Componenti e organizzazione di un elaboratore

Argomento:

- Programmazione in hardware/software
- Struttura di un elaboratore elementare
- Ciclo di un'istruzione
- Interconnessioni di un elaboratore

Materiale:

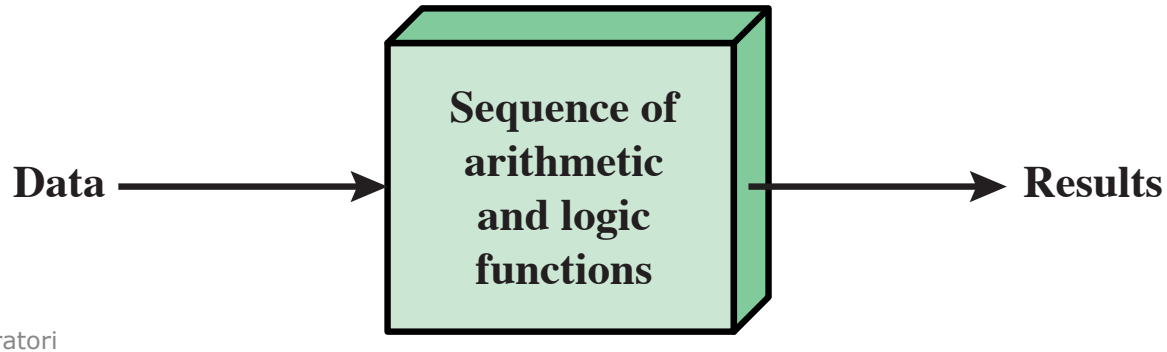
- Capitolo 3: 3.1, 3.2 (no sezione "Interrupts"), 3.3, 3.4

Programmazione in hardware e software

Come implementare un programma?

Programmazione in hardware (hardwired programming)

- Esiste un insieme di componenti logiche base (and, or, multiplexer, ROM,…)
- Un programma è un opportuno circuito costituito da questi elementi
- I dati di input vengono processati dal circuito per produrre il risultato di output



Programmazione in hardware

PRO

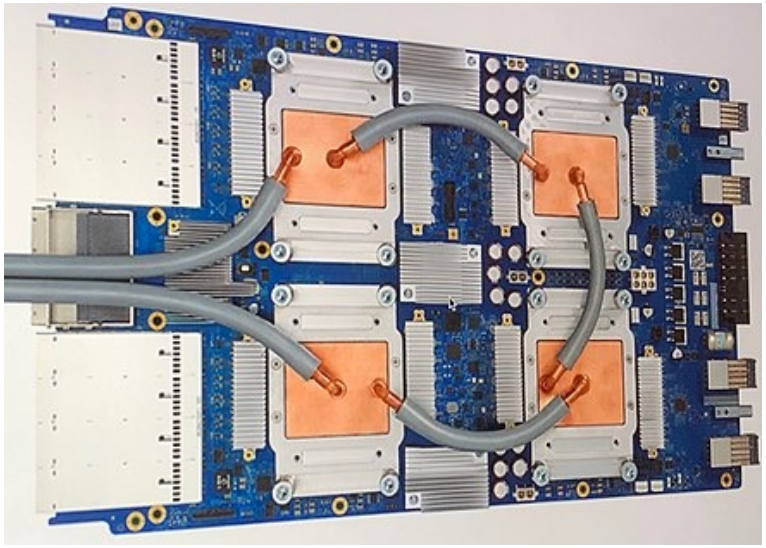
- Permette di implementare un programma con area del circuito minima
- Esecuzione veloce
- Risparmio energetico

CONTRO

- Il programma non può essere modificato
- Costo di implementazione
- Approccio specifico e non generale

Esempi di programmazione in hardware

- Questa tecnica è attualmente molto usata per accelerare l'esecuzione di problemi di machine learning



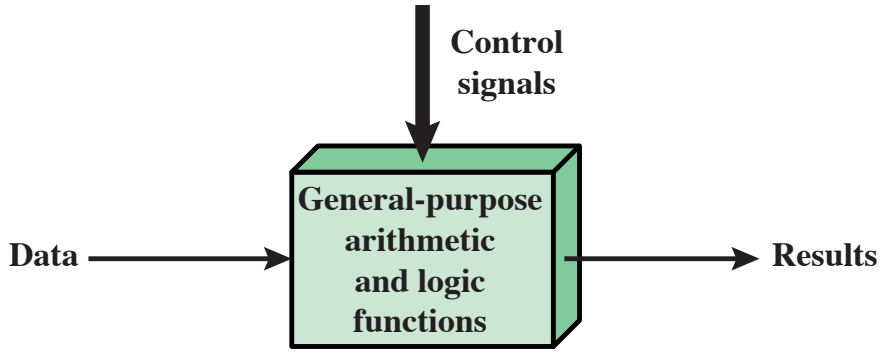
Google Tensor Processing Unit

Nvidia Quadro Volta

Programmazione in software

Programmazione in software:

- Una famiglia di funzioni logiche e aritmetiche è implementata in hardware
- Dei **segnali di controllo** indicano quali funzioni, e in che ordine, attivare sui dati di input
- Si possono implementare più funzioni variando i segnali di controllo



Programmazione in software

PRO

- Il programma può essere modificato
- Costo di implementazione di un programma ridotto
- Universalità

CONTRO

- Minor efficienza di calcolo
- Circuiti più estesi e costosi
- Maggior consumo energetico

Confronto

Programmazione in hardware

Programmazione in software

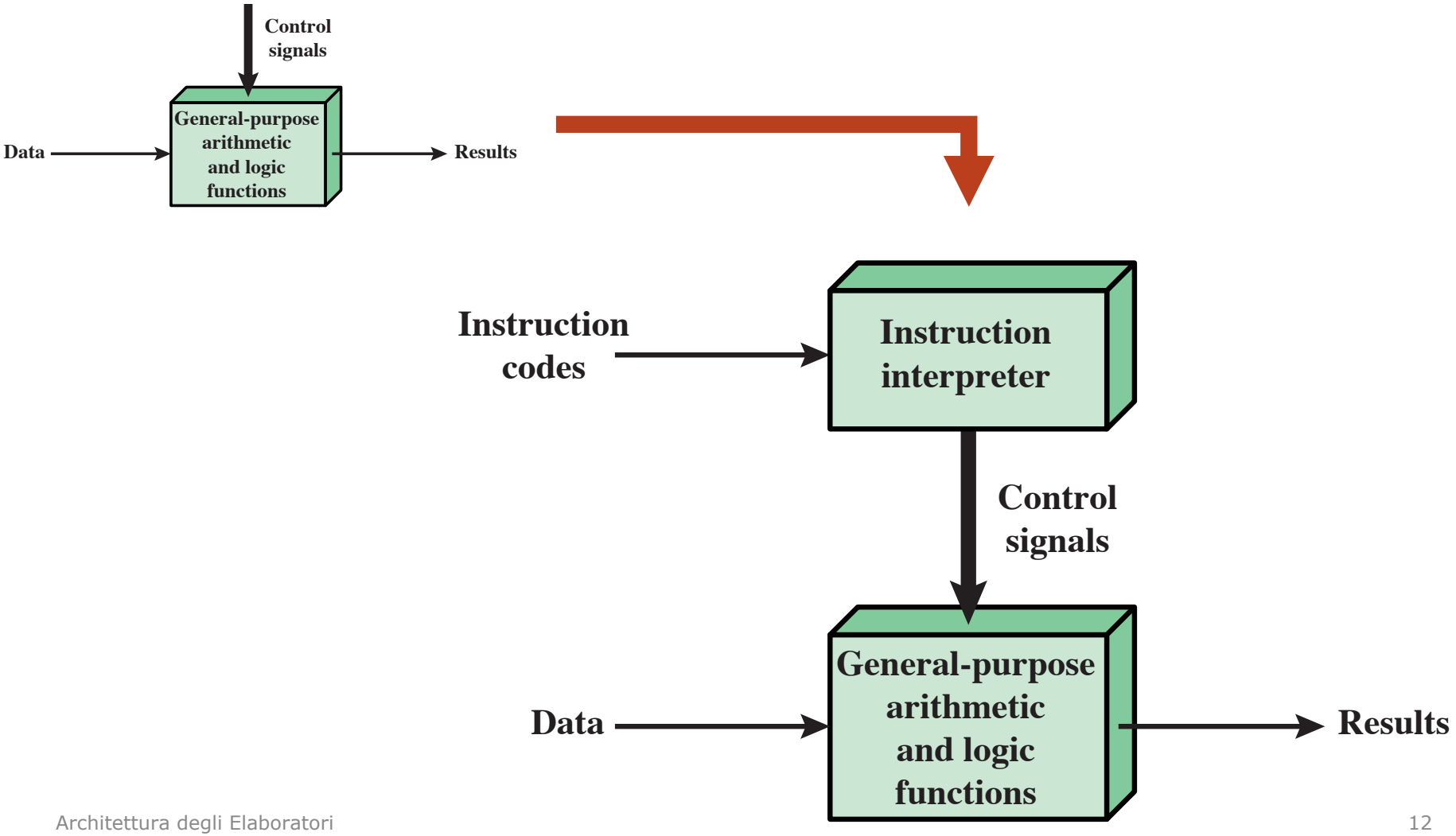
Struttura di un elaboratore

Un sistema per la programmazione in software

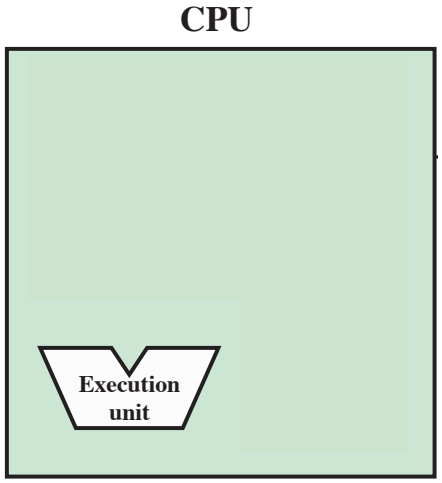
Come definire i segnali di controllo?

