Trabalho 1 - OAC

Felipe Lima Vaz - 17/0057852
Turma: C

1. Configurações do RARS

A configuração da memória é a compacta com o text adress iniciando no zero como foi recomendado em sala de aula.

Para configurar o Bitmap Display a unidade de largura e altura deve ser 8, o display da largura e da altura nos pixels deve ser 128 e o endereço base para o display deve ser o heap.

É importante deixar claro que a matriz de bytes 1 contém a geração atual de células e a matriz de bytes 2 contém a geração seguinte.

2. Descrição geral do programa

Para o usuário inserir suas entradas, primeiro insere o número de entradas e em seguida insere a linha da coordenada, aperta enter, insere a coluna da coordenada e aperta enter. O programa roda em loop infinito o jogo da vida. Não há condição de parada

1. Lê o endereço das duas matrizes de bytes.
2. Lê o endereço do display.
3. Salva a cor dos pixels “vivos” e “mortos”.
4. Chama a função input para criar a matriz que o usuário deseja.
5. While(1)
    a. Chama a função que cria a próxima geração de pixels na segunda matriz.
    b. Chama a função que copia a segunda matriz de bytes para a primeira.
    c. Chama uma função que atualiza o display com a nova geração
    d. Espera 300 ms antes de repetir o loop
6. End

3. Descrição das funções implementadas

a. input

Esta função não tem nenhum parâmetro de entrada visto que todas as suas variáveis são obtidas de endereços salvos na memória e, por ser simples, ela não retorna nenhum valor na saída pois ela simplesmente altera o conteúdo diretamente na memória.

Basicamente, esta função lê o número de entradas do usuário, lê as coordenadas dos pixels que serão atualizados e chama a função write para atualizar o valor dessas coordenadas.

1. Lê o número de entradas
2. While(numero de entradas > 0)
    a. Lê a coordenada da linha
    b. Lê a coordenada da coluna
    c. Chama a função write para atualizar o valor na matriz de bytes
    d. Chama a função plot_m para atualizar o display
    e. número de entradas--
3. Retorna

b. write

Suas entradas são a cópia da linha em a0, a cópia da coluna em a1 e a cópia do endereço da matriz de bytes em a2. Essa função não possui retorno.

Essa função percorre a matriz de bytes pixel a pixel até chegar nas coordenadas fornecidas em a0 e a1 e ao chegar nelas, faz uma operação XOR para inverter o valor do bit que estava armazenado.

1. While(numero da linha > 0)
    a. While(numero da coluna > 0)
        i. número da coluna--
        ii. avança ponteiro da matriz de bytes
    b. número da linha--
    c. número da coluna = 16
2. Inverte o pixel das coordenadas finais
3. Retorna

c. readm

Suas entradas são a cópia da linha em a0, a cópia da coluna em a1 e a cópia do endereço da matriz de bytes em a2. Seu retorno está na a3 e consiste em 1 se nas coordenadas existe o valor 1 na matriz de bytes e 0 se as coordenadas forem inválidas ou se existe o valor 0 nas coordenadas.

Primeiramente, verifica-se se as coordenadas são válidas e, caso sejam, percorre a matriz, byte a byte, até encontrar as coordenadas desejadas. Ao chegar no ponto desejado, salva em a3 o valor que foi encontrado lá.

1. If(0 < número da linha < 16 && 0 < número da linha < 16)
    a. While(numero de linhas > 0)
        i. While(numero de colunas > 0)
            1. número de colunas--
            2. avança ponteiro da matriz de bytes
        ii. número de linhas--
        iii. número de colunas = 16
    b. Retorna o valor do byte nas coordenadas (0 ou 1)
2. Else
    a. Retorna 0 (coordenada inválida)

d. plot_m

Essa função recebe como entrada apenas a cópia do endereço da matriz de bytes como entrada e não tem nada como saída pois ela apenas atualiza o display.

Sua utilidade é percorrer a matriz de bytes, pixel a pixel, e ir atualizando o display com os valores encontrados onde ele colore pixels cujo byte da matriz for “1” e descolore se for “0”. Ela repete este procedimento em todos os pixels da matriz.

