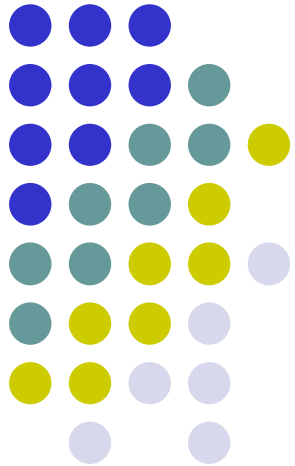
Laboratorio di Calcolatori 1

Corso di Laurea in Fisica

A.A. 2006/2007



Dott.Davide Di Ruscio

Dipartimento di Informatica

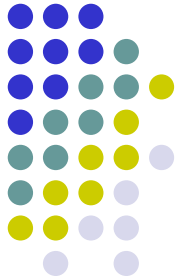
Università degli Studi di L'Aquila

Lezione del 23/01/07

Nota

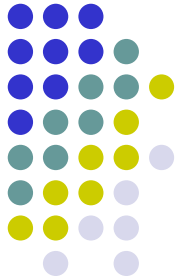
Questi lucidi sono tratti dal materiale distribuito dalla McGraw-Hill e basati su del materiale fornito dal Prof. Flammini Michele

Sommario (I parte)



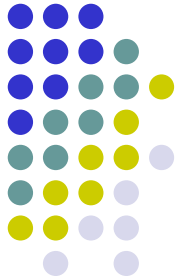
- Concetti fondamentali
- Aspetti architetturali di un sistema di calcolo
  - hardware
  - software
    - software di base
    - software applicativo
- Codifica dell'informazione
  - numeri naturali, interi, reali
  - caratteri
  - immagini
- Macchina di Von Neumann
  - CPU (UC, ALU, registri, clock)
  - memoria centrale
  - bus di bistema
  - periferiche
- Linguaggio macchina
- Linguaggio assembler
- Sistema operativo
- Ambiente di programmazione

Codifica immagini



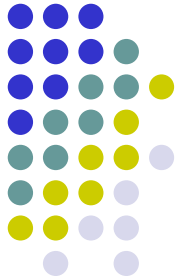
- Anche le immagini vengono codificate come una sequenza di bit
- Il passaggio da un'immagine a una sequenza binaria prende il nome di **digitalizzazione**
- Un'immagine viene vista come un insieme di punti (**pixel**) giacenti su una griglia bidimensionale
- Ad ogni punto viene associata un insieme di bit che ne codificano il colore o la tonalità di grigio (per bianco e nero)
- **Risoluzione**: numero di punti rappresentati per unità di superficie (punti per pollice quadrato o *dot per inch*)
- Grazie all'alta località, ossia al fatto che punti vicini tendono ad essere molto simili (basti pensare allo sfondo), è possibile utilizzare rappresentazioni o formati compressi in grado di ridurre il numero totale di bit utilizzati
- Le tecniche di compressione si distinguono in **tecniche lossless** e **tecniche lossly**
- Tra i formati più noti: TIFF, GIF, JPEG, …

Osservazioni codifica immagini



- Come i numeri reali, le immagini hanno uno spettro continuo, mentre le codifiche uno spettro discreto (come le approssimazioni razionali dei reali)
- Quindi nella digitalizzazione, ossia nella codifica di immagini con sequenze di bit (di lunghezza finita), si commette necessariamente un errore di rappresentazione
- All'aumentare della risoluzione e del numero di colori rappresentati, tale errore diventa impercettibile
- Al contrario, eventuali errori dovuti a trasmissioni, deterioramento del supporto di memorizzazione, malfunzionamenti hardware, ..., rispetto alle precedenti codifiche analogiche entro certi limiti di rumore sono recuperabili perché
  1. è più facile distinguere tra stati discreti (es. bit pari a 0 o ad 1) che tra stati continui (es. tra due reali)
  2. è possibile utilizzare codici a correzioni di errore capaci di correggere un numero limitato di bit comunque alterati

Codifiche analogiche e digitali

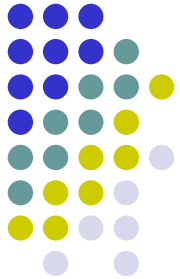


- La seguente tabella riassume le differenze fondamentali tra codifiche analogiche e digitali:

	Codifiche analogiche	Codifiche digitali
Errore di rappresentazione	Teoricamente nullo	Ammesso ma impercettibile
Separazione codifica originale - rumore	Generalmente impossibile	Possibile entro certi limiti di rumore

- Tali differenze sono direttamente riscontrabili confrontando tra loro TV terrestre e TV satellitare (digitale), dischi in vinile - cassette audio (analogici) e CD audio (digitali), foto su pellicola e foto digitali, ….

Il Linguaggio del Calcolatore



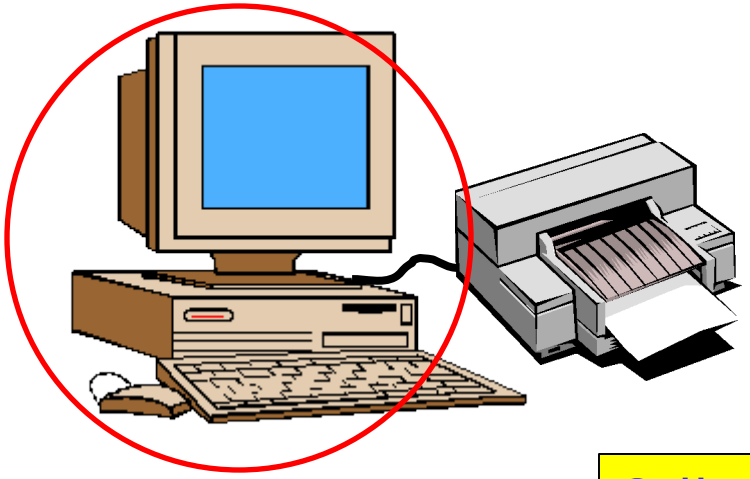
Introduciamo ora un esempio di linguaggio del calcolatore o "*linguaggio macchina*" semplificato in grado di essere eseguito direttamente su un'architettura hardware semplificata.

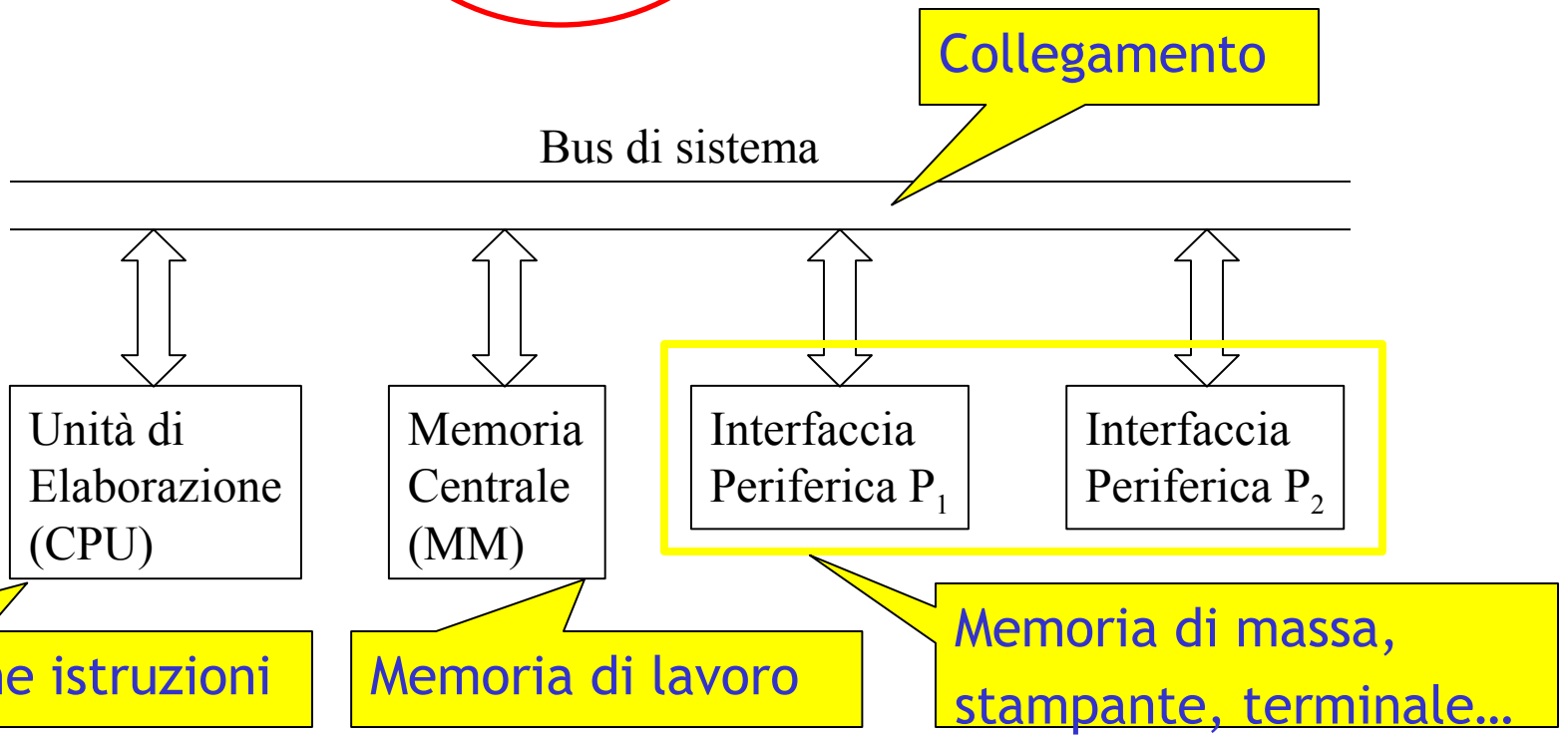
In particolare vedremo:

- un'architettura hardware semplificata
- il formato delle istruzioni
- l'esecuzione delle istruzioni
- le istruzioni principali

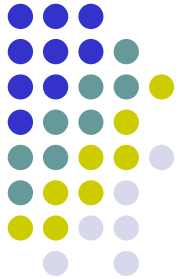
[Per il linguaggio macchina e le relative istruzioni fare riferimento al Cap. 4 (fino alla Sez. 4.5 inclusa) del libro *Istituzioni di Informatica, Linguaggio di riferimento Pascal* di S. Ceri, D. Mandrioli, L. Sbattella, Casa editrice McGraw-Hill, 1993 oppure *Informatica: istituzioni, Linguaggio di riferimento ANSI C* degli stessi autori e casa editrice.]

