Projeto de Introdução à Arquitetura de Computadores

LEIC

IST-Taguspark

Treino de ninjas

2017/2018

1 – Objetivos

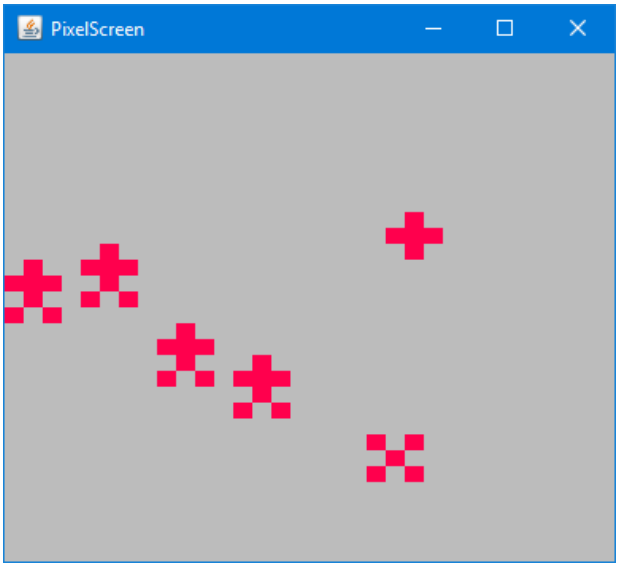
Este projeto pretende exercitar os fundamentos da área de Arquitetura de Computadores, nomeadamente a programação em linguagem *assembly*, os periféricos e as interrupções.

O objetivo deste projeto consiste em concretizar um jogo para treino de ninjas. No lado esquerdo de um ecrã estão 4 ninjas, que devem concentrar-se de modo que a sua energia lhes permita flutuar. No entanto, a gravidade faz-se sentir e cada ninja vai caindo. É preciso que o jogador lhes vá restaurando a energia de modo que não caiam. Se algum ninja bater no chão, morre e desaparece do jogo.

Do lado direito do ecrã aparecem armas, que podem ser estrela de lâminas (X) ou presentes (+), que se vão deslocando para a esquerda, ao encontro dos ninjas. Se um ninja for atingido por uma estrela, morre. Se for atingido por um presente, ganha 3 pontos, que se vão acumulando na pontuação do jogo, mostrada em displays de 7 segmentos. Podem aparecer duas armas simultaneamente, uma na metade superior do ecrã, outra na metade inferior, mas em alturas aleatórias (variam a cada lançamento).

É possível subir ou descer manualmente cada ninja, de modo a evitar as estrelas e tentar apanhar os presentes. O jogo acaba quando todos os ninjas morrerem.

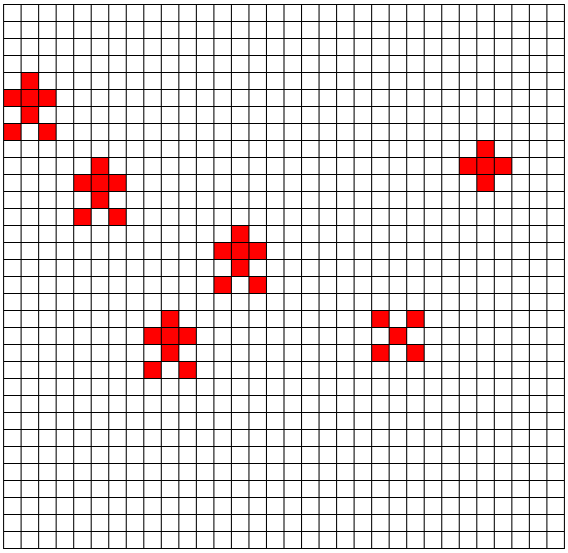
A figura seguinte ilustra um possível aspeto do ecrã jogo.



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 ninjas e das armas (a quadrícula não aparece no ecrã).

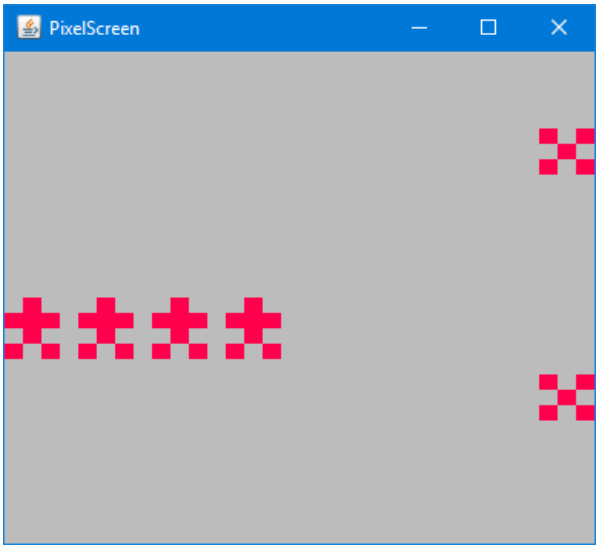


Os bonecos são apenas ilustrativos e podem ser usados outros. Junto com este enunciado está um ficheiro de excel (**ecra.xlsx**) com mais algum detalhe, podendo ser usado para planear outros bonecos.

