PROGRAMMAZIONE DI SISTEMA

Esercizi proposti e svolti di teoria
Tratti da compiti di esame

Gestione Memoria 3 (esercizi relativi a cap 10)

1. *(25/6/2018)* Si consideri la seguente sequenza di riferimenti a pagine in memoria: 577517111434123.
   Si utilizzi un algoritmo di sostituzione pagine di tipo working-set (versione esatta) con finestra Δ = 3, assumendo che siano disponibili al massimo 3 frame. Determinare quali e quanti page fault (accessi a pagine non presenti nel resident set) e page out (rimozioni di pagine dal resident set) si verificheranno. Si richiede la visualizzazione (dopo ogni accesso) del resident set.

   Si vuole inoltre definire una misura di località del programma svolto, basato sulla "Reuse Distance". La Reuse Distance al tempo T_i (RD_i), in cui si accede alla pagina P_i, viene definita come il numero di pagine (distinte e diverse da P_i) a cui si è fatto accesso a partire dal precedente accesso a P_i (assumendo, convenzionalmente, per il primo accesso a una pagina, il numero totale di pagine cui si è fatto accesso sino a quell'istante). Ad esempio, al tempo 3, in cui si fa il secondo accesso alla pagina 5, RD_3 = 1, in quanto tra i due accessi alla pagina 5 si è fatto accesso, due volte, a una sola pagina (7). Dati i vari RD_i, se ne calcoli il valor medio RD_{avg}. La località del programma svolto viene definita come L = 1 / (1+ RD_{avg}). Si calcolino i valori RD_i, RD_{avg} e L.

   Utilizzare, per questa domanda, lo schema seguente per svolgere l'esercizio. Si sono già indicati i riferimenti e la Reuse Distance ai tempi 0 e 3.

   R. Si noti che sono disponibili al massimo 3 frame, quindi la finestra Δ = 3 comprende l'accesso corrente al tempo T_i. La scelta del frame nel resident set è arbitraria, sono quindi possibili altre soluzioni, con frame permutati.

Riferimenti	5	7	7	5	1	7	1	1	1	4	3	4	1	2	3
Resident Set	5	5	5	5	5	5				4	4	4	4	4	3
		7	7	7	7	7	7	7			3	3	3	2	2
					1	1	1	1	1	1	1		1	1	1
Page Fault	x	x			x					x	x		x	x	X
Page Out							x		x			x		x	X
RD	0	1	0	1	2	2	1	0	0	3	4	1	2	5	3

Numero totale di page fault: 8….. di page out: 5… RD_{avg}: 25/15=5/3=1,67 L: 1/(1+5/3)=3/8=0,375

2. *(24/7/2019) Sia data la stringa di riferimenti a pagine 3, 4, 1, (3, 1, 4, 4, 3, 1, 1)*10, in cui la sintassi (…)*n indica che la stringa tra parentesi viene ripetuta/iterata n volte (la stringa può ad esempio, derivare da un costrutto iterativo).*

   Si utilizzi un algoritmo di sostituzione pagine di tipo ***working-set*** (versione ***esatta***) con finestra di durata Δ = 3. Si supponga di denominare *page-out* la rimozione di una pagina dal resident set (in quanto esce dal working set). Si visualizzino nello schema che segue i riferimenti e il resident set dopo ogni riferimento, indicando i page-fault (accessi a pagine non presenti nel resident set) e i page-out. Si richiede la visualizzazione solo fino alle prime due iterazioni della sotto-stringa ripetuta 10 volte. *ATTENZIONE: ogni riga del resident set rappresenta un frame, quindi una pagina presente in un frame non può cambiare riga quando rimane nel resident set. Nel caso di page-fault senza page-out, si utilizzi il primo frame libero dall'alto. Nel caso di page-out e (contemporaneo) page-fault, viene riutilizzato il frame appena liberato (da page-out). Qualora page-out e page-fault siano relativi alla stessa pagina, l'algoritmo di sostituzione ne tiene conto, evitando sia page-out che page-fault.*

Tempo	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Riferimenti	3	4	1	3	1	4	4	3	1	1	3	1	4	4	3	1	1
Resident Set	3	3	3	3	3	3			1	1	1	1	1	1	3	3	3
		4	4	4		4	4	4	4				4	4	4	4	
			1	1	1	1	1	3	3	3	3	3	3			1	1
Page Fault	x	x	x			x		x	x				x		x	x	
Page Out					x		x	x		x				x	x		x

