

Formule contigua, linkata e indicizzata

- 1) n. blocchi = $\text{size disco} / \text{size blocco}$;
- 2) $\text{size puntatore} = \log_2 \text{n. blocchi}$;
- 3) $\text{blocco del byte} = \text{byte da cercare} / \text{size blocco}$;
- 4) nella linkata fare byte da cercare / ($\text{size blocco} - \text{size puntatore}$);

SOLO INDICIZZATA

- 1) n. puntatori in un blocco = $\text{size blocco} / \text{size puntatore}$ (per difetto);
per l'indicizzata concatenata togliere il puntatore finale usato per puntare al prossimo blocco indice
- 2) n. blocchi indice $\leq \text{numero blocchi} / \text{n. puntatori in un blocco}$ (per eccesso);

Lista Linkata

- 1) $\text{blocchi occupati} = \text{n. byte file} / \text{size blocco senza il puntatore}$;
- 2) $\text{blocchi liberi} = \text{n. blocchi} - \text{blocchi occupati}$;

BITMAP

- 1) n. blocchi = $\text{size disco} / \text{size blocco}$;
- 2) $\text{size bitmap} = \text{numero blocchi}$;

FILE e RECORD

- 1) $\text{dim. File blocco indice} = \text{n. puntatore in un blocco} * \text{size blocco}$;
- 2) $\text{n. record in un blocco} = \text{size blocco} / \text{size record}$
- 3) $\text{n. blocchi file con record} = \text{size file} / \text{n. record in un blocco}$

FAT

- 1) $\text{size FAT} = \text{n. blocchi} * \text{size puntatore}$;
- 2) $\text{n. blocchi FAT} = (\text{size puntatore} * \text{n. blocchi}) / \text{size blocchi}$;
- 3) cercare Byte in una FAT = byte da cercare / size blocco (iniziare a contare dal blocco data nella traccia);
es. se il byte è nel blocco 3 e la traccia ci dice di partire da 6 seguiremo 3 blocchi partendo da 6 compreso;

SECONDA PARTE DEL CORSO

Sinconizzazione

```
Wait 1(semaphore s)
{
    while(s <= 0) ;
    s--;
}
```

```
Signal1(semaphore s)
{
    s++;
}
```

```
Wait 2(semaphore s)
{
    s-value--;
    if(s → value = 0)
    {
        block(p);
    }
}
```

```
Signal1(semaphore s)
{
    s → value ++;
    if(s → value <= 0)
        wakeup(p);
}
```

Pagine e frame

First Fit= il primo più grande che lo può contenere;

Best Fit= il blocco che lo può contenere consumando meno spazio;

Worst Fit= prende quello più grande del sistema che può contenerlo

size pagina= size frame

Dimensione entry table= numero di pagine;

Dimensione RAM= numero di frame;

Offset= logaritmo in base 2 della size pagine o della size frame;

Numeri bit pagina= logaritmo in base 2 del numero di pagine;

Numeri bit frame= logaritmo in base 2 del numero di frame;

Indirizzo logico=|bit pagine | bit offset|; // la dim= bit pagine+ bit offset

Indirizzo fisico=|bit frame | bit offset|; // la dim= bit frame+ bit offset

Memoria Virtuale= Si ha quando l'indirizzo logico ha più bit dell'indirizzo fisico ed ha obbligatoriamente il bit di validità;

EAT=%hit ratio (accesso TLB + accesso RAM)

+ % miss ratio(accesso TLB + 2*accesso RAM)

accedere alla page table è uguale a fare due accessi in memoria;

Page table=#frame +(bit validità,bit modifica,bit di riferimento)*

dim page table= num bit frame+(bit validità,bit modifica,bit di riferimento)*

convertito in byte moltiplicato per il #pagine;

Inverted page table= pid processo,#pag,
(bit validità,bit modifica,bit di riferimento)*

Dim Inverted page table= (pid processo + #pag,+(
bit validità +bit modifica +bit di riferimento))fratto 8
moltiplicato per il #frame pid processo di solito=X;

Algoritmi scheduling delle Pagine

FIFO= il primo che entra è il primo ad uscire;

Ottimale= si vede al futuro e viene cacciato quello
che sarà usato fra più tempo;

LRU=viene cacciato quello usato meno recentemente

Seconda Chance v.1= usa il bit di riferimento se è 1
viene messo a 0 e il primo che ha 0 viene cacciato;

