

**Università di Napoli Federico II – Scuola Politecnica e delle Scienze di Base**  
**Corso di Laurea in Ingegneria Informatica**



# **Corso di Calcolatori Elettronici I**

Modi di indirizzamento del processore MC68000



# Organizzazione della memoria



DIE  
TI.  
UNI  
NA

- 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

posizione del byte in memoria

MS byte	1	2	3	LS byte
0	1	2	3	
4	5	6	7	
..	..	..	..	
$2^{k-4}$	$2^{k-3}$	$2^{k-2}$	$2^{k-1}$	

Disposizione BIG-ENDIAN

MS byte	2	1	0	LS byte
3	2	1	0	
7	6	5	4	
..	..	..	..	
$2^{k-1}$	$2^{k-2}$	$2^{k-3}$	$2^{k-4}$	

Disposizione LITTLE-ENDIAN

# Parallelismo dell'Address Bus e dimensione dei registri indirizzo

## Address bus



Parallelismo bus indirizzi: registro indirizzi determina il numero di indirizzi “fisici” distinti che la CPU è in grado di generare all'esterno

Dimensione registri indirizzo (es. A0, PC): determina il numero di indirizzi “logici” distinti che la CPU può trattare nei programmi

Non è detto che le due dimensioni coincidano

Lo spazio di indirizzamento logico è in generale diverso dallo spazio di indirizzamento fisico

# Aliasing degli indirizzi

- Spazio di indirizzamento logico e spazio di indirizzamento fisico possono non coincidere
- *Causa* : nel MC68000 il parallelismo dell'Address Bus è 24 bit, la dimensione dei registri indirizzo (A0-A7, PC) è 32 bit
- *Conseguenza* : Valori diversi contenuti in un registro indirizzi possono attivare la stessa locazione fisica di memoria poiché differiscono solo per gli 8 bit più significativi
  - Ad es.: \$0000A3B2 e \$0A00A3B2
- Questo fenomeno prende il nome di **aliasing degli indirizzi**

# Caratteristiche del processore MC68000



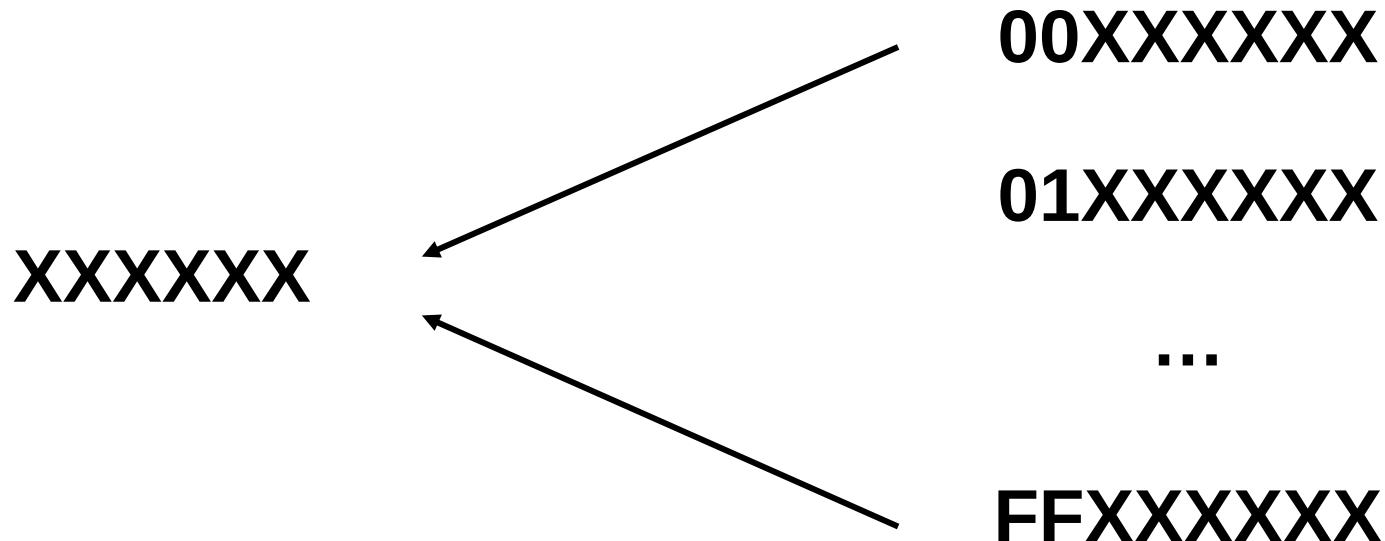
- Memoria Byte Addressable
- Parallelismo Registri Indirizzo: 32 bit
  - Spazio di indirizzamento logico: 4 GB
- Parallelismo Address Bus: 24 bit
  - Spazio di indirizzamento fisico: 16 MB
- Parallelismo Data Bus: 16 bit
  - Pur disponendo di istruzioni in grado di trattare dati a 32 bit, il processore 68000 è in grado di leggere/scrivere solo due locazioni consecutive alla volta (word allineate)
  - L'unità di controllo realizza accessi a 32 bit attraverso sequenze di due accessi da 16 bit

# Caratteristiche del processore MC68020

- Memoria Byte Addressable
- Parallelismo Registri Indirizzo: 32 bit
  - Spazio di indirizzamento logico: 4 GB
- Parallelismo Address Bus: 32 bit
  - Spazio di indirizzamento fisico: 4 GB
- Parallelismo Data Bus: 32 bit
  - Il processore 68020 è in grado di leggere/scrivere longword costituite da 4 locazioni consecutive attraverso un unico accesso alla memoria, purchè le longword siano allineate sui limiti di parola (cominciano ad un indirizzo pari)

# Aliasing nel MC68000

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



# 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 MC68000

- Register Direct
  - Data-register Direct
  - Address-register Direct
- Immediate (or Literal)
- Absolute
  - Short (16 bit)
  - Long (32 bit)
- Address-register Indirect
  - Auto-Increment
  - Auto-Decrement
- Indexed short
- Based
- Based Indexed
  - Short
  - Long
- Relative
- Relative Indexed
  - Short
  - Long

