RA: 120875

Arquitetura e Organização de Computadores

Tutorial sobre como configurar e utilizar a janela gráfica do MARS, juntamente com interface de teclado.

Aluno: Micael de Oliveira Pimpim

Docente: Prof. Dr. Fábio Cappabianco

Universidade Federal de São Paulo - UNIFESP

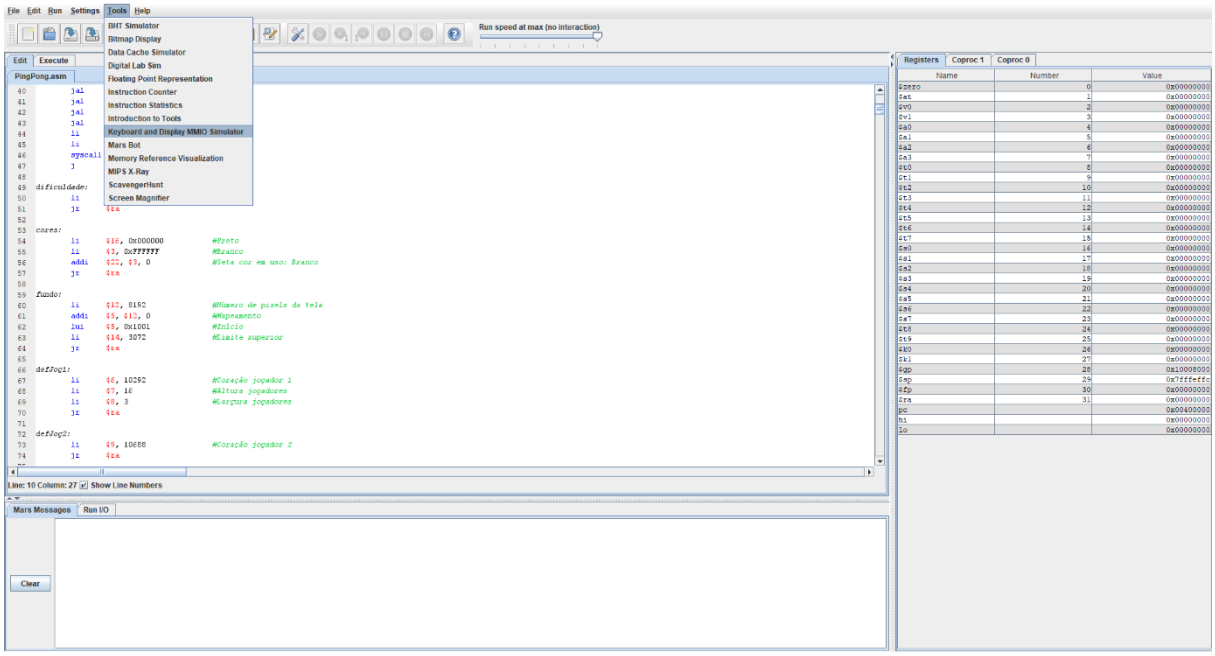
Instituto de Ciência e Tecnologia - Campus São José dos Campos

