Handling Data Hazards

Tom Kelliher, CS 240

Apr. 22, 2002

1 Administrivia

Announcements

Assignment

Read 6.8–9.

From Last Time

Pipeline control exercise.

Outline

1. Register-To-Register hazards.
2. Memory-to-Register hazards.

Coming Up

Superscalar execution.

2 Register-To-Register Hazards

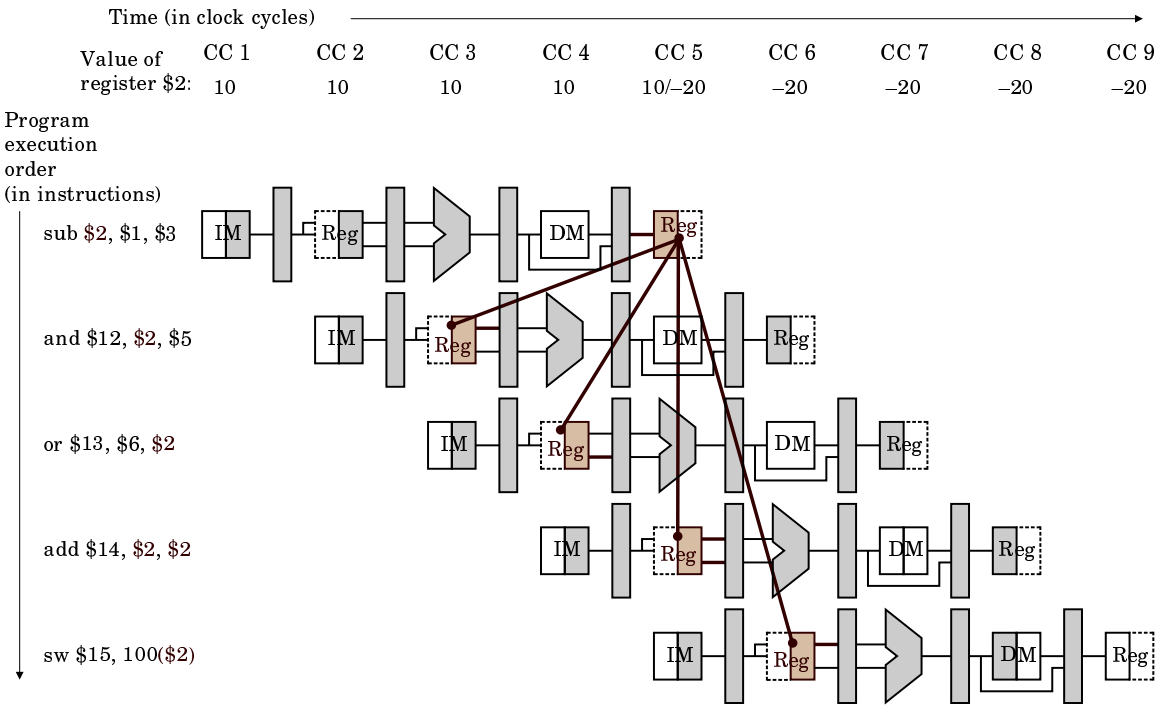
Consider the code segment:

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

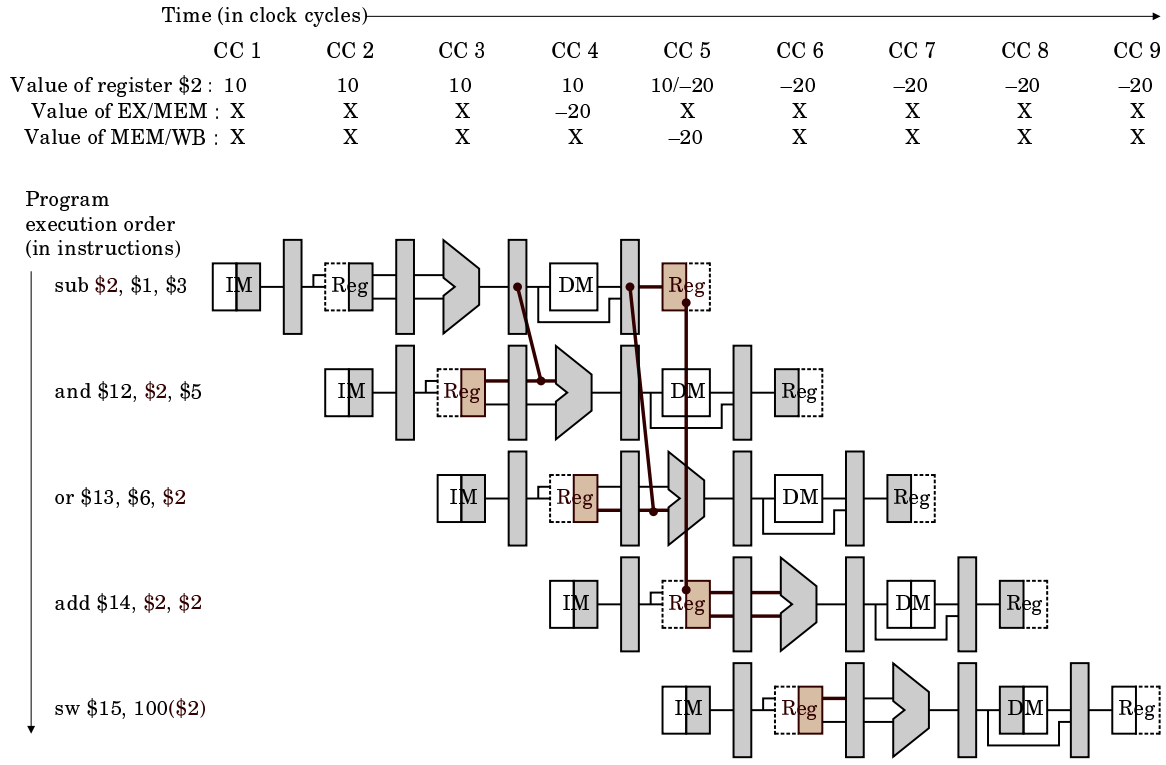
What are the dependencies?

