

PROVA SCRITTA DEL MODULO DI
CALCOLATORI ELETTRONICI
CORSO DI LAUREA IN INGEGNERIA ELETTRICA, ELETTRONICA ED INFORMATICA
CORSO DI LAUREA IN INGEGNERIA BIOMEDICA
24 gennaio 2019

NOME:

COGNOME:

MATRICOLA:

CFU:

ESERCIZIO 1 (8 punti)

Progettare una rete sequenziale che presenti un ingresso X e un'uscita Z posta a 1 ogni volta che viene riconosciuta la sequenza 10111. Si richiede:

1. (3 punti) il diagramma degli stati, la tabella di flusso e la tabella delle transizioni;
2. (5 punti) il calcolo delle forme minime delle variabili di eccitazione dei flip flop con le mappe di Karnaugh. Si usino flip flop JK. Calcolare anche la rete combinatoria per l'uscita Z.

ESERCIZIO 2 (8 punti)

Si consideri un calcolatore che dispone di una memoria principale di 256 Mbyte e di una memoria cache di 512 Kbyte. E' possibile accedere al singolo byte e la memoria è suddivisa in blocchi da 16 byte.

1) Spiegare, precisando il significato e la funzione dei diversi campi, come vengono interpretati gli indirizzi logici per recuperare l'informazione contenuta nella cache nel caso in cui venga usata la modalità di indirizzamento:

(a) (2 punti) diretto;

(b) (2 punti) "associativo su insiemi", e ciascun insieme contenga due blocchi.

2) (4 punti) Si consideri la cache di cui alla domanda precedente. Ipotizzare che il processore acceda in sequenza ai byte dall'indirizzo 0000000 a 00003FF e da 0380000 a 03803FF (in questo ordine) e ripeta la sequenza di accesso data per 10 volte consecutive. Si ipotizzi inoltre che la cache sia inizialmente vuota. Calcolare l'hit rate considerando la cache descritta al punto precedente, sia nel caso di indirizzamento diretto sia nel caso di indirizzamento "associativo su insiemi".

ESERCIZIO 3 (8 punti)

Implementare una procedura Assembly MIPS che, dati l'indirizzo iniziale di un vettore v (in \$4) e due indici i e j (rispettivamente in \$5 e in \$6), permuti v[i] con v[j] solo se v[i] < v[j]. Restituisca 1 memorizzandolo in \$7 se è stata effettuata la permuta.

In altri termini, il codice MIPS può implementare la seguente funzione C:

```
int elabora(int *v, int i, int j)
{
    int t, p;
    p=v[i]<v[j];
    if (p)
    {
        t=v[i];
        v[i]=v[j];
        v[j]=t;
    }
    return p;
}
```

ESERCIZIO 4 (9 punti)

