UNIVERSITÀ
DEGLI STUDI
DI TRIESTE

Controllo del datapath e Finite State Machine (FSM)

Prof.ssa Giulia Cisotto

giulia.cisotto@units.it

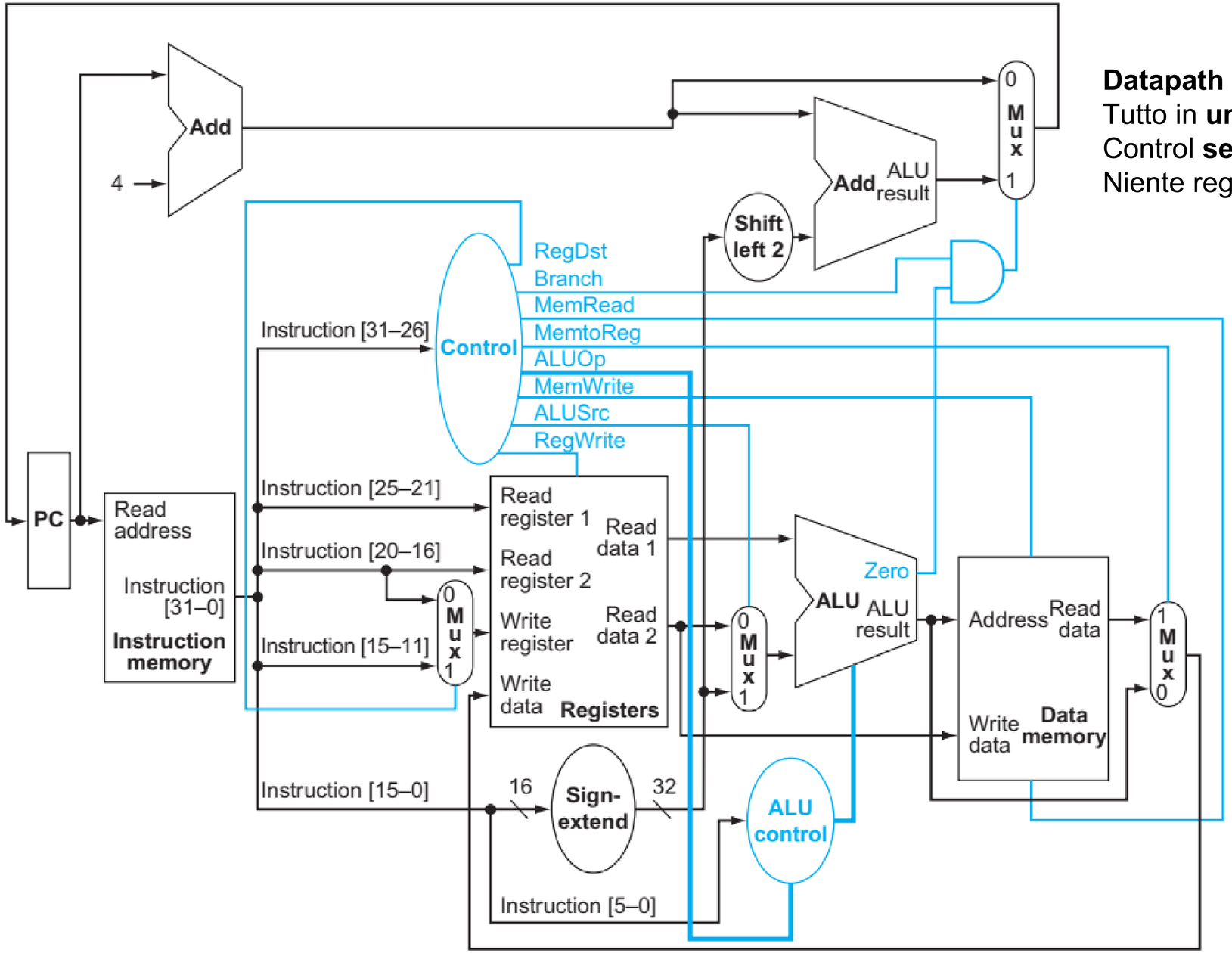
Trieste, 16 aprile 2026

AGENDA DELLA SETTIMANA 14-16/04

- *Datapath*
- *Programmable logical array (PLA)*
- *Controllo del datapth: requisiti*
- Finite State Machine (FSM)
- Control Unit (CU)
- Questionario di metà corso

Jigsaw!

DATAPATH SINGOLO-CICLO *(NO FSM)*

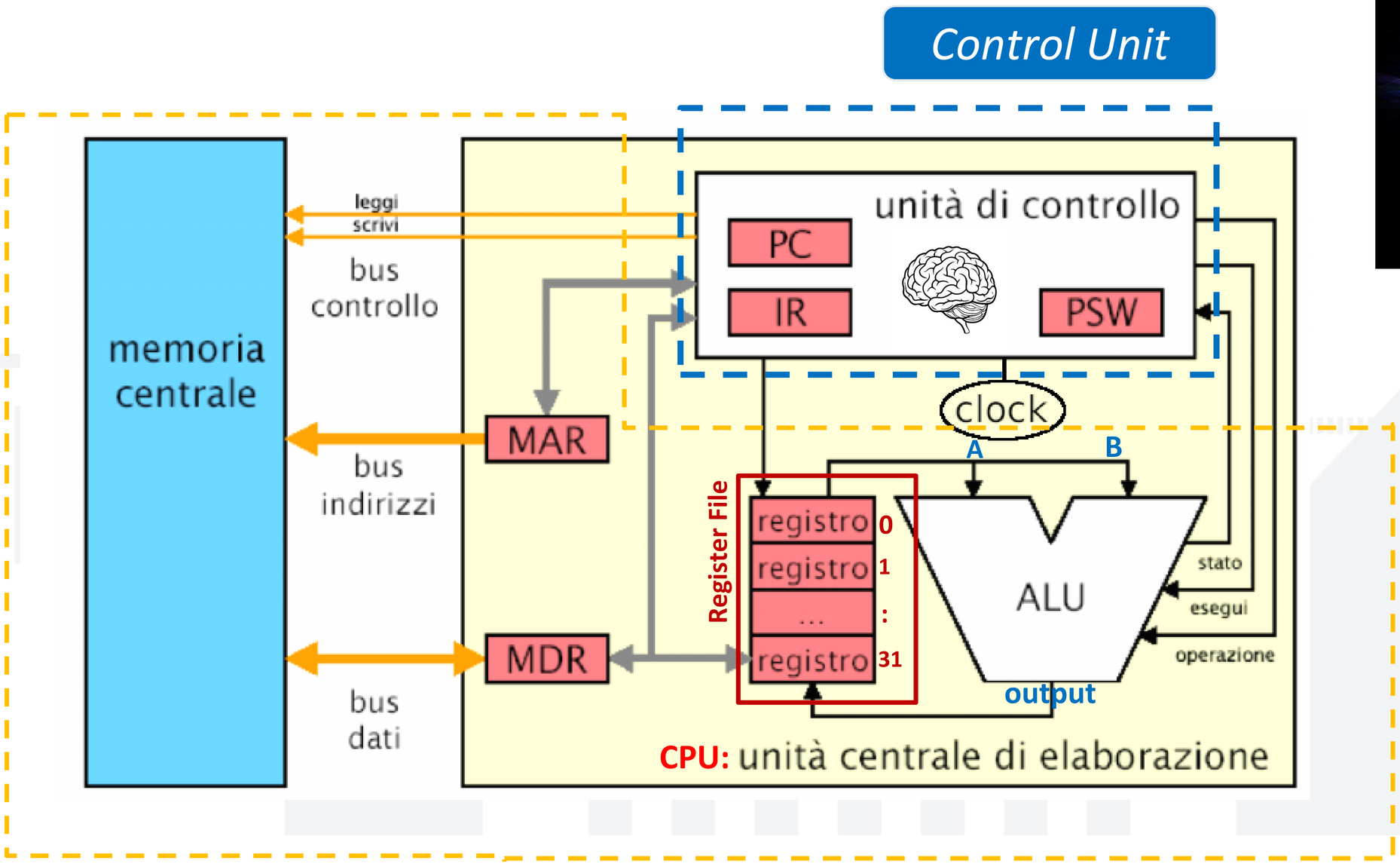


Datapath single-cycle
Tutto in **un solo ciclo di clock**
Control **semplice (combinatoria)**
Niente registri intermedi (oltre a PC e register file)

Cosa manca?

La «dinamica»!

