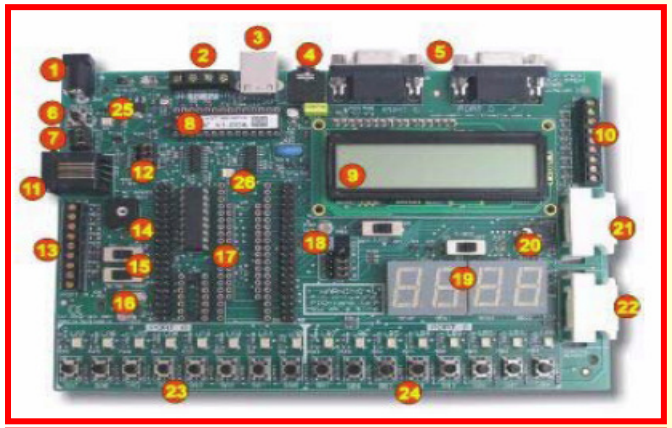
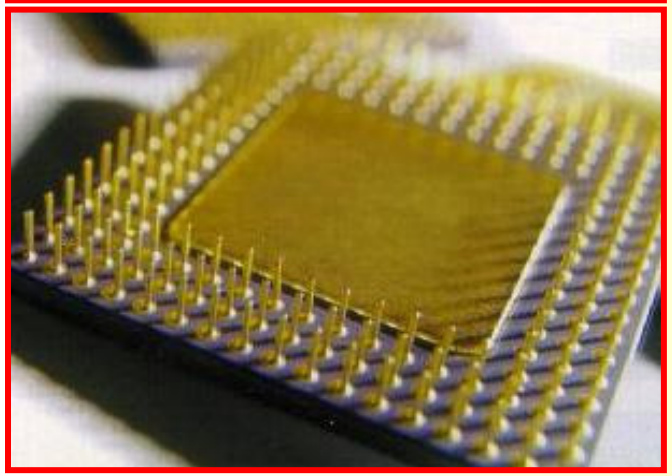
Università degli studi di Cassino

Corso di Laurea in
Ingegneria della Produzione Industriale





Corso di Informatica Applicata

Introduzione

Ing. Saverio De Vito
e-mail: saverio.devito@portici.enea.it
Tel.: +39 081 7723364

Contenuti del corso

- Architettura dell'unità di elaborazione;
- Modello di programmazione del processore;
- Modalità di indirizzamento della memoria;
- Modalità di I/O;
- Classi di Istruzioni;
- Programmazione MCUs in linguaggio Assembly.
- Programmazione MCUs in linguaggio C.
- Realizzazione di codici per l'interfacciamento con semplici sensori e attuatori (Laboratorio).

Testi

- Dispense dal corso
- Andrew S. Tanenbaum. *Structured Computer Organization, 5th Edition.* Prentice-Hall, 2005
- Martin P. Bates, *Introduction to Microelectronic Systems: The PIC 16F84 Microcontroller*
- Robert B Reese, *Microprocessors: From Assembly Language to C Using the PICI8FXX2*
- Appunti….

Informazioni Utili

Per contattare il docente:

Ing. Saverio De Vito
Centro Ricerche Enea -Portici (Na)
e-mail: *saverio.devito@portici.enea.it*
Skype: *saverux*
Tel.: +39 081 7723364
Web: www.afs.enea.it/devito

- Importante : Subject delle e-mail preceduto da :

  "[Corso I.APP]"

- Ricevimento Provvisorio : Lun 19-20 (?)

Applicazioni su µControllori

- In ambito produzione industriale, una cospicua parte delle applicazioni dell' informatica concerne l' utilizzo di microcontrollori e la pratica della programmazione embedded.
- In particolare:
  - Controllo di attuatori (robotica, device drivers, avionica, assistenza alla frenata, etc.)
  - Sensoristica estesa
  - Telecomunicazioni (SetTopBoxes, Transcodificatori, Routers, etc.)

Embedded: "Immerso" in un ambiente da controllare.
Si intende con programmazione embedded la programmazione di dispositivi integrati per il controllo in tempo reale di sensori ed attuatori.

µControllori & C…

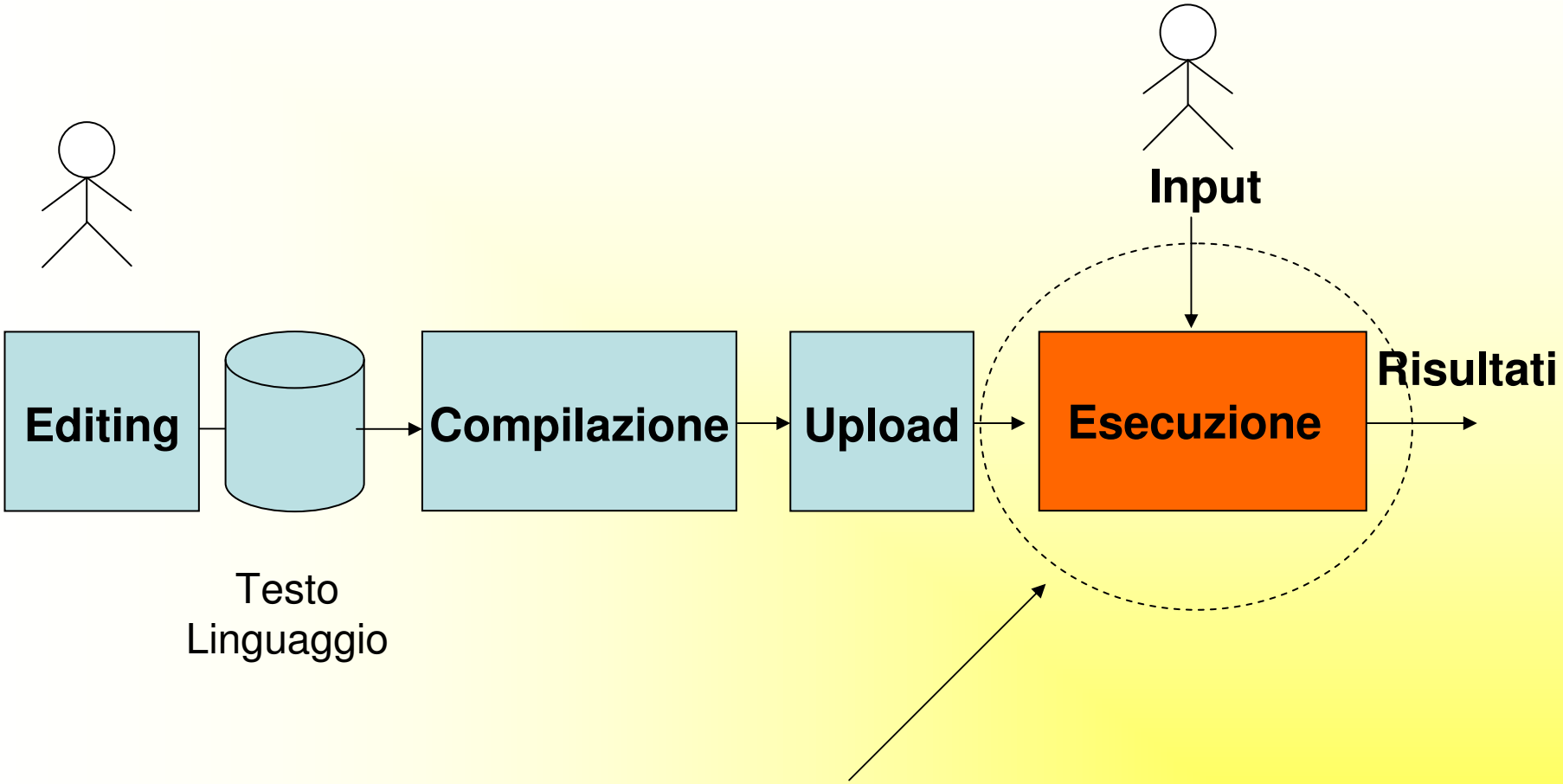
- In questo corso introdurremo la programmazione dei microcontrollori per applicazioni industriali.

- Guarderemo al microcontrollore come una unità di calcolo completa integrata dotata di opportune estensioni per l' interfacciamento di sensori/attuatori.

