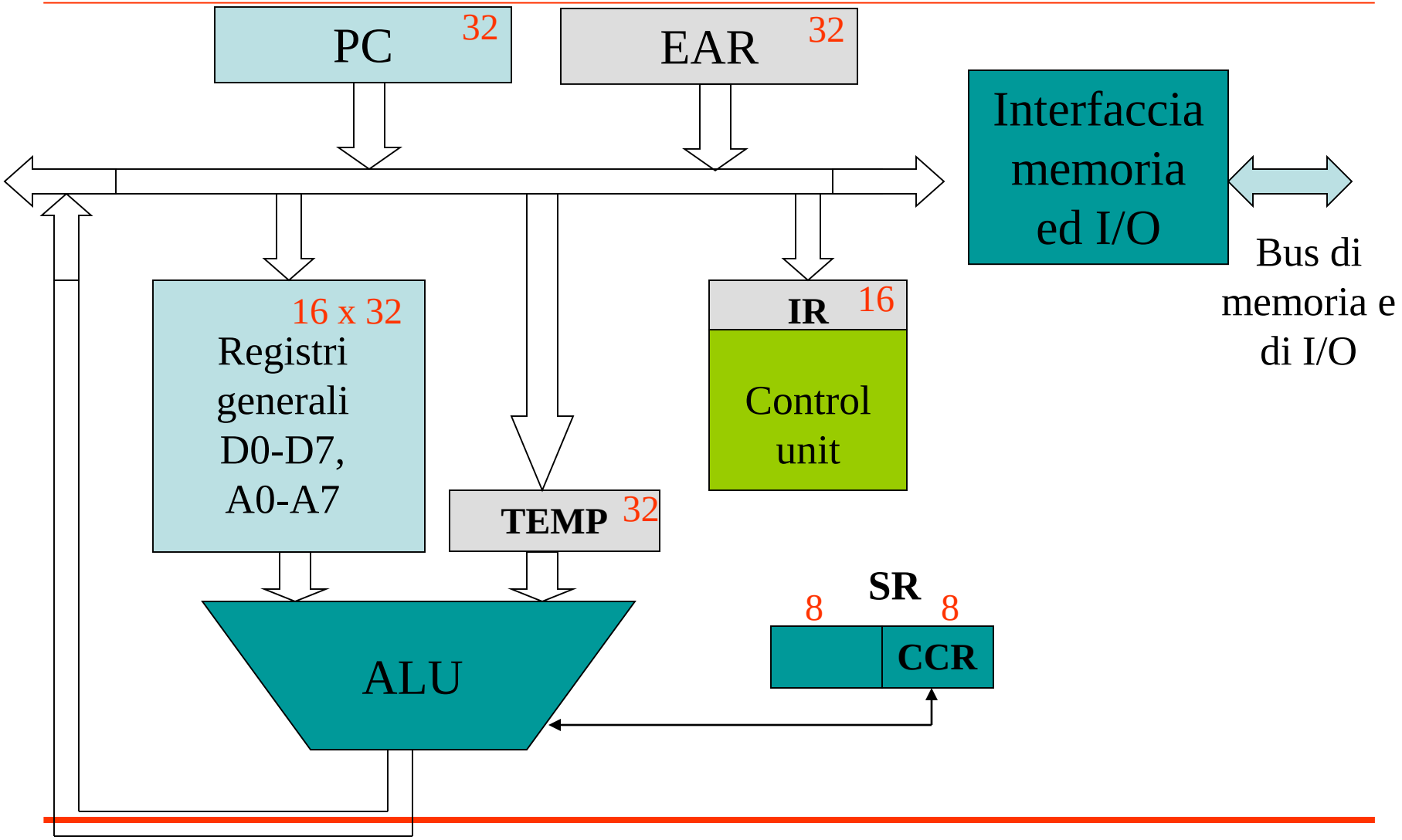
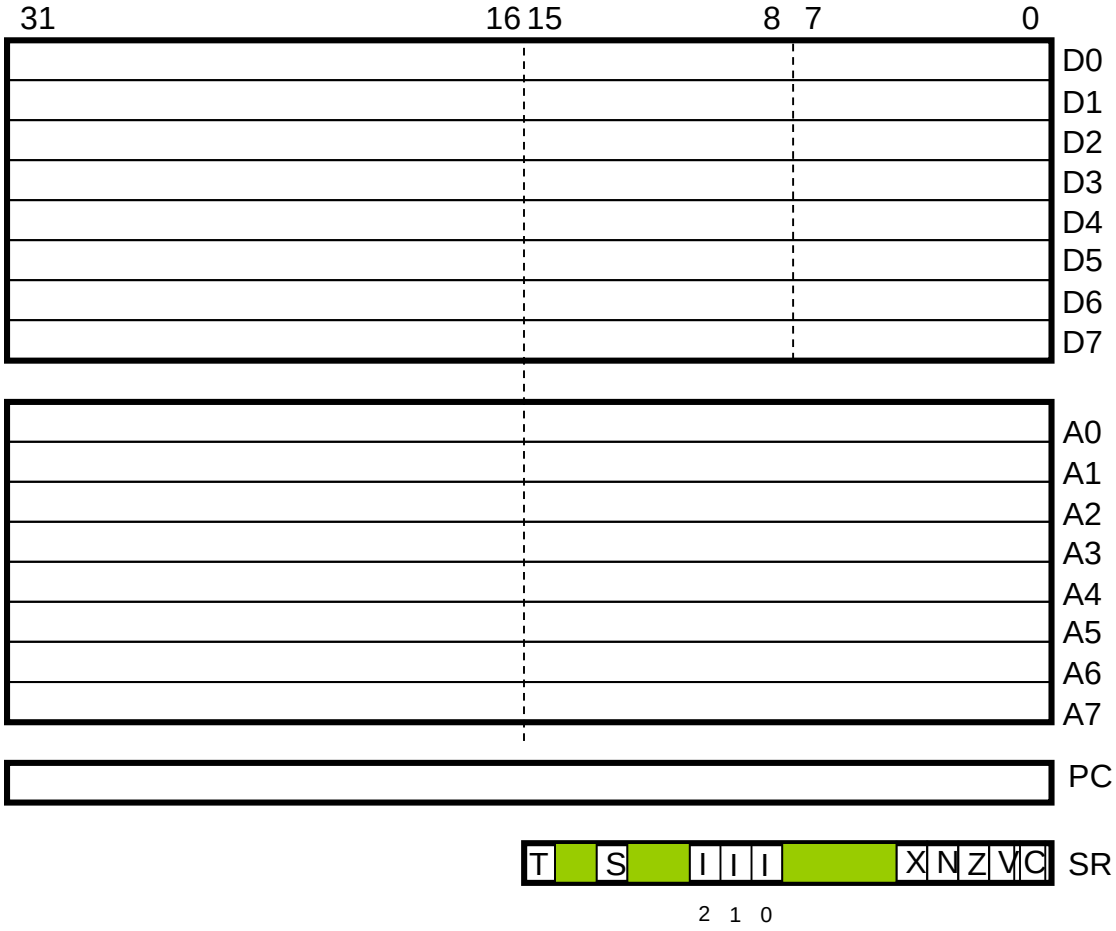
Architettura del processore MC 68000