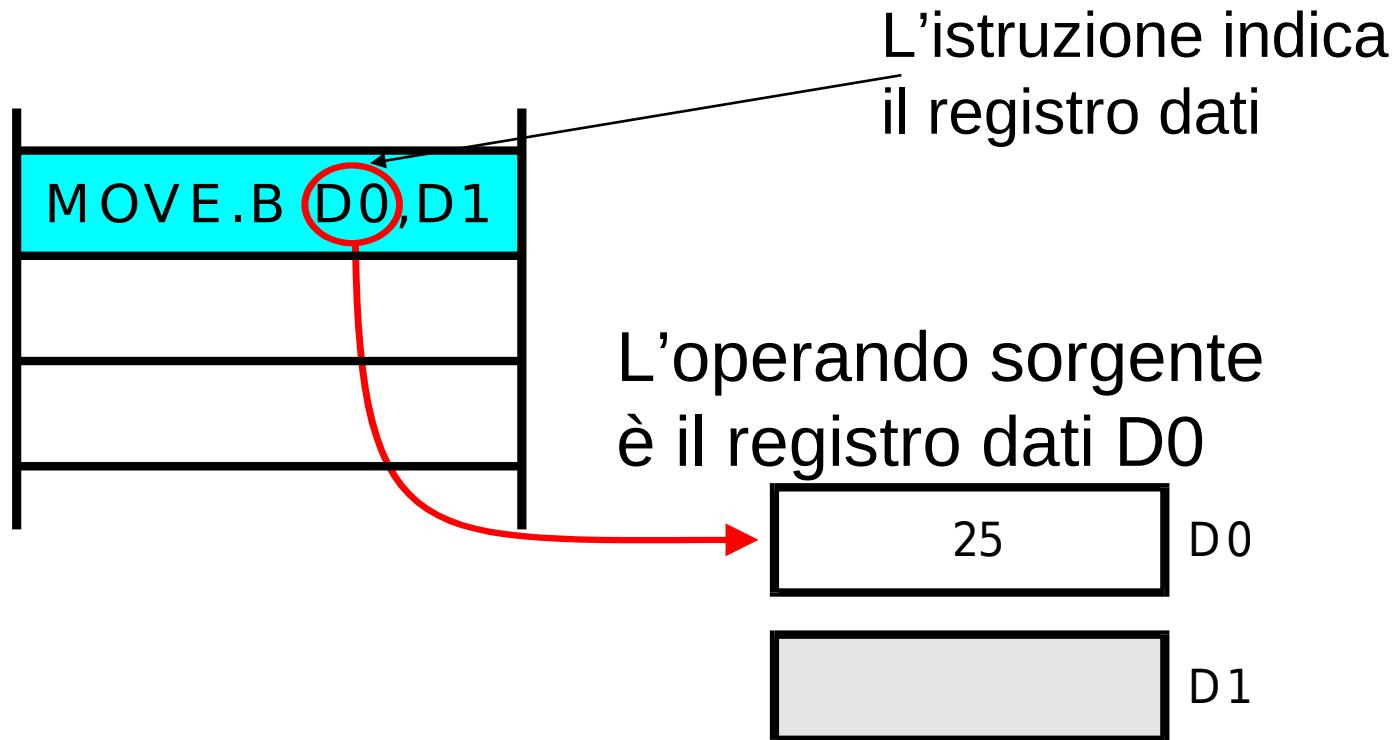
# Register Direct Addressing



- 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

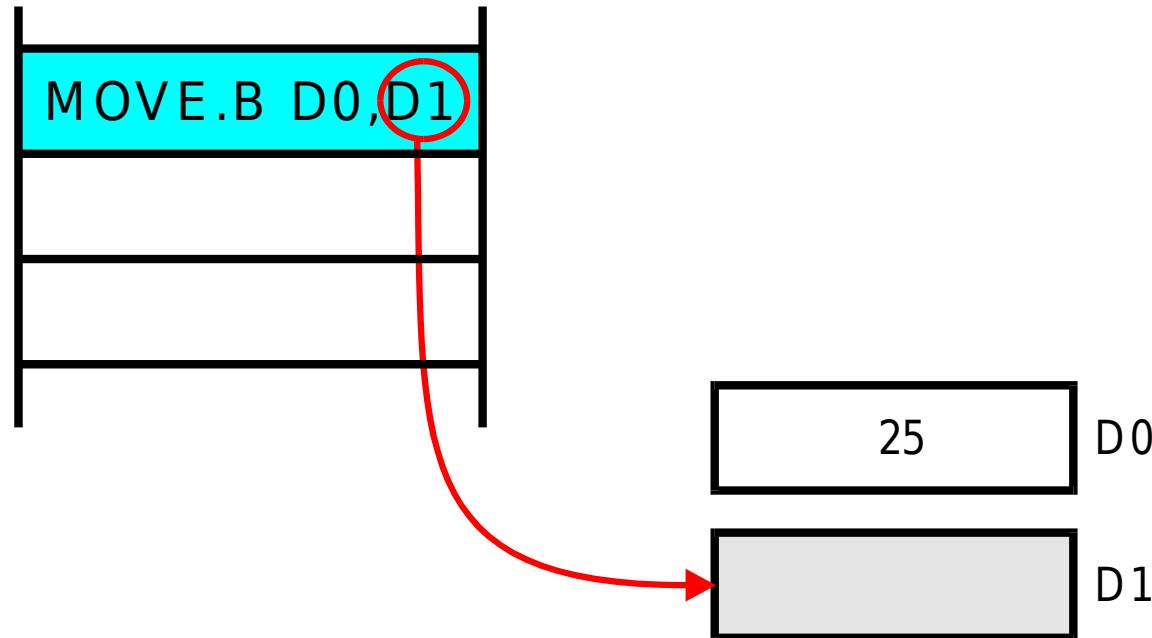
MOVE.B D0,D3	Copia l'operando sorgente in D0 nel registro D3
SUB.L A0,D3	Sottrae l'operando sorgente nel registro A0 da D3
CMP.W D2,D0	Confronta l'op. sorgente nel registro D2 con D0
ADD D3,D4	Somma l'operando sorgente nel registro D3 a D4

# Register Direct Addressing



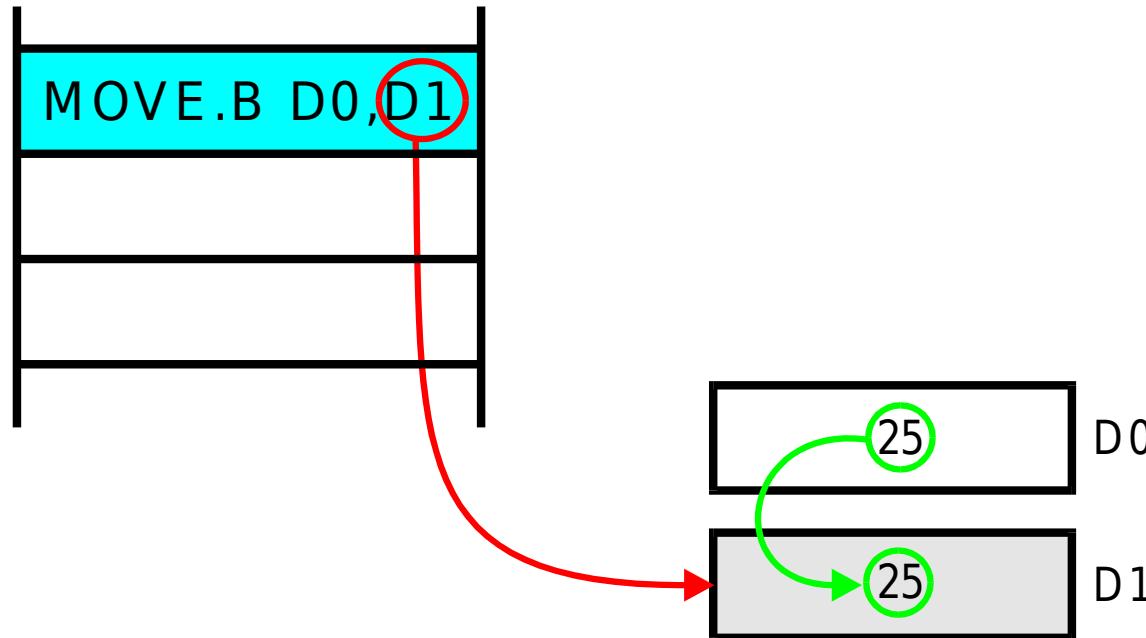
L'istruzione **MOVE.B D0,D1** usa registri dati sia per l'operando sorgente che per quello destinazione

