17. ESPAÇO DE I/O (ENTRADA/SAÍDA)

O objectivo principal de um CPU é providenciar interlocução com o exterior, no sentido de 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, monitor, discos, etc., onde o CPU escreve e lê informação. A escrita e a leitura nos dispositivos periféricos é realizada através de portos de entrada/saída (Input/Output), e que são acedidos pelo CPU de uma forma semelhante à que acede aos dispositivos de memória. Na verdade o PDS16_V1 não distingue o espaço de memória do espaço de I/O, ou seja, o I/O é tratado como um dispositivo de memória. Esta funcionalidade tem como vantagem o facto de não ser necessário disponibilizar instruções específicas para a entrada e saída de dados. A desvantagem é tornar a descodificação de endereços mais complexa devido à heterogeneidade existente entre as dimensões dos dispositivos de memória e os de I/O.

17.1 Portos paralelos de entrada e saída

Denomina-se porto paralelo a um porto de entrada ou de saída, que disponibiliza vários bits em simultâneo. Estes portos poderão 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 um protocolo de validação e controlo de fluxo, ou seja, um conjunto de regras estabelecidas entre os interlocutores que assegure a correcta transferência de informação entre ambos.

17.1.1 Porto paralelo de saída

Um porto paralelo de saída é constituído por um registo inserido no espaço de endereços do CPU no qual é possível escrever uma palavra. A palavra escrita no registo fica disponível em pinos através de um andar de saída, que lhe confere uma impedância capaz de interligar o CPU com os circuitos lógicos digitais mais comuns (Ex. TTL) bem como outro tipo de electrónica. A **Error! Reference source not found.**Figura 17-1 apresenta a estrutura característica de um porto de saída de 8 bits.

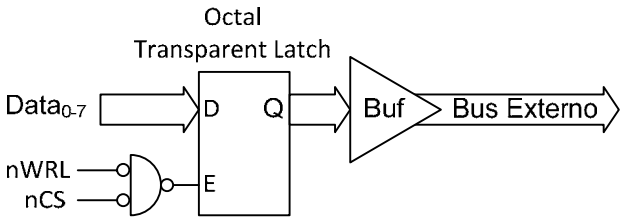

Figura 17-1 – Porto de Saída de 8 bits

17.1.2 Porto paralelo de entrada

Um porto paralelo de entrada, é constituído por um buffer TRISTAT que, quando endereçado pelo CPU, coloca em baixa impedância no bus de dados do CPU a informação presente nos pinos de entrada do porto. A Figura 17-2 apresenta a estrutura base de um porto de entrada paralelo de 16 bits.

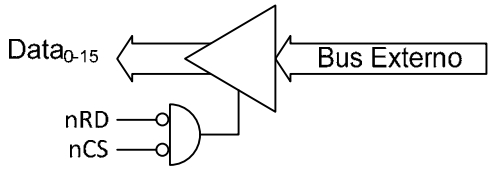

Figura 17-2 – Porto de Entrada de 16 bits

17.1.3 Exercício

Utilizando o processador PDS16, realizar um sistema para controlo de uma lâmpada temporizada L, accionada por um botão B. Ao premirmos o botão B o sistema tem o seguinte comportamento: se a lâmpada L se encontrar apagada, acende-se de imediato e permanece acesa durante 10 segundos, se já se encontrar acesa apaga-se de imediato.

Admita que: o botão B está ligado ao bit de peso 2 de um porto de entrada de 8 bits, localizado no endereço 0x8000, a lâmpada L está ligado ao bit de menor peso de um porto de saída de 8 bits, localizado no endereço 0x8000 e o CPU tem um *clock* de 1KHz (T=1ms).

```
        .EQU   B_POSITION,3
        .EQU   DELAY_5S,1000/50
        .code
main:   ldih   r0,#0x80
        ldi    r1,#0
        st     r1,[r0,#0]     ;apaga a lâmpada L
;detectar transição ascendente em B
main_1: ld     r1,[r0,#0]
        shr    r1,r1,#B_POSITION,0
        jc     main_1
main_2: ld     r1,[r0,#0]
        shr    r1,r1,#B_POSITION,0
        jnc    main_2
;detectou transição ascendente em B
        ldi    r1,#1
        st     r1,[r0,#0]     ;acende a lâmpada L
        ldi    r1,#DELAY_5S
        st     r1,time        ;prepara tempo de 5 segundos
;aguarda nova transição ascendente em B ou fim de tempo
main_3: jmpl   delay          ;4ms + delay 24ms
        jc     main           ;4ms
        ld     r1,[r0,#0]     ;6ms
        shr    r1,r1,#B_POSITION,0  ;4ms
        jc     main_3         ;4ms
main_4: jmpl   delay          ;4ms total=50ms
```

```
        jc      main
        ld      r1,[r0,#0]
        shr     r1,r1,#B_POSITION,0
        jnc     main_4
;detectou transição ascendente em B
        jmp     main
delay:  ld      r2,time         ;6ms
        dec     r2              ;4ms
        st      r2,time         ;6ms
        nop                     ;4ms
        ret                     ;4ms
        .data
time:   .space    1
```

17.2 GPIO (*General Propose Input Output*)

Em sistemas baseados em microprocessadores é usual dispormos de dispositivos de entrada/saída em paralelo com uma arquitectura programável. A arquitectura ser programável/configurável, permite uma fácil adaptação às especificidades do sistema que pretendamos construir. A título de exemplo consideremos um GPIO (*General Propose Input Output*) como é mostrado na Figura 17-3 e que denominaremos por GPIO_V1. O GPIO_V1 está mais adaptado a aplicações de controlo, uma vez que não põe disponíveis sinais de protocolo.