Proviamo per il momento……

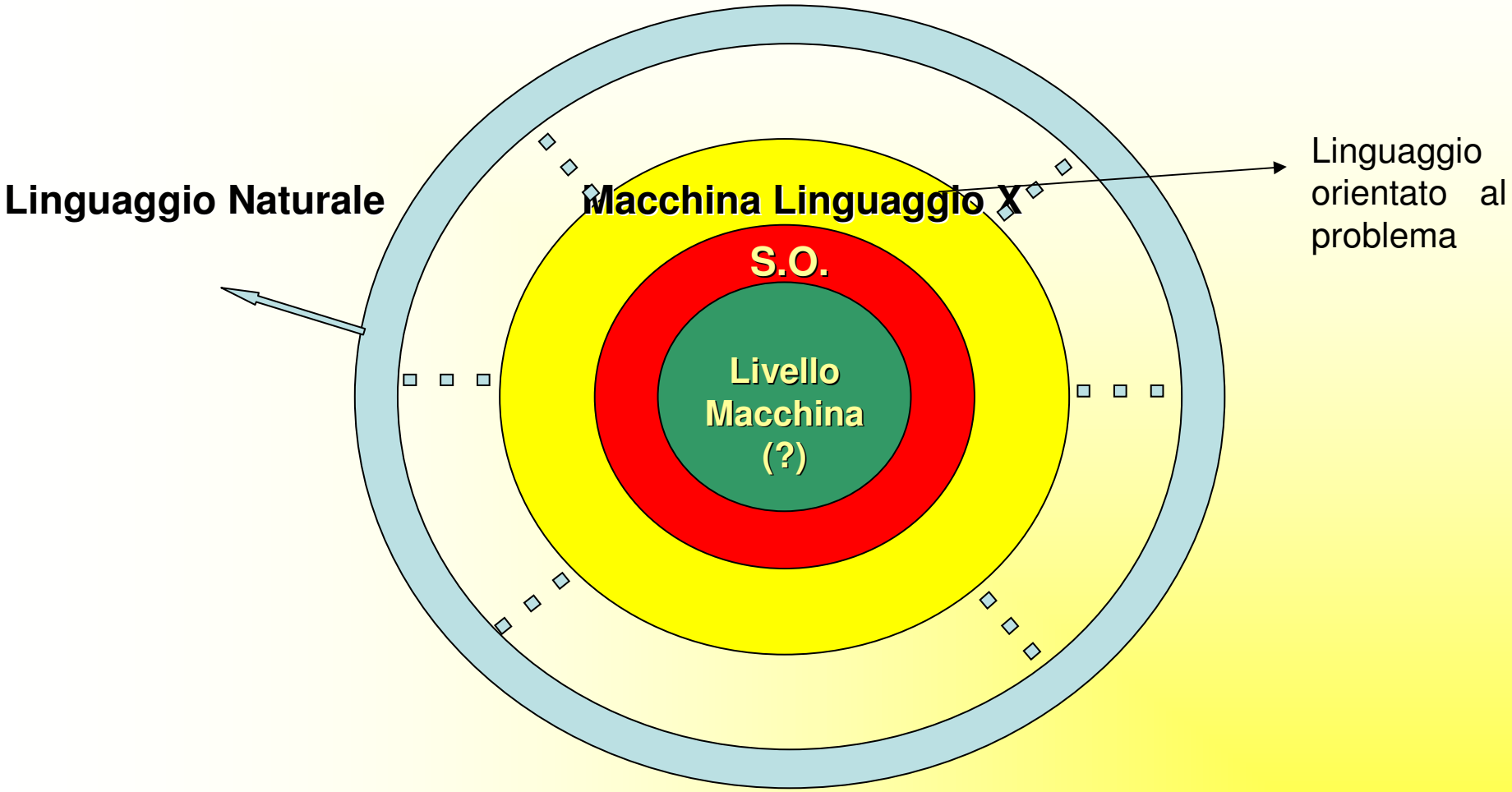
Ad astrarci dal contesto particolare della programmazione embedded ed a portarci nel contesto generale della programmazione delle architetture a microprocessore…. e guardiamo cosa succede alzando il coperchio dello scatolotto… ☺

Sviluppo ed esecuzione……



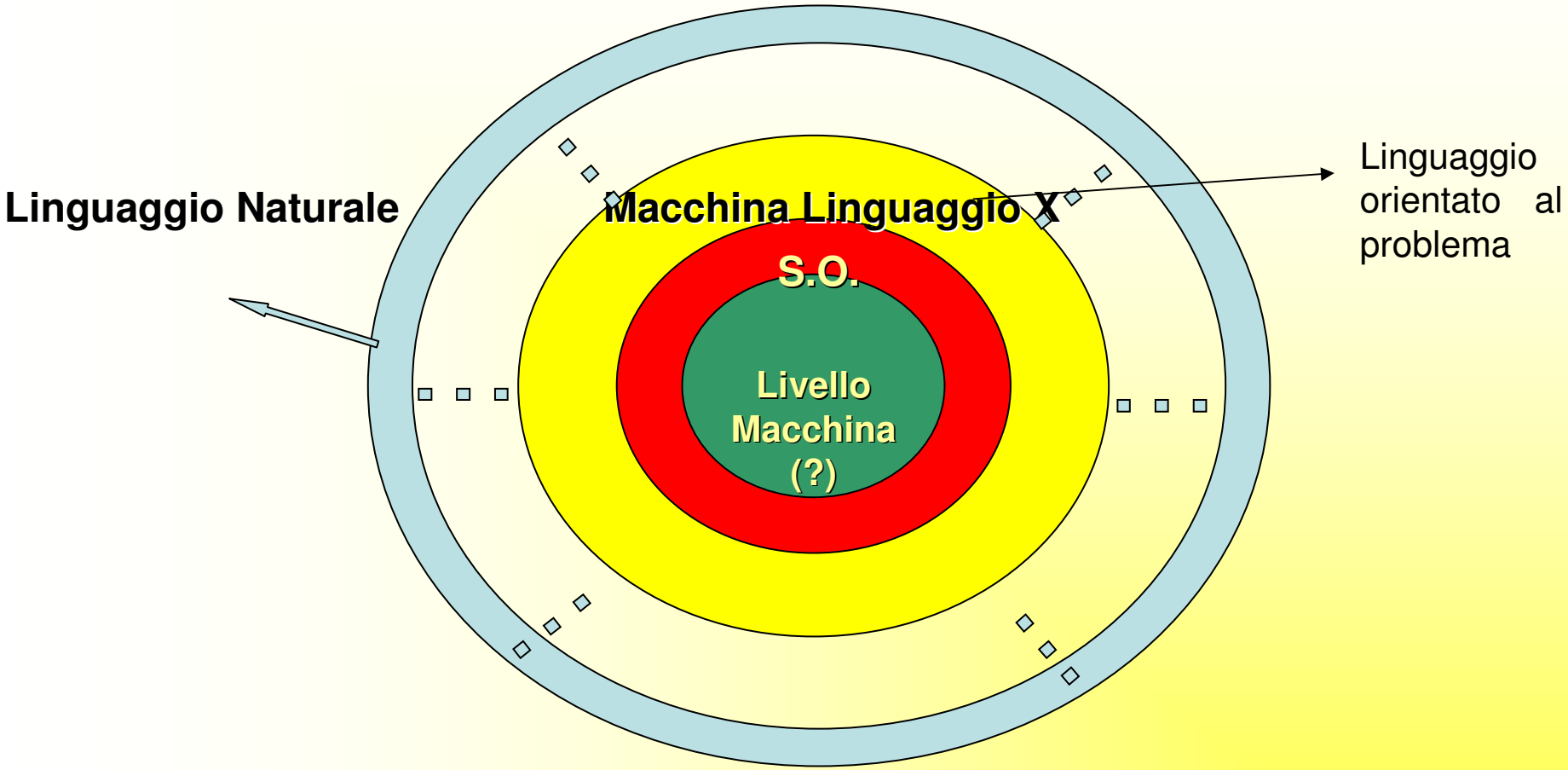
Che cosa succede qui ?