Architettura hardware semplificata



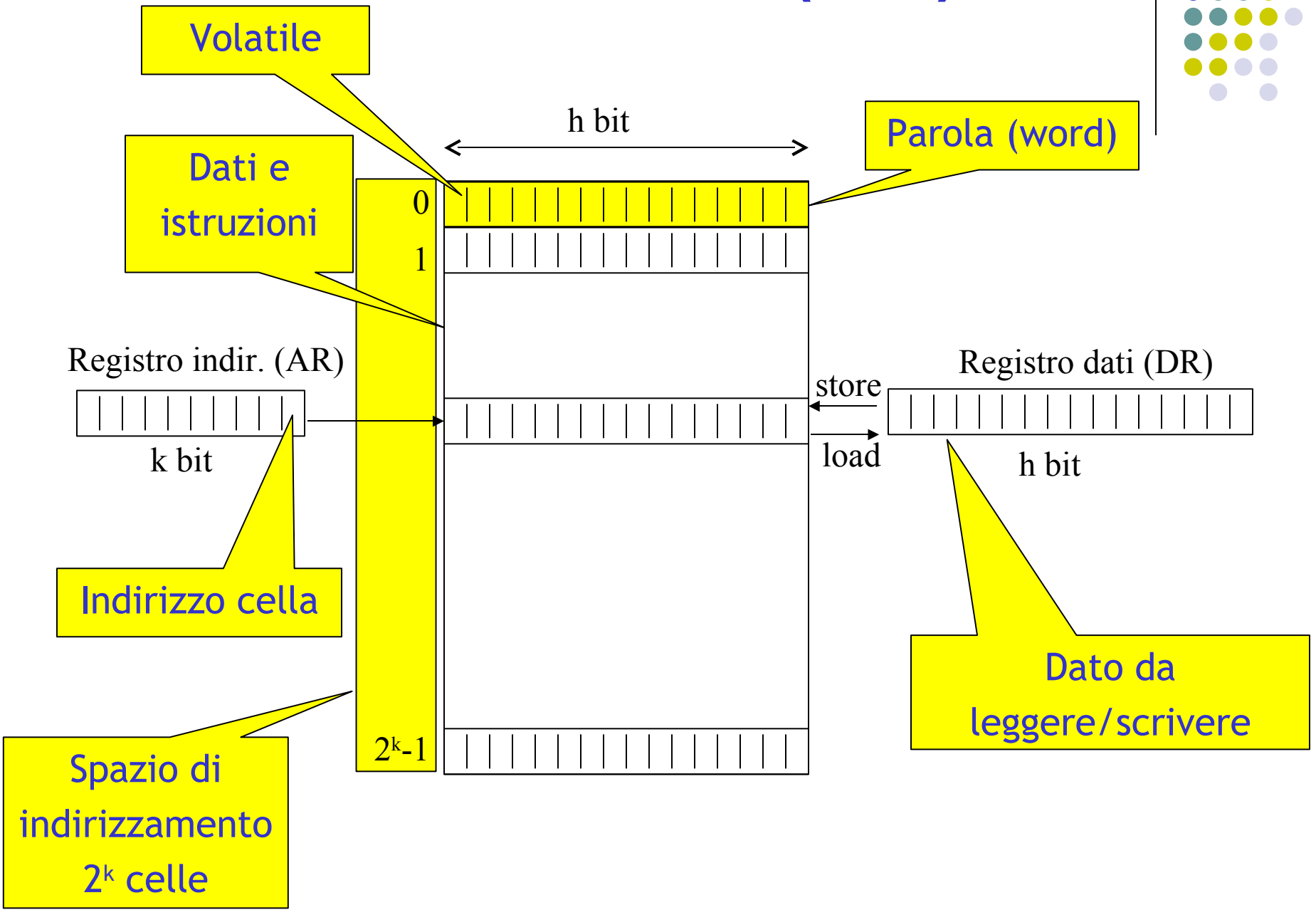


La memoria centrale (MM)

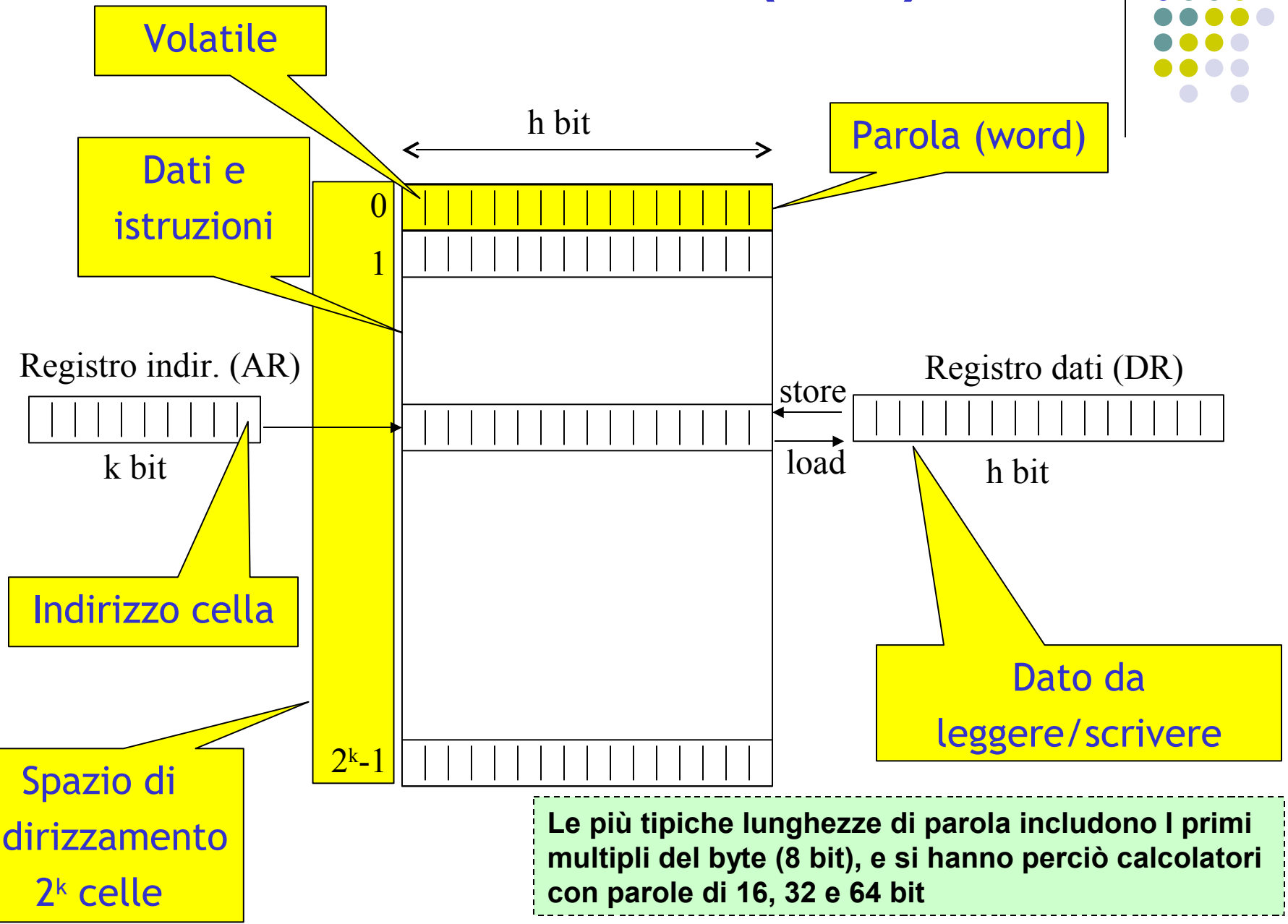


- Comunemente nota anche come RAM (Random Access Memory – lettura e scrittura) e spesso accompagnata da ROM (Read Only Memory – solo lettura)
- In essa transitano le istruzioni (in linguaggio macchina) che devono essere eseguite ed i dati su cui operano
- Può essere vista come una tabella in cui ogni entrata o cella è identificata da un indirizzo (indice) e può memorizzare una parola o word di memoria di lunghezza fissata
- Ha la caratteristica di essere
  - volatile: il suo contenuto viene perso quando viene spento l'elaboratore
  - veloce (ordine dei nanosecondi, ossia 10^{-9} secondi)
  - costosa
  - di dimensioni medio-piccole, tipicamente centinaia di megabyte (es. 512Mb) o qualche gigabyte

