Il processore: unità di elaborazione dati e unità di controllo

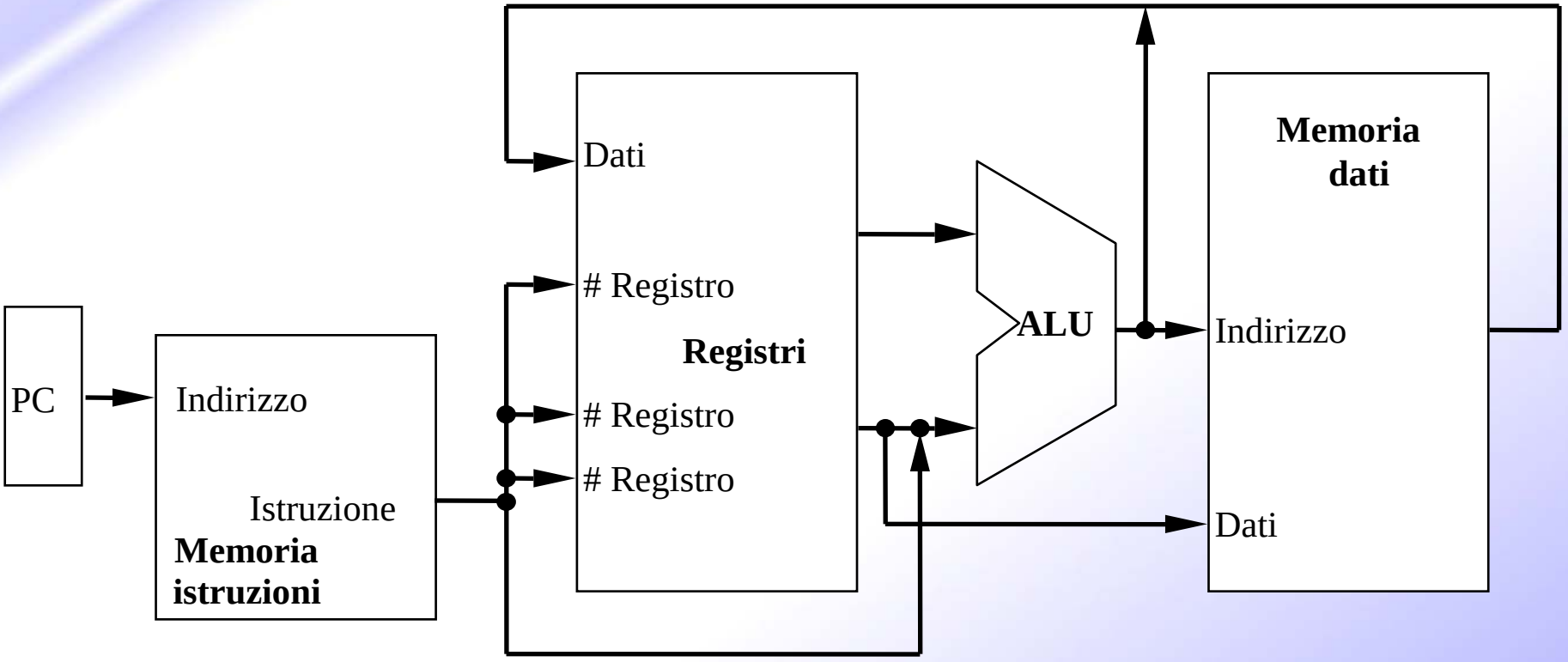
Introduzione

- Implementazione del MIPS semplificato:
  - istruzioni di accesso alla memoria:
    - lw e sw
  - istruzioni logico-aritmetiche:
    - add, sub, and, or, e slt
  - istruzioni di salto:
    - beq e j

Una visione generale

- Implementazione generica di un'istruzione:
  - usa il contatore di programma (PC) per ottenere l'indirizzo dell'istruzione
  - prende l'istruzione dalla memoria (*fetch*)
  - legge uno o due registri
  - usa l'istruzione per decidere cosa fare esattamente
- Tutte le istruzioni usano l'ALU dopo aver letto i registri:
  - per calcolare l'indirizzo in memoria
  - per eseguire l'operazione logico-aritmetica
  - per effettuare il test
- Dopo aver usato l'ALU le istruzioni si differenziano

Una visione ad alto livello

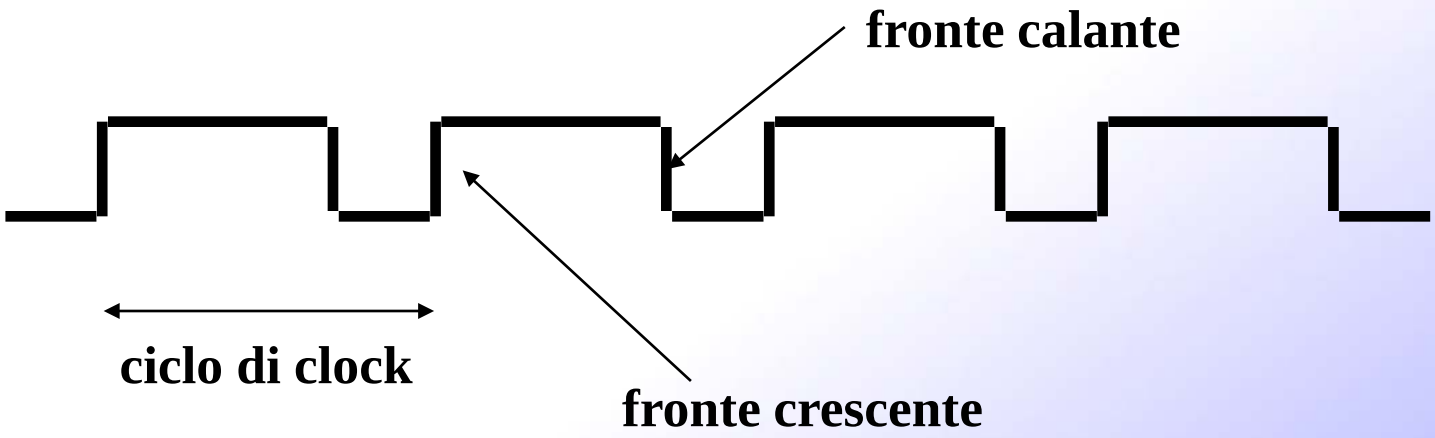


Convenzioni logiche

- Segnale affermato = logicamente vero
- Due tipi di unità funzionali:
  - elementi che operano sui valori dei dati (combinatori)
    - le uscite dipendono solo dagli ingressi
    - ALU
  - elementi che contengono uno stato (sequenziali)
    - almeno due ingressi (valore e clock) ed un'uscita (valore precedente)
    - flip-flop di tipo D
    - memorie e registri

