Lecture 8: Modern Dynamic Instruction Scheduling

Tomasulo weakness, data forwarding, reg mapping table, generic superscalar models, examples

Tomasulo Performance

Observation at the EX stage, how many cycles to execute this code?

LW R2, 45(R3)
ADD R6, R2, R4
SUB R10, R0, R6
ADD R10, R10, R12

Assume load takes 1 cycle, ALU 1 cycle

Tomasulo vs MIPS Pipeline

- How many cycles on the 5-stage MIPS pipeline?
- Why does the simple pipeline run faster?

Tomasulo Complexity and Efficiency

Modern processors employ deep pipeline

=> Can the rename stage be finished in one fast cycle?

=> How are register content storages?

Review Tomasulo Inst Scheduling

Both in RS, no contention on CDB or FU

ADD R2, R2, 45 # R2=>tag p, result = A
SUB R6, R2, R4 # R4 is ready, = B

Cycle 1: ADD starts at FU, producing A
Cycle 2: ADD broadcast p + A
Cycle 3: SUB matches on p and accepts A

A is produced at cycle 1, but consumed at cycle 3 -- unavoidable?

Review Data Forwarding

MIPS pipeline data forwarding
FU/MEM => FU
Why not in Tomasulo?

But tag broadcasting has one cycle delay!

When is it known that A will be ready?
Cycle 1: A is to be ready
Cycle 2: A and its tag are broadcast
If tag is broadcast one-cycle earlier...
**Revise Scheduling**

RS1: ADD R6, R2, R4
RS2: SUB R10, R5, R6
RS3: ADD R10, R10, R6

ADD(1) has been ready and selected:
1. - ADD(1)'s tag is broadcast, and operands are sent to FU.
   - SUB is waken up and selected;
2. - SUB's tag is broadcast, operands are sent to FU.
   - Forwarding logic replaces 2nd FU operand with FU output.
   - ADD(2) is waken up and accepts FU output, and is selected
3. So on and so forth...

RS can be centralized or distributed

*Updated

**Revise Pipeline Stages**

- FETCH
- ISSUE
- EXE
- WB
- COMMIT

**Examples: Intel P6**

- Decode
- Rename
- ROB Rd
- Pentium III
- ROB
- Data States

- 40-entry ROB
- 20-entry RS station
- Register Alias Table

**Rethink RS and ROB design**

- Data broadcasting to RS stations:
  - Broadcasting saves reg-write to reg-read delay
  - n child instructions can receive data simultaneously

However,
- Data forwarding can be used
- Not all n child instructions may fu-execute next cycle
- RS and ROB may store duplicate values

**Register Mapping Approach**

- Rename *architectural* register to *physical* register
- NO real architectural registers (now virtual register)
- RS \( \rightarrow \) issue queue
- Rename stage: allocate issue queue entry, allocate ROB, allocate physical register
- What is tag now?

- Mapping Table
  - alloc
  - free list

- p1
- p2
- p3

- p_n

Physical register: collection of all temporary register contents

- Physical register
- op
- Qj
- Qk
- bus
- Vj
- Vk
- p1
- p2
- p3
- p_n
Mis-speculation Recovery
- RS+ROB: no changes to arch. registers, so just clear pipeline and re-fetch
- Fundamental issue: software does not see wrong register contents

Recovery for mapping approach: Roll back mapping table to the mis-speculation point

Architectural registers => virtual registers

How to implement mapping table supporting recovery?

Change of pipeline

Example: Intel Pentium 4

Alpha 21264 Pipeline

Generic Superscalar Processor Models

Summary of Dynamic Scheduling
- Pipeline stages
  - Renaming (in-order)
  - Schedule
  - Commit (in-order)
- Two organizations
  - Mapping table + phy req + issue queue + ROB => REN => SCHD => REG
  - Reg alias table + RS + ROB, req in RS and ROB => REG
- Scheduling methods
  - Tag broadcasting vs. scoreboard (later)
  - CDC6600: introduces scoring
  - Tomasulo: introduces renaming and tag broadcasting
  - Reorder buffer: provides in-order commit
  - Real OOO processors very complicated (like a vehicle)
  - bring impl variants but still root in these basic designs

Source: Paracharla PhD thesis 1998