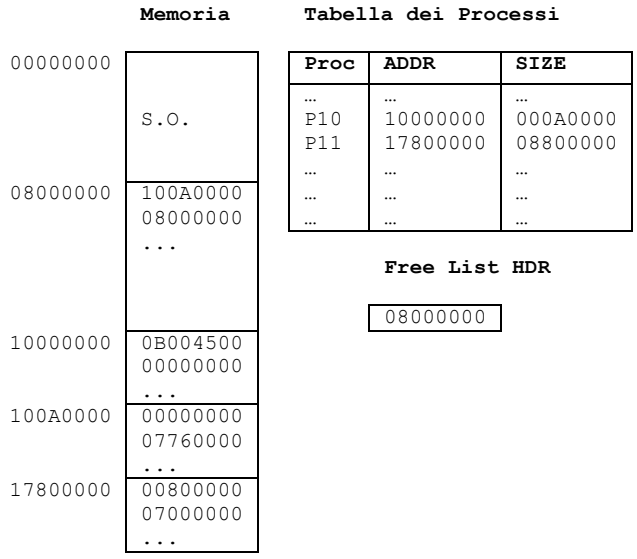
PROGRAMMAZIONE DI SISTEMA

Esercizi proposti e svolti di teoria
Tratti da compiti di esame

Gestione Memoria 1 (esercizi relativi a cap 9)

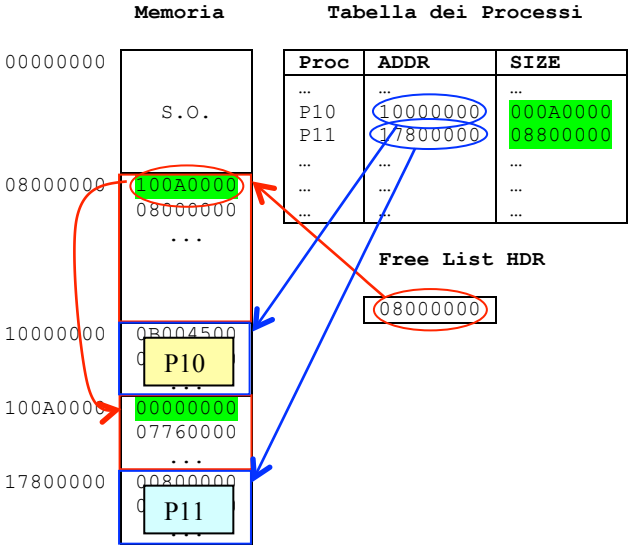
1. *(5/7/2011)* Sia dato un sistema con memoria fisica di dimensione 512MByte, in cui si utilizza uno schema di gestione a partizioni (contigue) variabili con unità minima di allocazione della memoria di 64 Byte (ovvero lo spazio di memoria viene allocato in multipli di 64 byte). Al Sistema Operativo sono allocati in modo permanente i primi 128MByte di memoria.
   La tabella dei processi contiene, per ogni processo attivo, l'indirizzo iniziale (ADDR) e la dimensione (SIZE) della relativa partizione in memoria. La memoria viene allocata con strategia Worst-Fit. Le partizioni libere sono gestite mediante una lista linkata ordinata per dimensione decrescente, in cui ogni nodo rappresenta una partizione libera; i nodi della lista sono costituiti da due campi: **(puntatore alla partizione successiva**, **dimensioni della partizione,** entrambi rappresentati su 4 byte, dimensioni e indirizzi rappresentati in Byte, con valore 0 usato come puntatore nullo**)** e sono memorizzati nei primi byte della partizione che rappresentano.



Si supponga che ad un dato istante la tabella dei processi e il puntatore alla prima partizione libera contengano le informazioni rappresentate in figura. Si rappresentino le modifiche alle partizioni in memoria, alla tabella dei processi e alla Free List, in seguito all'attivazione di 2 nuovi processi, P12 e P13, che richiedano rispettivamente 25MB e 150MB di memoria, seguita dalla terminazione del processo P11.

SOLUZIONE

Configurazione iniziale



Ecco la configurazione iniziale che mette in evidenza puntatori e dimensione delle partizioni in memoria.

Si noti che le partizioni nella free list contengono, nella loro intestazione, dimensione e puntatore al prossimo elemento in lista. Per le partizioni allocate l'informazione contenuta può essere diversa (in quanto nelle partizioni si sono caricate informazioni relative ai processi).

In forma più compatta, la configurazione iniziale può essere rappresentata come segue

La memoria è suddivisa in 5 partizioni. La prima è allocata al Sistema Operativo, delle restanti 4 partizioni, 2 sono allocate ai processi P10 e P11, e 2 sono nella free list. Per le partizioni nella free list i primi 8 Byte contengono dimensione della partizione e puntatore alla successiva. Per le partizioni allocate, l'informazione contenuta può essere di altro tipo.

Si consiglia di mantenere indirizzi e dimensioni in esadecimale. L'elenco delle partizioni iniziali, con indirizzo di partenza e dimensione, è:

Esadecimale
(00000000,08000000), (08000000,08000000), (10000000,000A0000), (100A00000,07760000), (17800000,08800000)

Le partizioni iniziali sono

Free list: (08000000,08000000), (100A0000,07760000)
P10: (10000000,000A0000)
P11: (17800000,08800000)

Allocazione a P12.

Servono 25MB (in esadecimale 19MB = 01900000 B) , che vengono allocate dalla prima partizione nella free list, da cui resta una partizione libera (di 08000000 B – 01900000 B = 06700000 B). La nuova partizione libera va in fondo alla free list in quanto è di dimensione inferiore all'altra partizione libera. Si rappresentano solo le variazioni.

