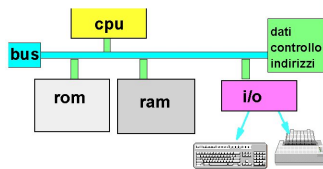
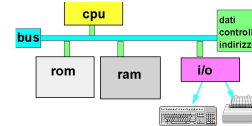


ARCHITETTURA DI UN ELABORATORE



Ispirata al modello della **Macchina di Von Neumann** (Princeton, Institute for Advanced Study, anni '40).

MACCHINA DI VON NEUMANN



UNITÀ FUNZIONALI fondamentali

- Processore (CPU)
- Memoria Centrale (RAM & ROM)
- Unità di I/O (ingresso / uscita)
- Bus di sistema

CPU & MEMORIA



- ALU (Arithmetic & Logic Unit)
- Unità di Controllo
- Registri

UNITÀ DI ELABORAZIONE (CPU)

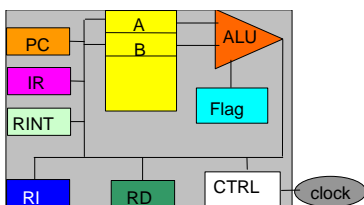
ALU (Arithmetic / Logic Unit)

Esegue le operazioni aritmetiche e logiche elementari

Unità di Controllo (*Control Unit*): controlla e coordina l'attività della CPU. (In particolare, controlla il trasferimento dei dati tra memoria e registri e la decodifica e l'esecuzione delle istruzioni)

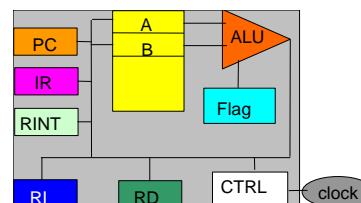
I **registri** sono *locazioni* usate per memorizzare dati, istruzioni, o indirizzi **all'interno della CPU**. L'accesso ai registri è *molto veloce*.

UNITÀ DI ELABORAZIONE (CPU)



Il **clock** dà la base dei tempi necessaria per mantenere il sincronismo fra le operazioni

UNITÀ DI ELABORAZIONE (CPU)



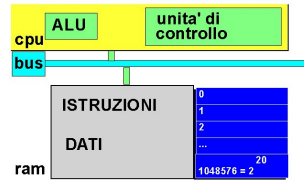
I **registri** (qui **A, B, PC, Flag,...**) sono *locazioni* usate per memorizzare dati, istruzioni, o indirizzi **all'interno della CPU**. L'accesso ai registri è *molto veloce*.

UNITÀ DI ELABORAZIONE (CPU)



La memoria centrale è una collezione di celle *numerate*, che possono contenere **DATI** e **ISTRUZIONI**.
Le istruzioni sono disposte in memoria in *celle di indirizzo crescente*.

UNITÀ DI ELABORAZIONE (CPU)

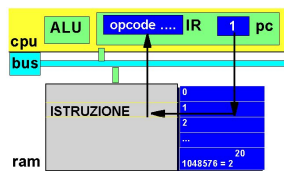


L'unità di controllo fa funzionare l'elaboratore.
Da quando viene acceso a quando è spento, essa esegue in continuazione il ciclo di *prelievo / decodifica / esecuzione* (**fetch / decode / execute**).

IL CICLO fetch / decode / execute

FETCH

- si accede alla **prossima istruzione** (cella il cui indirizzo è contenuto nel registro **PC**) ...
- ... e la si porta dalla memoria centrale, memorizzandola nel **Registro Istruzioni (IR)**



IL CICLO fetch / decode / execute

DECODE

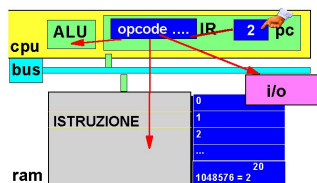
- si **decodifica** il tipo dell'istruzione in base al suo **OpCode** (codice operativo)

OpCode	OpCode	Operazione
1	1	Somma
2	2	Sottrazione
...

