

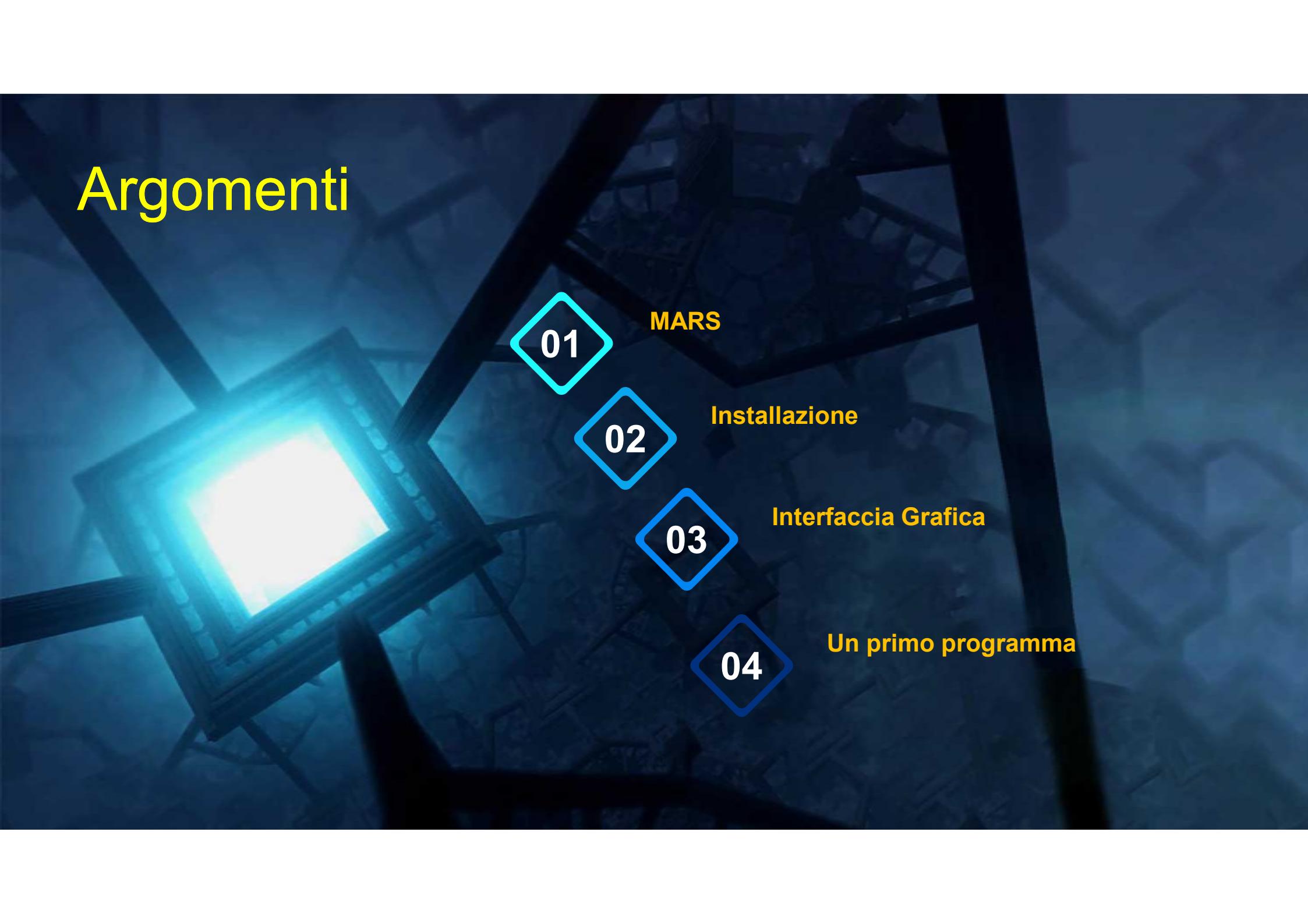


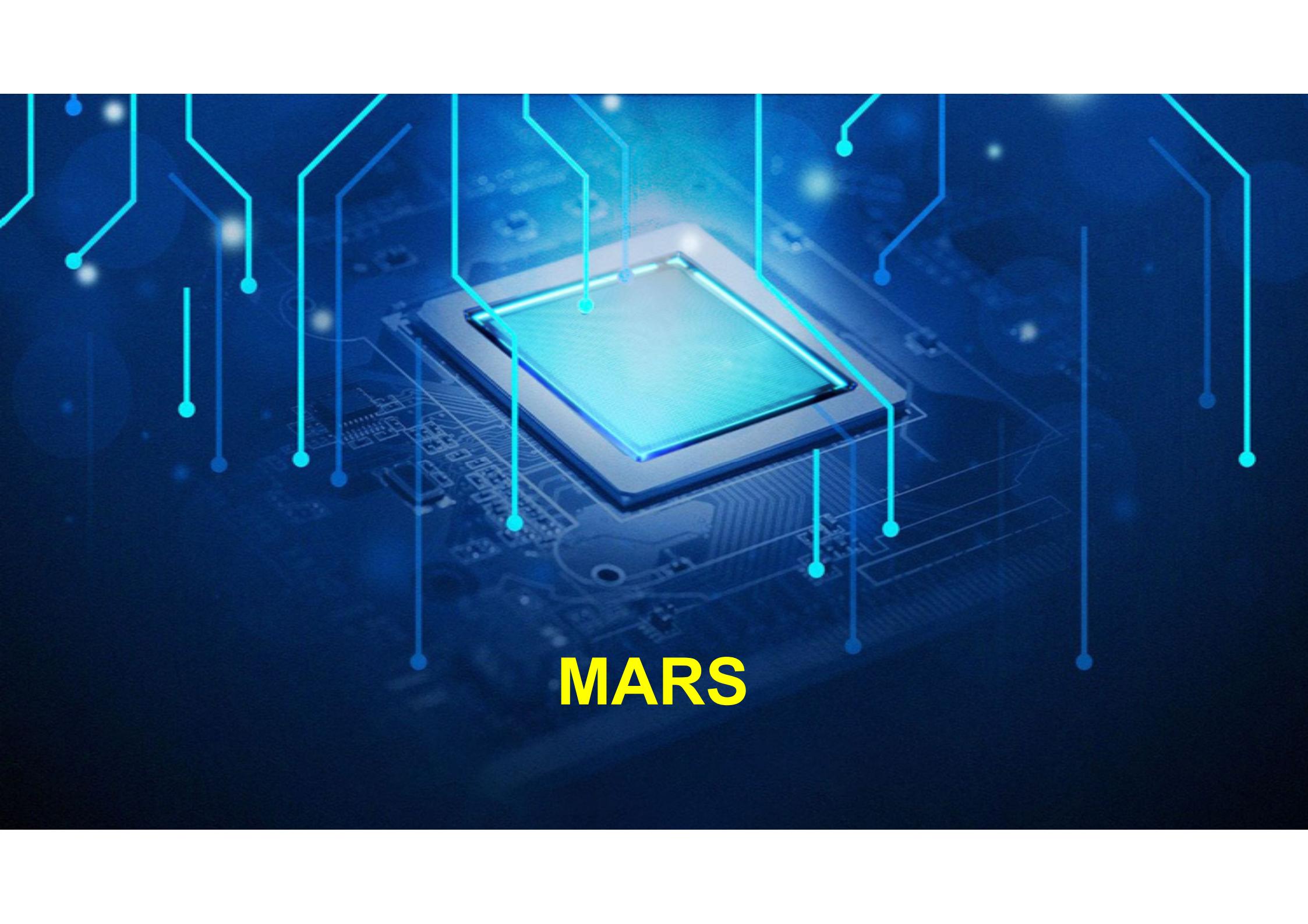
SAPIENZA  
UNIVERSITÀ DI ROMA

MARS

Dott. Franco Liberati

# Argomenti

- 
- 01 MARS
  - 02 Installazione
  - 03 Interfaccia Grafica
  - 04 Un primo programma



**MARS**

# Simulatore MIPS: MARS

I programmi in linguaggio assemblativo MIPS possono essere redatti ed eseguiti con diversi simulatori. In questo corso si utilizza il *MIPS Assembler and Runtime Simulator* (MARS) versione 4.5 rilasciato nell'Agosto del 2014 e reperibile presso il sito del Missouri State University all'indirizzo:

**<https://courses.missouristate.edu/KenVollmar/mars/download.htm>**





**Installazione**

# Installazione

L'installazione del programma MARS richiede che l'elaboratore elettronico sia dotato della piattaforma SDK Java versione 1.5.x (o successive) e debba avere un estrattore di documenti digitali in formato zip o jar.

In base al Sistema Operativo si esegue il programma in modi diversi:

## MARS installazione Windows

- Download del file MARS.jar
- Installazione di Java SE2 (versione 1.6 in poi)
- Se si usa l'interfaccia grafica **doppio click sull'icona Mars4\_5**
- Se si usa il file jar eseguire il comando da shell DOS:  
**java -jar Mars.jar**
- Se si usano le classi java eseguire il comando da shell DOS:  
**jar -xf Mars.jar**

