

SOLUZIONI DELLA PROVA SCRITTA DEL CORSO DI
CALCOLATORI ELETTRONICI
NUOVO ORDINAMENTO DIDATTICO
11 Settembre 2003

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

ESERCIZIO 1 (9 punti)

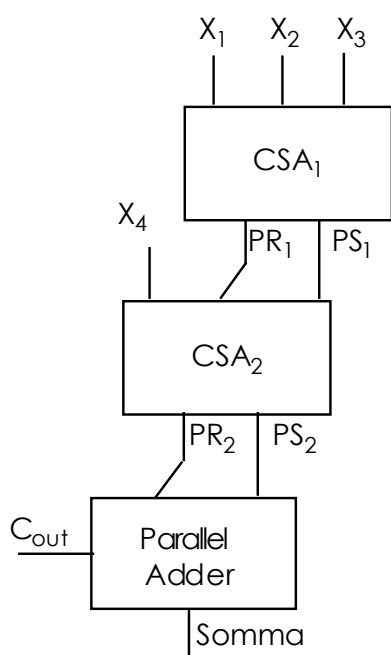
- 1) (4 punti) Spiegare in modo chiaro e sintetico le caratteristiche di un Carry-Save Adder, e in quale caso, e perché, presenta vantaggi rispetto al Parallel Adder.
- 2) (5 punti) Si vogliono sommare i seguenti quattro numeri a 8 bit: $X_1 = 01101010$, $X_2 = 10100100$, $X_3 = 00101010$, $X_4 = 00010010$. Disegnare lo schema che permette di eseguire tale somma usando due addizionatori del tipo "Carry Save Adder" ed un "Parallel Adder" finale. Gli addizionatori "Carry Save Adder" lavorano su tre operandi. Precisare il valore assunto dalla Pseudosomma e dal Pseudoriporto all'uscita del primo e del secondo "Carry Save Adder", e le operazioni eseguite per ottenere la somma finale. Calcolare il tempo di ritardo di tale sommatore esprimendolo come multiplo del tempo di ritardo d introdotto da una porta logica.

Soluzione

1) Vedi dispense del corso.

2)

Schema:



L'addizionatore "Carry Save Adder" fornisce in uscita due parole a 8 bit: la Pseudosomma e il Pseudoriporto. La Pseudosomma (PS) rappresenta il risultato della somma dei tre addendi senza considerare i riporti, mentre il Pseudoriporto (PR) rappresenta solo i riporti relativi a ciascuno stadio. Il risultato completo lo si ottiene sommando $PS + 2 PR$. Un possibile schema per eseguire la somma richiesta è riportato in figura, dove con un tratto obliquo si è indicato lo "scorrimento" verso sinistra (moltiplicazione per 2) necessario per sommare il PR al PS.

Somma di $X_1 + X_2 + X_3$:

$$PS_1 = 11100100, PR_1 = 00101010; 2 PR_1 = 01010100$$

Somma di $X_4 + PS_1 + 2 PR_1$:

$$PS_2 = 10100010, PR_2 = 01010100; 2 PR_2 = 10101000$$

$$\text{Somma finale} = PS_2 + 2 PR_2 = 01001010, C_{out} = 1$$

Vantaggio carry save adder rispetto ad addizionatore parallelo. Per effettuare la somma di 4 addendi usando solo parallel adder sarebbero necessari 3 parallel adder, dunque il vantaggio non è nel numero di componenti. Il vantaggio risiede nella maggior velocità dell'operazione: gli addizionatori del tipo carry save non hanno i ritardi dovuti alla propagazione del riporto, dunque il risultato si ottiene dopo un ritardo corrispondente a una rete a due o tre livelli, a seconda della rete usata per calcolare somma e riporto. L'unico ritardo dovuto alla propagazione del riporto è presente solo nell'ultimo stadio. Se indichiamo con d il ritardo per il calcolo di somma e riporto per un full-adder, un parallel adder a 8 bit produce il risultato con un ritardo pari a $8d$, mentre un carry save adder

produce il risultato in termini di PS e PR con un ritardo pari a d . Nel caso in esame (somma di 4 addendi con 2 addizionatori carry-save e un addizionatore parallelo) avremo un ritardo pari a $d + d + 8d = 10d$, mentre usando 3 parallel adder il ritardo nella produzione del risultato sarebbe pari a $3 \cdot 8d = 24d$

ESERCIZIO 2 (8 punti)

Il seguente codice Assembler MIPS è stato scritto per eseguire il prodotto scalare di due vettori v e w . Gli indirizzi iniziali $\&v[0]$ e $\&w[0]$ sono nei registri $\$4$ e $\$5$, la dimensione dei vettori n in $\$6$ e il risultato finale deve essere memorizzato in $\$7$. Tuttavia il programmatore ha compiuto alcuni errori. Si richiede di individuare e correggere tali errori, facendo sì che il codice funzioni correttamente.

```
Scalare:  add $29,$29, -12
          sw $29, 0($8)
          sw $29, 4($9)
          sw $29, 8($10)
          move $7, $0
          move $8, $0
For:      beq $8, $6, Exit
          sw $9, 0($4)
          sw $10, 0($5)
          mul $9, $9, $10
          add $7, $7, $10
          addi $4, $4, 1
          addi $5, $5, 1
          addi $8, $8, 4
          jr For
Exit:     lw $8, 0($29)
          lw $9, 0($29)
          lw $10, 0($29)
          j $31
```

Soluzione.

