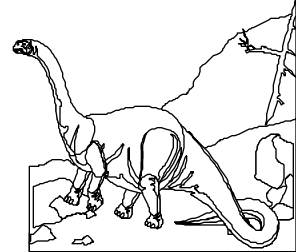
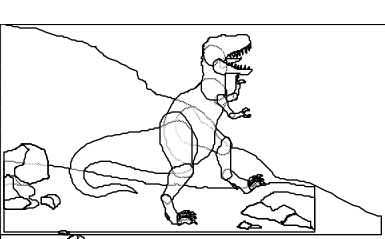


# Esercizio 1

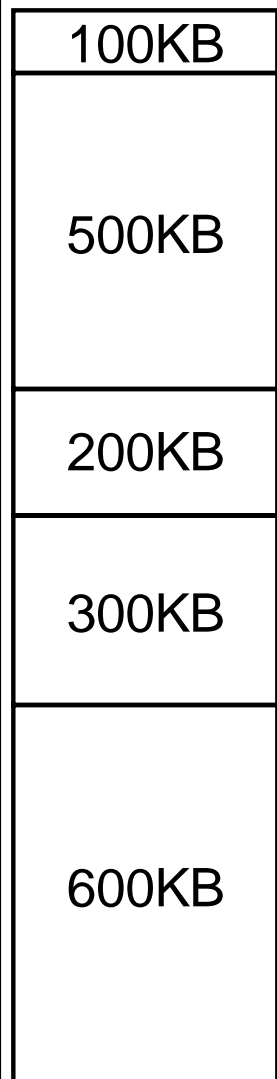
- Date le seguenti partizioni di memoria: 100KB, 500KB, 200KB, 300KB e 600KB, descrivere come sono disposti dagli algoritmi di *first-fit* e *best-fit* i processi di 212KB, 417KB, 112KB e 426KB (in ordine).



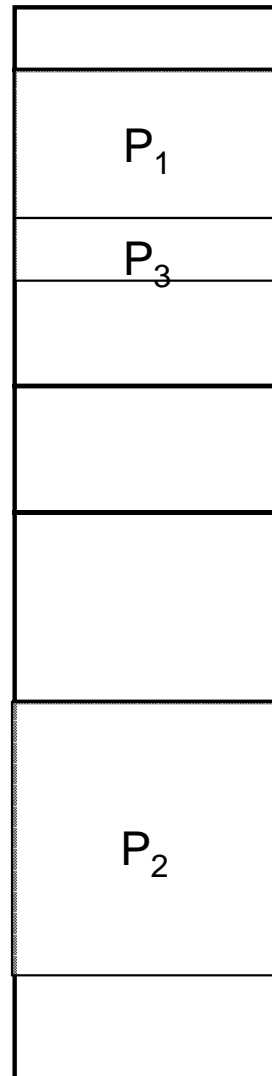


# Esercizio 1

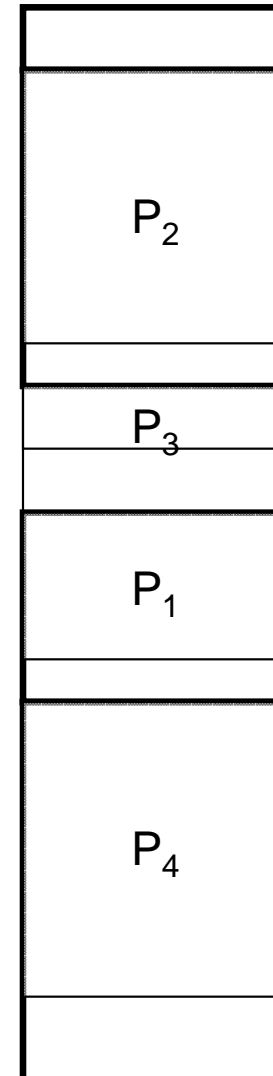
- Date le seguenti partizioni di memoria: 100KB, 500KB, 200KB, 300KB e 600KB, descrivere come sono disposti dagli algoritmi di *first-fit* e *best-fit* i processi di 212KB, 417KB, 112KB e 426KB (in ordine).



