Introduzione al Corso

Informatica B, AA 2020/21

Luca Cassano

16 Settembre 2021

luca.cassano@polimi.it

Per Evitare Confusione

…

Prof. Masseroli ha lo scaglione PAA - SAL

Prof. Cassano ha lo scaglione SAM - ZZZ

La suddivisione degli studenti nelle varie sezioni potrebbe cambiare nei prossimi giorni (riguarda solamente gli studenti "vicini" a SAM).

Perché studiare informatica?

Perché studiare informatica?

- Aiuta a sviluppare la capacità di *modellizzazione* e *problem solving*

- Deve far parte del bagaglio culturale/interdisciplinare di ogni ingegnere contemporaneo (**e vi servirà!**)

- Applicazioni dell'informatica sono dappertutto
  - > Capire
  - > Progettare
  - > Realizzare

Chi siamo

Chi siamo

Luca Cassano (docente)

- Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano
- homepage: http://cassano.faculty.polimi.it/
- email: luca.cassano@polimi.it
- telefono: 02-23994174
- ricevimento studenti:
  - Martedì pomeriggio
    - Ufficio 129 via Ponzio 34/5, Milano (Edificio 20 Campus Leonardo).

In **ogni caso** è consigliabile **prendere appuntamento via email.**

Chi siamo

Luca Cassano - Home Page

lucacassano.xoom.it

Luca Cassano, Ph.D.
Assistant Professor

Department of Electronics, Informatics and Bioengineering
Politecnico di Milano
via Ponzio 34/5, 20133, Milano, Italy
tel.: +39-02-23993494;
e-mail: luca.cassano@polimi.it

I am an assistant professor at the Department of Electronics, Informatics and Bioengineering, Politecnico di Milano, Italy.

I received my Bachelor's degree and my Master's degree in Computer Engineering in 2006 and 2009 respectively from the University of Pisa, Italy. In 2013 I received my Ph.D. in Information Engineering from the Department of Information Engineering of the University of Pisa.
From March 2013 until June 2013 I have been a post doctoral research fellow at Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", National Research Council, under the supervision of Doctor Stefania Gnesi. From July 2013 until January 2016 I have been a post doctoral research fellow at the Dipartimento di Elettronica, Informazione e Bioingegneria of the Politecnico di Milano, Italy, under the supervision of Professor Cristiana Bolchini. Finally, from February 2017 until September 2017 I worked as an Associate Memebr of the Technical Staff at Maxim Integrated.
During my Ph.D. course I spent a visiting period at the Department of Automation and Informatics of the Politecnico di Torino, Italy, under the supervision of Doctor Luca Sterpone, and a visiting period at the Cognitive Interaction Technology - Center of Excellence (CITEC) of the University of Bielefeld, Germany, under the supervision of Professor Mario Porrmann.

With my Ph.D. thesis, titled "Analysis and Test of the Effects of Single Event Upsets Affecting the Configuration Memory of SRAM-based FPGAs", I won the European semi-finals of the TTTC's E. J. McCluskey Doctoral Thesis Award held during ETS2014, in Paderborn, Germany, on May 26-30 2014, and I was the runner-up at the Award finals held during ITC2014, in Seattle, USA, on November 21-23 2014.

Here you can find my CV.

Research activity:

My research interests are:

1) Analysis of the effects of SEUs in the configuration memory of SRAM-based FPGA systems.
2) Fault Simulation, Automatic Test Pattern Generation and fault untestability analysis for digital circuits and systems.
3) Use of machine learning and data mining techniques for fault testing and diagnosis for digital circuits and systems.
4) Energy-aware modeling and simulation of Automatic Weather Station.
5) Use of formal and semi-formal methods for dynamic system modelling and analysis (functional verification, safety analysis...)

Publications

Professional activity:

Here you can find the list of my editorial activities.

Teaching activity:

Informatica B (In Italian)

Chi siamo

Ing. Amarildo Likmeta (esercitatore)

- email: amarildo.likmeta@polimi.it

Ing. Luca Frittoli (responsabile di laboratorio)

Ing. Mirko Salaris (responsabile di laboratorio)

Il Corso

http://cassano.faculty.polimi.it/InformaticaB.html

Il corso

Luca Cassano - Home Page

lucacassano.xoom.it

Luca Cassano, Ph.D.
Assistant Professor

Department of Electronics, Informatics and Bioengineering
Politecnico di Milano
via Ponzio 34/5, 20133, Milano, Italy
tel.: +39-02-23993494;
e-mail: luca.cassano@polimi.it

I am an assistant professor at the Department of Electronics, Informatics and Bioengineering, Politecnico di Milano, Italy.

I received my Bachelor's degree and my Master's degree in Computer Engineering in 2006 and 2009 respectively from the University of Pisa, Italy. In 2013 I received my Ph.D. in Information Engineering from the Department of Information Engineering of the University of Pisa. From March 2013 until June 2013 I have been a post doctoral research fellow at Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", National Research Council, under the supervision of Doctor Stefania Gnesi. From July 2013 until January 2016 I have been a post doctoral research fellow at the Dipartimento di Elettronica, Informazione e Bioingegneria of the Politecnico di Milano, Italy, under the supervision of Professor Cristiana Bolchini. Finally, from February 2017 until September 2017 I worked as an Associate Memebr of the Technical Staff at Maxim Integrated. During my Ph.D. course I spent a visiting period at the Department of Automation and Informatics of the Politecnico di Torino, Italy, under the supervision of Doctor Luca Sterpone, and a visiting period at the Cognitive Interaction Technology - Center of Excellence (CITEC) of the University of Bielefeld, Germany, under the supervision of Professor Mario Porrmann.

With my Ph.D. thesis, titled "Analysis and Test of the Effects of Single Event Upsets Affecting the Configuration Memory of SRAM-based FPGAs", I won the European semi-finals of the TTTC's E. J. McCluskey Doctoral Thesis Award held during ETS2014, in Paderborn, Germany, on May 26-30 2014, and I was the runner-up at the Award finals held during ITC2014, in Seattle, USA, on November 21-23 2014.

Here you can find my CV.

Research activity:

My research interests are:

1) Analysis of the effects of SEUs in the configuration memory of SRAM-based FPGA systems.
2) Fault Simulation, Automatic Test Pattern Generation and fault untestability analysis for digital circuits and systems.
3) Use of machine learning and data mining techniques for fault testing and diagnosis for digital circuits and systems.
4) Energy-aware modeling and simulation of Automatic Weather Station.
5) Use of formal and semi-formal methods for dynamic system modelling and analysis (functional verification, safety analysis...)

Publications

Professional activity:

Here you can find the list of my editorial activities.

Teaching activity:

Informatica B (In Italian)

Il corso

Informatica B (da SAM a ZZZ)

Calendario del corso
Il calendario del corso e' scaricabile da qui.

Ricevimento: To Be Defined With Students, Dip. di Elettronica, Informazione e Bioingegneria (2° piano Edificio 20, Campus Leonardo)
Email: luca.cassano@polimi.it

Ambiente di Sviluppo utilizzato in Laboratorio di C: Code Blocks.
Ambiente di Sviluppo utilizzato in Laboratorio di Matlab: Matlab (scaricabile anche da qui).
In alternativa utilizzare Virtual Desktop

Testi di riferimento
G. Boracchi, E. Di Nitto, D. Loiacono, M. Masseroli, M. Santambrogio, V. Zaccaria, F. Fummi. Materiale su sistemi informatici e principi di programmazione in C per il corso di Informatica b. Ed. McGraw-Hill Education.
A. Campi, E. Di Nitto, D. Loiacono, A. Morzenti, P. Spoletini. Introduzione alla programmazione in Matlab. Ed. Progetto Leonardo.

Materiale delle lezioni

Materiale delle esercitazioni

Materiale dei laboratori

Temi d'esame passati (dal 2008 al 2016) possono essere scaricati qui.

Home Page

Organizzazione

Lezioni: ~33 ore su WebEX Cassano

il Giovedì dalle 8.15 alle 11.15

Esercitazioni: ~24 ore in LM.1 e contemporaneamente su WebEX (Cassano o Likmeta)

Il Lunedì dalle 15.15 alle 17.15

- Studenti con codice persona pari

Il Lunedì dalle 17.15 alle 19.15

- Studenti con codice persona dispari

Organizzazione

Laboratori: 15 ore (5 incontri)
il Mercoledì dalle 8:15 alle 11:15

- Online nella stanza WebEX di Salaris

Controllate sempre il **calendario** del corso:

http://cassano.faculty.polimi.it/calendario.pdf

Il calendario

Argomenti Trattati

- Introduzione all'informatica
- Introduzione alla programmazione
- Fondamenti della programmazione in linguaggio C

Primo emisemestre

- Rappresentazione binaria dei numeri interi

- Fondamenti della programmazione in Matlab
- Argomenti avanzati di programmazione
- Composizione e organizzazione dei sistemi informatici

Secondo emisemestre

Bibliografia

G. Boracchi, E. Di Nitto, D. Loiacono, M. Masseroli, M. Santambrogio, V. Zaccaria, F. Fummi

"*Materiale su sistemi informatici e i principi di programmazione in C per il corso di Informatica B* "

Editore: Mc Graw Hill (*)

A. Campi, E. Di Nitto, D. Loiacono, A. Morzenti, B. Spoletini,

"*Introduzione alla programmazione in Matlab*", seconda edizione.

Bibliografia (nota importante)

Questo testo contiene i capitoli del "*Informatica Arte e Mestiere*“ che sono rilevanti per il corso di Informatica B.

Se siete in possesso di "*Informatica Arte e Mestiere*“, non dovete acquistare anche questo.

Cosa Imparerete:

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

La Pagina del Corso

La pagina del corso è

http://cassano.faculty.polimi.it/InformaticaB.html

Troverete:

- Materiale didattico usato a lezione (queste slides sono da considerare **un supporto** allo studio)
- Il materiale delle esercitazioni e dei laboratori
- Temi d'esame con soluzioni
- Calendario del corso (lezioni, esercitazioni, laboratori)
- Avvisi, esiti esami

Le esercitazioni

- Durante le esercitazioni verranno approfonditi gli argomenti trattati a lezione con ulteriori esempi pratici svolti alla lavagna
- Studenti divisi in due turni
  Il Lunedì dalle 15.15 alle 17.15
  - Studenti con codice persona pari

  Il Lunedì dalle 17.15 alle 19.15
  - Studenti con codice persona dispari
- Sarete assistiti da:
  - Un esercitatore
- La prima esercitazione si terrà Lunedì 20 Settembre

I Laboratori

- Nei laboratori vi sarà richiesto di **sviluppare autonomamente** degli esercizi.
- Tutti gli studenti nello stesso orario (un solo turno)
  - Pagina WebEX di Salaris
- Sarete assistiti da :
  - Due responsabili di laboratorio
  - Due/tre tutors (?!)

I Laboratori

- Utilizzare il **proprio laptop**,
  - Installare Code::Blocks per il C, versione con compilatore (http://www.codeblocks.org/)
  - Installare Matlab (per il secondo emisemestre)
- In alternativa è possibile usare il servizio VirtualDesktop

Il **primo laboratorio** si Mercoledì 13 Ottobre

(**controllare sempre** il calendario del corso!!!)

L’Esame ed il Ricevimento Studenti

Modalità di Verifica

Si terranno solo appelli regolari (**no prove in itinere**):

- **Esame scritto** su tutto il programma
  - Esercizio di programmazione C
  - Esercizio di programmazione Matlab
  - Terzo esercizio variabile
  - 2 domande di teoria

- NON è prevista una prova orale

Il laboratorio non sarà valutato (ma è molto molto utile)

Ricevimento Studenti

Domande e richieste di chiarimenti in aula sono sempre ben accette (**soprattutto durante la lezione a beneficio di tutta l'aula!!!**)

Ricevimento Studenti

Potete anche rivolgere domande via mail

In particolare:

- 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 dell'esercizio
- Ponete domande CHIARE!

Ricevimento Studenti

Potete chiedere un ricevimento «tradizionale» sia in presenza nel mio ufficio che online.

…prima di iniziare per davvero…

Raccomandazioni / Consigli / Note

- Per chi ha già programmato: le prime lezioni / esercitazioni potrebbero essere un po' noiose: **Abbiate pazienza**
- Per chi non ha mai programmato: le prime lezioni / esercitazioni potrebbero spaventare: **Abbiate coraggio**
- Per tutti: "SFRUTTATE" l'università e vivetela attivamente
  - Frequentate
  - Fate domande e partecipate al ricevimento studenti
  - Non trascinatevi: l'università non finisce da sé, la dovete finire voi!
- **STUDIATE**

Raccomandazioni / Consigli / Note

Ma soprattutto:

siate CONSAPEVOLI

LET’S ROCK!

Cos’è l’informatica?

Scienza della rappresentazione e dell’elaborazione dell’informazione.

[da «Informatica Arte e Mestiere»]

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.

[da «Informatica Arte e Mestiere»]

Cos'è l'Informatica?

*Studio sistematico degli **algoritmi** che descrivono e trasformano l'informazione:*

- la loro teoria,
- analisi,
- progetto,
- efficienza,
- realizzazione,
- applicazione.

[da Association for Computing Machinery (ACM)]

Gli Algoritmi

Algoritmo

Sequenza** di **istruzioni**, definite con **precisione**, che portano alla **realizzazione di un compito

Le istruzioni devono:

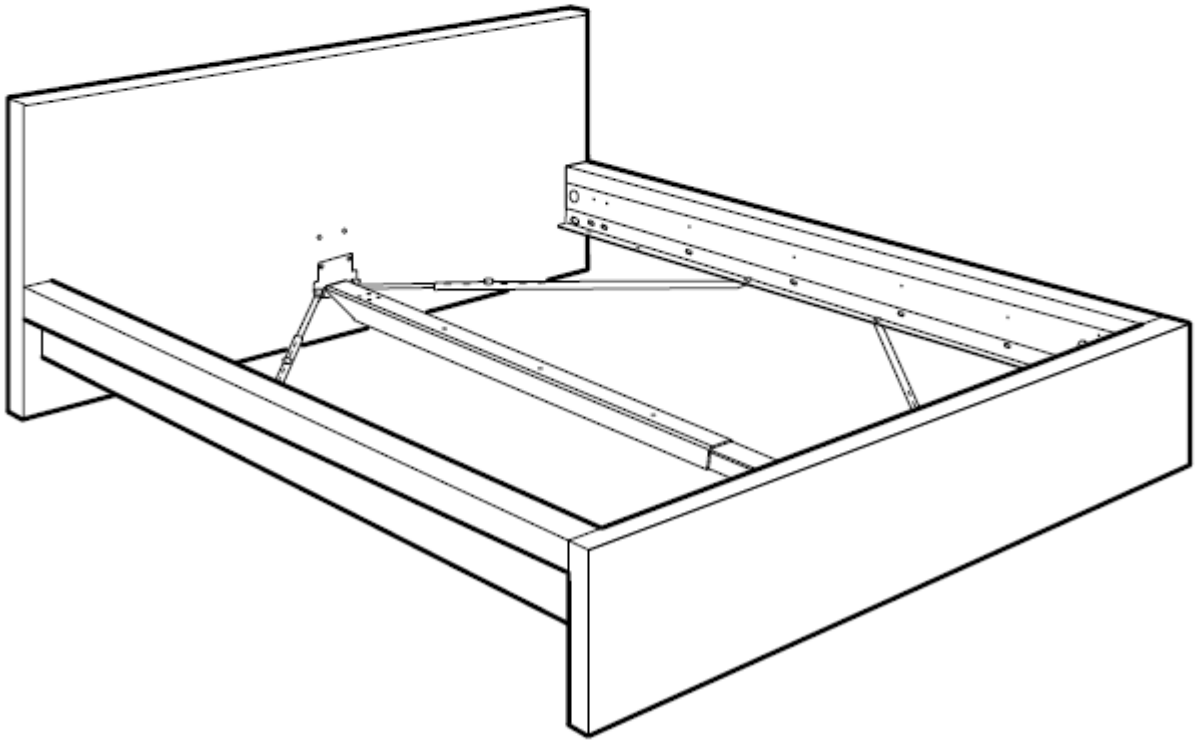
- essere **comprensibili** senza ambiguità
- essere **eseguibili** da uno strumento automatico: l'esecutore
- portare a realizzare un compito in **tempo finito** (devono contenere un numero finito di passi, ciascuno eseguibile in tempo finito)

Non è necessario un calcolatore per parlare di informatica!

Algoritmo

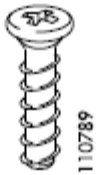
…….ora vedermo un esempio di algoritmo in cui vi sarà capitato di essere esecutori….

MALM

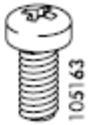


IKEA®
Design and Quality
IKEA of Sweden

12x



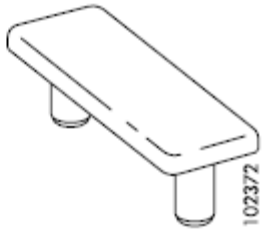
20x/22x



8x

12x

8x

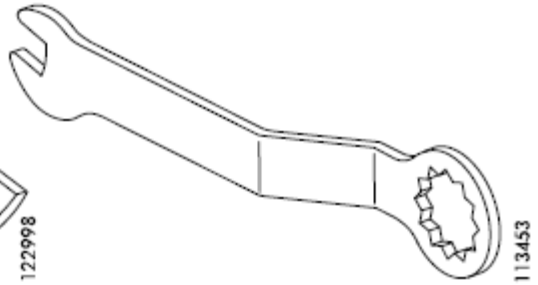


6x

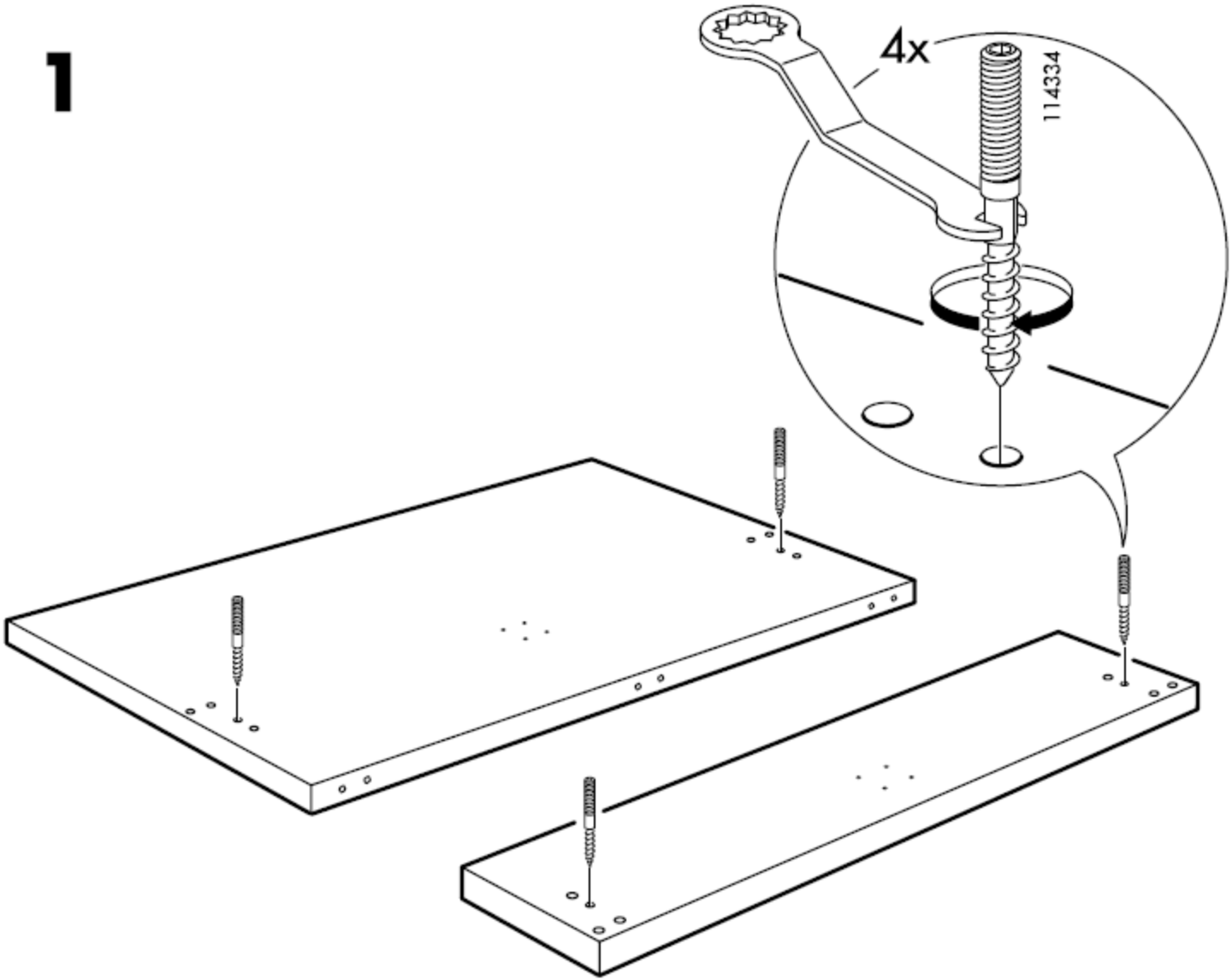
4x

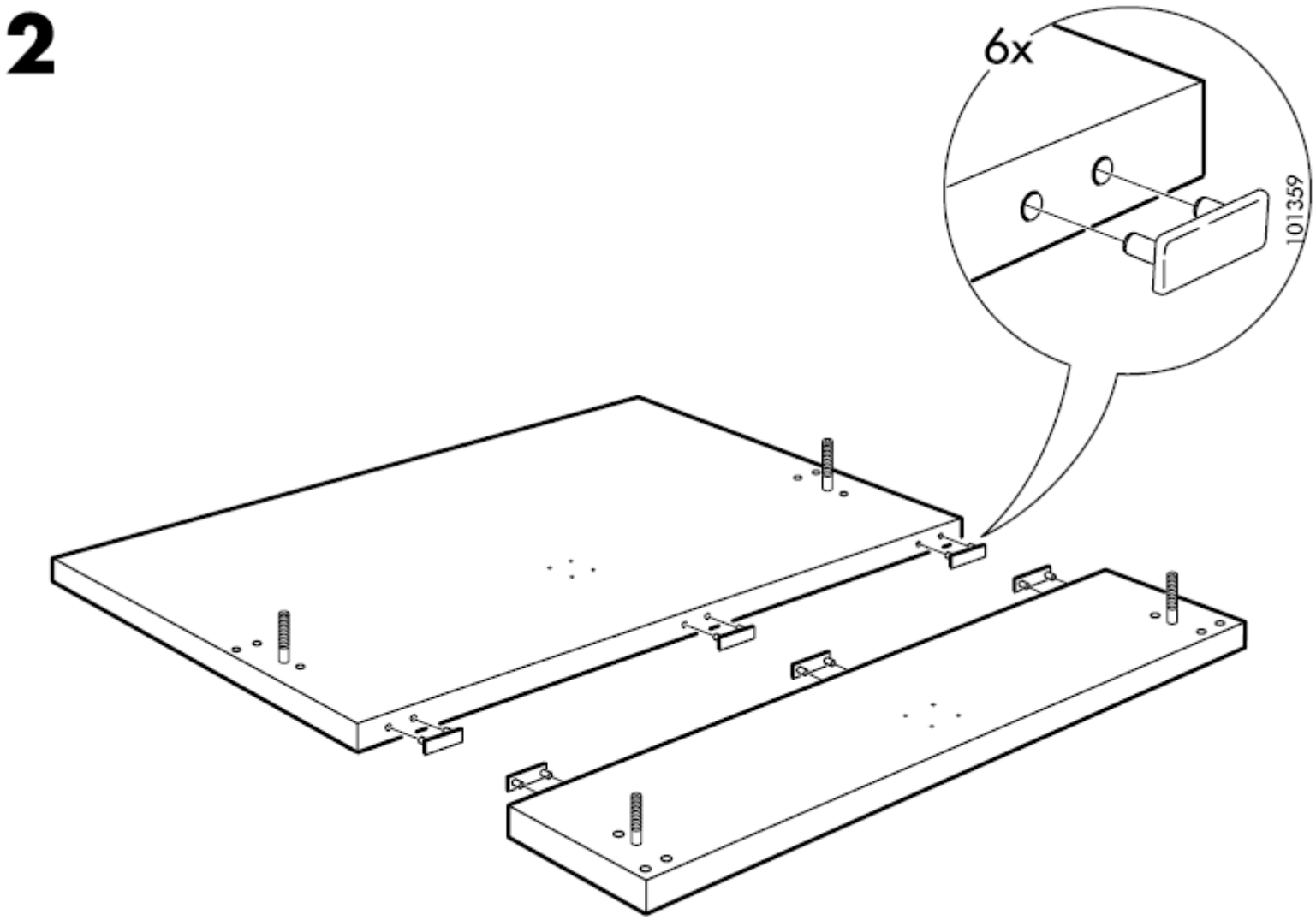
4x

4x



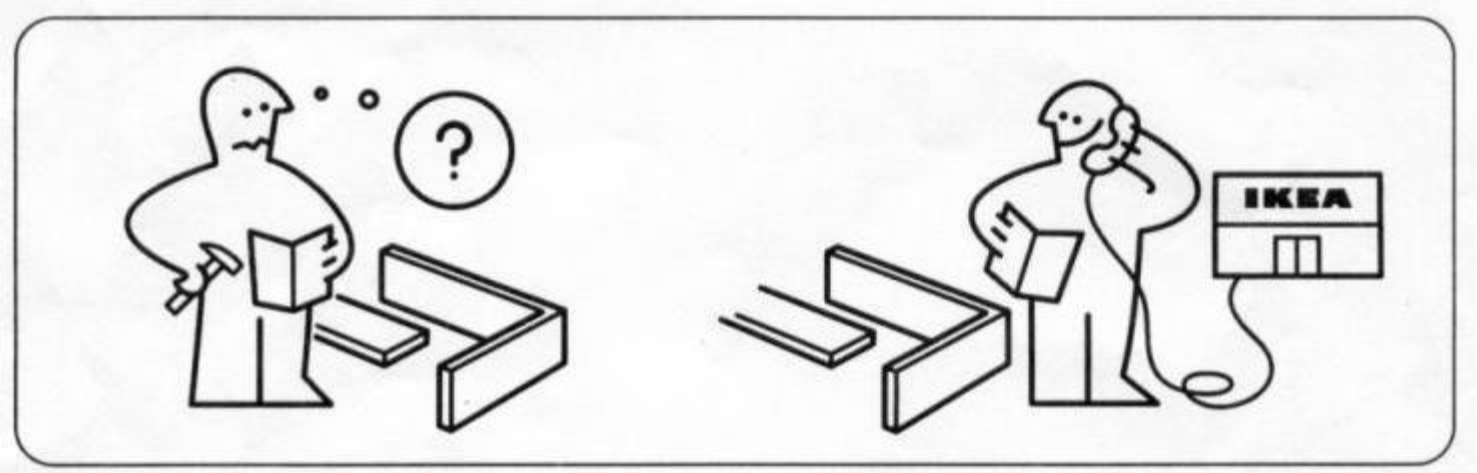
1x


1
4x
114334

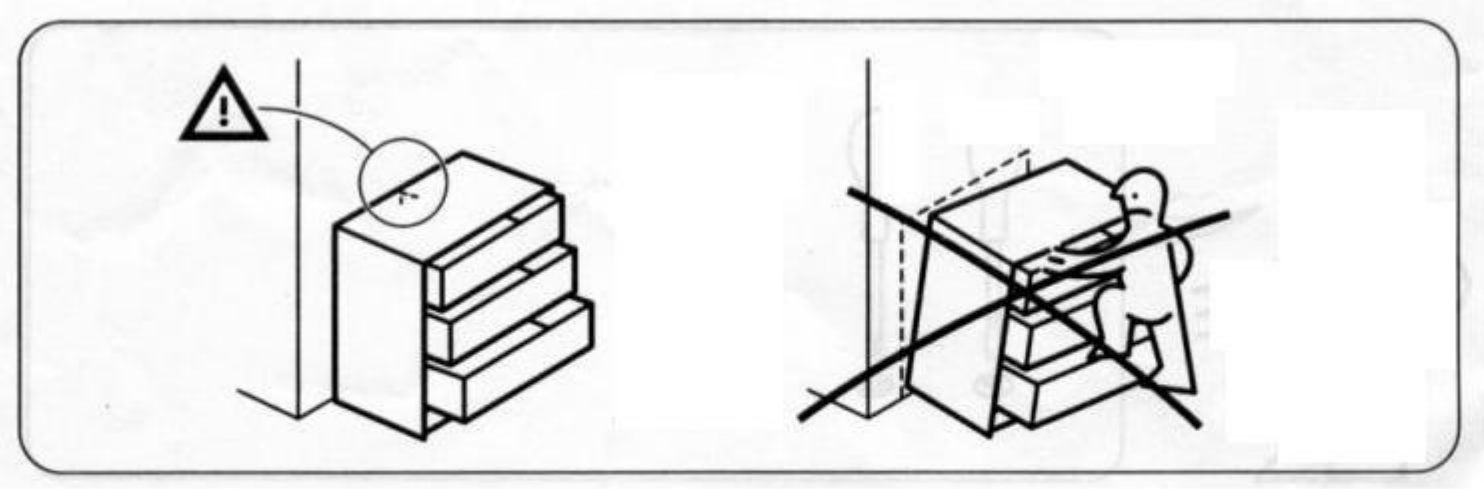

2
6x
101359

Il Linguaggio «IKEA»

- Le istruzioni IKEA sono fatte per esecutori intelligenti (nello specifico…noi) l'interpretazione dei disegni richiede diverse capacità



Il Linguaggio «IKEA»



- Quando l’esecutore è meno intelligente occorre esprimere le istruzioni in un linguaggio più preciso

Definiamo un nostro linguaggio per gli algoritmi!

Svilupperemo i prossimi algoritmi in italiano (utilizzando un linguaggio molto essenziale).

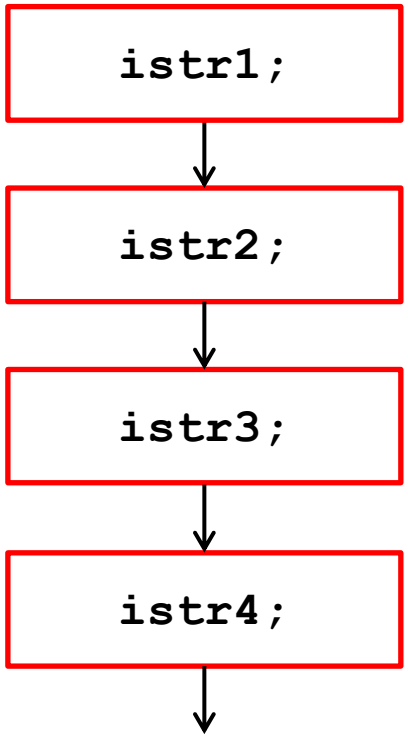
In particolare, il linguaggio sarà caratterizzato da:

- Sequenzialità delle istruzioni
- Costrutto condizionale
- Costrutto iterativo

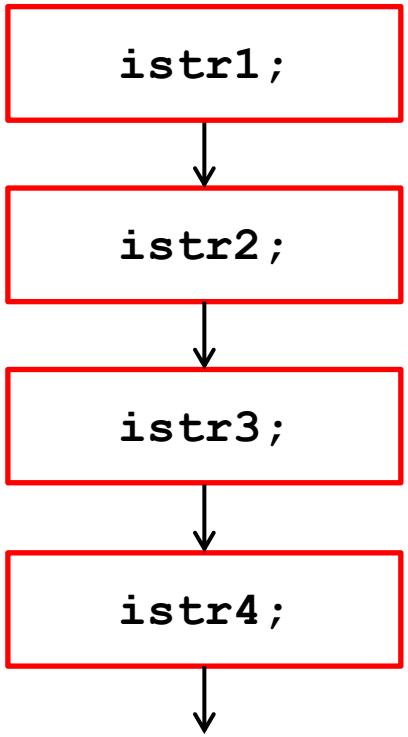
Inoltre, potremo utilizzare “foglietti” dove scrivere (registrare) alcuni valori

La Sequenzialità

La sequenzialità



La sequenzialità




```
istr1;
istr2;
istr3;
istr4;
...
```

Le istruzioni vengono eseguite dalla prima all'ultima.

