Sequential Execution Model

Any program execution is "correct" if the final architectural states (registers and memory contents) is the same as by sequential execution.

Single-cycle implementation is intuitively correct.

If instructions are not executed sequentially, what is "correct" execution?

Data Flow Execution

LD  F2, 0(R3)
MULTI  F0, F2, F4
LD  F6, 0(R2)
SUBD  F8, F6, F2
DIVD  F10, F0, F6
ADD  F12, F8, F2

Note: no branch in this code

Data Dependences

- Instruction J is dependent on I if:
  - J's output is used by J, or
  - J is dependent on K and K is dependent on I.

Loop: LD  F0, 0(R1)
ADD D  F4, F0, F2
S D  F4, 0(R1)
DADDUI R1, R1, # - 8
BNE  R1, R2, LOOP

Data Dependence Graph

Name Dependences

- Antidependence (WAR): one instruction overwrite a register or memory location that a prior instruction reads
  - LW R1, 100(R2)
  - ADD R2, R3, R4

- Output dependence (W/AW): two instructions write the same register or memory location
  - LW R1, 100(R2)
  - Add R2, R1, R2
  - Add R1, R3, R4

Those dependencies can be removed
Dependences vs Hazards
- Dependences are properties of programs
- Hazards are properties of pipelines
- Dependences indicates the potential of hazards
- Pipeline implementations determine actual hazards and the length of any stall

What hazards are exposed by MIPS 5-stage pipeline?

Dynamic Scheduling
- General idea: when an instruction stalls, look for independent instructions following it
  - $\text{DIV, F0, F2, F4}$
  - $\text{ADD, F10, F0, F8}$
  - $\text{SUB, F12, F8, F14}$

- Instruction window: how far to look ahead
- Out-of-order execution
- Respect data dependence

What hazards would be exposed?

Machine Correctness
- Let $E(M,P)$ be the execution of $P$ at a given machine $M$.
- $E(M,P)$ is correct if and only if $E(M,P) = E(S,P)$: The register and memory contents at the end of $E(D,P)$ are the same as those of a sequence execution.

Machine Correctness
- Let $E(D,P)$ be the execution of $P$ on a dynamically scheduled machine $D$.
- $E(D,P) = E(S,P)$ if:
  - $E(D,P)$ and $E(S,P)$ execute the same set of instructions
  - For any inst $i$, $i$ produces the same output as in $E(D,P)$ and $E(S,P)$
  - Any register or memory word receives the output from the same instruction in $E(D,P)$ and in $E(S,P)$

Machine Correctness
- For any inst $i$, $i$ produces the same output as in $E(D,P)$ and $E(S,P)$
- For any inst $i$, $i$ receives the same inputs in $E(D,P)$ as in $E(S,P)$
- For any inst $i$, $i$ receives the outputs in $E(D,P)$ of its parents in $E(S,P)$
- Any register or memory word receives the output from the same instruction in $E(D,P)$ and in $E(S,P)$
- Any register or memory word writes to the register or memory word in $E(S,P)$

Machine Correctness
- For any inst $i$, $i$ produces the same output as in $E(D,P)$ and $E(S,P)$
- For any inst $i$, $i$ receives the same inputs in $E(D,P)$ as in $E(S,P)$
- For any inst $i$, $i$ receives the outputs in $E(D,P)$ of its parents in $E(S,P)$
- Any register or memory word receives the output from the same instruction in $E(D,P)$ and in $E(S,P)$
- Any register or memory word writes to the register or memory word in $E(S,P)$

Machine Correctness
- Any register or memory word receives the output from the same instruction in $E(D,P)$ and in $E(S,P)$
- In $E(D,P)$ any register or memory word receives the output of inst $j$, where $j$ is the last instruction written to the register or memory word in $E(S,P)$
Data Dependence between Operations

<table>
<thead>
<tr>
<th>ALU to ALU</th>
<th>Load and other inst</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBD F5, F6, F2</td>
<td>LD F2, 0 (R3)</td>
</tr>
<tr>
<td>ADD F6, F8, F2</td>
<td>MULTI F0, F2, F4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SUBD</th>
<th>ADD</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF</td>
<td>IF</td>
</tr>
<tr>
<td>ID</td>
<td>ID</td>
</tr>
<tr>
<td>EX</td>
<td>--</td>
</tr>
<tr>
<td>WB</td>
<td>EX</td>
</tr>
<tr>
<td></td>
<td>WB</td>
</tr>
</tbody>
</table>

