Lezione 21

CPU a ciclo multiplo: l'unità di controllo

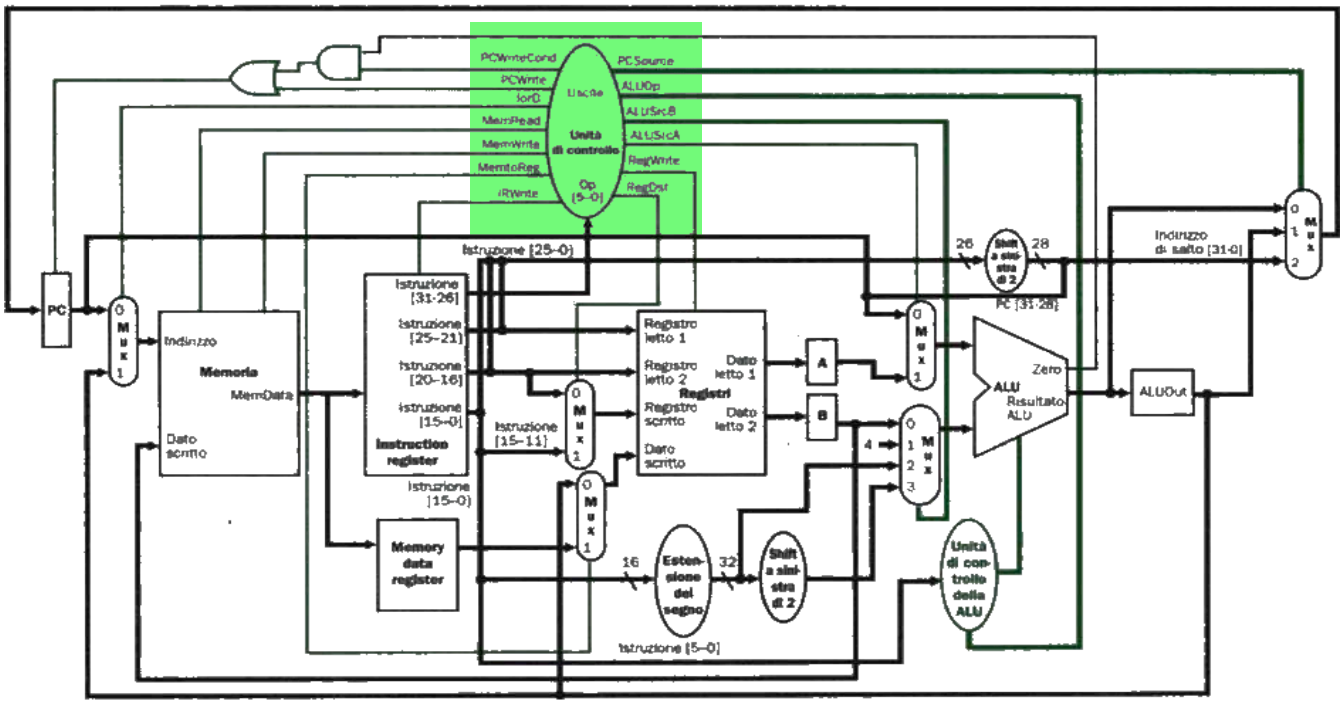
Proff. A. Borghese, F. Pedersini

Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano

Sommario

- I segnali di controllo della CPU multi-ciclo
- Sintesi dell'Unità di Controllo come Macchina a Stati Finiti

Struttura della CPU multi-ciclo



Segnali di controllo

- **6 Segnali di SELEZIONE**
  - **ALUSrcA:** Scelta del primo operando ALU (ALU a)
  - **ALUSrcB:** Scelta del secondo operando ALU (ALU b)
  - **IorD:** Selettore accesso memoria: (Dati: ALUOut / Istruzioni: PC)
  - **PCSrc:** Selettore del valore d'ingresso al PC
  - **RegDst**: Scelta del campo istruzione (rt/rd) che definisce il #RegWrite
  - **MemtoReg**: Selettore alla porta di scrittura del RF (MDR / ALUOut)

