Projeto de Arquitetura de Computadores

LEE/LETI

IST-Taguspark

Pac-Man modificado

2023/2024

1 – Objetivos

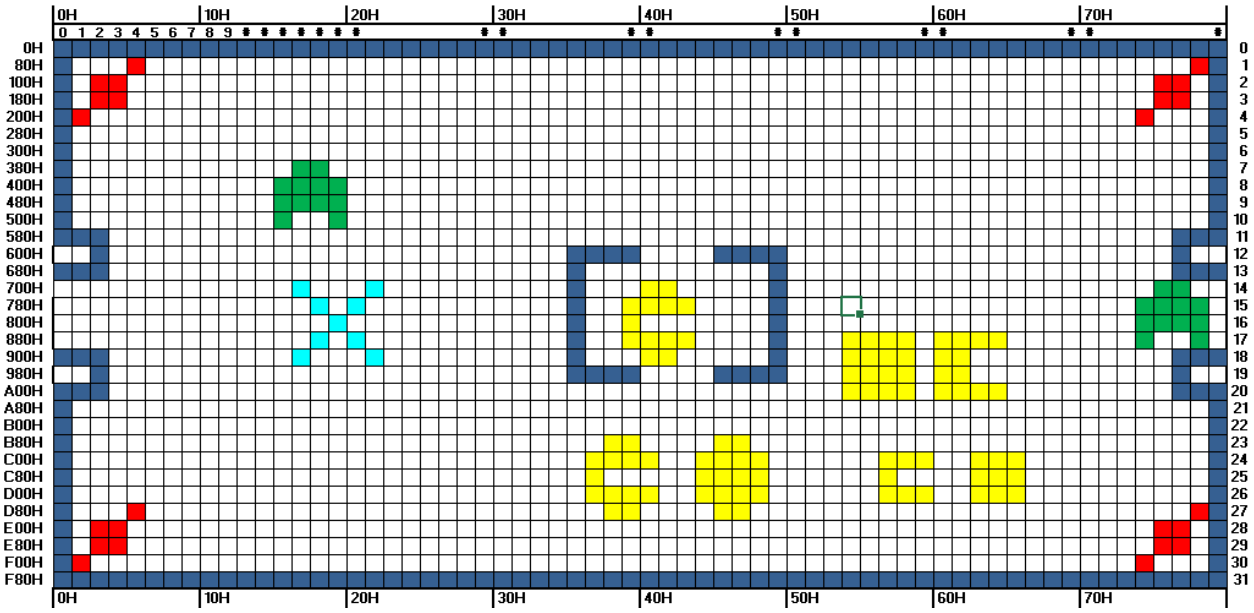
Este projeto pretende exercitar os fundamentos da área de Arquitetura de Computadores, nomeadamente a programação em linguagem *assembly*, os periféricos e as interrupções.

O objetivo deste projeto consiste em concretizar um jogo inspirado no Pac-Man .

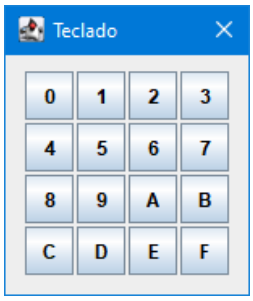
O Pac-Man movimenta-se no ecrã e tem por objectivo apanhar os quatro objectos (rebuçados) que se encontram nos quatro cantos, e ao mesmo tempo não ser apanhado pelos fantasmas representados por uns bonecos verdes. O jogo termina quando o Pac-man é apanhado pelos fantasmas (perde o jogo) ou quando apanha os 4 objectos dos cantos (ganha o jogo). A pontuação final, mostrada num dos displays de 7 segmentos, mede a qualidade do jogador e representa o tempo em segundos até apanhar os quatro objectos.