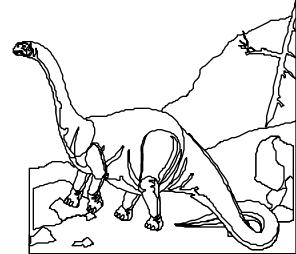
Situazione iniziale



Con first-fit  
P<sub>4</sub> non può essere allocato

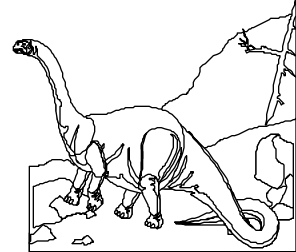


Con best-fit tutti possono essere allocati



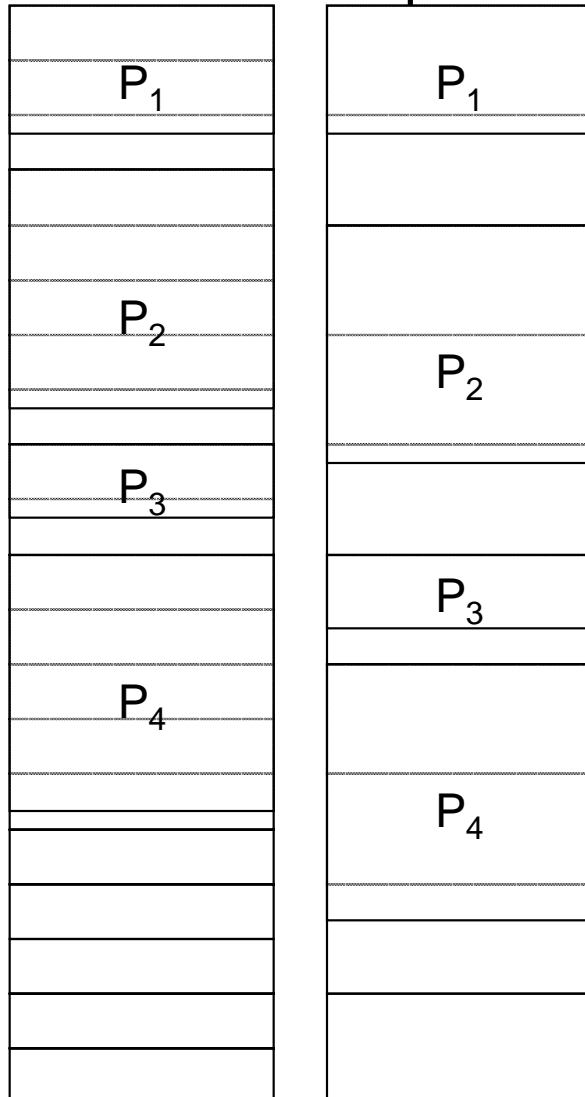
# Esercizio 2

- Se la memoria è 2000KB e viene gestita mediante la paginazione, con pagine da 100KB e 200KB, si descriva l'allocazione della memoria, relativamente a processi di dimensioni 212KB, 417KB, 112KB e 426KB.

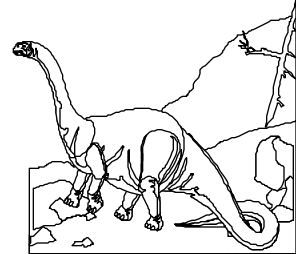


# Esercizio 2

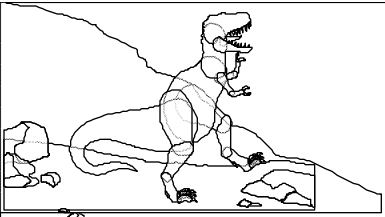
- Se la memoria è 2000KB e viene gestita mediante la paginazione, con pagine da 100KB e 200KB, si descriva l'allocazione della memoria, relativamente a processi di dimensioni 212KB, 417KB, 112KB e 426KB.



- La frammentazione interna è maggiore nel caso di pagine più grandi.
- La memoria libera è di 5 blocchi (500K) nel primo caso, uno solo (200K) nel secondo.



