Instituto Superior Técnico

Arquitectura de Computadores

2010/2011 2º Semestre

Projecto

"Line Breaker"

Versão 1.0

Março de 2011

Renato.Nunes@ist.utl.pt

1. Introdução

Com o objectivo de desenvolver a capacidade de programação estruturada em *assembly* e a capacidade de manipulação de estruturas de dados, propõe-se a realização de um jogo interactivo que se designou de "*Line Breaker*" e que será descrito em seguida.

2. Descrição do Jogo

O jogo "*Line Breaker*" possui um espaço de jogo rectangular (ilustrado na figura 1) em que as dimensões interiores são de 9 caracteres em largura por 20 em altura. O jogo decorre na janela de texto do simulador "p3sim" devendo ser usados os caracteres indicados para delimitar o espaço de jogo.

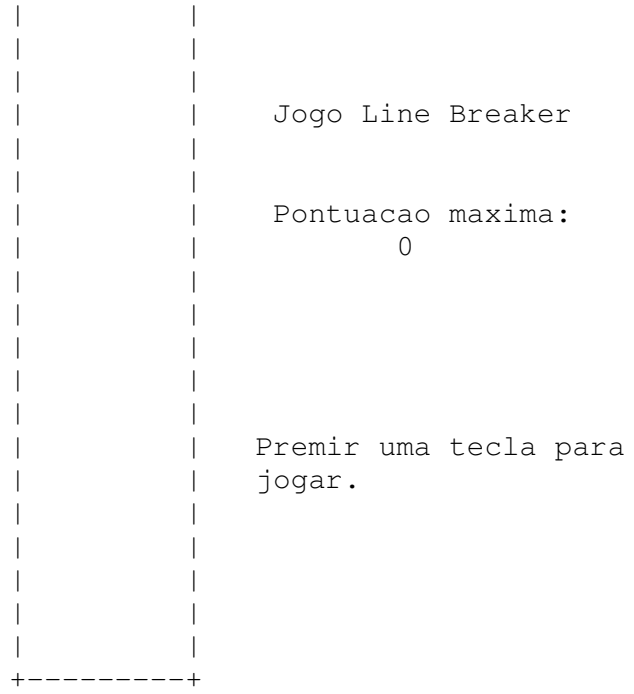

Figura 1 - Espaço de jogo

Na linha superior desse espaço, ao centro, aparece aleatoriamente uma das quatro peças ilustradas na figura 2.

Figura 2 - Tipos de peças

A peça executa um movimento de vaivém nessa linha, deslocando-se para a direita até atingir o bordo do espaço de jogo, altura em que inverte o seu movimento e passa a deslocar-se para a esquerda. Quando atinge o bordo esquerdo do espaço de jogo o seu movimento volta a inverter-se. Na figura 3 ilustra-se o movimento da peça '-'.

```
 t0: |    -    |  t1: |     -   |   t2: |      -  |   t3: |       - |
      123456789        123456789         123456789         123456789

 t4: |        -|  t5: |       - |   t6: |      -  |   t7: |     -   |
      123456789        123456789         123456789         123456789

 t8: |    -    |  t9: |   -     |  t10: |  -      |  t11: | -       |
      123456789        123456789         123456789         123456789

t12: |-        | t13: | -       |  t14: |  -      |  t15: |   -     |
      123456789        123456789         123456789         123456789

t16: |    -    |
      123456789
```

Figura 3 – Movimento de vaivém (sempre na linha superior)

O movimento da peça é simulado escrevendo um espaço em branco na posição que a peça ocupava (o que "apaga" a peça) e escrevendo a peça na nova posição.

O jogador pode desencadear a queda da peça clicando no botão **I0** na janela de entradas e saídas ("Janela Placa"). Quando isso acontece, a peça cai na vertical até atingir a última linha do espaço de jogo ou até encontrar outra peça. Se o jogador não premir nenhuma tecla, a peça efectua 16 movimentos e em seguida cai automaticamente (cai na coluna central do espaço de jogo). O jogo termina se já não existir espaço para uma nova peça cair.

