

**SOLUZIONI DELLA PROVA SCRITTA DEL CORSO DI**  
**CALCOLATORI ELETTRONICI**  
**NUOVO ORDINAMENTO DIDATTICO**  
25 Gennaio 2008

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

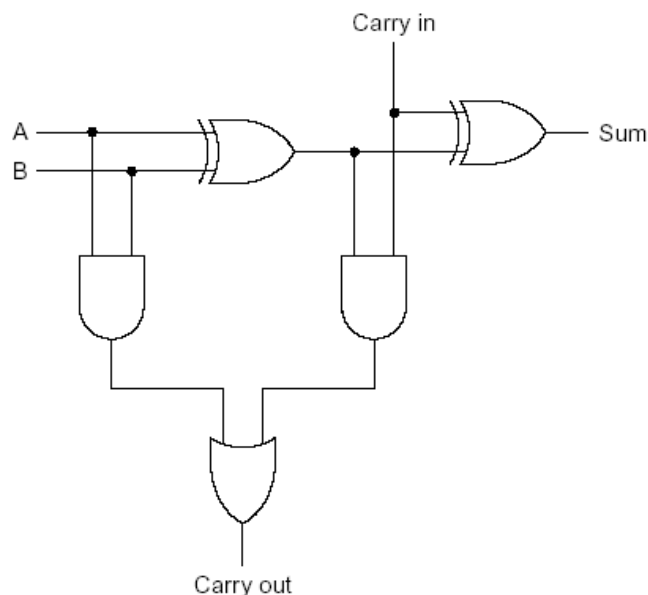
**ESERCIZIO 1 (9 punti)**

1. (4 punti) Si scriva la tabella di verità di un Full Adder e si rappresenti il relativo circuito impiegando il numero minimo di porte logiche.
2. (5 punti) Si definisca una rete logica realizzabile con N Full Adder, con relative caratteristiche e schema logico.

**Soluzione.**

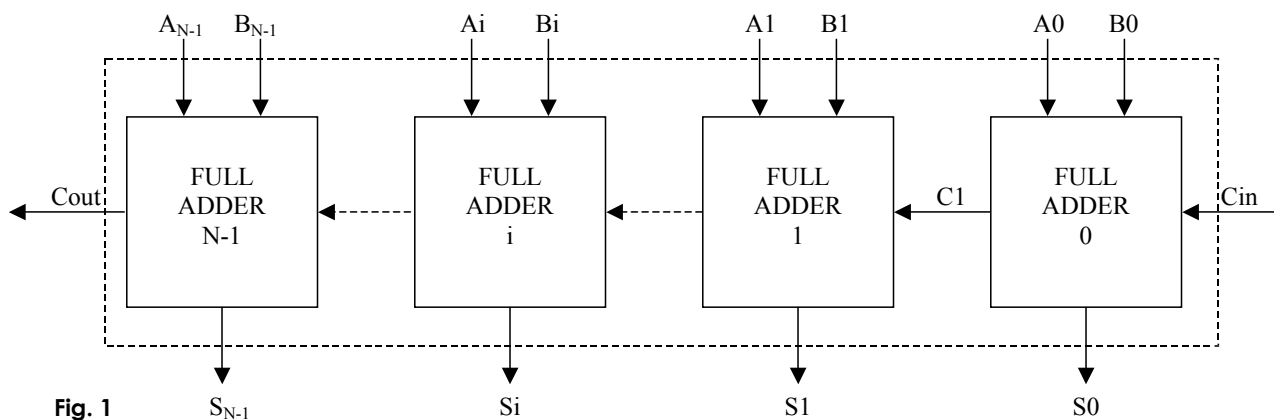
1.

