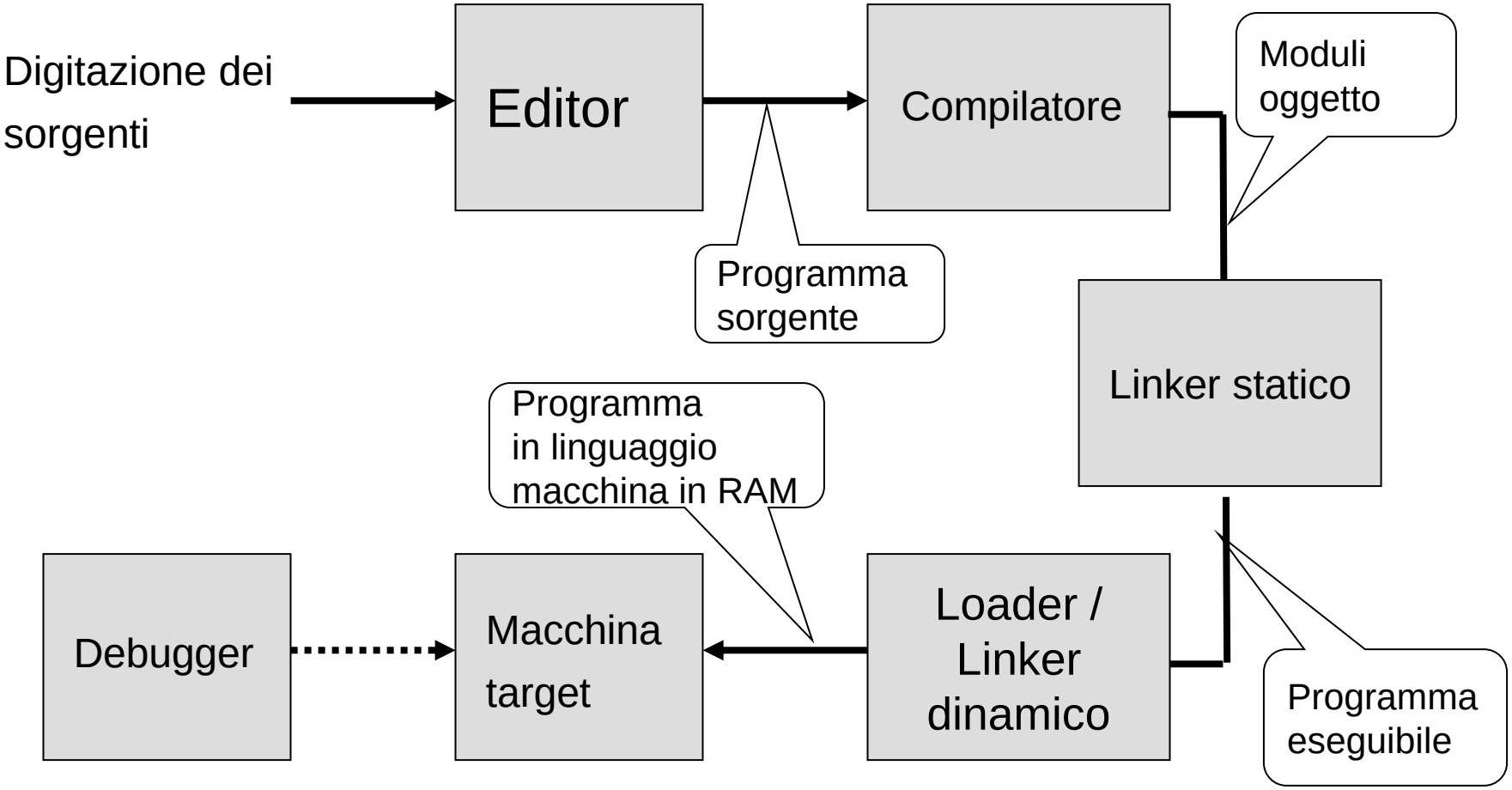
Corso di Calcolatori Elettronici I

Introduzione ai linguaggi assembly