# Esercizio 3

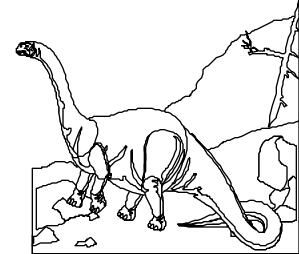


- Considerate la seguente tabella dei segmenti:

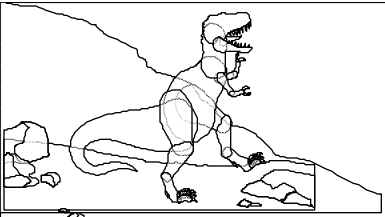
Segmento	Base	Lunghezza
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

- Calcolate gli indirizzi fisici corrispondenti ai seguenti indirizzi logici:

- a)  $\langle 0, 430 \rangle$
- b)  $\langle 1, 10 \rangle$
- c)  $\langle 2, 500 \rangle$
- d)  $\langle 3, 400 \rangle$
- e)  $\langle 4, 112 \rangle$



# Esercizio 3

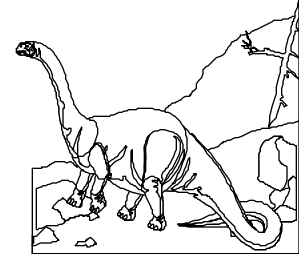


- Considerate la seguente tabella dei segmenti:

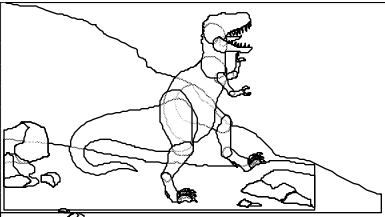
Segmento	Base	Lunghezza
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

- Calcolate gli indirizzi fisici corrispondenti ai seguenti indirizzi logici:

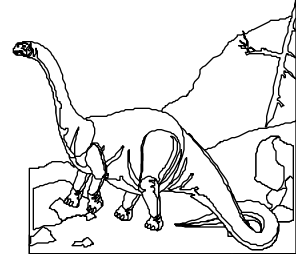
- a)  $\langle 0, 430 \rangle$        $\rightarrow$        $219 + 430 = 649$
- b)  $\langle 1, 10 \rangle$        $\rightarrow$        $2300 + 10 = 2310$
- c)  $\langle 2, 500 \rangle$        $\rightarrow$        **$500 > 100$  errore di segmentazione**
- d)  $\langle 3, 400 \rangle$        $\rightarrow$        $1327 + 400 = 1727$
- e)  $\langle 4, 112 \rangle$        $\rightarrow$        **$112 > 96$  errore di segmentazione**



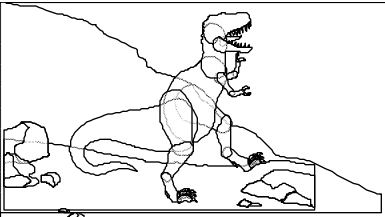
# Esercizio 4



- Un programma ha il segmento di testo di 200KB, quello dei dati di 100KB e uno stack di 50KB. Se la memoria è organizzata a pagine da 16KB:
  1. Quanto è lunga la tabella delle pagine del processo?
  2. Quanta memoria può al più essere gestita se si hanno a disposizione 32 bit per l'indirizzamento?



# Esercizio 4



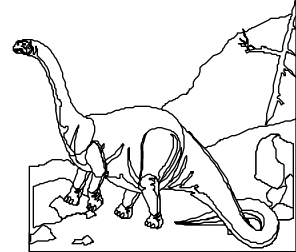
- Un programma ha il segmento di testo di 200KB, quello dei dati di 100KB e uno stack di 50KB. Se la memoria è organizzata a pagine da 16KB:
  1. Quanto è lunga la tabella delle pagine del processo?
  2. Quanta memoria può al più essere gestita se si hanno a disposizione 32 bit per l'indirizzamento?