- **6 Segnali di COMANDO**
  - **RegWrite**: Comando di scrittura nel RF (istruzioni con WriteBack)
  - **MemRead:** Comando di lettura Memoria
  - **MemWrite**: Comando di scrittura Memoria
  - **IRWrite:** Comando di scrittura dell' IR (durante la fase di fetch)
  - **PCWrite:** Comando di scrittura del PC
  - **PCWriteCond:** Abilitazione aggiornamento PC con indirizzo di salto

- **Scrittura nei registri A, B, ALUOut**:
  Comandata dal clock → non serve un segnale di controllo

Segnali di controllo

- Segnali di selezione:

Segnale	Valore	Effetto
ALUSrcA	**0**	1° operando è il valore attuale del PC
	1	1° operando proviene dalla porta di lettura 1 del RF
ALUSrcB	**00**	ALU_b = seconda porta di lettura del RF
	01	ALU_b = **+ 4**
	10	ALU_b = **offset** → estensione segno
	11	ALU_b = **offset** → estensione segno → shift sx di due posizioni
IorD	**0**	L'indirizzo della memoria proviene dal PC
	1	L'indirizzo della memoria proviene dalla ALU (**ALUOut**)
PCSrc	**00**	In PC viene scritta l'uscita della ALU (**PC + 4**)
	01	In PC viene scritta il contenuto di **ALUOut** (indirizzo di una **branch**)
	10	In PC viene scrittto l'indirizzo di destinazione della **jump**
RegDst	**0**	Il registro da scrivere è definito nel campo **rt** (bit 20–16)
	1	Il registro da scrivere è definito nel campo **rd** (bit 15–11)
MemToReg	**0**	Il contenuto da scrivere nel RF è preso **dall'uscita della ALU**
	1	Il contenuto da scrivere nel RF è preso **dalla memoria (MDR)**

Ciclo di esecuzione di un'istruzione

- Le diverse fasi vengono eseguite in momenti diversi
  - L'esecuzione del ciclo richiede **da 3 a 5 cicli di clock**
  - L' UC deve "ricordare" tutte le fasi di ogni istruzione

