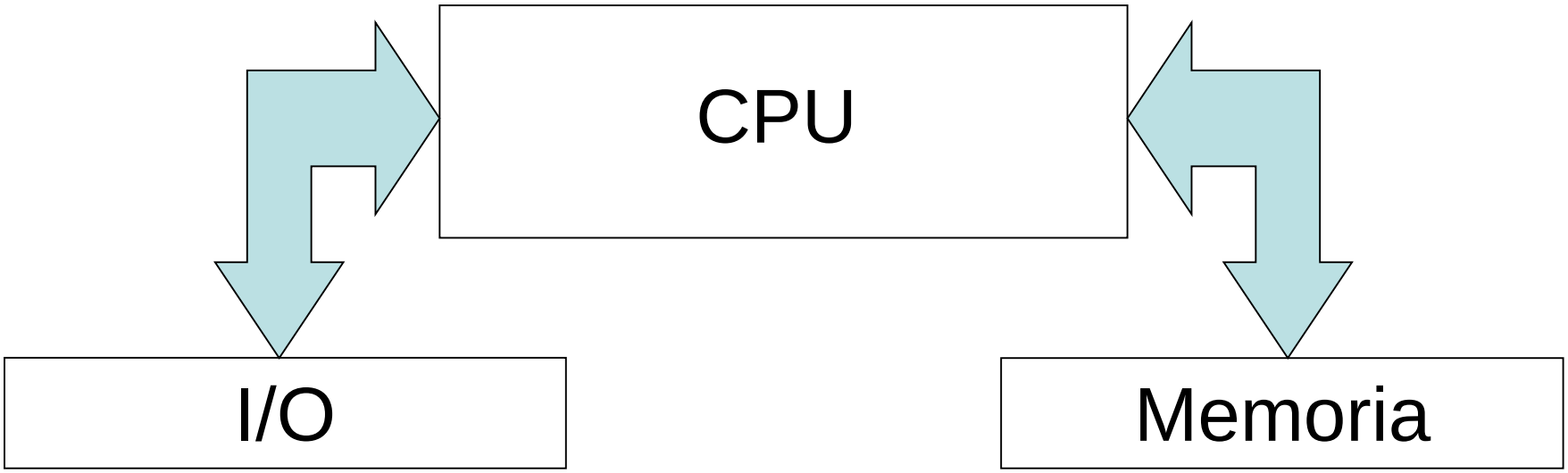
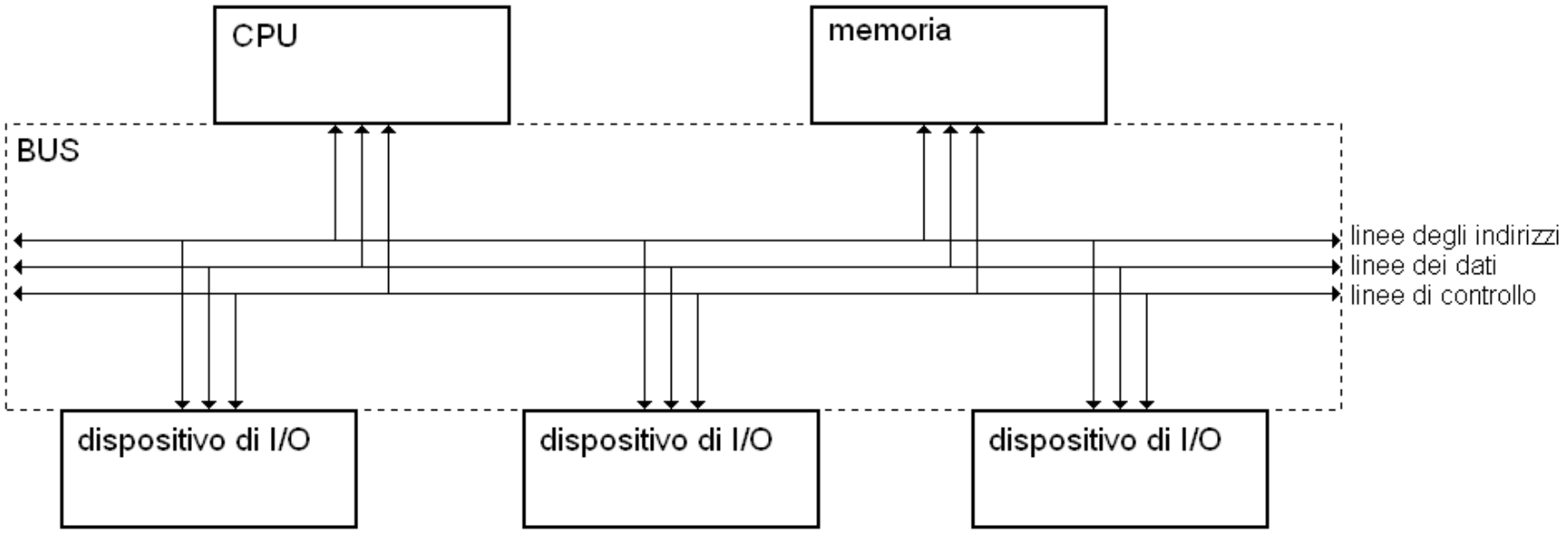
Calcolatore: sottosistemi

