

# Corso di Calcolatori Elettronici I

---

## Modi di indirizzamento del processore MC68000 (parte prima)

Prof. Roberto Canonico

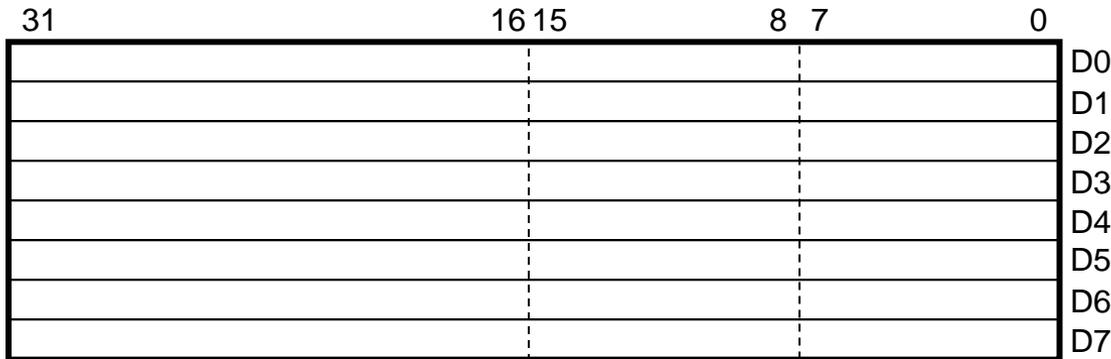


Università degli Studi di Napoli Federico II  
Facoltà di Ingegneria

---

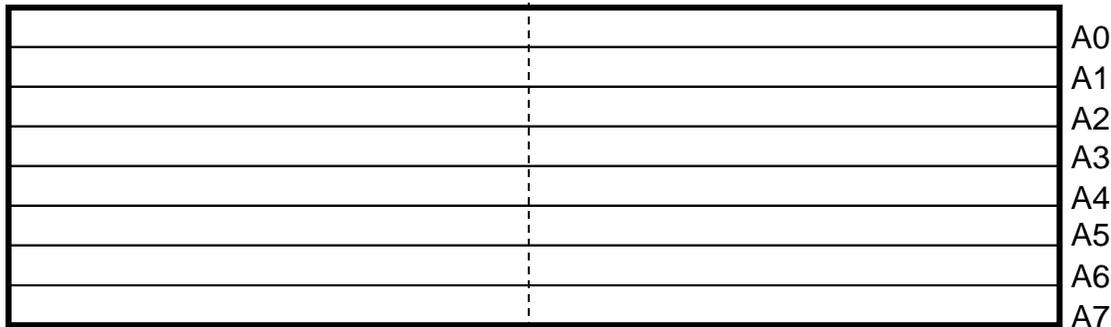
# Modello di programmazione del processore MC68000

---



## Registri dati

*utilizzabili come dati  
di 32 bit (Long Word),  
16 bit (Word) oppure  
8 bit (Byte)*



