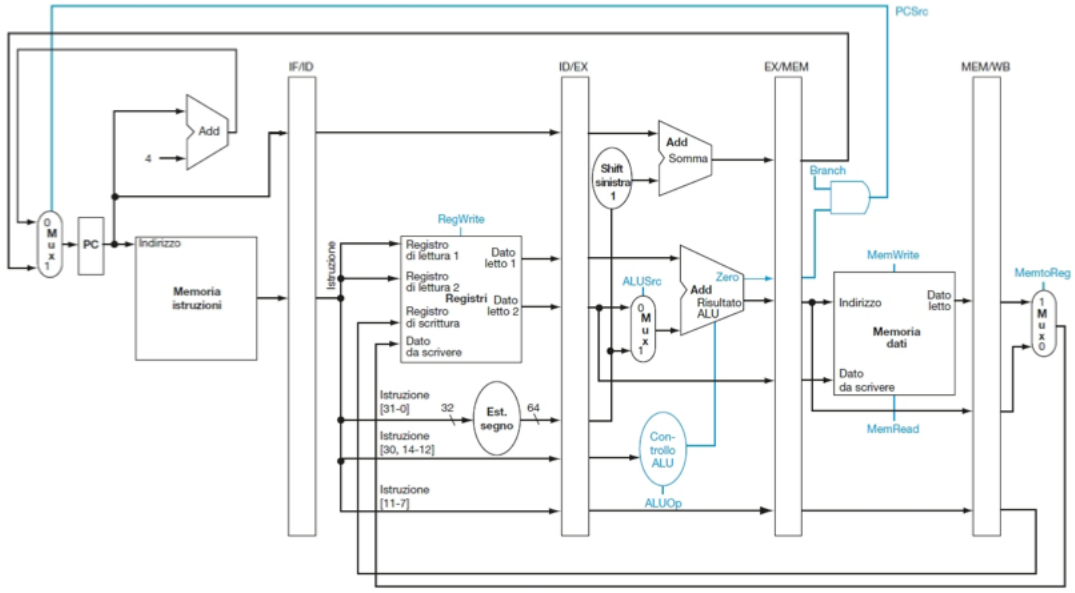
Esercizio 1

1) Descrivi le caratteristiche principali dell'architettura rappresentata nella figura seguente.

4 punti

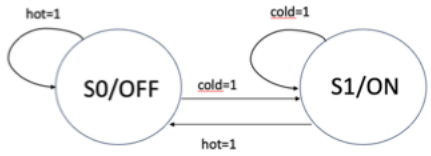


Possiamo notare che il nostro circuito utilizza la tecnica pipeline.
La pipeline è una tecnica utilizzata che permette l'ottimizzazione dell'esecuzione dei programmi consentendo alla CPU di lavorare su più istruzioni contemporaneamente .
La pipeline cerca di sfruttare tutte le risorse disponibili, infatti quando una risorsa è libera viene assegnata alla prima istruzione che ne ha bisogno aumentando il throughput complessivo.
Tutti gli stadi lavorano in contemporanea utilizzando risorse diverse.
L'incremento potenziale di velocita è determinato dal numero di stadi della pipeline anche se la velocita complessiva è influenzata dallo stadio più lento.
Per preservare il valore dell'istruzione in esecuzione, del PC e di altri valori si utilizzano dei registri aggiuntivi che sono opportunatamente etichettati per suddividere i diversi stadi della pipeline.
Come vediamo in figura abbiamo quattro registri : IF (instruction fetch), ID (instruction Decode), EX (execute) , Mem (Memory usata per operazioni load/store) e infine WB (Write Back usata per operazioni di tipo R).
Nella figura non è presente l'hazard dectetion unit, questa unità è molto importante per la pipeline in quanto mi permette di indivudare i diversi tipi di criticità (strutturali, dati e controllo).
In base al tipo di istruzione la nostra architettura si comporta in modo diverso, per esempio simuliamo l'esecuzione di un istruzione di tipo R.

Per tutti i tipi di istruzioni la fase Fetch e Decode è la stessa.
Nella Fetch prelevo la prossima istruzione da eseguire ed incremento il PC , è presente un MUX che controlla cio che viene scritto nel PC (PC + 4 o l'indirizzo di destinazione del branch). Questo MUX viene controllato da una porta logica che esegue l'AND tra il segnale Zero in uscita dalla alu e un segnale di controllo che mi indica che il tipo di istruzione è una branch, In questo caso Branch vale 0 in quanto si tratta di un'istruzione di tipo R.
Il RegWrite vale 1 e mi permette di eseguire la scrittura , nella fase EXECUTE il secondo operando della ALU proviene dal register file, questo perche ALUsrc (unita di controllo del mux collegato nel secondo ingresso della alu) vale 0.
Nella memoria dati il segnale MemRead vale 0 e il MemWrite vale 1 quindi il valore che si trova in memoria viene sovrascritto da quelllo che si trova nella linea "dato da scrivere".
Infine MemtoReg vale 1 in quanto il risultato viene mandato nel register file.

