

Laboratório de Introdução à Arquitetura de Computadores

IST - Taguspark

2017/2018

Interação do processador com memória e periféricos

Guião 4

16 a 20 de outubro de 2017

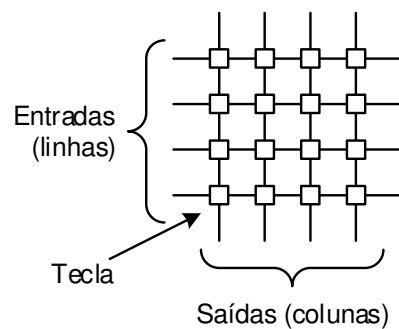
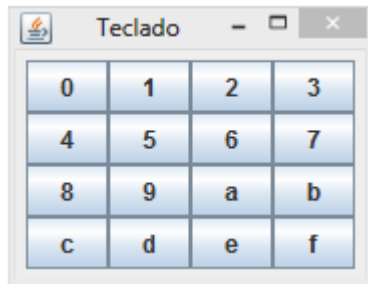
(Semana 5)

1 – Objetivos

Com este trabalho pretende-se que os alunos se iniciem na utilização do microprocessador PEPE (Processador Especial Para Ensino) para acesso a uma memória e periféricos (um teclado e dois displays de 7 segmentos).

2 – Funcionamento de um teclado

O teclado de 16 letras está organizado internamente como uma matriz de 4 linhas por 4 colunas. Neste caso as teclas disponíveis são os números 0 a 9 e ainda as letras de A a F. O que pretendemos fazer do ponto de vista do microprocessador é saber qual a tecla que foi premida (com o cursor em cima da tecla e fazendo clique, simulando o carregar na tecla com um dedo).



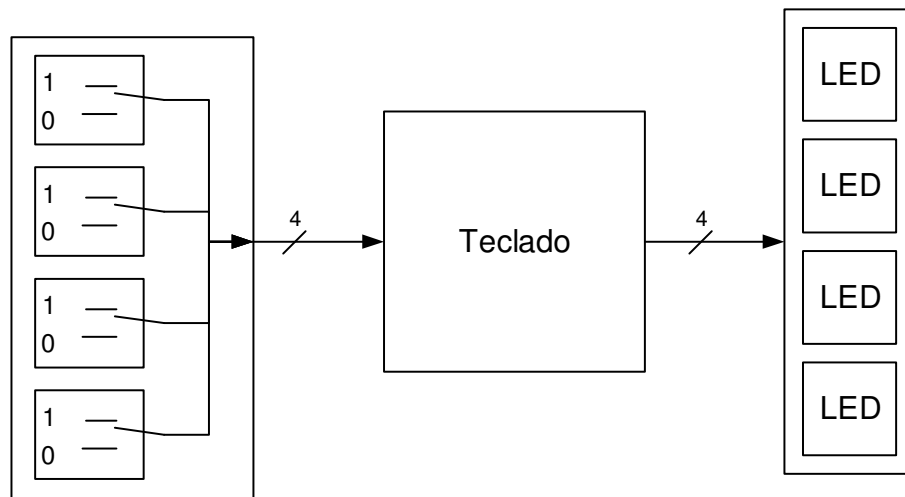
O teclado não indica diretamente qual a tecla que foi premida. Para descobrir qual foi, o programa tem de testar sucessivamente as várias linhas do teclado, para ver se alguma tecla foi premida. Quando tal acontecer, a tecla faz um curto-circuito entre a linha e a coluna que a definem, fazendo com que o bit presente na linha (seja 0, seja 1) apareça na coluna (saída). Coluna a que nenhuma linha ligue vale 0 no bit de saída.

Aplica-se sucessivamente os valores “0001”, “0010”, “0100” e “1000” em formato binário nas entradas (linhas). Para cada um dos valores anteriores as saídas do teclado (colunas) são lidas. No caso de ser aplicado o valor 0001b nas entradas, e se se obtiver nas saídas o valor 0001b, quer dizer que a tecla que foi premida foi a tecla “0”. Se o valor for 0010b, quer dizer que foi a tecla “1” que foi premida. Se for 0100b ou 1000b, a tecla premida será “2” ou “3”, respetivamente.

