TÉCNICO LISBOA

Arquitetura de Computadores

Programação Assembly
Mastermind

Professor Nuno Horta

Professor Paulo Lopes

Turno de Sexta-Feira 18h30

António Rouxinol Fragoso nº 79116

Diogo Moura nº 86976

5 de Maio de 2017

Objetivos

O objetivo deste trabalho de laboratório é implementar uma versão do jogo Mastermind. Para isso, desenvolvemos um programa em linguagem Assembly, direcionada para o processador P3.

O jogo implementado consiste em escolher quatro cores, iguais ou diferentes e com posições definidas e, por tentativa e erro com feedback chegar à solução. Ou seja, para uma solução predefinida, são feitas várias tentativas para obter a solução correta, em que cada uma dessas tentativas será avaliada de modo a que se chegue mais rapidamente à solução final.

Estruturas de Dados utilizadas

São utilizadas "strings" não só para representar o tabuleiro, através de uma tabela, como também para guardar as estatísticas do utilizadores nos vetores correspondentes "jogador1", "jogador 2", "jogador 3". A "seed" e o tempo de jogo são representadas por variáveis,isto é, em linguagem Assembly "WORD".

Exemplos das estruturas de dados utilizadas:

```
VarTexto4      STR   '|      Tentativa1   |                |                  |', FIM_TEXTO
randNumb       STR    0000h, 0000h, 0000h, 0000h,0000h,0000h, FIM_TEXTO
randNumbcopia  STR    0000h, 0000h, 00000h, 0000h,0000h,0000h, FIM_TEXTO
userinput      STR    0000h, 0000h, 0000h, 0000h,0000h,0000h, FIM_TEXTO
validacao_output    STR    ' ',' ',' ',' ',' ',' ', FIM_TEXTO
jogador1 STR 0000h, 0000h      ; posição1->tempo de jogo, posição2->tentativas
jogador2 STR 0000h, 0000h
jogador3 STR 0000h, 0000h
totaltentativas WORD 0000h
totaljogadores WORD 0000h
```

Inicialização da Seed

É contado o tempo que o utilizador demora, em intervalos de 100ms a clicar na tecla "s", para iniciar o jogo e é guardado no endereço de memória "seed".

O valor da seed é representado por M [seed].

Avaliar Jogada

Após ser verificado que as cores escolhidas são válidas, é agora necessário proceder à avaliação da jogada. Para isso, foi desenvolvido um algoritmo que compara a tentativa do jogador com a sequência gerada. Abaixo é feito um exemplo do seu funcionamento. Inicialmente, as cores que estão na mesma coluna são comparadas.

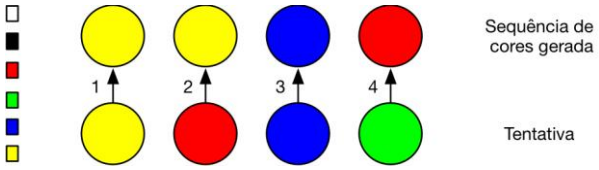

Figura 1 – Primeiro passo exemplificado

Caso sejam iguais, essas cores são eliminadas e é escrito, na validação, p (cor certa e posição certa).

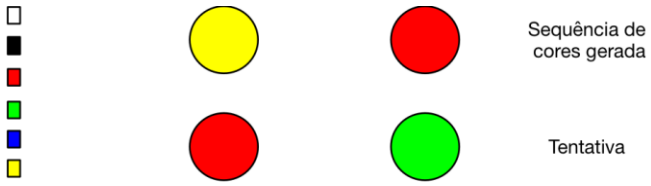

Figura 2 – Segundo passo exemplificado

Se ainda houver cores por eliminar, verifica-se se alguma delas está na posição errada. Para isso, comparamos a cor mais à esquerda da tentativa do utilizador com as cores da solução da esquerda para a direita.

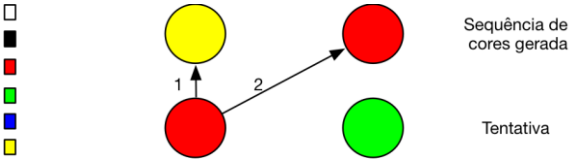

Figura 3 – Terceiro passo exemplificado

Caso haja coincidência de cor, é avaliada como B(cor certa na posição errada) e é eliminada essa cor , isto é , é substituída por uma cor inválida.

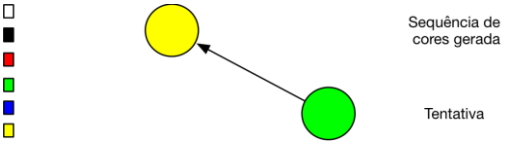

Figura 4 – Quarto passo exemplificado

Passa-se depois para a próxima cor à direita, caso exista. No final é feita a avaliação final da tentativa.

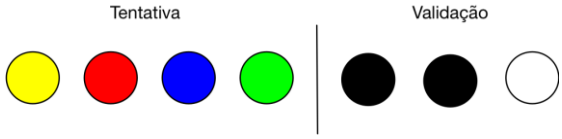

Figura 5 – Quinto passo exemplificado

Nível de dificuldade

O nível de dificuldade é escolhido no início do jogo,através da rotina MenuOpcoes, havendo 3 níveis disponíveis. Esse nível é definido pelo número de peças que vão ser usadas. Assim, o jogo é mais difícil, quando o número de espaços que o utilizador tem de preencher for maior. Se for escolhido o nível mais

fácil, são usados 4 espaços que o jogador deve preencher com as cores disponíveis. Se o nível escolhido for o médio, é adicionado mais um espaço. Caso seja selecionado o modo mais difícil, o utilizador deve preencher seis espaços com as cores ao seu dispor.

RandNumb

A solução gerada para cada jogo depende do nível de dificuldade escolhido no início do jogo. Quanto maior for o nível de dificuldade maior o número de randnumbs guardados neste vetor, sendo que no máximo são guardados 6.

É utilizada a seed e como número inicial NI e é realizado o seguinte algoritmo:

```
Mascara = 1001 1100 0001 0110b

    if (Ni0==0) /* Testa o bit de menor peso de Ni */
            Ni+1 = rotate_right(Ni);
    else
            Ni+1 = rotate_right (XOR (Ni, Mascara))
```

Algoritmo 1 – Geração de um número pseudoaleatório de 16 bits.

Número de jogadores

O número de jogares é definido no início do jogo, através da função MenuOpcoes. O utilizador escolhe se quer 1, 2 ou 3 jogadores e é guardado na posição de memória “numbjogadores” o valor em “M[numbjogdaores]”

Estatísticas do Jogo

Cada vez que um jogador termina a sua jogada ,é chamada a função GuardaStats. Esta função guarda o tempo que o utilizador demorou a jogar e o número de tentativas no vetor correspondente ao jogador, o qual é passado através da “stack” para a rotina.

Posteriormente é calculado o número médio das tentativas e é realizada a escrita no LCD através da função “EscLCD”.

A função EscLCD realiza a escrita para o LCD através dos portos:

```
LCD_WRITE       EQU     FFF5h
LCD_CURSOR      EQU     FFF4h
```