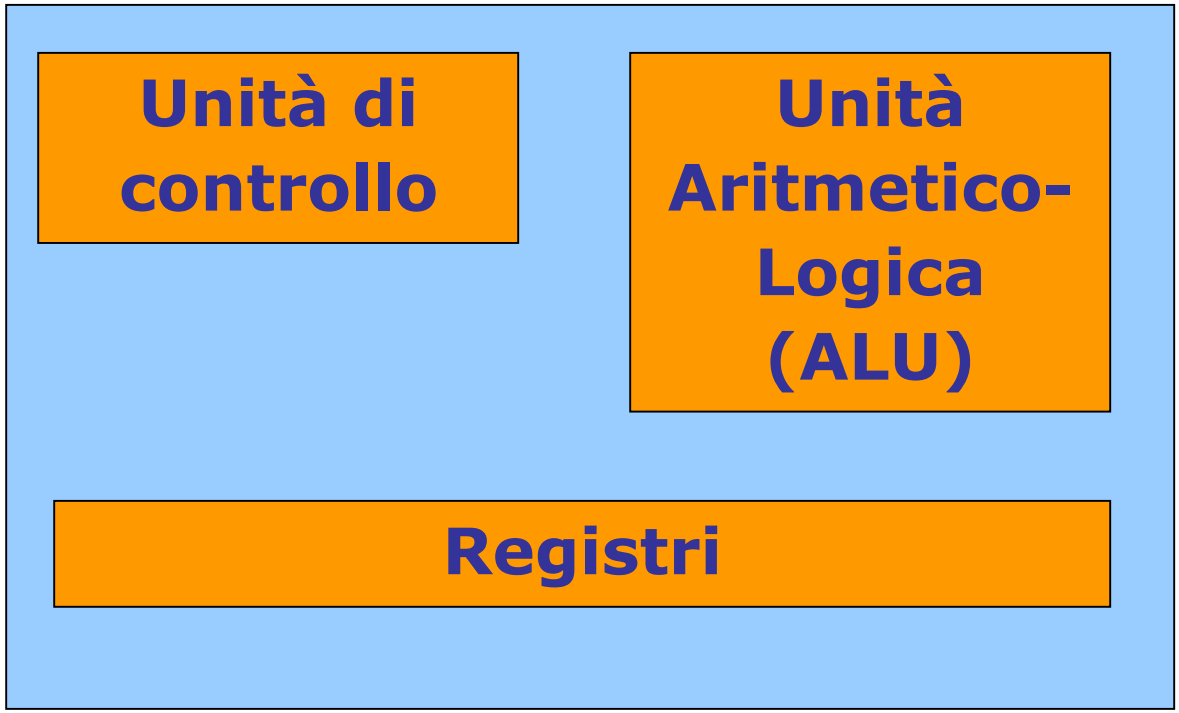
- Processore o CPU (*Central Processing Unit*)
- Memoria centrale
- Sottosistema di input/output (I/O)



