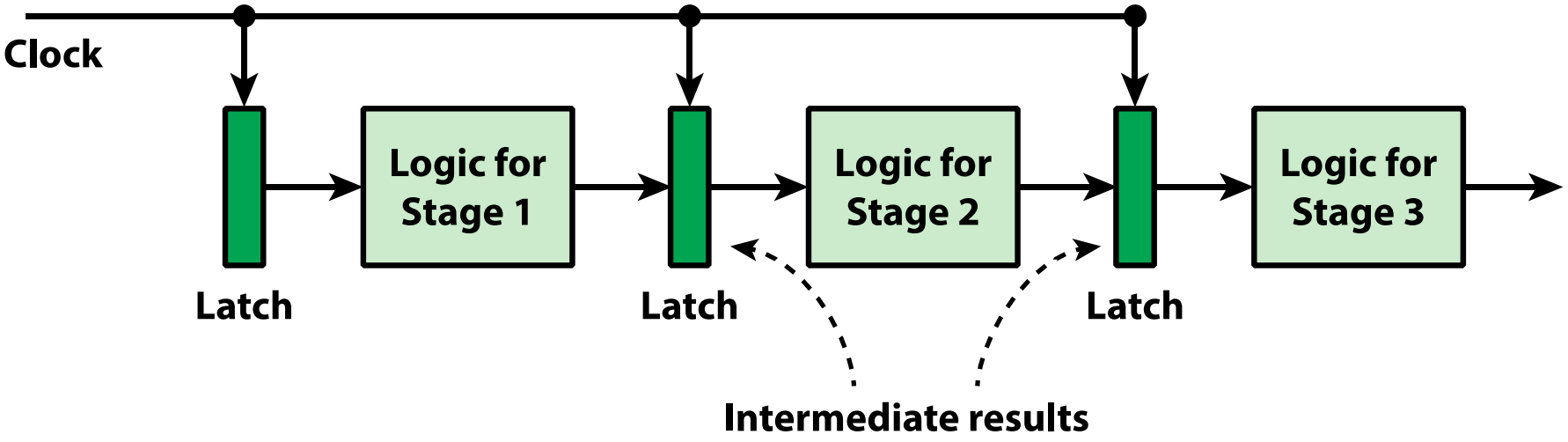
(a) Simplified view



Two-Stage Instruction Pipeline

- Consider subdividing instruction processing into two stages: fetch instruction and execute instruction.
- There are times during the execution of an instruction when main memory is not being accessed.
- This time could be used to fetch the next instruction in parallel with the execution of the current one.
- While the second stage is executing the instruction, the first stage takes advantage of any unused memory cycles to fetch and buffer the next instruction.
- Note that this approach, which involves instruction buffering, requires more registers.

Figure 16.9
Simplified Pipeline Architecture



In general, pipelining requires fast registers, called latches , that store intermediate values between stages. The latches serve to decouple the stages from each other.

