Codifica dei Numeri

Informatica ICA (LC)

30 Novembre 2017

Giacomo Boracchi

giacomo.boracchi@polimi.it

Rappresentazione dei Numeri Positivi

Codifica binaria e codifiche collegate

Codifica dei Numeri in Base 10

- Le cifre che abbiamo a disposizione sono 10
$$A_{10} = \{0, 1, \dots, 9\}$$
- Utilizziamo una **codifica posizionale**, quindi le cifre in posizioni differenti hanno un significato differente
- Es numero di 4 cifre
  - $\mathbf{3401} = \mathbf{3} \times 10^3 + \mathbf{4} \times 10^2 + \mathbf{0} \times 10^1 + \mathbf{1} \times 10^0$
- Con m cifre posso rappresentare 10^m numeri distinti:
$$0, \dots, 10^m - 1$$

Codifica dei Numeri in una Base Qualsiasi

- Ogni codifica ha un insieme di cifre (dizionario) A
  - In base 10, il dizionario è $A_{10} = \{0, \dots, 9\}$
- Un numero è una sequenza di cifre

  $$a_n a_{n-1} \dots a_1 a_0 \text{ con } a_i \in A$$

  - 8522 è una sequenza di 4 cifre di A_{10}, $\{8,5,2\} \subset A_{10}$.
- Manteniamo un **codifica posizionale**: ogni cifra assume un significato diverso in base alla sua posizione nel numero.
  - a_n è la cifra **più significativa**
  - a_0 è la cifra **meno significativa**

  Es: in **8522**, **8** è la cifra più significativa, **2** quella meno.

  8522 è diverso da 2852, 8252,… che pur contengono le stesse cifre

Codifica dei Numeri: Notazione Posizionale

- Dato un numero N_{10}, in base 10 contenente m cifre scritto come $a_{m-1}a_{m-2}\dots a_1a_0$ questo corrisponde a:

$$N_{10} = a_{m-1} \times 10^{m-1} + a_{m-2} \times 10^{m-2} + \dots + a_0 \times 10^0$$

$$(a_{m-1}a_{m-2}\dots a_1a_0)_{10} = \sum_{i=0}^{m-1} a_i \times 10^i \ , \qquad a_i \in A_{10}$$

Es: $(8522)_{10} = 8 \times 10^3 + 5 \times 10^2 + 2 \times 10^1 + 2 \times 10^0$

- Con m cifre in A_{10} quanti numeri posso esprimere: 10^m
- Considerando gli interi positivi, posso scrivere tutti numeri tra $[0, 10^m - 1]$
  - *Es:* $m = 1$ copro $[0,10 - 1]$ (cioè $[0,9]$ i.e., A_{10})

    $m = 3$ copro $[0,10^3 - 1]$ (cioè $[0, 999]$)

Rappresentazioni Posizionali in Base p

- Consideriamo **rappresentazioni posizionali in base p** (con $p > 0$) e chiamiamo A_p il dizionario di p cifre:
  - se $p \leq 10$ prendiamo le cifre di A_{10}, $A_p = \{0, \ldots, p-1\}$
  - se $p > 10$ aggiungiamo simboli $A_p = \{0, .., 9, A, B..\}$
- Un numero di m cifre in base p:

$$N_p = a_{m-1} \times p^{m-1} + a_{m-2} \times p^{m-2} + .. + a_0 \times p^0$$

$$N_p = a_{m-1}a_{m-2} \ldots a_1 a_0 = \sum_{i=0}^{m-1} a_i \times p^i, \qquad a_i \in A_p$$

- Con m cifre in A_p quanti numeri posso esprimere: p^m
- Considerando gli interi positivi, posso scrivere tutti numeri tra $[0, p^m - 1]$

Codifica dei numeri in base p: Esempi

- *Es:* $m = 1$ e $p = 7$, copro $[0, 7 - 1]$ (cioè $[0,6]$)

  $m = 4$ e $p = 7$, copro $[0, 7^4 - 1]$ (cioè $[0,2400]$)

  $m = 1$ e $p = 13$, copro $[0, 13 - 1]$ (cioè $[0,12]$)

  $m = 4$ e $p = 13$, copro $[0, 13^4 - 1]$ (cioè $[0,28560]$)

  Al crescere di p cresce il «potere espressivo» del dizionario (con lo stesso numero di cifre posso scrivere molti più numeri).

Codifica dei Numeri in Base 2

- I calcolatori sono in grado di operare con informazioni **binarie**. Quindi $p = 2$ e $A_2 = \{0, 1\}$

$$N_2 = a_{m-1} \times 2^{m-1} + a_{m-2} \times 2^{m-2} + .. + a_0 \times 2^0$$

$$N_2 = a_{m-1}a_{m-2} \dots a_1 a_0 = \sum_{i=0}^{m-1} a_i \times 2^i \ , \qquad a_i \in \{0,1\}$$

- Un bit (*binary digit*) assume valore 0/1 corrispondente ad un determinato *stato fisico* (alta o bassa tensione nella cella di memoria)
- Con m bit posso scrivere 2^m numeri diversi, ad esempio tutti gli interi nell'intervallo $[0, 2^m - 1]$
- Il byte è una sequenza di 8 bit ed esprime $2^8 = 256$ numeri diversi (ad esempio gli interi in $[0,255]$)

00000000, 00000001, 00000010, …, 11111111

Conversione Binario-Decimale

- È necessario imparare le potenze di 2!

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
1	2	4	8	16	32	64	128	256	512	1024

- E i loro legami con l'informatica:
  - Byte = 8 bit
  - KiloByte (kB) = 10^3 Byte
  - MegaByte (MB) = 10^6 Byte
  - GigaByte (GB) = 10^9 Byte
  - TheraByte (TB) = 10^{12} Byte

Altre Codifiche che consideriamo

- Codifica **ottale** (in **base 8**)
  - $A_8 = \{0, 1, \dots, 7\}$
  - con m cifre in A_8 scrivo i numeri da $[0, 8^m - 1]$

- Codifica **esadecimale**, (in **base 16**)
  - $A_{16} = \{0, 1, \dots, 9, A, B, C, D, E, F\}$,
  - Per le conversioni $A = 10$, …, $F = 15$.
  - con m cifre in A_{16} scrivo i numeri da $[0,16^m - 1]$.

Conversione Binario-Decimale

- Utilizziamo la definizione di numero in notazione posizionale

$$N_2 = a_{m-1} \times 2^{m-1} + a_{m-2} \times 2^{m-2} + .. + a_0 \times 2^0$$

Es.

$$(101)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (5)_{10}$$

$$(1100010)_2 = 1 \times 2^6 + 1 \times 2^5 + 1 \times 2 = (98)_{10}$$

Osservazioni

- In binario i numeri che terminano con **1** sono dispari, quelli con **0** sono pari.
  - L'unico modo per avere un numero dispari nella somma è aggiungere $2^0 = 1$
- Le conversioni di numeri con bit tutti a **1** si calcolano facilmente

$$(111111)_2 = (1000000)_2 - (1)_2 = 2^6 - 1 =$$

Conversione Decimale ➡ Binario

- Metodo delle divisioni successive:
- Per convertire 531 opero come segue:
  - 531 / 2 = 265 + 1
  - 265 / 2 = 132 + 1
  - 132 / 2 = 66 + 0
  - 66 / 2 = 33 + 0
  - 33 / 2 = 16 + 1
  - 16 / 2 = 8 + 0
  - 8 / 2 = 4 + 0
  - 4 / 2 = 2 + 0
  - 2 / 2 = 1 + 0
  - 1 / 2 = 0 + 1

Divisione intera tra il numero e 2

Il risultato della divisione precedente viene successivamente diviso

Si continua fino a quando il risultato della divisione non diventa o (e considero comunque il resto!)

Conversione Decimale ➡ Binario

- Metodo delle divisioni successive:
- Per convertire 531 opero come segue:
  - 531 / 2 = 265 + 1
  - 265 / 2 = 132 + 1
  - 132 / 2 = 66 + 0
  - 66 / 2 = 33 + 0
  - 33 / 2 = 16 + 1
  - 16 / 2 = 8 + 0
  - 8 / 2 = 4 + 0
  - 4 / 2 = 2 + 0
  - 2 / 2 = 1 + 0
  - 1 / 2 = 0 + 1

⇨ Cifra meno significativa

I resti della divisione intera, letti dall'ultimo al primo, identificano il numero binario

$$(531)_{10} = (1000010011)_2$$

⇨ Cifra più significativa

TODO: Conversione Decimale ➡ Binario

- Scrivere una funzione Matlab che esegue la conversione decimale binario e salva il risultato in un'opportuna struttura dati prima di visualizzarlo
- Modificare il programma convertire da decimale ad una base p qualunque con $p \leq 16$ specificata dall'utente durante l'esecuzione del programma.
- **NB:** l'algoritmo delle divisioni successive vale rispetto a qualunque base

Conversioni ottale/esadecimale ➡ decimale

- È possibile utilizzare le definizioni precedenti per convertire da ottale/esadecimale in base 10

$$N_p = a_{m-1}a_{m-2}\dots a_1a_0 = \sum_{i=0}^{m-1} a_i \times 16^i \ , \qquad a_i \in A_{16}$$

- *Es:* $(31)_8 =$
- $(A170)_{16} =$

  $=$
- $(623)_8 =$

  $=$
- $(623)_{16} =$

  $=$

Conversioni decimale ➡ ottale/esadecimale

- È possibile utilizzare l'algoritmo delle divisioni successive

- È tuttavia più comodo fare delle conversioni passando dalla rappresentazione binaria e
  - Esprimere ogni sequenza di 3 numeri binari in base 8
  - $(1231)_{10} = (10011001111)_2 = (\underbrace{010}\ \underbrace{011}\ \underbrace{001}\ \underbrace{111})_2$
  - $(1231)_{10} = (2317)_8$ $\qquad (2 \quad 3 \quad 1 \quad 7)_8$

  - Esprimere ogni sequenza di 4 numeri binari in base 16
  - $(1231)_{10} = (10011001111)_2 = (\underbrace{0100}\ \underbrace{1100}\ \underbrace{1111})_2$
  - $(1231)_{10} = (4CF)_{16}$ $\qquad (4 \quad C \quad F)_{16}$

Somma tra Numeri Binari

- Si eseguono «in colonna» e si opera cifra per cifra
- Si considera il riporto come per i decimali
  - 0 + 0 = 0 riporto 0
  - 1 + 0 = 1 riporto 0
  - 0 + 1 = 1 riporto 0
  - 1 + 1 = 0 riporto 1
- Occorre sommare il riporto della cifra precedente

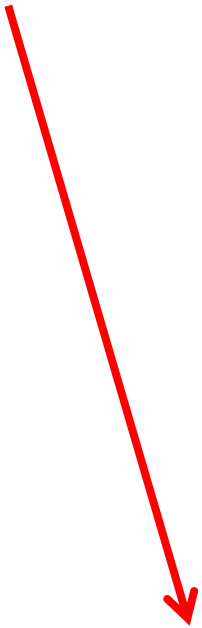
$$\begin{array}{r} \mathbf{1} \\ \mathbf{0101} + (5)_{10} \\ \mathbf{1001} = (9)_{10} \\ \hline \mathbf{1110} \quad (14)_{10} \end{array}$$

$$\begin{array}{r} \mathbf{111} \\ \mathbf{1111} + (15)_{10} \\ \mathbf{1010} = (10)_{10} \\ \hline (\mathbf{1})\mathbf{1001} \quad (25)_{10} \end{array}$$

Riporto

Somma tra numeri binari

- A volte i bit utilizzati per codificare gli addendi non bastano a contenere il risultato
  - In questi casi occorrono più bit per codificare il risultato
  - Si ha quindi un bit di **carry**



$$\begin{array}{r} \color{red}{1} \\ \mathbf{0101} + (5)_{10} \\ \mathbf{1001} = (9)_{10} \\ \hline \mathbf{1110} \quad (14)_{10} \end{array}$$

$$\begin{array}{r} \color{red}{111} \\ \mathbf{1111} + (15)_{10} \\ \mathbf{1010} = (10)_{10} \\ \hline \color{red}{(1)}\mathbf{1001} \quad (25)_{10} \end{array}$$

I numeri Interi

Positivi e Negativi

Rappresentazione Modulo e Segno

- È possibile dedicare **il primo bit** alla codifica del **segno**
  - `"1"` il numero che segue è negativo
  - `"0"` il numero che segue è positivo

- Con m cifre in binario e codifica modulo dedico 2^{m-1} per i positivi e 2^{m-1} per gli stessi cambiati di segno
  - posso rappresentare tutti i numeri nell'intervallo

$$X \in [-2^{m-1}+1, 2^{m-1}-1]$$

- Es
  - `01010 = + 10`
  - `11101 = - 13`
  - `-27 = 111011`
  - `122 = 01111010`

Rappresentazione Modulo e Segno

- Esempio $m = 3$
  - 0 = 000
  - 1 = 001
  - 2 = 010
  - 3 = 011
  - -0 = 100
  - -1 = 101
  - -2 = 110
  - -3 = 111

Ho due codifiche differenti lo zero

- C'è uno «spreco» nella codifica
- Ostacola realizzazione circuitale delle operazioni algebriche (non lo mostriamo)
- Occorre trovare una rappresentazione migliore!

