UNIVERSITÀ DEGLI STUDI DELL'INSUBRIA

ARCHITETTURE DEGLI ELABORATORI

Prova scritta del 12-1-2023

Cognome e nome Matricola

Esercizio 1. Sia F una funzione che riceve in ingresso un numero intero n rappresentato su 4 bit in complemento a 2. F assume valore 0 quando n vale -5,- 2, 1, 3, 4 o 6 e può assumere indifferentemente il valore 1 o 0 quando n vale 2, 5, o -6. F restituisce 1 per gli altri valori di n. Realizzare il circuito che implementa F usando le mappe di Karnaugh, sintetizzando in forma SoP. Riportare i passaggi e disegnare il circuito derivato.

Soluzione:

La funzione F è definita dalla seguente tabella di verità, che riporta per ogni valore di N rappresentabile su 4 bit in cp2, la relativa codifica, ed il valore restituito da F.

N	A	B	C	D	F
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	X
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	X
6	0	1	1	0	0
7	0	1	1	1	1
-8	1	0	0	0	1
-7	1	0	0	1	1
-6	1	0	1	0	X
-5	1	0	1	1	0
-4	1	1	0	0	1
-3	1	1	0	1	1
-2	1	1	1	0	0
-1	1	1	1	1	1

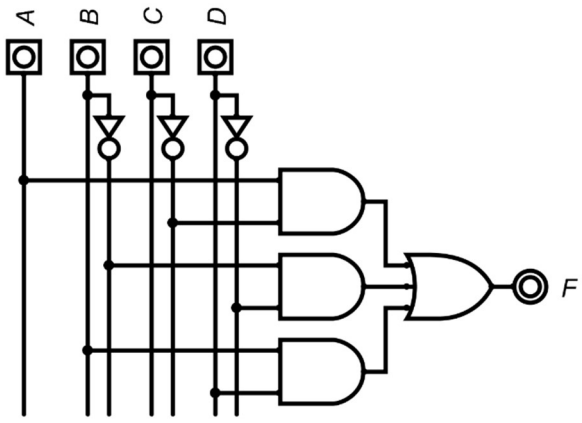
Dalla tabella si deriva la seguente mappa di Karnaugh:

F:

AB \ CD	00	01	11	10
00	1	0	0	X
01	0	X	1	0
11	1	1	1	0
10	1	1	0	X

Da cui: $F = A\overline{C} + \overline{BD} + BD$

Il circuito risulta pertanto:



Esercizio 2. Indicare la codifica floating point IEEE 754 in singola precisione del numero -104,3125. Si ricorda che l'esponente va rappresentato su 8bit e la mantissa su 23bit. Riportare il procedimento di calcolo seguito.

Soluzione:

Segno: - → bit segno =1

Modulo parte intera= 104 → 1101000

104	0
52	0
26	0
13	1
6	0
3	1
1	1
0	

Parte decimale: ,3125

	,3125
0	,625
1	,25
0	,5
1	,0

104,3125 → 1101000,0101 normalizzando otteniamo: $1,1010000101 * 2^6$

Rappresento l'esponente 6 in codice eccesso 127 su 8 bit.

6+127= 133 → 10000101

133	1
66	0
33	1
16	0
8	0
4	0
2	0
1	1
0	

Pertanto la codifica IEEE 754 risulta: 1 10000101 10100001010000000000000

Esercizio 3.

Progettare una ALU dotata degli ingressi A e B, e dell'uscita U, tutti su 4 bit, che in base ad un ingresso di selezione F gestisce le seguenti operazioni: A – B, A+B, o B-A, dove A e B sono interi con segno in cp2.

La ALU deve generare un bit di esito Z, che notifica se il risultato è uguale a 0. Trascurare gli eventuali overflow.

Soluzione:

Specifichiamo i valori di F per cui eseguire le varie operazioni

1. F=00: A-B
2. F=01: A + B
3. F=10 o F=11: B-A

Tutte le operazioni possono essere eseguite da un unico blocco sommatore, ma vanno ridefinite come somme:

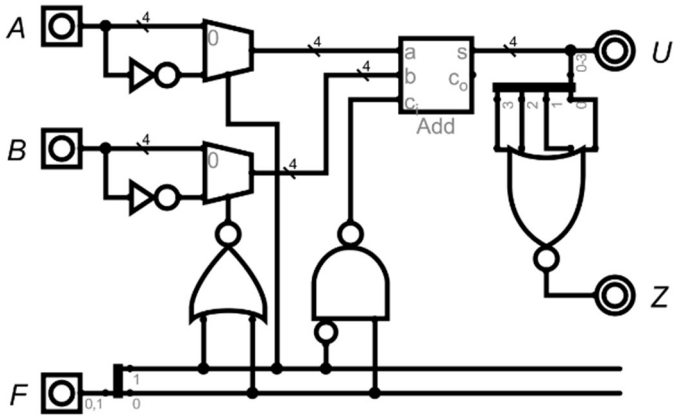
A-B è equivalente a A+cp1(B)+1, mentre B-A è equivalente a B+cp1(A)+1

