MODULE 6: Instruction Set Architecture

Lecture 6.4
Instruction-Level Pipelining

Prepared By:

- Scott F. Midkiff, PhD
- Luiz A. DaSilva, PhD
- Kendall E. Giles, PhD

Electrical and Computer Engineering

Virginia Tech

Lecture 6.4 Objectives

- Describe the use of pipelining to increase the effective execution rate of a processor
- Show how a k-stage pipeline could increase effective execution rate by up to k times
- Show how a conditional branch can affect instruction-level pipelining
- Describe how instruction level parallelism can increase effective execution rate

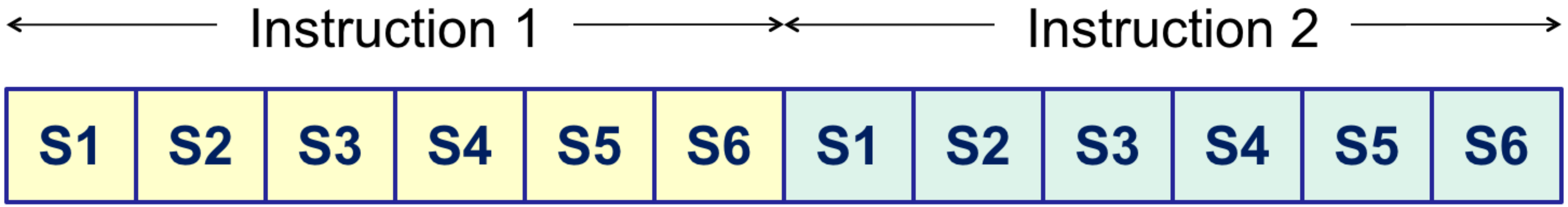
Basic Steps to Execute an Instruction

- A central processing unit (CPU) must perform steps to execute each instruction with a clock to sequence the steps
- Simple decomposition
  - Fetch cycle: fetch the instruction from memory (or an instruction cache)
  - Decode: determine the operation and addressing mode(s)
  - Execute: perform the instruction and update any registers and memory locations
- Other decompositions are possible

A Further Decomposition

- Another sequence to execute an instruction is:
  - Fetch the instruction: fetch the instruction from memory (or cache)
  - Decode: determine the operation from the opcode
  - Calculate operand addresses: determine addresses for operands based on addressing mode and values
  - Fetch the operands: fetch the (zero or more) operands from memory (or cache)
  - Execute the instruction: perform the specified operation
  - Store result: update registers and/or memory with results

