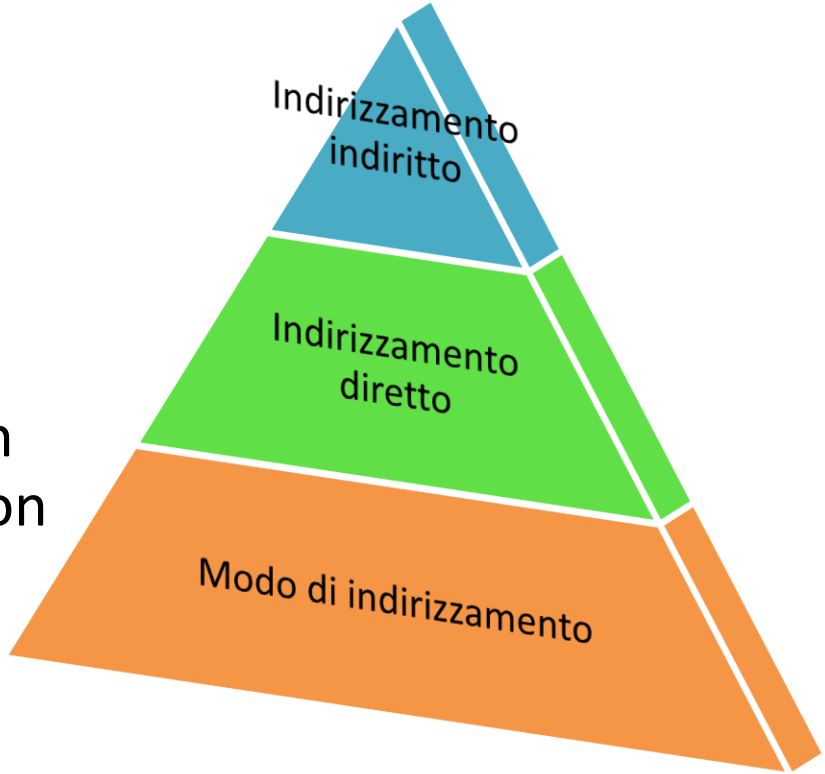
Architettura degli Elaboratori Elettronici

Dott. Franco Liberati
liberati@di.uniroma1.it

ARGOMENTI DELLA LEZIONE

- Definizione
- Indirizzamenti diretti: immediato, assoluto, a registro
- Indirizzamenti indiretti: indiretto con registro, con spiazzamento, relativo, con predecremento, con postincremento



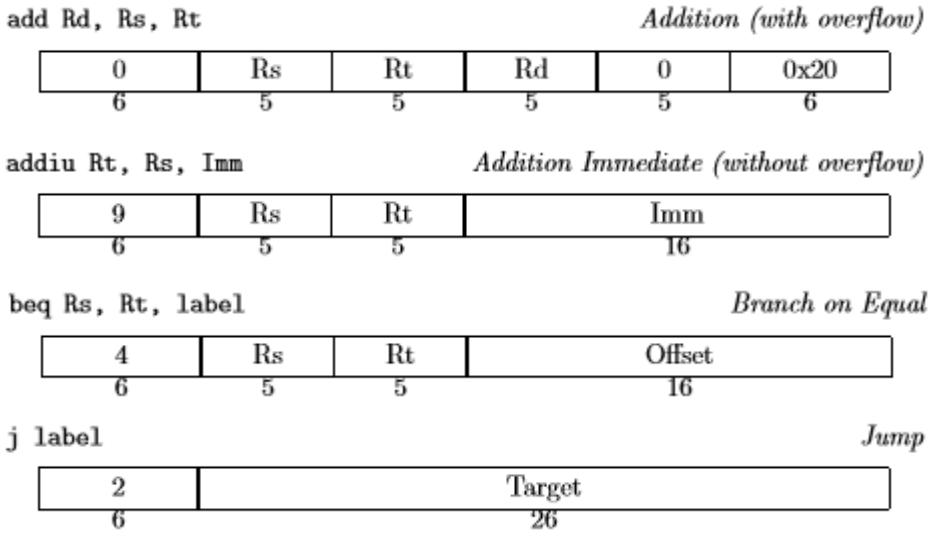
Modi di indirizzamento

MODI DI INDIRIZZAMENTO

Generalità

- Una istruzione macchina contiene una suddivisione in campi in cui una parte è **operazionale** (OPCODE o Codice operativo) e specifica la classe di istruzione da eseguire ed un secondo **campo indirizzo** (ADDRESS MODE o Campo Indirizzo o modo di indirizzamento) che indica quale è **l'operando**, cioè quale è il registro/i o la locazione di memoria che contiene l'operando sul quale si deve applicare l'istruzione oppure un **indirizzo** (es.: salti condizionati e incondizionati)
- Poiché si può far riferimento ad un operando o ad un indirizzo è lecito parlare di **indirizzo effettivo**

