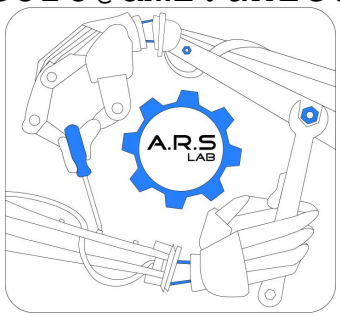
Inside the CPU

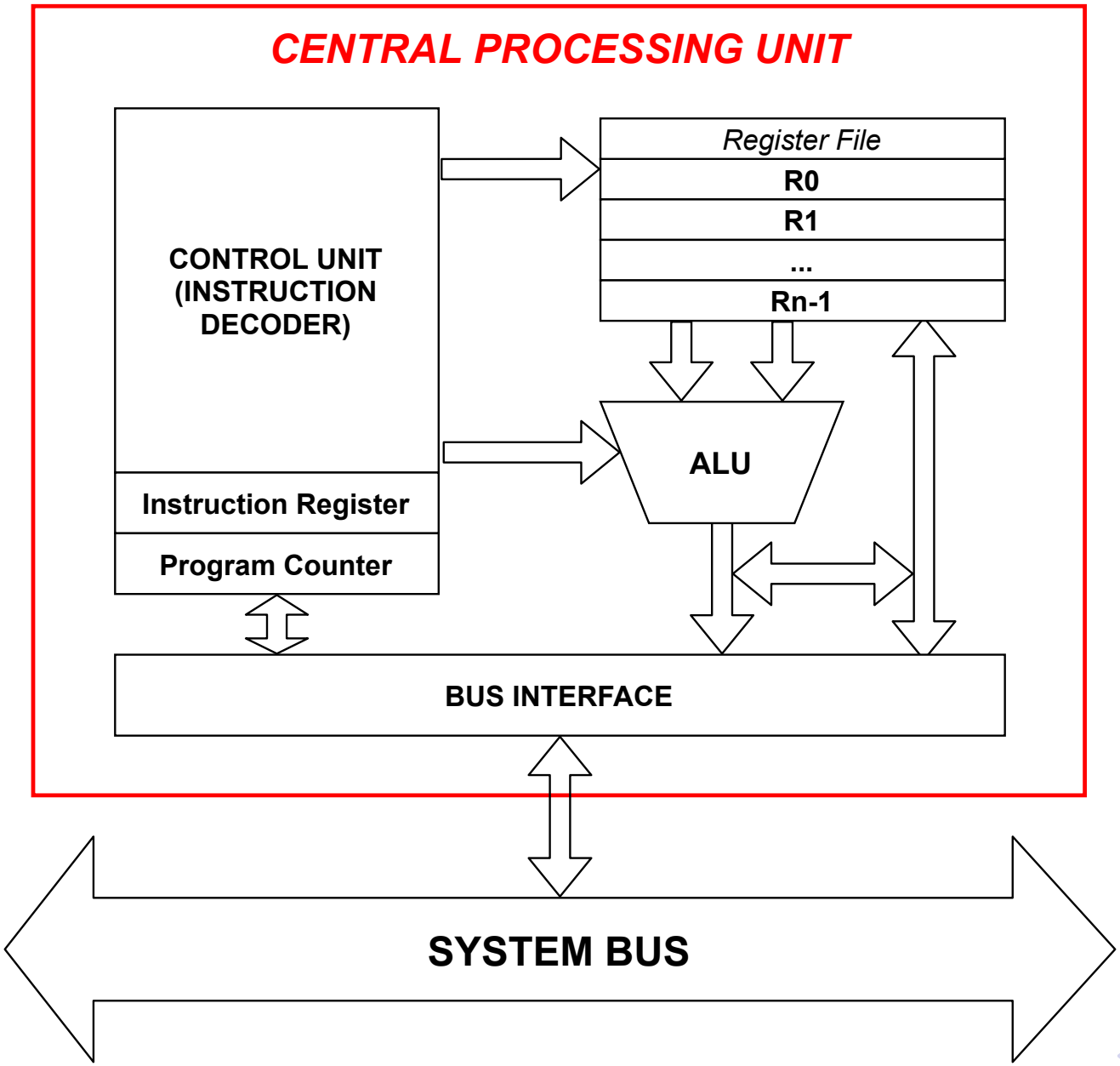
Corrado Santoro

Dipartimento di Matematica e Informatica
santoro@dmi.unict.it



Corso di Architettura degli Elaboratori

Architettura CPU



Classificazione Architetture CPU

RISC: Reduced Instruction Set Computer

- Set di istruzioni semplificato
- Istruzioni aritmetiche/logiche solo tra registri
- Supporto di un insieme ridotto di modalità di indirizzamento
- Ogni opcode (istruzione) occupa una word
- Esecuzione “a stadi”, con possiblità di “pipelining”

CISC: Complex Instruction Set Computer

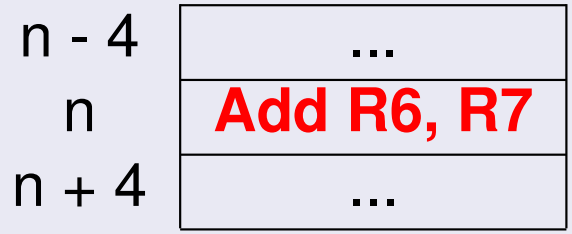
- Set di istruzioni ricco
- Istruzioni aritmetiche/logiche tra registri e tra registri e memoria
- Supporto di svariate modalità di indirizzamento
- Ogni opcode (istruzione) può occupare una o più word
- Programmi più “compatti” ma meno performanti

Esempi di Istruzione CISC

Istruzione CISC a 1 word

Add R6, [R7]

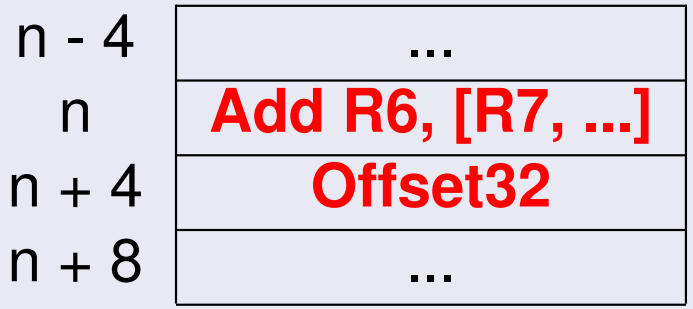
$R6 \leftarrow [R6] + [[R7]]$



Istruzione CISC a 2 word

Add R6, [R7, #Offset32]

