CS5375 Computer Systems Organization and Architecture

Lecture 14

Instructor: Yong Chen, Ph.D.
Department of Computer Science
Texas Tech University
Yong.Chen@ttu.edu, 806-834-0284

Announcements

- Final Reminder: Midterm exam on 10/20, Thursday, 8 a.m. - 9:20 a.m.
  - Must take the exam during the class time and in person in the classroom
  - Test subjects we discussed regarding Chapters 1 and 2, i.e., all lectures till Lecture 9
    - Only subjects we discussed in class, you should review lecture slides, HW, related textbook discussion
    - No questions regarding the research paper we studied "Amdahl's Law in the Multicore Era" and research papers we studied in homework
    - Final exam will test subjects we cover from Lecture 10
  - All multi-choice questions, closed-book, closed-note

  - Will enforce randomized seating and will email your seat number ~half-hour before the exam
  - Then locate your seat from the below link and display your university ID at the top-left corner

https://www.depts.ttu.edu/tlpdc/Classroom_Layouts_and_Resources/Petroleum_110.pdf

Review of Last Lecture

- Exploiting ILP Using Multiple Issue and Static Scheduling
  - Multiple issue (N-way multi-issue CPU)
  - Multiple pipelines, start multiple instructions per clock cycle
  - E.g., Dual Issue/Two-way/Two-issue CPU
  - Static scheduling example and loop unrolling
- Dynamic Scheduling
  - Rearrange order of instructions by CPU (hardware) to reduce stalls while maintaining data flow
    - i.e., Out-of-order (OOO) execution
  - Tomasulo's Approach
    - A method of implementing dynamic scheduling, invented by Robert Tomasulo
    - Tracks when operands are available, satisfying the data dependence
    - Introduces register renaming in hardware to remove name dependence
  - Three steps: issue, execute/dispatch, write result

Tomasulo's Algorithm

- Three Steps:
  - Issue
    - Get next instruction from FIFO queue
    - If available RS, issue the instruction to the RS with operand values if available
    - If operand values not available, stall the instruction
  - Execute/dispatch
    - When operand becomes available, store it in any reservation stations waiting for it
    - When all operands are ready, excute the instruction
    - Loads and store maintained in program order through effective address
  - Write result
    - Write result on CDB into reservation stations and store buffers

