Corso di Calcolatori Elettronici I

Repertorio istruzioni l/m del processore MC68000: istruzioni di salto

Prof. Roberto Canonico

Università degli Studi di Napoli Federico II
Dipartimento di Ingegneria Elettrica e
delle Tecnologie dell'Informazione (DIETI)

Istruzioni di salto

- Tutte le istruzioni di salto alterano il contenuto del PC
- Le istruzioni di salto possono classificarsi in:
  - Istruzioni di salto assoluto (*jump*) – contengono nel codice in linguaggio macchina l'indirizzo di destinazione da caricare in PC
  - Istruzioni di salto relativo (*branch*) – contengono nel codice in l/m un *offset* da sommare al PC per determinare l'indirizzo di destinazione
- … oppure anche in:
  - Istruzioni di salto condizionato – il salto è effettuato solo se una certa condizione sui flag N,Z,V,C è vera
  - Istruzioni di salto incondizionato – il salto è effettuato sempre
- Le due classificazioni sono ortogonali
- Nel processore MC68000 non esistono le istruzioni di salto assoluto condizionato

Istruzione di salto assoluto ed incondizionato JMP

Operazione: [PC] ← destination
Sintassi: JMP <ea>
ad esempio: JMP loop3
Attributi: --

Descrizione:

E' un'istruzione di **salto assoluto ed incondizionato**: nel codice in linguaggio macchina è espresso l'indirizzo dell'istruzione destinazione

Scrive il valore dell'operando destinazione nel Program Counter

Prevede differenti modi di indirizzamento per specificare l'operando destinazione (a differenza della Branch).

Non influenza i flag di stato:

X	N	Z	V	C
-	-	-	-	-

B*cc*: Branch on condition *cc*

Operazione: IF cc = 1 THEN [PC] ← [PC] + d

Sintassi: B*cc* <label>

Descrizione:

Famiglia di istruzioni di **salto relativo condizionato o incondizionato**

cc indica una di svariate possibili condizioni che è possibile verificare sul valore dei flag N, Z, V, C del registro di stato
Se la condizione logica specificata dalla condizione *cc* è verificata, viene caricato in PC il valore ottenuto sommando lo spiazzamento (*displacement*) **d** al valore precedente del PC
Lo spiazzamento **d** è rappresentato in complementi a due e può pertanto essere anche negativo (salto "indietro")

L'istruzione BRA è un caso particolare in cui cc=TRUE ed è un salto relativo incondizionato

In linguaggio assembly l'istruzione ha come operando l'etichetta <label> dell'istruzione di destinazione del salto. L'assemblatore traduce l'etichetta nel corrispondente valore di *displacement* **d** da inserire nel codice l/m

In linguaggio macchina il *displacement* **d** può essere codificato con 8-bit o 16-bit; l'assemblatore sceglie la variante opportuna

B*cc*: possibili condizioni *cc*

- **Single bit**
  - BCS branch on carry set — C = 1
  - BCC branch on carry clear — C = 0
  - BVS branch on overflow set — V = 1
  - BVC branch on overflow clear — V = 0
  - BEQ branch on equal (zero) — Z = 1
  - BNE branch on not equal — Z = 0
  - BMI branch on minus (i.e., negative) — N = 1
  - BPL branch on plus (i.e., positive) — N = 0
- **Signed**
  - BLT branch on less than (zero) — $N \oplus V = 1$
  - BGE branch on greater than or equal — $N \oplus V = 0$
  - BLE branch on less than or equal — $(N \oplus V) + Z = 1$
  - BGT branch on greater than — $(N \oplus V) + Z = 0$
- **Unsigned**
  - BLS branch on lower than or same — C + Z = 1
  - BHI branch on higher than — C + Z = 0

C, V, Z, N sono flag del registro di stato
C = flag di carry
V = flag di overflow
Z = flag di zero
N = flag di negative

Significato delle condizioni nella BRANCH

- Se i numeri sono interpretati come unsigned:

BHS	BCC	branch on higher than or same
BHI		branch on higher than
BLS		branch on lower than or same
BLO	BCS	branch on less than

- Se i numeri sono interpretati come signed

BGE		branch on greater than or equal
BGT		branch on greater than
BLE		branch on lower than or equal
BLT		branch on less than

Differenza *signed / unsigned* nel confronto

Ad esempio:

- $FF è maggiore di $10 se i numeri sono interpretati come **unsigned**, in quanto 255 è maggiore di 16
- Tuttavia se i numeri sono interpretati come **signed**, $FF è minore di $10, in quanto -1 è minore di 16.

