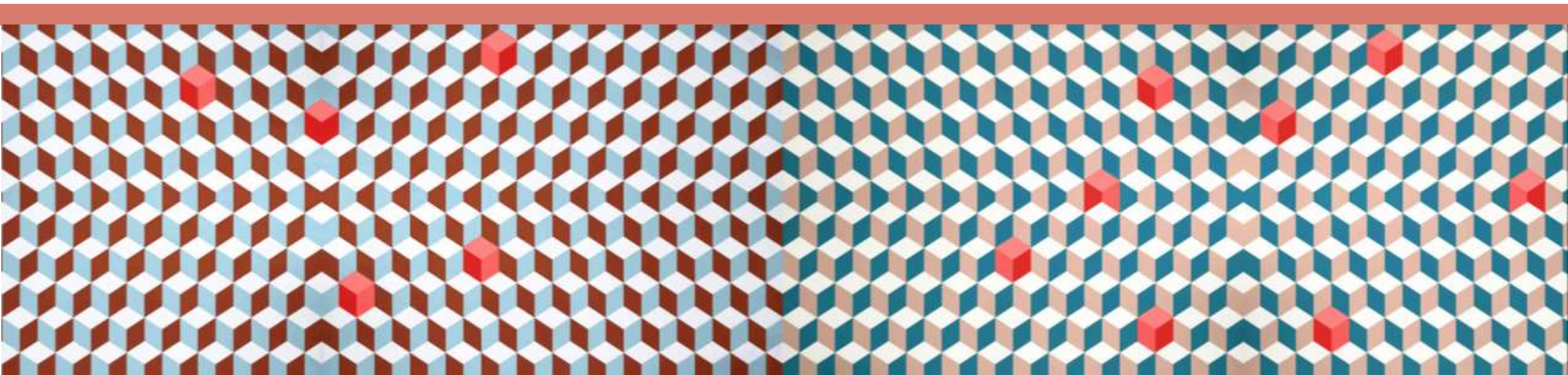