E' necessario selezionare gli operandi in base ad F: il primo operando delle 3 operazioni risulta sempre A o B, mentre il secondo B, cp1(B), o cp1(A). Quindi sembrerebbero necessari 2 mux per la selezione, rispettivamente con 2 e 4 ingressi dati. Osservo però che B-A è equivalente a -A + B, ovvero a cp1(A)+B+1. Rappresentare

l'operazione in questo formato ci consente di impiegare 2 mux da 2 ingressi dati, che selezionano rispettivamente A o cp1(A), e B o cp1(B), in base ad F. La selezione di A o cp1(A) dipende solo da MSB(F), mentre la selezione di B o cp1(B) da entrambi i bit (viene selezionato B solo con F=00).

Per completare l'inversione di segno degli operandi, è necessario impostare a 1 il carry-in. Il carry-in vale 0 solo quando F vale 01, altrimenti vale 1.

Infine, per il calcolo del bit di esito è sufficiente una sola porta *nor* a 4 ingressi.



Esercizio 4.

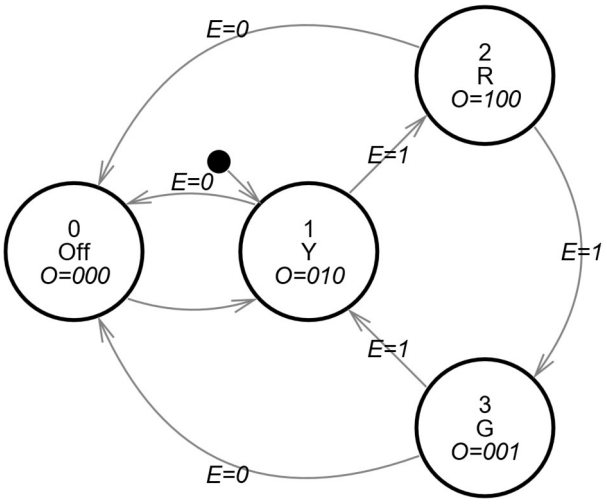
Progettare un circuito sequenziale sincrono che controlla un semaforo. Il circuito è dotato delle uscite O_2, O_1 e O_0 rispettivamente dedicate a gestire l'accensione e lo spegnimento della lanterna rossa, gialla e verde. Per semplicità ipotizziamo che le fasi di accensione abbiano tutte una lunghezza pari al periodo di clock.

Il circuito riceve un segnale E di attivazione. Quando E vale 1, ad ogni ciclo di clock si alternano periodicamente l'accensione del giallo, del rosso e del verde, mentre, quando E vale 0, si alterna il giallo acceso a tutte le luci spente. Si richiede di realizzare il circuito impiegando la codifica a singolo 1 (un solo bit per stato vale 1).

Per la realizzazione circuitale è necessario usare Flip-Flop D. Mostrare uno schema circuitale ad alto livello di astrazione, denotando come blocchi le reti combinatorie che compongono il circuito. Specificare le funzioni logiche implementate da tali reti.

Soluzione:

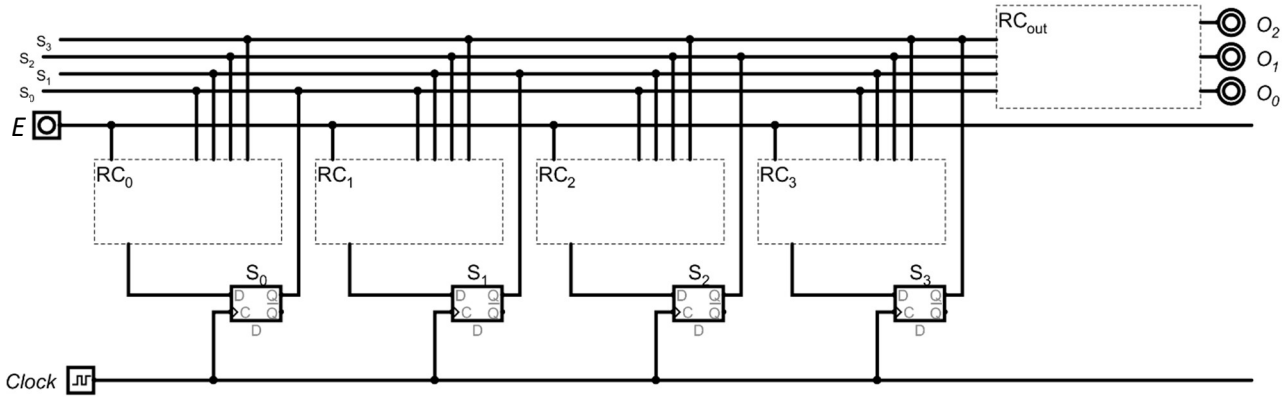
Il comportamento del semaforo è modellato dalla seguente macchina a stati. Ogni stato denota una configurazione delle lanterne semaforiche. Off: rappresenta lo stato in cui tutte le luci sono spente, mentre, Y, R, e G sono stati in cui risultano accese la luce gialla, rossa e verde. In ogni stato viene anche riferito il comando inoltrato sulle uscite O_2, O_1 e O_0 per accendere e spegnere le lanterne.



