Simuladores MIPS

Prof. Tiago Gonçalves Botelho

SPIN

- Simulador “clássico” multi-plataformas open-source, suas versões mais novas são desenvolvidas em Qt.
- Download: http://spimsimulator.sourceforge.net/
- Documentação: http://pages.cs.wisc.edu/~larus/spim.html#information

MARS

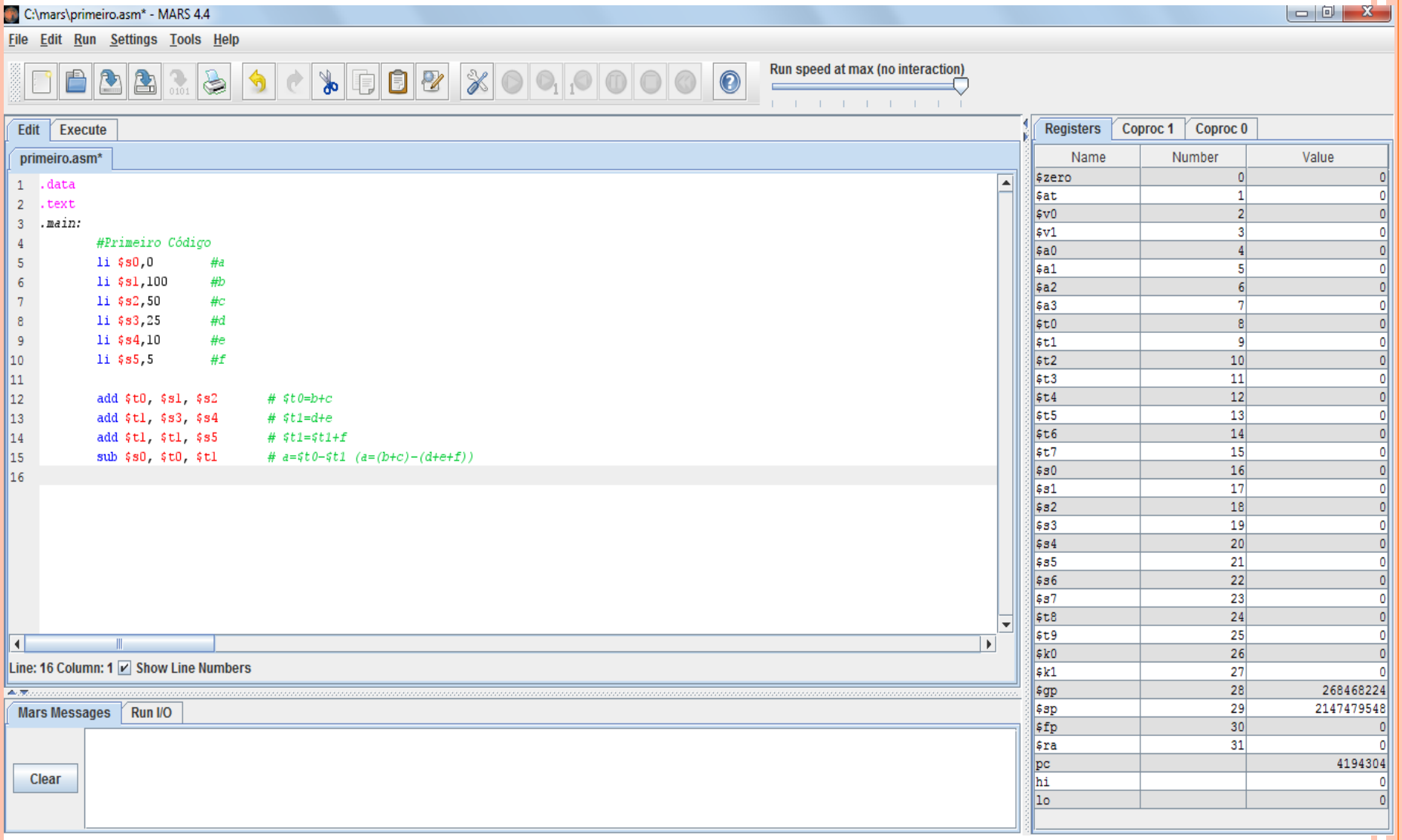
- Simulador “moderno” multi-plataformas, desenvolvido em JAVA é open-source.