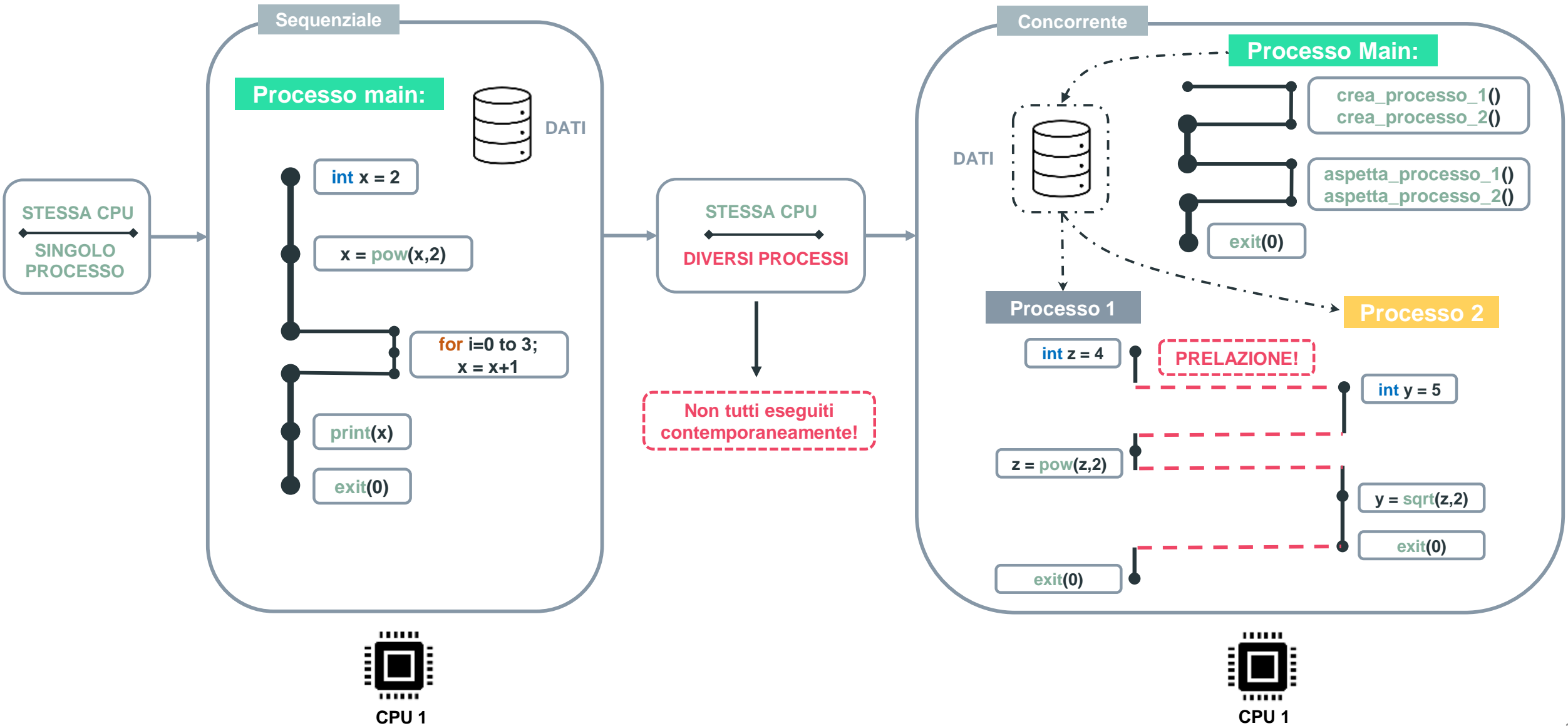
# Sincronizzazione:

Esercitazione e  
concetti

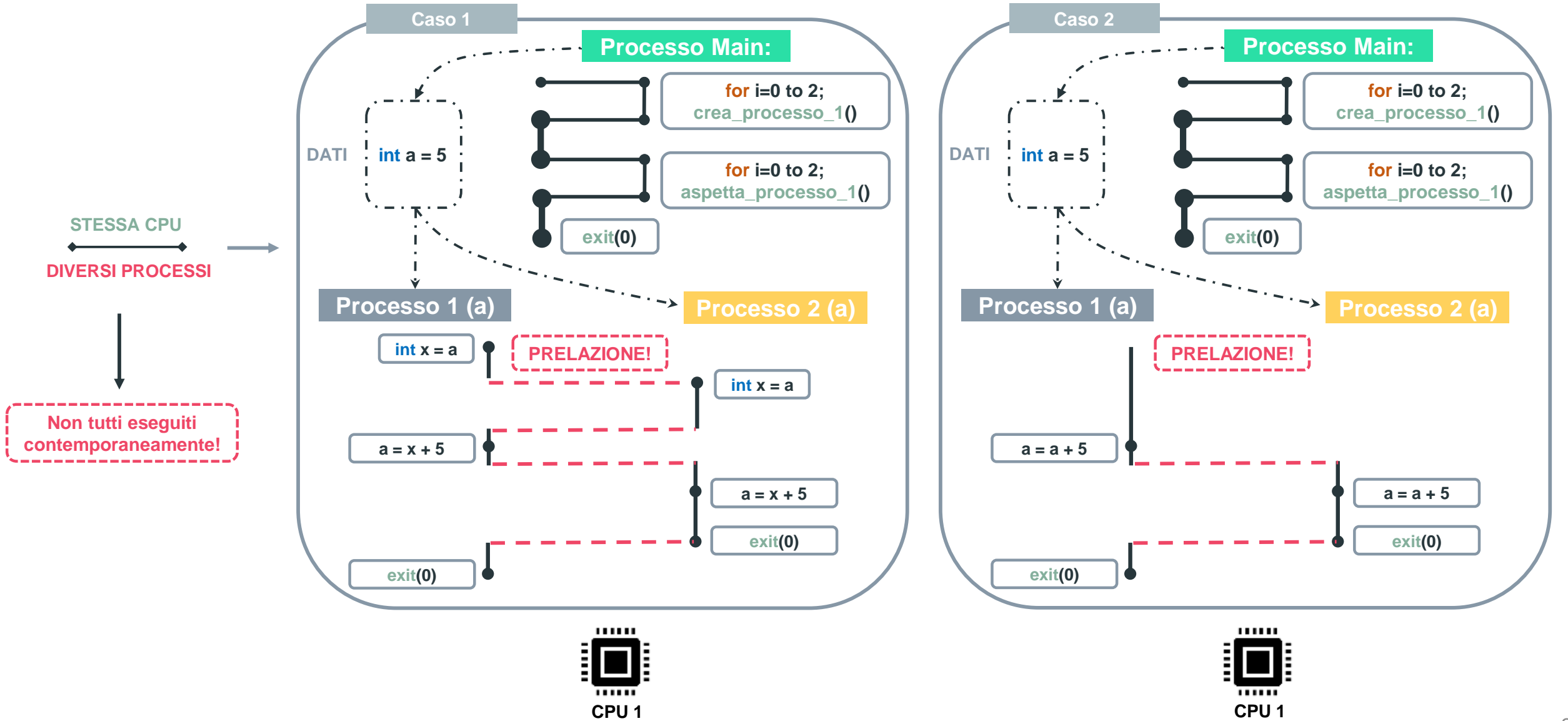
Tutor: Giovanni Hauber



# Sequenziale e