Projeto de Introdução à Arquitetura de Computadores

LEIC

IST-Taguspark

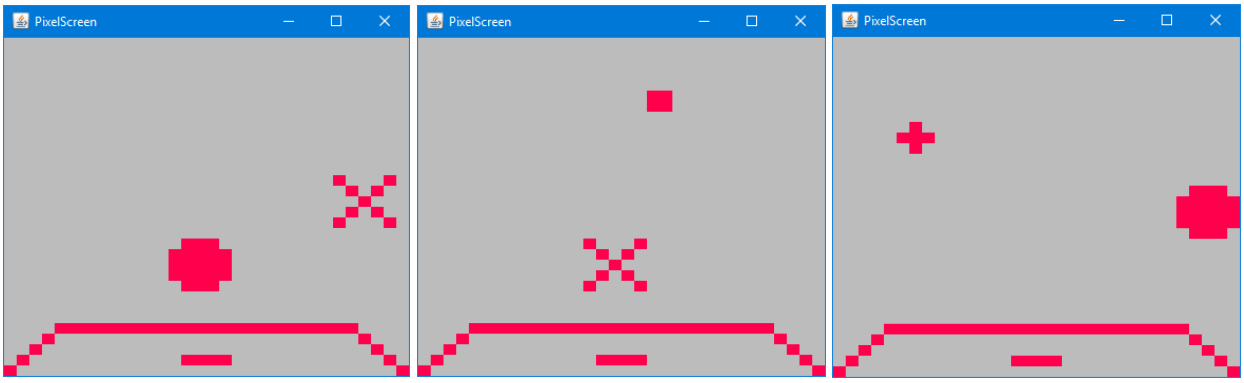
Campo de Asteroides

2018/2019

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 de simulação do voo de uma nave espacial através de um campo de asteroides. As figuras seguintes ilustram possíveis aspetos do ecrã do jogo.



No fundo de ecrã vê-se o painel de controlo da nave. Os asteroides partem de um ponto no infinito (linha de cima, mais ou menos ao meio do ecrã), muito pequenos (um pixel) e vão crescendo à medida que a nave se aproxima deles. Uns passam ao lado, quer à esquerda, quer à direita, mas outros estão mesmo em frente à nave, que colidirá com eles se nada for feito.

Há dois tipos de asteroides, bons e maus. Os bons têm minerais que a nave pode minerar e a colisão com eles faz aumentar a pontuação do jogo. Pelo contrário, a colisão com um asteroide mau destrói a nave e termina o jogo. Para o evitar, a nave pode disparar um míssil, que faz explodir o asteroide. Também pode rodar para a esquerda ou para a direita, evitando o asteroide com que iria colidir.

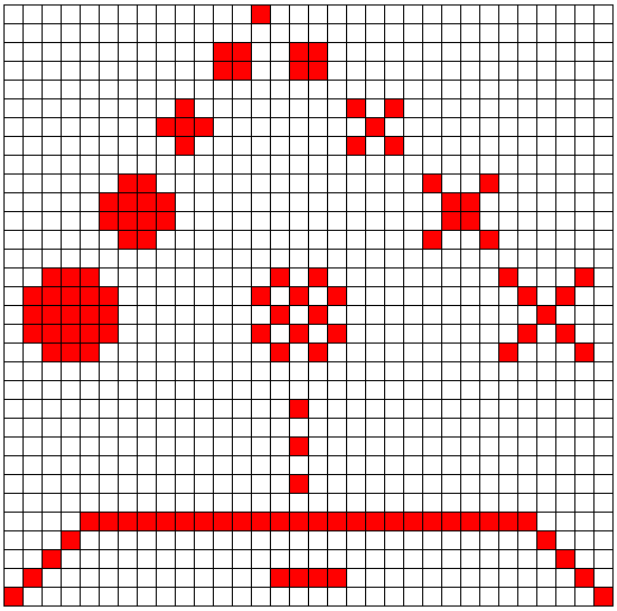
Quando um asteroide desaparece (perdendo-se pelos lados da nave, ou sendo explodido ou minerado), um novo surge muito ao longe. A direção que cada asteroide parece tomar em relação à nave (mais à esquerda, mais à direita ou mesmo em frente) é aleatória. Pretende-se que em cada momento possam existir pelo menos dois asteroides em movimento aparente no ecrã (mas com três ou mesmo quatro dá mais luta!).