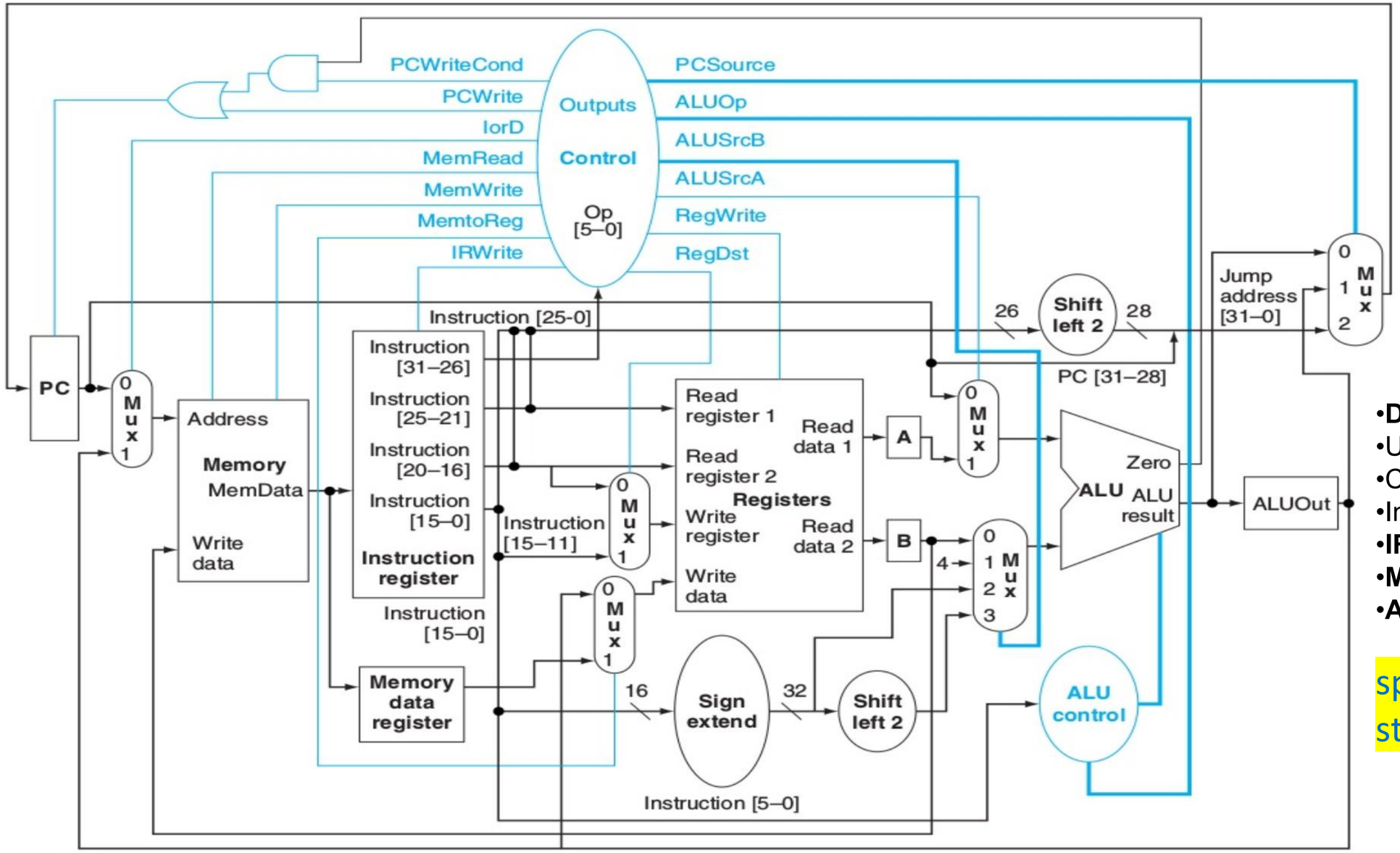
Il controllo è una sequenza nel tempo



CU = cervello
Datapath = corpo

UNIVERSITÀ DEGLI STUDI DI TRIESTE

DATAPATH MULTI-CICLO (CON FSM)



- **Datapath multi-cycle**
- Una istruzione in **più cicli**
- Control = **FSM (con stato)**
- Introduce registri intermedi:
- **IR (instruction register)**
- **MDR (memory data register)**
- **A, B, ALUOut**

spezzato in fasi con stato

Datapath multi-ciclo (Cap5, fig. 5.28)

UNIVERSITÀ DEGLI STUDI DI TRIESTE

DATAPATH MULTI-CICLO (1/2): METODOLOGIA DI CLOCKING

Singolo ciclo

- Ciclo singolo di lunghezza fissa uguale al tempo necessario per eseguire l'istruzione più lunga
- Ogni istruzione viene eseguita in un ciclo di clock
- *Svantaggi*:
  - Istruzioni potenzialmente più veloci sono rallentate (spreco di tempo)
  - Unità funzionali replicate (ad es. memoria, ALU)

Multi-ciclo

- Ciclo di lunghezza fissa più corto
- Ogni istruzione viene eseguita in più cicli di clock
- Istruzioni di tipo diverso vengono eseguite in un numero di cicli di clock diverso
- Le unità funzionali vengono usate più volte durante l'esecuzione della stessa istruzione in cicli di clock differenti -> meno replicazione
- Si usano registri aggiuntivi per memorizzare i risultati parziali nell'esecuzione delle istruzioni

DATAPATH MULTI-CICLO (2/2): DIFFERENZE ARCHITETTURALI

Registri aggiuntivi:

Memorizzano valori intermedi che vengono usati nel ciclo di clock successivo per continuare l'esecuzione della stessa istruzione:

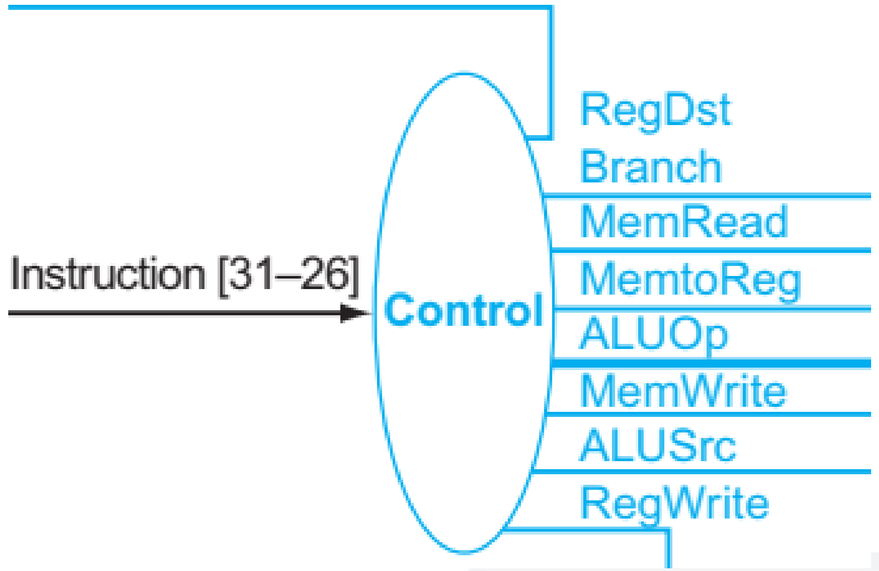
- IR - Instruction Register
- MDR - Memory Data Register
- A, B - Registri tra Register File e l'ingresso ALU
- ALUout - L’output della ALU

Riutilizzo di unità funzionali:

- ALU usata non solo per le operazioni aritmetico-logiche ma anche per calcolare l’indirizzo dei salti e per incrementare il PC
- Memoria usata sia per leggere le istruzioni che per leggere/scrivere i dati

AGENDA DI OGGI

- *Datapath*
- *Programmable logical array (PLA)*
- **Controllo del datapth: requisiti**
- *Finite State Machine (FSM)*
- *Control Unit (CU)*
- *Questionario di metà corso*



CONTROL UNIT E ALU CONTROL

Control Unit e ALU Control sono implementate come PLA per generare i segnali di controllo alle diverse parti dell'architettura.

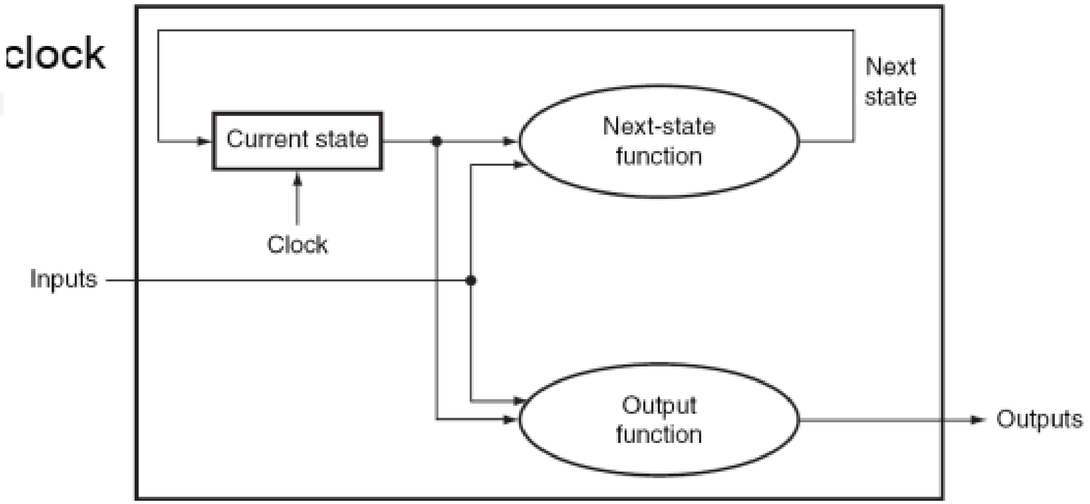
CONTROL UNIT

La Control Unit però non sfrutta solo il concetto di **PLA** ma anche di macchina a stati finiti (o **Finite State Machine, FSM**)

Prossimo stato

Input: opcode, funct,
stato corrente

output: risultato
dell'istruzione corrente



Lo **STATO** è un numero che va da 0 (fetch) a X (massimo numero di stati realizzabili dall'architettura) ed è definito dal set di segnali di controllo impostati.

