Corso di Calcolatori Elettronici I

Introduzione al linguaggio assembly MC68000

Prof. Roberto Canonico

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

Assembly MC6800: formato del codice sorgente

- Nel linguaggio assembly MC68000 usato nel corso, ciascuna linea di codice sorgente è costituita da quattro campi:
  - LABEL (opzionale)
    - Stringa alfanumerica
    - Definisce un nome simbolico per il corrispondente indirizzo
  - MNEMONIC
    - Codice mnemonico associato ad un'istruzione in linguaggio macchina o ad una direttiva di assemblaggio
  - OPERANDI
    - A seconda del codice mnemonico, possono essere zero, uno o due
    - Variano a seconda dell'OPCODE e del modo di indirizzamento
  - COMMENTI
    - Testo arbitrario inserito dal programmatore
- I campi sono separati da uno o più spazi
  - A sinistra del codice mnemonico è obbligatorio (almeno) uno spazio

Convenzioni (1)

- Gli spazi bianchi tra i diversi campi fungono esclusivamente da separatori
  - Due o più spazi in sequenza equivalgono ad uno solo per l'assemblatore
- Una linea che inizi con un asterisco (*) è una linea di commento
- Alcune istruzioni accettano un valore costante come operando
  - Un valore costante va preceduto dal simbolo # (*indirizzamento immediato*)
  - Un valore costante può anche essere rappresentato nel codice assembly mediante una espressione costante, cioè un'espressione aritmetica in cui compaiono solo valori costanti
    - Il calcolo della espressione costante è eseguito dall'assemblatore a tempo di assemblaggio
    - L'assemblatore inserisce come operando della istruzione in linguaggio macchina il risultato della espressione
    - Esempio: MOVE #3*2+5,D0 equivale a: MOVE #11,D0
  - Per default, valori numerici costanti si intendono espressi in decimale
  - Un valore numerico costante preceduto dal simbolo $ si intende in esadecimale
    - Esempio: MOVE.B #$20,D0 equivale a: MOVE.B #32,D0
  - Un valore numerico costante preceduto dal simbolo % si intende in binario
    - Esempio: MOVE.B #%01000001,D0 equivale a: MOVE.B #65,D0

Convenzioni (2)

- Per la maggior parte dei codici mnemonici è possibile specificare l'ampiezza in bit dell'operazione effettuata mediante un suffisso
  - Suffisso .L → ampiezza in bit = 32 bit (*longword*)
  - Suffisso .W → ampiezza in bit = 16 bit (*word*)
  - Suffisso .B → ampiezza in bit = 8 bit (*byte*)
- Per la maggior parte delle istruzioni, se in assembly non è indicato esplicitamente un suffisso, l'assemblatore assume per l'ampiezza in bit il valore di 16 bit (*word*)
  - Esempio: MOVE D1,D0 equivale a: MOVE.W D1,D0
  - A meno che l'istruzione non sia vincolata a lavorare su uno specifico tipo (es. longword)
  - Ad esempio:
    - L'istruzione LEA lavora esclusivamente sul tipo longword
    - L'istruzione ANDI #$FA,CCR lavora esclusivamente su un dato byte
  - Per questi dettagli è opportuna la consultazione del manuale Assembly MC68000

Program Location Counter PLC

- E’ una variabile interna dell’assemblatore
- Ogni istruzione del programma ha associato un valore del PLC
- Se il programma è un **codice assoluto** (*allocazione statica*), il PLC è l’indirizzo della locazione di memoria in cui l’istruzione sarà caricata dal *loader*
- Se il programma è un **codice rilocabile**, il valore del PLC rappresenta lo spiazzamento del codice dell’istruzione rispetto ad un indirizzo base
- Il valore del PLC può essere modificato con la direttiva “origin” (ORG)
- Durante il processo di assemblaggio, il suo valore è incrementato della dimensione (in byte) del codice dell’istruzione corrente
- E’ possibile, all’interno di un programma, fare riferimento al suo valore corrente, mediante il simbolo “*”

Direttive di assemblaggio: ORG, END

- NON sono istruzioni eseguite dal processore
  - sono direttive che regolano il processo di traduzione del programma assembler in programma eseguibile
  - essendo trattate come dei codici mnemonici, occorre inserire almeno uno spazio a sinistra del nome della direttiva
- La direttiva ORG
  - Viene usata per assegnare un valore al *Program Location Counter* (PLC), ovvero per indicare a quale indirizzo sarà posta la successiva sezione di codice o dati
  - **Esempio:** ORG $8100
- La direttiva END
  - Viene usata per indicare la fine del codice sorgente ed impostare l'*entry point* (prima istruzione da eseguire) del programma
  - **Esempio:** END ENTRYPOINT

