UNIVERSITÀ DEGLI STUDI DELL'INSUBRIA

ARCHITETTURE DEGLI ELABORATORI

20 Dicembre 2023

Esercitazione in aula

Esercizio 1.

Sia F una funzione che riceve in ingresso un numero intero n rappresentato su 4 bit in cp2. F assume valore 0 quando n vale -1, 3, 4, 6, 7 e può assumere indifferentemente il valore 1 o 0 quando n vale -8, -6, -5, -2 o 2. F restituisce 1 per gli altri valori di n.

a. Realizzare il circuito che implementa F usando le mappe di Karnaugh, sintetizzando in forma PoS. Riportare i passaggi e disegnare il circuito derivato.
b. Realizzare un circuito equivalente a quello derivato al punto a) usando solo porte NOR.

Soluzione:

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

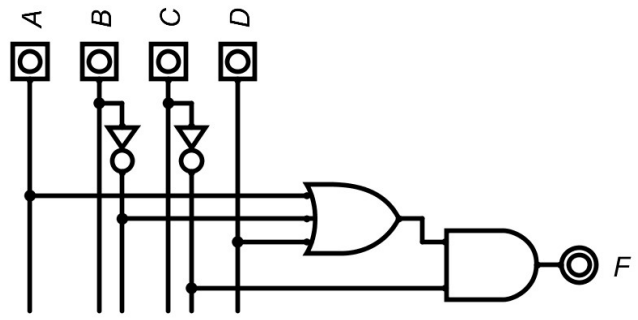
N	A	B	C	D	F
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	X
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	0
-8	1	0	0	0	X
-7	1	0	0	1	1
-6	1	0	1	0	X
-5	1	0	1	1	X
-4	1	1	0	0	1
-3	1	1	0	1	1
-2	1	1	1	0	X
-1	1	1	1	1	0

Dalla tabella si deriva la seguente mappa di Karnaugh:

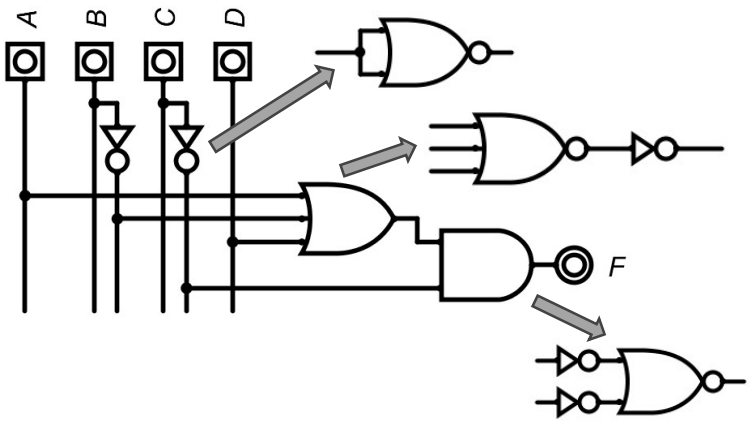
AB \ CD	00	01	11	10
00	1	1	0	X
01	0	1	0	0
11	1	1	0	X
10	X	1	X	X

$$F = \overline{C}\left(A + \overline{B} + D\right)$$

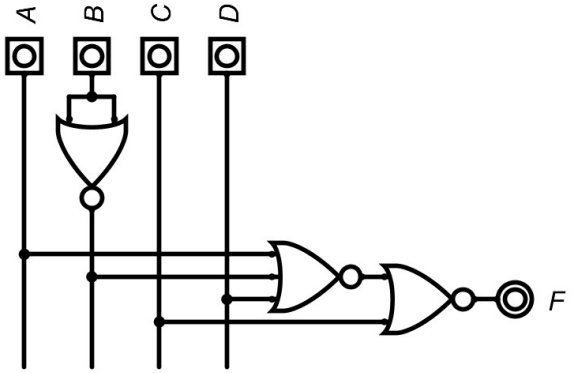
Il circuito che implementa F è il seguente:



Sostituiamo le porte AND, OR e NOT con composizioni di porte NOR che implementano il prodotto logico, la somma logica e l'inversione.



Dopo aver rimosso le doppie negazioni otteniamo il seguente circuito implementato usando solo porte NOR.



Esercizio 2.

Sia 11010011 la rappresentazione binaria di un numero intero N. Indicare il valore in base 10 di N assumendo che N sia rappresentato: a) in modulo e segno, b) in complemento a 2, c) in codice eccesso 2^{k-1} con K=8. Riportare il procedimento seguito per derivare ogni valore.