AGENDA DI OGGI

- *Datapath*
- *Programmable logical array (PLA)*
- *Controllo del datapth: requisiti*
- **Finite State Machine (FSM)**
- *Control Unit (CU)*
- *Questionario di metà corso*

MACCHINA A STATI FINITI/FINITE STATE MACHINE (FSM)

Finite State Machine (FSM): usate per descrivere i circuiti sequenziali

Composte da un set di stati e 2 funzioni:

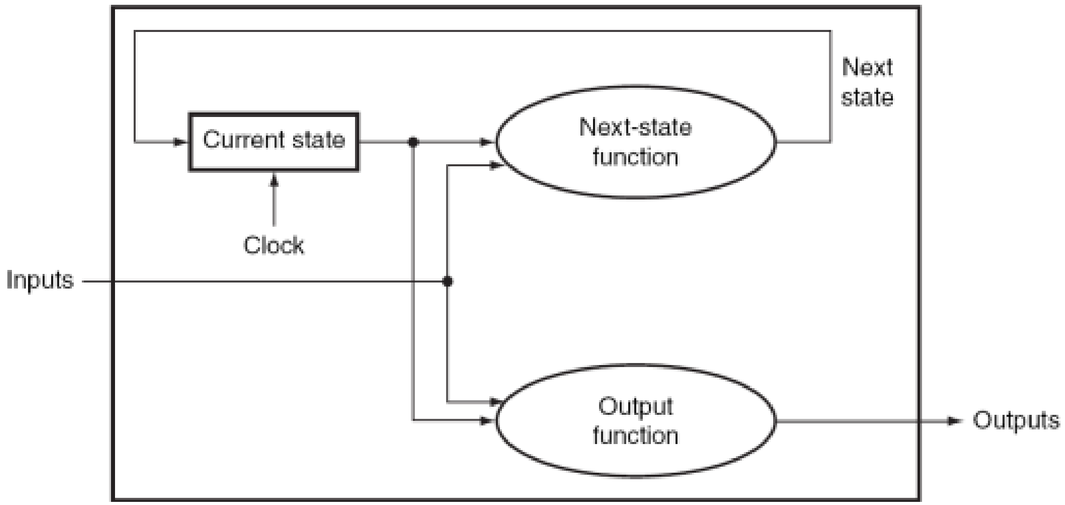
- Next state function

Determina lo stato successivo partendo dallo stato corrente e dai valori in ingresso

- Output function

Produce un insieme di risultati partendo dallo stato corrente e dai valori in ingresso

Sono sincronizzate con il clock



FSM: MOORE VS MEALY

Next state dipende sia dagli input che dallo stato corrente

Output:

- Se dipende solo dallo stato: ***Moore*** – usato come controller
- Se dipende dallo stato corrente e dagli input: ***Mealy***
- Moore and Mealy sono equivalenti e si possono trasformare automaticamente tra di loro
- *In questo corso usiamo FSM Moore*

DIAGRAMMA DEGLI STATI (ALIAS: MACCHINA A STATI FINITI)

Visione di alto livello

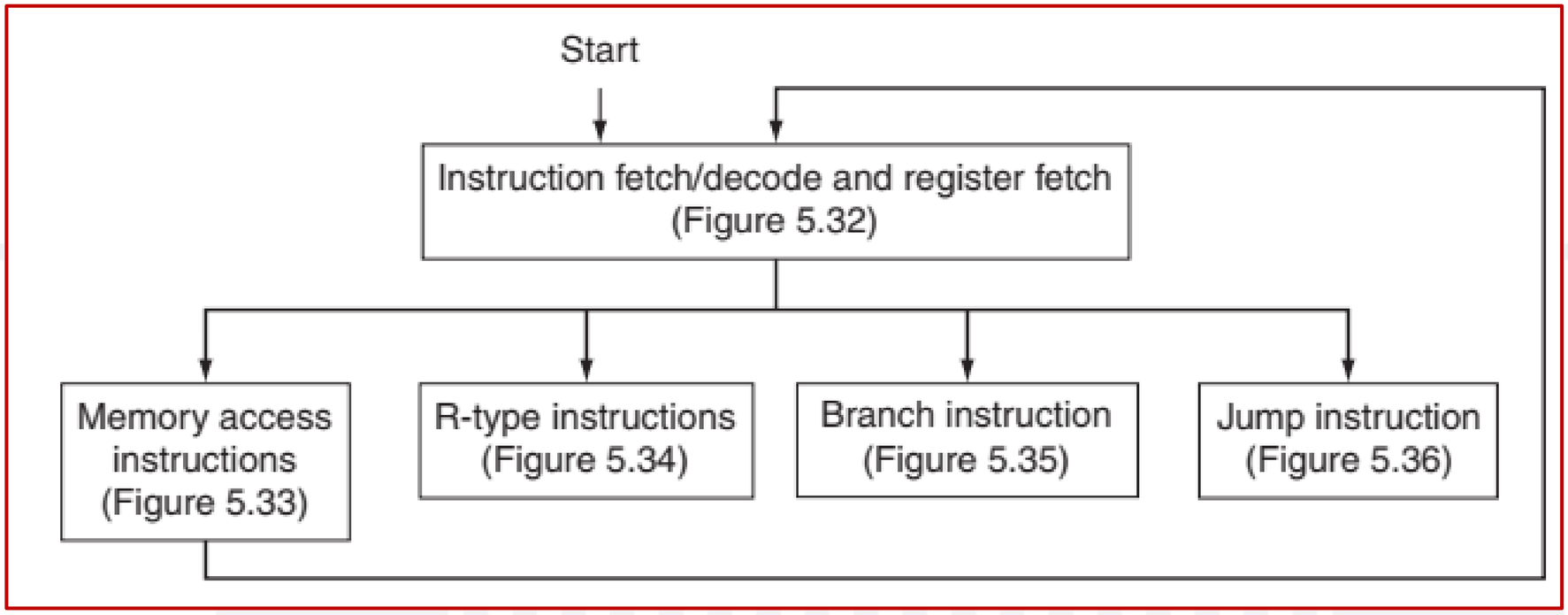
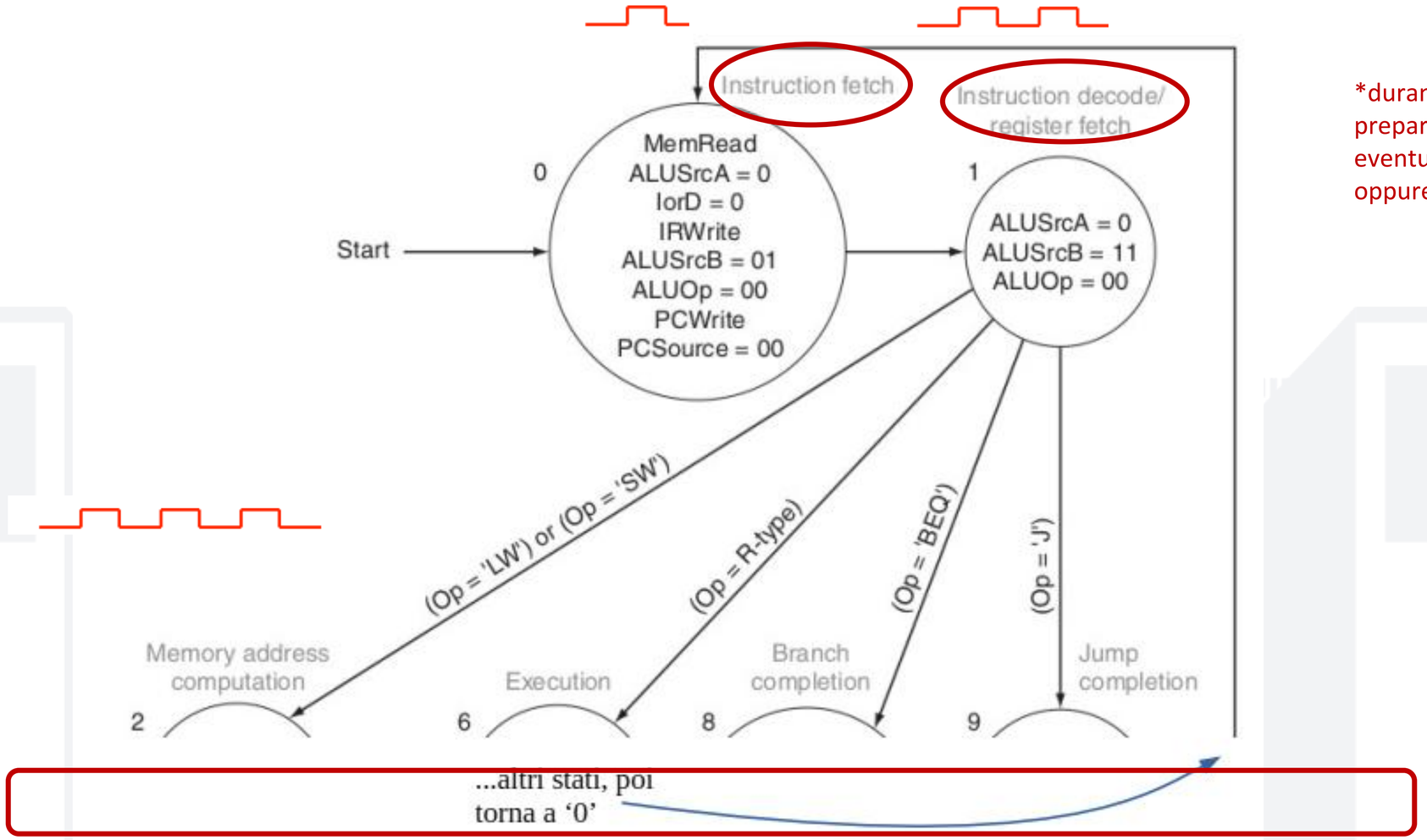