Rappresentazione in Complemento a 2 (CP2)

- Date m cifre binarie, disponibili 2^m configurazioni distinte
- In CP2 se ne usano:
  - $2^{m-1} - 1$ per valori positivi
  - 1 per lo zero
  - 2^{m-1} per i valori negativi
- Con m bit rappresento l'intervallo $[-2^{m-1}, 2^{m-1} - 1]$

Rappresentazione in Complemento a 2 (CP2)

- Sia $X \in [-2^{m-1}, 2^{m-1}-1]$ il numero da rappresentare in CP2, con m bit.
  - se X è **positivo** o nullo **scrivo $\boldsymbol{X}$** in binario con **$\boldsymbol{m}$ bit**
  - se X è **negativo scrivo $\boldsymbol{2^m - |X|}$** in binario con **$\boldsymbol{m}$ bit**
- Questo equivale alla seguente codifica:

$$N_{CP2} = a_{m-1}a_{m-2} \dots a_1 a_0$$

$$= -a_{m-1} \times 2^{m-1} + a_{m-2} \times 2^{m-2} + \dots + a_0 \times 2^0$$

$$= -a_{m-1} \times 2^{m-1} + \sum_{i=0}^{m-2} a_i \times 2^i \ , \qquad a_i \in \{0,1\}$$

Rappresentazione in Complemento a 2 (CP2)

- Sia $X \in [-2^{m-1}, 2^{m-1}-1]$ il numero da rappresentare in CP2, con m bit.
  - se X è **positivo** o nullo **scrivo X** in binario con **m bit**
  - se X è **negativo scrivo $2^m - |X|$** in binario con **m bit**
- Questo equivale alla seguente codifica:

$$N_{CP2} = a_{m-1}a_{m-2}\ldots a_1 a_0$$

$$= \boxed{-a_{m-1}\times 2^{m-1}} + a_{m-2}\times 2^{m-2} + \ldots + a_0 \times 2^0$$

$$= \boxed{-a_{m-1}\times 2^{m-1}} + \sum_{i=0}^{m-2} a_i \times 2^i \ , \qquad a_i \in \{0,1\}$$

i.e., viene cambiato il segno dell'addendo relativo alla cifra più significativa

Rappresentazione in CP2

- Esempio $m = 3$
  - `-4 =`
  - `-3 =`
  - `-2 =`
  - `-1 =`
  - `0 =`
  - `1 =`
  - `2 =`
  - `3 =`

Rappresentazione in CP2

- Esempio $m = 3$
  - `-4 = 100`
  - `-3 = 101`
  - `-2 = 110`
  - `-1 = 111`
  - `0 = 000`
  - `1 = 001`
  - `2 = 010`
  - `3 = 011`

Rappresentazione in CP2

- Con i positivi copro solo il range $[0, 2^{m-1}-1]$, quindi la prima cifra è 0 (il numero è minore di 2^{m-1})
- Con i negativi copro il range $[-2^{m-1}, -1]$ e scrivo $2^m - |X|$, e quindi la prima cifra è 1 (il numero è maggiore di 2^{m-1})
- Quindi, il primo bit **indica il segno** del numero
  - Attenzione: questo numero **non è il segno**: **cambiandolo non** si ottiene il **numero opposto**
  - $45 = (0101101)_{CP2}$ se cambio di segno alla prima cifra
  - $(1101101)_{CP2} \rightarrow -2^6 + 2^5 + 2^3 + 2^2 + 1 = \\ = -64 + 45 = -19$
- Inoltre, un solo valore per lo 0 (cioè m volte 0), nessuna configurazione "sprecata" dalla codifica

Rappresentazione in CP2

- *Es, definire un intervallo che contenga -23 e 45*
  - $m = 7$, copro $[-2^6, 2^6 - 1] = [-64, 63]$
  - ~~$m = 6$, copro $[-2^5, 2^5 - 1] = [-32, 32]$~~ (non cont. 45)
  - $-23 \rightarrow 2^7 - 23 = 128 - 23 = 105 = (1101001)_{CP2}$
  - $45 = (0101101)_{CP2}$
- **NB:** occorre utilizzare **sempre m bit**. Se non avessi messo lo 0 iniziale in $(45)_{CP2}$ avrei ottenuto un numero negativo a 6 bit!

Conversione Decimale ➡ CP2

Metodo "operativo" per rappresentare X ad m bit

1. Controllo che $X \in [-2^{m-1}, 2^{m-1} - 1]$, altrimenti m bit non bastano
2. Se X è positivo, scrivo X utilizzando m bit
   NB: ricordandosi di aggiungerei zeri se necessario all'inizio del numero!
3. Se X è negativo:
   a) Scrivo $|X|$ utilizzando m bit
   b) **Complemento** tutti i bit di X $(1 \rightarrow 0, 0 \rightarrow 1)$
   c) **Sommo 1** al numero ottenuto

Esempi Conversione Decimale ➡ CP2

Esempio: scrivere -56 in CP2 con il numero di bit necessari

Esempi Conversione Decimale ➡ CP2

Esempio: scrivere -56 in CP2 con il numero di bit necessari

i. $m = 7$ copre $[-2^6, 2^6 - 1] = [-64, 63]$

ii. Scrivo $(56)_{10} \rightarrow 0111000$

iii. Complemento $\rightarrow 1000111$

iv. Sommo 1 $\quad 1$

v. $(1001000)_{CP2} = (-56)_{10}$

56	0
28	0
14	0
7	1
3	1
1	1
0	

Esercizi

Esercizio: convertire in complemento a 2 i seguenti numeri, utilizzando il numero di bit necessario per esprimerli tutti

$$
\begin{aligned}
(12)_{10} &= \\
(-12)_{10} &= \\
(-8)_{10} &= \\
(1)_{10} &= \\
(-101)_{10} &= \\
(-54)_{10} &=
\end{aligned}
$$

Conversione CP2 ➡ Decimale

Possiamo utilizzare la definizione

$$N_{CP2} = a_{m-1}a_{m-2} \dots a_1 a_0$$

$$= -a_{m-1} \times 2^{m-1} + a_{m-2} \times 2^{m-2} + \dots + a_0 \times 2^0$$

$$= -a_{m-1} \times 2^{m-1} + \sum_{i=0}^{m-2} a_i \times 2^i \ , \qquad a_i \in \{0,1\}$$

$$Es\ (1001000)_{CP2} = -2^6 + 2^3 = -64 + 8 = (-56)_{10}$$

$$(10011011)_{CP2} = -2^7 + 2^4 + 2^3 + 2^1 + 2^0 =$$

$$= -128 + 16 + 8 + 2 + 1 = (-101)_{10}$$

NB convertite sempre in decimale con questo metodo per controllare le vostre operazioni

Conversione CP2 ➡ Decimale

… in alternativa è possibile utilizzare un metodo operativo:

1. Se $(X)_{CP2}$ inizia per 0, allora è positivo: lo converto normalmente
2. Se $(X)_{CP2}$ inizia per 1, allora è negativo
   a) **Complemento** tutti i bit di $(X)_{CP2}$ $(1 \rightarrow 0, 0 \rightarrow 1)$
   b) **Sommo 1** al numero ottenuto
   c) **Converto** in decimale e cambio di segno

Esempio

Esercizio: riconvertire in decimale i seguenti numeri in complemento a 2

$(1001010)_{CP2} \rightarrow (0110101) \rightarrow (0110110) \rightarrow (-54)_{10}$

$(1001010)_{CP2} = -2^6 + 2^3 + 2^1 = -64 + 8 + 2 = -54$

$(011)_{CP2}$

$(1101001)_{CP2}$

$(11111)_{CP2}$

$(10100)_{CP2}$

$(101)_{CP2}$

Somma tra Numeri in CP2

- In CP2 l'operazione di somma si realizza **come nella rappresentazione binaria posizionale**
- Grazie alla rappresentazione in CP2 **è possibile eseguire** anche **sottrazioni** tra numeri binari con lo stesso meccanismo (i.e., somme tra interi di segno opposto)

Carry e Overflow in CP2

- In CP2 occorre **ignorare il bit di carry**, cioè il riporto che cade sul bit che cade sul segno
- In CP2 occorre individuare **l'overflow**, i.e., casi in cui il risultato è fuori dall'intervallo rappresentabile con i bit utilizzati.
- Quando c'è **overflow il risultato è inconsistente** con gli addendi:
  - Somma di due addendi positivi da un numero negativo
  - Somma di due addendi negativi da un numero positivo
- **NB** non può esserci overflow quando sommo due numeri di segno opposto

Esempio no overflow

Esempio: $60 - 54$

diventa $60 + (-54)$

$$\begin{array}{rcl} & & 1\ 1\ \ 1\ 1 \\ (60)_{10} & = & (0\ 1\ 1\ 1\ 1\ 0\ 0)_{CP2} \\ (-54)_{10} & = & \underline{(1\ 0\ 0\ 1\ 0\ 1\ 0)_{CP2}} \\ & & (1)\ 0\ 0\ 0\ 0\ 1\ 1\ 0 \end{array}$$

Il riporto (carry) viene ignorato

Quando sommo numeri di segno opposto non può esserci overflow

Il risultato è positivo $(0000110)_{CP2} = (6)_{10}$

Esempio

Esempio: $(100)_{CP2} + (101)_{CP2}$

```
    1
  1 0 0 +
  1 0 1 =
-----------
[1] (1) 0 0 1
```

- Ignoro il bit di carry
- **Overflow:** la somma di due numeri negativi mi ha dato un numero positivo.
- L'overflow si indica quadre: [1] c'è overflow, [0] non c'è
- Il risultato non ha senso, occorre scrivere gli addendi con un bit in più per rappresentare il risultato dell'operazione

Esempi

- Esempi: con $m = 4$ bit
- Indico tra () bit di carry, tra [] bit di overflow

```
-3  =>
-4  =>
-7  =>
```

```
-3  =>
+6  =>
+3  =>
```

```
-3  =>
-7  =>
-10 =>
```

```
+2  =>
+5  =>
+7  =>
```

```
+3  =>
+6  =>
+9  =>
```

Esempio TDE 11/2009

a) Si dica qual è l'intervallo di valori interi rappresentabile con la codifica in complemento a due a 9 bit.

b) Con riferimento a tale codifica indicare, giustificando brevemente le risposte, quali delle seguenti operazioni possono essere effettuate correttamente:
   i. -254 – 255
   ii. + 254 – 253
   iii. -18 + 236
   iv. + 217 + 182

c) Mostrare in dettaglio come avviene il calcolo delle operazioni (i) e (ii), evidenziando il bit di riporto e il bit di overflow così ottenuti. (Il bit di overflow è pari ad 1 se si verifica overflow, o altrimenti.)

Esempio TDE 11/2009

a. Valori rappresentabili vanno da -256 a +255.

b. Le soluzioni:

   i. -254 – 255: NO, si ottiene un valore negativo troppo grande in valore assoluto

   ii. + 254 – 253: SI, si ottiene un valore piccolo in valore assoluto

   iii. -18 + 236: SI, si ottiene un valore positivo, grande in valore assoluto ma nei limiti

   iv. + 217 + 182: NO, si ottiene un valore positivo troppo grande in valore assoluto

c.

```
   100000010 (-254)
   100000001 (-255)
  ------------------
[1](1)000000011 (-509)
```

ii.

```
   011111110 (+254)
   100000011 (-253)
  ------------------
[0](1)000000001 (+1)
```

Architettura di un Sistema Informatico

Informatica ICA (LC) AA 2016/ 2017

Giacomo Boracchi

17 Novembre 2016

giacomo.boracchi@polimi.it

Un Sistema Informatico

Hardware

Software

Sistema Informatico

- Sistemi informatico: L'esecutore dei programmi
- PC, laptop, telefoni, smartphones, server con migliaia di utenti etc… tutti sistemi informatici
- Sono oggetti complessi, costruiti da diverse parti che interagiscono tra loro.
- I sistemi informatici sono composti da
  - **Hardware**: i componenti fisici del sistema
  - **Software**: i programmi eseguiti dal sistema (es. web browser, mail, suite office, sistema operativo, compilatori, IDE… i programmi che scriveremo)
- Consideriamo inizialmente l'hardware dei sistemi informatici.

La Macchina di Von Neumann

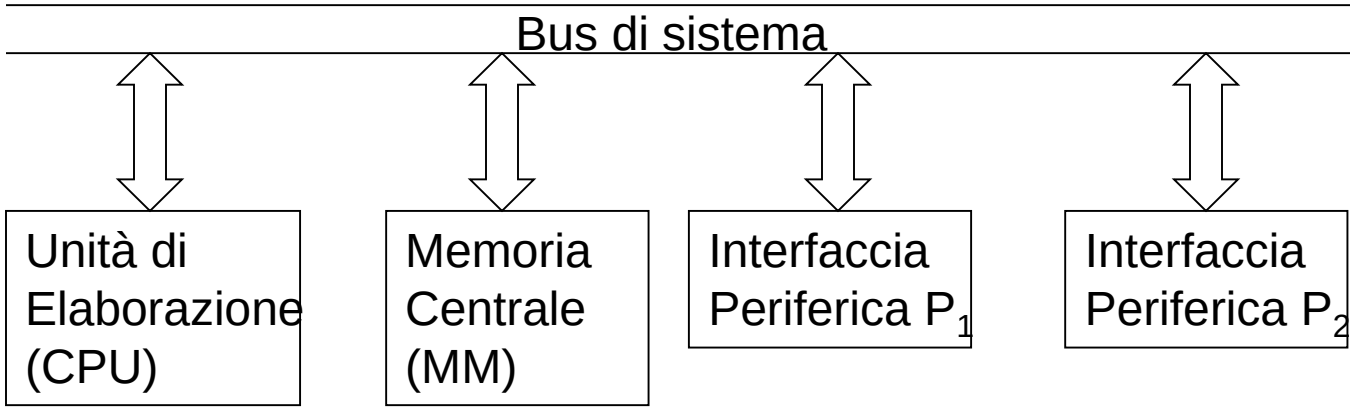
Un modello dell'architettura dei calcolatori

La Macchina di Von Neumann

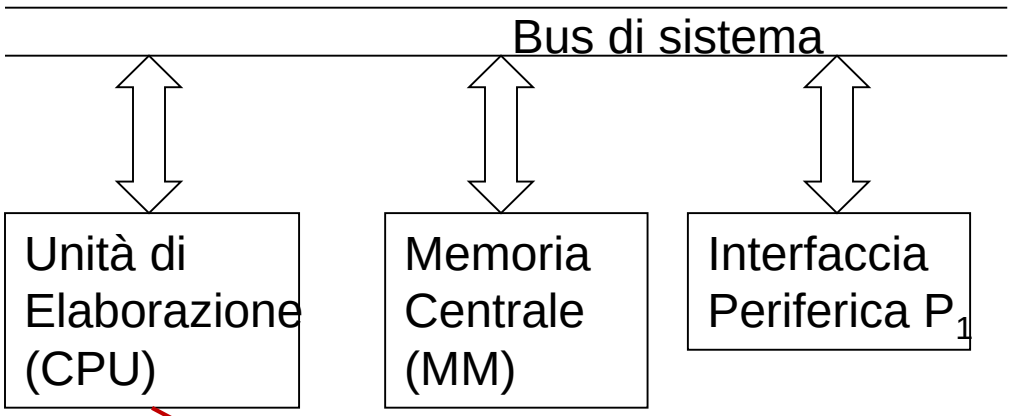
- Modello composto da **quattro elementi funzionali**
  - **Unità di elaborazione (CPU):** interpretazione ed esecuzione dei programmi, coordinamento macchina
  - **Memoria centrale:** contiene dati ed istruzioni
  - **Interfacce delle periferiche:** scambio informazioni con mondo esterno (e.g, stampante, tastiera, mouse, rete, schermo, HDD ..)
  - **Bus di sistema:** collega gli altri elementi funzionali