- Si assegna un **codice** univoco ad ogni possibile sequenza di segnali
- Si aggiunge una componente hardware per tradurre un codice in segnali
- Ogni codice denota un'**istruzione** che determina in modo univoco i segnali da attivare
- Un programma è definito dalla sequenza di istruzioni/codici (**software**)

Un sistema per la programmazione in software (2)



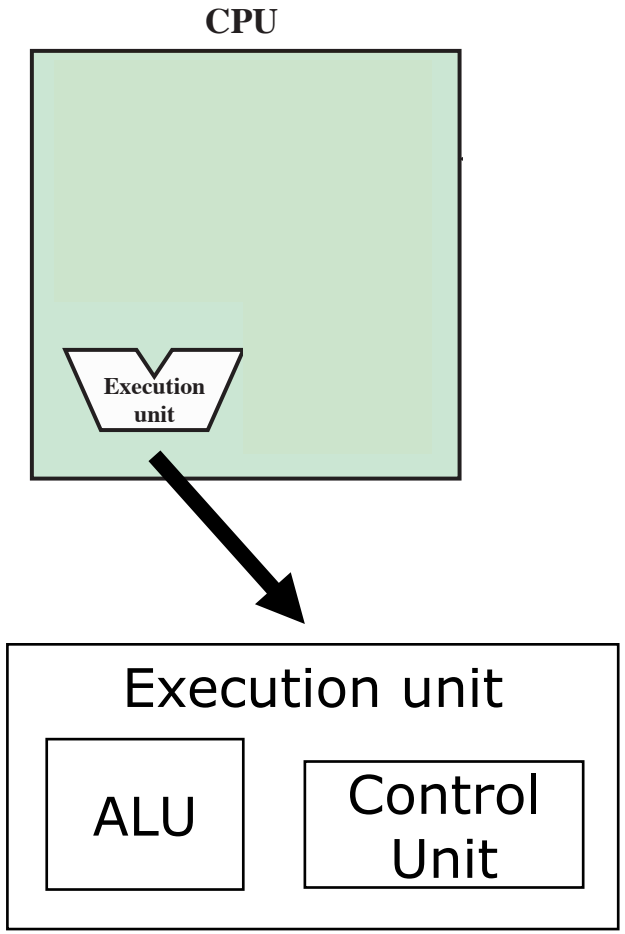
Un sistema per la programmazione in software: CPU



Serve un modulo per interpretare ed eseguire le istruzioni

Il modulo è chiamato **Central Processing Unit** (CPU)

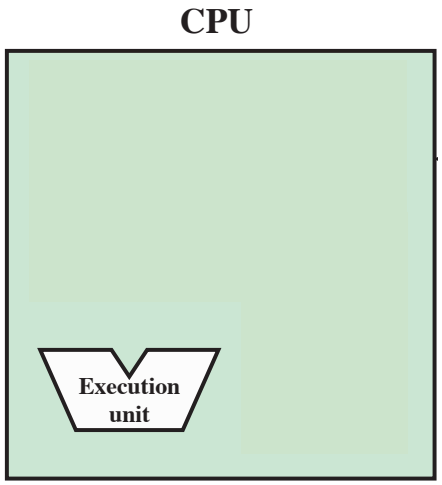
Un sistema per la programmazione in software: CPU (2)

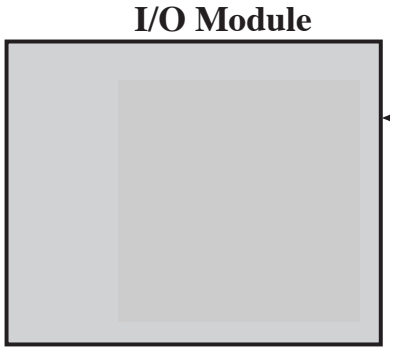


L'unità di esecuzione contiene delle unità per:

- Arithmetic and Logic Unit (ALU)
- Unità di controllo

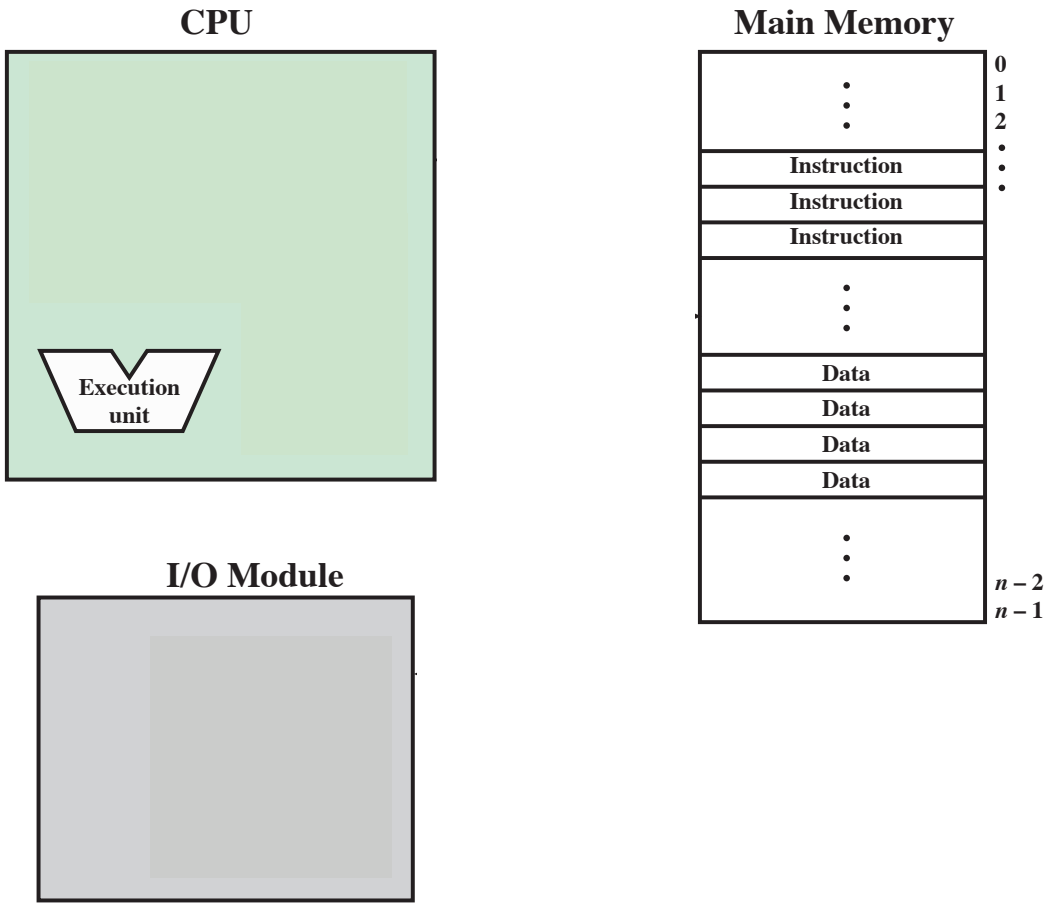
Un sistema per la programmazione in software: I/O





Serve un **modulo input/output** (I/O) per immettere l'input (dati e software) ed emettere l'output (risultato)

Un sistema per la programmazione in software: memoria

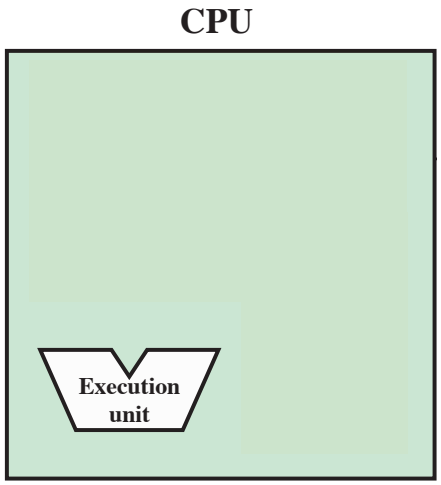


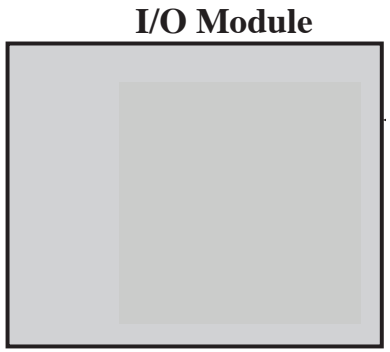
Serve una **memoria** per salvare:

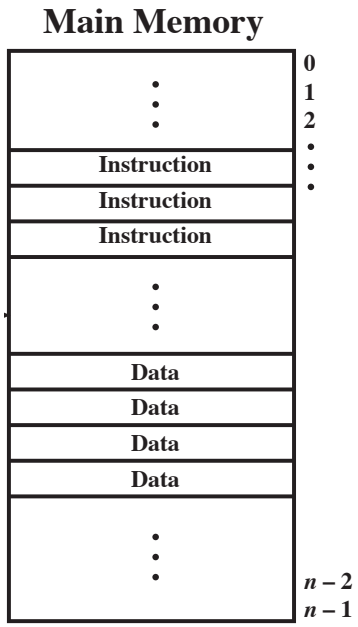
- Istruzioni
- Dati di input
- Dati temporanei

La memoria permette di accedere a dati/istruzioni durante l'esecuzione

Un sistema per la programmazione in software: memoria (2)