Calcolatore: organizzazione a bus

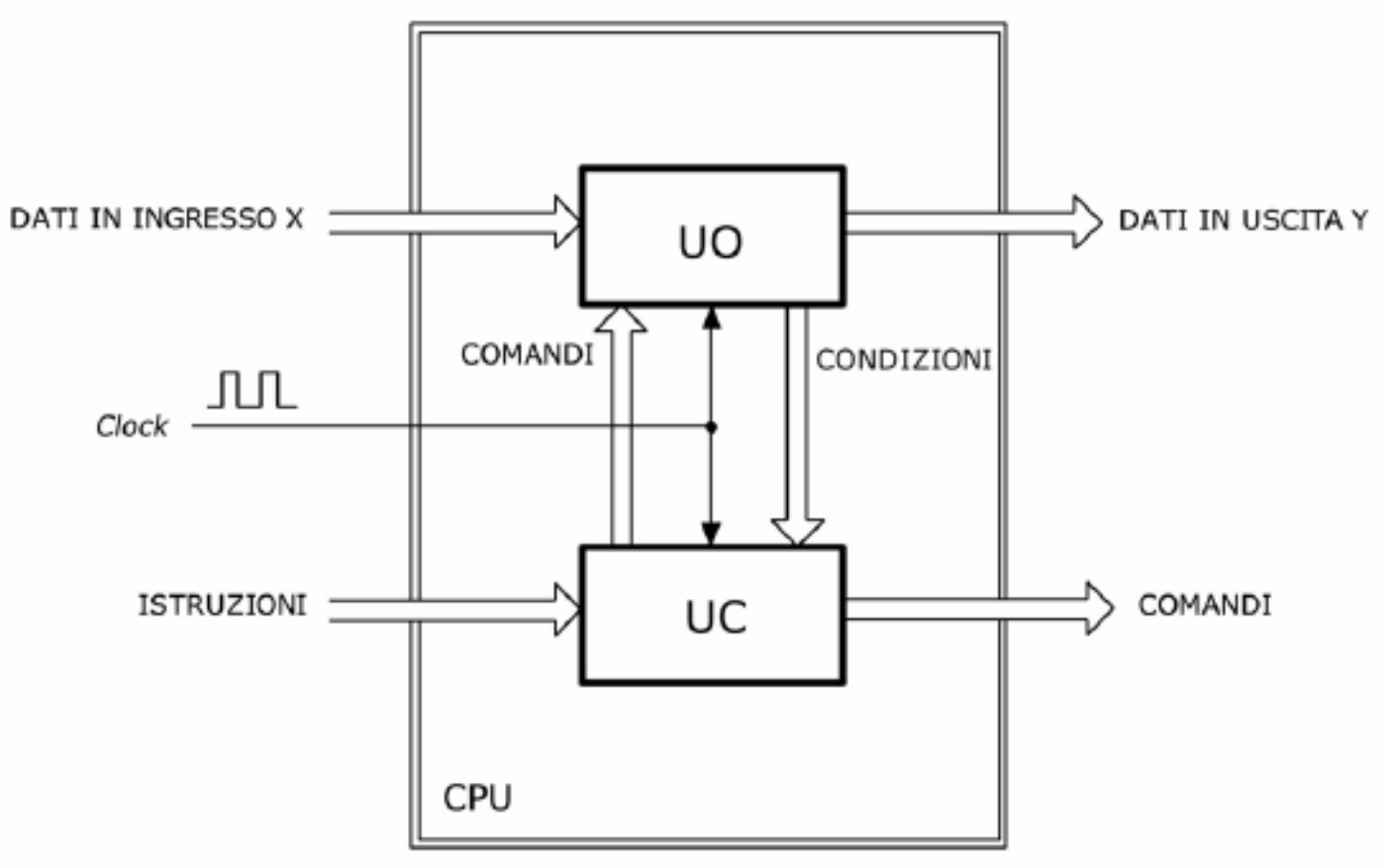


Il processore o CPU



Processore o CPU

CPU: struttura interna



Il funzionamento della CPU è scandito dal clock.

CPU: struttura interna (2)

- Componenti fondamentali del processore:
  - Unità di controllo
    - registro Program Counter (PC) o Prossima Istruzione
    - Instruction Register o registro di decodifica (IR o D)
    - registri di Macchina
  - Unità aritmetico-logica (ALU)
  - Sezione di Collegamento con la memoria
    - registro degli indirizzi di memoria o Memory Address Register MAR
    - registro di transito dei dati dalla memoria DTR o Memory Buffer MB
  - Sezione di Collegamento con Ingresso-Uscita

- Il *linguaggio macchina* di un processore è costituito dalla codifica in binario delle istruzioni eseguibili dal processore

Registri della CPU

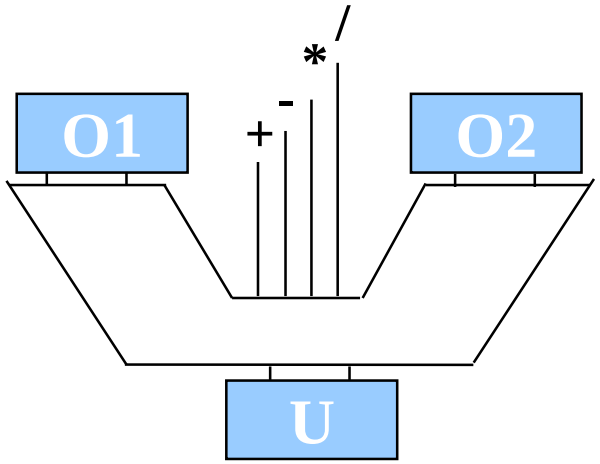
- Registri interni
  - Necessari al funzionamento del processore
  - Non direttamente visibili al programmatore
    - non appartengono al *modello di programmazione*
    - *Es. MAR, MDR, IR, …*
- Registri di macchina
  - Visibili al programmatore
    - appartengono al *modello di programmazione*
      - Registri generali (R0, R1, Rn-1)
      - Registri speciali (PC, SR, …)

Funzioni dei registri di macchina

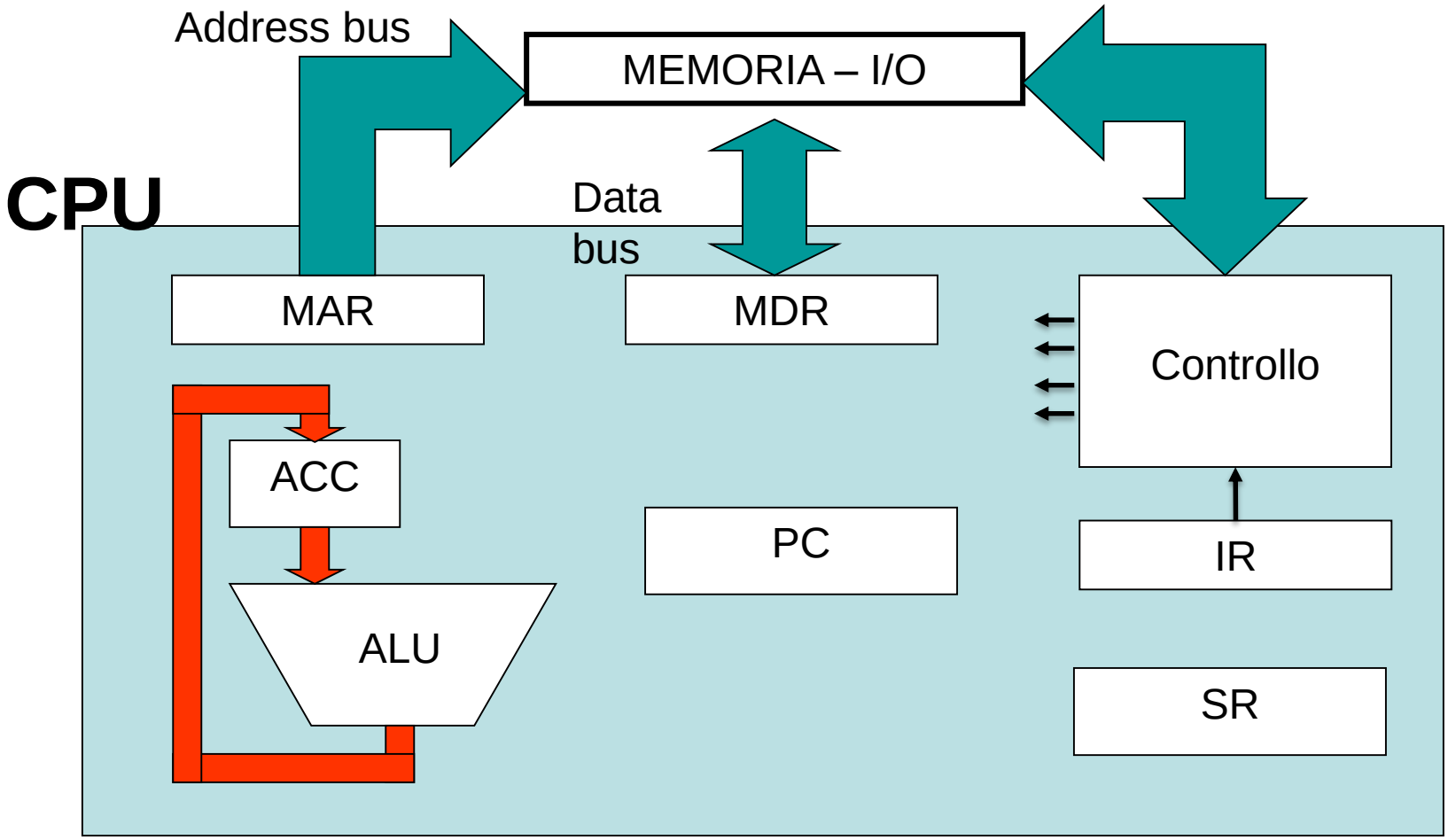
- Indirizzo dell'istruzione corrente (PC)
- Transito dati (qualunque registro generale)
- Accumulazione di risultati
  - es: R0 := NOT R0 oppure R0 := R0 + R1
- Indirizzamento
- Indicatori o flag (Registro di Stato)
- Altre funzioni speciali

