Pipelining a Datapath

Tom Kelliher, CS 220

Nov. 30, 2009

1 Administrivia

Announcements

Assignment

Read 5.1–5.2.

From Last Time

Overview of pipelining.

Outline


2. Simple example: a single \texttt{lw}.

Coming Up

Introduction to caches.
2 Pipelining

A pipelined datapath:

Consider four instructions: R-mode, a branch, LW, SW.

Observations:

1. Not a true pipeline: feedback.
2. How do we re-design control?

2.1 Hazards

1. Structural hazards.
   Example: unified L1 cache/memory.
2. Control hazards. Consider the following example:
Program execution order (in instructions)

Solutions:

(a) Stall.

(b) Predict.

Static prediction. Truly static. Compile-time determined.

Dynamic prediction. Branch history tables. One-, two-bit counters.

(c) Delayed branch.

Assumes you know branch outcome early.

Code scheduling:
a. From before
add $s1, $s2, $s3
if $s2 = 0 then
Delay slot
Becomes
if $s2 = 0 then
add $s1, $s2, $s3
b. From target
sub $t4, $t5, $t6
... 
add $s1, $s2, $s3
if $s1 = 0 then
Delay slot
Becomes
add $s1, $s2, $s3
if $s1 = 0 then
sub $t4, $t5, $t6
c. From fall through
add $s1, $s2, $s3
if $s1 = 0 then
sub $t4, $t5, $t6

Consideration: deeper pipelines.

3. Data hazards.

Data not available when needed.

ALU example:

<table>
<thead>
<tr>
<th>Time (in clock cycles)</th>
<th>CC 1</th>
<th>CC 2</th>
<th>CC 3</th>
<th>CC 4</th>
<th>CC 5</th>
<th>CC 6</th>
<th>CC 7</th>
<th>CC 8</th>
<th>CC 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value of register $s2:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10/−20</td>
<td>−20</td>
<td>−20</td>
<td>−20</td>
<td>−20</td>
<td>−20</td>
</tr>
</tbody>
</table>

Program execution order (in instructions)

- sub $s2, $1, $3
- and $12, $2, $5
- or $13, $6, $2
- add $14, $2, $2
- sw $15, 100($2)
Fixed by forwarding.

Memory example:

How can this be fixed?

2.2 Simple Example

Let’s follow a lw. What’s going on during each clock cycle?
lw
Execution

Instruction memory

Address

4

Add

Write data

Read register 1

Read register 2

Write register

Read data 1

Read data 2

Read data

Add result

Shift left 2

ALU

Write register

Write data

Address

Memory

Data

Write data

lw

IF/ID

EX/MEM

MEM/WB

PC

Add

Read data

16

Sign extend

32
Figure 06.15