TÉCNICO LISBOA

CURSO: ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

DISCIPLINA: ARQUITETURA DE COMPUTADORES

TRABALHO DE LABORATÓRIO III E IV

PROGRAMAÇÃO ASSEMBLY - MASTERMIND

Trabalho Realizado por: Xavier Abreu Dias Nº 87136

Diogo Martins Alves Nº 86980

Data: 02/06/2017

1. Introdução

Nestes dois laboratórios pretendeu-se implementar em *Assembly* o jogo que é conhecido como *Mastermind*. Inicialmente é escolhido uma combinação de cores pelo computador e um jogador terá que tentar acertar nessa combinação no menor número possível de tentativas (10 no máximo). Para o efeito, são dadas informações ao jogador em cada jogada, em concreto, quantas cores acertou na posição correta e quantas cores acertou mas que não estavam na posição correta. A dificuldade pode variar com o tempo que é disposto para cada tentativa bem como o número de cores a acertar. Em modo multi-jogador, ao fim de cada jogo serão mostradas as estatísticas desse jogo.

2. Representar tabuleiro, criar sequência aleatória e avaliar jogada (1ª semana)

Para iniciarmos o programa, criámos uma label ("Inicio"), onde começará a ser escrito o programa principal. Aqui, começámos por inicializar o Stack Pointer, a máscara de interrupções (a 8003h, uma vez que iremos precisar dos botões de pressão 0 e 1 e do temporizador) e a tabela de vetores de interrupção, associando a cada interrupção a sua rotina.

2.1. Representar o tabuleiro

Para representar o tabuleiro em si, criámos a rotina *GerarTabuleiro*, que imprime no ecrã uma série de strings, nomeadamente uma linha para o nome do jogo, uma linha para o "cabeçalho" da tabela que representa o tabuleiro, 10 linhas para as tentativas, que informam o jogador qual a sua tentativa atual e uma linha para a solução do jogo.

2.2. Sequência aleatória

Para gerarmos a sequência de cores aleatória, utilizámos o algoritmo do enunciado, em que o valor de Ni, que corresponde ao número aleatório que está sucessivamente a ser alterado, corresponde ao conteúdo da posição de memória *RandNum*. A partir deste valor, para gerarmos um número aleatório entre 0 e (número total de cores – 1), basta-nos dividir o valor obtido anteriormente pelo número total de cores e o valor pretendido corresponde ao resto dessa divisão. Deste modo, cada cor tem um código entre 0 e (número total de cores – 1).

A rotina *MostarSol* tem como objetivo mostrar na última linha do tabuleiro esta sequência chave. Esta rotina é chamada sempre que um jogador termina (perde ou ganha) um jogo. O botão de pressão 0 tem como objetivo fazer o jogador desistir do jogo, logo, quando é premido, o programa salta para a posição *Derrota*, onde esta rotina é executada.

2.3. Leitura e avaliação da jogada

Para aceitarmos as tentativas do jogador, criámos um ciclo dentro do programa principal (*CicloTentativas*) que corre 10 vezes (ou menos, caso o jogador acerte na sequência antes das 10 tentativas), e que em cada execução chama a rotina *EsperarTentativa*. Esta rotina recebe a sequência de cores correspondente à tentativa atual do jogador, mostra-a no ecrã e guarda-a no vetor *TentAtual*. Para mostrar a tentativa inserida pelo jogador na linha correta do ecrã, esta rotina recebe o número da tentativa atual através do STACK.