Quando uma peça cai e se imobiliza é verificado se numa mesma linha existem 3 ou mais peças iguais em posições contíguas, caso em que essas peças desaparecem, fazendo cair as peças que porventura estavam em cima delas. Na figura 4 apresentam-se dois exemplos. No primeiro, a peça '#' está a cair na coluna 3 dando origem ao ecrã imediatamente ao lado. No segundo exemplo a peça 'O' está a cair na coluna 5 dando origem ao ecrã ilustrado à sua direita. Para se compreender melhor este segundo exemplo são detalhados na figura 5 os vários passos que ocorrem.

```
|  #       |  |          |       |    O     |  |          |
|          |  |          |       |          |  |          |
|          |  |          |       |          |  |          |
|#         |->|          |       |   # -    |->|          |
|-O  O  --|  |#       --|       | ##- :--#|  |          |
|## ##:-##|  |-O  O:-##|       |#--O O::-|  |         #|
+---------+  +---------+       +---------+  +---------+
      Exemplo 1                       Exemplo 2
```

Figura 4 - Ilustração de peças a cair e eliminação de peças numa linha

```
|   # -    |    |          |    |          |    |          |
| ##- :--#| -> | ### ---#| -> |         #| -> |          |
|#--OOO::-|    |#--- :::-|    |#### ----|    |         #|
+---------+    +---------+    +---------+    +---------+
```

Figura 5 - Ilustração da eliminação de várias peças em várias linhas

Depois de uma peça cair e se imobilizar, e após a eliminação de todos os blocos de 3 ou mais peças do mesmo tipo numa mesma linha em toda a área de jogo, é gerada nova peça na posição central da primeira linha e é iniciado o movimento de vaivém já descrito. A nova peça é gerada aleatoriamente, com uma distribuição uniforme, de entre as 4 peças possíveis.

Para além do botão **I0** o utilizador pode também usar o botão **I1** que converte a peça que estava em movimento numa bomba, representada pelo carácter '@'. Uma bomba, ao cair, elimina a peça (uma só) que fica imediatamente por baixo dela (e a própria bomba desaparece). Uma bomba que caia na última linha do espaço de jogo não tem qualquer efeito e apenas desaparece.

Na figura 6 ilustram-se dois exemplos da queda de uma bomba.

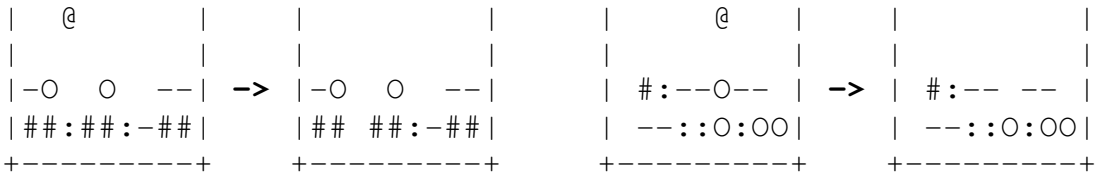

Figura 6 – Efeitos da queda de uma bomba

3. Evolução do Jogo e Pontuação

3.1 Evolução do jogo

O jogo pode ser decomposto em três fases principais:

1. Inicialização e desenho do espaço de jogo
2. Jogo
3. Fim de jogo e actualização da pontuação máxima

Na fase 1 são inicializadas as estruturas de dados do programa e é desenhado o espaço de jogo no ecrã (na janela de texto do simulador p3sim). O espaço de jogo inicial pode estar vazio, como ilustrado na figura 1, ou pode conter um conjunto de peças predefinidas. Existem 4 ecrãs iniciais possíveis que são escolhidos através dos interruptores de alavanca presentes na placa de entradas e saídas do simulador ("Janela Placa"). Para escolher o ecrã inicial apenas são relevantes os dois interruptores mais à direita que deverão ser interpretados como um valor binário:

$$00 \rightarrow \text{Ecrã 0},\ 01 \rightarrow \text{Ecrã 1},\ 10 \rightarrow \text{Ecrã 2}\ \text{ e }\ 11 \rightarrow \text{Ecrã 3}$$

O conteúdo de cada um dos 4 ecrãs possíveis está definido numa estrutura de dados própria que será descrita na secção 5.

