

**Corso di Calcolatori Elettronici I**  
**A.A. 2012-2013**

---

---

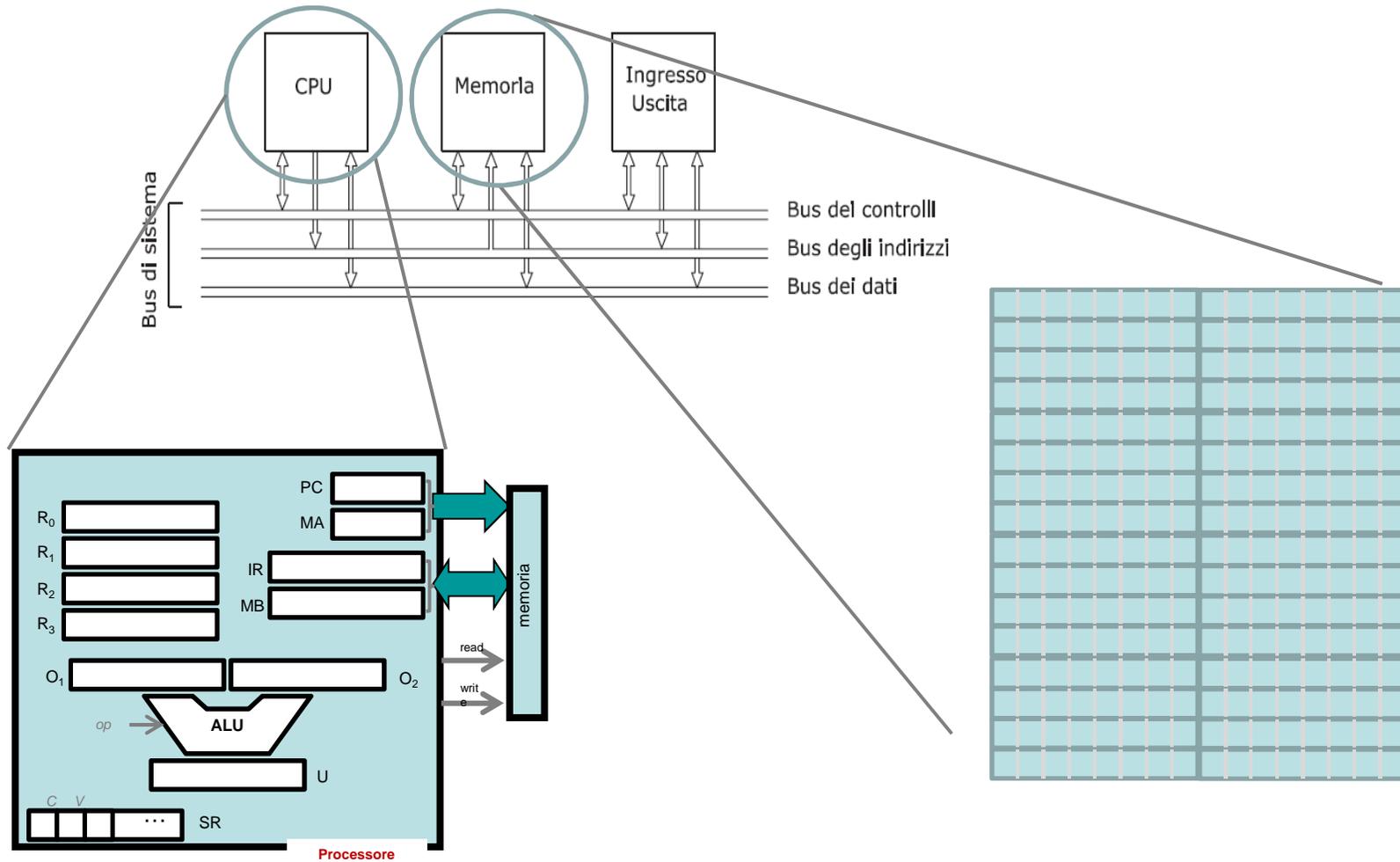
**Introduzione**  
**al linguaggio assembly**

