

L'architettura del calcolatore (Seconda parte)

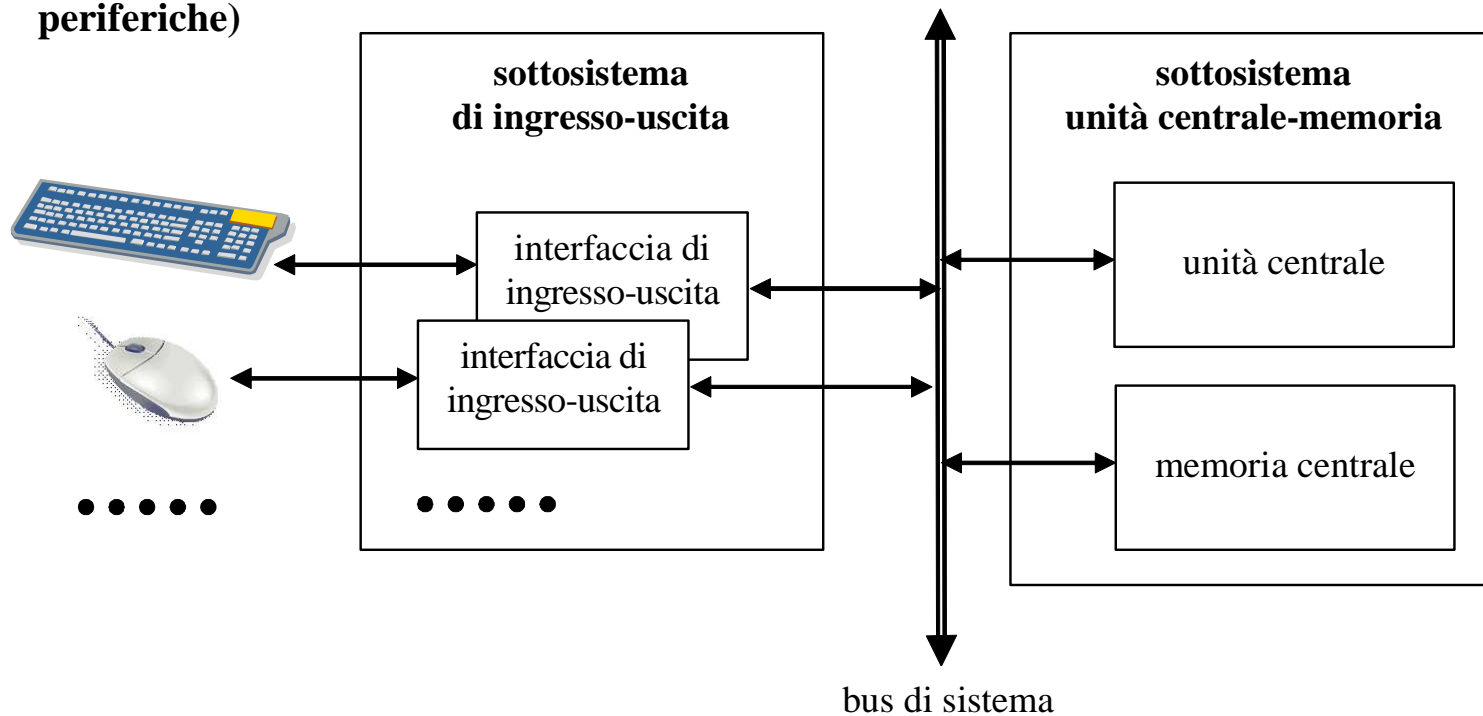
Percorso di Preparazione agli Studi di Ingegneria

Università degli Studi di Brescia

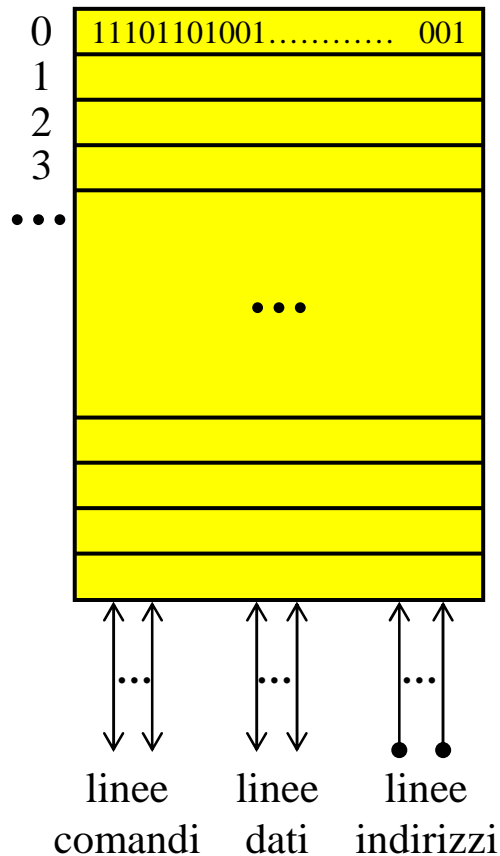
Docente: Massimiliano Giacomini

ORGANIZZAZIONE DEL CALCOLATORE: RICHIAMI

**Periferia
(insieme delle
periferiche)**



LA MEMORIA CENTRALE: RICHIAMI



- 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!)

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)

Capacità della memoria

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

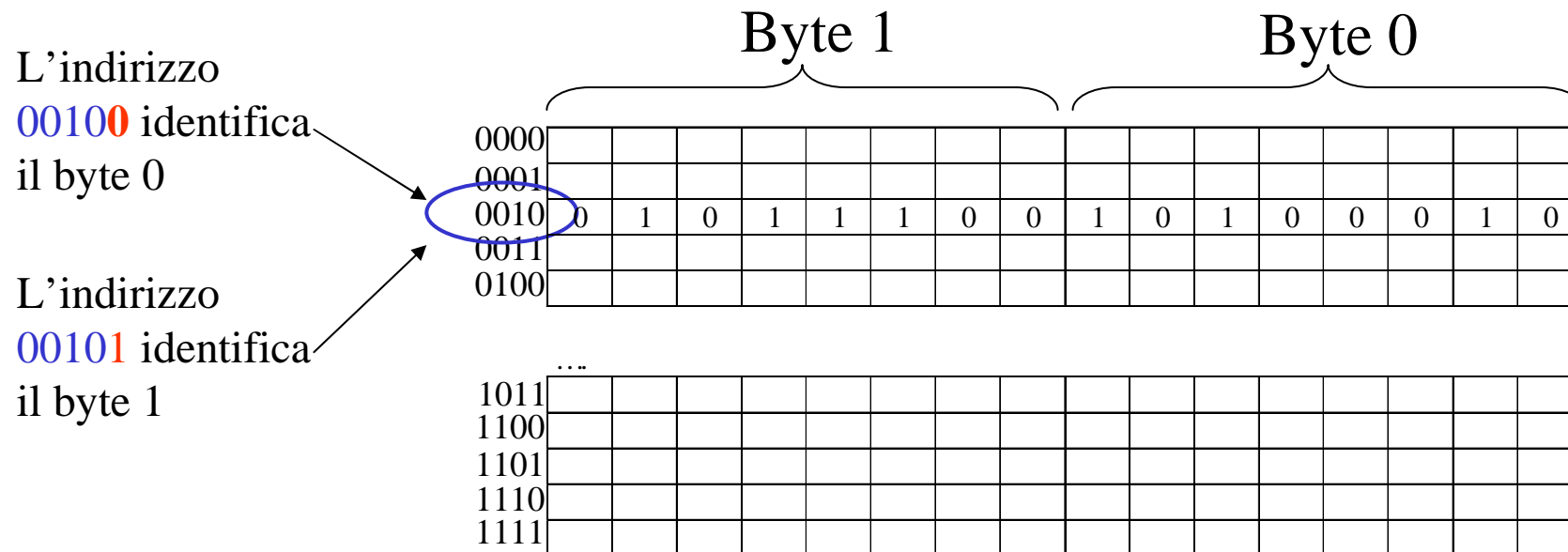
- **Unità di misura**

byte = 8 bit (2³ bit)

