



UNIVERSITA' DEGLI STUDI DI PARMA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

# Fondamenti di Informatica B

---

Lezione n. 11

Alberto Broggi – Gianni Conte

A.A. 2005-2006

# Fondamenti di Informatica B

---

## Lezione n.11

- ARCHITETTURA INTERNA
- ARCHITETTURA ESTERNA
- CODICE MACCHINA
- MODI DI INDIRIZZAMENTO
- ARCHITETTURE A PIU' INDIRIZZI

**In questa lezione verranno introdotti i concetti di base relativi alla architettura interna di una CPU e al linguaggio macchina (o linguaggio assembly).**

# Architettura interna ed esterna

---

**ARCHITETTURA INTERNA:** Struttura interna della CPU. Migliore compromesso possibile fra le prestazioni e i costi avendo come vincolo la tecnologia.

**ARCHITETTURA ESTERNA:** Come il processore è visto da chi lo deve programmare. Insieme delle istruzioni, dei registri, dei modi di indirizzamento e dei tipi di dato ammessi dalle istruzioni.

Attraverso livelli di interpretazione si realizzano le funzioni definite all'esterno:

- Linguaggio macchina (o assembly).
- Linguaggio di microprogramma.
- Comandi alla parte operativa.

# Linguaggio Macchina

---

Il linguaggio macchina o assembly di una CPU definisce:

- Le operazioni possibili
- Le risorse possibili e la loro utilizzazione

La maggior parte delle istruzioni è del tipo:

$$X_1 \leftarrow f(X_1, X_2, \dots, X_n)$$

con n eguale a 1, 2 o (raramente) 3.

Funzioni complesse sono realizzate sfruttando i livelli di interpretazione senza modificare l'architettura interna. L'architettura interna è direttamente influenzata dalle caratteristiche sintattiche dell'architettura esterna (tipi di dato, ...).

# Linguaggio Macchina

---

Ogni istruzione è definita da:

- Codice macchina:  $\Rightarrow$  10010101
- Codice mnemonico:

$\text{MOV A,B} \Rightarrow \text{"A} \leftarrow \text{B"}$

Corrispondenza 1 a 1 tra i due codici.

Ogni istruzione in linguaggio macchina deve definire:

- Operazione da svolgere
- Operandi coinvolti
- Posizione dell'istruzione successiva

# Linguaggio Macchina

---

## ISTRUZIONE SUCCESSIVA

- Le istruzioni sono eseguite in sequenza
- L'indicazione relativa all'istruzione successiva è spesso implicita
- Il PC (Program Counter) memorizza l'indirizzo della istruzione da eseguire
- Per alterare la sequenza sono introdotte istruzioni di salto:

$(PC \leftarrow X)$

# Linguaggio Macchina

---

- **CODICE MACCHINA**

In generale il codice macchina è suddiviso in campi:



Codice operativo

1<sup>^</sup> operando

2<sup>^</sup> operando

- **CODICE MNEMONICO**

**ADD A,B**

Operazione      operandi

**Il codice mnemonico rispecchia esattamente la struttura del codice macchina**

# Linguaggio Macchina

---

Ogni operando è associato a un dato di cui occorre conoscere la localizzazione attraverso il suo **INDIRIZZO**

Modi diversi per indicare la posizione:

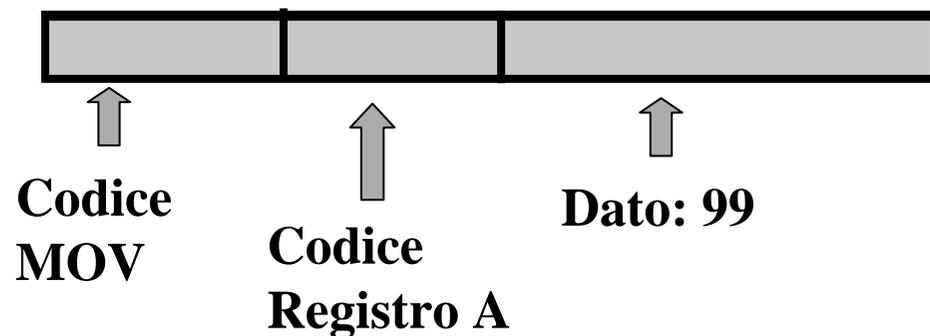
- IMMEDIATO
- DIRETTO
- INDIRECTO

## MODO IMMEDIATO

Il dato è contenuto nel codice macchina.

```
MOVI A, 99 ○
```

```
MOV A, #99
```