## MARS Installazione Unix

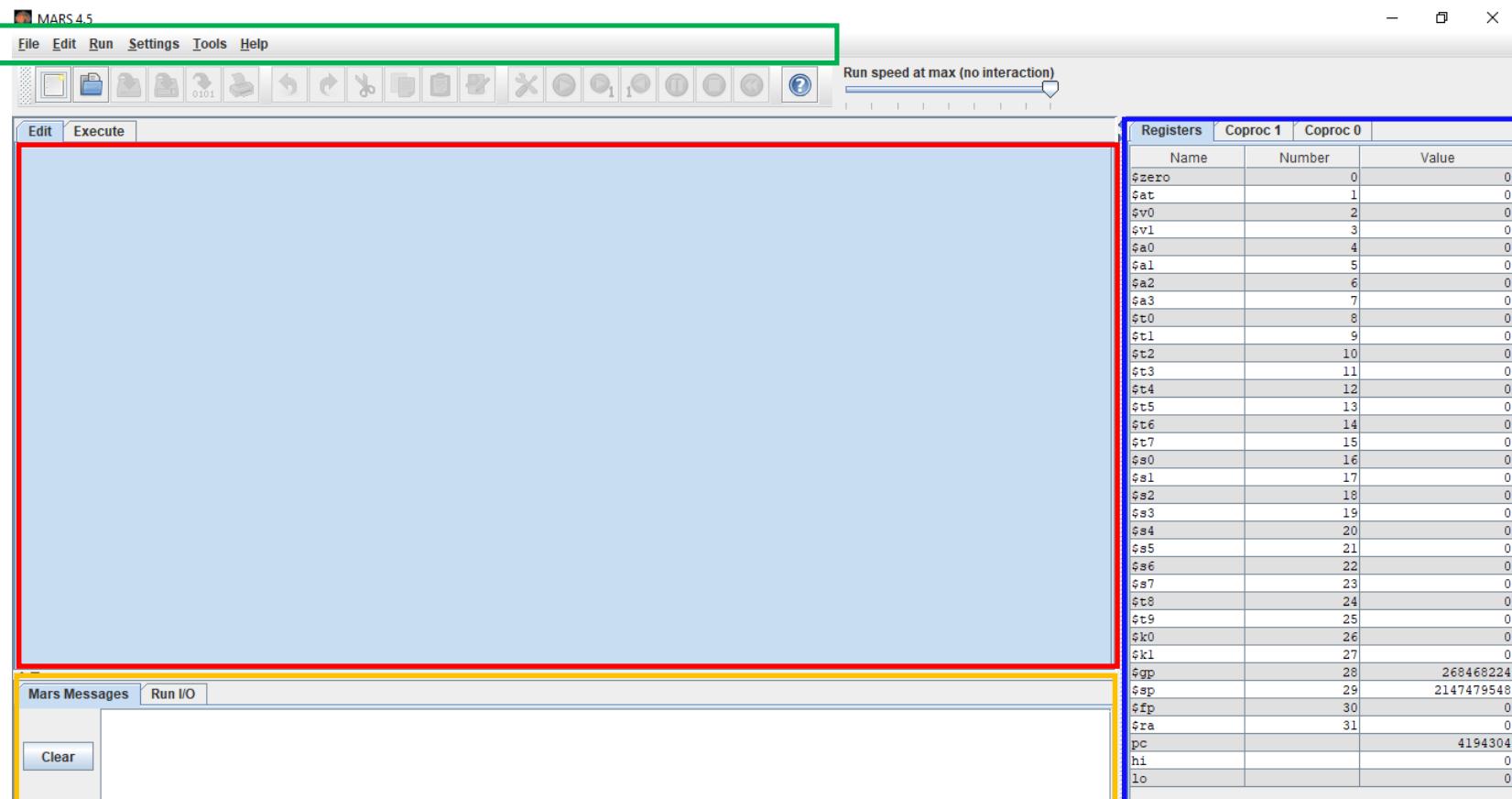
- Download del file MARS.jar
- Installazione di Java SE2 (versione 1.6 in poi)
- Se si usa il file jar eseguire il comando da shell:  
**java -jar Mars.jar**
- Se si usano le classi java eseguire il comando da shell:  
**jar -xf Mars.jar**

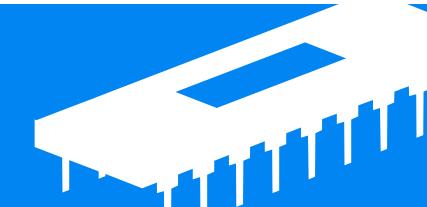
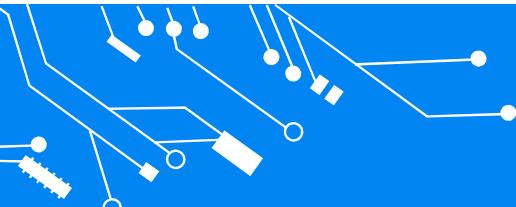


**Interfaccia grafica**

# Interfaccia grafica

Il simulatore MARS si presenta - dopo una *splash page* introduttiva che riporta i crediti e la versione - con una **barra dei comandi**; un **editor di testo**, utile per la stesura del programma in linguaggio assemblativo; i **registri dell'Unità di Elaborazione**; e, infine, delle **finestre di ausilio per il programmatore**