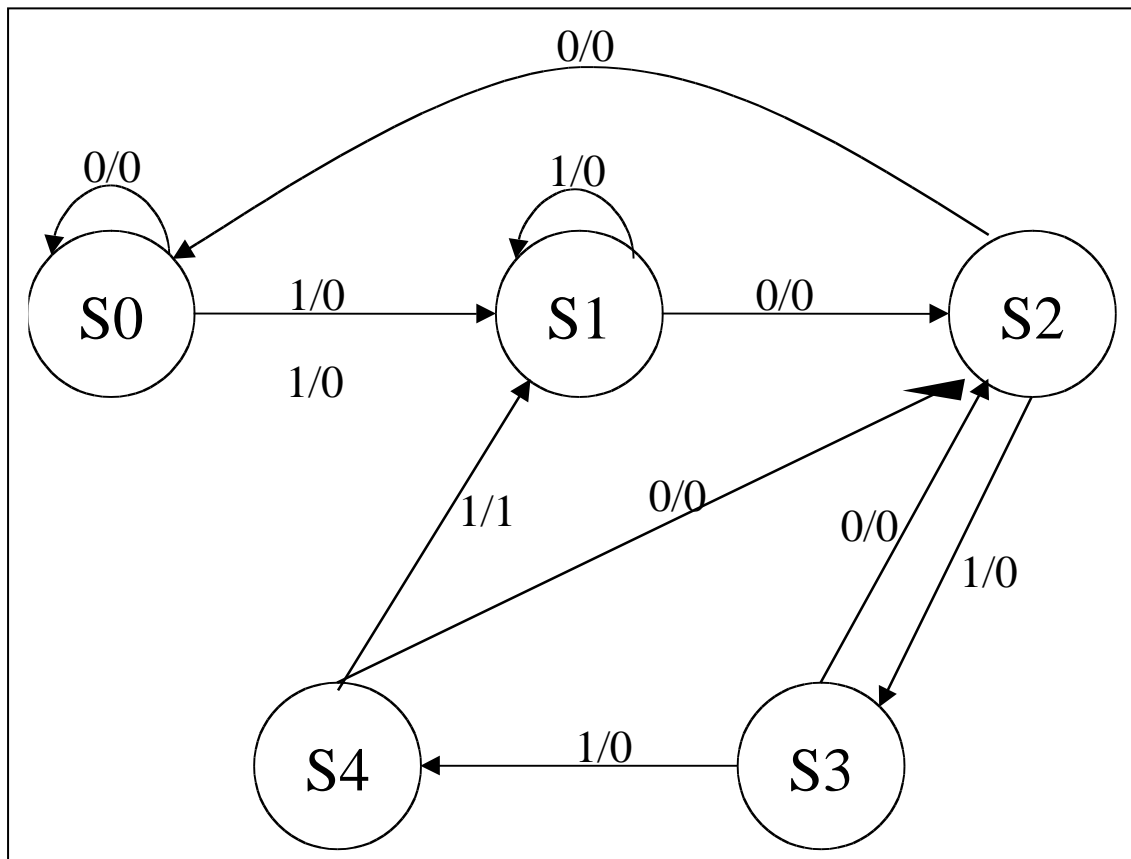
Si consideri un calcolatore in cui la CPU esegue 10^5 istruzioni/s. L'esecuzione di una istruzione richiede 5 cicli di clock, 3 dei quali tengono occupato il bus di sistema. Si ipotizzi che il 85% dell'Instruction Rate sia usato dalla CPU per eseguire programmi che non contengono trasferimenti di I/O. 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:

- a. LOAD parola dalla periferica al registro CPU
 - b. STORE parola da registro CPU a memoria
 - c. generazione indirizzo di memoria successivo
 - d. conteggio dati da trasferire.
1. (3 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.
 2. (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.
 3. (3 punti) Spiegare quali 'passi' sostituiscono le istruzioni nel caso DMA.

ESERCIZIO 1

Il diagramma degli stati è il seguente:



La tabella di flusso è data da:

Stato presente	Stato successivo/Uscita	
	X=0	X=1
S0	S0/0	S1/0
S1	S2/0	S1/0
S2	S0/0	S3/0
S3	S2/0	S4/0
S4	S2/0	S1/1

Per codificare 5 stati occorrono tre flip flop. La codifica è la seguente:

S0 → 0 0 0; ...; S4 → 1 0 0. Nel seguito indicheremo ciascun bit della codifica con le lettere A, B, C. L'apice indicherà il bit nell'istante successivo a quello considerato.

A partire dalla tabella di eccitazione del flip flop JK:

Q	Q'	J	K
0	0	0	D
0	1	1	D
1	0	D	1
1	1	D	0

A	B	C	X	A'	Ja	Ka	B'	Jb	Kb	C'	Jc	Kc	Z
0	0	0	0	0	0	D	0	0	D	0	0	D	0
0	0	0	1	0	0	D	0	0	D	1	1	D	0
0	0	1	0	0	0	D	1	1	D	0	D	1	0
0	0	1	1	0	0	D	0	0	D	1	D	0	0
0	1	0	0	0	0	D	0	D	1	0	0	D	0
0	1	0	1	0	0	D	1	D	0	1	1	D	0
0	1	1	0	0	0	D	1	D	0	0	D	1	0
0	1	1	1	1	1	D	0	D	1	0	D	1	0
1	0	0	0	0	D	1	1	1	D	0	0	D	0
1	0	0	1	0	D	1	0	0	D	1	1	D	1
1	0	1	0	D	D	D	D	D	D	D	D	D	D
1	0	1	1	D	D	D	D	D	D	D	D	D	D
1	1	0	0	D	D	D	D	D	D	D	D	D	D
1	1	0	1	D	D	D	D	D	D	D	D	D	D
1	1	1	0	D	D	D	D	D	D	D	D	D	D
1	1	1	1	D	D	D	D	D	D	D	D	D	D