- Download: http://courses.missouristate.edu/KenVollmar/MARS/download.htm

- Documentação: http://courses.missouristate.edu/KenVollmar/MARS/Help/MarsHelpIntro.html

MARS

- Fazendo o primeiro código assembly no MARS.

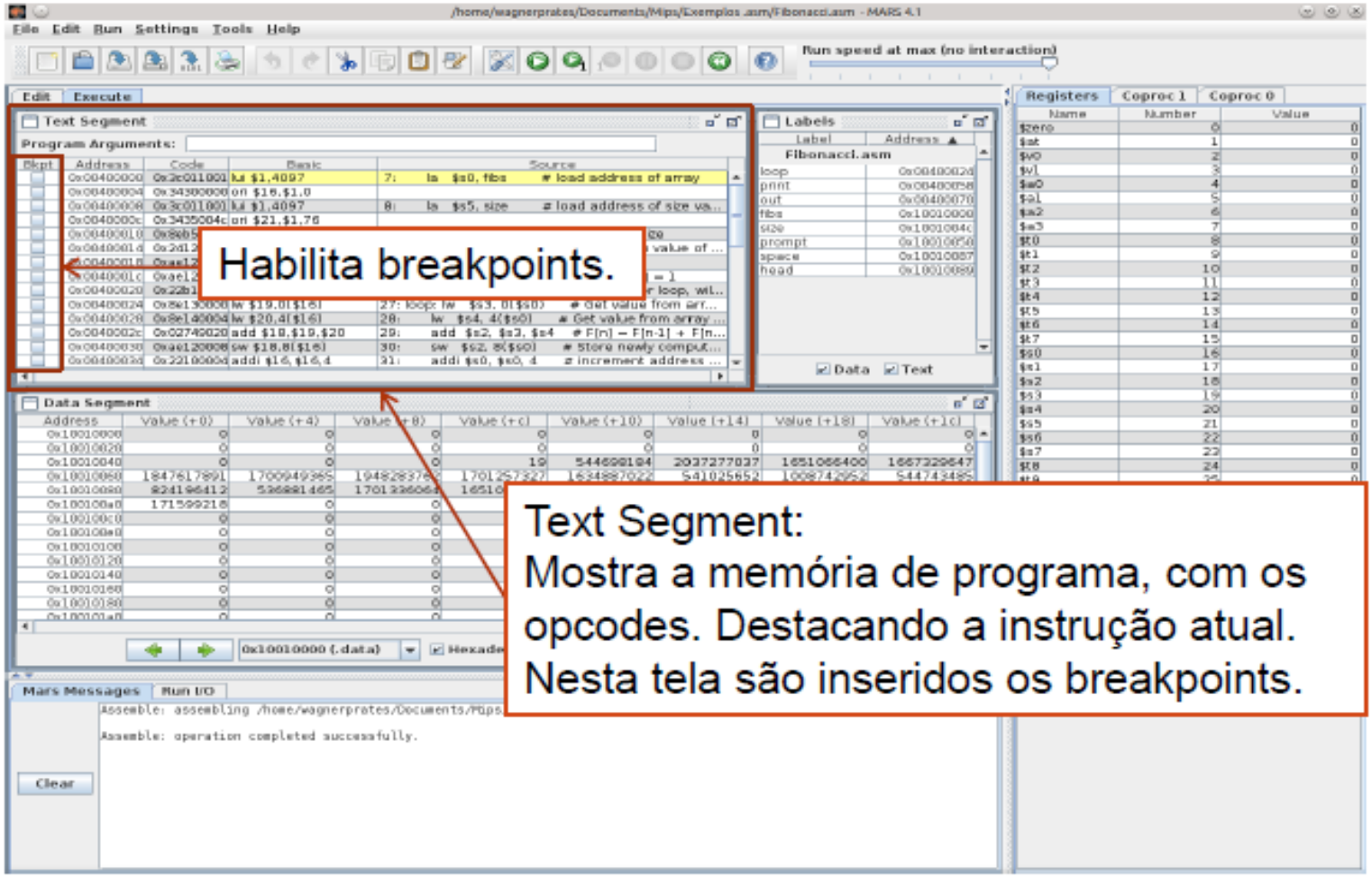


MARS

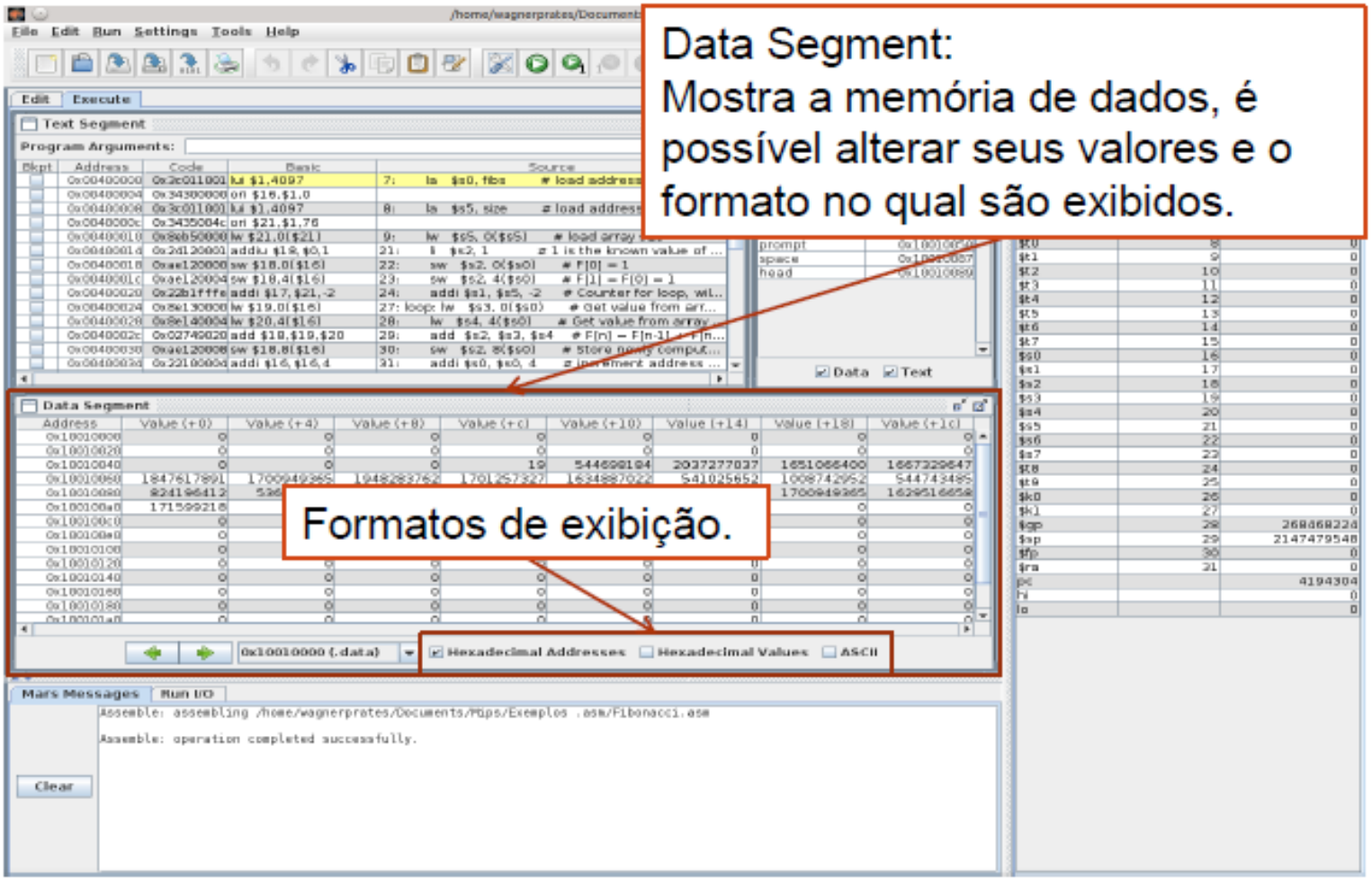
- INICIANDO A SIMULAÇÃO:
  - Para se realizar a simulação é necessário 'montar' o programa, para isso deve-se clicar no ícone, localizado no Menu RUN->ASSEMBLE:

  - Se o programa contiver pseudoinstruções é necessário habilitá-las.
  - O Mars NÃO tem suporte a pipeline, porém pode simular o atraso nos saltos.

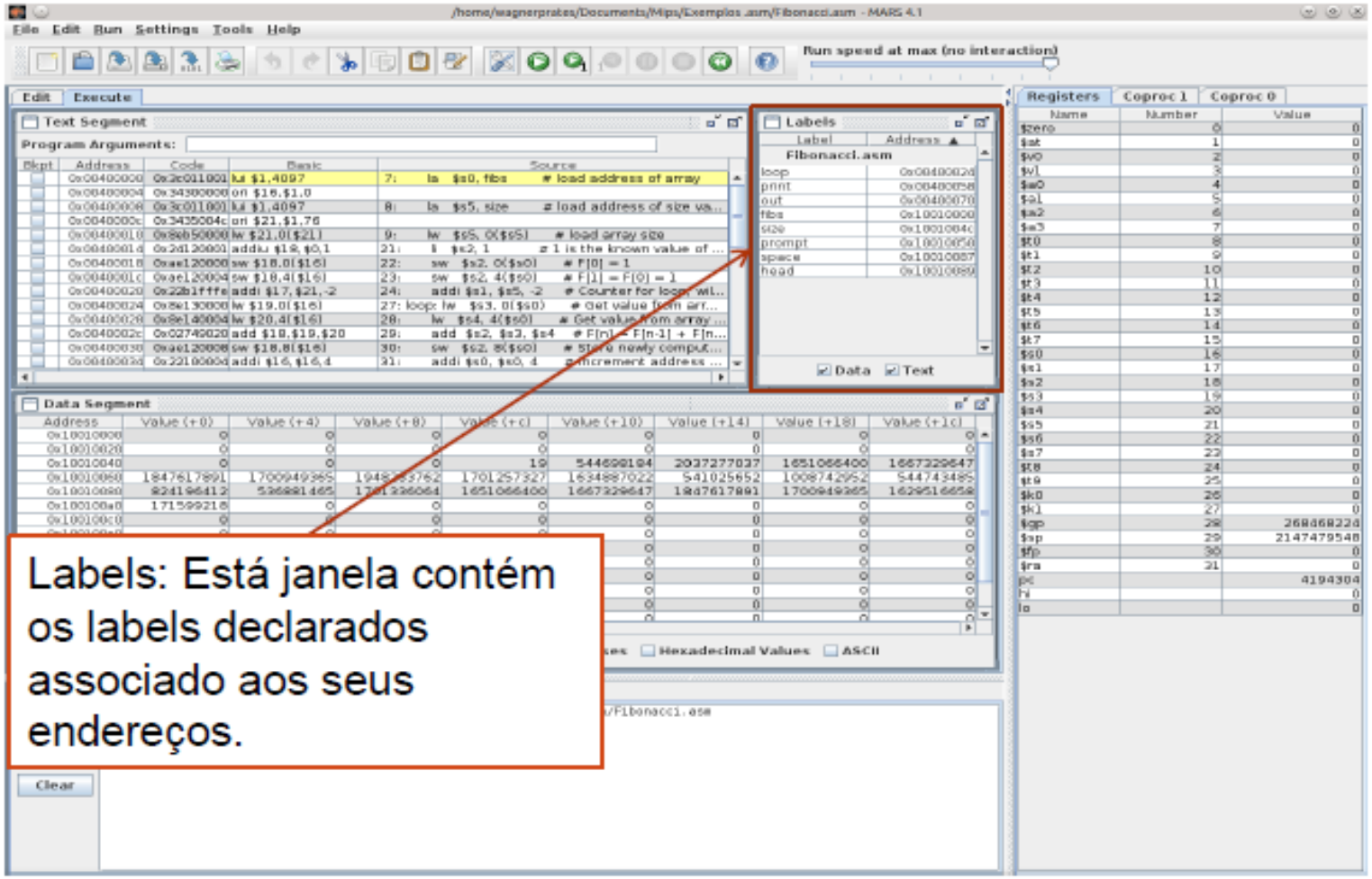
MARS


Habilita breakpoints.
Text Segment:
Mostra a memória de programa, com os opcodes. Destacando a instrução atual.
Nesta tela são inseridos os breakpoints.

