Università degli Studi di Milano
Dipartimento di Informatica “Giovanni Degli Antoni”
Corso di Laurea Triennale in Informatica

Architettura degli Elaboratori II Laboratorio

I «tools» di Mars

Aggiungere funzionalità a MARS

MARS può essere esteso sotto tre diversi aspetti

- Realizzazione di tools con cui i programmi assembly che scriviamo ed eseguiamo all'interno del simulatore possono interagire sia attivamente che passivamente
- Aggiungere syscall o riassegnarne il numero
- Estendere l'instruction set, mediante la definizione di nuove pseudo-istruzioni

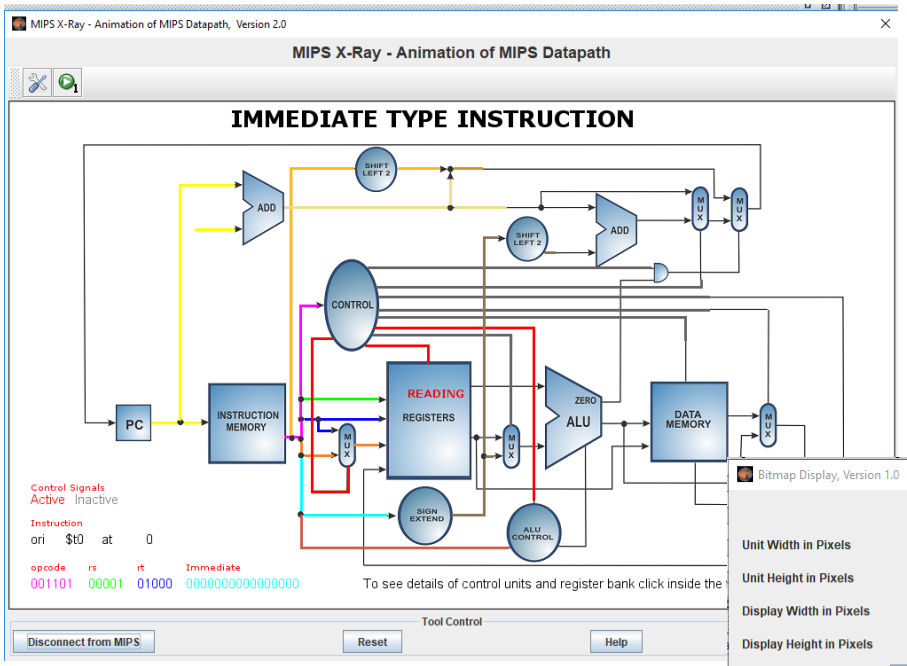
Per approfondire come estendere MARS:

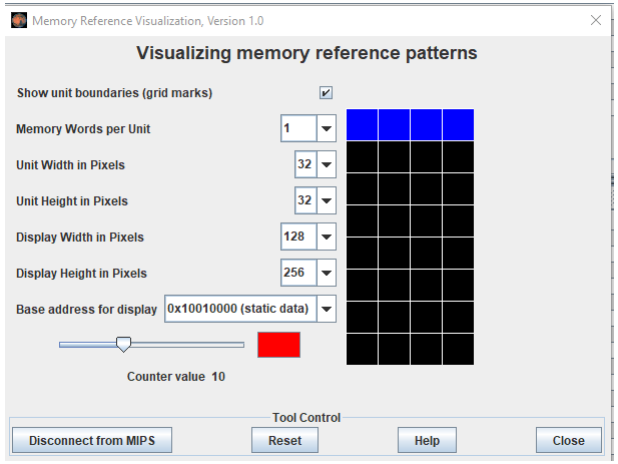
http://courses.missouristate.edu/KenVollmar/mars/Help/MarsHelpTools.html