**ing. Alessandro Cilardo**

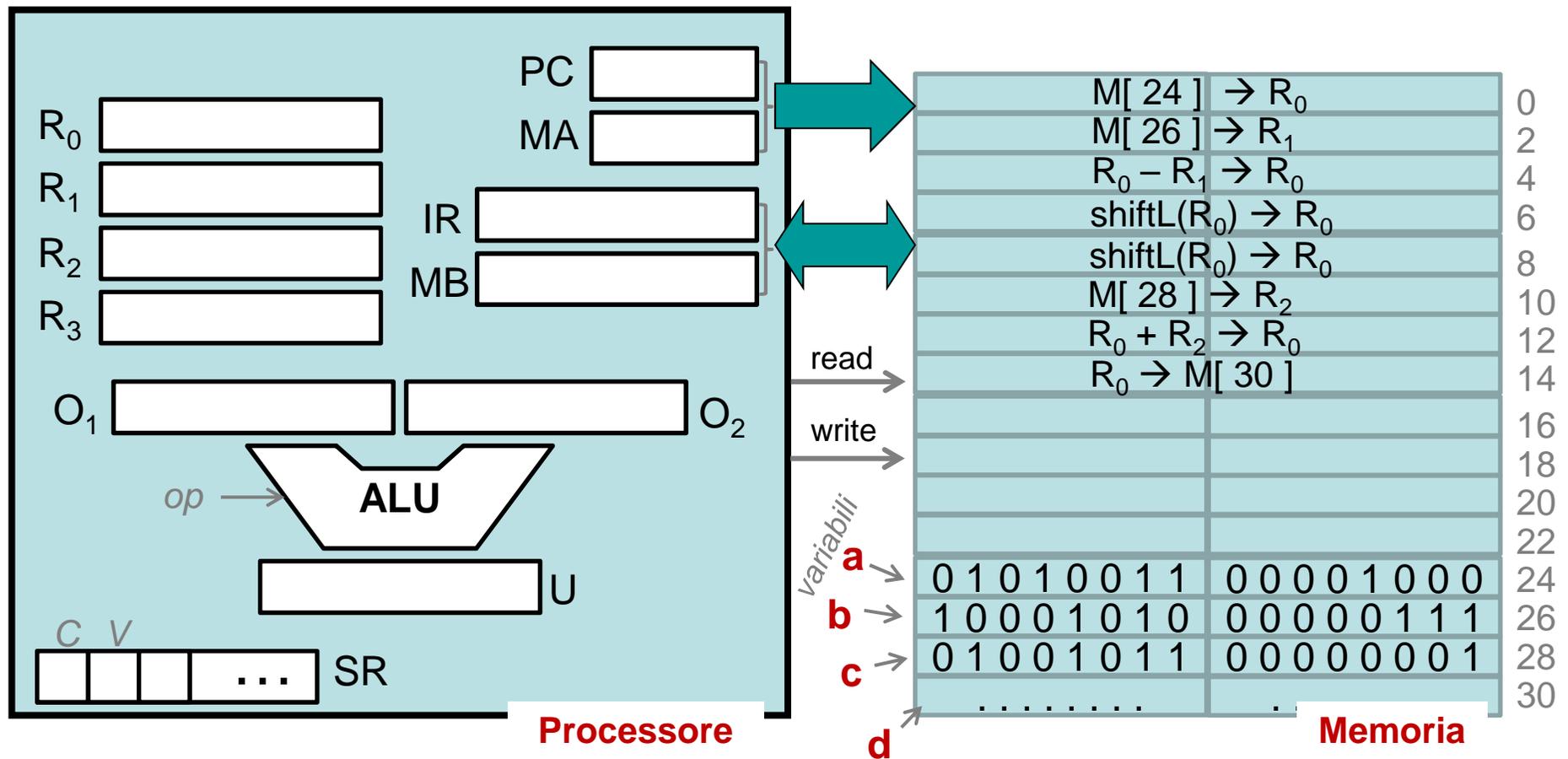
Accademia Aeronautica di Pozzuoli  
Corso Pegaso V "GArn Elettronici"

---

# Riepilogo



# Programma in memoria



# Codifica delle istruzioni

---

- Le istruzioni sono esse stesse informazioni da immagazzinare **in memoria** (come i dati)
- Il contenuto di un'istruzione è quindi oggetto di una **codifica**, che fa corrispondere una stringa di '0' e '1' ad ogni codice operativo e possibile combinazione di operandi

