Introdução à Arquitetura de Computadores, Grupo 79

Relatório do Projeto ''Jogo Dino''

Leonor Marques, nº 99262

Pedro Rodrigues, nº 99300

No âmbito da cadeira de Introdução à Arquitetura de Computadores foi-nos pedido que desenvolvêssemos o jogo 'Dino' em Assembly do processador P4. Para isso, foi criada uma sequência de rotinas que utilizam os vários periféricos disponíveis (Temporizador, Display de 7 segmentos, Interrupções de Teclado e Terminal).

- **Rotinas**

Foram implementadas diversas rotinas de modo a tornar o programa mais percetível, para isto, foram definidas diversas variáveis que, enquanto o programa corre, sofrem alterações, provocando o aparecimento de novos catos com altura variável entre 1 a 8 (potência de 2).

- **Campo de Jogo**

O campo de jogo consiste num vetor, mantido em memória, com 80 posições, número de colunas do terreno de jogo, e em que cada entrada do vetor contém num inteiro com a altura do cato que está nessa posição do terreno de jogo (se for 0 não existe cato). Este vetor foi guardado na memória a partir do endereço 0000h. Usámos um comando TAB para guardar informação relativa à dimensão do terreno de jogo.

- **Variáveis**

MAP_DIM – 80: variável correspondente ao tamanho do terreno de jogo;

CATO_MAX_ALTURA – 8: altura máxima possível para os catos gerados na função geracacto;

SALTO_MAX – 12: o dino ('D') consegue atingir uma altura máxima de 12 (superior a qualquer tamanho possível para os catos);

B – 5: valor arbitrário que permite guardar nesse mesmo registo números de 16 bits sem sinal.

- **Rotinas mais importantes**

Geracacto: Esta função cria um valor aleatório, que pode ser 0 (não é criado um cato) com uma probabilidade de 95%, ou cria um cato com um valor uniformemente distribuído entre 1 e 8.

Atualizajogo: Com esta função conseguimos deslocar uma posição para a esquerda todos os elementos do vetor (como exemplo disso temos um valor 5 que passará de um endereço 0021h para 0020h). Como tal, o valor guardado na posição mais à esquerda do terreno de jogo vai ser apagado (no caso deste código é o valor guardado em 0020h). Ao mesmo tempo, verificamos que o valor contido na posição mais à direita será ocupado pelo valor fornecido pela função *geracacto*.

Atualizamapa: Com esta função, é verificado se o jogo está a avançar (se o jogador não perdeu). Assim sendo, caso haja um cato na posição mais à esquerda apaga-o e copia todos os valores da direita para a casa imediatamente à esquerda. De seguida, na nova posição (posição mais à direita) gera um novo cato caso exista ou simplesmente na nova posição não gera nenhum cato.

INIT_TIMER: Esta função aliada à inicialização do temporizador permite contar a pontuação sabendo que é diretamente proporcional ao tempo, ou seja, a cada 1 segundo e 2 décimas a pontuação sobe 1.

- **Conclusão**

O jogo foi elaborado de um modo estruturado, existindo uma separação entre as rotinas de processamento dos periféricos, as rotinas que gerem o núcleo de jogo e as rotinas que gerem o processamento do terreno de jogo e de novos catos.

O jogo inicia quando o utilizador carrega no botão “0”, estando o terreno de jogo inicialmente sem catos. Quando o dinossauro (‘D’) colide com um cato imprime a mensagem ‘---> Game Over <---' no centro do écran e o programa fica a aguardar que o utilizador volte a carregar em “0”, para limpar o mapa e reiniciar com as variáveis e pontuação a 0.

Por último, recomendamos que se corra o programa a 100 kHz de modo a permitir uma boa jogabilidade.