Cada ninja pode apenas deslocar-se na vertical, dentro de uma coluna de 4 pixels de largura.

As armas podem apenas deslocar-se na horizontal, da direita para a esquerda, cada uma dentro da sua metade vertical do ecrã (mas a uma altura que pode variar, de lançamento para lançamento).

A figura seguinte ilustra um possível cenário, quando o jogo começa.



2.2 – Teclado e controlo do jogo

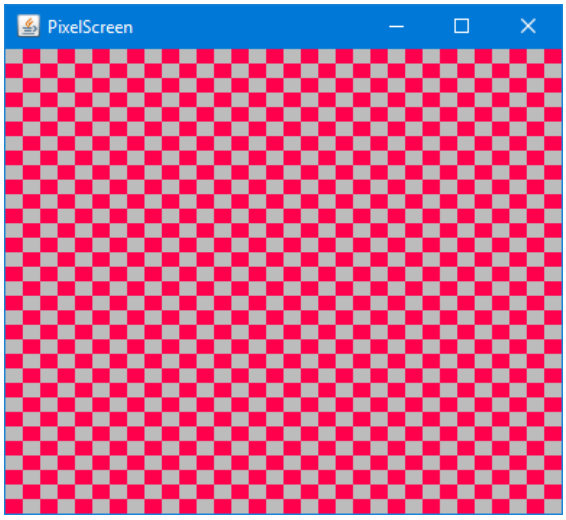
O jogo é controlado por meio de teclas num teclado, tal como o da figura seguinte:

Embora a atribuição de teclas a comandos seja livre, recomenda-se a seguinte distribuição:

- Teclas 0 a 3: subir de um pixel o ninja respetivo;
- Teclas 4 a 7: descer de um pixel o ninja respetivo;
- 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).

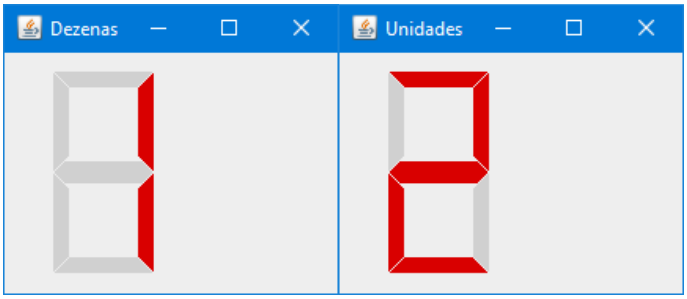
Cada tecla (atuada por clique do rato) dever executar apenas um comando. Para novo comando, mesmo que igual, tem de se largar a tecla e carregar de novo.

Recomenda-se um ecrã diferente, como por exemplo o seguinte, 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 qualquer ninja apanha (colide com) um presente.

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!

2.4 - Ninjas

Cada ninja vai caindo, se não se fizer nada. Se chegar ao chão, morre. Pode apenas movimentar-se na vertical, não podendo subir mais se chegar ao teto.

Se um ninja for atingido por uma estrela, morre. Se apanhar um presente, acrescenta 3 pontos à pontuação.

O jogador pode subir ou descer manualmente cada ninja, carregando nas teclas, tal como descrito na secção 2.2. Isto permite fugir das estrelas e apanhar os presentes.

Quando o jogo arranca, os quatro ninjas devem estar em posições pré-definidas. A figura da página 2 ilustra um exemplo, mas quaisquer outras posições iniciais são aceitáveis.

2.5 - Armas

Do lado direito do ecrã vão aparecendo armas, que podem ser estrela de lâminas (X) ou presentes (+), que se vão deslocando para a esquerda, ao encontro dos ninjas. Há duas armas simultâneas, cada uma na sua metade vertical do ecrã. No entanto, a sua linha de disparo (que se mantém depois enquanto a arma durar) pode variar aleatoriamente, em cada lançamento.

Uma arma desaparece quando embate num ninja ou na parede do lado esquerdo. Nessa altura, nova arma é lançada, mas com uma altura escolhida aleatoriamente, na mesma metade vertical do ecrã, e provavelmente de outro tipo. A escolha do tipo de arma (estrela ou presente) deve ser aleatória, com uma probabilidade de ser presente de 25% (ou 50%, se achar que os presentes são poucos...).

Se uma estrela atingir um ninja, este morre. Se um presente atingir um ninja, 3 pontos são acumulados à pontuação do jogo.

