Corso di Laurea triennale in Ingegneria Chimica

in condivisione con

Corso di Laurea triennale in

Ingegneria Navale e Scienze dei Materiali

Elementi di Informatica

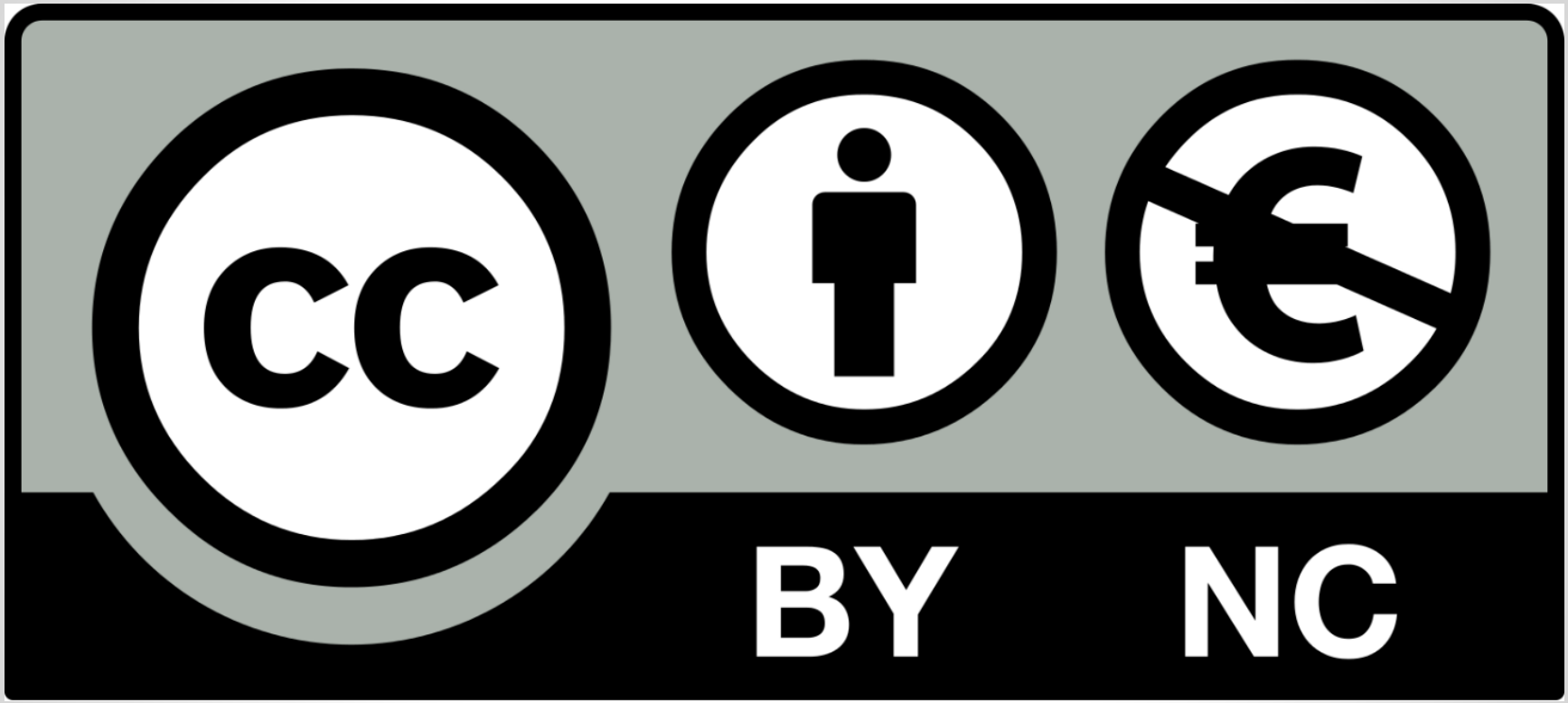
A.A. 2016/17

prof. Mario Barbareschi

Modello di Esecutore

Informazioni di Licenza

- Questo lavoro è licenziato con la licenza Creative Commons BY-NC



- Per consultare una copia della licenza visita: http://creativecommons.org/licenses/by-nc/3.0/legalcode

Elaborazione delle informazioni

- L'**elaborazione delle informazioni** è il mezzo con il quale l'informatica produce **nuova informazione: essa è trasformazione di dati.**
  - I **dati** sono una **forma di rappresentazione dei valori** delle informazioni
  - L'**elaborazione** è una trasformazione

$$Y = F(X)$$

Dove:

- X è l'insieme dei *dati iniziali* (o di ingresso)
- Y è l'insieme dei *dati finali* (o di uscita)
- F è la regola che fa corrispondere Y a X.

- Il processo di codifica delle informazioni ci permette di definire forme di rappresentazione dei dati.

Il processo di elaborazione: l'algoritmo

- Per effettuare **elaborazioni** è necessaria una sequenza di **azioni elaborative** che devono essere applicate sui dati, producendo dati intermedi e finali (**risultati**).
- L'algoritmo è una **sequenza (1) finita (2)** di **azioni elaborative (3)** che **risolve automaticamente un problema (4)**.
  1. Il concetto di **sequenzialità** di passi è fondamentale per la produzione dei dati intermedi e finali.
  2. La sequenza di passi deve essere **finita** per produrre risultati pratici.
  3. Un'**azione elaborativa** è una attività che produce una semplice trasformazione dei dati.
  4. L'algoritmo deve essere **deterministico** (produce sempre stessi risultati se rieseguito) e **automatico** (non occorrono interventi dall'esterno).

Algoritmi e programmi

- Gli algoritmi non strettamente si applicano solo ai dati, ma, in generale, ad una più ampia classe di ambienti e problemi:
  - L'algoritmo per individuare il massimo tra tre numeri.
  - L'algoritmo per calcolo della radice quadrata.
  - L'algoritmo per ordinare le carte in mano durante una partita a poker.
  - L'algoritmo per preparare una torta alla frutta.
- Gli algoritmi applicati ai dati producono le trasformazioni Y=F(X), ovvero generano nuova informazione!
- Nei sistemi di calcolo un **programma** è una **rappresentazione di un algoritmo**. Tale rappresentazione è in una forma "comprensibile" per l'esecutore.
  - Es: se le "ricette culinarie" sono gli algoritmi, allora i "programmi" sono le descrizioni delle ricette nella lingua adeguata ad i cuochi di un Paese.

Il calcolo automatico

- Sul piano teorico **il calcolo (automatico) si ottiene con**:
  - Dati iniziali
  - Un insieme di regole che costituiscono un programma.
  - Un elaboratore (automatico) che esegue il programma producendo dati intermedi e dati finali, ovvero risultati.

Sappiamo rappresentare le informazioni, ora approfondiamo
il modello di elaboratore, che è necessario conoscere per scrivere programmi.

Algoritmo, Processo, Processore e Programma

- Informalmente, un **algoritmo** descrive un lavoro, la descrizione del lavoro in un linguaggio comprensibile ad un calcolatore è detto **programma**.
- L'esecuzione di un algoritmo da parte di un esecutore si traduce **in una successione automatica e finita di azioni elaborative** che vengono effettuate nel tempo.
- Si definisce **processo** il lavoro (svolto e) che si sta svolgendo eseguendo l'algoritmo, e **processore** il suo esecutore.
  - Il processo è un programma in esecuzione, è le azioni svolte dipendono dal particolare insieme di dati input e delle condizioni che si susseguono nel tempo.
  - Pertanto, un programma può originare differenti processi, a seconda delle condizioni in cui il lavoro viene svolto, che determinano il comportamento dell'esecutore durante l'esecuzione.

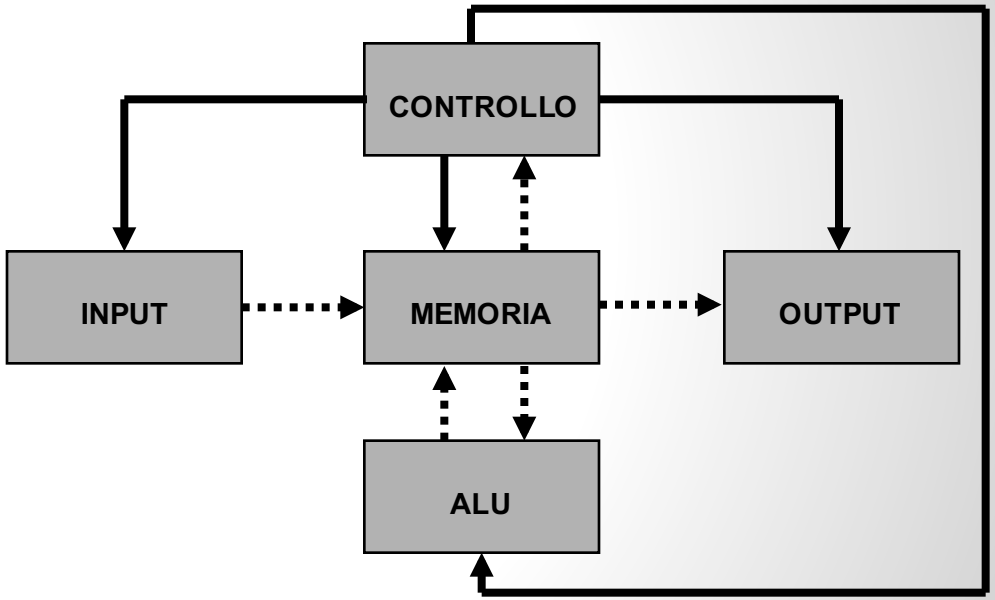
Calcolatore Elettronico

- I dispositivi moderni che eseguono calcoli automatici sono detti calcolatori elettronici.
- Sono dotati di capacità computazionali molto elevate ed eseguono operazioni di elaborazione **elementari** ad altissima velocità.
  - I oderni microprocessori lavorano a frequenze dell'ordine dei Ghz (10^9 cicli/secondo) eseguendo anche 300.000 milioni di istruzioni al secondo (MIPS, **M**illion **I**nstructions **P**er **S**econd)
  - Grazie all'altissima velocità di elaborazione anche operazioni complicate sono eseguite in tempi trascurabili nonostante siano scomposte in un gran numero di operazioni semplici.

Il modello di Von Neumann

- È uno schema di principio rappresentativo dei tradizionali computer che prende il nome da Von Neumann, il primo ricercatore che lo propose nel 1945.

  - La *Central Processing Unit* (*CPU*) coordina l'esecuzione delle operazioni fondamentali;
  - La ALU effettua le operazioni logico-aritmetiche;
  - La *memoria* contiene un programma che descrive le operazioni da eseguire e i dati su cui il programma stesso opera;
  - I dispositivi di *input* e *output* sono le interfacce della CPU nei confronti del mondo esterno

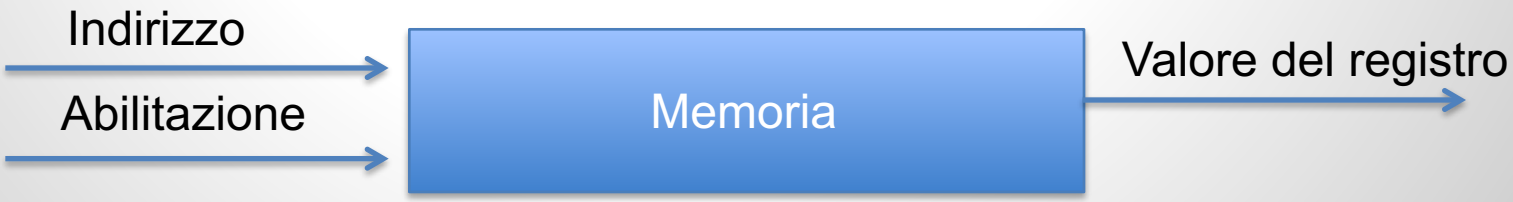


Stored Procedure

- Il modello di Von Neumann si basa sul concetto di **programma memorizzato**:
  - la macchina immagazzina nella propria memoria **i dati su cui lavorare** e **le istruzioni** per il suo funzionamento, conferendo al calcolatore ampia flessibilità operativa.
- Nelle linee generali il funzionamento interno di un qualsiasi computer odierno si può ricondurre al modello di Von Neumann.