Concorrente



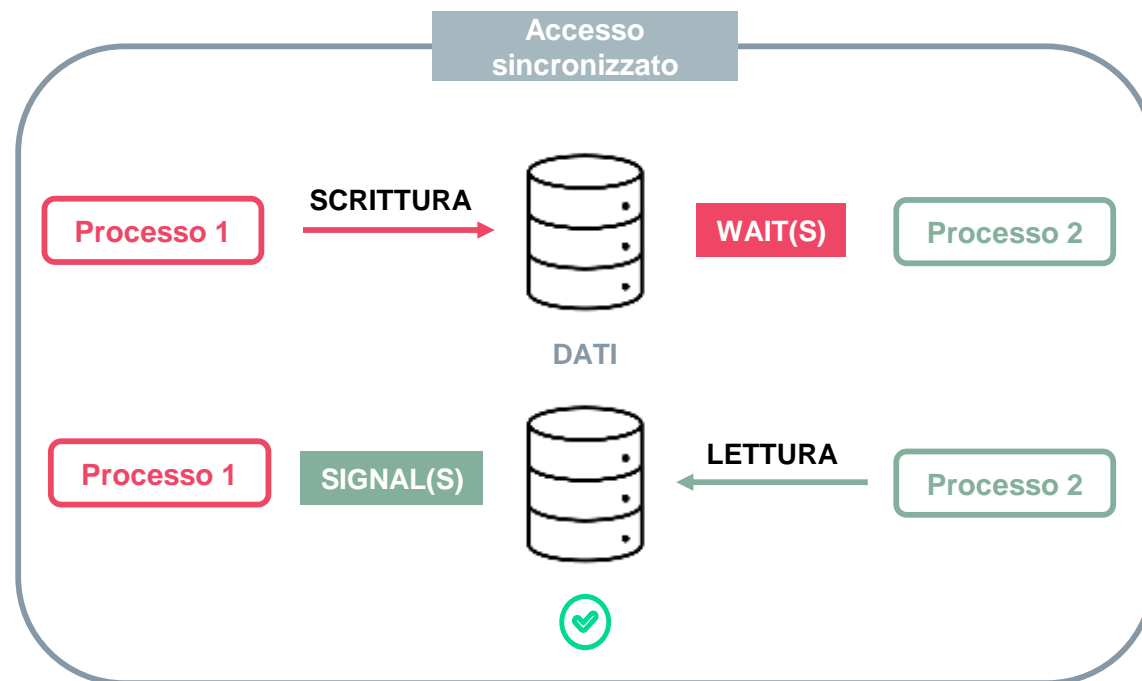
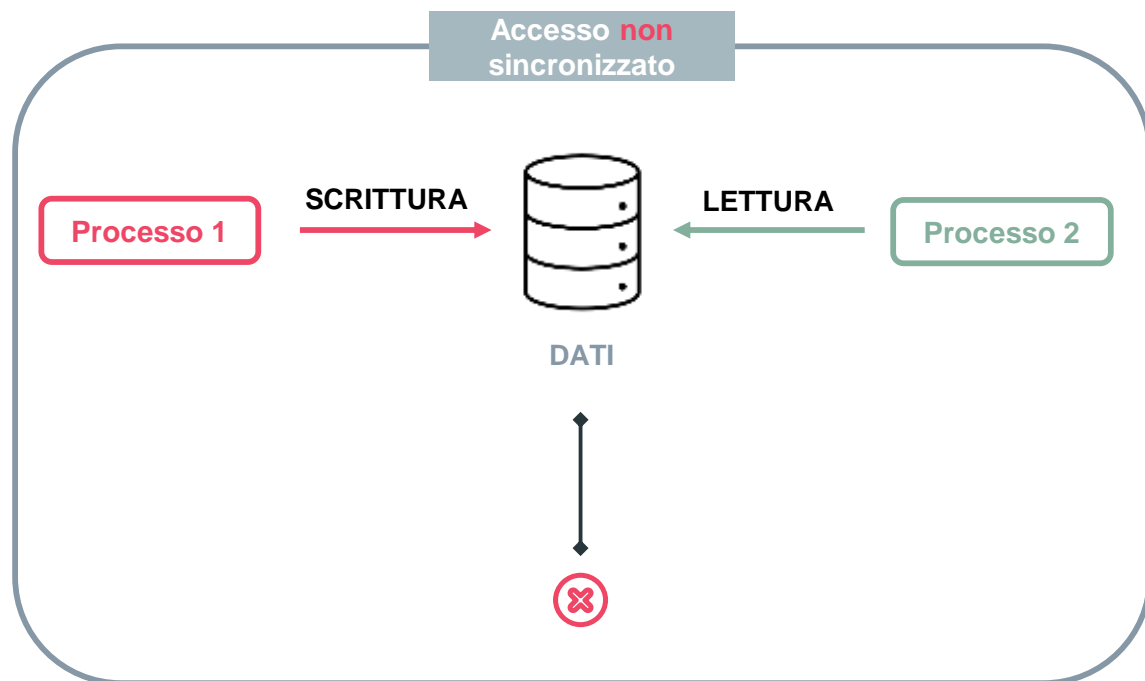
# Race Condition