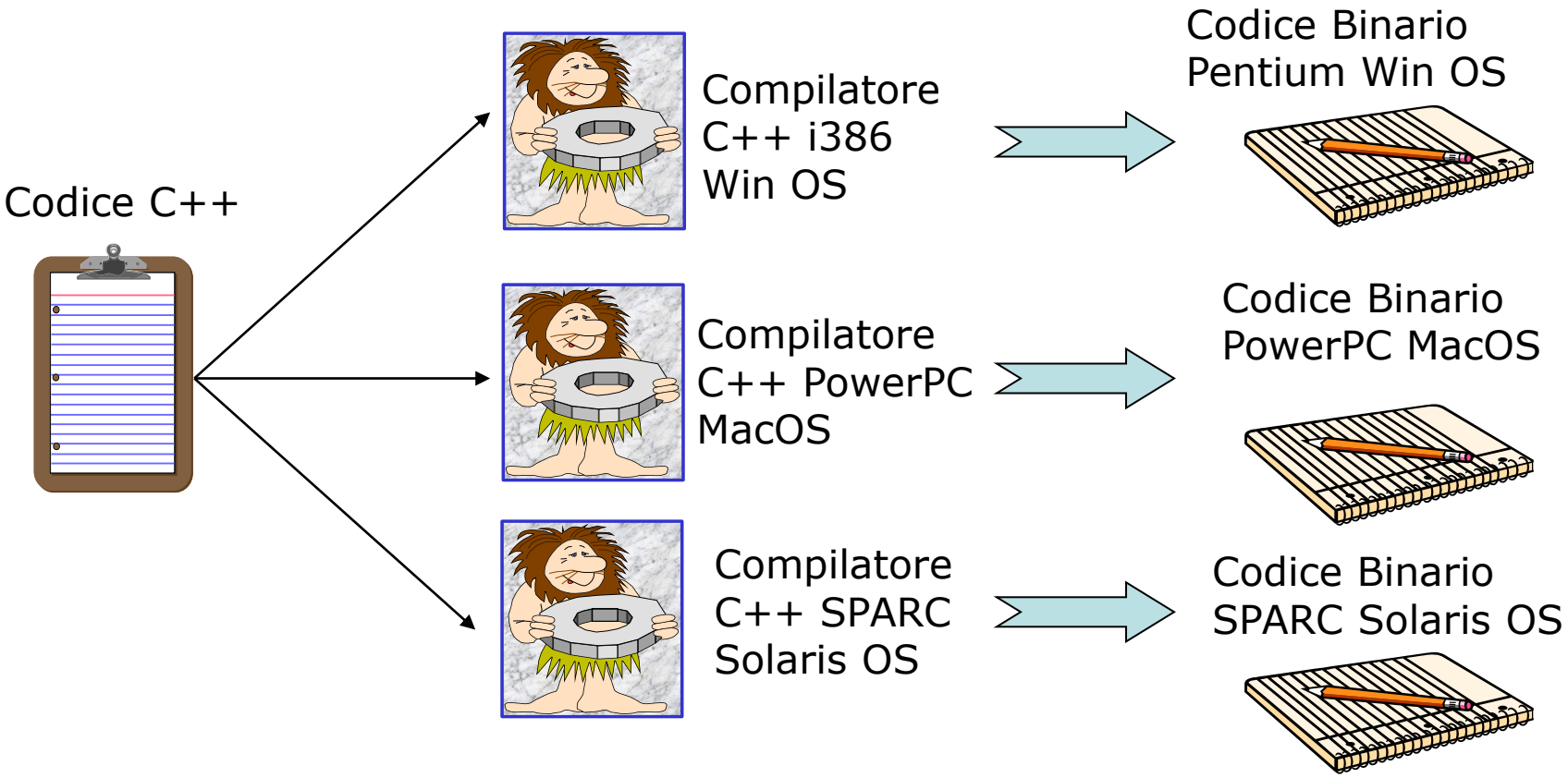
Prof. Roberto Canonico

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