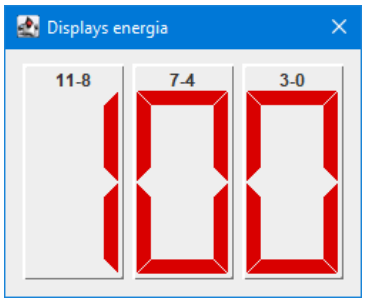
A interface do jogo consiste num ecrã, um teclado para controlo/navegação e um conjunto de displays, para mostrar o tempo decorrido no jogo.

Na figura seguinte exemplifica o cenário, o Pac-Man, quatro objetos nos cantos, dois fantasmas a verde e uma zona azul ondem nasce o Pac-man. Os fantasmas entram no ecran pela abertura esquerda ou direita. Para alem disso tem a ciam uma explosão, para a colisão entre o Pac-Man e os fantasmas.



O módulo MediaCenter do simulador possui variadas capacidades multimédia, permitindo definir imagens de fundo, reproduzir sons e vídeos, vários planos de imagens construídas pixel a pixel, um cenário frontal para letreiros, etc. O guião de laboratório 3 fornece mais detalhes sobre este módulo.





O teclado permite, fazendo clique em algumas das teclas, fazer o comando do jogo (*start*, *pause* e *stop*, para além de controlar o movimento do Pac-man).

Os displays permitem exibir o tempo decorrido desde o inicio do jogo.

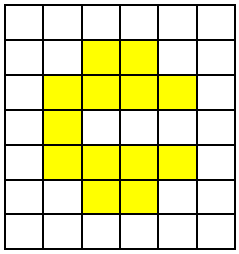
Cada estado do jogo (inicial, a jogar, em pausa, terminado, etc.) deve ter um cenário ou vídeo de fundo diferente, ou um letreiro (cenário frontal, uma imagem com letras e fundo transparente), de forma a dar uma indicação visual do estado do jogo. Também pode ser um vídeo, com as letras sobrepostas por meio de um cenário frontal (com fundo transparente).

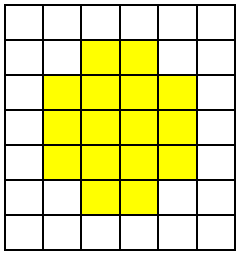
2 – Detalhes do projeto

2.1 – Objectos

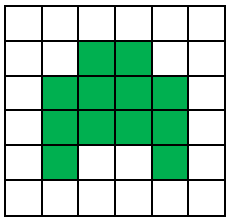
O ecrã de interação com o jogador tem 64 x 32 pixels, tantos quantas as quadrículas da figura anterior. Há quatro tipos de objetos no ecrã:

- Pac-Man. Neste enunciado exemplifica-se um desenho de 4 x 5 pixels. Mas deve ser fácil mudar o tamanho e aspeto do Pac-Man, sem alterar as instruções do programa (deve ser especificado por uma tabela de pixels). A figura seguinte ilustra um Pac-Man com apenas 4 x 5 pixels. Pode usar o Pac-Man que entender (mesmo maior). Quando o Pac-Man esta parado fica com a boca aberta, quando está em movimento abre e fecha a boca.

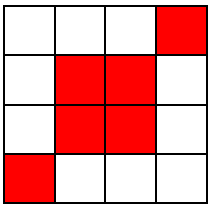




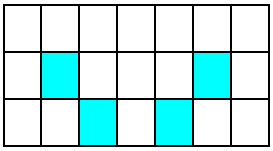
- Fantasma. Neste enunciado exemplifica-se um boneco de 4 x 4 pixels, mas as suas dimensões e aspeto devem também ser definidos por uma tabela de pixels e não diretamente por instruções.

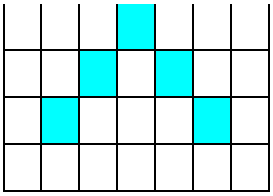


- Objectos do Canto. Representado por um rebuçado.