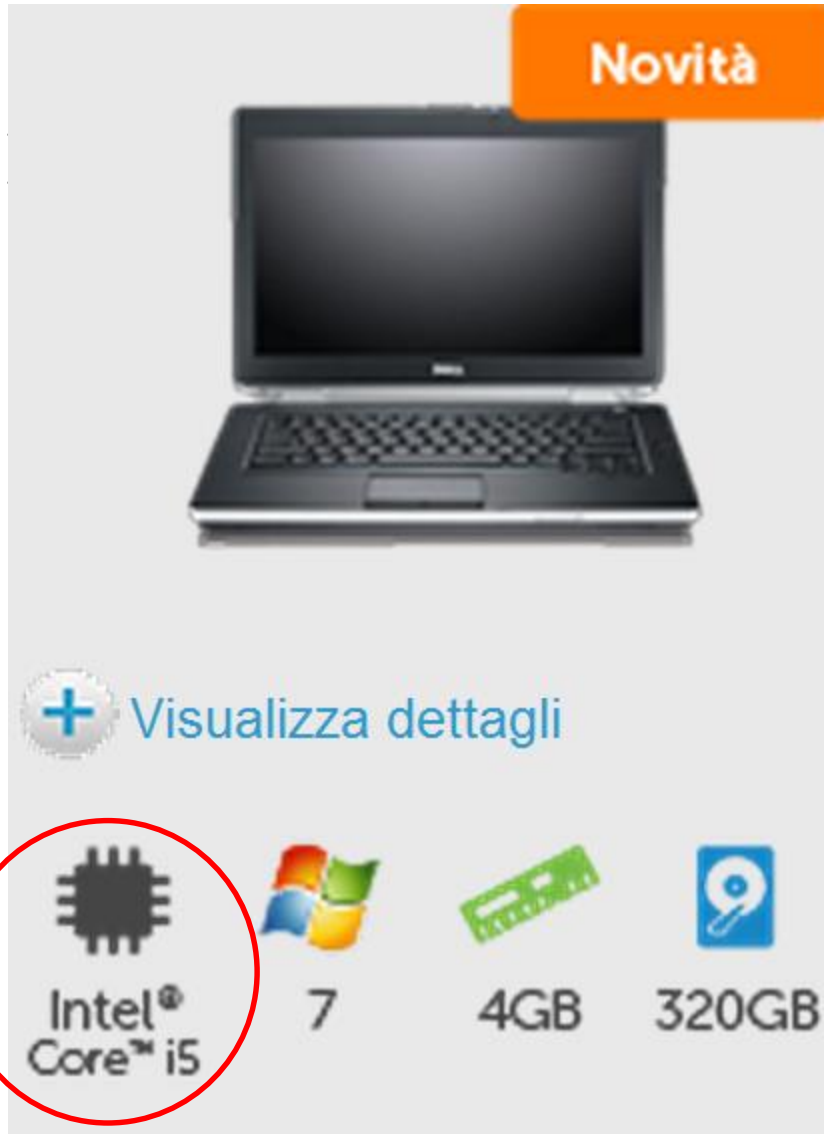
La Macchina di Von Neumann

- Modello composto da **quattro elementi funzionali**
  - **Unità di elaborazione (CPU):** interpretazione ed esecuzione dei programmi, coordinamento macchina
  - **Memoria centrale:** contiene dati ed istruzioni
  - **Interfacce delle periferiche:** scambio informazioni con mondo esterno (e.g, stampante, tastiera, mouse, rete, schermo, HDD ..)
  - **Bus di sistema:** collega gli altri elementi funzionali