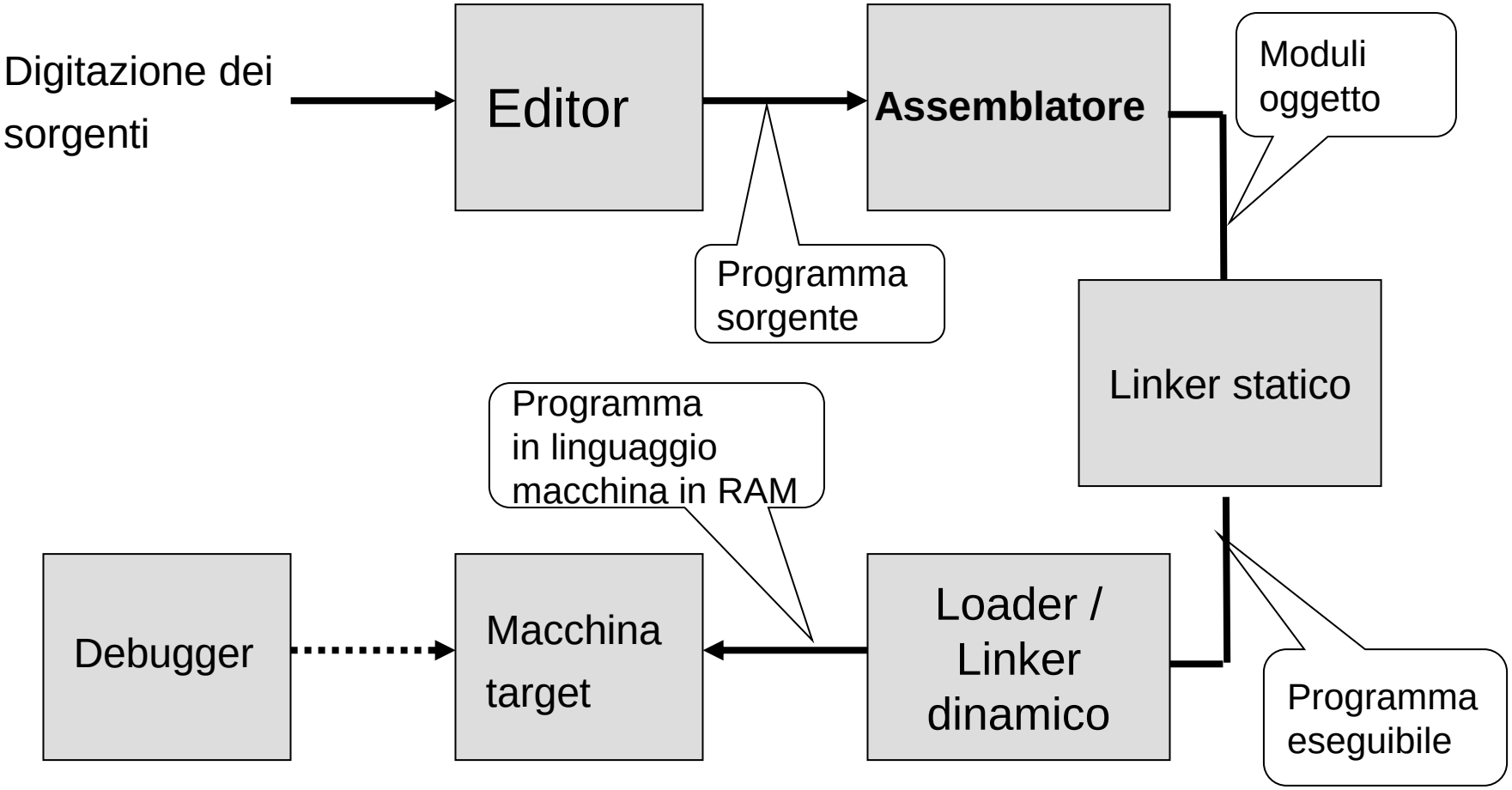
Ciclo di sviluppo/esecuzione per programmi in linguaggio di alto livello



