

# Corso di Calcolatori Elettronici I

---

## Primi programmi assembly MC68000

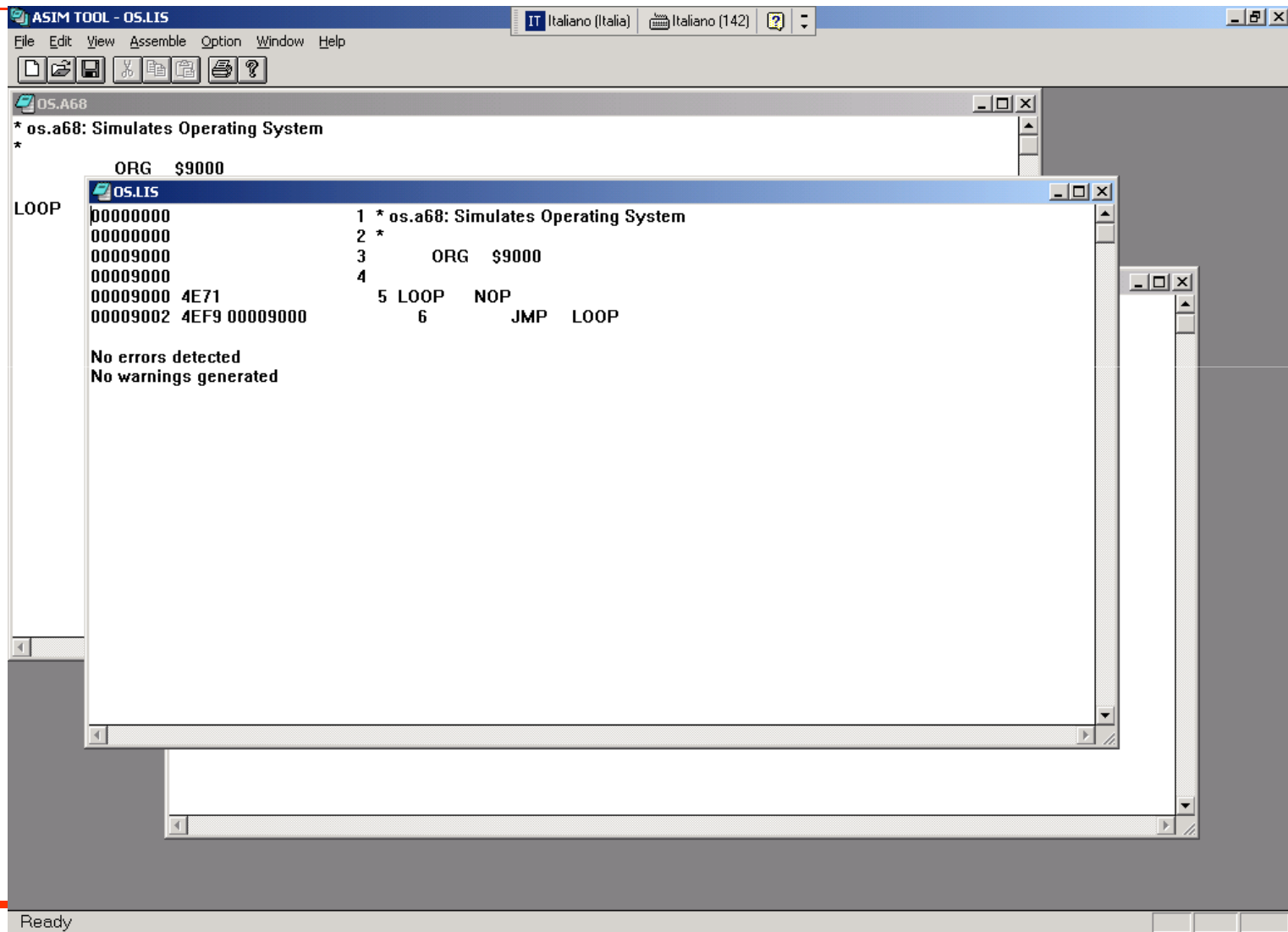
**Prof. Roberto Canonico**



Università degli Studi di Napoli Federico II  
Dipartimento di Ingegneria Elettrica  
e delle Tecnologie dell'Informazione  
Corso di Laurea in Ingegneria Informatica  
Corso di Laurea in Ingegneria dell'Automazione

