

# Esame 2021/07/05

Blascovich Alessio

## Domanda 1

MAX PUNTI: 6

**Domanda:**

Spiegare nel dettaglio cos'è e come viene usata la memoria TLB.

**Soluzione:**

TLB (Translation look-aside buffer) è una specie di cache per tradurre gli indirizzi di memoria in modo rapido, senza dover scorrere la tabella delle pagine.

Serve per evitare il problema del doppio accesso in memoria dato dal salvataggio in memoria della tabella delle pagine.

La/il TLB confronta contemporaneamente i numeri di tutte le pagine con quello fornito nell'indirizzo logico, se lo trova (TLB hit) lo concatena con l'offset, mentre se non lo trova (TLB miss) deve scannerizzare la tabella delle pagine.

La/il TLB essendo una forma di cache ha un rapporto dimensioni/costo molto elevato quindi al suo interno vengono memorizzate solo poche entry che verranno ripulite ad ogni context switch per evitare di mappare indirizzi errati.

## Domanda 2

MAX PUNTI: 6

**Domanda:**

Si consideri di avere i processi  $P_A$ ,  $P_B$ ,  $P_C$  e  $P_D$  con le risorse  $R_1$ ,  $R_2$ ,  $R_3$ ,  $R_4$  e  $R_5$ .

Sono illustrate nelle matrici sottostanti le risorse allocate e quelle massime per ogni processo:

*ALLOC*

*MAX*

	$R_1$	$R_2$	$R_3$	$R_4$	$R_5$
$P_A$	1	0	2	1	1
$P_B$	2	0	1	1	1
$P_C$	1	1	0	1	0
$P_D$	1	1	1	1	0

	$R_1$	$R_2$	$R_3$	$R_4$	$R_5$
$P_A$	1	1	2	1	3
$P_B$	2	2	2	1	1
$P_C$	2	1	3	1	0
$P_D$	1	1	2	2	1

Se il vettore delle risorse è  $A=(0,0,x,1,1)$ , quale valore deve avere  $x$  per rendere lo stato sicuro? (mostrare anche il procedimento che ha portato alla scelta di  $x$ )

**Soluzione:**

Provo ad usare l'algoritmo del banchiere spiegazione, quindi devo creare la matrice *Need*, definita come  $Need[i]=MAX[i]-ALLOC[i]$ .

	<i>Need</i>				
	$R_1$	$R_2$	$R_3$	$R_4$	$R_5$
$P_A$	0	1	0	0	2
$P_B$	0	2	1	0	0
$P_C$	1	0	3	0	0
$P_D$	0	0	1	1	1

Ora devo trovare una  $x$  per cui vale  $\forall i \in [A, \dots, D] : Need[i] \leq A$ , pongo  $work=A$ .

- $x=0 \Rightarrow work=(0,0,0,1,1)$ 
  - $(0,1,0,2) \leq (0,0,0,1,1) \Rightarrow \text{false}$ .
  - $(0,2,1,0,0) \leq (0,0,0,1,1) \Rightarrow \text{false}$ .
  - $(1,0,3,0,0) \leq (0,0,0,1,1) \Rightarrow \text{false}$ .
  - $(0,0,1,1,1) \leq (0,0,0,1,1) \Rightarrow \text{false}$ .

Se da dal primo ciclo non incrementato nulla posso passare alla prossima  $x$ .

- $x=1 \Rightarrow \text{work}=(0,0,1,1,1)$ 
  - $(0,1,0,0,2) \leq (0,0,1,1,1) \Rightarrow \text{false}$ .
  - $(0,2,1,0,0) \leq (0,0,1,1,1) \Rightarrow \text{false}$ .
  - $(1,0,3,0,0) \leq (0,0,1,1,1) \Rightarrow \text{false}$ .
  - $(0,0,1,1,1) \leq (0,0,1,1,1) \checkmark \Rightarrow \text{work}[] = \text{work}[] + \text{ALLOC}[P_D] = (1,1,2,2,1)$ .

Ma anche ripetendo non sono in uno stato safe quindi passo alla prossima  $x$ .

- $x=2 \Rightarrow \text{work}=(0,0,2,1,1)$ 
  - $(0,1,0,0,2) \leq (0,0,2,1,1) \Rightarrow \text{false}$ .
  - $(0,2,1,0,0) \leq (0,0,2,1,1) \Rightarrow \text{false}$ .
  - $(1,0,3,0,0) \leq (0,0,2,1,1) \Rightarrow \text{false}$ .
  - $(0,0,1,1,1) \leq (0,0,2,1,1) \checkmark \Rightarrow \text{work}[] = \text{work}[] + \text{ALLOC}[P_D] = (1,1,3,2,1)$ .

