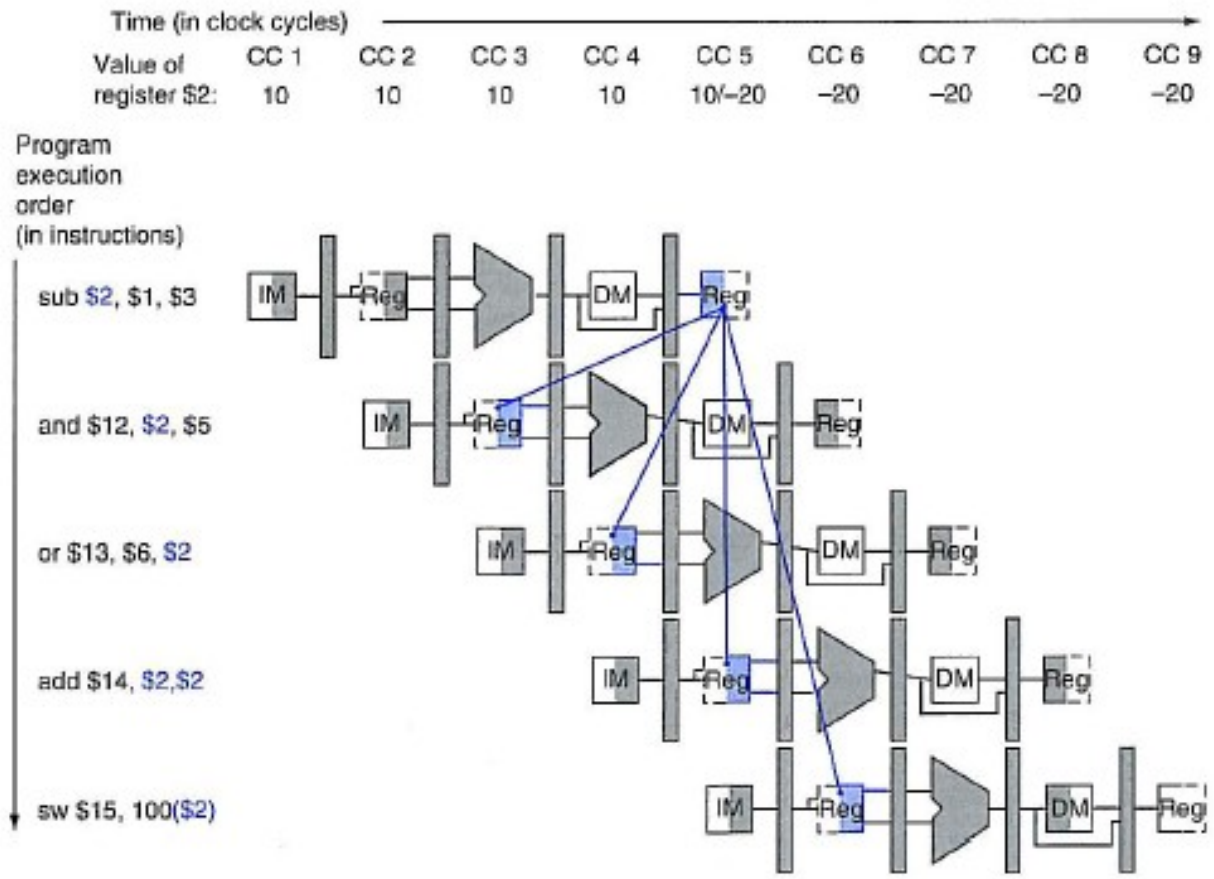
DATA HAZARDS 1/17
Ejemplo de Código con Dependencias

