Introduzione al Corso

Informatica B, a.a. 2021/2022

Francesco Trovò

francesco1.trovo@polimi.it

Siete nel Posto Giusto?

Prof. Terraneo ha lo scaglione E - LEM

Prof. Trovò ha lo scaglione LEM - P

Prof. Masseroli ha lo scaglione P - SAM

La suddivisione degli studenti nelle varie sezioni potrebbe cambiare leggermente nei prossimi giorni

Chi Siamo

Chi Eravamo

Giacomo Boracchi (docente)

Francesco Trovò (esercitazione)

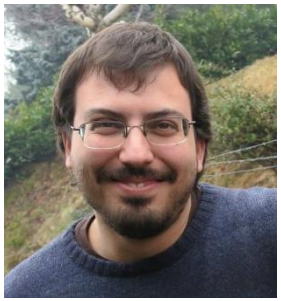
Marco D. Santambrogio (responsabile di laboratorio)

Diego Carrera (responsabile di laboratorio)

Giacomo Boracchi

(2012-2018)

Chi Siamo



Francesco Trovò (docente)

- Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano
- Homepage: https://trovo.faculty.polimi.it/
- E-mail francesco1.trovo@polimi.it
- Ricevimento studenti: online, su **appuntamento mandando un'e-mail**
- Ufficio: Edificio 21, Primo Piano, Ufficio 013, via Ponzio 34/5, Milano
- Tel 02 2399 3491

Chi Siamo

Dr. Mirco Mutti (esercitatore)

- Homepage: https://trovo.faculty.polimi.it/
- E-mail: mirco.mutti@polimi.it

Chi Siamo

Dr. Nicolò Folloni (responsabile di laboratorio)

Dr. Loris Giulivi (responsabile di laboratorio)

Il Corso

Organizzazione

Lezioni: 29 ore

Venerdì 8.30-11.00 online su webex

Esercitazioni: 26 ore

Martedì 13.30-15.00 in aula L.12 (Codice persona DISPARI)

Martedì 15.30-17.00 in aula L.12 (Codice persona PARI)

Laboratori: 15 ore (6 incontri)

Martedì dalle 8.30 alle 11.00 online su webex

I due laboratori introduttivi saranno in presenza

Potrebbero esserci degli scambi, quindi **controllate sempre** il calendario del corso:

https://trovo.faculty.polimi.it/infob_2021_2022.html

Argomenti Trattati

- Introduzione all'informatica
- Codifica binaria e algebra di Boole
- Composizione e organizzazione dei sistemi informatici
- Introduzione alla programmazione
  - Fondamenti della programmazione in C
  - Fondamenti della programmazione in Matlab
  - Argomenti avanzati di programmazione in Matlab
- Introduzione ai sistemi operativi

Cosa Imparerete

- Gli **elementi fondamentali** ed i **principi** che regolano il funzionamento di un **sistema informatico**
- Come **sviluppare algoritmi** per risolvere problemi
- Come **codificare** tali **algoritmi** in programmi che ne permettano l'automatizzazione
- Le basi della **programmazione**
- L'utilizzo dei linguaggi **C** e **Matlab**
- Alcune nozioni di base sui **sistemi operativi** e sulla **codifica binaria**

Bibliografia

Per C e argomenti teorici:

- S. Ceri, D. Mandrioli, L. Sbattella «*Informatica arte e mestiere*». McGraw-Hill, 2004. ISBN: 978-8838668487
- G. Boracchi, E. Di Nitto, D. Loiacono, M. Masseroli, M.D. Santambrogio, V. Zaccaria, F. Fummi «*Materiale su sistemi informatici e i principi di programmazione in C per il corso di Informatica B*», Mc Graw Hill, 2016. ISBN: 978-1308911731

Per Matlab:

- A. Campi, E. Di Nitto, D. Loiacono, A. Morzenti, B. Spoletini, «*Introduzione alla programmazione in Matlab*», Esculapio, 2011. ISBN: 978-8874884629

La Pagina del Corso

La pagina del corso è

https://trovo.faculty.polimi.it/

Troverete:

- Materiale didattico usato a lezione ed esercitazione
- Link al sito dei laboratori
- Tutorial per l'installazione dei programmi utilizzati
- Temi d'esame con soluzioni
- Calendario del corso (lezioni, esercitazioni, laboratori)
- Avvisi, esiti esami/prove intermedia

I Laboratori

- Nei laboratori è richiesto di **sviluppare autonomamente** gli elaborati
- Sarete assistiti (da remoto) da:
  - due responsabili di laboratorio e tre tutor
- Il Laboratorio è **necessario** per:
  - prendere familiarità con l'ambiente di sviluppo
  - consolidare la conoscenza dei linguaggi, dei metodi e degli strumenti introdotti a lezione
  - complementare lo studio e gli esercizi fatti su carta
- Il laboratorio non è sufficiente per saper programmare:
  - esercitarsi sul vostro pc sui programmi di esercitazione e laboratorio

I Laboratori

È possibile utilizzare il **proprio laptop**:

- Installare Code::Blocks per il C, versione con compilatore (http://www.codeblocks.org/)
- Installare Matlab (per il secondo emisemestre)
- Sul sito del laboratorio troverete tutte le istruzioni per assistervi nell'installazione

Il **primo laboratorio** si terrà in online Martedì 5 Ottobre!

Esame e Ricevimento Studenti

Modalità di Esame

Solo appelli regolari (niente compitini):

- esame scritto su tutto il programma
- **occorre un punteggio minimo in C && Matlab**
- l'orale è a discrezione del docente
- il laboratorio non sarà valutato
- potrebbero esserci delle competition che danno un punto addizionale per il voto finale

Modalità di Esame

Sul sito trovate **temi d'esame (TDE) svolti** degli anni precedenti:

- i temi d'esame più simili a quelli che verranno proposti sono quelli dell'a.a. 2017/2018 fino ad oggi
- **durante il TDE non sarà possibile consultare libri ed appunti**
- negli anni precedenti era possibile consultare gli appunti e quindi gli esercizi potrebbero variare per quanto riguarda il livello di difficoltà e il tempo richiesto per svolgerli
- forniremo con il TDE un CheatSheet per la programmazione in C ed in Matlab

CheatSheet per C

C Reference Card (ANSI) - versione semplificata
corso Informatica B Politecnico di Milano

1 Struttura del programma

struttura del programma principale

```
int main(void) {
  declarations
  statements
}
```

Dichiarazione di variabile: `type name;`

2 Comandi per il preprocessore

inclusione di un file di libreria `#include <filename>`
definizione di costante `#define name value`

3 Dichiarazioni

carattere (1 byte)	`char`
intero	`int`
short (16 bit integer)	`short`
long (32 bit integer)	`long`
numero senza segno	`unsigned`
constante	`type const name;`
numero reale	`float, double`
type enumerativo	`typedef enum name1, name2, ... typeName;`
struct	`typedef struct ... typeName;`
ridefinizione di tipo	`typedef oldTypeName newTypeName;`
tipo array	`typedef elementType arrayType[dim];`
tipo matrice	`typedef elementType matrixType[dim1][dim2];`

4 Operatori

accesso al campo di una struct	`structName.field`
accesso all'i-esimo elemento di un array	`arrayName[i]`
operazioni aritmetiche per interi	`+, -, *, /, %, ++, --`
operazioni aritmetiche per reali	`+, -, *, /`
operatori di confronto	`==, !=, >, <, >=, <=`
operatore di assegnamento	`=`
operatori logici	`&&, ||, !`

5 Flusso di controllo

terminatore di istruzione	`;`
delimitatore blocco	`{ ... }`
istruzione if	`if (espr) istruzione1 else istruzione2`
istruzione while	`while (espr) istruzione`
istruzione for	`for (istr1; espr; istr3) istr2`
istruzione do	`do istruzione while (espr);`
istruzione switch	`switch (espr) {case val1: istr1; break; ... default istr}`

6 Libreria ANSI C

`<assert.h> <ctype.h> <errno.h> <float.h> <limits.h> <locale.h> <math.h> <setjmp.h> <signal.h> <stdarg.h> <stddef.h> <stdio.h> <stdlib.h> <string.h> <time.h>`

7 Input/Output `<stdio.h>`

stampa formattata	`printf(''formato'', arg1, arg2, ...);`
acquisizione formattata	`scanf(''formato'', arg1, arg2, ...);`

8 Operazioni tra stringhe `<string.h>`

lunghezza di s	`strlen(s)`
copia di s1 in s2	`strcpy(s1, s2)`
concatenazione di s2 dopo s1	`strcat(s1, s2)`
confronto tra s1 e s2	`strcmp(s1, s2)` - restituisce 0 se uguali, < 0 se s1 precede s2, > 0 altrimenti

9 Mathematical Functions `<math.h>`

funzioni trigonometriche	`sin(x), cos(x), tan(x)`
funzioni trigonometriche inverse	`asin(x), acos(x), atan(x)`
arctan(y/x)	`atan2(y,x)`
funzioni trigonometriche iperboliche	`sinh(x), cosh(x), tanh(x)`
esponenziali & logaritmi	`exp(x), log(x), log10(x)`
elevamento a potenza	`pow(x,y), sqrt(x)`

Ricevimento Studenti

- Domande e richieste di chiarimenti sono sempre ben accette
- Potete anche rivolgere domande via e-mail

Domanda tde

Inviata da Polimi × polimi.it ×

[redacted]

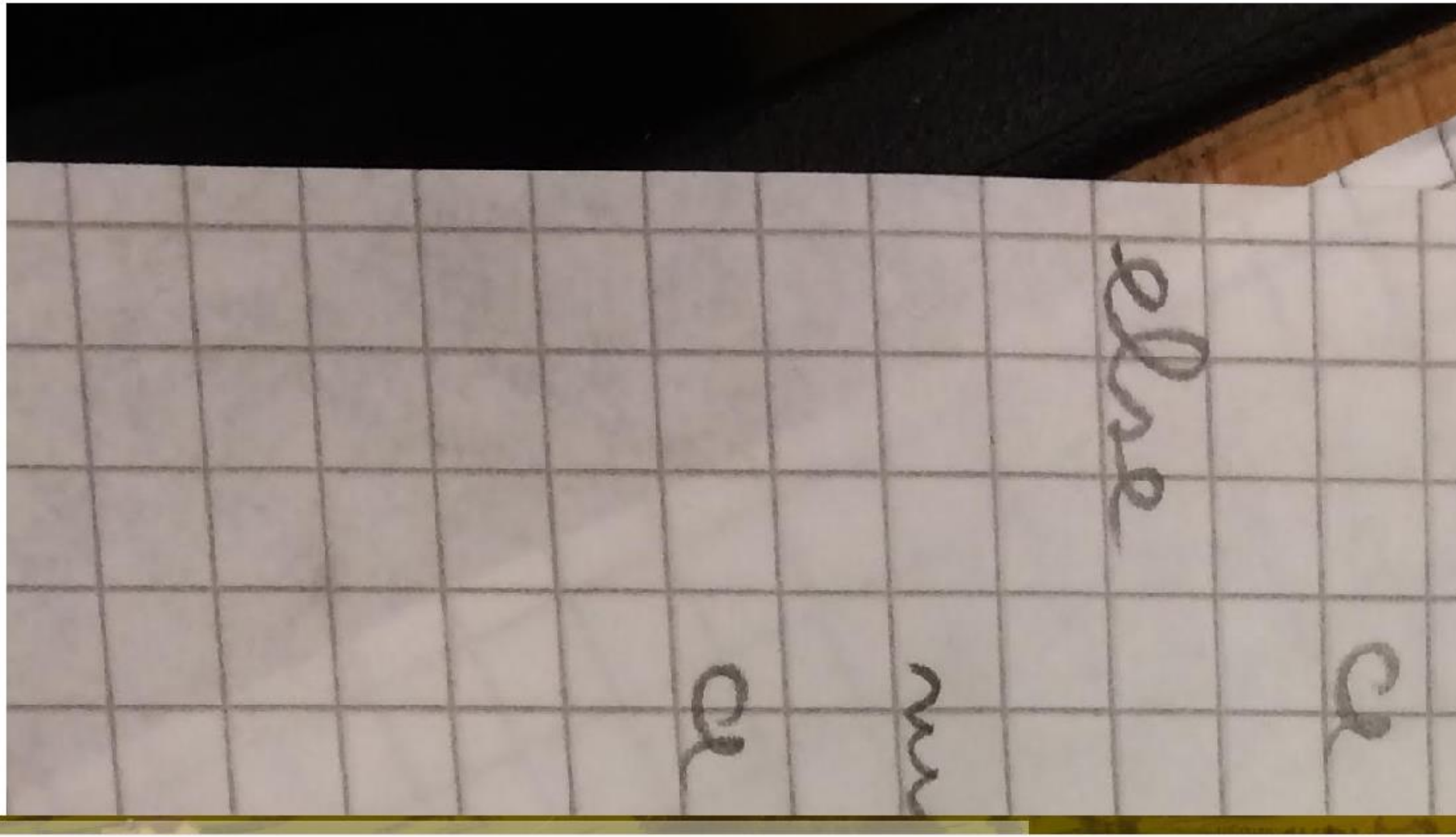
to Francesco, Giacomo

Buona sera,

È corretta questa soluzione per il punto 2 dell'esercizio 2 del tde 28/01/13?

Cordiali saluti

[redacted]



Esercizio 3 (4 punti)

Dati i seguenti due numeri in codifica IEEE 754 (virgola mobile, il bit più a sinistra è ovviamente il bit di segno)

A = S:0 E:01111111 M:00100000000000000000000

B = S:1 E:01111100 M:00100000000000000000000

Calcolare a quanto equivale la divisione A/B (in decimale).

Soluzione

I numeri A e B si differenziano solo per esponente e segno. Si può quindi calcolare la divisione A/B prendendo in considerazione solo gli esponenti:

```
A = - B * 2^3
```

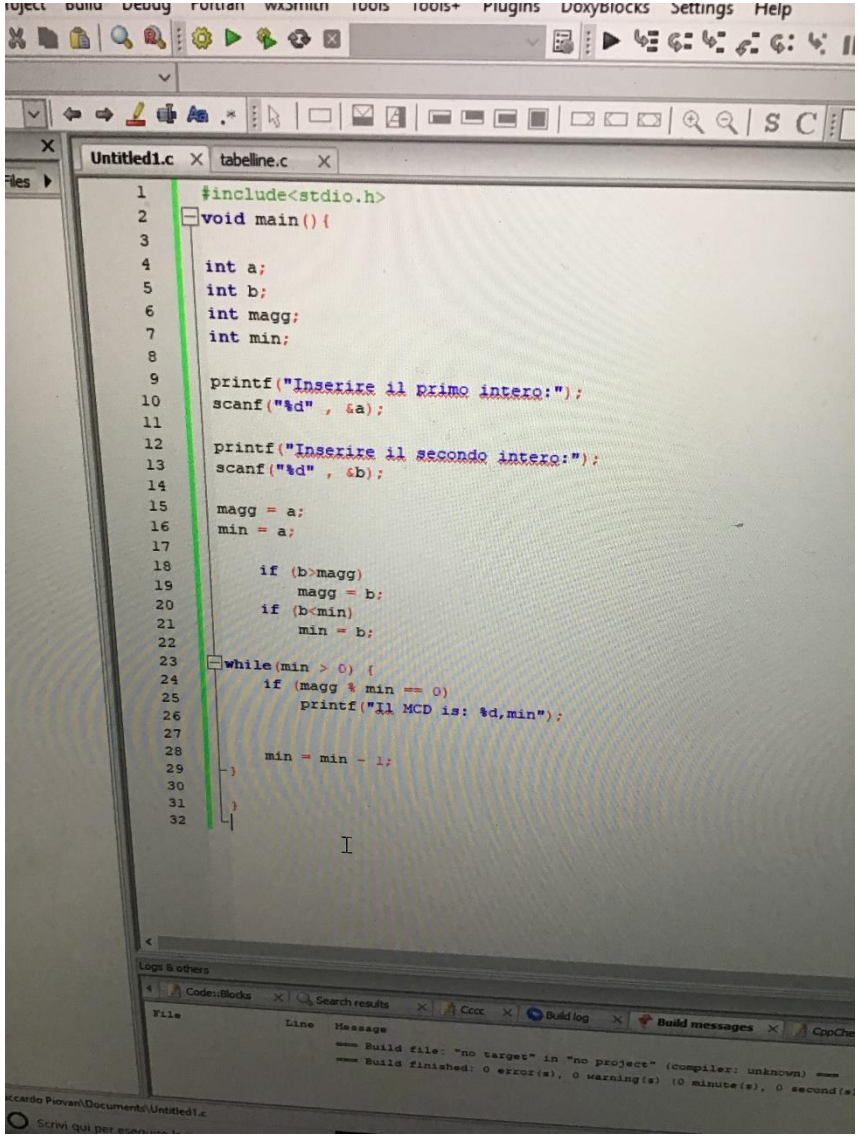
Quindi:

```
A/B = - 8
```

Si può anche notare che:

```
A =      (1 + 2^(-3))           =   1.125
B = -1 * (1 + 2^(-3)) * 2^(-3)  = - 0.140625
```

No alle Foto al Codice!



Ricevimento Studenti

- Domande e richieste di chiarimenti sono sempre ben accette
- Potete anche rivolgere domande via e-mail

In particolare:

- dite chi siete
- scrivete dalla mail ufficiale del poli
- inviate solo codici (C o Matlab) in file sorgenti
- dite chiaramente qual è il vostro problema e perché l'esercizio non funziona
- riportate il testo della domanda

Cos’è l’informatica?

Cos'è l'Informatica?

Scienza della rappresentazione e dell'elaborazione dell'informazione

- ***Scienza:*** ovvero una **conoscenza sistematica** e **rigorosa** di tecniche e metodi
- ***Informazione:*** l'oggetto dell'investigazione scientifica (informazione intesa sia come entità astratta, sia come tecnologie per la sua gestione)
- ***Rappresentazione:*** il modo in cui l'informazione viene strutturata e trasformata in dati fruibili da macchine
- ***Elaborazione:*** uso e trasformazione dell'informazione per un dato scopo. L'elaborazione deve poter essere eseguita da macchine che processano dati

Cos'è l'Informatica?

Studio sistematico degli algoritmi che descrivono e trasformano l'informazione: la loro teoria, analisi, progetto, efficienza, realizzazione, applicazione

Non viene mai citato il Calcolatore (è uno strumento):

- prima progetto il metodo con cui voglio elaborare i dati
- dopo utilizzo il calcolatore per gestire grandi quantità di dati

Gli Algoritmi

Algoritmo

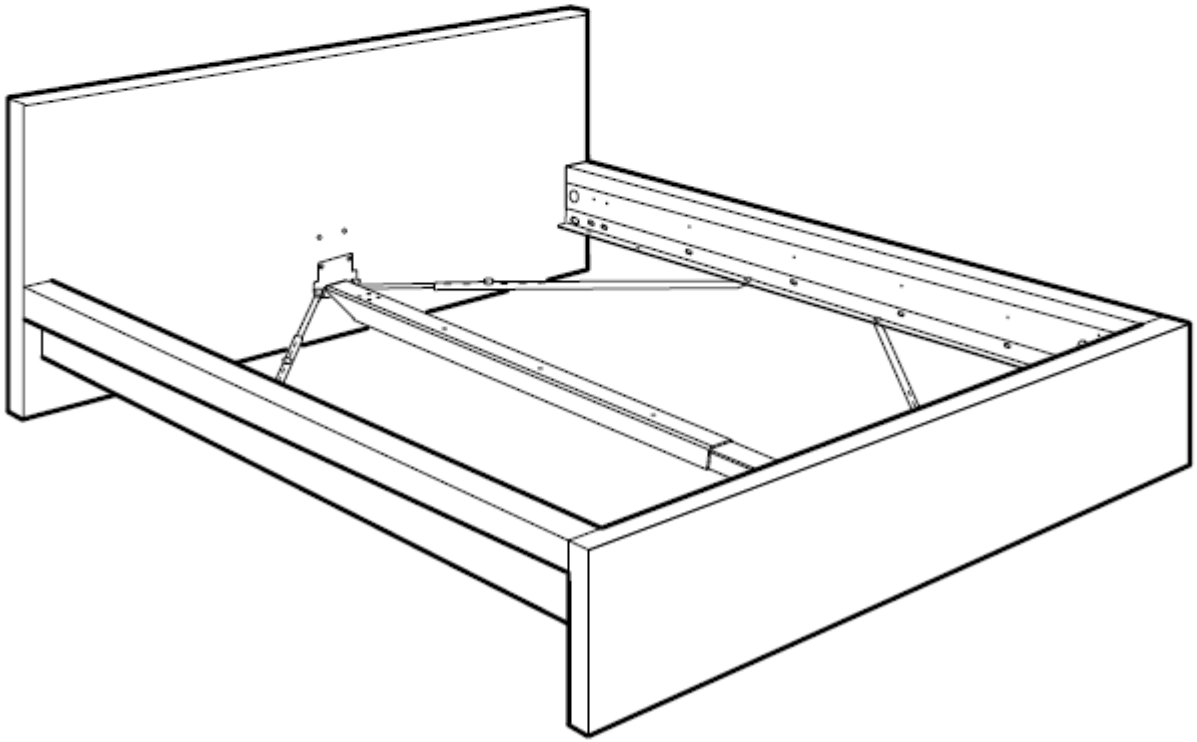
Sequenza *precisa di* ***operazioni****, definiti con* ***precisione****, che portano alla* ***realizzazione di un compito***

Le operazioni devono:

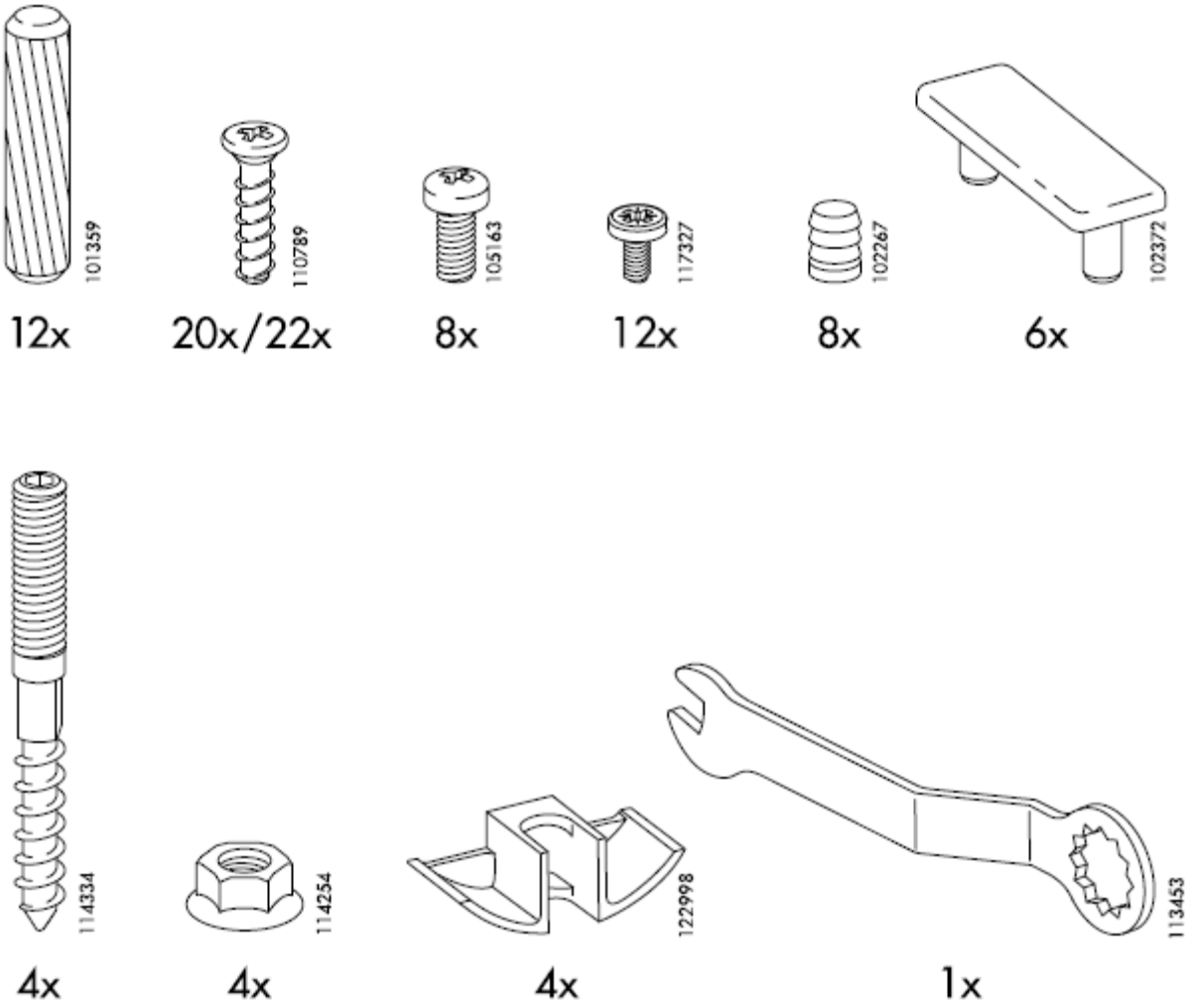
- essere **comprensibili,** senza ambiguità
- essere **eseguibili** da uno strumento automatico: l'esecutore
- portare a realizzare un compito in **tempo finito** (devono contenere un numero finito di passi, ciascuno eseguibile in tempo finito)

