

PRIMA PROVA INTERMEDIA DEL MODULO DI
CALCOLATORI ELETTRONICI
CORSO DI LAUREA IN INGEGNERIA ELETTRICA, ELETTRONICA ED INFORMATICA
CORSO DI LAUREA IN INGEGNERIA BIOMEDICA
ISCRITTI A.A. 2016/17
19 Aprile 2017

NOME:

COGNOME:

MATRICOLA:

CFU:

ESERCIZIO 1 (7 punti)

Si progetti una rete logica in grado di riconoscere in una sequenza di bit la stringa 1000 ponendo a 1 l'uscita solo quando si abbia il riconoscimento di tale stringa. Si sintetizzi la rete minima che realizza tale funzione utilizzando FF-D. Disegnare il circuito logico che rappresenta la rete minima sintetizzata.

ESERCIZIO 2 (10 punti)

Si consideri la gerarchia di memoria costituita da una memoria primaria di capacità 16 kB e una memoria cache di capacità 256 B, con blocchi di 2 B. E' possibile indirizzare il singolo byte.

1. (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 ed indirizzamento associativo su insiemi a due vie.
2. (4 punti) Supponendo la cache inizialmente vuota, si considerino le chiamate ai seguenti indirizzi (espressi in decimale): da 0 a 127, da 1024 a 1279 in questo ordine, ripetute per dieci volte consecutive. Si indichi il contenuto della cache, ovvero quali byte occupano le linee di cache, dopo l'ultima chiamata, utilizzando i due metodi di indirizzamento al punto 1.
3. (2 punti) Si calcoli l'hit ratio della cache sulla base delle chiamate al punto precedente per entrambi i metodi.
4. (2 punti) Se il tempo di accesso in cache è pari a 4 nsec mentre quello in primaria è pari a 40 nsec, calcolare il tempo medio di accesso alla gerarchia. Esprimere tutti i tempi **in nanosecondi**.

ESERCIZIO 3 (3 punti)

Si consideri un disco magnetico caratterizzato dai seguenti parametri: velocità 6000 rpm, 16 tracce da 64 B ciascuna, tempo di posizionamento da una traccia a quella adiacente 1 ms. Si calcoli il tempo medio di lettura di un file da 256 B supponendo che il file sia registrato su settori consecutivi di tracce adiacenti, con la testina posizionata all'istante iniziale sul primo settore utile.

ESERCIZIO 4 (10 punti)

I trasferimenti di parole a/dalla memoria di un calcolatore sono codificati utilizzando il codice di Hamming.

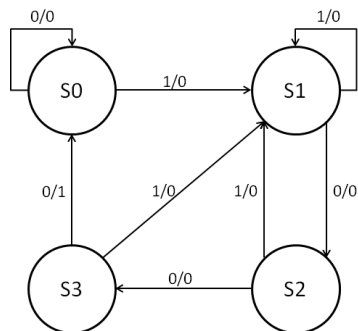
1. (2 punti) Codificare la stringa 10101101 (il bit meno significativo è a sinistra), secondo il codice di Hamming, motivando ogni passaggio intermedio. Scrivere la stringa codificata.
2. (3 punti) Se in ricezione si avesse una commutazione dell'ultimo bit della stringa data, spiegare come il sistema riuscirebbe a localizzare ed a correggere l'errore.
3. (5 punti) Si consideri ora il caso del codice BCD (Binary Coded Decimal) visto a lezione per codificare le cifre decimali da 0 a 9. Come dovrei modificare il codice BCD per crearne una versione in grado di rivelare, senza correggere, un errore singolo? Spiegare in modo chiaro e dettagliato la procedura proposta scrivendo per ogni cifra decimale da 0 a 9 la codifica BCD modificata in modo tale da rivelare, senza correggere, un errore singolo.

ESERCIZIO 5 (3 punti)

Si consideri una CPU con set d'istruzioni a zero indirizzi, dotata delle seguenti istruzioni: PUSH X ($M[X] \rightarrow \text{push}$), POP X ($\text{pop} \rightarrow M[X]$), ADD ($\text{pop} + \text{pop} \rightarrow \text{push}$), DIV ($\text{pop1} / \text{pop2} \rightarrow \text{push}$). Scrivere la sequenza d'istruzioni necessaria per implementare l'espressione algebrica: $Z = A/B + C/D + E$.

ESERCIZIO 1

Per prima cosa, occorre scrivere il grafo degli stati che corrisponde al seguente:



Utilizzando FF-D:

A	B	X	A'	DA	B'	JB	Z
0	0	0	0	0	0	0	0
0	0	1	0	0	1	1	0
0	1	0	1	1	0	0	0
0	1	1	0	0	1	1	0
1	0	0	1	1	1	1	0
1	0	1	0	0	1	1	0
1	1	0	0	0	0	0	1
1	1	1	0	0	1	1	0

Semplificando le espressioni delle rete logiche per la transizione dello stato:

AB \ X	00	01	11	10
0		1		1
1				

$$D_A = \bar{A}B\bar{X} + A\bar{B}\bar{X}$$

AB \ X	00	01	11	10
0				1
1	1	1	1	1

$$D_B = A\bar{B} + X$$

L'espressione dell'uscita è invece data da:

$$Z = AB\bar{X}$$

Si lascia allo studente la stesura del circuito.