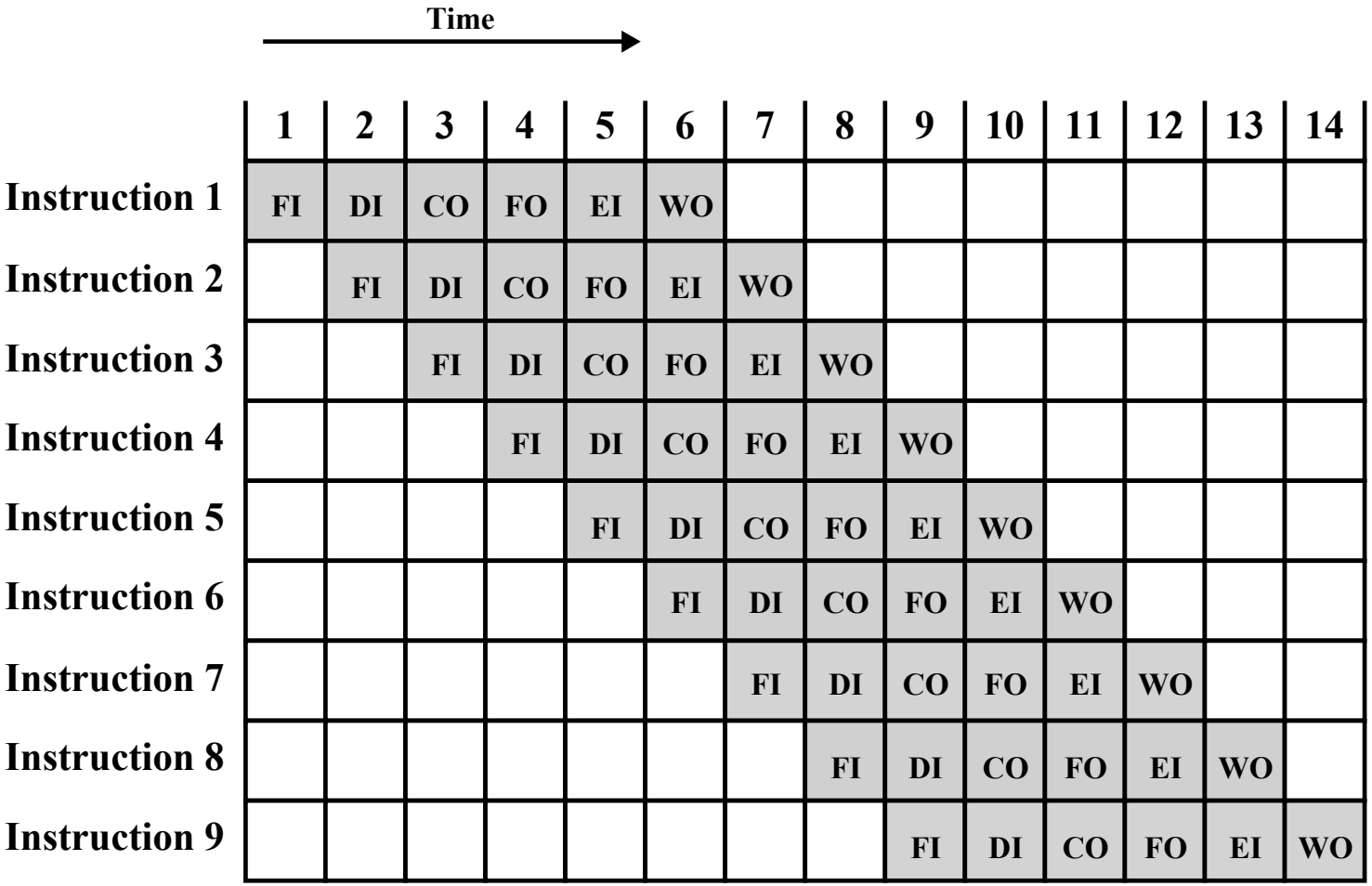
Two-Stage Instruction Pipeline

- Does it mean that we can double the execution rate?
- Unlikely for two reasons:
  1. The execution time will generally be longer than the fetch time. Execution will involve reading and storing operands and the performance of some operation. Thus, the fetch stage may have to wait for some time before it can empty its buffer.
  2. A conditional branch instruction makes the address of the next instruction to be fetched unknown. Thus, the fetch stage must wait until it receives the next instruction address from the execute stage. The execute stage may then have to wait while the next instruction is fetched.
- To gain further speedup, the pipeline must have more stages.

