

---

---

# Programmazione Assembly

## Nuovo Corso di Calcolatori Elettronici I

Dipartimento di Informatica e Sistemistica  
Università degli Studi di Napoli "Federico II"

*DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli*



---

---

## Roadmap

- Il livello Assembly
- Formato del codice sorgente
- Modello di programmazione del processore MC68K
- Il flag dello Status Register
- Le istruzioni di salto condizionato
- La comparazione della memoria
- Traduzione delle strutture di selezione ed iterative di alto livello nel linguaggio Assembly.

*DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli*



## Linguaggi Assembly

- Per una data macchina, esiste sempre almeno il linguaggio assembly definito dal costruttore
- In aggiunta, possono esistere linguaggi assembly forniti da terze parti
- Quando si definisce un linguaggio assembly
  - » Si ha libertà di scelta per quanto riguarda:
    - ◆ Gli mnemonics
    - ◆ Il formato delle linee del sorgente
    - ◆ I formati per specificare modi di indirizzamento, varianti delle istruzioni, costanti, label, pseudo-operatori, etc.
  - » Non si ha libertà di scelta per quanto riguarda:
    - ◆ L'effetto finale di ogni singola istruzione macchina

*DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli*



## Formato del codice sorgente

- Una linea di codice sorgente Assembly è costituita da quattro campi:
  - » LABEL
    - Stringa alfanumerica
    - Definisce un nome simbolico per il corrispondente indirizzo
  - » OPCODE
    - Codice mnemonico o pseudo-operatore
    - Determina la generazione di un'istruzione in linguaggio macchina o la modifica del valore corrente del Program Location Counter
  - » OPERANDS
    - Oggetti dell'azione specificata dall'OPCODE
    - Variano a seconda dell'OPCODE e del modo di indirizzamento
  - » COMMENTS
    - Testo arbitrario inserito dal programmatore

*DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli*



## Convenzioni

- Gli spazi bianchi tra i diversi campi fungono esclusivamente da separatori (vengono ignorati dall'assemblatore)
- Una linea che inizi con un asterisco (\*) è una linea di commento
- Nelle espressioni assembly, gli argomenti di tipo numerico si intendono espressi
  - » In notazione decimale, se non diversamente specificato
  - » In notazione esadecimale, se preceduti dal simbolo "\$"
  - » In notazione binaria, se preceduti dal simbolo "%"
  - » In notazione ottale, se preceduti dal simbolo "@"
- Nell'indicazione degli operandi, il simbolo "#" denota un indirizzamento immediato

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



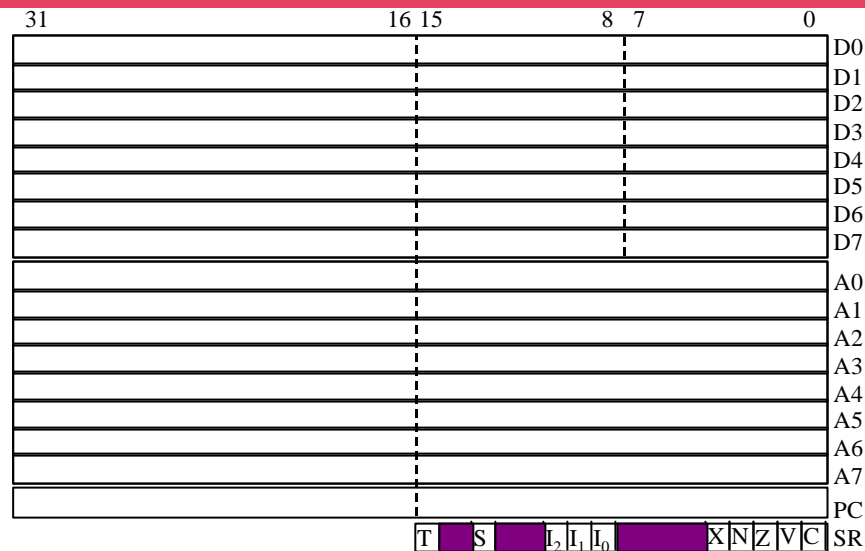
## Il Program Location Counter (PLC)

- E' una variabile interna dell'assemblatore
- Punta alla locazione di memoria in cui andrà caricata - a run time - l'istruzione assemblata
- Viene inizializzato dallo pseudo-operatore "origin" (ORG)
- Durante il processo di assemblaggio, il suo valore è aggiornato sia in funzione degli operatori, sia in funzione degli pseudo-operatori
- E' possibile, all'interno di un programma, fare riferimento al suo valore corrente, mediante il simbolo "\*" "

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## Modello di programmazione del MC68000



DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## 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