ESERCIZIO 2

1. Essendo la memoria primaria costituita da $16 \text{ kB} = 2^{14} \text{ B}$, l'indirizzamento è a 14 bit. Di questi, quello meno significativo compone l'offset (i blocchi sono di due parole). Il numero di linee di cache è invece 128 ($256 \text{ B} / 2 \text{ B}$), per cui il cache index è formato da 7 bit. I restanti 6 costituiscono il TAG. Nel metodo set associativo a due vie un bit di index è sufficiente per indirizzare i set. Si ottiene dunque un TAG di 7 bit.
2. Per verificare come i blocchi sono assegnati alle linee di cache ed indicare lo stato finale della cache, bisogna innanzi tutto vedere quali sono i valori di index nei due casi di indirizzamento. Cominciamo dal metodo diretto. La prima parola chiamata ha indirizzo 0. In questo caso offset, index e TAG hanno tutti lo stesso valore (0). Si ottiene dunque che la parola 0 è la prima parola di un blocco di due parole (dalla 0 alla 1) che verranno tutte memorizzate nella linea 0 di cache. Il blocco successivo a questo contiene le parole dalla 2 alla 3 e così via fino a completare il primo nucleo di 128 parole consecutive, che verranno nelle prime 64 linee di cache.

Il secondo nucleo inizia dalla parola $1024 = 2^{10}$. Ciò significa che è una stringa di undici bit dei quali solo quello in posizione più significativa è 1. Quindi anche in questo caso il cache index è 0 e la parola cercata è la prima di un blocco di due parole (1024, 1025) che andrà a sovrascrivere la linea di cache dove erano state memorizzate le parole 0-1. Le parole dalla 1152 alla 1279 resteranno stabilmente nelle linee 64-127. Alla fine lo stato della cache sarà:

DIRETTO	Offset di parola	
Index di linea	0	1
0	1024	1025
1	1026	1027
...
127	1278	1279

Per quanto riguarda il metodo set associativo a due vie, il procedimento è analogo al precedente ma con un dettaglio importante. Il bit di TAG in più non incide sui valori di set index (cache index) calcolati in precedenza. Quindi sia il primo blocco di parole 0-127 che la prima metà del secondo 1024-1151 vengono mappate nei primi 64 set della cache, ma questa volta poiché abbiamo due vie le parole del primo blocco occuperanno le prime linee di ciascun set, mentre le parole della seconda metà del secondo blocco occuperanno le seconde linee. Cosa succede alla seconda metà del secondo blocco? Indipendentemente dall'uso del metodo FIFO o LRU, per citare i due metodi visti principalmente a lezione, la seconda metà del secondo blocco andrà ad impegnare le linee impegnate dalle parole 0-127, alla prima iterazione. Alla seconda iterazione, le parole del primo blocco andranno a sostituire le linee impegnate dalle parole 1024-1151 e così via. In altre parole i tre blocchi 0-127, 1024-1151, e 1152-1279 a rotazione impegneranno le prime e seconde linee in quest'ordine:

- Prima iterazione (set 0-63): parole 0-127 → prime linee; 1024-1151 → seconde linee; 1152-1279: prime linee.
- Seconda iterazione: parole 0-127 → seconde linee; 1024-1151 → prime linee; 1152-1279: seconde linee.
- ...
- Decima iterazione: parole 0-127 → seconde linee; 1024-1151 → prime linee; 1152-1279: seconde linee.

Lo stato finale è dunque:

SET ASSOCIATIVO	Offset di parola	
Index di set	0	1
0	1024	1025
	1152	1153
1	1026	1027
	1153	1154
...

63	1150	1151
	1278	1279

3. Il calcolo del hit ratio è conseguente ai due diversi meccanismi di indirizzamento, ma sostanzialmente.

Nel caso dell'indirizzamento diretto, abbiamo 1 hit per ciascuno dei blocchi chiamati ($64 \cdot 3 = 192$ in tutto) per il primo ciclo. Dal secondo in poi i blocchi con le parole 0-127 e 1024-1151 si sovrascrivono l'un l'altro ma le parole 1152-1279 rimangono stabilmente in cache. Quindi si ha:

$$H_c = \frac{192 + 64 \cdot 9 + 64 \cdot 9 + 128 \cdot 9}{384 \cdot 10} = \frac{2496}{3840} = 0.65$$

Similmente, nel caso set-associativo, abbiamo 1 hit per ciascuno dei blocchi chiamati al primo ciclo. L'effetto di sovrascrittura del terzo nucleo di blocchi determina tuttavia una mancata massimizzazione degli hit nelle iterazioni successive. Si ottiene dunque:

$$H_c = \frac{192 \cdot 10}{384 \cdot 10} = \frac{1}{2} = 0.5$$

4. La formula del tempo medio di accesso alla gerarchia di memoria data è:

$$\bar{T} = T_c + (1 - H_c) \cdot T_p$$

Sostituendo i valori di H_c trovati nel precedente esercizio si ha:

$$\begin{aligned}\bar{T}_{Diretto} &= 4 + 0.35 \cdot 40 = 18nsec \\ \bar{T}_{SetAssociativo} &= 4 + \frac{1}{2} \cdot 40 = 24nsec\end{aligned}$$

ESERCIZIO 3

Il blocco da leggere occupa $256 \text{ B} = 2^8 \text{ B}$ che richiedono 4 tracce complete. Leggere i settori di un blocco simile richiede dunque quattro tempi di rotazione.

