Architettura degli Elaboratori

Il Processore:
l'Unità di Controllo Principale

Barbara Masucci
UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA
DIPARTIMENTO DI ECCELLENZA

0

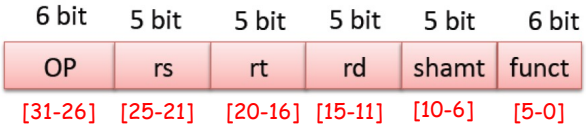
Punto della situazione

- Abbiamo visto come costruire l'Unità di Controllo della ALU del processore MIPS
  - Input: il segnale di controllo ALUOp (2 bit) e il campo funct (6 bit)
  - Output: i tre segnali di controllo (Operation2, Operation1, Operation0) per la ALU (non consideriamo AInvert)
- Oggi costruiremo l'Unità di Controllo Principale
  - Input: il codice operativo OpCode (6 bit) dell'istruzione da eseguire
  - Output: l'input ALUOp per l'Unità di Controllo della ALU e tutti gli altri segnali di controllo

1

Formato delle istruzioni

- Nel formato R

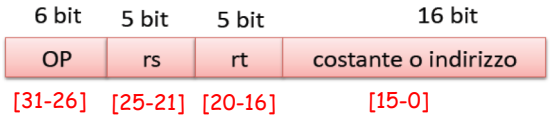


  - Il campo opcode è sempre 0 ed si trova nei bit [31-26]
  - Gli indirizzi dei due registri da leggere sono indicati dai campi
    - rs nei bit [25-21]
    - rt nei bit [20-16]
  - L'indirizzo del registro destinazione è nel campo rd [15-11]
  - L'operazione che la ALU dovrà eseguire è codificata nel campo funct, nei bit [5-0], e viene decodificata dall'unità di controllo della ALU
    - add, sub, and, or, slt

2

Formato delle istruzioni

- Nel formato I, per le istruzioni lw e sw

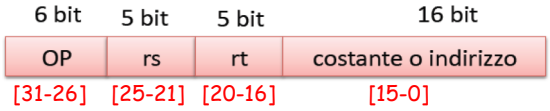


  - Il campo opcode si trova nei bit [31-26]
    - opcode =35_{10} per lw e opcode=43_{10} per sw
  - Sia per lw che per sw, il campo rs, nei bit [25-21], rappresenta l'indirizzo del registro base da leggere per calcolare l'indirizzo in memoria
  - Il campo rt, nei bit [20-16], contiene l'indirizzo
    - del registro da scrivere per l'istruzione lw
    - del registro da leggere, il cui contenuto va salvato in memoria, per l'istruzione sw
  - Il campo costante o indirizzo, nei bit [15-0], contiene l'offset

3

Formato delle istruzioni

Nel formato I, per l'istruzione beq

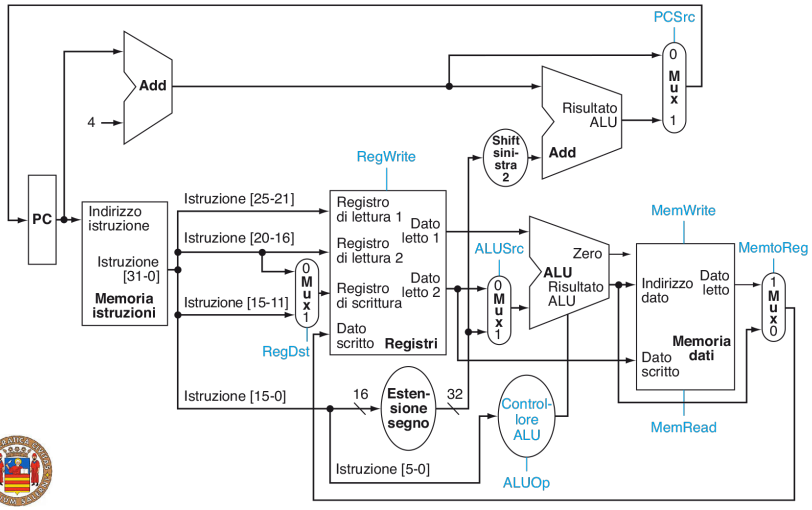


- Il campo opcode si trova nei bit [31-26] ed è uguale a 4_{10}
- I campi rs, nei bit [25-21], ed rt, nei bit [20-16], rappresentano gli indirizzi dei due registri da confrontare
- Il campo costante o indirizzo, nei bit [15-0], viene esteso a 32 bit, fatto scorrere a sx di 2 posizioni e sommato a PC+4 per il calcolo dell'indirizzo di salto