## 1. Risulta:

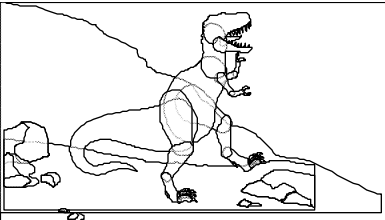
- Testo 200KB →  $200/16 = 13$  pagine
- Dati 100KB →  $100/16 = 7$  pagine
- Stack 50KB →  $50/16 = 4$  pagine

Per un totale di **24 pagine**

2. Per indirizzare pagine da 16KB, c'è bisogno ( $16K=2^{14}$ ) di 14 bit. Restano 18 bit per generare i numeri delle pagine → 256K pagine, pari a 4GB di memoria complessiva.



# Esercizio 5



- Si consideri un gestore della memoria basato su avvicendamento che utilizza una lista per rappresentare lo stato di allocazione della memoria.
- Supponiamo che la lista contenga le seguenti informazioni:

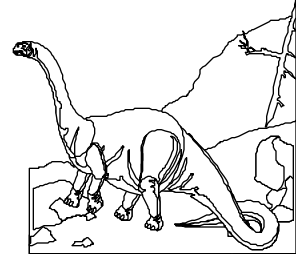
indirizzo \

$(P,0,12) \rightarrow (H,12,5) \rightarrow (P,17,6) \rightarrow (H,23,12) \rightarrow (P,35,5) \rightarrow (H,40,9) \rightarrow (P,49,10)$

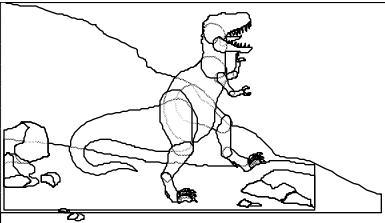
dimensione /

*(il simbolo "P" denota una partizione di memoria occupata da un processo, mentre il simbolo "H" denota una partizione di memoria libera)*

- Se al gestore della memoria arrivano in tempi successivi due richieste di allocazione di memoria per due processi, il primo di dimensione 7 e il secondo 10, dire come si comporta il gestore della memoria nel caso di politiche di allocazione best-fit e first-fit e mostrare in entrambi i casi la lista di allocazione della memoria risultante.



# Esercizio 5



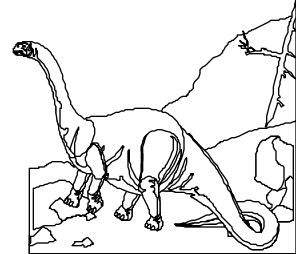
- Il best fit seleziona per il primo (H, 40, 9) e per il secondo (H, 23, 12), per cui la lista diventa:

(P,0,12)->(H,12,5)->(P,17,6)->(P,23,10)->(H,33,2)->(P,35,5)->(P,40,7)->(H,47,2)->(P,49,10)

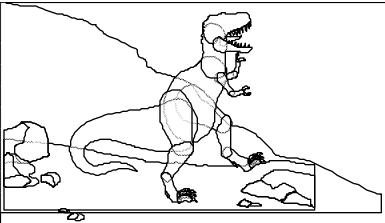
- Il first fit seleziona per il primo (H,23,12), per cui la lista diventa:

(P,0,12)->(H,12,5)->(P,17,6)->(P,23,7)->(H,30,5)->(P,35,5)->(H,40,9)->(P,49,10)

- A questo punto non ci sono più frammenti liberi di dimensioni sufficienti per contenere il secondo processo, per cui il gestore della memoria libera spazio di memoria scaricando un processo sul disco, scelto in base ad una politica di avvicendamento.



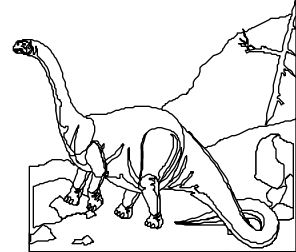
# Esercizio 6



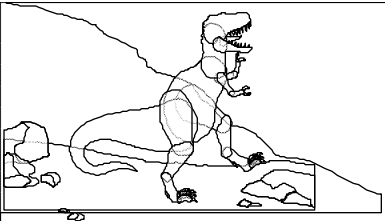
- In un sistema di gestione della memoria con avvicendamento dinamico, che utilizza un registro base  $B$  ed un registro limite  $L$  per assicurare la protezione degli spazi di indirizzamento, abbiamo 32M di RAM, di cui gli 8M di indirizzi bassi occupati dal Sistema Operativo. Consideriamo i seguenti programmi : A, di 4M, B di 16M, C di 12M e D di 6M.

