

# Corso di Calcolatori Elettronici I

---

## Introduzione al linguaggio macchina

**Prof. Roberto Canonico**



Università degli Studi di Napoli Federico II  
Dipartimento di Ingegneria Elettrica  
e delle Tecnologie dell'Informazione  
Corso di Laurea in Ingegneria Informatica  
Corso di Laurea in Ingegneria dell'Automazione

---

# Il processore

---

---

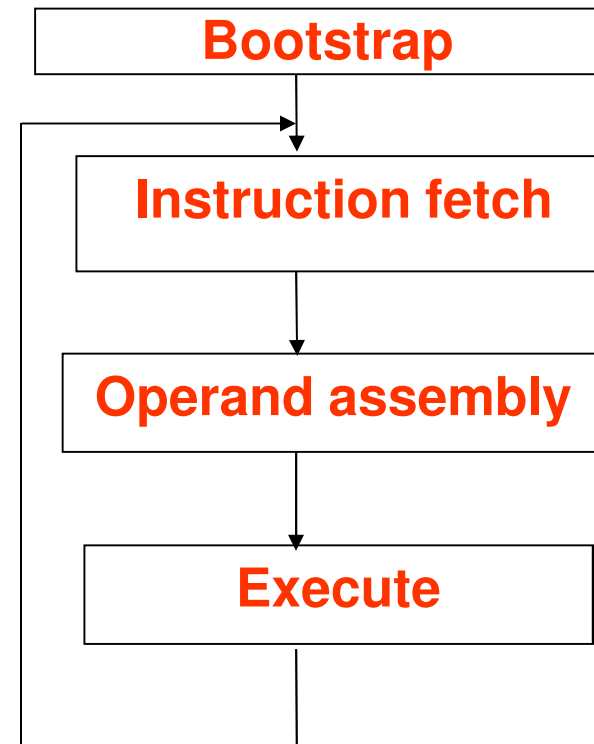
- Nell'architettura di un calcolatore si identificano i seguenti componenti fondamentali:
    - la CPU (*Central Processing Unit*) o processore
    - la memoria centrale
    - i dispositivi di Input/Output
    - il sistema di interconnessione
  - Il *processore* internamente è costituito da:
    - Unità di controllo
    - Registri
    - Unità aritmetico-logica (ALU)
    - Sezione di Collegamento con la memoria
      - Memory Address Register MAR
      - Memory Data Register MDR o Memory Buffer MB
    - Sezione di Collegamento con Ingresso-Uscita
-

# Algoritmo del processore

---

- Il processore opera secondo un algoritmo ciclico che consiste di tre fasi:
  1. Prelievo dell'istruzione
  2. Decodifica e preparazione degli operandi
  3. Esecuzione

**Nella fase di bootstrap il ciclo viene inizializzato;**  
**viene assegnato un valore iniziale opportuno a PC in modo da avviare l'esecuzione di un programma iniziale in ROM**



# Algoritmo del Processore

---

---

- **Prelievo dell'istruzione (Fetch)**
    - La CPU preleva dalla memoria l'istruzione il cui indirizzo è in PC
    - L'istruzione viene copiata nel registro IR
  - **Decodifica / prelievo degli operandi (Operand Assembly)**
    - L'unità di controllo esamina il contenuto di IR e ricava il tipo di operazione ed i relativi operandi
    - Eventuali operandi contenuti in memoria vengono prelevati
  - **Esecuzione dell'istruzione (Execute)**
    - L'unità di controllo richiede all'ALU di effettuare l'operazione specificata nell'istruzione ed invia il risultato ad un registro o alla memoria
-

# Istruzione I/m

---

- Il linguaggio macchina di un processore è costituito dalla codifica in binario delle istruzioni eseguibili dal processore
- Un'istruzione in linguaggio macchina è, sul piano astratto, una tripla strutturata:
  - $i = (f, P1, P2)$

ove:

- $f \in F$  insieme dei *codici operativi* del processore, cioè delle operazioni elementari definite al livello del linguaggio macchina;
  - $P1$  è un insieme di *operandi-sorgente*, cioè di valori e/o puntatori a registri (in senso proprio o registri di memoria) contenenti i valori su cui opera  $f$ ;
  - $P2$  è un insieme di *operandi-destinazione*, cioè di puntatori ai registri (in senso proprio o registri di memoria) cui sono destinati i risultati dell'elaborazione  $f$
-

# Rappresentazione di un'istruzione I/m

---

- Sul piano della sua rappresentazione, una istruzione è espressa come una informazione strutturata:

```
tipo istruzione=  
  cartesiano  
  codop: codice_operativo  
  parte-operandi: operandi  
end
```

- Tale informazione è codificata in macchina mediante codici a *lunghezza fissa* (tipicamente 32 bit, es. RISC) o a *lunghezza variabile* (nel Motorola 68000 multipli di 16 bit)
-

# Diversificazione delle istruzioni I/m sulla base degli operandi

---

Le istruzioni I/m, rispetto agli operandi su cui operano, si diversificano:

1. Per tipo degli operandi (es. intero a 8, 16 o 32 bit);
2. Per numero degli operandi espliciti (0, 1, 2 o 3);  
*e, per ciascun operando:*
3. Per la "natura" (ad esempio, se è una costante, se è il contenuto di un registro o di una locazione di memoria);
4. Per la tecnica di indirizzamento (fra l'altro se è implicito o esplicito);

**Nel seguito si propongono vari possibili criteri di classificazione delle istruzioni I/m**

---

# Classificazione delle istruzioni l/m per numero di operandi espliciti

---

- Tipiche istruzioni l/m hanno 0, 1, 2 o 3 operandi espliciti:
    - OP                      **es. ClearAccumulator**
    - OP **O1**                      **es. Clear R0**
    - OP **O1, O2**                      **es. Move R1, R2**
    - OP **O1, O2, O3**                      **es. Add R4, R6, R1**

ove O1, O2, O3 sono operandi espliciti
  - Laddove l'istruzione abbia operandi impliciti, si tratta tipicamente della costante zero oppure di un registro (ad esempio l'accumulatore, nelle macchine ad accumulatore)
-



# Classificazione delle istruzioni l/m per la natura degli operandi

---

- In funzione della natura degli operandi, le istruzioni sono classificate come:
    - memoria-immediato
    - memoria-registro
    - memoria-memoria
    - registro-immediato
    - registro-registro
  - In ciascuna coppia, il primo termine indica la natura dell'operando destinazione, mentre il secondo termine indica la natura dell'operando (o degli operandi) sorgente
  - Una CPU non supporta necessariamente tutte le possibili combinazioni sopra elencate; eccezioni sono possibili, anche per singole istruzioni
-

# Classificazione delle istruzioni I/m per codici operativi

---

- Ciascuna CPU è caratterizzata da un proprio repertorio di istruzioni I/m
  - Il repertorio di codici operativi di una CPU può essere più o meno ricco
    - CISC vs. RISC
  - Il repertorio può comunque essere suddiviso tipicamente in poche “classi” di istruzioni fondamentali
-

# Classi fondamentali di istruzioni l/m

## (1)

---

---

- Istruzioni di trasferimento dati
    - Copiano un dato dall'operando sorgente all'operando destinazione
  - Istruzioni aritmetiche
    - Effettuano operazioni aritmetiche sugli operandi sorgente e memorizzano il risultato nell'operando destinazione
    - Operano tipicamente su dati numerici di tipo intero
  - Istruzioni logiche e di scorrimento
    - Effettuano operazioni logiche booleane e di shift sugli operandi sorgente e memorizzano il risultato nell'operando destinazione
    - Operano tipicamente su dati di tipo “stringa di bit”
-