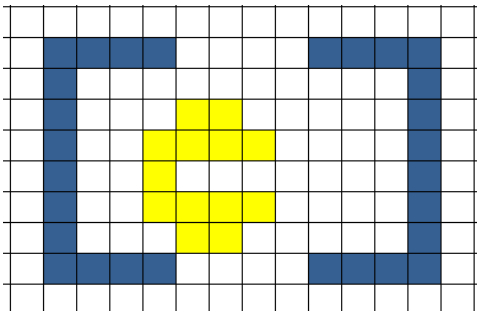


- Explosão quando houver uma colisão entre o fantasma e o Pac-Man, que dura um certo período de tempo.





- E por fim é colocada uma caixa no centro do ecrã de onde nascem os Pac-man. A zona a azul não podem ser atravessadas pelos fantasmas nem pelo pac-man.



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

O Pac-Man e os Fantasmas podem mover-se em qualquer direção (cima, baixo, esquerda, direita e direções a 45°), sob comando do jogador o primeiro e automaticamente o segundo.

O Pac-man nasce no centro do ecrã dentro da caixa, movem-se para saírem da caixa central.

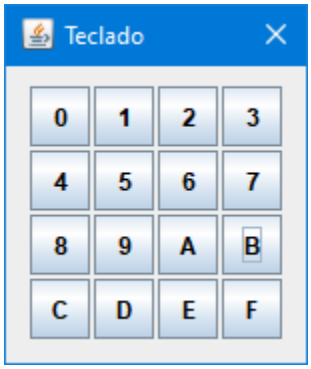
Os fantasmas entram no ecran pela abertura esquerda ou direita (linha azul). A partir dai os fantasmas evoluem de forma autónoma e automática, reduzindo de uma unidade, em cada evolução, a diferença em linha e coluna da sua posição face à posição do Pac-Man. Se o fantasma chocar com o Pac-Man, há uma colisão, e o jogo acaba. O ritmo de evolução dos fantasmas deve ser programável (sugere-se 3 a 5 segundos de período).

O tempo em que cada fantasma entra deve ser baseado num gerador de números aleatórios (ver secção 4, Estratégia de Implementação). O número de fantasmas deve poder ser alterado (por uma constante no programa) entre 0 e 4.

O Pac-Man e os Fantasmas não podem ir para cima da caixa central.

2.2 – Teclado e controlo do jogo

O jogo é controlado pelo utilizador 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:

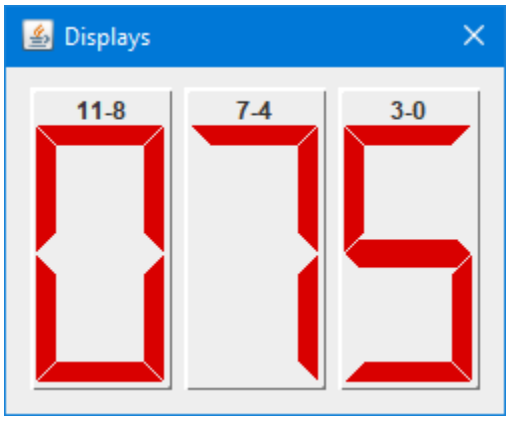
- Teclas 0, 1, 2: movimentar a Pac-Man para cima-esquerda, cima, cima-direita, respetivamente;
- Teclas 4 e 6: movimentar a Pac-Man para a esquerda e direita, respetivamente;
- Teclas 8, 9, A: movimentar a Pac-Man para baixo-esquerda, baixo, baixo -direita, respetivamente;
- Tecla C: começar o jogo;
- Tecla D: suspender/continuar o jogo;
- Tecla E: terminar o jogo (deve manter visível a pontuação).

IMPORTANTE - Com exceção das teclas de movimentar a Pac-Man, cada tecla dever executar apenas um comando. Para novo comando, mesmo que igual, tem de se largar a tecla e carregar de novo. O movimento da Pac-Man pode ser em "contínuo", enquanto a tecla estiver carregada.

O guião de laboratório 3 ensina a trabalhar com o teclado.