La memoria centrale (MM)

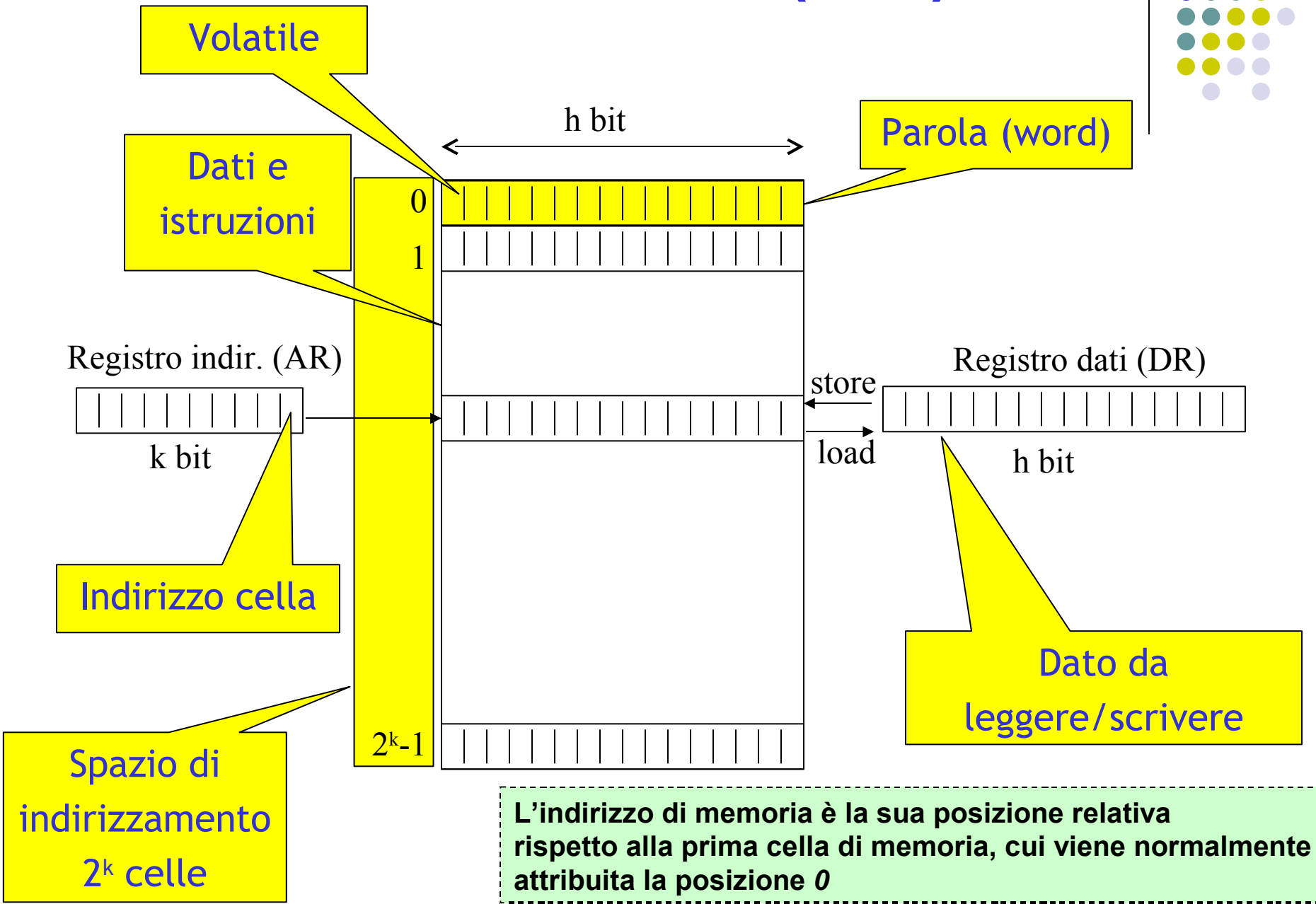


La memoria centrale (MM)



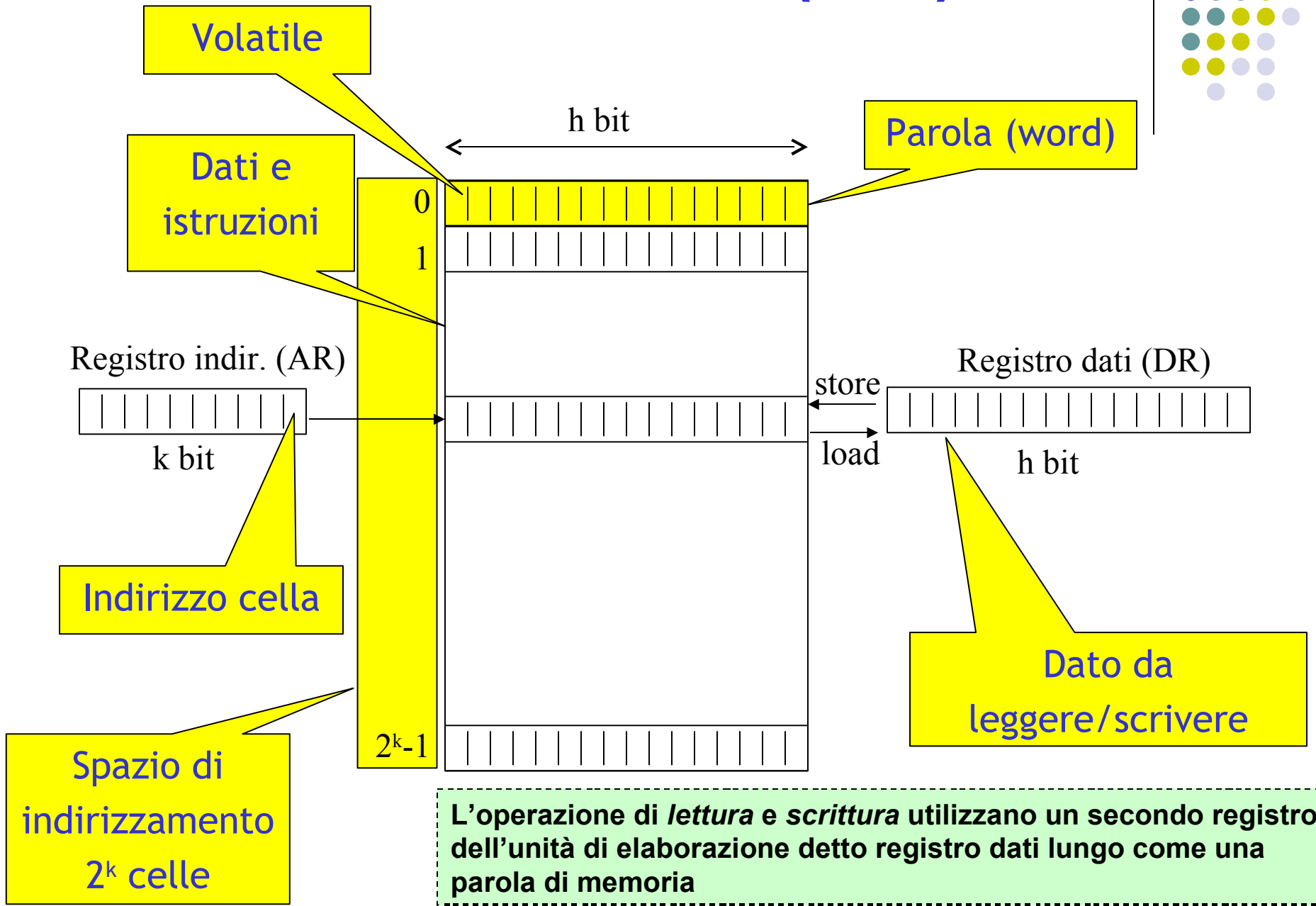
Le più tipiche lunghezze di parola includono I primi multipli del byte (8 bit), e si hanno perciò calcolatori con parole di 16, 32 e 64 bit

La memoria centrale (MM)



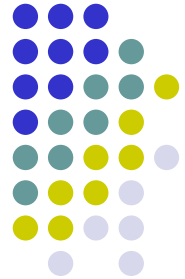
L'indirizzo di memoria è la sua posizione relativa rispetto alla prima cella di memoria, cui viene normalmente attribuita la posizione *0*

La memoria centrale (MM)



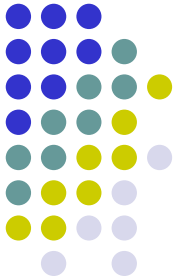
L'operazione di *lettura* e *scrittura* utilizzano un secondo registro dell'unità di elaborazione detto registro dati lungo come una parola di memoria

L’unità di elaborazione (CPU)