A	B	Carry in	Sum	Carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

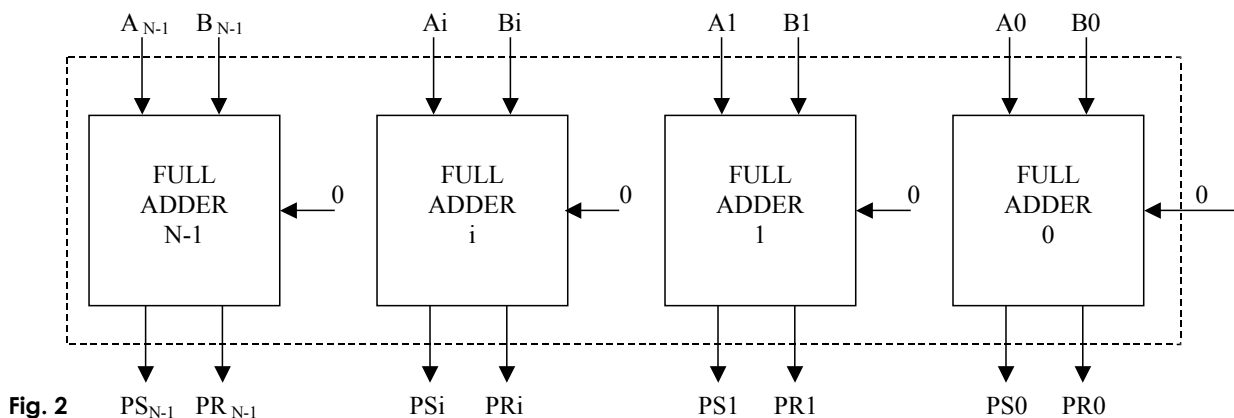


2.

Con N Full Adder è possibile realizzare un Parallel Adder. Se infatti si hanno due parole di N bit da sommare, i bit i-esimi di ciascuna parola sono gli ingressi del Full Adder i-esimo ( $i=0, \dots, N-1$ , col bit in posizione 0 nella funzione di bit meno significativo). Il riporto in uscita del Full Adder i-esimo va collegato al riporto in ingresso del Full Adder (i+1)-esimo. Il bit di riporto in ingresso del Parallel Adder corrisponde così al bit in ingresso del Full Adder relativo ai bit meno significativi. Lo schema progettuale è il seguente:



Un'altra rete implementabile è il Carry Save-Adder, costituito da N Full Adder in parallelo:



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.

## ESERCIZIO 2 (8 punti)

Si scriva il codice Assembler MIPS di una funzione che, ricevendo in ingresso l'indirizzo iniziale di un vettore di interi  $v$ , la sua dimensione  $N$ , un secondo vettore di interi  $w$ , la sua dimensione  $M$ , scriva nel vettore  $w$  il numero di occorrenze degli interi da 0 a  $M-1$ . In altri termini  $w[j]$  conterrà il numero di occorrenze del valore  $j$  nel vettore  $v$ . Si assuma che il contenuto di  $w$  sia inizializzato a 0.

Allocazione dei parametri di ingresso:  $\&v[0] \rightarrow \$4$ ,  $N \rightarrow \$5$ ,  $\&w[0] \rightarrow \$6$ ,  $M \rightarrow \$7$ .

Il codice MIPS potrebbe implementare il seguente codice C:

```
void occorrenze(int v[], int N, int w[], int M) {
    int i,j;

    for(i=0; i<N; i++)
    {
        j=v[i];
        if((j<M) && (j>=0))
            w[j]++;
    }
}
```

### Soluzione.

$\$8 \leftarrow i$ ;  $\$9 \leftarrow v[i]$ ;  $\$10 \leftarrow w[j]$ ;  $\$11 \leftarrow (v[i]<M)$ ;  $\$12 \leftarrow (v[i]<0)$ ;  
 $\$13 \leftarrow \&w[j]$

```
occorrenze:  addi $29, $29, -24      #salvataggio contesto
              sw $8, 0($29)
              sw $9, 4($29)
              sw $10, 8($29)
              sw $11, 12($29)
              sw $12, 16($29)
              sw $13, 20($29)
              move $8, $0           #i=0
for:         beq $8, $5, exit
              lw $9, 0($4)          #j=v[i]
              addi $4, $4, 4        #aggiorno indici
              addi $8, $8, 1
              slt $11, $9, $7       # $11 ← (v[i]<M)
              beq $11, $0, for      #if !($11) then for
              slt $12, $9, $0       # $12 ← (v[i]<0)
              bne $12, $0, for      #if ($12) then for
              muli $13, $12, 4
              add $13, $13, $6      #calcolo indirizzo di w[j]
              lw $10, 0($13)        #
              addi $10, $10, 1      #w[j]++
              sw $10, 0($13)        #
              j for
exit:        lw $8, 0($29)          #ripristino contesto
              lw $9, 4($29)
              lw $10, 8($29)
              lw $11, 12($29)
              lw $12, 16($29)
              lw $13, 20($29)
              addi $29, $29, 24
              jr $31               #ritorno a chiamante
```

### ESERCIZIO 3 (8 punti)

Si consideri una memoria primaria costituita da 32 parole e una memoria cache costituita da 8 parole, con blocchi da 2 parole. Si considerino i tre metodi di indirizzamento: diretto, associativo su insiemi a 2 vie e completamente associativo.

