Control Hazards on Branches

12: beq r1, r3, 24
16: and r2, r3, r5
20: or r6, r1, r7
24: add r8, r1, r9
...
36: xor r10, r1, r11 → Branch Target
# MIPS pipeline

**Control Hazard on Branches**

**Two Stage Stall**

12: `beq r1, r3, 24`

16: `and r2, r3, r5`

20: `or r6, r1, r7`

36: `xor r10, r1, r11`  \(\text{Target}\)

 Freeze or Flush (higher performance but must undo side effects)
Branch Stall Impact - MIPS R4000

- Deeper pipeline -> worse penalty

Assume CPI = 1.0 ignoring branches & data hazards
Assume solution was stalling for 3 cycles for every branch
If 20% branch, Stall 3 cycles

<table>
<thead>
<tr>
<th>Op</th>
<th>Freq</th>
<th>Cycles</th>
<th>CPI(i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Other</td>
<td>80%</td>
<td>1</td>
<td>0.8</td>
</tr>
<tr>
<td>Branch</td>
<td>20%</td>
<td>4</td>
<td>0.8</td>
</tr>
</tbody>
</table>

⇒ new CPI = 1.6 or 60% slower
Conditional and unconditional Branch

- Unconditional branches incur lower penalty
  - Assume 2 cycles vs 3 for unconditional branch

**Avg.Stall cycles** = \( \sum \text{Branch\_frequency} \times \text{Branch\_penalty} \)

<table>
<thead>
<tr>
<th>Branch type</th>
<th>Freq.</th>
<th>Penalty</th>
<th>Freq \times Penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unconditional</td>
<td>0.04</td>
<td>2</td>
<td>0.08</td>
</tr>
<tr>
<td>Conditional</td>
<td>0.16</td>
<td>3</td>
<td>0.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.56</td>
</tr>
</tbody>
</table>

\[
\text{Pipeline\_speedup} = \frac{\text{Pipeline\_depth}}{1 + \sum \text{Branch\_frequency} \times \text{Branch\_penalty}}
\]

\[
\text{Pipeline\_speedup} = \frac{8}{1.56} = 5.13
\]

(ignoring all other stalls)

Dealing with Branch in MIPS

- Branch penalty is significant
- Two part solution:
  - Determine branch taken or not sooner, AND
  - Compute taken branch address earlier
- **MIPS branch tests if register = 0 or \( \neq 0 \)**
- **MIPS Solution:**
  - Move Zero test to ID/RF stage
  - Adder to calculate new PC in ID/RF stage
  - 1 clock cycle penalty for branch versus 2
Original MIPS 5-stage pipeline


Memory Access

Write Back

Modified MIPS Pipeline


Memory Access

Write Back
Four Branch Hazard Alternatives - 1 & 2

#1: Stall until branch direction is known
#2: Static Prediction:

(a) Predict Branch Not Taken
- Execute successor instructions in sequence
- "Squash" instructions in pipeline if branch actually taken
- 47% MIPS branches not taken on average

(b) Predict Branch Taken
- 53% MIPS branches taken on average
- But haven't calculated yet branch target address
  » MIPS still incurs 1 cycle branch penalty
  » Other machines: branch target known before outcome

Static Prediction - Not Taken

Predict: guess Not Taken then back up if wrong
Impact: 0 lost cycles per branch instruction if right, 1 if wrong (right about 50% of time)
Need to "Squash" following instruction (LOAD) if wrong
**Predict Not Taken**

CPI for branch: 
\[(1 \times 0.47 + 2 \times 0.53) = 1.53\]

Total CPI: 
\[1 + 0.2 \times 0.53 \times 1 = 1.106 \text{ (20\% branch)}\]

Compare to Always Stall (Freeze)

---

**MIPS R4000**

Predict Taken - 2 (vs. 3) stall Cycles

---
CPI Penalty - MIPS R4000

<table>
<thead>
<tr>
<th>Branch scheme</th>
<th>Penalty unconditional</th>
<th>Penalty intaken</th>
<th>Penalty taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stall</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Predicted taken</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Additions to the CPI from branch costs