Il comportamento del resident set nelle 8 iterazioni non riportate sarà

- identico a quello dell'intervallo 10-16 (ripetuto 8 volte)? (SI/NO motivare)
  No, in quanto, pur se il periodo è di 7 tempi, le pagine 1 e 3 non si trovano nello stesso frame all'inizio e alla fine del periodo
- oppure ripeterà 4 volte l'intervallo 3-16? (SI/NO motivare)
  No, in quanto, pur essendo i 14 istanti l'unione dei primi due periodi di 7 istanti, il primo periodo ha un comportamento iniziale diverso per la presenza della pagina 4 al tempo 3 (il quarto)
- oppure avrà una diversa configurazione (quale?)?
  Si ripete 4 volte l'intervallo 3-16, a patto di eliminare la pagina 4 dal tempo 3 e il page-out al tempo 4. Oppure (detto in un altro modo) si ripete 8 volte l'intervallo 10-16, alternando ad ogni iterazione i contenuti del primo e del terzo frame.

Quanti page fault e page-out ci saranno in totale (comprese le 8 iterazioni mancanti?
9 PF e 7 PO nella parte visualizzata, 8*(3 PF e 3 PO) nella parte mancante: in totale 33 PF e 31 PO

3. *(6/9/2019)* Sia dato il frammento di programma rappresentato, che effettua un calcolo matriciale*:*

```
...
float M[512][512],V[512];
...
for (i=0; i<512; i++) {
 V[i]=0;
 for (j=0; j<=i; j++) {
   if (i%2==0) {
     V[i] += M[i][j];
   }
   else {
     V[i] -= M[i][511-j];
   }
 }
}
...
```

Il codice macchina generato da tali istruzioni viene eseguito in un sistema con memoria virtuale gestita mediante paginazione, con **pagine di 2Kbyte**, utilizzando come politica di sostituzione pagine la **SECOND CHANCE**. Si sa che un `float` ha dimensione **32 bit** e che le istruzioni in codice macchina, corrispondenti al frammento di programma visualizzato, sono contenute in una sola pagina. Si supponga che `M` e `V` siano allocati ad indirizzi logici contigui (prima `M`, poi `V`), a partire dall'indirizzo logico 0x5524AE00. La matrice `M` è allocata secondo la strategia "row major", cioè per righe (prima riga, seguita da seconda riga, …).

- Quante pagine (e frame) sono necessarie per contenere la matrice e il vettore ?

|V| = 512*sizeof(float) = 2KB = 2KB/2KB pagine = 1 pagina
|M| = 512*512*sizeof(float) = 1MB = 1MB/2KB pagine = 0.5K pagine = 512 pagine
L'indirizzo di partenza 0x5524AE00 NON è multiplo di pagina (dovrebbe terminare con 11 bit a 0), ma inizia a ¾ di pagina. quindi sono necessarie correzioni: V sta su 2 pagine e M su 513 (la prima in comune con V).

- Ipotizzando che le variabili i, j siano allocate in registri della CPU, quanti accessi in memoria (in lettura e scrittura) fa il programma proposto, per accedere a dati (non vanno conteggiati gli accessi a istruzioni) ?

Notazione: N_i numero iterazioni del for esterno
N_j numero iterazioni del for interno

Soluzione

```
for (i=0;i<512;i++){ // Ni = 512 iterazioni
V[i]=0;              // 1 Write per iterazione, in totale Ni(512) write
for (j=0;j<=i;j++){  // Nj = i+1 iterazioni, ripetute Ni(512) volte (i cambia)
V[i] += M[i][j];     // Nj = Σi=0..511(i+1) = 512*(512+1)/2 = 128K+256 = 131328
    oppure           // per ogni iterazione 2 Read e 1 Write, in totale
V[i] += M[i][511-j]; // 2*Nj read, Nj write
```

ATTENZIONE: Il calcolo fatto sopra è ***sufficiente***. Per completezza, si effettua sotto il calcolo dettagliato delle iterazioni e degli accessi differenziati tra `i` pari e `i` dispari.

```
V[i] += M[i][j];      // iterazioni con i pari: Nj,0 = 131328/2 = 65664 (*)
                      // per ogni iterazione 2 Read, 1 Write, in totale
                      // 2*Nj,0 read, Nj,0 write
V[i] += M[i][511-j];  // iterazioni con i dispari: Nj,1 = 131328/2 65664 (*)
                      // per ogni iterazione 2 Read, 1 Write, in totale
                      // 2*Nj,1 read, Nj,1 write
```

