

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

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

**ESERCIZIO 1 (NO: 7 punti – VO: 6 punti)**

Progettare una ALU che realizzi le seguenti funzionalità (l'apice indica la negazione):

S0	S1	Funzione
0	0	A
0	1	B
1	0	A+B
1	1	A-B

A e B sono i due operandi a N bit.

- (VO: 4 punti) Realizzare la ALU utilizzando un parallel adder e un'opportuna rete logica, illustrando graficamente lo schema progettuale.
- (solo VO: 2 punti) Realizzare la ALU utilizzando un MUX 4-1 in luogo delle reti logiche del punto precedente.

**Soluzione**

- a) L'esercizio si risolve semplicemente implementando le reti logiche che elaborano ogni bit degli operandi prima di passarli al parallel adder. Tali reti logiche si desumono per ispezione visiva della tabella di verità nel testo dell'esercizio:

		$S_0 S_1$			
$A_i$		00	01	11	10
	0				
1		1		1	1

$$A_i^{new} = (S_0 + \overline{S_1}) \cdot A_i$$

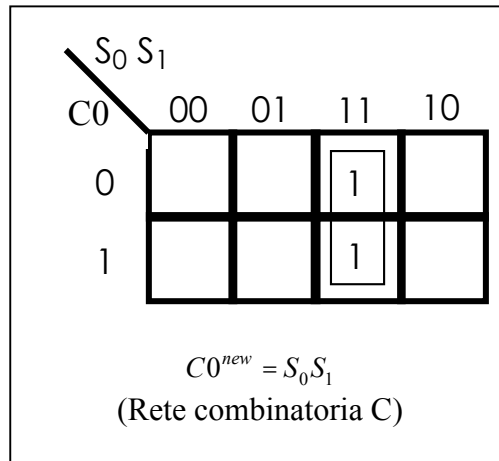
(Rete combinatoria A)

		$S_0 S_1$			
$B_i$		00	01	11	10
	0			1	
1			1		1

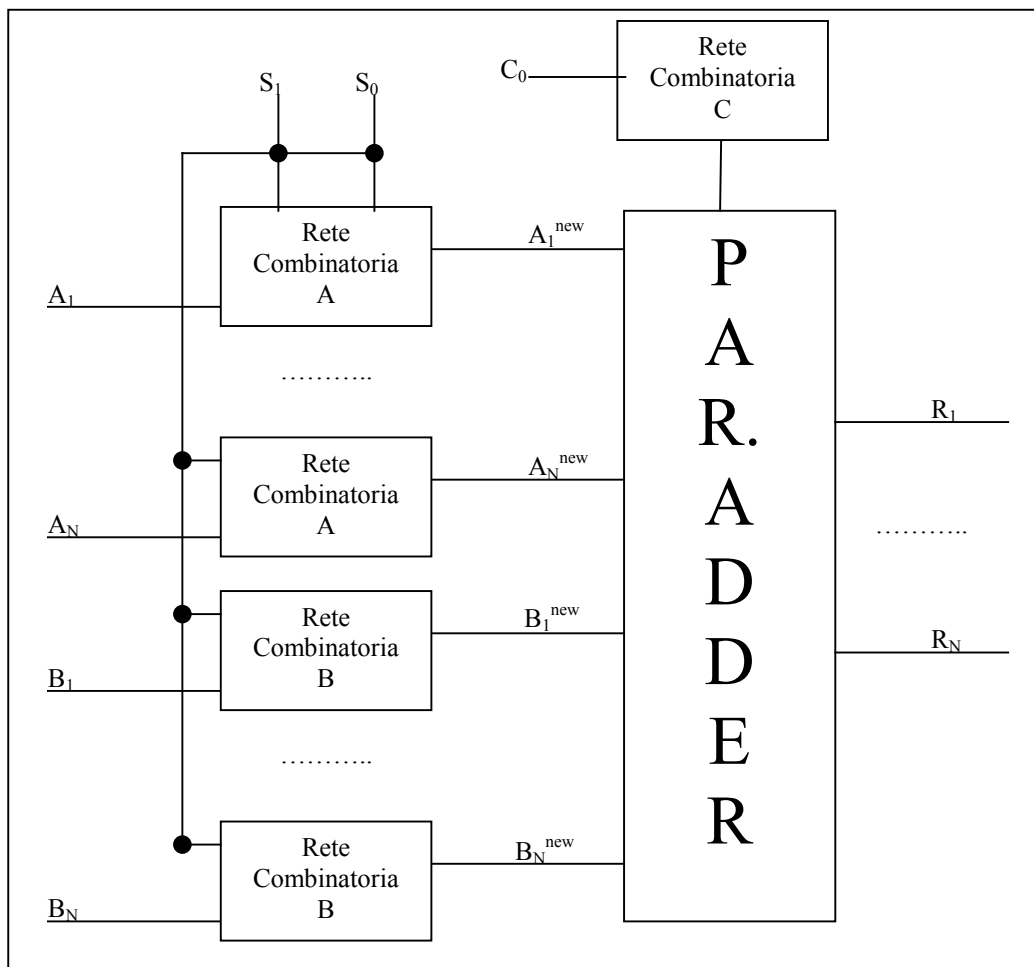
$$B_i^{new} = S_0 S_1 \overline{B_i} + \overline{S_0} S_1 B_i + S_0 \overline{S_1} B_i$$

