Lecture #9

****
Review -- 1 min
****

Scoreboard
- split ID into 2 parts: Issue, Read Operands
- 3 main parts of scoreboard
  - instruction status
  - fu status
  - register result status
- dynamically resolve RAW,
- stall on structural, WAW, WAR

Bug in slides

Thanks to Yang for catching this

Slides have persistent error
- should happen: after FU reads registers, set Rj and Rk to "No"

- Interpret Rj and Rk "Pending read of current contents"

- example: cycle 1-->2
  - after cycle 2, we've read the operands so no pending read to Rk

- example: cycle 8a == figure 4.4 in book
- example: cycle 19 == figure 4.5 in book

- --> Important for disambiguating RAW and WAR
- suppose I have

ADDD F0, F2, F4
MULD F6, F0, F8

(RAW)

Then when mult issues, it will look in reg result station, see that F0 is pending, and not read its operands. It will set Rj to "No"

When ADDD completes, it sees "No" in Rj

Interpret that "someone is reading F0 and isn't happy with the version that's there. They must want the version I just computed." --> write new version of F0

--> successfully solve RAW hazard

Suppose I have

DIVD F10, F12, F14
MULD F6, F10, F0
ADD F0, F2, F4

When Mult issues, it will say Fj is "No" (for F10) and Fk is "Yes" (for f0)

When add completes, it looks at Fk, sees that the multiply is happy with the version of F0 that is there now, so it had better not overwrite it

--> stall --> solve WAR

Last Example: Cycles 16-22

Review:

(Slide: Cycle 62)

In-order issue

out-of-order execution

out of order completion

****
Outline - 1 min
****
Tomasulo
Algorithm
Example

****
Lecture - 20 min
****

Tomasulo Algorithm

IBM 360/91 about 3 years after CDC6600 (1966)
{CDC announces fastest computer in world
--> IBM announces that they are building a faster machine
--> no one buys CDC)

Goal: High Performance w/o special compilers
IBM "Family" of implementations of 360 architecture
--> want 1 binary that runs well on many machines

Difference between IBM 360 and CDC6600 ISA
IBM has 2 registers per instruction (v. 3 for CDC)
***IBM has only 4 FP registers (v. 8 in CDC) ***

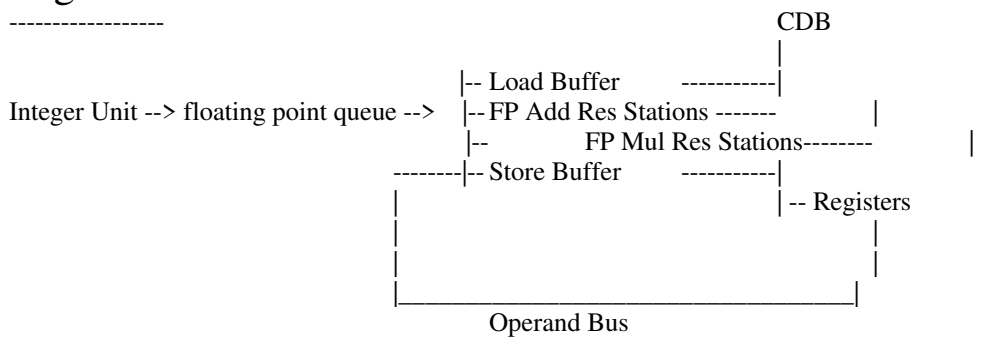
Tomasulo v. Scoreboard
control & bufers *distributed* w/ FUs (v. centralized in scoreboard)
(called "reservation stations")

registers in instructions replaced by pointers to reservation

stations
HW *renaming* of registers to avoid WAR, WAW
CDB: Common Data Bus broadcasts results to all FUs
Loads and Stores treated as FUs as well

Motivation for studying tomasulo
leads to Pentium Pro, PowerPC 604, ...

Organization



Reservation Station Components

Op -- operation to perform in theunit
Qj, Qk -- Reservation stations producing source registers
Vj, Vk -- Value of Source operands
Rj, Rk -- Flags indicating when Vj and Vk are ready
Busy

Register result status -- indicates which FU will write register

3 Stages
---
1. Issue -- get instruction from FP Op queue
    if reservation station free (no structural hazard)
    --> scoreboard issues instr and sends operands
    (RENAMES REGISTERS)

    Notice -- only stall on struct hazard
        not on WAW

    <stop; talk about why this is called register renaming>

2. Execution -- operate on operands (EX)
    if operands not ready, watch CBD for result
    when both ready, then execute

3. Write result
    write on CDB to all awaiting units
    mark reservation station available

    Normal bus : <data, destination> -- "Go to bus"
    CDB: <data, source> -- "Come from bus"

****
Admin - 3 min
****

****
Lecture - 24 min
****

example 1
---

example 2 Loop
---
format -- show a slide, have a student explain it
(start in back of class and just go student to student)

NOTE: Cycle 6 is the crucial one
-- load can issue because memory disambiguation
allows load to go in front of store
-- copy of load1 will never go to register F0

****
Summay - 1 min
****
o prevents register as bottleneck
o avoids WAR, WAW hazards
o allows loop unrolling in HW
o not limited to basic blocks
(integer unit could get ahead of FP)
o lasting contributions
-- dynamic scheduling
-- register renaming
-- load/store disambiguation
o 360/91 decendents are
PowerPC604, 620
MIPS R10000

HP-PA 8000
Intel PPro