Poll Q: How many stalls?

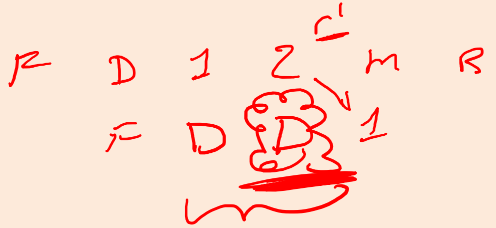
type (no enter) into Zoom chat

- Suppose EX is the longest (in time) pipeline stage
- To reduce CT, we split it in half. Given the following (new) pipeline:

  IF ID EX1 EX2 M WB

  Assume the input data must be available at the start of EX1 and the output is available after EX2
- **How many hardware stalls** would be required in the following code (assuming hardware forwarding wherever possible)?

```
add r1, r2, r3
add r4, r1, r3
```



CC BY-NC-ND Pat Pannuto – Many slides adapted from Dean Tullsen

Poll Q: How many stalls?

type (no enter) into Zoom chat

- Suppose EX is the longest (in time) pipeline stage
- To reduce CT, we split it in half. Given the following (new) pipeline:

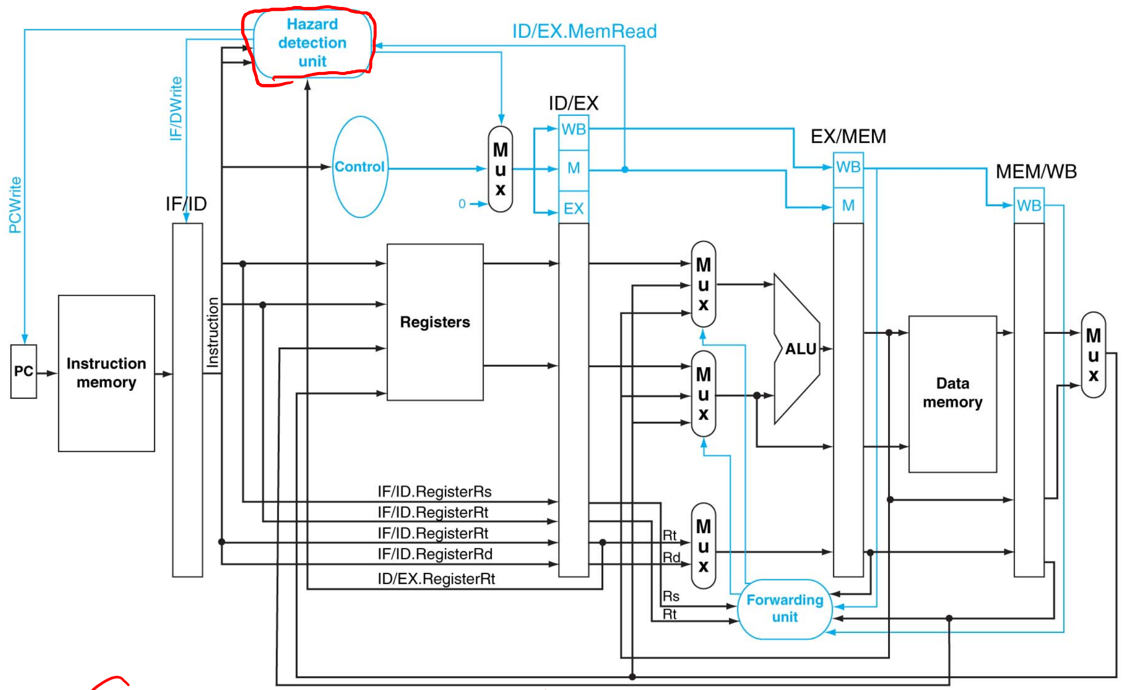
  IF ID EX1 EX2 M WB

  Assume the input data must be available at the start of EX1 and the output is available after EX2
- How many hardware stalls would be required in the following code (assuming hardware forwarding wherever possible)?

```
lw  r1, 0(r3)
add r2, r1, r3
```