Prefissi

- Kilo (K): $2^{10} = 1.024$
- Mega (M): $2^{20} = 1.048.576$
- Giga (G): $2^{30} = \dots$
- Tera (T): $2^{40} = \dots$

- Le memorie di solito **permettono di indirizzare i byte...**
 ⇒ più linee indirizzi rispetto all'indirizzamento di parola!
- Esempio con parole di 2 byte (16 bit):



ES:

- 16 parole: 32 byte → indirizzo 5 bit (4: parola + 1: byte)
- 128 parole: 256 byte → indirizzo 8 bit (7: parola + 1 byte)

Prestazioni

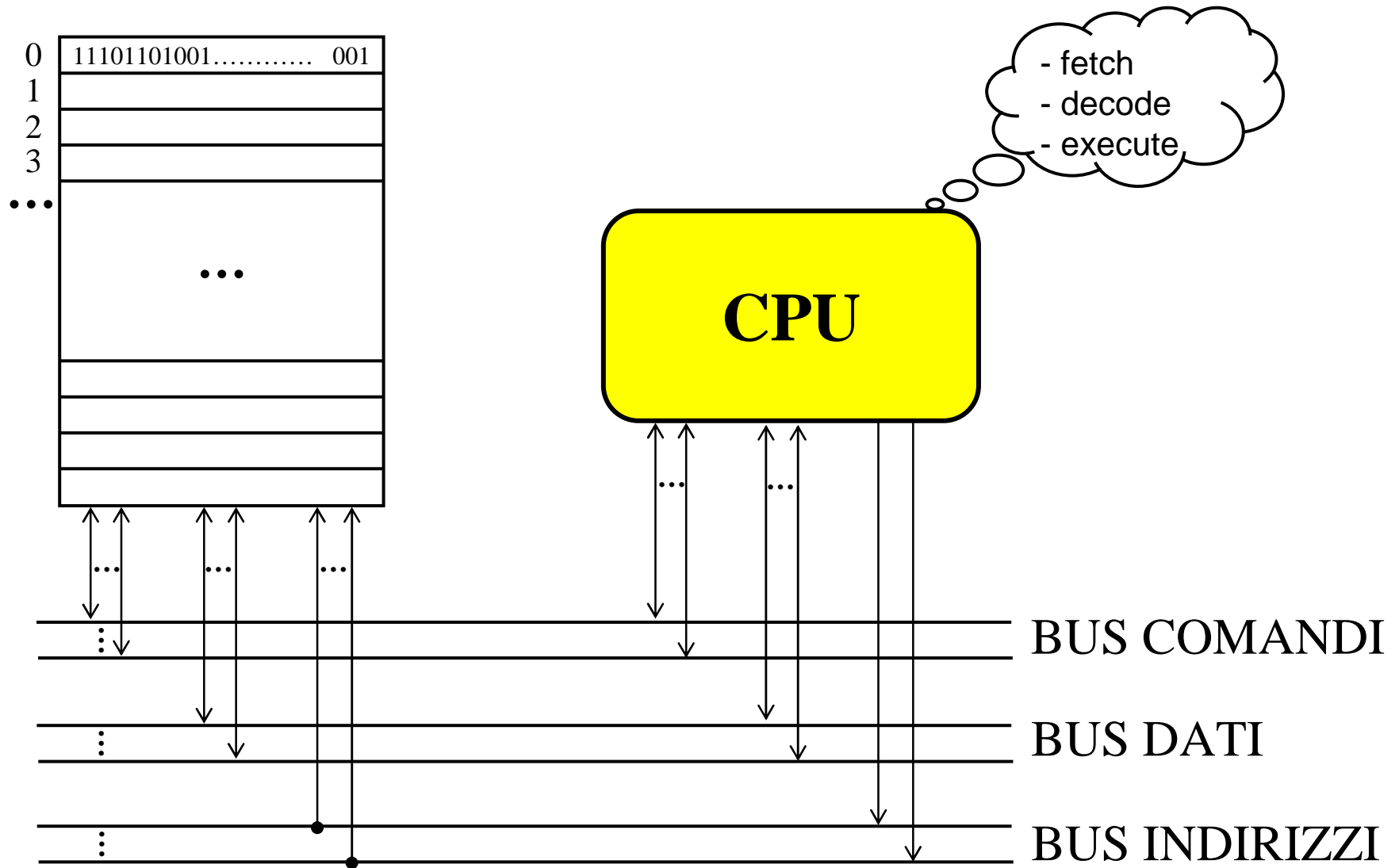
- **Tempo di accesso:**
tempo ingressi stabili – disponibilità dato (lettura)
o completamento memorizzazione (scrittura)
- **Tempo di ciclo:** tempo che intercorre tra un'operazione e la successiva
(tra un'operazione e l'altra può esserci un intervallo)
- **Velocità di trasferimento:** numero di byte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

Memoria centrale: *memoria ad accesso uniforme - RAM*

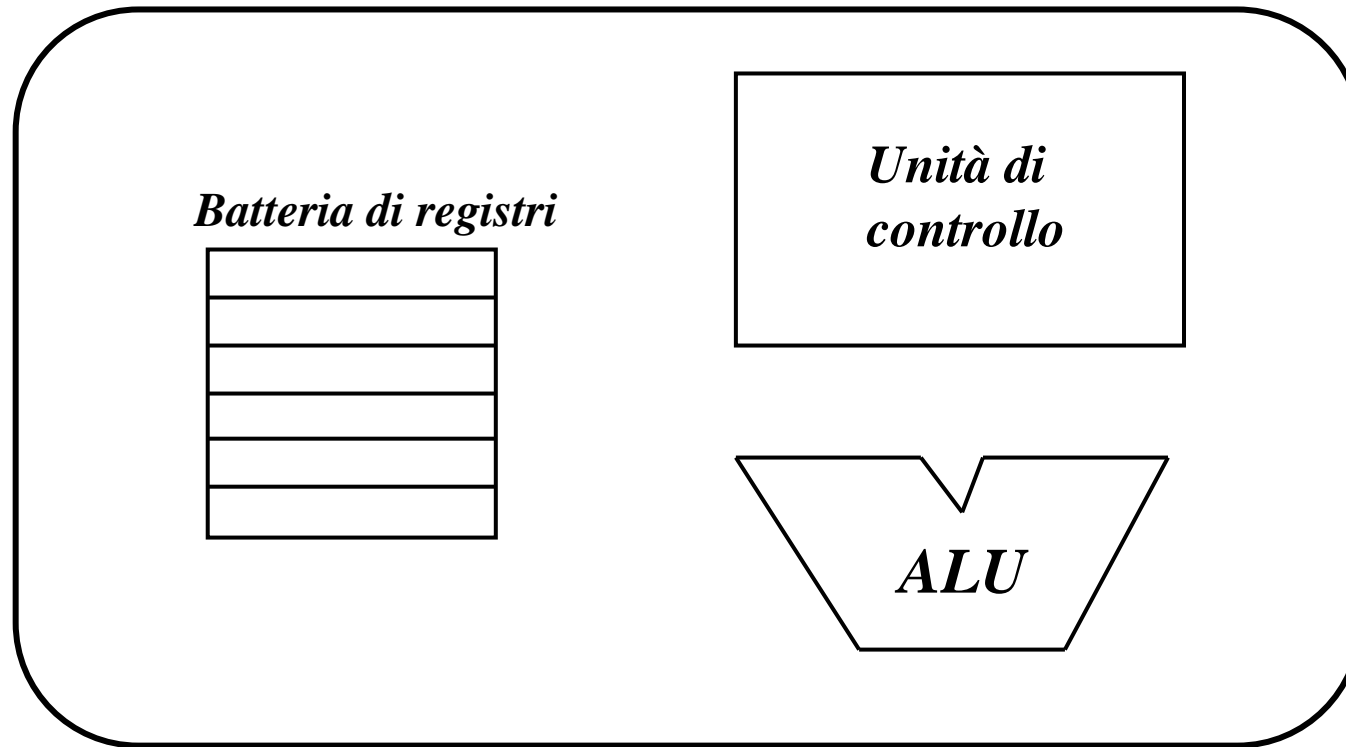
(stesso tempo di accesso e di ciclo per tutte le parole)