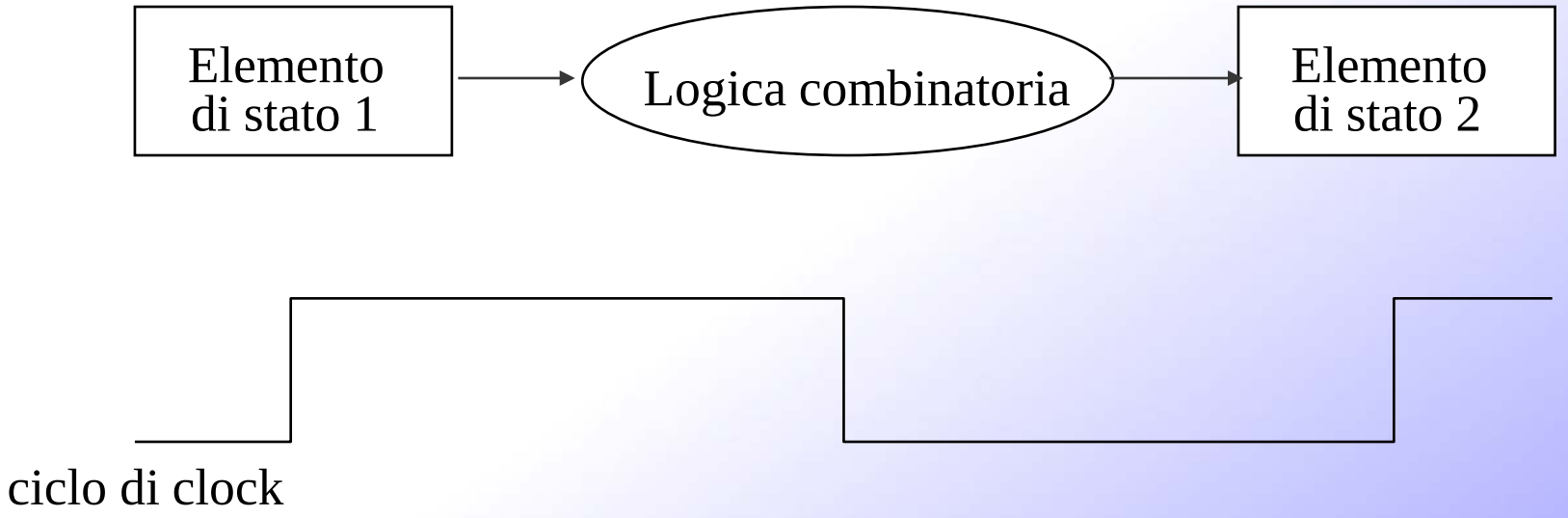
Temporizzazione

- Metodologia per decidere quando un elemento che contiene uno stato deve essere aggiornato
- Basata sulle transizioni (fronti dei segnali)



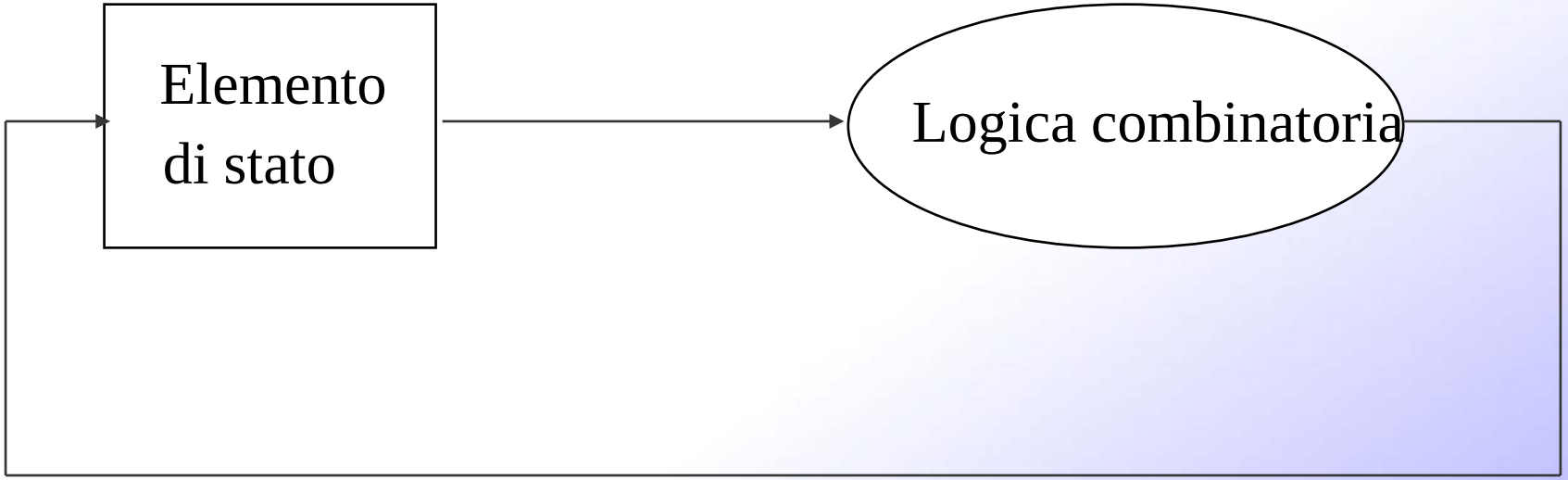
Durata minima del ciclo di clock

- Legge valore del primo elemento sequenziale
- Valore attraversa circuito combinatorio
- Valore raggiunge il secondo elemento sequenziale



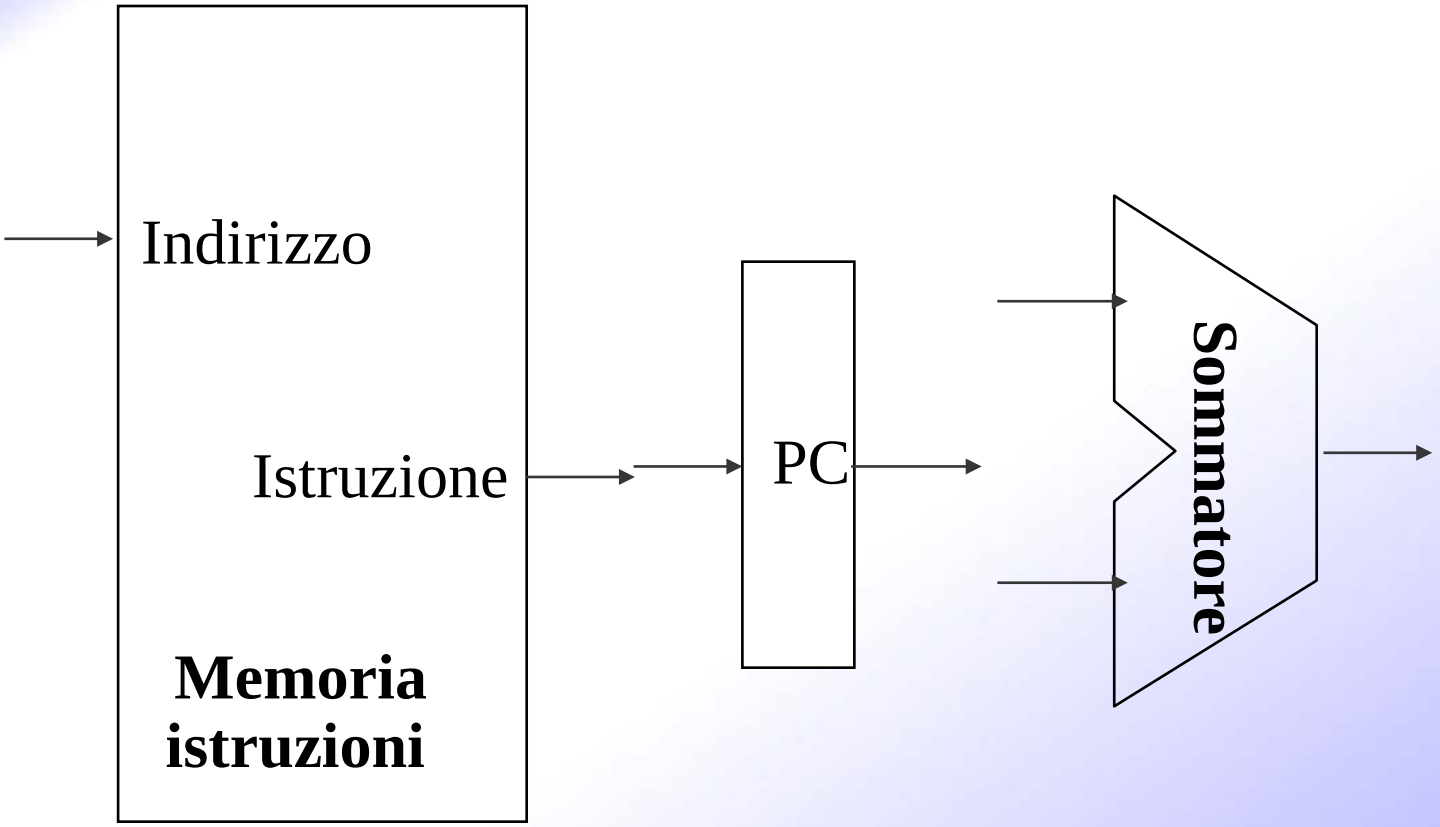
Vantaggio dell'uso dei fronti

- Permette di leggere e scrivere un elemento sequenziale nello stesso ciclo di clock
  - indifferente se scrittura o lettura avviene su fronte calante