(Rete combinatoria B)

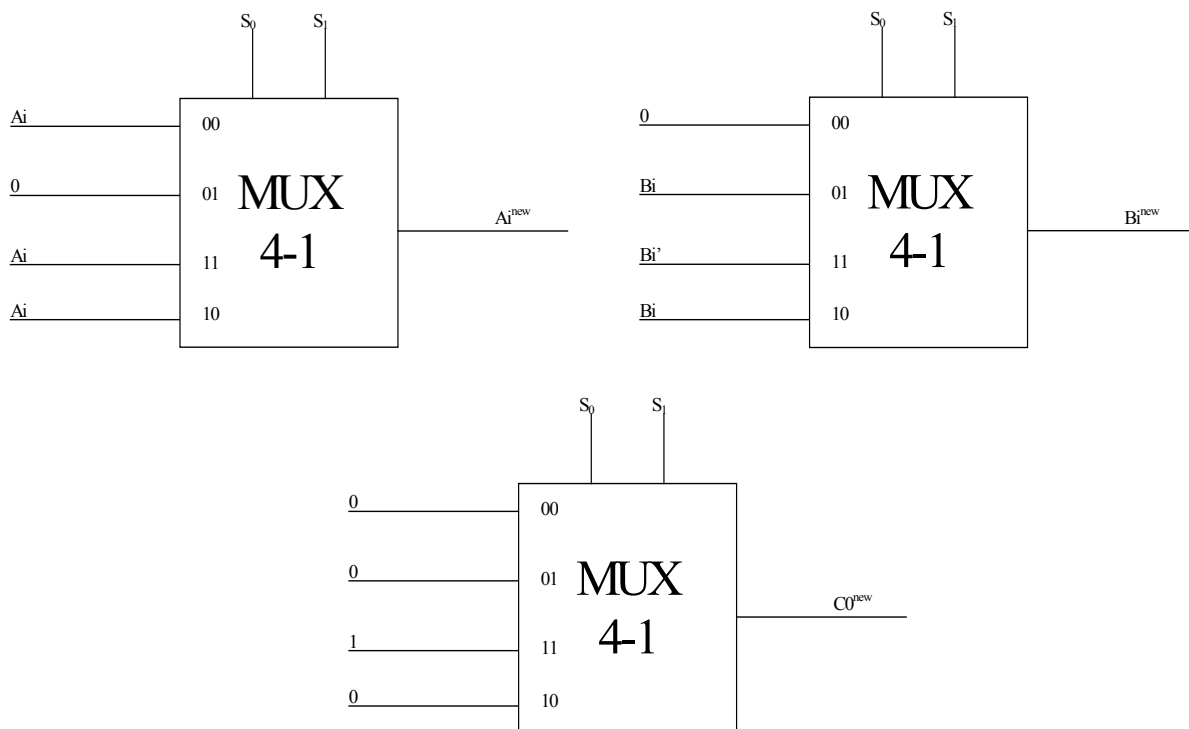
LA rete logica per il riporto iniziale è:



Lo schema progettuale è il seguente:



b) I tre MUX 4-1 devono essere così pilotati:



**ESERCIZIO 2 (NO: 9 punti – VO: 8 punti)**

Si consideri una memoria cache costituita da quattro blocchi di quattro parole. L'indirizzamento è gestito mediante metodo completamente associativo.