Memorie

- Le **memorie sono insiemi di registri**, ognuno associato ad **indirizzo di memoria** che lo identifica.
  - nelle memorie di tipo elettronico i flip-flop rappresentano informazioni binarie tramite valori di tensione.
  - nelle memorie di tipo magnetico è il verso di polarizzazione del supporto che è associato al valore di un bit.
  - nelle memorie di tipo ottico è la riflessione che offre il supporto ad un raggio laser che codifica il valore di un bit.
  - I differenti stati degli organi elementari di memoria sono associati al valore binario di un bit da rappresentare.
- Le moderne memorie hanno registri di 8 bit (1 byte): ogni indirizzo di memoria permette di identificare un registro di 8 bit



Operazioni sulla memoria

- **Load**:
  - preleva l'informazione contenuta nel registro all'indirizzo desiderato, senza alterare il contenuto del registro.
- **Store**:
  - inserisce una informazione nel registro all'indirizzo desiderato, sovrascrivendo il precedente valore.
- La memoria è un sistema che assolve al compito di conservare il dato, depositandolo in un registro nel caso di operazione di scrittura, e di fornire il dato precedentemente depositato in un registro in caso contrario di operazione

Funzionamento di una Memoria

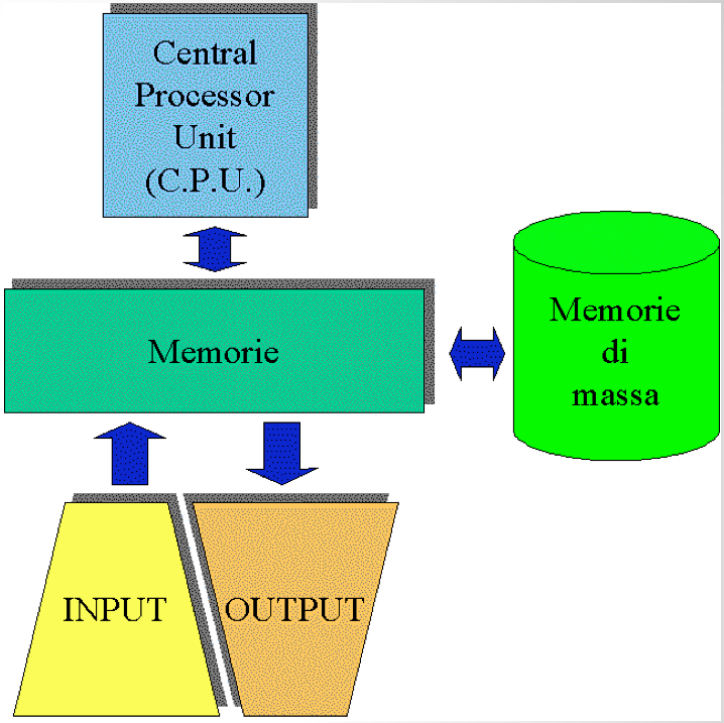
- La CPU indica l'indirizzo del registro interessato dall'operazione;
- La memoria abilita (attiva) solo il registro ad esso corrispondente affinché:
  - per una **store** copi il dato del buffer nel registro;
  - per un **load** copi il dato del registro nel buffer.
- Le operazioni di load e store richiedono tempi di attuazione che dipendono dalle tecnologie usate per la costruzione delle memorie e dalle modalità di accesso.
  - Nel caso di load, il **tempo di accesso** misura il tempo che trascorre tra **la selezione del registro di memoria** e la **disponibilità del suo contenuto** nel registro di buffer.
  - Il **tempo di accesso** nel caso dello store misura invece il tempo necessario alla selezione del registro e alla scrittura del contenuto del buffer nel registro.

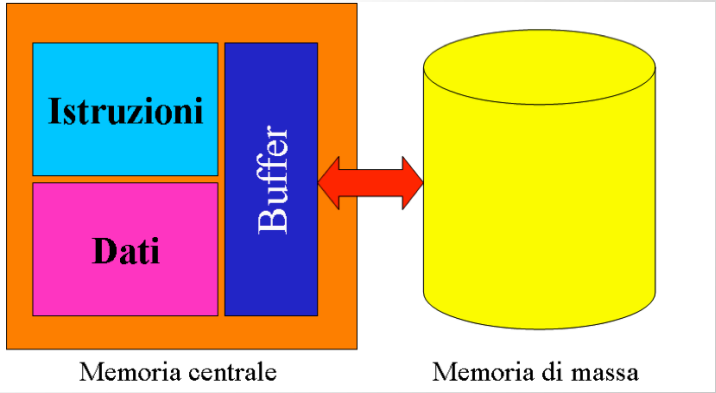
Classificazione delle memorie

- **Memoria ad accesso casuale (Random Access Memory, RAM)**
  - Se il tempo di accesso ai dati non dipende dalla loro posizione in memoria.
- **Memoria ad accesso sequenziale**
  - il tempo di accesso ai dati dipende dalla posizione dei dati (es.: nastri magnetici).
- Alcune memorie vengono realizzate in modo che sia possibile **una ed una sola scrittura di informazioni**. Tali memorie vengono dette **a sola lettura** o **Read Only Memory (ROM)**.
  - Queste memorie trovano impiego quando non è necessario alterare il valore delle istruzioni o dei dati che sono memorizzati sul dispositivo.
- Le memorie sono dette **volatili** quando perdono le informazioni in esse registrate quando il sistema viene spento, altrimenti sono dette **permanenti**.

Memorie di Massa

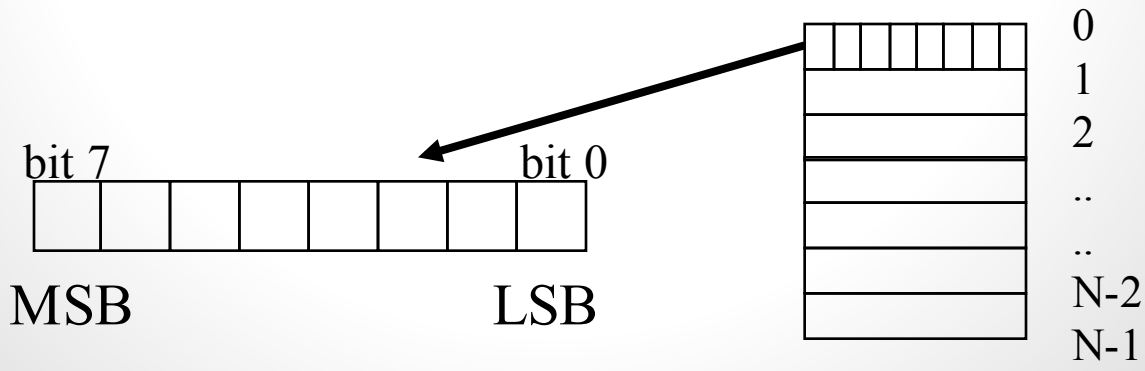
- Le **memorie di massa** sono memorie **ausiliarie** caratterizzate da una **elevata capacità**.
  - Per essere usate, le informazioni della memoria di massa devono prima essere trasferite nella memoria centrale.
  - Similmente, le informazioni da scrivere nelle memorie di massa devono passare per la memoria centrale.
- Le memorie di massa hanno **tempi di accesso maggiori** dovuti alle tecnologie (più economiche) impiegate per realizzarle.
- Per mitigare la differenza di velocità tra i due dispositivi si interpongono delle *aree di accumulo dei dati in transito* (dette **aree buffer**).





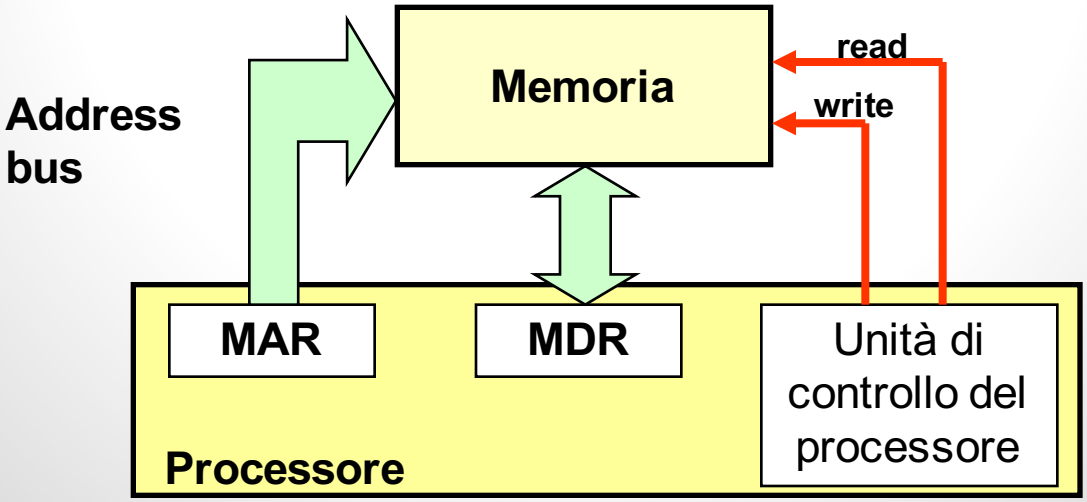
La memoria centrale

- La memoria centrale di un computer è organizzata come un **insieme numerato** di registri di lunghezza m, dette **locazioni.**
- Gli *m* bit di una locazione sono accessibili dal processore (in lettura/scrittura) mediante un'unica operazione (questo parametro indica il **parallelismo** della memoria).
- Ogni locazione è individuata da un **indirizzo**, cioè un intero compreso tra 0 e N-1. L'insieme di locazioni indirizzabili è detto **SPAZIO DI INDIRIZZAMENTO.**
- La memoria centrale è **ad accesso casuale** (RAM) perciò il tempo di accesso non dipende dalla posizione del dato (ovvero dall'indirizzo).



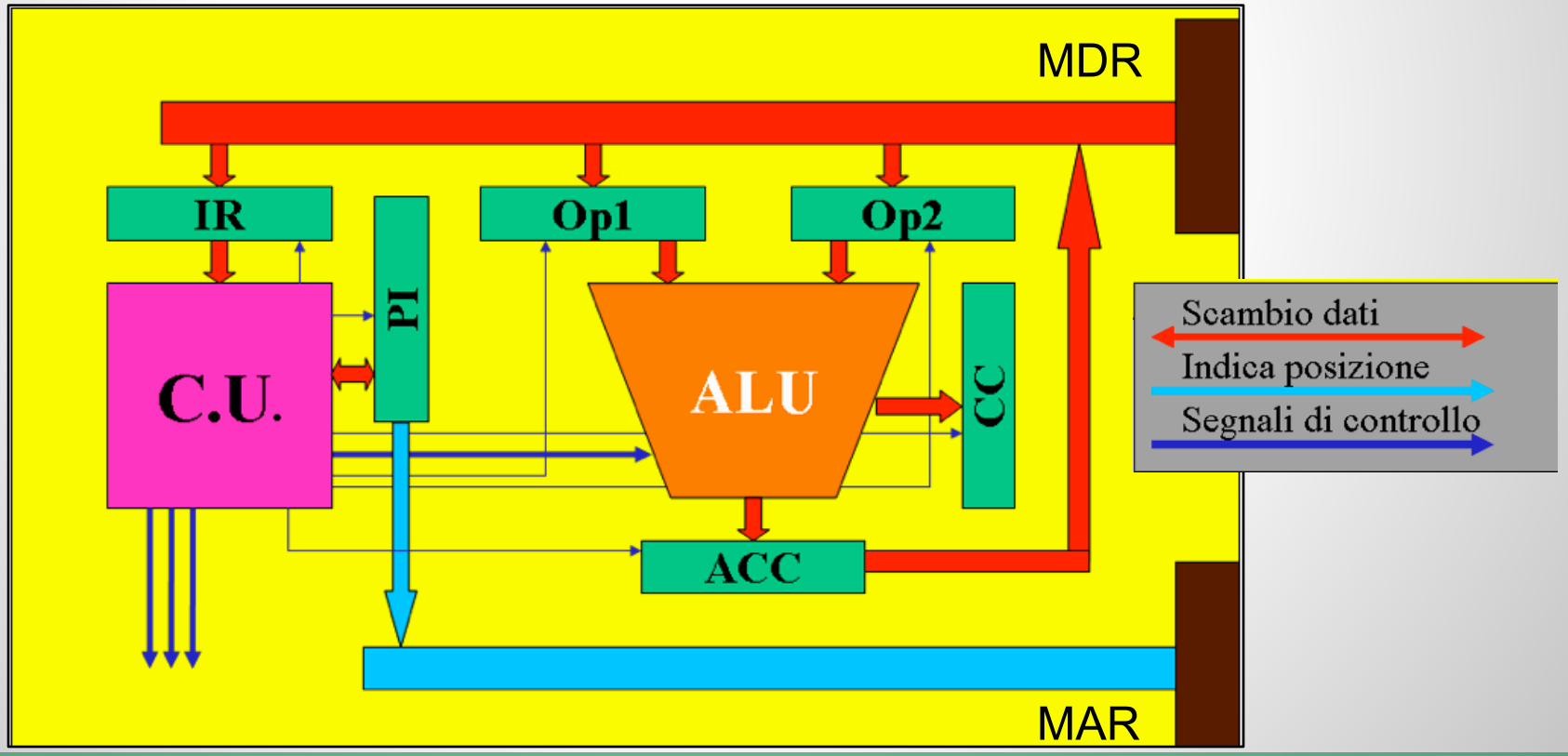
Interazione processore-memoria

- Le interazioni tra processore e memoria centrale avvengono tramite **due registri del processore** e **segnali di controllo**:
  - Il **Registro Memory Address** (MAR) fornisce alla memoria l'indirizzo del dato da leggere/scrivere.
  - Il **Registro Memory Buffer** o **Memory Data** (MBR/MDR) contiene il dato da **scrivere verso**, o da **leggere dalla memoria.**
  - I **Segnali di controllo read/write**, ordinano alla memoria il momento in cui leggere il valore del MAR e del MDR per effettuare le operazioni di lettura oppure di scrittura.



