CMP301B/CMPN301: Computer Architecture

Pipelining Hazards

Mayada Hadhoud
Computer Engineering Department
Cairo University

Agenda

- Pipelining Hazards
- Structural Hazards
  - Structural Hazards Solutions
- Data Hazards
  - Data Hazards Solutions
- Control Hazards
  - Control Hazards Solutions
- Exceptions

Question??

- Assume
  - Fetch = 10 ns
  - Decode = 6 ns
  - Execute = 8 ns
  - Memory = 10 ns
  - Write back = 6 ns

  What is the speedup of a pipelined processor Vs. Single cycle processor and Multi cycle processor ?

  Given : Number of instructions is 1000 instruction (50% add and sub instructions, 10% load , 10% store, 30% branch)

Pipeline Hazards

- **Hazards**: Situations that prevent an instruction from being executed in its designated clock cycle

- **Types of Hazards**
  1. **Structural Hazards**: two different instructions use same h/w in same cycle
  2. **Data Hazards**: An instruction depends on the results of previous instruction
  3. **Control Hazards**: When an instruction changes PC, like branches

- The simplest hazards solution is to **stall the pipeline** (some instructions are allowed to proceed, while other are delayed)

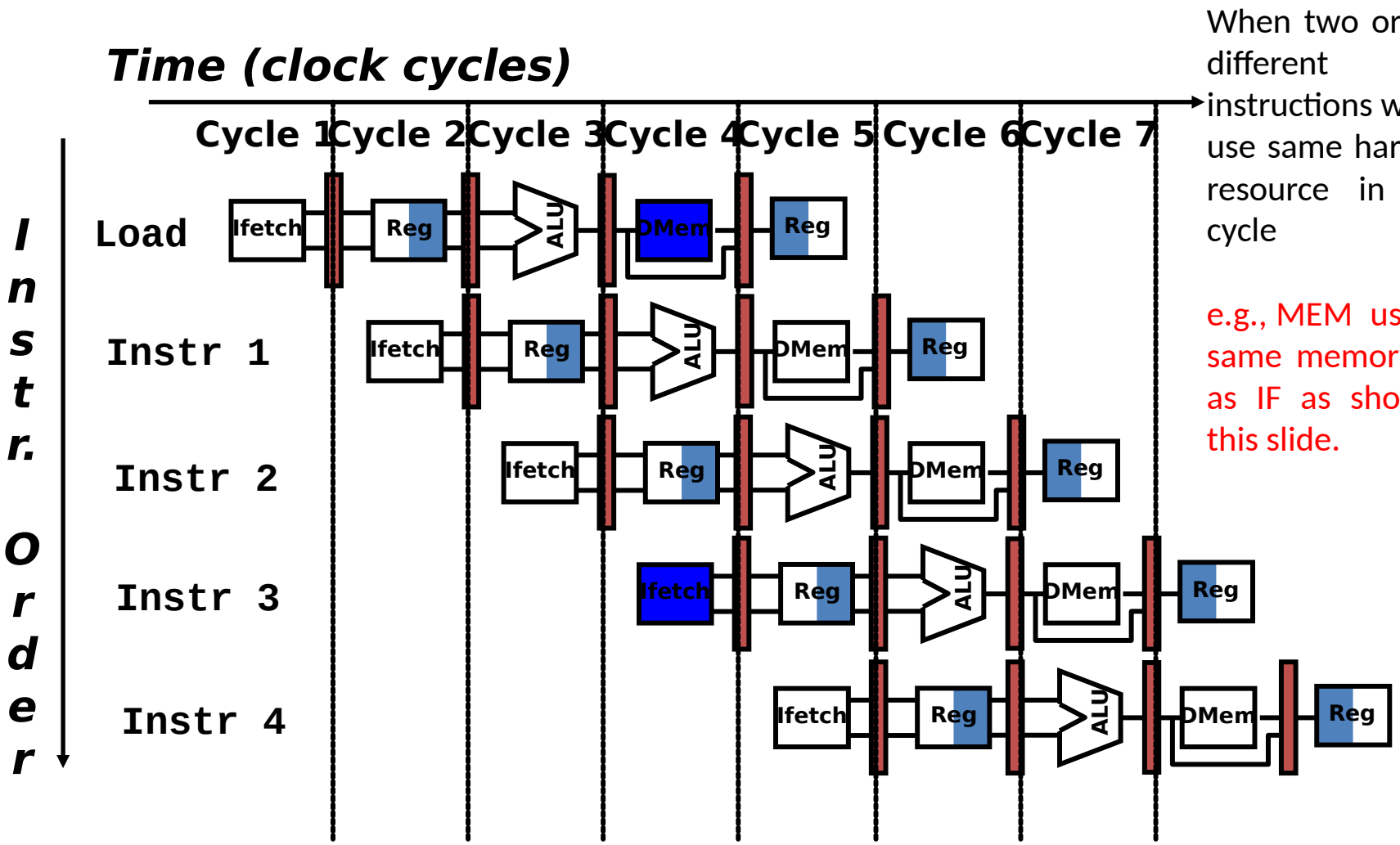
Structural hazards

- No two instructions are processed by the same module at the same time.
- **Solutions**:
  - Stalling the pipeline (inserting bubbles)
  -

Structural hazards

- **Examples**
  - **Instruction fetch is conflicting with data memory access**
    - Use two separate caches one for instructions and one for the data.
  - **Register file is accessed by the instruction in the decode (reading) and the instruction in the write-back (writing)**
    - Ensure that writing is done in the first half of the clock and reading is done at the second half of the clock