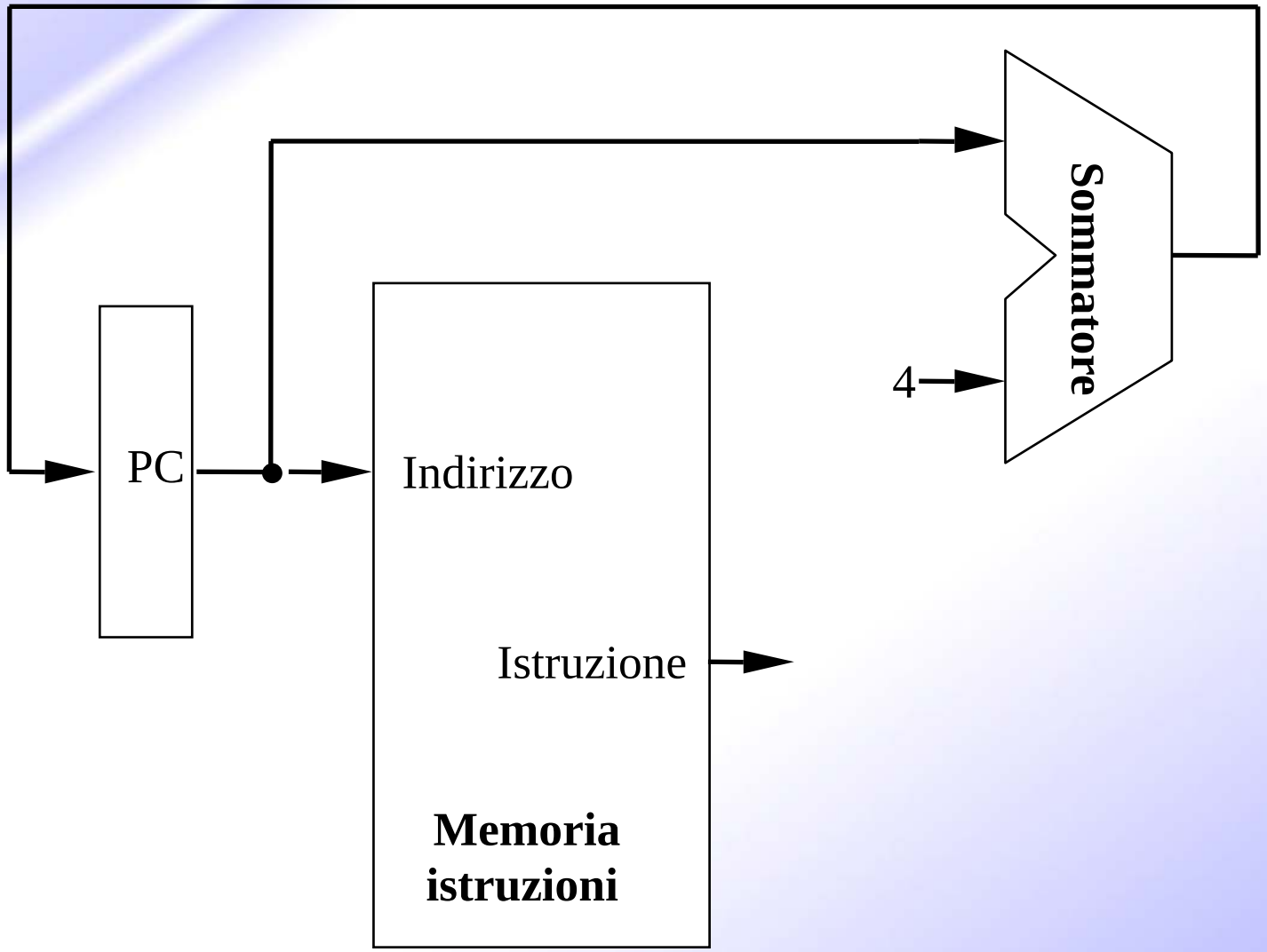


Di cosa abbiamo bisogno

- Per calcolare la prossima istruzione:

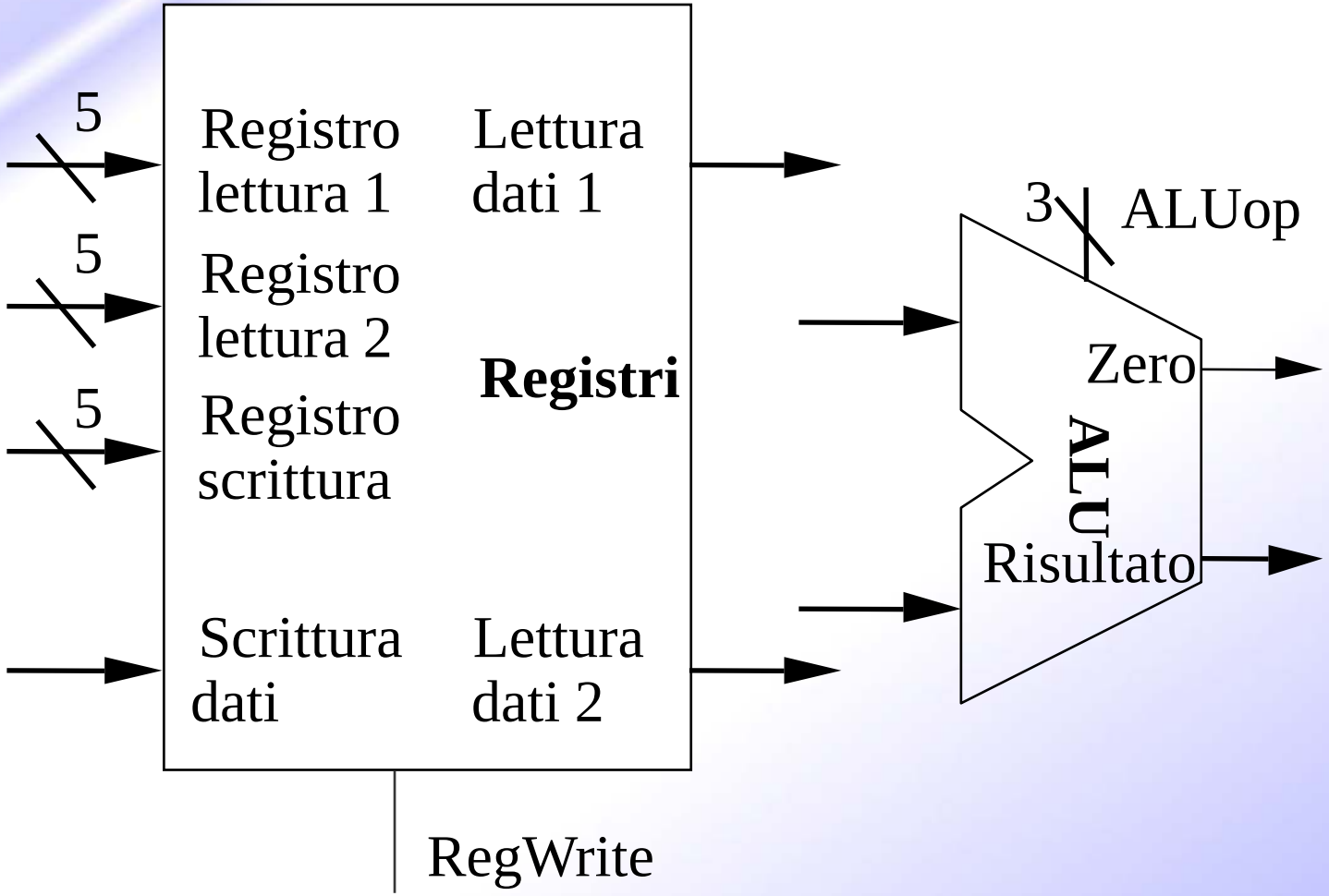


Il cammino di dati (1)