Simple (Non-Pipelined) Execution

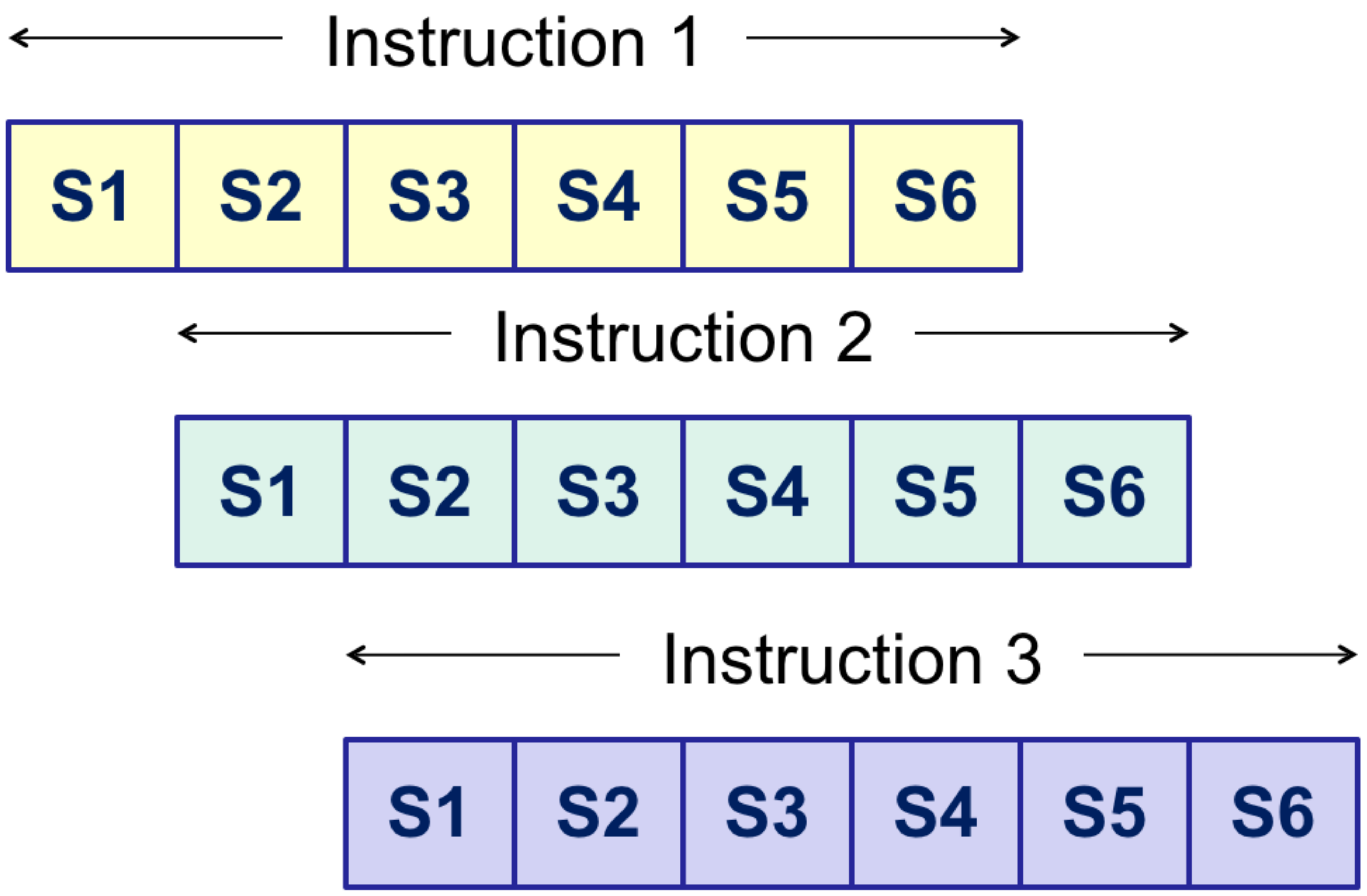


Six Execution Steps

S1: Fetch the instruction
S2: Decode opcode
S3: Calculate effective addresses
S4: Fetch operands
S5: Execute instruction
