Dato il frammento di codice assembler (riferito alla architettura MIPS semplificata vista a lezione) con condizioni iniziali mostrate alla sua sinistra e memorizzazione dei dati con big-endian:

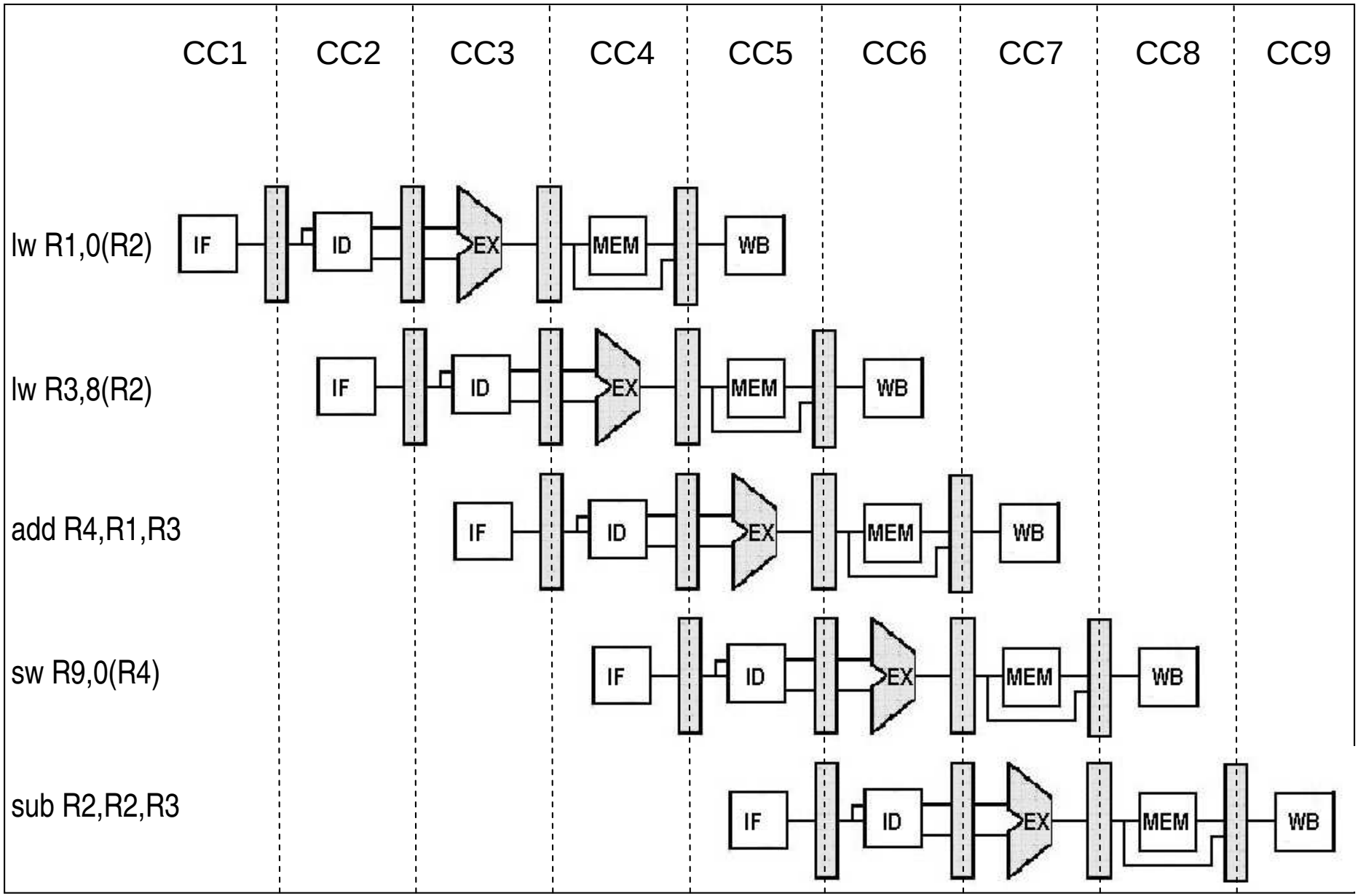
1. identificare le dipendenze dei dati per le prime 5 istruzioni e riportarle su un diagramma che esplicita le fasi della pipeline (vedi figura di seguito)
2. mostrare come evolve la pipeline durante l'esecuzione del codice per i primi 32 cicli di clock assumendo che la pipeline non possa effettuare data forwarding