4

MIPS: implementazione di base

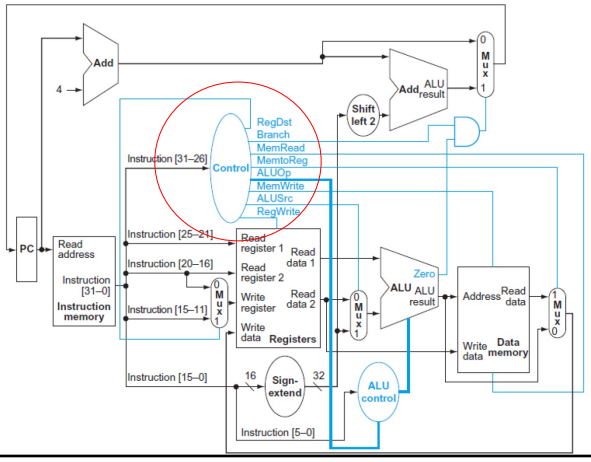
Allo schema di prima...



5

MIPS: implementazione di base

...aggiungiamo la Control Unit, che genera i segnali di controllo



6

Segnali di controllo a 1 bit

Nome del segnale	Effetto quando non asserito	Effetto quando asserito
RegDst	Il numero del registro di scrittura proviene dal campo rt (bit 20-16)	Il numero del registro di scrittura proviene dal campo rd (bit 15-11)
RegWrite	Nulla	Il dato viene scritto nel register file nel registro individuato dal numero del registro di scrittura
ALUSrc	Il secondo operando della ALU proviene dalla seconda uscita del register file (Dato letto 2)	Il secondo operando della ALU proviene dall'estensione del segno dei 16 bit meno significativi dell'istruzione
PCSrc	Nel PC viene scritta l'uscita del sommatore che calcola il valore di PC + 4	Nel PC viene scritta l'uscita del sommatore che calcola l'indirizzo di salto
MemRead	Nulla	Il dato della memoria nella posizione puntata dall'indirizzo viene inviato in uscita sulla linea «dato letto»
MemWrite	Nulla	Il contenuto della memoria nella posizione puntata dall'indirizzo viene sostituito con il dato presente sulla linea «dato scritto»
MemtoReg	Il dato inviato al register file per la scrittura proviene dalla ALU	Il dato inviato al register file per la scrittura proviene dalla Memoria Dati

L'effetto dei 7 segnali di controllo a 1 bit

7

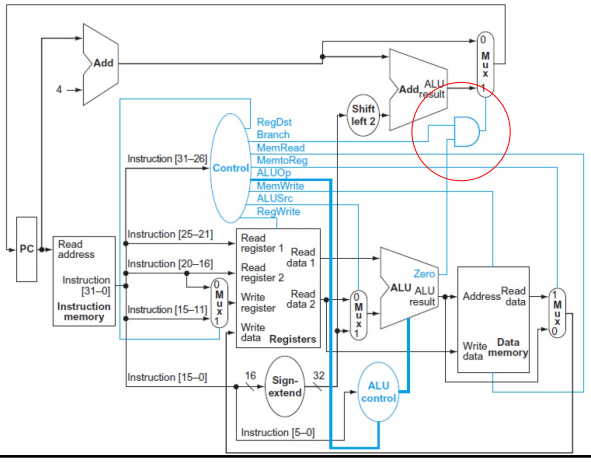
PCSrc

- Consideriamo il segnale di controllo PCSrc
  - PCSrc deve valere 1 quando l'istruzione è beq e l'uscita Zero della ALU è vera
  - Per generare PCSrc viene usata una porta AND i cui input sono
    - L'uscita Zero della ALU
    - Un nuovo segnale di controllo, chiamato Branch