No caso de colocar na entrada o valor 0010b, para as 4 possíveis saídas obtêm-se quando se carrega numa das teclas “4”, “5”, “6” e “7”, respetivamente. Raciocínio idêntico aplica-se às teclas da 3ª e 4ª linhas do teclado.

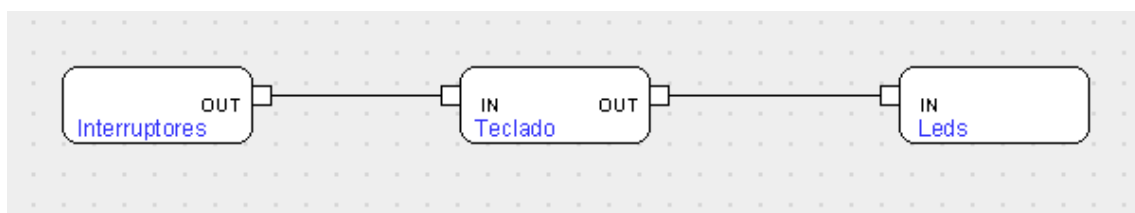
3 – Teste de um teclado

Primeiro vamos testar e verificar o funcionamento de um teclado com um circuito muito simples, descrito pela figura seguinte. Um conjunto de 4 interruptores permite injetar valores nas linhas do teclado (entradas) e um conjunto de 4 leds permite ver o estado das colunas (saídas).

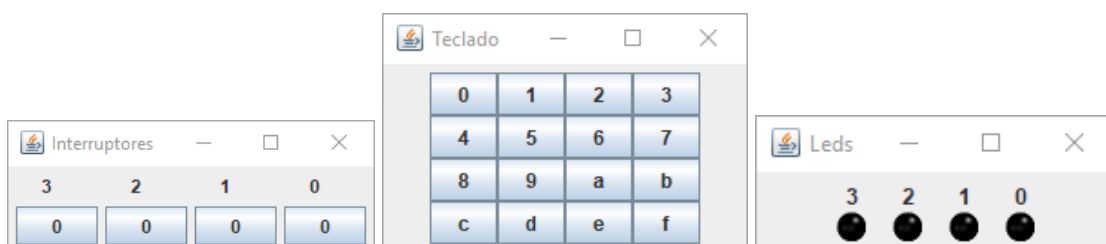


Um led acende apenas quando uma tecla é carregada e o bit injetado na linha da tecla está a 1.

O circuito que implementa este diagrama é dado já montado (ficheiro **lab4-1.cmod**).



Carregue este circuito no simulador (menu File e Load), passe para Simulação, abra as janelas de controlo de cada um destes dispositivos (com duplo clique – poderá ter de abrir mais estas janelas para ver o título) e carregue em Start no simulador.

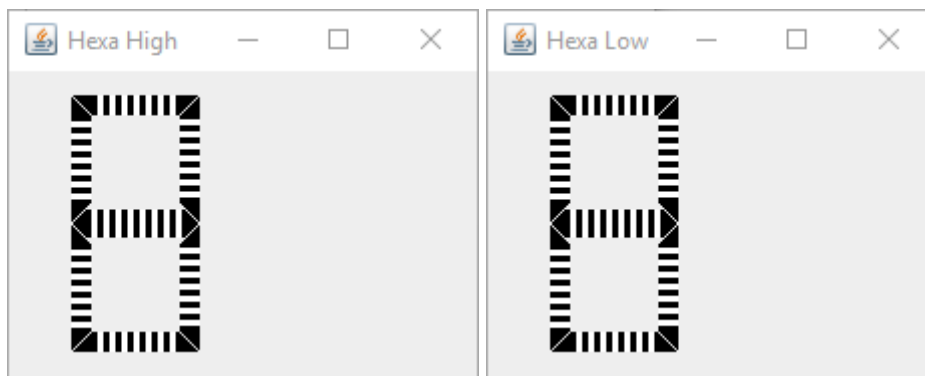


Experimente colocar o valor 0001b nos interruptores, clique nas várias teclas e verifique o que obtém na saída. Agora experimente sucessivamente com os valores 0010b, 0100b e 1000b, e verifique o que obtém na saída. Por fim, experimente com o valor 1111b na entrada, clique nas várias teclas e conclua sobre o funcionamento do teclado. Consegue detetar (pelos leds) qual a tecla carregada se houver mais do que um bit a 1 na entrada?

