

PROGETTO DI SISTEMI OPERATIVI
Ingegneria Informatica
6 maggio 2011
(teoria)

(si prega di rispondere descrivendo i passaggi e i risultati intermedi)

1. (6 punti) Sia dato un processore dotato di TLB (Translation Lookaside Buffer). Si supponga che un accesso a memoria RAM costi 180 ns, che un accesso alla TLB richieda 20 ns, e che la traduzione da indirizzi virtuali a fisici utilizzi una tabella delle pagine a due livelli.

- Quale deve essere la probabilità di successo (hit ratio) negli accessi alla TLB, in modo da ottenere un tempo medio di accesso a memoria di 220 ns ?

Sia dato un processo con spazio di indirizzamento virtuale di 1MB. Supponendo che la TLB abbia 16 righe, che frame e pagine abbiano dimensione 1KB, e che gli indirizzi siano su 32 bit.

- Quale è la hit ratio della TLB, nel caso di accesso "casuale" (il programma non ha località di accessi, ma può accedere, in ogni momento ad uno qualunque dei suoi indirizzi logici) ?
- Quale è la hit ratio della TLB, per un programma strettamente "sequenziale", che legge cioè tutti gli indirizzi logici in sequenza, una sola volta ciascuno ?

2. (6 punti) Quattro job (processi), identificati dalle lettere A-D, arrivano all'elaboratore agli istanti di tempo 0, 4, 2, 8, rispettivamente. I processi hanno tempi di esecuzione di (2-3-2), (5-5-2), (3-6) e (4-2) unità di tempo, rispettivamente. Le durate sono state espresse secondo il formato (a-b-...) per indicare fari CPU burst separati da richieste di I/O. Si supponga che tutte le richieste di I/O facciano riferimento allo stesso dispositivo IO#4, che serve le richieste di I/O secondo un criterio FIFO (cioè secondo l'ordine di arrivo), e completa ognuna delle richieste di I/O in 4 unità di tempo. Descrivere (mediante diagramma di Gantt) la sequenza di esecuzione dei job su un sistema dotato di 2 CPU e calcolare i tempi individuali di attesa (tempo trascorso nella coda "ready" e di turnaround (per ognuno dei processi), trascurando i tempi dovuti allo scambio di contesto. Si supponga una schedulazione con preemption di tipo round-robin, con quanto di tempo 4. Si rappresenti, oltre al diagramma di esecuzione sulle CPU, il diagramma relativo ai servizi sul dispositivo IO#4.

3. (6 punti) Sia dato un file system basato su File Allocation Table (FAT). Si supponga, per semplicità descrittiva, che il file system contenga unicamente 16 blocchi di 512 Byte (numerati da 0 a 15), nel quale siano immagazzinati 3 file ("a.txt", "b.dat", "c.exe"), allocati nei blocchi (7,10,2,13), (4,5,1) e (12,14,15), rispettivamente. La freelist è (11, 0, 3, 8, 9, 6). Si rappresenti la FAT corrispondente a tale allocazione. Supponendo che il file "c.exe" venga cancellato, si rappresenti la FAT dopo tale cancellazione. Supponendo che il file "b.dat" contenga una sequenza di record a lunghezza fissa, ognuno di 50 byte, si dica in quale blocco, e a quale offset, si trova il record n. 21 (i record sono numerati a partire da 0).

4. (6 punti) Siano date due funzioni di lettura da dispositivi di I/O, una di tipo sincrono (readSynch) e una di tipo asincrono (readAsynch). Nel seguito sono proposte alcune chiamate alle funzioni:

```
buf_t *buf1, *buf2;
...
for (i=0; i<n; i++) {
    readSynch(fp1, buf1, sizeof(buf_t),...);
    readAsynch(fp2, buf2, sizeof(buf_t),...);
    a[i] = f(buf1);
    b[i] = g(buf2);
}
...
```

Il programma non è completo (si sono utilizzati i ... per indicare parti mancanti). Le funzioni `readSynch` e `readAsynch` realizzano letture sincrone/asincrone tra un file ed un buffer, di cui si forniscono il puntatore e la dimensione. Si dica se il programma può funzionare, nella forma proposta, e/o come va modificato, o a quali condizioni lo si può utilizzare, affinché le chiamate ad I/O funzionino correttamente. Si chiede, in particolare, di discutere la consistenza dei dati presenti in `buf1` e `buf2` dopo le letture.