## Registri indirizzo

*utilizzabili per indirizzi  
di 32 bit (Long Word)  
oppure 16 bit (Word)*

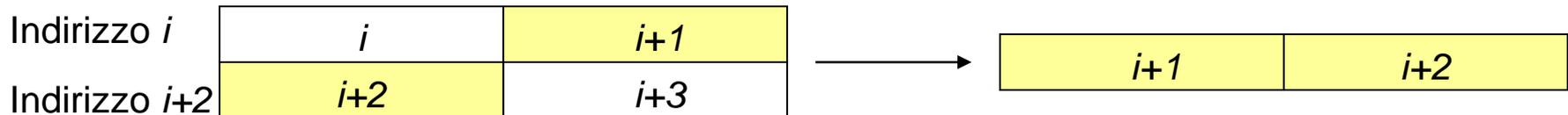


2 1 0

# Memoria:

## parole allineate e non allineate

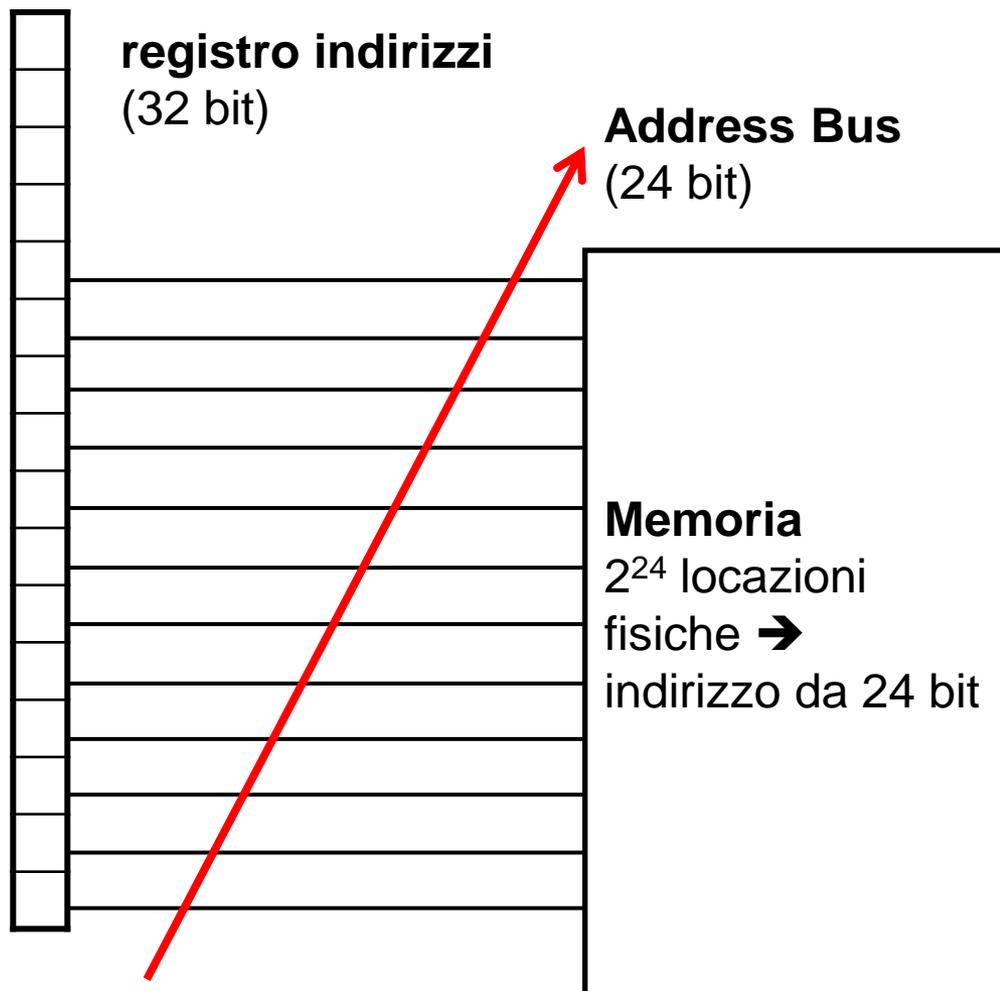
- Per un processore a parola di 16 bit o 32 bit, una *parola* che inizia ad un indirizzo pari si dice “allineata sul limite di parola”
- Tipicamente, un processore è in grado di accedere ai due byte che costituiscono una parola allineata mediante una sola operazione di lettura
- Il processore Intel 8086 consente l’accesso a parole non allineate, cioè parole che iniziano ad un indirizzo dispari, ma in tal caso sono necessari 2 distinti accessi in memoria
- Il 68000 NON consente l’accesso a parole non allineate



( $i$  pari)

La parola di 16 bit formata dai due byte **ombreggiati**  
**non è allineata sul limite di parola** (indirizzo multiplo di 2)

# Parallelismo dell'Address Bus e dimensione dei registri indirizzo



- Parallelismo bus 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*
    - Ad es.: \$0000A3B2 e \$0A00A3B2,  
poiché differiscono solo per gli 8 bit più significativi
  - 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