# Ma cos'è la sincronizzazione?

« La **sincronizzazione** dei processi è il task che ha come scopo quello di **coordinare l'accesso ai dati condivisi** in modo tale che non ci siano due processi che, contemporaneamente, effettuino l'accesso alle risorse potendo generare eventuali inconsistenze. L'obiettivo è quindi quello di garantire l'accesso **in mutua esclusione**.

»



# Semafori, Mutex e variabili di condizione

## MUTEX

Usato per l'**accesso esclusivo** alle risorse

Un **Mutex** può essere **rilasciato solo dal processo che lo ha acquisito**

## SEMAFORO

Usato per la sincronizzazione delle attività, e si dividono in: **Binario/Contatore!**

Un **Semaforo** può essere **preso e rilasciato da qualsiasi processo**

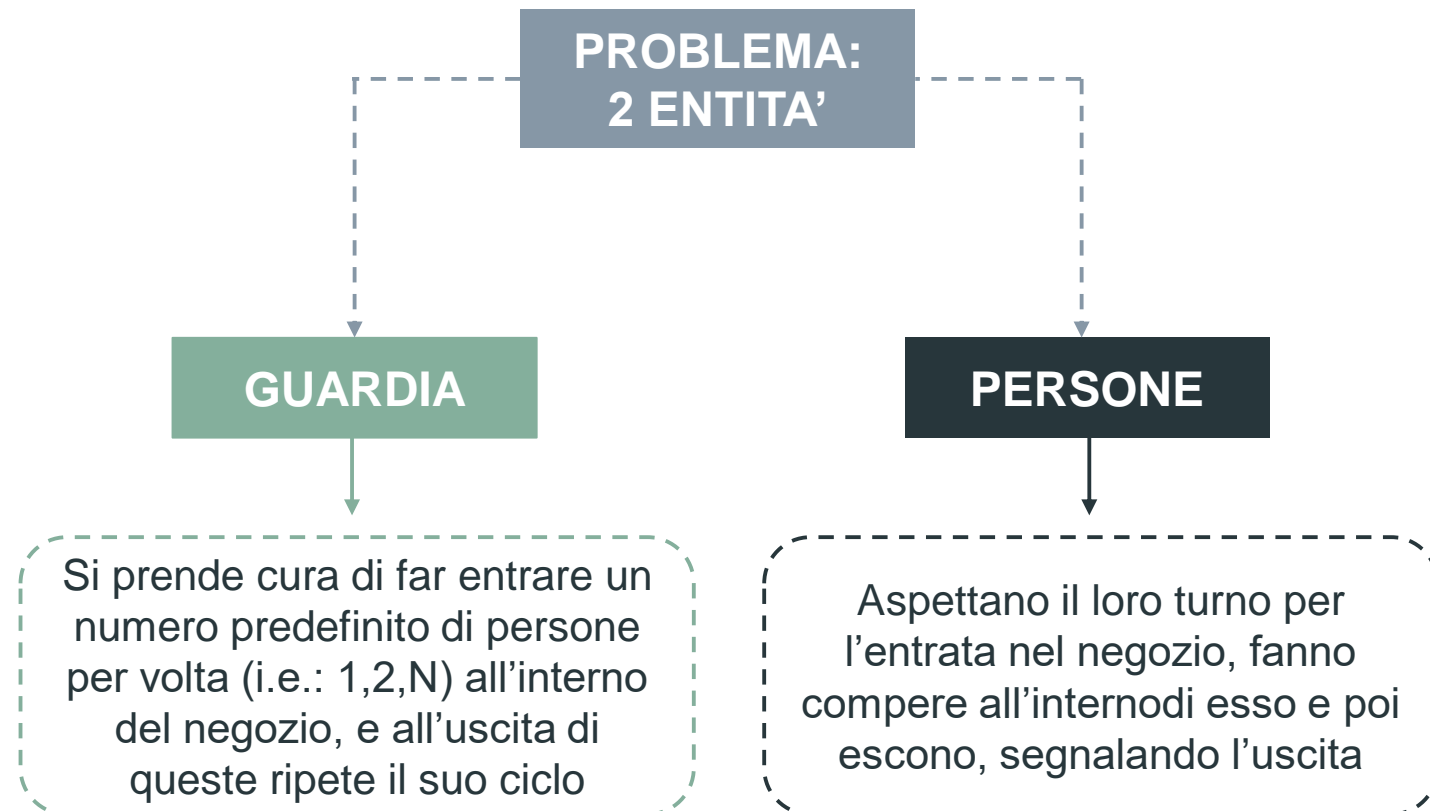
## VAR. COND.

Usata per **risvegliare più processi in attesa** di uno **stesso evento**

Come i Semafori; solitamente un **singolo processo si prende cura del risveglio**



# Fila ad un negozio



# L Possibile Soluzione

```
guardia() {
```

```
  repeat
```

```
    // Aspetta che arrivino persone
    wait(persone_in_fila)
```

```
    // Se ci sono posti liberi nel negozio
    wait(persone_ammesse)
```

```
    // Fai entrare una persona
    signal(turno)
```

```
  forever
```

```
}
```

```
semaforo contatore: persone_ammesse=N
semaforo contatore: persone_in_fila=0
semaforo contatore: turno=0
```

```
processo_main() {
```

```
  crea_processo(guardia)
```

```
  for i=0 to K:
```

```
    crea_processo(persona)
```

```
    // aspettare che i processi finiscano
```

```
}
```

```
persona() {
```

```
  // Aggiungiti alla coda delle persone
  signal(persone_in_fila)
```