La Central Processing Unit (CPU) (1/2)

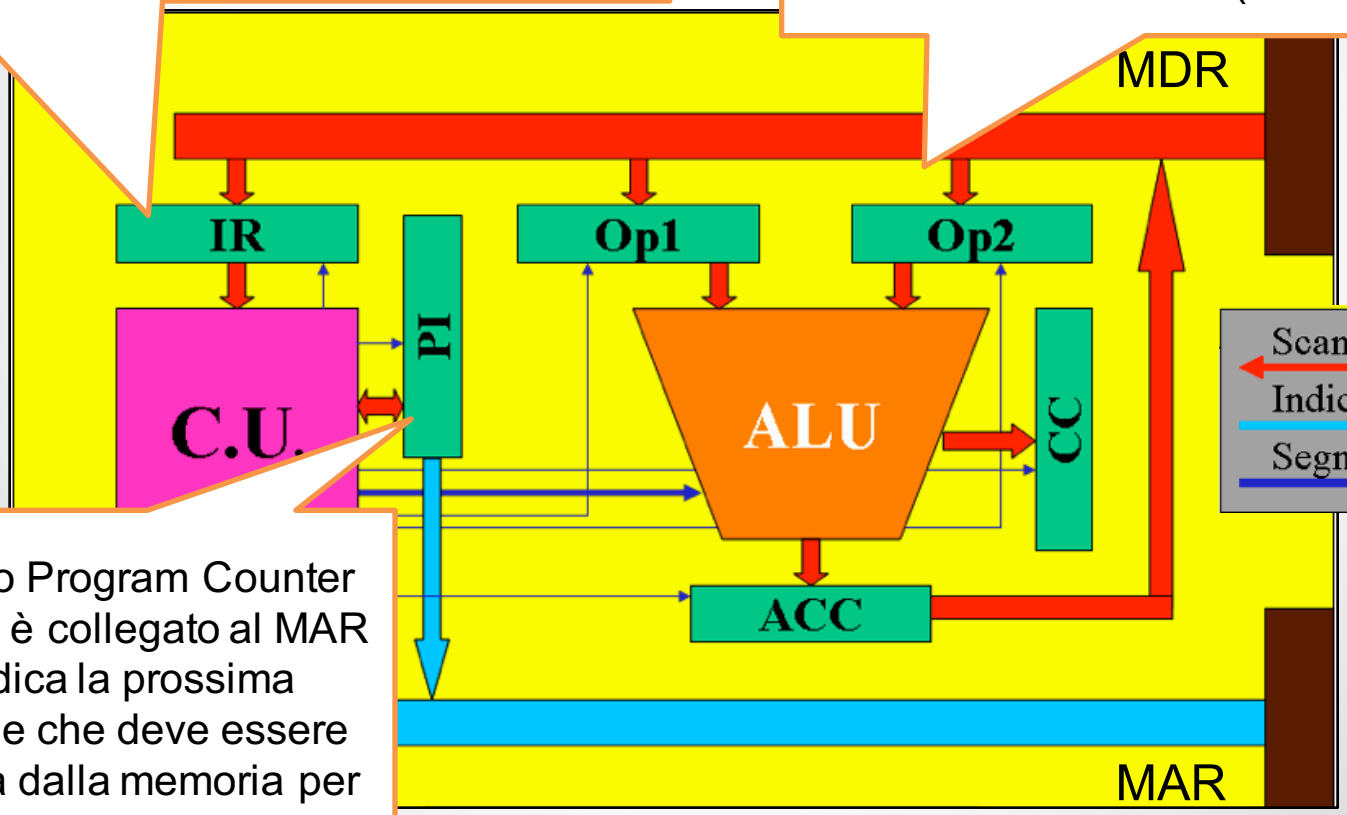
- Rispetto al modello di von Neumann, i microprocessori integrano nell'unità di controllo la ALU e ulteriori registri, chiamati **registri del processore**.
- I registri del processore che contengono dati che transitano dalla memoria sono collegati con l'MDR.
- I registri che indicano le locazioni dove scrivere o leggere i dati sono collegati con il MAR.



La Central Processing Unit (CPU) (2/2)

Il registro IR contiene l'istruzione corrente da eseguire: anche l'istruzione proviene dalla memoria (MDR) ed è in input all'unità di controllo (C.U.)

I dati transitano dalla memoria centrale (MDR) verso registri di input all'ALU. I risultati (ACC) ritornano nei registri, oppure sono spostati in memoria centrale (MDR).



Il registro Program Counter (PC o PI) è collegato al MAR ed indica la prossima istruzione che deve essere prelavata dalla memoria per l'esecuzione

Control Unit (C.U.)

- La CU è preposta all'interpretazione delle singole istruzioni e all'attivazione di tutti i meccanismi necessari ad eseguire le single istruzioni:
  - preleva ogni **istruzione** dalla memoria centrale e la **decodifica**
  - preleva dalla memoria i **dati** che servono all'**esecuzione dell'istruzione**
  - **eseguire** infine l'istruzione quando tutti i dati sono disponibili
    - Per esempio: se l'istruzione prelevata è una somma tra due operandi, la CU prepara gli operandi (eventualmente prelevandoli dalla memoria), poi attiva l'ALU affinché esegua l'operazione desiderata, ed infine deposita il risultato in memoria.
- Al termine dell'esecuzione di una istruzione la CU procede al prelievo dalla memoria della successiva istruzione, in maniera sequenziale
  - L'esecuzione di una nuova istruzione inizia solo se la precedente è stata portata a termine.

Aritmetic Logic Unit (ALU)

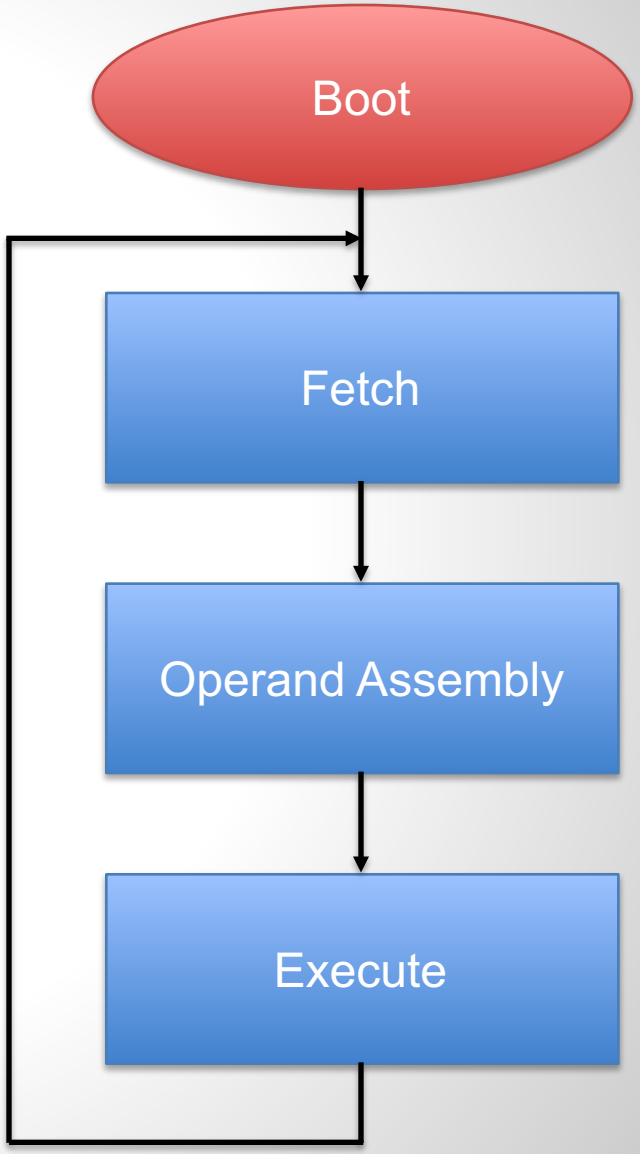
- **Esegue le operazioni elementari di elaborazione**, quali le aritmetiche, le operazioni di confronto o operazioni su bit, **sui dati dei registri interni.**
- L'ALU ha un registro per i risultati ed uno che da informazioni ulteriori sull'esito dell'elaborazione, chiamato **Condition Code (CC).**
- A seconda dei processori l'ALU può essere molto complessa: nei sistemi attuali l'ALU viene affiancata da **processori dedicati alle operazioni sui numeri in virgola** mobile detti **(co)processori matematici**.
- Per i risultati intermedi di una elaborazione la CU può servirsi dei registri interni piuttosto che dei registri della memoria, perché questi ultimi hanno tempi di accesso inferiori.

Registri Interni Speciali

- Il numero e tipo di tali registri varia a seconda dell'architettura della CPU, ma sono generalmente presenti:
  - *Instruction Register (IR)*
    - contiene l'istruzione prelevata dalla memoria e che l'unità di controllo sta eseguendo
  - *Prossima Istruzione (PI)* o *Program Counter (PC)*
    - ricorda alla CU la posizione in memoria della successiva istruzione da eseguire.
  - *Accumulatore (ACC)*
    - serve come deposito di dati da parte dell'ALU
  - *Condition Code (CC)*
    - indica le condizioni che si verificano durante l'elaborazione, quali risultato nullo, negativo e overflow

Il Ciclo del Processore

- Il ciclo inizia con una **fase di boot**
- Il processore, quindi continuamente esegue un ciclo composto da:
  - Fase di **fetch**
  - Fase di **operand assembly**
  - Fase di **execute**



La fase di Boot

- La **Fase di Boot** predispone la CPU per l'inizio del ciclo, impostando **il registro PI alla prima locazione di memoria che contiene istruzioni** per l'Unità di Controllo.
- Una volta avviato, il ciclo del processore non termina mai e ad ogni istruzione ne segue un'altra sino all'arresto del calcolatore.
  - Perché tutto ciò proceda nel rispetto del modello di Von Neumann, in memoria devono essere sempre presenti istruzioni e dati da elaborare.

Ciclo del Processore: Fase di Fetch

- La **fase fetch:**
  - Preleva l'istruzione riferita dal PI dalla memoria e la porta nell'IR.
  - L'UC **interpreta l'istruzione contenuta nell'IR** ovvero:
    - interpreta il **codice operativo dell'istruzione** (cosa fare, e se l'operazione coinvolge l'ALU).
    - Interpreta **i riferimenti ai dati di input ed output** (su quali dati eseguire l'operazione).
    - Interpreta **quale istruzione eseguire successivamente** (cosa fare al termine), e prepara l'indirizzo nel PI per il prossimo ciclo.
- La **fase fetch si conclude** con l'aggiornamento del registro PI con il valore del puntatore all'istruzione successiva.

Ciclo del Processore: Fase di Operand Assembly

- La fase **operand assembly** predispone gli operandi che servono all'esecuzione del codice operativo.
  - Il codice istruzione contenuto nell'IR contiene riferimenti ai dati usati nell'operazione.
  - Questa fase può richiedere ulteriori accessi in memoria centrale per preparare gli operandi in ingresso all'ALU.
- Alla fine di questa fase, sono disponibili tutte le informazioni per portare a termine l'istruzione in esecuzione:
  - E.g. se l'istruzione codifica per un'operazione di somma, alla fine della fase di operand assembly saranno disponibili entrambi gli addendi.

