

SOLUZIONI DELLA SECONDA PROVA INTERMEDIA DEL CORSO DI
CALCOLATORI ELETTRONICI
NUOVO ORDINAMENTO DIDATTICO
8 Giugno 2004

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

ESERCIZIO 1 (8 punti)

PARTE 1 (3 punti).

Un bus sincrono presenta le seguenti caratteristiche:

- frequenza di clock: 250 MHz
- durata di una trasmissione sul bus: 2 cicli di clock
- ampiezza linea dati: 32 bit

1. (1 punto) Illustrare chiaramente il protocollo di lettura su bus sincrono utilizzando l'opportuno grafico. (1 punti)
2. (2 punti) Calcolare la velocità di trasferimento durante una operazione di lettura di un dato dalla memoria, sapendo che le parole di memoria hanno dimensione pari a 64 bit, e il tempo di ciclo della memoria richiede 20 cicli di clock.

PARTE 2 (5 punti).

Si consideri un calcolatore in cui la CPU esegue 10^5 istruzioni/s. L'esecuzione di una istruzione richiede 5 cicli di clock, 3 dei quali tengono occupato il bus di sistema. Si ipotizzi che il 75% dell'Instruction Rate sia usato dalla CPU per eseguire programmi che non contengono trasferimenti di I/O. L'ampiezza della linea dati del bus è pari a 32 bit.

Si consideri il caso in cui il trasferimento dei dati avvenga mediante IO da programma, con le seguenti 4 istruzioni:

- 1) LOAD *parola* dalla periferica al registro CPU
 - 2) STORE *parola* da registro CPU a memoria
 - 3) generazione indirizzo di memoria successivo
 - 4) conteggio dati da trasferire.
- a) (2 punti) Calcolare la massima frequenza di trasferimento dati ottenibile (espressa in kB/s) fra una periferica collegata al bus di sistema e la memoria principale mediante I/O da programma.
 - b) (3 punti) Calcolare la massima frequenza di trasferimento dati ottenibile (espressa in kB/s) nel caso in cui si usi la modalità "transparent" DMA. Si ipotizzi che una operazione di lettura/scrittura della memoria richieda un ciclo di clock.

Soluzione PARTE 1:

La durata di un ciclo di clock è pari a $1/(250 \text{ MHz}) = 4 \text{ ns}$

La lettura su un bus sincrono avviene secondo il protocollo seguente:

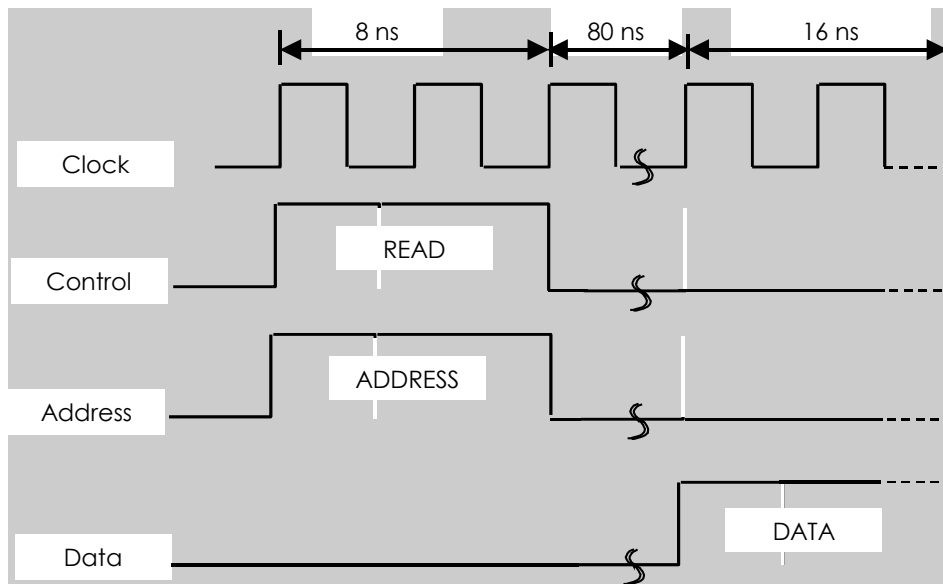
- Segnale di READ sulla linea di controllo e contemporaneamente l'indirizzo della locazione in cui risiede il dato sulla linea indirizzi:

2 cicli di clock = 8 ns

- Lettura della parola dalla memoria: **80 ns**
- Trasferimento della parola dalla memoria:
se la parola da leggere ha ampiezza pari a 64 bit

trasferimento della parola = 16 ns

Tempo totale per leggere una parola dalla memoria = 104 ns



Velocità di trasferimento per una parola di memoria da 64 bit:

$$\frac{64}{104 \cdot 10^{-9}} (\text{bit} / \text{sec}) \cdot \frac{1}{8} (B / \text{bit}) \cdot 2^{-20} (MB / B) \cong 73.36 (MB / \text{sec})$$