Soluzione:

a) Il MSB a 1 denota il segno negativo di N, mentre il resto dei bit rappresentano 83, il valore del modulo. Complessivamente, in modulo e segno N vale -83.
b) Il MSB ha peso negativo: -128, il resto dei bit rappresentano 83. Pertanto, N vale -128+83= -45
c) Sottraggo l'eccesso, 128, al valore derivato interpretando N come rappresentato con codifica posizionale binaria pura. N vale pertanto 83

Esercizio 3.

Realizzare una ALU dotata degli ingressi A e B, e dell'uscita U, tutti su 8 bit, e che in base ad un ingresso F gestisce le seguenti operazioni:

a. And bitwise di A e B
b. A mod 8, dove A è un intero senza segno
c. A div 8, dove A è un intero senza segno
d. Shift left 1 di B, dove B è un intero senza segno

La ALU genera un bit di esito V, che notifica eventuali overflow.

Soluzione:

Specifichiamo i valori di F per cui eseguire le varie operazioni

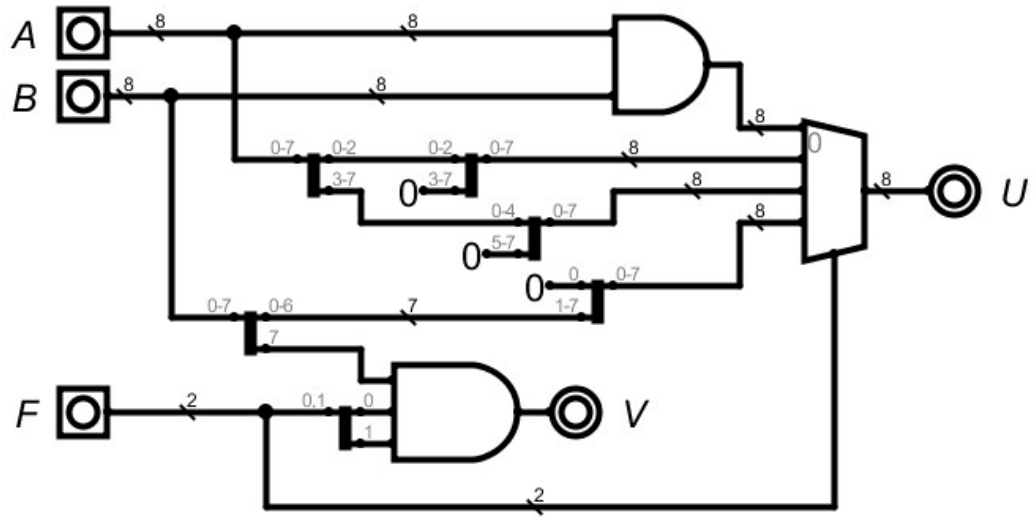
1. F=00: A and B
2. F=01: A mod 8
3. F=10: A div 8
4. F=11: B<<1

Gestisco le operazioni in parallelo su 4 linee di esecuzione. Serve pertanto un MUX con ingresso di selezione valorizzato ad F per ridirezionare su U il risultato dell'operazione selezionata.

Il resto della divisione intera di A per 8 corrisponde ai 3 bit meno significativi di A, mentre il quoto è definito dai 5 bit più significativi di A. Tali valori vanno rappresentati su 8 bit estendendo a 0.

L'unica operazione che può generare overflow è l'operazione di B<<1. Più precisamente, si verifica overflow nel se MSB(B) =1. Pertanto, V=1 sse F=11 e MSB(B)=1.

Il circuito risulta pertanto:



Esercizio 4.

Si vuole progettare un circuito sequenziale sincrono che controlla un array di 3 lucine LED integrato in un carica batterie. Il circuito riceve in ingresso un segnale C, su 2 bit, inoltrato dal caricatore, che riferisce il livello di carica della batteria. Più precisamente se C vale 0 lo stato di carica è inferiore al 33%. Se vale 1, la carica è tra il 33% e il 65%; se vale 2 è tra il 66% e 99%, mentre se vale 3 se la batteria è completamente carica. Il circuito è dotato di tre uscite L2, L1, e L0 impiegate per inoltrare ai led i comandi di accensione (1) e spegnimento (0).

