Credits: Dr. Yousif (Intel), Kubi@UCB

Advanced Instruction Re-Ordering: Dynamically Scheduled Pipelines: Tomasulo's Algorithm

Handout 11

October 21, 2004

Shoukat Ali

shoukat@umr.edu

UMR UNIVERSITY OF MISSOURI-ROLLA
The Name. The Degree. The Difference.

Dynamic Scheduling Using Tomasulo's Algorithm

- used in IBM 360/91 floating-point unit
  - three years after the CDC scoreboard
- goal: instruction re-ordering without special compilers
- lead to Alpha 21264, MIPS, Pentium, PowerPC, and others

A Definition

- definition: a pending operand for instruction X is the operand that yet has to be written by an earlier instruction
- e.g., in `ld r1 0(r2)` / `add r2, r1, r3` — r1 is pending operand

3

WAR Problems With Scoreboard

- WAR made worse in Scoreboard because Scoreboard required that ***all*** operands be read at the same time
  - even those that are available sooner!!

div *f0*, f2, f4
add f10, *f0*, f8
mult f8, f7, f14

add will be fetched, issued, but will stay in RO stage until f0 is produced by **div**

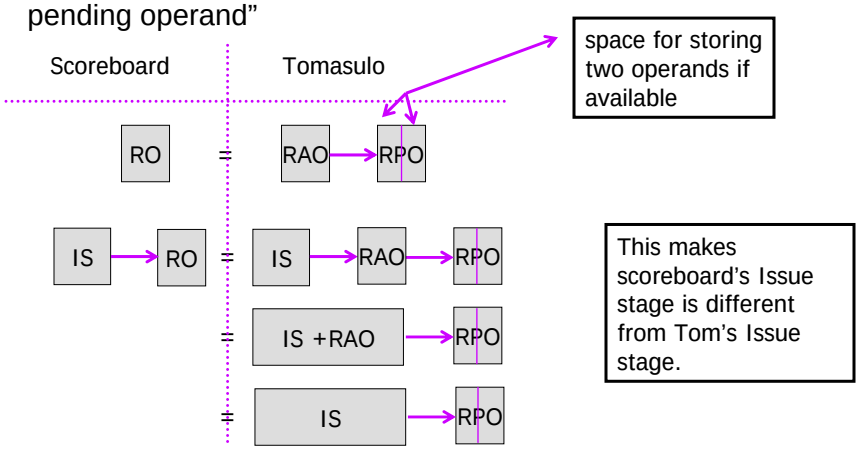
mult will be fetched, issued, its ops read, executed but its result will not be written back until f8 has been read by **add**

if f8 is read as soon as it is available, mult could write its result as soon as it is generated

4

Fixing Scoreboard's WAR Problems

- Scoreboard requires that ***all*** operands be read at the same time
- Tomasulo reads all available operands and then waits for pending operands
- RO stage should be split into "read available operand" and "read pending operand"



5

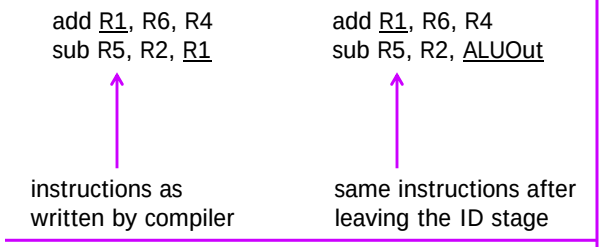
Scoreboard's Forwarding Problems

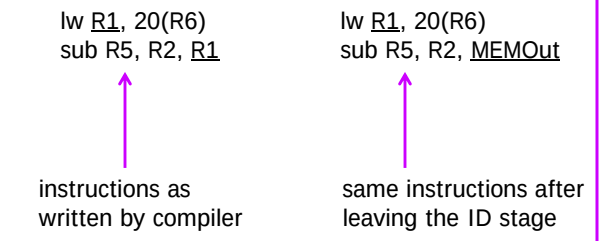
- Scoreboard requires that every source operand must be read from the register file
- Tomasulo re-instituted forwarding

6

How Was Forwarding Implemented in MIPS?

