

SOLUZIONI DELLA PROVA SCRITTA DEL CORSO DI CALCOLATORI ELETTRONICI

24 Febbraio 2000

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

ESERCIZIO 1 (6 punti)

Si consideri una rete sequenziale avente un ingresso x e una uscita z . L'uscita $z = 1$ quando viene riconosciuta la sottosequenza 0100. Negli altri casi l'uscita $z = 0$.

- Disegnare il diagramma degli stati (1 punto)
- Codificare gli stati e scrivere la tabella di flusso. Si scriva poi la tabella delle transizioni qualora si usino flip-flop di tipo JK. (2 punti)
- Calcolare le forme minime per le variabili di eccitazione dei flip-flop e per l'uscita impiegando le mappe di Karnaugh. (3 punti)

Soluzione

Grafo degli stati

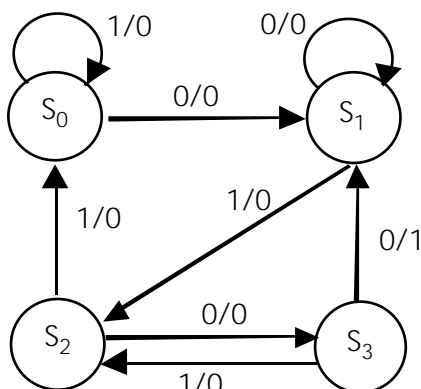


Tabella di flusso

Stato iniziale	Stato finale/uscita	
	$x = 0$	$x = 1$
S_0	$S_1/0$	$S_0/0$
S_1	$S_1/0$	$S_2/0$
S_2	$S_3/0$	$S_0/0$
S_3	$S_1/1$	$S_2/0$

Codifica degli stati

(2 bit - $Y_1 Y_0$)

$S_0 \rightarrow 00$ $S_1 \rightarrow 01$ $S_2 \rightarrow 10$ $S_3 \rightarrow 11$

Tabella di eccitazione di un flip-flop JK

$Q(t)$	$Q(t+1)$	J	K
0	0	0	d
0	1	1	d
1	0	d	1
1	1	d	0

Tabella delle transizioni (FF-JK)

Y' indica lo stato futuro

Y_1	Y_0	x	Y'_1	J_1	K_1	Y'_0	J_0	K_0	z
0	0	0	0	0	d	1	1	d	0
0	0	1	0	0	d	0	0	d	0
0	1	0	0	0	d	1	d	0	0
0	1	1	1	1	d	0	d	1	0
1	0	0	1	d	0	1	1	d	0
1	0	1	0	d	1	0	0	d	0
1	1	0	0	d	1	1	d	0	1
1	1	1	1	d	0	0	d	1	0

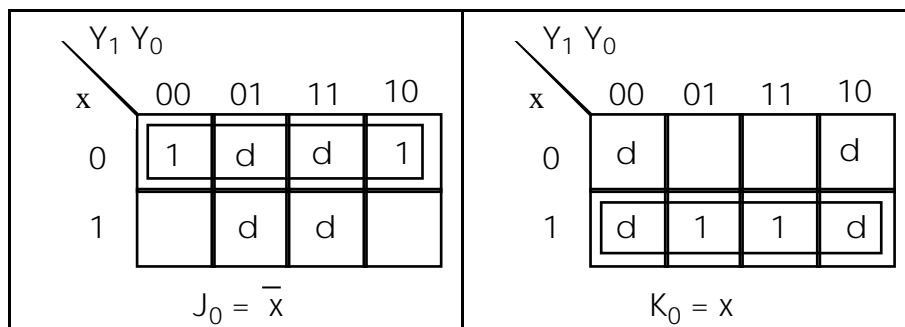
Uscita: $z = Y_1 Y_0 \bar{x}$

$Y_1 Y_0$		x			
		00	01	11	10
x	0			d	d
	1		1	d	d

$J_1 = Y_0 x$

$Y_1 Y_0$		x			
		00	01	11	10
x	0	d	d	1	
	1	d	d		1

$K_1 = \bar{Y}_0 x + Y_0 \bar{x}$



ESERCIZIO 2 (7 punti)

Si considerino i numeri 11001.010001 e 101.101 rappresentati in virgola fissa.

