

**PROVA SCRITTA DEL MODULO INTEGRATO E DEL CORSO DI
CALCOLATORI ELETTRONICI**

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA – 5/7 CFU
CORSO DI LAUREA IN INGEGNERIA ELETTRICA ED ELETTRONICA, INGEGNERIA BIOMEDICA - 6 CFU
27 gennaio 2016

NOME:

COGNOME:

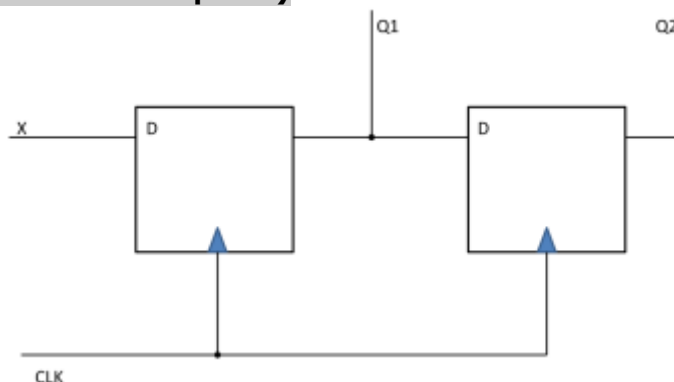
MATRICOLA:

CFU:

ESERCIZIO 1 – RETI LOGICHE (10 punti – 7 cfu: 8 punti)

1) (5 punti – 7 cfu: 4 punti)

Ricavare un flip flop JK da un flip flop T. Definire la tabella delle transizioni ed ottenere le forme minime delle espressioni della rete di transizione dello stato mediante le mappe di Karnaugh. Disegnare infine il circuito ottenuto.



2) (5 punti - 7 cfu: 4 punti) Si consideri la rete logica disegnata nella figura a fianco. Indicare qual è la funzione svolta dalla rete in funzione dell'ingresso X e scrivere il diagramma degli stati. CLK è il segnale di sincronismo.

ESERCIZIO 2 – UNITA' DI MEMORIA (8 punti – 7 cfu: 7 punti)

Si consideri un'unità di memoria primaria da 64 KB, ed una memoria cache da 32 B. Una parola è costituita da 1 B singolarmente indirizzabile. La memoria è organizzata in blocchi da 8 parole.

- 1) (1 punto) Spiegare, precisando bene il significato e la funzione dei diversi campi, come vengono interpretati gli indirizzi logici per recuperare l'informazione contenuta nella cache nel caso che il metodo di indirizzamento sia diretto.
- 2) (5 punti) Individuare in quali linee della memoria cache sono memorizzati i blocchi di primaria caratterizzati dai seguenti valori di *block frame*: 0, 8, 1024, 4096, 8191, 8192. Indicare chiaramente gli indirizzi delle parole di primaria che presentano tali block frame.
- 3) (2 punti – 7 cfu: 1 punto) Sapendo che l'hit ratio di cache è pari a 0.8 e che il tempo di accesso alla memoria primaria è pari a 50 ns, qual è il massimo tempo di accesso alla cache consentito per ottenere un tempo medio minore od uguale a 15 ns, sapendo che?

ESERCIZIO 3 – LINGUAGGIO ASSEMBLY MIPS (8 punti)

- 1) (5 punti) Durante l'esame di Calcolatori Elettronici, nel tradurre il codice C in basso a sinistra nel codice Assembly MIPS a destra, uno studente distratto ha commesso cinque errori che devono essere indicati e corretti. Il docente del corso vi ha incaricato di correggere tali errori, chiedendovi inoltre di spiegare il significato associato a ciascun registro utilizzato.

<pre>for(i=0; i<n; i++) v[i] = v[i] + h;</pre>	<pre>move \$8, \$0 while: beq \$8, \$5, exit lw \$7, v(\$9) add \$7, \$7, \$4 sw \$7, v(\$8) subi \$8, \$8, 1 move \$8, \$0 jal while exit: ...</pre>
---	--

- 2) (3 punti) Caricare nel registro \$10 il valore costante 65538_{10} utilizzando un'apposita sequenza di istruzioni Assembly MIPS.

