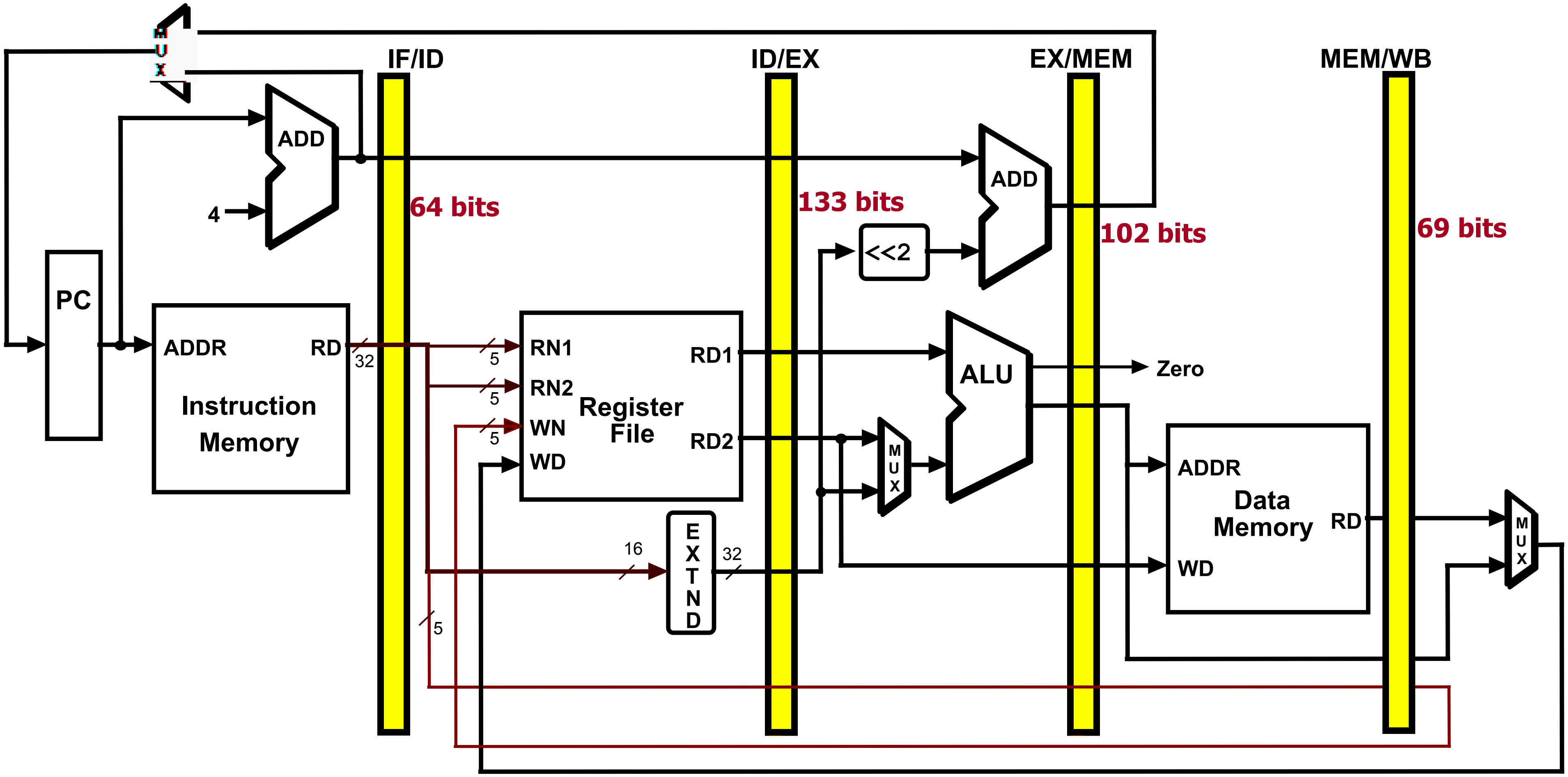
Enhancing Performance with Pipelining

Chapter Four of the Book of David A. Patterson

Corrected Datapath



Destination register number is also passed through ID/EX, EX/MEM and MEM/WB registers, which are now wider by 5 bits