1. While(numero de linhas > 0)
    a. While(numero de colunas > 0)
        i. Verifica o valor na matriz de bytes
        ii. If(Valor na matriz = 1)
            1. Colore o pixel
        iii. Else If(Valor na matriz = 0)
            1. Descolore o pixel
        iv. avança ponteiro da matriz de bytes
        v. avança ponteiro do display
        vi. número de colunas--
    b. número de linhas--
    c. número de colunas = 16
2. Retorna

e. next_gen

Essa função não recebe nada como entrada e não retorna nada na saída. Ela é praticamente a função principal do programa visto que irá criar na matriz de bytes 2 como será a próxima geração de células.

Seu funcionamento ocorre de forma similar às outras funções descritas anteriormente, onde percorremos pixel a pixel a matriz de bytes. Em seguida, chama-se a função live_or_die para saber se o pixel em questão irá viver ou morrer na próxima geração. Os valores das células na próxima geração serão salvos na matriz de bytes 2.

1. número de linhas = 1
2. número de colunas = 1
3. While(numero de linhas < 16)
    a. While(numero de colunas < 16)
        i. Chama a função live_or_die para determinar se a célula vive ou morre
        ii. Troca o valor das coordenadas atuais da matriz de bytes 2 pelo retorno da função live_or_die
        iii. avança ponteiro da matriz de bytes 1
        iv. avança ponteiro da matriz de bytes 2
        v. número de colunas++
    b. número de linhas++
    c. número de colunas = 1
4. Retorna

f. live_or_die

Recebe 4 parâmetros como entrada, sendo eles a5 o número da linha, a6 o número da coluna e a4 a cópia da matriz de byte1. Sua saída é armazenada na variável a7 e determina se a célula cuja as coordenadas está em a5 e a6 vive ou morre na próxima geração (0 = morre, 1 = vive/nasce).

Sua implementação foi feita de modo que chamamos a função readm em cada um dos 8 vizinhos de uma célula e incrementamos um contador sempre que um desses vizinhos estiver vivo, ou seja, têm o valor “1” armazenado nas suas coordenadas na matriz de bytes 1. Por fim, verifica-se se a célula tem 2 ou 3 vizinhos para que neste caso ela continue viva e, caso tenha 3 vizinhos e não tenha célula na posição informada, na próxima geração, uma nova célula nascerá nessas coordenadas e a7 será “1”.

1. Número de vivos é zero
2. Chama a função readm no vizinho superior esquerdo das coord. fornecidas
3. If(vizinho = 1)
    a. Número de vivos++
4. Chama a função readm no vizinho superior central das coord. fornecidas
5. If(vizinho = 1)

    a. Número de vivos++
6. Chama a função readm no vizinho superior direito das coord. fornecidas
7. If(vizinho = 1)
    a. Número de vivos++

8. Chama a função readm no vizinho esquerdo das coord. fornecidas
9. If(vizinho = 1)
    a. Número de vivos++
10. Chama a função readm no vizinho direito das coord. fornecidas
11. If(vizinho = 1)
    a. Número de vivos++
12. Chama a função readm no vizinho inferior esquerdo das coord. fornecidas
13. If(vizinho = 1)
    a. Número de vivos++
14. Chama a função readm no vizinho inferior central das coord. fornecidas
15. If(vizinho = 1)
    a. Número de vivos++
16. Chama a função readm no vizinho inferior direito das coord. fornecidas
17. If(vizinho = 1)
    a. Número de vivos++
18. If(Número de vivos = 3)
    a. Célula nas coordenadas atuais é viva (‘1’) na próxima geração
19. Else if(Numero de vivos = 2)
    a. If(Célula está viva na geração atual)
        i. Célula nas coordenadas atuais é morta (‘0’) na próxima geração
    b. Else if(Célula está morta na geração atual)
        i. Célula nas coordenadas atuais nasce (‘1’) na próxima geração
20. Else
    a. Célula nas coordenadas atuais é morta (‘0’) na próxima geração
21. Retorna

g. update_m

Recebe como parâmetros os endereços de memória da primeira e segunda matrizes de bytes em a0 e a1 respectivamente e não retorna nada.

Essa função é simple e tudo que ela faz é copiar o conteúdo da matriz 2 para a matriz 1.

1. While(numero de linhas > 0)
    a. While(numero de colunas > 0)
        i. Copia o valor da matriz de bytes 1 para a 2

        ii. Avança o ponteiro da matriz de bytes 1
        iii. Avança o ponteiro da matriz de bytes 2
        iv. número de colunas--
    b. número de linhas--
    c. número de colunas = 16
2. Retorna