

**PROVA SCRITTA DEL MODULO DI**  
**CALCOLATORI ELETTRONICI**  
**CORSI DI LAUREA IN**  
**INGEGNERIA ELETTRICA, ELETTRONICA ED INFORMATICA**  
**INGEGNERIA BIOMEDICA**  
13 giugno 2017

**NOME:**

**COGNOME:**

**MATRICOLA:**

**ESERCIZIO 1 (7 punti)**

Progettare una rete logica ad un ingresso X e ad un'uscita Z che sia posta ad 1 al riconoscimento della sequenza di bit 0110111. Si utilizzino FF-T dove necessario. Disegnare lo schema a blocchi della rete (non è necessario disegnare il circuito nel dettaglio).

**ESERCIZIO 2 (16 punti)**

Sia data un'area di memoria primaria costituita da 128 parole, a partire dall'indirizzo di memoria 1024. Ciascuna parola della memoria primaria è di 32 bit ma si consideri indirizzabile il singolo byte: in altri termini, la prima parola, di indirizzo 0, sarà ripartita nei byte intermedi indirizzati dai valori 1, 2, 3, così che la seconda parola si troverà all'indirizzo 4 e così via.

- 1) (2 punti) Motivando la risposta, indicare la dimensione minima della memoria primaria in KB, sulla base di quanto indicato sopra.
- 2) (2 punti) Si supponga un livello di memoria cache di 256 parole con linee di 16 parole ciascuna, sempre costituite da 32 bit ed indirizzate secondo lo schema di cui sopra. 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 associativo su insiemi a due vie, utilizzando il numero di bit di indirizzamento minimo trovato nel punto 1.

- 3) (11 punti) Dato il seguente frammento di codice Assembly MIPS:

	addi \$11, \$0, 128		slt \$1, \$2, \$0
	addi \$10, \$0, 10		beq \$1, \$0, label 4
	move \$8, \$0		add \$2, \$2, \$5
label1:	beq \$8, \$10, label5	label4:	sw \$2, 1024(\$12)
	move \$9, \$0		addi \$9, \$9, 1
label2:	beq \$9, \$11, label3		j label2
	muli \$12, \$9, 4	label3:	addi \$8, \$8, 1
	lw \$2, 1024(\$12)		j label1
	sub \$2, \$2, \$3	label5:	...

Precisare, motivando chiaramente la risposta:

- 3.1) (6 punti) Cosa fa ciascuna istruzione Assembly MIPS mediante commenti opportuni aggiunti al codice e trascrivere in pseudo-codice o codice C quanto svolto dal frammento.
- 3.2) (3 punti) Ipotizzando la cache di cui al punto 2 inizialmente vuota, lo stato finale della stessa, dopo l'esecuzione di questo frammento, con riferimento all'area di memoria specificata.
- 3.3) (2 punti) L'hit ratio di cache.
- 3.4) (1 punto) Il tempo medio di accesso alla gerarchia a due livelli considerando per il calcolo i seguenti tempi  $T_p=40$  ns,  $T_c=4$  ns, e per l'hit ratio di cache il valore calcolato al punto 3.3.

**ESERCIZIO 3 (6 punti)**

Si consideri la rappresentazione di numeri binari in virgola mobile secondo lo standard IEEE 754 per un campo di 32 bit. Motivando chiaramente ogni passaggio:

1. (2 punti) Spiegare in cosa consiste detta rappresentazione.
2. (2 punti) Indicare minimo e massimo valore rappresentabili, in valore assoluto ed escluso lo zero.
3. (2 punti) Rappresentare e sommare, secondo l'algoritmo dei calcolatori, i valori 174.25 e 35.75.