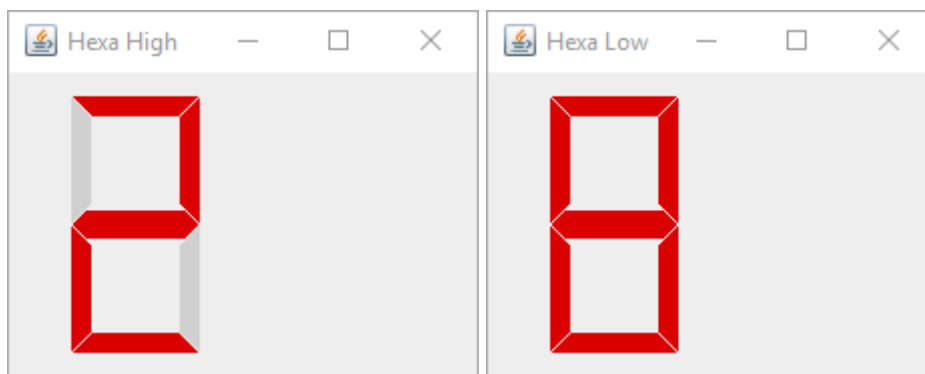
4 – Displays hexadecimais

Permitem visualizar dígitos hexadecimais, de 0 a F, ligando um ou mais dos 7 segmentos que compõem cada display. Têm apenas uma entrada, de 4 bits (para especificar uma das 16 hipóteses).

Para visualizar o valor, basta abrir o painel de controlo (duplo clique, em modo Simulação). Tem de se abrir um pouco os painéis para visualizar o título. Quando não estão inicializados, têm o seguinte aspeto:



Após a primeira escrita já ficam com um aspeto mais agradável, como por exemplo:

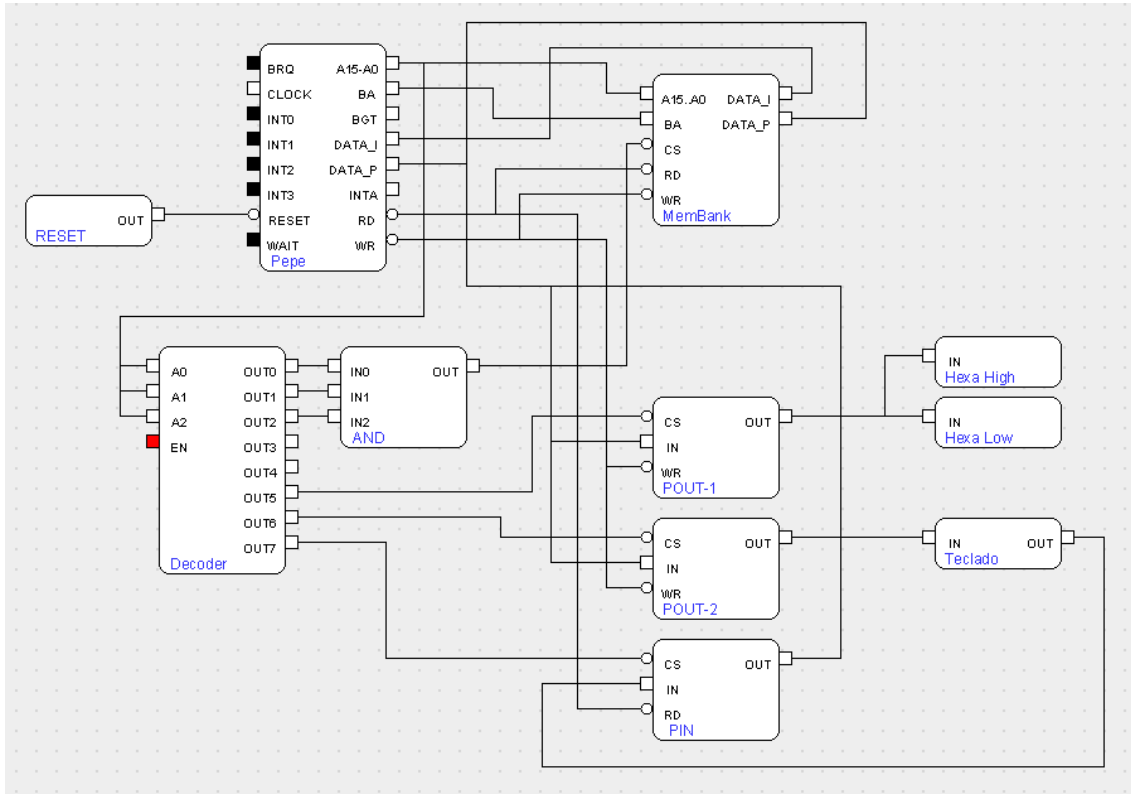


Os displays estão ligados ao periférico POUT-1 (de 8 bits) do circuito da secção seguinte. O Hexa Low liga aos quatro bits (nibble) de menor peso (daí o Low) e o Hexa High liga aos quatro bits (nibble) de maior peso (daí o High). Juntos, o display mostra os 8 bits, em hexadecimal, do periférico POUT-1.

5 – Leitura do teclado com o processador

5.1 – Circuito

O circuito para leitura do teclado com um programa é fornecido já montado (ficheiro **lab4-2.cmod**). Os ficheiros de ajuda sobre o PEPE estão disponíveis no site da cadeira.



São necessários os seguintes módulos:

- **PEPE** (cujo relógio é gerado internamente)
- **MemBank** (uma RAM de 16 bits com capacidade de endereçamento de byte)
- **POUT-1** (periférico de saída – liga aos dois displays)
- **POUT-2** (periférico de saída – liga às linhas do Teclado)
- **PIN** (periférico de entrada – liga às colunas do Teclado)
- **Teclado** (teclado de 4 linhas e 4 colunas)
- **Reset** (para inicializar o PEPE sempre que se faz reset no simulador)
- **Decoder e AND** (para permitir aceder aos dispositivos. Neste momento não é relevante. A descodificação de endereços será tratada no guião de laboratório 8)

NOTAS IMPORTANTES:

- Os acessos aos periféricos (quer em escrita quer em leitura) devem ser feitos com a instrução **MOVB**, pois estes periféricos são de apenas 8 bits;
- Os acessos à memória podem ser feitos quer com **MOV** (16 bits) ou **MOVB** (8 bits);







- A memória e periféricos estão disponíveis nos endereços indicados pela tabela seguinte:

Dispositivo	Endereços
RAM (MemBank)	0000H a 5FFFH
POUT-1 (porto de saída de 8 bits)	0A000H
POUT-2 (porto de saída de 8 bits)	0C000H
PIN (porto de entrada de 8 bits)	0E000H

5.2 – Programa de leitura do teclado

Abra o ficheiro **lab4.asm** (com um editor de texto do tipo NotePad++ para PC ou Brackets para Mac) e tente perceber o que faz, pelos comentários e pelo funcionamento descrito na secção 2.