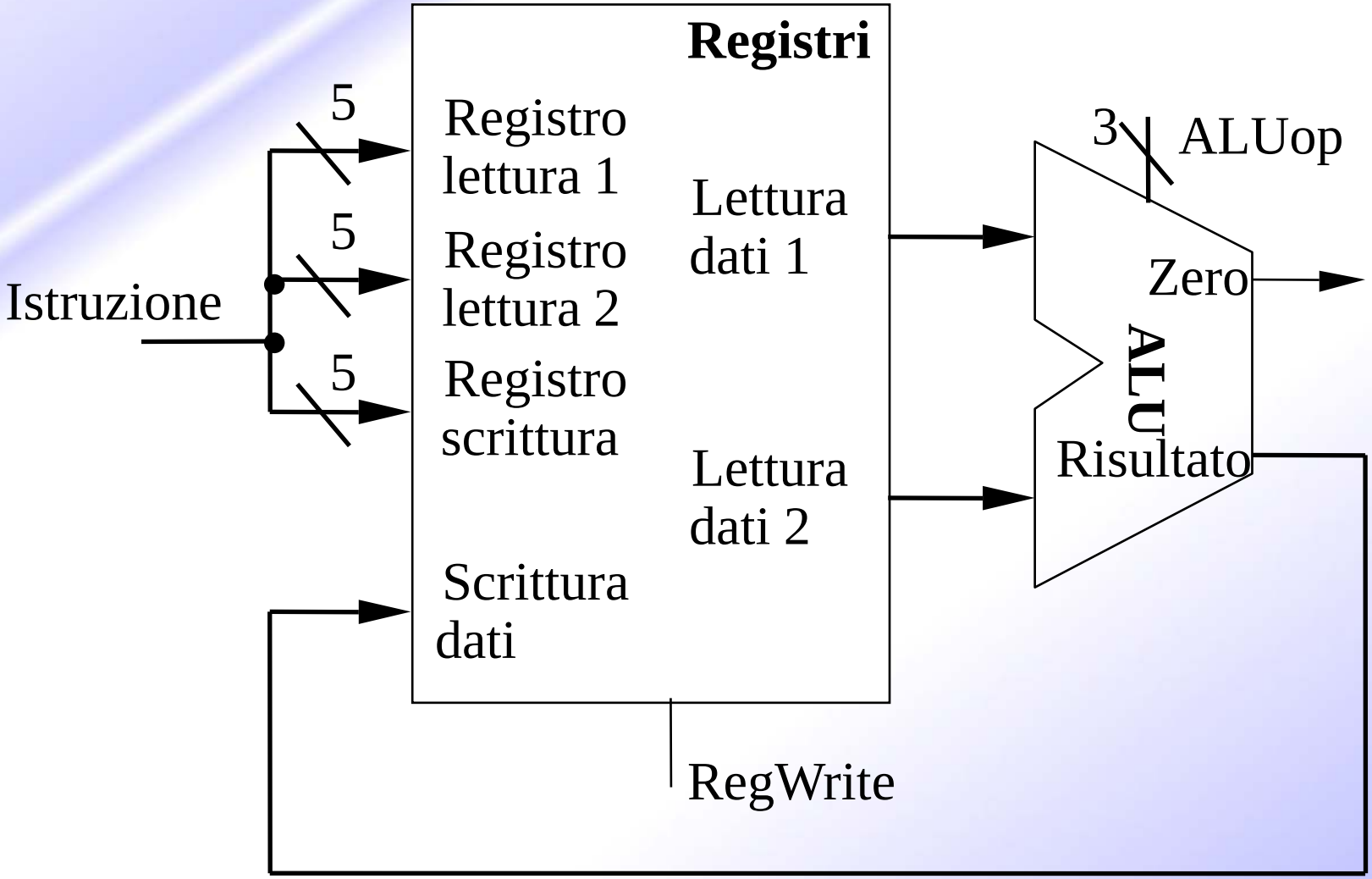


Di cosa abbiamo bisogno

- Per eseguire un'istruzione logica-aritmetica:

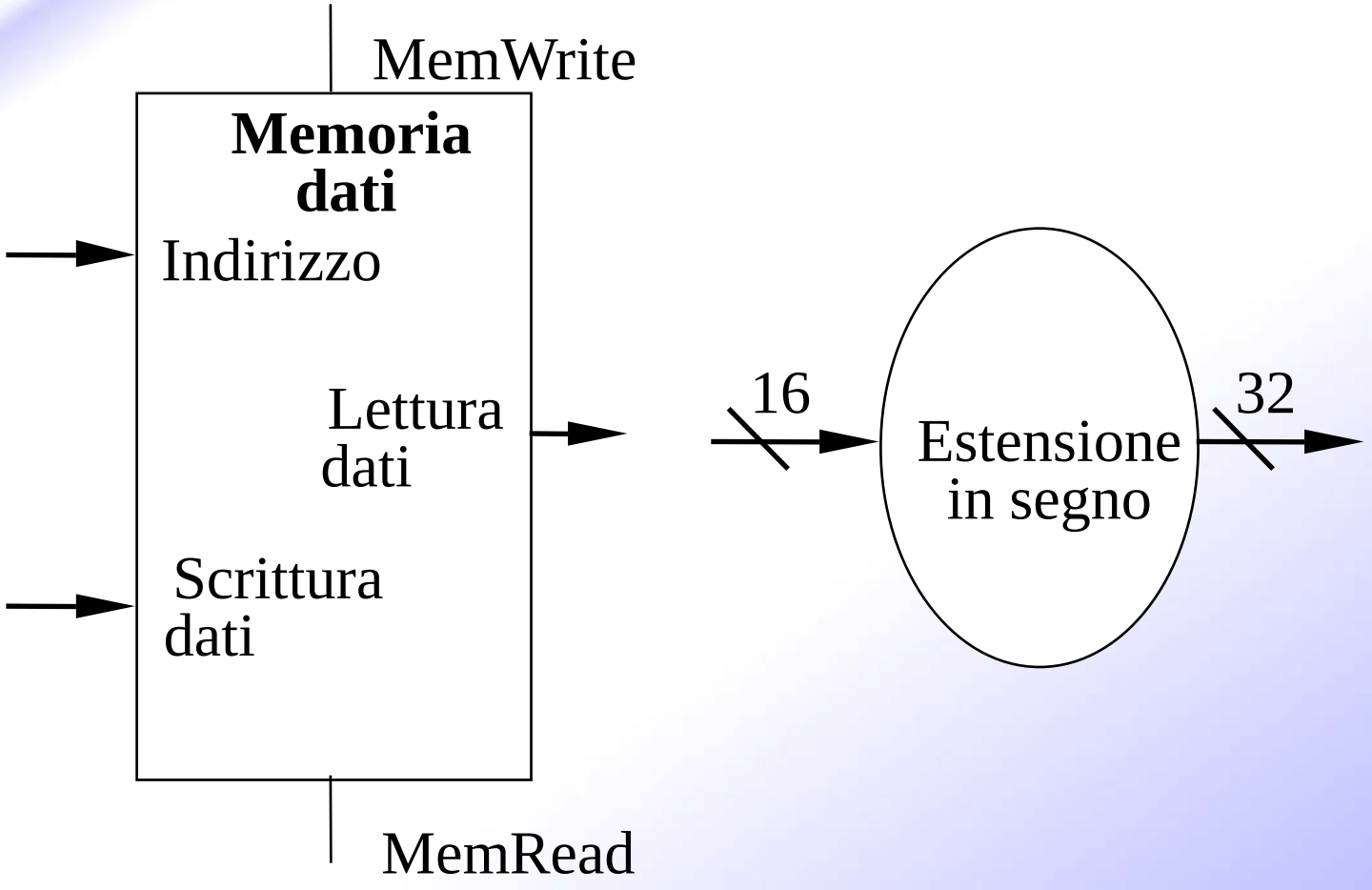


Il cammino di dati (2)