Tomasulo's Algorithm: An Example

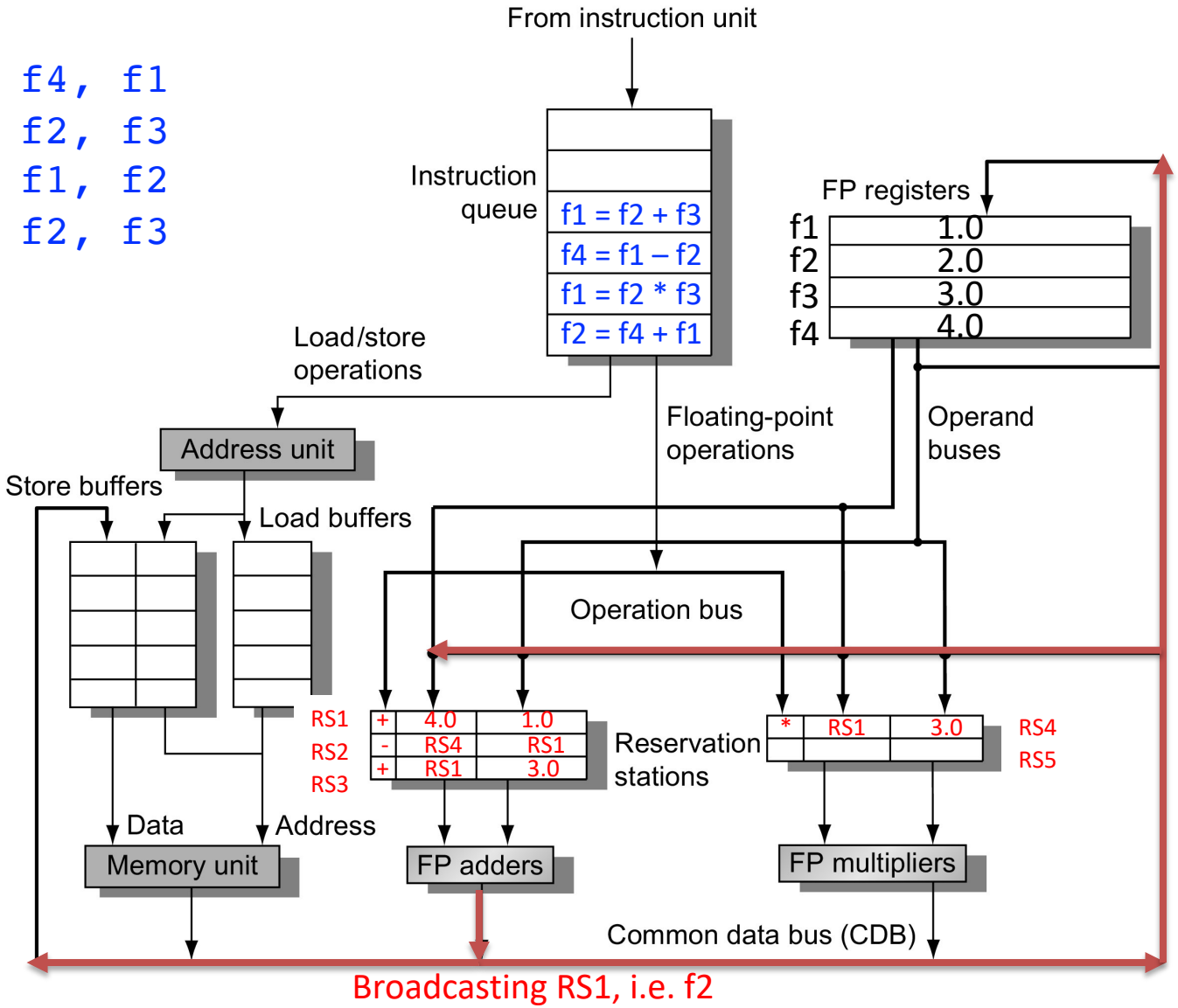
Program order

```
I1: fadd.d f2, f4, f1
I2: fmul.d f1, f2, f3
I3: fsub.d f4, f1, f2
I4: fadd.d f1, f2, f3
```

Register alias table

f1	~~RS4~~RS3
f2	RS1
f3	
f4	RS2

If there's no RS3, we cannot issue `I4`.



