Instituto Superior Técnico
Licenciatura em Engenharia Informática e de Computadores

Projeto de

Arquitetura de Computadores

Chuva de Meteoros

(Versão 1.0)

2012/2013

Índice

1 Objetivo 3
2 Descrição dos Blocos do Jogo 3
2.1 Espaço de jogo 3
2.2 Canhão 3
2.3 Meteoros 4
2.4 Dispositivos 5
2.5 Início do Jogo 5
2.6 Fim do Jogo 5
2.7 Versões avançadas 5
3 Implementação 6
3.1 Movimentação do canhão 6
3.2 Movimentação do meteoro 6
3.3 Movimentação dos mísseis 6
3.4 Temporizações 6
3.5 Valores aleatórios 6
4 Microprogramação 7
5 Plano de Desenvolvimento 7
5.1 Desenvolvimento do trabalho 7
5.2 Faseamento da codificação 8
6 Plano de Entrega 8
Anexo A – Geração de sequência pseudoaleatória 10

1 Objetivo

O projeto consiste no desenvolvimento de um jogo cujo objetivo é impedir os meteoros de chegar à Terra.

O jogo decorre na janela de texto. Os meteoros surgem na linha de cima e deslocam-se para baixo na vertical. Na linha de baixo existirá um canhão que o utilizador pode deslocar para a esquerda e para a direita e que permite disparar mísseis para abater os meteoros. O jogador receberá pontos por cada meteoro destruído. O jogo termina quando três meteoros atingirem a Terra.

Neste documento são descritos os detalhes de funcionamento pretendidos para o jogo.

O jogo será programado usando a linguagem *assembly* do P3 e será também microprogramada uma nova instrução. O desenvolvimento e teste do programa serão realizados usando o simulador do P3 (p3sim), sendo usados os diversos recursos disponibilizados, de que se destacam: a janela de texto, os *displays* de 7 segmentos, os LEDs, o *display* LCD e os interruptores. O projeto deverá depois ser demonstrado no laboratório na placa com a versão hardware do P3.

2 Descrição dos Blocos do Jogo

2.1 Espaço de jogo

O jogo irá desenrolar-se na Janela de Texto do simulador do P3. Esta janela corresponde a uma matriz de 80 colunas por 24 linhas em que, em cada posição, pode ser escrito um caráter ASCII.

A Figura 1 apresenta o espaço de jogo. No início, deve apenar conter:

- o chão: 80 carateres '-' a ocupar toda a linha 23;
- o canhão: centrado na linha 22 e simbolizado por '<^>'.

Os números na Figura 1 servem apenas para ajudar a identificar as várias linhas e colunas e não devem aparecer. O canto superior esquerdo corresponde à linha 0 e coluna 0.

2.2 Canhão

O canhão irá deslocar-se na horizontal sobre a linha 22. O jogador utilizará os botões de interrupção I0 e IB para fazer mover o canhão para a esquerda e para a direita, respetivamente. Por cada vez que um botão é premido, o canhão deverá movimentar-se apenas uma posição.

O botão de interrupção I2 faz disparar um míssil, que terá o símbolo '|'. Este desloca-se na vertical na coluna correspondente ao centro do canhão, no momento do disparo, partindo da linha 21, até à linha 0, ou até atingir um meteoro.

Sempre que um míssil é disparado, o canhão necessita de recargar um novo míssil, pelo que haverá um intervalo de tempo antes de ser possível um novo disparo.

