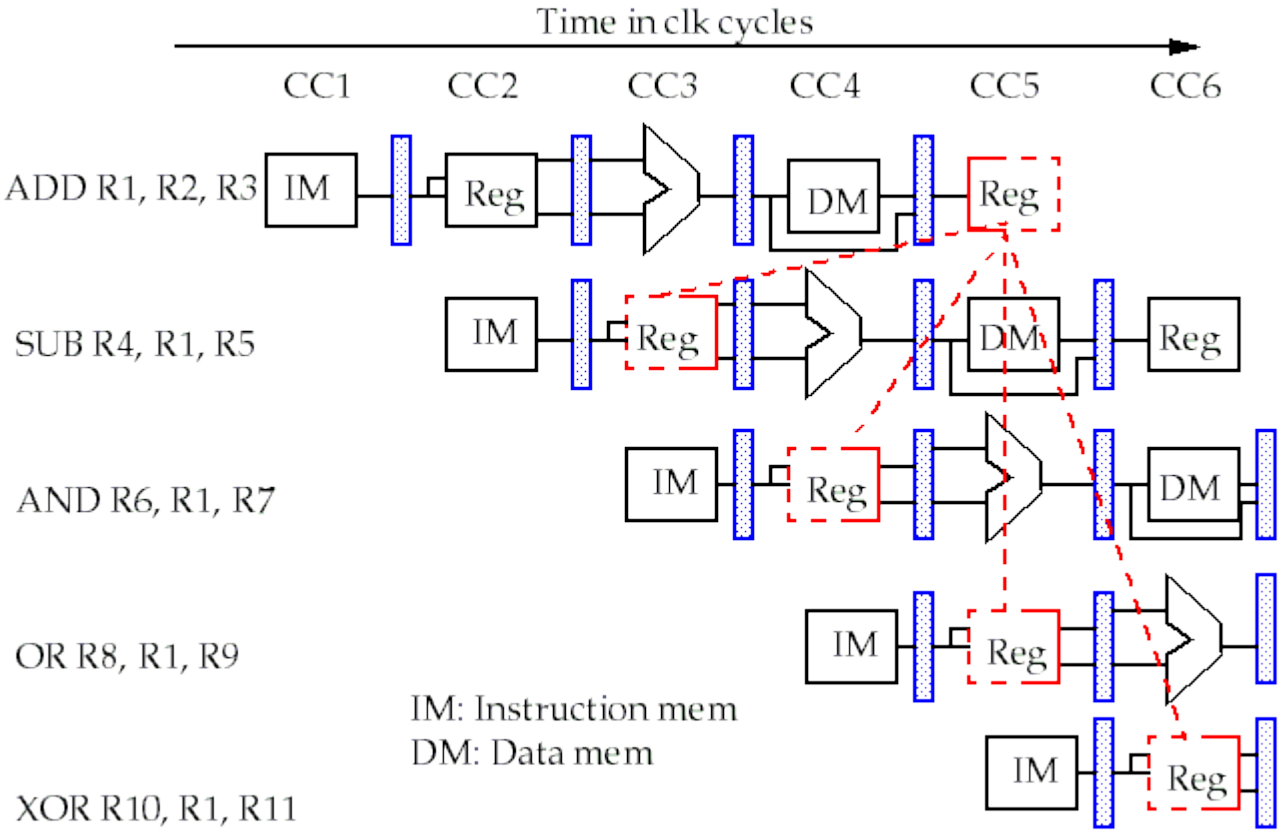
Data Hazards :

- *Pipelining changes the relative timing of instructions by overlapping them in time.*
- *This introduces possible hazards by reordering accesses*
- *To the register file (data hazards.)*
- *To the program counter (control hazards.)*

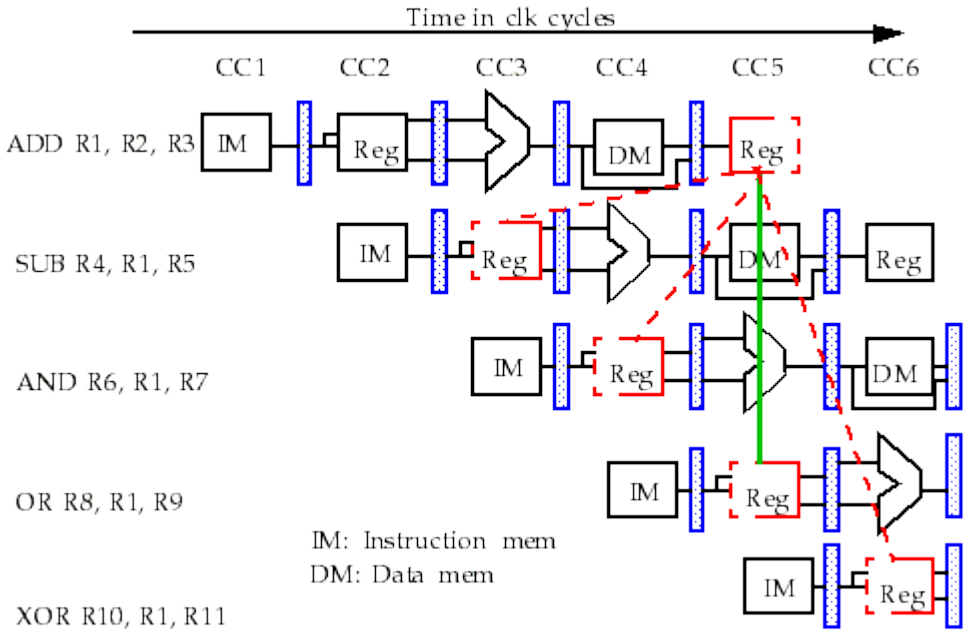
```
ADD R1, R2, R3
SUB R4, R5, R1
AND R6, R1, R7
OR R8, R1, R9
XOR R10, R1, R11
```