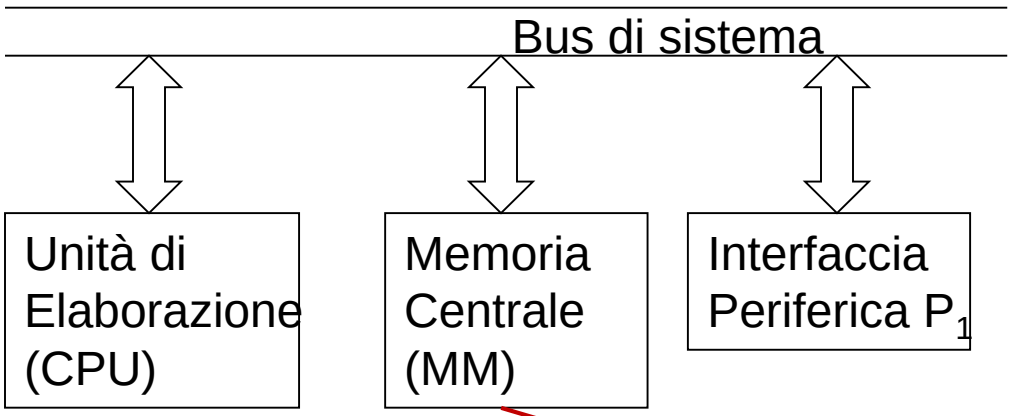


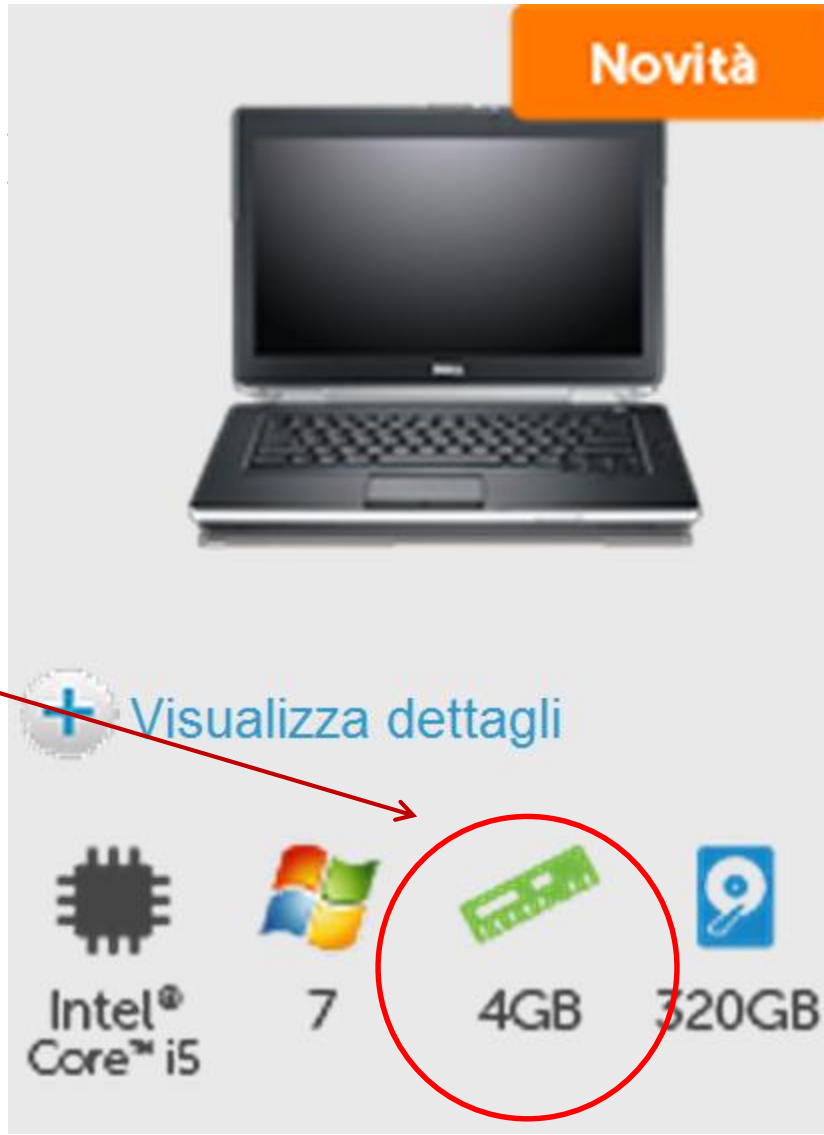
La Macchina di Von Neumann



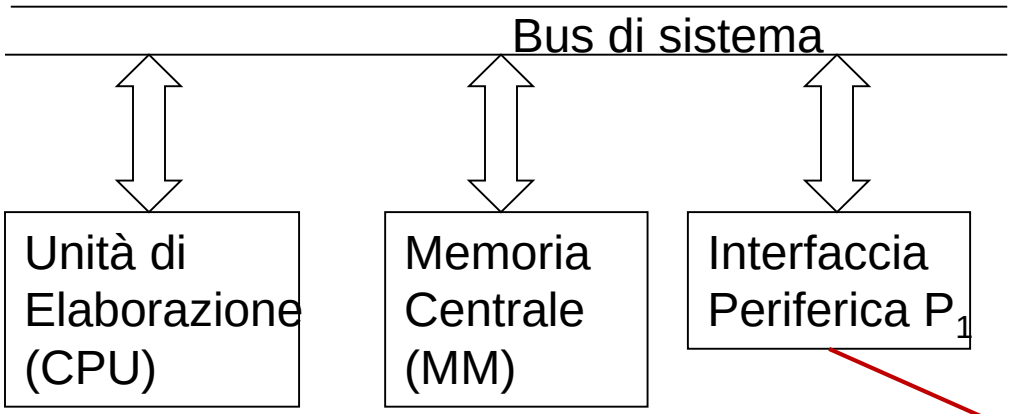


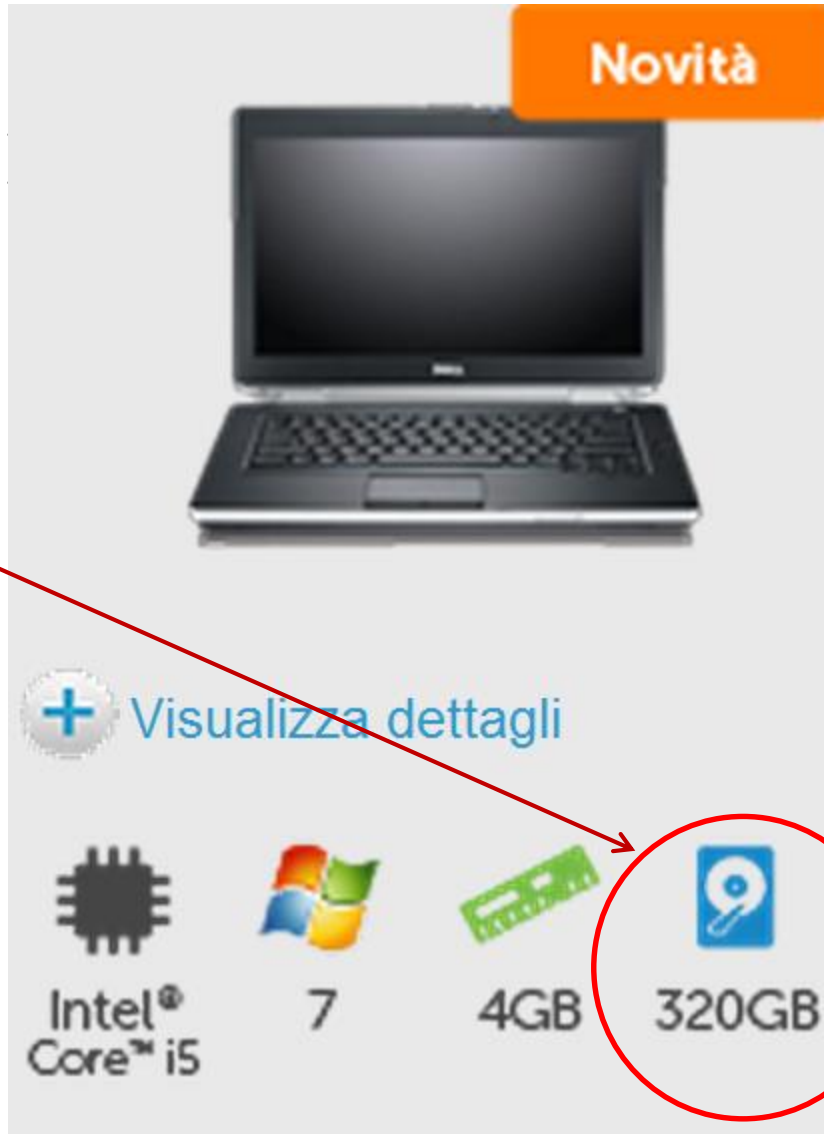
La Macchina di Von Neumann



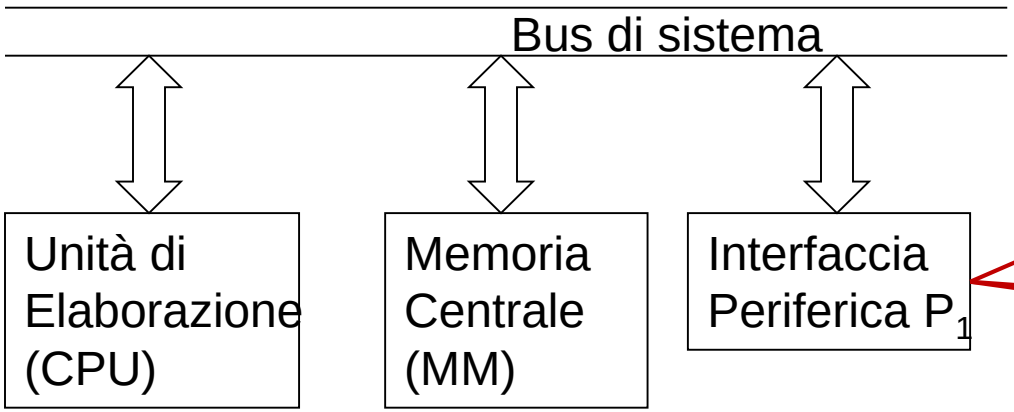


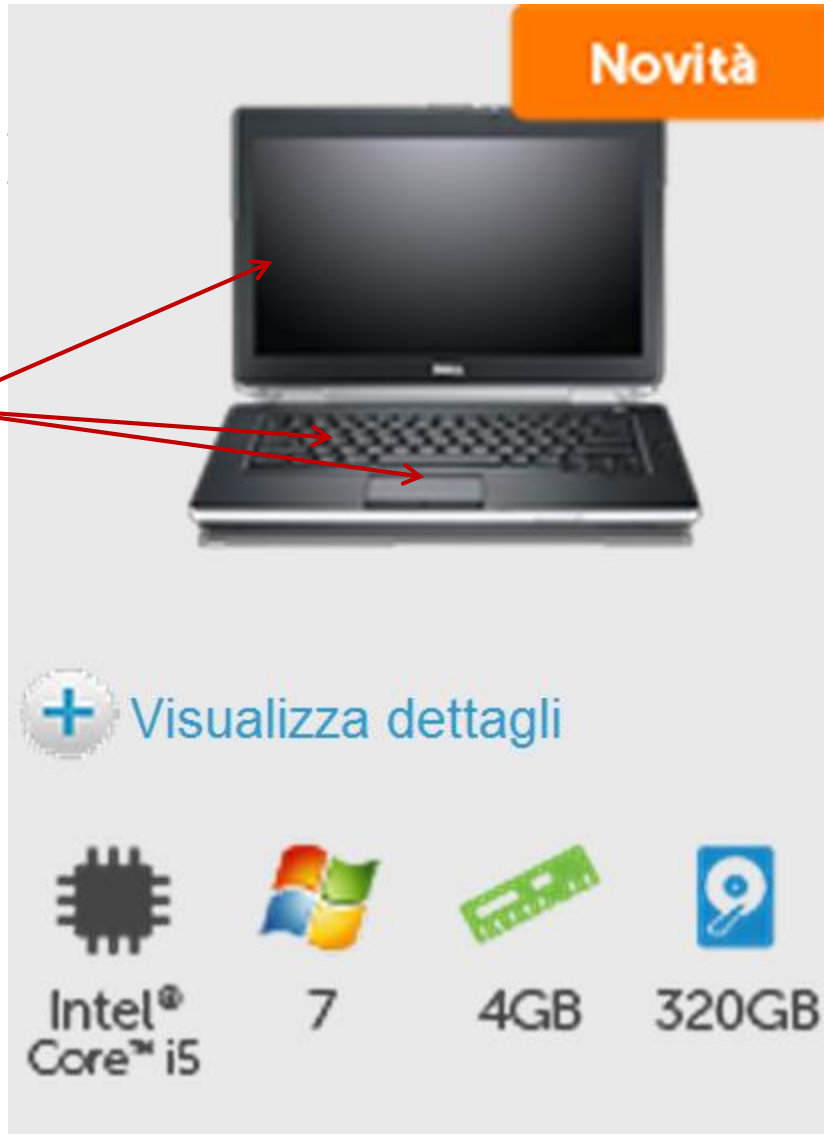
La Macchina di Von Neumann





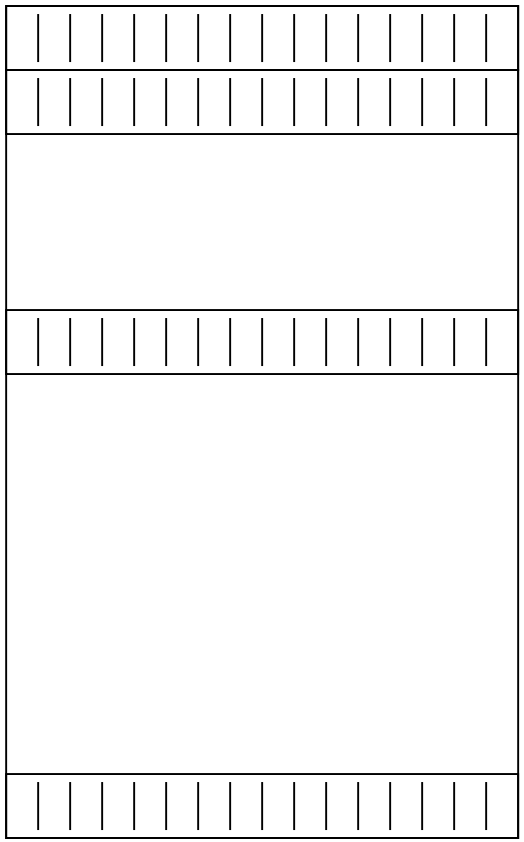
La Macchina di Von Neumann





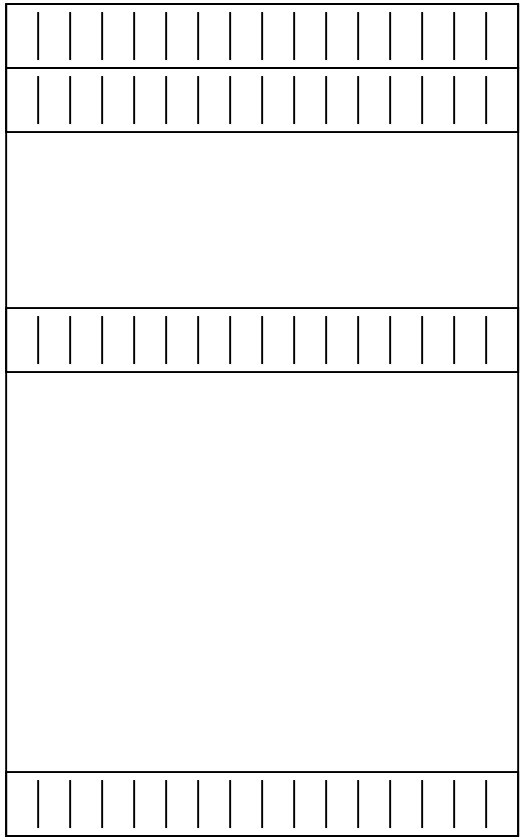
La Memoria Centrale (MM)

- Contiene i **programmi (sequenza di istruzioni)** in **esecuzione** ed i relativi **dati**



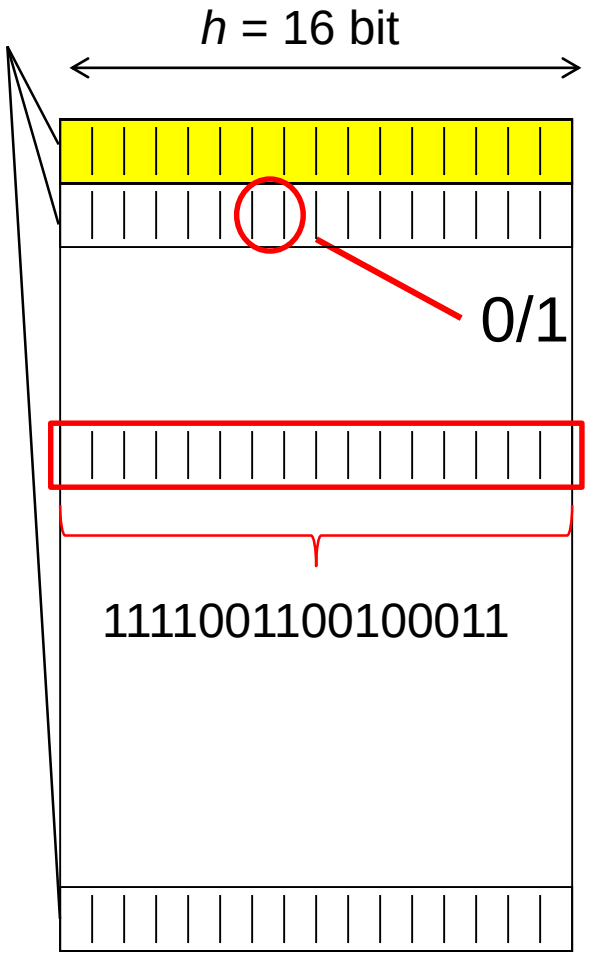
La Memoria Centrale (MM)

- Contiene i **programmi (sequenza di istruzioni)** in **esecuzione** ed i relativi **dati**
- È schematizzata come una sequenza di celle.



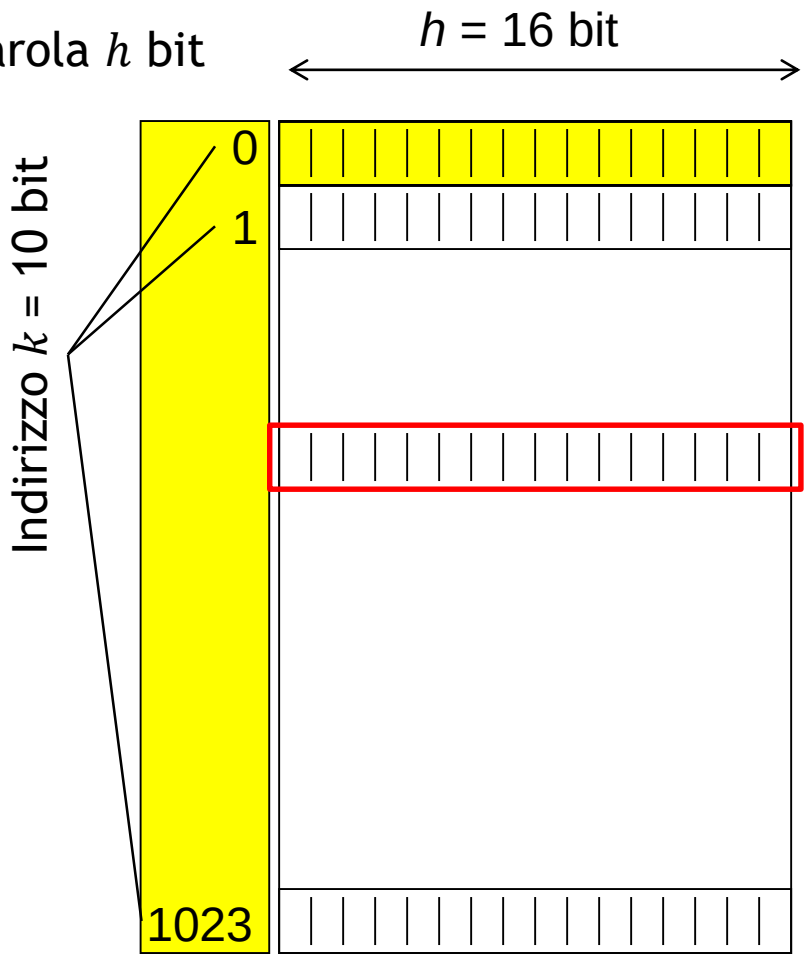
La Memoria Centrale (MM)

- Contiene i **programmi (sequenza di istruzioni)** in **esecuzione** ed i relativi **dati**
- È schematizzata come una sequenza di celle.
- Ogni cella contiene *h* bit, i.e., una Parola (*word*)



La Memoria Centrale (MM)

- Contiene i **programmi (sequenza di istruzioni)** in **esecuzione** ed i relativi **dati**
- È schematizzata come una sequenza di celle.
- Ogni cella contiene *h* bit, i.e., una Parola (*word*)
- Ogni cella ha un indirizzo
- Se ho a disposizione *k* bit per scrivere l'indirizzo, lo spazio di indirizzamento è 2^k celle



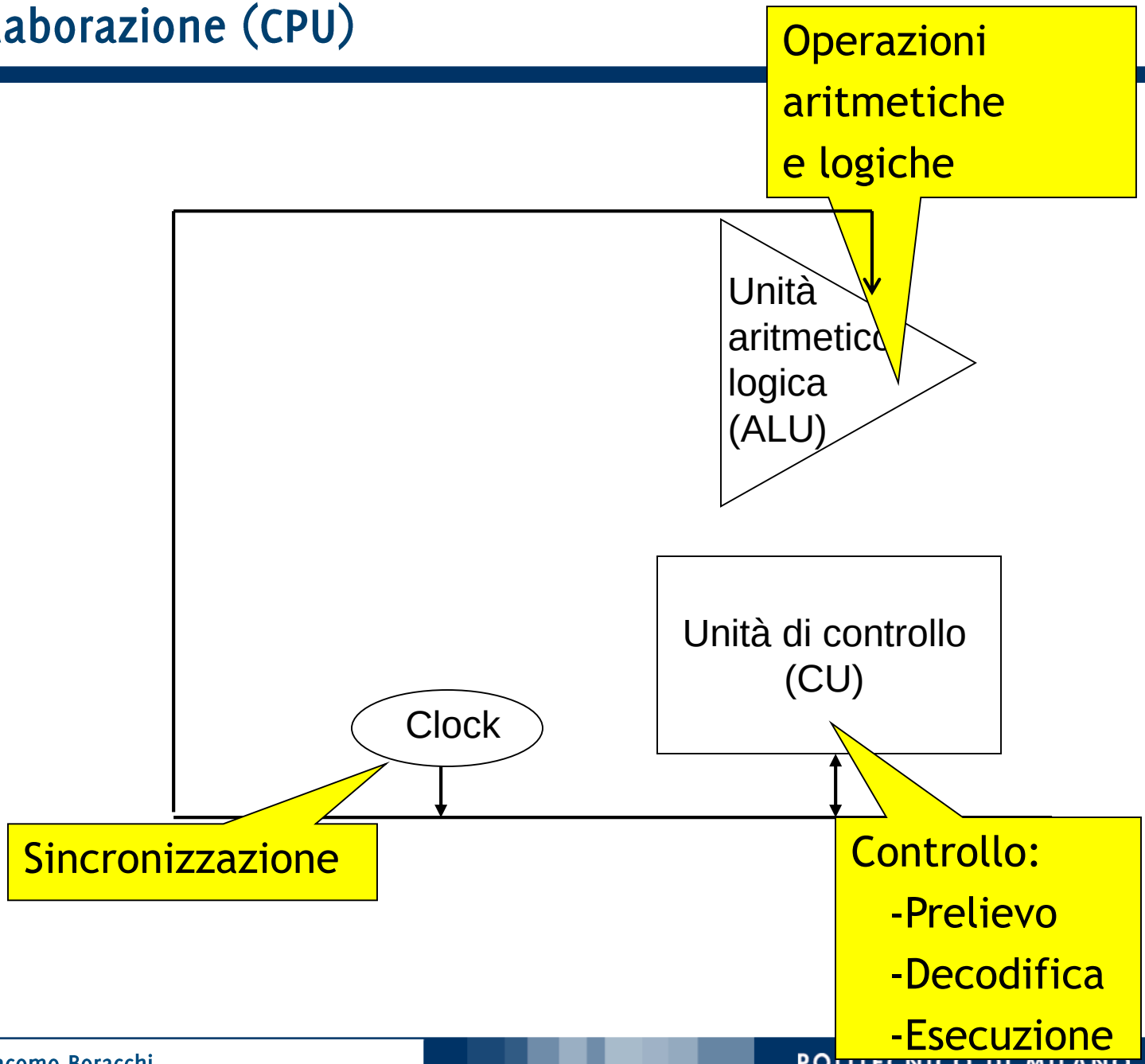
La Memoria Centrale (MM)

- La MM contiene i **programmi in esecuzione**: ogni **dato e ogni istruzione**, prima di essere elaborato, viene copiato in memoria centrale.
- Diversi tipi di Memorie
  - RAM (*Random Access Memory*) memoria volatile.
  - ROM *(Read Only Memory)* memoria permanente.
  - EPROM *(Erasable Programmable ROM)* riprogrammabile
- **L'HDD** è memoria permanente ma **non è memoria centrale** ed in riferimento alla macchina di Von Neumann è una periferica.