2.2 – Displays

Existem um display de 7 segmentos, que deve mostrar a pontuação do jogo que consiste no tempo decorrido até apanhar os quatro objetos do canto (em decimal, o que implica conversão a partir dos valores hexadecimais que o PEPE usa). A figura seguinte ilustra um possível valor após o jogo evoluir:



O guião de laboratório 3 ensina a trabalhar com os displays.

2.3 – Ecrã, bonecos, cenário e sons

O ecrã de interação com o jogador tem 32 x 64 pixels (linhas x colunas). Cada pixel pode ter uma cor diferente, em 4 componentes (Alpha, Red, Green e Blue, ou ARGB), todas com 4 bits (valores sem sinal, de 0 a F). A primeira componente define a opacidade (0 é totalmente transparente, F totalmente opaco). O pixel pode estar ligado (com a cor definida pelas 4 componentes) ou desligado (com tudo a zero, caso em que não se vê pois fica transparente).

Pac-man e fantasmas são bonecos desenhados pixel a pixel.

Por trás deste ecrã de pixels está a imagem ou vídeo de fundo (a que se vê nas imagens anteriores), que tipicamente tem uma resolução bem superior (embora deva ter um fator de forma semelhante, retangular 2 x 1, para que não apareça de forma distorcida).

É possível alterar a imagem/vídeo de fundo através do programa do PEPE. Por isso, espera-se que use um cenário diferente para cada situação. Nem sempre uma imagem tem de variar. Pode editá-la, colocando texto que indique qual a situação (pausa, por exemplo), gerando assim variantes da mesma imagem. Ou pode usar um cenário frontal, com uma imagem com as letras e fundo transparente, que aparece à frente da imagem ou vídeo de fundo. Há também comandos que o programa pode dar para ligar e desligar este cenário frontal.

Não é fornecida nenhuma imagem nem nenhum vídeo para cenários, mas existem inúmeras imagens adequadas que pode obter de forma gratuita na Web para este uso pessoal. O design multimédia fica ao seu gosto.

Também devem existir efeitos sonoros, que mais uma vez podem ser obtidos na Web. Ficheiros de som pequenos é o ideal. O módulo MediaCenter permite, no entanto, restringir os tempos de início e fim de um som (ou de um vídeo), caso haja necessidade.

Estes efeitos sonoros devem ser reproduzidos quando é há uma colisão, quando se apanha um rebuçado etc. Isto é um jogo!

Desenhar um boneco no ecrã (à frente do cenário de fundo) é desenhar os seus pixels, com cor ou desligado (transparente), em posições (linha e coluna) adjacentes do ecrã, de acordo com a forma definida para esse boneco.

Toma-se um dos pixels do boneco (canto superior esquerdo, por exemplo) como indicador da posição no ecrã desse boneco e todos os restantes pixels desse boneco são desenhados em posições relativas a esse pixel de referência.

Mover um boneco é apagá-lo na sua posição atual, mudar a sua posição e desenhá-lo na nova posição. O efeito visual é o boneco parecer mover-se.

O ficheiro Excel **ecrã-32x64.xlsx**, que é fornecido, simula as quadrículas do ecrã e mostra que o ecrã é na realidade uma memória (note os endereços de cada linha no lado esquerdo), em que cada pixel é uma palavra (2 bytes) dessa memória, com a cor do pixel. Há 64 pixels, ou 128 bytes, por cada linha do ecrã.

Pode usar este ficheiro para planear o tamanho e forma dos seus bonecos, pois não têm de ter as formas e tamanhos mostrados na figura anterior, nem as mesmas cores (e cada boneco pode ter pixels de cores diferentes). A figura é apenas uma sugestão ilustrativa. Aqui pode inventar.

Cada pixel pode ser desenhado escrevendo diretamente na memória do ecrã ou por meio de um comando. Escolha o seu método. O guião de laboratório 4 tem os detalhes de como usar o módulo MediaCenter.

O guião de laboratório 3 ensina a trabalhar com o módulo MediaCenter, em termos de pixels (e desenhar bonecos), ecrãs de pixels, cenários e sons.