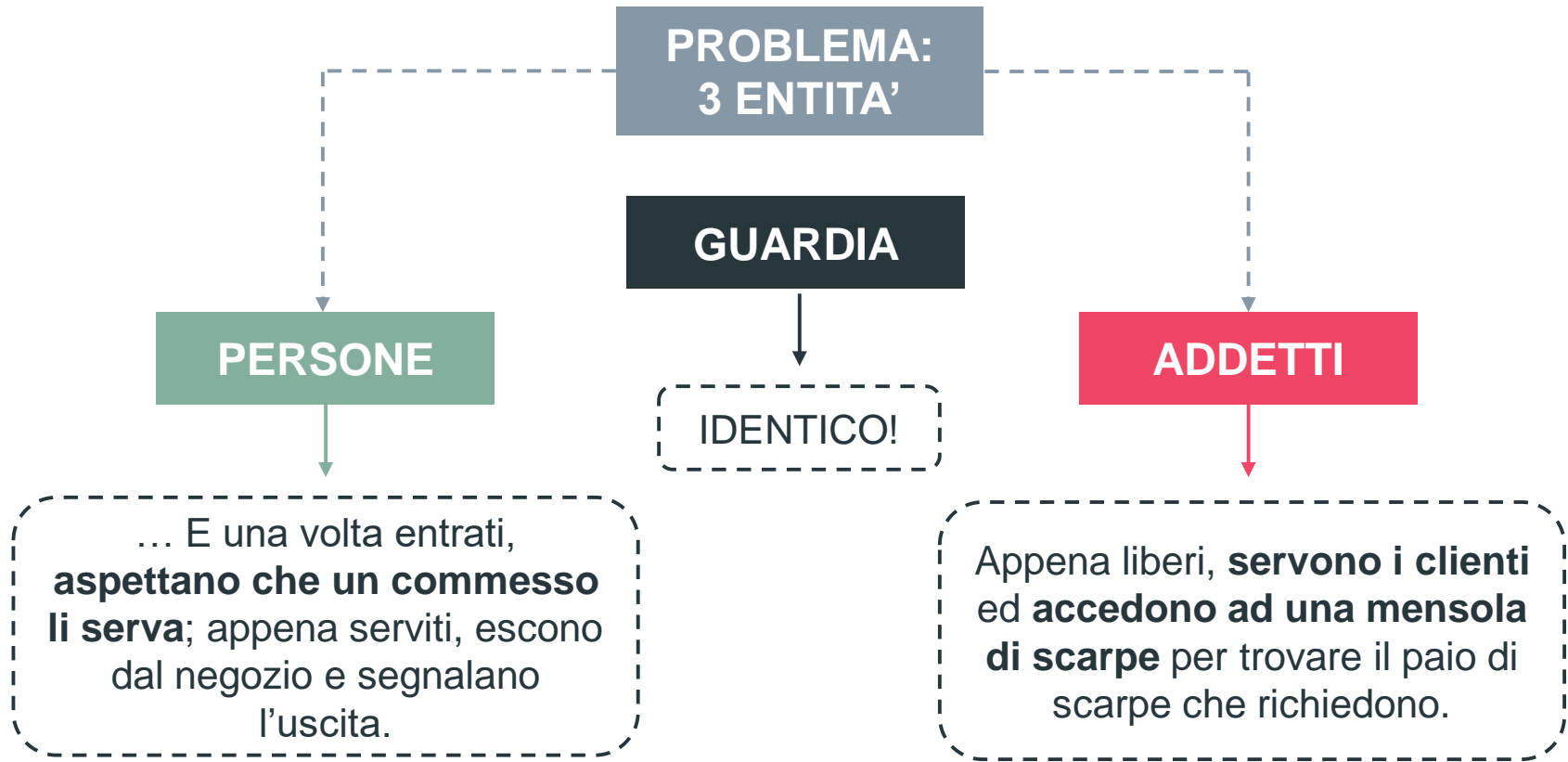
```
  // Aspetta il tuo turno
  wait(turno)
```

```
  // Fai compere nel negozio
  fai_compere();
```

```
  // Esci dal negozio
  signal(persone_ammesse)
```

```
}
```