FIGURE 5.38 The complete finite state machine control for the datapath shown in Figure 5.28. The labels on the arcs are conditions that are tested to determine which state is the next state; when the next state is unconditional, no label is given. The labels inside the nodes indicate the output signals asserted during that state; we always specify the setting of a multiplexor control signal if the correct operation requires it. Hence, in some states a multiplexor control will be set to 0.

Cap5 (p.331,p.339)

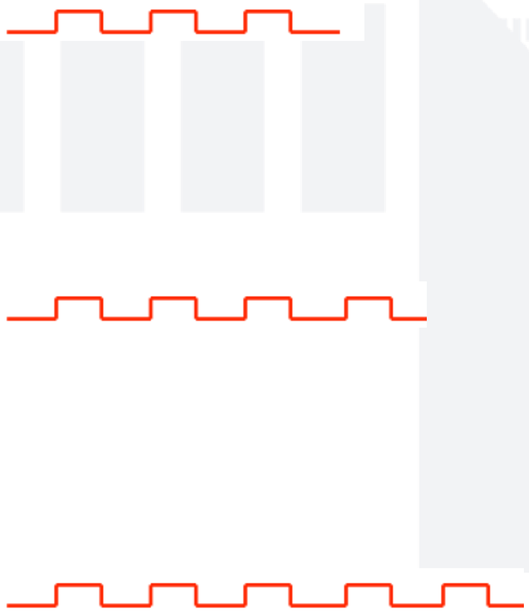
FSM PER DATAPATH: FETCH + DECODE + …

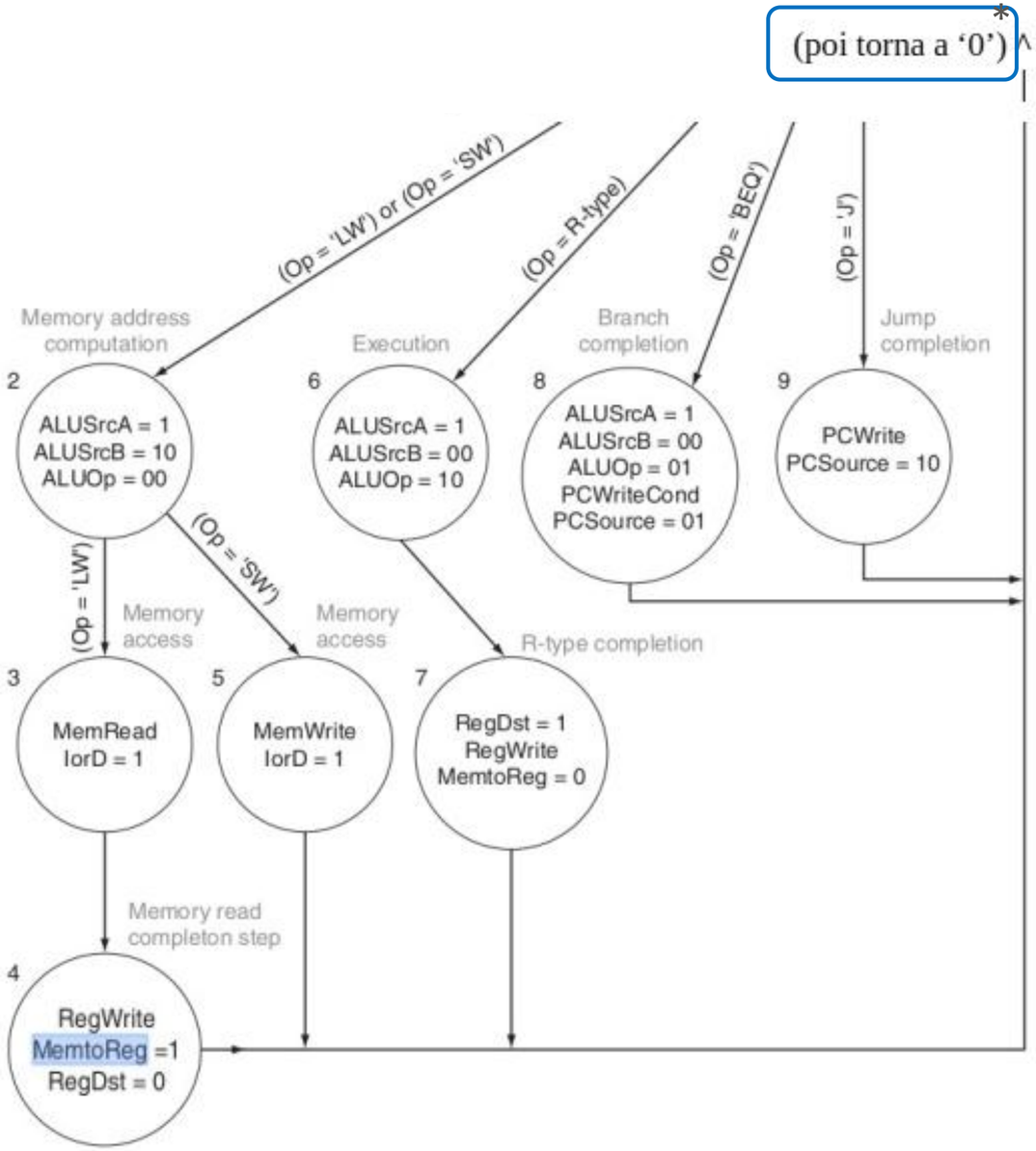


*durante la fase di decode si preparano A, B, branch (per una eventuale istruzione R-type oppure branch condizionato)