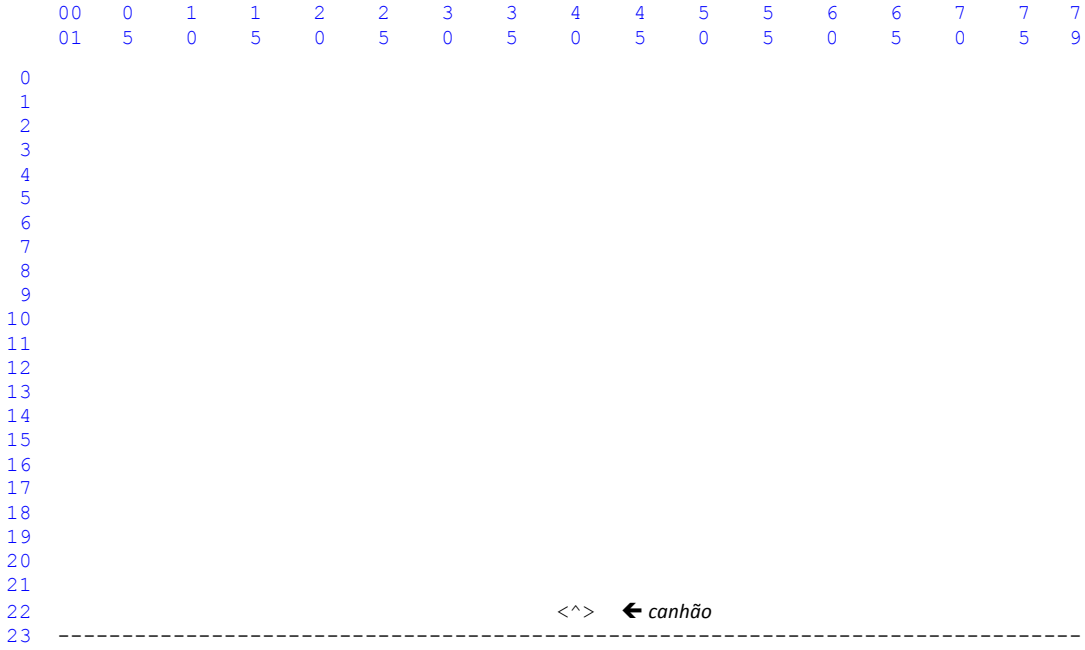

Figura 1 – Disposição inicial do espaço de jogo na Janela de Texto.

2.3 Meteoros

Os meteoros, simbolizados por '*', surgem na linha 0, numa coluna selecionada aleatoriamente entre os valores 1 e 78. Estes deslocam-se na vertical com velocidade constante.

Na versão base, apenas será gerado um novo meteoro quando o atual for destruído, ou por um disparo do canhão, ou por ter atingido a Terra (linha 23).

Por cada meteoro destruído será atribuída uma pontuação. O jogo termina quando três meteoros tiveram atingido a Terra.

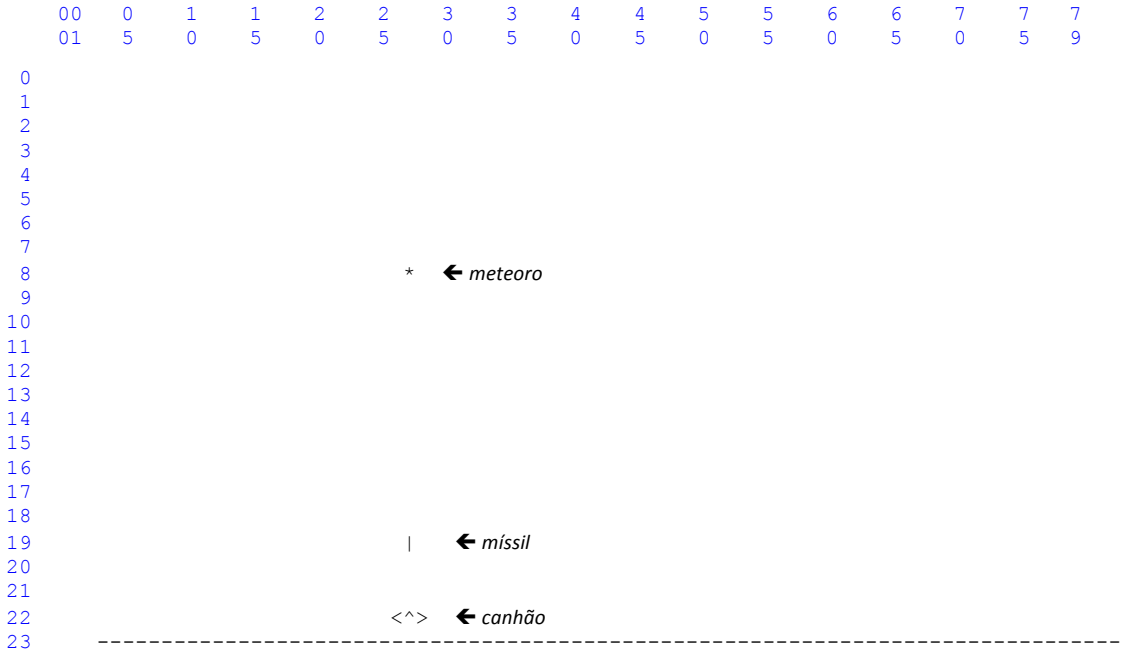

Figura 2 – Situação de jogo.