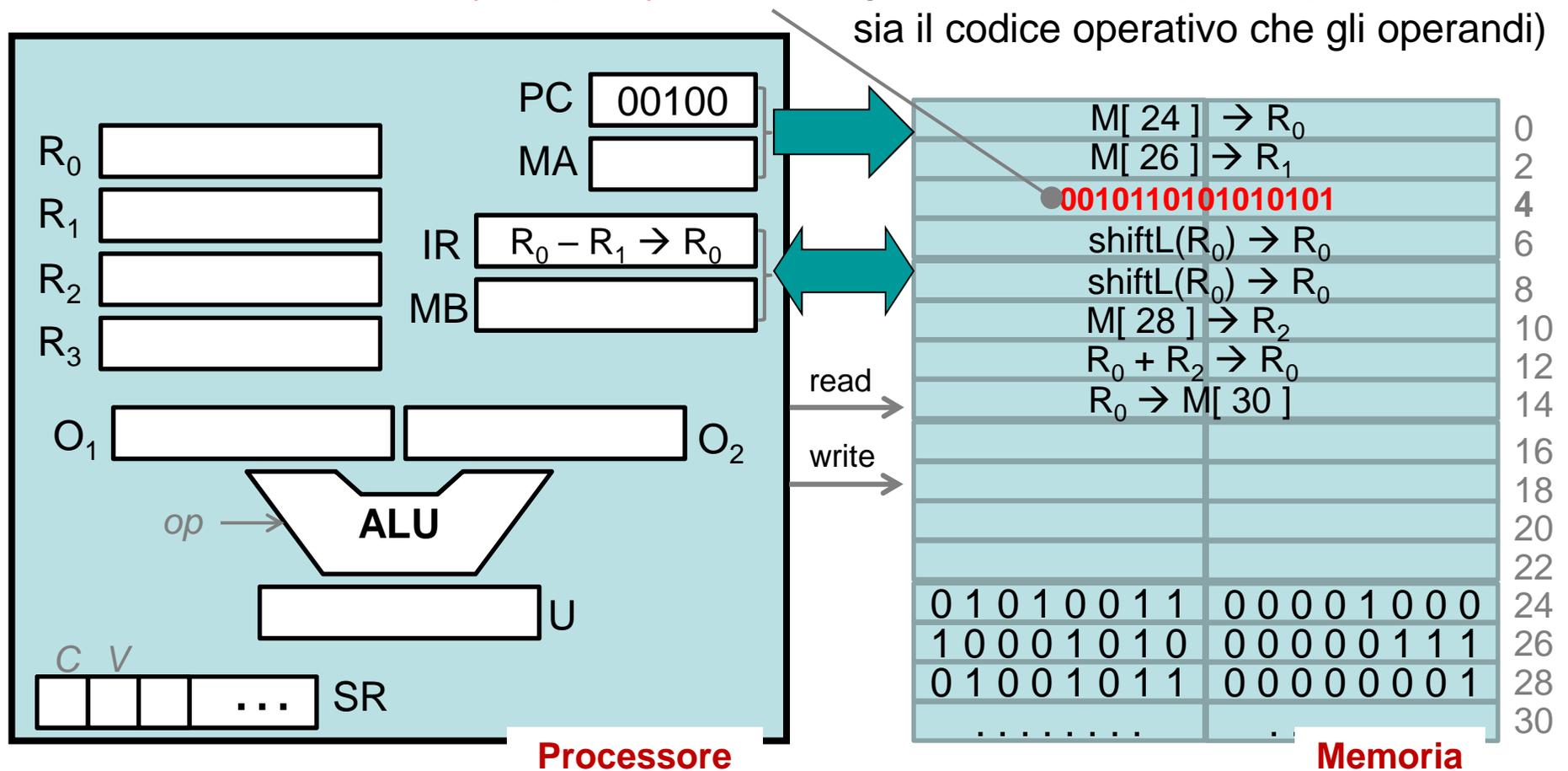
**ADD R<sub>3</sub>, R<sub>1</sub>, R<sub>0</sub>**       $\xrightarrow{\text{codifica}}$       **0010110101010101**

**SUB M[14], R<sub>2</sub>, R<sub>2</sub>**       $\xrightarrow{\text{codifica}}$       **1001010111101010**

- Tale informazione è codificata in macchina mediante
  - codici a *lunghezza fissa* (tipicamente 32 bit, es. RISC)
  - o a *lunghezza variabile* (es. nel 68000 multipli di 16 bit)

# Codifica delle istruzioni

In questo esempio, la locazione di memoria all'indirizzo 4 contiene materialmente la codifica dell'istruzione  $R_0 - R_1 \rightarrow R_0$  come stringa di 16 bit (usati per rappresentare sia il codice operativo che gli operandi)



# Codifica delle istruzioni

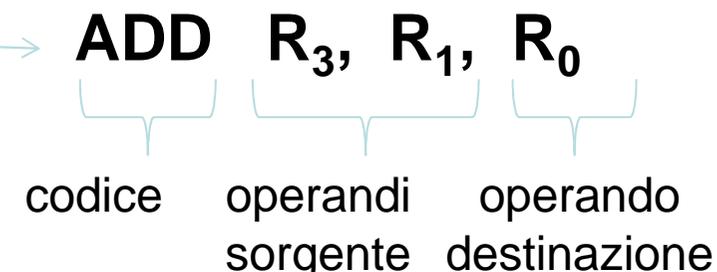
---

- Com'è strutturata *in generale* un'istruzione?
- E' in linea di principio una tripla  $i = (f, P_1, P_2)$ , dove:
  - $f \in F$  è l'insieme dei **codici operativi** del processore, cioè delle operazioni elementari definite al livello del linguaggio macchina;
  - $P_1$  è un insieme di **operandi-sorgente**, cioè di valori e/o indicazioni dei registri e/o indicazioni delle locazioni di memoria contenenti i valori su cui opera  $f$
  - $P_2$  è un insieme di **operandi-destinazione**, cioè di indicazioni dei registri o indicazioni delle locazioni di memoria cui sono destinati i risultati dell'istruzione  $f$
  - normalmente è presente un solo operando destinazione, che *può coincidere* con uno degli operandi sorgente