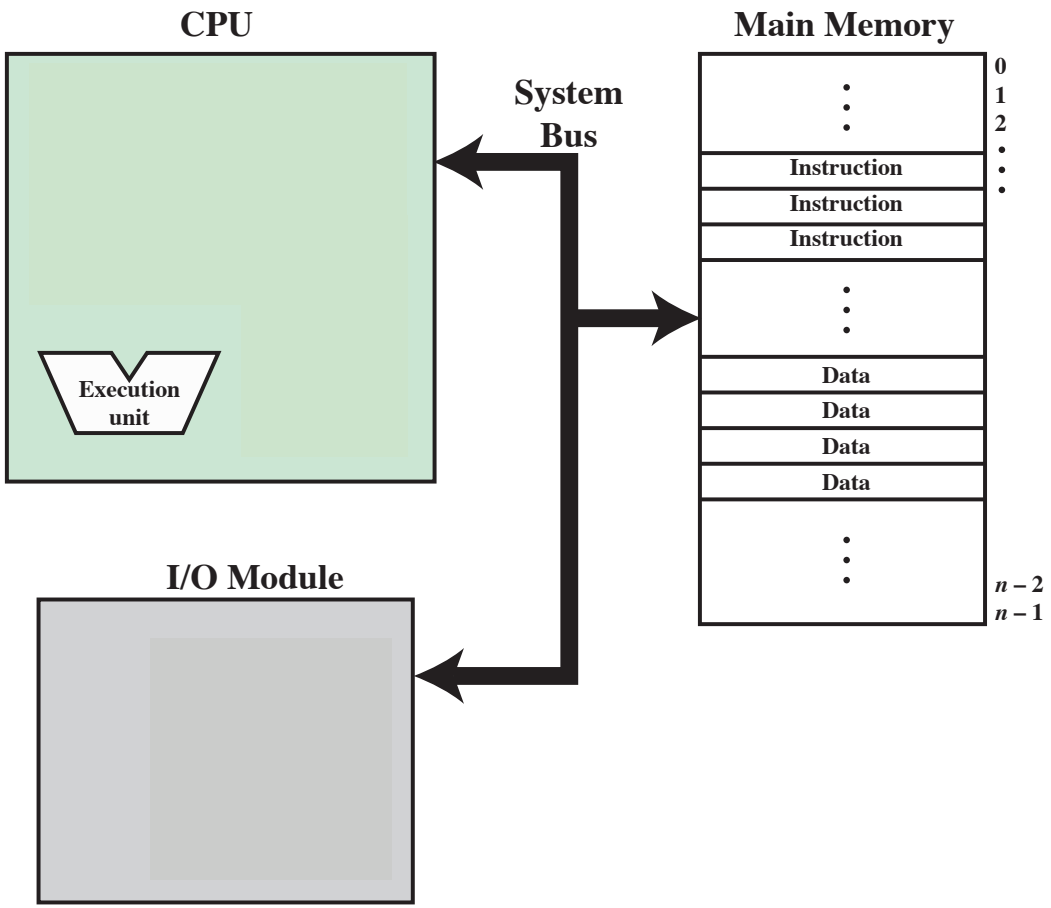




La memoria è divisa in locazioni

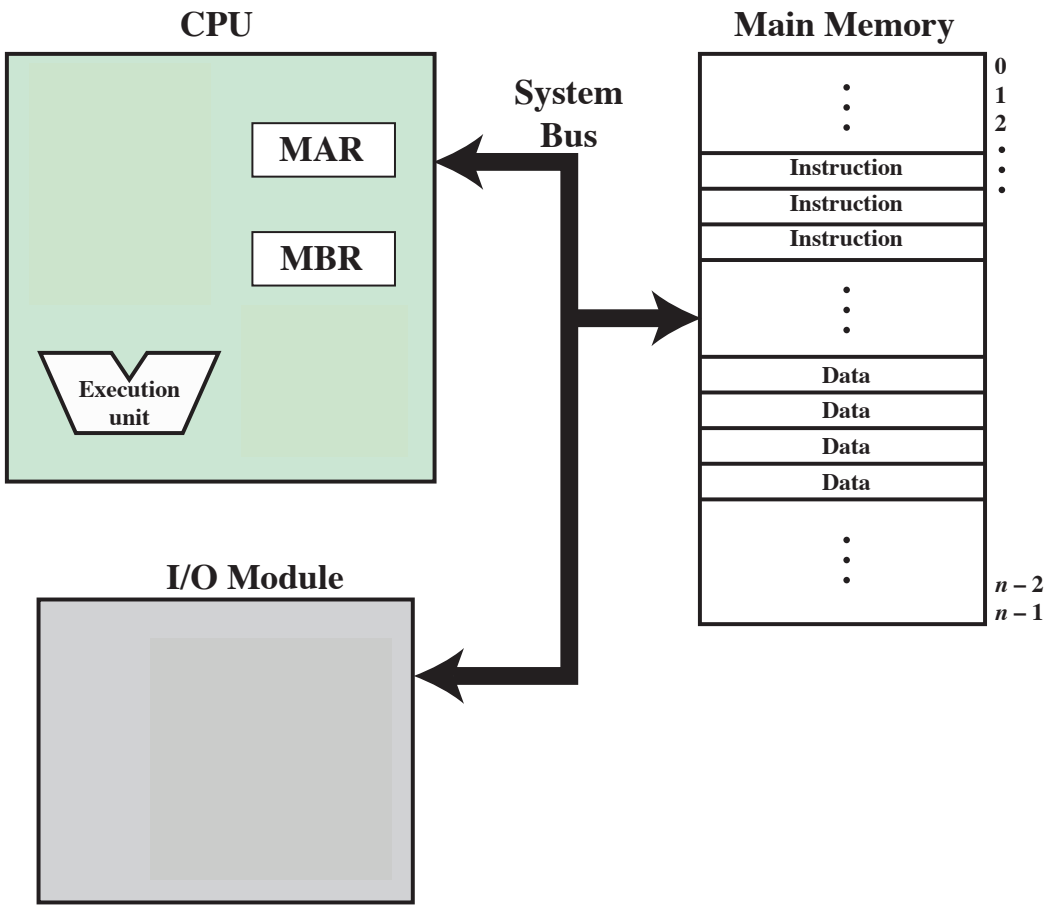
Ogni locazioni è accessibile tramite un indirizzo

Un sistema per la programmazione in software: bus



Serve un **bus** per connettere i vari moduli

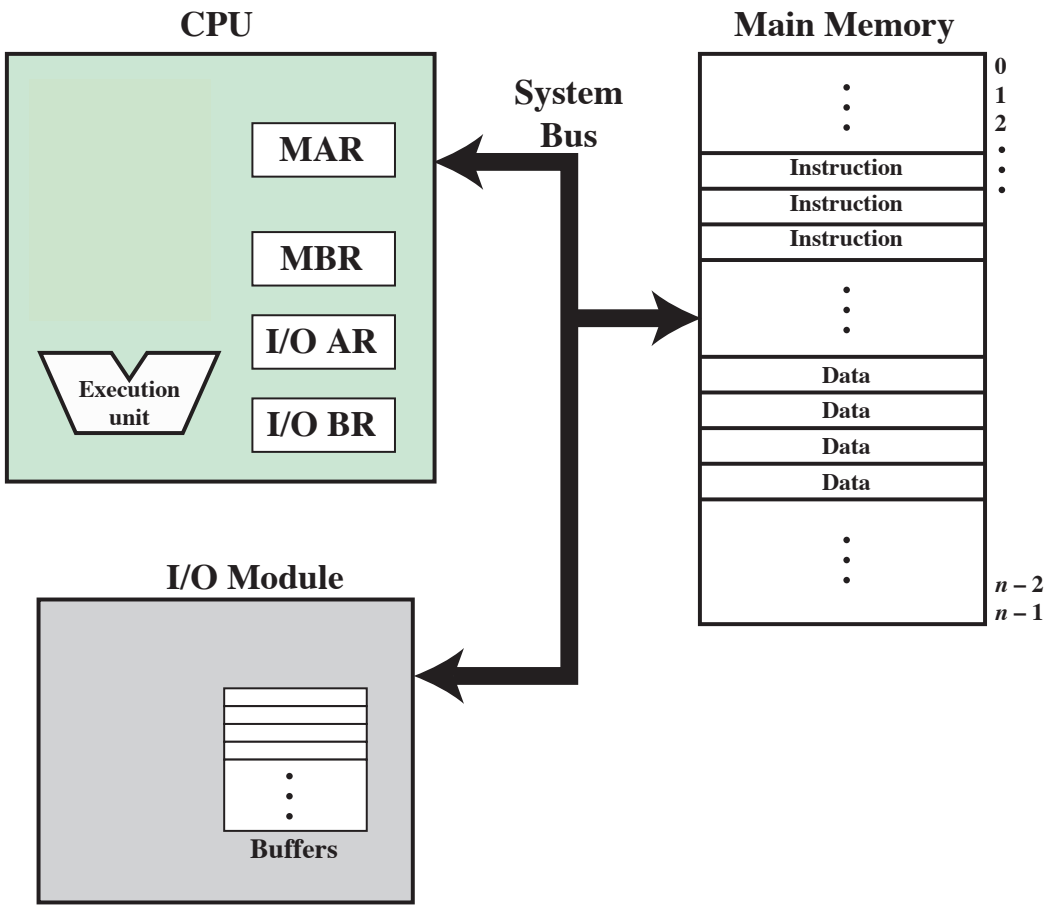
CPU e memoria



La CPU legge o scrive dati/istruzioni in memoria tramite due registri:

- **MAR**: memory address register
- **MBR**: memory buffer register

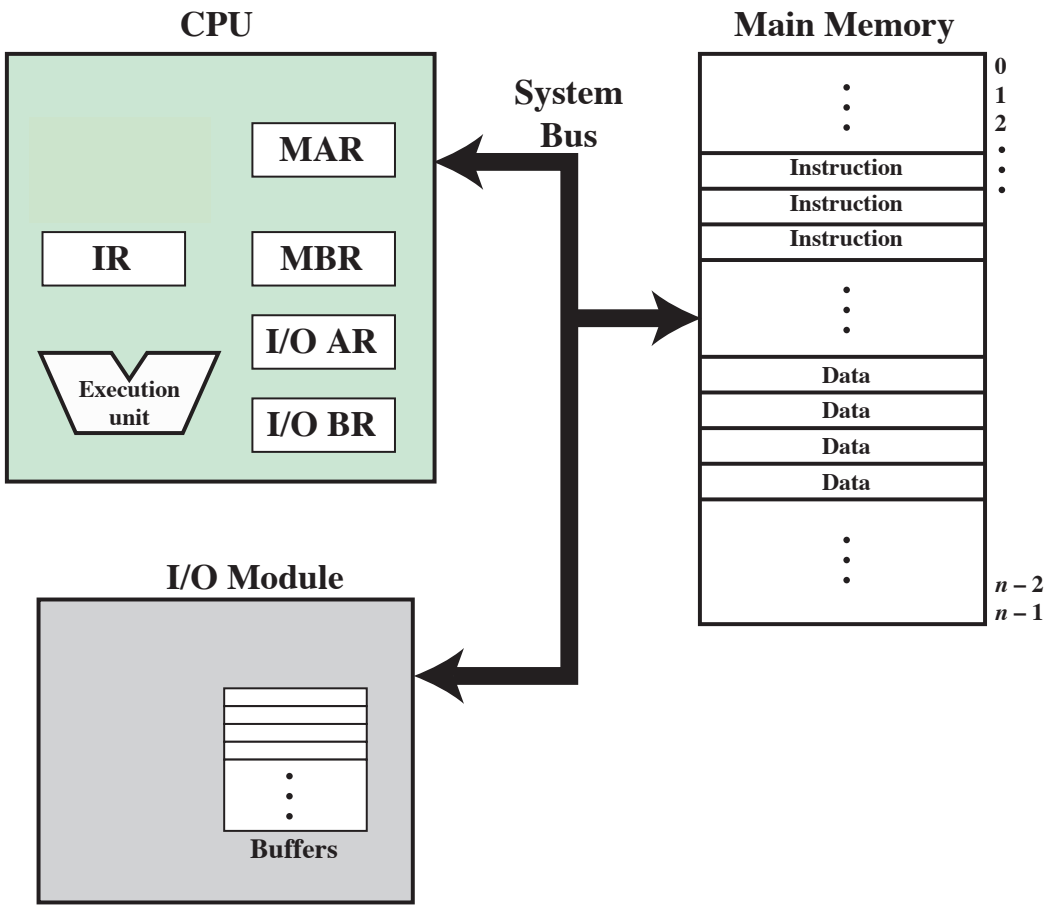
CPU e I/O



La CPU legge o scrive dati dall'I/O tramite due registri:

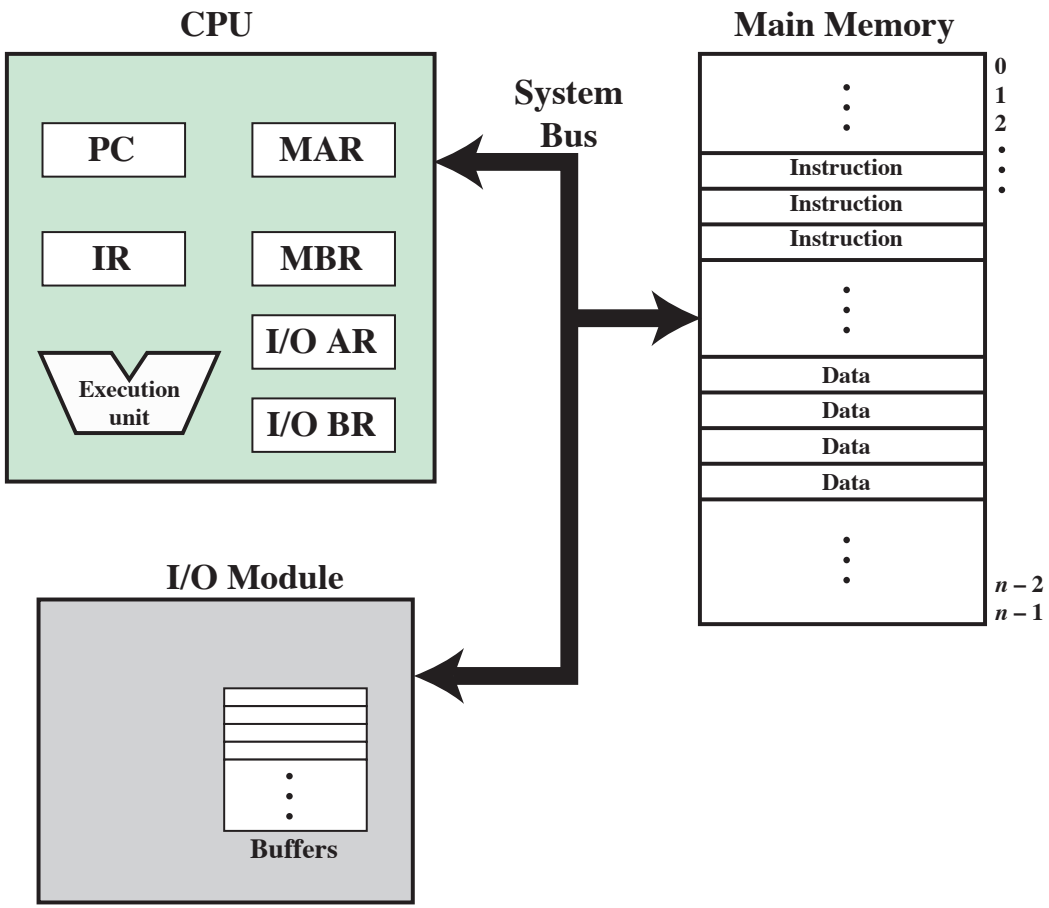
- **I/O AR**: I/O address register
- **I/O BR**: I/O buffer register