CC BY-NC-ND Pat Pannuto – Many slides adapted from Dean Tullsen

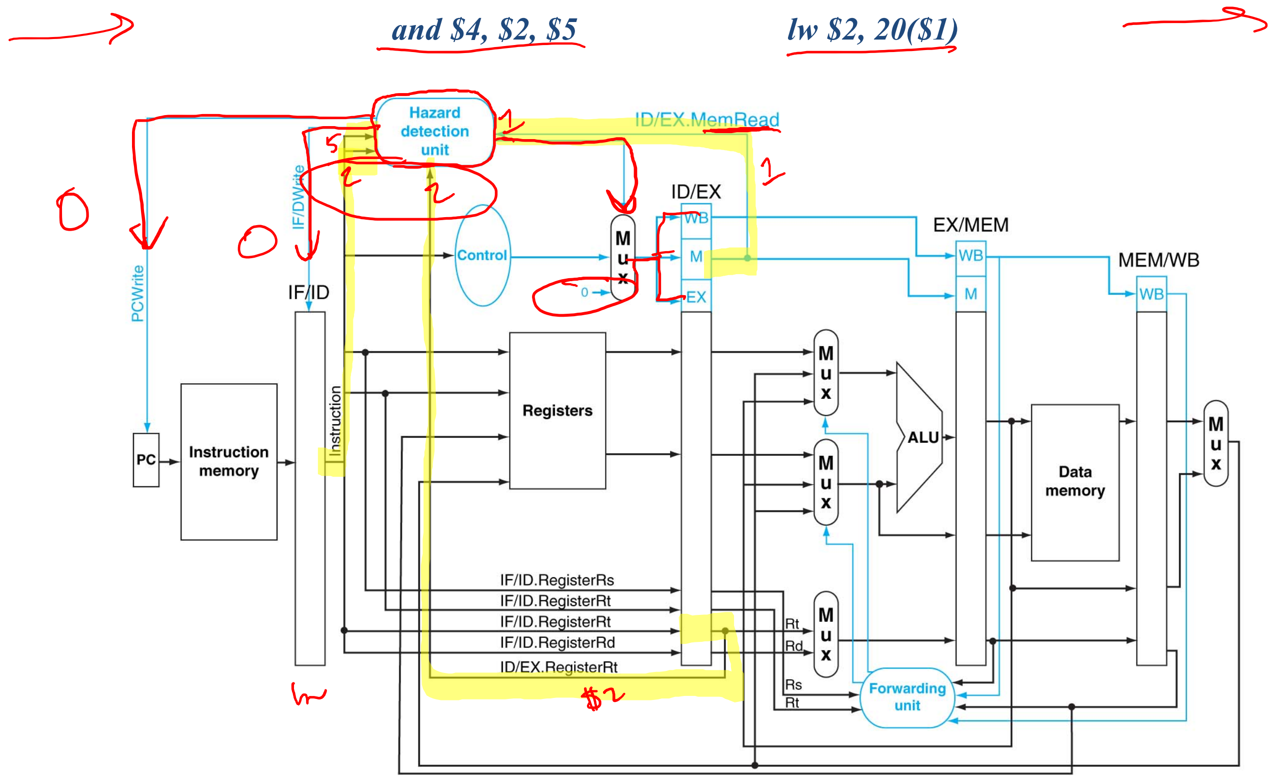
Datapath with Hazard-Detection



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

Hazard Detection

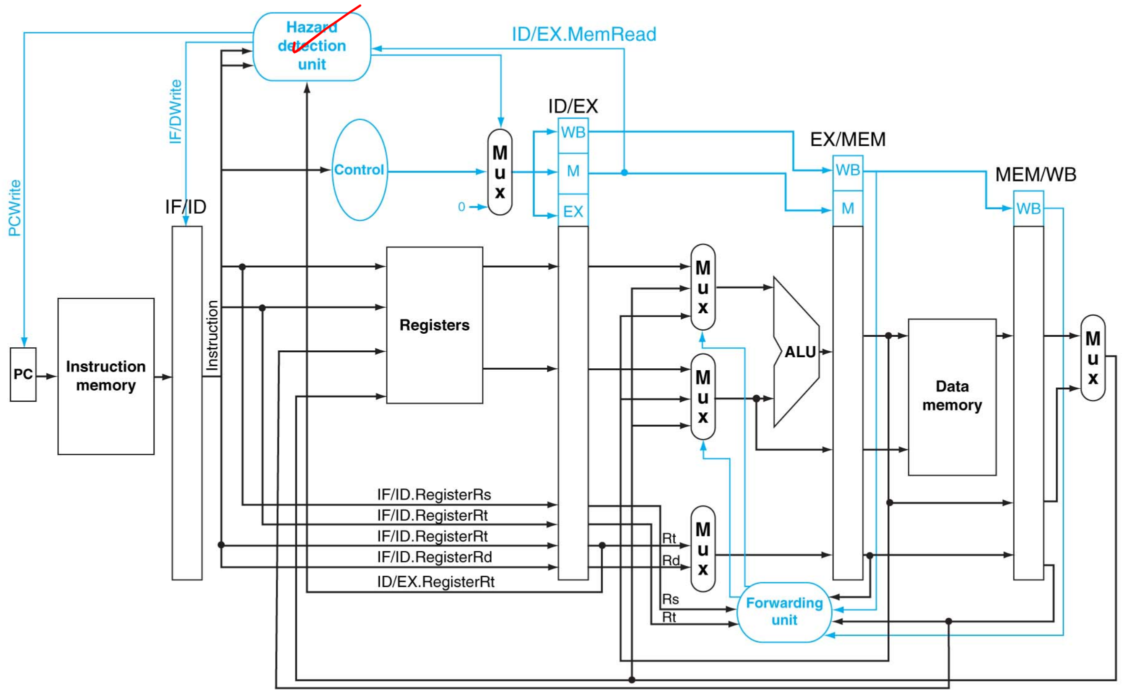
and \$4, \$2, \$5 *lw \$2, 20(\$1)*



Hazard Detection

~~and $4 2 5~~

and $4, $2, $5 *nop (bubble)* *lw $2, 20($1)*



What other hazards might we have to watch out for?

- Data hazards are when the result of one computation is used in a later computation
- Is there other re-use?

add r1, r2, r3

sub r4, r1, r7

beq

sw

CC BY-NC-ND Pat Pannuto – Many slides adapted from Dean Tullsen

Control Dependence

- Just as an instruction will be dependent on other instructions to provide its operands (data dependence), it will also be dependent on other instructions to determine whether it gets executed or not (control dependence, aka, branch dependence).
- Control dependences are particularly critical with conditional branches.

$6 - $7 == 0?

```
add $5, $3, $2                        somewhere: or $10, $5, $2
sub $6, $5, $2                                   add $12, $11, $9
beq $6, $7, somewhere   F D X                    ...
and $9, $6, $1            F D X
...
```

CC BY-NC-ND Pat Pannuto – Many slides adapted from Dean Tullsen

Branch Hazards

- Branch dependences can result in branch hazards (when they are too close to be handled correctly in the pipeline)
  - (sound familiar?)