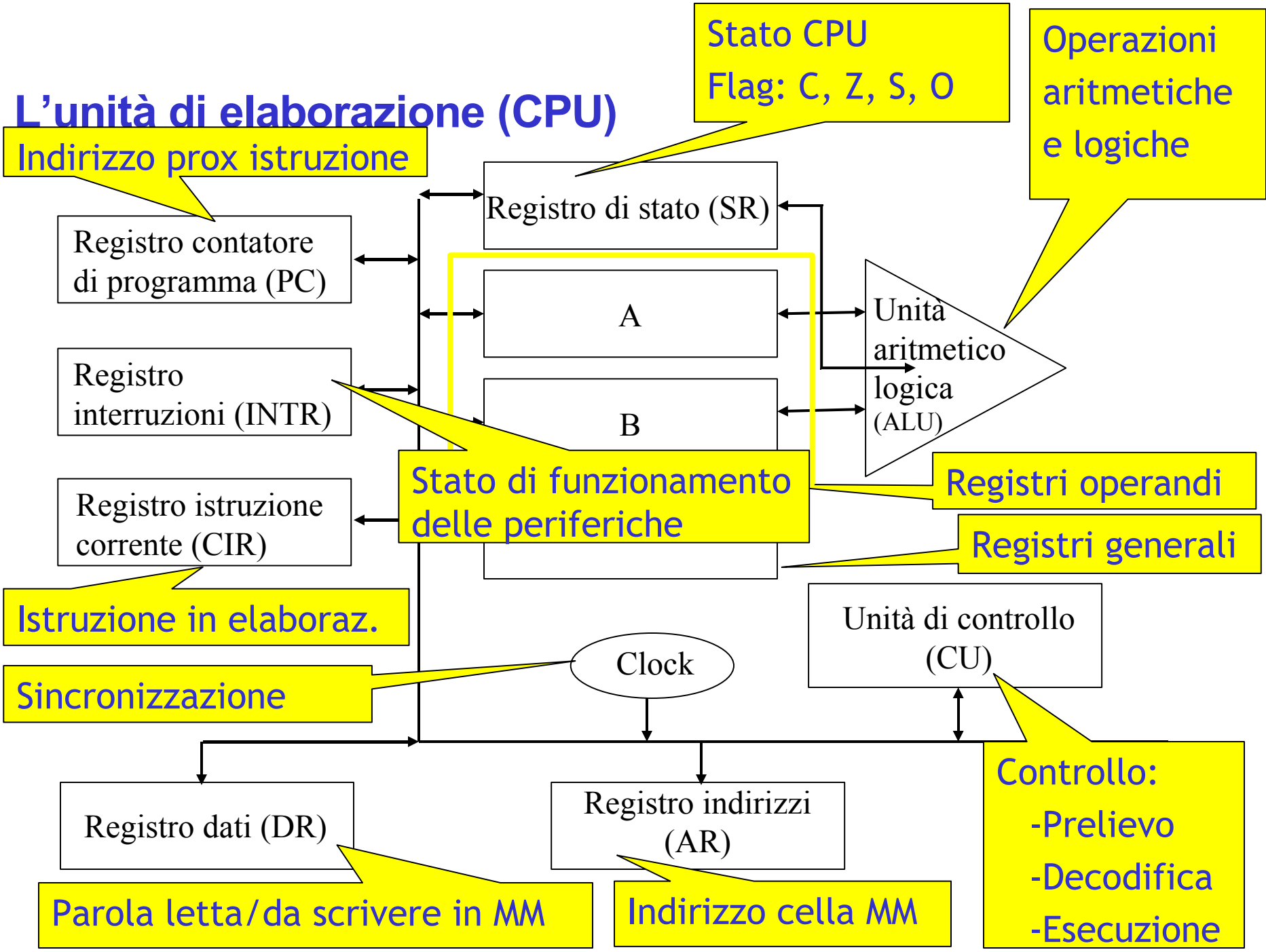
- L’unità di controllo ha il compito di *coordinare* l’intero sistema: *interpretando* le istruzioni del programma in esecuzione essa ne controlla *l’esecuzione* nella giusta sequenza

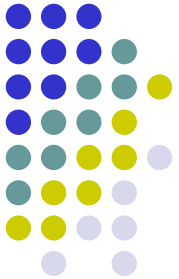
- Sotto la sua direzione le informazioni vengono *trasferite* o *elaborate* tramite la ALU internamente alla CPU, lette e scritte in memoria, scambiate con il mondo esterno



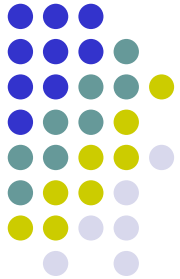
- Al suo interno è composta da:
  - C.U. (Unità di Controllo) : è responsabile del prelievo e della decodifica delle istruzioni nonché dell'invio dei segnali di controllo che provocano i trasferimenti o le elaborazioni necessarie per l'esecuzione dell'istruzione decodificata
  - A.L.U. (Unità Logico-Aritmetica): svolge le operazioni logiche ed aritmetiche eventualmente richieste per l'esecuzione delle istruzioni
  - CLOCK:
    - scandisce gli intervalli di tempo in cui agiscono in modo sincrono i dispositivi interni alla C.P.U.
    - determina la velocità della C.P.U., espressa come frequenza o numero di intervalli scanditi nell'unità di tempo (es., 512MHz, 1GHz, …)

L'unità di elaborazione (CPU)





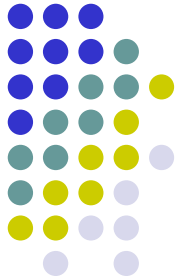
- REGISTRI: dispositivi elettronici capaci di memorizzare sequenze di bit fungendo da piccole memorie interne alla C.P.U.
  - **DR** (registro dati): in esso transitano le parole da leggere (load) o scrivere (store) nella memoria centrale
  - **AR** (registro indirizzi): contiene l'indirizzo della cella di memoria coinvolta nell'operazione di load o store
  - **CIR** (registro istruzione corrente): contiene l'istruzione correntemente in esecuzione
  - **PC** (registro contatore di programma): contiene l'indirizzo della cella di memoria in cui si trova la prossima istruzione da eseguire
  - **INTR** (registro interruzioni): per la gestione di eventi asincroni, contiene istruzioni circa lo stato di funzionamento delle periferiche



- **A,B**: memorizzano gli operandi e successivamente i risultati delle operazioni svolte dall'ALU. In particolare,
  - in precedenza essi vengono caricati con i valori dei due operandi
  - dopo l'esecuzione di un'operazione il registro A è caricato con il risultato dell'operazione.
  - nel caso della divisione intera, il resto della divisione è caricato nel registro B, mentre per le altre operazioni il contenuto del registro B non è definito
- **SR** (registro di stato): contiene informazioni circa l'esito delle operazioni svolte dall'ALU, ed in particolare i bit di carry (presenza di riporto), zero (presenza di valore nullo in A), segno (segno del risultato del'operazione svolta dall'ALU), overflow (per rilevare eventuali condizioni di overflow)
- …