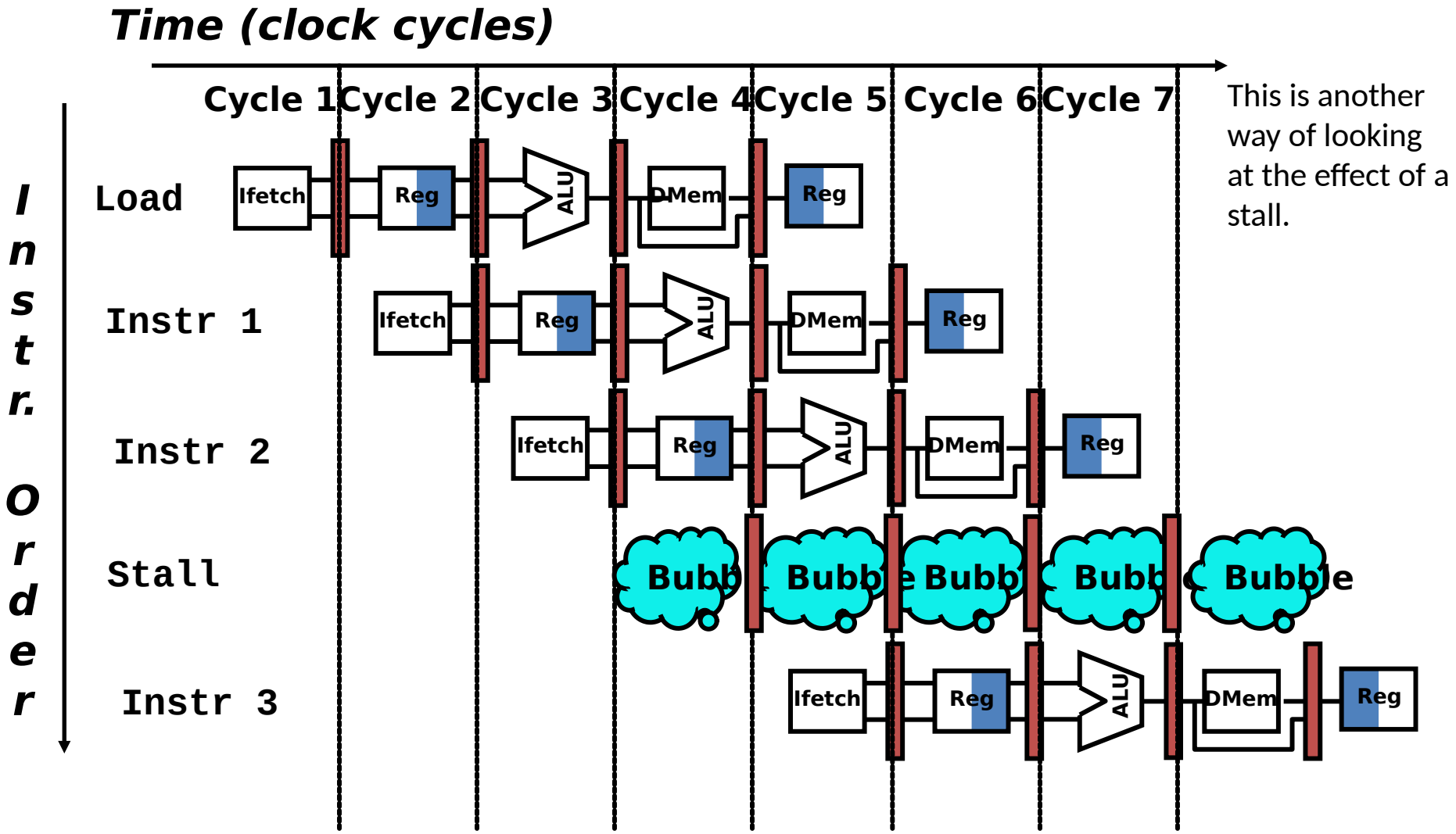
Structural Hazards



When two or more different instructions want to use same hardware resource in same cycle

e.g., MEM uses the same memory port as IF as shown in this slide.

Structural Hazards



This is another way of looking at the effect of a stall.

Data Hazards

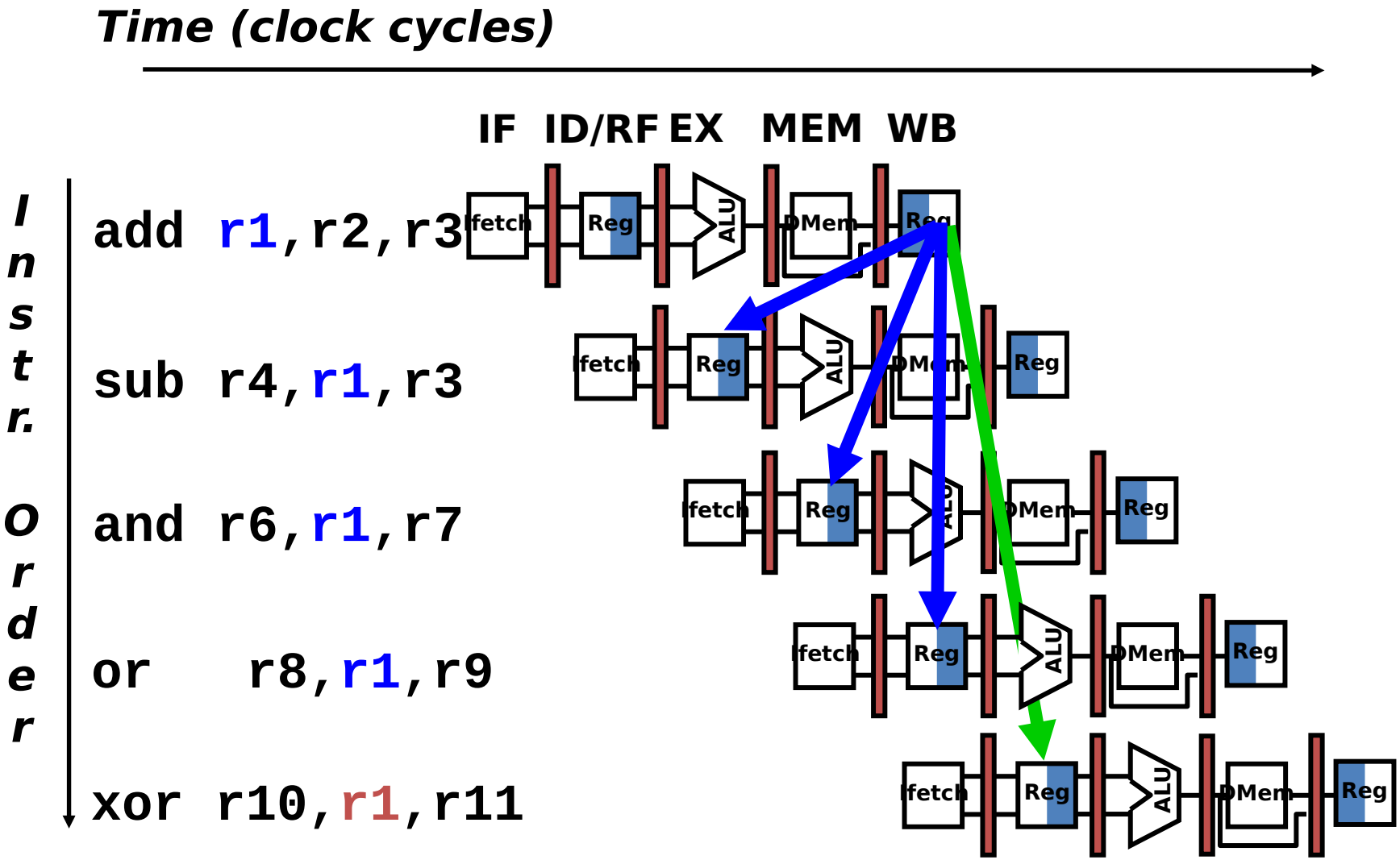
- **An occurrence in which a planned instruction cannot execute in the proper clock cycle because data that is needed to execute the instruction is not yet available.**