CC BY-NC-ND Pat Pannuto – Many slides adapted from Dean Tullsen

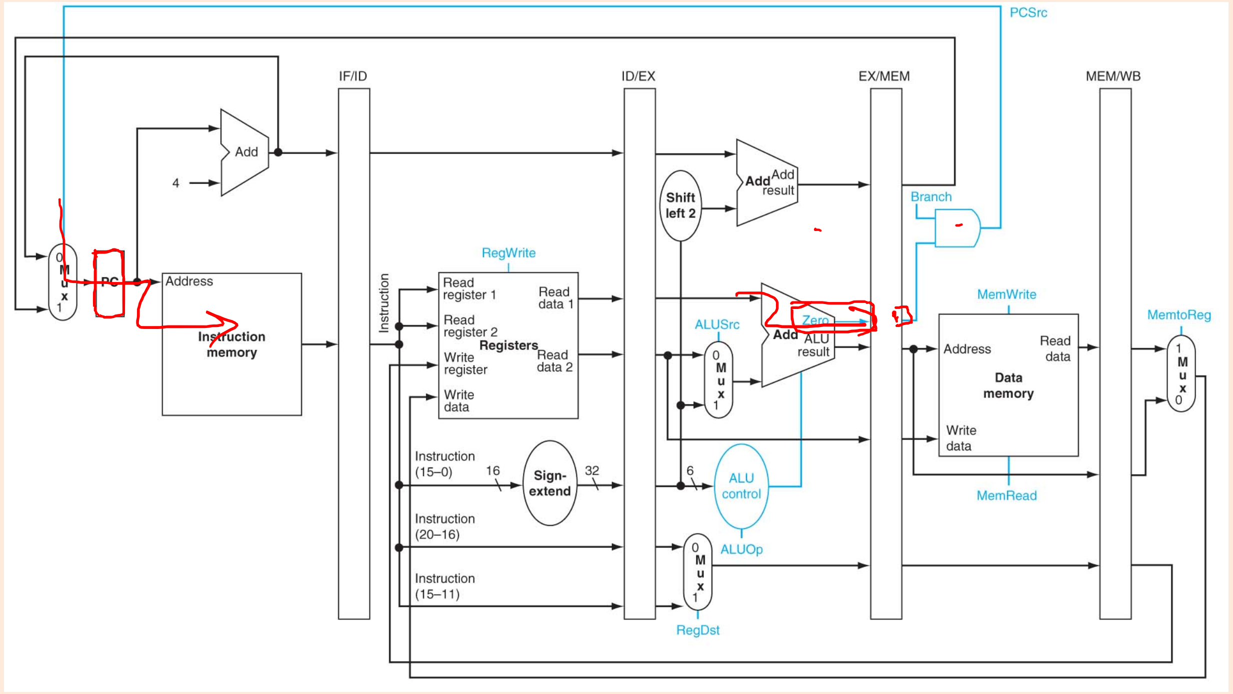
beq 1 F 2 D 3 X 4 M 5 WB

Stalling the pipeline

Given our current pipeline, let's assume we stall until we know the branch outcome (i.e., until the PC is known to be correct).

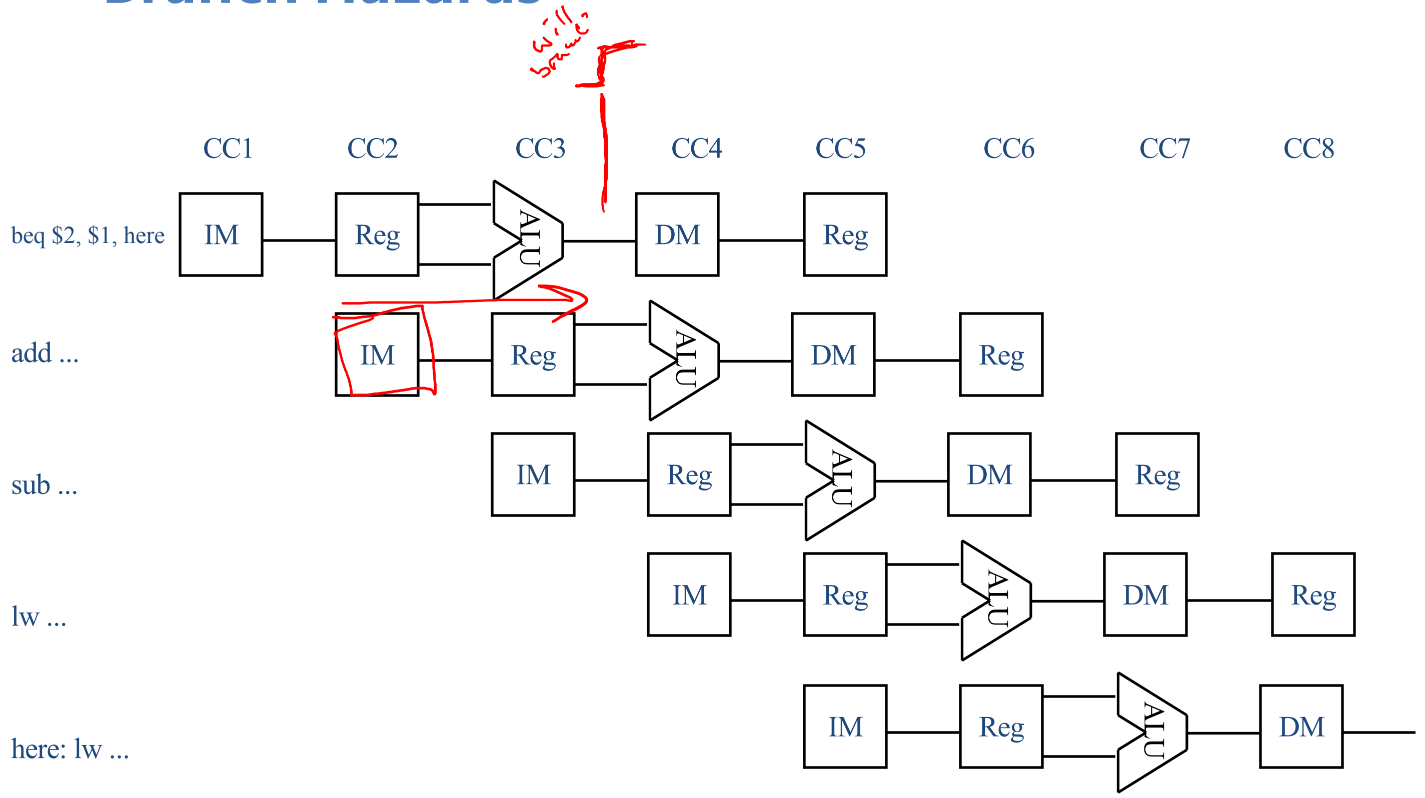
How many cycles will we lose per branch?