Modello a Cipolla



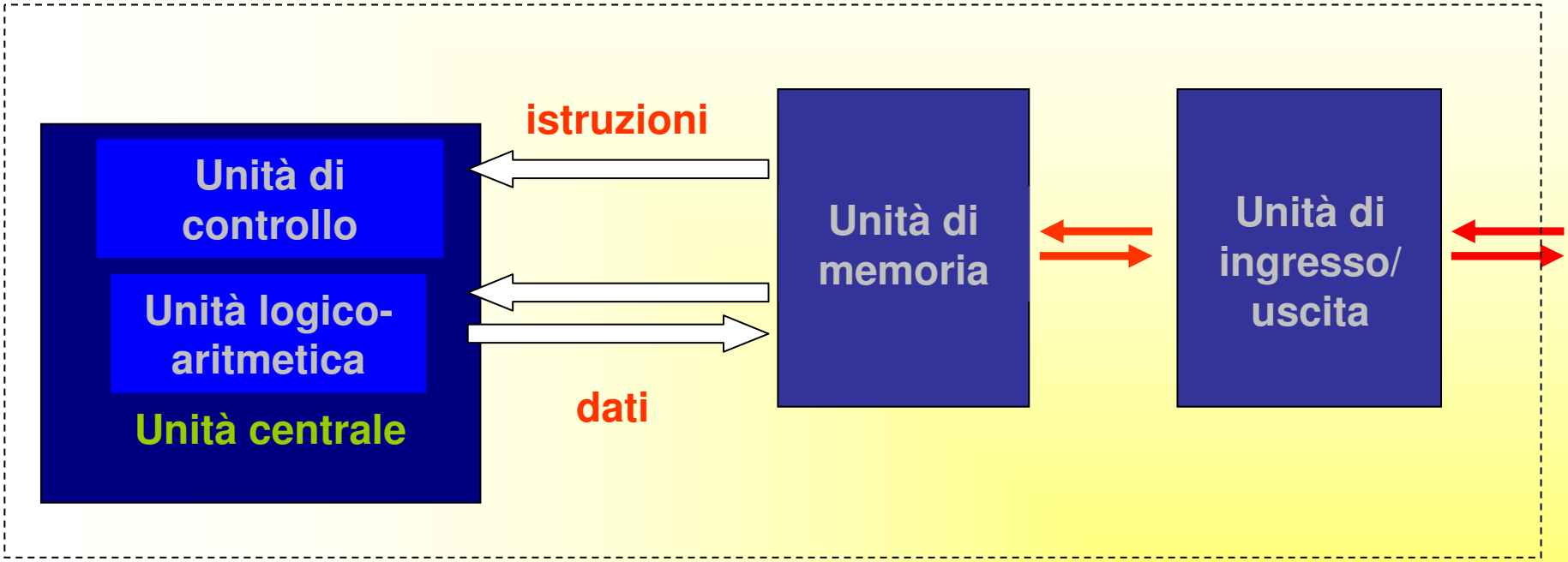
Per ogni linguaggio/astrazione si definisce una macchina virtuale in grado di eseguirne le istruzioni.

Modello a Cipolla



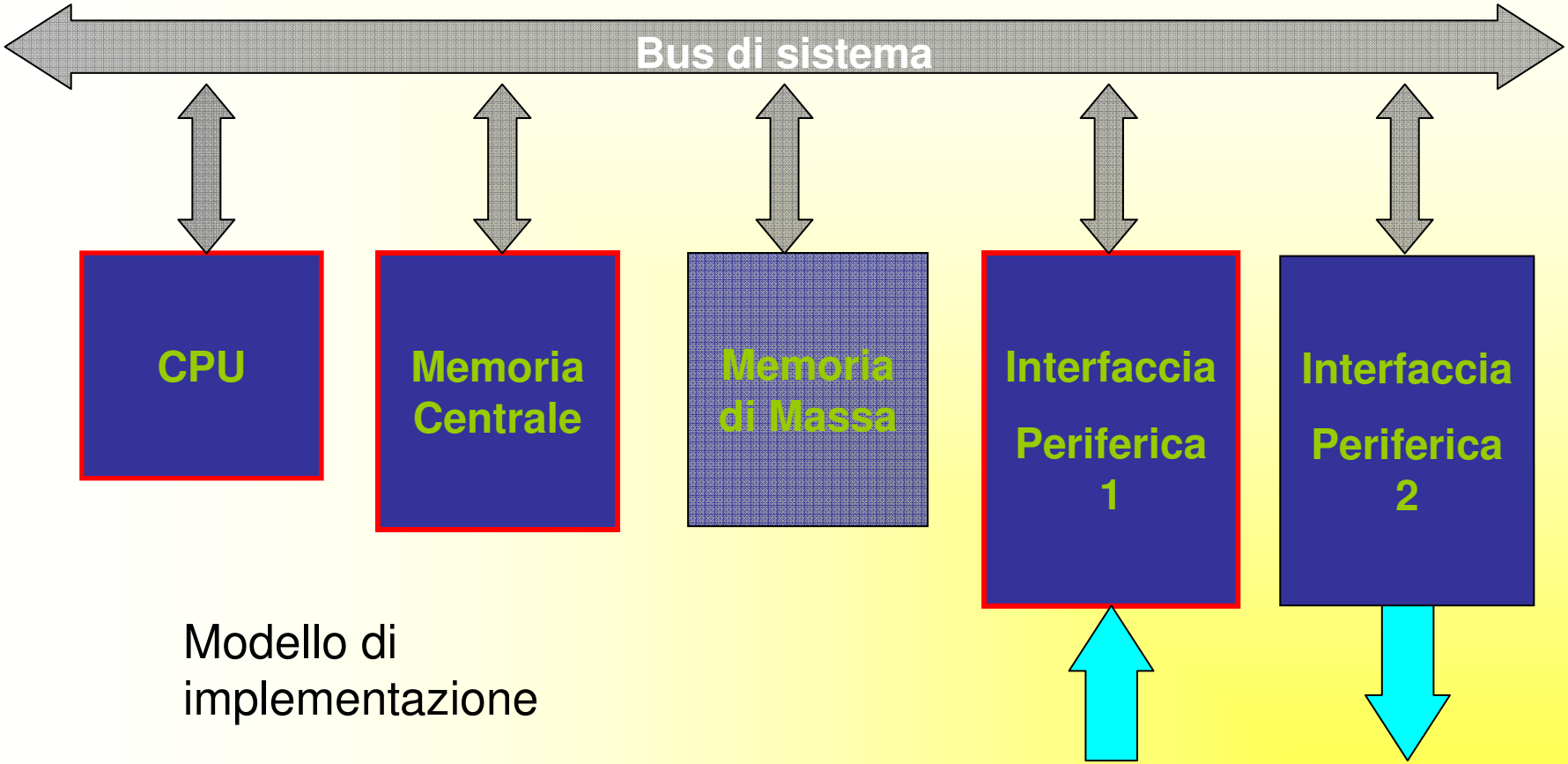
Ogni macchina virtuale sfrutta le capacità operative fornite dai livelli sottostanti, fornendo un set di istruzioni a livello semantico più alto per l' utilizzo da parte di livelli più evoluti. Il gap semantico tra uomo e macchina viene così superato.

Organizzazione generalizzata di un calcolatore



Istruzioni eseguite da una CPU operano su dati immagazzinati in unità di memoria eseguendo tra le altre operazioni di input/output da e verso il mondo esterno.

Modello di von Neumann



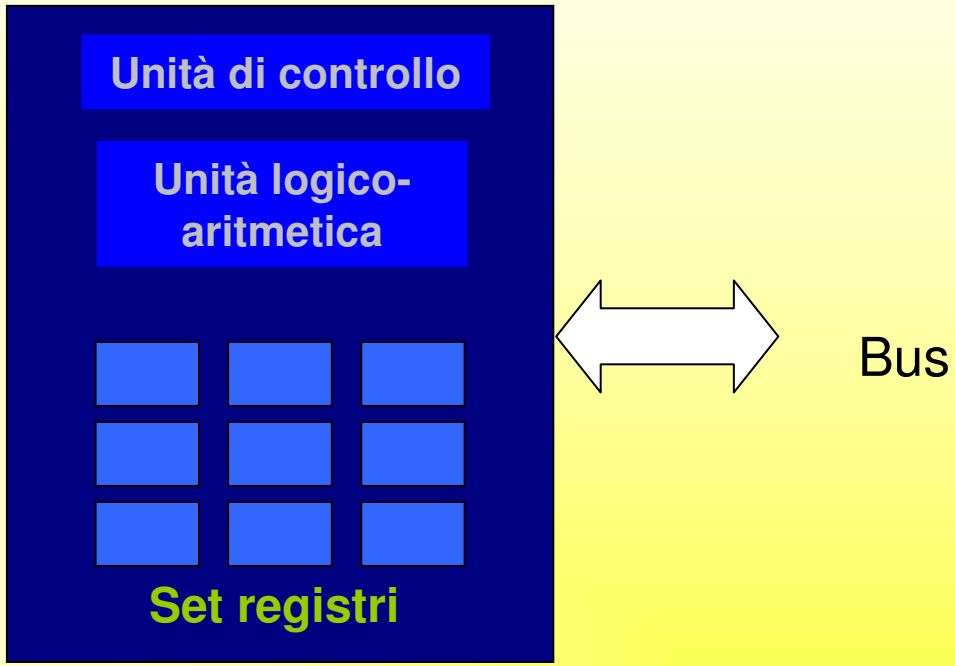
Modello di implementazione

CPU (Central Processing Unit)