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

DATA HAZARDS 2/17
Representación Multiclock de Dependencias



DATA HAZARDS 3/17
Notaciones y Tipos de Data Hazards

1a. EX/MEM.RegisterRd = ID/EX.RegisterRs

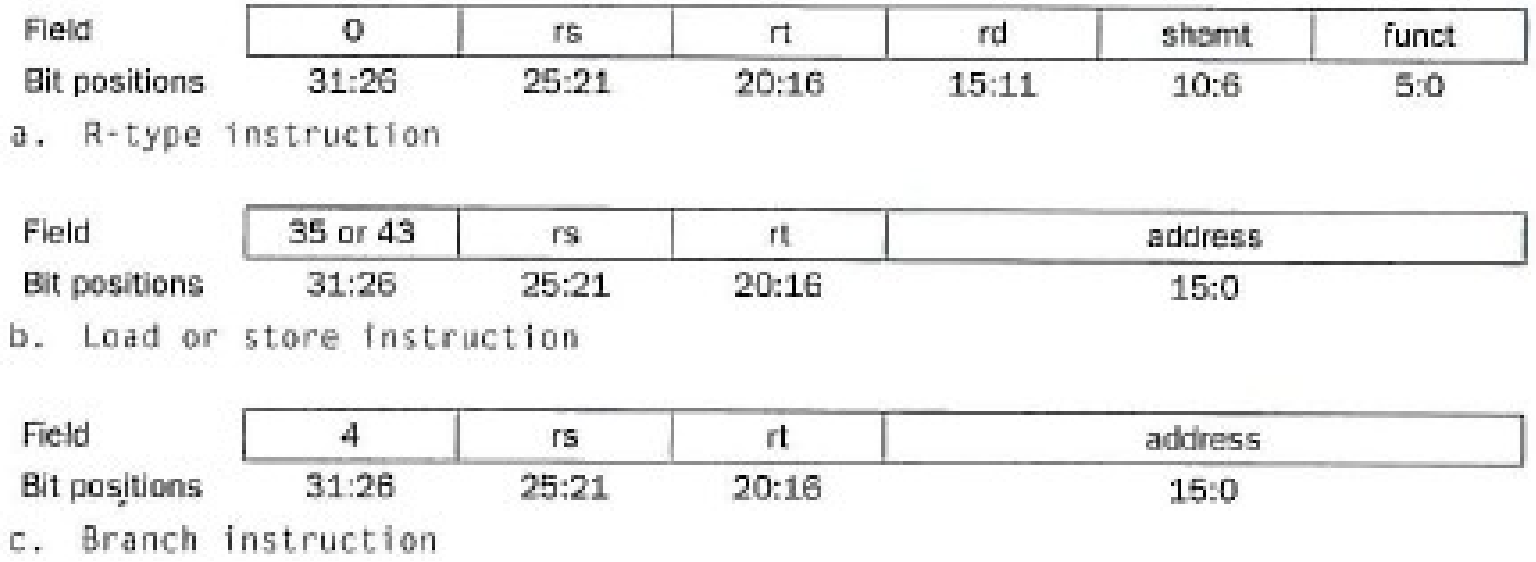
1b. EX/MEM.RegisterRd = ID/EX.RegisterRt

2a. MEM/WB.RegisterRd = ID/EX.RegisterRs

2b. MEM/WB.RegisterRd = ID/EX.RegisterRt

```
sub   $2,    $1, $3   # Register $2 set by sub
and   $12,   $2, $5   # 1st operand($2) set by sub
or    $13,   $6, $2   # 2nd operand($2) set by sub
add   $14,   $2, $2   # 1st($2) & 2nd($2) set by sub
sw    $15,   100($2) # Index($2) set by sub
```