	cycles
A	0
B	1
C	2
D	3
E	4



CC BY-NC-ND Pat Pannuto – Many slides adapted from Dean Tullsen

Branch Hazards



CC BY-NC-ND Pat Pannuto – Many slides adapted from Dean Tullsen

Dealing With Branch Hazards

- Ideas??

CC BY-NC-ND Pat Pannuto – Many slides adapted from Dean Tullsen

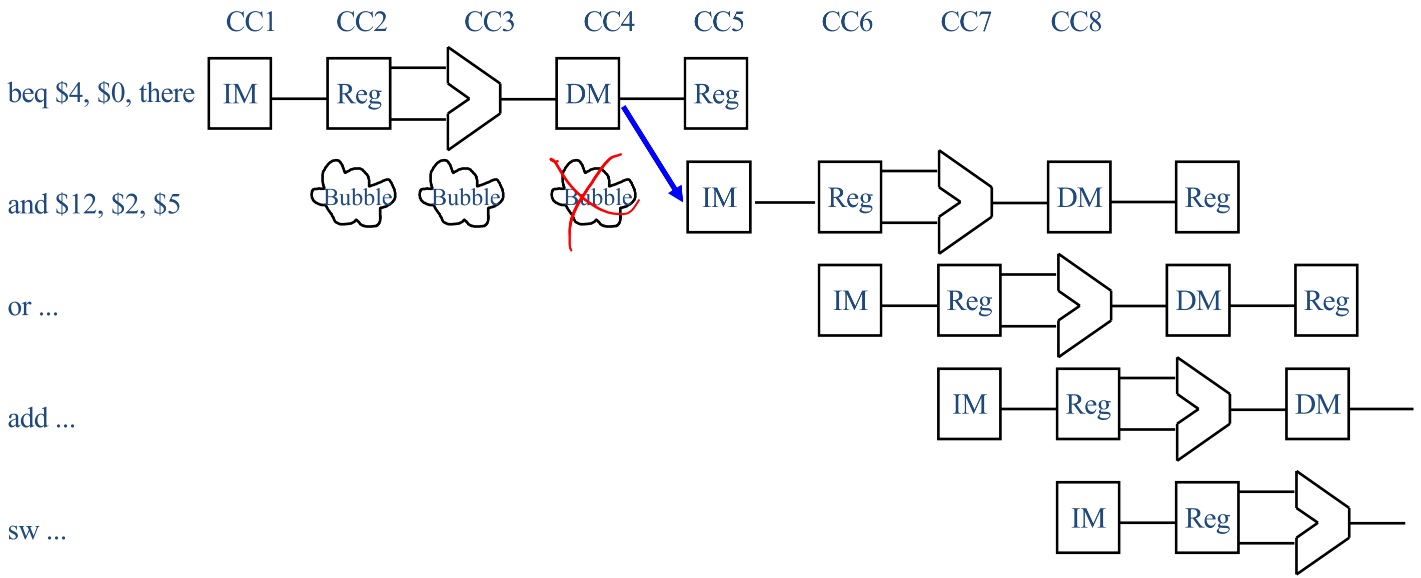
Dealing With Branch Hazards

- Hardware
  - ✗ stall until you know which direction
  - reduce hazard through earlier computation of branch direction
  - guess which direction
    - assume not taken (easiest)
    - more educated guess based on history
      - (requires that you know it is a branch before it is even decoded!)

Dealing With Branch Hazards

- Hardware
  - stall until you know which direction
  - reduce hazard through earlier computation of branch direction
  - guess which direction
    - assume not taken (easiest)
    - more educated guess based on history
      - (requires that you know it is a branch before it is even decoded!)
- Hardware/Software
  - nops
  - instructions that get executed either way (delayed branch).