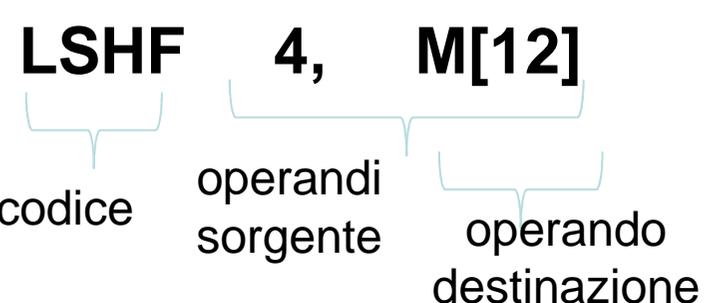
# Codifica delle istruzioni

---

---

- **ADD R<sub>3</sub>, R<sub>1</sub>, R<sub>0</sub>** → **ADD R<sub>3</sub>, R<sub>1</sub>, R<sub>0</sub>**  
calcola R<sub>3</sub>+R<sub>1</sub> e sposta la somma nel registro R<sub>0</sub>  


codice    operandi sorgente    operando destinazione
- **SUB M[14], R<sub>2</sub>, R<sub>2</sub>** → **SUB M[14], R<sub>2</sub>, R<sub>2</sub>**  
accedi alla memoria all'indirizzo 14, sottrai a questo valore il contenuto di R<sub>2</sub> e scrivi il risultato in R<sub>2</sub>  


codice    operandi sorgente    operando destinazione
- **LSHF 4, M[12]** → **LSHF 4, M[12]**  
trasla il contenuto della memoria all'indirizzo 12 di quattro posizioni a sinistra  


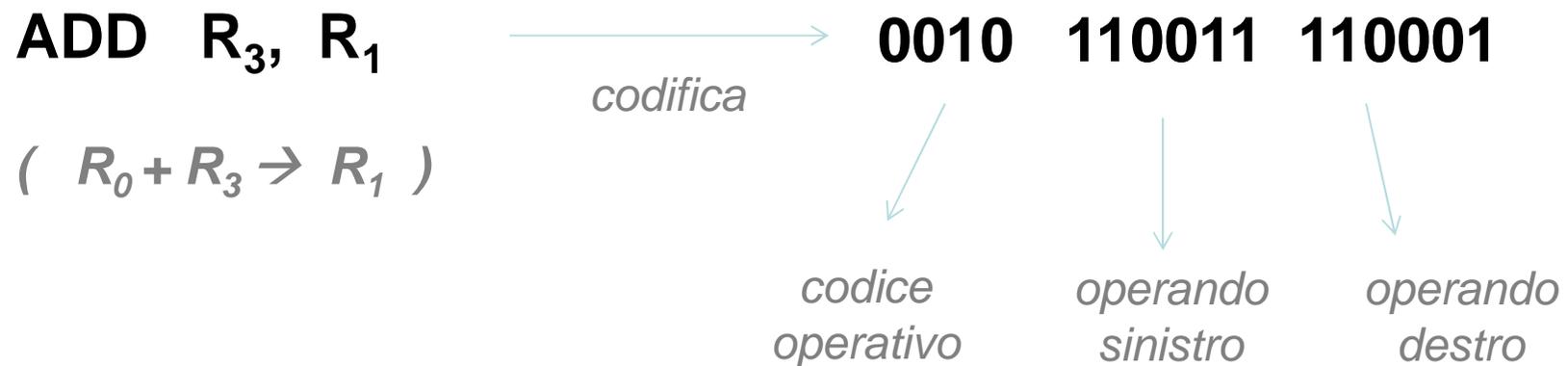
codice    operandi sorgente    operando destinazione

