ARCHITETTURA DELL' ELABORATORE

Studiare l'architettura di un elaboratore vuol dire, una volta stabilito il livello, studiare:

I TIPI DI DATI
- come rappresentarli
- quale tipo di accesso
- come operare

L'ORGANIZZAZIONE DEI REGISTRI
- Generali e
- Dedicati

I RIFERIMENTI AI DATI
- per nome
- per indirizzo
- per valore

L'INSIEME DELLE ISTRUZIONI
- Per tipologia di formato e di operazione

L'ORGANIZZAZIONE DELLA MEMORIA

Architettura di un elaboratore

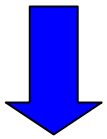
- Architettura → progetto dello schema funzionale (organizzazione) di un elaboratore:
  - Set di istruzioni
  - Componenti HW

- Due parti essenziali
  - ***Instruction Set Architecture (ISA)*****:** definizione del set di istruzioni a livello di linguaggio macchina
  - ***Hardware System Architecture (HSA)*****:** definizione del progetto logico delle parti (sottosistemi) HW e della organizzazione del flusso di dati tra tali sottosistemi

ISA e HSA

ISA

Specifica un elaboratore dal punto di vista del programmatore



Determina le caratteristiche "computazionali" dell'elaboratore

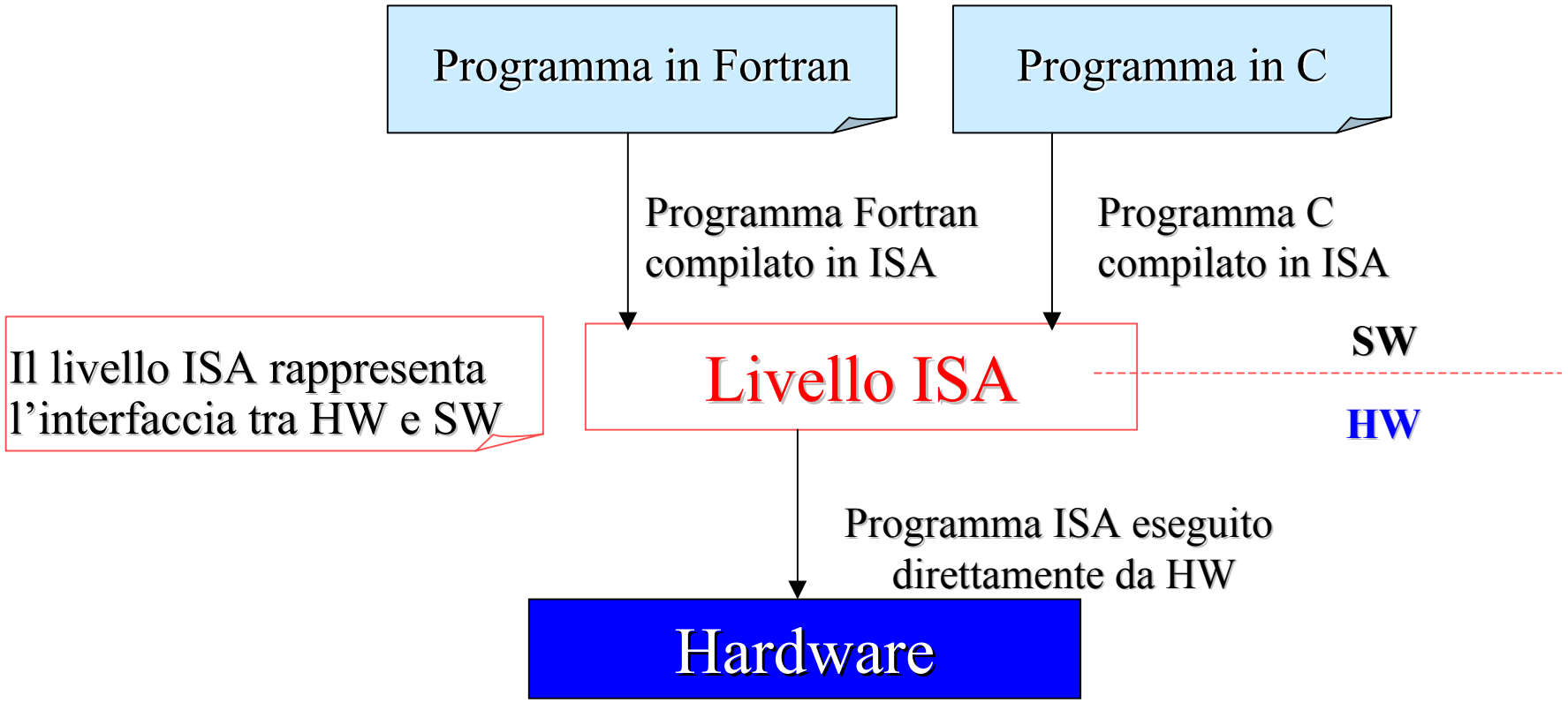
HSA

Specifica l'organizzazione dei sottosistemi hardware

Determina le caratteristiche "strutturali" dell'elaboratore

Instruction Set Architecture (ISA)

Il livello ISA specifica come il livello di ***linguaggio macchina*** interagisce con il livello HW



Le strutture di informazione a livello di Linguaggio Macchina (ISA)

1. **I riferimenti alle informazioni** *si realizzano specificando la* ***LOCAZIONE*** *ed il* ***TIPO***

- *la memoria centrale*
- *i registri(dedicati e generali)*
- *lo stack*

- *indirizzi espliciti*
- *indirizzi impliciti*

2. **Le istruzioni di macchina** *per specificare al processore le operazioni da effettuare.*

codice operativo	**uno o più operandi (campi indirizzo)**

3. **I dati**

Componenti: - indirizzo
- valore
- tipo

Tipi di dati

- ***Dato:*** informazione codificata in modo da poter essere trattata dall'elaboratore nel modo specificato dal codice operativo
- ***Tipi di dati*** (***Data type***): insieme di valori e operazioni definite su essi
  - Es. Integer: set di interi con le operazioni +,-,*,:

    Stringa: sequenze di caratteri alfabetici con operazioni di concatenazione, selezione, ecc…;

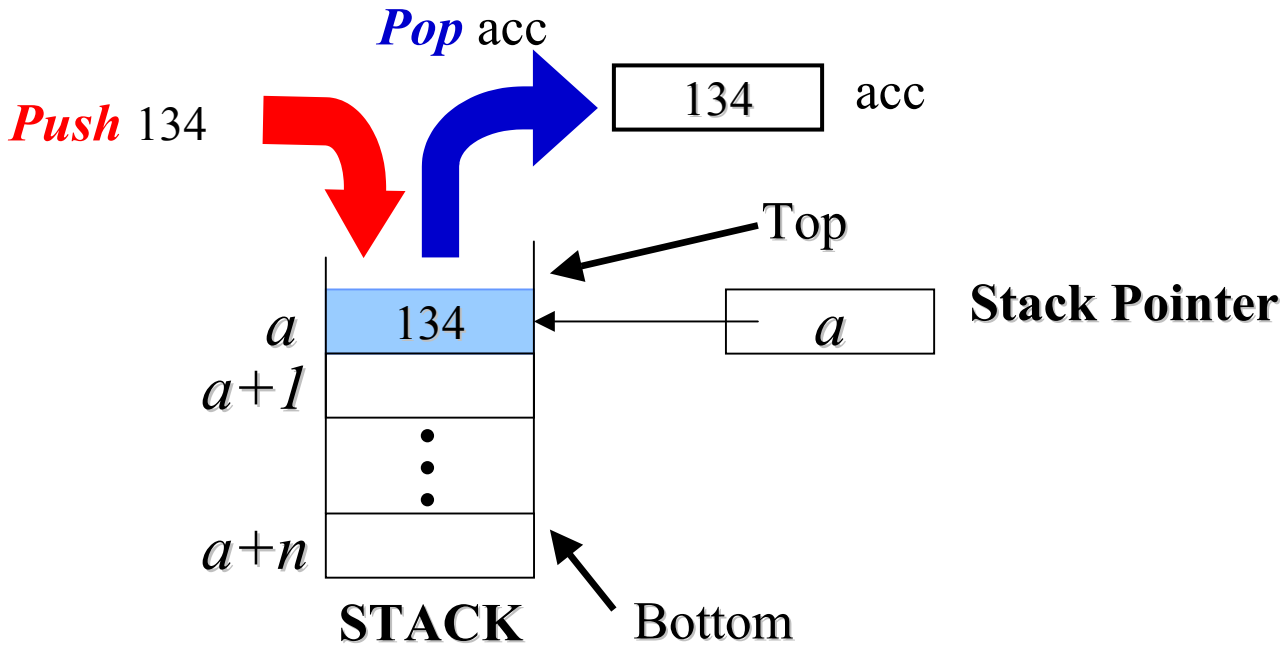
- Ad ogni livello dell'elaboratore corrisponde un insieme diverso di dati
  - Dati a livello di linguaggio macchina diversi dai dati a livelli superiori
  - Es. dato: indirizzo di memoria
    - Essenziale a livello di linguaggio macchina
    - Assente in linguaggi ad alto livello
  - Es. dato di tipo "record"
    - Assente in linguaggio macchina
    - Presente in linguaggio ad alto livello

Riferimento ai dati

- Il riferimento a un dato è il modo in cui nell'istruzione si specifica la struttura fisica che supporta il dato
  - ***Memoria***
    - ***Riferimento esplicito*** all'indirizzo di una locazione di memoria
  - ***Registro***
    - ***Riferimento esplicito*** al numero di registro
  - ***Stack***
    - ***Riferimento implicito*** ad un registro (stack pointer) che contiene l'address del top dello stack

Lo Stack

- ***Stack***: struttura dati con una organizzazione L.I.F.O. (Last In First Out) delle operazioni di inserimento (push) ed estrazione (pop) di un dato.



