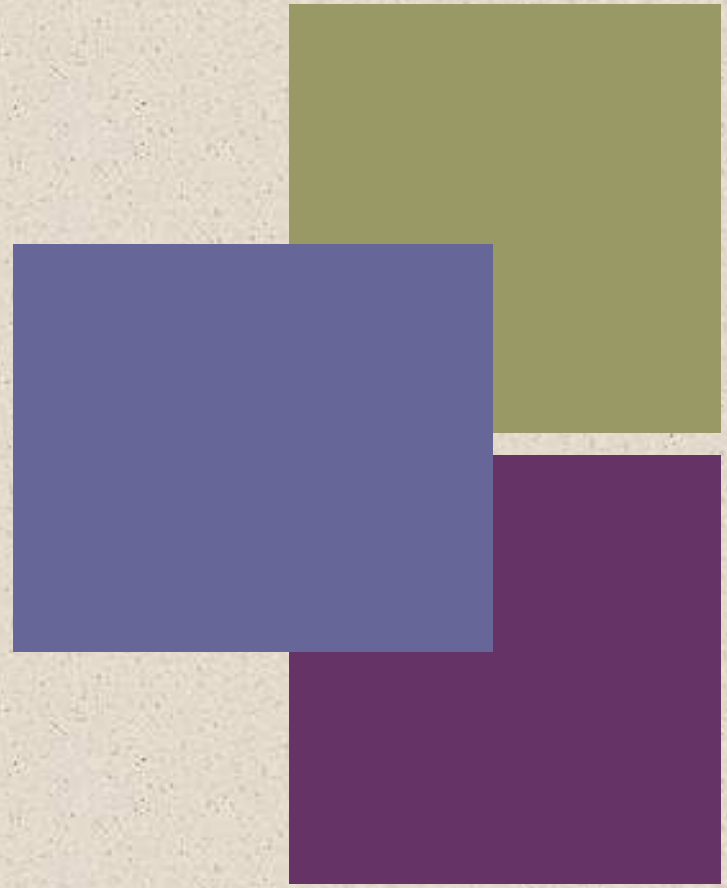
# William Stallings Computer Organization and Architecture 10th Edition



# Chapter 13

## Instruction-Level Parallelism and Superscalar Processors

# Superscalar Overview

Term first coined in 1987

Refers to a machine that is designed to improve the performance of the execution of scalar instructions

In most applications the bulk of the operations are on scalar quantities

Represents the next step in the evolution of high-performance general-purpose processors

Essence of the approach is the ability to execute instructions independently and concurrently in different pipelines

Concept can be further exploited by allowing instructions to be executed in an order different from the program order

# Table 16.1
# Reported Speedups of Superscalar-Like Machines

| Reference | Speedup |
|---|---|
| [TJAD70] | 1.8 |
| [KUCK77] | 8 |
| [WEIS84] | 1.58 |
| [ACOS86] | 2.7 |
| [SOHI90] | 1.8 |
| [SMIT89] | 2.3 |
| [JOUP89b] | 2.2 |
| [LEE91] | 7 |

