Corso di Laurea in Informatica, A.A. 2021/2022

Architetture degli Elaboratori e Sistemi Operativi (AESO corso A e B)

Primo Appello – 24 maggio 2022

Scrivere in modo comprensibile e chiaro rispondendo alle domande/esercizi riportati in questo foglio. Sviluppare le soluzioni dei primi due esercizi (prima parte) in un foglio separato rispetto alla soluzione degli ultimi due esercizi (seconda parte), in modo da facilitare la correzione da parte dei docenti. Riportare su tutti i fogli consegnati nome, cognome, numero di matricola e corso (A o B).

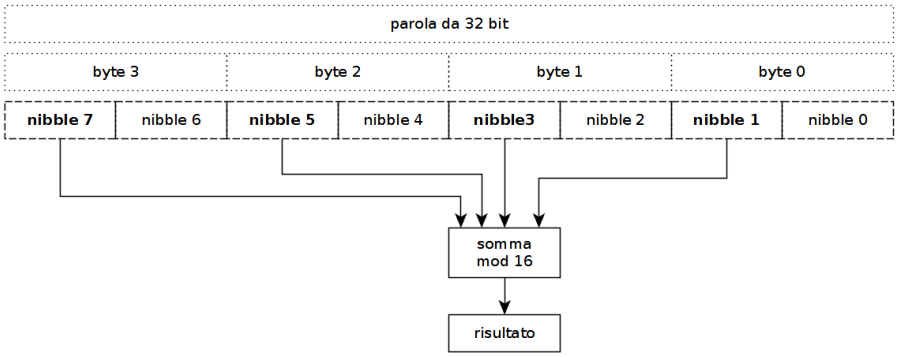
Parte 1

Esercizio 1

Si fornisca il codice Assembler ARMv7 di una funzione con tipo

```
int hash(int x);
```

che calcola un hash di 4 bit a partire da un intero a 32 bit (parametro x) come somma dei nibble dispari (il nibble 0 sarà quello più a sinistra, il meno significativo) calcolata modulo 16, come illustrato nello schema sottostante. Il risultato è di soli 4 bit ma è rappresentato comunque come un valore a 32 bit, con i 28 bit più significativi a 0. Dunque, per esempio `hash(0x10372841)` dovrà essere calcolato come la somma di 4 (nibble 1), 2 (nibble 3), 3 (nibble 5), e 1 (nibble 7) modulo 16 e dare quindi come risultato 10 rappresentato in un registro da 32 bit come `0x0000000a`.



Esercizio 2

Data la tabella di verità in figura:

- fornire la forma canonica in somma di prodotti che la rappresenta;
- trovare un'espressione minima della funzione Booleana utilizzando le mappe di Karnaugh;
- mostrare i passi che permettono di derivare l'espressione trovata utilizzando le mappe di Karnaugh a partire dalla forma canonica in somma di prodotti utilizzando le proprietà dell'Algebra Booleana.

a	b	c	z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Parte 2

Esercizio 3

3-a) Descrivere in massimo 6 righe l'algoritmo di scheduling Round Robin.

3-b) Un sistema schedula i thread implementati a livello kernel con la tecnica MFQ (Multilevel Feedback Queue). La politica di scheduling prevede il prerilascio, che avviene immediatamente dopo l'evento che lo provoca, senza attendere l'esaurimento del quanto di tempo corrente. Quando un thread va in esecuzione gli viene assegnato un intero quanto di tempo pari a 10 ms, indipendentemente dal tempo consumato nel precedente turno di esecuzione. La priorità di un thread viene incrementata di 1, fino ad arrivare al valore massimo 3, ogni volta che il thread si sospende (o subisce il prerilascio) senza aver consumato per intero il quanto di tempo. La priorità del thread viene invece decrementata di 1, fino al valore minimo di 1, ogni volta che il thread consuma per intero il suo quanto di tempo.

