Architettura degli Elaboratori

L'architettura della CPU – Esercizio su hazard e pipeline

SAPIENZA
UNIVERSITÀ DI ROMA

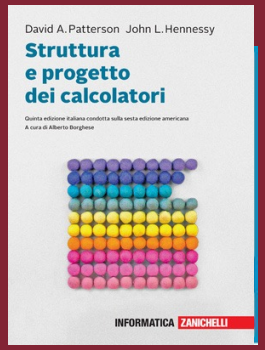
Alessandro Checco

checco@di.uniroma1.it

Special thanks and credits:

Andrea Sterbini, Iacopo Masi,
Claudio di Ciccio

[S&PdC]
4.5 - 4.7



Esercizio

Esercizio (esame 16-9-14)

1. Individuare i **data** e **control hazard**
   - assumendo la **decisione** dei **branch** in fase **ID**
2. Individuare i casi in cui il **forwarding è applicabile**
3. Individuare gli **stalli necessari**
4. Calcolare la **durata totale** in cicli di clock **con forwarding**
5. Calcolare la **durata senza forwarding**
6. **Riordinare le istruzioni** per ridurre gli stalli (con fwd.)
7. Calcolare la **durata** del programma **ottimizzato** (**4.**)
8. Cosa c'è in **pipeline** al **16° CC** (v. programma originale)?

Nota: Assumiamo che
`bge $t1, $zero, ciclo`
sia un'istruzione del set standard. In realtà, è una pseudoistruzione tradotta in
`slt $1, $9, $0`
`beq $1, $0, <offset istr. ciclo>`

```
.data
vettore: .word 7, 14, … , 48, 91 # 70 pari, 30 dispari
.text
main:
     li   $t0, 0              # somma parziale
     li   $t1, 396            # offset ultimo
ciclo:
     lw   $t3, vettore($t1)
     andi $t2, $t3, 1         # è dispari?
     beqz $t2, salta          # se non lo è, lo ignoro
     add  $t0, $t0, $t3       # altrimenti, lo sommo
salta:
     addi $t1, $t1, -4        # prossimo elemento
     bge  $t1, $zero, ciclo   # fine del ciclo?
     li   $v0, 1              # stampa…
     move $a0, $t0            # … la somma
     syscall
```

Hazard, stalli, durata con forwarding (punti 1., 2., 3., 4.)

```
.data
vettore: .word 7, 14, … , 48, 91 # Tot. 70 pari, 30 dispari
.text
main:
      li   $t0, 0                # F D E M W @5°
      li   $t1, 396              #   F D E M W @6°
ciclo:
      lw   $t3, vettore($t1)     #     F D E M W @7°      F D E M W @17°
      andi $t2, $t3, 1           #       » F D E M W @9°    » F D E M W @19°
      beqz $t2, salta            #           » F D E M W @11°   » F D E M W @21°
      add  $t0, $t0, $t3         #               F D E M W @12°
salta:
      addi $t1, $t1, -4          #                 F D E M W @13°   » F D E M W @23°
      bge  $t1, $zero, ciclo     #                   » F D E M W @15°   » F D E M W @25°
      li   $v0, 1                #                                                  @1006°
      move $a0, $t0              # ciclo = 30*(6+3+1) + 70*(5+3+2) - 1              @1007°
      syscall                    # totale = 4 + 2 + 999 + 3 = 1008                  @1008°
```

4 (riempimento)

10 colpi di clock (dispari)

10 colpi di clock (pari)