- ID stage would implement forwarding





In the ID stage, an instruction:

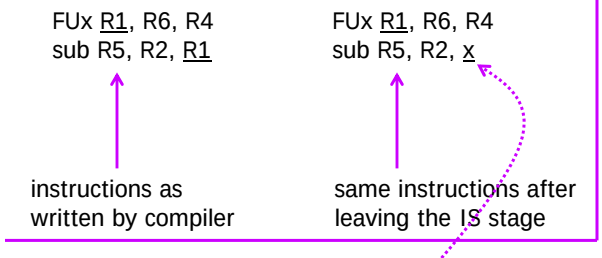
1) reads its available operand

2) for unavailable operand, its register name is replaced with the name of the FU that will forward the value

7

How Is Forwarding Implemented in Tom's Alg?

- IS stage would implement forwarding



In the IS stage, an instruction:

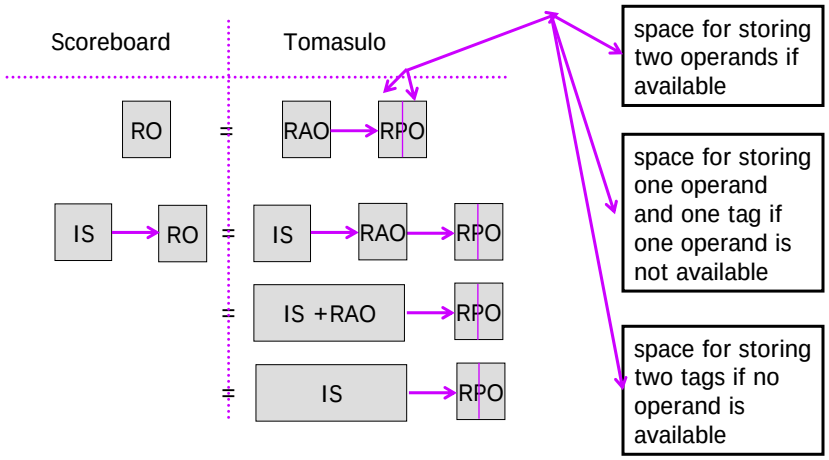
1) reads its available operand

2) for unavailable operand, its register name is replaced with the tag of the FU that will forward the value

tag of the unavailable operand is simply the functional unit number of the producer instruction

8

How Is the RPO Buffer Space Being Used?



NOTE the difference:
In Scoreboard, only the instruction was buffered before being executed.
In Tom's Alg, instruction as well as operands or tags are buffered before instruction gets executed.

9

Reservation Stations

- name for the buffer associated with the RPO stage
- any waiting required in RPO stage happens in a "reservation station"
- a reservation station does the following:
  - buffers an instruction
  - buffers the available operand(s)
  - for any pending operands, RS buffers *tags* of the pending operands

10

WAW Problems With Scoreboard

- Scorboard says an instruction X cannot be sent from Issue to Read Operand stage if there is an active instruction that has yet to write to destination register of X

```
div f0, f2, f4
add f0, f7, f8
sub f6, f10, f0
```

- assume div is executing in the divide unit
- will add be fetched, be issued?