# Barra dei Comandi

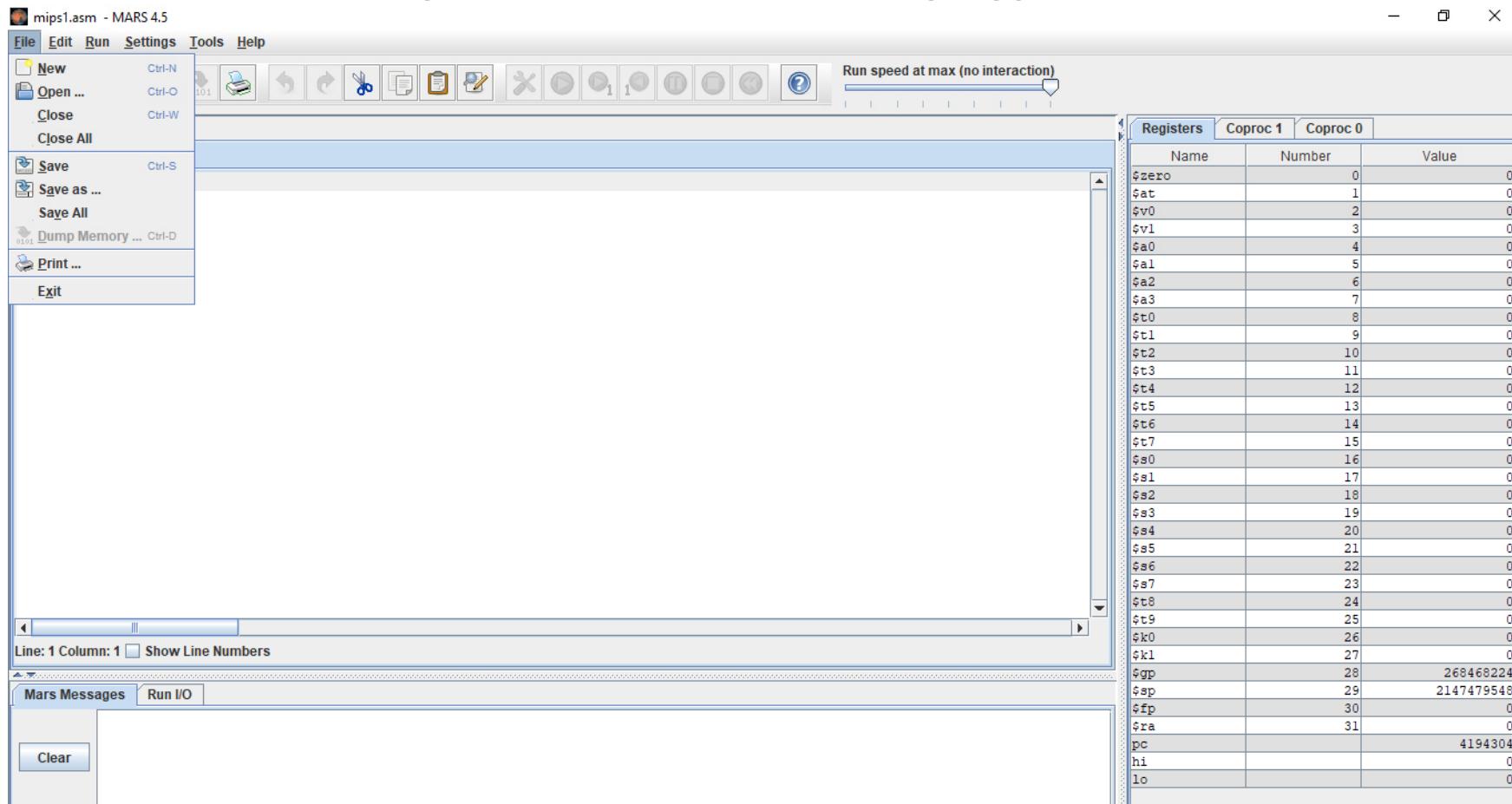
Nella parte in alto del programma è presente la **barra dei comandi** che consente l'attivazione di sei sottomenu che offrono, tra l'altro, le funzioni per la stesura del programma, l'assemblaggio, l'esecuzione e l'analisi di quanto in corso di esecuzione.



