### Predict Not Taken

<table>
<thead>
<tr>
<th>Branch</th>
<th>IF</th>
<th>ID</th>
<th>EX</th>
<th>MEM</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>I+1</td>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
</tr>
<tr>
<td>I+2</td>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
</tr>
<tr>
<td>I+3</td>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
</tr>
</tbody>
</table>

Branch Target

<table>
<thead>
<tr>
<th>Branch</th>
<th>IF</th>
<th>ID</th>
<th>EX</th>
<th>MEM</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>T+1</td>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
</tr>
</tbody>
</table>

### Delayed Branch

<table>
<thead>
<tr>
<th>Branch</th>
<th>IF</th>
<th>ID</th>
<th>EX</th>
<th>MEM</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>I+1 (delay slot)</td>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
</tr>
<tr>
<td>I+2</td>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
</tr>
<tr>
<td>I+3</td>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
</tr>
</tbody>
</table>

Branch Target

<table>
<thead>
<tr>
<th>Branch</th>
<th>IF</th>
<th>ID</th>
<th>EX</th>
<th>MEM</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>T+1</td>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
</tr>
</tbody>
</table>

### Filling the delay slot (e.g., in the compiler)

- Can be done when?
- Improves performance when?

```
lw R1, 10000(R7)
add R5, R6, R7
beqz R5, label:
```

```
sub R8, R1, R3
add R4, R8, R9
and R2, R4, R8
```

```
label: lw R2, 1024(R8)
```

### Problems filling delay slot

- **1.** need to predict direction of branch to be most effective
- **2.** limited by correctness restrictions
- correctness restriction can be removed by a canceling branch
  - branch likely or branch not likely
  - e.g.,
    - `beqz likely` delay slot instruction
    - `delay slot instruction`
    - `fall-through instruction`
    - squashed/nullified/canceled if branch not taken
Branch Likely

Branch likely

I+1 (delay slot)

I+2

I+3

Branch Target

T+1

Delay Slot Utilization

- 18% of delay slots left empty
- 11% of delay slots (1) use canceling branches and (2) end up getting canceled

Branch Performance

CPI = BCPI + pipeline stalls from branches per instruction

= 1.0 + branch frequency * branch penalty

assume 20% branches, 67% taken:

branch taken not taken
scheme penalty penalty CPI

Delay Slots, the scorecard

- Pros
- Cons

stall predict taken predict not taken delayed branch
Static Branch Prediction

- Static branch prediction takes place at compile time, dynamic branch prediction during program execution
- static bp done by software, dynamic bp done in hardware
- Static branch prediction enables
  - more effective code scheduling around hazards (how?)
  - more effective use of delay slots

Misprediction rate

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>0%</th>
<th>5%</th>
<th>10%</th>
<th>15%</th>
<th>20%</th>
<th>25%</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcc</td>
<td>22%</td>
<td>18%</td>
<td>12%</td>
<td>12%</td>
<td>5%</td>
<td>6%</td>
</tr>
<tr>
<td>li</td>
<td>11%</td>
<td>5%</td>
<td>5%</td>
<td>6%</td>
<td>10%</td>
<td>15%</td>
</tr>
<tr>
<td>le</td>
<td>9%</td>
<td>10%</td>
<td>12%</td>
<td>15%</td>
<td>20%</td>
<td>25%</td>
</tr>
</tbody>
</table>

But now, the real world interrupts...

- Pipelining is not as easy as we have made it seem so far...
  - interrupts and exceptions
  - long-latency instructions

Exceptions and Interrupts

- Transfer of control flow (to an exception handler) without an explicit branch or jump
- are often unpredictable
- examples
  - I/O device request
  - OS system call
  - arithmetic overflow/underflow
  - FP error
  - page fault
  - memory-protection violation
  - hardware error
  - undefined instruction
Classes of Exceptions

- synchronous vs. asynchronous
- user-initiated vs. coerced
- user maskable vs. nonmaskable
- within instruction vs. between instructions
- resume vs. terminate

when the pipeline can be stopped just before the faulting instruction, and can be restarted from there (if necessary), the pipeline supports *precise exceptions*

Exceptions Can Occur In Several Places in the pipeline

- IF -- page fault on memory access, misaligned memory access, memory-protection violation
- ID -- illegal opcode
- EX -- arithmetic exception
- MEM -- page fault, misaligned access, memory-protection
- WB -- none

(and, of course, asynchronous can happen anytime)

Basic Exception Methodology

- turn off writes for faulting instruction and following
- force a trap into the pipeline at the next IF
- save the PC of the faulting instruction (not quite enough for delayed branches)

Simplifying Exceptions in the ISA

- Each instruction changes machine state only once
  - autoincrement
  - string operations
  - condition codes
- Each instruction changes machine state at the end of the pipeline (when you know it will not cause an exception)
Handling Multicycle Operations

- Unrealistic to expect that all operations take the same amount of time to execute
- FP, some memory operations will take longer
- This violates some of the assumptions of our simple pipeline

