

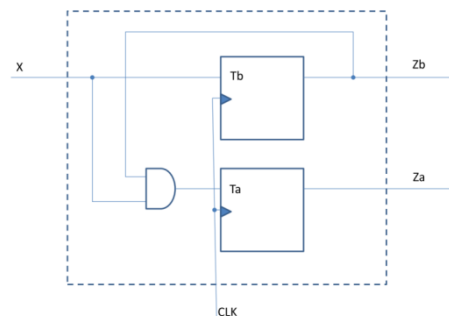
**PROVA SCRITTA DEL MODULO DI**  
**CALCOLATORI ELETTRONICI**  
 CORSO DI LAUREA IN INGEGNERIA ELETTRICA, ELETTRONICA, ED INFORMATICA  
 CORSO DI LAUREA IN INGEGNERIA BIOMEDICA  
 2 aprile 2019

**NOME:** \_\_\_\_\_ **COGNOME:** \_\_\_\_\_ **MATRICOLA:** \_\_\_\_\_ **CFU:** \_\_\_\_\_

**ESERCIZIO 1 (8 punti)**

Sulla base della rete logica tracciata in figura (X è l'ingresso, Za e Zb le uscite, CLK è il segnale di sincronismo):

- 1) (5 punti) Tracciare la tabella di transizione ed il grafo degli stati.
- 2) (3 punti) Dire quale funzione è espletata dalla rete motivando la risposta.



**ESERCIZIO 2 (8 punti)**

Si consideri una memoria primaria costituita da 512 B e una memoria cache costituita da 16 B, con blocchi di 4 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 8 a 15, da 40 a 47 in questo ordine, per due 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 di cache sulla base delle chiamate al punto precedente per entrambi i metodi.

**ESERCIZIO 3 (9 punti)**

Implementare una procedura Assembly MIPS che, dati l'indirizzo iniziale di un vettore di valori numerici v in \$4 e la sua dimensione N in \$5, restituisca il minimo valore contenuto nel vettore in \$6.

**ESERCIZIO 4 (8 punti)**

Siano dati 12 bit per la rappresentazione di valori numerici in virgola fissa ed in virgola mobile. Per la prima, si disponga di un campo di 7 bit per la parte intera e 4 per la parte frazionaria. Per la seconda, si consideri una mantissa M frazionaria e normalizzata in segno e valore con modalità 1.M (bit di parte intera implicito), esponente a 6 bit in eccesso 32, e bit di segno.

- 1) (2 punti) Spiegando bene ogni passo del ragionamento, indicare il minimo ed il massimo numero rappresentabili, in valore assoluto ed escluso lo zero, nei due casi.
- 2) (3 punti) Rappresentare i valori 38.75 e 11.25, qui espressi in notazione decimale, nella notazione in virgola mobile indicata nel testo, indicando chiaramente i valori dei bit nei campi segno, esponente e mantissa e quantificando l'eventuale perdita di precisione nell'operazione.
- 3) (3 punti) Sommare i valori al punto 2, secondo l'algoritmo di somma utilizzato nei calcolatori elettronici, spiegando nel dettaglio ogni passaggio intermedio e quantificando l'eventuale perdita di precisione nell'operazione.

## ESERCIZIO 1

Soluzione.

1)

Indicando con  $Q_a$  e  $Q_b$  le uscite dei FF-T con ingressi rispettivi  $T_a$  e  $T_b$ , si vede subito che:

$$T_a = Q_b * X$$

$$T_b = X$$

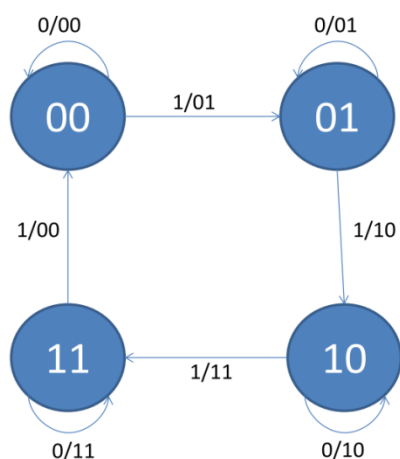
$$Z_a = Q_a$$

$$Z_b = Q_b$$

Le uscite  $Z_a$  e  $Z_b$  coincidono proprio con  $Q_a$  e  $Q_b$ .