# Register Direct Addressing



L'operando destinazione  
è il registro dati D1

# Register Direct Addressing



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 programmatore lo usano per memorizzare variabili che sono usate di frequente

# Immediate Addressing

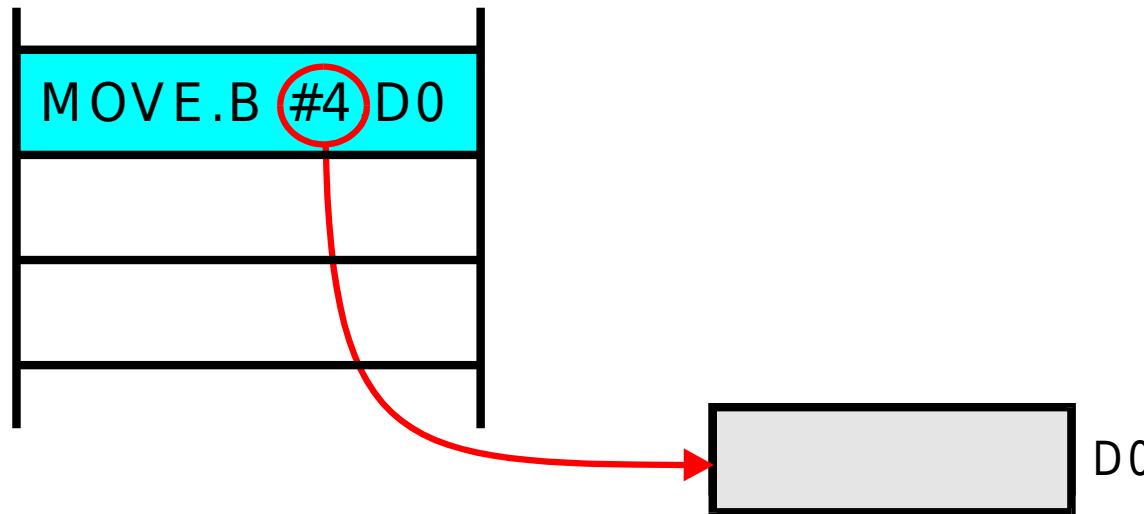


- L'operando effettivo costituisce parte dell'istruzione
- Può essere usato unicamente per specificare un operando sorgente (non si può scrivere su una costante!)
- È indicato da un simbolo # davanti all'operando sorgente
- Un operando immediato è anche chiamato *literal*

Esempio:

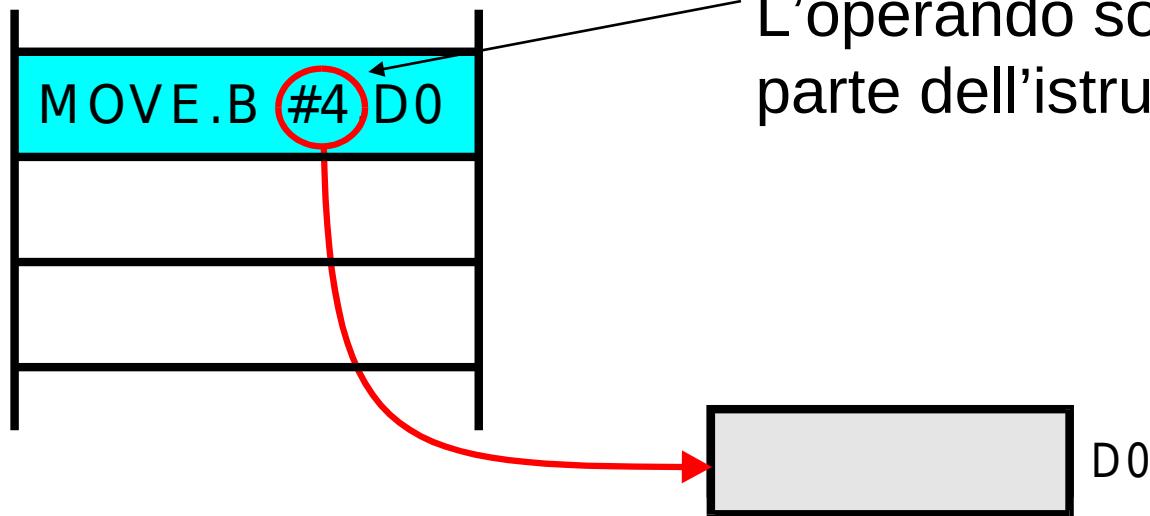
MOVE.B #4,D0      Usa l'operando sorgente immediato 4

# Immediate Addressing - Funzionamento



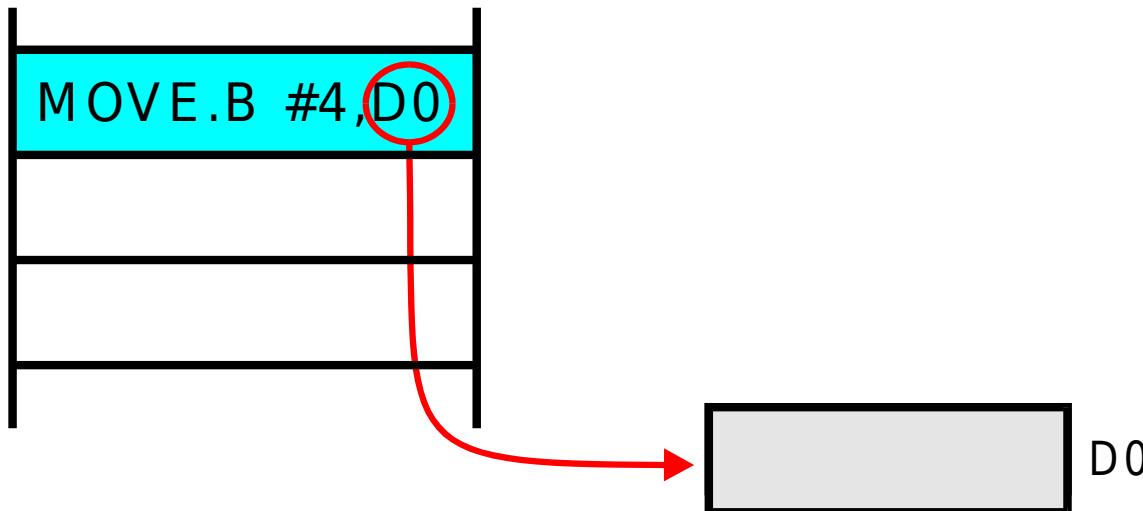
L'istruzione `MOVE.B #4,D0` usa un operando sorgente immediato ed un operando destinazione register direct