- (5 punti) Indicare lo stato finale della cache nel caso il processore acceda in sequenza alle parole di indirizzo da 0 a 3 e da 16 a 19 in questo ordine, e ripeta la sequenza di accesso per due volte, nei tre casi di indirizzamento indicati. Gli indirizzi dati sono espressi nel formato decimale.
- (3 punti) Nell'ipotesi che il tempo di accesso alla memoria primaria sia 10 volte il tempo di accesso alla memoria cache, indicare di quanto si riduce il tempo di accesso alla memoria con l'uso della cache nei tre casi di indirizzamento considerati nelle domande precedenti.

#### Soluzione.

1.

Metodo diretto: < TAG 2 bit > < Index 2 bit > < Offset 1 bit >  
Metodo set-associativo: < TAG 3 bit > < Index 1 bit > < Offset 1 bit >  
Metodo associativo: < Block Frame (TAG) 4 bit > < Offset 1 bit >

Stato finale della cache nei tre casi:

Linea	Diretto	Insieme	Set-Associativo	Associativo
0	16 17	0	0 1	0 1
1	18 19	1	16 17	2 3
2			2 3	16 17
3			18 19	18 19

Le parole 0-1 e 2-3 presentano index pari a 0 e 1 rispettivamente. Le parole 16-17 e 18-19 presentano lo stesso valore di index delle precedenti. Col metodo diretto vengono pertanto mappate nelle stesse linee di cache (la prima e la seconda, come mostrato in figura). Col metodo set-associativo, invece, pur presentando lo stesso index, possono venire mappate in blocchi eventualmente liberi dello stesso insieme. Infine, col metodo associativo l'index non è presente (NB è l'analogo del metodo associativo su insiemi a quattro vie) quindi i blocchi di primaria vengono allocati nei corrispondenti blocchi liberi di cache, nell'ordine di chiamata.

2.

Per calcolare l'entità della riduzione bisogna valutare il rapporto fra tempo medio di accesso alla gerarchia cache-primaria (quindi il tempo medio richiesto con uso della cache) e tempo medio di accesso della sola primaria (senza uso della cache). Indicando con T il tempo medio di accesso alla gerarchia cache-primaria della domanda precedente e con  $T_p$  il tempo di accesso in primaria, l'entità della riduzione è data da:

$$1 - T/T_p$$

Il tempo medio di accesso alla gerarchia cache-primaria è pari a:

$$T = T_c + (1 - H) * T_p$$

Con  $T_c$ : tempo di accesso cache e  $H$  = hit ratio cache.

Poiché  $T_p = 10 * T_c$ , si ha

$$T = T_c + (1 - H) * 10 * T_c$$

Il rapporto  $T/T_p$  è dunque funzione di  $H$  ed è pari a:

$$1 - (1 + (1 - H) * 10)/10$$

Poiché si ha:

Metodo diretto:  $H = 8/16 = 0.5$

Metodo set-associativo e associativo:  $H = (4 + 8) / 16 = 0.75$

Si ha che con l'indirizzamento diretto la riduzione è pari a 0.4 e negli altri due casi la riduzione è pari a 0.65. In altri termini la riduzione è del 40% nel primo caso e del 65% negli altri due.

**ESERCIZIO 4 (8 punti)**

1. (4 punti) Il sistema di memoria virtuale di un calcolatore viene gestito mediante la tecnica di "paginazione su richiesta". Il sistema operativo indirizza 8 pagine virtuali, mentre la memoria del calcolatore contiene 4 pagine fisiche. La dimensione di ciascuna pagina è uguale a 1024 parole. Ad un certo istante si abbia il contenuto della PMT riportata a lato.  
Elencare tutti gli indirizzi virtuali, espressi in decimale, che provocano un "page fault".
2. (4 punti) Spiegare in modo chiaro e sintetico in cosa consiste il problema del "trashing" nella paginazione su richiesta, e indicare i principali algoritmi di "trashing" utilizzati.

Indirizzo di pagina virtuale	Indirizzo di pagina fisica
0	2
1	-
2	-
3	-
4	3
5	-
6	1
7	0

**Soluzione**

1.

Gli indirizzi virtuali che generano un "page fault" sono quelli che corrispondono a pagine virtuali non caricate in memoria. Nell'esempio riportato si tratta delle pagine 1, 2, 3, 5, cui corrispondono rispettivamente gli indirizzi virtuali da 1024 a 2047, da 2048 a 3071, da 3072 a 4095 e da 5120 a 6143 (NB: l'indirizzo della prima parola della pagina  $x$  si calcola come  $x \cdot 1024$ , mentre l'indirizzo dell'ultima parola è  $(x + 1) \cdot 1024 - 1$ ).

2.

Vedi lucidi del corso.