Superscalar Processors

ENCS5331: Advanced Computer Architecture

Fall 2023/2024

Instructor: Dr. Ayman Hroub

Special Thanks to Dr. Muhamed Mudawar (KFUPM) and Dr. Onur Mutlu (ETHZ) for most of the Slides

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

Making the Pipeline Superscalar

- Pipelines studied so far are scalar
  - Fetch, decode, and dispatch one instruction per cycle
  - Write-back and Commit one instruction per cycle
- Fundamentally limited to CPI ≥ 1
- Superscalar pipelines can do more …
  - Can fetch, decode, and dispatch multiple instructions per cycle
  - Can execute, write-back, and commit multiple instructions per cycle
  - Can reduce the CPI below 1 (CPI < 1)
  - **IPC = Instructions per Cycle = 1 / CPI**
- Two types of superscalar processors
  - In-order execution: based on program order
  - Out-of-order execution: based on data dependences

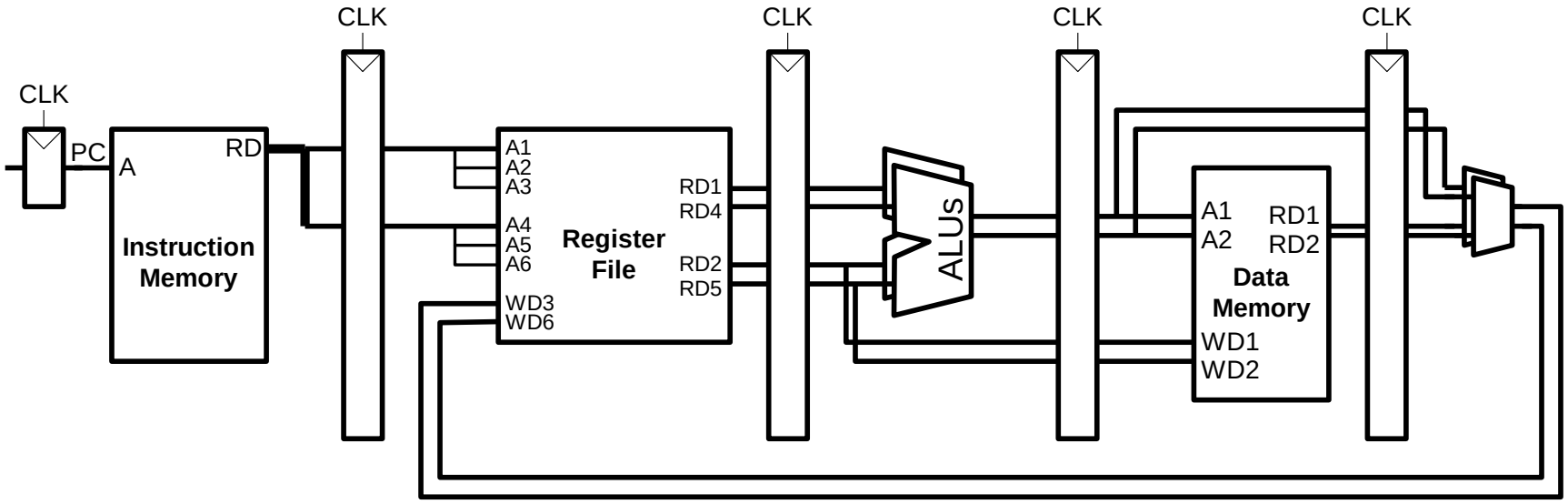
Superscalar Execution

- Idea: **hardware** Fetches, decodes, executes, retires multiple instructions per cycle
  - N-wide superscalar → N instructions per cycle
- Need to add the hardware resources for doing so
- Hardware performs the dependence checking between concurrently-fetched instructions
- Superscalar execution and out-of-order execution are orthogonal concepts
  - Can have all four combinations of processors:

    [in-order, out-of-order] x [scalar, superscalar]

In-Order Superscalar Processor Example

- Multiple copies of datapath: Can fetch/decode/execute multiple instructions per cycle
- Dependencies make it tricky to issue multiple instructions at once