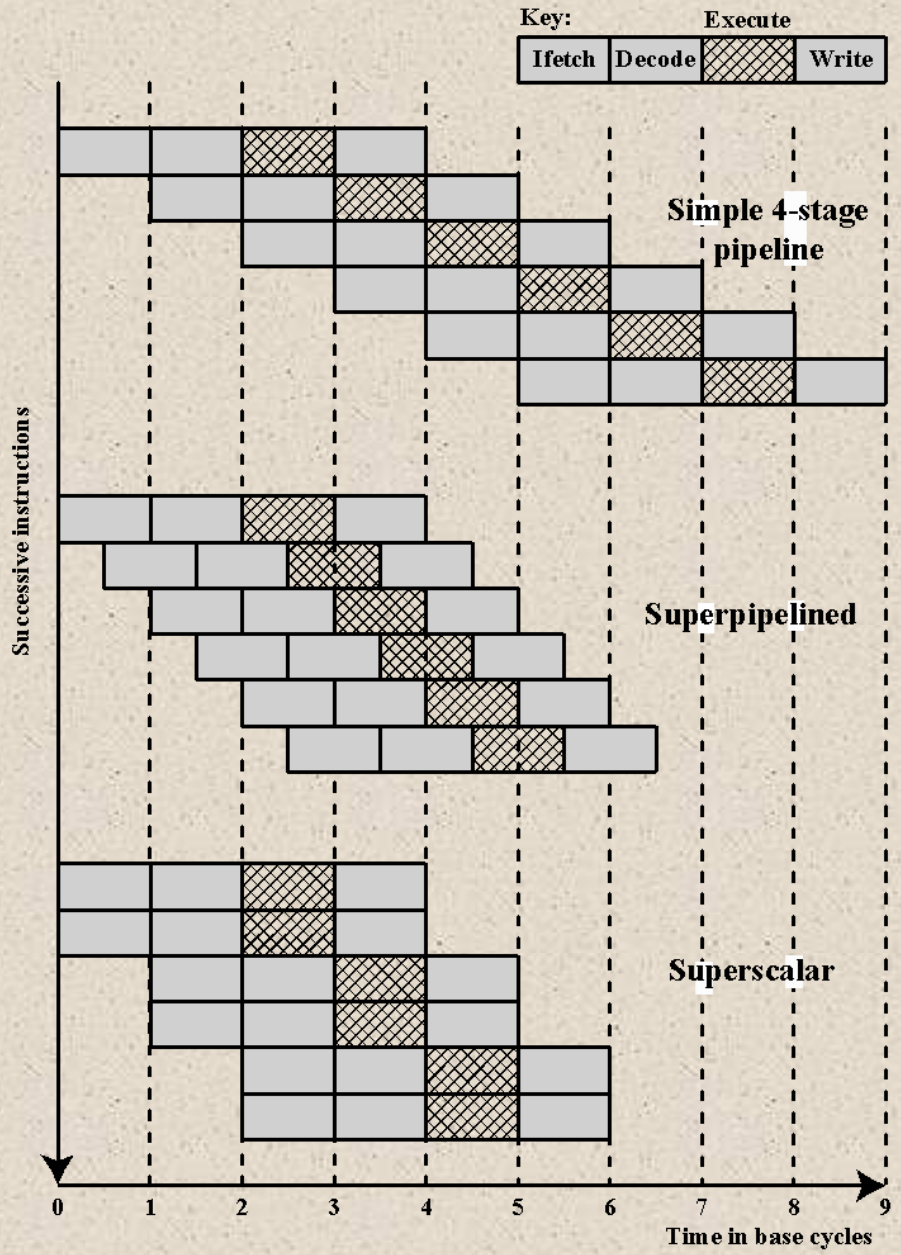



**Figure 16.2 Comparison of Superscalar and Superpipeline Approaches**

# Constraints

- Instruction level parallelism
  - Refers to the degree to which the instructions of a program can be executed in parallel
  - A combination of compiler based optimization and hardware techniques can be used to maximize instruction level parallelism
- Limitations:
  - True data dependency
  - Procedural dependency
  - Resource conflicts
  - Output dependency
  - Antidependency