# Barra dei Comandi: FILE

## FILE

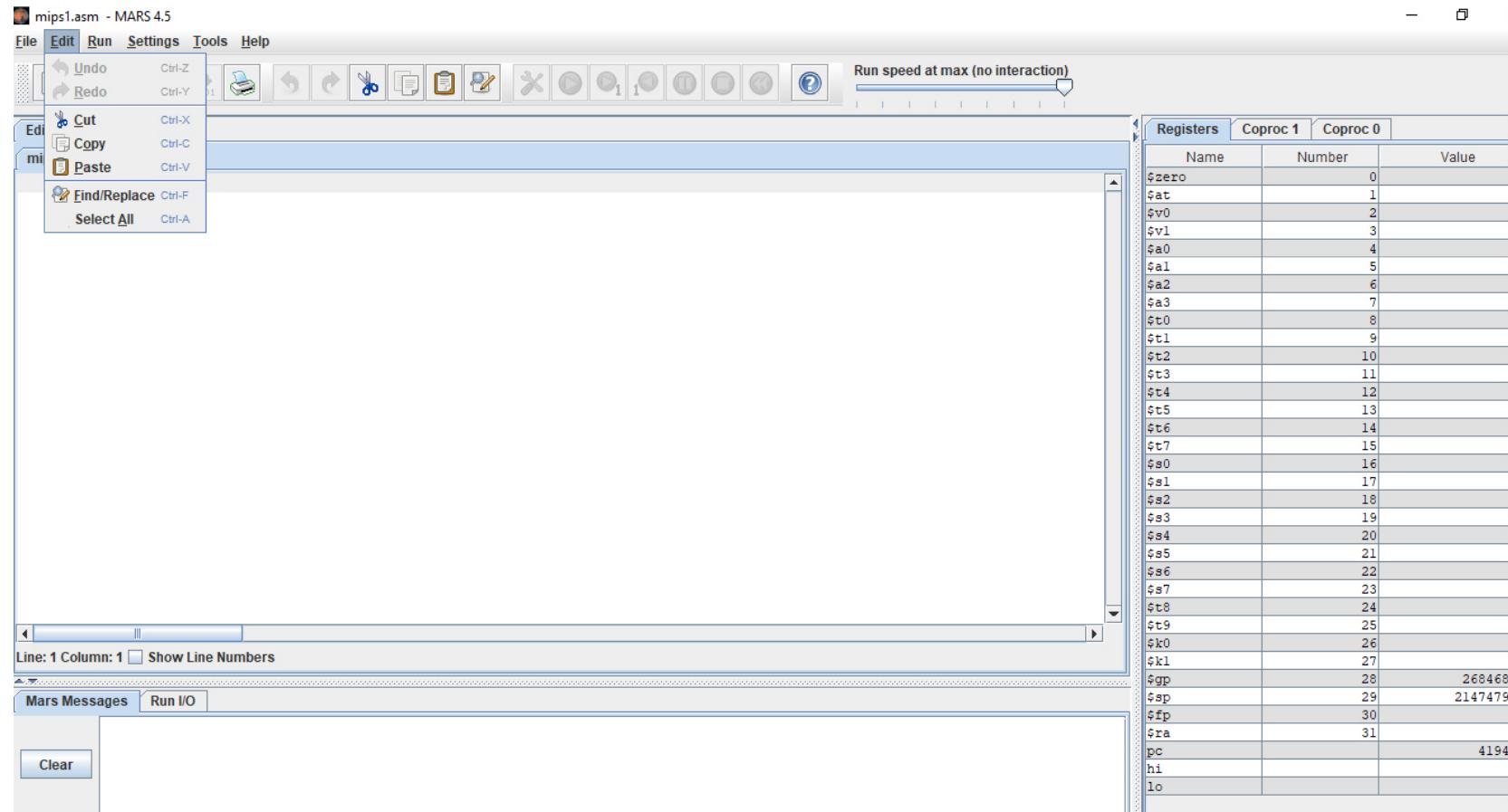
La prima sezione permette la gestione dei documenti in linguaggio assemblativo



# Barra dei Comandi: EDIT

## EDIT

Le scelte del secondo sottomenu vertono sulla **gestione testuale** di quanto scritto nel documento



# Barra dei Comandi: RUN

## RUN

Questa parte consente l'assemblaggio e l'esecuzione del programma editato

The screenshot shows the MARS 4.5 assembly editor interface. The main window displays assembly code for a MIPS program named 'mips1.asm'. The code defines a global variable 'main' containing a simple addition routine, and a data segment with three variables: 'valore1', 'valore2', and 'risultato'. The right side of the interface features a 'Registers' window showing the initial values of all 32 general-purpose registers (\$zero to \$lo) set to 0. The top menu bar has 'Run' selected, indicating the current mode of operation.

```
mips1.asm
#####
#Descrizione: somma di due numeri
#Informazioni: la somma di due numeri in memoria
#Data: 2021-01

.text
.globl main
main:
    lw $t0, valore1
    lw $t1, valore2
    add $t2,$t0,$t1
    sw $t2,risultato
    li $v0, 10
    syscall

.data
valore1 : .word 23
valore2 : .word 12
risultato: .word 0

#####
#Direttiva del Segment text (Segmento Testo)
#Direttiva che indica la subroutine con etichetta main come globale
#Etichetta main: inizio del programma
#Prelievo del primo operando dalla locazione di memoria con etichetta valore1 nel registro $t0
#Prelievo del secondo operando dalla locazione di memoria con etichetta valore2 nel registro $t1
#Somma degli operandi e scrittura del risultato nel registro $t2
#Salvataggio dell'addizione nella locazione di memoria risultato
#Richiesta del servizio di interruzione del programma
#Attivazione del servizio
#Linea vuota non richiesta, ma lasciata per una maggiore leggibilità
#Direttiva del Segment Data (Segmento Dati)
#Definizione della variabile valore1 (intero a 32bit inizializzato a 23)
#Definizione della variabile valore2 (intero a 32bit inizializzato a 12)
#Definizione della variabile risultato (intero a 32bi inizializzato a 0, pulizia)
```