8

Segnali di controllo

Generazione del segnale di controllo PCSrc



9

Segnali di controllo

- In definitiva, i segnali di controllo sono 9
  - I 7 segnali a 1 bit: RegDst, RegWrite, ALUSrc, MemRead, PCSrc, MemWrite, MemtoReg
  - Il segnale Branch a 1 bit
  - Il segnale ALUOp a 2 bit ALUOp1, ALUOp0
- Questi segnali vengono impostati dall'Unità di Controllo in base al codice operativo dell'istruzione (6 bit)
  - Ciascun segnale può valere 0, 1, o X (indifferente) in funzione del valore del codice operativo

Istruzione	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
Tipo R	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

10

Segnali di controllo: Istruzioni in formato R

- Analizziamo la riga per le istruzioni in formato R

Istruzione	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
Tipo R	1	0	0	1	0	0	0	1	0

- RegDst=1 perché il registro destinazione è nel campo rd
- AluSrc=0 perché il secondo operando viene dal campo rt
- MemtoReg=0 perché il risultato proviene dalla ALU
- RegWrite=1 perché il risultato va scritto in un registro
- MemRead=MemWrite=0 perché non c'è accesso alla memoria dati
- Branch=0 perché l'indirizzo della prossima istruzione è PC+4
- ALUOp1=1 e ALUOp0=0 sono i segnali di controllo richiesti per le istruzioni di tipo R

11

Segnali di controllo: Istruzione lw

- Analizziamo la riga per l'istruzione lw

Istruzione	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
lw	0	1	1	1	1	0	0	0	0

- RegDst=0 perché il registro destinazione è nel campo rt
- AluSrc=1 perché il secondo operando viene dall'estensione a 32 bit
- MemtoReg=1 perché la word proviene dalla memoria dati
- RegWrite=1 perché la word va scritta in un registro
- MemRead=1 perché c'è accesso in lettura alla memoria dati
- MemWrite=0 perché non c'è accesso in scrittura alla memoria dati
- Branch=0 perché l'indirizzo della prossima istruzione è PC+4
- ALUOp1=ALUOp0=0 sono i segnali di controllo richiesti per lw

12

Segnali di controllo: Istruzione sw

- Analizziamo la riga per l'istruzione sw

Istruzione	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
sw	X	1	X	0	0	1	0	0	0

- RegDst=X perché RegWrite=0
- AluSrc=1 perché il secondo operando viene dall'estensione a 32 bit
- MemtoReg=X perché RegWrite=0
- RegWrite=0 perché non viene scritto nulla nel register file
- MemRead=0 perché non c'è accesso in lettura alla memoria dati
- MemWrite=1 perché c'è accesso in scrittura alla memoria dati
- Branch=0 perché l'indirizzo della prossima istruzione è PC+4
- ALUOp1=ALUOp0=0 sono i segnali di controllo richiesti per sw

13

Segnali di controllo: Istruzione beq

- Infine, analizziamo la riga per l'istruzione beq

Istruzione	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
beq	X	0	X	0	0	0	1	0	1

- RegDst=X perché RegWrite=0
- AluSrc=0 perché il secondo operando viene dal campo rt
- MemtoReg=X perché RegWrite=0
- RegWrite=0 perché non viene scritto nulla nel register file
- MemRead=0 perché non c'è accesso in lettura alla memoria dati
- MemWrite=0 perché non c'è accesso in scrittura alla memoria dati
- Branch=1 perché l'indirizzo della prossima istruzione dipende anche dall'indirizzo del salto e dall'ouput Zero della ALU
- ALUOp1=0 e ALUOp0=1 sono i segnali di controllo richiesti per beq

