# 8 Insieme di istruzioni macchina

**Parola di memoria (memory word)**
●Il calcolatore non lavora su singoli bit ma su gruppi di bit detti PAROLE di lunghezza da 8 a 64 bit (sempre potenze di 2)

●La dimensione delle parole dipende dall'architettura del calcolatore

●I dati posso occupare da un singolo byte a diverse parole

●Le istruzioni possono occupare una o più parole

cosa è una memory word? --**esame**
è la dimensione del' unita di dato che un sistema di elaborazione può gestire e memorizzare in un singolo ciclo di clock di una CPU, varia in base a se l'architettura è a 8,16,32,64 bit,
queste memory word vengono lette e scritte nella memoria RAM
L'informazione è immagazzinata in memoria sotto forma di un vettore di parole (parole in successione)
ogni word ha quindi un indirizzo in memoria

**l'indirizzamento è quell'operazione che attribuisce ad un valore un indirizzo di memoria, in modo che possa essere trovato dalle operazioni che lo richiedono** (la CPU legge o scrive dati nell'indirizzo corrispondente)
Di norma l'unità minima di informazione indirizzabile in memoria è il byte

Esistono 2 metodi di indirizzamento di byte:

1. **BIG-ENDIAN**: indirizzo aumenta al diminuire del peso aritmetico del byte

2. **LITTLE-ENDIAN**: indirizzo aumenta all'aumentare del peso aritmetico del byte

In sintesi l'indirizzamento permette di localizzare ogni byte in memoria mentre l'ordinamento di questi byte denota come sono strutturate le words

**ISA**
è l'interfaccia tra il software e l'hardware del PC definisce il set di istruzioni che una CPU può eseguire.
**Quanti tipi ne esistono?**
questo insieme di istruzioni include operazioni logiche, aritmetiche, controllo, ecc... **ogni CPU ha il proprio set ISA (CISC o RISC)**

Ovviamente la CPU è in grado capire solo 0 e 1 infatti i programmi si eseguono cosi (di solito (es. C)):
LINGUAGGIO ALTO LIVELLO → compilatore → LINGUAGGIO MACCHINA
ASSEMBLATIVO → assemblatore → LINGUAGGIO MACCHINA BINARIO

Il set di comandi di assembly è leggermente diverso proprio per le differenze che ci sono tra CISC e RISC, ma esistono anche assembly generici

**Le direttive dell'assemblatore** sono istruzioni speciali che non vengono tradotte in linguaggio macchina e non producono codice eseguibile. Servono a fornire indicazioni all'assemblatore su come gestire il processo di traduzione e organizzazione del programma. Le direttive aiutano nella definizione di dati, etichette, segmenti di memoria, e altri aspetti strutturali, ma non influenzano direttamente l'esecuzione del programma.

La sintassi di assembly segue questo schema:
**SINTASSI ASSEMBLY**
ETICHETTA OPERAZIONE OPERANDI COMMENTO
(facoltativa) (facoltativa)

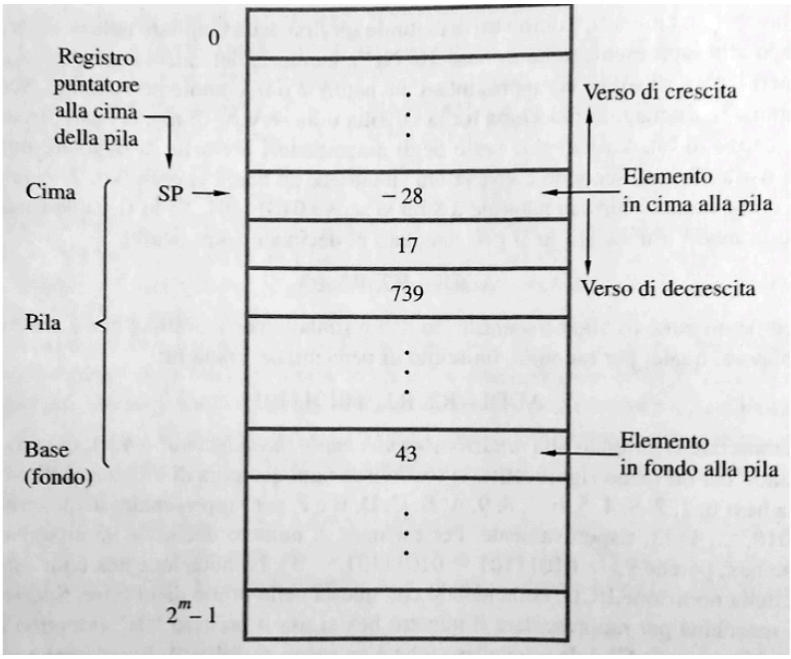
SOMMA ADD AX, BX ; Somma il contenuto di BX a AX

**Pila (Stack)**
●Lista di elementi (parole) dove si immaginano i dati posizionati uno sull'altro

●Gli elementi possono essere solo aggiunti e prelevati dalla cima della pila: l'ultimo elemento inserito è il primo ad essere prelevato **(Last In First Out – LIFO)**

●Stack Pointer – SP: registro che punta alla cima della pila

**●Gli elementi della pila hanno indirizzi in ordine decrescente dalla base alla cima**



si possono fare delle operazioni sulle pile, come ad esempio
**PUSH e POP ← esame**

PUSH
serve ad **aggiungere un elemento in cima** alla pila, in sostanza **diminuisco di 1 l'indirizzo puntato dallo stack-pointer** (SP), e **poi scrivo il valore nella nuova cima** della pila

Subtract SP, SP, #4 //sottraggo 4 ad SP cosi mi sposto di un indirizzo e creo la nuova cima
Store Rj, (SP) //salvo il valore