Data Hazards

- Data hazards arise from the dependence of one instruction on an earlier one that is still in the pipeline.
- **Example**:

```
add     $s0, $t0, $t1
sub     $t2, $s0, $t3
```

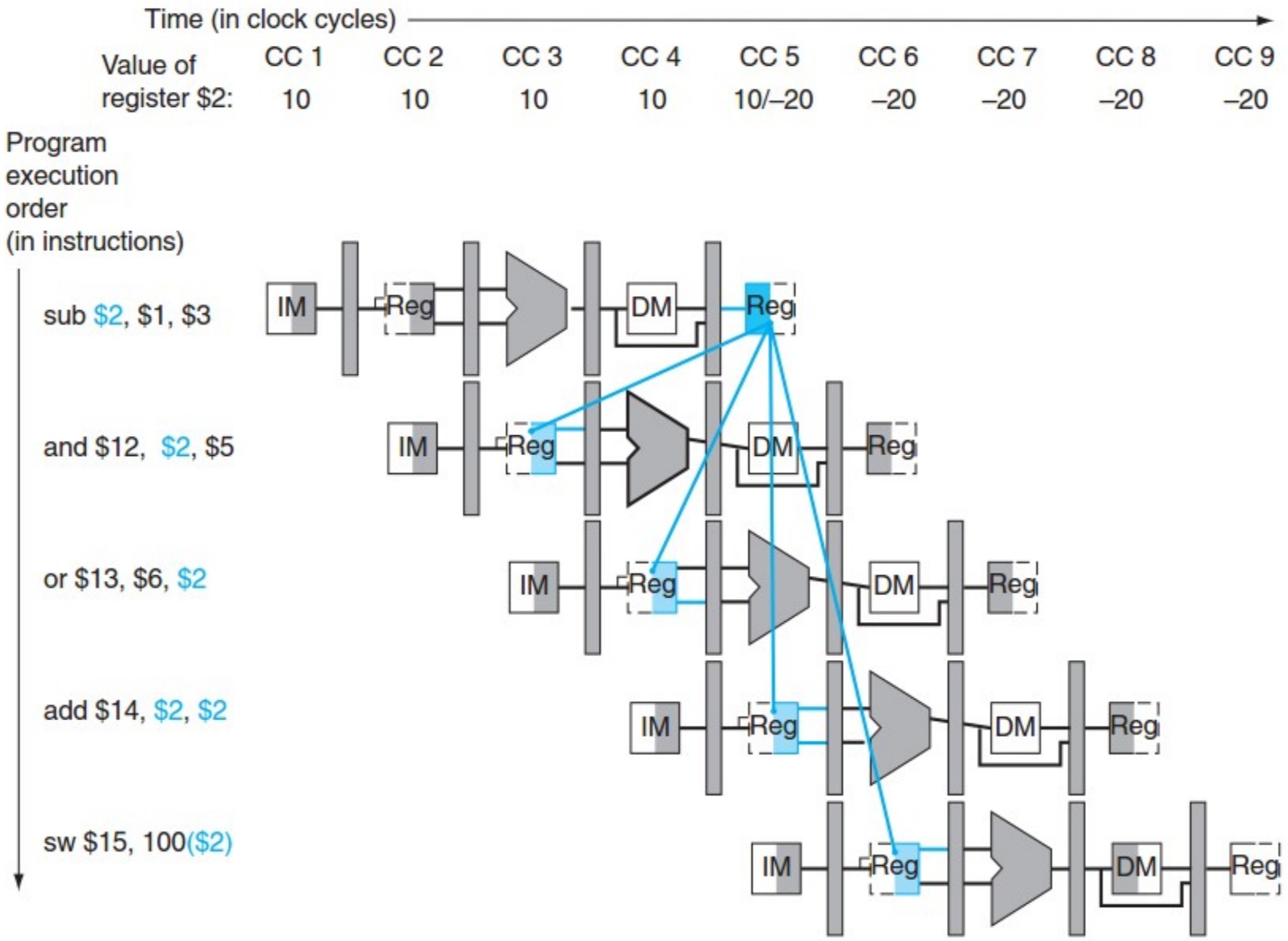
Example


Time (clock cycles)
IF
ID/RF
EX
MEM
WB
Instr. Order
add r1,r2,r3
sub r4,r1,r3
and r6,r1,r7
or r8,r1,r9
xor r10,r1,r11
Ifetch
Reg
ALU
DMem
Reg