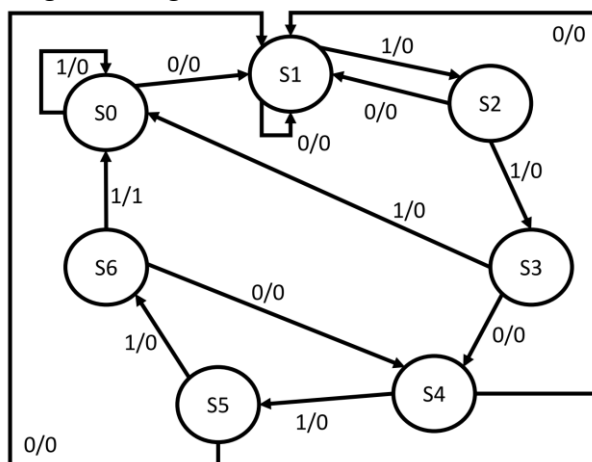
**ESERCIZIO 4 (4 punti)**

Scrivere e spiegare il protocollo di arbitraggio centralizzato a festone con due periferiche e tre linee ACK, REQ, GRANT, dal momento di richiesta del bus della prima periferica, e supponendo che durante la trasmissione dati della prima periferica, il bus venga richiesto dalla seconda.

## Soluzioni

### ESERCIZIO 1

Dalle specifiche si ricava il grafo degli stati:



Indicando gli stati con la codifica a tre bit:  $S0 \rightarrow 000$ ,  $S1 \rightarrow 001$ , ...,  $S6 \rightarrow 110$ , la tabella delle transizioni è:

A	B	C	X	A'	TA	B'	TB	C'	TC	Z
0	0	0	0	0	0	0	0	1	1	0
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	0	0
0	0	1	1	0	0	1	1	0	1	0
0	1	0	0	0	0	0	1	1	1	0
0	1	0	1	0	0	1	0	1	1	0
0	1	1	0	1	1	0	1	0	1	0
0	1	1	1	0	0	0	1	0	1	0
1	0	0	0	0	1	0	0	1	1	0
1	0	0	1	1	0	0	0	1	1	0
1	0	1	0	0	1	0	0	1	0	0
1	0	1	1	1	0	1	1	0	1	0
1	1	0	0	1	0	0	1	0	0	0
1	1	0	1	0	1	0	1	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)

Sulla base della tabella, abbiamo le seguenti mappe di Karnaugh per la semplificazione della rete di transizione dello stato:

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

$$T_A = ABX + A\bar{B}\bar{X} + BC\bar{X}$$

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

$$T_B = B\bar{X} + AB + CX$$

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

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

Per quanto riguarda l'uscita  $Z = AB\bar{C}X$ . Si lascia allo studente il disegno dello schema a blocchi, comunque recuperabile dalle slide del corso (Cap. 2).

## ESERCIZIO 2

Soluzione alla domanda 1.

Per sapere il numero di bit di indirizzamento, dobbiamo conoscere l'indirizzo massimo raggiungibile dalle 128 parole a partire dall'indirizzo 1024. Poiché ognuna di queste 128 parole è costituita da 4 byte indirizzabili, abbiamo:  $128 \cdot 4 = 512$  byte in tutto, ciascuno indirizzabile. L'ultimo indirizzo utile è quindi  $1024 + 511 = 1535$ . La memoria primaria deve essere costituita da almeno 1536 byte indirizzabili, ovvero 384 parole da 32 bit.

Poiché con 10 bit possiamo indirizzare solo 1024 byte, dobbiamo utilizzare almeno 11 bit coi quali possiamo indirizzare 2048 byte, ovvero 512 parole da 32 bit.

La dimensione minima della memoria primaria è dunque 2 KB, con indirizzamento a 11 bit dei singoli byte.

Soluzione alla domanda 2.

Una memoria cache di 256 parole necessita 8 bit di base, ai quali dobbiamo aggiungerne due nella parte meno significativa in quanto ogni parola è costituita da 4 byte indirizzabili. Poiché ogni linea è formata da 16 parole, che necessitano 4 bit, avremo 6 bit in tutto per l'offset. Il cache index è invece formato da 4 bit in quanto le linee sono 16. Poiché sono raggruppate a due alla volta, per via del metodo associativo su insiemi a due vie, il cache index "cede" il bit più significativo al TAG divenendo di soli 3 bit.

