Projeto de Arquitetura de Computadores

LEE, LETI

IST-Taguspark

Tiro aos Aviões

2014/2015

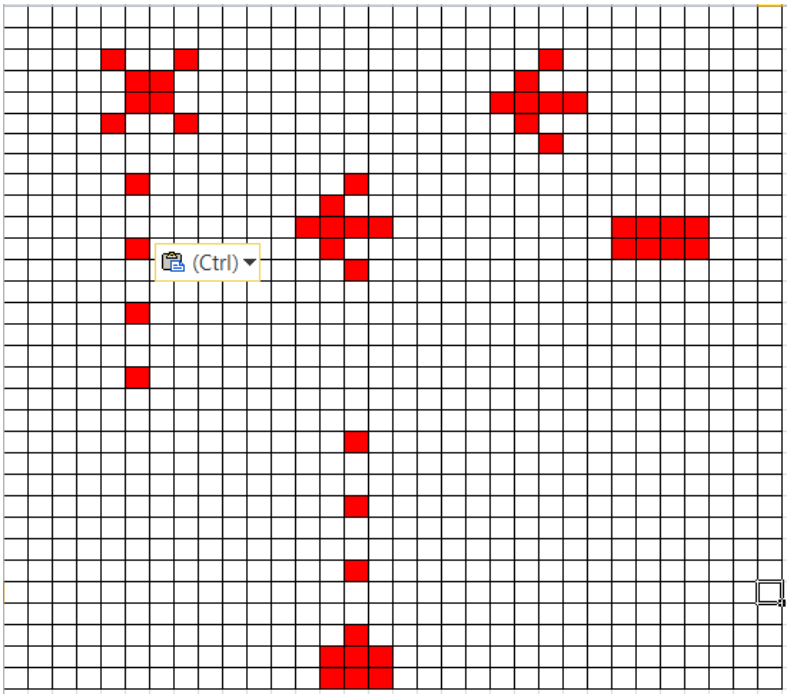
1 – Objetivos

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 de tiro aos aviões.

Vários aviões entram na parte superior do ecrã da direita para a esquerda. Um canhão na parte de baixo do ecrã dispara tiros. Se acertar no avião ganha um ponto. Quando acerta no avião este explode. Para não ficar sem balas tem de acertar em caixas de balas que andam na mesma zona dos aviões. O jogo termina quando se quiser e a pontuação final, mostrada num dos displays de 7 segmentos, mede a qualidade do atirador.

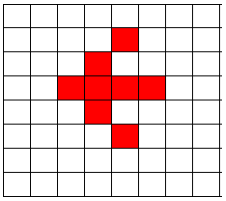
Na figura seguinte exemplifica o cenário, com o canhão, dois aviões, um avião que já foi atingido e explodiu, e uma caixa de munições que tem de acertar para não ficar sem balas.



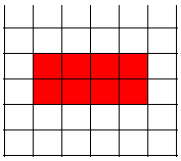
2 – Especificação do projeto

O ecrã de interação com o jogador tem 32 x 32 pixels, tantos quantas as quadrículas da figura anterior. Há cinco tipos de objetos no ecrã:

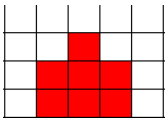
- Avião. Neste enunciado exemplifica-se um desenho de 4 x 5 pixels. Mas deve ser fácil mudar o tamanho e aspeto do avião, sem alterar as instruções do programa (deve ser especificado por uma tabela de pixels). A figura seguinte ilustra um avião com apenas 4 x 5 pixels. Pode usar o avião que entender (mesmo maior);



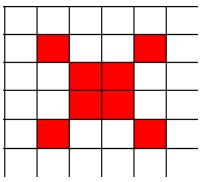
- Bala, que consiste em apenas um pixel;
- Cartucho de munições;



- Canhão, que dispara as balas. Neste enunciado exemplifica-se um boneco de 3 x 3 pixels, mas as suas dimensões e aspeto devem também ser definidos por uma tabela de pixels e não diretamente por instruções.



- Explosão. Sempre que um avião é atingido por uma bala, o simbolo deve mudar para uma explosão;



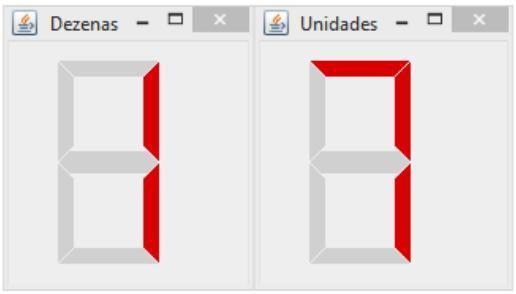
Todos os objectos devem ser fáceis de modificar o seu aspeto, sem alterar as instruções do programa (cada objeto deve ser especificado por uma tabela de pixels).