Funzione:
eseguire i programmi immagazzinati in memoria principale prelevando le istruzioni (e i dati relativi), interpretandole ed eseguendole una dopo l'altra

E' costituita da:

- **Unità di controllo**
- **Unità logico aritmetica**
- **Set di registri di memoria**



L'Unità di controllo

E' l'unità che si occupa di dirigere e coordinare le attività interne alla CPU che portano all'esecuzione di una istruzione

Ciclo del processore

L'esecuzione di una istruzione avviene attraverso alcune fasi:

->Fetch

L'istruzione da eseguire viene prelevata dalla memoria e trasferita all'interno della CPU

->Decode

L'istruzione viene interpretata e vengono avviate le azioni interne necessarie per la sua esecuzione

->Operand Assembly

Vengono prelevati dalla memoria i dati su cui eseguire l' operazione prevista dalla istruzione

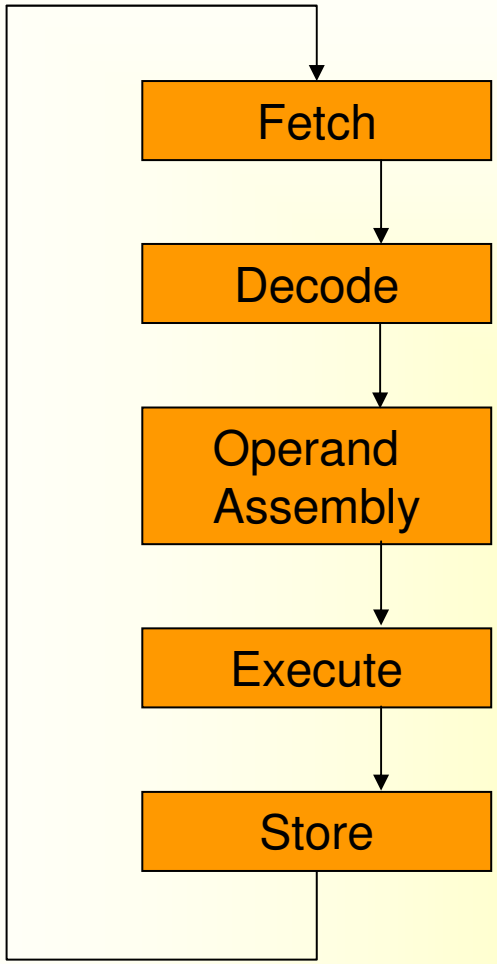
->Execute

Viene portata a termine l'esecuzione dell'operazione prevista dalla istruzione

->Store

Viene memorizzato il risultato dell'operazione prevista dalla istruzione

L'Unità di controllo



L'unità di controllo realizza ciclicamente le fasi per eseguire la sequenza di istruzioni che costituiscono il programma.

Essa coordina la circuteria* sottostante al fine di eseguire correttamente le singole fasi.

L'Unità Logico Aritmetica

E' l'unità che si occupa di eseguire le operazioni logiche ed aritmetiche eventualmente richieste per completare un'istruzione

Tipiche Operazioni Aritmetiche

ADD (Addizione)

SUB (Sottrazione)

MUL (Moltiplicazione)

DIV (Divisione)

Tipiche Operazioni Logiche

CMP (Comparazione)

AND (Congiunzione)

OR

NOT (Negazione)

Le operazioni logiche sono fondamentali per gestire il flusso di esecuzione in maniera condizionale (se <X> allora..)

I registri

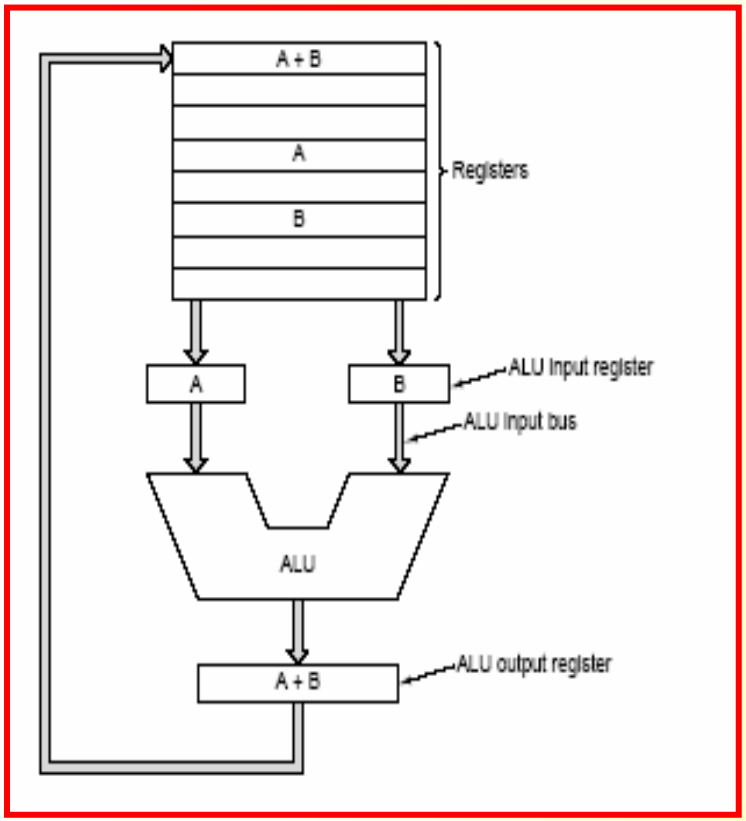
Hanno la funzione di memorizzare all'interno della CPU dati e istruzioni necessari all'esecuzione

- **Registri generali**
- **Registri speciali**
  - Program Counter (PC)
  - Mem. Address Reg. (MAR)
  - Mem. Data Register (MDR)
  - Istrunction Register (IR)

I registri speciali non sono tipicamente accessibili dalle istruzioni della macchina standard.

Data Path del processore

Il Data Path descrive il percorso dei dati attraverso i sottosistemi componenti nell' esecuzione delle istruzioni



Esempio:
Istruzione Aritmetica su registri verso registro:

ADD reg1,reg2

1. Gli operandi A e B vengono utilizzati per precaricare i registri speciali dell' ALU
2. La ALU calcola il risultato deponendolo nel registro speciale di output dell' ALU
3. Il risultato viene posto nel registro di destinazione designato (Molte architetture hanno un registro designato per contenere gli output di operazioni aritmetico/logiche)

Memoria principale

•La memoria centrale rappresenta il “magazzino” dove vengono conservate le istruzioni ed i dati relativi al programma in esecuzione

•Una memoria interagisce con il processore secondo le seguenti modalità di azione:

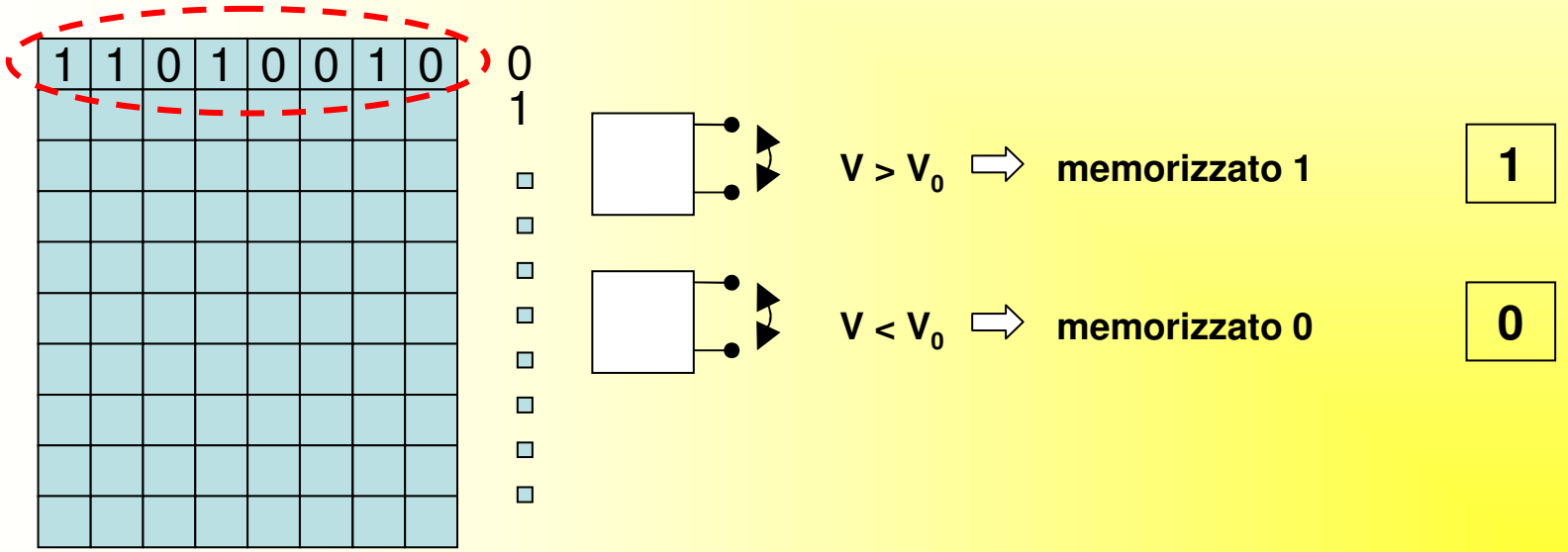
–Lettura: trasferimento dati dall’ unità di memoria al processore
–Scrittura: trasferimento dati dal processore all’ unità di memoria

I trasferimenti avvengono da celle di memoria (registri di memoria) a registri interni del processore che la controlla

Organizzazione della memoria

- Una memoria è costituita da elementi di informazione elementari (bit) organizzati in locazioni (parole, registri di memoria) singolarmente indirizzabili dal processore o dagli altri dispositivi del sistema.

- Il dispositivo utilizzato per memorizzare un singolo bit è un **elemento bistabile**, cioè un dispositivo elettronico che può assumere uno tra due stati stabili (es. due livelli differenti di tensione), ognuno dei quali viene fatto corrispondere a 0 o a 1 (**cella di memoria**).



Il registro di memoria

- L' insieme delle N celle elementari bistabili di cui è composto un registro di memoria può assumere uno tra 2^N stati possibili. In particolare un insieme di 8 bit forma un byte. N è detto parallelismo di accesso.

- Il registro costituisce un supporto per la memorizzazione di un'informazione che può assumere uno tra 2^N valori possibili in corrispondenza dei 2^N stati del registro (codifica).

- Sul registro sono possibili operazioni di lettura e scrittura che interessano contemporaneamente tutte le celle di memoria contenute nel registro

Il problema della codifica

- Un calcolatore può trattare diversi tipi di dati: numeri (interi, reali), testo, immagini, suoni, ecc. che vanno comunque immagazzinati nei registri di memoria.
- Una **codifica** di un tipo di dato qualsiasi consiste nell' individuare una corrispondenza biunivoca (iniettiva) tra l' insieme dei valori che può assumere il tipo di dato in considerazione e l' insieme dei 2N stati in cui può trovarsi un registro di memoria.
- Esempio:

{a,b,c,…..z} <=> {00000,00001,00010….}

Ulteriore Esempio

Utilizzando registri da un byte $\Rightarrow 2^8 = 256$ stati possibili.

Che cosa è possibile codificare ?

Numeri naturali [0,255]

0 ↔ 00000000
1 ↔ 00000001
....
255 ↔ 11111111

Numeri interi [-128,127]

-128 ↔ 00000000
-127 ↔ 00000001
0 ↔ 10000000
+127 ↔ 11111111

Numeri reali [0,1[

0.0000 ↔ 00000000
0.0039 ↔ 00000001
0.0078 ↔ 00000010
....
0.9961 ↔ 11111111

Caratteri

A ↔ 01000001
a ↔ 01100001

0 ↔ 00110000
1 ↔ 00110001

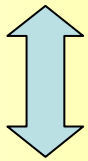
La codifica implica una rappresentazione dei dati limitata e discreta

La codifica di un tipo di dati può estendersi su più registri di memoria

Possiamo codificare le istruzioni?

- Certamente le istruzioni macchina rappresentano un insieme codificabile poiché tipicamente limitate e numerabili.

Addiziona 1 al registro di memoria di indirizzo 4



01001010 00100100

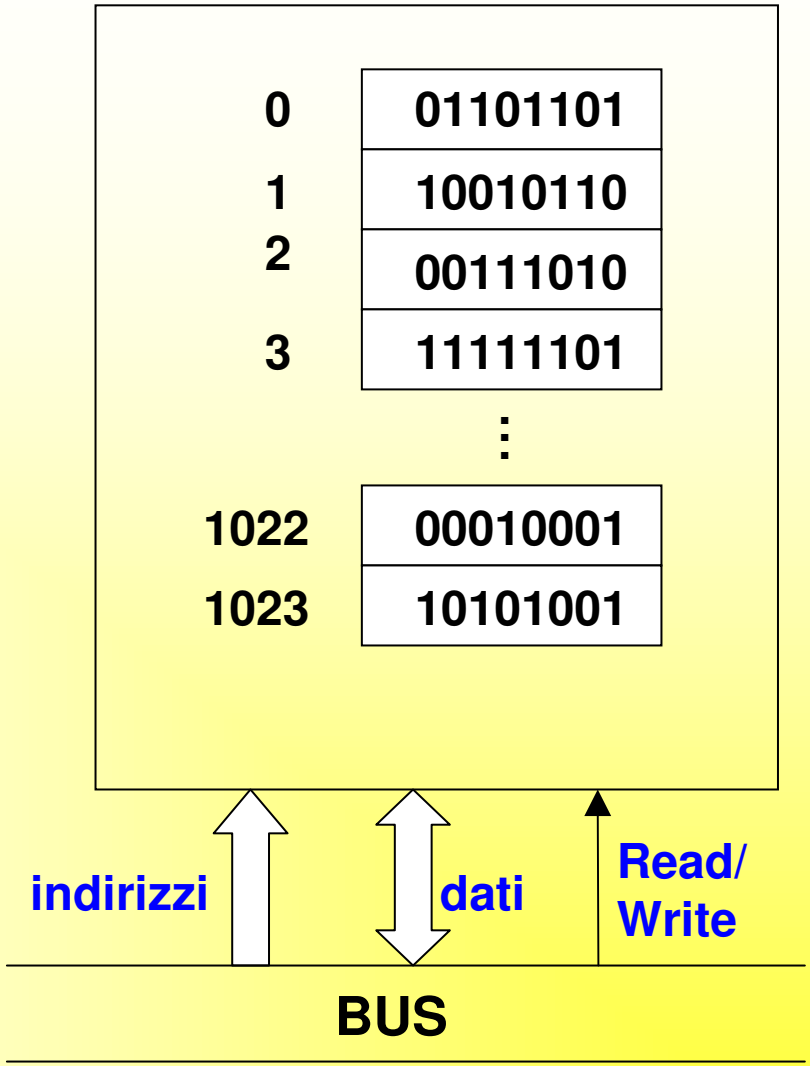
Non c'è nulla che possa aiutarci a distinguere dati da istruzioni e dati di tipo diverso dal solo valore contenuto nel registro (Mai fatto il dump di un sistema?).

Organizzazione della memoria (2)

Il modulo di memoria principale è fisicamente connesso al processore e ai restanti dispositivi del sistema tramite il BUS.

Generalmente, sono presenti tre gruppi di linee:

- **linee indirizzi**
- **linee dati**
- **linee Read/Write**



Durante un operazione sulla memoria…..