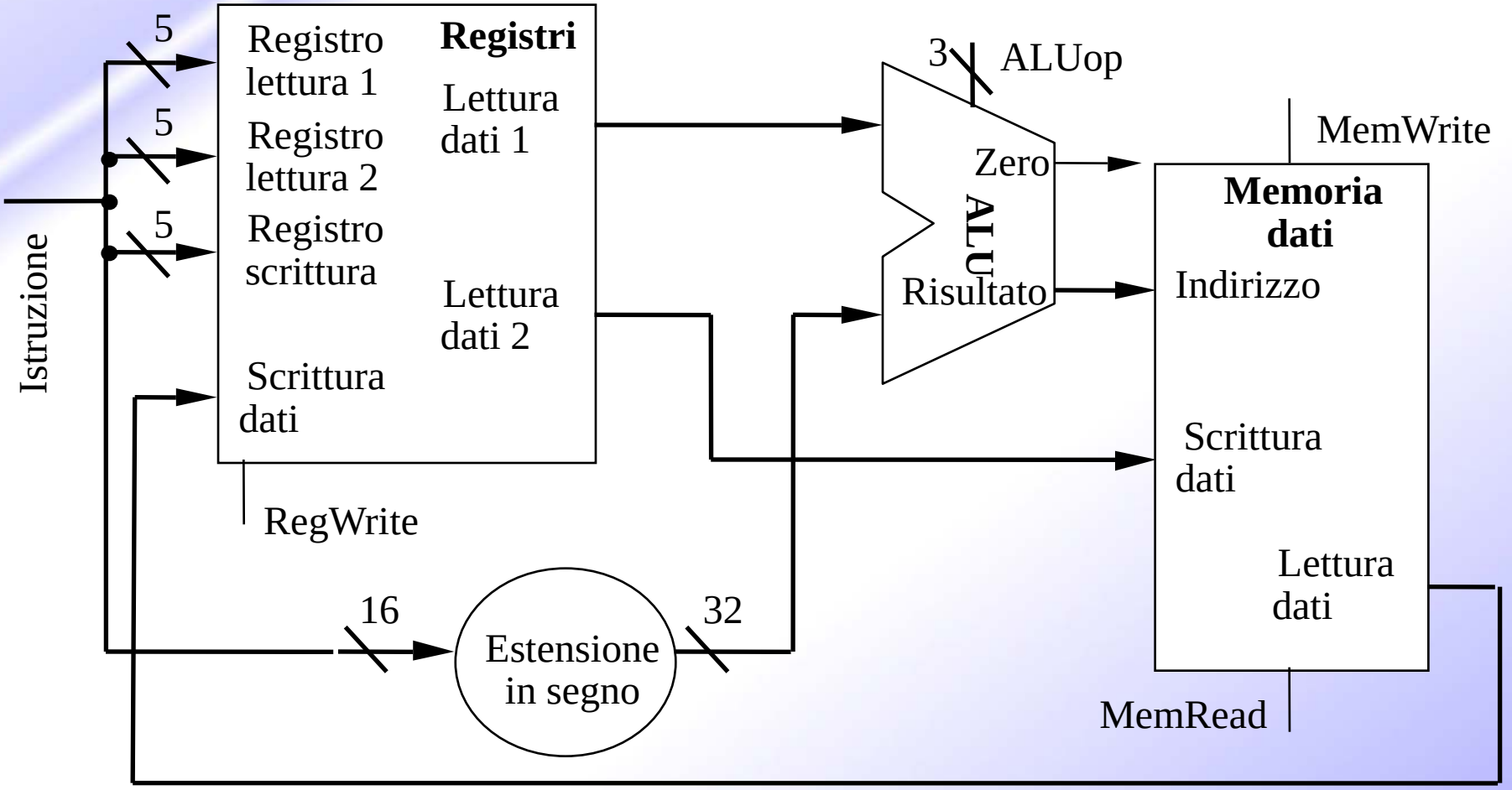


Di cosa abbiamo bisogno

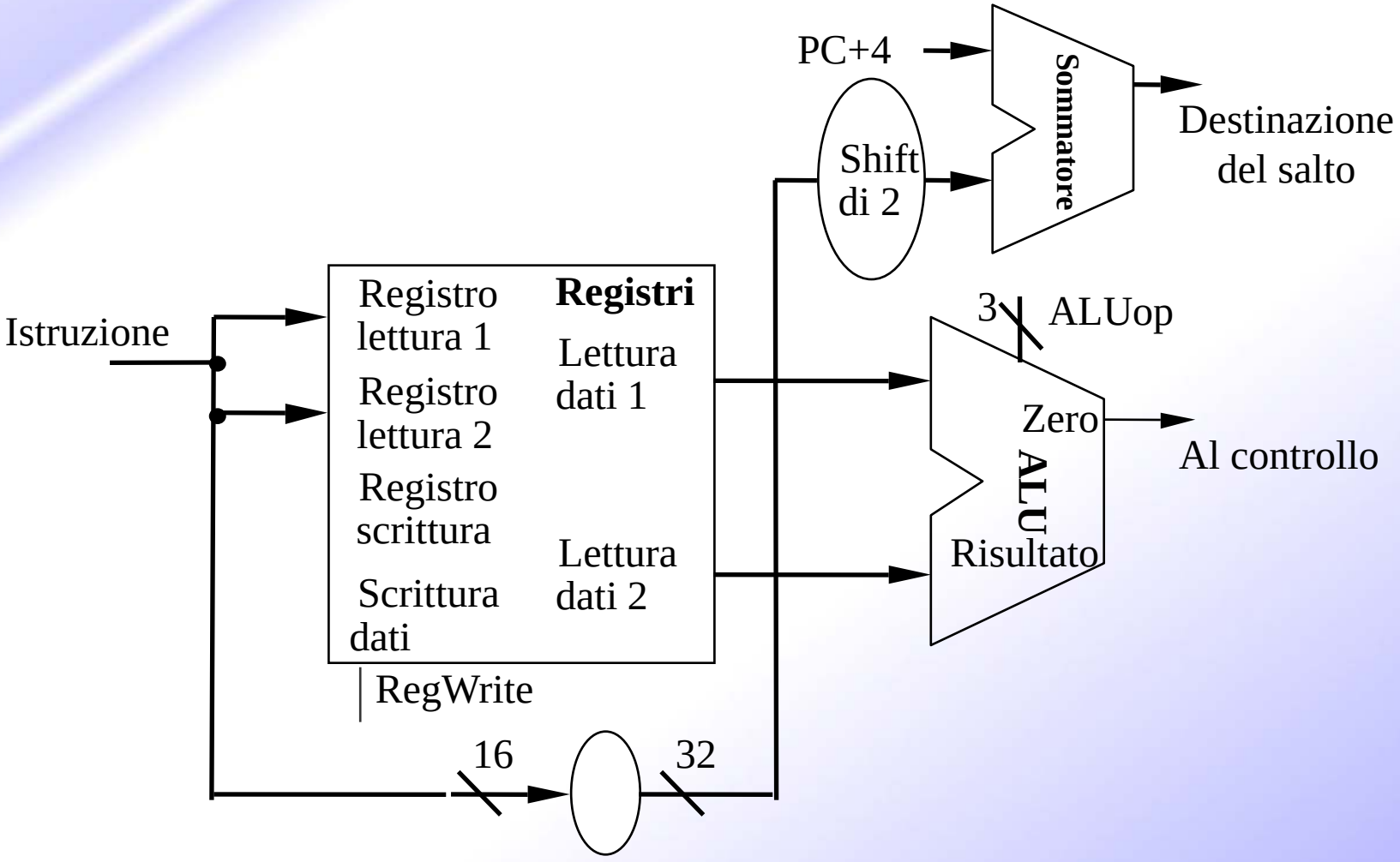
- Per eseguire lettura/scrittura in memoria:



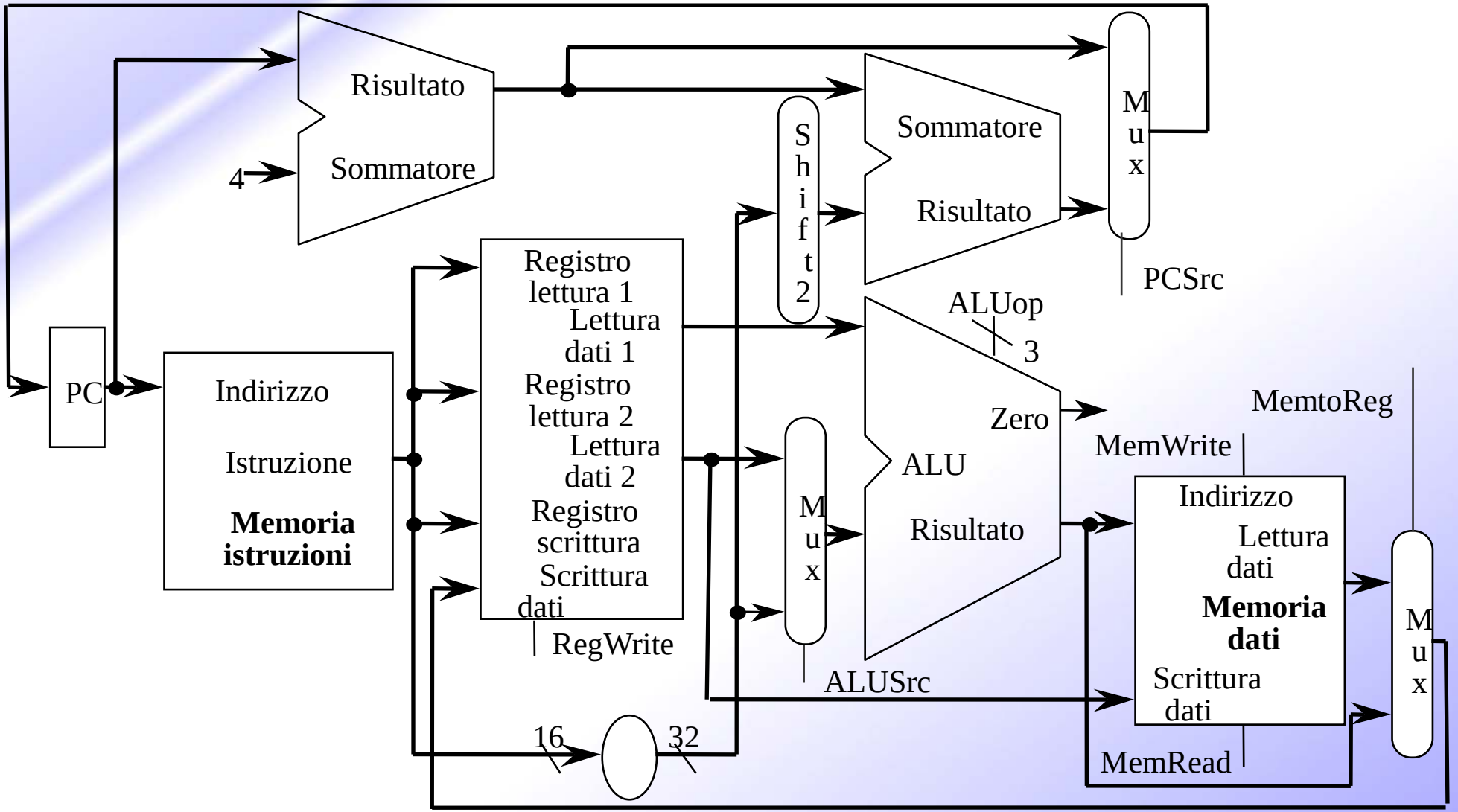
Il cammino di dati (3)



Il cammino di dati (4)



Il cammino dei dati



Formati delle istruzioni

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
- Registri da leggere sempre rs e rt (bit 25-21 e 20-16)
- Registro base per lw/sw sempre rs (bit 25-21)
- Offset sempre in posizione 15-0
- Registro destinazione rd (15-11) per istruzioni di tipo R, rt (20-16) per istruzioni lw (necessario mux)

Il controllo

- Seleziona le operazioni da eseguire
  - ALU, lettura/scrittura
- Controlla il flusso dei dati
  - ingressi dei multiplexer
- Riceve l'informazione dei 32 bit dell'istruzione
  - Esempio: lw $1, 100($2)

35	2	1	100

op	rs	rt	offset di 16 bit

Controllo dell'ALU

- Cosa deve fare l'ALU con l'istruzione corrente
  - Bit di controllo in Input alla ALU che detrminano la operazione eseguita
    - 000 and
    - 001 or
    - 010 add
    - 110 sub
    - 111 slt