11

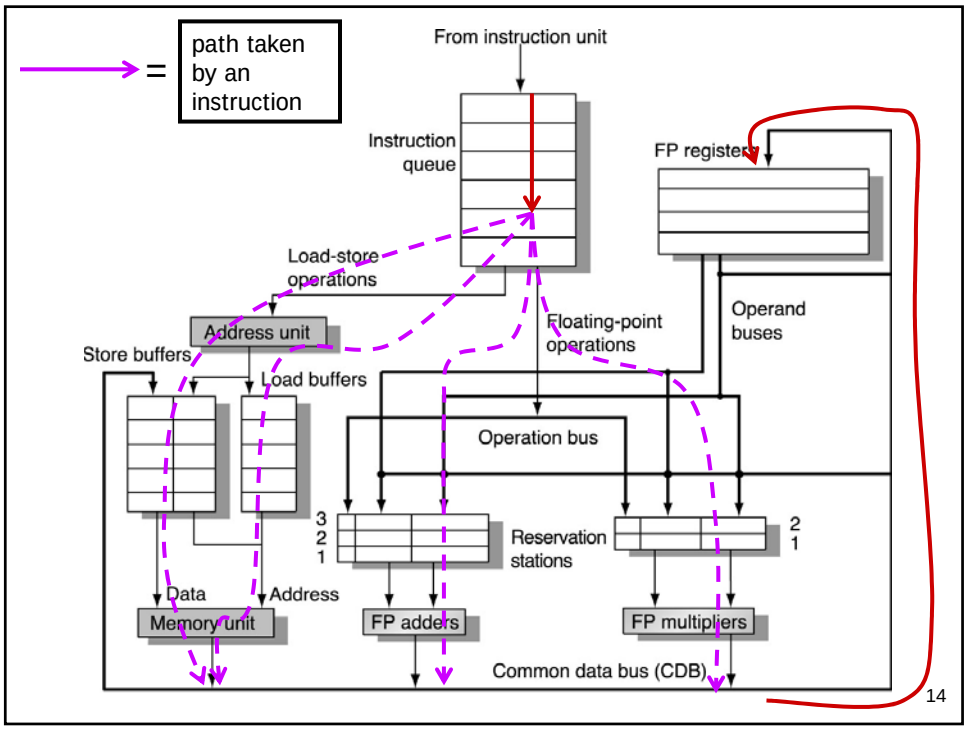
Fixing Scoreboard's WAW Problems

- will see the solution when we do an example tomorrow

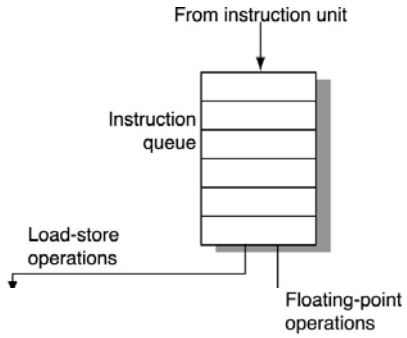
12

A Pipeline Scheduled Dynamically Using Tomasulo's Algorithm

13



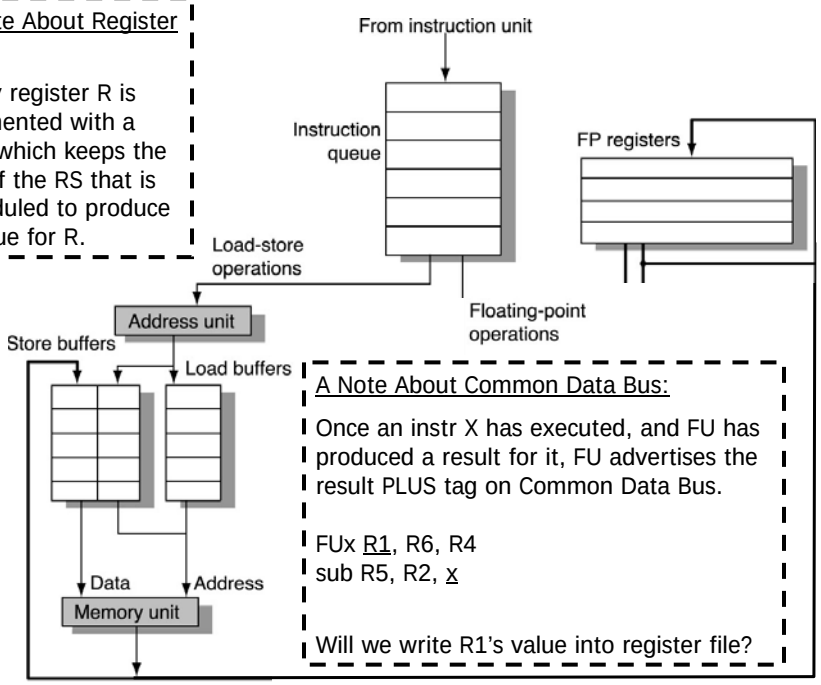
14



15

A Note About Register File:

Every register R is augmented with a field which keeps the tag of the RS that is scheduled to produce a value for R.



A Note About Common Data Bus:

Once an instr X has executed, and FU has produced a result for it, FU advertises the result PLUS tag on Common Data Bus.

FUx R1, R6, R4
sub R5, R2, x

Will we write R1's value into register file?