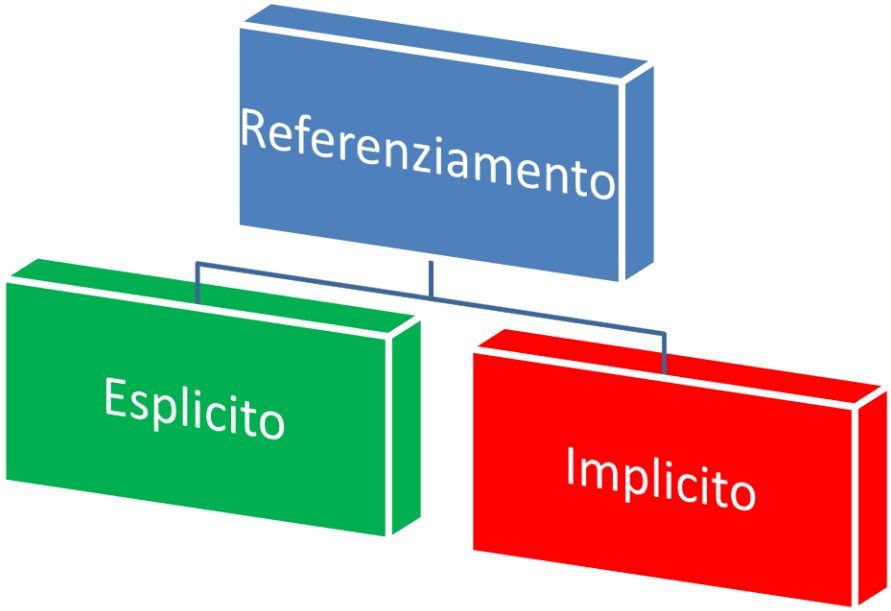
Codice operativo (OPCODE)	Modo di indirizzamento (ADDRESS MODE)



MODI DI INDIRIZZAMENTO

Generalità

- Il modo di indirizzamento può essere espresso in maniera **esplicita** oppure può essere omesso, in questo ultimo caso di parla di **modalità implicita**



MODI DI INDIRIZZAMENTO

Generalità: implicito

- In alcune macchine (es. Intel 8085) 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 degli accumulatori)
- Grazie al **referenziamento implicito** è possibile utilizzare delle istruzioni con **lunghezza fissa** e **minima**
- L'indirizzamento implicito **non può essere utilizzato per le operazioni di trasferimento** come la LOAD e la STORE perché in questo caso è necessario esplicitare dove trasferire o da dove prelevare il dato in memoria

Codice operativo (OPCODE)

SET ISTRUZIONI 8085

PCHL Trasferisce il contenuto nel registro HL nel program counter

RRC svolge l'operazione di rotate di una posizione verso destra di un valore contenuto in un accumulatore della ALU

ADD B Somma il contenuto del registro B con un dato salvato nell'accumulatore della ALU e restituisce il risultato in un altro accomulatore interno alla ALU

INR B Incrementa il contenuto del registro B di una unità

MODI DI INDIRIZZAMENTO

Limite indirizzamento implicito

- **È possibile constatare come NON sia possibile ricorrere all'indirizzamento implicito qualora vi siano operazioni di trasferimento in memoria**

```
LDA 2050  #Copia il contenuto della locazione di memoria 2050 in un
          #accumulatore della ALU
MVI B 05  #Copia il valore 5 nel registro B
ADD B     #Somma  il valore del registro B con il valore contenuto nell'accumulatore
          #dell'ALU il risultato si trova in un accumulatore-risultato dell'ALU
STA 2054  #Copia il valore nell'accumulatore-risultato dell'ALU nella locazione 2054
```

MODI DI INDIRIZZAMENTO

Generalità: esplicito

- La maniera più semplice e diretta di individuare un operando in memoria è quella di indicare il suo indirizzo (**indirizzamento assoluto**) nell'ADDRESS MODE

Codice operativo (OPCODE)	Modo di indirizzamento (ADDRESS MODE)

```
MOVE d0,0x123456      #modo di indirizzamento assoluto:
                      #si esplicita la locazione in memoria
                      #dove risiede l'operando
                      # in MIPS è
                      # lw $t0, 0x123456
```

Esplicito:

```
LW $t0,ETICHETTA_CELLA_MEMORIA
MUL $t1,$t3,$t4
J ETICHETTA_INTERNA_AL PROGRAMMA
```

MODI DI INDIRIZZAMENTO

Indirizzo effettivo

- Prima di entrare nel dettaglio dei più comuni modi di indirizzamento bisogna esplicitare il concetto di **indirizzo effettivo** (EA o *effective address*)
- Nelle istruzioni di trasferimento dati o in quelle logico-aritmetiche l'**indirizzo effettivo è l'indirizzo degli operandi interessati** (si accede in un'area di memoria centrale nel quel risiedono i dati – Memoria Dati MIPS)
- In una istruzione di salto o di chiamata a sottoprogramma, l'**indirizzo effettivo è quello dell'istruzione a cui si vuole saltare** (si accede ad un'area di memoria nella quale è conservato il programma – Memoria Istruzioni MIPS)

MODI DI INDIRIZZAMENTO

Esplicito: proprietà

- Il modo di indirizzamento esplicito consente:
  - di **accedere ad una locazione di memoria** il cui indirizzo non è noto nel momento in cui il programma è scritto; ma è calcolato nel momento il cui il programma è eseguito (accesso a strutture dati come vettori, liste)
  - di **manipolare gli indirizzi**, cioè permettere delle operazioni su di essi
  - 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

Etichetta

- Parlando di modi di indirizzamento, inoltre, si fa spesso riferimento all'**etichetta**
- Una **etichetta** è 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

Principali
Modi di indirizzamento

MODI DI INDIRIZZAMENTO

Indirizzamento IMMEDIATO

- L'**indirizzamento immediato** è così definito perché l'operando si trova nella posizione di memoria immediatamente successiva all'istruzione e pertanto, è nel corpo del programma e non in una area dati
- L'indirizzamento immediato è utile per inizializzazioni o per la definizione di costanti (maschere)
- Un uso eccessivo può incrementare la lunghezza del programma in memoria
- In alcune macchine nel caso in cui si voglia inserire numero lunghi 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 per problemi di efficienza (si preferisce escludere due accessi in memoria) alla lunghezza della parola. Nel caso di valori superiori di 16bit l'istruzione immediate si sdoppia**