S6: Store result

Pipelined Instruction Execution

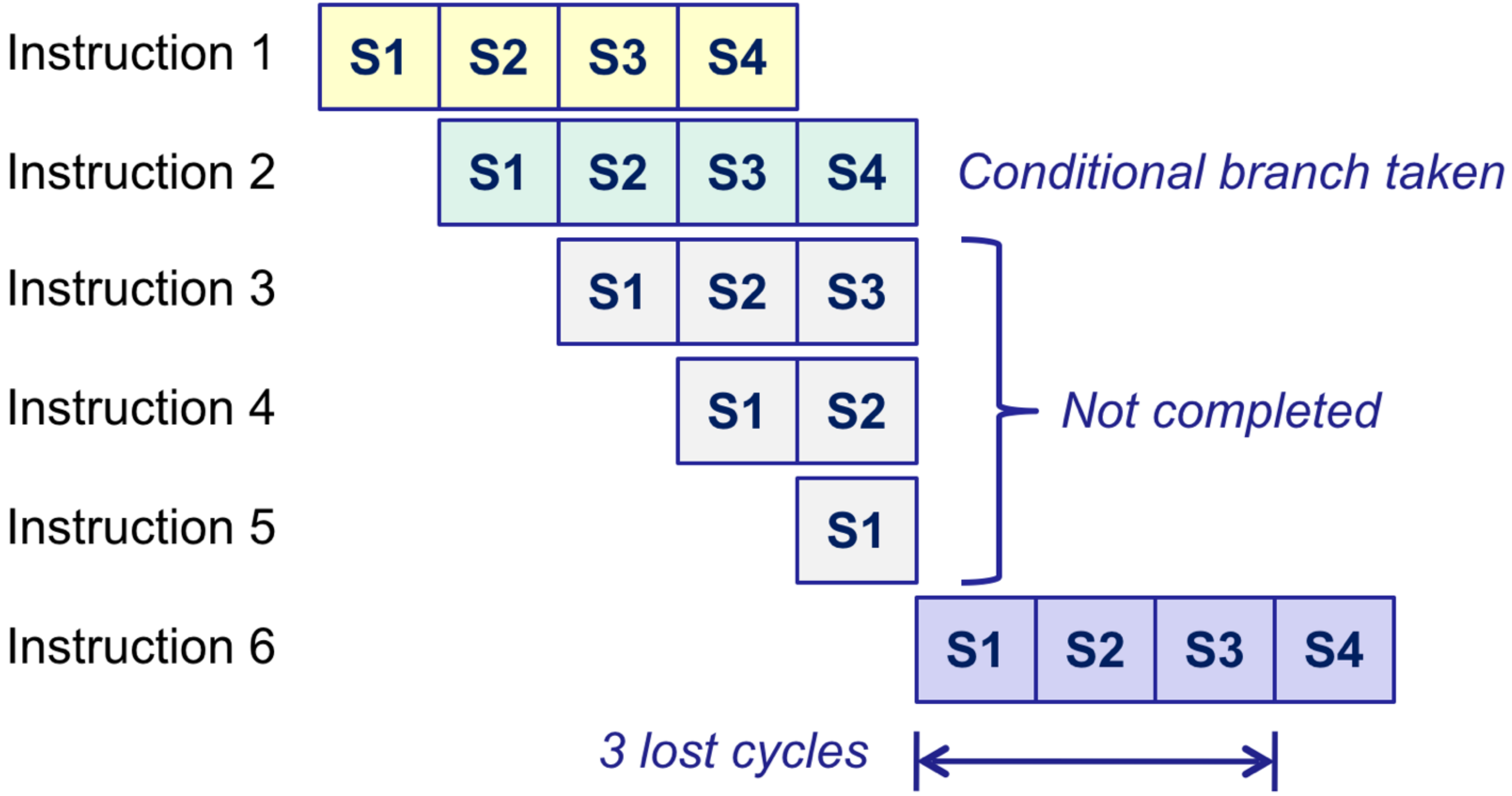
- The “production” of instructions can be increased by pipelining (with an analogy to an assembly line)