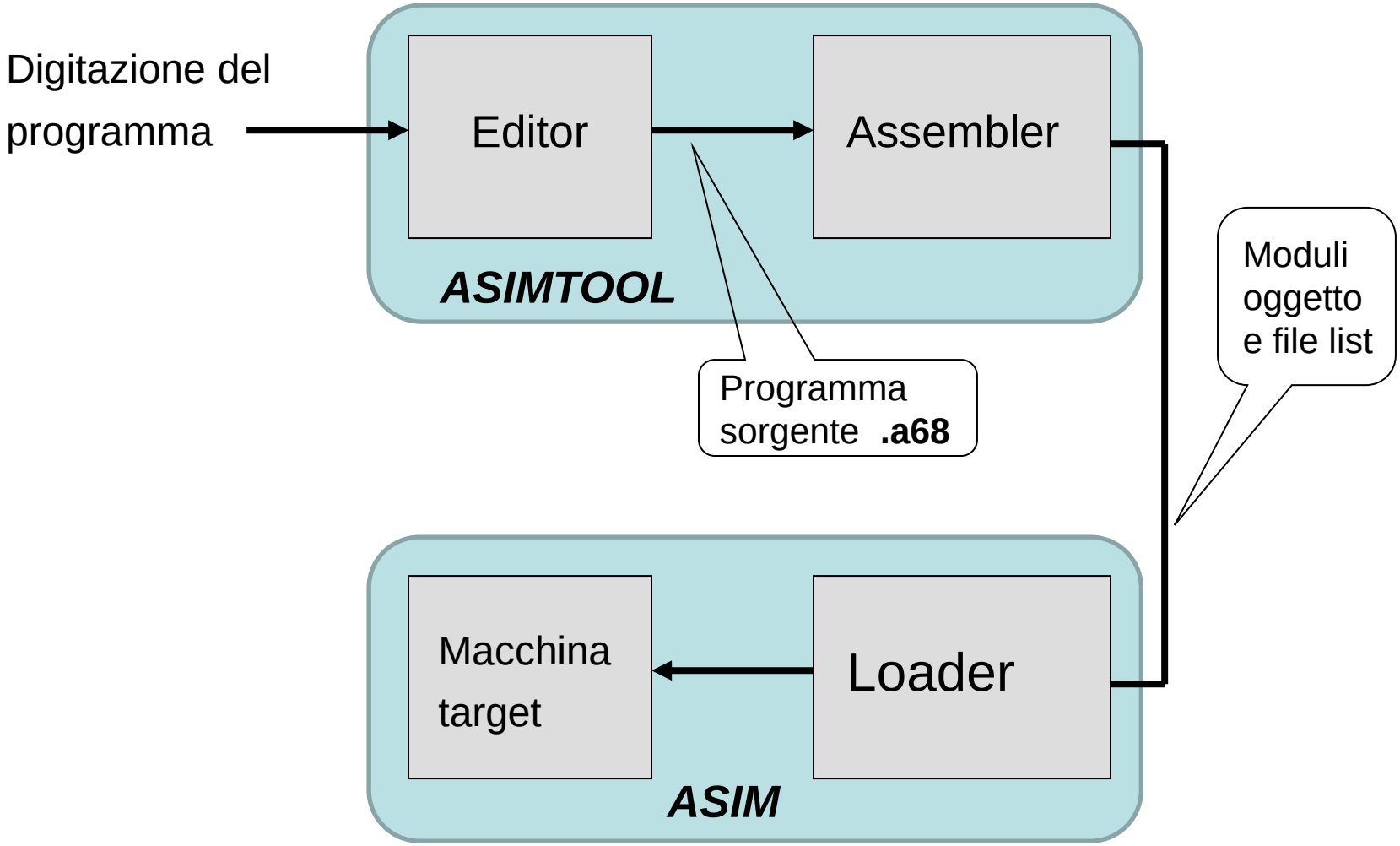
Linguaggi e dipendenza dalla piattaforma di esecuzione



Ciclo di sviluppo/esecuzione per programmi in linguaggio assembly



Ciclo di sviluppo semplificato di programmi assembly MC68000 nel sistema didattico ASIM



