

I sistemi di elaborazione (Seconda parte)

Fondamenti di Informatica A

Percorso di Preparazione agli Studi di Ingegneria

Università degli Studi di Brescia

Docente: Massimiliano Giacomini

Offerta speciale!

PC Desktop

Intel Pentium 4 524 3.06 GHz

RAM 2048 MB

HD 250 GB

DVD+CDRW

Scheda audio on board

Sistema Operativo Windows
Vista

Monitor LCD 19"

699 E

799 E

Notebook

Intel Pentium M 740 1.73 GHz

2 MB L2 RAM 1024 MB

Display 15,4" TFT WXGA TrueBrite

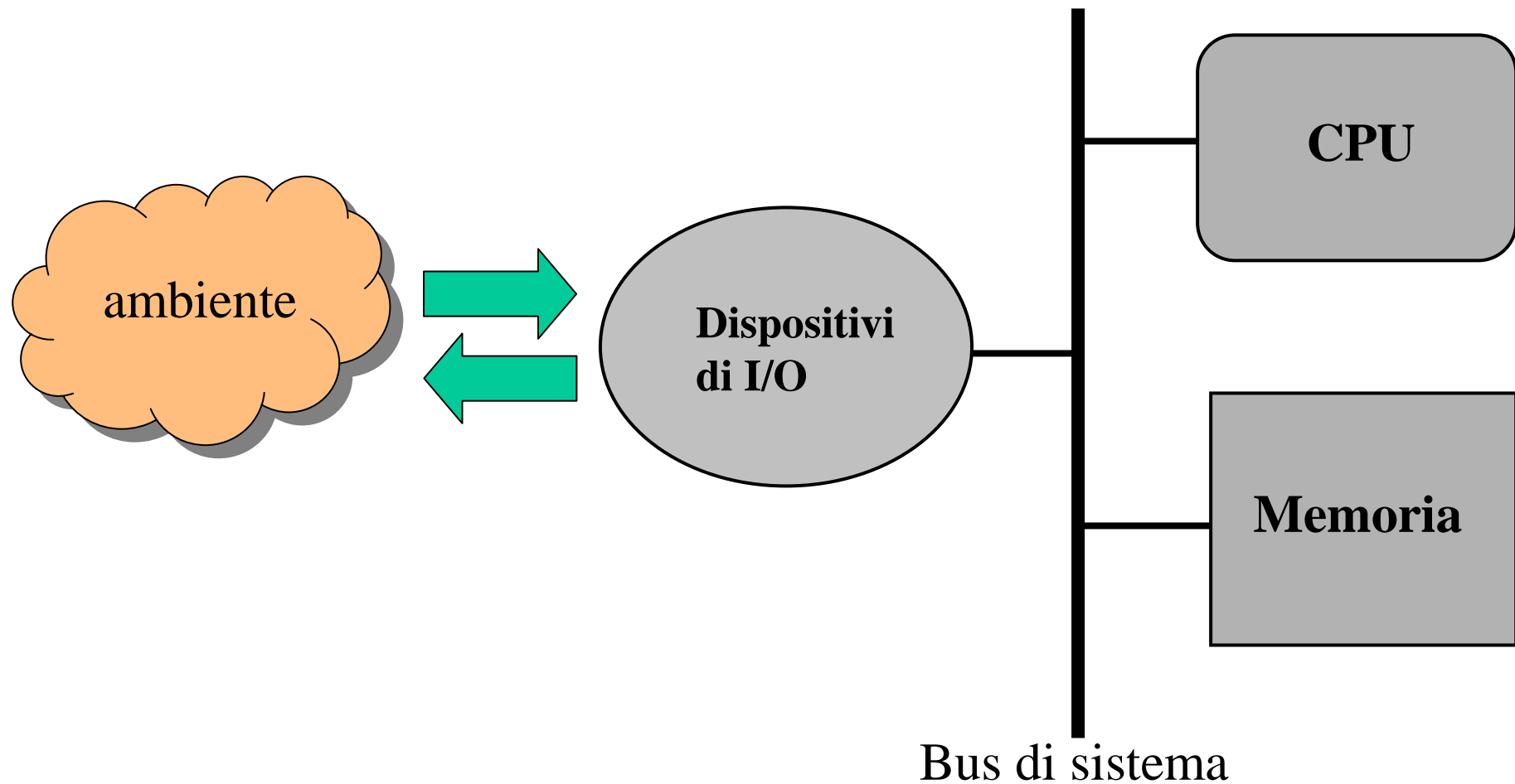
HD 100 GB

DVD/CD-RW

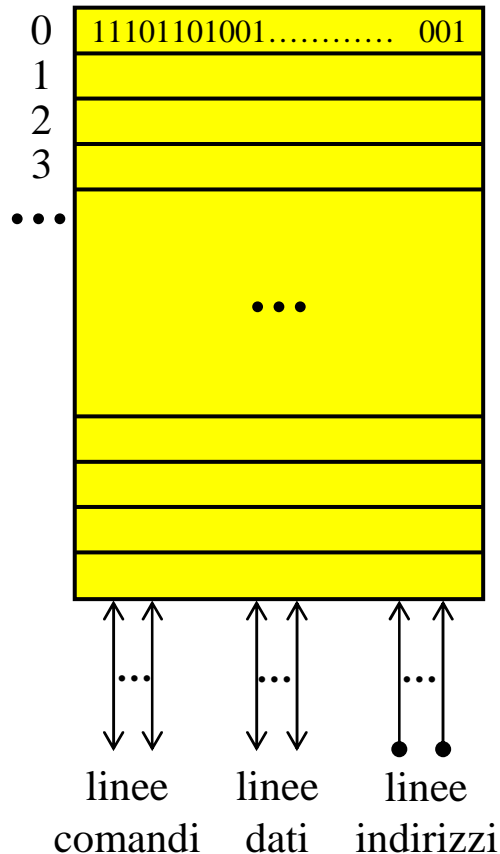
LAN 10/100 - Modem 56k - Wi-Fi
802.11b/g

3 USB 2.0 - IEEE 1394

L'architettura di Von Neumann



LA MEMORIA CENTRALE



- Un insieme di *parole di memoria* consecutive, ciascuna identificata da un *indirizzo*
- Ogni parola memorizza una sequenza di n bit, dove n è lo stesso per tutte le parole e dipende dal calcolatore (es: 16, 32, 64 bit)
- Due operazioni possibili: *lettura* e *scrittura* di una parola all'indirizzo specificato (“cancellazione” non ha senso!)

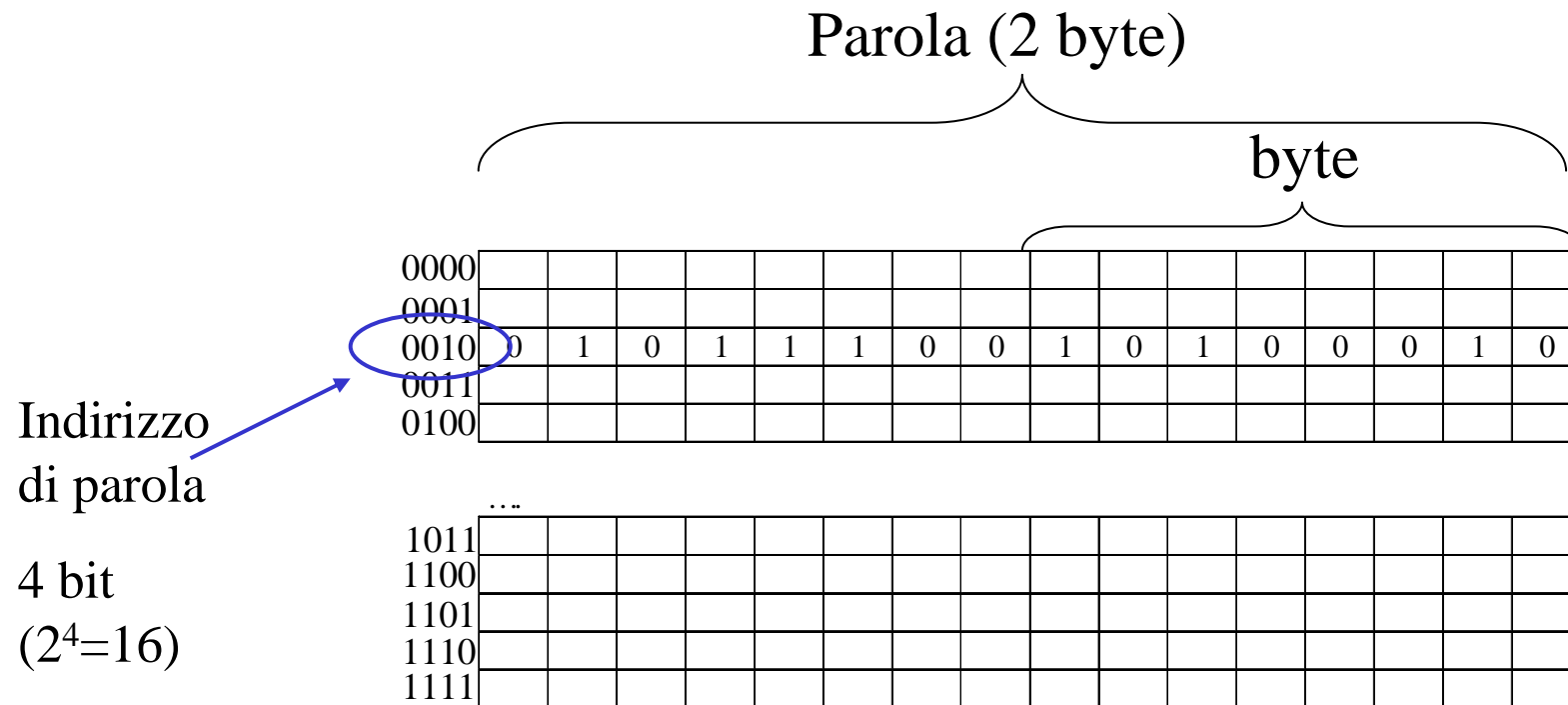
- **Lettura:**

- 1) indirizzo su linee indirizzi
- 2) comando di lettura su linee comandi
- 3) memoria pone dato sulle linee dati