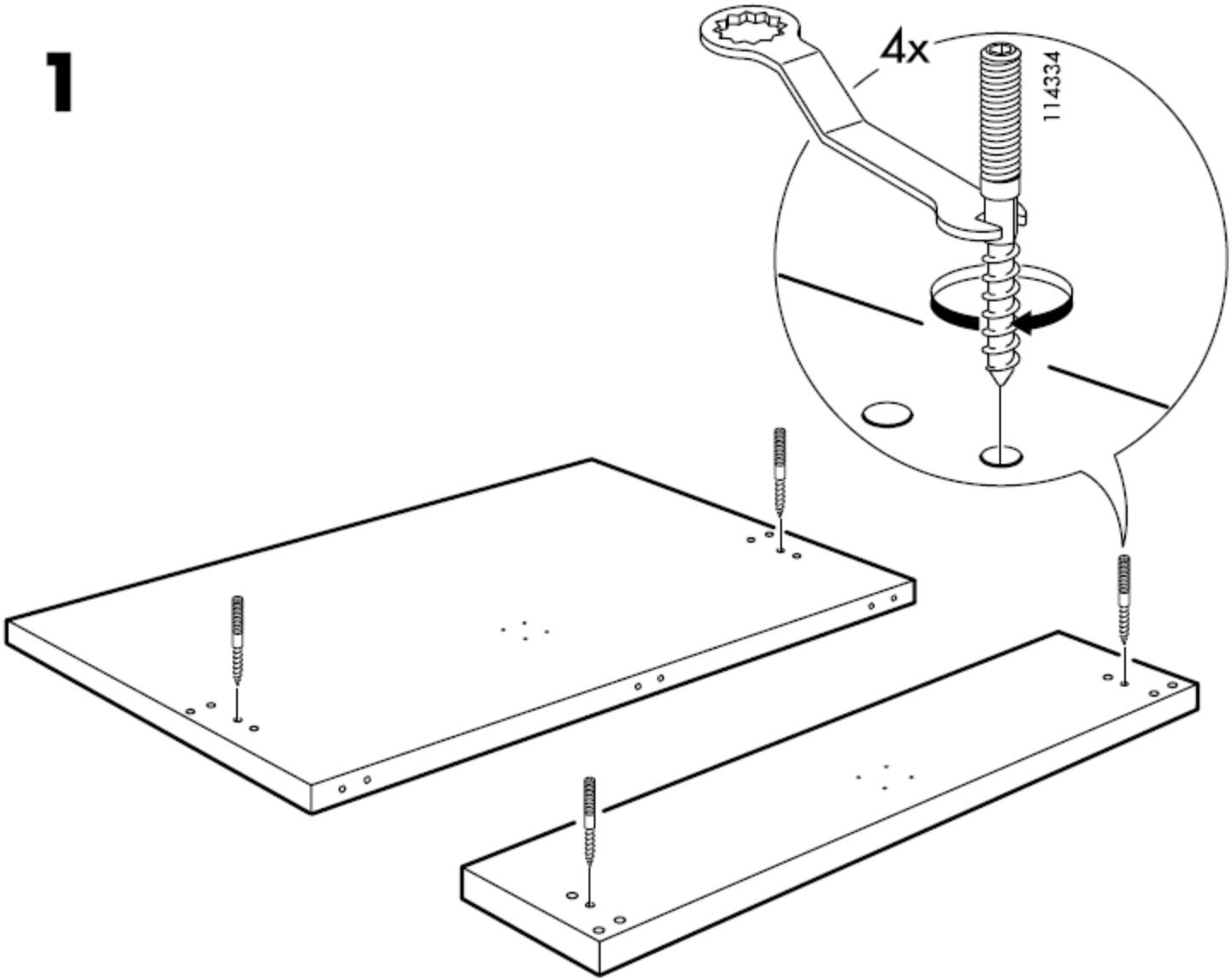
…vedermo ora un esempio di algoritmo in cui vi sarà capitato di essere esecutori

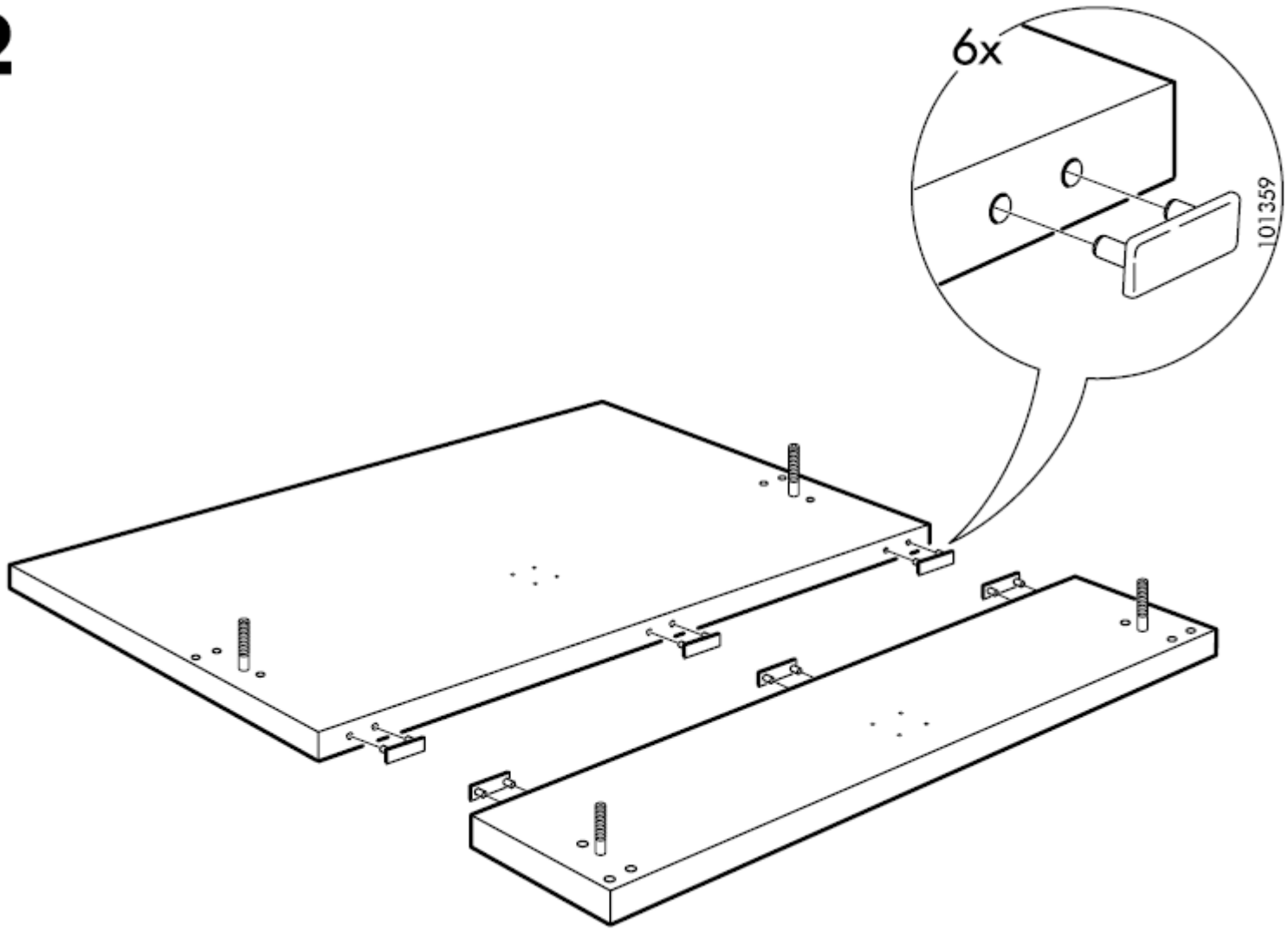
MALM



Design and Quality
IKEA of Sweden


101359
110789
105163
117327
102267
102372
12x
20x/22x
8x
12x
8x
6x
114334
114254
122998
113453
4x
4x
4x
1x


1
4x
114334


2
6x
101359

Il Linguaggio «IKEA»

- Le istruzioni IKEA sono fatte per esecutori intelligenti l'interpretazione dei disegni richiede diverse capacità
- Quando l'esecutore è meno intelligente occorre esprimere le istruzioni in un linguaggio più preciso

Un Nostro Linguaggio per gli Algoritmi

Svilupperemo i nostri primi algoritmi in italiano (utilizzando un linguaggio molto essenziale)

In particolare, il linguaggio sarà caratterizzato da:

- sequenzialità delle istruzioni
- costrutto condizionale
- costrutto iterativo

Inoltre, potremo utilizzare «foglietti» dove scrivere (registrare) dei valori

Durante le esercitazioni per progettare gli algoritmi utilizzerete un primo linguaggio intermedio per poi convertirlo in linguaggio di programmazione vero e proprio

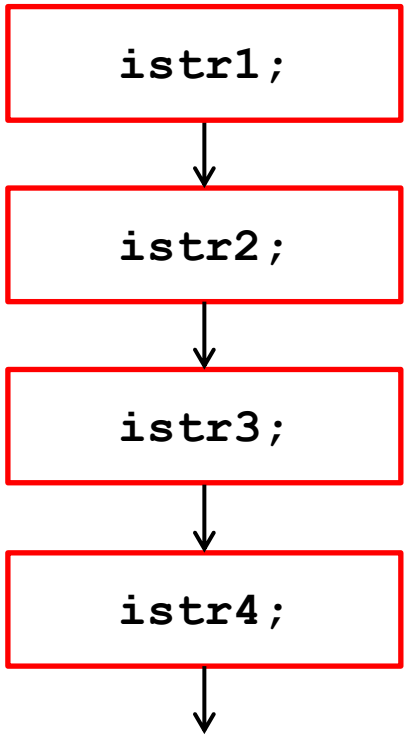
La Sequenzialità

Esempio: Algoritmo per Andare in Università

- Mi alzo quando suona la sveglia
- Esco
- Mi dirigo verso la cucina
- Indosso la mascherina
- Mi lavo
- Mi vesto
- Bevo un caffè
- Corro a prendere il treno

Manca una coerenza temporale delle istruzioni

La Sequenzialità

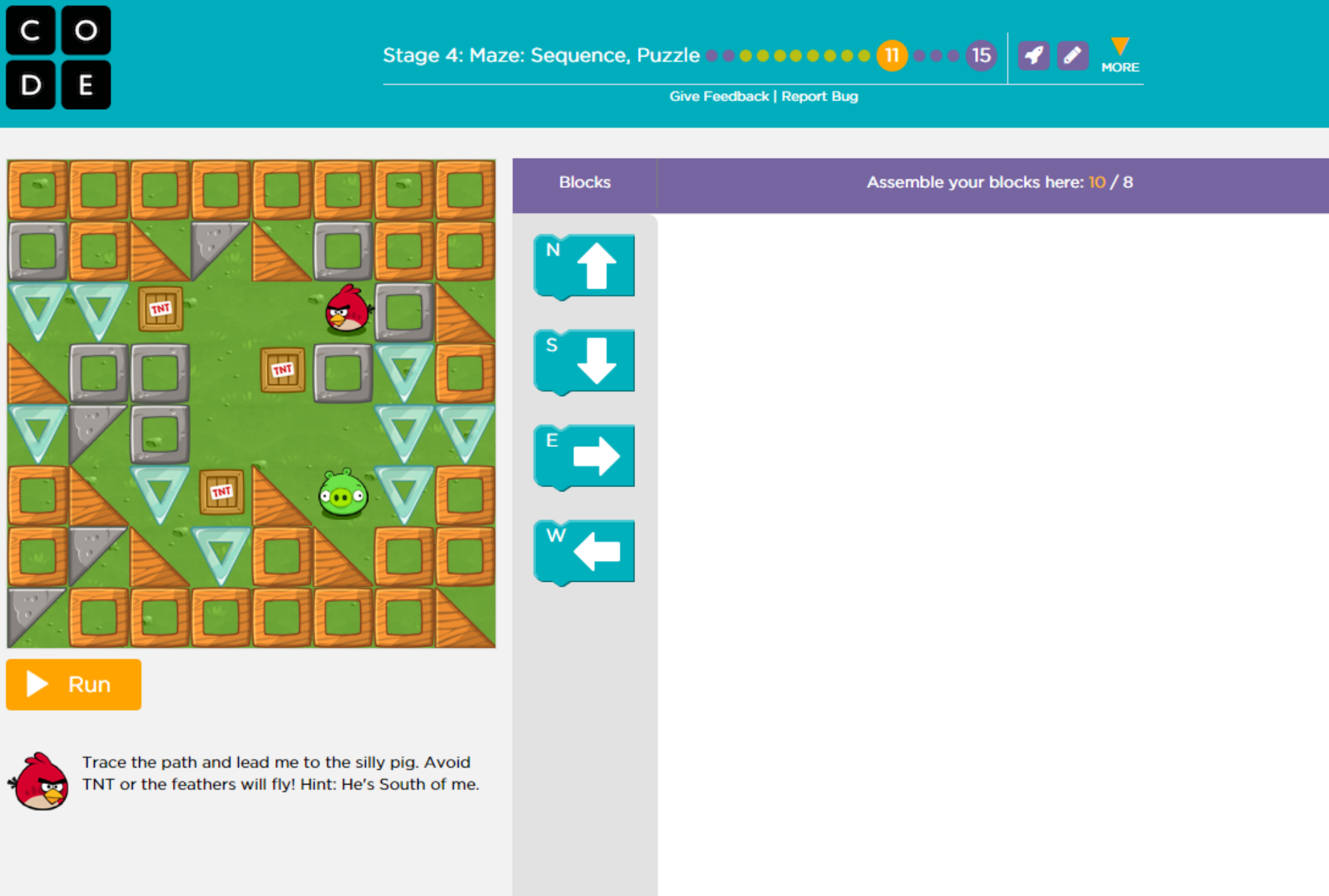



```
istr1;
istr2;
istr3;
istr4;
…
```

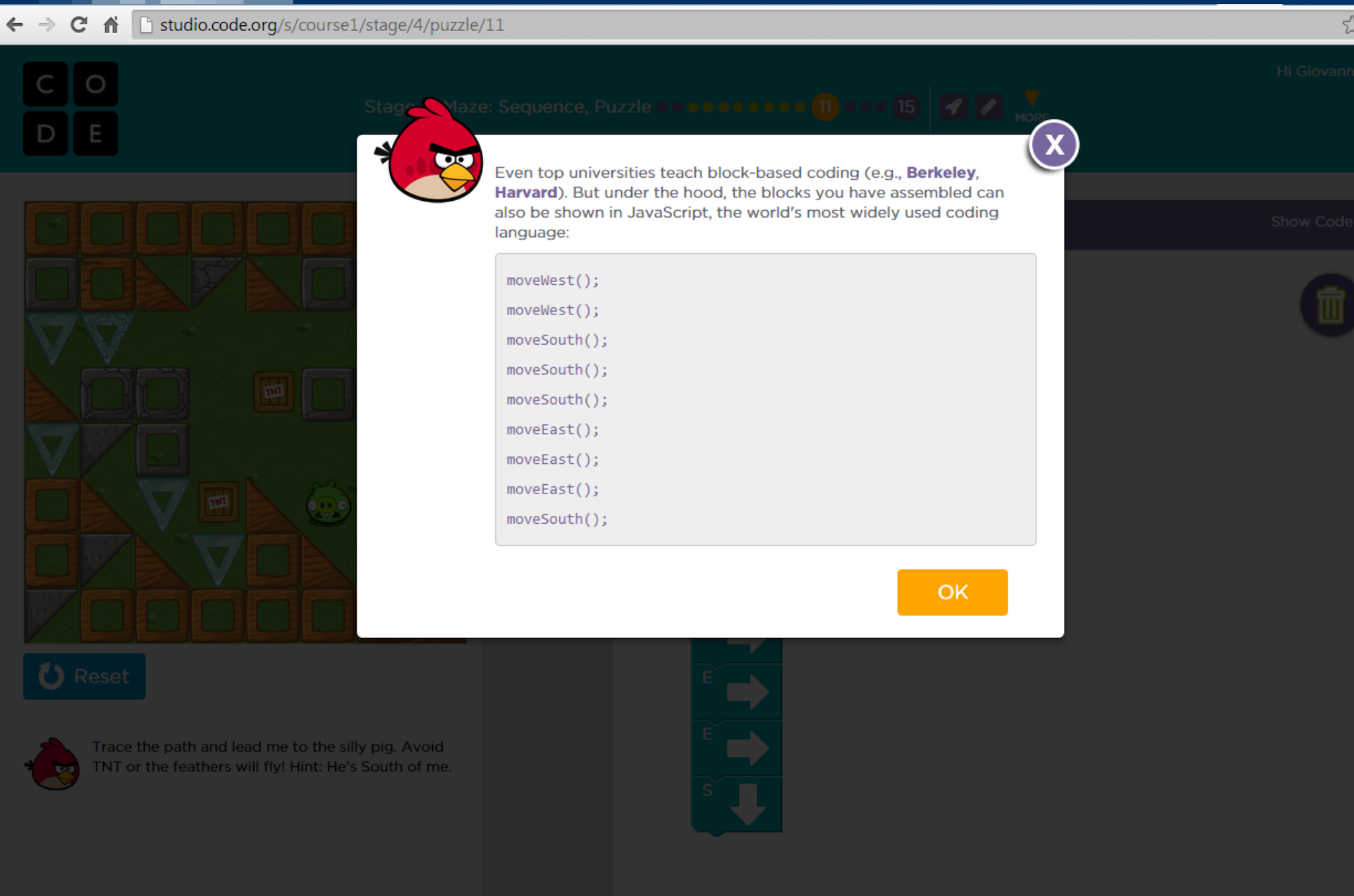
Le istruzioni vengono eseguite dalla prima all'ultima

Terminata la *i-sima* istruzione, si esegue la *(i+1)-sima*

La Sequenzialità per i Bimbi su code.org



La Sequenzialità per i Bimbi su code.org



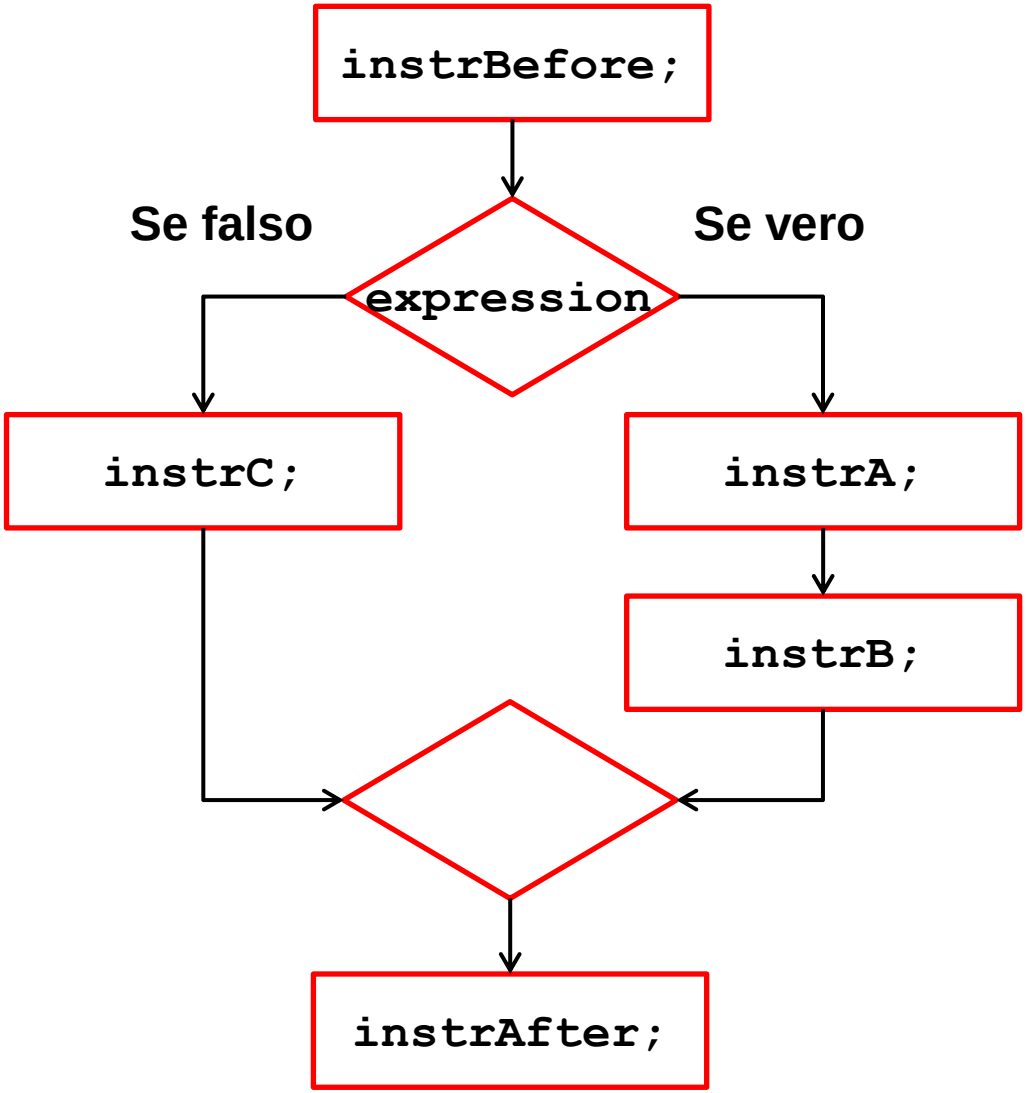
Costrutto Condizionale

Esempio: Algoritmo per Andare in Università

- Mi alzo quando suona la sveglia
- Mi dirigo verso la cucina
- Mangio e bevo un caffè
- Mi lavo e mi vesto
- Indosso la mascherina
- **Se** devo mangiare fuori
  - prendo il pranzo