IL CICLO fetch / decode / execute

EXECUTE

- si individuano i **dati** usati dall'istruzione
- si **trasferiscono** tali dati nei registri opportuni
- si **esegue** l'istruzione.



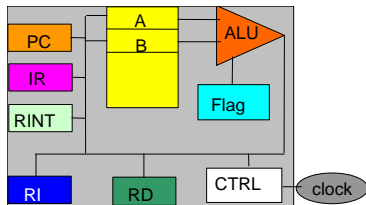
IL CICLO fetch / decode / execute

ATTENZIONE

Istruzioni particolari possono **alterare il prelievo delle istruzioni da celle consecutive**:

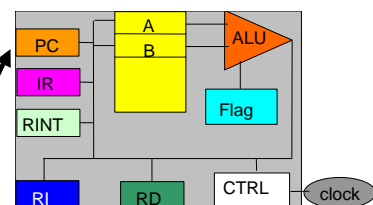
- istruzioni di **salto**
- istruzioni di **chiamata a sotto-programmi**
- istruzioni di **interruzione**

I REGISTRI



I registri sono *locazioni di memoria interne a CPU*, e come tali *molto veloci*.

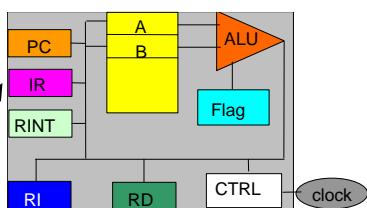
I REGISTRI



Program Counter (PC)

Indica l'indirizzo della cella di memoria che contiene la prossima istruzione da eseguire

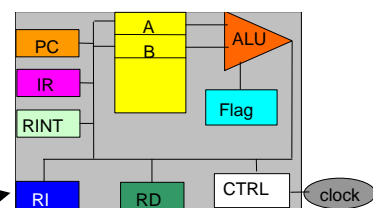
I REGISTRI



Instruction Register (IR)

Contiene l'istruzione da eseguire.

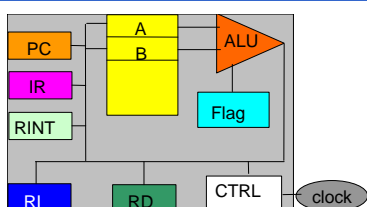
I REGISTRI



Registro Indirizzi (RI)

Contiene l'indirizzo della cella di memoria da selezionare per il trasferimento di un dato con la CPU

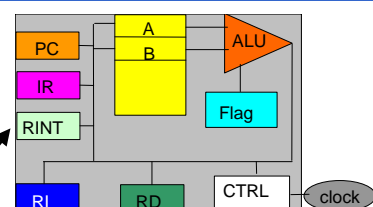
I REGISTRI



Registro Dati (RD) o Accumulatore

Contiene il dato attualmente oggetto di elaborazione e il risultato al termine dell'esecuzione

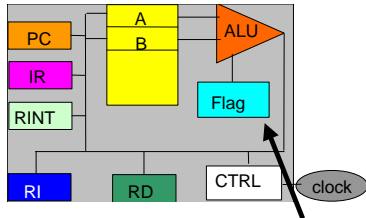
I REGISTRI



Registro Interruzioni (RINT)

Serve per scopi particolari (non discussi)

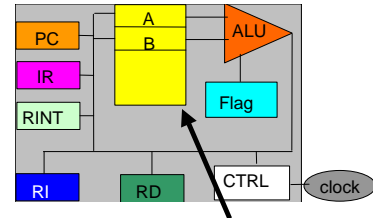
I REGISTRI



Registro dei Flag (Flag)

Ogni flag indica la presenza/assenza di una proprietà nell'ultimo risultato generato dalla ALU. Altri bit riassumono lo stato del processore.

I REGISTRI



Registri di uso generale (A,B,C,...)

Sono usati per contenere sia dati (in particolare, operandi di operazioni aritmetico/logiche) sia indirizzi.