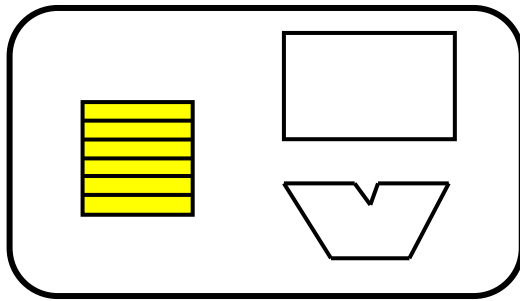
Realizzata con tecnologie elettroniche

L'UNITA' CENTRALE (CPU - processore)



Componenti della CPU





I registri della CPU

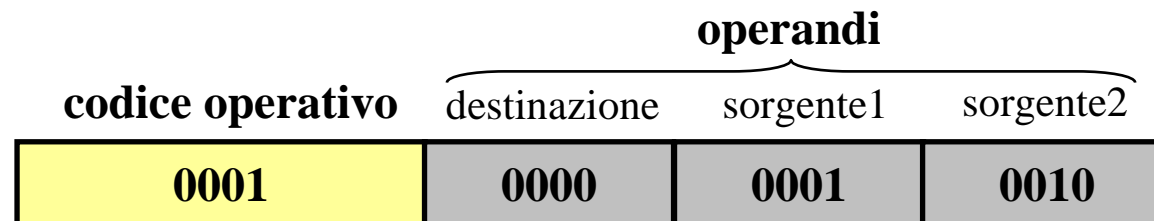
- Costituiscono la “memoria a breve termine” del calcolatore:
 - celle di memoria
 - contengono sequenze di bit di dimensioni limitate (es. 32 bit)
 - tempo di accesso molto ridotto
- Il loro numero e la loro capacità dipendono dallo specifico processore
 - ES: processori a 16, 32 o 64 bit
 - ES: processore con 16, 32, 64... registri
- Utilizzati per immagazzinare informazioni necessarie per eseguire le istruzioni
- Si dividono in *registri di uso generale* e *registri speciali*

Registri di uso generale

- Sono quelli visti finora: contengono operandi e i risultati delle istruzioni da eseguire
- Gli indirizzi di registro sono specificati nei campi operando delle istruzioni

ESEMPIO GIA' VISTO:

sub \$r0, \$r1, \$r2 \\ \$r0 = \$r1-\$r2



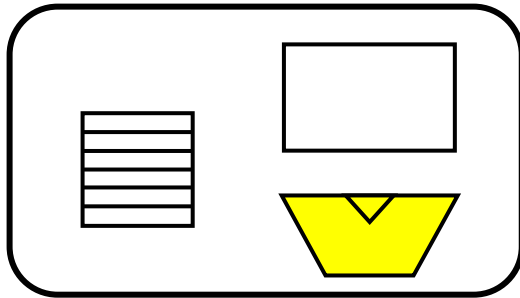
Registri speciali

- ***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

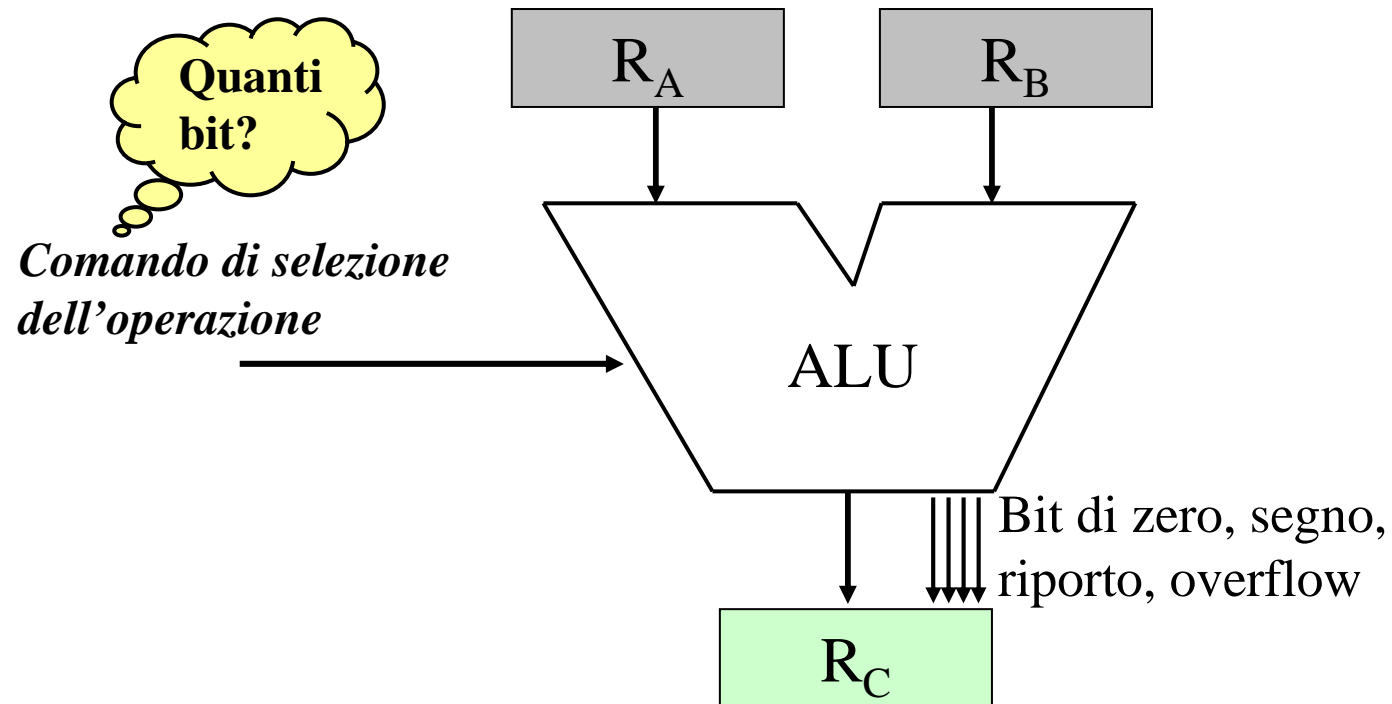
- ***Instruction register*** (IR): memorizza l'istruzione da eseguire
(prelevata dalla memoria)

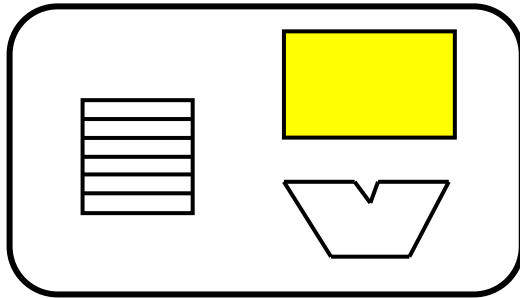
- ***Processor Status Word*** (PSW), detto anche FLAGS:
contiene informazioni sullo “stato” del processore, p.es.
memorizzando le condizioni che si verificano in seguito
all'esecuzione di una istruzione o le modalità di
funzionamento del processore



ALU (Unità Aritmetico/Logica)

