

**SOLUZIONI DELLA PROVA SCRITTA DEL CORSO DI  
CALCOLATORI ELETTRONICI  
NUOVO E VECCHIO ORDINAMENTO DIDATTICO**  
28 Settembre 2006

**MOTIVARE IN MANIERA CHIARA LE SOLUZIONI PROPOSTE A CIASCUNO DEGLI ESERCIZI SVOLTI**

**ESERCIZIO 1 (8 punti)**

- 1) (NO: 4 punti –VO: 3 punti) Progettare un Full Adder. Mostrare la tabella di verità e le porte logiche che implementano tale rete combinatoria.
- 2) (NO: 4 punti – VO: 3 punti) Disegnare lo schema di un Parallel Adder con due addendi di quattro bit, precisando il ruolo di ogni Full Adder componente. Indicare cos'è e come viene calcolato il tempo di ritardo introdotto da un Parallel Adder. Assumere che 'd' sia il ritardo introdotto da un singolo modulo Full Adder.
- 3) (solo VO: 2 punti) Disegnare lo schema di un Carry Save-Adder con due addendi di quattro bit, precisando il ruolo di ogni Full Adder componente.

**Soluzione**

1) e 2) V. dispense del corso, capitolo 6, pp. 37-38.

Fig.1 mostra lo schema di un Parallel Adder con due addendi di quattro bit, indicati con A e B rispettivamente (A0 e B0 sono i bit meno significativi). C0 è il riporto in ingresso, Cout quello in uscita, C1...C3 i riporti intermedi. Se d è il tempo di ritardo del FA singolo, il tempo di ritardo complessivo è dato da  $4 \cdot d$ . Ricordiamo che il tempo di ritardo è il tempo richiesto al Parallel Adder per presentare le quattro uscite S0...S3 (il risultato della somma A+B). Poiché ogni bit somma Si dipende dal ritardo degli stadi precedenti, il ritardo complessivo è dato dalla somma dei ritardi dei singoli FA.

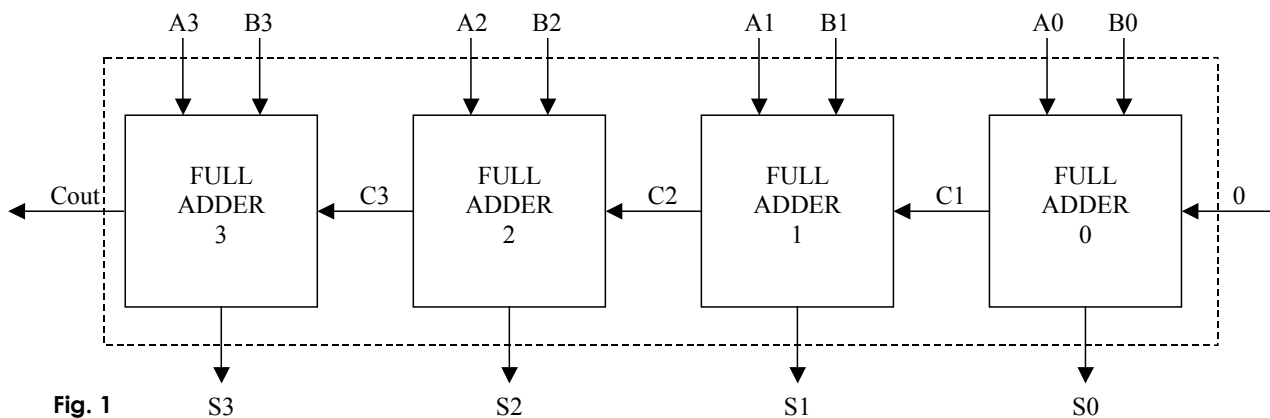


Fig.2 mostra lo schema di un Carry-Save Adder con due addendi di quattro bit. PS e PR non sono somma e riporto dei due bit, bensì soltanto Pseudo-Somma e Pseudo-Riporto. Per ottenere la somma complessiva, bisogna moltiplicare per 2 lo Pseudo-Riporto (shift) e poi effettuare la somma  $PS + 2PR$  con un Parallel Adder. E' evidente che in questo caso un CSA determina un ritardo complessivo superiore a quello di un Parallel Adder singolo.