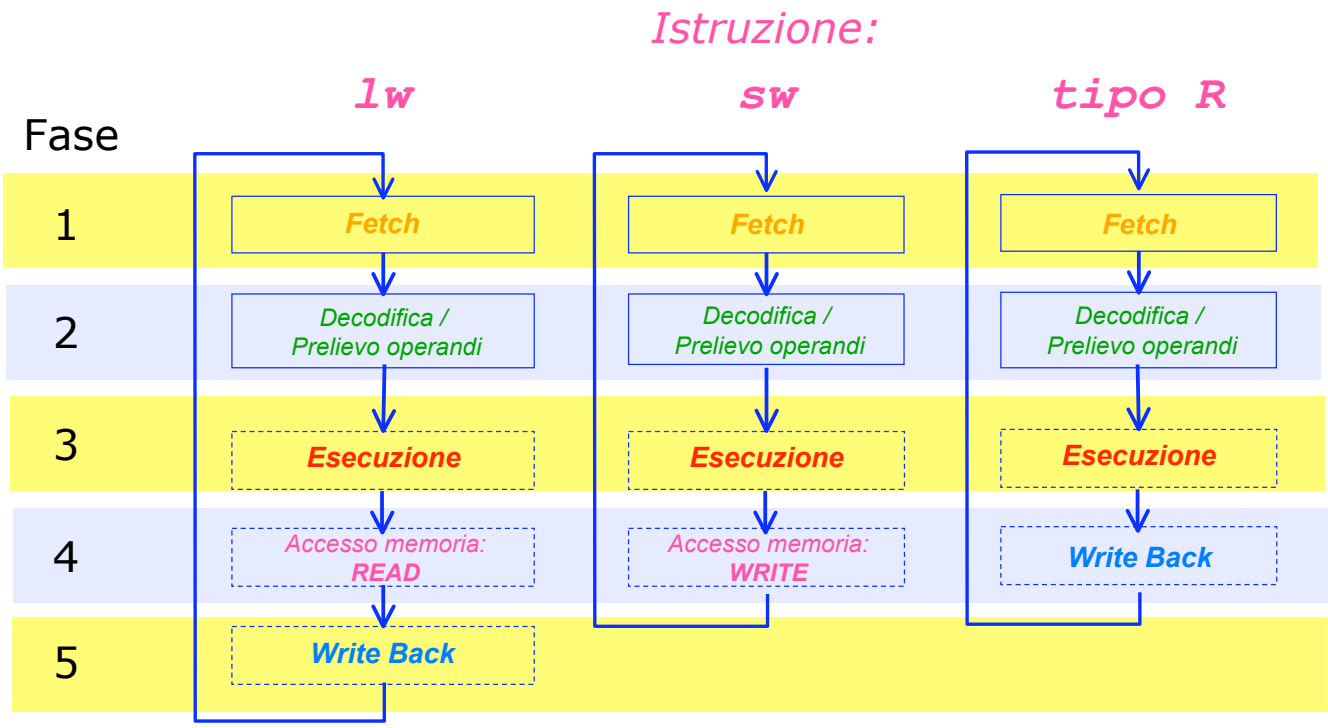
↓

L' UC è una macchina a stati finiti

Suddivisione in passi:

- Le operazioni elementari che hanno bisogno di unità funzionali diverse possono essere eseguite in **parallelo**
  - contemporaneamente, **nello stesso passo**
- Le operazioni elementari che hanno bisogno della stessa unità funzionale devono essere eseguite in **sequenza**
  - in **passi successivi** del ciclo di esecuzione