DATA HAZARDS 4/17
Campos de Instrucciones



Field	0	rs	rt	rd	shamt	funct
Bit positions	31:26	25:21	20:16	15:11	10:6	5:0

a. R-type instruction

Field	35 or 43	rs	rt	address
Bit positions	31:26	25:21	20:16	15:0

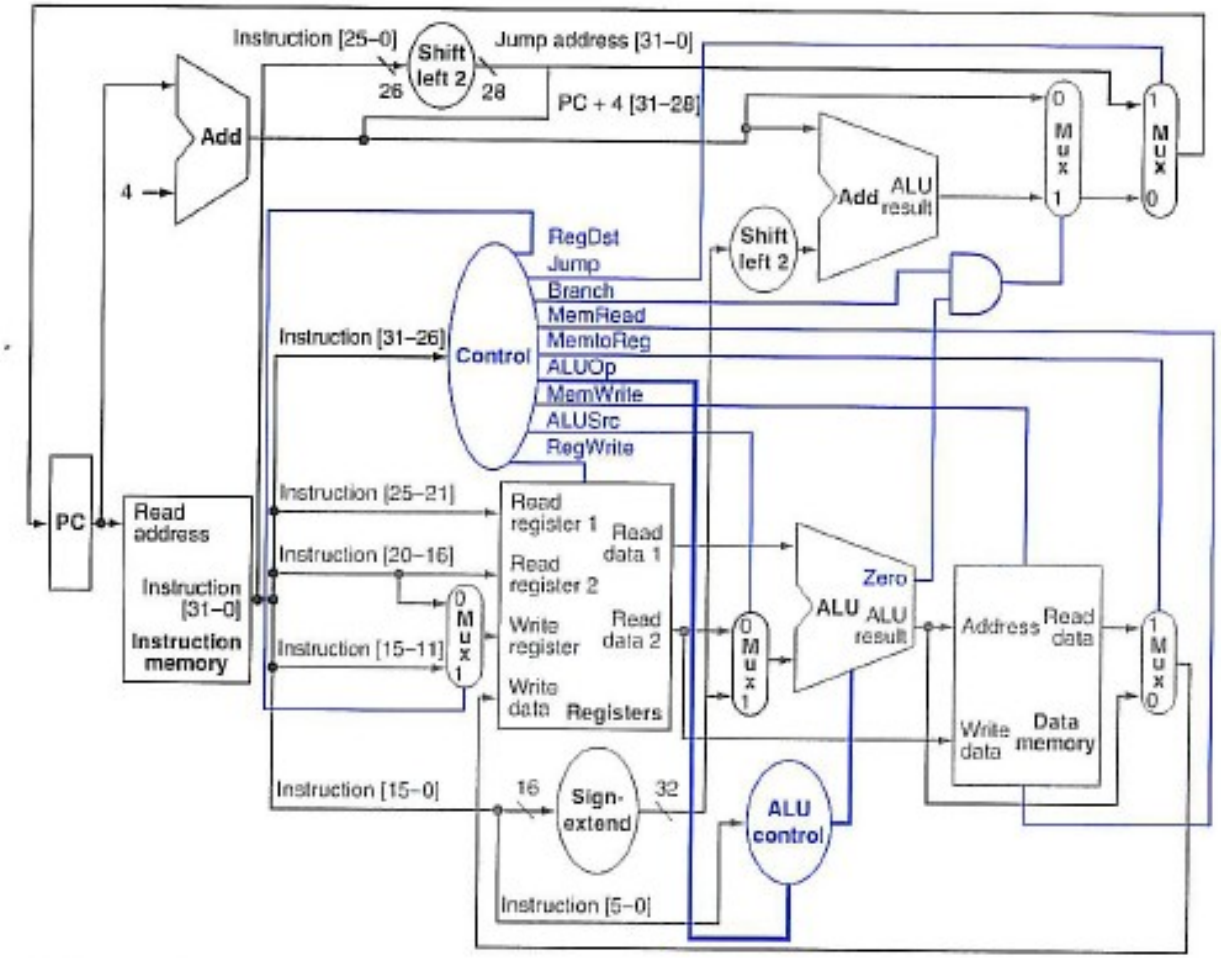
b. Load or store instruction

Field	4	rs	rt	address
Bit positions	31:26	25:21	20:16	15:0

c. Branch instruction

add rd,rs,rt
and rd,rs,rt
or rd,rs,rt
sw rt,base(rs)
lw rt base(rs)