5. (6 punti) Sia riportano nel seguito (retro del foglio) alcune parti del file `dumbvm.c` di OS161. Le righe sono numerate. Si sono volutamente riportate alcune parti ritenute più significative. Si spieghi brevemente lo scopo delle funzioni riportate. Si dica poi come viene gestito in tale contesto l'allocazione di uno spazio di memoria

virtuale. E' possibile liberare (utilizzando `as_destroy`) una parte di memoria, affinché sia disponibile per una nuova allocazione ? In caso positivo, si dica perché; in caso negativo, si suggerisca una modifica (non è richiesto il codice C, ma sono un'indicazione di strutture dati e/o algoritmi) ai programmi sotto-elencati, tale da consentire tale liberazione e ri-allocazione.

```
00156 struct addrspace *
00157 as\_create(void)
00158 {
00159     struct addrspace *as = kmalloc(sizeof(struct addrspace));
00160     if (as==NULL) {
00161         return NULL;
00162     }
00163
00164     as->as_vbase1 = 0;
00165     as->as_pbase1 = 0;
00166     as->as_npages1 = 0;
00167     as->as_vbase2 = 0;
00168     as->as_pbase2 = 0;
00169     as->as_npages2 = 0;
00170     as->as_stackpbase = 0;
00171
00172     return as;
00173 }
00174
00175 void
00176 as\_destroy(struct addrspace *as)
00177 {
00178     kfree(as);
00179 }
00180
00181 void
00182 ...
00197 int
00198 as\_define\_region(struct addrspace *as, vaddr\_t vaddr, size_t sz,
00199                  int readable, int writeable, int executable)
00200 {
00201     size_t npages;
00202
00203     /* Align the region. First, the base... */
00204     sz += vaddr & ~(vaddr\_t)PAGE\_FRAME;
00205     vaddr &= PAGE\_FRAME;
00206
00207     /* ...and now the length. */
00208     sz = (sz + PAGE\_SIZE - 1) & PAGE\_FRAME;
00209
00210     npages = sz / PAGE\_SIZE;
00211
00212     ...
00216
00217     if (as->as_vbase1 == 0) {
00218         as->as_vbase1 = vaddr;
00219         as->as_npages1 = npages;
00220         return 0;
00221     }
00222
00223     if (as->as_vbase2 == 0) {
00224         as->as_vbase2 = vaddr;
00225         as->as_npages2 = npages;
00226         return 0;
00227     }
00228
00229     /*
00230      * Support for more than two regions is not available.
00231      */
00232     kprintf("dumbvm: Warning: too many regions\n");
00233     return EUNIMP;
00234 }
```

PROGETTO DI SISTEMI OPERATIVI

Ingegneria Informatica (a.a. 2008/2009 e precedenti)

6 maggio 2011

(teoria)

(si prega di rispondere descrivendo i passaggi e i risultati intermedi)

1. (6 punti) Sia dato un processore dotato di TLB (Translation Lookaside Buffer). Si supponga che un accesso a memoria RAM costi 180 ns, che un accesso alla TLB richieda 20 ns, e che la traduzione da indirizzi virtuali a fisici utilizzi una tabella delle pagine a due livelli.

- Quale deve essere la probabilità di successo (hit ratio) negli accessi alla TLB, in modo da ottenere un tempo medio di accesso a memoria di 220 ns ?