What will happen during execution?

Dependencies must be resolved for correct execution.



What can we do to fix this? What are our options? (Stall, forwarding hardware, compiler) Cost, benefits?



Dependency detection:

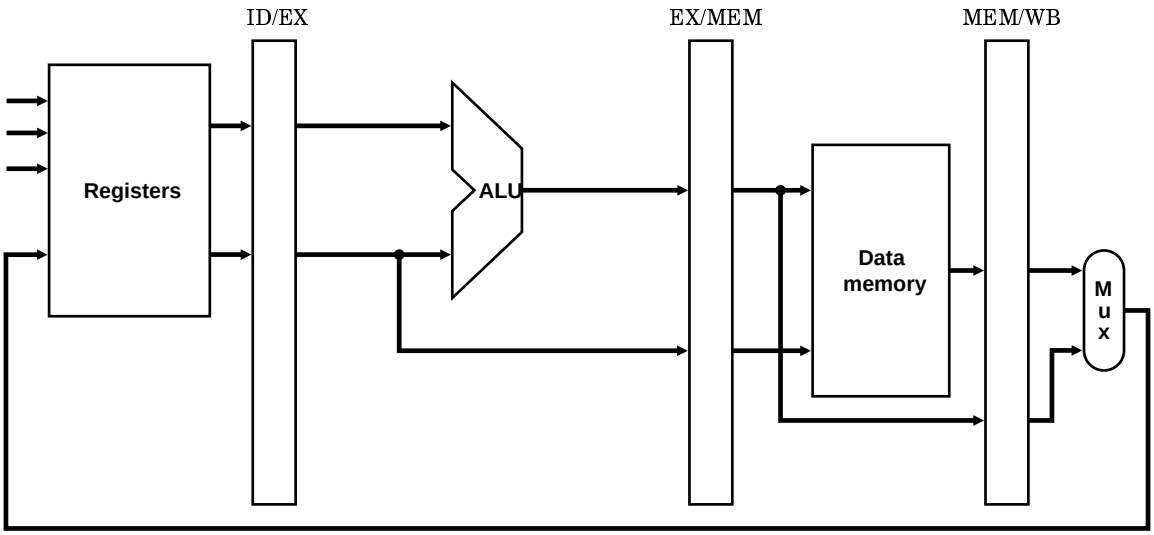
1. What do we need to compare?
2. Are there any "special conditions?" (RegWrite)
3. What about something like:

```
add $0, $1, $2
or $10, $0, $0
```

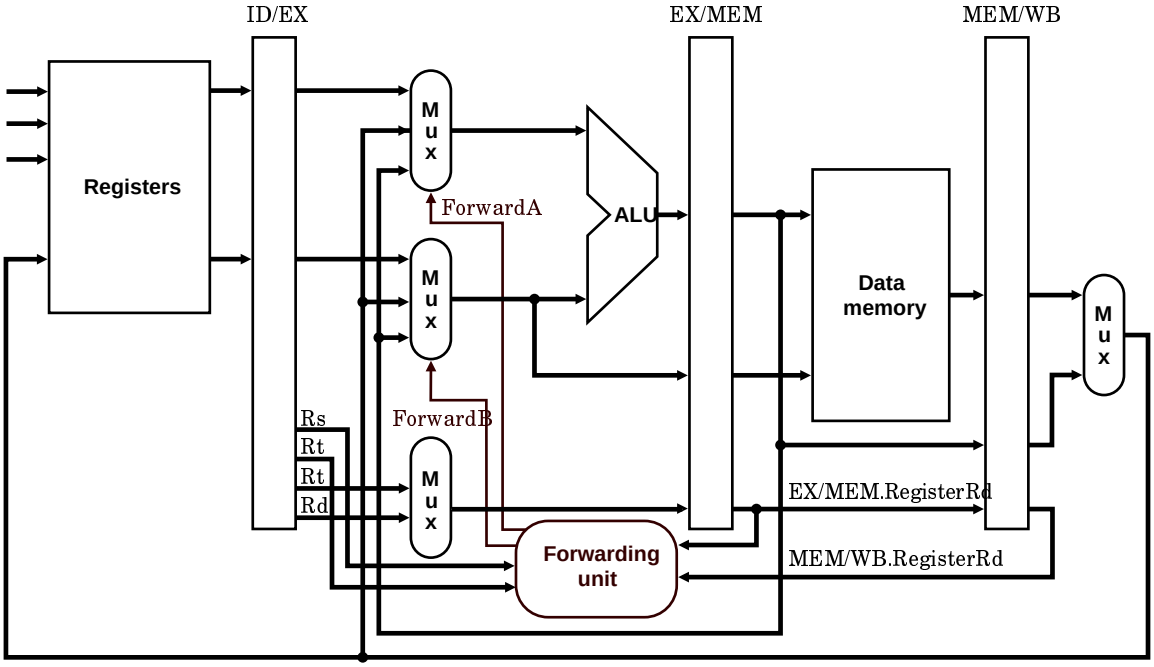
4. What about a conflict between the pipeline source:

```
add $1, $1, $2
add $1, $1, $3
add $1, $1, $4
```