2 – Especificação do projeto

2.1 – Ecrã e bonecos

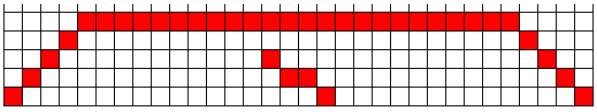
O ecrã de interação com o jogador tem 32 x 32 pixels, tantos quantas as quadrículas da figura seguinte, em que se ilustram possíveis posições dos asteroides e respetivos bonecos, a diversas distâncias. Os da esquerda são asteroides maus, os da direita são bons (note que nos dois tamanhos mais pequenos não se nota se um asteroide é bom ou mau, pois ainda está a grande distância).

Vê-se também um asteroide ao meio, em explosão (boneco igual para os asteroides bons e maus), bem como o percurso do míssil (apenas um pixel) até atingir o asteroide, altura em que explode.

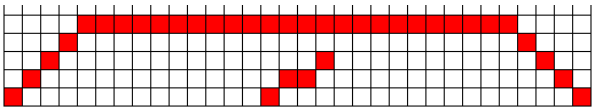


Em baixo, vê-se o painel de instrumentos da nave. A barra de direção ao meio indica se a nave está a andar a direito (situação da figura acima) ou a rodar:

- Para a direita



- Para a esquerda



Os bonecos são apenas ilustrativos e podem ser usados outros. Junto com este enunciado está um ficheiro de excel (**ecra.xlsx**) com mais algum detalhe, podendo ser usado para planear outros bonecos.

É a nave que percorre o campo de asteroides, mas como o jogador está na nave, o que vê é os asteroides a mover-se na sua direção e a crescer à medida que a nave vai ficando mais perto deles. Movem-se um pixel de cada vez, e por cada 3 movimentos crescem um tamanho, até ao máximo.

Deve haver pelo menos dois asteroides ativos e visíveis no ecrã, com um dado ritmo de evolução.

Os asteroides “nascem” com apenas 1 pixel no topo do ecrã, mais ou menos ao meio, e podem apenas deslocar-se de forma descendente, na vertical ou a 45º, para a esquerda ou para a direita. Estes últimos perdem-se pelas laterais do ecrã, mas os que descem na vertical colidirão com a nave (painel de instrumentos) se esta não fizer nada.

A nave pode disparar um míssil, que sai do meio do painel de instrumentos e se desloca na vertical ascendente, um pixel de cada vez. No entanto, tem um alcance limitado, de 12 pixels. Se até lá não atingir nenhum asteroide, perde-se. Só pode ser lançado um míssil de cada vez.

A nave pode também rodar para a esquerda ou para a direita, sendo o efeito visível (para o jogador) os asteroides e o míssil (se este estiver em movimento) somarem à sua deslocação normal na horizontal um valor fixo de pixels na direção oposta. Exemplo: um asteroide a deslocar-se na diagonal para a esquerda anda um pixel para a esquerda em cada movimento (além de outro para baixo). Se a nave começar a rodar para a direita com uma taxa de rotação de 2 pixels por cada movimento dos asteroides, então o asteroide passa a deslocar-se ainda mais para a esquerda, com 3 pixels em cada movimento (1+2). Se a nave começar a rodar para a esquerda com uma taxa de rotação de 2 pixels por cada movimento dos asteroides, então o asteroide passa a deslocar-se para a direita com 1 pixel em cada movimento (1 pixel para a esquerda de movimento normal do asteroide conjugado com 2 para a direita devido ao efeito de rotação da nave).