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)

Program counter

Registro di stato

Registro di stato (Status register)

- Contiene:
  - La interrupt mask (8 livelli)
  - I codici di condizione (CC) - oVerflow (V), Zero (Z), Negative (N), Carry (C), e eXtend (X)
  - Altri bit di stato - Trace (T), Supervisor (S)
- I Bits 5, 6, 7, 11, 12, e 14 non sono definiti e sono riservati per espansioni future

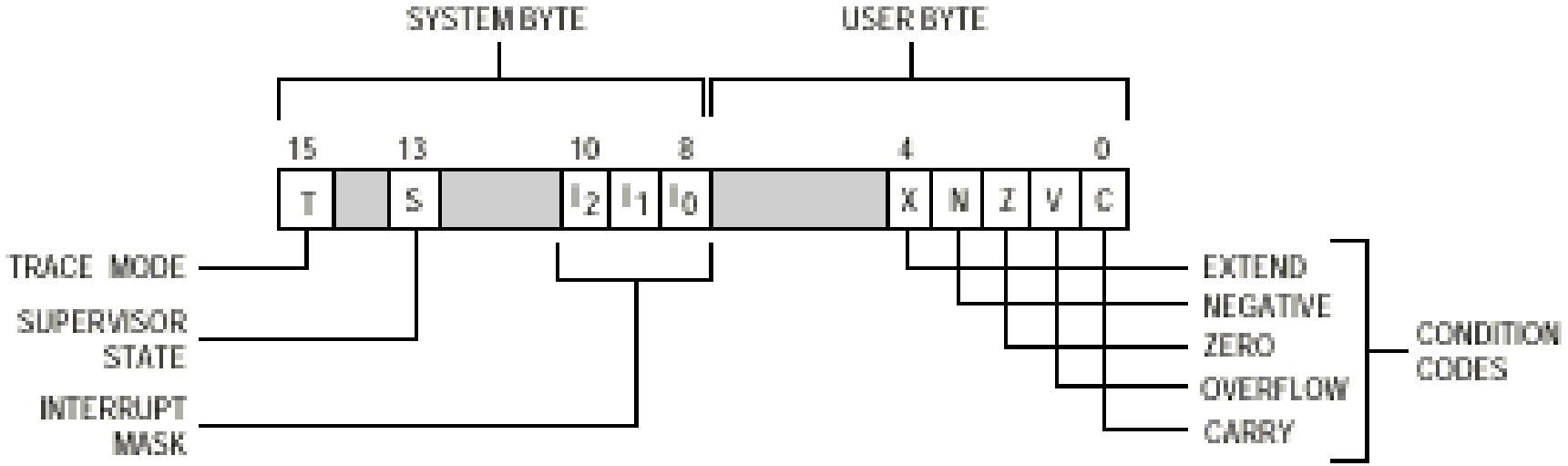
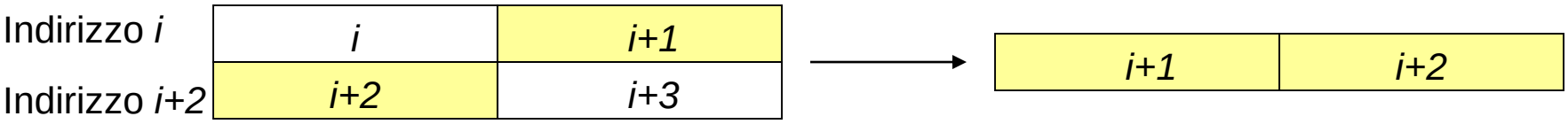


Figure 2-4. Status Register

Memoria: parole allineate e non allineate

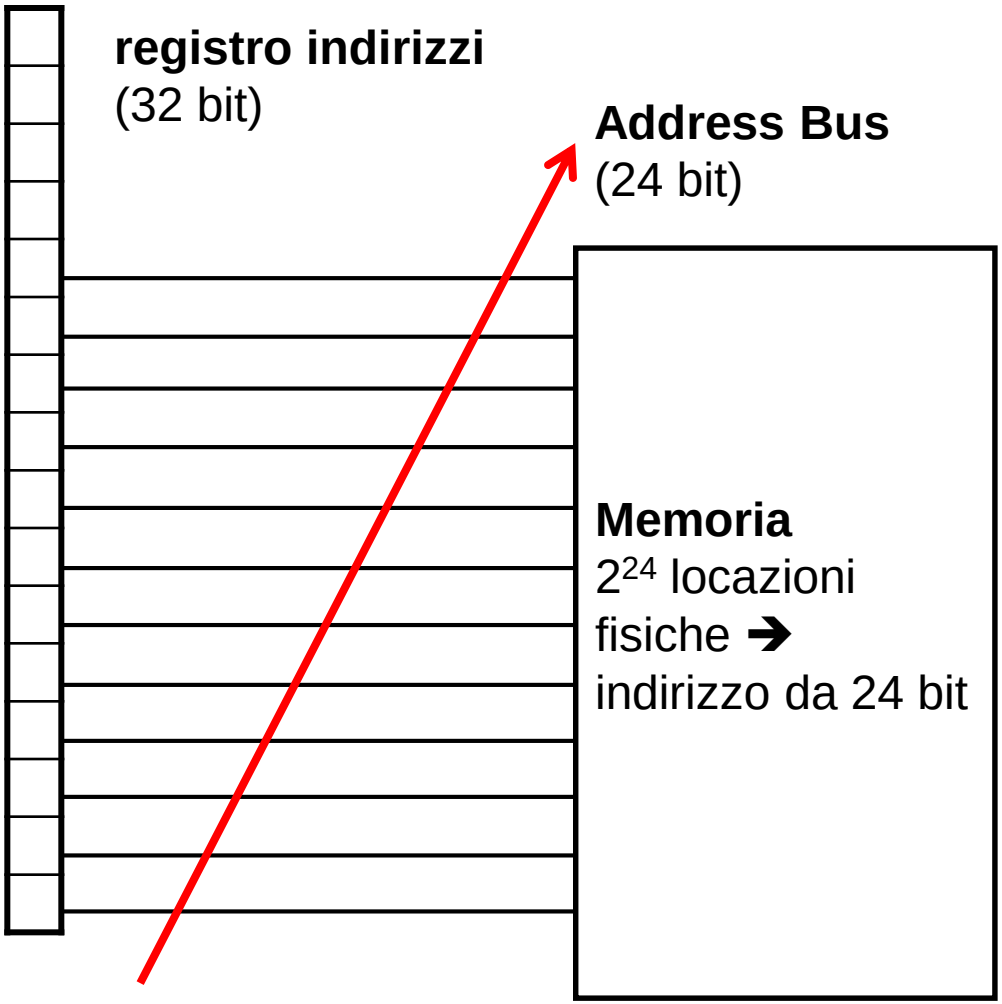
- Per un processore a parola di 16 bit o 32 bit, una *parola* che inizia ad un indirizzo pari si dice “allineata sul limite di parola”
- Tipicamente, un processore è in grado di accedere ai due byte che costituiscono una parola allineata mediante una sola operazione di lettura
- Il processore Intel 8086 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