# Modi di Indirizzamento

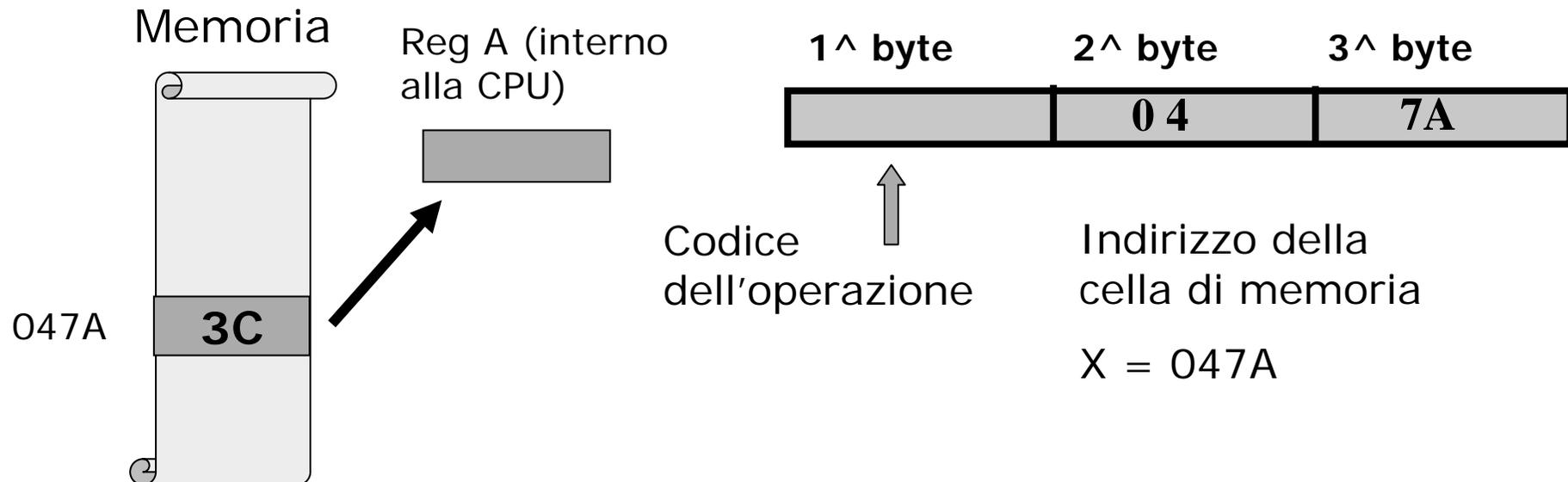
## MODO DIRETTO

Il codice macchina contiene l'indirizzo del dato.

**MOV A,X**

**X può essere:**

- **Indirizzo della cella di memoria contenente il dato.**
- **Codice registro interno.**