5. What's a forwarding equation look like?



a. No forwarding



b. With forwarding

How expensive is this solution?

Is this the optimal position for the forwarding unit? What about the CPU's critical path?

Consider this example:

```
add $7, $20, $21
```

```
add $8, $22, $23
add $6, $7, $8
add $6, $6, $6
```

Show the forwarding.

3 Memory-To-Register Hazards

Consider the following code segment:

```
lw $2, 20($1)
and $4, $2, $5
or $8, $2, $5
add $9, $4, $2
slt $1, $6, $7
```

What are the dependencies?

What will happen during execution?

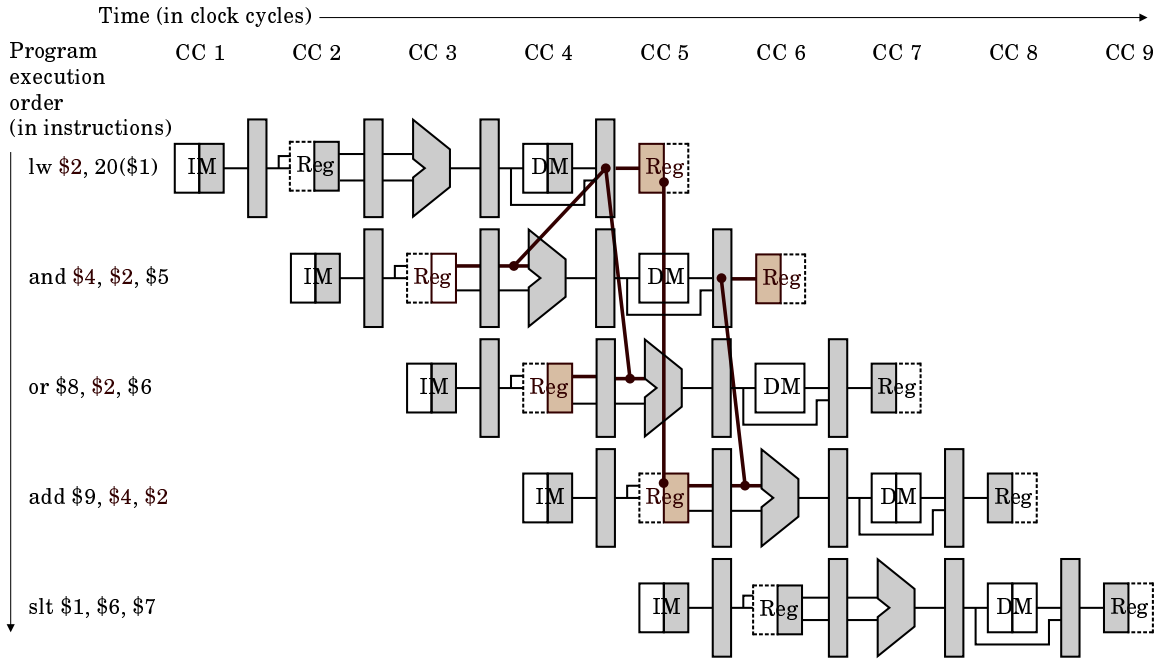
Why can't we forward? So, what do we do?

What about this situation:

```
lw $2, 0($10)
sw $2, 0($11)
add $10, $10, $3      # $3 = 4
add $11, $11, $3
```

Do we need to stall the `sw`?

What can the compiler do to help out?



1. What do we need to look at to stall the pipe? (ID/EX.MemRead. Why there?)

2. How do we actually stall?

   Set all control bits in ID stage to 0. Hold the PC and IF/ID reg.

   Why do we hold the IF/ID reg? (PC has already advanced.)