Con la codifica a singolo 1 gli stati possono essere rappresentati su 4 bit come segue:

Off: 0001, Y: 0010, R: 0100, e G: 1000

Lo schema di riferimento per la realizzazione circuitale è il seguente:



La rete combinatoria RC_{out} che calcola le uscite è definita partendo dalla seguente tabella di verità:

S_3	S_2	S_1	S_0	O_2	O_1	O_0
0	0	0	1	0	0	0
0	0	1	0	0	1	0
0	1	0	0	1	0	0
1	0	0	0	0	0	1

Sfruttando la codifica a singolo 1 abbiamo che:

$O_2 = S_2$

$O_1 = S_1$

$O_0 = S_3$

Le reti combinatorie che gestiscono le transizioni di stato sono:

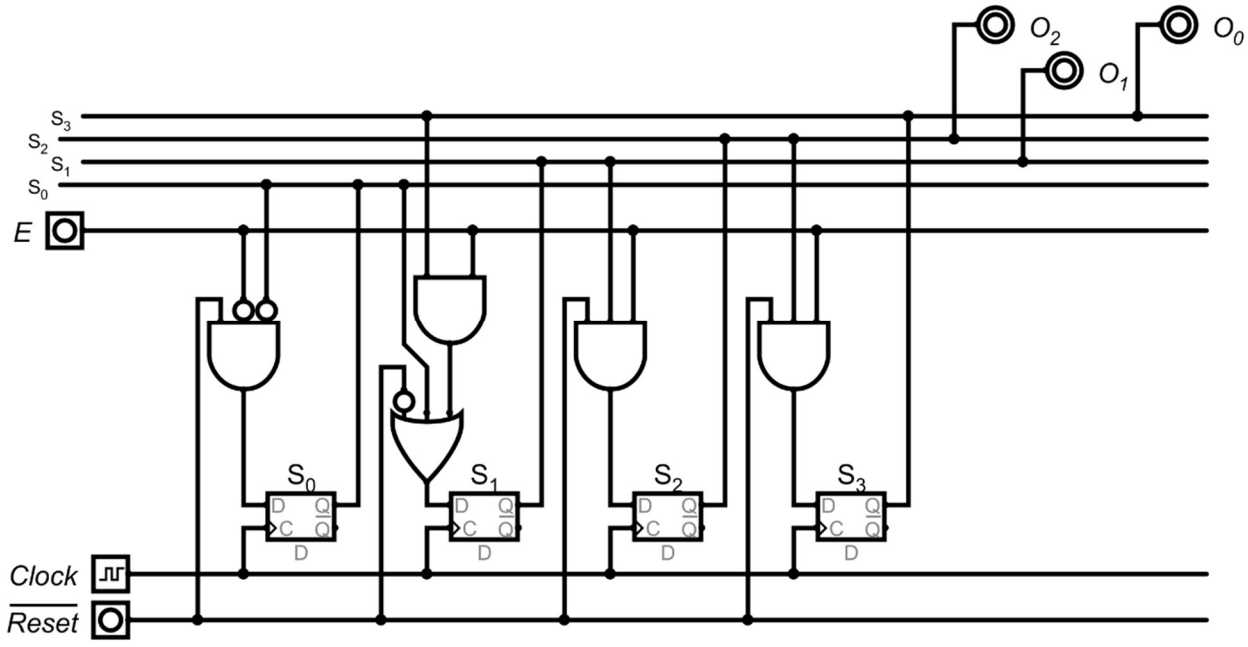
RC_0: S_0'=1 sse S_3 !E + S_1 !E+ S_2!E = !E (S_3 + S_1 + S_2) = !E ! S_0 (sfrutto la codifica a singolo 1)

RC_1: S_1'=1 sse S_0 + S_3 E

RC_2: S_2'=1 sse S_1 E

RC_3: S_3'=1 sse S_2 E

Infine, usiamo un segnale di Reset attivo basso per impostare lo stato iniziale:



Esercizio 5.

