

## ARGOMENTO: Paginazione Memoria Centrale

### TRACCIA:

Sia data la seguente sequenza di riferimenti a pagine di uno spazio d'indirizzamento logico.

1 2 3 1 4 3 6 1 4 6 1 6 1 7

Supposto di disporre di una memoria fisica costituita da 3 blocchi, indicare il **contenuto dei blocchi indicati nel caso di algoritmo di rimozione FIFO**. Indicare se si è verificato un page fault.

Most Recently Used page														
Least Recently Used page														
page fault (Y/N)														

### SVOLGIMENTO:

	1	2	3	1	4	3	6	1	4	6	1	6	1	7
MRU	1	2	3	1	4	3	6	1	4	6	1	6	1	7
LRU	1	1	1	2	2	2	4	4	6	1	4	4	4	6
PF (Y/N)	Y	Y	Y	N	Y	N	Y	Y	N	N	N	N	N	Y

### COMMENTI E INFO AGGIUNTIVE:

Con 3 blocchi di memoria, vedo se nelle tre pagine MEMORIZZATE in Mem.Centrale ho la mia attuale.

*Least Recently Used* : pagina usata meno di recente fra quelle in memoria

*Most Recently Used*: pagina usata più di recente fra quelle in memoria

*FIFO*: First in First out (struttura di una coda)

*PF*: Page Fault – si verifica se non trovo la pagina che mi interessa tra quelle memorizzate in Mem. Centrale e sono costretto, quindi, a recuperarla dalla Mem. Di massa.

E' come se avessimo un vettore di tre elementi facente la nostra memoria [ X, Y, Z] in cui andiamo a eliminare tramite First In First Out le pagine per far spazio a quelle nuove. Sostanzialmente eliminiamo sempre la pagina più a sinistra, ed inseriamo le nuove a destra (come in una coda!)

## ARGOMENTO: Indirizzo logico in fisico

### TRACCIA:

Si voglia *tradurre l'indirizzo logico* (5, 1100) di un processo in corrispondente indirizzo fisico. Consultando la Page Table seguente, dove si troverà la pagina richiesta e, se in memoria, quale sarà il risultato della traduzione assumendo che la dimensione di una pagina sia di 4 Kb?

Page #	Invalid Bit	Page Frame#	EPFT reference
....	....	....	....
1	1	30	40
2	0	45	42
3	1	22	11
4	1	38	8
5	1	33	25
6	0	56	33
7	0	21	50
8	1	50	34
9	1	23	21

### SVOLGIMENTO:

La mia pagina è 5. Controllo nella colonna “Page” in corrispondenza della riga con valore 5 lo stato del invalid bit. Poiché  $ib=1$  la pagina E' PRESENTE in memoria. ( D: dove si troverà la pagina richiesta? R: In memoria centrale).

Considero il valore sotto la colonna “Page Frame” (se  $ib=0$  consideravamo EPFT) ed effettuo un semplice calcolo:

$$\text{Indirizzo fisico} = (33 * 4096) + 1100 = 136268$$

Ove 4096 sono 4Kb espressi in byte ( $4 * 1024$ ). E' importante conoscere tale valore perchè 1Byte = 1 cella di memoria.

### COMMENTI E INFO AGGIUNTIVE:

*EPFT (External Page Frame Table)*: contiene tanti record quante sono le pagine contenute nella memoria virtuale

## ARGOMENTO: Hard Disk

### TRACCIA:

Si supponga che le testine di un disco fisso siano posizionate sul cilindro 5. Scrivere l'espressione (e calcolarne quindi il valore) del **tempo medio (tempo massimo)** richiesto dalla lettura del blocco (CYL=10, TRK=10, SEC=10) se il seek time del disco è di 0,1 msec/cyl e la velocità di rotazione è pari a 5000 giri/minuto.

### SVOLGIMENTO:

L'espressione del tempo medio/massimo è la seguente:

$$\mathbf{t \text{ massimo} = (tseek + rotational \ delay)}$$

$$\mathbf{t \text{ medio} = (tseek + rotational \ delay)}$$

Calcoliamo prima il rotational delay medio e massimo, ovvero il tempo richiesto per una rotazione:

$$\mathbf{5000giri : 60 \ secondi = 1 \ giro : X_{max}}$$

$$\mathbf{X_{max} = 60/5000 = 12ms}$$

$$\mathbf{5000giri : 60 \ secondi = 0,5 \ giro : X_{med}}$$

$$\mathbf{X_{med} = (0,5*60)/5000 = 6ms}$$

Individuiamo ora il tempo di seek, cioè il tempo richiesto per posizionarsi sul cilindro 5 al cilindro 10:

$$\mathbf{tseek = (10-5)cyl*0,1ms/cyl = 0,5ms}$$

Il valore finale sarà:

$$\mathbf{t_{medio} = (0,5ms + 6ms) = 6,5ms}$$

$$\mathbf{t_{massimo} = (0,5ms + 12ms) = 12,5ms}$$

### COMMENTI E INFO AGGIUNTIVE:

Gli altri due valori ovvero TRK (traccia) e settore (SEC) non hanno importanza ai fini del calcolo in quanto è il posizionamento del cilindro che conta.

