Projeto de Introdução à Arquitetura de Computadores

LEIC

IST-Taguspark

Tetris Invaders

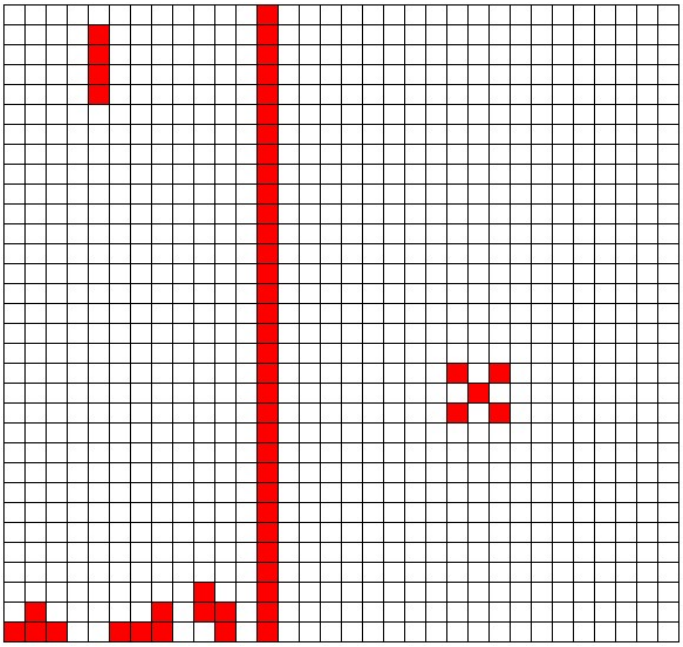
2016/2017

1 – Objetivos

Este projeto pretende exercitar os fundamentos da área de Arquitetura de Computadores, nomeadamente a programação em linguagem *assembly*, os periféricos e as interrupções.

O objetivo deste projeto consiste em concretizar um jogo, mistura simplificada dos jogos Tetris e Space Invaders. Num ecrã vão caindo peças, que o jogador pode rodar e deixar cair mais rapidamente, quando está na posição pretendida. Param quando batem no fundo do ecrã ou noutra peça que já exista. Se se completar uma linha, o jogador ganha 5 pontos. Pode também aparecer um monstro, a mover-se na horizontal, da direita para a esquerda. Se o monstro atingir o lado esquerdo do ecrã, o jogo termina. O jogador pode destruir o monstro deixando cair uma peça em cima dele, caso em que ganha 2 pontos. A pontuação até ao momento é mostrada em displays de 7 segmentos. O jogador pode terminar, recomeçar ou suspender/continuar o jogo quando quiser.

A figura seguinte ilustra um possível aspeto do jogo (quadrícula não existe no ecrã do jogo).

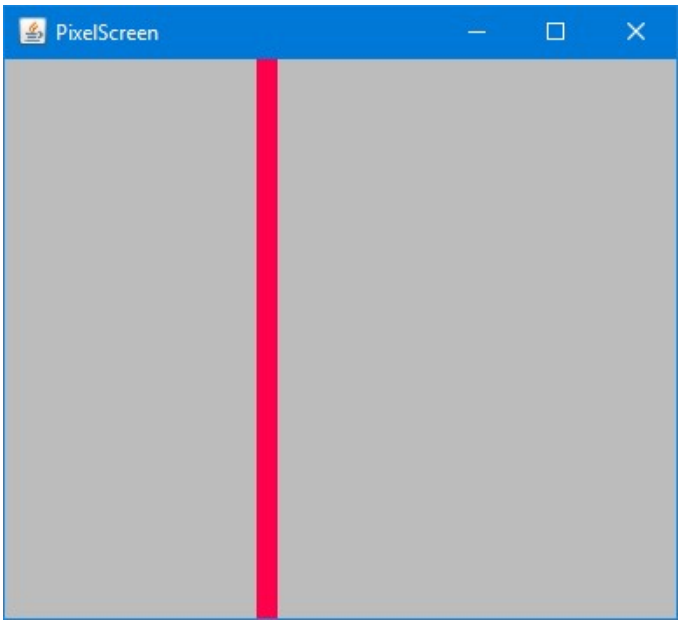


2 – Especificação do projeto

2.1 - Ecrã

O ecrã de interação com o jogador tem 32 x 32 pixels, tantos quantas as quadrículas da figura anterior.