Al tempo *t* sono presenti nel sistema i seguenti thread:

- Thread A, con priorità 2, che al tempo *t* passa in stato *esecuzione*;
- Thread B, con priorità 2, che al tempo *t* è in stato di *attesa* sul semaforo *Sem1*;
- Thread C, con priorità 2, che al tempo *t* è in stato di *pronto*;
- Thread D, con priorità 2, che al tempo *t* è in stato di *pronto* e detiene la spinlock SL;
- Thread E, con priorità 1, che al tempo *t* è in stato di *pronto.*

La coda con priorità 2 dei thread pronti al tempo t è quindi C ->D, dove C è il primo thread della coda.

Si chiede quale è il thread in esecuzione e la composizione delle 3 code al termine della seguente sequenza di eventi:

- al tempo *t+2* il thread in esecuzione V(Sem1);
- al tempo *t+4* il thread in esecuzione esegue P(Sem1);
- al tempo *t+6* il thread in esecuzione esegue spinlockAquire(SL);
- al tempo *t+16* il thread in esecuzione esegue spinlockRelease(SL);
- al tempo *t+17* il thread in esecuzione esegue V(Sem1);
- al tempo *t+18* il thread in esecuzione esegue spinlockAcquire(SL).

Esercizio 4

4-a) Descrivere in massimo 6 righe la paginazione dinamica multilivello.

4-b) Si consideri un sistema che gestisce la memoria con paginazione dinamica a due livelli con le seguenti caratteristiche:

- indirizzi logici di 32 bit e ampiezza dello spazio logico di ogni processo pari a 4 GB;
- pagine logiche e blocchi fisici di 8 KB;
- tabella di primo livello comprende 512 elementi; le tabelle di secondo livello hanno 1024 elementi ciascuna;
- ogni entry della tabella di primo e di secondo livello ha una dimensione di 24 bit dei quali 3 sono utilizzati per i flags (pagina caricata, modificata, riferita) mentre i restanti 21 bit codificano l'indirizzo di un blocco fisico.

Nel sistema è presente il processo P, che al tempo *t* ha il segmento codice dati e stack di lunghezza rispettivamente uguale a 40MB e 800 MB e 8 MB. Codice e dati sono collocati consecutivamente a partire dall'indirizzo logico 0 mentre lo stack parte dall'indirizzo logico 2^{32}-1 e si espande verso il basso.

Si chiede:

1. Il formato dell'indirizzo logico indicando la lunghezza delle componenti che indicizzano la tabella di primo livello, quelle di secondo livello ed il campo offset;
2. Lo spazio occupato in memoria dalla tabella di primo livello e da ogni tabella di secondo livello;
3. La massima dimensione della memoria fisica che il sistema può gestire espressa in numero di blocchi fisici ed in byte (espressi come potenze di 2);
4. Il numero di tabelle di secondo livello necessarie per la traduzione degli indirizzi del processo P al tempo *t*;
5. Nell'ipotesi che tutte le tabelle delle pagine necessarie per la traduzione degli indirizzi del processo P siano caricate in memoria, lo spazio (in byte) occupato complessivamente dalla tabella di primo livello e da quelle di secondo livello;
6. Indicare quali entry delle tabelle di primo e secondo livello sono usate per tradurre l'indirizzo logico 0x80EFFF del processo P al tempo *t*.

Soluzione Esercizio 1

```
.text
.global hash
.type hash, %function

hash: mov r1, #4        @ numero di iterazioni
mov r2, #0        @ accumulatore
loop: lsr r0, r0, #4    @ butta via nibble pari (0)
and r3, r0, #15   @ maschera i nibble alti
add r2, r2, r3    @ somma
lsr r0, r0, #4    @ via il nibble processato
subs r1, r1, #1   @ aggiorna variabile for
bne loop          @ prossima iterazione
end:  and r0, r2, #15   @ restituisci somma mod 16
mov pc, lr        @ ritorno al chiamante
```