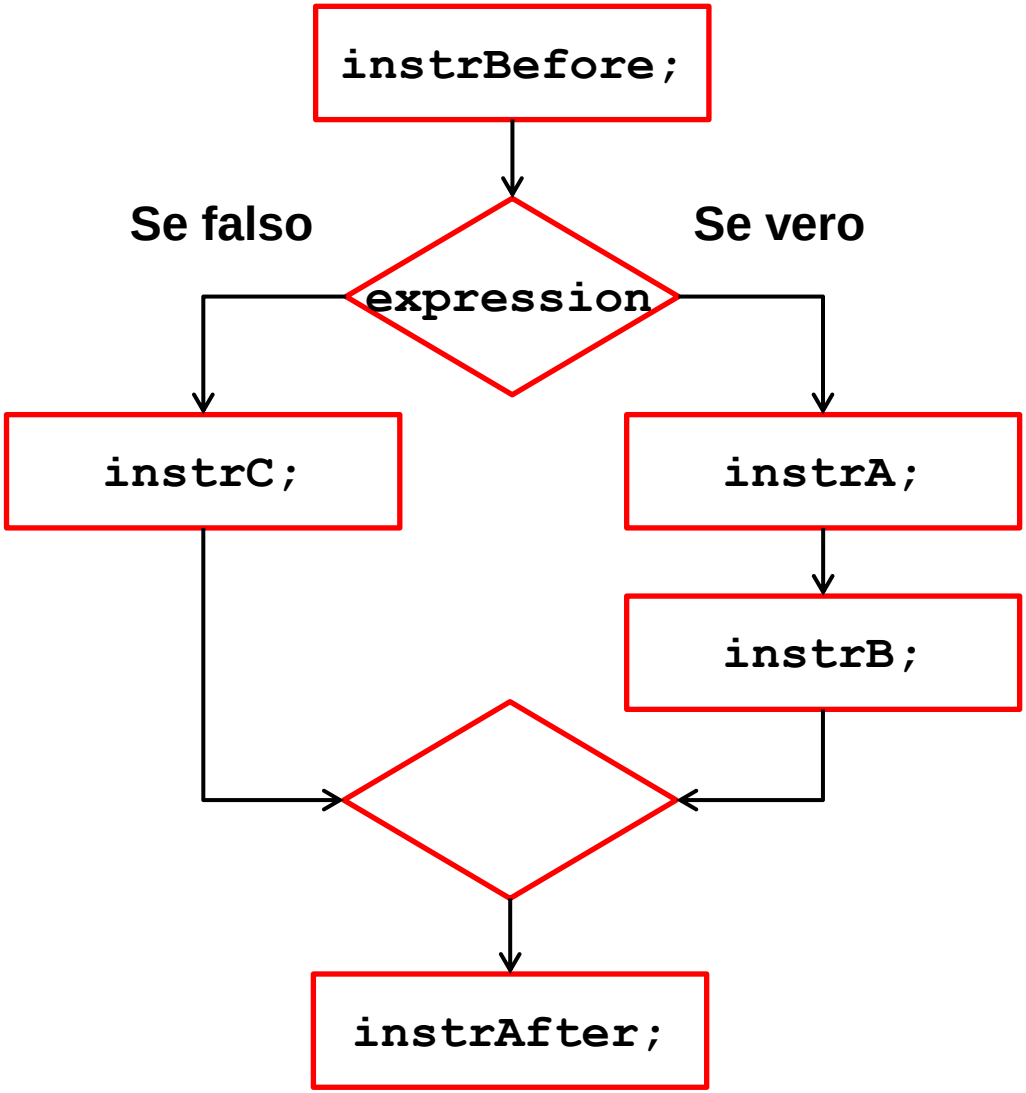
Terminata la *i-sima* istruzione, si esegue la *(i+1)-sima*

Esempio: algoritmo per andare in università

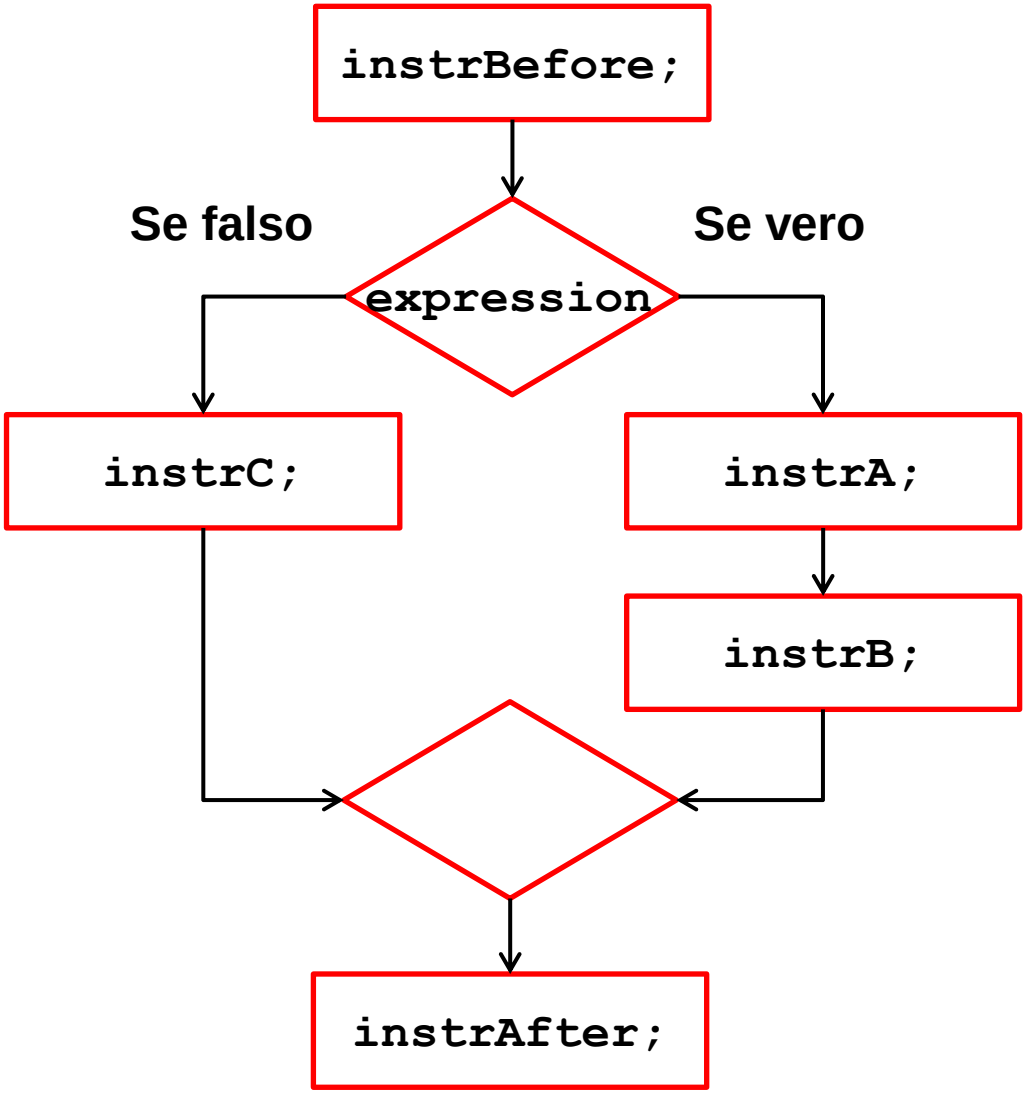
- Mi alzo quando suona la sveglia
- Mi dirigo verso la cucina
- Mangio un cornetto e bevo un caffè
- Mi lavo
- Mi vesto
- Esco
- Corro

Costrutto Condizionale

Costrutto Condizionale:



Costrutto Condizionale:



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

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

Altrimenti eseguo **`instrC`** (**ramo *else***)

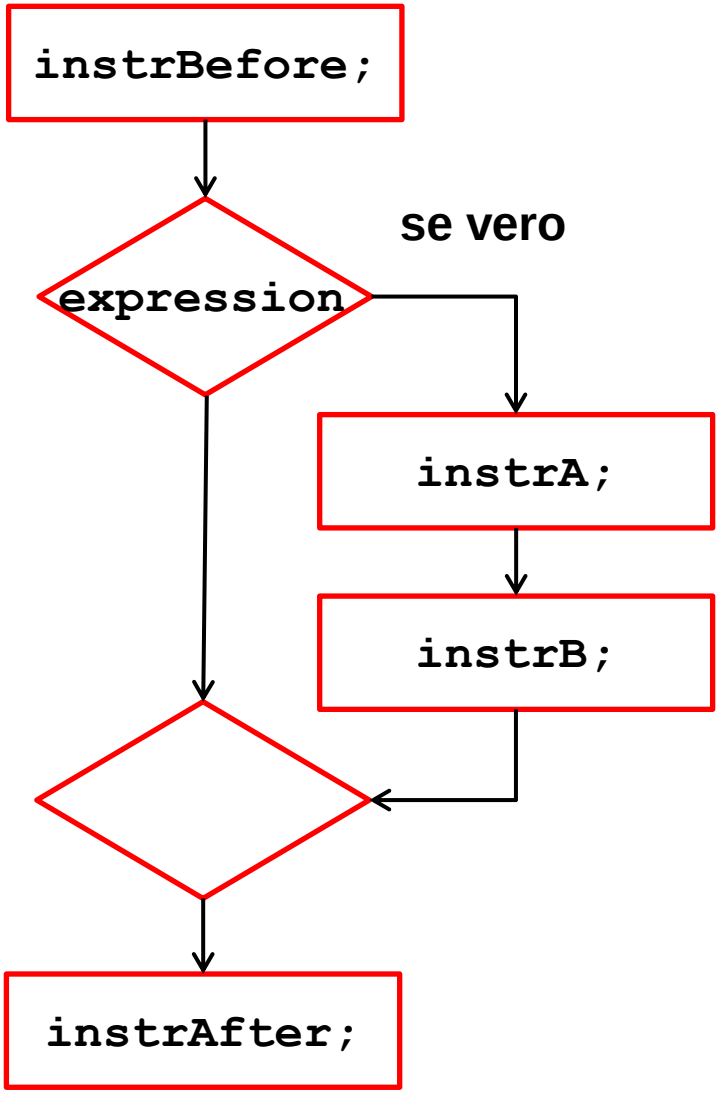
Poi eseguo **`instrAfter`**

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** devo mangiare fuori
  - prendo il pranzo

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

Costrutto Condizionale:



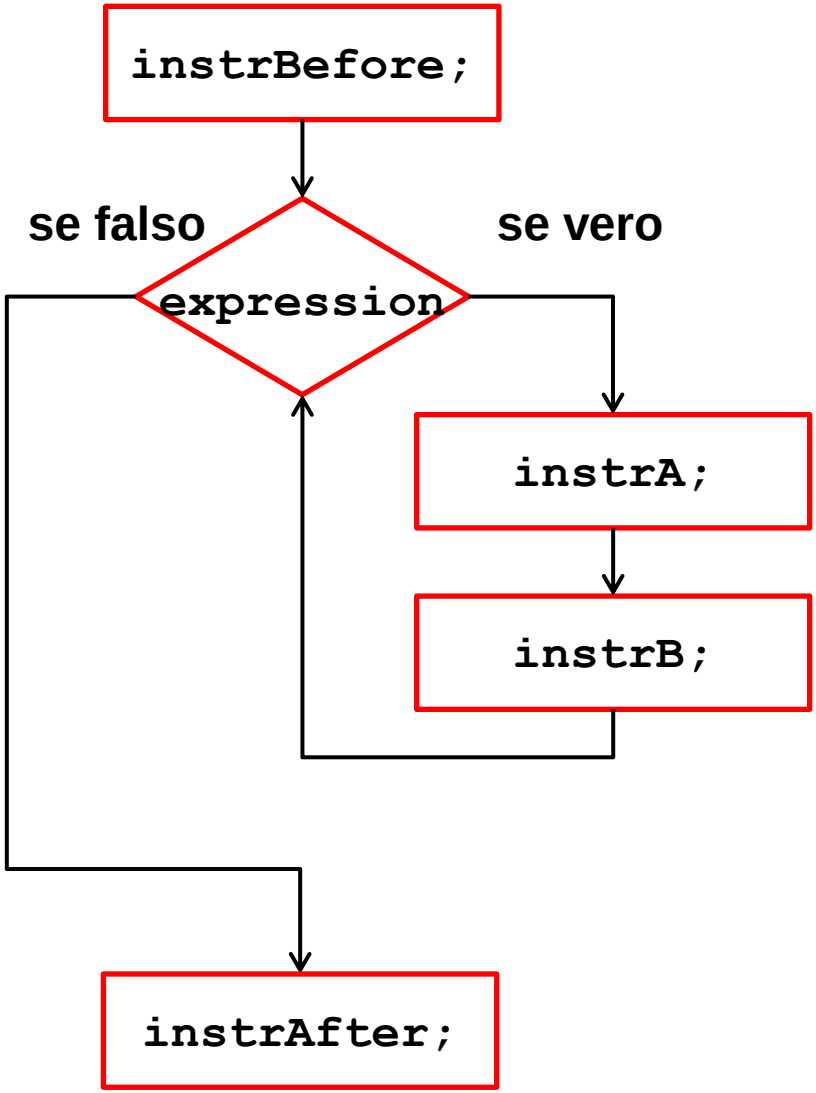
Non è obbligatorio prevedere azioni specifiche nel caso in cui **`expression`** fosse falsa

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
- Esco
- Corro

Costrutto Iterativo

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

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**: "guarda il treno in arrivo: non ferma a Bovisa?"
  - se vero, attendi il possimo treno
- Sali sul treno
- Arrivi a lezione, wow

Esempi di Algoritmi

Altri Esempi:

- Algoritmo per invertire il contenuto di due bicchieri A e B, supponendo di avere un terzo bicchiere C.

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

- Algoritmo per controllare che la maggioranza degli studenti abbia capito gli esercizi sopra.

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

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

Algoritmo per scambiare i valori di due variabili A e B

Altri Esempi:

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

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

- Algoritmo per controllare che la maggioranza degli studenti abbia capito gli esercizi sopra.

Esempio: ricerca del prodotto migliore

1. Prendi in mano il primo prodotto: assumi che sia il più economico
2. Procedi fino al prossimo prodotto
3. Confrontalo con quello che hai in mano
4. **Se** il prodotto davanti a te è più economico: 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
6. Hai in mano il prodotto più economico.

Esempio: ricerca del prodotto migliore

1. Prendi in mano il primo prodotto: assumi che sia il più economico
2. Procedi fino al prossimo prodotto
3. Confrontalo con quello che hai in mano
4. **Se** il prodotto davanti a te è più economico: 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
6. Hai in mano il prodotto più economico.

Algoritmo per trovare il minimo/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 più economico nella corsia del supermercato, passando per la corsia una sola volta.

- Algoritmo per controllare che la maggioranza degli studenti abbia capito gli esercizi sopra.

Esempio: assicurarsi che il 50% 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 array

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 (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)

La parte più difficile è spesso la prima.

- Prima dobbiamo aver chiaro «cosa far fare alla macchina», poi traduciamolo correttamente.

Proprietà essenziali dei programmi (algoritmi)

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

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

- **Diversi criteri** quindi per definire l'efficienza a seconda delle risorse
  - tempo,
  - memoria,
  - numero di letture/scritture da disco

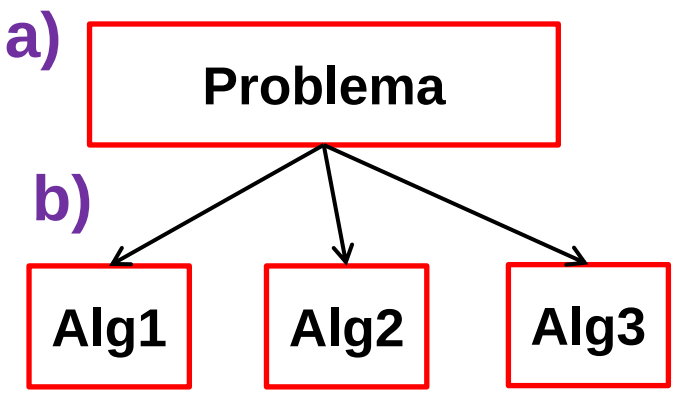
Riepilogando: Come Procedere

a) Partirete dall'analisi di un problema

Riepilogando: Come Procedere

a) Partirete dall'analisi di un problema

b) Individuerete uno o più algoritmi che risolvono il problema in modo più o meno efficiente

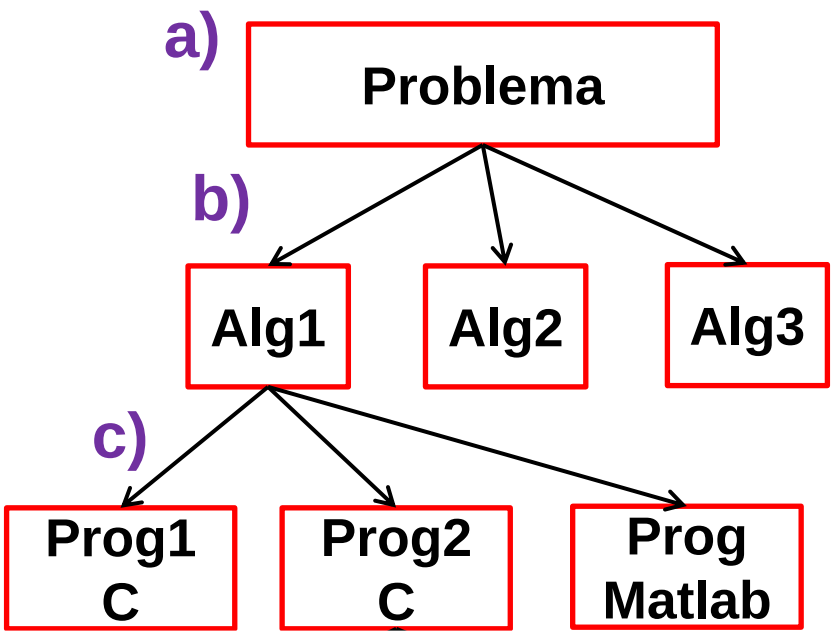


Riepilogando: Come Procedere

a) Partirete dall'analisi di un problema

b) Individuerete uno o più algoritmi che risolvono il problema in modo più o meno efficiente

c) Codificherete un algoritmo in un linguaggio di programmazione,



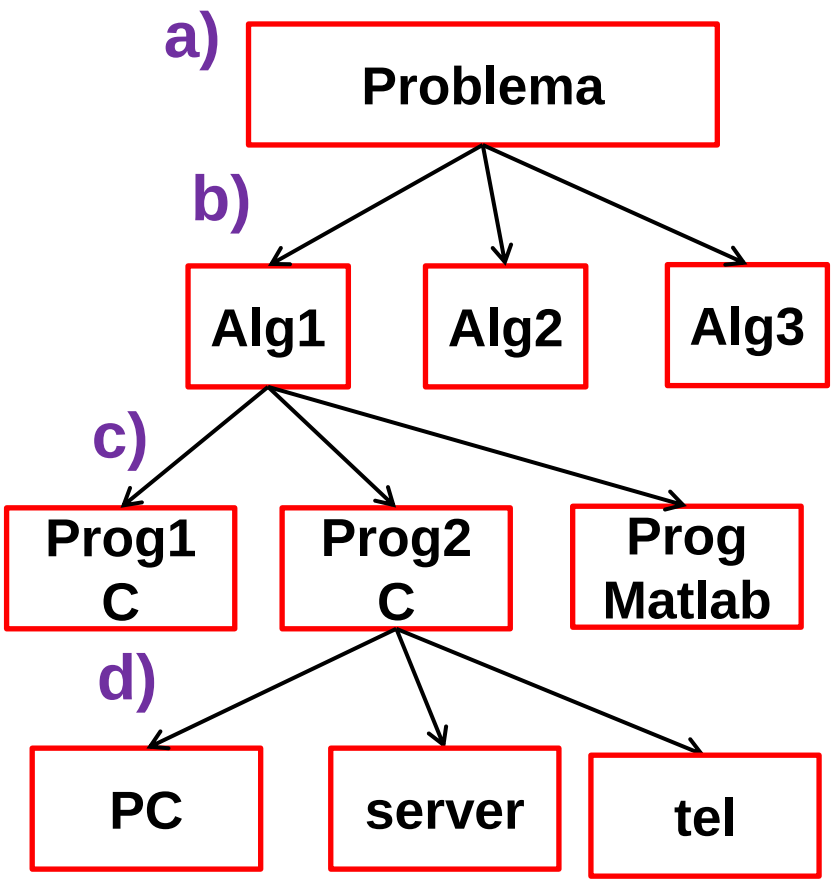
Riepilogando: Come Procedere

a) Partirete dall'analisi di un problema

b) Individuerete uno o più algoritmi che risolvono il problema in modo più o meno efficiente

c) Codificherete un algoritmo in un linguaggio di programmazione

d) Il programma potrà essere «utilizzato» su diverse macchine



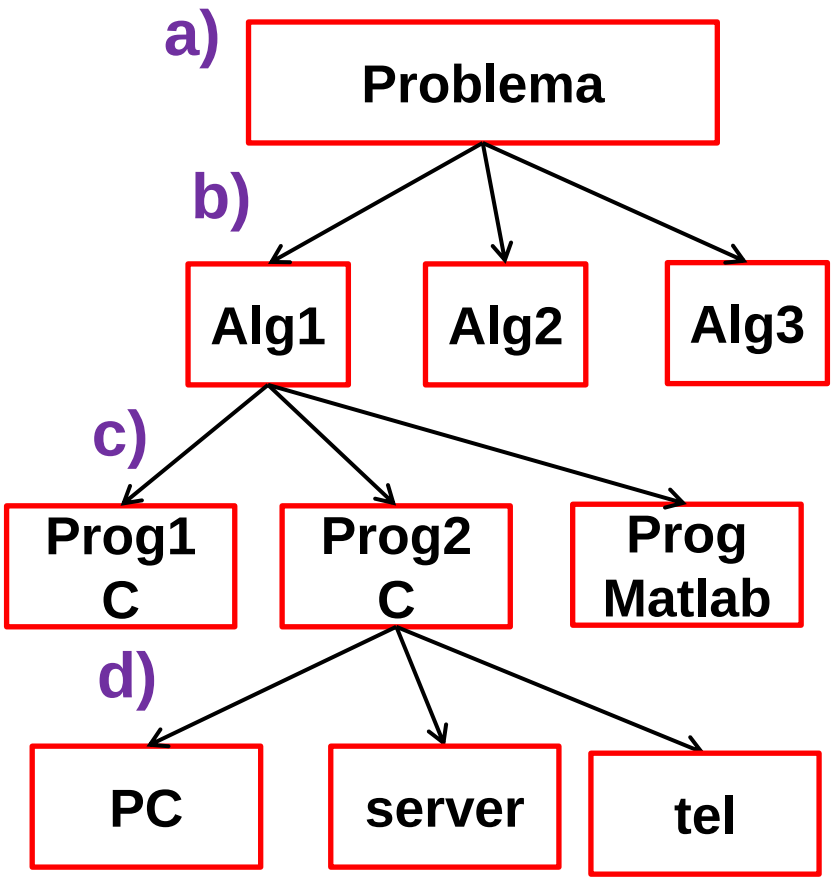
Riepilogando: Cosa Fare

a) Dovrete (capire e) definire correttamente il problema

b) **Ricavare l'algoritmo** è la parte più **difficile**. Richiede, oltre a 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) All'ultimo step pensa la macchina… vedremo poi la compilazione



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.

[da «Informatica Arte e Mestiere»]

Rappresentazione dell'informazione

I calcolatori sono in grado di operare con informazioni **binarie**. Gli unici simboli che possiamo utilizzare sono $\{0,1\}$

Rappresentazione dell'informazione

I calcolatori sono in grado di operare con informazioni **binarie**. Gli unici simboli che possiamo utilizzare sono $\{0,1\}$

Un **bit** (*binary digit*) assume valore 0/1 corrispondente ad un determinato *stato fisico* (alta o bassa tensione nella cella di memoria)

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

Rappresentazione dell'informazione

I calcolatori sono in grado di operare con informazioni **binarie**. Gli unici simboli che possiamo utilizzare sono $\{0,1\}$

Un **bit** (*binary digit*) assume valore 0/1 corrispondente ad un determinato *stato fisico* (alta o bassa tensione nella cella di memoria)

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

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

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

Aritmetica Binaria

Un po’ di operazioni in base 2….

Codifica dei Numeri in Base 10

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

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

Codifica dei Numeri in Base 10

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

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

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

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

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)

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: Esempi

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

Codifica dei numeri in base 2: Esempi

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 elementi diversi? -> 2^m

Codifica dei numeri in base 2: Esempi

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 elementi 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

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.

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

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

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

Nel corso vedremo diverse codifiche per

- I numeri naturali
- I numeri interi (positivi e negativi)

ed inoltre vedremo come eseguire la somma di interi

… e tutte le altre potenze di 2?

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 = 2^3 bit
- KiloByte (kB) = 10^3 Byte = 2^{13} bit
- MegaByte (MB) = 10^6 Byte = 2^{23} bit
- GigaByte (GB) = 10^9 Byte = 2^{33} bit
- TheraByte (TB) = 10^{12} Byte = 2^{43} bit

Rappresentazione 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 (esempi)

A ⇔ 65 ⇔ 01000001

a ⇔ 97 ⇔ 01100001

.

.

.

Z ⇔ 90 ⇔ 01011010

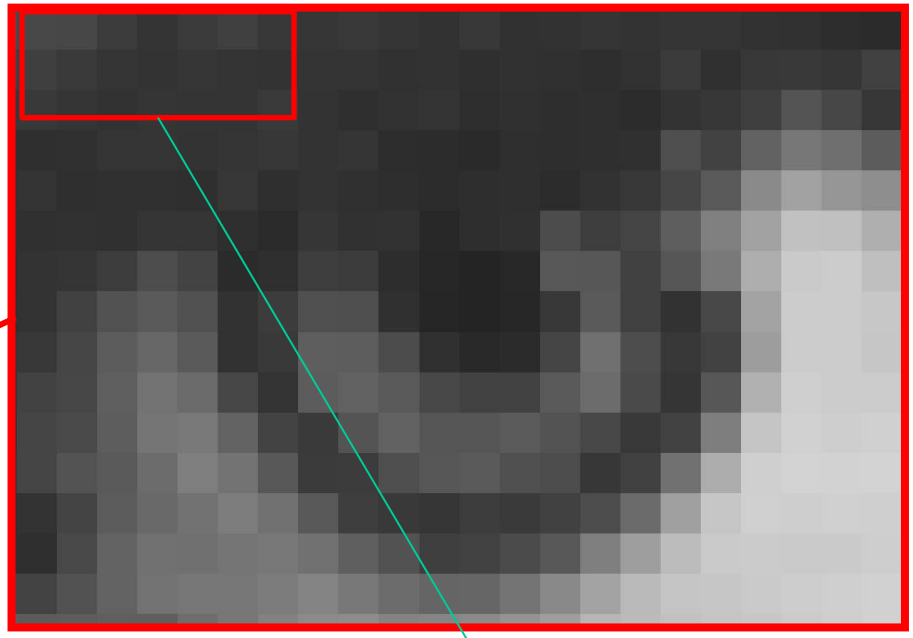
z ⇔ 122 ⇔ 01111010

La codifica ASCII (parziale)

DEC	CAR	DEC	CAR	DEC	CAR	DEC	CAR	DEC	CAR
48	0	65	A	75	K	97	a	107	k
49	1	66	B	76	L	98	b	108	l
50	2	67	C	77	M	99	c	109	m
51	3	68	D	78	N	100	d	110	n
52	4	69	E	79	O	101	e	111	o
53	5	70	F	80	P	102	f	112	p
54	6	71	G	81	Q	103	g	113	q
55	7	72	H	82	R	104	h	114	r
56	8	73	I	83	S	105	i	115	s
57	9	74	J	84	T	106	j	116	t
				85	U			117	u
				86	V			118	v
				87	W			119	w
				88	X			120	x
				89	Y			121	y
				90	Z			122	z

Codifica delle Immagini

- Le immagini nei calcolatori sono digitali, i.e. 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

- Le immagini nei calcolatori sono digitali, i.e. tabella di pixel, ciascuno caratterizzato da uno o più valori di intensità.

Canale rosso

Canale verde

Canale blu

Architettura di un Sistema Informatico

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.

[da «Informatica Arte e Mestiere»]

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, CAD… 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

John von Neumann - Wikipedia

https://it.wikipedia.org/wiki/John_von_Neumann

von neumann

Accesso non effettuato discussioni contributi Registrati Entra

Voce Discussione Leggi Modifica Modifica wikitesto Cronologia Cerca all'interno di Wikipedia

WIKIPEDIA
L'enciclopedia libera

Pagina principale
Ultime modifiche
Una voce a caso
Vetrina
Aiuto
Sportello informazioni

Comunità
Portale Comunità
Bar
Il Wikipediano
Fai una donazione
Contatti

Strumenti
Puntano qui
Modifiche correlate
Carica su Commons
Pagine speciali
Link permanente
Informazioni pagina
Elemento Wikidata
Cita questa voce

Stampa/esporta
Crea un libro
Scarica come PDF
Versione stampabile

In altri progetti
Wikimedia Commons
Wikiquote

In altre lingue
Català
Deutsch
Ελληνικά
English

Wiki Loves Monuments: fotografa un monumento, aiuta Wikipedia e vinci!

John von Neumann

Da Wikipedia, l'enciclopedia libera.

Questa voce o sezione sugli argomenti matematici e fisici non cita le fonti necessarie o quelle presenti sono insufficienti.
Puoi migliorare questa voce aggiungendo citazioni da fonti attendibili secondo le linee guida sull'uso delle fonti.

John von Neumann, nato **János Lajos Neumann**: IPA: ˈjaːnoʃ ˈlɒjoʃ ˈnɔjmɒn [in effetti: Margittai Neumann János Lajos] (Budapest, 28 dicembre 1903 – Washington, 8 febbraio 1957), è stato un matematico, fisico e informatico ungherese naturalizzato statunitense.

Generalmente considerato come uno dei più grandi matematici della storia moderna oltre ad essere una delle personalità scientifiche preminenti del XX secolo, a lui si devono contributi fondamentali in numerosi campi come la teoria degli insiemi, analisi funzionale, topologia, fisica quantistica, economia, informatica, teoria dei giochi, fluidodinamica e in molti altri settori della matematica.