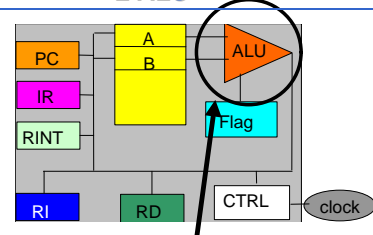
MULTITASKING

Poiché i registri compendiano *tutto lo stato dell'elaborazione* di un certo processo,

- salvando in memoria il contenuto di tutti i registri è possibile **accantonare un processo per passare a svolgerne un altro**
- ripristinando dalla memoria il contenuto di tutti i registri precedentemente salvati è possibile **ripristinare lo stato di un processo accantonato, riprendendone l'esecuzione come se nulla fosse accaduto.**

Questa possibilità è ciò che consente a un sistema operativo di eseguire *più compiti* "allo stesso tempo"

L'ALU



Esegue operazioni aritmetiche, logiche e confronti sui dati della memoria centrale o dei registri. Può essere semplice oppure (più spesso) molto complessa e sofisticata.

L'ALU (segue)

ESEMPIO SEMPLICE:

ALU in grado di eseguire **somma, sottrazione, prodotto, divisione** con due operandi contenuti nei registri A e B.

1. I due operandi vengono caricati nei registri A e B;
2. La ALU viene attivata da un comando inviato dalla CPU che specifica il tipo di operazione;
3. Nel registro A viene caricato il risultato dell'operazione eseguita dalla ALU;
4. Il registro FLAG riporta sui suoi bit indicazioni sul risultato dell'operazione (riporto, segno, etc.).



Alterazione di due bit nel registro **Flag**: **carry** (riporto) e **sign** (segno)

LA MEMORIA CENTRALE

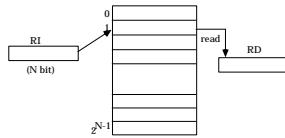
INDIRIZZAMENTO

- E' l'attività con cui l'elaboratore seleziona una particolare cella di memoria
- Per farlo, l'elaboratore pone l'indirizzo della cella desiderata nel Registro Indirizzi (RI).
 - **se il RI è lungo N bit, si possono indirizzare 2^N celle di memoria** (numerate da 0 a 2^N-1)
 - esempio: $N=10 \Rightarrow 1024$ celle.
- **Oggi, RI è lungo tipicamente 32 bit**
fi **SPAZIO INDIRIZZABILE di 4 Gb**

LA MEMORIA CENTRALE (2)

OPERAZIONI

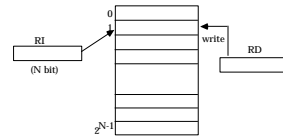
- **Lettura (Read):** il contenuto della cella di memoria indirizzata dal Registro Indirizzi è copiato nel Registro Dati.



LA MEMORIA CENTRALE (3)

OPERAZIONI

- **Scrittura (Write):** il contenuto del Registro Dati è copiato nella cella di memoria indirizzata dal Registro Indirizzi.



DISPOSITIVI DI MEMORIA

DISPOSITIVI FISICI

- **RAM:** Random Access Memory (ad accesso casuale): su di essa si possono svolgere operazioni sia di lettura che di scrittura
- **ROM:** Read Only Memory (a sola lettura): non volatili e non scrivibili dall'utente (che la ordina con un certo contenuto); in esse sono contenuti i dati e programmi per inizializzare il sistema
- **PROM:** Programmable ROM. Si possono scrivere soltanto una volta, mediante particolari apparecchi (detti programmatori di PROM).

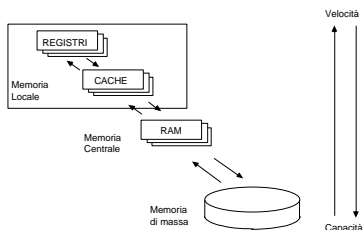
DISPOSITIVI DI MEMORIA (segue)

DISPOSITIVI FISICI (segue)

- **EPROM:** Erasable-Programmable ROM (si cancellano sottoponendole a raggi ultravioletti).
- **EEPROM:** Electrically-Erasable-PROM (si cancellano elettricamente).