000000	10001	10010	01000	00000	100000

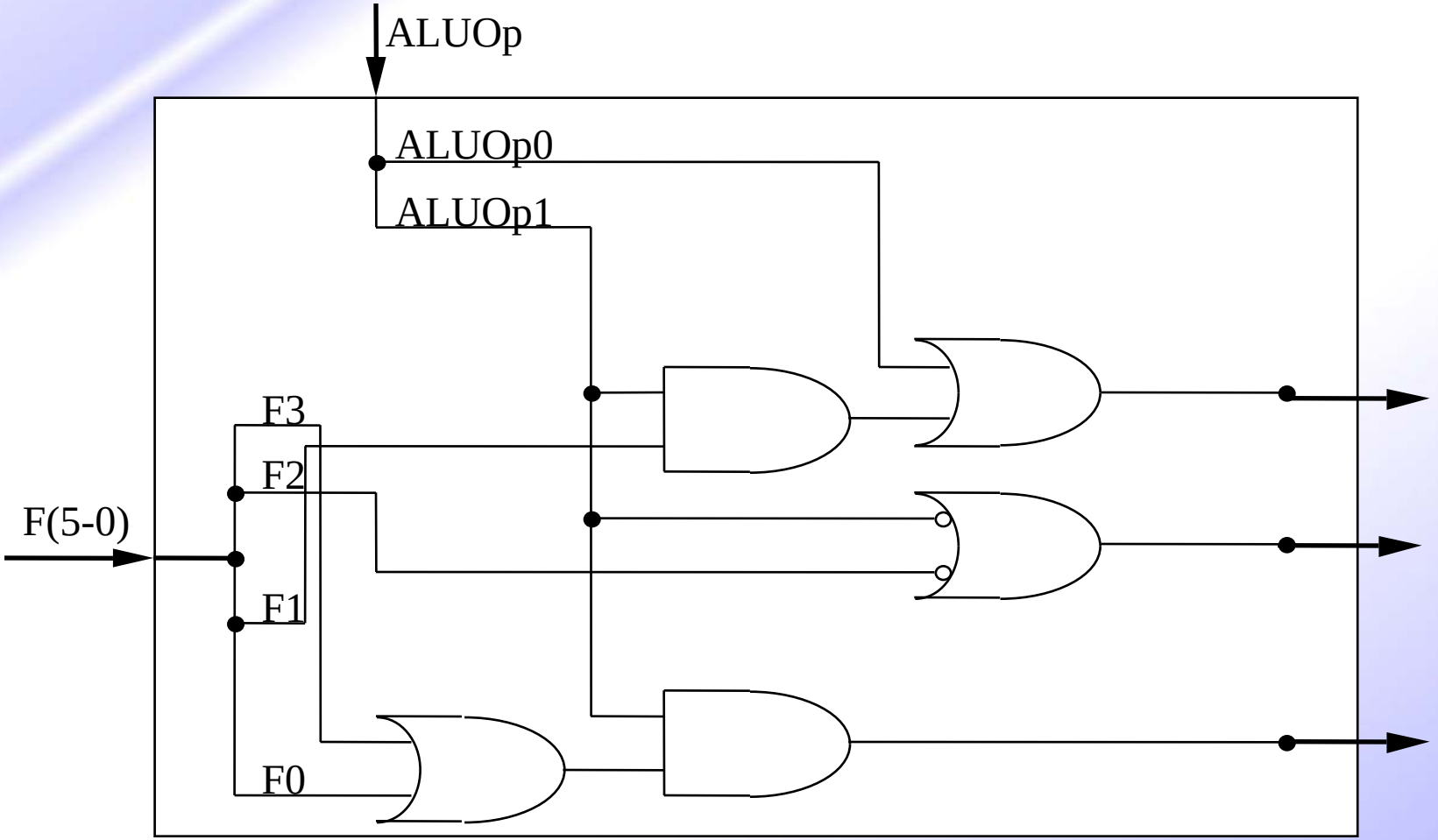
op	rs	rt	rd	shamt	funct

- Esempio: add $8,$17,$18

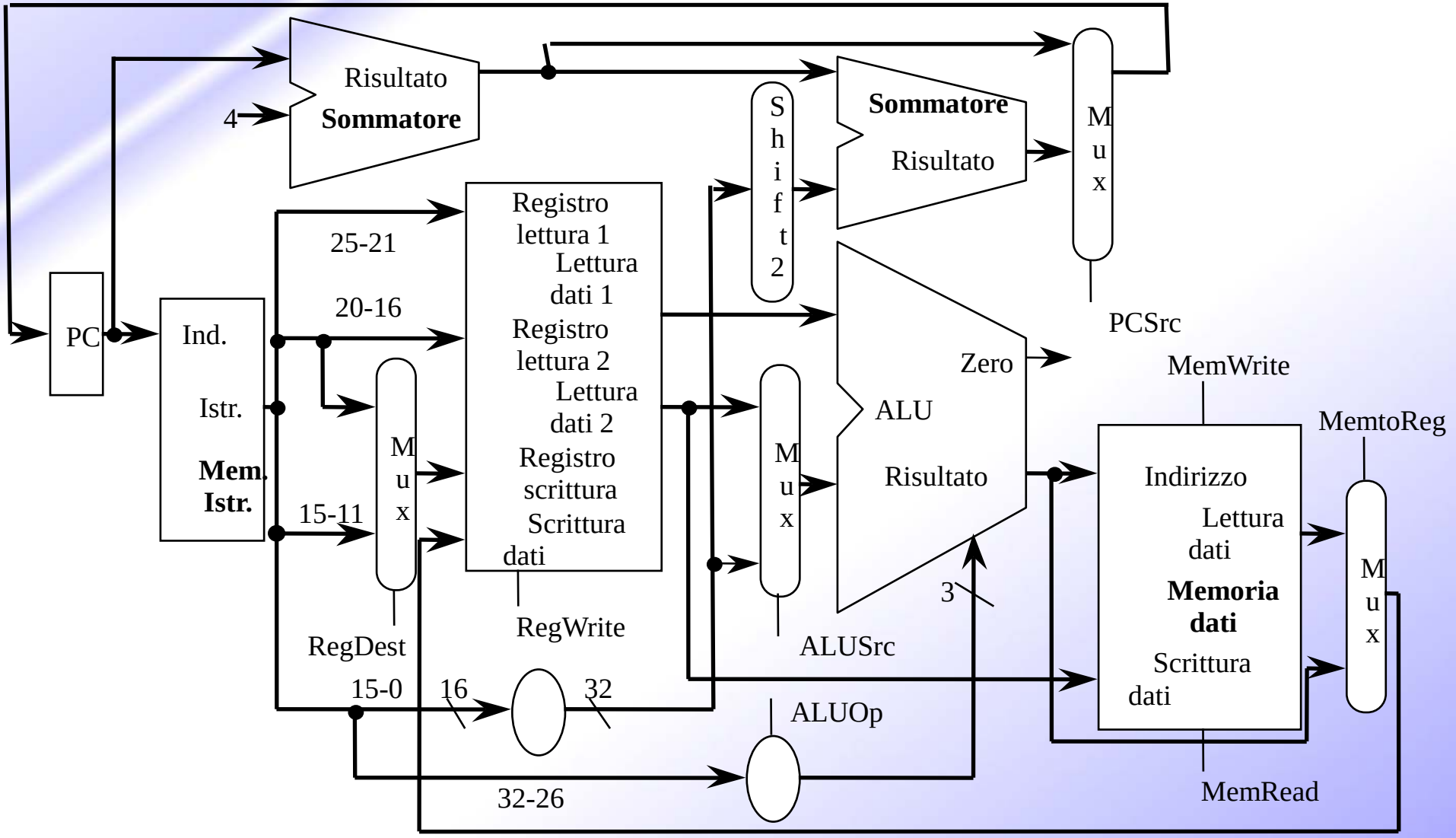
- I Bit di controllo dell'ALU vengono definiti in base a:
  - tipo di istruzione: 00 = lw, sw, 01 = beq, 10 = logico-aritmetica
  - codice della funzione (campo funct)
- Lo descriviamo mediante una tabella delle verità:

Tipo		Campo funct						Operazione
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	010
X	1	X	X	X	X	X	X	110
1	X	X	X	0	0	0	0	010
1	X	X	X	0	0	1	0	110
1	X	X	X	0	1	0	0	000
1	X	X	X	0	1	0	1	001
1	X	X	X	1	0	1	0	111

Hardware necessario



Controllo principale



I segnali di controllo

- MemRead
  - Asserito: Lettura dati:= contenuto indirizzo
- MemWrite
  - Asserito: Contenuto indirizzo:= scrittura dati
- RegWrite
  - Asserito: Contenuto registro scrittura:= scrittura dati
- ALUSrc
  - Non asserito: Secondo operando ALU:= lettura dati 2
  - Asserito: Secondo operando ALU:= bit 15-0

- RegDest
  - Non asserito: Registro scrittura:= rt
  - Asserito: Registro scrittura:= rd
- PCSrc
  - Non asserito: PC:=PC+4
  - Asserito: PC:=sommatore
- MemtoReg
  - Non asserito: Scrittura dati registri:=ALU
  - Asserito: Scrittura dati registri:=lettura dati memoria

I valori dei segnali di controllo

	RegDest	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	PCSrc	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