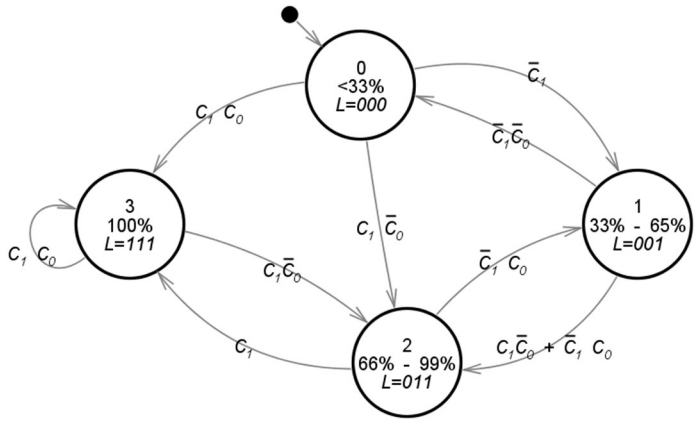
I led sono impiegati per denotare lo stato di carica della batteria inserita e il funzionamento del caricatore. Il circuito di controllo, infatti, mantiene acceso un numero di led pari al livello di carica riferito da C, e rappresenta la condizione di caricamento con un led lampeggiante. Più precisamente, quando C=0, per denotare che la batteria è scarica ma si sta caricando, ad ogni ciclo di clock si alternano le seguenti configurazioni di accensione dei led : 000, 001 (i led sono riferiti per posizione da 2 a 0). Quando C=1, per indicare un minimo livello di carica della batteria, il led 0 rimane sempre acceso, e si alternano le configurazioni di accensione: 001, 011. Con C=2, rimangono accesi sia led 0 che led 1, e la sequenza di accensione risulta 011, 111. Infine, quando C=3, tutti e 3 i led rimangono accesi.

Infine, si richiede di dotare il circuito di un ingresso di reset attivo basso. Il segnale di reset viene attivato dal caricatore in assenza di batterie. Durante la permanenza in tale situazione, tutti i led sono spenti.

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:



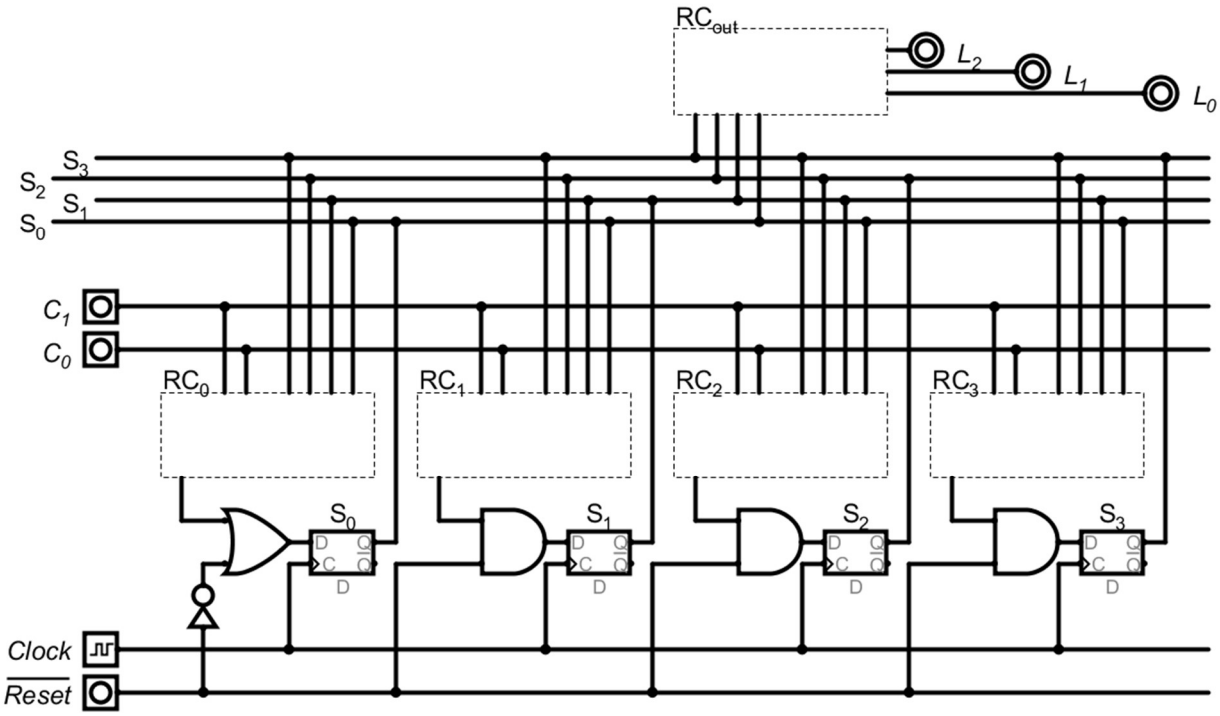
Il comportamento del sistema è modellato dalla FSM a fianco, dove ogni stato denota un livello di carica della batteria.