New problems

- structural hazards
  - divide unit
  - WB stage
- WAW hazards are possible
- out-of-order completion
- RAW hazards still not possible

Multiple Execution Pipelines

<table>
<thead>
<tr>
<th>FU</th>
<th>Latency</th>
<th>Initiation interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Memory</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FP add</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>FP multiply</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>FP divide</td>
<td>24</td>
<td>24</td>
</tr>
</tbody>
</table>

structural hazards and WAW hazards

- structural hazards
  - divide unit
  - WB stage

- WAW hazards

```plaintext
... IF ID A1 A2 A3 A4 MEM WB
ADD F8, ... IF ID MEM WB
LD F8, ... IF ID MEM WB
```
Hazard Detection in the ID stage

- An instruction can only *issue* (proceed past the ID stage) when:
  - there are no structural hazards (divide unit is free, WB port will be free when needed)
  - no RAW data hazards
  - no WAW hazards with instructions in long pipes

MIPS R4000 Pipeline

- scalar, superpipelined
  - IF—first half of fetching of instruction; PC selection happens here as well as initiation of instruction cache access.
  - IS—second half of access to instruction cache.
  - RF—instruction decode and register fetch, hazard checking and also instruction cache hit detection.
  - EX—execution, which includes effective address calculation, ALU operation, and branch target computation and condition evaluation.
  - DF—data fetch, first half of access to data cache.
  - DS—second half of access to data cache.
  - TC—tag check, determine whether the data cache access hit.
  - WB—write back for loads and register-register operations.

R4000 Data Load Hazard

- Is there an integer arithmetic data hazard?

Two solutions (with slight variations)
  - imprecise interrupts (possibly with software fixes to make it look precise)
  - buffer results, forcing in-order WB’s

MULTD IF ID M1 M2 M3 M4 M5 M6 M7 ...
ADD D IF ID A1 A2 A3 A4 MEM WB
LD IF ID EX MEM WB

o-o-o completion and precise exceptions

MULTD IF ID M1 M2 M3 M4 M5 M6 M7 ...
ADD D IF ID A1 A2 A3 A4 MEM WB
LD IF ID EX MEM WB

- Two solutions (with slight variations)
  - imprecise interrupts (possibly with software fixes to make it look precise)
  - buffer results, forcing in-order WB’s
R4000 Data Load Hazard

lw R1,... IF IS RF EX DF DS TC WB
add R3, R1, R2 IF IS RF (stall) (stall) EX DF

- 2-cycle load delay

R4000 Branch Hazard

- predict not taken, branch delay slot
- not taken -> no penalty (unless branch likely or no delay slot instruction)
- taken -> 2 stall cycles if delay slot instruction used

MIPS R4000 Floating Point

- FP Adder, FP Multiplier, FP Divider
- Last step of FP Multiplier/Divider uses FP Adder HW
- 8 kinds of stages in FP units:

<table>
<thead>
<tr>
<th>Stage</th>
<th>Functional unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>FP adder</td>
<td>Mantissa ADD stage</td>
</tr>
<tr>
<td>D</td>
<td>FP divider</td>
<td>Divide pipeline stage</td>
</tr>
<tr>
<td>E</td>
<td>FP multiplier</td>
<td>Exception test stage</td>
</tr>
<tr>
<td>M</td>
<td>FP multiplier</td>
<td>First stage of multiplier</td>
</tr>
<tr>
<td>N</td>
<td>FP multiplier</td>
<td>Second stage of multiplier</td>
</tr>
<tr>
<td>R</td>
<td>FP adder</td>
<td>Rounding stage</td>
</tr>
<tr>
<td>S</td>
<td>FP adder</td>
<td>Operand shift stage</td>
</tr>
<tr>
<td>U</td>
<td>Unpack FP numbers</td>
<td></td>
</tr>
</tbody>
</table>

MIPS FP Pipe Stages

<table>
<thead>
<tr>
<th>FP Instr</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add, Subtract</td>
<td>U</td>
<td>S+A</td>
<td>A+R</td>
<td>R+S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiply</td>
<td>U</td>
<td>E+M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>N</td>
<td>N+A</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Square root</td>
<td>U</td>
<td>E</td>
<td>(A+R)^108</td>
<td>...</td>
<td>A</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Negate</td>
<td>U</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Absolute value</td>
<td>U</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FP compare</td>
<td>U</td>
<td>A</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Stages:

| M | First stage of multiplier |
| N | Second stage of multiplier |
| R | Rounding stage             |
| S | Operand shift stage        |
| U | Unpack FP numbers           |

CSE 240  Dean Tullsen  CSE 240  Dean Tullsen
R4000 Performance

Key Points

- Data Hazards can be significantly reduced by forwarding
- Branch hazards can be reduced by early computation of condition and target, branch delay slots, branch prediction
- Data hazard and branch hazard reduction require complex compiler support
- Exceptions are hard, precise exceptions are really hard
- variable-length instructions introduce structural hazards, WAW hazards, more RAW hazards