Si consideri un sistema di memoria composto da una memoria di lavoro di 4 GB, e da una cache di 512KB. Il sistema gestisce blocchi di 128 parole di 32 bit. Assumendo che la memoria sia indirizzata a livello di byte, si chiede di indicare la struttura degli indirizzi in scenari dove la cache impiegata è: a) una cache a indirizzamento diretto, b) una cache completamente associativa, e c) una cache set-associativa a 4 vie. In particolare, indicare quali bit di un indirizzo denotano i tag, quali gli indici di linea (se presenti), quali il byte offset nei blocchi. Riportare i procedimenti di calcolo seguiti.

Soluzione.

Memory size: 4GB = 2^{32} byte → indirizzamento su 32 bit

Block size: 128 * (32 / 8) byte= 512 byte → byte offset su 9 bit

Num blocchi in cache= 512KB / 512 byte= 2^{10} blocchi in cache →1024 blocchi

a. *Scenario a)*
   Con una cache a indirizzamento diretto abbiamo 1 blocco per linea, e 2^{10} blocchi in cache → indice di linea su 10 bit.
   Il tag è riferito su 32-(9+10)= 13bit
   Un indirizzo di 32 bit è strutturato in: 13 bit tag | 10 bit indice di linea | 9 bit di byte offset

b. *Scenario b)*
   Con una cache completamente associativa il tag è riferito su 23bit (ovvero 32-9)
   Un indirizzo di 32 bit è strutturato in: 23 bit tag | 9 bit di byte offset nel blocco

c. *Scenario c)*
   Con una cache set associativa a 4 vie abbiamo 4 blocchi per set. Poiché in cache sono presenti 2^{10} blocchi, avremo un numero di set pari a 2^{10} / 2^2 = 256→ indice di set su 8 bit.
   Il tag è riferito su 32-(9+8)= 15bit
   Un indirizzo di 32 bit è strutturato in: 15 bit tag | 8 bit indice di linea | 9 bit di byte offset nel blocco

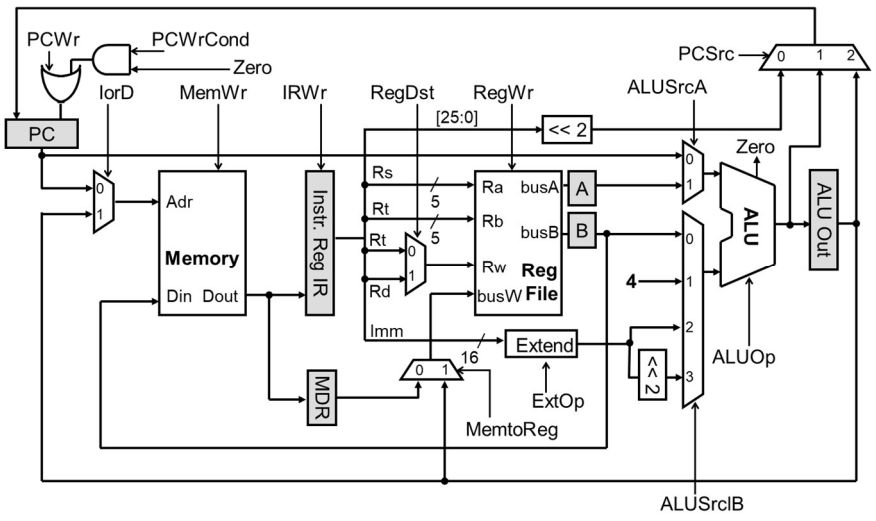
Esercizio 6.

Facendo riferimento al datapath multiciclo del MIPS in figura, descrivere brevemente le fasi del ciclo di esecuzione dell'istruzione: `bnq rt, rs, imm16.`

L'istruzione *bnq* (branch on not equal) verifica se il contenuto del registro *rs* è diverso da quello di *rt*.

Se tale condizione si verifica la prossima istruzione da eseguire è quella che è all'indirizzo PC+4+(SignExt(imm16)<<2).

Altrimenti, se la condizione non è verificata, la prossima istruzione da mandare in esecuzione sarà quella adiacente in memoria all'istruzione corrente. Per ogni fase indicare l'uso dei registri, motivando brevemente le risposte date.



Fase	**Trasferimenti tra registri**	**Motivo** *(breve spiegazione, max 3 righe)*
1)	IR ← Mem[PC] PC ← PC+4	…
2)	A ← R[IR[rs]] B ← R[IR[rt]] ALUOut=PC+(SignExt(IR[Im16])<<2)	…
3)	If (A-B != 0) PC=ALUOut	…
4)	--	
5)	--	