Supponiamo inoltre che i programmi passino in esecuzione nell'ordine seguente: A, B, C, D, A.

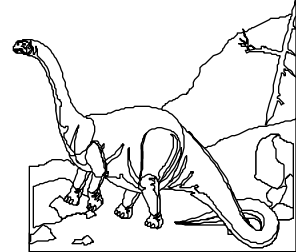
- Ogni volta che un nuovo processo va in esecuzione discutere:
  - a) la mappa della memoria centrale
  - b) che cosa contengono i registri *base* e *limite*.



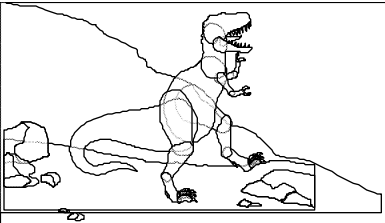
# Esercizio 6



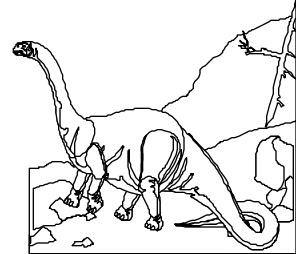
- 1) Va in esecuzione A, che viene caricato all'indirizzo 8M,  $L=4M$  e  $B=8M$ .  
La mappa di memoria è  $SO(\text{ind } 0, \text{lung } 8M)$ ,  $A(\text{ind } 8M, \text{lung } 4M)$
- 2) Va in esecuzione B, che viene caricato all'indirizzo 12M,  $L=16M$  e  $B=12M$ .  
La mappa è  $SO(0, 8M)$ ,  $A(8M, 4M)$ ,  $B(12M, 16M)$ .
- 3) Va in esecuzione C, lo spazio residuo è solo 4M, quindi viene scaricato B e C  
viene caricato all'indirizzo 12M,  $L=12M$  e  $B=12M$ .  
La mappa è  $SO(0, 8M)$ ,  $A(8M, 4M)$ ,  $C(12M, 12M)$ .
- 4) Va in esecuzione D, lo spazio residuo è sufficiente a contenerlo e D viene  
caricato all'indirizzo 24M,  $L=6M$  e  $B=24M$ .  
La è  $SO(0, 8M)$ ,  $A(8M, 4M)$ ,  $C(12M, 12M)$ ,  $D(24M, 6M)$ .
- 5) Va in esecuzione A, che si trova già in memoria, quindi  $L=4M$  e  $B=8M$ .  
La mappa di memoria é quella del passo precedente.



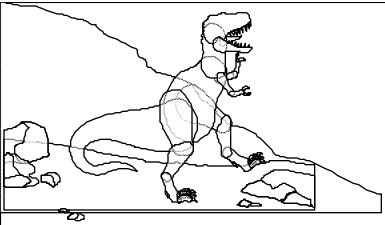
# Esercizio 7



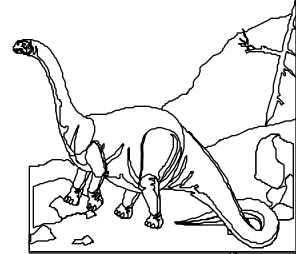
- Un computer ha indirizzi a 32 bit, e pagine su due livelli. Gli indirizzi virtuali sono composti da:
  - ☞ 9 bit per il numero della pagina principale
  - ☞ 11 bit per il numero della pagina di secondo livello
  - ☞ 12 bit per l'offset nella pagina
- Quanto sono grandi le pagine e quante pagine virtuali ci sono?



