Esercizio 1

a) Nel caso del singolo ciclo il tempo di esecuzione di una istruzione corrisponde a quello della lw (istruzione più lunga) e risulta quindi

$T_{SC} = T_{fetch} + T_{read\ registri} + T_{ALU} + T_{mem} + T_{write\ registro} = (2+1+2+2+1)\ ns = 8\ ns$

Nel caso del multiciclo, si ha:

- $T_{clock} = 2$ ns
- $T_{MC} = CPI * T_{clock} = [5f_{lw} + 4(f_{sw} + f_{TIPOR}) + 3(f_{beq} + f_J)]*2$ ns
  e tenendo conto del fatto che $f_{sw} + f_{TIPOR} = 0.5$
  e quindi anche che $(f_{beq} + f_J) = 0.5 – f_{lw}$
  risulta $T_{MC} = 4f_{lw} + 7$ ns

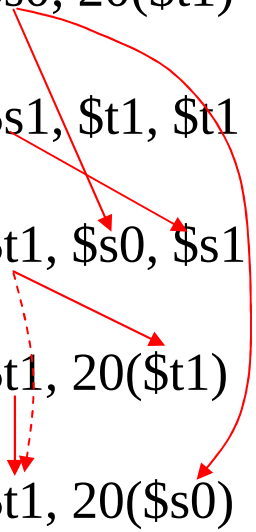
Quindi, il singolo ciclo risulta più conveniente se $T_{SC} < T_{MC}$ ovvero se $8 < 4f_{lw} + 7$, vale a dire se $f_{lw} > 25\%$ [infatti lw "privilegia" singolo ciclo]

b) In questo caso tutti gli stadi sono perfettamente bilanciati, di conseguenza il multiciclo non potrà essere migliore del singolo ciclo (potrebbe avere le stesse prestazioni solo se eseguissimo sempre la lw, ma ciò non è il caso).
Infatti, rispetto al primo punto cambia solo $T_{SC} = 10$ ns, da cui si ottiene $T_{SC} < T_{MC}$ se e solo se $f_{lw} > 75\%$, ma questo è impossibile dato che $f_{sw} + f_{TIPOR} = 0.5$
Quindi, il singolo ciclo non può essere mai migliore del multiciclo.

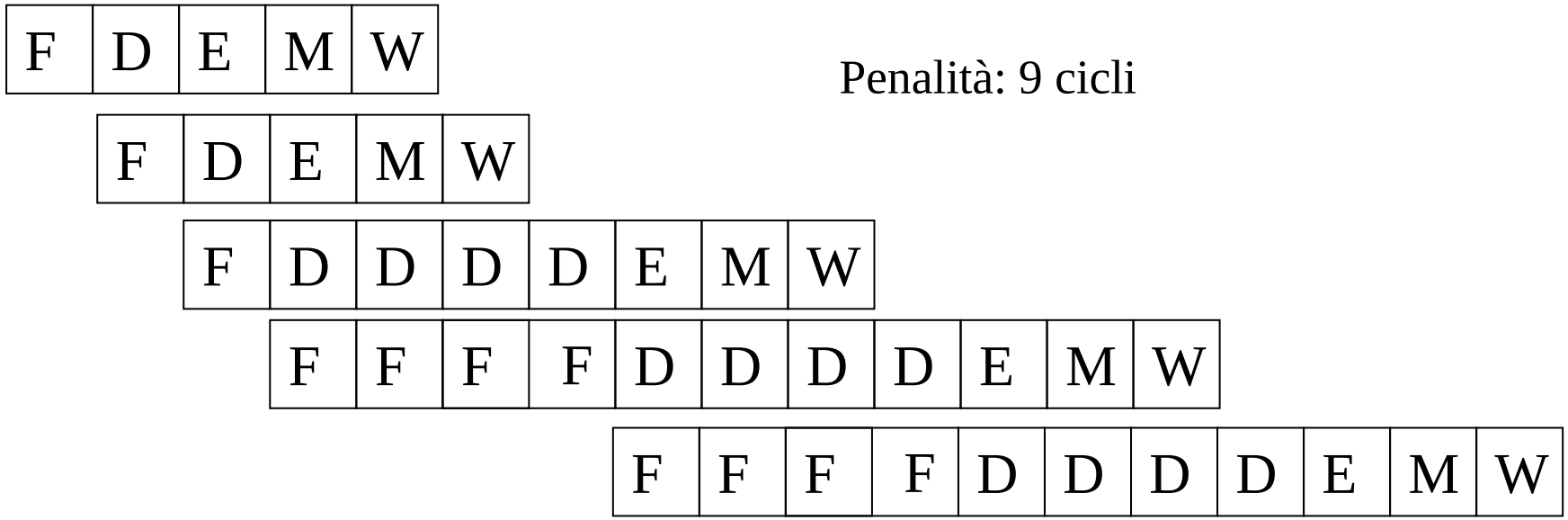
Esercizio 2

a) Dipendenze:

```
lw    $s0, 20($t1)
sub   $s1, $t1, $t1
add   $t1, $s0, $s1
lw    $t1, 20($t1)
sw    $t1, 20($s0)
```