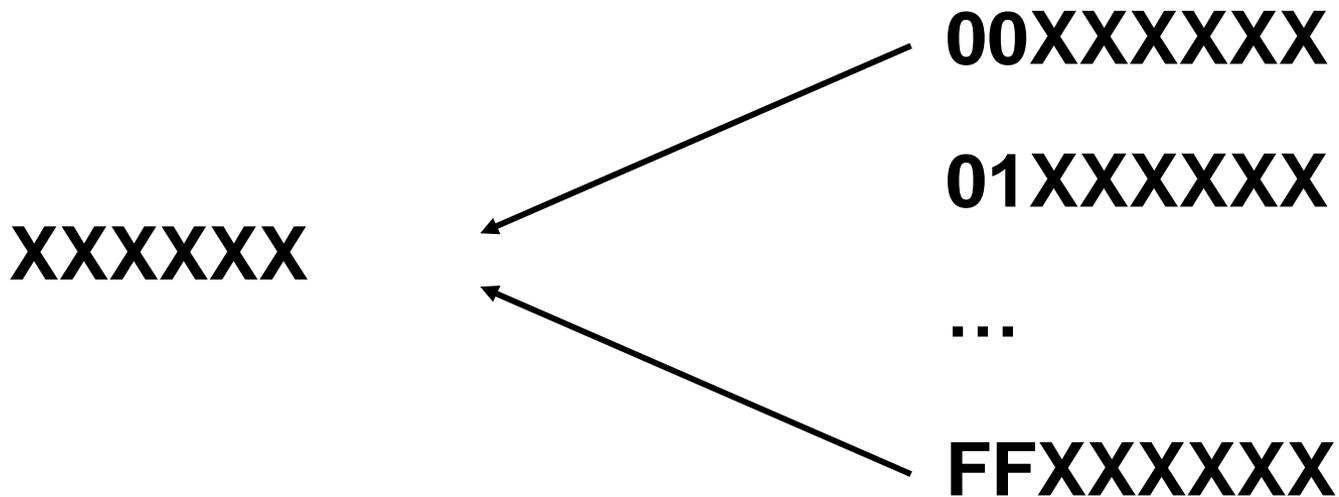
---

- 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 diversi modi di indirizzamento, in particolare per accedere ad operandi di tipo memoria
    - Il processore MC68000 ne supporta un numero notevole
-

# 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

---

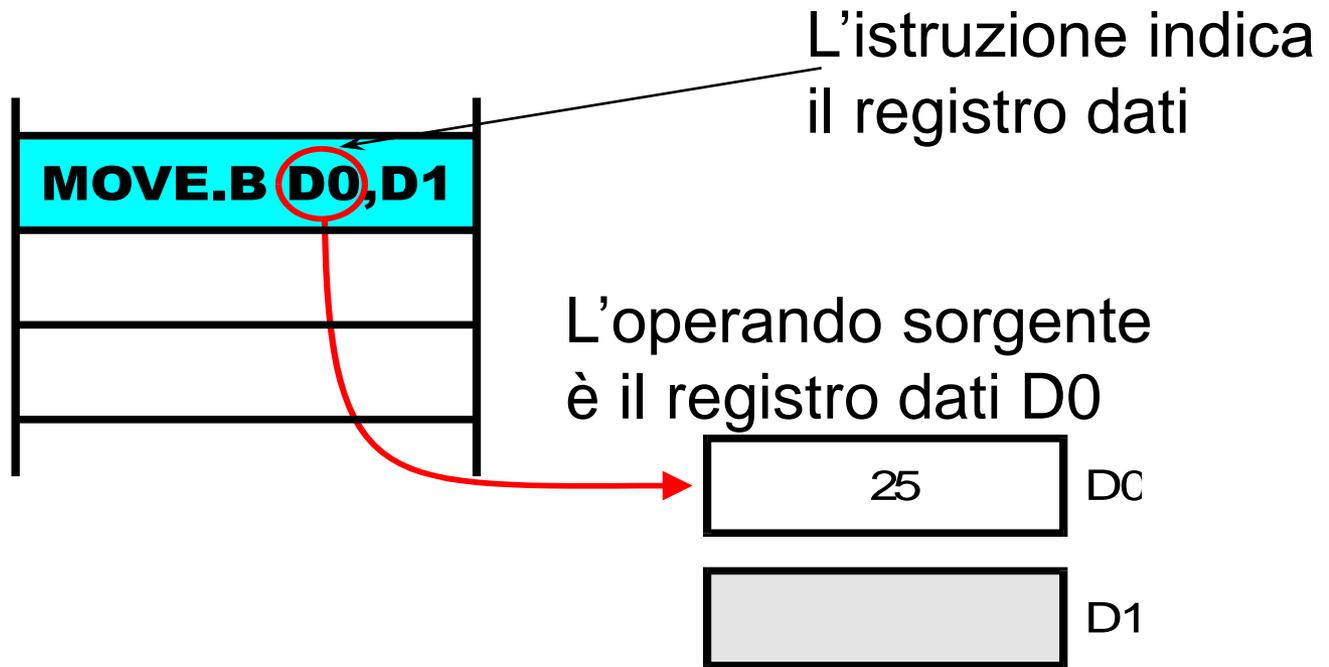
- È il modo di indirizzamento più semplice
- La sorgente o la destinazione di un operando è un registro dati o un registro indirizzi
- Se il registro è un operando sorgente, il contenuto del registro specificato fornisce l'operando sorgente
- Se il registro è un operando destinazione, esso viene caricato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