1. (NO: 7 punti – VO: 6 punti) Assumendo la cache inizialmente vuota, si indichi lo stato finale della cache e si calcoli l'hit ratio in seguito alle seguenti richieste di parole (indirizzi in decimale): 32, 4, 30, 15, 16, 14, 31, 32, 15, 16, 10, 12, 13, 4, 6 (è sufficiente indicare il block frame). La strategia di rimpiazzamento seguita è la LRU.
2. (2 punti) Sapendo che il tempo di accesso in cache è 2 ns e il tempo di accesso in primaria è 45 ns, si calcoli il tempo medio di accesso alla gerarchia usando l'hit ratio calcolato al punto precedente.

Soluzione.

Si noti che in questo caso block frame == tag. Per ottenerlo basta effettuare la seguente divisione: IndirizzoParola / DimBlocco. In tal caso DimBlocco=4.

Nel seguente schema la prima colonna presenta i quattro blocchi di cache da 0 a 3, e l'ultima colonna è lo stato finale della stessa con strategia LRU. Abbiamo indicato, lungo le chiamate, il block frame di primaria che va ad occupare il corrispondente blocco di cache.

	32	4	30	15	16	14	31	32	15	16	10	12	13	4	6	
<b>0</b>	8				4					4						<b>4</b>
<b>1</b>		1						8						1	1	<b>1</b>
<b>2</b>			7				7				2					<b>2</b>
<b>3</b>				3		3			3			3	3			<b>3</b>
						H	H		H	H		H	H		H	

Abbiamo anche tenuto conto degli hit di cache. L'hit ratio è dunque pari a 7/15. Per calcolare il tempo medio di accesso alla gerarchia basta considerare la formula:

$$\bar{T} = H_c T_c + (1 - H_c)(T_c + T_p) = T_c + (1 - H_c)T_p = 2 + \frac{8}{15}45 = 1 + 24 = 25ns$$

### ESERCIZIO 3 (solo NO: 9 punti)

Scrivere una funzione Assembler MIPS che, ricevendo in ingresso due vettori di interi  $v$  e  $w$  di pari lunghezza  $N$ , restituisca il valore  $f$  dato dalla seguente formula:

$$f = \frac{1}{N} \sum_{i=1}^N (v_i - w_i)^2$$

Si assuma che  $\&v[0] \rightarrow \$4$ ,  $\&w[0] \rightarrow \$5$ ,  $N \rightarrow \$2$ , valore di uscita  $\rightarrow \$6$ .

Poiché il MIPS non è dotato di un'esplicita operazione di divisione, si ipotizzi di disporre di una funzione  $\text{div}(x, y)$  che immette il risultato della divisione  $x/y$  in  $\$6$ , con  $x \rightarrow \$4$  e  $y \rightarrow \$5$  (passaggio dei parametri).

Per esempio il codice MIPS potrebbe implementare il seguente codice C:

```
int sumsquare(int v[], int w[])
{
    int i, ssq;

    ssq=0;
    for(i=0; i<N; i++)
        ssq += (v[i]-w[i])*(v[i]-w[i]);

    ssq = div(ssq, N);
    return ssq ;
}
```

### Soluzione.

Utilizzeremo i seguenti registri:  $i \rightarrow \$8$ ,  $v[i] \rightarrow \$9$ ,  $w[i] \rightarrow \$10$ .

```
sumsquare:  addi $29, $29, -12    #salvataggio contesto
            sw $8, 0($29)
            sw $9, 4($29)
            sw $10, 8($29)
            move $6, $0          #ssq=0
            move $8, $0          #i=0
for:        beq $8, $2, exit      #if(i==N) exit
            lw $9, 0($4)         #carico v[i]
            lw $10, 0($5)        #carico w[i]
            sub $9, $9, $10      #$9 contiene i parziali
            mul $9, $9, $9
            add $6, $6, $9       #aggiorno ssq
            addi $8, $8, 1       #i++
            addi $4, $4, 4       #prossimo elemento di v
            addi $5, $5, 4       #prossimo elemento di w
            j for
exit:       move $27, $4         #sposto cautelativamente
            move $28, $5         #i parametri di sumsquare
            move $4, $6          #passo i parametri a div
            move $5, $2
            jal div
            move $4, $27         #ripristino i parametri
            move $5, $28         #del chiamante
            lw $8, 0($29)        #ripristino del contesto
            lw $9, 4($29)
            lw $10, 8($29)
            addi $29, $29, 12
            jr $31              #ritorno al prog.principale
```

