

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

11 Febbraio 2008

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

**ESERCIZIO 1 (NO: 7 punti – VO: 5 punti)**

- 1) (NO: 4 punti – VO: 3 punti) Progettare una rete logica che calcoli il complemento a 2 di un operando a tre bit.
- 2) (NO: 3 punti- VO: 2 punti) Indicare a quale tipo di reti logiche appartiene la rete progettata e spiegarne il motivo.

**Soluzione.**

Tabella di verità della rete in oggetto :

Ingressi			Uscite		
A	B	C	A'	B'	C'
0	0	0	0	0	0
0	0	1	1	1	1
0	1	0	1	1	0
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	0	1	1
1	1	0	0	1	0
1	1	1	0	0	1

AB \ C	00	01	11	10
0		1		1
1	1	1		

AB \ X	00	01	11	10
0		1	1	
1	1			1

$$A' = \bar{A}B + \bar{A}C + A\bar{B} \cdot \bar{C}$$

$$B' = B\bar{C} + \bar{B}C$$

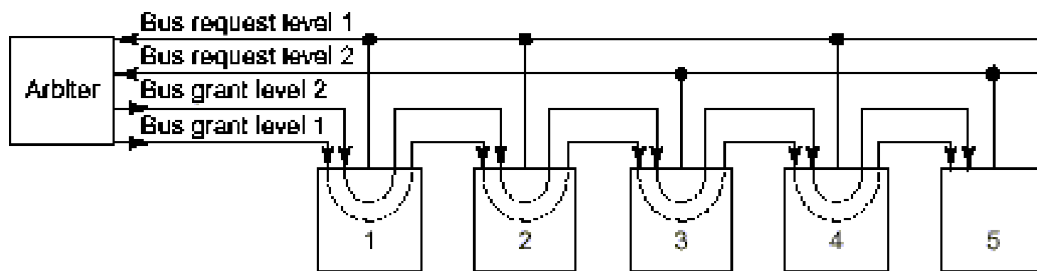
AB \ C	00	01	11	10
0				
1	1	1	1	1

$$C' = C$$

Si tratta ovviamente di una rete logica combinatoria, in quanto le uscite dipendono soltanto dai valori degli ingressi ad un dato istante.

## ESERCIZIO 2 (NO: 8 punti - VO: 6 punti)

Si consideri lo schema di arbitraggio del bus rappresentato in figura. Le linee di "bus request" e "bus grant" a 2 livelli vengono impiegate per gestire una diversa priorità delle periferiche.



- 1) (NO: 4 punti – VO: 3 punti) A quali schemi, fra quelli studiati nel corso, si ispira quello mostrato in figura? Spiegare come può essere gestita in questo caso la gerarchia di priorità fra le periferiche.
- 2) (NO: 4 punti – VO: 3 punti) Illustrare la sequenza di segnali di controllo che vengono scambiati fra periferica e arbitro del bus prima di iniziare un trasferimento di dati attraverso il bus.

### Soluzione

- 1) Lo schema in figura è un ibrido fra la tecnica di arbitraggio a richieste indipendenti e quella con collegamenti in catena "daisy chaining". Rispetto ai livelli 1 e 2 le periferiche sono divise in due gruppi indipendenti. All'interno di ciascun gruppo le periferiche sono collegate a catena. Pertanto vi è una gerarchia di priorità gestita direttamente dall'arbitro del bus che, in caso di richieste contemporanee da parte dei due livelli, può essere programmato per servire l'una o l'altra richiesta. All'interno di ciascun gruppo vi è un'ulteriore gerarchia di priorità "cablata", data dalla successione delle periferiche nella catena: ottiene il bus la prima periferica della catena, fra quelle che hanno fatto richiesta, che riceve il grant.
- 2) Una periferica che debba trasmettere dati sul bus, ne fa richiesta all'arbitro attraverso la linea di bus request. L'arbitro del bus risponde a questa richiesta inviando un segnale di bus grant sulla linea appropriata. Il bus grant attraversa serialmente le periferiche (nota: il grant relativo a un certo livello di priorità viene fatto passare da periferiche con livello di priorità diverso) e viene bloccato dalla prima periferica che aveva fatto richiesta del bus. A questo punto la periferica invia il segnale di bus busy, per indicare che il bus è impegnato e inizia il trasferimento dei dati.