14

Istruzioni in formato R

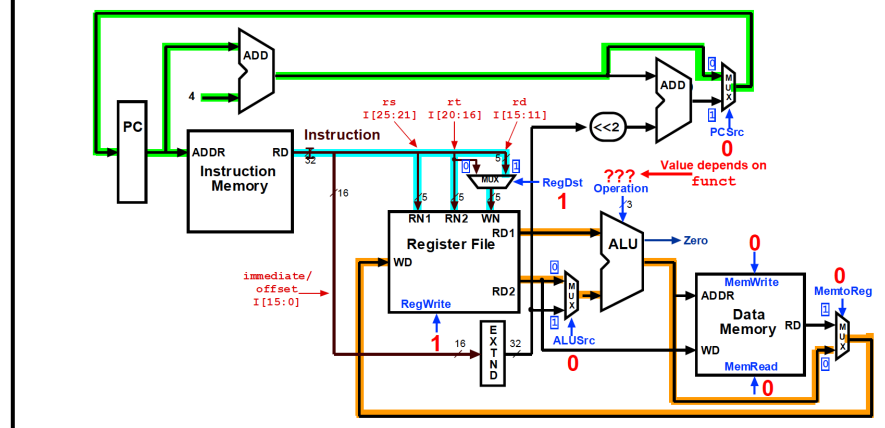
- Vediamo come vengono eseguite le istruzioni in formato R
- L'istruzione add $t1, $t2, $t3 richiede 4 passi
  - L'istruzione viene prelevata dalla memoria istruzioni e il PC viene incrementato di 4
  - Il contenuto dei registri $t2 e $t3 viene letto dal register file mentre l'Unità di Controllo calcola il valore da attribuire ai segnali di controllo
  - La ALU elabora i dati letti dal register file; il campo funct viene utilizzato per selezionare l'operazione della ALU
  - Il risultato del calcolo della ALU viene scritto nel registro destinazione $t1 all'interno del register file

Tutto questo avviene in un solo ciclo di clock

15

Istruzioni in formato R

Segnali di controllo



Istruzione	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
Tipo R	1	0	0	1	0	0	0	1	0

16

Istruzione lw

- L'istruzione lw $t1, offset($t2) richiede 5 passi
  - L'istruzione viene prelevata dalla memoria istruzioni e il PC viene incrementato di 4
  - Il contenuto del registro $t2 viene letto dal register file mentre l'Unità di Controllo calcola il valore da attribuire ai segnali di controllo
  - La ALU somma il valore letto dal register file e l'offset esteso a 32 bit
  - Il risultato del calcolo della ALU viene usato come indirizzo per la memoria dati
  - Il dato proveniente dalla memoria dati viene scritto nel registro destinazione $t1 all'interno del register file

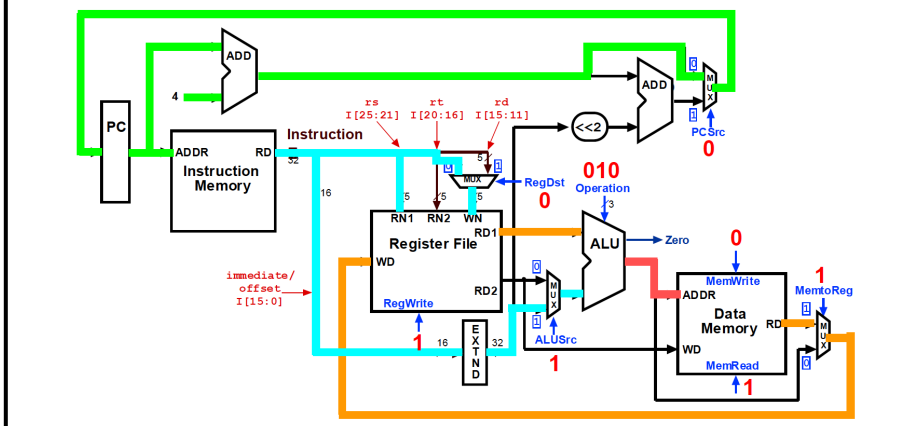
6 bit	5 bit	5 bit	16 bit
OP	rs	rt	costante o indirizzo

Tutto questo avviene in un solo ciclo di clock

17

Istruzione lw

Segnali di controllo



Istruzione	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
lw	0	1	1	1	1	0	0	0	0