Input/output del controllo

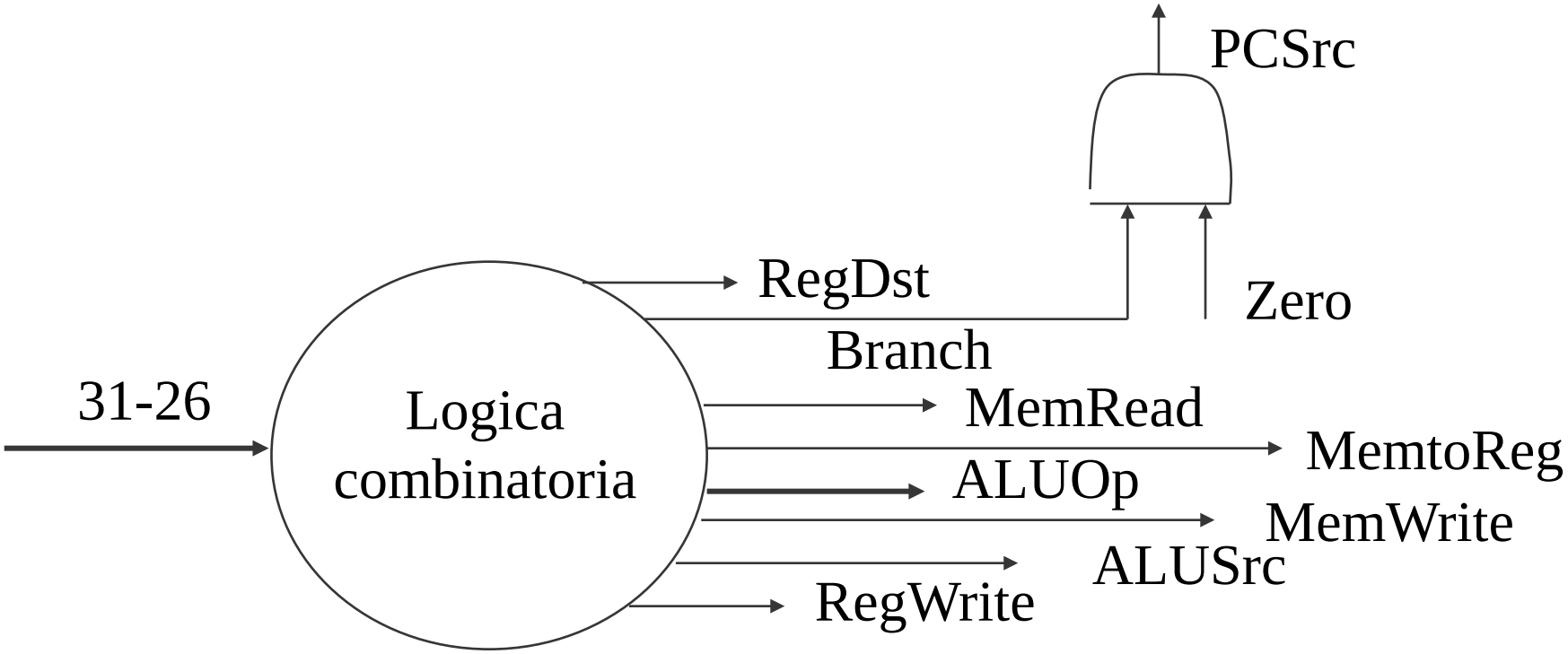
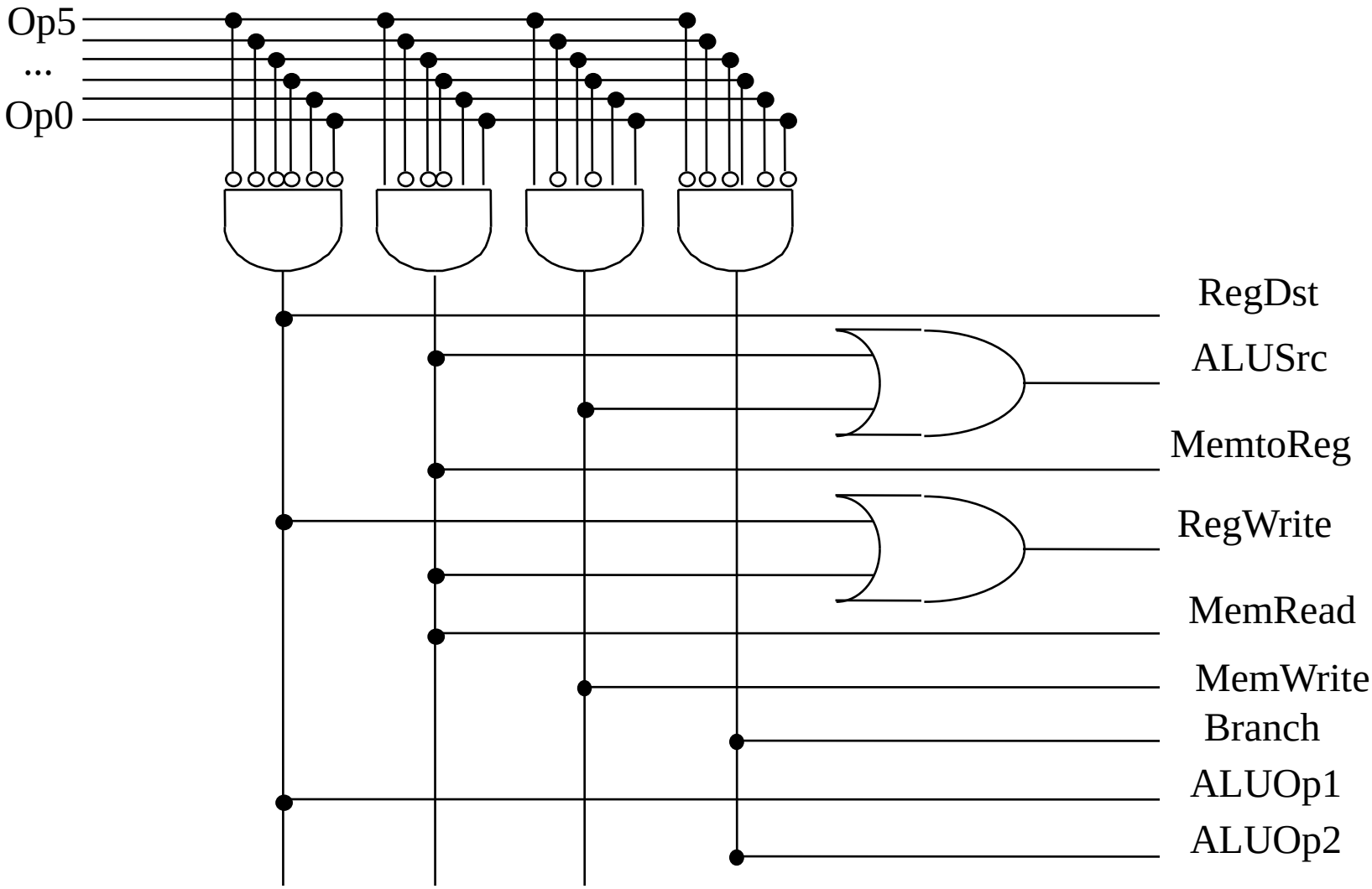


Tabella di verità

		Tipo R	lw	sw	beq
Ingressi	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
Uscite	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

Implementazione hardware



Esercizio

- Come estendere la realizzazione affinchè includa anche l'istruzione di salto incondizionato?

  - concatenazione di:
    - 4 bit più significativi di PC+4
    - 26 bit di indirizzo dell'istruzione
    - due bit a 0

  - mux aggiuntivo per scegliere tra PC+4, indirizzo di salto condizionato ed indirizzo di salto incondizionato
    - nuovo segnale di controllo per gestire mux

Problemi del singolo ciclo

- Durata ciclo di clock pari a percorso più lungo
  - percorso più lungo dovuto ad istruzione di caricamento
- Assumiamo:
  - memoria (2ns), ALU e sommatori (2ns), registri (1ns)
  - nessun ritardo

Istruzione	Memoria istruzioni	Lettura registri	Operazione ALU	Memoria dati	Scrittura registri	Totale
Tipo R	2	1	2	0	1	6ns
lw	2	1	2	2	1	8ns
sw	2	1	2	2		7ns
beq	2	1	2			5ns
j	2					2ns

- Assumiamo:
  - N istruzioni di cui
    - 24% lw, 12% sw, 44% tipo R, 18% beq, 2% j
- Implementazione a ciclo singolo
  - 8N ns
- Se l'implementazione fosse a ciclo variabile:
  - Durata media delle istruzioni:
    - 8 x 0.24 + 7 x 0.12 + 6 x 0.44 + 5 x 0.18 + 2 x 0.2 = 6.3 ns
  - 6.3N ns

Conclusioni

- Problemi del ciclo singolo:
  - prezzo da pagare significativo
    - ancora di più se si considerano istruzioni in virgola mobile
  - spreco di area
    - unità funzionali duplicate
- Ciclo variabile costoso
- Soluzione:
  - usare ciclo più piccolo
  - far si che un'istruzione richieda più cicli
  - un cammino dei dati multi-ciclo