Unità Aritmetico-Logica (ALU)

- L'Unità di controllo fornisce alla ALU gli operandi, insieme ad un comando che indica l'operazione da effettuare
- Gli operandi sono copiati nei registri di ingresso della ALU (O1, O2)
- La ALU esegue l'operazione e pone il risultato nel registro risultato (U); inoltre, altera il valore dei flag del registro di stato (SR) in funzione del risultato



Modello architetturale di un processore: modello ad accumulatore



Processori ad accumulatore

- In un processore ad accumulatore tutte le istruzioni aritmetiche, logiche e di confronto hanno un operando in memoria ed un altro (riferito implicitamente) contenuto in un registro interno del processore detto *accumulatore*
- Esempio: per realizzare [x]+[y] $\rightarrow$ z con una macchina ad accumulatore (es. Motorola 6809) occorre eseguire una sequenza di istruzioni del tipo

`LDA x`	[x] $\rightarrow$ accumulatore (Istruzione LOAD)
`ADDA y`	[y]+[accumulatore] $\rightarrow$ accumulatore
`STA z`	[accumulatore] $\rightarrow$ z (Istruzione STORE)

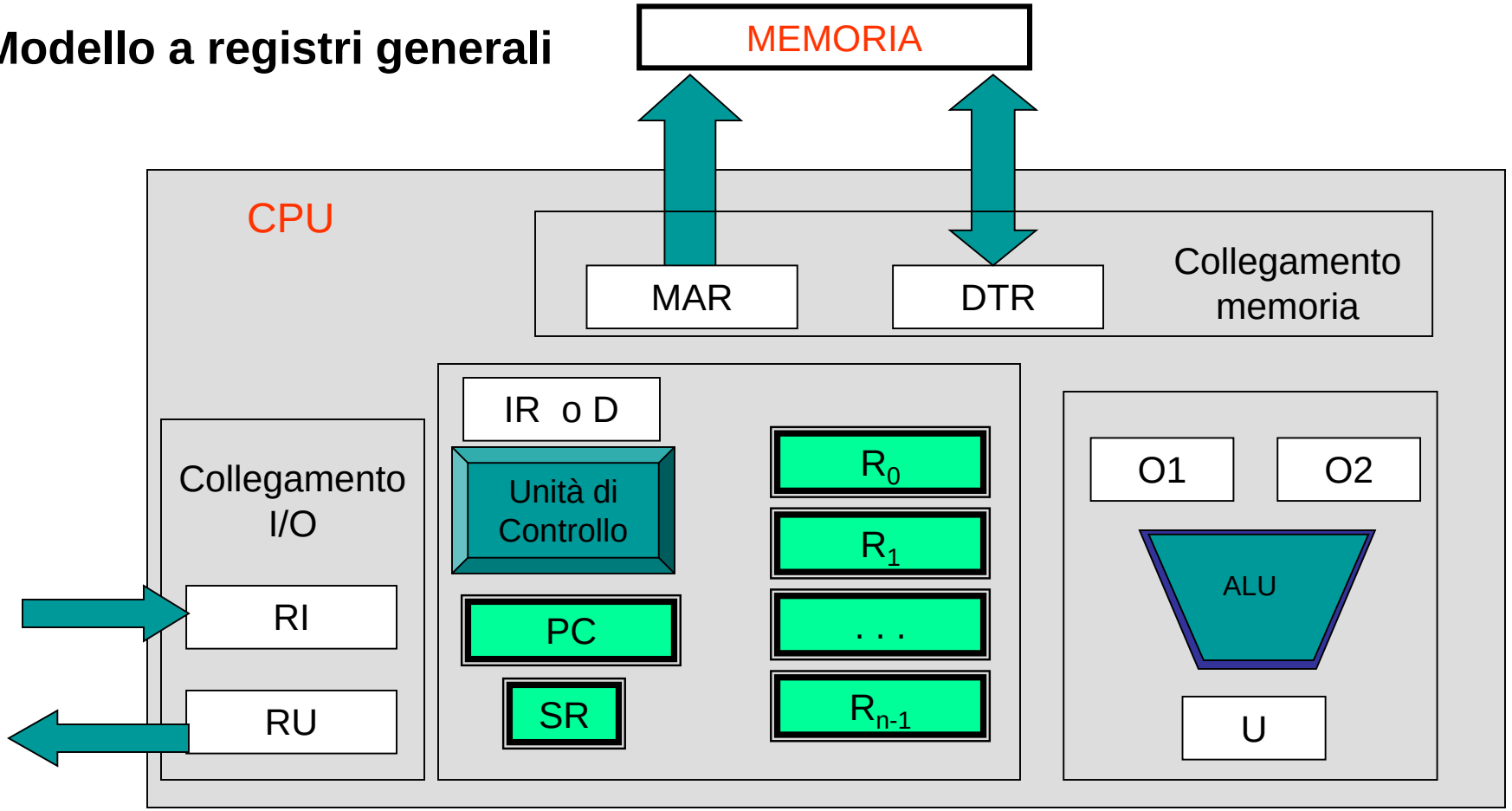
- Dimensione e velocità di esecuzione dei programmi penalizzate dal fatto che tutte le istruzioni devono indirizzare un dato in memoria