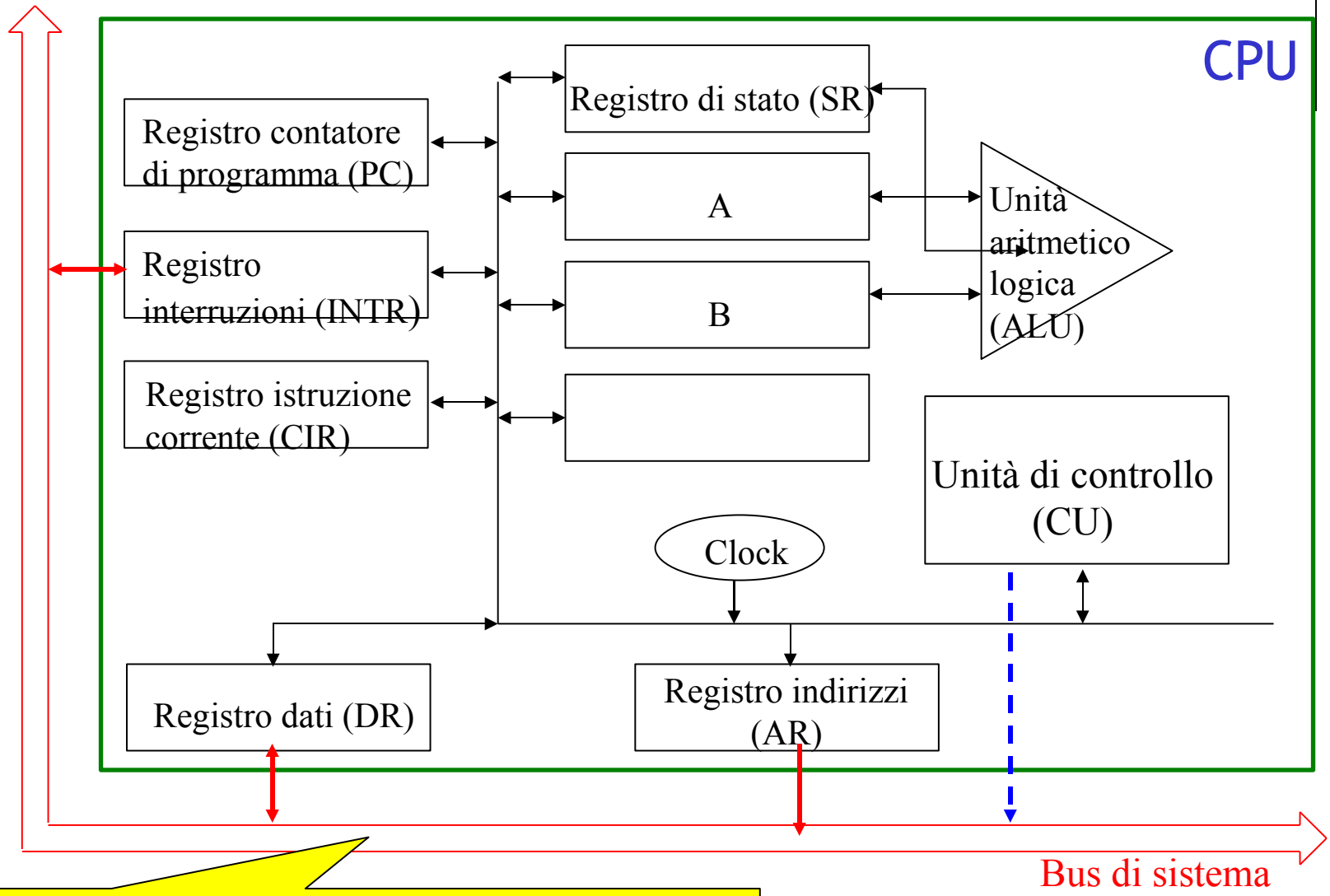
…

Il bus di sistema

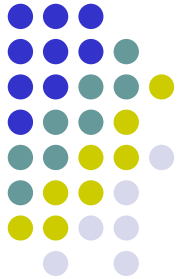


- Costituito da un insieme di connessioni elementari lungo le quali viene trasferita l'informazione
- Consente la comunicazione tra le varie componenti
- In ogni istante di tempo, il bus è dedicato a collegare due unità funzionali (una trasmette, l'altra riceve)
- Le possibili interconnessioni sono tra l'unità di elaborazione e la memoria, oppure tra l'unità di elaborazione e l'interfaccia di una specifica periferica
- E' in genere sotto il controllo dell'unità di elaborazione, che seleziona l'interconnessione da attivare e indica l'operazione da compiere
  - In tal caso l'unità di elaborazione esercita il ruolo di **master**
  - Le altre unità funzionali assumono il ruolo di **slave**

Il bus di sistema



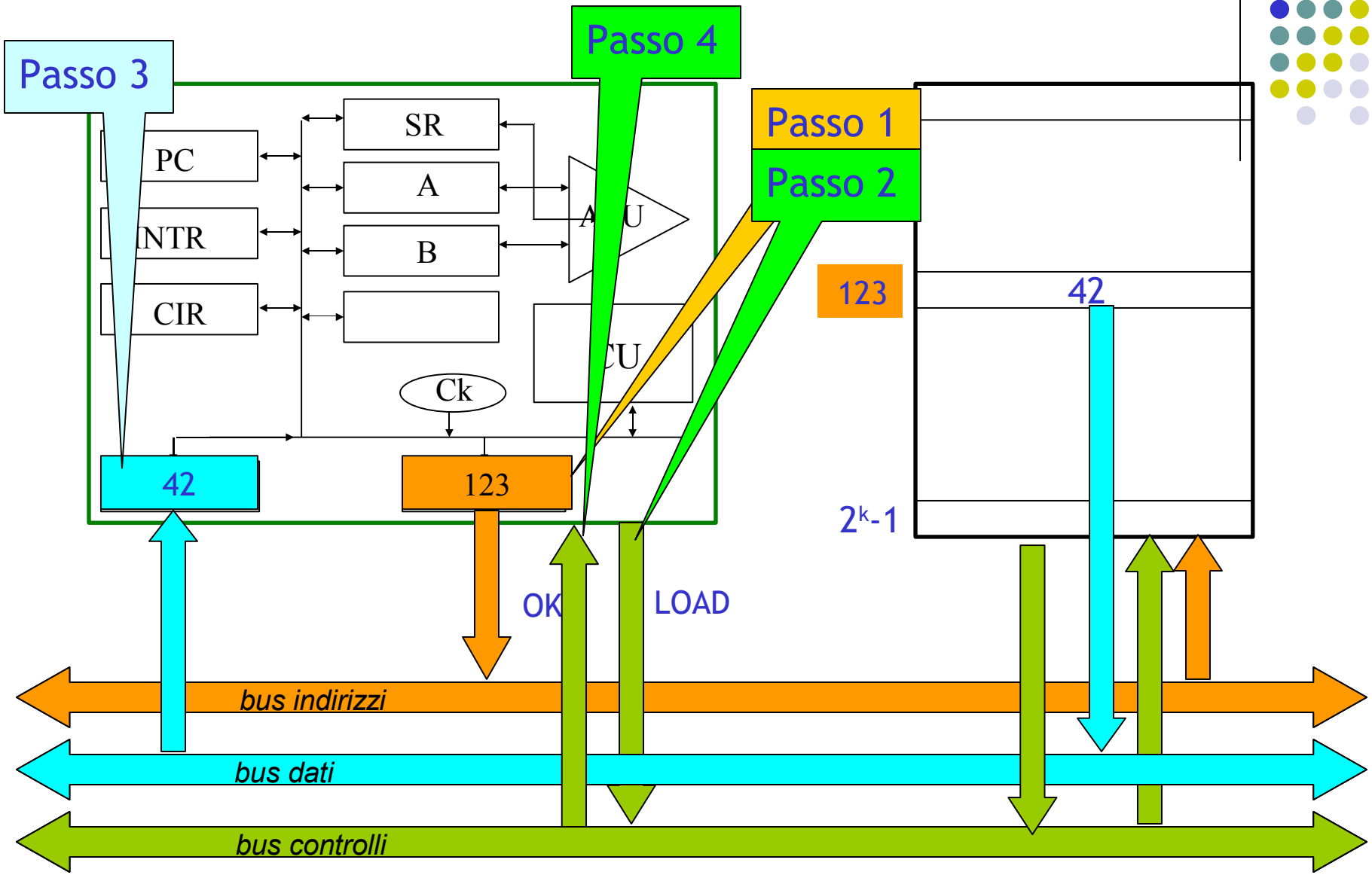
Bus dati, Bus indirizzi, Bus controlli



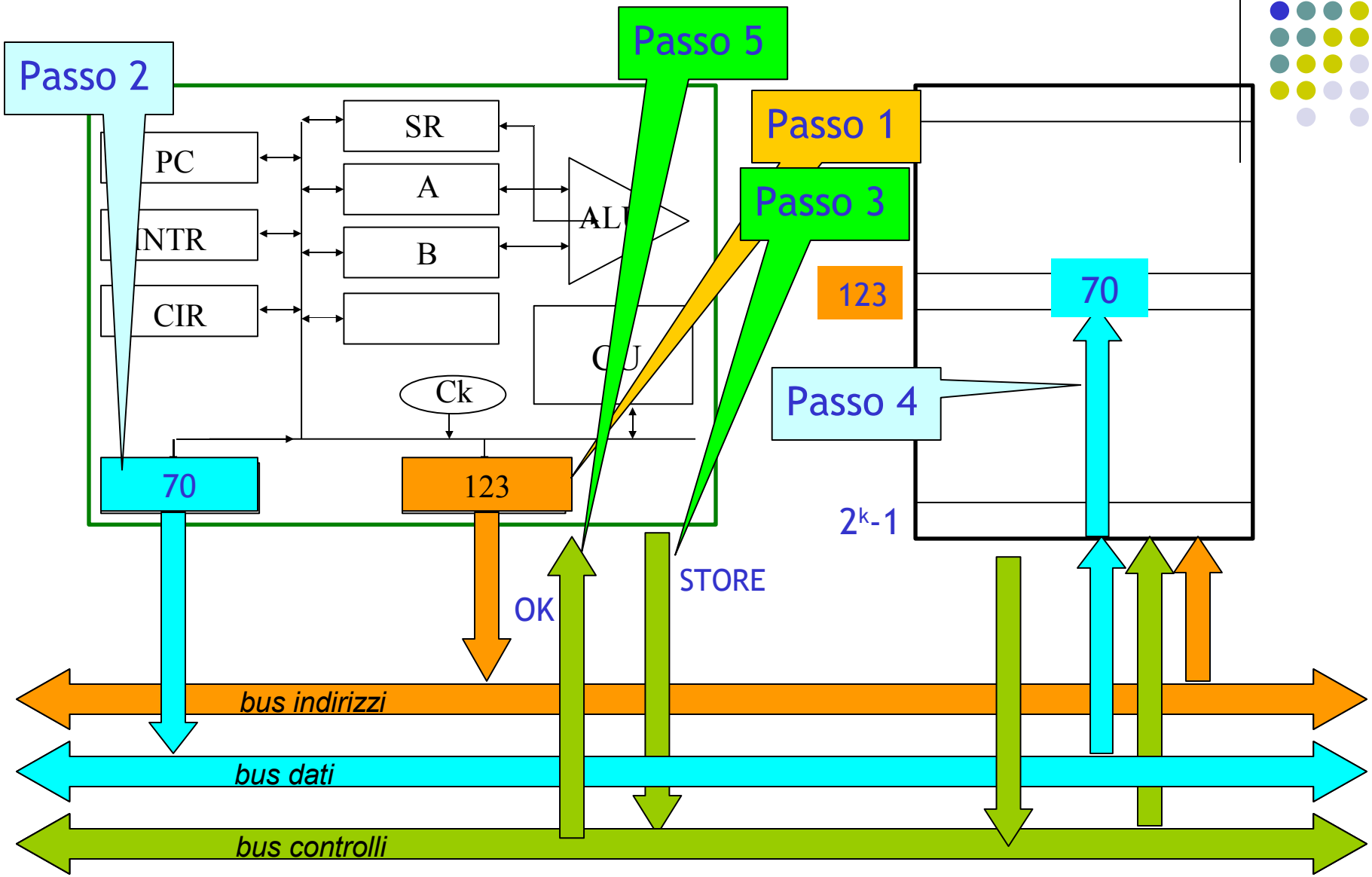
- Le linee del bus vengono specificate funzionalmente e suddivise in tre categorie a seconda del tipo di informazione trasportata:
  - Bus dati: per la trasmissione di dati dall'unità master all'unità slave o viceversa
  - Bus indirizzi: per la trasmissione di indirizzi di memoria centrale
  - Bus di controllo: per la trasmissione di comandi alle varie unità e di informazioni di controllo. In particolare, trasferisce dall'unità master all'unità slave un codice corrispondente all'istruzione da eseguire e dall'unità slave all'unità master informazioni relative all'avvenuto completamento dell'operazione richiesta