Na fase 1, após desenhar o espaço de jogo inicial, o programa fica a aguardar que o utilizador prima uma tecla qualquer antes de continuar – ver figura 7. Enquanto o programa está em ciclo, a aguardar que seja premida uma tecla, está sistematicamente

a chamar uma rotina que calcula um número aleatório (ver secção 4). Este mecanismo permite melhorar significativamente a qualidade dos valores aleatórios gerados pois não é possível antecipar em que instante o utilizador irá premir uma tecla.

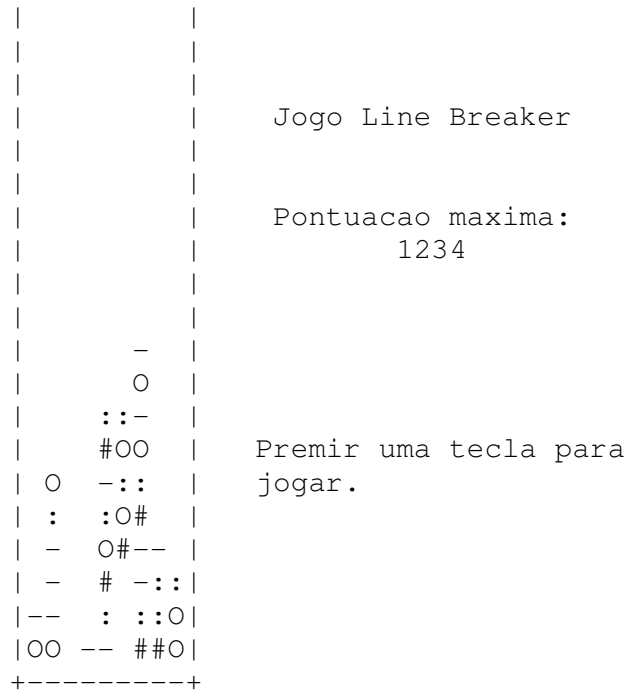

Figura 7 – Início de um jogo

Na fase 2 é gerada uma peça que é escrita na posição central da primeira linha do espaço de jogo. A peça é gerada de forma aleatória e equiprovável de entre as 4 possíveis ('#', 'O', ':' e '-'). Na secção 4 é descrito o algoritmo a usar para gerar números aleatórios.

Inicia-se então o movimento de vaivém da peça. Entre cada posição da peça irá decorrer um intervalo de tempo que depende do nível de dificuldade do jogo – ver a sub-secção 3.3. A marcação do tempo será efectuada usando o temporizador ("timer") do P3.

Durante o movimento de vaivém da peça o jogador pode clicar no botão **I0** o que desencadeia a queda da peça. Se não o fizer, ao fim de 16 movimentos (quando a peça volta a atingir a coluna central) ela cai por si só. O jogador também pode clicar no botão **I1**, o que origina a transformação da peça numa bomba ('@') que efectua o movimento de queda.

Recorda-se que os botões **I0** e **I1** estão associados às interrupções 0 e 1 do p3sim. Para simplificar a interacção com o jogo, evitando o uso obrigatório do rato, podem ser usadas as teclas '**q**' e '**a**' do teclado do computador. Premir essas teclas é equivalente, respectivamente, a clicar no botão **I0** e **I1**.

