# Architettura degli elaboratori

## Modi di indirizzamento

*Dott. Franco Liberati*

# ARGOMENTI DELLA LEZIONE

- Definizione modi indirizzamento
- Indirizzamento implicito
- Indirizzamento esplicito
  - Indirizzamenti diretti: immediato, assoluto, a registro
  - Indirizzamenti indiretti: indiretto con registro, con spiazzamento, relativo, con predecremento e postincremento

# Architettura degli elaboratori

## *Modi di indirizzamento*

# MODI DI INDIRIZZAMENTO

## Generalità

- Un'istruzione macchina ha una suddivisione in campi in cui una parte è il **codice operativo** (OPCODE), che specifica la classe ed il tipo di operazioni da eseguire, e un'altra parte è il **Campo Indirizzo** (ADDRESSING MODE), che indica l'operando o dove risiede il dato da elaborare (cioè quale è il registro/i o la locazione di memoria finale contenente l'informazione)


**add Rd,Rs,Rt** (somma con gestione overflow)


**addiu Rd,Rs,Imm** (somma con operando senza gestione overflow)


**beq Rs,Rt,target** (salto nel caso di uguaglianza di operandi)


**j target**(salto incondizionato)

# MODI DI INDIRIZZAMENTO

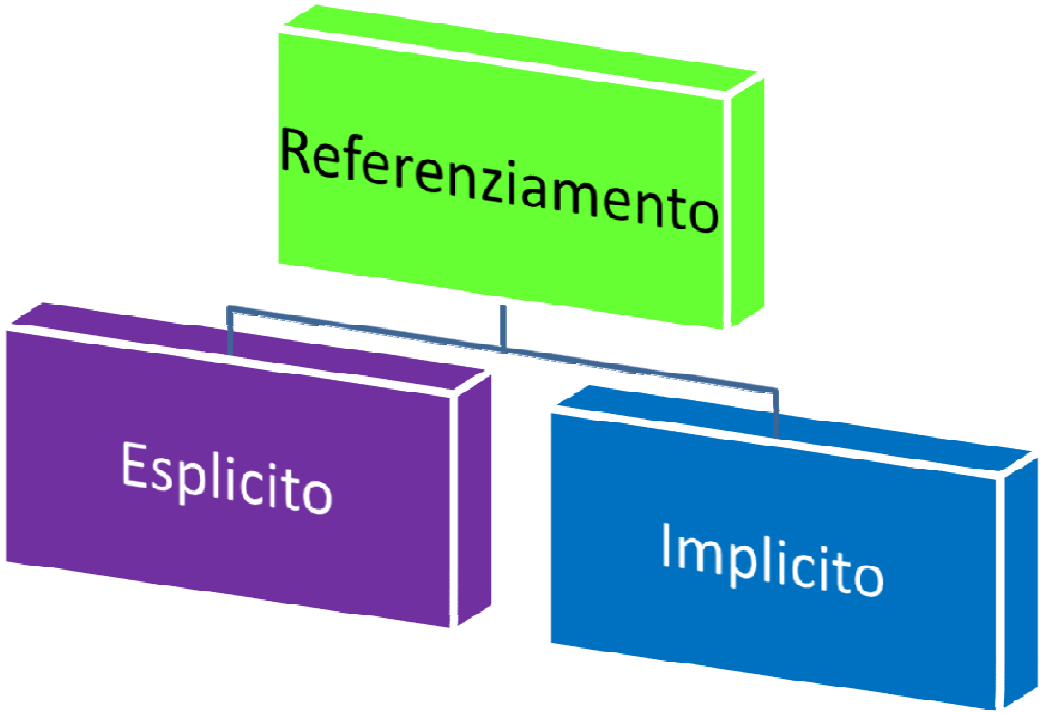
## Generalità: Proprietà

- Il modo di indirizzamento consente:
  - di accedere ad una locazione di memoria il cui indirizzo non è noto nel momento in cui il programma è scritto
  - di manipolare gli indirizzi, cioè permettere delle operazioni su di essi nel momento il cui il programma è eseguito (accesso a strutture dati come vettori, liste)
  - di poter calcolare gli indirizzi relativamente alla posizione dell'istruzione in modo tale che il programma possa essere caricato in memoria in qualsiasi parte della memoria senza prevedere la risoluzione degli indirizzi locali o globali (***program indipendent code***, **PIC**)

# MODI DI INDIRIZZAMENTO

## Generalità: classificazione

- Il luogo dove risiede il dato può essere espresso in maniera **esplicita** oppure può essere omesso, in questo ultimo caso di parla di **modalità implicita**
- Nella modalità esplicita il programmatore deve riportare nell'istruzione il luogo dove risiede il dato da elaborate; nella modalità implicita, durante la fase di esecuzione dell'istruzione, l'Unità di Controllo sa dove andare a prelevare e posizionare i dati per eseguire l'operazione perché implicitamente specificato dall'OPCODE



# MODI DI INDIRIZZAMENTO

## Esplicito

- La maniera più comune di individuare un dato in memoria è quella di indicare il suo **indirizzo effettivo** (EA o *effective address*) nell'ADDRESSING MODE

| Codice operativo (OPCODE) | Modo di indirizzamento (ADDRESSING MODE) |
|---|---|

```
MOVE R0,0x123456      #modo di indirizzamento assoluto
                      #MOTOROLA68000:
                      #si esplicita la locazione in memoria
                      #dove risiede l'operando

lw $t0,pippo          #modo di indirizzamento assoluto nel
                      #MIPS:
                      #si esplicita la locazione in memoria
                      #dove risiede l'operando tramite una
                      #etichetta che è convertita nell'indirizzo
                      #dall'assemblatore o compilatore
```

# MODI DI INDIRIZZAMENTO

## Esplicito: indirizzo effettivo

- Nelle istruzioni di trasferimento dati o in quelle logico-aritmetiche l'indirizzo effettivo è l'indirizzo (memoria) o l'etichetta del registro dove risiedono gli operandi
- In una istruzione di salto o di chiamata a sottoprogramma, l'indirizzo effettivo è quello dell'istruzione a cui si vuole saltare


# MODI DI INDIRIZZAMENTO

## Esplicito: indirizzo effettivo (etichetta)

- Parlando di modi di indirizzamento si fa spesso riferimento alle **etichette**
- Una **etichetta** non è niente altro che un identificatore che nel linguaggio assembly è il **designatore simbolico di un indirizzo in memoria**
- A differenza degli identificatori dei linguaggi ad alto livello (Pascal, C,C++,Java,…) in cui una etichetta ha un valore che le viene assegnato nel momento in cui è definita e questo valore può cambiare; in linguaggio assembly alla etichetta, durante la traduzione, è sostituito il valore che essa rappresenta
- Questa etichetta, esiste solo nel linguaggio assembly e scompare con la traduzione in linguaggio macchina


# MODI DI INDIRIZZAMENTO

## Implicito

- In alcune macchine si utilizza l'**indirizzamento implicito**
- In tale modo non sono esplicitati gli indirizzi dove risiedono gli operandi, ma questi ultimi si trovano in una posizione predeterminata (di solito gli accumulatori) che la macchina conosce a priori
- Grazie al **referenziamento implicito** è possibile utilizzare delle istruzioni con **lunghezza fissa** e **minima**
- L'indirizzamento implicito **esclude alcune istruzioni di trasferimento** (come la LOAD e la STORE) e le istruzioni di salto perché in questo caso è necessario esplicitare da dove prelevare il dato in memoria o l'indirizzo in cui effettuare il salto