Para avaliarmos a jogada atual do jogador, recorremos à rotina *AvaliarTentativa*, que, tal como a anterior, recebe o número da tentativa atual pelo STACK. Esta rotina começa por criar uma cópia da tentativa atual do jogador em *TentAtual_Copia* e também da sequência correspondente à solução em *Sequencia_Copia*. Depois vai comparar estes dois vetores para ver se existem "P", isto é, cores da tentativa do jogador que existam na solução e estejam na posição correta. Para o fazer, irá comparar o conteúdo de posições iguais dos vetores *TentAtual_Copia* e *Sequencia_Copia*. Se encontrar um "P", guarda no vetor *Avaliacao*, na posição respetiva, o caracter 'P', e altera as posições correspondentes dos vetores *TentAtual_Copia* e *Sequencia_Copia* para VAZIO (um valor que não representa nenhuma cor), para que esta posição não volte a ser contabilizada. De seguida, o programa vai verificar se existem "B", isto é, cores da tentativa atual do jogador que existam na solução, mas que não estejam na posição correta. Para tal, o programa irá comparar cada posição do vetor *TentAtual_Copia* com todas as posições do vetor *Sequencia_Copia*. Se encontrar um "B", o comportamento é idêntico ao de um "P" - guarda no vetor *Avaliacao*, na posição respetiva, o caracter 'B', e altera as posições correspondentes dos vetores *TentAtual_Copia* e *Sequencia_Copia* para VAZIO, para que esta posição não volte a ser contabilizada. Por fim, para mostrar a avaliação da jogada no ecrã, tal como no jogo original, são mostrados na linha da tentativa respetiva primeiro todos os "P" existentes e só a seguir os "B".

No programa principal, após o jogador inserir cada tentativa, é chamada a rotina *VerificarVit*, que verifica se todos os elementos do vetor *Avaliacao* são "P". Se isto acontecer, devolve o valor 1 através do registo R2. O programa avalia depois este registo e se for 1, salta para a posição Vitoria, onde é mostrada a solução e uma mensagem de vitória, chamando a rotina *MostrarVitoria*. Para além da mensagem de vitória, esta rotina faz uma chamada à rotina *Leds_vitoria*, que mete os leds verdes a piscar. Caso as tentativas se acabem e o jogador não acertar na solução, o programa salta para a posição Derrota, onde é mostrada a solução e uma mensagem de derrota chamando a rotina *MostrarVitoria*, que mete também os leds vermelhos a piscar.

3. Ecrã inicial, Cronómetro e Demo da 1ª fase (2ª semana)

3.1 Ecrã inicial

Para criar o ecrã inicial bastou-nos declarar uma série de strings correspondentes a informações sobre regras e comandos de jogo e imprimi-las no ecrã, antes do tabuleiro de jogo, chamando a rotina *EcraInicial*. No entanto, este ecrã inicial tem uma outra funcionalidade – gerar a semente para o algoritmo de geração de números aleatórios. Esta semente corresponde ao valor de um registo quando o utilizador clica numa tecla. Este registo está a incrementar num ciclo e o programa só sai desse ciclo quando o utilizador premir numa tecla. Deste modo, o valor do registo pode ser considerado aleatório e funciona como uma semente para o algoritmo, uma vez que o utilizador nunca leva o mesmo tempo a clicar na tecla – há sempre ligeiras variações.

3.2 Cronómetro

Para implementarmos o cronómetro, recorremos à interrupção 15 (temporizador) do P3. Para tal criámos a rotina *IniciarContagem*, que é chamada no início de cada tentativa e que serve para iniciar/reiniciar a contagem decrescente do cronómetro. O que esta rotina faz é transferir para o conteúdo da posição de memória *ContagemTempo* o valor em segundos correspondente ao tempo de jogo escolhido pelo jogador (1,2 ou 4 minutos). Para além disto, esta rotina também arranca o temporizador, metendo M[TIMER_COUNT] a 10 (10*100ms = 1s) e M[TIMER_START] a 1. Precisámos também de definir a rotina de interrupção do temporizador - *RotinaInt15*. Esta rotina volta a arrancar o temporizador, decrementa o valor de *ContagemTempo* e verifica se esta é zero. Se for zero, significa que o jogador perdeu o jogo e então alteramos o endereço de retorno da rotina de modo a que esta não volte para o sitio onde tinha ficado quando foi chamada, mas sim para a posição *Derrota*. Por fim, definimos ainda uma outra rotina – *ParарContagem*, que é executada quando o jogador termina um jogo (perde ou ganha) e que mete o valor de *ContagemTempo* a zero e para o temporizador.