Ciclo di esecuzione istruzioni

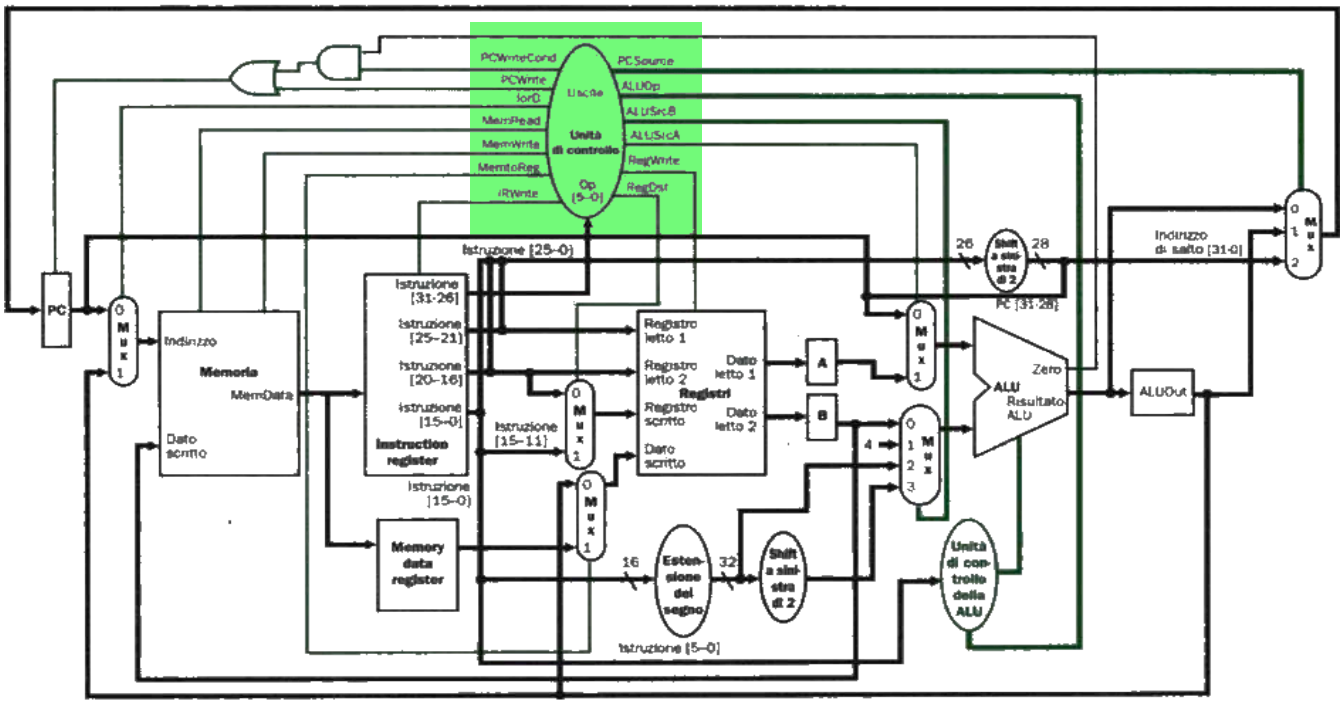


Riassunto dell'esecuzione

istruzioni→ fase ↓	Istruzioni a/l Tipo R	Istruz. di accesso a memoria (`lw`,`sw`)	Salto condizionato	Salto non condizionato
Fetch	IR = Memory[PC] PC = PC + 4			
Decodifica / Prelievo dati	A = Reg[IR[25-21]] B = Reg[IR[20-16]] ALUout = PC + (sign_ext(IR[15-0]) << 2)			
Esecuzione	ALUOut = A oper B	ALUOut = A + sign_ext(IR[15-0])	If (A==B) then PC = ALUOut	PC = PC[31-28] || IR[25-0]<<2
`lw/sw`: mem tipo R: WB	Reg(IR[15-11]) = ALUOut	**`lw`:** MDR=Memory[ALUOut] **`sw`:** Memory[ALUOut] = B		
`lw/sw`: WB		**`lw`:** Reg[IR[20-16]] = MDR		

Le istruzioni richiedono da **3** a **5** cicli di clock

CPU multi-ciclo



Macchina a Stati Finiti (di Moore)

- Una Macchina a Stati Finiti (MSF) è definita dalla quintupla:

$$< X,\ I,\ Y,\ f(\cdot),\ g(\cdot) >$$

X: insieme degli stati (in numero finito).

I: alfabeto di ingresso: l'insieme dei simboli che si possono presentare in ingresso. Con **n** ingressi, avremo 2^n possibili configurazioni.

Y: alfabeto di uscita: l'insieme dei simboli che si possono generare in uscita. Con **m** uscite, avremo 2^m possibili configurazioni

$f(\cdot)$: funzione stato prossimo: $X^* = f(X, I)$
Definisce l'evoluzione della macchina nel tempo, in modo deterministico

$g(\cdot)$: funzione di uscita: $Y = g(X)$ (macchina di **Moore**)
$Y = g(X, I)$ (macchina di **Mealy**)

- Per il buon funzionamento della macchina è previsto uno **stato iniziale**, al quale la macchina può essere portata mediante un comando di **reset**.