$R6 \leftarrow [R6] + [[R7 + Offset32]]$

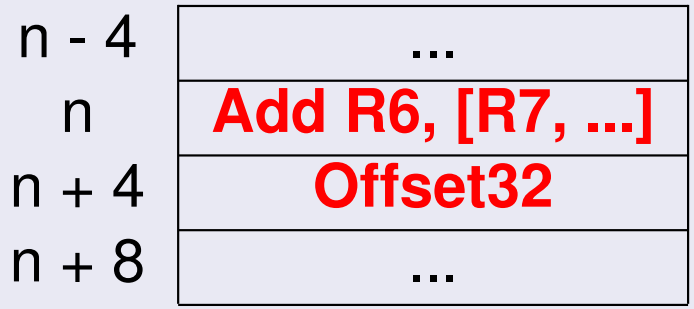


Esempi di Istruzione CISC

Istruzione CISC a 2 word

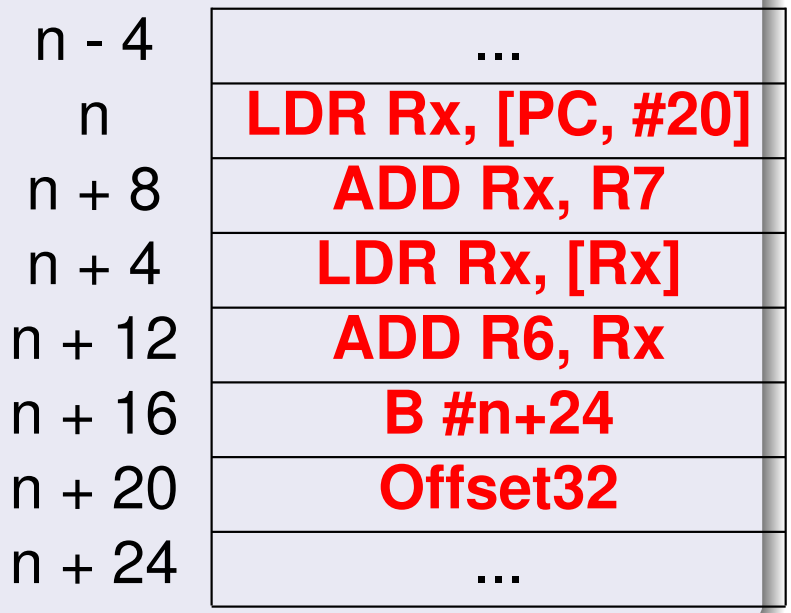
Add R6, [R7, #Offset32]

$R6 \leftarrow [R6] + [[R7 + Offset32]]$



Equivalente RISC

$Rx \leftarrow Offset32$
$Rx \leftarrow [Rx] + [R7]$
$Rx \leftarrow [[Rx]]$
$R6 \leftarrow [R6] + [Rx]$



Ciclo di lavoro CPU

Fasi della CPU

- **Fetch Phase**: trasferimento su IR della word contenuta nella locazione puntata dal PC:

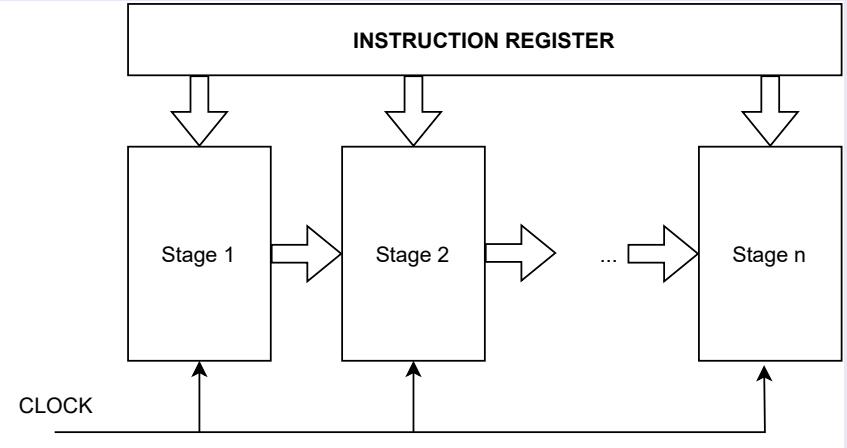
$$IR \leftarrow [[PC]]$$

- incremento del PC in modo da puntare alla word successiva:

$$PC \leftarrow [PC] + 4$$

- **Execution Phase**: decodifica ed esecuzione dell'istruzione presente su IR

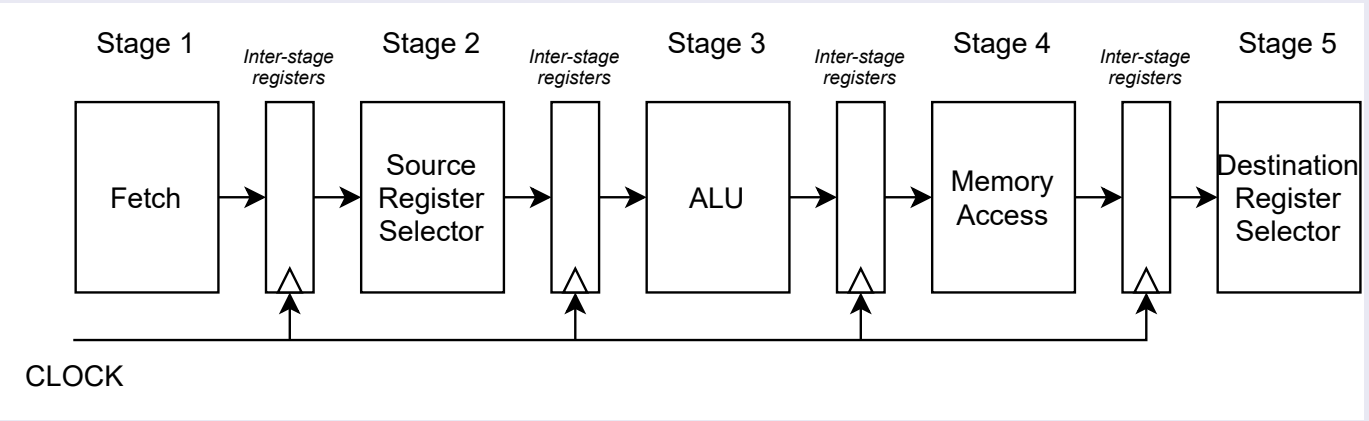
Stage-based Architecture



- L’unità di elaborazione è composta da diversi **stadi (stages)** posti in sequenza
- Ogni stadio è specializzato in un compito preciso (op aritmetiche/logiche, trasferimenti di memoria, etc.)
- L’operazione specifica eseguita dal singolo stadio è selezionata da opportuni bit dell’istruzione
- I dati trattati nell’istruzione pertanto “scorrono” attraverso i vari stage
- L’insieme di stage viene denominato **Data Path**

