Projeto de Arquitetura de Computadores

LEE, LETI

IST-Taguspark

Batalha Naval

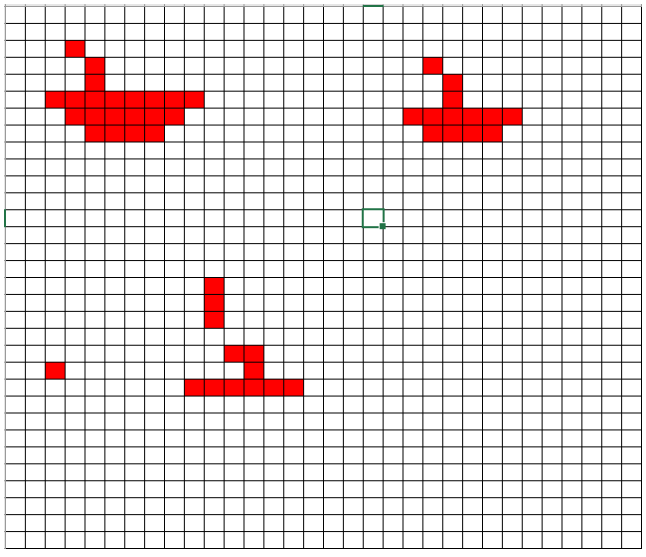
2018/2019

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 acertar nos barcos sem ser afundado.

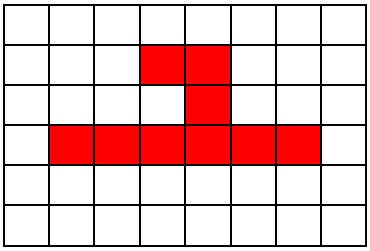
Inicia o jogo com o submarino na posição central na metade de baixo do ecrã. Vindos do lado esquerdo no terço superior do ecrã vão surgindo barcos que pertencem a um comboio de barcos. O objetivo do submarino é afundar o máximo número de barcos, lançando torpedos. No entanto o submarino pode ser afundado por balas disparadas por barcos. Estes barcos não se veem, estão fora do desenho no lado esquerdo do ecrã. Apenas se vê as balas a aparecer do lado esquerdo. Quando o submarino atinge um barco ganha um ponto. A pontuação final, mostrada num dos displays de 7 segmentos, mede a qualidade do jogador e representa os pontos obtidos a acertar nos barcos. No máximo tem dois barcos a moverem-se, um submarino, um torpedo e uma bala. Na figura seguinte exemplifica o cenário, o submarino, dois barcos, uma bala e um torpedo. O ecrã é visto em perspetiva, do ponto de vista aéreo.



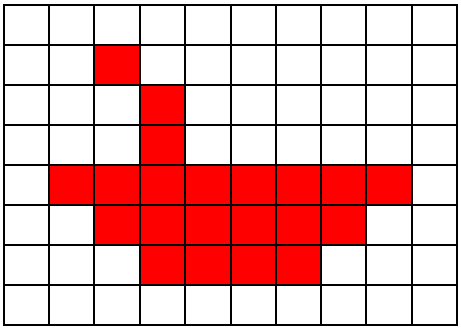
2 – Especificação do projeto

O ecrã de interação com o jogador tem 32 x 32 pixels, tantos quantas as quadrículas da figura anterior. Existem dois tipos de figuras no ecrã, as figuras que se movem automaticamente e a figura que se move recebendo direções através do teclado:

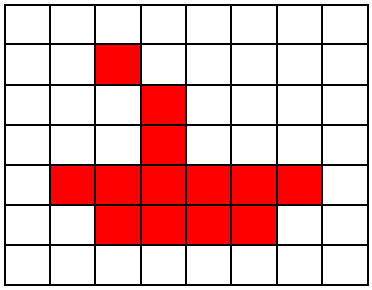
- Submarino. Neste enunciado exemplifica-se um desenho de 6 x 3 pixels. Mas deve ser fácil mudar o tamanho e aspeto do submarino, sem alterar as instruções do programa (deve ser especificado por uma tabela de pixels). A figura seguinte ilustra um submarino com apenas 6 x 3 pixels. Pode usar o submarino que entender (mesmo maior);