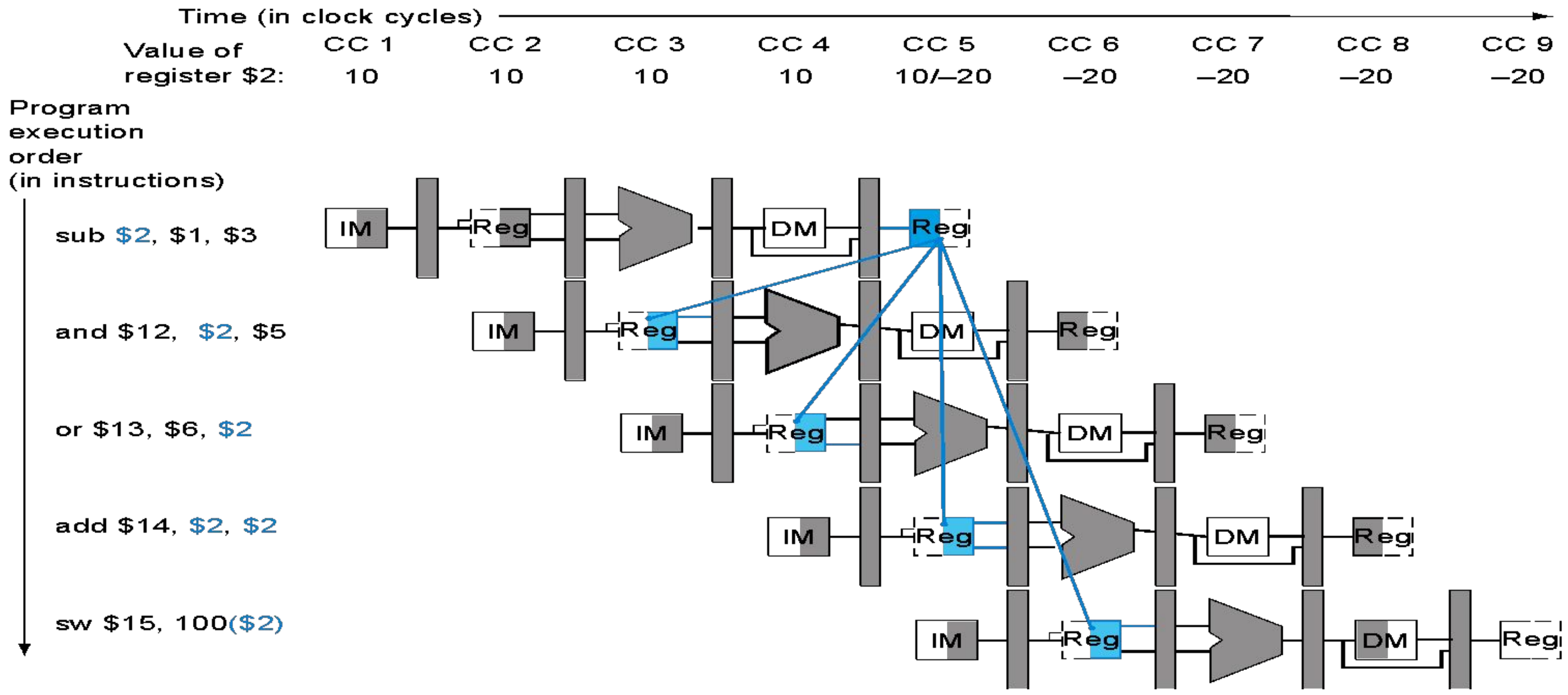
Data Hazards

- ✔ Consider the following code segments:

```
        sub  $2, $1, $3          //$2 written by sub
        and  $12, $2, $5       //1st operand depends on sub
        or    $13, $6, $2      //2nd operand depends on sub
    add  $14, $2, $2     //1st and 2nd operand depend on sub
        sw   $15, 100($2)    //base depends on sub
```

- ✔ One hazard resolved by the design of register file. Write in the first half of the clock cycle and the read in the second half. So read delivers what is written.

