Formule contigua, linkata e indicizzata

1) n. blocchi= size disco/size blocco;
2) size puntatore= log2 n.blocchi;
3) blocco del byte= byte da cercare / size blocco;
4) nella linkata fare byte da cercare / (size blocco – size puntatore);

SOLO INDICIZZATA

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

Lista Linkata

1) blocchi occupati= n. byte file / size blocco senza il puntatore;
2) blocchi liberi= n. blocchi - blocchi occupati;

BITMAP

1) n. blocchi= size disco/size blocco;
2) size bitmap= numero blocchi;

FILE e RECORD

1) dim. File blocco indice = n. puntatore in un blocco * size blocco;
2) n. record in un blocco= size blocco / size record
3) n. blocchi file con record= size file / n. record in un blocco

FAT

1)size FAT= n. blocchi * size puntatore;
2) n. blocchi FAT= (size puntatore* n. blocchi) / 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 di memoria che lo può contenere consumando meno spazio;
Worst Fit= prende il blocco 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 (tempo di accesso alla TLB + tempo accesso alla RAM)
+
% miss ratio(tempo di accesso alla TLB + tempo accesso alla RAM+ tempo accesso alla RAM)

*esprimere la percentuale in valore decimale es. 90%=0.9

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

Page table=#frame +(bit validità,bit modifica,bit di riferimento)*
*opzionale
dim page table= num bit frame+(bit validità,bit modifica,bit di riferimento)*
convertito in byte moltiplicato per il #pagine; *opzionale

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

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