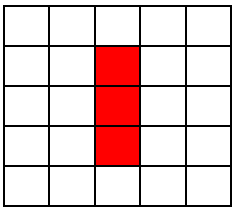
- Barco. Existem dois tipos de barcos, como mostrado nas figuras seguintes. Movem-se da esquerda para a direita no terço superior do ecrã. No máximo temos dois barcos em movimento. Podem aparecer a várias alturas. A posição exata em que aparecem deve ser aleatoriamente escolhida. Mas deve ser fácil mudar o tamanho e aspeto dos Barcos, sem alterar as instruções do programa (deve ser especificado por uma tabela de pixels). Pode usar o e criar outras Barcos (mesmo maiores). A velocidade dos Barcos é controlada pelos relógios de tempo real.
- Barco 1. Representado pela seguinte figura:



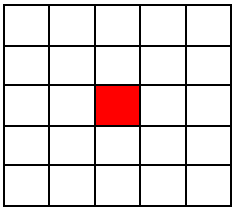
- Barco 2. Representado pela seguinte figura:



- Torpedo. Representado por um "|". Lançados pelo submarino servem para afundar os barcos.



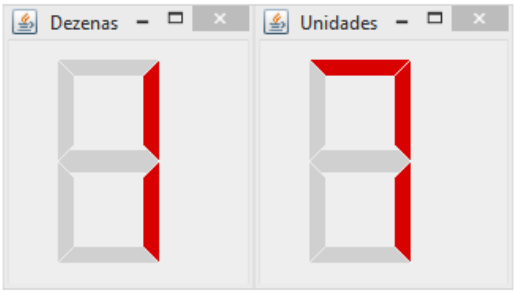
- Bala. Representada por um pixel. Veem do lado esquerdo do ecrã para o lado direito. Quando acertam no submarino este é afundado e termina o jogo. A velocidade a que se desloca é controlado pelo relógio de tempo real.



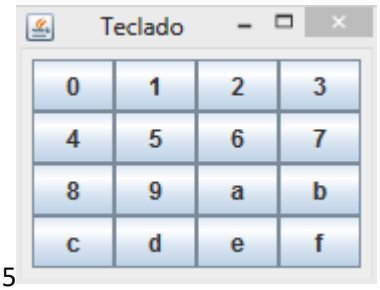
Todas as figuras devem ser fáceis de modificar o seu aspeto, sem alterar as instruções do programa (cada objeto deve ser especificado por uma tabela de pixels).

O Submarino controlado pelo teclado pode mover-se em qualquer direção (cima, baixo, esquerda, direita e direções a 45°), sob comando do jogador.

A pontuação final, mostrada num dos displays de 7 segmentos, mede a qualidade do jogador e representa os pontos obtidos por acertar nos Barcos. Considera-se que o torpedo acertou no Barco se a caixa de 6x5 do Barco 2 se sobrepor a caixa de 1x3 do Torpedo em pelo menos uma linha ou coluna.



O comando do jogo é feito por um teclado, tal como o da figura seguinte:



5

São necessários os seguintes comandos:

- Movimentar o Submarino nas 8 direções (cima, baixo, esquerda, direita e direções a 45°);
- Começar o jogo (ou recomeçar, em qualquer altura, mesmo durante um jogo);
- Terminar o jogo (para recomeçar, carrega-se na tecla de Começar).

A escolha de qual 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.

Um jogo terminado deve mostrar o ecrã em branco (todos os pixels apagados) ou outro ecrã específico à sua escolha (exemplo: FIM do JOGO). No entanto, o valor dos contadores deve ser mantido e só colocado a zero quando o novo jogo for iniciado.

Juntamente com este documento, encontrará um ficheiro Excel (**Jogo.xlsx**), que reproduz os pixels do ecrã. Pode usá-lo para desenhar novas figuras, bem como algumas letras grandes (exemplo: FIM do JOGO), e traduzi-los para uma tabela, de forma a produzir um jogo mais personalizado. Este aspeto é opcional.

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 valorizadas!

3 – Entrega do Projeto

A versão intermédia do projeto deverá ser entregue até ao dia 26 de Abril de 2019, 23h59. A entrega, a submeter no Fenix (Projeto AC 2018-19) deve consistir de um ficheiro asm com o código, pronto para ser carregado no simulador e executado (grupoXX.asm, em que XX é o número do grupo). As funcionalidades a serem apresentadas nesta parte são:

- Teclado a funcionar.
- Desenhar o Submarino no ecrã.
- Mover o Submarino no ecrã sob comando do teclado.