RISC Processors

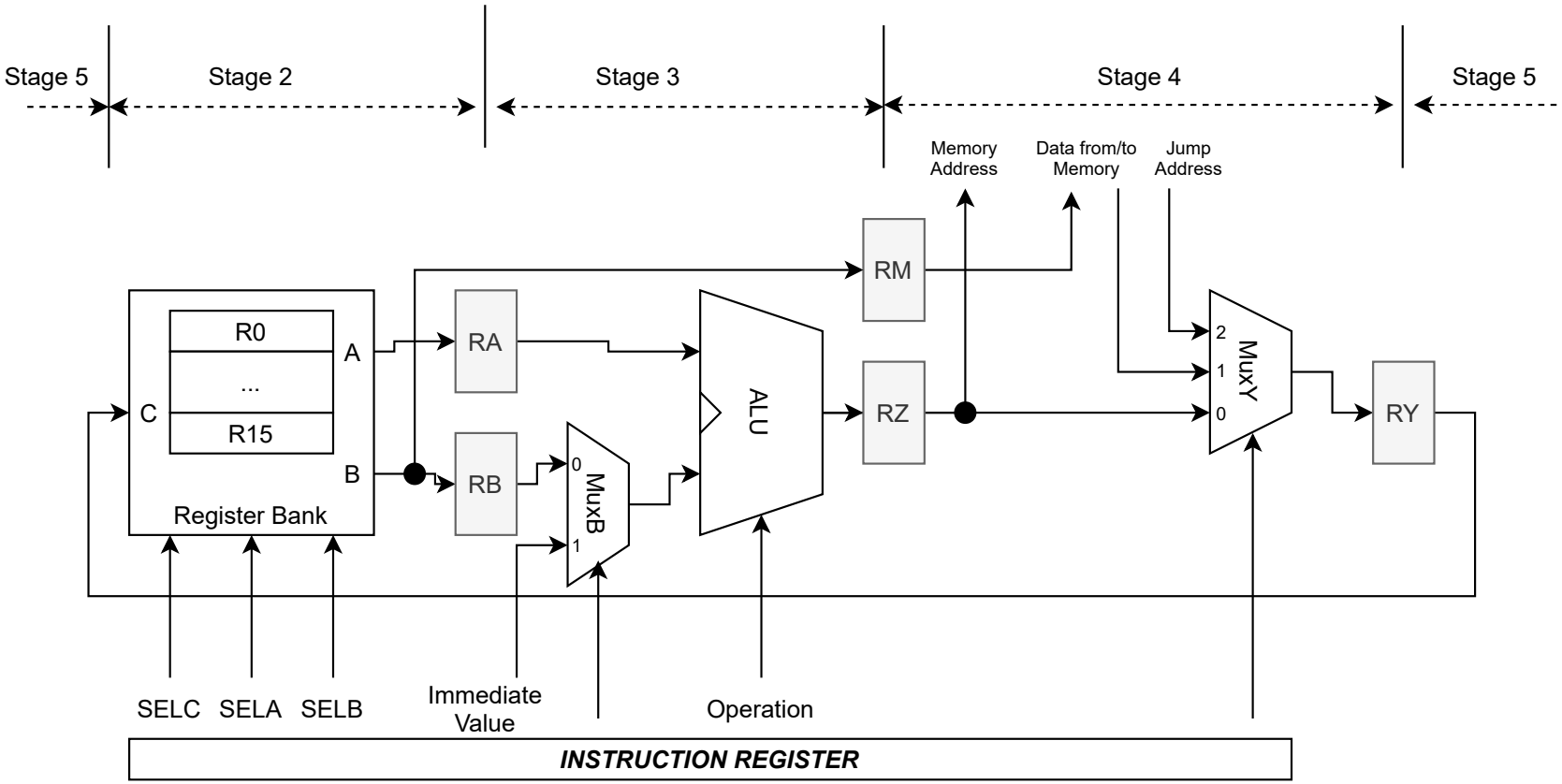
5-Stage Architecture: RISC Processors



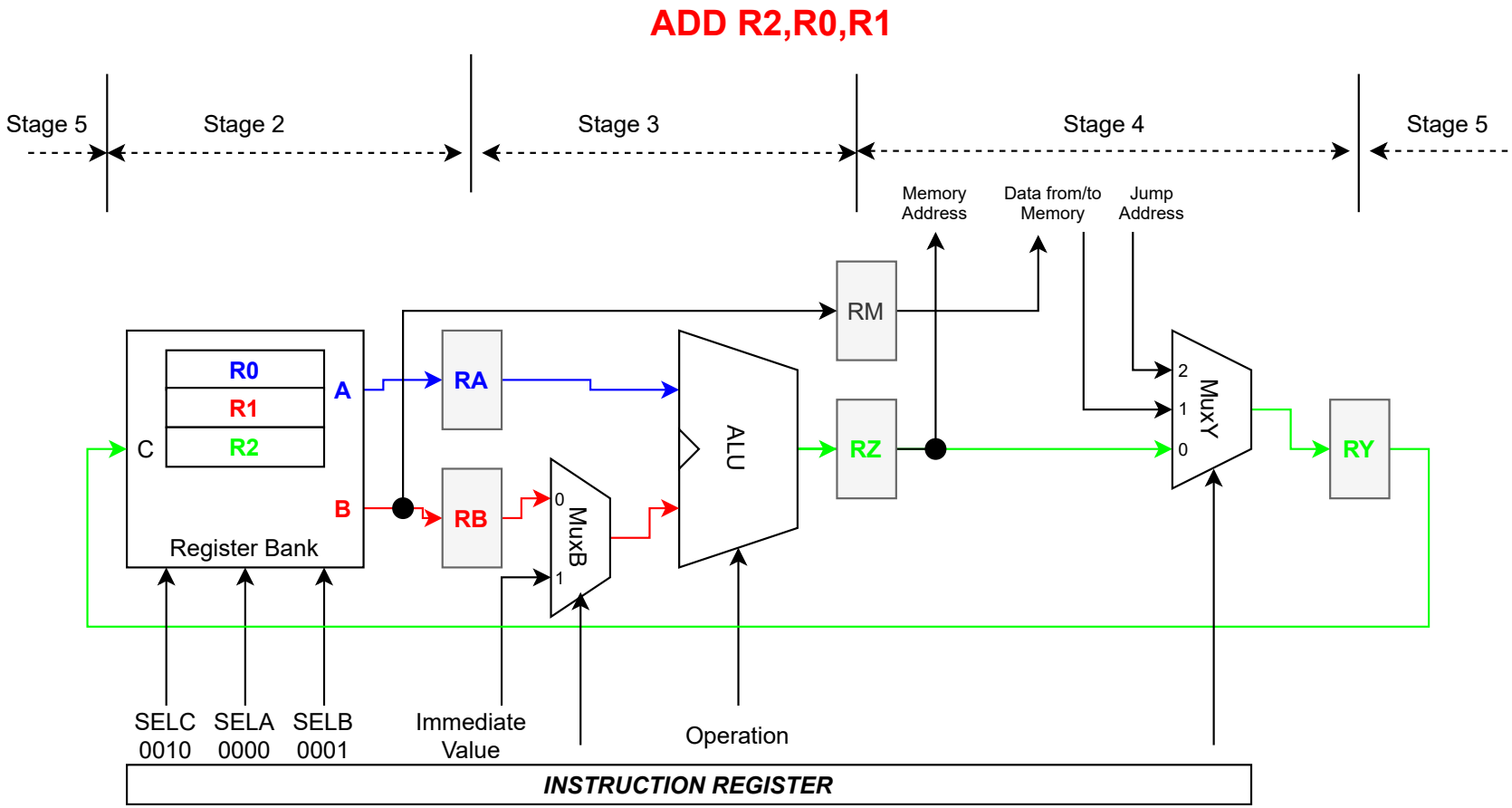
1. Prelievo dell'istruzione
2. Selezione dei registri "sorgente" per l'istruzione
3. Esecuzione dell'eventuale operazione artimetico/logica
4. Eventuale accesso alla memoria (load o store)
5. Memorizzazione su registro destinazione

Inter-stage registers: registri (interni) dove vengono memorizzati i dati parziali elaborati da ogni stadio