Si assume che i valori di C possano solo crescere gradualmente (ad es. non si può passare da livello di carica 1 a 3). L'unico incremento non graduale è consentito all'inserimento della batteria (ad es. da 0 a 2).

Rappresento gli stati con codifica a singolo 1 come segue:

0→ 0001; 1→0010; 2→0100; 3→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:

S3	S2	S1	S0	L2	L1	L0
0	0	0	1	0	0	0
0	0	1	0	0	0	1
0	1	0	0	0	1	1
1	0	0	0	1	1	1

Si deriva che:

L0= !S0 L1=S2+S3 L2=S3

Derivo le reti combinatorie che gestiscono le transizioni di stato dalla FSM:

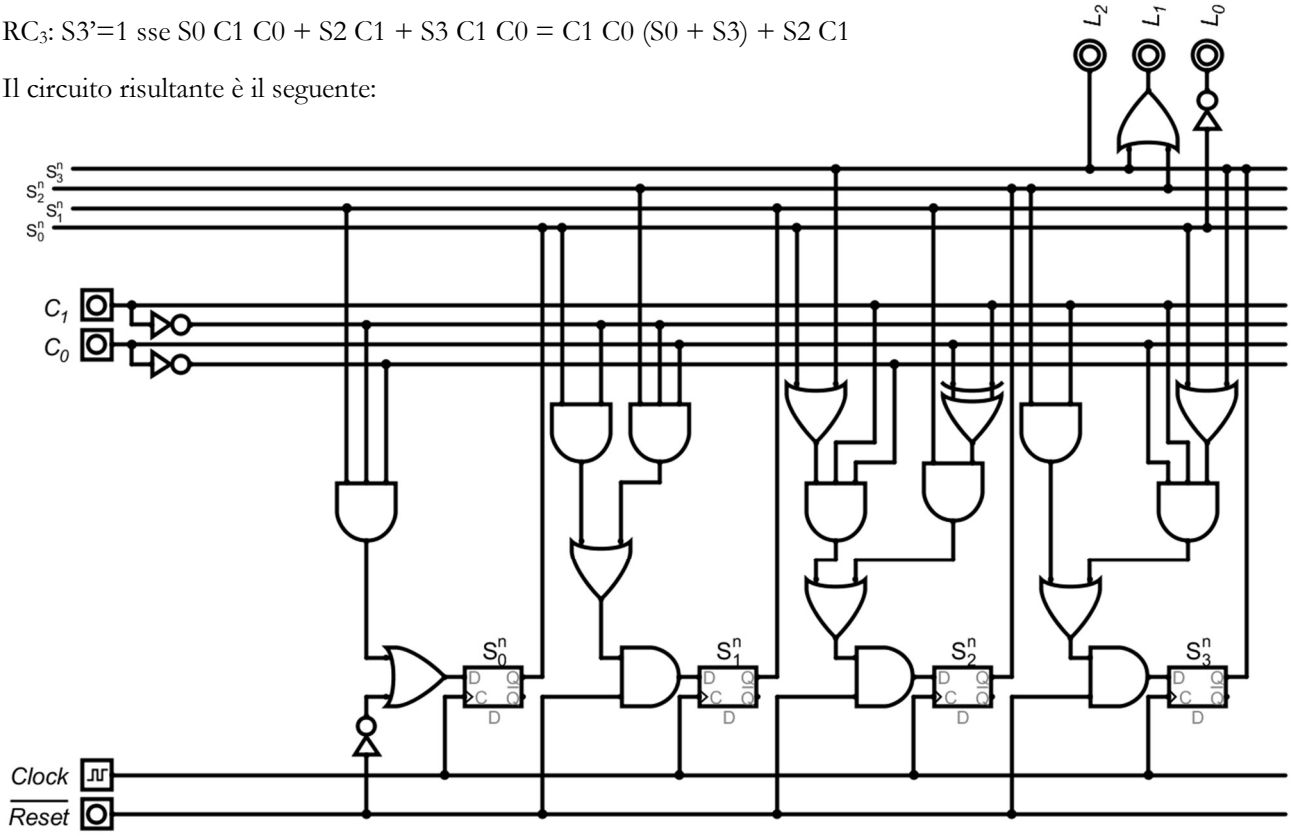
RC_0: S0'=1 sse S1 !C1 !C0

RC_1: S1'=1 sse S0 !C1 + S2 !C1 C0

RC_2: S2'=1 sse S0 C1 !C0 +S1(C1 xor C0) + S3 (C1 !C0) = (C1 !C0)(S0 +S3) + S1(C1 xor C0)