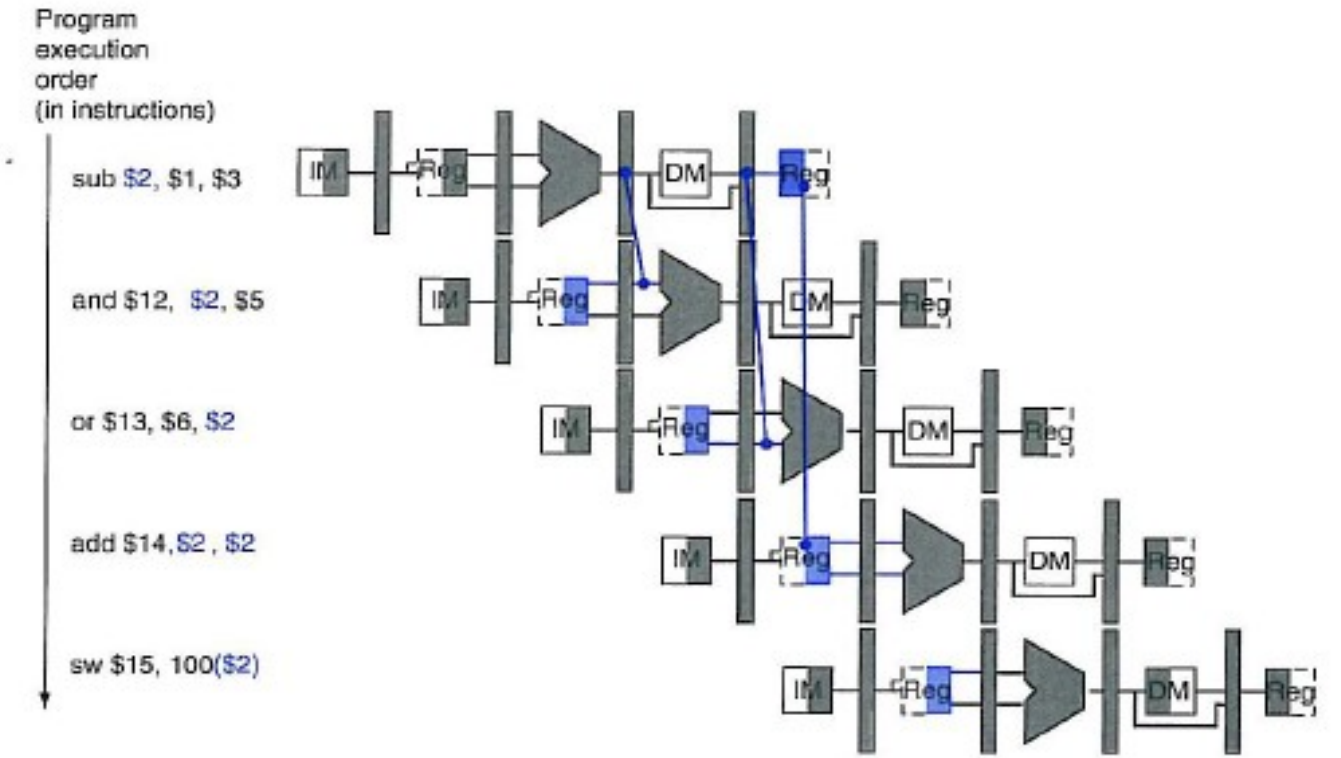
DATA HAZARDS 5/17
Campos de Instrucciones



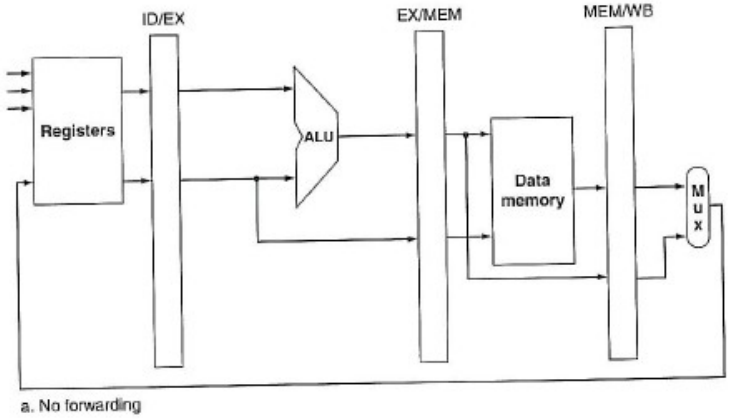
DATA HAZARDS 6/17
Forwarding

Time (in clock cycles) →

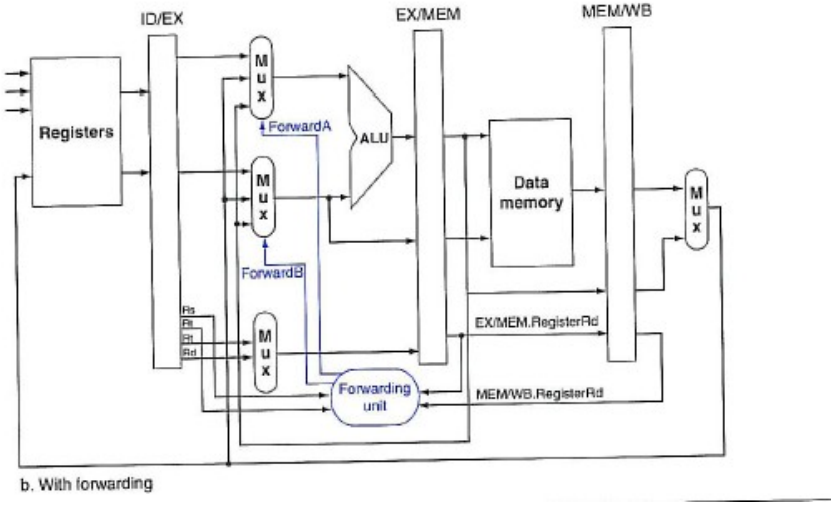
	CC 1	CC 2	CC 3	CC 4	CC 5	CC 6	CC 7	CC 8	CC 9
Value of register $2:	10	10	10	10	10/–20	–20	–20	–20	–20
Value of EX/MEM:	X	X	X	–20	X	X	X	X	X
Value of MEM/WB:	X	X	X	X	–20	X	X	X	X