O canhão pode mover-se em qualquer direção (cima, baixo, esquerda, direita e direções a 45°), sob comando do jogador. O objetivo é acertar nos aviões. Balas que ultrapassam a parte superior do ecrã desaparecem. Não pode sair do ecrã nem invadir o topo do ecrã, onde os aviões se movimentam. O canhão só se pode movimentar na metade inferior do ecrã.

Cada avião ou cartucho de balas entra pelo lado direito do ecrã e sai pelo lado esquerdo se não for atingido. O tempo em que cada avião ou cartucho entra deve ser baseado num gerador de números aleatórios (ver secção 4, Estratégia de Implementação). Os aviões ou os cartuchos de balas podem entrar em duas linhas a alturas diferentes a contar do topo do ecran. A entrada de um avião ou cartucho de balas por uma linha ou outra é escolhido aleatoriamente.

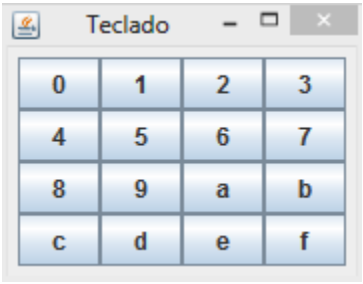
Considera-se que uma bala atinge um avião ou cartucho de balas se no seu percurso a sua posição no ecrã coincidir com um dos pixels do avião ou cartucho. Considera-se perdida a bala quando atinge o topo superior do ecrã. Quando atinge o avião deve ser actualizada a pontuação.

Existem dois displays de 7 segmentos, que mostram o conteúdo de um contador de dois dígitos decimais (hexadecimal não), que é incrementado sempre que o canhão atinge um avião. O valor inicial do contador é zero, com máximo 99. A figura seguinte ilustra um possível valor desse contador.



Existe outro conjunto de dois displays de 7 segmentos, para guardar a contagem das munições disponíveis. Quando atinge o cartucho de munições a contagem deve ser incrementada com o novo valor de munições. Sempre que dispara uma bala a contagem deve ser decrementada.

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



São necessários os seguintes comandos:

- Movimentar o canhão 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 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.

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

Juntamente com este documento, encontrará um ficheiro Excel (**ecra.xlsx**), que reproduz os pixels do ecrã. Pode usá-lo para desenhar novos aviões, canhões, explosões ou cartuchos de balas, bem como algumas letras grandes (exemplo: GAME OVER), 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 final do projeto deverá ser entregue até ao dia 20 de Maio de 2015, 23h59. A entrega, a submeter no Fenix (Projeto AC 2014-15) 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);
- Canhão (para controlar os movimentos do canhão);
- Aviões (para controlar as ações dos aviões e dos cartuchos das balas);
- Tiro (para controlar o movimento das balas e atualizar o contador e os displays);
- 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. Rotinas de ecrã (desenhar/apagar um pixel numa dada linha e coluna, desenhar/apagar canhão/avião – represente os objetos pelas coordenadas de um determinado pixel (canto superior esquerdo, por exemplo, e desenhe-os relativamente às coordenadas desse pixel);

2. Teclado (um varrimento de cada vez, inserido num ciclo principal onde as rotinas que implementam processos vão ser chamadas);
3. Canhão (desenho do canhão, com deslocamentos de um pixel por cada tecla carregada no teclado);
4. Processos cooperativos (organização das rotinas preparada para o ciclo de processos);
5. Bala (pode começar só com uma, com deslocamento de um pixel numa dada direção quando se carrega numa dada tecla; mais tarde, de forma autónoma com interrupções);
6. Avião (pode começar só um, com deslocamento de um pixel num dado sentido horizontal quando se carrega numa dada tecla; mais tarde, de forma autónoma com interrupções);
7. Interrupções (há duas, uma para animar o movimento das balas e outra para animar o movimento dos aviões);
8. Controlo;
9. 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.)
- Quer o processo Bala quer o processo Avião trata de vários objetos 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 objeto (ou o nº da bala ou avião que depois indexa uma tabela com esses dados).