2.4 Dispositivos

O jogo propriamente dito decorre na janela de texto. No entanto, são usados também os seguintes dispositivos:

- o LCD deverá mostrar, na primeira linha, a pontuação atual (`xxxx`) e o número de mísseis que já atingiram a Terra (`z`), e na segunda linha a pontuação máxima atual (`YYYY`):

```
Pontos: XXXX *Z
Máximo: YYYY
```

- os *displays* de 7 segmentos deverão indicar o valor do último meteoro destruído;
- os leds são usados para indicar quanto tempo falta até o próximo míssil estar disponível. Os 16 leds acesos indica que se pode efetuar o disparo. Quando se carrega em I2, os leds devem apagar e ir acendendo um a um em intervalos regulares, do led 0 para o 15.

2.5 Início do Jogo

Quando arranca, o programa deverá escrever, centrado respetivamente nas linhas 12 e 14, as mensagens "`Bem-vindo à Chuva de Meteoros!`" e "`Prima o interruptor I1 para começar`".

Quando I1 for premido:

- a janela de texto deverá ficar como indicado na Figura 1;
- é gerado um primeiro meteoro, que depois terá o comportamento indicado na Secção 2.3;
- a pontuação atual deve ser colocada a 0;
- os 16 leds devem acender;
- os *displays* de 7 segmentos devem mostrar o valor 0.

2.6 Fim do Jogo

O jogo termina quando três meteoros atingem a Terra. Nesta altura, o programa deverá escrever, centrado respetivamente nas linhas 12 e 14, as mensagens "`Fim do Jogo`" e "`Prima o interruptor I1 para recomeçar`". A pontuação máxima deve ser também atualizada. Quando I1 for premido o jogo deve recomeçar.

2.7 Versões avançadas

Na versão base, apenas existirá um meteoro de cada vez. Só após este ser destruído, por um míssil ou por ter chegado à Terra, será gerado um novo meteoro. Nesta versão, a velocidade de todos os meteoros é a mesma.

Esta versão base pode ser estendida de duas formas, independentes entre elas:

- Existência de vários meteoros em simultâneo. Neste caso, quando o jogo tem início e é criado o primeiro meteoro, deverá ser calculado dentro de quanto tempo deverá ser criado o próximo meteoro, e assim sucessivamente.
- Velocidades diferentes para os meteoros. Quando um meteoro é criado, a sua velocidade é determinada de entre um conjunto de valores possíveis.

Naturalmente, estas variantes apenas devem ser consideradas depois da versão base estar afinada.

3 Implementação

Uma vez iniciado, o jogo consistirá num ciclo em que se atualizam as posições do canhão, meteoros e mísseis. Os movimentos são conseguidos apagando o símbolo da posição atual (*i.e.*, escrevendo um espaço por cima) e escrevendo-o na nova posição.

3.1 Movimentação do canhão

A posição do canhão é atualizada por cada vez que o jogador premir os interruptores I0 e IB, movendo uma posição para a esquerda ou para a direita, respetivamente. Este movimento implica a atualização dos três caracteres que definem o canhão. Naturalmente, é necessário verificar os limites da janela de texto.

3.2 Movimentação do meteoro

A nova posição do meteoro é sempre a posição a baixo da atual. O ritmo de atualização da sua posição é determinado pela sua velocidade, com as temporizações indicadas na Secção 3.4.

Antes da escrita na nova posição é necessário verificar se o meteoro atingiu a Terra, ou se foi atingido por um míssil, casos em que este meteoro desaparece.

3.3 Movimentação dos mísseis

O movimento dos mísseis é semelhante ao dos meteoros, no sentido inverso e com temporizações diferentes. Neste caso, é necessário verificar se o míssil já chegou ao topo da janela, ou se atingiu um meteoro, caso em que deve desaparecer.

3.4 Temporizações

O ritmo de atualização das posições dos meteoros e mísseis é controlado através do temporizador disponível no simulador do P3 (isto é, só deverá ser executado um novo ciclo de jogo após o temporizador expirar). Os valores sugeridos para as temporizações a utilizar no programa estão indicadas em baixo (nota: o simulador pode não conseguir atingir estes valores, pelo que o programa ficará mais lento do que o desejável):

Velocidade do míssil: 200ms entre cada movimento

Tempo para novo míssil: 4,4s

Pontuação: valor aleatório dentro do intervalo [10, 30]

Velocidade do meteoro:

versão base: todos com 200ms entre cada movimento
versão avançada: valor aleatório entre {100ms, 200ms, 300ms, 400ms}

Intervalo para novo meteoro (versão avançada): [1s,4s]

3.5 Valores aleatórios

Em diferentes fases do jogo será necessário obter um valor aleatório, por exemplo, a pontuação do meteoro destruído e a coluna onde aparece o meteoro. O Anexo A descreve um algoritmo para a geração de valores aleatórios de 16 bits.