Pipelined Dependences



Fixing Data Hazard by Forwarding

Instr. Order

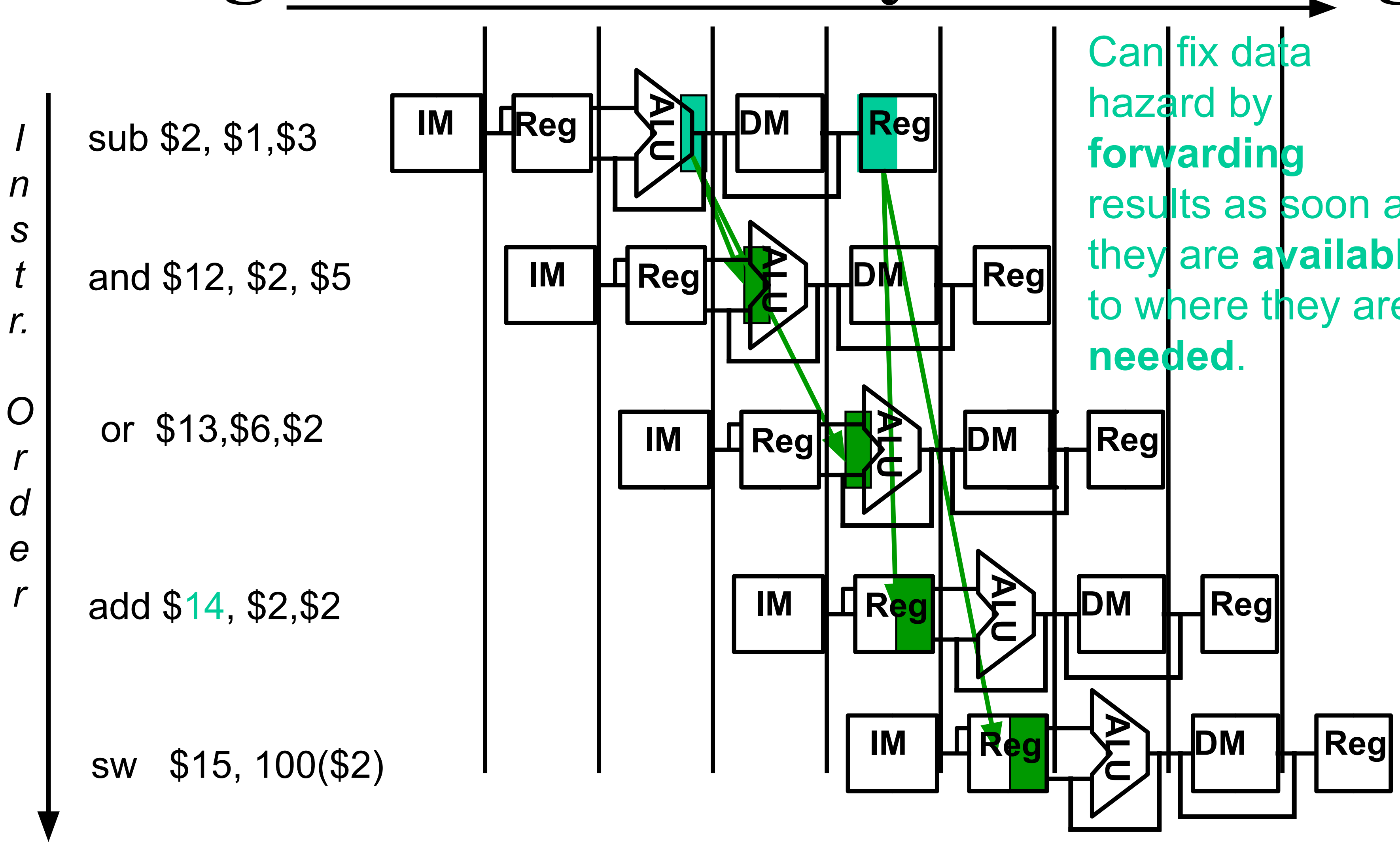
sub $2, $1,$3

and $12, $2, $5

or $13,$6,$2

add $14, $2,$2

sw $15, 100($2)



Can fix data hazard by **forwarding** results as soon as they are **available** to where they are **needed**.

Data Forwarding

- ✔ Data can be forwarded from EX/MEM register and MEM/WB registers.
- ✔ Forward by taking the inputs to the ALU from any pipeline register rather than just ID/EX by
  - adding multiplexors to the inputs of the ALU so can pass Rd back to either (or both) of the EX's stage Rs and Rt ALU inputs
    00: normal input (ID/EX pipeline registers)
    10: forward from previous instr (EX/MEM pipeline registers)
    01: forward from instr 2 back (MEM/WB pipeline registers)
  - adding the proper control hardware
- ✔ With forwarding can run at full speed even in the presence of data dependencies

Hazard Conditions

<u>EX/MEM Hazard:</u>

1a.if (EX/MEM.RegisterRd ==ID/EX.RegisterRs)

  forwardA=10 [Between sub and and]

1b. If (EX/MEM.RegisterRd == ID/EX.RegisterRt)

  forwardB=10

<u>EX/WB Hazard:</u>

2a. If (MEM/WB.RegisterRd == ID/EX.RegisterRs)

  forwardA=01

2b. If (MEM/WB.RegisterRd == ID/EX.RegisterRt)

  forwardB=01 [Between sub and or]

Hazard Conditions

1. EX/MEM hazard:

```
if (EX/MEM.RegWrite
and (EX/MEM.RegisterRd == ID/EX.RegisterRs))
        ForwardA = 10
if (EX/MEM.RegWrite
and (EX/MEM.RegisterRd == ID/EX.RegisterRt))
        ForwardB = 10
```

Forwards the result from the previous instr. to either input of the ALU **provided it writes.**

2. MEM/WB hazard:

```
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd == ID/EX.RegisterRs))
        ForwardA = 01
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd == ID/EX.RegisterRt))
        ForwardB = 01
```

Forwards the result from the second previous instr. to either input of the ALU **provided it writes.**