- **Scrittura:**

- indirizzo + dato da scrivere su linee indirizzi e linee dati
- comando di scrittura su linee comandi (al termine, ev.le notifica memoria)

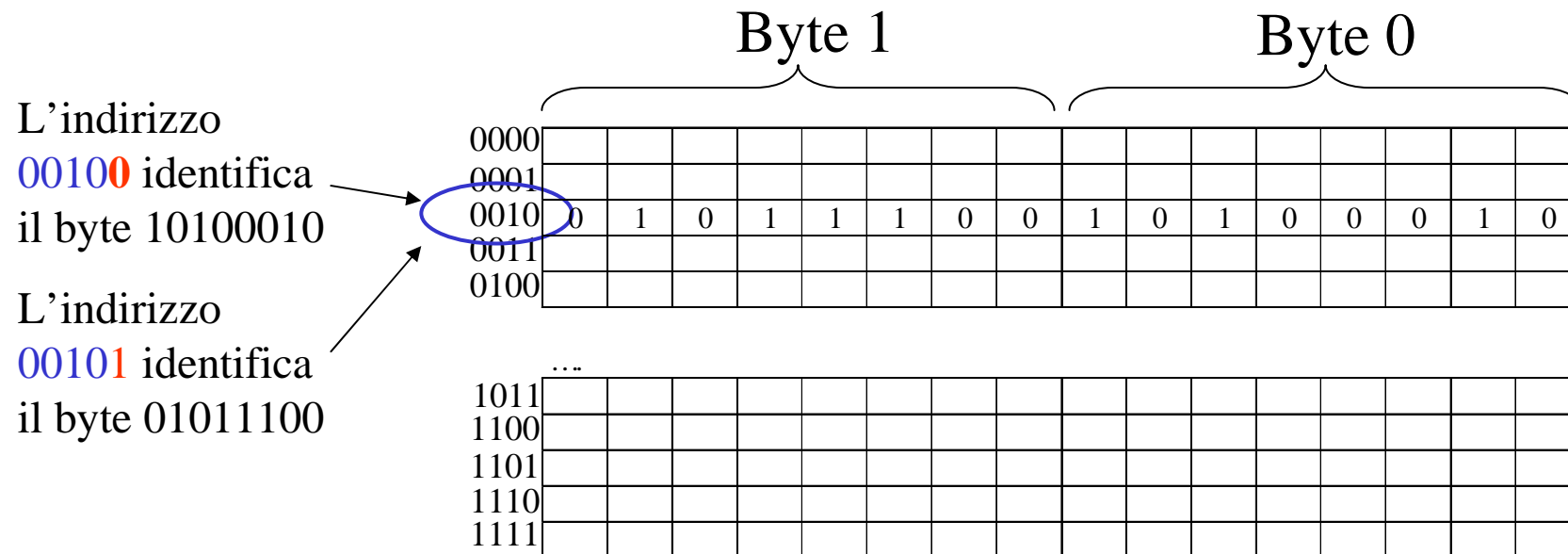
Esempio: memoria di 16 parole



La parola il cui indirizzo è 0010 vale 0101110010100010

- In questo caso quindi si avranno:
 - 16 linee dati (16 bit di parola)
 - 4 linee indirizzi (4 bit per identificare i 16 indirizzi)

- In realtà, le memorie di solito **permettono di indirizzare i byte...**
 \Rightarrow servono più linee indirizzi!
- Nell'esempio precedente, si ha:



- Memoria di 16 parole = 32 byte (ogni parola è di 2 byte)
 \Rightarrow indirizzo di 5 bit ($2^5=32$)

Capacità della memoria

Numero di byte che possono essere memorizzati:
numero di parole x numero di byte per parola

Unità di misura

- 1 byte = 8 bit = 2^3
- Kilo = K = $2^{10} = 1.024$
- Mega = M = $2^{20} = 1.048.576$
- Giga = G = $2^{30} = \dots$
- Tera = T = $2^{40} = \dots$
- NB: quando si parla di memoria, 1 K non è 1000, 1 M non è un milione e 1 G non è un miliardo!!!

Prestazioni

- **Tempo di accesso:**
tempo ingressi stabili – disponibilità dato (lettura)
o completamento memorizzazione (scrittura)
- **Tempo di ciclo:** tempo che intercorre tra un'operazione e la successiva
(tra un'operazione e l'altra può esserci un intervallo)
- **Velocità di trasferimento:** numero di byte che possono essere
trasferiti nell'unità di tempo

Memoria centrale: *memoria ad accesso uniforme - RAM*

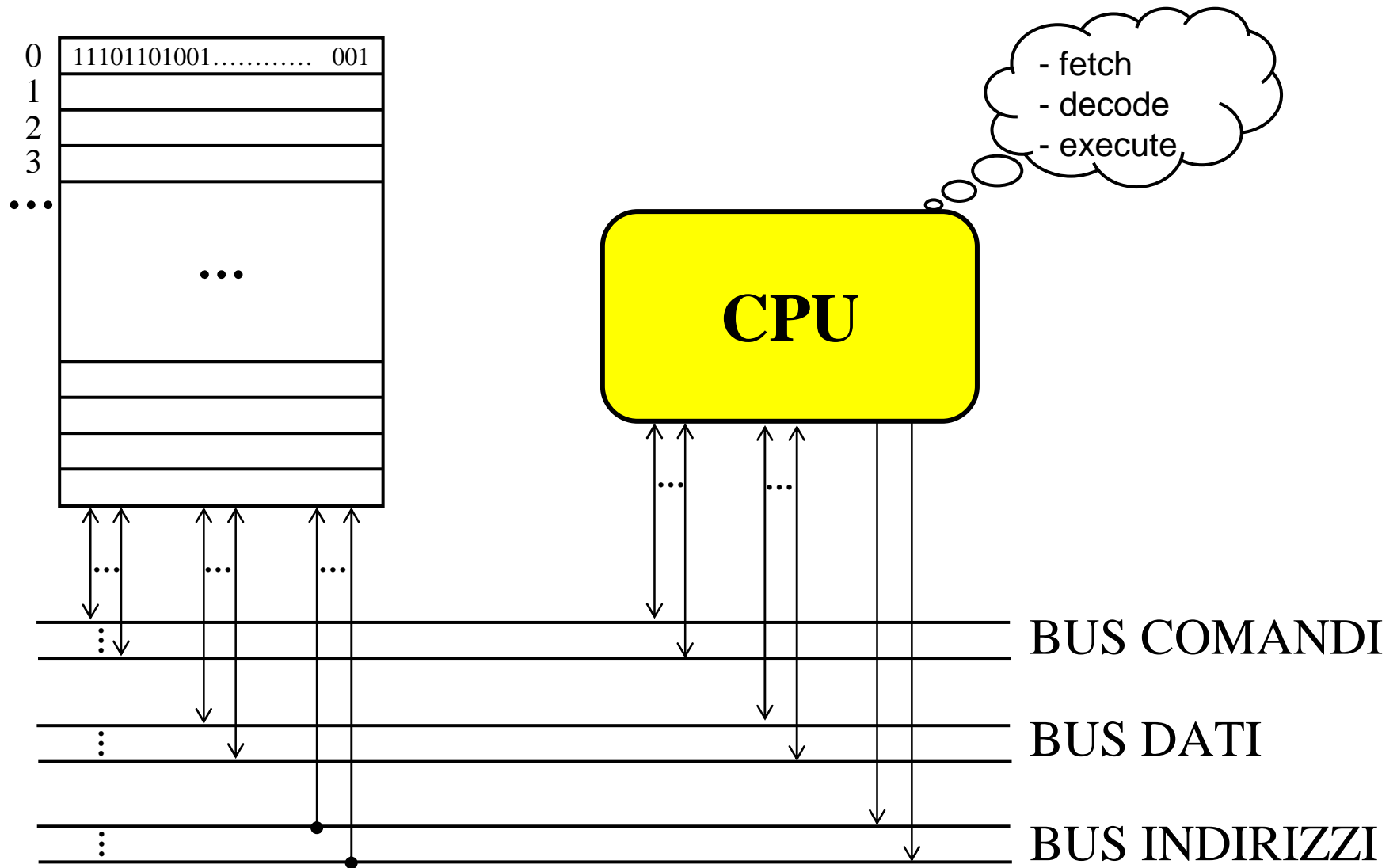
(stesso tempo di accesso e di ciclo per tutte le parole)

Realizzata con tecnologie elettroniche

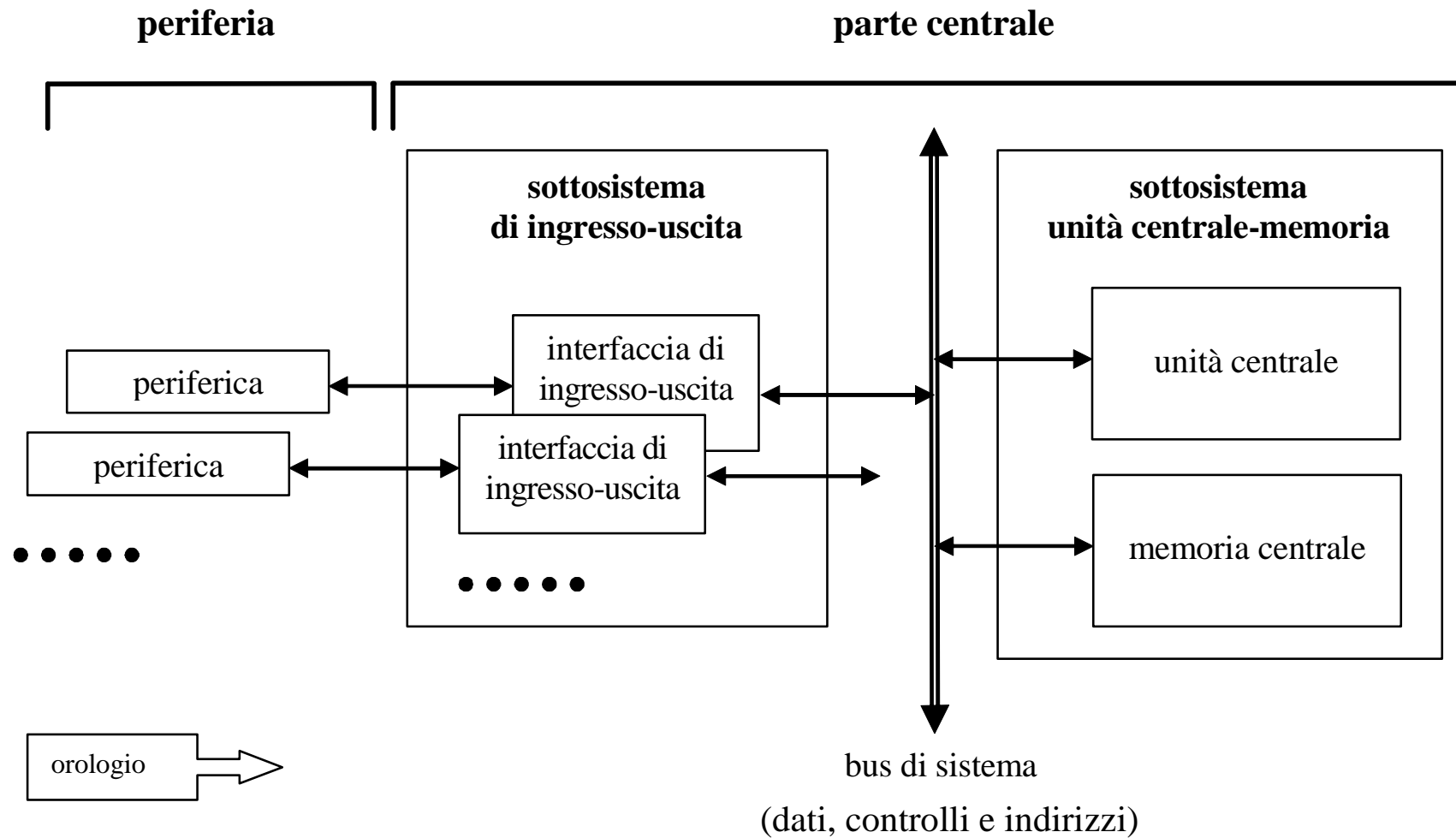
Memoria RAM e memoria ROM

- La memoria centrale è divisa in realtà in due parti:
 - *memoria RAM*: si può leggere e scrivere, **volatile**
 - *memoria ROM*: **memoria a sola lettura, non volatile**
- La memoria ROM include il **BIOS** (operazioni fondamentali, es. gestione periferiche fondamentali, inizializzazione del calcolatore)
- Oggi si usano **memorie flash**: non sono volatili e consentono la lettura e la scrittura (cfr. aggiornamento del BIOS)

L'UNITA' CENTRALE (CPU - processore)



Il quadro completo



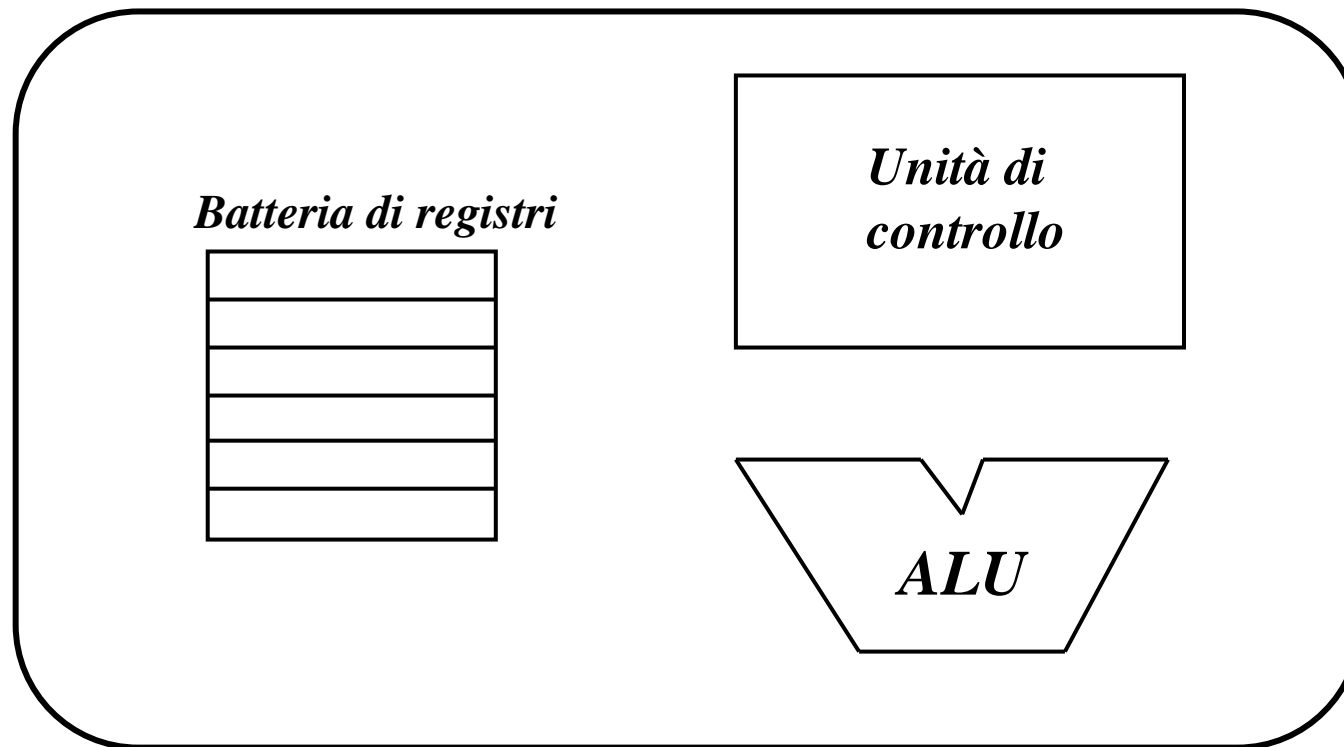
Il bus di sistema

- Insieme di collegamenti (*linee*) che permettono di trasferire dati da più sorgenti a più destinazioni (componenti del calcolatore)
- Il bus può essere suddiviso funzionalmente in:
 - Bus dati (trasferisce i dati scambiati tra componenti)
 - Bus indirizzi (selezionano le parole della memoria o le interfacce di ingresso-uscita coinvolte nel trasferimento)
 - Bus comandi o “di controllo” (segnali di controllo che regolano le operazioni del sistema di elaborazione)

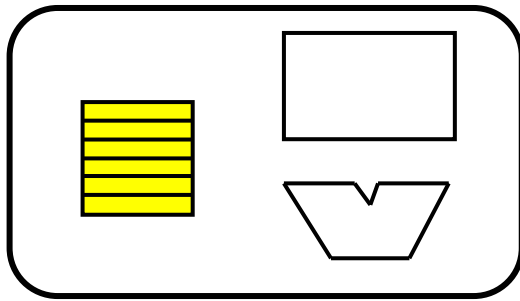
Funzionamento dell'architettura di Von Neumann

- *Dati e istruzioni* del programma da eseguire sono *memorizzati* nella memoria centrale [sistema a *programma memorizzato*]
- L'unità centrale legge dalla memoria centrale le istruzioni e le *esegue in modo sequenziale*, ripetendo ciclicamente i passi:
 1. Reperimento della istruzione da eseguire
 2. Esecuzione dell'istruzione
- *Nell'eseguire le istruzioni, l'unità centrale può leggere e scrivere in memoria* per acquisire gli operandi e per memorizzare i risultati delle istruzioni eseguite!
- Le singole *operazioni* necessarie per l'esecuzione delle istruzioni sono *scandite da un orologio di sistema (clock)* che definisce l'evolvere del tempo all'interno della macchina

Componenti della CPU



Processori commerciali: Intel Pentium, AMD Atlon, Cell, ...