Tabella di transizioni e grafo degli stati si ricavano dalle formule di cui sopra:

$Q_a$	$Q_b$	$X$	$T_a$	$Q_a'$	$T_b$	$Q_b'$
0	0	0	0	0	0	0
0	0	1	0	0	1	1
0	1	0	0	0	0	1
0	1	1	1	1	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	1
1	1	0	0	1	0	1
1	1	1	1	0	1	0



- 3) La struttura della rete e del grafo degli stati permette di dedurre che la funzione espletata è quella di un contatore ciclico 0-3 (00-11 in binario): si può infatti notare dalle uscite che esse commutano di un bit ogni volta che viene ricevuto un 1 (ovvero  $X=1$ ).

## ESERCIZIO 2

- Essendo la memoria primaria costituita da  $512 \text{ B} = 2^9 \text{ B}$ , l'indirizzamento è a 9 bit. Di questi, i due meno significativi compongono l'offset (i blocchi sono di quattro parole). Il numero di linee di cache è invece 4 ( $16 \text{ B} / 4 \text{ B}$ ), per cui il cache index è formato da 2 bit. I restanti 5 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 6 bit.
- 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 8. Dividendo per la dimensione dei blocchi si ha:  $8/4=2$  con resto 0. L'index si calcola con l'ulteriore divisione  $2/4$  che da resto 2. Si ottiene dunque che la parola 8 è la prima parola di un blocco di quattro parole (dalla 8 alla 11) che verranno tutte memorizzate nella linea 2 di cache. Il blocco successivo a questo contiene le parole dalla 12 alla 15 a completare il primo nucleo di parole consecutive, che verranno tutte allocate nella linea 3 di cache. Il secondo nucleo inizia dalla parola 40. Ripetendo lo stesso procedimento si ha  $40/4=10$ , con resto 0. Per trovare l'index si esegue l'ulteriore divisione  $10/4=2$  con resto 2. Anche questa parola, e le restanti dello stesso blocco (41, 42, 43) verranno allocate nella linea 2, sovrascrivendo le precedenti. Stesso dicasi per le parole 44-47 che sovrascriveranno quelle presenti nella linea 3. Le richieste dei due nuclei sono ripetute un'altra volta generando la stessa sequenza di scritture e sovrascritture. Alla fine lo stato della cache sarà:

<b>DIRETTO</b>	<b>Offset di parola</b>			
<b>Index di linea</b>	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>
<b>0</b>				
<b>1</b>				
<b>2</b>	40	41	42	43
<b>3</b>	44	45	46	47

Per quanto riguarda il metodo set associativo a due vie, il procedimento inizia calcolando la divisione  $8/4=2$  con resto 0. L'ulteriore divisione è  $2/2=1$  con resto 0. Quindi questa parola, e tutto il resto del blocco (parole 9-11) viene allocato nella prima linea libera dell'insieme 0. Il secondo gruppo di quattro parole, dalla 12 alla 15, viene invece allocato nella prima linea libera dell'insieme 1.

Il secondo nucleo inizia con la parola  $40/4=10$  con resto 0. Per calcolare il set index:  $10/2=5$  con resto 0. Anch'essa, assieme alle parole 41-43 componenti il blocco, vengono memorizzate nel set 0, ma nella seconda linea libera. Analogamente le parole dalla 45 alla 47 verranno memorizzate nel set 1, seconda linea libera. In questo caso non è stata necessaria alcuna sovrascrittura per cui la seconda chiamata dei numeri di parole le troverà tutte presenti in cache, secondo lo stato finale:

<b>SET ASSOCIATIVO</b>	<b>Offset di parola</b>			
<b>Index di set</b>	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>
<b>0</b>	8	9	10	11
	40	41	42	43
<b>1</b>	12	13	14	15
	44	45	46	47

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

Nel caso dell'indirizzamento diretto, abbiamo 3 hit per ciascuno dei blocchi chiamati (quattro in tutto) in entrambi i cicli. Quindi si ha:

$$H_c = \frac{3 \cdot 4 \cdot 2}{16 \cdot 2} = \frac{3}{4} = 0.75$$

Nel caso set-associativo, abbiamo 3 hit per ciascuno dei blocchi chiamati al primo ciclo, mentre al secondo ciclo abbiamo tutti hit. Si ottiene dunque:

$$H_c = \frac{3 \cdot 4 + 16}{16 \cdot 2} = \frac{7}{8} = 0.875$$

### ESERCIZIO 3

#### Soluzione

\$4 ← v[0]; \$5 ← N

```
max:      addi $29, $29, -16
          sw $4, 0($29)
          sw $8, 4($29)
          sw $9, 8($29)
          sw $10, 12($29)
          lw $6, 0($4)      #assumo che max <- v[0]
          addi $8, $0, 1    #inizializza i=1
          addi $4, $4, 4    #calcola in $4 <- &v[1]=&v[0]+4
for:      beq $8, $5, exit  #i==N, exit
          lw $9, 0($4)      #carico v[i] in $9
          slt $10, $9, $6    #$9 < $6 ->$10
          bne $10, $0, min   #se $10 diverso da 0, aggiorno il max
          j inc             #altrimenti salto a inc

min:      move $6, $9        #aggiorno il minimo
inc:      addi $8, $8, 1     #i++
          addi $4, $4, 4     #&v[i]++
          j for

exit:     lw $4, 0($29)
          lw $8, 4($29)
          lw $9, 8($29)
          lw $10, 12($29)
          addi $29, $29, 16
          jr $31
```

#### ESERCIZIO 4

1)

Per quanto riguarda i campi in virgola fissa, si ha che il valore minimo è ottenuto in corrispondenza di tutti i bit di parte intera pari a zero e un unico 1 nella posizione meno significativa della parte frazionaria. Quindi il valore minimo sarà pari a  $2^{-4}$ . Il valore massimo invece si ottiene ponendo ad 1 tutti i bit di parte frazionaria e intera, ottenendo quindi il valore  $2^7 - 2^{-4}$ .

Per quanto riguarda la rappresentazione in virgola mobile, abbiamo undici bit utili a parte il bit di segno, dei quali cinque per la mantissa e sei per l'esponente.

Il minimo valore rappresentabile si ottiene considerando il minimo valore dell'esponente, che essendo in eccesso 32, è, per definizione di eccesso k, pari a  $-k$ , ovvero  $-32$ , e il minimo valore della mantissa, corrispondente ad un campo formato da tutti zeri poiché il bit implicito per definizione è sempre impostato ad 1. Il minimo valore è dunque pari a  $1.0 \cdot 2^{-32}$ . Il massimo valore rappresentabile si ottiene valutando il massimo esponente, che si ricava dalla formula  $2^{n-1-k}$ , con n numero di bit usati per l'esponente. Il risultato è  $64-1-32=31$ . La massima mantissa, di cinque bit, si ottiene ponendo tutti i bit disponibili a 1, considerato anche il bit implicito, pari a  $2-2^{-5}$ . Il massimo valore rappresentabile è dunque  $(2-2^{-5}) \cdot 2^{31}$ , ovvero  $2^{32}-2^{26}$ .

2)

Per rappresentare i due valori 38.75 e 11.25 in virgola mobile, vanno innanzi tutto convertiti in binario mediante gli algoritmi della divisione successiva per la parte intera, e della moltiplicazione successiva per la parte frazionaria, ottenendo rispettivamente 100110.11 e 1011.01.

Non ci resta che normalizzare la mantissa ottenendo i relativi esponenti:  $1.0011011 \cdot 2^5$  e  $1.01101 \cdot 2^3$ . Gli esponenti vanno scritti con sei bit in eccesso 32. Per ottenere la rappresentazione in eccesso 32 è sufficiente sommare appunto l'eccesso al valore numerico da rappresentare, e poi convertirlo in binario. Si ha  $5+32=37 \rightarrow 100101$  e  $3+32=35 \rightarrow 100011$ .

Quindi i valori sono rappresentati nel campo a 8 bit come segue:

Valore	S	Esponente						Mantissa				
38.75	0	1	0	0	1	0	1	0	0	1	1	0
11.25	0	1	0	0	0	1	1	0	1	1	0	1

Come si può notare lo spazio per la mantissa non è sufficiente per memorizzare il primo valore, che risulterà così troncato a 38. Il bit di segno è lo stesso per entrambi i valori, dal momento che sono positivi (non è rilevante se sia 0 od 1).

3)

L'algoritmo dei calcolatori prevede i seguenti passaggi: allineamento delle mantisse degli operandi, somma delle mantisse degli operandi, normalizzazione della mantissa del risultato ed eventuale modifica dell'esponente.

Allineamento delle mantisse: la differenza fra gli esponenti del primo e del secondo numero è 2. Si divide quindi il secondo valore, più piccolo, per 4, onde avere la mantissa allineata con il numero maggiore.

Somma delle mantisse:  $1.00110 + 0.0101101 = 1.1000101$

Normalizzazione della mantissa e modifica dell'esponente, in questo caso non necessaria. Si ha comunque un'ulteriore perdita di precisione:

Valore	S	Esponente						Mantissa				
49.25?	0	1	0	0	1	0	1	1	0	0	0	1

La somma corrisponde al valore:  $1.10001 \cdot 2^5 = 110001 = 49$ .