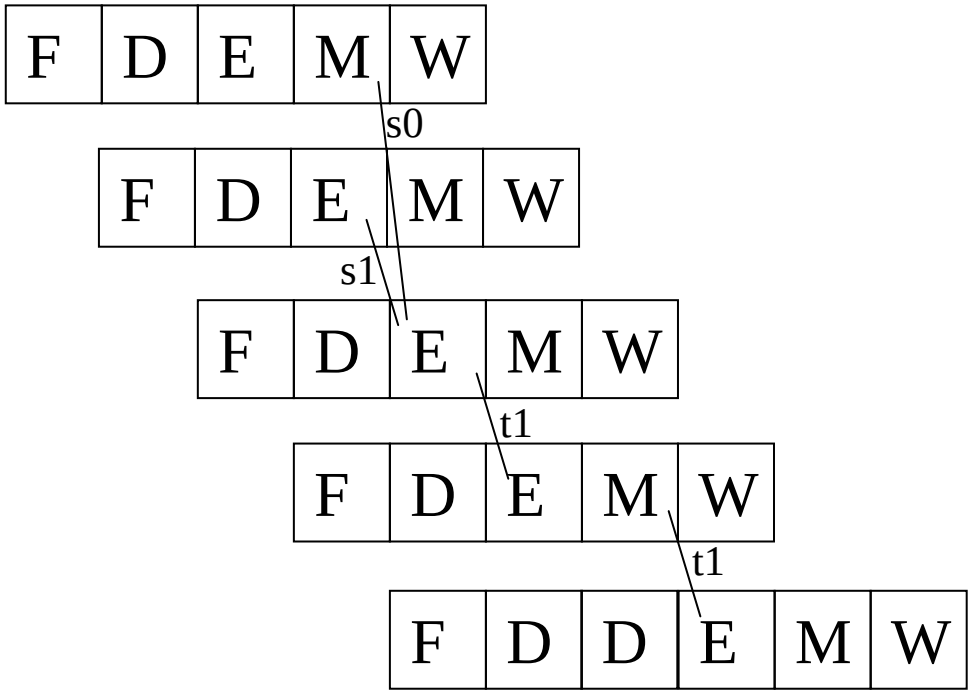


b1) No unità di propagazione:



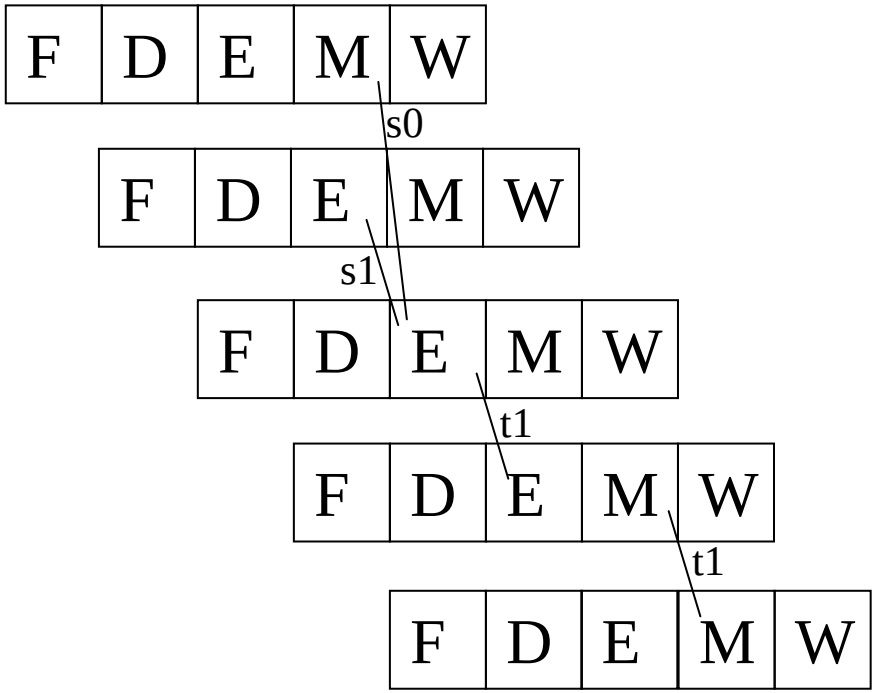
Penalità: 9 cicli

b2) Unità di propagazione verso E:



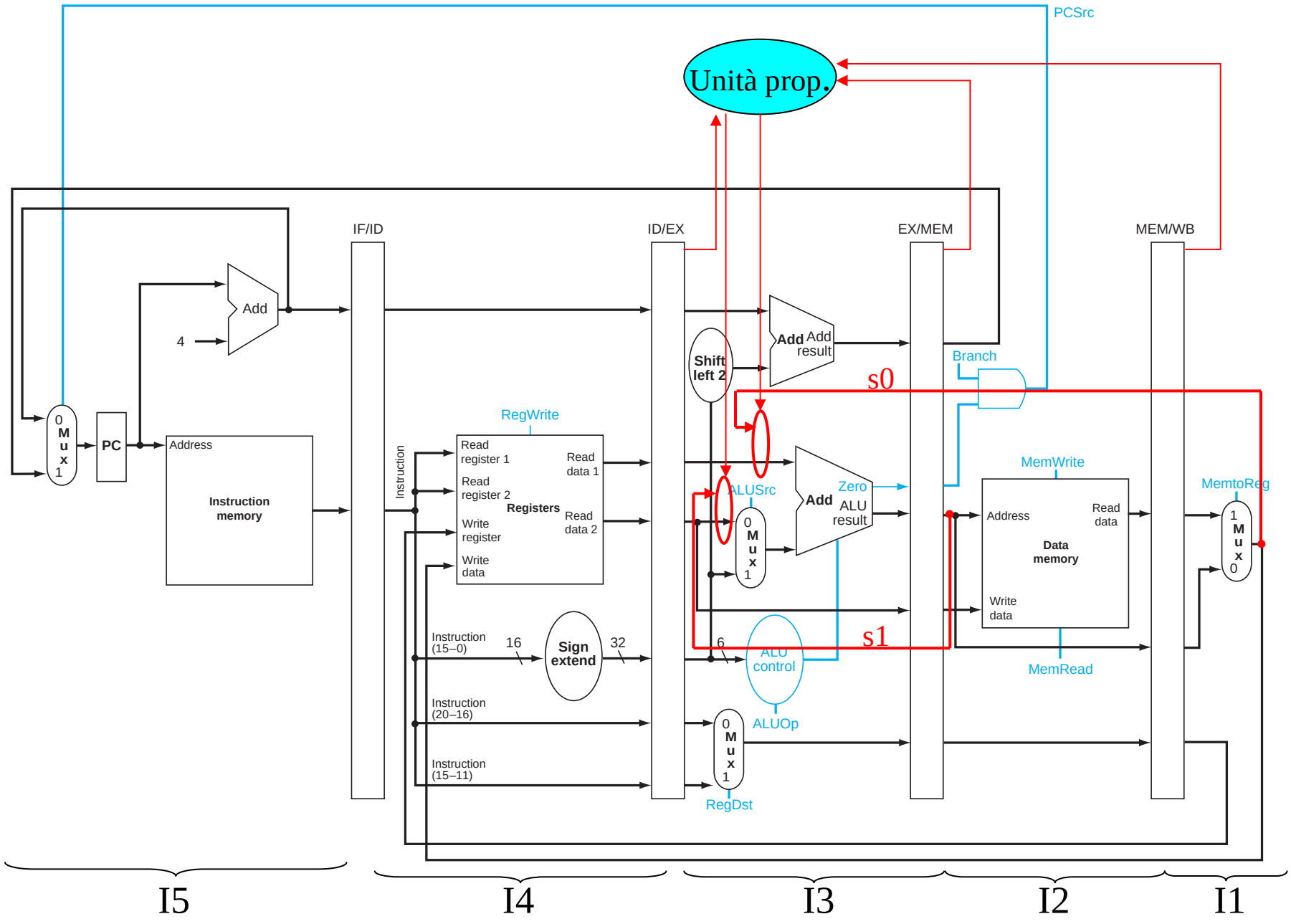
Penalità: 1 ciclo

b3) Unità di propagazione verso E e verso M:



Nessuna penalità

Esercizio 3


PCSrc
Unità prop.
IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
M u x
0
1
PC
Address
Instruction memory
Instruction
RegWrite
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Shift left 2
Add
Add result
Branch
s0
ALUSrc
Zero
ALU result
MemWrite
MemtoReg
Address
Data memory
Read data
Write data
MemRead
Instruction (15–0)
16
Sign extend
32
6
ALU control
s1
ALUOp
Instruction (20–16)
Instruction (15–11)
RegDst
I5
I4
I3
I2
I1

