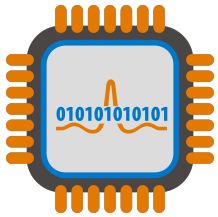
4 Sistemi a microprocessore

Fondamenti di Informatica P2

Ingegneria Meccatronica



Stefano Mattoccia
Dipartimento di Informatica
Università di Bologna

Elaborazione delle informazioni

Un essere vivente continuamente elabora delle informazioni e fornisce delle risposte o agisce in un determinato modo in base alle informazioni in ingresso.

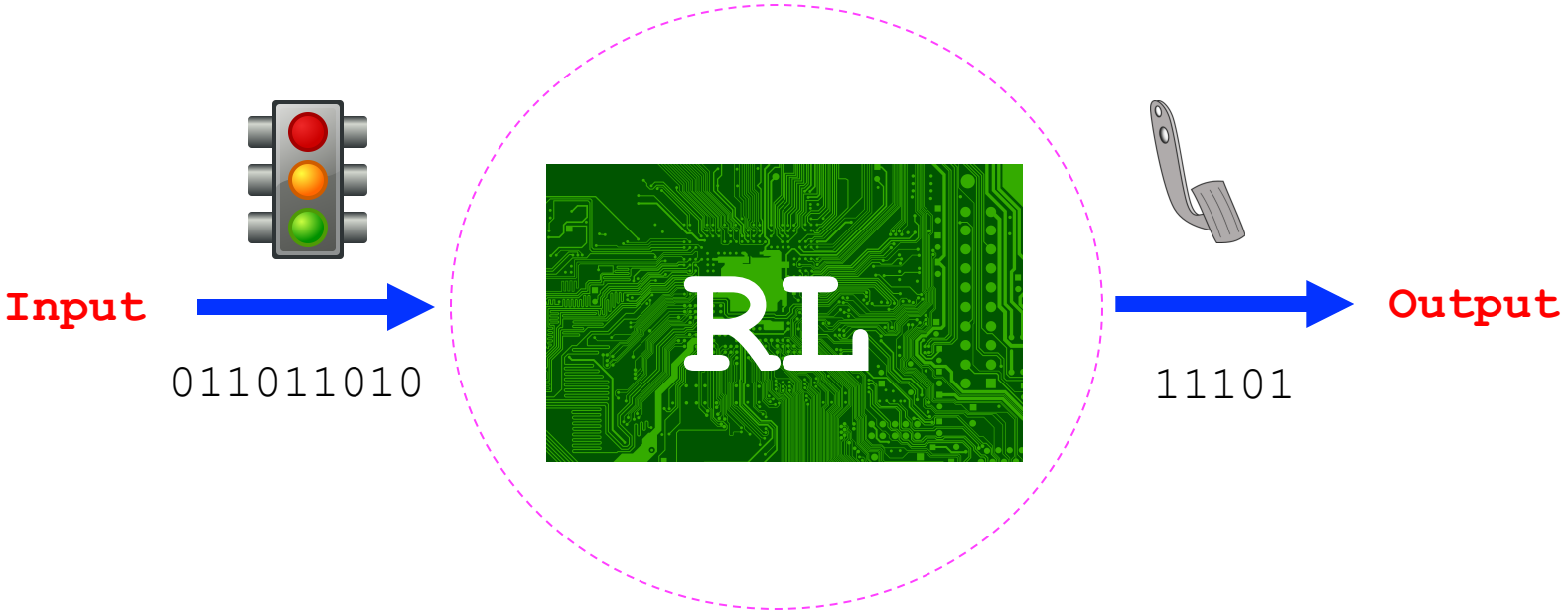
Un sistema biologico per l'elaborazione delle informazioni isolato dall'esterno servirebbe a ben poco (meglio, nulla).

Input e output debbono essere codificati in modo appropriato

Elaborazione delle informazioni

Ovviamente, in questo corso, siamo interessati a sistemi di natura per elaborare le informazioni.

In particolare, per le ragioni evidenziate nella prima parte del corso, siamo interessati a sistemi digitali



Elaborazione delle informazioni

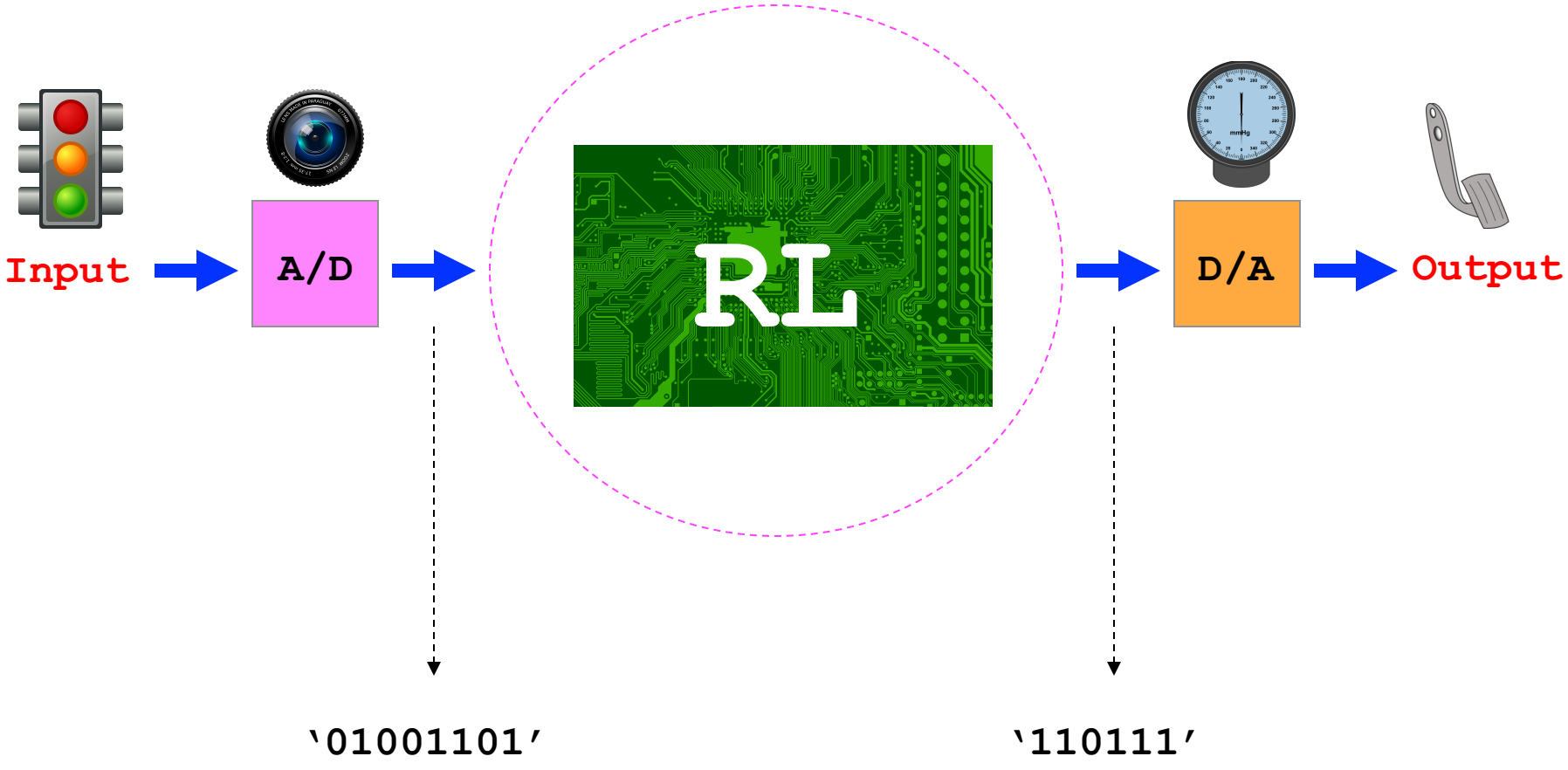
Un primo problema: le reti logiche elaborano informazioni di tipo digitale