**Codice operativo (OPCODE)**

**Esempi indirizzamento implicito:**

**MUL #moltiplicazione**

**ADD #somma**

**Nel MIPS esistono le istruzioni bit set e clear del registro di stato:**

**BSC #set del bit C nello status register**

# MODI DI INDIRIZZAMENTO

## Implicito: limite

In dettaglio è possibile vedere come non sia possibile ricorrere all'indirizzamento implicito (si evidenzia la necessità di utilizzare istruzioni di trasferimento con un modo di indirizzamento esplicito per il reperimento degli operandi dalla memoria)

**Esplicito**

```
LW $t0,pippo
LW $t1,paperino
LW $t2,pluto
MUL $t3,$t0,$t1
ADD $t4,$t3,$t2
SW $t4, risultato
```

**Implicito**

```
LWA1 pippo
#Sposta l'operando nella locazione pippo
#nel primo accomulatore
LWA2 paperino
#Sposta l'operando nella locazione paperino
#nel primo accomulatore
MUL
#effettua la moltiplicazione e pone il risultato nel primo accom.
LWA2 pluto
ADD #effettua la somma e pone il risultato nel primo accom.
SWA1 risultato
```

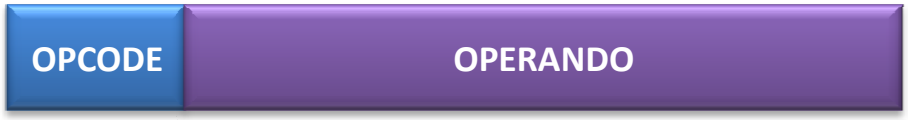
# Architettura degli elaboratori

## *Principali Modi di indirizzamento*

# MODI DI INDIRIZZAMENTO

## Indirizzamento IMMEDIATO

- L'**indirizzamento immediato** è così definito perché l'operando si trova nell'istruzione e pertanto, è nel corpo del programma e non in una area dati
- Nel caso in cui si voglia inserire un numero lungo quanto la parola prestabilita dal progettista della macchina si ricorre ad una **istruzione a lunghezza variabile** dove il resto dell'operando si trova nella parola successiva a quella dell'istruzione
- **Nel MIPS il campo è limitato perché le istruzioni hanno lunghezza fissa. Nel caso di un operando superiore a 16bit, l'istruzione immediate si sdoppia (vedi esempio al lato)**
- Un uso eccessivo può incrementare la lunghezza del programma in memoria e può richiedere più accessi in memoria
- L'indirizzamento immediato è utile per le inizializzazioni dei registri




**Esempio**.

In assembly 68000 un indirizzamento immediato è **move d0,#35**

In MIPS un indirizzamento immediato è :

**li $t0,4000000000**

**# In $t0 c'è: 11101110011010110010100000000000**

è tradotta dall'assemblatore in

**lui $at,0xffffee6b**

**#$at ← 11101110 01101011 00000000 00000000**

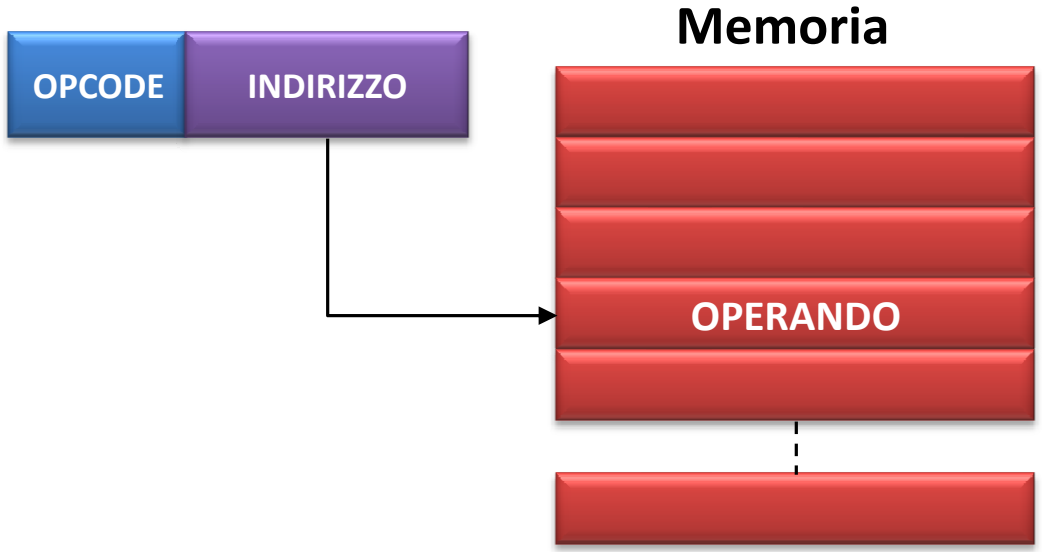
**ori $t0,$at,0x00002800**

**#$t0←$at+00101000 00000000**

# MODI DI INDIRIZZAMENTO

## Indirizzamento ASSOLUTO

- L'**indirizzamento assoluto** specifica l'indirizzo di una locazione di memoria
- L'indirizzo è fissato nel momento in cui si scrive il programma anche se è descritto da una etichetta
- Nel caso di un eccesso di spazio disponibile l'indirizzo è riportato nella parola successiva (o come accade nel MIPS l'istruzione è sdoppiata in due istruzioni)
- *Osservazione: in alcuni testi è riportato come indirizzamento* ***diretto***

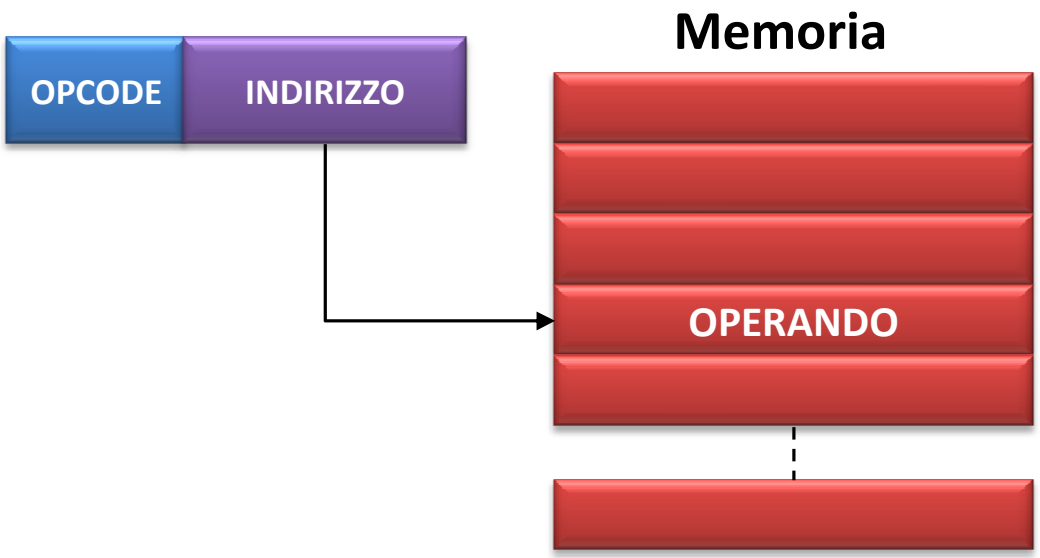



**Esempio**.
Nell'assembly 68000 un indirizzamento assoluto è:
**move d0,112346** #carica il valore contenuto #nell'indirizzo 112346 nel registro d0
Nell'assembly MIPS:
**lw $t0,pippo** #carica il contenuto dell'indirizzo #rappresentato dall'etichetta pippo nel registro $t0