São José dos Campos - Brasil

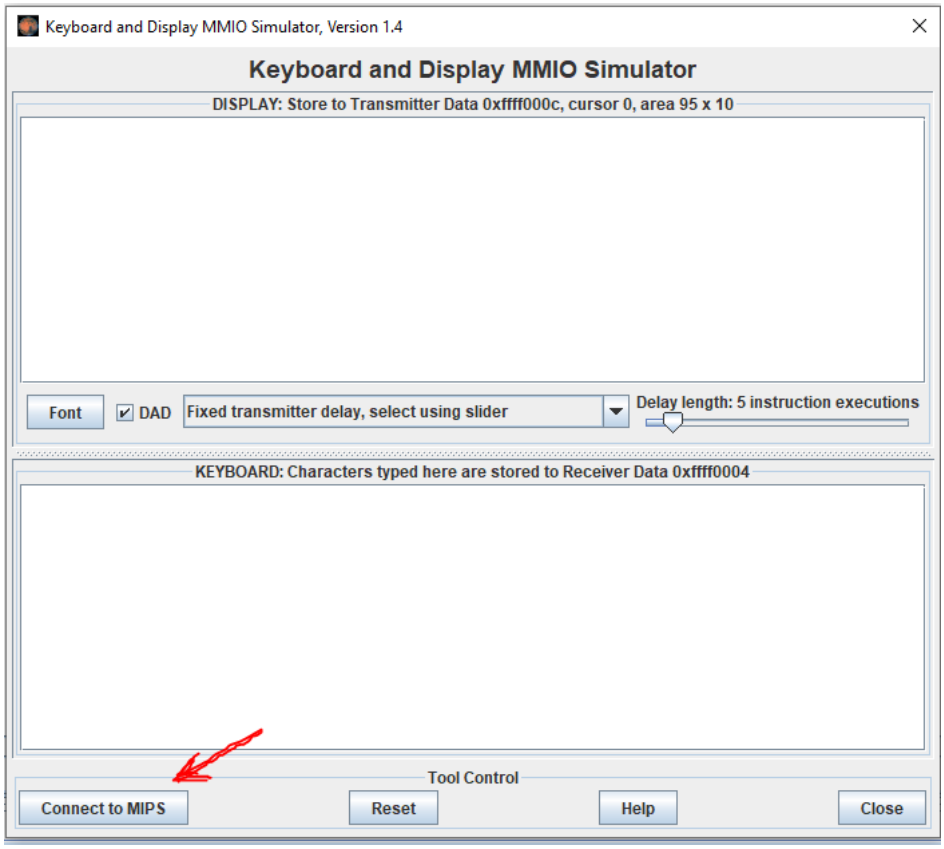
Dezembro de 2019

1 KEYBOARD AND DISPLAY MMIO SIMULATOR

Quando desejamos fazer a leitura de um comando em teclado no *MARS*, podemos utilizar o *Keyboard and Display MMIO Simulator*, que é uma funcionalidade disponível na sessão *Tools* no menu superior do *MARS*.

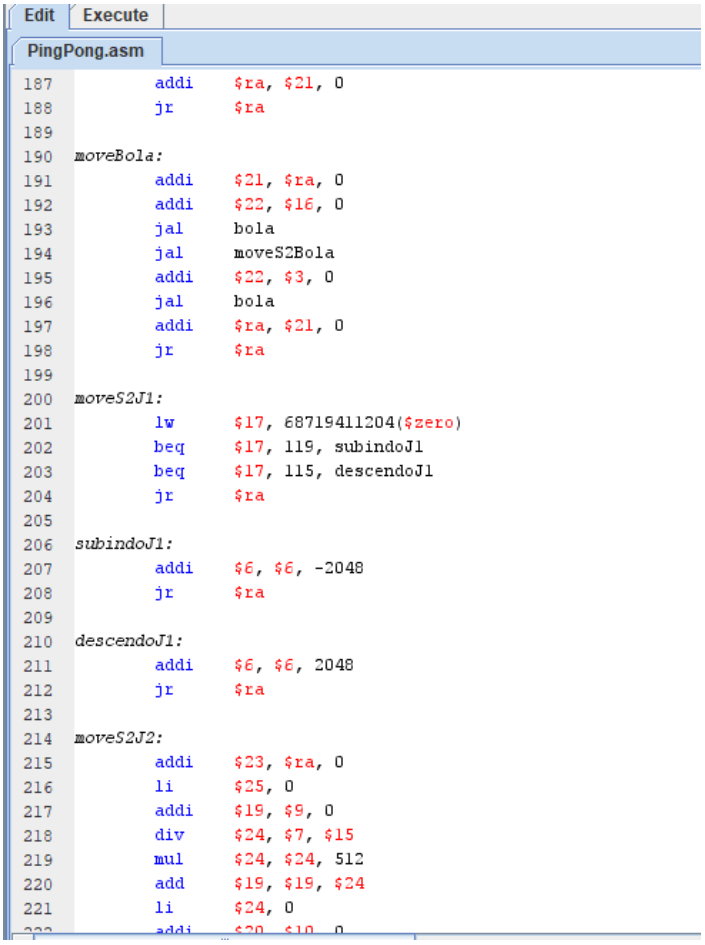


Ao clicar nessa opção, uma janela de interação será aberta, na qual deve-se clicar em *Connet to MIPS* para iniciar a conexão.