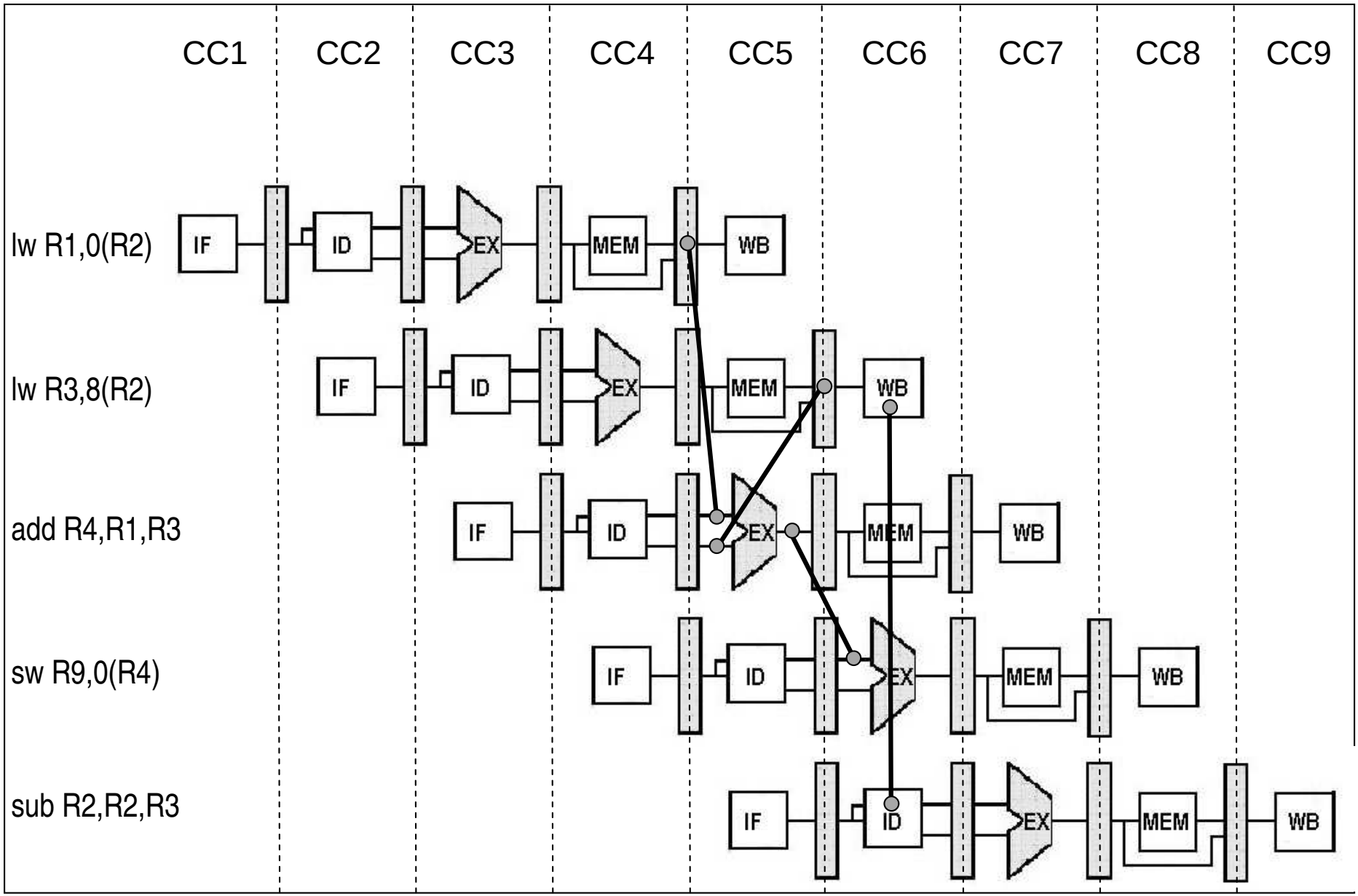
[R2] == 4
[R9] == 3

Memoria (contenuto rap. In esadecimale)

0	00	00	00	00
4	00	00	00	00
8	00	00	00	09
12	00	00	00	04

```
loop:  lw R1,0(R2)   !  R1 ← Mem[[R2] + 0]
       lw R3,8(R2)   !  R3 ← Mem[[R2] + 8]
       add R4,R1,R3  !  R4 ← [R1] + [R3]
       sw R9,0(R4)   !  Mem[[R4]+0] ← [R9]
       sub R2,R2,R3  !  R2 ← [R2] - [R3]
       bnez R2,loop  !  If [R2] != 0 goto loop
       add R7,R8,R9  !  R7 ← [R8] + [R9]
       add R8,R7,R2  !  R8 ← [R7] + [R2]
```


CC1
CC2
CC3
CC4
CC5
CC6
CC7
CC8
CC9
lw R1,0(R2)
lw R3,8(R2)
add R4,R1,R3
sw R9,0(R4)
sub R2,R2,R3
IF
ID
EX
MEM
WB


CC1
CC2
CC3
CC4
CC5
CC6
CC7
CC8
CC9
lw R1,0(R2)
lw R3,8(R2)
add R4,R1,R3
sw R9,0(R4)
sub R2,R2,R3
IF
ID
EX
MEM
WB