Molti input e output di particolare interesse non sono di natura digitale ma analogica (quella preferita dagli esseri umani)

- **Prossimità a un semaforo o un ostacolo**
- **Monitor**
- **Movimento del mouse**
- **Voce/Audio**
- **Pressione tasti**
- **Touch screen**
- **etc**

Elaborazione delle informazioni

Pertanto, è necessario (spesso) convertire segnali di natura analogica in segnali digitali e viceversa



In ingresso e in uscita dal sistema di elaborazione solo segnali digitali (binari)

Elaborazione delle informazioni

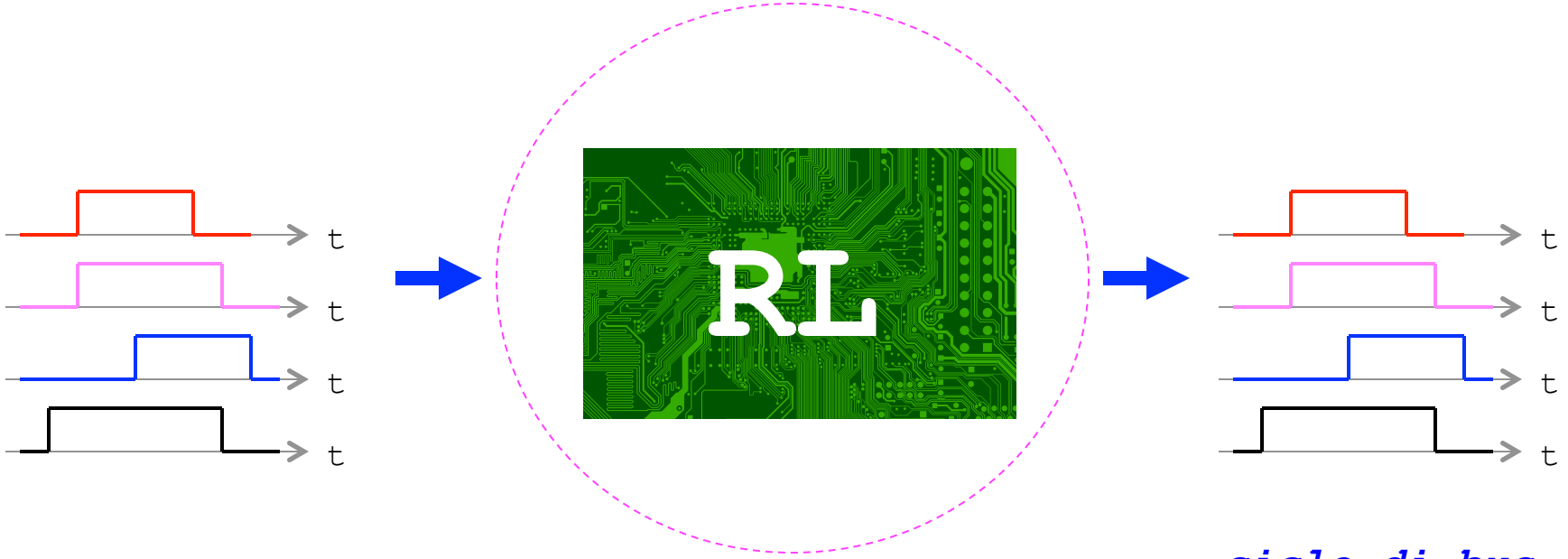
Altro problema: la rete logica che elabora le informazioni è spesso molto efficienti/veloce/etc ma allo stesso tempo poco flessibile per quanto riguarda il *linguaggio* utilizzabile per fornire le informazioni (digitali) in input e *interpretare* le informazioni (digitali) elaborate in output.

Se si desidera interagire con un sistema di elaborazione digitale e necessario adeguarsi al *linguaggio* che la rete logica utilizza.

Questo linguaggio è codificato dall'evoluzione temporale di alcuni segnali (il minimo indispensabile) emessi o ricevibili dalla rete logica. L'evoluzione temporale di questi segnali si definisce *ciclo di bus* (*bus cycle*).

Elaborazione delle informazioni

Quindi, l'unico modo per interagire con un sistema digitale per l'elaborazione delle informazioni consiste nell'adottare questo *linguaggio* (descritto in dettaglio sui *datasheet* che il produttore del sistema rende sempre disponibili).



ciclo di bus

ciclo di bus

Quale rete logica utilizzare?

Sono disponibili diverse architetture di elaborazione (i,e, reti logiche). Quale scegliere?

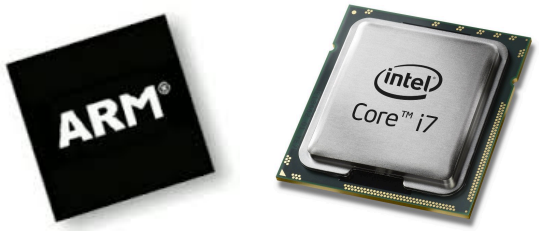
Dipende dal contesto applicativo e dalle prestazioni, consumi, ingombri, peso, etc:

- sistema general purpose
- sistema embedded
- sistema per videogiochi
- etc

Una tipologia di architettura, basata sul modello di Von Neumann, è molto più flessibile delle altre e può essere utilizzata, anche se non sempre con risultati ottimali, in ogni contesto. Questo modello è l'oggetto di questo corso e utilizza come elemento di base una CPU (microprocessore).

