CS 223 Computer Organization & Architecture

Lecture 30 [23.04.2020]

Dynamic Scheduling to Explore ILP

John Jose

Assistant Professor

Department of Computer Science & Engineering

Indian Institute of Technology Guwahati, Assam.

Dynamic Scheduling

- **Rearrange execution order of instructions to reduce stalls while maintaining data flow**
- Advantages:
  - Compiler doesn't need to have knowledge of micro-architecture
  - Handles cases where dependencies are unknown at compile time
- Disadvantage:
  - Substantial increase in hardware complexity
  - Complicates exceptions

How dynamic scheduling works ?

- Limitation of simple pipelining.
  - In-order instruction issue and execution.
  - Instructions are issued in program order.
  - If an instruction is stalled in the pipeline, no later instructions can proceed.

```
add r1,r2,r3
sub r4,r1,r3
and r6,r1,r7
or r8,r1,r9
```

How dynamic scheduling works ?

- Limitation of simple pipelining.
  - In-order instruction issue and execution.
  - Instructions are issued in program order.
  - If an instruction is stalled in the pipeline, no later instructions can proceed.
- If instruction j depends on a long-running instruction i, currently in execution in the pipeline, then all instructions after j must be stalled until i is finished and j can execute.

```
DIV.D     F0,F2,F4
ADD.D     F10,F0,F8
SUB.D     F12,F8,F14
```

How dynamic scheduling works ?

- Separate the issue process into two parts:
  - checking for any structural hazards.
  - waiting for the absence of a data hazard.

```
DIV.D F0,F2,F4
ADD.D F10,F0,F8
SUB.D F12,F8,F14
```

- Use in-order instruction issue but we want an instruction to begin execution as soon as its data operands are available.
- out-of-order execution → out-of-order completion.
- OOO execution introduces the possibility of WAR and WAW hazards

How dynamic scheduling works ?

```
DIV.D     F0,F2,F4
ADD.D     F6,F0,F8
SUB.D     F8,F10,F14
MUL.D     F6,F10,F8
```

- WAR and WAW hazards – solved by register renaming
- Possibility of imprecise exception (2 possibilities).
  - The pipeline may have already completed instructions that are later in program order than the instruction causing the exception.
  - The pipeline may have not yet completed some instructions that are earlier in program order than the instruction causing the exception

How dynamic scheduling works ?

- To allow out-of-order execution, **split the ID stage into two**
  - **Issue**—Decode instructions, check for structural hazards.
  - **Read operands**—Wait until no data hazards, then read operands.
- In a dynamically scheduled pipeline, all instructions pass through the **issue stage in order** (in-order issue); however, they can be stalled or bypass each other in the second stage (read operands) and thus enter execution out of order.
- Done by - **score boarding technique**
- Approach used - **Tomasulo's algorithm**

Register Renaming

DIV.D F0,F2,F4

ADD.D F6,F0,F8

S.D F6,0(R1)

SUB.D F8,F10,F14

MUL.D F6,F10,F8

name dependence with F6

DIV.D F0,F2,F4

ADD.D S,F0,F8

S.D S,0(R1)

SUB.D T,F10,F14

MUL.D F6,F10,T

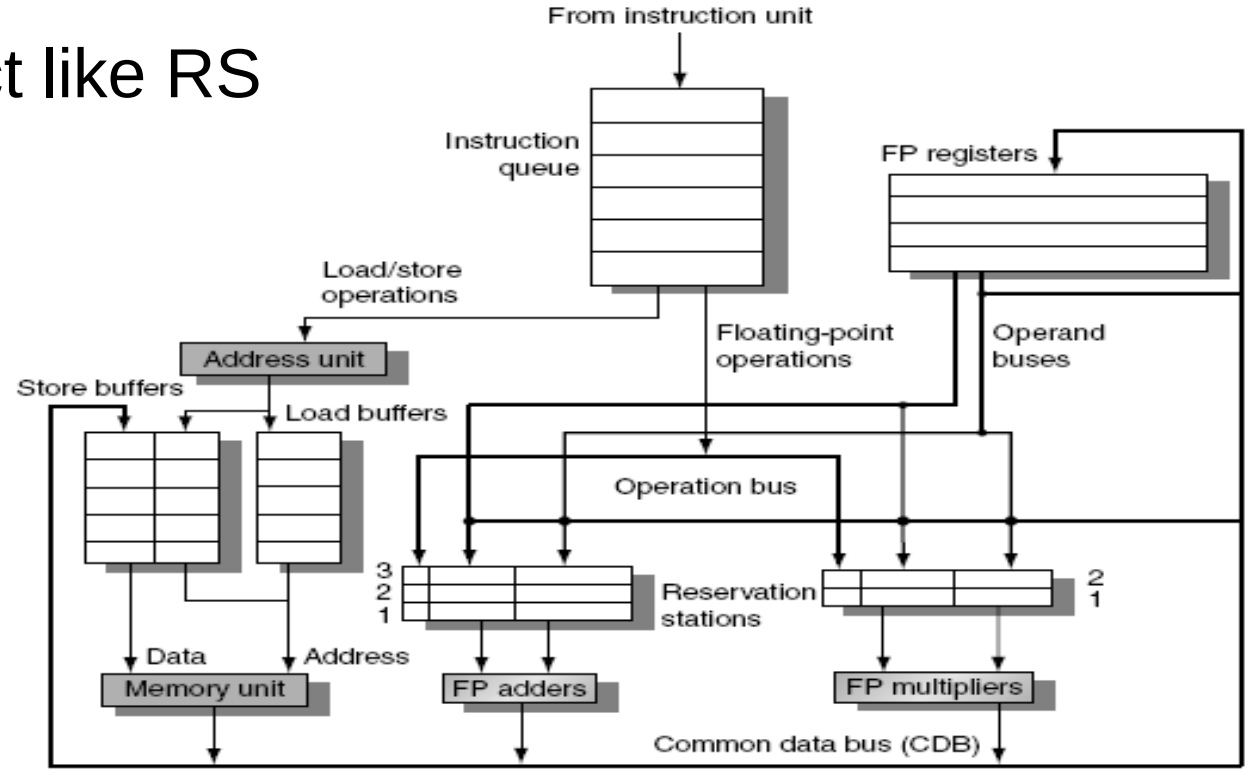
RAW hazard on T

Register Renaming

- Register renaming is done by reservation stations (RS)
- Each RS Contains:
  - The instruction (operation to be done)
  - Buffered operand values (when available)
  - Reservation station number of instruction providing the operand values
- RS fetches and buffers an operand as soon as it becomes available (not necessarily involving register file)
- Pending instructions designate the RS that will provide input
- Result values broadcast on common data bus (CDB)

Tomasulo's Algorithm

- Load and store buffers contain data and addresses.
- They also act like RS



Dynamic Scheduling - Tomasulo

- **Issue**
  - Get next instruction from FIFO queue
  - If RS available, issue the instruction to the RS with operand values if available
  - If operand values not available, stall the instruction

Dynamic Scheduling - Tomasulo

- **Execute**
  - When operand becomes available, store it in any reservation stations waiting for it
  - When all operands are ready, execute the instruction
  - Loads and store uses buffers
  - No instruction will initiate execution until all branches that precede it in program order have completed

Dynamic Scheduling - Tomasulo

- **Write result**
  - Write result into CDB (there by it reaches the reservation station, store buffer and registers file) with name of execution unit that generated the result.
  - Stores must wait until address and value are received

johnjose@iitg.ac.in
http://www.iitg.ac.in/johnjose/