# Codifica delle istruzioni

---

---

- Esempio

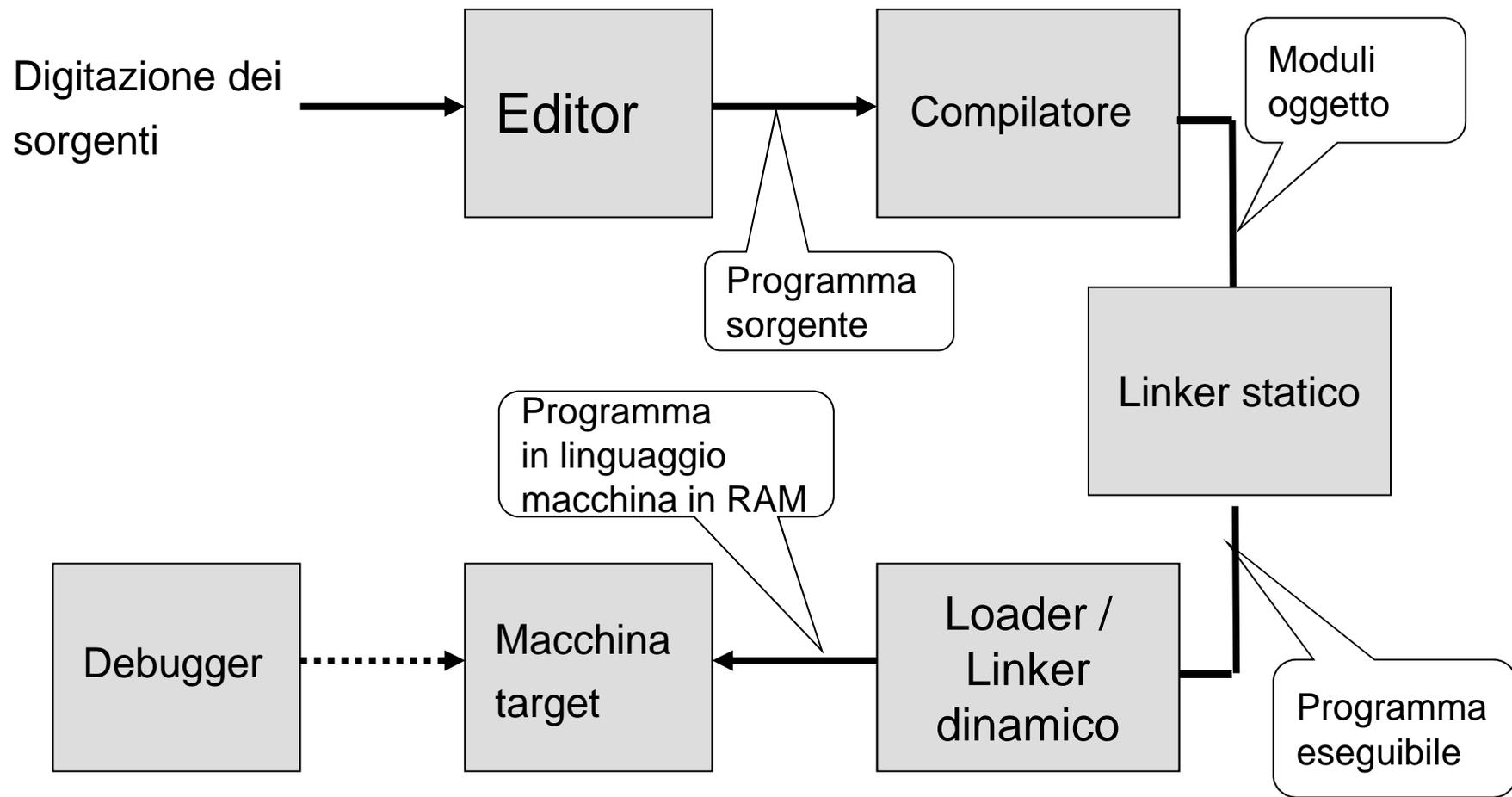


# Codifica delle istruzioni

---

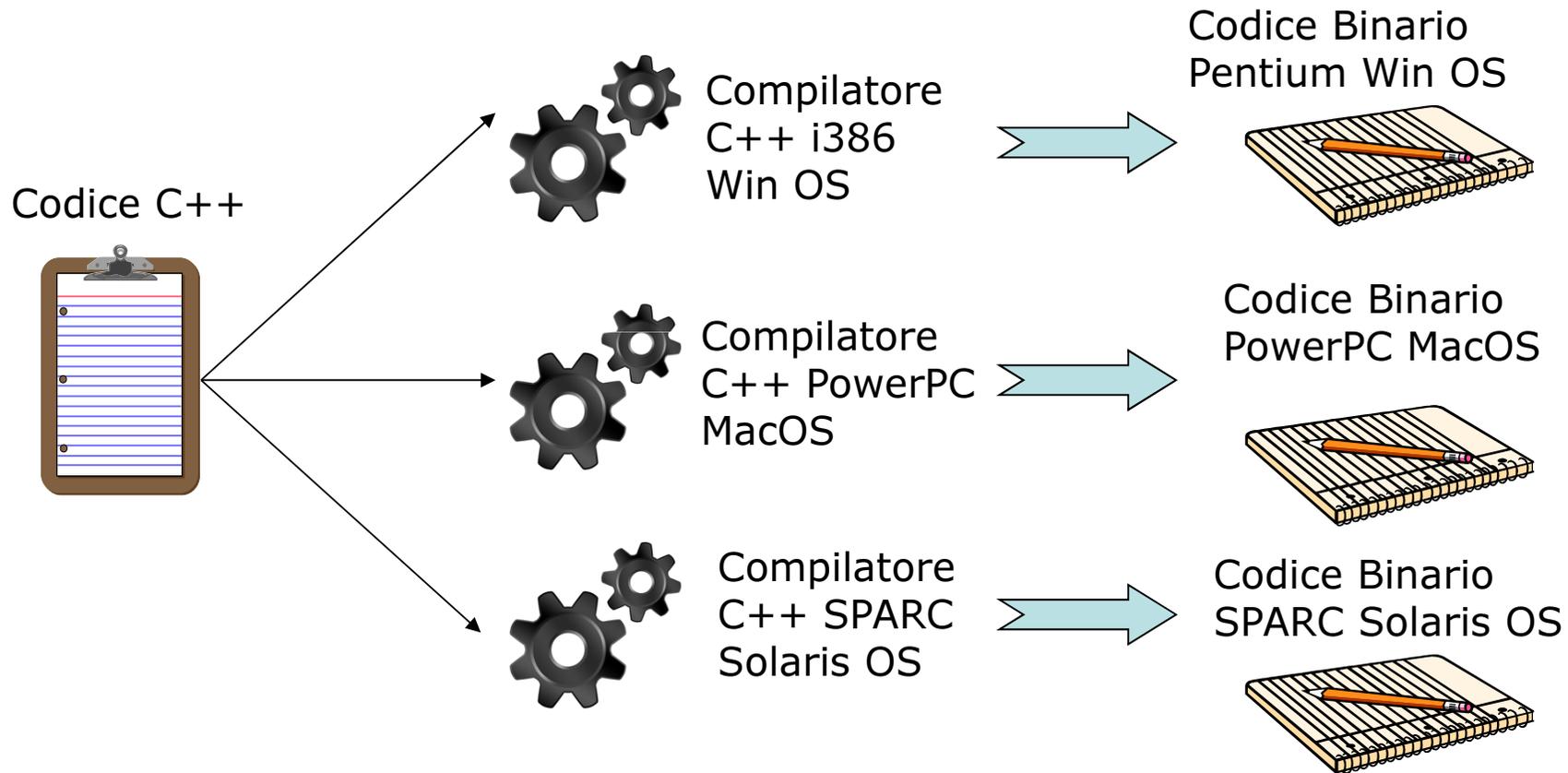
- Dobbiamo scrivere manualmente la codifica di ogni istruzione nel nostro programma?