CPU e istruzioni



La CPU mantiene una copia locale dell'istruzione da eseguire nel registro **IR** (Instruction Register)

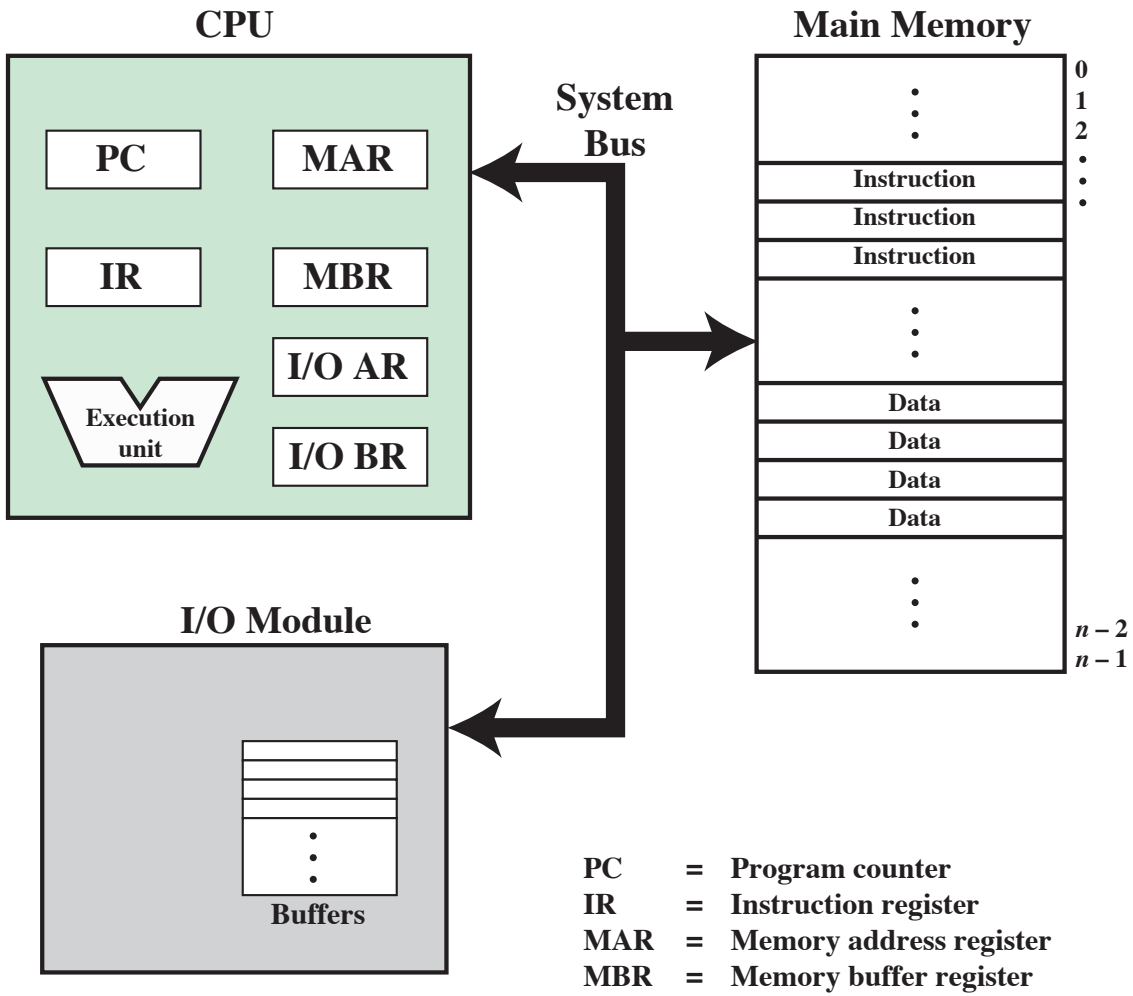
CPU e istruzioni (2)



La CPU deve conoscere quale istruzione eseguire.

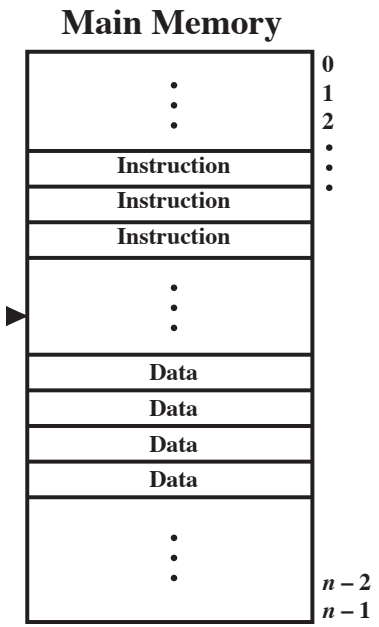
Il registro **PC** (program counter) mantiene l'indirizzo in memoria della **prossima** istruzione da eseguire

Architettura di von Neumann



Architettura di von Neumann (2)

- Dati e istruzioni condividono la stessa memoria
  - Concetto di **stored-program**, introdotto da von Neumann
  - Precedentemente il programma era salvato in hardware o su altri dispositivi
- Le locazioni di memoria possono contenere dati o istruzioni
- Un programma in memoria può essere modificato

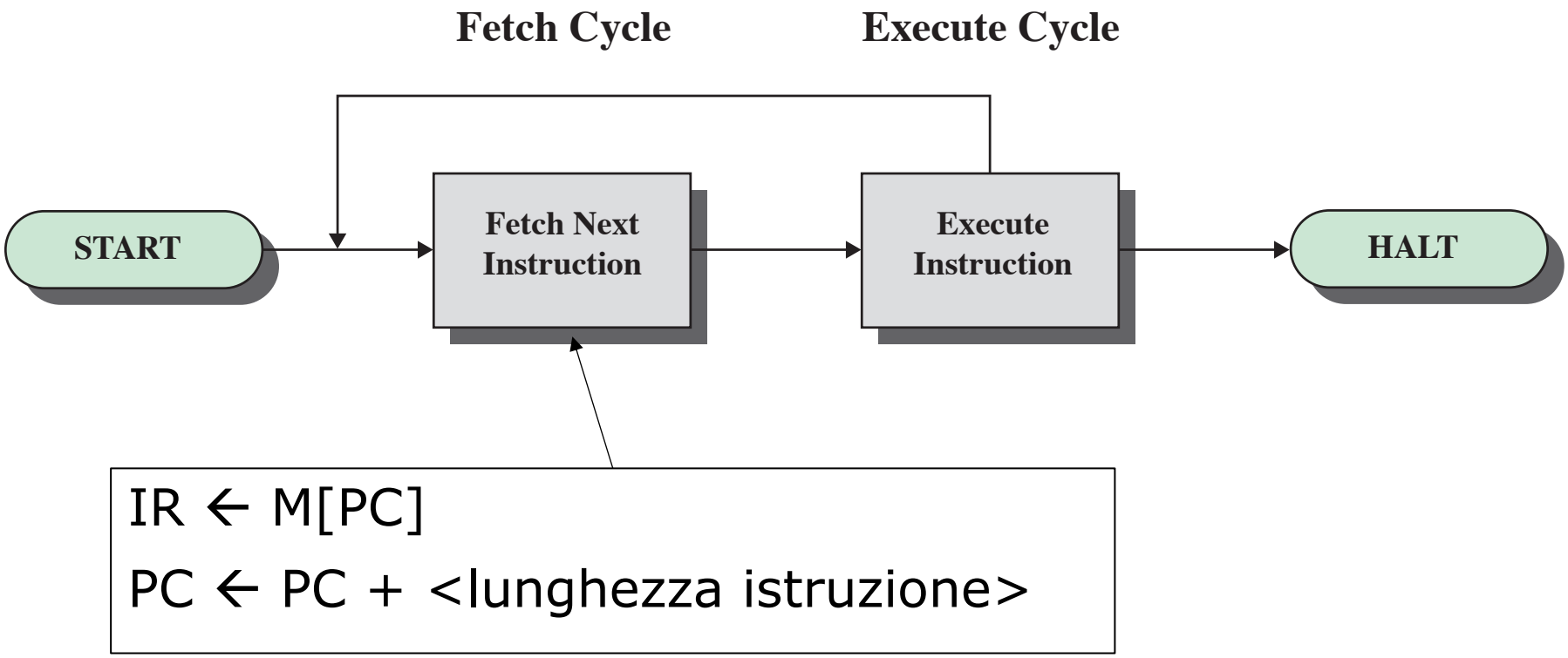


Ciclo di un'istruzione

Esecuzione di un programma

1. Viene recupera dalla memoria la prossima istruzione da eseguire (**fetch**)
   - Il registro PC contiene l'indirizzo in memoria della prossima istruzione
2. L'istruzione viene salvata nel registro IR
3. Il registro PC viene incrementato per puntare alla prossima istruzione
4. L'istruzione nel registro IR viene eseguita
5. Ritorna al punto 1

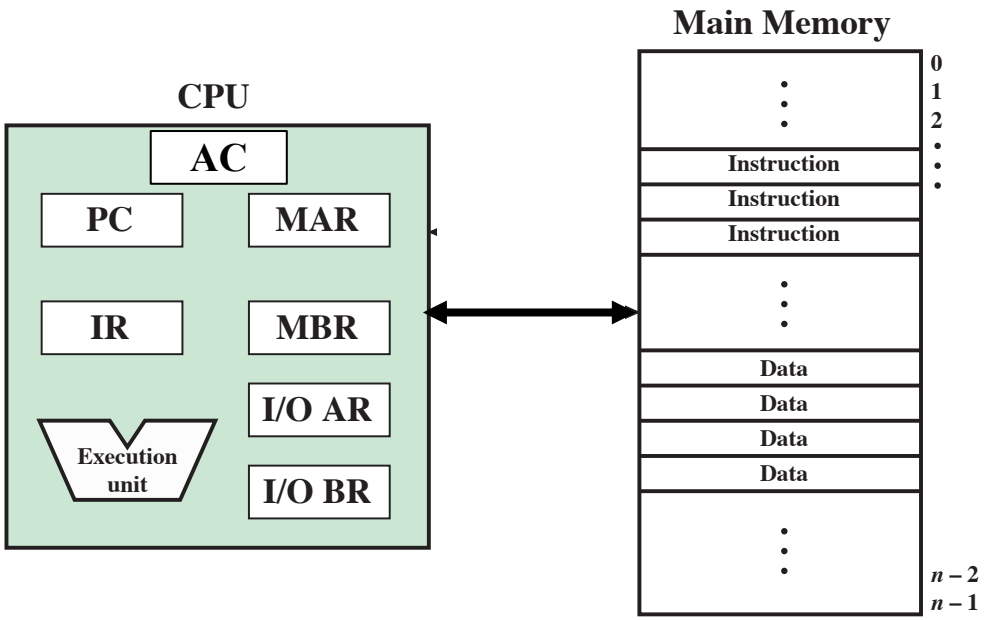
Ciclo di un'istruzione: fetch and execute