Sintesi della FSM della CPU

Ingressi (I) → codice operativo, ALU.zero

Uscita (Y) → segnali di controllo

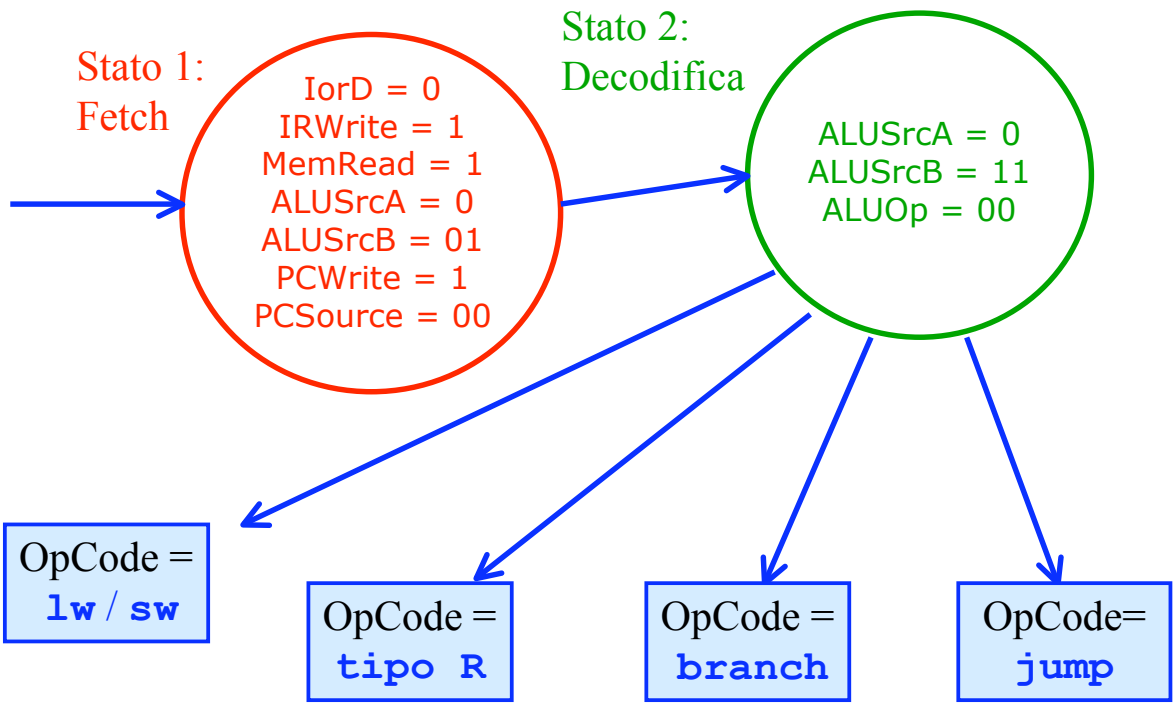
Stato (X) → istruzione corrente, fase di esecuzione

- I valori dei segnali di controllo (uscita) dipendono:
  - dalla fase di esecuzione dell'istruzione corrente → STATO
- Il passo successivo dell'istruzione (stato prossimo) dipende:
  - dal codice operativo → INGRESSO
  - dalla fase di esecuzione corrente → STATO