Free list: (100A0000,07760000), (09900000,06700000)
P12: (08000000,01900000)

Allocazione a P13.

Servono 150MB (in esadecimale 96MB = 09600000 B). Nessuna delle partizioni è grande a sufficienza. Sono possibili varie soluzioni, quali de-frammentazione (mediante rilocazione dei processi), swap-out di P10 o di P11. Si rappresenta la soluzione più semplice: swap-out di P10, che permette di generare un'unica partizione contigua libera di dimensione 0DF00000 B (data dalla somma 06700000+000A0000+07760000), in cui viene allocata la nuova partizione per P13.

Free list: (12F000000,04900000)
P13: (09900000,09600000)

Termine di P11.

La partizione allocata a P11 viene liberata, aumentando la dimensione dell'unico blocco nella free list.

Free list: (12F000000,0D100000)

A questo punto si potrebbe fare swap-in di P10…

2. *(1/9/2008)* Sia dato un sistema di memoria virtuale con paginazione, nel quale vengono indirizzati i Byte. Il sistema dispone di TLB (Translation Look-aside Buffer), su cui si misura sperimentalmente un "hit ratio" del 99 %. La tabella delle pagine ("page-table") viene realizzata con uno schema a due livelli, nel quale un indirizzo logico di 32 bit viene suddiviso (da MSB a LSB) in 3 parti: p1, p2, d, rispettivamente di 10 bit, 11 bit, 11 bit. Non si utilizzano ulteriori strutture dati (quali tabelle di hash o inverted page table) per velocizzare gli accessi.

    - Si dica che cosa si intende con "hit ratio"
    - Si illustri lo schema della page-table e la sua dimensione complessiva, per un processo P1 avente spazio di indirizzamento virtuale di 100 MByte.
    - Si calcolino frammentazione esterna e interna per il processo P1 (vedi punto precedente).
    - Supponendo che la memoria RAM abbia tempo di accesso di 300 ns, si calcoli il tempo effettivo di accesso (EAT) per il caso proposto (hit ratio = 99 %)

R	• Per il primo punto si veda la teoria (slides/libro). • Si tenga conto che 100 MByte < 128 MByte = 2^{27} Byte e che pagine e frame hanno dimensione 2Kbyte = 2^{11} Byte (ricavate da d: 11 bit). Ipotizzando che ogni entry nelle tabelle delle pagine utilizzi 32 bit (4 Byte), una tabella delle pagine di secondo livello avrà 2^{11} caselle (11 da p2) e dimensione 2^{11}*4 Byte, mentre una tabella delle pagine di primo livello avrà 2^5 caselle (servono solo 5 bit su 10 di p1) e dimensione di 2^5*4 Byte. Si omette il disegno. • La frammentazione esterna è 0 per definizione. Quella interna è, per un qualunque processo: o caso peggiore: 2 Kbyte o caso medio: 1 Kbyte per P1 la frammentazione interna è 0 perché lo spazio virtuale è multiplo di 2Kbyte. • EAT = (0.99 * 300 + 0.01 * 3 * 300) ns = 1.02 * 300ns = 306 ns ATTENZIONE: nel caso di TLB MISS ci sono 3 accessi non 2 (la page table e a due livelli, dunque 2 accessi per la PT più uno per l'accesso voluto in RAM)

2. *(25/6/2018)* Si descrivano brevemente vantaggi e svantaggi di una inverted page table (IPT), rispetto a una tabella delle pagine standard (eventualmente gerarchica). Sia dato un processo avente spazio di indirizzamento virtuale di 32 GB, dotato di 8GB di RAM, su una architettura a 64 bit (in cui si indirizza il Byte), con gestione della memoria paginata (pagine/frame da 1KB). Si vogliono confrontare una soluzione basata su tabella delle pagine standard (una tabella per ogni processo) e una basata su IPT. Si calcolino le dimensioni della tabella delle pagine (ad un solo livello) per il processo e della IPT. Si ipotizzi che il `pid` di un processo possa essere rappresentato su 16 bit. Si utilizzino 32 bit per gli indici di pagina e/o di frame. Si dica infine, utilizzando la IPT proposta (32 bit per un indice di pagina/frame), quale è la dimensione massima possibile per lo spazio di indirizzamento virtuale di un processo.

R

Vantaggi

- Risparmio di memoria: la tabella ha la dimensione della RAM fisica invece che quella dello spazio di indirizzamento virtuale
- C'e una unica tabella per tutti i processi (ogni frame è associato a un solo processo)

Svantaggi

- Il vero svantaggio è la lentezza, occorre cercare una pagina, anziché accedervi in modo diretto. Per questo di solito le IPT sono associate a tabelle di hash.

Per i calcoli delle dimensioni, si trascurano eventuali bit di validità/modifica.

Page Table standard:

N pagine = 32GB/1KB = 32M (corrisponde al numero di indici di frame nella page table)
|Page Table| = 32M*4B = 128MB

IPT

La tabella, unica per tutti i processi, ha dimensione fissa, in quanto contiene un indice di pagina per ogni frame in RAM (si consideri, per semplicità, di gestire tutta la RAM, compresa quella assegnata al sistema operativo). Ogni riga della IPT contiene un indice di pagina (4B) e il pid del processo (2B)

N frame = 8GB/1KB = 8M
|IPT| = 8M*(4B+2B) = 48MB

Spazio di indirizzamento virtuale

Il numero massimo di pagine virtuali di un processo è limitato dalla dimensione degli indici di pagina (32 bit): tale numero è quindi 4G. Siccome ogni pagina ha dimensione 1KB, lo spazio di indirizzamento virtuale ha dimensione massima 4G*1KB = 4TB.