Quindi si avrà:

TAG: 2 bit – Set (cache) index: 3 bit – Offset: 6 bit

Soluzione alla domanda 3.

3.1)

Il codice Assembly:

addi \$11, \$0, 128	slt \$1, \$2, \$0
addi \$10, \$0, 10	beq \$1, \$0, label 4
move \$8, \$0	add \$2, \$2, \$5
label11: beq \$8, \$10, label5	label14: sw \$2, 1024(\$12)
move \$9, \$0	addi \$9, \$9, 1
label12: beq \$9, \$11, label3	j label2
mul \$12, \$9, 4	label13: addi \$8, \$8, 1
lw \$2, 1024(\$12)	j label11
sub \$2, \$2, \$3	label15: ...

Sviluppa il seguente codice C, associando opportune variabili ai registri:

$n \rightarrow \$10$ ,  $m \rightarrow \$11$

$A = 1024$

$i \rightarrow \$8$ ,  $j \rightarrow \$9$ ,  $h \rightarrow \$3$ ,  $k$  (copia di  $A[j]$ )  $\rightarrow \$2$ ;  $z \rightarrow \$5$

$m = 128$ ;

$n = 10$ ;

**for** ( $i=0$ ;  $i < n$ ;  $i++$ )

**for** ( $j=0$ ;  $j < m$ ;  $j++$ )

    {

$k = A[j]$ ;

$k = k - h$ ;

**if** ( $k < 0$ )  $k = k + z$ ;

$A[j] = k$ ;

    }

...

Si lascia allo studente il commento delle singole istruzioni Assembly MIPS richiesto dall'esercizio.

3.2)

In base a quanto ricavato nel punto precedente, si ha per 10 volte consecutive una doppia chiamata alle 128 parole a partire dall'indirizzo 1024, la prima volta in lettura (istruzione  $lw$ ) la seconda volta in scrittura (istruzione  $sw$ ). Le parole, sebbene alterate nei contenuti (cosa che interessa l'hit ratio, non tanto lo stato finale), sono comunque corrispondenti agli stessi indirizzi.

128 parole corrispondono a 8 blocchi di primaria, ed altrettante linee di cache. La prima parola di indirizzo 1024, e relativo blocco di altre 15 parole, verrà memorizzata nella prima linea libera del set 0. Il secondo blocco verrà memorizzato nella prima linea libera del set 1 e così via, fino al set 7, la cui prima linea libera verrà occupata dall'ottavo blocco di parole chiamato dal sistema.

Le chiamate si ripetono altre nove volte ma a quel punto tutte le parole sono in cache dove indicato e là verranno alterate fino a fine programma, dopo il quale le parole in cache verranno copiate in primaria per mantenere intatta la coerenza dei contenuti delle memorie della gerarchia.

Lo stato finale della cache è dunque il seguente:

	Set 0	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7
Linea 0	1024- 1087	1088- 1151	1152- 1215	1216- 1279	1280- 1343	1344- 1407	1408- 1471	1472- 1535
Linea 1	-	-	-	-	-	-	-	-

3.3)

L'hit ratio esprime il rapporto tra il numero di volte in cui una parola chiamata è stata trovata in cache (in lettura o scrittura) ed il numero complessivo di chiamate.

Come si evince dal ragionamento fatto nel punto precedente, è possibile sostenere un numero complessivo di 8 miss, ovvero parole non trovate in lettura, soltanto al primo accesso, con 15 hit in lettura, ciascuno di questi seguito da 1 hit in scrittura, per un totale di 16 hit in scrittura. Dal successivo abbiamo 16 hit in lettura e 16 in scrittura.