MOVE.B D0,D3  
SUB.L D3,A0  
CMP.W D2,D0  
ADD D3,D4

Copia l'operando sorgente in D0 nel registro D3  
calcola [A0] - [D3] e risultato in A0  
Confronta i valori dei registri D2 e D0 ([D0]-[D2])  
Somma il contenuto di D3 e D4 e risultato in D4

# Register Direct Addressing

---



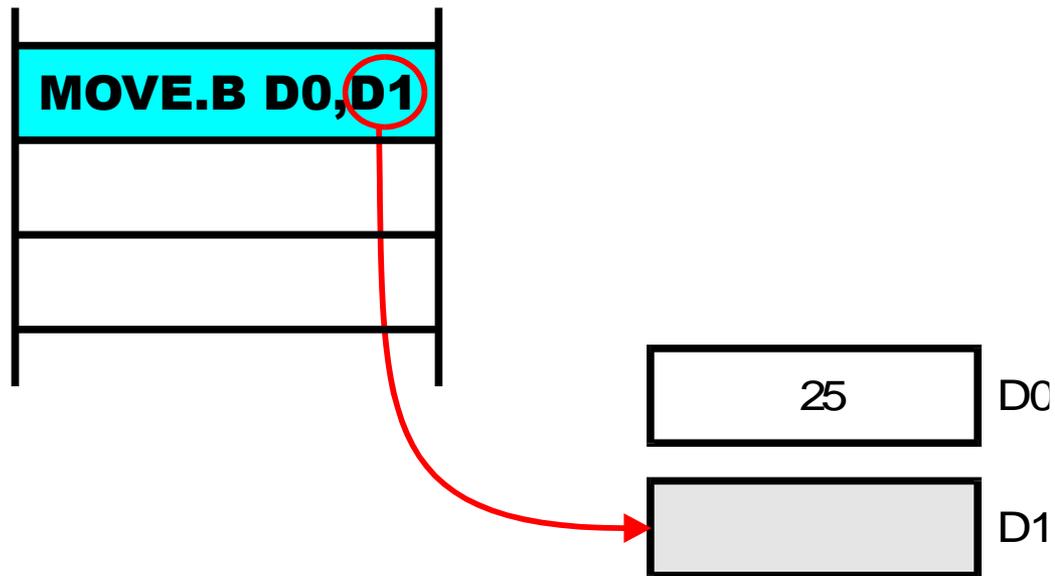
L'istruzione MOVE.B D0,D1 usa registri dati sia per l'operando sorgente che per quello destinazione

---

# Register Direct Addressing

---

---



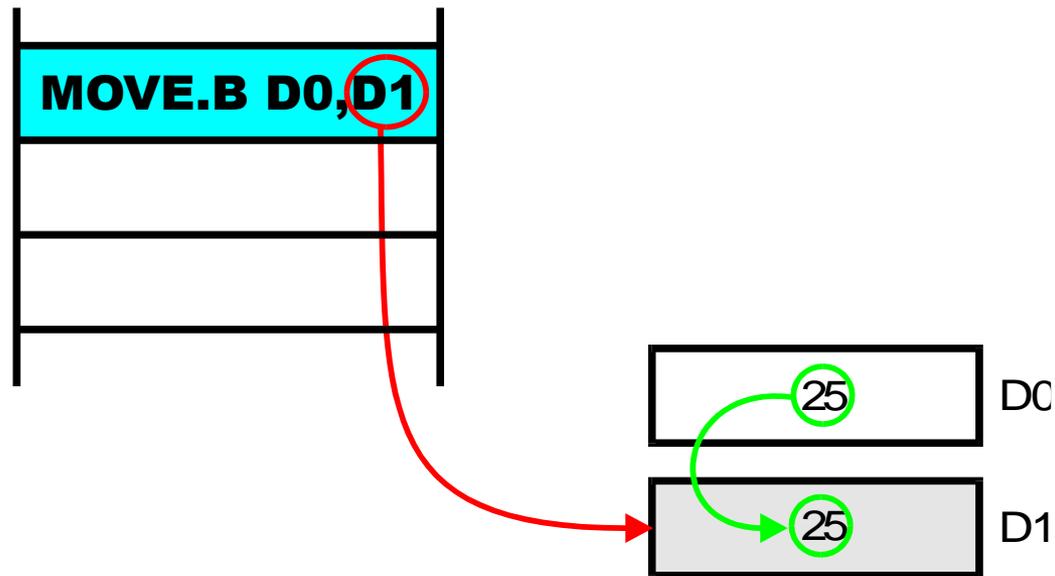
L'operando destinazione  
è il registro dati D1