RC_3: S3'=1 sse S0 C1 C0 + S2 C1 + S3 C1 C0 = C1 C0 (S0 + S3) + S2 C1

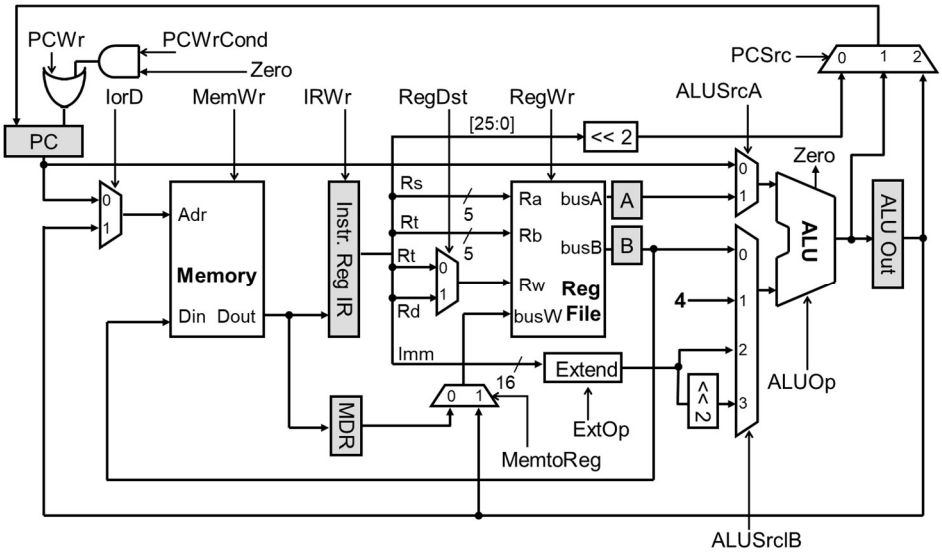
Il circuito risultante è il seguente:



Esercizio 5.

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

Per ogni fase indicare l'uso dei registri, motivando brevemente le risposte date. L'istruzione *subui* memorizza nel registro riferito da *rt* il risultato della somma tra il contenuti del registro con indice *rs* ed il campo immediato imm16.



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]]	
3)	ALUout ←A - ZeroExt(IR[imm16])	
4)	R[IR[rt]] ← ALUout	
5)		

Esercizio 6. Si consideri un sistema di memoria caratterizzato da una memoria di lavoro di 32 KB indirizzata a livello di Byte, e una cache ad indirizzamento diretto che gestisce 8 blocchi. Il sistema gestisce blocchi composti da 64 parole da 16 bit. Considerando la sequenza di richieste alla memoria riportata in tabella, si chiede di completare la tabella che illustra il comportamento della cache nel rispetto delle indicazioni seguenti:

- Nella colonna "esito" riportare H (hit) se il blocco richiesto si trova nella cache, M (miss) se invece il blocco deve essere caricato dalla memoria.
- Nelle colonne "dati" deve essere riportato l'indice del blocco della memoria (in decimale), presente nel corrispondente blocco della cache.
- Nella colonna "azione indicare il blocco a cui si accede in caso di hit, o in cui si caricano i dati della memoria in caso di miss e l'indice del blocco caricato (in decimale).

	Esito	Linea 0			Linea 1			Linea 2			Linea 3			Linea 4			Linea 5			Linea 6			Linea 7			Azione
		Valido	Etichetta	Dati	Valido	Etichetta	Dati	Valido	Etichetta	Dati	Valido	Etichetta	Dati	Valido	Etichetta	Dati	Valido	Etichetta	Dati	Valido	Etichetta	Dati	Valido	Etichetta	Dati	
1) Richiesta *addr* 10111 111 0010111	M																						1	10111	191	Carico il blocco 191 in linea 7
2) Richiesta *addr* 01111 101 0010011	M																1	01111	125							Carico il blocco 125 in linea 5
3) Richiesta *addr* 10111 110 0010001	M																			1	10111	190				Carico il blocco 190 in linea 6
4) Richiesta *addr* 10111 111 0010111	H																									Accedo al blocco 191 in linea 7

Mem size: 32KB (2^{15} *byte*) → indirizzamento a livello di byte su 15 bit; Block size: 64 word * 16 bit= 128 byte → byte offset nel blocco su 7 bit
Cache size (senza tag): 8*128 byte = 1024 byte (2^{10} *byte*); Num linee = num blocchi = 8 → line idx su 3 bit;
Tag su 5 bit (*15-(7+3)*); Scomposizione indirizzo memoria: 5bit tag + 3bit line idx + 7bit byte offset;
Indice del blocco in memoria riferito dai 6 bit più significativi (i bit di *tag e line idx*).