Esempio: y=a*b+c*d

`LDA a`	[a] → accumulatore (Istruzione LOAD)
`MULU b`	[b]*[accumulatore] → accumulatore
`STA t`	[accumulatore] → t (Istruzione STORE)
`LDA c`	[c] → accumulatore (Istruzione LOA D)
`MULU d`	[d]*[accumulatore] → accumulatore
`ADDA t`	[t]+[accumulatore] → accumulatore
`STA y`	[accumulatore] → y (Istruzione STORE)

Modello architetturale di un processore: modello a registri generali

Modello a registri generali



Processore a registri generali

- Il processore dispone di un set di registri R_0, R_1, ..., R_{N-1} utilizzabili indifferentemente dal programmatore
- Le istruzioni che operano su registri sono più veloci di quelle che operano su locazioni di memoria
- Il programmatore può utilizzare i registri del processore per memorizzare i dati di uso più frequente
  - concetto di gerarchia di memorie
- Istruzioni con operandi registri:
  $[R_0] + [R_1] \rightarrow R_1$
- Istruzioni con operandi memoria-registri:
  $[R_0] + M[1000] \rightarrow R_0$ *memory-to-register*
  $M[1000] + [R_1] \rightarrow M[1000]$ *register-to-memory*

Architetture a stack

- In un’architettura a stack si impiega una struttura di memoria organizzata a stack
- Lo stack può essere realizzato all’interno della CPU e/o utilizzando memorie esterne
- Gli operandi devono essere memorizzati sullo stack ed il risultato di una qualsiasi operazione (eseguita sullo stack) sostituisce successivamente gli operandi
- Di solito il valore in cima allo stack viene «duplicato» all’interno della CPU