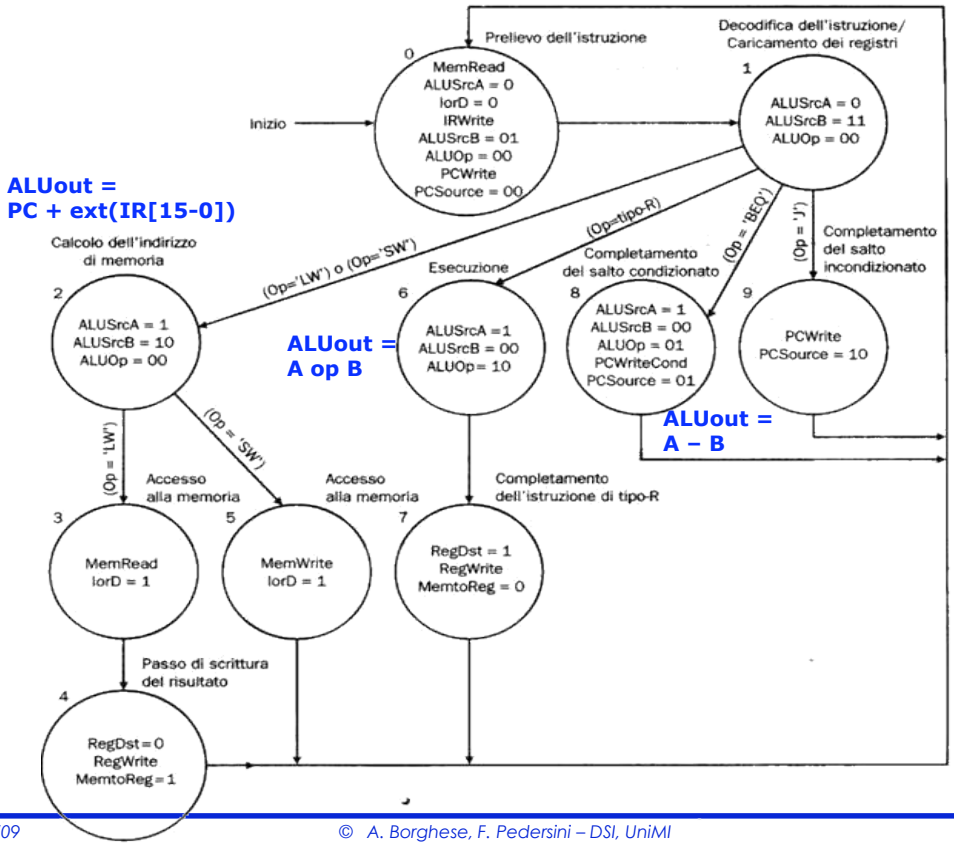
$$Y = f(X)$$
$$X^* = f(I, X)$$

→ Macchina di MOORE

Fase di fetch e decodifica



FSM – State Transition Graph (STG)



Segnali di controllo

Segnale controllo ➔ / Passo esecuzione ↓	IorD	MemRead	MemWrite	IRWrite	ALUScrA	ALUSrcB	ALUop	PCSource	PCWrite	PCWrCond	RegWrite	RegDst	MemtoReg
Fase fetch	0	1	0	1	0	01	00	00	1	0	0	X	X
Decodifica	X	0	0	0	0	11	00	X	0	0	0	X	X
Exec I - beq	X	0	0	0	1	00	01	01	0	1	0	X	X
Exec I – j	X	0	0	0	X	X	X	10	1	0	0	X	X
Exec I - R	X	0	0	0	1	00	10	X	0	0	0	X	X
Exec I – lw	X	0	0	0	1	10	00	X	0	0	0	X	X
Exec I - sw	X	0	0	0	1	10	00	X	0	0	0	X	X
Exec II – sw	1	0	1	0	X	X	X	X	0	0	0	X	X
Exec II – R	X	0	0	0	X	X	X	X	0	0	1	1	0
Exec II – lw	1	1	0	0	X	X	X	X	0	0	0	X	X
Exec III – lw	X	0	1	0	X	X	X	X	0	0	1	0	1

FSM – State Transition Table (STT)

Ingressi (I) + Segnali (Y) → / Passo esec. = Stato (X) ↓		OpCode = R	OpCode = sw	OpCode = lw	OpCode = beq	OpCode = j	IorD	MemRead	MemWrite	IRWrite	ALUScrA	ALUSrcB	ALUop	PCSource	PCWrite	PCWriteCond	RegWrite	RegDst	MemtoReg
Fase fetch	**0**						**0**	**1**	**0**	**1**	**0**	**01**	**00**	**00**	**1**	**0**	**0**	X	X
Decodifica	**1**						X	**0**	**0**	**0**	**0**	**11**	**00**	X	**0**	**0**	**0**	X	X
Exec I – beq	**8**						X	**0**	**0**	**0**	**1**	**00**	**01**	**01**	**0**	**1**	**0**	X	X
Exec I – jump	**9**						X	**0**	**0**	**0**	X	X	X	**10**	**1**	**0**	**0**	X	X
Exec I – a/l R	**6**						X	**0**	**0**	**0**	**1**	**00**	**10**	X	**0**	**0**	**0**	X	X
Exec I – lw/sw	**2**						X	**0**	**0**	**0**	**1**	**10**	**00**	X	**0**	**0**	**0**	X	X
Exec II – sw	**5**						**1**	**0**	**1**	**0**	X	X	X	X	**0**	**0**	**0**	X	X
Exec II – R	**7**						X	**0**	**0**	**0**	X	X	X	X	**0**	**0**	**1**	**1**	**0**
Exec II – lw	**3**						**1**	**1**	**0**	**0**	X	X	X	X	**0**	**0**	**0**	X	X
Exec III – lw	**4**						X	**0**	**1**	**0**	X	X	X	X	**0**	**0**	**1**	**0**	**1**