ESERCIZIO 4 – UNITA' LOGICO-ARITMETICA (7 punti)

Siano dati 12 bit per la rappresentazione di valori numerici in virgola fissa ed in virgola mobile. Per la prima, si disponga di un campo di 7 bit per la parte intera e 4 per la parte frazionaria. Per la seconda, si consideri una mantissa M frazionaria e normalizzata in segno e valore con modalità 1.M (bit di parte intera implicito), esponente a 6 bit in eccesso 32, e bit di segno.

- 1) (3 punti) Spiegando bene ogni passo del ragionamento, indicare il minimo ed il massimo numero rappresentabili, in valore assoluto ed escluso lo zero, nei due casi.
- 2) (2 punti) Rappresentare i valori 38.75 e 11.25, qui espressi in notazione decimale, nella notazione in virgola mobile indicata nel testo, indicando chiaramente i valori dei bit nei campi segno, esponente e mantissa e quantificando l'eventuale perdita di precisione nell'operazione.
- 3) (2 punti) Sommare i valori al punto 2, secondo l'algoritmo di somma utilizzato nei calcolatori elettronici, spiegando nel dettaglio ogni passaggio intermedio e quantificando l'eventuale perdita di precisione nell'operazione.

ESERCIZIO 5 – SISTEMI OPERATIVI (solo 7cfu: 3 punti)

Descrivere in modo sintetico e chiaro cosa si intende per paginazione della memoria.

ESERCIZIO 1 – RETI LOGICHE (10 punti – 7 cfu: 8 punti)

1)

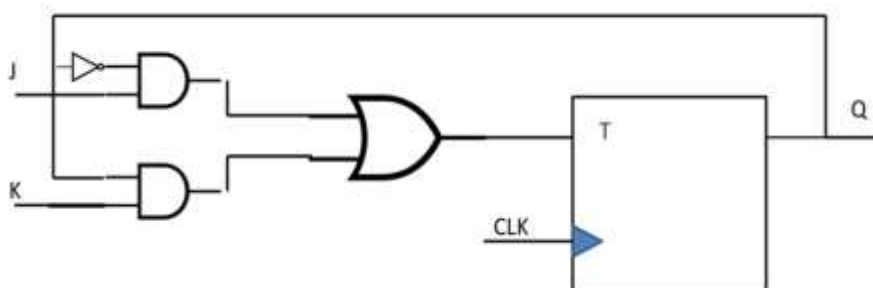
Per ricavare un FF-JK è sufficiente scrivere la tabella delle transizioni. La funzione di transizione dello stato Q' fornirà il valore in ingresso al FF-D.

J	K	Q	Q'	T
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	1

Da cui:

JK Q				
	00	01	11	10
0			1	1
1		1	1	

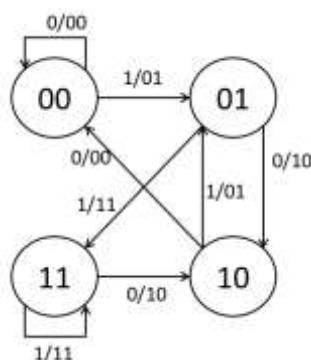
$$D = J\bar{Q} + KQ$$



2)

Per capire la funzionalità svolta dalla rete logica del riquadro è sufficiente osservare che, trattandosi di due semplici FF-D connessi in serie, il primo, tramite Q_1 , passerà all'altro il valore ricevuto all'istante t , con un colpo di clock di ritardo. In altre parole si tratta di un registro a scorrimento a due bit, pilotato dall'ingresso X.