# Esercizio 7



- Numero di pagine:  $2^9 \times 2^{11} = 2^{20} \approx 10^6$
- Dimensione di ciascuna pagina:  $2^{12} = 4 \times 2^{10} = 4K$



# Esercizio 8

- Si consideri un processo A con 8 pagine logiche di 1K, e le seguenti configurazioni della tabella delle pagine

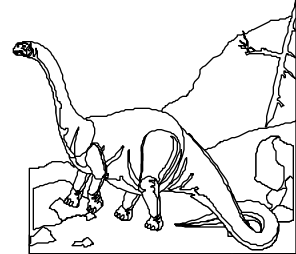
	Presente – Assente	P Fisica
0	1	435
1	0	0
2	1	171
3	0	0
4	1	248
5	1	732
6	0	0
7	1	222

e del TLB (memoria associativa) di 4 posizioni

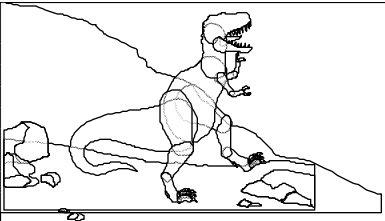
	Valido	Pagina Virtuale	Pagina Fisica
0	1	2	171
1	1	4	248
2	0	0	0
3	0	0	0

Discutere cosa accade quando A cerca di accedere alle seguenti parole:

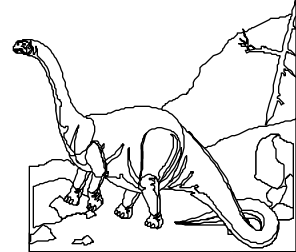
- 400
- 2100
- 1050



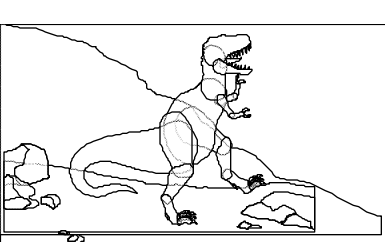
# Soluzione



- Si accede alla pagina 0, che non è presente nel TLB ma è in memoria, quindi il descrittore della pagina 0 viene caricato nel TLB nella posizione 2.
- Si accede alla pagina logica 2 il cui descrittore si trova nel TLB, quindi la traduzione dell'indirizzo non richiede modifiche del TLB
- Si accede alla pagina logica 1 che non è presente nel TLB e non è presente in memoria, verrà pertanto generato un page fault, dopodiché il corrispondente descrittore di pagina verrà caricato nel TLB.



# Esercizio 9

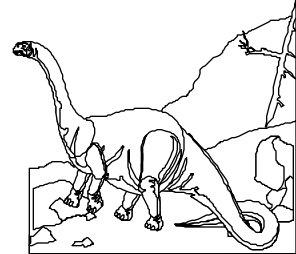


```
#define N 256
int a[N][N];
int i, j;
for (i=0; i < N; i++)
    for (j=0; j < N; j++)
        a[i][j] = 0;
```

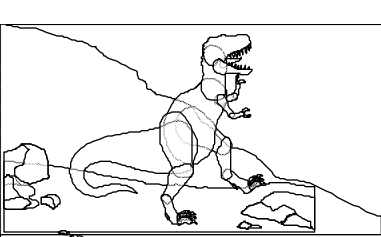
L'array è situato all'indirizzo virtuale 10240. La dimensione della pagina è 1024 e `sizeof(int) = 4`. Quale è la stringa di riferimento generata?

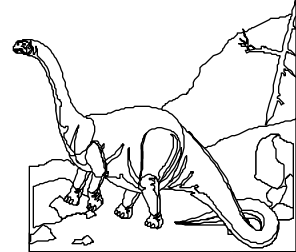
E se facciamo questa modifica?

```
for (i=0; i < N; j++)
    for (j=0; j < N; i++)
        a[i][j] = 0;
```

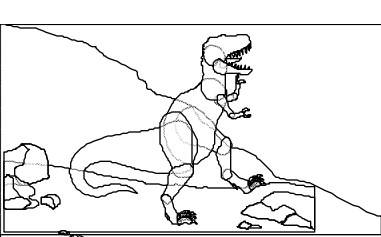


# Esercizio 10

- 
- Considerate un sistema di paginazione con la tabella delle pagine nella memoria centrale.
    1. Se il riferimento alla memoria richiede 200ns, dite quanto tempo richiede un riferimento alla memoria paginata.
    2. Se si aggiunge TLB e il 75% di tutti i riferimenti alla tabella delle pagine si trova in quest'ultimo, dite quant'è il tempo effettivo di accesso alla memoria, nel caso che:
      - 📄 tempo di reperimento nel TLB = 0;
      - 📄 tempo di reperimento nel TLB = 20ns.
    3. In quest'ultimo caso, quanto dovrebbe essere l'hit ratio del TLB, affinché si abbia un tempo di accesso effettivo di 240ns?