# MODI DI INDIRIZZAMENTO

## Indirizzamento ASSOLUTO

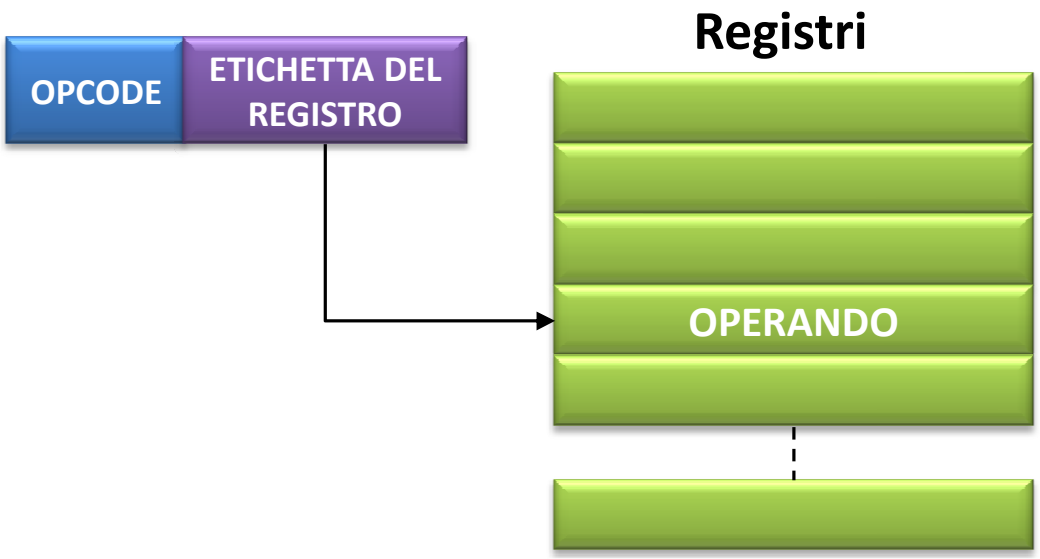
- L'indirizzamento assoluto è utile quando si deve operare con una dato che si trova in una posizione di memoria fissata o quando si deve fare un salto ad una istruzione che si trova ad una qualsiasi locazione in memoria
- L'indirizzamento assoluto potrebbe essere l'unico implementabile perché consente di raggiungere ogni locazione di memoria
- In ogni caso, come nell'immediato, occupa spazio in memoria ed è meno efficiente rispetto altri modi di indirizzamento (può richiedere almeno due accessi in memoria)



# MODI DI INDIRIZZAMENTO

## Indirizzamento A REGISTRO

- L'**indirizzamento a registro** specifica l'etichetta del registro in cui è presente l'operando
- Ogni CU è dotata di un certo numero di registri interni ad uso generale (poche unità a qualche centinaio per le grandi macchine)
- Le istruzioni che utilizzano un indirizzamento a registro sono eseguite più velocemente per due motivi principali:
  - il campo ADDRESSING MODE è breve perché servono pochi bit per selezionare i registri interni (facilita l'uso delle istruzioni a lunghezza fissa)
  - per rintracciare gli operandi non occorrono accessi alla memoria principale perché i registri sono integrati nella CU
- Per questo motivo si usa questo modo per migliorare le prestazioni della macchina (è molto utile quando si ha un operando che è utilizzato molto spesso durante l'esecuzione del programma)



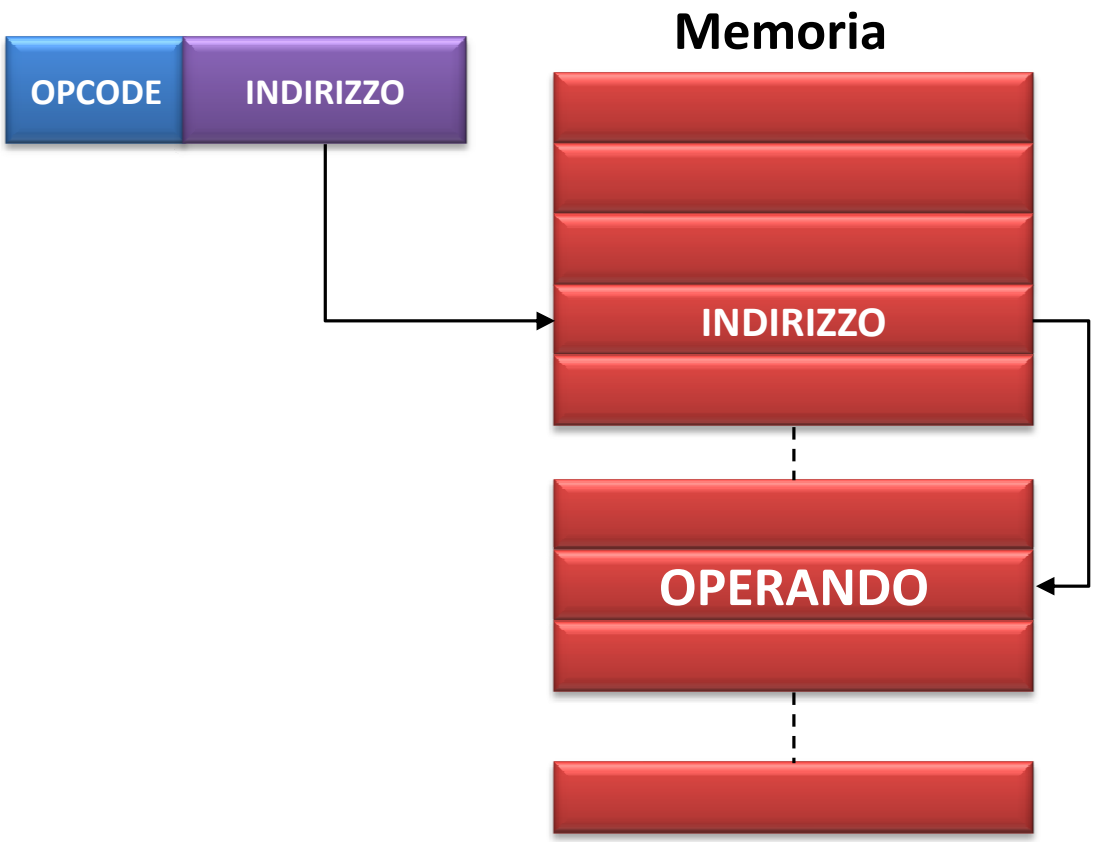

**Esempio**.

In MIPS un indirizzo a registro è facilmente riconoscibile nelle istruzioni logiche aritmetiche (es: ADD $t0,$t1,$t2)