Hazard, stalli, durata senza forwarding (punto 5.)

```
.data
vettore: .word 7, 14, … , 48, 91 # Tot. 70 pari, 30 dispari
.text
main:
      li   $t0, 0                 # F D E M W @5°
      li   $t1, 396               #   F D E M W @6°
ciclo:
      lw   $t3, vettore($t1)      #     » » F D E M W @9°          F D E M W @22°
      andi $t2, $t3, 1            #           » » F D E M W @12°      » » F D E M W @25°
      beqz $t2, salta             #                 » » F D E M W @15°      …      @28°
      add  $t0, $t0, $t3          #                       F D E M W @16°
salta:
      addi $t1, $t1, -4           #                         F D E M W @17°          @30°
      bge  $t1, $zero, ciclo      #                           » » F D E M W @20°    @33°
      li   $v0, 1                 #                                                 @1308°
      move $a0, $t0               # ciclo = 30*(6+6+1) + 70*(5+6+2) - 1             @1309°
      syscall                     # totale = 4 + 2 + 2 + 1299 + 3 = 1310            @1310°
```

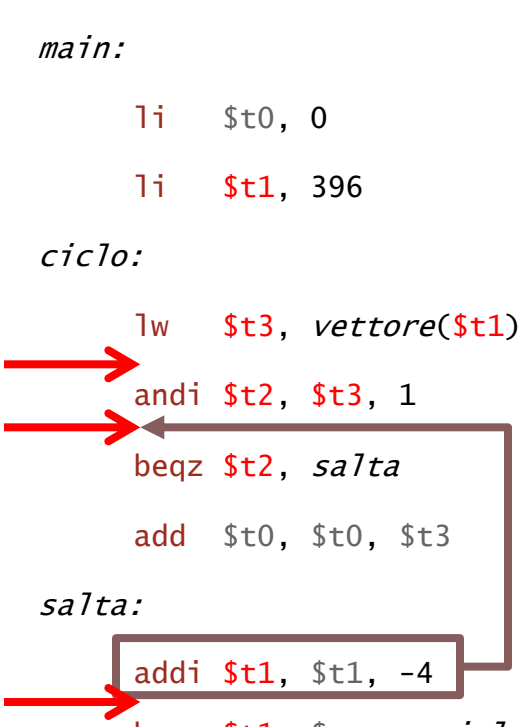
4 (riempimento)

13 colpi di clock (dispari)

13 colpi di clock (pari)

Riordinamento con forwarding (punto 6.)

```
.data
vettore: .word 7, 14, … , 48, 91 # Tot. 70 pari, 30 dispari
.text
main:
      li   $t0, 0                 # F D E M W @5°
      li   $t1, 396               #   F D E M W @6°
ciclo:
      lw   $t3, vettore($t1)      #     F D E M W @7°       F D E M W @17°
      andi $t2, $t3, 1            #       » F D E M W @9°     » F D E M W @19°
      beqz $t2, salta             #           » F D E M W @11°    » F D E M W @21°
      add  $t0, $t0, $t3          #               F D E M W @12°
salta:
      addi $t1, $t1, -4           #                 F D E M W @13°    » F D E M W @23°
      bge  $t1, $zero, ciclo      #                   » F D E M W @15°  » F D E M W @25°
      li   $v0, 1                 #                                                    @1006°
      move $a0, $t0               # ciclo = 30*(6+3+1) + 70*(5+3+2) - 1                @1007°
      syscall                     # totale = 4 + 2 + 999 + 3 = 1008                    @1008°
```



Riordinamento con forwarding (punto 6.)

2 stalli sono stati **eliminati** all'interno di un **ciclo ripetuto 100** volte → **200 cicli di clock** necessari **in meno** → **risparmio** sul totale di **~20%**

```
.data
vettore: .word 7, 14, … , 48, 91 # Tot. 70 pari, 30 dispari
.text
main:                                          4 (riempimento)
      li   $t0, 0                 # F D E M W @5°
      li   $t1, 396               #   F D E M W @6°
ciclo:                                                  8 colpi di clock (dispari)
      lw   $t3, vettore($t1)      #     F D E M W @7°        F D E M W @15°
      andi $t2, $t3, 1            #       » F D E M W @9°      » F D E M W @17°
      addi $t1, $t1, -4           #           F D E M W @10°       F D E M W @18°
      beqz $t2, salta             #             F D E M W @11°       F D E M W @19°
      add  $t0, $t0, $t3          #               F D E M W @12°
salta:
      bge  $t1, $zero, ciclo      #                 F D E M W @13°     F D E M W @20°
      li   $v0, 1                 #                                                    @806°
      move $a0, $t0               # ciclo = 30*(6+1+1) + 70*(5+1+2) - 1                @807°
      syscall                     # totale = 4 + 2 + 799 + 3 = 808                     @808°
```

Non rimovibile mediante rioordinam.