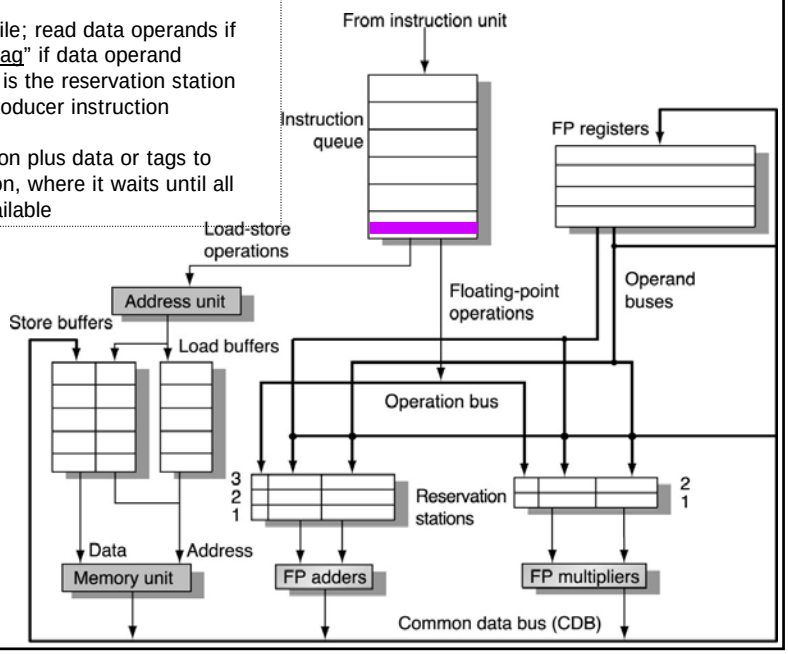
16

Stage 1: Issue

1) check for structural hazard; stall in issue stage if no free reservation station

2) read register file; read data operands if available; read "tag" if data operand unavailable; tag is the reservation station number of the producer instruction

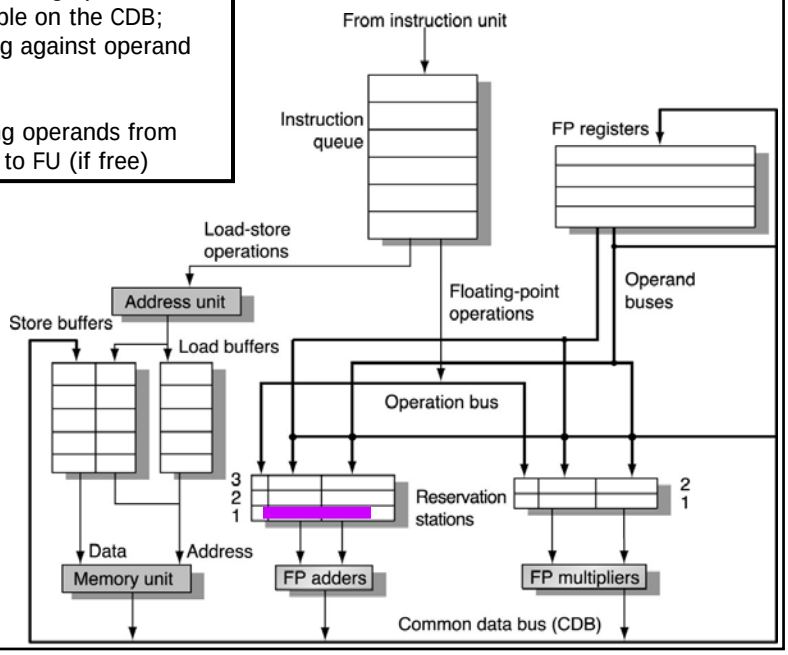
3) route instruction plus data or tags to reservation station, where it waits until all operands are available



Stage 2: Read Pending Operands

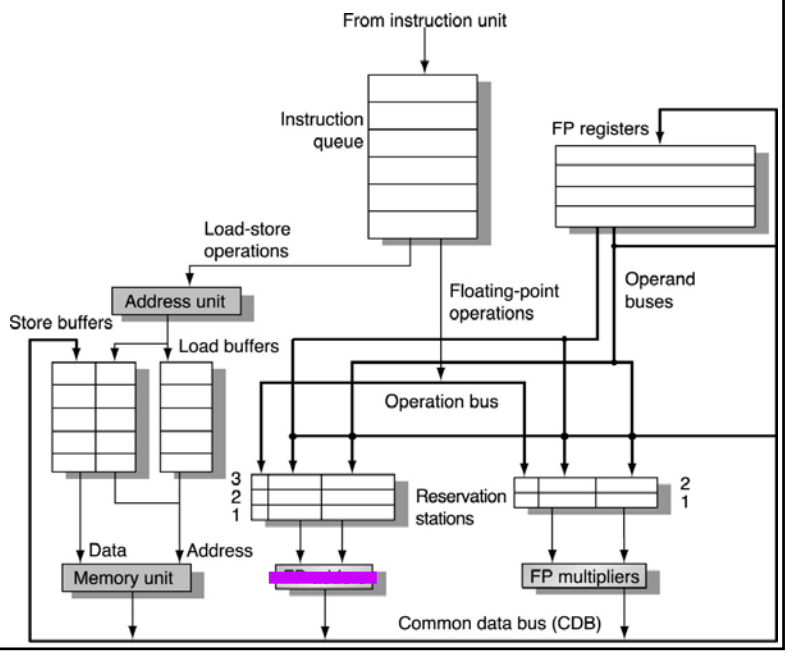
RPO (Read Pending Operands)
1) wait until pending operands become available on the CDB; (match CDB tag against operand tags)