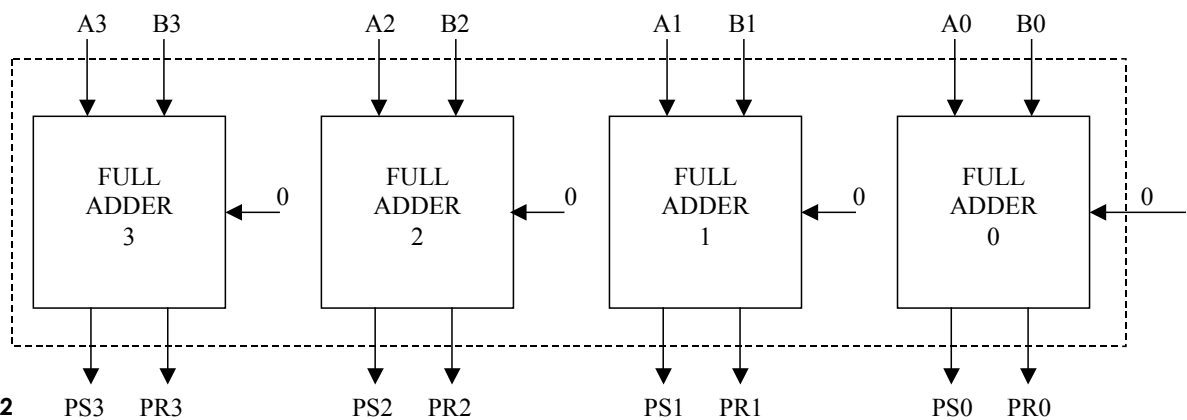


Fig. 2

## ESERCIZIO 2 (8 punti)

- (2 punti) Mostrare la suddivisione nei relativi campi TAG, Index, Offset di un indirizzo di memoria primaria di 1Kbyte, disponendo di una cache da 32 byte, secondo il metodo diretto con blocchi da 4 byte.
- (NO: 6 punti - VO: 4 punti) Si consideri la sequenza di riferimenti alla memoria indicati come indirizzi di parole (il primo indirizzo è 0): 389, 719, 387, 697, 306, 308, 194, 198, 699, 310. Indicare i cache *hit* e il contenuto finale della cache, nella configurazione al punto 1. E' sufficiente indicare, per ciascun blocco di cache, l'indirizzo del corrispondente blocco di primaria in esso eventualmente presente.
- (solo VO: 2 punti) Indicare almeno una possibile alternativa per aumentare il numero di hit in cache, senza alterare la dimensione di memoria primaria e di cache e il metodo di indirizzamento adottato.  
(Suggerimento. Ricordare il principio di località: chiamate vicine in ordine di tempo e di spazio)

### Soluzione:

- < TAG 5 bit> < Cache Index 3 bit> <Offset 2 bit>
- Applicando per ciascuna chiamata le formule per il calcolo di block frame e cache index, si ricavano i seguenti valori per ciascun set:

Indirizzo	389	719	387	697	306	308	194	198	699	310
Blocco primaria	97	179	96	174	76	77	48	49	174	77
Blocco cache	1	3	0	6	4	5	0	1	6	5
Hit									X	X

Il contenuto finale della cache è dunque:

Blocco Cache	0	1	2	3	4	5	6	7
Blocco Primaria	48	49		179	76	77	174	

- La cosa più semplice per ridurre il numero di miss in cache è accrescere la grandezza dei blocchi. Ciò si può notare dal fatto che le richieste in primaria distano spesso meno di otto parole (e.g. prima e terza chiamata, quinta, sesta e ultima chiamata). Ad es. possiamo utilizzare blocchi da 8 byte. In questo caso la cache conterrà solo 4 blocchi, con la seguente sequenza:

<b>Indirizzo</b>	389	719	387	697	306	308	194	198	699	310
<b>Blocco primaria</b>	48	89	48	87	38	38	24	24	87	38
<b>Blocco cache</b>	0	1	0	3	2	2	0	0	3	2
<b>Hit</b>			X			X		X	X	X

Come si nota il numero di hit è passato da 2 a 5.

### ESERCIZIO 3 (solo VO: 6 punti)

I trasferimenti di parole a/dalla memoria di un calcolatore sono codificate utilizzando il codice di Hamming. Si consideri la stringa di 12 bit 101001101110 (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) (1 punto) 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.

### Soluzione.

- 1) Deve venire rispettata la condizione:

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

dove K è il numero di bit di controllo inseriti. Essendo  $N + K = 12$ , 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:

$c_0$	$c_1$	$b_0$	$c_2$	$b_1$	$b_2$	$b_3$	$c_3$	$b_4$	$b_5$	$b_6$	$b_7$
1	0	1	0	0	1	1	0	1	1	1	0

Dove  $c_0...c_3$  sono i quattro bit costituenti il vettore di controllo, e  $b_0...b_7$  gli otto bit trasmessi. La sequenza ricevuta è 10111110.

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

$$c'_0 = b_0 \oplus b_1 \oplus b_3 \oplus b_4 \oplus b_6 = 0$$

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

$$c'_2 = b_1 \oplus b_2 \oplus b_3 \oplus b_7 = 0$$

$$c'_3 = b_4 \oplus b_5 \oplus b_6 \oplus b_7 = 1$$

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):

$$e_0 = c_0 \oplus c'_0 = 1$$

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

$$e_2 = c_2 \oplus c'_2 = 0$$

$$e_3 = c_3 \oplus c'_3 = 1$$

Poiché il vettore risultante 1011 non è nullo, vi è un errore nella stringa di 12 bit data e precisamente nella posizione indicata dal vettore di errore tradotto in notazione decimale. Il bit sbagliato è quindi l'undicesimo ( $b_6$ ), e la parola corretta è 10111100.

### ESERCIZIO 3 (solo NO: 9 punti)

Indicare in modo chiaro e sintetico qual è la funzione svolta da ognuno dei tre frammenti MIPS seguenti. Si accetta come risposta anche la traduzione in linguaggio C dei frammenti. In ogni caso sia chiaramente indicata la funzione dei registri e delle aree di memoria.