I registri della CPU

- **Celle di memoria**, costituiscono la “memoria a breve termine” del calcolatore:
 - Contengono sequenze di bit di dimensioni limitate (es. 32 bit)
 - Tempo di accesso molto breve
- Utilizzati per immagazzinare informazioni necessarie per eseguire le istruzioni
- Il loro numero e bit dipendono dal tipo di CPU (es. CPU a 32 o 63 bit)
- Si dividono in *registri di uso generale* e *registri speciali*

Registri di uso generale

- Contengono operandi e i risultati delle istruzioni da eseguire
- I campi operando delle istruzioni possono contenere un indirizzo di registro:
 - Es: CPU a 32 registri \Rightarrow 5 bit necessari

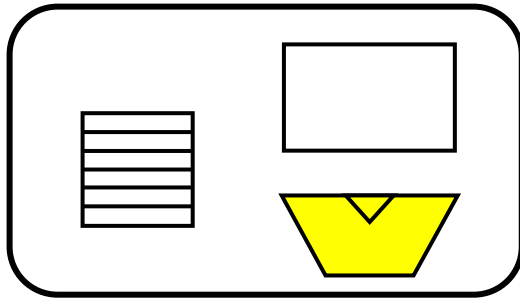
Registri speciali

- *Program counter* (PC), anche chiamato Instruction Pointer:
memorizza l'indirizzo della prossima istruzione da eseguire
(con cui accedere alla memoria)

NB: deve essere incrementato ad ogni istruzione eseguita

- *Instruction register* (IR): memorizza l'istruzione da eseguire
(prelevata dalla memoria)

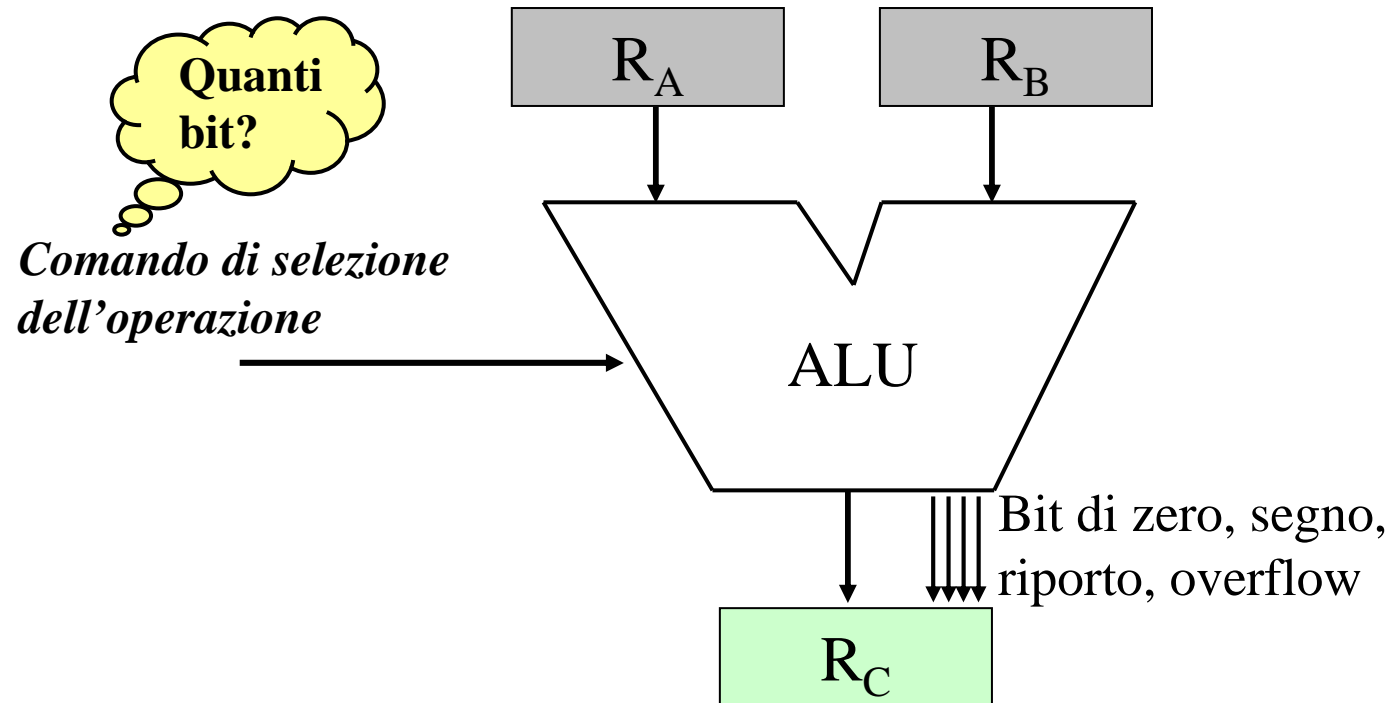
- *Processor Status Word* (PSW), detto anche FLAGS:
contiene informazioni sullo “stato” del processore, p.es.
memorizzando le condizioni che si verificano in seguito
all'esecuzione di una istruzione o le modalità di
funzionamento del processore
- Altri...

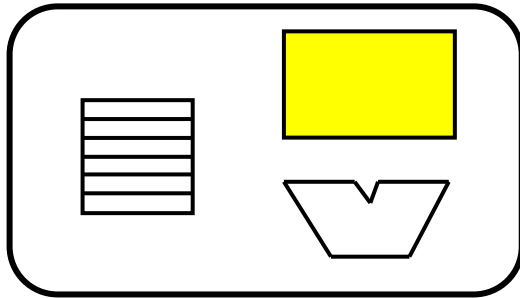


ALU

Unità Aritmetico/Logica

- E' il circuito che svolge le operazioni aritmetiche (+, -, ...) e logiche (and, or, not, ...) sugli operandi forniti in ingresso

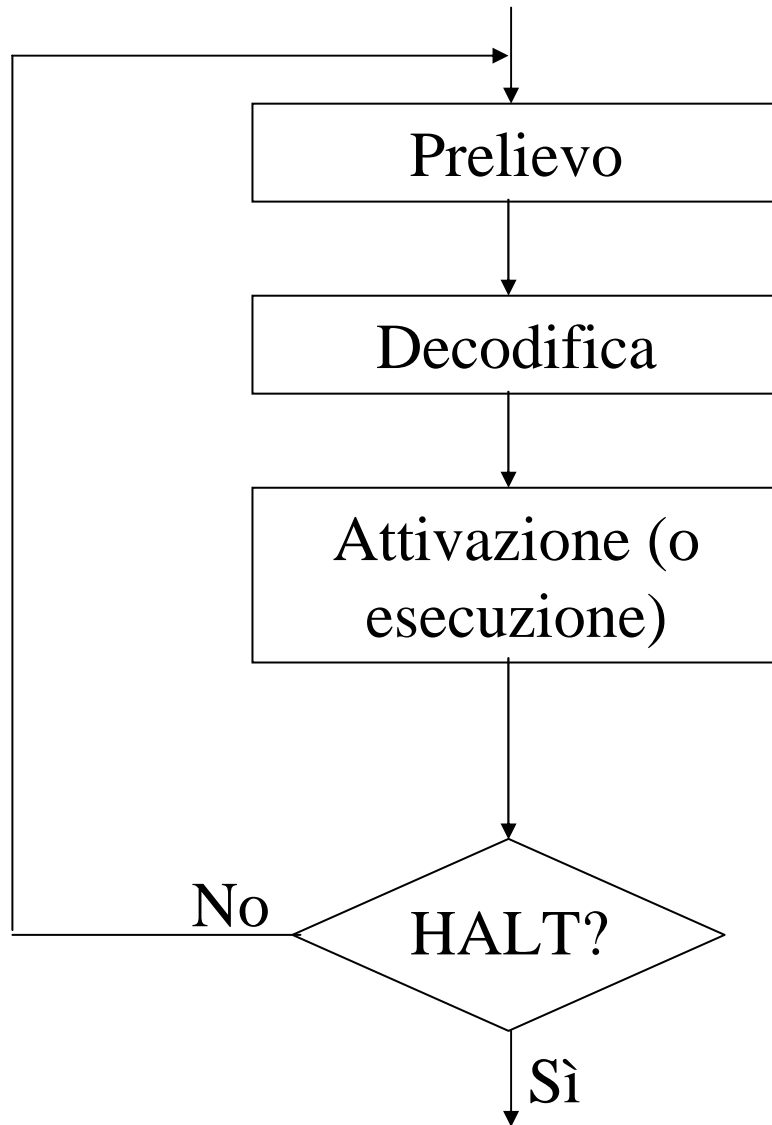




Unità di Controllo

- E' il componente che **coordina** il funzionamento delle varie unità della CPU nell'esecuzione dei programmi
- Esegue il **ciclo macchina** della CPU:
 - **decodifica** le istruzioni in base al loro codice operativo
 - manda attraverso il bus comandi opportuni **comandi** (segnali di controllo) alla ALU (per la selezione dell'operazione), ai registri (per leggere o scrivere valori), alla memoria e alle interfacce di ingresso uscita...