  Altrimenti
  - prendo uno snack per metà mattina
- Esco
- Corro

Costrutto Condizionale

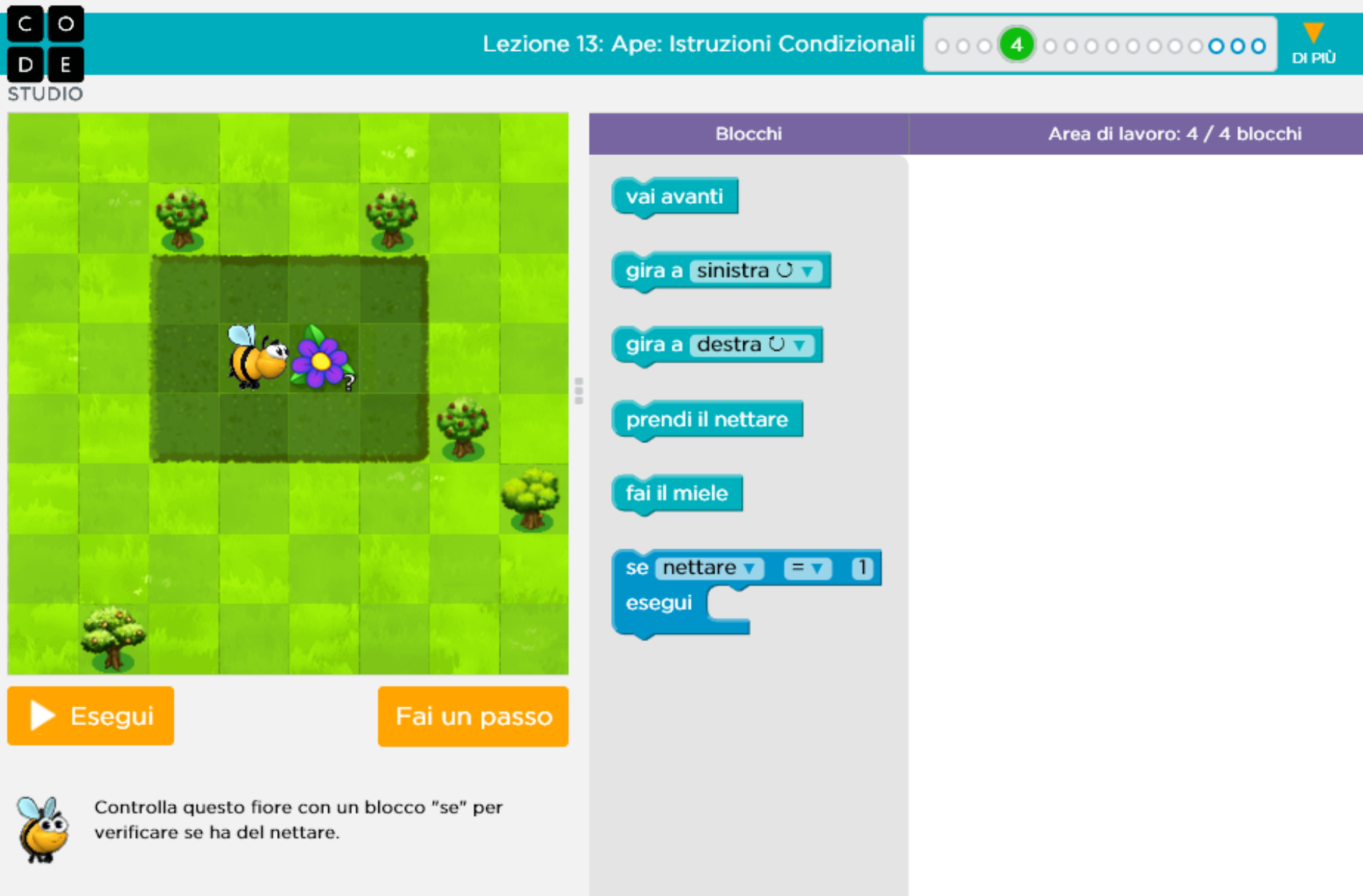


Dopo aver eseguito **`instrBefore`** si valuta **`expression`**

Se **`expression`** è vera eseguo il ramo di istruzioni contenente **`instrA;`** e **`instrB`**

Altrimenti eseguo **`instrC;`**

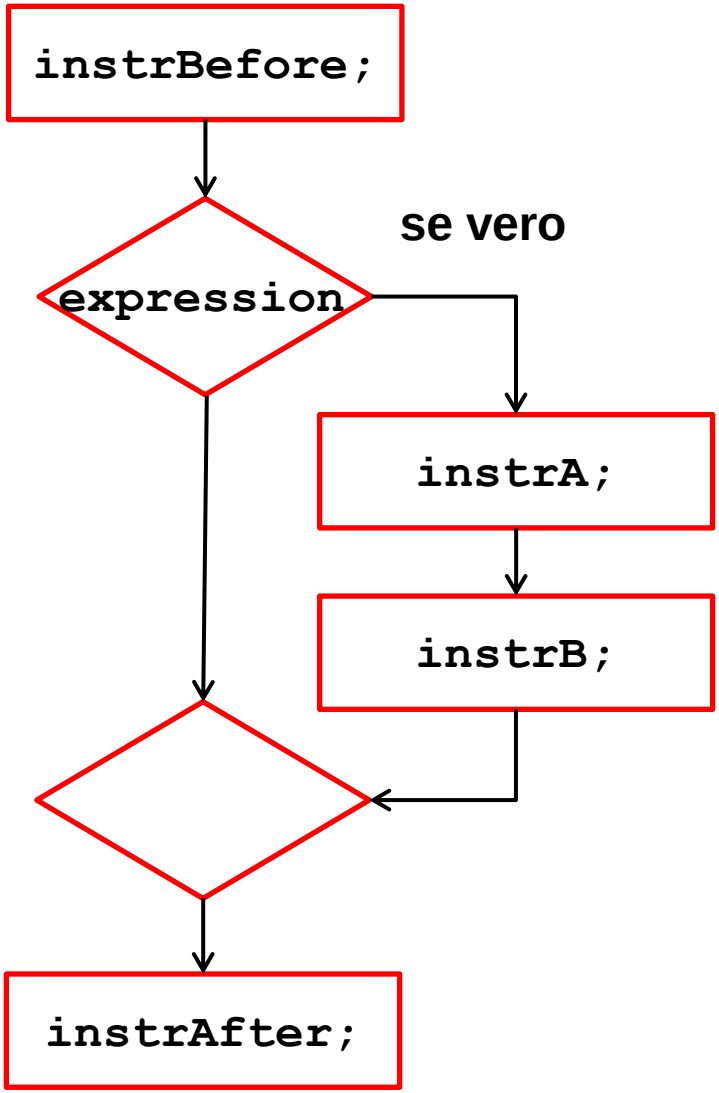
Il Costrutto Condizionale su code.org



Esempio: Algoritmo per Andare in Università

- Mi alzo quando suona la sveglia
- Mi dirigo verso la cucina
- Mangio e bevo un caffè
- Mi lavo e mi vesto
- **Se** c'è il laboratorio di informatica in presenza
  - Prendo il laptop
- Esco
- Corro

Costrutto Condizionale:



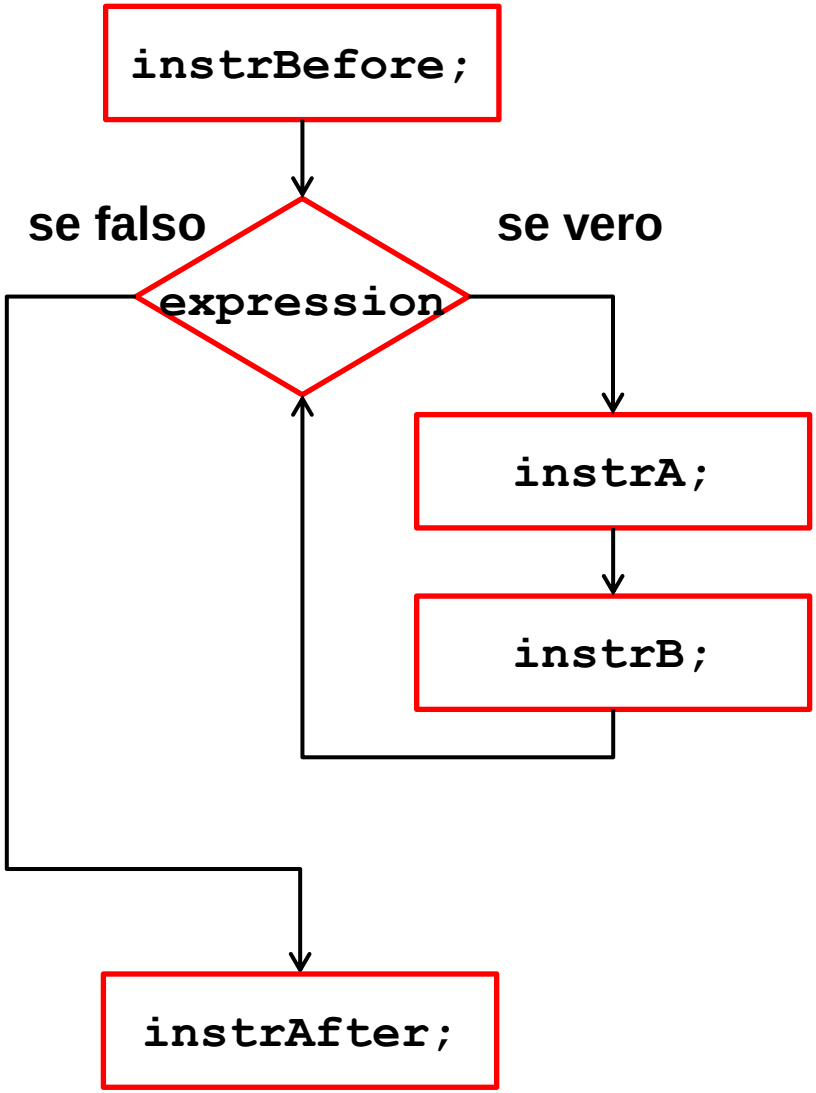
Non è necessario prevedere azioni specifiche nel caso in cui **`expression`** fosse falsa

Costrutto Iterativo

Esempio: Algoritmo per Andare in Università

- Mi alzo quando suona la sveglia
- Mi dirigo verso la cucina
- Bevo un caffè
- Mi lavo e mi vesto
- **Se** c'è il laboratorio di informatica
  - Prendo il laptop
- Vado in stazione
- **Ripeti finché** sei in stazione
  - Attendi un treno
  - Se passa per Bovisa, sali sul treno
- Arrivi a lezione

Costrutto Iterativo



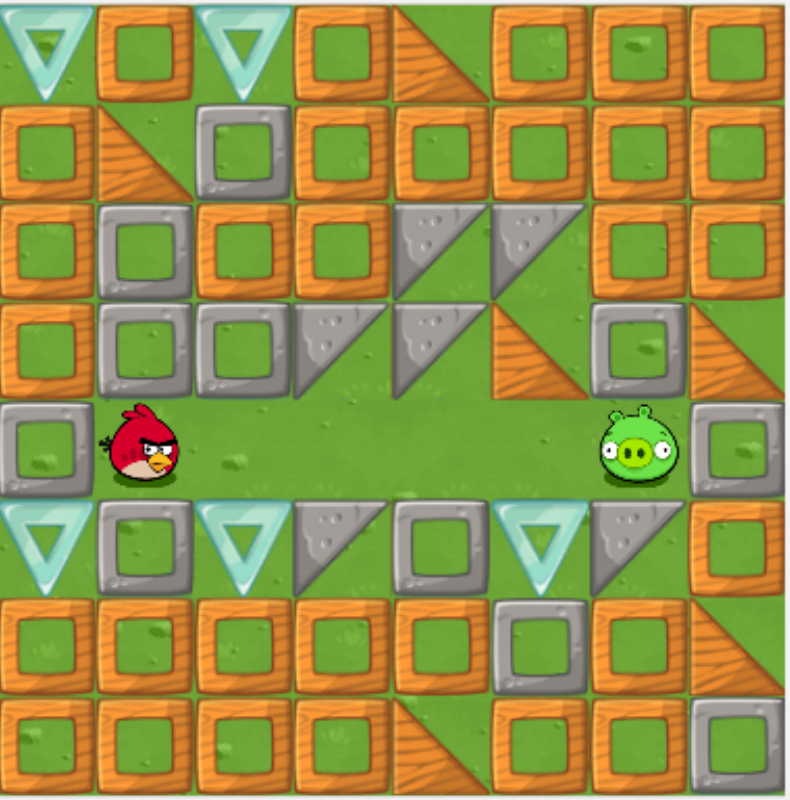
Se **`expression`** è vera eseguo il ramo di istruzioni contenente **`instrA;`** e **`instrB;`** (corpo del ciclo)

Al termine valuta nuovamente **`expression`**

Se **`expression`** è falsa, prosegui oltre, altrimenti esegui le istruzioni nel corpo del ciclo

Il Costrutto Iterativo su code.org

STUDIO



Esegui

Ok, prova ad usare il nuovo blocco "ripeti fino a che". Esso mi farà "ripetere" le azioni "fino a che" raggiungo quel fastidioso maiale.

Hai bisogno di aiuto?

Guarda questi video e suggerimenti

Blocchi | Area di lavoro: 3 / 3 blocchi

vai avanti

gira a sinistra

gira a destra

ripeti fino a che

esegui

Il Costrutto Iterativo su code.org

Esegui

Caro umano. Io zombie. Io affamato. Devo ... arrivare ... al girasole. Riesci a farmi arrivare là con solo 5 blocchi?

Hai bisogno di aiuto?

Guarda questi video e suggerimenti

Blocchi | Area di lavoro: 6 / 6 blocchi

vai avanti

gira a sinistra

gira a destra

ripeti fino a che

esegui

Esempio: algoritmo per andare in Università

- Mi alzo quando suona la sveglia
- Mi dirigo verso la cucina
- Mangio e bevo un caffè
- Mi lavo e mi vesto
- **Se** c'è il laboratorio di informatica
  - Prendo il laptop
- Vado in stazione
- **Ripeti finché** il treno in arrivo non ferma a Bovisa
  - Attendi il prossimo treno
- Sali sul treno
- Arrivi a lezione, wow

Esempi di Algoritmi

Il Pallottoliere

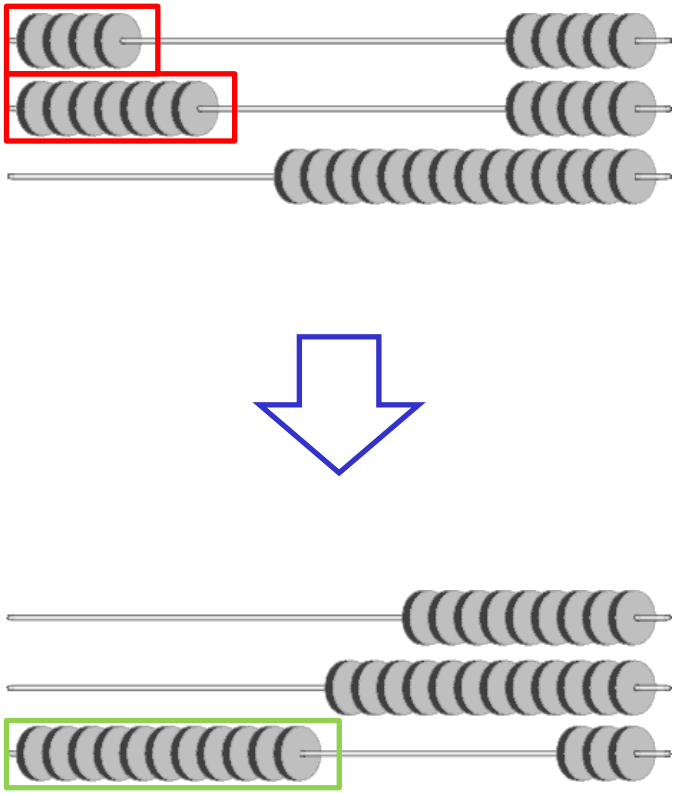
Supponiamo di avere un bambino che sa contare, ma non sa fare operazioni aritmetiche e non sa scrivere

Scriviamo le istruzioni per utilizzare il pallottoliere

Algoritmo per Sommare col Pallottoliere

Per semplificarci la vita suppoiamo che :

- Primo addendo è riportato sulla prima riga a sinistra
- Secondo addendo è sulla seconda riga a sinistra
- Risultato deve apparire nella terza riga
- Operiamo con numeri «piccoli», non c'è bisogno del riporto



Algoritmo del Pallottoliere

1. Sposta una pallina da sinistra a destra nella prima riga, al contempo da destra a sinistra nella terza riga
2. **Ripeti** operazione precedente **fino a** svuotare parte sinistra prima riga
3. Sposta pallina da sinistra a destra nella seconda riga, al contempo da destra a sinistra nella terza.
4. **Ripeti** operazione precedente **fino a** svuotare parte sinistra seconda riga
5. Conta quante palline trovi sulla terza riga

Proprietà Fondamentali degli Algoritmi

Correttezza:

- l'algoritmo arriva alla soluzione del problema per cui è stato progettato

Efficienza:

- l'algoritmo usa risorse in modo minimale (o almeno ragionevole)

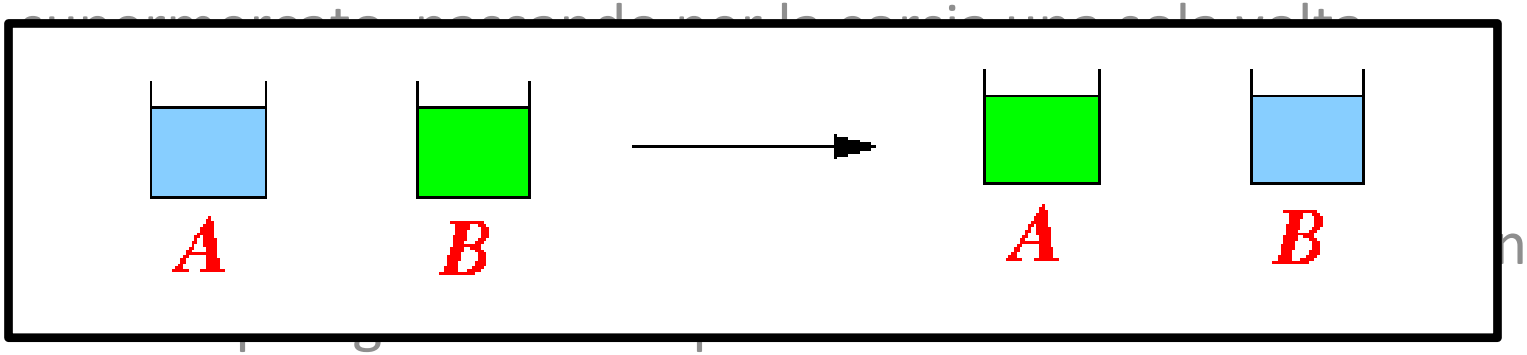
Altri Esempi

- Algoritmo per invertire il contenuto di due bicchieri, supponendo di avere un terzo bicchiere
- Algoritmo per trovare il prodotto migliore nella corsia del supermercato, passando per la corsia una sola volta
- Algoritmo per controllare che la maggioranza degli studenti abbia capito gli esercizi sopra
- Algoritmo per ricercare i libri in biblioteca

Altri Esempi

- Algoritmo per invertire il contenuto di due bicchieri, supponendo di avere un terzo bicchiere

- Algoritmo per trovare il prodotto più buono nella corsia del supermercato, passando per la corsia una sola volta

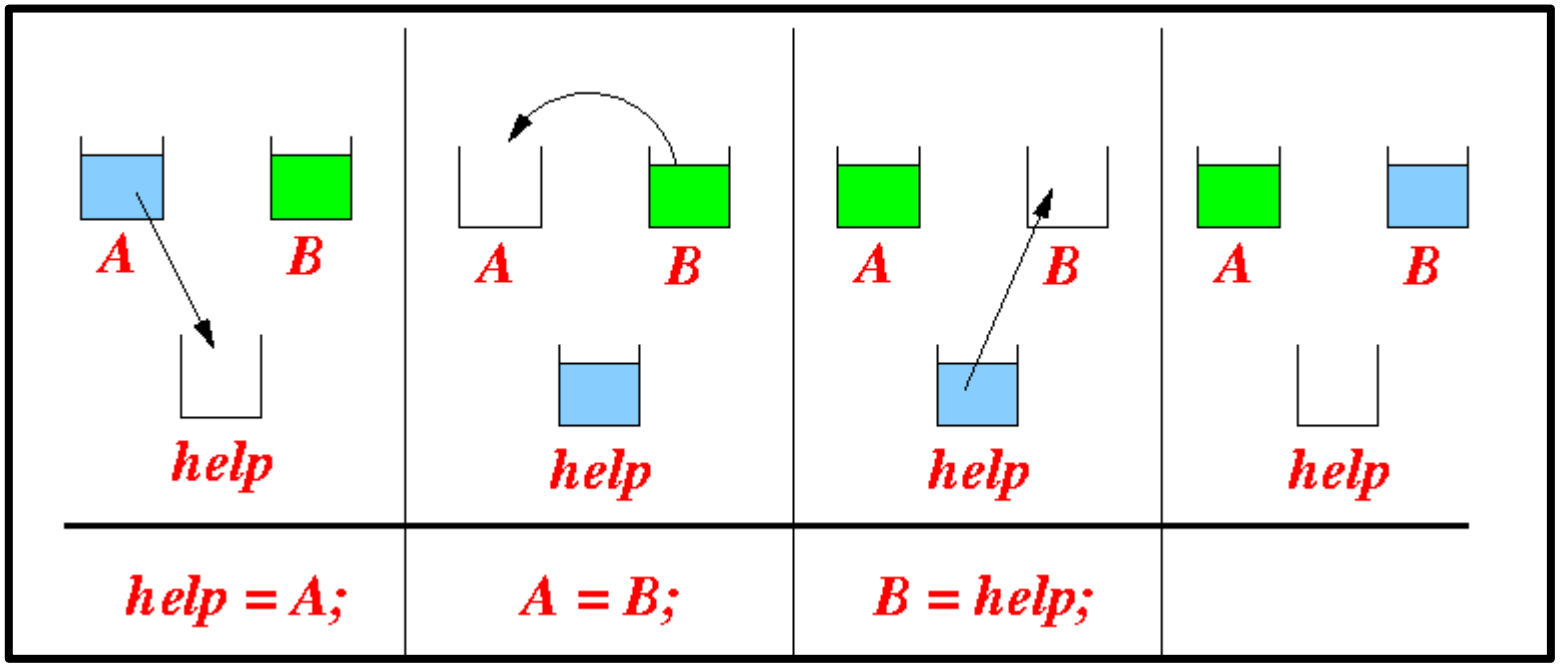


- [illegible]nti

- Algoritmo per ricercare i libri in Biblioteca

Esempio: Invertire il Contenuto di A e B

1. Prendi un terzo bicchiere C
2. Rovescia il contenuto del bicchiere A nel bicchiere C
3. Rovescia il contenuto di B in A
4. Rovescia il contenuto di C in B



Altri Esempi

- Algoritmo per invertire il contenuto di due bicchieri, supponendo di avere un terzo bicchiere
- Algoritmo per trovare il prodotto migliore nella corsia del supermercato, passando per la corsia una sola volta
- Algoritmo per controllare che la maggioranza degli studenti abbia capito gli esercizi sopra
- Algoritmo per ricercare i libri in biblioteca

Esempio: Ricerca del Prodotto Migliore

1. Prendi in mano il primo prodotto e assumi che sia il migliore
2. Procedi fino al prossimo prodotto
3. Confrontalo con quello che hai in mano
4. **Se** il prodotto davanti a te è migliore: abbandona il prodotto che hai in mano e prendi quello sullo scaffale
5. **Ripeti** i passi **2 - 4 fino a** raggiungere la fine della corsia

Algoritmo per trovare il massimo di una sequenza numerica

Altri Esempi

- Algoritmo per invertire il contenuto di due bicchieri, supponendo di avere un terzo bicchiere
- Algoritmo per trovare il prodotto migliore nella corsia del supermercato, passando per la corsia una sola volta
- Algoritmo per controllare che la maggioranza degli studenti abbia capito gli esercizi sopra
- Algoritmo per ricercare i libri in biblioteca

Esempio: Assicurarsi che la Maggioranza Abbia Capito

1. Prendi due **fogli**, uno per contare chi non ha capito (N) ed uno per contare tutti gli studenti (T)
2. Avvicinati al primo studente
3. Chiedi se ha capito
4. **Se** risponde no, metti un segno su N
5. Metti un segno su T
6. Passa al prossimo studente
7. **Ripeti** i passi **3 – 6 fino** all'ultimo studente
8. Conta i segni su N e su T
9. **Se** il numero di N è maggiore della metà di T, la condizione è non verificata

Esempio: Assicurarsi che Tutti Abbiano Capito

1. Prendi un solo **foglio**
2. Avvicinati al primo studente
3. Chiedi se ha capito
4. **Se** risponde no, metti un segno sul foglio
5. Passa al prossimo studente
6. **Ripeti** i passi **3 – 5 fino** all'ultimo studente
7. Se il foglio non ha segni, tutti hanno capito

Esempio: Assicurarsi che Tutti Abbiano Capito

1. Prendi un solo **foglio**
2. Avvicinati al primo studente
3. Chiedi se ha capito
4. **Se** risponde no, metti un segno sul foglio
5. Passa al prossimo studente
6. **Ripeti** i passi **3 – 5 fino** all'ultimo studente **o fino** a quando uno studente dice di non aver capito
7. Se il foglio non ha segni, tutti hanno capito

Variante più efficiente

Esempio: Assicurarsi che Tutti Abbiano Capito

1. Prendi un solo **foglio**
2. Avvicinati al primo studente
3. Chiedi se ha capito
4. **Se** risponde no, metti un segno sul foglio
5. Passa al prossimo studente
6. **Ripeti** i passi **3 – 5 fino** all'ultimo studente **o fino** a quando uno studente dice di non aver capito
7. Se il foglio non ha segni, tutti hanno capito

Algoritmo per verificare che una condizione sia soddisfatta da tutti gli elementi di un insieme

Altri Esempi

- Algoritmo per invertire il contenuto di due bicchieri, supponendo di avere un terzo bicchiere
- Algoritmo per trovare il prodotto migliore nella corsia del supermercato, passando per la corsia una sola volta
- Algoritmo per controllare che la maggioranza degli studenti abbia capito gli esercizi sopra
- Algoritmo per ricercare i libri in biblioteca

Gestione di una Biblioteca

- Si supponga di avere una biblioteca gestita mediante archivio cartaceo
- Ogni libro ha una data posizione identificata da SCAFFALE e POSIZIONE
- Esiste un archivio ordinato per autore che contiene, per ogni libro, un foglio con autore, titolo, scaffale e posizione

AUTORE / I:
GHEZZI CARLO,
JAZAYERI MEHDI.