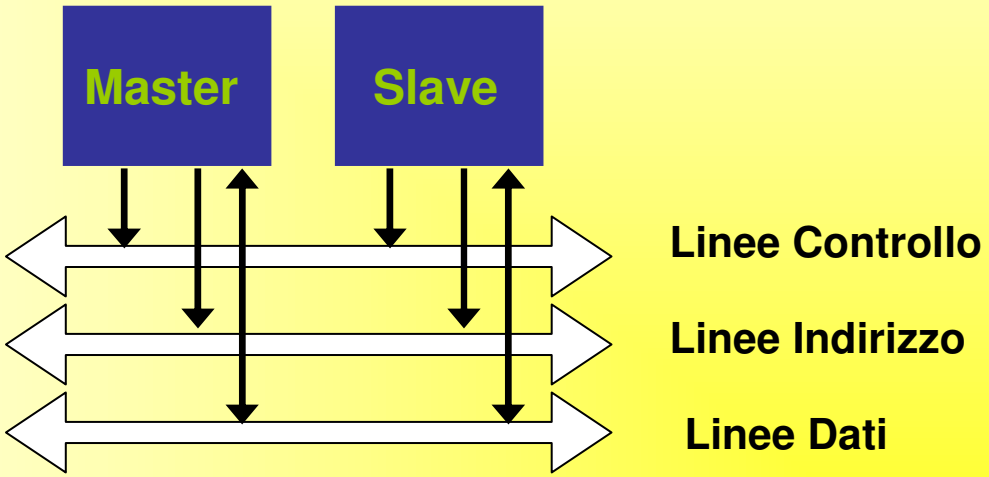
- **Linee indirizzi**
  - **I valori che le linee indirizzi assumono rappresentano la localizzazione fisica del registro di memoria oggetto delle operazioni. L' ampiezza delle linee (8-16-32-64 bit) ci dà ragione della quantità di memoria indirizzabile da un sistema.**

- **Linee dati**
  - **I valori che troviamo sulle linee dati rappresentano i dati scambiati da e verso la memoria**

- **Linee Read/Write**
  - **I valori di queste linee ci danno ragione della tipologia di operazione in corso**

Più in generale il bus….

- Forma un canale di comunicazione tra le varie unità del calcolatore.
- Tipicamente è possibile un solo colloquio alla volta tra due unità: un **master**, che ha la capacità di controllare il bus ed inizia la comunicazione, ed uno **slave**, che viene attivato dal master.
- Il bus è formato da un insieme di linee su cui viaggiano i segnali. Le linee si dividono in
  - **linee dati**
  - **linee indirizzi**
  - **linee controllo**



Trasferimento CPU-memoria

- Qualunque sia il trasferimento da realizzare, la CPU (master) deve precisare l'indirizzo del dato da trasferire.

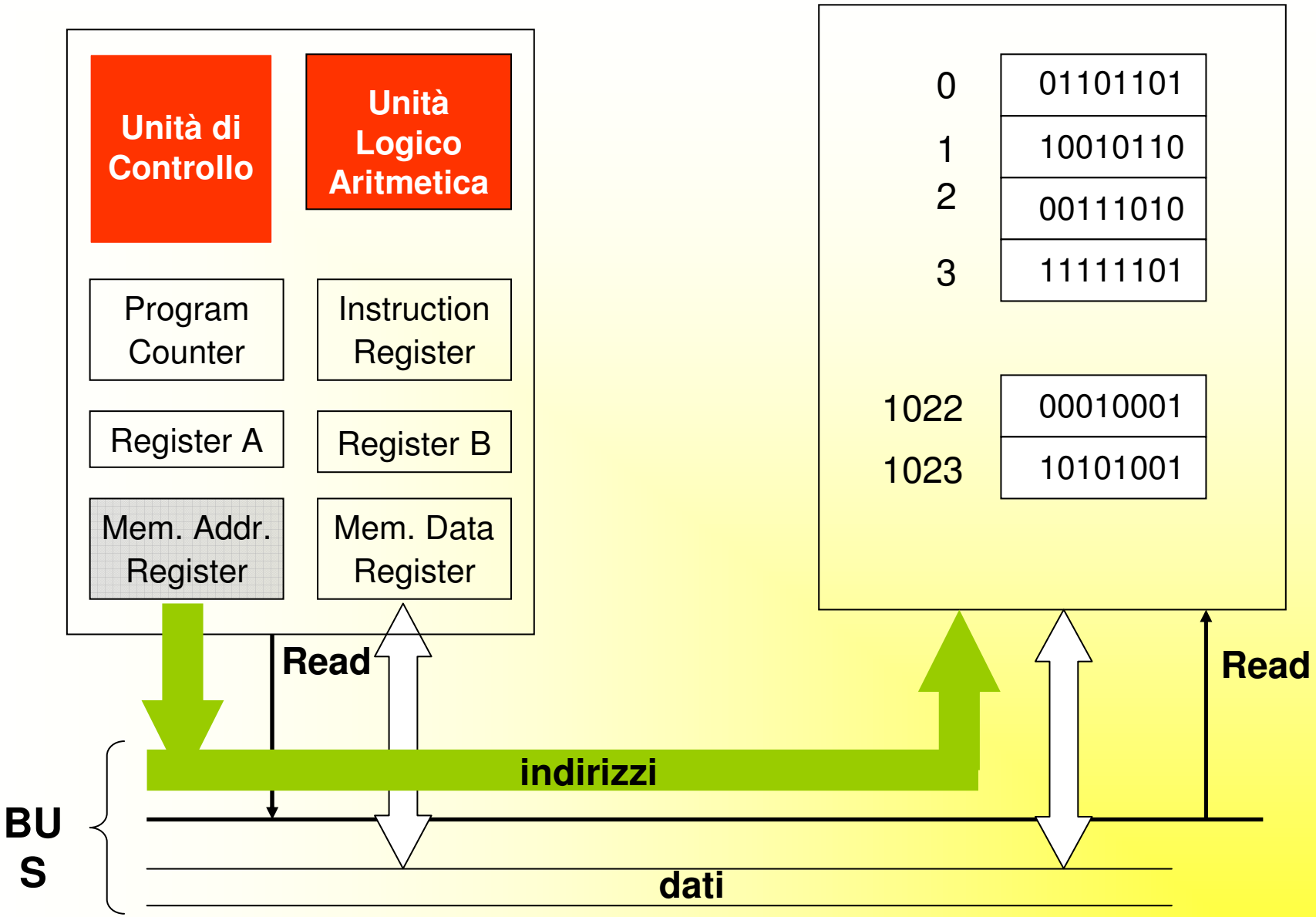
- In queste operazioni, la memoria è comunque uno slave e "subisce" l'iniziativa della CPU, ricevendo da questa l'indirizzo del dato da trasferire e l'informazione sull'operazione da realizzare (lettura o scrittura)

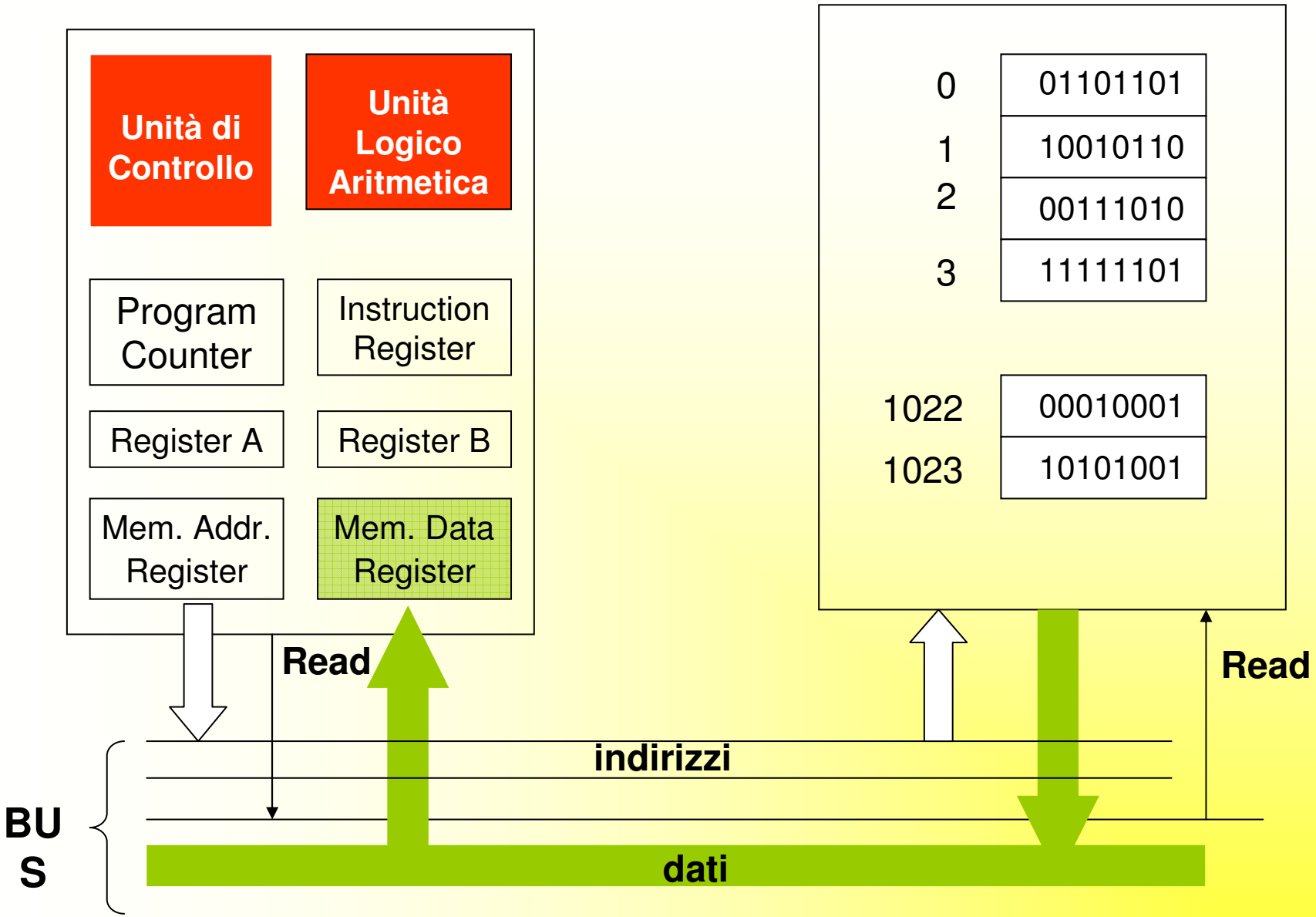
Trasferimento memoria → CPU (lettura)