Comportamento di un'istruzione

- Processore-memoria
  - Spostamento di un dato dal processore alla memoria (o viceversa)
- Processore-I/O
  - Spostamento di un dato dal processore ad un modulo I/O
- Elaborazione dati
  - Esecuzione di operazioni logiche o aritmetiche
- Controllo
  - Modifica della sequenza delle istruzioni (e.g., modifica di PC)

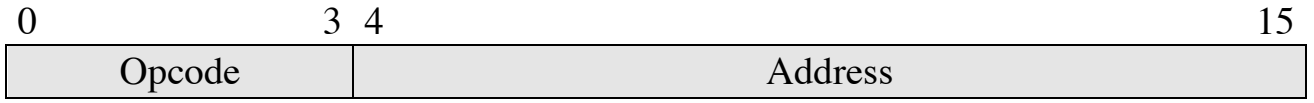
Esempio di esecuzione



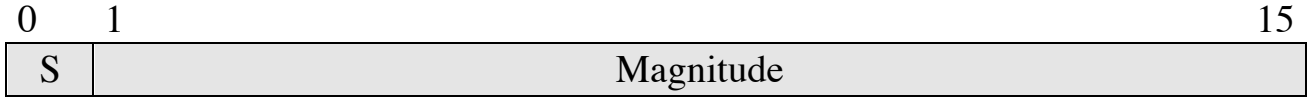
- Aggiungiamo un registro AC (accumulatore) per salvare dei dati temporanei
- Ignoriamo il modulo di I/O

Istruzioni

- Istruzioni/dati di 16 bit



(a) Instruction format

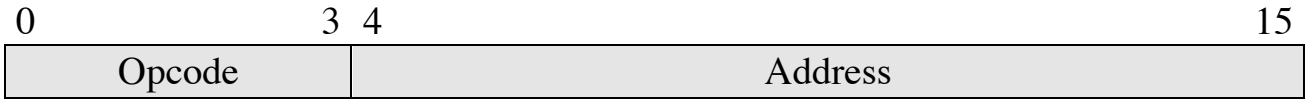


(b) Integer format

- Istruzione:
  - Opcode (4 bit): determina il tipo di operazione (load, store, add)
  - Address (12 bit): indirizzo in memoria

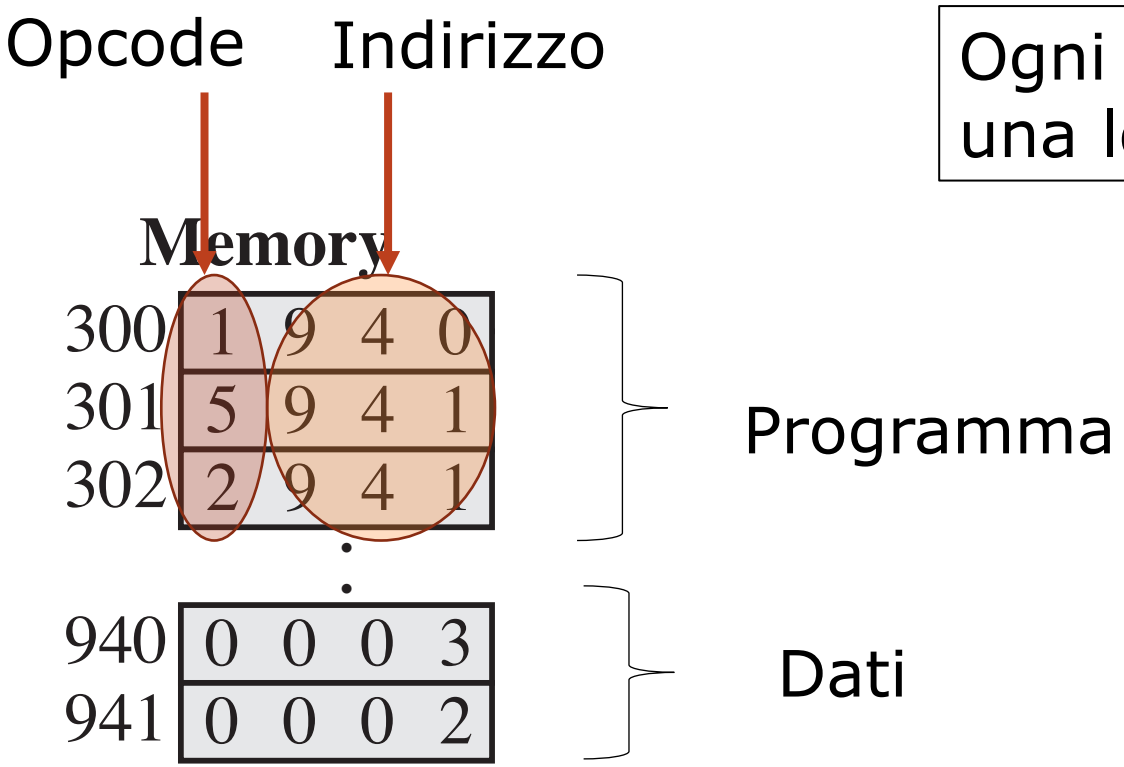
Istruzioni (2)

- **Load**: carica nel registro AC il dato contenuto in memoria all'indirizzo indicato nel campo "Address"
  - OPCODE: 0x1
- **Store**: scrivi il contenuto del registro AC in memoria all'indirizzo indicato nel campo "Address"
  - OPCODE: 0x2
- **Add**: somma il contenuto del registro AC con il dato contenuto in memoria all'indirizzo indicato nel campo "Address"
  - OPCODE: 0x5



