

# Architettura degli Elaboratori Elettronici Esercitazioni

*MARS*

Franco Liberati  
[liberati@di.uniroma1.it](mailto:liberati@di.uniroma1.it)



# MARS

*MIPS Assembler and Runtime Simulator*





# MARS

## *Introduzione*

- ❑ **MARS** è un ambiente di sviluppo interattivo (IDE) leggero per la programmazione in linguaggio assembly MIPS, destinato all'uso a livello didattico
- ❑ MARS è un **simulatore** che esegue programmi per le architetture MIPS - R2000/R3000
- ❑ MARS può **leggere ed assemblare programmi scritti in linguaggio assembly MIPS**
- ❑ MARS contiene un **debugger** per poter analizzare il funzionamento dei programmi



# MARS

*Download ed installazione*

❑ Il **download di MARS** si esegue al collegamento

<https://computerscience.missouristate.edu/mars-mips-simulator.htm>

**MARS: Mips Assembler and Runtime Simulator**



Version 3.8 Copyright (c) 2003-2010  
Pete Sanderson and Kenneth Vollmar



# MARS

*Esecuzione*

- ☐ Salvare il file Mars.jar sul desktop. Eseguire MARS facendo doppio clic sull'icona
- ☐ Aprire il Shell DOS (Terminale o CMD). Eseguire MARS con il comando DOS  
**java -jar Mars.jar**
- ☐ Shell DOS che utilizza classi Java. Estrarre il file MARS con il comando DOS **jar -xf Mars.jar**. Eseguire MARS con il comando DOS  
**java Mars**



BARRA COMANDI

# MARS

*Interfaccia grafica*

REGISTRI

AREA PER LA SCRITTURA DEL FILE ASSEMBLATIVO

TERMINALE DI USCITA (MONITOR)

Registers			
Coproc 1		Coproc 0	
Name	Number	Value	
\$zero	0	0x00000000	
\$at	1	0x00000000	
\$v0	2	0x00000000	
\$v1	3	0x00000000	
\$a0	4	0x00000000	
\$a1	5	0x00000000	
\$a2	6	0x00000000	
\$a3	7	0x00000000	
\$t0	8	0x00000000	
\$t1	9	0x00000000	
\$t2	10	0x00000000	
\$t3	11	0x00000000	
\$t4	12	0x00000000	
\$t5	13	0x00000000	
\$t6	14	0x00000000	
\$t7	15	0x00000000	
\$s0	16	0x00000000	
\$s1	17	0x00000000	
\$s2	18	0x00000000	
\$s3	19	0x00000000	
\$s4	20	0x00000000	
\$s5	21	0x00000000	
\$s6	22	0x00000000	
\$s7	23	0x00000000	
\$t8	24	0x00000000	
\$t9	25	0x00000000	
\$k0	26	0x00000000	
\$k1	27	0x00000000	
\$gp	28	0x10000000	
\$sp	29	0x7fffffc0	
\$fp	30	0x00000000	
\$ra	31	0x00000000	
pc		0x00400000	
hi		0x00000000	
lo		0x00000000	



# MARS

## *Barra dei comandi – Principali comandi*

☐ Tra i comandi principali ci sono:

### **File ⇒ New**

Permette di editare un file in linguaggio assembly e ne consente il salvataggio (estensione .s, .asm)

### **File ⇒ Open**

Carica un file scritto in assembly (estensione .s, .asm) e ne assembla il contenuto in memoria

### **Comando RUN ⇒ Assemble**

Converte il file scritto in assembler in eseguibile e lo carica nei rispettivi segmenti della memoria centrale

### **Comando RUN ⇒ GO**

Esegue il programma

### **Comando RUN ⇒ STEP**

*Esegue una istruzione alla volta. Questa modalità permette di studiare, nel dettaglio, il funzionamento del programma*





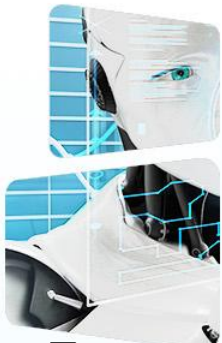
# MARS

## Registri

- ❑ Il **pannello registri** mostra il valore di tutti i registri della CPU e della FPU MIPS (Coproc1)
- ❑ I registri generali sono identificati sia dal numero progressivo sia dal mnemonico (es.: \$8/\$t0)
- ❑ I registri nei quali sono presenti gli operandi elaborati dal coprocessore matematico (o Unità Floating Point) sono riportati nella tabella denominata come COPROC 1

