

**SOLUZIONI DELLA PROVA SCRITTA DEL CORSO DI**  
**CALCOLATORI ELETTRONICI**  
**NUOVO E VECCHIO ORDINAMENTO DIDATTICO**  
25 Settembre 2003

**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.

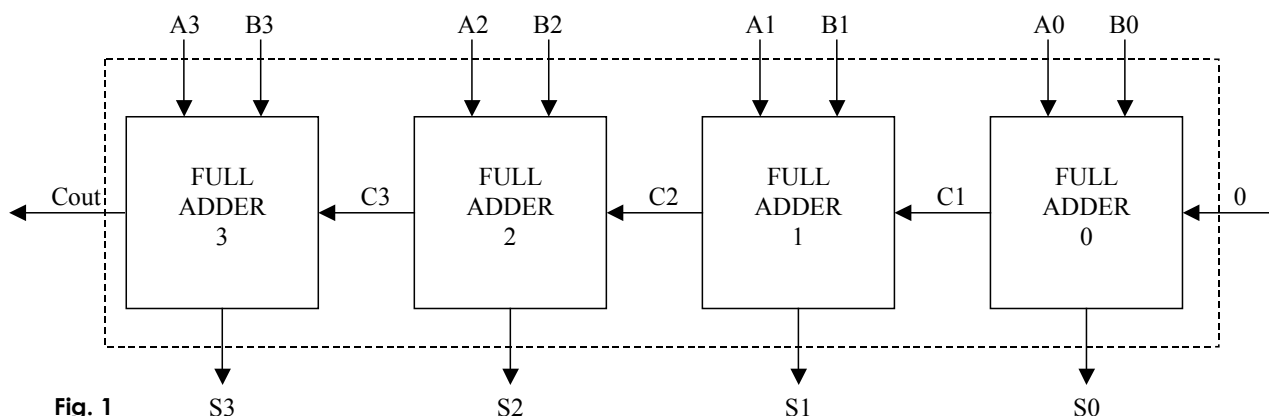


Fig.2 mostra lo schema di un Carry-Save Adder con due addendi di quattro bit. PS e PR non sono somma e riporto dei due bit, bensì soltanto Pseudo-Somma e Pseudo-Riporto. Per ottenere la somma complessiva, bisogna moltiplicare per 2 lo Pseudo-Riporto (shift) e poi effettuare la somma PS + 2PR con un Parallel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

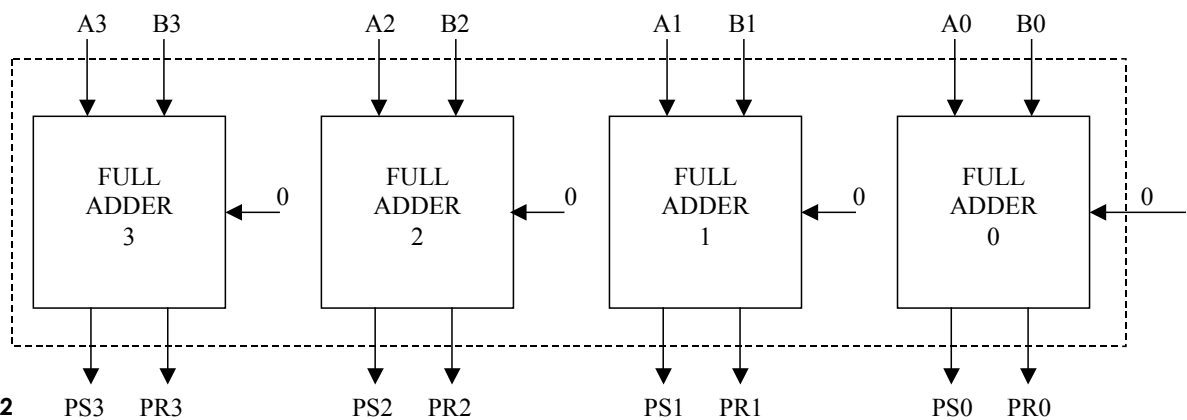


Fig. 2

## ESERCIZIO 2 (solo NO: 9 punti)

Scrivere una funzione Assembler MIPS che, ricevendo in ingresso un vettore  $v$  di dimensione  $N$  e un valore  $t$ , produca in uscita la somma degli elementi di  $v$  che risultano strettamente maggiori di  $t$ .

Si assuma che  $\&v[0] \rightarrow \$4, N \rightarrow \$5, t \rightarrow \$6$ . L'uscita dev'essere posta in  $\$7$ .