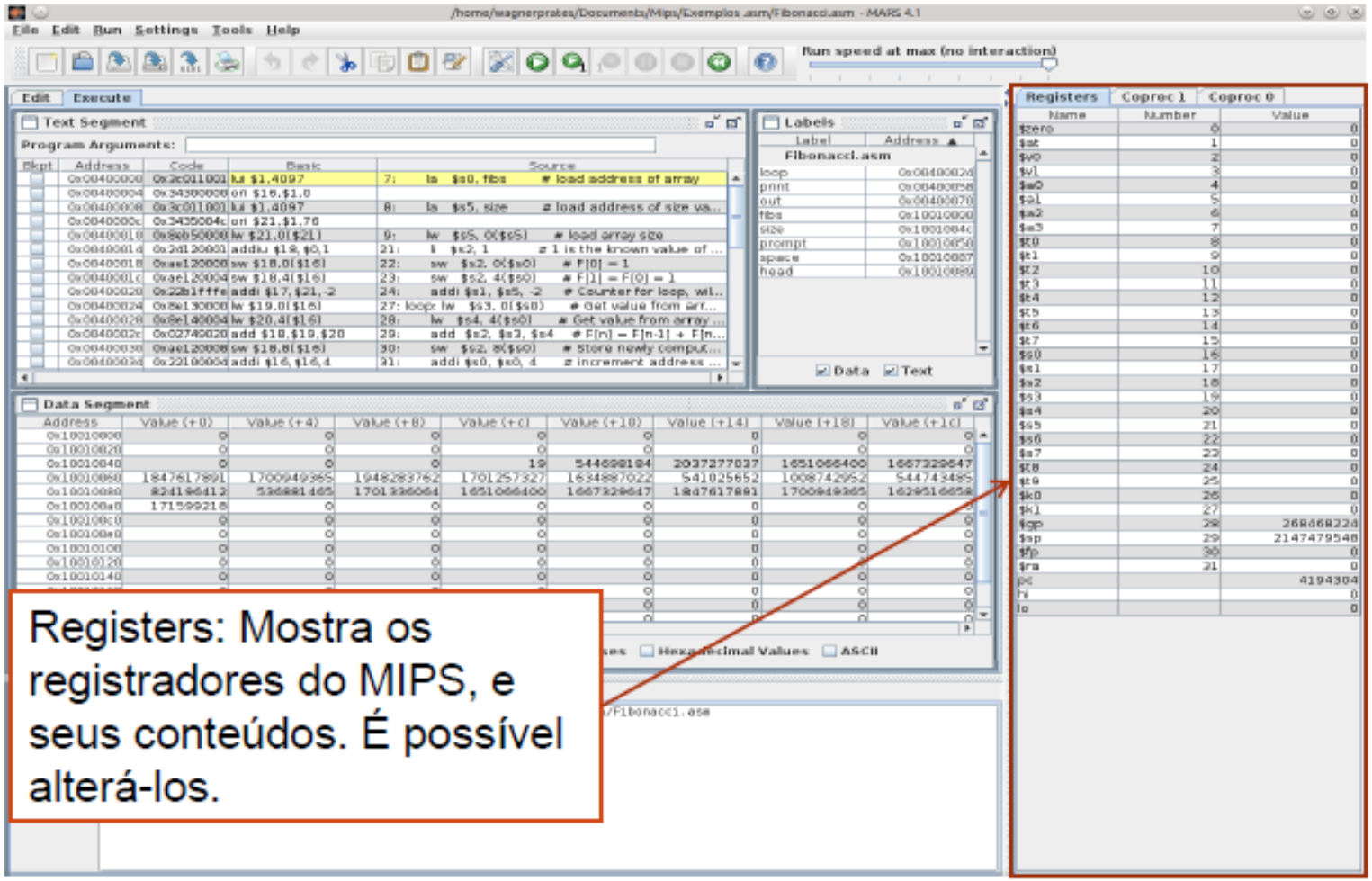
MARS


Data Segment:
Mostra a memória de dados, é possível alterar seus valores e o formato no qual são exibidos.
Formatos de exibição.