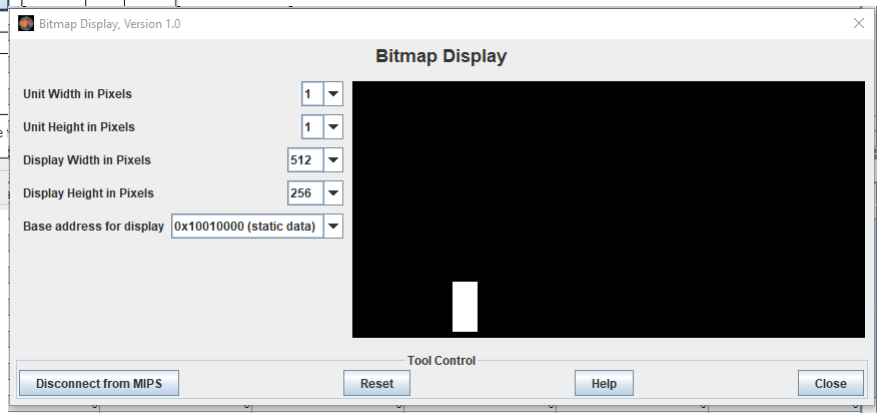
https://courses.missouristate.edu/KenVollmar/mars/CCSC-CP%20material/MARS%20Tutorial.doc

Cosa sono i tools in MARS?

I tools sono programmi esterni collegati a MARS che osservano l'attività della memoria e dei registri all'interno del simulatore e/o ne registrano l'attività durante l'esecuzione di un programma per scopi differenti.







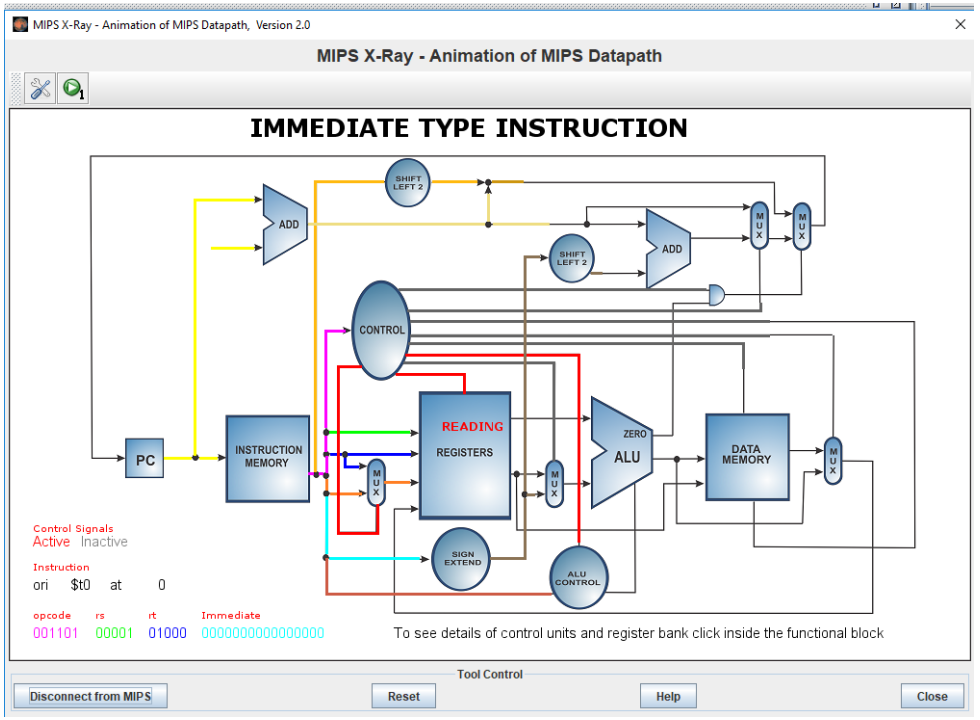
Tools utili (1)

MIPS X-Ray

Il tool permette di visualizzare il comportamento di un processore MIPS, generando un'animazione per ogni istruzione eseguita, evidenziandone il percorso seguito all'interno del data path

