Lezione 19

CPU a singolo ciclo: l'unità di controllo, esecuzione istruzioni tipo J

Proff. A. Borghese, F. Pedersini

Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano

ISA MIPS – tipi di istruzione: R, I (`lw/sw,branch`)

- I tre formati delle istruzioni sono:

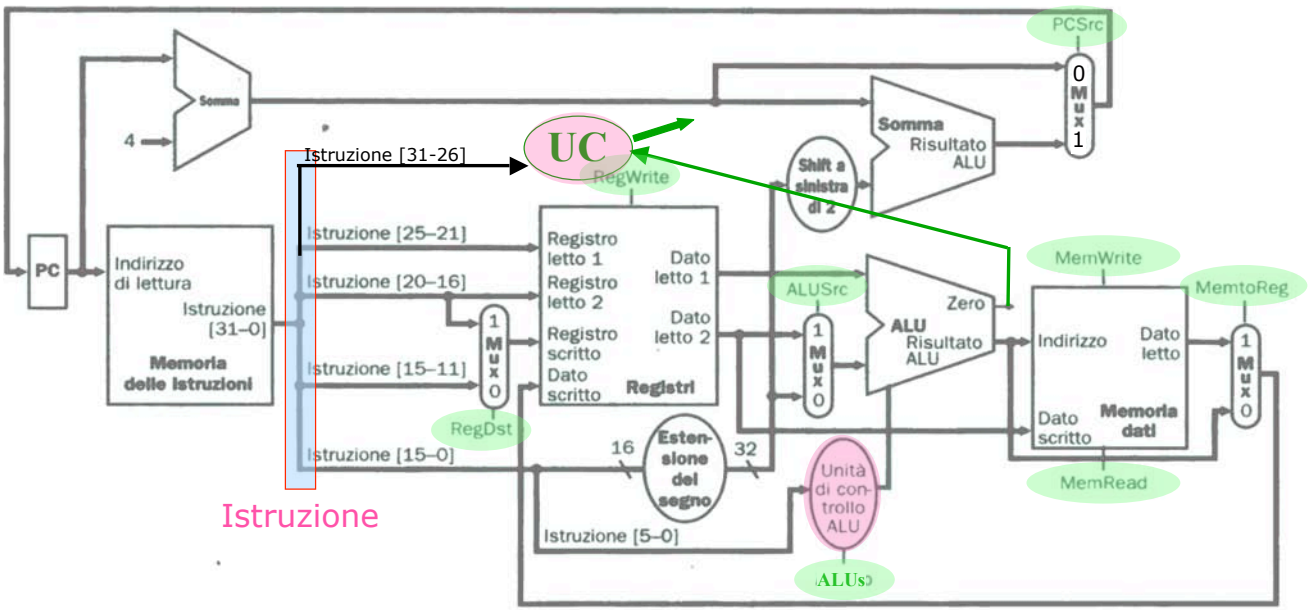
31-26	25-21	20-16	15-11	10-6	5-0
op	rs	rt	rd	shamt	funct

31-26	25-21	20-16	15-0
35/43	rs	rt	offset

31-26	25-21	20-16	15-0
4	rs	rt	indirizzo

- Campo op sempre contenuto nei primi 6 bit (in seguito, Op[5-0])
- Registri da leggere sempre rs e rt, di posizione 25-21 e 20-16
- Registro base per lettura/scrittura sempre rs, di posizione 25-21
- Offset sempre in posizione 15-0
- Registro destinazione rd (15-11) per istruzioni di tipo R, rt (20-16) per istruzioni lettura (necessario multiplexer)

Schema generale (`lw/sw + R + beq`)



Unità di Controllo CPU: $Segnali\_Controllo = f(\texttt{Opcode})$

Controllore della ALU: $ALUs = f(\texttt{Opcode}, \texttt{funct})$

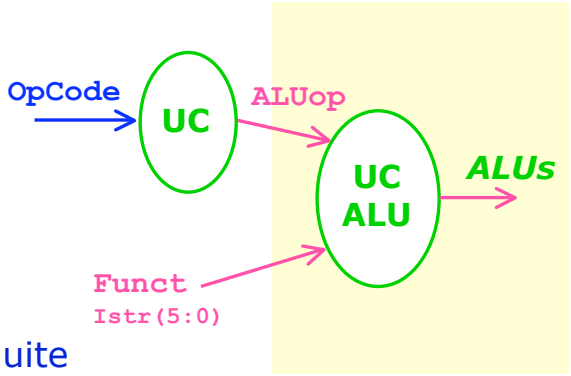
Sommario

Unità di controllo:

- Controllore della ALU
- Unità di Controllo Principale

UC e ALU

- Data l'istruzione, UC deve inviare il comando opportuno alla ALU:
  - **ALUop** (dip. dall'istruzione)
  - **Funct**



- Quali operazioni devono essere eseguite per le diverse istruzioni?