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

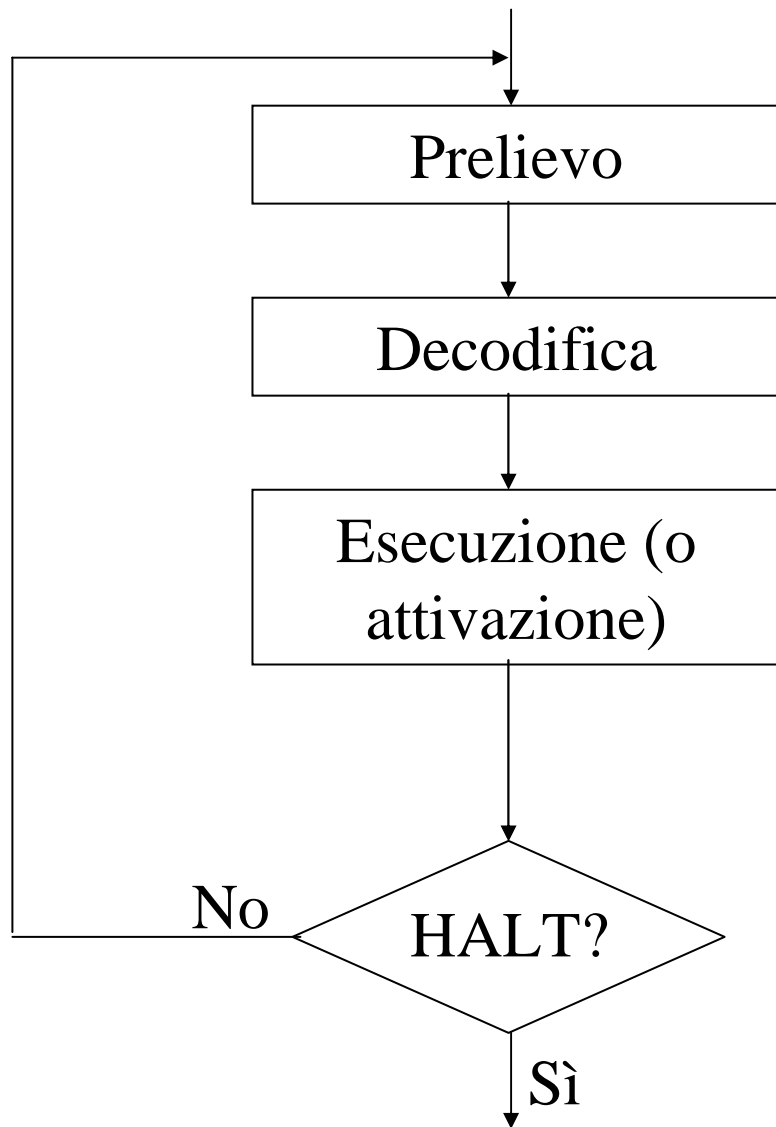




Unità di Controllo

- Coordina il funzionamento delle varie unità della CPU nell'esecuzione dei programmi
- Per farlo, manda segnali di controllo alla ALU (selezione operazione), ai registri (lettura o scrittura valori) e attraverso il bus comandi alla memoria (lettura o scrittura) e alle interfacce di ingresso uscita...
- Il suo comportamento consiste nell'esecuzione del **ciclo macchina** del processore

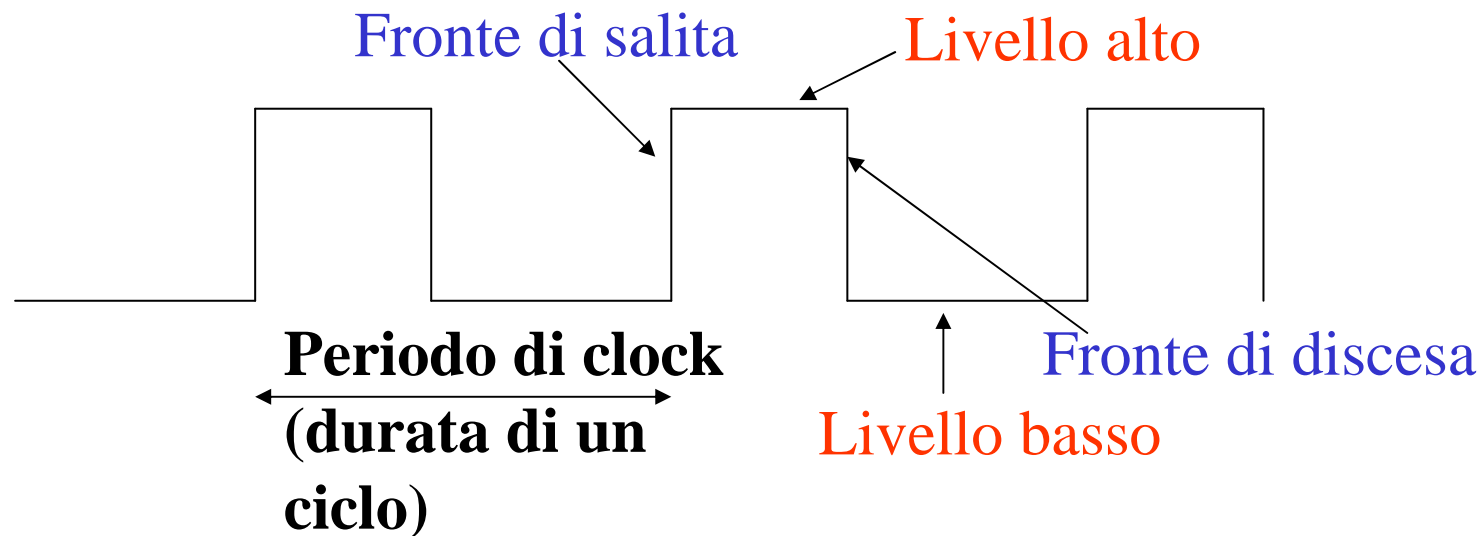
Ciclo macchina



- La CPU opera in modo ciclico, ripetendo indefinitamente i passi:
 1. **Prelievo** dell'istruzione (*fetch*)
 2. **Decodifica** dell'istruzione (*decode*)
 3. **Esecuzione** dell'istruzione (*execute*) detta anche attivazione