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194304
hi		0
lo		0

# Barra dei Comandi: SETTING

## SETTING

In questa sezione sono presenti le funzionalità utili per impostare il simulatore e ottimizzare il programma

The screenshot shows the MARS 4.5 assembly editor interface. The title bar reads "F:\Arch2\_2019\MARS\Cod\_3\_A.asm - MARS 4.5". The menu bar includes File, Edit, Run, **Settings**, Tools, and Help. The main window has tabs for "Edit" and "Execute". The "Edit" tab displays an assembly code listing:

```
#Descrizione: #Informazioni
#
#Data
#2021-01-01
#####
.text
.globl main
main:
    lw $t0, 10
    lw $t1, 12
    add $t0, $t1
    sw $t0, 20
    li $v0, 10
    syscall
    .data
    valore1 : .word 23
    valore2 : .word 12
    risultato: .word 0
```

The "Settings" tab is selected and shows various configuration options:

- Show Labels Window (symbol table) (checked)
- Program arguments provided to MIPS program
- Popup dialog for input syscalls (5,6,7,8,12) (checked)
- Addresses displayed in hexadecimal
- Values displayed in hexadecimal
- Assemble file upon opening
- Assemble all files in directory
- Assembler warnings are considered errors
- Initialize Program Counter to global 'main' if defined (checked)
- Permit extended (pseudo) instructions and formats (checked)
- Delayed branching
- Self-modifying code

Below the settings are buttons for Editor..., Highlighting..., Exception Handler..., and Memory Configuration... along with a scrollable text area containing assembly comments.

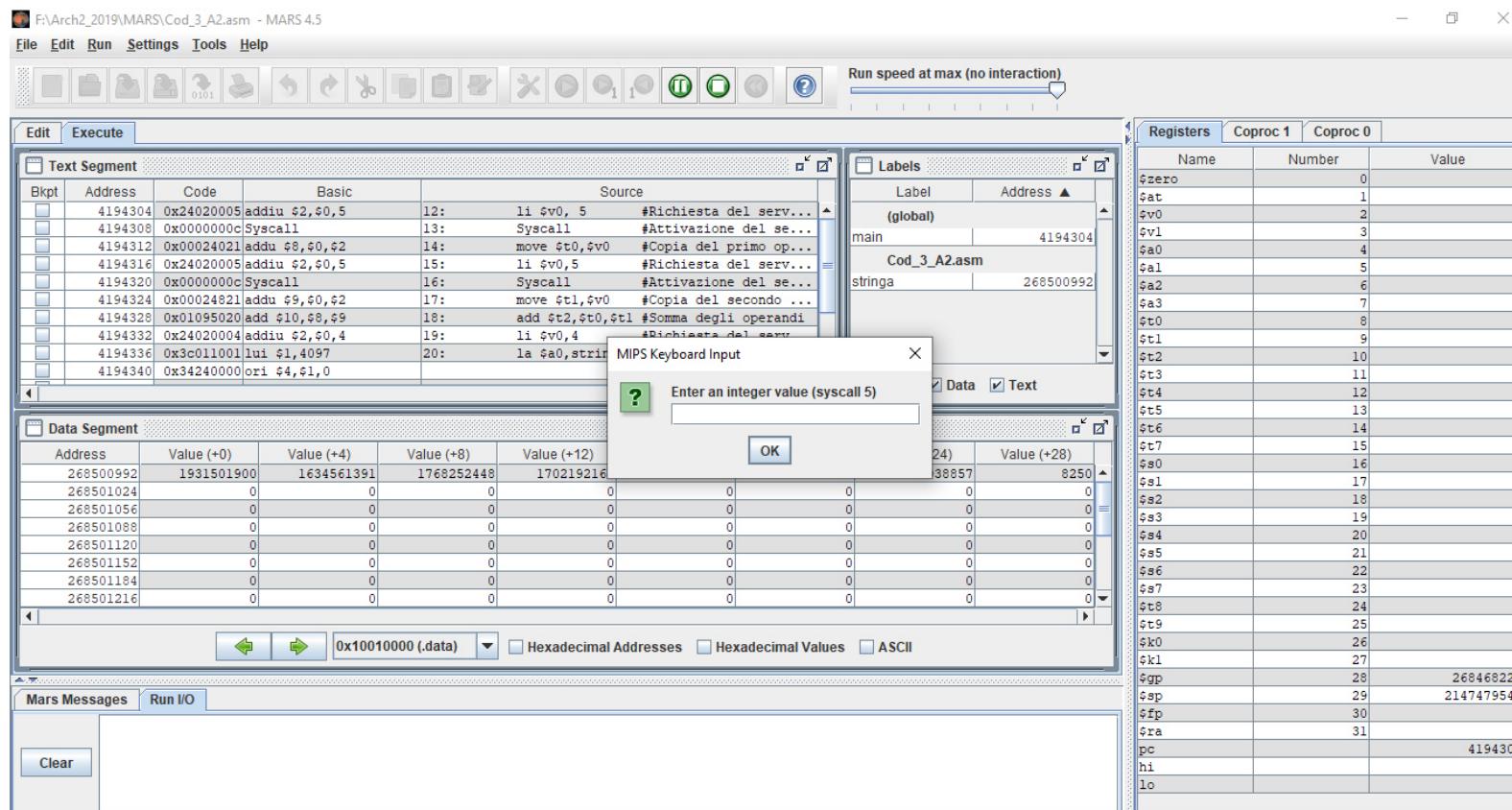
The right side of the interface features a "Registers" window showing the following table:

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194304
hi		0
lo		0

# Barra dei Comandi: SETTING

## SETTING

L'opzione *Popup Dialog for Input syscall* consente l'**attivazione delle finestre di dialogo** utili per inserire parametri in ingresso che siano valori interi, reali o stringhe dopo la richiesta di una Chiamata di Sistema



# Barra dei Comandi: TOOLS

## TOOLS

Questa sezione supporta una **visione analitica ed in tempo reale del funzionamento del processore**. Per utilizzare i diversi servizi è necessario dapprima assemblare il programma; poi bisogna scegliere il servizio e connetterlo al processore MIPS; infine si esegue il programma e si studia l'evoluzione dell'elaborazione nei pannelli dedicati.

The screenshot shows the MARS 4.5 assembly editor interface. On the left, the assembly code for 'mips1.asm' is displayed:mips1.asm\* Cod\_3\_A.  
#####  
#Descrizione: Un pr  
#Informazioni: Operai  
#Analisi:  
#Data Progra  
#2021-01-01 Franc  
#####  
.text  
.globl main  
main:  
 lw \$t0, valore1  
 lw \$t1, valore2  
 add \$t2,\$t0,\$t1  
 sw \$t2,risultato  
 li \$v0, 10  
 syscall  
 #Linea vuota non richiesta, ma lasciata per una maggiore leggibilità  
.data  
valore1 : .word 23  
valore2 : .word 12  
risultato: .word 0  
#####

The central pane shows the assembly code with annotations in green. The right pane displays the register values:| Name | Number | Value |
| --- | --- | --- |
| szero | 0 | 0 |
| sat | 1 | 0 |
| \$v0 | 2 | 0 |
| \$v1 | 3 | 0 |
| sa0 | 4 | 0 |
| sal | 5 | 0 |
| sa2 | 6 | 0 |
| sa3 | 7 | 0 |
| st0 | 8 | 0 |
| st1 | 9 | 0 |
| st2 | 10 | 0 |
| st3 | 11 | 0 |
| st4 | 12 | 0 |
| st5 | 13 | 0 |
| st6 | 14 | 0 |
| st7 | 15 | 0 |
| ss0 | 16 | 0 |
| ss1 | 17 | 0 |
| ss2 | 18 | 0 |
| ss3 | 19 | 0 |
| ss4 | 20 | 0 |
| ss5 | 21 | 0 |
| ss6 | 22 | 0 |
| ss7 | 23 | 0 |
| st8 | 24 | 0 |
| st9 | 25 | 0 |
| sk0 | 26 | 0 |
| sk1 | 27 | 0 |
| sgp | 28 | 268468224 |
| ssp | 29 | 2147479548 |
| efp | 30 | 0 |
| fra | 31 | 0 |
| pc |  | 4194304 |
| hi |  | 0 |
| lo |  | 0 |

# Barra dei Comandi: HELP

## HELP

L'ultima sezione riporta i **crediti** sul gruppo di lavoro che ha realizzato il simulatore e il **set di istruzione** più alcuni esempi

The screenshot shows the MARS 4.5 assembly editor interface. The main window displays the assembly code for a MIPS program named `mips1.asm*`. The code defines two variables (`valore1` and `valore2`) in the `.data` section, both initialized to 23 and 12 respectively. In the `.text` section, the `main` routine adds these values and stores the result (`risultato`) at address 0. The editor also includes a memory dump panel and a messages panel.