Seconda Chance v.2= usa la coppia bit di riferimento e
bit di modifica e sceglie la prima coppia con il valore minore
(0-0, 0-1,1-0,1-1);

<p>Pagine e frame First Fit= il primo più grande che lo può contenere; Best Fit= il blocco che lo può contenere consumando meno spazio; Worst Fit= prende quello più grande del sistema che può contenerlo size pagina= size frame Dimensione RAM= numero di frame; Dimensione entry table= numero di pagine; Offset= logaritmo in base 2 della size pagine o della size frame; Numeri bit pagina= logaritmo in base 2 del numero di pagine; Numeri bit frame= logaritmo in base 2 del numero di frame; Indirizzo logico= bit pagine bit offset ; // la dim= bit pagine+ bit offset Indirizzo fisico= bit frame bit offset ; // la dim= bit frame+ bit offset Memoria Virtuale=Si ha quando l'indirizzo logico ha più bit dell'indirizzo fisico ed ha obbligatoriamente il bit di validità;</p>	<p>EAT=%hit ratio (accesso TLB + accesso RAM) + % miss ratio(accesso TLB + 2*accesso RAM) accedere alla page table è uguale a fare due accessi in memoria;</p> <p>Page table=#frame +(bit validità,bit modifica,bit di riferimento)* dim page table= num bit frame+(bit validità,bit modifica,bit di riferimento)* convertito in byte moltiplicato per il #pagine; Inverted page table= pid processo,#pag, (bit validità,bit modifica,bit di riferimento)* Dim Inverted page table= (pid processo + #pag,+(bit validità +bit modifica +bit di riferimento))fratto 8 moltiplicato per il #frame pid processo di solito=X;</p>
<p>Algoritmi scheduling delle Pagine FIFO= il primo che entra è il primo ad uscire; Ottimale= si vede al futuro e viene cacciato quello che sarà usato fra più tempo; LRU=viene cacciato quello usato meno recentemente Seconda Chance v.1= usa il bit di riferimento se è 1 viene messo a 0 e il primo che ha 0 viene cacciato; Seconda Chance v.2= usa la coppia bit di riferimento e bit di modifica e sceglie la prima coppia con il valore minore (0-0, 0-1,1-0,1-1);</p>	
<p>Pagine e frame First Fit= il primo più grande che lo può contenere; Best Fit= il blocco che lo può contenere consumando meno spazio; Worst Fit= prende quello più grande del sistema che può contenerlo size pagina= size frame Dimensione RAM= numero di frame; Dimensione entry table= numero di pagine; Offset= logaritmo in base 2 della size pagine o della size frame; Numeri bit pagina= logaritmo in base 2 del numero di pagine; Numeri bit frame= logaritmo in base 2 del numero di frame; Indirizzo logico= bit pagine bit offset ; // la dim= bit pagine+ bit offset Indirizzo fisico= bit frame bit offset ; // la dim= bit frame+ bit offset Memoria Virtuale=Si ha quando l'indirizzo logico ha più bit dell'indirizzo fisico ed ha obbligatoriamente il bit di validità;</p>	<p>EAT=%hit ratio (accesso TLB + accesso RAM) + % miss ratio(accesso TLB + 2*accesso RAM) accedere alla page table è uguale a fare due accessi in memoria;</p> <p>Page table=#frame +(bit validità,bit modifica,bit di riferimento)* dim page table= num bit frame+(bit validità,bit modifica,bit di riferimento)* convertito in byte moltiplicato per il #pagine; Inverted page table= pid processo,#pag, (bit validità,bit modifica,bit di riferimento)* Dim Inverted page table= (pid processo + #pag,+(bit validità +bit modifica +bit di riferimento))fratto 8 moltiplicato per il #frame pid processo di solito=X;</p>
<p>Algoritmi scheduling delle Pagine FIFO= il primo che entra è il primo ad uscire; Ottimale= si vede al futuro e viene cacciato quello che sarà usato fra più tempo; LRU=viene cacciato quello usato meno recentemente Seconda Chance v.1= usa il bit di riferimento se è 1 viene messo a 0 e il primo che ha 0 viene cacciato; Seconda Chance v.2= usa la coppia bit di riferimento e bit di modifica e sceglie la prima coppia con il valore minore (0-0, 0-1,1-0,1-1);</p>	