Suggerimento: il codice Assembler MIPS potrebbe implementare la seguente funzione C:

```
int somma(int *v, int N, int t)
{
    int i, sum;

    sum=0;
    for(i=0; i<N; i++)
        if(v[i]>t)
            sum += v[i];

    return sum;
}
```

## Soluzione.

```
Somma:    addi $29, $29, -12 #riservo spazio per il salvataggio registri
          sw $8, 0($29)      #conterrà il contatore i
          sw $9, 4($29)      #conterrà v[i]
          sw $10, 8($29)     #conterrà il valore di verità (v[i]>t)
          move $7, $0        #sum=0
          move $8, $0        #i=0
For:       beq $8, $5, Exit   #finché i<N
          lw $9, 0($4)        #carica v[i]
          slt $10, $6, $9     #(t<v[i]) → $10
          beq $10, $0, Continue
          add $7, $7, $9      #sum += v[i]
Continue: addi $4, $4, 4      #++v
          addi $8, $8, 1      #i++
          j For
Exit:     lw $8, 0($29)       #ripristina il contesto
          lw $9, 4($29)
          lw $10, 8($29)
          addi $29, $29, 12
          jr $31
```

**ESERCIZIO 2 (solo VO: 5 punti)**

Si consideri una memoria a disco con le seguenti caratteristiche: velocità di rotazione 5400 giri/min, tempo medio di posizionamento pari a 10 msec, settori da 512 byte, 120 settori per traccia.

Calcolare il tempo necessario per leggere un file di 20 kbyte, ipotizzando che sia stato registrato sul disco in blocchi da 2 kbyte, e che ciascun blocco si trova in settori consecutivi di una stessa traccia.

**Soluzione**

$$T_{rot} = 60/V_{rot} = 11,11 \text{ ms}$$

$$T_{lat} = T_{rot}/2 = 5,56 \text{ ms}$$

$$T_{lett} \text{ 1 settore} = T_{lett} \text{ traccia} / (\text{n.ro settori}) = T_{rot}/120 = 92,6 \mu\text{s}$$

$$1 \text{ blocco} = 4 \text{ settori} (2 \text{ kbyte}/512 \text{ byte})$$

$$T_{lett} \text{ 1 blocco} = 4 \cdot T_{lett} \text{ 1 settore} = 0,37 \text{ ms}$$

$$T_{lett} \text{ file} = 10 \cdot (10 \text{ ms} + 5,56 \text{ ms} + 0,37 \text{ ms}) = 159 \text{ ms}$$

**ESERCIZIO 3 (8 punti)**

Sono dati i seguenti formati per la rappresentazione dei numeri in un calcolatore (campo complessivo 28 bit):

- interi senza segno
- reali in virgola fissa con bit di segno, parte intera e sei bit per la parte frazionaria
- reali in virgola mobile con bit di segno, esponente in eccesso 63 (7 bit) e mantissa frazionaria e normalizzata in segno e valore (1.M)

Si richiede:

- 1) (NO: 4 punti – VO: 3 punti) il minimo e massimo valore raggiungibili escluso lo zero e in valore assoluto
- 2) (NO: 4 punti – VO: 3 punti) rappresentare in virgola mobile i numeri 121.25 e 32.75 e sommarli secondo l'algoritmo usato nei calcolatori.
- 3) (solo VO: 2 punti) Spiegare in modo chiaro e sintetico i componenti di una Floating Point Unit di una ALU.

**Soluzione.**

1)

Interi senza segno: min: 1 ; max  $2^{28}-1$ .

Virgola fissa: min  $2^{-6}$  ; max  $2^{21}-2^{-6}$

Virgola mobile: min  $2^{-63}$  ; max  $(2-2^{-20}) \cdot 2^{64}$

2)

$$121.25 = 1.11100101 \cdot 2^6$$

$$32.75 = 1.0000011 \cdot 2^5$$

6 in eccesso 63: 1000101

5 in eccesso 63: 1000100

$$121.25 \quad 1 \quad 1000101 \quad 11100101000000000000$$

$$32.75 \quad 1 \quad 1000100 \quad 00000110000000000000$$

Per sommare i due numeri, bisogna innanzi tutto uniformare gli esponenti, poi sommare le mantisse:

$$\begin{array}{r} 1.11100101 + \\ 0.10000011 = \\ \hline \end{array}$$

10.01101000

E' necessario normalizzare il risultato, che in virgola mobile si rappresenta:

154.00      1    1000110    00110100000000000000

3) V. dispense del corso

**ESERCIZIO 4** (NO: 8 punti – VO: 7 punti)

a) (2 punti) Spiegare in modo chiaro e sintetico le differenze tra memoria paginata e memoria segmentata.