---

# 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 programmatori lo usano per memorizzare variabili che sono usate di frequente
-

# Immediate Addressing

---

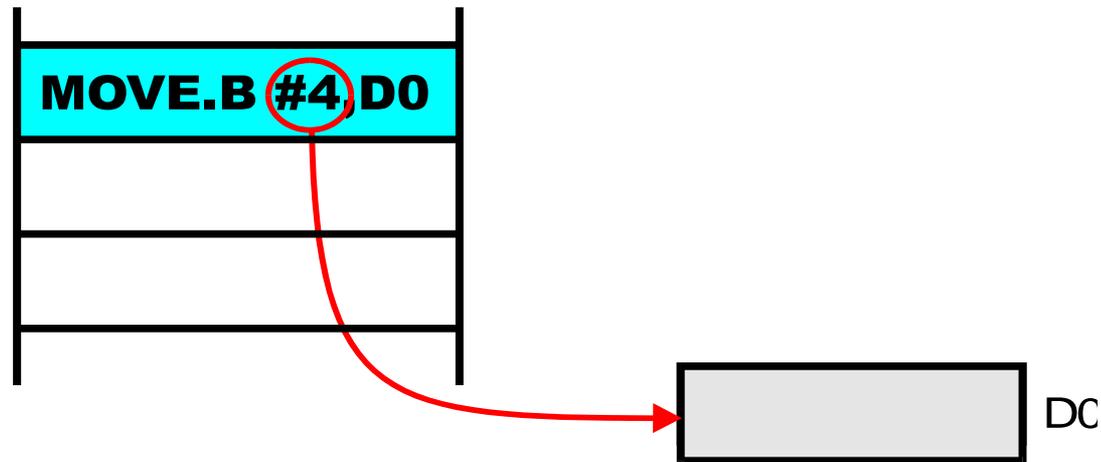
- L'operando effettivo costituisce parte dell'istruzione
  - Può essere usato unicamente per specificare un operando sorgente (non si può scrivere su una costante!)
  - È indicato da un simbolo # davanti all'operando sorgente
  - Un operando immediato è anche chiamato *literal*
- Esempio:

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

---

# Immediate Addressing - Funzionamento

---



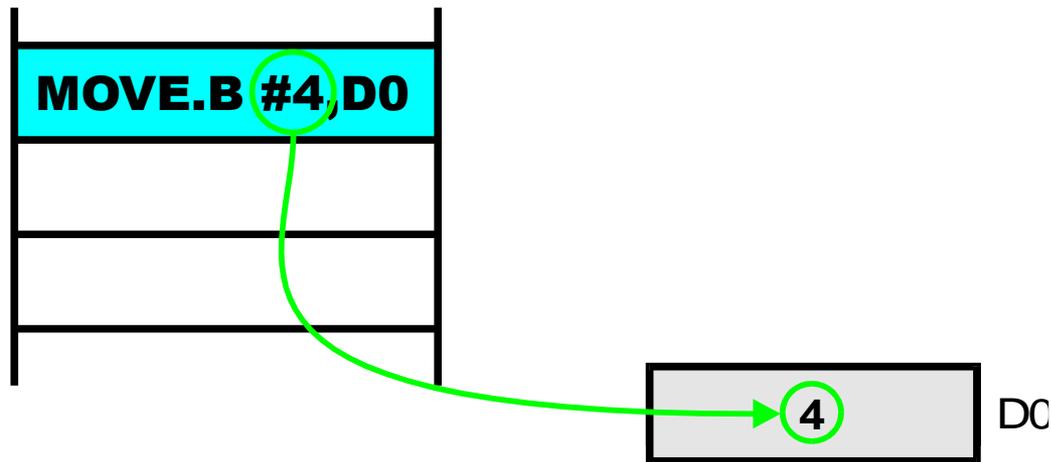
L'istruzione `MOVE.B #4, D0` usa un operando sorgente immediato ed un operando destinazione register direct