A área de jogo propriamente dita consiste nas N colunas da esquerda, sendo N uma constante do programa que deve poder ser mudada. O exemplo usa N=12, mas pode-se usar qualquer outro valor entre 9 e 15. Deve ser desenhada uma "parede" vertical para separar a área de jogo da restante.

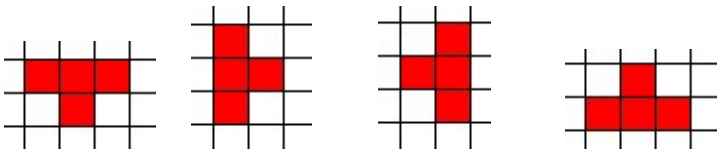


Há dois tipos de objetos no ecrã:

- Tetraminó, peças sempre de 4 pixels (característica que deu origem ao nome do jogo Tetris). Há quatro tetraminós diferentes, em forma de I, T, L e S. Todos estão representados na figura anterior;
- Monstro, representado na figura anterior por uma estrela, mas cuja forma pode ser qualquer outra.

2.2 - Tetraminós

Cada tetraminó deve poder tomar uma de quatro representações possíveis. Tipicamente pode rodar-se, mas também é possível usar representações espelhadas face a outras. O tetraminó em forma de I pode apenas tomar duas representações, "em pé" e "deitado". As figuras seguintes exemplificam as diversas representações do tetraminó em forma de T:



Quando se inicia o jogo, começam a cair tetraminós da parte de cima da área de jogo, um de cada vez. A escolha de qual tetraminó cai, dos quatro possíveis, deve ser feita de forma aparentemente aleatória.

O ritmo de queda deve poder ser controlado. Durante a queda, o jogador pode:

- Rodar o tetraminó, circulando pelas suas quatro representações possíveis;
- Deslocar o tetraminó para a esquerda ou para a direita, um pixel de cada vez;
- Deixá-lo cair de forma rápida (à velocidade máxima que o programa permita, mas passando por todas as posições intermédias de forma a visualizar-se a queda).

Cada tetraminó desce até bater no fundo ou noutra peça que já exista. Só nessa altura pode começar a cair um novo tetraminó.

Quando uma linha ficar totalmente preenchida, deve ser eliminada, copiando cada uma das linhas acima para a linha que lhe está por baixo. Poderá suceder que um tetraminó, chegando ao fundo, complete mais do que uma linha. Todas as que forem completadas devem ser eliminadas.

Se os tetraminós se empilharem demasiado (porque as linhas não vão sendo completamente preenchidas) e a pilha chegar ao topo, impedindo um novo tetraminó de sequer ser desenhado, o jogo deve ser terminado.

2.3 - Monstro

Sempre que um tetraminó começa a cair, pode aparecer o monstro, algures na zona do lado direito (à escolha), começando a movimentar-se para o lado esquerdo do ecrã, atravessando mesmo a parede (onde pode ficar um buraco). O ritmo de movimento do monstro deve ser controlado, de forma independente do ritmo de queda dos tetraminós.

Se o monstro conseguir chegar à coluna mais à esquerda do ecrã, o jogo deve ser terminado. O jogador pode destruir o monstro, fazendo cair o tetraminó de forma rápida em cima do monstro, caso em que o tetraminó é também destruído e novo tetraminó deve começar a cair.

O monstro deve aparecer apenas em cerca de 25% dos tetraminós, em média, com um ritmo de aparecimento aparentemente aleatório.

2.4 - Controlo

O sistema tem três estados possíveis:

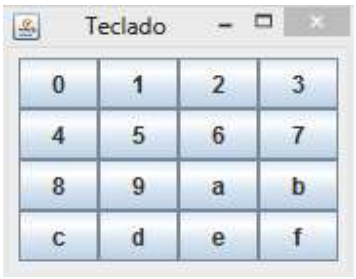
- Jogo parado, em que no ecrã se visualiza apenas a “parede” (situação ilustrada na secção 2.1). É o estado inicial após execução do programa e também pode ocorrer por o jogo ter sido terminado, quer pelo programa, quer pelo jogador;
- Jogo suspenso, devido a um comando do jogador;
- Jogo em execução.