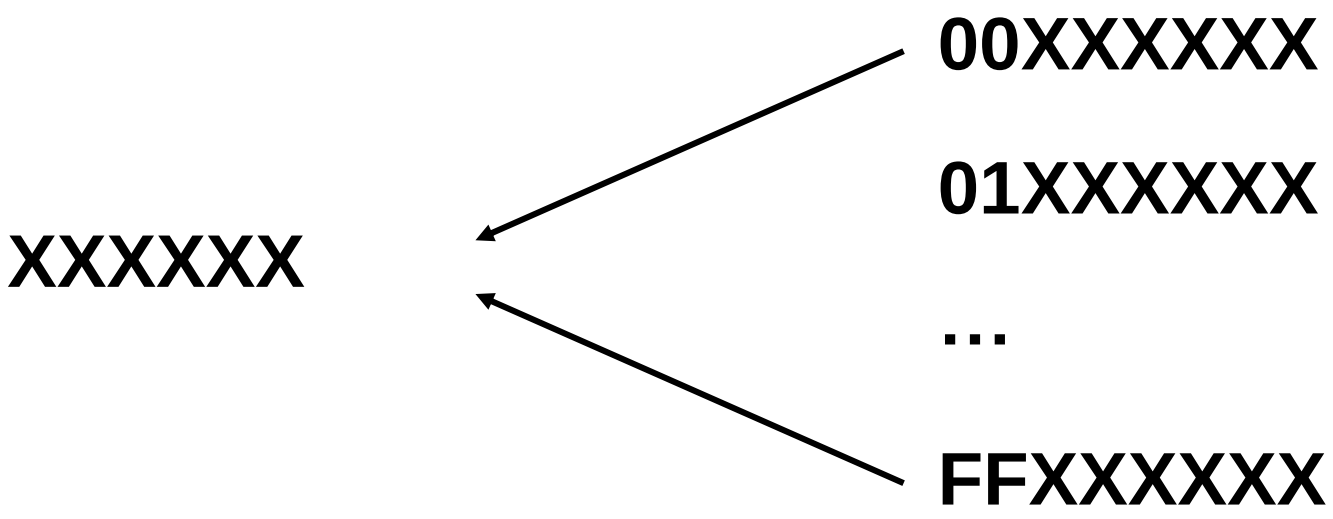
- Parallelismo bus indirizzi: determina il numero di indirizzi "fisici" distinti che la CPU è in grado di generare all'esterno
- Dimensione registri indirizzo (es. A0, PC): determina il numero di indirizzi "logici" distinti che la CPU può trattare nei programmi
- Non è detto che le due dimensioni coincidano
- 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**

Aliasing nel MC68000

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



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)

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

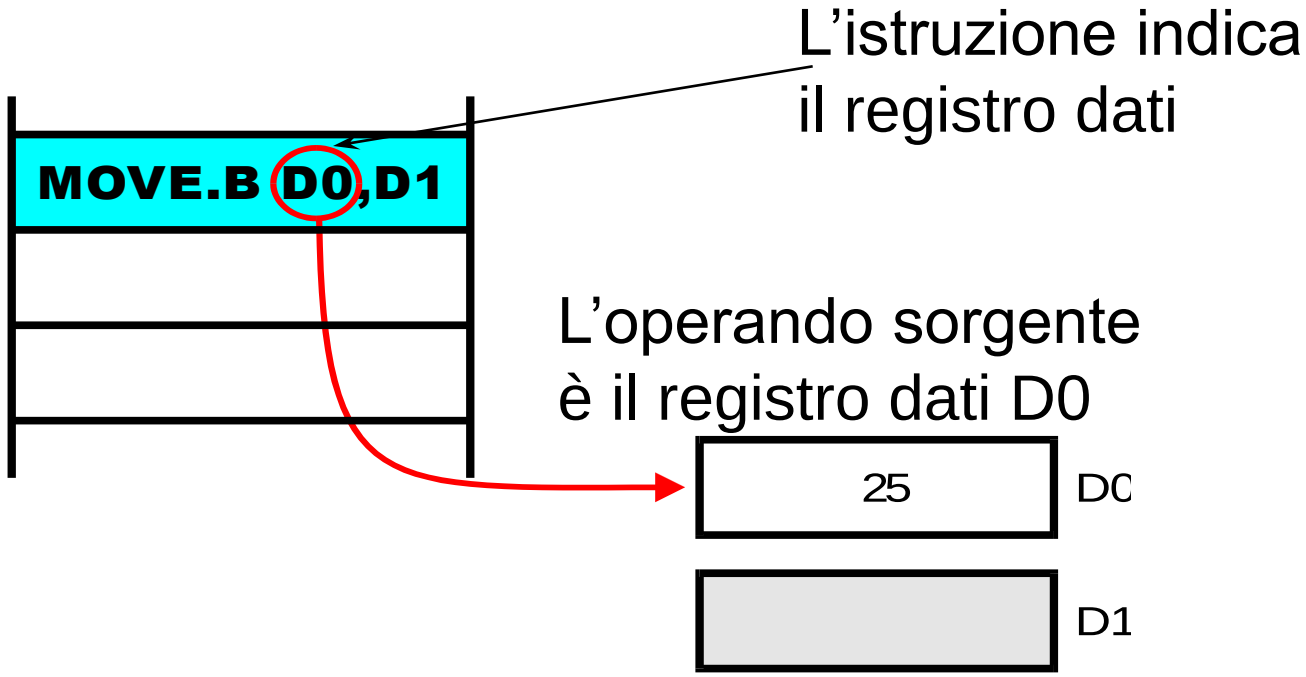
- Register Direct
  - Data-register Direct
  - Address-register Direct
- Immediate (or Literal)
- Absolute
  - Short (16 bit)
  - Long (32 bit)
- Address-register Indirect
  - Auto-Increment
  - Auto-Decrement
- Indexed short
- Based
- Based Indexed
  - Short
  - Long
- Relative
- Relative Indexed
 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

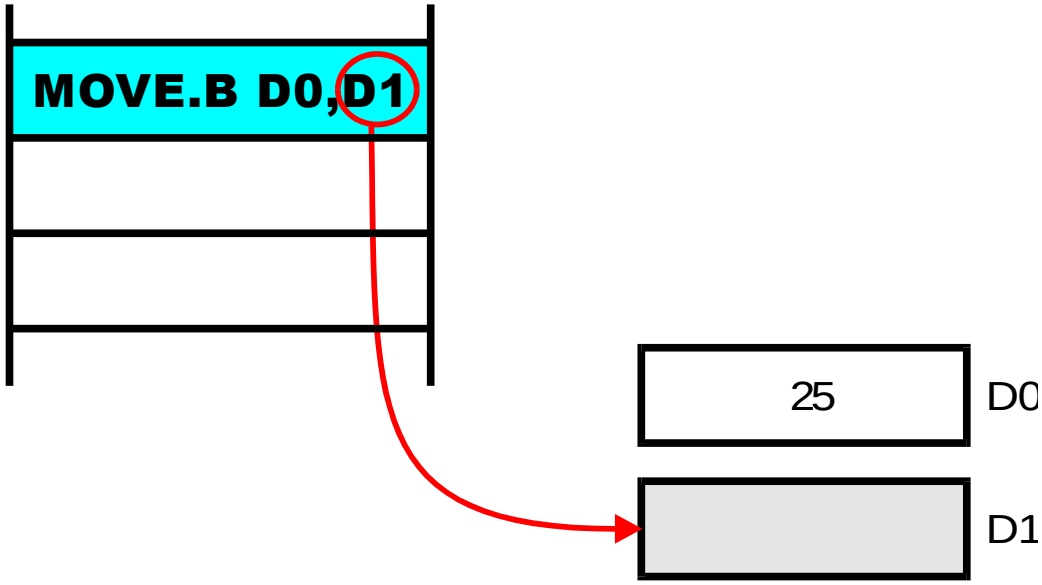
```
MOVE.B  D0,D3       Copia l'operando sorgente in D0 nel registro D3
SUB.L   D3,A0       Calcola [A0] – [D3] e risultato in A0
CMP.W   D2,D0       Confronta i valori dei registri D2 e D0 ([D0]-[D2])
ADD     D3,D4       Somma il contenuto di D3 e D4 e risultato in D4
```