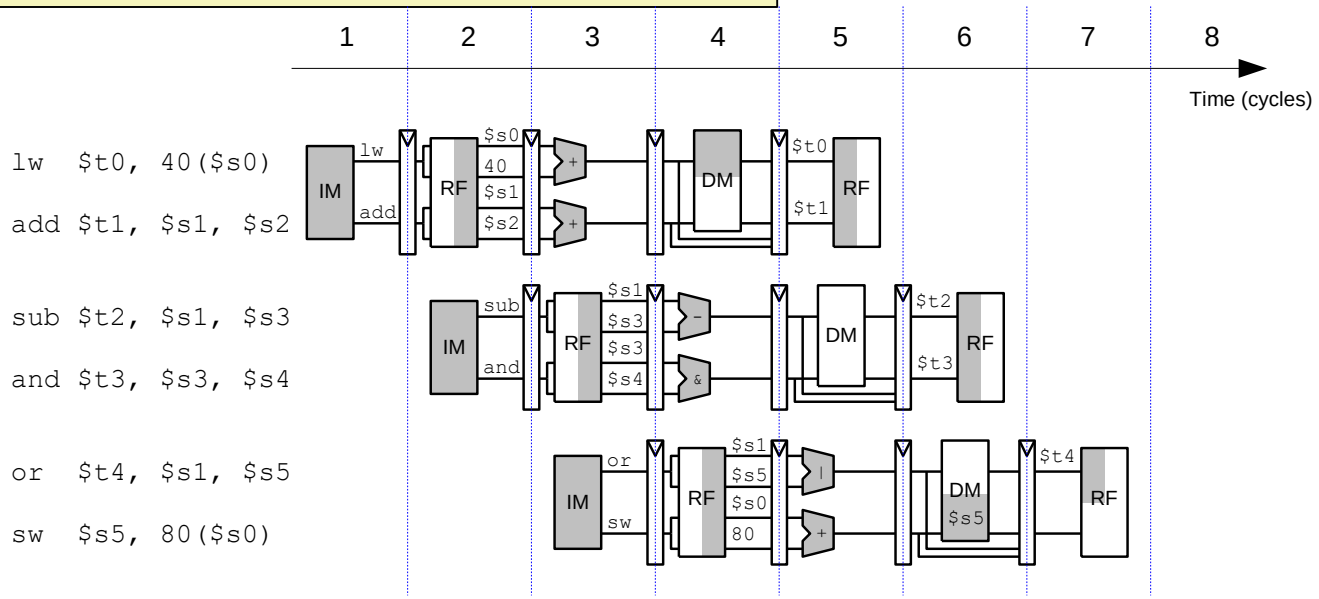


Here: Ideal IPC = 2

In-Order Superscalar Performance Example

```
lw  $t0, 40($s0)
add $t1, $s1, $s2
sub $t2, $s1, $s3
and $t3, $s3, $s4
or  $t4, $s1, $s5
sw  $s5, 80($s0)
```