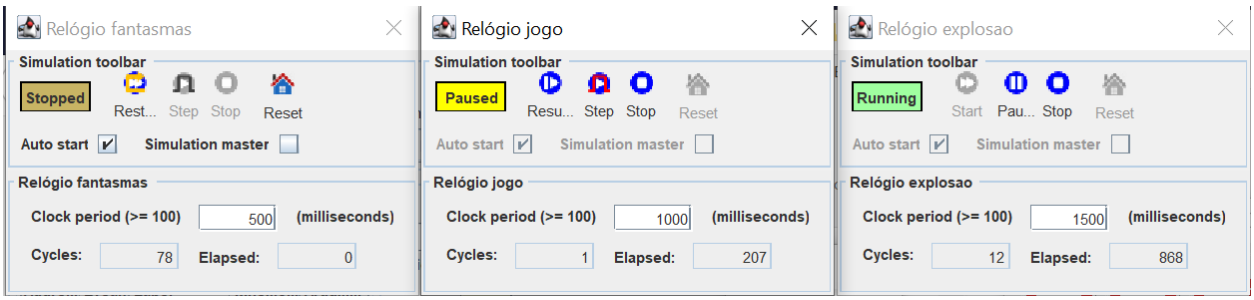
2.4 – Temporizações

A evolução do jogo requer 3 temporizações diferentes:

- Movimento dos fantasmas (período de 500 milissegundos);
- Contagem do tempo decorrido do jogo (período de 1000 milissegundos).
- Tempo que dura a explosão (período de 1500 milisegundos).

Os períodos indicados, que marcam o ritmo a que cada um dos eventos ocorre, são gerados por 3 relógios de tempo real, que geram um sinal de um bit que varia periodicamente entre 0 e 1, com um dado período. Sem o tempo real marcado por estes relógios, o jogo evoluiria de forma muito mais rápida e de forma não controlável, dependendo apenas da velocidade de processamento do computador que executa o simulador.

Estes relógios estão incluídos no circuito usado neste jogo e estão pré-programados com estes tempos, mas pode alterá-los para melhorar a jogabilidade do jogo ou fazer testes.



O arranque dos relógios é automático, pelo que nem precisa de abrir as janelas de simulação deles.

O guião de laboratório 5 ensina a utilizar relógios para marcação de temporizações.

2.5 – Escolhas pseudo-aleatórias

Quando é necessário colocar um novo Fantasma no display, use o número aleatório para gerar um atraso aleatório. Deve usar um número pseudo aleatório para decidir quando surge um fantasma.

Estas escolhas devem ser feitas de forma razoavelmente aleatória. Como o PEPE não tem mecanismos para gerar valores aleatórios, usa-se um truque simples. A leitura de um periférico de entrada (PIN, no circuito do projeto) gera valores aleatórios nos bits que estejam "no ar", ou seja, não ligados a algo que force um valor. Tal é o caso dos bits 7 a 4 do PIN, cujos bits 3 a 0 ligam ao teclado mas nada liga aos bits 7 a 4. Faz-se uma leitura do periférico (com MOVB, são lidos 8 bits), seguida de um deslocamento para a direita (instrução SHR) de 4 bits, o que coloca os bits 7 a 4 (aleatórios) do periférico nos bits 3 a 0 do registo, ficando-se assim com um valor aleatório entre 0 e 15. A partir daqui, pode obter:

- Se quiser gerar um número aleatório com 25% de probabilidade (isolando os 2 bits de menor peso, o que dá 4 hipóteses);

3 – Faseamento do projeto

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

IMPORTANTE – Não se esqueça de identificar os ficheiros de código (grupoXX.asm, em que XX é o número do grupo) em comentários, logo no início do programa, com:

- o número do grupo;
- o número e nome dos alunos que participaram no desenvolvimento do programa.