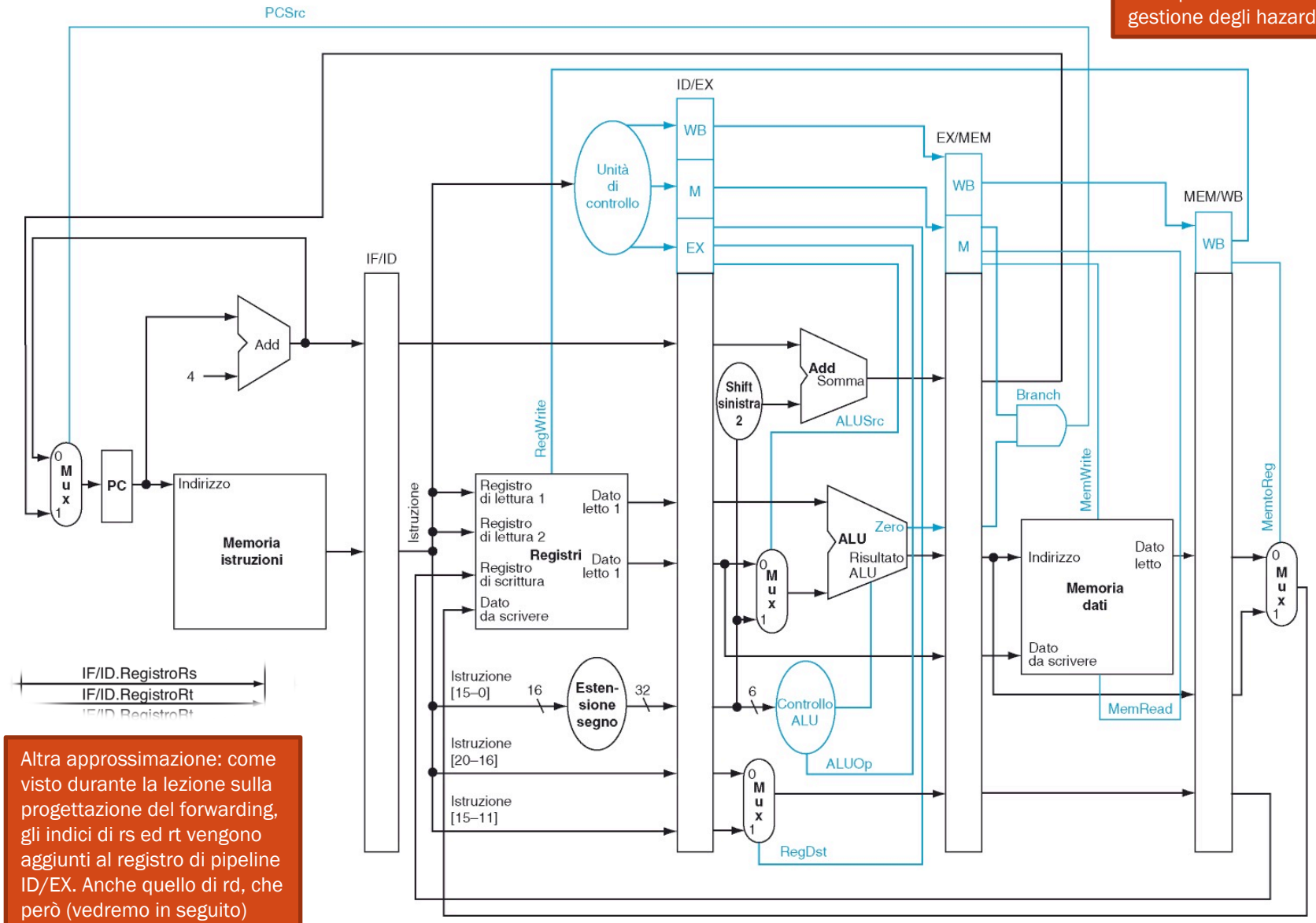
8 colpi di clock (pari)

