



Projeto de Introdução à Arquitetura de Computadores



Objectivo

O projeto consiste no desenvolvimento de um jogo muito simples, que é uma versão simplificada do jogo Dino, que surge no browser Chrome quando a máquina não consegue aceder à Internet. (Na maioria dos sistemas operativos, o jogo também pode ser acedido através do endereço `chrome://dino/` no Chrome.)

O programa será escrito em Assembly do P4. O desenvolvimento e teste do programa serão realizados utilizando o simulador do P4 e a placa de desenvolvimento.

Implementação

A implementação está dividida em duas fases, com diferentes datas de entrega.

Primeira parte

A primeira parte do projeto consiste num programa com um conjunto de funções e um simples código de teste, escrito em Assembly do P4, para simular o estado do terreno de jogo.

O estado consistirá num vetor (ou tabela), mantido em memória, com N posições em que N é o número de colunas do terreno de jogo, e em que cada entrada do vetor consiste num inteiro com a altura do cacto que está nessa posição do terreno de jogo. Por exemplo, um vector com o seguinte conteúdo albergado na memória a partir do endereço 4000h:

Ender.	4000h	4001h	4002h	4003h	4004h	4005h	...	404Dh	404Eh	404Fh
Conteúdo	4	0	0	0	0	0		0	0	3

Corresponde a um terreno de jogo com N=80 colunas e com um cacto em cada uma das extremidades do écran, um deles (pode-se convencionar que é o do lado esquerdo) com altura de 4 linhas, e o outro (do lado direito) com 3 linhas.

Na primeira parte do projeto deverá implementar a função atualizajogo, que atualiza o terreno de jogo com a seguinte especificação.

Nome da função:

atualizajogo

Parâmetros:

1. endereço de memória do início do vector com o terreno de jogo
2. dimensão (número de posições) do vector

Valor de retorno:

nenhum

Especificação da função:

A função vai deslocar todos os elementos do vector uma posição para a esquerda (isto é, do endereço n para o endereço n-1). Como tal, o valor contido na posição mais à esquerda vai ser eliminado do vector. Por sua vez, o valor contido na posição mais à direita será preenchido com o valor de retorno da seguinte função auxiliar:

Nome da função:

geracacto

Parâmetros:

1. valor máximo (altura máxima do cacto, deve ser uma potência de 2)

Valor de retorno:

1. valor aleatório gerado

Especificação da função:

A função gera um número aleatório, que pode ser zero com uma probabilidade prédefinida de 95%, ou um valor uniformemente distribuído entre 1 e o valor máximo passado como parâmetro. O (pseudo-)código em Python da função é o seguinte:

```

x = 5
# "semente" de valor arbitrário -- alterar para obter nova sequência
# x guarda números inteiros de 16 bits sem sinal

def geracacto(altura): # altura deve ser uma potência de 2
    global x
    bit = x & 1    # AND bit a bit
    x = x >> 1     # Shift para a direita
    if bit:
        x = XOR(x, 0xb400) # altera potencialmente qualquer bit de x

    if x < 62258: # o intervalo [0, 62258] abrange aproximadamente 95% dos
        return 0 # valores representáveis com 16 bits.

    # garante que o valor retornado pertence ao intervalo [1, altura].
    # x & (altura - 1) = mod(x, altura), quando altura é uma potência de 2.
    # Porquê?
    return (x & (altura - 1)) + 1

```

Qualquer parte do programa que não esteja completamente especificada no enunciado pode ser implementada da forma que entenderem melhor, desde que justificada no relatório, e tentando privilegiar a jogabilidade.

Adicionalmente, deverá entregar no código principal do programa onde estão as duas funções acima um pequeno programa simples, que permita testar as funções.

Segunda parte

A segunda parte do projeto será disponibilizada na semana de 16 de novembro. Nesta segunda parte irá implementar a lógica do dinossauro, da detecção de colisões, de entradas/saídas, e atualização periódica do terreno de jogo.

Entregas

1ª Parte: dia 16 de novembro, 23h59

Deverão ser entregues os ficheiros de código criados (ficheiros com formato .as), assim como um documento com uma breve descrição (max. 1 página A4) das funções e do programa de teste (bem como instruções claras sobre como testar o código das funções).

2ª Parte: dia 4 de dezembro, 23h59

Deverão ser entregues os ficheiros de código criados (ficheiros com formato .as) e o relatório do projeto (em formato .pdf).



Projeto de Introdução à Arquitetura de Computadores

Segunda parte

A segunda parte do projeto consiste no desenvolvimento de um programa, em Assembly do P4, que implementa o jogo "dino" utilizando as funções desenvolvidas na primeira parte do projeto.

Saída ("Output") no terminal

No terminal do P4 deverá desenhar, e periodicamente atualizar, o terreno de jogo (ver secção 5.1 do Manual do P4). Têm liberdade total de utilizar os caracteres ASCII que quiserem para representar o dinossauro, os cactos, e o solo. O dinossauro será sempre desenhado na mesma coluna do écran, e ocupando (quando não estiver a saltar) algumas linhas imediatamente acima da linha onde está representado o solo.

Salto do dinossauro

O salto é desencadeado quando o utilizador carrega na tecla "seta para cima", mas apenas no caso de não estar nenhum salto em curso. A trajetória do dinossauro deve ser implementada de uma forma muito simples, através de um movimento a velocidade constante (por exemplo, uma linha por cada atualização da saída no terminal), inicialmente para cima até atingir uma altura máxima (determinada por uma constante no código), seguido de uma descida também a velocidade constante até atingir novamente a linha imediatamente acima do solo.

Início e final do jogo

O jogo inicia quando o utilizador carrega no botão "0", estando o terreno de jogo inicialmente sem cactos. Quando o dinossauro colide com um cacto deve aparecer a frase "G A M E O V E R" no centro do écran e o programa fica a aguardar que o utilizador volte a carregar em "0".

Pontuação

Tal como no jogo original, à medida que o jogo progride a pontuação vai sendo aumentada, sendo mostrada, em decimal, nos mostradores de 7 segmentos. (Ao contrário do original, não é necessário manter o "high score".)

Implementação

- Antes de começarem a implementação devem planejar a estrutura geral do código. Se for útil devem discutir esta estrutura com o docente de Laboratório (nas aulas de dúvidas, na aula de Laboratório, ou por e-mail).
- A informação que descreve o estado do jogo deve ser mantida em variáveis mantidas em memória. Naturalmente, do ponto de vista de eficiência da execução do código, deverá trazer as variáveis da memória para os registos antes destas serem manipuladas, se possível mantê-las em registos durante um trecho de código significativo (por exemplo, na execução de uma função), e voltar a escrevê-las em memória após a sua utilização.
- A passagem do tempo e consequente atualização do terreno de jogo é controlada através do temporizador, cujas interrupções devem ser programadas adequadamente. Notem que em geral as interrupções devem ser rotinas muito curtas, normalmente limitando-se a ajustar o valor de uma ou mais variáveis, que depois serão utilizadas para controlar a execução do programa principal.
- Irá ser avaliada a qualidade do código, nomeadamente no que diz respeito à sua modularidade (e.g., uso de funções reutilizáveis e fáceis de manter), à passagem de parâmetros de forma que obedeça às convenções ensinadas na cadeira, ou à utilidade dos comentários.

Entrega

Dia 4 de dezembro, 23h59

Deverão ser entregues os ficheiros de código criados (ficheiros com formato .as) e o relatório do projeto (máximo duas páginas, 11pt, em formato .pdf). O relatório deve descrever a estrutura geral do código, um sumário das funções implementadas, e justificar as decisões principais no desenho da aplicação.