### ESERCIZIO 3 (solo NO: 8 punti)

Si progetti una funzione assembler MIPS che, ricevendo in ingresso un vettore  $v$ , e due interi  $i$  e  $j$ , permuti  $v[i]$  con  $v[j]$  solo se il primo risulta minore del secondo. La funzione deve restituire il valore 1 se la sostituzione è avvenuta, 0 altrimenti.

Il passaggio dei parametri presenta le specifiche:  $\&v[0] \rightarrow \$4$ ,  $i \rightarrow \$5$ ,  $j \rightarrow \$6$ , valore in uscita  $\rightarrow \$7$ .

Ad esempio, il codice MIPS potrebbe implementare il seguente codice C:

```
int swap(int *v, int i, int j)
{
    int temp, comp;

    comp=v[i]<v[j];
    if(comp)
    {
        temp=v[i];
        v[i]=v[j];
        v[j]=temp;
    }

    return comp;
}
```

### Soluzione

$\$9 \rightarrow v[i]$ ,  $\$10 \rightarrow v[j]$

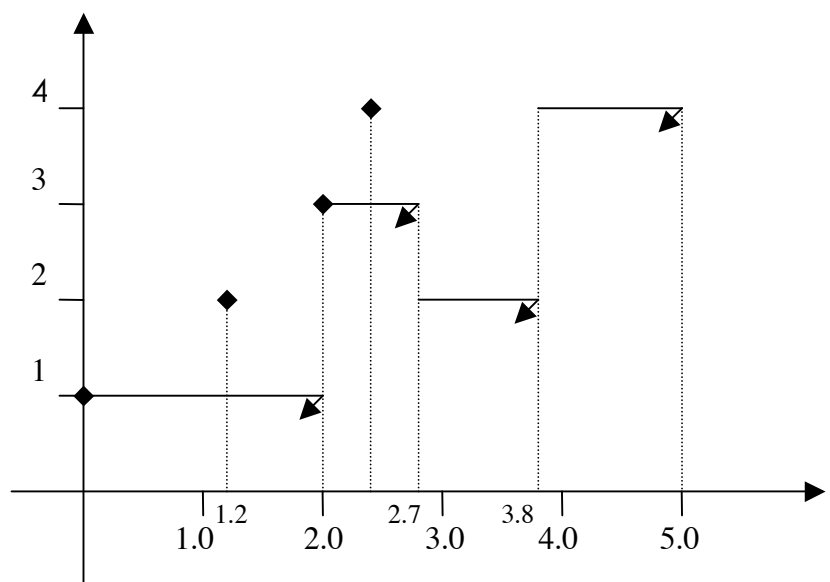
```
swap:
    addi $29, $29, -8 #salvataggio contesto
    sw $9, 0($29)
    sw $10, 4($29)
    muli $5, $5, 4      #calcolo indirizzo di v[i] e v[j]
    muli $6, $6, 4
    add $5, $5, $4
    add $6, $6, $4
    lw $9, 0($5)        #prelevo v[i] e v[j]
    lw $10, 0($6)
    slt $7, $9, $10     #comp=v[i]<v[j]
    beq $7, $0, exit    #se (!comp) salto a exit
    sw $9, 0($6)        #altrimenti permuto v[i] con v[j]
    sw $10, 0($5)
exit:
    lw $9, 0($29)       #ripristino del contesto
    lw $10, 4($29)
    addi $29, $29, 8
    jr $31              #ritorno al chiamante
```

**ESERCIZIO 3 (solo VO: 7 punti)**

Sia data la seguente lista di processi (si supponga che l'istante iniziale sia 0):

Job	Tempo di Arrivo	Tempo di CPU richiesto
1	0.0	2.0
2	1.2	1.1
3	2.0	0.7
4	2.5	1.2

Mostrare la sequenza di esecuzione dei job usando un grafico (tempo, job). Calcolare il tempo di *turnaround* medio e il tempo di *turnaround pesato* medio, qualora si impieghi la politica di scheduling SJF.