1. la CPU scrive l'indirizzo del dato da trasferire sul MAR che lo propagherà alle linee indirizzi del bus. Contemporaneamente, segnala sulle linee di controllo che si tratta di una lettura.

2. la memoria riceve, tramite il bus, l'indirizzo e l'indicazione dell'operazione da effettuare. Copia il dato dal registro individuato sulle linee dati del bus.

3. il dato richiesto, tramite le linee dati del bus, arriva al MDR della CPU. Da qui sarà spostato verso gli altri registi interni.

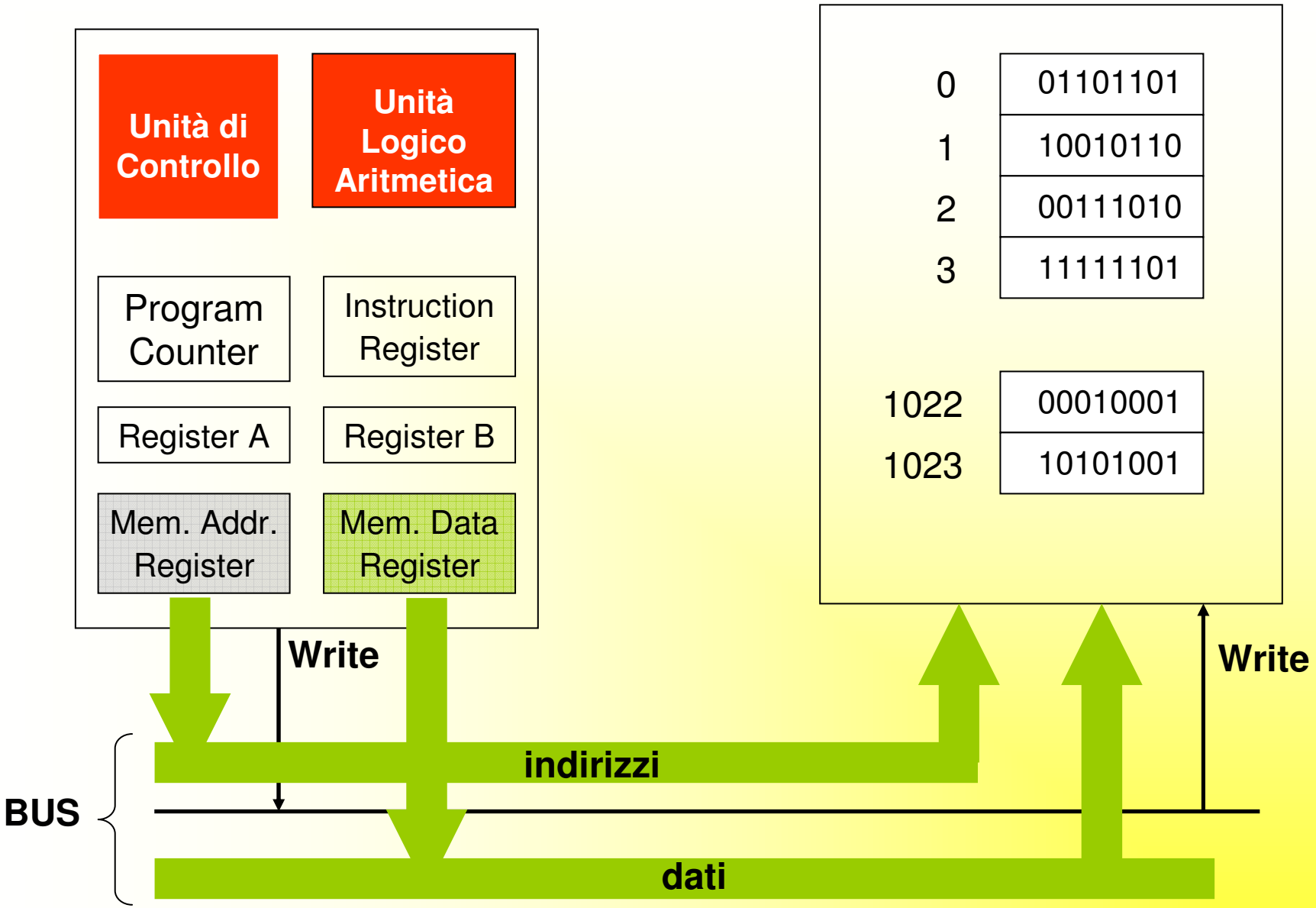

Unità di Controllo
Unità Logico Aritmetica
Program Counter
Instruction Register
Register A
Register B
Mem. Addr. Register
Mem. Data Register
0
01101101
1
10010110
2
00111010
3
11111101
1022
00010001
1023
10101001
Read
Read
indirizzi
BUS
dati