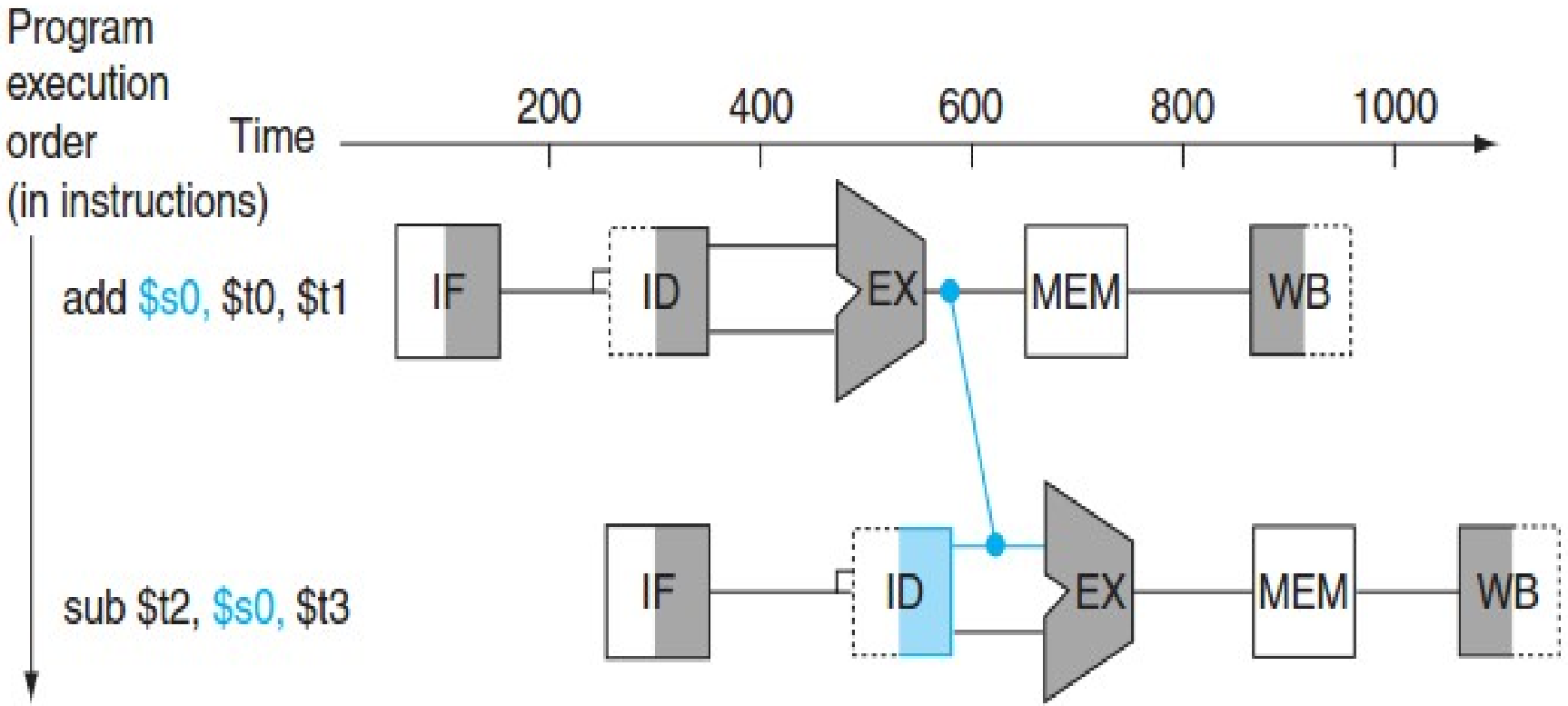
Data Hazards- Solutions

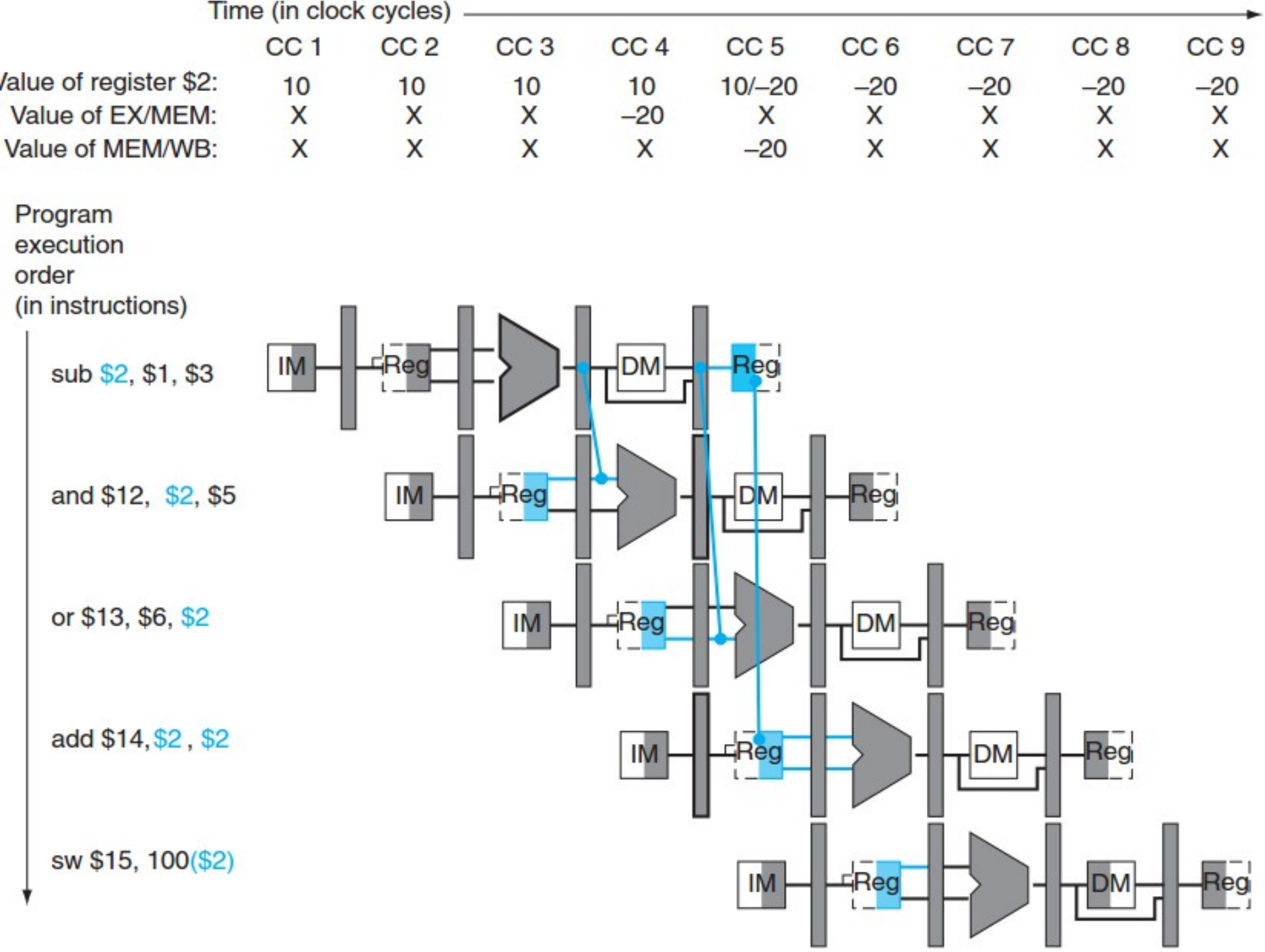
- **Solution (1)**: **Software-solution**: Compiler arranges instructions to avoid data dependencies
- **Solution (2)**: **Data Forwarding "or bypassing":** forward the data back to the requesting stage "immediately when it is available"
- **Solution (3)**: **Stalling the pipeline**: inserting bubbles and stall the instruction till the data is written back