Soluzione parte 2

a) Nel caso di trasferimento mediante I/O da programma, per trasferire una parola occorrono 8 istruzioni. La CPU è impegnata per l'75% del tempo a eseguire istruzioni che non coinvolgono l'I/O, dunque può usare solo il 25% del tempo per eseguire istruzioni di trasferimento dati con periferiche. In termini di istr./sec questo tempo è pari a **$0.25 \times 10^5 \text{ istr./s} = 2.5 \times 10^4 \text{ istr./s}$** . Dal momento che per trasferire una parola servono quattro istruzioni, la velocità di trasferimento è pari a:

$2.5 \times 10^4 \text{ istr./s} / (8 \text{ istr./parola}) = 3125 \text{ parole/s}$. La dimensione di una parola è pari a 64 bit (8 byte), da cui si ricava la velocità di trasferimento di **24.41 kB/s** ,

b) Nel caso di 'transparent DMA' posso trasferire i dati tutte le volte che il bus di sistema è libero. Nel caso in esame questo tempo è pari alla somma del 25% del tempo lasciato libero dall'esecuzione di istruzioni che non coinvolgono I/O, più i due cicli/istruzione in cui il bus è libero. Pertanto durante il 75% del tempo posso trasferire due parole/istr.:

$$0.75 \times 2 \times (1/2) \text{ parole/istr} \times 10^5 \text{ istr./s} = 0.75 \times 10^5 \text{ parole/s}$$

Nel restante 5% del tempo posso trasferire 5 parole/istr.:

$$0.25 \times 5 \times (1/2) \text{ parole/istr.} \times 10^5 \text{ istr./s} = 0.625 \times 10^5 \text{ parole/s}$$

In **totale**, nel caso di trasferimento con DMA la velocità totale di trasferimento è pari a:

$$(0.75 + 0.625) \times 10^5 \text{ parole/s} = 1.375 \times 10^5 \text{ parole/s} = 1074.22 \text{ kB/s}$$

ESERCIZIO 2 (6 punti)

Considerato un campo di 64 bit, siano dati i seguenti formati:

1. rappresentazione di interi senza segno;
2. rappresentazione in virgola fissa con 20 bit di parte frazionaria;
3. rappresentazione in virgola mobile con mantissa frazionaria e normalizzata in segno e valore (1.M) ed esponente a 8 bit in eccesso 128.

a) (2 punti) Calcolare il minimo e il massimo valore rappresentabile in valore assoluto nei tre casi.

c) (4 punti) Sommare i due numeri, $(12.5)_{10}$ $(5.25)_{10}$, esprimendoli in virgola mobile secondo la rappresentazione 3, con l'algoritmo dei calcolatori.

Soluzione

a)

1. Minimo: 1 Max: $2^{64}-1$.
2. Minimo: 2^{-20} Max: $2^{43}-2^{-20}$
3. Minimo: 2^{-128} Max: $2^{127}(2-2^{-55})$.

b) $(12.5)_{10} = 1100.1 = 1.1001 \cdot 2^3$
 $(5.25)_{10} = 101.01 = 1.0101 \cdot 2^2$

I due numeri si possono rappresentare nel seguente modo:

Segno	Esponente	Mantissa
0	10000011	1001000000000000000000...0
0	10000010	0101000000000000000000...0

Poiché il primo ha esponente maggiore del secondo ($3 > 2$) di quest'ultimo si fa scorrere la mantissa a destra di una posizione.

I due numeri da sommare sono:

$$\begin{array}{r} 1.10010 + \\ 0.10101 = \\ \hline 10.00111 \quad (*2^3) \end{array}$$

E' necessario normalizzare il risultato

Segno	Esponente	Mantissa
0	10000100	0001110000000000000000...0

ESERCIZIO 3 (8 punti)

Scrivere una funzione Assembler MIPS che, ricevendo in ingresso due vettori x e w, restituisca la massima differenza in valore assoluto delle rispettive componenti. La dimensione dei due vettori è di N elementi.

Si assuma che: $\&v[0] \rightarrow \$4$, $\&w[0] \rightarrow \$5$, $N \rightarrow \$6$, valore di uscita $\rightarrow \$7$.

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

```
int maxabs(int *v, int *w, int N)
{
    int i, max, diff;
    max=0;
    for(i=0; i<N; i++)
    {
        diff=v[i]-w[i];
        if(diff<0) diff = -diff;
        if(diff>max) max = diff;
    }
    return max;
}
```

Soluzione.

