

PROVA SCRITTA DEL MODULO DI
CALCOLATORI ELETTRONICI
CORSI DI LAUREA IN INGEGNERIA ELETTRICA, ELETTRONICA ED INFORMATICA e
INGEGNERIA BIOMEDICA
24 luglio 2019

NOME:

COGNOME:

MATRICOLA:

CFU:

ESERCIZIO 1 (7 punti)

- 1) (4 punti) Progettare una rete logica che, date due stringhe binarie A e B di n bit, presenti un'uscita Z pari ad 1 se $A = B$. Disegnare il circuito.
- 2) (3 punti) Scrivere la tabella di transizione dello stato del FF-JK ed evidenziare la differenza con l'analoga tabella del FF-SR.

ESERCIZIO 2 (6 punti)

Si consideri una memoria primaria costituita da 256 blocchi e una memoria cache di 128 B, con blocchi di 8 B. E' possibile indirizzare il singolo byte.

1. (1 punto) Calcolare la dimensione in KB della memoria primaria.
2. (2 punti) Spiegare, precisando il significato e la funzione dei diversi campi, come vengono interpretati gli indirizzi logici per recuperare l'informazione contenuta nella cache nel caso di indirizzamento diretto.
3. (3 punti) Supponendo la cache inizialmente vuota, si considerino le chiamate ai seguenti indirizzi (espressi in decimale): da 0 a 55, da 80 a 95 in questo ordine, per dieci volte consecutive. Indicare lo stato finale della cache e calcolare l'hit ratio.

ESERCIZIO 3 (14 punti)

Implementare l'algoritmo di ordinamento di un vettore noto come "selection-sort" descritto dalle tre funzioni C scritte sotto. Implementare **tutte e tre** le funzioni.

<pre>void scambia(int *v, int i, int j) { int temp; temp=v[i]; v[i]=v[j]; v[j]=temp; } void selectionSort(int* v, int N) { int i, imin; for (i=0; i<N-1; i++) { imin=trovaMinimo(v, i, N); scambia(v, i, imin); } }</pre>	<pre>int trovaMinimo(int *v, int start, int stop) { int i; int vmin=v[start]; int imin=start; for (i=start+1; i<stop; i++) if (v[i]<vmin) { vmin=v[i]; imin=i; } return imin; }</pre>
--	---

Il passaggio dei parametri per le tre funzioni avviene coi seguenti registri:

- (3 punti) `scambia`: $v \rightarrow \$4$, $i \rightarrow \$5$, $j \rightarrow \$6$
- (6 punti) `trovaMinimo`: $v \rightarrow \$4$, $start \rightarrow \$5$, $stop \rightarrow \$6$, $imin \rightarrow \$7$
- (5 punti) `selectionSort`: $v \rightarrow \$4$, $N \rightarrow \$5$

ESERCIZIO 4 (6 punti)

Disponendo di sette linee di controllo in tutto, si illustri e disegni un possibile schema di arbitraggio distribuito precisando il massimo numero di periferiche gestibili.

ESERCIZIO 1

- 1) La rete richiesta presenta due livelli di logica: il primo è costituito da n porte XOR che ricevono in ingresso le coppie A_i e B_i delle due stringhe ($i=0, \dots, n-1$). Ciascuna di queste porte permette di sapere se i bit differiscono. Il secondo livello è una porta che deve fornire 1 se tutte le n porte del livello precedente hanno fornito 0. Tale porta è chiamata NAND in quanto nega l'uscita di una porta AND. Si lascia allo studente il disegno del circuito.
- 2) Si vedano le dispense del corso.

ESERCIZIO 2

1. Essendo la memoria primaria costituita da $256 = 2^8$ blocchi, ciascuno di $8 = 2^3$ B, si ottiene una memoria di 2^{11} B, ovvero 2 KB.
2. I tre bit meno significativi dell'indirizzo di primaria compongono l'offset (i blocchi sono di otto parole). Il numero di linee di cache è invece 16 ($128 \text{ B} / 8 \text{ B}$), per cui il cache index è formato da 4 bit. Si ottiene quindi la sequenza TAG-Cache Index-Offset pari a: 4-4-3.
3. Dalla lettura del testo si nota che le prime 56 chiamate sono parole contigue a partire dal blocco 0 di primaria, per un totale di 7 blocchi. Per definizione di metodo diretto, ciascuno di questi blocchi presenta in cache index diverso (da 0 a 6) e vengono così allocati nelle corrispondenti linee di cache.