**ESERCIZIO 3 (solo VO: 6 punti)**

Si supponga di disporre di tre macchine: a pila, a uno e a due indirizzi. Per ognuna di queste si abbiano le seguenti istruzioni:

A pila		A un indirizzo		A due indirizzi	
Istruzione	Semantica	Istruzione	Semantica	Istruzione	Semantica
PUSH X	$M[X] \rightarrow \text{push}$	STORE X	$\text{ACC} \rightarrow M[X]$	MOV X1,X2	$M[X1] \rightarrow M[X2]$
POP X	$\text{pop} \rightarrow M[X]$	LOAD X	$M[X] \rightarrow \text{ACC}$	ADD X1,X2	$M[X1] + M[X2] \rightarrow M[X2]$
ADD	$\text{pop} + \text{pop} \rightarrow \text{push}$	ADD X	$\text{ACC} + M[X] \rightarrow \text{ACC}$	DIV X1,X2	$M[X1] / M[X2] \rightarrow M[X2]$
DIV	$\text{pop}(2) / \text{pop}(1) \rightarrow \text{push}$	DIV X	$\text{ACC} / M[X] \rightarrow \text{ACC}$	SUB X1,X2	$M[X1] - M[X2] \rightarrow M[X2]$
SUB	$\text{pop}(2) - \text{pop}(1) \rightarrow \text{push}$	SUB X	$\text{ACC} - M[X] \rightarrow \text{ACC}$		

ACC è il registro accumulatore;  $M[X]$  indica il dato nella locazione di memoria X.

- (4 punti) Facendo attenzione a non sovrascrivere i contenuti iniziali della memoria, si scriva, per ognuna delle tre macchine, la sequenza delle istruzioni necessarie per realizzare la seguente operazione:

$$Z = A / (B - C) + D$$

(suggerimento: si usi uno o più registri dove depositare i risultati parziali)

- (2 punti) Spiegare in modo chiaro e sintetico i vari tipi di indirizzamento di un'istruzione.

**Soluzione.**

A zero Indirizzi	A un indirizzo	A due indirizzi
PUSH D	LOAD B	MOV C,Z
PUSH C	SUB C	SUB B,Z
PUSH B	STORE Z	DIV A,Z
SUB	LOAD A	ADD A,Z
PUSH A	DIV Z	
DIV	ADD D	
ADD	STORE Z	
POP Z		

- Vedi dispense del corso.

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

Sia data la seguente lista di processi:

Processo	Tempo di arrivo	Tempo di CPU	Memoria
1	0.0	1.6	120K
2	1.0	1.0	50K
3	1.8	2.0	300K

Testo NO:

La memoria è gestita con partizioni statiche nell'ordine di 100K, 500K e 150K con strategia di allocazione First Fit.

1. (NO: 6 punti – VO: 5 punti) Mostrare, utilizzando il metodo grafico, la sequenza di esecuzione dei processi qualora si impieghi la politica di scheduling FIFO multiprogrammata, indicando in quale partizione i processi vengono allocati (memoria inizialmente vuota).
2. (2 punti) Calcolare il tempo di *turnaround* medio e il tempo di *turnaround* pesato medio.

Testo VO:

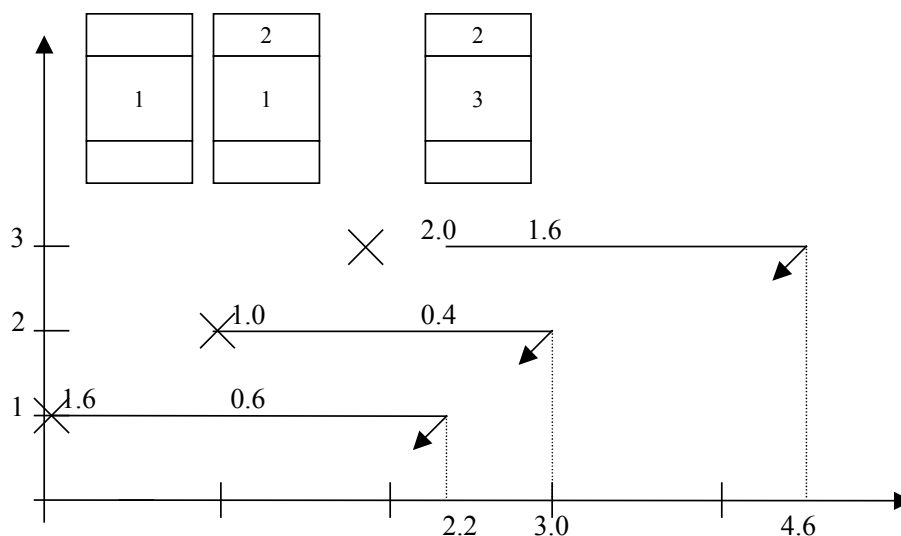
La memoria, di 300K, è gestita con partizioni dinamiche rilocabili.

1. (5 punti) Mostrare, utilizzando il metodo grafico, la sequenza di esecuzione dei processi qualora si impieghi la politica di scheduling FIFO multiprogrammata, indicando quali operazioni sono necessarie per l'allocazione dei processi (memoria inizialmente vuota).
2. (2 punti) Calcolare il tempo di *turnaround* medio e il tempo di *turnaround* pesato medio.

### Soluzione.

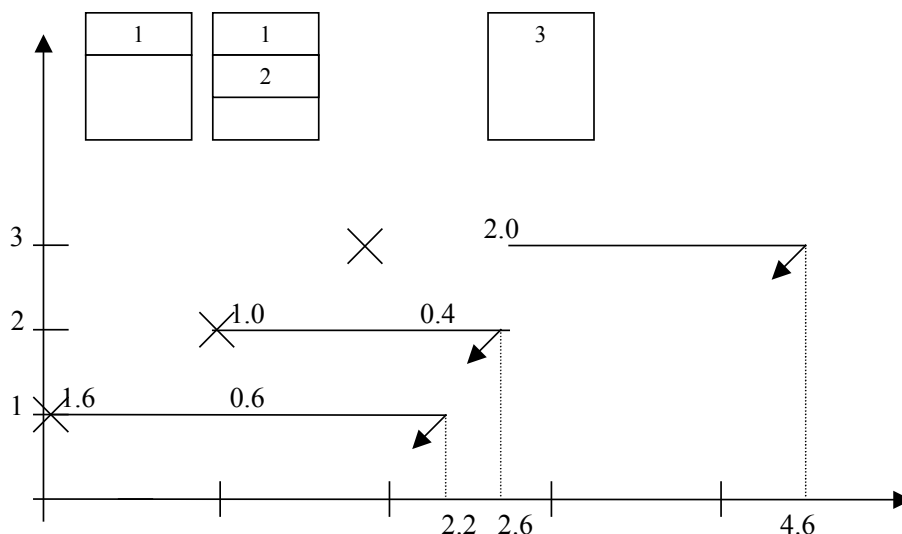
NO:

Grafico processo-tempo con politica FIFO multiprogrammata e allocazione First Fit:



Processo	Arrivo	Fine	Turnaround	WT
1	0.0	2.2	2.2	1.4
2	1.0	3.0	2.0	2.0
3	1.8	4.6	2.8	1.4
Media			2.3	1.6

VO:



Il processo 3 non viene allocato immediatamente, ma solo dopo che i processi 1 e 2 sono terminati. Dopo la terminazione, i processi vengono deallocati e il sistema operativo effettua un'operazione di compattamento delle partizioni e ne aggiorna la tabella (rilocalizzazione) onde eliminare la frammentazione e consentire l'allocazione del processo 3.

Processo	Arrivo	Fine	Turnaround	WT
1	0.0	2.2	2.2	1.4
2	1.0	2.6	1.6	1.6
3	1.8	4.6	2.8	1.4
<b>Media</b>			2.2	1.5

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

Spiegare in modo chiaro e sintetico la classificazione "strutturale" delle architetture parallele a seconda che si faccia riferimento:

- 1) (3 punti) all'organizzazione della memoria;
- 2) (2 punti) alla rete di interconnessione.

### Soluzione.

Vedere le dispense del corso.