Register Direct Addressing



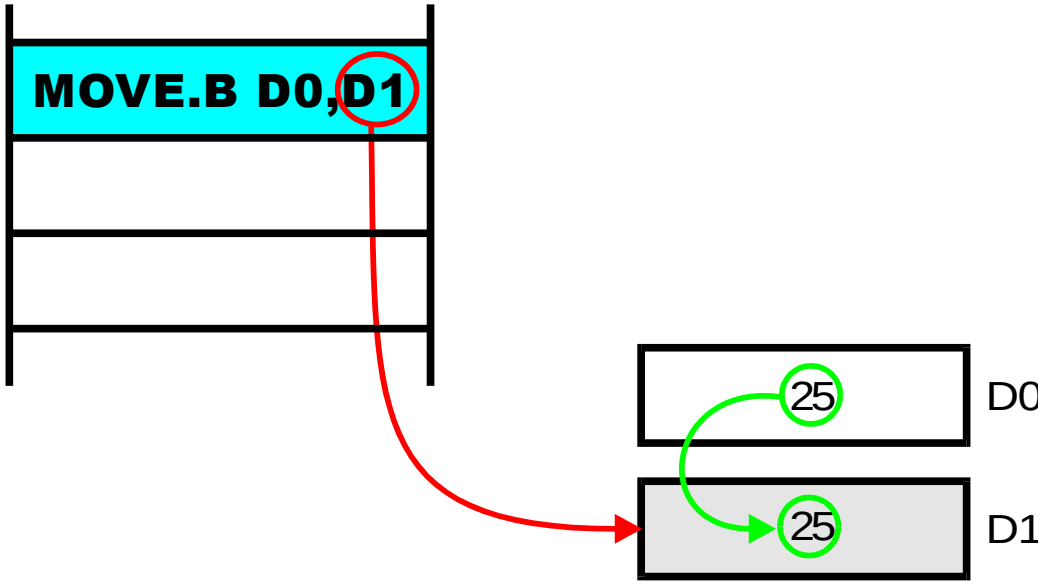
L’istruzione MOVE.B D0,D1 usa registri
dati sia per l’operando sorgente che per quello destinazione

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

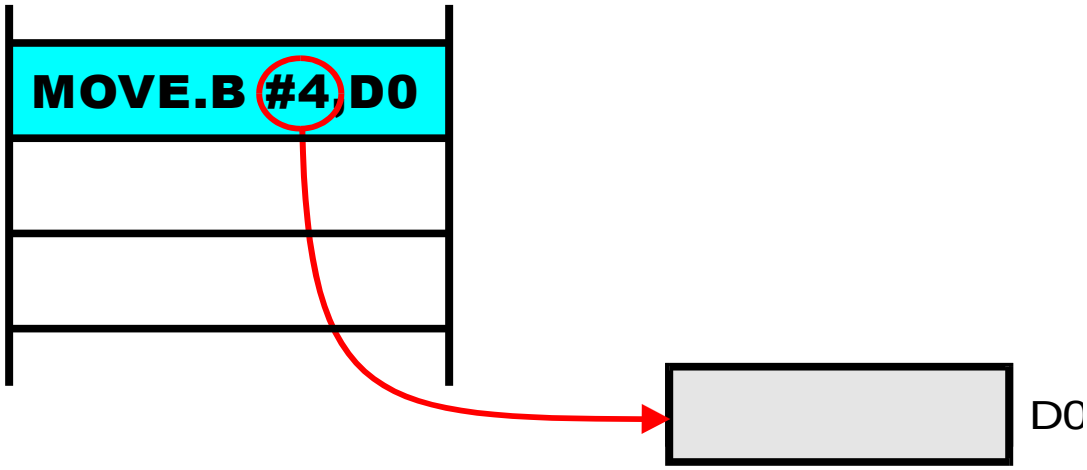
Immediate Addressing

- 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:

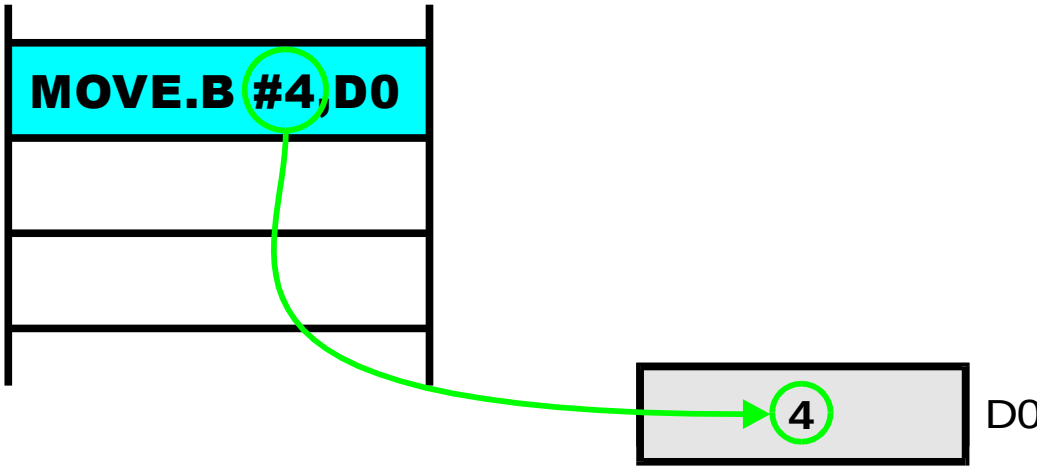
MOVE.B #4,D0 Usa l’operando sorgente immediato 4