18

Istruzione sw

- L'istruzione sw $t1, offset($t2) richiede 4 passi
  - L'istruzione viene prelevata dalla memoria istruzioni e il PC viene incrementato di 4
  - Il contenuto dei registri $t1 e $t2 viene letto dal register file mentre l'Unità di Controllo calcola il valore da attribuire ai segnali di controllo
  - La ALU somma il contenuto del registro $t2 letto dal register file e l'offset esteso a 32 bit
  - Il risultato del calcolo della ALU viene usato come indirizzo per la memoria dati e il dato proveniente dal registro $t1 viene scritto nella memoria dati

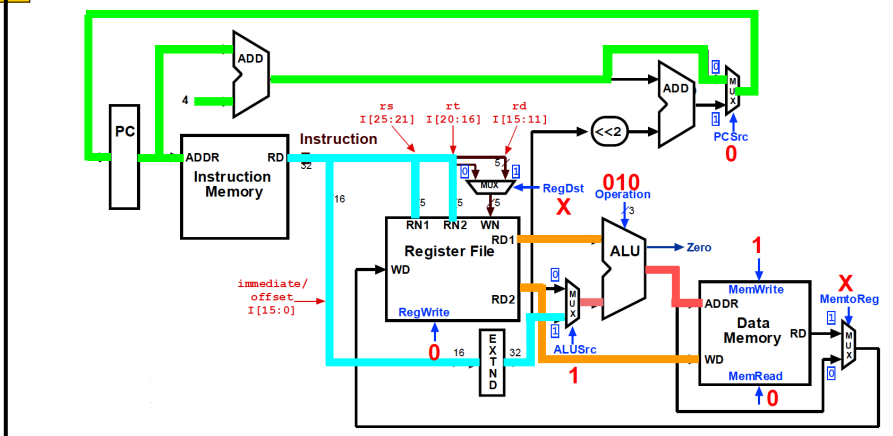
6 bit	5 bit	5 bit	16 bit
OP	rs	rt	costante o indirizzo

Tutto questo avviene in un solo ciclo di clock

19

Istruzione sw

Segnali di controllo



Istruzione	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
SW	X	1	X	0	0	1	0	0	0

20

Istruzione beq

- L'istruzione beq $t1, $t2, offset richiede 4 passi
  - L'istruzione viene prelevata dalla memoria istruzioni e il PC viene incrementato di 4
  - Il contenuto dei registri $t1 e $t2 viene letto dal register file mentre l'Unità di Controllo calcola il valore da attribuire ai segnali di controllo
  - La ALU esegue la sottrazione tra i contenuti dei due registri; il valore PC+4 viene sommato all'offset (esteso a 32 bit e shiftato di due posizioni a sinistra), determinando l'indirizzo del salto
  - L'output Zero dell'ALU viene utilizzato per determinare da quale sommatore prendere l'indirizzo successivo da scrivere nel PC

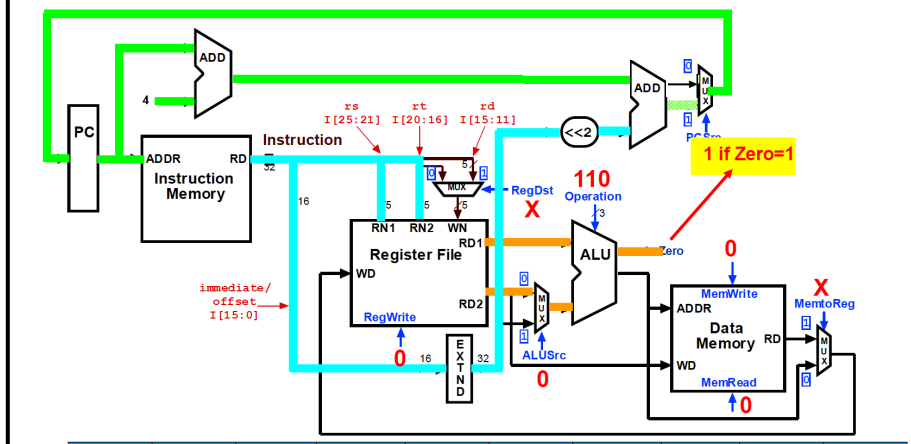
6 bit	5 bit	5 bit	16 bit
OP	rs	rt	costante o indirizzo

Tutto questo avviene in un solo ciclo di clock

21

Istruzione beq

Segnali di controllo



Istruzione	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
beq	X	0	X	0	0	0	1	0	1