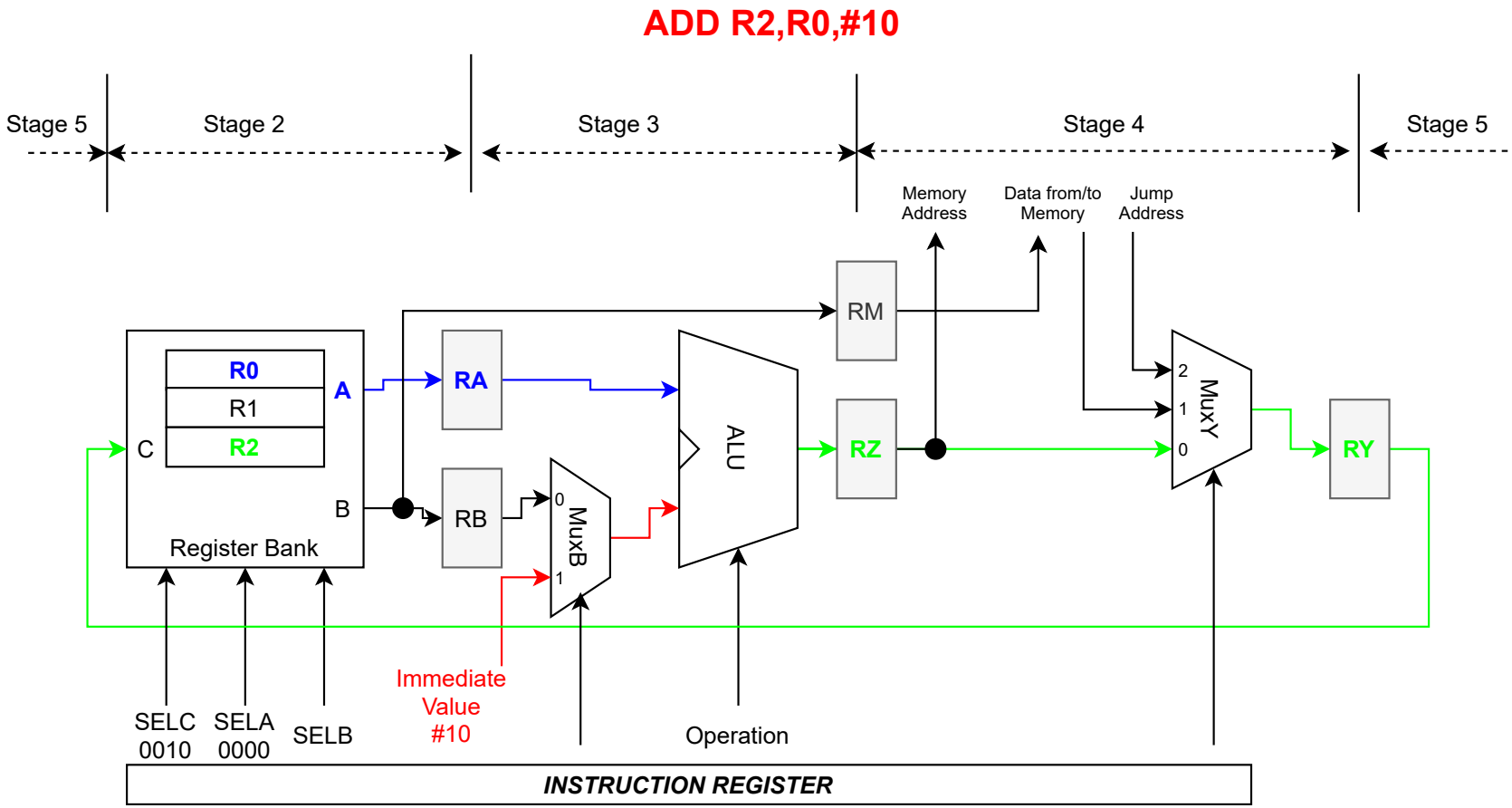
5-Stage Architecture: RISC Processors



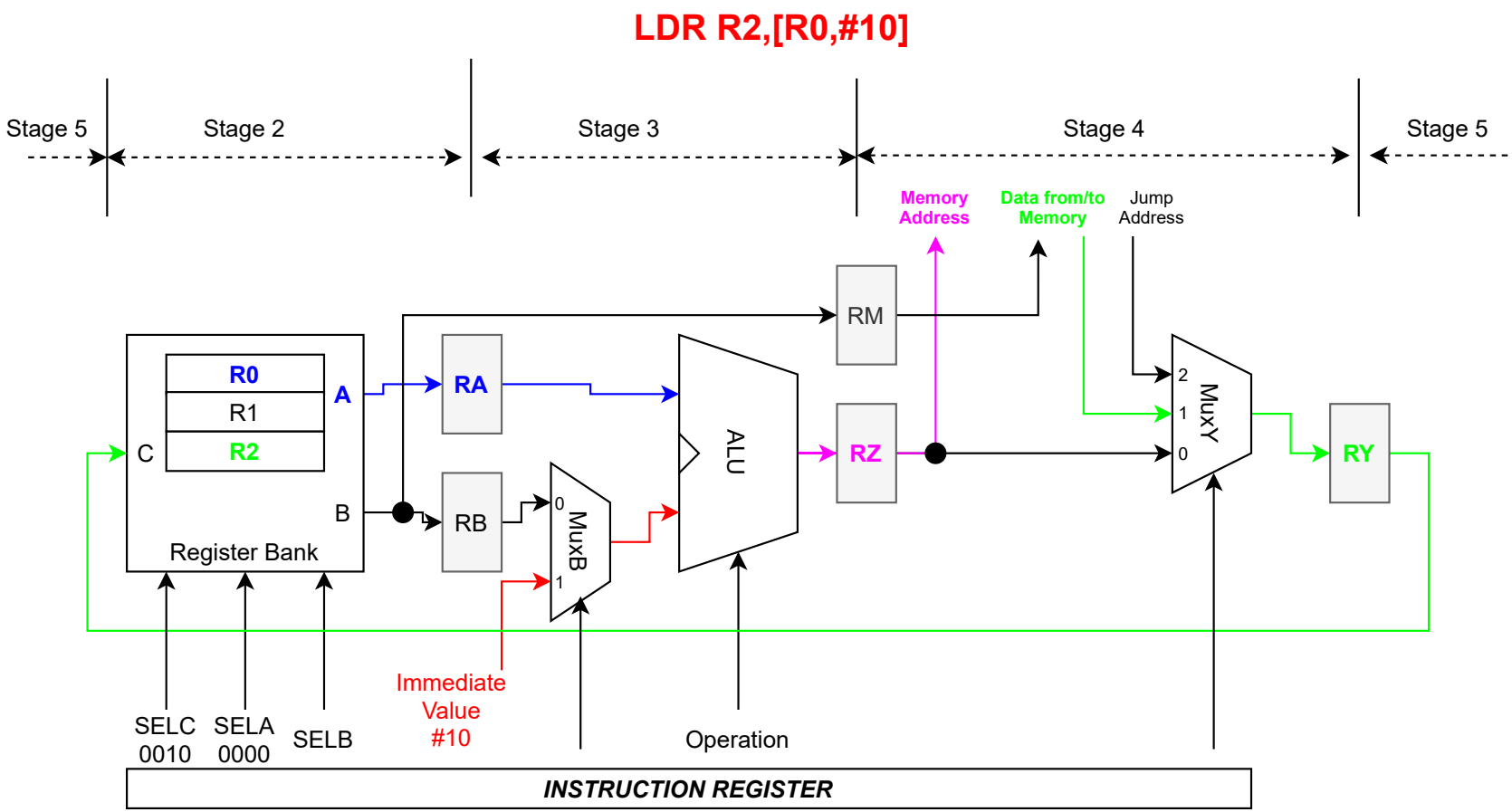
5-Stage Architecture: Esempio di istruzione ADD