Ideal IPC = 2

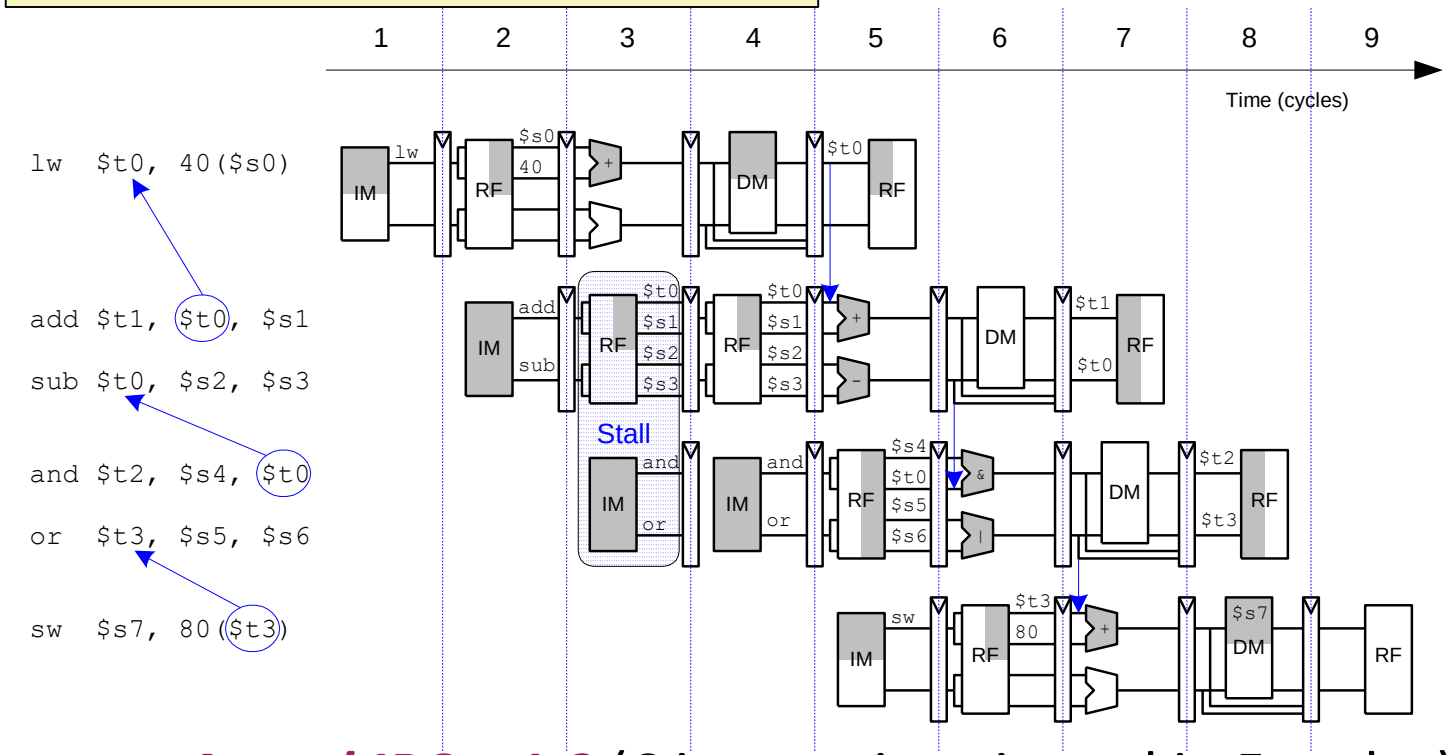


Actual IPC = 2 (6 instructions issued in 3 cycles)