Se quiser ser mais realista, o efeito de rotação da nave pode depender da distância aparente dos asteroides (0 para os dois tamanhos mais pequenos, e de 1, 2 e 3 pixels para os tamanhos mais perto). É apenas simulação da perspetiva e da distância aparente.

Se houver um míssil em movimento, deve também ser afetado pelo efeito de rotação da nave (se esta estiver a rodar), com o mesmo valor de pixels usado para os tamanhos maiores dos asteroides.

2.2 – Teclado e controlo do jogo

O jogo é controlado por meio de teclas num teclado (atuado por clique do rato), tal como o da figura seguinte:

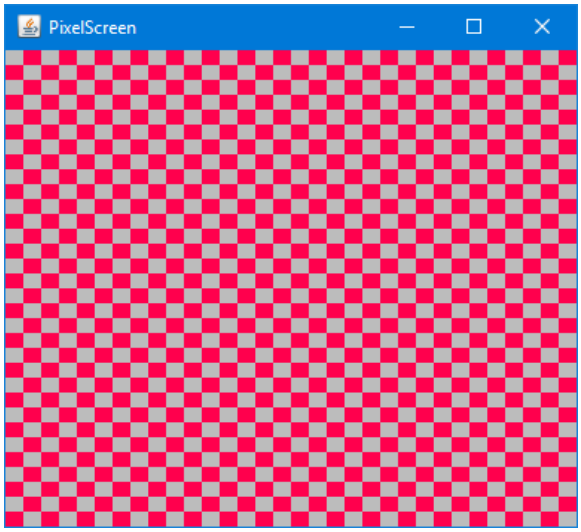
A atribuição de teclas a comandos é livre e ao seu gosto. Como exemplo, uma possível atribuição é a seguinte:

- Tecla 0: rodar a nave para a esquerda (apenas enquanto a tecla estiver a ser carregada);
- Tecla 3: rodar a nave para a direita (apenas enquanto a tecla estiver a ser carregada);
- Teclas 1 ou 2: dispara míssil;

- Tecla C: começar o jogo (deve reiniciar a pontuação);
- Tecla D: suspender/continuar o jogo;
- Tecla E: terminar o jogo (deve manter visível a pontuação obtida).

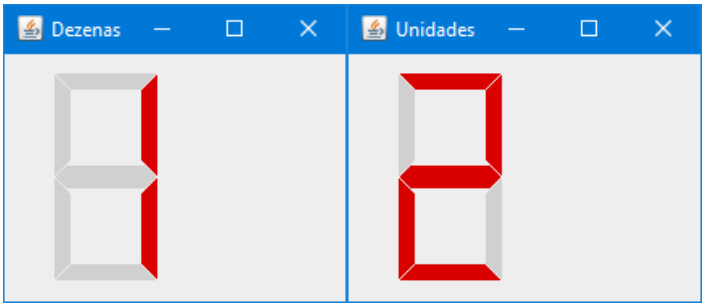
Com exceção das teclas de rodar a nave, cada tecla dever executar apenas um comando. Para novo comando, mesmo que igual, tem de se largar a tecla e carregar de novo.

Recomenda-se um ecrã diferente, como por exemplo o seguinte (mas pode ser bem mais criativo), para a situação em que o jogo está terminado:



2.3 - Pontuação

Existem dois displays de 7 segmentos (Dezenas e Unidades), que devem mostrar 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 3 unidades sempre que a nave colida com um asteroide bom.

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, parabéns! "Ganda" jogador(a)!