# MODI DI INDIRIZZAMENTO

## Indirizzamento INDIRETTO

- L'**indirizzamento indiretto** fa accedere ad un l'operando attraverso un indirizzo presente nell'istruzione che '*punta*' in memoria ad un altro indirizzo
- Tale tecnica è usata, per esempio, per gestire strutture dati dinamiche come liste, code, grafi e anche per implementare tecniche di programmazione ad alto livello di "puntatori a puntatori" (utili per il passaggio per riferimento anziché per valore tra funzioni e per la gestione di matrici e stringhe)



# MODI DI INDIRIZZAMENTO

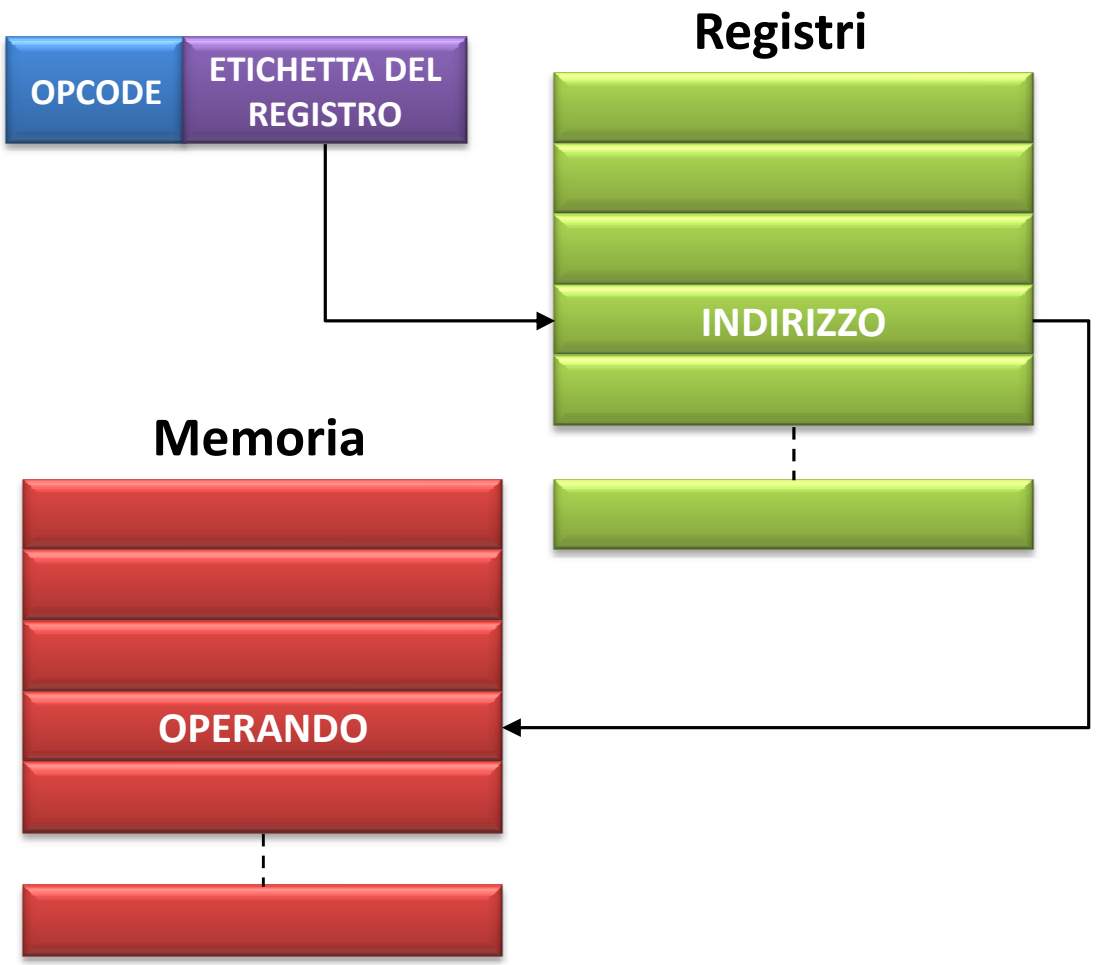
## Indirizzamento INDIRETTO A REGISTRO

- L'**indirizzamento indiretto a registro** prevede che il registro specificato nella istruzione non contiene l'operando ma l'indirizzo (definito anche **puntatore**) dell'operando
- L'utilità di tale metodo di indirizzamento è quella di:
  - poter fare riferimento ad un operando in memoria con una istruzione breve (come il modo di indirizzamento a registro)
  - modificare l'indirizzo contenuto nel registro per puntare a dati diversi usando sempre la stessa istruzione (strategia utilizzata per scorrere liste e vettori)

**Esempio**. In MIPS un indirizzo indiretto a registro è ottenibile con

**lw $t0,($t1)**

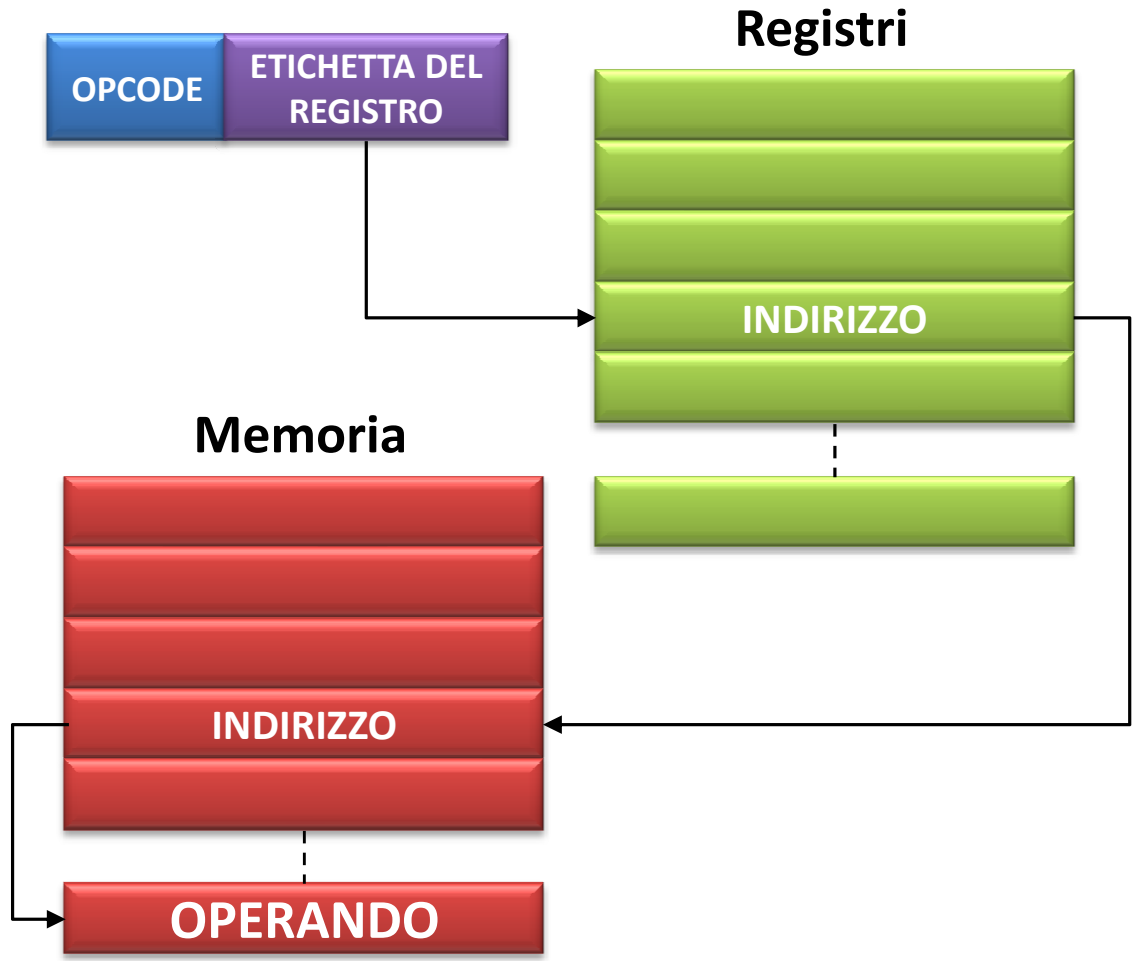
**#sposta, il contento dell'operando puntato da $t1 (cioè sito all'indirizzo contenuto da $t1) nel registro $t0**