Utilizzo:

- Assemblare il programma da eseguire
- Aprire il tool dal menu: tools/MIPS-X-Ray
- Cliccare su Connect to MIPS
- Cliccare sul pulsante play in alto a sinistra nella finestra del tool per eseguire la prossima istruzione



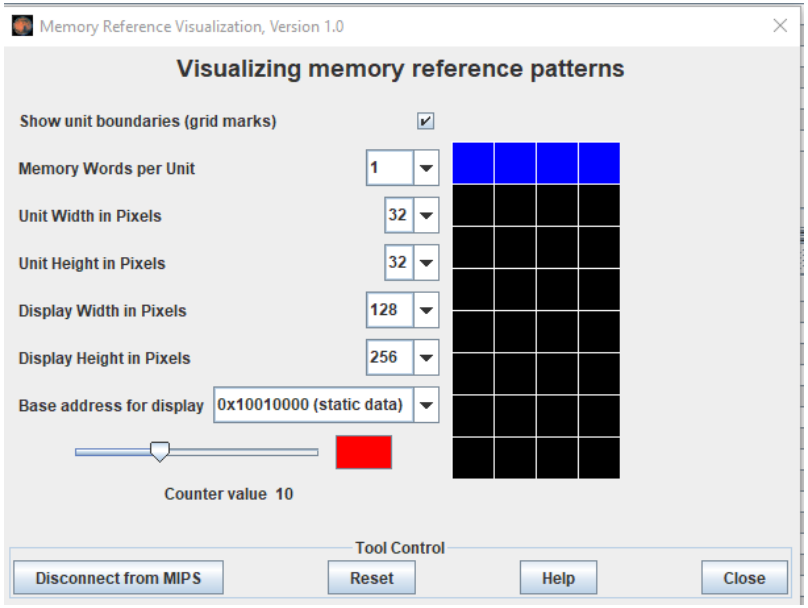
Tools utili (2)

Memory Reference Visualization

Il tool permette di visualizzare graficamente pattern di indirizzamento in memoria. Opera colorando celle di memoria contenenti 1 o più word di un colore diverso ogni volta che quella cella è letta o scritta, dimensioni di blocchi e griglia possono essere configurate a piacere per determinare l'area di memoria da osservare

Utilizzo:

- Assemblare il programma da eseguire
- Aprire il tool dal menu: tools/Memory Reference Visualization
- Cliccare su Connect to MIPS
- Cliccare sul pulsante play nell'interfaccia principale di MARS per avviare il programma



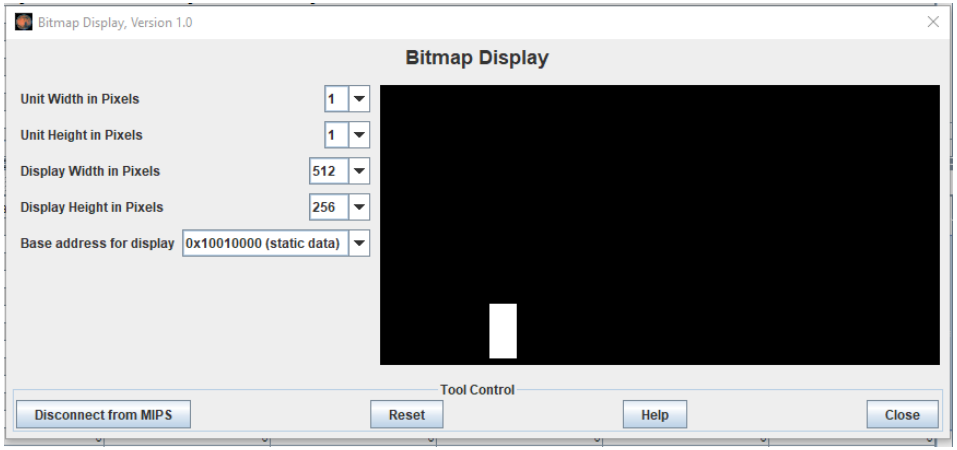
Tools utili (3)