Stalling for Branch Hazards



CC BY-NC-ND Pat Pannuto – Many slides adapted from Dean Tullsen

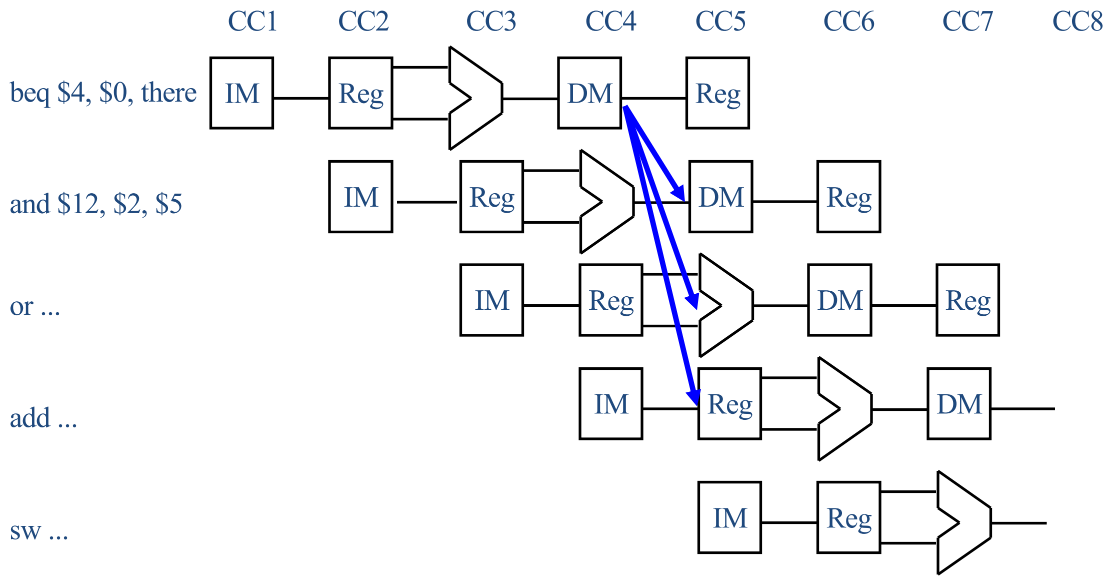
Stalling for Branch Hazards

- Seems wasteful, particularly when the branch isn’t taken.
- Makes all branches cost 4 cycles.

CC BY-NC-ND Pat Pannuto – Many slides adapted from Dean Tullsen

Assume Branch *Not Taken*

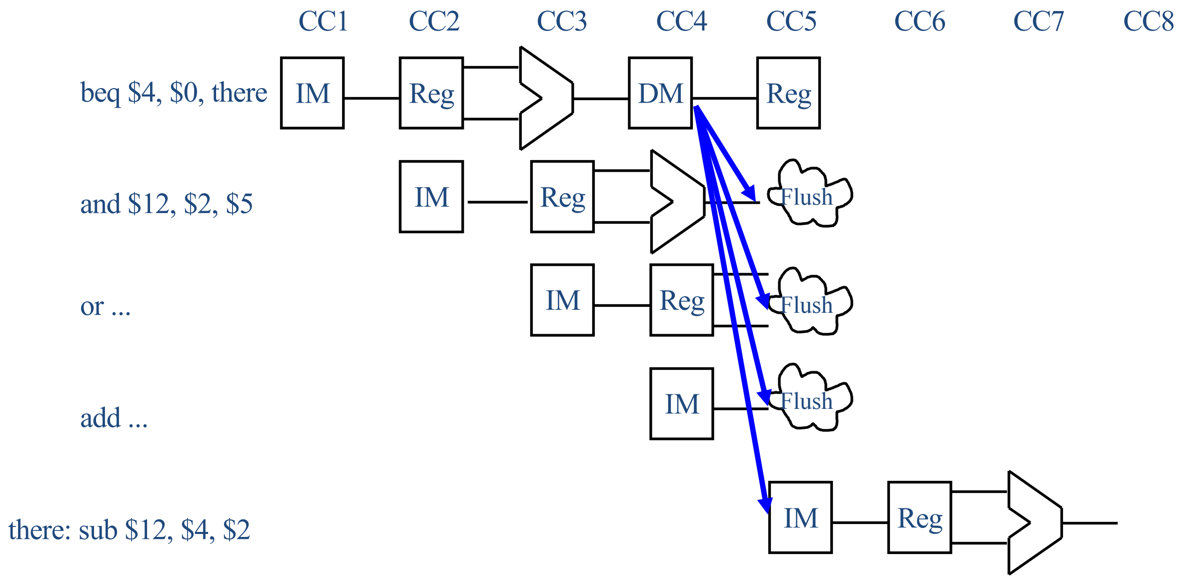
- works pretty well when you're right!