John von Neumann

Indice [nascondi]

Biografia [modifica | modifica wikitesto]

Origini e formazione [modifica | modifica wikitesto]

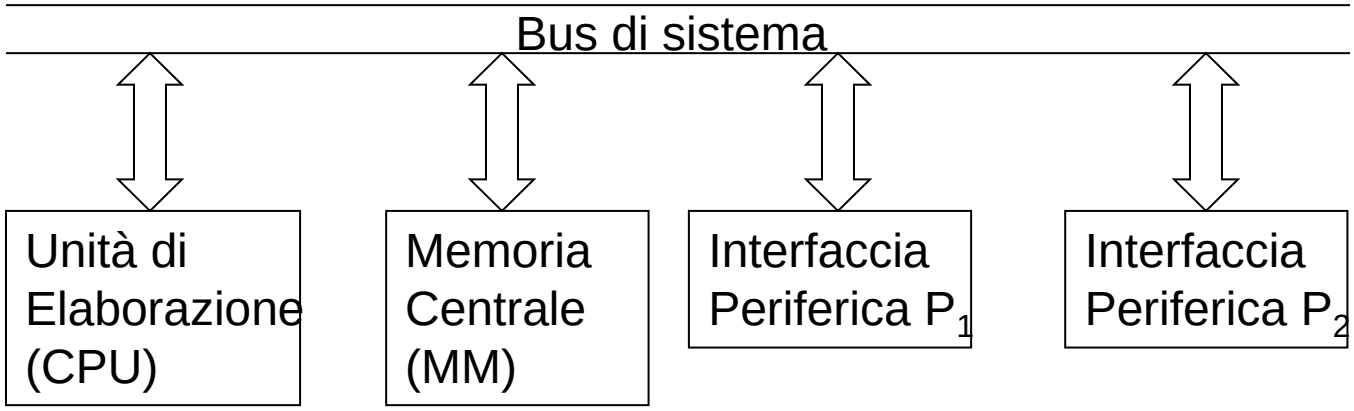
Von Neumann nacque a Budapest il 28 dicembre del 1903 da una famiglia di banchieri ebrei. Già a sei anni intratteneva gli ospiti di famiglia con la sua prodigiosa memoria, ripetendo all'istante intere pagine di elenco telefonico che gli erano state mostrate solo per pochi istanti e eseguendo rapidamente a mente divisioni con due numeri da otto cifre. Non contento si divertiva conversando in greco antico, arrivando a padroneggiare, intorno ai dieci anni, sei lingue

La Macchina di Von Neumann

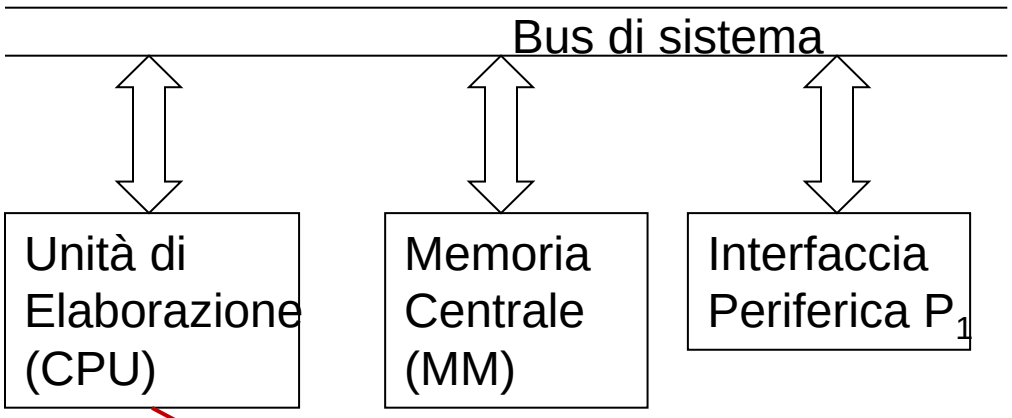
- Modello composto da **quattro elementi funzionali**
  - **Unità di elaborazione (CPU):** interpretazione ed esecuzione dei programmi, coordinamento macchina
  - **Memoria centrale:** contiene dati ed istruzioni
  - **Interfacce delle periferiche:** scambio informazioni con mondo esterno (e.g, stampante,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

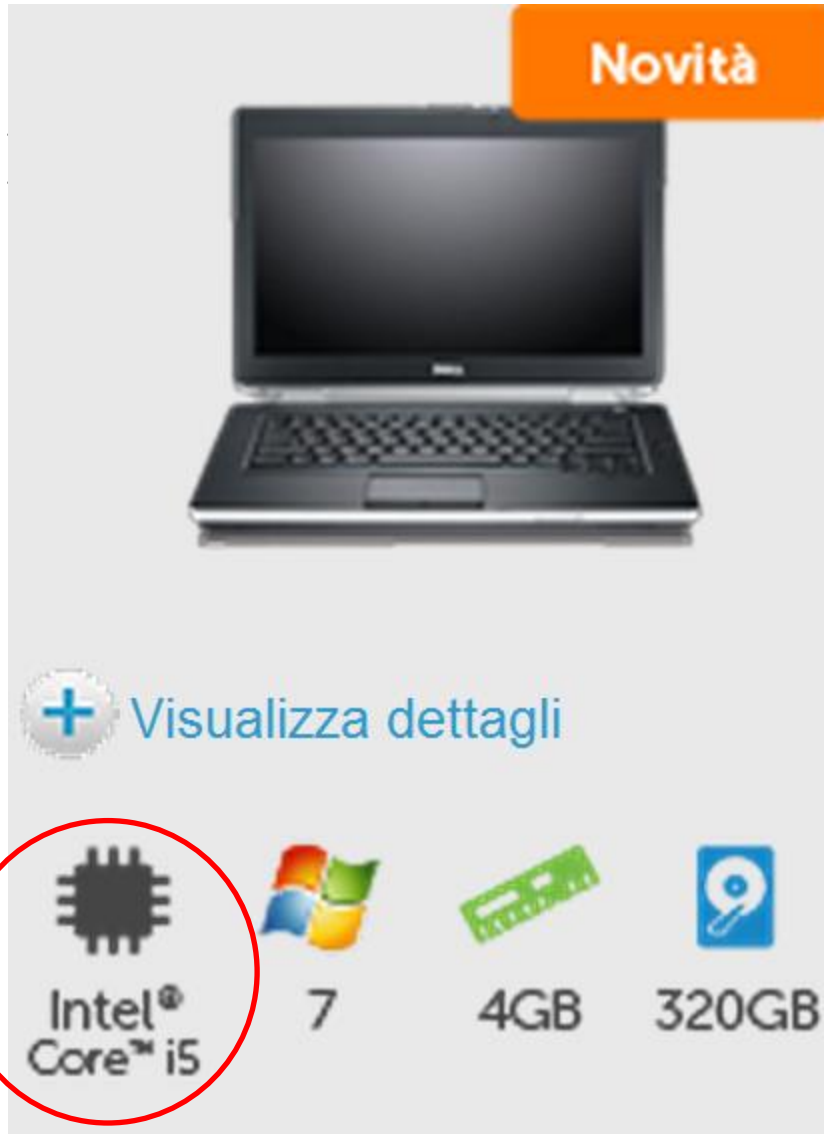
La Macchina di Von Neumann

- 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

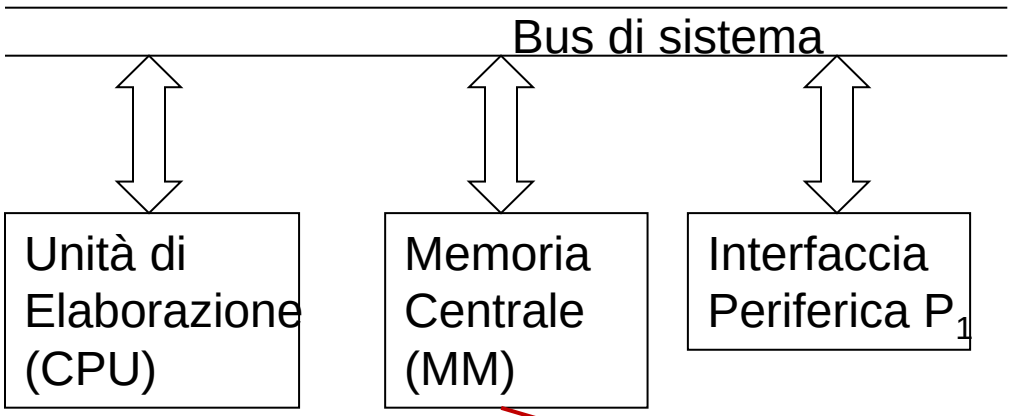


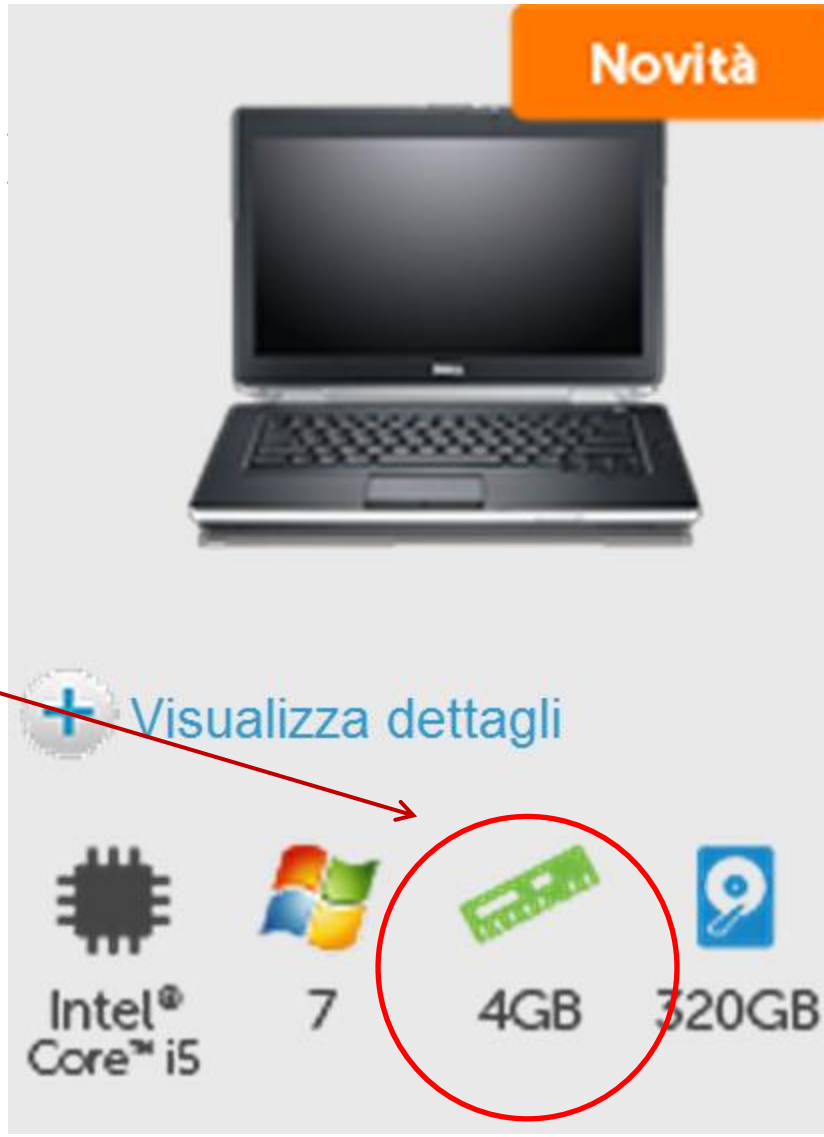
La Macchina di Von Neumann



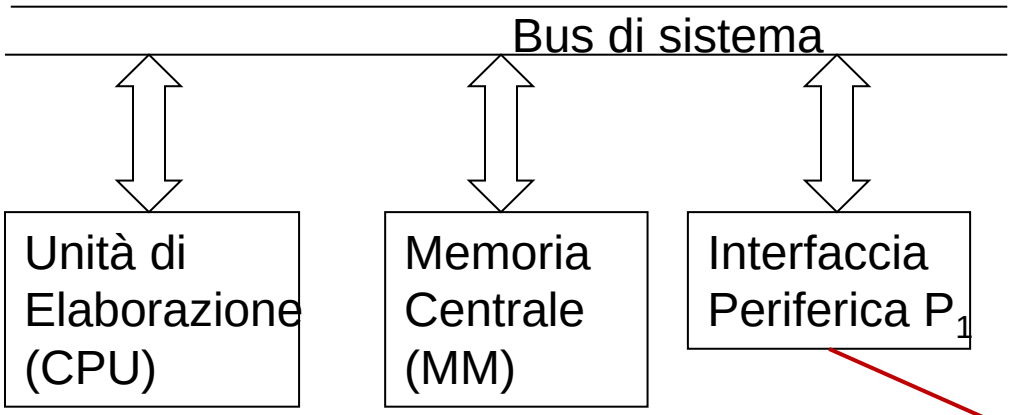


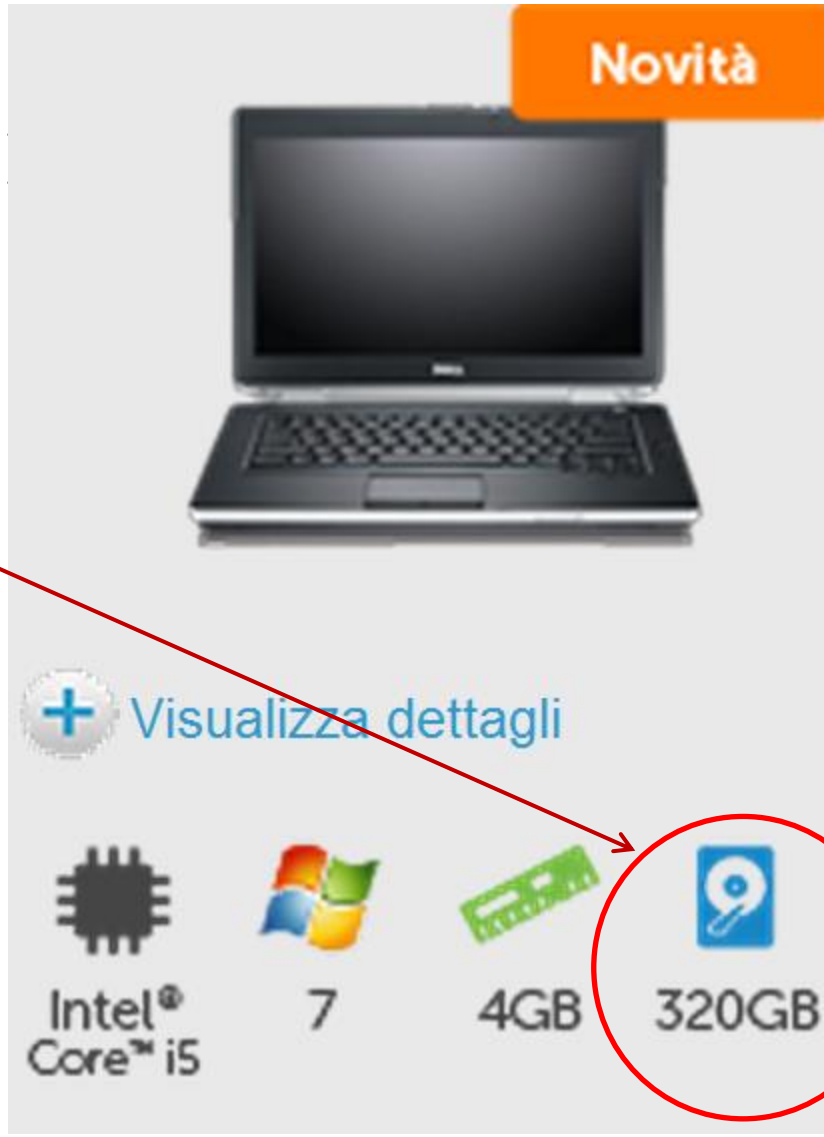
La Macchina di Von Neumann



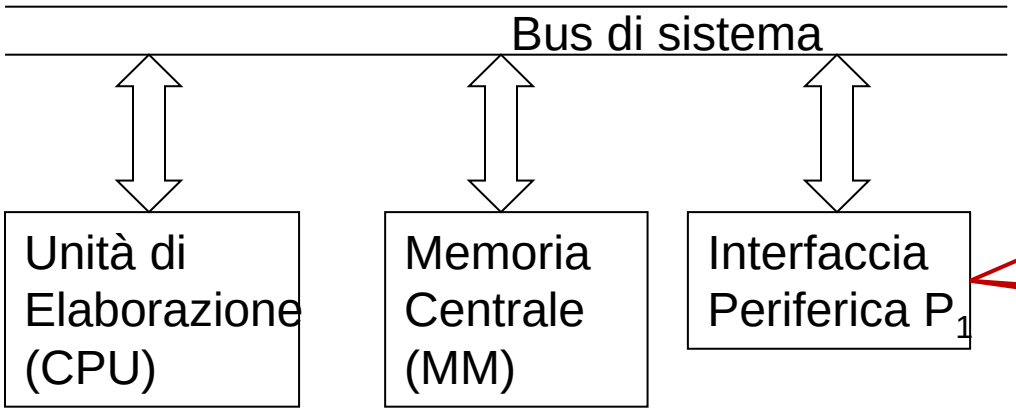


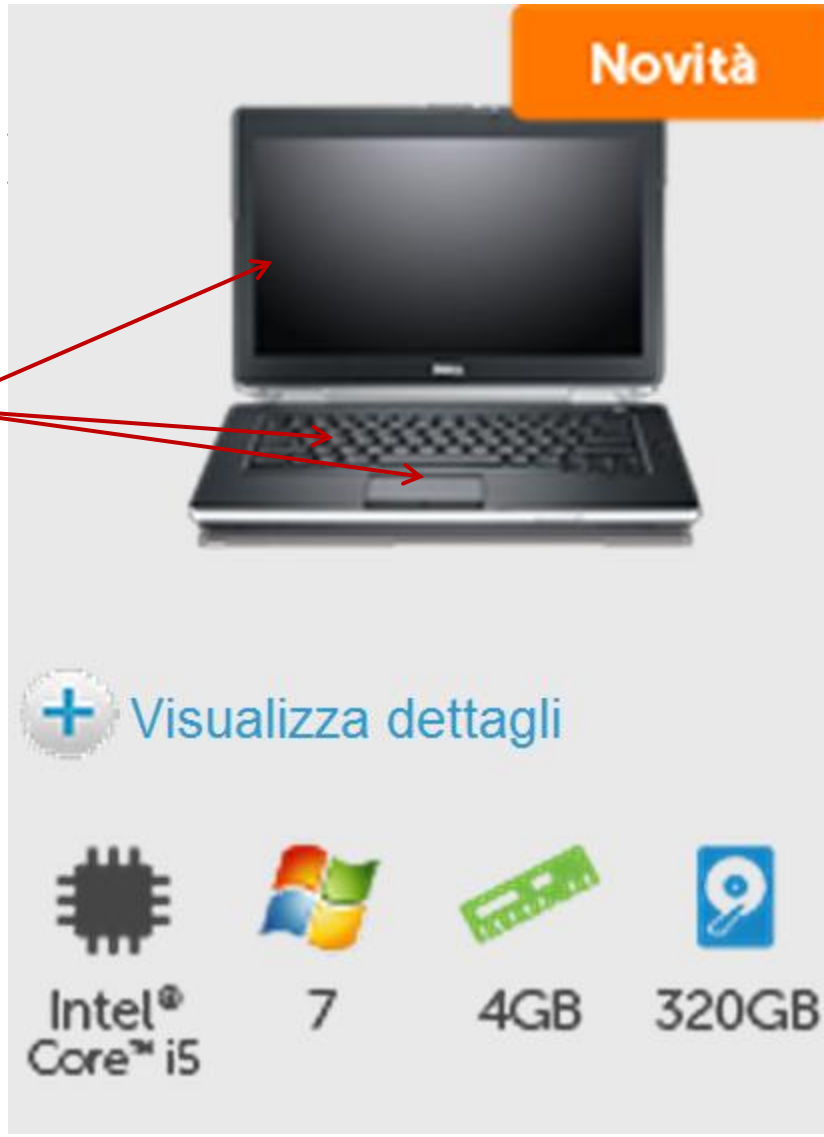
La Macchina di Von Neumann





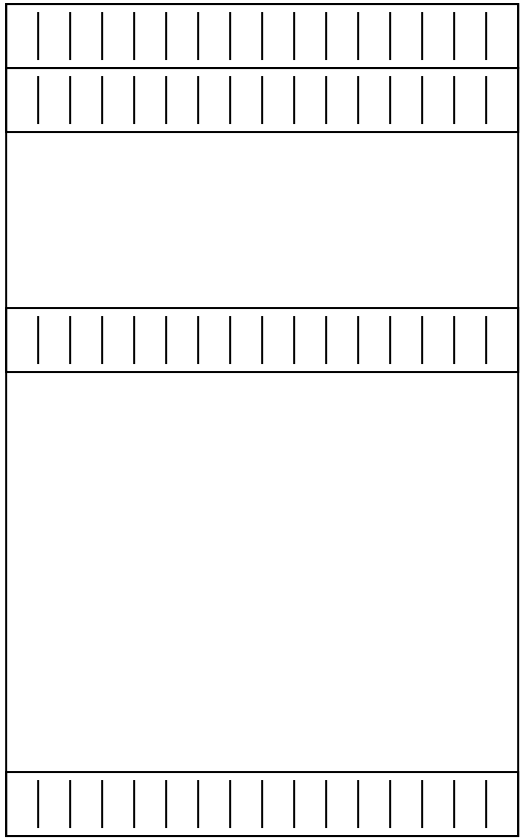
La Macchina di Von Neumann





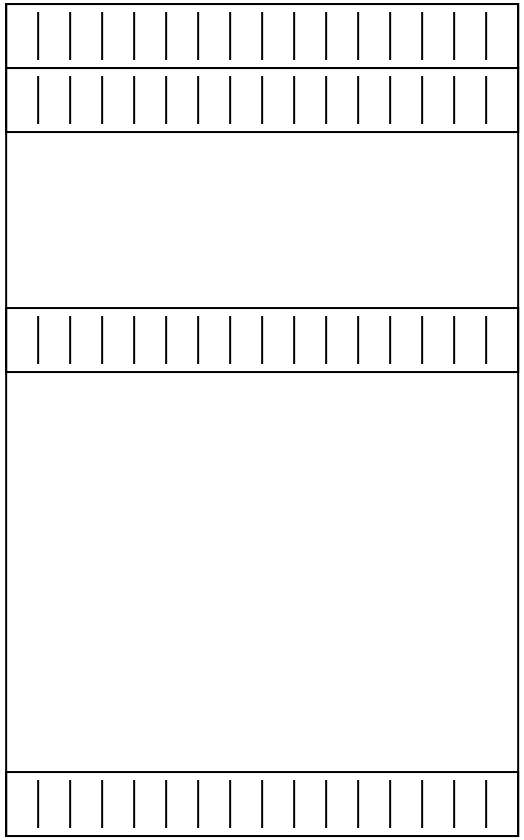
La Memoria Centrale (MM)

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



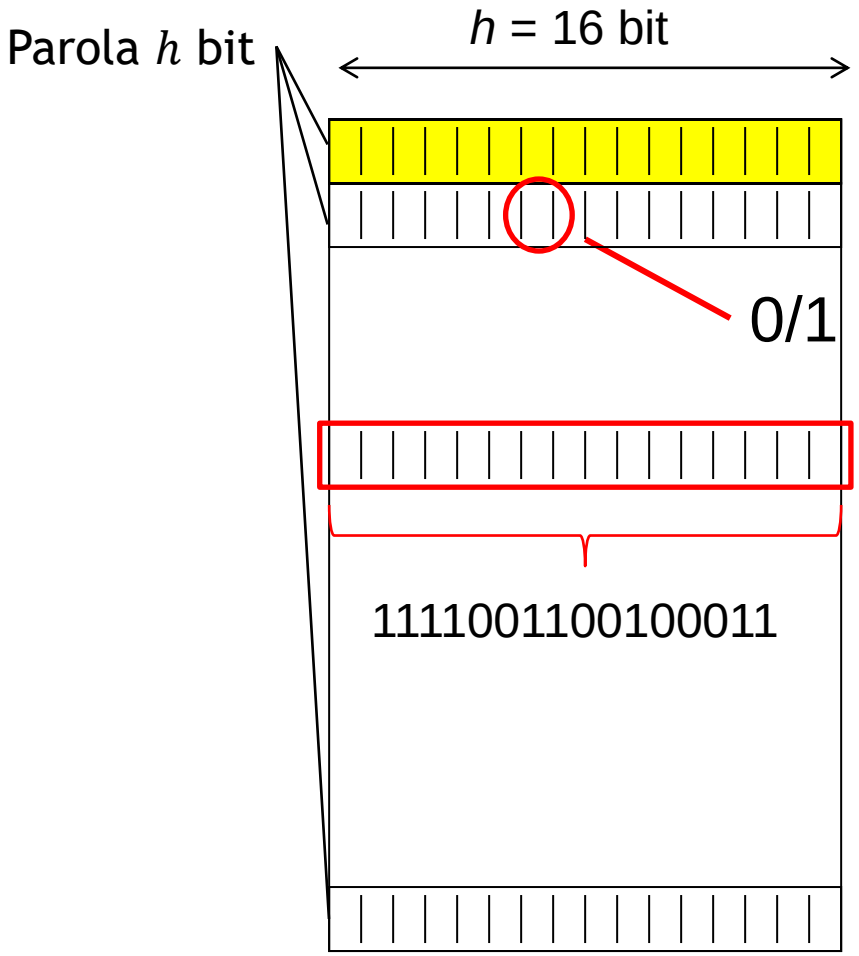
La Memoria Centrale (MM)

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



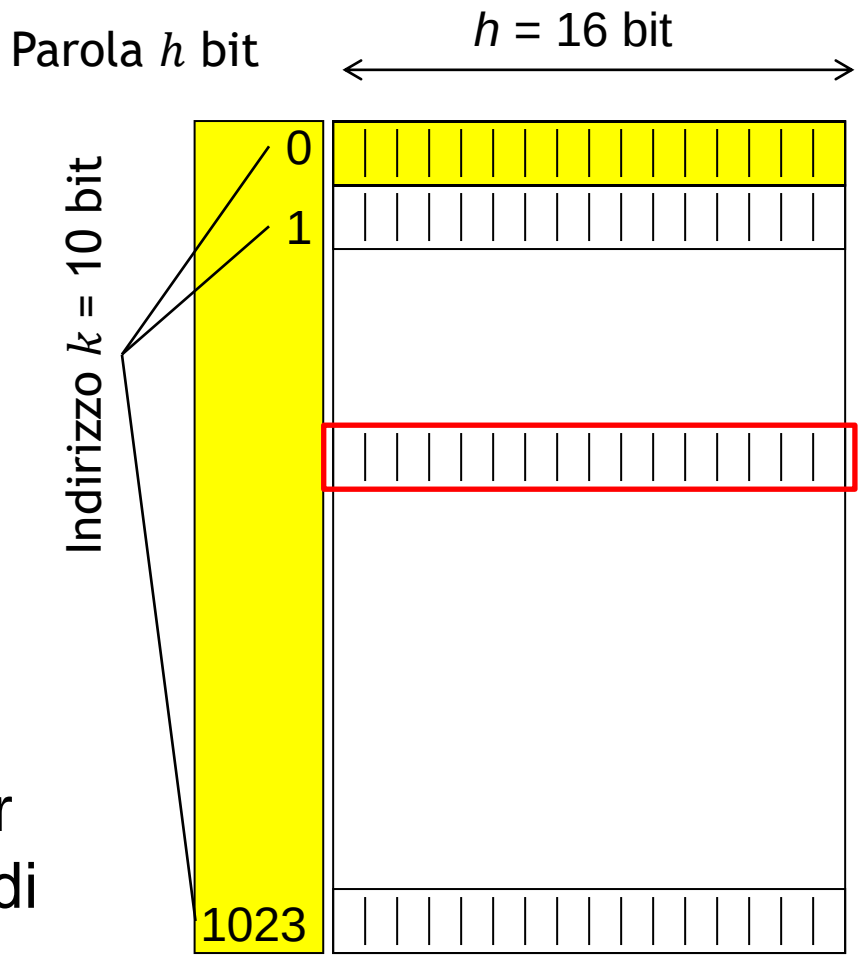
La Memoria Centrale (MM)

- 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



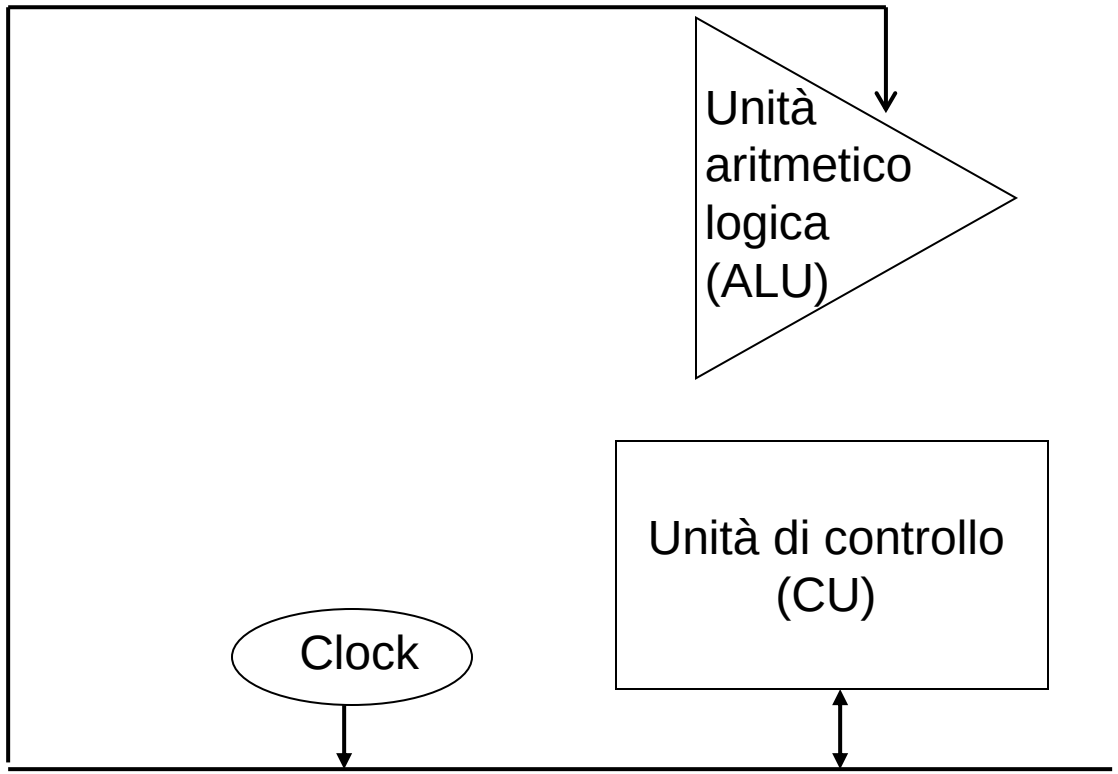
La Memoria Centrale (MM)

- 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'HD** è memoria permanente ma **non è memoria centrale** ed in riferimento alla macchina di Von Neumann è una periferica.