POP
serve per **prelevare un elemento dalla cima** della pila, in sostanza **copio il valore puntato da SP** in un registro della CPU e **poi aumento il valore di SP** cosi da spostarmi alla nuova cima

Load Rj, (SP) //prendo il valore dalla cima e lo salvo in Rj
Add SP, SP, #4 //aumento l'indirizzo cosi da spostarmi alla nuova cima

**Per vedere degli esempi di istruzioni e altre definizioni di queste guarda il PowerPoint 8 dalla slide 17 alla 33** → 8 Insieme-di-istruzioni-macchina.pdf

**Funzioni**
L'area di attivazione in assembly è quella porzione di memoria usata per gestire le chiamate a funzione o le subroutine, ogni volta che viene chiamata una funzione viene creato un nuovo stack frame in cima allo stack esistente. Lo stack frame contiene:

- i parametri della funzione che devi chiamare
- Le variabili locali
- e il puntatore all'area di attivazione
  **Una funzione si chiama cosi:**
  Call INDIRIZZO
  **Per rientrare:**
  Return
  **Parametri**
  Posso passare i parametri in 2 modi diversi:

1. usando i registri: metto un parametro in un registro (quindi quando finisco i registri finisco i parametri da poter usare)
2. usando uno Stack (pila): cosi impilo i parametri nella pila (parametri virtualmente illimitati)

Le istruzioni di assembly cambiano in base all'architettura della CPU ad esempio una CPU a 32 bit ha 3 forme di istruzioni: → **esame**

- **Formato con operandi in registro**
  - Sono quelle istruzione che operando sui registri della CPU, un'esempio è l'istruzione ADD R1,R2
- **Formato con operando immediato**
  - Sono quelle istruzioni in cui un operando è fornito in modo immediato come la mov R1,#5 dove l'operando #5 viene fornito in modo immediato
- **Formato per chiamata**
  - Sono quelle istruzioni in cui il valore immediato è fornito sin da subito (non ci sono esempi immediati)

**Register Transfer notation**: la notazione di trasferimento del registro in assembly è il metodo utilizzato per descrivere il movimento dei dati nei registri della CPU,indica in modo specifico come i valori vengono manipolati i dati nel passaggio da un registro ad un'altro.

- $R_1 \leftarrow [R_2]$ la freccia indica un trasferimento di valore
  Inoltre viene usato per specificare il funzionamento delle delle operazioni aritmetico-logiche
- $C \leftarrow [A] + [B]$
  Le parentesi quadre ([ ]) servono a indicare il valore contenuto in un registro

Spesso può capitare di dover salvare/caricare 1 byte (8 bit) e esistono 2 istruzioni che lo permettono:

LoadByte Rdst, LOCBYTE → legge 8 bit e li salva nel registro
registro, indirizzo di memoria

StoreByte Rsrc, LOCBYTE → salva 8 bit di Rsrc nella locazione di memoria indicata da LOCBYTE

Esistono anche delle istruzioni per moltiplicare e dividere degli operandi (anche se queste istruzioni non sono disponibili in tutti i processori)

Multiply Rk, Ri, Rj
moltiplica Ri e Rj e salva in Rk

Divide Rk, Ri, Rj
divide Ri e Rj e mette il quoziente in Rk (il resto non viene calcolato)

**Differenze tra CISC e RISC → esame**

# Differenze tra stili RISC e CISC

## RISC:

- Modi di indirizzamento semplici
- Meno istruzioni, tutte occupanti una singola parola
- Operazioni aritmetiche e logiche solo su registri
- Non sono possibili trasferimenti diretti tra due locazioni di memoria
- Possibile l'elaborazione a stadi
- Programmi di dimensioni maggiori

## CISC:

- Modi di indirizzamento complessi
- Tante istruzioni complesse, occupanti più parole di memoria
- Operazioni aritmetiche e logiche con operandi sia su registri che locazioni di memoria
- Trasferimenti diretti tra due locazioni di memoria tramite istruzione Move
- Programmi di dimensioni ridotte

**Come detto in precedenza su CISC esiste l'istruzione**
Move destinazione, sorgente

●L'operando destinazione può essere il nome di un registro o un indirizzo di memoria

●L'operando sorgente può essere il nome di un registro, un indirizzo di memoria o un valore immediato

●Nel caso che il sorgente sia un registro o un indirizzo di memoria l'istruzione esegue la seguente funzione espressa in **RTN**: destinazione ← [sorgente]

●Nel caso che il sorgente sia un valore immediato l'istruzione esegue la seguente funzione espressa in **RTN**: destinazione ← sorgente

Posso anche **incrementare di 1**, ad esempio:
(Registro)+

es. Move ELEMENTO, (SP)+

prende il valore di SP e lo mette in ELEMENTO, dopo incrementa SP di 1
(può essere usato nell'operazione di POP)

Analogamente posso **decrementare di 1**, ad esempio:
-(registro)

es. Move -(SP), NUOVOELEMENTO

in questo caso prima decrementa SP e poi fa l'operazione
(può essere usato nell'operazione di PUSH)

- **I bit di esito o condizione** sono bit speciali immagazzinati in un registro interno al processore chiamato **registro di stato**.

- I bit di esito tengono traccia dell'esito di svariate operazioni, utili per valutare le condizioni di salto. Vengono aggiornati quando avviene un operazione aritmetica e logica o trasferimento di dato.

- I bit di esito più comuni sono:

| Bit di esito | Significato |
|---|---|
| N (negativo) | 1 se risultato negativo, 0 se positivo o nullo |
| Z (zero) | 1 se risultato nullo, 0 altrimenti |
| V (trabocco) | 1 se trabocco in comp. a due, 0 altrimenti (oVerflow) |
| C (riporto) | 1 se trabocco in binario naturale, 0 altrimenti (Carry) |