CC BY-NC-ND Pat Pannuto – Many slides adapted from Dean Tullsen

Assume Branch *Not Taken*

- *same performance as stalling* when you're wrong

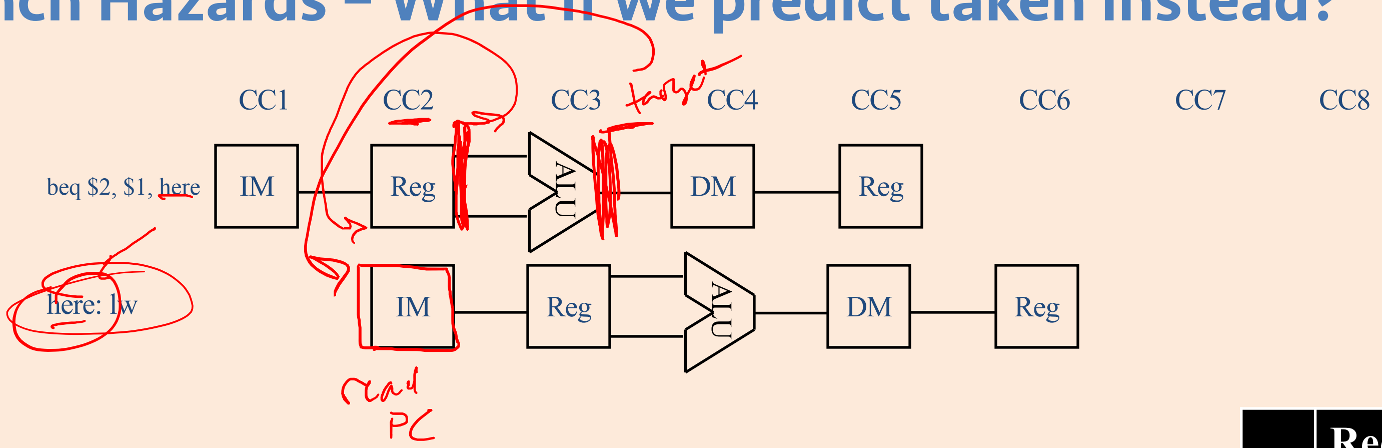


Assume Branch *Not Taken*

- Performance depends on percentage of time you guess right
- Flushing an instruction means to prevent it from changing any permanent state (registers, memory, PC)
  - sounds a lot like a bubble...
  - But notice that we need to be able to insert those bubbles *later* in the pipeline

CC BY-NC-ND Pat Pannuto – Many slides adapted from Dean Tullsen

Branch Hazards – What if we predict taken instead?



Required information to predict Taken:

1. Whether an instruction is a branch (before decode)
2. The target of the branch
3. ~~The outcome of the branch condition~~

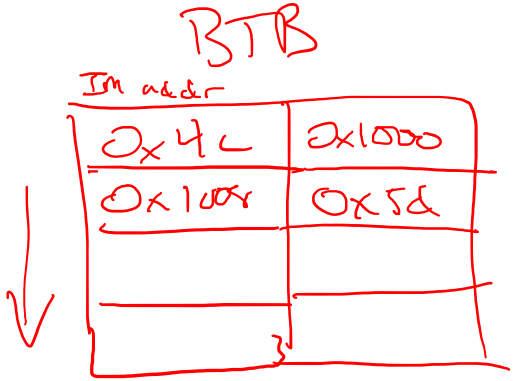
	Required knowledge
A	2 , 3
B	1 , 2 , 3
[✓ C]	1 , 2
D	2
E	None of the above

CC BY-NC-ND Pat Pannuto – Many slides adapted from Dean Tullsen

Branch Target Buffer

aka, how to know it's a branch before you know it's a branch

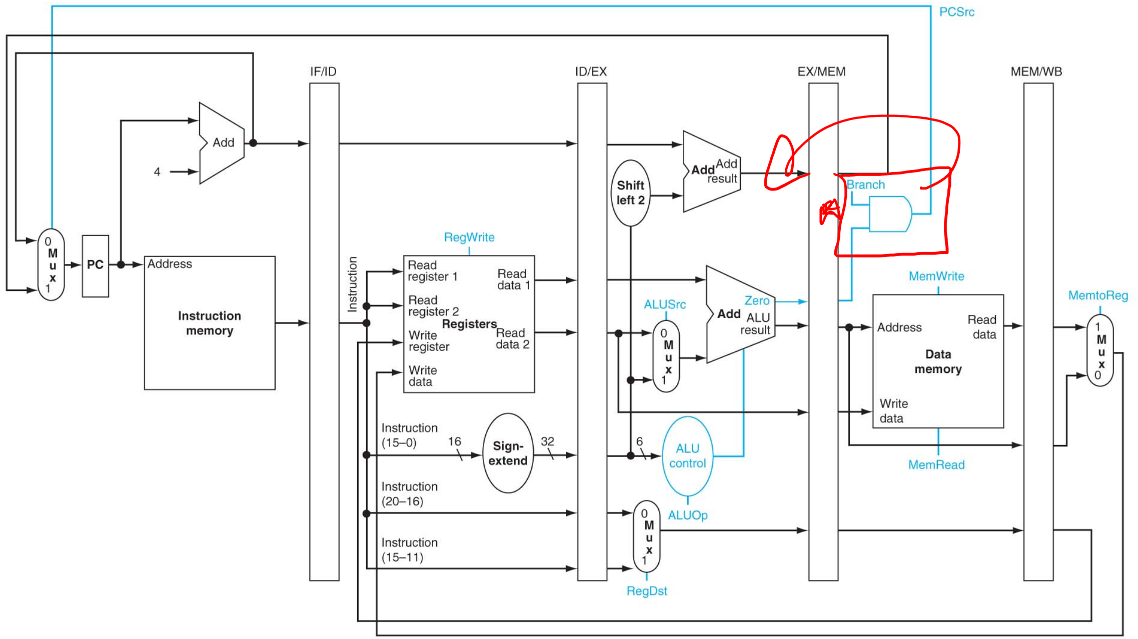
- Keeps track of the PCs of recently seen branches and their targets.
- Consult during Fetch (in parallel with Instruction Memory read) to determine:
  - Is this a branch?
  - If so, what is the target