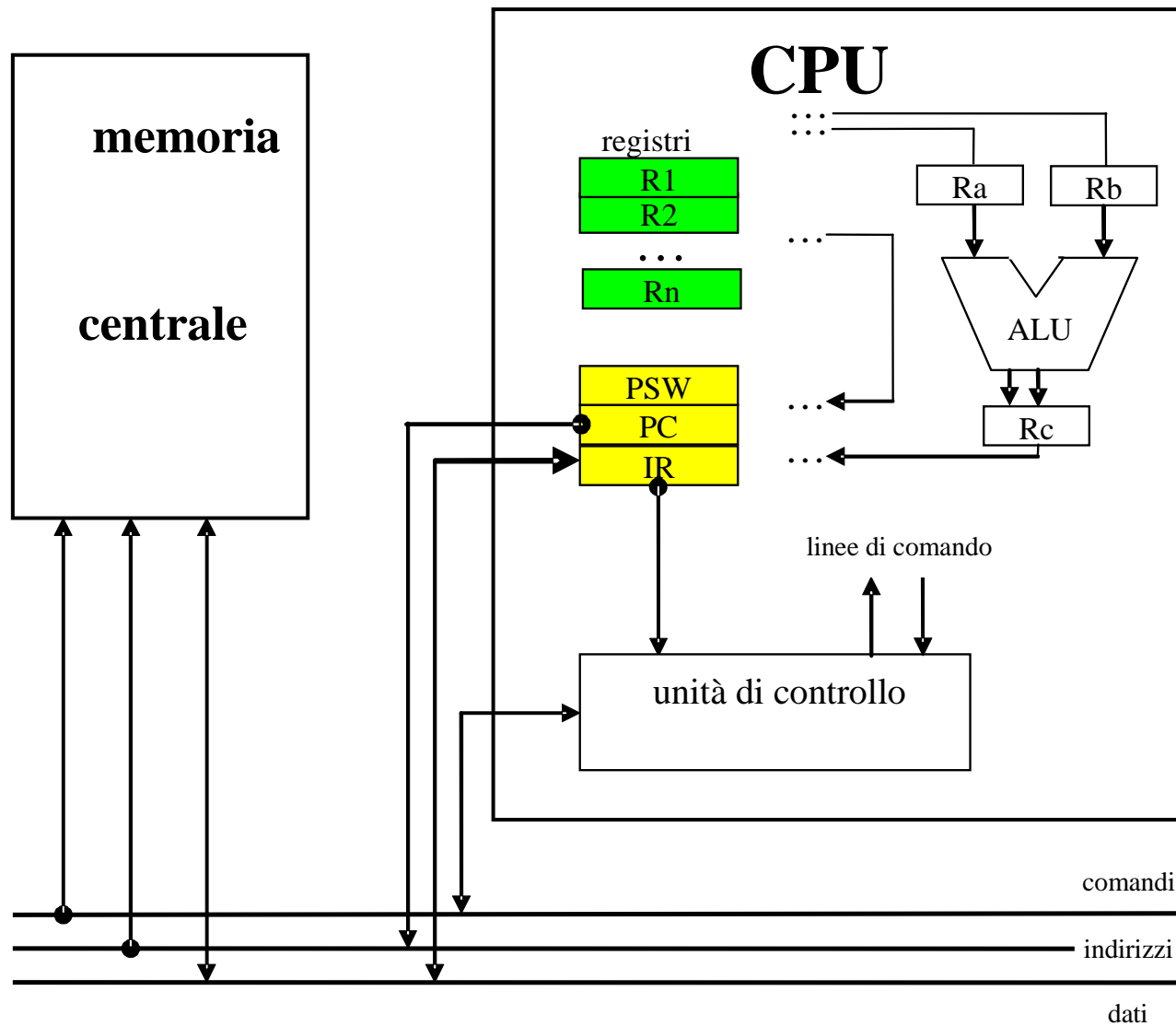
Ciclo macchina



- La CPU può essere intesa come un dispositivo che **opera in modo ciclico**, ripetendo indefinitamente seguenti passi:
 - 1. Prelievo** dell'istruzione (*fetch*)
 - 2. Decodifica** dell'istruzione (*decode*)
 - 3. Attivazione o esecuzione** dell'istruzione (*execute*)

CERCHIAMO DI METTERE TUTTO INSIEME,
CONSIDERANDO UN ESEMPIO...

Un calcolatore elementare



Ciclo macchina (in dettaglio)

PRELIEVO

- L'unità di controllo manda il comando di lettura alla memoria con l'indirizzo in PC della prossima istruzione da eseguire e l'istruzione letta dalla memoria viene copiata in IR:
 - 1) invio sul bus indirizzi del valore presente in PC
 $[PC] \rightarrow \text{bus indirizzi}$
 - 2) invio sul bus di controllo del comando di lettura dalla memoria
 $\dots M([PC]) \rightarrow \text{bus dati}$
 - 3) lettura dal bus dati e memorizzazione dell'istruzione in IR
 $\text{bus dati} \rightarrow \text{IR}$
- Contestualmente, incremento del contenuto di PC
 $[PC] + 1 \rightarrow PC$

DECODIFICA

- Esame dell'istruzione in IR per determinare le operazioni da svolgere
Unità di controllo riceve il codice operativo da IR

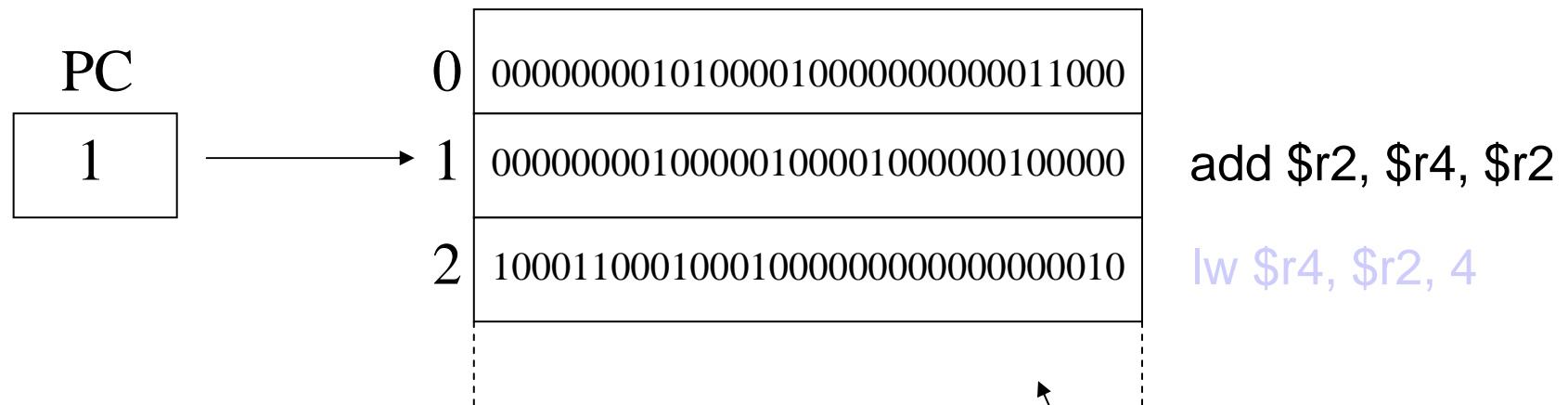
ATTIVAZIONE (o esecuzione)

- Le unità interessate all'esecuzione vengono opportunamente comandate dall'unità di controllo (sulla base del codice operativo)

Il ciclo ricomincia dalla fase di prelievo dell'istruzione successiva

Esempio: esecuzione di una istruzione add

add \$r2, \$r4, \$r2 ➔ somma il contenuto del registro R_4 al contenuto del registro R_2 e memorizza il risultato in R_2



Programma in memoria
in linguaggio macchina


- PASSO 1: Fetch

Carica istruzione in IR (si indica di solito con $IR \leftarrow (PC)$) :

$IR \leftarrow 000000001000001000010000000100000$

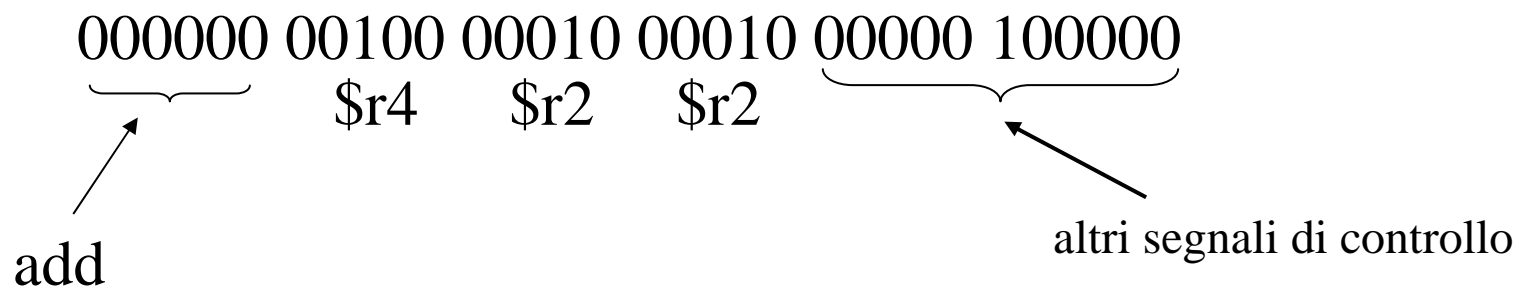
... e aggiorna PC

$PC \leftarrow PC + 1$ (“punta” alla prossima istruzione)