# Fila ad un negozio: compere di scarpe





# Possibile Soluzione

```
guardia() {
    .....
}
```

```
persona() {
```

```
// Aggiungiti alla coda delle persone
signal(persone_in_fila)
```

```
// Aspetta il tuo turno
wait(turno)
```

```
// Entra nel negozio e segnala la presenza
signal(cliente)
```

```
// Aspetta l'aiuto dell'addetto
wait(addetto)
```

```
// Paga le scarpe prese
paga_scarpe();
```

```
// Esci dal negozio
signal(persone_ammesse)
```

```
}
```

```
semaforo contatore: persone_ammesse=N
semaforo contatore: turno=0
semaforo contatore: persone_in_fila=0
```

```
// Due nuovi semafori contatore
semaforo contatore: cliente=0
semaforo contatore: addetto=J
```

```
// Mutex per mutua esclusione
mutex: accesso_scaffale
```

```
// Risorsa: array di P elementi
// con valore 100 ognuno
array: scaffale[P] = 100
```

```
processo_main() {
```

```
-- crea_processo(guardia)
```

```
for i=0 to J:
```

```
    crea_processo(addetto)
```

```
for i=0 to K:
```

```
    crea_processo(persona)
```

```
// aspettare che i processi finiscano
```

```
}
```

```
addetto() {
```

```
repeat
```

```
// Aspetta che un cliente arrivi in negozio
wait(cliente)
```

```
// Genera posto casuale dello scaffale
posto_scarpe = rand(1,P)
```

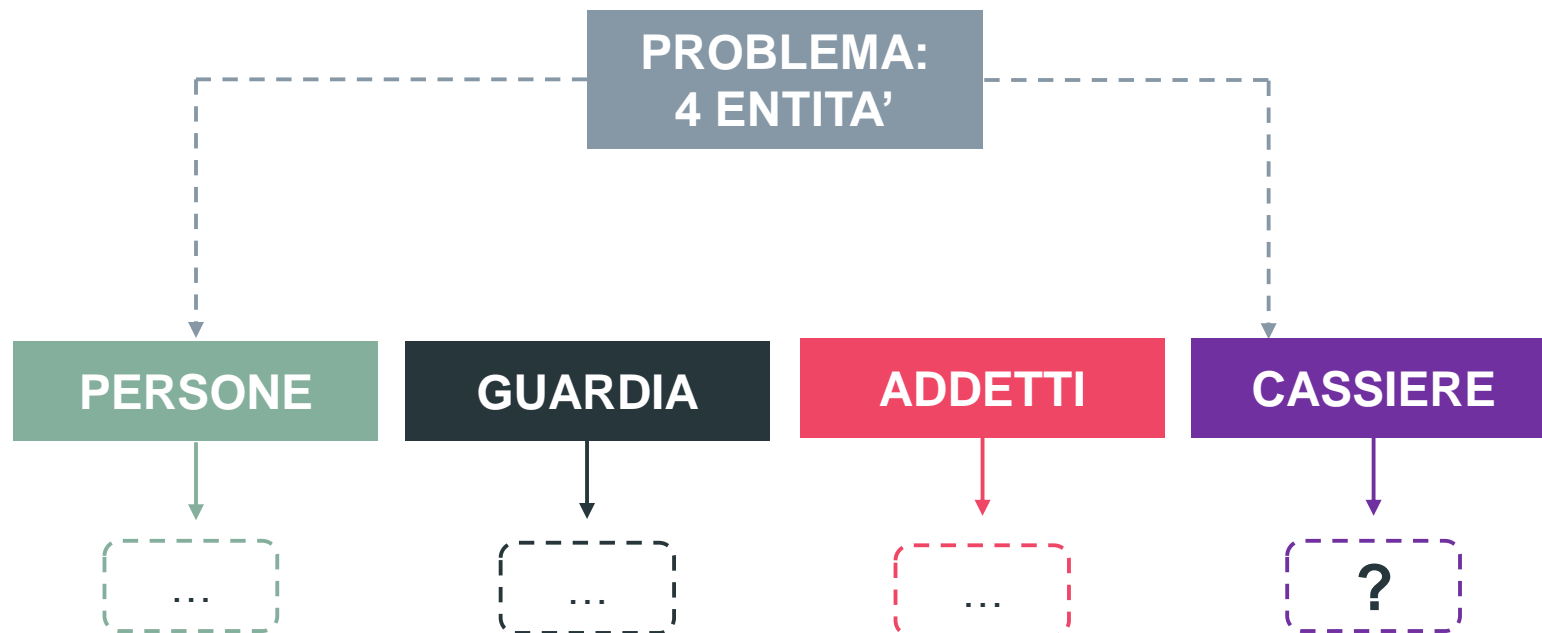
```
// Accedi allo scaffale in mutua esclusione:
// Decrementa di 1 la disponibilità dello scaffale
// nel posto P, quindi rilascia la mutua esclusione
wait(Mutex)
scaffale[P] -= 1
signal(Mutex)
```

```
// Segnala al cliente che può andare
signal(addetto)
```

```
forever
```

```
}
```