Soluzione Esercizio 2

La forma in somma di prodotti è data da

z = ~~a~~bc+~~a~~b~~c~~+~~a~~bc+~~a~~b~~c~~

La mappa di karnaugh corrispondente alla tabella di verità è quella che segue:

ab\c	0	1
00	0	1
01	1	1
11	0	0
10	0	1

Sono evidenziati I due implicanti:

- ~~A~~b
- ~~B~~c

Quindi z = ~~a~~b + ~~b~~c

Questa deriva dalla somma di prodotti:
z = ~~a~~bc+~~a~~b~~c~~+~~a~~bc+~~a~~b~~c~~ = ~~a~~bc+~~a~~b(~~c~~+c)+ ~~a~~bc = (~~a~~+a)~~b~~c+ ~~a~~b = ~~b~~c+ ~~a~~b

Soluzione Esercizio 3:

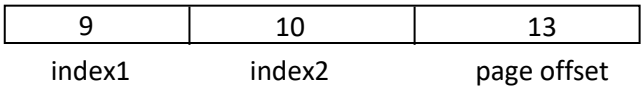
3-b) Completare la tabella – A(2) vuol dire che il thread A ha priorità attuale 2

t+	Evento	Subito dopo l'evento					
		Esecuzione	**Coda Pronti prio 1**	**Coda Pronti prio 2**	**Coda Pronti prio 3**	**Sem1 val, coda**	**SpinLock Valore**
0	A in esecuzione	A(2)	E	C->D	-	0, B(2)	BUSY
2	V(Sem1)	A(2)	E	C->D->B	-	0,-	BUSY
4	P(Sem1)	C(2)	E	D->B	-	0, A(3)	BUSY
6	spinlockAcquire	C(2)	E	D->B	-	0, A(3)	BUSY
14	Scade QdT	D(2)	E->C	B	-	0, A(3)	BUSY
16	spinlockRelease	D(2)	E->C	B	-	0, A(3)	FREE
17	V(Sem1)	A(3)	E->C	B	D	0, -	FREE
18	spinlockAcquire	A(3)	E->C	B	D	0, -	BUSY

Soluzione Esercizio 4:

4-b)

1. I 13 bit meno significativi corrispondono al campo offset, seguono 10 bit che indicizzano la tabella di secondo livello ed infine i 9 bit più significativi indicizzano la tabella di primo livello.



2. Lo spazio occupato in memoria dalla tabella di primo livello è $3 * 2^9$ byte = 1.5 KB;
Lo spazio occupato in memoria da ogni tabella di secondo livello è $3 * 2^{10}$ byte = 3 KB

3. La massima dimensione della memoria fisica gestita dal sistema è di 2^{21} blocchi ossia $2^{21} * 2^{13} = 2^{34}$ byte (pari a 16 GB).

4. Il numero di tabelle di secondo livello necessarie per la traduzione degli indirizzi del processo P è:

a. Per il codice e i dati, che occupano 840 MB dello spazio logico, servono:
$840 * 2^{20}/2^{13} = 105 * 2^{10}$ pagine. Ogni tabella è in grado di ospitare 2^{10} indirizzi di pagina fisica quindi serviranno 105 tabelle di secondo livello.

b. Per lo stack, che occupa 8 MB, serviranno $8 * 2^{20}/2^{13} = 2^{10}$ pagine, quindi esattamente una tabella di secondo livello

5. Lo spazio in byte occupato complessivamente dalle tabelle di primo e secondo livello è:

$3 * (1*2^9 + 106 * 2^{10})$ byte = 319.5 KB

6. La rappresentazione binaria dell'indirizzo logico 0x80EFFF è
0000 0000 1000 0000 1110 1111 1111 1111 quindi:

l'indice della tabella di primo livello (index1) è 000000001 (entry 2)

l'indice della tabella di secondo livello (index2) è 0000000111 (entry 8)

l'offset di pagina è 0111111111111

Per tradurre l'indirizzo logico 0x80EFFF si usa l'entry 2 della tabella di primo livello e l'entry 8 della tabella di secondo livello.