FSM – State Transition Table (STT)

Ingressi (I) + Segnali (Y) → / Passo esec. = Stato (X) ↓		OpCode = R	OpCode = sw	OpCode = lw	OpCode = beq	OpCode = j	IorD	MemRead	MemWrite	IRWrite	ALUScrA	ALUSrcB	ALUop	PCSource	PCWrite	PCWriteCond	RegWrite	RegDst	MemtoReg
Fase fetch	**0**	**1**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**01**	**00**	**00**	**1**	**0**	**0**	X	X
Decodifica	**1**	**6**	**2**	**2**	**8**	**9**	X	**0**	**0**	**0**	**0**	**11**	**00**	X	**0**	**0**	**0**	X	X
Exec I – beq	**8**	X	X	X	**0**	X	X	**0**	**0**	**0**	**1**	**00**	**01**	**01**	**0**	**1**	**0**	X	X
Exec I – jump	**9**	X	X	X	X	**0**	X	**0**	**0**	**0**	X	X	X	**10**	**1**	**0**	**0**	X	X
Exec I – a/l R	**6**	**7**	X	X	X	X	X	**0**	**0**	**0**	**1**	**00**	**10**	X	**0**	**0**	**0**	X	X
Exec I – lw/sw	**2**	X	**5**	**3**	X	X	X	**0**	**0**	**0**	**1**	**10**	**00**	X	**0**	**0**	**0**	X	X
Exec II – sw	**5**	X	**0**	X	X	X	**1**	**0**	**1**	**0**	X	X	X	X	**0**	**0**	**0**	X	X
Exec II – R	**7**	**0**	X	X	X	X	X	**0**	**0**	**0**	X	X	X	X	**0**	**0**	**1**	**1**	**0**
Exec II – lw	**3**	X	X	**4**	X	X	**1**	**1**	**0**	**0**	X	X	X	X	**0**	**0**	**0**	X	X
Exec III – lw	**4**	X	X	**0**	X	X	X	**0**	**1**	**0**	X	X	X	X	**0**	**0**	**1**	**0**	**1**

UC: sintesi funzione uscita: Y = g(X)

Segnali (Y) → / Stato (X) ↓		IorD	MemR	MemW	IRWrite	ALUSA	ALUSB	ALUop	PCSrc	PCWrit	PCWC	RegWrt	RegDst	MmReg
Fase fetch	**0: 0000**	**0**	**1**	**0**	**1**	**0**	**01**	**00**	**00**	**1**	**0**	**0**	X	X
Decodifica	**1: 0001**	X	**0**	**0**	**0**	**0**	**11**	**00**	X	**0**	**0**	**0**	X	X
Exec I – beq	**8: 1000**	X	**0**	**0**	**0**	**1**	**00**	**01**	**01**	**0**	**1**	**0**	X	X
Exec I – jump	**9: 1001**	X	**0**	**0**	**0**	X	X	X	**10**	**1**	**0**	**0**	X	X
Exec I – R	**6: 0110**	X	**0**	**0**	**0**	**1**	**00**	**10**	X	**0**	**0**	**0**	X	X
Exec I – lw,sw	**2: 0010**	X	**0**	**0**	**0**	**1**	**10**	**00**	X	**0**	**0**	**0**	X	X
Exec II – sw	**5: 0101**	**1**	**0**	**1**	**0**	X	X	X	X	**0**	**0**	**0**	X	X
Exec II – R	**7: 0111**	X	**0**	**0**	**0**	X	X	X	X	**0**	**0**	**1**	**1**	**0**
Exec II – lw	**3: 0011**	**1**	**1**	**0**	**0**	X	X	X	X	**0**	**0**	**0**	X	X
Exec III – lw	**4: 0100**	X	**0**	**1**	**0**	X	X	X	X	**0**	**0**	**1**	**0**	**1**