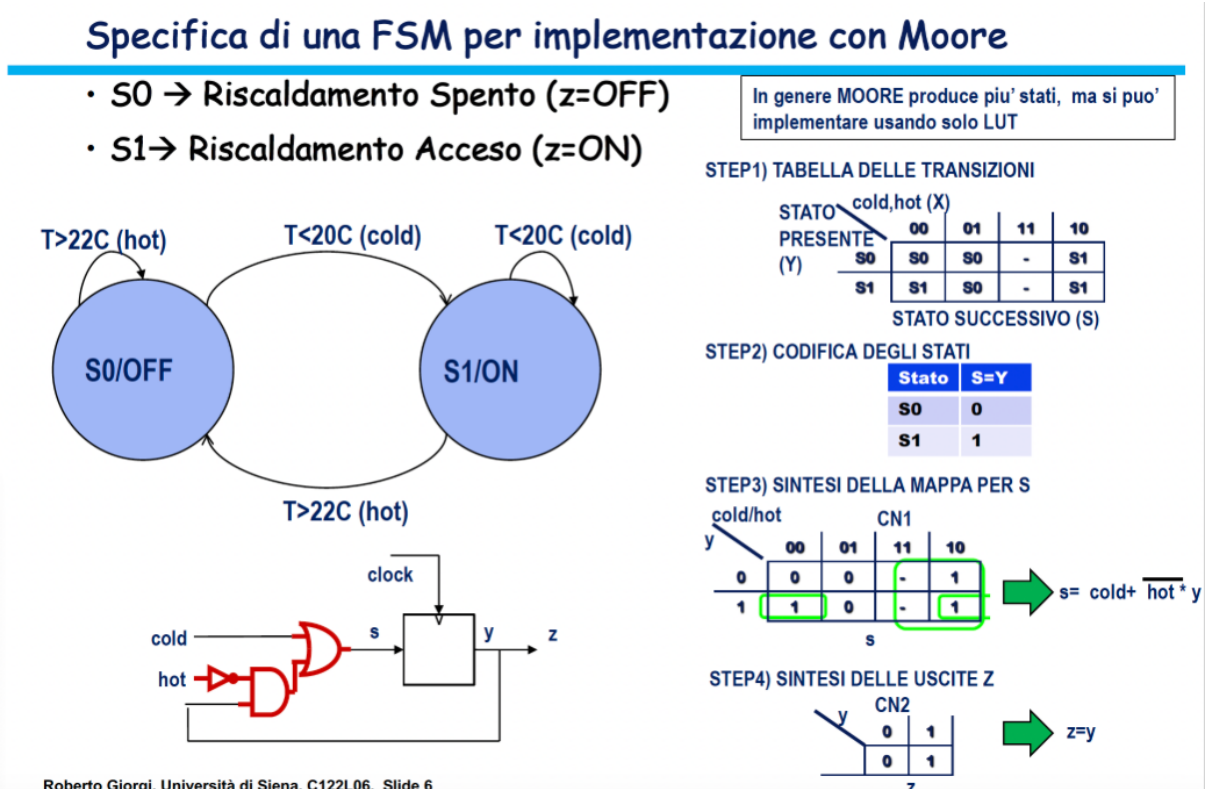
Esercizio 2

5) Sintetizza un circuito sequenziale alla Moore (con D-latch) per un controllore (con uscita ON/OFF) di un impianto di riscaldamento attivato da un termostato attraverso i segnali hot e cold e macchina a stati definita come segue:



4 punti

La soluzione per questo esercizio è uguale al seguente:



Esercizio 6

6) Spiega il comportamento del seguente programma multithreaded.

```
array A = {0,0,0}
int i=0
int k=0
```

Thread T1:	Thread T2:	Thread T3:

while (i<2){
A[k]=1;
i=i+1;
}
	A[i]=2;
	k=k+1;
		A[k]=3;

Non possiamo sapere con certezza il valore del risultato in quanto esistono diversi tipi di combinazione per risolvere i thread, diversi ordini di esecuzione dei thread portano a risultati diversi.
Facciamo finta che i Thread vengano svolti in ordine : T1, T2 e T3.
Iniziamo con i = 0 ed entriamo nel ciclo , A[0] = 1 e incrementiamo i portandolo a 1.
Rientriamo ancora nel ciclo e di nuovo A[0] = 1 e ora i vale 2.

Usciamo dal Thread 1 e passiamo al Thread 2.
A[2] = 2, e K ora vale 1.
Passiamo a Thread 3 dove A[1] è uguale a 3.
Quindi alla fine abbiamo come valori:
i = 2
K = 1
A[0] = 1
A[1] = 3
A[2] = 2

Osservazioni finali:

Non determinismo: Questo programma dimostra come l'ordine di esecuzione dei thread influenzi il risultato finale.
Race condition: L'accesso simultaneo e non sincronizzato alle variabili condivise A, i, e k causa race condition, portando a un comportamento imprevedibile.