Esercizio 4

$T_{clock} = 2$ ns

$CPI = 1 + \delta_{dip}$

$\Big\}\ T_{esecuzione} = (1 + \delta_{dip}) * 2$ ns

Caso a)

$\delta_{dip} = 0.2* (0.2 + 0.1 + 0.15)*1 = 0.09$

(criticità carica e usa: lw seguita da istruzioni che ne utilizzano il risultato in E o in M – ovvero TIPO-R ed entrambe le sw – dato che è presente unità di propagazione solo verso E)

e risulta quindi $T_{esecuzione} = 2.18$ ns

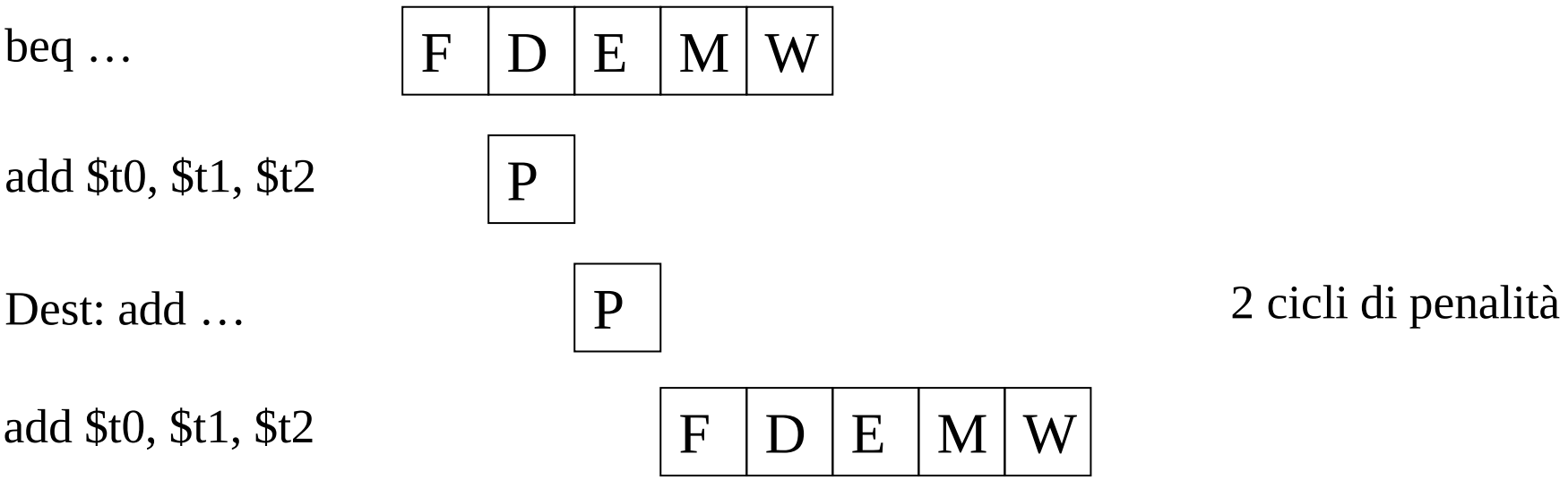
Caso b)

$\delta_{dip} = 0.2* (0.2 + 0.1)*1 = 0.06$

(lw seguita da istruzioni che ne utilizzano il risultato in E – ovvero TIPO-R e sw per il calcolo dell'indirizzo – dato che è presente unità di propagazione verso E e verso M)

e risulta quindi $T_{esecuzione} = 2.12$ ns

Esercizio 5



NB: la beq in F tramite BPB decide di saltare, ma è necessario aspettare il calcolo dell'indirizzo di destinazione: quindi si impedisce nel ciclo di clock successivo il caricamento dell'istruzione seguente.
Al terzo ciclo l'istruzione di destinazione viene caricata, tuttavia E della beq calcola la condizione (non salto) e si riparte dall'istruzione successiva alla beq (Purge della fase di fetch dell'istruzione di destinazione).

Esercizio 6

- TLB va invalidato
  (pagine virtuali di processi distinti in generale possono essere mappate in pagine fisiche distinte)
- Cache non deve essere invalidata
  (si riferisce esclusivamente agli indirizzi fisici i cui valori corrispondenti sono ancora validi visto che la RAM non è stata modificata
  - invalidare la cache sarebbe inefficiente, perché è possibile che indirizzi fisici presenti in essa siano riferiti da nuovi processi)