Hazard Conditions

1. EX/MEM 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
```

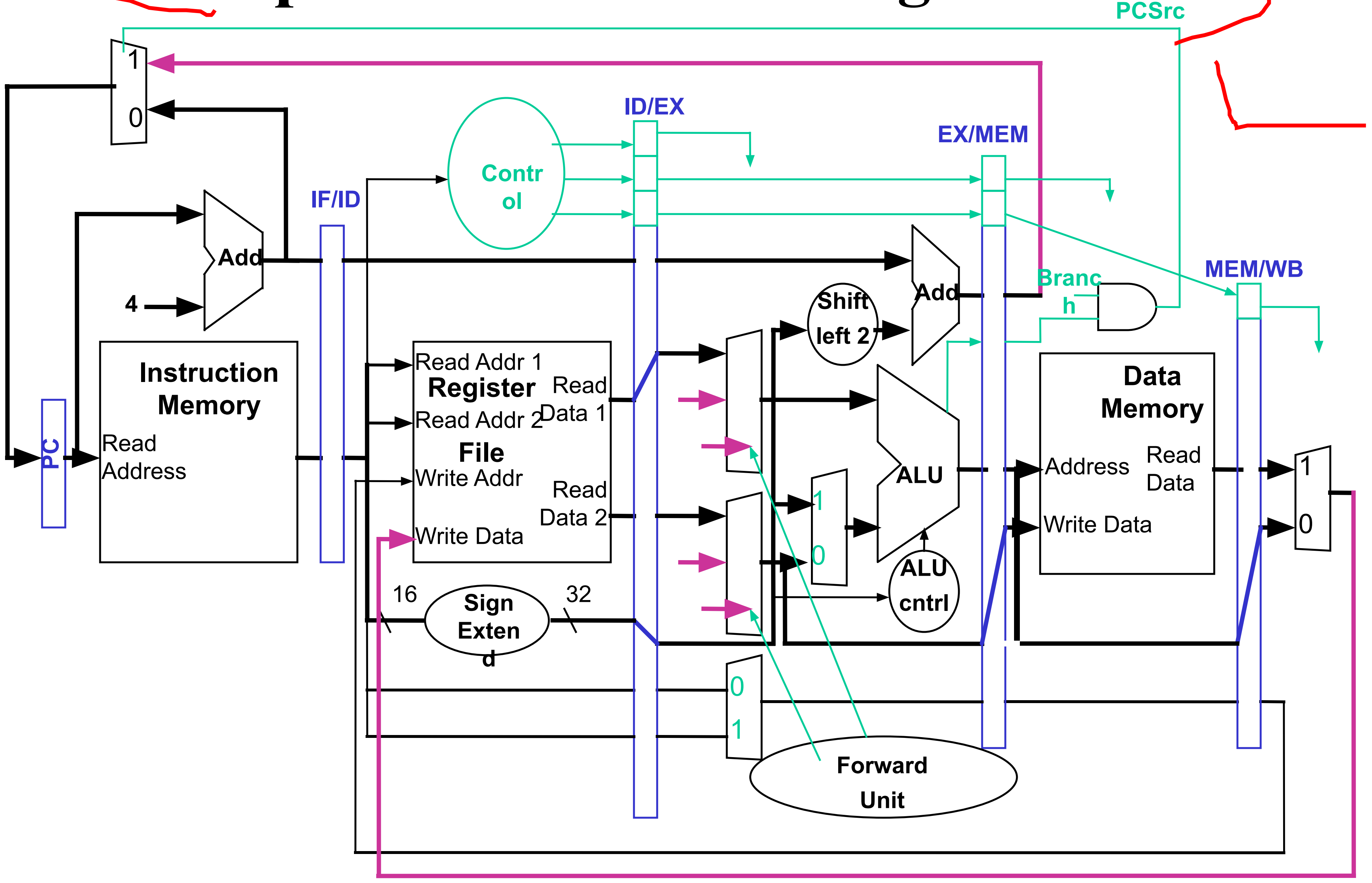
Forwards the result from the previous instr. to either input of the ALU provided it writes **and != $0.**

2. MEM/WB hazard:

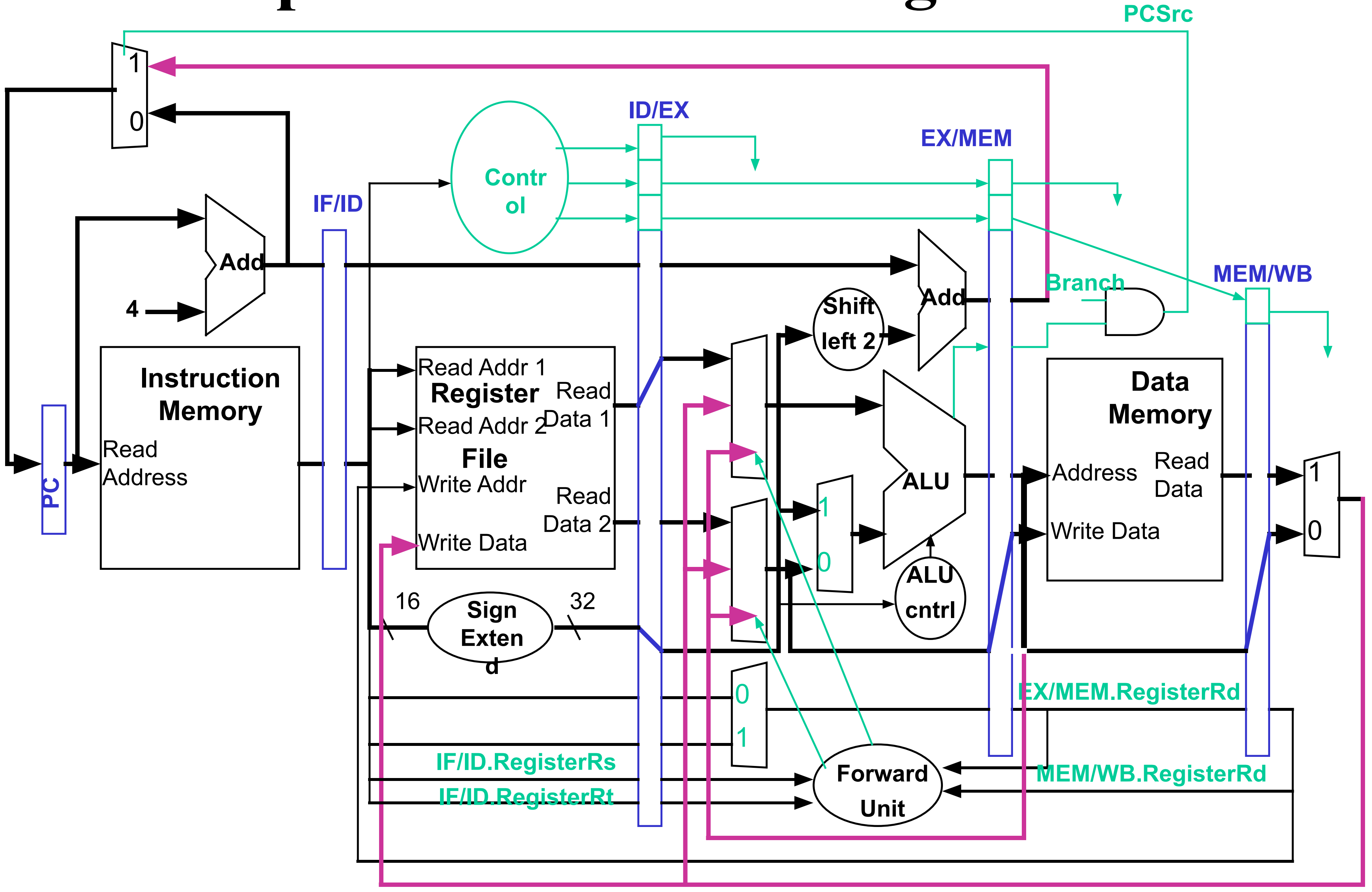
```
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd != 0)
and (MEM/WB.RegisterRd == ID/EX.RegisterRs))
        ForwardA = 01
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd != 0)