- *All of the instructions after ADD use the result of the ADD instruction.*
- *Since the standard pipeline waits until WB to write the value back, the SUB, AND and OR instructions* **read** *the wrong value.*
- *Also, the error may not be deterministic if an interrupt occurs between the ADD and the AND, which would allow the ADD to write its result.*



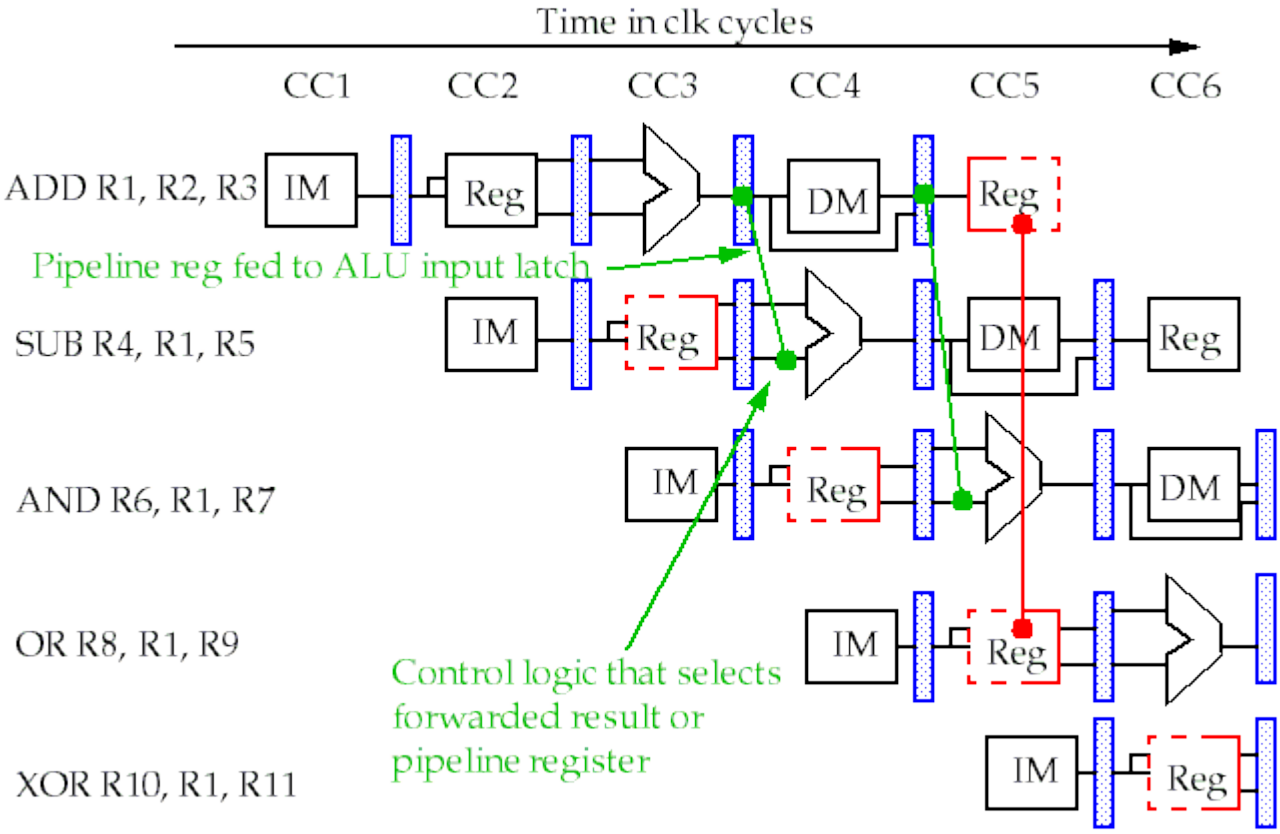
- ***Fixing data hazards:***

- **Simple solution**
  - *The first thing that most pipelines do to avoid hazards is to write the register file in the first half of the cycle and read it in the second half.*