Adesso ripeto il ciclo escludendo  $P_D$  con  $\text{work}[]=(1,1,3,2,1)$ .

- $(0,1,0,0,2) \leq (1,1,3,2,1) \Rightarrow \text{false}$ .
- $(0,2,1,0,0) \leq (1,1,3,2,1) \Rightarrow \text{false}$ .
- $(1,0,3,0,0) \leq (1,1,3,2,1) \checkmark \Rightarrow \text{work}[] = \text{work}[] + \text{ALLOC}[P_C] = (2,2,3,3,1)$ .

Adesso ripeto il ciclo escludendo  $P_D$  e  $P_C$  con  $\text{work}[]=(2,2,3,3,1)$ .

- $(0,1,0,0,2) \leq (2,2,3,3,1) \Rightarrow \text{false}$ .
- $(0,2,1,0,0) \leq (2,2,3,3,1) \checkmark \Rightarrow \text{work}[] = \text{work}[] + \text{ALLOC}[P_B] = (4,2,4,4,2)$ .

Adesso ripeto il ciclo escludendo  $P_D$ ,  $P_C$  e  $P_B$  con  $\text{work}[]=(4,2,4,4,2)$ .

- $(0,1,0,0,2) \leq (4,2,4,4,2) \checkmark \Rightarrow \text{work}[] = \text{work}[] + \text{ALLOC}[P_A] = (5,2,6,5,3)$

Iterando l'algoritmo ho scoperto che il più piccolo vettore A è  $(0,0,2,1,1)$  quindi con  $x=2$ .

## Domanda 3

**MAX PUNTI:** 6

**Domanda:**

Spiegare nel dettaglio come funzionano gli i-node e come contengono informazioni.

**Soluzione:**

Gli i-node sono usati nel mondo UNIX per rappresentare i file, in un i-node sono memorizzati i metadati di un file e fa anche da indice per il file.

Gli i-node sono gestiti dal O.S. quindi sono permanentemente memorizzati in una porzione riservata del disco (solitamente all'inizio).

Dopo i metadati nell'i-node sono memorizzati anche:

- 10 puntatori per l'accesso diretto al file.
- 1 puntatore ad una tabella di indici di 1° livello.
- 1 puntatore ad una tabella di indici di 2° livello.
- 1 puntatore ad una tabella di indici di 3° livello.

## Domanda 4

MAX PUNTI: 6

**Domanda:**

Data la stringa di frame di memoria: 1,3,2,4,1,4,3,5,1,3,2,4,5,6,1,3,4,6,4,1,2,6,1,3,4,2.

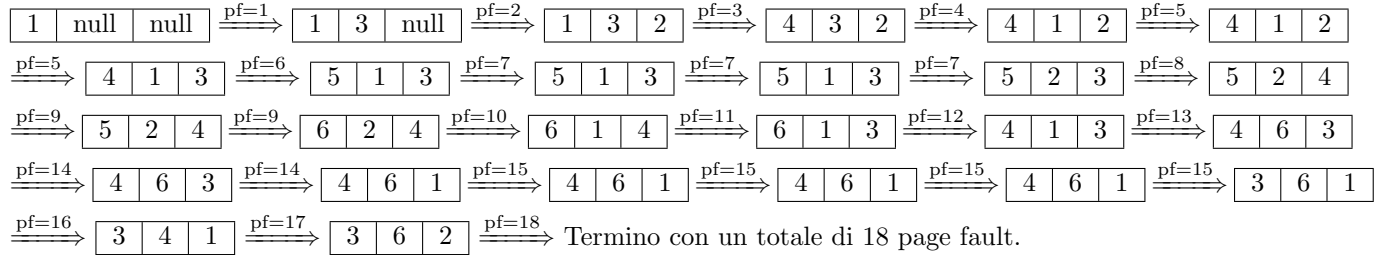
Determinare il numero di page fault dato dagli algoritmi *FIFO*, *LRU* e *Ideale* ipotizzando si avere 3 frame a disposizione inizialmente vuoti.

**Soluzione:**

Si consideri la sigla pf come l'acronimo di page-fault.