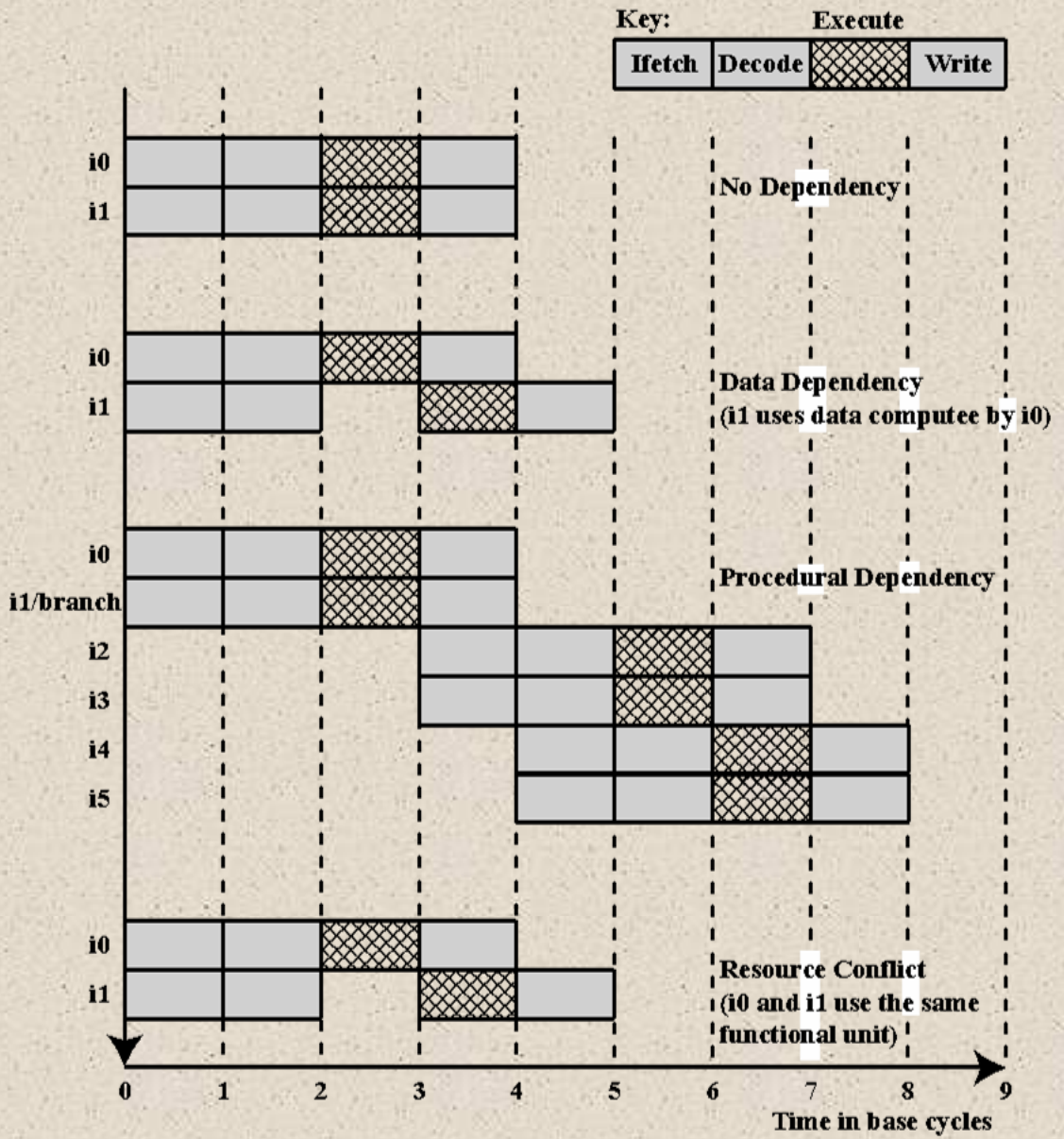



Figure 16.3 Effect of Dependencies

# Design Issues

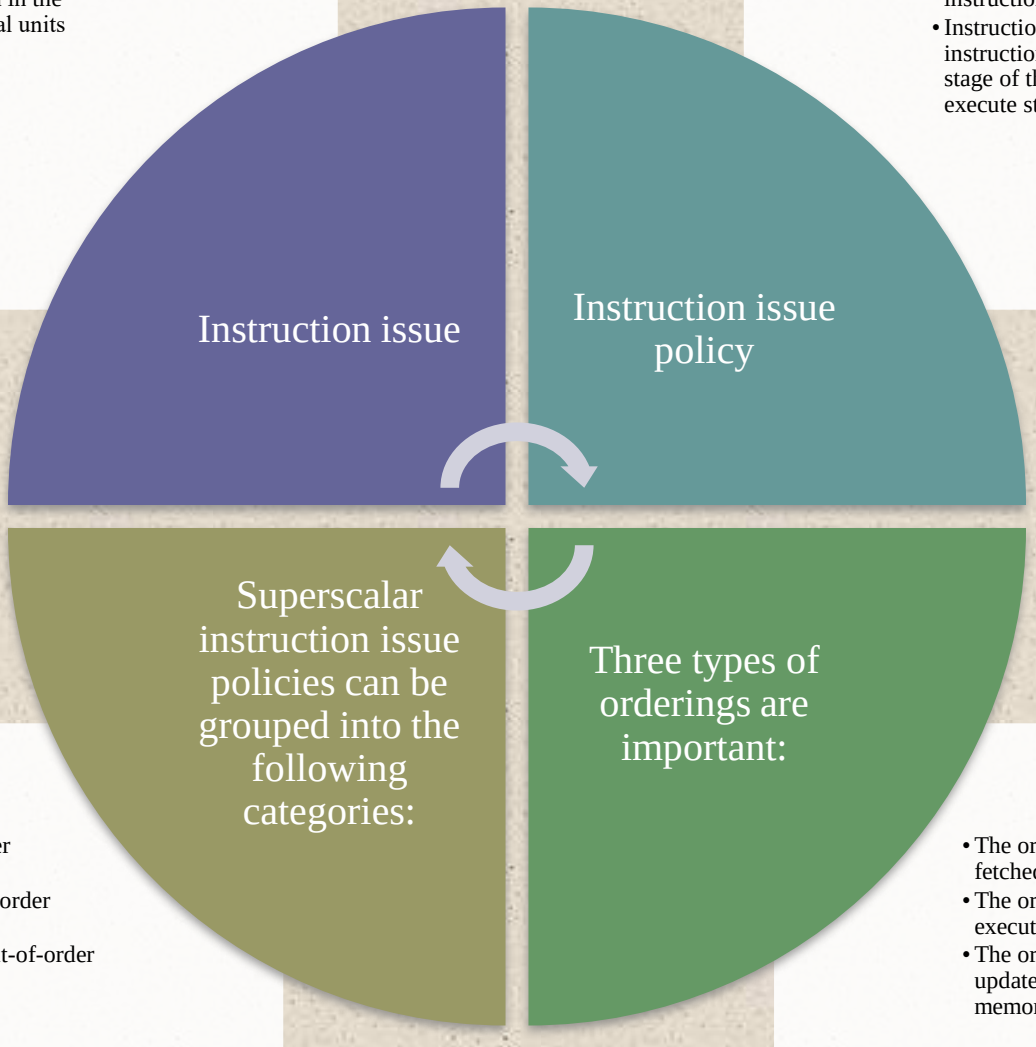
## Instruction-Level Parallelism and Machine Parallelism