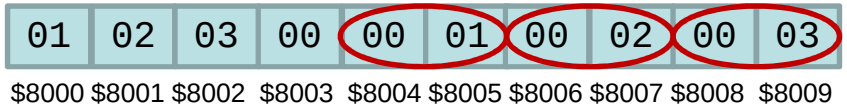
Direttive di assemblaggio: DC, DS, EQU

- La direttiva EQU
  - Viene usata per definire nel sorgente assembler l'associazione tra un valore costante ed un identificatore specificato nel campo etichetta
  - **Esempio:** A EQU 5 associa al simbolo A il valore costante 5
- La direttiva DS
  - Viene usato per riservare spazio di memoria per contenere una variabile (ad es. un array) la cui dimensione è specificata come operando
  - A DS.B N oppure A DS.W N oppure A DS.L N
  - Incrementa il Program Location Counter (PLC) di N*d
    dove: d=1 se .B, d=2 se .W, d=4 se .L
  - Il simbolo usato come etichetta è l'indirizzo iniziale dell'area di memoria
  - **Esempio:** V DS.W 10 riserva un'area di memoria di dimensione 10 word
- La direttiva DC
  - Viene usata per inizializzare a tempo di caricamento un'area di memoria con un valore costante o una sequenza di valori costanti
  - **Esempio:** A DC.W 10,20,30
    inizializza le tre word consecutive a partire da PLC con i valori 10, 20 e 30

Direttiva DC con operandi multipli

- Se nel campo operandi di una direttiva DC il programmatore inserisce una sequenza di valori separati da virgole, l'assemblatore alloca per ciascuno dei valori specificati un byte, una word o una longword a seconda del suffisso specificato a fianco di DC
- L'assemblatore tratta la direttiva DC senza suffisso come DC.W
- Se il valore corrente di PLC è dispari, con DC.W e DC.L l'assemblatore incrementa preventivamente PLC di uno per ottenere dati allineati al margine di parola
- Esempi:

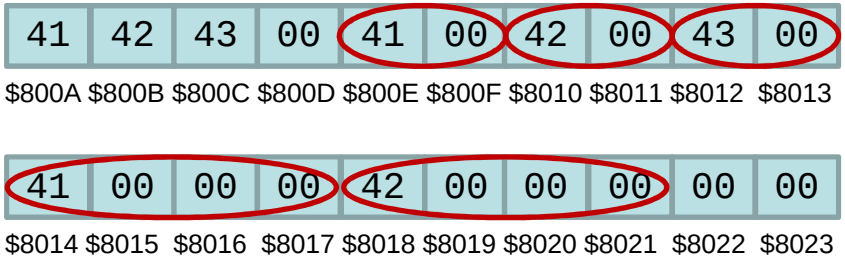
```
   ORG $8000
A  DC.B 1,2,3
B  DC.W 1,2,3
```



Direttiva DC con operandi stringa

- Se nel campo operandi di una direttiva DC il programmatore inserisce delle sequenze di caratteri tra apici, l'assemblatore sostituisce ai caratteri i rispettivi codici ASCII
- Esempio:

```
   ORG   $800A
C  DC.B  'ABC'
D  DC.W  'A','B','C'
E  DC.L  'A','B'
```



- Per ottenere un'area di memoria allocata con i caratteri di una stringa, un carattere per byte e con un carattere finale di codice ASCII zero (tappo), come fa il compilatore C, si scrive:

```
   ORG   $800A
S  DC.B  'Hello world!',0
```

Esempio assembly MC68000 - Moltiplicazione di due interi

```
*   Programma per moltiplicare MCND e MPY
*
        ORG       $8000
*
MULT    CLR.W     D0          D0 accumula il risultato
        MOVE.W    MPY,D1      D1 e' il contatatore di ciclo
        BEQ       DONE        Se il contatore e' zero e' finito
LOOP    ADD.W     MCND,D0     Aggiunge MCND al prodotto parziale
        ADD.W     #-1,D1      Decrementa il contatore
        BNE       LOOP        e ripete il giro
DONE    MOVE.W    D0,PROD     Salva il risultato
LAST    JMP       LAST        Ciclo infinito per terminare
        ORG       $8100
PROD    DS.W      1           Riserva spazio di memoria per PROD
MPY     DC.W      3           Definisce il valore di MPY
MCND    DC.W      4           Definisce il valore di MCND
        END       MULT
```

AsimTool: esempio di file list

```
PLC      contenuto                 label   opcode   operands
00000000                      1  *  Programma per moltiplicare MCND e MPY
00000000                      2  *
00008000                      3         ORG      $8000
00008000  4240                4  MULT   CLR.W
00008002  3239 00008102       5         MOVE.W   MPY,D1
00008008  6700 000E           6         BEQ      DONE
0000800C  D079 00008104       7  LOOP   ADD.W    MCND,D0
00008012  0641 FFFF           8         ADD.W    #-1,D1
00008016  66F4                9         BNE      LOOP
00008018  33C0 00008100      10  DONE   MOVE.W   D0,PROD
0000801E  4EF9 0000801E      11  LAST   JMP      LAST
00008100                     12         ORG      $8100
00008100                     13  PROD   DS.W     1
00008102  0003               14  MPY    DC.W     3
00008104  0004               15  MCND   DC.W     4
00008106                     16         END      MULT
No errors detected
No warnings generated
```