Additional Stages

- Fetch instruction (FI)
  - Read the next expected instruction into a buffer
- Decode instruction (DI)
  - Determine the opcode and the operand specifiers
- Calculate operands (CO)
  - Calculate the effective address of each source operand
  - This may involve displacement, register indirect, indirect, or other forms of address calculation
- Fetch operands (FO)
  - Fetch each operand from memory
  - Operands in registers need not be fetched
- Execute instruction (EI)
  - Perform the indicated operation and store the result, if any, in the specified destination operand location
- Write operand (WO)
  - Store the result in memory

Figure 16.10
Timing Diagram for Instruction Pipeline Operation



Instruction Pipeline Operation Enhancement Factors

- The diagram assumes that each instruction goes through all six stages of the pipeline which is not always the case.
- For example, a load instruction does not need the WO stage. However, to simplify the pipeline hardware, the timing is set up assuming that each instruction requires all six stages.
- Also, the diagram assumes that all of the stages can be performed in parallel. In particular, it is assumed that there are no memory conflicts. For example, the FI, FO, and WO stages involve a memory access. The diagram implies that all these accesses can occur simultaneously.

Instruction Pipeline Operation Enhancement Factors

- Recall other factors serve to limit the performance enhancement.
  - If the six stages are not of equal duration, there will be some waiting involved at various pipeline stages.
  - Conditional branch instruction, which can invalidate several instruction fetches. A similar unpredictable event is an interrupt.

Figure 16.11
The Effect of a Conditional Branch on Instruction Pipeline Operation