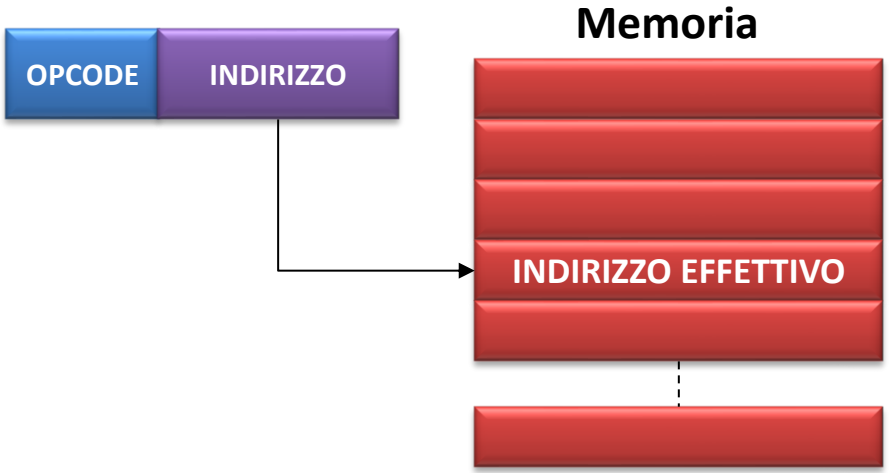
Esempio.
In assembly 68000 un indirizzamento immediato è **MOVE d0,#35**

In MIPS un indirizzamento immediato è :
li $t0,35 **#pone il valore 35 nel registro $t0**
NB: l'istruzione Load Immediate è tradotta dall'assemblatore MIPS in:
li $t0,35 corrisponde a **addui $t0,$0,35**
li $t0,70000 corrisponde a
lui $at,1 **#copia il valore 1 nei primi sedici bit del #registro $at**
ori $t0,$at,4464 **#svolge l'or del valore 4464 con $at**

MODI DI INDIRIZZAMENTO

Indirizzamento DIRETTO

- L'**indirizzamento diretto** specifica l'indirizzo effettivo di una parola di memoria: è pertanto allocato nella parola immediatamente successiva a quella contente l'istruzione che deve operarlo (istruzione a lunghezza variabile)
- L'indirizzo è fissato nel momento in cui si scrive il programma anche se è descritto da una etichetta (l'assemblatore tradurrà l'etichetta in indirizzo)
- Osservazione: in alcuni testi è riportato come indirizzamento **assoluto**

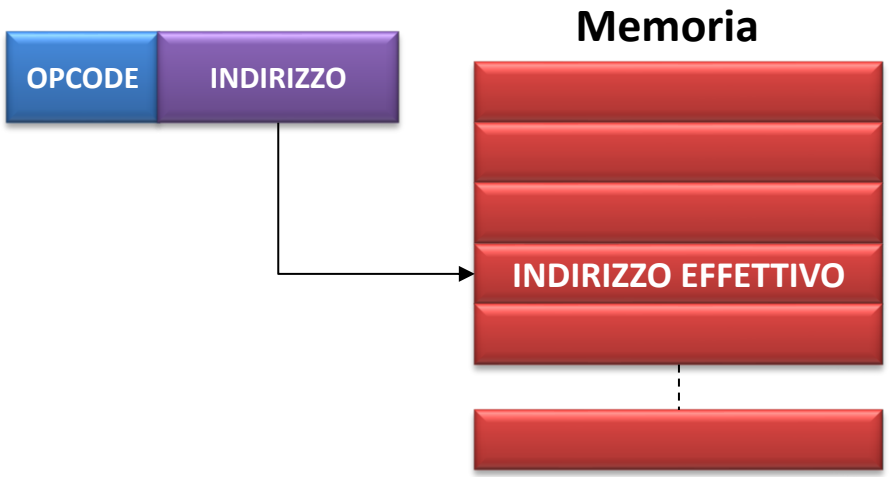


Esempio.
Nell'assembly 68000 un indirizzamento assoluto è :
MOVE d0,112346 (carica il valore contenuto nell'indirizzo 112346 nel registro d0)
In MIPS si ha **lw $t0, etichetta**

MODI DI INDIRIZZAMENTO

Indirizzamento DIRETTO

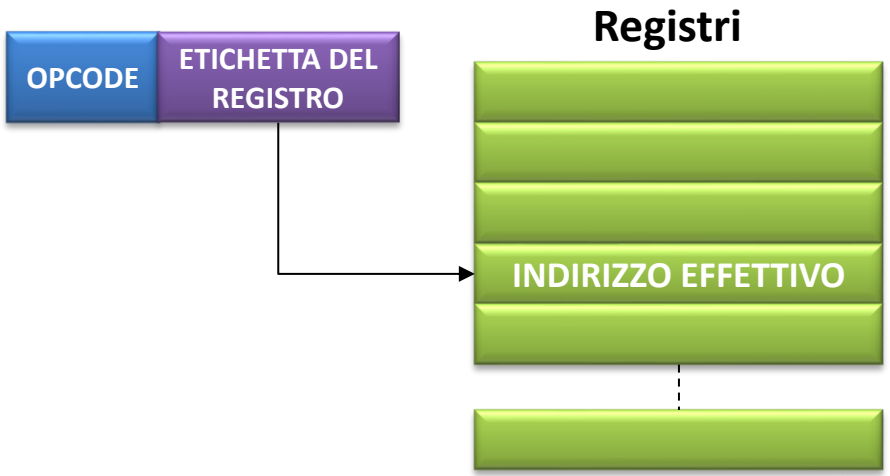
- L'**indirizzamento diretto** è utile quando si deve operare con una dato che si trova in una posizione di memoria fissata o quando si deve fare un salto ad una istruzione che si trova ad una posizione prestabilita in memoria
- In ogni caso, come nell'immediato, occupa spazio in memoria ed è meno efficiente rispetto altri modi di indirizzamento (richiedere almeno un ulteriore accesso in memoria se si ha un indirizzo con un valore molto alto)