Dependences between Operations

<table>
<thead>
<tr>
<th>Store to load</th>
</tr>
</thead>
<tbody>
<tr>
<td>//R3+100==R4?</td>
</tr>
<tr>
<td>S.D F6, 100 (R3)</td>
</tr>
<tr>
<td>L.D F2, 0 (R4)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S.D</th>
<th>L.D</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF</td>
<td>IF</td>
</tr>
<tr>
<td>ID</td>
<td>ID</td>
</tr>
<tr>
<td>EX</td>
<td>EX</td>
</tr>
<tr>
<td>MEM</td>
<td>MEM</td>
</tr>
<tr>
<td>WB</td>
<td>MEM</td>
</tr>
</tbody>
</table>

Dynamic Scheduling

<table>
<thead>
<tr>
<th>L.D</th>
<th>F2, 0 (R3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MULTI</td>
<td>F0, F2, F4</td>
</tr>
<tr>
<td>L.D</td>
<td>F6, 0 (R2)</td>
</tr>
<tr>
<td>SUB.D</td>
<td>F8, F6, F2</td>
</tr>
<tr>
<td>DIV.D</td>
<td>F10, F0, F6</td>
</tr>
<tr>
<td>ADD.D</td>
<td>F12, F8, F2</td>
</tr>
</tbody>
</table>

How to schedule pipeline operations?

Is This Working?

<table>
<thead>
<tr>
<th>Inst</th>
<th>IF</th>
<th>ID</th>
<th>Schd</th>
<th>EXE</th>
<th>MEM</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>MULT</td>
<td>1</td>
<td>2</td>
<td>3-5</td>
<td>6-11</td>
<td>-</td>
<td>12</td>
</tr>
<tr>
<td>LD</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5-2</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>SUBD</td>
<td>2</td>
<td>3</td>
<td>4-6</td>
<td>7-18</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>DIVD</td>
<td>3</td>
<td>4</td>
<td>5-11</td>
<td>12-31</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>AddD</td>
<td>3</td>
<td>4</td>
<td>5-8</td>
<td>9-10</td>
<td>12</td>
<td>14</td>
</tr>
</tbody>
</table>

Assume (1) two-way issue; (2) FU delay as implied

Dynamic Scheduling Implementation

Scoreboarding:  
1966: scoreboard in CDC6600

Tomasulo:  
3 years later in IBM 360/91  
Introducing register renaming  
Use tag-based instruction wakeup

Name Dependences and Register Renaming

Original code:  
ADD R3, R1, R2  
SUB R4, R3  
ADD R3, R6, R7  
SUB R3, R3, R4  
What prevents parallelism?

Renamed code:  
R3, R4, R3, R3 renamed to  
P6, P7, P8, P9 sequentially  
ADD P6, R1, R2  
SUB P7, R4, P6  
ADD P8, R6, R7  
SUB P9, R5, P7  
Finally R3 <= P9, R4 <= P7
Register Renaming and Correctness
- E(D, P) and E(S, P) execute the same set of instructions
- For any inst i, i receives the outputs in E(D, P) of its parents in E(S, P)
- Any register or memory word receives the output of inst j, where j is the last instruction writes to the register or memory word in E(S, P)

Renaming Implementation
First proposed in Tomasulo (1969)
- Use register status table
- Renamed to reservation station
- No separate architectural/physical registers; no copy-back

In P-III
- Use register alias table
- Renamed arch. register to physical register
- Data copied back to arch. register

Speculative Execution
- Modern processors must speculate!
  - Branch prediction: SPEC2k INT has one branch per seven instructions!
  - Precise interrupt
  - Memory disambiguation
  - More performance-oriented speculations

Two disjointed but connected issues:
1. How to make the best prediction
2. What to do when the speculation is wrong

Control Speculation
- Branch prediction - control speculation
  - Must predict on branches
  - What to predict
  - Branch direction
  - Branch target address

What info can be used
- PC value
- Previous branch outcomes
- Also use branch patterns in complex branch predictors

What building blocks are needed
- Branch prediction table (BPT), branch target buffer (BTB), pattern registers, and some logic

Generic Superscalar Processor Models
Issue queue based
- Instruction queue
- Scheduler
- Execute
- Commit

Reservation based
- Reservation
- Execute
- Commit

Revised from Parachara PhD thesis 1998