Review single cycle processor

Insight: we can execute the next instruction after finishing an inst in a processing unit => pipelining
Each instruction is executed in one stage, and then the next instruction is executed in the stage.
Limits to pipelining

° Can we execute instructions consecutively all the time?

No! **Hazards** prevent next instruction from executing during its designated clock cycle

° Types of hazards

  • **Structural hazards**: Two instructions that try to access one HW component at the same time.

  • **Control hazards**: Instructions after a branch can’t be executed until the branch is determined.

  • **Data hazards**: An instruction depends on result of prior instruction still in the pipeline.
Structural Hazard #1: Single Memory

° When an inst accesses data memory (Mem) and the other accesses instruction memory (IF), then the latter stalled.

• By having instruction cache and data cache, we can allow inst memory and data memory can be accessed at the same cycle.
Structural Hazard #2: Registers

- One inst tries to read a register (ID), another tries to write data onto the register (WB). Registers can’t be read and written at the same time!

- Solution: Use the fact that register access is very fast: takes less than half the time of ALU stage

- always Write to Registers during first half of each clock cycle

- always Read from Registers during second half of each clock cycle

- Result: can perform Read and Write during same clock cycle
Control Hazard: Branching (1/3)

° The decision-making done for a branch instruction in ALU stage!

• This means two instructions after a branch can’t be executed.
We can reduce stalls by making decision earlier – one cycle stall.
Control Hazard: Branching (3/3)

Further optimization

• Define a branch so that the single instruction immediately following the branch gets executed whether the branch is taken or not (called the branch-delay slot)

• In many cases, we can find an instruction preceding the branch which can be placed in the branch-delay slot without affecting flow of the program

• Re-ordering instructions can be done by compiler.
What is a data hazard?

- An instruction reads the source registers in ID stage.
- An instruction writes its data at WB stage.
- If the register values are to be written by any of its preceding instructions and those instructions haven’t reached WB stage, then data values are not available.
Data Hazards (2/4)

° Insights
  • For ALU instructions (ADD, SHIFT, etc), the output is available after EX stage.
  • Even though an instruction reads its source register in ID stage, it is ALU stage where the register values are really used.

° Forwarding
  • A data value which is not written to the register file can be forwarded as an input in ALU stage of the next instruction.
If an instruction right after a load depends on the load, the data value can’t be forwarded.

\[
\text{lw } \text{\$t0,0($t1)}
\]

\[
\text{sub } \text{\$t3,\$t0,\$t2}
\]

• Must stall instruction dependent on load, then forward (more hardware)
Data Hazard: Loads (3/4)

- Instruction slot after a load is called "load delay slot"

- If that instruction uses the result of the load, then the hardware interlock will stall it for one cycle.

- If the compiler puts an unrelated instruction in that slot, then no stall

- Letting the hardware stall the instruction in the delay slot is equivalent to putting a nop in the slot (except the latter uses more code space)
Data Hazard Solution: Forwarding
Example

° Consider the following 5 stage pipeline processor. And assume there are no structural hazards.
Example

° In the following instructions, how many cycles need to be smalled without forwarding?

shli r3, r2, 3
addu r4, r3, r1

° How about with forwarding?
Example

° In the following instructions, how many cycles need to be smalled without forwarding?

\[ \text{lw} \quad \text{r4, 4(r4)} \]
\[ \text{mulu} \quad \text{r7, r3, r4} \]

° How about with forwarding?
Example

° The decision for branch instruction is made in ID stage, thus one instruction right after a branch is executed.

° Suppose the branch decision is made in IF stage using some prediction logic. Then how can we define the meaning of a branch?
Example

In the following instructions:

```
0  addi  r5, r0, 50
4  lw    r1, 16(r19)
8  lw    r2, 32(r19)
12 loop:  shli  r3, r2, 3
16  addu  r4, r3, r1
20  lw    r3, 0(r4)
24  lw    r4, 4(r4)
28  mulu  r7, r3, r4
32  subu  r5, r5, r7
36  bnez  r2, loop
40  addi  r2, r2, -1
```
Example

Fill in the table below with at least 5 more entries. Find at least one instruction that will stall for a cycle or more. “Source Inst” is the instruction that is writing the register.

<table>
<thead>
<tr>
<th>Source Inst</th>
<th>Dest Inst</th>
<th>Register</th>
<th>Forwarding Required?</th>
<th>Stall even if Forward?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Extra
Example

° In the following instructions, how many cycles need to be smalled without forwarding?

\[
\begin{align*}
\text{shli} & \quad r3, r2, 3 \\
\text{addu} & \quad r4, r3, r1 \\
\end{align*}
\]

Two cycles

° How about with forwarding?

No stalls
Example

° In the following instructions, how many cycles need to be smalled without forwarding?

\[
\text{lw r4, 4(r4)} \\
\text{mulu r7, r3, r4}
\]

Two cycles

° How about with forwarding?

One cycle
Example

° The decision for branch instruction is made in ID stage, thus one instruction right after a branch is executed.

° Suppose the branch decision is made in IF stage using some prediction logic. Then how can we define the meaning of a branch?

We don’t need any branch delay slot because branch direction can be determined in the first cycle (IF).
## Example

<table>
<thead>
<tr>
<th>Source Inst</th>
<th>Dest Inst</th>
<th>Register</th>
<th>Forwarding Required?</th>
<th>Stall even if Forward?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>32</td>
<td>R5</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>R1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>R2</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>36</td>
<td>R2</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>40</td>
<td>R2</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>12</td>
<td>16</td>
<td>R3</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>16</td>
<td>20</td>
<td>R4</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>16</td>
<td>24</td>
<td>R3</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>20</td>
<td>28</td>
<td>R4</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>24</td>
<td>28</td>
<td>R4</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>28</td>
<td>32</td>
<td>R7</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>