Em termos de controlo do jogo, o jogador tem três comandos possíveis:

- Iniciar jogo. Começa um novo jogo, inicializando a pontuação a zero;
- Suspender/continuar jogo. Suspende um jogo ou continua-o, a partir do mesmo ponto em que este foi suspenso;
- Terminar jogo. Termina tudo, mas mantém a pontuação, e o ecrã deve ficar como inicialmente. Em alternativa opcional, pode usar uma mensagem sugestiva, como por exemplo "GAME OVER" (mas as letras terão de ser desenhadas no ecrã, tal como os tetraminós e o monstro).

2.5 - Teclado

O jogo é controlado por meio de teclas num teclado, tal como o da figura seguinte:



São necessários sete comandos:

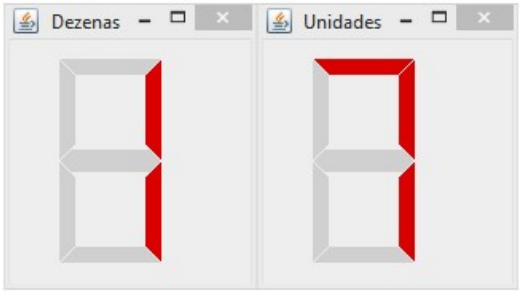
- Tetraminós: rodar, mover para a esquerda, mover para a direita, deixar cair;
- Jogo: começar, suspender/continuar, terminar.

A escolha de que tecla faz o quê é à escolha do grupo. Teclas sem função devem ser ignoradas quando premidas.

A funcionalidade de cada tecla é executada quando essa tecla é premida, mas só depois de ser largada é que pode funcionar novamente. Manter uma tecla premida não deve parar a queda dos tetraminós nem o movimento do monstro.

2.6 - Pontuação

Existem dois displays de 7 segmentos (Dezenas e Unidades), que mostram a pontuação atual do jogo (em decimal). A figura seguinte ilustra um possível valor da pontuação:



A pontuação começa com zero (00) e é incrementada de:

- 5 unidades sempre que uma linha é completamente preenchida (e eliminada);
- 2 unidades sempre que o monstro é destruído.

Se o jogo terminar, a pontuação deve manter-se. Só se um novo jogo for iniciado a pontuação deve voltar a zero. A pontuação máxima é 99. Se tal acontecer, o jogo deve ser terminado (e parabéns!).

Em alternativa aos displays de 7 segmentos, pode também visualizar a pontuação no próprio ecrã (terá de desenhar os dígitos). Este aspeto é opcional.

2.7 - Diversos

Juntamente com este documento, encontrará um ficheiro Excel (**ecra.xlsx**), que reproduz os pixels do ecrã. Pode usá-lo para desenhar/visualizar as várias representações dos tetraminós (ou mesmo tetraminós diferentes), bem como eventualmente algumas letras grandes (exemplo: GAME OVER) ou dígitos, e traduzi-los para uma tabela, de forma a produzir um jogo mais personalizado.

NOTA - Cada grupo é livre de mudar as especificações do jogo, desde que não seja para ficar mais simples e melhore o jogo em si. A criatividade e a demonstração do domínio da tecnologia são sempre indicadoras de qualidade!

3 – Faseamento do projeto

O projeto decorrerá em duas fases, versão intermédia e final.

A versão intermédia:

- Vale 20% da nota do projeto, ou 6% da nota final;
- Deve cumprir (pelo menos) o passo 1 da estratégia de implementação descrita na secção seguinte;
- Deve ser submetida no Fenix (Projeto IAC 2016-17 - versão intermédia) através de um ficheiro (grupoXX.asm, em que XX é o número do grupo) com o código do programa, tal como ele estiver na altura, até às 23h59 do dia 29 de outubro de 2016. Sugere-se criar uma cópia da versão mais recente do código, limpando eventual "lixo" e coisas temporárias, de modo a compilar e executar a funcionalidade pedida. Organização do código e comentários serão avaliados, tal como na versão final;
- Deve ser mostrada a funcionar ao docente, no laboratório. O docente poderá fazer algumas perguntas sobre o programa e dar algumas sugestões.