---

# AsimTool



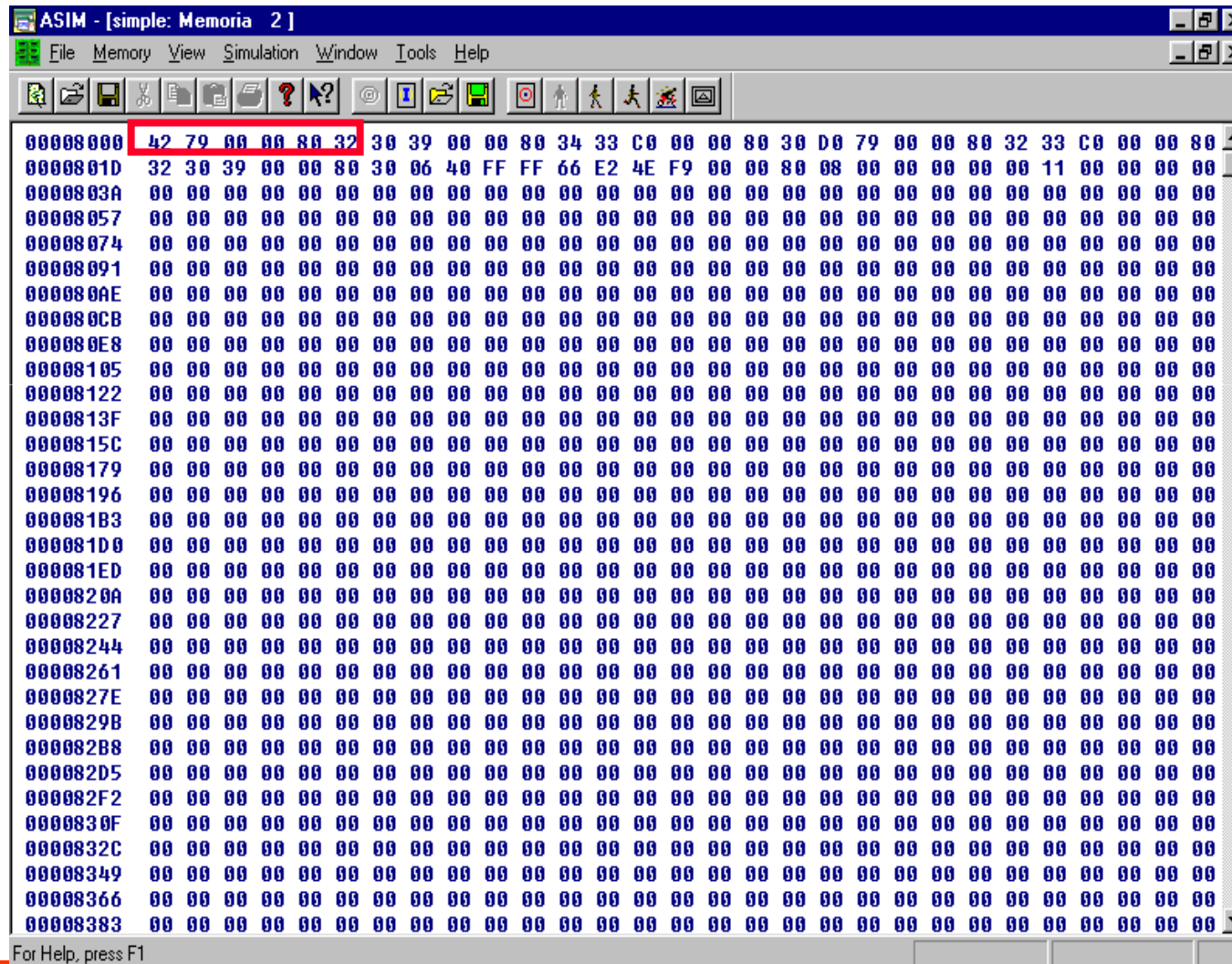
# AsimTool: esempio di file list

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 11 00 00 00 00	
0000803A	00 00	
00008057	00 00	
00008074	00 00	
00008091	00 00	
000080AE	00 00	
000080CB	00 00	
000080E8	00 00	
00008105	00 00	
00008122	00 00	
0000813F	00 00	
0000815C	00 00	
00008179	00 00	
00008196	00 00	
000081B3	00 00	
000081D0	00 00	
000081ED	00 00	
0000820A	00 00	
00008227	00 00	
00008244	00 00	
00008261	00 00	
0000827E	00 00	
0000829B	00 00	
000082B8	00 00	
000082D5	00 00	
000082F2	00 00	
0000830F	00 00	
0000832C	00 00	
00008349	00 00	
00008366	00 00	
00008383	00 00	

For Help, press F1

# Esempio #1: somma n interi

---

---

- Scrivere un programma che sommi i primi n interi
    - $n = 17$
  - Assemblare il programma con ASIMTOOL ed eseguirlo sul simulatore ASIM
  - 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
-

# Esempio #1 - sumnnums.a68

---

---

```
                ORG      $8000
START          CLR.W    D0
              MOVE.W   N, D1
LOOP          ADD.W    D1, D0
              ADD.W    #-1, D1
              BNE     LOOP
              MOVE.W   D0, SUM
LAST          JMP     LAST
              ORG     $8200
N             DC.W    17
SUM          DS.W    1
              END     START
```

---

## Esempio #2 - 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
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	

---

## Esempio #3: prodotto scalare

---

- Scrivere un programma che esegua il prodotto scalare tra due vettori di interi word A e B
    - A e B allocati staticamente ed inizializzati con DC
  - Assemblare il programma con ASIMTOOL ed eseguirlo sul simulatore ASIM
  - La dimensione N è costante
-



# DBcc: Test condition, decrement, and branch

---

**Operazione:** IF (cc false) THEN  
                  [*Dn*] ← [*Dn*] - 1  