Data Forwarding

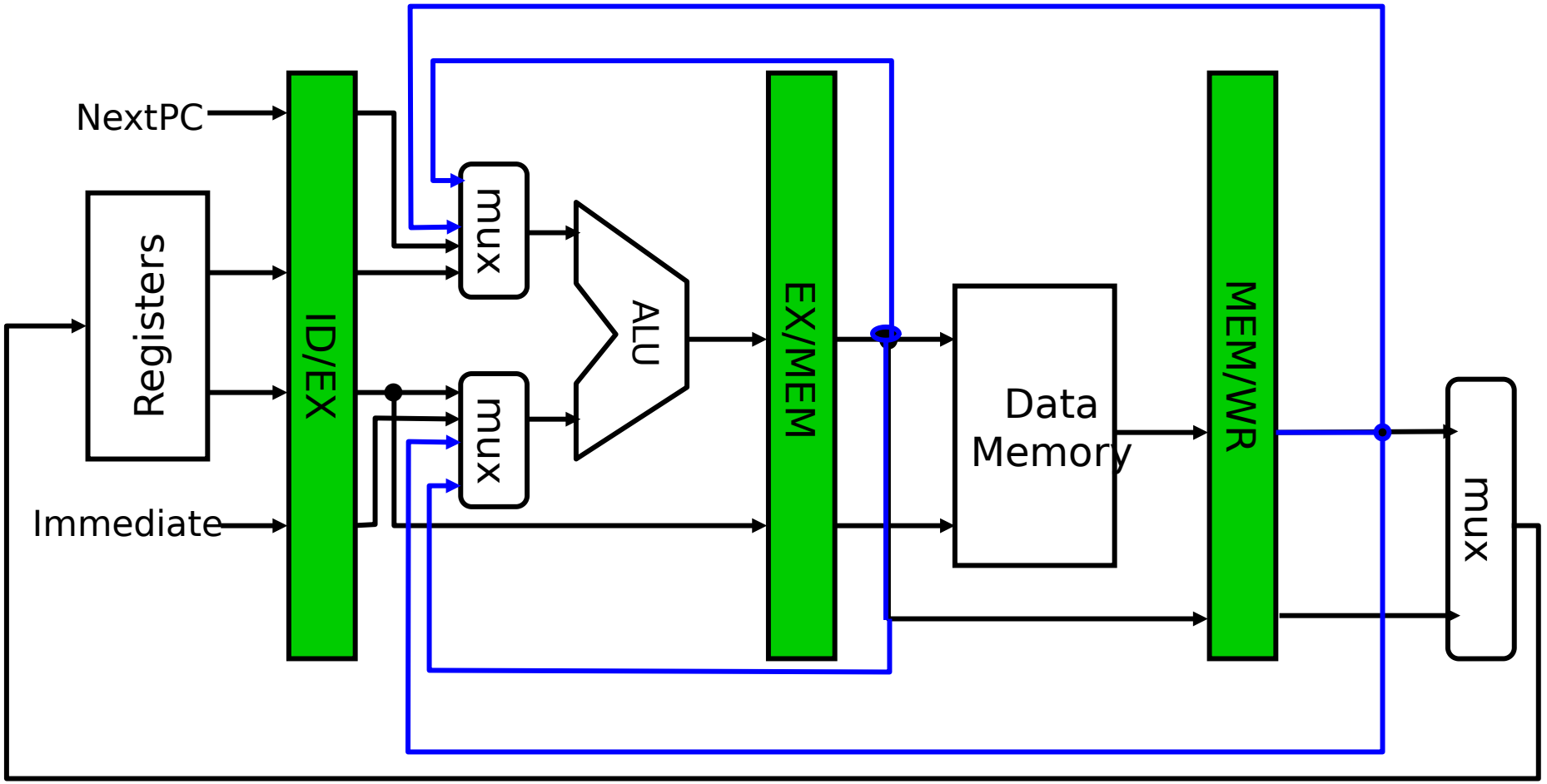


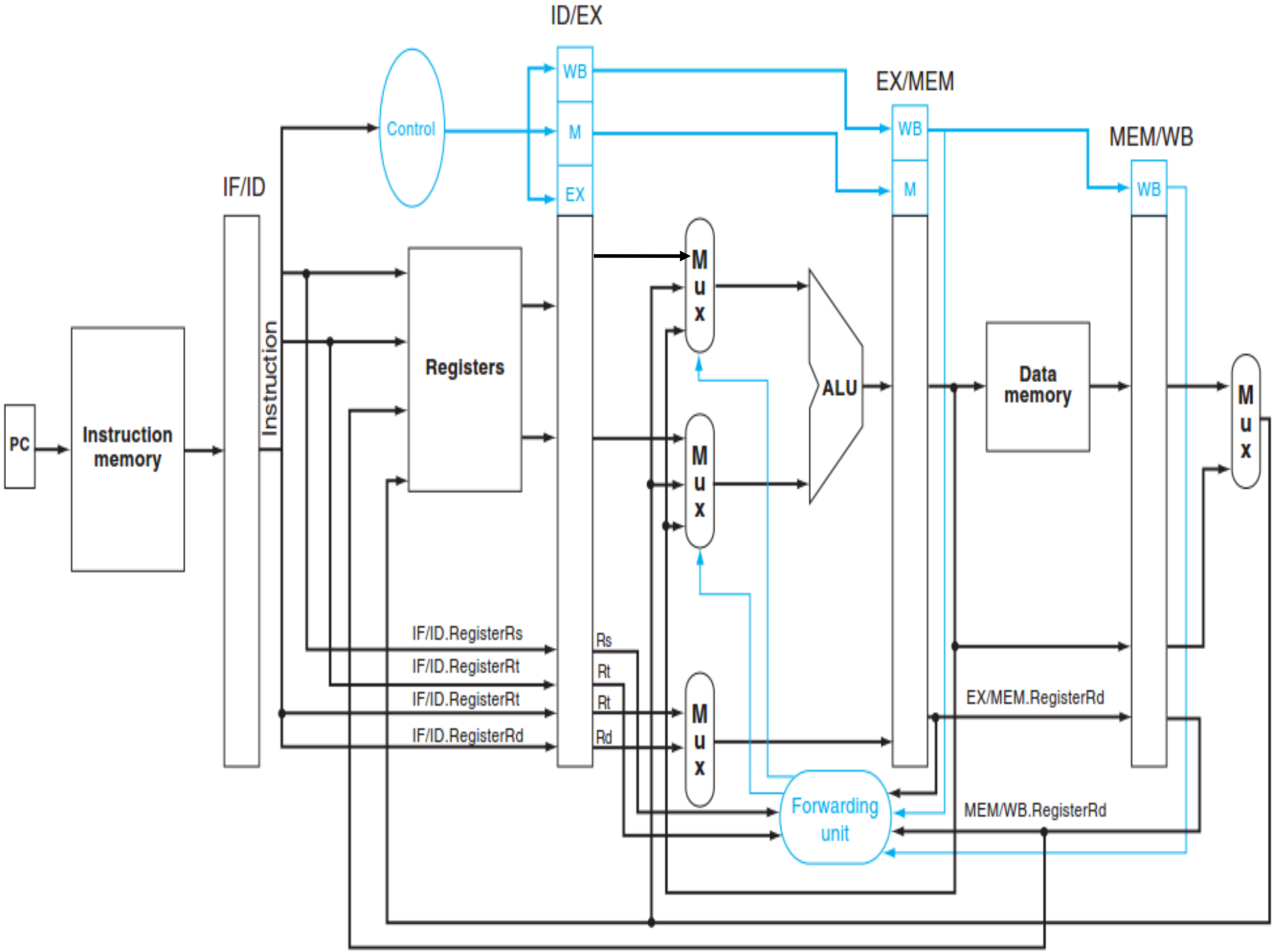
Data Forwarding




Time (in clock cycles)
CC 1
CC 2
CC 3
CC 4
CC 5
CC 6
CC 7
CC 8
CC 9
Value of register $2:
10
10
10
10
10/–20
–20
–20
–20
–20
Value of EX/MEM:
X
X
X
–20
X
X
X
X
X
Value of MEM/WB:
X
X
X
X
–20
X
X
X
X
Program execution order (in instructions)
sub $2, $1, $3
and $12, $2, $5
or $13, $6, $2
add $14, $2 , $2
sw $15, 100($2)
IM
Reg
DM
Reg

HW Change for Forwarding




ID/EX
EX/MEM
MEM/WB
IF/ID
Control
WB
M
EX
WB
M
WB
PC
Instruction memory
Instruction
Registers
Mux
Mux
Mux
ALU
Data memory
Mux
IF/ID.RegisterRs
IF/ID.RegisterRt
IF/ID.RegisterRt
IF/ID.RegisterRd
Rs
Rt
Rt
Rd
EX/MEM.RegisterRd
Forwarding unit
MEM/WB.RegisterRd