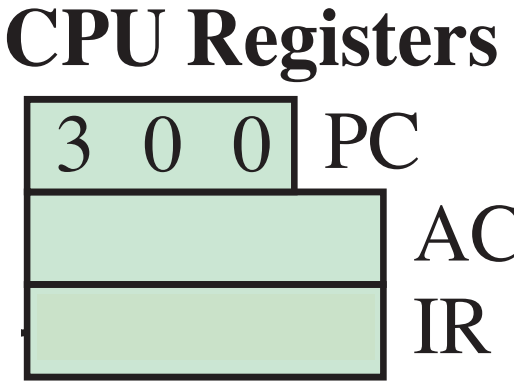
(a) Instruction format

Contenuto della memoria

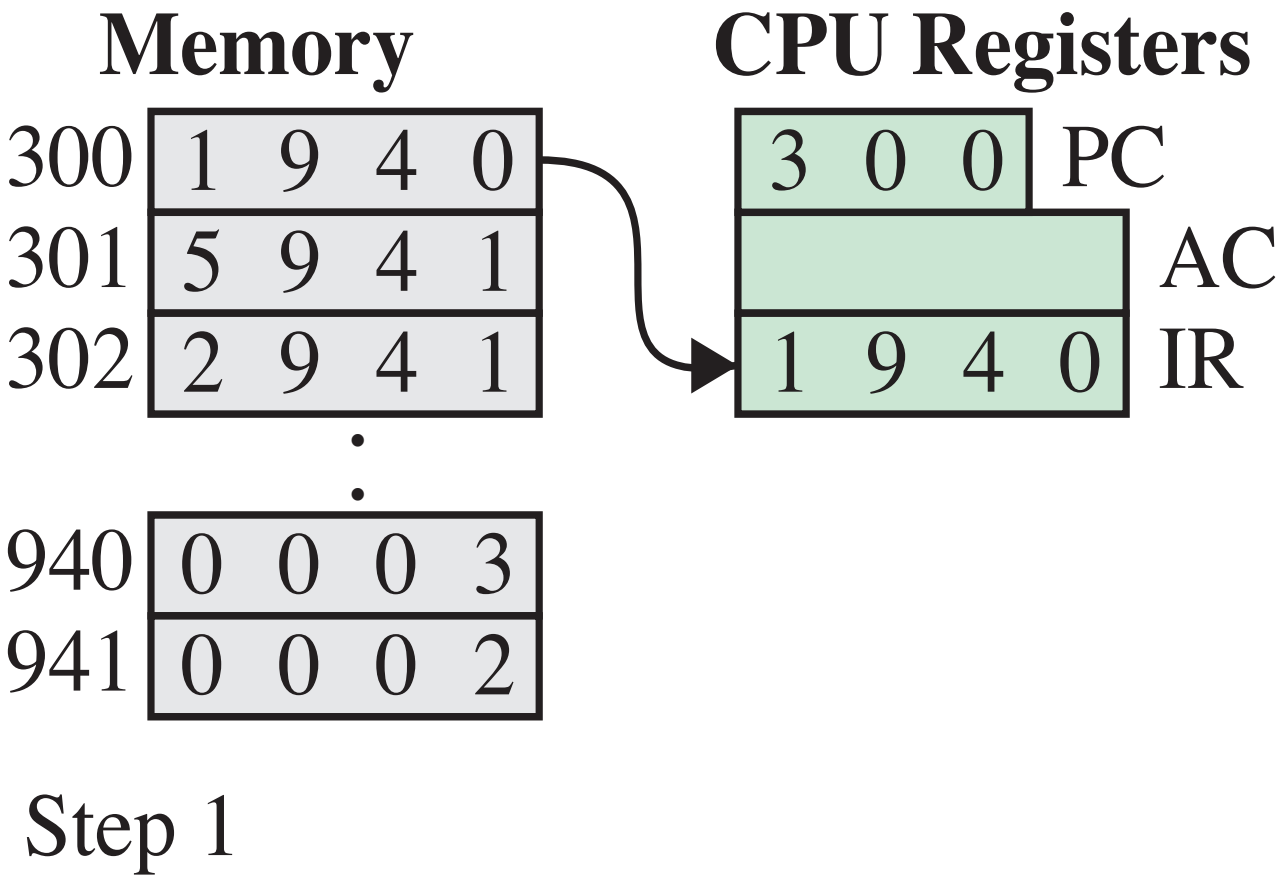


Ogni indirizzo punta ad una locazione di 16 bit

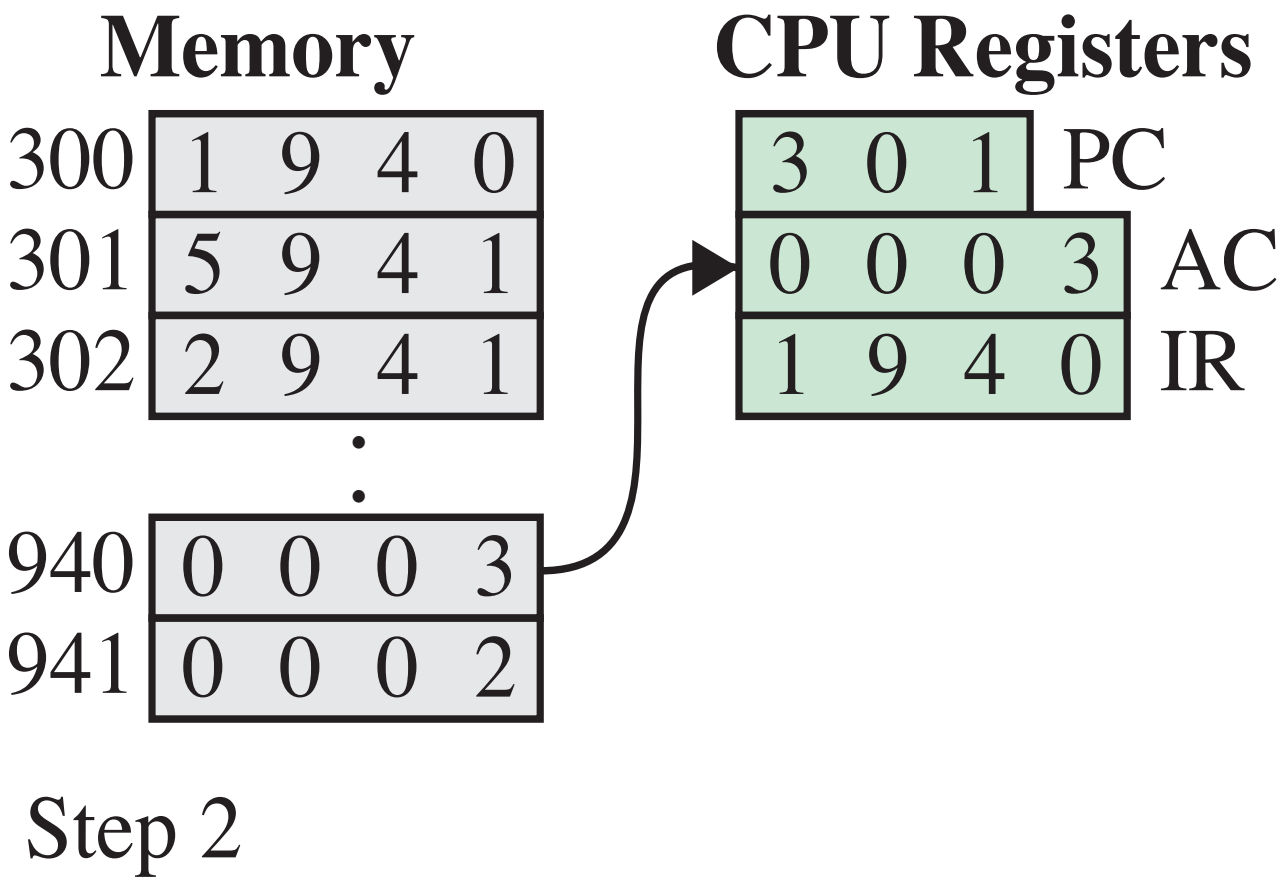
Contenuto dei registri



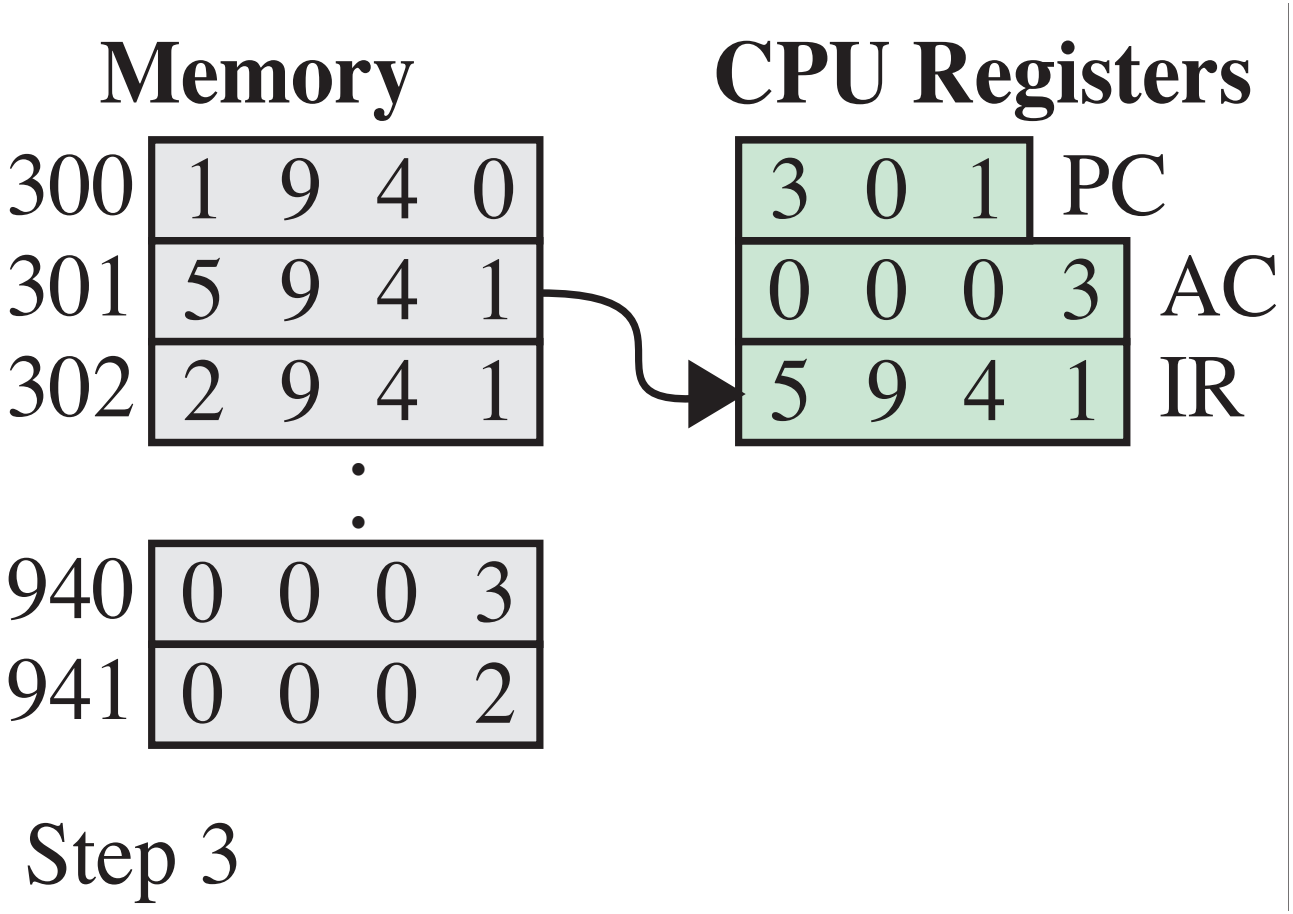
Esecuzione: step 1



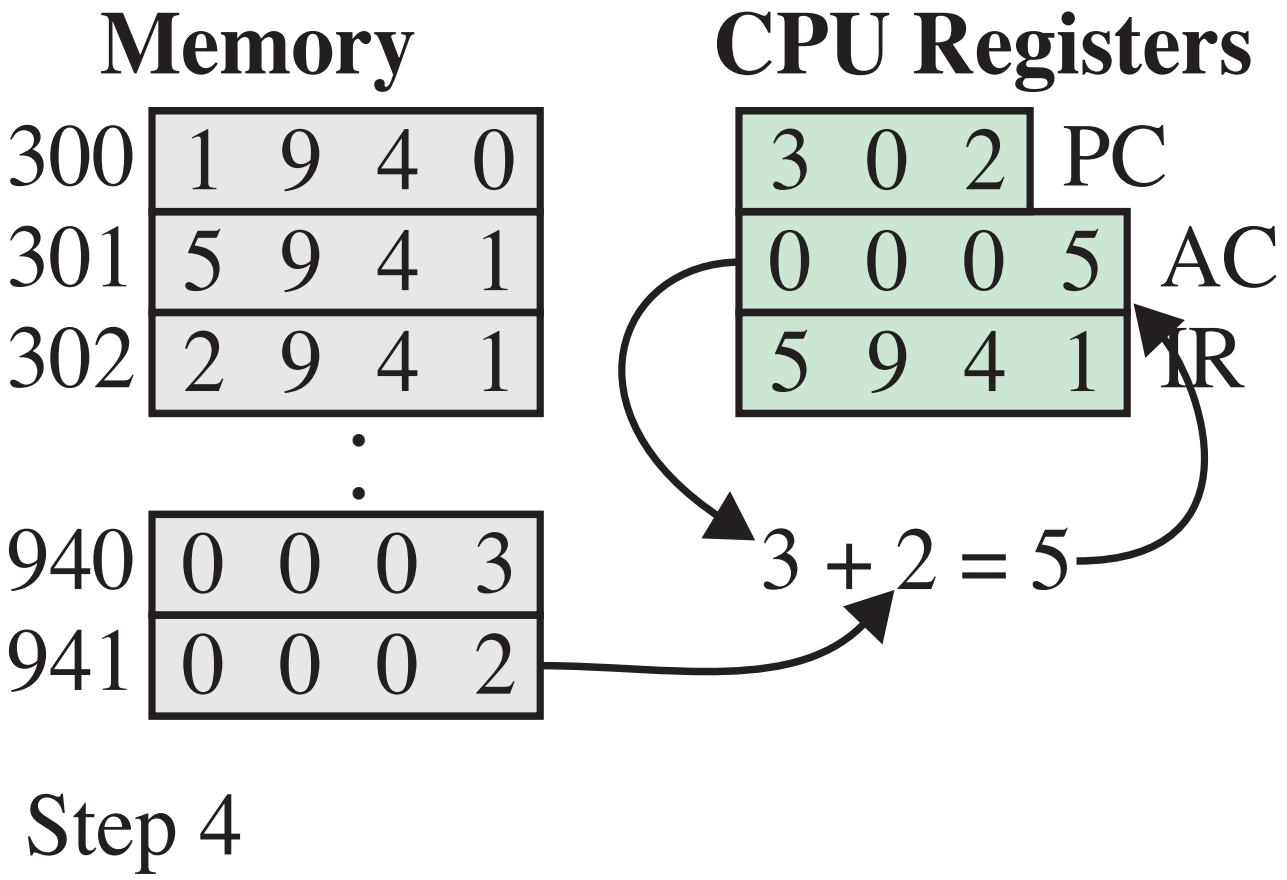
Esecuzione: step 2



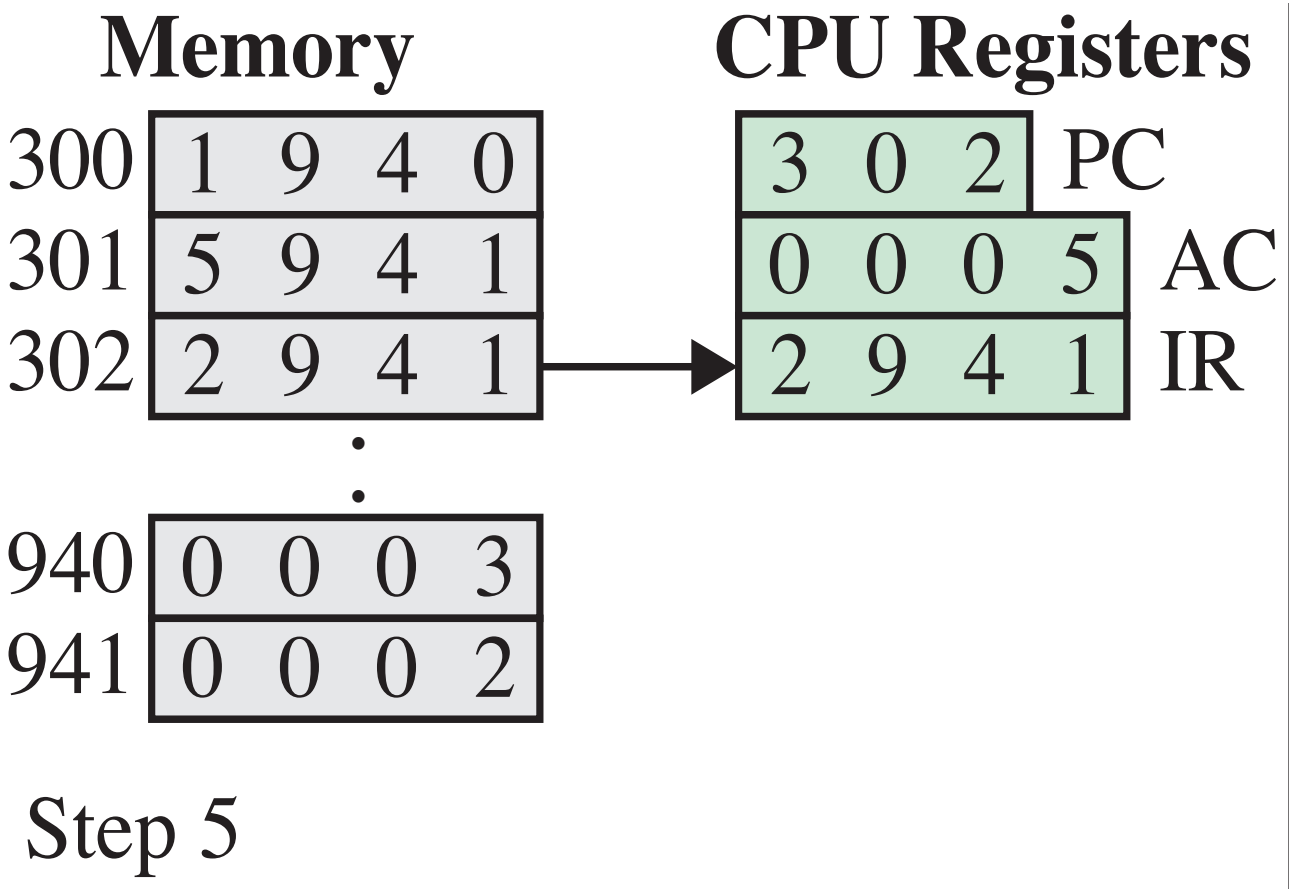
Esecuzione: step 3



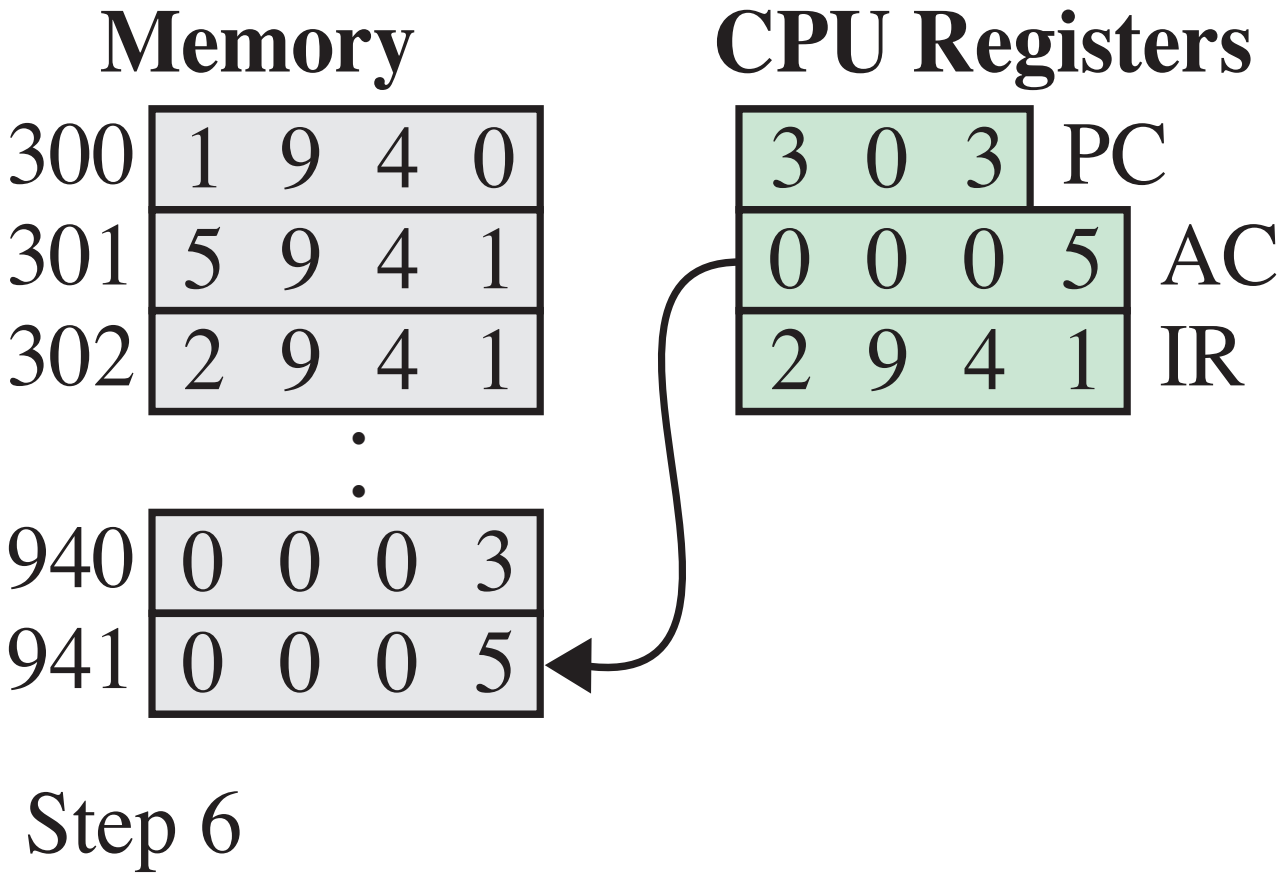
Esecuzione: step 4



Esecuzione: step 5



Esecuzione: step 6

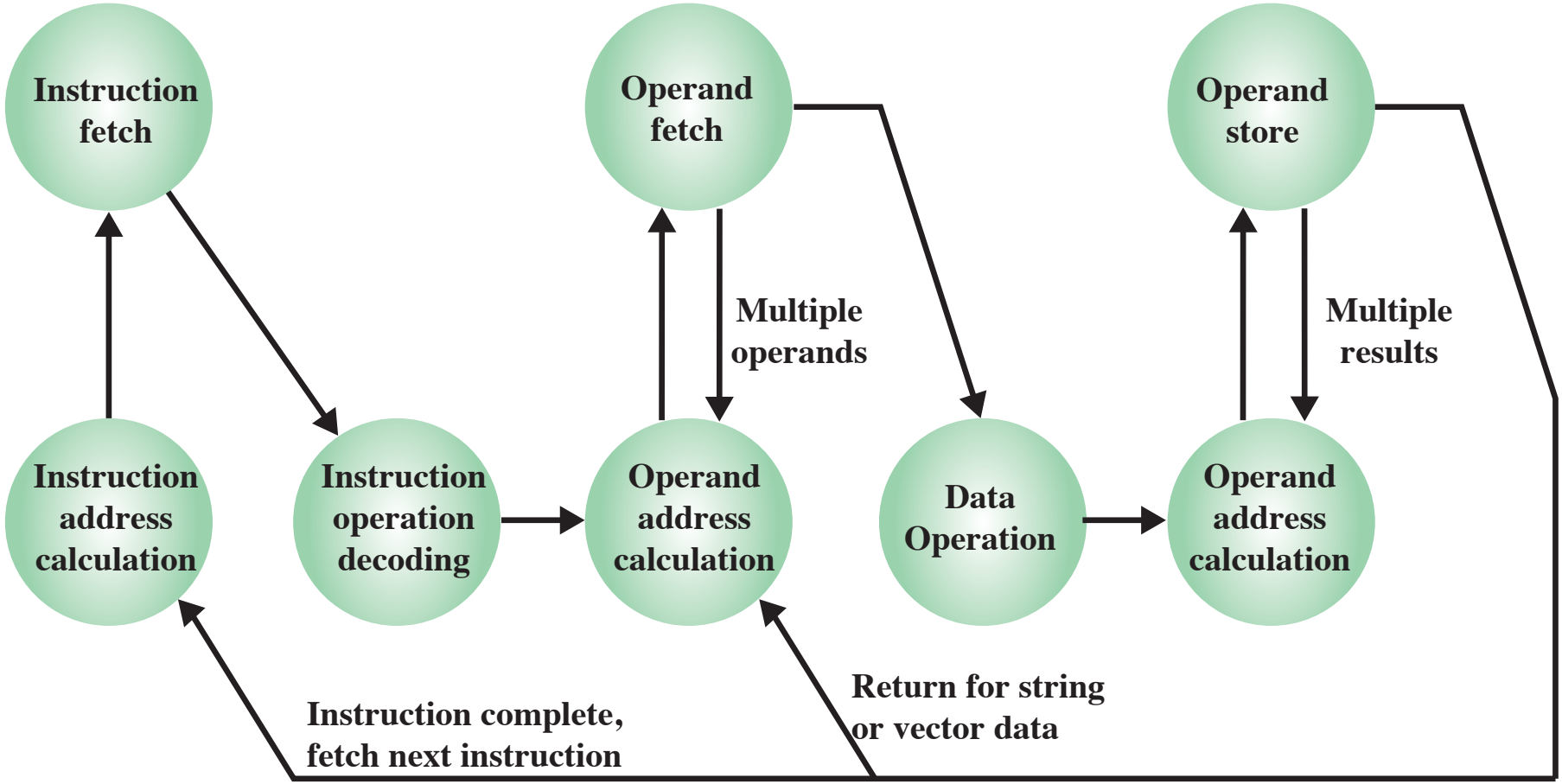