Il **Firmware** è costituito da software memorizzato su ROM, EPROM, etc. (codice microprogrammato).

GERARCHIA DELLE MEMORIE



MEMORIE CACHE

PROBLEMA:

Sebbene la RAM sia veloce, non è **abbastanza** veloce da "star dietro" ai moderni processori.

CONSEGUENZA:

il processore *perde tempo* ad aspettare l'arrivo dei dati dalla RAM.

MEMORIE CACHE (2)

SOLUZIONE:

Inserire **tra processore e RAM** una **memoria particolarmente veloce** dove tenere i dati usati più spesso (**memoria cache**)

In questo modo,

- ♦ **la prima volta** che il microprocessore carica dei dati dalla memoria centrale, tali dati vengono caricati anche sulla cache
- ♦ **le volte successive**, i dati possono essere letti dalla cache (veloce) invece che dalla memoria centrale (più lenta)

MEMORIE CACHE (3)

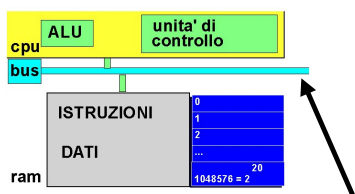
DUBBIO:

Ma se abbiamo memorie così veloci, **perché non le usiamo per costruire tutta la RAM?**

Semplice...
perché costano molto!!

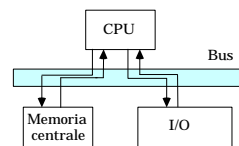
OGGI, la cache è spesso già presente dentro al processore (**cache di I° livello**), e altra può essere aggiunta (**cache di II° livello**)

BUS DI SISTEMA



Il **Bus di Sistema** interconnette la CPU, la memoria e le interfacce verso dispositivi periferici (I/O, memoria di massa, etc.)

BUS DI SISTEMA (2)

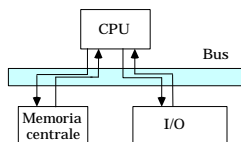


Il Bus collega **due unità funzionali alla volta**:

- una trasmette...
- ... e l'altra riceve.

Il trasferimento dei dati avviene o sotto il controllo della CPU.

BUS DI SISTEMA (3)



Il Bus è in realtà un insieme di linee diverse:

- bus dati (**data bus**)
- bus indirizzi (**address bus**)
- bus comandi (**command bus**)

BUS DI SISTEMA (4)

BUS DATI

- **bidirezionale**
- serve per trasmettere dati **dalla memoria o viceversa**.

BUS INDIRIZZI

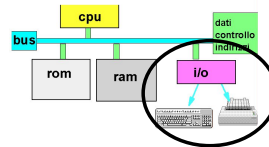
- **unidirezionale**
- serve per trasmettere **il contenuto del registro indirizzi alla memoria** (si seleziona una specifica cella su cui viene eseguita o un'operazione di lettura o una operazione di scrittura)

BUS DI SISTEMA (5)

BUS COMANDI

- **bidirezionale**
- tipicamente usato per **inviare comandi verso la memoria** (es: lettura o scrittura) o **verso una periferica** (es. stampa verso la stampante → interfaccia)
- può essere usato in alcuni casi per **inviare comandi verso il processore**

INTERFACCE DI I/O



Le interfacce sono molto diverse tra loro, e dipendono dal tipo di unità periferica da connettere.

Una **interfaccia** è un dispositivo che consente all'elaboratore di **comunicare con una periferica** (tastiere, mouse, dischi, terminali, stampanti, ...).

OLTRE la macchina di Von Neumann

- **Problema:** nella Macchina di Von Neumann le operazioni sono **strettamente sequenziali**.
- Altre soluzioni introducono forme di **parallelismo**
 - **processori dedicati** (*coprocessori*) al calcolo numerico, alla gestione della grafica, all'I/O.
 - **esecuzione in parallelo** delle varie fasi di un'istruzione: mentre se ne esegue una, si acquisiscono e decodificano le istruzioni successive (**pipeline**)
 - **architetture completamente diverse:** sistemi multi-processore, macchine dataflow, LISP machine, reti neurali.