

**PROVA SCRITTA DEL CORSO DI**  
**CALCOLATORI ELETTRONICI**  
**NUOVO ORDINAMENTO DIDATTICO**  
24 Settembre 2009

**NOME:**

**COGNOME:**

**MATRICOLA:**

**ESERCIZIO 1 (9 punti)**

1. (4 punti) Progettare un Full Adder. Mostrare la tabella di verità e le porte logiche che implementano tale rete combinatoria.
2. (5 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.

**ESERCIZIO 2 (6 punti)**

I trasferimenti di parole a/dalla memoria di un calcolatore sono codificati utilizzando il codice di Hamming. Si consideri la stringa di 12 bit 001001101110 (il bit meno significativo è a sinistra), risultata dalla codifica di una parola di N bit secondo il codice di Hamming.

1. (2 punti) calcolare N, supponendo di aver fatto uso del numero minimo di bit di controllo necessario per una stringa di 12 bit;
2. (1 punto) scrivere la parola di N bit a partire dalla stringa data;
3. (3 punti) indicare eventuali errori nella stringa codificata, specificando quale dei bit è stato alterato.

**ESERCIZIO 3 (9 punti)**

Si scriva il codice Assembly 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:  $\&v[0] \rightarrow \$4$ ,  $N \rightarrow \$5$ ,  $\&w[0] \rightarrow \$6$ ,  $M \rightarrow \$7$ .

Es. il codice MIPS potrebbe implementare il seguente codice C:

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

**ESERCIZIO 4 (9 punti)**

Si consideri il seguente insieme di processi.

Processo	Tempo di arrivo	Tempo di CPU richiesto	Memoria richiesta
1	0.00	2.00	125K
2	0.50	1.00	100K
3	1.00	0.20	40K
4	2.00	1.50	200K
5	2.70	1.00	115K

La memoria disponibile, di 500K, è gestita dal sistema operativo mediante partizionamento statico con quattro partizioni di 200K, 100K, 50K, 150K, con strategia di allocazione First Fit. All'istante iniziale, si consideri la memoria vuota. I processi sono gestiti con strategia FIFO multiprogrammata "round robin".

1. (6 punti) Mostrare graficamente lo sviluppo di ciascun processo e lo stato della memoria.
2. (3 punti) Calcolare il tempo di turnaround medio e di turnaround pesato medio.

## ESERCIZIO 1

### Soluzione

Si vedano le dispense del corso, capitolo 7, pp. 45-46.

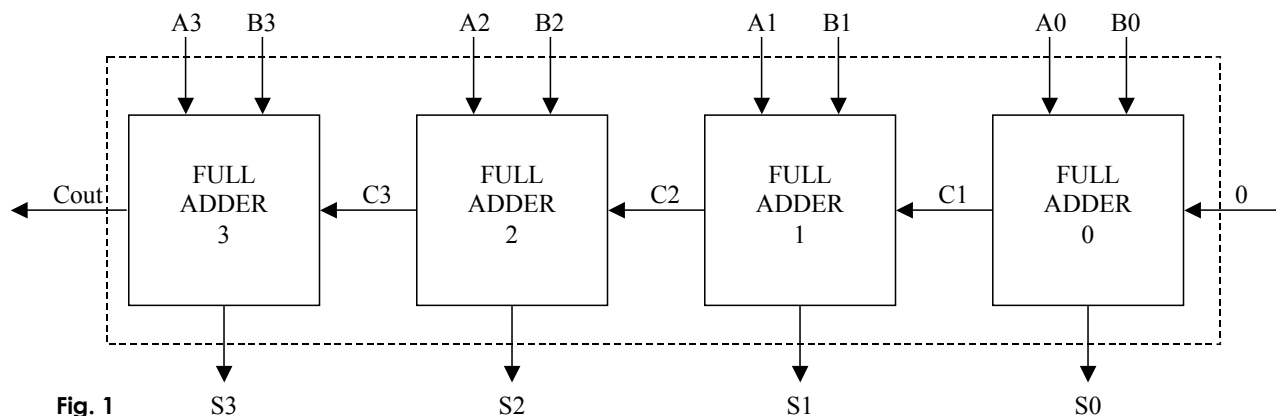


Fig. 1

Fig.1 mostra lo schema di un Parallel Adder con due addendi di quattro bit, indicati con A e B rispettivamente (A<sub>0</sub> e B<sub>0</sub> sono i bit meno significativi). C<sub>0</sub> è il riporto in ingresso, Cout quello in uscita, C<sub>1</sub>...C<sub>3</sub> i riporti intermedi. Se d è il tempo di ritardo del FA singolo, il tempo di ritardo complessivo è dato da 4\*d. Ricordiamo che il tempo di ritardo è il tempo richiesto al Parallel Adder per presentare le quattro uscite S<sub>0</sub>...S<sub>3</sub> (il risultato della somma A+B). Poiché ogni bit somma S<sub>i</sub> dipende dal ritardo degli stadi precedenti, il ritardo complessivo è dato dalla somma dei ritardi dei singoli FA.

## ESERCIZIO 2

### Soluzione

- 1) Deve essere 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:

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

Dove  $c_0 \dots c_3$  sono i quattro bit costituenti il vettore di controllo, e  $b_0 \dots b_7$  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):

$$e_0 = c_0 \oplus c'_0 = 0$$

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

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

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

Poiché il vettore risultante 1010 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 il decimo ( $b_5$ ), e la parola corretta è 10111010.

### ESERCIZIO 3

#### 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, -28      #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)
              sw $4, 24($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)
              lw $4, 24($29)
              addi $29, $29, 28
              jr $31                #ritorno a chiamante
```

### Soluzione

Istante 0.0: Processo 1 in partizione 1 (200K). Avvio del processo 1.

Istante 0.5: Processo 2 in partizione 2 (100K). Avvio del processo 2.

Istante 1.0: Processo 3 in partizione 3 (50K). Avvio del processo 3.

Istante 2.0: Non c'è spazio in memoria per allocare il processo 4, che viene posto in stato di attesa.

Istante 2.7: Processo 5 in partizione 4 (150K). Avvio del processo 5.

Istante 3.7: Processo 4 in partizione 1 (200K). Avvio del processo 4.

Da questo istante i processi vengono gestiti regolarmente con FIFO multiprogrammata "round robin" fino al termine di tutti i processi.

Processo	Arrivo	Avvio	Termine	Turnaround	W.Tourn.
1	0.00	0.00	3.70	3.70	1.85
2	0.50	0.50	2.70	2.20	2.20
3	1.00	1.00	1.60	0.60	3.00
4	2.00	3.70	5.70	3.70	2.47
5	2.70	2.70	4.70	2.00	2.00
<b>Media</b>				2.44	2.30