Versão intermédia:

- Vale 25% da nota do projeto (ou 12,5% da nota final de AC);
- Deve ser submetida no Fenix (Projeto AC 2023-24 - versão intermédia) até às **23h59** do dia **10 de Maio de 2024** através de um ficheiro (**grupoXX.zip**, em que XX é o número do grupo) com os seguintes ficheiros:
- Um ficheiro **grupoXX.asm** com o código, pronto para ser carregado no simulador e executado (deve ter o número do grupo, números de aluno e nomes). Deve 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;
- Todos os ficheiros de imagem e som usados no módulo "MediaCenter";
- Um ficheiro **projetoXX.cir** com o circuito do projeto, mas guardado depois de definir no módulo "MediaCenter" todos os ficheiros de imagem e som usados. Não se esqueça que estes ficheiros devem estar guardados no mesmo diretório do circuito, ou num subdiretório deste;

IMPORTANTE - Use o circuito do projeto, **projeto.cir** (e não o de qualquer guião de laboratório). Note que o periférico dos displays é de 16 bits (deve usar MOV) e não de 8 bits (MOVB);

- Numa aula de laboratório seguinte, o docente poderá fazer algumas perguntas sobre o programa entregue e dar feedback com algumas sugestões;

- **Deve cumprir os seguintes objetivos**:
  - Colocar no ecrã uma imagem de cenário de fundo adequada para este projeto;
  - Desenhar o pac-man na parte central do ecrã (posição inicial do jogo) e pelo menos um fantasma, na sua posição. Não desenhe o pac-man e o fantasma manualmente, pixel a pixel. Faça uma rotina que leia uma tabela com o valor dos pixels do pac-man e do fantasma. Fica assim mais genérico e permite desenhar qualquer objeto;
  - Implemente uma rotina para executar um efeito sonoro adequado para este projeto, invocada quando se carrega numa tecla (só pode invocar novamente quando se larga a tecla e se carrega de novo!);
  - Implemente um contador **decimal** de 0 até 100, com uma rotina para aumentar e outra para diminuir (ambas mostram o novo valor do contador após a variação). O contador não deve sair do intervalo [0 .. 100];
  - Estas rotinas devem ser invocadas repetidamente enquanto se carrega no teclado, numa dada tecla (defina duas ao seu gosto). Quando se larga a tecla, o contador deve parar de evoluir. Se a evolução for muito rápida, pode atrasá-la com uma rotina que apenas faz um ciclo de um número elevado de iterações (1000 ou mais, por exemplo).

Versão final:

- Vale 75% da nota do projeto (ou 37,5% da nota final);
- **Deve cumprir todas as especificações do enunciado**;
- Deve ser submetida no Fenix (Projeto AC 2023-24 - versão final) até às **23h59** do dia **24 de Maio de 2024**, através de um ficheiro (**grupoXX.zip**, em que XX é o número do grupo) com os seguintes ficheiros:
  - Um ficheiro **grupoXX.pdf**, relatório de formato livre, mas com a seguinte informação (juntamente com este enunciado, é fornecido um possível modelo de relatório):
    - 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 (também deve ter o número do grupo, números de aluno e nomes);
- Todos os ficheiros de imagem, vídeo e som usados no módulo "MediaCenter";
- Um ficheiro **projetoXX.cir** com o circuito do projeto, mas guardado depois de definir no módulo "MediaCenter" todos os ficheiros de imagem, vídeo e som usados. Não se esqueça que estes ficheiros devem estar guardados no mesmo diretório do circuito, ou num subdiretório deste.