Durante a queda, a peça percorre todas as posições até atingir a linha de fundo do espaço de jogo ou até atingir outra peça, situação em que se imobiliza. Esse movimento de queda pode ser tão rápido quanto possível, sem necessidade de usar um

tempo específico entre cada posição. Verifica-se o mesmo para o movimento de queda da bomba.

Após uma peça se imobilizar é executado o algoritmo de detecção e eliminação de grupos de 3 ou mais peças do mesmo tipo numa linha. Recordar o exemplo ilustrado na figura 5 em que ocorrem várias eliminações numa mesma linha e em várias linhas.

3.2 Pontuação

Por cada peça que é eliminada o jogador ganha 1 ponto. Para o exemplo da figura 5 o jogador ganhava 17 pontos. Sempre que o jogador recorre a uma bomba perde 10 pontos (caso a bomba elimine alguma peça isso não dá pontos ao jogador). Se o jogador atingir uma pontuação negativa, o jogo termina. Isso ocorre quando o jogador usa uma bomba e tem menos de 10 pontos. O jogo termina também quando a peça se imobiliza na primeira linha do espaço de jogo.

A pontuação do jogador é afixada em cada momento no "display" de sete segmentos do simulador p3sim.

3.3 Nível de dificuldade

O nível de dificuldade do jogo está associado à rapidez do movimento da peça.

Entre cada movimento da peça irá decorrer um intervalo de tempo que pode variar entre 1 segundo e 0,3 segundos. Inicialmente o tempo será de 1 segundo, tornando-se o movimento mais rápido à medida que o número de peças eliminadas aumenta. O tempo inicial de 1 segundo é reduzido de 0,1 segundos por cada 5 peças eliminadas (o valor 5 deverá poder ser facilmente alterado para qualquer outro valor). Assim, após o jogador ter eliminado 35 peças (7 x 5 peças) passa a usar o intervalo de tempo de 0,3 segundos, valor que não reduz mais.

O nível de dificuldade do jogo é assinalado nos "leds" do simulador. Para o intervalo de 1 segundo só está aceso o primeiro "led" da direita. Por cada redução de 0,1 s acende mais um "led". Assim, quando o intervalo de tempo for 0,3 segundos estarão acesos 8 "leds" (nível de dificuldade máximo).

Recorda-se que a marcação de tempo deverá usar o temporização ("timer") do P3.

3.4 Fim do jogo

No fim de um jogo (fase 3) é actualizada a pontuação máxima (caso o jogador tenha obtido mais pontos que os jogadores anteriores) e é apresentada a mensagem "GAME OVER" seguida de "Premir uma tecla para iniciar um jogo novo." – ver figura 8.

Após o utilizador premir uma tecla são apagadas as mensagens "GAME OVER" e "Premir uma tecla para iniciar um jogo novo." (são escritos espaços em branco ' ' sobre esses caracteres) e passa-se para a fase 1 já descrita acima.