Assembly: 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
  - MNEMONIC
    - Determina la generazione di un'istruzione in linguaggio macchina e la modifica del valore corrente del Program Location Counter
    - Può, in alternativa, essere una *direttiva di assemblaggio*
  - OPERANDS
    - Oggetti dell'azione specificata dal codice MNEMONIC
    - Variano a seconda dell'istruzione e del modo di indirizzamento
  - COMMENTS
    - Testo arbitrario inserito dal programmatore

Assembly: caratteristiche generali

- Di regola, una linea di codice assembly corrisponde ad una istruzione l/m
- Eccezioni:
  - Macro: 1 linea assembler $\rightarrow$ n istruzioni l/m
  - Pseudo istruzioni: 1 linea assembler $\rightarrow$ 0 istr. l/m
- Variabili interamente gestite dal programmatore
  - Allocazione: memoria o registri CPU
  - No dichiarazione

Esempio – Assembly X86 a 32 bit

```
DES_std_crypt:
        movl 4(%esp),%edx
        pushl %ebx
        movl DES_count,%ecx
        xorl %ebx,%ebx
        movq (%edx),K1
        movq 32(%edx),K2
        movq K1,tmp1
        movq 8(%edx),K3
        movq 16(%edx),K4
        DES_copy(24, 40)
        …
        DES_copy(112, 120)
        movq DES_IV,R
        xorl %edx,%edx
        movq DES_IV+8,L
DES_loop:
        …
```

Esempio – Assembly Alpha

```
DES_std_crypt:
        ldgp $29,0($27)
DES_std_crypt..ng:
        subq $30,56,$30
        lda tmp1,DES_IV
        lda tmp2,DES_count
        lda SPE,DES_SPE_F
        ldq R,0(tmp1)
        ldq L,8(tmp1)
        ldq count,0(tmp2)
        ldq K1,0(kp)
        ldq K2,8(kp)
        ldq K3,16(kp)
        ldq K4,24(kp)
        xor K1,R,D
        ldq K5,32(kp)
        ldq K6,40(kp)
        ldq K7,48(kp)
        …
        ldq K8,56(kp)
        stq K9,0($30)
        stq K10,8($30)
        stq K11,16($30)
        stq K12,24($30)
        stq K13,32($30)
        stq K14,40($30)
        stq K15,48($30)
        ldq K9,64(kp)
        ldq K10,72(kp)
        ldq K11,80(kp)
        ldq K12,88(kp)
        ldq K13,96(kp)
        ldq K14,104(kp)
        ldq K15,112(kp)
        ldq K16,120(kp)
DES_loop:
        DES_2_ROUNDS(K2,K3)
        …
```

Esempio – Assembly SPARC

```
DES_std_crypt:
        …
        save %sp,-120,%sp
        st %i7,[%fp-24]
        sethi %hi(DES_SPE_L),SPE_L_0
        sethi %hi(DES_SPE_L+0x400),SPE_L_4
        add SPE_L_0,0x808,SPE_H_0
        …
        ldd [kp],Dl
        ldd [SPE_L_4+0xC08],Rl
        …
        ld  [SPE_L_4+0xC18],count
DES_loop:
        DES_2_ROUNDS(kp)
        …
        std Rl,[out]
        std Ll,[out+8]
        ret
        restore
        …
```

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

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 "%"
- Nell'indicazione degli operandi, il simbolo "#" denota un indirizzamento immediato (valore costante scritto nel codice della istruzione)

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 “*”

Etichette (label)

- Sono stringhe di testo arbitrarie (opzionali) anteposte ad una istruzione o ad un dato all'interno del programma assembler
- Servono a riferirsi al particolare indirizzo che contiene quella istruzione o dato
  - usati per gestire i salti
  - usati per gestire variabili (manipolate nel programma assembler attraverso le loro etichette in maniera simile alle variabili di un linguaggio di programmazione di alto livello)
- Ad esempio:
  - ALOOP è un'etichetta usata per riferisti all'istruzione MOVE, SUM è una etichetta usata per gestire una variabile, mentre IVAL è una costante

```
ALOOP    MOVE.W      D0,CNT
         ADD.W       SUM,D0
   …   … …   …
SUM      DS.W        1
IVAL     EQU         17
   … … … … … …
```