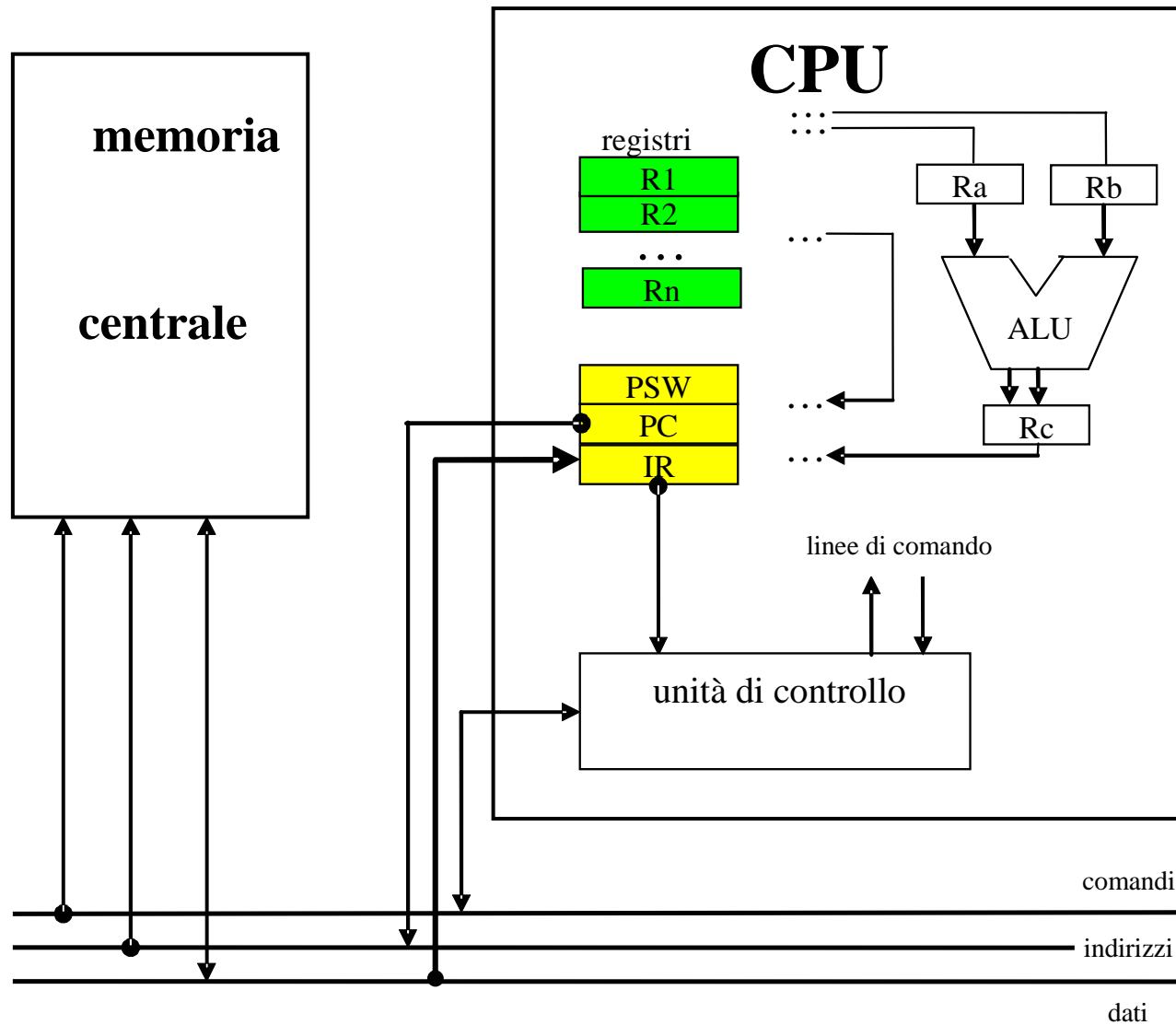
Coordinamento delle unità CPU tramite segnale di clock

- *Segnale di clock*: segnale generato da un generatore di impulsi elettrici, raggiunge tutte le unità
- Tutte le attività elementari della CPU sono sincronizzate rispetto al segnale di clock
- Segnale periodico: **periodo** (o tempo di ciclo) predeterminato



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

Un calcolatore elementare



Ciclo macchina (più 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]→bus indirizzi
 - 2) invio sul bus di controllo del comando di lettura dalla memoria
...M([PC])→bus dati
 - 3) lettura dal bus dati e memorizzazione dell'istruzione in IR
bus dati→IR
- Contestualmente, incremento del contenuto di PC
[PC] + "1"→PC

DECODIFICA

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

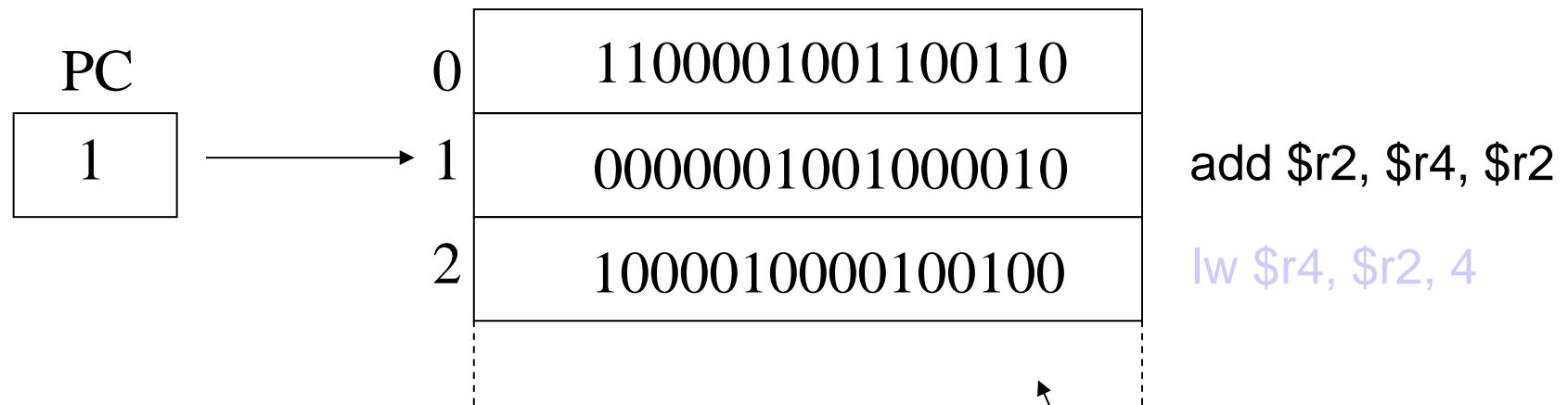
ESECUZIONE

- 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

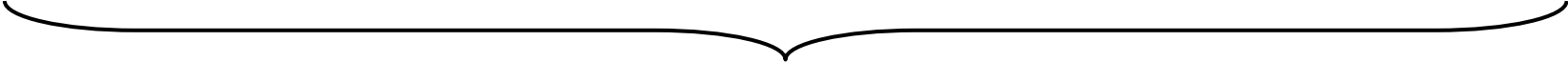
PASSO 1: fetch (primo ciclo di clock)