L'Unità di Elaborazione (CPU)

- La *Central Processing Unit* (CPU) **coordina** il funzionamento del **calcolatore** ed **esegue** i **programmi**:
  estrae, decodifica ed esegue le istruzioni in memoria.
- Le istruzioni possono comportare **elaborazione** o **trasferimento** dell'informazione
- La CPU contiene a sua volta:
  - l'**Unità di Controllo** che preleva e decodifica istruzioni dalla MM, invia segnali per eseguire le istruzioni
  - Il **Clock di sistema**,
  - L'**Unità Aritmetico Logica**

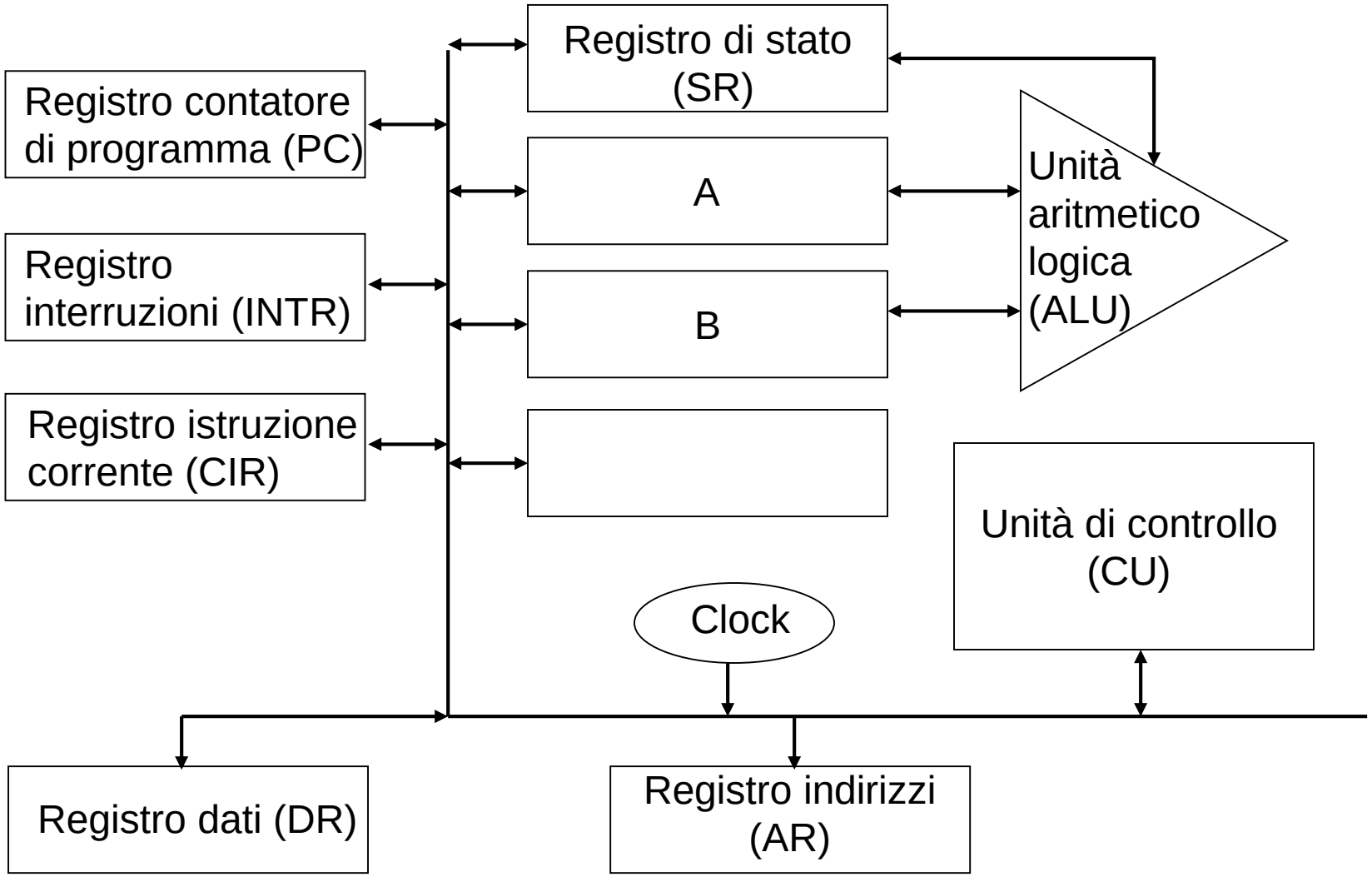
L'Unità di Elaborazione (CPU)



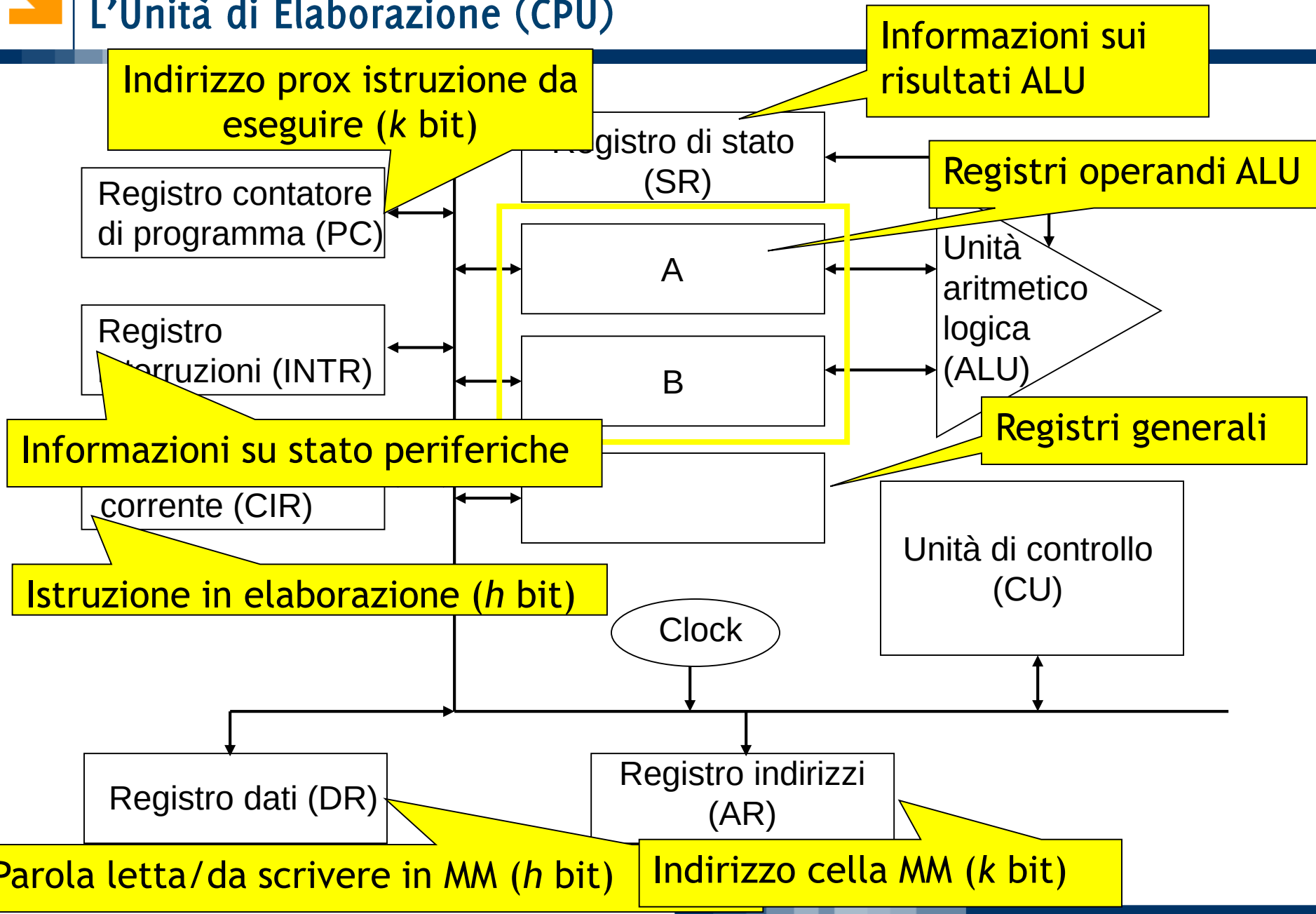
L'Unità di Elaborazione (CPU)

- La *Central Processing Unit* (CPU) **coordina** il funzionamento del **calcolatore** ed **esegue** i **programmi**:
  estrae, decodifica ed esegue le istruzioni in memoria
- Le istruzioni possono comportare **elaborazione** o **trasferimento** dell'informazione
- La CPU contiene a sua volta:
  - l'**Unità di Controllo** che preleva e decodifica istruzioni dalla MM, invia segnali per eseguire le istruzioni .
  - Il **Clock** di sistema,
  - l'**Unità Aritmetico Logica**
- La CPU contiene inoltre molti **registri**: memorie «rapide» per informazioni richieste dalla CU (es due numeri da sommare, il loro risultato)

L'Unità di Elaborazione (CPU)



L'Unità di Elaborazione (CPU)