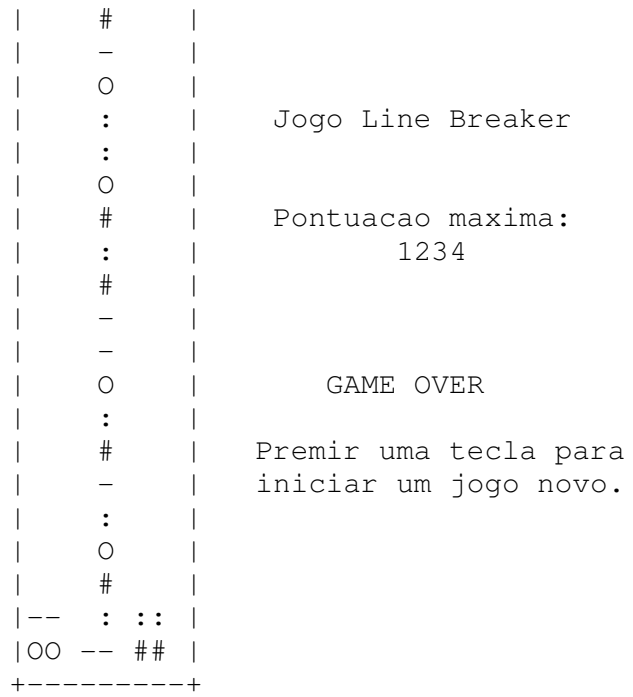

Figura 8 – Fim de um jogo

4. Geração de um Número Aleatório

Para gerar um número aleatório será usado um algoritmo que simula um registo de deslocamento modificado com realimentação, o qual permite gerar uma sequência pseudo-aleatória de números de 16 bits, com um passo de repetição longo e com uma distribuição uniforme (os números gerados são equiprováveis).

O algoritmo a usar recorre a um padrão de bits (*rnd_mask*) e a uma variável (*random*). A variável *random* é usada como base para o cálculo de um novo valor, o qual é guardado nessa mesma variável. O valor $random_{bit0}$ representa o bit 0 (bit menos significativo) da variável *random*.

```
rnd_mask = 1000 0000 0001 0110

if (random_bit0 = 0) /* testa o bit de menor peso */
   random = rotate_right (random)
else
   random = rotate_right (random XOR rnd_mask)
```

Figura 9 – Algoritmo de geração de um número aleatório

Os valores gerados são grandezas de 16 bits que deverão ser convertidos na gama de valores desejados (por exemplo, entre 0 e 3). Este aspecto fica ao cuidado dos alunos.

5. Estruturas de Dados

Para uniformizar as implementações do jogo e também para simplificar a sua concepção, é obrigatório usar as estruturas de dados que se descrevem em seguida.

A primeira estrutura de dados designa-se de "JOGO" e representa o espaço de jogo. É composta por um vector com 180 palavras que corresponde ao interior do espaço de jogo.

```
JOGO_DIM   EQU   180

JOGO       TAB   JOGO_DIM
```

Figura 10 – Declaração do vector JOGO

O vector JOGO está organizado em 20 grupos de 9 palavras que representam as 20 linhas do espaço de jogo. As primeiras 9 posições do vector correspondem à primeira linha, as 9 posições seguintes correspondem à segunda linha e assim por diante.

Cada posição do vector JOGO contém o código ASCII do espaço em branco ' ' (0020 h) ou de uma das peças '#' (0023 h), 'O' (004F h), ':' (003A h) ou '-' (002D h).

O vector JOGO só é actualizado quando uma peça se imobiliza.

O vector é usado pelo algoritmo que detecta grupos de 3 ou mais peças do mesmo tipo numa linha, caso em que essas peças são eliminadas do vector (e o ecrã é actualizado). Em seguida é analisado se existem peças por cima das que foram eliminadas, caso em que essas peças se deslocam para a linha abaixo (recordar o exemplo apresentado na figura 5). As linhas modificadas no vector deverão ser escritas no ecrã para que seja visualizada a situação actual do jogo.

Existem ainda 4 estruturas de dados adicionais correspondentes a 4 vectores designados de "INI_0_JOGO", "INI_1_JOGO", "INI_2_JOGO" e "INI_3_JOGO". Cada um destes vectores segue a mesma lógica do vector JOGO e contém um espaço de jogo inicial. Para uniformizar o teste do jogo é obrigatório que o vector "INI_0_JOGO" corresponda a um espaço de jogo vazio (sem nenhuma peça) e que o vector "INI_1_JOGO" tenha o conteúdo presente na figura 11.

Os vectores "INI_2_JOGO" e "INI_3_JOGO" podem ter um conteúdo qualquer, ficando ao cuidado dos alunos a sua definição.