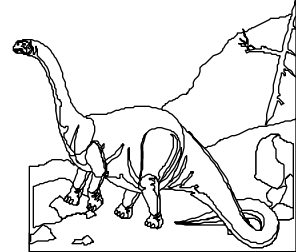


# Esercizio 10

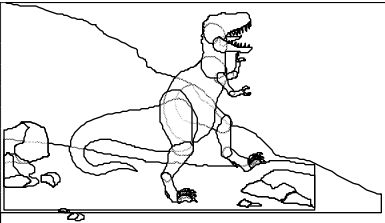
- 
- Considerate un sistema di paginazione con la tabella delle pagine nella memoria centrale.
    1. Se il riferimento alla memoria richiede 200ns, dite quanto tempo richiede un riferimento alla memoria paginata.
    2. Se si aggiunge TLB e il 75% di tutti i riferimenti alla tabella delle pagine si trova in quest'ultimo, dite quant'è il tempo effettivo di accesso alla memoria, nel caso che:
      - ☞ tempo di reperimento nel TLB = 0;
      - ☞ tempo di reperimento nel TLB = 20ns.
    3. In quest'ultimo caso, quanto dovrebbe essere l'hit ratio del TLB, affinché si abbia un tempo di accesso effettivo di 240ns?

## Soluzione

1. Per accedere ad un elemento in memoria, bisogna prima accedere alla tabella delle pagine => 400ns
2.  $EAT = (\gamma + \varepsilon) \alpha + (2\gamma + \varepsilon)(1 - \alpha)$ 
  1.  $\alpha = 0,75, \gamma = 200ns, \varepsilon = 0 \Rightarrow$   
 $EAT = 200ns * 0,75 + 400ns * 0,25 = 250ns$
  2.  $\alpha = 0,75, \gamma = 200ns, \varepsilon = 20ns \Rightarrow$   
 $EAT = 220ns * 0,75 + 420ns * 0,25 = 270ns$
3.  $220\alpha + 420(1 - \alpha) = 240 \Rightarrow \alpha = 90\%$



# Esercizio 11



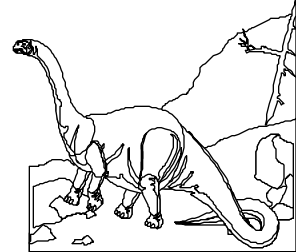
## ■ Detti:

- ☞  $s$  dimensione media processo,
- ☞  $p$  dimensione pagine,
- ☞  $e$  dimensione di un elemento nella tabella delle pagine,  
ciascun processo occupa  $s/p$  pagine e la tabella delle pagine  $se/p$ .

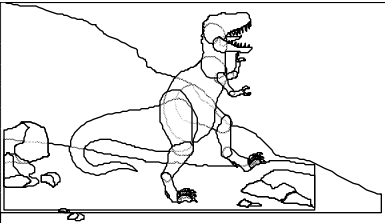
Poiché la frammentazione è  $p/2$ , il costo (spazio) della paginazione è dato da:

$$\text{costo} = se/p + p/2$$

Calcolare la dimensione  $p$  delle pagine ottimale per  $s = 128K$ ,  $e = 8K$ .



# Esercizio 11



## ■ Detti:

- ☞  $s$  dimensione media processo,
- ☞  $p$  dimensione pagine,
- ☞  $e$  dimensione di un elemento nella tabella delle pagine,  
ciascun processo occupa  $s/p$  pagine e la tabella delle pagine  $se/p$ .

Poiché la frammentazione è  $p/2$ , il costo (spazio) della paginazione è dato da:

$$\text{costo} = se/p + p/2$$

Calcolare la dimensione  $p$  delle pagine ottimale per  $s = 128K$ ,  $e = 8K$ .

Per ottenere il minimo, deriviamo rispetto a  $p$ :

$$-se/p^2 + 1/2 = 0$$

da cui  $p = \sqrt{2se}$      $p = 1448$  byte