Bitmap Display

Il tool emula la presenza di un display connesso alla macchina emulata. È possibile accedere al frame-buffer del display e quindi settare i colori dei pixel semplicemente scrivendo i colori in un area precedentemente allocata di memoria statica (.data)

Utilizzo:

- Assemblare il programma da eseguire
- Aprire il tool dal menu: tools/Bitmap Display
- Cliccare su Connect to MIPS
- Cliccare sul pulsante play nell'interfaccia principale di MARS per avviare il programma



Note integrative su caratteri e stringhe

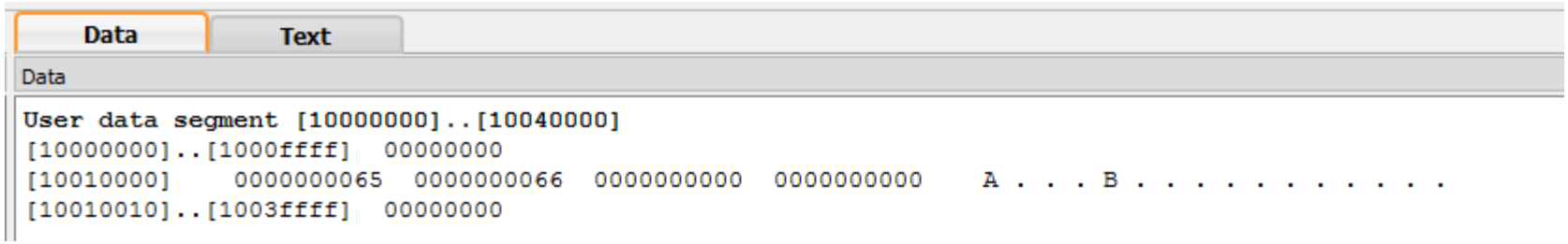
Note su caratteri e stringhe

Service	System Call Code	Arguments	Result
print_int	1	$a0 = integer	
print_float	2	$f12 = float	
print_double	3	$f12 = double	
print_string	4	$a0 = string	
read_int	5		integer (in $v0)
read_float	6		float (in $f0)
read_double	7		double (in $f0)
read_string	8	$a0 = buffer, $a1 = length	
sbrk	9	$a0 = amount	address (in $v0)
exit	10		
print_character	11	$a0 = character	
read_character	12		character (in $v0)
open	13	$a0 = filename,	file descriptor (in $v0)
		$a1 = flags, $a2 = mode	
read	14	$a0 = file descriptor,	bytes read (in $v0)
		$a1 = buffer, $a2 = count	
write	15	$a0 = file descriptor,	bytes written (in $v0)
		$a1 = buffer, $a2 = count	
close	16	$a0 = file descriptor	0 (in $v0)
exit2	17	$a0 = value	

Note su caratteri e stringhe

```
.data
arr: .space 8

.text
main:

# leggo il primo carattere
li $v0, 12
syscall
move $s1, $v0

# scrivo il carattere in memoria
la $t0, arr
sw $s1 0($t0)
```

