Projecto de Arquitectura de Computadores
Ano lectivo 2023/2024

Departamento de Ensino e Investigação de Ciências da Computação
Faculdade de Ciências Naturais da Universidade Agostinho Neto

Sumário

Objectivo

O objectivo deste projecto é de exercitar os fundamentos da área de Arquitectura de Computadores, nomeadamente a programação em linguagem assembly, os periféricos e as interrupções.

Introdução

Pretende-se criar um jogo, que consiste em um Boneco apanhar Objectos que andam no ecrã. O Boneco obedece aos comandos do jogador.

O jogo tem início com um Boneco no centro do ecrã. O Objecto começa em qualquer ponto da borda do ecrã e move-se a 45 graus em relação à borda do ecrã. Este Objecto tem de ser apanhado pelo Boneco. Quando o Boneco apanha um Objecto ganha um ponto. A pontuação final, mostrada em um dos displays de 7 segmentos, mede a qualidade do jogador e representa os pontos obtidos por apanhar os Objectos. Após apanhar um Objecto desenha um novo Objecto aleatoriamente num dos lados do ecrã e que se vai mover a 45 graus. No máximo tem dois Objetos a moverem-se e um Boneco. Se os dois Objectos chocarem passam um pelo outro sem interferirem no seu andamento.

Na figura seguinte exemplifica o cenário, o Boneco e dois Objectos a moverem-se a 45 graus. Os objectos são mostrados em sucessivas posições apenas para ilustração. Mas quando desenhar os Objectos sempre que desenha o Objecto numa nova posição apaga o desenho da antiga posição.

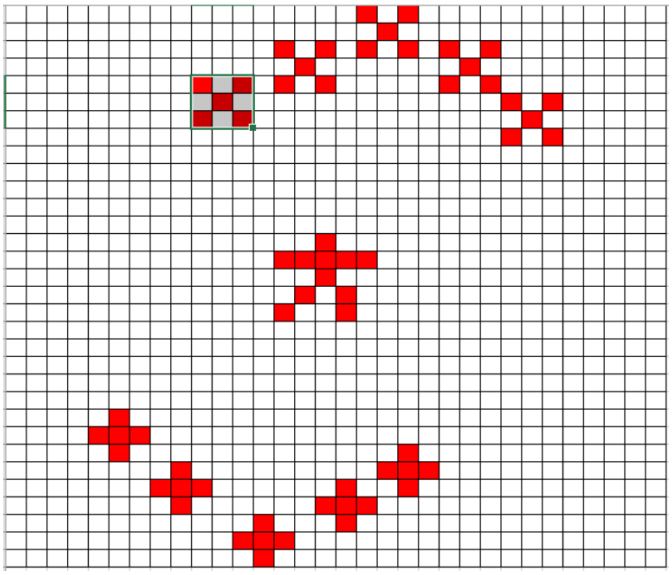

Figura 1: Ilustração do cenário do jogo.

Especificação do Projecto

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 estáticas e a figura que se move recebendo direcções através do teclado:

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

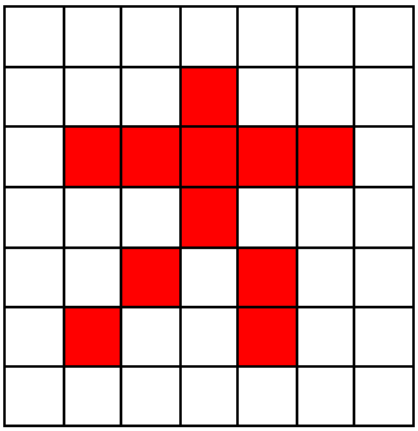

Figura 2: Exemplo de boneco

- **Objectos**. Existem quatro objectos diferentes, escolha qual lançar aleatoriamente. Neste enunciado exemplifica-se um desenho de 3 x 3 pixels. Movem-se a 45 graus em relação aos lados do ecrã. Podem-se mover no sentido dos ponteiros do relógio ou ao seu contrário, escolha um sentido aleatoriamente. Quando batem numa parede fazem ricochete e movem-se noutra direcção a 45 graus novamente. Mas deve ser fácil mudar o tamanho e aspecto dos Objectos, sem alterar as instruções do programa (deve ser especificado por uma tabela de pixels). Pode usar ou/e criar outros Objectos (mesmo maiores). A velocidade dos Objectos é controlada pelos relógios de tempo real, um para cada Objecto.
    - **Objecto 1**. Representado por "traço"

