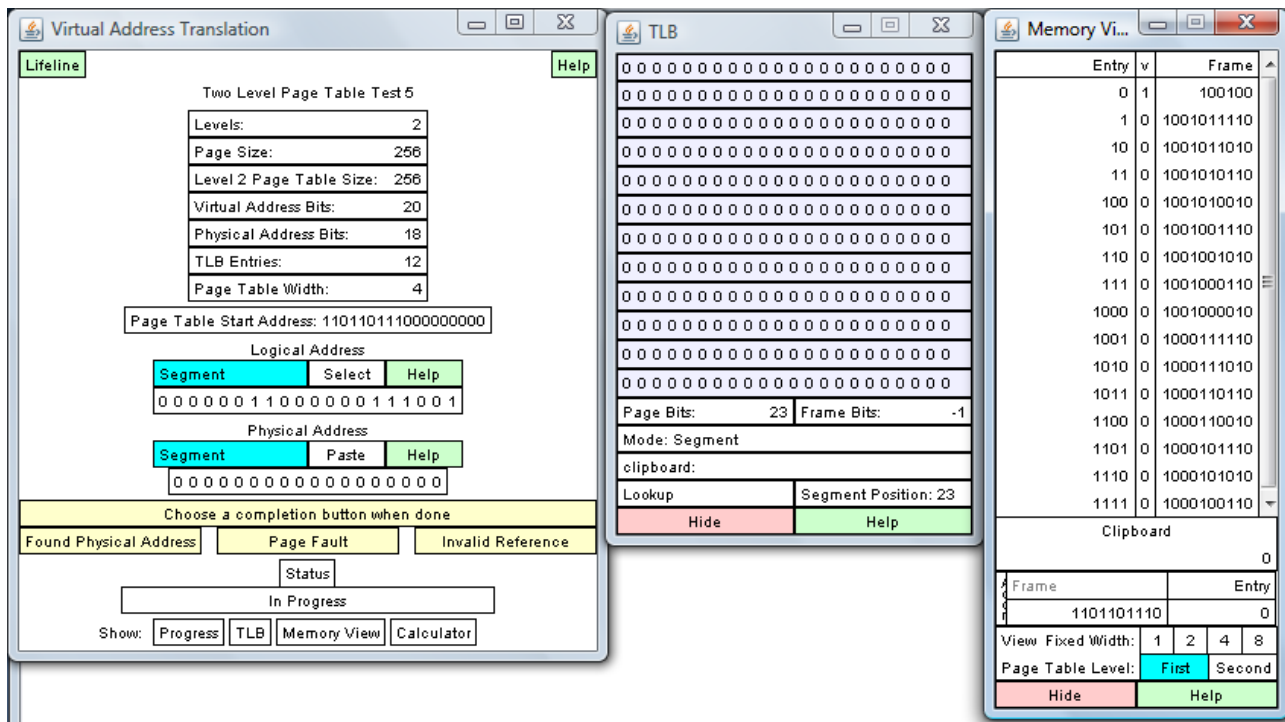


PROGETTO DI SISTEMI OPERATIVI
Ingegneria Informatica
13 luglio 2010
(teoria)

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

1. (6 punti) Sia data la figura che segue, nella quale si rappresenta una visualizzazione del simulatore di traduzione da indirizzi logici a fisici di UTSA.



- Si determinino le dimensioni degli spazi di indirizzamento logico e fisico.
 - Si dica come va segmentato l'indirizzo logico.
 - Si dica se l'indirizzo logico proposto genera TLB hit, oppure miss. Perché? A quali bit della TLB si accede in lookup?
 - Si dica se è necessario l'accesso alle page table. In caso affermativo, si dica a quale locazione della page table di primo livello e di secondo livello si accede.
 - Si supponga che la page table di secondo livello contenga, nella prima riga, il valore binario 100000000 e, ad ogni riga successiva, l'indirizzo di quella precedente incrementato di 1. Tutti i bit di validità di tale pagina sono a 1. Si determini se per l'indirizzo logico proposto si genererà page-fault oppure si troverà l'indirizzo fisico corrispondente (in tal caso si dica quale indirizzo fisico verrà generato).
2. (6 punti) Si scriva una funzione in linguaggio C in grado di gestire richieste di accesso a disco secondo la politica SSTN (Shortest Seek Time Next). La funzione, il cui prototipo è riportato in seguito

```
int sstnNextBlk (int currBlk, int *reqArray, int nReq);
```