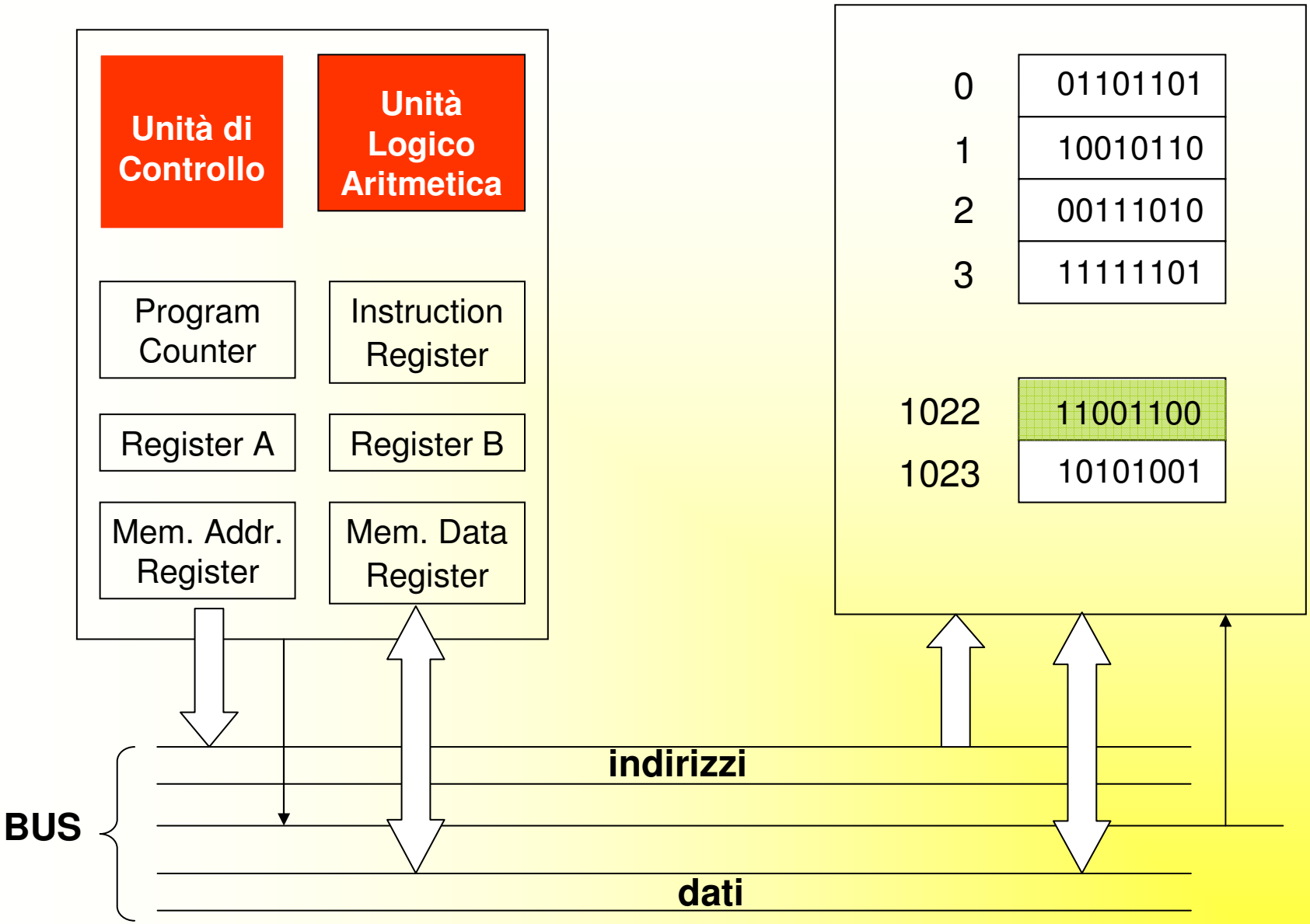

Unità di Controllo
Unità Logico Aritmetica
Program Counter
Instruction Register
Register A
Register B
Mem. Addr. Register
Mem. Data Register
Read
0
01101101
1
10010110
2
00111010
3
11111101
1022
00010001
1023
10101001
Read
indirizzi
BU
S
dati

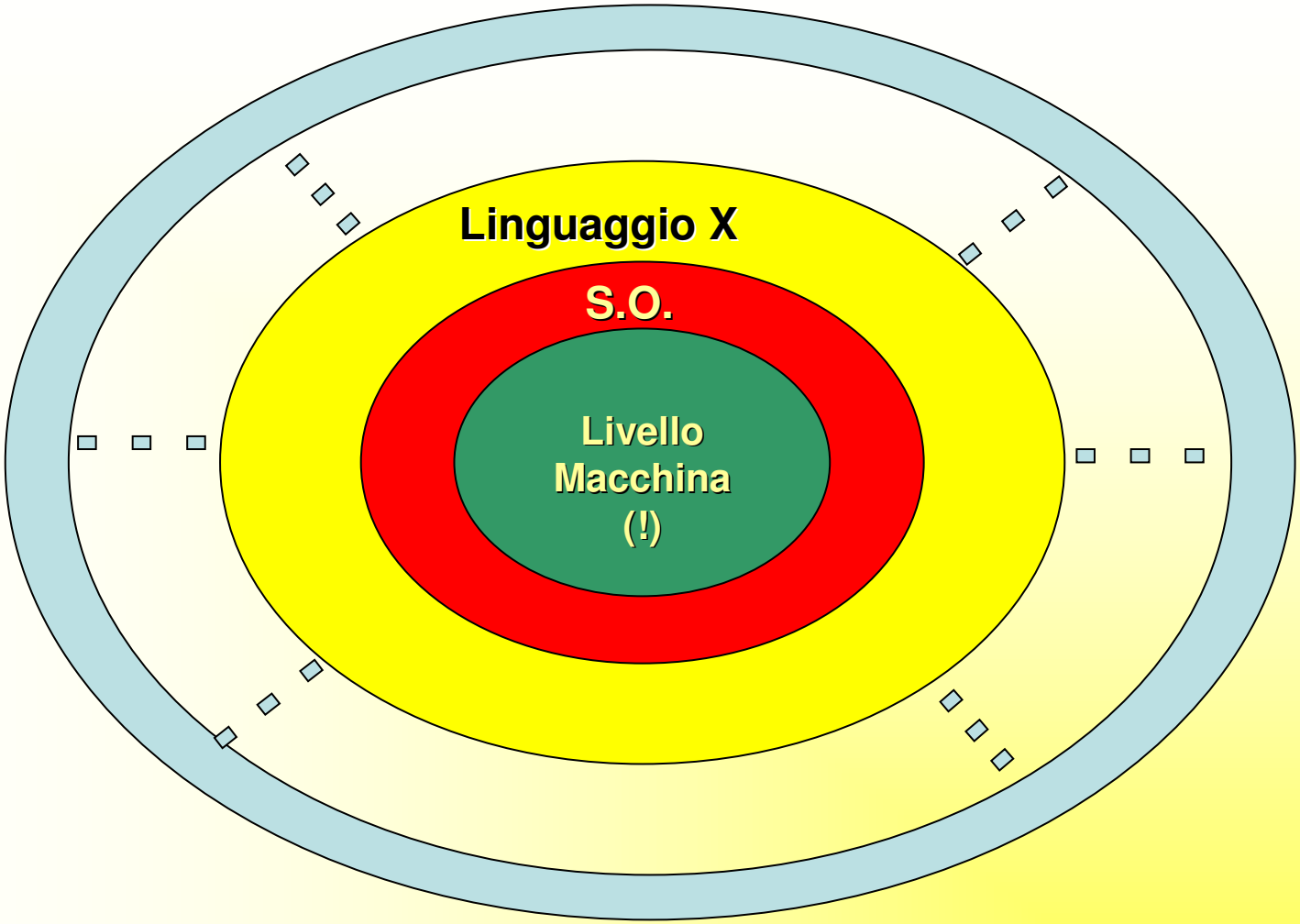
Trasferimento CPU → memoria (scrittura)

1. la CPU scrive l'indirizzo del dato da trasferire sul MAR, mentre il dato viene copiato sul MDR. Il contenuto dei due registri viene propagato sulle linee indirizzi e dati del bus. Contemporaneamente, la CPU segnala sulle linee di controllo che si tratta di una scrittura.

2. la memoria riceve, tramite il bus, l'indirizzo, il dato e l'indicazione dell'operazione da effettuare. Copia il dato dalle linee dati del bus al registro individuato dall'indirizzo.


Unità di Controllo
Unità Logico Aritmetica
Program Counter
Instruction Register
Register A
Register B
Mem. Addr. Register
Mem. Data Register
Write
0
1
2
3
01101101
10010110
00111010
11111101
1022
1023
00010001
10101001
Write
indirizzi
BUS
dati


Unità di Controllo
Unità Logico Aritmetica
Program Counter
Instruction Register
Register A
Register B
Mem. Addr. Register
Mem. Data Register
0
01101101
1
10010110
2
00111010
3
11111101
1022
11001100
1023
10101001
indirizzi
BUS
dati



In realtà la complessità del modello a cipolla può estendersi al di sotto del Livello della macchina standard…….

Microprogrammazione (Livello della)

- A questo punto dovrebbe essere chiaro che le istruzioni fornite dal livello della macchina standard sono troppo complesse da essere eseguite da un hardware (il datapath del processore) che rimanga relativamente semplice

- In un trade off tra costo di hardware complesso e tempo impiegato nell' esecuzione di una macchina virtuale si sceglie di far eseguire le istruzioni standard da una macchina virtuale che interpreti le suddette frammentandole in molteplici "microistruzioni" che il datapath è in grado di eseguire in un clock (del datapath)

- Nel nucleo (?) del modello troviamo dunque le microistruzioni che possono essere tradotte facilmente in azioni (livelli logici) con cui pilotare i dispositivi logici (Registri, contatori etc.)

Riepilogo et al.:

- Introduzione e descrizione dello scenario applicativo
- Il modello di von Neumann
  - Componenti principali dell' architettura
  - Ciclo di von Neumann
  - Il modello a cipolla nella programmazione

Le trasparenze utilizzate in questa lezione sono in larga parte tratte dalle dispense del Prof. Tortorella, che qui si ringrazia per il permesso all' utilizzo, per il corso di Calcolatori I per Ingegneria delle Telecomunicazioni.