

**SOLUZIONI DELLA PROVA SCRITTA DEL CORSO DI  
CALCOLATORI ELETTRONICI  
NUOVO ORDINAMENTO DIDATTICO**  
14 Settembre 2005

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

**ESERCIZIO 1 (8 punti)**

- 1) (4 punti) Progettare una rete logica combinatoria che calcoli il complemento a 2 di un operando a tre bit.
- 2) (4 punti) Realizzare la stessa funzione utilizzando delle porte logiche ed un addizionatore parallelo.

**Soluzione.**

1)

Tabella di verità della rete in oggetto :

Ingressi			Uscite		
A	B	C	A'	B'	C'
0	0	0	0	0	0
0	0	1	1	1	1
0	1	0	1	1	0
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	0	1	1
1	1	0	0	1	0
1	1	1	0	0	1

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

$$A' = \overline{A}B + \overline{A}C + A\overline{B} \cdot \overline{C}$$

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

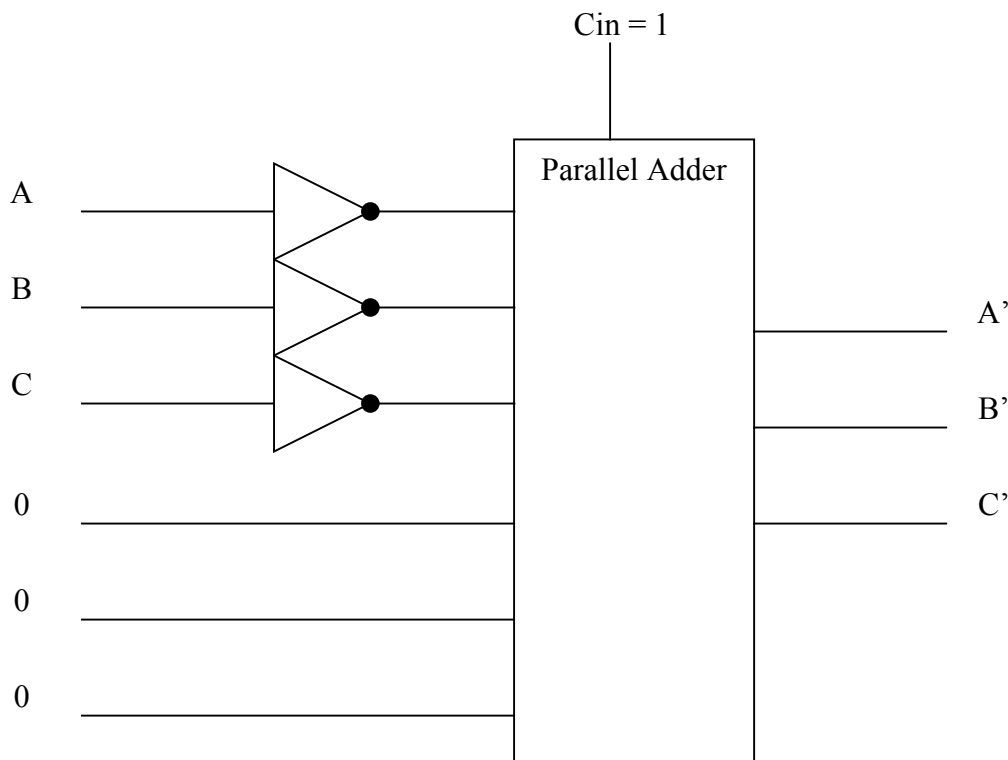
$$B' = B\overline{C} + \overline{B}C$$

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

$$C' = C$$

2)

La stessa funzione può essere realizzata con un addizionatore parallelo a cui vengono forniti i tre ingressi negati come primo operando ed il carry-in posto ad uno. I bit del secondo operando sono tutti a zero. Bastano quindi 3 porte NOT ed un addizionatore parallelo. La figura seguente mostra lo schema della rete logica.