Cosa c'è in pipeline a t=16? (punto 7.)

```
.data
vettore: .word 7, 14, … , 48, 91 # Tot. 70 pari, 30 dispari
.text
main:                          # 1 2 3 4 5 6 7 8 9 A B C D E F
     li   $t0, 0               # F D E M W @5°
     li   $t1, 396             #   F D E M W @6°
ciclo:                                                    E
     lw   $t3, vettore($t1)    #     F D E M W @7°       F D E M W @17°
     andi $t2, $t3, 1          #       » F D E M W @9°     » F D E M W @19°
     beqz $t2, salta           #           » F D E M W @11°  » F D E M W @21°
     add  $t0, $t0, $t3        #               F D E M W @12°  W
salta:
     addi $t1, $t1, -4         #                 F D E M W @13°  » F D E M W @23°
     bge  $t1, $zero, ciclo    #                   » F D E M W @15°  » F D E M W @25°
     li   $v0, 1               #                                                  @1006°
     move $a0, $t0             # ciclo = 30*(6+3+1) + 70*(5+3+2) - 1              @1007°
     syscall                   # totale = 4 + 2 + 999 + 3 = 1008                  @1008°
```

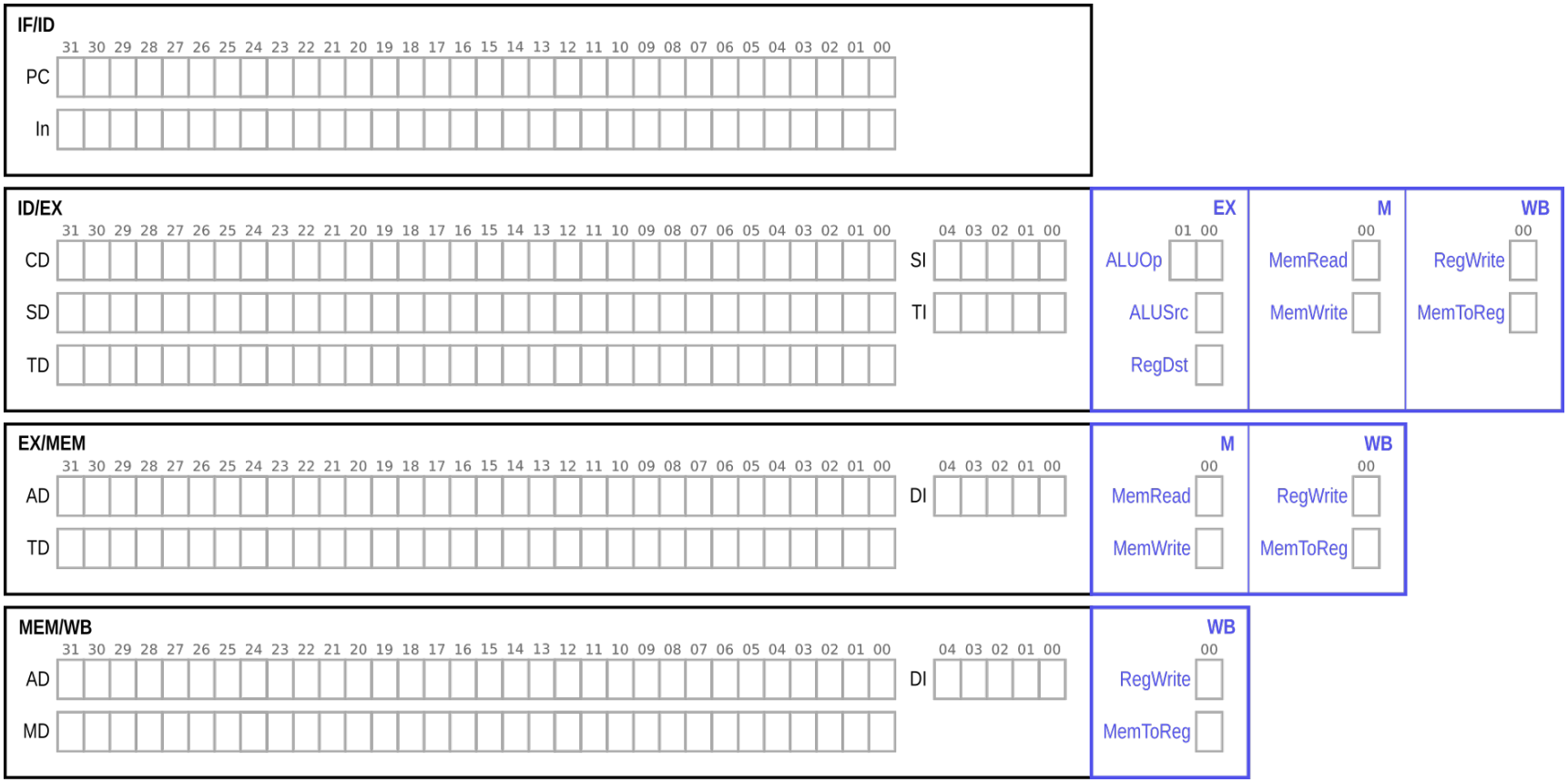
Architettura con pipeline (attenzione: incompleta!)

