LEIC-A 2017/2018
Introdução à Arquitetura de Computadores
Bernardo Cunha Matos - 89419
João Carlos Morgado David - 89471

Relatório do Projeto

Mastermind

Introdução

O Projeto consiste em replicar o jogo Mastermind no processador P3. As regras do jogo nesta implementação consistem em:

- É gerado um código secreto aleatoriamente com algarismos entre 1 e 6;
- O jogador tem 12 tentativas para acertar a sequência;
- Se demorar mais de 8 segundos a realizar uma jogada, o jogador perde automaticamente o jogo.
- Por cada número na posição correta aparece um 'x' no ecrã, um 'o' para cada número correto na posição errada, e '-' para os restantes falhados. O resultado não deverá indicar a posição da sequência onde se acertou/falhou.

Estrutura do jogo

O programa é jogado com os 6 botões de pressão da placa do P3, I1 a I6, para escolher os dígitos da sequência. Também é utilizado o botão IA para começar e iniciar o jogo. O jogo joga como se esperaria

Ao terminar o jogo, são mantidos os valores do tempo restante da jogada nos LEDs tal como o turno em que se perdeu ou ganhou, no display de 7 segmentos. No LCD é apresentada a pontuação máxima (quanto menor, melhor).

Implementação geral

A implementação é feita utilizando 4 variáveis principais: escolha, codigo, resultado. Também são utilizadas outras variáveis como flags, como o comecou, que indica se foi premido o botão IA. O pode_jogar indica se se está à espera do Input do jogador. A utilização dos periféricos segue o enunciado. As rotinas não são demasiado complexas, logo, e na nossa opinião não necessitam de ser abordadas em relatório, até porque consideramos que os comentários explicam o código bastante bem, no entanto, há uns aspectos que deveriam ser referidos.

No fim do relatório está um fluxograma que explica a lógica geral do programa, o que não está no fluxograma resume-se a cálculos e à utilização dos periféricos.

Bernardo Cunha Matos - 89419
João Carlos Morgado David - 89471

>Pushing e Popping:

Para preservar registos entre rotinas, podemos recorrer a um *push*, no início da rotina, aos registos que sofrerão alterações, e um *pop* para os mesmos registos no fim da rotina. No entanto, nem sempre é necessário este ser realizado este processo.

No nosso projeto, preservamos os registos principalmente em rotinas "auxiliares". Não há razão para preservar registos entre os módulos principais, uma vez que são sempre reescritos para satisfazer as necessidades do módulo. No entanto, quando precisamos de chamar uma função auxiliar (de algum cálculo, por exemplo) dentro dum outro módulo maior, faz todo o sentido preservar os registos, para que não se estrague a lógica do módulo principal, mas é desnecessário recorrer tantas vezes à memória quando sabemos que iremos passar para um módulo completamente diferente.

>Flag - pode_jogar:

Inicialmente, esta varíavel servia apenas para indicar que o jogador podia escrever os digitos. Na versão final, utilizamos a flag para mais outros casos. Quando pode_jogar = 0, o programa executa o resto dos procedimentos, por exemplo, compara as duas sequências. Desta forma, utilizámos também a flag para parar temporariamente o temporizador, evitando descontar tempo injustamente ao jogador. Ao verificar se pode_jogar é 0 no ciclo verificar_input, conseguiu-se simplificar o método anterior, que esperava que a sequência ocupa-se 15 bits para seguir para jogada, fazendo um SHR no início dessa etiqueta. Na nova implementação, metemos pode_jogar a 1 quando a escolha ocupa 12 bits, evitando assim fazer o último SHL, nunca ultrapassando o limite de 12 bits.

Conclusão

Consideramos que o projeto teve um balanço positivo, tanto pela nossa realização. Conseguiu-se implementar todas as funcionalidades pedidas no enunciado, com apenas ligeiras adaptações a partes menos explícitas. Divergiu-se ligeiramente na apresentação da pontuação atual. Em vez de o display de segmentos apresentar o número de jogadas realizadas (começando em 0), apresenta o número da jogada atual (começando em 1). O resultado é o final é o mesmo, uma vez que só atualiza o número da jogada quando não se acerta, ou seja, se se acerta no turno 6, o valor permanece a 6.

Como funcionalidades adicionais, permitimos ao jogador que recomece o jogo a qualquer momento, não tendo de esperar pelo fim do jogo. Optamos assim porque entendemos que criava uma melhor experiência ao jogador. Também decidimos preservar o valor da jogada e do tempo restante no fim do jogo, e apenas apagá-las quando se recomeça. Assim, ao terminar um jogo, é possível observar na placa quanto tempo restava na jogada e o turno em que se terminou.

Concluindo, consideramos que realizamos um bom projeto, com código bem estruturado, funcional e bem comentado e achamos que o nível de exigência foi apropriado aos nossos conhecimentos, de maneira que não houve nenhum aspecto demasiado complicado de implementar sem deixar de ser uma experiência e um desafio enriquecedor.

LEIC-A 2017/2018
Introdução à Arquitetura de Computadores
Bernardo Cunha Matos - 89419
João Carlos Morgado David - 89471