➔ il processore non tiene conto del tipo di rappresentazione quando setta i flag di condizione. Sta al programmatore conoscere il formato dei dati manipolati ed interpretare correttamente gli effetti sui flag di stato.

DBcc: Test condition, decrement, and branch

Operazione: IF (*cc* false) THEN

[Dn] ← [Dn] – 1

IF [Dn] = -1 THEN [PC] ← [PC] + 2

ELSE [PC] ← [PC] + d

ELSE [PC] ← [PC] + 2

Sintassi: DBcc Dn,<label>

Attributi: Size = word

Descrizione:

Fintantoché la condizione *cc* rimane falsa, decrementa il registro D*n,* e se questo non era zero prima del decremento (ovvero se non vale -1) salta all'istruzione a distanza *d*. Negli altri casi, passa all'istruzione seguente.

Fornisce un modo sintetico per gestire i cicli, sostituendo con un'unica istruzione il decremento di un registro di conteggio e la verifica di una condizione normalmente fatti con istruzioni separate.

Supporta tutti i cc usati in Bcc. Inoltre, ammette anche le forme DBF e DBT (F = false, e T = true) per ignorare la condizione ed usare solo il registro di conteggio.

Esempio di programma assembler 68000

PLC	contenuto		label	opcode	operands
00000000		1	* Programma per sommare i primi 17 interi		
00000000		2	*		
00008000		3		ORG	$8000
00008000	4279 00008032	4	START	CLR.W	SUM
00008006	3039 00008034	5		MOVE.W	ICNT,D0
0000800C	33C0 00008030	6	ALOOP	MOVE.W	D0,CNT
00008012	D079 00008032	7		ADD.W	SUM,D0
00008018	33C0 00008032	8		MOVE.W	D0,SUM
0000801E	3039 00008030	9		MOVE.W	CNT,D0
00008024	0640 FFFF	10		ADD.W	#-1,D0
00008028	66E2 **E2=OFFSET**	11		BNE	ALOOP
0000802A	4EF9 00008008 → **DEST**	12		JMP	SYSA
00008030	=00008008	13	SYSA	EQU	$8008
00008030		14	CNT	DS.W	1
00008032		15	SUM	DS.W	1
00008034	=00000011	16	IVAL	EQU	17
00008034	0011	17	ICNT	DC.W	IVAL

Symbol Table

ALOOP	800C	CNT	8030	IVAL	0011
START	8000	SUM	8032	ICNT	8034

Esempio - Somma di n interi

```
        ORG         $8000
START   CLR.W        SUM
        MOVE.W      ICNT,D0
ALOOP   MOVE.W      D0,CNT
        ADD.W       SUM,D0
        MOVE.W      D0,SUM
        MOVE.W      CNT,D0
        ADD.W       #-1,D0
        BNE         ALOOP
FINE    JMP         FINE
        ORG         $8100
CNT     DS.W        1
SUM     DS.W        1
IVAL    EQU         17
ICNT    DC.W        IVAL
        END         START
```

COMMENTO SUL CODICE

Questo codice è estremamente inefficiente, perché fa un numero eccessivo di accessi in memoria per prelevare o salvare dati che potrebbero essere mantenuti in registri del processore.

Nel programma ci si limita ad usare per dati temporanei solo il registro D0 quando potrebbero essere usati altri registri.

La scrittura del risultato in memoria con l'istruzione MOVE.W D0,SUM può essere fatta una sola volta all'uscita dal ciclo.

Si veda la raccolta di esercizi per una versione dello stesso programma più efficiente