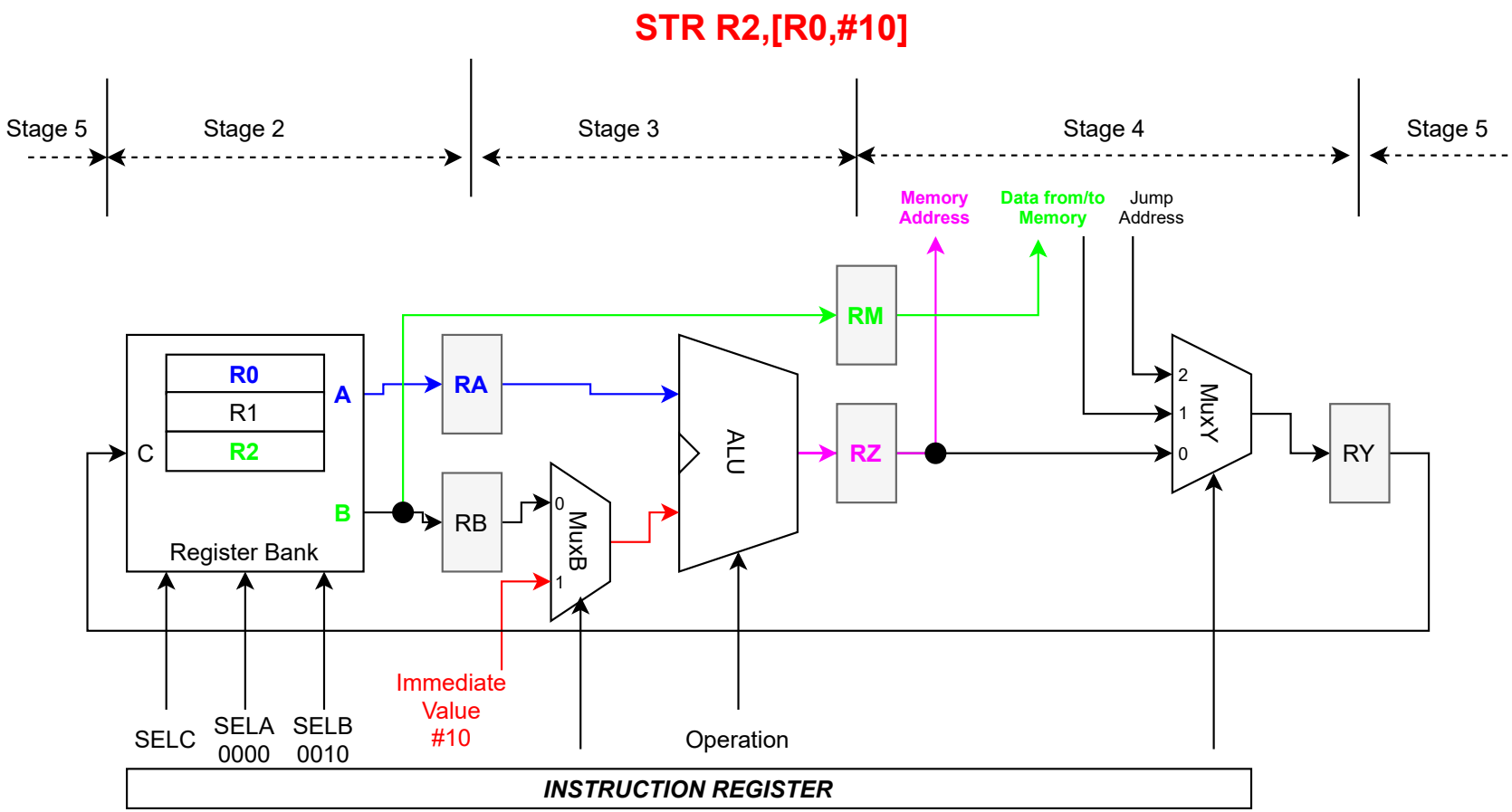
5-Stage Architecture: Esempio di istruzione ADD