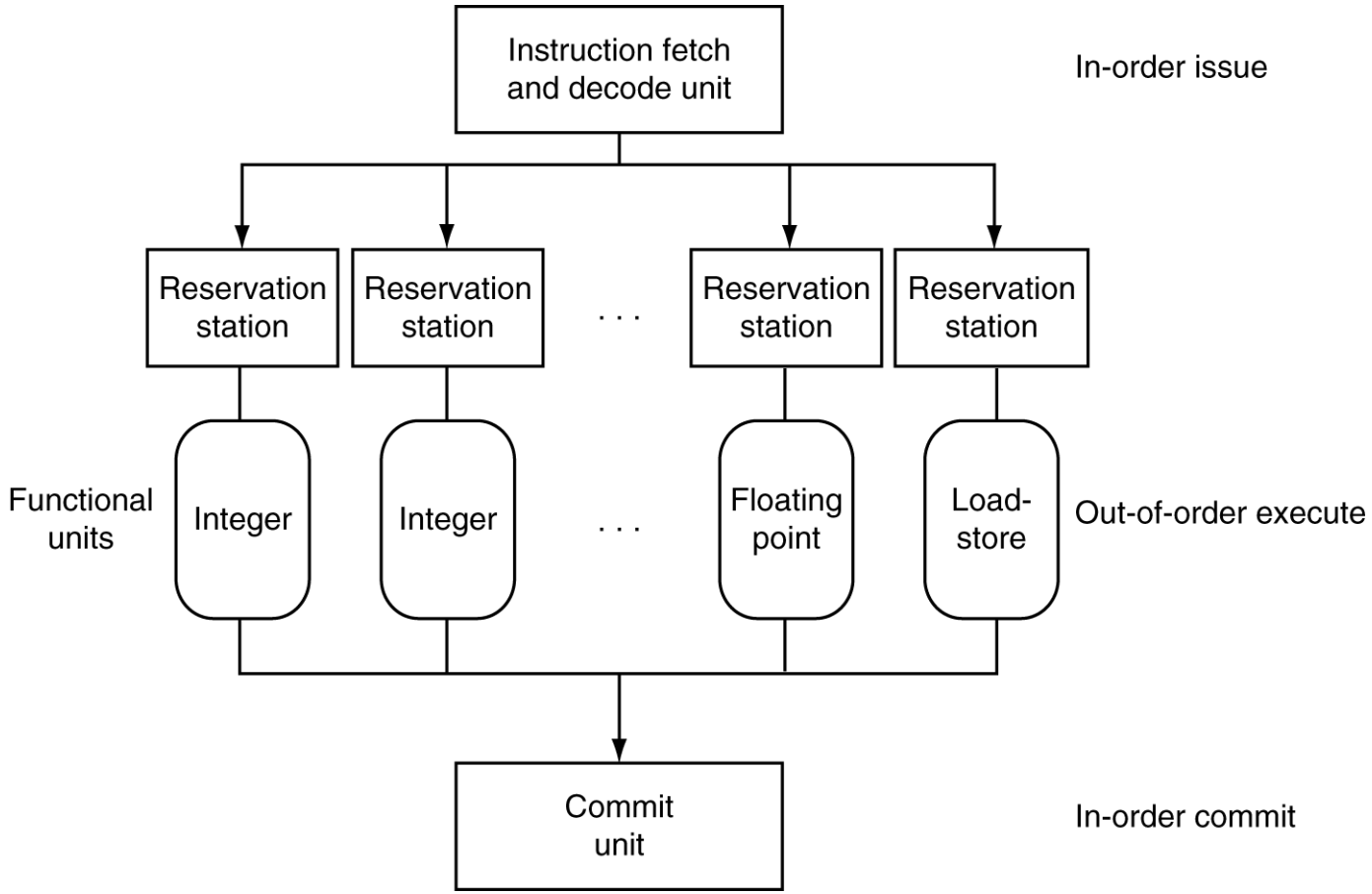
Such a process continues as issuing "in order" but executing "out of order".

CPUs nowadays use similar mechanism to issue all instructions with 100s instructions in IQ.

Dynamic Multiple Issue

- Also known as “superscalar” processors, or “dynamic pipeline scheduling”
- CPU decides whether to issue 0, 1, 2, or more instructions each cycle
  - Avoiding structural and data hazards
- Avoids the need for compiler scheduling

Dynamic Multiple Issue CPU



Dynamic Scheduling, Multiple Issue, and Speculation

- Modern microarchitectures:
  - Dynamic scheduling + multiple issue + speculation
- How much to speculate
  - Mis-speculation degrades performance and power relative to no speculation
    - May cause additional misses (cache, TLB)
  - Prevent speculative code from causing higher costing misses (e.g. L2)
- Speculating through multiple branches
  - Complicates speculation recovery
- Speculation and energy efficiency
  - Note: speculation is only energy efficient when it significantly improves performance

Outline

- Dynamic Scheduling
- HW1 and HW2 Review

Readings

- Chapter 3, 3.4 – 3.8