Write $X_1$ to $X_2$

Read $X_1$, $X_2$

What cycle of a pipelined CPU running this code?

What registers are being read and written in the

Review Problem 31
20% Slow

$\text{CPU} = \frac{8 \times 1 + 1.2 \times 2}{8 + 4} = 1.2$

CPI if all other instructions take 1 cycle, and branches are 20% of instructions?

Solution #1: Stall

Delay loading next instruction, load no-op instead
3 = 0.8 + 0.3

\[ CF: \ 8 \times 1 + 2(5 \times 1 + 5 \times 2) = 8 + 1 + 2 \times 1.5 \]

CPI if 50% of branches actually not taken, and branch frequency 20%?

Solution #2: Branch Prediction

Guess all branches not taken, squash if wrong
Solution #3: Branch Delay Slot

Compiler/Assembler fills the delay slot

Instruction after branch is the delay slot

Redefined branches: Instruction directly after branch always executed
Consider the following code:

Data Hazards
Design Register File Carefully

124

1. ORR
2. AND
3. SUB
4. ADD

What if reads see value after write during the same cycle?
Forwarding
have problems with hazards?

Do the unconditional branch instructions (B, BR)

Review Problem 33
Compare sources of current instruction to destinations of previous 2.
Remember destination register for operation.
Requires values from last two ALU operations.

Forwarding (cont.)
Data Hazards on Loads

- LDRR X0, [X31, 0]
- SUB X3, X0, X4
- AND X5, X0, X6
- ORR X7, X0, X8
- EOR X9, X0, X10

Clock:
- Cycle 1
- Cycle 2
- Cycle 3
- Cycle 4
- Cycle 5
- Cycle 6
- Cycle 7
- Cycle 8
- Cycle 9

 fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR

For: Fetch
- Mem
- REG/DEC
- EXEC
- WR
Fill delay slot or insert no-op.

Force compiler to not allow register reads within a cycle of load.

Use same forwarding hardware & register file for hazards 2+ cycles later.

Solution:

Data Hazards on Loads (cont.)
Pipeline cycle: CPI = 1.0, cycle time = 4.5 ns.

Pipelined cycle time = 1 ns.

<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>CPI</th>
<th>Cycles * Frequency</th>
<th>Type Cycles</th>
<th>Type Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch</td>
<td>1.5</td>
<td>20%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Store</td>
<td>1.0</td>
<td>10%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Load</td>
<td>0.3</td>
<td>20%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALU</td>
<td>0.5</td>
<td>50%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Delay for 1 M instr: (1.2 * CPI + 0.5 ns)

Delay for 1 M instr: 4.5 ns.

129
Pipelined CPU Summary

Concerns:
- Speed up branches
- Load delay
- Forwarding
- Control hazards
- Data hazards

Structural hazards: All in 5 cycles
- Same order
- Speed up branches
- Load delay
- Forwarding
- Control hazards
- Data hazards