- 1) Rappresentare i due numeri nel formato in virgola mobile IEEE 754 (32 bit, con mantissa frazionaria e normalizzata rappresentata in segno e valore e esponente a 8 bit in eccesso 127). (1 punto)
- 2) Sommare i due numeri rappresentati in virgola mobile seguendo i passi dell'algoritmo usato nei calcolatori (*senza dimenticare il bit implicito!*). (4 punti)
- 3) Descrivere gli elementi di una ALU che effettua somme fra numeri in virgola mobile. (2 punti)

Soluzione

1)

11001.010001 (esponente +4)

Segno	Esponente	Mantissa
Mantissa		
0	10000011	100101000100000000000000

101.101 (esponente +2)

Segno	Esponente	Mantissa
Mantissa		
0	10000001	011010000000000000000000

2) Algoritmo di somma:

Confronto esponenti: il numero 11001.010001 ha esponente maggiore, dunque il secondo numero deve essere denormalizzato in modo da eguagliare gli esponenti.

Allineamento mantisse

Per uguagliare gli esponenti, la mantissa del secondo numero va fatta scorrere di due posizioni a destra (divisione per quattro). Pertanto le due mantisse (con esponente comune uguale a 4) diventano:

Primo numero: (1).1001010001 Secondo numero: (0).0101101

dove, fra parentesi, è stato evidenziato il valore del bit implicito

Per sommare le mantisse tenendo conto anche del bit implicito entrambi i numeri devono essere ulteriormente divisi per due (scorrimento a destra di una posizione):

Primo numero: 0.11001010001 Secondo numero: 0.00101101

A questo punto l'esponente comune dei due numeri è +5.

Somma delle mantisse:

$0.11001010001 + 0.00101101 = 0.1110111001$

Questa mantissa non è normalizzata. Per normalizzarla nel formato IEEE 754 occorre far scorrere la mantissa di una posizione verso sinistra e diminuire l'esponente di una unità. Pertanto il risultato della somma verrà rappresentato nel calcolatore come (l'esponente è ora +4):

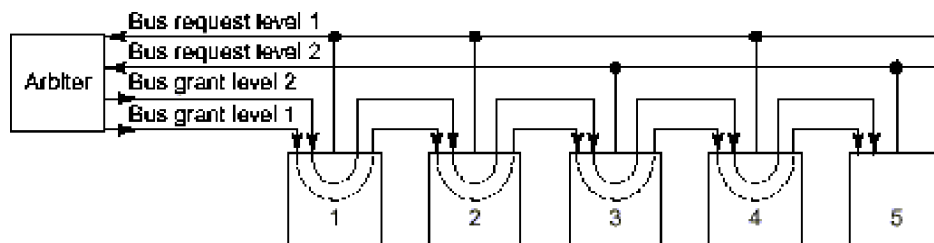
Segno	Esponente	Mantissa
Mantissa		
0	10000011	111011100100000000000000

- 3) La "floating point unit" (FPU) della ALU di un calcolatore può essere realizzata usando due unità aritmetiche in virgola fissa, una per l'esponente e una per la mantissa, che vengono accoppiate. L'unità per la mantissa si occupa di eseguire le operazioni aritmetiche di base sulla mantissa (somma algebrica, moltiplicazione e divisione). L'unità per l'esponente è più semplice in quanto deve eseguire solo operazioni di somma algebrica e di confronto fra numeri interi. Il confronto può essere effettuato attraverso una sottrazione degli esponenti.

Per eseguire la somma di due numeri in FP si fa la differenza degli esponenti. Il segno del risultato indica quale dei due esponenti è il più piccolo mentre il modulo indica il numero di scorrimenti verso destra che devono essere eseguiti sulla mantissa del numero più piccolo. Il registro che contiene la mantissa deve dunque essere del tipo a scorrimento. Gli scorrimenti possono essere pilotati da un contatore caricato con la differenza fra gli esponenti: ad ogni scorrimento il contatore viene decrementato finché non viene raggiunto il valore zero.

ESERCIZIO 3 (7 punti)

Si consideri lo schema di arbitraggio del bus rappresentato in figura. Le linee di "bus request" e "bus grant" a 2 livelli vengono impiegate per gestire una diversa priorità delle periferiche.