Fra le altre cose, questo schema non presenta la fase di decisione dei salti in ID e non riporta le strutture di gestione degli hazard.

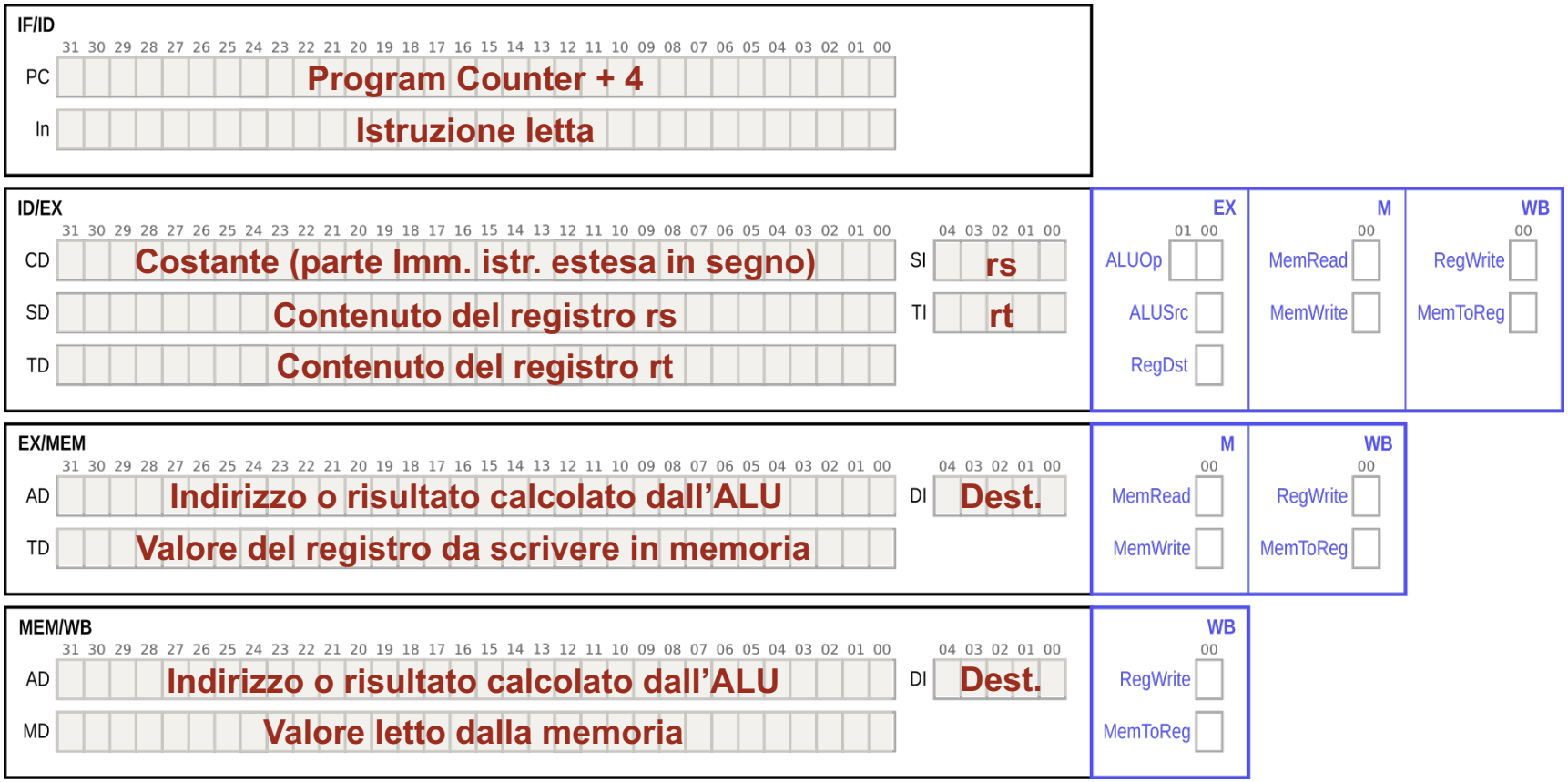


Altra approssimazione: come visto durante la lezione sulla progettazione del forwarding, gli indici di rs ed rt vengono aggiunti al registro di pipeline ID/EX. Anche quello di rd, che però (vedremo in seguito) resta comunque incluso nella parte imm. di 32 bit