- Na última semana de aula, a partir de 28 de Maio, não haverá aulas de laboratório. Essa semana está reservada para discussão do projeto com o docente das aulas de laboratório (durante o horário normal de laboratório do seu grupo). A data e local (físico ou virtual) 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 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);
- Pac-Man (para controlar a direção, disparar balas);
- Fantasmas (para controlar as ações e evolução de cada um dos Fantasmas. Note que várias Fantasmas correspondem apenas a instâncias diferentes do mesmo processo, ou várias chamadas à mesma rotina com um parâmetro que indique o número do fantasma);
- Gerador (gera um número pseudoaleatório, usado para escolher quando lança um novo fantasma);
- 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 desenho do fantasma (cobertos pela versão intermédia);
2. Movimentos do pac-man, de acordo com a tecla carregada;
3. Rotinas de ecrã, para desenhar/apagar:
   - um pixel numa dada linha e coluna (de 0 a 31 e 0 a 63, respetivamente), se ainda o não tiver feito na versão intermédia;
   - um objeto genérico, descrito por uma tabela que inclua a sua largura, altura e o valor de cada um dos seus pixels, e ainda as componentes ARGB. 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;

4. Um só fantasma (tem de definir tabelas com a representação dos fantasmas e tabelas para escolher o boneco para o pac-man);
5. 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 o pac-man);
6. Processos Controlo e Gerador;
7. Movimento do fantasma regulado por uma interrupção.
8. Deteção de colisão do pac-man com o fantasma;
9. Deteção de colisão do pac-man com um objeto;
10. Deteção de colisão do pac-man e dos fantasmas com as zonas onde não podem estar os bonecos.
11. Resto das especificações, incluindo extensão para mais do que um fantasma. Esta extensão tem algumas complicações, pelo que é melhor ter um só um fantasma 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 regulados por 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 fantasma para ele se deslocar para a esquerda), as variáveis correspondentes devem ser de 16 bits (declaradas com WORD, e não Byte).

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, bem como à ordem relativa;
- 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;
- 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 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;
- Os periféricos de 8 bits e as tabelas com BYTE devem ser acedidos com a instrução MOVB. As variáveis definidas com WORD (que são de 16 bits) e periféricos de 16 bits devem ser acedidos com MOV;
- **ATENÇÃO!!!** Ao contrário dos guiões de laboratório, o periférico POUT-1 é de 16 bits (por causa dos 3 displays) e não de 8 (deve ser acedido com MOV, e não MOVB);
- Não duplique código (com *copy-paste*). Use uma rotina com parâmetros para contemplar os diversos casos em que o comportamento correspondente é usado.