UNIVERSITÀ DEGLI STUDI DI TRIESTE

FSM PER DATAPATH: .. + EXECUTE



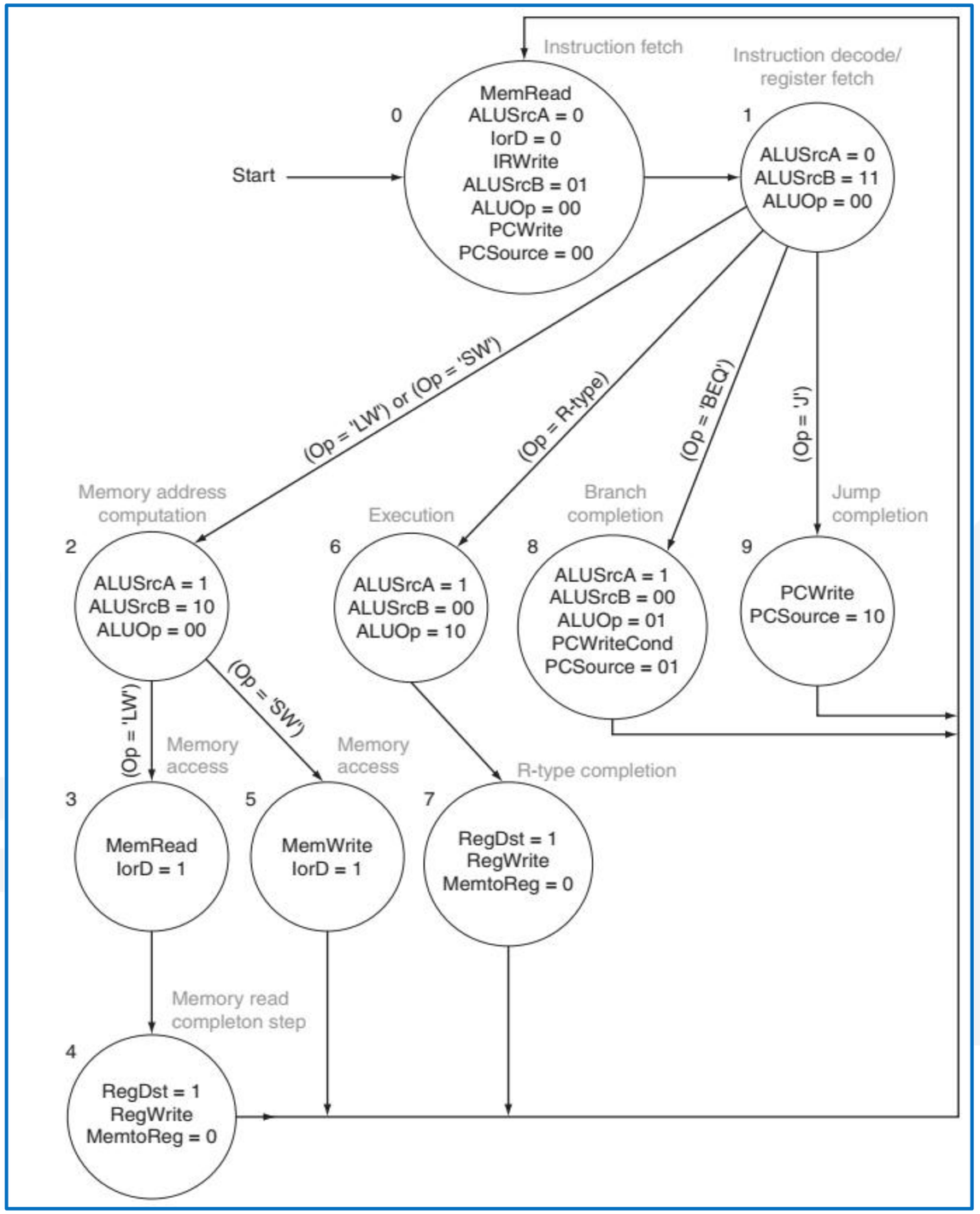


*per poter eseguire la prossima istruzione

FSM PER IL DATAPATH MULTI-CICLO

In **un singolo ciclo di clock**:

- Si possono fare più operazioni in parallelo, ma una per ogni risorsa hardware.
- più blocchi possono lavorare insieme (ALU + memoria + register file, ecc.)



AGENDA DI OGGI

- *Datapath*
- *Programmable logical array (PLA)*
- *Controllo del datapth: requisiti*
- *Finite State Machine (FSM)*
- **Control Unit (CU)**
- *Questionario di metà corso*

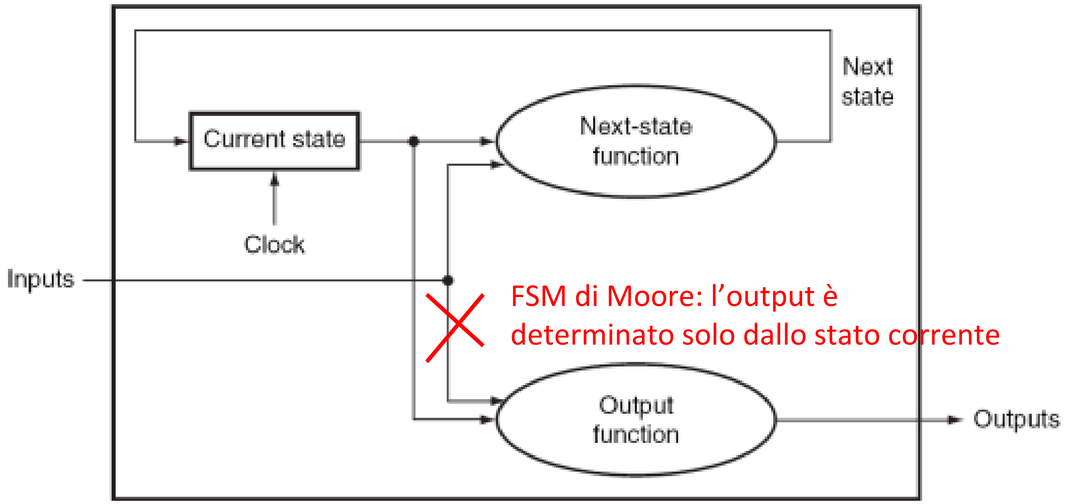
CONTROL UNIT

La Control Unit però non sfrutta solo il concetto di **PLA** ma anche di macchina a stati finiti (o **Finite State Machine, FSM**)

Prossimo stato

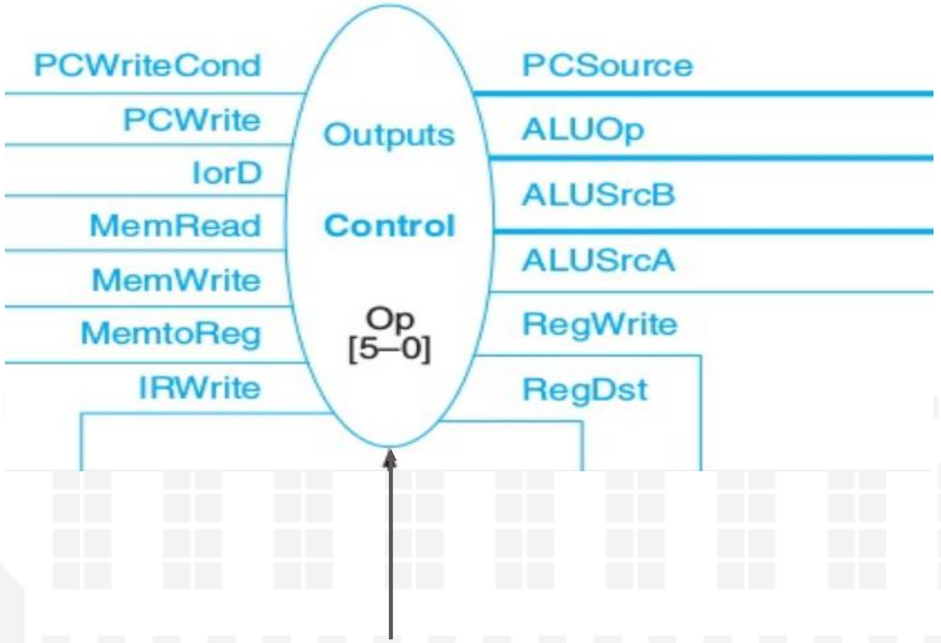
Input: opcode, funct,
stato corrente

output: risultato
dell'istruzione corrente



Lo **STATO** è un numero che va da 0 (fetch) a X (massimo numero di stati realizzabili dall'architettura) ed è definito dal set di segnali di controllo impostati.