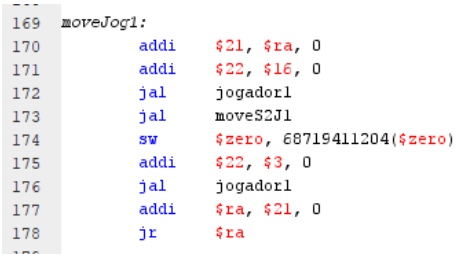
A partir desse momento, quando um programa estiver rodando, ao se digitar alguma tecla no teclado, tal tecla será impressa na região branca inferior e o correspondente em *ASCII* dessa tecla será salvo na posição da memória 0xffff0004, que equivale a 68719411204 em decimal. Tal posição da memória pode então ser acessada por meio de um *lw* ou *load word* que acesse tal posição.

No exemplo abaixo podemos ver uma implementação dessa lógica na linha 201:



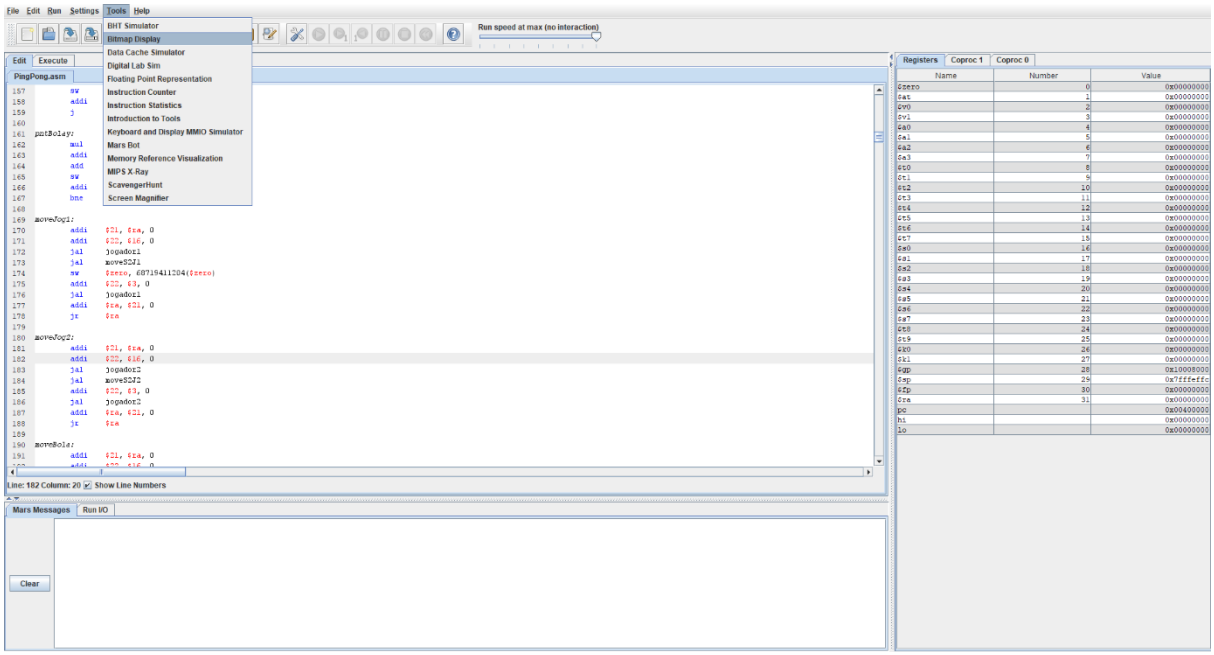
A posição 68719411204 da memória é acessada por meio de um deslocamento imediato de 68719411204 com relação ao registrador $zero que contém o zero, e seu conteúdo é passado para o registrador $17. A partir desse momento, o registrador $17 pode ser comparado com os números 119 e 115 (que representam os códigos *ASCII* para “w” e “s” respectivamente) nas linhas 202 e 203 para implementar uma lógica de movimentação para cima ou para baixo.