MODI DI INDIRIZZAMENTO

Indirizzamento A REGISTRO

- L'**indirizzamento a registro** specifica l'etichetta del registro in cui è presente l'indirizzo effettivo (di solito un operando o anche un indirizzo, es: MOTOROLA 68000 aveva dei *Registri Indirizzi*)
- Ogni CU è dotata di un certo numero di registri interni detti registri a 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 riservato per l'indirizzo è breve perché servono pochi bit per selezionare un registro interno (possibilità di utilizzare istruzioni a dimensione fissa)
  - per rintracciare gli operandi non occorrono accessi alla memoria principale perché i registri sono integrati nella CU
- Si usa questo modo quando si ha un operando o un indirizzo che è utilizzato molto spesso durante l'esecuzione del programma

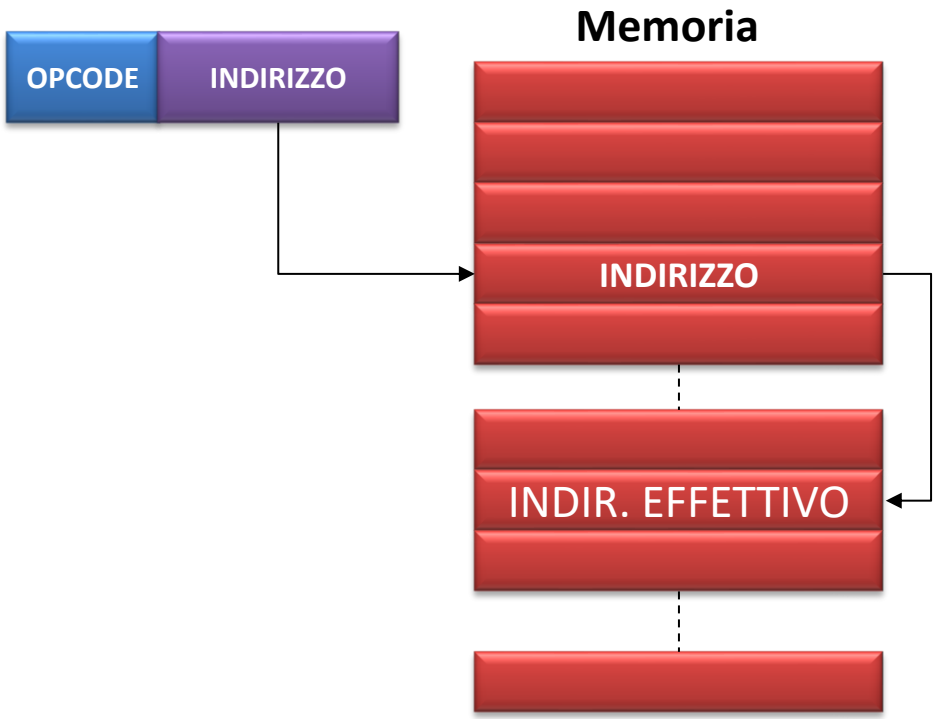


Esempio.
In MIPS un indirizzo a registro è facilmente riconoscibile nell'istruzione **move $t0,$t1** (copia il contento del registro $t1 nel registro $t0) o **add $t0,$t1,$t2**

MODI DI INDIRIZZAMENTO

Indirizzamento INDIRETTO

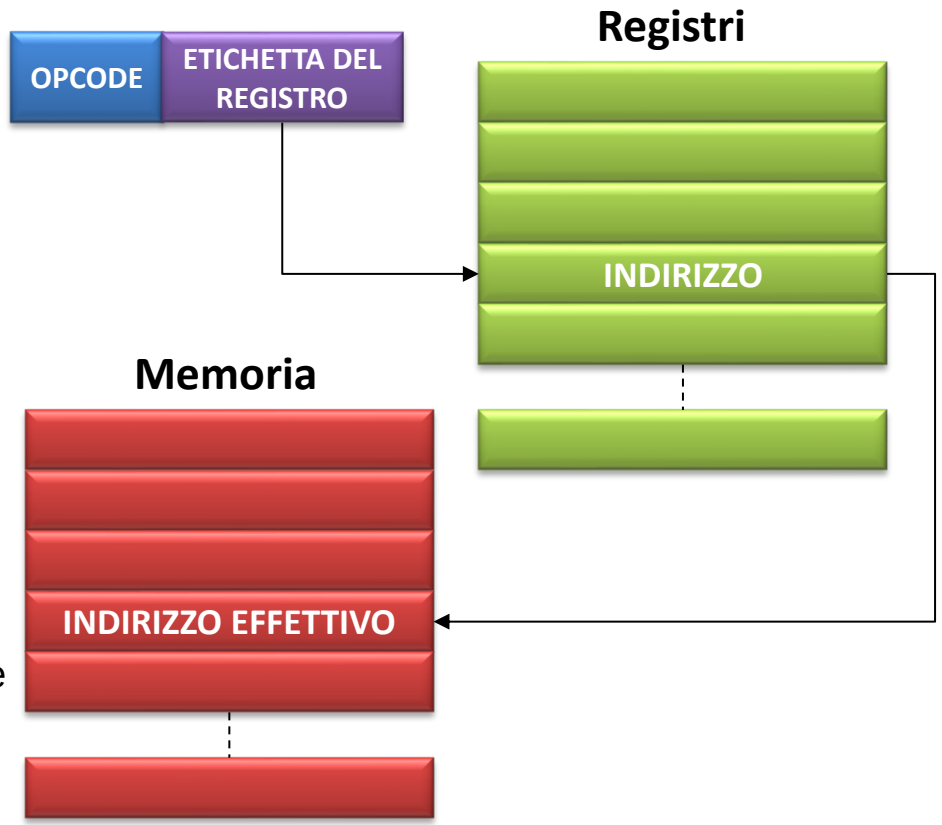
- L'**indirizzamento indiretto** fa accedere ad un indirizzo effettivo attraverso un indirizzo presente nell'istruzione che punta in memoria ad un altro indirizzo
- Tale tecnica è usata, per esempio, per condividere delle variabili tra il programma principale e una funzione (passaggio per riferimento). In questo modo la funzione chiamata dal programma principale è in grado di manipolare il valore della variabile accedendo alla locazione di memoria in cui l'operando è conservato
- Questo modo di indirizzamento, usato anche nel Motorola 68000, è impiegato nell'interruzione vettorizzata