3 – Faseamento do projeto

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

A versão intermédia:

- Vale 20% da nota do projeto, ou 6% da nota final;
- Deve cumprir os objetivos indicados a seguir;
- Deve ser submetida no Fenix (Projeto IAC 2017-18 - versão intermédia) através de um ficheiro (**grupoXX.asm**, em que XX é o número do grupo) com o código do programa, tal como ele estiver na altura, até às 23h59 do dia 28 de outubro de 2017. 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;
- Deve ser mostrada a funcionar ao docente, no laboratório. O docente poderá fazer algumas perguntas sobre o programa e dar algumas sugestões.

Objetivo para a versão intermédia: Um programa que implemente uma máquina de calcular rudimentar em decimal, de acordo com as seguintes especificações:

- Use o circuito do projeto (e não o de qualquer guião de laboratório);
- Os dois displays hexadecimais mostram um valor decimal, entre 00 e 99 (inicialmente, 00);
- A máquina sabe fazer as seguintes operações:
  - Tecla C seguida de uma tecla entre 0 e 9: soma em decimal esse dígito ao valor anterior do display, e o resultado aparece no display. Se o resultado for superior a 99, limita a 99;
  - Tecla D seguida de uma tecla entre 0 e 9: subtrai em decimal esse dígito ao valor anterior do display, e o resultado aparece no display. Se o resultado for inferior a zero, limita a 00;
  - Tecla E: soma 3 ao valor do display, que mostra depois o resultado (suponhamos que um ninja apanhou um presente). Máximo valor: 99;
  - Tecla F: subtrai 3 ao valor do display, que mostra depois o resultado. Mínimo valor: 00;
- Se houver uma sequência errada de teclas, a máquina deve ignorar.

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

A versão final do projeto deverá ser entregue até às 23h59 do dia 1 de dezembro de 2017. A entrega, a submeter no Fenix (Projeto IAC 2017-18 - versão final) deve consistir de um zip (**grupoXX.zip**, em que XX é o número do grupo) com dois ficheiros:

- Um ficheiro **relatorio.pdf**, de formato livre, mas com a seguinte informação:
  - Identificação do número do grupo, números de aluno e nomes;
  - Definições relevantes, se tiverem feito algo diferente do que o enunciado pede (teclas diferentes, funcionalidade a mais, etc.)
  - Indicação concreta das funcionalidades pedidas que o código enviado NÃO satisfaz;
  - Eventuais outros comentários.
- Um ficheiro **grupoXX.asm** com o código, pronto para ser carregado no simulador e executado.

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 inicial 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 4);
- Ninja (para controlar as ações de cada um dos ninjas);
- Arma (para controlar as ações de cada arma);
- Gerador (para gerar um número aleatório, usado para escolher a altura a que cada arma aparece, bem como o seu tipo);
- Controlo (para tratar das teclas de começar, suspender/continuar e terminar).

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

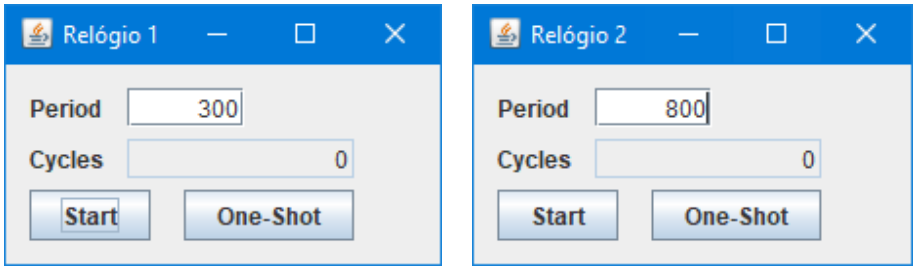
1. Teclado e displays (coberto pela versão intermédia);
2. Rotinas de ecrã (desenhar/apagar um pixel numa dada linha e coluna, desenhar/apagar ninjas ou armas – represente os objetos pelas coordenadas de um determinado pixel, como por exemplo o canto superior esquerdo, e desenhe-os relativamente às coordenadas desse pixel);
3. Ninja (apenas um, em vez de quatro, com deslocamentos de um pixel para cima e para baixo por meio do teclado e morte se chegar ao fundo);
4. Processos cooperativos (organização das rotinas preparada para o ciclo de processos);
5. Interrupção de queda do ninja;
6. Arma (apenas uma, incluindo a respetiva interrupção que regula o seu movimento, bem como outras alterações ao programa que seja necessárias, nomeadamente deteção de colisão);
7. Processos Controlo e Gerador;
8. Resto das especificações, incluindo extensão para quatro ninjas e duas armas.