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



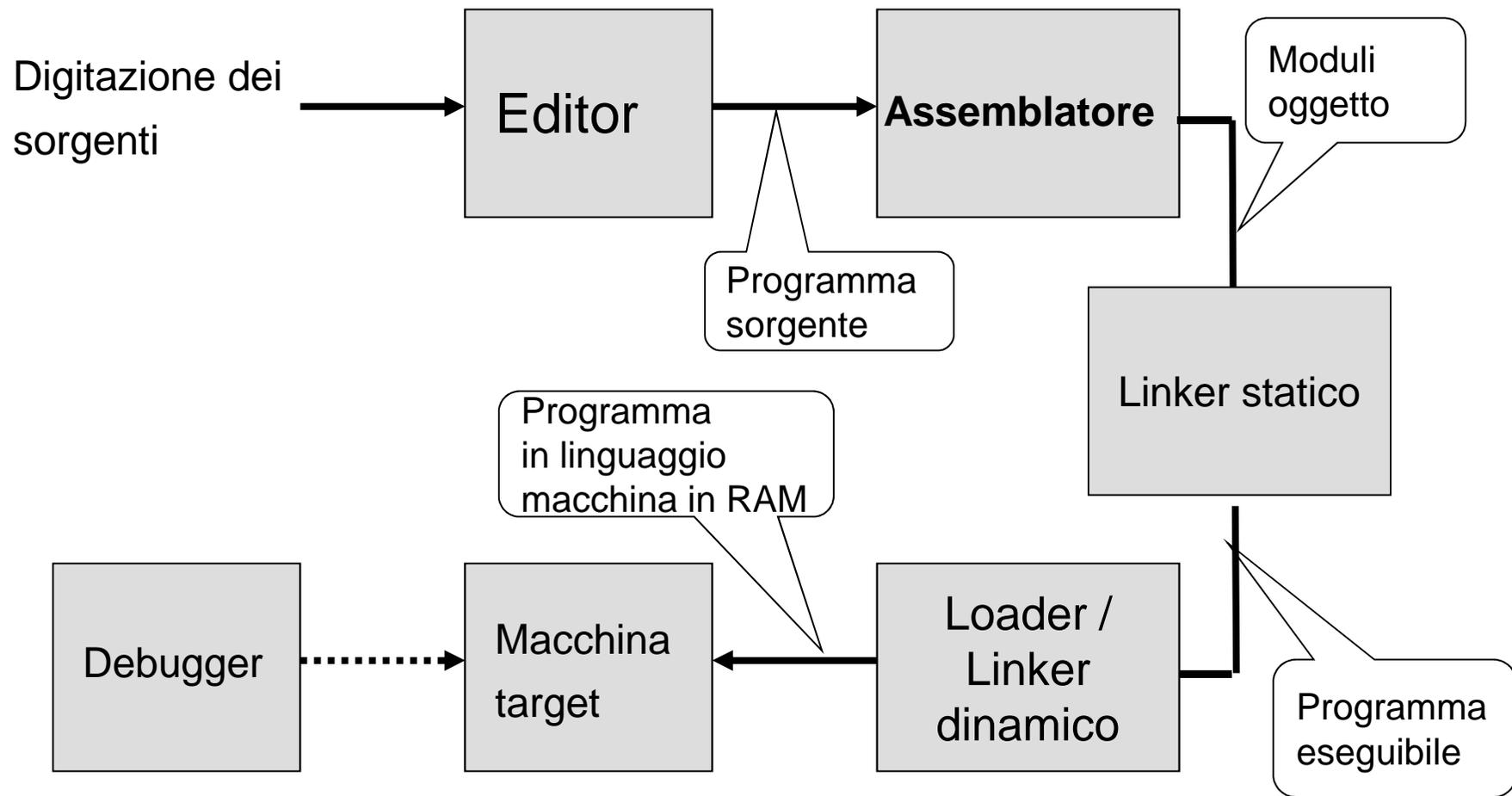
# Linguaggi e dipendenza dalla piattaforma di esecuzione