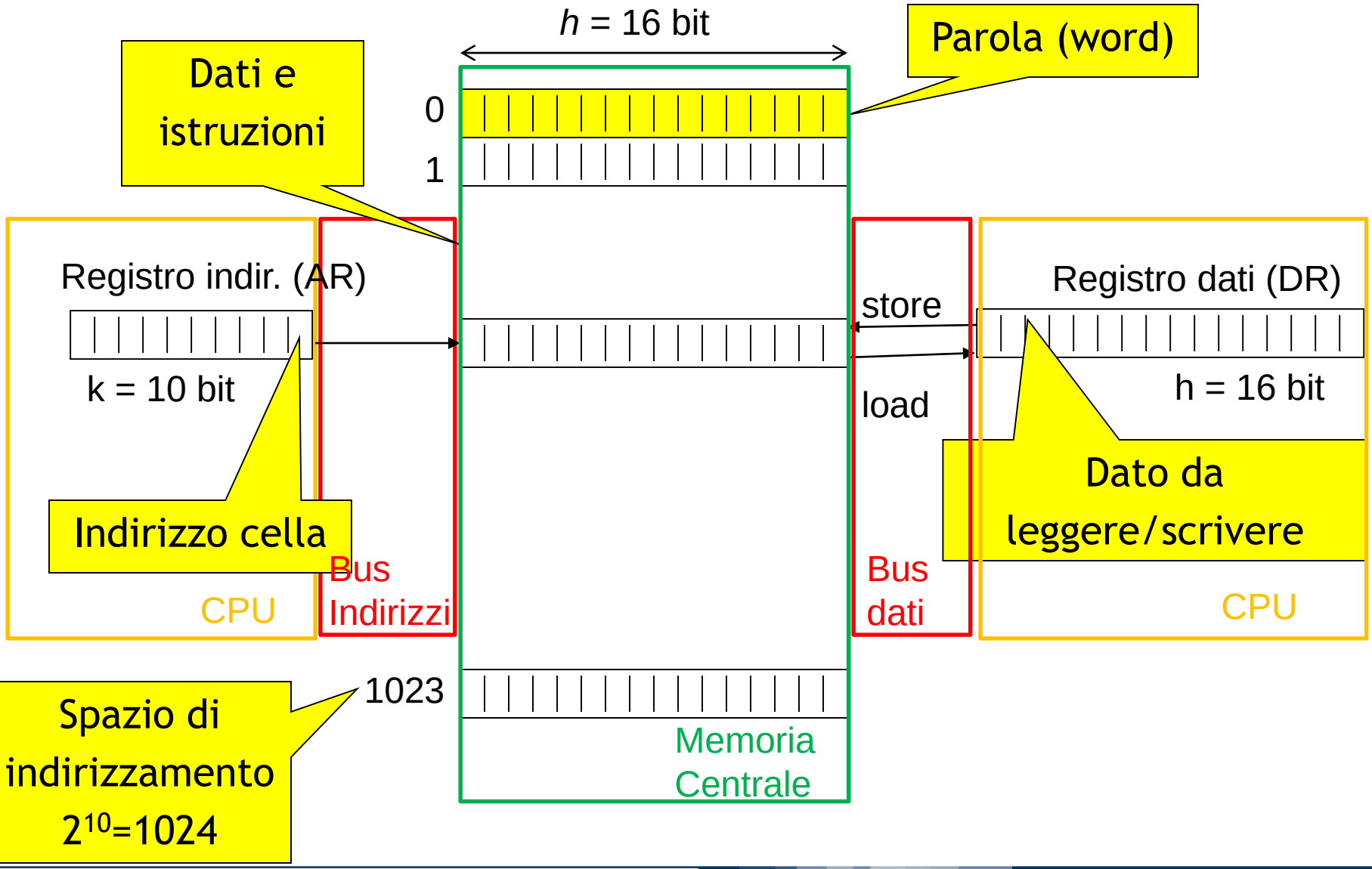


Bus di Sistema

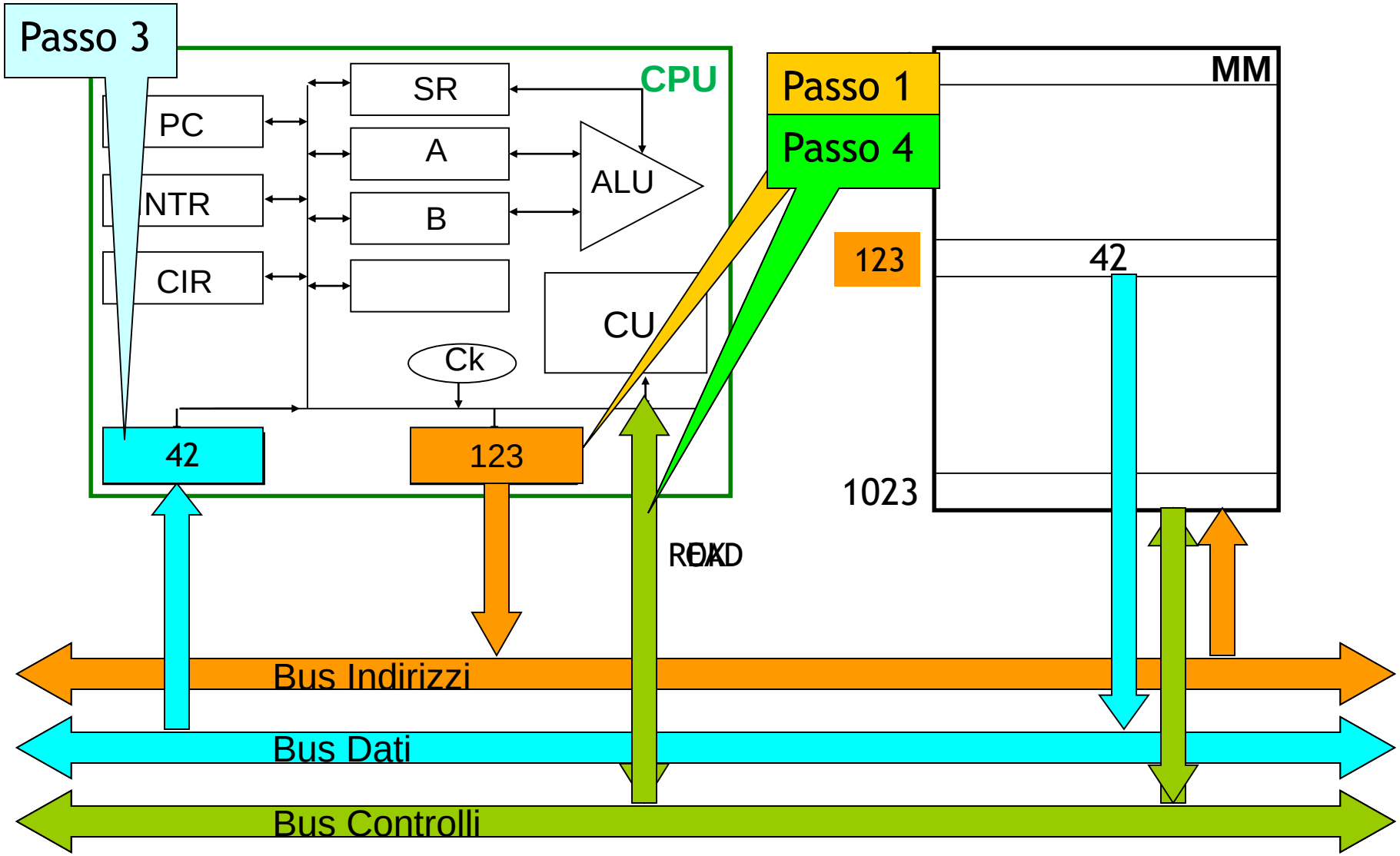
- È un insieme di connessioni che permettono di trasferire l'informazione tra **due** entità funzionali (una trasmette l'altra riceve)
- Due soli tipi di connessioni logiche, **stabilite** dalla **CPU**:
  - CPU (**master**) - memoria (**slave**)
  - CPU (**master**) - interfaccia periferica (**slave**)

  le connessioni fisiche sono sempre presenti.
- Ci sono **tre tipi di linee**, con tre funzionalità diverse
  - Bus dati
  - Bus indirizzi
  - Bus controlli (il master lo usa per trasmettere allo slave i codici relativi alle istruzioni da eseguire, lo slave per dare feedback sull'esecuzione)

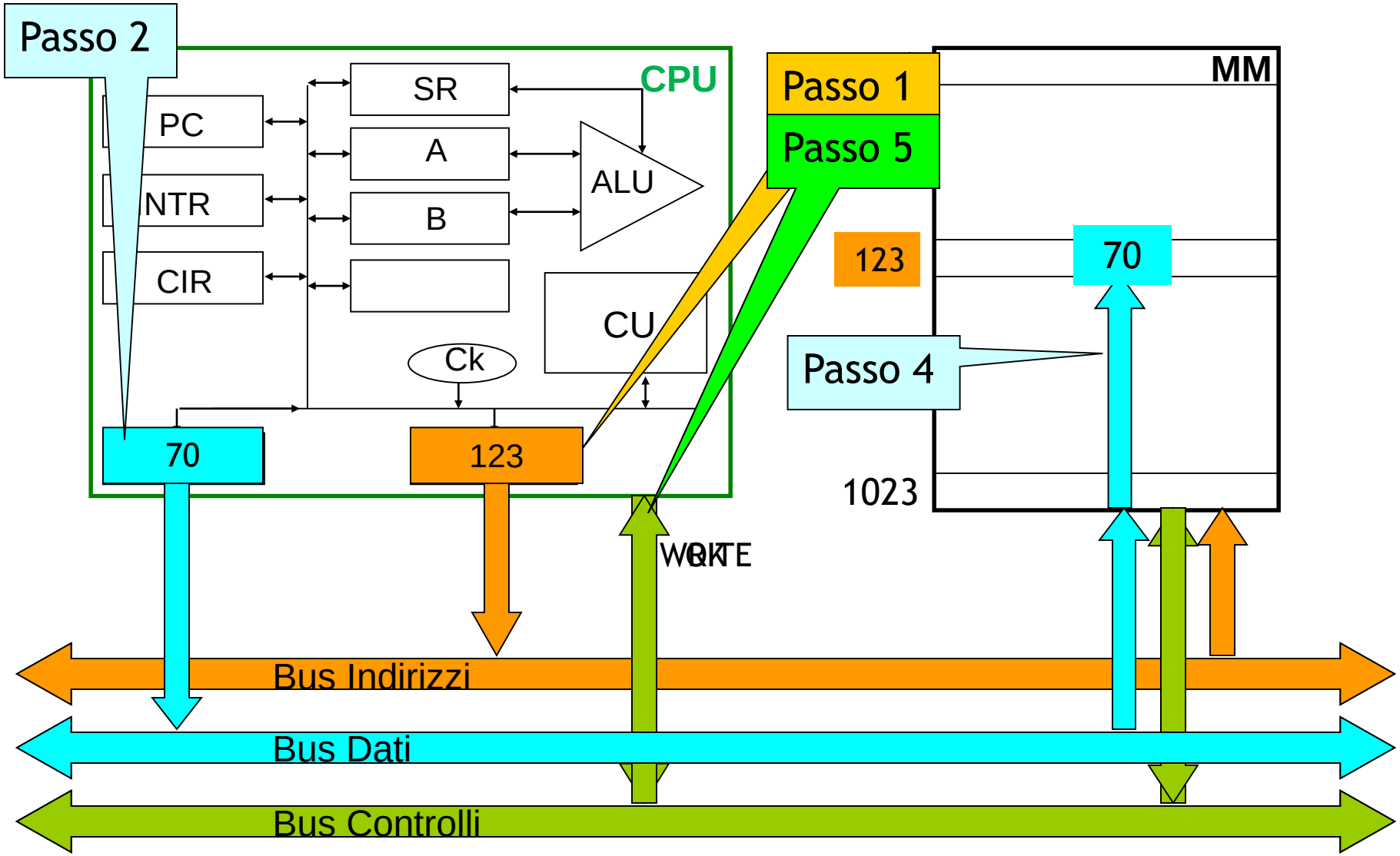
Lettura e Scrittura dalla Memoria Centrale



Sequenza di Lettura



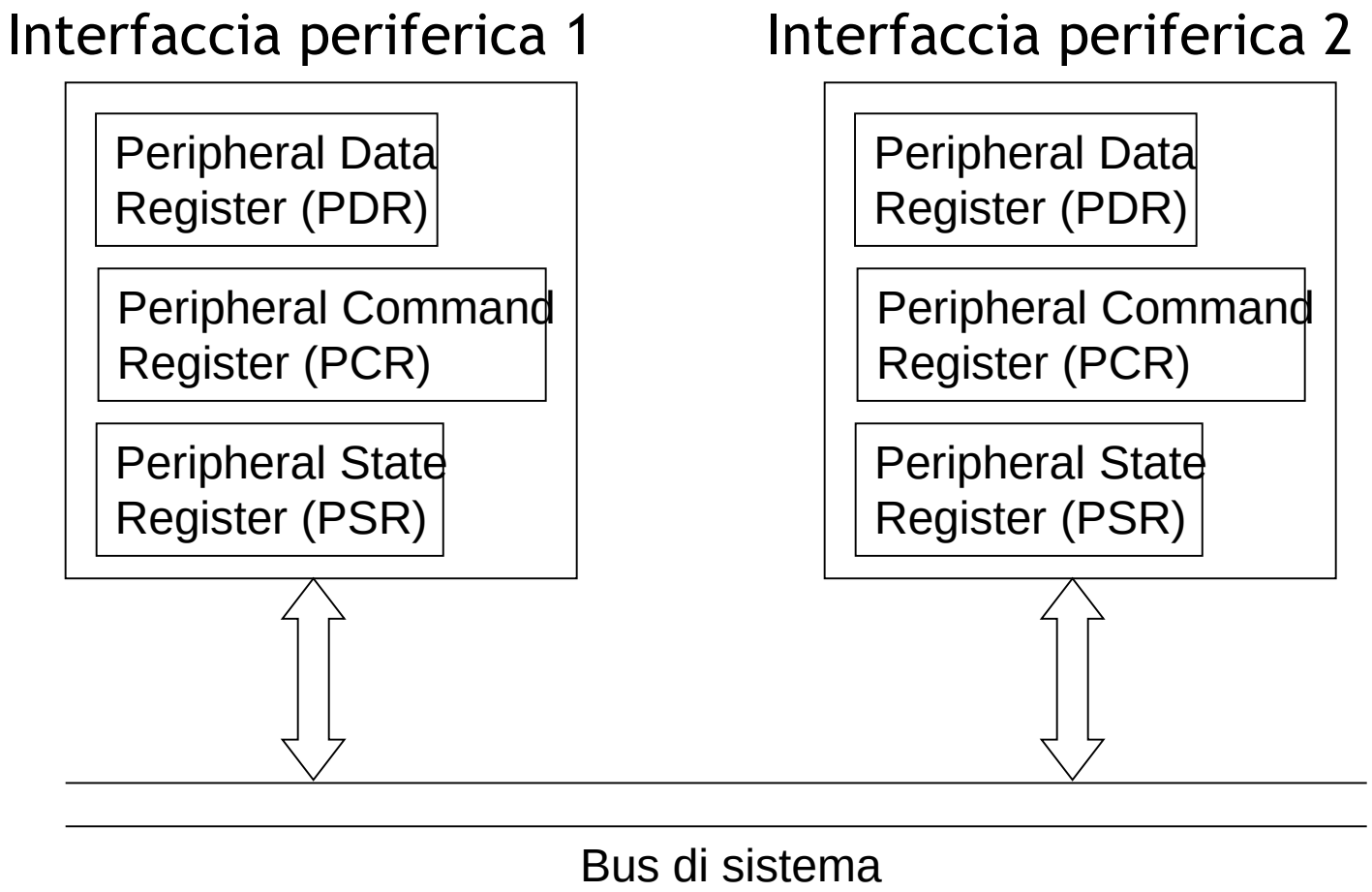
Sequenza di Scrittura



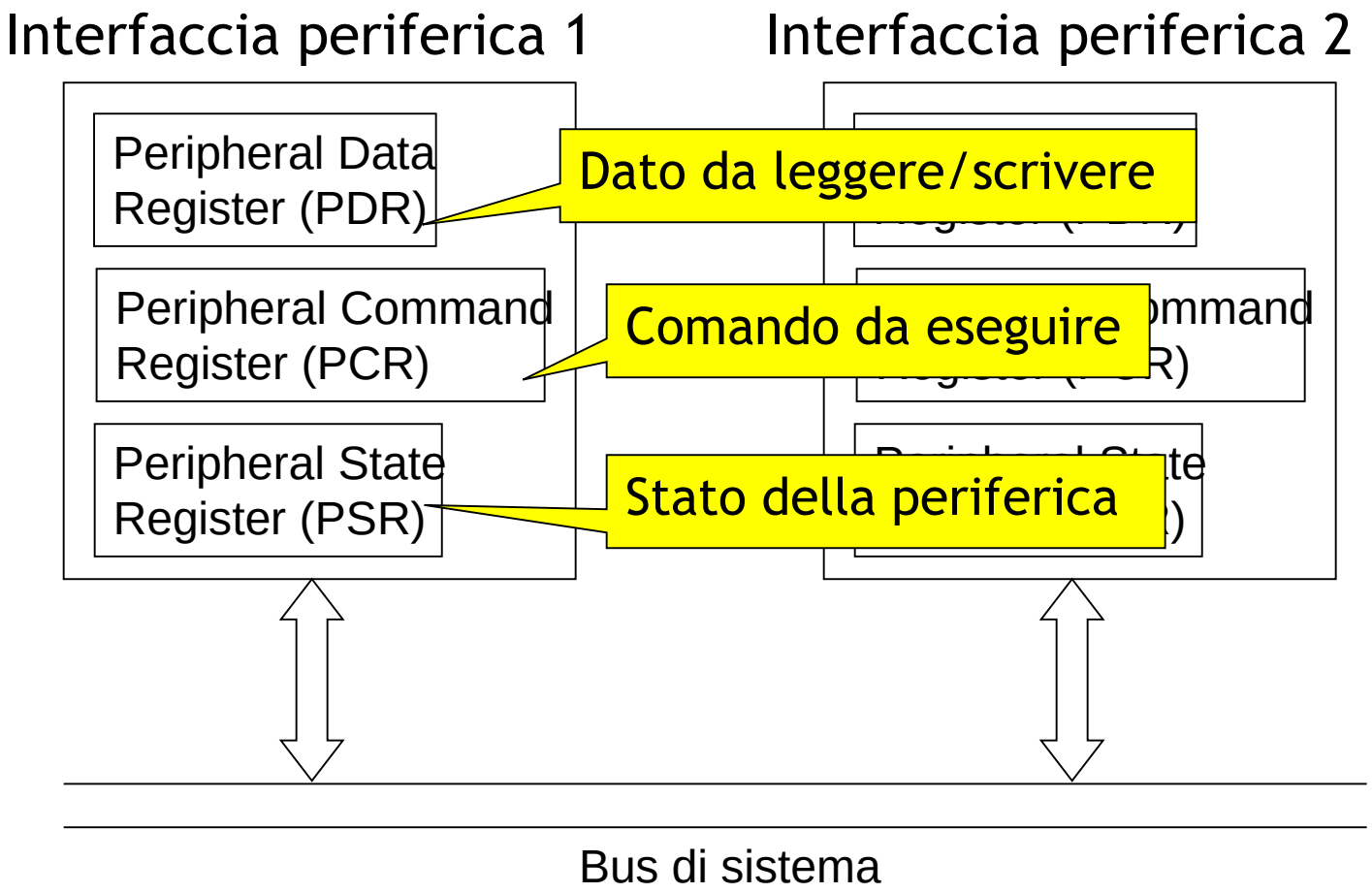
Interfacce alle Periferiche

- Le interfacce collegano il calcolatore a periferiche esterne
- Ogni interfaccia contiene dei registri per lo scambio dei dati con la periferica
  - Registro dati
  - Registro comando della periferica
  - Registro di stato

Interfacce alle Periferiche



Interfacce alle Periferiche



I programmi

nella macchina di Von Neumann

I Programmi nella Macchina di Von Neumann

Le **istruzioni** sono (necessariamente) codificate in **binario** e, **come i dati,** sono salvate in **parole nella MM**