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

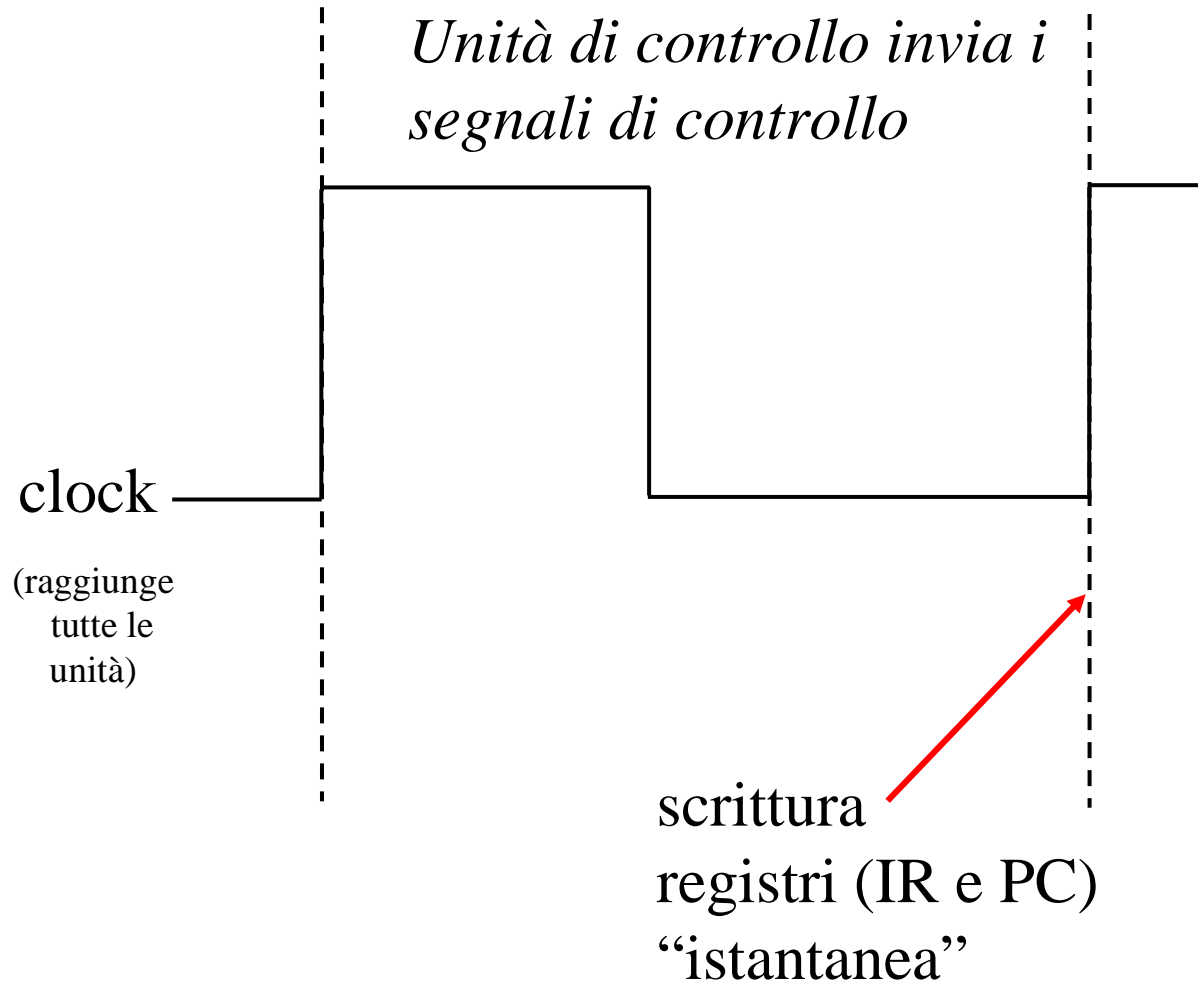
$IR \leftarrow 0000001001000010$

... 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 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)
 - La CPU incrementa il valore in PC

NOTA



PASSO 2: decode (secondo ciclo di clock)

ISTRUZIONE in IR

$$\begin{array}{cccc} 0000 & 0010 & 0100 & 0010 \\ \underbrace{\hspace{1.5cm}} & \underbrace{\hspace{1.5cm}} & \underbrace{\hspace{1.5cm}} & \underbrace{\hspace{1.5cm}} \\ \text{add} & \$r2 & \$r4 & \$r2 \end{array}$$

In pratica, l'unità di controllo

-legge (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 3.2: 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

