Corso di Calcolatori Elettronici I
A.A. 2010-2011

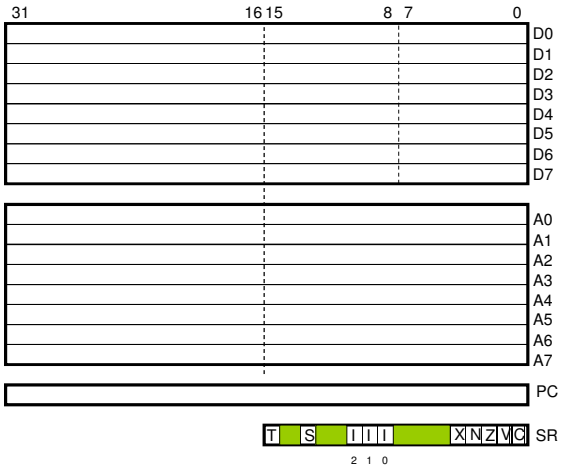
Modi di indirizzamento parte 1

Lezione 21

Prof. Roberto Canonico

Università degli Studi di Napoli Federico II
Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica (allievi A-DE+Q-Z)
Corso di Laurea in Ingegneria dell'Automazione

Modello di programmazione del processore MC68000



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

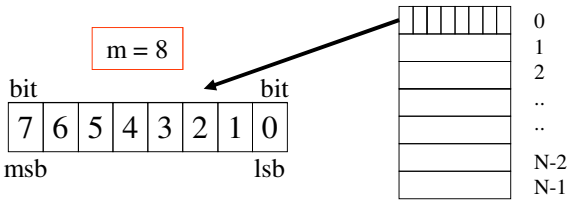
Registri indirizzo
utilizzabili per indirizzi di 32 bit (Long Word) oppure 16 bit (Word)

Program counter

Registro di stato

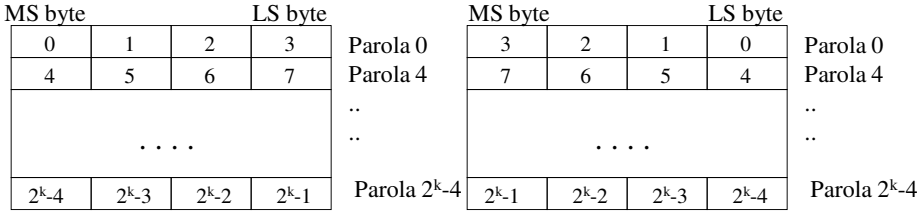
Memoria centrale

- 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

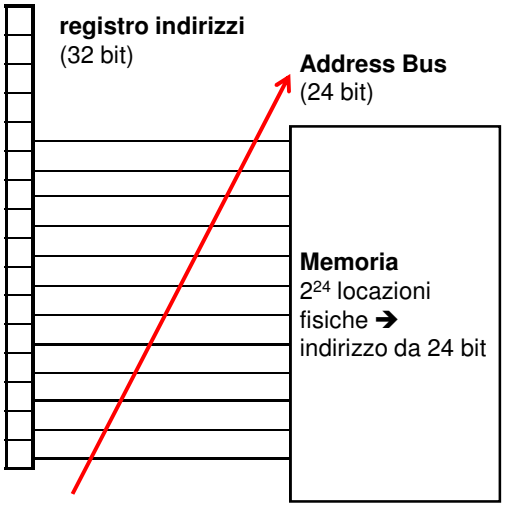


Organizzazione della memoria

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



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

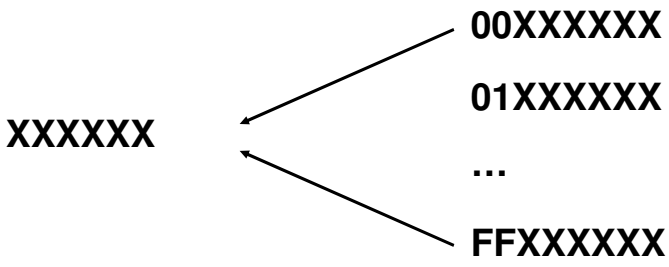
- Memoria Byte Addressable
- Parallelismo Registri Indirizzo: 32 bit
  - Spazio di indirizzamento logico: 4 GB
- Parallelismo Address Bus: 24 bit
  - Spazio di indirizzamento fisico: 16 MB
- Parallelismo Data Bus: 16 bit
  - Pur disponendo di istruzioni in grado di trattare dati a 32 bit, il processore 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:



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

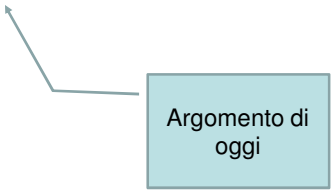
	0000
0000000000000000	0000000000000000
	7FFE
0000000000000000	0111111111111110
	8000
1111111111111111	1000000000000000
	FFFE
1111111111111111	1111111111111110

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



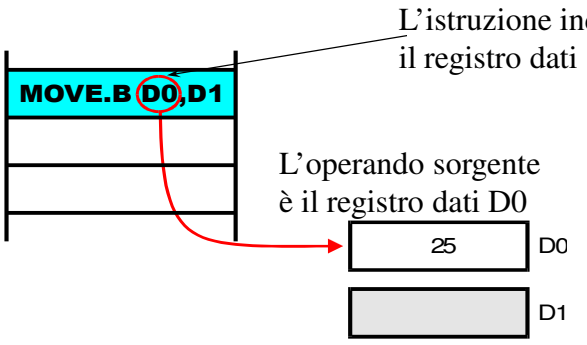
- Address-register Indirect
- Auto-Increment
- Auto-Decrement
- Indexed short
- Based
- Based Indexed
  - Short
  - Long
- Relative
- Relative Indexed
  - Short
  - Long

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

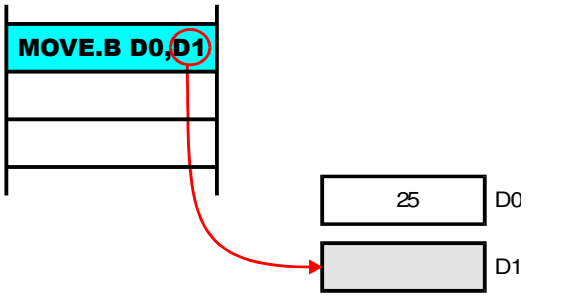
MOVE.B D0,D3	Copia l'operando sorgente in D0 nel registro D3
SUB.L A0,D3	Sottrae l'operando sorgente nel registro A0 dal registro D3
CMP.W D2,D0	Confronta l'op. sorgente nel registro D2 con il registro D0
ADD D3,D4	Somma l'operando sorgente nel registro D3 al registro D4

Register Direct Addressing – Funzionamento

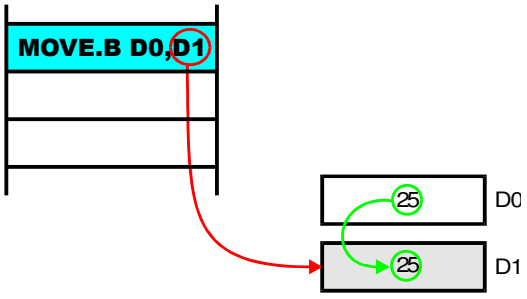


L'istruzione MOVE.B D0,D1 usa registri dati sia per l'operando sorgente che per quello destinazione

Register Direct Addressing – Funzionamento



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
- I programmatori lo usano per memorizzare variabili che sono usate di frequente (scratchpad storage)

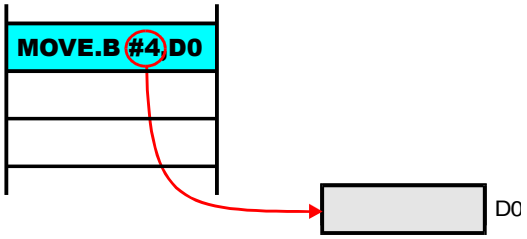
Immediate Addressing

- L'operando effettivo costituisce parte dell'istruzione
- Può essere usato unicamente per specificare un operando sorgente
- È indicato da un simbolo # davanti all'operando sorgente
- Un operando immediato è anche chiamato literal

Esempio:

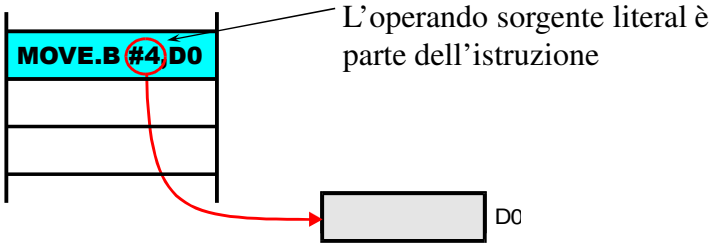
MOVE.B #4,D0 Usa l'operando sorgente immediato 4

ImmediateAddressing - Funzionamento

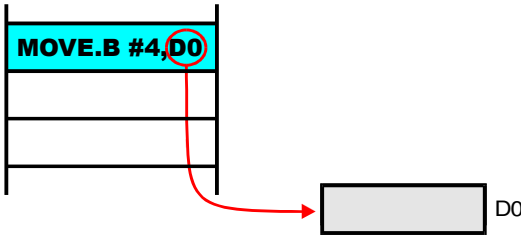


L'istruzione MOVE.B #4,D0 usa un operando sorgente literal ed un operando destinazione register direct

ImmediateAddressing - Funzionamento

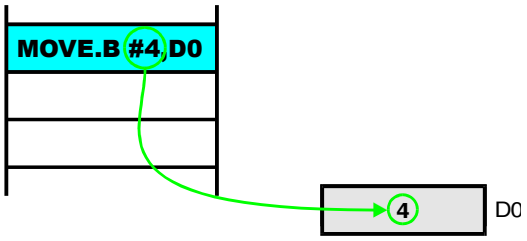


ImmediateAddressing - Funzionamento



L'operando destinazione è un registro dati

ImmediateAddressing - Funzionamento

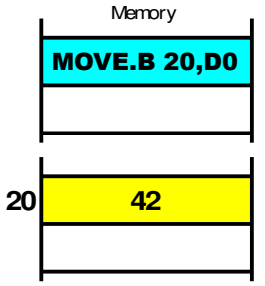


L'effetto di questa istruzione è quello di copiare il valore de
literal 4 nel registro dati D0

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
  - Il secondo è per accedere all'operando effettivo
- Esempio:
  - CLR.B 1234 azzera il contenuto della locazione di memoria 1234

Absolute Addressing - Funzionamento

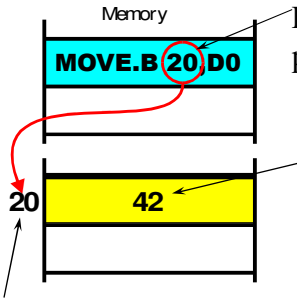


Questa istruzione ha un operando absolute

L'operando sorgente è in memoria

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

Absolute Addressing - Funzionamento

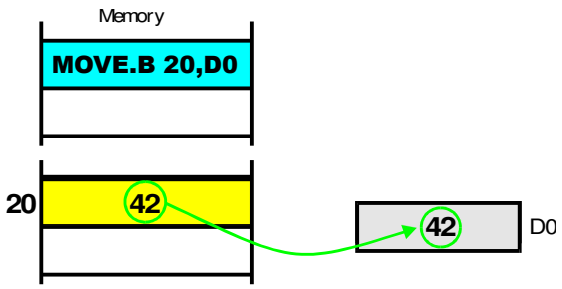


L'indirizzo dell'operando forma parte dell'istruzione

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:

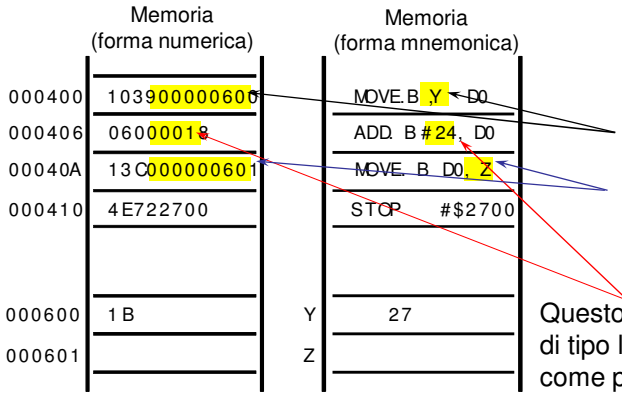
```
        ORG    $400        Inizio del codice
        MOVE.B Y,D0
        ADD    #24,D0
        MOVE.B D0,Z

        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

Riepilogo modi fondamentali

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