I Programmi nella Macchina di Von Neumann

Le **istruzioni** sono (necessariamente) codificate in **binario** e, **come i dati,** sono salvate in **parole nella MM**

Supponiamo una MM con parole da $h = 16$ bit ed indirizzi da $k = 10$ bit, con istruzioni così codificate:

Codice operativo (4bit) oo Indirizzo Operando (10bit)

ad esempio, 0100000000010000

I Programmi nella Macchina di Von Neumann

Le **istruzioni** sono (necessariamente) codificate in **binario** e, **come i dati,** sono salvate in **parole nella MM**

Supponiamo una MM con parole da $h = 16$ bit ed indirizzi da $k = 10$ bit, con istruzioni così codificate:

Codice operativo (4bit) oo Indirizzo Operando (10bit)

ad esempio, 0100000000010000

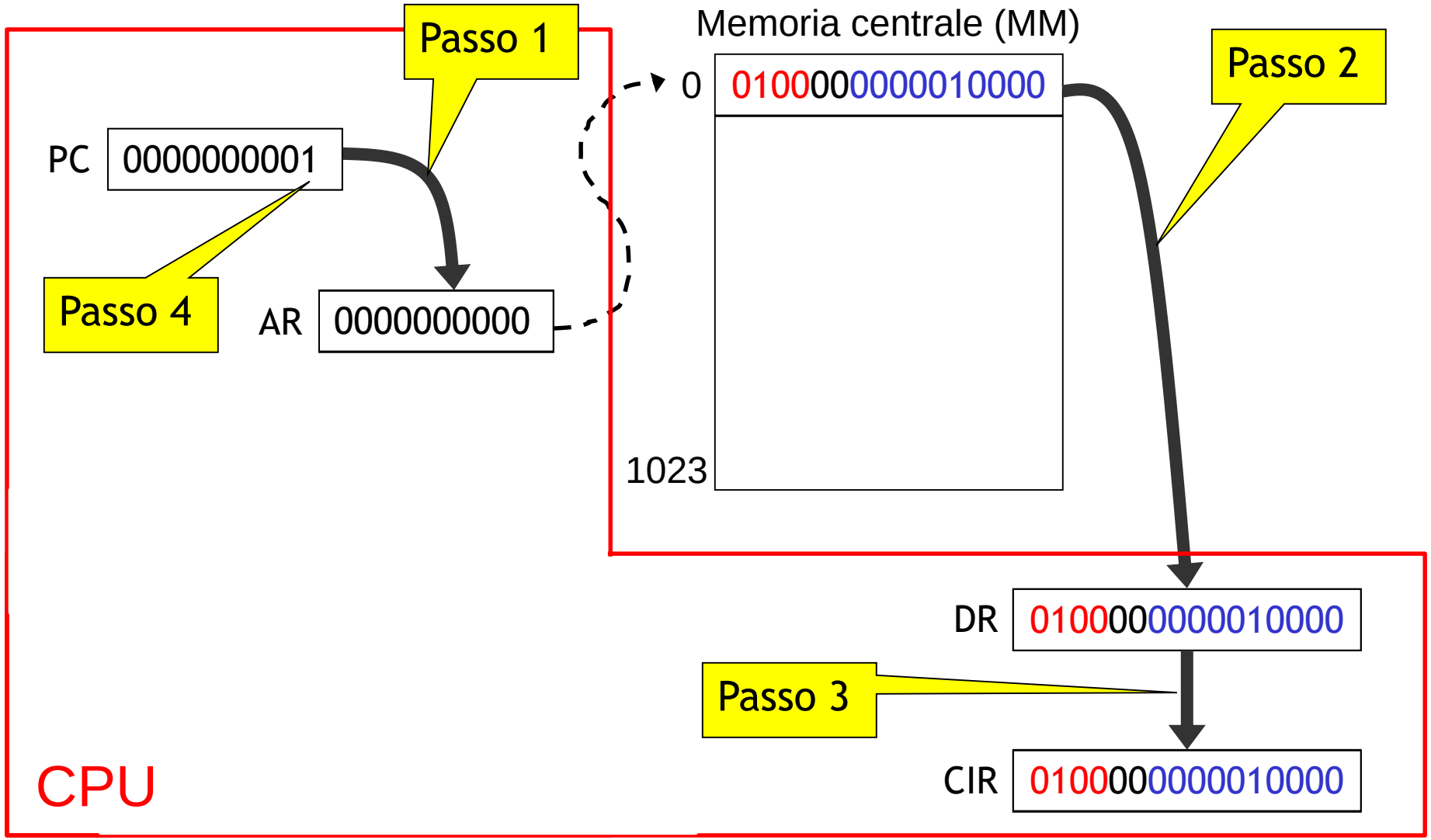
Consideriamo le seguenti istruzioni eseguibili dalla CPU

- Lettura da periferica, scrittura su periferica
- Caricare un dato da MM in un registro della CPU (*load*)
- Salvare in MM un dato di un registro della CPU (*store*)
- Operazioni aritmetiche (le gestisce la ALU)
- Istruzioni di salto (per cambiare il flusso di esecuzione)

Le Tre Fasi Per Eseguire un'Istruzioni

1. **Fetch:** Acquisizione dell'istruzione dalla MM
   1. Trasferimento da PC a AR dell' indirizzo cella contenente l'istruzione da eseguire.
   2. Lettura dalla MM della cella all'indirizzo in AR, contenuto trasferito sul DR (l'istruzione è un dato)
   3. Sposta da DR a CIR (riferimento istr. in esecuzione)
   4. Incrementa PC (definisce la prossima istruzione: incremento di 1 = sequenzialità)
2. **Decodifica**: riguarda il codice operativo, legge dal CIR
3. **Esecuzione**: dipende dall'istruzione specifica.

Fase di Fetch

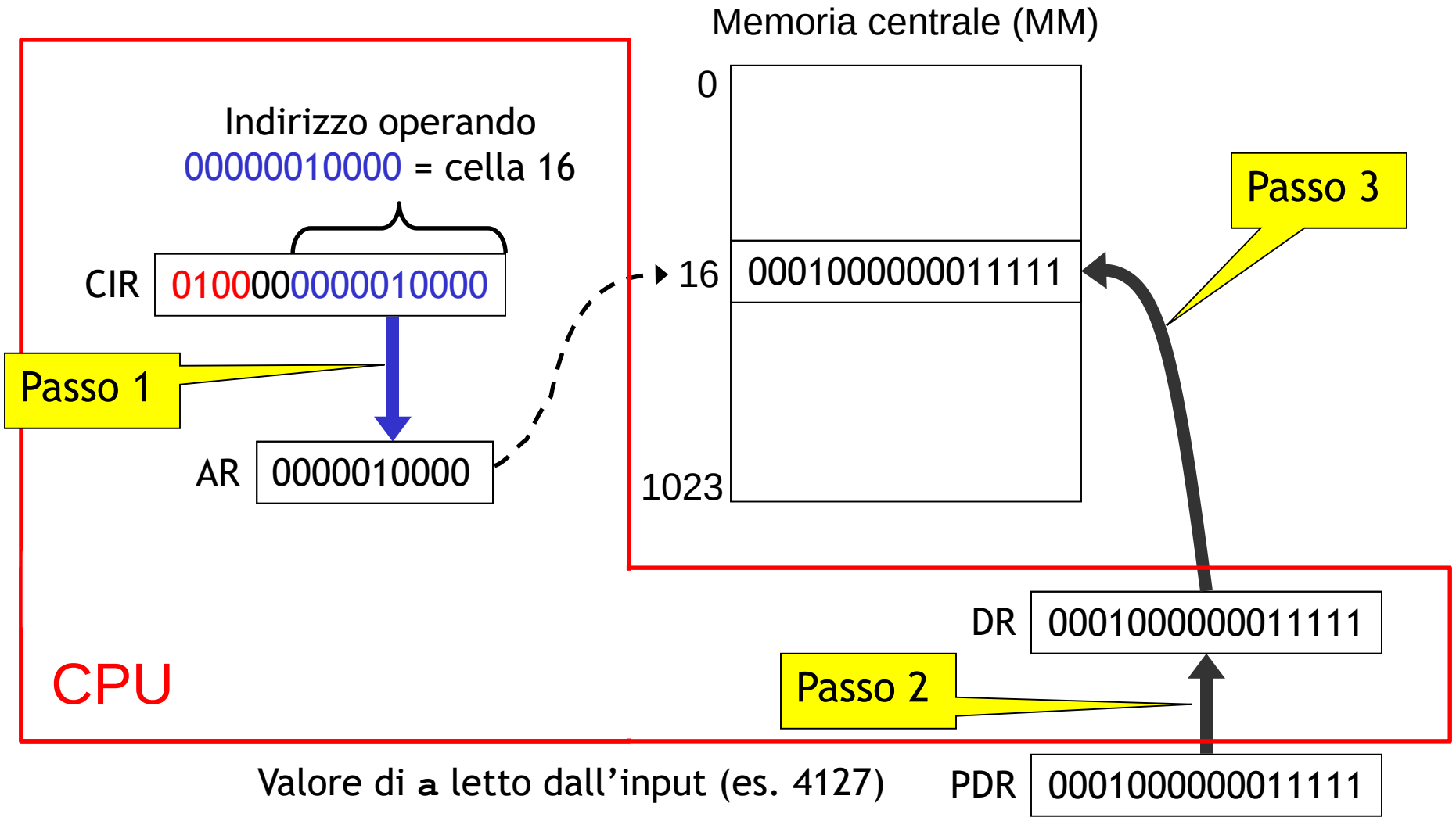


Fase di Decodifica 1ª istruzione

CIR 0100000000010000

Codice operativo 0100 = leggi da input

Fase di esecuzione: esempio lettura da input



Esempio

- Si consideri una Macchina di Von Neumann con Parola a 16 bit, indirizzi a 10 bit e codice operativo 4 bit
- Vogliamo calcolare il valore dell'espressione:

  $$(a+b) \cdot (c+d)$$

  leggendo i valori delle variabili **a**, **b**, **c**, **d** dal dispositivo di ingresso e scrivendo il risultato della valutazione sul dispositivo di uscita.

Esempio

1. Leggi dal dispositivo di ingresso il valore delle variabili **`a`**, **`b`**, **`c`**, **`d`**
2. Somma il valore di **`a`** al valore di **`b`**
3. Salva il risultato parziale ottenuto
4. Somma il valore di **`c`** al valore di **`d`**
5. Moltiplica il risultato parziale appena ottenuto con quello precedentemente salvato
6. Scrivi sul dispositivo di uscita il risultato della valutazione complessiva
7. Termina l'esecuzione del programma.

Esempio

1. Leggi dal dispositivo di ingresso il valore delle variabili **`a`**, **`b`**, **`c`**, **`d`**
2. Somma il valore di **`a`** al valore di **`b`**
3. Salva il risultato parziale ottenuto
4. Somma il valore di **`c`** al valore di **`d`**
5. Moltiplica il risultato parziale appena ottenuto con quello precedentemente salvato
6. Scrivi sul dispositivo di uscita il risultato della valutazione complessiva
7. Termina l'esecuzione del programma.

Il programma deve essere tradotto in opportuni codici operativi e indirizzi di celle di memoria!

Esempio: più nel dettaglio

1. Leggi dal dispositivo di ingresso il valore delle variabili **a**

1. Scrivi nella cella di memoria centrale riservata al valore di **a** il valore letto dal registro dati della periferica.
   - Occorre la posizione di **a,b,c,d**

Esempio: più nel dettaglio

1. Leggi dal dispositivo di ingresso il valore delle variabili **a**
2. Somma il valore di **a** al valore di **b**

1. Scrivi nella cella di memoria centrale riservata al valore di **a** il valore letto dal registro dati della periferica.
   - Occorre la posizione di **a,b,c,d**
2. Somma il valore di **a** al valore di **b**

   2.1 Copia il contenuto della cella di memoria riservata ad **a** nel registro A

   2.2 Copia il contenuto della cella di memoria riservata a **b** nel registro B

   2.3 Somma contenuto dei registri A e B

Esempio: più nel dettaglio

1. Leggi dal dispositivo di ingresso il valore delle variabili **a**
2. Somma il valore di **a** al valore di **b**
3. Salva il risultato parziale ottenuto

1. Scrivi nella cella di memoria centrale riservata al valore di **a** il valore letto dal registro dati della periferica.
   - Occorre la posizione di **a,b,c,d**
2. Somma il valore di **a** al valore di **b**

   2.1 Copia il contenuto della cella di memoria riservata ad **a** nel registro A

   2.2 Copia il contenuto della cella di memoria riservata a **b** nel registro B

   2.3 Somma contenuto dei registri A e B
3. Salva il risultato parziale, contenuto nel registro A, in una cella di memoria predisposta per il risultato (**z**).