Con facilità si può desumere il diagramma degli stati (in questo caso coincidenti con le uscite) che è il seguente (il bit a sinistra è Q_2 , quello a destra è Q_1):



ESERCIZIO 2 – UNITA' DI MEMORIA (8 punti – 7 cfu: 7 punti)

La memoria primaria è di 216 B, quindi indirizzata con 16 bit. La memoria cache è invece indirizzata con 5 bit. Risultando i blocchi di 8 B, ciascun byte è indirizzabile con 3 bit. Di conseguenza la cache è dotata di sole quattro linee. La formattazione dell'indirizzo di primaria è dunque dato da: < TAG 11 bit > < Cache index 2 bit > < Offset 3 bit >

Per capire in quali linee sono indirizzati i blocchi forniti è sufficiente riscrivere i block frame attraverso le potenze di due: 0, 2^3 , 2^{10} , 2^{12} , $2^{13}-1$, 2^{13} . Si evince chiaramente che i primi quattro block frame stanno tutti quanti nella stessa linea di cache (la prima per l'esattezza). Il quinto è allocato nell'ultima, mentre il terzo valore non rientra fra quelli ammissibili essendo rappresentato con 14 bit mentre come si vede il block frame è fatto di soli 13 bit. Una verifica è fatta dividendo ciascun block frame per 4, e considerando il resto della divisione come valore di cache index.

Gli indirizzi delle parole si ottengono dalle formule : $BF * Dim\text{-}Blocco$, con la quale si ha l'indirizzo della prima parola del blocco, e $BF * Dim\text{Blocco} + Dim\text{Blocco} - 1$, con la quale si ha l'indirizzo dell'ultima parola del blocco. Si ottiene dunque lo schema :

Block Frame	Indirizzo prima parola	Indirizzo ultima parola
0	0	7
8	64	71
1024	8192	8199
4096	2^{15}	$2^{15}+7$
8191	65528	65535 (pari a $2^{16}-1$)

Per il calcolo del massimo tempo medio si applica la disuguaglianza:

$$T_{max} \geq T_c + (1 - H_c) \cdot T_p$$

Sapendo che: $H_c = 0.8$; $T_p = 50$; $T_{max} = 15$, si ha:

$$15 \geq T_c + (1 - 0.8) \cdot 50; T_c \leq 15 - 10 = 5$$

Il tempo di cache massimo sarà dunque pari a 5 ns.

ESERCIZIO 3 – LINGUAGGIO ASSEMBLY MIPS (8 punti)

1)

Nel codice \$8 sembra essere un contatore, ma non è utilizzato correttamente come tale; il registro \$7 contiene v[i] nelle intenzioni ma non nella realtà, in quanto l'offset in \$9 non è definito. \$5 è evidentemente pari ad n.

Evidenziamo in rosso gli errori:

```
        move $8, $0
while:  beq $8, $5, exit
        lw $7, v($9)           # cosa contiene $9??
                                   #Probabilmente manca un'istruzione

        add $7, $7, $4
        sw $7, v($8)           #attenzione: indirizzamento a gruppi di 4 byte!
        subi $8, $8, 1          #i=i-1???
move $8, $0                  #istruzione fuori luogo
jatl while                   #salto a funzione while inesistente
exit:   ...
```

Correggiamo il codice di conseguenza:

```
        move $8, $0
while:  beq $8, $5, exit
        muli $9, $8, 4          #istruzione mancante aggiunta
        lw $7, v($9)
        add $7, $7, $4
        sw $7, v($9)           #utilizzo del registro corretto
        addi $8, $8, 1          #istruzione corretta
        j while                 #istruzione corretta
exit:   ...
```

2)

Per caricare il valore 65538 in \$10, basta riscriverlo come $2^{16}+2$. Non è dunque possibile utilizzare una semplice istruzione `addi` in quanto esso richiede più di 16 bit per la sua rappresentazione. Dobbiamo quindi utilizzare l'istruzione `lui`, considerando i sedici bit più significativi e i sedici bit meno significativi della costante separatamente.

Dalla struttura del valore, si evince subito che il raggruppamento relativo alla parte più significativa corrisponde al valore 1 in decimale. Infatti 2^{16} presenta soltanto il primo bit più significativo, il diciassettesimo, pari ad 1. Ad esso sarà sufficiente aggiungere il valore 2 relativo alla parte meno significativa. Per cui:

```
addi $10, $0, 2    #carica 2 nel registro - parte meno significativa
lui $10, 1         #carica 1 nella parte più significativa del registro
```

è la sequenza di istruzioni richiesta.

