INSTITUTO SUPERIOR TÉCNICO

LICENCIATURA EM ENGENHARIA INFORMÁTICA E DE COMPUTADORES

PROJETO DE INTRODUÇÃO À ARQUITETURA DE COMPUTADORES (PARTE 1)

MASTER-MIND

PARTE I

2017 / 2018

ÍNDICE

Objetivo 3

Descrição do Jogo 3

Exemplo da execução do jogo 3

Implementação 3

Saída (output) do programa 4

Valores aleatórios 4

Dúvidas na especificação do enunciado 5

Plano de Desenvolvimento 5

Faseamento da codificação 5

Qualidade do código 5

Data de Entrega 5

Dia 29 de Outubro, até às 23h59 5

Anexo A – Geração de sequência pseudoaleatória 6

OBJETIVO

O projeto consiste no desenvolvimento de uma versão simples do jogo de tabuleiro MasterMind. Nesta versão o objetivo é descobrir uma combinação de dígitos secreta gerada internamente pelo programa. O jogo decorre numa janela de texto onde vão sendo mostradas as jogadas e o resultado.

Neste documento são descritos os detalhes de funcionamento pretendidos para o jogo. O jogo será programado usando a linguagem Assembly para o processador P3. O desenvolvimento e teste do programa serão realizados usando o simulador do P3 (p3sim).

DESCRIÇÃO DO JOGO

O objetivo deste jogo é identificar uma sequência de 4 dígitos (entre 1 e 6) desconhecida do jogador (Descodificador - D) criada no jogo real por outro jogador, ou, no caso deste projeto, automaticamente pelo programa (Codificador - C).

O jogo decore num tabuleiro onde o jogador D coloca, em tentativas sucessivas, determinados dígitos pela ordem que ele considera serem a escolhida pelo jogador C.

Após cada tentativa, o programa dá indicação ao jogador D se os dígitos por ele escolhidos fazem parte do conjunto secreto e/ou se encontram na posição correta. Esta informação é indicada usando dois símbolos: 'x' que corresponde a dígito existente na sequência secreta na posição certa, e 'o' que corresponde a dígito existente, mas na posição errada.

O jogo termina quando o jogador D descobre a sequência secreta ou ao fim de 12 tentativas sem sucesso.

EXEMPLO DA EXECUÇÃO DO JOGO

Sequência secreta: 1-1-4-4

Jogada	Tentativa	Resultado
1	2-3-1-4	xo--
2	2-4-5-2	o---
3	6-6-3-5	----
4	1-3-5-6	x---
5	6-1-2-4	xx--
6	1-1-2-4	xxx-
7	1-1-4-4	xxxx

Mais detalhes sobre o jogo original e variantes podem ser encontrados aqui:
https://en.wikipedia.org/wiki/Mastermind_(board_game)

IMPLEMENTAÇÃO

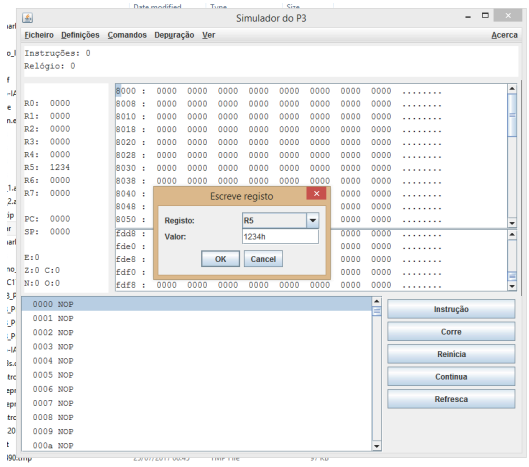
A lógica principal do jogo a desenvolver consiste na leitura de um registo (R1) com a sequência de dígitos secreta e comparação com o conteúdo de outro registo (R2) que contém a jogada atual do jogador D.

O programa começa por gerar uma sequência de 4 dígitos aleatória[i] e coloca esta sequência no registo com a sequência secreta (R1). De seguida inicia-se o ciclo do jogo, onde recebe a jogada do jogador no registo R2 e avalia a semelhança com a sequência secreta. Esta semelhança é colocada no registo R3. Tem a liberdade de escolher a codificação para o conteúdo destes registos, com a restrição que uma sequência (correspondente a uma jogada) terá de ser representada num registo

(isto é, por no máximo 16 bits). Esta lógica deverá ser repetida até atingir o limite de jogadas ou a sequência secreta ser descoberta.