Esempio: più nel dettaglio

1. Leggi dal dispositivo di ingresso il valore delle variabili **a**
2. Somma il valore di **a** al valore di **b**
3. Salva il risultato parziale ottenuto
4. Somma il valore di **c** al valore di **d**

4. Somma il valore di **c** al valore di **d**
   4.1 Copia il contenuto della cella di memoria riservata a **c** nel registro A
   4.2 Copia il contenuto della cella di memoria riservata a **d** nel registro B
   4.3 Somma il contenuto dei registri A e B

Esempio: più nel dettaglio

1. Leggi dal dispositivo di ingresso il valore delle variabili **a**
2. Somma il valore di **a** al valore di **b**
3. Salva il risultato parziale ottenuto
4. Somma il valore di **c** al valore di **d**
5. Moltiplica il risultato parziale appena ottenuto con quello precedentemente salvato

4. Somma il valore di **c** al valore di **d**
   - 4.1 Copia il contenuto della cella di memoria riservata a **c** nel registro A
   - 4.2 Copia il contenuto della cella di memoria riservata a **d** nel registro B
   - 4.3 Somma il contenuto dei registri A e B
5. Moltiplica
   - 5.1 Copia il contenuto della cella **z** nel registro B (**z** e B contengono ora **a + b**, mentre A contiene **c + d**)
   - 5.2 Moltiplica contenuto dei registri A e B

Esempio: più nel dettaglio

6. Scrivi sul dispositivo di uscita il risultato della valutazione complessiva

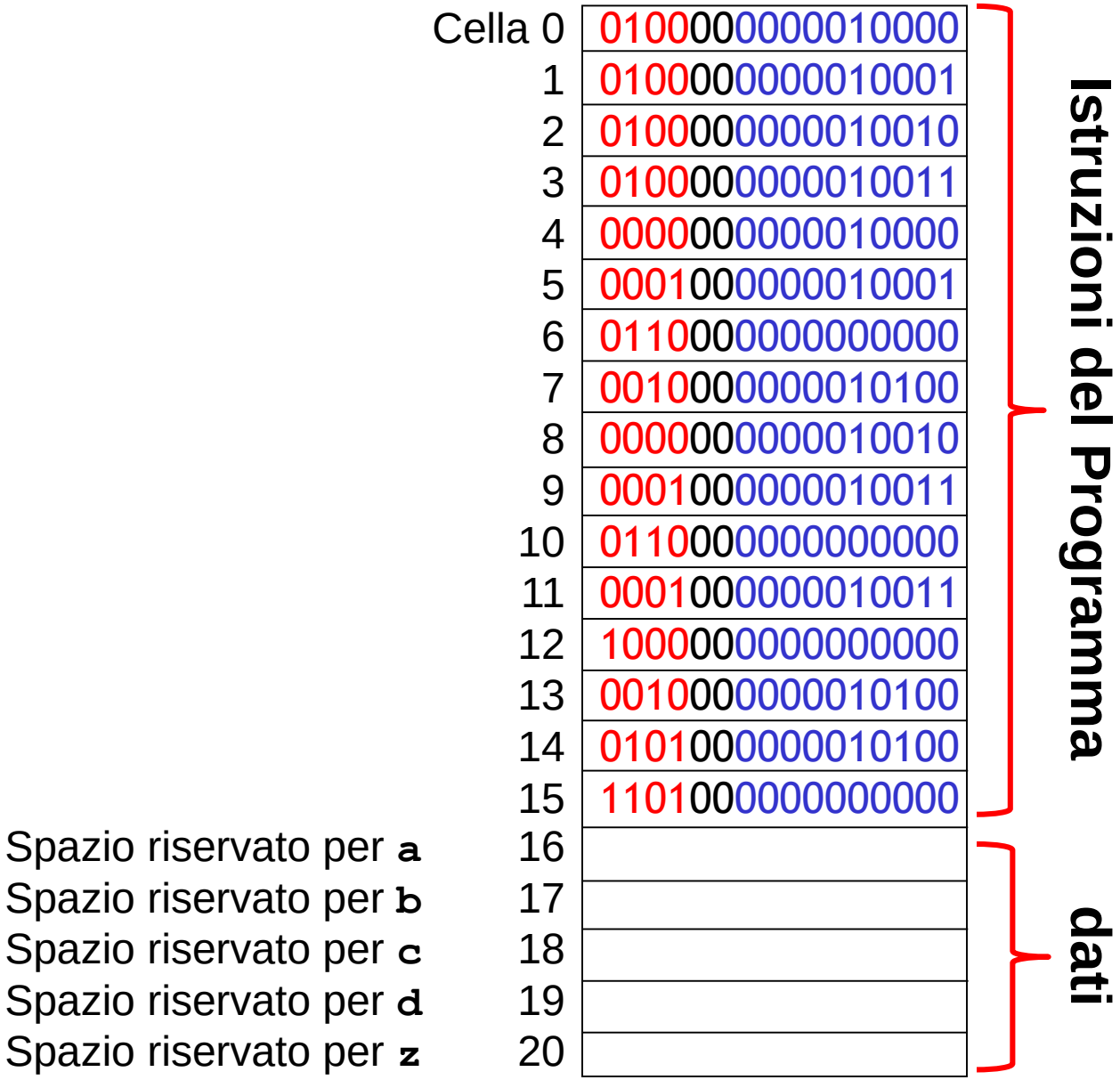
6. Scrivi sul dispositivo in uscita
   6.1 Memorizza il risultato calcolato (disponibile nel registro A) nella cella di memoria riservata a **z**
   6.2 Copia il contenuto della cella di memoria riservata a **z** nel registro dati della periferica di uscita

Esempio: più nel dettaglio

6. Scrivi sul dispositivo di uscita il risultato della valutazione complessiva
7. Termina l'esecuzione del programma

6. Scrivi sul dispositivo in uscita
   6.1 Memorizza il risultato calcolato (disponibile nel registro A) nella cella di memoria riservata a **z**
   6.2 Copia il contenuto della cella di memoria riservata a **z** nel registro dati della periferica di uscita
7. Manda il comando di Halt

Programma in Memoria Centrale



Forma Binaria del Programma

0100000000010000	**Leggi** un valore dall'input e mettilo nella cella 16 (`a`)
0100000000010001	**Leggi** un valore dall'input e mettilo nella cella 17 (`b`)
0100000000010010	**Leggi** un valore dall'input e mettilo nella cella 18 (`c`)
0100000000010011	**Leggi** un valore dall'input e mettilo nella cella 19 (`d`)
0000000000010000	**Carica** il contenuto della cella 16 (`a`) **nel registro A**
0001000000010001	**Carica** il contenuto della cella 17 (`b`) **nel registro B**
0110000000000000	**Somma** i registri A e B
0010000000010100	**Scarica** il contenuto di A nella cella 20 (`z`) (ris.parziale)
0000000000010010	**Carica** il contenito della cella 18 (`c`) **nel registro A**
0001000000010011	**Carica** il contenito della cella 19 (`d`) **nel registro B**
0110000000000000	**Somma** i registri A e B
0001000000010100	**Carica** il contenuto della cella 20 (`z`) (ris. parziale) **in B**
1000000000000000	**Moltiplica** i registri A e B
0010000000010100	**Scarica** il contenuto di A nella cella 20 (`z`) (ris. totale)
0101000000010100	**Scrivi** il contenuto della cella 20 (`z`) (ris. totale) output
1101000000000000	**Halt**

Forma Binaria del Programma

0100000000010000	**Leggi** un valore dall'input e mettilo nella cella 16 (**a**)
0100000000010001	**Leggi** un valore dall'input e mettilo nella cella 17 (**b**)
0100000000010010	**Leggi** un valore dall'input e mettilo nella cella 18 (**c**)
0100000000010011	**Leggi** un valore dall'input e mettilo nella cella 19 (**d**)
0000000000010000	**Carica** il contenuto della cella 16 (**a**) **nel registro A**
0001000000010001	**Carica** il contenuto della cella 17 (**b**) **nel registro B**
0110000000000000	**Somma** i registri A e B
0010000000010100	**Scarica** il contenuto di A nella cella 20 (**z**) (ris.parziale)
0000000000010010	**Carica** il contenito della cella 18 (**c**) **nel registro A**
0001000000010011	**Carica** il contenito della cella 19 (**d**) **nel registro B**
0110000000000000	**Somma** i registri A e B
0001000000010100	**Carica** il contenuto della cella 20 (**z**) (ris. parziale) **in B**
1000000000000000	**Moltiplica** i registri A e B
0010000000010100	**Scarica** il contenuto di A nella cella 20 (**z**) (ris. totale)
0101000000010100	**Scrivi** il contenuto della cella 20 (**z**) (ris. totale) output
1101000000000000	**Halt**

Forma Binaria del Programma

0100000000010000	**Leggi** un valore dall'input e mettilo nella cella 16 (`a`)
0100000000010001	**Leggi** un valore dall'input e mettilo nella cella 17 (`b`)
0100000000010010	**Leggi** un valore dall'input e mettilo nella cella 18 (`c`)
0100000000010011	**Leggi** un valore dall'input e mettilo nella cella 19 (`d`)
0000000000010000	**Carica** il contenuto della cella 16 (`a`) **nel registro A**
0001000000010001	**Carica** il contenuto della cella 17 (`b`) **nel registro B**
0110000000000000	**Somma** i registri A e B
0010000000010100	**Scarica** il contenuto di A nella cella 20 (`z`) (ris.parziale)
0000000000010010	**Carica** il contenito della cella 18 (`c`) **nel registro A**
0001000000010011	**Carica** il contenito della cella 19 (`d`) **nel registro B**
0110000000000000	**Somma** i registri A e B
0001000000010100	**Carica** il contenuto della cella 20 (`z`) (ris. parziale) **in B**
1000000000000000	**Moltiplica** i registri A e B
0010000000010100	**Scarica** il contenuto di A nella cella 20 (`z`) (ris. totale)
0101000000010100	**Scrivi** il contenuto della cella 20 (`z`) (ris. totale) output
1101000000000000	**Halt**

Forma Binaria del Programma

0100000000010000	**Leggi** un valore dall'input e mettilo nella cella 16 (**a**)
0100000000010001	**Leggi** un valore dall'input e mettilo nella cella 17 (**b**)
0100000000010010	**Leggi** un valore dall'input e mettilo nella cella 18 (**c**)
0100000000010011	**Leggi** un valore dall'input e mettilo nella cella 19 (**d**)
0000000000010000	**Carica** il contenuto della cella 16 (**a**) **nel registro A**
0001000000010001	**Carica** il contenuto della cella 17 (**b**) **nel registro B**
0110000000000000	**Somma** i registri A e B
0010000000010100	**Scarica** il contenuto di A nella cella 20 (**z**) (ris.parziale)
0000000000010010	**Carica** il contenito della cella 18 (**c**) **nel registro A**
0001000000010011	**Carica** il contenito della cella 19 (**d**) **nel registro B**
0110000000000000	**Somma** i registri A e B
0001000000010100	**Carica** il contenuto della cella 20 (**z**) (ris. parziale) **in B**
1000000000000000	**Moltiplica** i registri A e B
0010000000010100	**Scarica** il contenuto di A nella cella 20 (**z**) (ris. totale)
0101000000010100	**Scrivi** il contenuto della cella 20 (**z**) (ris. totale) output
1101000000000000	**Halt**

Forma Binaria del Programma

0100000000010000	**Leggi** un valore dall'input e mettilo nella cella 16 (`a`)
0100000000010001	**Leggi** un valore dall'input e mettilo nella cella 17 (`b`)
0100000000010010	**Leggi** un valore dall'input e mettilo nella cella 18 (`c`)
0100000000010011	**Leggi** un valore dall'input e mettilo nella cella 19 (`d`)
0000000000010000	**Carica** il contenuto della cella 16 (`a`) **nel registro A**
0001000000010001	**Carica** il contenuto della cella 17 (`b`) **nel registro B**
0110000000000000	**Somma** i registri A e B
0010000000010100	**Scarica** il contenuto di A nella cella 20 (`z`) (ris.parziale)
0000000000010010	**Carica** il contenito della cella 18 (`c`) **nel registro A**
0001000000010011	**Carica** il contenito della cella 19 (`d`) **nel registro B**
0110000000000000	**Somma** i registri A e B
0001000000010100	**Carica** il contenuto della cella 20 (`z`) (ris. parziale) **in B**
1000000000000000	**Moltiplica** i registri A e B
0010000000010100	**Scarica** il contenuto di A nella cella 20 (`z`) (ris. totale)
0101000000010100	**Scrivi** il contenuto della cella 20 (`z`) (ris. totale) output
1101000000000000	**Halt**

Forma Binaria del Programma

0100000000010000	**Leggi** un valore dall'input e mettilo nella cella 16 (`a`)
0100000000010001	**Leggi** un valore dall'input e mettilo nella cella 17 (`b`)
0100000000010010	**Leggi** un valore dall'input e mettilo nella cella 18 (`c`)
0100000000010011	**Leggi** un valore dall'input e mettilo nella cella 19 (`d`)
0000000000010000	**Carica** il contenuto della cella 16 (`a`) **nel registro A**
0001000000010001	**Carica** il contenuto della cella 17 (`b`) **nel registro B**
0110000000000000	**Somma** i registri A e B
0010000000010100	**Scarica** il contenuto di A nella cella 20 (`z`) (ris.parziale)
0000000000010010	**Carica** il contenito della cella 18 (`c`) **nel registro A**
0001000000010011	**Carica** il contenito della cella 19 (`d`) **nel registro B**
0110000000000000	**Somma** i registri A e B
0001000000010100	**Carica** il contenuto della cella 20 (`z`) (ris. parziale) **in B**
1000000000000000	**Moltiplica** i registri A e B
0010000000010100	**Scarica** il contenuto di A nella cella 20 (`z`) (ris. totale)
0101000000010100	**Scrivi** il contenuto della cella 20 (`z`) (ris. totale) output
1101000000000000	**Halt**

Forma Binaria del Programma

0100000000010000	**Leggi** un valore dall'input e mettilo nella cella 16 (`a`)
0100000000010001	**Leggi** un valore dall'input e mettilo nella cella 17 (`b`)
0100000000010010	**Leggi** un valore dall'input e mettilo nella cella 18 (`c`)
0100000000010011	**Leggi** un valore dall'input e mettilo nella cella 19 (`d`)
0000000000010000	**Carica** il contenuto della cella 16 (`a`) **nel registro A**
0001000000010001	**Carica** il contenuto della cella 17 (`b`) **nel registro B**
0110000000000000	**Somma** i registri A e B
0010000000010100	**Scarica** il contenuto di A nella cella 20 (`z`) (ris.parziale)
0000000000010010	**Carica** il contenito della cella 18 (`c`) **nel registro A**
0001000000010011	**Carica** il contenito della cella 19 (`d`) **nel registro B**
0110000000000000	**Somma** i registri A e B
0001000000010100	**Carica** il contenuto della cella 20 (`z`) (ris. parziale) **in B**
1000000000000000	**Moltiplica** i registri A e B
0010000000010100	**Scarica** il contenuto di A nella cella 20 (`z`) (ris. totale)
0101000000010100	**Scrivi** il contenuto della cella 20 (`z`) (ris. totale) output
1101000000000000	**Halt**

Forma Binaria del Programma

0100000000010000	**Leggi** un valore dall'input e mettilo nella cella 16 (**a**)
0100000000010001	**Leggi** un valore dall'input e mettilo nella cella 17 (**b**)
0100000000010010	**Leggi** un valore dall'input e mettilo nella cella 18 (**c**)
0100000000010011	**Leggi** un valore dall'input e mettilo nella cella 19 (**d**)
0000000000010000	**Carica** il contenuto della cella 16 (**a**) **nel registro A**
0001000000010001	**Carica** il contenuto della cella 17 (**b**) **nel registro B**
0110000000000000	**Somma** i registri A e B
0010000000010100	**Scarica** il contenuto di A nella cella 20 (**z**) (ris.parziale)
0000000000010010	**Carica** il contenito della cella 18 (**c**) **nel registro A**
0001000000010011	**Carica** il contenito della cella 19 (**d**) **nel registro B**
0110000000000000	**Somma** i registri A e B
0001000000010100	**Carica** il contenuto della cella 20 (**z**) (ris. parziale) **in B**
1000000000000000	**Moltiplica** i registri A e B
0010000000010100	**Scarica** il contenuto di A nella cella 20 (**z**) (ris. totale)
0101000000010100	**Scrivi** il contenuto della cella 20 (**z**) (ris. totale) output
1101000000000000	**Halt**