- 1) A quali schemi, fra quelli studiati nel corso, si ispira quello mostrato in figura? Spiegare come può essere gestita in questo caso la gerarchia di priorità fra le periferiche. (4 punti)
- 2) Illustrare la sequenza di segnali di controllo che vengono scambiati fra periferica e arbitro del bus prima di iniziare un trasferimento di dati attraverso il bus. (3 punti)

Soluzione

- 1) Lo schema in figura è un ibrido fra la tecnica di arbitraggio a richieste indipendenti e quella con collegamenti in catena "daisy chaining". Rispetto ai livelli 1 e 2 le periferiche sono divise in due gruppi indipendenti. All'interno di ciascun gruppo le periferiche sono collegate a catena. Pertanto vi è una gerarchia di priorità gestita direttamente dall'arbitro del bus che, in caso di richieste contemporanee da parte dei due livelli, può essere programmato per servire l'una o l'altra richiesta. All'interno di ciascun gruppo vi è un'ulteriore gerarchia di priorità "cablata", data dalla successione delle periferiche nella catena: ottiene il bus la prima periferica della catena, fra quelle che hanno fatto richiesta, che riceve il grant.
- 2) Una periferica che debba trasmettere dati sul bus ne fa richiesta all'arbitro attraverso la linea di bus request. L'arbitro del bus risponde a questa richiesta inviando un segnale di bus grant sulla linea appropriata. Il bus grant attraversa serialmente le periferiche (nota: il grant relativo a un certo livello di priorità viene fatto passare da periferiche con livello di priorità diverso) e viene bloccato dalla prima periferica che aveva fatto richiesta del bus. A questo punto la periferica invia il segnale di bus busy, per indicare che il bus è impegnato e inizia il trasferimento dei dati.

ESERCIZIO 4 (7 punti)

Si consideri un calcolatore che dispone di una memoria cache di 32 byte. La metodo di indirizzamento della cache sia "associativo su insiemi", e ciascun insieme contenga due blocchi. L'indirizzamento usato è a 8 bit, è possibile accedere al singolo byte e la memoria sia suddivisa in blocchi da 4 byte.

1. Spiegare come vengono interpretati gli indirizzi logici a 8 bit per recuperare l'informazione contenuta nella cache. (3 punti)
2. Sia data la seguente sequenza di accessi in memoria primaria, in termini di indirizzi esadecimali delle parole della memoria primaria (il primo indirizzo sia 0):

21, 04, BF, 16, 2B, 68, 2C, 10, 72, 5A, 07, 2D

Per ciascuna richiesta dire se c'è stato un **cache miss** o un **cache hit**, mostrando il contenuto della cache. (N.B.: si ipotizzi la cache inizialmente vuota). Usare la tecnica di rimpiazzamento dei blocchi nella cache FIFO. (4 punti)

Soluzione:

1. I due bit meno significativi vengono usati per indirizzare il singolo byte all'interno di un blocco. I restanti 6 bit costituiscono il "block offset". La cache di 32 byte con insiemi da due blocchi (8 byte) contiene esattamente 4 insiemi. Pertanto l'insieme nella cache viene indirizzato dai 2 bit meno significativi del "block offset". Una volta individuato l'insieme, si cerca la parola in uno dei due blocchi contenuti in ciascun insieme confrontando il "tag", formato dai 4 bit più significativi. Se la parola non è presente la si trasferisce dalla memoria primaria e, se l'insieme è pieno, si sostituisce un blocco usando una delle politiche FIFO, LRU o LFU.

2. 21 ==> 2 | 00 | 01 ==> insieme 0 (M); 04 ==> 0 | 01 | 00 ==> insieme 1 (M)
BF ==> B | 11 | 11 ==> insieme 3 (M); 16 ==> 1 | 01 | 10 ==> insieme 1 (M)
2B ==> 2 | 10 | 11 ==> insieme 2 (M); 68 ==> 6 | 10 | 00 ==> insieme 2 (M)
2C ==> 2 | 11 | 00 ==> insieme 3 (M); 10 ==> 1 | 00 | 00 ==> insieme 0 (M)
72 ==> 7 | 00 | 10 ==> insieme 0 (M); 5A ==> 5 | 10 | 10 ==> insieme 2 (M)
07 ==> 0 | 01 | 11 ==> insieme 1 (H); 2D ==> 2 | 11 | 01 ==> insieme 3 (H)