3 – Faseamento do projeto

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

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

Versão intermédia:

- Vale 25% da nota do projeto (ou 10% da nota final de IAC);
- Deve ser submetida no Fenix (Projeto IAC 2018-19 - versão intermédia) através de um ficheiro (**grupoXX.asm**, em que XX é o número do grupo) com o código do programa, até às 23h59 do dia 28 de outubro de 2018. 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;
- Numa aula de laboratório seguinte, o docente poderá fazer algumas perguntas sobre o programa entregue e dar algumas sugestões;
- **Deve cumprir os seguintes objetivos**:
    - Desenhar no ecrã o painel de instrumentos da nave, com a barra da direção;
    - Desenhar a barra da direção inclinada para a esquerda ou para a direita (ver secção 2.1) enquanto se carrega no teclado, na tecla correspondente (quando se larga a tecla, a barra deve voltar a ficar horizontal);
    - De cada vez que carrega numa outra tecla à sua escolha, os displays devem aumentar o seu valor de 3 unidades (contagem em decimal, entre 00 e 99, com valor inicial 00). Para aumentar de novo, a tecla tem de ser largada e carregada de novo (funcionamento diferente das teclas de controlo da barra de direção);
    - Use o circuito do projeto, **jogo.cmod** (e não o de qualquer guião de laboratório).

Versão final:

- Vale 75% da nota do projeto (ou 30% da nota final);
- **Deve cumprir todas as especificações do enunciado**;
- Deve ser submetida no Fenix (Projeto IAC 2018-19 - versão final) até às 23h59 do dia 23 de novembro de 2018, através de um ficheiro (**grupoXX.zip**, em que XX é o número do grupo) com dois ficheiros:
    - Um ficheiro **grupoXX.pdf**, relatório de formato livre, mas com a seguinte informação (pode ver o modelo de relatório disponível no Fenix, secção Laboratório):
        - Identificação do número do grupo, números de aluno e nomes;
        - Definições relevantes, se tiverem feito algo diferente do que o enunciado pede ou indica (teclas diferentes, funcionalidade a mais, etc.);

        - Indicação concreta das funcionalidades pedidas que o código enviado NÃO satisfaz;
        - Eventuais outros comentários ou conclusões.
    - Um ficheiro **grupoXX.asm** com o código, pronto para ser carregado no simulador e executado.
- Nas últimas semanas de aulas, a partir de 3 de dezembro, não haverá aulas de laboratório. Essas semanas estão reservadas para discussão do projeto com o docente das aulas de laboratório (durante o horário normal de laboratório do seu grupo). A data e local em que o seu grupo deverá comparecer serão anunciados no Fenix, após a data de entrega da versão final do projeto.

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 intermédia, quer na versão final.

Devem ser usados processos cooperativos (guião de laboratório 6) 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 3);
- Asteroide (para controlar as ações e evolução de cada um dos asteroides. Note que vários asteroides correspondem apenas a instâncias diferentes do mesmo processo, ou várias chamadas à mesma rotina com um parâmetro que indique o número do asteroide);
- Nave (para controlar a direção e disparar o míssil);
- Míssil (para controlar a evolução do míssil no espaço);
- Gerador (para gerar um número pseudo-aleatório, usado para escolher a direção que cada asteroide toma, bem como o seu tipo, bom ou mau);
- Controlo (para tratar das teclas de começar, suspender/continuar e terminar o jogo).

Como ordem geral de implementação do projeto, recomenda-se a seguinte estratégia (pode naturalmente adotar outra):

1. Teclado, displays e painel de instrumentos da nave (cobertos pela versão intermédia);
2. Rotinas de ecrã, para desenhar/apagar:
    - um pixel numa dada linha e coluna (ambas de 0 a 31);
    - um objeto genérico, descrito por uma tabela que inclua a sua largura, altura e o valor de cada um dos seus pixels. Use um desses pixels, por exemplo o canto superior esquerdo do objeto, como referência da posição do objeto (linha e coluna) e desenhe os restantes pixels relativamente às coordenadas desse pixel de referência;
3. Um só asteroide (tem de definir tabelas com a representação dos vários bonecos dos asteroides e tabelas para escolher o boneco adequado consoante o tamanho e tipo do asteroide);