riceve come parametri, rispettivamente, la posizione corrente (numero di blocco) della testina, il vettore delle richieste di accesso (a blocchi), il numero di richieste presenti nel vettore. La funzione deve calcolare e ritornare il numero del blocco selezionato, tra le richieste presenti in reqArray, secondo la politica SSTN. Supponendo che la funzione venga chiamata una prima volta con parametri: 30, {23,49,35}, 3, si dica quale valore viene ritornato. Dopo tale chiamata (e prima della successiva), arrivano due ulteriori richieste per i blocchi 55 e 27. Si dica con quali parametri verrà chiamata, la seconda volta, e quale sarà il valore di ritorno.

3. (6 punti) Quattro job (processi), identificati dalle lettere A-D, arrivano all'elaboratore agli istanti di tempo 0, 6, 2, 12, rispettivamente. I processi hanno tempi di esecuzione di 6, 15, 12 e 10 unità di tempo, rispettivamente. I processi B e C, effettuano una richiesta di I/O ogni 7 unità di tempo di esecuzione. Si supponga che tali richieste di I/O siano soddisfatte in 6 unità di tempo. Descrivere (mediante diagramma di Gantt) la sequenza di esecuzione

dei job su un sistema dotato di 1 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: si effettuino 2 simulazioni, rispettivamente con quanto di tempo 4 e 8. Si dica infine quale dei due quanti di tempo permette di minimizzare il tempo medio (per i 4 processi) di attesa. Si motivi la risposta calcolando il tempo medio di attesa per i due casi.

4. (6 punti) Sia riportano nel seguito alcune parti della funzione OS161 `load_elf`. Le righe sono numerate. Si sono volutamente riportate le parti ritenute più significative. Si spieghi brevemente lo scopo di tale funzione. Si descriva successivamente il ruolo svolto da ognuna delle parti sotto-riportate.

```
00090 load_elf(struct vnode *v, vaddr_t *entrypoint)
00091 {
00092     Elf_Ehdr eh; /* Executable header */
00093     Elf_Phdr ph; /* "Program header" = segment header */
00094     int result, i;
00095     struct uio ku;

00101     mk_kuio(&ku, &eh, sizeof(eh), 0, UIO_READ);
00102     result = VOP_READ(v, &ku);

00152     for (i=0; i<eh.e_phnum; i++) {
00153         off_t offset = eh.e_phoff + i*eh.e_phentsize;
00154         mk_kuio(&ku, &ph, sizeof(ph), offset, UIO_READ);
00155
00156         result = VOP_READ(v, &ku);
00178         result = as_define_region(curthread->t_vmspace,
00179                                   ph.p_vaddr, ph.p_memsz,
00180                                   ph.p_flags & PF_R,
00181                                   ph.p_flags & PF_W,
00182                                   ph.p_flags & PF_X);
00186     }
00188     result = as_prepare_load(curthread->t_vmspace);
00197     for (i=0; i<eh.e_phnum; i++) {
00198         off_t offset = eh.e_phoff + i*eh.e_phentsize;
00199         mk_kuio(&ku, &ph, sizeof(ph), offset, UIO_READ);
00200
00201         result = VOP_READ(v, &ku);
00223         result = load_segment(v, ph.p_offset, ph.p_vaddr,
00224                                ph.p_memsz, ph.p_filesz,
00225                                ph.p_flags & PF_X);
00229     }
00230 }
00239 }
```

5. (6 punti) Si descriva, nel contesto della gestione dei dispositivi di I/O, la differenze tra I/O sincrono e asincrono, bloccante e non bloccante. I/O non bloccante e asincrono sono equivalenti ? Sia dato il frammento di codice C seguente

```
buf_t *buf = malloc(...);
for (i=0; i<ndati; i++) readAsynch(fp1, buf, sizeof(buf_t),...);
...
for (i=0; i<ndati; i++) writeAsynch(fp2, buf, sizeof(buf_t),...);
```

Il programma non è completo (si sono utilizzati i ... per indicare parti mancanti). Le funzioni `readAsynch` e `writeAsynch` realizzano letture/scritture 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 asincrono funzionino correttamente.