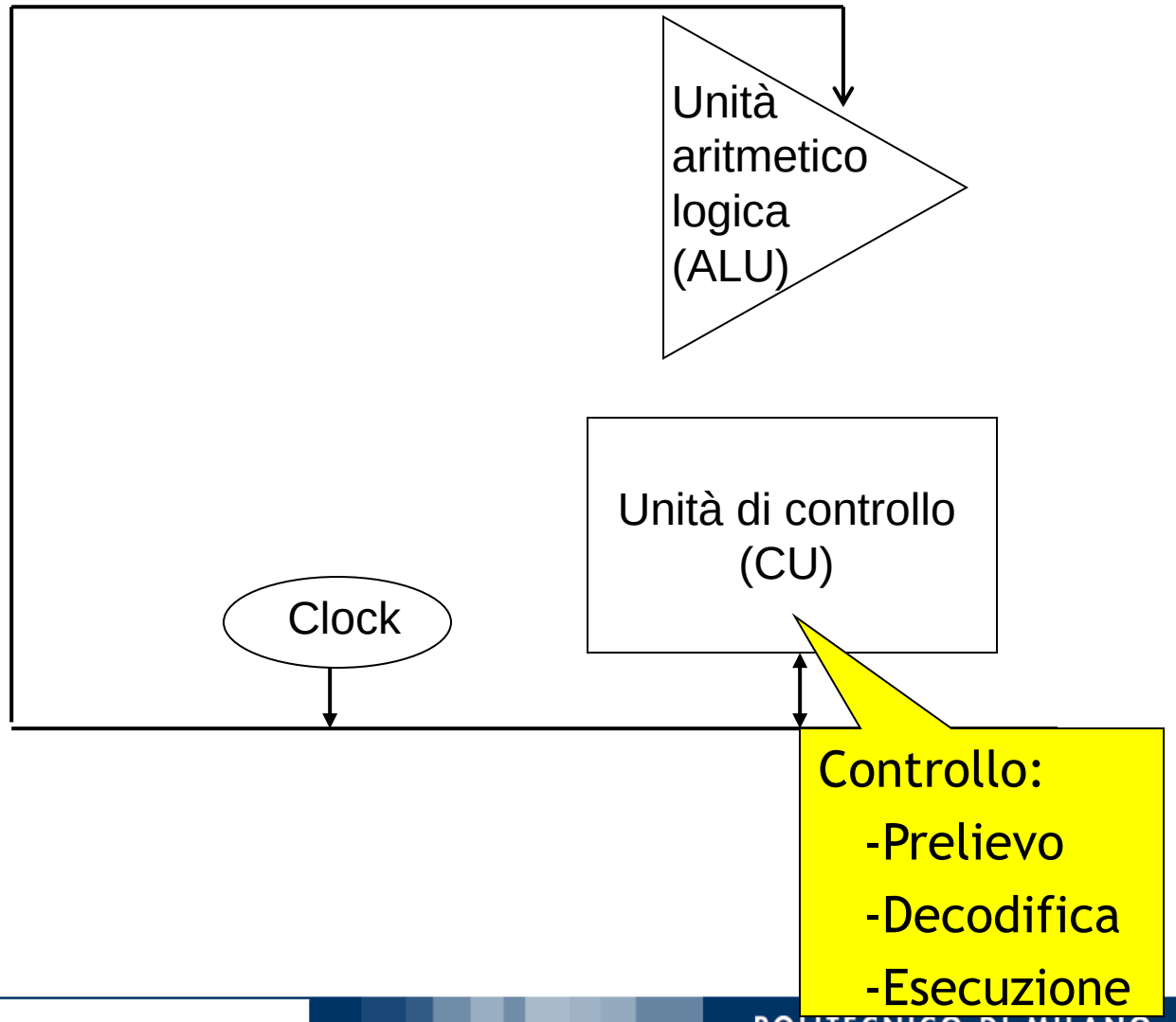
L’Unità di Elaborazione (CPU)

- 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**

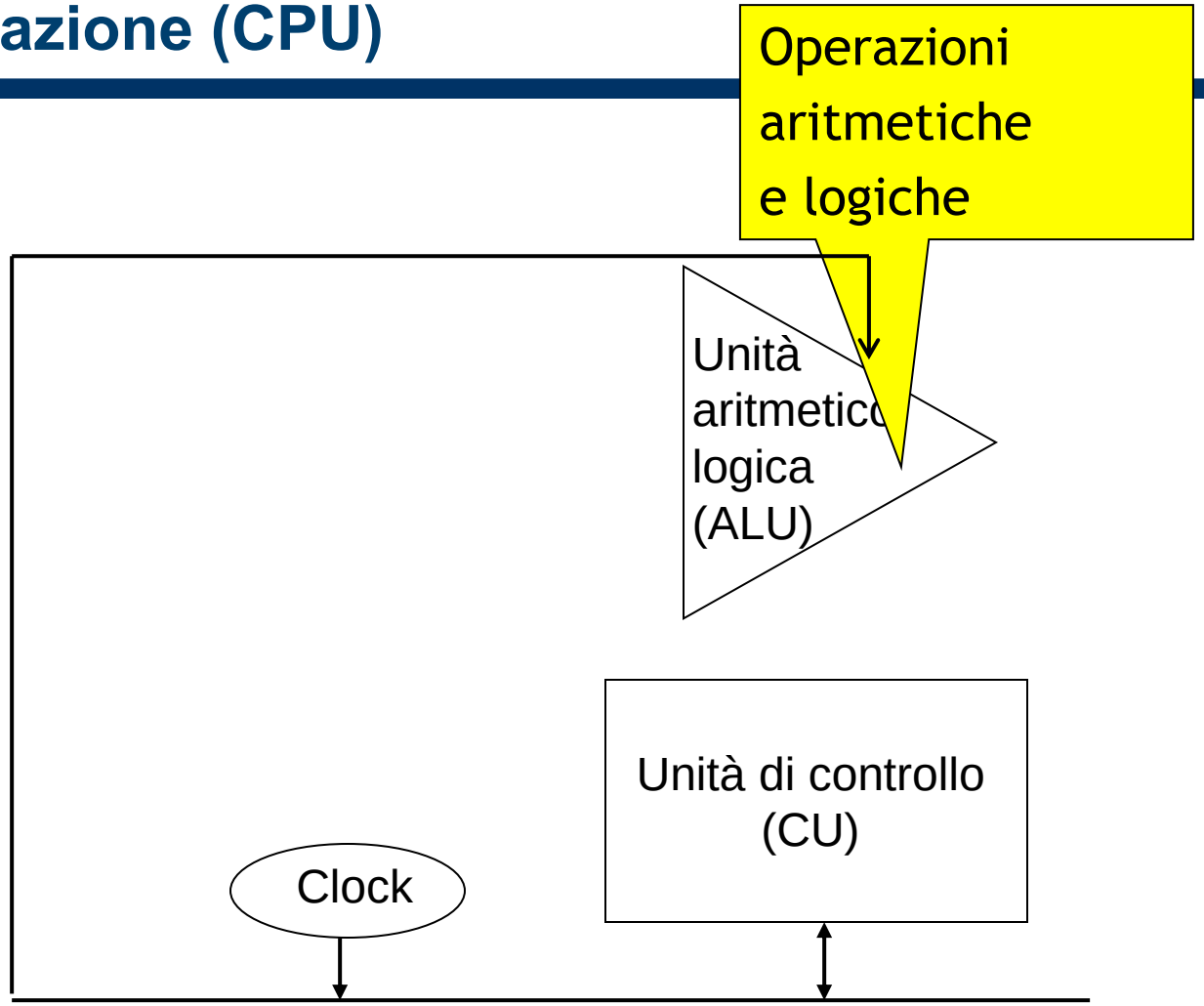
L’Unità di Elaborazione (CPU)



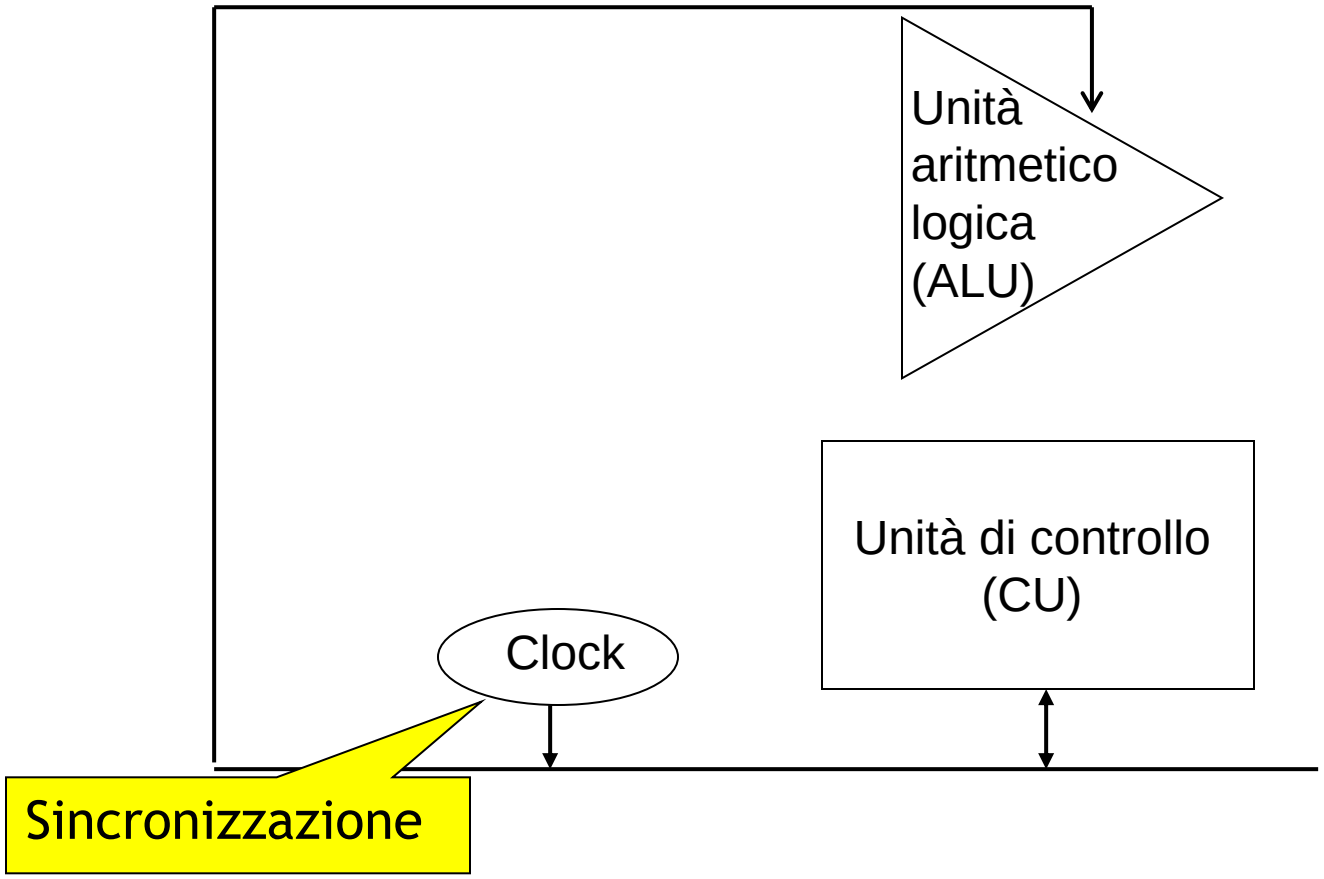
L'Unità di Elaborazione (CPU)



L'Unità di Elaborazione (CPU)



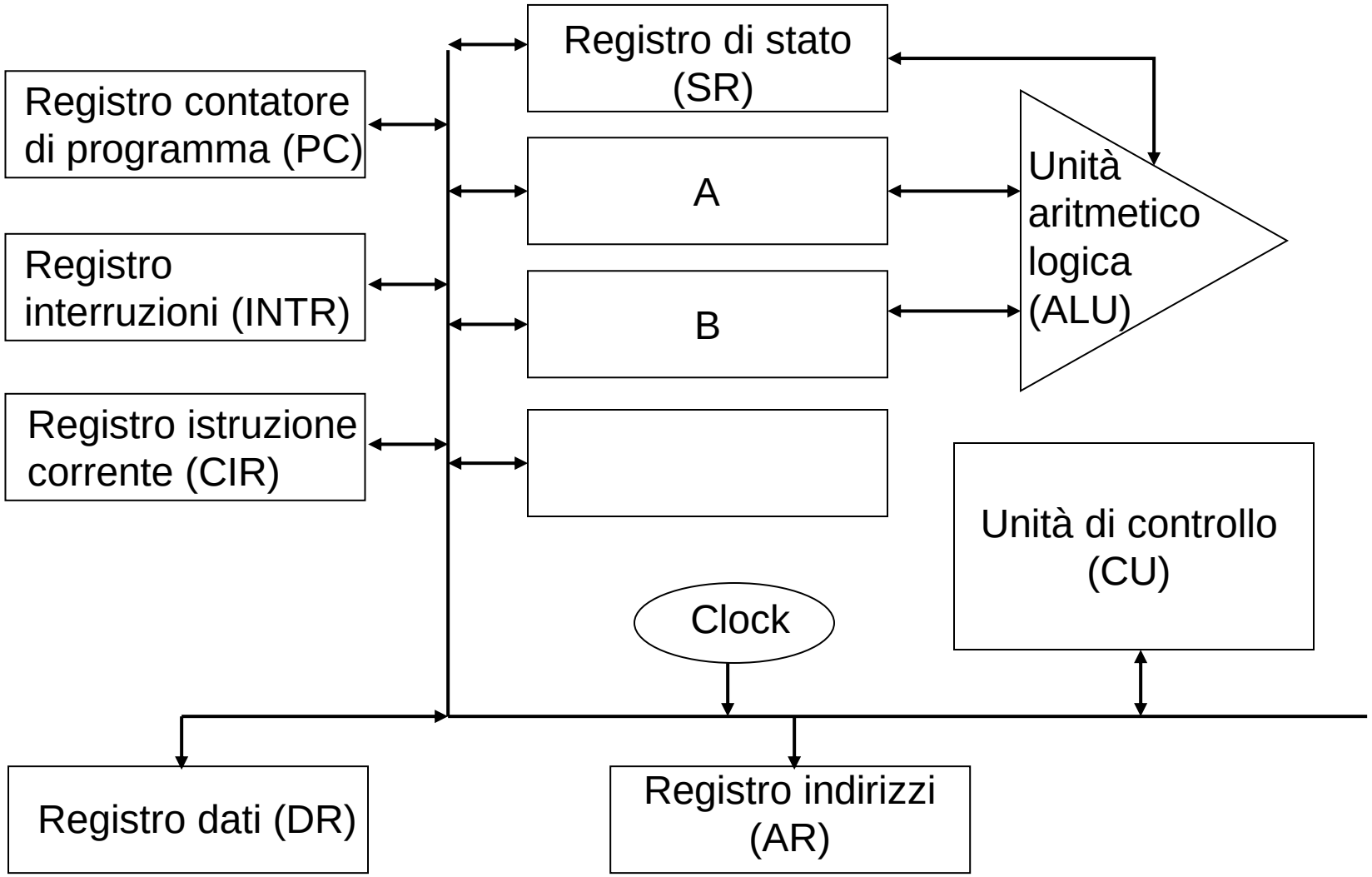
L'Unità di Elaborazione (CPU)



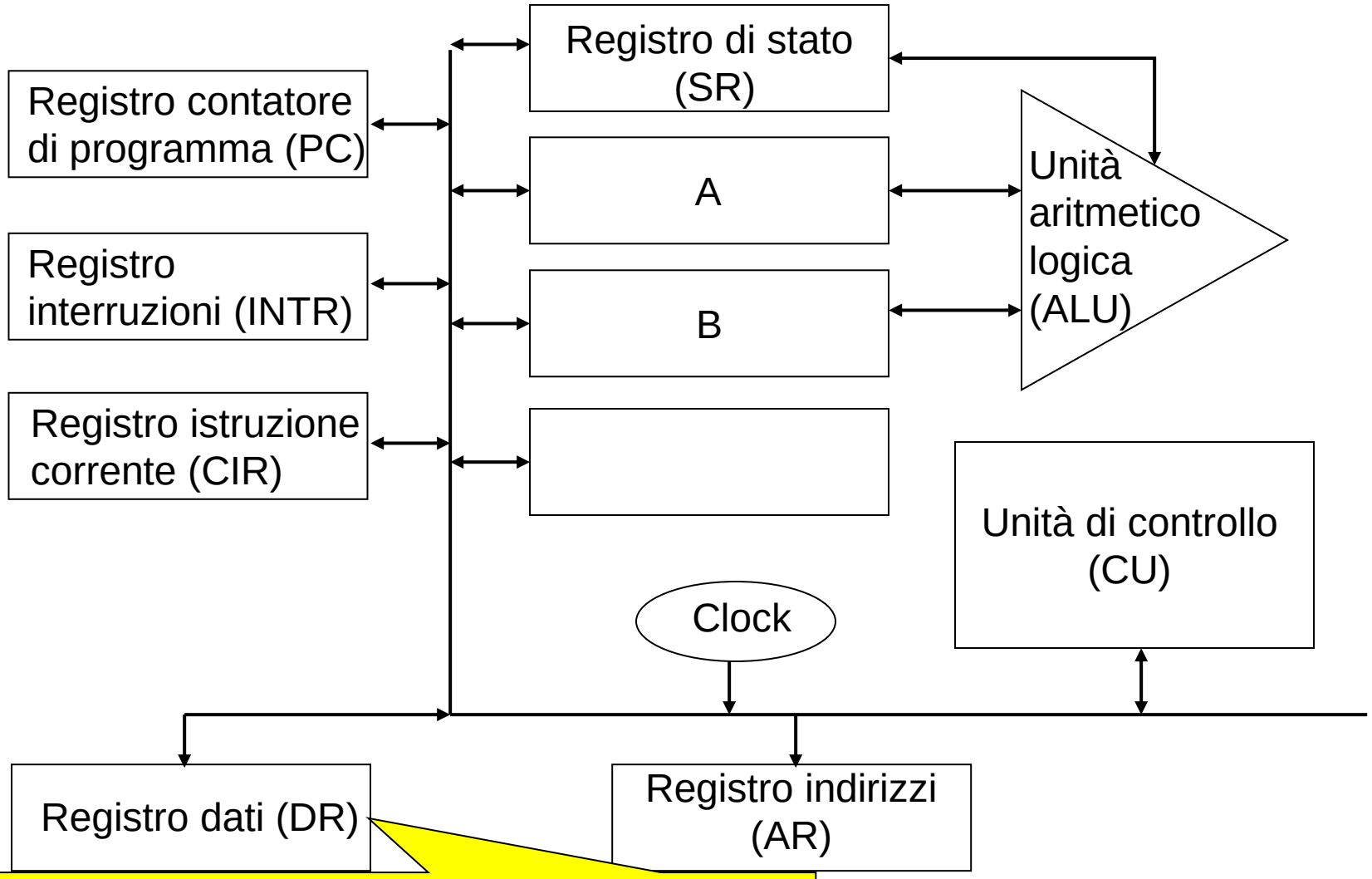
L'Unità di Elaborazione (CPU)

- 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)

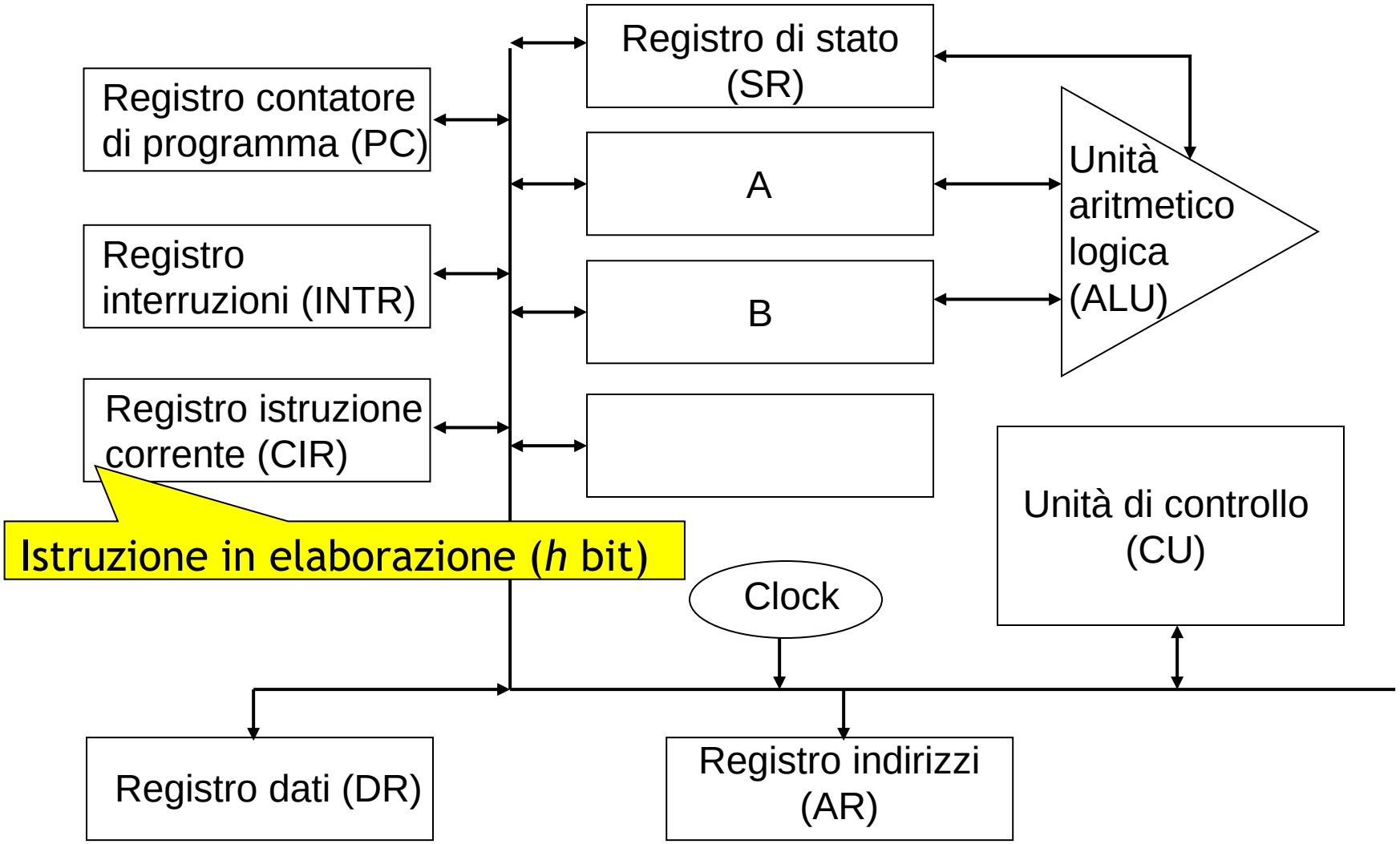


L'Unità di Elaborazione (CPU)

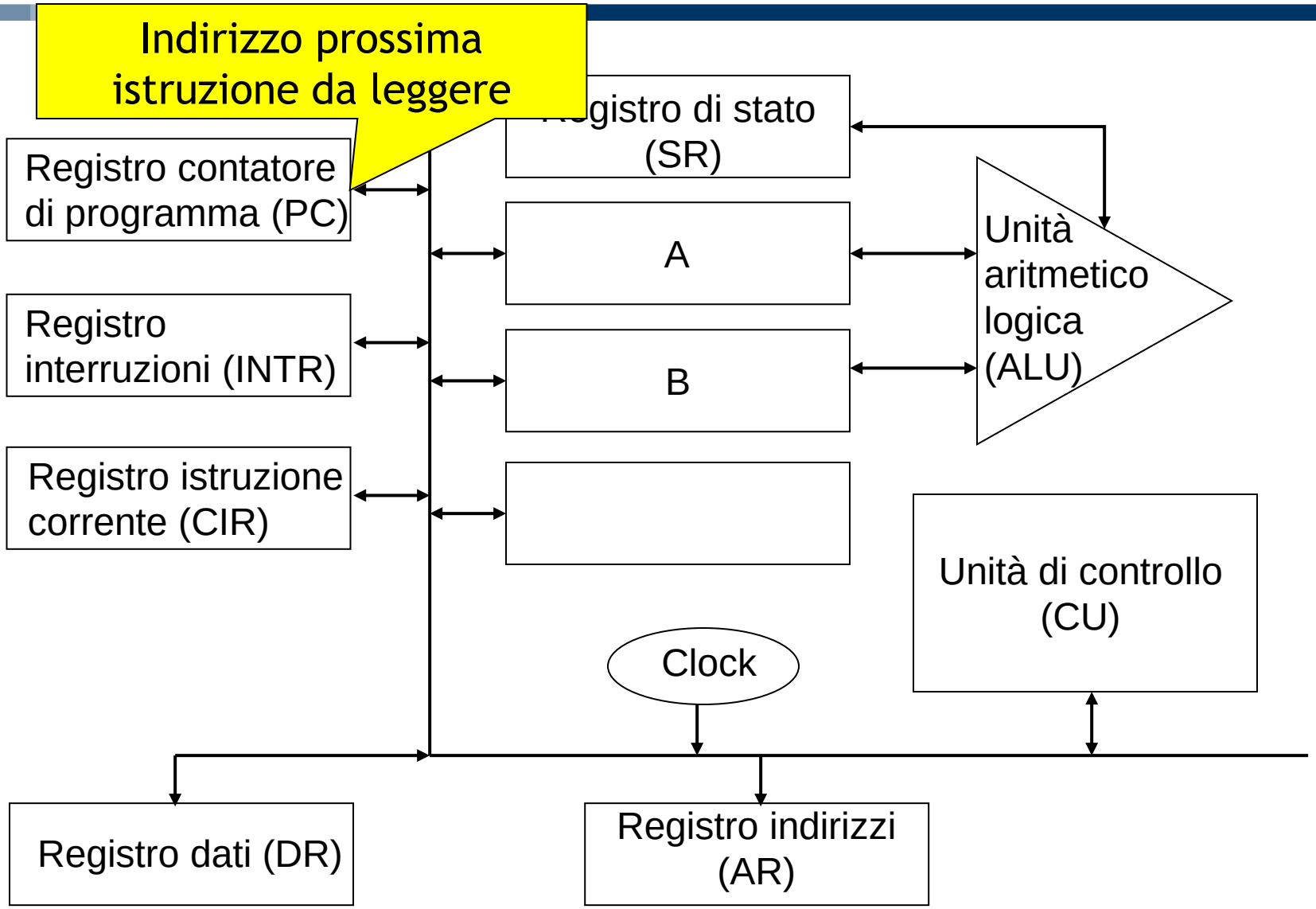


Parola letta/da scrivere in MM (*h* bit)

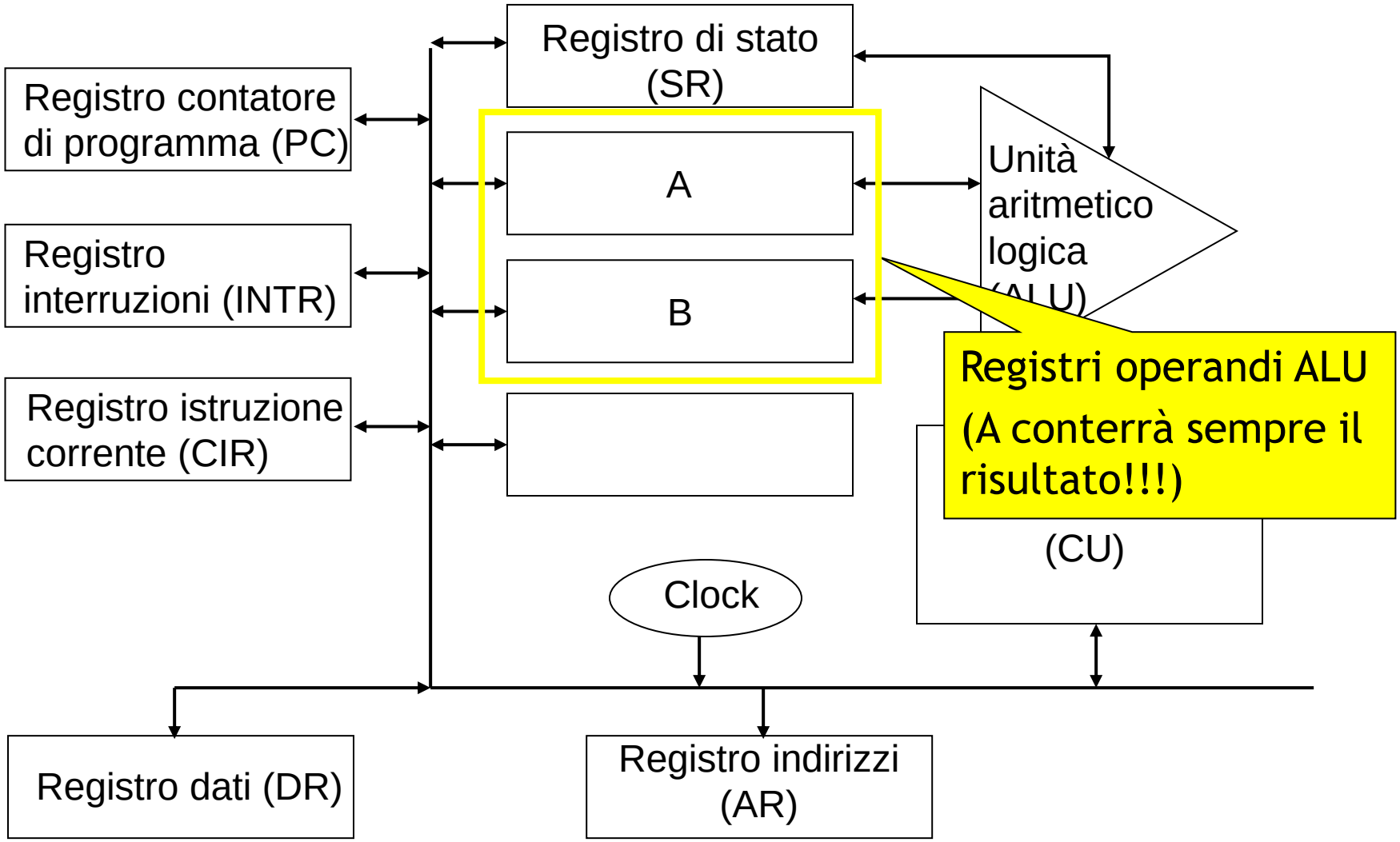
L'Unità di Elaborazione (CPU)



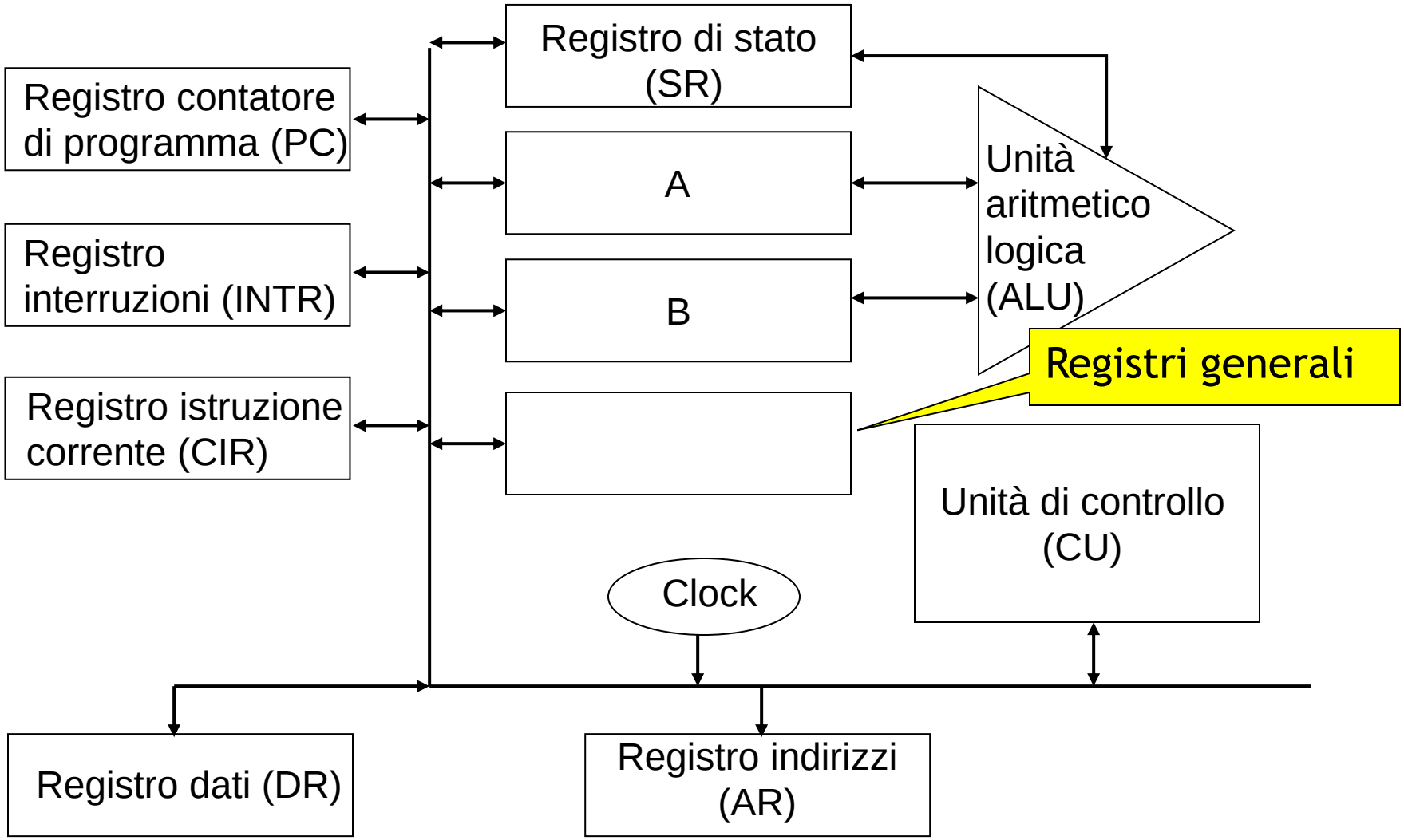
L'Unità di Elaborazione (CPU)