---

# Immediate Addressing: funzionamento

---

---



L'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 necessarie ulteriori (lenti) accessi in memoria
-

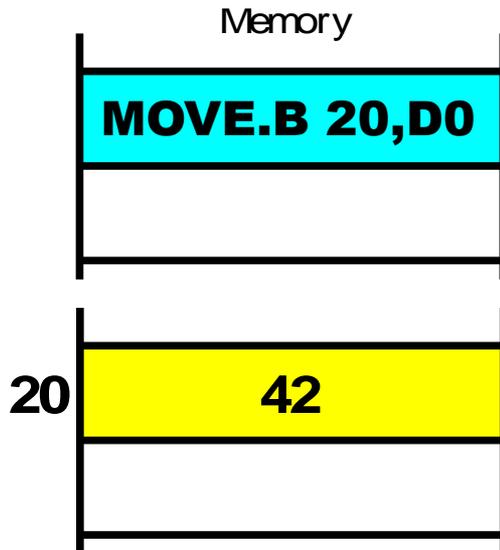
# Absolute Addressing

---

- 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

---



L'operando sorgente  
è in memoria

Questa istruzione ha un operando  
absolute



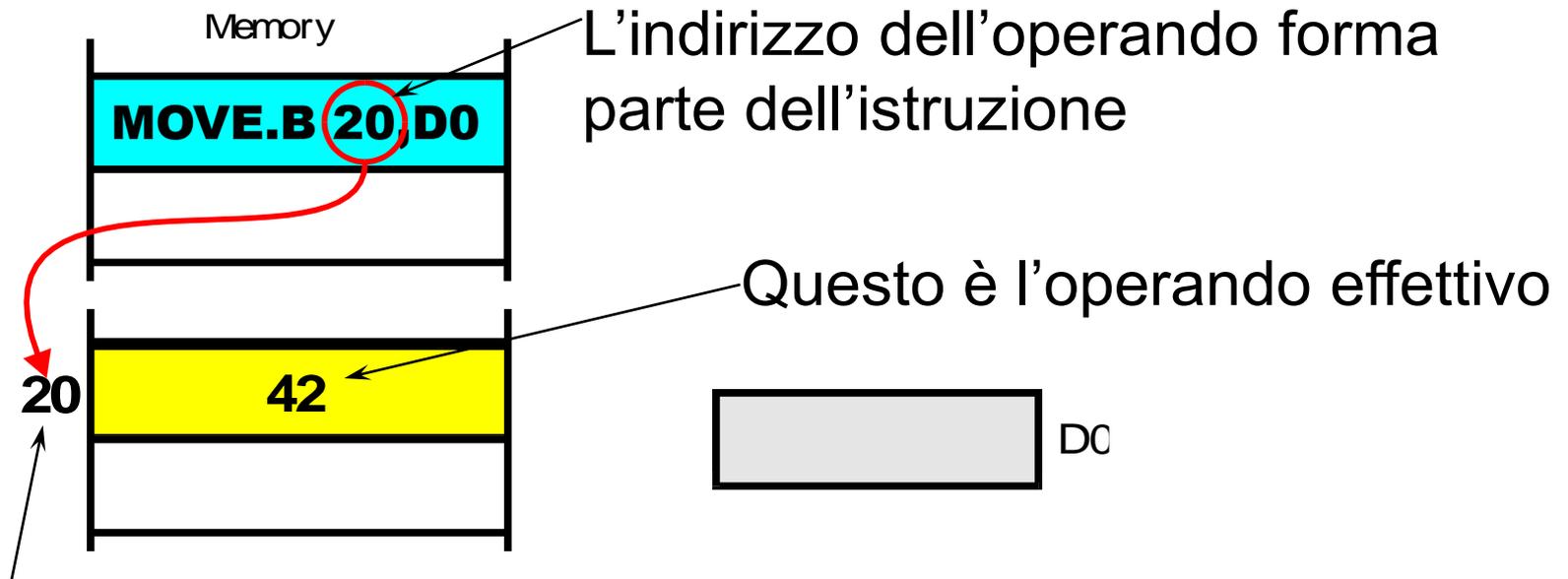
L'operando destinazione usa  
il direct addressing per un registro  
dati