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

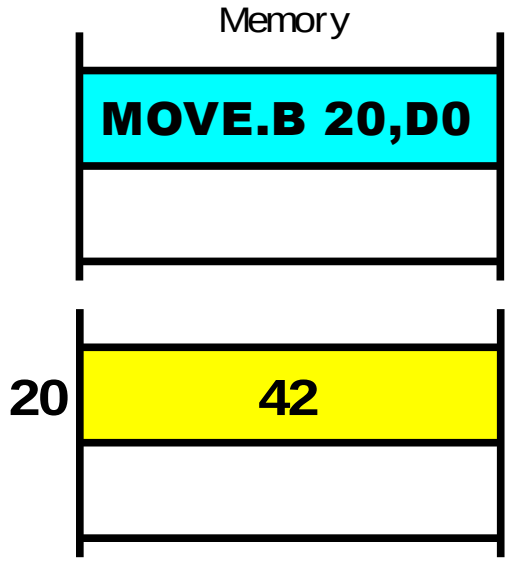
Immediate Addressing: caratteristiche

- 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

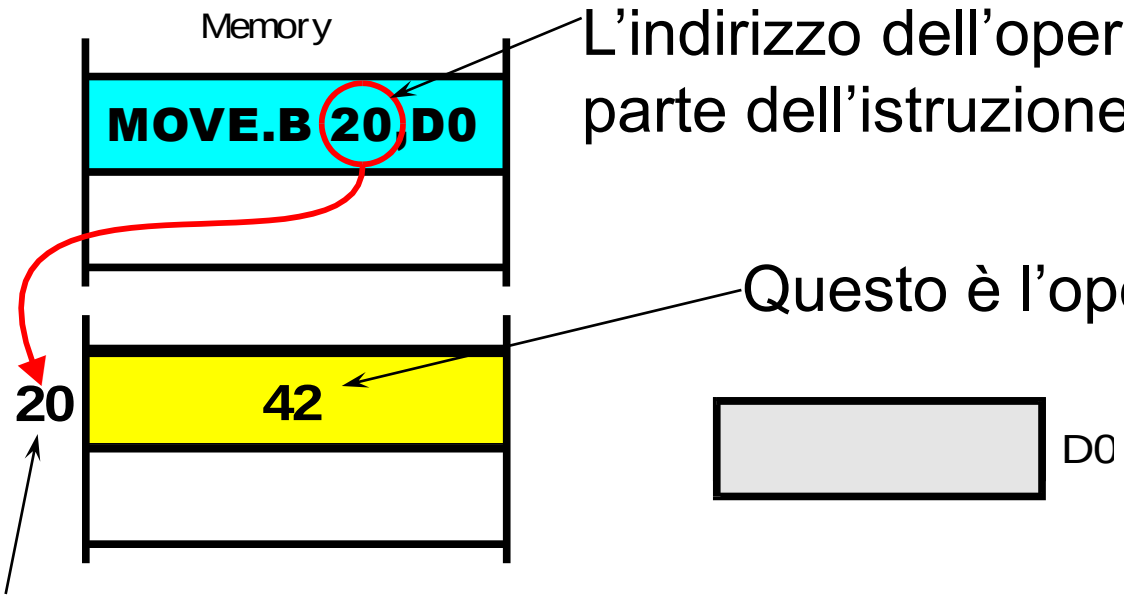


Questa istruzione ha un operando absolute

L'operando sorgente è in memoria

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

Absolute Addressing: funzionamento

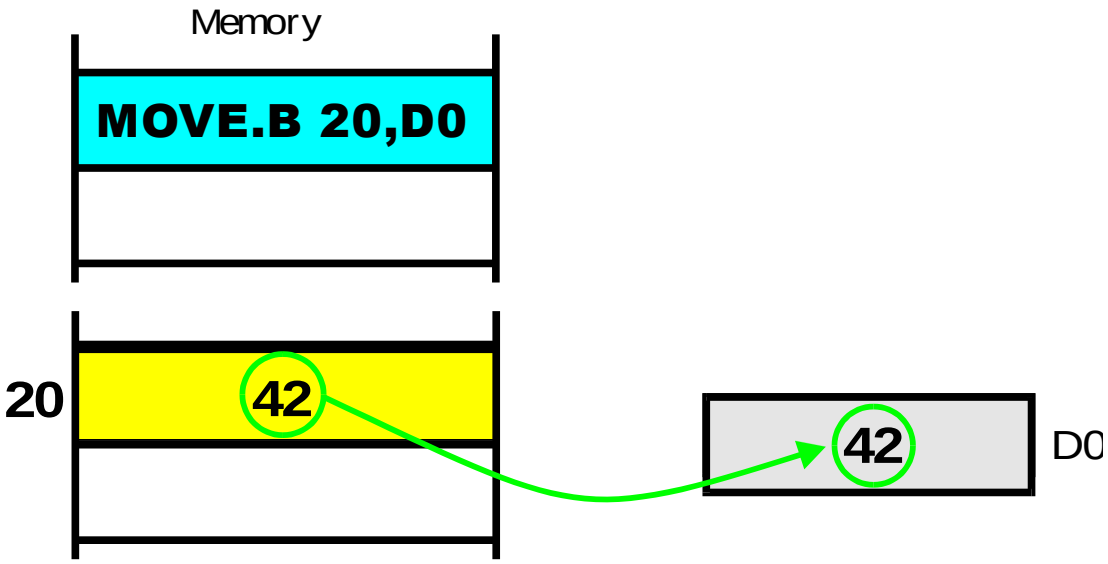


L'indirizzo dell'operando forma parte dell'istruzione

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

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

0000

0000000000000000	**0000000000000000**

7FFE

0000000000000000	**0111111111111110**

8000

1111111111111111	**1000000000000000**

FFFE

1111111111111111	**1111111111111110**

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:

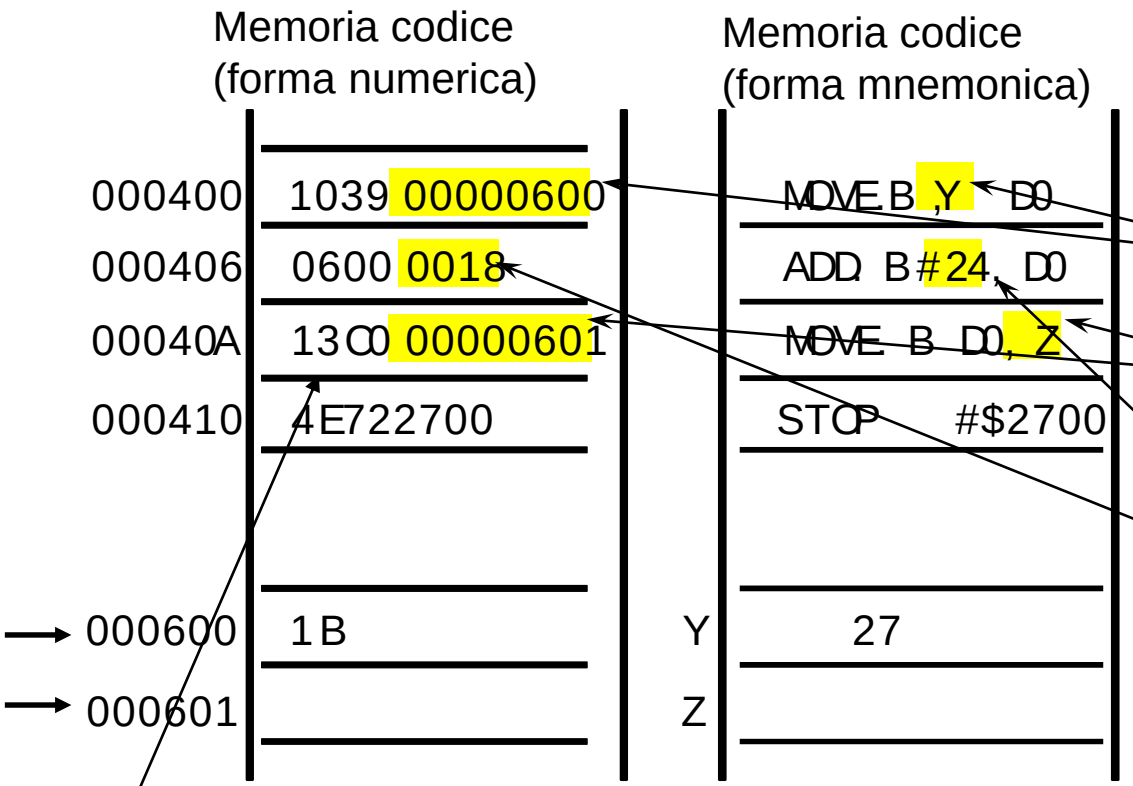
```
        ORG     $400            Inizio del codice
        MOVE.B  Y,D0
        ADD     #24,D0
        MOVE.B  D0,Z

        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

LEA: Load Effective Address

Operazione: [An] ← <ea>

Sintassi: LEA <ea>,An

Esempio: LEA table,A3

Attributi: Size = longword

Descrizione:

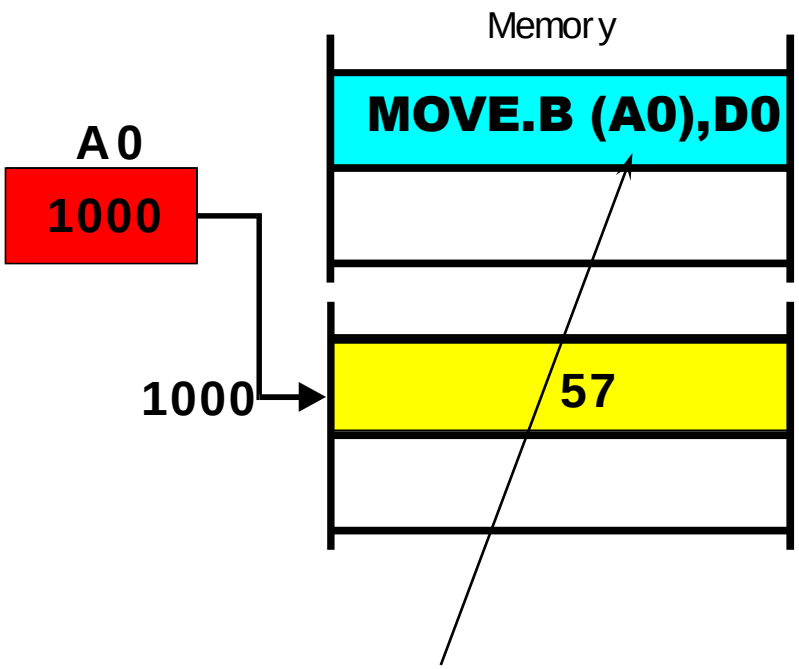
Calcola l'indirizzo effettivo (<*ea*>) del primo operando, generalmente espresso in forma simbolica, e lo pone nel registro indirizzo specificato dal suo secondo operando. Non influenza i flag di stato:

X	N	Z	V	C
-	-	-	-	-

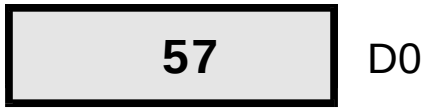
Address Register Indirect Addressing

- 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

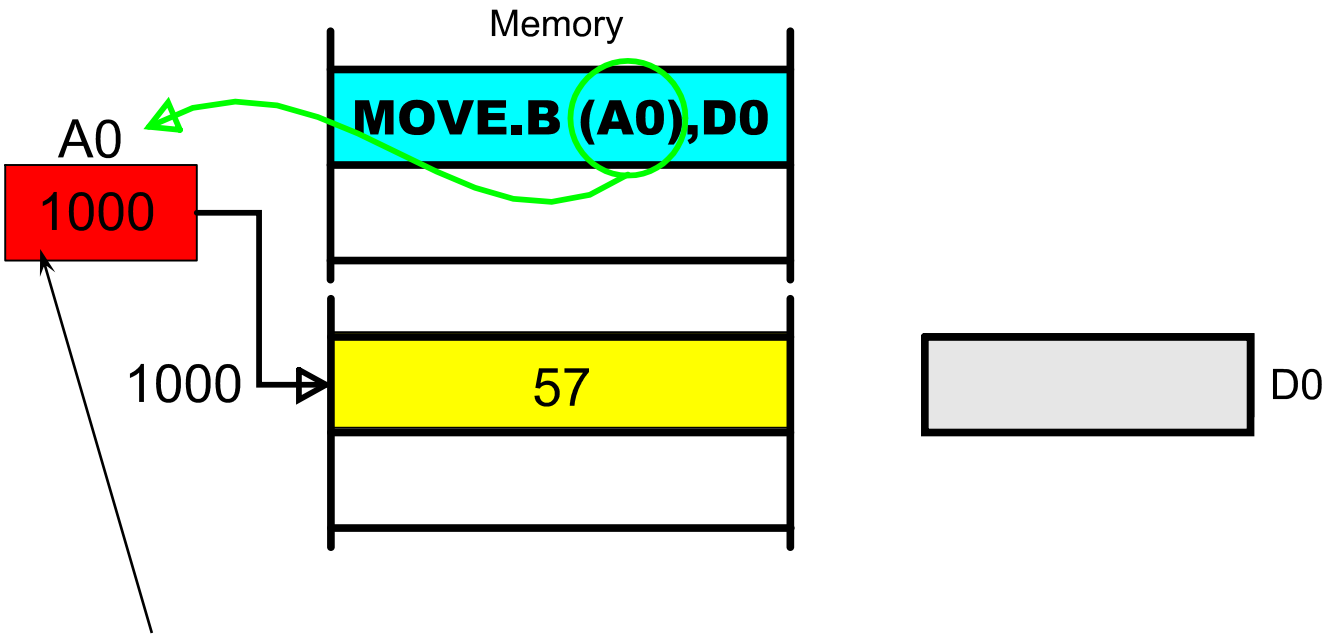


Questa istruzione significa: carica D0 con il contenuto della locazione puntata dal registro indirizzo A0



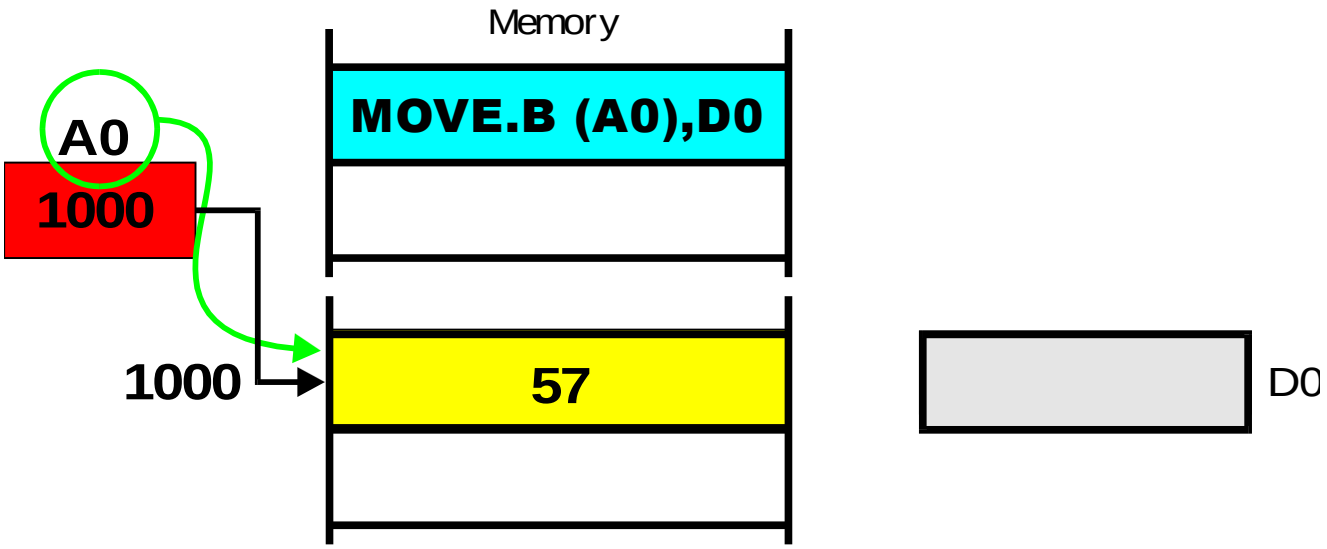
L'istruzione specifica l'operando sorgente come (A0)

Address Register Indirect: funzionamento



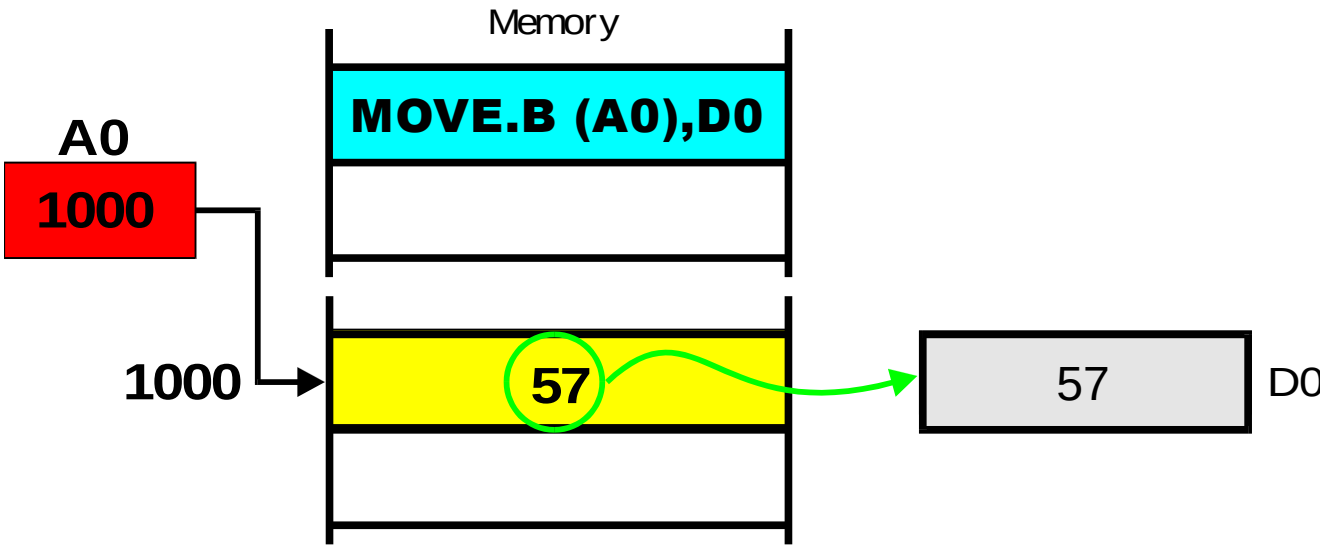
Il registro indirizzo nell'istruzione specifica un registro indirizzo che contiene l'indirizzo dell'operando

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-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 puntanto da A0

Auto-pre-decrement

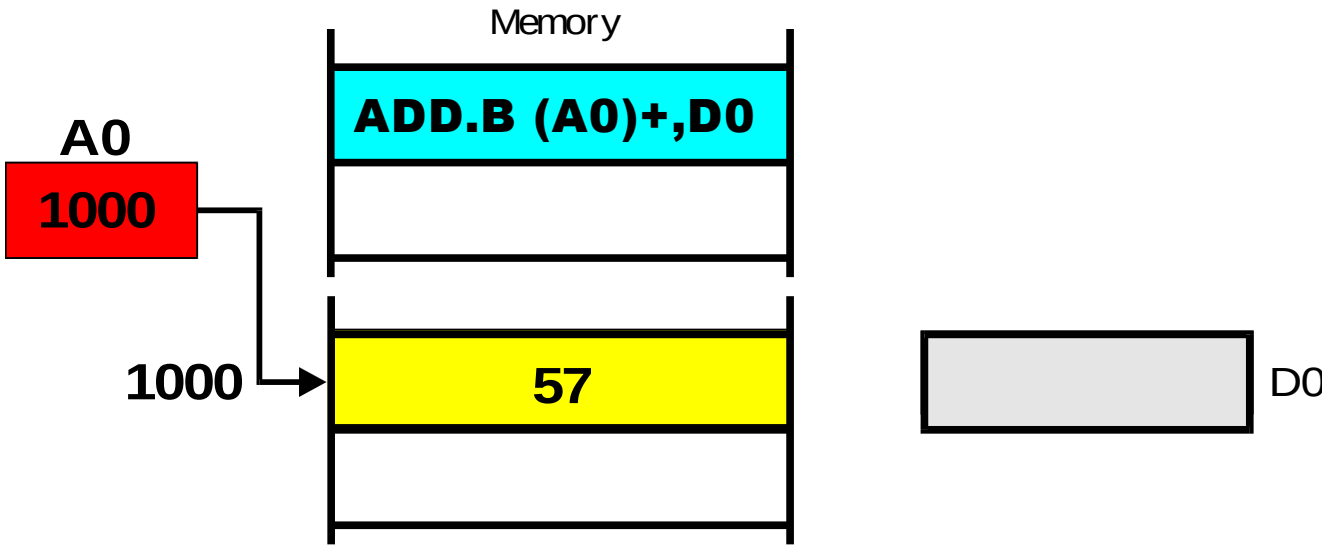
- L'istruzione specifica uno dei registri indirizzo
- Se il modo di indirizzamento è specificato come

  -(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)

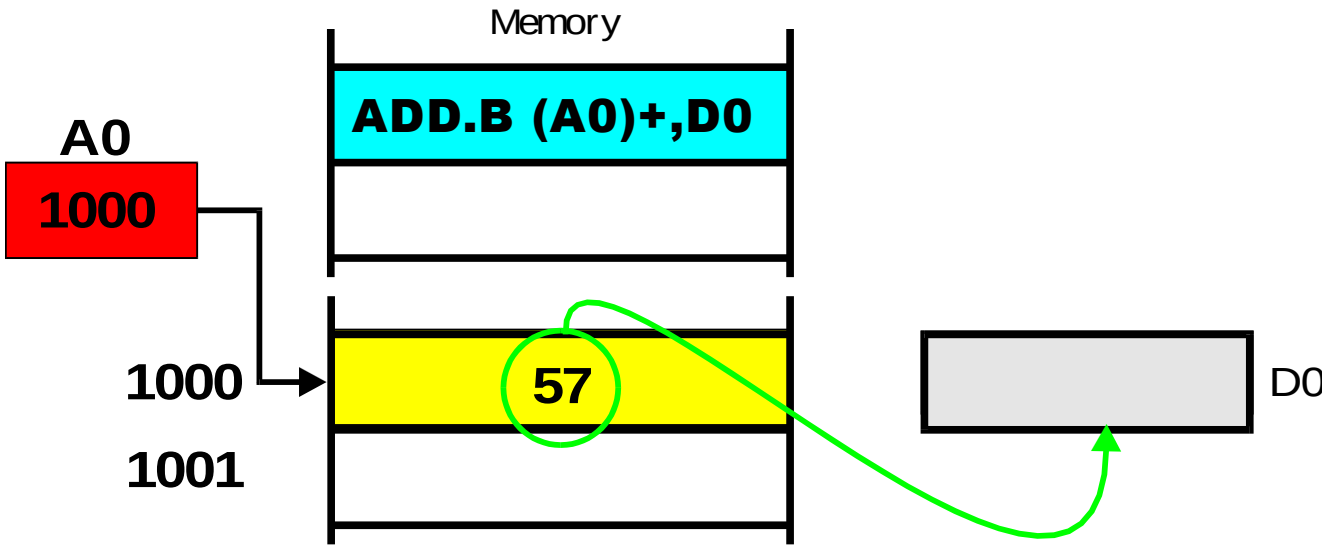