CPU: struttura interna ad 1 bus

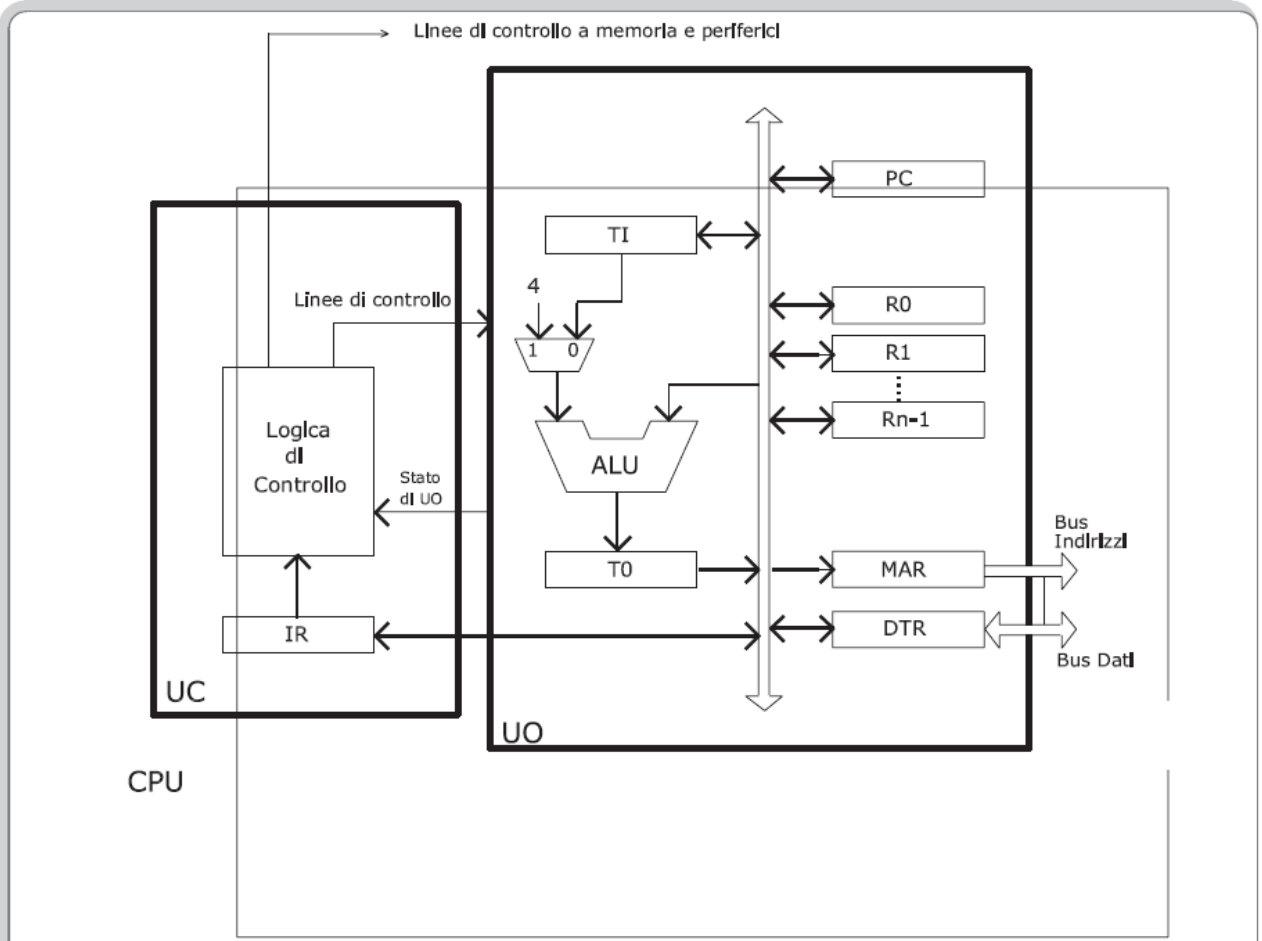
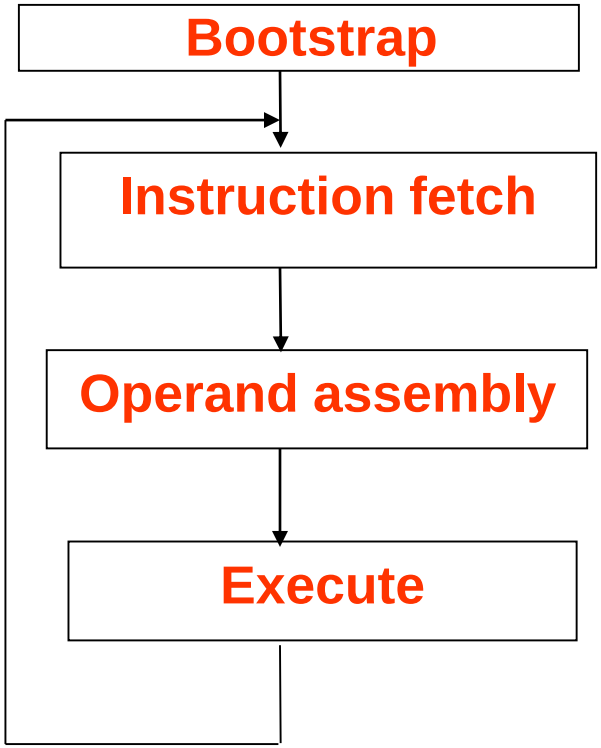


Figura 5.10 - Organizzazione del percorso dati a singolo bus interno.

Algoritmo del processore

- L'unità di controllo opera in un ciclo infinito:
  1. Prelievo
  2. Preparazione degli operandi
  3. Esecuzione



Nella fase di bootstrap il ciclo viene inizializzato;

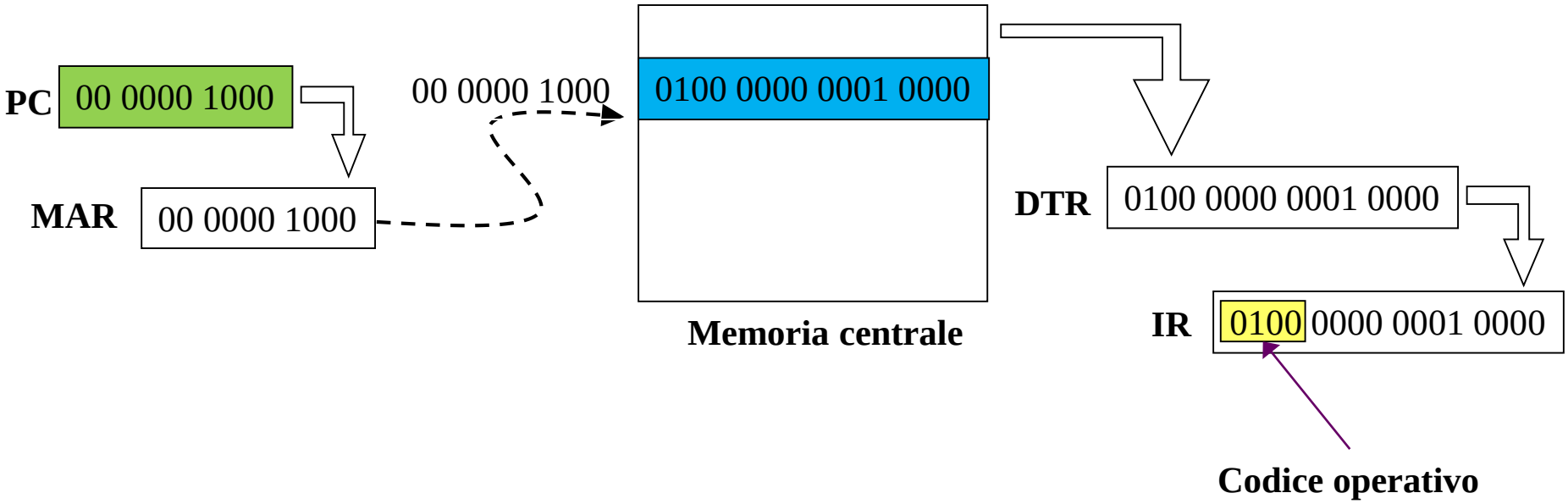
viene avviata l'esecuzione di un programma iniziale in ROM assegnando un valore iniziale opportuno a PC