**Registers Panel:**

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194304
hi		0
lo		0



**Un primo programma**

# Un primo programma

Il primo programma proposto è la **somma di due operandi siti in due locazioni della Memoria dei Dati**, rappresentate dalle variabili **valore1** e **valore2**, e l'archiviazione del risultato in un nuovo indirizzo in memoria

Etichetta	Direttiva/Istruzione	Commento
1	<b>.text</b>	#Direttiva del Segment text (Segmento Testo)
2	<b>.globl main</b>	#Direttiva che indica la subroutine con etichetta main come globale
3	<b>main:</b>	#Etichetta main: inizio del programma
4	<b>lw \$t0, valore1</b>	#Prelievo del primo operando dalla locazione di memoria con etichetta valore1 #nel registro \$t0
5	<b>lw \$t1, valore2</b>	#Prelievo del secondo operando dalla locazione di memoria con etichetta #valore2 nel registro \$t1
6	<b>add \$t2,\$t0,\$t1</b>	#Somma degli operandi e scrittura del risultato nel registro \$t2
7	<b>sw \$t2,risultato</b>	#Salvataggio dell'addizione nella locazione di memoria risultato
8	<b>li \$v0, 10</b>	#Richiesta del servizio di interruzione del programma
9	<b>syscall</b>	#Attivazione del servizio
10		#Linea vuota non richiesta, ma lasciata per una maggiore leggibilità
11	<b>.data</b>	#Direttiva del Segment Data (Segmento Dati)
12	<b>valore1 : .word 23</b>	#Definizione della variabile valore1 (intero a 32bit inizializzato a 23)
13	<b>valore2 : .word 12</b>	#Definizione della variabile valore2 (intero a 32bit inizializzato a 12)
14	<b>risultato : .word 0</b>	#Definizione della variabile risultato (intero a 32bi inizializzato a 0, pulizia)

# Un primo programma

## Inserimento operandi da tastiera e stampa del risultato su videoterminal

Il programma può essere reso più usabile, consentendo all'utente di inserire gli operandi attraverso la tastiera e mostrando il risultato sul videoterminal

	Etichetta	Direttiva/Istruzione	Commento
1		.text	#Direttiva del Segmento Testo
2		.globl main	#Direttiva che indica la subroutine con etichetta main come globale
3	main:		#Etichetta <b>main</b> : inizio del programma
4		li \$v0, 5	#Richiesta del servizio per la lettura di un intero da tastiera
5		syscall	#Attivazione del servizio (l'operando acquisito è in \$v0) <i>#Il programma rimane in attesa fino a quando l'utente non inserisce un valore intero nella finestra di dialogo proposta dal sistema e preme il tasto Enter</i>
6		move \$t0,\$v0	#Copia del primo operando nel registro temporaneo \$t0
7		li \$v0,5	#Richiesta del servizio per la lettura di un intero da tastiera
8		syscall	#Attivazione del servizio (l'operando è in \$v0)
9		move \$t1,\$v0	#Copia del secondo operando nel registro temporaneo \$t1
10		add \$t2,\$t0,\$t1	#Somma degli operandi
11		move \$a0,\$t2	#Copia del risultato nel registro adibito al passaggio di parametri per la routine di servizio di stampa a videoterminal
12		li \$v0,1	#Richiesta del servizio di stampa a videoterminal
13		syscall	#Attivazione del servizio <i>#La stampa avviene nella console del simulatore MARS (Run I/O Panel)</i>
14		li \$v0,10	#Richiesta del servizio di terminazione del programma
15		syscall	#Attivazione del servizio

# Un primo programma

## Somma operandi con inserimento dati e stampa di stringhe informative

Per avere una migliore leggibilità dei risultati mostrati in uscita si anticipa anche la routine di sistema che consente la stampa di una stringa sul videoterminal.

```
1      .text          #Direttiva del Segmento Testo
2      .globl main    #Direttiva che indica la subroutine con etichetta main come globale
3 main:                                #Etichetta main: inizio del programma
4      li $v0, 5       #Richiesta del servizio per la lettura di un intero da tastiera
5      syscall        #Attivazione del servizio (l'operando è in $v0)
6      move $t0,$v0    #Copia del primo operando nel registro temporaneo $t0
7      li $v0,5       #Richiesta del servizio per la lettura di un intero da tastiera
8      syscall        #Attivazione del servizio (l'operando è in $v0)
9      move $t1,$v0    #Copia del secondo operando nel registro temporaneo $t1
10     add $t2,$t0,$t1 #Somma degli operandi
11     li $v0,4       #Richiesta del servizio di stampa su videoterminal di una stringa
12     la $a0,stringa #Settaggio del parametro (si pone in $a0 l'indirizzo iniziale della stringa che si vuole stampare)
13     syscall        #Attivazione del servizio
14     move $a0,$t2    #Copia del risultato nel registro adibito al passaggio di parametri per la routine di servizio di stampa a videoterminal
15     li $v0,1       #Richiesta del servizio di stampa a videoterminal
16     syscall        #Attivazione del servizio
17     li $v0,10      #La stampa avviene nella consolle del simulatore MARS (Run I/O Panel)
18     syscall        #Richiesta del servizio di terminazione del programma
19
20     .data          #Direttiva del Segmento Data
21     stringa:.asciiz "La somma dei due operandi e': " #Definizione di una stringa
```



**FINE**