TITOLO:
PROGRAMMING LANGUAGE CONCEPTS.
1981.

SCAFFALE 35
POSIZIONE 21

Esempio: Ricerca di un Libro

1. **ricerca** la scheda del libro nello schedario
2. trovata la scheda, **segna** su un **foglio** numero scaffale e posizione del libro
3. raggiungi lo scaffale indicato
4. individuato lo scaffale, **ricerca** la posizione del libro
5. prendi il libro

AUTORE / I:
GHEZZI CARLO,
JAZAYERI MEHDI.

TITOLO:
PROGRAMMING LANGUAGE CONCEPTS.
1981.

SCAFFALE 35
POSIZIONE 21

Criticità nella Ricerca di un Libro

Non tutti gli esecutori sono in grado di «cercare», e comunque la ricerca può essere fatta in diversi modi, ad esempio:

1. esamina la **prima** scheda dello schedario
2. **se** autore e titolo coincidono con quelli cercati
   - ricerca **conclusa** con successo

   altrimenti passa a scheda successiva
3. **Ripeti** istruzione **2, fino a conclusione o fino a raggiungere l'ultima scheda**
4. **se** trovata ⇒ ricerca conclusa con successo
   - **altrimenti** ⇒ ricerca conclusa con insuccesso

Algoritmo: Ricerca nell'Archivio

Assumendo che l'archivio abbia N schede:

Algoritmo: Ricerca nell'Archivio

L'algoritmo precedente è:

- ben definito e semplice: le istruzioni sono precise e atomiche
- flessibile: funziona in qualunque modo siano disposte le schede
- inefficiente: non sfrutta l'ordinamento delle schede nell'archivio

Algoritmo: Ricerca tra Elementi Ordinati

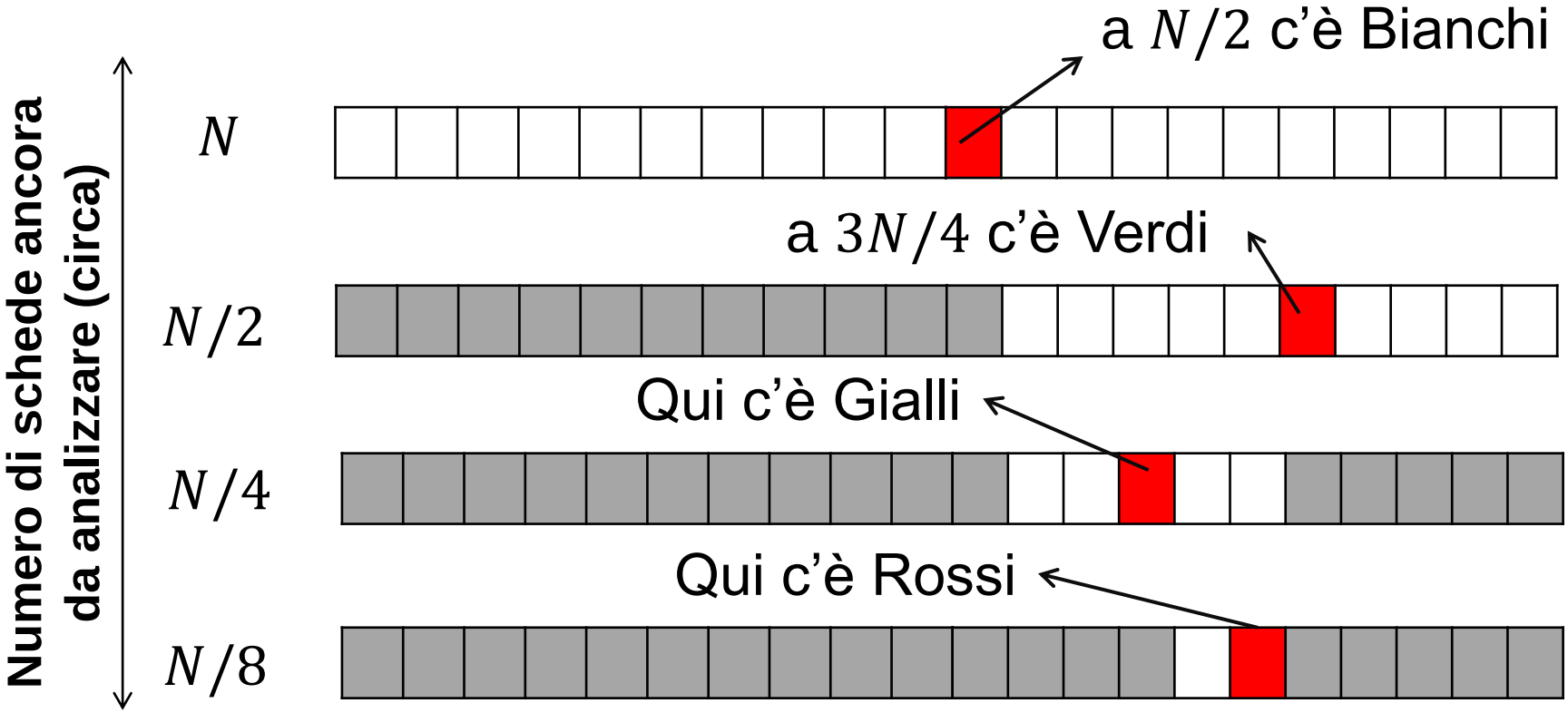
Lo schedario contiene N schede ordinate in cui cercare

1. prendi la scheda alla posizione $\lfloor N/2 \rfloor$
2. **se** è la scheda cercata, termina con successo,
   altrimenti,
3. **se** la scheda cercata segue alfabeticamente quella esaminata,
   - continua la **ricerca** nella seconda metà dello schedario
   - **altrimenti** continua la **ricerca** nella prima metà

Questo algoritmo ricerca **è ricorsivo**, perché richiama se stesso (riduce però il numero di schede da analizzare e si ferma ad un certo punto)

Algoritmo: Ricerca tra Elementi Ordinati

es: ricerca dell'autore ROSSI



N.B: cosa succede se non esiste il nome cercato nell'archivio? L'algoritmo deve terminare anche in questo caso!

Algoritmo: Ricerca tra Elementi Ordinati

Lo schedario contiene N schede ordinate in cui cercare

1. prendi la scheda alla posizione $\lfloor N/2 \rfloor$
2. **se** è la scheda cercata, termina con successo,
   altrimenti,
3. **se** la scheda cercata segue alfabeticamente quella esaminata,
   - continua la **ricerca** nella seconda metà dello schedario
   - **altrimenti** continua la **ricerca** nella prima metà

Devo aggiungere che cosa fare nel caso in cui sono arrivato ad avere una zona di ricerca vuota

Algoritmo: Ricerca tra Elementi Ordinati

Lo schedario contiene N schede ordinate in cui cercare

1. **se** N è zero (porzione di schedario vuota) allora termina la ricerca, **altrimenti,** prendi la scheda alla posizione $\lfloor N/2 \rfloor$
2. **se** è la scheda cercata, termina con successo,

   altrimenti,
3. **se** la scheda cercata segue alfabeticamente quella esaminata,
   - continua la **ricerca** nella seconda metà dello schedario
   - **altrimenti** continua la **ricerca** nella prima metà

I Programmi

dall’algoritmo al codice

Il Calcolatore

Il **calcolatore** è un potente **esecutore di algoritmi**

- ↑ È rapido: permette di gestire quantità di informazioni altrimenti intrattabili
- ↑ È preciso: non commette mai errori
- ↓ Non ha spirito critico

I **programmi** sono **algoritmi codificati** in **linguaggi** comprensibili dal calcolatore

Il Programmatore...Ossia Voi

Compito dell'informatico (e di chiunque programmi) è:

1. **ideare l'algoritmo**: conoscere la soluzione del problema e esprimerla in rigorosi passi
2. **codificare l'algoritmo in un programma**: conoscere il linguaggio dell'esecutore (linguaggio di programmazione)

strettamente in questo ordine!!!

- La parte più difficile è spesso la prima
- Prima dobbiamo aver chiaro «cosa far fare alla macchina», poi lo traduciamo correttamente

Proprietà Essenziali dei Programmi

Essendo delle traduzioni degli algoritmi richiediamo le stesse proprietà degli algoritmi per valutarne le qualità

Correttezza: l'algoritmo risolve il compito senza errori o difetti

Efficienza: l'algoritmo usa risorse in modo minimale e/o ragionevole in termini di:

- tempo
- memoria
- numero di letture/scritture da disco
- ...

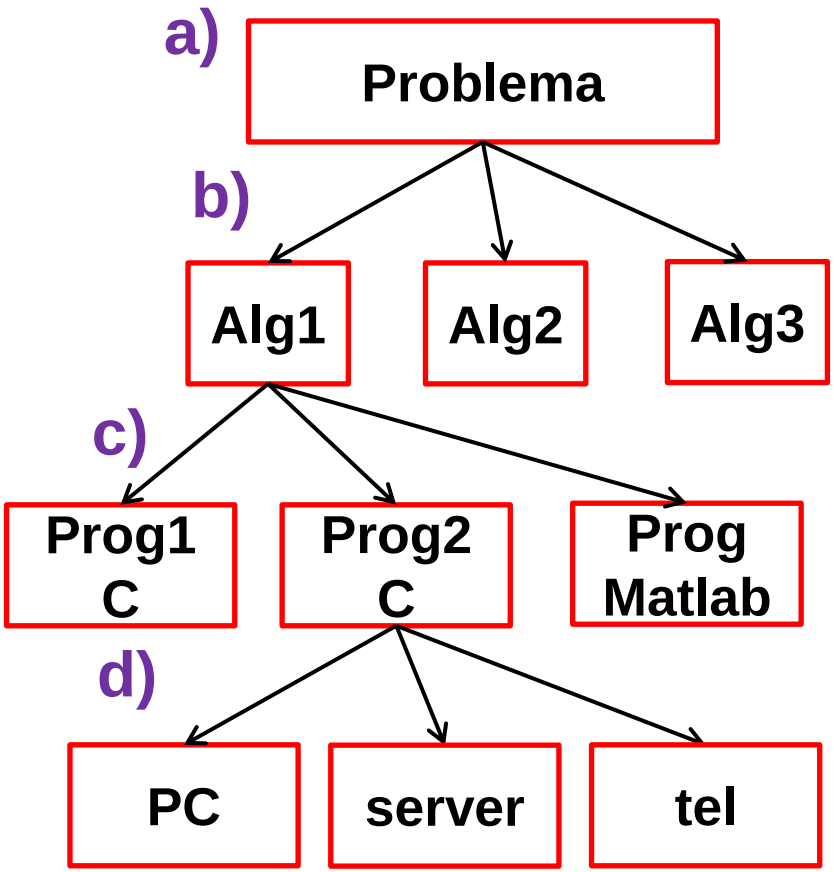
Riepilogando: Cosa Fare

a) Dovrete (capire e) definire correttamente il problema

b) **Ricavare l'algoritmo** è la parte più **difficile** e richiede, oltre all'esperienza, competenze specifiche, creatività e anche rigore perché occorre specificarlo in modo formale

c) La codifica è più semplice ma il programma deve essere efficiente e corretto

d) Alla compilazione (traduzione in linguaggio macchina) pensa il calcolatore



La Rappresentazione dell’Informazione

Cos'è l'Informatica?

Scienza della rappresentazione e dell'elaborazione dell'informazione

- ***Scienza:*** ovvero una **conoscenza sistematica** e **rigorosa** di tecniche e metodi
- ***Informazione:*** l'oggetto dell'investigazione scientifica (informazione intesa come entità astratta e come tecnologie per la sua gestione)
- ***Rappresentazione:*** il modo in cui l'informazione viene strutturata e trasformata in dati fruibili da macchine
- ***Elaborazione:*** uso e trasformazione dell'informazione per un dato scopo. L'elaborazione deve poter essere eseguita da macchine che processano dati

Rappresentazione dell'informazione

I calcolatori sono in grado di operare con informazioni **binarie**. Gli unici simboli che possiamo utilizzare sono 0 e 1

Il **bit** (*binary digit*) assume valore 0/1 corrispondente ad uno *stato fisico della memoria* (alta o bassa tensione)

Il **Byte** è una sequenza di 8 bit ed esprime $2^8 = 256$ numeri diversi

Es: 00000000, 00000001, 00000010, …, 11111111

Codifica binaria di un numero: la sua **rappresentazione** come una **sequenza di 0 e 1**

In questo corso vedremo la **codifica binaria di numeri interi** e semplici **operazioni aritmetiche**

Aritmetica Binaria

...o contare in maniera efficiente

Codifica dei Numeri in Base 10

- Utilizziamo una **notazione posizionale**:
  - Le cifre all'interno di un numero hanno un significato differente a seconda della loro posizione
- Le **cifre** che abbiamo a disposizione sono 10

$$\{0, 1, \dots, 9\}$$

- Es: il numero di 4 cifre
  - $\mathbf{3401} = \mathbf{3} \times 10^3 + \mathbf{4} \times 10^2 + \mathbf{0} \times 10^1 + \mathbf{1} \times 10^0$

  è diverso da altre combinazioni delle stesse cifre:
  - $\mathbf{1403} = \mathbf{1} \times 10^3 + \mathbf{4} \times 10^2 + \mathbf{0} \times 10^1 + \mathbf{3} \times 10^0$
  - $\mathbf{0314} = \mathbf{0} \times 10^3 + \mathbf{3} \times 10^2 + \mathbf{1} \times 10^1 + \mathbf{4} \times 10^0$

Codifica dei Numeri in Base 10

- Le **cifre** sono **gli elementi** dell'alfabeto che abbiamo a disposizione per scrivere i numeri
- In base 10 abbiamo questo alfabeto:

$$A_{10} = \{0,1, \dots, 9\}$$

- I **numeri** sono sequenze ordinate di cifre (es. 3401)
- Un numero di m cifre $(3401, m = 4$, e $332, m = 3)$ si può scrivere come:

$$(N)_{10} = a_{m-1}a_{m-2} \dots a_1 a_0 = \sum_{i=0}^{m-1} a_i \times 10^i \, , a_i \in A_{10}$$

- Con m cifre posso rappresentare 10^m numeri distinti:

$$\{0, \dots, 10^m - 1\}$$

Codifica dei Numeri in Base 2

- In base 2 abbiamo questo alfabeto:

$$A_2 = \{0,1\}$$

- Un numero di m cifre si può scrivere come:

$$(N)_2 = a_{m-1}a_{m-2} \dots a_1 a_0 = \sum_{i=0}^{m-1} a_i \times 2^i \, , a_i \in A_2$$

- Con m cifre posso rappresentare 2^m numeri distinti:

$$\{0, \dots, 2^m - 1\}$$

Es: un numero di 5 cifre

- $\mathbf{10010} = \mathbf{1} \times 2^4 + \mathbf{0} \times 2^3 + \mathbf{0} \times 2^2 + \mathbf{1} \times 2^1 + \mathbf{0} \times 2^0$

Range dei Numeri in Base 2

Qual è il numero massimo che posso scrivere con m bit?

Quante combinazioni di m elementi posso realizzare se ogni elemento lo scelgo tra 2 diversi? -> 2^m

Ad esempio:

Con 1 cifra in base 2, copro $[0, 2^1 - 1]$ (cioè $[0, 1]$)

Con 4 cifre in base 2, copro $[0, 2^4 - 1]$ (cioè $[0, 15]$)

Con 8 cifre in base 2, copro $[0, 2^8 - 1]$ (cioè $[0, 255]$)

Con 16 cifre in base 2, copro $[0, 2^{16} - 1]$ (cioè $[0, 65535]$)

In binario ho bisogno di molte più cifre rispetto al decimale per esprimere gli stessi numeri ma con meno simboli

Conversione Binario to 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.

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

$$(1100010)_2 = 1 \times 2^6 + 1 \times 2^5 + 1 \times 2 =$$

$$= 64 + 32 + 2 = (98)_{10}$$

Nel corso vedremo diverse codifiche per i numeri interi ed inoltre vedremo come eseguire la somma di interi

Un po' di Notazione

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 il loro legame con l'informatica:

- Byte = 8 bit = 2^3bit
- KiloByte (kB) = 10^3 Byte
- MegaByte (MB) = 10^6 Byte
- GigaByte (GB) = 10^9 Byte
- TeraByte (TB) = 10^{12} Byte

Codifica dei Caratteri

Ogni carattere viene mappato in un numero intero (che è espresso da sequenza di bit) utilizzando dei codici

Il codice più usato è l'*ASCII (American Standard Code for Information Interchange)* a **8 bit** che contiene:

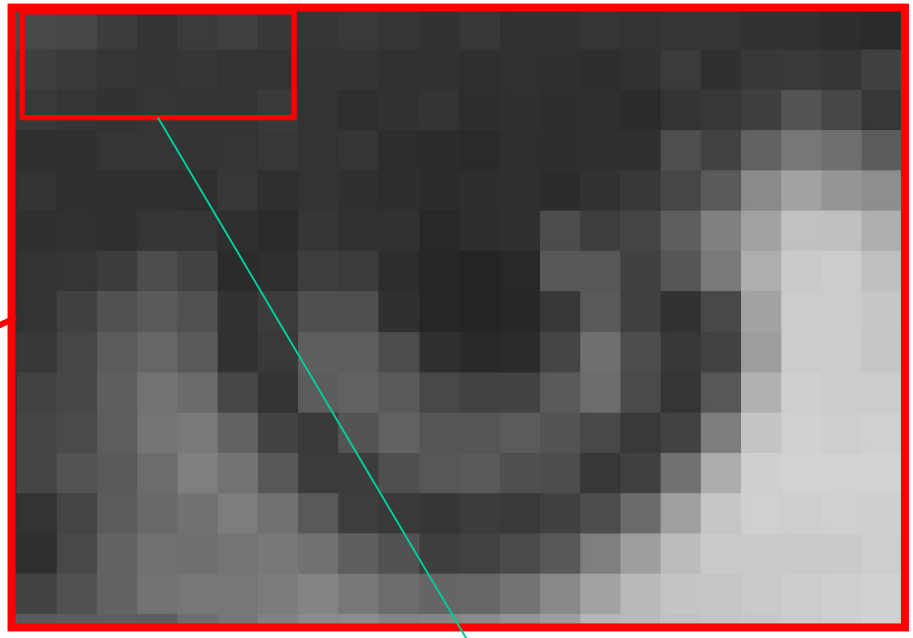
- Caratteri alfanumerici
- Caratteri simbolici (es. punteggiatura, @&%$ etc.)
- Caratteri di comando (es. termina riga, vai a capo, tab)

La Codifica ASCII

Caratt.	Byte	Caratt.	Byte	Caratt.	Byte	Caratt.	Byte
NUL	00000000	SPACE	00100000	@	01000000	`	01100000
SOH	00000001	!	00100001	A	01000001	a	01100001
STH	00000010	"	00100010	B	01000010	b	01100010
EXT	00000011	#	00100011	C	01000011	c	01100011
EQT	00000100	$	00100100	D	01000100	d	01100100
EQN	00000101	%	00100101	E	01000101	e	01100101
ACK	00000110	&	00100110	F	01000110	f	01100110
BEL	00000111	'	00100111	G	01000111	g	01100111
BS	00001000	(	00101000	H	01001000	h	01101000
HT	00001001	)	00101001	I	01001001	i	01101001
LF	00001010	*	00101010	J	01001010	j	01101010
VT	00001011	+	00101011	K	01001011	k	01101011
FF	00001100	,	00101100	L	01001100	l	01101100
CR	00001101	-	00101101	M	01001101	m	01101101
SO	00001110	.	00101110	N	01001110	n	01101110
SI	00001111	/	00101111	O	01001111	o	01101111
DLE	00010000	0	00110000	P	01010000	p	01110000
DC1	00010001	1	00110001	Q	01010001	q	01110001
DC2	00010010	2	00110010	R	01010010	r	01110010
DC3	00010011	3	00110011	S	01010011	s	01110011
DC4	00010100	4	00110100	T	01010100	t	01110100
NAK	00010101	5	00110101	U	01010101	u	01110101
SYN	00010110	6	00110110	V	01010110	v	01110110
ETB	00010111	7	00110111	W	01010111	w	01110111
CAN	00011000	8	00111000	X	01011000	x	01111000
EM	00011001	9	00111001	Y	01011001	y	01111001
SUB	00011010	:	00111010	Z	01011010	z	01111010
ESC	00011011	;	00111011	[	01011011	{	01111011
FS	00011100	<	00111100	\	01011100	ò	01111100
GS	00011101	=	00111101	é	01011101	à	01111101
RS	00011110	>	00111110	^	01011110	ù	01111110
US	00011111	?	00111111	_	01011111	DEL	01111111

Codifica delle Immagini

Le immagini nei calcolatori sono digitali, ossia tabella di pixel, ciascuno caratterizzato da uno o più valori di intensità



123	122	134	121	132	133	145	134
122	121	125	132	124	121	116	126
119	127	137	119	139	127	128	131

Codifica delle Immagini

Per le immagini a colori ho bisogno di avere un valore per ogni colore fondamentale

Canale rosso

Canale verde

Canale blu

Codifica delle Immagini

- Esistono diverse codifiche dell'immagine, non sempre questa viene scritta pixel per pixel
- È spesso conveniente rappresentare l'immagine secondo codifiche che permettano di ridurre le dimensioni
  - Codifiche *lossless*: permettono, senza perdita di informazione, di comprimere l'immagine
  - Codifiche *lossy:* comprimono perdendo però in qualità

Sistemi Informatici

Cos'è l'Informatica?