Algoritmo del processore

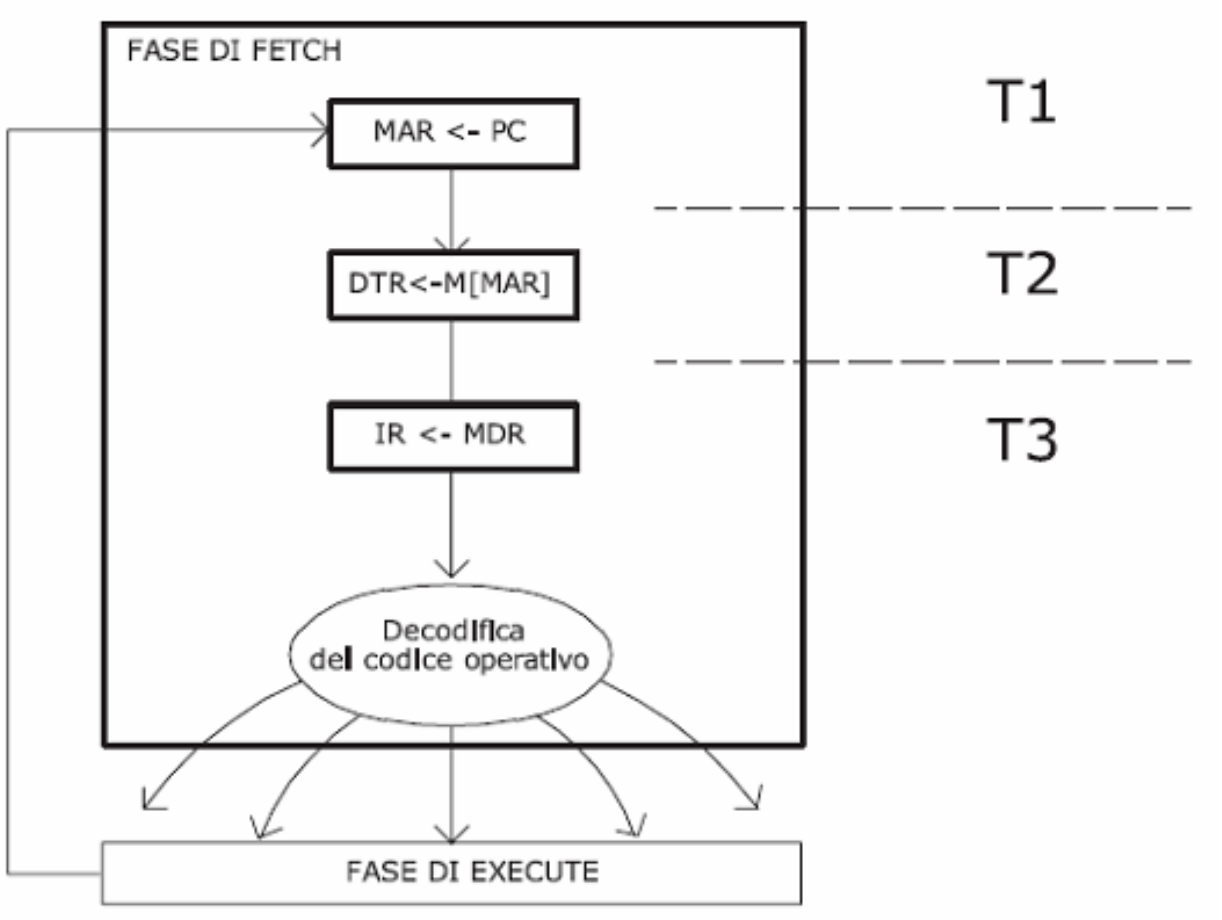
- Prelievo dell'istruzione (Fetch)
  - La CPU preleva dalla memoria l'istruzione il cui indirizzo è in PC
  - L'istruzione viene copiata nel registro IR
- Decodifica / prelievo degli operandi (Operand Assembly)
  - L'unità di controllo esamina il contenuto di IR e ricava il tipo di operazione ed i relativi operandi
  - Eventuali operandi contenuti in memoria vengono prelevati
- Esecuzione dell'istruzione (Execute)
  - L'unità di controllo richiede all'ALU di effettuare l'operazione specificata nell'istruzione ed invia il risultato ad un registro o alla memoria