# Fila ad un negozio: cassiere



# L Esercizio

- In una fabbrica,  $N$  operai preparano piastrelle da far cuocere in un forno, capace di cuocerne  $M$  contemporaneamente.
- All'uscita dal forno  $K$  operai visionano le piastrelle per decorarle secondo tale sequenza di passi:
  - se trova una piastrella difettata inizia a prenderne dal forno 2 alla volta
  - altrimenti ne prende 1 alla volta

# L Soluzione: Main

```
Semaforo contatore: posti_informatore=10
Semaforo contatore: posti_decoratore=10
Semaforo contatore: forno_pieno=0
Semaforo contatore: forno_vuoto=10
Mutex: accesso_risorse=1
int ind_piastrelle = -1
int qualita_mattonella
int arr_piastrelle[1000]
int flagPiastrellaDifettata = false
```

Nota: la qualità della mattonella può essere  
 1 - non difettata  
 0 - difettata!

Ipotizziamo che su N/K operati totali, solo  
 10 operai alla volta (decoratori e  
 informatori) possono lavorare!

```
Main {

    // Creo 20 processi che decorano
    for i = 0 to 20:
        crea_OperarioDecoratore()

    // Creo 20 processi che infornano
    for i = 0 to 20:
        crea_OperarioInformatore()

    // Aspetto che finiscano, se finiscono

}
```

# L Soluzione: Produttori

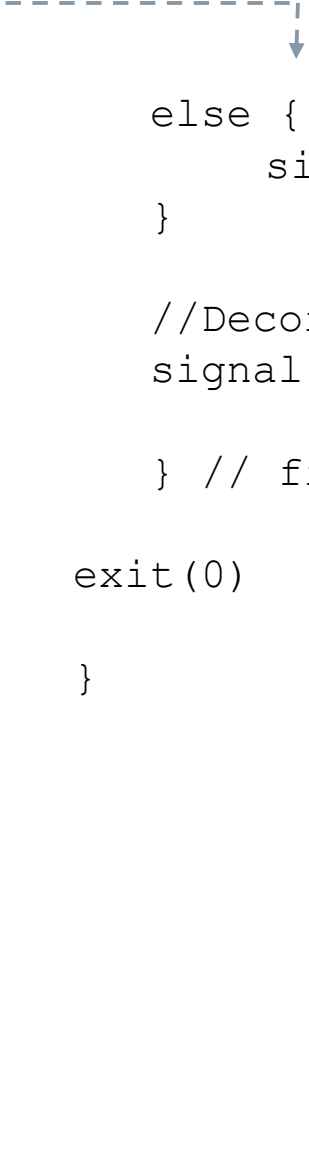
```
operaioInformatore() {
    while(1) {
        wait(posto_informatore)
        wait(forno_vuoto)
        wait(accesso_risorse)
        ind_piastrelle++
        mattonella = rand(0,1)
        arr_piastrelle[ind_piastrelle] = mattonella;
        signal(accesso_risorse)
        signal(forno_pieno)
        //CuociPiastrella();
        signal(posti_informatore)
    }
    exit(0)
}
```

# L Soluzione: Consumatori

```

operaioDecoratore() {
    while(1) {
        wait(posti_decoratore)
        wait(forno_pieno)
        wait(accesso_risorse)
        if (arr_piastrelle[ind_piastrelle]==0) {
            signal(accesso_risorse)
            wait(forno_pieno)
            wait(accesso_risorse)
            ind_piastrelle -=2;
            signal(accesso_risorse)
            signal(forno_vuoto)
            signal(forno_vuoto)
        }
        else if(arr_piastrelle[ind_piastrelle]==1)
        {
            ind_piastrelle--;
            signal(accesso_risorse)
            signal(forno_vuoto)
        }
    }
}

```



```

    else {
        signal(accesso_risorse)
    }

    //DecoraPiastrelle();
    signal(posti_decoratore)

    } // fine while

exit(0)

}

```