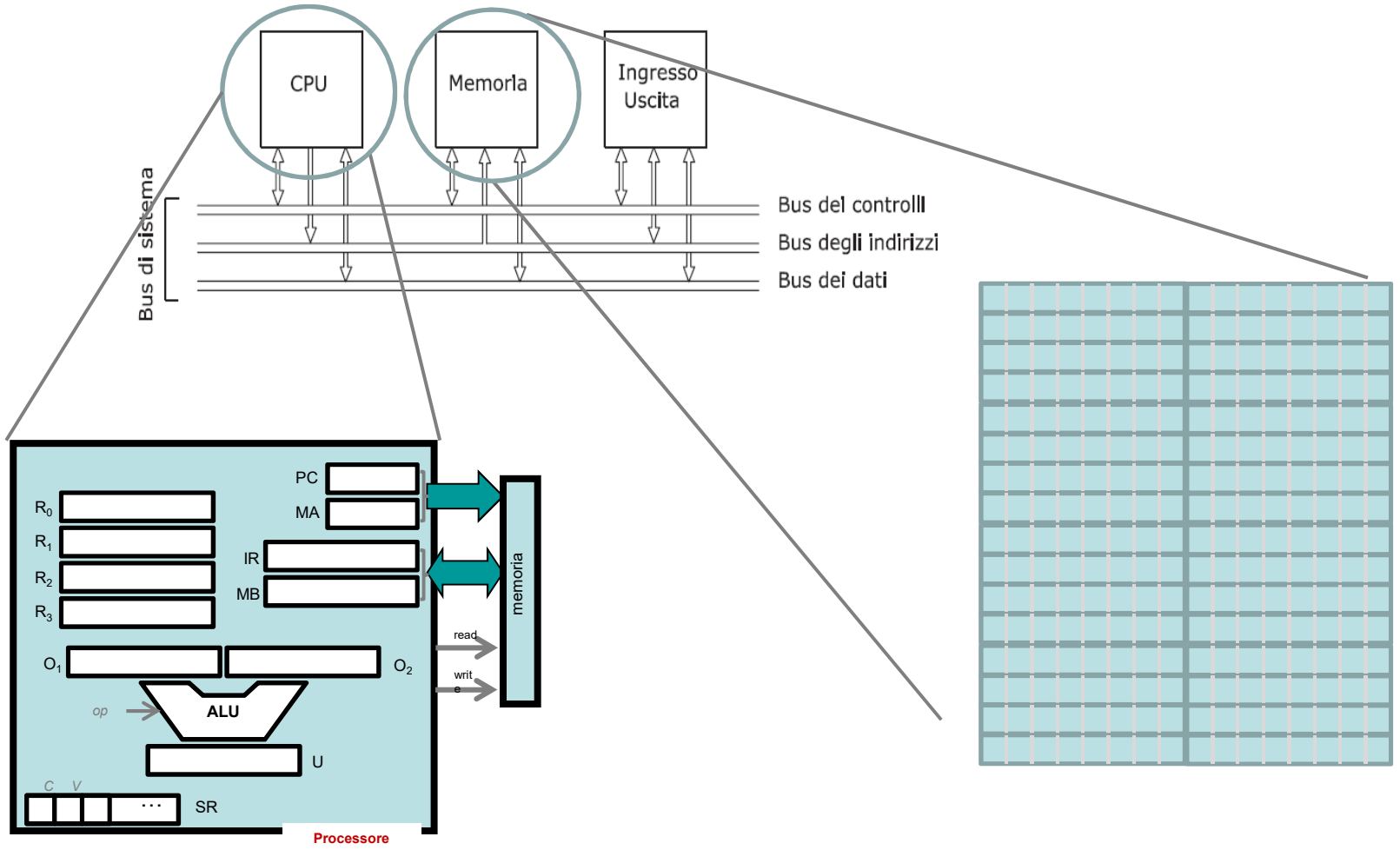
Corso di Calcolatori Elettronici I

# Introduzione al linguaggio assembly

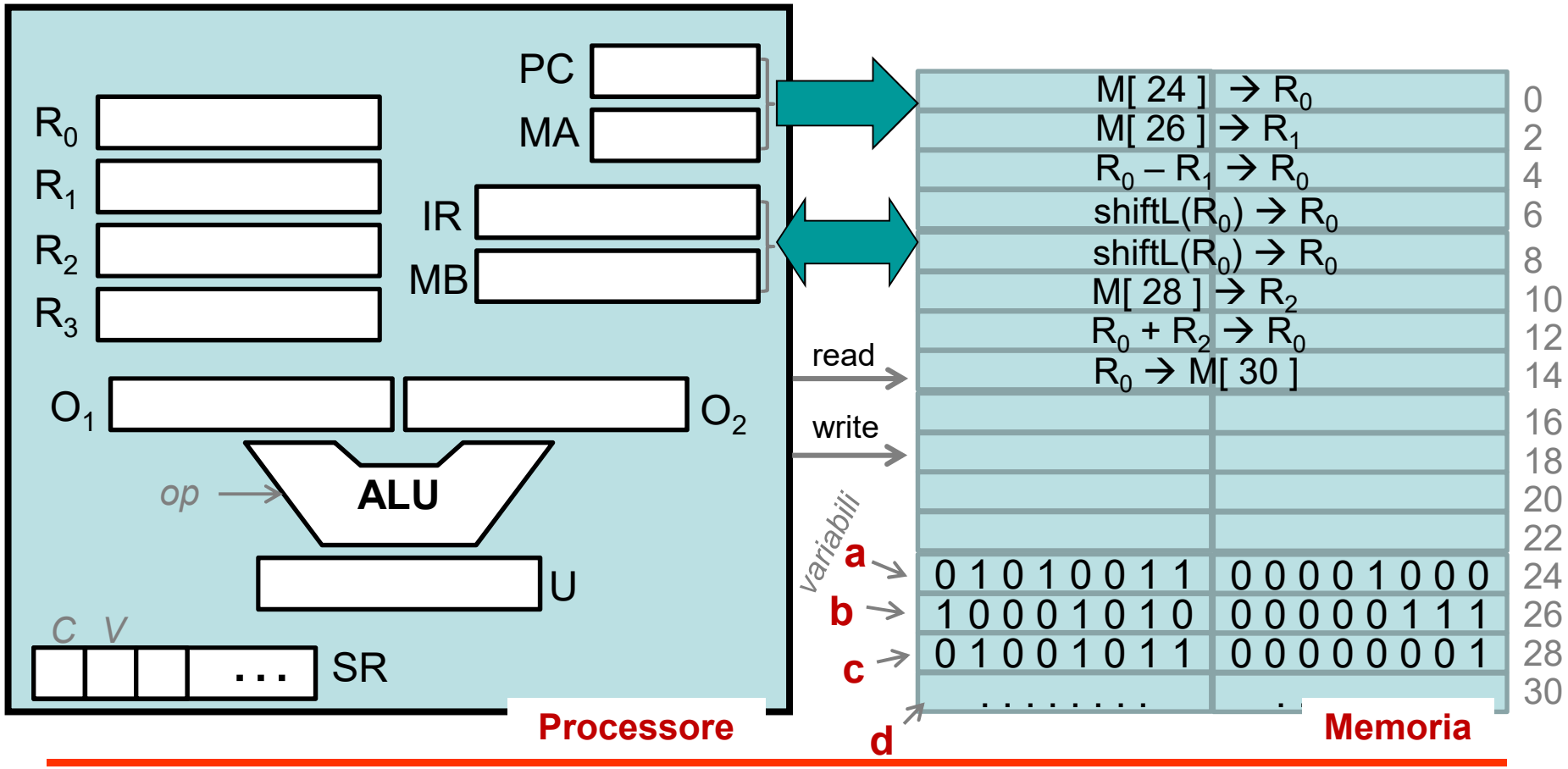
**ing. Alessandro Cilardo**

Corso di Laurea in Ingegneria Biomedica

# Riepilogo


CPU
Memoria
Ingresso
Uscita
Bus di sistema
Bus del controlll
Bus degli indirizzi
Bus dei dati
PC
MA
IR
MB
R0
R1
R2
R3
O1
O2
op
ALU
U
C
V
SR
read
writ
e
memoria
Processore

