17. ESPAÇO DE ENTRADA/SAÍDA

De entre as tarefas próprias de um processador, destaca-se neste capítulo a interlocução com o exterior, no sentido de providenciar os meios que permitam receber, processar e expedir informação de acordo com algoritmos que lhe são determinados por programas carregados em memória. Esta interlocução é estabelecida através de dispositivos periféricos, tais como teclados, monitores, discos, sensores e actuadores, etc., de onde o processador escreve ou lê informação. A interação com este tipo de dispositivos periféricos, para escrita ou leitura de dados, é realizada através de portos de entrada/saída (E/S), usualmente designados por I/O (do inglês *Input*/*Output*), que são acedidos directamente pelo processador. No caso do P16, não há distinção entre os modos de acesso ao espaço de memória e ao espaço de I/O, uma vez que ambos são mapeados no mesmo espaço de endereçamento (único e genérico). Assim, os dispositivos periféricos, mapeados no espaço de I/O, são tratados como dispositivos de memória. Esta particularidade tem como vantagem o facto de não serem necessárias instruções específicas para suportar a entrada/saída de dados. A eventual desvantagem surge no acréscimo de complexidade do módulo de descodificação de endereços, devido à disparidade de dimensões que se verifica entre os dispositivos de memória e os de I/O.

17.1 Portos paralelos de entrada e saída

Havendo diversas formas para suportar a interlocução com dispositivos periféricos, os portos de comunicação em modo paralelo, que são particularmente eficientes a muito curtas distâncias (poucos metros), constituem o exemplo mais simples e exemplificativo dos mecanismos envolvidos ao mais baixo nível.

Denomina-se como porto paralelo um porto, de entrada ou de saída, que permite a leitura ou escrita, respetivamente, de vários bits em simultâneo. Estes portos podem ser utilizados para controlo individual de actuadores, observação instantânea de sensores ou troca de informação entre sistemas. Na troca de informação entre sistemas é necessário recorrer a protocolos de validação e controlo de fluxo, cuja explicação não se inclui neste capítulo.

17.1.1 Porto paralelo de saída

Um porto paralelo de saída de *n* bits consiste, fundamentalmente, num registo de *n* bits cujas saídas estão disponíveis em pinos acessíveis no exterior do sistema e no respectivo circuito de controlo. Acessível no espaço de endereços do processador, é possível escrever, instantaneamente, no porto palavras de dados com a referida dimensão. Uma vez escrita no registo, essa informação fica disponível (estável e inalterada) até nova escrita.

O circuito digital que implementa o porto de saída assegura imediatamente a impedância e os níveis de tensão adequados para interligar com os circuitos lógicos digitais mais comuns (TTL, etc.). Para outro tipo de circuitos, a adaptação será conseguida através da instalação de componentes electrónicos apropriados, habitualmente designados por *buffers* ou *line drivers*.

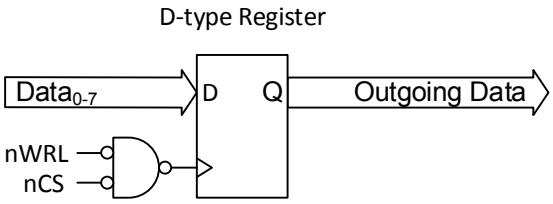

Figura 17-1 – Porto Paralelo de Saída de 8 bits

Na Figura 17-1 apresenta-se a estrutura típica de um porto paralelo de saída de 8 bits, que está ligado à interface de barramento do P16. O registo poderá ter característica *latch* ou *edge-triggered*, desde que a memorização se dê no momento em que a informação se encontra estável no barramento de dados do sistema emissor ($Data_{0-7}$), ou seja, no final do impulso do sinal nWRL, durante a evocação do endereço do respectivo porto, reflectido no sinal nCS.

17.1.2 Porto paralelo de entrada

Um porto paralelo de entrada de *n* bits é constituído, fundamentalmente, por um componente capaz de garantir a comutação de impedância na ligação ao barramento de dados do sistema receptor (i.e. o processador) e pelo respectivo controlo. Tais componentes são comummente conhecidos por *tri-state buffer*s. Acessível no espaço de endereços do processador, quando endereçado, o porto de entrada estabelece os valores de tensão do referido barramento ($Data_{0-15}$), possibilitando assim a leitura dos dados presentes nos *n* pinos de entrada (*Incoming Data*). Sempre que não esteja endereçado (todo o restante tempo), o porto mantém-se em alta impedância – condição semelhante a circuito aberto –, libertando assim o barramento.