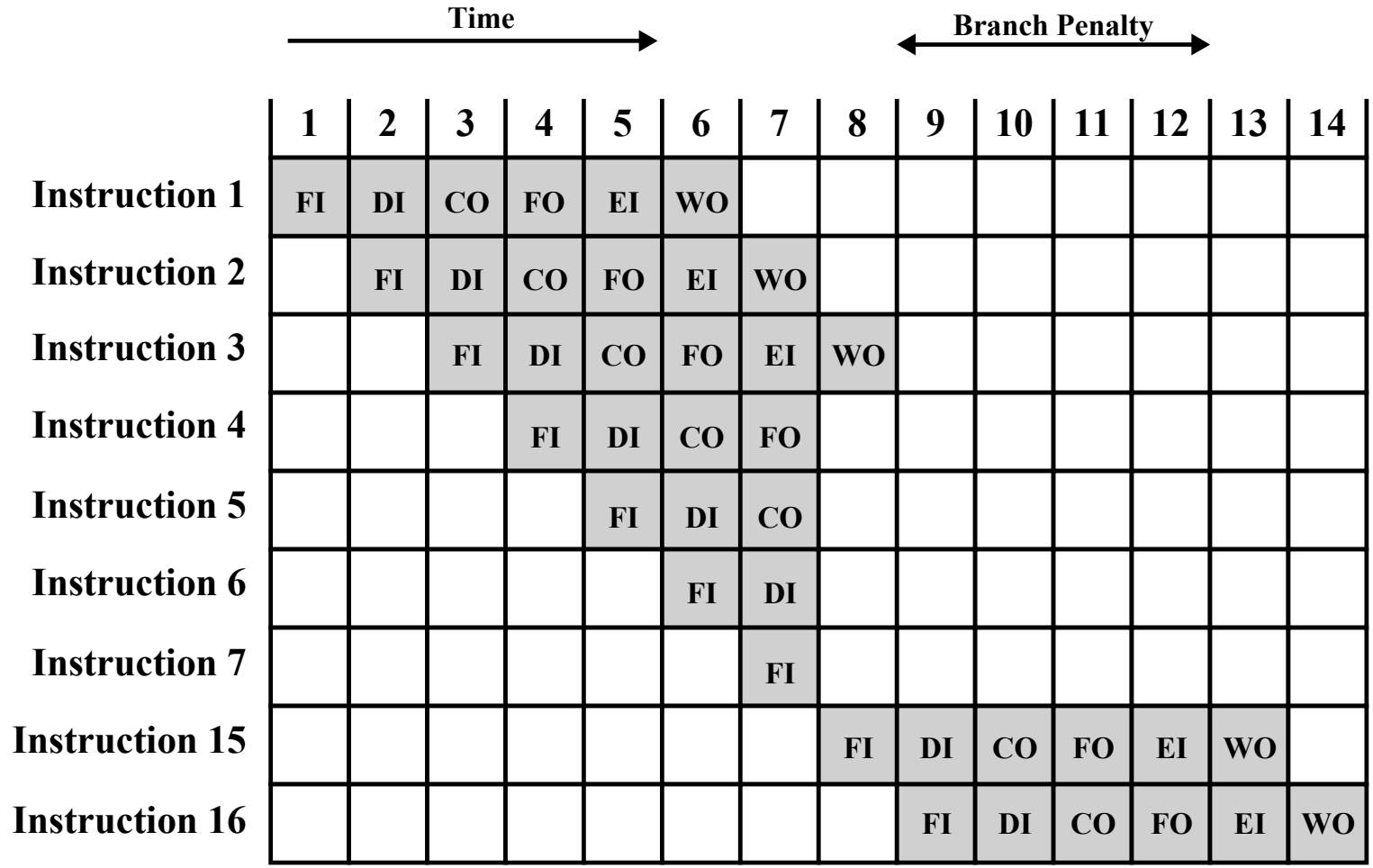
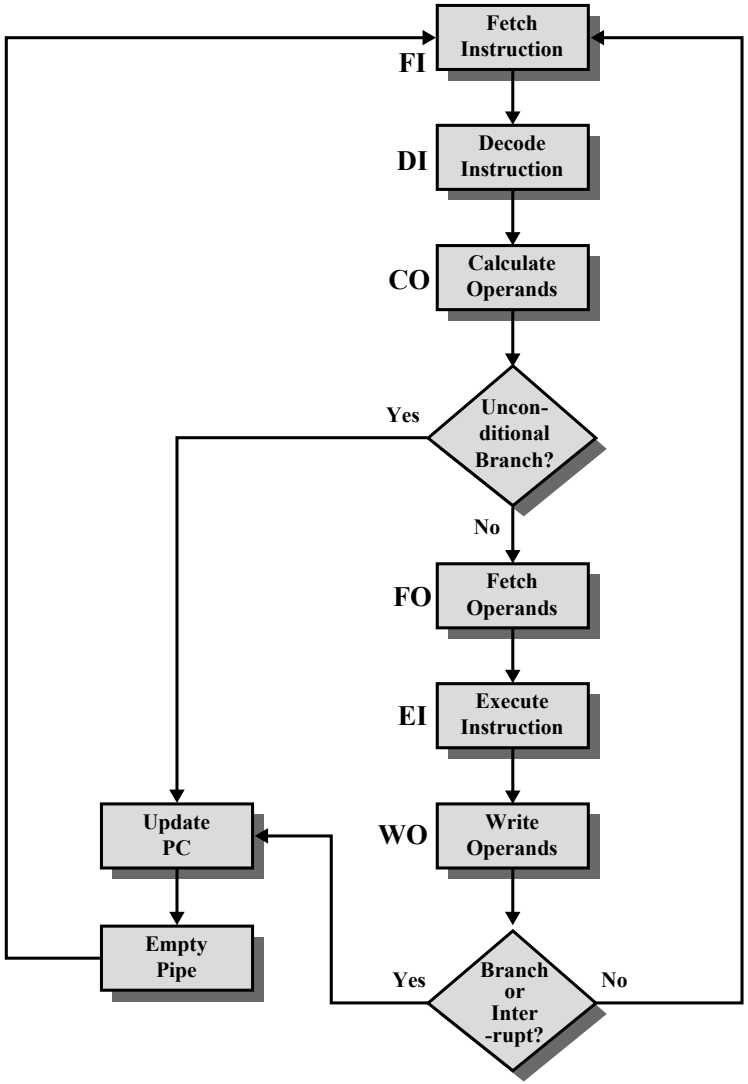


Figure 16.12
Six-Stage CPU Instruction Pipeline



More Stages in the Pipeline?

- It might appear that the greater the number of stages, the faster the execution rate, BUT consider:
  1. The overhead involved at each stage of the pipeline in moving data from buffer to buffer
  2. The amount of control logic required to handle memory and register dependencies increases enormously with the number of stages. This can lead to complex logic control than the stages being controlled.

Copyright