Esempi:

MemToReg: Y_0 = (Stato == 4)

RegWrite: Y_2 = (Stato == 7) **OR** (Stato == 4)

ALUSrcA: Y_{11} = (Stato == 8) **OR** (Stato == 6) **OR** (Stato == 2)

Sintesi stato futuro

- Funzione stato futuro: **X* = *f* (X , I)**
  - devo codificare gli ingressi (OpCode) e lo stato (passo esec.):

OpCode (I) → / Passo esec. (X) ↓		R: 000000	sw: 101011	lw: 100011	beq: 000100	jump: 000010
Fase fetch	**0: 0000**	**0001**	**0001**	**0001**	**0001**	**0001**
Decodifica	**1: 0001**	**0110**	**0010**	**0010**	**1000**	**1001**
Exec I – beq	**8: 1000**	X	X	X	**0000**	X
Exec I – jump	**9: 1001**	X	X	X	X	**0000**
Exec I – a/l R	**6: 0110**	**0111**	X	X	X	X
Exec I – lw/sw	**2: 0010**	X	**0101**	**0011**	X	X
Exec II – sw	**5: 0101**	X	**0000**	X	X	X
Exec II – R	**7: 0111**	**0000**	X	X	X	X
Exec II – lw	**3: 0011**	X	X	**0100**	X	X
Exec III – lw	**4: 0100**	X	X	**0000**	X	X

STT Codificata – sintesi "stato futuro": X* = f(X,I)

Sintesi di $x_i^*(X,I),\ i=0..3:\quad X=\langle x_3\,|\,x_2\,|\,x_1\,|\,x_0\rangle,\quad I=\langle i_5\,|\,i_4\,|\,i_3\,|\,i_2\,|\,i_1\,|\,i_0\rangle;$

OpCode (I) → / (X) ↓	R: 000000	sw: 101011	lw: 100011	beq: 000100	jump: 000010
Passo esec.	$x_3x_2x_1x_0$	$x_3^*\,x_2^*\,x_1^*\,x_0^*$			
Fase fetch	0: 0000	0001	0001	0001	0001	0001
Decodifica	1: 0001	0110	0010	0010	1000	1001
Exec I – beq	8: 1000	X	X	X	0000	X
Exec I – jump	9: 1001	X	X	X	X	0000
Exec I – a/l R	6: 0110	0111	X	X	X	X
Exec I – lw/sw	2: 0010	X	0101	0011	X	X
Exec II – sw	5: 0101	X	0000	X	X	X
Exec II – R	7: 0111	0000	X	X	X	X
Exec II – lw	3: 0011	X	X	0100	X	X
Exec III – lw	4: 0100	X	X	0000	X	X

Sintesi di: x_0^*

Sintesi: X* = f(X , I)

Sintesi di $x_i^*(X,I),\ i=0..3:\quad X=\langle x_3\,|\,x_2\,|\,x_1\,|\,x_0\rangle,\quad I=\langle i_5\,|\,i_4\,|\,i_3\,|\,i_2\,|\,i_1\,|\,i_0\rangle;$

$$x_0^* = \overline{x}_0\overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_0 x_1\overline{x}_3 + \overline{i}_0 i_1\overline{x}_3 = \overline{x}_3\left(\overline{x}_0\left(\overline{x}_1\overline{x}_2 + x_1\right) + \overline{i}_0 i_1\right) = \overline{x}_3\left(\overline{x}_0\left(\overline{x}_2 + x_1\right) + \overline{i}_0 i_1\right)$$

$$\vdots$$

$$x_3^* = x_0\overline{x}_1\overline{x}_2\overline{x}_3\overline{i}_0\left(i_1 + i_2\right)$$