Trabalho Prático Final

Esse repositório contém o TPF desenvolvido na disciplina de AOC1 no semestre 2019-1 pela dupla Frederico Bueno Da Silva Schaun (M1) e Cesar Augusto Vitoria Martins Junior (M1), que tem como objetivo implementar o jogo da vida de John Conway.

O jogo é um automato celular definido por um campo que pode conter uma quantidade qualquer de células. Essas células podem assumir dois estados: Viva/Populada ou Morta/Despopulada.

Sabendo disso, o jogo considera quatro regras básicas:

- Para qualquer celula viva
  - Se tem menos de dois vizinhos vivos morre (de solidão).
  - Se mais de três vizinhos vivos morre (de superpopulação).
  - Se tem dois ou três vizinhos vivos continua no mesmo estado para a próxima geração.
- Para qualquer celula morta
  - Se tem exatamente três vizinhos vivos torna-se uma célula viva.

Em pseudo-código as regras são equivalentes à algo como:

```
if (celula está morta)
  if(celula tem 3 vizinhos) estado = viva

if (celula está viva)
   if(celula tem 2 ou 3 vizinhos) estado = vida
   else estado = morta
```

Configuração

Primeiramente, para o jogo funcionar corretamente devem estar habilitadas as seguintes configurações no *MARS* (e as demais desabilitadas): `Settings > Assemble all files in directory`, `Settings > Initialize Program Counter to global 'main' if defined` e `Settings > Permit extended (pseudo) instructions and formats`. Além disso, o campo é representado pela ferramenta `BitmapDisplay` do *MARS* e, portanto, deve estar habilitada também em `Tools > Bitmap Display` e devidamente configurada de acordo com os campos definidos na parte de dados do [`Main.asm`][] (Certifique-se também que a ferramenta está conectada ao Mars).

Implementação

Na nossa implementação cada célula é representada por um pixel, sendo o seu estado uma cor (Preto = Morta, Com cor = Viva) e o campo que contém as células é uma região continua na memória. Para poder guardar todas as essas informações para o funcionamento adequado do jogo foi definida uma estrutura que é guardada na memória que contém dados como o endereço de dois campos (o atual e o anterior), o tamanho do `BitmapDisplay` e de cada pixel nele, para haver adaptação à diferentes tamanhos de tela, entre outros. Assim, bastou passar o endereço do primeiro elemento dessa estrutura para às funções para estabelecer uma comunicação entre elas.

A organização do código é feita da seguinte forma: Cada função é definida no seu próprio arquivo e exporta somente o endereço para fazer a chamada dela, através

da diretiva `globl` .

O jogo começa por um menu simples que tem três opções:

- 1. Jogar(Inicializa o jogo direto).
- 2. Regras do jogo(Permite ao jogador visualizar as quatro regras existentes no Jogo da Vida de Conway).
- 3. Sair(Finaliza a execução).

Logo após é incializado um campo com células em estados aleatórios (usando `syscall` para gerar um estado aleatório para cada uma) e depois entra em um loop, onde são feitas as seguintes operações:

- O campo anterior é sobrescrito com o campo atual.
- O campo atual é apagado (para não haver pixels residuais no `BitmapDisplay`)
- São aplicadas as regras sobre o campo anterior, escrevendo os resultados na posição de memória do campo atual.
- Se o campo gerado for completamente morto a execução é terminada.

A implementação das regras é feita atravéz de uma serie de testes dentro do arquivo "makenew frame", com os nove testes necessários para aplicar as regras do jogo no bitmap, que são:

- Testes para todos os cantos do bitmap, permitindo que possamos realizar os testes com os três vizinhos possíveis para cada um dos quatro cantos, o teste consiste em "teste1" ao "teste4" e "return11" ao "return14" comparando o pixel do canto com todos os três vizinhos e aplicando as regras.
- Testes para todas as paredes do bitmap sem contar os cantos, permitindo que possamos realizar os testes com os cinco vizinhos possíveis para cada pixel das bordas, o teste consiste em "teste5" ao "teste8" e "return51" ao "return56" comparando o pixel das laterais com todos os cinco vizinhos e aplicando as regras.
- Testes para todos pixels do meio(que contem nove vizinhos), o teste consiste em "teste9" e "return91" ao "return99" comparando os pixels do meio e aplicando as regras.
- Todos os testes utilizam o "pegarpixel" que permite calcularmos o endereço do pixel

---

Execução

Para executar o trabalho basta carregar qualquer arquivo de código do projeto no simulador *MARS* configurado conforme especificado no tópico *Configuração*, fazer *assemble* e rodar o programa.