Sia dato un processo con spazio di indirizzamento virtuale di 1MB. Supponendo che la TLB abbia 16 righe, che frame e pagine abbiano dimensione 1KB, e che gli indirizzi siano su 32 bit.

- Quale è la hit ratio della TLB, nel caso di accesso "casuale" (il programma non ha località di accessi, ma può accedere, in ogni momento ad uno qualunque dei suoi indirizzi logici) ?
- Quale è la hit ratio della TLB, per un programma strettamente "sequenziale", che legge cioè tutti gli indirizzi logici in sequenza, una sola volta ciascuno ?

2. (6 punti) Quattro job (processi), identificati dalle lettere A-D, arrivano all'elaboratore agli istanti di tempo 0, 4, 2, 8, rispettivamente. I processi hanno tempi di esecuzione di (2-3-2), (5-5-2), (3-6) e (4-2) unità di tempo, rispettivamente. Le durate sono state espresse secondo il formato (a-b-...) per indicare fari CPU burst separati da richieste di I/O. Si supponga che tutte le richieste di I/O facciano riferimento allo stesso dispositivo IO#4, che serve le richieste di I/O secondo un criterio FIFO (cioè secondo l'ordine di arrivo), e completa ognuna delle richieste di I/O in 4 unità di tempo. Descrivere (mediante diagramma di Gantt) la sequenza di esecuzione dei job su un sistema dotato di 2 CPU e calcolare i tempi individuali di attesa (tempo trascorso nella coda "ready" e di turnaround (per ognuno dei processi), trascurando i tempi dovuti allo scambio di contesto. Si supponga una schedulazione con preemption di tipo round-robin, con quanto di tempo 4. Si rappresenti, oltre al diagramma di esecuzione sulle CPU, il diagramma relativo ai servizi sul dispositivo IO#4.

3. (6 punti) Sia dato un file system basato su File Allocation Table (FAT). Si supponga, per semplicità descrittiva, che il file system contenga unicamente 16 blocchi di 512 Byte (numerati da 0 a 15), nel quale siano immagazzinati 3 file ("a.txt", "b.dat", "c.exe"), allocati nei blocchi (7,10,2,13), (4,5,1) e (12,14,15), rispettivamente. La freelist è (11, 0, 3, 8, 9, 6). Si rappresenti la FAT corrispondente a tale allocazione. Supponendo che il file "c.exe" venga cancellato, si rappresenti la FAT dopo tale cancellazione. Supponendo che il file "b.dat" contenga una sequenza di record a lunghezza fissa, ognuno di 50 byte, si dica in quale blocco, e a quale offset, si trova il record n. 21 (i record sono numerati a partire da 0).

4. (6 punti) Siano date due funzioni di lettura da dispositivi di I/O, una di tipo sincrono (readSynch) e una di tipo asincrono (readAsynch). Nel seguito sono proposte alcune chiamate alle funzioni:

```
buf_t *buf1, *buf2;
...
for (i=0; i<n; i++) {
    readSynch(fp1, buf1, sizeof(buf_t),...);
    readAsynch(fp2, buf2, sizeof(buf_t),...);
    a[i] = f(buf1);
    b[i] = g(buf2);
}
...
```

Il programma non è completo (si sono utilizzati i ... per indicare parti mancanti). Le funzioni readSynch e readAsynch realizzano letture sincrone/asincrone tra un file ed un buffer, di cui si forniscono il puntatore e la dimensione. Si dica se il programma può funzionare, nella forma proposta, e/o come va modificato, o a quali condizioni lo si può utilizzare, affinché le chiamate ad I/O funzionino correttamente. Si chiede, in particolare, di discutere la consistenza dei dati presenti in buf1 e buf2 dopo le letture.

5. (6 punti) Si descriva brevemente il meccanismo di gestione di un driver generico per i dispositivi di I/O, identificando le funzioni svolte da processo utente, processo driver e interrupt handler. Che cosa si intende con il termine *interrupt vector* (vettore di interruzione).