PROGETTO DI SISTEMI OPERATIVI
Ingegneria Informatica (a.a. 2008/2009 e precedenti)
13 luglio 2010

(teoria)

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

1. (6 punti) Sia dato un sistema di memoria virtuale con paginazione. Gli indirizzi virtuali (su 32 bit) sono formati (partendo dal MSB) da 12 bit per il segmento, 10 bit per la pagina e 10 bit di offset. Si supponga che il segment table base register e il page table base register contengano, rispettivamente, i valori (espressi in esadecimale su 32 bit) 00A40000 e 02050000, e che all'indirizzo virtuale 0C4A2805 corrisponda l'indirizzo reale 01BE4805. Si calcolino i contenuti della segment table e della page table, per le sole parole interessate alla traduzione del precedente indirizzo. Si illustri infine la sequenza di letture in memoria effettuate (indirizzo e dato letto) per effettuare la traduzione. N.B. per semplicità si ipotizzi che non esistano TLB né inverted page table. (NB. Si possono ipotizzare gestioni di segmentazione paginata sia del tipo Pentium, con indirizzo lineare intermedio, che di tipo gerarchico puro).
2. (6 punti) Si scriva una funzione in linguaggio C in grado di gestire richieste di accesso a disco secondo la politica SSTN (Shortest Seek Time Next). La funzione, il cui prototipo è riportato in seguito:

```
int sstnNextBlk (int currBlk, int *reqArray, int nReq);
```

riceve come parametri, rispettivamente, la posizione corrente (numero di blocco) della testina, il vettore delle richieste di accesso (a blocchi), il numero di richieste presenti nel vettore. La funzione deve calcolare e ritornare il numero del blocco selezionato, tra le richieste presenti in reqArray, secondo la politica SSTN. Supponendo che la funzione venga chiamata una prima volta con parametri: 30, {23,49,35}, 3, si dica quale valore viene ritornato. Dopo tale chiamata (e prima della successiva), arrivano due ulteriori richieste per i blocchi 55 e 27. Si dica con quali parametri verrà chiamata, la seconda volta, e quale sarà il valore di ritorno.

3. (6 punti) Quattro job (processi), identificati dalle lettere A-D, arrivano all'elaboratore agli istanti di tempo 0, 6, 2, 12, rispettivamente. I processi hanno tempi di esecuzione di 6, 15, 12 e 10 unità di tempo, rispettivamente. I processi B e C, effettuano una richiesta di I/O ogni 7 unità di tempo di esecuzione. Si supponga che tali richieste di I/O siano soddisfatte in 6 unità di tempo. Descrivere (mediante diagramma di Gantt) la sequenza di esecuzione dei job su un sistema dotato di 1 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: si effettuino 2 simulazioni, rispettivamente con quanto di tempo 4 e 8. Si dica infine quale dei due quanti di tempo permette di minimizzare il tempo medio (per i 4 processi) di attesa. Si motivi la risposta calcolando il tempo medio di attesa per i due casi.
4. (6 punti) Si descriva, nel contesto della gestione dei dispositivi di I/O, la differenza tra I/O sincrono e asincrono, bloccante e non bloccante. I/O non bloccante e asincrono sono equivalenti? Sia dato il frammento di codice C seguente:

```
buf_t *buf = malloc(...);  
for (i=0; i<ndati; i++) readAsynch(fp1, buf, sizeof(buf_t),...);  
...  
for (i=0; i<ndati; i++) writeAsynch(fp2, buf, sizeof(buf_t),...);
```

Il programma non è completo (si sono utilizzati i ... per indicare parti mancanti). Le funzioni readAsynch e writeAsynch realizzano letture/scritture asincrone da un file ad 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 asincrono funzionino correttamente.

5. (6 punti) Si descriva in breve il funzionamento di una inverted page table. Si rappresenti lo schema di traduzione di indirizzi, utilizzando tale tabella, e si fornisca un esempio di traduzione, con indirizzo virtuale, indirizzo fisico e parte interessata della inverted page table.