Istruz.		Operaz. ALU	Codice **ALUop**
R	→	dipende dal campo **funct**:	10
lw	→	somma:	00
sw	→	somma:	00
beq	→	confronto → differenza:	01

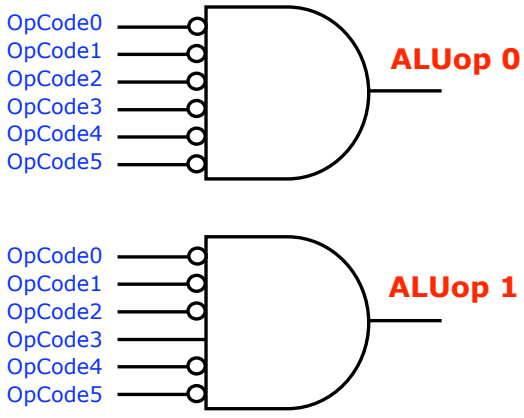
- 3 classi distinte (2 bit → **ALUop**)

Istr	ALUop
R	10
lw/sw	00
beq	01

Controllo della ALU: sintesi *ALUop*

- Sintesi del circuito logico: **ALUop = *f* (OpCode)**

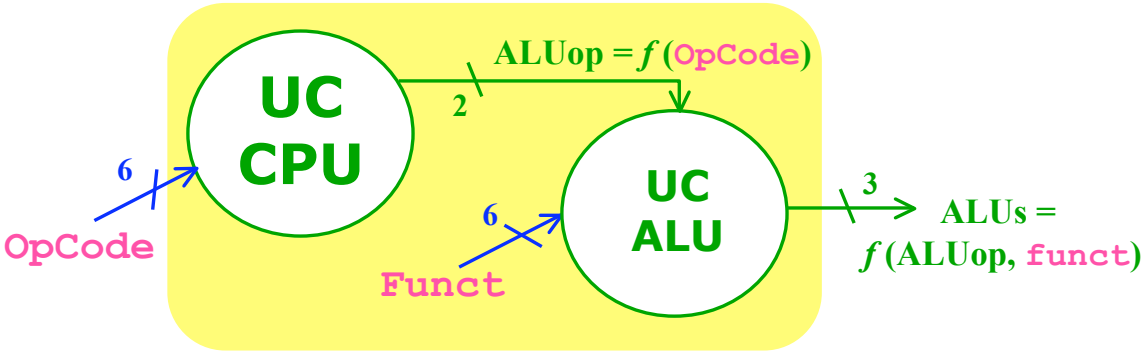
Istr	OpCode						ALUop	
	0	1	2	3	4	5	0	1
lw	1	0	0	0	1	1	0	0
sw	1	0	1	0	1	1	0	0
beq	0	0	0	1	0	0	0	1
add	0	0	0	0	0	0	1	0
sub	0	0	0	0	0	0	1	0
and	0	0	0	0	0	0	1	0
or	0	0	0	0	0	0	1	0
slt	0	0	0	0	0	0	1	0



$$ALUop0 = \overline{OpCode0} \cdot \overline{OpCode1} \cdot \overline{OpCode2} \cdot \overline{OpCode3} \cdot \overline{OpCode4} \cdot \overline{OpCode5}$$

$$ALUop1 = \overline{OpCode0} \cdot \overline{OpCode1} \cdot \overline{OpCode2} \cdot OpCode3 \cdot \overline{OpCode4} \cdot \overline{OpCode5}$$

UC–ALU: Controllo gerarchico



3 classi distinte

1. R
2. lw/sw
3. branch

OpCode	ALUop
R	10
lw/sw	00
beq	01

tipo R → ALUs dipende da funct

funct	ALUs
and	000
or	001
add	010
sub	110
slt	111

Controllo della ALU

ALUs = *f* (ALUop, Funct)