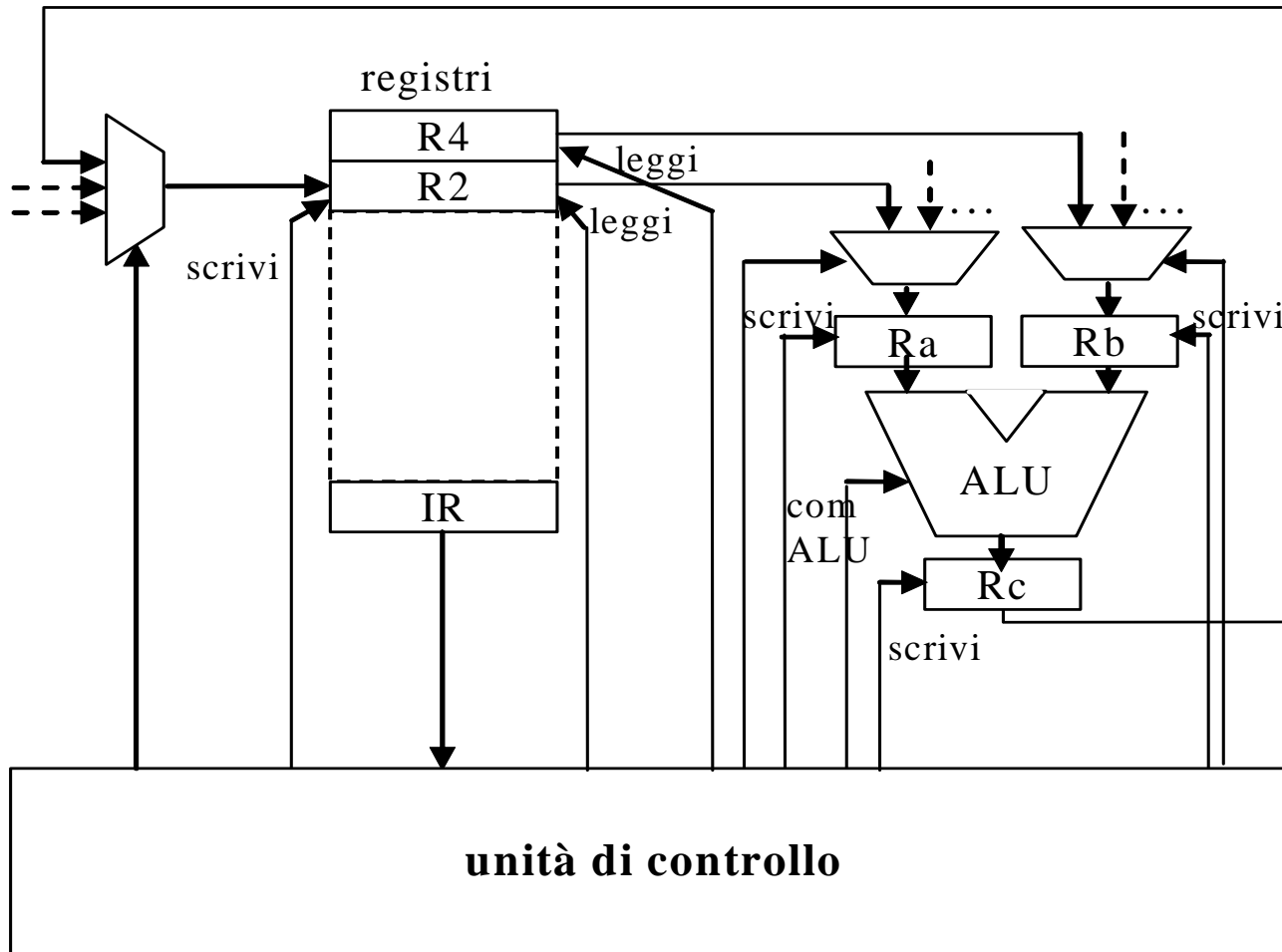
- Passo 3.3: Memorizza risultato in R_2

$$R_2 \leftarrow R_C$$

Unità di controllo
manda segnali di
lettura e scrittura

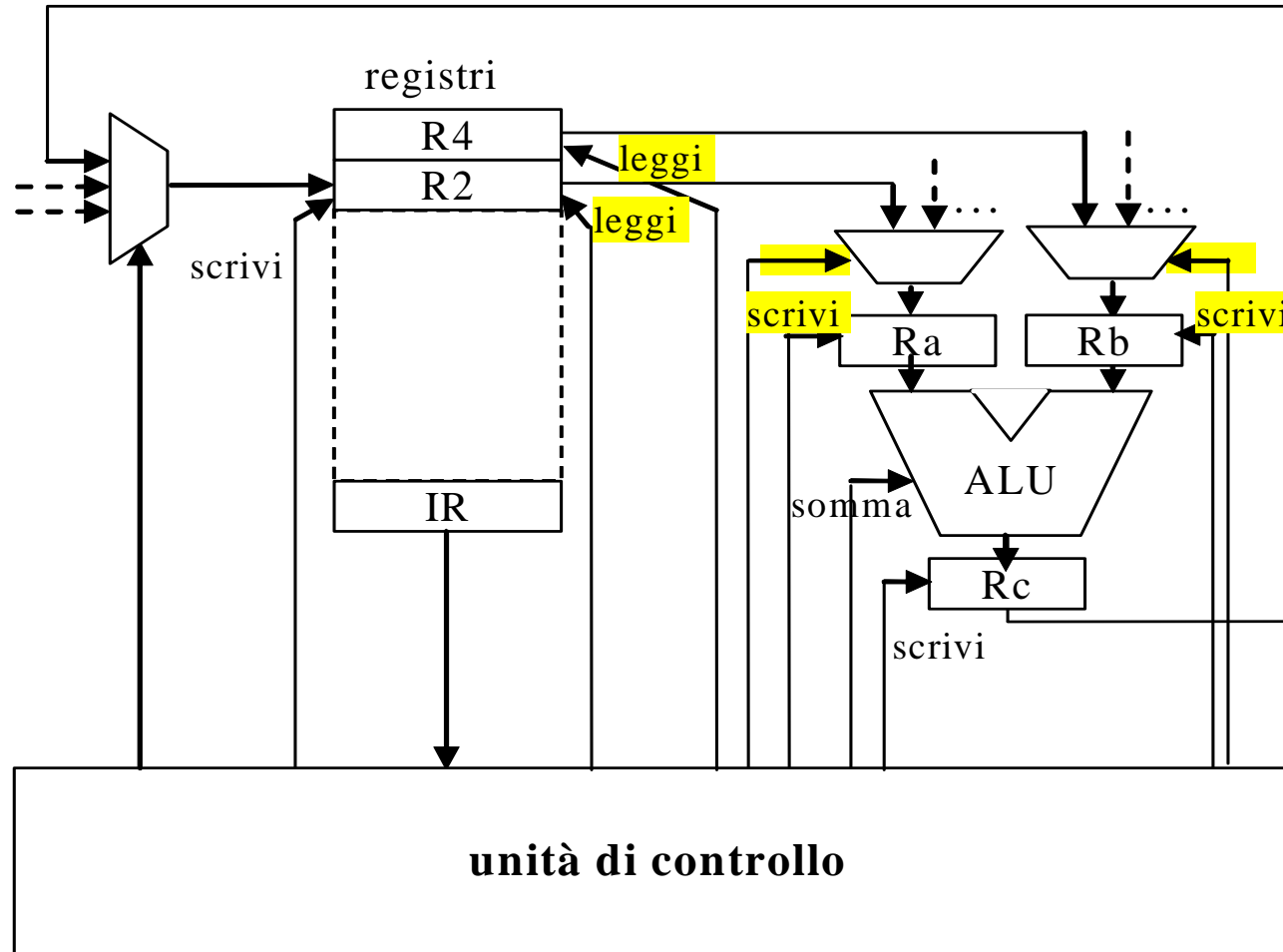
TOTALE = 5 cicli di clock

Schema più dettagliato con l'uso di multiplexer (selettori)

