

Corso di Calcolatori Elettronici I

Strutture di controllo del flusso di esecuzione in assembler

Prof. Roberto Canonico



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

Istruzioni di selezione in assembler (1)

Linguaggio di alto livello:

```
if (espressione)
    istruzione
istruzione_successiva
```

NOTA: istruzione può essere un *compound statement*

Linguaggio assembler (processore MC 68000):

```
        B(NOT condizione) labelA
        istruzione
        ...
labelA  istruzione_successiva
```

Esempio:

| | |
|--------------|---------------------|
| if (D0 == 5) | CMPI.L #5, D0 |
| D1++; | BNE SKIP |
| D2 = D0; | ADDQ.L #1, D1 |
| | SKIP MOVE.L D0, D2 |

Istruzioni di selezione in assembler (2)

Linguaggio di alto livello:

```
if (espressione)
    istruzione1
else
    istruzione2
istruzione_successiva
```

Linguaggio assembler (processore MC 68000):

```
        B(NOT condizione) labelA
        istruzione1
        ...
        BRA labelB
labelA  istruzione2
        ...
labelB  istruzione_successiva
```

Strutture iterative in assembler (1)

Linguaggio di alto livello:

```
do
    istruzione
while (condizione == TRUE);
istruzione_successiva
```

Linguaggio assembler (processore MC 68000):

```
labelA    istruzione
          ...
          Bcc labelA
          istruzione_successiva
```

Esempio: calcola 3^N ($N > 0$)

```
D0 = 1; D1 = 1;
```

```
do {
```

```
    D0 = D0 * 3;
```

```
    D1++;
```

```
} while (D1 <= N);
```

```
MOVE.B #N, D2
```

```
MOVE.B #1, D1
```

```
MOVE.W #1, D0
```

```
LOOP MULU.W #3, D0
```

```
ADDQ.B #1, D1
```

```
CMP.B D2, D1
```

```
BLE LOOP
```

Strutture iterative in assembler (2)

Linguaggio di alto livello:

```
while (condizione == TRUE)
    istruzione;
istruzione_successiva
```

Linguaggio assembler (processore MC 68000):

```
                BRA labelB
labelA           istruzione
                ...
labelB          Bcc labelA
                istruzione_successiva
```

Esempio: calcola 3^N ($N \geq 0$)

| | | |
|-------------------|------|---------------|
| D0 = 1; D1 = 1; | | MOVE.B #N, D2 |
| while (D1 <= N) { | | MOVE.B #1, D1 |
| D0 = D0 * 3; | | MOVE.W #1, D0 |
| D1++; | | BRA TEST |
| }; | LOOP | MULU.W #3, D0 |
| | | ADDQ.B #1, D1 |
| | TEST | CMP.B D2, D1 |
| | | BLE LOOP |

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 *Dn*, 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.

L'istruzione Decrement and Branch always: DBRA (*)

DBRA equivale a DBF: caso particolare di DBcc con cc=FALSE

Esempio:

equivale a:

| | | | | | |
|------|---------|------------|------|---------|------------|
| | MOVE.L | #N, D1 | | MOVE.L | #N, D1 |
| | SUBQ.L | #1, D1 | | SUBQ.L | #1, D1 |
| | MOVEA.L | #NUM, A2 | | MOVEA.L | #NUM, A2 |
| | CLR.L | D0 | | CLR.L | D0 |
| LOOP | ADD.W | (A2) +, D0 | LOOP | ADD.W | (A2) +, D0 |
| | DBRA | D1, LOOP | | SUBQ | #1, D1 |
| | MOVE.L | D0, SOMMA | | BGE | LOOP |
| | | | | MOVE.L | D0, SOMMA |