Istr	OpCode						ALUop		Funct						ALUs		
	0	1	2	3	4	5	0	1	0	1	2	3	4	5	0	1	2
lw	1	0	0	0	1	1	0	0	x	x	x	x	x	x	0	1	0
sw	1	0	1	0	1	1	0	0	x	x	x	x	x	x	0	1	0
beq	0	0	0	1	0	0	0	1	x	x	x	x	x	x	1	1	0
add	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0
sub	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1	1	0
and	0	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0
or	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	0	1
slt	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1

$$ALUs_0 = \overline{Ao_0} Ao_1 + Ao_0 \overline{Ao_1} \left(f_0 \overline{f_1} \overline{f_3} f_4 \overline{f_5} \right) \quad SOP$$

$$ALUs_1 = \left(\overline{Ao_0} + Ao_1 \right) \left(\overline{f_0} + f_1 + f_2 + \overline{f_3} + f_4 \right) \quad POS$$

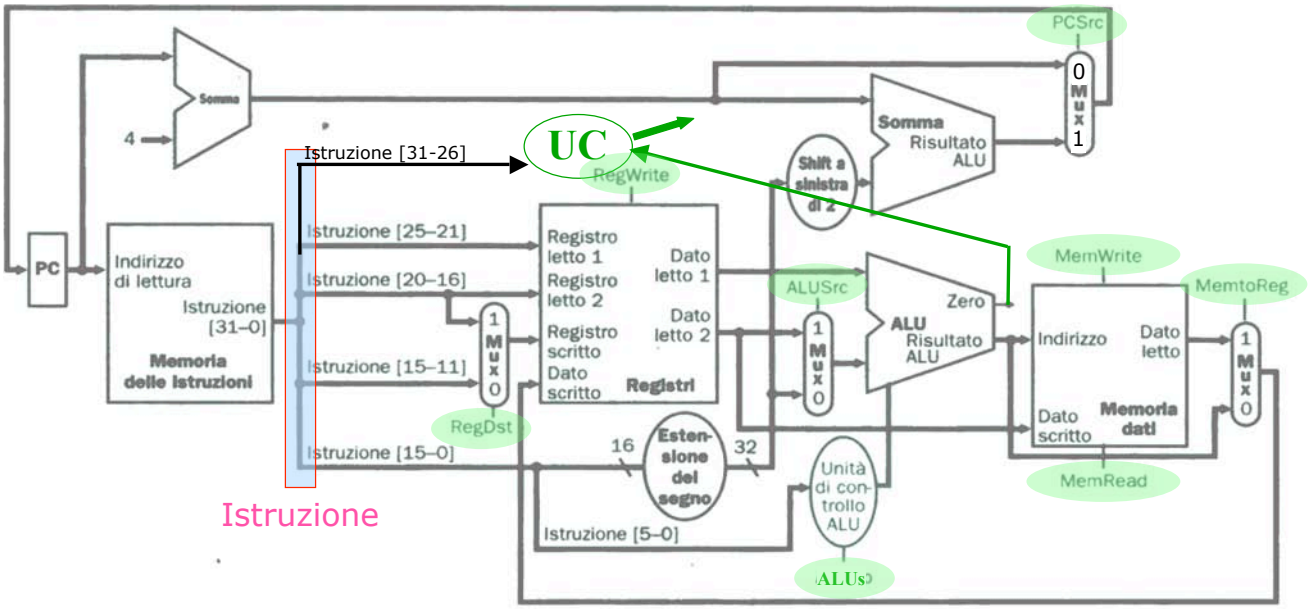
$$ALUs_2 = Ao_0 \overline{Ao_1} f_0 \overline{f_1} \left(\overline{f_2} f_3 \overline{f_4} f_5 + f_2 \overline{f_3} f_4 \overline{f_5} \right) \quad SOP$$

Sommario

Unità di controllo:

- Controllore della ALU
- Unità di Controllo Principale

Schema generale (`lw/sw` + `R` + `beq`)



Unità di Controllo CPU: **Segnali_Controllo** $= f($`Opcode`$)$

Controllore della ALU: **ALUs** $= f($`Opcode , funct`$)$

Segnali di **controllo CPU**

Segnali di **selezione**:

Segnale	Effetto quando è **negato (0)**	Effetto quando è **affermato (1)**
RegDst	Il numero del registro destinazione proviene dal campo **rd** (bit **15-11**)	Il numero del registro destinazione proviene dal campo **rt** (bit **20-16**)
ALUSrc	Il secondo operando della ALU proviene dal campo **offset** (estesa a 32 bit)	Il secondo operando della ALU proviene dalla **II porta di lettura** del Register File
PCSrc	Il valore del PC viene sostituito dall'uscita del sommatore che calcola **PC+4**	Il valore del PC viene sostituito dall'uscita del sommatore che calcola l'**indirizzo di salto**
MemtoReg	Il valore inviato a ContenutoWrite del Register File proviene dalla **ALU**	Il valore inviato a ContenutoWrite del Register File proviene dalla **memoria**