# MODI DI INDIRIZZAMENTO

## Indirizzamento DIFFERITO INDIRETTO

- L'**indirizzamento differito indiretto** individua l'operando mediante un indirizzo memorizzato in un registro presente nell'istruzione che punta in memoria ad un altro indirizzo
- Questo modo di indirizzamento ha analogie con il sistema di **interruzione vettorizzato**. In generale consente di posizionare gli operandi o le strutture dati collegate o anche le funzioni in diverse parti della memoria

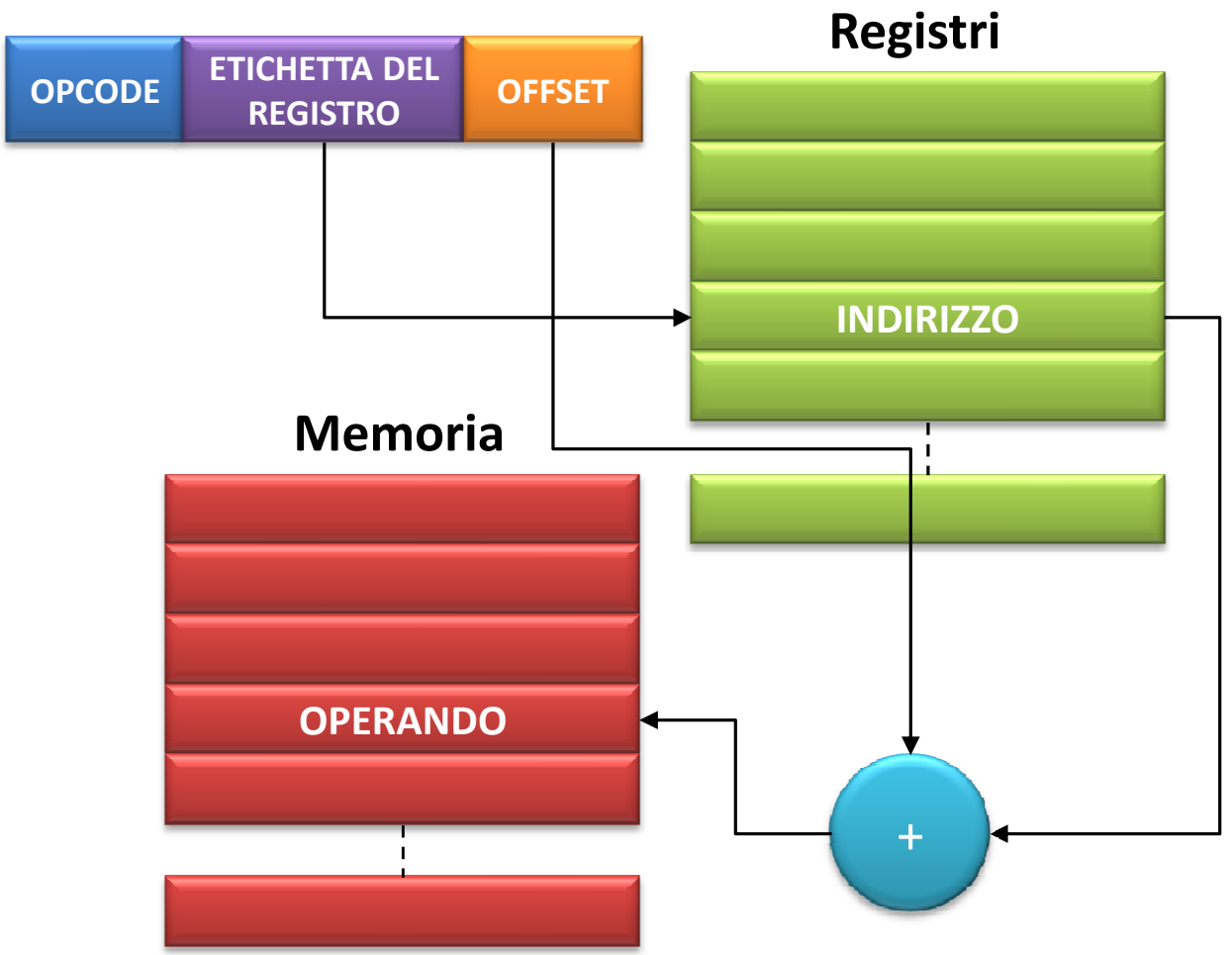


# MODI DI INDIRIZZAMENTO

## Indirizzamento CON SPIAZZAMENTO

- L'**indirizzamento con spiazzamento** consente di raggiungere l'operando dopo aver sommato al contenuto di un registro, uno spiazzamento (*offset*) contenuto nella parola successiva all'istruzione
- L'indirizzamento con spiazzamento è utile quando si hanno delle strutture dati con informazioni disposte in memoria in maniera sequenziale (stringhe, vettori, matrici, aggregati o record)