MARS

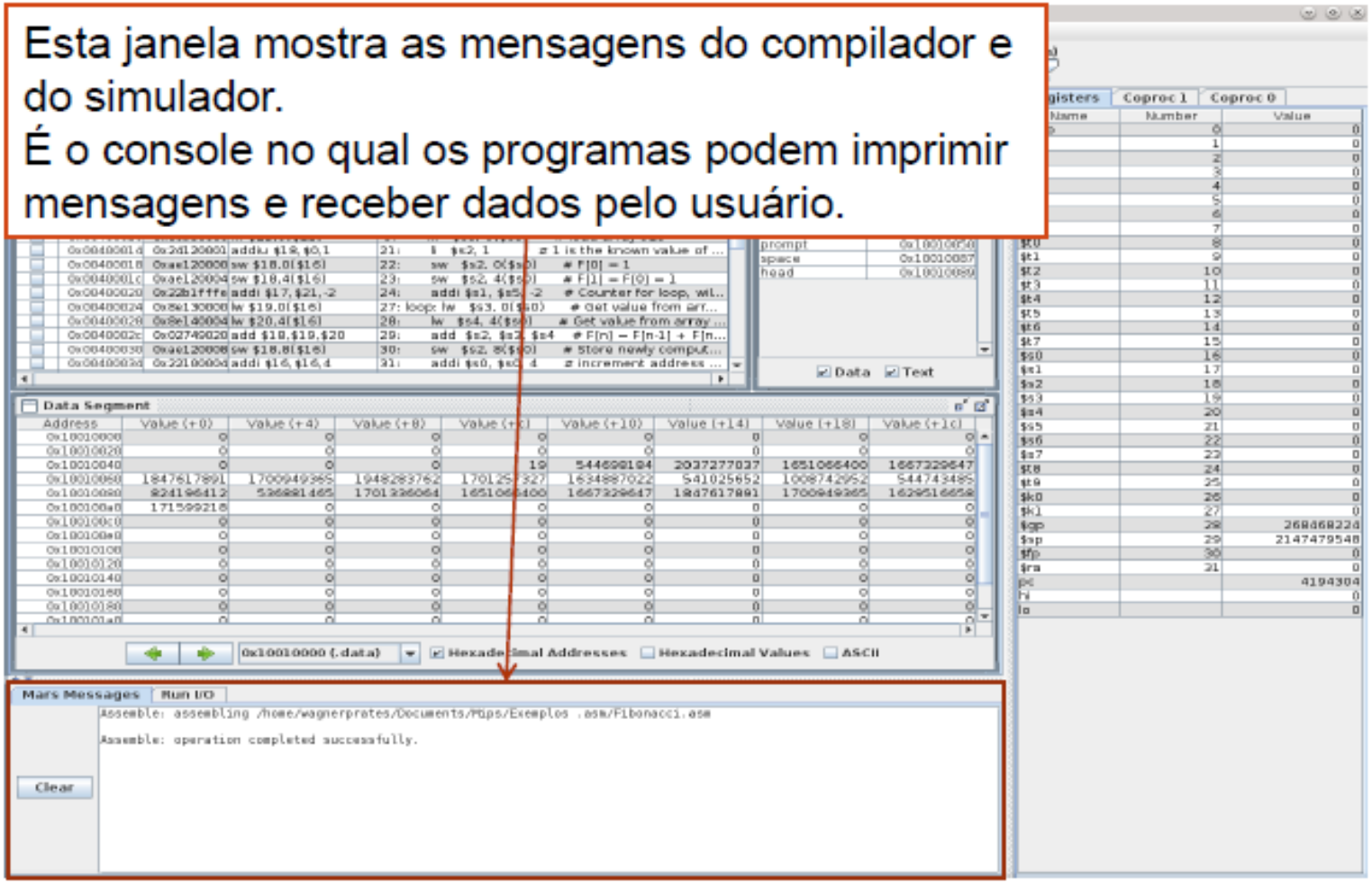

Labels: Está janela contém os labels declarados associado aos seus endereços.

MARS


Registers: Mostra os registradores do MIPS, e seus conteúdos. É possível alterá-los.

MARS

Esta janela mostra as mensagens do compilador e do simulador.
É o console no qual os programas podem imprimir mensagens e receber dados pelo usuário.



MARS - Simulação

Executa o programa até o fim ou próximo breakpoint.

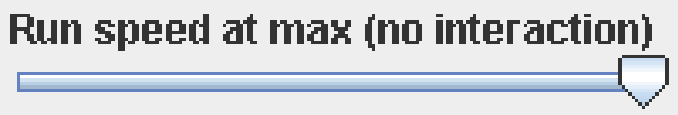
Executa a instrução atual.

Desfaz última instrução.

Reinicia o simulador, registradores e memória.

MARS - Simulação

- A barra deslizante controla a velocidade de execução.



- Permitindo assim “ver a ação” ao invés do resultado final somente se a velocidade for reduzida.

MIPS em resumo – (1/2)

```
li   $r0, C              $r0 <- C
lui  $r0, C              $r0 <- 2^16*C
move $r0, $r1            $r0 <- $r1

add  $r0, $r1, $r        $r0 <- $r1 + $r2
addi $r0, $r1, C         $r0 <- $r1 + C
sub  $r0, $r1, $r2       $r0 <- $r1 - $r2
div  $r0, $r1, $r2       $r0 <- $r1 / $r2
div  $r1, $r2            $lo <- $r1 / $r2, $hi <- $r1 mod $r2
