

PROVA SCRITTA DEL MODULO DI
CALCOLATORI ELETTRONICI
CORSO DI LAUREA IN INGEGNERIA ELETTRICA, ELETTRONICA, ED INFORMATICA CORSO DI
LAUREA IN INGEGNERIA BIOMEDICA
25 giugno 2019

NOME:	COGNOME:	MATRICOLA:	CFU:
--------------	-----------------	-------------------	-------------

ESERCIZIO 1 (9 punti)

Progettare una rete sequenziale che riconosca le due sequenze binarie 10101 e 10001, attraverso un'uscita posta ad 1 quando una delle due sequenze viene individuata. Disegnare il grafo degli stati, la tabella delle transizioni e si considerino FF-D nel progetto della rete di transizione dello stato. Semplificare le reti di transizione dello stato con le mappe di Karnaugh. Scrivere anche l'espressione della rete di transizione dell'uscita.

ESERCIZIO 2 (8 punti)

Si consideri una memoria primaria costituita da 512 B e una memoria cache costituita da 16 B, con blocchi di 8 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.
2. (4 punti) Supponendo la cache inizialmente vuota, si considerino le chiamate ai seguenti indirizzi (espressi in decimale): da 8 a 15, da 48 a 55 in questo ordine, per un numero di volte consecutive. Si indichi il contenuto finale della cache, ovvero quali byte occupano le linee di cache, dopo l'ultima chiamata.
3. (2 punti) Quanto vale $\frac{1}{N}$ per avere un hit ratio di cache superiore a 0.95?

ESERCIZIO 3 (9 punti)

Implementare una funzione Assembly MIPS che, dati l'indirizzo iniziale di un vettore di valori interi v in $\$4$ e la sua dimensione N in $\$5$, restituisca in $\$7$ il numero di valori di v che risultano multipli di un intero x passato in $\$6$. Per implementare la funzione si faccia uso di un'ulteriore funzione `div` che riceve due valori interi a e b rispettivamente in $\$4$ e $\$5$ e restituisce in $\$6$ il resto della divisione a/b .

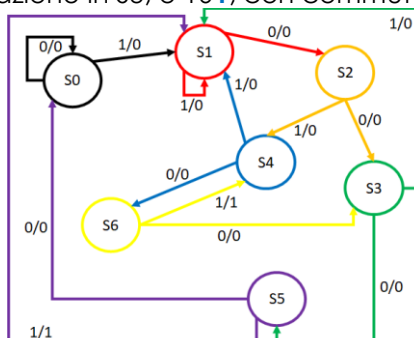
ESERCIZIO 4 (7 punti)

Le parole trasferite a/dalla memoria di un calcolatore sono codificate utilizzando il codice di Hamming. Si consideri la stringa di 13 bit 1010011011101 (il bit meno significativo è a sinistra), risultato della codifica di una parola di N bit secondo il codice di Hamming.

1. (1 punto) calcolare N , supponendo di aver fatto uso del numero minimo di bit di controllo necessari.
2. (2 punti) scrivere la parola di N bit a partire dalla stringa data;
3. (4 punti) indicare eventuali errori nella stringa codificata, specificando quale dei bit è stato alterato.

ESERCIZIO 1

Per chiarire meglio la struttura del grafo degli stati, evidenziamo con colori diversi ciascun nodo ed i relativi archi in uscita. Le uscite dallo stato S2 evidenziano i percorsi logici seguiti alla ricezione del terzo bit utile: 100, con commutazione in S3, o 101, con commutazione in S4.



Identificato ciascun nodo con la tripla di bit ABC configurati in modo crescente $S0 \leftarrow 000$, $S1 \leftarrow 001$, ..., $S6 \leftarrow 110$, la tabella di transizione è la seguente:

A	B	C	X	A'	DA	B'	DB	C'	DC	Z
0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1	1	0
0	0	1	0	0	0	1	1	0	0	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	0	0	1	1	1	1	0
0	1	0	1	1	1	0	0	0	0	0
0	1	1	0	1	1	0	0	1	1	0
0	1	1	1	0	0	0	0	1	1	0
1	0	0	0	1	1	1	1	0	0	0
1	0	0	1	0	0	0	0	1	1	0
1	0	1	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	1	1	1
1	1	0	0	0	0	1	1	1	1	0
1	1	0	1	1	1	0	0	0	0	1
1	1	1	0	D	D	D	D	D	D	(D)
1	1	1	1	D	D	D	D	D	D	(D)

Per semplificare le reti di transizione dello stato usiamo le mappe di Karnaugh:

AB CX	00	01	11	00
00				1
01		1	1	
11			D	
10		1	D	

AB CX	00	01	11	00
00		1	1	1
01				
11			D	
10	1		D	

AB CX	00	01	11	00
00		1	1	
01	1			1
11	1	1	D	1
10		1	D	

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

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

$$DC = B\bar{X} + BC + \bar{B}X$$

$$Z = A\bar{B}CX + A\bar{B}\bar{C}X$$

ESERCIZIO 2

- Essendo la memoria primaria costituita da $512 \text{ B} = 2^9 \text{ B}$, l'indirizzamento è a 9 bit. Di questi, i tre meno significativi compongono l'offset (i blocchi sono di otto parole). Il numero di linee di cache è invece 2 ($16 \text{ B}/8 \text{ B}$), per cui il cache index è formato da 1 bit. I restanti 5 costituiscono il TAG.
- Per verificare come i blocchi sono assegnati alle linee di cache ed indicare lo stato finale della cache, è sufficiente sapere se il block frame a cui appartengono le varie parole ha valore pari o dispari. Come si può notare i due gruppi da 8 a 15 e da 48 a 55 sono formati entrambi da otto parole. Per sapere se queste otto parole appartengono allo stesso blocco, basta verificare che il resto della divisione indirizzo/dimBlocco dia zero per la prima parola dei due gruppi. Si verifica in entrambi i casi che $8/8 \rightarrow$ resto 0, blocco 0, blocco 1, che è dispari quindi indirizzato nella linea 1 di cache, $48/8 \rightarrow$ resto 0, blocco 6, che è pari e quindi sarà indirizzato nella linea 0 di cache. Avendo entrambe le parole indirizzo con campo offset nullo, esse rappresentano le prime parole di un blocco di primaria che, a cache vuota verranno interamente copiate nella cache, saturandola, sulle due linee libere. La ripetizione delle chiamate è ininfluente ai fini dell'allocazione e quindi si avrà il seguente stato finale di cache:

Cache index	Offset di parola							
	0	1	2	3	4	5	6	7
0	48	49	50	51	52	53	54	55
1	8	9	10	11	12	13	14	15

- Se n è il numero di volte che si ripete la chiamata alla sequenza 8-15 e 48-55, è evidente che la prima volta si verichino 14 hit su 16 chiamate. A partire dalla seconda chiamata, il numero di hit è 16 perché le parole sono tutte in cache. Il numero complessivo di hit si può esprimere come $N_H = 14 + 16(n - 1)$. Il numero complessivo di chiamate è dato da $N = 16n$. Da cui:

$$H_c = \frac{N_H}{N} = \frac{14 + 16(n - 1)}{16n}$$

Per rispondere alla domanda, si risolve in funzione della disequazione $H_c > 0.95$, ottenendo $n > 2.5$. Poiché n dev'essere intero, il suo minimo valore che consente di ottemperare alla richiesta è 3.

ESERCIZIO 3

Utilizziamo i seguenti registri:

- \$8 come offset i del vettor v;
- \$9 come indirizzo di v[i].

Inoltre utilizziamo i registri \$4 per v[i], \$5 per x, mentre \$7 come da richiesta deve presentare il numero di multipli di x presenti in v.

Una copia dei registri \$4 e \$5 sarà dunque necessaria e per lo scopo destiniamo i registri \$11 e \$12.

La modalità per il salvataggio del contesto è "callee save".

```
conta:    addi $29, $29, -16      #salva il contesto
          sw $8, 0($29)
          sw $9, 4($29)
          sw $11, 8($29)
          sw $12, 12($29)
          move $11, $4           #copia i registri
          move $12, $5
          move $5, $6             #x sarà divisore di v[i]
          move $7, $0            #contatore=0
          move $8, $0            #i=0
for:      beq $8, $12, exit      #while i<N
          muli $9, $8, 4
          add $9, $9, $11
          lw $4, 0($9)          #carico v[i] in $4
          jal div                #$6 ← resto di v[i]/x
          bne $6, $0, upd        #se il resto non è 0, v[i] non è multiplo
          addi $7, $7, 1         #altrimenti aggiorna contatore
upd:      addi $8, $8, 1         #i=i+1
          j for
exit:     move $6, $5            #ripristina i valori iniziali nei registri
          move $5, $12
          move $4, $11
          lw $8, 0($29)         #ripristina il contesto
          lw $9, 4($29)
          lw $11, 8($29)
          lw $12, 12($29)
          addi $29, $29, 16
          jr $31                #ritorna al chiamante
```

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 $N + K = 13$, si evince dalla (1) che il numero minimo di bit di controllo richiesto è 4. Da cui $N = 8$.

2) Nella codifica di Hamming, la sequenza in ingresso presenta la seguente struttura:

c0	c1	b0	c2	b1	b2	b3	c3	b4	b5	b6	b7	b8
1	0	1	0	0	1	1	0	1	1	1	0	1

Dove $c_0 \dots c_3$ sono i 4 bit costituenti il vettore di controllo, e $b_0 \dots b_8$ i 9 bit trasmessi. La sequenza ricevuta è 101111101.

3) Per verificare la presenza di un errore, dobbiamo ricalcolare il vettore di controllo a partire dalla sequenza ricevuta. Si ha:

$$c'_0 = b_0 \oplus b_1 \oplus b_3 \oplus b_4 \oplus b_6 \oplus b_8 = 1$$

$$c'_1 = b_0 \oplus b_2 \oplus b_3 \oplus b_5 \oplus b_6 = 1$$

$$c'_2 = b_1 \oplus b_2 \oplus b_3 \oplus b_7 \oplus b_8 = 1$$

$$c'_3 = b_4 \oplus b_5 \oplus b_6 \oplus b_7 \oplus b_8 = 0$$

Il passo successivo è calcolare il vettore di errore dato dalla differenza dei vettori di controllo c e c' (ricordiamo che somma e differenza tra bit producono lo stesso risultato):

$$e_0 = c_0 \oplus c'_0 = 0$$

$$e_1 = c_1 \oplus c'_1 = 1$$

$$e_2 = c_2 \oplus c'_2 = 1$$

$$e_3 = c_3 \oplus c'_3 = 0$$

Poiché il vettore risultante 0110 non è nullo, vi è un errore nella stringa di 13 bit data e precisamente nella posizione indicata dal vettore di errore tradotto in notazione decimale. Il bit sbagliato è quindi il sesto bit (b_2), e la parola corretta è 100111101.