IMPORTANTE – Não se esqueça de identificar o código com o número do grupo e número e nome dos alunos que participaram na construção do programa (em comentários, logo no início da listagem).

A versão final do projeto deverá ser entregue até às 23h59 do dia 2 de dezembro de 2016. A entrega, a submeter no Fenix (Projeto IAC 2016-17 - versão final) deve consistir de um zip (grupoXX.zip, em que XX é o número do grupo) com dois ficheiros:

- Um relatório (modelo já disponível no Fenix);
- O código, pronto para ser carregado no simulador e executado.

4 – Estratégia de implementação

Alguns dos guiões de laboratório contêm objetivos parciais a atingir, em termos de desenvolvimento do projeto. Tente cumpri-los, de forma a garantir que o projeto estará concluído nas datas de entrega, quer na versão inicial quer na versão final.

Devem ser usados processos cooperativos para suportar as diversas ações do jogo, aparentemente simultâneas. Recomendam-se os seguintes processos:

- Teclado (varrimento e leitura das teclas, tal como descrito no guião do laboratório 4);
- Tetraminó (para controlar as ações dos tetraminós);
- Monstro (para controlar as ações do monstro);
- Gerador (para gerar um número aleatório, usado para escolher a tetraminó a cair e quando é que o monstro aparece);
- Controlo (para tratar das teclas de começar, suspender/continuar e terminar).

Como ordem geral de implementação, recomenda-se a seguinte estratégia:

1. Teclado (um varrimento de cada vez, inserido num ciclo principal onde as rotinas que implementam processos vão ser chamadas). Como teste temporário, escreva o valor da tecla nos displays quando se está a carregar numa tecla e outro valor qualquer quando não há tecla carregada. **Atualize os displays apenas quando se carregar ou libertar a tecla** (e não estar a atualizar continuamente);
2. Rotinas de ecrã (desenhar/apagar um pixel numa dada linha e coluna, desenhar/apagar tetraminós ou o monstro – represente os objetos pelas coordenadas de um determinado pixel (canto superior esquerdo, por exemplo, e desenhe-os relativamente às coordenadas desse pixel);
3. Tetraminó (desenho de um dos vários tetraminós possíveis, com deslocamentos de um pixel ou mudança de representação por cada tecla carregada no teclado);
4. Processos cooperativos (organização das rotinas preparada para o ciclo de processos);
5. Processo Controlo;
6. Interrupção de queda dos tetraminós (e alterações ao processo tetraminó para implementar a queda e a eliminação de linhas completamente preenchidas);
7. Resto das especificações, excluindo o que disser respeito ao monstro;

8. Processo Monstro (incluindo a respetiva interrupção que regula o seu movimento, bem como outras alterações ao programa que seja necessárias).

Para cada processo, recomenda-se:

- Um estado 0, de inicialização. Assim, cada processo é responsável por inicializar as suas próprias variáveis. O programa fica mais bem organizado e modular;
- Planeie os estados (situações estáveis) em que cada processo poderá estar. O processamento (decisões a tomar, ações a executar) é feito ao transitar entre estados;
- Veja que variáveis são necessárias para manter a informação relativa a cada processo, entre invocações sucessivas (posição, modo, etc.).

O processo Gerador pode ser simplesmente um contador (variável de 16 bits) que é incrementado em cada iteração do ciclo de processos. Quando for preciso um número aleatório entre 0 e 3 (para escolher um dos tetraminós possíveis ou em quais tetraminós o monstro aparece), basta ler esse contador e usar apenas os seus dois bits menos significativos. Como o ciclo de processos se repete muitas vezes durante a evolução dos tetraminós e do monstro, esses dois bits parecerão aleatórios.