Set di registri

Tutte le operazioni sono eseguite sui dati contenuti nei registri interni al processore

- ***Registri generali*** : possono essere controllati dal programmatore
  - usati per conservare e manipolare dati (operandi e risultati)

- ***Registri dedicati*** : non possono essere usati dal programmatore
  - PC (Program Counter), IR (Instruction Register), MAR (Memory Address Register), PSW (Program Status Word)

L'architettura ed i Registri

Classificazione delle architetture in base al numero ed al tipo di *registri* in essa presenti

- **Architettura ad accumulatore**
- **Architettura a stack**

  Architettura che usa lo stack per conservare gli operandi ed il risultato di un'operazione

  Es. Addizione: C=A+B → Push A; Push B; Add; Pop C
- **Architettura a registri:**
  - **General purpose**

    Macchina con set di registri che possono essere usati come accumulatori, registri di indirizzo, stack, registri generali. Il riferimento ai registri contenenti gli operandi deve essere specificato nella istruzione

    Es. Addizione: C=A+B → Load R1, A; Load R2, B; Add R1,R2; Store C, R2
  - **Special purpose**

    Macchina con set di registri generali e dedicati le cui funzioni sono predefinite

Insieme delle istruzioni

- Parte visibile al programmatore e al progettista di compilatori
- Istruzione: parola del linguaggio macchina
- Insieme delle istruzioni: vocabolario del linguaggio macchina
- Formato di istruzione: sintassi della parola
- Caratteristiche del set di istruzioni:

Completezza

Il set di istruzioni deve includere tutte le operazioni di base per i tipi di dati presenti nell'ISA

Ortogonalità

Istruzioni non ridondanti: un solo modo per effettuare una istruzione

Compatibilità

Una famiglia di elaboratori deve poter utilizzare gli gli stessi programmi

IL FORMATO DI ISTRUZIONE

- Formati a lunghezza fissa:
  - Per ogni istruzione una parola sufficientemente lunga da ospitare tutti i formati
- Formati a lunghezza variabile:
  - Per ogni istruzione una parola di lunghezza variabile
- Formati ortogonali
  - Ogni istruzione contiene un campo “modo” per specificare le informazioni

Campo modo

IL FORMATO DI ISTRUZIONE (2)

a lunghezza fissa o a lunghezza variabile

Campi MODO per specificare il tipo di indirizzamento
Architettura TAGGED quando si usano campi separati per specificare il tipo di dato (B,W,…)

Quando più informazioni sono specificate in campi separati, si parla di formato di istruzione ortogonale. L'alternativa è di avere tanti codici operativi diversi (anche per una stessa operazione) per quanti sono i tipi di indirizzamento previsti.

Le istruzioni fanno uso di un numero intero di parole o di un loro sottomultiplo.

La lunghezza dell'istruzione dipende dal numero di operandi e la lunghezza del codice operativo è funzione del numero di istruzioni diverse previste.

Formato non ortogonale

- Si parla di formato non ortogonale quando per ogni operazione esistono diversi codici operativi a seconda del numero di operandi e della loro lunghezza

- Es. ADDB2 Op1, Op2: Op1← (Op1)+(Op2)

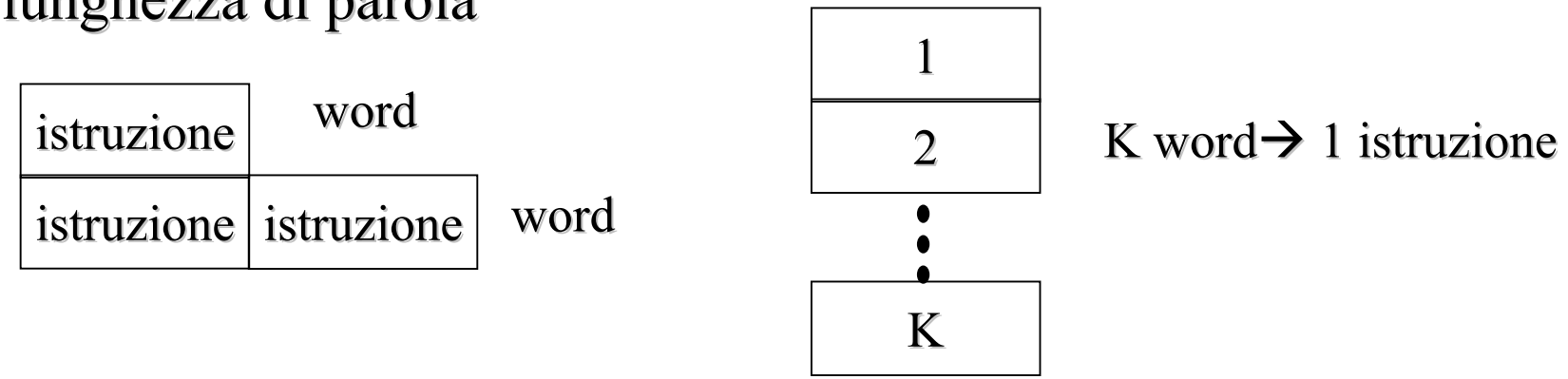
  ADDB3 Op1, Op2, Op3 Op3← (Op1)+(Op2)+(Op3)

  ADDW2 Op1, Op2 Op1← (Op1)+(Op2)

  B=Byte
  W=Word

Lunghezza di istruzione

- Diversi formati di istruzione → diverse lunghezze di istruzione
- ***Lunghezza di istruzione*** : multiplo o sottomultiplo della lunghezza di parola



Scelta architetturale:

- **Istruzioni brevi**
  - Minore occupazione di memoria
  - Maggiore velocità di trasferimento e quindi di esecuzione delle istruzioni
- **Istruzioni lunghe**
  - Minor numero di trasferimenti
  - Complessità maggiore per l'unità di controllo

IL FORMATO DI ISTRUZIONE (3)