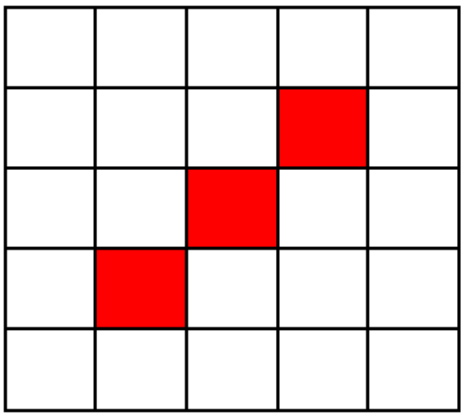

Figura 3: Modelo de Objecto

- **Objecto 2**. Representado por um “L”.

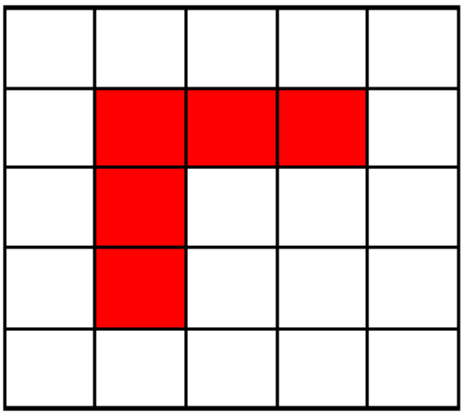

Figura 4: Modelo de Objecto

- **Objecto 3**. Representado por um “x”.

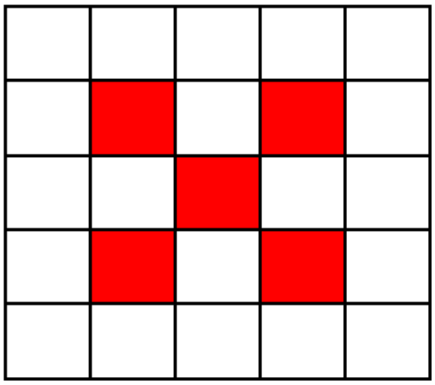

Figura 5: Modelo de Objecto.

- **Objecto 4**. Representado por um “+”.

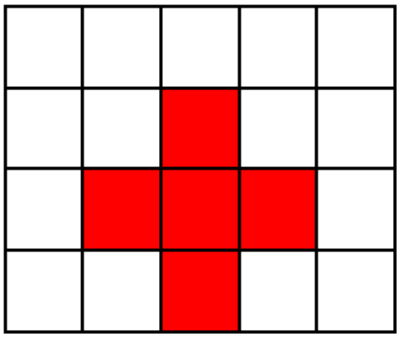

Figura 6: Modelo de Objecto.

Todas as figuras devem ser fáceis de modificar o seu aspecto, sem alterar as instruções do programa (cada objecto deve ser especificado por uma tabela de pixels, com excepção da caixa do centro).

O Boneco controlado pelo teclado pode mover-se em qualquer direcção (cima, baixo, esquerda, direita e direcções 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 apanhar Objectos. Considera-se que o boneco apanhou o Objecto se a caixa de 5x5 do Boneco se sobrepor a caixa de 3x3 do Objecto em pelo menos uma linha ou coluna.

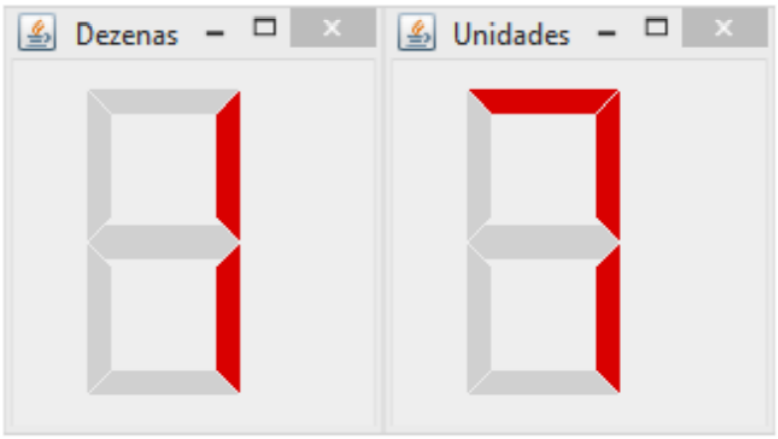

Figura 7: Displays de 7 segmentos para apresentar os pontos do jogador.