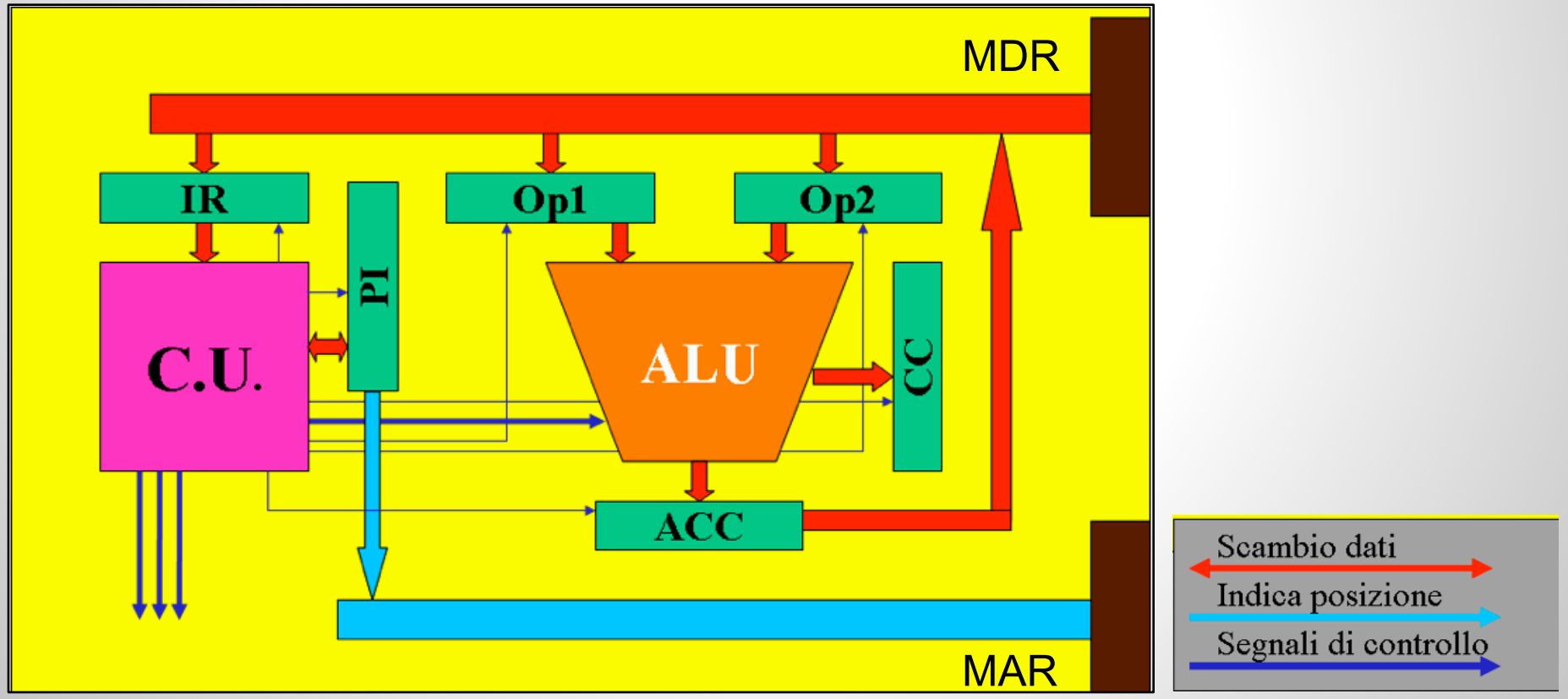
Ciclo del Processore: Fase di Execute

- La **fase execute** consiste nell'eseguire l'operazione di elaborazione indicata dal codice operativo.
  - Se l'istruzione richiede l'attivazione della ALU (e.g. una somma), la CU attiva l'ALU, che esegue l'operazione: il risultato prodotto è conservato in uno dei registri (es.: ACC) e l'ALU aggiorna il CC.
- I risultati, se prodotti, verranno memorizzati negli indirizzi specificati dal codice istruzione.

Fase di Boot

Fase di Boot

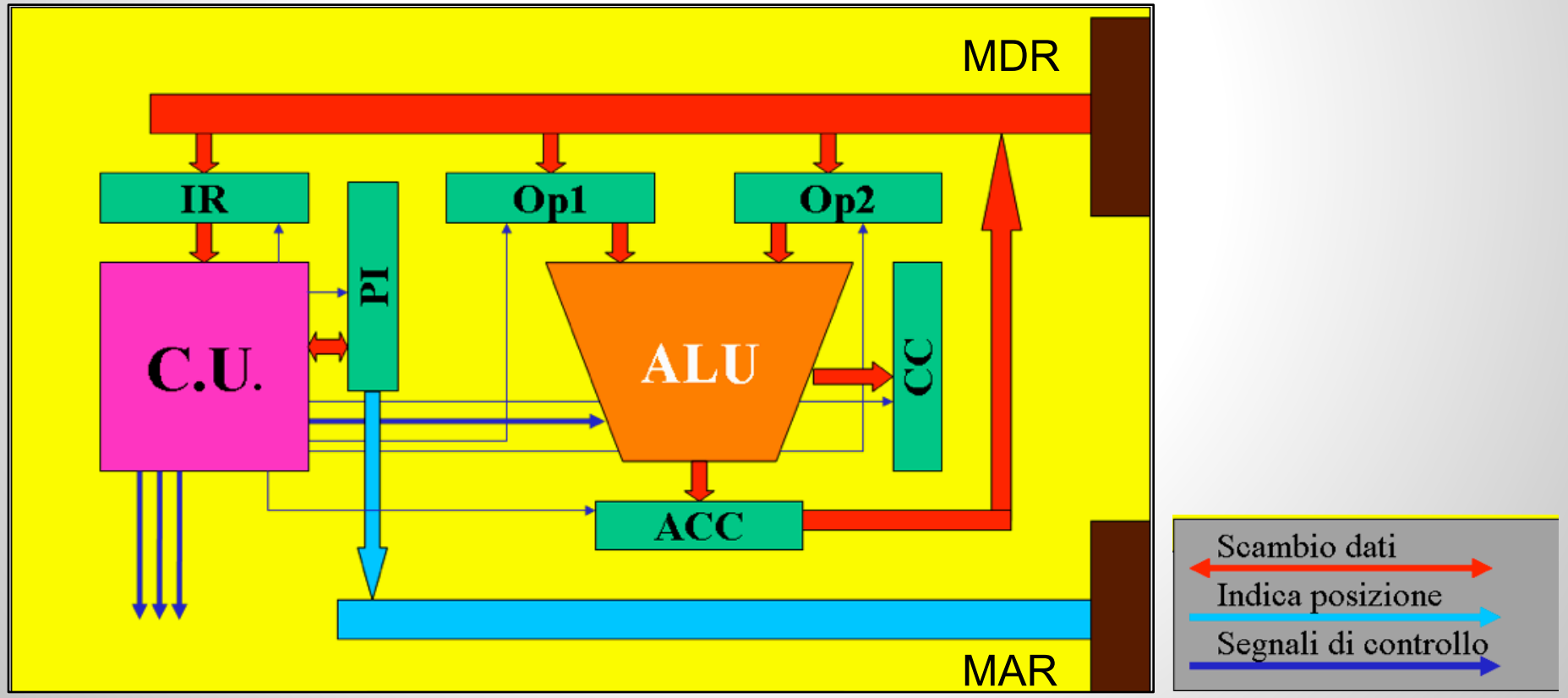
All'avvio il boot inizializza i registri ed inserisce nel PI l'indirizzo della prima istruzione del programma residente nella memoria principale.



Fase di Fetch

Fase di Fetch

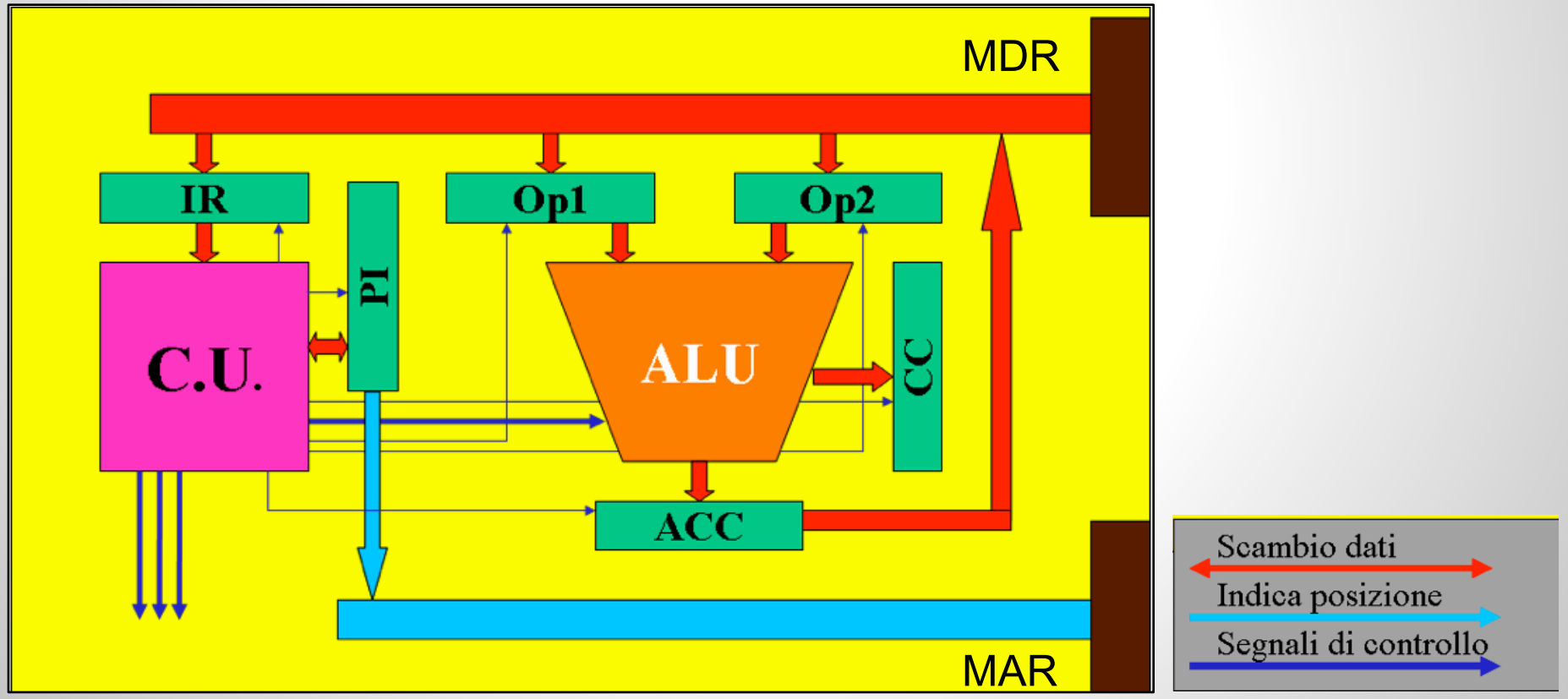
1. La CU ricopia il contenuto del PI nel MAR e viene effettuato un accesso in lettura nella memoria centrale.
2. La memoria copia il dato del registro appena letto nel MDR.
3. Il contenuto del MDR viene ricopiato nel registro IR.
4. Nel frattempo PI si aggiorna automaticamente al valore della locazione successiva (ovvero già punta alla successiva istruzione da usare al prossimo fetch).