5-Stage Architecture: Esempio di istruzione LDR

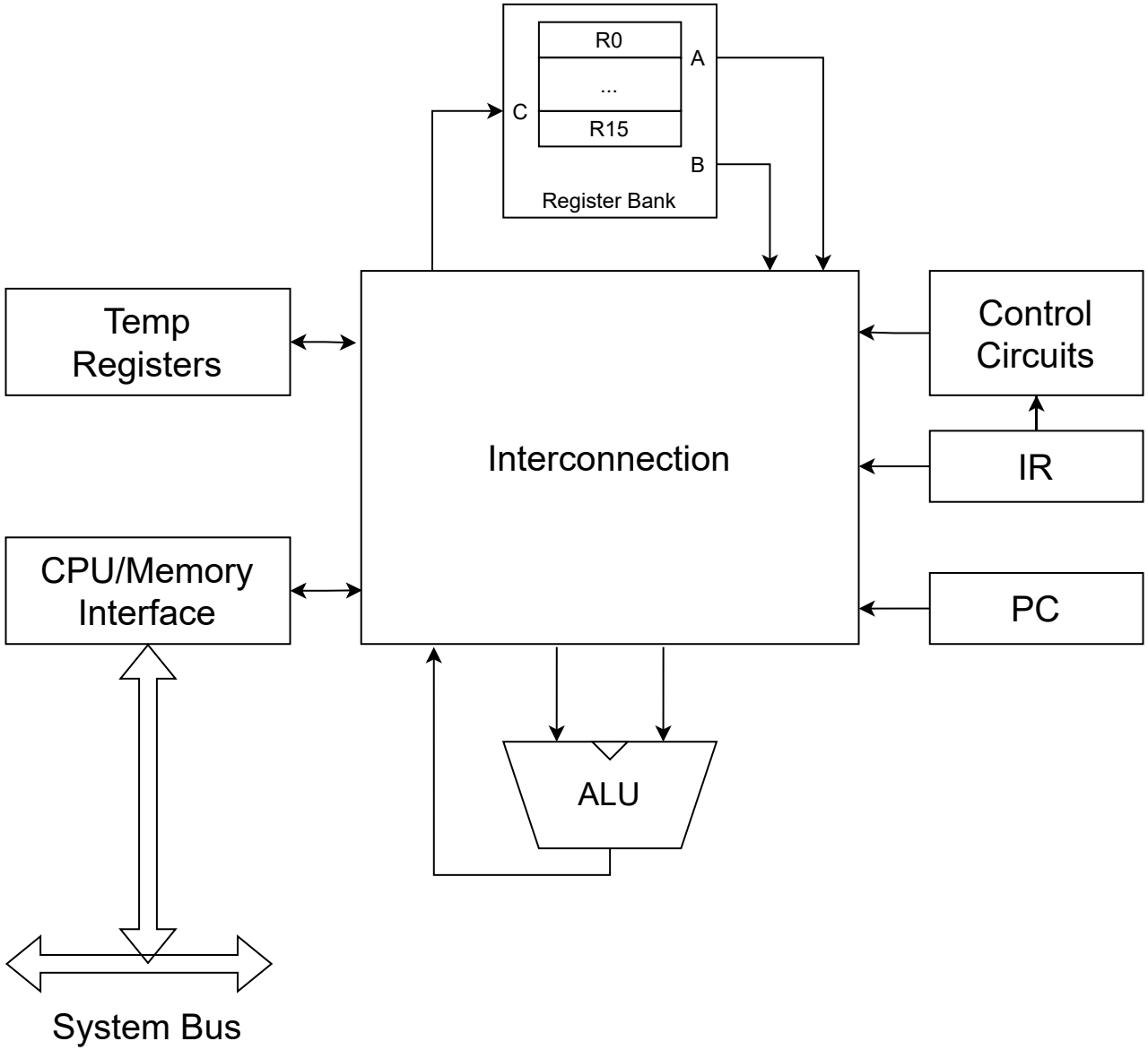


5-Stage Architecture: Esempio di istruzione STR

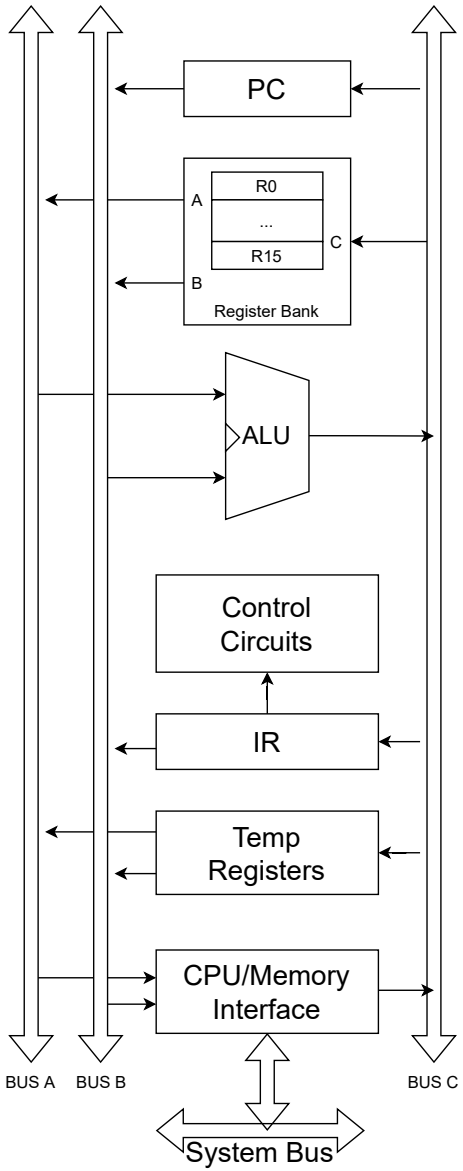


CISC Processors

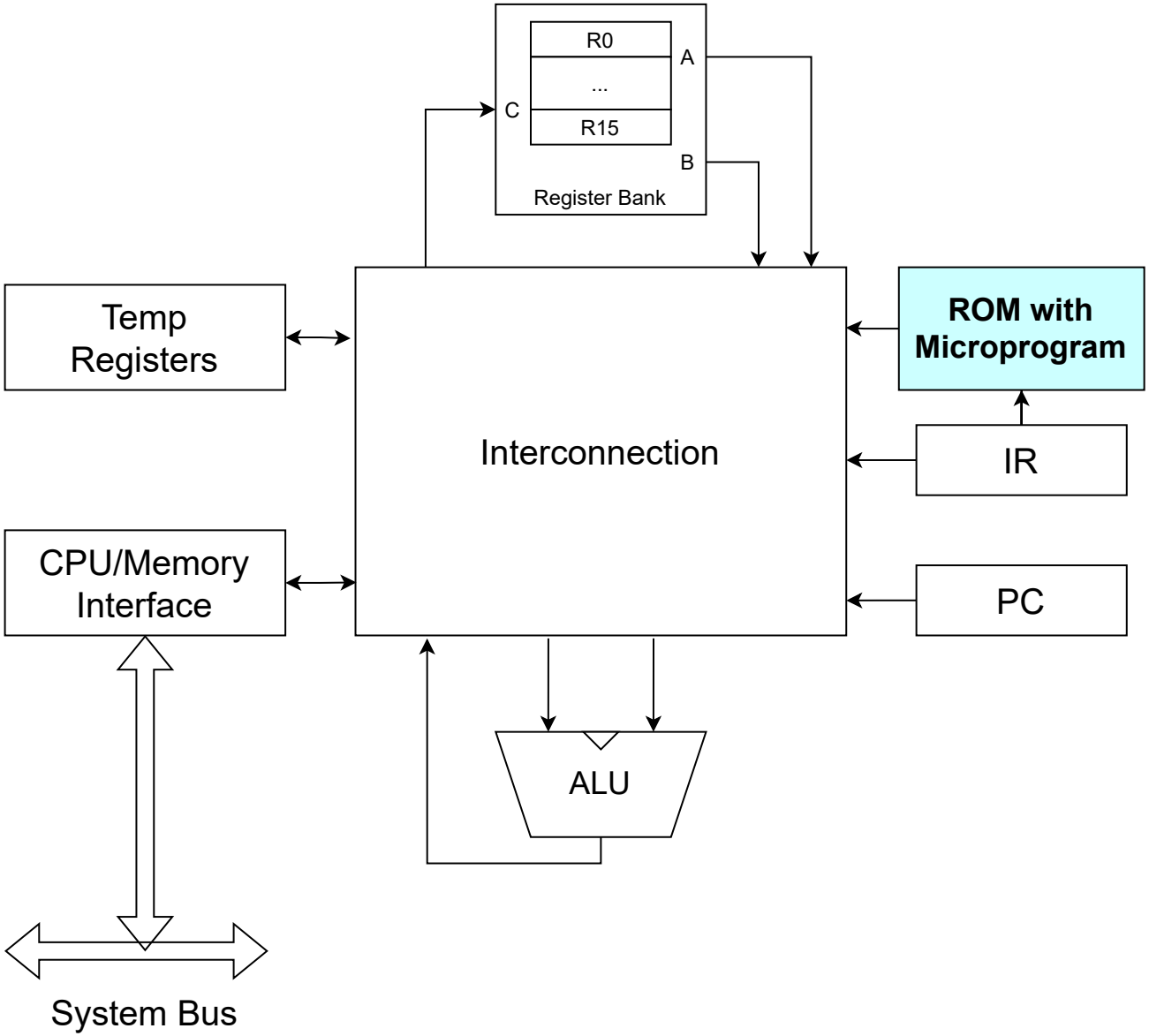
Schema base di un processore CISC



Interconnessione basata su Bus interni



Interconnessione basata su Bus microprogramma



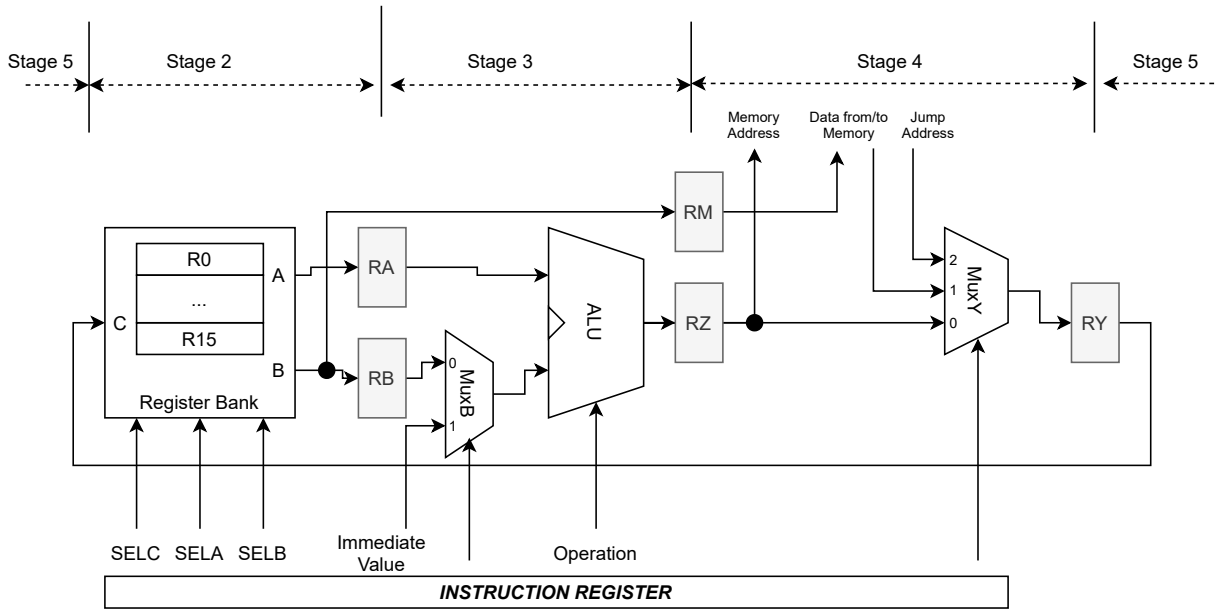
Esempio di Microprogramma

Add R6, [R7, #Offset]

1. $RM \leftarrow [PC], PC \leftarrow [PC] + 4$
2. $IR \leftarrow [[RM]]$
3. Decodifica Istruzione
4. $RM \leftarrow [PC], PC \leftarrow [PC] + 4$
5. $RTemp1 \leftarrow [[RM]]$
6. $RTemp1 \leftarrow [RTemp1] + [R7]$