<table>
<thead>
<tr>
<th>Branch scheme</th>
<th>Unconditional branches</th>
<th>Untaken conditional branches</th>
<th>Taken conditional branches</th>
<th>All branches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency of event</td>
<td>4%</td>
<td>6%</td>
<td>10%</td>
<td>20%</td>
</tr>
<tr>
<td>Stall pipeline</td>
<td>0.08</td>
<td>0.18</td>
<td>0.30</td>
<td>0.56</td>
</tr>
<tr>
<td>Predicted taken</td>
<td>0.08</td>
<td>0.18</td>
<td>0.20</td>
<td>0.46</td>
</tr>
<tr>
<td>Predicted intaken</td>
<td>0.08</td>
<td>0.00</td>
<td>0.30</td>
<td>0.38</td>
</tr>
</tbody>
</table>

CPI_Stall = 1.56,  CPI_P/T = 1.46,  CPI_P/UT = 1.38

Speedup_(P/UT vs. Stall) = 1.56/1.38 = 1.13

Four Branch Hazard Alternatives - 3 & 4

**#3: Delayed Branch**
- Define branch to take place **AFTER** a following instruction(s)

```
branch instruction
  sequential successor1
  sequential successor2
  ..........
  sequential successorn
  ;
  branch target if taken
```

- 1 slot delay allows proper decision and branch target address in 5 stage pipeline
- MIPS uses this

**#4: Dynamic Branch Prediction**
Impact: 0 clock cycles per branch instruction if can find useful instruction to put in "slot" (≈ 50% of time)

Scheduling the branch delay slot

From before
ADD R1, R2, R3
If R2=0 then
Delay Slot

From target
ADD R4, R5, R6
If R1=0 then
Delay Slot

From fall-through
ADD R1, R2, R3
If R1=0 then
Delay Slot

ADD R4, R5, R6
If R1=0 then
ADD R4, R5, R6

ADD R4, R5, R6
If R1=0 then
OR R7, R8, R9
SUB R4, R5, R6
Delayed Branch -

♦ Where to get instructions to fill branch delay slot?
  • Before branch instruction
  • From the target address: only valuable when branch taken
  • From fall through: only valuable when branch not taken
  • Canceling branches allow more slots to be filled

♦ Compiler effectiveness for single branch delay slot:
  • Fills about 60% of branch delay slots
  • About 80% of instructions executed in branch delay slots useful in computation
  • About 48% (60% x 80%) of slots usefully filled
  • CPI = 1 + Prob(Branch) x Prob(Un-usefull_fill) = 1 + .2 x .52 = 1.104

♦ Delayed Branch downside: 8-10 stage pipelines, multiple instructions issued per clock (superscalar)

Evaluating Branch Alternatives
for MIPS

UnCond: 4%, NotTaken_Cond: 6%, Taken_Cond: 10%
Slot filled usefully: 48%
CPI (Delayed_Branch) = 1 + .2 x .52 = 1.104

<table>
<thead>
<tr>
<th>Scheduling scheme</th>
<th>Branch penalty</th>
<th>CPI</th>
<th>speedup v. stall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stall pipeline</td>
<td>1</td>
<td>1.2</td>
<td>1.0</td>
</tr>
<tr>
<td>Predict taken</td>
<td>1</td>
<td>1.2</td>
<td>1.0</td>
</tr>
<tr>
<td>Predict not taken</td>
<td>0/1</td>
<td>1.14</td>
<td>1.05</td>
</tr>
<tr>
<td>Delayed branch</td>
<td>0.52</td>
<td>1.104</td>
<td>1.087</td>
</tr>
</tbody>
</table>
### MIPS R4000 - Delayed Branch and Static Prediction (Not-Taken)

#### (2) Taken

<table>
<thead>
<tr>
<th>Instruction number</th>
<th>Clock number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Branch instruction</td>
<td>IF</td>
</tr>
<tr>
<td>Delay slot</td>
<td>IF</td>
</tr>
<tr>
<td>Stall</td>
<td></td>
</tr>
<tr>
<td>Stall</td>
<td></td>
</tr>
<tr>
<td>Branch target</td>
<td>IF</td>
</tr>
</tbody>
</table>

#### (1) Not-Taken

<table>
<thead>
<tr>
<th>Instruction number</th>
<th>Clock number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Branch instruction</td>
<td>IF</td>
</tr>
<tr>
<td>Delay slot</td>
<td>IF</td>
</tr>
<tr>
<td>Branch instruction + 2</td>
<td>IF</td>
</tr>
<tr>
<td>Branch instruction + 3</td>
<td>IF</td>
</tr>
</tbody>
</table>