Fase di Operand Assembly

Fase di Operand Assembly

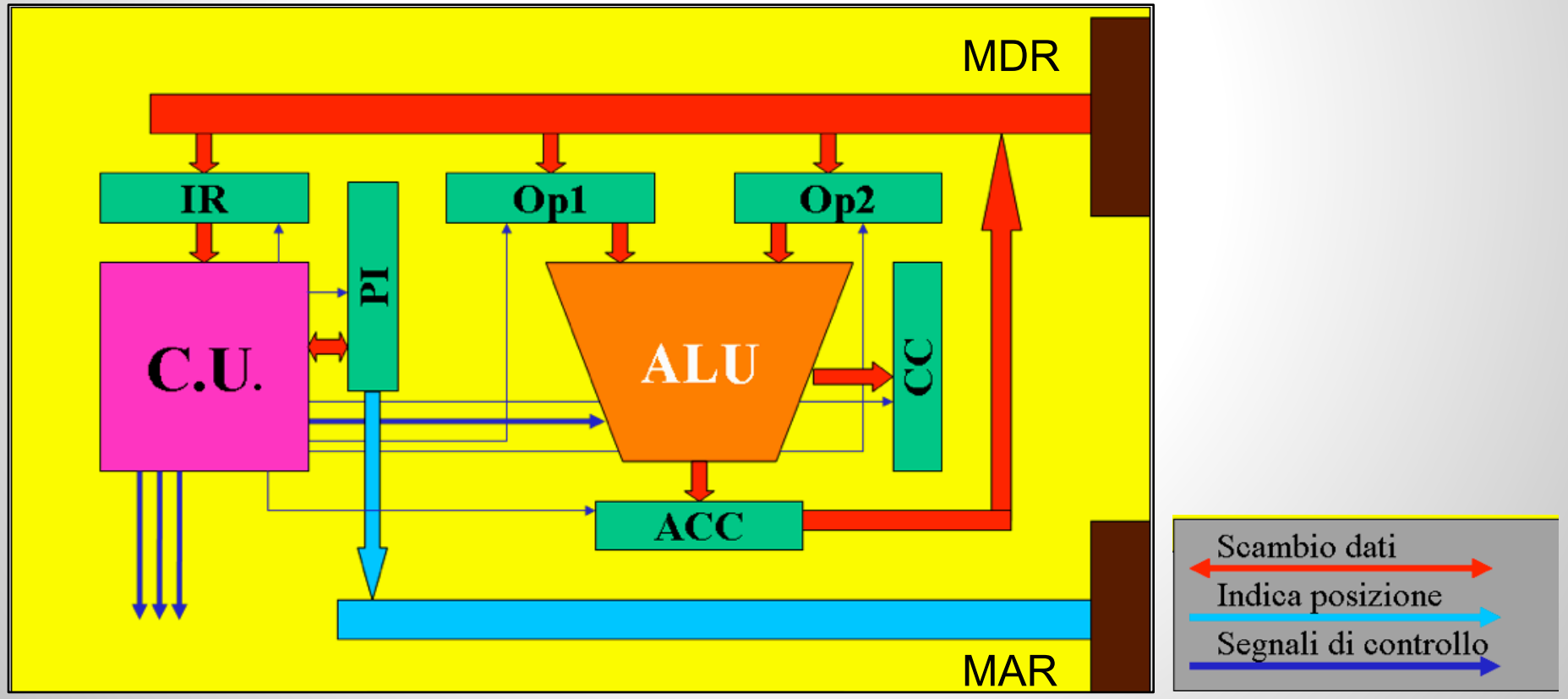
1. L'istruzione viene interpretata dalla CU: questa può comportare ulteriori accessi in memoria per caricare altri dati se l'istruzione completa non è contenuta nel dato prelevato in precedenza.
2. Se, per esempio, l'istruzione da eseguire è una somma, i registri Op1 e Op2 verranno caricati con i numeri da sommare.



Fase di Execute

Fase di Execute

1. In questa fase la CU esegue l'istruzione richiesta.
2. Se l'istruzione richiede l'attivazione della ALU (es.: una somma), la CU attiva l'ALU, che esegue l'operazione: il risultato prodotto è conservato in uno dei registri (es.: ACC) e l'ALU aggiorna il CC.



Istruzioni

- Le istruzioni sono operazioni elementari:
  - trasferimento dati da un registro ad un altro
    - da memoria a memoria,
    - da memoria a registri della CPU o viceversa,
    - da memoria a output, da input a memoria;
  - operazioni aritmetiche o logiche eseguite dall'ALU (e.g. somma e moltiplicazione);
  - controllo di condizioni riportate dal registro CC o deducibili dal confronto di due registri (e.g. operazione di confronto).
  - Operazioni che determinano la prossima istruzione da eseguire, in caso in cui sia diversa da quella successiva contigua
- Il repertorio delle istruzioni che un microprocessore è capace di eseguire ne determina l'architettura:
  - E.g. i processori della Intel hanno un set di istruzioni diverso dai processori ARM

Istruzione in Linguaggio Macchina

- Concettualmente un'istruzione è una quadrupla:

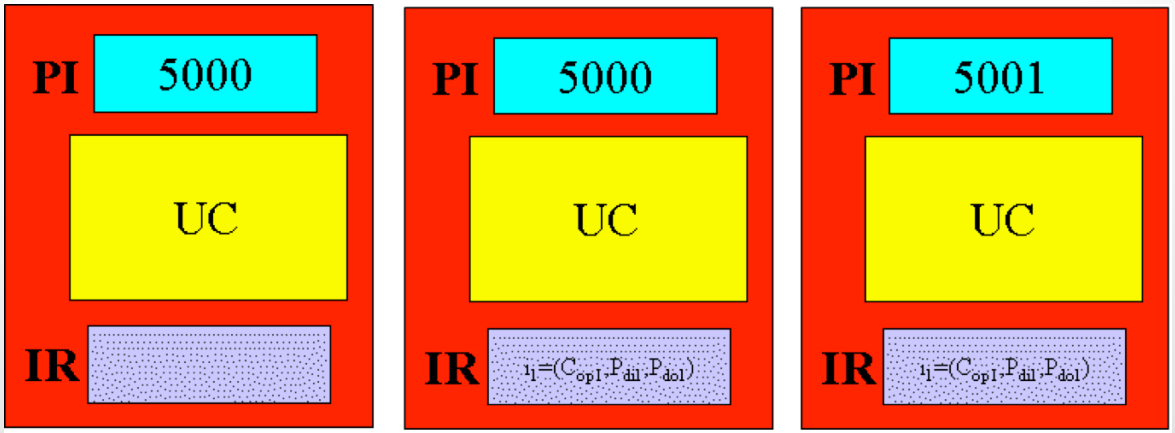
$$i = (C_{op}, P_{di}, P_{do}, P_{is})$$

1. C_{op} è il codice che indica l'operazione da compiere;
   - l'insieme dei C_{op} prendere il nome di **repertorio di istruzioni** e dipende dalla specifica architettura di CPU (es.: Intel x86, ARM)
2. P_{di} sono i dati (e/o i riferimenti ai registri che contengono i dati) che sono in input all'operazione C_{op}
3. P_{do} sono i riferimenti ai registri che devono contenere i dati in output all'operazione C_{op}
4. P_{is} è un riferimento che è usato per determinare l'istruzione da svolgere al termine di quella corrente (in base al valore di PI).

- Non tutte le operazioni hanno dati in input e/o dati in output!

Allocazione contigua di istruzioni

- Per garantire che il PI si aggiorni in moto tale da caricare la successiva istruzione del programma, il programmatore dispone tutte le istruzioni in memoria ad indirizzi consecutivi
  - Vi sono istruzioni speciali che permettono di modificare arbitrariamente il valore di PI con un indirizzo di memoria diverso.



T1: invio alla memoria del contenuto di PI

T2: istruzione inserita in IR

T3: aggiornamento di PI

Gli operandi dell'istruzione: modi di Indirizzamento (1/2)

- Nella costruzione dell'indirizzo di un operando di una istruzione le tecniche di indirizzamento più diffuse sono:
  - indirizzamento **immediato** che indica che il valore è contenuto già nell'istruzione come costante (dato presente nell'istruzione stessa);
    - E.g. **LOAD A; 5** (carica nel registro A il valore 5)
  - indirizzamento **diretto** con il quale viene riportato nell'istruzione qual è il registro interno del processore che contiene il valore o nel quale depositare il valore (dato presente in un registro interno del processore);
    - E.g. **ADD A; B** (carica nel registro A la somma tra A e B)

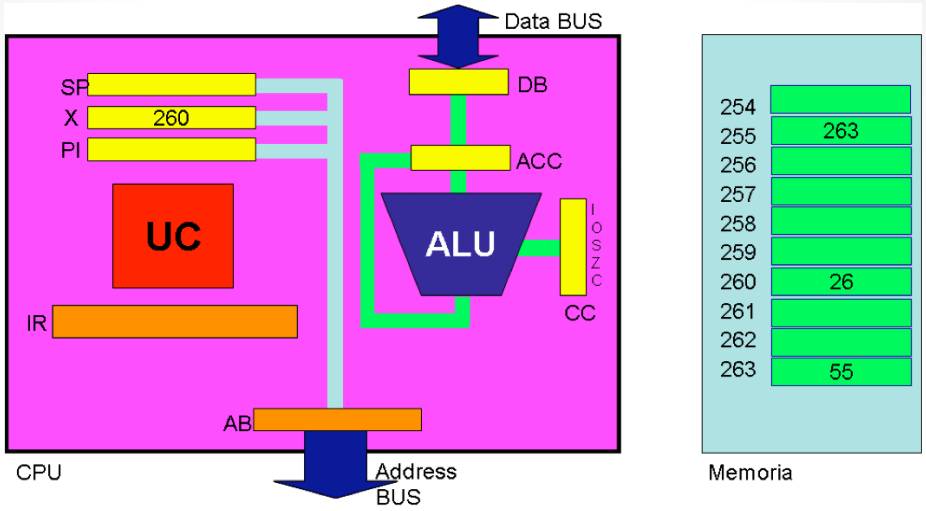
Gli operandi dell'istruzione: modi di Indirizzamento (2/2)

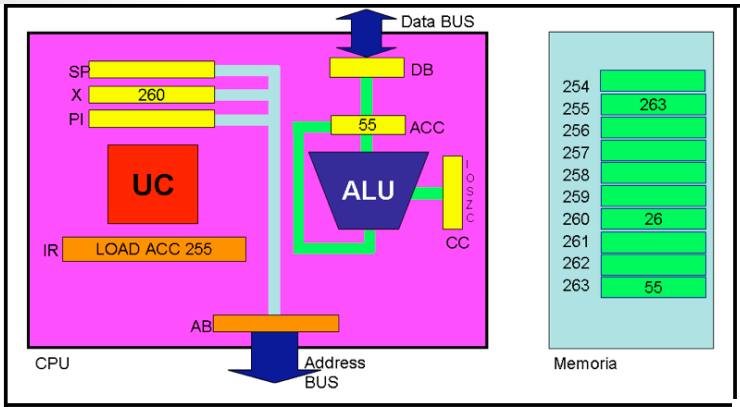
- indirizzamento **indiretto** che riporta nell'istruzione qual è il registro interno del processore al cui interno è specificato l'indirizzo del registro di memoria dal quale prelevare un valore o nel quale depositare un valore (dato presente in memoria);
  - E.g. **STORE 0; (A)** (carica il valore 0 all'indirizzo di memoria contenuto nel registro A)
- indirizzamento **assoluto** con il quale l'indirizzo del registro di memoria che contiene il valore o nel quale depositare il valore è specificato direttamente nell'istruzione come costante (dato presente in memoria).
  - E.g. **STORE 0; (15)** (carica il valore 0 all'indirizzo di memoria 15)

Modi di Indirizzamento: considerazioni

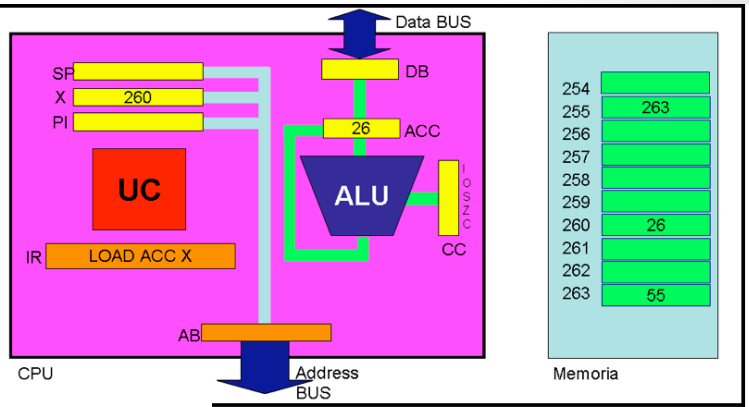
- Le diverse tecniche di indirizzamento vengono indicate nell'istruzione o diversificando il codice operativo o aggiungendo dei bit di codifica appositi il cui valore indica alla UC come costruire l'indirizzo.
- La specifica del modo di indirizzamento è fondamentale perché si possano eseguire le istruzioni:
  - Come precedentemente illustrato, i campi di una istruzione possono codificare simboli che però vanno correttamente interpretati:
    - Come valore immediato
    - Come valore di spiazzamento
    - Come valore per indirizzamento registro
    - Come valore per indirizzamento memoria