Instruction-Level Pipelining

- Benefits (Consider a k-stage pipeline, k=6 for our example)
  - An instruction can complete as often as every 1 time unit instead of every k time units
  - Increases effective instruction execution rate by up to k times
- Problems
  - Conditional branches, exceptions, and other actions that change program flow may “break” the pipeline, so k times speedup is the ideal maximum
  - Resource conflicts may halt the pipeline
  - Introduces additional hardware and design complexity

Conditional Break Example (k=4 stages)



CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can do the following:

- Describe the use of pipelining to increase the effective execution rate of a processor
- Describe how a k-stage pipeline could increase effective execution rate by up to k times
- Describe how a conditional branch can affect instruction-level pipelining

If you have any difficulties, please review the lecture video before continuing.

Improvements

- Compiler techniques to reduce impact of conditional branches and minimize resource conflicts
- Branch prediction in the CPU to “guess” the instruction flow and, thus, probabilistically reduce occurrence of pipeline breaks
- Parallelism
    - Explicitly parallel instruction computers (EPIC), with parallelism determined by user or, more likely, the compiler
    - Implicit parallelism determined at run time by the processor

Parallelism

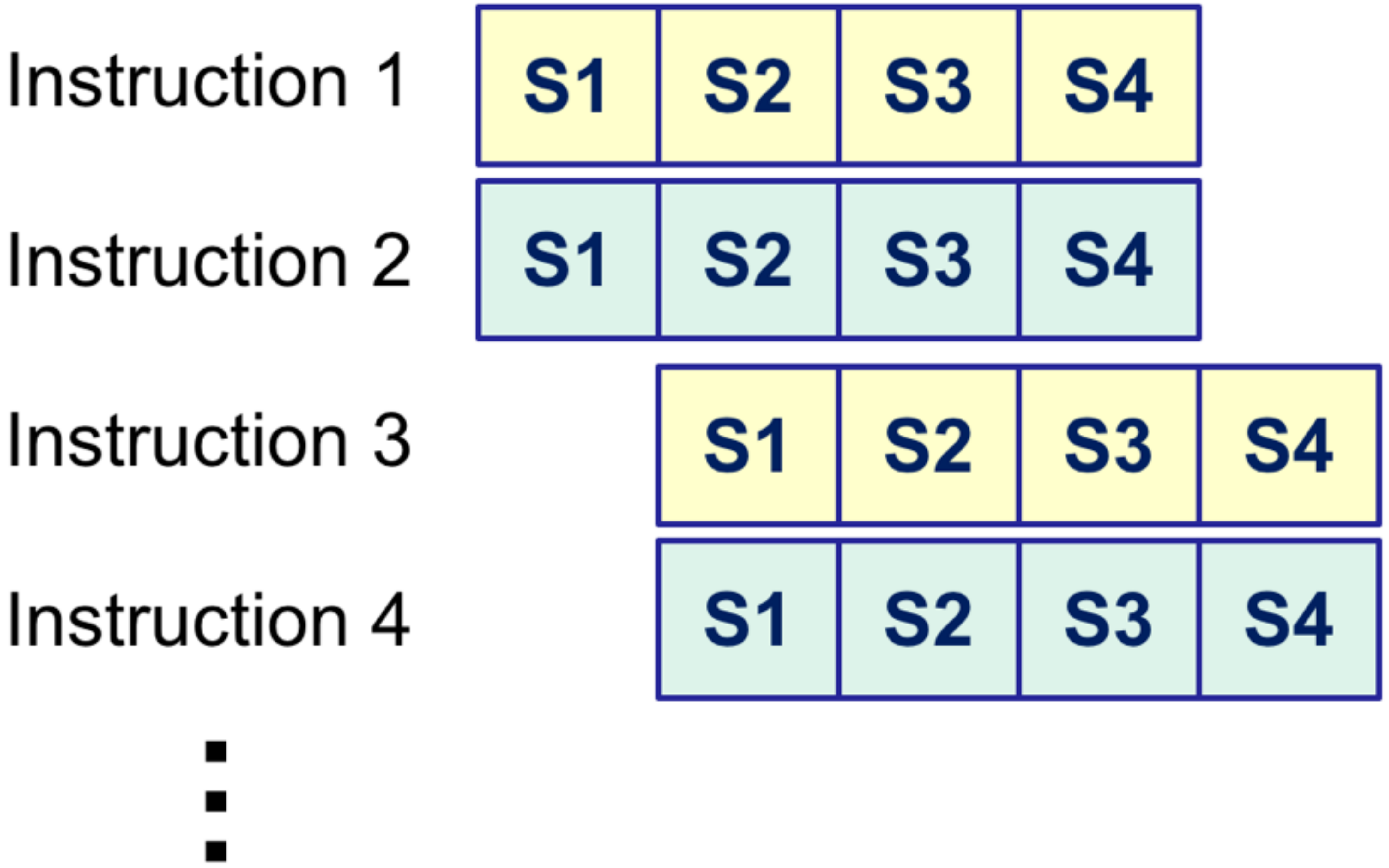
- Parallelism can further increase processor performance
  - Instruction-level parallelism (ILP) – multiple instructions are executed at the same time (the same stage is active for two or more instructions)
  - Program-level parallelism (PLP) – two or more procedures are executed in parallel
- ILP is usually left to the compiler, PLP usually requires some level of programmer specification
- ILP and PLP can be used separately or together

Some Terminology

- Superscalar architectures – provide parallel execution of different instructions, so multiple instructions can be executed at the same time
  - Processor can improve ordering to increase utilization of multiple pipelines, but compiler can help
- Very long instruction word (VLIW) architectures – each instruction can specify multiple operations and operands and these different operations are performed at the same time
  - Burden is on the compiler to maximize use of resources
- Superpipelining architectures – divide instruction execution into a large number of stages

Superscalar Example

- Example: 2-way parallelism with 4-stage pipelining
- Conditional branches, resource contention, exceptions, etc. can limit parallism and/or pipelining achieved
- Need a good compiler



CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can do the following:

- Describe how instruction-level parallelism can increase the effective execution rate

If you have any difficulties, please review the lecture video before continuing.

Summary

- Pipelining divides instruction execution into multiple steps or stages and performs different stages for different instructions at the same time – assembly line analogy
- A k-stage pipeline could increase effective execution rate k times, but this is rarely sustained due to resource contention, conditional branches, exceptions, and other factors
- Instruction level parallelism can increase effective execution rate

MODULE 6: Instruction Set Architecture

Lecture 6.4
Instruction-Level Pipelining

Prepared By:
- Scott F. Midkiff, PhD
- Luiz A. DaSilva, PhD
- Kendall E. Giles, PhD

Electrical and Computer Engineering
Virginia Tech