Para além destas, definimos também a rotina *EscreverDisp7Seg*, que escreve no display de 7 segmentos o valor de *ContagemTempo*, em que os dois displays mais à direita correspondem aos segundos e o terceiro aos minutos.

4. Multi-jogador, Níveis de Jogo e Estatísticas (3ª semana)

Para implementarmos as novas funcionalidades da 3ª semana de laboratório, como é necessário perguntar alguns parâmetros de jogo ao utilizador, decidimos criar um segundo ecrã, que é mostrado a seguir ao ecrã inicial, e que pergunta ao jogador o número de jogadores que terá o jogo (entre 1 e 9), o número de cores da sequência que terá que ser adivinhada (4, 5 ou 6) e o número de minutos de cada tentativa (1,2 ou 4). Todos estes dados são guardados na memória, respetivamente nas posições *NumJogadores*, *NumeroCoresSequencia* e *TempoJogo*.

4.1 Versão Multi-Jogador

Para implementarmos a versão multi-jogador ao código que já estava escrito bastou-nos substituir a ocorrências anteriores do número de jogadores por M[*NumJogadores*]. Tivemos também que criar mais dois ciclos no programa principal – *CicloJogadores*, que corre uma vez por cada jogador; e *CicloNovoJogo*, que é corrido quando o utilizador pede para iniciar um novo jogo (após o jogo anterior terminar).

4.2 Níveis de jogo

A implementação de novos níveis de jogo foi relativamente simples – bastou-nos substituir no código antigo as ocorrências do número de cores da sequência por M[*NumeroCoresSequencia*] e as ocorrências do tempo total de uma tentativa por M[*TempoJogo*]. Tivemos ainda que aumentar a largura do tabuleiro de modo a que este tivesse espaço para um máximo de 6 cores por cada tentativa.

4.3 Estatísticas

Para a parte das estatísticas, começámos por criar as seguintes rotinas, que são chamadas caso o jogador atual consiga acertar na solução: *AtualizarMelhorJogador*, que substitui o valor de M[*MelhorJogador*] por M[*JogadorAtual*], caso o jogador atual tenha conseguido acertar na solução em menos jogadas que o antigo melhor jogador; *AtualizarEstatisticas_Jogo*, que atualiza as posições M[*AccJogadasAcerto*] (acumulador das jogadas até ao acerto, de todos os jogadores), M[*NumVitorias*], M[*AccTemposJogo*] (acumulador dos tempos de jogo de todos os jogadores) e M[*MelhorTempoJogada*]; e *AtualizarEstatisticas_Jogador*, que atualiza as estatísticas como a rotina anterior, mas referentes apenas ao jogador atual.

Para mostrar ao utilizador as estatísticas de jogo, é feita uma chamada à rotina *FimJogo* dentro das rotinas *MostrarVitoria* e *MostrarDerrota*, caso todos os jogadores tenham feito a sua jogada. Esta rotina mostra o vencedor na janela de texto e chama a rotina *MostrarEstatisticas*, que mostra no LCD o melhor tempo de jogo no formato minutos e segundos, a média dos tempos de jogo, também neste formato e a média de jogadas para acerto arredondado às décimas. Nesta fase, o programa fica em ciclo enquanto o jogador não clicar no botão de pressão 1. Quando o fizer, é iniciado um novo jogo.

Para mostrar as estatísticas de cada jogador, é chamada a rotina *MostrarEstatisticas_Jogador* quando um jogador termina a sua jogada. Esta rotina faz exatamente o mesmo que a anterior, mas mostra as estatísticas melhor tempo de jogo, média dos tempos de jogo e média de jogadas de acerto relativas ao jogador em questão (*JogadorAtual* guarda o valor do jogador atual) e mostra estas estatísticas na parte direita da janela de texto.