# Immediate Addressing - Funzionamento



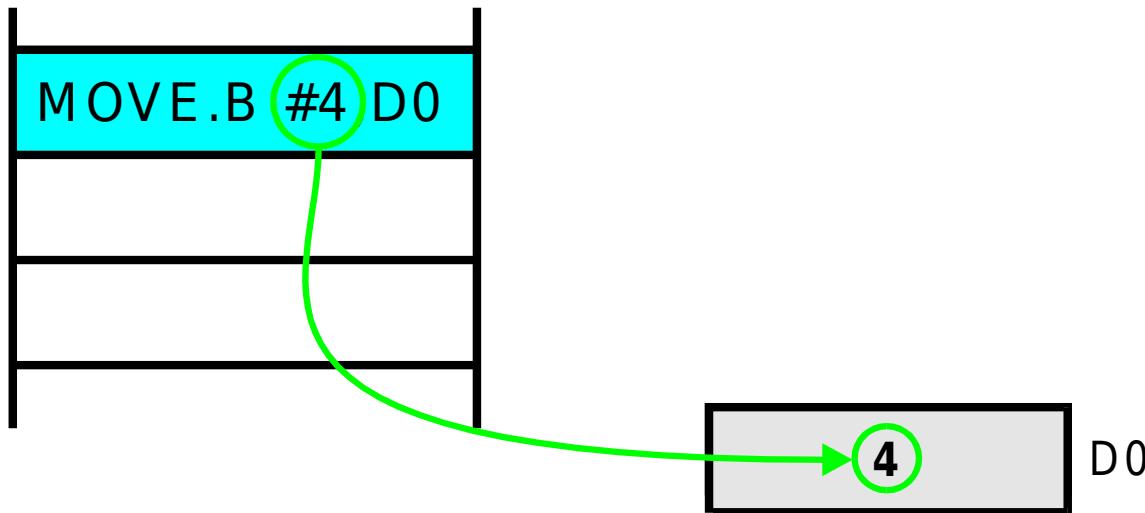
L'operando sorgente immediato è parte dell'istruzione

# Immediate Addressing - Funzionamento



L'operando destinazione è un registro dati

# Immediate Addressing - Funzionamento



L'effetto di questa istruzione è quello di copiare il valore della costante 4 nel registro dati D0

# Immediate Addressing: caratteristiche



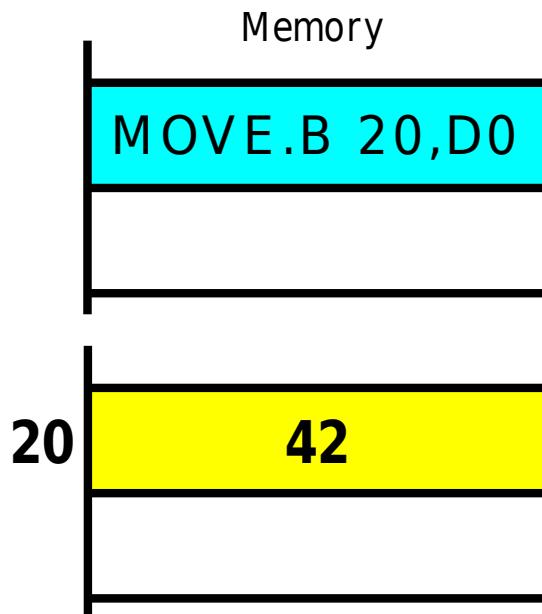
- Se la costante è “lunga”, è necessario usare una o più parole aggiuntive che seguono la parola codice (extra word)
- Se la costante da manipolare ha dimensioni ridotte (pochi bit) è possibile codificarla direttamente nei 16 bit dell’istruzione
  - non sono necessarie parole aggiuntive per codificare il literal oltre alla parola codice di 16 bit
  - non sono necessari ulteriori (lenti) accessi in memoria