Os 4 vectores descritos são usados para definir 4 possíveis espaços de jogo iniciais. A determinação de qual o vector a usar é feita através de 2 interruptores de alavanca existentes na janela de entradas e saídas (ver sub-secção 3.1). Esses interruptores são lidos na fase 1 do jogo e, em função do seu valor, determinam que vector será copiado para a estrutura de dados "JOGO" que será desenhada no ecrã.

```
INI_1_JOGO  STR  '          '
INI_1_L02   STR  '          '
INI_1_L03   STR  '          '
INI_1_L04   STR  '          '
INI_1_L05   STR  '          '
INI_1_L06   STR  '          '
INI_1_L07   STR  '          '
INI_1_L08   STR  '          '
INI_1_L09   STR  '          '
INI_1_L10   STR  '   O :    '
INI_1_L11   STR  '   - -    '
INI_1_L12   STR  '   # :    '
INI_1_L13   STR  '   # O    '
INI_1_L14   STR  '   : -    '
INI_1_L15   STR  '   - :    '
INI_1_L16   STR  '   # O    '
INI_1_L17   STR  '   O #    '
INI_1_L18   STR  '  O# -## '
INI_1_L19   STR  ' ##- :--#'
INI_1_L20   STR  '#--O O::-'
```

Figura 11 – Exemplo de declaração do vector INI_1_JOGO

6. Microprogramação

Pretende-se microprogramar uma nova instrução para o P3 que ajude a determinar se uma linha do espaço de jogo possui 3 ou mais peças do mesmo tipo em posições consecutivas.

A nova instrução terá apenas um operando sendo representada por: I1OP *op1*

O operando *op1* deverá conter o endereço da primeira posição de uma dada linha do espaço do jogo. A instrução determinará se existem 3 peças do mesmo tipo em posições consecutivas. Em caso afirmativo, devolve em *op1* o endereço da primeira peça do grupo e assegura que a flag Z (zero) está inactiva. Se percorreu as 9 posições de uma linha e não encontrou 3 ou mais peças idênticas consecutivas, retorna 0 e activa a flag Z.

O código (*opcode*) da instução I1OP é 010111 (17h).

Na figura 12 apresenta-se um exemplo de chamada da instrução I1OP.

```
MOV    R1, JOGO
ADD    R1, 18      ; índice da 3.linha
I1OP   R1
BR.Z   NADA_A_APAGAR
; apagar peça em M[R1] e posições seguintes
...
...
```

Figura 12 – Exemplo de chamada da nova instrução I1OP

7. Plano de Desenvolvimento

7.1 Desenvolvimento do trabalho

Sugere-se o seguinte plano de desenvolvimento:

1. Desenhe o fluxograma que descreve cada um dos procedimentos da aplicação, com especial atenção à relação entre o fluxo do programa principal e as várias rotinas de tratamento de interrupção. Estes fluxogramas e a lógica funcional do programa principal deverão ser apresentados na 1ª aula de projecto.
2. Para o conjunto de procedimentos que definiu, identifique claramente as entradas, as saídas e os registos modificados na sua execução.
3. Programe e teste minuciosamente cada uma das rotinas que efectuam a interface com os dispositivos de entrada e de saída (janela de texto, *LEDS*, *display* de sete segmentos e botões), com especial atenção à passagem de parâmetros entre estas rotinas e o programa principal.
4. Associe as rotinas que tratam os botões **I0** e **I1** ao vector de interrupção respectivo.
5. Configure o temporizador disponibilizado pelo simulador e associe o vector de interrupção respectivo com a rotina a executar periodicamente. Configure o ciclo de jogo para que este seja sincronizado pela mudança de valor de uma dada variável, modificada pela rotina de tratamento da interrupção do temporizador.
6. Realize a ligação entre os vários procedimentos, de forma a obter o comportamento desejado e especificado.
7. Embora a verificação se uma linha possui 3 ou mais peças iguais em posições consecutivas vá ser realizada pela nova instrução I1OP, comece por realizar uma implementação provisória do procedimento utilizando uma rotina *assembly*. Deste modo poderá iniciar, o mais cedo possível, o desenvolvimento dos vários procedimentos envolvidos no jogo. Esta rotina deverá ser substituída pela invocação da nova instrução logo após a 2ª aula de projecto.
8. Comente e indente devidamente o código desenvolvido.