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, 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 aleatório (entre 0 e 7, por exemplo, para dar alguma variabilidade na altura das armas, dentro de cada metade vertical do ecrã), basta ler esse contador e usar apenas os seus bits menos significativos (3, no caso de 0 a 7). Como o ciclo de processos se repete muitas vezes durante a iteração dos vários processos, esses bits parecerão aleatórios.

As temporizações de queda dos ninjas e do deslocamento das armas 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.



Finalmente:

- 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 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;
- 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.

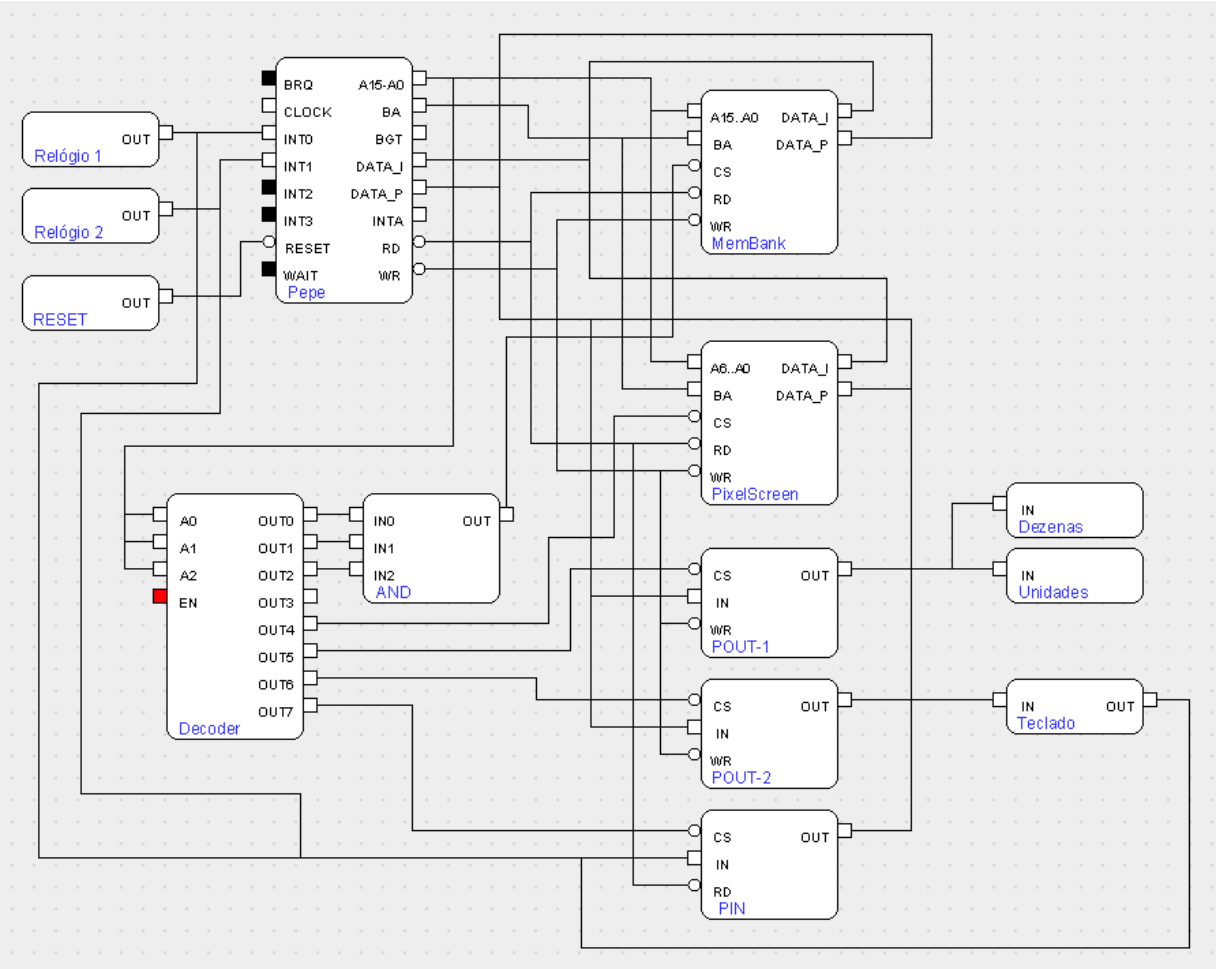
5 – Critérios de avaliação

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 básica	3	6
Estrutura dos dados e código, interrupções		5
Comentários	1	2
Vários ninjas e armas		3
Total	**4**	**16**

6 – Implementação

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 das armas. Este sinal também liga ao bit 4 do periférico de entrada PIN, que poderá ir lendo para animar o movimento das armas, numa versão anterior a usar interrupções;
- Relógio 2 – Relógio de tempo real, para ser usado como base para a temporização da queda dos ninjas. 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 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

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 o byte alterado no mesmo endereço;
- 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 durante o varrimento das teclas;
- 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.