Para tornar o desenvolvimento mais incremental, num primeiro passo do desenvolvimento do código, sugerimos começar por considerar que as 12 jogadas são escolhidas de uma forma arbitrária, sem qualquer tipo de raciocínio com base nos resultados sucessivos da semelhança. (Por exemplo, pode programar cada jogada para ser um simples incremento dos valores da jogada anterior.)

Na versão final desta parte do projeto, para poder jogar o jogo de forma interativa, deve alterar o programa anterior de forma a que o jogador introduza a sua jogada num registo do P3 pelo menu "Depuração → Escreve registo", tal como na figura abaixo. Neste caso, o programa deve colocar o registo R2 a zero após ler uma nova jogada, e ter um ciclo que testa se R2 mudou de zero para um valor diferente de zero, e desta forma detetar quando o registo foi escrito.



SAÍDA (OUTPUT) DO PROGRAMA

A saída do programa é através da janela de texto, transferindo para tal os caracteres ASCII a serem escritos no écran para o endereço de memória FFFEh. Para visualizar a saída do programa na janela de texto deve carregar em "Ver → Janela Texto" no simulador do P3. O seguinte código exemplifica a escrita de 4 caracteres no écran, sendo um deles o caracter 0A que faz uma mudança de linha.

```
IO    EQU    FFFEh
NL    EQU    000Ah
      ORIG   0000h
MOV   R1, 'x'
MOV   M[IO], R1
MOV   R1, 'y'
MOV   M[IO], R1
MOV   R1, NL
MOV   M[IO], R1
MOV   R1, 'z'
MOV   M[IO], R1
```

VALORES ALEATÓRIOS

Quando inicia um novo jogo, é necessário obter um valor aleatório para determinar a nova sequência de dígitos. O Anexo A descreve um algoritmo para a geração de valores aleatórios de 16 bits.

DÚVIDAS NA ESPECIFICAÇÃO DO ENUNCIADO

Sempre que o enunciado não defina completamente como deve implementar alguma parte da solução, deve tomar a decisão que achar mais apropriada, tendo sempre em mente o objetivo de melhorar a experiência de jogo. Estas decisões devem ser justificadas no relatório e na discussão (ambos irão decorrer após a segunda parte, no final do semestre).

PLANO DE DESENVOLVIMENTO

FASEAMENTO DA CODIFICAÇÃO

Não deve tentar codificar todo o programa de uma só vez. Implemente as várias funcionalidades do programa de uma forma faseada e efetue os testes necessários para verificar o seu correto funcionamento. Não prossiga para a implementação de funcionalidades mais avançadas sem ter garantido que as que lhe servem de base estão corretamente implementadas.

QUALIDADE DO CÓDIGO

Na avaliação será tida particularmente em conta a qualidade do código produzido. Em particular, deverá ter o cuidado de comentar o código de forma a facilitar a sua compreensão. Para além disso, deverá tornar o código modular através do uso de funções, em que os parâmetros de entrada e valor de retorno dessas funções são passados pela pilha.

DATA DE ENTREGA

DIA 29 DE OUTUBRO, ATÉ ÀS 23H59

Entrega do código do projeto via fénix. As instruções de entrega serão anunciadas no fénix.

ANEXO A – GERAÇÃO DE SEQUÊNCIA PSEUDOALEATÓRIA

O seguinte algoritmo gera uma sequência aparentemente aleatória de números de 16 bits, com distribuição uniforme (isto é, os números são equiprováveis), com um passo de repetição elevado:

```
Máscara = 1000 0000 0001 0110 b
if(n0==0) /*Testa o bit de menor peso*/
    Ni+1 = rotate_right (Ni);
else
    Ni+1 = rotate_right (XOR (Ni, Mascara));
```

Em cada invocação desta função lê-se o valor anterior Ni e gera-se um novo valor pseudoaleatório, Ni+1. Para obter um valor aleatório entre 0 e M pode dividir o valor de Ni por M e retirar o resto, mas sempre sem modificar o valor de Ni.

*** Fim ***

[i] Para iniciar o desenvolvimento pode assumir uma sequência conhecida e constante, e mais tarde usar a sequência aleatória.