Scienza della rappresentazione e dell'elaborazione dell'informazione

- ***Scienza:*** ovvero una **conoscenza sistematica** e **rigorosa** di tecniche e metodi
- ***Informazione:*** l'oggetto dell'investigazione scientifica (informazione intesa come entità astratta e come tecnologie per la sua gestione)
- ***Rappresentazione:*** il modo in cui l'informazione viene strutturata e trasformata in dati fruibili da macchine
- ***Elaborazione:*** uso e trasformazione dell'informazione per un dato scopo. L'elaborazione deve poter essere eseguita da **macchine** che processano dati

Sistema Informatico

- Sistema informatico: l'esecutore dei programmi
- PC, laptop, telefoni, smartphones, server con migliaia di utenti sono 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, editor di testo, sistema operativo, compilatori, IDE e i programmi che scriveremo)
- O più semplicemente l'hardware è la parte del computer che puoi prendere a calci, il software quella contro cui puoi solo imprecare

La Macchina di Von Neumann

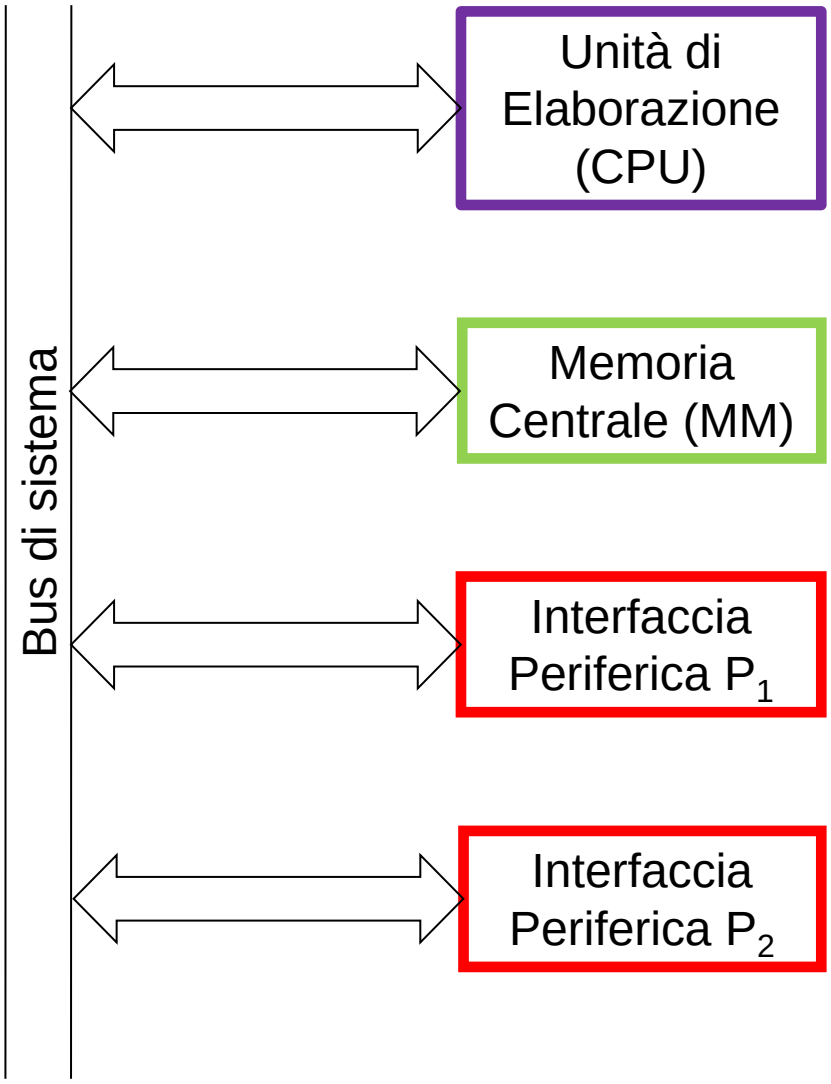
un modello per l’architettura dei calcolatori

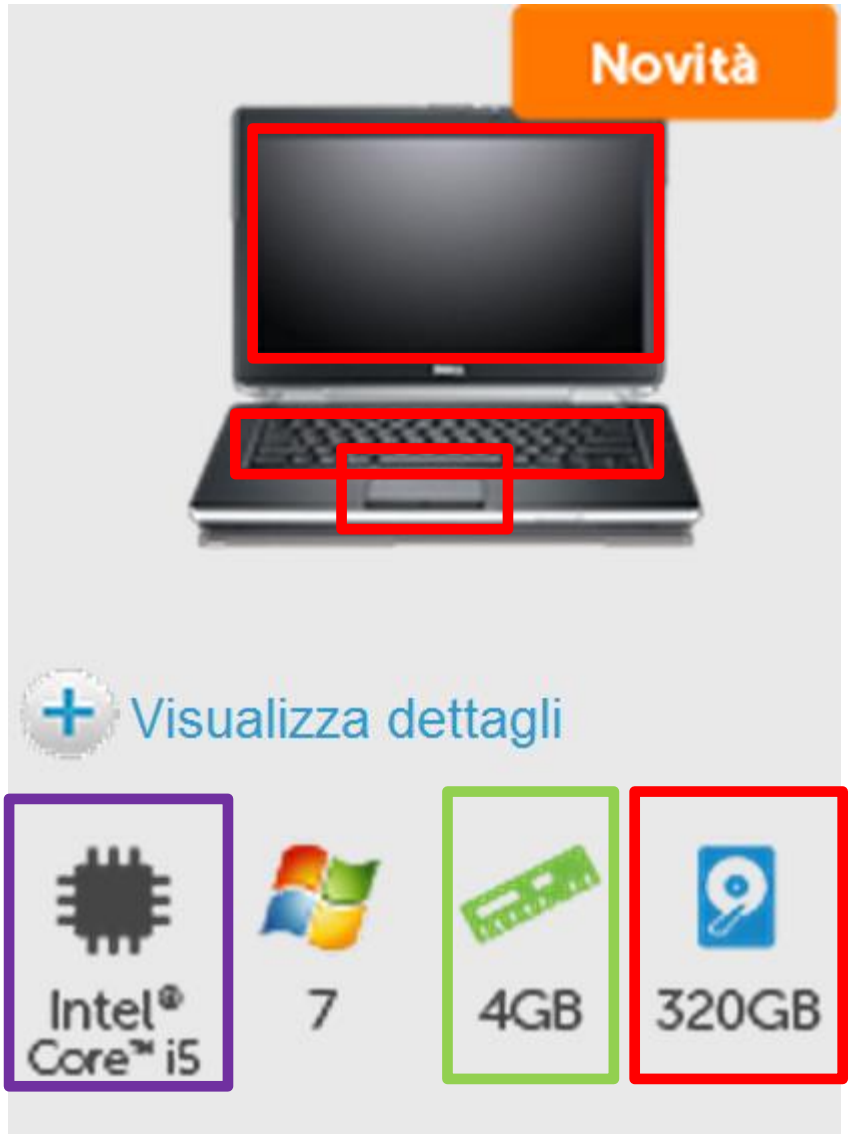
La Macchina di Von Neumann

Modello composto da **quattro elementi funzionali:**

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

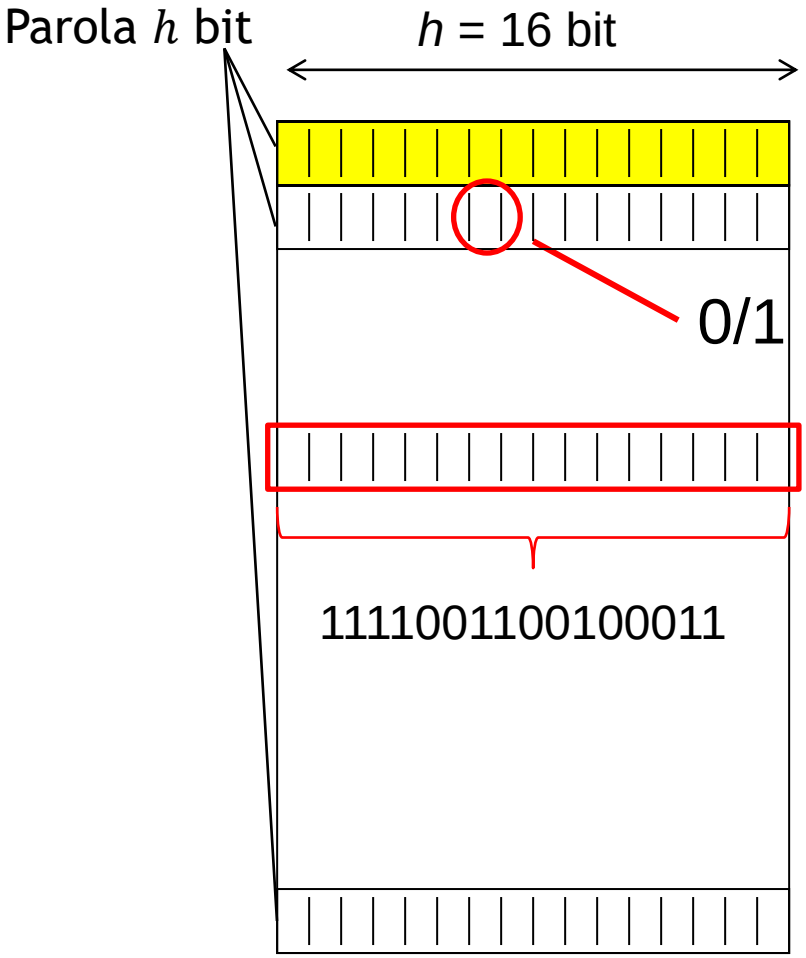
La Macchina di Von Neumann





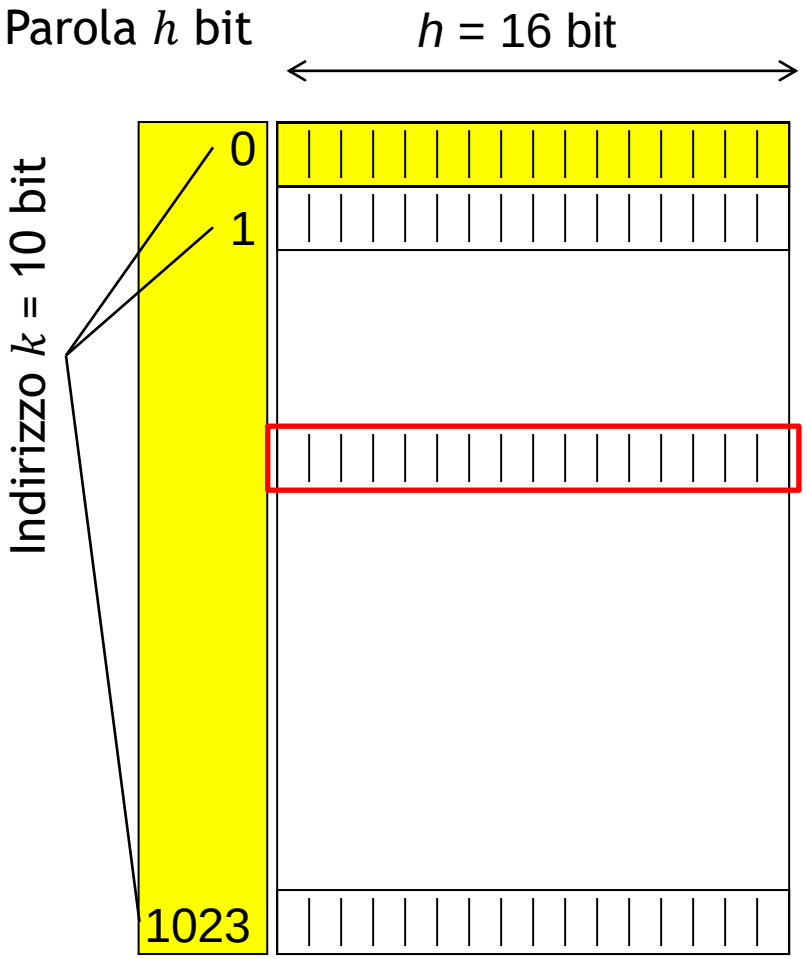
La Memoria Centrale (MM)

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



Avere più celle dello spazio di indirizzamento è come un palazzo senza scale che ha meno pulsanti dell'ascensore dei piani

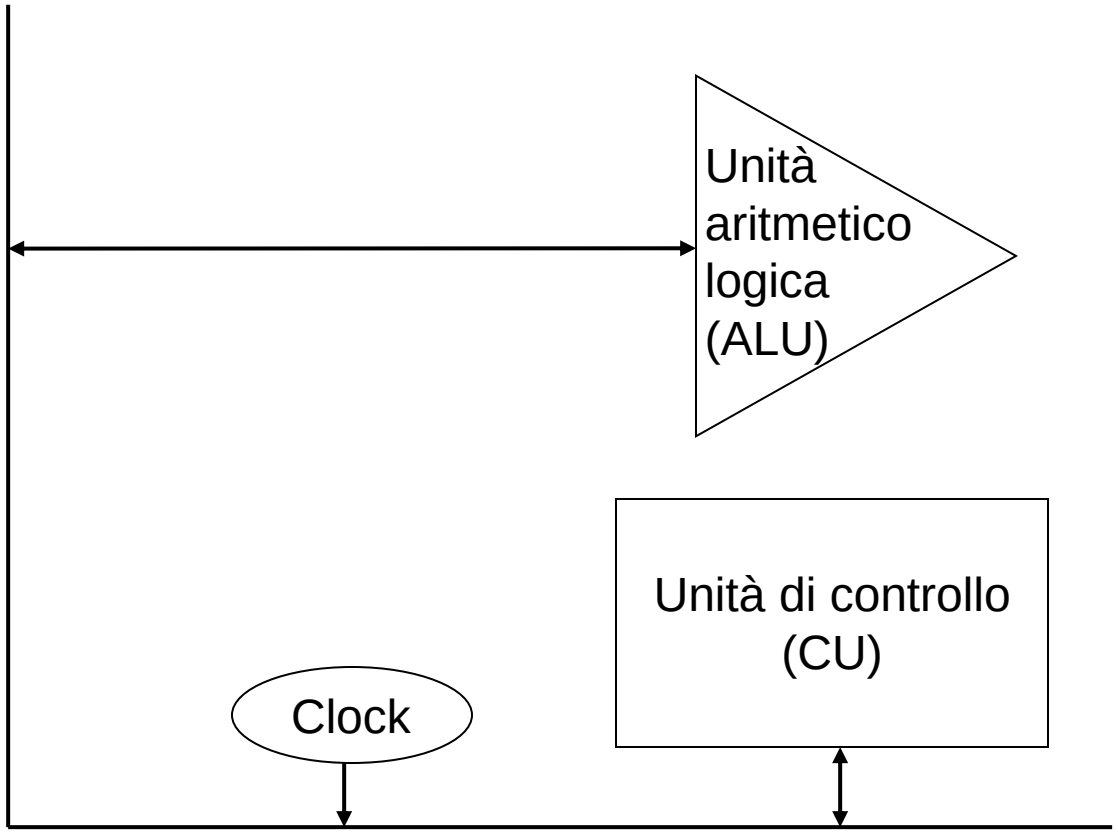
La Memoria Centrale (MM)

- 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)* memoria riprogrammabile
- **L'HDD** è memoria permanente ma **non è memoria centrale** ed, in riferimento alla macchina di Von Neumann, vedrmo che è una periferica

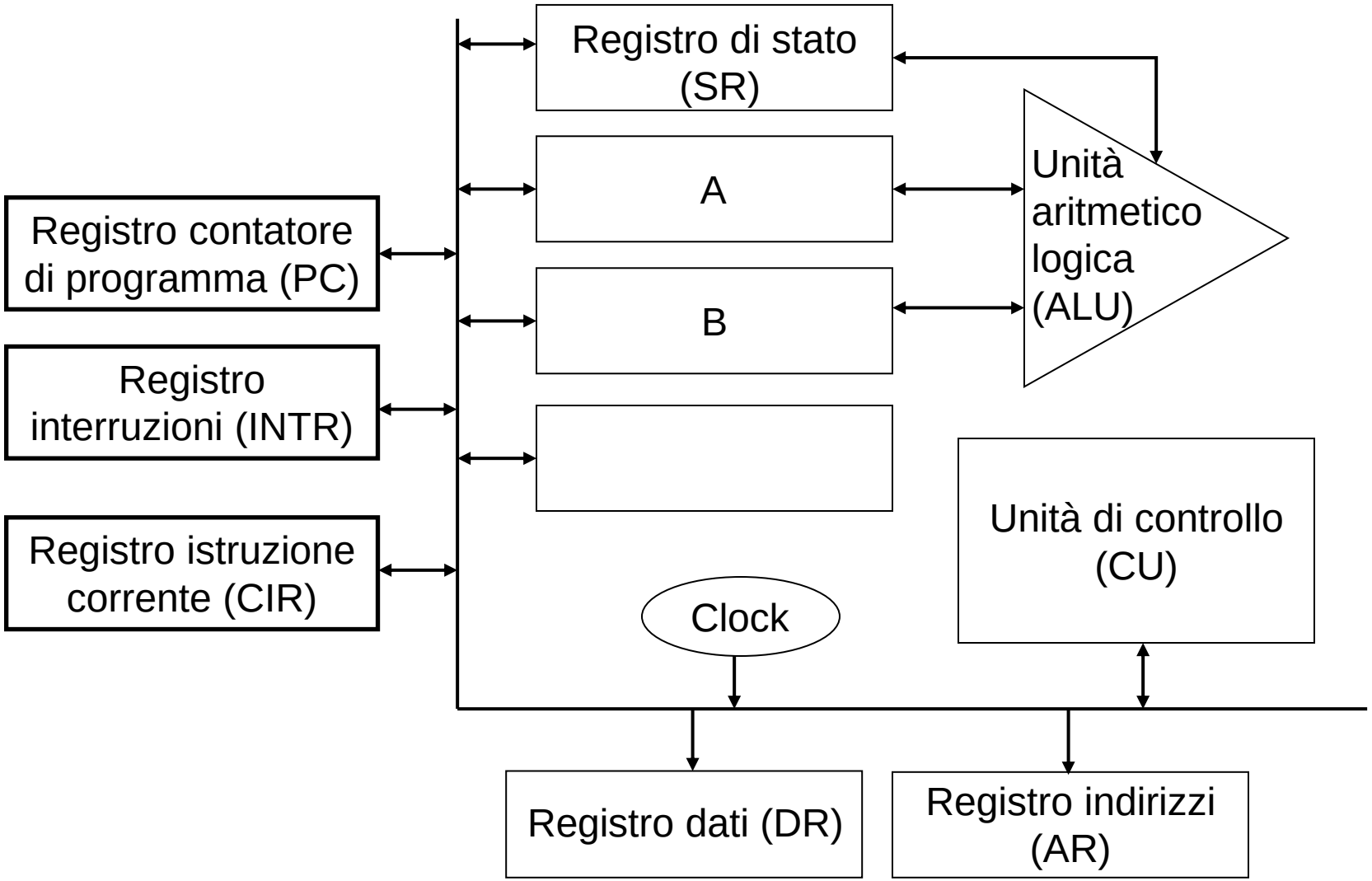
L'Unità di Elaborazione (CPU)

- 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 (CU):** preleva e decodifica istruzioni dalla MM, invia segnali per eseguire le istruzioni
  - Il **Clock di sistema**: sincronizza le operazioni della CPU
  - L'**Unità Aritmetico Logica (ALU):** operazioni aritmetiche e logiche

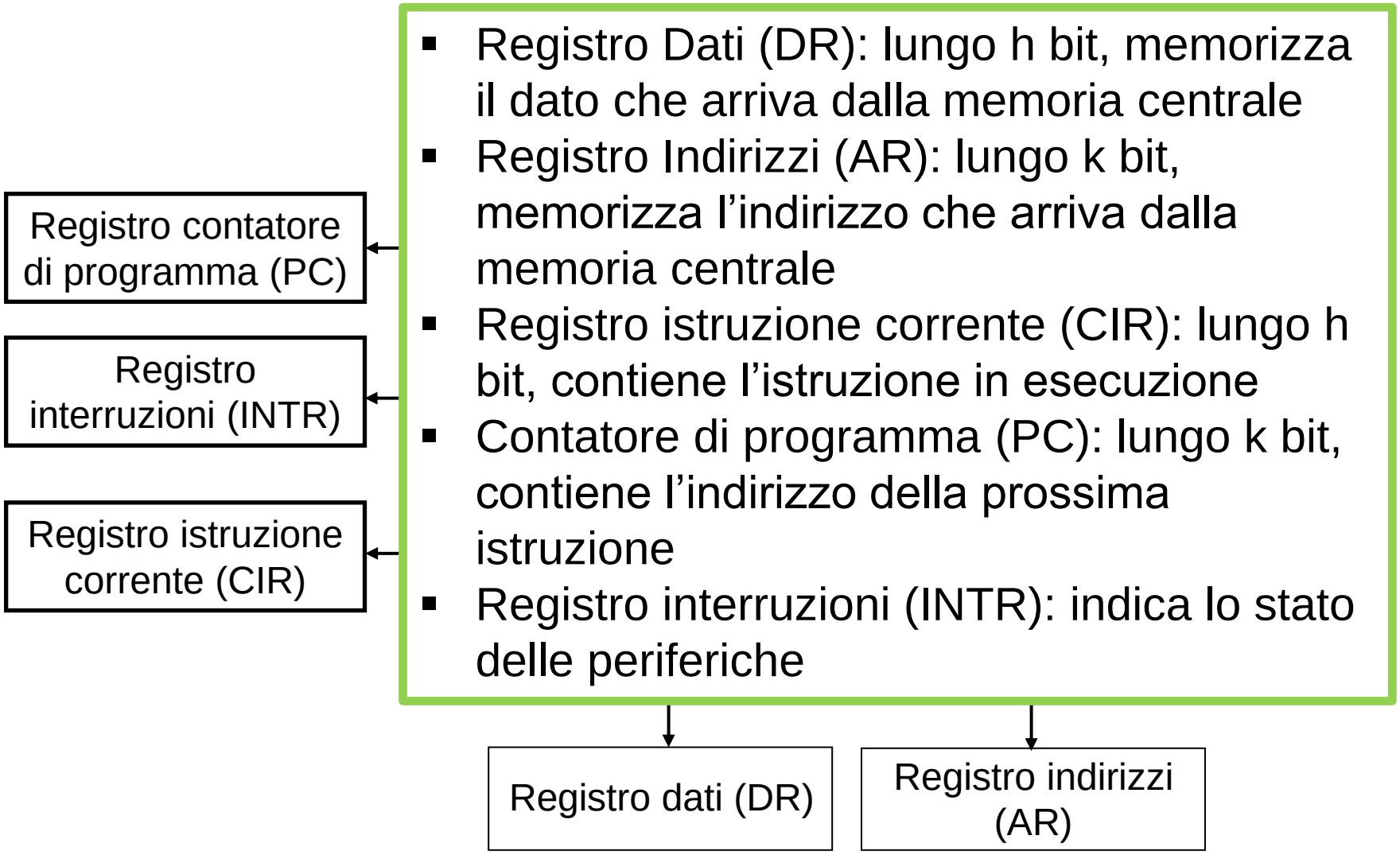
L'Unità di Elaborazione (CPU)



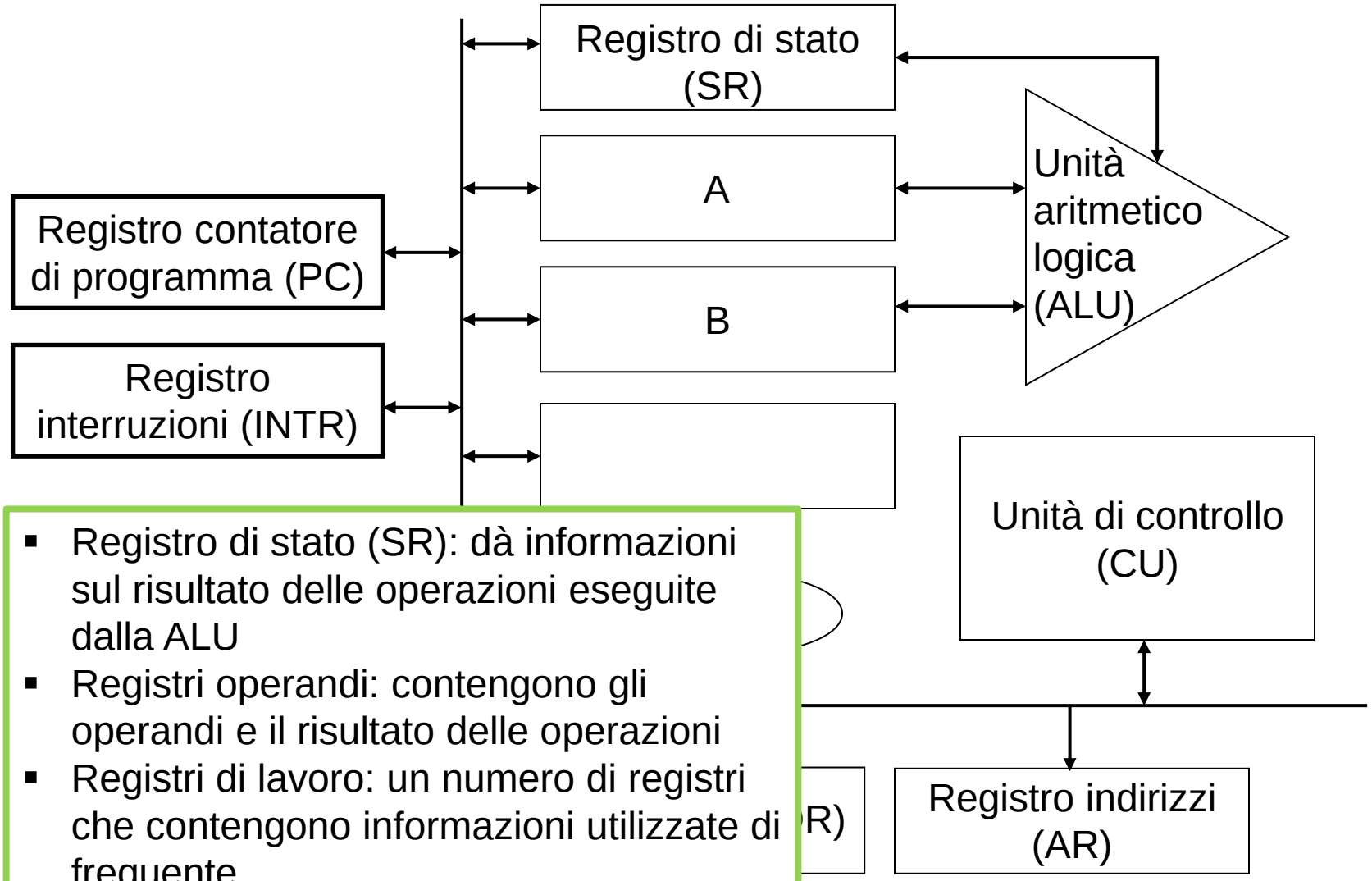
L'Unità di Elaborazione (CPU)



