Tomasulo’s Algorithm

- Architecture to increase ILP
  - Removes WAR and WAW dependencies during issue
- WAR and WAW
  - Name Dependencies
  - Artifact of using the same storage location (variable name)
  - Can be avoided by renaming the conflicting source or destination operands
  - Last class: renames destinations by allocating new physical register
  - Tomasulo’s scheme: renames source operand
Renaming using Logical/Physical Registers

Example

A:   DIVD F0, F2, F4  
     RAW dependency between (A,B)
B:   ADDD F6, F0, F8  
     WAR dependency between (B,C)
C:   SUBD F8, F10, F12 
     RAW dependency between (C,D)
D:   DIVD F0, F8, F10

• Rename Destination Register prior to Issue

A:   [ F0:T0; F2: T2; F4: T4] [F0: T14]  
     DIVD T14, T2, T4
B:   [F6: T2; F0: T14; F8: T8] [F6:T15]  
     ADDD T15, T14, T8
C:   [F8: T8; F10: T10; F12: T12][F8: T16]  
     SUBD T16, T10, T12
D:   [F0: T14; F8: T16; F10:T10 ] [F0: T17]  
     DIVD T17, T16, T10
Renaming Source Registers

Example

A:   DIVD F0, F2, F4  RAW dependency between (A,B)
B:   ADDD F6, F0, F8  WAR dependency between (B,C)
C:   SUBD F8, F10, F12 RAW dependency between (C,D)
D:   DIVD F0, F8, F10

• Rename Source Registers during Issue

A:   [F2: T0; F4: T1]  DIVD   F0, T0, T1
B:   [F0: T2; F8: T3]  ADDD   F6, T2, T3
C:   [F10: T4; F12: T5] SUBD   F8, T4, T5
D:   [F8: T6; F0: T7]  DIVD   F0, T6, T7
Renaming Source Registers

Example

A: \text{DIVD} \ F0, F2, F4 \quad \text{(RAW dependency between (A,B))}
B: \text{ADDD} F6, F0, F8 \quad \text{(WAR dependency between (B,C))}
C: \text{SUBD} F8, F10, F12 \quad \text{(RAW dependency between (C,D))}
D: \text{DIVD} F0, F8, F10

• Need to **Initialize** Renamed Registers

A: \[ T0 = F2; \ T1 = F4 \] \quad \text{DIVD} \ F0, T0, T1
B: \[ T2 = ?; \ T3 = F8 \] \quad \text{ADDD} \ F6, T2, T3
C: \[ T4 = F10; \ T5 = F12 \] \quad \text{SUBD} \ F8, T4, T5
D: \[ T6 = ?; \ T7 = F10 \] \quad \text{DIVD} \ F14, T6, T7

If the source register has a “stable” value (*i.e.* it is not the destination of an in-flight instruction): **Copy its value to the renamed register**

If the value is a “future value” (*i.e.* it is destination of an in-flight instruction): **Obtain it from the producing instruction when it is available**
MIPS Pipeline with Tomasulo’s Algorithm

Common Data Bus (CDB)
Overview (Tomasulo’s Algorithm)

**RS: Reservation Station**
Set of Internal registers: (T registers)

- Temporary storage corresponding to renamed register
- Each RS register initialized with either
  - Value of the actual source register
  - Information identifying the producer of the register value
- Issued instructions wait in the Reservation Station till it can be Dispatched to the Execution Unit
Overview (Tomasulo’s Algorithm)

CDB: Common Data Bus

Broadcast bus

- Used to transmit results of completing instructions in the WRITE stage to units that are waiting for the completion of an instruction:
  - Register file
  - Reservations Stations
Overview (Tomasulo’s Algorithm)

Issue Unit:

- Stall if no free RS register available (Structural Hazard)
- Does not stall pipeline for a WAW dependency
- Transform the current instruction by renaming the source operands
  - Avoids WAR hazards
- Store transformed instruction in selected reservation station (RS) register
- Record dependency information (data flow graph) in RS register
Overview (Tomasulo’s Algorithm)

Dispatch Unit:

- Choose one (or more) instruction(s) from the RS whose
  - **Operands** are available
  - **Functional unit** is free
- **Dispatch** to appropriate functional unit

Write Unit:

- Select a completed instruction to write back
- Broadcast the value (and identifying information) to all units waiting for it
WAR Dependencies