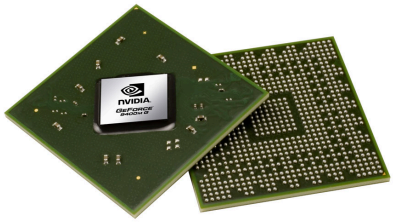
Esempi di architetture di elaborazione

- CPU (Central Processing Unit)
- CPU Multicore
- CPU Embedded



- SOC (System on a Chip)

- GPU (Graphic Processing Unit)



- FPGA (Field Programmable Gate Array)

- Sistemi ibridi (e.g. FPGA + CPU)

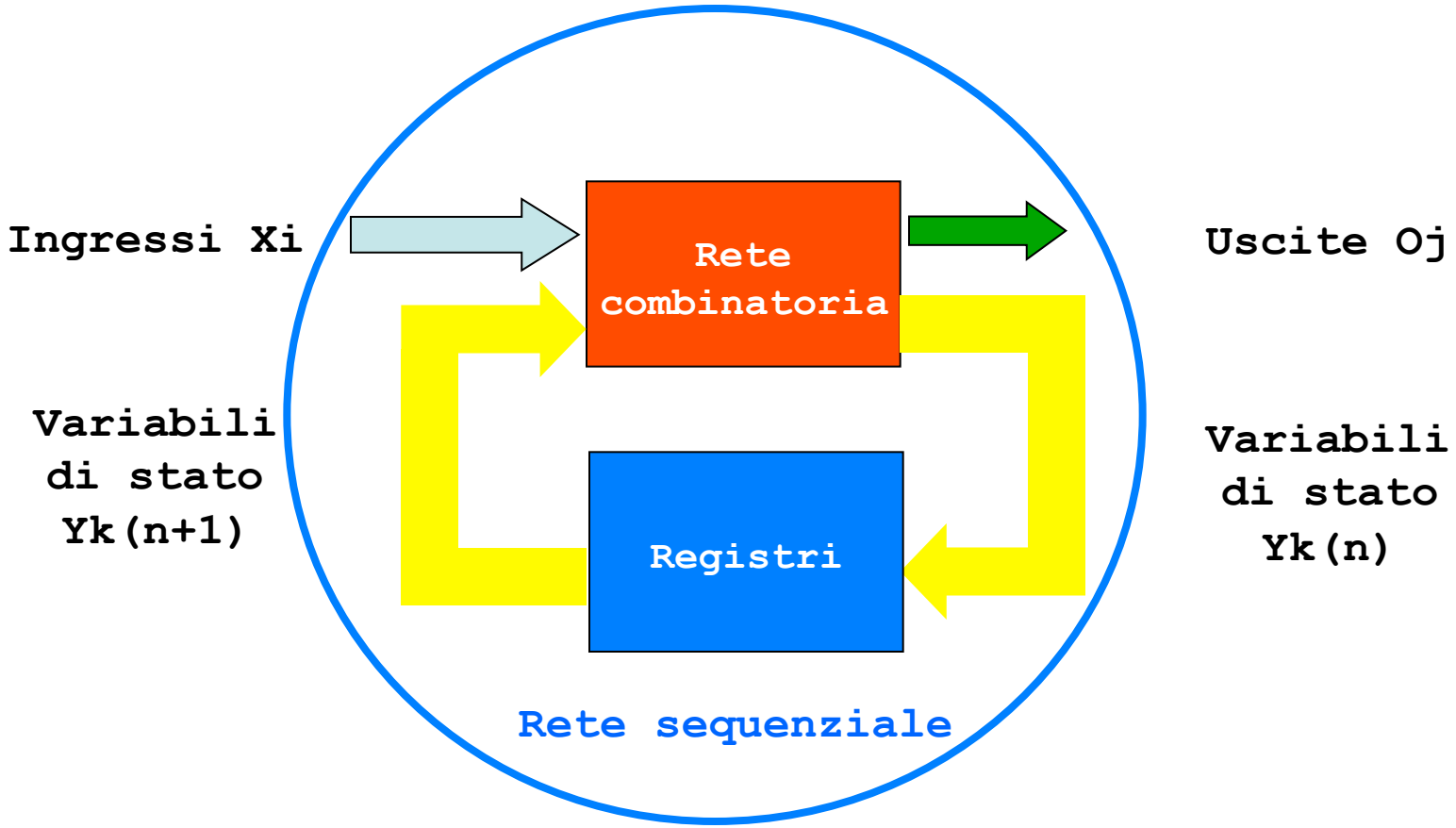
- . . .

Il modello di riferimento: Von Neumann

- Architettura del tutto generale che porta a realizzazioni poco dipendenti dal funzionamento desiderato
- Il funzionamento desiderato è espresso in termini di
  - sequenza di istruzioni (programma)
  - memorizzate su un supporto di memoria
- Per cambiare funzionamento è sufficiente cambiare il programma: questo di fatto "modifica" la rete logica di controllo per ogni istruzione eseguita
- L'architettura è adatta a trattare problemi molto più complessi di quelli visti nella prima parte del corso ma con efficienza molto minore.
- L'importanza e la diffusione dei calcolatori dipende fortemente dall'unicità di questo modello