7. $RM \leftarrow [RTemp1]$
8. $RTemp1 \leftarrow [[RM]]$
9. $RTemp1 \leftarrow [RTemp1] + R6$
10. $R6 \leftarrow [RTemp1]$

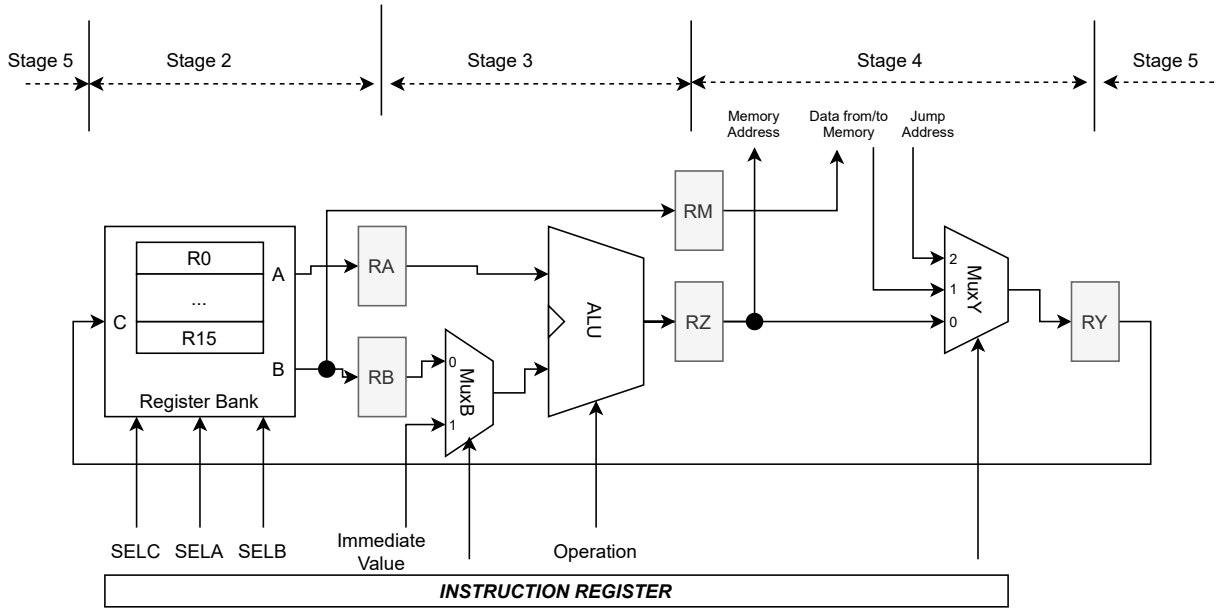
Data Path, Istruzioni e Stages

ALU Instructions



Step	Action
1	Memory address ← [PC], Read memory, IR ← Memory data, PC ← [PC] + 4
2	Decode instruction, RA ← [R4], RB ← [R5]
3	RZ ← [RA] + [RB]
4	RY ← [RZ]
5	R3 ← [RY]

Load Instruction

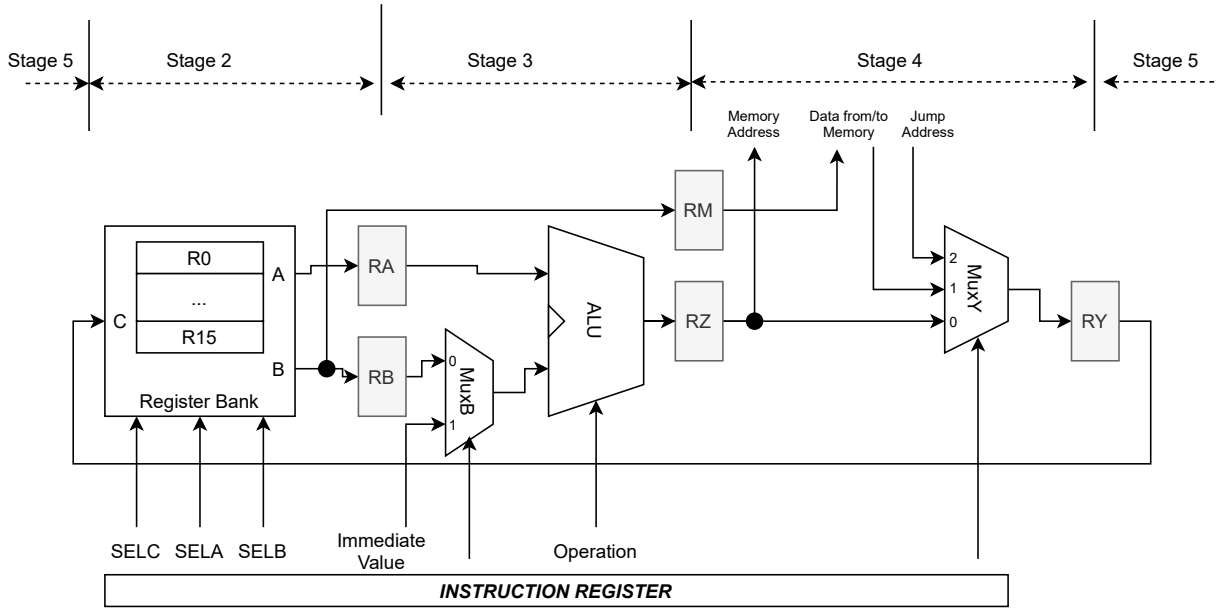


Step	Action
1	Memory address ← [PC], Read memory, IR ← Memory data, PC ← [PC] + 4
2	Decode instruction, RA ← [R7]
3	RZ ← [RA] + Immediate value X
4	Memory address ← [RZ], Read memory, RY ← Memory data
5	R5 ← [RY]

