Projeto de Introdução à Arquitetura de Computadores

Bruno Miguel Vaz e Campos, 99187, e João Henriques Sereno, 99249

Primeira Parte:

Desenvolvemos as funções pedidas no enunciado e elaborámos um pequeno programa de teste que foi enviado em conjunto com este documento.

Este pequeno programa de teste irá correr indefinidamente ambas as funções (na segunda parte do projeto será implementada, por exemplo, a deteção de colisões que irá parar o "loop infinito"). No entanto, não impossibilita de todo a verificação de que realmente funciona. Antes pelo contrário, pois permite ao utilizador correr o programa o tempo que desejar.

No ínicio do programa foi definido o vetor "vetor" com 20 posições na posição de memória 4000h. De seguida, a variável "x" (em concordância com o código python disponibilizado no enunciado) que irá guardar o valor da "seed" na posição 4200h (razão esclarecida no programa). Inicializamos a pilha com o "stack pointer" no endereço 8000h.

atualizajogo: A função, com o auxílio do loop "loop_SHL", começa por deslocar cada valor de uma posição *n* para uma posição *n-1*, isto é, uma posição para a esquerda. Com isto, o valor correspondente à primeira posição é descartado. Chegada à última posição (mais à direita), é invocada a sub-rotina **geracacto**, que gera e devolve, no registo R3, o valor que será armazenado nessa posição (que corresponderá à altura do novo cato, caso este valor seja maior do que 0). Para efeitos de jogo, o objetivo será, em cada frame, correr esta sub-rotina para atualizar a tela de jogo, dando o efeito de movimento contínuo e ir gerando, ou não, novos obstáculos na coluna mais à direita.

geracacto: Guarda no registo R2 o valor da "seed" (variável x) e procede às operações definidas no código python do enunciado. Na linha de código 76 "BR.C et2" é verificado se na comparação anterior gerou Carry, contraintuitivamente a verificar se o resultado deu negativo. A razão para tal reside no facto de o p4 assumir, na instrução CMP, os valores em complemento para dois. Assim, o bit mais significativo corresponderá ao sinal do valor, sendo, portanto, esse valor, em decimal, -3278 ao invés de 62258 (este último necessitaria de 16 bits para ser representável, visto que a sua representação em binário é 1111 0011 0011 0010). Então, o significado original das "flags" nesta comparação não pode ser o original.

Nota: as primeiras três linhas de código, bem como o código correspondente à etiqueta *teste* permitem introduzir valores de teste no vetor, valores esses provenientes de uma lista, guardada em memória.

Instruções para teste de código:

- Regular a velocidade máxima de processamento (slider na parte de cima do simulador), por forma a ser possível ver as alterações a acontecerem em tempo real, ao nível do deslocamento dos valores no vetor e consequente introdução de novo valor na última posição (aconselha-se entre 100Hz e 1000Hz)
- Iniciar o programa
- Verificar, no espaço de memória de dados, o intervalo de endereços entre 4000h e 4014h, correspondentes às 20 posições do vetor. Por cada ciclo de deslocamento, um novo valor será introduzido no endereço de memória 4014h (de notar que esse valor poderá assumir os valores inteiros entre 0 e 4, dado que a altura máxima por nós introduzida, para efeitos de teste, foi 4)