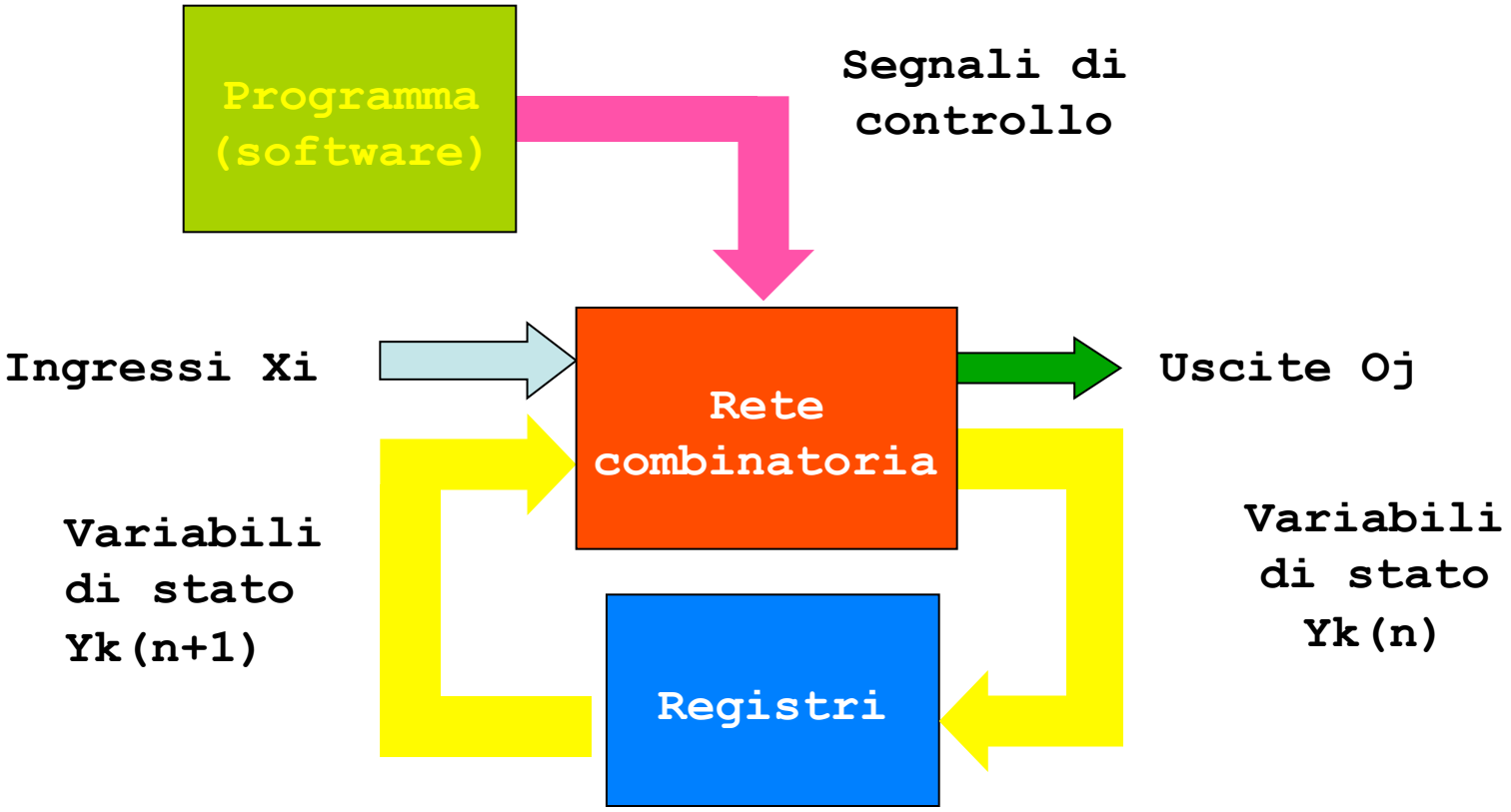
Sistema di elaborazione: rete sequenziale

Il sistema di elaborazione può essere schematizzato in modo astratto come una rete sequenziale (sincrona).

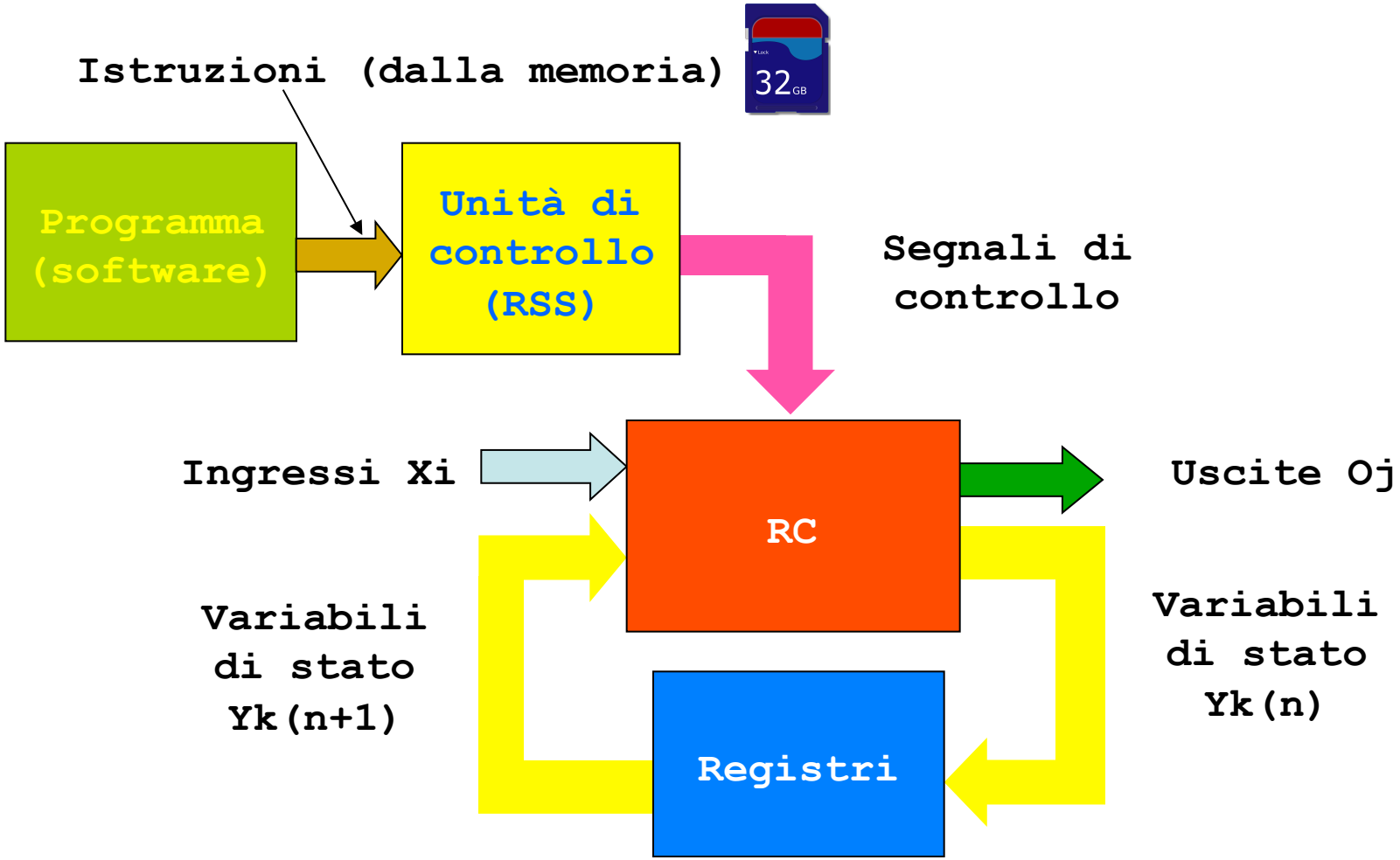


Come *cambiare* il funzionamento della RL?

Rispetto a Reti Logiche, la novità è il software (programma) che consente di variare il funzionamento della rete in base alle esigenze desiderate.