(*) il risultato è approssimato. Si assume che il numero di iterazioni su j con `i` pari e con `i` dispari coincidano. Il calcolo esatto sarebbe (assumendo `k=i/2`):

$N_{j,0} = \Sigma_{k=0..255}(2k+1) = 1+3+5\ldots+511 = 2*\Sigma_{k=0..255}(k+1)-256 = 256*(256+1)-256 = 256^2 = 65536$

$N_{j,1} = \Sigma_{k=0..255}(2k+2) = 2+4+6\ldots+512 = 2*\Sigma_{k=0..255}(k+1) = 256*(256+1) = 256*257 = 65792$

Soluzione alternativa

Un modo (forse) più semplice di ragionare è il seguente: il numero totale di iterazioni N_j coincide con la dimensione di una sottomatrice triangolare inferiore (comprende la diagonale). In realtà le iterazioni con `i` pari percorrono una riga a partire da sinistra, quelle con `i` dispari a partire da destra. Tuttavia il numero totale di iterazioni non cambia: (dimensione della matrice)/2 + (mezza diagonale) = 512*512/2 + 256.
(Calcolo dettagliato `i` pari / `i` dispari, NON richiesto) Le iterazioni da destra sono leggermente inferiori a quelle da sinistra (per ogni coppia i-pari i-dispari, la seconda fa una iterazione in più, complessivamente le iterazioni dispari superano le pari di 256). Quindi vale l'equazione:

$2*N_{j,0}+256 = N_j = 128K+256$
$2*N_{j,0} = 128K$
$N_{j,0} = 64K$
$N_{j,1} = N_{j,0} + 256 = 64K + 256$

- Detto N_T il numero totale di accessi a dati in memoria, ed N_L il numero di accessi a dati nella stessa pagina di uno dei precedenti 10 accessi, si definisca come **località** del programma per i dati il numero $L = N_L/N_T$. Si calcoli la località del programma proposto.

SI SVOLGE L'ESERCIZIO SUPPONENDO ALLINEAMENTO A PAGINA. LA SOLUZIONE ESATTA RICHIEDEREBBE MINIME CORREZIONI

Siccome ogni riga di M occupa esattamente una pagina, il fatto di percorrere la riga in avanti o all'indietro non ha alcun impatto su località e page fault. Località e page fault sono quindi gli stessi che si avrebbero con l'algoritmo semplificato:

```
for (i=0; i<512; i++) {
   V[i]=0;
   for (j=0; j<=i; j++) {
     V[i] += M[i][j];
   }
}
```

Accessi a M (in lettura)

Per ogni valore di `i` (per ogni riga di M), l'unico accesso non locale è il primo (j=0) perché accede a una nuova pagina (nuova riga), a cui non si era ancora fatto accesso. Gli accessi non locali sono quindi 512

Accessi a V (in lettura o scrittura)

V sta tutto in una sola pagina. L'unico accesso non locale è il primo. Quindi 1 accesso non locale

In totale gli accessi NON locali sono 1+512 = 513

$N_T = N_i$ (azzeramenti del vettore) $+ 3*N_j$ (iterazioni interne: 2 Read e 1 Write)

$N_L = N_T - 513$ (accessi NON locali)

$L = L = N_L/N_T = (N_T - 513)/N_T = (512 + 3*(128K + 256) - 513) / (512 + 3*(128K + 256)) =$
$= 1 - 513 / (512 + 3*(128K + 256)) \approx 1 - 512/3*128K = 1 - 1/3*256 =$
$= 1 - 0{,}0013 = 9{,}9987$

- Calcolare il numero di page fault generati dal programma proposto, supponendo che siano allocati per esso **10 frame**, di cui uno utilizzato (già all'inizio dell'esecuzione) per le istruzioni. (motivare la risposta)

1 solo page fault per V *(2 tenendo conto del non allineamento)*
Su M 1 page fault per ogni riga/pagina. Poi si continua a farvi accesso.
In totale 1+512 = 513 page fault *(2 + 512 tenendo conto del non allineamento)*