Modi di indirizzamento: esempi





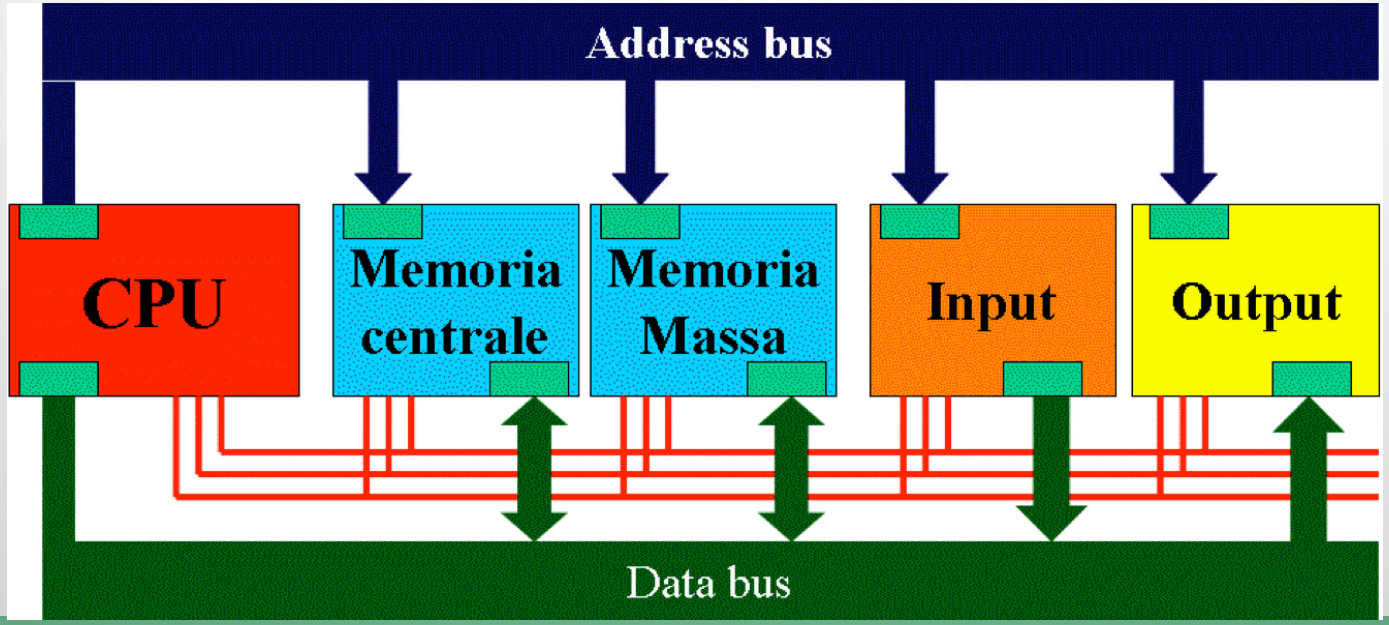
Indiretto



Assoluto

Il Bus

- I componenti dell'architettura di un calcolatore comunicano tra loro sfruttando un media condiviso, chiamato Bus.
- Fisicamente, il bus costituito da uno o più connessioni (e.g. fili di rame, piste conduttive) su cui transitano più informazioni (bit) parallelamente.
- A seconda delle informazioni trasportate si hanno:
  - *bus dati (data bus)*
  - *bus indirizzi (address bus)*
  - *bus comandi o di controllo (command o control bus)*



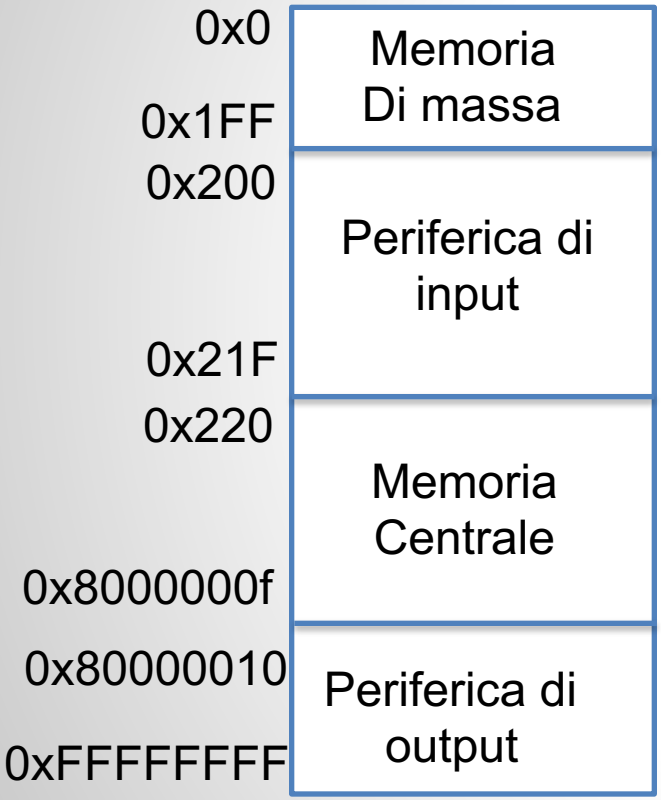
Address Bus

- L'address bus serve alla CU per comunicare l'indirizzo interessato da una operazione di lettura o scrittura.
- L'indirizzo si riferisce all'intero spazio di memoria disponibile all'interno del calcolatore, periferiche incluse:
  - Infatti, la CU può comunicare con la memoria centrale, ma anche con dispositivi di input/output utilizzando lo stesso identico meccanismo di indirizzamento.
  - Si dice che la memoria ha dei "buchi", che corrispondono ad indirizzi virtuali assegnati alle periferiche
  - Tutti i componenti del sistema (memoria, input, output, memoria di massa, etc.) devono essere dotati della capacità di riconoscere sull'address bus il proprio indirizzo.

Attraverso l'address bus la CU effettua la selezione del dispositivo a cui sono rivolti i comandi e i dati.

Indirizzamento di un Calcolatore

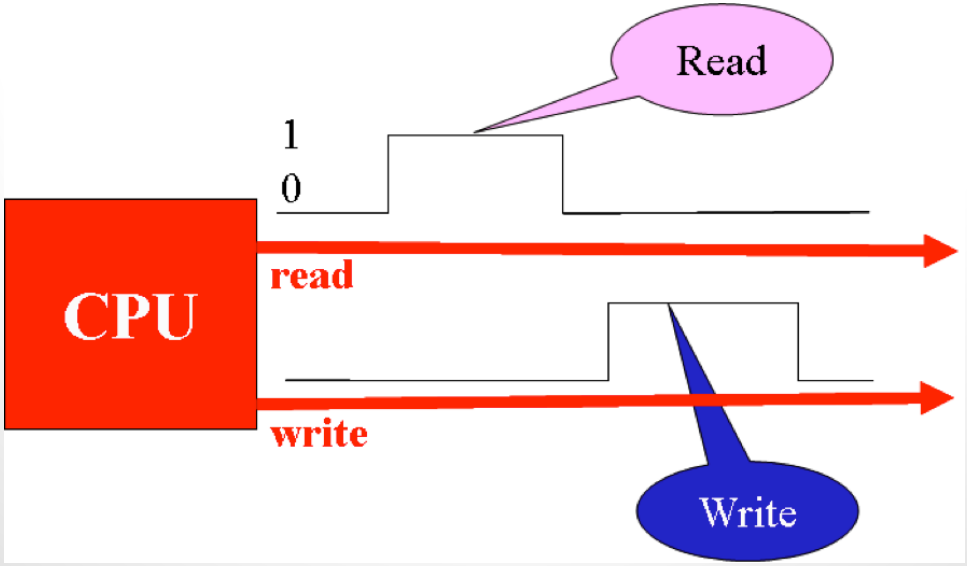
- Esempio di architettura di memoria con bus da 32 bit



- Solo una parte dello spazio di indirizzamento è dedicata alla memoria centrale;
- La restante parte è dedicata alle periferiche (e.g. di input ed output), incluse altre memorie, come per esempio la memoria di massa.
  - Questa tecnica è detta “memory mapped”, poiché tutte le periferiche sono raggiungibili dal medesimo spazio di memoria

Control Bus e Data Bus

- Il control bus è utilizzato dai componenti connessi al bus per arbitrare l'operazione da effettuare ad un dato indirizzo:
  - Infatti, tipici segnali del control bus sono quelli di *read e write,* mediante i quali la CU indica al dispositivo selezionato con l'address bus se devono leggere un dato dal data bus (read) o scriverlo su di esso (write).
- Il data bus, invece, è utilizzato per la trasmissione effettiva del dato, sia in caso di lettura (operazione di LOAD) che di scrittura (STORE) da parte della CPU.



Modello di Stato/Controllo/Dato

Master e Slave di Bus

- La CPU è l'unico componente che può attivare il bus per operazioni di lettura e scrittura:
  - Infatti la CPU ricopre sempre il ruolo di **master**.
- Le periferiche, e.g. memorie e dispositivi di input/output, devono *ascoltare* l'address bus per attivarsi quando su di esso compare il proprio indirizzo identificativo (ruolo di **slave**):
  - nel caso della memoria l'attivazione avviene quando viene riconosciuto l'indirizzo corrispondente ad uno dei registri di cui essa è composta;
  - il dispositivo attivo deve interpretare i segnali del control bus per eseguire i comandi della CU;
  - le memorie prelevano dati dal data bus o immettono dati in esso in funzione del comando impartito dalla CU;
- I dispositivi di input possono solo immettere dati sul data bus;
- Viceversa i dispositivi di output possono solo prelevare dati dal data bus.

Bus Seriale e Parallelo

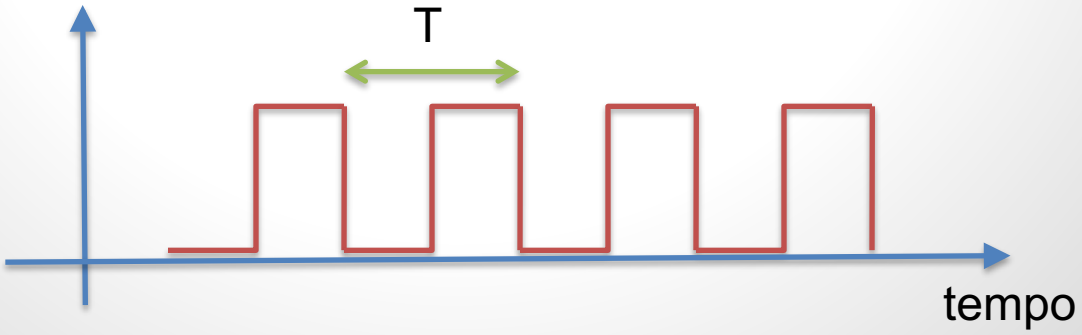
- Un bus costituito da una singola linea di trasmissione è chiamato bus *seriale*, cioè su di esso i bit transitano uno dietro l'altro.
- Un bus costituito da n linee parallele è chiamato bus *parallelo* perché su di esso transitano n bit alla volta.
  - Tipici sono i bus a parallelismo 8, 32, 64, etc. sui quali si possono trasferire rispettivamente 8, 32, 64 bit alla volta.
- L'address e il data bus sono paralleli e le loro dimensioni caratterizzano i sistemi di calcolo.
  - Il numero di bit dell'address bus indica la *capacità di indirizzamento* della CPU: ossia la sua capacità di gestire la dimensione della memoria centrale e il numero di dispositivi di input ed output (**spazio di indirizzamento**).
    - Infatti un address bus con n bit consente di selezionare un registro tra 2^n.
  - La dimensione del data bus condiziona invece la velocità di scambio (chiamata **throughput**) delle informazioni tra i diversi dispositivi in quanto con m fili solo m bit possono viaggiare contemporaneamente.

Gestione Input Output

- tecnica *memory-mapped*
  - … l'UC usa le stesse istruzioni utilizzate per leggere e scrivere in memoria anche per accedere ai dispositivi di I/O.
  - I dispositivi di I/O hanno quindi dei propri indirizzi che devono essere riservati e non sovrapposti a quelli usati per la memoria. I dispositivi di I/O controllano il bus indirizzi e rispondono solo quando riconoscono un indirizzo a loro assegnato
    - Il vantaggio dell'uso del memory-mapped è che, non richiedendo da una parte hardware aggiuntivo per la gestione della periferia e dall'altra un insieme di istruzioni specifiche, consente la realizzazione di CPU con una complessità inferiore, più economiche, veloci e facili da costruire.
- Tecnica *I/O-mapped*.
  - … vengono invece usate istruzioni specifiche per l'esecuzione dell'input/output.
  - I dispositivi di I/O hanno uno spazio indirizzi separato da quello della memoria, e un segnale del control bus serve alla UC per specificare se si tratta di un accesso alla memoria o ad un dispositivo periferico.