# 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

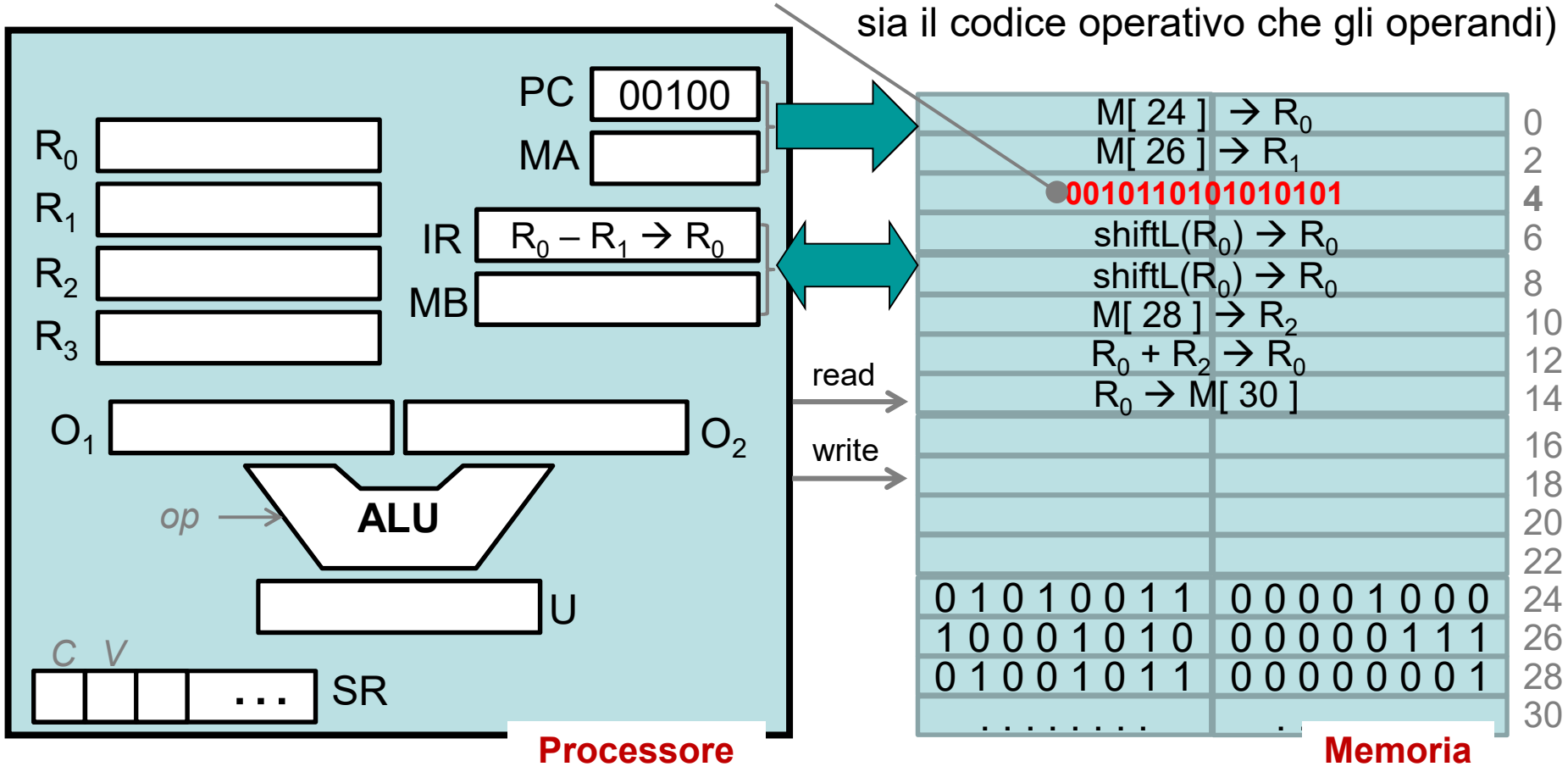
**ADD $R_3$, $R_1$, $R_0$** → *codifica* → **0010110101010101**

**SUB M[14], $R_2$, $R_2$** → *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)



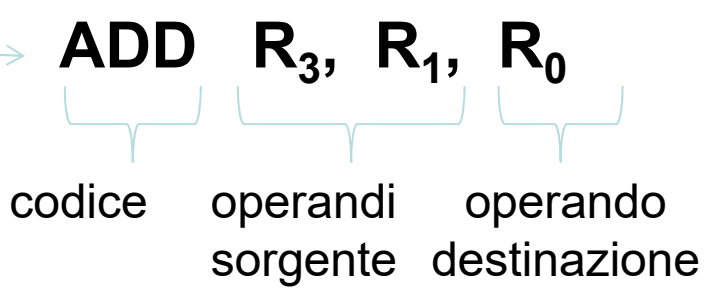
# Codifica delle istruzioni

- 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 $R_3$, $R_1$, $R_0$**

  calcola $R_3$+$R_1$ e sposta la somma nel registro $R_0$

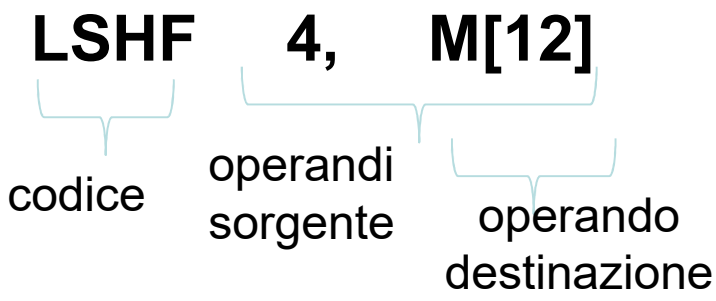



- **SUB M[14], $R_2$, $R_2$**

  accedi alla memoria all'indirizzo 14, sottrai a questo valore il contenuto di $R_2$ e scrivi il risultato in $R_2$