---

# Absolute Addressing: funzionamento

---

---



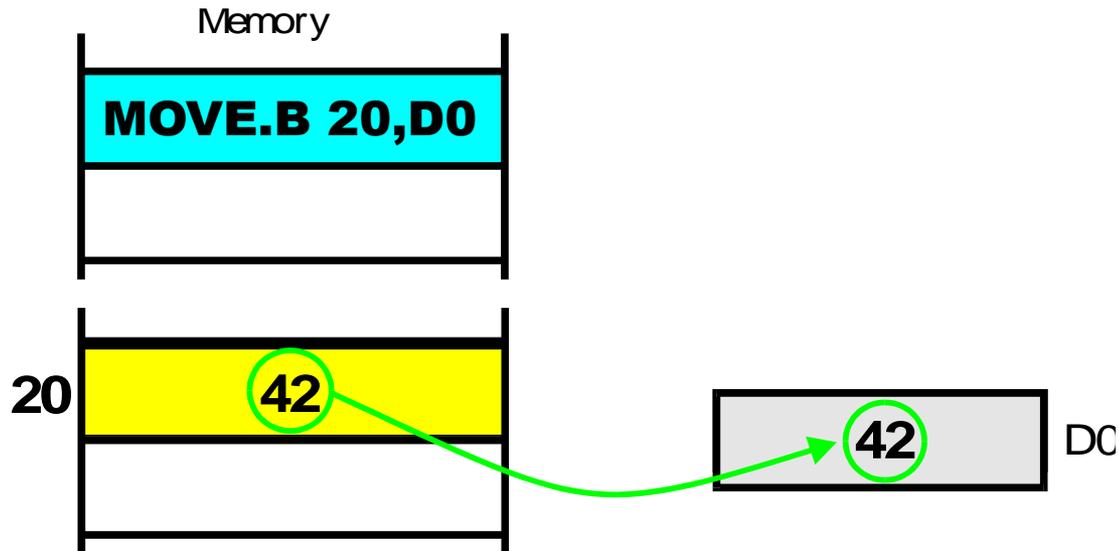
Una volta che la CPU ha letto l'indirizzo dell'operando dall'istruzione, la CPU accede all'operando effettivo

---

---

# 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 a assoluto 16 bit

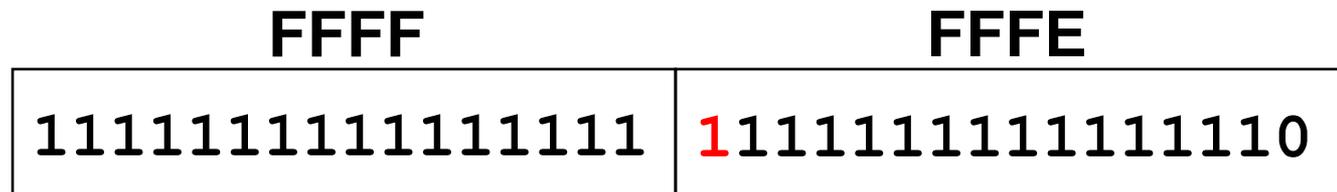
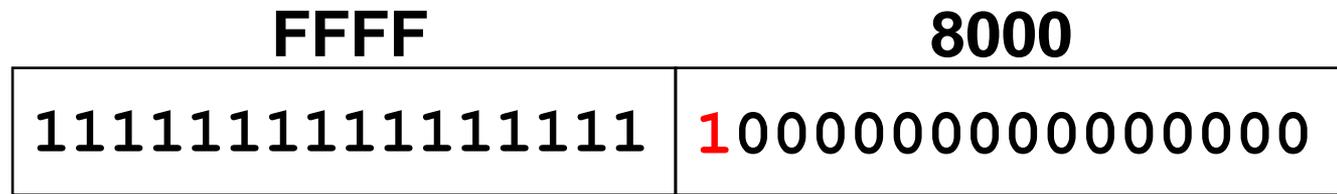
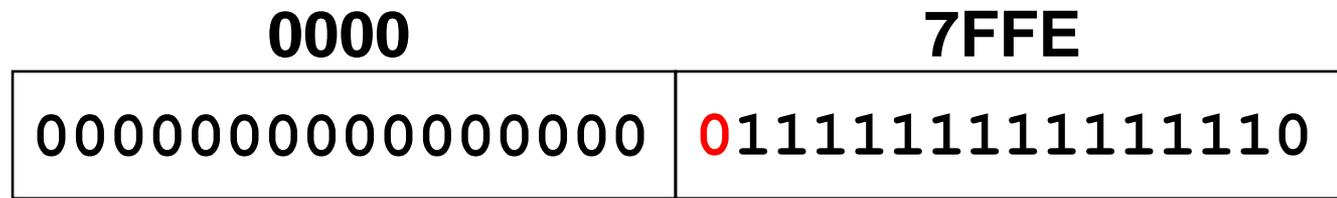
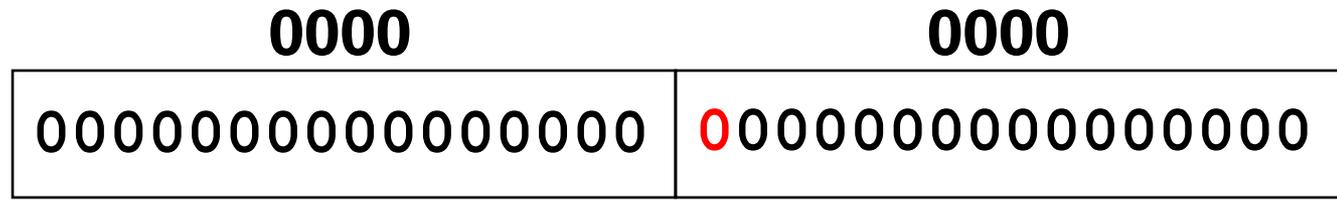
---