Finalmente:

- Faça PUSH e POP de todos os registos que use numa rotina e não constituam valores de saída. É muito fácil não reparar que um dado registo é alterado durante um CALL, causando erros que podem ser difíceis de depurar. Atenção ao emparelhamento dos PUSHs e POPs;
- Vá testando todas as rotinas que fizer e quando as alterar. É muito mais difícil descobrir um bug num programa já complexo e ainda não testado;
- Estruture bem o programa, com zona de dados no início e rotinas auxiliares de implementação de cada processo junto a ele;
- Não coloque constantes numéricas (com algumas exceções, como 0 ou 1) pelo meio do código. Defina constantes simbólicas no início e use-as depois no programa;
- Produza comentários abundantes, não se esquecendo de cabeçalhos para as rotinas com descrição, registos de entrada e de saída;
- Não duplique código (com copy-paste). Use uma rotina com parâmetros para contemplar os diversos casos em que o comportamento correspondente é usado.

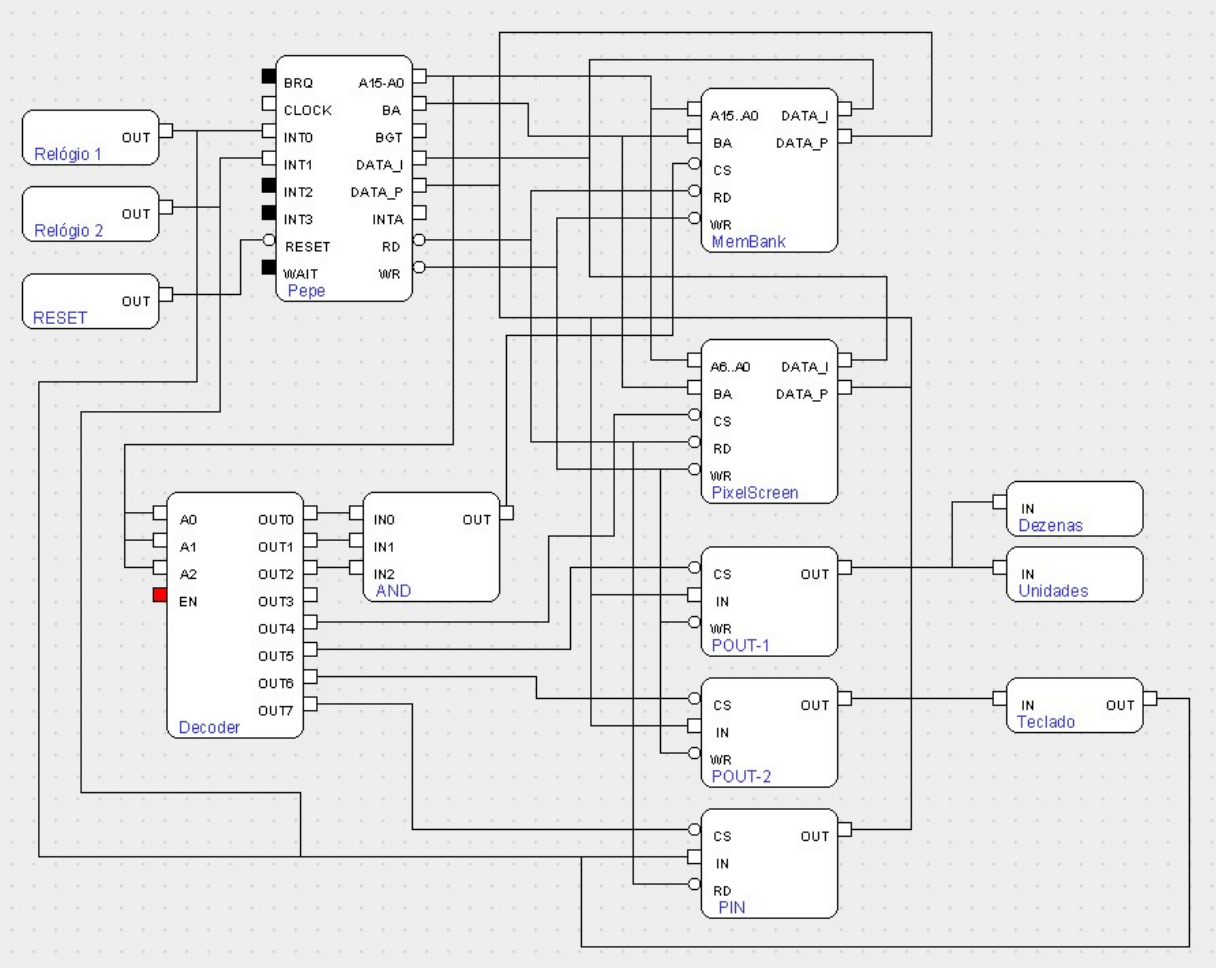
5 – Critérios de avaliação

Os critérios de avaliação e o seu peso relativo na nota final do projeto (expressos numa cotação em valores) estão indicados na tabela seguinte:

Critério	Versão intermédia	Versão final
Cumprimento da funcionalidade	2	4
Estrutura dos dados (tabelas, variáveis)	1	3
Estrutura do código (inclui processos)		3
Uso correto das interrupções		2
Qualidade dos comentários	1	2
Monstro		2
Total	4	16

6 – Implementação

A figura seguinte mostra o circuito a usar (fornecido, ficheiro **jogo.cmod**).



Os módulos seguintes têm painel de controlo em execução (modo Simulação):

- Relógio 1 – Relógio de tempo real, para ser usado como base para a temporização do movimento dos tetraminós. Este sinal também liga ao bit 4 do periférico de entrada PIN, que poderá ir lendo para animar a queda dos tetraminós numa versão anterior a usar interrupções;
- Relógio 2 – Relógio de tempo real, para ser usado como base para a temporização do movimento do monstro. Em versões intermédias pode ser útil lê-lo num ciclo de instruções. Por isso, também liga ao bit 5 do periférico de entrada PIN;