Ciclo completo di un'istruzione

- Il ciclo semplificato fetch-execute può essere scomposto in più fasi.

- Nell'esempio precedente possiamo notare più fasi:
  - Recupero dell'istruzione
  - Aggiornamento del PC
  - Lettura dell'input
  - Esecuzione dell'istruzione
  - Scrittura dell'output
  - …

Ciclo completo di un'istruzione



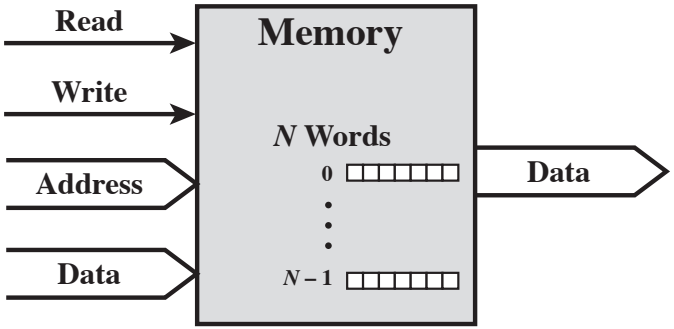
Interconnessioni di un elaboratore

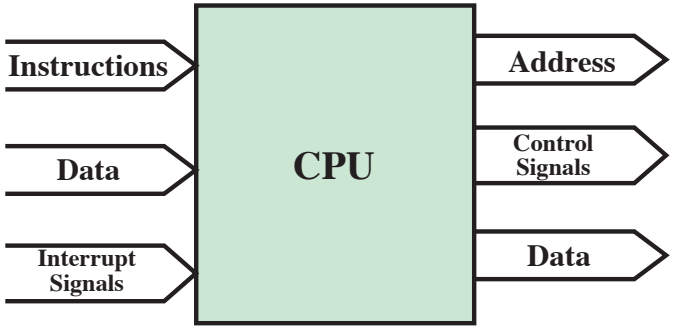
Interconnessione delle componenti

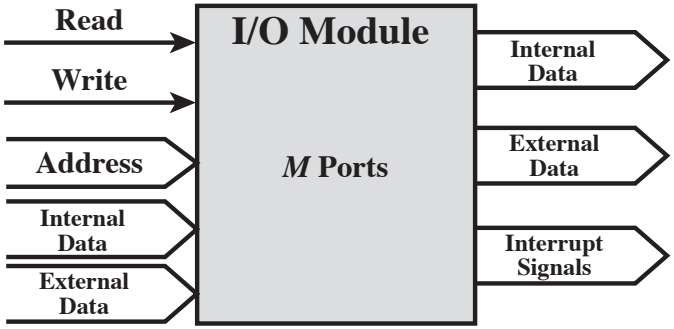
Le componenti devono essere connesse per supportare lo spostamento di dati/istruzioni