NB: La presenza di molte linee in un bus consente di trasferire dati in "parallelo"

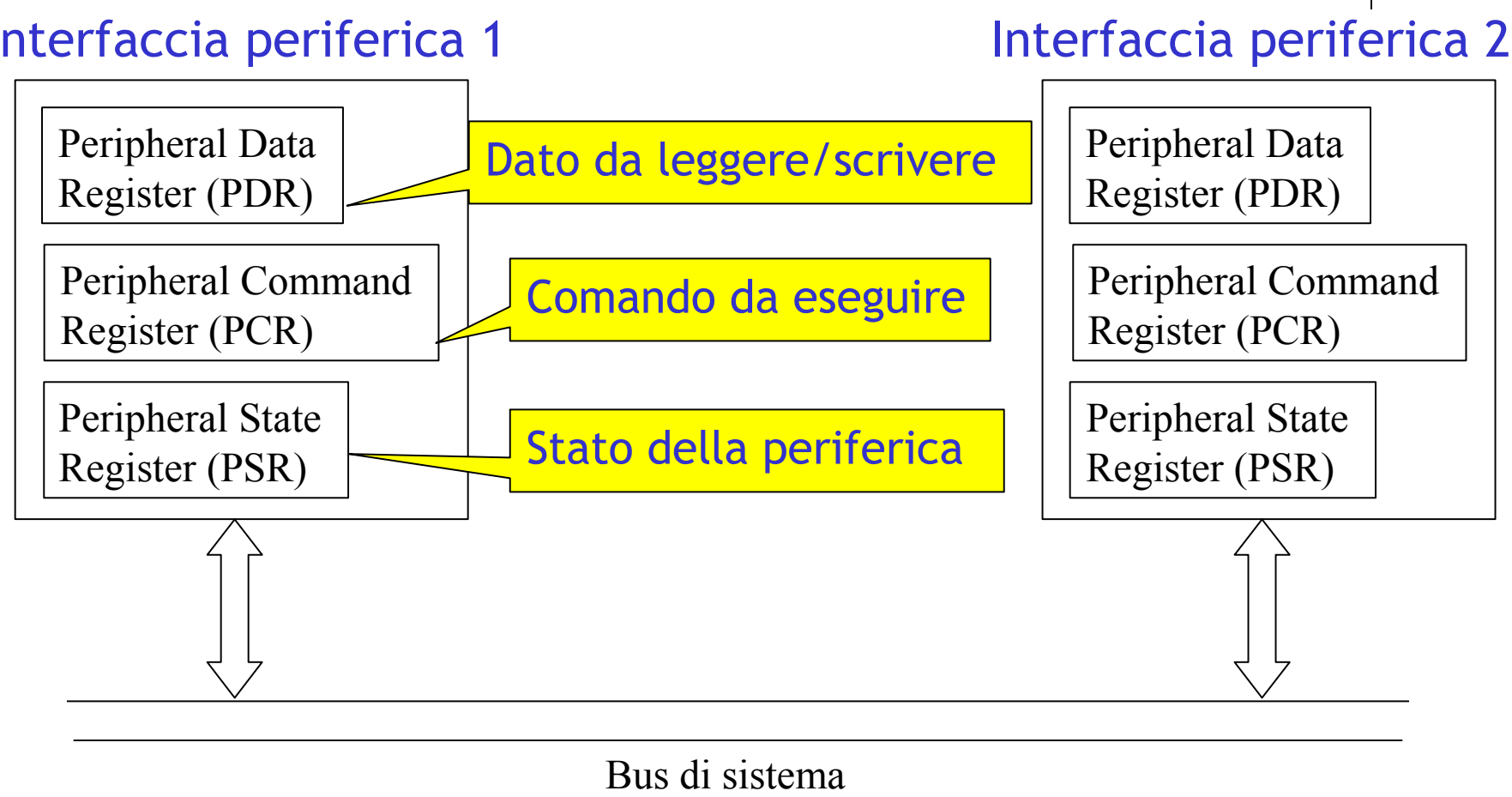
Sequenza di lettura in MM (load)



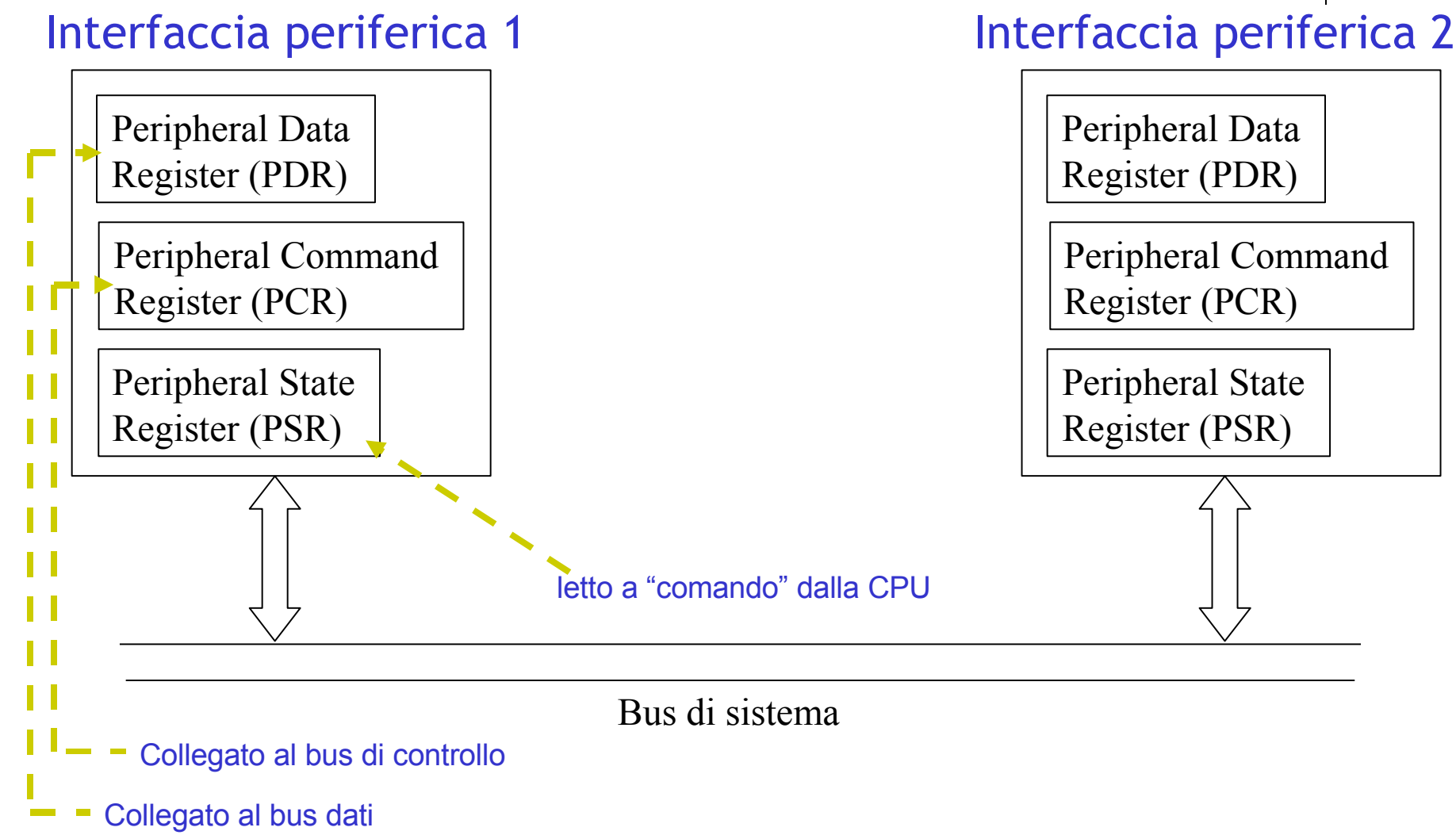
Sequenza di scrittura in MM (store)