Contenuto dei registri

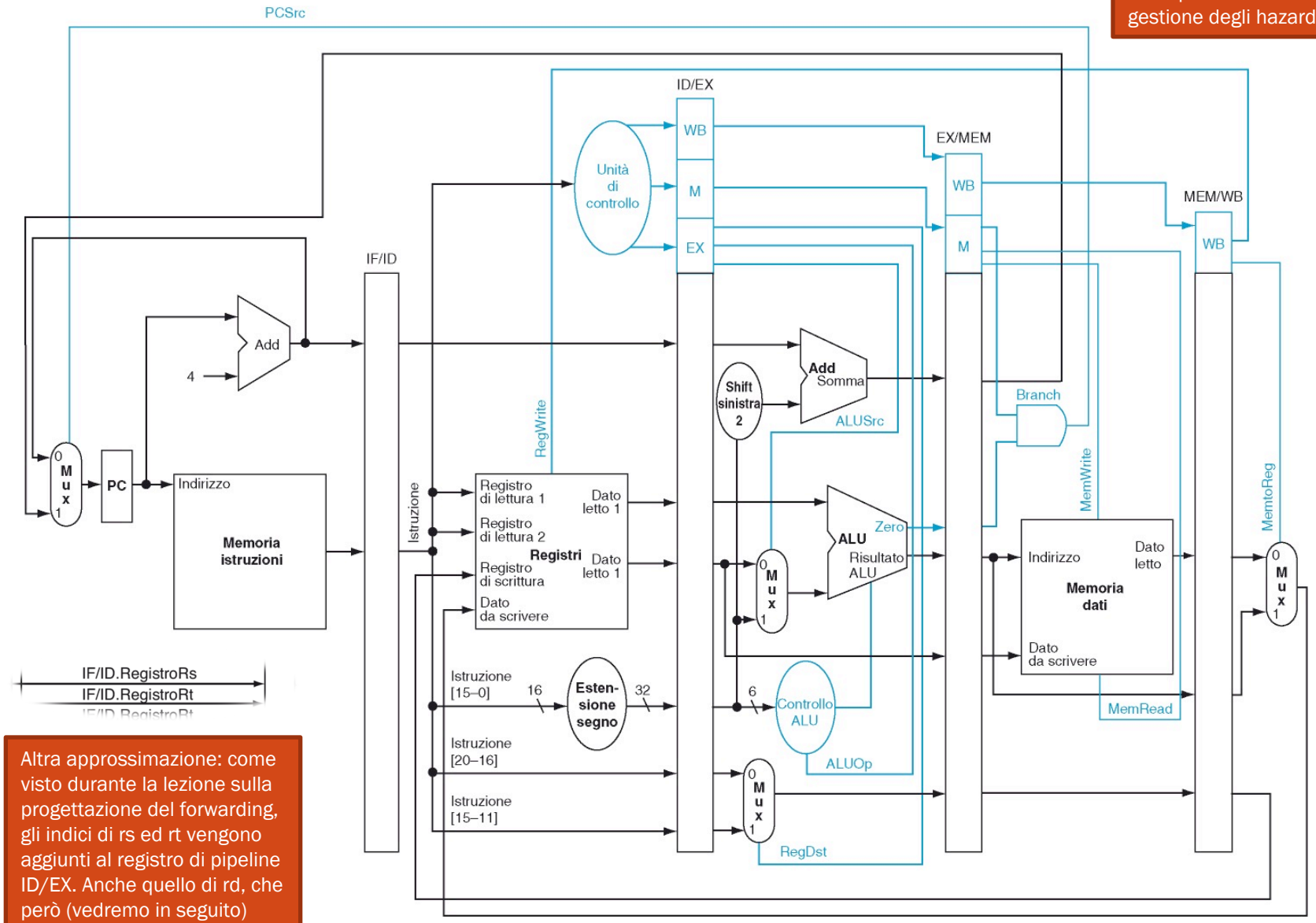


Contenuto dei registri



Architettura con pipeline (attenzione: incompleta!)

Fra le altre cose, questo schema non presenta la fase di decisione dei salti in ID e non riporta le strutture di gestione degli hazard.



Altra approssimazione: come visto durante la lezione sulla progettazione del forwarding, gli indici di rs ed rt vengono aggiunti al registro di pipeline ID/EX. Anche quello di rd, che però (vedremo in seguito) resta comunque incluso nella parte imm. di 32 bit

Contenuto dei registri

IF/ID

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

PC	Indirizzo della `beqz $t2, salta` (andrà in stallo)
In	Istruzione `beqz $t2, salta` (andrà in stallo)

ID/EX

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

CD	0000...0001 (parte imm. di `andi $t2, $t3, 1`)	SI	0xB
SD	Contenuto del registro `$t3`	TI	0xA
TD	Contenuto del registro `$t2`		

EX (01 00)	M (00)	WB (00)
ALUOp 0 0	MemRead 0	RegWrite 1
ALUSrc 1	MemWrite 0	MemToReg 0
RegDst 0		

EX/MEM

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

AD	(stallo della `andi $t2, $t3, 1`)	DI	0xA
TD	(stallo della `andi $t2, $t3, 1`)		

M (00)	WB (00)
MemRead 0	RegWrite 0
MemWrite 0	MemToReg 0

MEM/WB

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

AD	Indirizzo pari a `vettore` + 0x188	DI	0xB
MD	Valore letto da `lw $t3, vettore($t1)`		

WB (00)
RegWrite 1
MemToReg 1

```
lo:
   lw   $t3, vettore($t1)      #     F D E M W @7°        F D E M W @17
   andi $t2, $t3, 1            #       » F D E M W @9°      » F D E M W
   beqz $t2, salta             #           » F D E M W @11°   » F D E
   add  $t0, $t0, $t3          #               F D E M W @12°   W
```

Contenuto dei registri