# Classi fondamentali di istruzioni I/m

## (2)

---

- Istruzioni di comparazione
    - Alterano i flag del registro di stato del processore (*Processor Status Word* o *Status Register*) in base all'esito del confronto tra due operandi sorgente espliciti (istruzioni di *Compare* propriamente dette) o tra un operando sorgente esplicito ed uno implicito (tipicamente zero, come per l'istruzione *Test*)
  - Istruzioni di salto
    - Alterano il flusso sequenziale che caratterizza la normale esecuzione delle istruzioni, consentendo la realizzazione di diramazioni (*if-then-else*) e *cicli*
    - Agiscono modificando il registro *Program Counter*
    - Possono essere *condizionate* (alla verità di un predicato logico funzione dei flag del registro di stato) o *non-condizionate*
-

# Classi fondamentali di istruzioni I/m

## (3)

---

---

- Istruzioni di collegamento a sottoprogramma
    - Sono istruzioni di salto che implementano i meccanismi necessari a consentire il ritorno al programma chiamante (salvataggio e ripristino dell'indirizzo dell'istruzione successiva al salto a subroutine)
  - Istruzioni di input/output
    - Alcune CPU sono dotate di istruzioni apposite per il trasferimento di dati da/verso le interfacce delle periferiche di input/output
-

# Istruzioni di trasferimento dati

---

- Copiano un dato dall'operando sorgente all'operando destinazione
  - Tipicamente a due operandi espliciti
    - **MOVE sorgente,destinazione**
  - Nelle CPU ad accumulatore, uno dei due operandi è implicito: l'accumulatore
    - **LoadAccumulator #5**  $ACC \leftarrow 5$
    - **StoreAccumulator 1000**  $M[1000] \leftarrow ACC$
  - Le istruzioni di tipo Clear assumono la costante zero come operando sorgente implicito
    - **Clear R1**  $R1 \leftarrow 0$
  - Le istruzioni che operano sul tipo "indirizzo di memoria" sono tipicamente considerate a parte
-

# Istruzioni aritmetiche

---

- Effettuano operazioni aritmetiche unarie (cambia segno) o binarie (addizione, sottrazione, moltiplicazione, divisione) su dati interi espressi su 8, 16, 32 bit

$a = (\text{op}) b$     operazione unaria

$a = b (\text{op}) c$     operazione binaria

- Alcune CPU sono dotate di istruzioni I/m per l'aritmetica in virgola mobile
  - In altri casi, un apposito coprocessore fornisce l'estensione del set di istruzioni per il supporto alla virgola mobile
  - Operazioni aritmetiche più complesse (es. radice quadrata) o funzioni trigonometriche ed esponenziali sono di solito supportate da coprocessori o realizzate in software
-

# Istruzioni aritmetiche (2)

---

- Alcune CPU impongono il vincolo che l'operando destinazione coincida con un operando sorgente
    - $a = (\text{op}) a$  operazione unaria
    - $a = a (\text{op}) b$  operazione binaria
  - Ciò consente di lavorare con istruzioni a due soli operandi espliciti
    - Es. nel Motorola 68000:
      - $\text{ADD } D0, D1$   $D1 \leftarrow [D0] + [D1]$
  - Il formato di istruzioni a 3 operandi espliciti è tipico delle CPU RISC; in esse, però, c'è il vincolo che i tre operandi siano tutti di tipo registro
  - Altre limitazioni sulla natura e sui modi di indirizzamento degli operandi valgono anche per le CPU CISC
    - Ad esempio, nel 68000 le istruzioni aritmetiche devono avere necessariamente un operando di tipo registro
-



# Istruzioni logiche

---

- Effettuano operazioni logiche booleane “bit a bit” sia unarie (NOT) che binarie (AND, OR, XOR) su dati di tipo “stringa di bit” espressi su 8, 16, 32 bit

$a = (\text{op}) b$       operazione unaria

$a = b (\text{op}) c$     operazione binaria

- Alcune CPU impongono il vincolo che l’operando destinazione coincida con un operando sorgente