# Modi di Indirizzamento

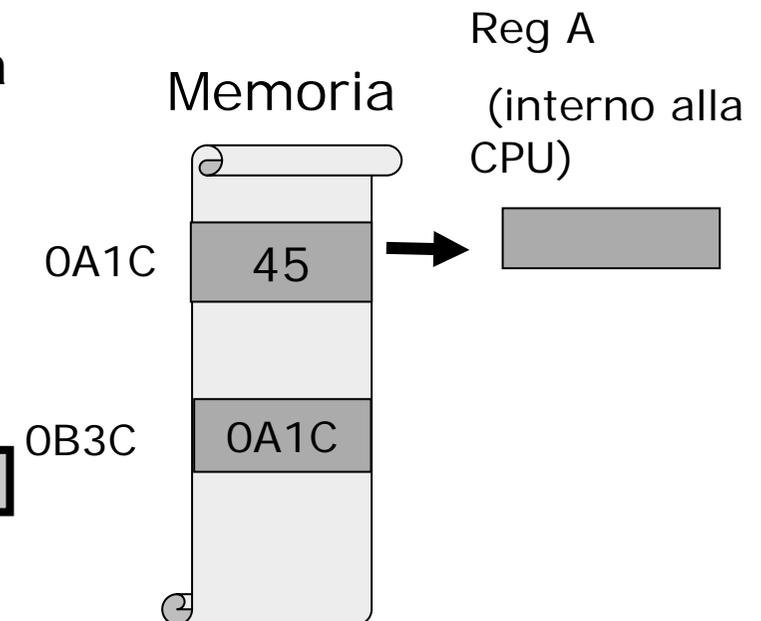
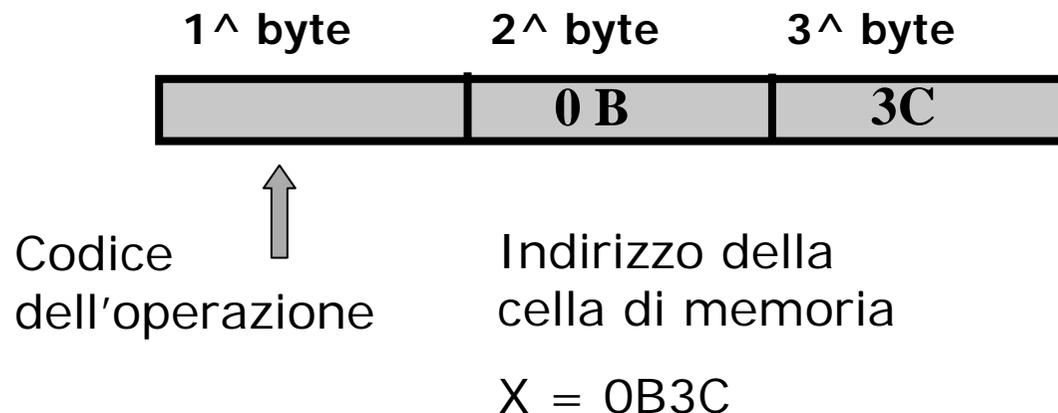
## MODO INDIRECTO

Il codice macchina contiene l'indirizzo della locazione di memoria che contiene l'indirizzo del dato:

`MOV A, (X)`

**X** può essere:

- Indirizzo della cella di memoria contenente l'indirizzo del dato
- Codice registro interno contenente l'indirizzo del dato



# Tipi di Indirizzo

---

**Il valore dell'indirizzo può essere espresso in modo:**

- **ASSOLUTO:**

L'indirizzo completo compare nel campo operando.  
Svantaggio: la lunghezza del campo indirizzo genera codice di dimensione elevata.

- **RELATIVO:**

Nel campo operando compare solo lo spiazzamento (scostamento) relativo (differenza rispetto al valore contenuto) al PC.

Lo spiazzamento può essere:

- contenuto in un byte
- un valore negativo

# Tipi di Indirizzo

---

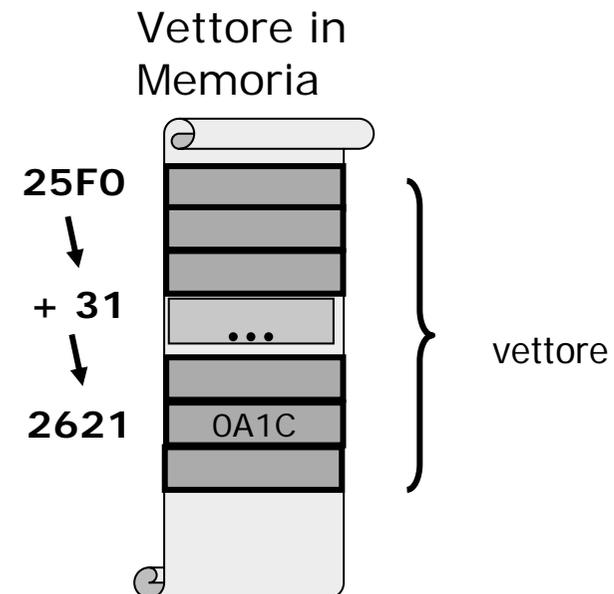
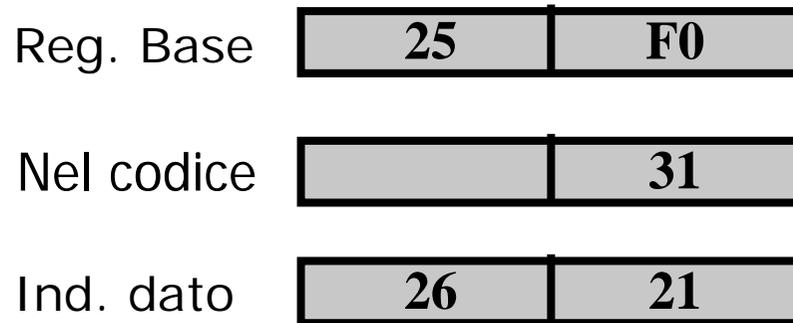
## INDIRIZZO COMPOSTO

Estendendo il concetto di indirizzo relativo, al posto del PC, si può utilizzare:

- Registro Base
- Registro Indice

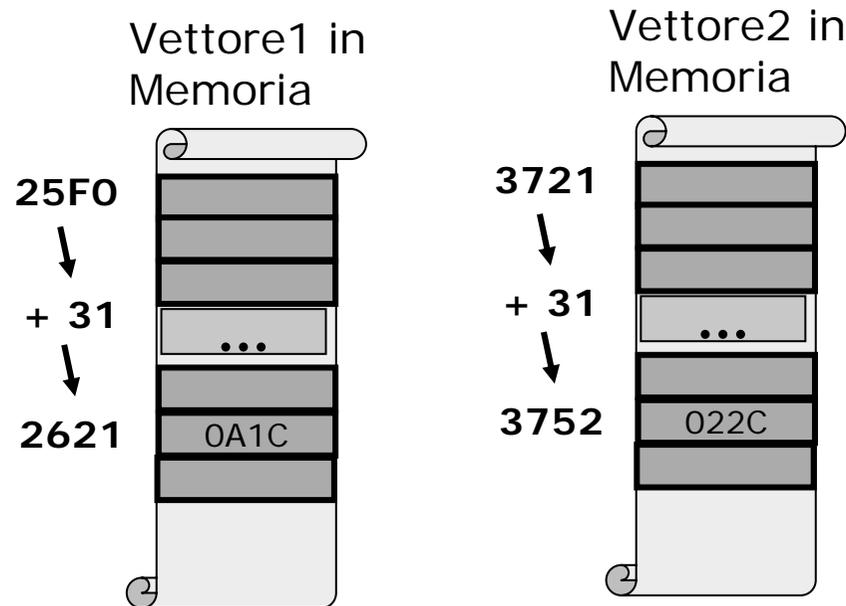
# Registro Base

- Nel codice operativo compare solo lo spiazzamento rispetto a un valore contenuto nel registro base
- L'indirizzo effettivo si ottiene sommando lo spiazzamento al contenuto del registro base
- Permette al processore di accedere ad una nuova zona di memoria solo cambiando il contenuto del registro base



# Registro Base - esempio

- Cambiando solamente il contenuto del registro base si può accedere a un dato con lo stesso indice in vettori diversi