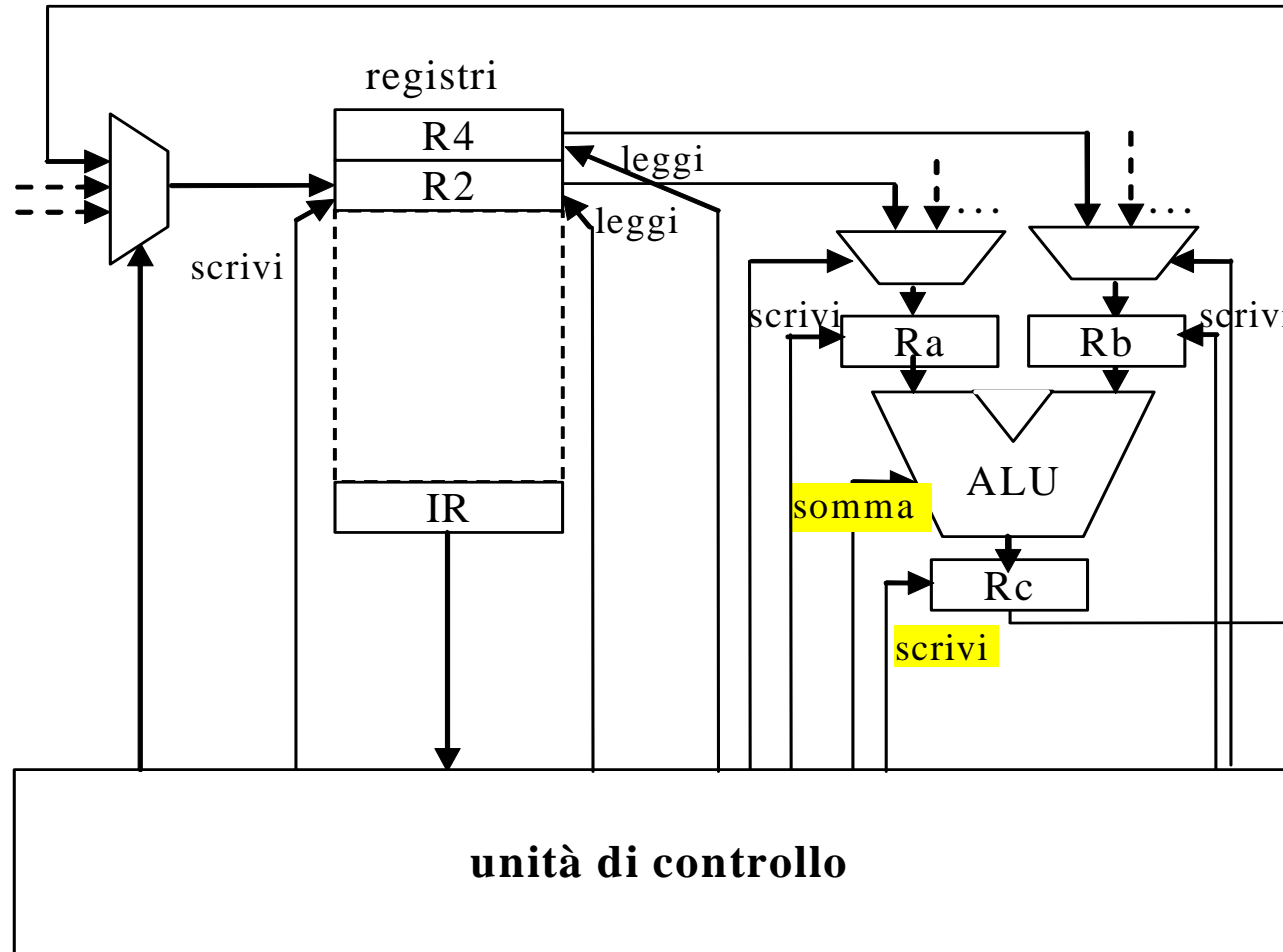


NB: non è indicato il segnale di clock

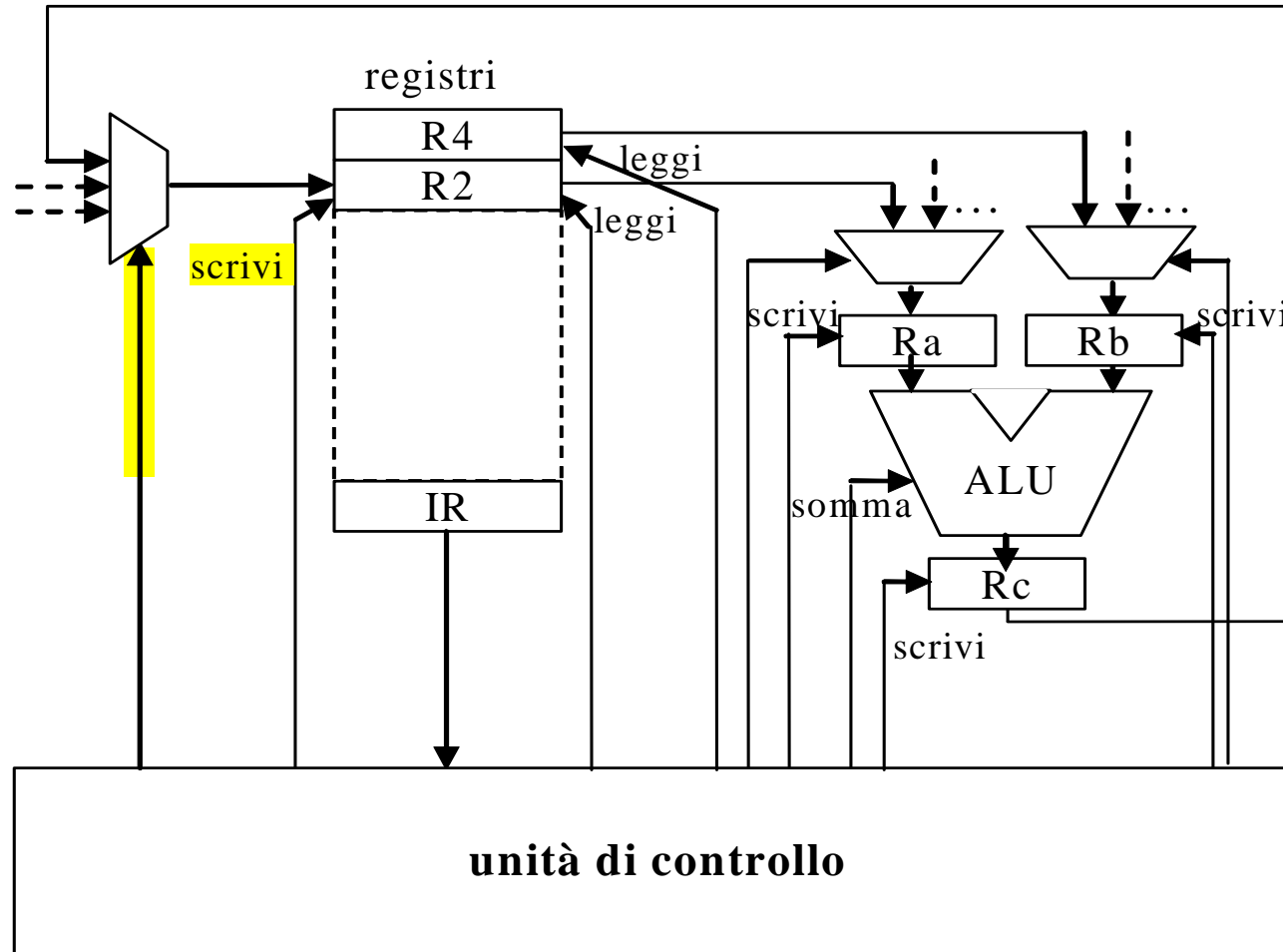
PASSO 3.1



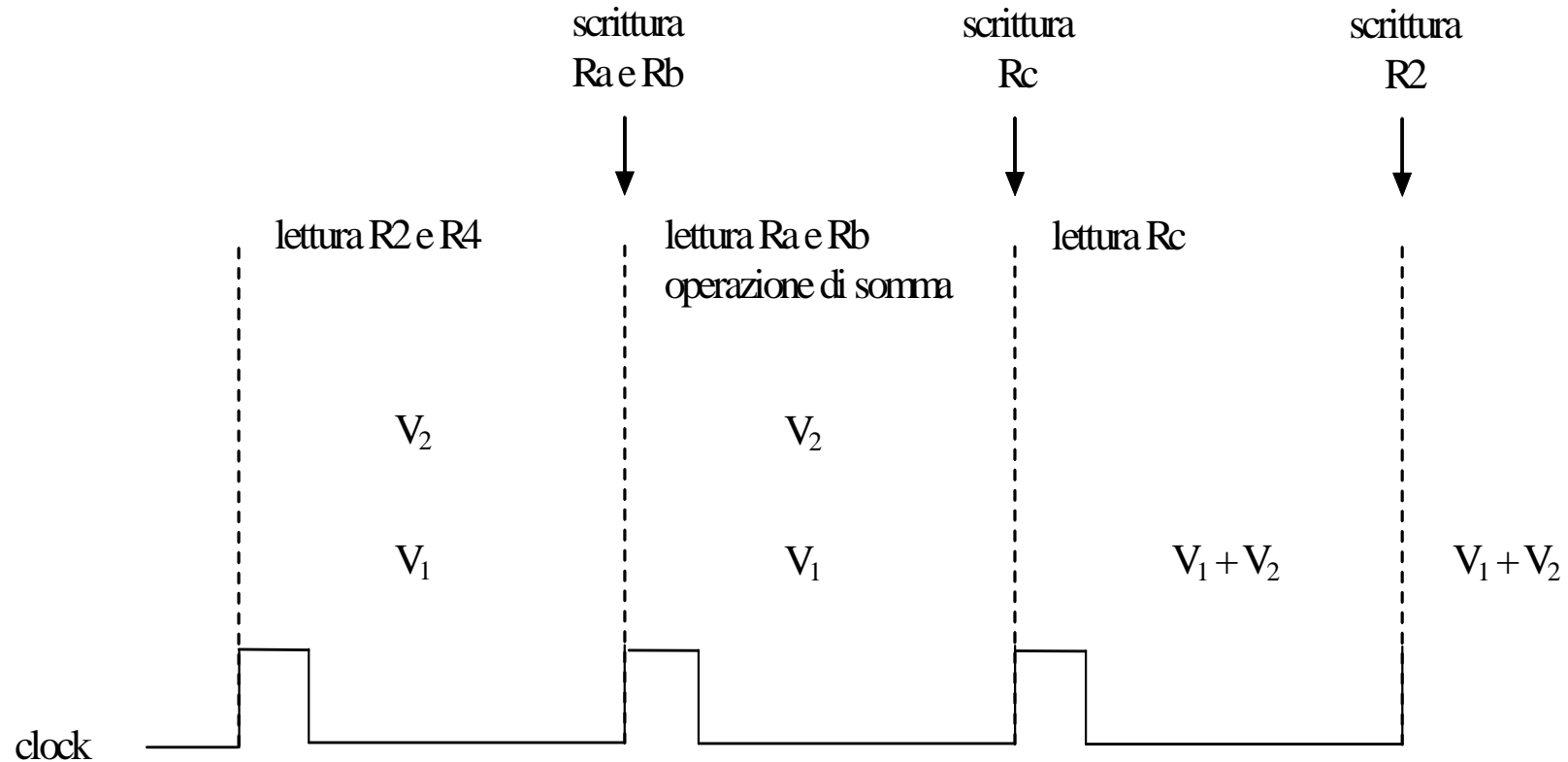
PASSO 3.2



PASSO 3.3



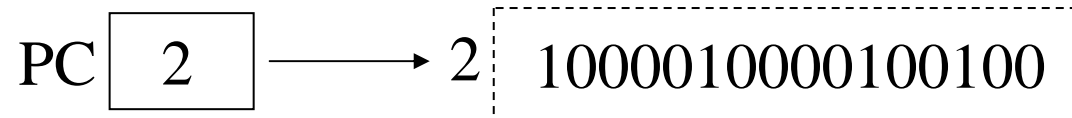
- NB: temporizzazione ai passi 3.1, 3.2 e 3.3



Per ogni passo un ciclo di clock

L'esempio continua: esecuzione di una istruzione lw

- 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

$$\begin{array}{cccc} \underbrace{1000} & \underbrace{0100} & \underbrace{0010} & \underbrace{0100} \\ lw & \$r4 & \$r2 & 4 \end{array}$$

- 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 cicli di clock

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 cicli di clock

Segnale di clock: misure

- Periodo di clock: normalmente misurato in **ns** (**nanosecondi**, 10^{-9} secondi)
- Frequenza di clock: misurata in **Hertz** (**1Hz** = 1 ciclo/secondo)
- Esempio:
 - clock con frequenza 2Ghz: 2 miliardi di impulsi al secondo, periodo = $1/(2 \times 10^9) = 0,5 \times 10^{-9} = 0,5$ ns
- *Come visto, i passi per compiere le operazioni 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!

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
- NB: per valutare le prestazioni di tutto il calcolatore, non basta considerare la CPU (es: memoria, periferiche, bus, ecc.)