Na Figura 17-2 apresenta-se a estrutura base de um porto paralelo de entrada de 16 bits, que está ligado à interface de barramento do P16. A ligação entre o exterior e o barramento interno está delimitada no tempo pelo sinal nRD e condicionada à evocação do endereço do porto, estabelecida pelo sinal nCS.

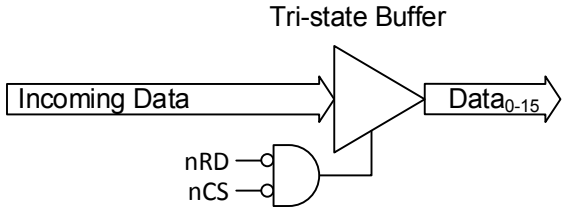

Figura 17-2 – Porto Paralelo de Entrada de 16 bits

17.2 Exercício

Utilizando o sistema SDP16, realize um sistema para controlo de uma lâmpada temporizada L, accionada por um botão B. O sistema tem o seguinte comportamento: ao premir-se o botão B, se a lâmpada L estiver apagada, ela acende-se e assim permanece durante dez segundos; se a lâmpada já se encontrar acesa, apaga-se de imediato. Admita que o botão B está ligado ao bit de peso 2 do porto de entrada, localizado no endereço 0xFF00, a lâmpada L está ligada ao bit de menor peso do porto de saída, igualmente localizado no endereço 0xFF00, e a frequência do sinal de relógio aplicado ao P16 é 50 kHz (T=20 μs).

No programa seguinte apresenta-se uma solução, escrita em linguagem C. O código entre as linhas quatro e sete é responsável pela deteção do pressionar do botão, evento que origina um acesso ao porto de saída para acender a lâmpada (linha oito). O código entre as linhas nove e treze garante que a lâmpada permanece acesa, por 10 s, no máximo, podendo ser apagada antes deste tempo se o botão for novamente pressionado. As funções auxiliares **`port_input`** e **`port_output`** realizam, respetivamente, a leitura do porto de entrada e a escrita no porto de saída.

```
1   int main() {
2        while (true) {
3             port_output(0); /* turns off the lamp */
4             while ((port_input() & BUTTON_MASK) != 0)
5                  ;
6             while ((port_input() & BUTTON_MASK) == 0)
7                  ;
8             port_output(LAMP_MASK); /* turns the lamp on */
9             delay_counter = TIME_DELAY;
10            while ((port_input()& BUTTON_MASK)!=0 && --delay_counter > 0)
11                 ;
12            while ((port_input()& BUTTON_MASK)==0 && --delay_counter > 0)
13                 ;
14       }
15  }
```

Na implementação em linguagem *assembly* P16 poderão observar-se os detalhes de implementação ao nível do processador.

```
        .equ    SDP16_PORTS_ADDRESS, 0xFF00
        .equ    BUTTON_MASK, 2
        .equ    LAMP_MASK, 1
        .equ    TIME_DELAY, 16600

        .section .startup
        b       _start
        b       .
```

```
_start:
        ldr     sp, stack_top_addr
        bl      main
        b       .

stack_top_addr:
        .word stack_top

        .text
main:
        push    lr
while:
        mov     r0, #0
        bl      port_output             ; port_output(0)
        mov     r2, BUTTON_MASK
while1:                                 ; while ((port_input()
        bl      port_input              ; & BUTTON_MASK) != 0)
        and     r0, r0, r2
        bzc     while1
while2:                                 ; while ((port_input()
        bl      port_input              ;     & BUTTON_MASK) == 0)
        and     r0, r0, r2
        bzs     while2
        mov     r0, LAMP_MASK
        bl      port_output             ; port_output(LAMP_MASK)
        mov     r3, TIME_DELAY & 0xff   ; delay_counter = TIME_DELAY
        movt    r3, TIME_DELAY >> 8
while3:
        bl      port_input              ; while ((port_input()
        and     r0, r0, r2              ;    & BUTTON_MASK) != 0
        bzs     while3_end
        sub     r3, r3, #1                  ;   && --delay_counter > 0
        bzc     while3
while3_end:
while4:
        bl      port_input              ; while ((port_input()
        and     r0, r0, r2              ;   & BUTTON_MASK) == 0
        bzc     while4_end
        sub     r3, r3, #1                  ;   && --delay_counter > 0
        bzc     while4
while4_end:
        b       while
        pop     pc

        .section .stack
        .space  64                        ; STACK_SIZE
stack_top:
```

Nesta implementação, as funções auxiliares são implementadas da seguinte forma:

```
port_input:
        ldr     r1, port_addr
        ldrb    r0, [r1]
        mov     pc, lr

port_output:
        ldr     r1, port_addr
        strb    r0, [r1]
        mov     pc, lr

port_addr:
        .word   SDP16_PORTS_ADDRESS
```