ESERCIZIO 4 – UNITA' LOGICO-ARITMETICA (7 punti)

1)

Per quanto riguarda i campi in virgola fissa, si ha che il valore minimo è ottenuto in corrispondenza di tutti i bit di parte intera pari a zero e un unico 1 nella posizione meno significativa della parte frazionaria. Quindi il valore minimo sarà pari a 2^{-4} . Il valore massimo invece si ottiene ponendo ad 1 tutti i bit di parte frazionaria e intera, ottenendo quindi il valore $2^7 - 2^{-4}$.

Per quanto riguarda la rappresentazione in virgola mobile, abbiamo undici bit utili a parte il bit di segno, dei quali cinque per la mantissa e sei per l'esponente.

Il minimo valore rappresentabile si ottiene considerando il minimo valore dell'esponente, che essendo in eccesso 32, è, per definizione di eccesso k, pari a $-k$, ovvero -32 , e il minimo valore della mantissa, corrispondente ad un campo formato da tutti zeri poiché il bit implicito per definizione è sempre impostato ad 1. Il minimo valore è dunque pari a $1.0 \cdot 2^{-32}$. Il massimo valore rappresentabile si ottiene valutando il massimo esponente, che si ricava dalla formula 2^{n-1-k} , con n numero di bit usati per l'esponente. Il risultato è $64-1-32=31$. La massima mantissa, di cinque bit, si ottiene ponendo tutti i bit disponibili a 1, considerato anche il bit implicito, pari a $2-2^{-5}$. Il massimo valore rappresentabile è dunque $(2-2^{-5}) \cdot 2^{31}$, ovvero $2^{32}-2^{26}$.

2)

Per rappresentare i due valori 38.75 e 11.25 in virgola mobile, vanno innanzi tutto convertiti in binario mediante gli algoritmi della divisione successiva per la parte intera, e della moltiplicazione successiva per la parte frazionaria, ottenendo rispettivamente 100110.11 e 1011.01.

Non ci resta che normalizzare la mantissa ottenendo i relativi esponenti: $1.0011011 \cdot 2^5$ e $1.01101 \cdot 2^3$.

Gli esponenti vanno scritti con sei bit in eccesso 32. Per ottenere la rappresentazione in eccesso 32 è sufficiente sommare appunto l'eccesso al valore numerico da rappresentare, e poi convertirlo in binario. Si ha $5+32=37 \rightarrow 100101$ e $3+32=35 \rightarrow 100011$.

Quindi i valori sono rappresentati nel campo a 8 bit come segue:

Valore	S	Esponente						Mantissa				
38.75	0	1	0	0	1	0	1	0	0	1	1	0
11.25	0	1	0	0	0	1	1	0	1	1	0	1

Come si può notare lo spazio per la mantissa non è sufficiente per memorizzare il primo valore, che risulterà così troncato a 38. Il bit di segno è lo stesso per entrambi i valori, dal momento che sono positivi (non è rilevante se sia 0 od 1).

3)

L'algoritmo dei calcolatori prevede i seguenti passaggi: allineamento delle mantisse degli operandi, somma delle mantisse degli operandi, normalizzazione della mantissa del risultato ed eventuale modifica dell'esponente.

Allineamento delle mantisse: la differenza fra gli esponenti del primo e del secondo numero è 2. Si divide quindi il secondo valore, più piccolo, per 4, onde avere la mantissa allineata con il numero maggiore.

Somma delle mantisse: $1.00110 + 0.0101101 = 1.1000101$

Normalizzazione della mantissa e modifica dell'esponente, in questo caso non necessaria. Si ha comunque un'ulteriore perdita di precisione:

Valore	S	Esponente						Mantissa				
49.25?	0	1	0	0	1	0	1	1	0	0	0	1

La somma corrisponde al valore: $1.10001 \cdot 2^5 = 110001 = 49$.

ESERCIZIO 5 – SISTEMI OPERATIVI (solo 7cfu: 3 punti)

V. dispense del corso.