- Instruction level parallelism
  - Instructions in a sequence are independent
  - Execution can be overlapped
  - Governed by data and procedural dependency
- Machine Parallelism
  - Ability to take advantage of instruction level parallelism
  - Governed by number of parallel pipelines

# Instruction Issue Policy

- Refers to the process of initiating instruction execution in the processor's functional units

- Refers to the protocol used to issue instructions
- Instruction issue occurs when instruction moves from the decode stage of the pipeline to the first execute stage of the pipeline




- In-order issue with in-order completion
- In-order issue with out-of-order completion
- Out-of-order issue with out-of-order completion

- The order in which instructions are fetched
- The order in which instructions are executed
- The order in which instructions update the contents of register and memory locations

# Register Renaming

Output and antidependencies occur because register contents may not reflect the correct ordering from the program

May result in a pipeline stall

Registers allocated dynamically

# Branch Prediction

- Any high-performance pipelined machine must address the issue of dealing with branches
- Intel 80486 addressed the problem by fetching both the next sequential instruction after a branch and speculatively fetching the branch target instruction
- RISC machines:
  - Delayed branch strategy was explored
  - Processor always executes the single instruction that immediately follows the branch
  - Keeps the pipeline full while the processor fetches a new instruction stream
- Superscalar machines:
  - Delayed branch strategy has less appeal
  - Have returned to pre-RISC techniques of branch prediction

