Afonso Ribeiro, 89400

Margarida Ferro, 86375

O projeto de IAC consistia na construção de um jogo, o Mastermind, na linguagem de programação assembly para o processador P3, com uma série de instruções, restrições e funcionalidades acrescidas.

De um modo geral, o nosso projeto está organizado da seguinte forma:

1. Definem-se constantes e reservam-se espaços na memória para algumas variáveis que vão ser utilizadas ao longo do programa;
2. Definem-se as interrupções associadas aos botões IA, 1,2,3,4,5,6 e ao Temporizador;
3. Escreve-se na janela de texto “Carregue no botão IA para iniciar”;
4. Incrementa-se uma variável (Até se carregar em IA) que vai estar associada ao numero aleatório gerado;
5. A interrupção associada ao botão IA limpa a janela de texto e faz com que o programa inicie um novo jogo;
6. Inicia-se um novo jogo: Chama-se a rotina que gera um numero pseudo-aleatorio e dá-se reset às variáveis em memória de jogos anteriores;
7. Inicia-se uma nova jogada: Limpa-se o resultado e o numero do utilizador da jogada anterior, preenchem-se os 12 bits do resultado com “traços” e ativam-se o temporizador e os LEDs (Para contar o tempo da jogada);
8. O programa fica em Loop à espera que o utilizador introduza a sua jogada ou que termine o tempo;
9. Colocam-se os algarismos do numero secreto e do numero do utilizador na pilha e inicia-se o corpo do programa que vai comparar estes 2 numeros;
10. CicloXis: Detetam-se os algarismos do numero do utilizador que existem no numero secreto e estão na posição certa e atribui-se um resultado codificado em 12 bits respetivo;
11. CicloBola: Detetam-se os algarismos do numero do utilizador que existem no numero secreto mas estão na posição errada (Sem avaliar os que já foram detetados pelo CicloXis anteriormente) e atribui-se um resultado codificado em 12 bits respetivo;
12. Descodifica-se o resultado dá-se print na janela de texto do resultado em ‘x’,’o’ ou ‘-‘;
13. No caso da jogada estar incorreta, incrementa-se o valor da pontuação nos Displays e recomeça-se uma nova jogada (Volta ao ponto 7.);
14. No caso da jogada estar correta, de se ter chegado ao final das 12 jogadas ou de o jogador ter ficado sem tempo numa das jogadas, imprime-se as mensagens “Fim do Jogo!” e “Carregue em IA para recomeçar” na janela de texto e acaba-se o jogo atual;
15. Compara-se a pontuação atual com a melhor pontuação (No caso de o jogador ter acertado no número), atualiza-se o LCD com a atual no caso desta ter sido melhor e inicia-se um novo jogo (Volta ao ponto 6.)

Ao nível dos **periféricos**, todos os requesitos foram implementados tal como pedido no enunciado. Houve algumas implementações que não eram bem especificadas e que fizemos como achamos mais conveniente, tais como:

- Na representação da pontuação atual nos Displays de 7 segmentos, não era especificado se a pontuação deveria estar em decimal ou hexadecimal, pelo que optamos por representá-la em decimal, usando os 2 últimos displays para tal;
- Na representação do tempo nos 16 LEDs, não era dito o que deveria acontecer após o jogador acertar na combinação secreta, pelo que nós decidimos desligar os LEDs todos quando tal acontecesse, ao invés de os deixar correr normalmente (deixá-los desligar sucessivamente a cada 500ms);

Quanto à compreensão da representação da jogada, do número secreto (chave) e do resultado em 12 bits:

- **Jogada e Chave** - Como a jogada e a chave só podiam ter algarismos de 1 a 6, a sua compreensão para 12 bits era relativamente simples, pois os números de 1 a 6 em decimal só ocupam no máximo 3 bits em binário, logo os 12 bits serviam para comprimir o número em binário da seguinte forma: 0000 xxx xxx xxx xxx , sendo cada conjunto de 3 x's um algarismo. Mais tarde, quando quisessemos utilizar os algarismos em separado, bastava fazer AND com 0007h (Ultimos 3 bits a 1 em binário) para pegar no último algarismo e SHR de 3 unidades para passar o próximo algarismo para a posição do último (Para depois se fazer um novo AND).

- **Resultado** – Optámos por codificar o resultado da seguinte forma:

  X (Algarismo certo na posição certa) = 001b = 1 em decimal

  O (Algarismo certo na posição errada) = 010b = 2 em decimal

  - (Algarismo errado) = 011b = 3 em decimal

Como o resultado vai ter 4 "algarismos", só vai ocupar no máximo 12 bits com esta codificação (Até poderia ser feito para ocupar apenas 8 bits com esta codificação)

Exemplo: 0000 011 010 001 001 → **- o x x**

Na passagem do resultado para a janela de texto fizemos um descodificador que pega nestes valores, descodifica-os para o respetivo 'x', 'o' ou '-' e imprime-os na janela de texto.

Em suma, penso que conseguimos implementar todas as funcionalidades pedidas pelo enunciado e foi um projeto desafiante e interessante de se fazer.