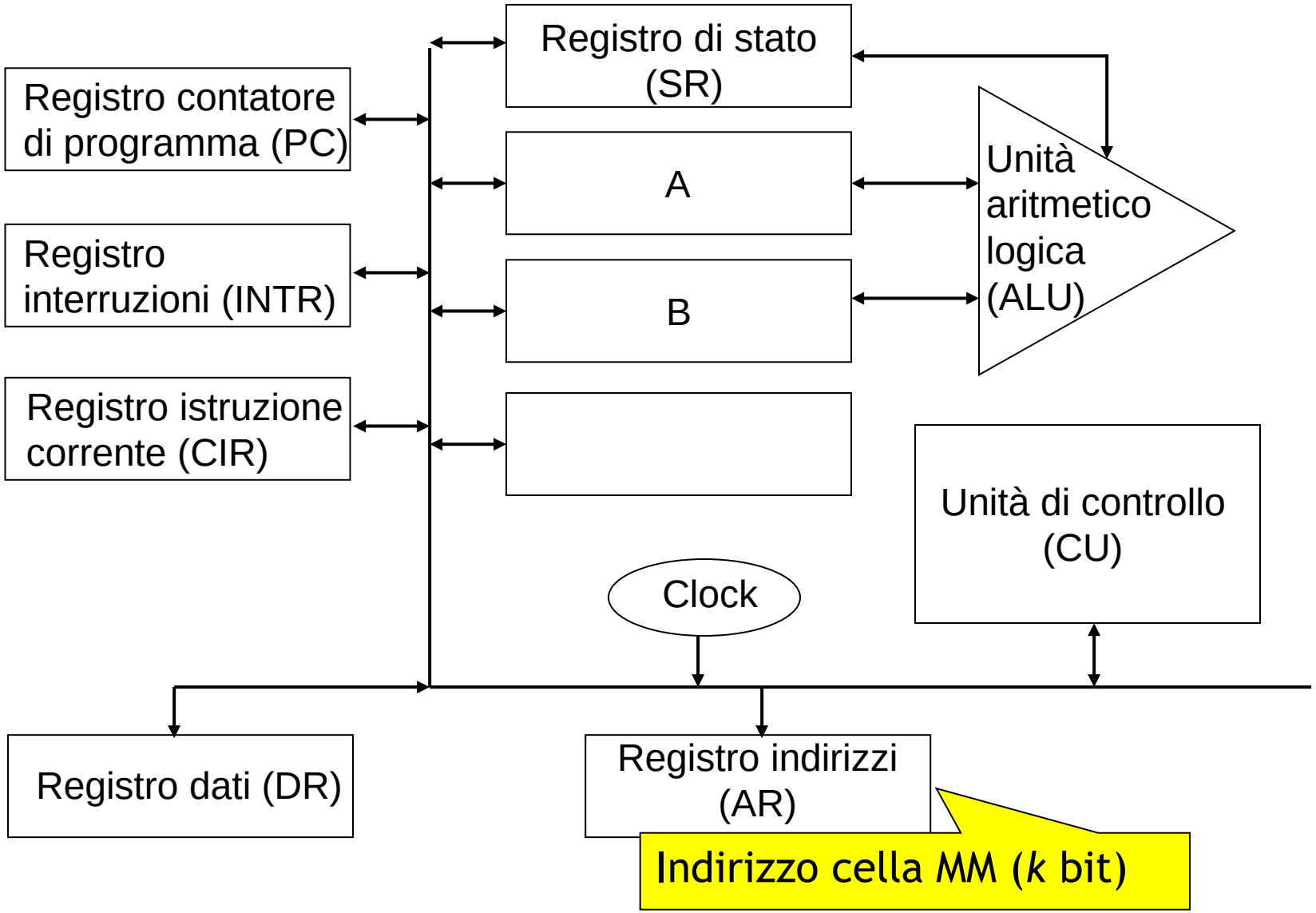
L'Unità di Elaborazione (CPU)



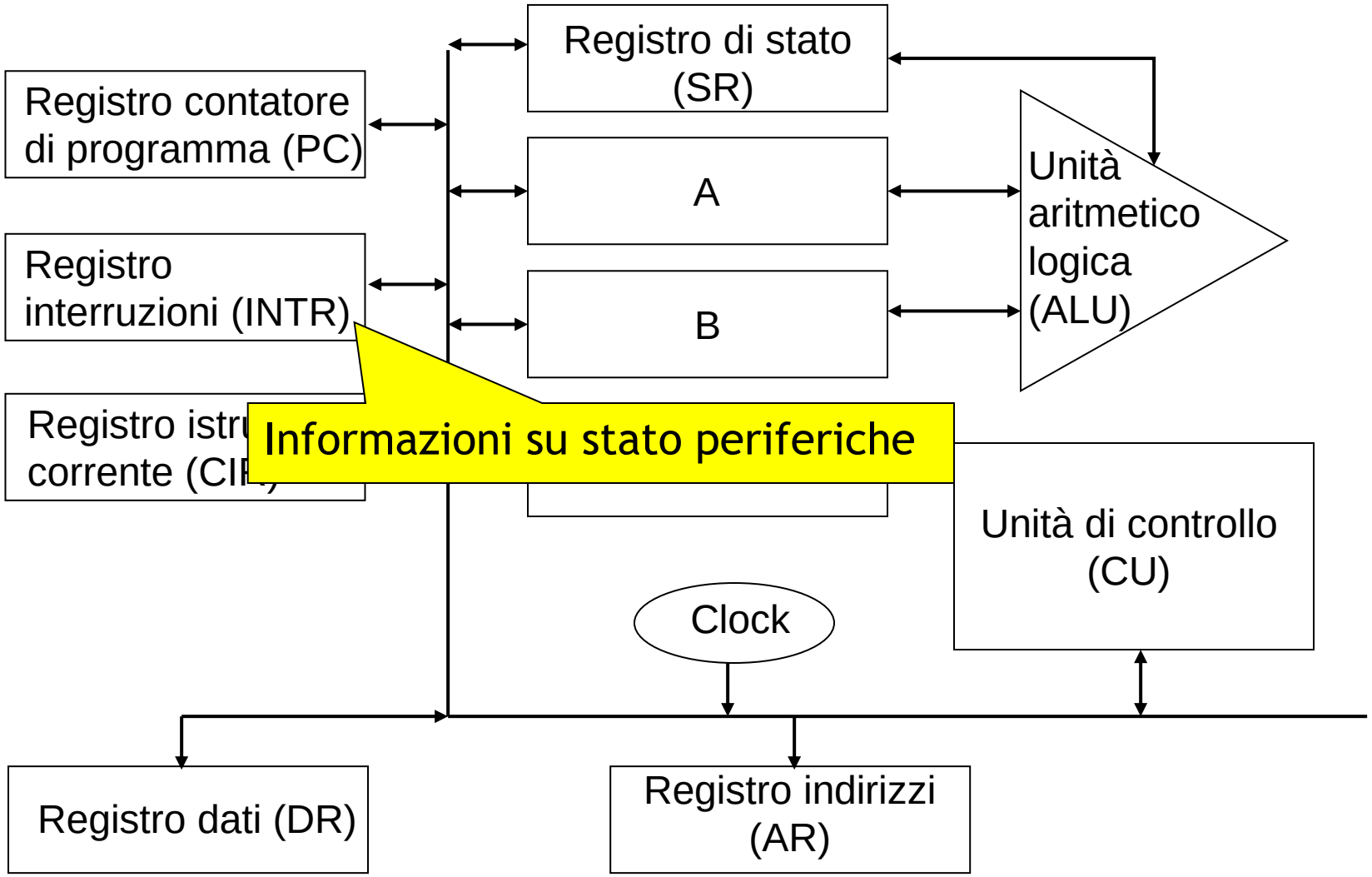
L'Unità di Elaborazione (CPU)



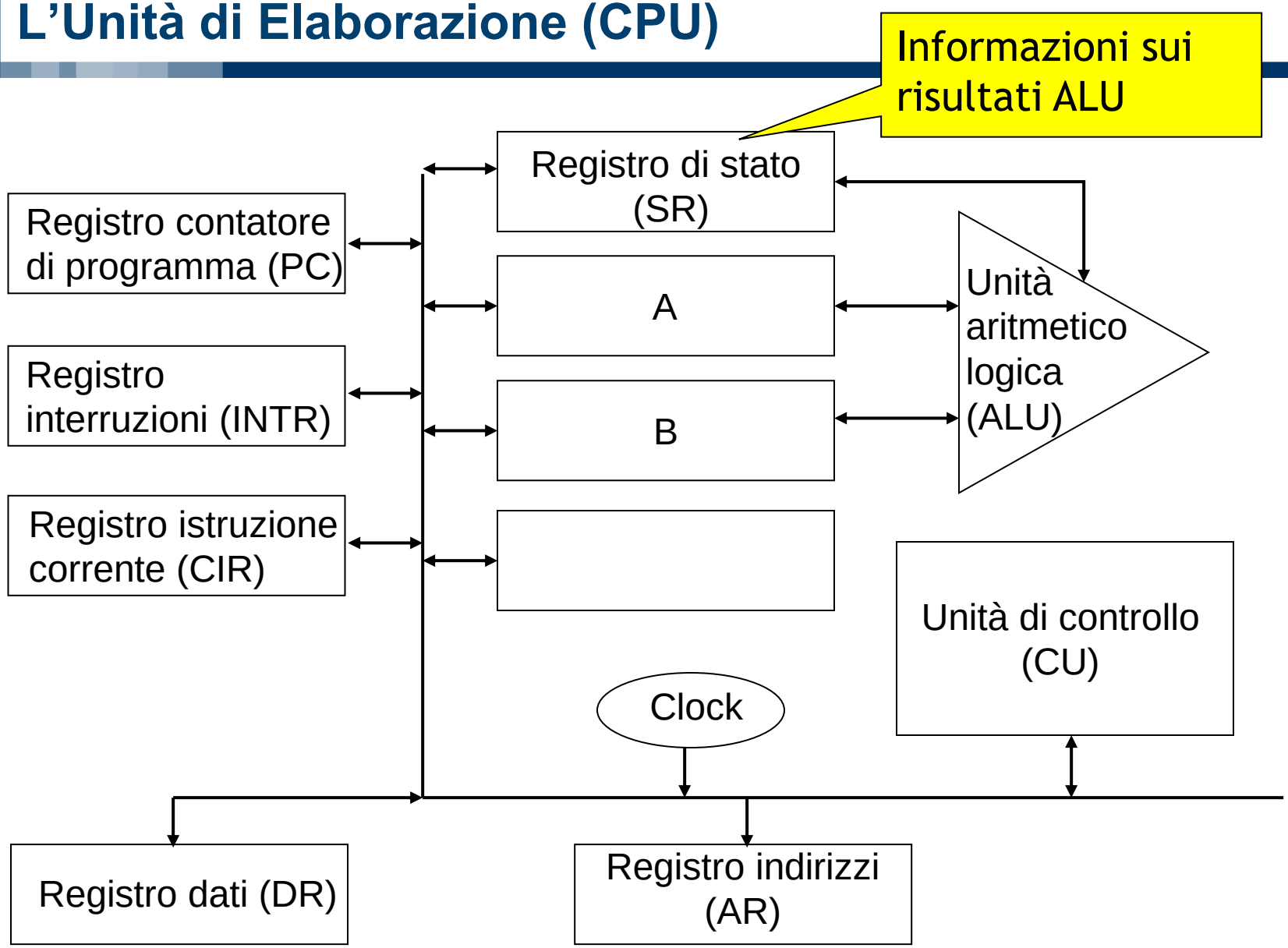
L'Unità di Elaborazione (CPU)



L'Unità di Elaborazione (CPU)



L'Unità di Elaborazione (CPU)

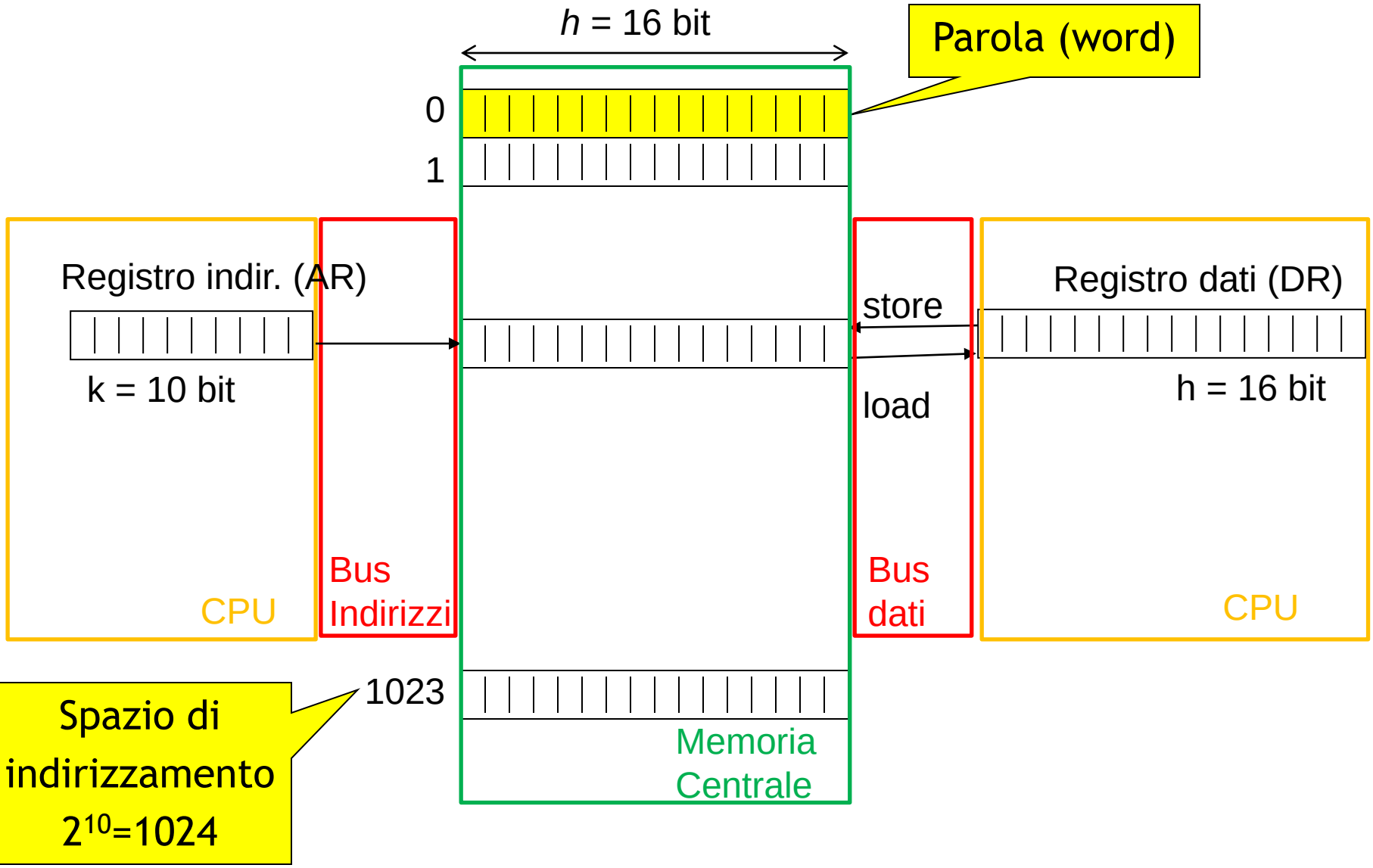


Bus di Sistema

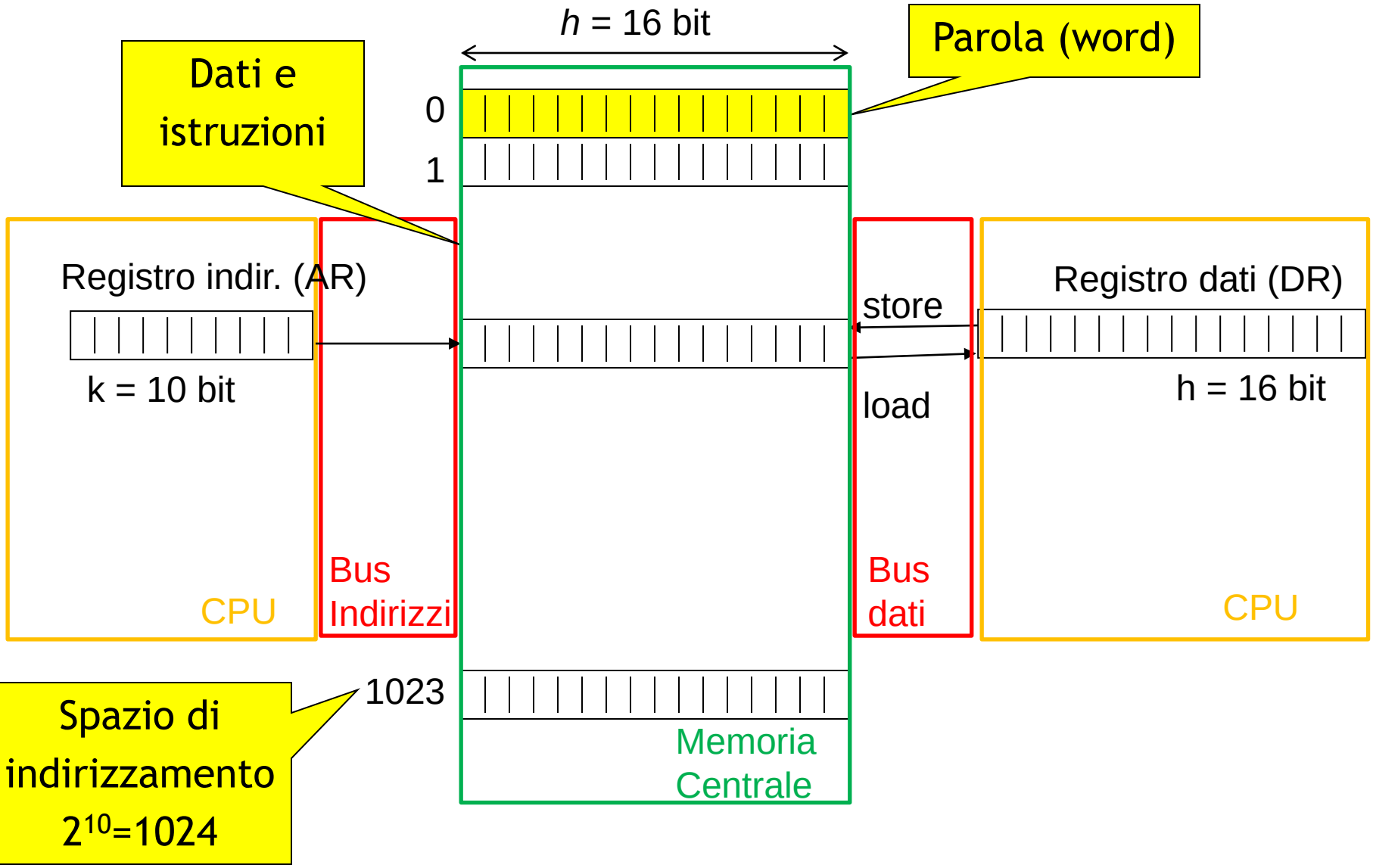
- È un insieme di connessioni che permettono di trasferire l'informazione tra **due** entità funzionali (una trasmette l'altra riceve)
- Due soli tipi di connessioni logiche, **stabilite** dalla **CPU**:
  - CPU (**master**) - memoria (**slave**)
  - CPU (**master**) - interfaccia periferica (**slave**)

  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)

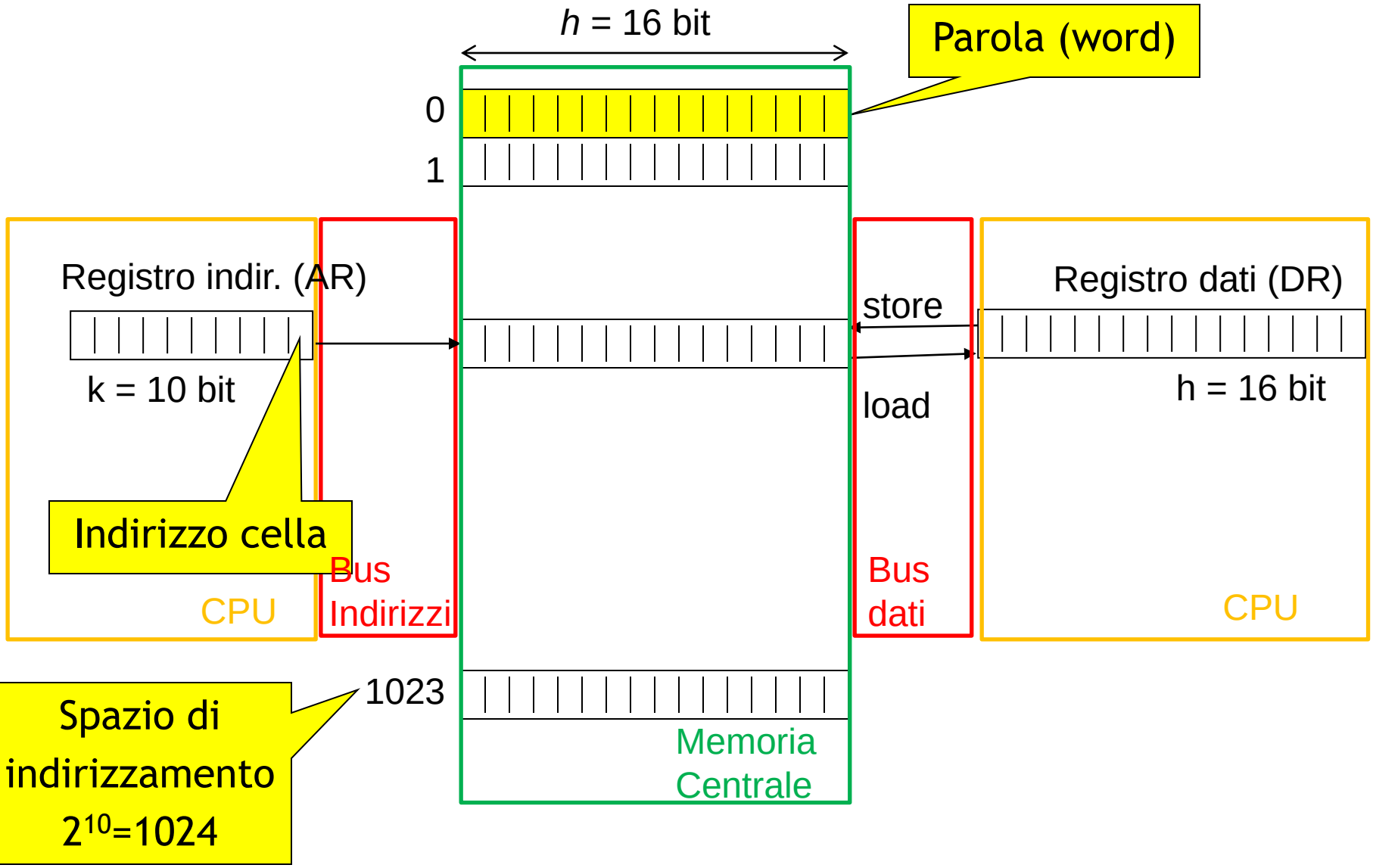
Lettura e Scrittura dalla Memoria Centrale



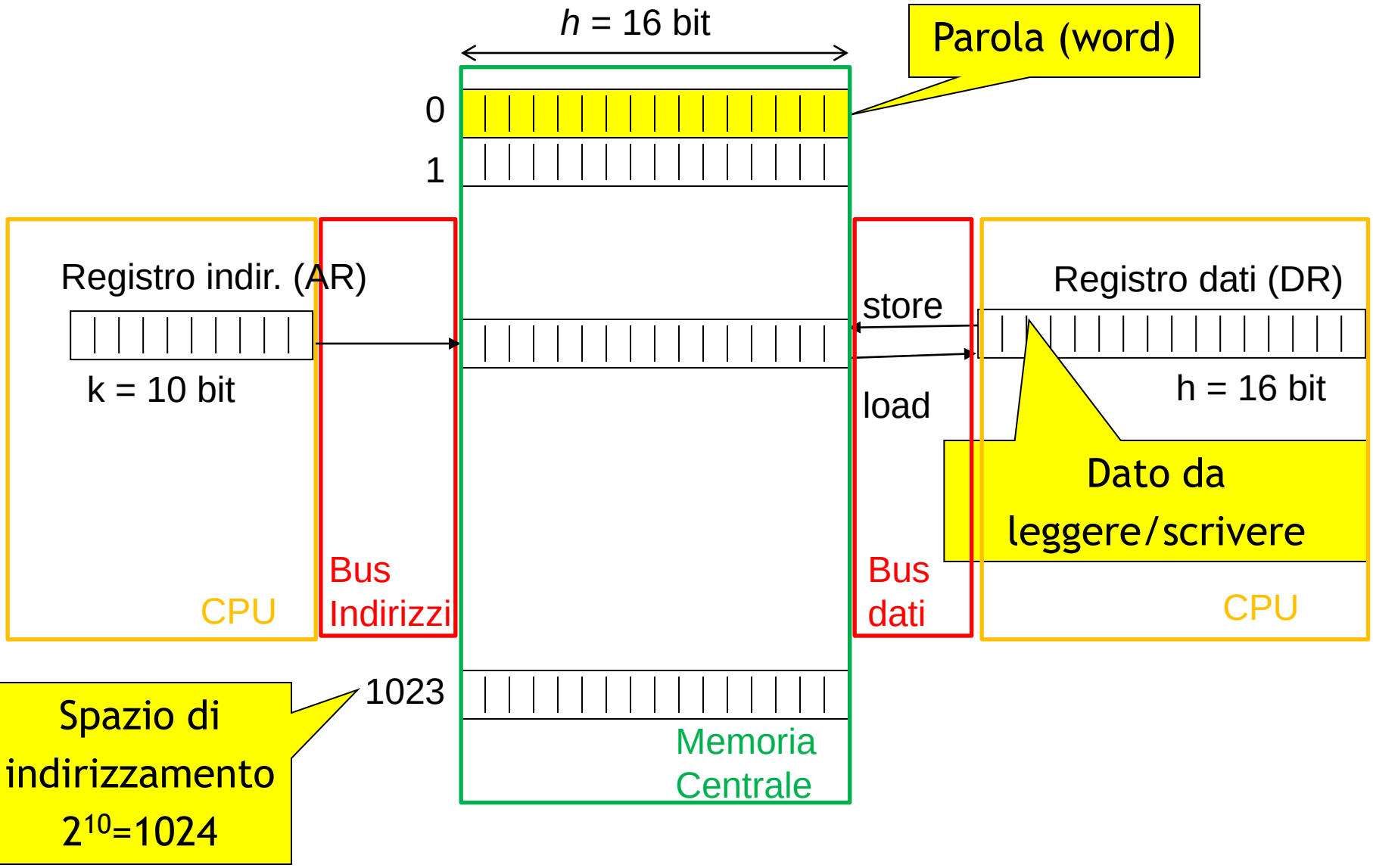
Lettura e Scrittura dalla Memoria Centrale



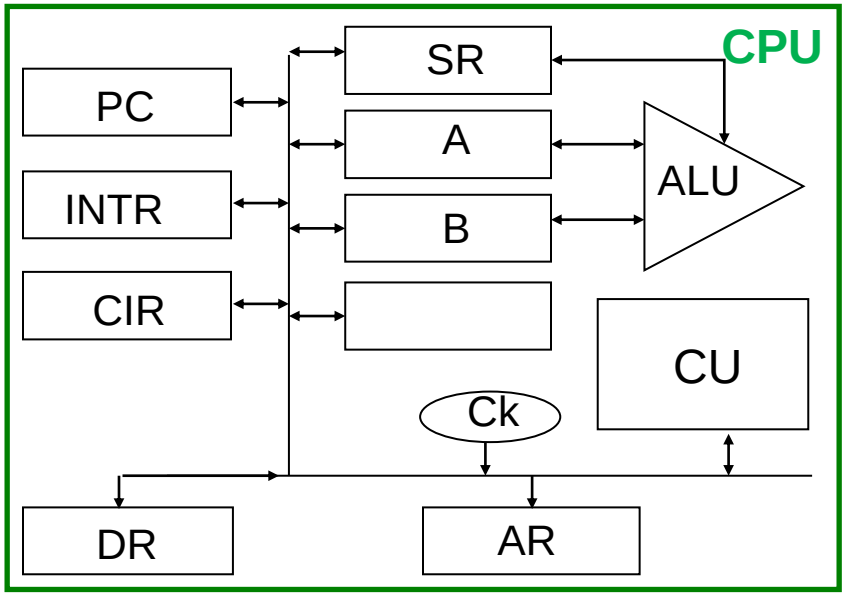
Lettura e Scrittura dalla Memoria Centrale

