Data path multiciclo

📖 Per l’utilizzo di data path multicilo è necessario implemenatre dei registri aggiuntivi per permetterne il corretto funzionamento.

I registri aggiuntivi necessari sono:

- registro per memorizzare dati intermedi
- registro che contiene una copia dell’istruzione in esecuzione
- due registri A e B che contengono gli indirizzi di memoria dei registri su cui l’istruzione opera
- un registro chiamato MDR che contiene i dati estratti dalla memoria
- un registro ALUout che contiene l’output intermedio dell’ALU

❓ Data path multicilo può essere utilizzata anceh per calcolare le jump e incrementare il PC del sistema.

Come vengono eseguite le istruzioni in un DP multiciclo

Le istruzioni in un DP multi ciclo vengono eseguite in più di un cuclo di clock ed è quinid necessario prendere alcuni accorgimenti per evitare la perdita dei dati durante il fronte del clock.

L’istruzione che deve essere eseguita vine divisa in passi ed ogni passo ha la durata di un clock è quindi importante che la quantità di lavoro da svolgere sia ben suddivisa per evitare di sforare il “tempo” a disposizione.

Ogni passo genera dei valori intermedi che sono salvati in degli appositi registri *es ALUout* e sono disponibili per il passo successivo.

Come vengono eseguiti i diversi tipi di istruzioni

▼ Fetch *comune a tutte*

- Carico in **IR** il valore del PC
- Viene controllato lo stato di alcuni segnali:
  - **MemRead** se serve leggere.
  - **IRWrite** se serve scrivere.
- **IorD** indica l'indirizzo in cui leggere o scrivere.
- Incremento del valore PC.
- **PCWrite** per sovrascrivere il valore del PC

▼ Decode *comune a tutte*

- Carico in **A, B** gli indirizzi di memoria dei registri a cui devo accedere.
- Per un eventuale calcolo del branch eseguo alcune **istruzioni dell'ALU**.

▼ Execute

▼ lw e sw

- Salvo in **ALUout** A con la sua estensione di segno
- Utilizzo altre **istruzioni ALU** per il calcolo effettivo.

▼ R-type

- Salvo in **ALUout** il risultato ricavato tra A e B a seconda del caoclo richiesto
- Utilizzo altre **istruzioni ALU** per il calcolo dell'inidirizzo di memoria per lw o sw.

▼ beq

- Eseguo il controllo di **uguaglianza** tra **A** e **B**
- Qualora **l'esito** fosse **positivo salvo** in **PC** il **valore** di **ALUOut**

▼ jump

- Salvo in **PC** l'indirizzo di destinazione della jump.

▼ Riassunto

Step name	Action for R-type instructions	Action for memory-reference instructions	Action for branches	Action for jumps
Instruction fetch	IR <= Memory[PC] PC <= PC + 4			
Instruction decode/register fetch	A <= Reg [IR[25:21]] B <= Reg [IR[20:16]] ALUOut <= PC + (sign-extend (IR[15:0]) << 2)			
Execution, address computation, branch/jump completion	ALUOut <= A op B	ALUOut <= A + sign-extend (IR[15:0])	if (A == B) PC <= ALUOut	PC <= {PC [31:28], (IR[25:0]),2'b00)}
Memory access or R-type completion	Reg [IR[15:11]] <= ALUOut	Load: MDR <= Memory[ALUOut] or Store: Memory [ALUOut] <= B		
Memory read completion		Load: Reg[IR[20:16]] <= MDR		