MODI DI INDIRIZZAMENTO

Indirizzamento INDIRETTO A REGISTRO

- L'**indirizzamento indiretto a registro** prevede che il registro specificato nella istruzione non contiene l'indirizzo effettivo (in questo caso l'operando) ma un indirizzo (definito anche **puntatore**) all'indirizzo effettivo (cioè l'operando)
- L'utilità di tale metodo di indirizzamento è quella di:
  - poter fare riferimento ad un operando in memoria principale con una istruzione breve (come 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)



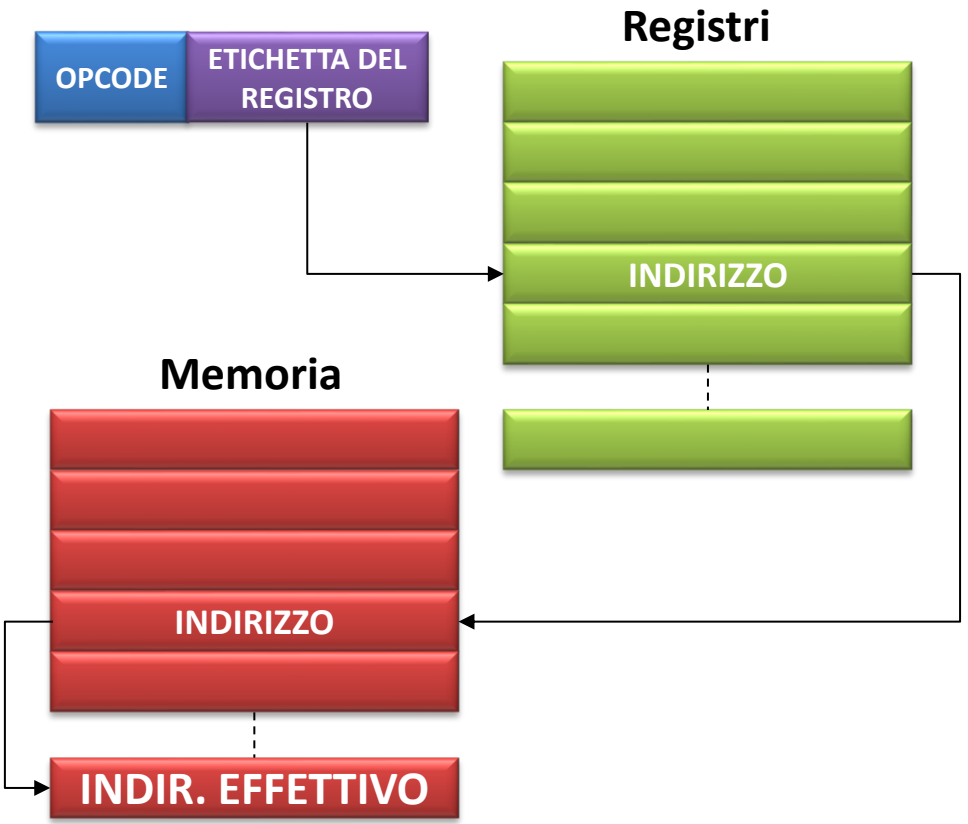
Esempio. In MIPS un indirizzo indiretto a registro è ottenibile con

```
lw $t0,($t1) #copia, il contento dell'operando
             #'puntato' dal registro $t1 nel registro $t0
```