(M) Cache Miss (H) Cache Hit

Contenuto finale della cache

	Insieme 0	Insieme 1	Insieme 2	Insieme 3
Blocco 0	70 ÷ 73	04 ÷ 07	58 ÷ 5B	BC ÷ BF
Blocco 1	10 ÷ 13	14 ÷ 17	68 ÷ 6B	2C ÷ 2F

ESERCIZIO 5 (6 punti)

Sia data la seguente lista di processi (si supponga che l'istante iniziale sia 0):

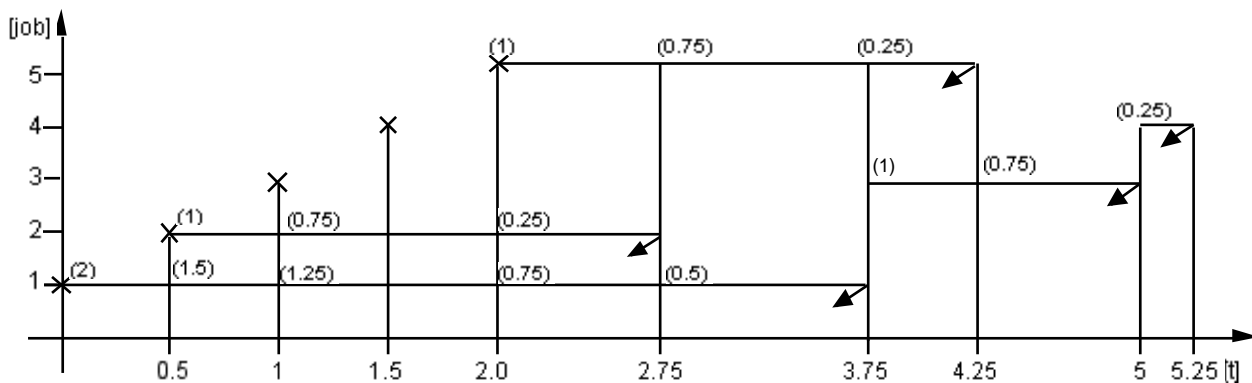
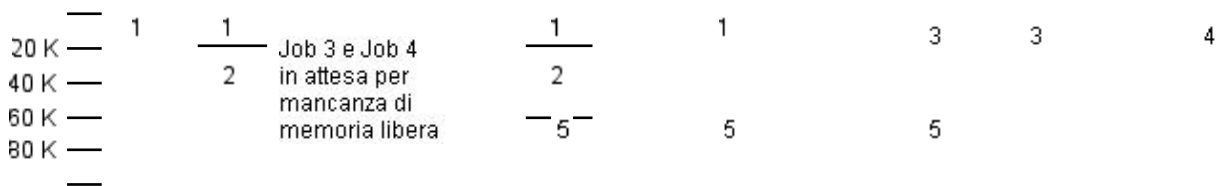
Job	Tempo di Arrivo	Tempo di CPU Richiesto	Richiesta di Memoria
1	0.0	2.0	20K
2	0.5	1.0	40K
3	1.0	1.0	50K
4	1.5	0.25	60K
5	2.0	1.0	20K

Il sistema ha 100K di memoria disponibile, e viene gestita con partizioni dinamiche non rilocabili.

- 1) Mostrare, utilizzando il metodo grafico, la sequenza di esecuzione dei job qualora si impieghi la politica di scheduling FIFO multiprogrammata "round robin" (3 punti)
- 2) Indicare, negli istanti in cui un job va in esecuzione e termina, la partizioni di memoria occupate da ciascun job e le partizioni libere, giustificando l'eventuale "messa in attesa" dei job (2 punti).

3) Calcolare il tempo di *turnaround* medio e il tempo di *turnaround* pesato medio (1 punto).

Soluzione:



Job	t_{arrivo}	t_{start}	t_{finish}	Turnaround time	Weighted Turnaround time
1	0	0	3.75	3.75	1.875
2	0.5	0.5	2.75	2.25	2.25
3	1	3.75	5	4	4
4	1.5	5	5.25	3.75	15
5	2	2	4.25	2.25	2.25
Media				3.2	5.075