Durante a execução do programa, sempre que essa seção do código for lida, o conteúdo dessa posição de memória será acessado e a última tecla digitada pelo usuário será interpretada. Vale lembrar que essa posição de memória não se limpa automaticamente, ou seja, provavelmente será interessante limpar essa posição da memória depois de efetuada a leitura, por exemplo com um comando de *sw* ou *store word*, como visto abaixo, na linha 174:

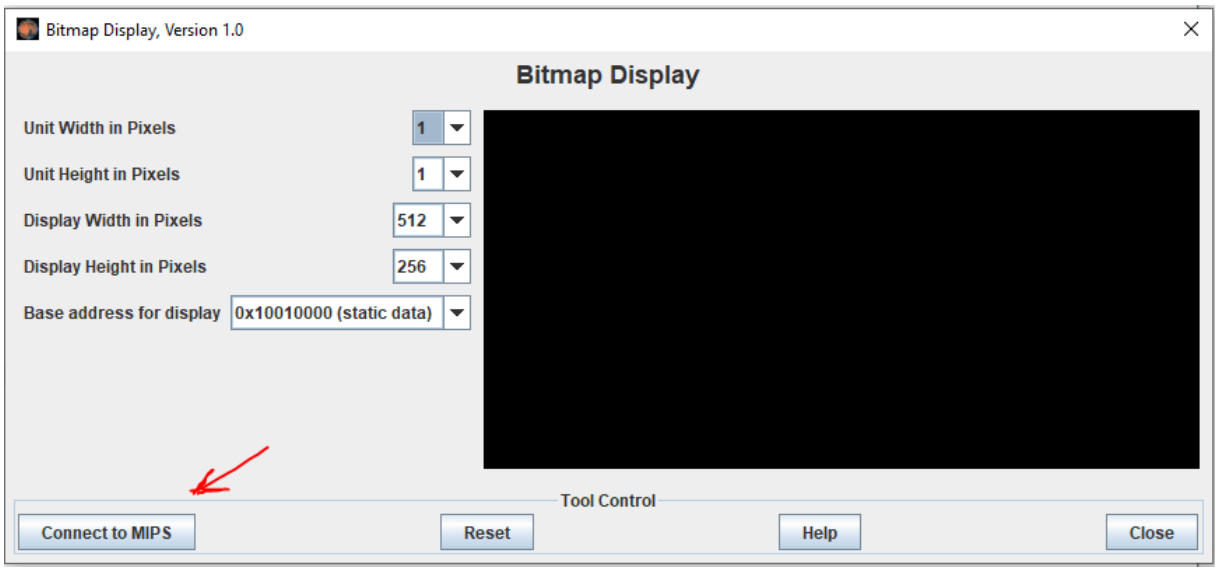


2 BITMAP DISPLAY

Quando desejamos utilizar uma janela gráfica no *MARS*, podemos utilizar o *Bitmap Display*, que é uma funcionalidade disponível na sessão *Tools* no menu superior do *MARS*.



Ao clicar nessa opção, uma janela de interação será aberta, na qual deve-se clicar em *Connet to MIPS* para iniciar a conexão.



Nessa janela existem opções de configuração, como largura e altura de uma unidade da interface gráfica em pixels, altura e largura do *display* em pixels e endereço base para o *display*. Normalmente usa-se 4 como altura e largura de uma unidade da interface gráfica em pixels. Já a altura e a largura do *display* dependem muito da intenção do programador, mas o mais comum é utilizar 512 para largura e 256 para altura (formato retangular padrão) ou 512 para ambos (formato quadrado padrão). Para endereço base é usual utilizar a opção marcada na figura acima.