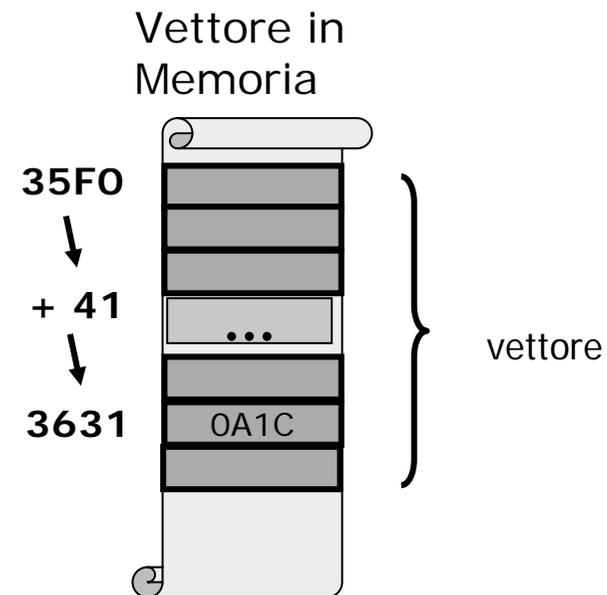
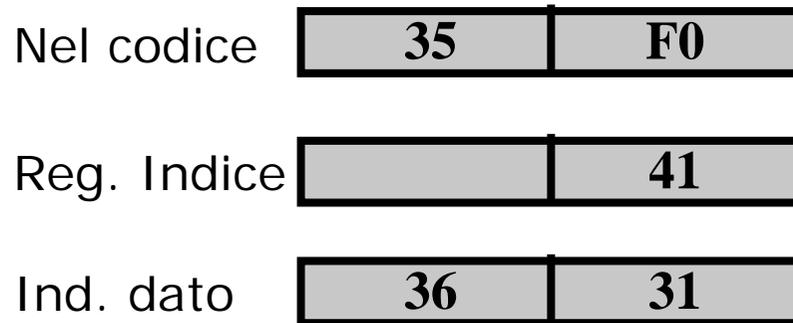
## ESEMPIO

In un vettore che memorizza informazioni anagrafiche la data di nascita è memorizzata sempre nella stessa posizione

Per accedere alla data di nascita di un'altra persona occorre solamente cambiare il dato nel registro di base

# Registro Indice

- Nel codice compare l'indirizzo iniziale di un vettore; la posizione all'interno del vettore è individuata mediante un registro
- Il vettore  $X_0, X_1, \dots, X_N$  è memorizzato in locazioni consecutive
- Il codice operativo contiene l'indirizzo di  $X_0$
- Il registro R contiene l'indice i



# Altri modi di Indirizzamento

---

## MODIFICA PUNTATORI

Un altro modo di indirizzamento:

```
MOV A, (X)+
```

X è un registro che al termine dell'esecuzione dell'istruzione viene incrementato

IN GENERALE :

- (X)+ : post-increment
- (X)- : post-decrement
- +(X) : pre-increment
- -(X) : pre-decrement

# Lo Stack

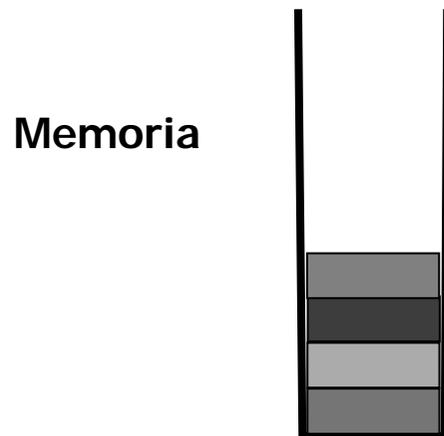
---

Lo STACK è una struttura di memoria nella quale i dati sono gestiti mediante una politica LIFO (Last In – First out)

Le operazioni possibili sono:

- PUSH – Inserimento di un dato nello Stack
- POP – Prelievo di un dato dalla Stack

## Operazione di POP



# Stack Pointer

---

Lo stack pointer è un registro che gestisce le operazioni di PUSH e di POP

PUSH A (inserisce il registro A nello Stack)

equivale a `MOV -(SP),A`

Il registro SP viene decrementato e quindi “punta” alla cella di memoria nel quale deve essere inserito il nuovo dato

POP A (estrae il dato dallo Stack e lo sposta nel registro A)