- Matriz de pixels (PixelScreen) – ecrã de 32 x 32 pixels. É acedido como se fosse uma memória de 128 bytes (4 bytes em cada linha, 32 linhas). Atenção, que o pixel mais à esquerda em cada byte (conjunto de 8 colunas em cada linha) corresponde ao bit mais significativo desse byte. Um bit a 1 corresponde a um pixel a vermelho, a 0 um pixel a cinzento;
- Dois displays de 7 segmentos, ligados aos bits 7-4 e 3-0 do periférico POUT-1, para mostrar a pontuação do jogo;
- Teclado, de 4 x 4 teclas, com 4 bits ligados ao periférico POUT-2 e 4 bits ligados ao periférico PIN (bits 3-0). A deteção de qual tecla foi carregada é feita por varrimento.

O mapa de endereços (em que os dispositivos podem ser acedidos pelo PEPE) é o seguinte:

Dispositivo	Endereços
RAM (MemBank)	0000H a 5FFFH
PixelScreen	8000H a 807FH
POUT-1 (periférico de saída de 8 bits)	0A000H
POUT-2 (periférico de saída de 8 bits)	0C000H
PIN (periférico de entrada de 8 bits)	0E000H

Notas **MUITO IMPORTANTES**:

- Os periféricos de 8 bits e as tabelas com STRING devem ser acedidos com a instrução MOVB. As variáveis definidas com WORD (que são de 16 bits) devem ser acedidas com MOV;
- A quantidade de informação mínima a escrever no PixelScreen é de um byte. Por isso, para alterar o valor de um pixel, tem primeiro de se ler o byte em que ele está localizado, alterar o bit correspondente a esse pixel e escrever de novo no mesmo byte;
- Os relógios que ligam às interrupções do PEPE e o teclado partilham o mesmo periférico de entrada, PIN (bits 4-5 e 3-0, respetivamente). Por isso, terá de usar uma máscara ou outra forma para isolar os bits que pretender, após ler este periférico durante o varrimento das teclas;
- As rotinas de interrupção param o programa principal enquanto estiverem a executar. Por isso, devem apenas atualizar variáveis em memória, que os processos sensíveis a essas interrupções devem estar a ler. O processamento deve ser feito pelos processos e não pelas rotinas de interrupção, cuja única missão é assinalar que houve uma interrupção.