DATA HAZARDS 7/17
Forwarding



a. No forwarding



b. With forwarding

DATA HAZARDS 8/17
Forwarding

Mux control	Source	Explanation
ForwardA = 00	ID/EX	The first ALU operand comes from the register file.
ForwardA = 10	EX/MEM	The first ALU operand is forwarded from the prior ALU result.
ForwardA = 01	MEM/WB	The first ALU operand is forwarded from data memory or an earlier ALU result.
ForwardB = 00	ID/EX	The second ALU operand comes from the register file.
ForwardB = 10	EX/MEM	The second ALU operand is forwarded from the prior ALU result.
ForwardB = 01	MEM/WB	The second ALU operand is forwarded from data memory or an earlier ALU result.

DATA HAZARDS 9/17
Condiciones para Forward

1. *EX 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
```

2. *MEM 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)) ForwardB = 01
```

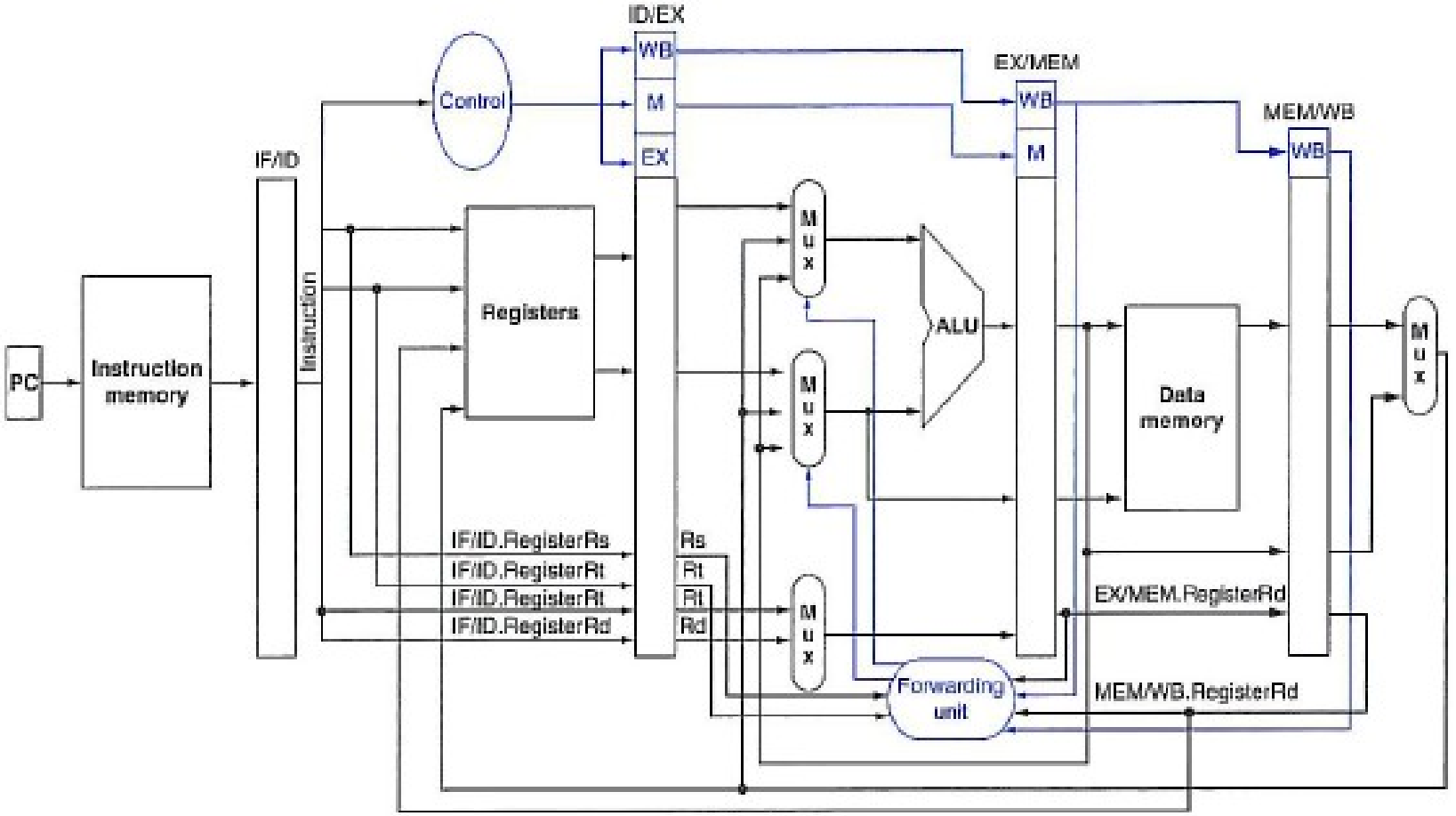
DATA HAZARDS 10/17
Condiciones para Forward