Reducing the Branch Delay

CC BY-NC-ND Pat Pannuto – Many slides adapted from Dean Tullsen

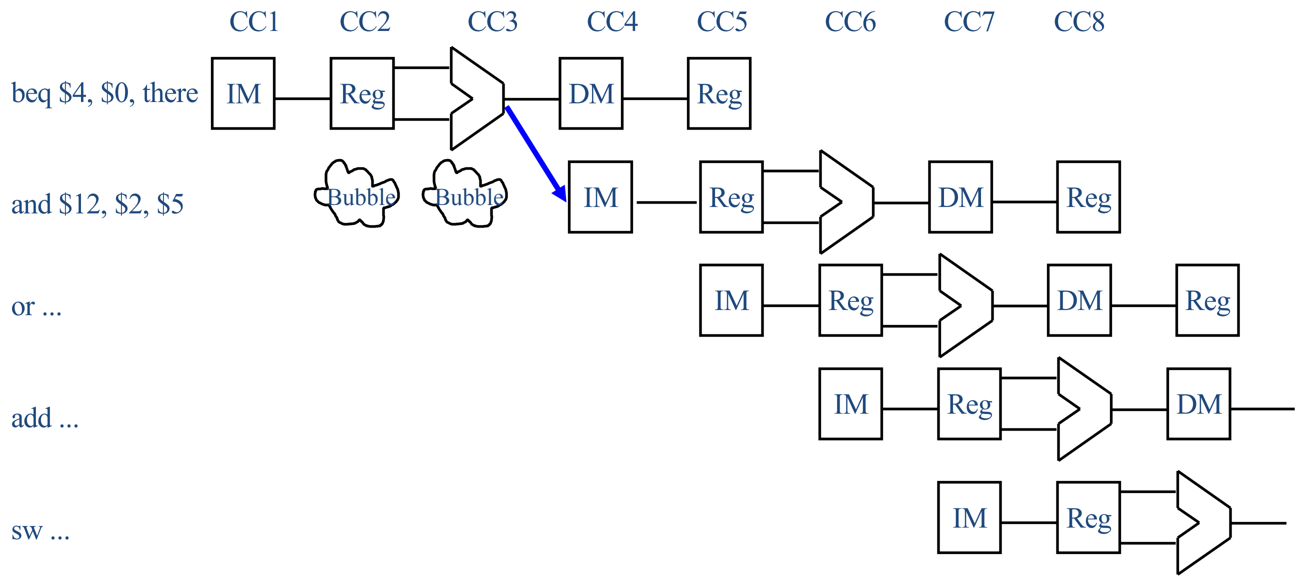
Reducing the Branch Delay



•can easily get to 2-cycle stall

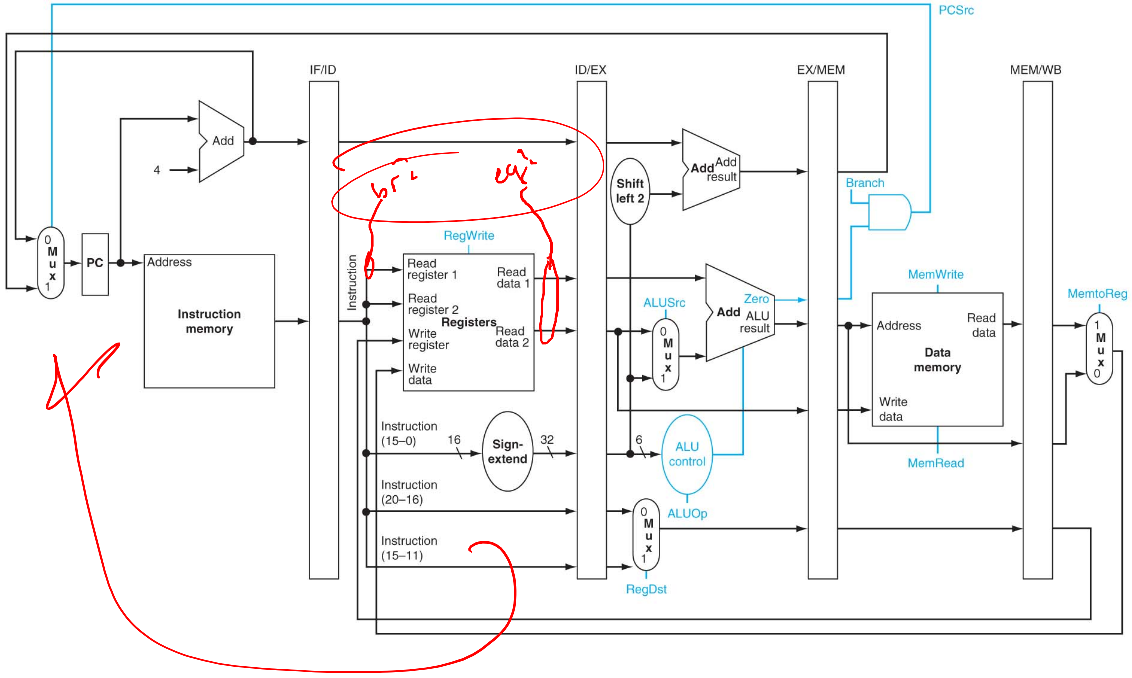
CC BY-NC-ND Pat Pannuto – Many slides adapted from Dean Tullsen

Stalling for Branch Hazards



Reducing the Branch Delay

Bq F D X m W
F
F?