L'Unità di Elaborazione (CPU)



L'Unità di Elaborazione (CPU)



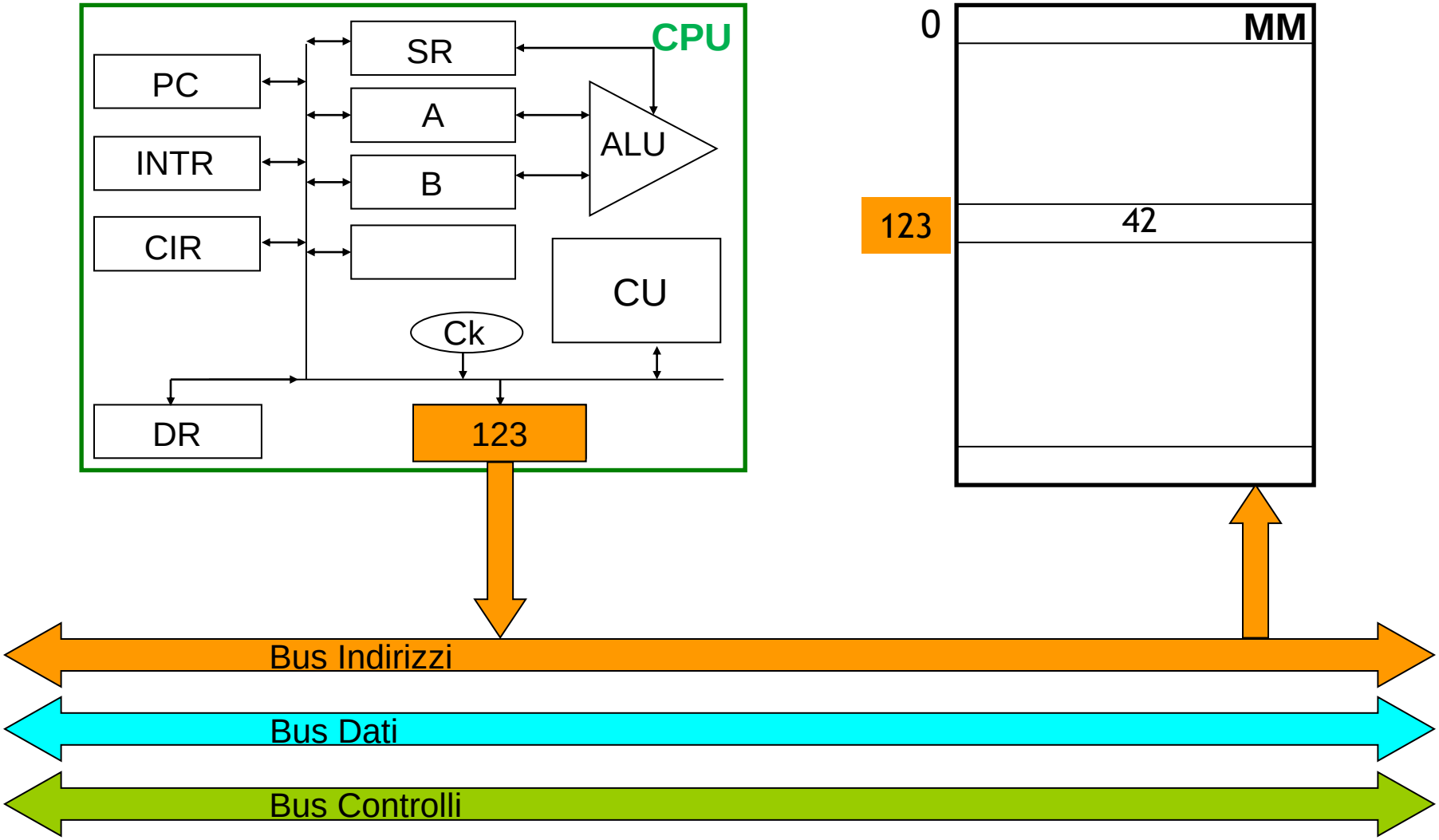
- Registro di stato (SR): dà informazioni sul risultato delle operazioni eseguite dalla ALU
- Registri operandi: contengono gli operandi e il risultato delle operazioni
- Registri di lavoro: un numero di registri che contengono informazioni utilizzate di frequente

Bus di Sistema

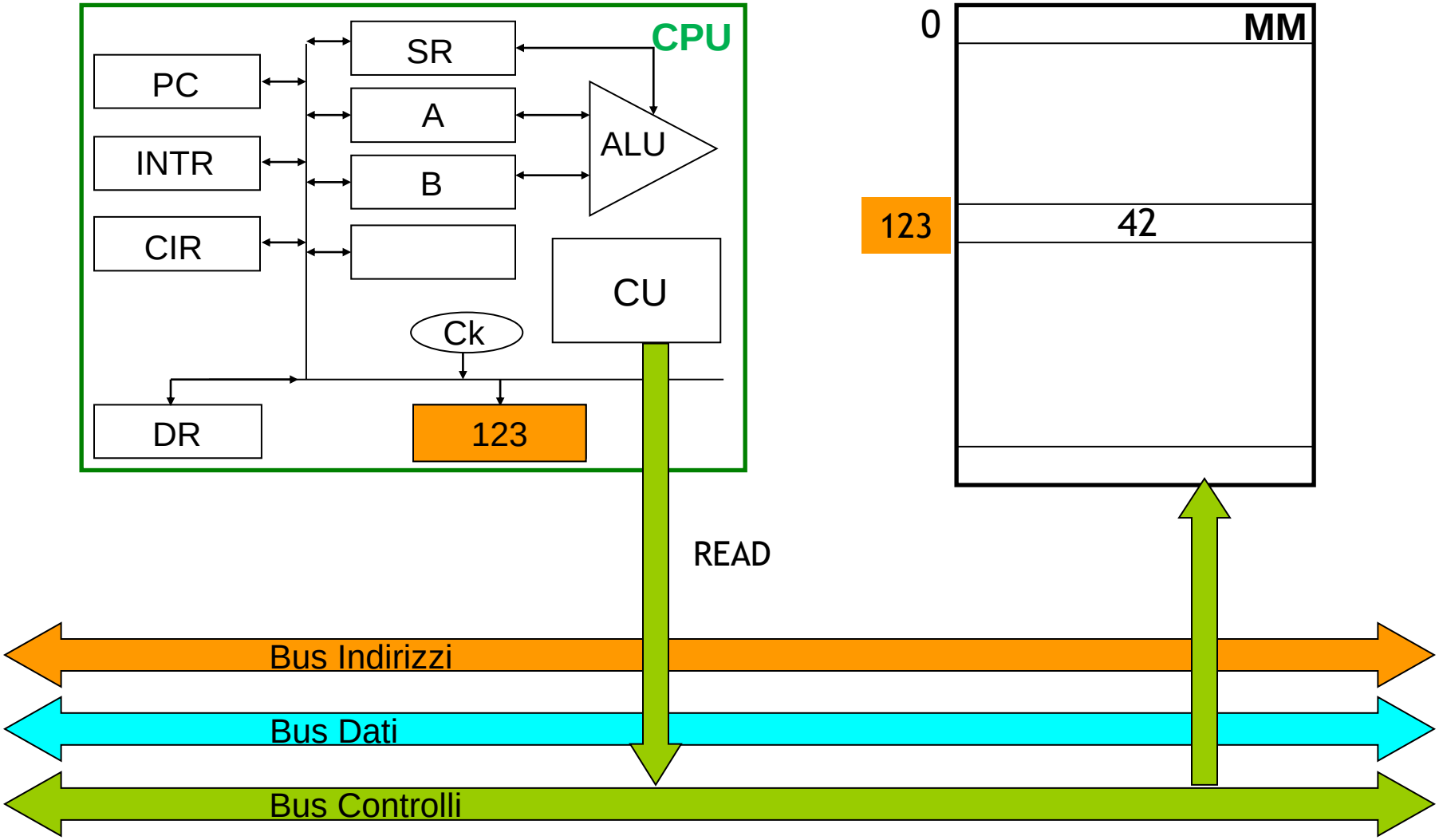
- È un insieme di connessioni che permettono di trasferire l'informazione tra **due** entità funzionali (un'entità trasmette, l'altra riceve)
- Due soli tipi di connessioni logiche, **stabilite** dalla **CPU**:
  - CPU (**leader**) - MM (**follower**)
  - CPU (**leader**) - interfaccia periferica (**follower**)

  le connessioni fisiche sono sempre presenti
- Ci sono **tre tipi di linee**, con tre funzionalità diverse
  - Bus **dati**
  - Bus **indirizzi**
  - Bus **controlli**

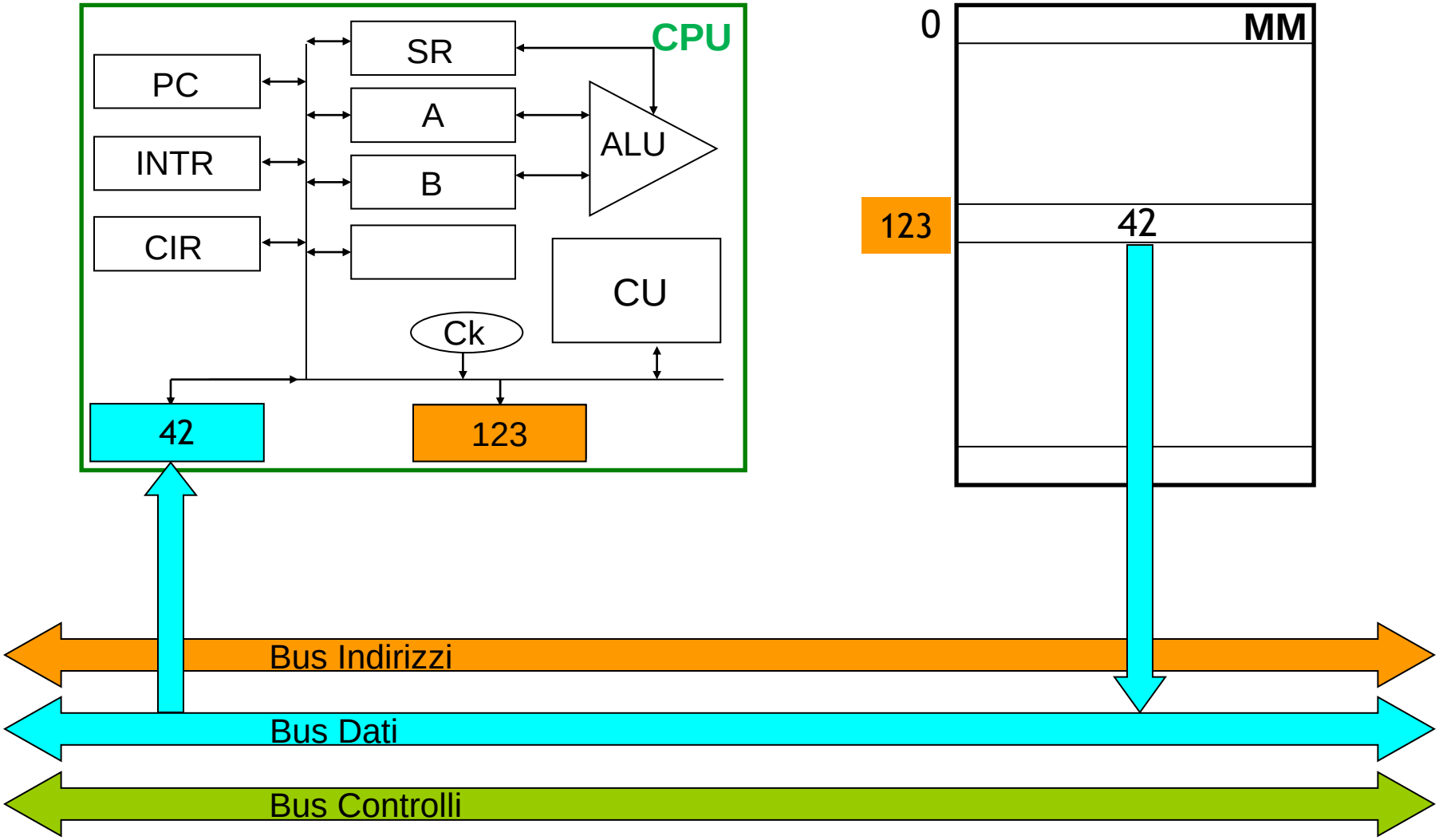
Sequenza di Lettura: Passo (1)



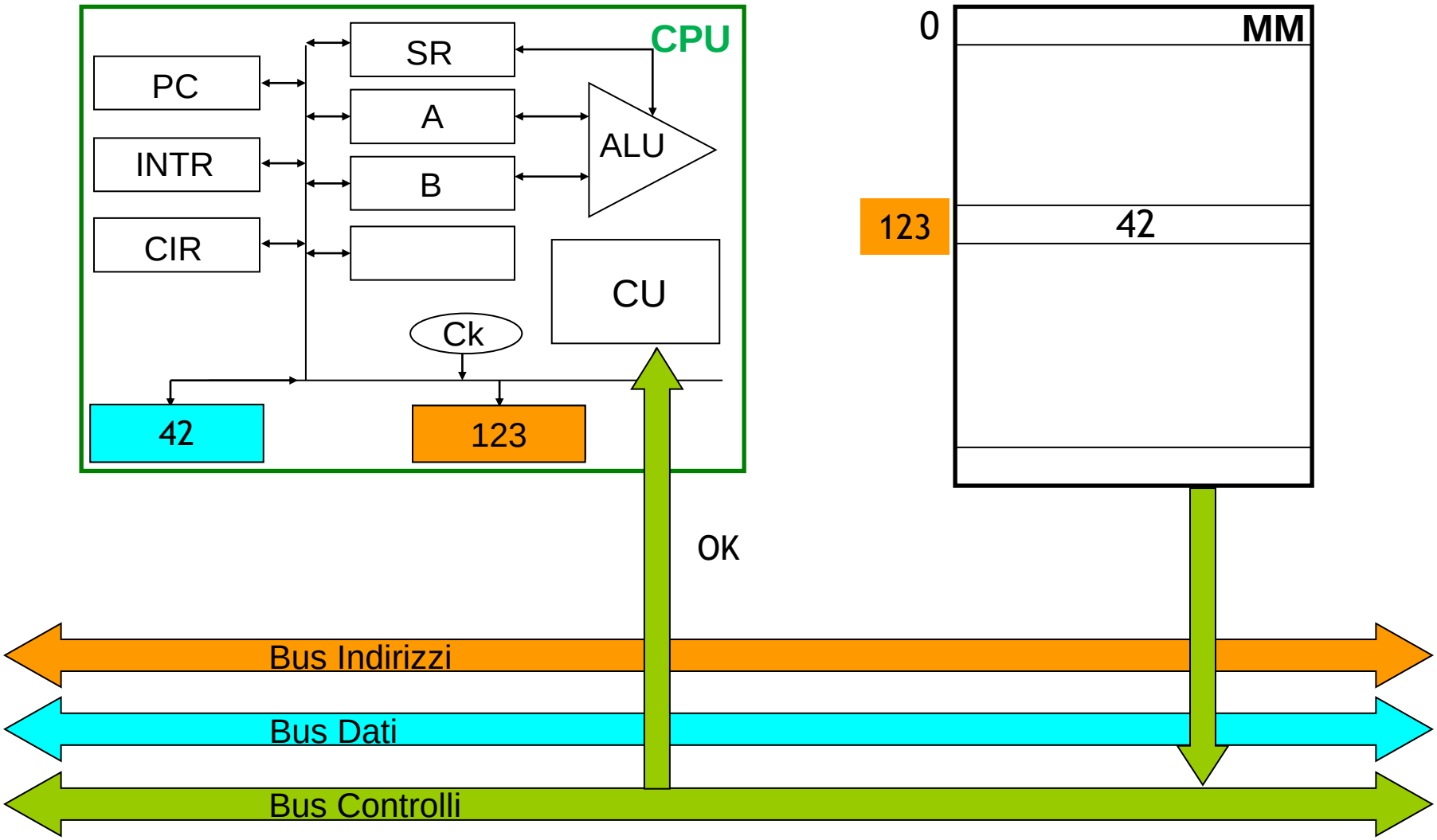
Sequenza di Lettura: Passo (2)



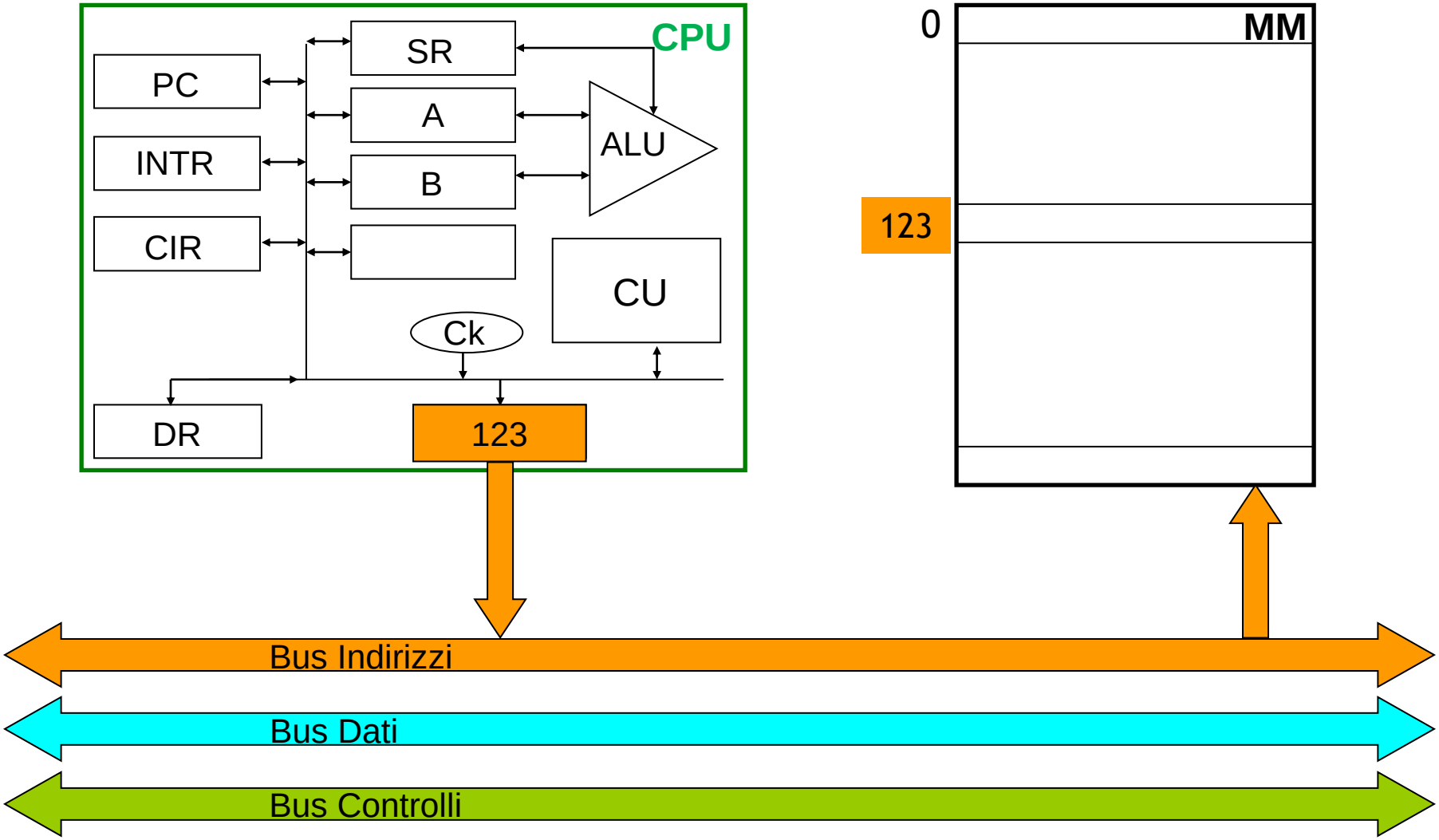
Sequenza di Lettura: Passo (3)



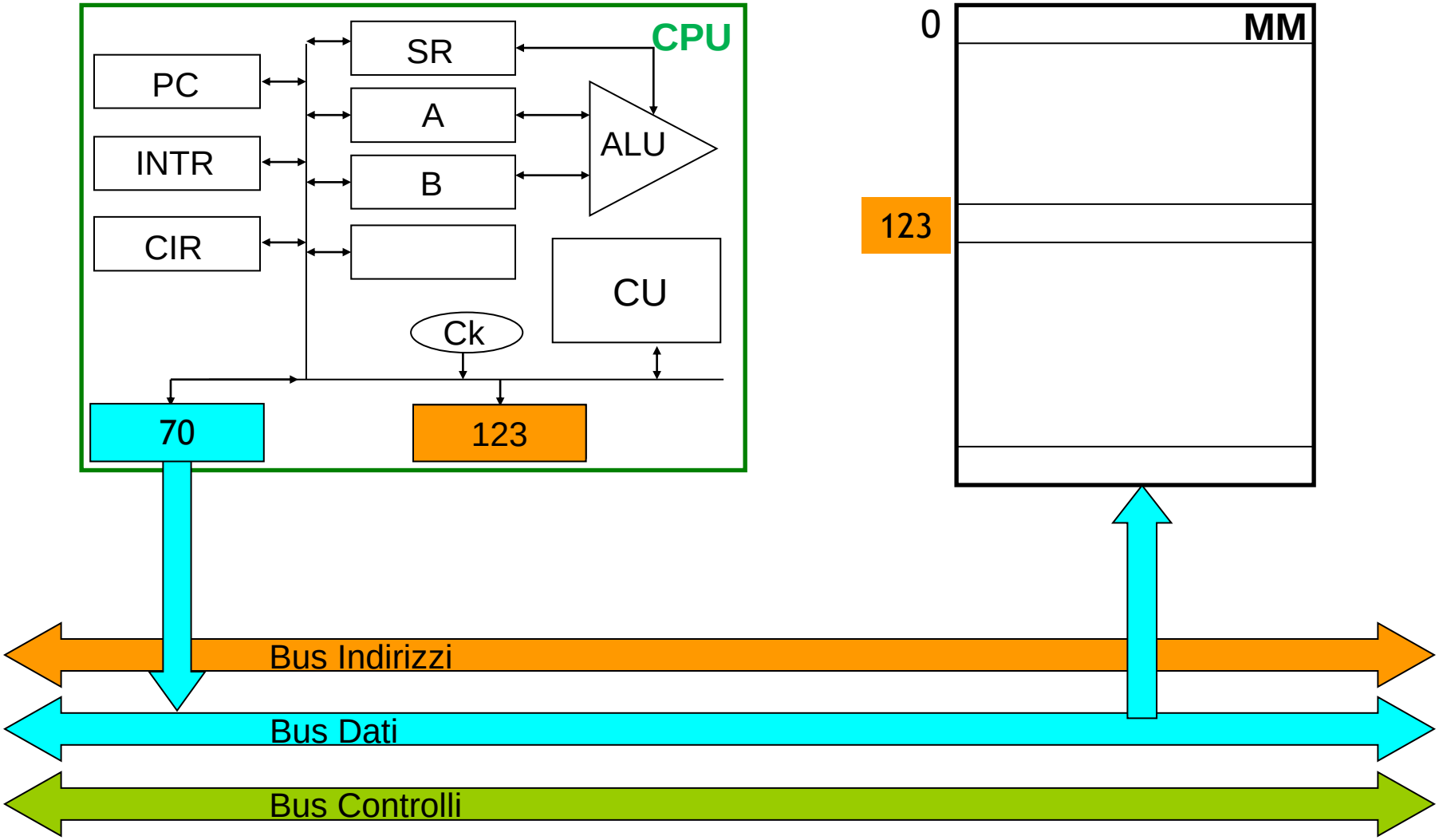
Sequenza di Lettura: Passo (4)