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

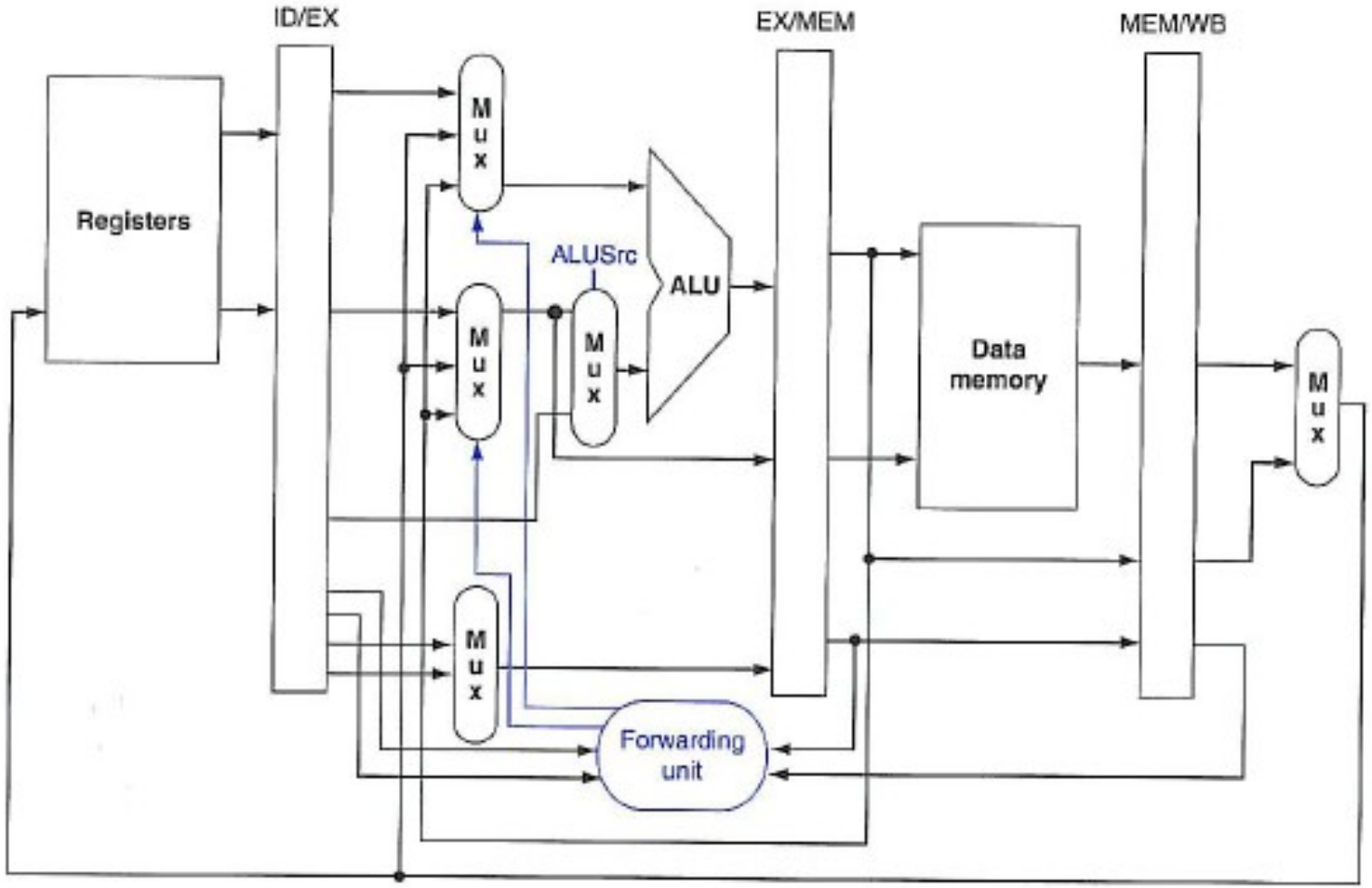
```
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd ≠ 0)
and not(EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
        and (EX/MEM.RegisterRd ≠ ID/EX.RegisterRs)
and (MEM/WB.RegisterRd = ID/EX.RegisterRs)) ForwardA = 01

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
```