Frammento 1.	Frammento 2.	Frammento 3.
<pre>muli \$15, \$16, 4 lw \$14, 1024(\$15) add \$14, \$14, \$13 subi \$15, \$15, 4 sw \$14, 1024(\$15)</pre>	<pre>move \$16, \$0 lui \$16, 60<sub>10</sub> addi \$16, \$16, 2305<sub>10</sub></pre>	<pre>Loop: lw \$8, save(\$9)       add \$9, \$9, \$20       bne \$8, \$21, Loop</pre>

#### Soluzione.

Frammento 1.

Si osserva che il registro \$15 è utilizzato come offset per l'accesso ad una locazione di memoria con indirizzo base pari a 1024. Il registro \$14 è invece utilizzato per memorizzare i dati nella locazione di indirizzo 1024+[\$15] ([\$x] significa contenuto del registro \$x). A questo valore viene sottratto il contenuto di \$13, e successivamente viene memorizzato in un'area di memoria adiacente alla precedente, in quanto \$15 viene decrementato di un valore pari a 4.

Per tradurre questo frammento in C, è sufficiente ad esempio assegnare a \$13 una certa variabile *y* e utilizzare un tipo di dato indicizzato come un vettore indicato simbolicamente come *x[i]*, dove *i* è l'indice memorizzato in \$16. Quindi in C si ottiene l'istruzione:

$$x[i-1] = x[i] + y;$$

Frammento 2.

La presenza dell'istruzione *lui* indica chiaramente che si sta memorizzando nel registro \$16 un valore in modalità immediata eccedente il numero di bit a disposizione. Tale valore equivale a 60<sub>10</sub> nei 16 bit più significativi, a 2305<sub>10</sub> in quelli meno significativi. Il valore che si tenta di caricare è dunque: 3934465<sub>10</sub>. In C il frammento si può tradurre:

$$x = 3934465;$$

Frammento 3.

La presenza dell'istruzione di salto condizionato *bne* e il nome "loop" utilizzato per l'area di memoria contenente l'istruzione a cui il salto fa riferimento suggeriscono inequivocabilmente che il frammento implementi un ciclo di istruzioni. La condizione di terminazione del ciclo si può ricavare osservando i valori contenuti nei registri \$8 e \$21 nell'istruzione *bne*. Il valore in \$21 è utilizzato solo in *bne*, si può quindi ipotizzare si tratti di un valore di riferimento. Il valore in \$8 viene prelevato da un'area di memoria con indirizzo iniziale *save*, mentre l'offset è dato dal valore in \$9. Ricapitolando: la condizione di terminazione di ciclo prevede che il valore contenuto in \$8 caricato da un'area di memoria con indirizzo iniziale *save* e indicizzata da \$9 sia diverso da un valore contenuto in \$21. L'indice viene aggiornato attraverso il valore in \$20. In C il frammento si può tradurre come un ciclo *while*:

$$\text{while}(\text{save}[h] \neq k) \quad h=h+c;$$

**ESERCIZIO 4 (NO: 8 punti – VO: 6 punti)**

La memoria di un calcolatore è gestita con una tecnica di 'paginazione su richiesta'. Si consideri la seguente richiesta di pagine: 0, 7, 2, 7, 5, 8, 9, 2, 4, 5, 2, 5. Mostrare il "page trace" e calcolare il numero di "page faults" per una memoria di dimensione pari a quattro pagine, nei seguenti due casi:

- 1) strategia di rimpiazzamento delle pagine FIFO (NO: 4 punti – VO: 3 punti).
- 2) strategia di rimpiazzamento delle pagine LRU (NO: 4 punti – VO: 3 punti).

**Soluzione**

*Page trace nel caso di strategia di rimpiazzamento delle pagine FIFO (x = hit)*

tempo	1	2	3	4	5	6	7	8	9	10	11	12
Richieste	0	7	2	7	5	8	9	2	4	5	2	5
Pagine	0	7	2	2	5	8	9	9	4	4	2	5
		0	7	7	2	5	8	8	9	9	4	2
			0	0	7	2	5	5	8	8	9	4
					0	7	2	2	5	5	8	9
Hit				x				x		x		

Page faults: 9 - Hit ratio =  $3/12 = 1/4$

*Page trace nel caso di strategia di rimpiazzamento delle pagine LRU (x = hit)*

tempo	1	2	3	4	5	6	7	8	9	10	11	12
Richieste	0	7	2	7	5	8	9	2	4	5	2	5
Pagine	0	7	2	7	5	8	9	2	4	5	2	5
		0	7	2	7	5	8	9	2	4	5	2
			0	0	2	7	5	8	9	2	4	4
					0	2	7	5	8	9	9	9
Hit				x							x	x

Page faults: 9 - Hit ratio =  $3/12 = 1/4$

**ESERCIZIO 5 (solo VO: 5 punti)**

Illustrare in modo chiaro e sintetico i vantaggi delle architetture RISC rispetto alle architetture CISC.

**Soluzione.**

Vedi dispense del corso.