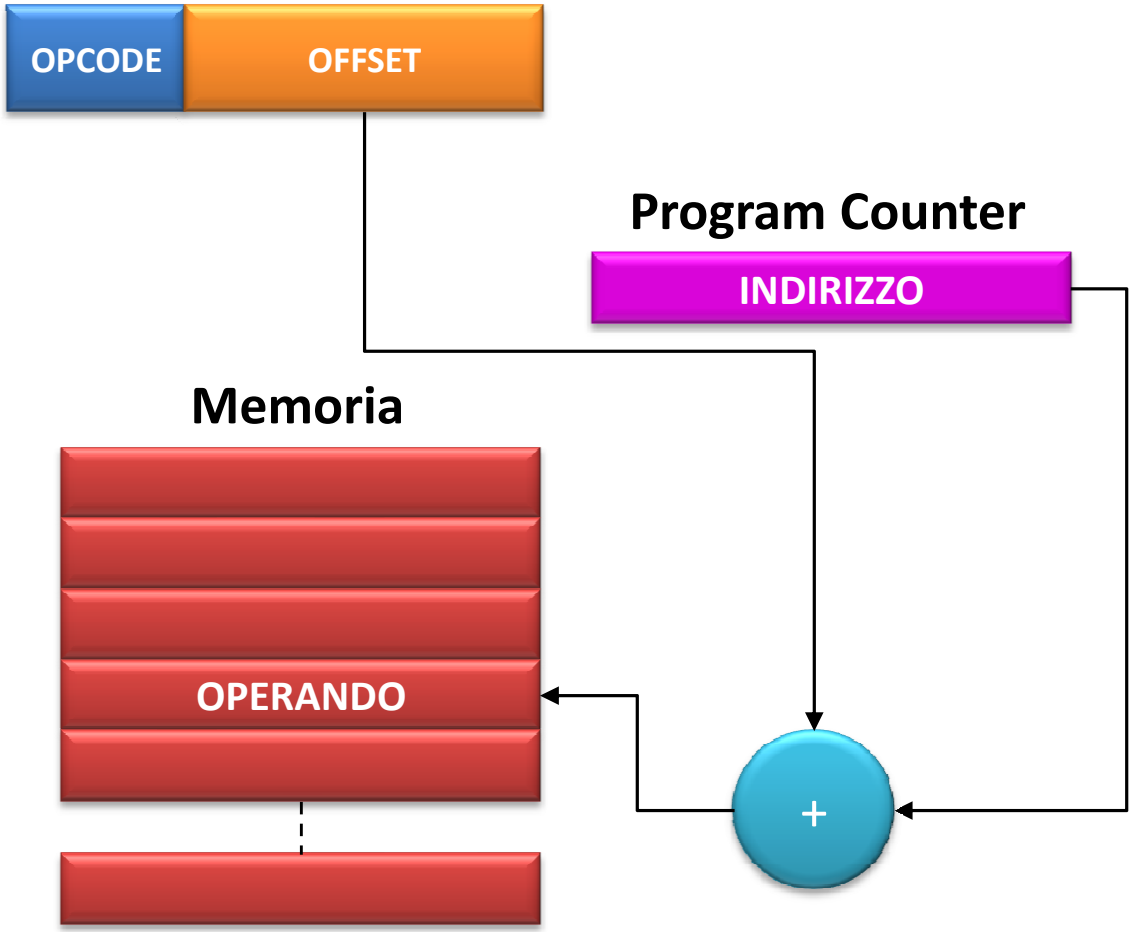
**Esempio.** In MIPS il principale modo di indirizzamento e quello con spiazzamento:
**lw $v0,2+($t0)**
**(copia il contento dell'operando cui indirizzo è dato dalla somma dell'etichetta con il contenuto del registro $t0).**



# MODI DI INDIRIZZAMENTO

## Indirizzamento RELATIVO

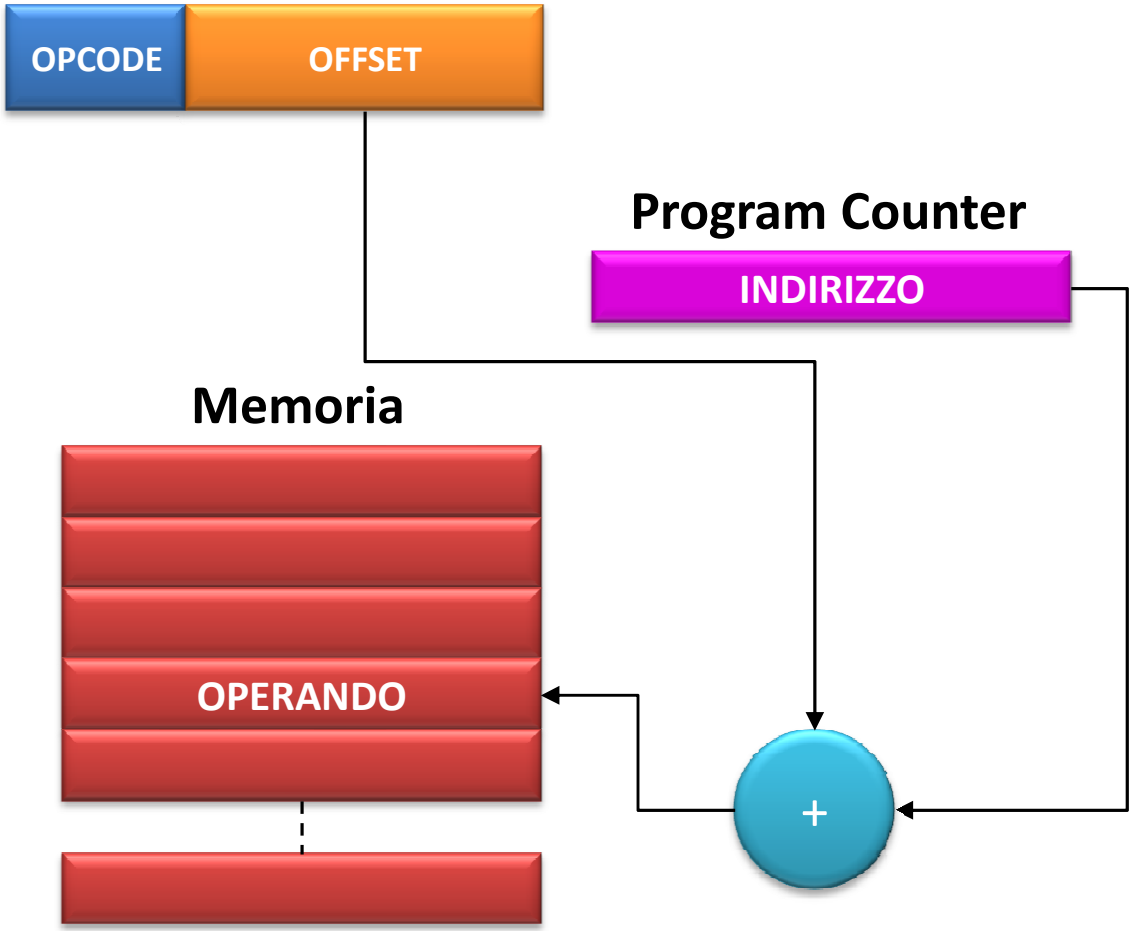
- Quando si parla di **indirizzamento relativo** si fa riferimento al fatto che l'indirizzo da considerare è relativo al contatore di programma
- L'indirizzo effettivo, in questo caso, è ottenuto dalla somma tra lo spiazzamento (*offset*) presente nell'istruzione ed il contenuto del PC (ovvero è un indirizzamento indiretto con spiazzamento il cui registro di riferimento è il PC)



# MODI DI INDIRIZZAMENTO

## Indirizzamento RELATIVO

- Tale modo di indirizzamento è utile per realizzare programmi aventi la caratteristica di essere **indipendenti dalla posizione del codice** (**PIC**, *Position Indipendent Code*)
- Infatti, i programmi PIC, usano **salti relativi** che non fanno riferimento ad alcun indirizzo assoluto, ma solamente alla distanza tra le istruzioni



# MODI DI INDIRIZZAMENTO

## Indirizzamento RELATIVO (esempio)

$$f(x,y) = \begin{cases} x+1 \quad y-1 \quad se \quad x \le 0 \\ x-1 \quad y+1 \quad altrimenti \end{cases}$$