## ESERCIZIO 2 (9 punti)

Si consideri un calcolatore che dispone di una memoria principale di 256 Mbyte e di una memoria cache di 512 Kbyte. E' possibile accedere al singolo byte e la memoria è suddivisa in blocchi da 16 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 venga usata la modalità di indirizzamento
  - (a) Diretto
  - (b) "associativo su insiemi", e ciascun insieme contenga due blocchi
- 2) (7 punti) Si consideri la cache di cui alla domanda precedente. Ipotizzare che il processore acceda in sequenza ai byte dall'indirizzo 0000000 a 00007FF e da 0180000 a 01807FF in questo ordine, e ripeta questa sequenza di accesso per 5 volte consecutive. Si ipotizzi inoltre che la cache sia inizialmente vuota. Calcolare il numero di *miss* sia nel caso di modalità ad indirizzamento diretto che nel caso di modalità ad indirizzamento associativo su insiemi spiegata nel punto precedente. Calcolare quindi l'hit ratio.

### Soluzione:

- 1) Per indirizzare 256 Mbyte occorre un indirizzo di almeno 28 bit. Per indirizzare il singolo byte all'interno di un blocco occorrono 4 bit ( $16 = 2^4$ ), che coincidono con i 4 bit meno significativi dell'indirizzo di memoria primaria. I restanti 24 bit costituiscono l'indirizzo del "block frame". Per indirizzare la cache, il "block frame" viene interpretato diversamente a seconda che l'indirizzamento sia di tipo "diretto" o "associativo su insiemi".
  - (a) Indirizzamento diretto. In questo caso devo poter indirizzare ciascuno dei 32K blocchi contenuti nella cache ( $512\text{Kbyte}/(16\text{byte}/\text{blocco})$ ). Occorrono 15 bit che coincidono con i 15 bit meno significativi del "block frame". Pertanto i 28 bit di indirizzo della memoria primaria vengono interpretati come:

tag	cache index	Offset
9 bit	15 bit	4 bit

(b) Indirizzamento "associativo su insiemi". In questo caso devo poter indirizzare ciascuno dei 16 insiemi in cui sono suddivisi i blocchi contenuti nella cache ( $\frac{512\text{Kbyte}}{\frac{2\text{blocchi}}{\text{insieme}} \cdot \frac{16\text{byte}}{\text{blocco}}} = 16\text{K}$  insiemi). Occorrono 14 bit che coincidono con i 14 bit meno significativi del "block frame"

Pertanto i 28 bit di indirizzo della memoria primaria vengono interpretati come:

tag	cache index	offset
10 bit	14 bit	4 bit

- 2) La memoria è divisa in blocchi di 16 byte ciascuno in modo che la richiesta di un dato non presente in cache causa il trasferimento del blocco a cui appartiene il dato richiesto dalla memoria principale alla cache. Nel caso proposto i dati richiesti sono così suddivisi (N.B. per semplicità i valori di index sono riportati in formato decimale, mentre gli indirizzi in esadecimale):

indirizzi 0000000 ÷ 000000F ⇒ index 0; indirizzi 0000010 ÷ 000001F ⇒ index 1; ... ; 00007F0 ÷ 00007FF ⇒ index 127; 0180000 a 018000F ⇒ index 0; ... ; 01807F0 a 01807FF ⇒ index 127. (N.B. in questa sequenza l'index è identico nel caso di indirizzamento diretto e associativo su insiemi).

La prima volta che viene richiesta la parola 0000000 avremo un "cache miss" che provoca il caricamento del blocco 0 nella linea di cache di indirizzo 0. Le successive richieste dei dati di indirizzo 0000001 ÷ 000000F vengono quindi soddisfatte dalla cache ("cache hit"). Analogamente avviene per tutti i blocchi fino al blocco 127, che vengono allocati in linee consecutive della cache fino alla 127. Quando viene richiesta la parola 0180000 (index 0), nel caso di indirizzamento diretto questa sovrascrive il blocco 0000000 ÷ 000000F e così via per tutte le richieste consecutive. Nel caso di indirizzamento associativo su insiemi a due vie uno stesso index corrisponde a un insieme di due blocchi. Pertanto le parole da 0180000 a 01807FF non sovrascrivono quelle precedentemente inserite, ma vengono memorizzate nel secondo blocco disponibile in ciascun insieme memorizzato.