In realtà le cose sono più complesse

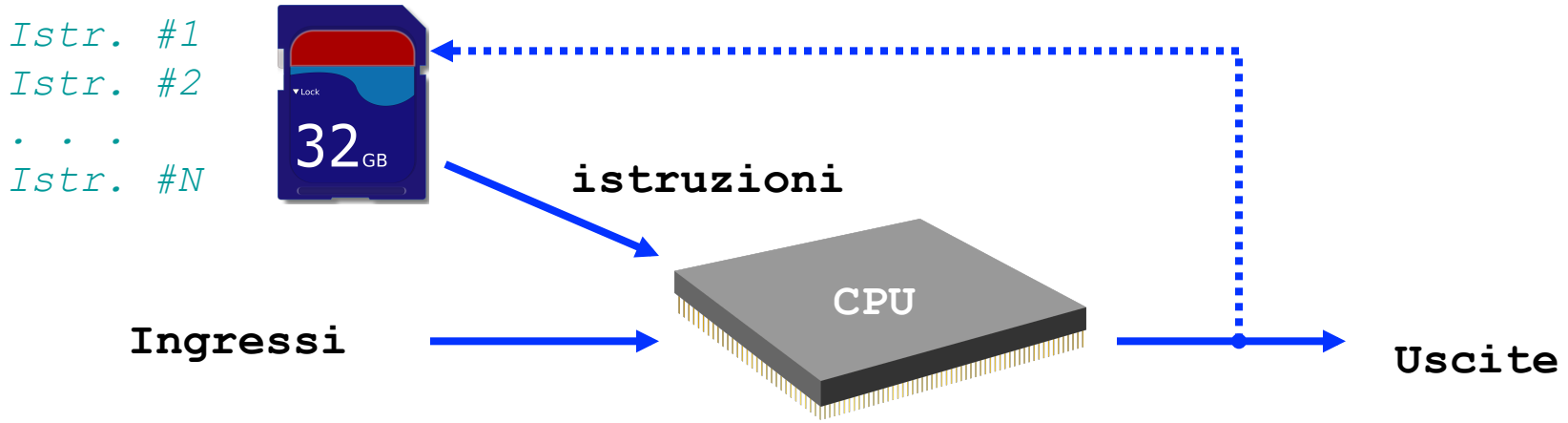


L'unità di controllo non solo governa la rete combinatoria ma anche tutte le altre reti logiche presenti nel sistema

Esempio: abilita gli ingressi e le uscite quando necessario, etc

Modello di esecuzione del programma

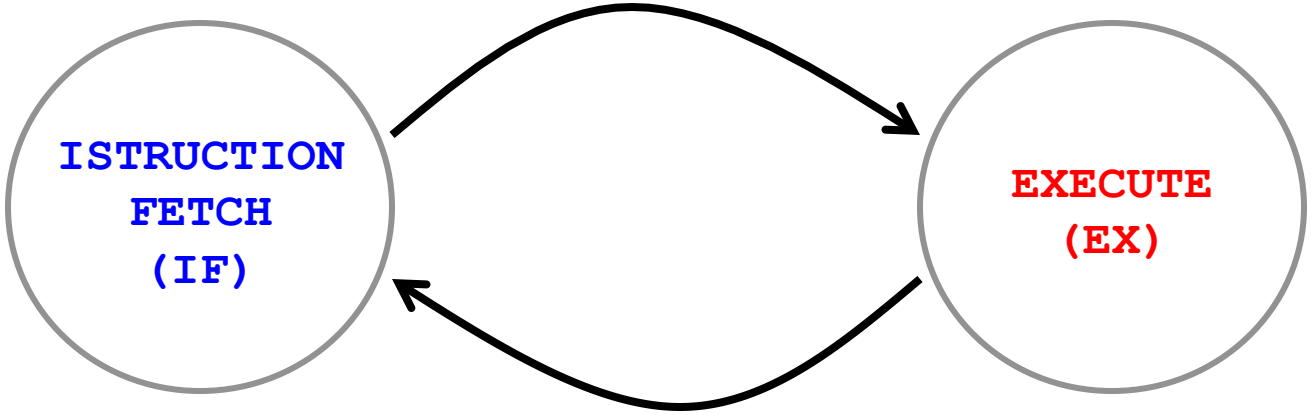
- **Il programma risiede in memoria ed è costituito da istruzioni codificate in forma binaria (linguaggio macchina)**
- **In memoria risiedono anche gli operandi delle istruzioni, cioè i dati elaborati e da elaborare (forma binaria)**
- **Le istruzioni vengono eseguite in sequenza dalla CPU**
- **La CPU è una macchina sequenziale sincrona (con un clock)**



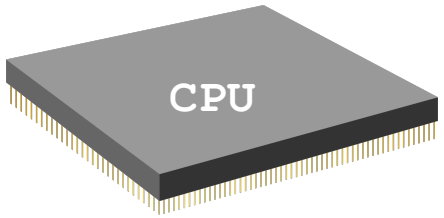
A livello di massima astrazione il funzionamento dell'intero sistema può essere descritto mediante due soli stati:

- **Stato in cui la CPU legge in memoria la prossima istruzione da eseguire (INSTRUCTION FETCH o IF)**

- **Stato in cui la CPU esegue l'istruzione letta in IF (EXECUTE o EX)**



Istr. #1
Istr. #2
. . .
Istr. #N



Rete logica CPU e istruzioni

- **Le reti logiche elaborano e generano segnali binari**
- **Se la CPU è una RL, come sono codificate le istruzioni?**
- **Ovviamente in binario...**
- **Esempio**

```
istruzione #1    -> 00010100000101111101010000010011
istruzione #2    -> 10110101100100011001010000011001
   . . .                          . . .
istruzione #N    -> 01010110100101010101010110011110
```

- **Ogni istruzione indica alla CPU quale operazione deve svolgere/eseguire**
- **Non sembra essere un modo molto comodo (per gli esseri umani)**

Esempi di istruzioni

- Quali istruzioni possono essere eseguite da una CPU?
  - Somme
  - Moltiplicazioni
  - Divisioni
  - Confronto
  - Letture dalla memoria o da altri dispositivi*
  - Scritture in memoria o verso altri dispositivi*
  - . . .