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



# 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.B    #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
```

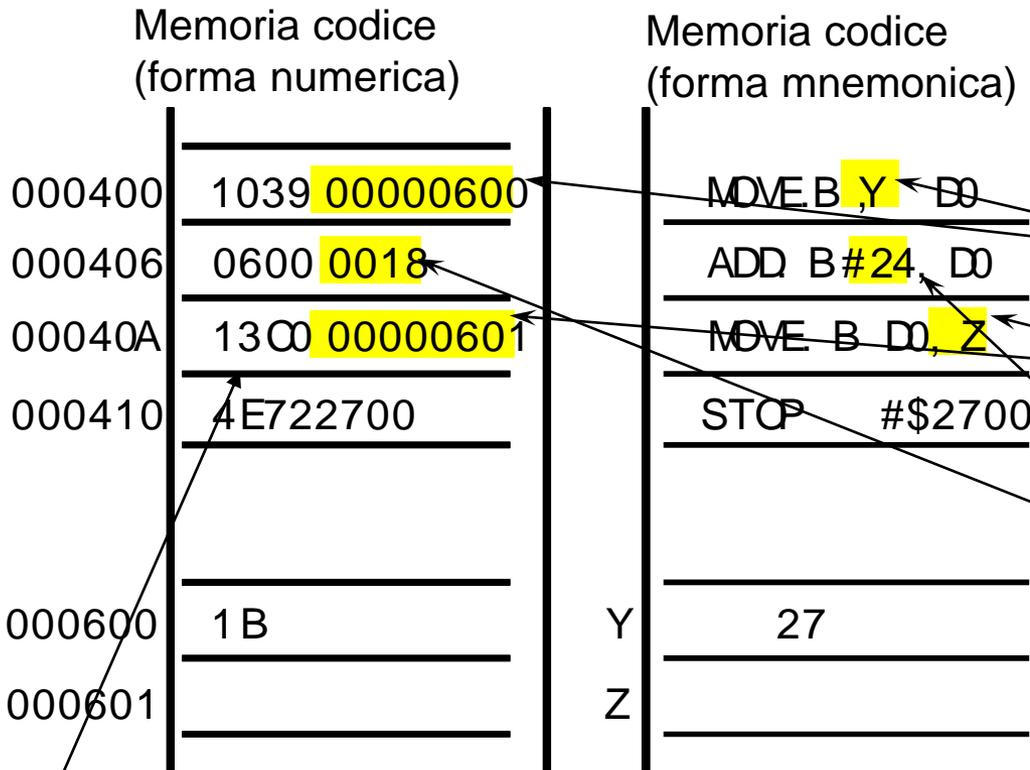
# Il Programma Assemblato

---

```
1      00000400                                ORG      $400
2      00000400 1039000000600                 MOVE.B   Y, D0
3      00000406 060000018                     ADD.B    #24, D0
4      0000040A 13C0000000601                 MOVE.B   D0, Z
5      00000410 4E722700                       STOP     #$2700
6
6      *
7      00000600                                ORG      $600
8      00000600 1B                             Y        DC.B    27
9      00000601 000000001                       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
-