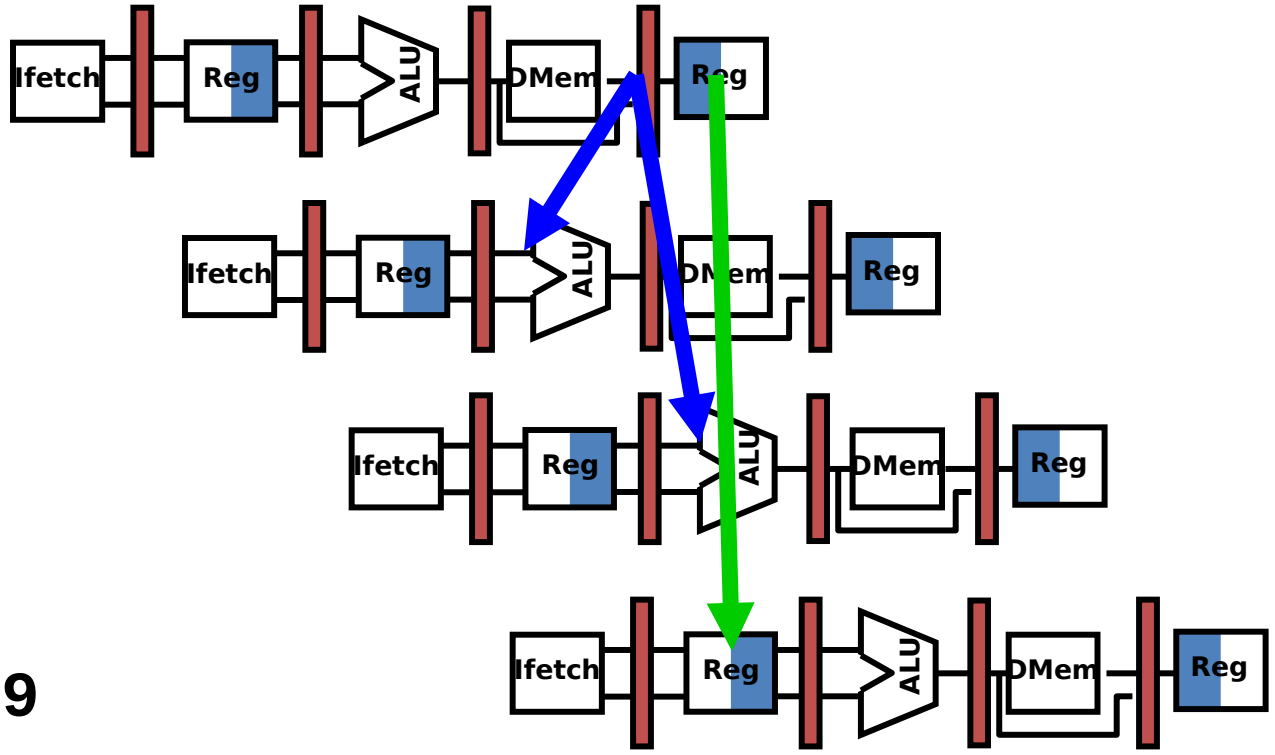
Load-Use Data Hazard

- A specific form of data hazard in which the data requested by a load instruction has not yet become available when it is requested.

Time (clock cycles)

Instr. Order

```
lw r1, 0(r2)
sub r4,r1,r6
and r6,r1,r7
or   r8,r1,r9
```



Software Scheduling to Avoid Load Hazards

Try producing fast code for

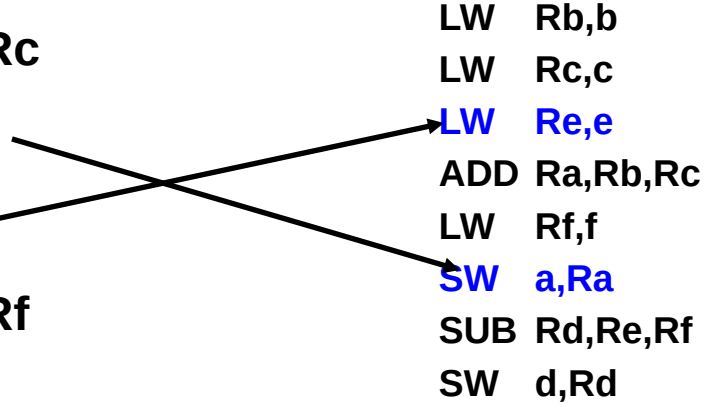
a = b + c;

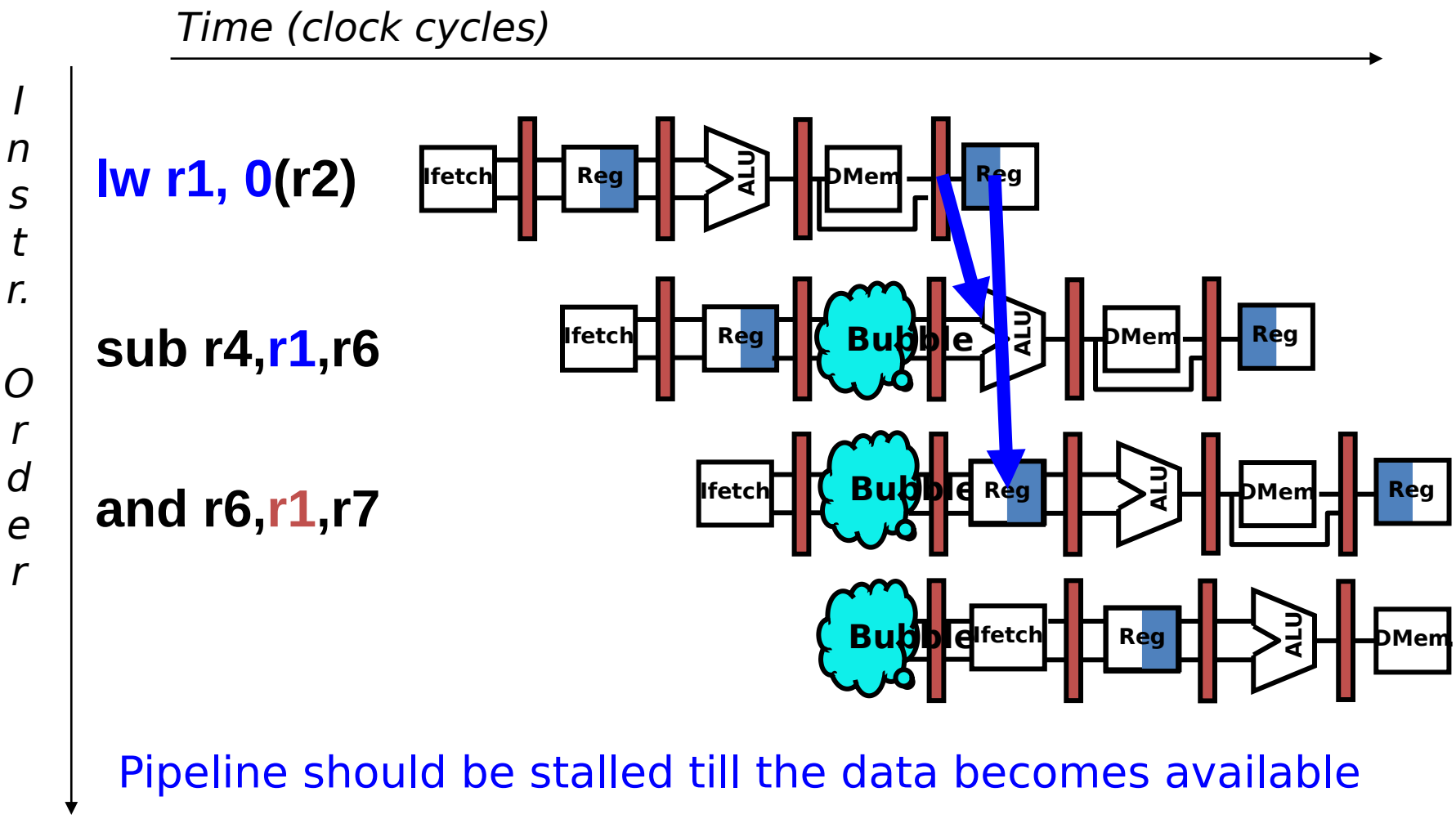
d = e – f;