MODI DI INDIRIZZAMENTO

Indirizzamento DIFFERITO INDIRETTO

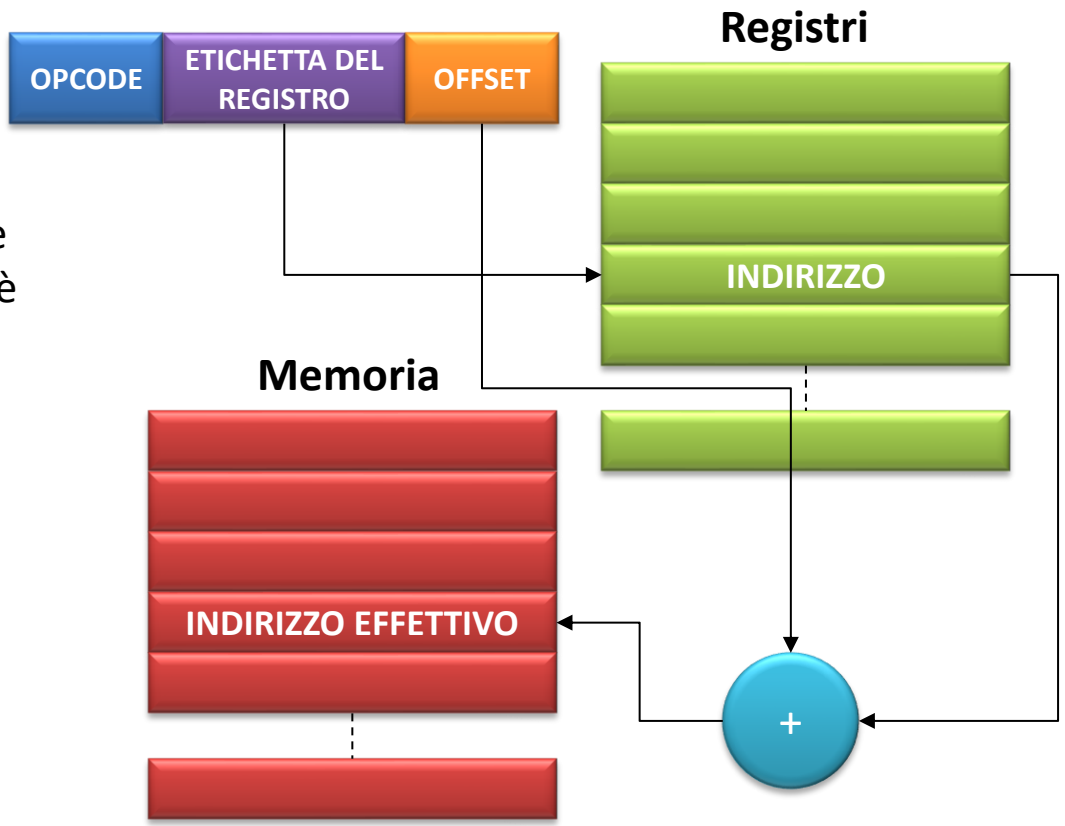
- L'**indirizzamento differito indiretto** individua l'indirizzo effettivo (un operando) mediante un indirizzo memorizzato in registro presente nell'istruzione che punta in memoria ad un altro indirizzo
- Questo modo di indirizzamento è utile per gestire strutture dati



MODI DI INDIRIZZAMENTO

Indirizzamento CON SPIAZZAMENTO

- L'**indirizzamento con spiazzamento** consente di raggiungere l'indirizzo effettivo 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)

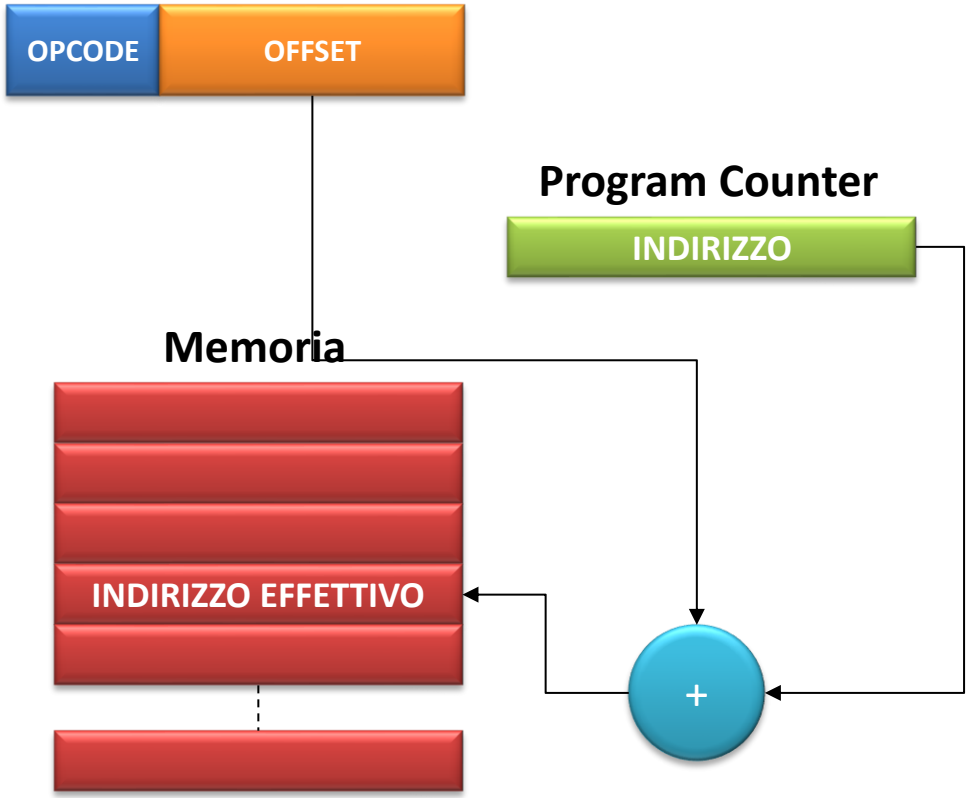


Esempio. In MIPS il principale modo di indirizzamento e quello con spiazzamento:
lw $v0,2+($t0) **#copia il contento #dell'operando cui indirizzo è dato dalla #somma dell'etichetta con il contenuto del #registro $t0**