# 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 azzerà il contenuto della locazione di memoria 1234

# Absolute Addressing - Funzionamento



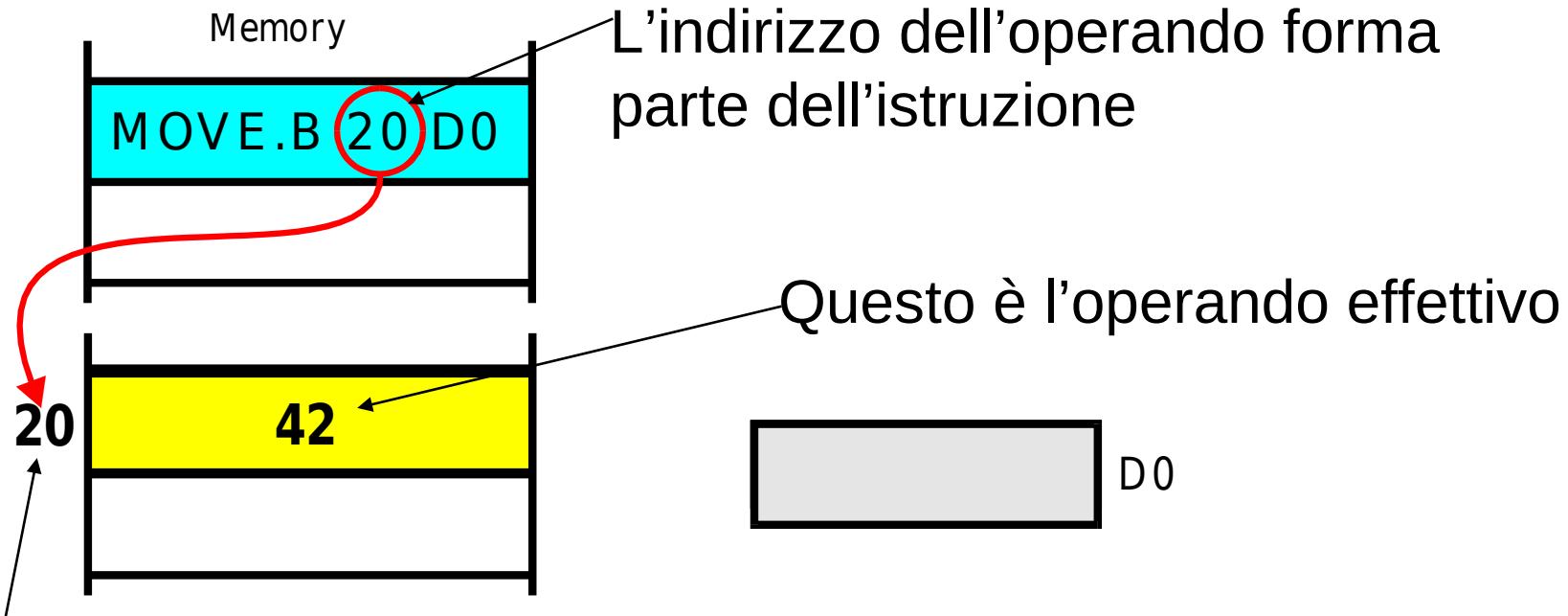
L'operando sorgente  
è in memoria

Questa istruzione ha un operando  
absolute



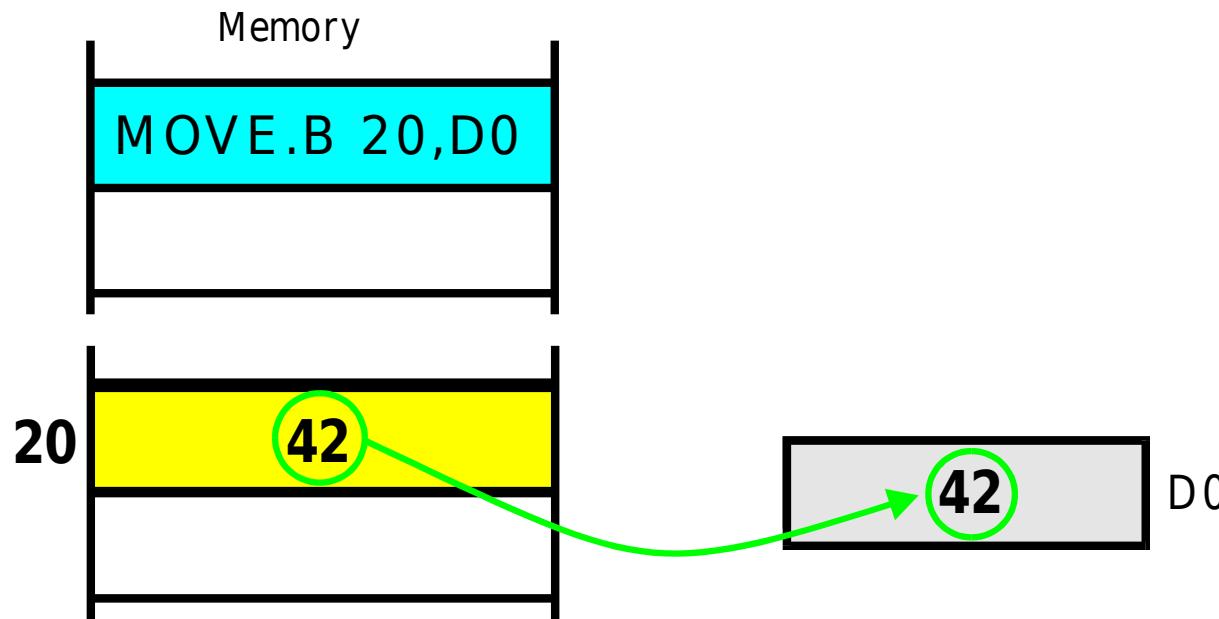
L'operando destinazione usa  
il direct addressing per un registro  
dati

# Absolute Addressing - Funzionamento



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

# MC68000: indirizzamento assoluto a 16 bit

- Il processore MC68000 presenta anche un modo di indirizzamento assoluto a 16 bit
  - Absolute Short
- L'indirizzo da 16 bit viene esteso su 32 bit con la tecnica di estensione del bit più a sinistra (impropriamente detto bit-segno)
- Supponendo di estendere un indirizzo di 16 bit con il MSB, individuare la regione dello spazio di indirizzamento a 32 bit acceduta

# Indirizzamento a 16 bit con estensione del MSB

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

0000	
0000000000000000	0000000000000000

7FFE	
0000000000000000	0111111111111110

- Gli indirizzi tra 8000 e FFFE vengono mappati sugli ultimi 32KB dello spazio di 4GB

8000	
1111111111111111	1000000000000000

FFFE	
1111111111111111	1111111111111110

# Esempio modi fondamentali



DIE  
TI.  
UNI  
NA

- 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
---	------	---	-----------------------

