UNIVERSIDADE DE BRASÍLIA
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

116394 ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES

Trabalho III: Programação Assembly

OBJETIVO

Este trabalho objetiva a prática da programação em *assembly* do RISC-V utilizando os recursos do simulador RARS. O trabalho consiste em desenvolver uma implementação do *Jogo da Vida*, exibindo o resultado de sua evolução através da interface gráfica do RARS, com o auxílio da ferramenta de exibição de saída gráfica mapeada em memória.

DESCRIÇÃO

A. Display gráfico mapeado em memória

- O RARS oferece, dentre as suas ferramentas de apoio ao desenvolvimento de aplicações em *assembly* RISC-V, uma janela gráfica baseada em *pixels*. Suas principais características são as seguintes:
  - Resolução configurável, *default* 512 x 256 *pixels*.
  - Cada *pixel* representado por uma palavra de 32 bits, no formato RGB, um *byte* para cada cor. *Red* = 0x00FF0000, *Green* = 0x0000FF00, *Blue* = 0x000000FF.
  - *Display* mapeado em memória. O ponto superior esquerdo da tela corresponde ao pixel com coordenadas (0, 0). A coordenada *x* cresce para a direita e a coordenada *y* cresce para baixo.
  - O endereço correspondente ao primeiro pixel é configurável. Opções: *global data, global pointer, static data, heap, memory map.*
  - O desenho de um pixel na tela é realizado pela escrita de uma palavra contendo a descrição de sua cor RGB na posição de memória correspondente.

B. Jogo da Vida

O jogo da vida, na realidade não é um jogo, mas uma simulação proposta por John Conway, para o nascimento, morte e sobrevivência de micro-organismos. Nessa simulação, o mundo é representado por uma matriz onde cada posição pode conter ou não uma bactéria. As evoluções desse mundo se baseiam em um conjunto de regras muito simples:

1. **Nascimento**: Em cada posição vazia do mundo, nascerá uma nova bactéria se houverem 3 posições vizinhas contendo uma bactéria;
2. **Sobrevivência**: Uma bactéria sobreviverá na próxima geração se houverem 2 ou 3 outras bactérias em sua vizinhança;
3. **Morte**: Uma bactéria pode morrer por dois motivos:
    1. *Solidão*: Se houverem menos do que 2 bactérias nas posições vizinhas;
    2. *Fome*: Se houverem mais de 3.

Por vizinhança entendemos as 8 casas vizinhas na matriz que representa o mundo. Esse jogo pode ser descrito como uma simulação de um autômato celular bidimensional. Autômatos celulares são máquinas abstratas genéricas que são bastante utilizadas na prática, em diversos contextos. Por exemplo, é usada para a geração automática de padrões de teste em circuitos integrados auto-testáveis. A simulação proposta por Conway produz sequências de gerações

muitas vezes interessantes, e é bastante utilizada também como protetor de tela em diversos ambientes.

C. Implementação

Criar duas matrizes A e B no RARS de 16 x 16 elementos com valores booleanos. Pode-se utilizar o tipo ".byte" para armazenar os dados dessas matrizes. "0" indica ausência de indivíduo e "1" indica presença. A matriz A recebe os valores iniciais especificados pelo usuário e seu conteúdo inicial é exibido na tela gráfica. Note que o tamanho do pixel é configurável através da interface do RARS. A matriz A é processada, com o resultado sendo armazenado na matriz B. O processo se repete alternando-se A e B. Utilizar a chamada de sistema *sleep* para dar tempo de visualizar cada matriz exibida.

Cada célula [i][j] da matriz tem como vizinhos as células [i-1][j-1], [i-1][j], [i-1][j+1], [i][j-1], [i][j+1], [i+1][j-1], [i+1][j] e [i+1][j+1]. Os vizinhos que se situam fora da matriz (bordas) tem valor "0" .

O programa deve conter ao menos as seguintes funções:

- `readm(i, j, mat) - onde` ***`a0`*** `= i,` ***`a1`*** `= j e` ***`a2`*** `= endereço inicial da matriz: retorna o valor da célula da matriz em` ***`a0`***`.`
- `write(i, j, mat) - idem, onde a escrita` ***`inverte`*** `o valor da célula (xor)`
- `plotm(mat) - onde` ***`a0`*** `= endereço inicial da matriz: desenha a matriz na tela gráfica`

D. Interface com o usuário

A matriz de células inicial deve ser definida diretamente na área de dados do RARS.

Ex:

```
.data
mat1:  .byte

              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
              0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0
              0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0
              0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
              0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0
              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
              0 0 0 1 1 0 0 0 0 0 1 1 1 0 0 0
              0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0
              0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

mat2:  .byte  0:255
```

Mat1 contém a matriz inicial e mat2 reserva espaço em memória para a segunda matriz.

A saída de dados se realiza plotando a matriz no Display gráfico do RARS. A cor de exibição dos pontos é arbitrária. Sugere-se utilizar a cor de fundo como ausência de indivíduo, e uma cor de fácil visualização (branco, amarelo...) para representar os indivíduos.

Sugere-se configurar o *Bitmap Display* com 8 x 8 *pixels* para cada célula da matriz e janela de 128 x 128 *pixels* para exibir a matriz completa.

Sugestão de padrões de teste (já descritos na matriz acima):

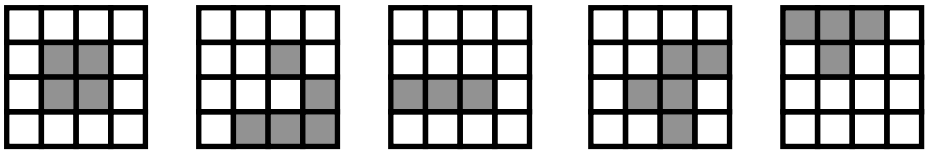

Imagem inicial:

ENTREGA

Entregar no Moodle em um arquivo compactado:

- Identificar o arquivo com o número de matrícula do aluno
- Relatório de implementação:
  - *cabeçalho*: título do trabalho, nome e matrícula do aluno, identificação da turma
  - *descrever as configurações adotadas para o RARS:* configuração de memória, endereço inicial do Bitmap Display
  - *descrição geral do programa*: pseudo-código indicando as funções chamadas
  - *descrever as funções implementadas:* nome, parâmetros e funcionamento
- código assembler: arquivo asm