PART I: 28A

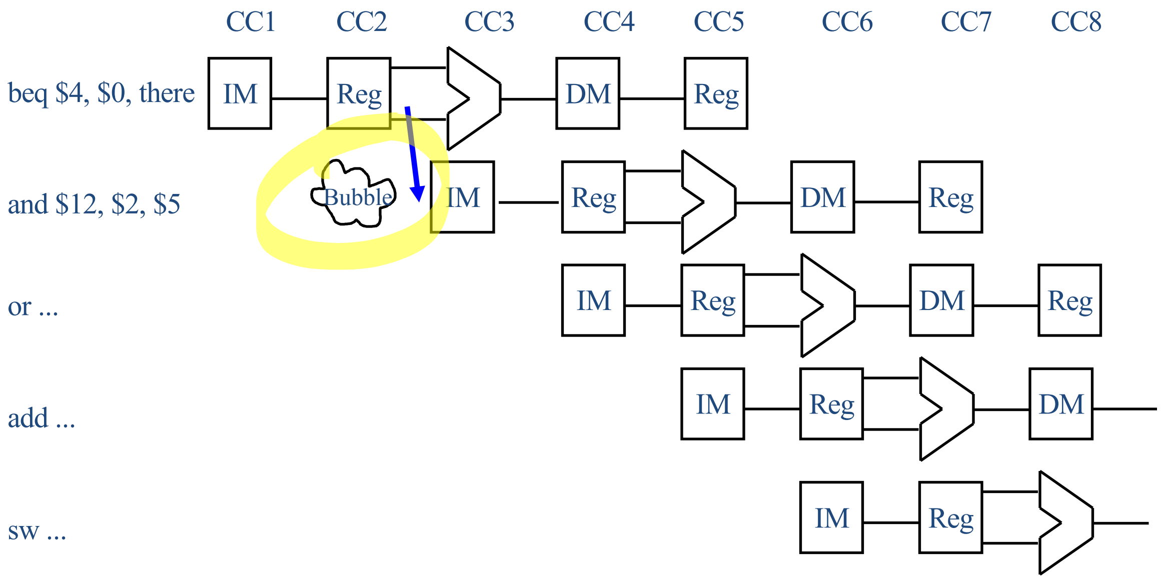
MIPS has ✓
beq
~~bgt/blt~~

→ gt/lt

•Harder, but possible, to get to 1-cycle stall

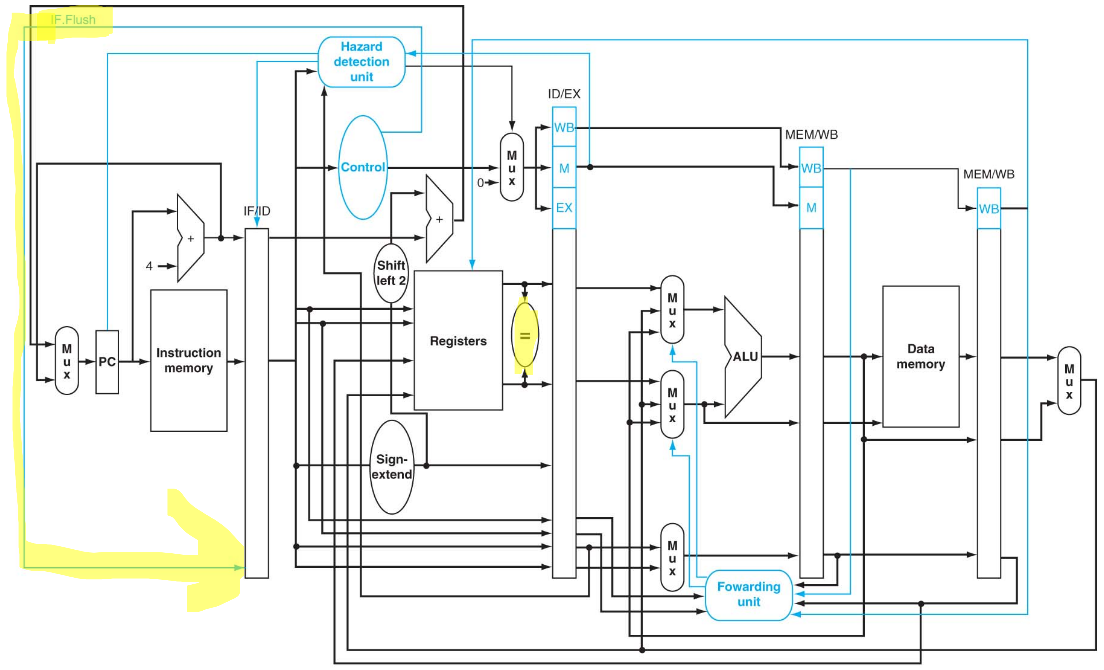
CC BY-NC-ND Pat Pannuto – Many slides adapted from Dean Tullsen

Stalling for Branch Hazards



CC BY-NC-ND Pat Pannuto – Many slides adapted from Dean Tullsen

The Pipeline with flushing for taken branches



- Notice the IF/ID flush line added.

CC BY-NC-ND Pat Pannuto – Many slides adapted from Dean Tullsen

Eliminating the Branch Stall

A cute idea, but not one used by any modern core

- There's no rule that says we have to see the effect of the branch immediately. Why not wait an extra instruction before branching?
- The original SPARC and MIPS processors each used a single *branch delay slot* to eliminate single-cycle stalls after branches.
- The instruction after a conditional branch is *always executed* in those machines, regardless of whether the branch is taken or not!

CC BY-NC-ND Pat Pannuto – Many slides adapted from Dean Tullsen