4. Processos cooperativos (organização das rotinas preparada para o ciclo de processos, começando por converter em processos o código que já tem para o teclado e para a Nave);
5. Processos Controlo e Gerador;
6. Movimento dos asteroides por meio de uma interrupção (incluindo efeito da rotação da nave);
7. Processo míssil (incluindo a respetiva interrupção que regula o seu movimento e efeito da rotação da nave);
8. Deteção de colisão de um asteroide com o painel de instrumentos da nave ou com o míssil (como os asteroides são vários, é mais fácil serem eles a detetar a colisão e explodirem ou terminarem o jogo);
9. Resto das especificações, incluindo extensão para mais do que um asteroide. Esta extensão tem algumas complicações, pelo que é melhor ter um só asteroide a funcionar do que tentar logo vários e correr o risco de não conseguir nenhum.

IMPORTANTE:

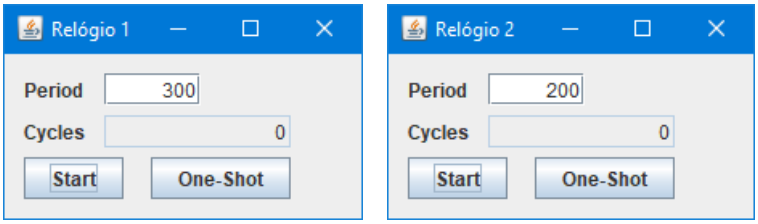
- As rotinas de interrupção param o programa principal enquanto estiverem a executar. Por isso, devem apenas atualizar variáveis em memória, que os processos sensíveis a essas interrupções devem ir lendo. O processamento deve ser feito pelos processos e não pelas rotinas de interrupção, cuja única missão é assinalar que houve uma interrupção;
- Se usar valores negativos (por exemplo, -1 para somar à coluna de um asteroide para ele se deslocar para a esquerda, ou valores de coluna negativos quando um asteroide desaparece pela lateral esquerda), as variáveis correspondentes devem ser de 16 bits (declaradas com WORD, e não STRING).

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 de cada boneco no ecrã, 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 pseudo-aleatório (para escolher a direção e tipo de um asteroide), basta ler esse contador e obter o resto da divisão desse contador por N (com a instrução MOD), obtendo-se um valor entre 0 e N-1. Como o ciclo de processos se repete muitas vezes durante a iteração dos vários processos, esse valor parecerá aleatório.

As temporizações de movimento dos asteroides e do míssil são feitas por interrupções, ligadas a dois relógios de tempo real, Relógio 1 e Relógio 2. Ajuste os períodos para o que for mais adequado.



Tenha ainda em consideração as seguintes recomendações:

- Faça PUSH e POP de todos os registos que use numa rotina e não constituam valores de saída (mas não sistematicamente de todos os registos, de R0 a R11!). 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, quer de constantes, quer de variáveis, 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;
- Como boa prática, as variáveis devem ser de 16 bits (WORD), para suportarem valores negativos sem problemas. O PEPE só sabe fazer aritmética em complemento para 2 com 16 bits;
- Produza comentários abundantes, não se esquecendo de cabeçalhos para as rotinas com descrição, registos de entrada e de saída (veja exemplos nos guiões de laboratório);
- Não duplique código (com *copy-paste*). Use uma rotina com parâmetros para contemplar os diversos casos em que o comportamento correspondente é usado.

