

**SOLUZIONI DELLA PROVA SCRITTA DEL CORSO DI
CALCOLATORI ELETTRONICI
NUOVO E VECCHIO ORDINAMENTO DIDATTICO**
28 Settembre 2004

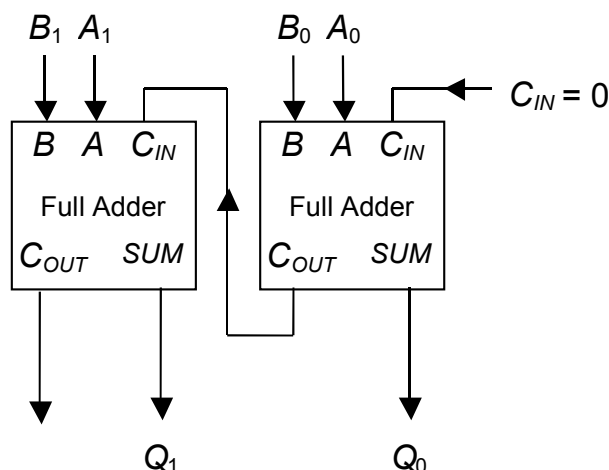
MOTIVARE IN MANIERA CHIARA LE SOLUZIONI PROPOSTE A CIASCUNO DEGLI ESERCIZI SVOLTI

ESERCIZIO 1 (NO: 8 punti – VO: 6 punti)

- (1) (NO: 5 punti – VO: 4 punti) Si progetti, con il metodo delle mappe di Karnaugh, e disegni la rete logica minima che implementa un addizionatore parallelo con due addendi di due bit. Il disegno deve mostrare tutte le porte logiche ed i loro collegamenti.
- (2) (NO: 3 punti – VO: 2 punti) Assumendo che una porta logica introduca un ritardo pari a T , quale è il tempo necessario ad effettuare la somma ?

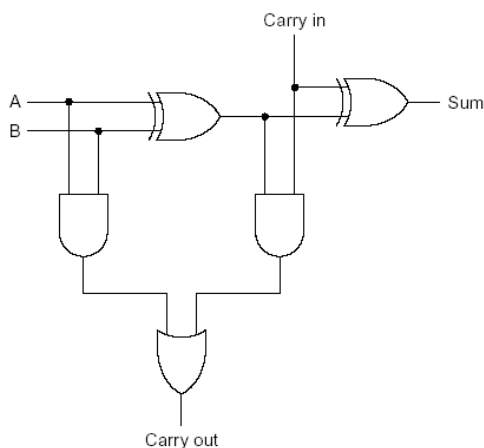
Soluzione.

Un addizionatore parallelo ("parallel adder") a due bit è costituito dalla connessione in serie di due unità dette "full adder" (v. dispense, pagg.40-41), come rappresentato in figura. Abbiamo indicato con A_1A_0 i bit del primo addendo, B_1B_0 i bit del secondo addendo (A_1 e B_1 sono i bit più significativi).



Ciascun full adder è caratterizzato dalla seguente rete logica:

A	B	Carry in	Sum	Carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Poiché si tratta di porte logiche a due livelli, ogni full adder produce un ritardo pari a $2T$. Conseguentemente, il ritardo complessivo del parallel adder sarà $4T$.

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

Si consideri un calcolatore che dispone di una memoria principale di 64 kbyte suddivisa in blocchi di 8 byte. E' possibile accedere al singolo byte e la modalità di indirizzamento usata per la cache, costituita da 32 blocchi indirizzabili, sia quella "diretta".

- (1) (NO: 2 punti – VO: 1 punto) Spiegare, precisando il significato e la funzione dei diversi campi, come vengono interpretati gli indirizzi logici per recuperare l'informazione contenuta nella cache.
- (2) (2 punti) Indicare in quali blocchi di primaria si trovano i seguenti byte (indirizzi in esadecimale): 113B, C334, 1138, AAAA.
- (3) (3 punti) Indicare in quali blocchi di cache devono essere memorizzati i byte del passo precedente. Se tali parole venissero richieste sequenzialmente, quanti sarebbero gli hit di cache (ipotizzando la cache inizialmente vuota) ?
- (4) (2 punti) Si supponga che il byte di indirizzo 1A1B sia memorizzato in cache. Indicare gli indirizzi di tutti gli altri byte memorizzati nello stesso blocco di cache.

Soluzione

(a) <Tag 8 bit> <Cache Index 5 bit> <Offset 3 bit>

(b-c)

```

113B → 0001 0001 | 0011 1 | 011 → Block frame (551)10 → Cache index 7
C334 → 1100 0011 | 0011 0 | 100 → Block frame (6246)10 → Cache index 6
1138 → 0001 0001 | 0011 1 | 000 → Block frame (551)10 → Cache index 7
AAAA → 1010 1010 | 1010 1 | 010 → Block frame (5461)10 → Cache index 21

```

Se le parole venissero richieste sequenzialmente si verificherebbe un hit (la terza parola è già stata memorizzata in cache dopo il primo accesso).