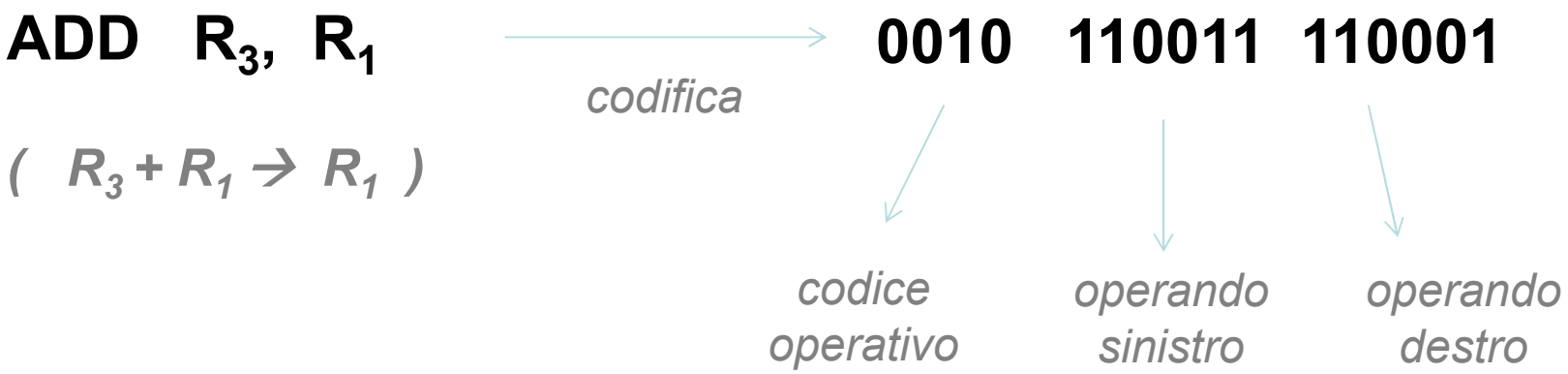

- **LSHF 4, M[12]**

  transla il contenuto della memoria all'indirizzo 12 di quattro posizioni a sinistra