Sottrae 2 ad A0 e poi lo usa per la MOVE (2 poiché l'accesso è di tipo .W = 2 byte). Di datto, l'istruzione esegue un push di D0 sullo stack puntato da A0.

Auto-post-increment: funzionamento



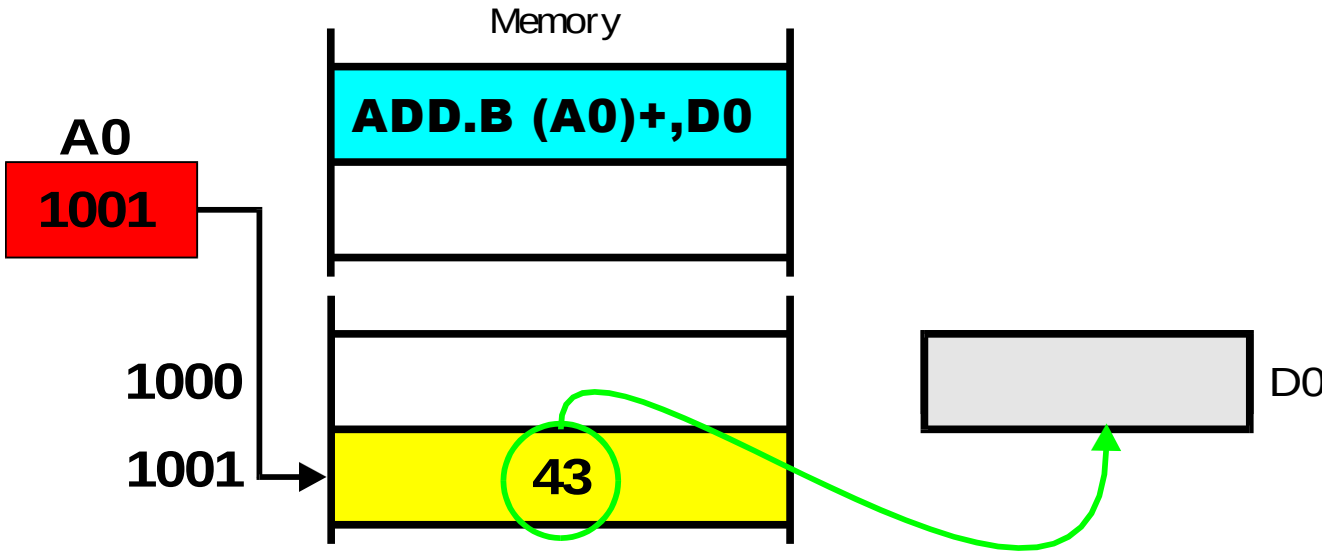
Il registro indirizzo contiene 1000
ovvero “punta” alla locazione 1000

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

Utilità di ulteriori modi di indirizzamento

- 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

Utilità di ulteriori modi di indirizzamento (cont.)

- Ci sono situazioni in cui un solo grado di libertà attraverso un registro An non è sufficiente
- Esempi
  - Accedere agli elementi di una matrice A(i,j)
  - Accedere ai campi di un array di record
  - Accedere ai campi di un record la cui posizione è determinata a tempo di esecuzione
    - Es. record di attivazione di una subroutine
- Soluzione:
  - Metodi di indirizzamento che costruiscono l'EA mediante due o più componenti (detti anche *modi con modifica di indirizzo*)
    - EA = C + R1 + R2 + … + Rk
      con C = valore costante espresso su n bit contenuto nella istruzione
  - Il processore MC68000 presenta diversi ulteriori modi di indirizzamento che rientrano in questa categoria

Indexed addressing

- Detto anche **diretto con registro indice**
- In generale, l'Indexed Addressing combina due componenti mediante somma, per formare l'EA
  - EA = C + R = B + I
  - C è specificato nella istruzione e rappresenta l'indirizzo base (*base address*) *B*
  - R è contenuto in un registro indice (*index register*) e contiene il valore I da sommare al *base address* per ottenere l'EA (*spiazzamento*)