**Soluzione**

Job	Arrivo	Start	Finish	CPU time	Turnaround	W.turnaround
1	0.00	0.00	2.00	2.00	2.00	1.00
2	1.20	2.70	3.80	1.10	2.60	2.36
3	2.00	2.00	2.70	0.70	0.70	1.00
4	2.50	3.80	5.00	1.20	2.50	2.08
Average					1.95	1.61

**ESERCIZIO 4 (NO: 10 punti – VO: 8 punti)**

Si consideri una gerarchia di memoria a tre livelli costituita da: cache, primaria e disco.

- Il disco presenta le seguenti caratteristiche: 7200 giri/min, 120 settori per traccia, tempo medio di posizionamento 2 ms, 256 B per settore.
- L'indirizzamento di una parola della memoria primaria è a 24 bit.
- La memoria cache è di 1024 KB, è suddivisa in blocchi da 512 B (ovvero 512 parole), ed è indirizzata secondo il metodo diretto.

Si richiede:

- 1) (NO: 3 punti – VO: 2 punti) Il calcolo del tempo medio di lettura di un blocco di 512 B da disco, nell'ipotesi che la testina si trovi in un punto qualsiasi del disco all'istante iniziale e che il blocco sia registrato su settori contigui della stessa traccia.
- 2) (NO: 2 punti – VO: 2 punti) Il partizionamento dell'indirizzo di primaria secondo il metodo di indirizzamento suddetto.
- 3) (NO: 3 punti – VO: 3 punti) La stima dell'"hit ratio" di cache, considerando la cache vuota all'istante iniziale e che vengano prodotte le seguenti richieste di accesso espresse in esadecimale:

21A5BC
21A4F1
F98018
21A492
21A5AA
F9815B
21A5B6
F981EA

- 4) (NO: 2 punti – VO: 1 punto) Il calcolo del tempo medio di accesso alla gerarchia, in nanosecondi, utilizzando i valori trovati al punto 1 (per il tempo di accesso al disco) e al punto 3 (per l' "hit ratio" di cache), e considerando che l'"hit ratio" di primaria sia pari a 0.9, il tempo di accesso in cache sia pari a 4 ns e quello in primaria pari a 50 ns.

**Soluzione.**

- 1) E' sufficiente valutare il tempo di latenza e il tempo di posizionamento una volta sola, in quanto il blocco è registrato su 2 settori contigui.

$$TROT = 60 / 7200 = 0.0083 \text{ secondi}$$

$$TLAT = TROT / 2 = 0.00415 \text{ secondi}$$

$$Tlett = TROT / 120 = 0.0694 \text{ ms}$$

Tempo di lettura di un blocco =

$$TLAT + TPOS + 2 * Tlett = 4.15 + 2 + 2 * 0.0694 = 6.2888 \text{ ms}$$

- 2) La cache è formata da  $2^{20}$  parole, suddivise in blocchi da  $2^9$  parole, pertanto il numero di blocchi in cache è  $2^{11}$ , indirizzabili con 11 bit. Per indirizzare le parole di ogni blocco occorrono invece 9 bit. Il partizionamento è il seguente:

< TAG 4 bit > < Cache Index 11 bit > < Offset 9 bit >

- 3) Per stimare l' "hit ratio" di cache, è sufficiente eseguire il rapporto fra numero di "hit" e numero totale di accessi. Se all'istante iniziale la cache è vuota, la sequenza di richieste produce i risultati in tabella. Indichiamo ciascuna cifra esadecimale con il relativo quartetto di bit evidenziando, in neretto, il campo "cache index":