O processo Gerador é mais simples e 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 avião no display, use o número aleatório para gerar um atraso aleatório. 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 avião, esses dois bits parecerão aleatórios, do ponto de vista do lançamento do avião, quando este acaba uma ação e vai ver que nova deverá executar.

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

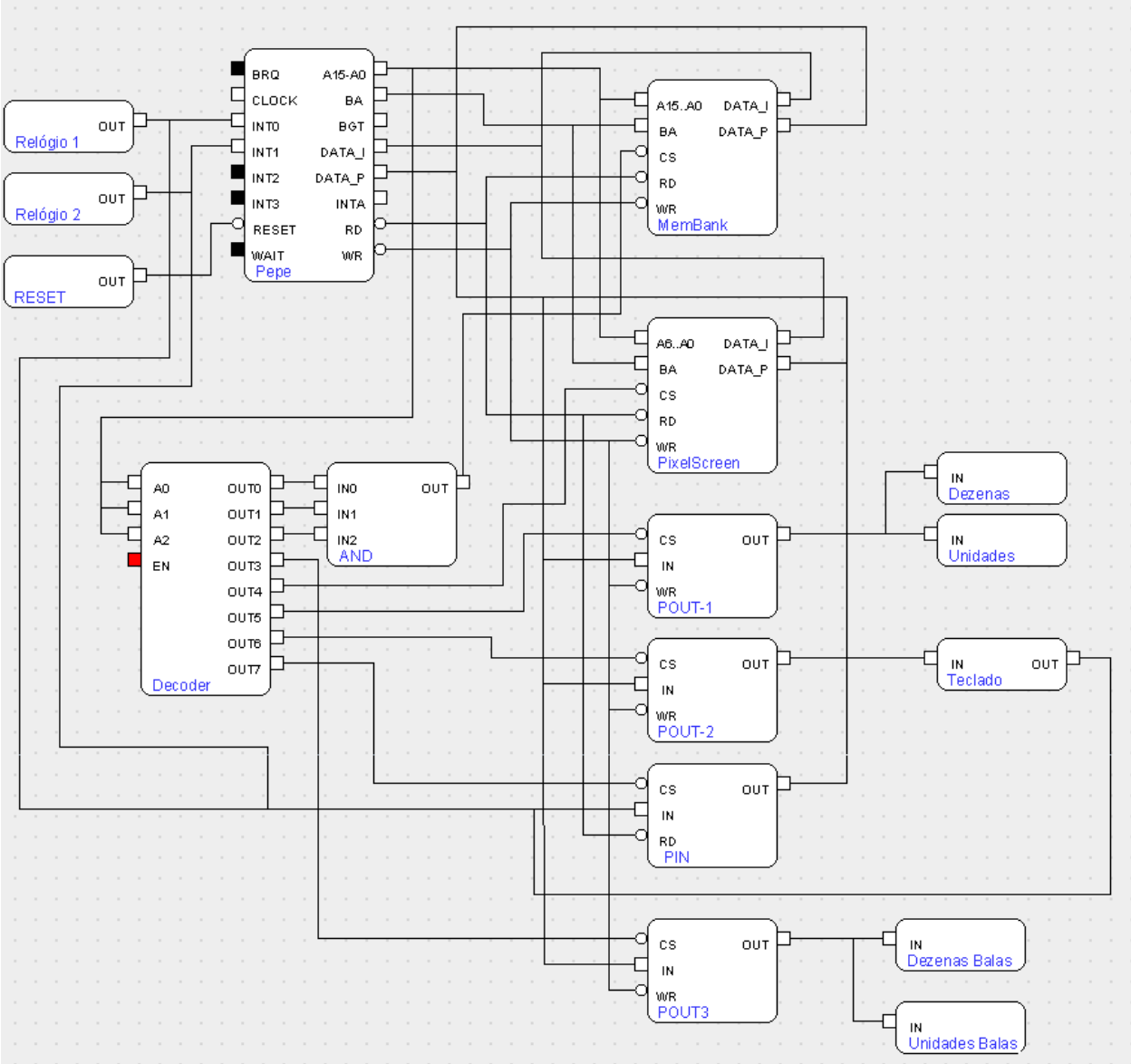
Finalmente:

- 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 a temporização do movimento das balas. 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 a temporização do movimento dos aviões e cartuchos das 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 o contador de bolas apanhadas;
- Dois displays de 7 segmentos, ligados aos bits 7-4 e 3-0 do periférico POUT-3, para mostrar o contador de balas disponíveis;
- 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, terá de usar uma máscara ou outra forma para isolar os bits que pretender, após ler este periférico;
- 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.