- Da memoria a processore / da processore a memoria
- Da modulo I/O a processore / da processore a modulo I/O
- Da modulo I/O a memoria / da memoria a modulo I/O

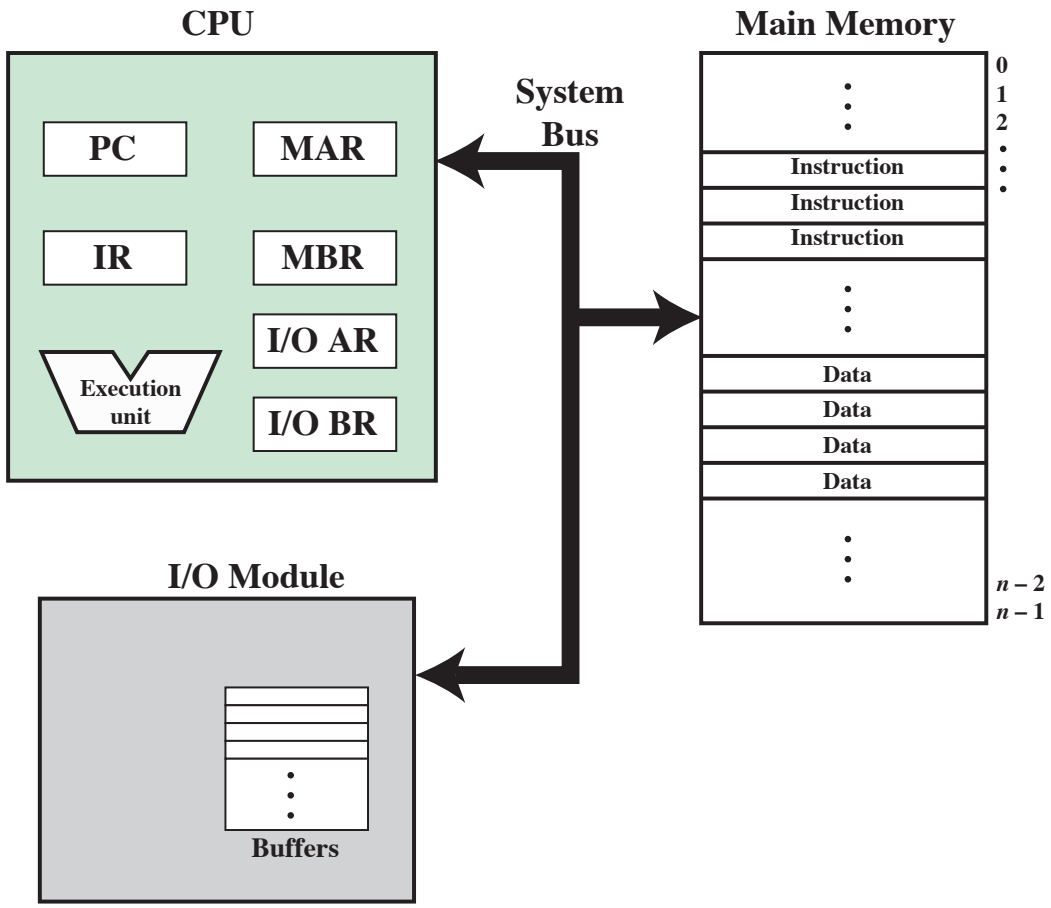
Interfaccia delle componenti







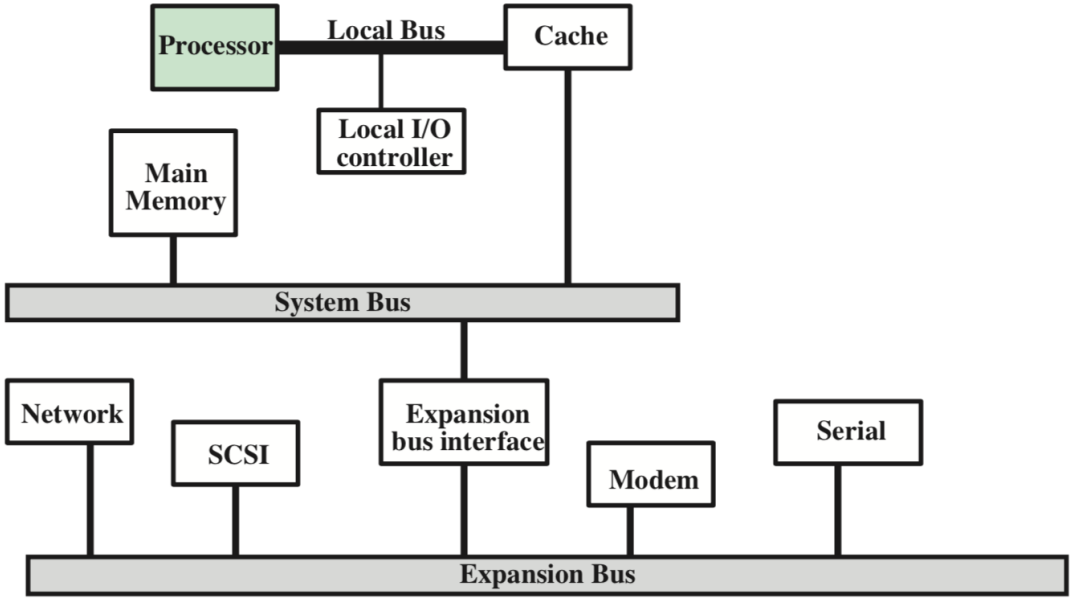
Interconnessione con bus



- **Bus**: mezzo di trasmissione condiviso che collega due o più componenti
- Ad ogni istante, solo un componente può inviare segnali in un bus
- I segnali inviati da un componente sono disponibili a tutte le componenti collegate al bus

Interconnessione con bus (2)

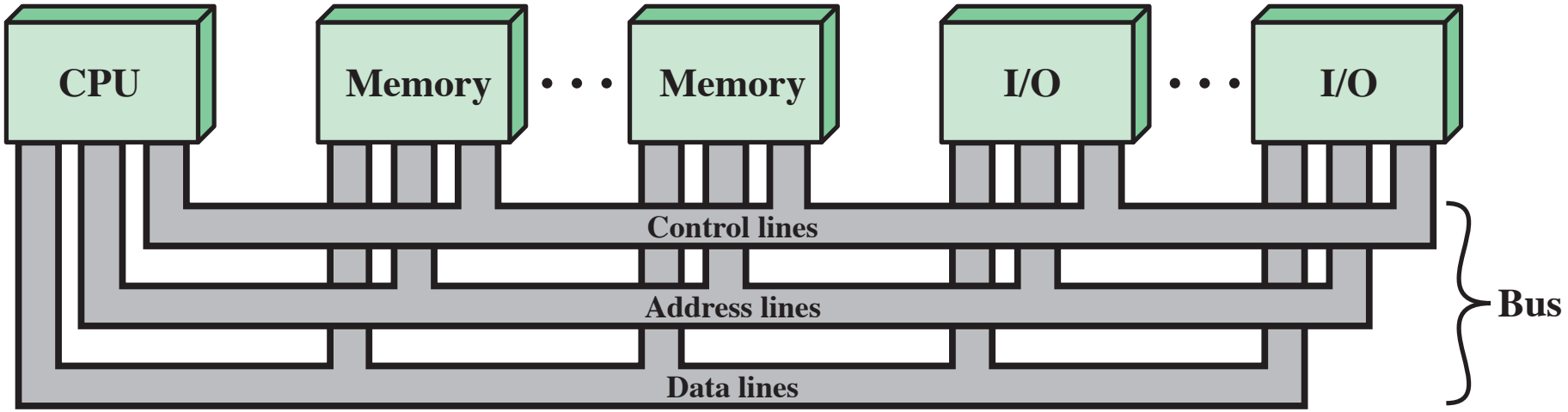
- Un elaboratore contiene più bus e più interconnessioni sono possibili
- **System bus**: bus che collega le principali componenti (CPU, memoria, dispositivi di I/O)



Tipi di bus

- Un bus consiste di più linee di comunicazione
  - Ogni linea trasmette un bit di informazione: il valore 0 o il valore 1
- Le linee di ogni bus possono essere classificate in tre gruppi:
  - Linee usate per trasmettere dati → **data bus**
  - Linee usate per trasmettere indirizzi → **address bus**
  - Linee usate per trasmettere segnali di controllo → **control bus**

Linee dati, indirizzo e controllo



Data Bus

- Il data bus viene usato per trasmettere dati tra componenti: ad esempio il valore contenuto in una certa locazione della memoria
- Il **numero di linee** (bus width) indica il numero di bit che possono essere trasmessi contemporaneamente
- Il data bus può consistere di 32, 64, 128 o più linee
- Per trasmettere n bits bisogna inviare w bit per n/w volte

Address Bus

- Utilizzato per indicare la sorgente/destinazione del dato inviato tramite il data bus
  - Per leggere un dato in memoria, il processore deve comunicare alla memoria l'indirizzo del dato tramite l'address bus.
  - Il dato verrà restituito dalla memoria tramite il data bus
- Il numero di linee indica il massimo numero di possibili posizioni (indirizzi):
  - Con w linee si possono indicare 2^w indirizzi
- L'address bus può essere utilizzato per accedere a moduli di I/O
  - Esempio: I bit più significativi indicano l'accesso alla memoria o ad un particolare modulo di I/O; i restati bit indicano una locazione di memoria o un registro all'interno del modulo

Control Bus

- Usato per coordinare l'accesso al bus dati/indirizzi tra i vari dispositivi collegati al bus
- Usato per inviare segnali di controllo tra dispositivi:
  - Ad esempio per spostare un dato dalla memoria al processore
- Usato per inviare segnali di temporizzazione
  - Ad esempio per indicare quando un segnale nel bus dati è valido

Esercizi

Esercizio 1

Supponiamo di aggiungere le seguenti istruzioni all'elaboratore semplificato visto precedentemente:

- Leggi un dato da un modulo di I/O e scrivilo nel registro AC (opcode: 0x3; 12 bit per indicare il modulo di I/O)
- Scrivi il dato contenuto nel registro AC in un modulo di I/O (opcode: 0x7; ; 12 bit per indicare il modulo di I/O)

Scrivere un programma che: 1) legge un dato dal dispositivo 4; 2) lo somma con il dato in memoria all'indirizzo 0x904; 3) scrive il risultato in memoria all'indirizzo 0x905.

Indicare il contenuto della memoria, assumendo che il programma sia posizionato a partire dall'indirizzo 0x220.

Esercizio 2

Si consideri un bus organizzato in un bus dati di 32 bit e in un bus indirizzi di 16 bit.

1. Quante locazioni in memoria possiamo indicizzare?
2. Quant'è la dimensione (in byte) massima di una memoria supportata dal bus, assumendo che una locazione di memoria contenga 1 byte
3. Quant'è la dimensione (in byte) massima di una memoria supportata dal bus, assumendo che una locazione di memoria contenga 4 byte
4. Se il bus dati trasmette 32 bit ogni 100 ns, quanto tempo è richiesto per trasmettere 4MB?