# Codifica delle istruzioni

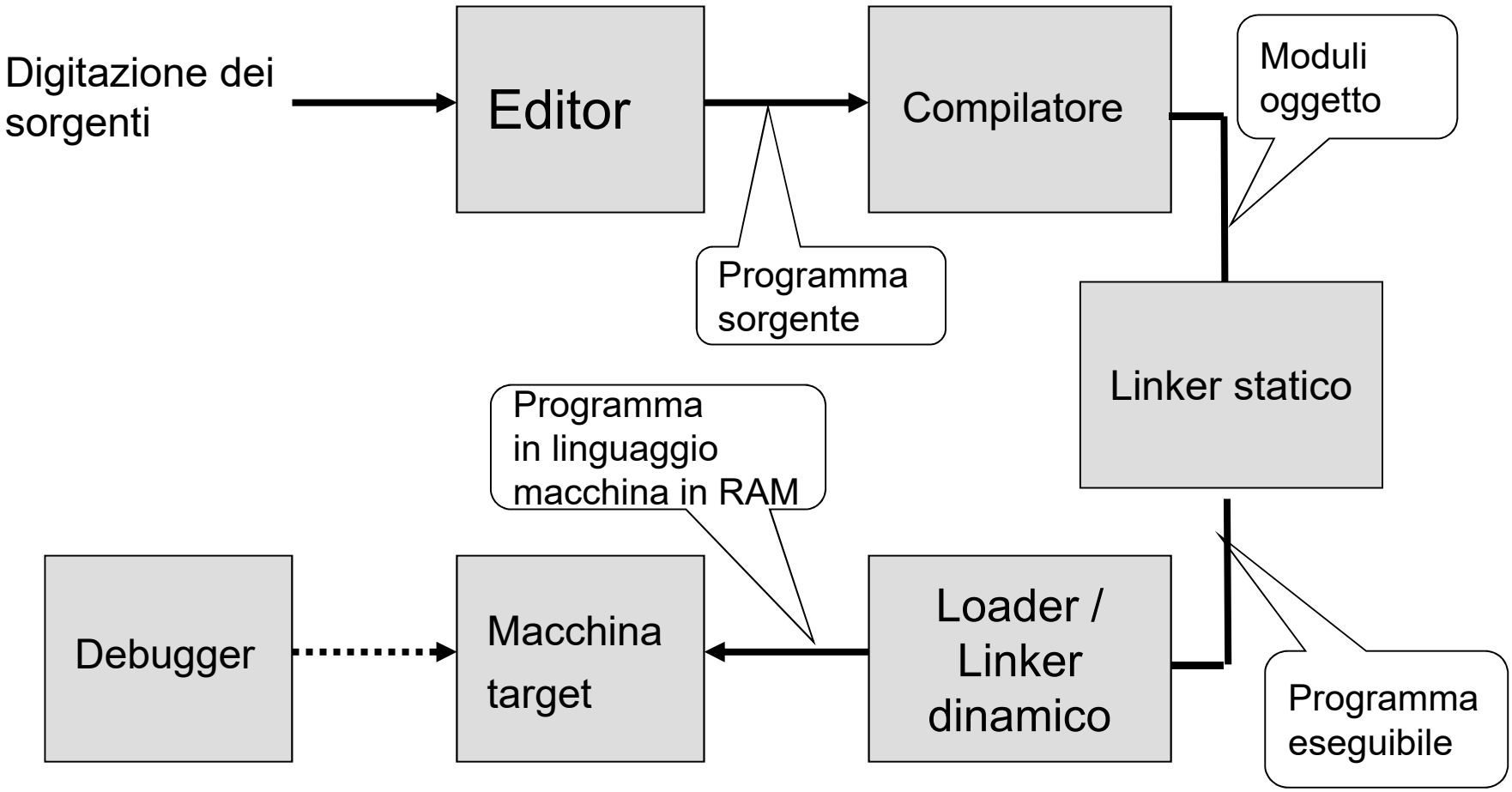
- Esempio



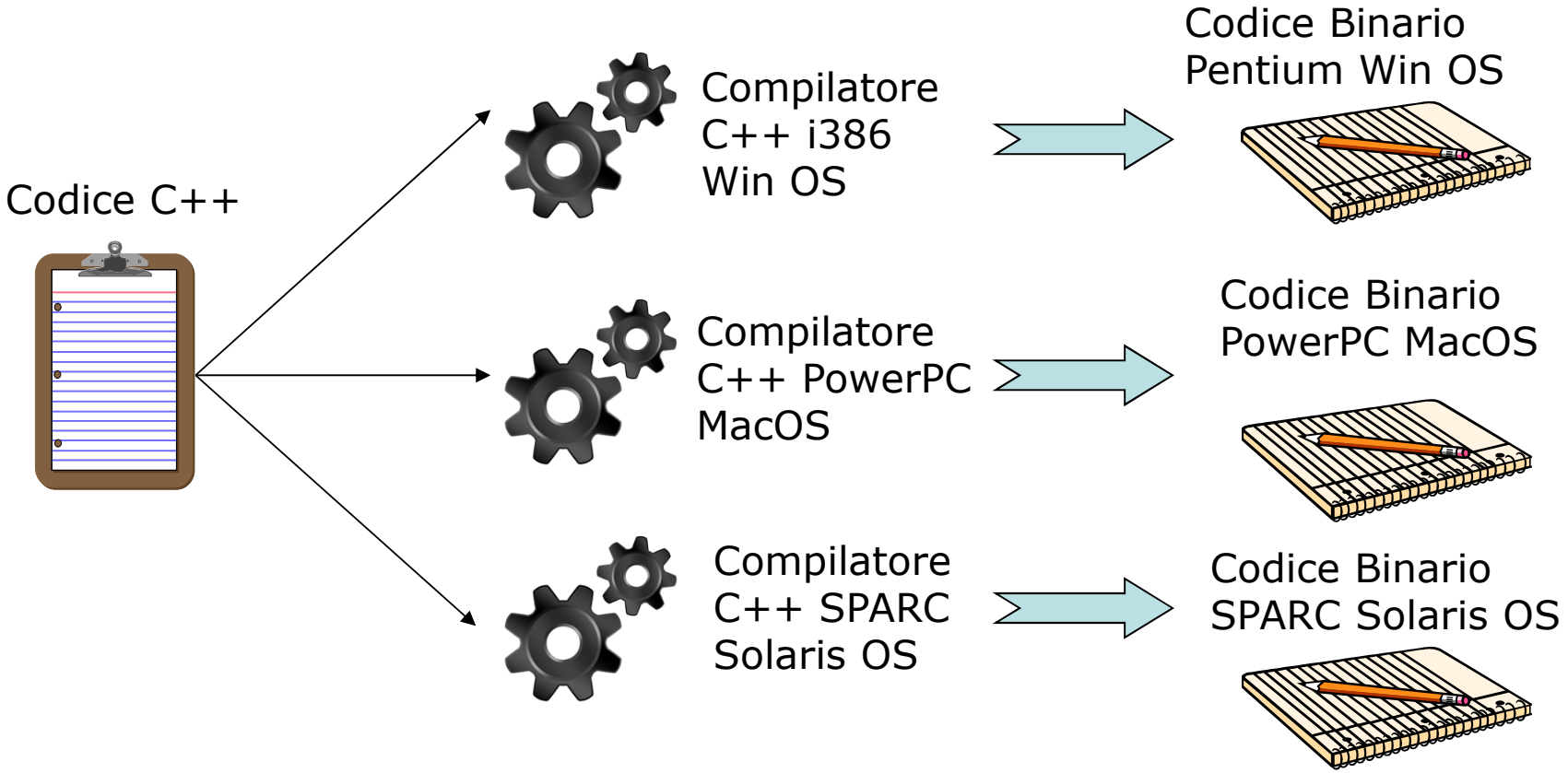