• **FIFO:**

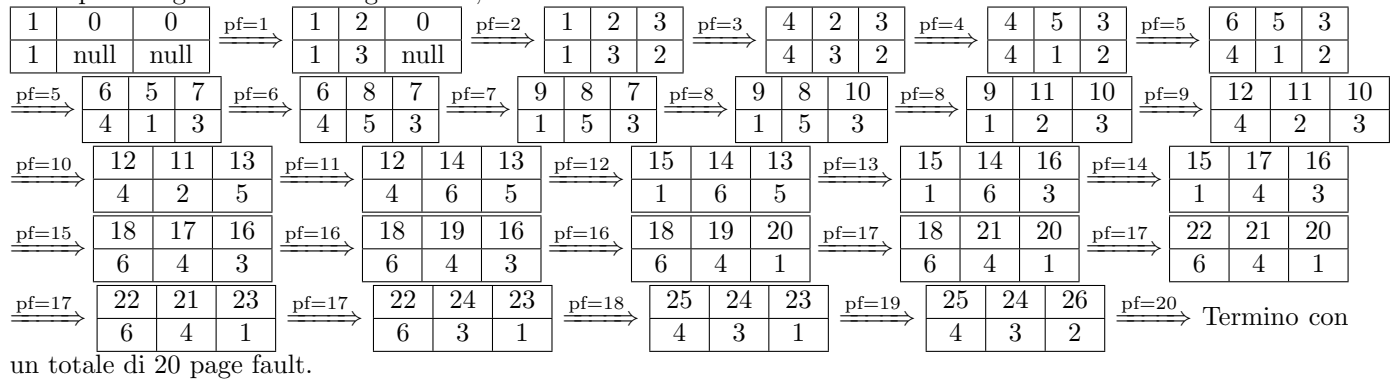
Il metodo *FIFO* gestisce la memoria come una semplice coda, il primo segmento ad essere entrato è il primo ad essere fatto uscire.



• **LRU:**

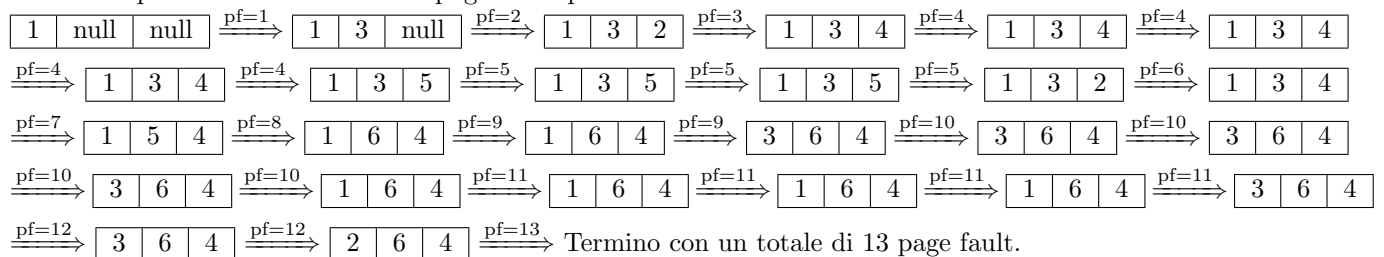
L'algoritmo *LRU* tiene associato ad ogni frame il clock della CPU nell'istante in cui quel frame è stato usato per l'ultima volta, verrà tolto il frame che non viene usato da più tempo.

Nella prima riga sono indicati gli istanti, mentre nella seconda il numero del frame.



• **Ideale:**

Come dice il nome questo è un algoritmo perfetto, perchè dovrebbe prevedere il futuro, infatti dovrebbe sapere quali saranno le prossime richieste che un programma potrebbe fare.



## Domanda 5

MAX PUNTI: 8

**Domanda:**

Si ipotizzi di avere una toilette pubblica unisex che viene gestita secondo i seguenti punti:

1. La toilette può essere libera, occupata da donnae, occupata da uomini.
2. Se è libera la persona che arriva la occupa.

3. Se la toilette è occupata da una o più donne, possono entrare solo donne.
4. Se la toilette è occupata da uno o più uomini, possono entrare solo uomini.

Progettare un sistema per la sincronizzazione con il metodo che si vuole.  
Non fare alcuna assunzione sul numero di uomini/donne.

**Soluzione:**

*Variabili in comune*

```
Toilette wc; //Risorsa da usare
List<Uomo> codaMaschile; //Coda attesa maschile
List<Donna> codaFemminile; //Coda attesa femminile
Monitor monitor=free; //Inizialmente il monitor segna la toilette come libera

Uomo(){
    while(true){
        codaMaschile.append(new Uomo());
        if(monitor.isFree()){
            monitor.lock();
            while(codaMaschile.isEmpty()){
                Uomo tmp=codaMaschile.remove();
                tmp.use(wc);
            }
            monitor.free();
        }
    }
}

Donna(){
    while(true){
        codaFemminile.append(new Donna());
        if(monitor.isFree()){
            monitor.lock();
            while(codaFemminile.isEmpty()){
                Uomo tmp=codaFemminile.remove();
                tmp.use(wc);
            }
            monitor.free();
        }
    }
}
```