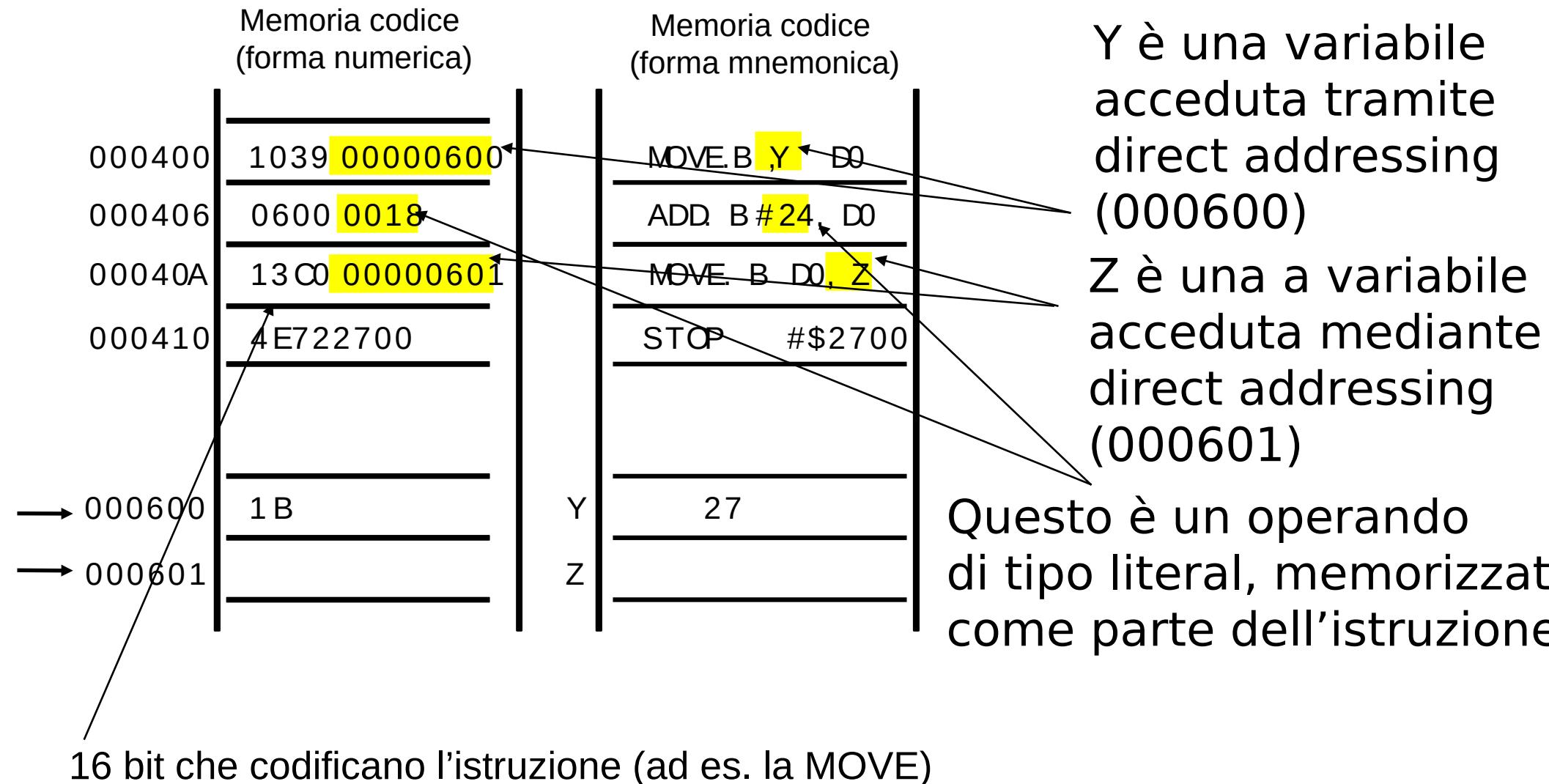
# II Programma Assemblato



DIE  
TI.  
UNI  
NA

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



# Riepilogo modi fondamentali



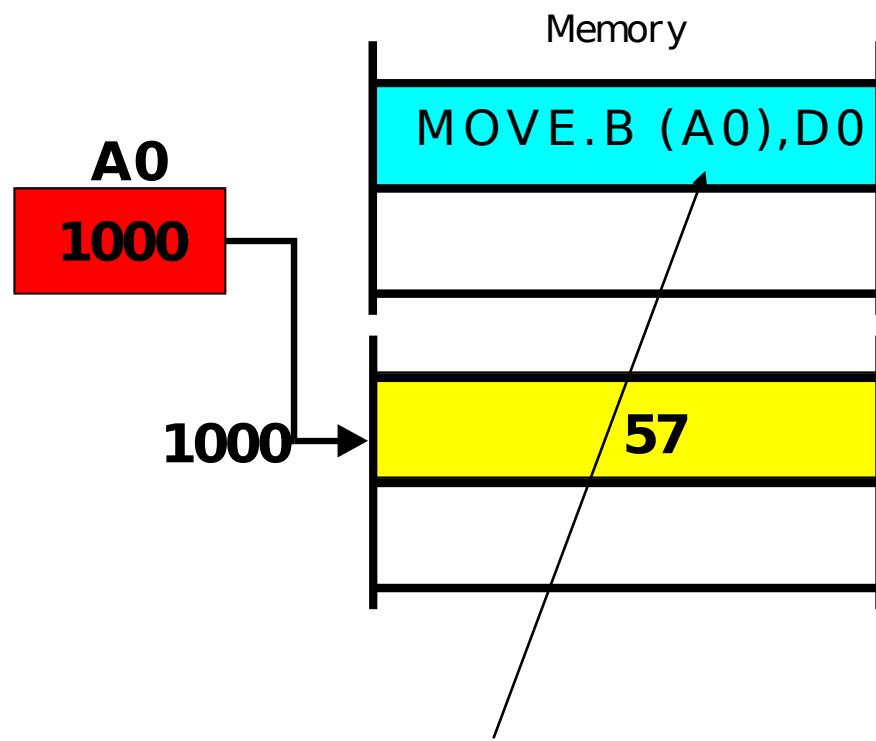
- **Register direct addressing** - È usato per variabili che possono essere mantenute in registri di memoria
- **Literal (immediate) addressing** - È usato per costanti che non cambiano
- **Direct (absolute) addressing** - È usato per variabili che risiedono in memoria

# 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

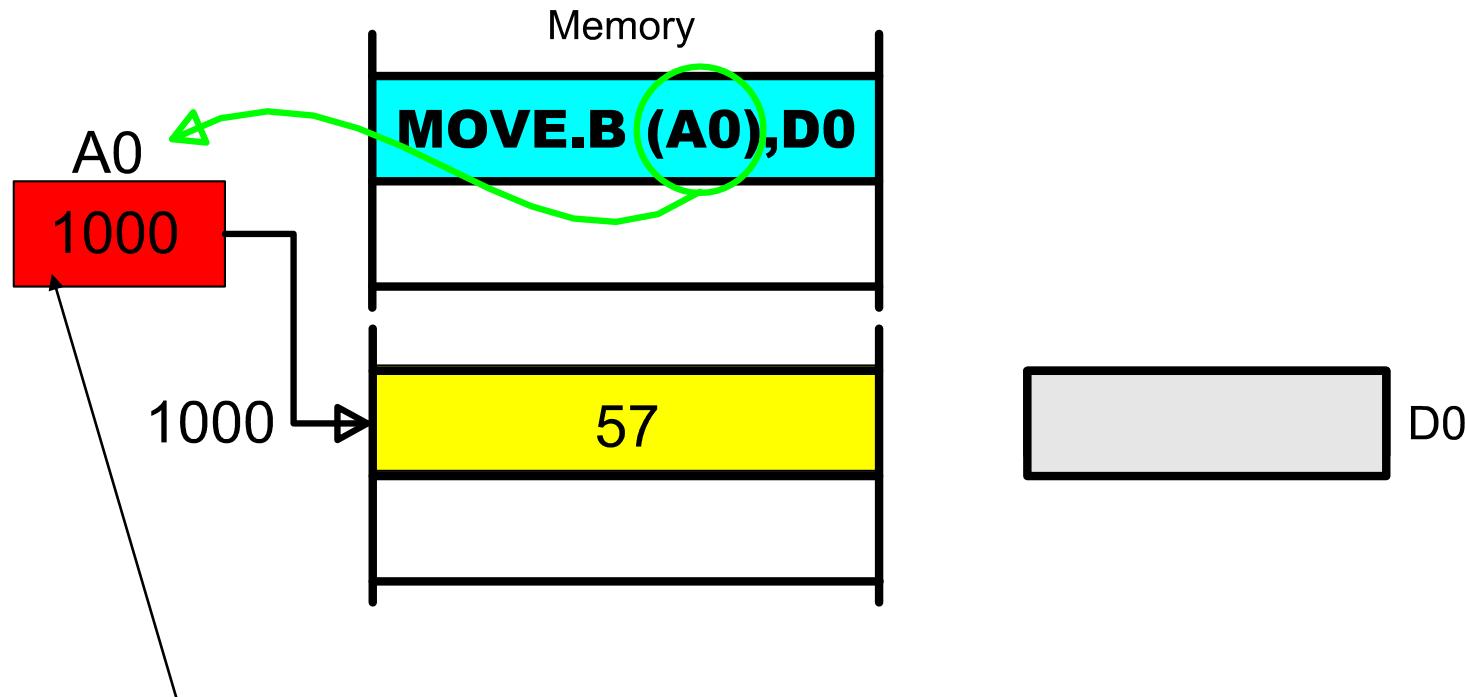


L'istruzione specifica l'operando sorgente come (A0)