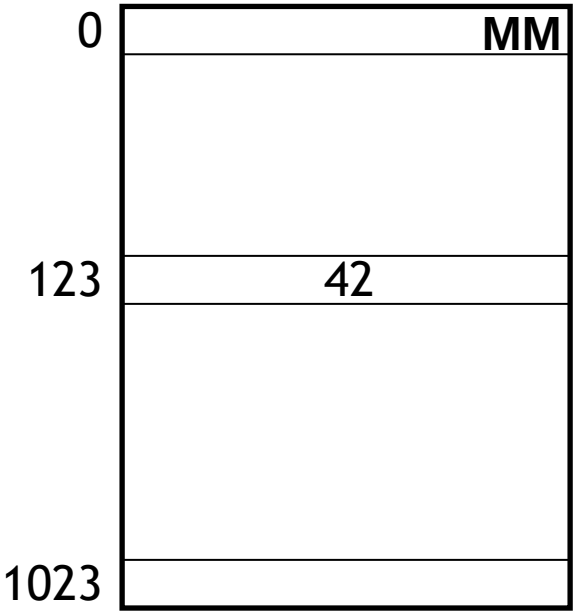


Lettura e Scrittura dalla Memoria Centrale

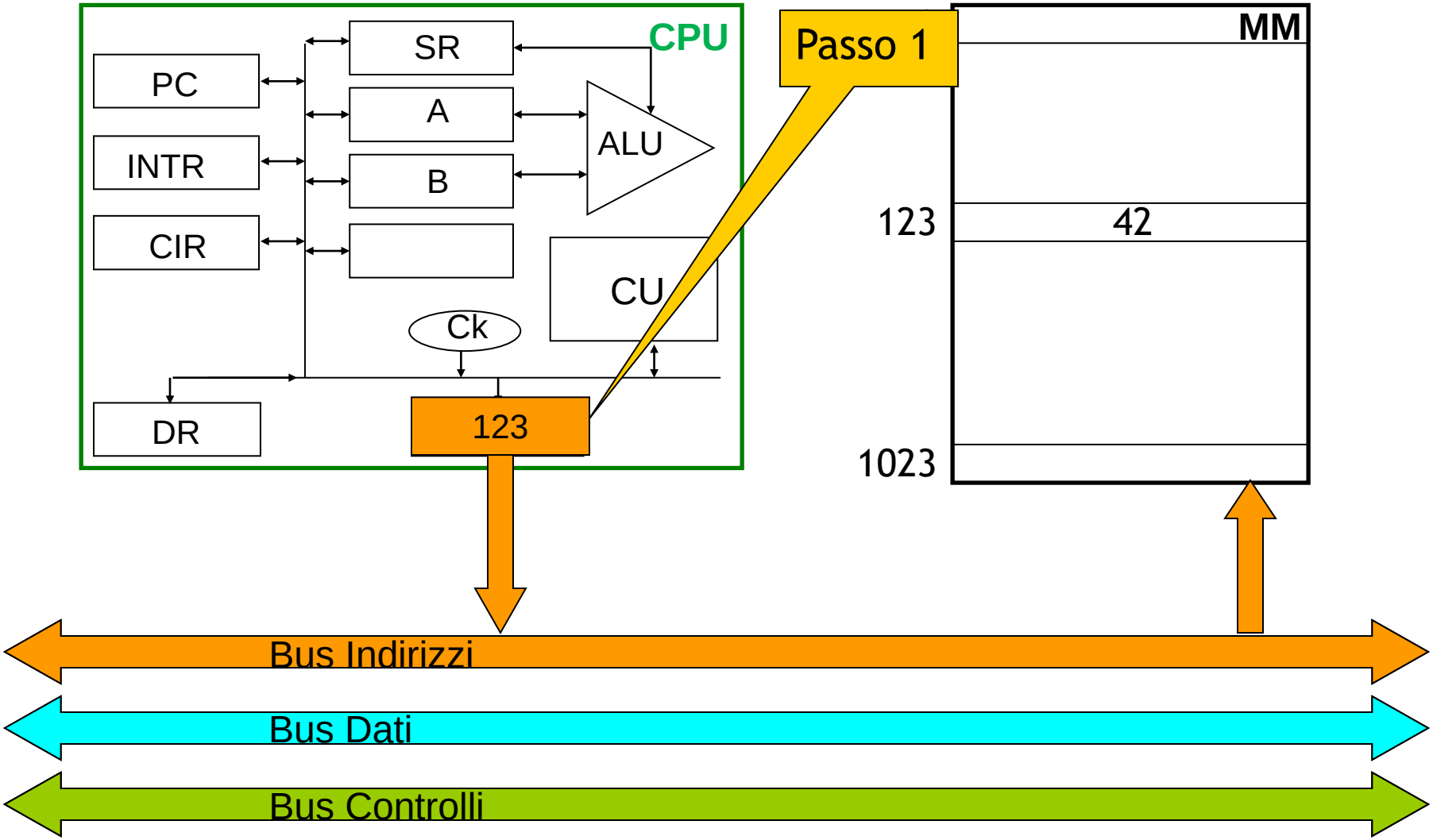


Sequenza di Lettura

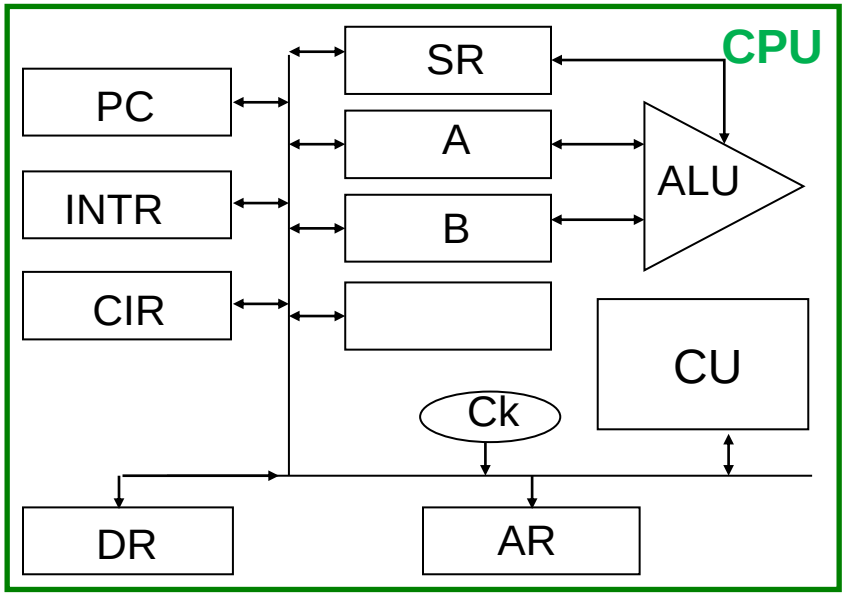


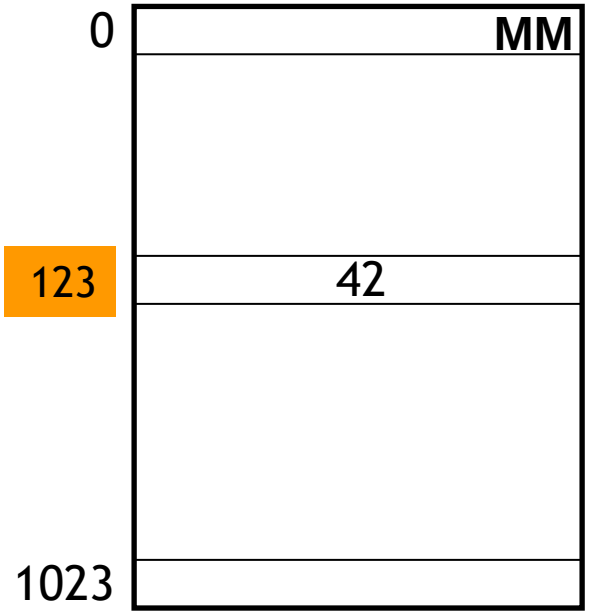


Sequenza di Lettura

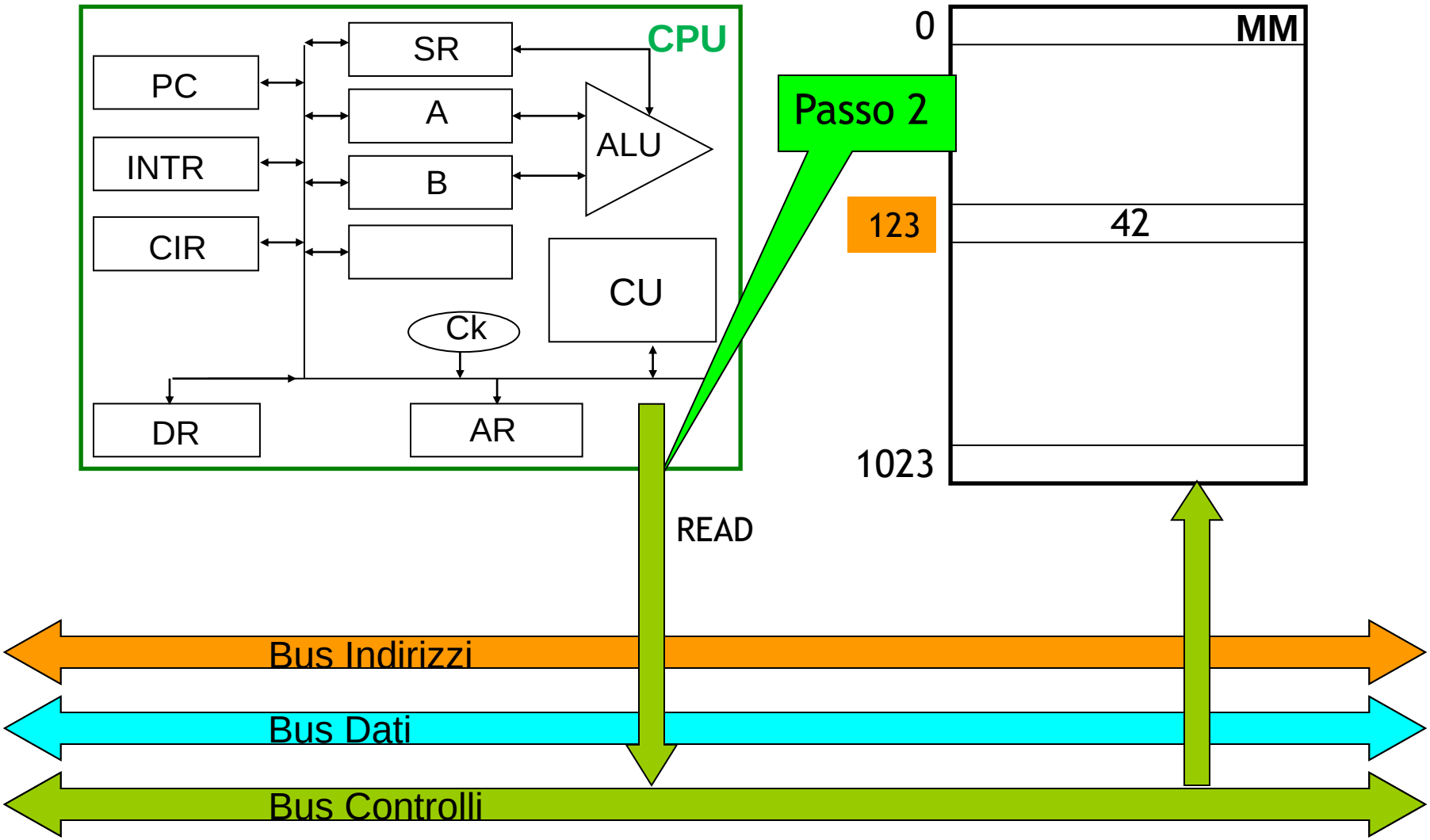


Sequenza di Lettura

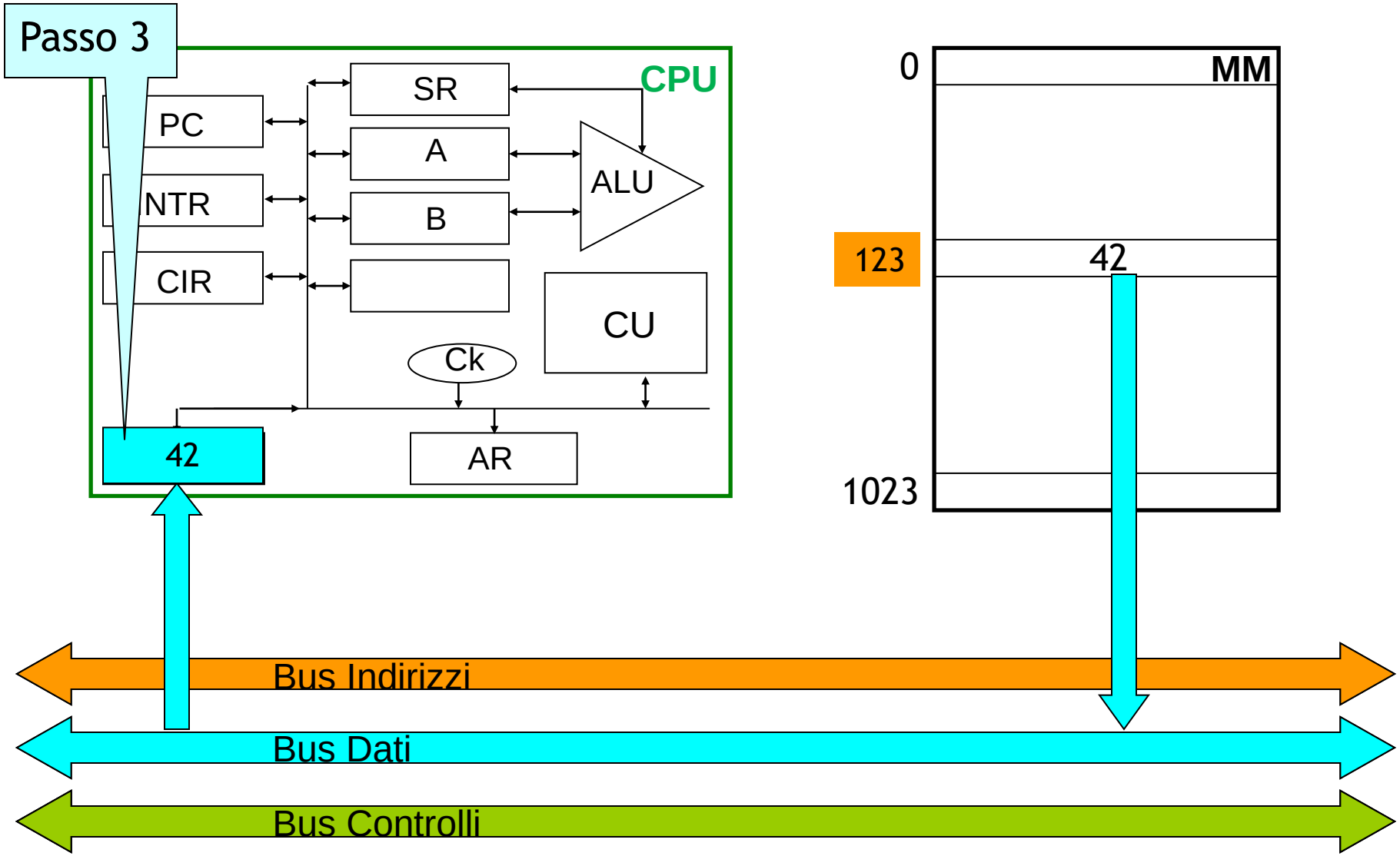




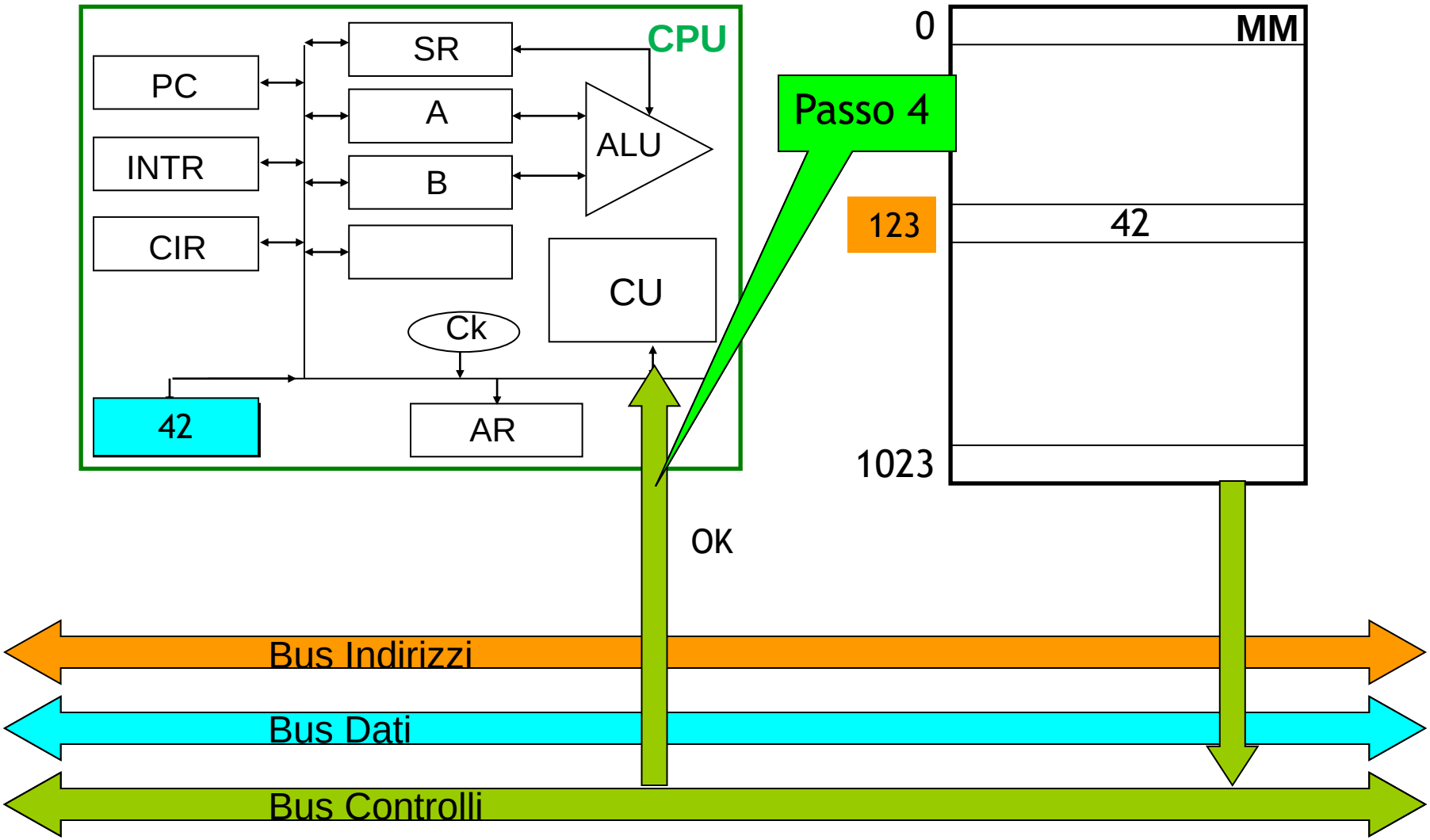
Sequenza di Lettura



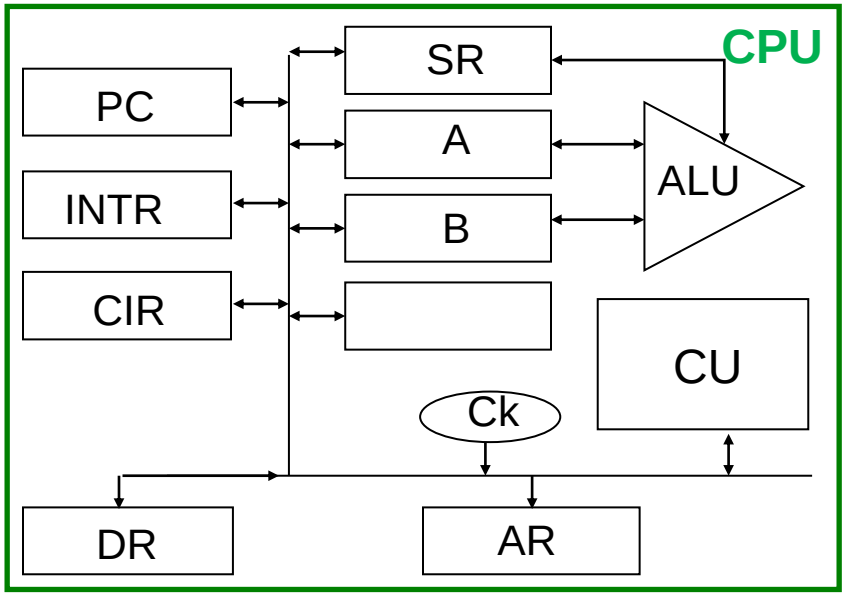
Sequenza di Lettura

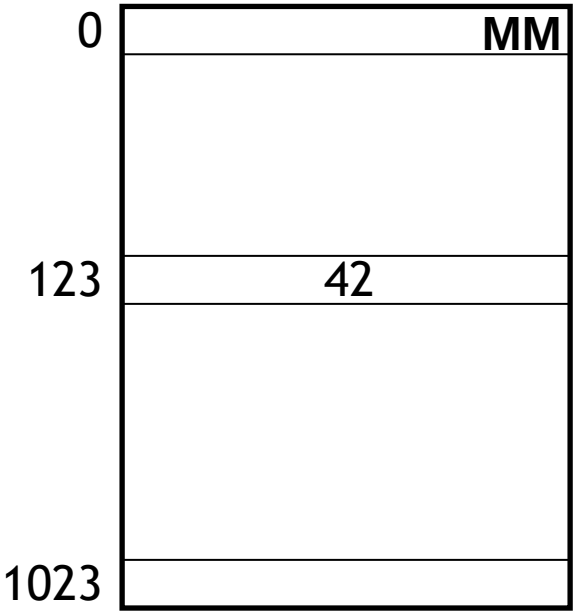


Sequenza di Lettura

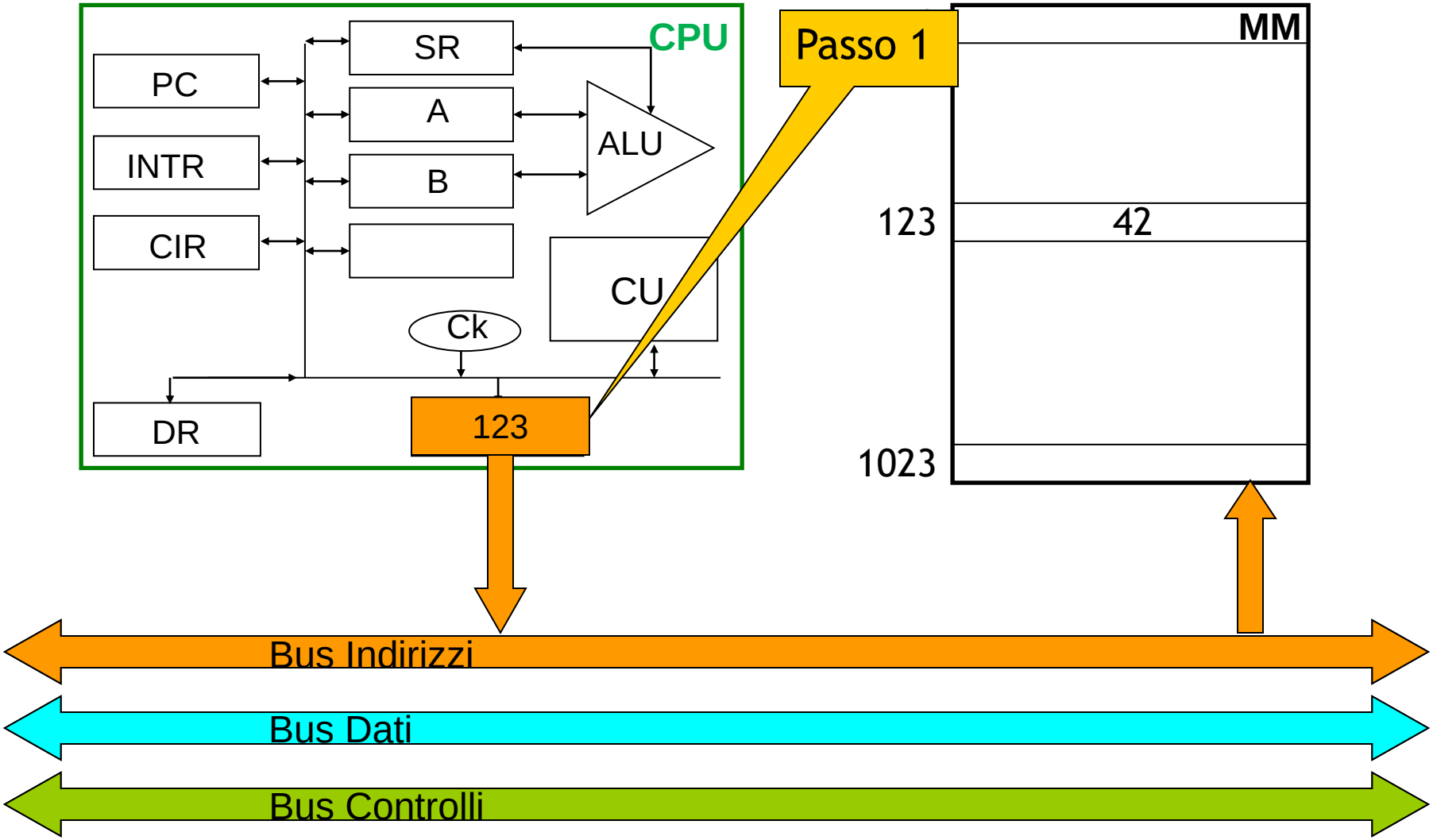


Sequenza di Scrittura

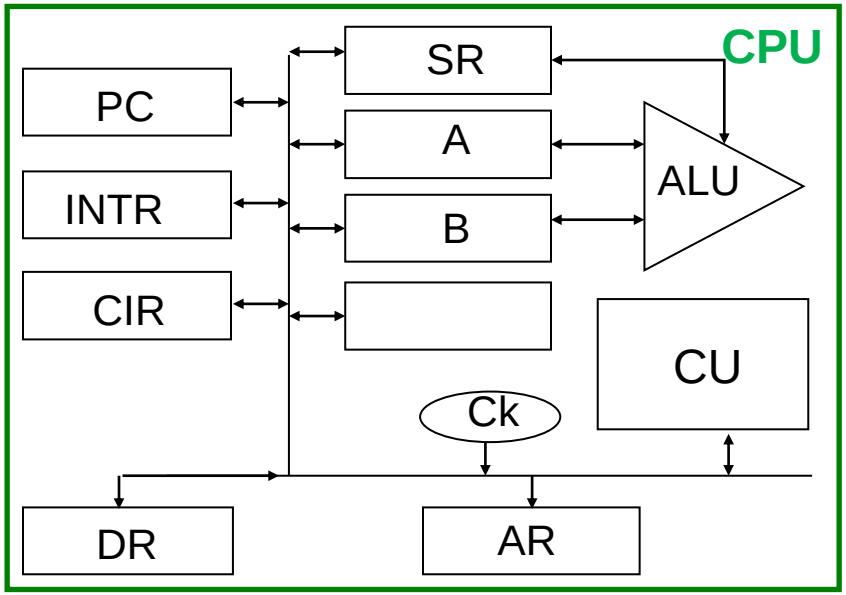


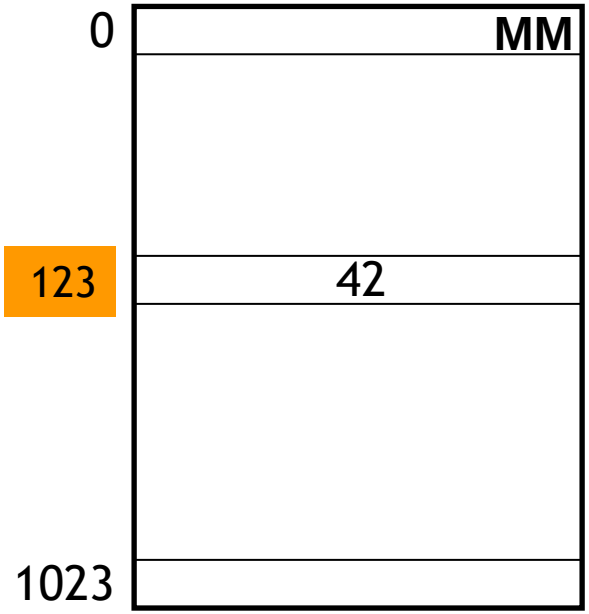


Sequenza di Scrittura

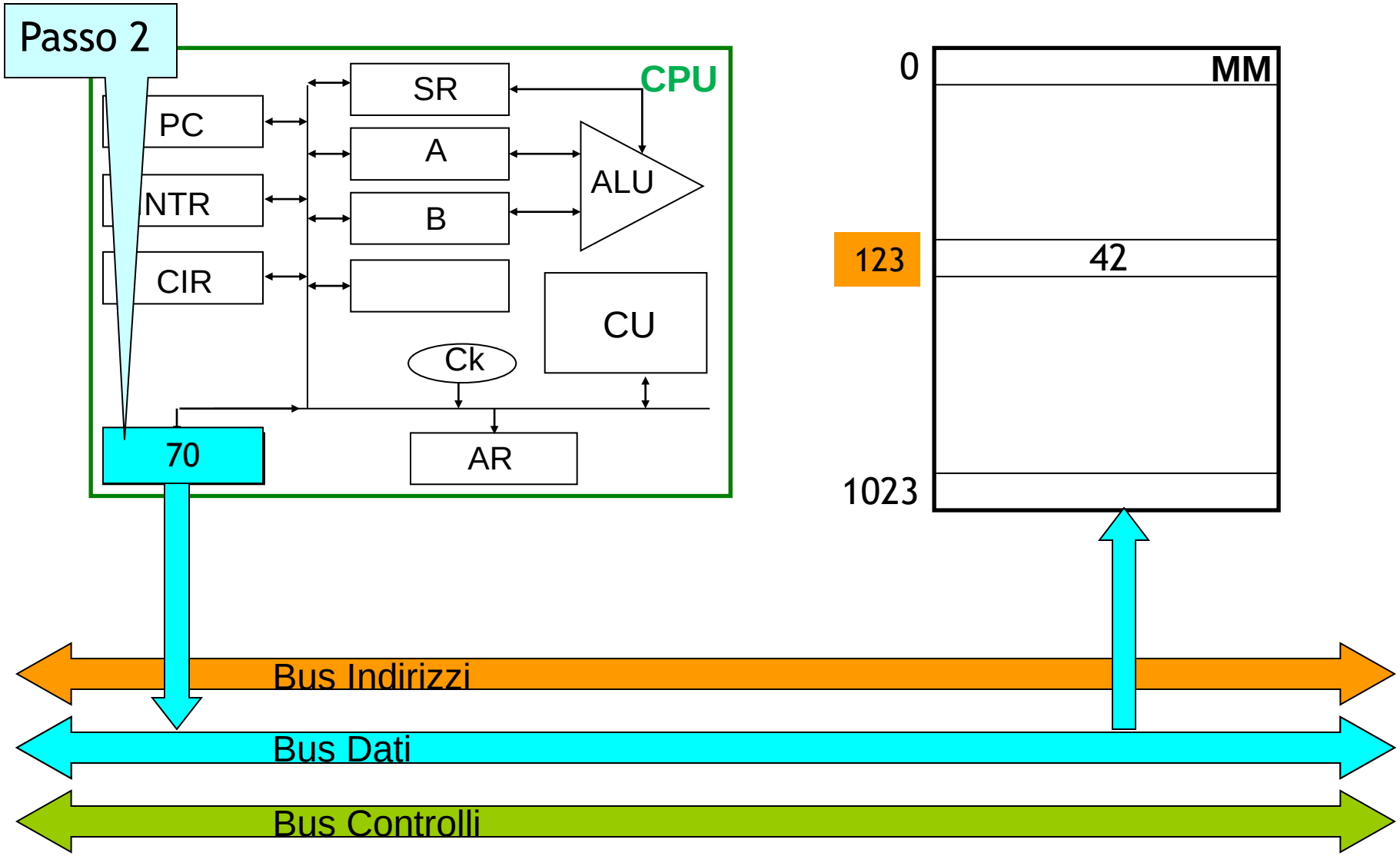


Sequenza di Scrittura

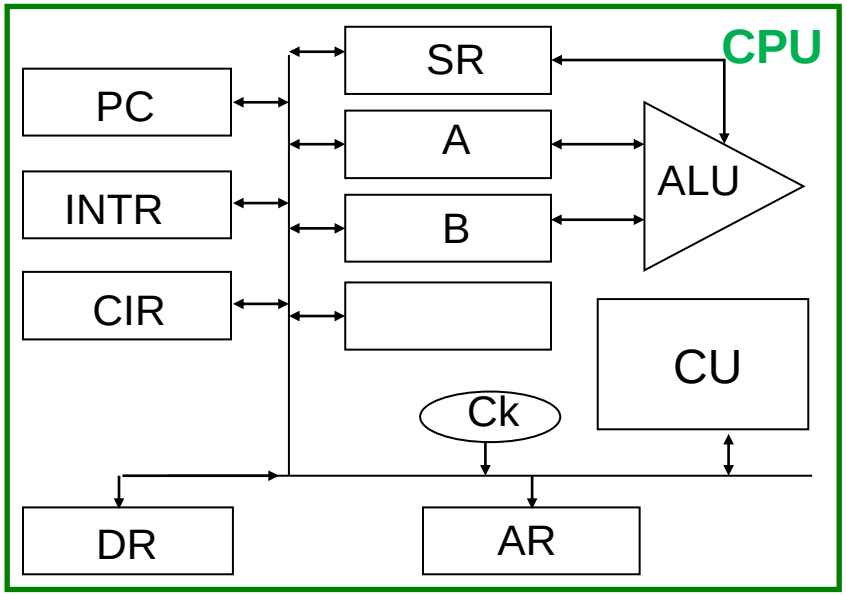


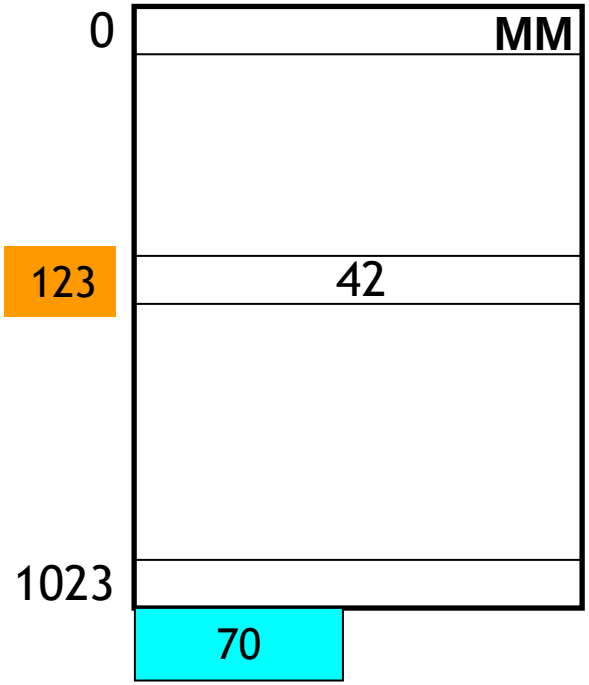


Sequenza di Scrittura