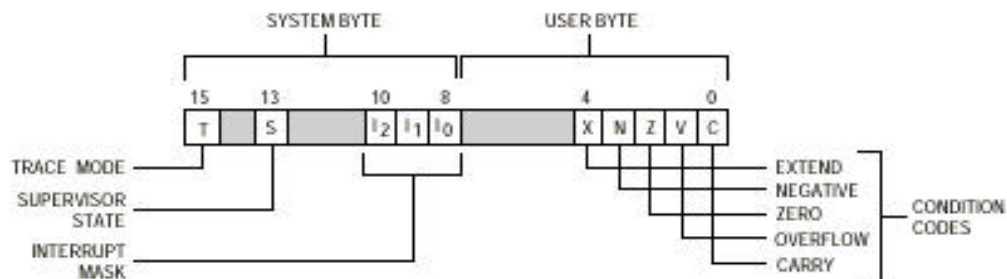


Figure 2-4. Status Register

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## I flag di condizione: significati.

- **C=Carry**: segnala un riporto/prestito uscente dal bit di maggior peso in operazioni aritmetiche;
- **V=overflow**: overflow in rappresentazione in complementi;
- **Z=Zero**: dato nullo;
- **N=Negative**: dato negativo;
- **X=extend**: copia il flag C in alcune istruzioni aritmetiche ed è usato per l'estensione ad aritmetica in precisione multipla.

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## Esempio - 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 contatore 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

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       Fine ass., salto a entry point
```

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## Osservazioni

- Eseguire il programma sul simulatore e sperimentare:
  - » L'effetto di DC e la rappresentazione esadecimale in memoria
  - » L'effetto dell'istruzione CLR su registro
  - » L'effetto dell'istruzione MOVE da memoria a registro
  - » L'effetto dell'istruzione BEQ sul PC
  - » L'effetto dell'istruzione ADD tra memoria e registro
  - » L'effetto dell'istruzione ADD tra immediato e registro
  - » L'effetto dell'istruzione BNE sul PC
  - » L'effetto dell'istruzione JMP sul PC
  - » L'effetto dell'istruzione MOVE da registro a memoria e la rappresentazione esadecimale in memoria

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## Esempio - Somma di n interi

\* Programma per sommare i primi 17 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

CNT          DS.W       1
SUM          DS.W       1
IVAL        EQU        17
ICNT        DC.W       IVAL
END          START
```

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## Bcc: Branch on condition cc

**Operation:** IF  $cc = 1$  THEN  $[PC] \leftarrow [PC] + d$   
**Syntax:** Bcc <label>  
**Attributes:** Bcc takes an 8-bit or 16-bit offset (displacement)

### Description:

If the specified logical condition is met, program execution continues at location  $[PC] + \text{displacement}$ ,  $d$ .

The displacement is a two's complement value.

The value in the PC corresponds to the current location plus two. The range of the branch is -126 to +128 bytes with an 8-bit offset, and -32K to +32K bytes with a 16-bit offset.

A short branch to the next instruction is impossible, since the branch code 0 indicates a long branch with a 16-bit offset.

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## Bcc: Branch on condition 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	$C + Z = 0$
BLO branch on less than	$C = 1$ (usa quindi lo stesso OpCode di BCS)
BHS branch on higher than or the same	$C = 0$ (usa quindi lo stesso OpCode di BCC)

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## CMP Compare

**Operation:** [destination] - [source]

**Syntax:** CMP <ea>, Dn

**Sample syntax:** CMP (Test, A6, D3.W), D2

**Attributes:** Size = byte, word, longword

**Description:**

Subtract the source operand from the destination operand and set the condition codes accordingly. The destination must be a data register. The destination is not modified by this instruction.

**Condition codes:**

X	N	Z	V	C
-	*	*	*	*

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## CMPM Compare memory with memory

**Operation:** [destination] - [source]

**Syntax:** CMPM (Ay)+, (Ax)+

**Attributes:** Size = byte, word, longword

**Description:**

Subtract the source operand from the destination operand and set the condition codes accordingly. The destination is not modified by this instruction. The only permitted addressing mode is address register indirect with post-incrementing for both source and destination operands.

**Application:**

Used to compare the contents of two blocks of memory.

**Condition codes:**

X	N	Z	V	C
-	*	*	*	*

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## DBcc Test condition, decrement, and branch

**Operation:** if (condition false) then  
                  [Dn] ← [Dn] - 1     {decrement loop counter}  
                  if [Dn] = -1 then  
                      [PC] ← [PC] + 2 {goto next instruction}  
                      else     [PC] ← [PC] + d {take branch}  
                      else     [PC] ← [PC] + 2 {goto next instruction}

**Syntax:** DBcc Dn, <label>

**Attributes:**     Size = word

**Description:** The DBcc instruction provides an automatic looping facility and replaces the usual decrement counter, test, and branch instructions. Three parameters are required by the DBcc instruction: a

branch condition (specified by 'cc'), a data register that serves as the loop down-counter, and a label that indicates the start of the loop. The 14 branch conditions supported by Bcc are also supported by DBcc, as well as DBF and DBT (F = false, and T = true). Note that many assemblers permit the mnemonic DBF to be expressed as DBRA (i.e., decrement and branch back).