`Load R5,X(R7)`

`LDR R5,[R7, #X]` (ARM)

Store Instruction

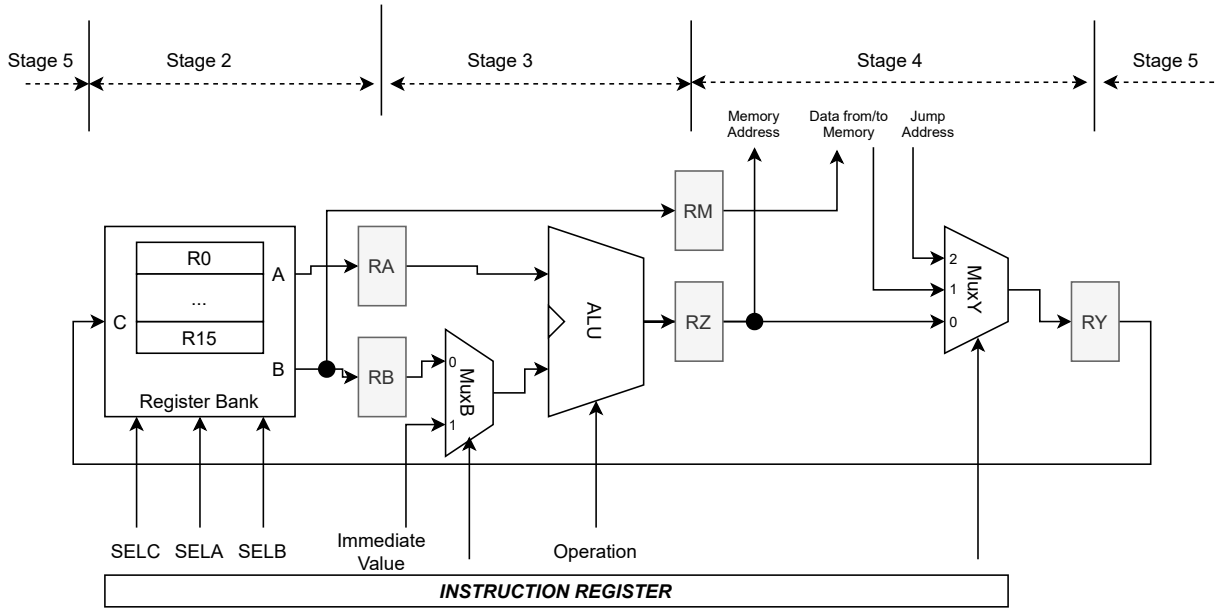


Step	Action
1	Memory address ← [PC], Read memory, IR ← Memory data, PC ← [PC] + 4
2	Decode instruction, RA ← [R8], RB ← [R6]
3	RZ ← [RA] + Immediate value X, RM ← [RB]
4	Memory address ← [RZ], Memory data ← [RM], Write memory
5	No action

`Store R6,X(R8)`

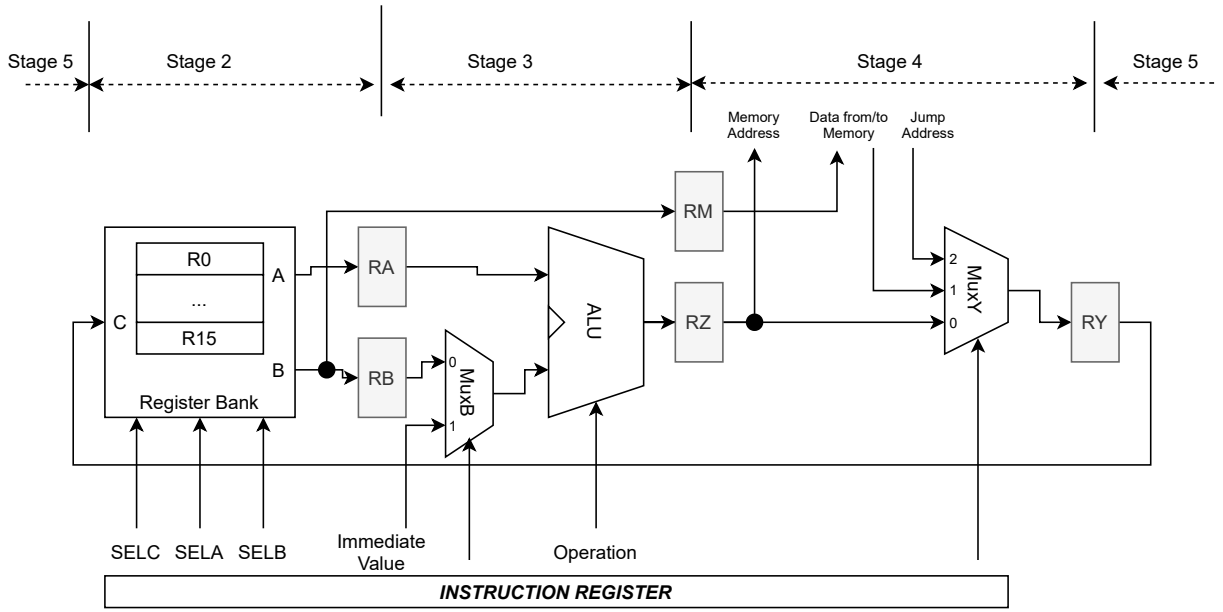
`STR R6,[R8, #X]` (ARM)

Branch Instruction



Step	Action
1	Memory address ← [PC], Read memory, IR ← Memory data, PC ← [PC] + 4
2	Decode instruction
3	PC ← [PC] + Branch offset
4	No action
5	No action

Compare and Branch Instruction

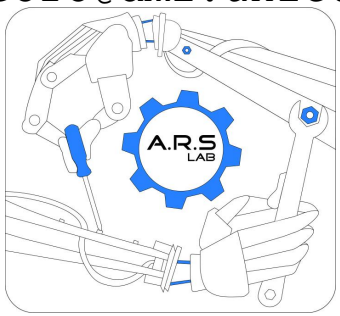


Step	Action
1	Memory address ← [PC], Read memory, IR ← Memory data, PC ← [PC] + 4
2	Decode instruction, RA ← [R5], RB ← [R6]
3	Compare [RA] to [RB], If [RA] = [RB], then PC ← [PC] + Branch offset
4	No action
5	No action

Inside the CPU

Corrado Santoro

Dipartimento di Matematica e Informatica
santoro@dmi.unict.it



Corso di Architettura degli Elaboratori