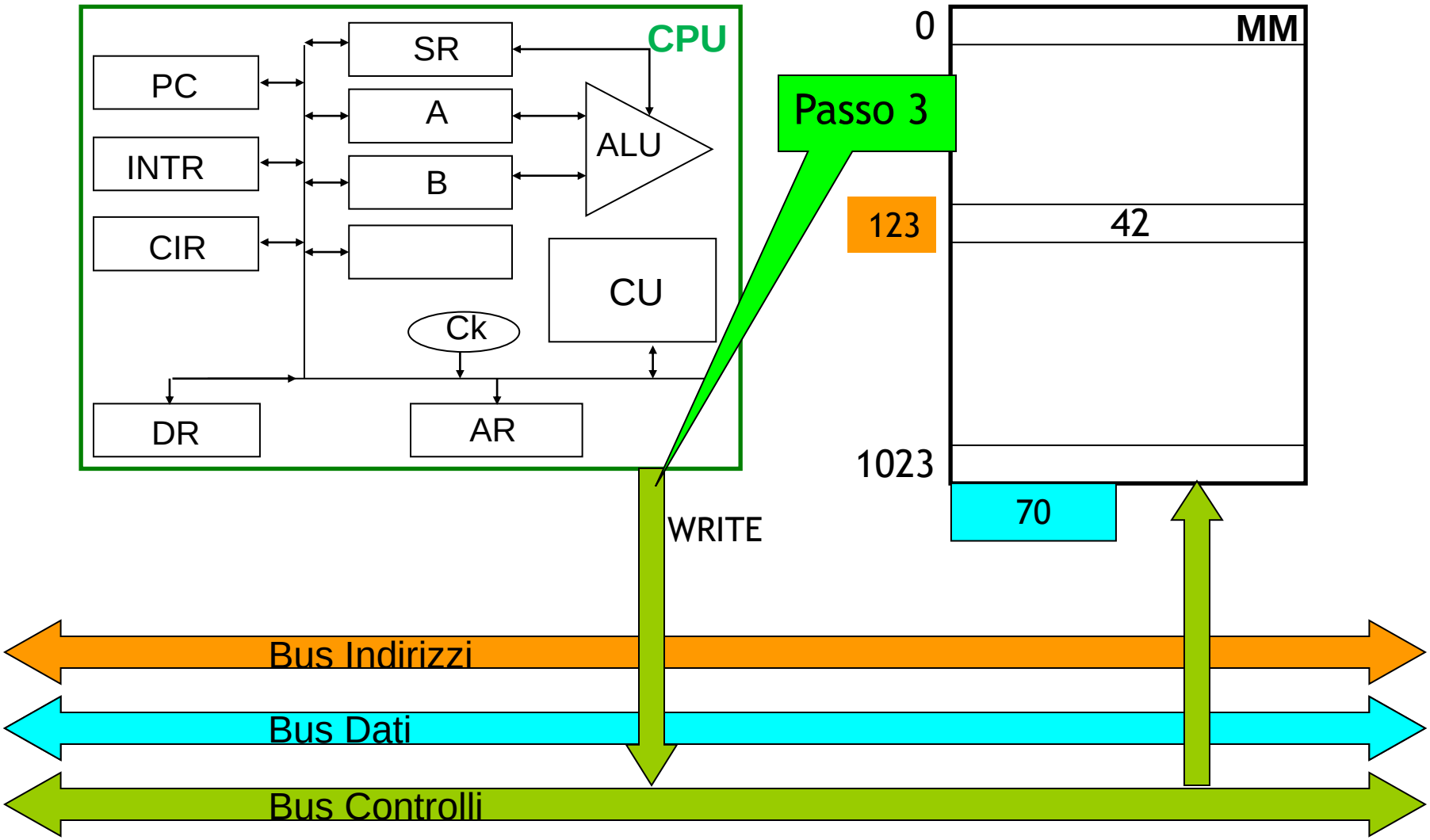


Sequenza di Scrittura

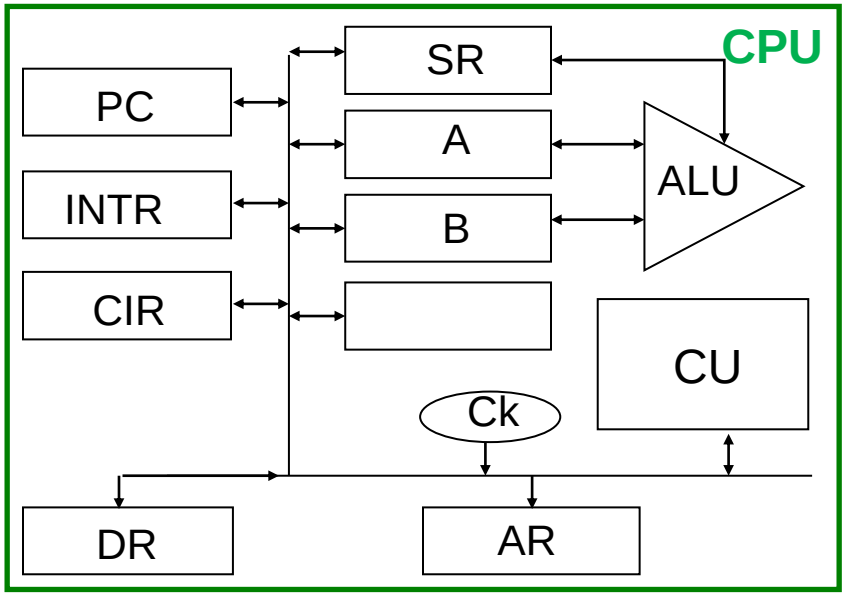


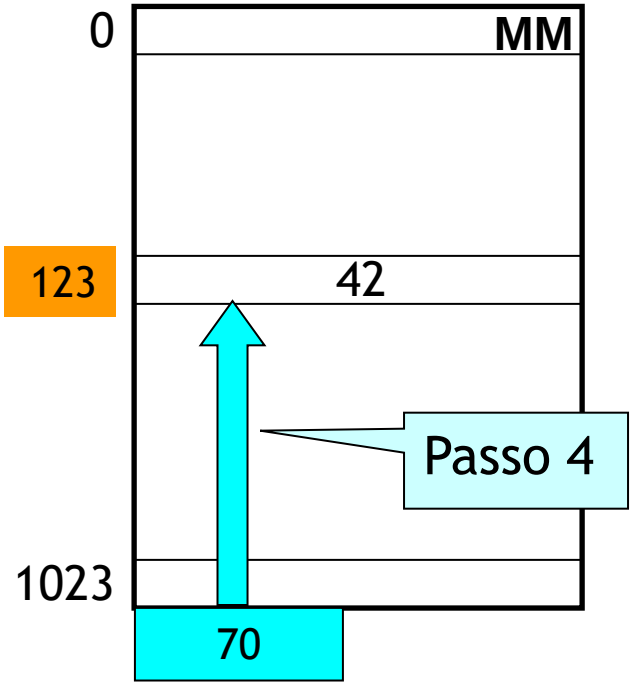


Sequenza di Scrittura



Sequenza di Scrittura



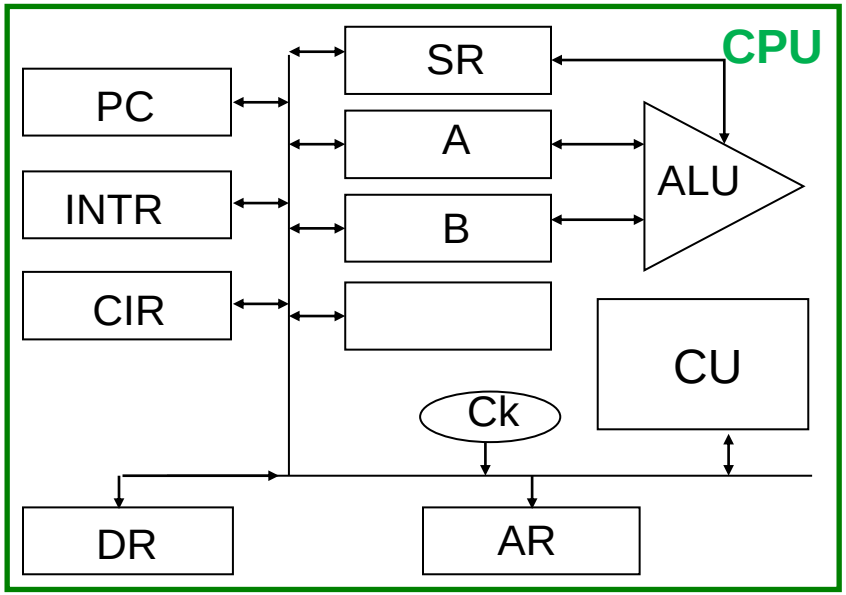


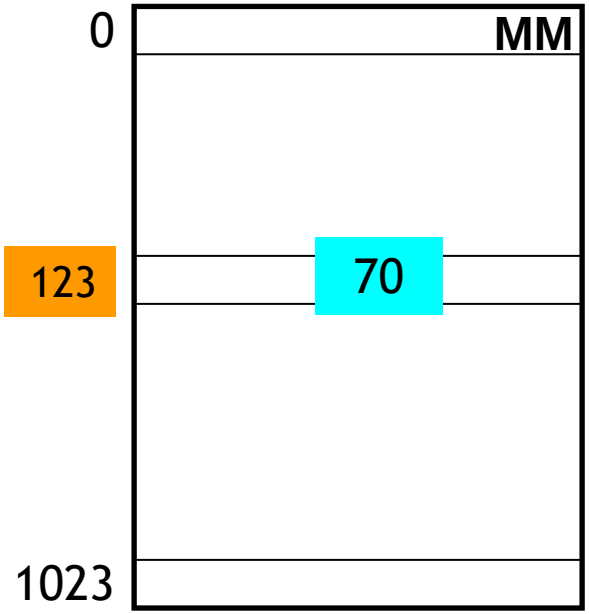
Bus Indirizzi

Bus Dati

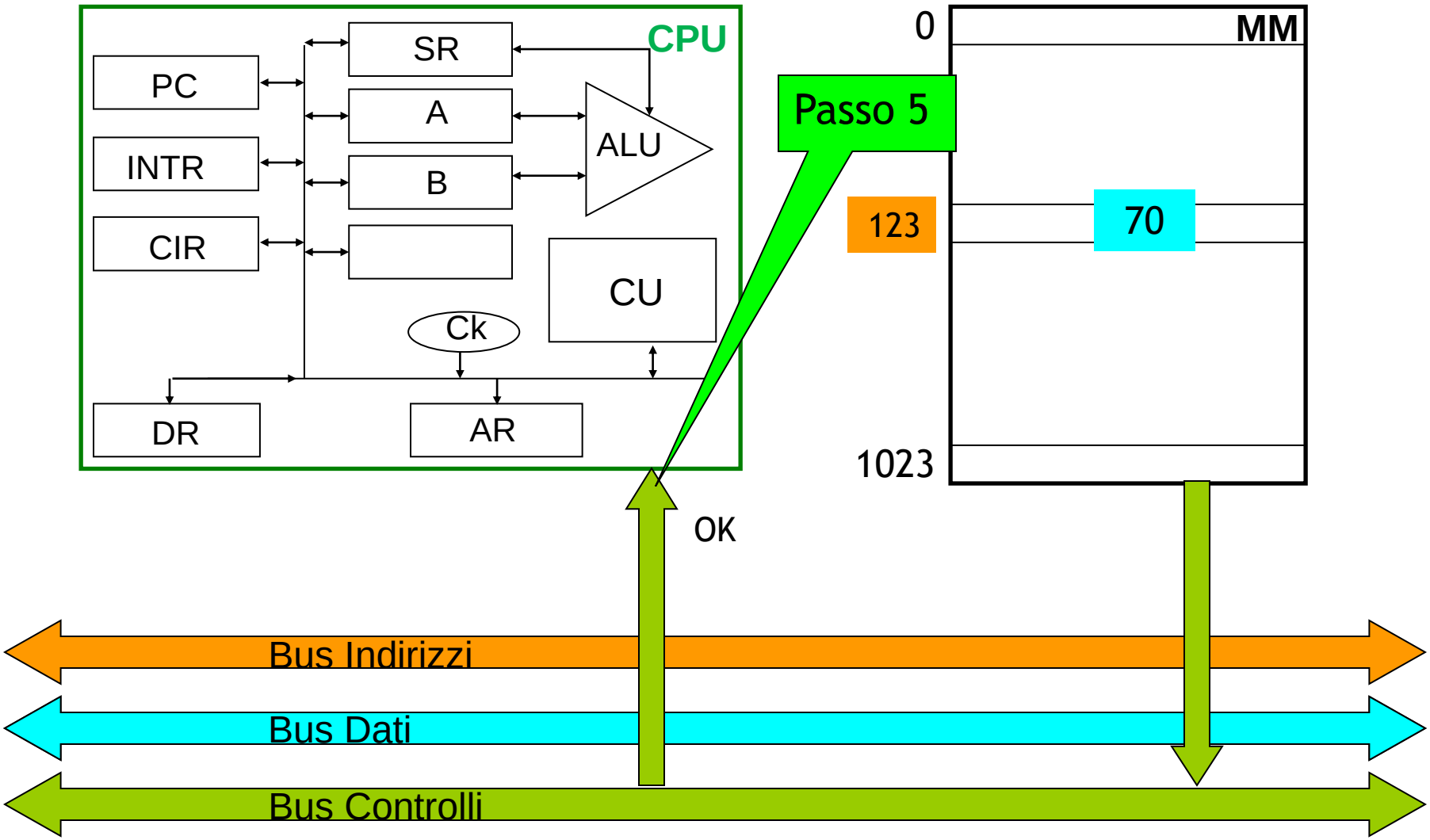
Bus Controlli

Sequenza di Scrittura





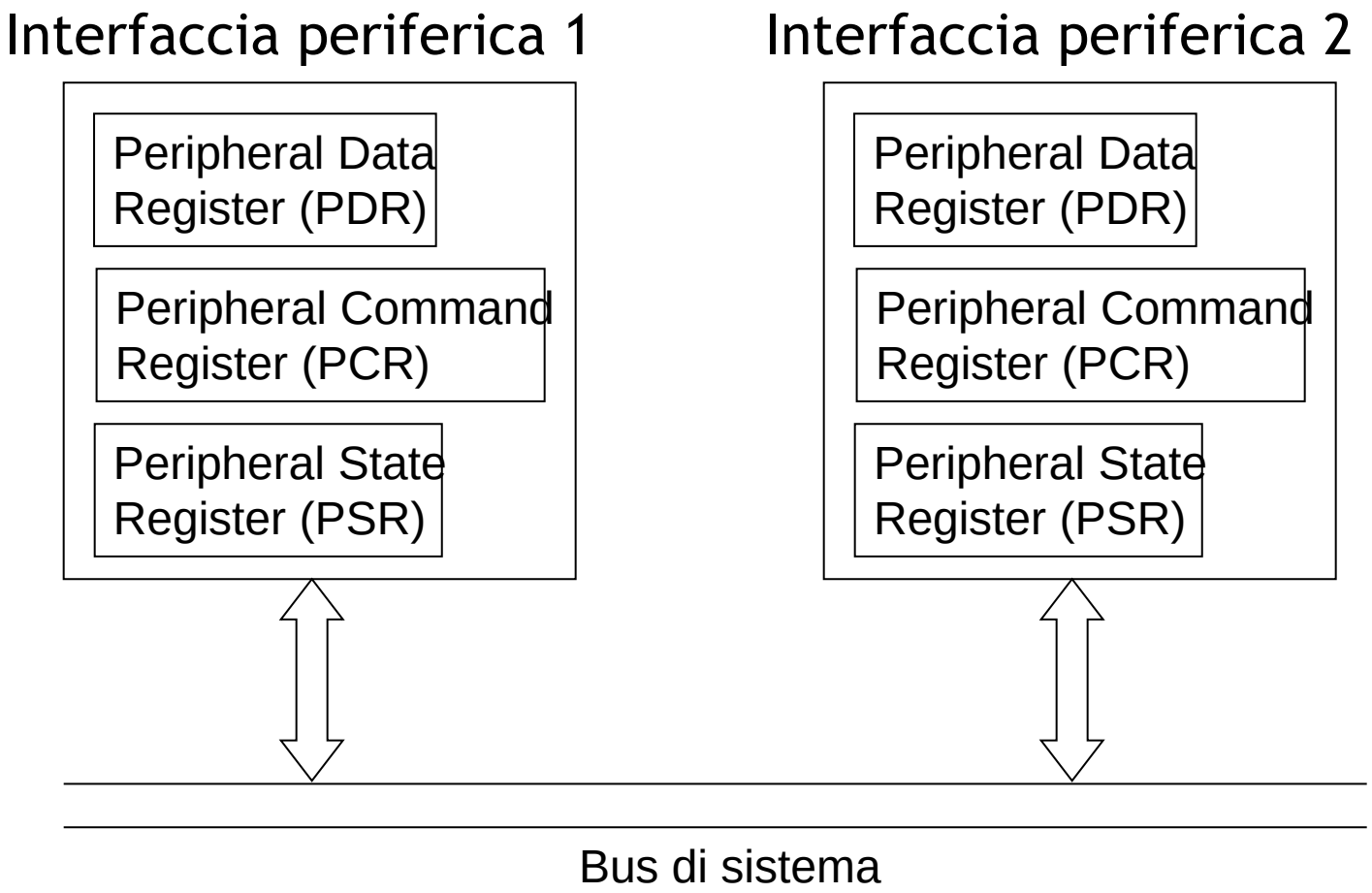
Sequenza di Scrittura



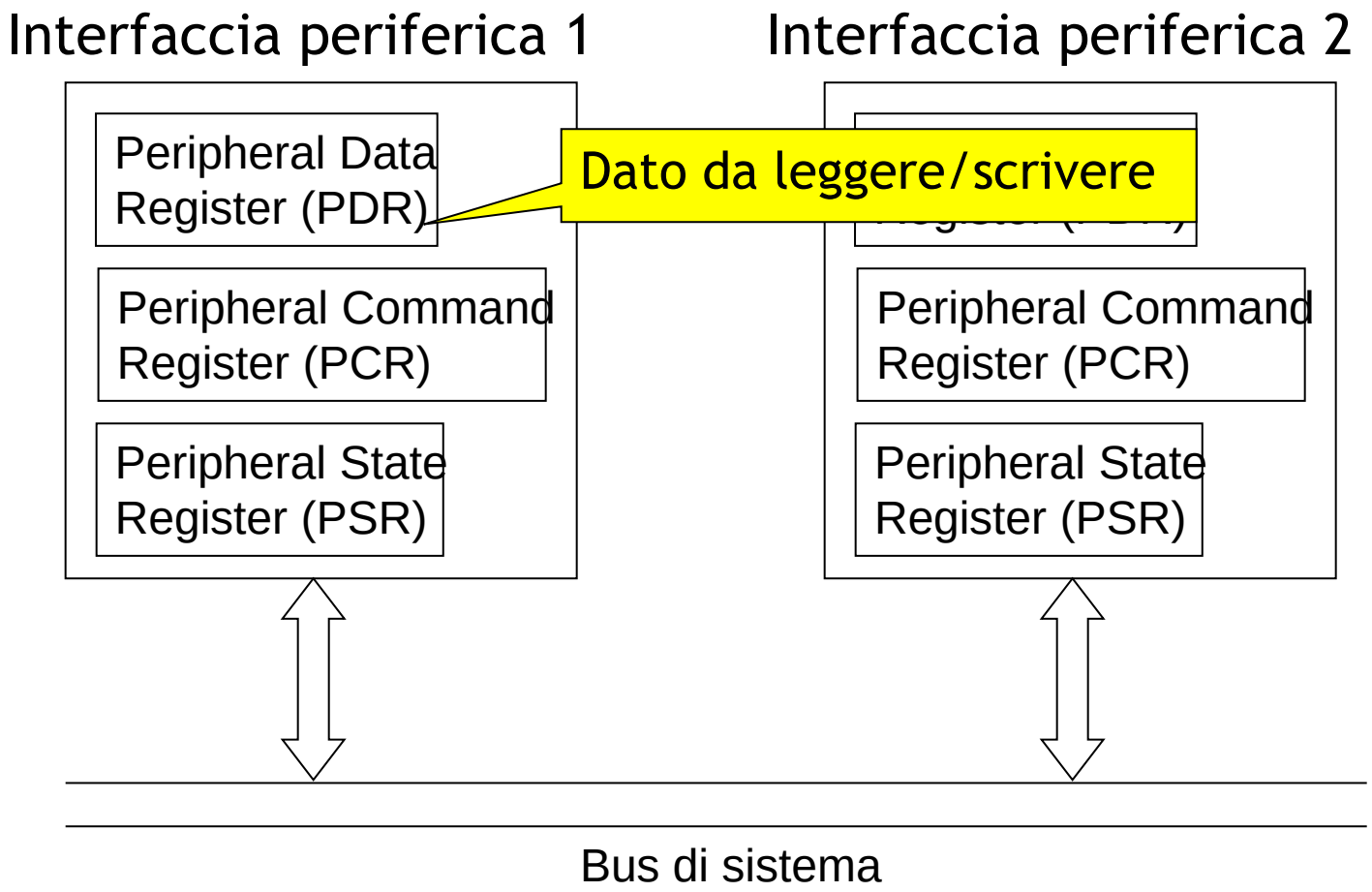
Interfacce alle Periferiche

- 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

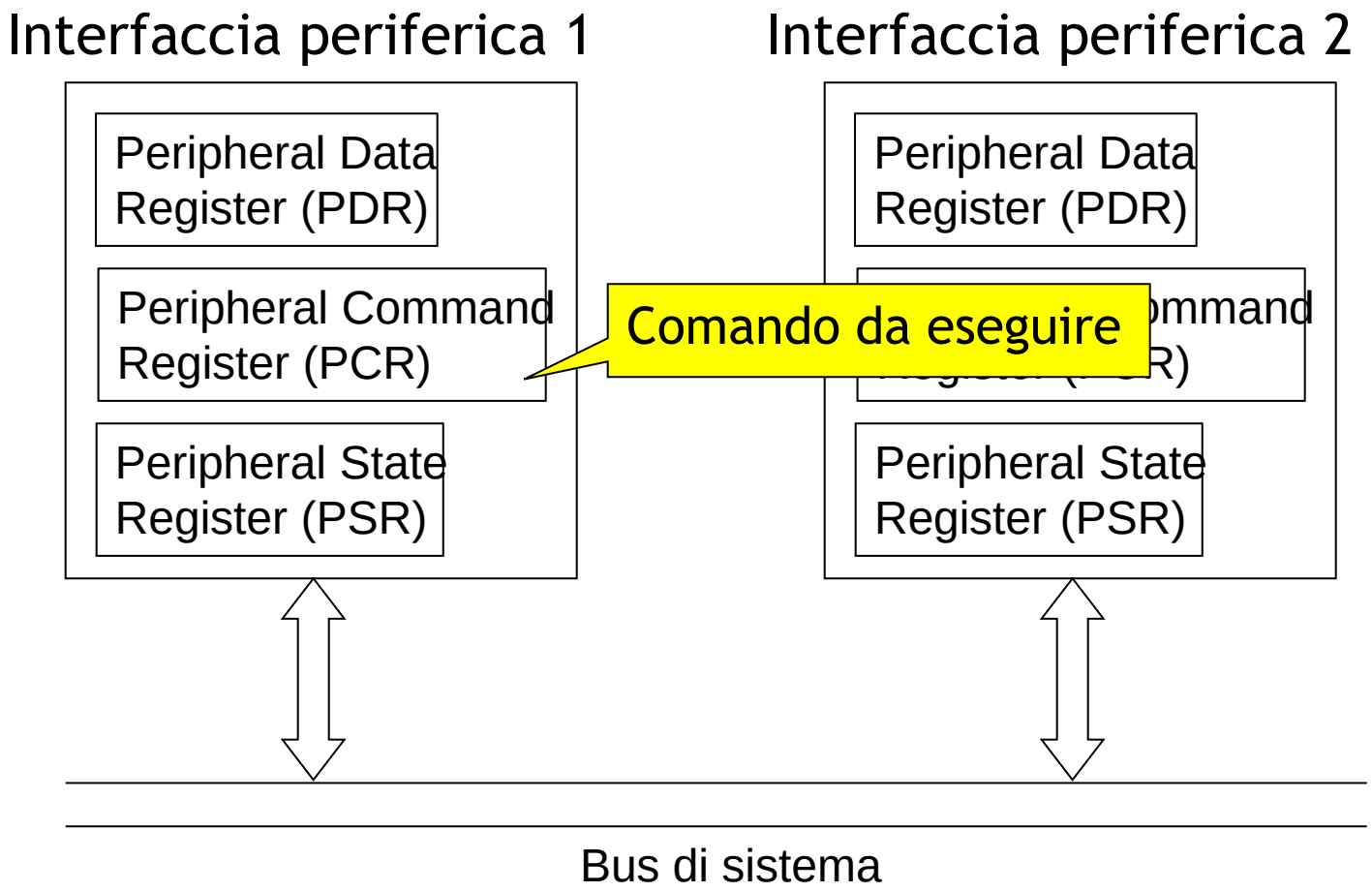
Interfacce alle Periferiche



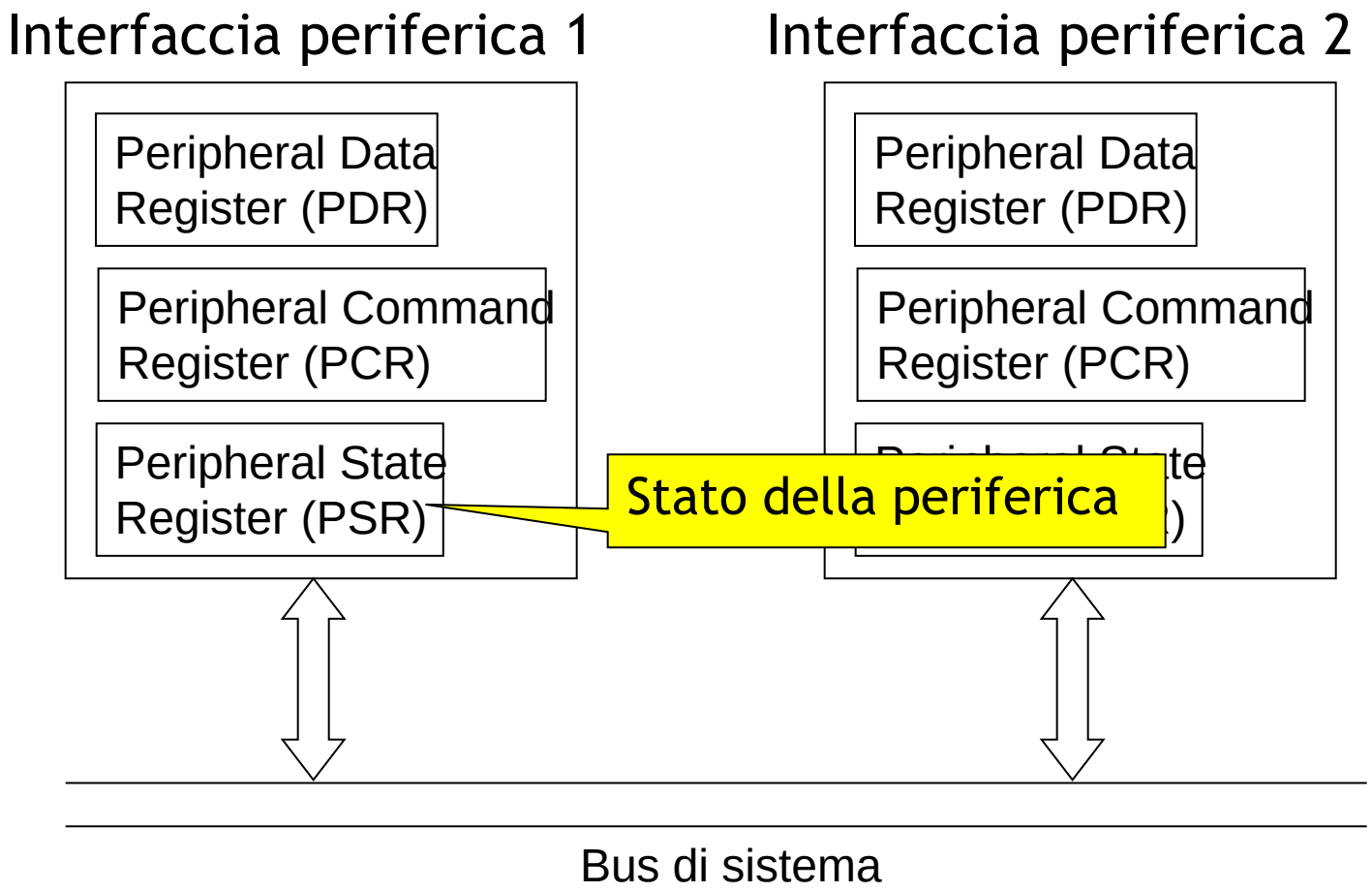
Interfacce alle Periferiche



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**

- **Codice operativo**: indica quale istruzione si deve eseguire
- **Indirizzo operando**: indica in quale punto della memoria si trova l'operando