(d)

Essendo:

1A1B → 0001 1010 0001 1 | 011,

si ottiene facilmente che gli altri byte contenuti nello stesso blocco sono: 1A18 (offset 000), 1A19 (offset 001), 1A1A (offset 010), 1A1C (offset 100), 1A1D (offset 101), 1A1E (offset 110), 1A1F (offset 111).

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)

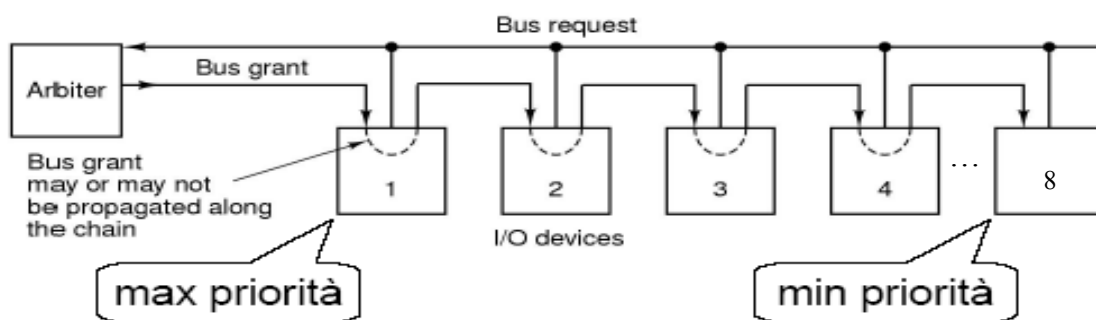
Si consideri il caso di un Bus di I/O sul quale devono essere collegate 8 periferiche esterne.

1) (4 punti) Si ipotizzi che sulla parte di controllo del Bus si abbiano solo 2 segnali di controllo liberi da poter utilizzare per gestire l'arbitraggio delle 8 periferiche da collegare. Descrivere la tecnica di arbitraggio utilizzabile in una tale situazione, disegnando un possibile schema di collegamento delle periferiche e spiegando chiaramente le implicazioni del suo utilizzo sulla gestione delle richieste di I/O delle periferiche.

2) (3 punti) Nelle ipotesi di cui al punto precedente, sarebbe utilizzabile la tecnica di arbitraggio centralizzato con richieste indipendenti per ciascuna periferica? In caso di risposta negativa, cosa sarebbe necessario modificare per renderla utilizzabile?

Soluzione.

1) Con soli 2 segnali liberi è possibile solo utilizzare la tecnica "daisy chain" che ha solo 2 segnali (request e grant). Ovviamente questa tecnica implica una gestione rigida delle priorità, che sono definite una volta per tutte.



2) Non è ovviamente possibile. Bisognerebbe avere almeno $8 \times 2 = 16$ segnali di controllo liberi.

ESERCIZIO 4 (NO: 8 punti - VO: 7 punti)

I trasferimenti di parole a/dalla memoria di un calcolatore sono codificate utilizzando il codice di Hamming. Si consideri la stringa seguente che rappresenta una parola codificata con il codice di Hamming (il bit meno significativo è a sinistra): 011101010001.

1. (2 punti) Quale è la dimensione della parola trasmessa (escludendo i bit di codice)?
2. (4 punti) Verificare se la stringa ricevuta contiene un errore e in caso affermativo correggerlo.
3. (NO: 2 punti – VO: 1 punto) Scrivere la parola corretta che si voleva trasmettere.

Soluzione.

La stringa trasmessa contiene 12 bit, dunque i bit di codice sono 4 e la parola trasmessa ha dimensione di 8 bit (il numero di bit di codice m è scelto come il più piccolo intero che soddisfa la relazione $m + b \leq 2^m - 1$, dove b indica il numero di bit della parola da codificare. In questo caso $m+b=12$ e il più piccolo intero che soddisfa la relazione precedente è 4).

Se indichiamo con b i bit della parola inviata e con c i bit di codice, la stringa di 12 bit deve essere interpretata nel modo seguente:

c_0	c_1	b_0	c_2	b_1	b_2	b_3	c_3	b_4	b_5	b_6	b_7
0	1	1	1	0	1	0	1	0	0	0	1

Il calcolo dei bit di errore è il seguente:

$$e_0 = c_0 \oplus b_0 \oplus b_1 \oplus b_3 \oplus b_4 \oplus b_6 = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 = 1$$

$$e_1 = c_1 \oplus b_0 \oplus b_2 \oplus b_3 \oplus b_5 \oplus b_6 = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 1$$

$$e_2 = c_2 \oplus b_1 \oplus b_2 \oplus b_3 \oplus b_7 = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 1$$

$$e_3 = c_3 \oplus b_4 \oplus b_5 \oplus b_6 \oplus b_7 = 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 = 0$$

Posizione dell'errore = 7

Nella stringa il 7° bit (che corrisponde a b_3) è errato.

La parola di 8 bit trasmessa (bit meno significativo a sinistra) è dunque: 10110001.

ESERCIZIO 5 (solo VO: 5 punti)

In una macchina RISC, descrivere quali problemi sono determinati dalle istruzioni di salto e in che modo vengono risolti.

Soluzione.

V. dispense del corso.