Registers	Coproc 1	Coproc 0
Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7fffffc0
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000





# MARS

## Area Codice

- ❑ L'Area Codice mostra le istruzioni in linguaggio macchina

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x24080005	addiu \$8,\$0,5	1: li \$t0,5
<input type="checkbox"/>	0x00400004	0x24090005	addiu \$9,\$0,5	2: li \$t1,5
<input type="checkbox"/>	0x00400008	0x01095020	add \$10,\$8,\$9	3: add \$t2,\$t0,\$t1

- ❑ Descrizione degli elementi

1. **Address:** indirizzo di memoria (in esadecimale) dove è memorizzata l'istruzione
2. **Code:** codifica numerica (in esadecimale) dell'istruzione in linguaggio macchina
3. **Basic:** Istruzione presente nel file assembly

*Nota: Ad alcune istruzioni assembly possono corrispondere più istruzioni in linguaggio macchina (pseudoistruzioni)*

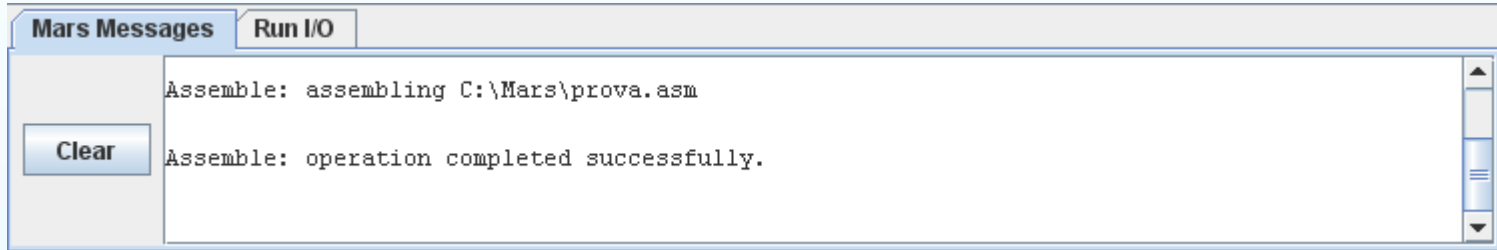
4. **Source:** Descrizione mnemonica dell'istruzione



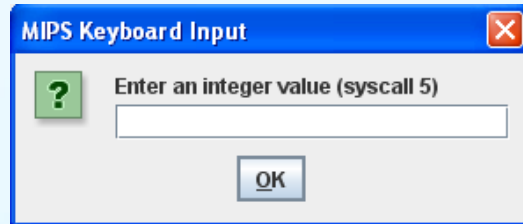
# MARS

## Messaggi e console

- ❑ La **console** è la sessione utilizzata da MARS per mostrare informazioni all'utente, come ad esempio messaggi di errore o di corretto caricamento del file ed esecuzione



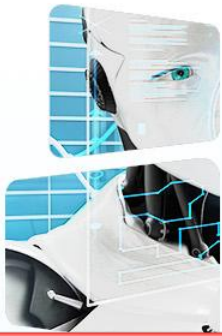
- ❑ Esiste anche una console che consente l'interazione con l'assemblatore (inserimento/lettura dati)



# MIPS

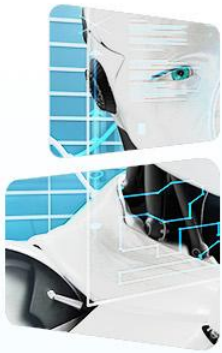
## Un primo programma





# Programma 1

Implementare la somma di due numeri definiti in memoria e riportare il risultato nel registro \$t0



# Programma 1

```
.text  
.globl main
```

```
main:
```

```
lw $t1,pippo  
lw $t2,paperino  
add $t0,$t1,$t2
```

```
#Lettura del primo operando dalla memoria  
#Lettura del secondo operando dalla memoria  
#Somma dei due operandi
```

```
li $v0,10  
syscall
```

```
.data  
pippo:    .word 4  
paperino: .word 6
```

Fine