Symbol Table

```
MULT=$00008000      DONE=$00008018      PROD=$00008100      MCND=$00008104
LOOP=$0000800C      LAST=$0000801E      MPY=$00008102
```

Tabella dei simboli

- Durante il processo di assemblaggio, l'assemblatore mantiene nella tabella dei simboli le corrispondenze tra simboli e valori
- Un nuovo simbolo viene creato ogni volta che si assegna un'etichetta ad una istruzione o ad una direttiva DC e DS
- In questi casi, al simbolo viene associato come valore il valore corrente del Program Location Counter (PLC)

```
    ORG    $8100
A   DC.W   5
```

→ ad A è associato il valore $00008100

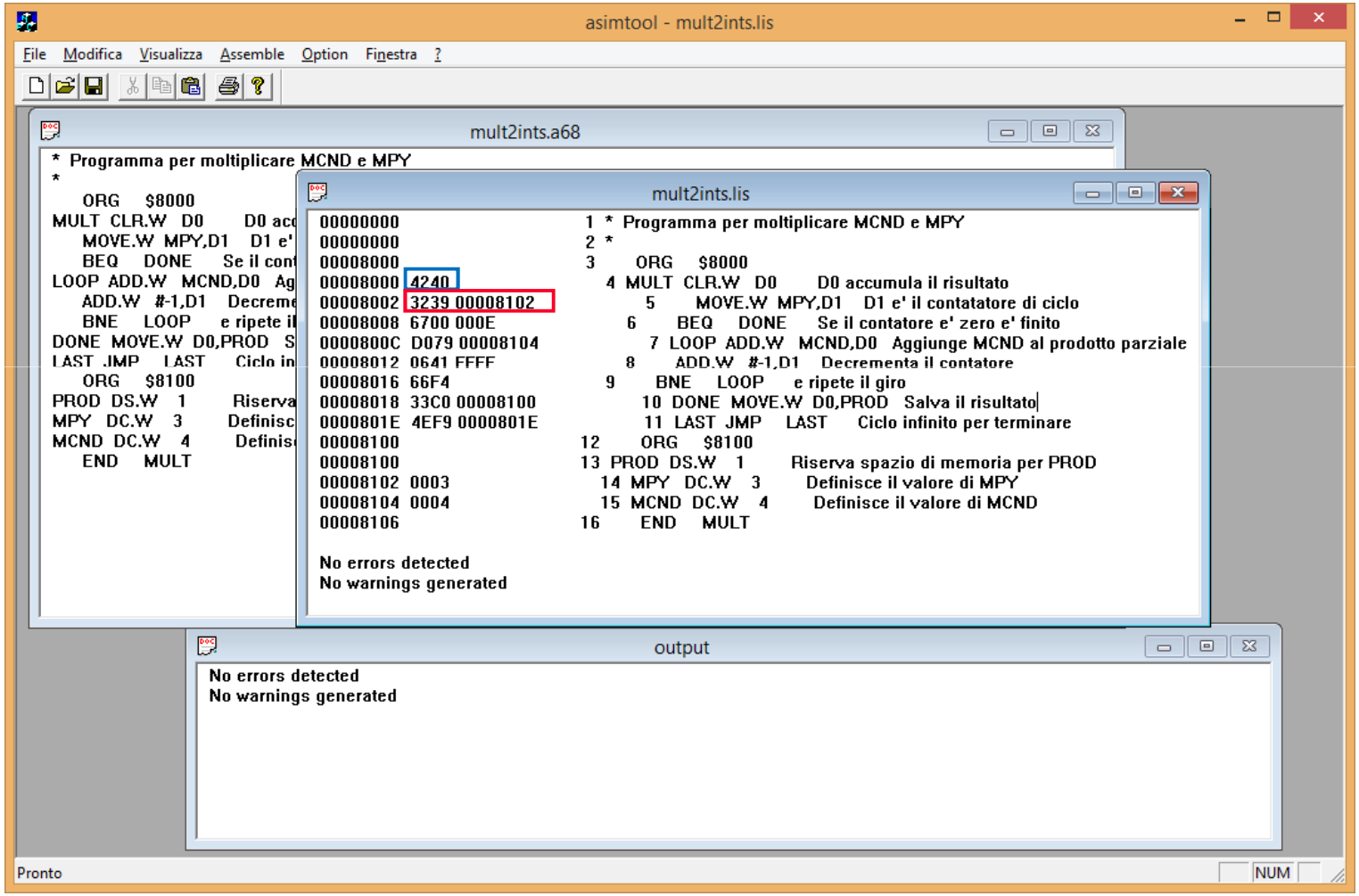
- Una direttiva EQU, invece, associa al simbolo usato come etichetta il valore specificato come operando di EQU

```
X   EQU    2
```

→ ad X è associato il valore 2

- Eventuali riferimenti in avanti sono risolti dall'assemblatore facendo una doppia scansione del codice sorgente (*assemblatori a due passi*)

AsimTool: text editor + assemblatore



ASIM: programma caricato in memoria