# Codifica delle istruzioni

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

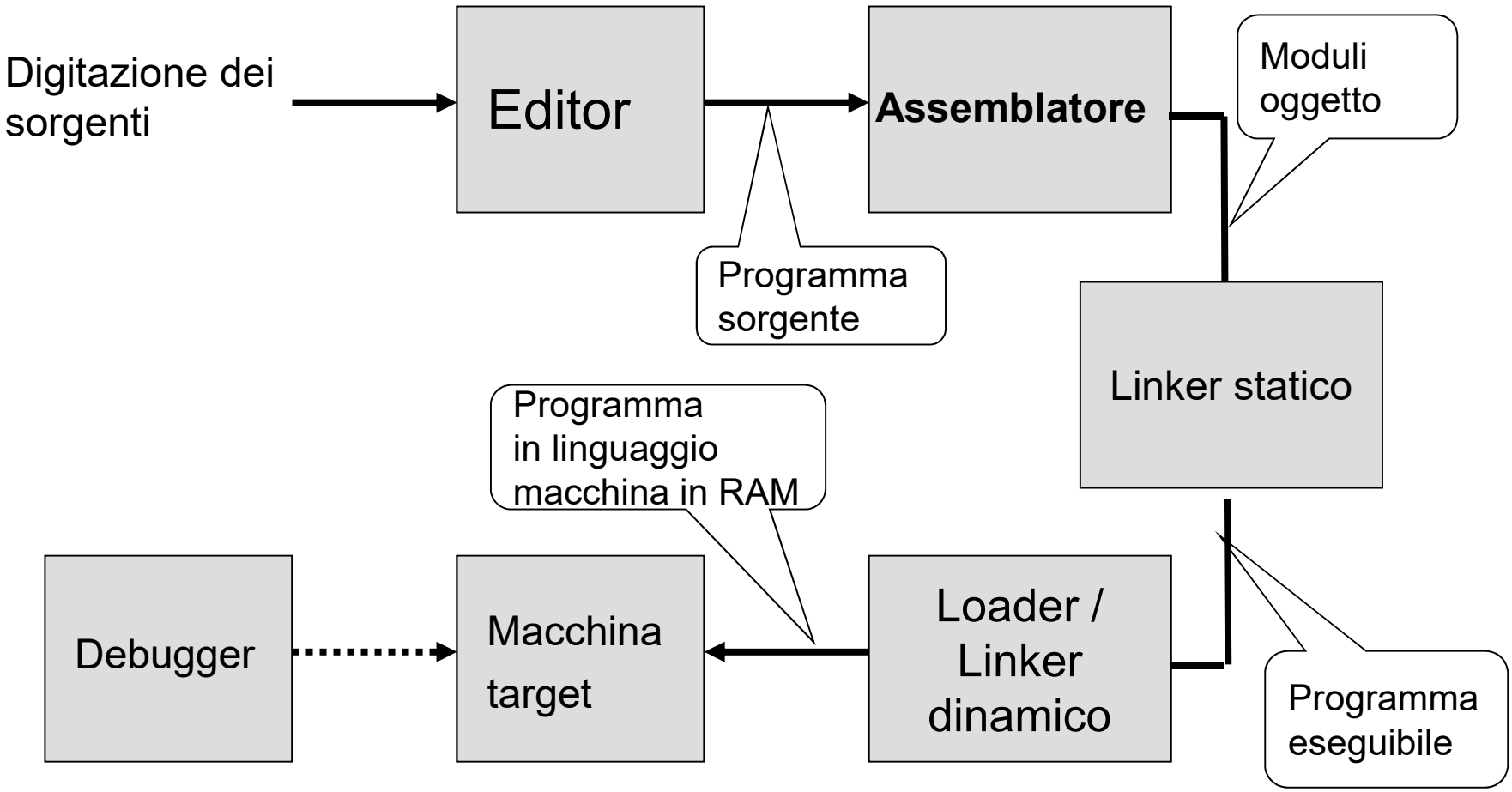
# Ciclo di sviluppo/esecuzione per