Trabalho realizado por: Grupo 48
Amândio Faustino Nº 83422, João Sousa Nº 83487, Pedro Lopes Nº 83540

Introdução

O projeto de Introdução a Arquitetura de Computadores consiste na realização de uma versão de um jogo já conhecido, *flapy bird*. Neste jogo o objetivo é manter um pássaro no ar, saltando, de forma a evitar os obstáculos que vão aparecendo conseguindo percorrer a maior distância possível e passar o maior número de obstáculos. Esta versão é programada em linguagem Assembly para ser executada pelo processador P3.

Quando se inicia o jogo aparece um menu em que é possível selecionar o nível a que queremos jogar usando a interrupção I1 para diminuir e I2 para aumentar. As diferenças entre níveis são a velocidade a que os obstáculos se movem. Quando se toca num obstáculo, ou nos limites perde-se e para recomeçar é pressionar a interrupção.

Desenvolvimento

Estrutura geral do jogo

Para implementar o jogo usamos uma rotina principal, Jogo, que chama varias rotinas chamadas nesse grupo que servem para desempenhar os aspetos mais complexos e importantes do jogo, como é por exemplo a Obstáculos, Gravidade e a Colisão. Estas por sua vez no seu funcionamento vão chamar outras rotinas que servem para a não repetição de código.

Para efeitos de utilidade e simplicidade, definimos que R2 possui sempre a próxima posição do corpo do pássaro, sendo que R2+1 possui o bico e R1 possui sempre a posição atual do pássaro. A rotina Pássaro é a rotina responsável pela escrita do pássaro, esta só atualiza a posição quando R1 é diferente de R2,o que significa que este se moveu.

Como estruturas de dados usamos um vetor para controlar os obstáculos de forma a mantermos os obstáculos sempre organizados. Para além disso, também usamos inúmeras variáveis para controlar os vários processos a serem desempenhados durante o programa. Assim como várias constantes para certificar que estes valores se mantinham como pretendidos ao longo da execução.

Gravidade

Esta rotina permite que o pássaro desça com um movimento acelerada. Para implementar a gravidade utilizamos a equação paramétrica da posição, y = y0 + v0t +at^2, em que v0t é sempre 0. A aceleração é dada por uma constante que pode ser alterada na tabela das constantes para o movimento ser mais ou menos acelerado, y0 é a posição atual do pássaro guardada em R1, o tempo é dado pela variável Contador Tempo. É possível também alterar a velocidade de atualização do pássaro com a constante duraçãotick que está definido para 1 para ser de 0.100s em 0.100s (o mínimo). Com estes dados é calculada a próxima posição do pássaro que é colocada no final da rotina em R2

Colisão

Esta rotina é a rotina de verificação se o pássaro bateu em algum dos obstáculos ou limites. Como as coordenadas dos tubos estão guardadas em vetores, a rotina vai procurar o primeiro vetor não nulo que vai ser consequentemente o primeiro a ser ultrapassado e verifica se este vetor já se encontra na sua coluna, caso se verifique compara a posição do pássaro com o espaçamento da coluna, isto vai devolver se o pássaro bateu ou se se encontrava no

espaçamento. Se o pássaro não bater, a coluna avança e a rotina vai buscar o próximo vetor que contem a próxima coluna a passar pelo pássaro visto que os vetores estão organizados por ordem de criação.

Obstáculos

Esta rotina tem a função de criar os objetos que vão ter de ser ultrapassados pelo jogador. Primeiramente a rotina vê se já passou o tempo necessário para mover os obstáculos presentes no ecrã. Caso já tenha passado o espaço pré-definido (constante CRIA_TUBO) entre o ultimo tubo a ser criado e o início da janela é criado um novo obstáculo, para isto é chamada a rotina Randomizer que dá um valor aleatório entre 1 e 17 que é a linha do primeiro espaço vazio da coluna, este espaçamento vai ser atribuído ao obstáculo indicado pela variável POS_TUBOS (variável que controla a memoria onde são escritos os obstáculos). Posto isto, a rotina vai percorrer todos os índices do vetor que possuem as memórias dos obstáculos e que estes têm a coluna diferente de 0, isto é aqueles visíveis no ecrã, e vai apagar aqueles presentes na janela de texto e escrevê-los na coluna a seguir, movendo-os assim.

Níveis

Para alterar o nível, ou seja alterar a velocidade de descolamento do pássaro que ao nível do código consiste em precisar de menos interrupções de relógio para voltar a mover os obstáculos 1 coluna utilizamos um menu, em que só é possível alterar o nível neste menu não sendo possível alterar durante o jogo. O nível é selecionado através das interrupções I1 e I2 em que I1 diminui o nível e I2 aumenta o nível, isto traduz-se num aumento e um decréscimo do contador de nível respetivamente.

Conclusão

Para concluir, foi necessário para a realização deste projeto o desenvolvimento de varias sub-rotinas, ou rotinas auxiliares para o código ser mais fácil de ler, mais coeso. Maioritariamente usamos o modo de passagem de parâmetros por registo utilizando a pilha salvaguardar os seus valores iniciais. Tentamos maioritariamente seguir o enunciado cumprindo as suas exigências mas tivemos alguns problemas na parte da gravidade visto que implementamos de uma maneira que nos pareceu simular a gravidade mas que seria matematicamente incorretos. Não seguimos também o facto de as colunas terem de possuir um espaçamento de 5 casas pois achamos a jogabilidade demasiado difícil mas desenvolvemos o código de modo a poder implementar este espaçamento mudando apenas uma variável.

Tirando estes pequenos problemas, consideramos que a parte mais difícil do projeto foi programar para que o código funcionasse na placa do P3 pois nas primeiras versões do código que experimentamos funcionava no simulador mas na placa não, apresentando vários bugs.

Adiciona-mos ainda um menu ao programa, que nos permite selecionar o nível no início do jogo, esta pareceu a maneira mais intuitiva de realizar a mudança de nível sendo que desta forma não é possível alterar o nível do jogo enquanto o jogo esta a decorrer.

Em jeito de conclusão, achamos que este projeto foi realizado de uma forma satisfatória ou mesmo boa, sendo que aprendemos a programar Assembly de uma forma coesa com um código compreensível e executável tanto na placa como no simulador.