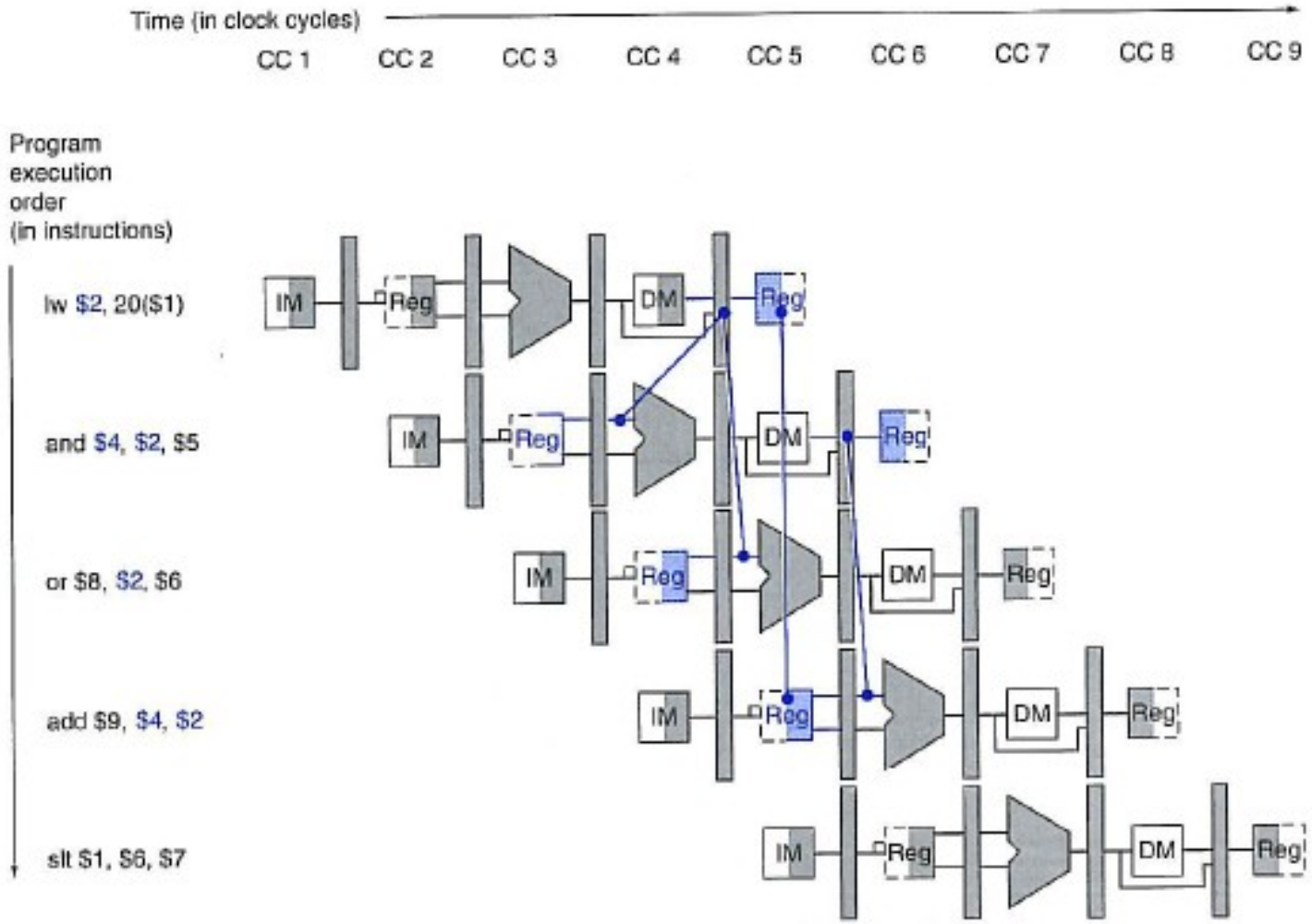
DATA HAZARDS 11/17
Condiciones para Forward