Il secondo gruppo di chiamate, da 80 a 95, è costituito da 16 parole contigue. Dividendo l'indirizzo della prima per il numero di blocchi si ottiene dal resto pari a 0 che essa è la prima parola di un blocco, e l'ulteriore divisione del block frame (10) per il numero di linee (16) ci fa calcolare in quale linea va allocato il blocco, permettendoci anche di sapere l'indirizzo di cache del blocco successivo. Il blocco 80-87 viene allocato nella linea 10 e il blocco 88-95 nella linea 11, senza sovrapporsi ai blocchi del primo gruppo che occupano le linee da 0 a 6.

Le successive chiamate ripetono la sequenza dall'inizio e non determina variazioni di allocazione perché tutte le parole sono già in cache. Schematizzando la cache come segue, lo stato finale è dunque:

Cache Index	Parole	Cache Index	Parole
0	0-7	8	
1	8-15	9	
2	16-23	10	80-87
3	24-31	11	88-95
4	32-39	12	
5	40-47	13	
6	58-55	14	
7		15	

Abbiamo 7 hit per ciascuno dei 9 blocchi chiamati al primo ciclo, mentre dal secondo ciclo abbiamo tutti hit. Si ottiene dunque:

$$H_c = \frac{7 \cdot 9 + 9 \cdot 8 \cdot 9}{10 \cdot 8 \cdot 9} = \frac{7 + 72}{80} = 0.9875$$

ESERCIZIO 4

Dal momento che con l'arbitraggio distribuito c'è bisogno di una linea che attesti la richiesta del possesso del bus per ogni periferica, un segnale di grant attivo propagato lungo un festone e una linea di bus busy comune che impedisca l'accesso al bus alle altre periferiche, con sette linee possiamo servire al massimo cinque periferiche connesse in "daisy chain". Il grant è connesso a massa se attivo basso o ad alimentazione se attivo alto.

ESERCIZIO 3

<p>Funzione swap v → \$4, i → \$5, j → \$6</p> <pre>swap: addi \$29, \$29, -16 sw \$8, 0(\$29) sw \$9, 4(\$29) sw \$10, 8(\$29) sw \$11, 12(\$29) muli \$8, \$5, 4 add \$8, \$8, \$4 lw \$10, 0(\$8) muli \$9, \$6, 4 add \$9, \$9, \$4 lw \$11, 0(\$9) sw \$10, 0(\$9) sw \$11, 0(\$8) lw \$8, 0(\$29) lw \$9, 4(\$29) lw \$10, 8(\$29) lw \$11, 12(\$29) addi \$29, \$29, 16 jr \$31</pre> <p>Funzione selectionSort v → \$4, N → \$5</p> <pre>selectionSort: addi \$29, \$29, -16 sw \$8, 0(\$29) sw \$11, 4(\$29) sw \$12, 8(\$29) sw \$31, 12(\$29) move \$11, \$5 subi \$12, \$5, 1 move \$8, \$0 forSort: beq \$8, \$12, exitSort move \$5, \$8 move \$6, \$11 jal trovaMinimo move \$6, \$7 jal scambia addi \$8, \$8, 1 j forSort exitSort: move \$5, \$11 lw \$8, 0(\$29) lw \$11, 4(\$29) lw \$12, 8(\$29) lw \$31, 12(\$29) addi \$29, \$29, 16 jr \$31</pre>	<p>Funzione trovaMinimo v → \$4, start → \$5, stop → \$6, imin → \$7</p> <pre>trovaMinimo: addi \$29, \$29, -20 sw \$8, 0(\$29) sw \$9, 4(\$29) sw \$10, 8(\$29) sw \$11, 12(\$29) sw \$12, 16(\$29) move \$7, \$5 muli \$8, \$5, 4 add \$8, \$8, \$4 lw \$10, 0(\$8) addi \$8, \$5, 1 forMinimo: beq \$8, \$6, exitMinimo muli \$9, \$8, 4 add \$9, \$9, \$4 lw \$11, 0(\$9) slt \$12, \$11, \$10 beq \$12, \$0, updMinimo move \$7, \$8 move \$10, \$11 updMinimo: addi \$8, \$8, 1 j forMinimo exitMinimo: lw \$8, 0(\$29) lw \$9, 4(\$29) lw \$10, 8(\$29) lw \$11, 12(\$29) lw \$12, 16(\$29) addi \$29, \$29, 20 jr \$31</pre>
--	--