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

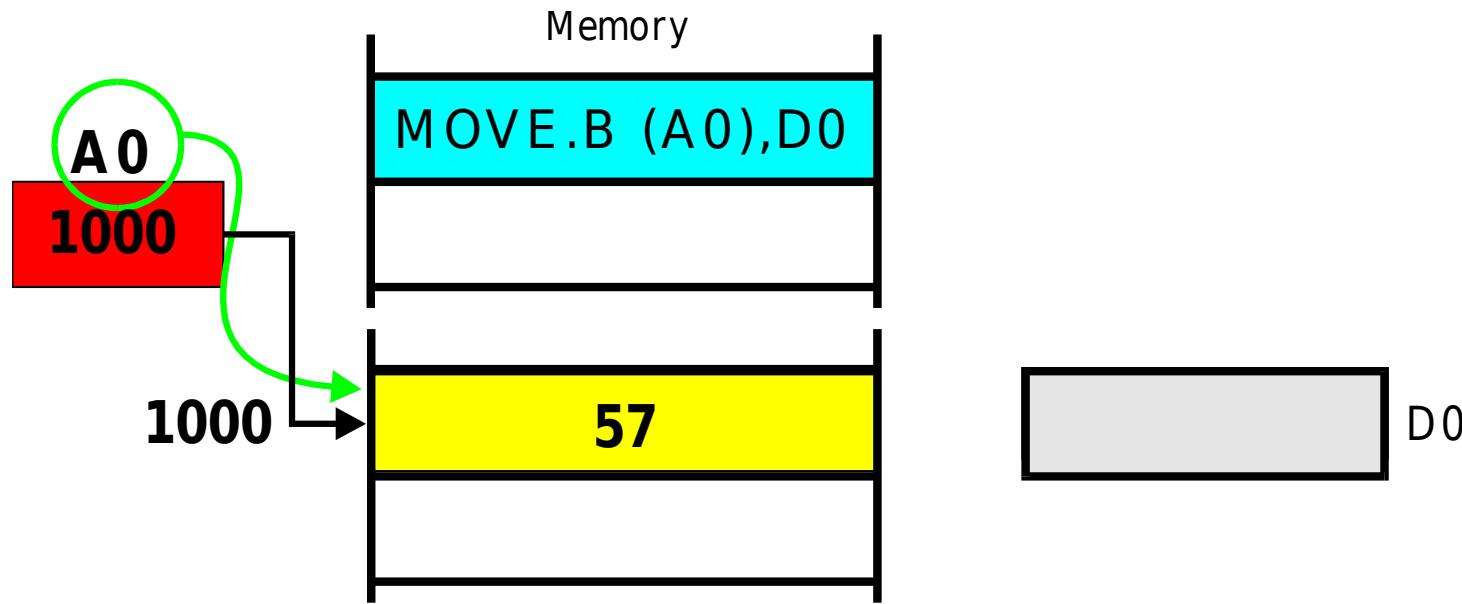


# Address Register Indirect: funzionamento



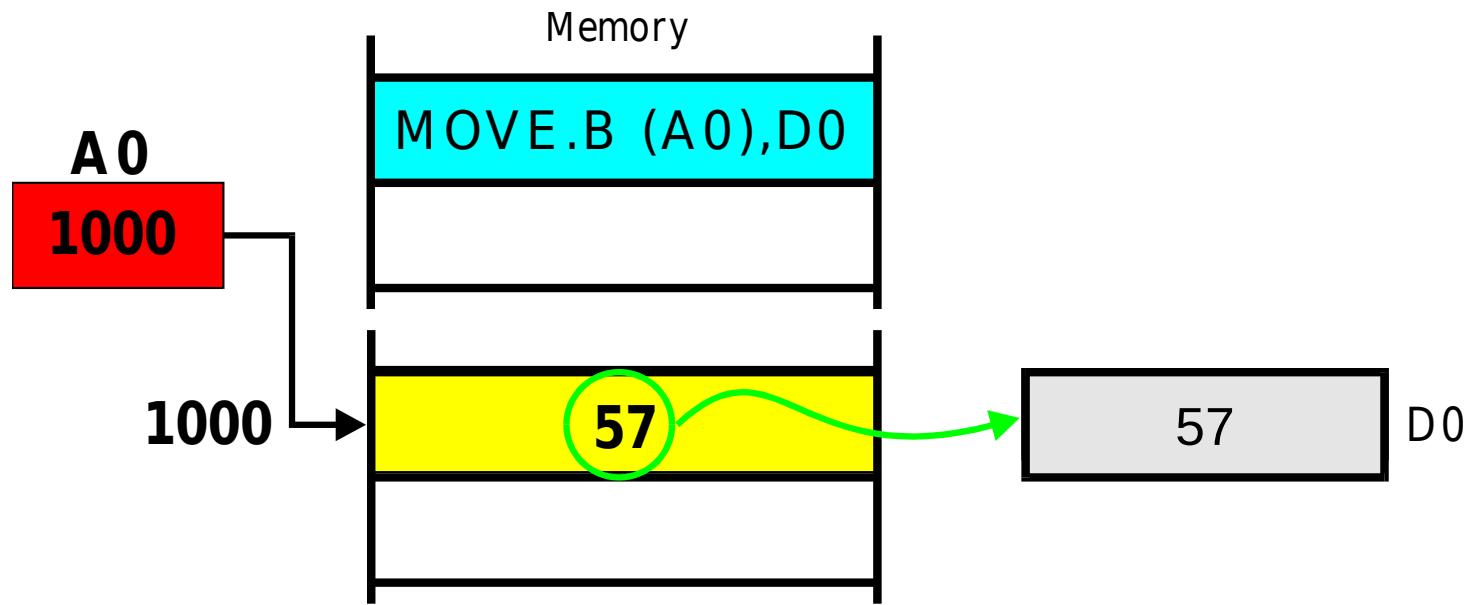
Il registro indirizzo nell'istruzione specifica un registro indirizzo che contiene l'indirizzo dell'operando

# 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-post-increment

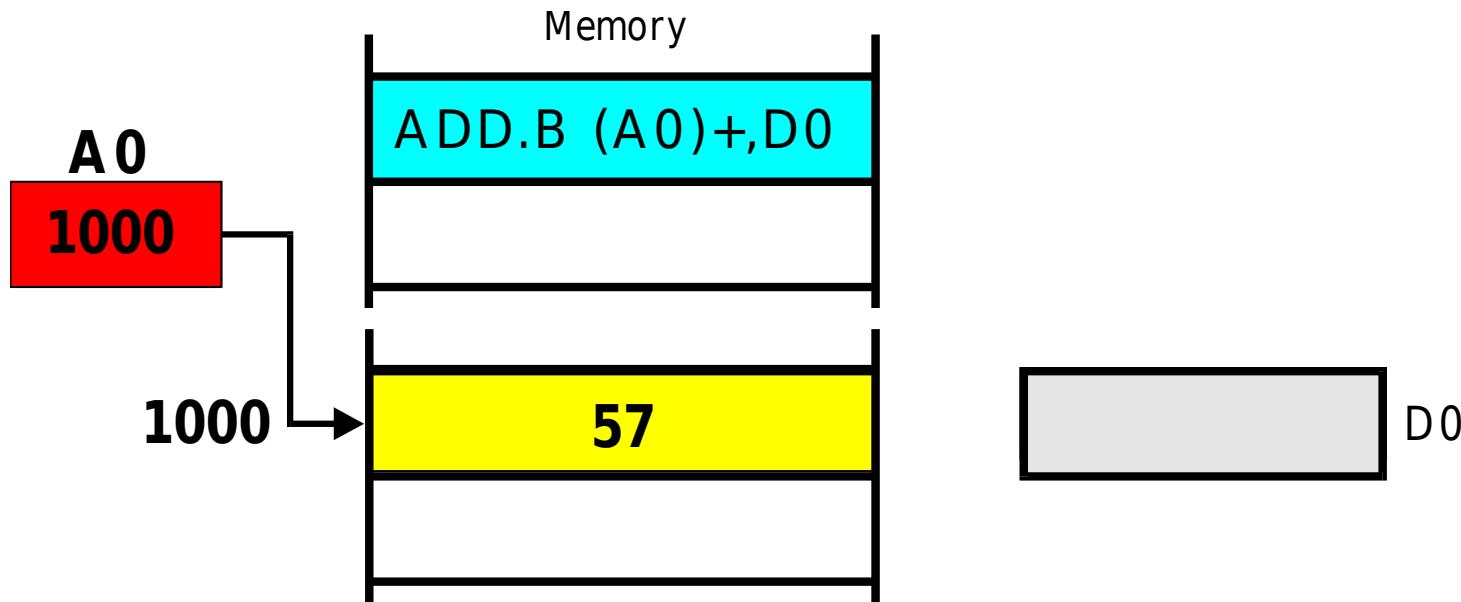


- L'istruzione specifica uno dei registri indirizzo, usando la modalità Address Register Indirect
- Se il modo di indirizzamento è specificato come (An)+, il contenuto del registro indirizzo è incrementato di una quantità pari alla dimensione dell'operando dopo l'uso ("post-incremento")
- Esempio:
  - 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 puntato da A0