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

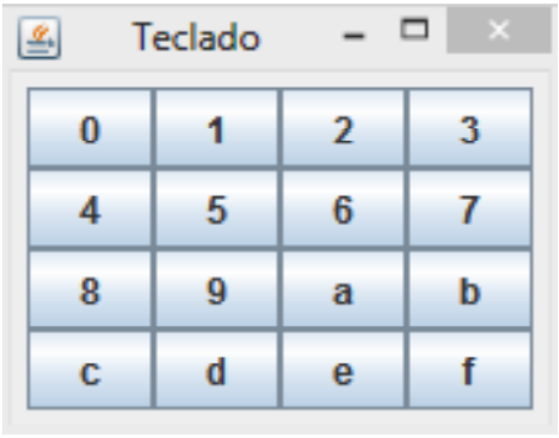

Figura 8: Matriz de pressão (teclado para entrada de comandos pelo jogador).

São necessários os seguintes comandos:

- Movimentar o Boneco nas 8 direcções (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: 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 (ac2023-projecto-design.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 aspecto 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!

Entrega do Projecto

A versão intermédia do projecto deverá ser entregue até ao dia 26 de Janeiro de 2024, 23h59. A entrega, a submeter no GitHub (repositório previamente partilhado com o docente) 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 a ser atribuído pelo docente) e o ficheiro README.md adequado. As funcionalidades a serem apresentadas nesta parte são:

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

A versão final do projecto deverá ser entregue até ao dia 24 de Fevereiro de 2024, até às 23h59. A entrega, a submeter no email da disciplina (fcuan.doc@gmail.com) deve consistir de um zip (grupoXX.zip, em que XX é o número do grupo) com dois ficheiros:

- Um relatório (modelo disponível brevemente no repositório), ficheiro pdf;
- O código, pronto para ser carregado no simulador e executado, ficheiro asm.

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

Estratégia de Implementação

Alguns dos guias de laboratório contêm objectivos parciais a atingir, em termos de desenvolvimento do projecto. Tente cumpri-los, de forma a garantir que o projecto estará concluído na data de entrega.

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

- Teclado (varrimento e leitura das teclas, tal como descrito no guia do laboratório);
- Boneco (para controlar os movimentos do Boneco que é controlado pelo teclado);
- Objectos (para controlar as acções dos Objetos);
- Controlo (para tratar das teclas de começar e terminar).
- Gerador (para gerar um número aleatório).

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

- Rotinas de ecrã (desenhar/apagar um pixel numa dada linha e coluna, desenhar/apagar objectos/Boneco – represente os objectos pelas coordenadas de um determinado pixel (canto superior esquerdo, por exemplo, e desenhe-os relativamente às coordenadas desse pixel);
- Teclado (um varrimento de cada vez, inserido num ciclo principal onde as rotinas que implementam processos vão ser chamadas);
- Boneco (desenho do Boneco controlado pelo teclado, com deslocamentos de um pixel por cada tecla carregada no teclado);
- Objectos (desenho dos Objectos com movimento a 45 graus e que aparecem numa posição aleatória);
- Processos cooperativos (organização das rotinas preparadas para o ciclo de processos);
- Interrupções (para contar o tempo do contador)
- Controlo;
- 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, acções 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, direcção, modo, etc.)
- O processo objectos/plataforma trata de vários objectos 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 objecto.

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 acção 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 acção 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 acções.

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 excepções, 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.

Implementação

A figura seguinte mostra o circuito a usar (fornecido, ficheiro ac2023-projecto-circuito.cmod).

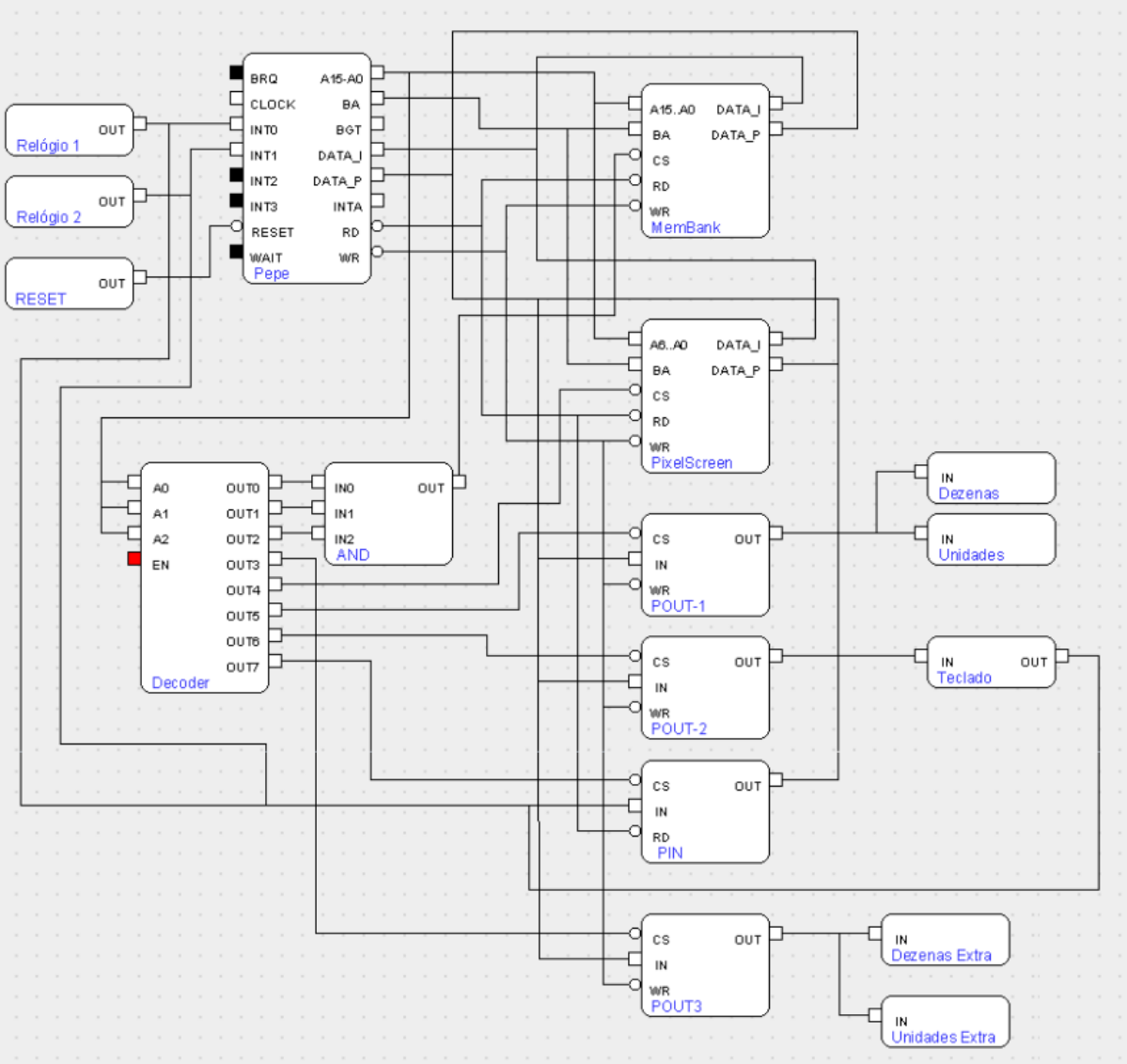

Figura 9: Circuito do projecto.

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 do primeiro Objecto. 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 do segundo Objecto. 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 projecto;
- 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 detecção 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:

Tabela 1: Mapa de endereços.

Dispositivo	**Endereçamento**
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 actualizar uma variável 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.

Bom trabalho!