- 
- L'indirizzo dell'istruzione contenuto nel PC viene inviato sul *bus indirizzi*
 - Contemporaneamente viene attivato il segnale ‘leggi’ del *bus di controllo*
 - La CPU incrementa il valore in PC
 - La memoria accede alla cella indirizzata e ne pone il contenuto (la prossima istruzione da eseguire) sul *bus dati*
 - Tale istruzione viene quindi trasferita in IR (U.C. attiva il segnale di scrittura)

- PASSO 2: Decode

Decodifica istruzione in IR



In pratica, l'unità di controllo

- riceve il contenuto di IR
- effettua la decodifica dell'istruzione (dai bit del codice operativo scopre che è una **add**)
- identifica gli operandi (i registri coinvolti nell'operazione)

- PASSO 3 (Execute): può essere suddiviso in diversi passi

- Passo 3.1: Caricamento valori dei registri

$$R_A \leftarrow R_2$$

$$R_B \leftarrow R_4$$

Unità di controllo
manda segnali di
lettura e scrittura

- Passo 4: Somma (operazione con ALU)

$$R_C \leftarrow R_A + R_B$$

Unità di controllo
manda segnali di
lettura e scrittura +
comando ALU per
selezione operazione

- Passo 5: Memorizza risultato in R_2

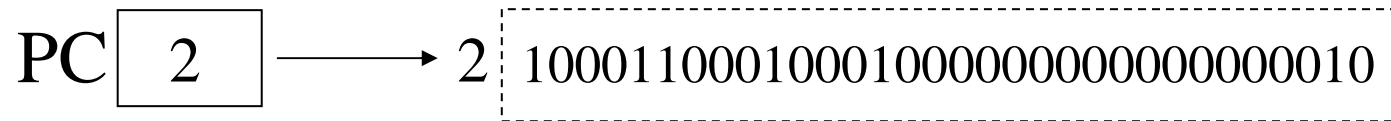
$$R_2 \leftarrow R_C$$

Unità di controllo
manda segnali di
lettura e scrittura

TOTALE = 5 passi

Esempio: esecuzione di una istruzione lw (lettura da memoria)

- Proseguendo il precedente esempio, il PC è stato aggiornato e punta all'istruzione successiva alla add



- Istruzione lw \$r4, \$r2, 4:

carica nel registro R_4 il valore della parola presente all'indirizzo di memoria ottenuto sommando 4 al contenuto in R_2

- Passo 1: Carica istruzione in IR e aggiorna PC:

$$IR \leftarrow (PC)$$

$$PC \leftarrow PC + 1$$

- Passo 2: Decodifica istruzione in IR

$$\begin{array}{cccc} \underbrace{100011} & \underbrace{00010} & \underbrace{00100} & \underbrace{0000000000000000100} \\ lw & \$r2 & \$r4 & offset= 4 \end{array}$$

- Passo 3: Copia 4 e R_2 nei registri usati dalla ALU

$$R_A \leftarrow 4$$

$$R_B \leftarrow R_2$$

- Passo 4: Somma contenuto registri:

$$R_C \leftarrow R_A + R_B$$

- Passo 5: Poni il contenuto di R_C sul bus indirizzi, attiva il segnale di lettura per la memoria e carica il dato disponibile nel bus dati in R_4 :

$$R_4 \leftarrow (R_C)$$

TOTALE = 5 passi

Esempio: esecuzione di una istruzione beq (salto condizionato)

- Istruzione `beq $r1, $r2, Alfa`

Se il contenuto di `$r1` è uguale al contenuto di `$r2`
allora salta all'istruzione all'indirizzo Alfa

- Passo 1: Carica istruzione in IR e aggiorna PC

$$IR \leftarrow (PC)$$

$$PC \leftarrow PC + 1$$

- Passo 2: Decodifica istruzione in IR
- Passo 3: Copia R_1 e R_2 nei registri usati dalla ALU

$$R_A \leftarrow R_1$$

$$R_B \leftarrow R_2$$

- Passo 4: Sottrai il contenuto di R_B dal contenuto di R_A
$$R_C \leftarrow R_A - R_B \quad \text{e bit di zero}$$
- Passo 5: Poni il bit di zero in PSW
- Passo 6: Se $R_A - R_B = 0$ (bit di zero asserito) copia il valore dell'indirizzo Alfa in PC

TOTALE = 6 passi

APPROFONDIMENTI:

- come riesce l'unità di controllo durante un passo a “collegare” correttamente sorgenti, ALU e destinazione?

Uso di “selettori” (multiplexer)

- come avviene la sincronizzazione tra la lettura di uno o più registri e la scrittura su un registro destinazione in un passo?

Comandi di scrittura (e lettura) ai registri +

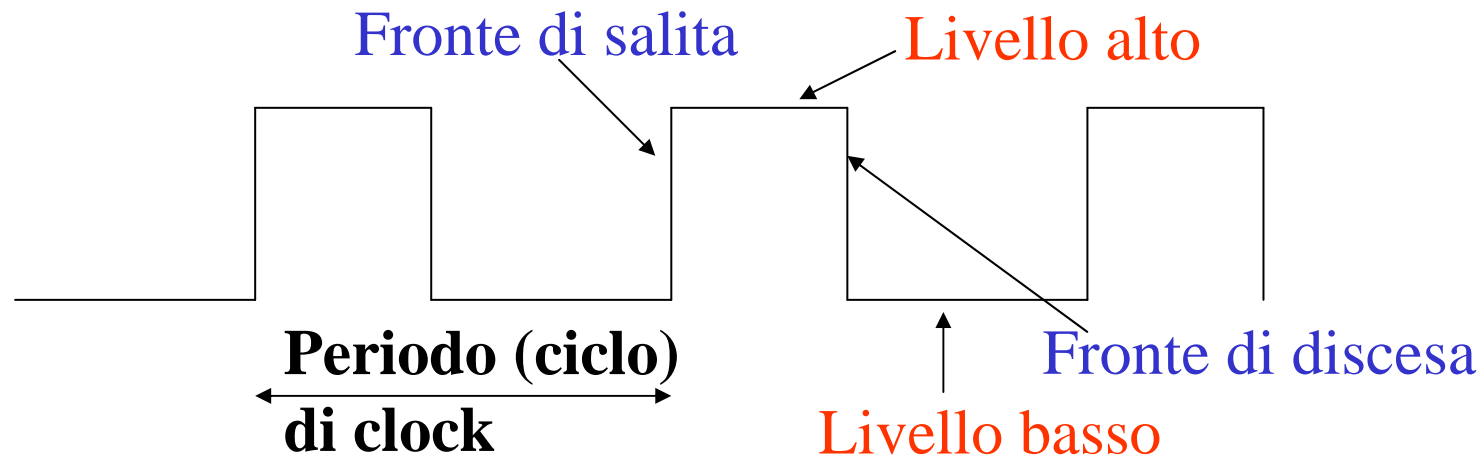
Segnale di clock (a tutti i registri) – vedi prossimi lucidi

- come fa l'unità di controllo a produrre “nei diversi passi” i comandi opportuni per l'esecuzione di una istruzione macchina?

Macchina “a stati finiti”...