programmi in linguaggio di alto livello



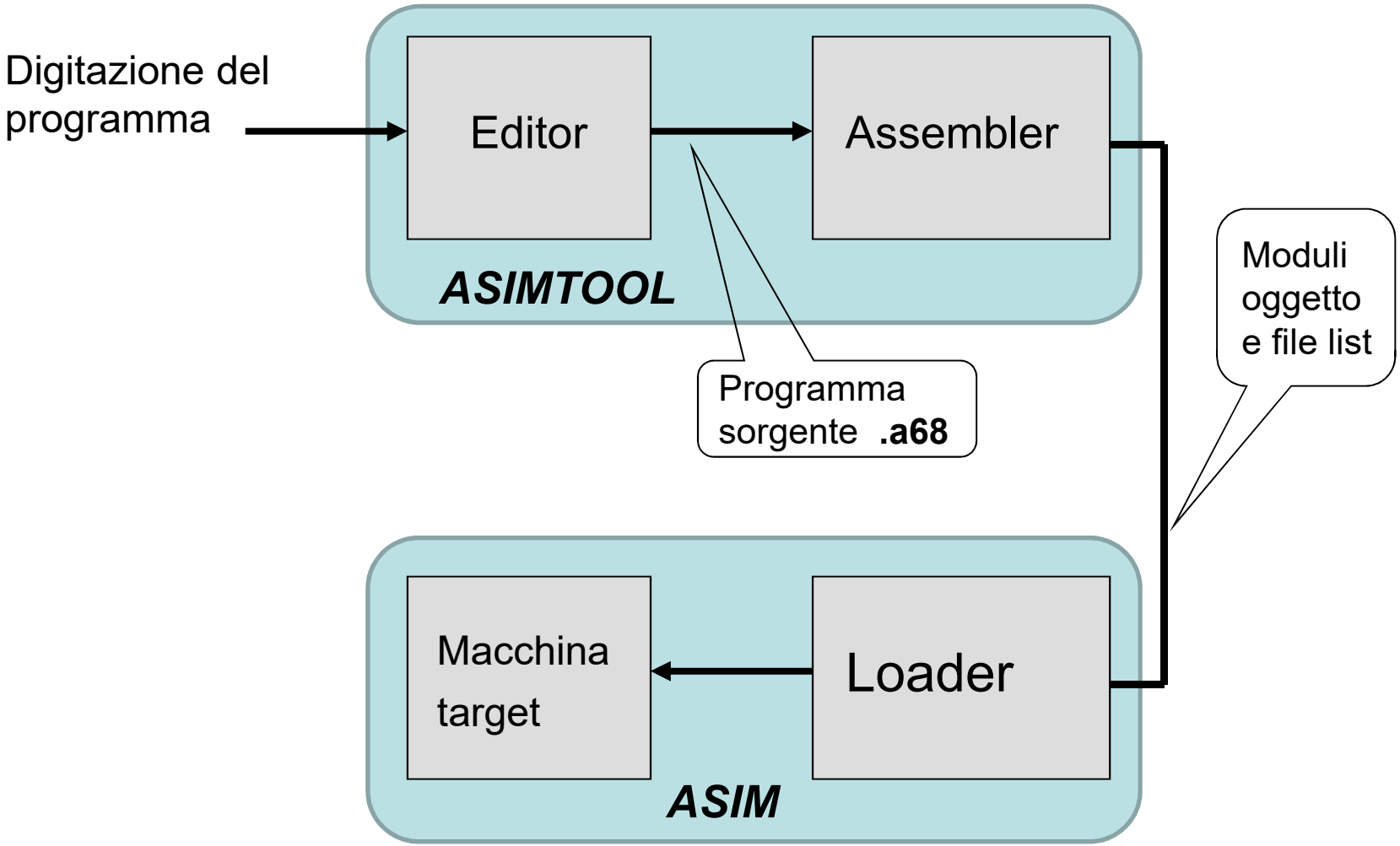
# Linguaggi e dipendenza dalla piattaforma di esecuzione



