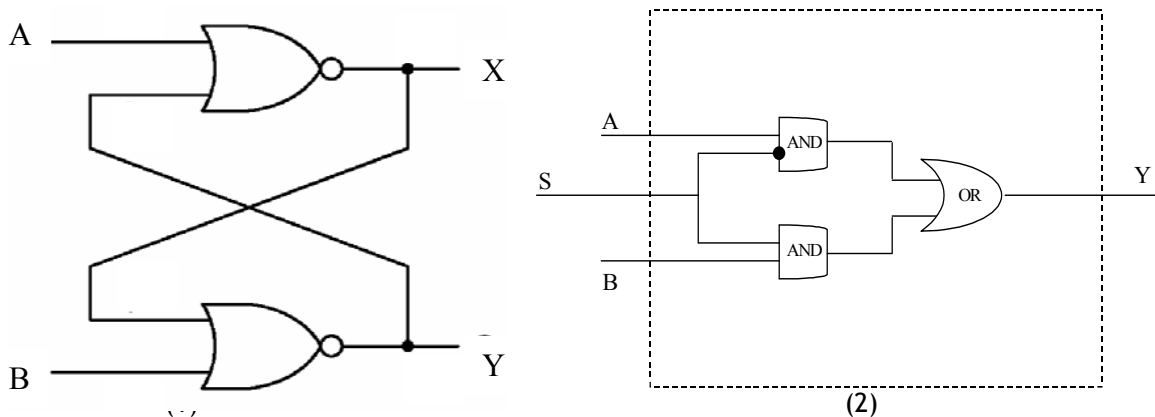


**SOLUZIONI DELLA PROVA SCRITTA DEL CORSO DI
CALCOLATORI ELETTRONICI
NUOVO ORDINAMENTO DIDATTICO
27 Gennaio 2005**

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

ESERCIZIO 1 (8 punti)

- (a) (3 punti) Spiegare in modo chiaro e sintetico quale delle due seguenti reti logiche è sequenziale e perché.



- (b) (5 punti) Precisare qual è la funzionalità svolta dalle due reti.

Soluzione.

- (a) La rete sequenziale è ovviamente la (1) perché presenta delle “retroazioni”.
 (b) La rete (1) non è altro che un latch SR asincrono. Scrivendo infatti l’espressione booleana della funzione svolta si ha:

$$Y(t + \tau) = \overline{B + X(t)} = \overline{B + A + Y(t)} = \overline{B} \cdot (\overline{A + Y(t)})$$

che corrisponde alla formula di un latch SR in cui: $A=S$, $B=R$, $Y=Q$, $X=Q'$.

La rete (2) è un multiplexer 2-1. Infatti:

$$Y = A\overline{S} + BS$$

Si ottiene così un MUX 2-1 dove A e B sono i segnali di input e S è il segnale di controllo, tale che per $S=0$, $Y=A$, per $S=1$, $Y=B$.

ESERCIZIO 2 (9 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) (4 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) (5 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

- a) Nel caso di trasferimento mediante I/O da programma, per trasferire una parola occorrono 4 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:
 $2.5 \times 10^4 \text{ istr./s} / (4 \text{ istr./parola}) = 6250 \text{ parole/s}$. La dimensione di una parola è pari a 32 bit (4 byte), da cui si ricava la velocità di trasferimento di **24.41 kB/s**,
- b) Nel caso di 'trasparent 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 una parola/istr.:
 $0.75 \times 2 \text{ parole/istr} \times 10^5 \text{ istr./s} = 1.5 \times 10^5 \text{ parole/s}$
Nel restante 5% del tempo posso trasferire 2.5 parole/istr.:
 $0.25 \times 5 \text{ parole/istr.} \times 10^5 \text{ istr./s} = 1.25 \times 10^5 \text{ parole/s}$
In totale, nel caso di trasferimento con DMA la velocità totale di trasferimento è pari a: $(1.5 + 1.25) \times 10^5 \text{ parole/s} = 2.75 \times 10^5 \text{ parole/s} = 1074.22 \text{ kB/s}$

ESERCIZIO 3 (8 punti)

E' dato il seguente codice Assembler MIPS.

F:	addi \$29, \$29, -16		lw \$11, 0(\$4)
	sw \$8, 0(\$29)		slt \$9, \$11, \$6
	sw \$9, 4(\$29)		beq \$9, \$0, A
	sw \$10, 8(\$29)		move \$6, \$11
	sw \$11, 12(\$29)		j A
	lw \$6, 0(\$4)	B:	lw \$8, 0(\$29)
	move \$10, \$0		lw \$9, 4(\$29)
A:	addi \$10, \$10, 1		lw \$10, 8(\$29)
	slt \$8, \$10, \$5		lw \$11, 12(\$29)
	beq \$8, \$0, B		addi \$29, \$29, 16
	addi \$4, \$4, 4		jr \$31

Spiegare il significato di ogni istruzione e individuare qual è la funzione svolta dal codice presentato. Si ricordi che lo stack pointer è contenuto in \$29, che \$31 è tipicamente usato per il ritorno da procedura (contiene il PC).

(La risposta è valida anche traducendo il codice MIPS in C)

Soluzione. Il seguente blocco:

```
Fun:    addi $29, $29, -16
        sw $8, 0($29)
        sw $9, 4($29)
        sw $10, 8($29)
        sw $11, 12($29)
```

effettua il salvataggio dei registri \$8...\$11 nello stack (puntato da \$29).

Le istruzioni successive:

```
lw $6, 0($4)
move $10, $0
```

caricano in \$6 il valore puntato da \$4, e azzerano il contenuto di \$10.

Il frammento di codice etichettato con 'A':

```
A:      addi $10, $10, 1
        slt $8, $10, $5
        beq $8, $0, B
        addi $4, $4, 4
        lw $11, 0($4)
        slt $9, $11, $6
        beq $9, $0, A
        move $6, $11
        j A
```

aggiorna il valore \$10 e controlla se \$10 risulta minore di \$5. Se no, salta a B.

Se sì, preleva il valore contenuto nella locazione adiacente a \$4 e lo confronta con \$11. Se $S11 < \$6$, allora esso viene spostato in \$6, altrimenti si ritorna ad A e si accede alla successiva locazione puntata da \$4.

Il frammento di codice etichettato con 'B' effettua il ripristino dei parametri dallo stack, e ritorna al chiamante.

```
B:      lw $8, 0($29)
        lw $9, 4($29)
        lw $10, 8($29)
        lw $11, 12($29)
        addi $29, $29, 16
        jr $31
```

Il codice proposto ha dunque lo scopo di trovare il più piccolo numero entro una sequenza di 5 celle di memoria adiacenti, il cui indirizzo iniziale si trova in \$4. Tale minimo viene depositato in \$6.

ESERCIZIO 4 (8 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) (4 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	100000111001000000000000000000...0	
0	100000100101000000000000000000...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	100001000001110000000000000000...0	