Ora possiamo disegnare le mappe di Karnaugh

		AB			
		00	01	11	10
CX	00			d	d
	01			d	d
	11		1	d	d
	10			d	d

$J_A = BCX$

		AB			
		00	01	11	10
CX	00	d	d	d	1
	01	d	d	d	1
	11	d	d	d	d
	10	d	d	d	d

$K_A = 1$

		AB			
CX		00	01	11	10
	00		d	d	1
	01		d	d	
	11		d	d	d
	10	1	d	d	d

$$J_B = C\bar{X} + A\bar{X}$$

		AB			
CX		00	01	11	10
	00	d	1	d	d
	01	d		d	d
	11	d	1	d	d
	10	d		d	d

$$K_B = \bar{C} \cdot \bar{X} + CX$$

		AB			
CX		00	01	11	10
	00			d	
	01	1	1	d	1
	11	d	d	d	d
	10	d	d	d	d

$$J_C = X$$

		AB			
CX		00	01	11	10
	00	d	d	d	d
	01	d	d	d	d
	11		1	d	d
	10	1	1	d	d

$$K_C = B + \bar{X}$$

Infine, per quanto riguarda l'uscita Z:

$$Z = A \cdot \bar{B} \cdot \bar{C} \cdot X$$

Volendo utilizzare anche i don't care:

		AB			
CX		00	01	11	10
	00			d	
	01			d	1
	11			d	d
	10			d	d

$$Z = AX$$

ESERCIZIO 2

1) Per indirizzare 256 Mbyte occorre un indirizzo di 28 bit. Per indirizzare il singolo byte all'interno di un blocco occorrono 4 bit ($16 = 2^4$), che coincidono con i 4 bit meno significativi dell'indirizzo di memoria primaria. I restanti 24 bit costituiscono l'indirizzo del "block frame". Per indirizzare la cache, il "block frame" viene interpretato diversamente a seconda che l'indirizzamento sia di tipo "diretto" o "associativo su insiemi".

(a) Indirizzamento diretto. In questo caso devo poter indirizzare ciascuno dei 32K blocchi contenuti nella cache ($512\text{Kbyte}/(16\text{byte}/\text{blocco})$). Occorrono 15 bit che coincidono con i 15 bit meno significativi del "block frame". Pertanto i 28 bit di indirizzo della memoria primaria vengono interpretati come:

tag	cache index	offset
9 bit	15 bit	4 bit

(b) Indirizzamento "associativo su insiemi". In questo caso devo poter indirizzare ciascuno dei 16 insiemi in cui sono suddivisi i blocchi contenuti nella cache. Occorrono 14 bit che coincidono con i 14 bit meno significativi del "block frame". Pertanto i 28 bit di indirizzo della memoria primaria vengono interpretati come:

tag	cache index	offset
10 bit	14 bit	4 bit

2) La memoria è divisa in blocchi di 16 byte ciascuno in modo che la richiesta di un dato non presente in cache causa il trasferimento del blocco a cui appartiene il dato richiesto dalla memoria principale alla cache. Nel caso proposto i dati richiesti sono così suddivisi (N.B. per semplicità i valori di index sono riportati in formato decimale, mentre gli indirizzi in esadecimale):

indirizzi $0000000 \div 000000F \Rightarrow$ index 0; indirizzi $0000010 \div 000001F \Rightarrow$ index 1; ... ; $00003F0 \div 00003FF \Rightarrow$ index 63; $0380000 \div 038000F \Rightarrow$ index 0; ... ; $03803F0 \div 03803FF \Rightarrow$ index 63.

(N.B. in questa sequenza l'index è identico nel caso di indirizzamento diretto e associativo su insiemi). La prima volta che viene richiesta la parola 0000000 avremo un "cache miss" che provoca il caricamento del blocco 0 nella linea di cache di indirizzo 0. Le successive richieste dei dati di indirizzo $0000001 \div 000000F$ vengono quindi soddisfatte dalla cache ("cache hit"). Analogamente avviene per tutti i blocchi fino al blocco 63, che vengono allocati in linee consecutive della cache fino alla 63. Quando viene richiesta la parola 0380000 (index 0), nel caso di indirizzamento diretto questa sovrascrive il blocco $0000000 \div 000000F$ e così via per tutte le richieste consecutive.