# 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

# 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

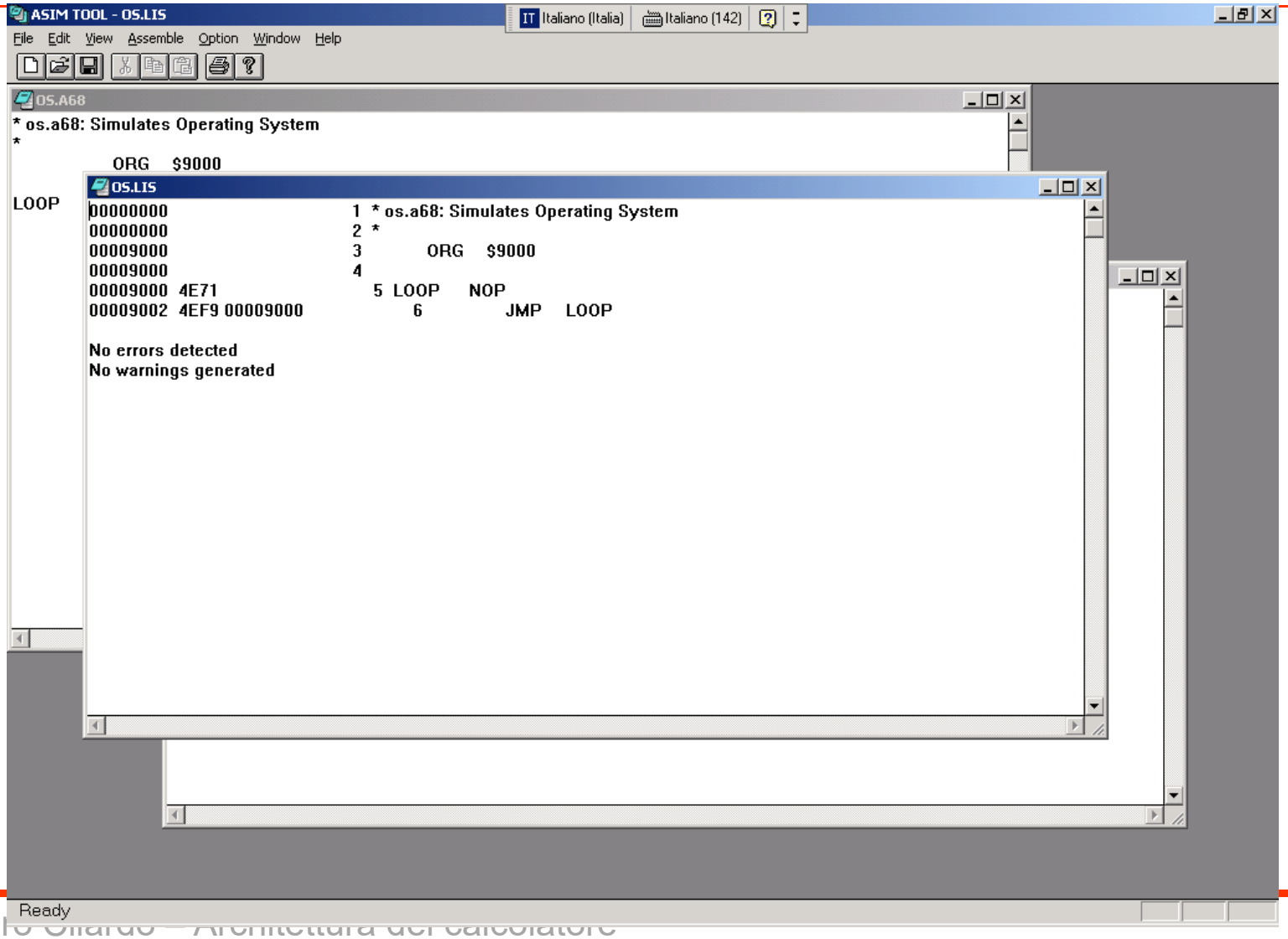
- E' una variabile interna dell'assemblatore
- Punta alla locazione di memoria in cui andrà caricata – a run time – l'istruzione assemblata
- Viene inizializzato dallo pseudo-operatore "origin" (ORG)
- Durante il processo di assemblaggio, il suo valore è aggiornato sia in funzione degli operatori, sia 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