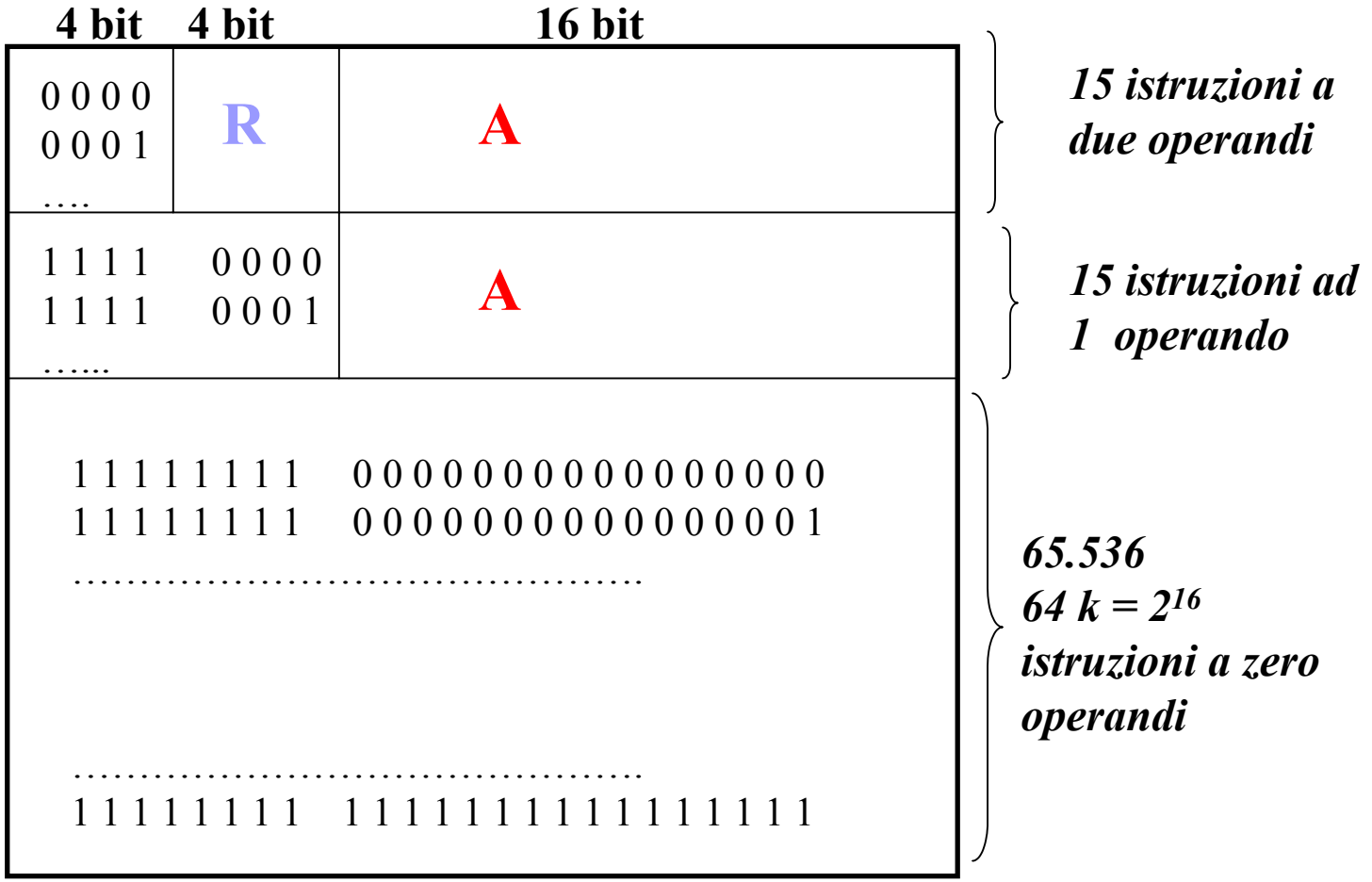
Codici Operativi a lunghezza variabile

Un codice operativo a *n* bit permette di definire 2^n istruzioni distinte

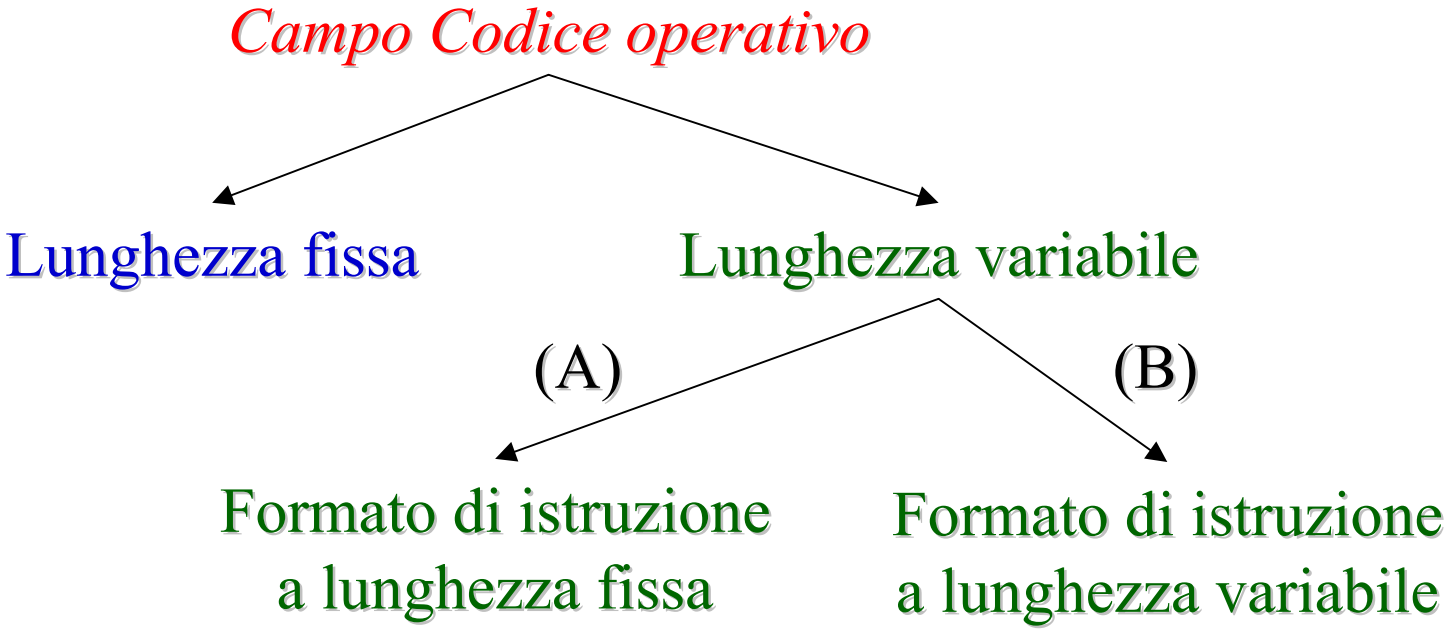
Esempio:

R = riferimento a (operando in) registro

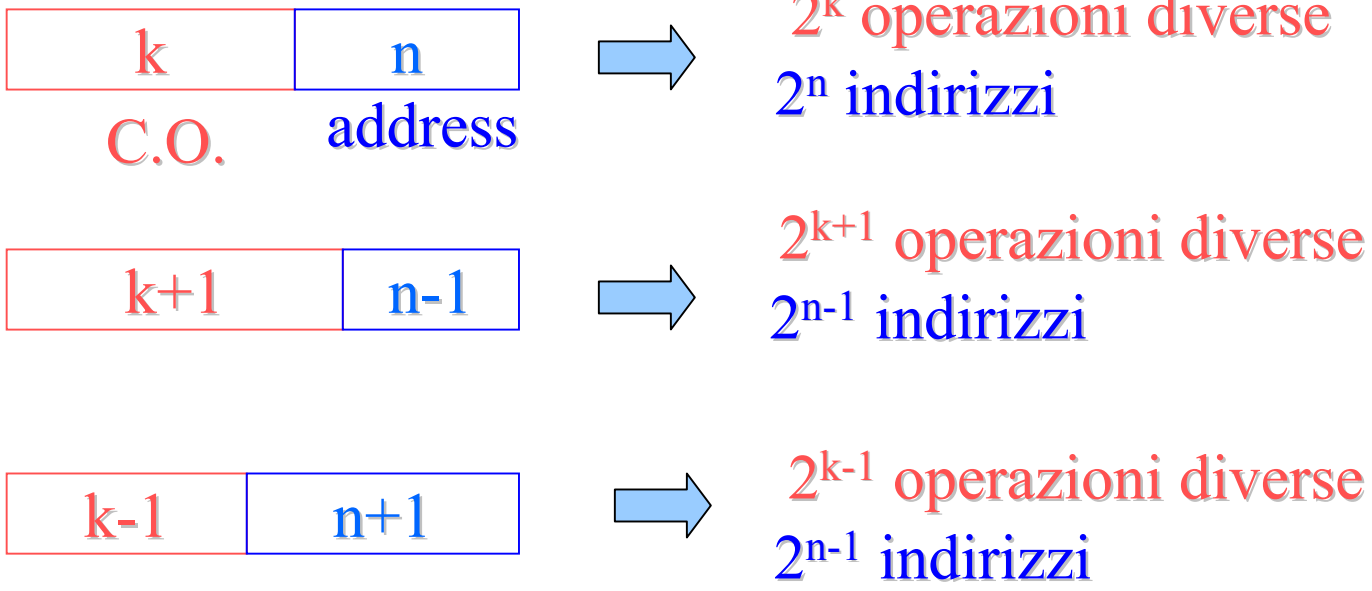
A= riferimento a (operando in memoria



Campo Codice operativo

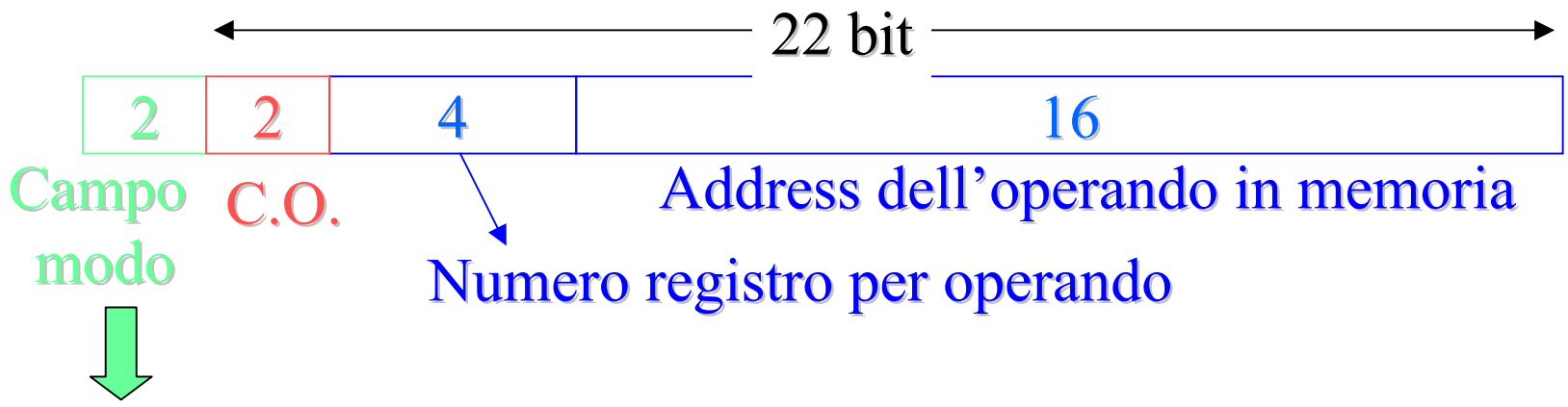


(A) Codice operativo a lunghezza variabile in formati di istruzione a lunghezza fissa



- Compromesso tra numero di istruzioni e numero di locazioni di memoria indirizzabili

Esempio



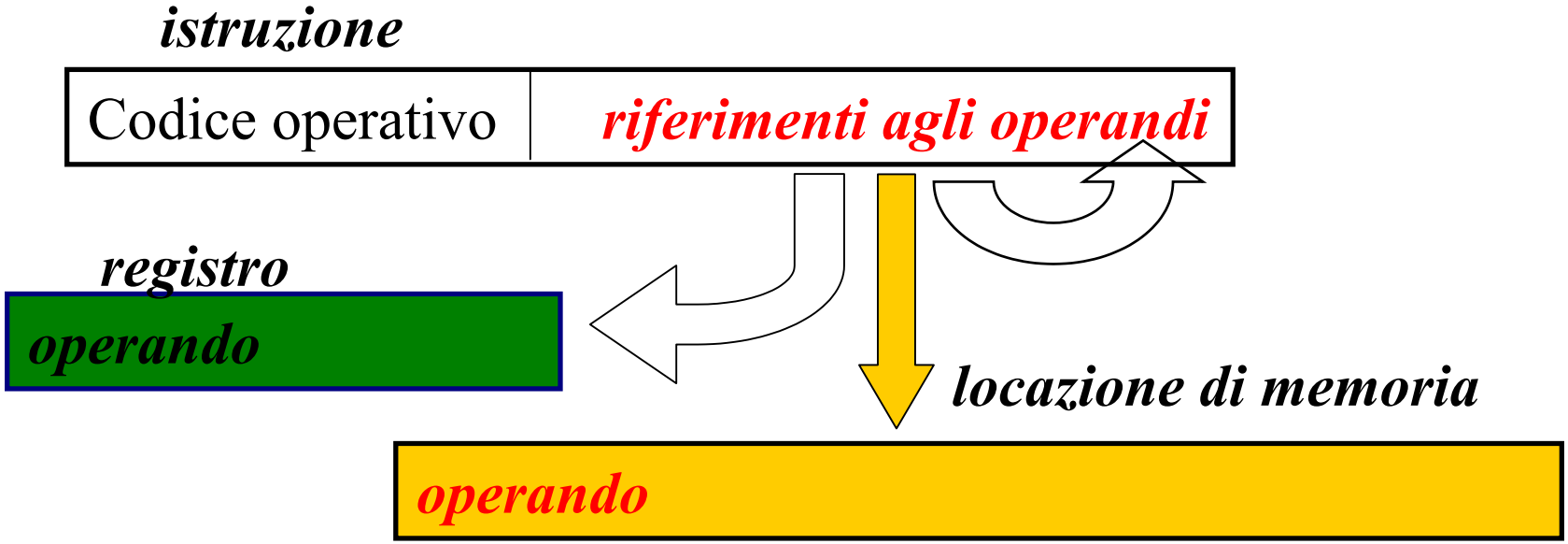
00 → C.O. a 22 bit→ 2^{22} istruzioni a zero operandi (0 address)