$a = (\text{op}) a$                       operazione unaria

$a = a (\text{op}) b$                     operazione binaria

- Ciò consente di lavorare con istruzioni a due soli operandi espliciti

– Es. nel Motorola 68000:

- AND D0,D1

$D1 \leftarrow [D0] \text{ AND } [D1]$

---

## Istruzioni logiche (2)

---

---

- L'operazione di AND può essere utilizzata per mettere selettivamente a zero alcuni bit in un registro o in una locazione di memoria
    - AND.B #%11111100,D1  
mette a zero i due bit meno significativi di D1
  - L'operazione di OR può essere utilizzata per mettere selettivamente a uno alcuni bit in un registro o in una locazione di memoria
    - OR.B #%00000011,D1  
mette ad uno i due bit meno significativi di D1
-

# Istruzioni logiche (3)

---

---

- L'operazione di XOR può essere utilizzata per negare selettivamente alcuni bit in un registro o in una locazione di memoria
  - XOR.B #%00000011,D1  
inverte i due bit meno significativi di D1

# Istruzioni di scorrimento

---

- Similmente alle operazioni logiche operano su dati di tipo “stringa di bit”
  - Operazioni tipiche:
    - *Shift-Left* sia aritmetico che logico
    - *Shift-Right* sia aritmetico che logico
    - *Circular-Shift-Left*
    - *Circular-Shift-Right*
  - Il numero di scorrimenti può essere fisso (tipicamente uno) o variabile (espresso da un ulteriore operando, immediato o registro)
-

# Istruzioni di comparazione

---

- Alterano i flag del registro di stato del processore (*Processor Status Word* o *Status Register*) in base all'esito del confronto tra due operandi sorgente espliciti (istruzioni di *Compare* propriamente dette) o tra un operando sorgente esplicito ed uno implicito (tipicamente zero, come per l'istruzione *Test*)
  - Tipicamente queste istruzioni precedono le istruzioni di salto condizionato, e congiuntamente ad esse consentono di realizzare figure di programmazione quali le strutture di controllo *if-then-else* ed i *cicli*, tipici dei linguaggi di programmazione di alto livello
-

# Istruzioni di salto

---

---

- Alterano il flusso sequenziale che caratterizza la normale esecuzione delle istruzioni
  - Agiscono modificando il registro *Program Counter*
  - Possono essere *condizionate* (alla verità di un predicato logico funzione dei flag del registro di stato) o *non-condizionate*
  - In I/m si distingue anche tra salti **assoluti** (*Jump*) e **relativi** (*Branch*)
  - Le istruzioni di Jump contengono nel codice I/m l'indirizzo destinazione
  - Le istruzioni di Branch contengono nel codice I/m un offset che, sommato al PC attuale, determina l'indirizzo destinazione
-

# Istruzioni di collegamento a sottoprogramma

---

- Le istruzioni di salto a sottoprogramma (*Jump To Subroutine* o *Call*) salvano il valore del PC per consentire il ritorno al programma chiamante
  - Le istruzioni di ritorno da sottoprogramma (*Return From Subroutine*) ripristinano il valore del PC salvato per realizzare il ritorno al programma chiamante
  - Il valore del PC può essere salvato in un apposito registro (*Link Register*, CPU RISC) o sulla cima dello stack di sistema (soluzione tipica delle CPU CISC)
-

# Istruzioni di input/output

---

- Alcune CPU sono dotate di istruzioni apposite per il trasferimento di dati da/verso le interfacce delle periferiche di input/output
    - Istruzioni IN e OUT
  - Si tratta in sostanza di istruzioni di trasferimento dati che operano su uno spazio di indirizzamento (quello delle interfacce di I/O) distinto da quello della memoria
  - Nei sistemi nei quali spazio di indirizzamento di I/O e spazio di indirizzamento di memoria coincidono (sistemi con I/O memory mapped) le operazioni di I/O vengono eseguite tramite normali istruzioni di trasferimento dati
-



# Modello di programmazione del processore MC68000