Parola	$a_3a_2a_1a_0$	<b><math>b_3b_2b_1b_0</math></b>	<b><math>c_3c_2c_1c_0</math></b>	<b><math>d_3d_2d_1d_0</math></b>	$e_3e_2e_1e_0$	$f_3f_2f_1f_0$	Hit
21A5BC	2	<b>0001</b>	<b>1010</b>	<b>0101</b>	B	C	
21A4F1	2	<b>0001</b>	<b>1010</b>	<b>0100</b>	F	1	Sì
F98018	F	<b>1001</b>	<b>1000</b>	<b>0000</b>	1	8	
21A492	2	<b>0001</b>	<b>1010</b>	<b>0100</b>	9	2	Sì
21A5AA	2	<b>0001</b>	<b>1010</b>	<b>0101</b>	A	A	Sì
F9815B	F	<b>1001</b>	<b>1000</b>	<b>0001</b>	5	B	Sì
21A5B6	2	<b>0001</b>	<b>1010</b>	<b>0101</b>	B	6	Sì
F981EA	F	<b>1001</b>	<b>1000</b>	<b>0001</b>	E	A	Sì

L' "hit ratio" di cache stimato è pari a:  $6/8 = 0.75$ .

- 4) Con tutti i dati a nostra disposizione è sufficiente valutare la formula:

$$\bar{T} = H_c T_c + (H_p - H_c)(T_p + T_c) + (1 - H_p)(T_d + T_p + T_c)$$

$$\bar{T} = 0.75 \cdot 4 + 0.15 \cdot 54 + 0.1 \cdot 6288854 = 628896.5ns$$

**ESERCIZIO 5 (solo VO: 7 punti)**

Si consideri una architettura basata su accumulatore. Il contenuto della memoria a un certo istante è il seguente (per semplicità usiamo la notazione decimale): l'indirizzo 100 contiene il valore 300; l'indirizzo 250 contiene 150; l'indirizzo 300 contiene 500; l'indirizzo 400 contiene 250. Dire quale sarà il contenuto dell'accumulatore dopo le seguenti istruzioni:

Istruzioni	Modalità indirizz. (1)	Modalità indirizz. (2)	Modalità indirizz. (3)
LOAD 100	immediato	diretto	diretto
ADD 400	diretto	diretto	indiretto

nei tre casi riportati nelle tre colonne.

**Soluzione**

*Nel caso di indirizzamento immediato, il campo indirizzi contiene l'operando; nel caso di indirizzamento diretto il campo indirizzi contiene l'indirizzo di memoria ove risiede l'operando; infine, nel caso di indirizzamento indiretto il campo indirizzi contiene l'indirizzo di memoria che a sua volta contiene l'indirizzo di memoria ove risiede l'operando.*

Caso (1):

'LOAD 100' carica il valore 100 nell'accumulatore  $AC \leftarrow 100$

'ADD 400' somma al contenuto dell'accumulatore il valore in memoria all'indirizzo 400:  $AC \leftarrow AC + M[400]$ ,  $AC \leftarrow 100 + 250$ . Il contenuto finale dell'accumulatore è 350

Caso (2):

'LOAD 100' carica nell'accumulatore il valore in memoria all'indirizzo 100:  $AC \leftarrow M[100]$ ,  $AC \leftarrow 300$

'ADD 400' somma al contenuto dell'accumulatore il valore in memoria all'indirizzo 400:  $AC \leftarrow AC + M[400]$ ,  $AC \leftarrow 300 + 250$ . Il contenuto finale dell'accumulatore è 550.

Caso (3):

'LOAD 100' carica nell'accumulatore il valore in memoria all'indirizzo 100:  $AC \leftarrow M[100]$ ,  $AC \leftarrow 300$

'ADD 400' somma al contenuto dell'accumulatore il valore in memoria all'indirizzo contenuto in 400:

$AC \leftarrow AC + M[M[400]]$ ,  $AC \leftarrow 300 + M[250]$ ,  $AC \leftarrow 300 + 150$ . Il contenuto finale dell'accumulatore è 450.