Esistono sostanzialmente due tipologie di CPU:

- RISC (Reduced Intruction Set Computer)
  Poche e semplici istruzioni, reti logiche semplici e molto veloci (frequenza di clock elevata). E.g., ARM

- CISC (Complex Intruction Set Computer)
  Molte istruzioni, alcune molto complesse, reti logiche complicate. E.g., Intel e AMD

RISC vs CISC

- **Spesso a una sola istruzione CISC corrispondono più istruzioni RISC**
- **Tuttavia, ogni singola istruzione RISC è eseguita spesso più rapidamente di una istruzione CISC**
- **Spesso, il codice RISC, anche se più *denso*, è più veloce**
- **Le RL RISC sono tipicamente più semplici di quelle CISC**
- **Se le reti sono più semplici lo spazio (silicio) può essere utilizzato per altre finalità (registri, cache, etc)**
- **I processori RISC sono molto diffusi (smartphone)**
- **Anche i processori CISC sono molto diffusi (PC) per via del software esitente***
- **I processori CISC moderni internamente sono RISC**

Istruzioni in forma più comprensibile

- **Indipendentemente dal tipo di CPU, le istruzioni in forma binaria non sono facilmente interpretabili e per questo non utilizzate in questa forma quando si scrive il codice**

- **Il linguaggio che si utilizza è l'assembler:**

  ADD R1,R2,R3; pone in R1 la somma tra registri R2 e R3

  Questa istruzione potrebbe essere codificata con:

  00010100000101111101010000010011

  Il traduttore assembler -> codice macchina in binario è una Look Up Table (LUT), ovvero una tabella

  L'assembler sembra essere un passo avanti notevole ma...

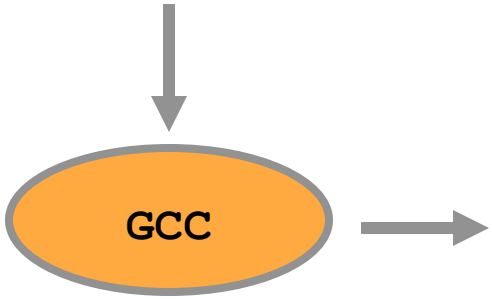
  Perché non avete mai utilizzato il linguaggio assembler nonostante scrivete codice dal primo anno?

Compilatore e istruzioni binarie

- **Il motivo è che avete scritto codice ad alto livello (C) e utilizzato un compilatore (e.g., GCC)**

- **Il compilatore, converte il codice scritto in linguaggio ad alto livello in istruzioni macchina binarie**

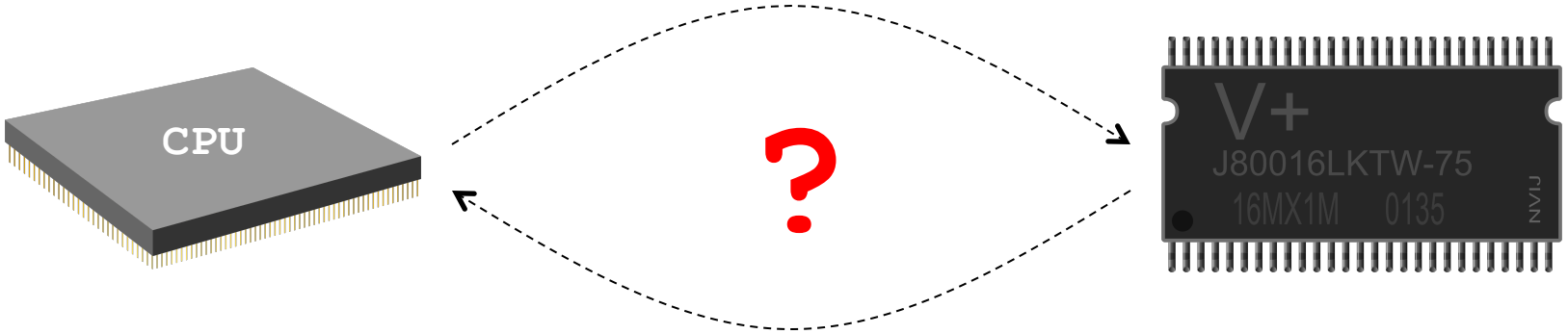
```
#include <stdio.h>
int main(int argc, char **argv)
{
   int a=5;
   int b=6;
   int c;

   c = a + b;

   printf("La somma tra %d e %d è %d\n", a, b, c);
}
```



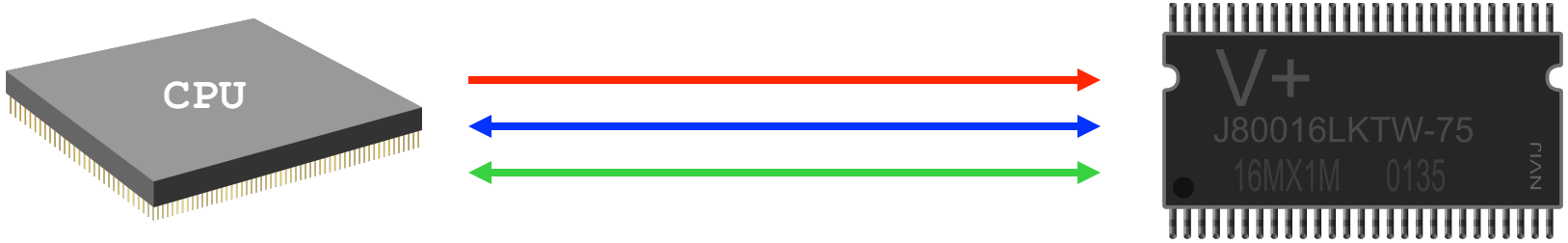

```
00010100000101111101010000010011
10110101100100011001010000011001
 . . .
01010110100101010101010110011110
```

Come si accede alla memoria (e non solo)?

- Sappiamo che il codice nel modello di Von Neumann risiede in memoria
- Come si legge (e scrive dalla memoria)?
- Come si leggono e scrivono i dati?



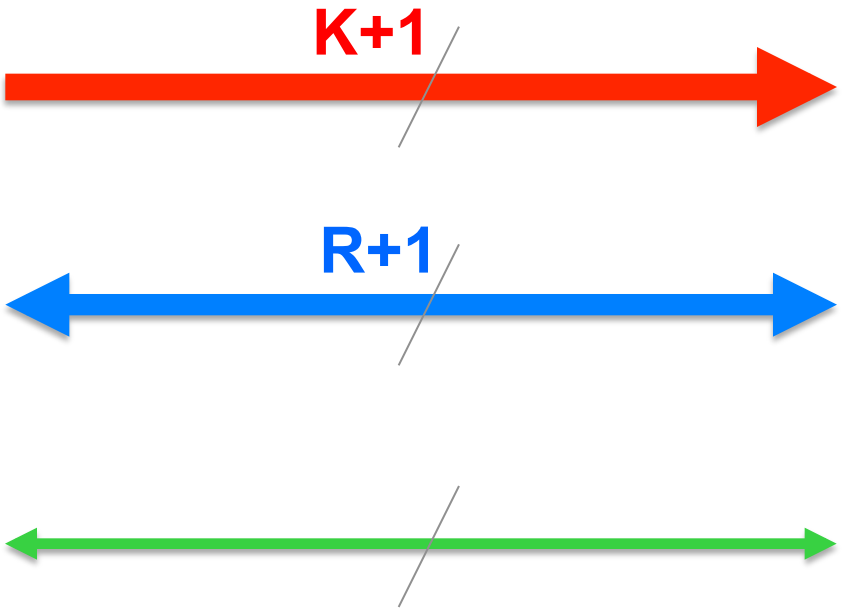
- L'unico modo è attraverso dei segnali predefiniti