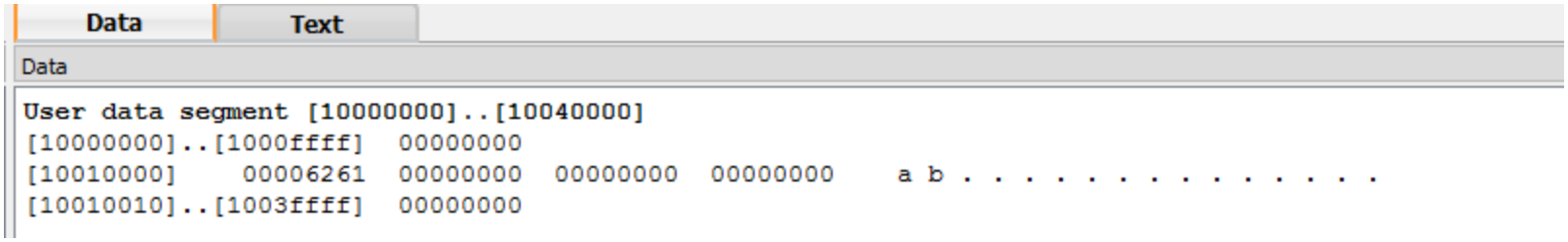
```
# leggo il secondo carattere
li $v0, 12
syscall
move $s1, $v0

# scrivo il carattere in memoria
la $t0, arr
sw $s1 4($t0)

# exit
li $v0, 10
syscall
```



Note su caratteri e stringhe

```
.data
arr: .space 2

.text
main:

# leggo il primo carattere
li $v0, 12
syscall
move $s1, $v0

# scrivo il carattere in memoria
la $t0, arr
sb $s1 0($t0)
```

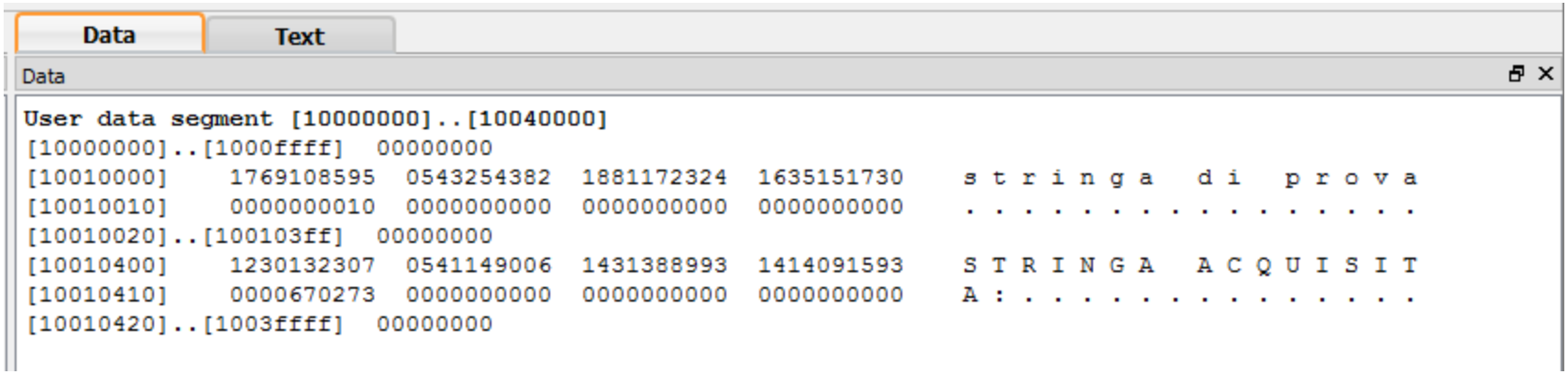
```
# leggo il secondo carattere
li $v0, 12
syscall
move $s1, $v0

# scrivo il carattere in memoria
la $t0, arr
sb $s1 1($t0)

# exit
li $v0, 10
syscall
```



Note su caratteri e stringhe

```
.data
buffer: .space 1024
s: .asciiz "STRINGA
ACQUISITA:\n"

.text
main:
# acquisizione di un stringa
li $v0, 8 # read string
la $a0, buffer
li $a1, 1024
syscall
```

```
# stampo la stringa acquisita
li $v0, 4 # print string
la $a0, s
syscall

li $v0, 4 # print string
la $a0, buffer
syscall

jr $ra
```



Università degli Studi di Milano
Dipartimento di Informatica “Giovanni Degli Antoni”
Corso di Laurea Triennale in Informatica

Architettura degli Elaboratori II Laboratorio