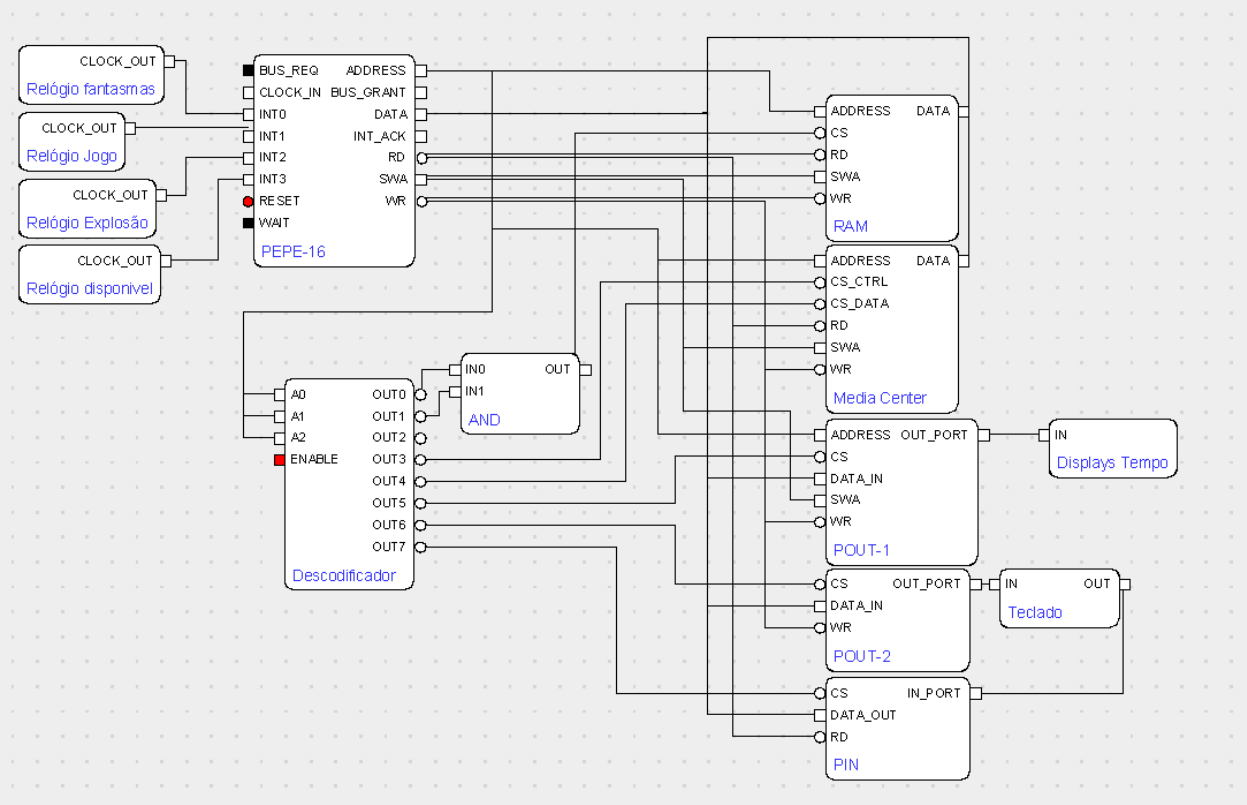
5 – Critérios de avaliação

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	1	4
Estrutura dos dados e do código	3	6
Comentários	1	2
Vários fantasmas: funcionalidade, dados e código		3
Total	**5**	**15**

6 – Circuito do projeto

A figura seguinte mostra o circuito a usar (fornecido, ficheiro **projeto_PacMan.cir**). Use este circuito para o projeto (os dos guiões não servem).



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

- Relógio fantasma – Relógio de tempo real, para ser usado como base para a temporização do movimento dos fantasmas. Está ligado ao pino de interrupção 0 do PEPE;
- Relógio tempo decorrido – Relógio de tempo real, para ser usado como base para a tempo que passou. Está ligado ao pino de interrupção 1 do PEPE;
- MediaCenter – módulo multimédia que inclui um ecrã de 32 x 64 pixels. Este ecrã é acedido por comandos ou como se fosse uma memória de 2048 pixels (ou 4096 bytes: 128 bytes em cada linha, 32 linhas). Este periférico tem 2 *chip selects*, um para acesso pela memória e outro para acesso pelos comandos. Pode ver no ficheiro de excel **ecrã-32x64.xlsx** os endereços de cada byte (relativos ao endereço de base do ecrã). O guião de laboratório 3 fornece mais detalhes;

- Três displays de 7 segmentos, ligados aos bits 11-8, 7-4 e 3-0 do periférico POUT-1, para mostrar a energia da nave. **ATENÇÃO!!!**: ao contrário dos guiões de laboratório, este periférico é de 16 bits e deve ser acedido com MOV (e não MOVB);
- 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. Atenção, que estes periféricos são de 8 bits e devem ser acedidos com MOVB (e não MOV). Note também que só os 4 bits de menor peso (3 a 0) são significativos. Os restantes (7 a 4) estão no ar e leem valores aleatórios. Por isso, deve usar uma máscara para os eliminar ao tentar detetar teclas premidas;
- Memória (RAM), que tem 16 bits de dados e 14 bits de endereço, com capacidade de endereçamento de byte, tal como o PEPE e o MediaCenter;
- PEPE-16 (processador de 16 bits).

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

Dispositivo	Endereços
RAM	0000H a 3FFFH
MediaCenter (acesso aos comandos)	6000H a 6069H (ver guião de laboratório 3)
MediaCenter (acesso à sua memória)	8000H a 80FFH
POUT-1 (periférico de saída de 16 bits)	0A000H
POUT-2 (periférico de saída de 8 bits)	0C000H
PIN (periférico de entrada de 8 bits)	0E000H