Introdução à Arquitetura de Computadores

Relatório do Projeto, Grupo 29

MasterMind

André Silva, nº 89408

Rafael Henriques, nº 89503

O projeto ***MasterMind*** no âmbito da cadeira de Introdução à Arquitetura de Computadores consiste num jogo em *Assembly* do processador P3. Para tal, criámos um conjunto de rotinas para interagir com os periféricos, fazer os cálculos necessários à execução do jogo e uma rotina principal. Esta abordagem permitiu-nos modularizar e fasear o desenvolvimento do projeto facilitando, portanto, a integração das várias funcionalidades no programa principal.

Organização do Programa

O programa encontra-se estruturado em 3 partes: a **rotina principal, as rotinas de periféricos e cálculos, rotinas interrupções.**

A **rotina principal** corresponde à chamada de rotinas por uma ordem correspondente ao funcionamento do jogo. Esta rotina pode ser acompanhada através do fluxograma presente no anexo.

As **rotinas de periféricos e cálculos** utilizadas no decorrer do jogo são as seguintes:

- **clrscr**: Esta rotina imprime o caracter ‘ ‘ 1896 vezes, correspondentes às 24 por 79 posições (não utilizamos a coluna no 80 dadas as restrições do P3, de maneira a simplificar o uso da janela de texto) utilizadas da janela de texto. O cursor inicia em 0000h e é incrementado até ao fim dessa linha, altura em que o valor da coluna é reposto a 0 e o valor da linha é incrementado.
- **print_texto_LCD**: Esta rotina tem como função inicializar o *LCD*, ou seja, colocar o bit mais significativo a 1, e imprimir a string “Highscore:” seguida de dois “-“ nas posições seguintes, onde posteriormente será imprimido uma pontuação máxima
- **print_str**: Esta rotina é utilizada ao longo de todo o programa. Recebe como argumentos a posição inicial do cursor e a primeira posição de memória da *string* a imprimir. A *string* é percorrida caracter a caracter, sendo imprimido cada um na posição seguinte à do anterior, até ser encontrado o caracter “@”. É utilizado o mesmo mecanismo aplicado na rotina *clrscr* para controlar o cursor.
- **random**: Esta rotina recebe a semente, a partir da qual é gerado um código secreto, onde cada dígito de 1 a 6 é codificado em 3 bits, o que corresponde a 12 bits no final. Devolve o código gerado e o estado final da semente, a ser utilizada no início do próximo jogo.
- **input_jogada**: Esta rotina devolve a jogada introduzida pelo jogador. Corresponde a um ciclo que, após serem inicializados o temporizador e os *LEDS*, são verificadas as *flags* dos interruptores (tanto do temporizador como dos botões da jogada). Quando a jogada for introduzida o ciclo é interrompido e a jogada (codificada em 12 bits da mesma maneira que o código) é devolvida. Se o tempo limite for excedido é retornado o valor 0000h, valor que representa uma jogada inválida a ser reconhecida no corpo principal.

- **verifica**: Esta rotina recebe o código e a jogada. São chamadas as rotinas verifica_x, que devolve os argumentos com o valor 0 nos algarismos que tiveram correspondência e a semelhança (codificada em 8 bits, 00 = '-' 01 = 'x' 10 = 'o'), e verifica_o, que devolve o resultado final da comparação. É utilizado o mesmo mecanismo que na rotina anterior, sendo colocado o valor 0 nos algarismos com correspondência, para evitar serem encontradas semelhanças repetidas.
- **print_resultado**: Esta rotina imprime a jogada introduzida pelo jogador e o resultado da comparação. Este resultado é primeiramente colocado em posição e depois imprimido de acordo com a codificação.
- **atualiza_LCD**: Esta rotina recebe o valor a imprimir no *LCD*, e imprime a sua correspondência em base decimal nas duas posições do *LCD* correspondentes. Para tal é necessário primeiro separar as dezenas das unidades, sendo depois os algarismos convertidos para *ASCII*.
- **sete_seg**: Esta rotina recebe o valor a imprimir no *display* de sete segmentos, e imprime a sua correspondência em base decimal. É utilizado o mesmo mecanismo para separar as dezenas das unidades que foi utilizado na rotina anterior, e depois são colocados os valores nas posições correspondentes.

As **rotinas de interrupção** correspondentes aos **interruptores 1 a 6**, colocam em R1 o valor correspondente ao da sua interrupção, registo este que é utilizado na única rotina que utiliza estas interrupções. As rotinas correspondentes às **interrupções A e B** incrementam e decrementam, respetivamente, a *flag* utilizada no corpo principal. A rotina do temporizador altera o valor da *flag* e recomeça a contagem caso o tempo limite ainda não tenha sido excedido.

Estruturas de dados relevantes:
De maneira a armazenar o código secreto, a jogada introduzida e o resultado optámos por utilizar uma codificação *BCD*. No código secreto e a jogada, uma vez que podem conter algarismos de 1 a 6 apenas, cada algarismo pode ser representado por 3 bits, resultando num total de 12 bits, facilmente descodificáveis.
No resultado da comparação entre estes dois, decidimos implementar uma codificação onde cada caracter é representado por 2 bits, uma vez que existem apenas três caracteres possíveis ('x','o' ou '-'). Usámos, portanto, apenas 8 bits na codificação desta comparação, onde 01 representa 'x', 10 representa 'o' e 00 representa '-'.

Funcionalidades Adicionais:
Após a introdução de uma jogada, é imprimido não só o resultado da comparação, bem como a jogada introduzida, de maneira a facilitar a memória do jogador. Após cada jogo é também dada a indicação de vitória ou derrota. Foi também implementado um ecrã final, caso o jogador não pretenda continuar a jogar, que contem os créditos.

Principais Dificuldades:
No decorrer da realização do projeto as principais dificuldades com que nos deparámos foi na procura de *bugs*, uma vez que por vezes demorámos algum tempo a encontrar pequenos erros elementares (e.g. RETN 3 em vez de RETN 2, fez o nosso projeto deixar de funcionar corretamente ao fim da 2ª jogada)

Conclusão:
O projeto realizado segue todas as especificações descritas no enunciado. Atribuímos, portanto, um balanço positivo ao resultado final do projeto.