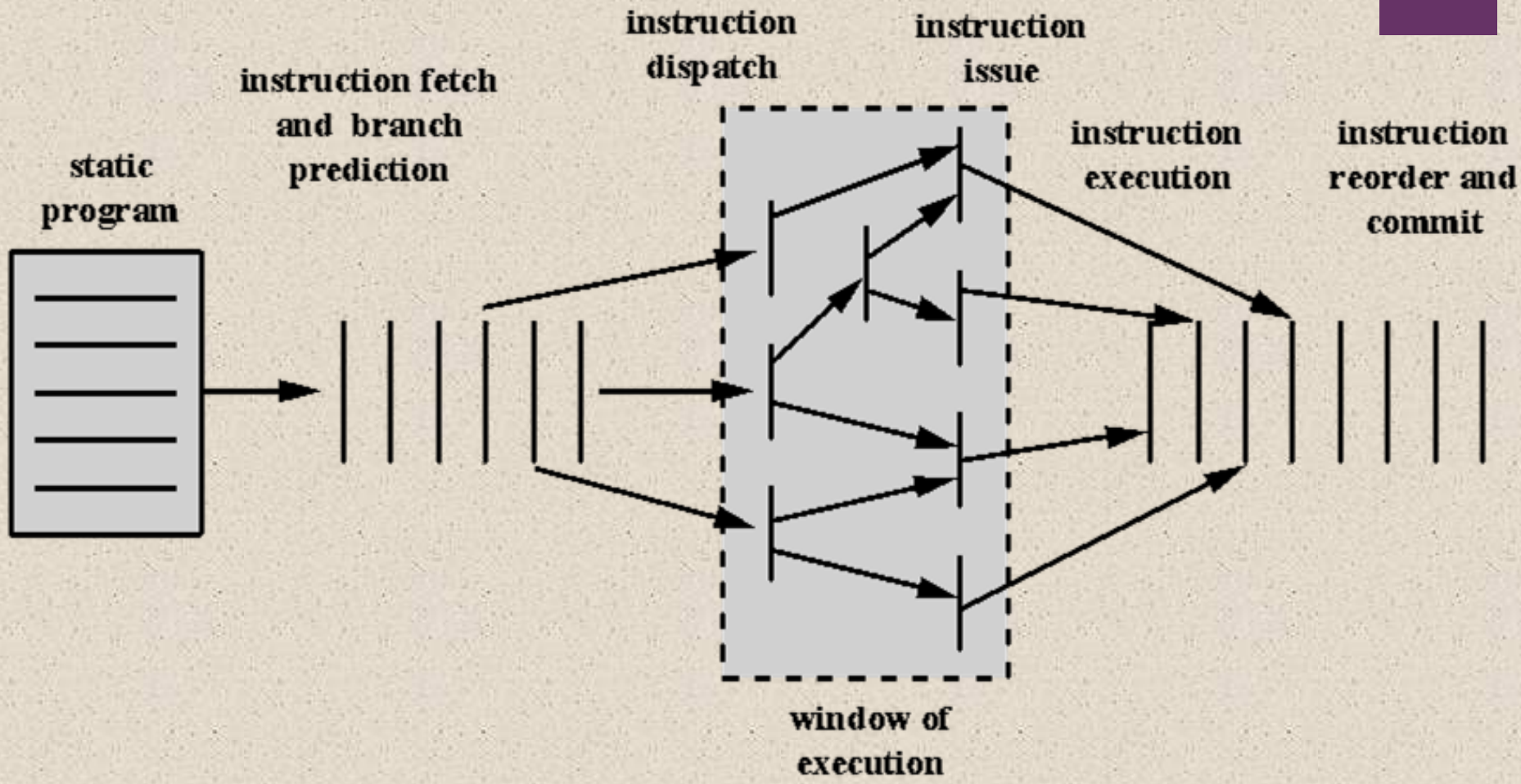



Figure 16.7 Conceptual Depiction of Superscalar Processing

# Superscalar Implementation

Key elements:

- Instruction fetch strategies that simultaneously fetch multiple instruction
- Logic for determining true dependencies involving register values, and mechanisms for communicating these values to where they are needed during execution
- Mechanisms for initiating, or issuing, multiple instructions in parallel
- Resources for parallel execution of multiple instructions, including multiple pipelined functional units and memory hierarchies capable of simultaneously servicing multiple memory references
- Mechanisms for committing the process state in correct order

# Branch Prediction Unit

- Helps the instruction fetch unit fetch the most likely instruction to be executed by predicting the various branch types:
  - Conditional
  - Indirect
  - Direct
  - Call
  - Return
- Uses dedicated hardware for each branch type
- Enables the processor to begin executing instructions long before the branch outcome is decided
- A branch target buffer (BTB) is maintained that caches information about recently encountered branch instructions

# Instruction Queue and Decode Unit

- Fetched instructions are placed in an instruction queue
  - From there the decode unit scans the bytes to determine instruction boundaries
  - The decoder translates each machine instruction into from one to four micro-ops
    - Each of which is a 118-bit RISC instruction
- A few instructions require more than four micro-ops so they are transferred to microcode ROM, which contains the series of micro-ops (five or more) associated with a complex machine instruction
- The resulting micro-op sequence is delivered to the rename/allocator module

# Out-of-Order Execution Logic

- This part of the processor reorders micro-ops to allow them to execute as quickly as their input operands are ready
- Allocate stage
  - Allocates resources required for execution
  - Performs the following functions:
    - If a needed resource is unavailable for one of the three micro-ops arriving at the allocator during a clock cycle, the allocator stalls the pipeline
    - Allocates a reorder buffer (ROB) entry which tracks the completion status of one of the 126 micro-ops that could be in process at any time
    - Allocates one of the 128 integer or floating-point register entries for the result data value of the micro-op, and possibly a load or store buffer used to track one of the 48 loads or 24 stores in the machine pipeline
    - Allocates an entry in one of the two micro-op queues in front of the instruction schedulers