Il segnale di clock

- Le varie unità della CPU operano in modo coordinato
- Un **orologio** generatore di impulsi elettrici (**clock**) fornisce una cadenza temporale a cui tutte le attività elementari della CPU sono sincronizzate
- Un segnale di clock evolve con un **periodo** (tempo di ciclo) predeterminato e costante (periodo = intervallo di tempo fra 2 impulsi di clock consecutivi). La **frequenza di clock** è l'inverso del periodo (numero di impulsi – cicli - di clock al secondo)

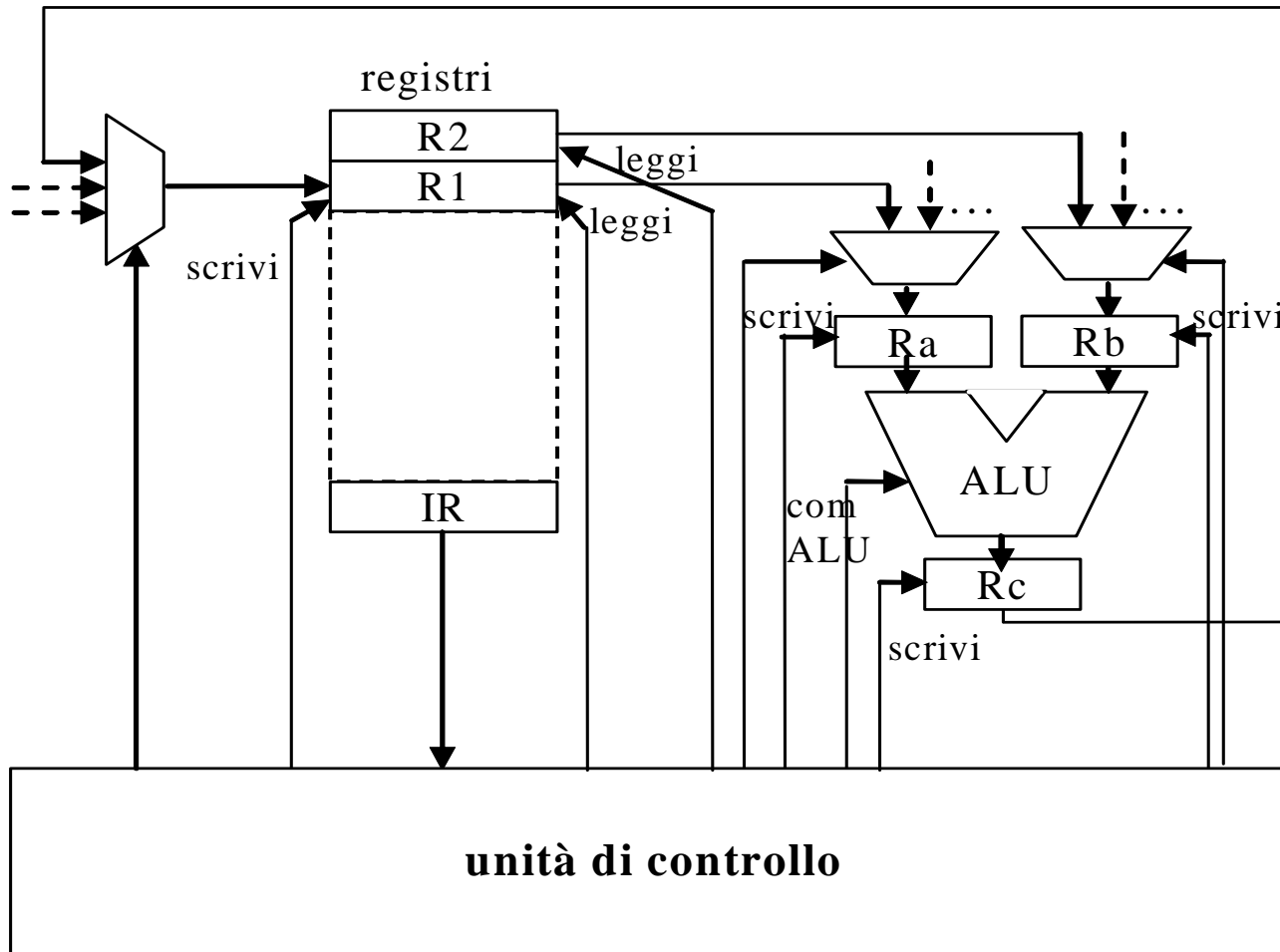


- Frequenza di clock: misurata in **Hertz** (**1Hz** = 1 ciclo/secondo)
- Periodo di clock: normalmente misurato in **ns** (**nanosecondi**, 10^{-9} secondi)
- Esempio:
 - clock con frequenza 2Ghz: 2 miliardi di impulsi al secondo,
periodo = $1/(2 \times 10^9) = 0,5 \times 10^{-9} = 0,5 \text{ ns}$
- *I passi per compiere le operazioni viste sono in genere eseguiti in cicli di clock successivi:*
 - Es. 5 passi (5 cicli di clock). Se un ciclo è di 0.5 ns, occorrono 2.5 ns per svolgere una istruzione di addizione
- il numero di passi è diverso da istruzione a istruzione e dipende da come è fatta la specifica CPU!

Vediamo un esempio:

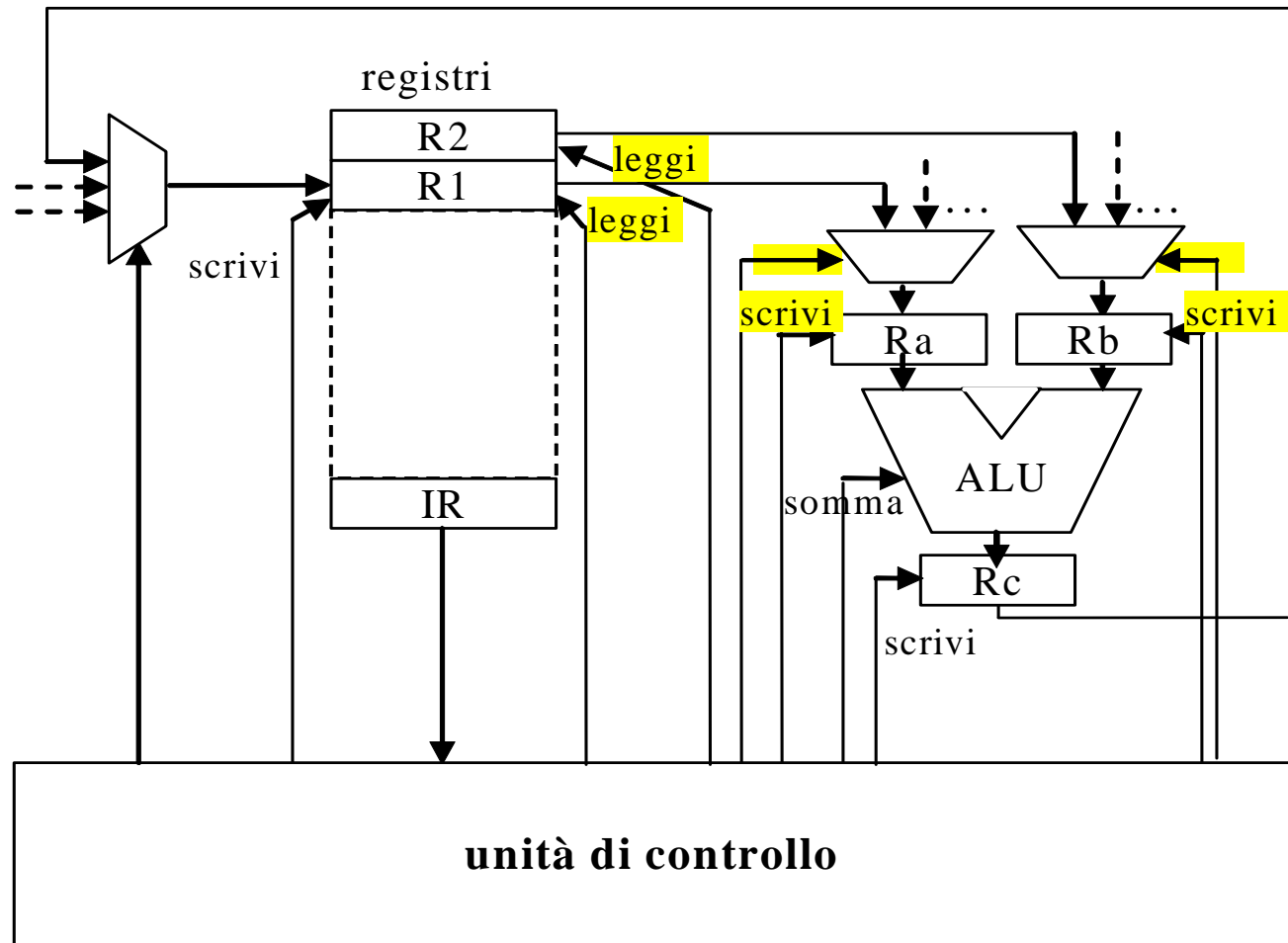
l'esecuzione della somma `add $r1, $r1, $r2`

(solo la fase di attivazione, che impiega 3 passi)