01 → C.O. a 16 bit→ 64 istruzioni a 1 operando (1 address)

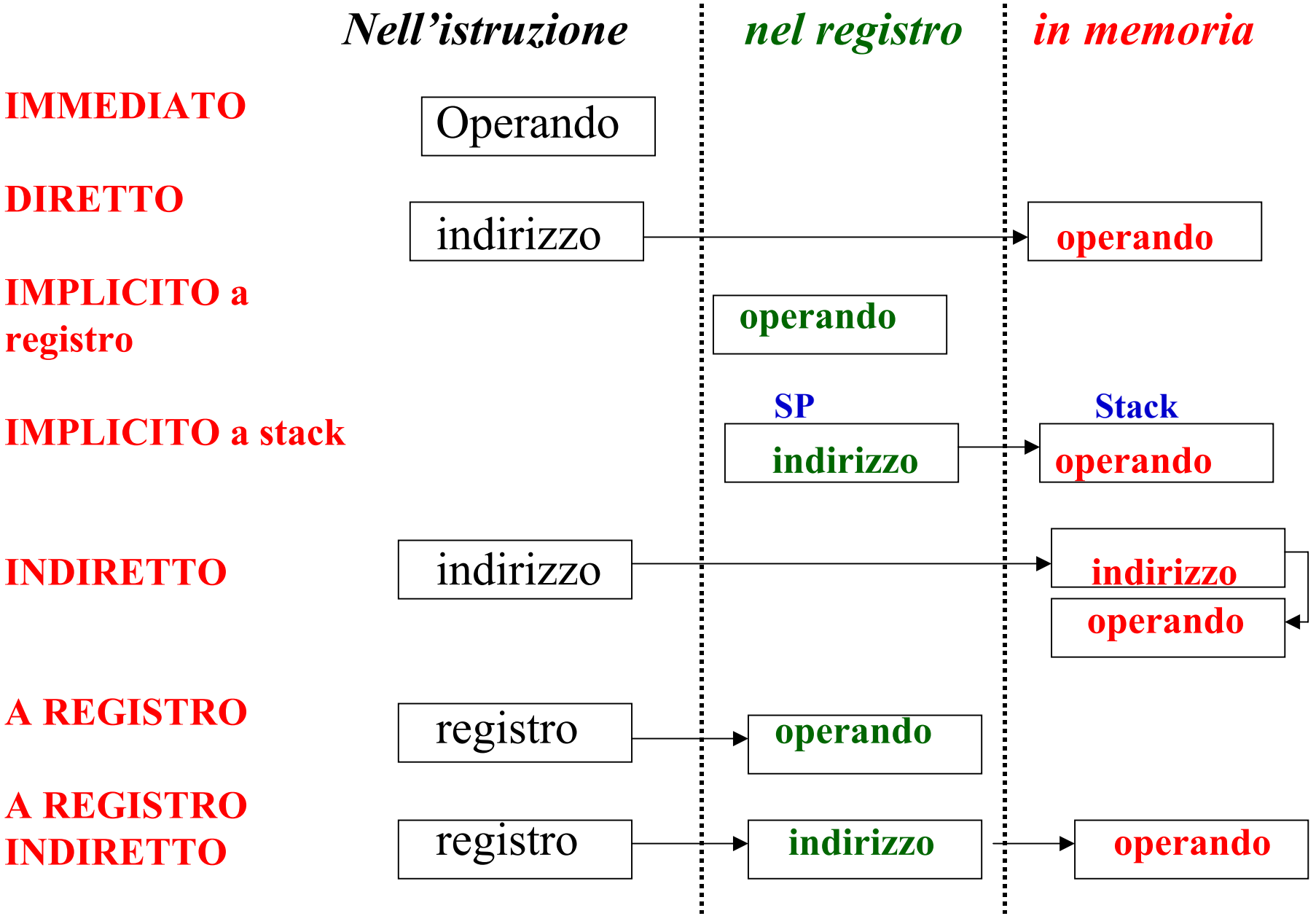
10 → C.O. a 2 bit→ 4 istruzioni a 2 operandi (2 address)

I METODI DI INDIRIZZAMENTO (1)

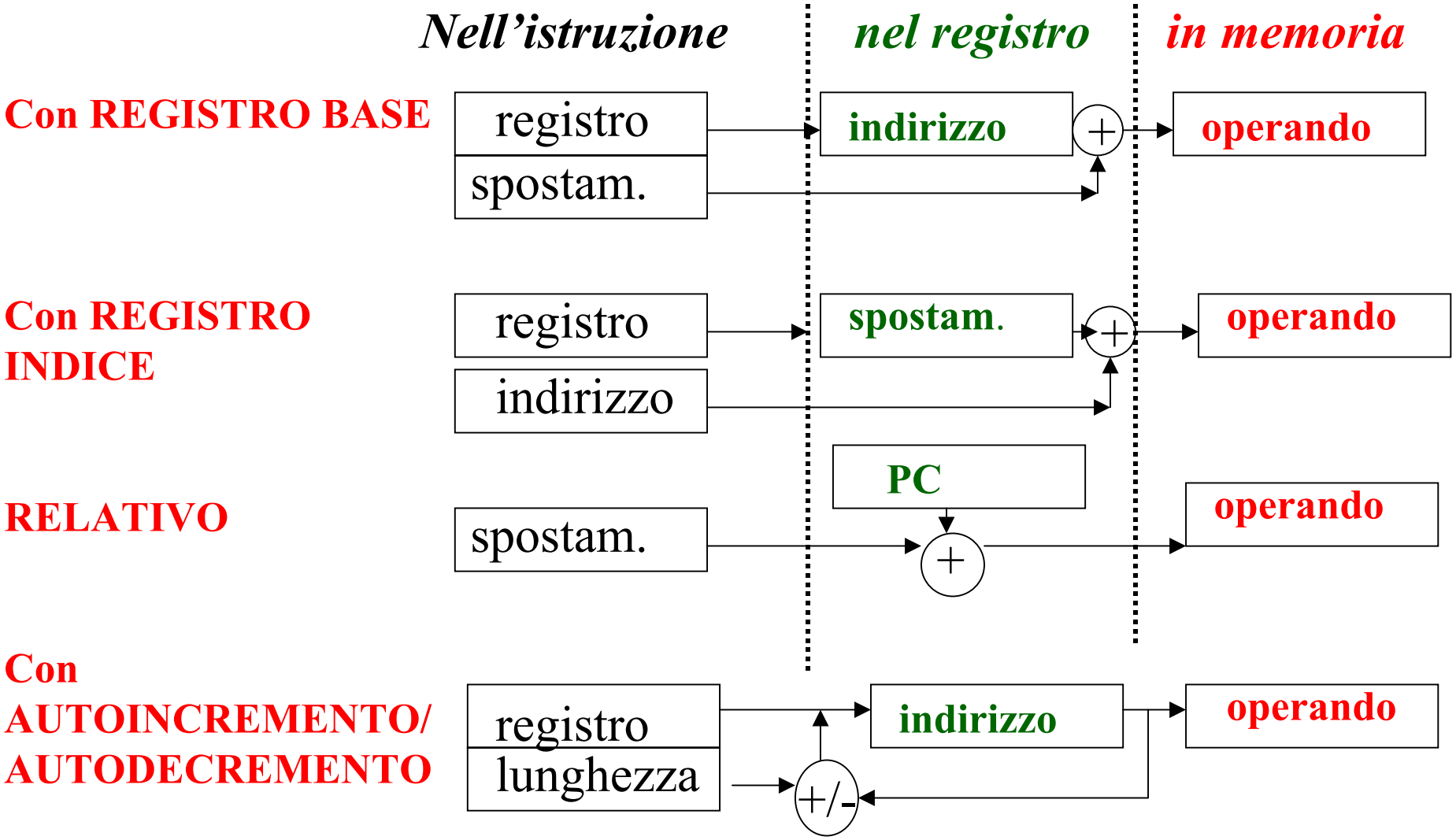
Rappresentano i modi con cui il processore interpreta l'informazione presente nell'istruzione per risalire agli operandi.



I METODI DI INDIRIZZAMENTO (2)



I METODI DI INDIRIZZAMENTO (3)



Indirizzamento immediato

- L'operando è collocato direttamente nel campo operandi dell'istruzione

  - VANTAGGIO: L'esecuzione della istruzione è immediata poiché non richiede accessi in memoria
  - SVANTAGGIO: il valore dell'operando è limitato dalla lunghezza del campo address

ADD Ax,3 (Ax ← Ax+3)

MOV Ah,5 (Ah ← 5)

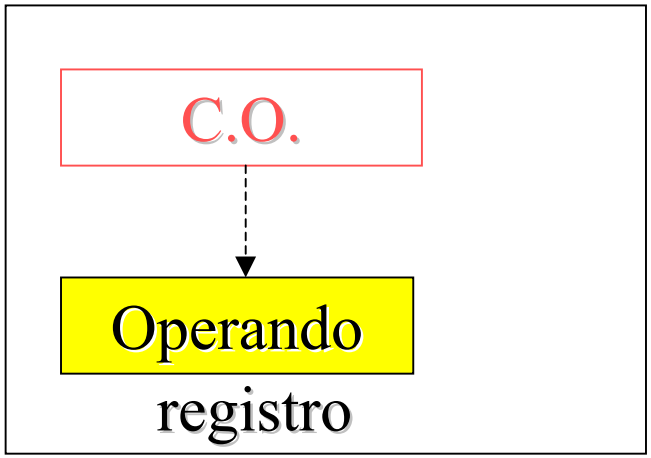
Indirizzamento implicito

- L'operando è contenuto in un particolare registro della CPU specificato IMPLICITAMENTE nel codice operativo dell'istruzione

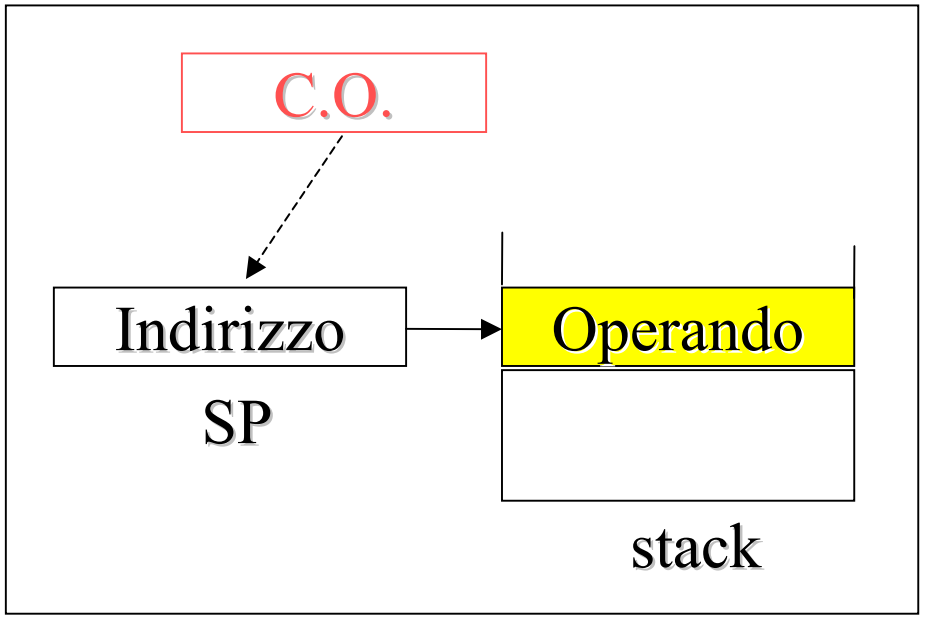
  - Program Counter → Indirizzo della successiva istruzione
  - Accumulatore → Operando
  - Stack pointer → Operando al top dello stack

Indirizzamento implicito

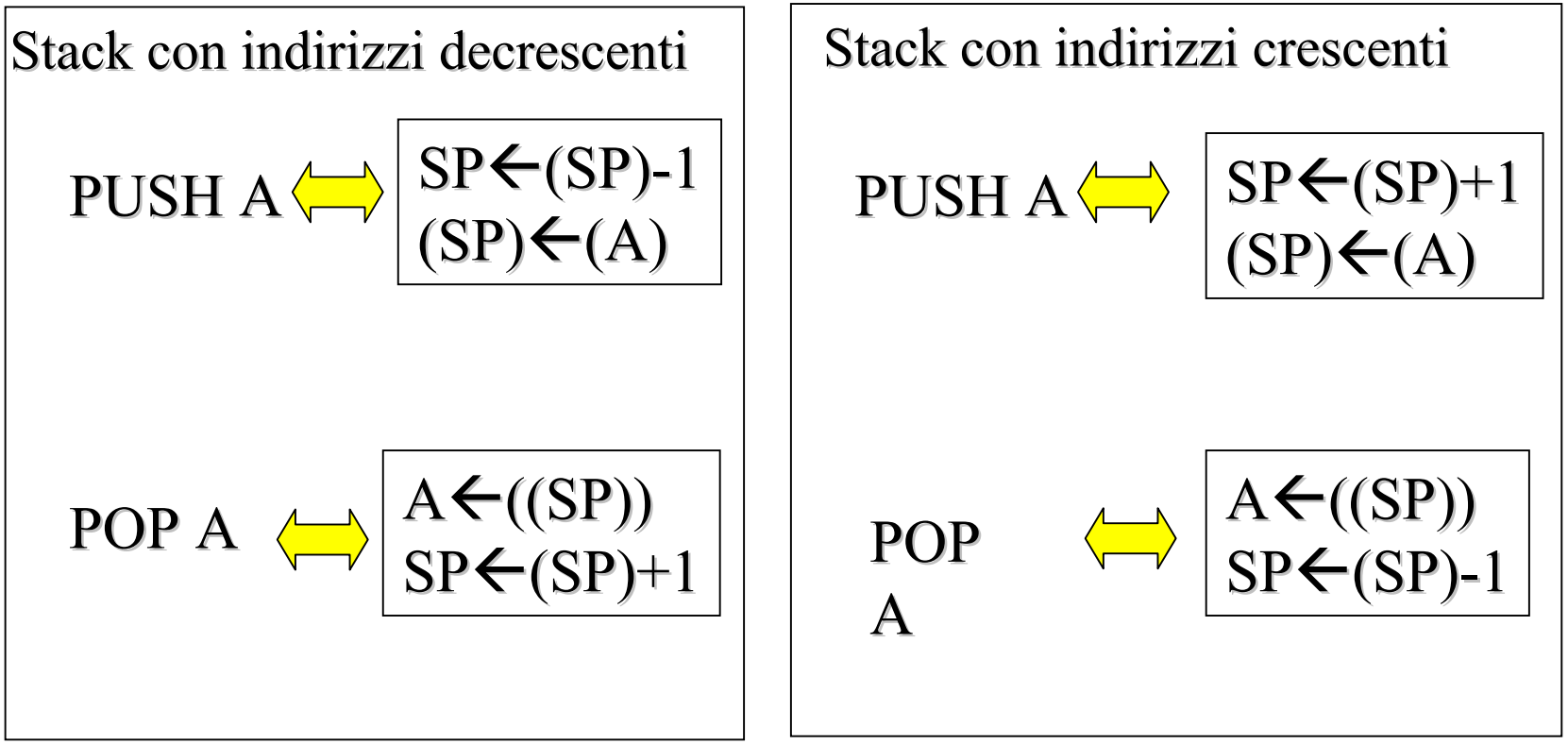
Indirizzamento implicito a registro



Indirizzamento implicito a stack



Istruzioni con indirizzamento implicito a stack



Indirizzamento implicito

- Traduzione del codice C=A+B

Indirizzamento implicito a registro (accumulatore)

```
Load A
Add B
Store C
```

Indirizzamento implicito a stack

```
Push A
Push B
Add
Pop C
```

Indirizzamento a registro

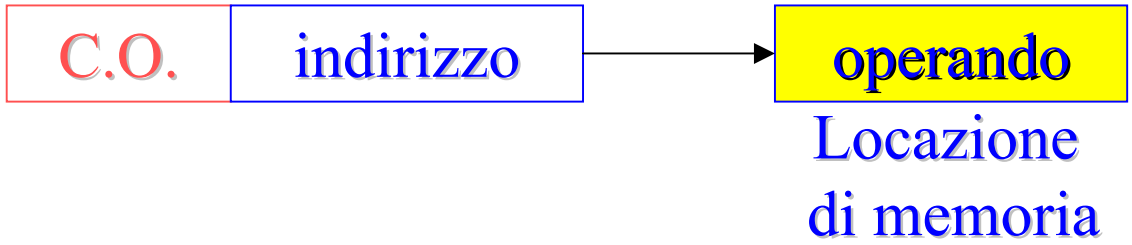
- L'operando è collocato in un registro specificato nel campo operandi dell'istruzione

  - VANTAGGIO: Accesso molto rapido ai registri
  - SVANTAGGIO: n° bit per l'address molto piccolo

Es. ADD R1, R2 (R1←R1+R2)

Indirizzamento diretto o assoluto

- L'operando è collocato in memoria principale all'indirizzo specificato nel campo address dell'istruzione

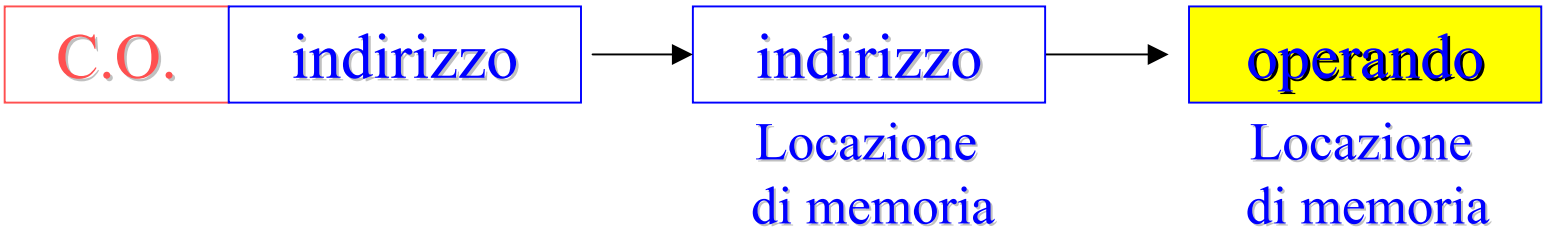


  - Utile per accedere a dati statici
  - Dispendioso per la rappresentazione degli indirizzi nel formato di istruzione

Es. ADD R1, (1001)

Indirizzamento indiretto

- L'operando è collocato in memoria centrale all'indirizzo specificato in un'altra locazione di memoria il cui indirizzo è specificato nel campo address dell'istruzione

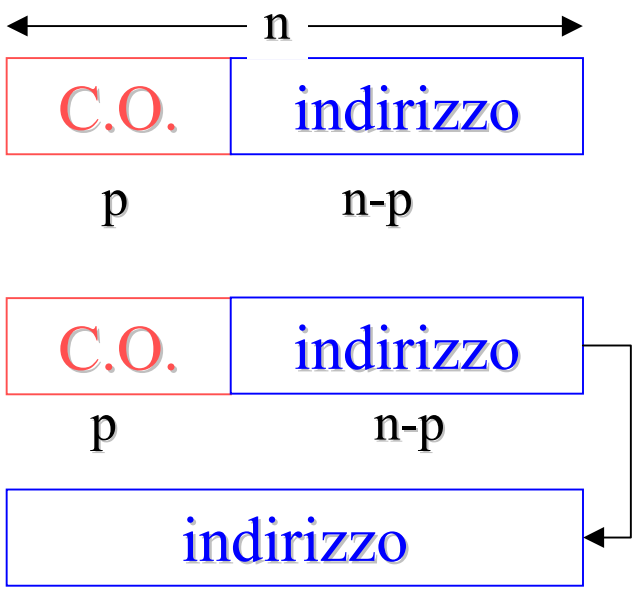


  - SVANTAGGIO: Occorrono due accessi alla memoria, uno per il puntatore e uno per l'operando
  - VANTAGGIO: Allargamento "spazio degli indirizzi" per architetture con istruzioni a lunghezza fissa

L'indirizzamento indiretto può avvenire tramite registro

Indirizzamento indiretto

- Allargamento spazio degli indirizzi per architetture con istruzioni a lunghezza fissa



Indirizzamento diretto

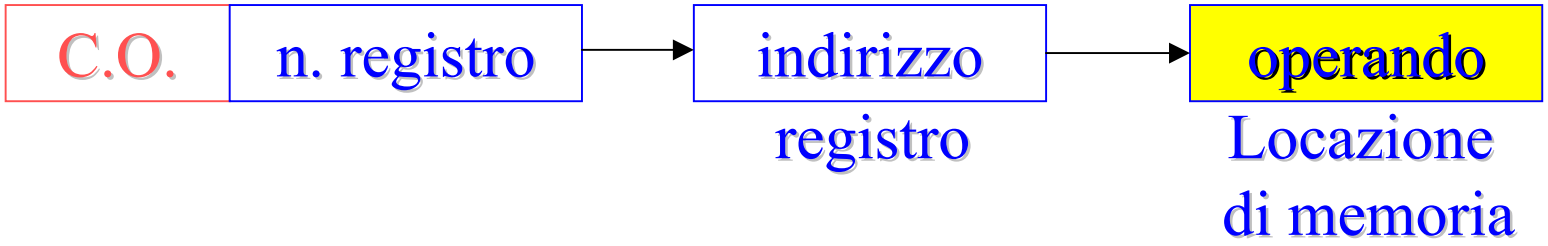
Spazio di indirizzi di 2^{n-p} locazioni

Indirizzamento indiretto

Spazio di indirizzi di 2^{n} locazioni

Indirizzamento a registro indiretto (o differito)

- L'operando è collocato in memoria principale all'indirizzo specificato nel registro riferito nel campo address dell'istruzione



Es. ADD R1, (R1)

Locazione di memoria il cui indirizzo è contenuto nel registro R1

- VANTAGGIO: istruzione con lunghezza piccola

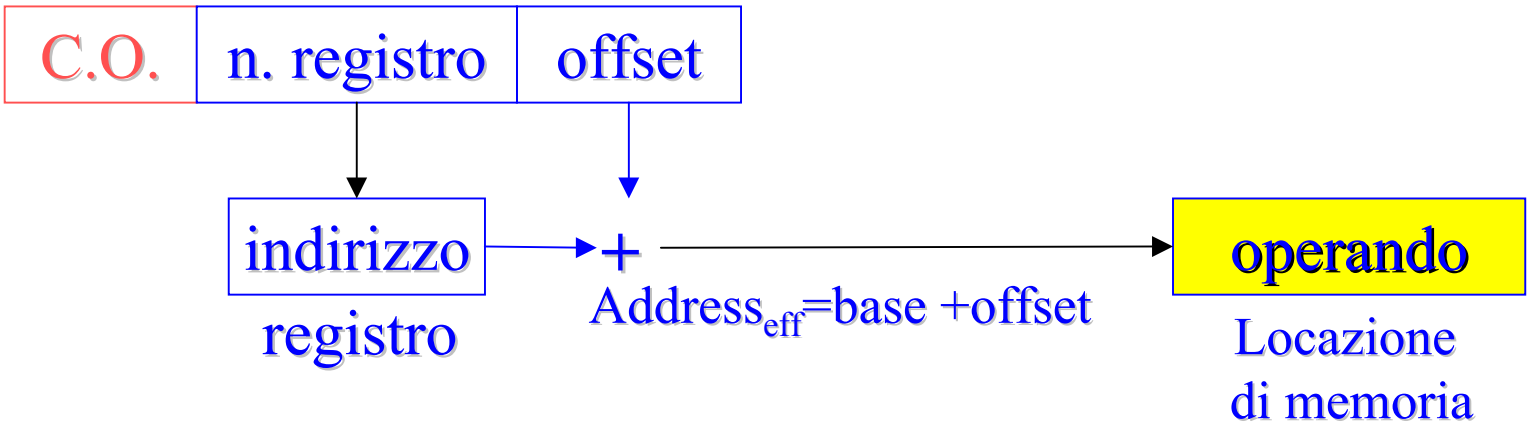
Indirizzamento indicizzato

- L'indirizzo "***effettivo***" dell'operando è calcolato sommando due valori:
  - Uno dei due valori è contenuto esplicitamente nell'istruzione
  - l'altro è contenuto in un registro

- Due tipi di indirizzamento indicizzato a seconda del contenuto del registro:
  - **Con registro base**
  - **Con registro indice**

Indirizzamento indicizzato con registro base

- Nel campo address dell'istruzione è presente il riferimento al registro che contiene un ***indirizzo*** e un valore detto ***spostamento*** (offset o displacement)



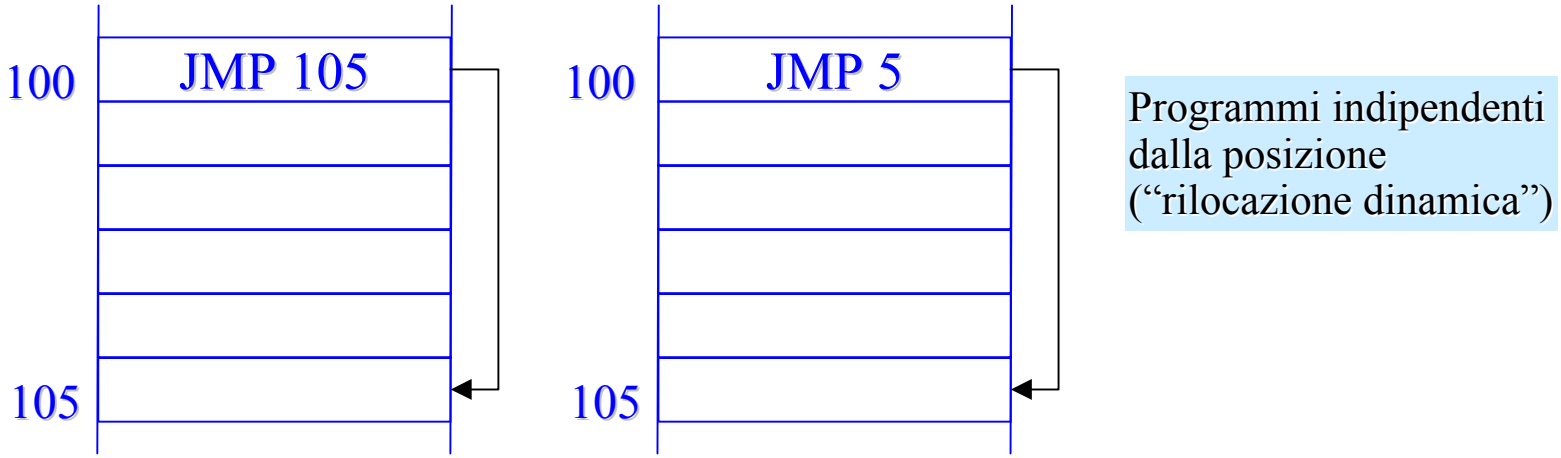
  - Vantaggio: usando più registri base si possono indirizzare aree di memoria ovunque dislocate

Indirizzamento indicizzato con registro base

- Es.: Famiglia Intel 80x86
  Indirizzamento con 4 registri base detti di "segmento", ognuno dei quali punta alla base di un segmento di memoria di 64K locazioni (16-bit address)
  - CS: segmento istruzioni (codice)
  - SS: segmento stack
  - SD: segmento dati
  - SE: segmento extra

Indirizzamento relativo

- Indirizzamento indicizzato con registro base che assume implicitamente come registro base il Program Counter

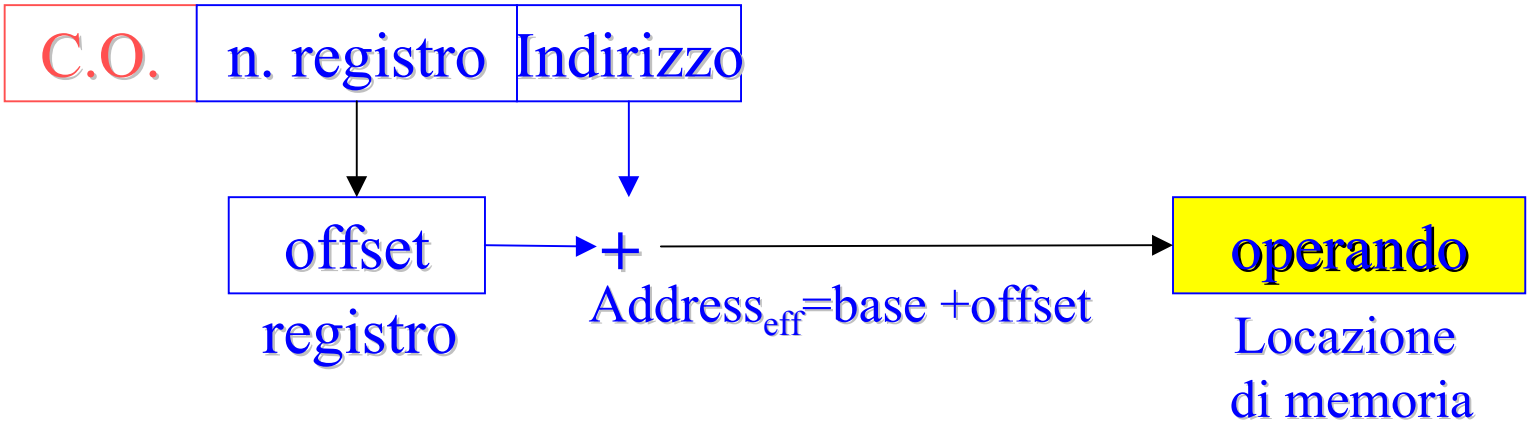


Programmi indipendenti dalla posizione ("rilocazione dinamica")

N.B.: Un programma può essere caricato in aree diverse senza aggiornare alcun registro (il PC assume automaticamente i valori relativi all'area specifica)

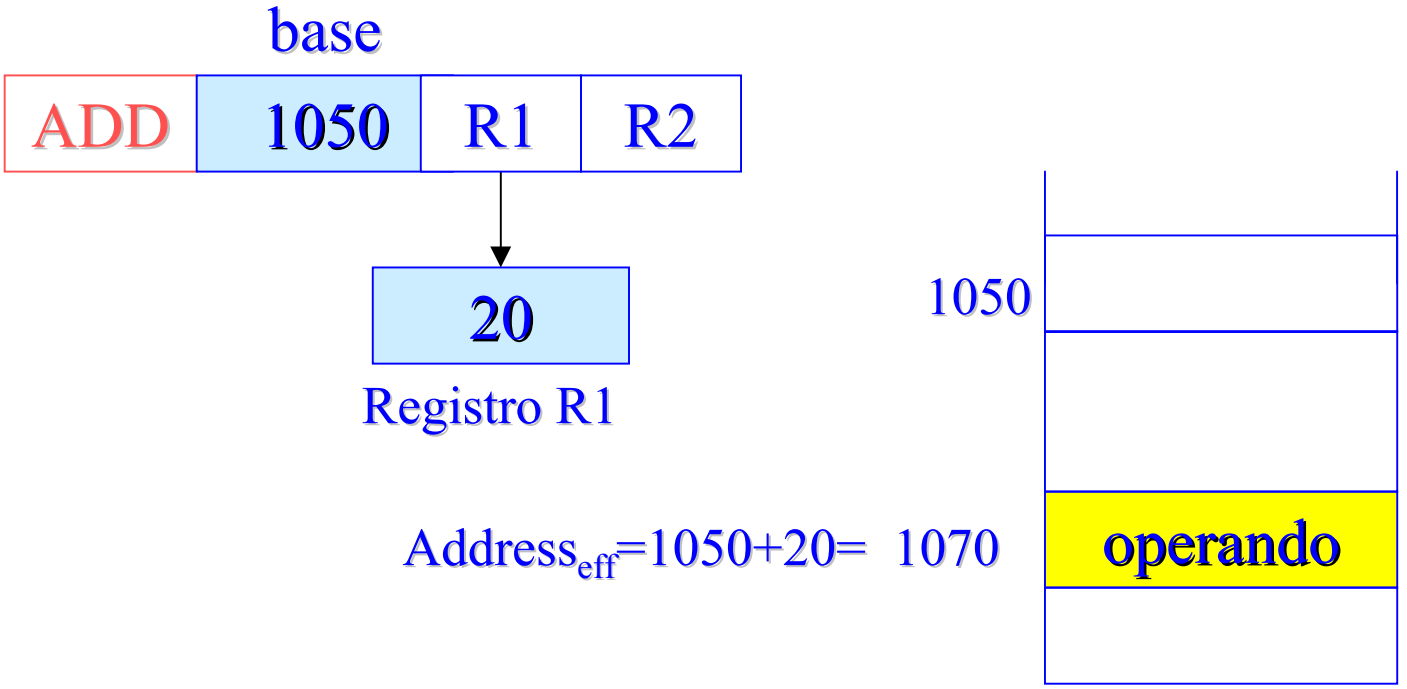
Indirizzamento indicizzato con registro indice

- Nel campo address dell'istruzione è presente il riferimento al registro che contiene lo ***spostamento*** ed un valore che rappresenta un ***indirizzo***



Indirizzamento indicizzato con registro indice

- Es. ADD 1050(R1), R2



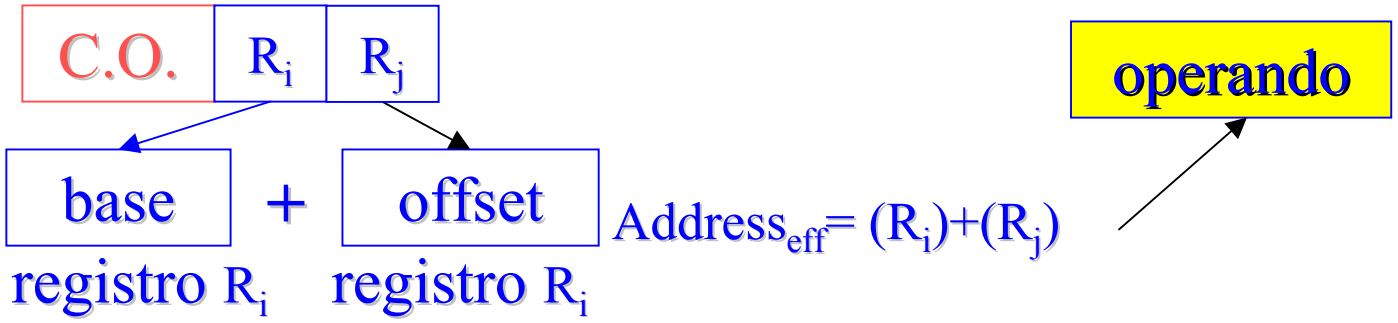
Indirizzamento indicizzato con registro indice

- Efficiente per operazioni su vettori
  - La base è l'indirizzo del 1° elemento del vettore
  - L'offset è l'indice dell'elemento del vettore
  - Es. Move A(R0), B(R0): Sposta il contenuto della locazione A+k nella locazione B+k, se k è il contenuto di R0

Con indirizzamento indicizzato	Con indirizzamento diretto o indiretto
Stessa istruzione per k variabile da 1 a N	N istruzioni, ossia una istruzione per ogni componente del vettore

Indirizzamento indicizzato

- Diverse varianti della modalità di indirizzamento indicizzato:
  - Il contenuto di un secondo registro può essere usato al posto della costante X

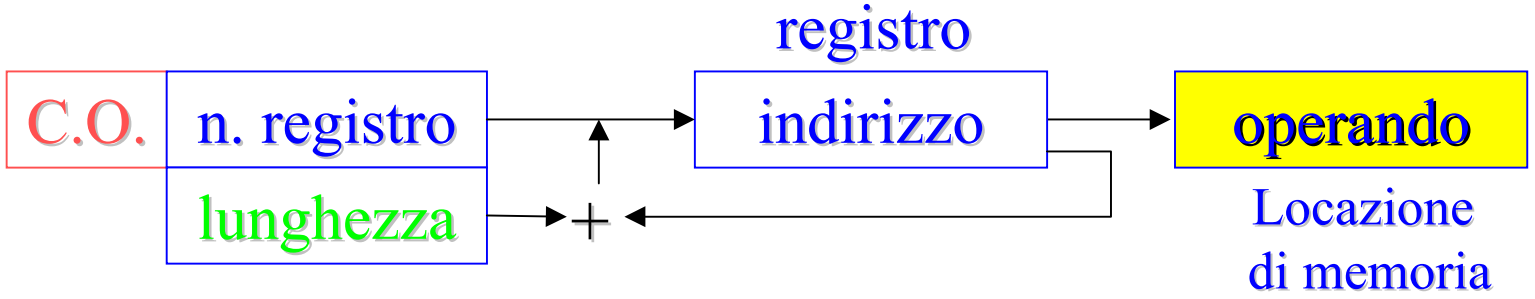


  - Due registri + costante

$$Address_{eff} = X+(R_i)+(R_j)$$

Indirizzamento con autoincremento

- Indirizzamento a registro indiretto + autoincremento automatico del contenuto del registro



- Il parametro lunghezza rappresenta la dimensione in word o byte dell'operando e può essere *esplicita* o *implicita* (nel codice operativo)

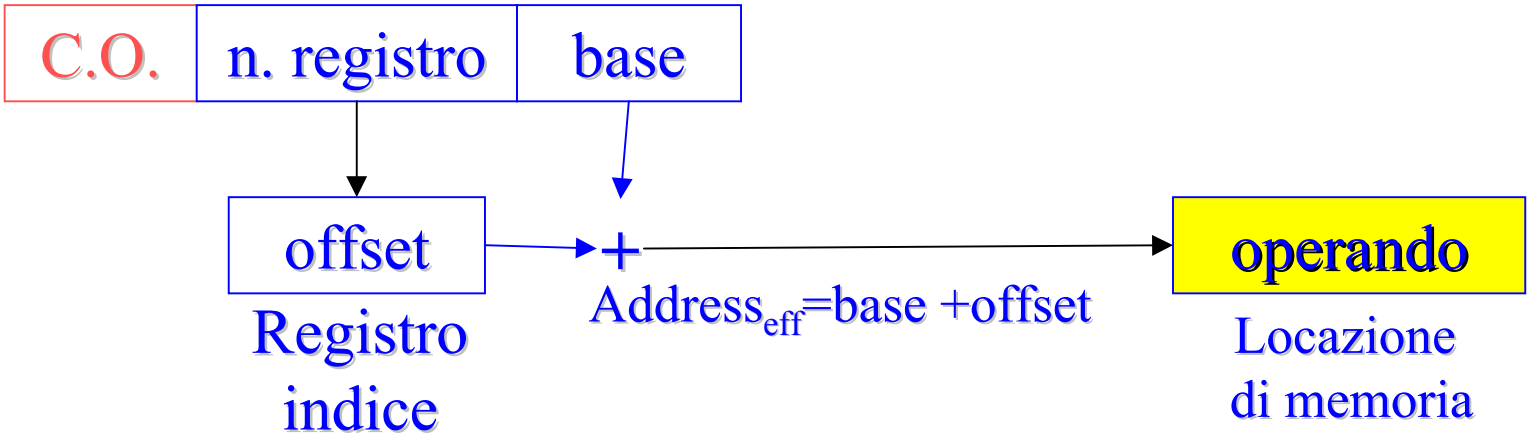
NB: Se il registro è lo SP …. Operazione di POP

Indirizzamento con autoincremento

- Esempio: Add R1, (R2)+
  - Addiziona il dato contenuto in R1 al dato contenuto nella locazione di memoria il cui indirizzo è in R2
  - Incrementa R2 di una quantità uguale alla lunghezza dell'operando
- Autoincremento: Incremento dell'address dopo l'utilizzo
  - Accesso sequenziale agli elementi di un vettore o ai caratteri di una stringa
  - Accesso agli elementi di una stringa

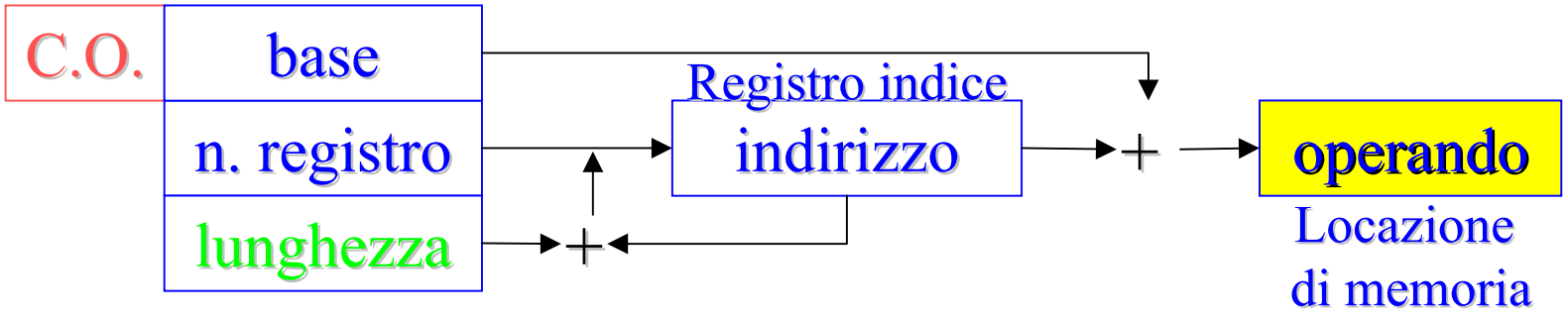
Indirizzamento con autoincremento

- Registro=registro indice
  - Incremento automatico del contenuto del registro per puntare all'elemento successivo dell'array



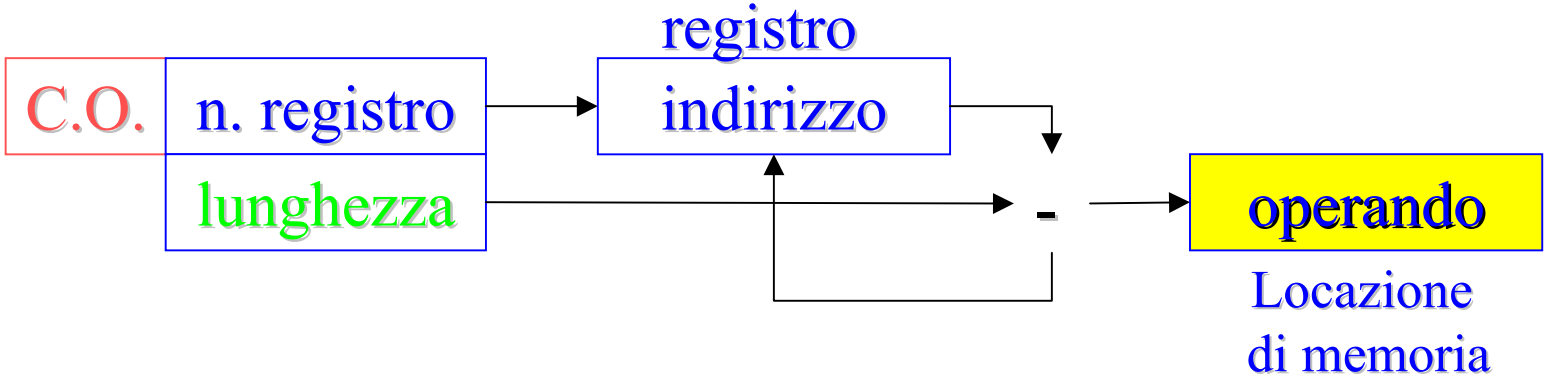
Indirizzamento con autoincremento

- Registro=registro indice
  - Incremento automatico del contenuto del registro per puntare all'elemento successivo dell'array



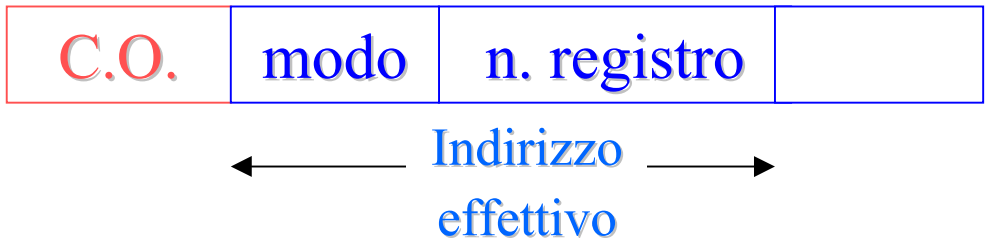
Indirizzamento con autodecremento

- Decremento dell'address prima dell'utilizzo
  - Facilità di manipolazione dello stack



Indirizzamento mediante modo e registro

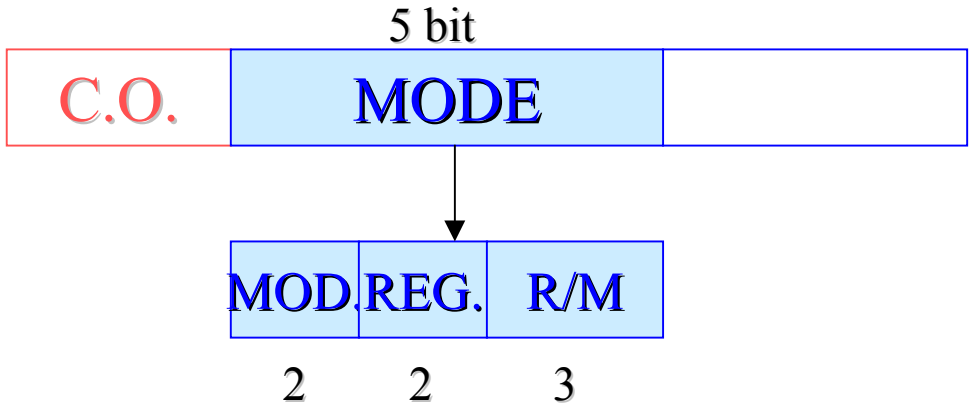
- Metodo che specifica il tipo di indirizzamento utilizzando sempre l'informazione contenuta nel campo 'modo' e in un registro



- Es. PC IBM, VAX Digital

Indirizzamento mediante modo e registro

- Processori 80x86



- Il campo MODE (5 bit) specifica la modalità di indirizzamento (25 modalità possibili)
  - Un operando è specificato dal campo MOD di 2 bit e dal campo R/M di 3 bit
  - L'altro operando è sempre in un registro specificato dal valore del campo REG