assuming a, b, c, d ,e, and f in memory.

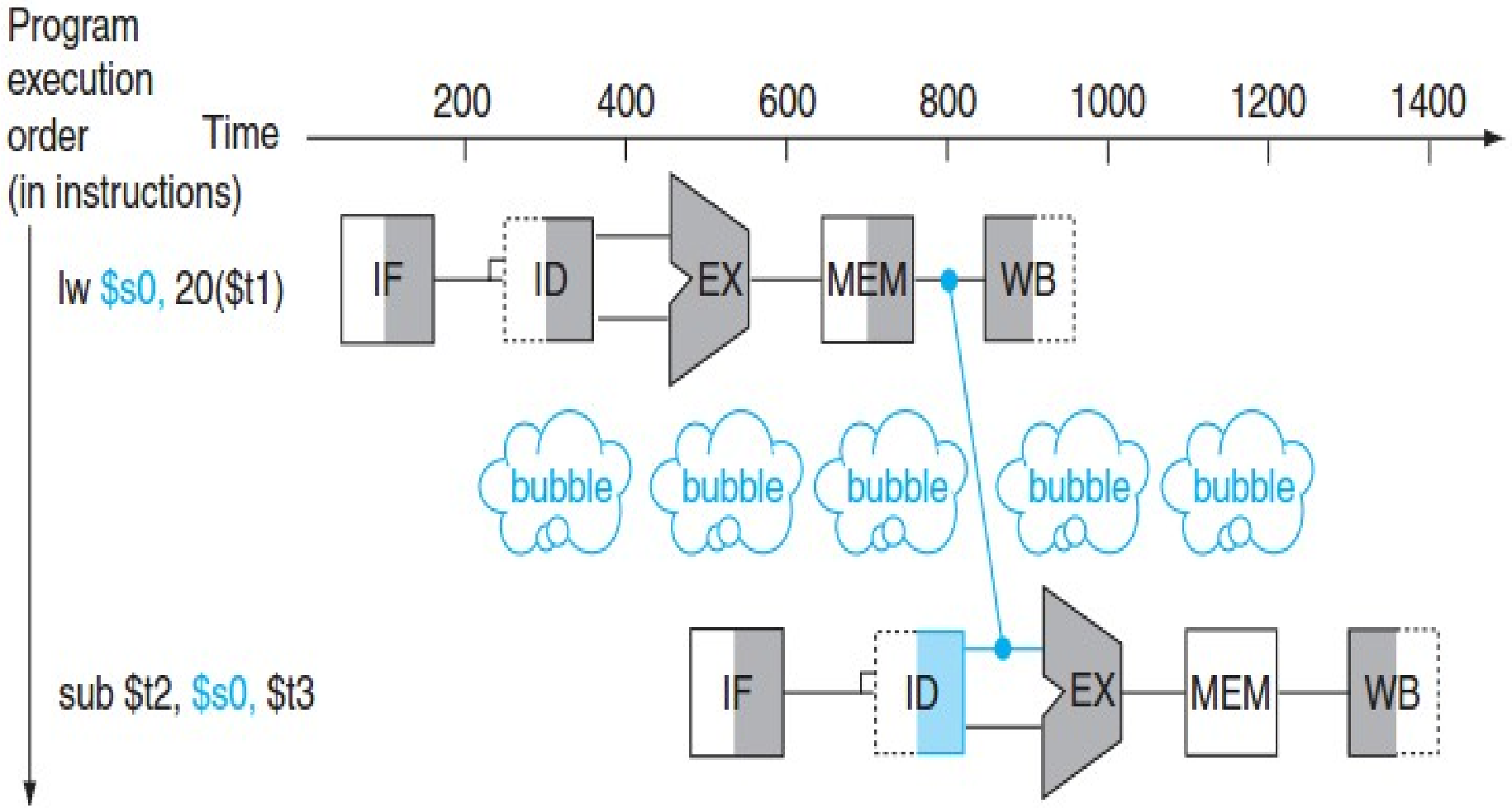
Slow code:

```
LW  Rb,b
LW  Rc,c
ADD     Ra,Rb,Rc
SW      a,Ra
LW  Re,e
LW  Rf,f
SUB     Rd,Re,Rf
SW  d,Rd
```




Time (clock cycles)
Instr. Order
lw r1, 0(r2)
sub r4,r1,r6
and r6,r1,r7
Ifetch
Reg
ALU
DMem
Reg
Bubble
Pipeline should be stalled till the data becomes available

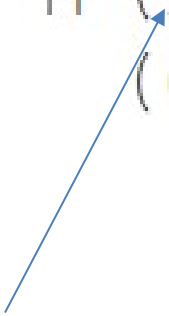
Load-Use Data Hazard



Load-Use Data Hazard

- Load-Use Hazard check

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



Is the instruction is load ??

How can we insert NOP in the pipe??

Data Hazard Detection H/W