Segnali di **comando**:

Segnale	Effetto quando è **negato (0)**	Effetto quando è **affermato (1)**
RegWrite	Nessuno	Nel registro specificato su #RegWrite viene scritto il valore all'ingresso: ContenutoWrite
MemRead	Nessuno	Comando di **lettura** della memoria dati
MemWrite	Nessuno	Comando di **scrittura** della memoria dati

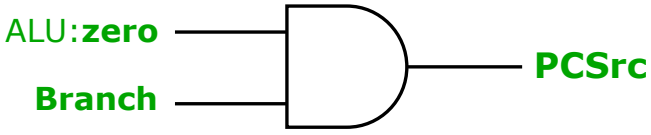
Segnali di controllo del Data-path

Istr	OpCode	Reg Dst	ALU src	Mem toReg	Reg Write	Mem Read	Mem Write	Branch	ALU op
R	000000	0	1	0	1	0	0	0	10
lw	100011	1	0	1	1	1	0	0	00
sw	101011	x	0	x	0	0	1	0	00
beq	000100	x	1	x	0	0	0	1	01

- Relazione tra **PCSrc** e **Branch**:

PCSrc = Branch **AND** *[condiz. salto verificata]*

Es: **beq:**



UC principale: Segnali di controllo

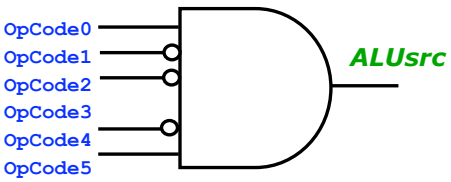
- Tabella di sintesi:
  - ingressi: **OpCode, ALU.zero**
  - uscite: segnali di controllo CPU

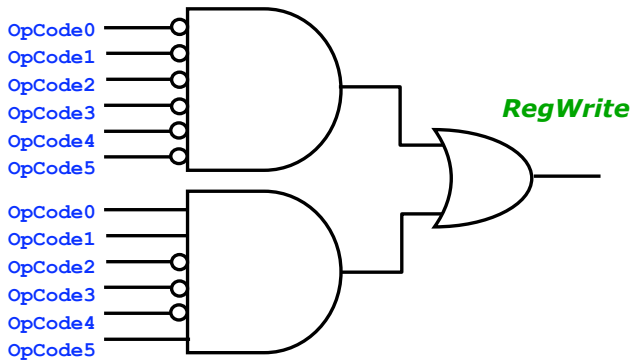
Ingressi			Uscite							
Istr	OpCode	ALU: Zero	Reg Dst	ALU src	Mem toReg	Reg Write	Mem Read	Mem Write	Branch	ALU op
R	000000	x	0	1	0	1	0	0	0	10
lw	100011	x	1	0	1	1	1	0	0	00
sw	101011	x	x	0	x	0	0	1	0	00
beq	000100	1	x	1	x	0	0	0	1	01

RegDst = OpCode0
MemtoReg = OpCode0
MemRead = OpCode5 ·~OpCode3
PCsrc = Branch ·ALUzero
RegWrite = NOT(OpCode2 + OpCode3 + ~OpCode5)
ALUsrc = NOT(OpCode0)
MemWrite = OpCode3
Branch = OpCode2

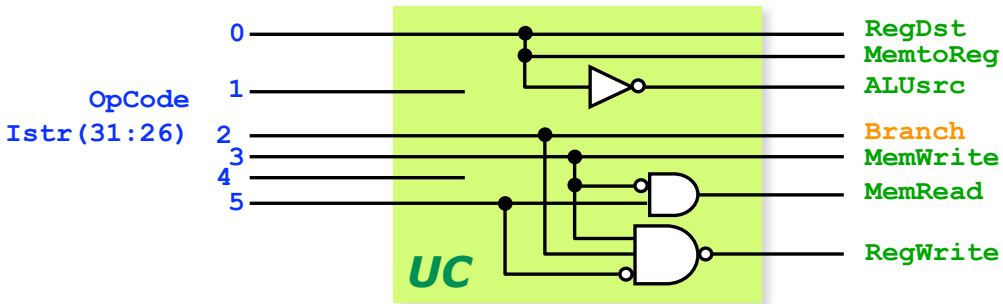
Sintesi di alcuni segnali di controllo

- Su ISA completo:

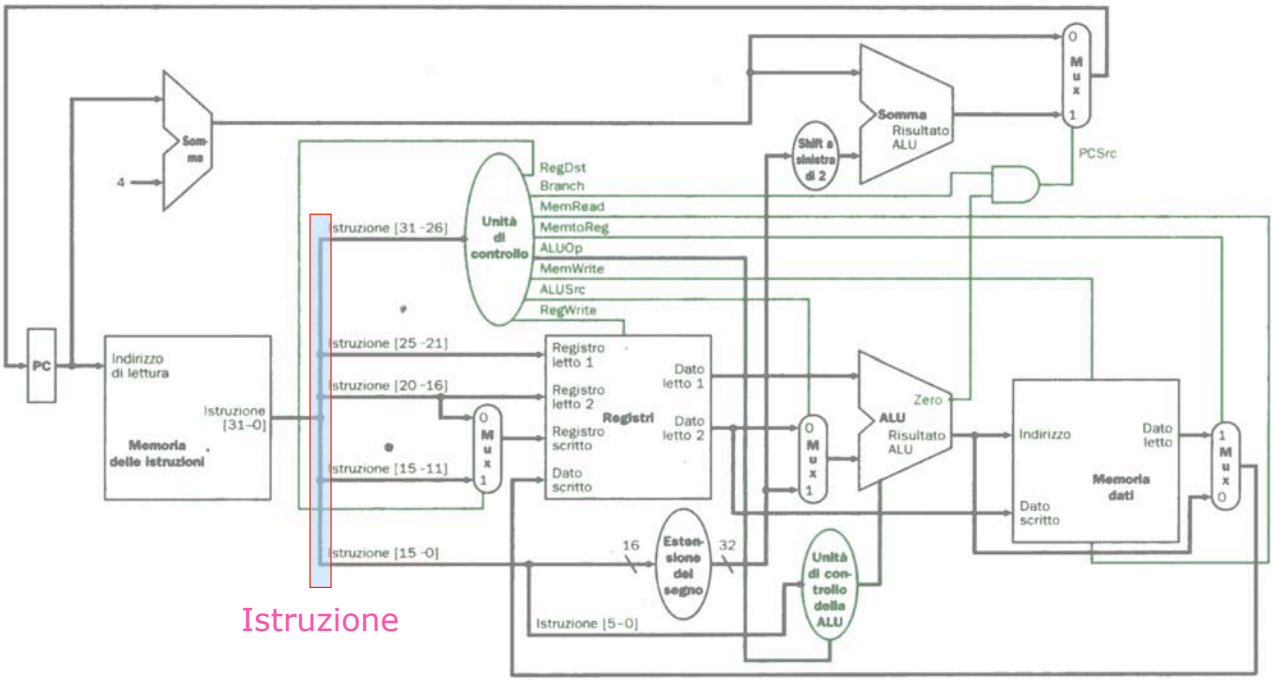




- Sul nostro ISA ridotto:



CPU Singolo Ciclo + UC

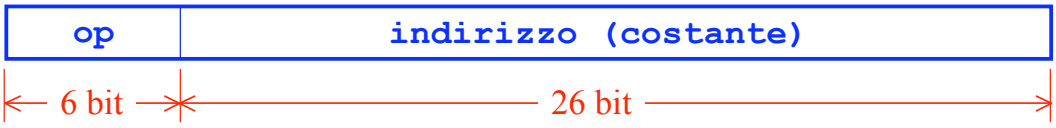


Istruzione

Patterson, Hennessy – Fig. 5.19

L'istruzione **jump** *(formato J)*

Formato J (jump)



- Indirizzo di salto determinato in due passi:
  - Calcolo parte bassa indirizzo: jad = (indirizzo * 4)
  - Deteminazione dell'indirizzo di salto: PC(31:28) | jad(27:0)

```
j 80040 :
```

Base (PC): 0100 1000 0011 0001 1011 1011 1011 10 11
Campo indirizzo: 1000 0110 0111 0000 0000 0001 00 00 =