Indirizzi BA[K..0]

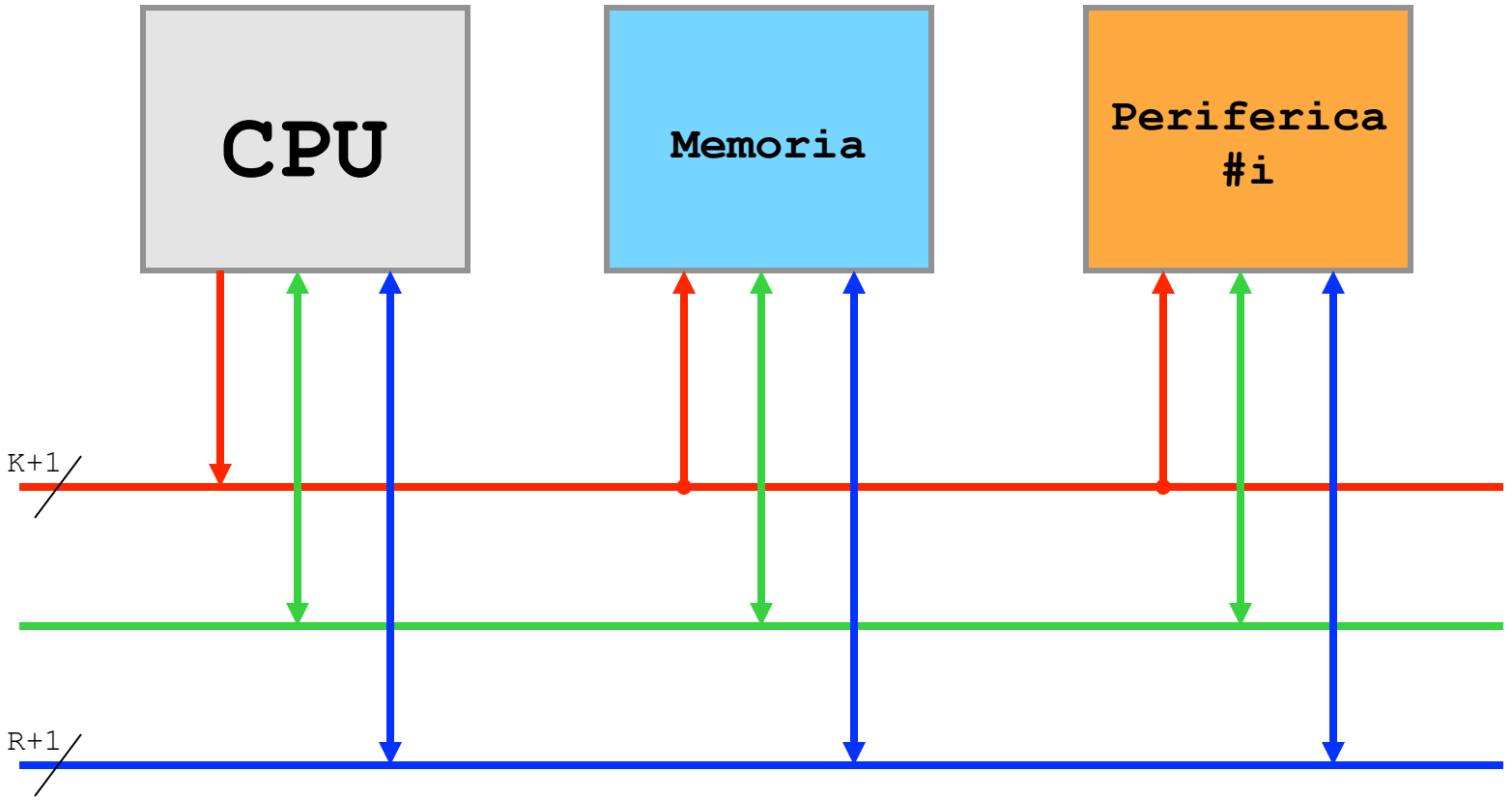
Dati BD[R..0]