mul  $r0, $r1, $r2       $r0 <- $r1 * $r2 (sem overflow)
neg  $r0, $r1            $r0 <- -$r1

slt  $r0, $r1, $r2       $r0 <- 1 se $r1 <  $r2, $r0 <- 0 senão
slti $r0, $r1, C         $r0 <- 1 se $r1 <  C,   $r0 <- 0 senão
sle  $r0, $r1, $r2       $r0 <- 1 se $r1 <= $r2, $r0 <- 0 senão
seq  $r0, $r1, $r2       $r0 <- 1 se $r1 =  $r2, $r0 <- 0 senão
sne  $r0, $r1, $r2       $r0 <- 1 se $r1 <> $r2, $r0 <- 0 senão
```

MIPS em resumo – (2/2)

```
la      $r0, adr              $r0 <- adr

lw      $r0, adr              $r0 <- mem[adr]

sw      $r0, adr              mem[adr] <- $r0

beq     $r0, $r1, label       salto se $r0 = $r1
beqz    $r0, label            salto se $r0 = 0
bgt     $r0, $r1, label       salto se $r0 > $r1
bgtz    $r0, label            salto se $r0 > 0


beqzal  $r0, label            salto se $r0 = 0, $ra <- $pc + 1
bgtzal  $r0, label            salto se $r0 > 0, $ra <- $pc + 1

j       label                 salto para label
jal     label                 salto para label, $ra <- $pc + 1
jr      $r0                   salto para $r0
jalr    $r0                   salto para $r0, $ra <- $pc + 1
```