- È adatto per accedere ai valori di array e di tabelle
- Il processore MC68000 non supporta esplicitamente il modo Direct Indexed

Based Addressing

- Detto anche **con registro base**
- Based Addressing è esattamente l'inverso dell'Indexed Addressing
- Forma l'EA combinando due componenti mediante somma:
  - EA = C + R = I + B
  - Il primo componente C, specificato come parte dell'istruzione e quindi costante, assume il significato di *spiazzamento* (*displacement* I)
  - Il secondo componente è contenuto in un registro e rappresenta l'indirizzo base della struttura dati da accedere (*base address* B)
- 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

- Il modo Based Indexed Addressing forma l'EA combinando una componente costante C e due componenti variabili mediante somma:
  - EA = C + R1 + R2 = C + B + I
  - B ha il significato di indirizzo base
  - I ha il significato di *displacement* ed è preso da un registro indice
- 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

MC68000: Short Based Indexed e Long Based Indexed

- 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
- d8 (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

Relative Addressing

- “Relative” indica che il calcolo dell’indirizzo è relativo al Program Counter (PC)
- Questo modo di indirizzamento calcola l’indirizzo effettivo come la somma di un displacement fisso specificato nell’istruzione e del valore corrente del PC
  - EA = PC + displacement
- 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
  - EA = PC + Xi + displacement
- Può essere usato per saltare ad aree di memoria read-only, contenenti dati o istruzioni