and (MEM/WB.RegisterRd == ID/EX.RegisterRt))
        ForwardA = 01
```

Forwards the result from the second previous instr. to either input of the ALU provided it writes **and != $0**

Datapath with Forwarding Hardware

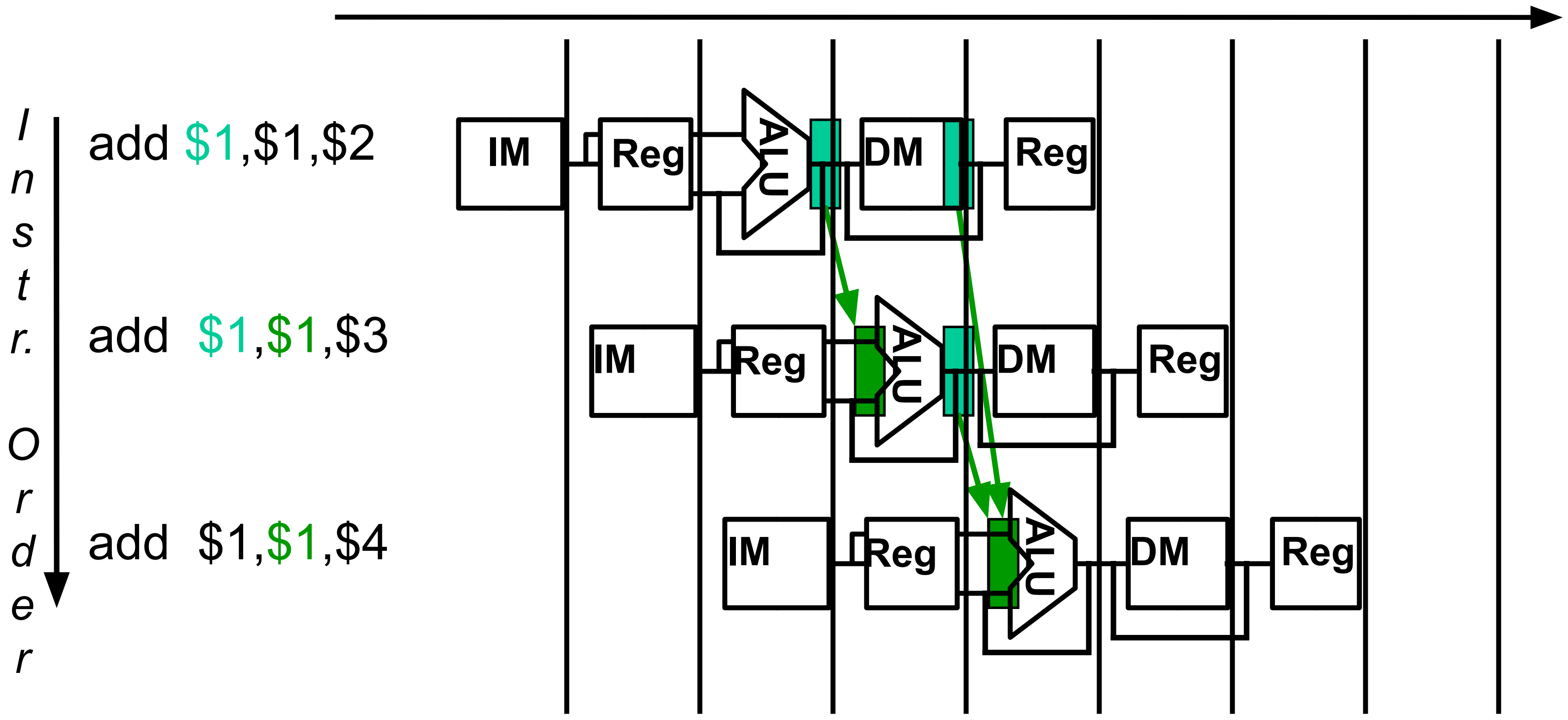


Datapath with Forwarding Hardware



Yet Another Complication!

- Another potential data hazard can occur when there is a conflict between the result of the WB stage instruction and the MEM stage instruction – which should be forwarded? More recent result!



Hazard Conditions

2. **MEM/WB hazard:**

```
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd != 0)
and not (EX/MEM.RegWrite and (EX/MEM.RegisterRd!=0)
    and (EX/MEM.RegisterRd != ID/EX.RegisterRs))
and(MEM/WB.RegisterRd == ID/EX.RegisterRs))
            ForwardA = 01

if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd != 0)
and  not (EX/MEM.RegWrite and (EX/MEM.RegisterRd!=0)
    and (EX/MEM.RegisterRd != ID/EX.RegisterRt ))
and (MEM/WB.RegisterRd == ID/EX.RegisterRt))
        ForwardB = 01
```