MODI DI INDIRIZZAMENTO

Indirizzamento RELATIVO

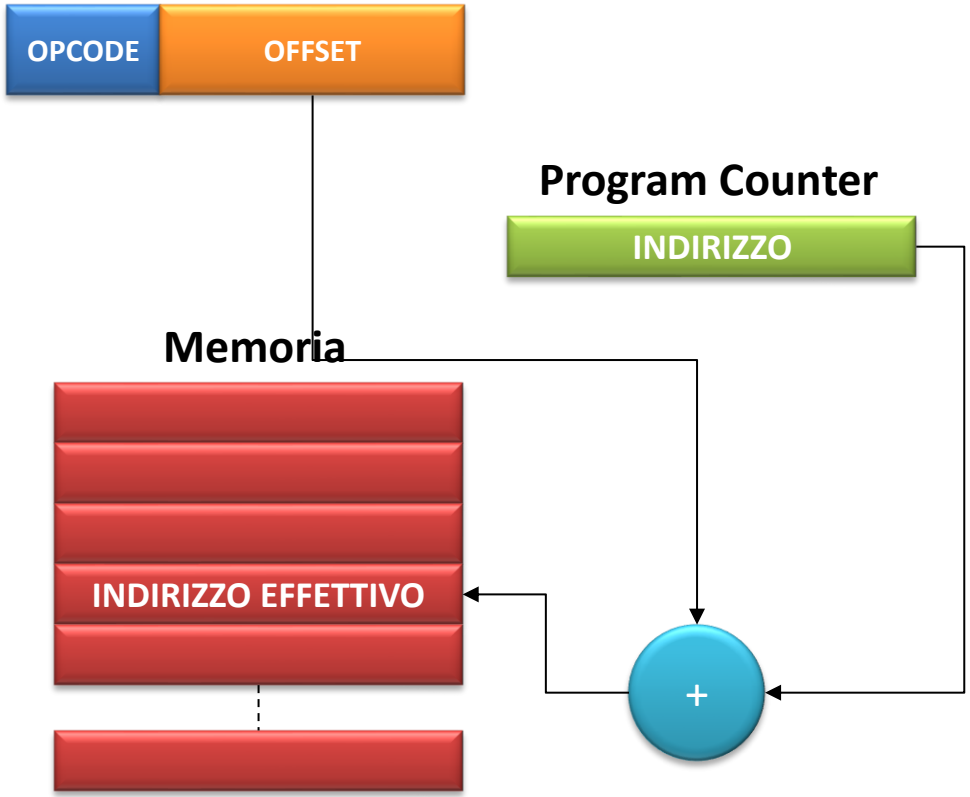
- Quando si parla di **indirizzamento relativo** si fa riferimento al fatto che l'indirizzo è relativo al Program Counter
- L'indirizzo effettivo, in questo caso, è dato dalla somma tra lo spiazzamento (offset) presente nell'istruzione ed il contenuto del PC (questo si può vedere come un indirizzamento 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)

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

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	MUL $t2,$t0,$t1
140	MOVE $a0,$t2
144	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	MUL $t2,$t0,$t1
140	MOVE $a0,$t2
144	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	MUL $t2,$t0,$t1
240	MOVE $a0,$t2
244	END

NON CORRETTO

INDIRIZZAMENTO RELATIVO

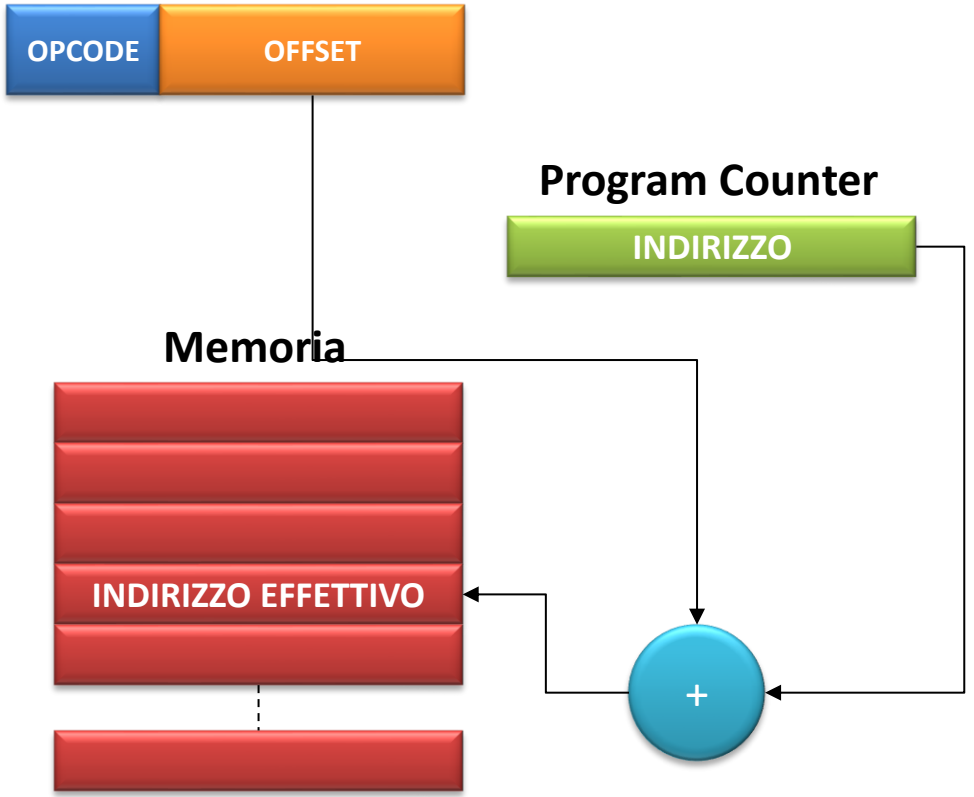
Indirizzo di memoria	Istruzione
200	START
204	LI $t0,5
208	LI $t1,6
212	BGTZ $t0,16($PC)
216	ADD $t0,$t0,1
220	SUB $t1,$t1,1
224	J 12($PC)
228	SUB $t0,$t0,1
232	ADD $t1,$t1,1
236	MUL $t2,$t0,$t1
240	MOVE $a0,$t2
244	END