Il segnale di Clock

- Le attività di tutti i dispositivi vengono sincronizzate tra loro mediante un orologio interno, detto clock, che scandisce i ritmi di lavoro.
- Il *clock* è un segnale periodico di periodo fisso, vale a dire un'onda quadra caratterizzata da un periodo T (detto ciclo) misurato in **secondi**, il cui reciproco rappresenta la frequenza f (f=1/T) misurata in **Hertz** (Hz).
  - Ad esempio un clock composto da 10 cicli al secondo ha la frequenza f = 10 Hz e il periodo T= 100ms.
    - La frequenza dei clock presenti nei moderni sistemi spazia dai MHz (1 MHz corrisponde a un milione di impulsi al secondo) ai GHz (1 GHz corrisponde a un miliardo di impulsi al secondo).



Clock e velocità di elaborazione

- La velocità di elaborazione di una CPU dipende dalla frequenza del suo clock, infatti più veloce è il clock maggiore è la velocità di esecuzione.
- Alla frequenza del clock è legato il numero di operazioni elementari che vengono eseguite nell'unità di tempo dalla CU
  - E.g., se si assume che ad ogni ciclo di clock corrisponde esattamente l'esecuzione di una sola operazione, allora la frequenza del clock indica il numero di operazioni che vengono eseguite nell'unità di tempo dalla UC.
    - Con un clock a 3 GHz si ha che il processore è in grado di eseguire 3 miliardi di operazioni al secondo.
    - In realtà tale ipotesi non è sempre vera in quanto l'esecuzione di un'operazione può richiedere più cicli consecutivi di clock, sia per la complessità delle operazioni che per la lentezza dei dispositivi collegati alla CPU.

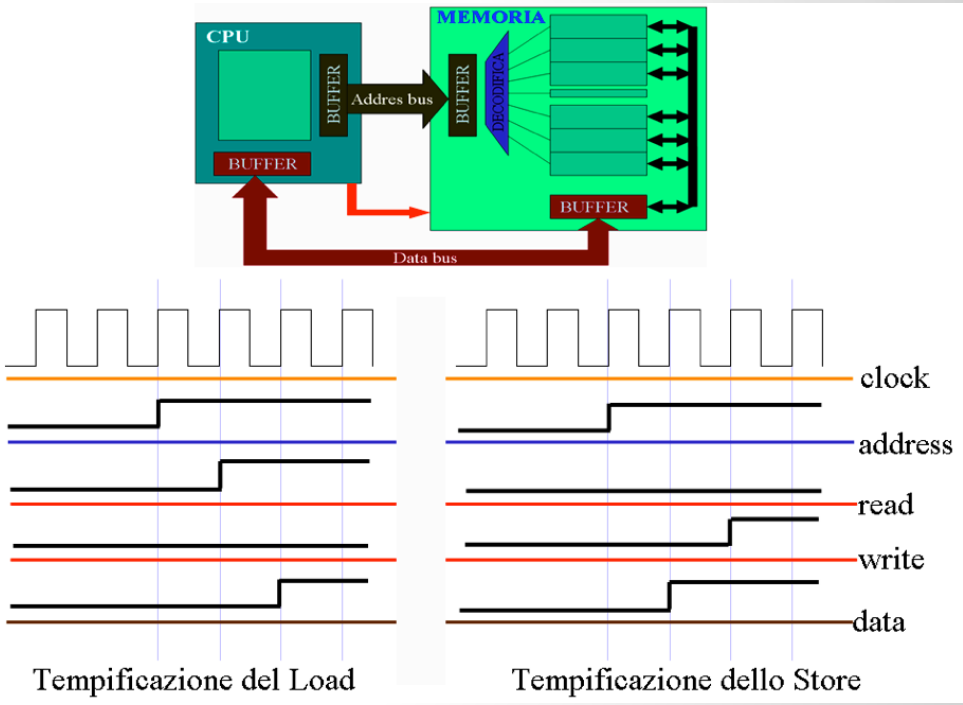
Esempio di Tempificazione del Clock

LOAD

- con il primo ciclo di clock si pone l'indirizzo del registro di memoria di cui vuole leggere il contenuto sull'address bus;
- con il secondo ciclo si segnala alla memoria che si tratta di una operazione di read;
- con il terzo ciclo si prende il dato dal data bus dove la memoria ha provveduto a depositarlo.

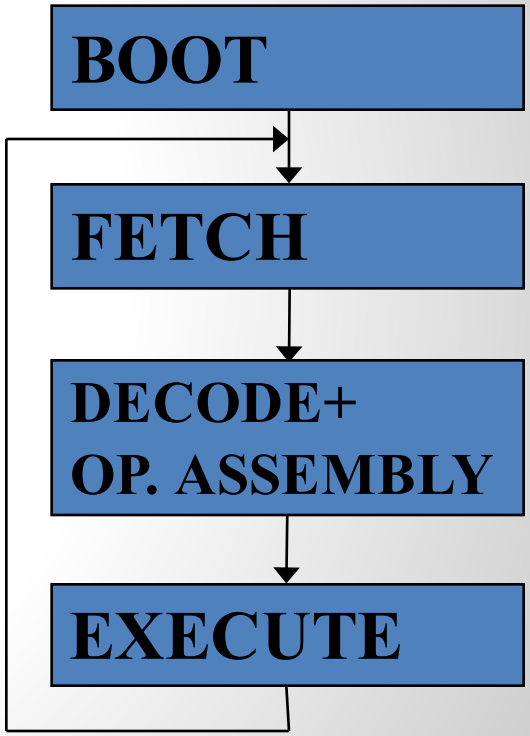
STORE

- con il primo ciclo di clock si pone l'indirizzo del registro di memoria di cui vuole leggere il contenuto sull'address bus;
- con il secondo ciclo si deposita il dato sul data bus;
- con il terzo ciclo si segnala alla memoria che si tratta di una operazione di write e quindi che il dato è pronto per essere depositato nel registro selezionato.



Evoluzione del modello di Von Neumann: le Interruzioni

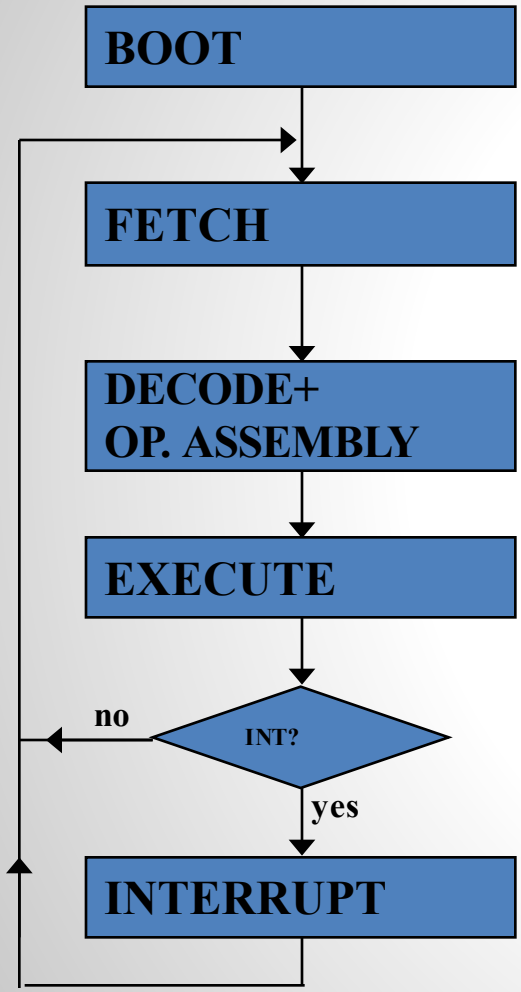
- Il ciclo di esecuzione previsto dal modello di Von Neumann non prevede alcuna eccezione di flusso, rendendo impossibile però alcune operazioni vitali, come:
  - Interrompere l'esecuzione di un programma a favore di un altro;
  - Gestire gli errori di esecuzione di un programma (e.g. un programma che divide per 0, un programma che esegue un ciclo di esecuzione infinito).



Evoluzione del modello di Von Neumann: le Interruzioni

- La soluzione comunemente adottata consiste nel prevedere che, in caso di eventi eccezionali, l'esecuzione del programma venga **interrotta** per gestire tali eventi:
  - Evento **sincrono**: è una condizione eccezionale che si verifica durante l'esecuzione di una delle 3 fasi del ciclo di Von Neumann, cioè causato dall'esecuzione di una istruzione del programma;
  - Evento **asincrono**: è **un evento esterno** che non dipende dall'esecuzione del programma;
- E.g. di eventi sincroni:
  - Divisione per 0;
  - Istruzione non codificata correttamente;
- E.g. eventi asincroni:
  - Inserimento di un carattere dalla tastiera;
  - Reset del processore.

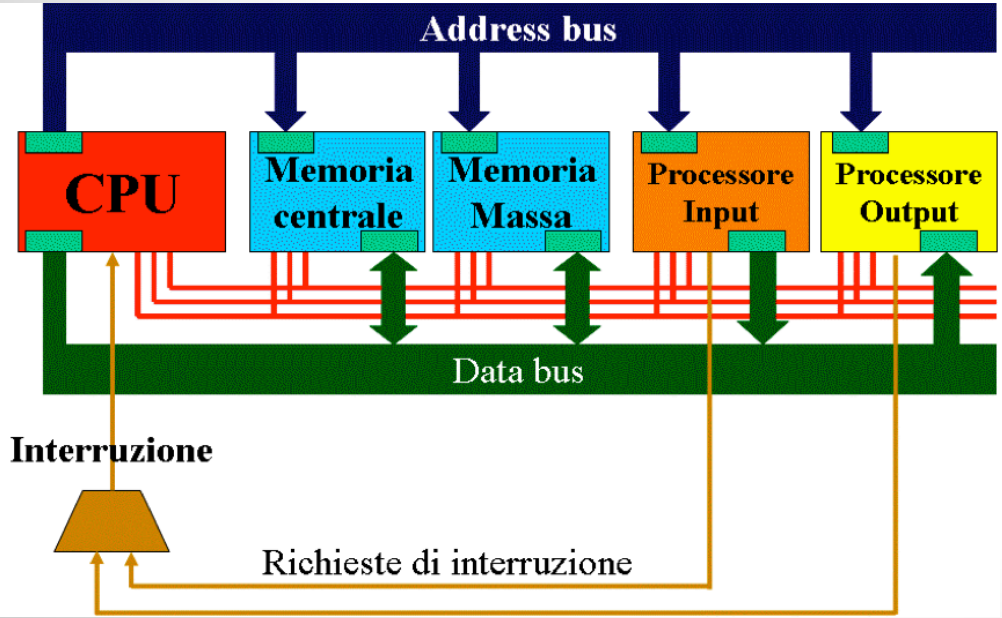
Ciclo del processore con interruzioni



- Affinché la CU possa interrompere il ciclo di Von Neumann, il registro di condizione CC deve essere dotato di un bit che segnala la presenza di una interruzione.
- La CU, dunque, controlla tale bit al termine dell'esecuzione di ogni istruzione:
  - se è uguale zero (assenza di interruzione) procede normalmente con il prelievo dell'istruzione successiva;
  - in caso contrario comincia l'esecuzione di un diverso programma, detto **ISR** (interrupt service routine), che ha come compito primario di gestire l'interruzione;
    - Nel caso si accorga della presenza di più richieste stabilisce quale servire per prima secondo criteri di importanza o priorità di intervento;
  - Quando la ISR termina, la CU riprende ad eseguire il programma precedente nel punto in cui era stato interrotto.

Le interruzioni esterne: Gestione delle Periferiche

- Introducendo il concetto di interruzione, la CPU può attivare una periferica del sistema disinteressandosi delle sue specifiche attività:
  - Quando la periferica (e.g. periferica di I/O) termina il suo compito, avanza una richiesta di interruzione al processore centrale e aspetta che gli venga rivolta attenzione (e.g. la tastiera interrompe per salvare in memoria un carattere);
  - Mentre le periferiche lavorano, la CPU può lavorare anch'essa continuando ad eseguire programmi, sino a successive interruzioni.