**INDIRIZZAMENTO ASSOLUTO**

| Indirizzo di memoria | Istruzione |
|---|---|
| 100 | START |
| 104 | LI $t0,5 |
| 108 | LI $t1,6 |
| 112 | BGTZ $t0,128 |
| 116 | ADD $t0,$t0,1 |
| 120 | SUB $t1,$t1,1 |
| 124 | J 136 |
| 128 | SUB $t0,$t0,1 |
| 132 | ADD $t1,$t1,1 |
| 136 | END |

**INDIRIZZAMENTO RELATIVO**

| Indirizzo di memoria | Istruzione |
|---|---|
| 100 | START |
| 104 | LI $t0,5 |
| 108 | LI $t1,6 |
| 112 | BGTZ $t0, 16($PC) |
| 116 | ADD $t0,$t0,1 |
| 120 | SUB $t1,$t1,1 |
| 124 | J 12($PC) |
| 128 | SUB $t0,$t0,1 |
| 132 | ADD $t1,$t1,1 |
| 136 | END |

# MODI DI INDIRIZZAMENTO

## Indirizzamento RELATIVO (esempio)

Comportamento nel caso di traslazione del programma in memoria

**INDIRIZZAMENTO ASSOLUTO**

| Indirizzo di memoria | Istruzione |
|---|---|
| 200 | START |
| 204 | LI $t0,5 |
| 208 | LI $t1,6 |
| 212 | BGTZ $t0,128 |
| 216 | ADD $t0,$t0,1 |
| 220 | SUB $t1,$t1,1 |
| 224 | J 136 |
| 228 | SUB $t0,$t0,1 |
| 232 | ADD $t1,$t1,1 |
| 236 | END |

**NON CORRETTO**

**INDIRIZZAMENTO RELATIVO**

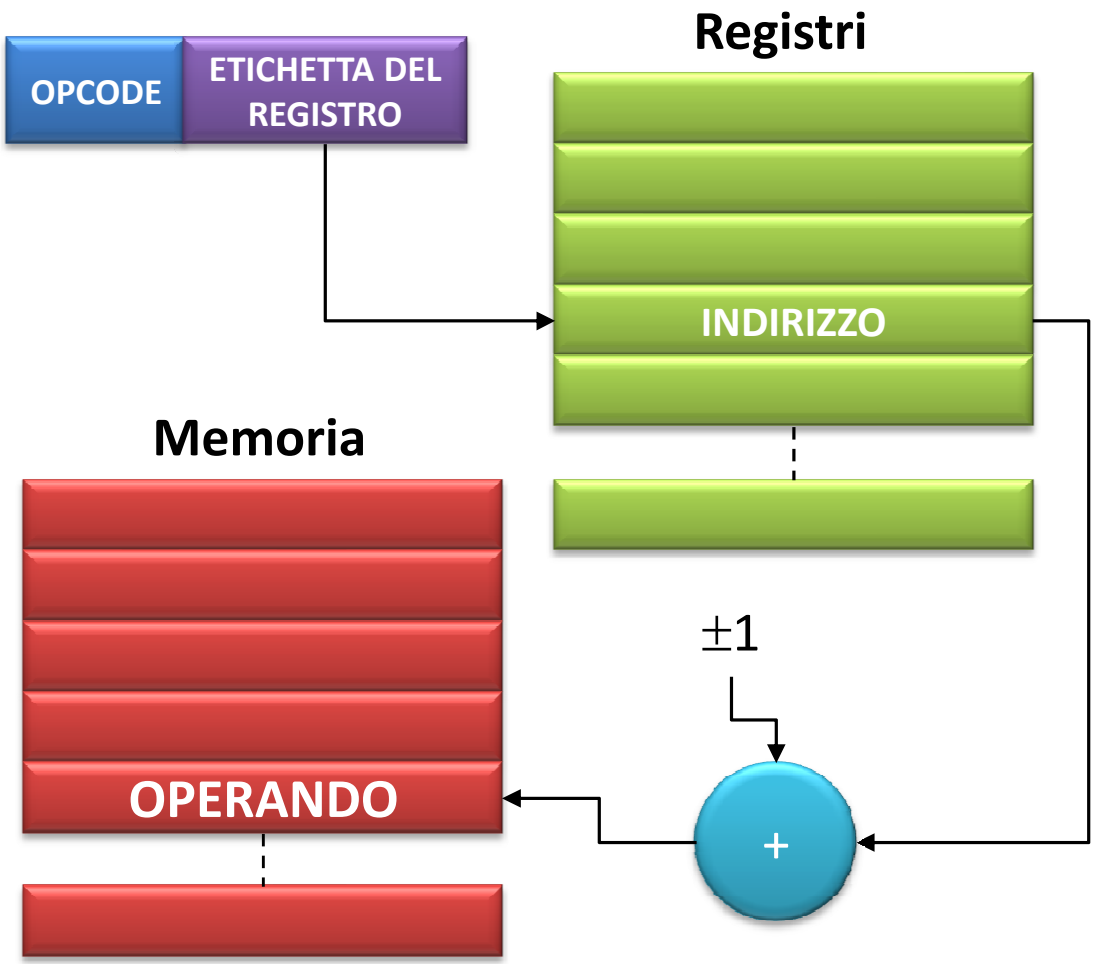
| Indirizzo di memoria | Istruzione |
|---|---|
| 200 | START |
| 204 | LI $t0,5 |
| 208 | LI $t1,6 |
| 212 | BGTZ $t0, 16($PC) |
| 216 | ADD $t0,$t0,1 |
| 220 | SUB $t1,$t1,1 |
| 224 | J 12($PC) |
| 228 | SUB $t0,$t0,1 |
| 232 | ADD $t1,$t1,1 |
| 236 | END |

