Pipeline Data Hazards

Warning, warning, warning!

Dealing With Data Hazards

- In Software
- In Hardware

Data Hazards are caused by *instruction dependences*. For example, the add is data-dependent on the subtract:

```
subi $5, $4, #45
add   $8, $5, $2
```

Dealing with Data Hazards in Software

How Many No-ops?

```
sub $2, $1, $3
and $4, $2, $5 or $8, $2, $6
add $9, $4, $2
slt $1, $6, $7
```
Are No-ops Really Necessary?

sub $2, $1, $3
and $4, $2, $5
or $8, $3, $6
add $9, $2, $8
slt $1, $6, $7

Pipeline Stalls

sub $2, $1, $3
and $12, $2, $5
or $13, $6, $2
add $14, $2, $2
sw $15, 100($2)

Pipeline Stalls

• To insure proper pipeline execution in light of register dependences, we must:
  – the hazard
  – the pipeline
Knowing When to Stall

- 6 types of data hazards
  - two reg reads * 3 reg writes

The Pipeline

- What comparisons tell us when to stall?

Stalling the Pipeline

- Once we detect a hazard, then we have to be able to stall the pipeline (insert a bubble).
- Stalling the pipeline is accomplished by
  - preventing the and stages from making progress
    - the ID stage because it cannot proceed until the dependent instruction completes
    - the IF stage because we do not want to lose any instructions.
  - essentially, inserting "nops"
    - set all control signals propagating to EX/MEM/WB to _______

CSE 141

Dean Tullsen
Reducing Data Hazards Through Forwarding

How else can we deal with data hazards?

Reducing Data Hazards Through Forwarding

• The Previous Data Path handles two types of data hazards
  – hazard
  – hazard
• We assume the register file handles the third (hazard)
  – if the register file is asked to read and write the same register in the
    same cycle, we assume that the reg file allows the write data to be
    forwarded to the output

EX Hazard:
if (EX/MEM.RegWrite
    and (EX/MEM.RegisterRd != 0)
    and (EX/MEM.RegisterRd = ID/EX.RegisterRs)) ForwardA = 10
if (EX/MEM.RegWrite
    and (EX/MEM.RegisterRd != 0)
    and (EX/MEM.RegisterRd = ID/EX.RegisterRt)) ForwardB = 10
(similar for the MEM stage)
Eliminating Data Hazards via Forwarding

```
IM  Reg  DM  Reg
CC1  CC2  CC3  CC4  CC5  CC6  CC7  CC8
sub $2, $1, $3
and $6, $2, $5
or $13, $6, $2
add $14, $2, $2
sw $15, 100($2)
```

Forwarding in Action
Eliminating Data Hazards via Forwarding??

lw $2, 10($1)
and $12, $2, $5
or $13, $6, $2
add $14, $2, $2
sw $15, 100($2)

Eliminating Data Hazards via Forwarding and stalling

lw $2, 10($1)
and $12, $2, $5
or $13, $6, $2
add $14, $2, $2
sw $15, 100($2)

Try this one...

Show stalls and forwarding for this code

add $3, $2, $1
lw $4, 100($3)
and $6, $4, $3
sub $7, $6, $2
add $9, $3, $6
Datapath with Hazard-Detection

if (ID/EX.MemRead and ((ID/EX.RegisterRt = IF/ID.RegisterRs) or (ID/EX.RegisterRt = IF/ID.RegisterRt)))
then stall the pipeline

Hazard Detection

Data Hazard Key Points

- Pipelining provides high throughput, but does not handle data dependences easily.
- Data dependences cause data hazards.
- Data hazards can be solved by:
  - software (nops)
  - hardware stalling
  - hardware forwarding
- Our processor, and indeed all modern processors, use a combination of forwarding and stalling.
- ET = IC * CPI * CT