A: ADD.D F0, F2, F4  RAW: (A, B)
B: DIV.D F6, F0, F8  WAR: (B, C)
C: MUL.D F8, F20, F12

• Remove WAR dependency by renaming the source register (F8) in B
  A: ADD.D F0, F2, F4
  B: DIV.D F6, F0, T0
  C: MUL.D F8, F20, F12

• T0 must be initialized with value equal to current value of F8
  • Initialize T0 by copying current value of F8 to T0 during Issue
WAR Dependencies

A: ADD.D F0, F2, F4  \hspace{2cm} \text{RAW: (A, B)}
B: DIV.D F6, F0, F8  \hspace{2cm} \text{WAR: (B, C)}
C: MUL.D F0, F20, F12 \hspace{2cm} \text{WAW: (A,C)}

• Remove WAR dependency by renaming the source register (F0) in B
  A: ADD.D F0, F2, F4
  B: DIV.D F6, T1, F8
  C: MUL.D F0, F20, F12

• T1 must be initialized with value produced by A (RAW dependence)
  • A is still in-flight when B is issued
    • \textit{Cannot} initialize T1 by copying current value of F0 to T1
    • T1 must be initialized with result of A
      • Uses forwarding via the CDB
      • When B is issued tag it to indicate its producer is A
WAW Dependencies

A: ADD.D F0, F2, F4  RAW: (A, B)
B: DIV.D F6, F0, F8  WAR: (B, C)
C: MUL.D F0, F20, F12 WAW: (A, C)

- Avoid potential WAW hazard by canceling A’s write to F0 if A is in-flight

- Why is it OK to cancel A’s write of F0?
  - B gets its value produced by A directly from the CDB using forwarding
  - True of all instructions between A and C that read F0
  - Instructions following this instruction want the new value if F0
Issue Unit

1. Find a free RS register to issue current instruction I (call it RS$_I$); else stall

2. Suppose register S is a source register for instruction I
   - If S has its most recent value (not destination of any in-flight instruction) then its value is copied into RS$_I$
   - Any later instruction is free to overwrite S (removes WAR dependency)

   (b) If S is pending a write by instruction J: (ID of instruction is important!!)
      - Record that the source operand of RS$_I$ is produced by instruction J
      - The result of J is passed directly to RS$_I$ (bypass the register) using CDB
      - Later instructions that write S will not affect I (no WAR)

3. Suppose register D is a destination register for instruction I
   - If J is an in-flight instruction (the most recent) that has D as a destination
   - Make I the writer of D and cancel the write by J.
   - Avoids WAW hazards

   - Only need to track the id of the most recent writer to a register
   - An in-flight instruction I will be identified by the number of the Reservation Station to which it is issued.
### Example (Issue Unit)

**Issue A**: DIVD F0, F2, F4

- **TAG**: RS_A
- **SOURCE1**: v₂
- **SOURCE2**: v₄
- **OP**: DIVD

- *v₂, v₄*: Values read from registers F2, F4 during issue
- *Record A as the current writer of F0*

**Issue B**: ADDD F6, F0, F8

- **TAG**: RS_B
- **SOURCE1**: v₈
- **OP**: ADDD

- *v₈*: Value read from F8
- *Other operand will be result of instruction A*
Example (Issue Unit)

**Issue C:** SUBD F8, F10, F12

<table>
<thead>
<tr>
<th>RS_C</th>
<th>TAG</th>
<th>SOURCE1</th>
<th>SOURCE2</th>
<th>OP</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS_C</td>
<td>v_{10}</td>
<td>v_{12}</td>
<td>SUBD</td>
<td></td>
</tr>
</tbody>
</table>

v_{10}, v_{12}: Values read from registers F10, F12 during instruction issue

**Issue D:** DIVD F14, F8, F10

<table>
<thead>
<tr>
<th>RS_D</th>
<th>RS_D</th>
<th>v_{10}</th>
<th>DIVD</th>
</tr>
</thead>
</table>

- v_{10}: Value read from F10
- Other operand will be result of instruction C
Dispatch and WB Units

- Dispatch unit selects instructions from RS whose:
  - operands are available and functional unit is free
- In the example A, C are ready to execute.
- When execution is complete the Write Unit is notified

Write Unit selects one of the completed instruction in EX/WB register for WB
For the selected instruction, say RS₁
- Broadcast its result on the CDB
- Broadcast the TAG (RS₁) along with the value

- All units that are waiting on the result of the completing instruction (tag comparison): copy the broadcast value
- Reservation stations copy the value into the RS registers with matching tags
- Register file copies the value into the destination register of the instruction.