Nel caso di indirizzamento associativo su insiemi a due vie uno stesso index corrisponde a un insieme di due blocchi. Pertanto le parole da 0380000 a $03803FF$ non sovrascrivono quelle precedentemente inserite, ma vengono memorizzate nel secondo blocco disponibile in ciascun insieme memorizzato. Nei cicli successivi al primo il processore richiede nuovamente tutti i dati, a partire da 0. Nel caso di indirizzamento diretto avremo un miss per il caricamento della prima parola di ciascun blocco e 15 hit per le parole dello stesso blocco, per tutti i 128 blocchi. Nel caso di indirizzamento associativo su insiemi si hanno solo hit perché tutti i blocchi sono presenti in cache. In sintesi:

Nel caso di indirizzamento diretto avremo per ciascun ciclo 128 miss e $15 \cdot 128$ hit. L'hit ratio H della gerarchia di memoria risulta pertanto pari a:

$$H = \text{cachehit} / \text{richieste totali} = (10 \cdot 15 \cdot 128) / (10 \cdot 16 \cdot 128) = 0.9375$$

Nel caso di indirizzamento associativo su insiemi a due vie avremo nel primo ciclo 128 miss e $15 \cdot 128$ hit, mentre nei 9 cicli successivi avremo $16 \cdot 128$ hit. L'hit ratio H della gerarchia di memoria risulta pertanto pari a:

$$H = \text{cachehit} / \text{richieste totali} = (15 \cdot 128 + 9 \cdot 16 \cdot 128) / (10 \cdot 16 \cdot 128) = 0.99375$$

ESERCIZIO 4

$\$9 \leftarrow v[i]; \$10 \leftarrow v[j]$

```
elabora:    addi $29, $29, -16
            sw $9, 0($29)
            sw $10, 4($29)
            sw $5, 8($29)
            sw $6, 12($29)
            muli $5, $5, 4
            add $5, $5, $4      #indirizzo di v[i] in $5
            muli $6, $6, 4
            add $6, $6, $4      #indirizzo di v[j] in $6
            lw $9, 0($5)
            lw $10, 0($6)
            slt $7, $9, $10     #p=v[i]<[v[j]
            beq $7, $0, exit     #if v[i]>=v[j] exit
            sw $9, 0($6)        #else swap
            sw $10, 0($5)
exit:        lw $9, 0($29)
            lw $10, 4($29)
            lw $5, 8($29)
            lw $6, 12($29)
            addi $29, $29, 16
            jr $31
```

ESERCIZIO 5

1. Nel caso di trasferimento mediante I/O da programma, per trasferire una parola occorrono 4 istruzioni. La CPU è impegnata per l'85% del tempo a eseguire istruzioni che non coinvolgono l'I/O, dunque può usare solo il 15% del tempo per eseguire istruzioni di trasferimento dati con periferiche. In termini di istr./sec questo tempo è pari a $0.15 \times 10^5 \text{ istr./s} = 1.5 \times 10^4 \text{ istr./s}$. Dal momento che per trasferire una parola servono due istruzioni, la velocità di trasferimento è pari a:
 $1.5 \times 10^4 \text{ istr./s} / (4 \text{ istr./parola}) = 3750 \text{ parole/s}$. La dimensione di una parola è pari a 32 bit (4 byte), da cui si ricava la velocità di trasferimento di **14.65 kB/s**.
2. 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 15% 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 l'85% del tempo posso trasferire due parole/istr.:
 $0.85 \times 2 \text{ parole/istr} \times 10^5 \text{ istr./s} = 1.7 \times 10^5 \text{ parole/s}$
Nel restante 15% del tempo posso trasferire 5 parole/istr.:
 $0.15 \times 5 \text{ parole/istr.} \times 10^5 \text{ istr./s} = 0.75 \times 10^5 \text{ parole/s}$
In **totale**, nel caso di trasferimento con DMA la velocità totale di trasferimento è pari a: **$(1.7 + 0.75) \times 10^5 \text{ parole/s} = 2.45 \times 10^5 \text{ parole/s} = 239 \text{ kB/s}$** .
3. Il 'controller' DMA esegue le operazioni di generazione indirizzi e conteggio dati trasferiti usando registri interni al controller oltre, ovviamente, a trasferire i dati dalla periferica alla memoria.