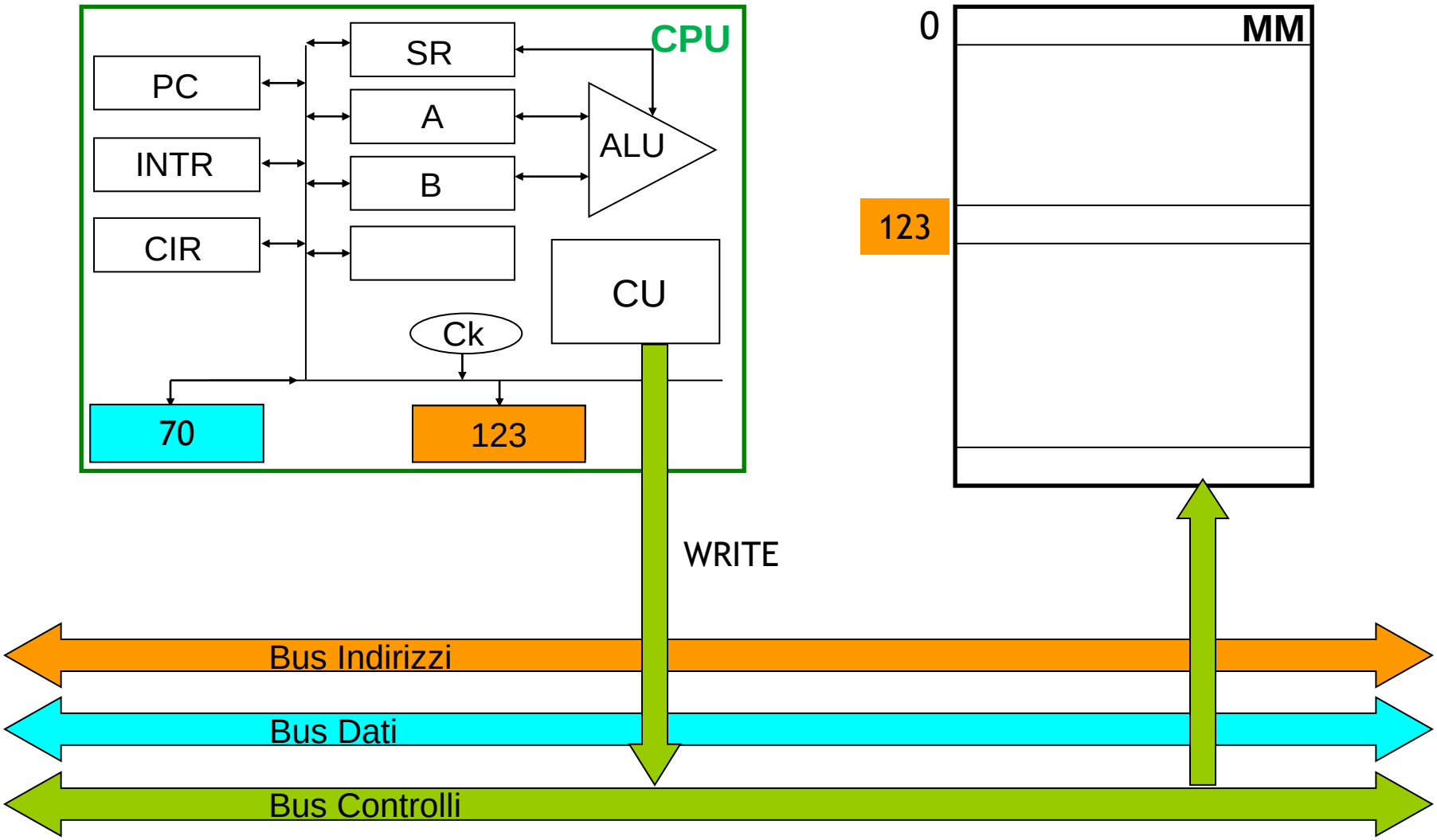
Sequenza di Scrittura: Passo (1)



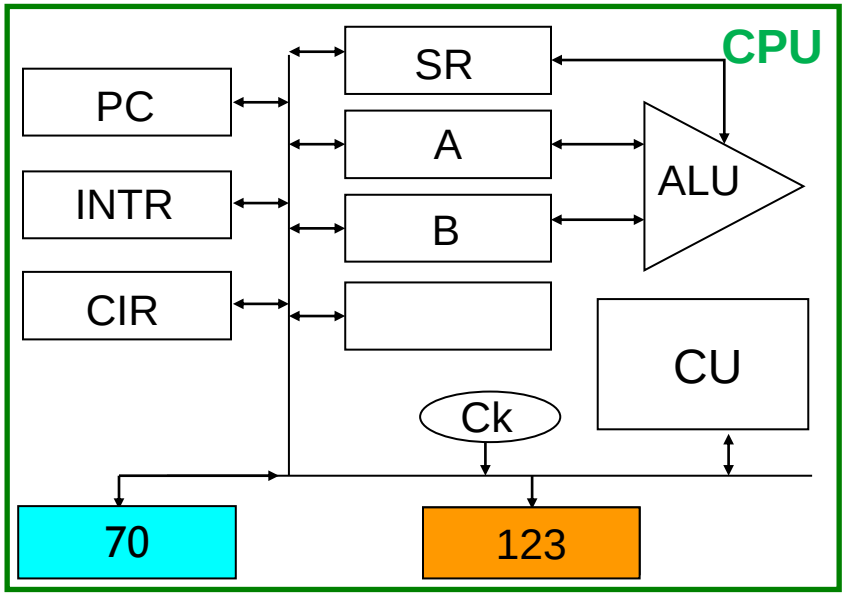
Sequenza di Scrittura: Passo (2)

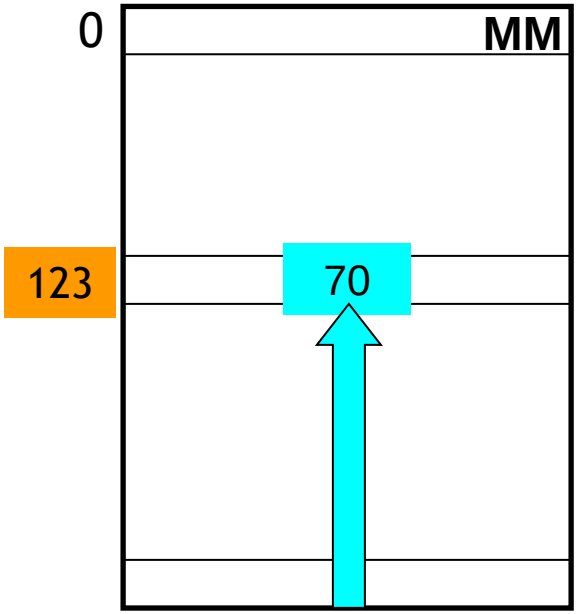


Sequenza di Scrittura: Passo (3)

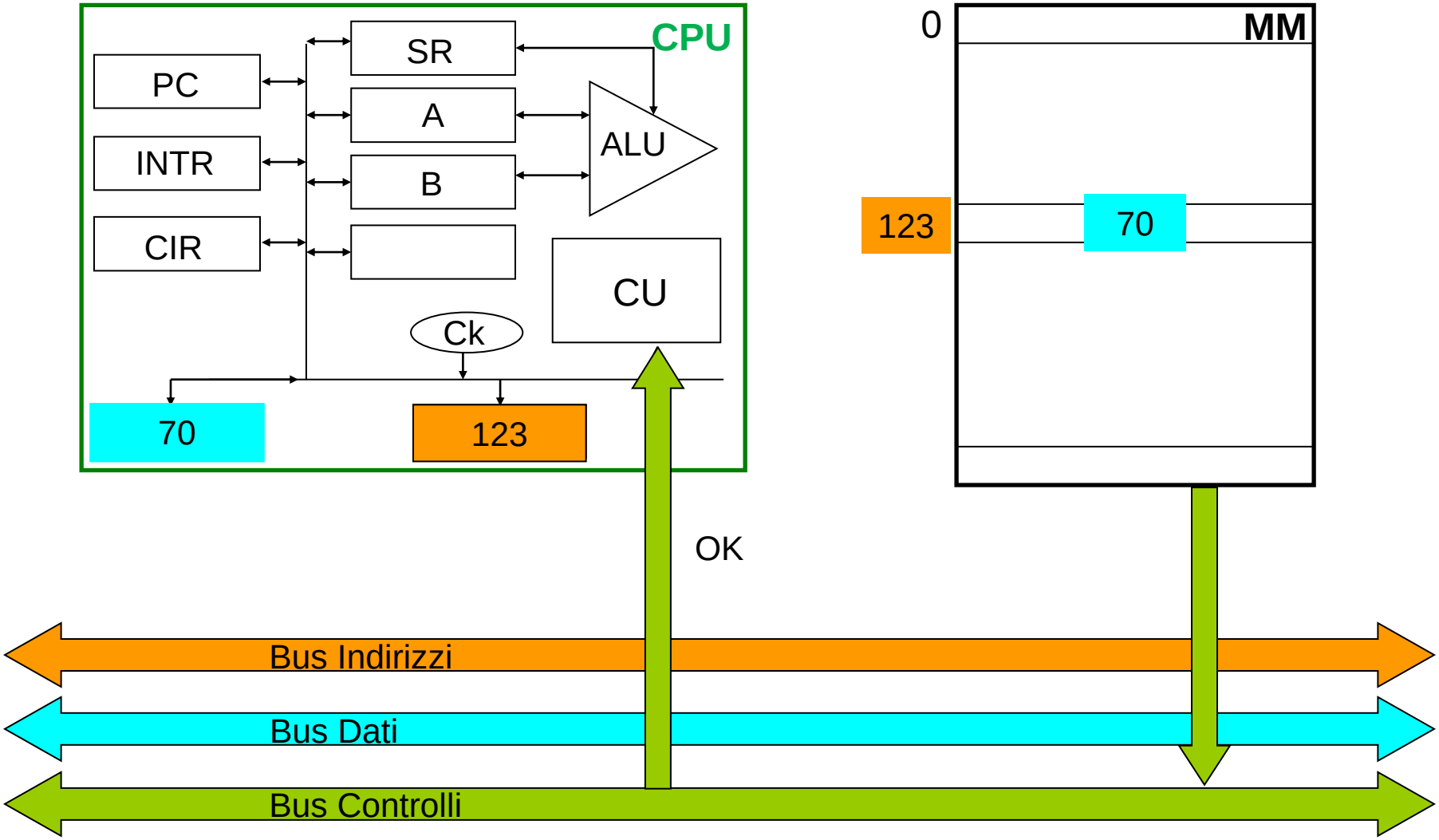


Sequenza di Scrittura: Passo (4)





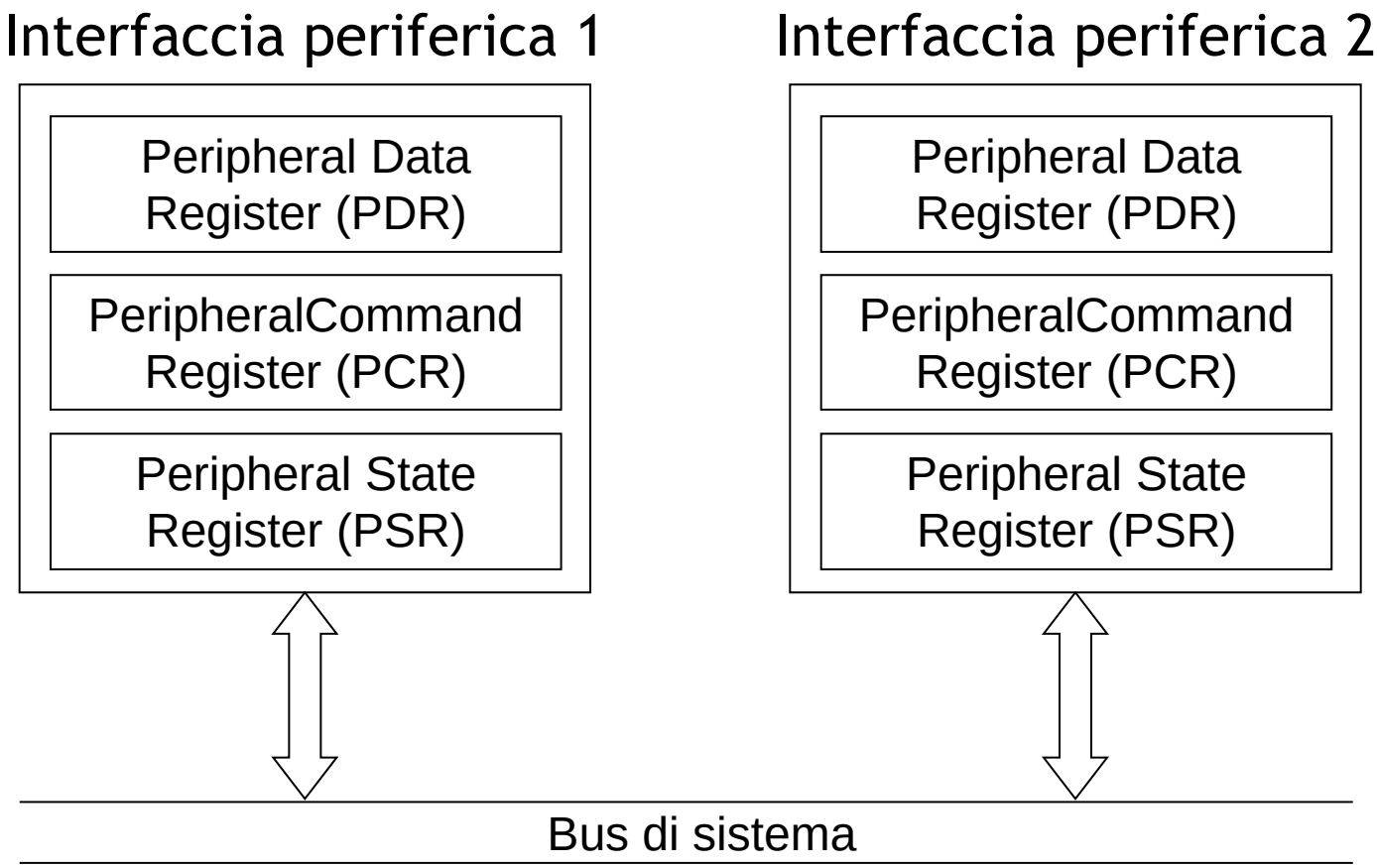
Sequenza di Scrittura: Passo (5)



Interfacce delle Periferiche

- Le interfacce collegano il calcolatore a periferiche esterne
- Ogni interfaccia contiene dei registri per lo scambio dei dati con la periferica
  - Registro dati della periferica (PDR): per scambiare dati con la periferica
  - Registro comando della periferica (PCR): contiene il comando dato alla periferica
  - Registro di stato (PSR): contienre le informazioni di stato della periferica
- Le periferiche comunicano solo con la CPU utilizzando i BUS

Interfacce delle 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**

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

Codice operativo (4bit) 00 Indirizzo Operando (10bit)

ad esempio, 0100000000010000

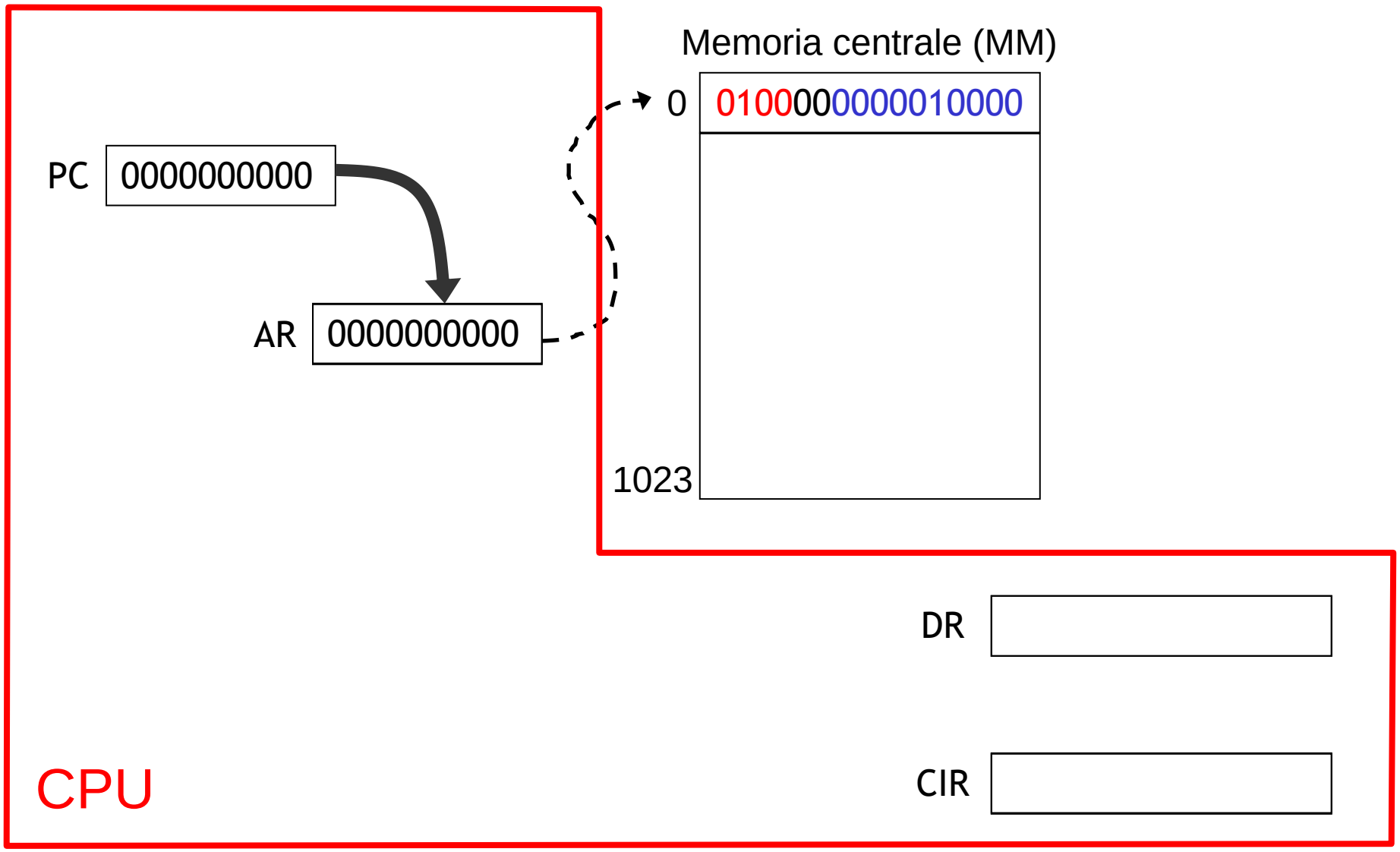
Consideriamo le seguenti istruzioni eseguibili dalla CPU

- 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)

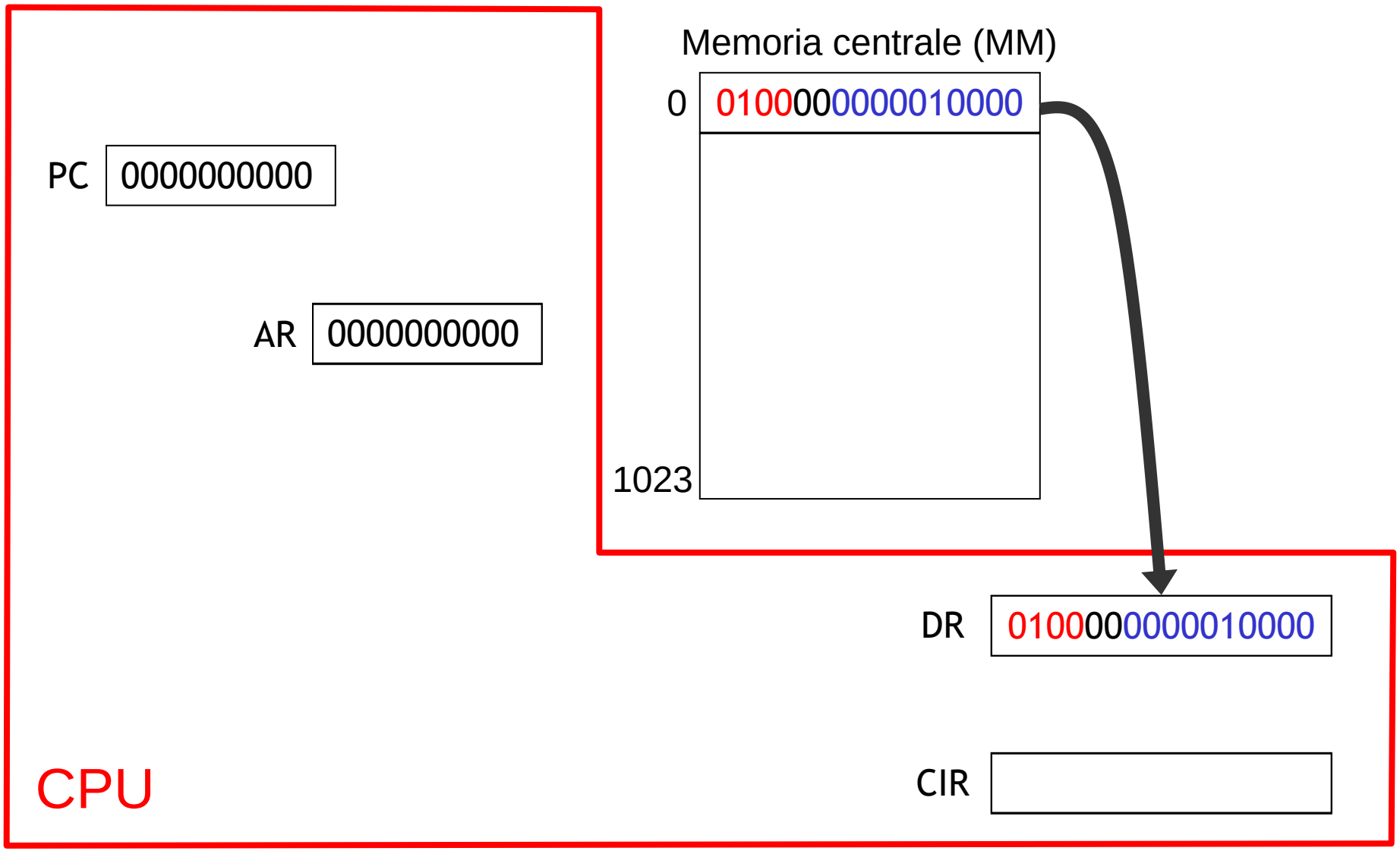
Le Tre Fasi Per Eseguire un'Istruzioni

1. **Fetch:** Acquisizione dell'istruzione dalla MM
   a. Trasferimento da PC ad AR dell'indirizzo cella contenente l'istruzione da eseguire
   b. Lettura dalla MM della cella all'indirizzo in AR (istruzione), contenuto trasferito sul DR
   c. Sposta da DR a CIR (riferimento istruzione in esecuzione)
   d. Incrementa PC
2. **Decodifica**: riguarda il codice operativo, legge dal CIR
3. **Esecuzione**: dipende dall'istruzione specifica