Nei cicli successivi al primo il processore richiede nuovamente tutti i dati, a partire da 0. Nel caso di indirizzamento diretto avremo un miss per il caricamento della prima parola di ciascun blocco e 15 hit per le parole dello stesso blocco, per tutti i 256 blocchi. Nel caso di indirizzamento associativo su insiemi si hanno solo hit perché tutti i blocchi sono presenti in cache.

In sintesi:

- nel caso di indirizzamento diretto avremo per ciascun ciclo 256 miss, quindi 1280 miss in totale (256\*5);
- nel caso di indirizzamento associativo su insiemi a due vie avremo soltanto 256 miss in totale, tutti relativi al primo ciclo.

Poiché il numero totale di accessi è sempre 256\*5\*16, si ricava facilmente l'hit ratio  $H$  dato che:

$$H = 1 - \frac{\text{numeroMiss}}{\text{numeroAccessi}}$$

Quindi  $H(\text{diretto})=0,9375$ ;  $H(\text{associativo su insiemi})=0,9875$ .

### ESERCIZIO 3 (8 punti)

Implementare in Assembler MIPS una funzione che, dati l'indirizzo iniziale di un vettore v (in \$4) e la sua dimensione N (in \$5), immetta nel vettore w (ind. Iniziale in \$6) i valori di v strettamente positivi e immetta tutti gli altri nel vettore u (ind. Iniziale in \$7). Anche w e u hanno dimensione massima pari a  $N > 0$ . Vincolo: si usi il minimo numero di registri, senza preoccuparsi di preservare i valori contenuti nei registri destinati ai parametri della funzione.

#### Soluzione.

$\$9 \leftarrow v[i]; \$10 \leftarrow (v[i] > 0)$

funzione:	addi \$29, \$29, -8 sw \$9, 0(\$29) sw \$10, 4(\$29)	else:	sw \$9, 0(\$7) addi \$7, \$7, 4 j for
for:	beq \$5, \$0, exit subi \$5, \$5, 1 lw \$9, 0(\$4) slt \$10, \$0, \$9 beq \$10, \$0, else sw \$9, 0(\$6) addi \$6, \$6, 4 j for	exit:	lw \$9, 0(\$29) lw \$10, 4(\$29) jr \$31

### ESERCIZIO 4 (8 punti)

Considerato un campo di 64 bit, siano dati i seguenti formati:

1. rappresentazione di interi senza segno;
2. rappresentazione in virgola fissa con 20 bit di parte frazionaria;
3. rappresentazione in virgola mobile con mantissa frazionaria e normalizzata in segno e valore (1.M) ed esponente a 8 bit in eccesso 127.

a) (4 punti) Calcolare il minimo e il massimo valore rappresentabile in valore assoluto nei tre casi.

b) (4 punti) Sommare i due numeri,  $(12.5)_{10}$   $(5.25)_{10}$ , esprimendoli in virgola mobile secondo la rappresentazione 3, con l'algoritmo dei calcolatori.

#### Soluzione

a)

1. Minimo: 1 Max:  $2^{64}-1$ .

2. Minimo:  $2^{-20}$  Max:  $2^{43}-2^{-20}$

3. Minimo:  $2^{-127}$  Max:  $2^{128}(2-2^{-55})$ .

b)  $(12.5)_{10} = 1100.1 = 1.1001 \cdot 2^3$

$(5.25)_{10} = 101.01 = 1.0101 \cdot 2^2$

I due numeri si possono rappresentare nel seguente modo:

Segno	Esponente	Mantissa
0	10000010	10010000000000000000...0
0	10000001	01010000000000000000...0

Poiché il primo ha esponente maggiore del secondo ( $3 > 2$ ) di quest'ultimo si fa scorrere la mantissa a destra di una posizione.

I due numeri da sommare sono:

$$\begin{array}{r} 1.10010 + \\ 0.10101 = \\ \hline 10.00111 \quad (*2^3) \end{array}$$

E' necessario normalizzare il risultato

Segno	Esponente	Mantissa
0	10000011	00011100000000000000...0