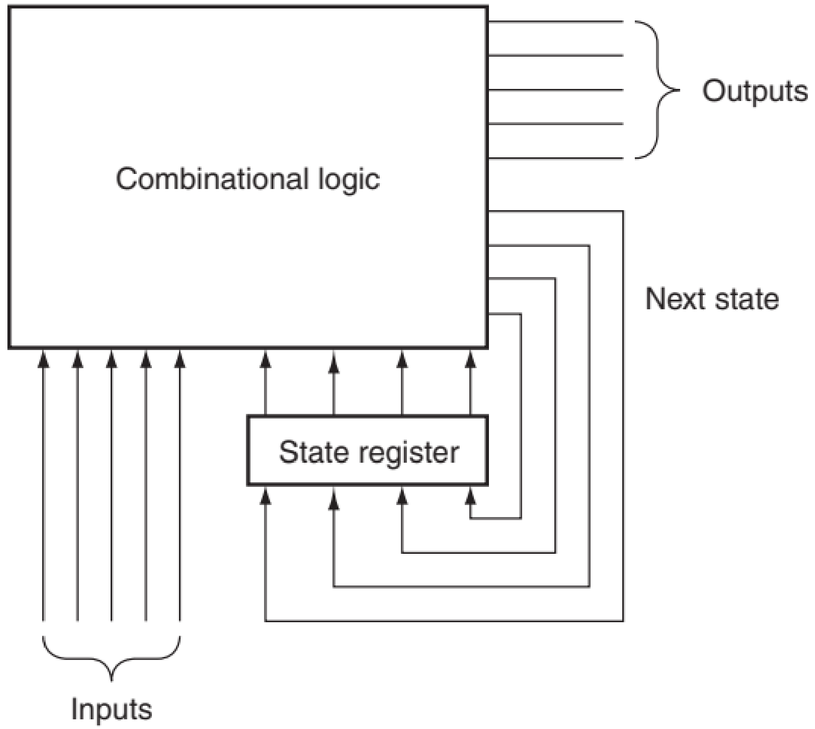
LOGICA PER IL CONTROLLO



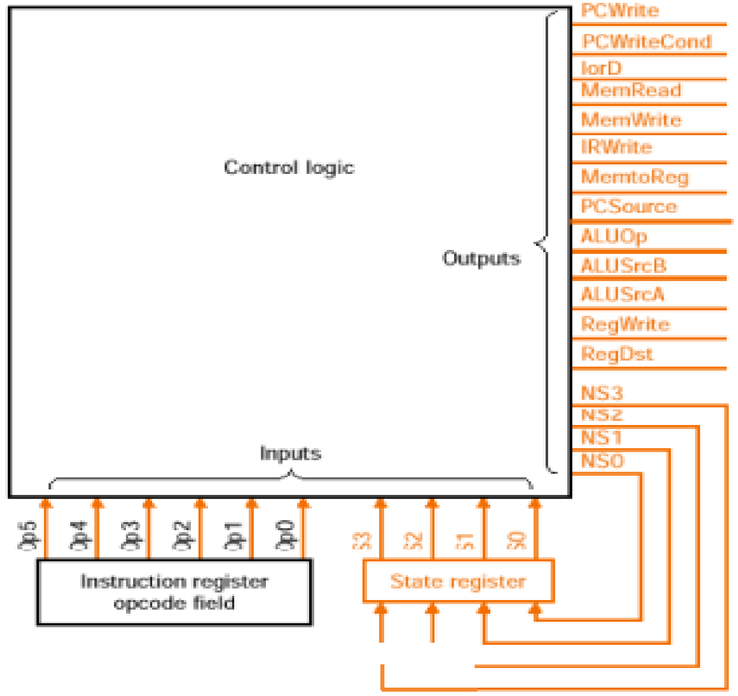
Instruction[31-26]

- **Controllo a due livelli: OPCODE** per determinare il tipo di istruzione e anche **FUNC CODE** per R-type
- **State register** memorizza lo stato corrente
- **Blocco combinatorio (PLA)** per il calcolo di NEXT_STATE e OUTPUT (memorizzate in ROM)

LOGICA (COMB. + SEQ.) PER IL CONTROLLO



Nel dettaglio:



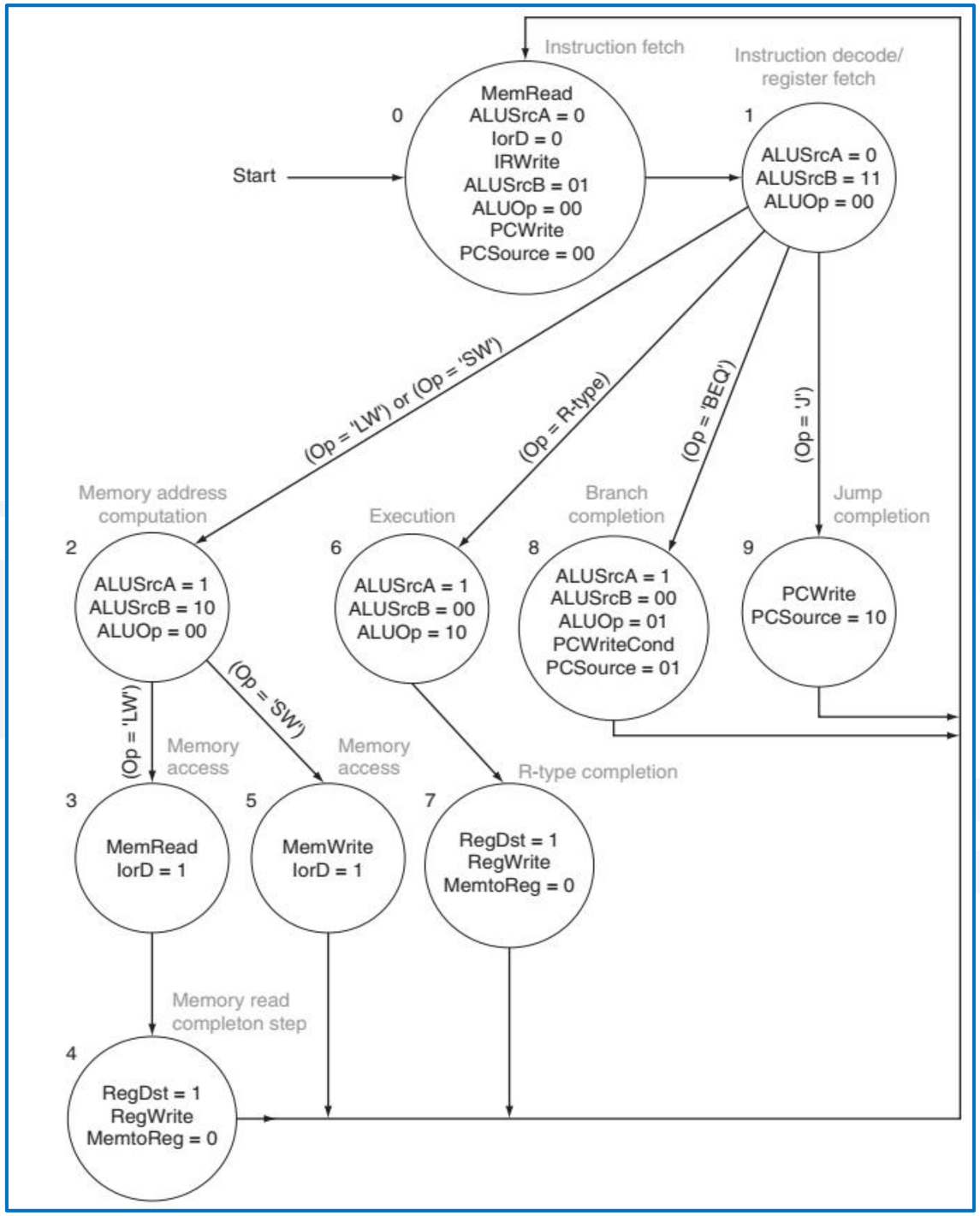
Next state

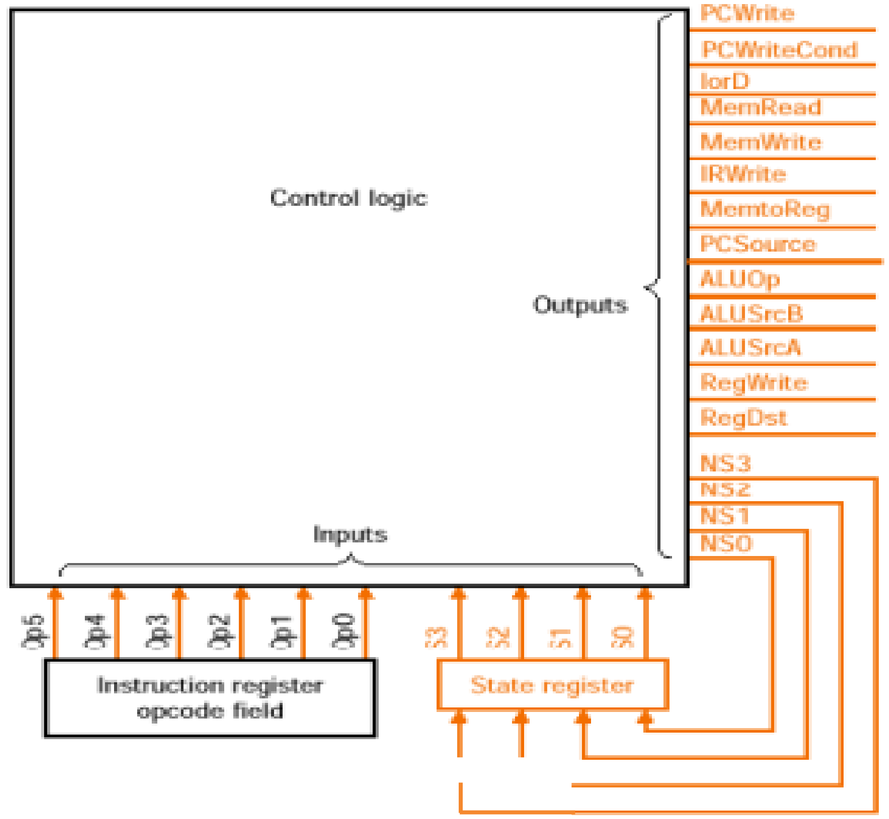
Current state

- **Controllo a due livelli: OPCODE** per determinare il tipo di istruzione e anche **FUNC CODE** per R-type
- **State register** memorizza lo stato corrente
- **Blocco combinatorio (PLA)** per il calcolo di NEXT_STATE e OUTPUT (memorizzate in ROM)

QUANTI STATI?