7.2 Faseamento da codificação

Não deve tentar codificar todo o programa de uma só vez. Implemente as várias funcionalidades do programa de uma forma faseada e efectue os testes necessários para verificar o seu correcto funcionamento. Não prossiga para a implementação de funcionalidades mais avançadas sem ter garantido que as que lhe servem de base estão correctamente implementadas.
Estando o sistema a funcionar correctamente, pode incluir eventuais funções opcionais que entretanto tenha desenvolvido.

8. Plano de Entrega

Todas as entregas devem ser feitas num envelope identificado com o **dia e hora do turno** e **número do grupo**.
A entrega final deve conter ainda:

- Relatório.
- Código impresso. Cada ficheiro deverá ser impresso utilizando 2 páginas por face, frente e verso, numeradas; sem linhas cortadas; devidamente comentado e indentado; agrafado; com a identificação do grupo como comentário na primeira página. Para tal deverá utilizar a aplicação **p3print** fornecida na página da cadeira.
- CD-ROM com o código, microprogramação da nova instrução e relatório (devidamente identificado com o número do grupo).

A data limite de entrega é dia 19 de Maio, até às 17h, na Reprografia do DEI. Essa data é a mesma para todos os grupos. Recorda-se, no entanto, que a verificação do funcionamento dos programas será feita no turno correspondente de cada grupo.

Os projectos que sejam entregues após a data limite serão aceites com uma penalização de 2 valores por cada dia de atraso (incluindo sábados, domingos e outros dias não úteis).

O calendário das discussões será acordado com o docente do turno respectivo. Para os projectos entregues atempadamente a discussão terá lugar, preferencialmente, entre os dias 26 de Maio a 1 de Junho. As discussões dos trabalhos entregues fora de prazo poderão ser adiadas.

1ª Aula (7 a 13 de Abril)	
Demonstração:	• O programa a apresentar deverá (no mínimo) desenhar o espaço de jogo (incluindo as mensagens de texto) e implementar o movimento de vaivém de uma peça na primeira linha.
Entrega:	• Fluxogramas da aplicação e dos principais procedimentos que lhe servem de base (mesmo que ainda não estejam implementados), com especial atenção à relação entre o fluxo do programa principal, os vários módulos funcionais e as rotinas de tratamento de interrupção.

2ª Aula (5 a 11 de Maio)	
Demonstração:	• Execução da instrução I1OP utilizando um programa de teste que demonstre vários casos possíveis.

Entrega:	• Microcódigo da instrução I1OP usando linguagem de transferência de registos; • Tabela com a codificação binária e hexadecimal das microinstruções e o mapeamento da ROM A (use a tabela 1 da ficha do 6º Trabalho de Laboratório); • Código *assembly* de um programa de teste da instrução I1OP realizada.

3ª Aula (12 de Maio a 18 de Maio)	
Demonstração: (na aula de cada grupo)	• Funcionamento do jogo concebido.
Entrega: (19 de Maio, na Reprografia do DEI, até às 17h)	• Breve relatório com a descrição do projecto realizado, organização do programa e explicação dos aspectos mais relevantes da implementação. Na conclusão deverá ser feito um balanço do que foi realizado, com indicação dos aspectos nos quais o projecto tenha divergido do enunciado base (funcionalidades adicionais implementadas, funcionalidades não implementadas, outras variações ou divergências, etc.). • Fluxogramas finais da aplicação e dos principais procedimentos que lhe servem de base. • Código desenvolvido devidamente comentado e indentado (impresso frente e verso a duas páginas por face conforme indicado anteriormente).