2) grab pending operands from CDB and issue to FU (if free)



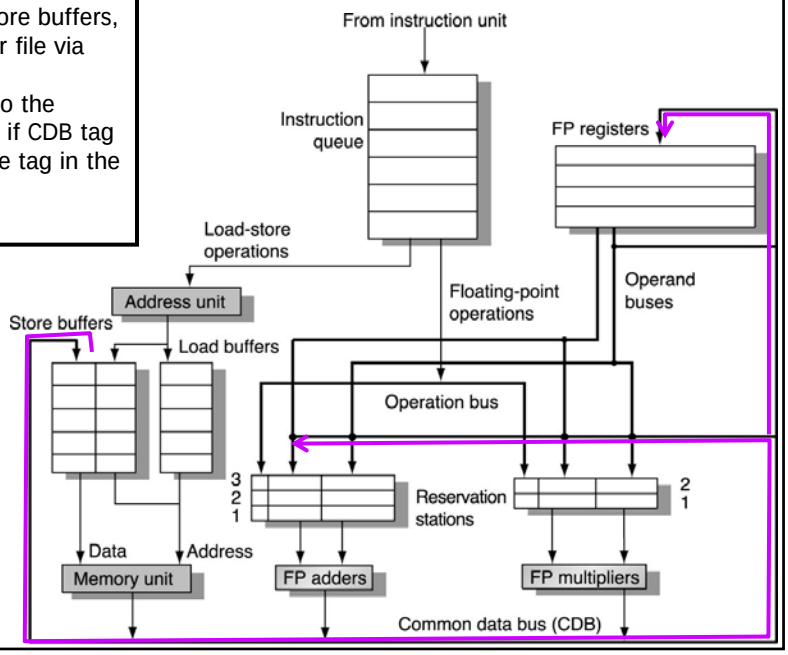
Just execute...

Stage 3: Execute



1) broadcast result + tag to all reservation stations, store buffers, and register file via the CDB
2) only write to the register file if CDB tag matches the tag in the register file

Stage 4: Write Result Back



Repeat: First Two Stages After Fetch of Tomasulo's Alg

- issue
  - check for structural hazard
    - stall in issue stage if no free reservation station
  - read register file
    - read data operands if available
    - read "tag" if data operand unavailable
      - a tag is the reservation station number of the producer instruction
  - route instruction plus data or tags to reservation station, where it waits until all operands are available
- RPO (Read Pending Operands)
  - wait until pending operands become available on the CDB (match CDB tag against operand tags)
  - grab pending operands from CDB and issue to FU (if free)

21

Repeat: Last Two Stages of Tomasulo's Alg

- execute
- write result
  - broadcast result + tag to all reservation stations, store buffers, and register file via the CDB
  - only write register file if CDB tag matches the tag in the register file

22

Textbook Terminology Difference

- textbook specifies only three stages for Tomasulo
- I have used four to contrast with scoreboard technique

23

Key Aspects of Tomasulo's Algorithm – 1

- all available register operands are read at issue stage
  - they are buffered along with the instruction in "reservation stations"
  - CDC: only buffers the instruction, all operands are read from register file when all operands are ready (operands read at issue stage)
    - CDC – late reads
    - Tomasulo – early reads: early reads help WAR condition

24

Key Aspects of Tomasulo's Algorithm – 2

- unavailable registers are ***renamed*** at "read pending operands" stage
  - waiting instructions replace a pending operand's register specifier with a "tag" indicating the producer instruction
  - register specifiers are used only once, at dispatch!
  - renaming eliminates WAR/WAW hazards
    versus
  - CDC: WAR avoided by stalls

- successive writes to a register
  - only last one is actually used to update register
    - helps WAW condition
  - CDC: stall in issue until WAW hazard goes away

25

Other Differences With CDC Scoreboard

- results to FU from RS, not through registers, over Common Data Bus (CDB) that broadcasts results to all FUs and register file
  - don't have to wait for value to go through register file (i.e., use bypasses)
  - functional units don't contend for register file ports
    - each RS can read its value off the CDB simultaneously

- distributed control
  - each reservation station monitors the CDB for pending operands
    versus
  - CDC: scoreboard monitors register file for pending operands

26