10 possibili stati: servono **4 bit** per rappresentare ciascuno di loro (numeri da 0 a 9)





UNIVERSITÀ DEGLI STUDI DI TRIESTE

ESEMPIO FINALE (DATAPATH + CONTROLLO + FSM)

R-type

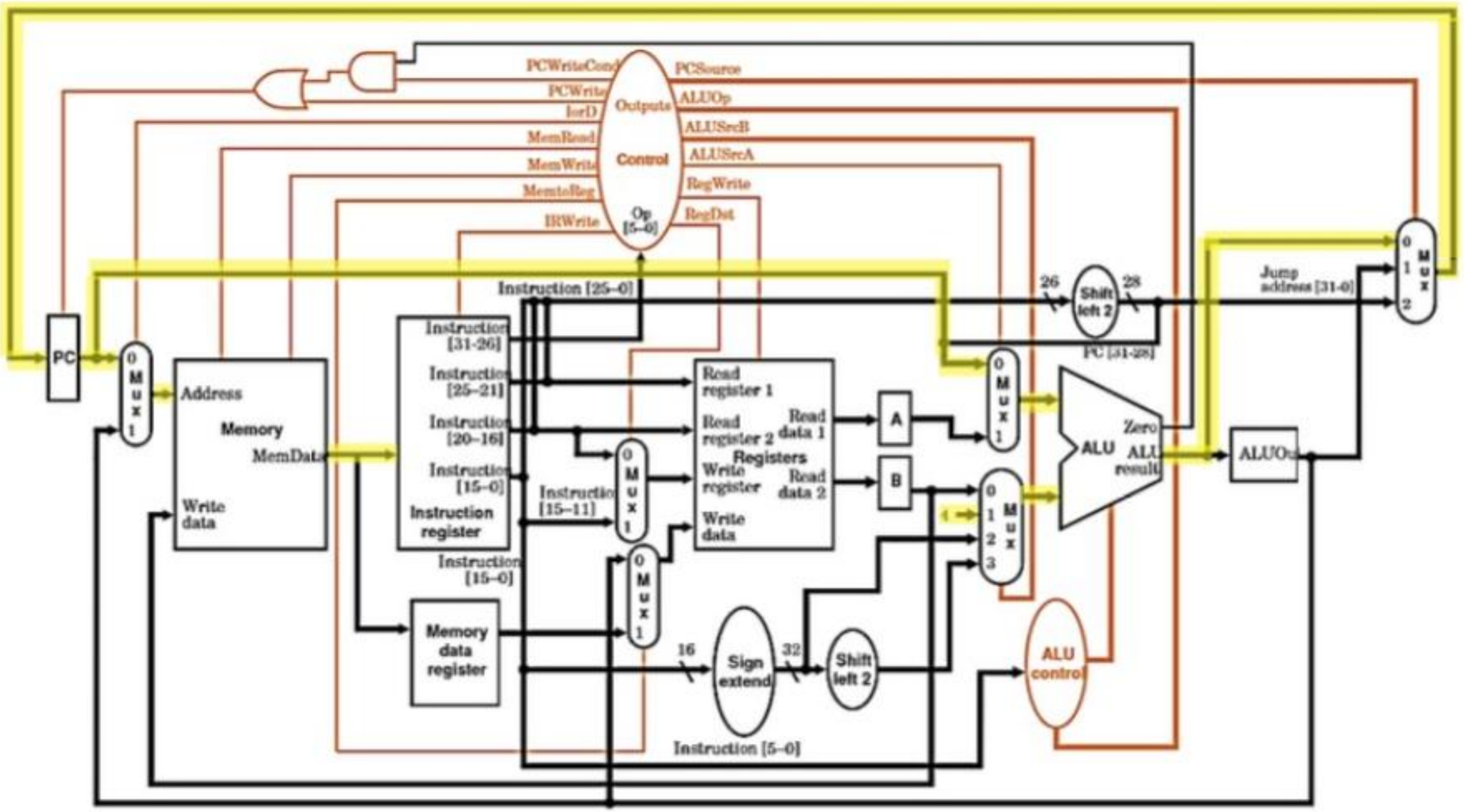
```
add $s0, $s1, $2
```

Questa istruzione richiede 4 cicli di clock per essere realizzata: fetch (1 ciclo), decode (1 ciclo), execute (2 cicli)

Fetch

- dal PC legge l'istruzione in memoria
- aggiorna il PC → PC + 4

Signal	Value
PCWrite	1
IorD	0
MemRead	1
MemWrite	0
IRWrite	1
PCSource	00
ALUOp	00
ALUSrcB	01
ALUSrcA	0
RegWrite	0



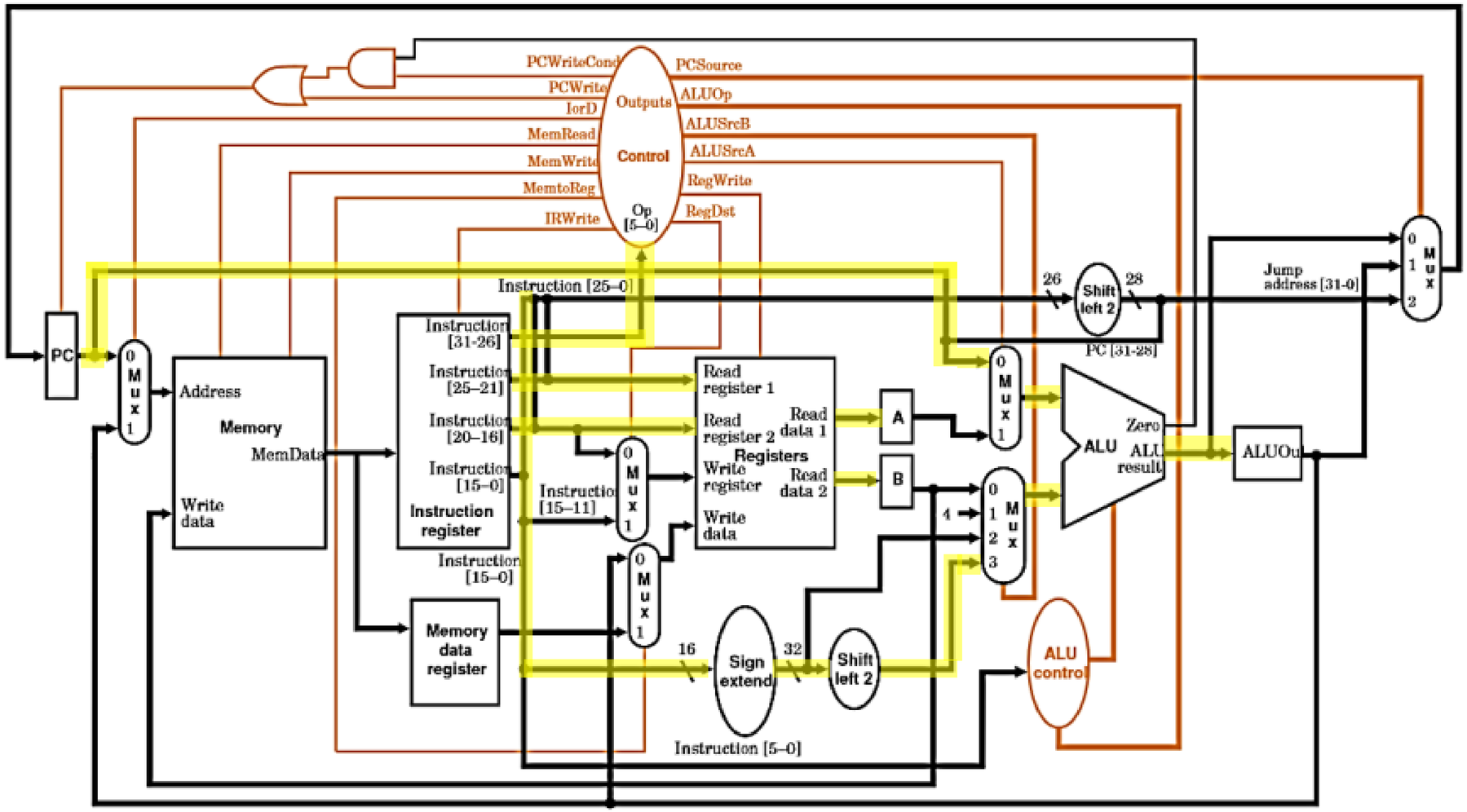
Decode

- legge i registri
- prepara eventuali calcoli futuri (branch target)

Register file

```
A ← R[rs] = R[$s1]
B ← R[rt] = R[$s2]
```

Signal	Value
ALUOp	00
ALUSrcB	11
ALUSrcA	0



N.B. anche se in questo caso (istruzione R-type) non sarà necessario, è comunque calcolato il valore di branch che viene utilizzato per l'aggiornamento del PC nel caso di istruzioni di salto condizionato

