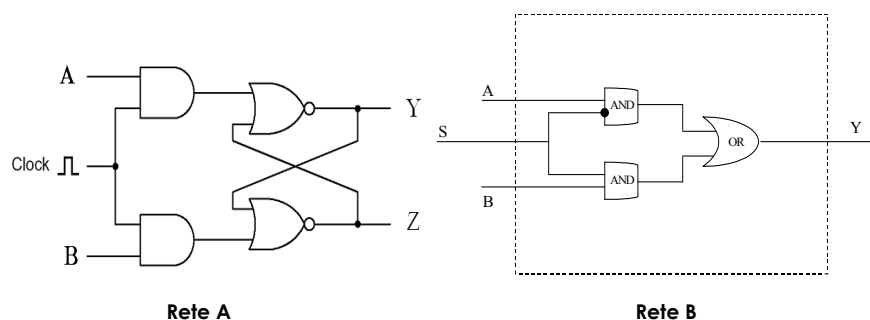


**SOLUZIONI DELLA PROVA SCRITTA DEL CORSO DI
CALCOLATORI ELETTRONICI
NUOVO ORDINAMENTO DIDATTICO
29 Gennaio 2004**

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

ESERCIZIO 1 (8 punti)

- (a) (5 punti) Progettare un multiplexer 2-1. Definire la tabella di transizione, semplificandola con le mappe di Karnaugh, e disegnare il relativo circuito con le porte logiche.
- (b) (3 punti) Spiegare in modo chiaro e sintetico quale delle due seguenti reti logiche è sequenziale e perché. Spiegare in modo chiaro e sintetico la funzione implementata dalla Rete A.



Soluzione.

Dati due ingressi A e B, è sufficiente un bit di controllo S per "pilotare" opportunamente l'uscita Y. Assumiamo che, se S=0, allora Y=A, altrimenti Y=B:

Da cui:

S	A	B	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$$Y = \bar{S}A + SB$$

Circuito:

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

ESERCIZIO 2 (8 punti)

Scrivere una funzione Assembler MIPS che ricevendo in ingresso un vettore di interi v di dimensione N e un intero t, sposti all'inizio del vettore tutti i valori di v minori di t. Si supponga che &v[0] sia memorizzato nella locazione 1000, \$5 ← t, \$6 ← N. Ad esempio, se v = {11, 10, 20, 9, 5} e t = 10, allora, alla fine della funzione, si deve avere v = {9, 5, 20, 11, 10}.

In altri termini, la funzione MIPS potrebbe implementare la seguente funzione C:

```
void verifica(int *v, int t, int N) {
    int i, j, temp;

    j=0;
    for(i=0; i<N; i++)
        if(v[i]<t) {
            temp=v[j];
            v[j]=v[i];
            v[i]=temp;
            j++;
        }
}
```

Soluzione.

Assumiamo che:

i → \$8, j → \$9, temp → \$10, (v[i]<t) → \$11, v[i] → \$12

```
verifica:
    addi $29, $29, -20    #salvataggio del contesto
    sw $8, 0($29)
    sw $9, 4($29)
    sw $10, 8($29)
    sw $11, 12($29)
    sw $12, 16($29)

    muli $6, $6, 4
    move $9, $0           #j=0
    move $8, $0           #i=0
```

```
for:
    beq $8, $6, exit      #if(i==N) exit
    lw $12, 1000($8)      #$12 ← v[i]
```

```

slt $t1, $t2, $5      #$t1 ← v[i]<t
beq $t1, $0, update_i  #if(!(v[i]<t)) update_i
lw $t0, 1000($9)      #$t0 ← v[j], ovvero temp=v[j]
sw $t2, 1000($9)      #v[j]=v[i]
sw $t0, 1000($8)      #v[i]=temp
addi $9, $9, 4        #j++

update_i:
addi $8, $8, 4
j for

exit:
lw $8, 0($29)         #ripristino del contesto
lw $9, 4($29)
lw $t0, 8($29)
lw $t1, 12($29)
lw $t2, 16($29)
addi $29, $29, 20
jr $31               #ritorno al chiamante

```

ESERCIZIO 3 (9 punti)

Si consideri un calcolatore che dispone di una memoria principale di 64 Mbyte 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".

- (a) (2 punti) Spiegare, precisando il significato e la funzione dei diversi campi, come vengono interpretati gli indirizzi logici per recuperare l'informazione contenuta nella cache.
- (b) (2 punti) Indicare in quali blocchi di primaria si trovano i seguenti byte (indirizzi in esadecimale): 111B, C334, D01D, AAAA.
- (c) (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) ?
- (d) (2 punti) Si supponga che il byte di indirizzo 1A1A 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)

```

111B → 0001 0001 | 0001 1 | 011 → Block frame (547)10 → Cache index 3
C334 → 1100 0011 | 0011 0 | 100 → Block frame (6246)10 → Cache index 6
D01D → 1101 0000 | 0001 1 | 101 → Block frame (6659)10 → Cache index 3
AAAA → 1010 1010 | 1010 1 | 010 → Block frame (5461)10 → Cache index 21

```

Poiché le quattro parole sono localizzate in blocchi di primaria differenti, se esse venissero richieste sequenzialmente non si verificherebbe nessun hit.

(d)

Essendo:

1A1A → 0001 1010 0001 1 | 010,

si ottiene facilmente che gli altri byte contenuti nello stesso blocco sono:

1A18 (offset 000), 1A19 (offset 001), 1A1B (offset 011), 1A1C (offset 100), 1A1D (offset 101),
1A1E (offset 110), 1A1F (offset 111).

ESERCIZIO 4 (8 punti)

(a) (4 punti) Il sistema di memoria virtuale di un calcolatore viene gestito mediante la tecnica di "paginazione su richiesta". Il sistema operativo indirizza 8 pagine virtuali, mentre la memoria del calcolatore contiene 4 pagine fisiche. La dimensione di ciascuna pagina è uguale a 1024 parole. Ad un certo istante si abbia il contenuto della PMT riportata a lato.

Elencare tutti gli indirizzi virtuali, espressi in decimale, che provocano un "page fault".

(b) (4 punti) Spiegare in modo chiaro e sintetico in cosa consiste il problema del "trashing" nella paginazione su richiesta, e indicare i principali algoritmi di "trashing" utilizzati.

Indirizzo di pagina virtuale	Indirizzo di pagina fisica
0	2
1	-
2	-
3	-
4	3
5	-
6	1
7	0

Soluzione

(a) Gli indirizzi virtuali che generano un "page fault" sono quelli che corrispondono a pagine virtuali non caricate in memoria. Nell'esempio riportato si tratta delle pagine 1, 2, 3, 5, cui corrispondono rispettivamente gli indirizzi virtuali da 1024 a 2047, da 2048 a 3071, da 3072 a 4095 e da 5120 a 6143 (NB: l'indirizzo della prima parola della pagina x si calcola come $x \cdot 1024$, mentre l'indirizzo dell'ultima parola è $(x + 1) \cdot 1024 - 1$).

(b) Vedi lucidi del corso.