Superscalar Performance with Dependencies

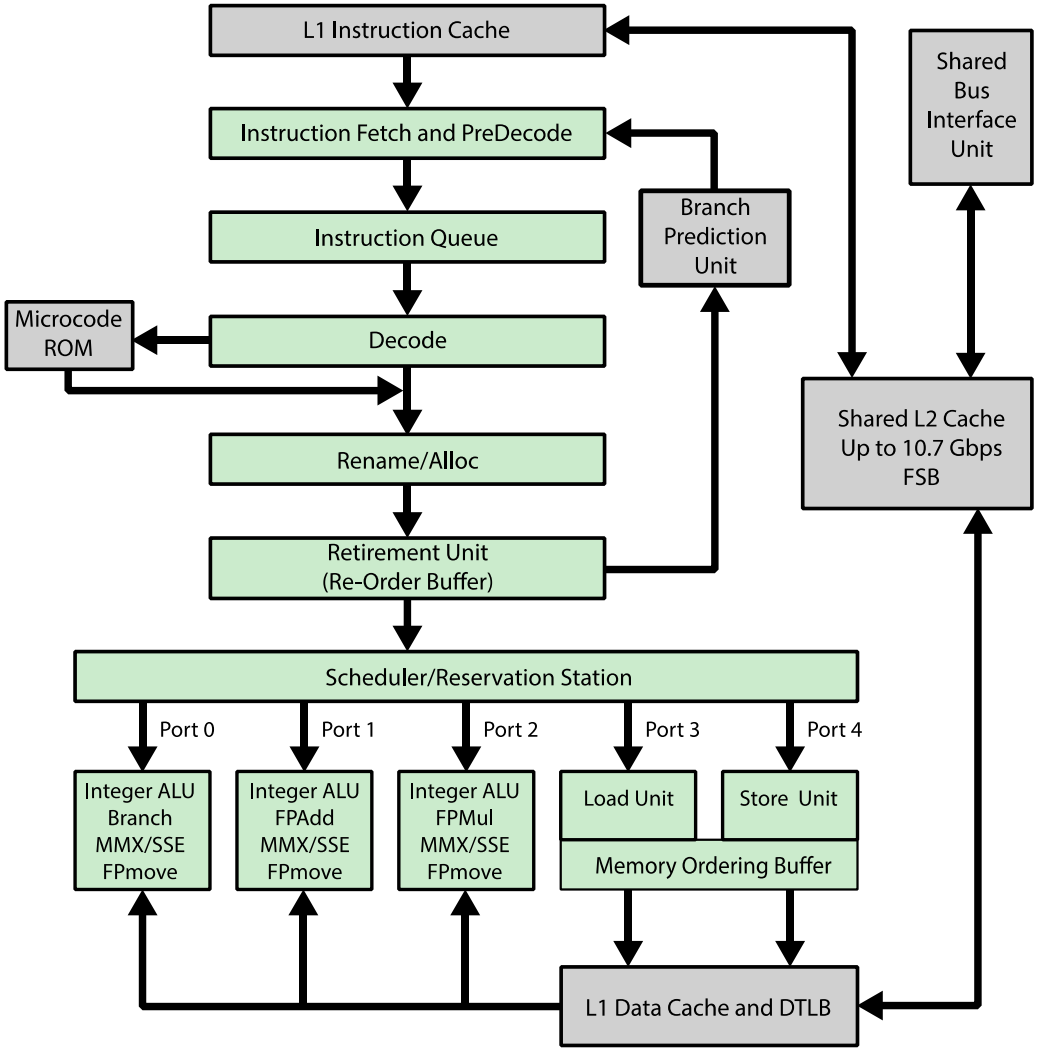
```
lw  $t0, 40($s0)
add $t1, $t0, $s1
sub $t0, $s2, $s3
and $t2, $s4, $t0
or  $t3, $s5, $s6
sw  $s7, 80($t3)
```

Ideal IPC = 2



Actual IPC = 1.2 (6 instructions issued in 5 cycles)

Intel Core Microarchitecture



Superscalar Execution Tradeoffs

- Advantages
  - Higher IPC (instructions per cycle)
- Disadvantages
  - Higher complexity for dependency checking
    - Require checking within a pipeline stage
    - Renaming becomes more complex in an OoO processor
  - More hardware resources needed

Superscalar Complexities

- Must fetch a group of instructions each cycle (not just one)
  - An accurate branch predictor is essential for correct fetching
- Must decode a group of instructions each cycle
  - Must check dependences between instructions in a single group
- Must dispatch a group of instructions each cycle
  - Requires more read ports for the register file and reorder buffer
  - Read ports increases with the number of dispatched instructions
- Must write multiple results each cycle
  - Requires multiple result busses (not just one common data bus)
  - Requires multiple write ports for the ROB (not just one)
  - More comparators for tag matching in reservation stations
- Must commit multiple results each cycle
  - Requires multiple write ports for the register file

Example: Intel Core i7 Pipeline Structure

Pipeline depth = 14 stages

Branch penalty = 15 cycles

Intel x86 instructions translated into micro-operations

Micro-ops: RISC-like instructions

4 instructions decoded per cycle

Loop stream detector and buffer

36-entry centralized RS

Six independent function units

Up to 6 micro-ops can be executed per cycle