Per cui abbiamo alla prima sequenza di chiamate:  $15+16=31$  hit per ciascun blocco di primaria (linea di cache), e dalla seconda sequenza in poi abbiamo sempre 32 hit complessivi. Quindi si ha, approssimando alla terza cifra frazionaria:

$$H_c = \frac{31 \cdot 8 + 32 \cdot 8 \cdot 9}{32 \cdot 8 \cdot 10} = \frac{319}{320} = 0.997$$

3.4)

Si ha dalla formula:

$$T = T_c + (1 - H_c) \cdot T_p = 4 + 0.003 \cdot 40 = 4.12ns$$

### ESERCIZIO 3

Soluzione alla domanda 1.

La rappresentazione IEEE 754 per numeri a 32 bit è nella forma:

$$\pm 1.M \cdot 2^{e+127} = \pm \left( 2^0 + \sum_{i=-1}^{-23} b_i \cdot 2^i \right) \cdot 2^{e+127}$$

Con  $M = \{b_{-1}, \dots, b_{-23}\}$ , i 23 bit di parte frazionaria in segno e valore ed  $e$  esponente, rappresentato, secondo lo standard richiesto, con 8 bit in eccesso 127. Vi è anche un bit implicito relativo all'unico in parte intera.

Soluzione alla domanda 2.

Il minimo valore si ha in corrispondenza di una mantissa tutta di 0, con l'1 in corrispondenza del bit implicito. Il minimo esponente si ricava dall'eccesso  $k$ , e poiché l'eccesso è 127 il minimo esponente è pari a -127.

Il massimo valore si ha in corrispondenza di una mantissa tutta di 1, alla quale va aggiunto il bit implicito. Il massimo esponente si ricava a partire dal massimo valore ottenibile con otto bit, pari a 255, al quale va sottratto l'eccesso 127, ottenendo così 128.

Quindi:

Minimo valore:  $1.0 \dots 0 \cdot 2^{-127} = 2^{-127}$

Massimo valore:  $1.1 \dots 1 \cdot 2^{128} = \left( 1 + \sum_{i=-1}^{-23} 2^i \right) \cdot 2^{128} = (2 - 2^{-23}) \cdot 2^{128}$

Soluzione alla domanda 2.

Convertendo parte intera e parte frazionaria dei valori dati si ha:

$$174.25_{10} = 10101110.01_2 = 1.010111001 \cdot 2^{7+127} = 1.010111001 \cdot 2^{134}$$

La conversione di 7 in eccesso 127 è semplice:  $7 + 127 = 134 \rightarrow 10000110$

$$35.75_{10} = 100011.11_2 = 1.0001111 \cdot 2^{5+127} = 1.0001111 \cdot 2^{132}$$

La conversione di 5 in eccesso 127 è semplice:  $5 + 127 = 132 \rightarrow 10000100$

La rappresentazione dei due numeri in binario è dunque:

174.25 <sub>10</sub>	0 10000110 010111001000000000000000
35.75 <sub>10</sub>	0 10000100 000111100000000000000000

Soluzione alla domanda 3.

L'algoritmo della somma di due numeri in virgola mobile dei calcolatori prevede i seguenti passaggi:

- 1) Confronto degli esponenti
- 2) Allineamento delle mantisse rispetto al numero con massimo esponente
- 3) Somma delle mantisse
- 4) Eventuale normalizzazione del risultato

Riprendendo i quattro passaggi:

- 1)  $7-5=2$ . Essendo la differenza positiva, il secondo esponente è più piccolo del primo
- 2) Scorrimento della mantissa del valore più piccolo di due posizioni a destra (equivalente ad una divisione per quattro)
- 3) Somma delle mantisse allineate:

$$1.010111001000000000000000+$$

$$0.000011110000000000000000=$$

$$1.011010111000000000000000=$$

- 4) Non vi è bit di riporto rispetto all'1 più significativo, quindi possiamo scrivere:

210.00 <sub>10</sub>	0 10000110 011010111000000000000000
----------------------	-------------------------------------

#### ESERCIZIO 4

Il riferimento è al cap. 7, ed in particolare alla seguente figura dove il protocollo richiesto è perfettamente illustrato:

