Indirizzamento

Corso di Calcolatori Elettronici I

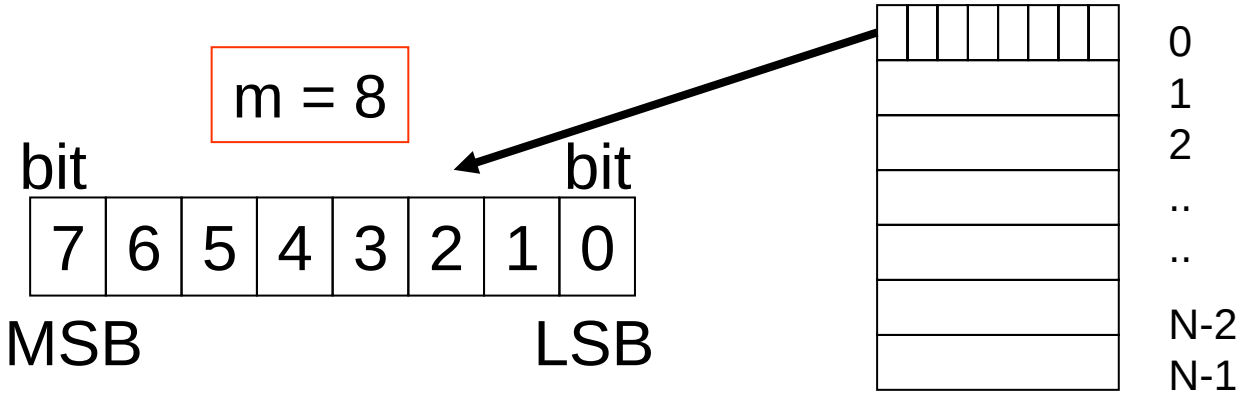
Dipartimento di Informatica e Sistemistica
Università degli Studi di Napoli “Federico II”

Sommario

- Concetti preliminari
- Modi di indirizzamento fondamentali
- Altri modi di indirizzamento (cenni)

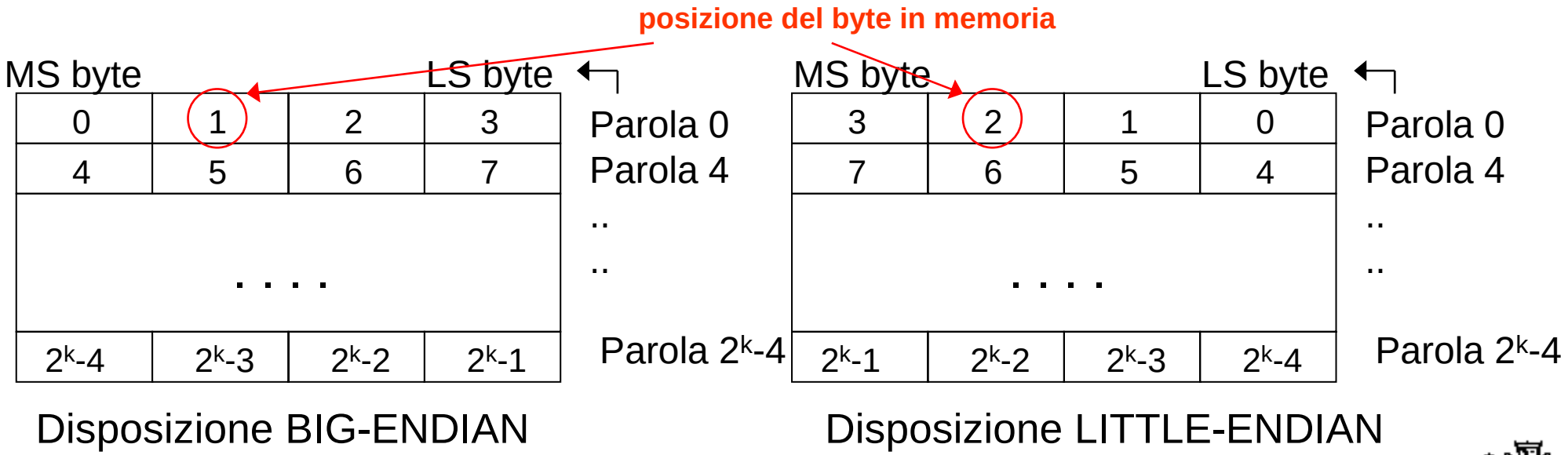
Memoria centrale

- La memoria centrale di un computer è organizzata come un array di stringhe di bit di lunghezza m, dette *parole* o *word* (m = LUNGHEZZA DI PAROLA)
- Gli m bit di una parola sono accessibili dal processore (in lettura/scrittura) mediante un'unica operazione
- Ogni parola è individuata da un *indirizzo*, cioè un intero compreso tra 0 e N-1 (SPAZIO DI INDIRIZZAMENTO), con $N = 2^c$



Organizzazione della memoria

- I processori “a parola” accedono alla memoria con un parallelismo di 16 bit, 32 bit o 64 bit (larghezza della parola)
- Tipicamente, la memoria è sempre *byte-addressable*, cioè la più piccola unità di memoria indirizzabile è il byte (*locazione*), anche quando la larghezza della parola è maggiore (2, 4 oppure 8 byte)
- I byte costituenti una word possono essere disposti in due modi alternativi: big endian e little endian



Dimensioni degli indirizzi e dei registri

- Address size:
  - Numero di bit che compongono un indirizzo di memoria
- Register size:
  - Numero di bit che compongono un registro
- Non è detto che le due dimensioni coincidano. Tipicamente, la seconda è uguale o maggiore della prima
- Non è detto che tutti i bit del registro vengano fisicamente connessi con la memoria. Ciò implica che:
  - Lo spazio di indirizzamento logico è in generale diverso dallo spazio di indirizzamento fisico
  - Possono sorgere problemi di “aliasing”

Dimensioni degli indirizzi e dei registri

In questo esempio, il registro indirizzi presente all'interno del processore ha 16 bit, ma la memoria esterna ha solo 2^{12} locazioni fisiche, quindo solo 12 bit del registro indirizzi sono collegate al bus indirizzi della memoria

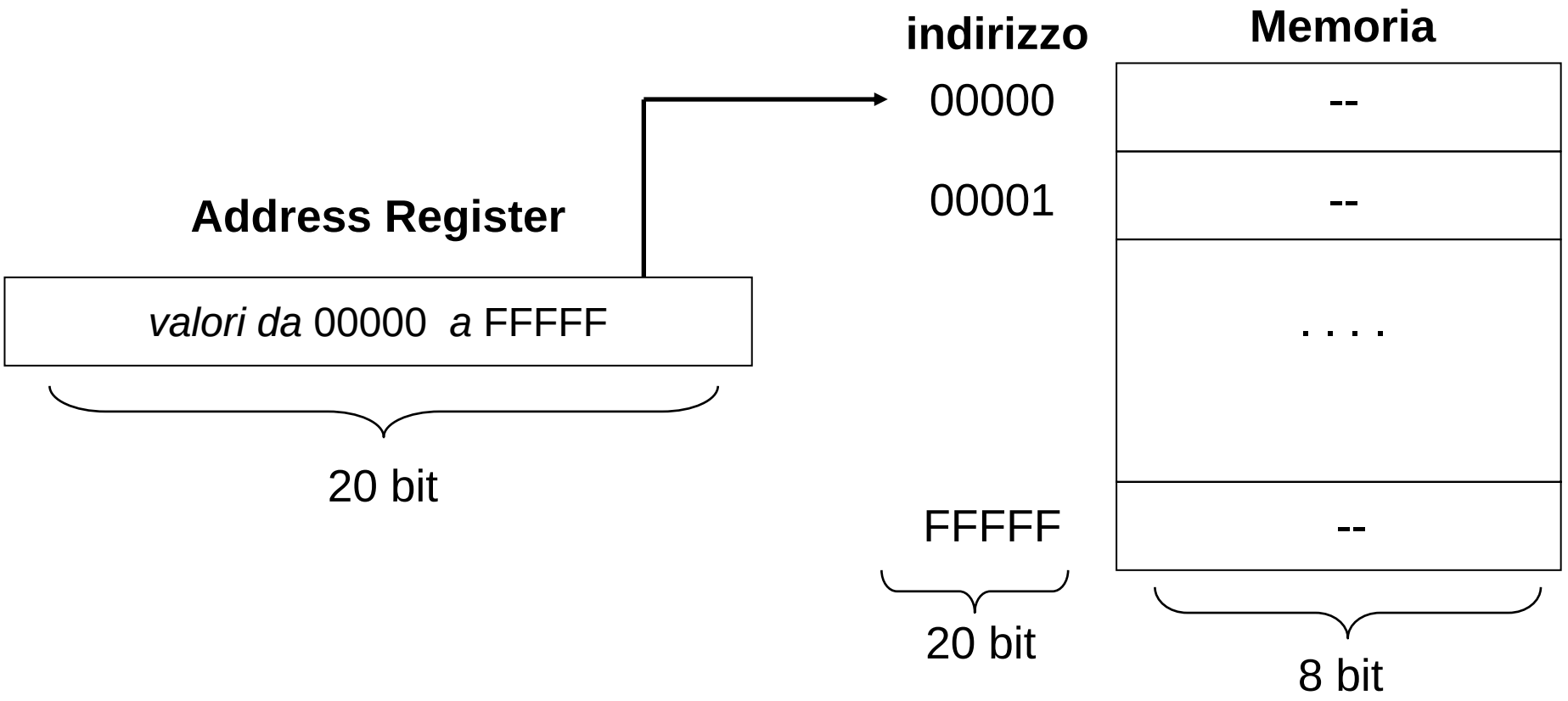
Diversi valori del registro indirizzi possono attivare la stessa locazione di memoria. Ad es.: $A3B2 e $93B2, poiché differiscono solo per i 4 bit più significativi

Questo fenomeno prende il nome di **aliasing**

Esercizio

- Tracciare lo schema di un'architettura di memoria con le seguenti caratteristiche:
  - Spazio di indirizzamento logico: 1MB
  - Spazio di indirizzamento fisico: 1MB
  - Ampiezza di parola: 1 byte
  - Granularità dell'accesso: byte addressable

Soluzione

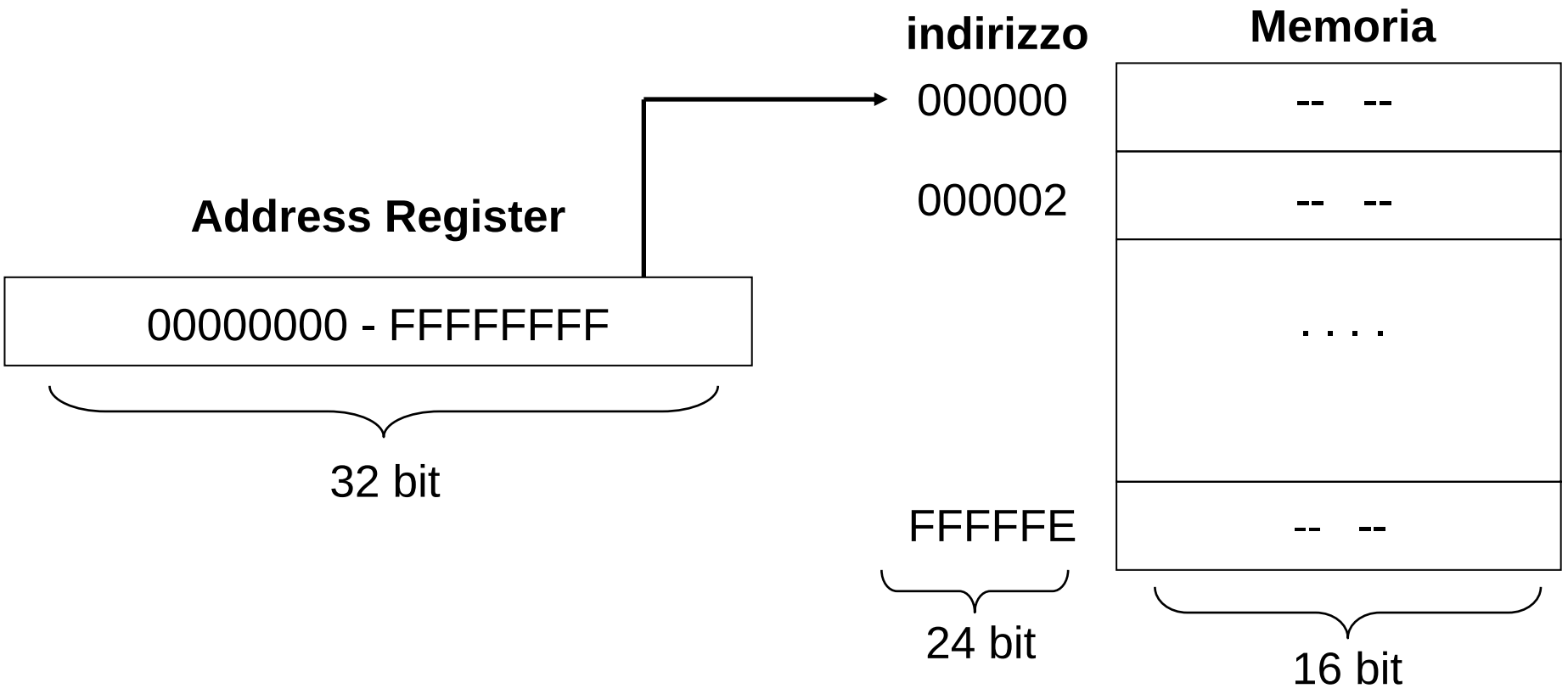


- Nota: Il processore MC68008 è organizzato in maniera simile

Esercizio

- Tracciare lo schema di un'architettura di memoria con le seguenti caratteristiche:
  - Spazio di indirizzamento logico: 4GB
  - Spazio di indirizzamento fisico: 16MB
  - Ampiezza di parola: 2 byte
  - Granularità dell'accesso: byte addressable

Soluzione

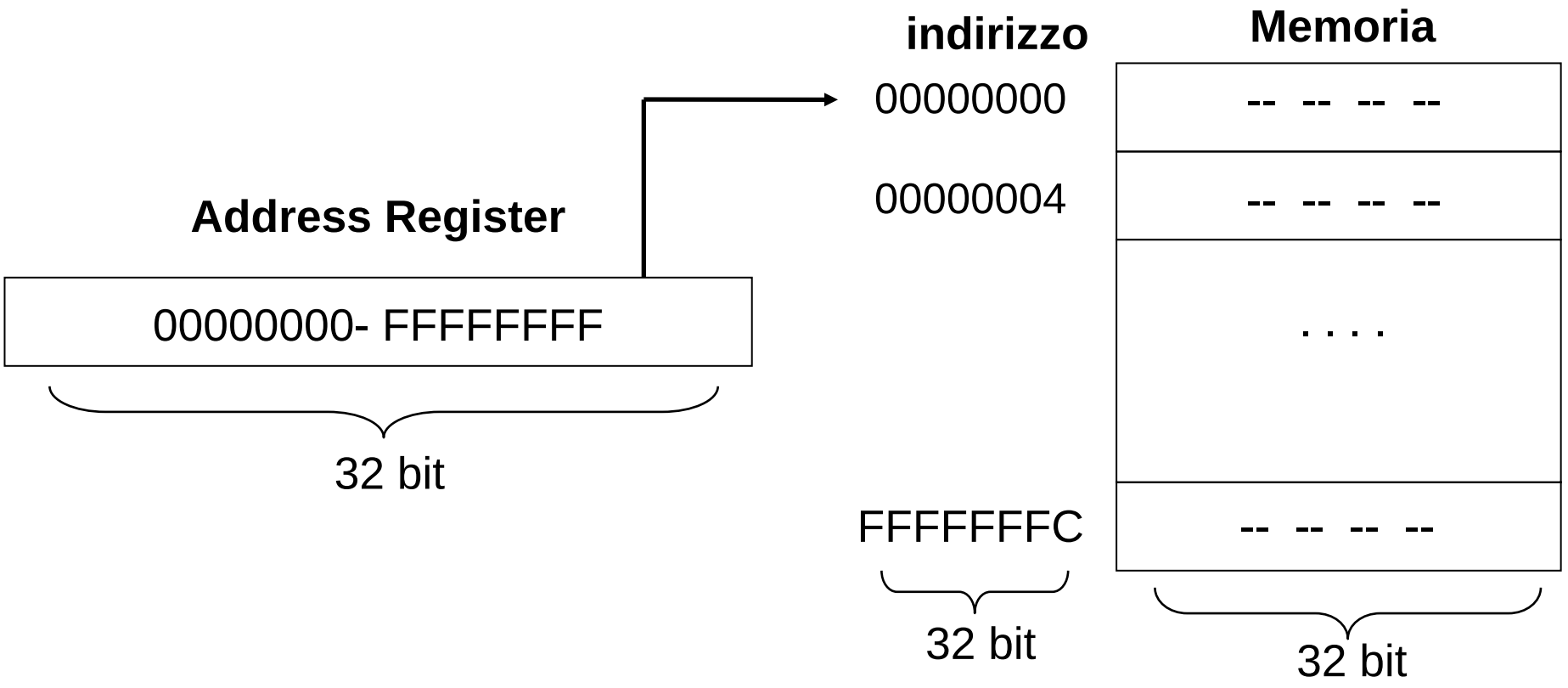


- Nota: I processori MC68000 e MC68010 sono organizzati in maniera simile

Esercizio

- Tracciare lo schema di un'architettura di memoria con le seguenti caratteristiche:
  - Spazio di indirizzamento logico: 4GB
  - Spazio di indirizzamento fisico: 4GB
  - Ampiezza di parola: 4 byte
  - Granularità dell'accesso: byte addressable

Soluzione



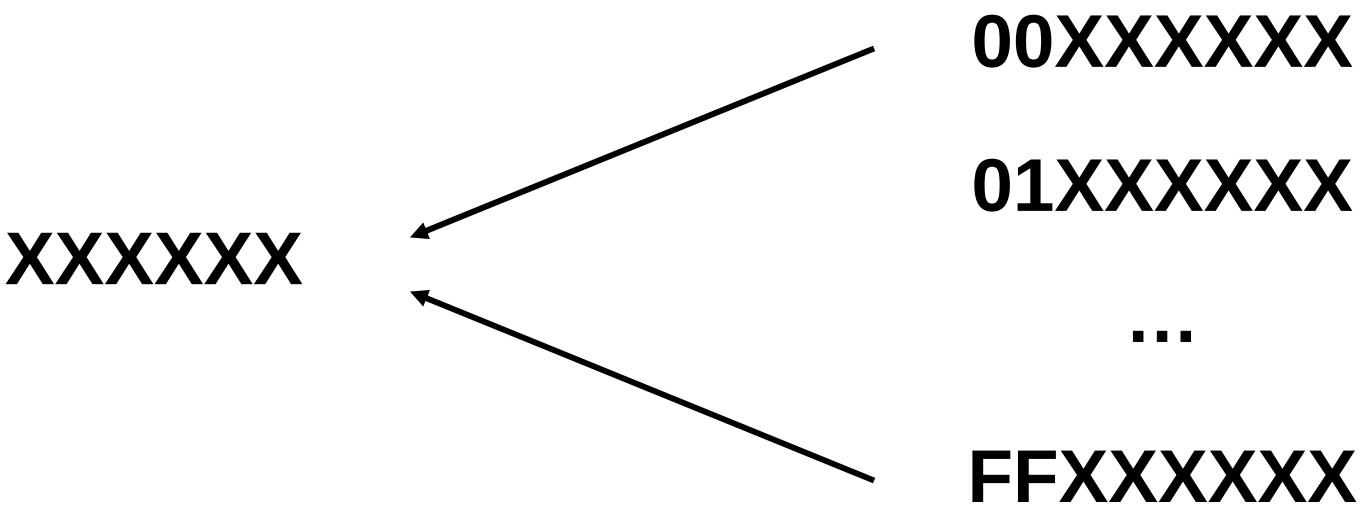
- Nota: I processori MC68020 e successivi sono organizzati in maniera simile

Esercizio

- Il processore MC68000 ha la seguente architettura di memoria:
  - Spazio di indirizzamento logico: 4GB (32 bit)
  - Spazio di indirizzamento fisico: 16MB (24 bit)
- Il processore MC68020 ha la seguente architettura di memoria:
  - Spazio di indirizzamento logico: 4GB (32 bit)
  - Spazio di indirizzamento fisico: 4GB (32 bit)
- Individuare le regioni di aliasing tra i due processori

Soluzione

- Esistono, per ogni indirizzo del processore MC68000, 256 indirizzi distinti del processore MC68020.
- Le regioni di aliasing sono individuate dalla corrispondenza:



Esercizio

- Supponendo di estendere un indirizzo di 16 bit con il MSB, individuare la regione dello spazio di indirizzamento a 32 bit acceduta

Soluzione

- Gli indirizzi tra 0000 e 7FFE vengono mappati sui primi 32KB dello spazio di 4GB
- Gli indirizzi tra 8000 e FFFE vengono mappati sugli ultimi 32KB dello spazio di 4GB

	0000
0000000000000000	**0000000000000000**
	7FFE
0000000000000000	**0111111111111110**
	8000
1111111111111111	**1000000000000000**
	FFFE
1111111111111111	**1111111111111110**

Modi di indirizzamento

- Indicano come la CPU accede agli operandi usati dalle proprie istruzioni
- La loro funzione è quella di fornire un indirizzo effettivo (EA) per l'operando di un'istruzione
  - Es: In un'istruzione per la manipolazione di un dato, l'indirizzo effettivo è l'indirizzo del dato da manipolare
  - Es: In un'istruzione di salto, l'indirizzo effettivo è l'indirizzo dell'istruzione a cui saltare
- Sono possibili moltissimi modi di indirizzamento. Nessun processore li supporta tutti, ma il 68000 ne supporta una buona parte

Modi di Indirizzamento del 68K

- Register Direct
  - Data-register Direct
  - Address-register Direct
- Immediate (or Literal)
- Absolute
  - Short
  - Long
- Address-register Indirect
- Auto-Increment
- Auto-Decrement

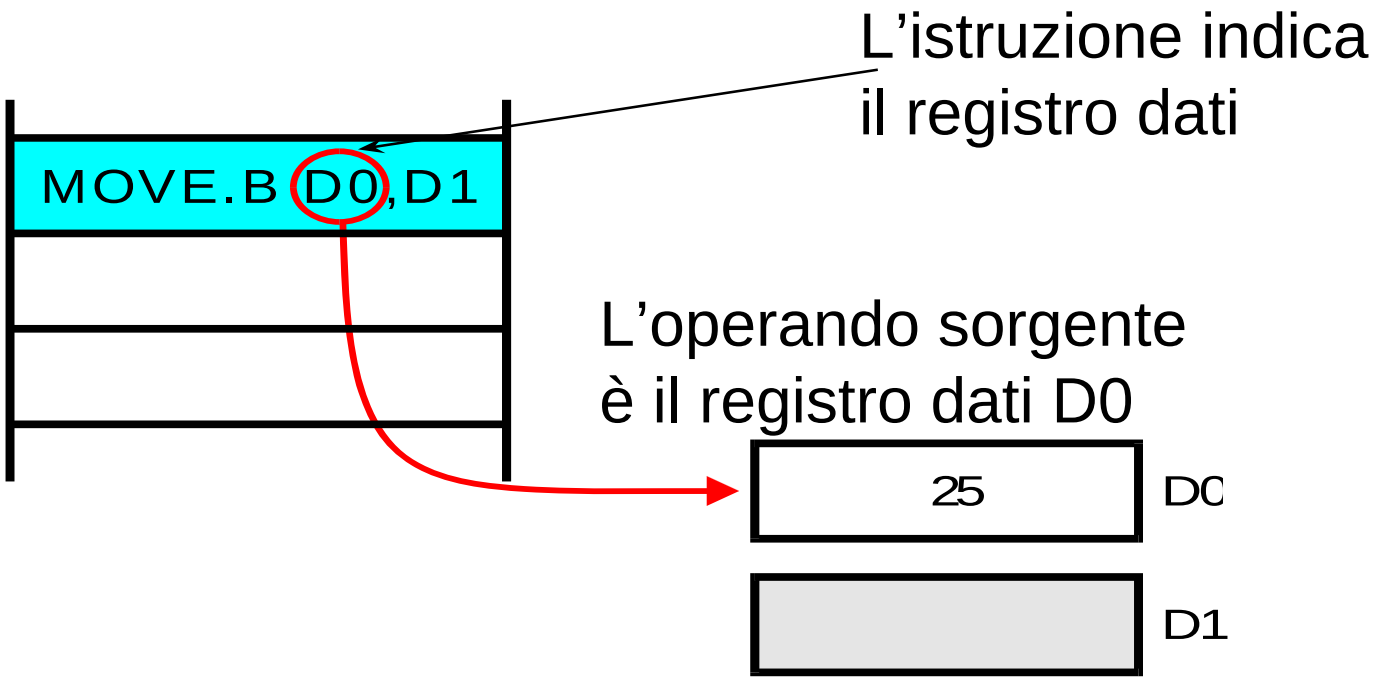
- Indexed short
- Based
- Based Indexed
  - Short
  - Long
- Relative
- Relative Indexed
  - Short
  - Long

Register Direct Addressing

- È il modo di indirizzamento più semplice
- La sorgente o la destinazione di un operando è un registro dati o un registro indirizzi
- Se il registro è un operando sorgente, il contenuto del registro specificato fornisce l'operando sorgente
- Se il registro è un operando destinazione, esso viene caricato con il valore specificato dall'istruzione

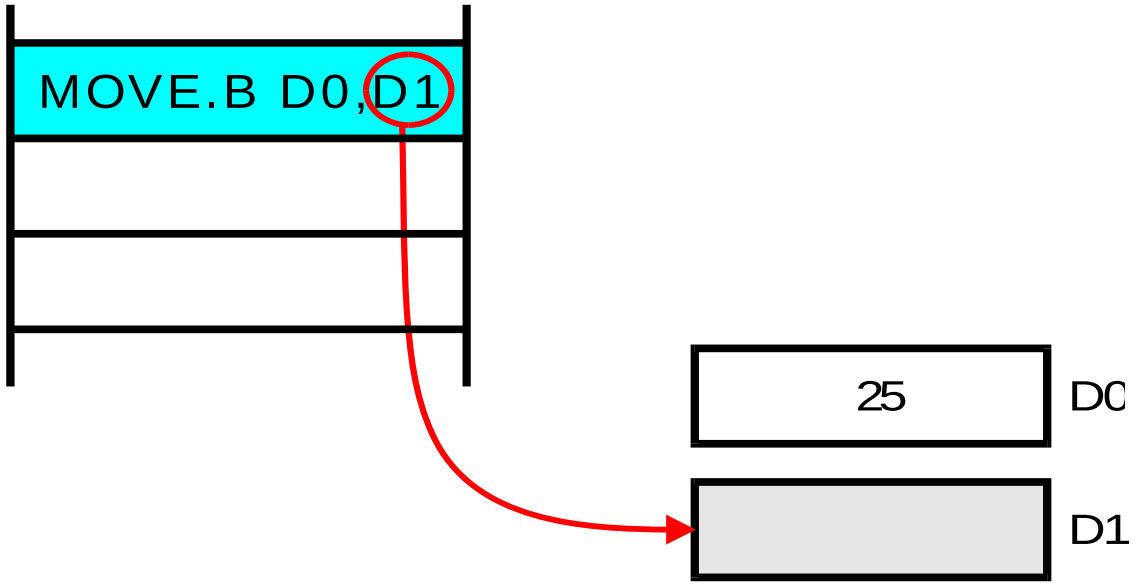
MOVE.B D0,D3	Copia l'operando sorgente in D0 nel registro D3
SUB.L A0,D3	Sottrae l'operando sorgente nel registro A0 dal registro D3
CMP.W D2,D0	Confronta l'op. sorgente nel registro D2 con il registro D0
ADD D3,D4	Somma l'operando sorgente nel registro D3 al registro D4

Register Direct Addressing – Funzionamento



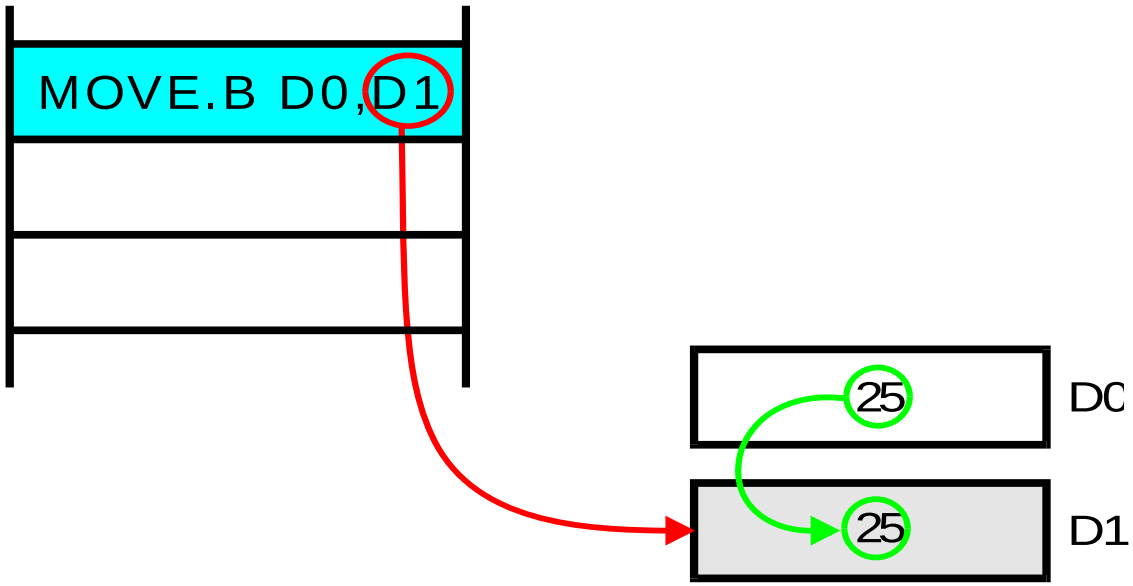
L'istruzione MOVE.B D0,D1 usa registri
dati sia per l'operando sorgente che per quello destinazione

Register Direct Addressing – Funzionamento



L’operando destinazione
è il registro dati D1

Register Direct Addressing – Funzionamento



L’effetto di questa istruzione è quello di copiare il contenuto del registro dati D0 nel registro dati D1

Register Direct Addressing - Caratteristiche

- È veloce, perché non c'è bisogno di accedere alla memoria esterna
- Fa uso di istruzioni corte, perché usa soltanto tre bit per specificare uno degli otto registri dati
  - Mode = 0, reg = 0-7 per Dn
  - Mode = 1, reg = 0-7 per An
  - Ad esempio, per codificare la MOVE D0,D1 bastano 16 bit di parola codice (non sono necessarie parole aggiuntive)
- I programmatori lo usano per memorizzare variabili che sono usate di frequente (scratchpad storage)

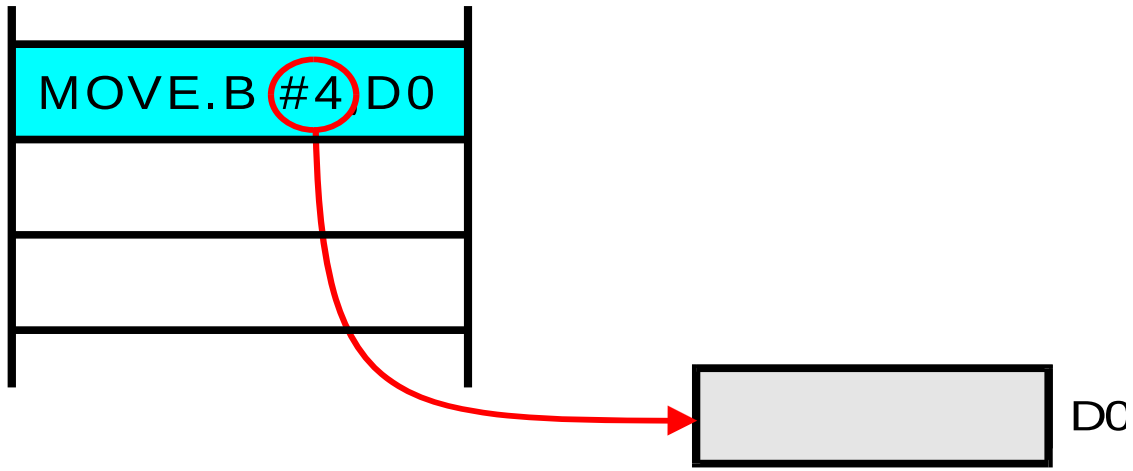
Immediate Addressing

- L'operando effettivo costituisce parte dell'istruzione
- Può essere usato unicamente per specificare un operando sorgente (non si può scrivere su una costante!)
- È indicato da un simbolo # davanti all'operando sorgente
- Un operando immediato è anche chiamato *literal*

Esempio:

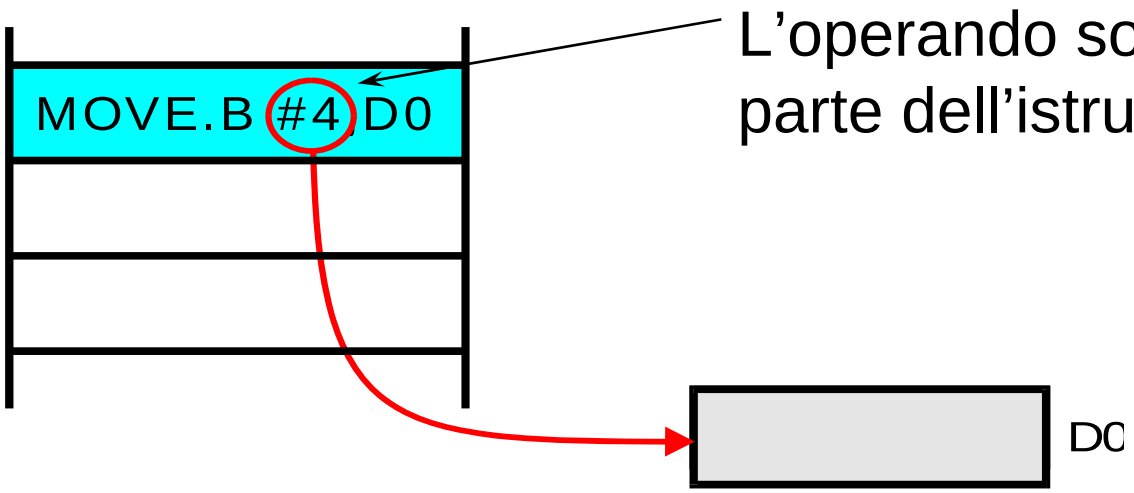
MOVE.B #4,D0 Usa l'operando sorgente immediato 4

Immediate Addressing - Funzionamento

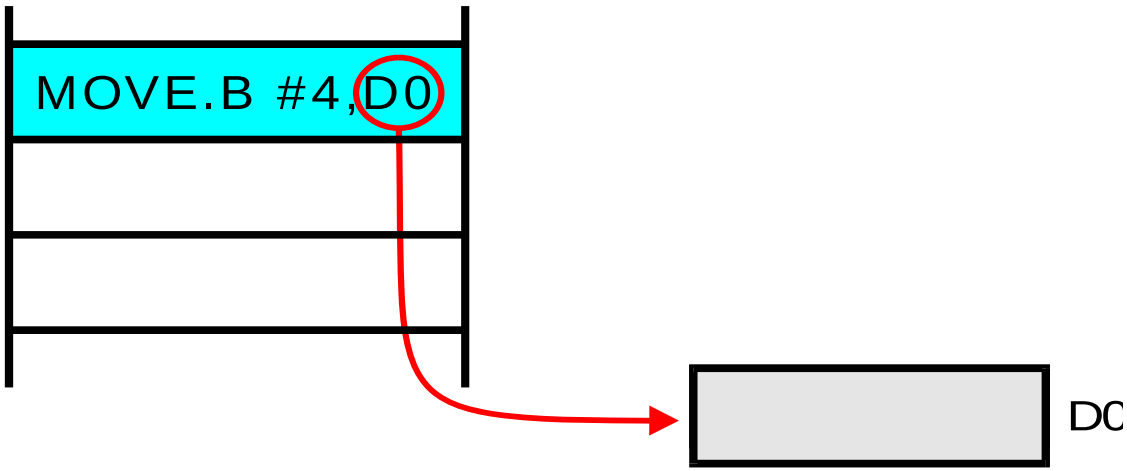


L’istruzione MOVE.B #4,D0 usa un operando sorgente literal ed un operando destinazione register direct

Immediate Addressing - Funzionamento

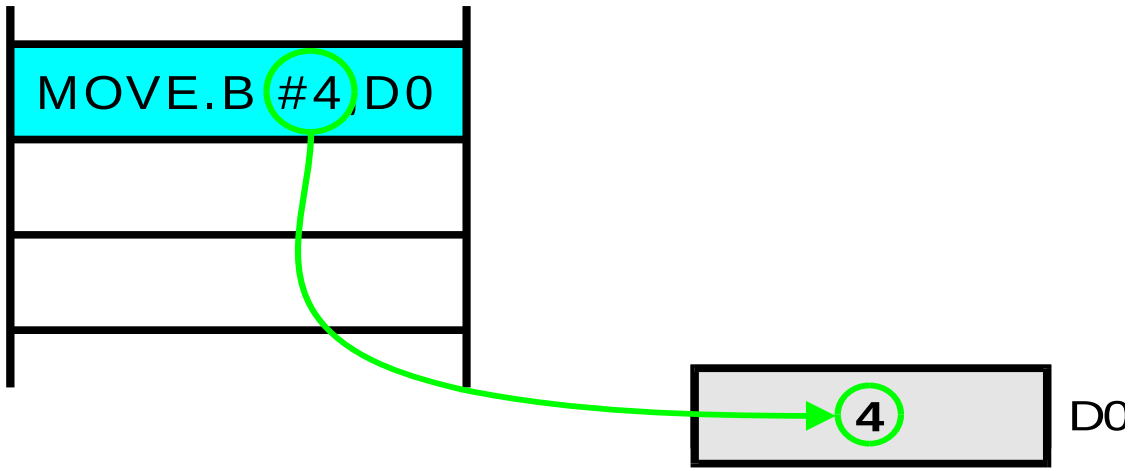


Immediate Addressing - Funzionamento



L’operando destinazione è un registro dati

Immediate Addressing - Funzionamento



L’effetto di questa istruzione è quello di copiare il valore del literal 4 nel registro dati D0

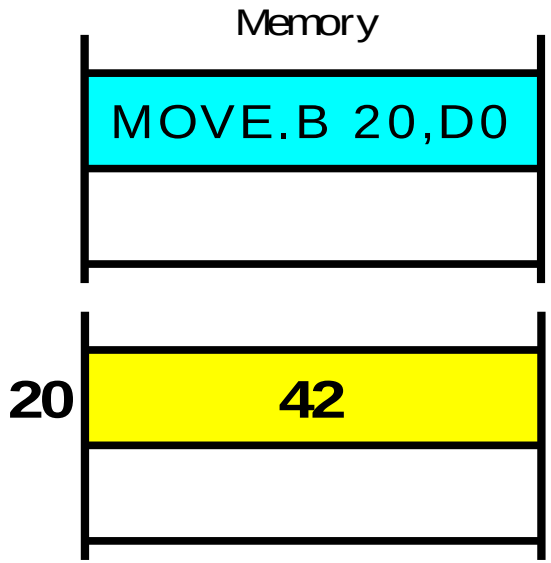
Immediate Addressing - Caratteristiche

- Se la costante da manipolare ha dimensioni ridotte (pochi bit) è possibile codificarla direttamente nei 16 bit dell'istruzione
  - non sono necessarie parole aggiuntive per codificare il *literal* oltre alla parola codice di 16 bit
  - se l'operando destinazione è un registro (*register direct addressing*), l'intera istruzione viene codificata su 16 bit
  - non sono necessarie ulteriori (lenti) accessi in memoria
- Se la costante è "lunga", è necessario usare una o più parole aggiuntive che seguono la parola codice

Absolute Addressing (o Direct Addressing)

- È il modo più semplice per specificare un indirizzo di memoria completo
- L'istruzione fornisce l'indirizzo dell'operando in memoria
- Richiede due accessi in memoria:
  - Il primo è per prelevare l'istruzione e l'indirizzo assoluto
  - Il secondo è per accedere all'operando effettivo
- Esempio:
  - CLR.B 1234 azzera il contenuto della locazione di memoria 1234

Absolute Addressing - Funzionamento

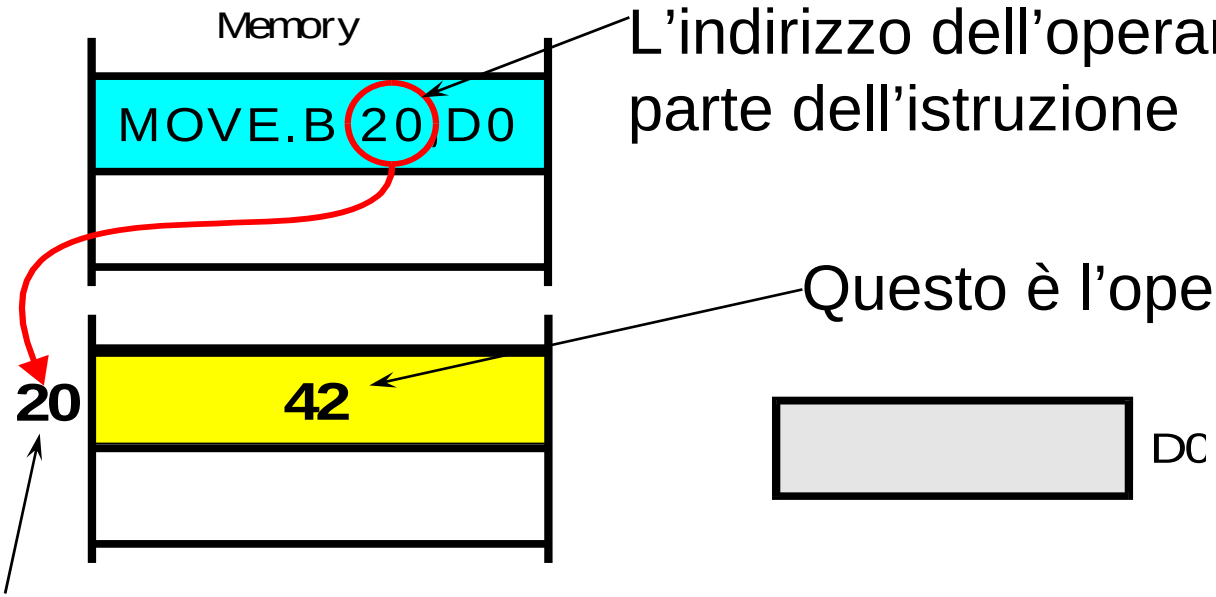


Questa istruzione ha un operando absolute

L'operando sorgente è in memoria

L'operando destinazione usa il direct addressing per un registro dati

Absolute Addressing - Funzionamento

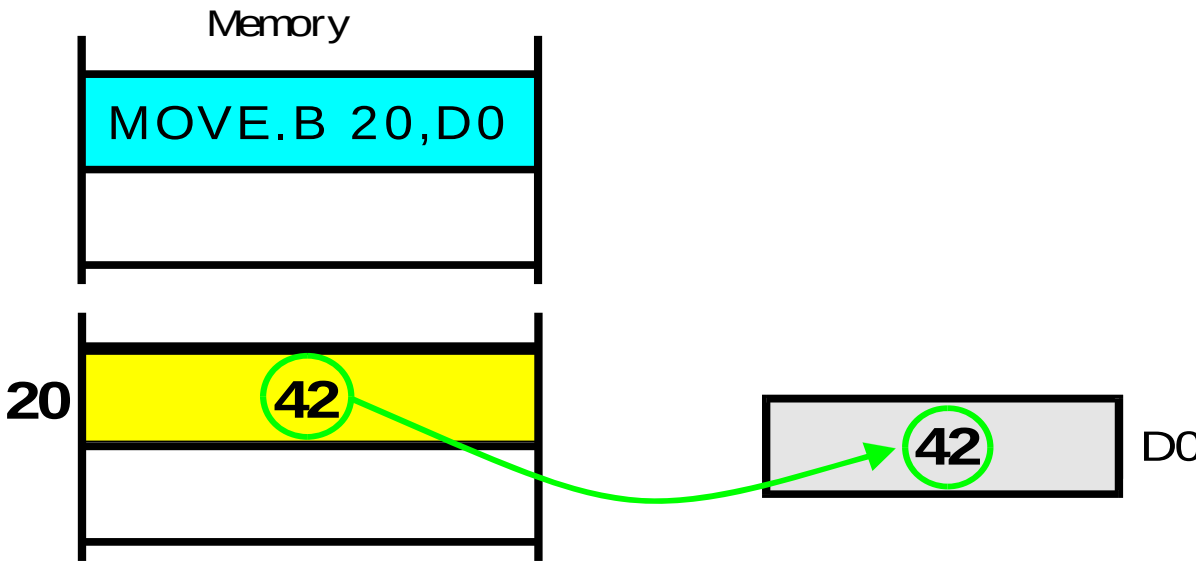


L'indirizzo dell'operando forma parte dell'istruzione

Questo è l'operando effettivo

Una volta che la CPU ha letto l'indirizzo dell'operando dall'istruzione, la CPU accede all'operando effettivo

Absolute Addressing - Funzionamento



L'effetto di MOVE.B 20,D0
è quello di leggere il contenuto della locazione
di memoria 20 e copiarlo nel registro D0

Esempio modi fondamentali

- Consideriamo questo statement in linguaggio di alto livello:

```
char z, y = 27;
z = y + 24;
```

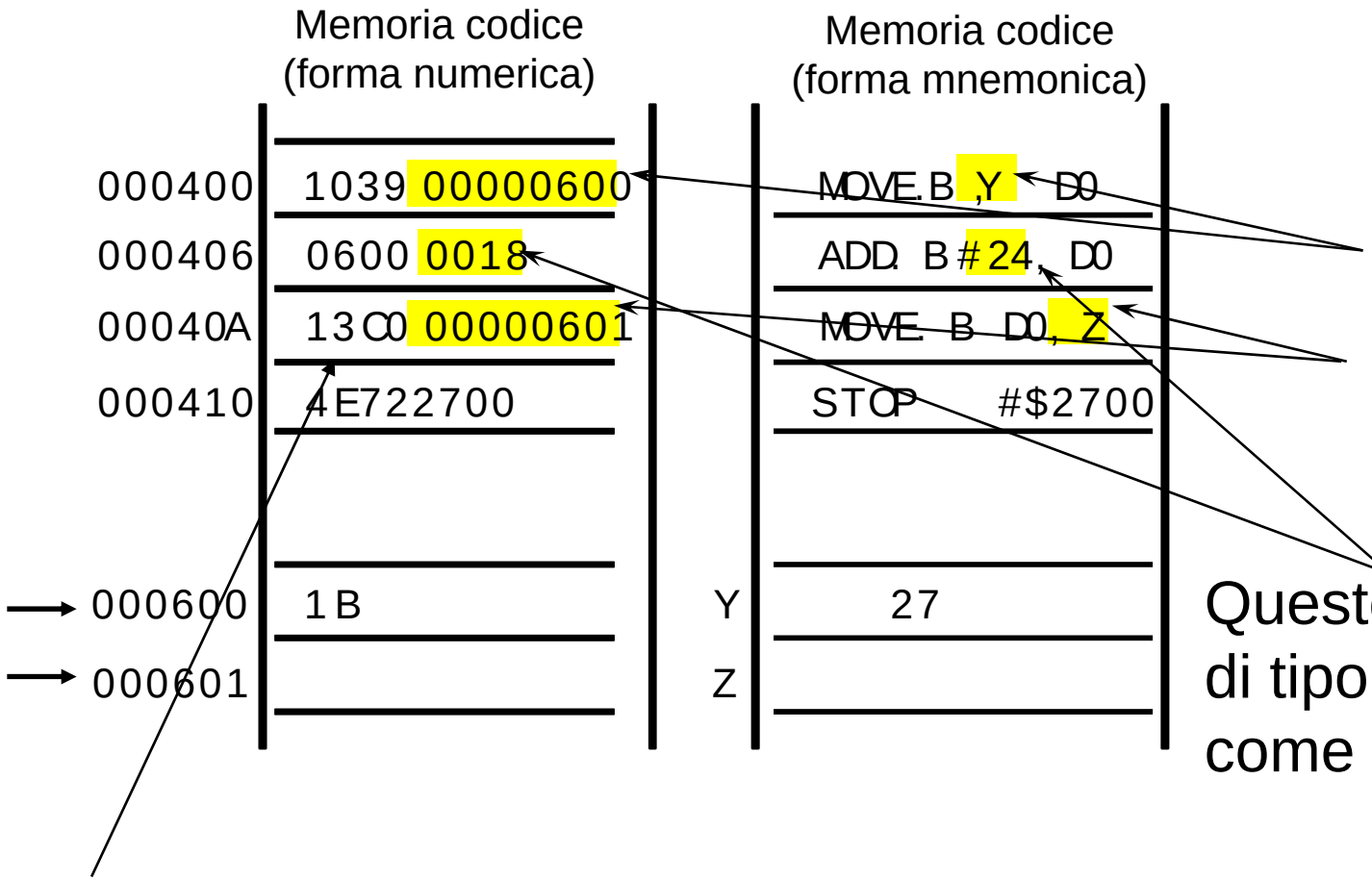
Il seguente frammento di codice lo implementa:

```
        ORG     $400        Inizio del codice
        MOVE.B  Y,D0
        ADD     #24,D0
        MOVE.B  D0,Z

        ORG     $600        Inizio dell'area dati
Y       DC.B    27          Memorizza la costante 27 in memoria
Z       DS.B    1           Riserva un byte per Z
```

Il Programma Assemblato

```
1     00000400                          ORG     $400
2     00000400 103900000600             MOVE.B  Y,D0
3     00000406 06000018                 ADD.B   #24,D0
4     0000040A 13C000000601             MOVE.B  D0,Z
5     00000410 4E722700                 STOP    #$2700
6              *
7     00000600                          ORG     $600
8     00000600 1B               Y:      DC.B    27
9     00000601 00000001         Z:      DS.B    1
```

Mappa della Memoria del programma



Y è una variabile acceduta tramite direct addressing (000600)

Z è una a variabile acceduta mediante direct addressing (000601)

Questo è un operando di tipo literal, memorizzato come parte dell'istruzione

16 bit che codificano l'istruzione (ad es. la MOVE)

Riepilogo modi fondamentali

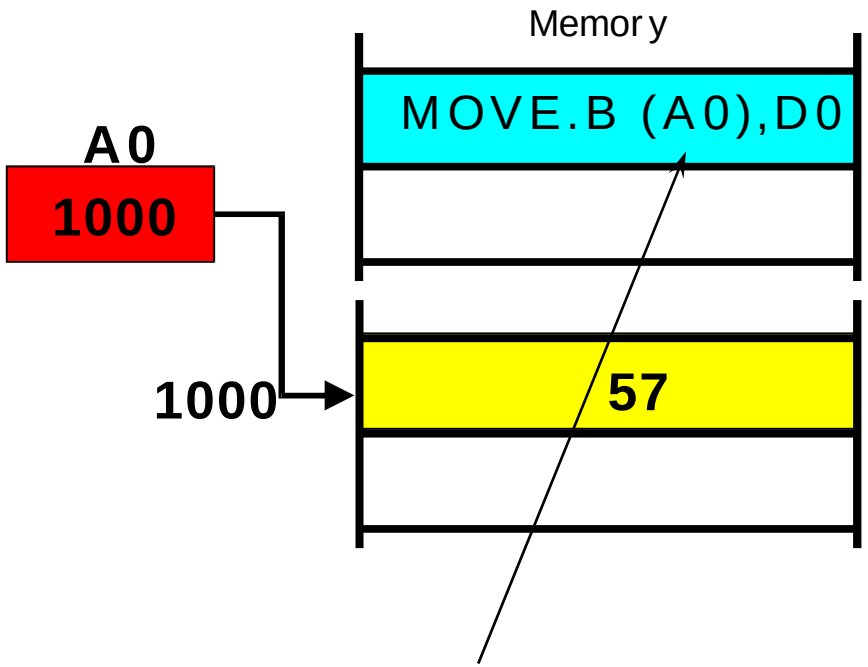
- Register direct addressing - È usato per variabili che possono essere mantenute in registri di memoria
- Literal (immediate) addressing - È usato per costanti che non cambiano
- Direct (absolute) addressing - È usato per variabili che risiedono in memoria

Address Register Indirect Addressing

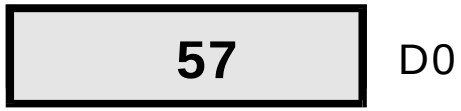
- L'istruzione specifica uno dei registri indirizzo
- Il registro indirizzo specificato contiene l'indirizzo effettivo dell'operando
- Il processore accede all'operando puntato dal registro indirizzo

- Esempio:
  - MOVE.B (A0),D0

Address Register Indirect - Funzionamento

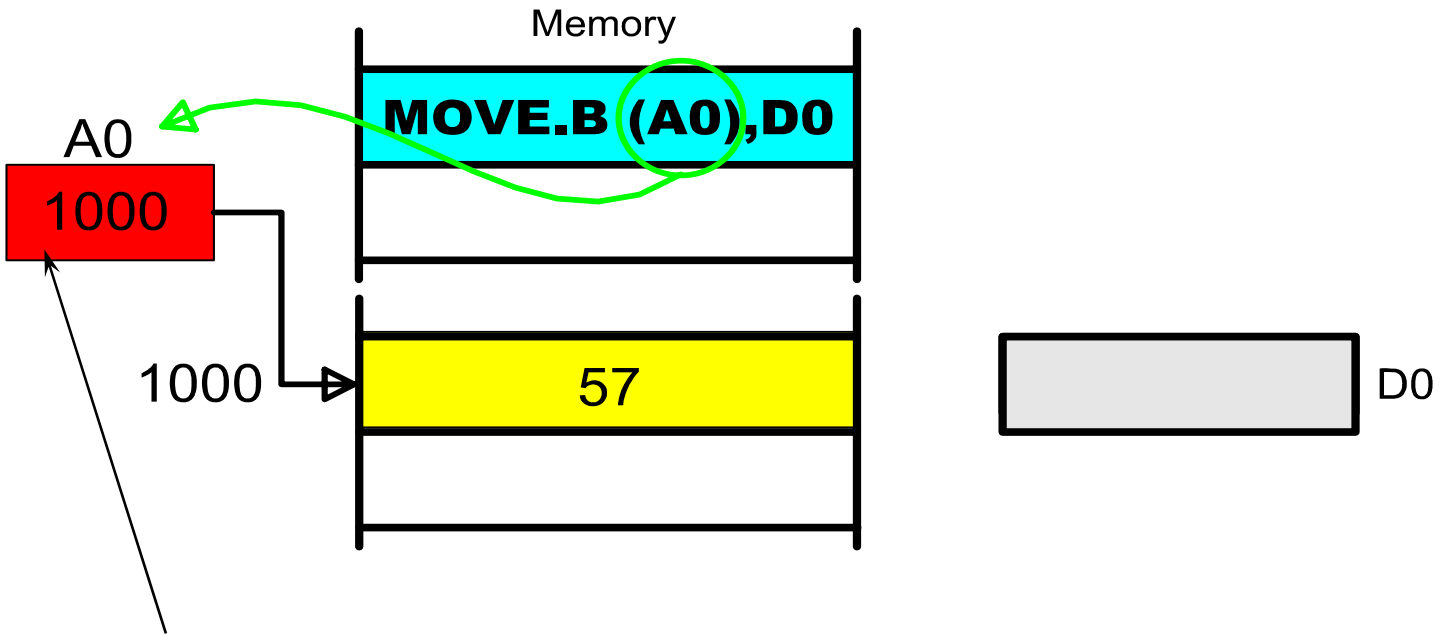


Questa istruzione significa: carica D0 con il contenuto della locazione puntata dal registro indirizzo A0



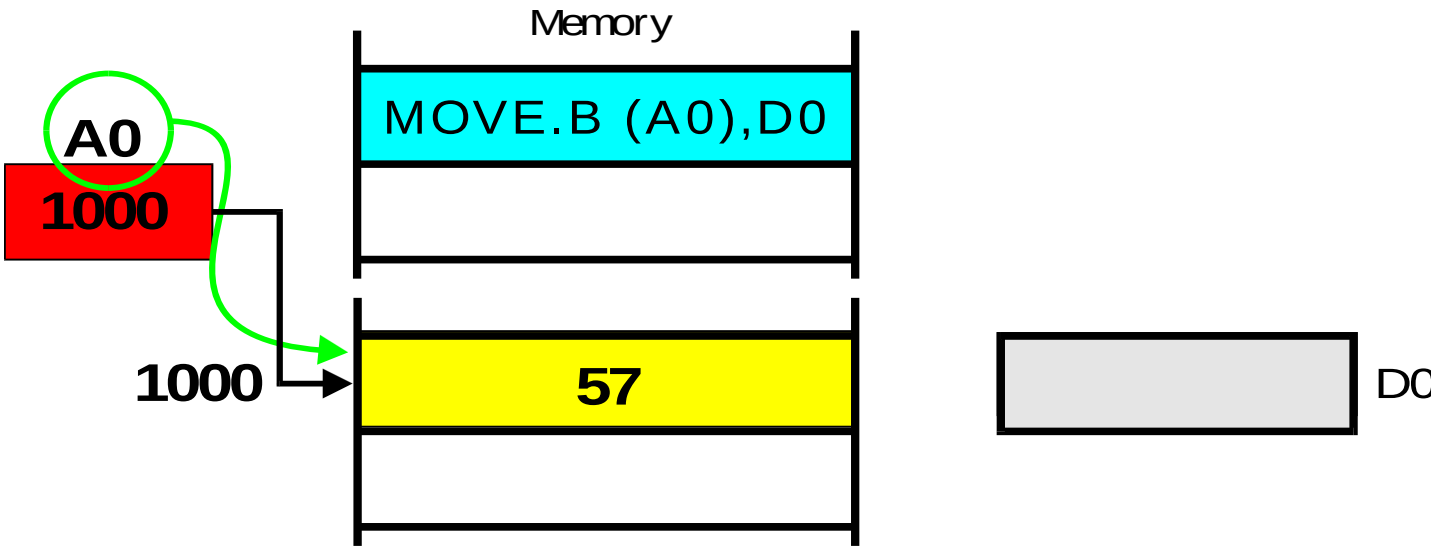
L'istruzione specifica l'operando sorgente come (A0)

Address Register Indirect - Funzionamento



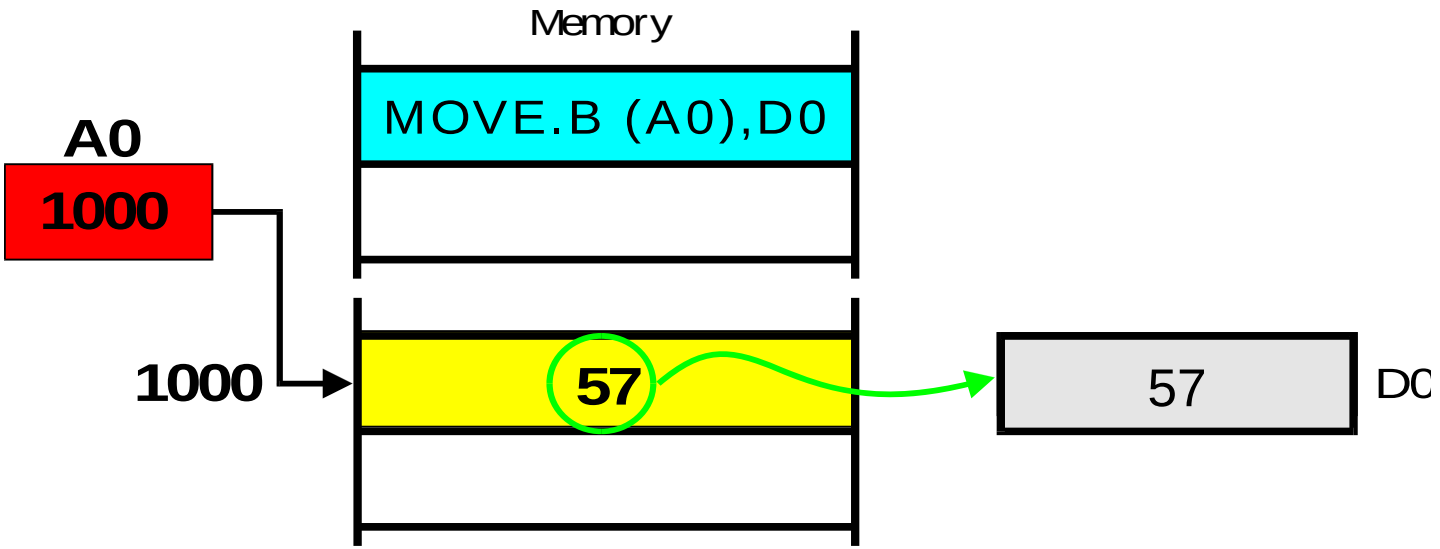
Il registro indirizzo nell'istruzione specifica un registro indirizzo che contiene l'indirizzo dell'operando

Address Register Indirect - Funzionamento



Il registro indirizzo è usato per accedere all'operando in memoria

Address Register Indirect - Funzionamento



Alla fine, il contenuto della locazione puntata da A0 viene copiato nel registro dati

Auto-increment

- L'istruzione specifica uno dei registri indirizzo, usando la modalità Address Register Indirect.
- Se il modo di indirizzamento è specificato come (An)+, il contenuto del registro indirizzo è incrementato di una quantità pari alla dimensione dell'operando *dopo l'uso* ("post-incremento")

- Esempio:
  - MOVE.W (A0)+, D0

    Usa A0 per la MOVE e poi gli aggiunge 2 (2 poiché l'accesso è di tipo .W = 2 byte). Di fatto, l'istruzione esegue un pop in D0 dallo stack puntanto da A0

Auto-decrement

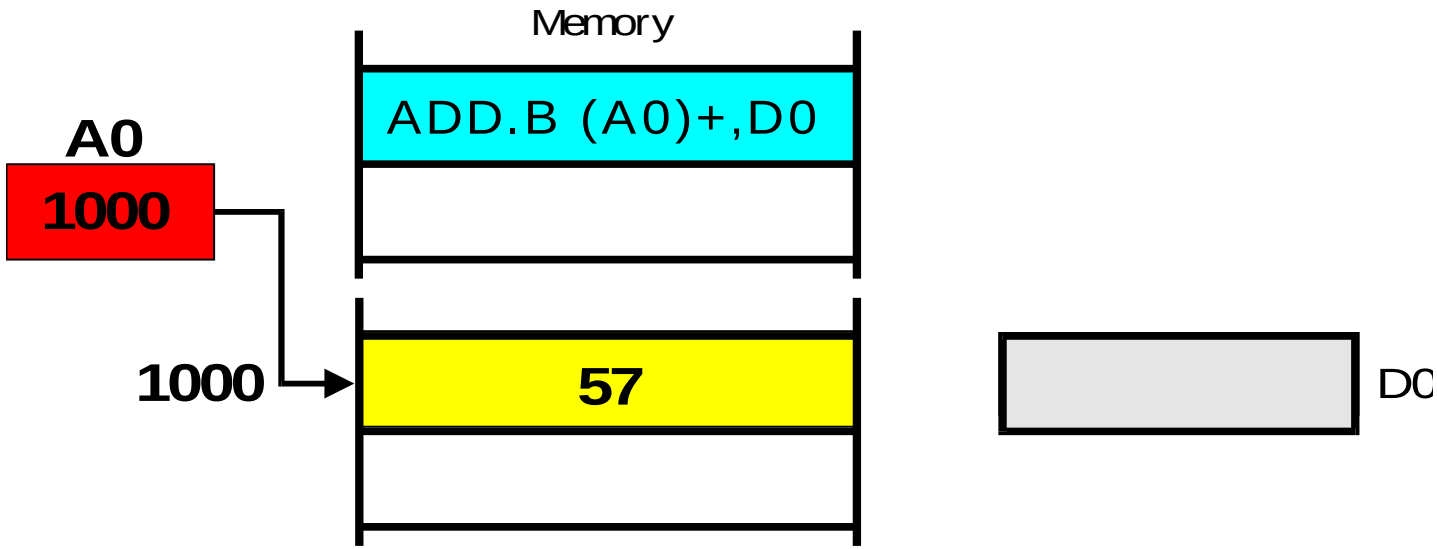
- L'istruzione specifica uno dei registri indirizzo
- Se il modo di indirizzamento è specificato come

  -(An), il contenuto del registro indirizzo è decrementato di una quantità pari alla dimensione dell'operando *prima dell'uso* ("pre-decremento")

- Esempio:
  - MOVE.W D0,-(A0)

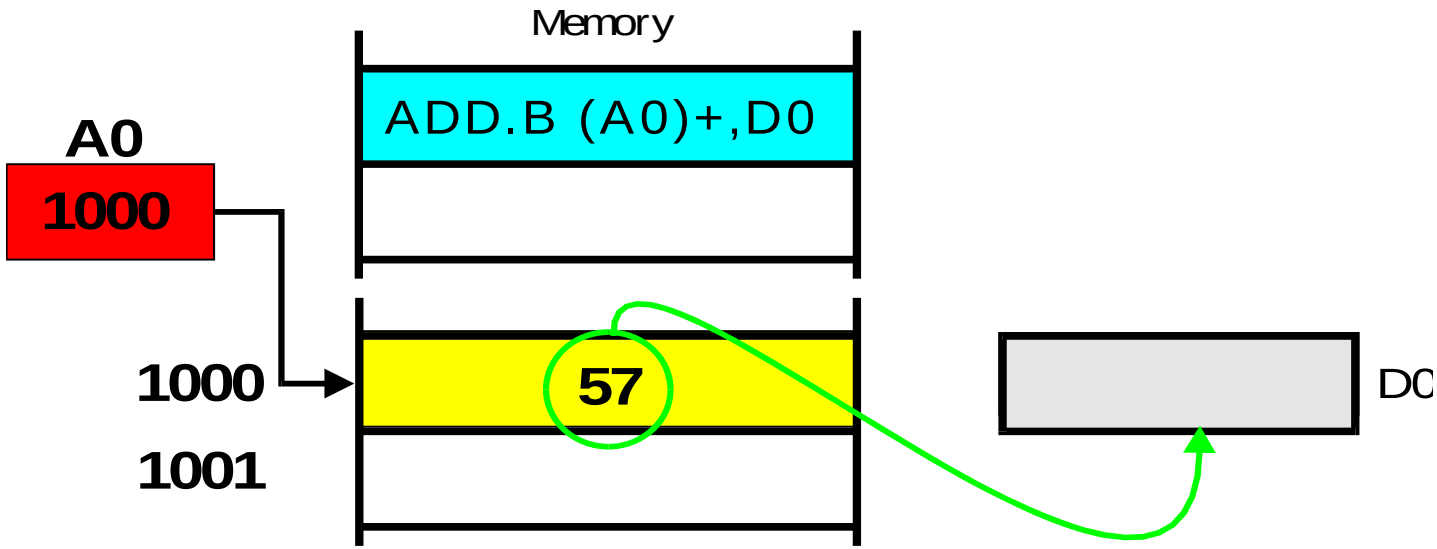
    Sottrae 2 ad A0 e poi lo usa per la MOVE (2 poiché l'accesso è di tipo .W = 2 byte). Di datto, l'istruzione esegue un push di D0 sullo stack puntato da A0.

Auto-increment - Funzionamento



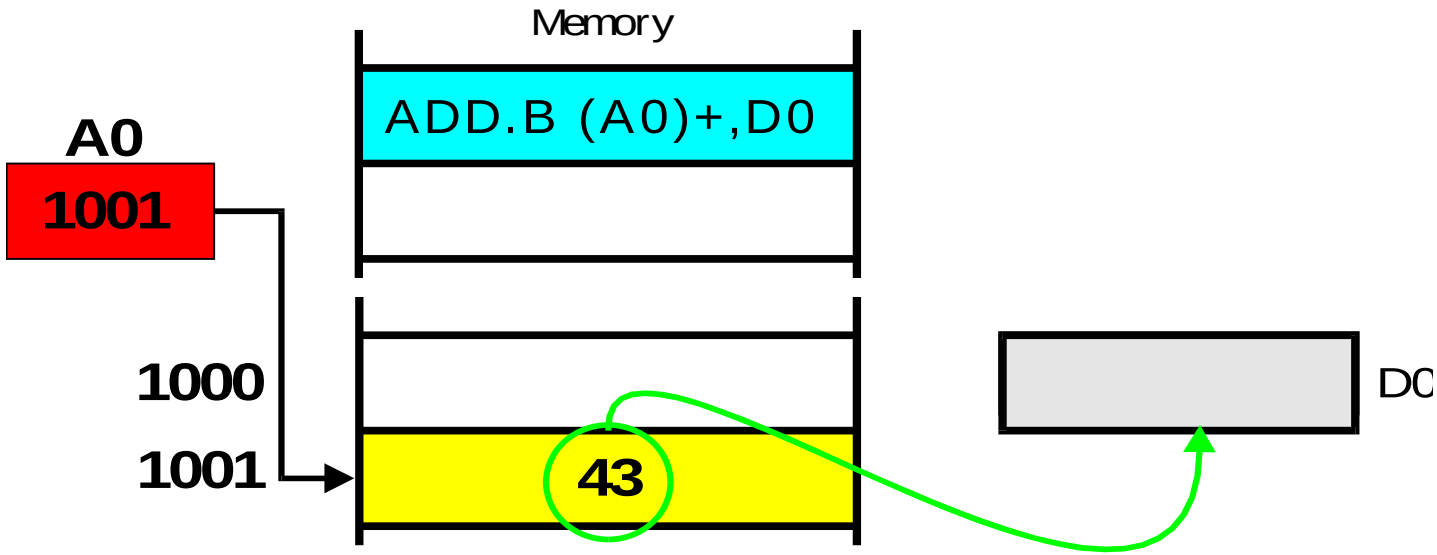
Il registro indirizzo contiene 1000
ovvero “punta” alla locazione 1000

Auto-increment - Funzionamento



Il registro A0 viene usato per accedere alla locazione di memoria 1000 e il contenuto di questa locazione (57) viene sommato a D0

Auto-increment - Funzionamento



Dopo che l'istruzione è stata eseguita,
il contenuto di A0 viene incrementato, per
puntare alla locazione successiva

Esempio - Codice

- Il seguente frammento di codice usa l'address register indirect addressing con post-increment per sommare cinque numeri memorizzati in locazioni di memoria consecutive.

```
        MOVE.B  #5,D0       Cinque numeri da sommare
        LEA     Table,A0    A0 punta alla tabella dei numeri
        CLR.B   D1          Inizializza la somma
Loop    ADD.B   (A0)+,D1    Somma il numero al totale
        SUB.B   #1,D0       Decrementa D0 per gestire il conteggio
        BNE     Loop        Itare fino a 5
        STOP    #$2700
*
Table   DC.B    1,4,2,6,5   Dati d'esempio
```

Esempio – Trace del programma

```
>DF
PC=000400 SR=2000 SS=00A00000 US=00000000      X=0
A0=00000000 A1=00000000 A2=00000000 A3=00000000 N=0
A4=00000000 A5=00000000 A6=00000000 A7=00A00000 Z=0
D0=00000000 D1=00000000 D2=00000000 D3=00000000 V=0
D4=00000000 D5=00000000 D6=00000000 D7=00000000 C=0
---------->MOVE.B   #$05,D0

>TR
PC=000404 SR=2000 SS=00A00000 US=00000000      X=0
A0=00000000 A1=00000000 A2=00000000 A3=00000000 N=0
A4=00000000 A5=00000000 A6=00000000 A7=00A00000 Z=0
D0=00000005 D1=00000000 D2=00000000 D3=00000000 V=0
D4=00000000 D5=00000000 D6=00000000 D7=00000000 C=0
---------->LEA.L    $0416,A0

Trace>
PC=00040A SR=2000 SS=00A00000 US=00000000      X=0
A0=00000416 A1=00000000 A2=00000000 A3=00000000 N=0
A4=00000000 A5=00000000 A6=00000000 A7=00A00000 Z=0
D0=00000005 D1=00000000 D2=00000000 D3=00000000 V=0
D4=00000000 D5=00000000 D6=00000000 D7=00000000 C=0
---------->CLR.B    D1
```

D0 è stato caricato con 5

Questa istruzione carica A0 con il valore $0416

A0 ora contiene $0416

Esempio – Trace del programma

```
Trace>
PC=00040C SR=2004 SS=00A00000 US=00000000      X=0
A0=00000416 A1=00000000 A2=00000000 A3=00000000 N=0
A4=00000000 A5=00000000 A6=00000000 A7=00A00000 Z=1
D0=00000005 D1=00000000 D2=00000000 D3=00000000 V=0
D4=00000000 D5=00000000 D6=00000000 D7=00000000 C=0
---------->ADD.B    (A0)+,D1

Trace>
PC=00040E SR=2000 SS=00A00000 US=00000000      X=0
A0=00000417 A1=00000000 A2=00000000 A3=00000000 N=0
A4=00000000 A5=00000000 A6=00000000 A7=00A00000 Z=0
D0=00000005 D1=00000001 D2=00000000 D3=00000000 V=0
D4=00000000 D5=00000000 D6=00000000 D7=00000000 C=0
---------->SUBQ.B   #$01,D0

Trace>
PC=000410 SR=2000 SS=00A00000 US=00000000      X=0
A0=00000417 A1=00000000 A2=00000000 A3=00000000 N=0
A4=00000000 A5=00000000 A6=00000000 A7=00A00000 Z=0
D0=00000004 D1=00000001 D2=00000000 D3=00000000 V=0
D4=00000000 D5=00000000 D6=00000000 D7=00000000 C=0
---------->BNE.S    $040C
```

Questa istruzione carica il contenuto della locazione puntata da A0 in D1

Siccome l'operando era (A0)+, il contenuto di A0 viene incrementato

ADD.B (A0)+,D1 somma l'operando sorgente a D1

Esempio – Trace del programma

```
Trace>
PC=00040C SR=2000 SS=00A00000 US=00000000      X=0
A0=00000417 A1=00000000 A2=00000000 A3=00000000 N=0
A4=00000000 A5=00000000 A6=00000000 A7=00A00000 Z=0
D0=00000004 D1=00000001 D2=00000000 D3=00000000 V=0
D4=00000000 D5=00000000 D6=00000000 D7=00000000 C=0
---------->ADD.B    (A0)+,D1

Trace>
PC=00040E SR=2000 SS=00A00000 US=00000000      X=0
A0=00000418 A1=00000000 A2=00000000 A3=00000000 N=0
A4=00000000 A5=00000000 A6=00000000 A7=00A00000 Z=0
D0=00000004 D1=00000005 D2=00000000 D3=00000000 V=0
D4=00000000 D5=00000000 D6=00000000 D7=00000000 C=0
---------->SUBQ.B   #$01,D0

Trace>

PC=000410 SR=2000 SS=00A00000 US=00000000      X=0
A0=00000418 A1=00000000 A2=00000000 A3=00000000 N=0
A4=00000000 A5=00000000 A6=00000000 A7=00A00000 Z=0
D0=00000003 D1=00000005 D2=00000000 D3=00000000 V=0
D4=00000000 D5=00000000 D6=00000000 D7=00000000 C=0
---------->BNE.S    $040C
```

Al ciclo successivo l' istruzione ADD.B (A0)+,D1 usa A0 come operando sorgente e poi incrementa il contenuto di A0

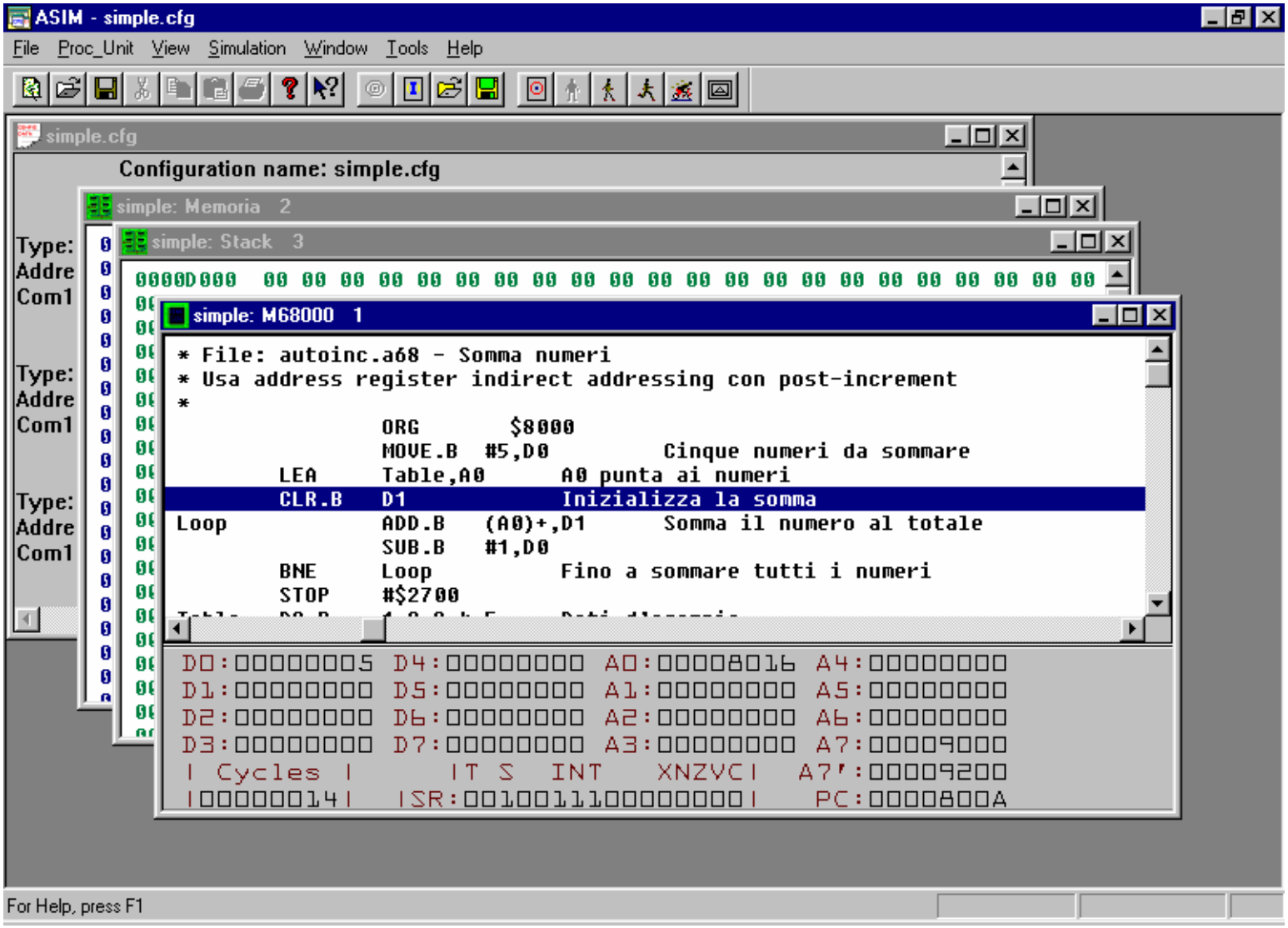
Problema

- Scrivere un programma assembly che sommi cinque numeri memorizzati in locazioni di memoria consecutive
- Assemblare ed eseguire sul simulatore
- Sperimentare:
  - L'effetto dell'istruzione LEA
  - L'effetto dell'autoincremento

Soluzione - Codice

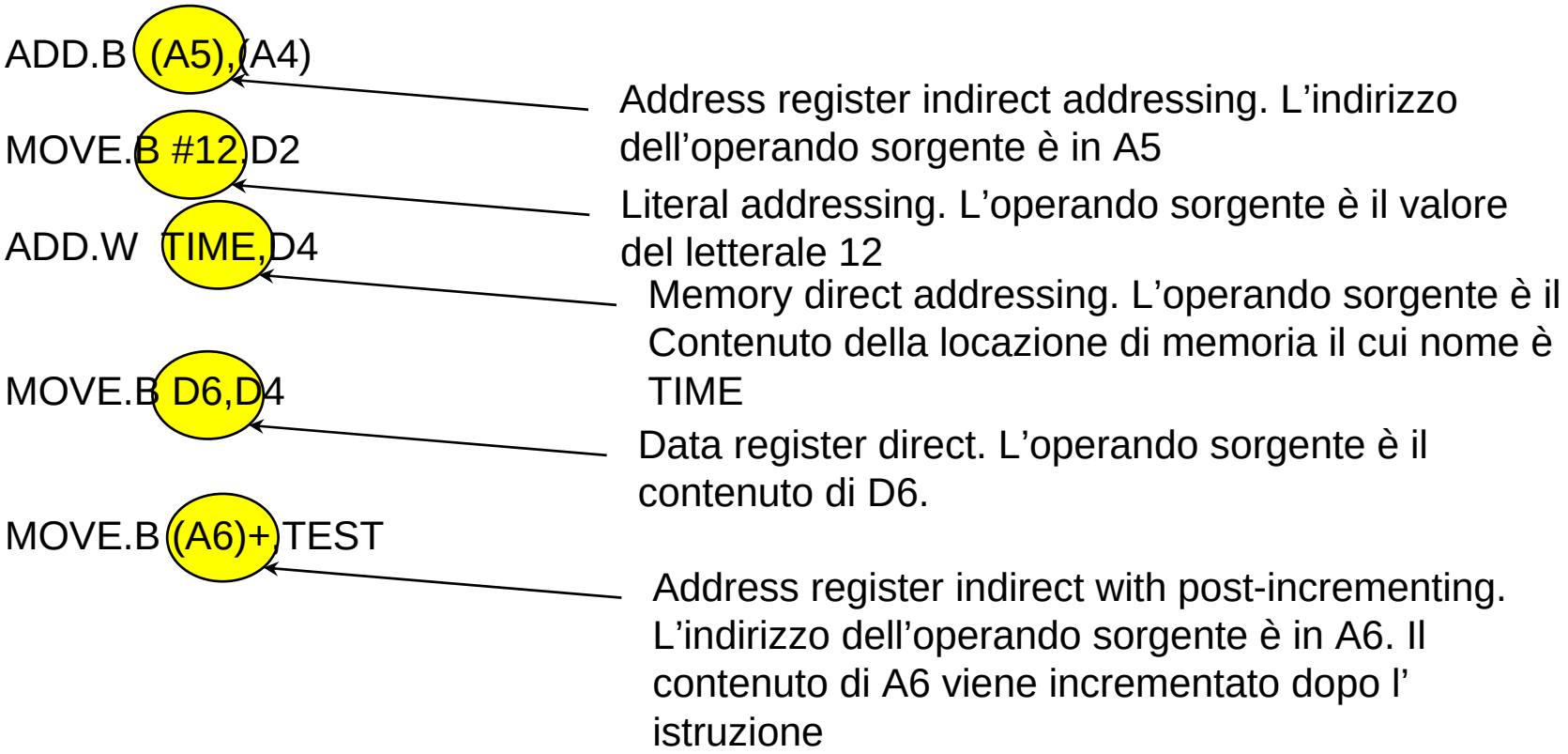
```
* Esempio - Somma numeri

          ORG     $8000
          MOVE.B  #5,D0
          LEA     Table,A0        A0 punta ai numeri
          CLR.B   D1              Inizializza la somma
Loop      ADD.B   (A0)+,D1        Somma il numero al totale
          SUB.B   #1,D0
          BNE     Loop
STOP      #$2700
Table     DC.B    1,2,3,4,5       Dati d'esempio
```

Soluzione - Esecuzione



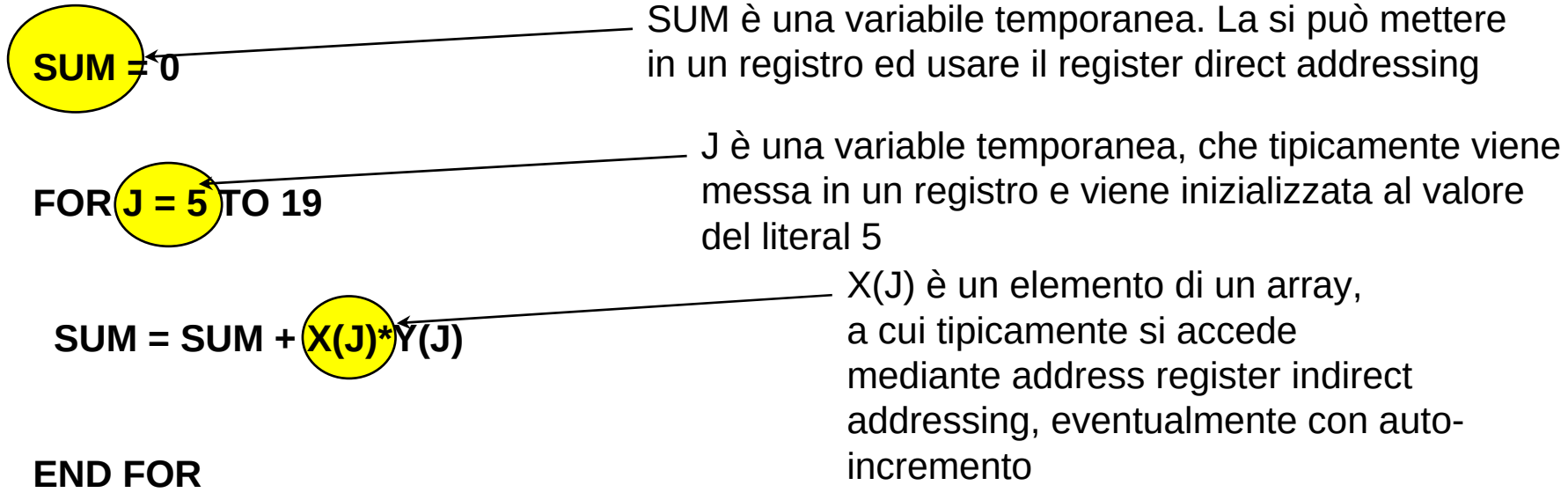
Problema

- Identificare l'addressing mode usato per l'operando sorgente in ciascuna delle seguenti istruzioni



Problema

- Se doveste tradurre il seguente frammento di pseudo-codice in linguaggio assembly, quali modi di indirizzamento utilizzereste?



Indexed

- In generale, l'Indexed Addressing combina due componenti mediante somma, per formare l'EA
  - Il primo componente è detto *base address* ed è specificato come parte dell'istruzione (come nell'absolute addressing)
  - Il secondo componente è detto *index register* e contiene il valore da sommare al base address per ottener l'EA
- È adatto per accedere ai valori di array e di tabelle
- Il processore MC68000 non supporta esplicitamente l'Indexed Addressing. Tuttavia, è possibile usare l'Indexed Short Addressing nei (32+32)Kbyte agli estremi dei 4GB dello spazio di memoria
- Esempio:

```
MOVEA.W     I,AO
MOVE.B      CLIST-1(AO),D1          Leggi clist[i]
```

Based Addressing

- Based Addressing è esattamente l'inverso dell'Indexed Addressing, in quanto combina due componenti mediante somma, per formare l'EA, ma:
  - Il primo componente è detto *displacement* ed è specificato come parte dell'istruzione (come nell'absolute addressing)
  - Il secondo componente è detto *base address* ed è contenuto in un registro
- È adatto per accedere ai valori di array e di tabelle di cui si conosca la posizione relativa ad assembly time, ma non quella iniziale
- Il processore MC68000 supporta il Based Addressing come l'Indexed

Based Indexed

- Based Indexed Addressing combina due componenti mediante somma, per formare l'EA, ma:
  - Il primo componente è detto registro base e contiene il *base address*
  - Il secondo componente è detto registro indice e contiene il *displacement*
- Consente di calcolare a run time sia la posizione iniziale che quella relativa di tabelle ed array
- Il processore MC68000 supporta lo Short Based Indexed ed il Long Based Indexed

Relative Addressing

- “Relative” indica che il calcolo dell’indirizzo è relativo al Program Counter (PC)
- Questi modi di indirizzamento calcolano l’indirizzo effettivo come la somma di un displacement fisso specificato nell’istruzione e del valore corrente del PC
- Fanno spesso uso di displacement piccoli, di 8 o 16 bit, per specificare indirizzi vicini all’istruzione corrente, anziché ricorrere a indirizzi assoluti di 32 bit
- Il 68000 non consente di utilizzare questi modi di indirizzamento per specificare operandi che potrebbero essere modificati

Relative Indexed Addressing

- Variante del Relative
- Funziona come il Based Indexex, ma il base register è sostituito dal PC
- Può essere usato per saltare ad aree di memoria read-only, contenenti dati o istruzioni