DATA HAZARDS 12/17
Condiciones para Forward



DATA HAZARDS 13/17
Stalls



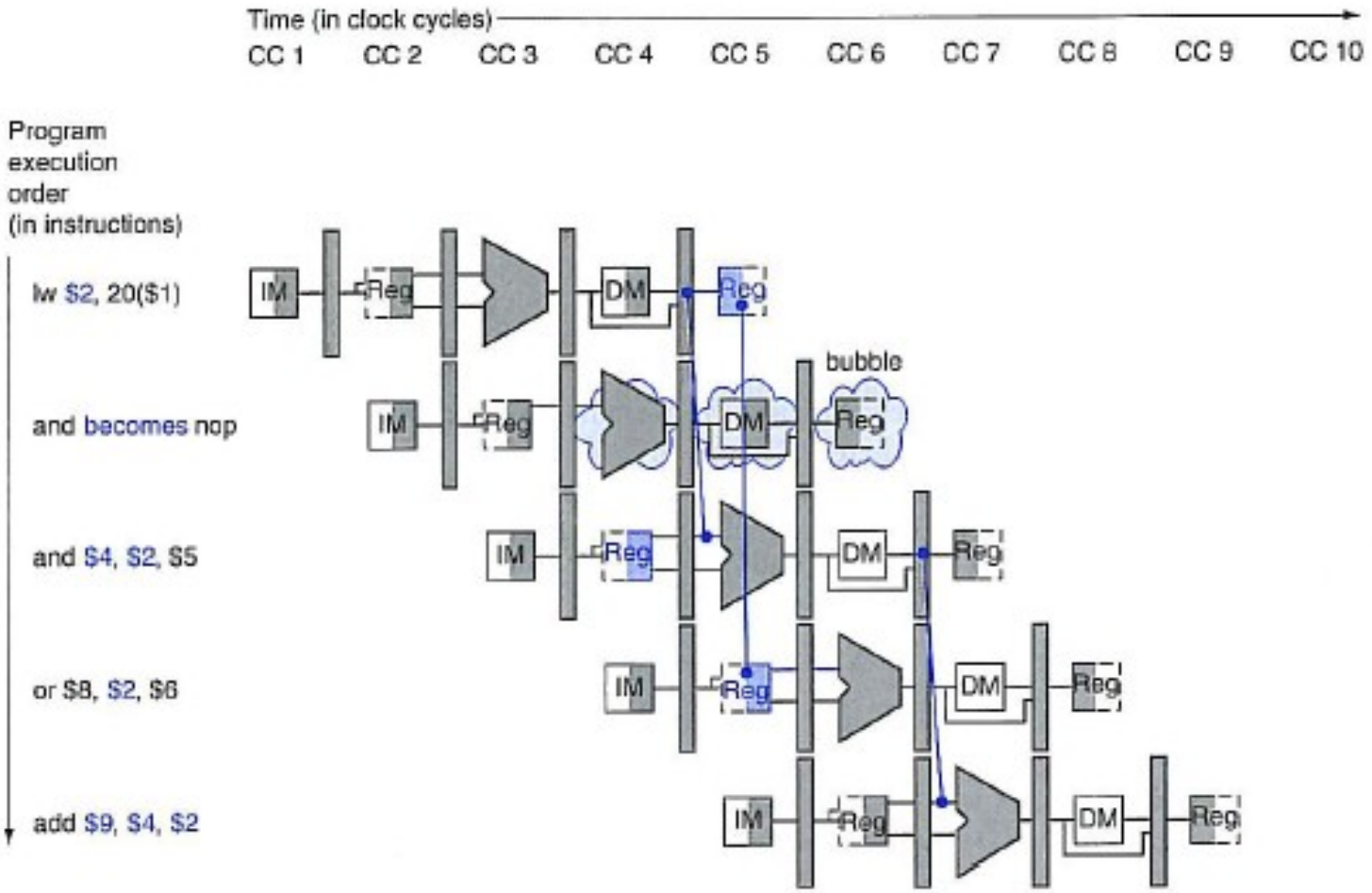
DATA HAZARDS 14/17
Stalls

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

DATA HAZARDS 15/17
Stalls

Instruction	Execution/address calculation stage control lines				Memory access stage control lines			Write-back stage control lines	
	RegDst	ALUOp1	ALUOp0	ALUSrc	Branch	Mem-Read	Mem-Write	Reg-Write	Memto-Reg
R-format	1	1	0	0	0	0	0	1	0
lw	0	0	0	1	0	1	0	1	1
sw	X	0	0	1	0	0	1	0	X
beq	X	0	1	0	1	0	0	0	X

DATA HAZARDS 16/17
Stalls



DATA HAZARDS 17/17
Stalls