                  IF [*Dn*] = -1 THEN [*PC*] ← [*PC*] + 2  
                                      ELSE [*PC*] ← [*PC*] + *d*  
                  ELSE [*PC*] ← [*PC*] + 2

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

**Attributi:** Size = word

## Descrizione:

Fintantoché la condizione *cc* rimane falsa, decrementa il registro *Dn*, e se questo non era zero prima del decremento (ovvero se non vale -1) salta all'istruzione a distanza *d*. Negli altri casi, passa all'istruzione seguente.

Fornisce un modo sintetico per gestire i cicli, sostituendo con un'unica istruzione il decremento di un registro di conteggio e la verifica di una condizione normalmente fatti con istruzioni separate.

Supporta tutti i cc usati in Bcc. Inoltre, ammette anche le forme DBF e DBT (F = false, e T = true) per ignorare la condizione ed usare solo il registro di conteggio.

---

# Esempio #3 – scalprod.a68

---

```

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

---

# Esempio #4 - Ricerca di un token in una stringa di caratteri

---

- Scrivere un programma che:
    - Riconosca un token (un carattere speciale noto) in una stringa di caratteri
    - La lunghezza della stringa non sia nota
    - La fine della stringa segnalata dal byte zero (come in C/C++)
    - Memorizzi l'indirizzo della prima istanza del token in una locazione di memoria TOKENA
  - Assemblare ed eseguire il programma sul simulatore
-

# Esempio #4 – token.a68

---

```
ORG      $8000
START    MOVEA.L #STRING, A0
         MOVE.B  #TOKEN, D0
LOOP     TST.B   (A0)
         BEQ    DONE
         CMP.B  (A0)+, D0
         BNE   LOOP
FOUND    SUBQ.L  #1, A0
DONE     MOVE.L  A0, TOKENA
LAST     JMP    LAST
         ORG    $8100
STRING   DC.B   'QUI QUO:QUA', 0
TOKEN    EQU    ':'
TOKENA   DS.L   1
         END    START
```

---