---

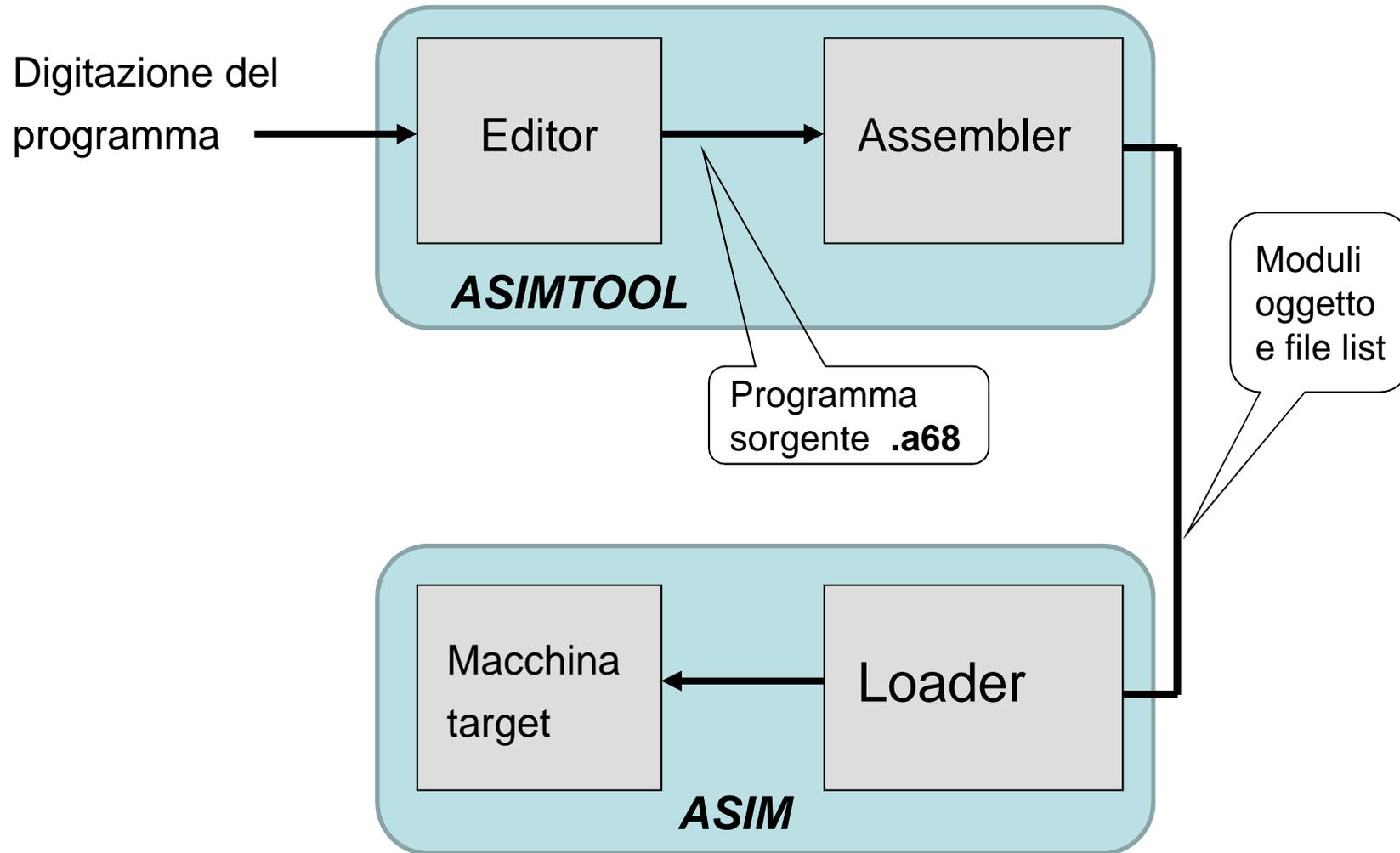


# Ciclo di sviluppo/esecuzione per programmi in linguaggio assembly

---



# 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 (esempio riferito al processore M68000):
  - **LABEL** (*opzionale*)
    - Stringa alfanumerica
    - Definisce un nome simbolico per il corrispondente indirizzo
  - carattere **TAB** + **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
  - carattere **TAB** + **OPERANDI**
    - Oggetti dell'azione specificata dall'OPCODE
    - Variano a seconda dell'OPCODE e del modo di indirizzamento
  - carattere **TAB** + **COMMENTI** (*opzionale*)
    - Testo arbitrario inserito dal programmatore

