Relatório de implementação Jogo da vida

Otho Teixeira Komatsu
Matricula: 17/0020142
Turma: C

April 16, 2019

1 Configurações

Para o uso de memória, tanto na janela de representação Bitmap quanto na criação da matriz, foram utilizadas espaços do tipo *word*. Tal implementação foi adotada para fins de facilitação na manipulaçao das áreas de memória, já que o RISCV não permite o acesso direto a endereços específicos de bytes, somente a endereços de *words*(4 bytes).

Na área de memória, foi alocado 1 matriz 16x16 elementos do tipo *word*(1024 bytes) para a janela bitmap(janela de preenchimento de cor e simulaçao dos seres), 2 matrizes 18x18 elementos do tipo *word* para representar as matrizes booleanas do problema(1: ser vivo, 0: vazio), sendo 16x16 interno usado para essa análise, enquanto o restante para representar a borda. Todos os elementos dessas matrizes foram inicializados com o valor 0.

Foi alocado também endereços com labels correspondentes às cores que elas representam na sua numeração hexadecimal RGB(vermelho, azul e verde, respectivamente.).

Para uma interface com usuário, foram alocadas strings com descrição da solicitação das entradas que serão requeridas para o prosseguimento do programa.

O Bitmap Display inicia seu valor no endereço correspondente a área de `static data`, ou seja, em **0x10010000**. Cada pixel tem 8 unidades de largura e altura, e seu display tem 128 de altura e largura.

2 Descrição do código

O programa inicia com uma função responsável por receber os parâmetros do usuário, e registrá-la em uma matriz. Logo depois, passa por uma subrotina responsável por analisar a vizinhança de cada pixel, e determinar seu futuro com base nos dados de sua vizinhança(como determinado pelo enunciado do *Jogo da Vida de Conway*). Após essa análise, a matriz futura resultante é repassada e atualizada à matriz que representa o estado atual do ambiente. Assim, a matriz é exibida na tela e repete-se o processo de análise de vizinhança.

Algorithm 1 Game of life

```
1: procedure INIT
2: procedure PLOTM(Matrix1)
3: loop
4:     y = 0                                          ▷ Pixel's y coordinate
5:     for y < 16 do
6:         x = 0                                      ▷ Pixel's x coordinate
7:         for x < 16 do
8:             procedure NEIGHBOUR(Matrix1, Matrix2, x ,y)
9:                 procedure READM(Matrix1, x ,y)
10:                procedure WRITEM(Matrix2, x, y)
11:            x = x + 1
12:        y = y + 1
13:    procedure COPYMATRIX(Matrix2, Matrix1)
14:    procedure PLOTM(Matrix1)
```

3 Funções implementadas

3.1 INIT

Input: -
Output: -
Parâmetros:

- `s1` - Enderenço da Matriz 2
- `s3` - Endereço da Matriz 1
- `s4` - Coordenada x
- `s5` - Coordenada y
- `s7` - Contador de pontos do usuário

Descrição: Função responsável por imprimir na tela as requisições ao usuário e pela leitura dos dados do usuário, no caso a quantidade de pontos e suas coordenadas. As coordenadas selecionadas são preenchidas com 1's(valor correspondente a ocupado por ser vivo).

3.2 PLOTM

Input:

- `a0` - Enderenço da Matriz

Output: -
Parâmetros:

- `t0` - Enderenço da cor azul
- `t1` - Valor hexadecimal do azul
- `t2` - Endereço da janela Bitmap
- `t3` - Estado do espaço
- `t4` - Iterador da Coluna
- `t5` - Iterador da Linha
- `t6` - Constante correspondente a dimensão da matriz Bitmap(16)

Descrição: Essa função recebe a matriz(18x18) dos seres vivos, a adequa às dimensões da janela bitmap(16x16), e preenche na janela bitmap as casas dos seres vivos correspondentes com a cor azul.

3.3 NEIGHBOUR

Input: -
Output: -
Parâmetros:

- `s1` - Constante correspondente a dimensão da matriz(16)
- `s2` - Endereço da Matriz 2
- `s3` - Endereço da Matriz 1
- `s4` - Iterador da Coluna
- `s5` - Iterador da Linha
- `s6` - Constante correspondente a 2
- `s7` - Constante correspondente a 3

Descrição: Analisa as proximidades do pixel correspondente às coordenadas (s4,s5) na Matriz 1. Atendendo as regras do Jogo da vida, o valor do pixel é atualizado na Matriz 2, usando as funções `readm` e `writem`. Esse procedimento é realizado em todos os pixels da Matriz 1, percorrendo as colunas de cada linha.

3.4 READM

Input:

- `a0` - Coluna
- `a1` - Linha
- `a2` - Endereço da Matriz

Output:

- `a0` - Valor lido nas coordenadas da matriz

Parâmetros:

- `t0` - Offset
- `t4` - Constante offset de uma linha(18x4)
- `t5` - Define o offset em linha(72 por linha)
- `t6` - Define o offset em colunas(4 por coluna)

Descrição: Essa função recebe as coordenadas inseridas como parâmetro e analisa a coordenada correspondente na matriz input inserida. A localização do elemento na matriz se dá pelo cálculo do offset em bytes, considerando sua distância em linhas e colunas. Após a análise, é retornado o valor analisado em a0.

3.5 WRITEM

Input:

- `a0` - Coluna
- `a1` - Linha
- `a2` - Endereço da Matriz

Output: -
Parâmetros:

- `t0` - Offset
- `t2` - Valor lido na coordenada
- `t3` - Constante 1, usada para inversão no XOR
- `t4` - Constante offset de uma linha(18x4)
- `t5` - Define o offset em linha(72 por linha)
- `t6` - Define o offset em colunas(4 por coluna)

Descrição: Essa função localiza o elemento de acordo com as coordenadas inseridas, seguindo o método descrito anteriormente, e inverte o valor armazenado no elemento. *I.e.*, insere 1 no valor armazenado no endereço quando há 0; e 0, no caso inverso(processo que se dá pela operação do XOR de 1 com o valor do elemento).

3.6 COPYMATRIX

Input: -
Output: -
Parâmetros:

- `s1` - Endereço da Matriz 1
- `s2` - Endereço da Matriz 2
- `s3` - Quantidade de elementos

Descrição: Essa função acessa a matriz 2, que determina o próximo estado da matriz do jogo, e copia seus valores de elemento em elemento na matriz 1, o que significa a atualização da matriz atual com os valores encontrados resultante do estado anterior.