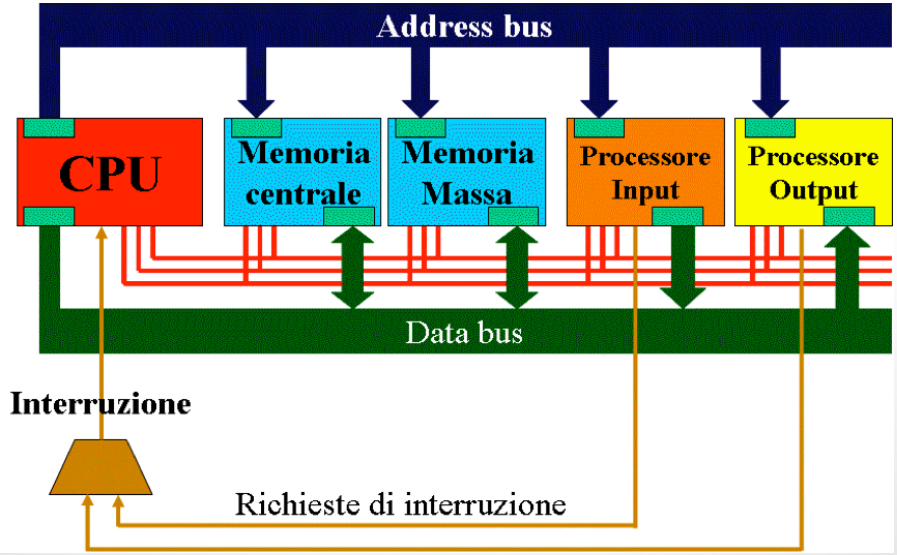


Sequenza di passi:

1. Il Dispositivo interrompe la CPU;
2. il Processore interrompe il programma in esecuzione;
3. il Dispositivo viene informato che l'interrupt è stato ricevuto (interrupt acknowledgment);
4. Viene eseguita la procedura di servizio dell'interruzione (ISR);
5. Si ripristina il Programma originale

Identificazione dei dispositivi (1/2)

- Se ci sono più dispositivi che possono interrompere, il processore deve essere in grado di identificare il dispositivo che ha generato l'interruzione, poiché ad ogni specifica interruzione corrisponde una ISR diversa.
  - I dispositivi hanno una linea comune attraverso la quale segnalano richieste di interruzioni (INT)
  - Quando INT è alto si pone il problema di identificare da quale dispositivo è partita la richiesta

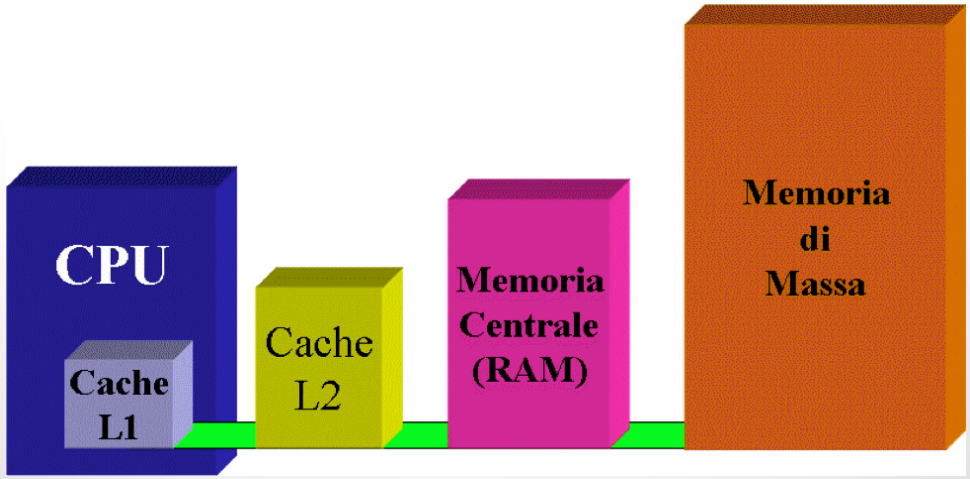


Identificazione dei dispositivi (2/2)

- INT potrebbe alzarsi anche in seguito a richieste “contemporanee” di due o più dispositivi
- Esistono diverse soluzioni a questo problema, e tutte impiegano un misto di hardware e di software. Tutte le soluzioni dipendono fortemente sia dall’architettura del sistema che da quella del processore
  - **Polling**
    - Si interroga il registro di stato di tutti i dispositivi. L’ordine con cui vengono interrogati i dispositivi potrebbe dipendere dalla posizione degli stessi (daisy chain) o dalla loro priorità (stabilita a priori)
  - **Interrupt vettorizzato o autovettorizzato**
    - Il Dispositivo manda un identificativo in risposta al segnale di ack oppure il dispositivo viene direttamente associato ad un livello di priorità in base al quale parte la ISR

Gerarchie di Memorie

- Lo schema della memoria nell'architettura di Von Neumann, dal punto di vista della CPU, è di tipo gerarchico.
- Tale scelta deriva dalle seguenti motivazioni:
  - Nella gerarchia, i livelli più prossimi alla CPU sono quelli più veloci, ma sono anche quelli con elevato costo, dunque hanno dimensioni più piccole;
  - I livelli più lontani sono quelli che mostrano una capacità massima ed anche tempi di accesso maggiori;
  - Consente di offrire ai programmi l'illusione di avere una memoria grande e veloce;



Evoluzione del modello di Von Neumann: le Cache

- Per ridurre i tempi di trasferimento dalla memoria centrale ai registri interni della CPU, viene replicata una porzione di memoria e posta tra memoria e CPU stessa. Tale memoria, molto veloce, viene chiamata *cache* e fa da buffer per il prelievo di informazioni dalla memoria centrale:
  - Con operazioni particolari istruzioni e dati vengono trasferiti dalla memoria centrale nella cache secondo la capacità di quest'ultima.
    - La CU procede nelle tre fasi del suo ciclo al prelievo di istruzioni e operandi dalla cache.
    - Quando la CU si accorge che il prelievo non può avvenire, scatta un nuovo travaso dalla memoria centrale.
  - Se la cache è interna alla CPU viene detta di primo livello (L1);
  - Le cache di secondo livello (L2) sono invece esterne e solitamente un più lente di quelle di primo livello, ma più veloci della memoria centrale.
    - la cache L2 risulta circa 4 o 5 volte più lenta della cache L1 mentre la RAM lo è addirittura 20 o 30 volte.

Classificazione dei Processori

- I processori possono essere caratterizati dai seguenti parametri principali:
  - **Parallelismo** esterno espresso come numero di bit trasferiti o prelevati in un singolo accesso in memoria (8, 16, 32, 64,...) e caratterizzanti quindi il suo data bus;
  - **Capacità di indirizzamento** legata alla dimensione in bit del suo address bus
  - Numero, tipo e parallelismo dei registri interni;
  - **Tecniche di indirizzamento** intese come la modalità con la quale costruire **l'indirizzo logico con il quale prelevare o salvare il valore dell'operando** di una istruzione;
  - Gestione delle periferiche di **input ed output**;
  - **Repertorio delle istruzioni** inteso come numero e tipo di istruzioni costituenti il suo linguaggio macchina;
  - **Tempi necessari all'esecuzione di alcune istruzioni fondamentali** come l'addizione da utilizzare per la valutazione del MIPS con il quale effettuare confronti sulle prestazioni.

Processori RISC e CISC

- **CISC (Complex Instruction Set Computer)**
  - Il linguaggio macchina ha istruzioni con potenza espressiva prossima a quella dei linguaggi di programmazione di alto livello;
  - Sono caratterizzati quindi da un ampio repertorio di istruzioni:
    - Sviluppo di programmi più semplice;
    - Codici ridondanti;
    - Maggiore complessità costruttiva;
- **RISC (Reduced Instruction Set Computer).**
  - Repertorio costituito da un ridotto ed essenziale insieme di istruzioni al fine di ottenere processori più veloci e di costo ridotto, data la minore complessità del loro progetto;
  - L'obiettivo fondamentale dell'approccio RISC è di disporre di un insieme fondamentale di istruzioni per ridurre al minimo il numero dei cicli di macchina (clock) necessari per loro esecuzione.
    - **Tutte le istruzioni RISC fondamentali hanno la stessa durata (un ciclo macchina), la stessa lunghezza e lo stesso formato**

Linguaggio macchina e linguaggio assembly

- Un'istruzione macchina è una stringa binaria suddivisa in campi, diversi nel formato e nel significato a seconda dell'istruzione;
- Nel riferirsi alle istruzioni macchina si preferisce la rappresentazione simbolica a quella numerica, costituita dal **linguaggio assembly** (o assembler)

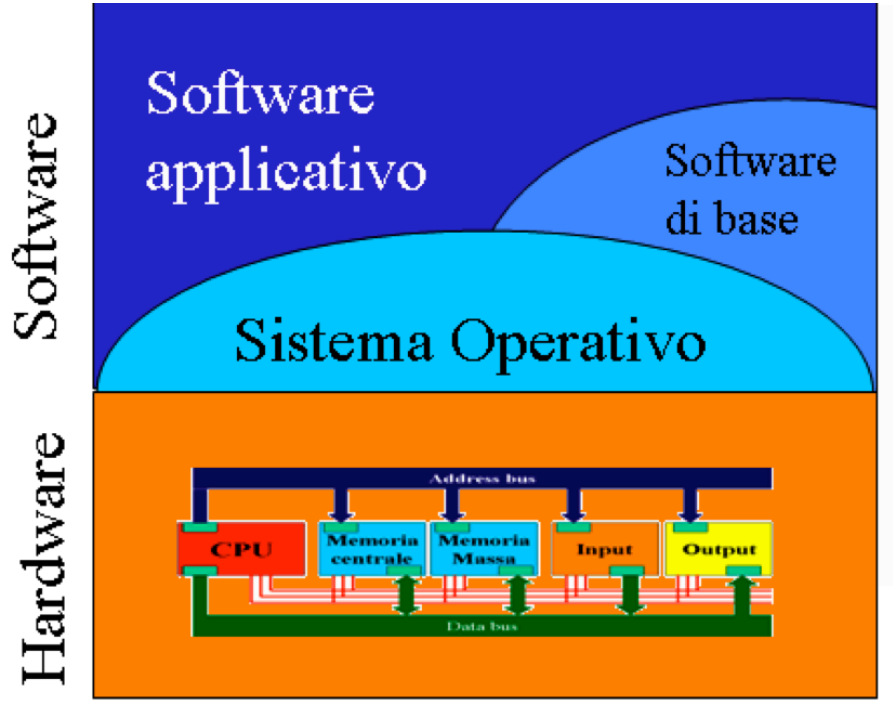
- E.g. MOVE R1; R2 → 00 001 010
- E.g. LOAD R0; (R2) → 01 000 010

Assemblatore

- L'assemblatore è un programma che esegue la traduzione di un programma, scritto in linguaggio assemblativo, in linguaggio macchina;
- E' il più semplice traduttore di linguaggi presente in informatica:
  - fa corrispondere ai codici mnemonici del codice operativo il rispettivo codice binario;
  - converte da decimale a binario indirizzi e valori dei dati;
  - determina gli indirizzi delle etichette associate alle istruzioni;
  - converte dati alfanumerici nella loro rappresentazione binaria.

Hardware vs. Software

- programmi che servono a tutti gli utenti del sistema
  - classificati come *software di base*
    - sistemi operativi e traduttori dei linguaggi di programmazione
- programmi che risolvono problemi specifici
  - *software applicativo.*



Il Sistema Operativo

- Il sistema operativo è un insieme di programmi che garantisce la gestione delle risorse hardware al fine:
  - **ottimizzare** i parametri prestazionali;
  - Gestire le **risorse hardware comuni** alle applicazioni (e.g. memoria, CPU, periferiche);
  - **semplificare** lo sviluppo di applicativi software;
- I primi calcolatori non avevano il sistema operativo.
  - In essi il programmatore doveva prevedere tutto
  - Al termine dell'esecuzione del programma il programmatore o l'operatore del sistema doveva provvedere ad un nuovo caricamento in memoria ed ad una successiva attivazione.
- Con il sistema operativo il passaggio da una applicazione ad un'altra è svolto in **automatico**:
  - La CPU si trova così ad eseguire i programmi del sistema operativo in alternanza con quelli applicativi.

Middleware

- E' il software che fornisce *un'astrazione di programmazione* che maschera l'eterogeneità di elementi sottostanti
  - reti, hardware, sistemi operativi, linguaggi di programmazione
- Il middleware definisce una macchina generalizzata fissandone modalità di interazione con le applicazioni.