Non sapendo se servirà una rotazione completa o meno per posizionarsi correttamente, possiamo calcolare tmedio assumendo mezza rotazione e tmassimo assumendo 1 rotazione completa.

## ARGOMENTO: Algoritmo del Banchiere

### TRACCIA:

Si consideri un sistema che si trovi nello stato descritto di seguito:

	<u>Allocation</u>				<u>Max</u>				<u>Available</u>			
	A	B	C	D	A	B	C	D	A	B	C	D
P1	0	0	1	1	0	0	1	2	1	5	1	1
P2	0	0	3	4	2	3	4	6				
P3	1	0	0	0	2	7	5	1				
P4	1	3	3	4	2	6	5	6				
P5	0	6	3	2	0	6	5	2				

Il sistema è in uno **stato ammissibile**? Perché?

Il sistema è in uno **stato sicuro**? Perché?

### SVOLGIMENTO:

L'Algoritmo del banchiere è utilizzato per prevenire i Deadlock nell'allocazione delle risorse. In particolare questo algoritmo può indicare se un sistema si venga a trovare in uno stato sicuro o meno nel caso assegnasse una risorsa ad uno dei processi richiedenti. Lo stato sicuro quindi è uno stato in cui è possibile allocare tutte le risorse richieste da un processo senza che quest'ultimo comporti un deadlock con un altro processo. Il fatto che il sistema si trovi in uno stato sicuro non implica che tutte le allocazioni di risorse avverranno con successo, ma solo che esiste almeno un modo sicuro per allocare tutte le risorse.

1) Per verificare che il sistema sia ammissibile, il numero massimo di risorse richieste da ciascun processo (la matrice max), sia minore del numero massimo degli esemplari totali delle risorse disponibili. **Nell'esempio sopra avremo che ogni riga di  $MAX[i] < A(3), B(14), C(11), D(12)$**  (somma di tutte le risorse compreso available). Inoltre è necessario verificare che le risorse specificate nella matrice allocation, siano minori di quelle in max, per ogni processo.

Poiché entrambe le condizioni sono soddisfatte, il sistema è ammissibile ed ha quindi senso applicare l'algoritmo del banchiere.

2) Prima di tutto è necessario calcolare la tabella "NEED" ottenibile da una semplice differenza:

$$NEED[i] = MAX[i] - ALLOCATION[i]$$

	<u>Need</u>			
	A	B	C	D
P1	0	0	0	1
P2	2	3	1	2
P3	1	7	5	1
P4	1	3	2	2
P5	0	0	2	0

Verifichiamo ora se esiste una sequenza sicura di allocazione delle risorse:

- Le risorse disponibili (available) ci permettono di servire P1 infatti  
 **$need[P1](0,0,0,1) < Available(1,5,1,1)$**   
Il nostro available diventa  **$allocation[P1] + Available = (1,5,2,2)$**
- Le risorse disponibili ci permettono di servire P4 infatti  
 **$need[P4](1,3,2,2) < Available(1,5,2,2)$**

Il nostro available diventa **allocation[P4] + Available** = (1,3,3,4) + (1,5,2,2) = (2,8,5,6)

3. Le risorse disponibili ci permettono di servire **P2, P3 e P5**. Possiamo quindi accettare come sequenza sicura:

**<P1, P4, P2, P3, P5>**

**Poiché esiste almeno una sequenza sicura, il sistema è in uno stato sicuro.**

### TRACCIA:

Si consideri un sistema che si trovi nello stato sicuro descritto nel seguito:

Available											
R1	R2	R3	R4								
2	1	0	2								
				Allocation				Need			
Process				R1	R2	R3	R4	R1	R2	R3	R4
P1				0	8	0	5	0	0	5	3
P2				1	1	0	1	1	0	0	1
P3				2	0	2	0	1	3	2	0
P4				4	2	2	1	3	3	4	2
P5				0	2	2	1	1	1	0	1

È ammissibile che il processo P3 richieda (1, 1,1,0) risorse? Perché?  
E rimarrà il sistema in uno stato sicuro? Perché?

### SVOLGIMENTO:

1)Prima di tutto è necessario calcolare la tabella “Max”:

$$\mathbf{Max[i] = Need[i]+Allocation[i]}$$

	Max			
Process	R1	R2	R3	R4
P1	0	8	5	8
P2	2	1	0	2
P3	3	3	4	0
P4	7	5	6	3
P5	1	3	2	2

Poiché il P3 sta effettuando una nuova **request** di risorse, è necessario riscrivere la tabella allocation sostituendo la riga P3 con **Allocation[P3] + P3request**:

	Allocation			
Process	R1	R2	R3	R4
P1	0	8	0	5
P2	1	1	0	1
P3	3	1	3	0
P4	4	2	2	1
P5	0	2	2	1

Poiché **Allocation[P3] < Max[P3]** e **MAX[i] < Somma risorse** il sistema è ancora ammissibile ed è quindi ammissibile tale richiesta.

2) Per verificare se il sistema è ancora sicuro, è necessario applicare l'algoritmo del banchiere.