# Auto-pre-decrement

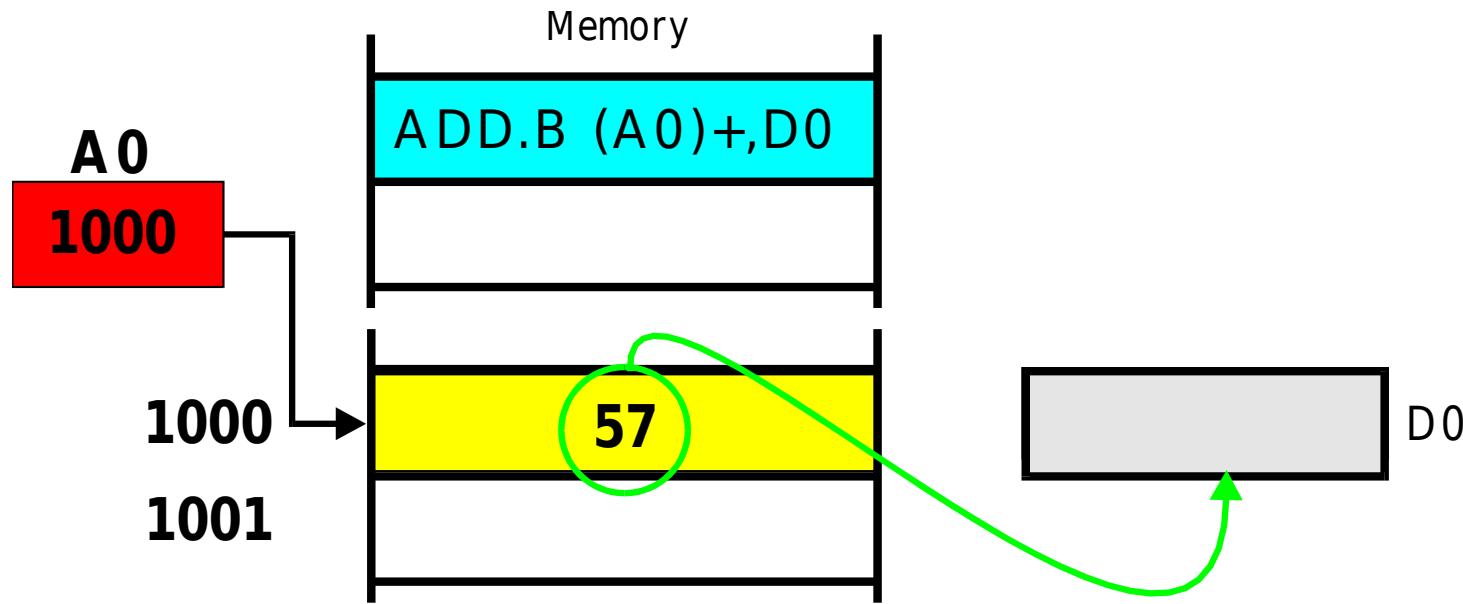
- L'istruzione specifica uno dei registri indirizzo
- Se il modo di indirizzamento è specificato come -(An), il contenuto del registro indirizzo è decrementato di una quantità pari alla dimensione dell'operando *prima dell'uso* ("pre-decremento")
- Esempio:
  - MOVE.W D0,-(A0) Sottrae 2 ad A0 e poi lo usa per la MOVE (2 poiché l'accesso è di tipo .W = 2 byte). Di fatto, l'istruzione esegue un push di D0 sullo stack puntato da A0.

# Auto-post-increment: funzionamento



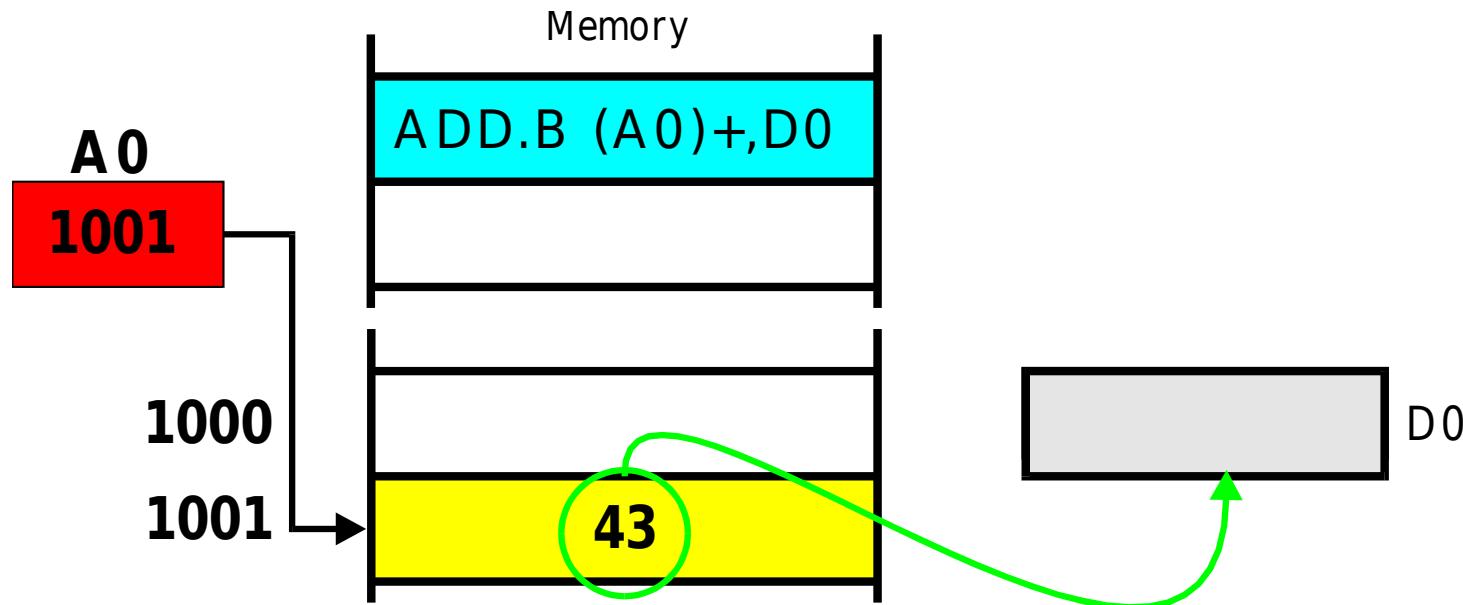
Il registro indirizzo contiene 1000  
ovvero “punta” alla locazione 1000

# Auto-post-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-post-increment: funzionamento



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

- Quando un programma deve accedere a dati in memoria può usare:
  - Indirizzamento assoluto: Es. MOVE \$8100,D0
  - Indirizzamento indiretto: Es. MOVE (A1),D0
- Vantaggio del modo indiretto: l'indirizzo è determinato a runtime, e la stessa istruzione (ad es. all'interno di un ciclo) può operare su dati posti in locazioni diverse
- Ci sono situazioni in cui un solo grado di libertà attraverso un registro An non è sufficiente
  - es. accedere ai campi di un array di record
- Soluzione: Metodi di indirizzamento che costruiscono l'EA mediante due o più componenti

# Based Addressing

- Based Addressing combina due componenti mediante somma, per formare l'EA:
  - 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 a campi di record di cui si conosca la posizione relativa ad assembly time, ma non quella iniziale
- Il processore MC68000 supporta il Based Addressing attraverso il modo **Indirect with displacement** d16(An)
- Es. MOVE.L 6(A0),D2

# Based Indexed Addressing



- 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
  - Anche detti **Indirect with displacement and index**



- d8(An,Xm) **Short Based Indexed**
  - d8 spiazzamento costante ad 8 bit in [-128,127]
  - An registro indirizzo
  - Xm è un qualunque registro D0-D7 o A0-A7 che svolge la funzione di registro indice e del quale si prendono solo i 16 bit meno significativi estesi a 32
- d16 (An,Xm.L) **Long Based Indexed**
  - d16 spiazzamento costante a 16 bit in [-32k,32k-1]
  - An registro indirizzo
  - Xm è un qualunque registro D0-D7 o A0-A7 che svolge la funzione di registro indice e del quale si considerano tutti i 32 bit