4 Microprogramação

No âmbito do projeto será microprogramada uma nova instrução *assembly* que será usada para escrever um carácter na janela de texto. A sua utilização será:

```
I2OP op1,op2
```

em que:

- `I2OP` é o nome da nova instrução. É um nome genérico reservado no simulador para a implementação de uma nova instrução de 2 operandos, e tem o código de instrução (opcode) 101111 (2Fh).
- `op1`, é o carácter a escrever.
- `op2`, contém as coordenadas da janela de texto onde escrever o carácter, em que os oito bits mais significativos representam a linha e os oito bits menos significativos representam a coluna.

Nenhum dos operandos nem bits de estado deve ser alterado.

5 Plano de Desenvolvimento

5.1 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 projeto.
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 efetuam a interface com os dispositivos de entrada (interruptores) e os dispositivos de saída (janela de texto, LEDs, *display* 7 segmentos e LCD), com especial atenção à passagem de parâmetros entre estas rotinas e o programa principal.
4. 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.
5. Realize a ligação entre os vários procedimentos, de forma a obter o comportamento desejado e especificado.
6. Embora a escrita na janela de texto vá ser realizada pela nova instrução `I2OP`, comece por realizar uma implementação provisória deste procedimento utilizando uma rotina escrita em linguagem *assembly*. Deste modo poderá iniciar, o mais cedo possível, o desenvolvimento dos vários procedimentos envolvidos no controlo do jogo. Esta rotina deverá ser substituída pela invocação da nova instrução.
7. Comente e indente devidamente o código desenvolvido. Inclua nos comentários referências aos fluxogramas que irá entregar para auxiliar a leitura e compreensão do programa.

5.2 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 efetue os testes necessários para verificar o seu correto funcionamento. Não prossiga para a implementação de funcionalidades mais avançadas sem ter garantido que as que lhe servem de base estão corretamente implementadas.

Estando o sistema a funcionar corretamente, pode incluir eventuais funções opcionais que entretanto tenha desenvolvido.

6 Plano de Entrega

1ª Aula (8 a 12 de Abril)	
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.
Demonstração:	• O programa a apresentar deverá fazer o desenho do cenário de jogo, a colocação do canhão e o seu movimento.
2ª Aula (6 a 10 de Maio)	
Entrega:	• Microcódigo da instrução `I2OP`; • Código *assembly* do programa de teste da instrução `I2OP` realizada.
Demonstração:	• Execução da instrução `I2OP` utilizando um programa de teste.

3ª Aula (13 a 17 de Maio)	
Entrega:	• Breve relatório com a descrição do projeto 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 projeto 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).
Demonstração:	• Funcionamento do jogo concebido.

As duas primeiras entregas devem ser feitas num envelope identificado com o **dia e hora do turno** e **número do grupo**.

A entrega final deverá ser submetida no Fénix num ficheiro zip com o nome no formato `tAaBBgC.zip`, em que: `A` é o dia da semana (2 a 6); `BB` é a hora de início (basta a hora, com 2 digitos); `C` é o número do grupo. Este documento deverá conter:

- O relatório em **PDF**.
- Código, quer ficheiro fonte, quer um PDF gerado com a aplicação **p3print** fornecida na página da cadeira.

Notem que o Fénix terá uma hora limite para a submissão. Não serão aceites projetos por outro meio que não o Fénix.

O calendário das discussões será acordado com o docente do turno respectivo. A discussão terá lugar, preferencialmente, na semana de 20 a 24 de Maio.

Anexo A – Geração de sequência pseudoaleatória

O seguinte algoritmo gera uma sequência aparentemente aleatória de números de 16 bits, com distribuição uniforme (isto é, os números são equiprováveis), com um passo de repetição elevado:

```
Máscara = 1000 0000 0001 0110 b
if (n0 = 0)        /* Testa o bit de menor peso */
    Ni+1 = rotate_right (Ni);
else
    Ni+1 = rotate_right (XOR (Ni, Mascara));
```

Em cada invocação desta função lê-se o valor anterior N_i e gera-se um novo valor pseudoaleatório, N_{i+1}. A raiz desta sequência ($N_0 \neq 0$) pode ser obtida a partir de um parâmetro que varie com o decorrer do jogo (por exemplo, o número de ciclos de programa ou ciclos de espera entre a inicialização do programa e o início efetivo do jogo).