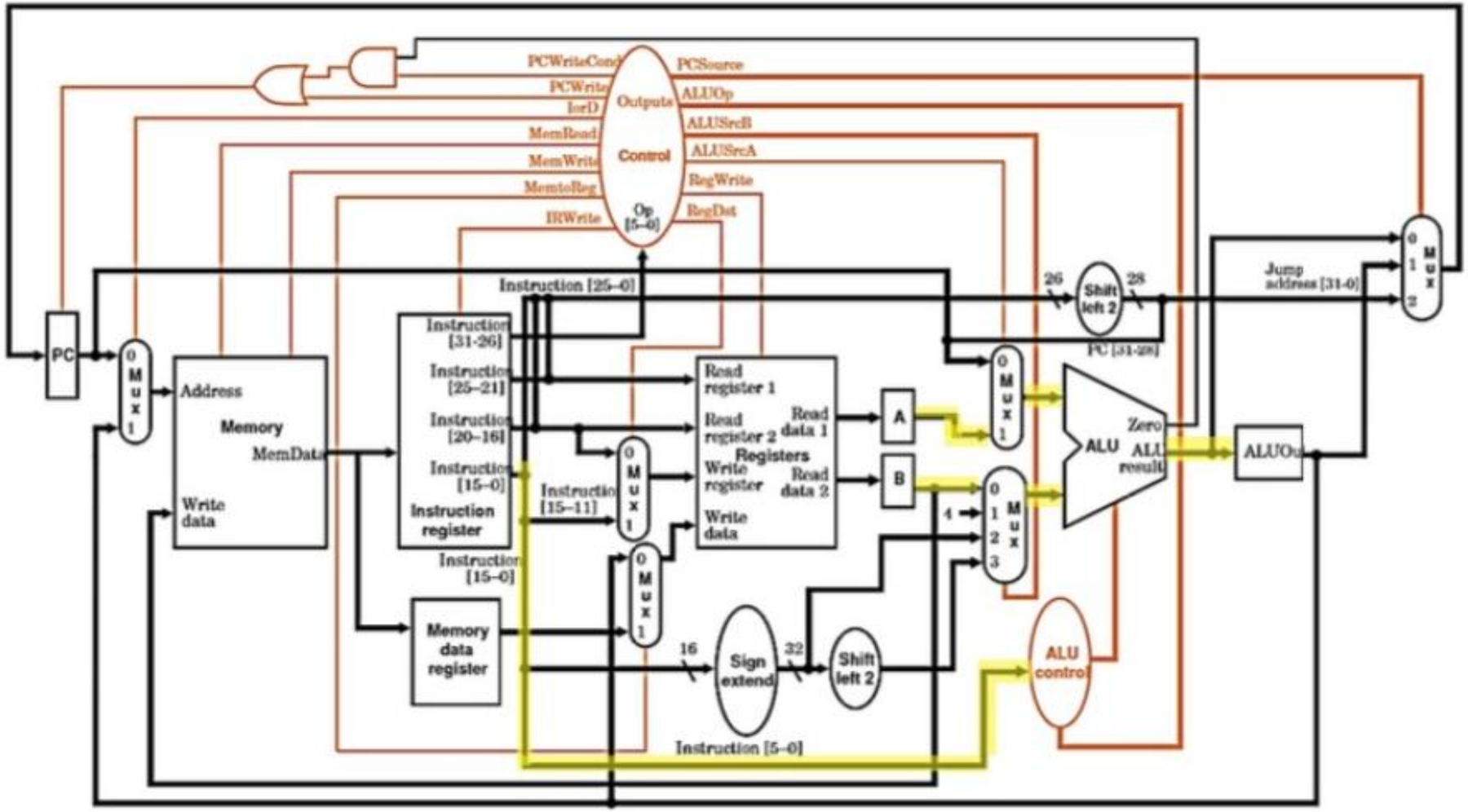
ingresso B = **offset 16 bit esteso << 2**

è sub → per verificare la condizione di branch (beq)

Execute (1)

`ALUSrcA = 1` → ingresso A = registro A (cioè `$s1`)
`ALUSrcB = 00` → ingresso B = registro B (cioè `$s2`)
`ALUOp = 10` → operazione decisa dal funct field (→ add)

Signal	Value
ALUOp	10
ALUSrcB	00
ALUSrcA	1



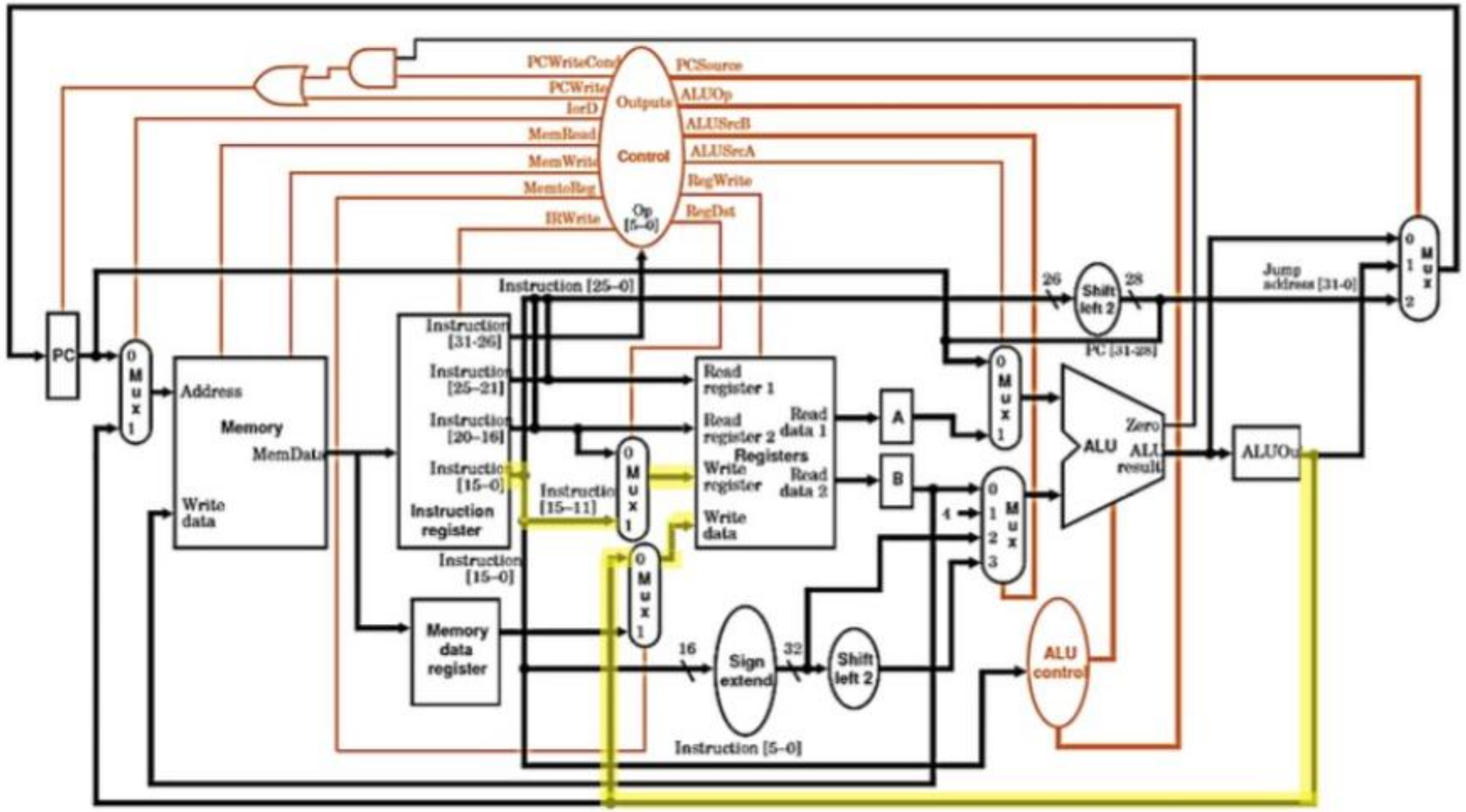
Ingressi ALU
A = R[$s1]
B = R[$s2]

ALUOut ← R[$s1] + R[$s2]

Execute (2)

- si scrive il risultato nel registro di destinazione

Signal	Value
MemtoReg	0
RegWrite	1
RegDst	1



AGENDA DELLA SETTIMANA 14-16/04

- *Datapath*
- *Programmable logical array (PLA)*
- *Controllo del datapth: requisiti*
- *Finite State Machine (FSM)*
- *Control Unit (CU)*
- **Questionario di metà corso**

QUESTIONARIO DI METÀ CORSO

SCAN ME

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Materiale per la lezione

- Patterson & Hennessy, capitolo 4
- Patterson & Hennessy, Appendice B
- Capitolo 5 (Datapath)

*Prossima lezione: 21 aprile, **h.14, aula 4B***