Le altre due componenti della lettura sono latenza e posizionamento. Delle due è presente soltanto quella di posizionamento con tre spostamenti complessivi da una traccia a quella adiacente in quanto siamo in modalità totalmente deframmentata.

In sintesi, il tempo di lettura sarà dato da:

$$T = 4T_R + 3$$

Essendo $T_R = \frac{60}{6000} = 10msec$, si ha $T = 40 + 3 = 43msec$

ESERCIZIO 4

- 1) Deve essere rispettata la condizione:

$$2^K \geq N + K + 1 \quad (1),$$

dove K è il numero di bit di controllo inseriti. Essendo $N = 8$, si evince dalla (1) che $K = 4$.

Nella codifica di Hamming, la sequenza in uscita deve presentare la seguente struttura:

	c_0	c_1	b_0	c_2	b_1	b_2	b_3	c_3	b_4	b_5	b_6	b_7
			1		0	1	0		1	1	0	1
e_0	1		1		1		1		1		1	
e_1		1	1			1	1			1	1	
e_2				1	1	1	1					1
e_3								1	1	1	1	1

Dove $c_0...c_3$ sono i quattro bit costituenti il vettore di controllo, e $b_0...b_7$ gli otto bit trasmessi.

I quattro bit devono essere tali che il corrispondente vettore di errore $e_3e_2e_1e_0$ indichi il bit alterato in caso di errore. L'intersezione fra gli uni presenti nel vettore di errore ci dice univocamente quale bit sia stato alterato. Scrivendo per ciascuno dei bit della sequenza intera le configurazioni di errore, otteniamo infatti i valori di parità che ciascun bit di errore deve assumere in corrispondenza di ciascuno dei bit della stringa completa. Calcolando i bit di controllo rispetto alla sequenza $b_0...b_7$ ricevuta si ottiene:

$c_0 = 0$

$c_1 = 1$

$c_2 = 0$

$c_3 = 1$

La stringa codificata è dunque **011001011101**.

- 2) Una commutazione del bit b_7 altera i bit che lo controllano ovvero c_3 e c_2 come si evince dalla tabella soprastante. Ciò porta alla configurazione di errore 1100, che permette di individuare e correggere, invertendolo, il bit alterato.
- 3) Il codice BCD visto a lezione (Capitolo 2) codifica le cifre decimali con 4 bit (con le parole di 4 bit che vanno da 0000 a 1001). Il modo più semplice di modificare tale codifica in modo tale da rivelare, senza correggere, un errore singolo è quello di aggiungere ad ogni parola di codice un bit ridondante detto di parità che viene posto a 0 o a 1 in modo tale che la somma degli uni della parola di codice BCD così modificata sia pari. Esempi: Codice BCD originale 0000, Codice BCD modificato 00000, Codice BCD originale 0001, Codice BCD modificato 00011. Analoga codifica per tutte le altre parole del codice BCD. Un eventuale errore singolo nella parola di codice BCD così modificata farà "saltare" il vincolo di parità, segnalando così la presenza di un errore singolo nella parola, senza ovviamente consentire di individuare e correggere tale errore. Per gli allievi che desiderassero approfondire il tema vale la pena notare che le parole del codice BCD così modificato presentano "distanza di Hamming" maggiore o uguale a due (https://it.wikipedia.org/wiki/Distanza_di_Hamming). Se avessimo voluto rilevare e anche correggere errori singoli sarebbe servita una codifica con "distanza di Hamming" maggiore o uguale a tre.

ESERCIZIO 5

A partire dalla semantica fornita nel testo, una possibile sequenza è:

Istruzione	Semantica
PUSH B	$M[B] \rightarrow \text{push}$
PUSH A	$M[A] \rightarrow \text{push}$
DIV	$\text{pop1}(A) + \text{pop2}(B) \rightarrow \text{push}(A/B)$
PUSH D	$M[D] \rightarrow \text{push}$
PUSH C	$M[C] \rightarrow \text{push}$
DIV	$\text{pop1}(C) + \text{pop2}(D) \rightarrow \text{push}(C/D)$
ADD	$\text{pop}(C/D) + \text{pop}(A/B) \rightarrow \text{push}(A/B + C/D)$
PUSH E	$M[E] \rightarrow \text{push}$
ADD	$\text{pop}(E) + \text{pop}(A/B + C/D) \rightarrow \text{push}(A/B + C/D + E)$
POP Z	$\text{pop}(A/B + C/D + E) \rightarrow M[Z]$