**CORRETTO**

# MODI DI INDIRIZZAMENTO

## Indirizzamento PRE-POST INCREMENTO

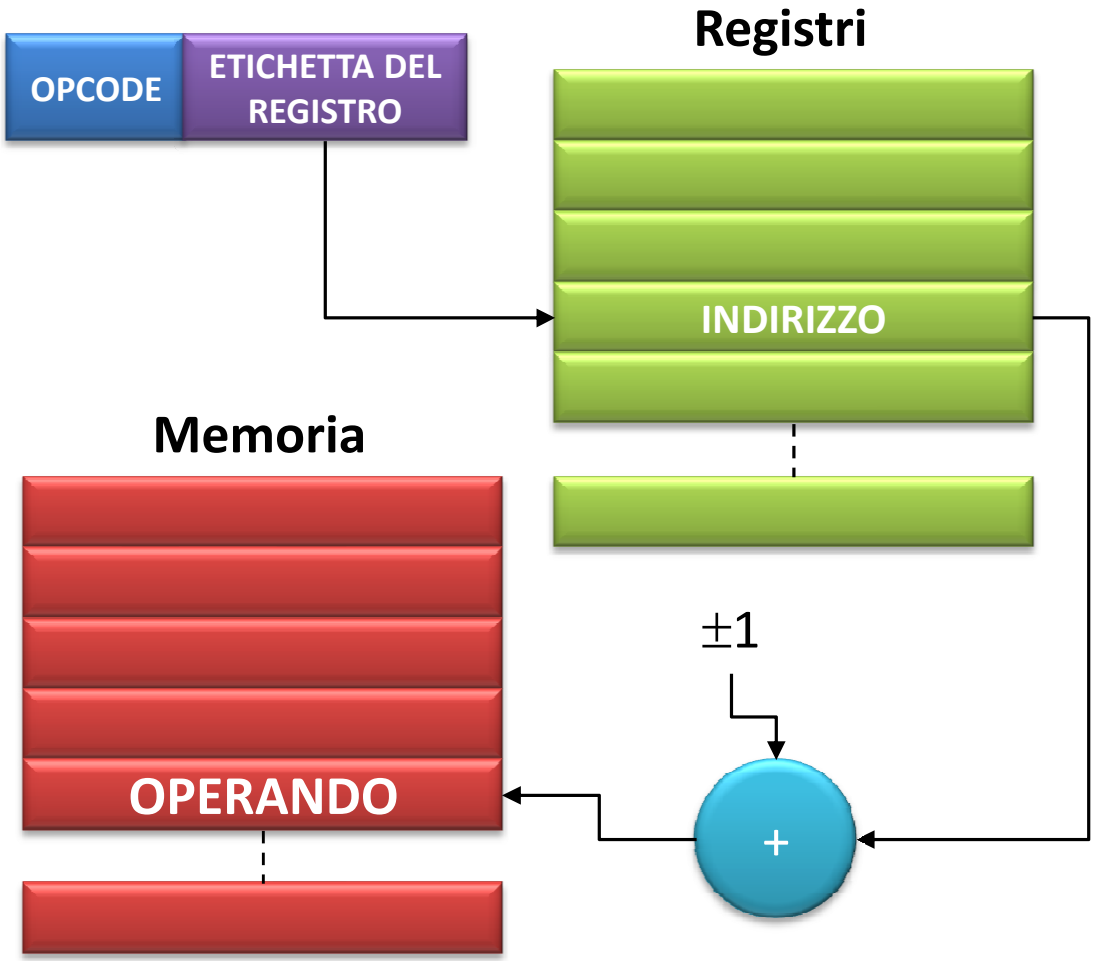
- L'**indirizzamento con pre/postdecremento (pre/postincremento)** è simile all'indiretto a registro solo che il contenuto nel registro è automaticamente decrementato (incrementato) prima o dopo l'esecuzione dell'istruzione stessa
- Si elimina quindi l'istruzione di decremento (incremento) che si eseguivano sul registro usato come puntatore; offrendo una maggiore velocità di esecuzione perché non è richiesta una suppletiva fase di fetch per svolgere l'esecuzione dell'istruzione di decremento (incremento)

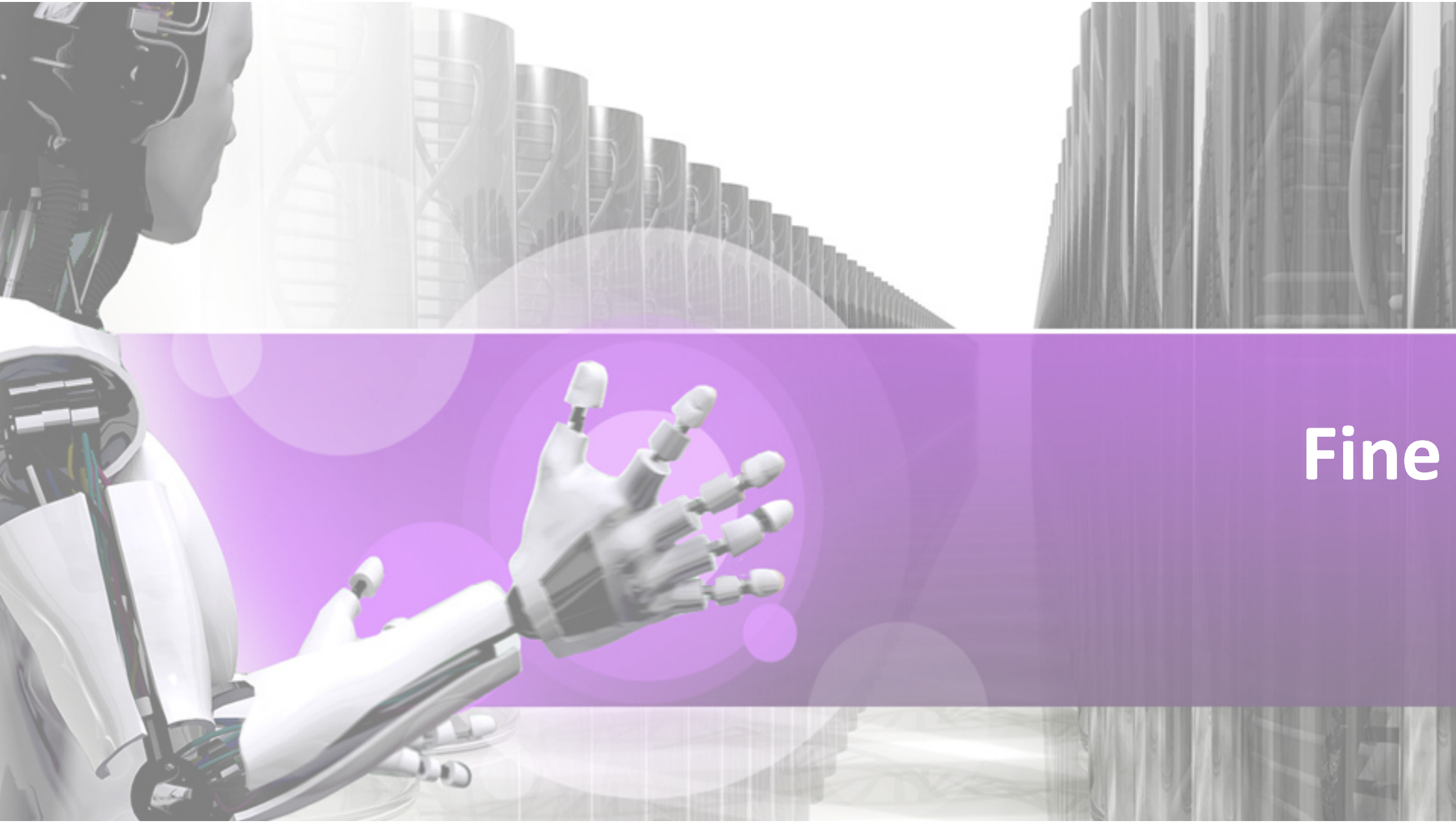


# MODI DI INDIRIZZAMENTO

## Indirizzamento PRE-POST INCREMENTO

- L'utilità sta nel poter accedere facilmente ad insiemi di dati allocati in memoria sequenzialmente (cioè dati disposti uno di seguito all'altro, come i vettori o le stringhe) ed in particolare per le operazioni di **estrazioni** (POP) ed **inserimento** (PUSH) **della pila** (o canasta) che ha un modalità LIFO (Last In First out) ed a cui è dedicata una zona di memoria (*stack zone*) molto utile per la gestione dell'annidamento di subroutine




Fine