b) (NO: 6 punti – VO: 5 punti) Si consideri un sistema operativo con gestione della memoria segmentata e paginata. Ciascun job viene suddiviso al massimo in 3 segmenti. Ciascun segmento ha una dimensione massima di 128 pagine. Le pagine hanno dimensione 1KB. Si abbia ad esempio la seguente situazione:

| Tabella dei Segmenti |                      |
|----------------------|----------------------|
| Segmento             | Tabella delle Pagine |
| 00                   | 2                    |
| 01                   | 0                    |
| 10                   | 1                    |

| Tabella delle Pagine 0 |               |                 |              |
|------------------------|---------------|-----------------|--------------|
| Pagina Virtuale        | Pagina Fisica | Bit di Validità | Solo lettura |
| 0000000                | 0001010       | 1               | 0            |
| 0000001                | 0010001       | 0               | 0            |
| 0000010                | 1011001       | 1               | 0            |
| 0000011                | 1011010       | 1               | 0            |
| 0000100                | 0011101       | 1               | 0            |
| 0000101                | 0101111       | 0               | 0            |
| 0000110                | 0110111       | 1               | 0            |
| 0000111                | 0100000       | 1               | 0            |
| 0001000                | 0100100       | 1               | 0            |
| 0001001                | 0001001       | 1               | 0            |

| Tabella delle Pagine 1 |               |                 |              |
|------------------------|---------------|-----------------|--------------|
| Pagina Virtuale        | Pagina Fisica | Bit di Validità | Solo lettura |
| 0000000                | 0000011       | 0               | 0            |
| 0000001                | 0010110       | 0               | 0            |
| 0000010                | 1001001       | 1               | 0            |
| 0000011                | 1001010       | 1               | 0            |
| 0000100                | 1010101       | 1               | 0            |
| 0000101                | 0011101       | 1               | 0            |
| 0000110                | 0111111       | 1               | 0            |
| 0000111                | 1011101       | 1               | 0            |
| 0001000                | 1010011       | 1               | 0            |
| 0001001                | 0001111       | 1               | 0            |
| 0001010                | 0011011       | 1               | 0            |
| 0001011                | 0100010       | 1               | 0            |

| Tabella delle Pagine 2 |               |                 |              |
|------------------------|---------------|-----------------|--------------|
| Pagina Virtuale        | Pagina Fisica | Bit di Validità | Solo lettura |
| 0000000                | 0100001       | 1               | 1            |
| 0000001                | 0101110       | 1               | 1            |
| 0000010                | 0110110       | 0               | 1            |
| 0000011                | 0000110       | 1               | 1            |
| 0000100                | 1100011       | 0               | 1            |
| 0000101                | 1000011       | 1               | 1            |
| 0000110                | 0010101       | 1               | 1            |

Il bit di validità indica se la pagina virtuale richiesta è presente (1) o no (0) nella memoria principale. Si considerino le seguenti richieste e mostrare: a quale indirizzo fisico corrispondono gli indirizzi virtuali, se l'operazione può essere conclusa con successo o se viene generato un errore (segmento non valido, pagina non valida, violazione di protezione) o un page fault:

1) Read 0000000010101001101

2) Write 1000001000101011001

1. Read 0100001011001110011

2. Write 0000000111100100110
  3. Read 0100010000001100001
  4. Write 1110000100000100111
- 

**Soluzione:**

- a) V. dispense del corso.
- b)
- 1) Indirizzo fisico valido 01011100101001101
- 2) Indirizzo fisico valido 10101010101011001
- 3) Indirizzo fisico non valido (pagina non valida)
- 4) Violazione di scrittura sull'indirizzo fisico 00001101100100110
- 5) Indirizzo fisico valido 01001000001100001
- 6) Segmento non valido

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

Si consideri un calcolatore multiprocessore con  $p$  processori. Il tempo di esecuzione di una certa applicazione  $M$  su tale calcolatore sia pari a  $90/p$  msec, e il tempo necessario per le comunicazioni fra processori sia pari a  $10p$  msec ( $p > 1$ ).

1. (1 punto) Calcolare il tempo necessario per eseguire l'applicazione in un'architettura monoprocessore.
2. (4 punti) Calcolare il numero di processori per ottenere il tempo minimo di esecuzione. Calcolare il tempo minimo di esecuzione.

Soluzione.

- 1) Poiché c'è un solo processore,  $p = 1$ , quindi il tempo è dato da 90 msec.
- 2) Il tempo complessivo è dato da:  $40/p + 10p$ . Derivando rispetto a  $p$  ed eguagliando a zero:

$$\frac{dT}{dp} = -\frac{90}{p^2} + 10 = 0 \Rightarrow p = 3$$

Il tempo minimo si ottiene per  $p=3$ :  $90/3 + 10 \cdot 3 = 30 + 30 = 60$  msec.