# Assembly: caratteristiche generali

---

---

- Di regola, una linea di codice assembly corrisponde ad una istruzione macchina
- Eccezioni:
  - Macro: 1 linea assembler → *diverse* istruzioni macchina
  - Pseudo istruzioni: 1 linea assembler → *nessuna* istruzioni macchina
- 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],D1
ldd [SPE_L_4+0xC08],R1
...
ld [SPE_L_4+0xC18],count
```

DES\_loop:

```
DES_2_ROUNDS(kp)
...
std R1,[out]
std L1,[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*** (i codici testuali usati per indicare le istruzioni)
    - 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 (M68000)

---

- 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 "\$"
- Nell'indicazione degli operandi, il simbolo "#" denota un indirizzamento immediato

# 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 “\*”

# Strumenti per il corso

---

- Compilatore/Assemblatore:  
**AsimTool**
- Simulatore:  
**ASIM**

# AsimTool

The screenshot shows the ASIM TOOL - OS.LIS application window. The main window displays assembly code for a program named 'os.a68: Simulates Operating System'. The code includes an 'ORG \$9000' directive and a 'LOOP' section with several instructions. A smaller window titled 'OS.LIS' is overlaid on top, showing the same code with line numbers and a status message: 'No errors detected' and 'No warnings generated'. The application's menu bar includes 'File', 'Edit', 'View', 'Assemble', 'Option', 'Window', and 'Help'. The status bar at the bottom indicates 'Ready'.

```
ASIM TOOL - OS.LIS
File Edit View Assemble Option Window Help
[Icons]

OS.A68
* os.a68: Simulates Operating System
*
  ORG $9000
LOOP
00000000          1 * os.a68: Simulates Operating System
00000000          2 *
00009000          3   ORG $9000
00009000          4
00009000 4E71      5 LOOP  NOP
00009002 4EF9 00009000 6   JMP  LOOP

No errors detected
No warnings generated

OS.LIS
1 * os.a68: Simulates Operating System
2 *
3   ORG $9000
4
5 LOOP  NOP
6   JMP  LOOP

No errors detected
No warnings generated

Ready
```

# AsimTool: esempio di file list

PLC	contenuto	label	opcode	operands	comments
00000000		1	*		Somma 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	11		BNE	ALOOP
0000802A	4EF9 00008008	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

# ASIM: programma caricato in memoria

ASIM - [simple: Memoria 2]

File Memory View Simulation Window Tools Help

00008000	42	79	00	00	80	32	30	39	00	00	80	34	33	C0	00	00	80	30	D0	79	00	00	80	32	33	C0	00	00	80
00008010	32	30	39	00	00	80	30	06	40	FF	FF	66	E2	4E	F9	00	00	80	08	00	00	00	00	00	11	00	00	00	00
0000803A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008057	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008074	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008091	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000080AE	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000080CB	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000080E8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008105	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008122	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000813F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000815C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008179	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008196	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000081B3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000081D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000081ED	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000820A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008227	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008244	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008261	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000827E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000829B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000082B8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000082D5	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000082F2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000830F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000832C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008349	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008366	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008383	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

For Help, press F1

# Pseudo-operatori

---

---

- NON sono istruzioni eseguite dal processore
  - sono direttive che regolano il processo di traduzione del programma assembler in programma eseguibile
- Lo pseudo-operatore ORG
  - Viene usato per inizializzare il Program Location Counter (PLC), ovvero per indicare a quale indirizzo sarà posta la successiva sezione di codice o dati
  - **Esempio:** ORG \$8100
- Lo pseudo-operatore END
  - Viene usato per terminare il processo di assemblaggio ed impostare l'*entry-point* (prima istruzione da eseguire) nel programma
  - **Esempio:** END TARGETLAB

# Pseudo-operatori

---

- Lo pseudo-operatore DS
  - Viene usato per incrementare il Program Location Counter (PLC), in modo da riservare spazio di memoria per una variabile
  - **Esempio:** LABEL DS.W NUMSKIPS
- Lo pseudo-operatore DC
  - Viene usato per inizializzare il valore di una variabile
  - **Esempio:** LABEL DC.W VALUE
- Lo pseudo-operatore EQU
  - Viene usato per definire una costante usata nel sorgente assembler
  - **Esempio:** LABEL EQU VALUE

# 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 riferirsi 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
          ... ..
```