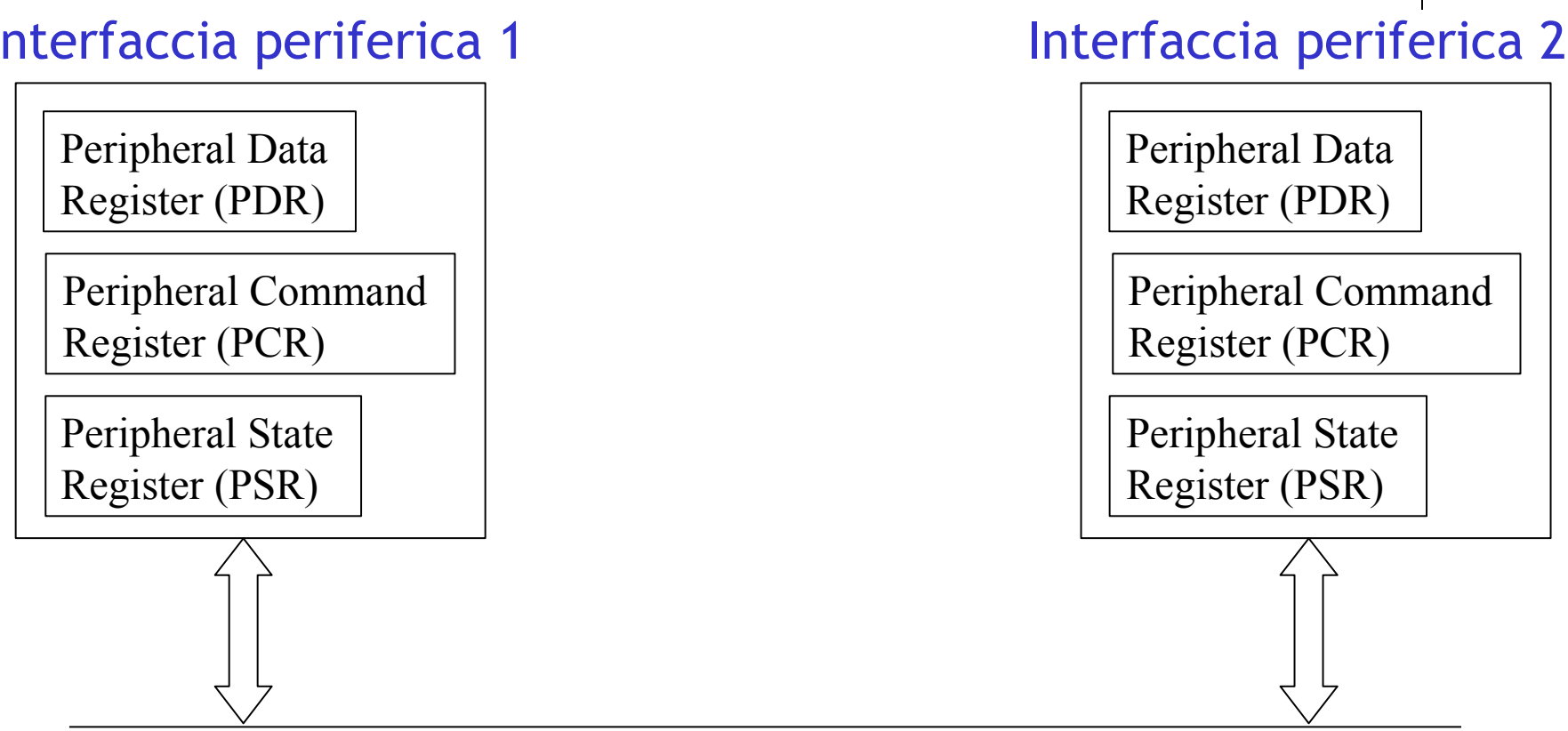
Le interfacce delle periferiche



Le interfacce delle periferiche

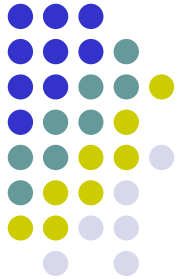


Le interfacce delle periferiche



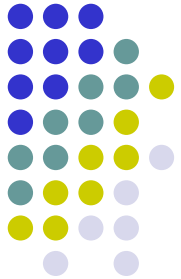
Es. Un terminale ha due registri dati distinti, uno che consente all'elaboratore di acquisire dati dalla tastiera, e uno che consente di produrre dati da visualizzare a video

Periferiche: memoria di massa



- Anche se è una componente fondamentale, non fa parte della macchina di Von Neumann in senso stretto
- È costituita dai dischi rigidi, nastri, CD e DVD ROM, …
- Rispetto alla memoria centrale ha la caratteristica di essere
  - non volatile
  - lenta (per hard disk ordine dei millisecondi, ossia 10^{-3} secondi)
  - economica
  - di grandi dimensioni (per hard disk centinaia di gigabyte)

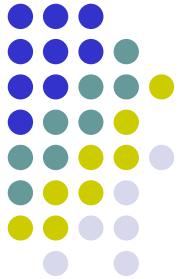
Periferiche: unità di Input/Output



- Consentono la comunicazione dell'elaboratore con l'esterno ed in particolare la lettura di dati in input e la restituzione dei risultati delle elaborazioni in output.

- Ne fanno parte terminali (tastiera e schermo), mouse, stampanti, scanner, …

Il formato delle istruzioni



Le istruzioni del linguaggio sono costituite da sequenze di bit e si dividono in due parti:

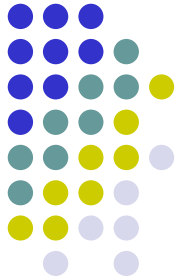
- *codice operativo*

  è sempre presente e serve per specificare l'operazione da compiere

- parte *operandi*

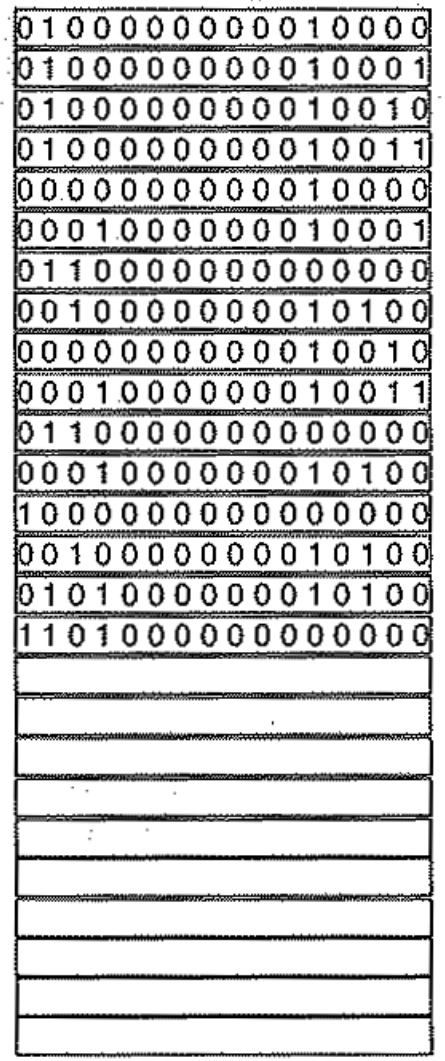
  serve per specificare una o più locazioni di memoria a cui l'istruzione fa riferimento; in alcune istruzioni non devo essere specificati e questa parte viene ignorata

Esempio Programma in binario



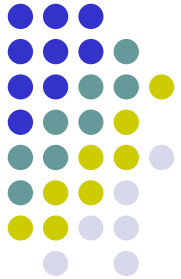
0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0	leggi un valore in ingresso e ponilo nella cella numero 16 (variabile *a*)
0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1	leggi un valore e ponilo nella cella numero 17 (variabile *b*)
0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0	leggi un valore e ponilo nella cella numero 18 (variabile *c*)
0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 1	leggi un valore e ponilo nella cella numero 19 (variabile *d*)
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0	carica il registro A con il contenuto della cella 16 (valore di *a*)
0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1	carica il registro B con il contenuto della cella 17 (valore di *b*)
0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0	somma i contenuti dei registri A e B
0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0	copia il contenuto del registro A nella cella numero 20 (risultato parziale)
0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0	carica il registro A con il contenuto della cella 18 (valore di *c*)
0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 1	carica il registro B con il contenuto della cella 19 (valore di *d*)
0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0	somma i contenuti dei registri A e B
0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0	carica il registro B con il contenuto della cella 20 (risultato parziale)
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	moltiplica i contenuti dei registri A e B
0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0	copia il contenuto del registro A nella cella numero 20 (risultato)
0 1 0 1 0 0 0 0 0 0 0 1 0 1 0 0	scrivi in output il contenuto della cella numero 20 (risultato)
1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0	arresta l'esecuzione del programma
.....	spazio per la variabile *a* (cella 16)
.....	spazio per la variabile *b* (cella 17)
.....	spazio per la variabile *c* (cella 18)
.....	spazio per la variabile *d* (cella 19)
.....	spazio per il risultato (cella 20)

Esempio Programma in MM



cella numero 0
cella numero 1
cella numero 2
cella numero 3
cella numero 4
cella numero 5
cella numero 6
cella numero 7
cella numero 8
cella numero 9
cella numero 10
cella numero 11
cella numero 12
cella numero 13
cella numero 14
cella numero 15
cella numero 16 riservata per la variabile *a*
cella numero 17 riservata per la variabile *b*
cella numero 18 riservata per la variabile *c*
cella numero 19 riservata per la variabile *d*
cella numero 20 riservata per il risultato

celle di memoria libere



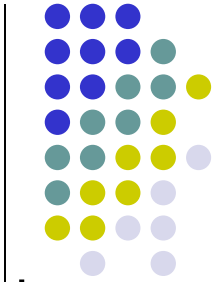
Nel nostro linguaggio macchina semplificato tutte le istruzioni hanno la stessa lunghezza e possono avere al più un solo operando.