Proponiamo una soluzione che fa uso dei seguenti registri per i dati intermedi:
 $\$8 \leftarrow i$; $\$9 \leftarrow \text{diff}$; $\$10 \leftarrow v[i]$; $\$11 \leftarrow w[i]$; $\$12 \leftarrow (\text{diff} < 0)$; $\$13 \leftarrow (\text{max} < \text{diff})$

```
maggiore: addi $29, $29, -24 #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)
           move $7, $0           #max=0
           move $8, $0           #i=0
For:       beq $8, $6, Exit       #if(i==N) Exit
           lw $10, 0($4)         #prelevo v[i]
           lw $11, 0($5)         #prelevo w[i]
           sub $9, $10, $11      #diff = v[i] - w[i]
           slt $12, $9, $0       #se diff>=0
           beq $12, $0, continue1 #allora salta a continue1
           sub $9, $0, $9        #altrimenti diff=-diff
continue1: slt $13, $7, $9       #se diff<=max
           beq $13, $0, continue2 #allora salta a continue2
           move $7, $9          #altrimenti max=diff
continue2: addi $4, $4, 4        #locazione v[i+1]
           addi $5, $5, 5        #locazione w[i+1]
           addi $8, $8, 1        #i++
           j For
Exit:      lw $8, 0($29)         #ripristino del contesto
           lw $9, 4($29)
           lw $10, 8($29)
           lw $11, 12($29)
           lw $12, 16($29)
           lw $13, 20($29)
           addi $29, $29, 24
           jr $31               #ritorno al chiamante
```

ESERCIZIO 4 (9 punti)

Si consideri il seguente insieme di processi.

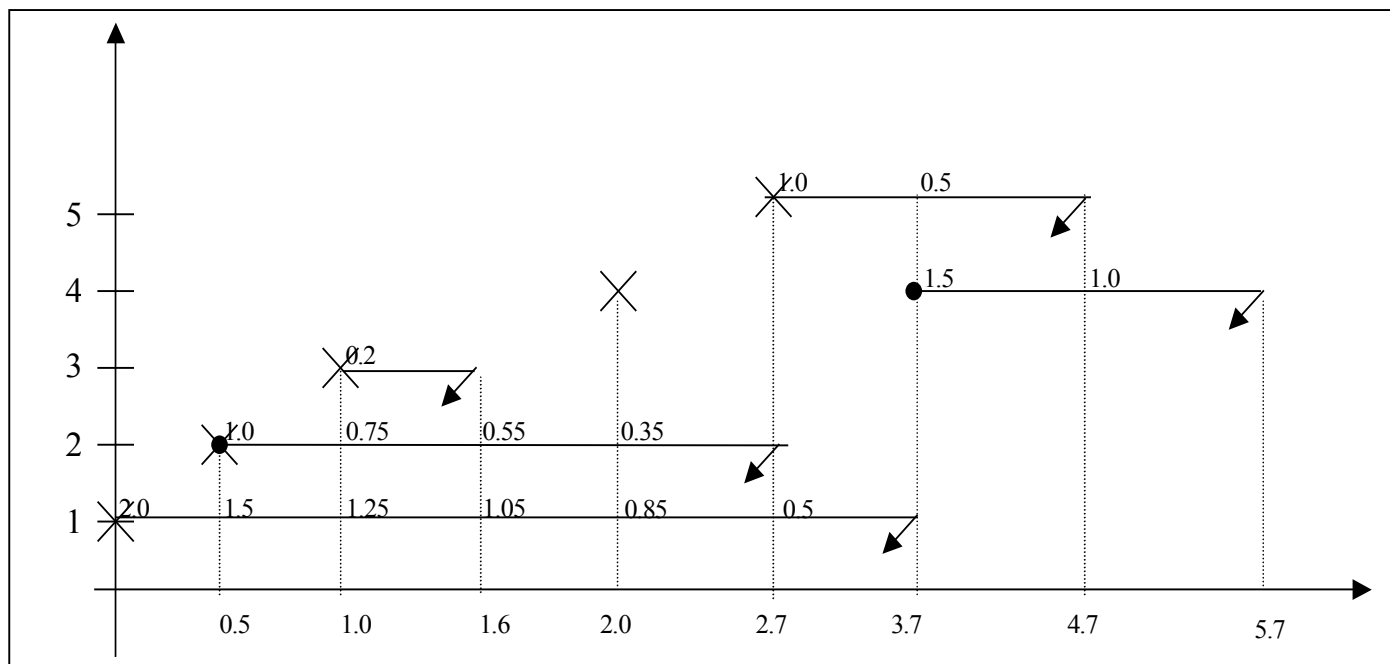
Processo	Tempo di arrivo	Tempo di CPU richiesto	Memoria richiesta
1	0.00	2.00	125K
2	0.50	1.00	100K
3	1.00	0.20	40K
4	2.00	1.50	200K
5	2.70	1.00	115K

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".

- Mostrare col metodo grafico lo sviluppo di ciascun processo e lo stato della memoria (7 punti).
- Definire e calcolare il tempo di turnaround medio e di turnaround pesato medio (2 punti).

Soluzione.



Stato della memoria.

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
Media				2.44	2.30