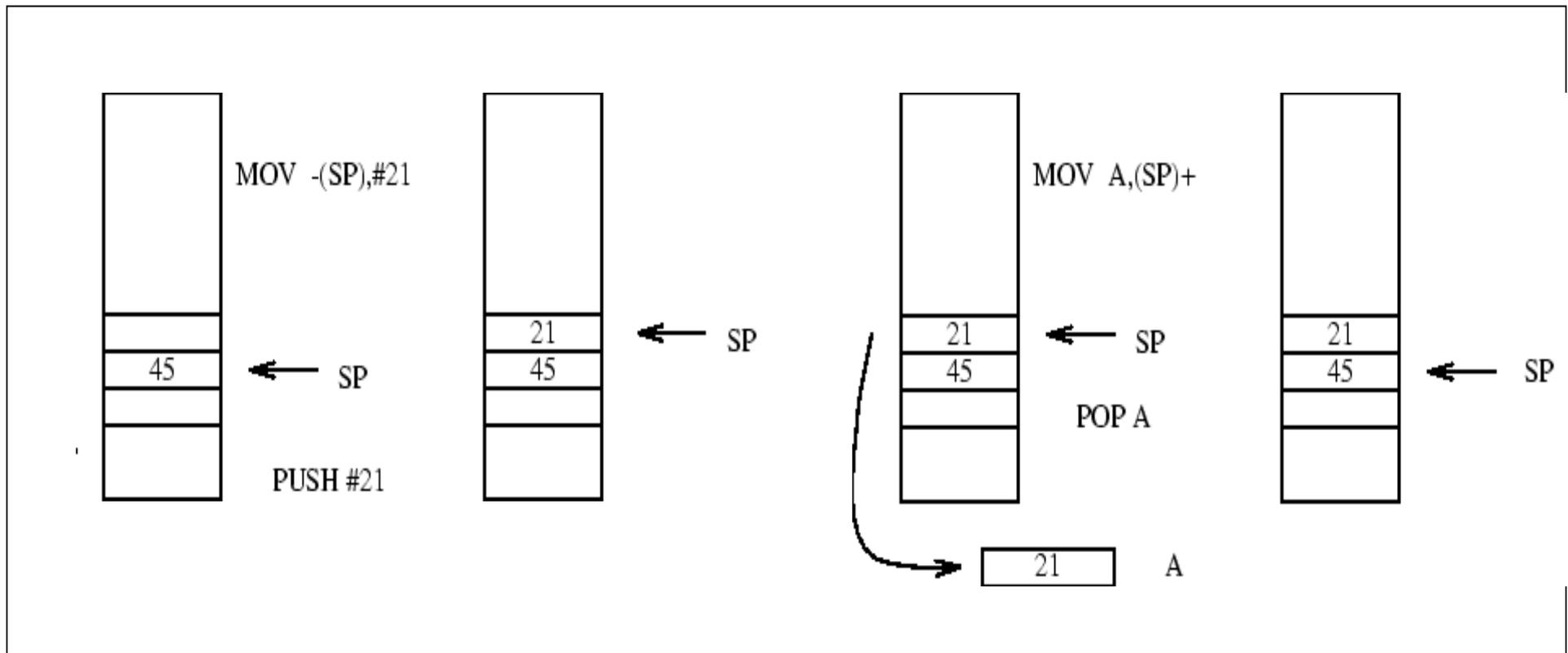
equivale a `MOV A, (SP)+`

Il registro SP “punta” alla cella di memoria dal quale deve essere estratto il dato da riportare in A; successivamente il registro SP viene incrementato in modo che indirizzi il dato successivo nello stack

# Stack Pointer

**PUSH**

**POP**





# Architettura Interna

---

La gestione efficace di operazioni particolari svolte frequentemente richiede la presenza di funzioni specializzate:

- Registri dedicati:
  - Registro indice
  - Registri base
  - Registro di stato
  - Stack Pointer
- Gestione sottoprogrammi e interruzioni

# Architettura Estesa

---