Controllo READ, WRITE, …

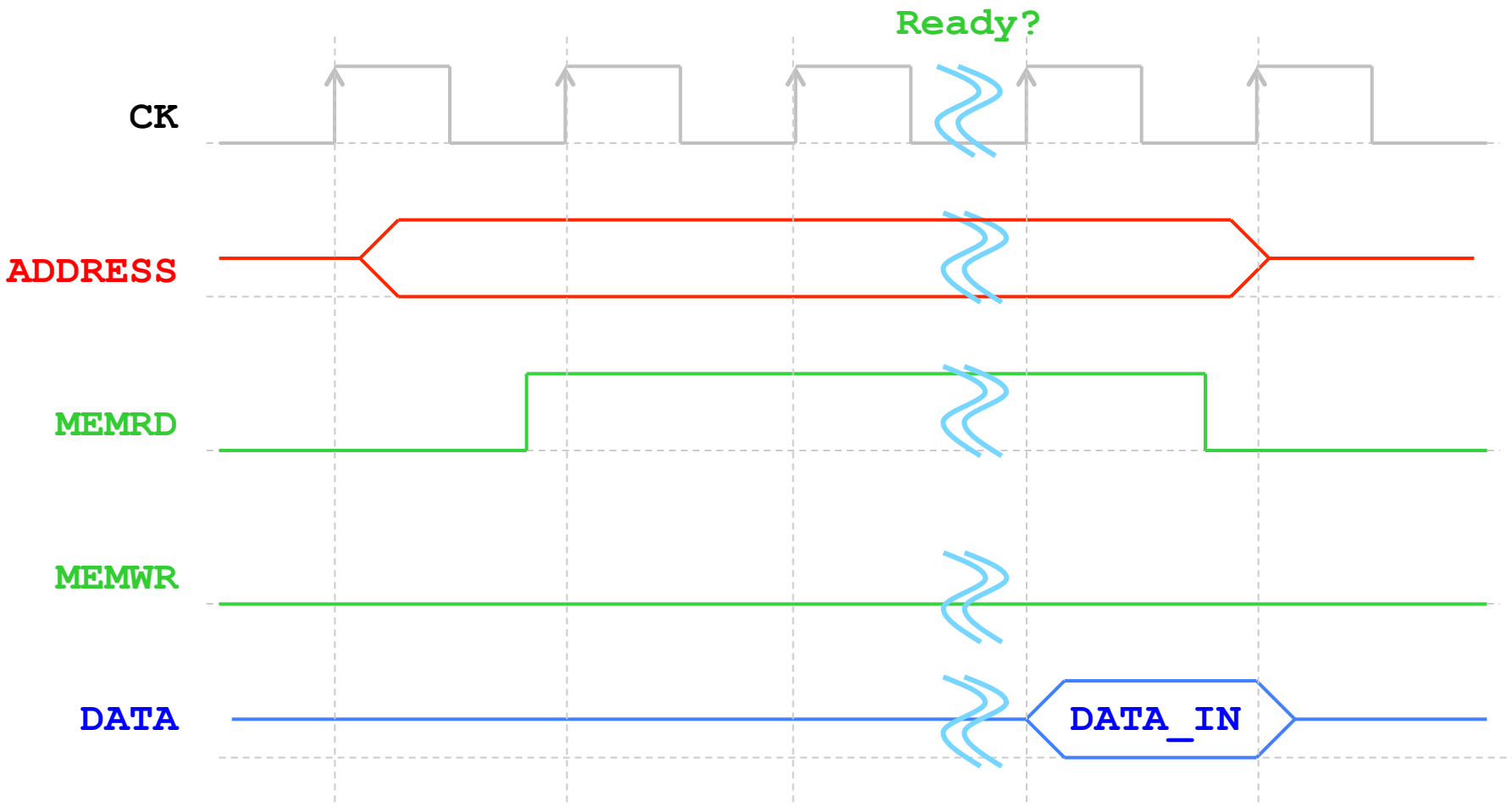


Come avviene la comunicazione?

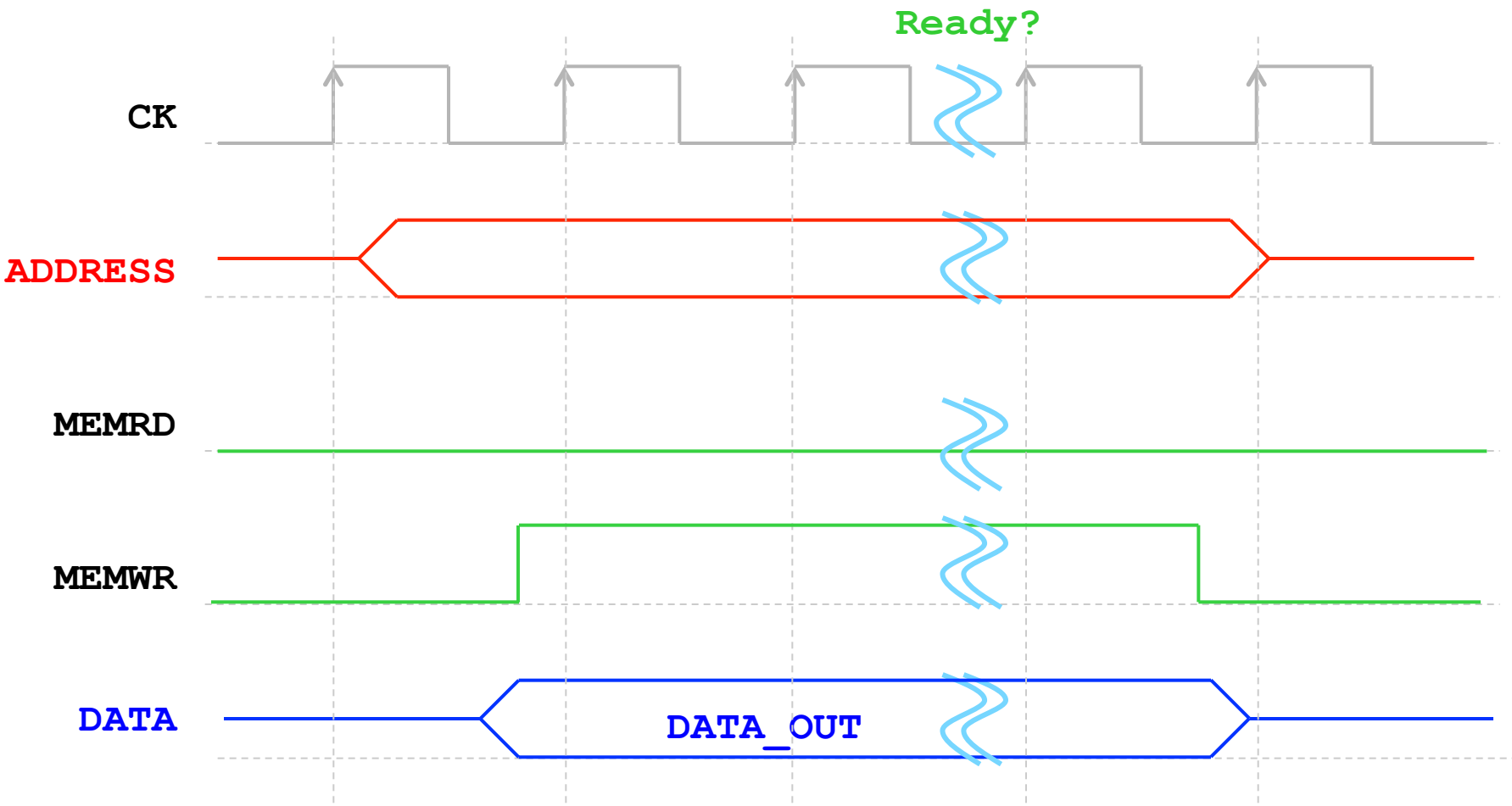
- **Tutto viaggia sui bus di sistema**
- **Tutto è regolato da cicli di bus**



Esempio di ciclo di lettura



Esempio di ciclo di scrittura



Range di indirizzi

- Se sono disponibili 32 bit (BA[31..0]) è possibile avere accesso a 2^32 elementi (memorie, periferiche)
- 32 segnali di indirizzo, 4 GB
- A ogni indirizzo è associato un dispositivo (memorie, periferiche)
- E' necessario decodificare l'indirizzo emesso dalla CPU per determinare con quale dispositivo la CPU intende comunicare
- Quanti byte possono essere trasferiti?
  Dipende dall'ampiezza del bus dati

Relazione tra hardware e software

- **Consideriamo una istruzione per leggere dalla memoria un byte a un determinato indirizzo**

```
xxx    LB destinazione, indirizzo ; lettura di un byte
```

Per prima cosa è eseguito il fetch dell'istruzione all'indirizzo xxx. come? Con un ciclo di bus, naturalmente

Come facciamo a conoscere l'indirizzo xxx?
La CPU ha accesso al program counter PC

Una volta letta e decodificata, l'istruzione è eseguita.

Durante l'esecuzione è eseguito un ciclo di bus di lettura

Durante il ciclo di bus è emesso l'indirizzo