Fase fetch

- IR=M[PC]; PC=PC+k

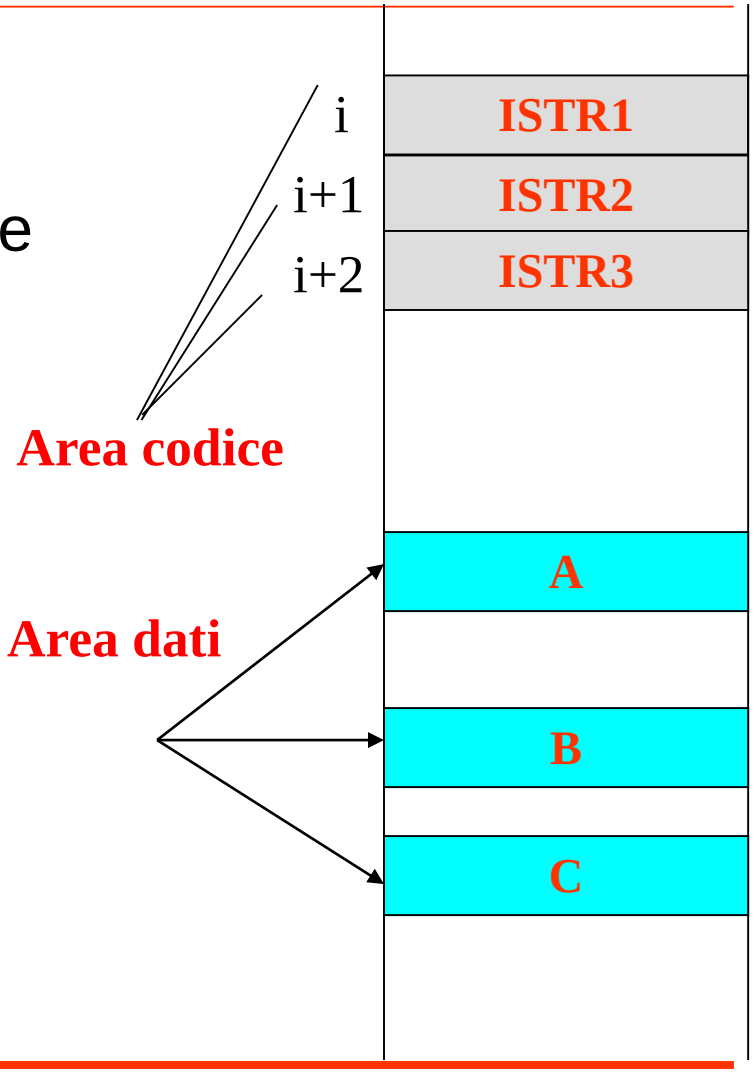


Fase fetch: sottopassi



Esecuzione sequenziale delle istruzioni

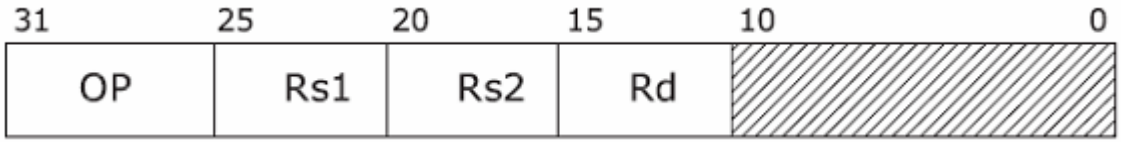
- Alla fine della fase fetch:
  - PC=PC+k
  - k = lunghezza istruzioni in byte
- serve a far sì che PC punti all'istruzione posta subito dopo
- Esecuzione delle istruzioni in sequenza così come sono memorizzate
- Per cicli e figure di controllo (if-then, if-then-else, switch) occorrono istruzioni di salto



Codifica delle istruzioni

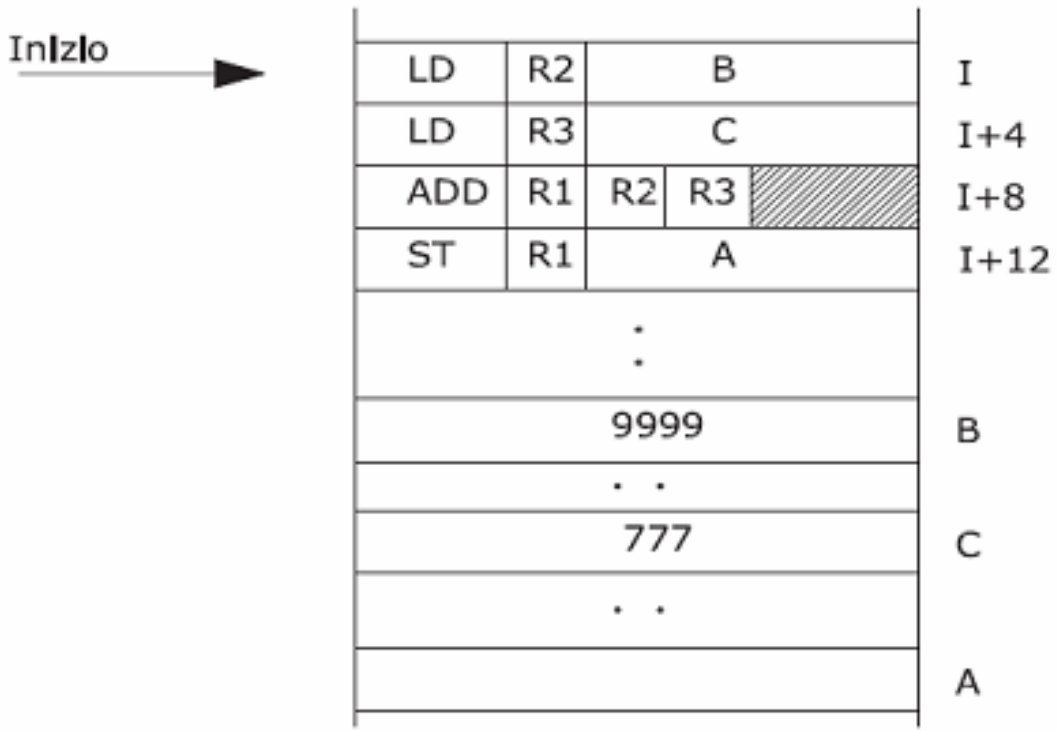
ESEMPIO: una CPU con istruzioni a lunghezza fissa di 32 bit

LD R1, Var ; R1 ← M[Var]



ADD R1, R2, R3 ; R1 ← R2 + R3

Sequenze di istruzioni in memoria



Lo statement *a* = *b* + *c* si traduce come:

```
LD   R2, B          ;B indirizzo a cui è allocata la parola b
LD   R3, C          ;C indirizzo a cui è allocata la parola c
ADD R1, R2, R3
ST   A, R1          ;A indirizzo a cui è allocata la parola a
```