Codice originale

```
Scalare:  add $29,$29, -12
          sw $29, 0($8)
          sw $29, 4($9)
          sw $29, 8($10)
          move $7, $0
          move $8, $0
For:      beq $8, $6, Exit
          sw $9, 0($4)
          sw $10, 0($5)
          mul $9, $9, $10
          add $7, $7, $10
          addi $4, $4, 1
          addi $5, $5, 1
          addi $8, $8, 4
          jr For
Exit:     lw $8, 0($29)
          lw $9, 0($29)
          lw $10, 0($29)
          j $31
```

Codice corretto

```
#addi $29,$29, -12
#sw $8, 0($29)
#sw $9, 4($29)
#sw $10, 8($29)
move $7, $0
move $8, $0
For:      beq $8, $6, Exit
          lw $9, 0($4)
          lw $10, 0($5)
          mul $9, $9, $10
          add $7, $7, $9
          addi $4, $4, 4
          addi $5, $5, 4
          addi $8, $8, 1
          jr For
Exit:     lw $8, 4($29)
          lw $10, 8($29)
          #addi $29,$29, 12
          #jr $31
```

ESERCIZIO 3 (8 punti)

- 1) (4 punti) Spiegare in modo chiaro e sintetico le caratteristiche della memoria cache. Indicare quali sono i principali metodi di indirizzamento e spiegarne in dettaglio il funzionamento, specificando come vengono gestiti i campi di un indirizzo di memoria primaria per l'allocazione del blocco in cache.
- 2) (4 punti) Si supponga di avere una memoria cache di sedici parole. Si indichi lo stato finale della cache, inizialmente vuota, date le parole di indirizzo (decimale): 48, 13, 46, secondo le seguenti modalità di indirizzamento:
 - completamente associativo, con blocchi di quattro parole;
 - diretto con blocchi di due parole;
 - associativo su insiemi a due vie, e blocchi di due parole.

Soluzione.

- 1) Vedi dispense del corso (Capitolo 4)

2)

Metodo completamente associativo

Blocco 0	Blocco 1	Blocco 2	Blocco 3
48	12	44	
49	13	45	
50	14	46	
51	15	47	

Metodo diretto

Blocco 0	Blocco 2	Blocco 4	Blocco 6
48			12
49			13
Blocco 1	Blocco 3	Blocco 5	Blocco 7
			46
			47

Associativo su insiemi a due vie

Insieme 0	Insieme 1	Insieme 2	Insieme 3
Blocco 0	Blocco 0	Blocco 0	Blocco 0
48		12	46
49		13	47
Blocco 1	Blocco 1	Blocco 1	Blocco 1

ESERCIZIO 4 (8 punti)

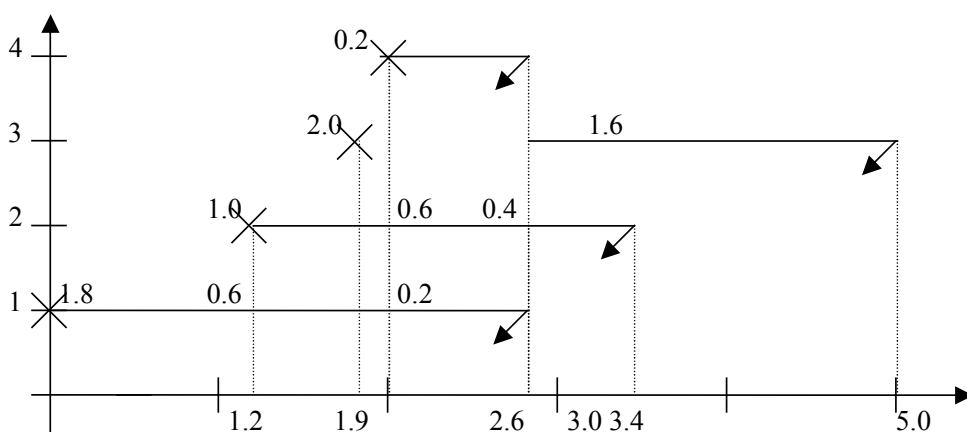
Si consideri la seguente lista di processi. Il Sistema Operativo usa uno scheduling di tipo round robin multiprogrammato, con strategia di allocazione FIFO. Inoltre gestisce una memoria di tipo paginato.

Processo	Arrivo	CPU	#pagine
1	0.0	1.8	4
2	1.2	1.0	3
3	1.9	2.0	5
4	2.0	0.2	2

Considerando una memoria complessiva di 9 pagine:

- (5 punti) Mostrare, utilizzando il metodo grafico, la sequenza di esecuzione dei processi.
- (3 punti) Calcolare il tempo di *turnaround* medio e il tempo di *turnaround* pesato medio.

Soluzione.



Processo	Arrivo	Fine	Turnaround	WT
1	0.0	2.6	2.6	1.4
2	1.2	3.4	2.2	2.2
3	1.9	5.0	3.1	1.6
4	2.0	2.6	0.6	3.0
Media			2.1	2.1