ASIM TOOL - OS.LIS
Italiano (Italia)
Italiano (142)
File Edit View Assemble Option Window Help
OS.A68
* os.a68: Simulates Operating System
*
ORG $9000
LOOP
OS.LIS
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
Ready

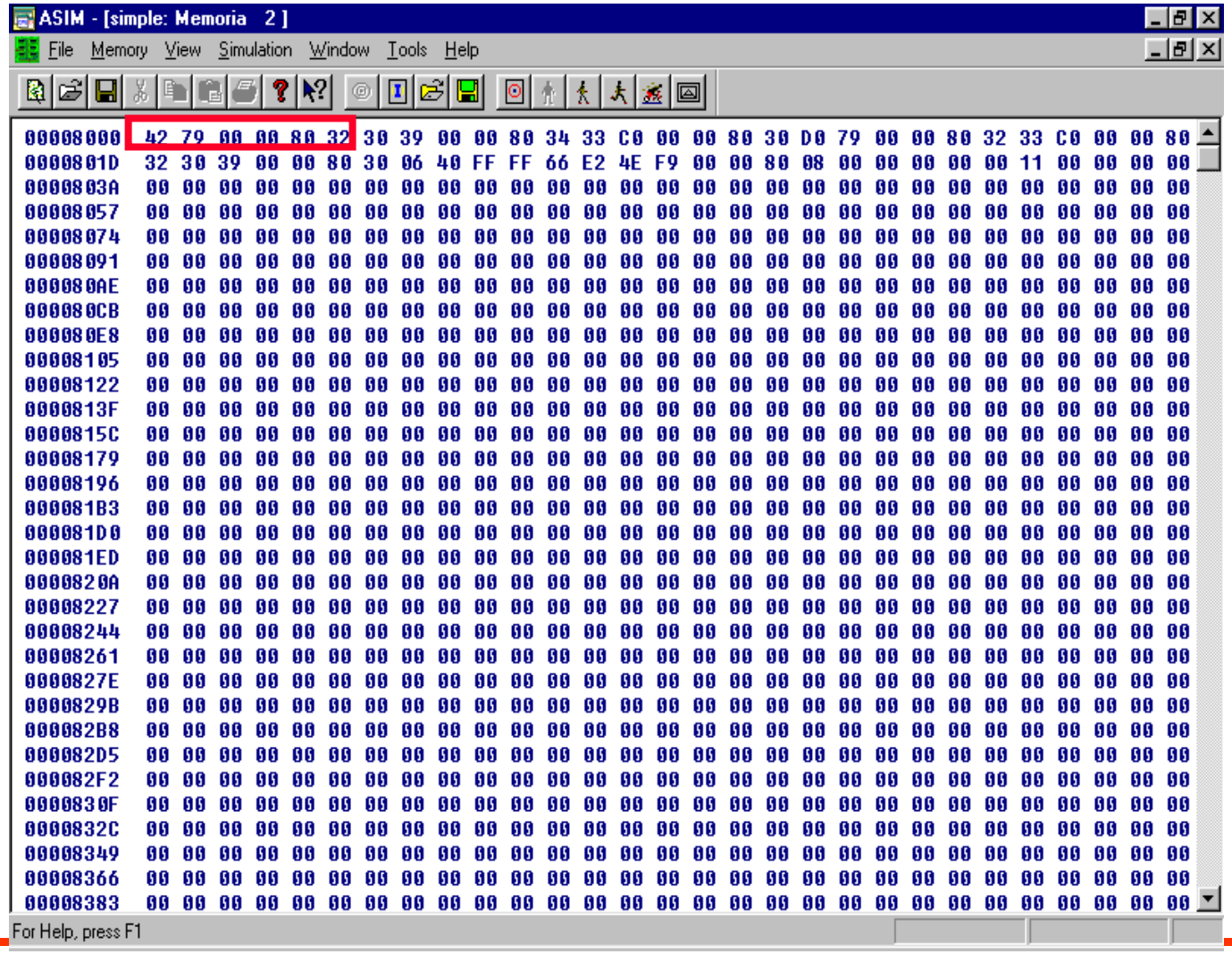
# AsimTool: esempio di file list

```
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



# 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
   … … … … … …
```