Poniamo quindi:

- *s* = lunghezza di una istruzione
- *m* = numero di bit dedicati al codice operativo
- *n* = numero di bit dedicati all'eventuale operando

Quindi $s=m+n$

Set istruzioni e dimensionamento macchina hardware



Set istruzioni: insieme di tutte le istruzioni del linguaggio macchina

Quante sono?

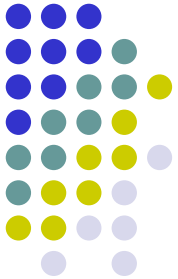
Poiché ognuna è identificata da un codice operativo, sono al più 2^m

Viceversa, se abbiamo p istruzioni, allora m $m \geq \lceil \log_2 p \rceil$

Inoltre, se la memoria centrale ha 2^k locazioni, per poterle indirizzare tutte deve essere $n \geq k$

Infine, se s e la lunghezza (in termini di bit) h delle parole della memoria centrale coincidono, ogni istruzione sarà contenuta in esattamente una cella di memoria; nel seguito assumeremo sempre vera questa ipotesi.

Tabella o elenco delle istruzioni

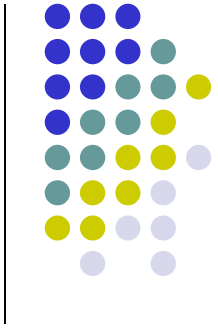


LOADA	0000
LOADB	0001
STOREA	0010
STOREB	0011
READ	0100
WRITE	0101
ADD	0110
DIF	0111
MUL	1000
DIV	1001
JUMP	1010
JUMPZ	1011
NOP	1100
HALT	1101

Codice simbolico o mnemonico

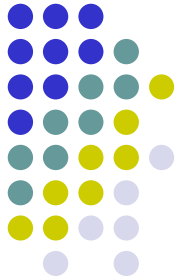
Codice operativo

Tabella o elenco delle istruzioni



lettura in memoria	LOADA	0000
	LOADB	0001
scrittura in memoria	STOREA	0010
	STOREB	0011
lettura dalla periferica	READ	0100
scrittura sulla periferica	WRITE	0101
operazioni aritmetiche	ADD	0110
	DIF	0111
	MUL	1000
	DIV	1001
salto a branch	JUMP	1010
	JUMPZ	1011
nessuna operazione	NOP	1100
terminazione	HALT	1101

Dato che il linguaggio comprende *14* istruzioni diverse, per il codice operativo sono necessari $m \geq 4$ bit

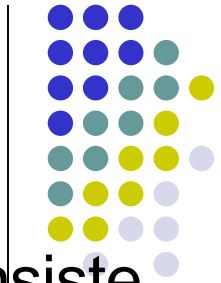


Per completare il dimensionamento della macchina hardware fissiamo $s=h=16$, in modo che la lunghezza delle parole di memoria (word) sia un multiplo di 8

Quanto è grande la memoria centrale?

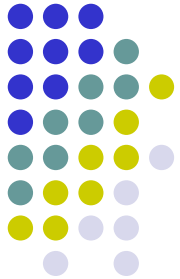
La dimensione è di $h2^k=2^4 2^{12}=64K$bit, o analogamente $h2^k h/2^3=2^4 2^{12}/2^3=8K$byte

L'esecuzione delle istruzioni



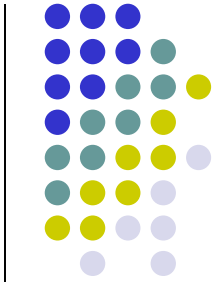
- Il normale funzionamento dell'elaboratore consiste nell'esecuzione in sequenza delle istruzioni dei programmi in linguaggio macchina
- L'esecuzione di una singola istruzione, prevede due fasi fondamentali:
  - Fase di Fetch: acquisizione dell'istruzione da eseguire dalla memoria centrale
  - Fase di Execute: interpretazione ed esecuzione dell'istruzione stessa
- Ogni fase prevede l'esecuzione di alcune micro-istruzioni
- Micro-istruzione: operazione elementare che consiste nel trasferimento di dati tra registri o tra registri e la memoria centrale e in operazioni aritmetico-logiche
- Tutte le istruzioni hanno una fase di fetch identica e si distinguono solo per la fase di execute

Micro-istruzioni fase di fetch



- Il contenuto del registro *PC* viene trasferito nel registro *AR*
- Il contenuto della cella di memoria il cui indirizzo è contenuto in *AR* viene trasferito in *DR*
- Il contenuto di *DR* viene trasferito in *CIR*
- Il valore di *PC* viene incrementato di uno per predisporre la fase di fetch della prossima istruzione. Infatti, così facendo il registro *PC* contiene l'indirizzo della successiva istruzione da eseguire, che si trova nella locazione seguente rispetto a quella corrente, ossia attualmente caricata in *CIR*.

Fase di fetch

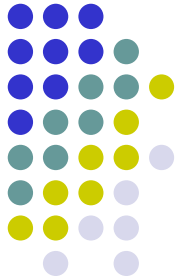


In sintesi, la fase di fetch viene realizzata dalle seguenti quattro micro-istruzioni riportate in una forma più semplice e compatta:

- PC→AR
- MEM[AR]→DR
- DR→CIR
- PC+1→PC

- Questa notazione implica un trasferimento dati dall'elemento a sinistra della freccia verso l'elemento di destra.
- MEM[AR] indica il contenuto della locazione di memoria il cui indirizzo è specificato in AR

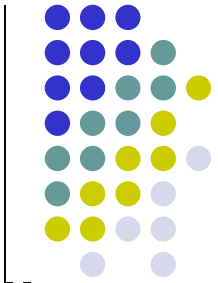
L'istruzione *loada*



- *loada ind*: carica il contenuto della cella di memoria il cui indirizzo è specificato nell'operando, ossia *ind*, nel registro A

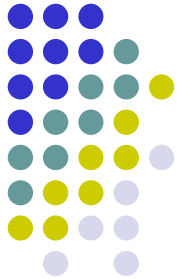
- Viene eseguita dalle seguenti micro-istruzioni:

  - L'operando dell'istruzione corrente (ossia in CIR) OP(CIR) viene copiato in AR
  - Il contenuto della cella di memoria corrispondente all'indirizzo in AR, ossia MEM[AR], viene copiato nel DR
  - Il contenuto di DR viene copiato nel registro A



- In sintesi comporta quindi l'esecuzione delle seguenti tre microistruzioni:
  - OP(CIR)→AR
  - MEM[AR] →DR
  - DR→A
- N.B.: OP(CIR) indica il contenuto degli ultimi *n* bit del registro CIR

L'istruzione *loadb*

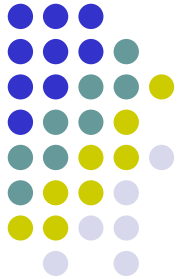


- *loadb ind*: carica il contenuto della cella di memoria di indirizzo *ind* nel registro B

- Viene eseguita dalle seguenti micro-istruzioni:

  - OP(CIR)→AR
  - MEM[AR] →DR
  - DR→B

L'istruzione *storea*

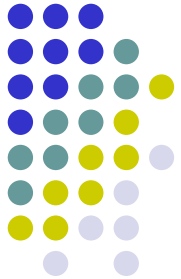


- *storea ind*: carica il contenuto del registro A nella cella di memoria di indirizzo *ind*

- Viene eseguita dalle seguenti micro-ostruzioni:

  - A→DR
  - OP(CIR)→AR
  - DR→MEM[AR]

L'istruzione *storeb*

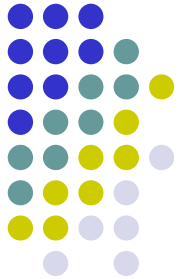


- *storeb ind*: carica il contenuto del registro B nella cella di memoria di indirizzo *ind*

- Viene eseguita dalle seguenti micro-istruzioni:

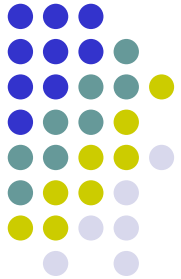
  - B→DR
  - OP(CIR)→AR
  - DR→MEM[AR]

L'istruzione *read*



- *read ind*: acquisisce un valore dalla periferica di input (ad es. una tastiera) e lo inserisce nella cella di memoria di indirizzo *ind*

- Viene eseguita dalle seguenti micro-istruzioni:
  - Il contenuto del registro PDR dell'interfaccia della periferica viene trasferito tramite il bus in DR
  - L'operando in OP(CIR) viene trasferito in AR
  - Viene eseguita una scrittura in memoria centrale, ossia Il contenuto di DR viene memorizzato nella cella di memoria il cui indirizzo è specificato in AR, ossia in MEM[AR]



In sintesi, viene eseguita dalle seguenti tre micro-istruzioni :

- PDR→DR
- OP(CIR)→AR
- DR→MEM[AR]

N.B. Per poter effettuare l'operazione, il registro PDR deve contenere il dato letto dalla periferica; questa condizione può essere verificata utilizzando il registro di stato PSR della periferica o tramite il registro RINT dell'unità centrale