CORRETTO

MODI DI INDIRIZZAMENTO

Indirizzamento RELATIVO

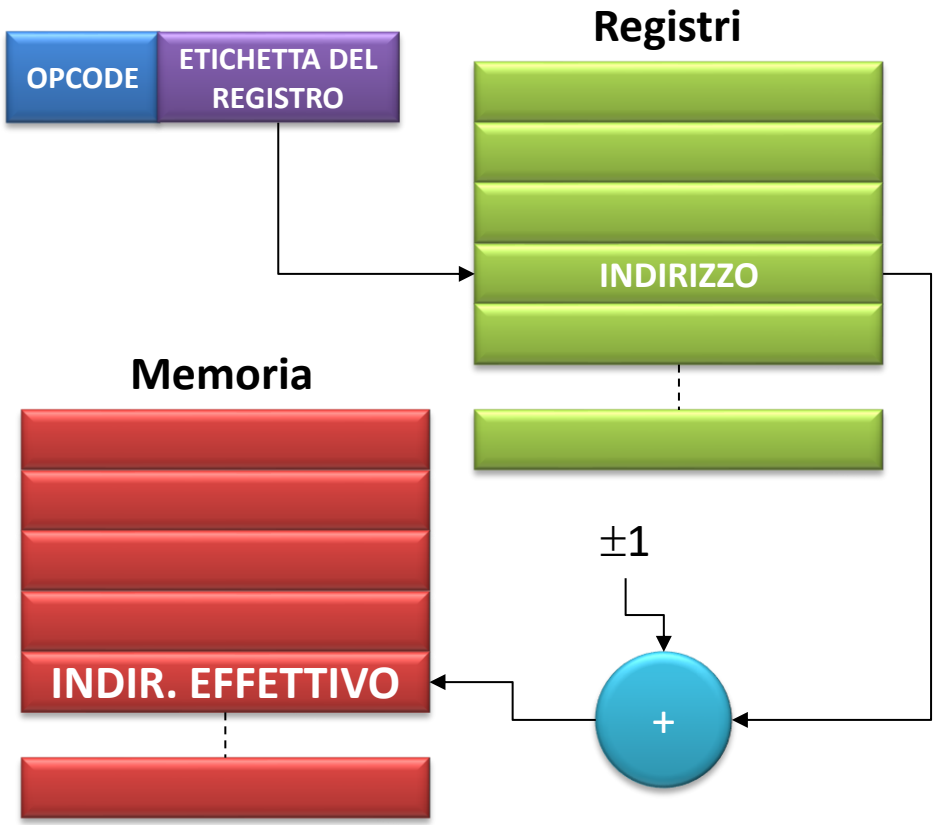
- Se non si usa il modo di indirizzamento relativo il programma va quanto meno riassemblato e linkato
- In seguito al PIC furono proposti il Memory Management Unit (discussa in seguito) e, poi, la paginazione-Memoria Virtuale (discussa in seguito) dove però ci sono componenti hardware aggiuntivi che incrementano i costi



MODI DI INDIRIZZAMENTO

Indirizzamento PRE-POST INCREMENTO

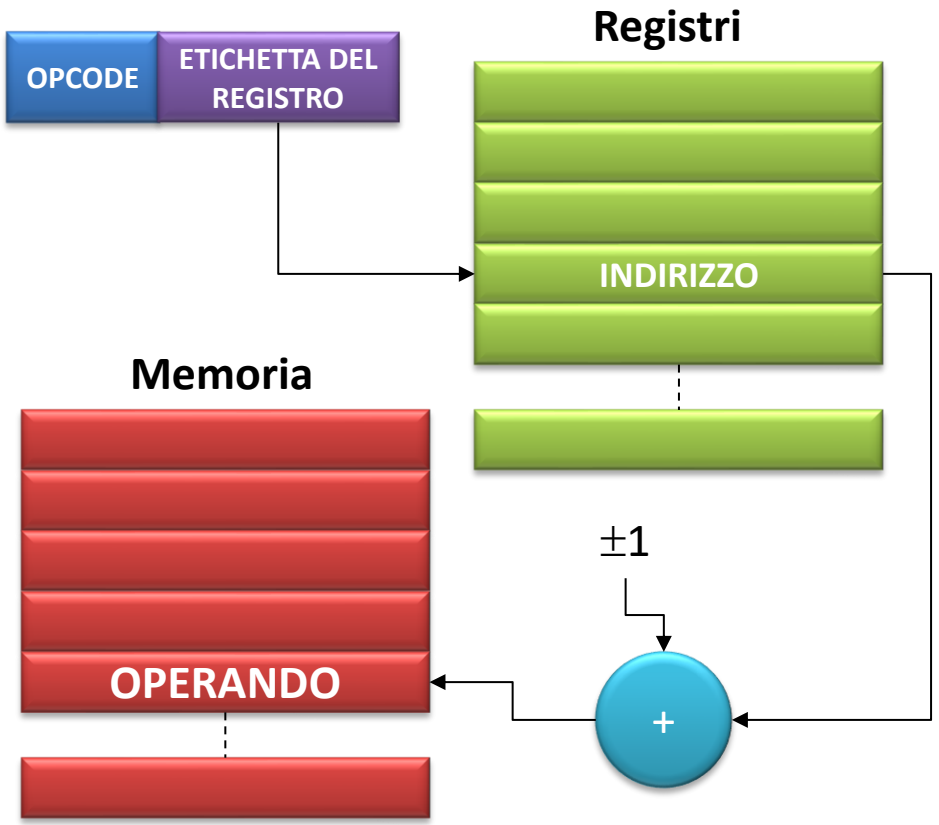
- L'**indirizzamento con pre/postdecremento (pre/postincremento)** è simile all'indiretto a registro solo che il contenuto nel registro è automaticamente decrementato (incrementato) prima o dopo l'esecuzione dell'istruzione stessa
- Si elimina quindi l'istruzione di decremento (incremento) che si eseguivano 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*)



Fine