De seguida, execute os seguintes passos:

1. Carregue o circuito do ficheiro **lab4-2.cmod** (menu File e Load) e passe para modo de simulação;
2. Abra o módulo PEPE e faça Compile&Load () do ficheiro **lab4.asm** (NÃO carregue em START no simulador);
3. Abra os painéis de controlo do teclado e dos dois displays hexadecimais (com duplo clique);
4. Execute o programa em modo contínuo () , o que faz automaticamente START ao simulador, e carregue numa tecla da linha 4 (a de baixo). Verifique que os displays exibem a linha (8) e a coluna (1, 2 4 ou 8) correspondentes à tecla em que carregou;
5. Verifique também que estes valores são exibidos apenas enquanto a tecla estiver a ser carregada. Verifique no programa o que é que permite ter esta distinção de comportamento, consoante a tecla está carregada ou não;
6. Verifique que em qualquer altura pode fazer clique numa instrução, o que define um *breakpoint* (ficando a instrução a roxo). Da próxima vez que o processador tentar executar essa instrução, a execução para, ficando a instrução a azul. Pode então executar-se uma instrução de cada vez, ou passo a passo, carregando no botão STEP () do PEPE. Em qualquer altura, pode voltar a executar em modo contínuo () . Note que não se consegue detetar teclas carregadas em execução passo a passo, pois o cursor do rato é só um (e ou se carrega em  ou no teclado).
7. Termine a execução do programa, carregando no botão  do PEPE;
8. Altere o valor da constante LINHA (no ficheiro **lab4.asm**) para 1, 2 ou 4. Guarde o ficheiro (no editor de texto) e volte ao passo 2 acima, verificando que agora o programa deteta teclas noutra linha (indicada pela constante LINHA).

6 – Objetivos a cumprir para o Projeto

O programa do ficheiro **lab4.asm** dá apenas para uma tecla numa linha. Pretende-se detetar qualquer tecla em qualquer linha. O teclado é um componente essencial para a execução do projeto.

Na semana seguinte à deste guião, deve ser mostrar ao docente um programa que faça o varrimento das quatro linhas do teclado e detete a tecla premida, obtendo o seu valor (0 a FH).

Para tal, deve fazer os passos seguintes:

- Faça um programa (pode usar o programa do ficheiro **lab4.asm** como base) que varre continuamente, em ciclo, as várias linhas do teclado, verificando se alguma das teclas foi premida e, quando tal acontecer, sai do ciclo com a linha e a coluna da tecla em dois registos (use um *breakpoint* para verificar se os valores obtidos estão certos);
- Na realidade, o que quer obter é um valor entre 0 e FH, correspondente ao valor da tecla carregada, em vez de apenas informação separada sobre a linha e coluna da tecla. Acrescente código para converter esta informação no valor entre 0 e FH. Sugestão: o valor da tecla é igual a $4 * \text{linha} + \text{coluna}$, em que tanto linha como coluna são números entre 0 e 3. Logo, terá de converter 0001b, 0010b, 0100b e 1000b (isto é, 1, 2, 4 e 8, valores possíveis quer para a linha, quer para a coluna) em 0, 1, 2 e 3, respetivamente. Tal pode ser feito contando o número de SHR que se tem de fazer ao valor da linha (ou da coluna) até este ser zero (use um *breakpoint* para verificar se o valor obtido está certo, e se não estiver reinicie o programa e corra-o em *single-step* desde o *breakpoint* do ponto anterior, verificando os registos relevantes em cada passo);
- Acrescente código para mostrar o valor da última tecla carregada no display Low (o display High fica a zero), em ciclo. Sempre que uma tecla é detetada e o seu valor mostrado, o programa deve voltar a esperar a deteção de uma nova tecla;
- Mas, mesmo realmente, o que quer é usar as teclas detetadas para executar funcionalidades no projeto. Simule esta capacidade fazendo com que os displays High e Low mostrem agora o valor de um contador hexadecimal (valor de um registo), que é incrementado de duas unidades quando se carrega numa dada tecla (escolha qual) e decrementado de uma unidade quando se carrega numa dada tecla (escolha qual). Por cada tecla carregada, o contador só deve variar uma vez.

A tabela A.9 do livro contém informação sobre cada instrução do PEPE.

O objetivo deste exercício é fazer o software básico de leitura do teclado, que constitui o coração do controlo do programa do projeto. Será apenas questão de, mais tarde, adaptar o que se faz com o valor da tecla lida.