  - *Fixes the hazard shown in green.*

- ***Fixing data hazards:***
- *Forwarding (also called bypassing and short-circuiting)*
  - *A key observation is that the necessary register value is often available but is not in the right place, i.e. the register file.*
  - *This occurs because of the structure of our pipeline.*
  - *The fix: Allow the CPU to move a value directly from one instruction to another without going through the register file.*
  - *This is done by feeding back the data values from the pipeline registers to the inputs of functional units behind them in the datapath.*
  - *The values are forwarded to future instructions.*
    - *They are actually moving backward in the datapath.*



The forwarding is possible between:

- *The output of one functional unit and the input of the same functional unit (previous example).*
- *The output of one functional and the input of another functional unit:*

```
ADD R1, R2, R3
LW R4, 0(R1)
SW 12(R1), R4
```

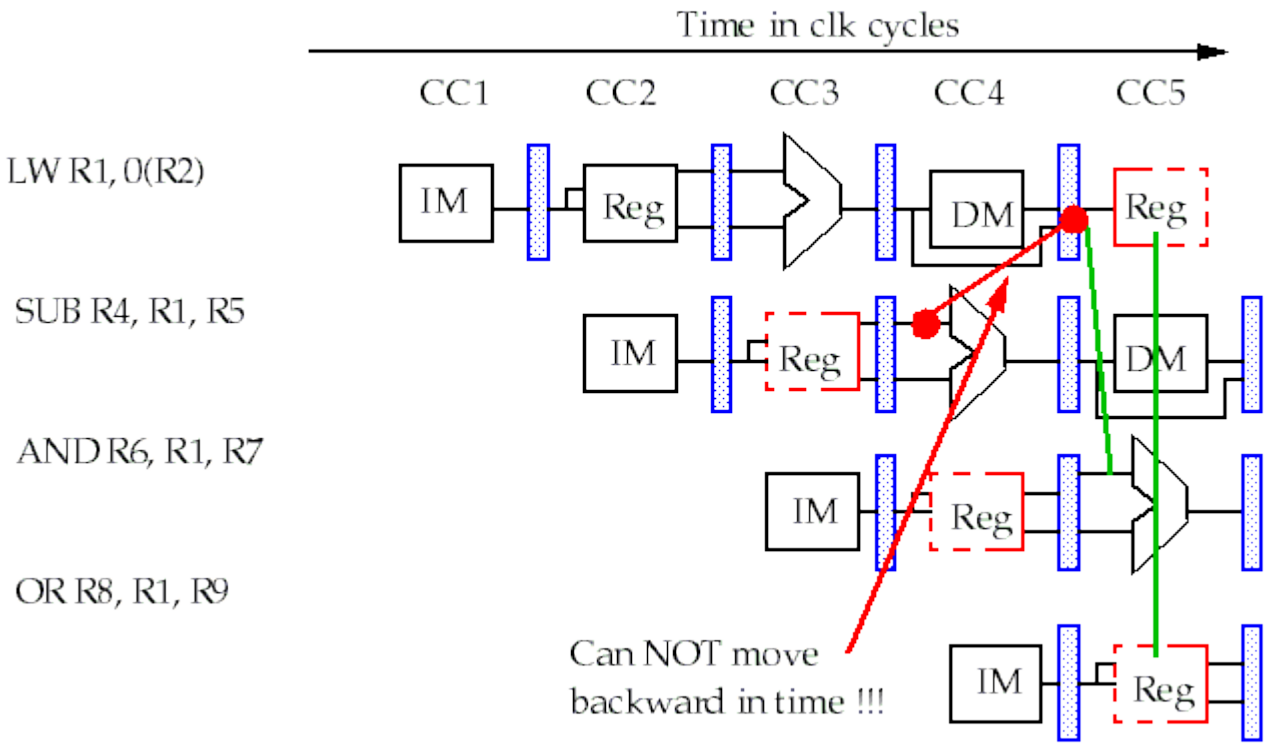
  - *In this case, the pipeline register values for R1 and R4 need forwarded to the input of the ALU and data memory inputs.*
    - *R1 (in EX/MEM) to the input of the ALU for the LW instruction.*
    - *R1 (in MEM/WB) to the input of the ALU for the SW instruction.*
    - *R4 (in MEM/WB) (from memory) to the input of DM for the SW instruction (to memory.)*

- ***Problematic data hazards:***
  - *The forwarding always works in the pipeline for Reg-Reg instructions (prevents stalls.)*
    - *Because all Reg-Reg operations do the real work in the EX stage.*

- *This may not always be the case for other instructions,*
    - *i.e.* ***Load instruction .***
- *Forwarding helps Loads, but it does NOT solve all the problems.*
- *The Load is not completed until after MEM, which is after the EX stage that the following instruction completes.*
- *Sometimes no amount of forwarding can help, as with a Load followed by an ALU operation.*

Data Hazards

- *For example :*



- *Stalling is necessary in this case for proper execution.*
    - *This is done with a pipeline interlock , which stalls the pipeline until the hazard is cleared.*
- *This inserts a bubble into the pipeline just as the structural hazard did.*
    - *Just as with structural hazards, no instructions are started during the cycle in which the bubble is inserted.*
    - *This increases the number of cycles required and thus the CPI.*