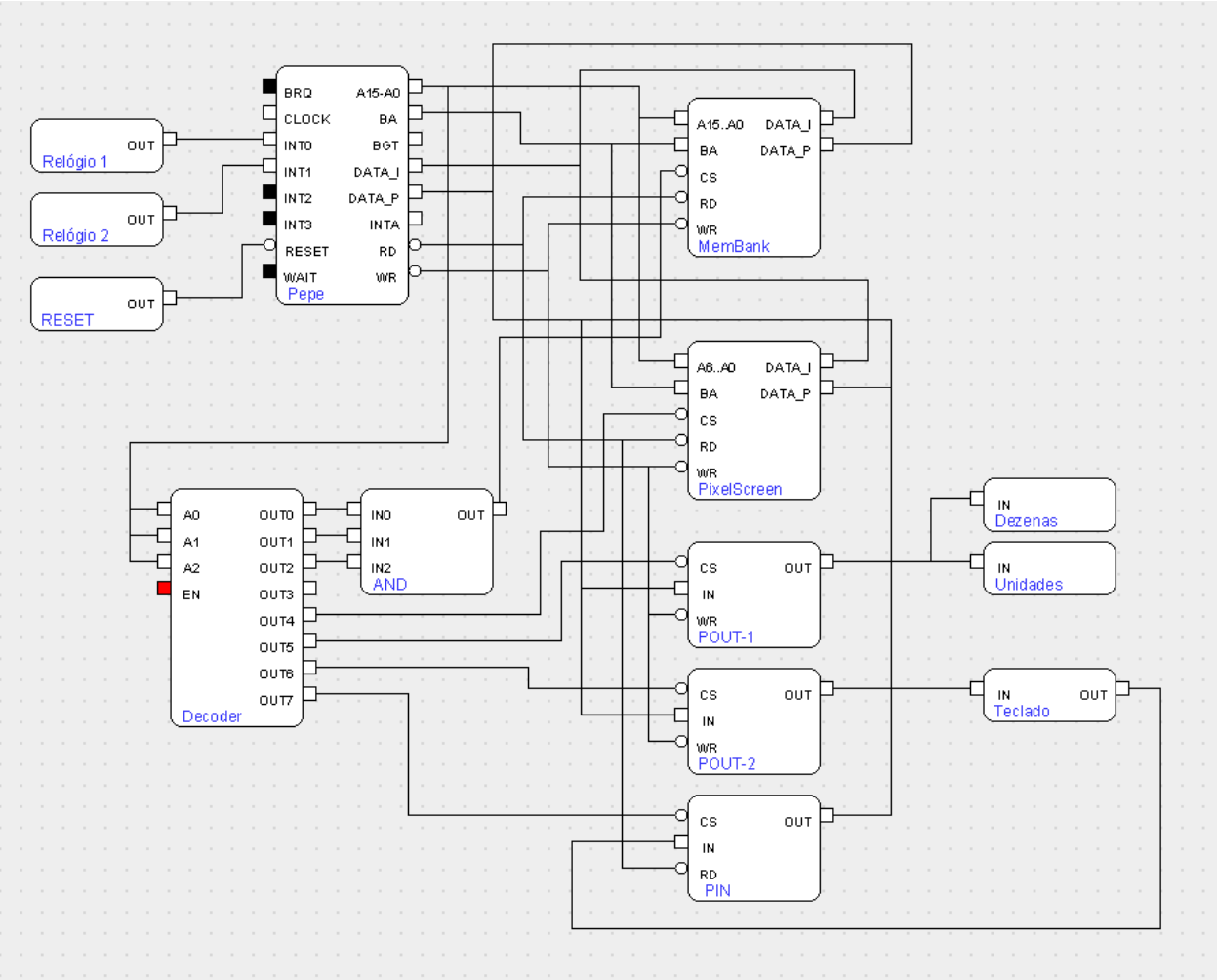
5 – Critérios de avaliação

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
Funcionalidade de base	2	6
Estrutura dos dados e do código	1	5
Comentários	1	2
Vários asteroides: funcionalidade, dados e código		3
Total	**4**	**16**

6 – Circuito do projeto

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 asteroides;
- Relógio 2 – Relógio de tempo real, para ser usado como base para a temporização do movimento do míssil;
- 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). Pode ver no ficheiro de excel **ecra.xlsx** os endereços de cada byte (relativos ao endereço de base do ecrã). 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

IMPORTANTE:

- 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 o byte alterado no mesmo endereço.