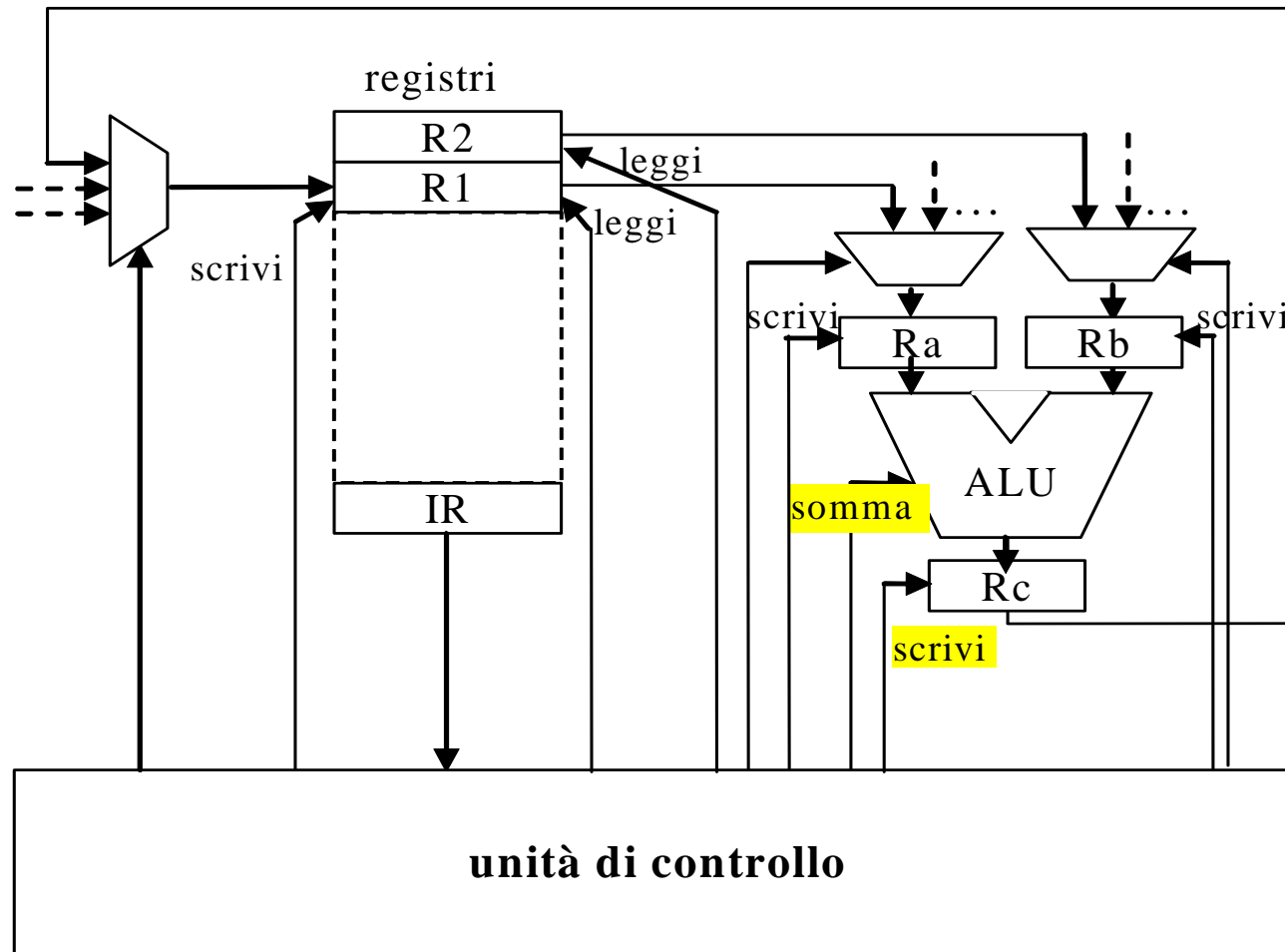


NB: non è indicato
lo schema
completo

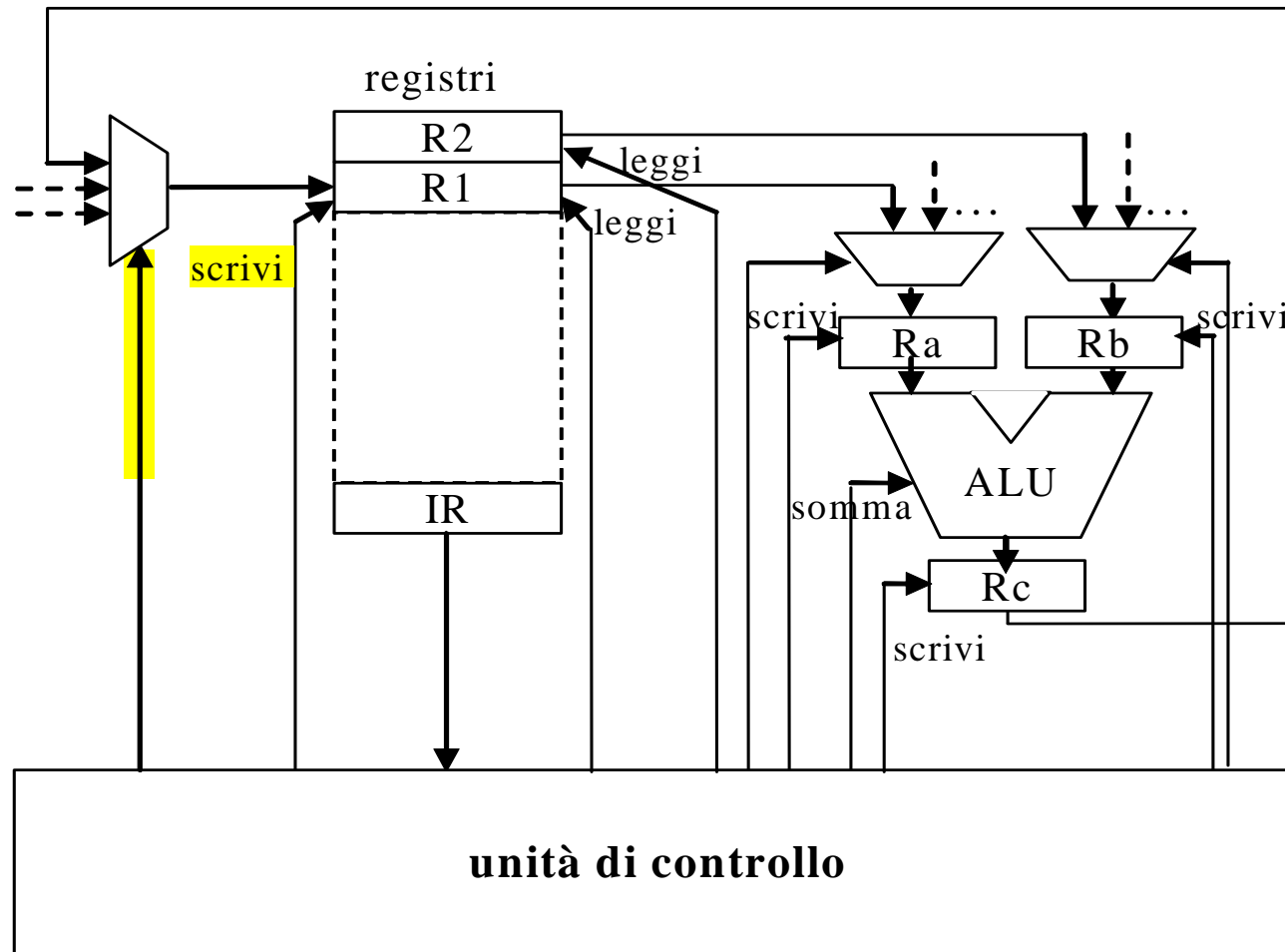
PASSO 3



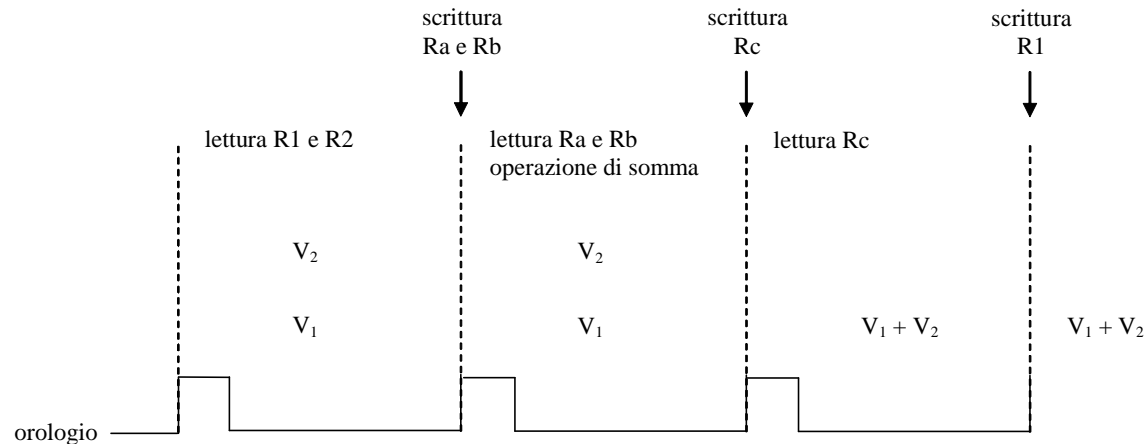
PASSO 4



PASSO 5



- NB: temporizzazione ai passi 3, 4 e 5



Per ogni passo un ciclo di clock

- Unità di controllo come macchina a stati finiti:
 - durante un ciclo: stato corrente
 - uscite (comandi): funzione dello stato corrente
 - transizione di stato: funzione stato corrente e ingressi U.C.

Realizzazione: cablata (hardwired) vs. microprogrammata

Prestazioni della CPU

- Matematicamente:

$$\begin{aligned}T_{\text{CPU}} &= n_{\text{istr}} * C_{\text{medio}} * T_{\text{clock}} \\ &= (n_{\text{istr}} * C_{\text{medio}}) \setminus f_{\text{clock}}\end{aligned}$$

- Ovviamente, maggiore è la frequenza meglio è, ma non è tutto...
Due filosofie: **CISC** (Complex Instruction Set Computer) vs
RISC (Reduced Instruction Set Computer)
- In generale, le prestazioni dipendono dallo specifico programma (o tipologia di programma) eseguito - cfr. benchmark
- E, per valutare le prestazioni di tutto il calcolatore, non basta considerare la CPU (es: memorie, periferiche, bus, ecc.)