22

Unità di controllo

Implementazione

- Passiamo ora all'implementazione dell'Unità di Controllo a partire dalla tabella

Istruzione	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
Tipo R	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

- L'unità di controllo ha come
  - Input: il codice operativo dell'istruzione (6 bit)
    - CodOp5 CodOp4 CodOp3 CodOp2 CodOp1 CodOp0
  - Output: i segnali di controllo in tabella
- Per ciascuno degli output scriviamo la tavola di verità, usando la codifica binaria dei codici operativi

23

Segnali di controllo:
Tavola di verità

0_{10} 35_{10} 43_{10} 4_{10}

Input o Output	Nome del segnale	Formato R	lw	sw	beq
Input	CodOp5	0	1	1	0
	CodOp4	0	0	0	0
	CodOp3	0	0	1	0
	CodOp2	0	0	0	1
	CodOp1	0	1	1	0
	CodOp0	0	1	1	0
Output	RegDst	1	0	X	X
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp2	0	0	0	1

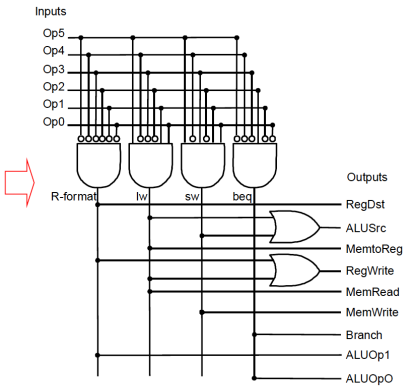
24

Unità di controllo
Implementazione

Tabella di verita' dell'unita' di Controllo

	Segnale	form. R	lw	sw	beq
Inputs =Opcode	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Outputs	RegDst	1	0	x	x
	ALUSrc	0	1	1	0
	MemtoReg	0	1	x	x
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOP0	0	0	0	1

Rete Combinatoria realizzabile tramite PLA

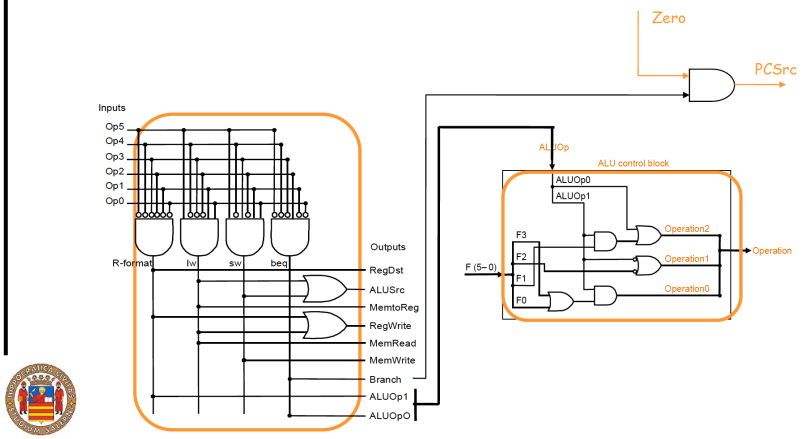


25

Unità di controllo
Implementazione

Rete combinatoria definita da una coppia di tabelle di verita'

Controllo Principale e Controllo ALU



26

Implementazione a Ciclo Singolo

- L'implementazione a ciclo singolo non viene utilizzata perché è inefficiente
  - Motivo: la lunghezza del ciclo di clock (uguale per tutte le istruzioni) è determinata dal cammino di elaborazione più lungo all'interno del processore
  - L'istruzione più lenta è la lw
    - Utilizza 5 unità funzionali in sequenza: memoria istruzioni, register file, ALU, memoria dati, register file
  - Questo comporta un ciclo di clock molto lungo, mentre molte istruzioni possono essere eseguite in tempo minore
- Vedremo una seconda implementazione, più efficiente, basata su pipeline
  - Consente di sovrapporre l'esecuzione di diverse istruzioni

27

Riepilogo e riferimenti

- Unità di Controllo principale
  - [PH] par. 4.4 (seconda parte)

28