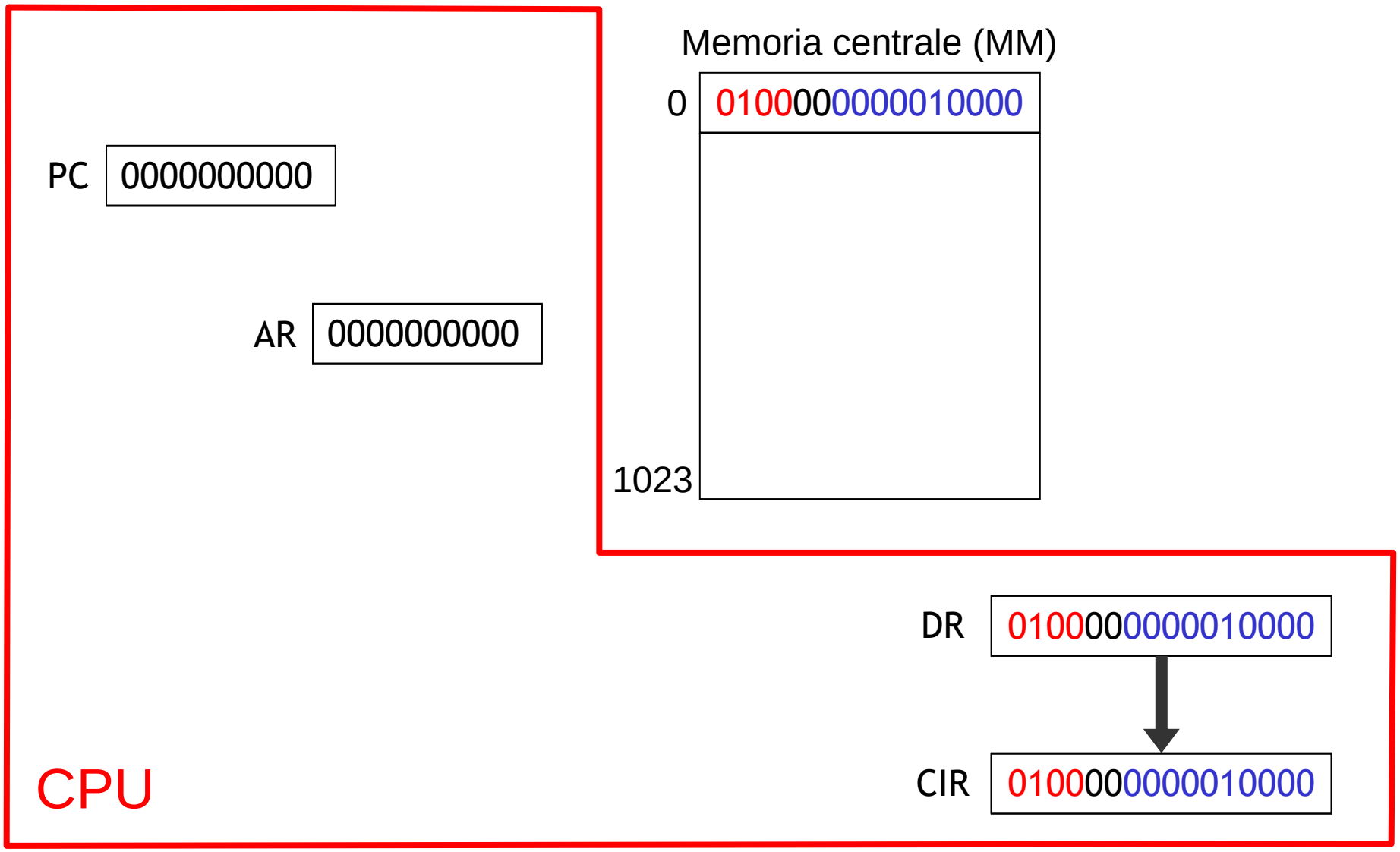
Fase di Fetch: Passo (1)



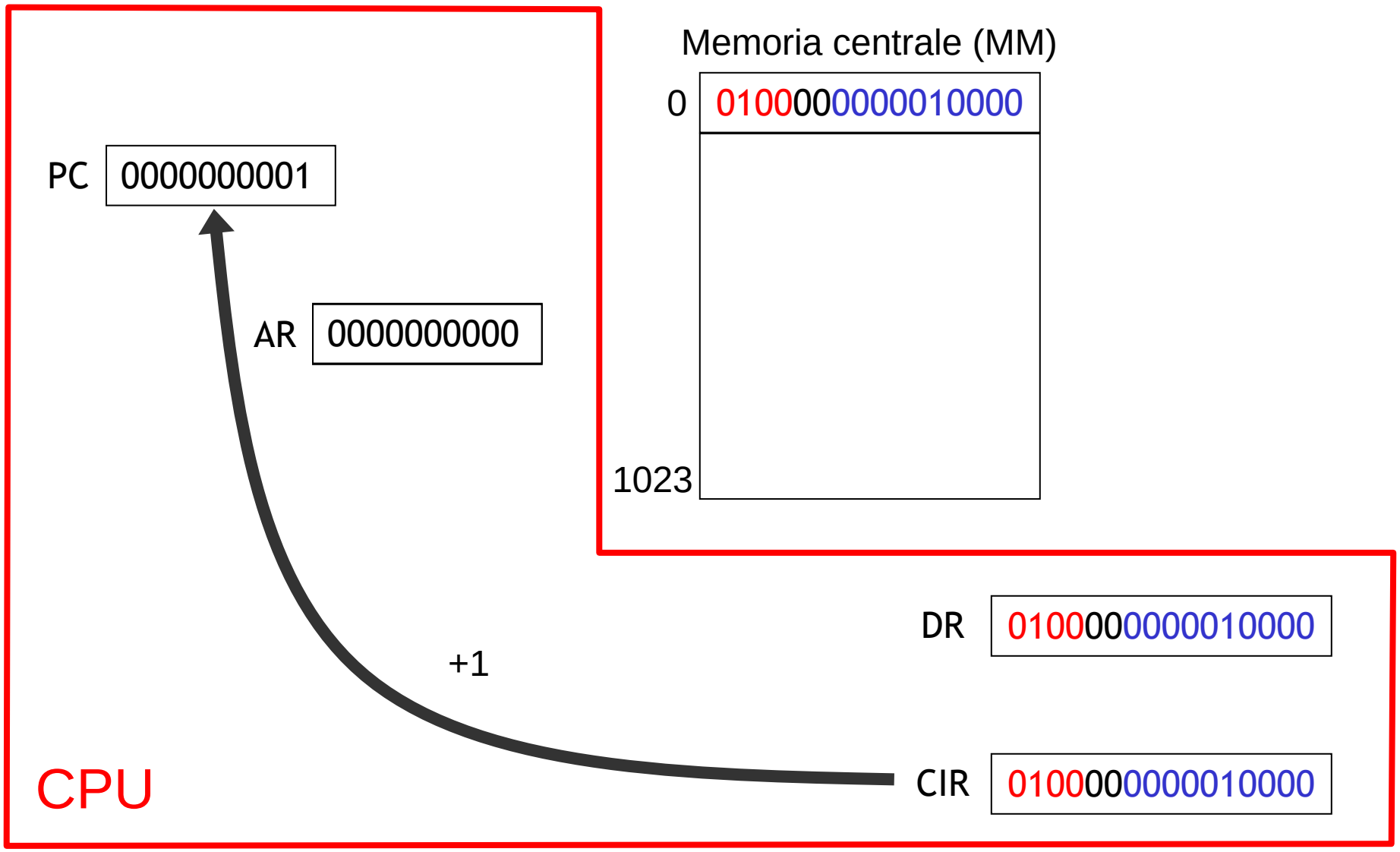
Fase di Fetch: Passo (2)



Fase di Fetch: Passo (3)

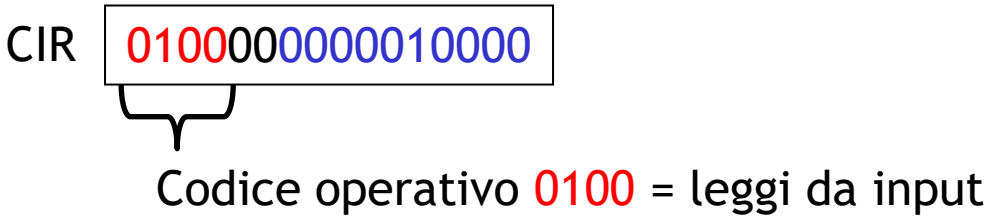


Fase di Fetch: Passo (4)

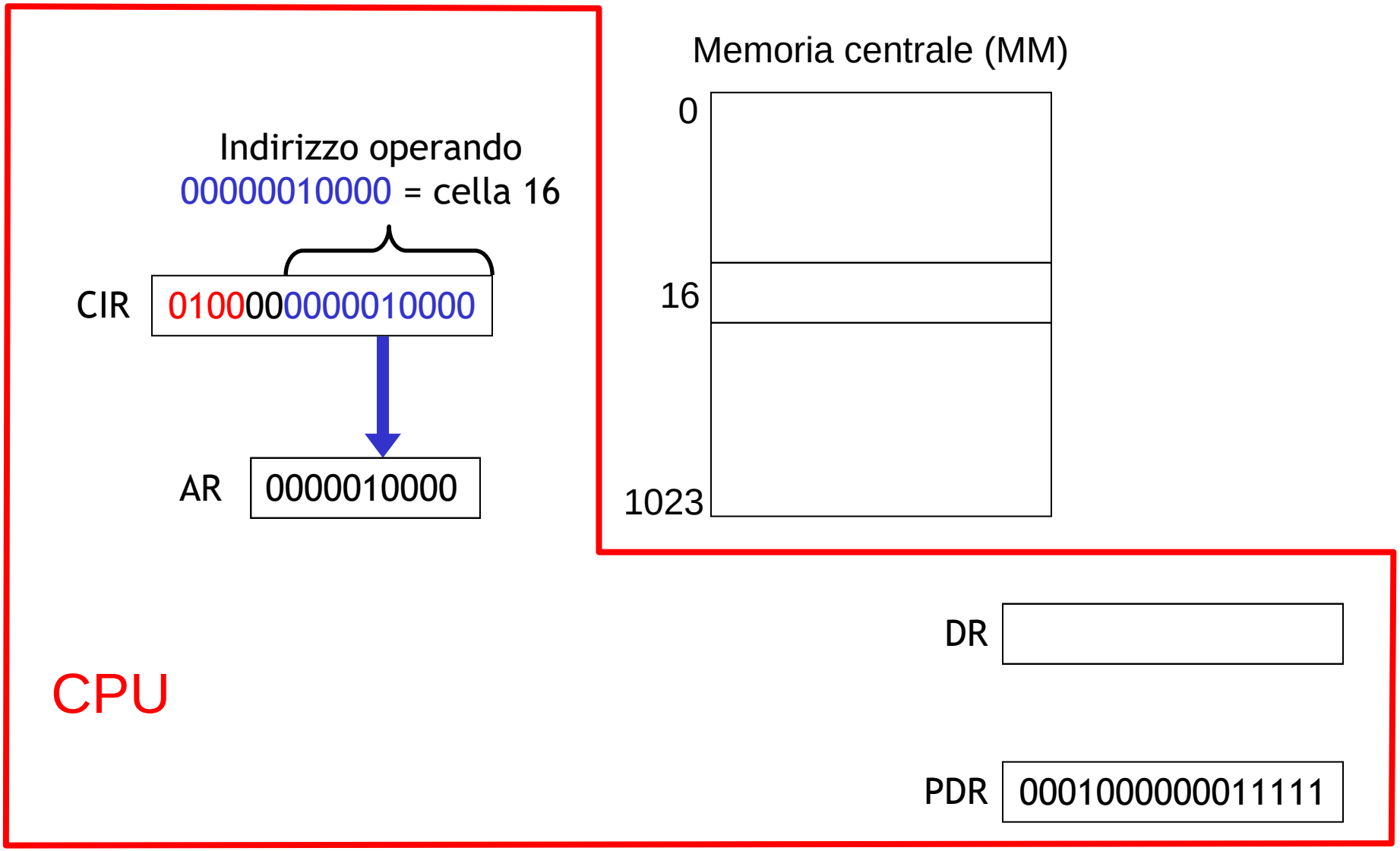


Fase di Decodifica

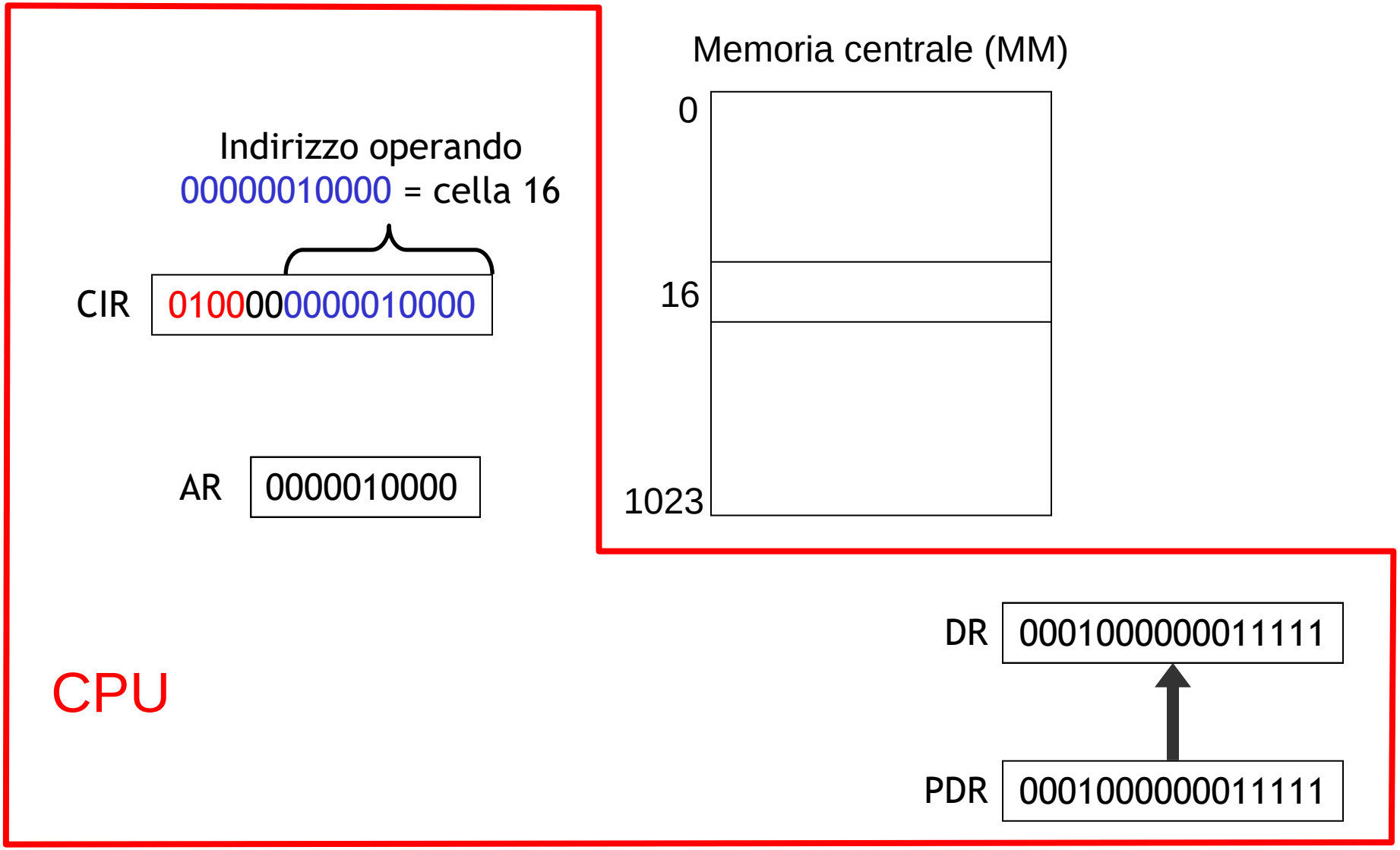
- Interpreto la parola (word) per capire quale operazione fare nella fase di esecuzione



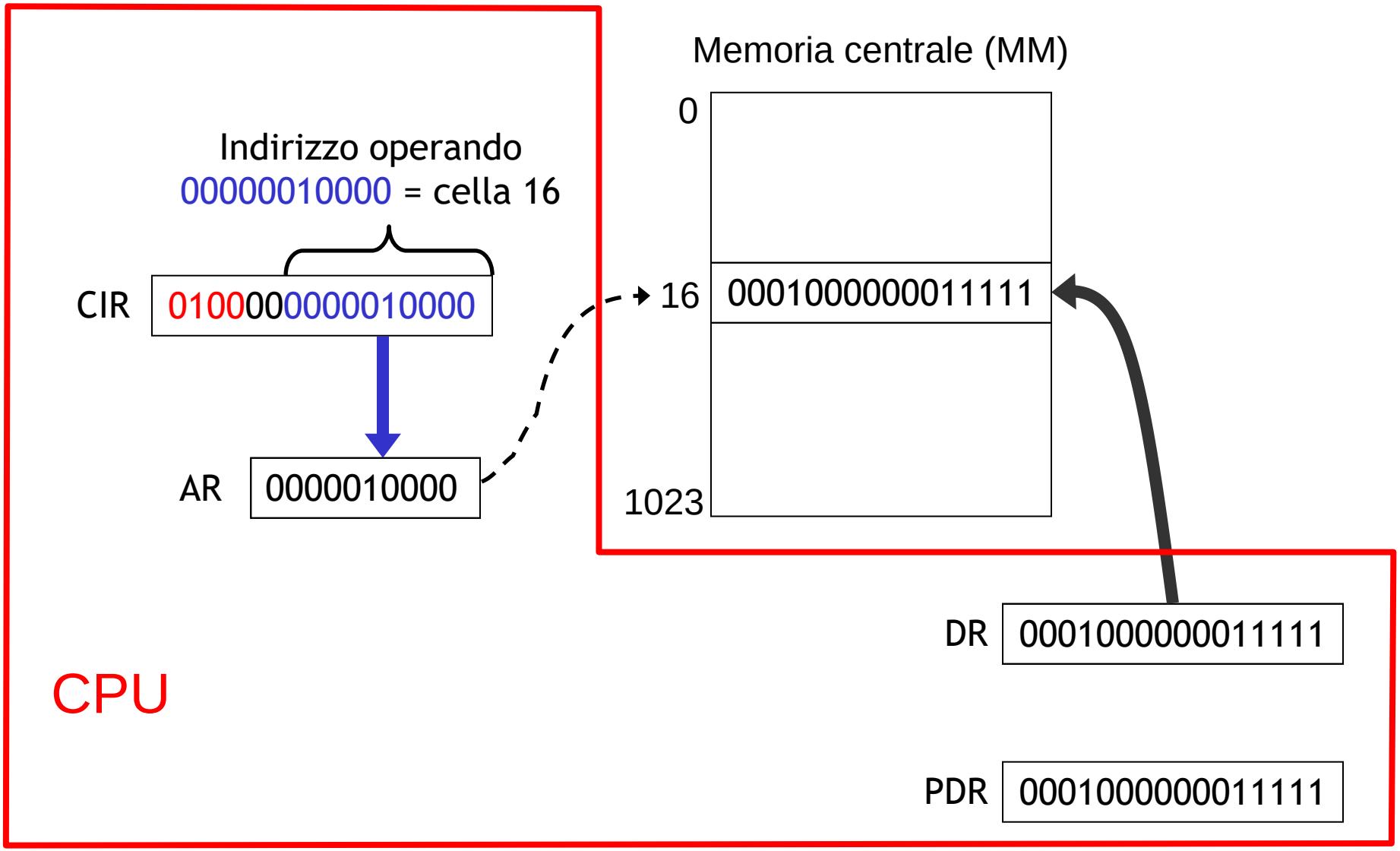
Fase di Esecuzione: Lettura da Input (1)



Fase di Esecuzione: Lettura da Input (2)



Fase di Esecuzione: Lettura da Input (3)



Esercizio

- 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) * (c+d)`

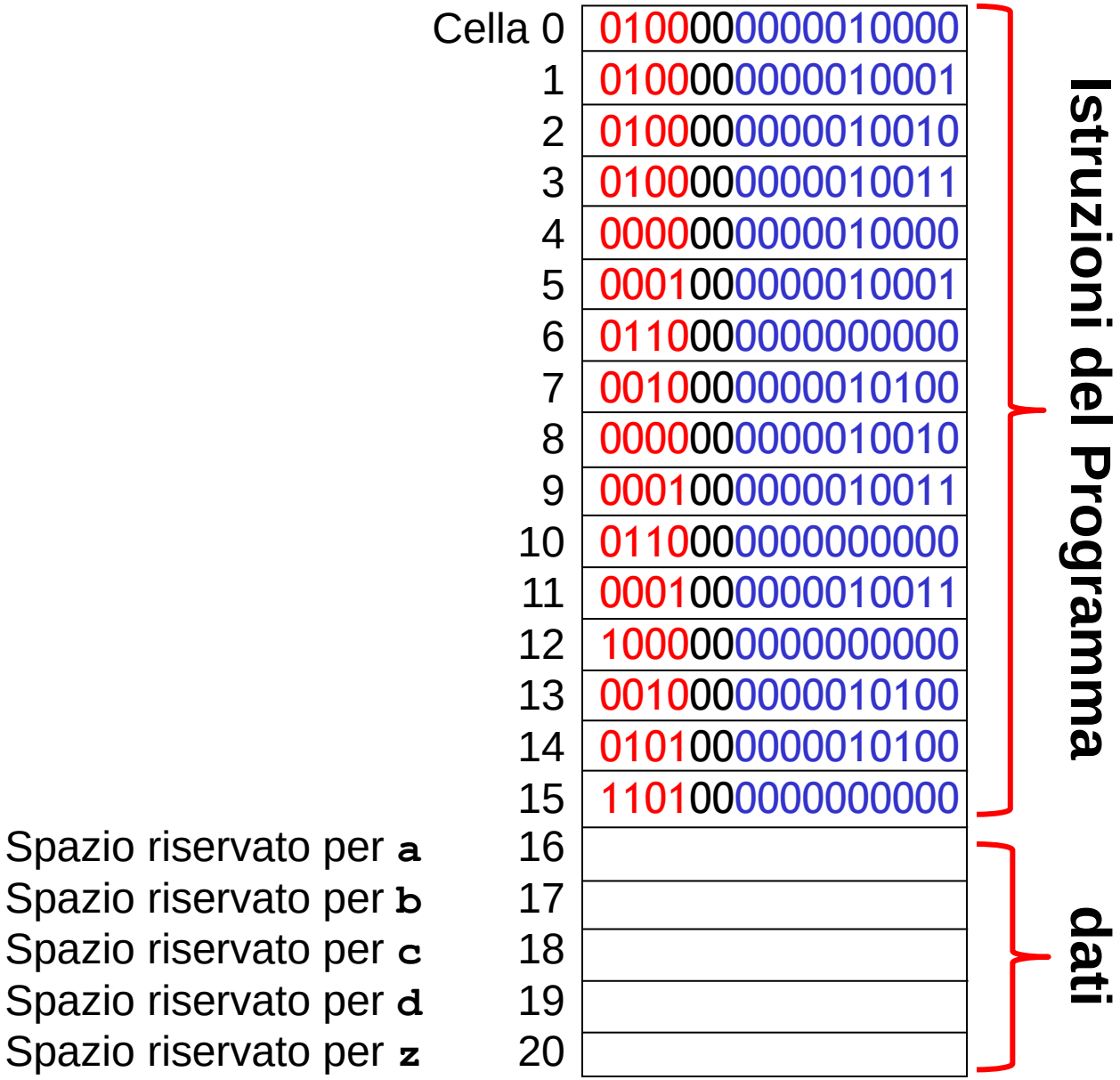
- Leggendo i valori delle variabili **`a`**, **`b`**, **`c`**, **`d`** dal dispositivo di ingresso e scrivendo il risultato della valutazione sul dispositivo di uscita

Esercizio

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!

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**