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## SUBQ Subtract quick

**Operation:** [destination] ← [destination] - <literal>

**Syntax:** SUBQ #<data>, <ea>

**Attributes:**     Size = byte, word, longword

**Description:**

Subtract the immediate data from the destination operand.

The immediate data must be in the range 1 to 8.

Word and longword operations on address registers do not affect condition codes. A word operation on an address register affects the entire 32-bit address.

**Condition codes:**

X	N	Z	V	C
*	*	*	*	*

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## Problema

- Scrivere un programma che sommi i primi n interi
- Assemblare ed eseguire il programma sul simulatore
- Sperimentare:
  - » L'effetto dell'istruzione CLR in memoria
  - » L'effetto dell'istruzione MOVE da memoria a registro
  - » L'effetto dell'istruzione ADD tra memoria e registro
  - » L'effetto delle varie istruzioni sui codici di condizione
  - » L'effetto dell'istruzione BNE sul PC
  - » L'effetto dell'istruzione JMP sul PC

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## Soluzione - Codice

```
* Programma per sommare i primi 17 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  
                JMP         SYSA  
SYSA          EQU          $8008  
CNT           DS.W         1  
SUM           DS.W         1  
IVAL         EQU          17  
ICNT          DC.W         IVAL  
                END          START
```

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## Istruzioni di selezione in assembler – 1/2

### Linguaggio di alto livello:

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

NOTA: istruzione può anche essere un blocco di istruzioni.

### 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

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## Istruzioni di selezione in assembler – 2/2

### Linguaggio di alto livello:

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

### Esempio:

```
if (D0==5)
    D1++;
else
    D1--;
D0=D2;
```

### Linguaggio assembler (processore MC 68000):

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

### Esempio:

```
                CMP.L      #5,D0
                BNE       L1
                ADDQ.L    #1,D1
                BRA       L2
L1              SUBQ.L    #1,D1
L2              MOVE.L    D2,D0
```

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## Strutture iterative in assembler - 1/2

### 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
```

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## Strutture iterative in assembler - 2/2

### 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;
while (D1 <= N) {
    D0 = D0 * 3;
    D1++;
};
```

```
MOVE.B #N,D2
MOVE.B #1,D1
MOVE.W #1,D0
      BRA  TEST
LOOP   MULU.W #3,D0
      ADDQ.B #1,D1
TEST   CMP.B D2,D1
      BLE  LOOP
```

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## 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
<b>DBRA</b>	<b>D1,LOOP</b>	<b>SUBQ</b>	<b>#1,D1</b>
MOVE.L	D0,SOMMA	<b>BGE</b>	<b>LOOP</b>
		MOVE.L	D0,SOMMA

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## Problema

- Scrivere un programma che esegua il prodotto scalare tra due vettori di interi
- Assemblare ed eseguire il programma sul simulatore
- Sperimentare:
  - » L'effetto dell'istruzione MOVEA anziché MOVE.L nelle prime due istruzioni

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## Soluzione - Codice

```
ORG      $8000
START MOVE.L #A,A0
        MOVE.L #B,A1
        MOVE  #N,D0
        SUBQ  #1,D0
        CLR   D2
LOOP  MOVE (A0)+,D1
      MULS (A1)+,D1
      ADD  D1,D2
      DBRA D0,LOOP
      MOVE D2,C
      BREAK
N      EQU   $000A
      ORG   $80B0
A      DC   1,1,1,1,1,1,1,1,1,1,1
      ORG   $80D0
B      DC   1,1,1,1,1,1,1,1,1,1,1
C      DS.L 1
```

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## Problema

- Scrivere un programma che:
  - » Cerchi un token in una stringa
  - » Trovato il token, esca
- Assemblare ed eseguire il programma sul simulatore

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli



## Soluzione - Codice

```
* File: token.a68 - Find token in string
TOKEN EQU      ':'
          ORG      $8000
START MOVEA.L   #STRING,A0
          MOVEA.L   #0,A1
          MOVE.B    #TOKEN,D0
LOOP    CMP.B   #0,(A0)
          BEQ     END
          CMP.B   (A0)+,D0
          BNE    LOOP
          SUBQ.L  #1,A0
          MOVEA.L A0,A1
END     BREAK
          ORG      $8100
STRING DC.B    'QUI QUO:QUA'
TAPPO  DC.B    $0
```

DIS - Dipartimento di Informatica e Sistemistica - Università di Napoli