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

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

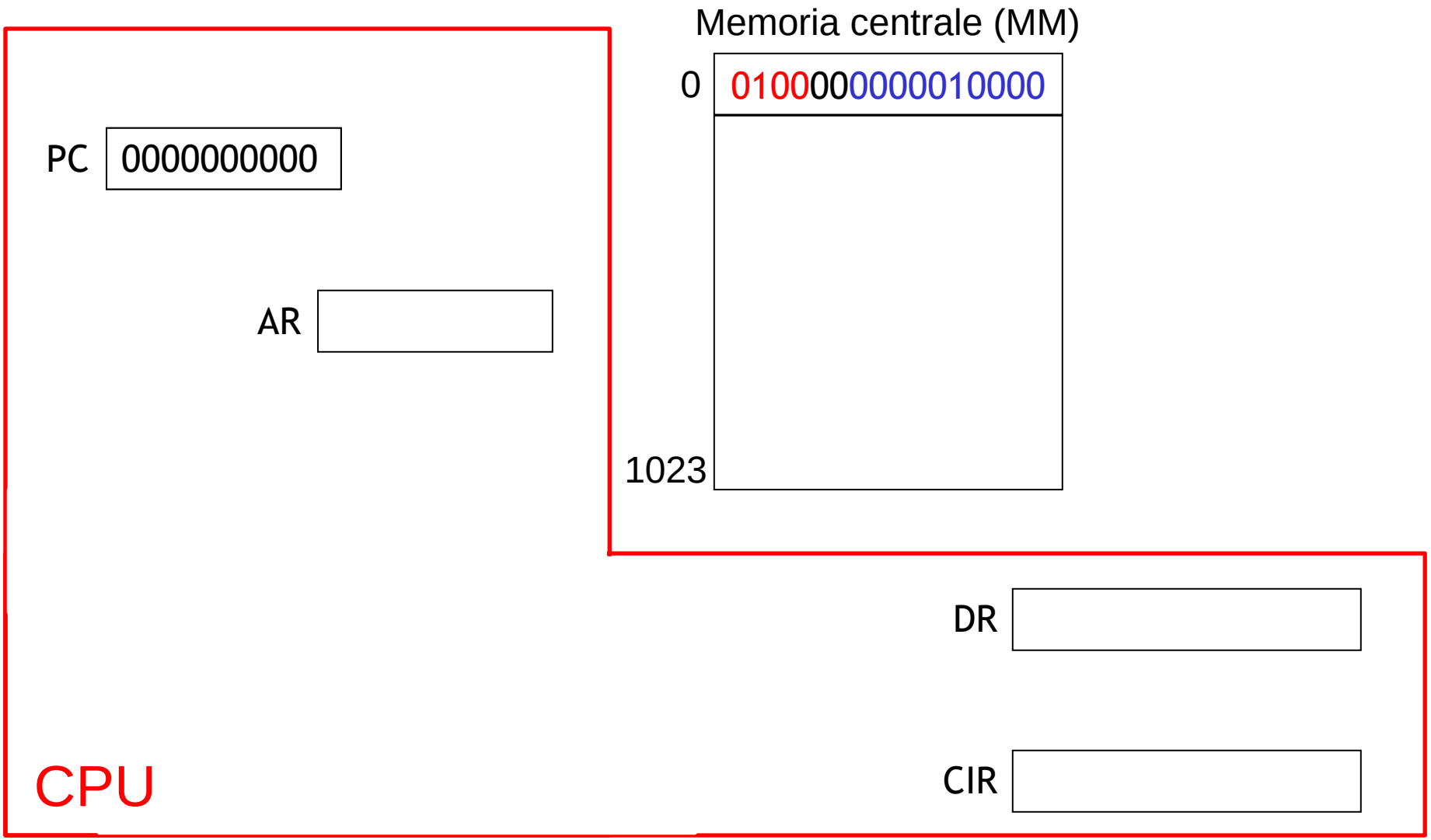
Consideriamo le seguenti istruzioni eseguibili dalla CPU

- 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)

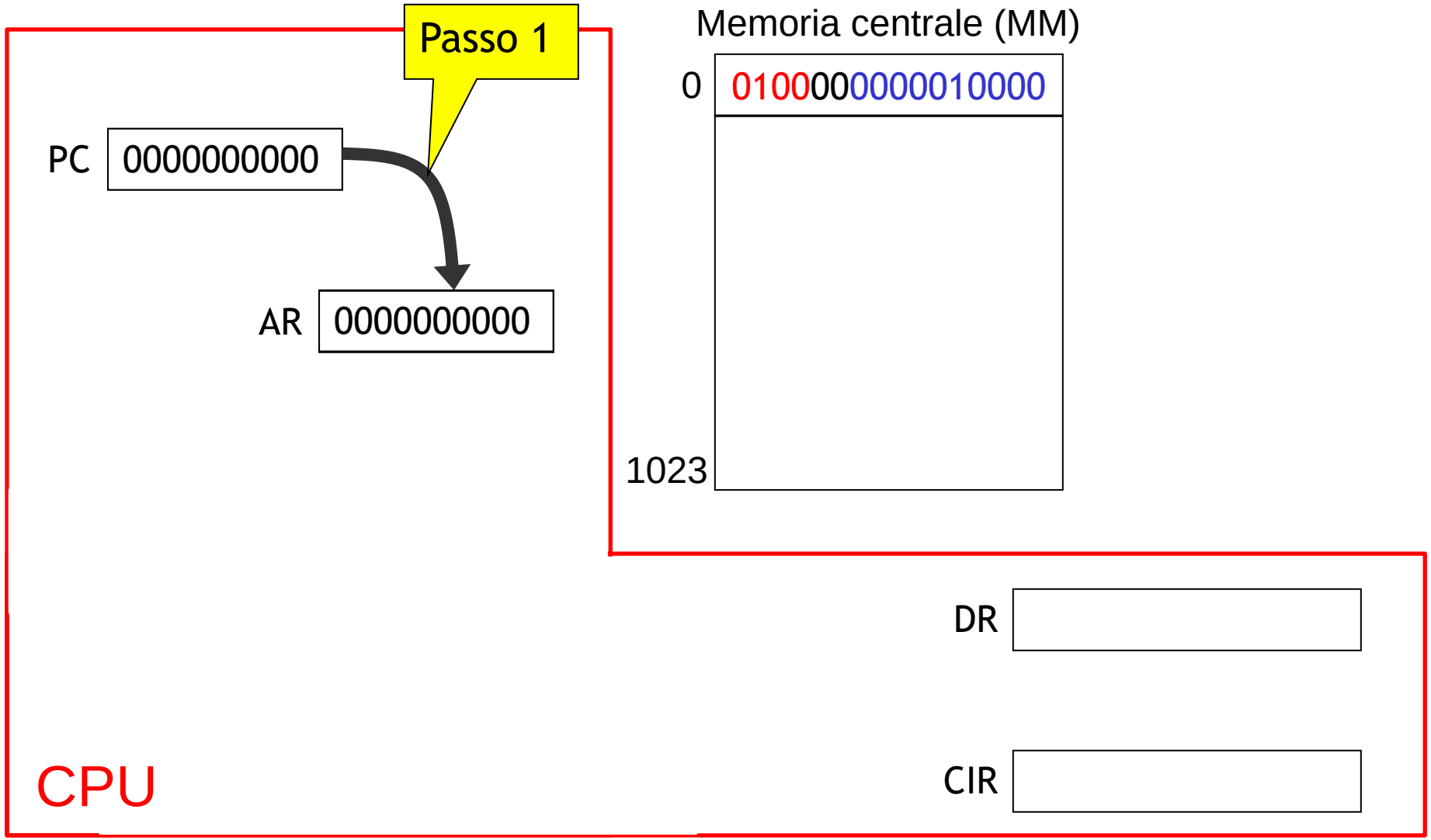
Le Tre Fasi Per Eseguire un'Istruzioni

1. **Fetch:** Acquisizione dell'istruzione dalla MM
   1. Trasferimento da PC a AR dell' indirizzo della 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.

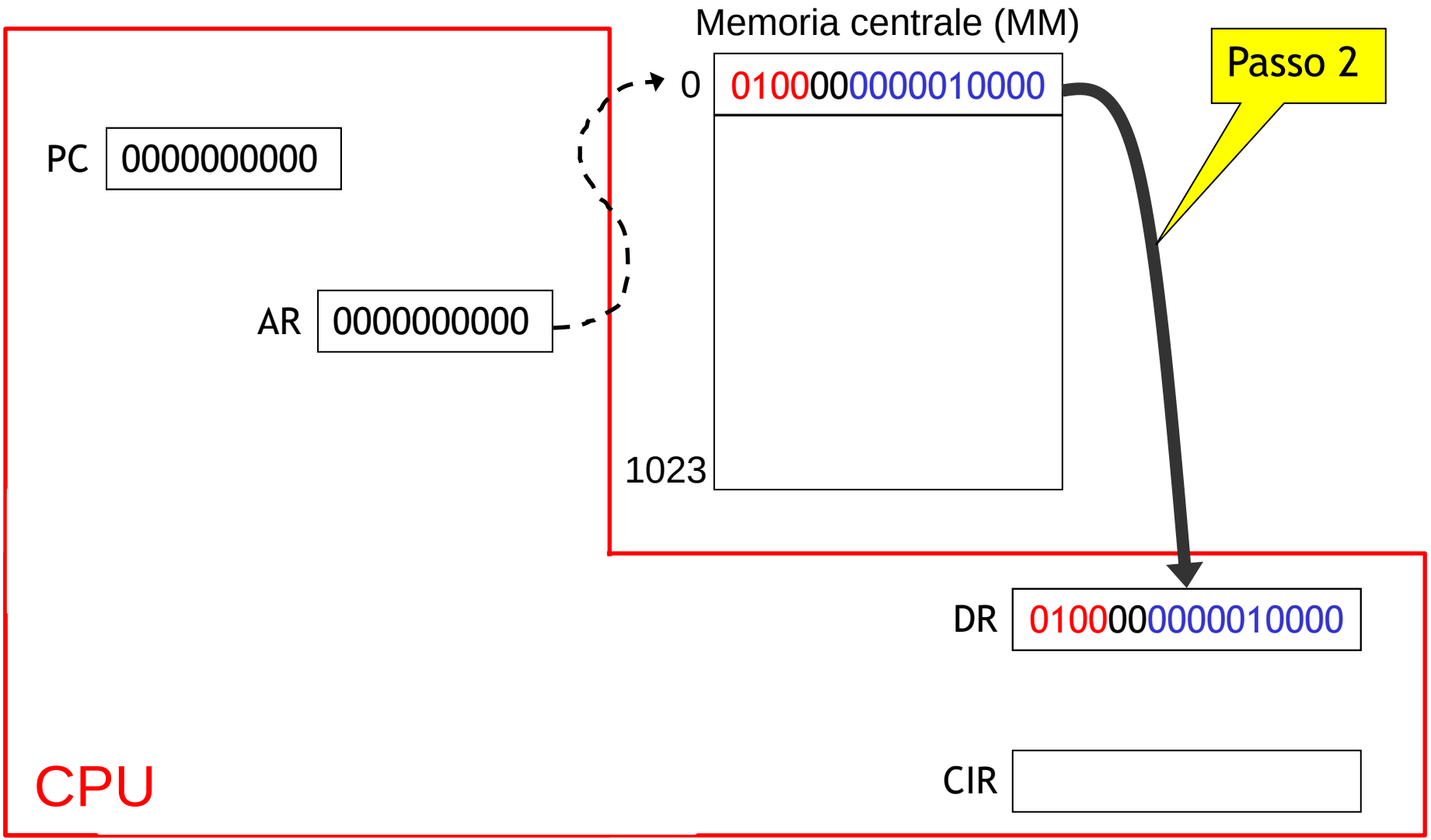
Fase di Fetch



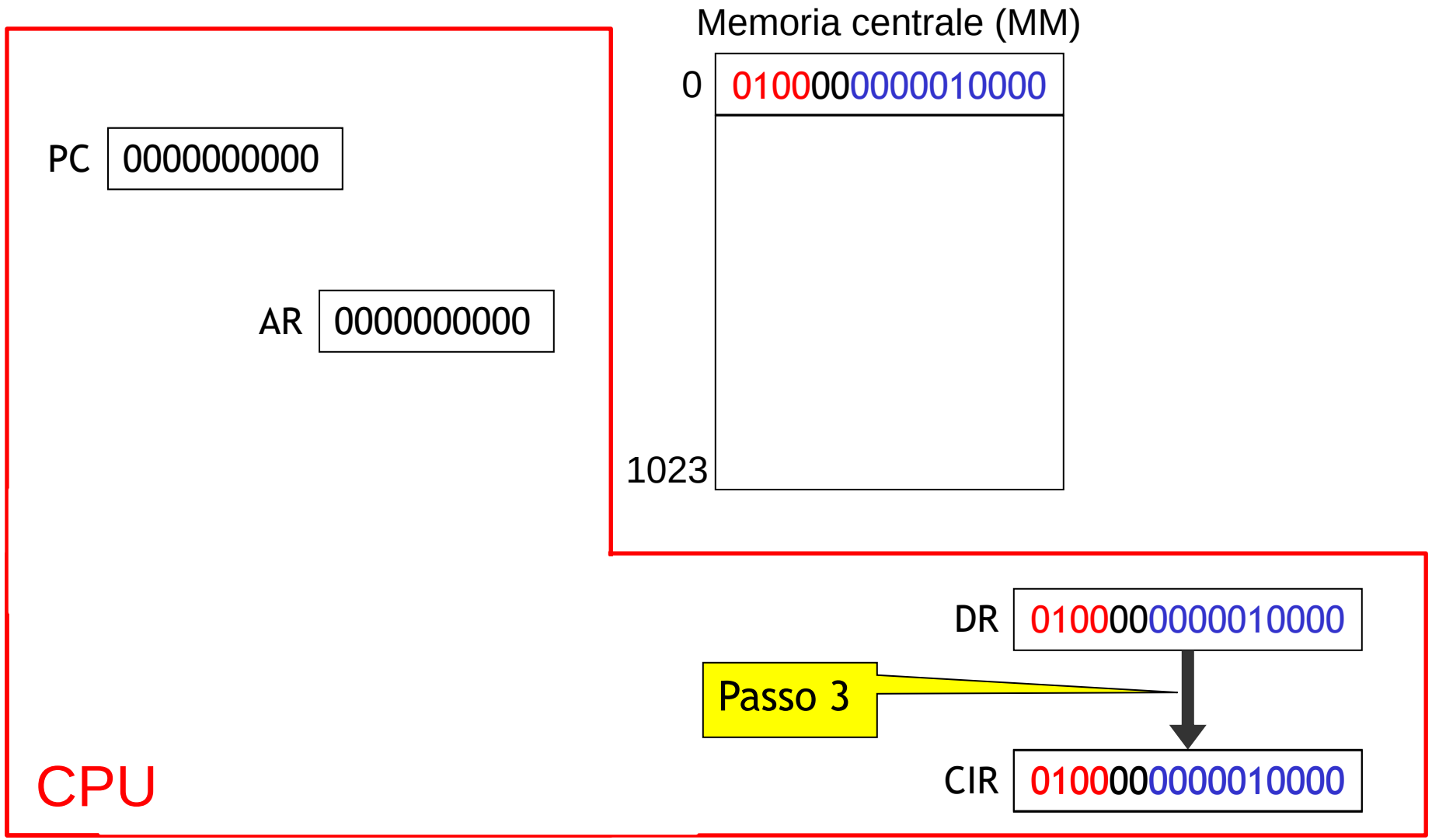
Fase di Fetch



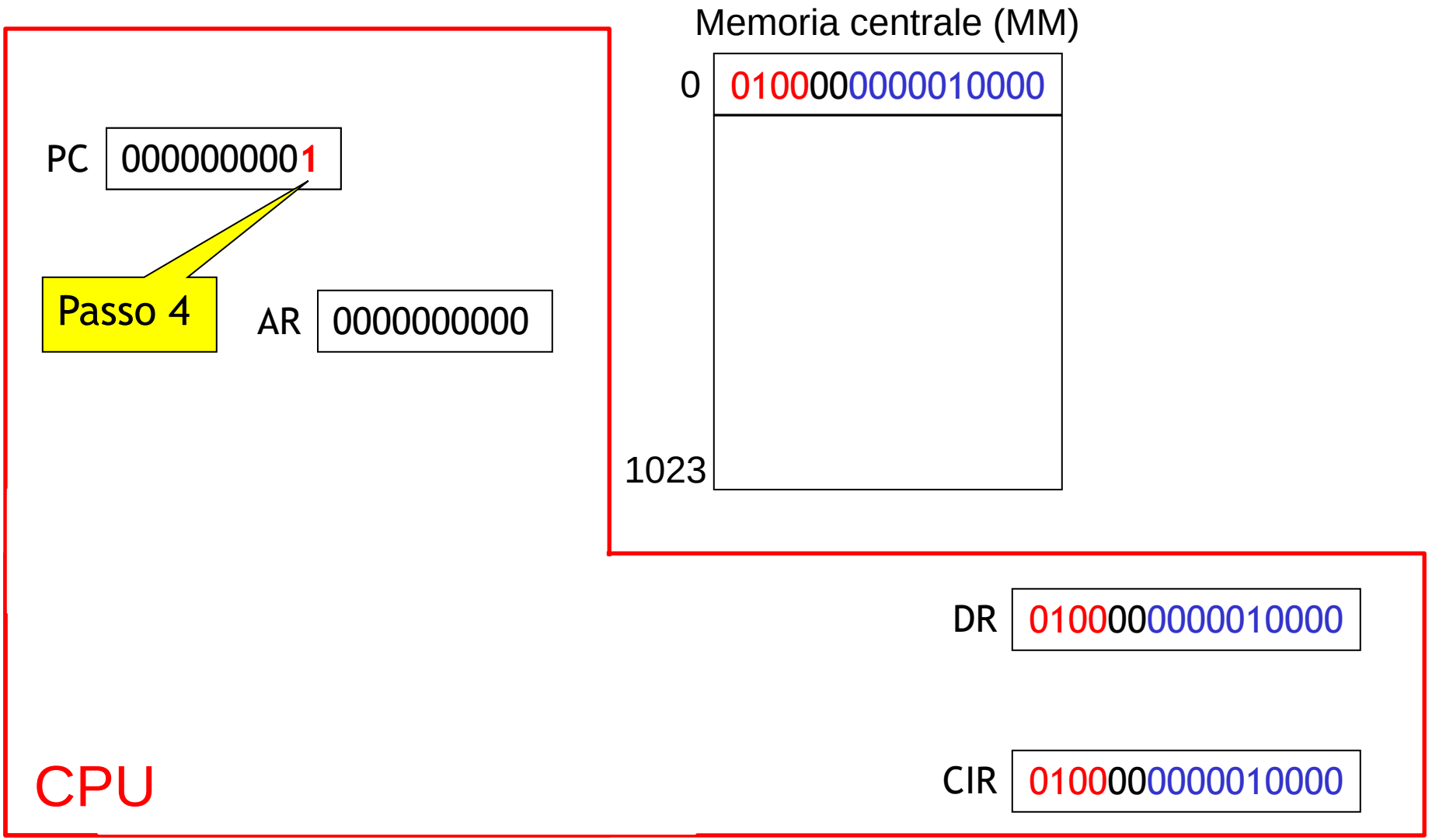
Fase di Fetch



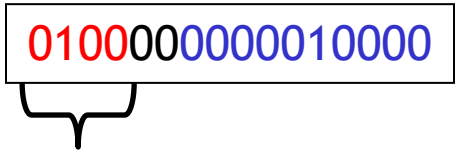
Fase di Fetch



Fase di Fetch

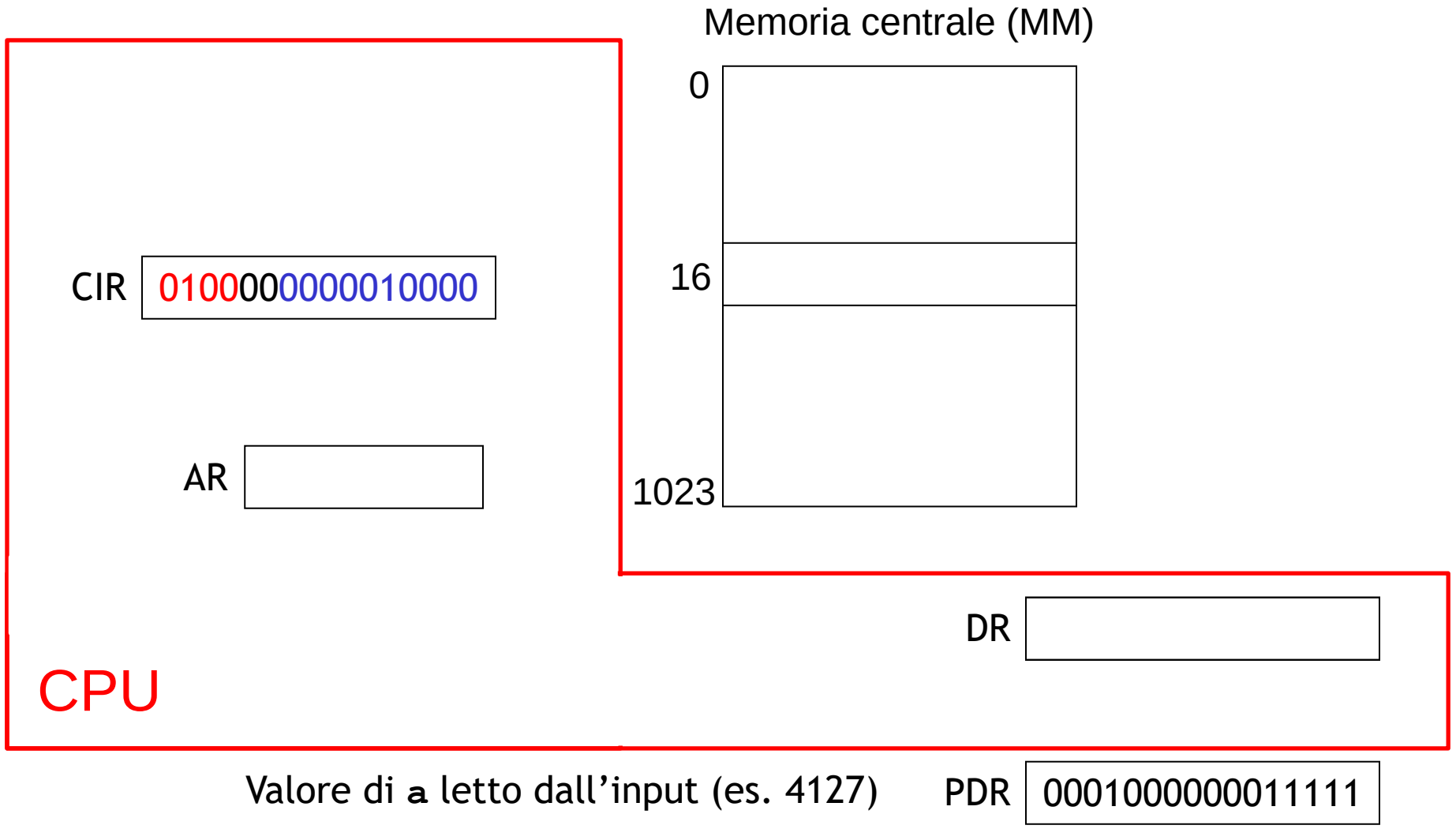


Fase di Decodifica 1ª istruzione

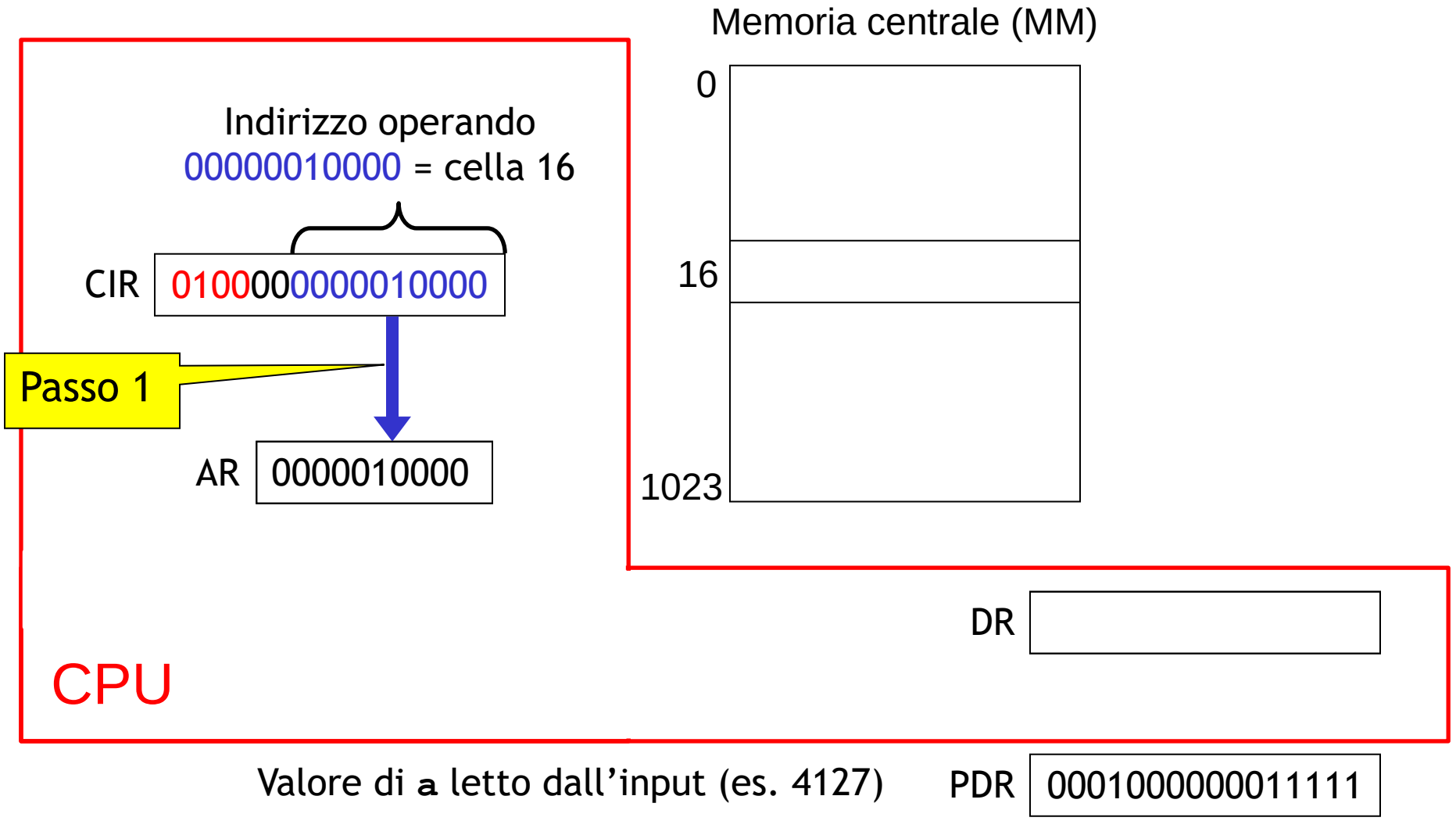
CIR 

Codice operativo 0100 = leggi da input e salva il dato letto all'indirizzo specificato come operando (000000010000)

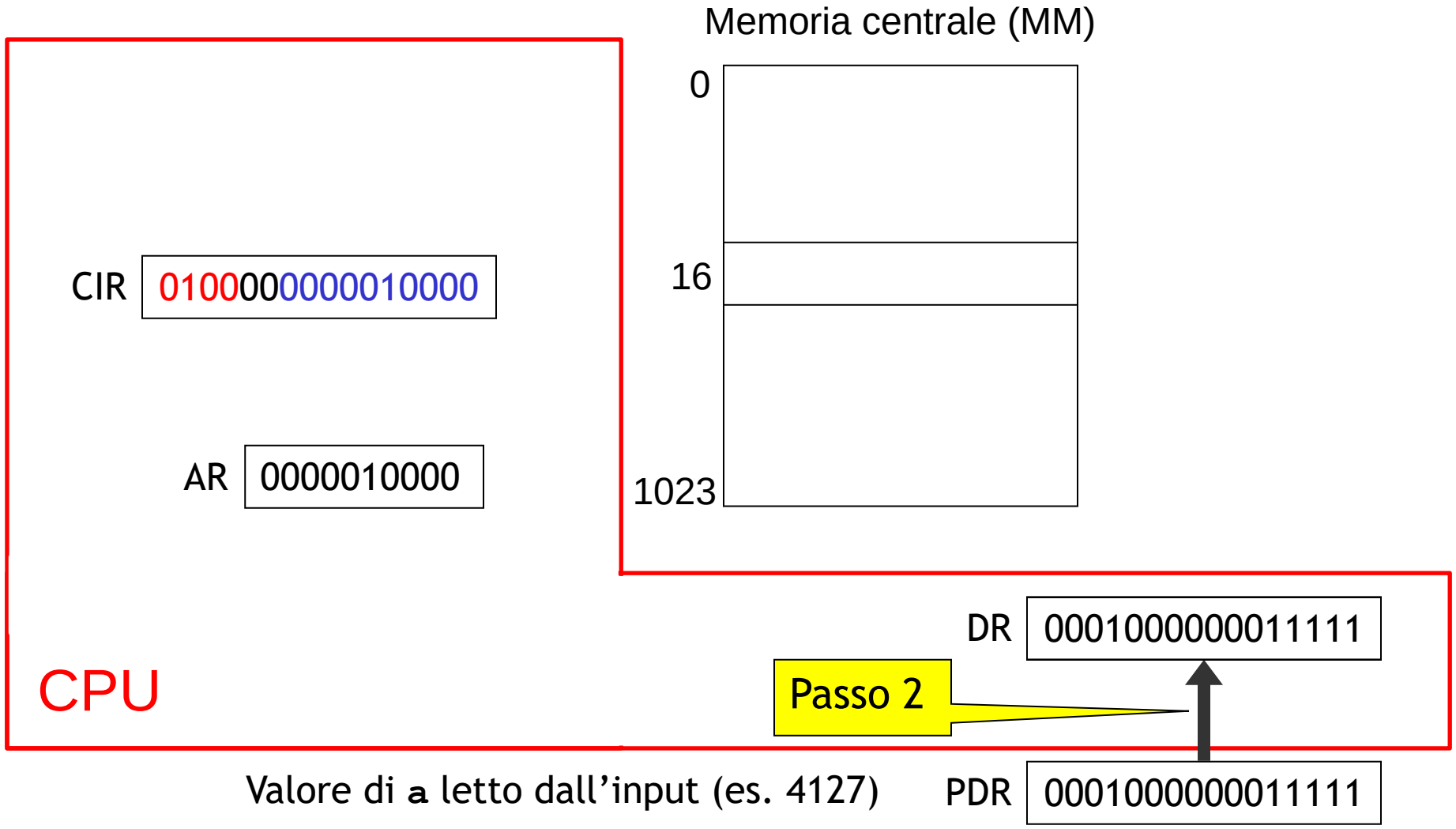
Fase di esecuzione: esempio lettura da input



Fase di esecuzione: esempio lettura da input



Fase di esecuzione: esempio lettura da input



Fase di esecuzione: esempio lettura da input