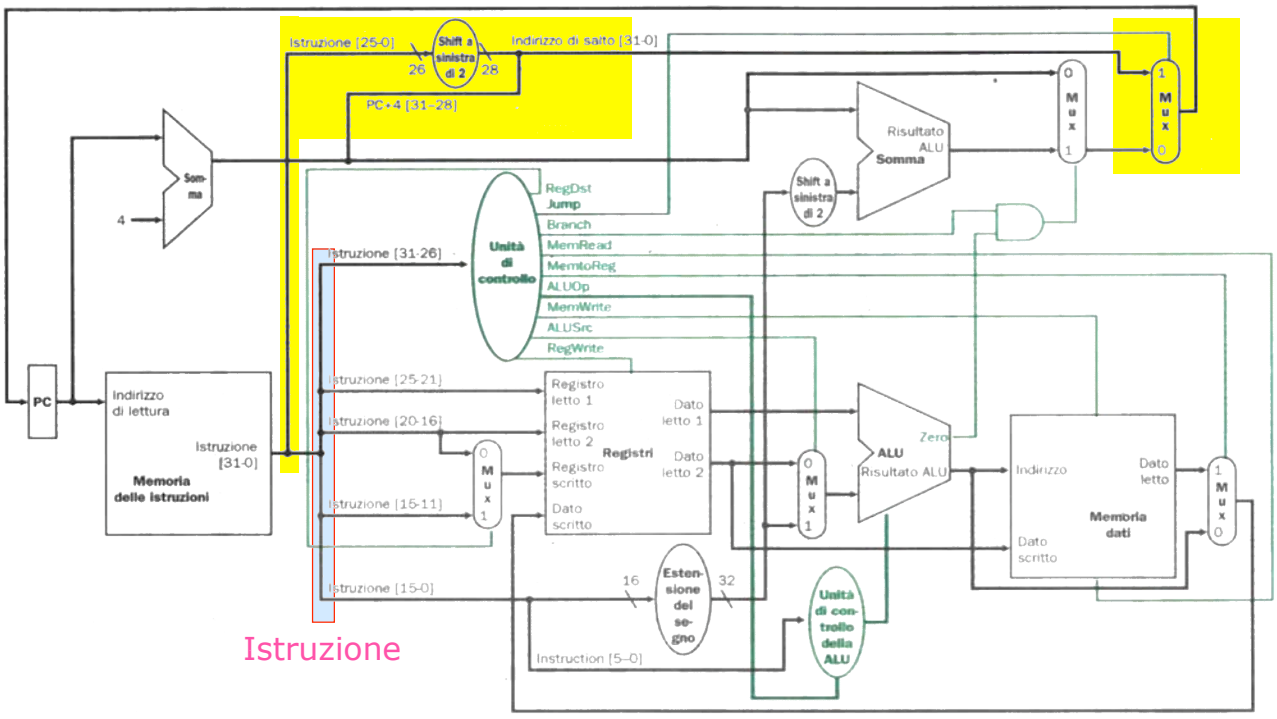
Indirizzo salto: 0100 1000 0110 0111 0000 0000 0001 00 00

Segnali di controllo

Segnale	Effetto quando è **negato (0)**	Effetto quando è **affermato (1)**
RegDst	Il numero del registro destinazione proviene dal campo **rd** (bit **15-11**)	Il numero del registro destinazione proviene dal campo **rt** (bit **20-16**)
RegWrite	Nessuno	Nel registro specificato su #RegWrite viene scritto il valore all'ingresso: ContenutoWrite
ALUSrc	Il secondo operando della ALU proviene dal campo **offset** (estesa a 32 bit)	Il secondo operando della ALU proviene dalla **II porta di lettura** del Register File
PCSrc	Il valore del PC viene sostituito dall'uscita del sommatore che calcola **PC+4**	Il valore del PC viene sostituito dall'uscita del sommatore che calcola l'**indirizzo di salto**
MemRead	Nessuno	Comando di **lettura** della memoria dati
MemWrite	Nessuno	Comando di **scrittura** della memoria dati
MemtoReg	Il valore inviato a ContenutoWrite del Register File proviene dalla **ALU**	Il valore inviato a ContenutoWrite del Register File proviene dalla **memoria**

Jump	PC viene impostato a **PC+4** oppure all'indirizzo **destinazione** della **branch**	PC viene impostato al valore ottenuto dal campo **dato** della **jump**

CPU + UC completa (aggiunta di jump)



CPU a Ciclo singolo

- CPU a CICLO SINGOLO:

  Ad ogni ciclo di clock viene eseguita un'istruzione completa.