A versão final do projeto deverá ser entregue até ao dia 22 de Maio de 2019, 23h59. A entrega, a submeter no Fenix (Projeto AC 2018-19) 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.

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).

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 na data de entrega.

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 3);
- Submarino (para controlar os movimentos do Submarino que é controlada pelo teclado);
- Barcos (para controlar as ações dos Barcos);
- Bala (para controlar as ações das balas);
- Torpedos (para controlar as ações dos torpedos);
- Controlo (para tratar das teclas de começar e terminar).
- Gerador (para gerar um número aleatório).

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

1. Teclado (um varrimento de cada vez, inserido num ciclo principal onde as rotinas que implementam processos vão ser chamadas);
2. Rotinas de ecrã (desenhar/apagar um pixel numa dada linha e coluna, desenhar/apagar Barcos/Submarino – represente os objetos pelas coordenadas de um determinado pixel (canto superior esquerdo, por exemplo, e desenhe-os relativamente às coordenadas desse pixel);
3. Submarino (desenho da Submarino controlada pelo teclado, com deslocamentos de um pixel por cada tecla carregada no teclado);
4. Barcos (desenho dos Barcos com movimento e que aparecem numa posição aleatória);
5. Torpedos (desenho dos Torpedos disparados pelo submarino);
6. Balas (desenho das Balas com movimento e que aparecem numa posição aleatória);
7. Processos cooperativos (organização das rotinas preparada para o ciclo de processos);
8. Interrupções (para contar o tempo do contador)
9. Controlo;
10. Resto das especificações.

Para cada processo, recomenda-se:

- Um estado 0, de inicialização. Assim, (re)começar um jogo é pôr todos os processos no estado 0, em que cada um inicializa as suas próprias variáveis. Fica mais 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, direção, modo, etc.)
- O processo Barcos trata de vários Barcos independentes. Para cada um deles, recomenda-se que se invoque a mesma rotina, passando como argumento o endereço da zona de dados onde está o estado relativo a cada Barco.

O processo Gerador pode ser simplesmente um contador (variável de 16 bits) que é incrementado em cada iteração do ciclo de processos. Quando é necessário colocar um novo objecto no display, use o número aleatório para gerar a nova posição. Se precisa de gerar um número aleatório entre 0 e 3 por exemplo, basta ler esse contador e usar apenas os seus dois bits menos significativos. Como o ciclo de processos executa muitas vezes durante a execução de uma ação de lançar um objecto, esses dois bits parecerão aleatórios, do ponto de vista do lançamento do objecto, quando este acaba uma ação e vai ver que nova ação deverá executar.

Pode invocar o processo Gerador mais do que uma vez no mesmo ciclo de processos (por exemplo, entre a invocação de uma plataforma e outro) para aumentar a aleatoriedade das ações.

Finalmente:

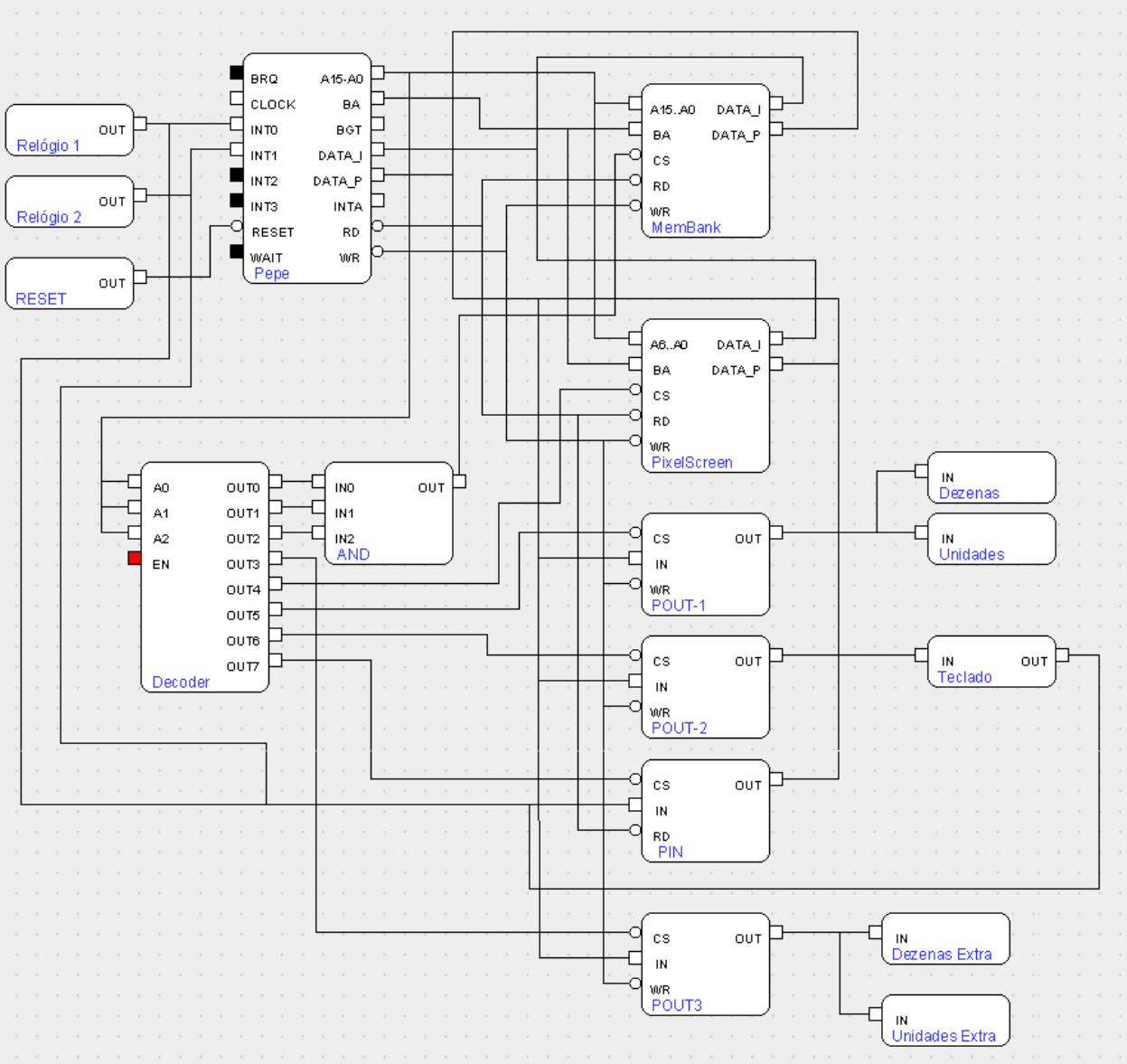
- Como norma, faça PUSH e POP de todos os registos que use numa rotina e não constituam valores de saída. É muito fácil não reparar que um dado registo é alterado durante um CALL,

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 eles;
- Não coloque constantes numéricas (com algumas exceções, como 0 ou 1) pelo meio do código. Defina constantes simbólicas 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 – Implementação

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



Podem observar-se os seguintes módulos, cujo painel de controlo deverá ser aberto em execução (modo Simulação):

- Relógio 1 – Relógio de tempo real, para ser usado como base para o movimento dos Barcos. Em versões intermédias poderá ser útil lê-lo num ciclo de instruções. Por isso, também liga ao bit 4 do periférico de entrada PIN;
- Relógio 2 – Relógio de tempo real, para ser usado como base para o movimento dos Torpedos e Balas. 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 contagem dos pontos;
- Dois displays de 7 segmentos, ligados aos bits 7-4 e 3-0 do periférico POUT-3, não usado neste projeto;
- Teclado, de 4 x 4 botões, com 4 bits ligados ao periférico POUT-2 e 4 bits ligados ao periférico PIN (bits 3-0). A deteção de qual botão está carregado 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
POUT-3 (periférico de saída de 8 bits)	06000H
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 após ativar as interrupções, terá de usar uma máscara ou outra forma para isolar os bits do teclado. Se não isolar os bits do teclado este deixará de funcionar bem após ativar as interrupções;
- As rotinas de interrupção param o programa principal enquanto estiverem a executar. Por isso, devem apenas atualizar uma 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.