Já no código, devem ser usados os seguintes comandos para configurar a posição do pixel inicial em um registrador:

```
59  fundo:
60          li      $12, 8192               #Número de pixels da tela
61          addi    $5, $12, 0              #Mapeamento
62          lui     $5, 0x1001              #Início
```

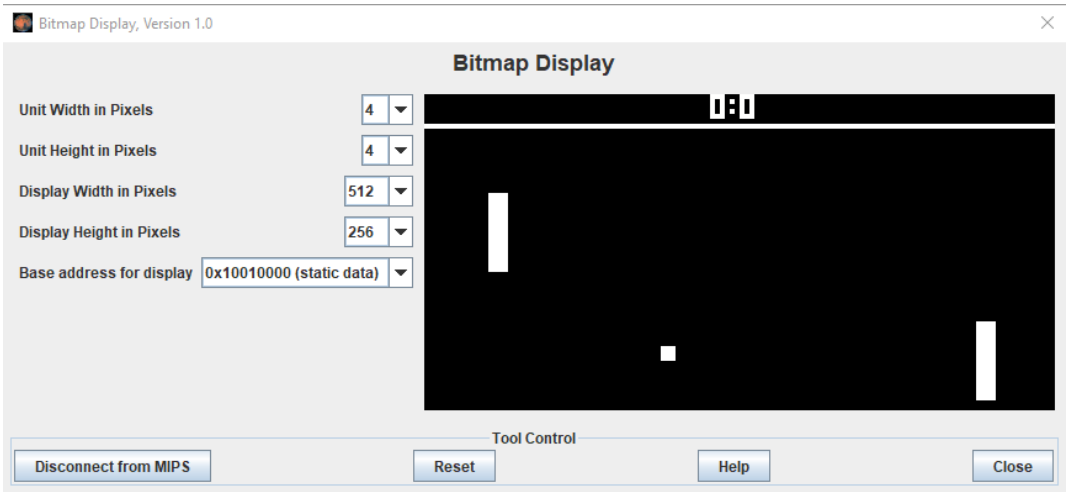
No código acima, o registrador $12 recebe a quantidade de pixels (unidades gráficas fazendo o papel de pixels) da tela. Se cada unidade gráfica possui 4 pixels de altura e 4 de largura, cada uma possui 16 pixels. A tela tem 512 x 256 no nosso exemplo, o que resulta no valor 131072, o qual dividido por 16 nos dá a quantidade de 8192 unidades gráficas do nosso *display* (número de pixels simulados).

O registrador $5 recebe então essa mesma quantidade. Isso se deve ao fato de que o conteúdo do registrador $5 provavelmente será alterado durante o código, mas a quantidade de pixels não se perderá pois estará sempre armazenada no registrador $12 no nosso exemplo, no qual não é interessante mexer para alterar seu conteúdo.

Em seguida, o comando *lui $5, 0x1001* indica qual a posição da memória vai receber a posição do primeiro pixel da tela. A partir desse momento, o registrador $5 vai apontar para essa posição e se um comando de *sw* ou *store word* for utilizado usando-se o registrador $5 como base, tal posição da memória pode assumir um valor hexadecimal que representa uma cor, como mostrado abaixo:

```
1485  pntWin4x:
1486          beqz    $25, contWin4
1487          addi    $5, $5, 4
1488          li      $22, 0xFFFFFE
1489          sw      $22, ($5)
1490          addi    $25, $25, -1
1491          j       pntWin4x
```

No código acima, na linha 1488 o registrador $22 recebe a cor branca representada por 0xFFFFFE em hexadecimal e então esse valor é salvo na posição de memória contida no registrador $5 (que representa um pixel na tela), fazendo com que tal pixel seja preenchido na janela do *Bitmap Display*, como mostrado abaixo:



Cada pixel branco na tela foi pintado utilizando-se essa técnica. O uso de uma planilha no excel com os valores das posições dos pixels é recomendado para auxiliar no mapeamento das posições dos pixels.