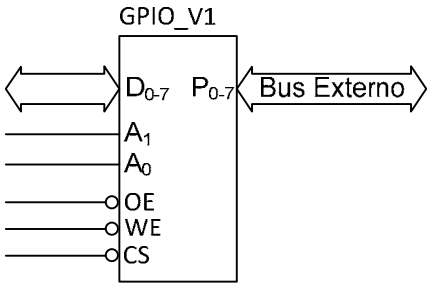

Figura 17-3 – diagrama do GPIO_V1

O GPIO_V1 apresenta a seguinte especificação:

- Cada pino pode ser configurado como entrada ou saída através da escrita no registo CTR. Esta funcionalidade permite uma boa adaptação em termos de número de pinos necessários para entrada e saída. Cada um dos 8 bits do registo CTR determina se o bit do porto é de entrada ou de saída.

- O porto de saída disponibiliza três endereços: um para escrita de uma palavra (*byte wide*), outro para colocar pinos ao valor lógico "1" (*set*), e um terceiro para colocar pinos ao valor lógico "0" (*clear*). Esta funcionalidade diminui a complexidade do programa e o tempo necessário para colocar a um ou a zero um ou mais pinos, sem interferir com os restantes.

Na Tabela 17-1 são apresentados os endereços e o significado de cada bit do registo de CTR e das várias acções sobre os bits do registo de saída (WIDE, SET e CLR).

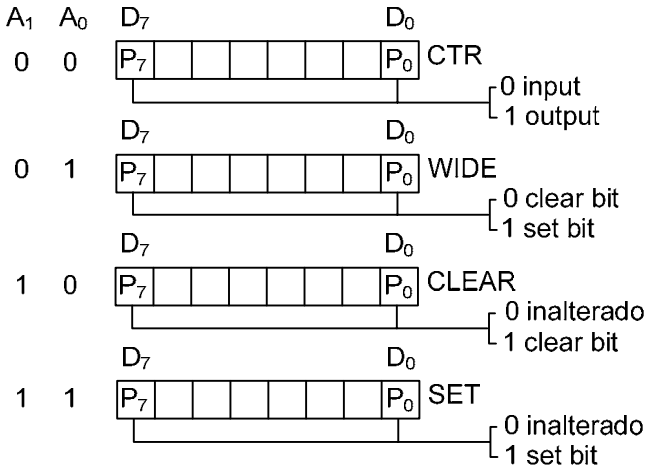

Tabela 17-1 – Registos aplicativos do GPIO_V1

Na Figura 17-4 é apresentada o diagrama de blocos de uma possível arquitectura para o GPIO_V1. Como se pode observar, quando realizamos a leitura do porto também se lêem os valores presentes nos bits configurados como saídas. A implementação assíncrona mostrada na Figura 17-4 só é possível se o CPU garantir a estabilidade do bus de dados antes da activação do sinal nWE, de outro modo é possível realizar acções *set* ou *reset* em células não seleccionadas.

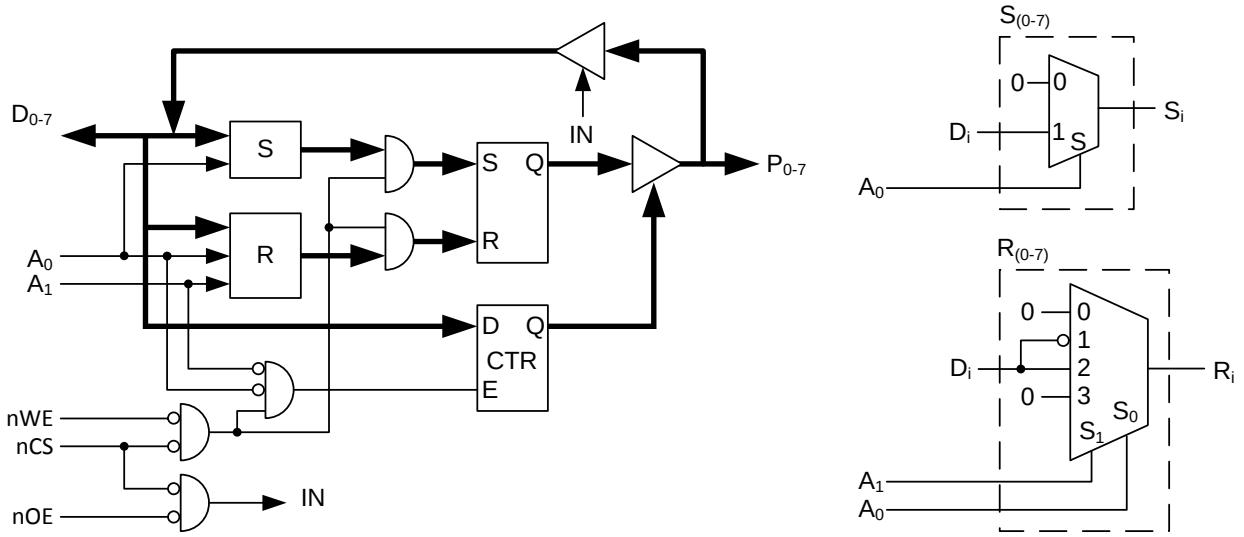

Figura 17-4 – Diagrama de blocos do GPIO_V1 implementação assíncrona

Como o PDS16 não garante a estabilidade do bus de dados antes da activação do sinal WE é necessário que implementação seja síncrona. Na Figura 17-5 é apresentada o diagrama de blocos duma possível implementação utilizando por exemplo uma PAL que disponibilize saídas registadas com *flip-flops* tipo D *edge-trigger*.

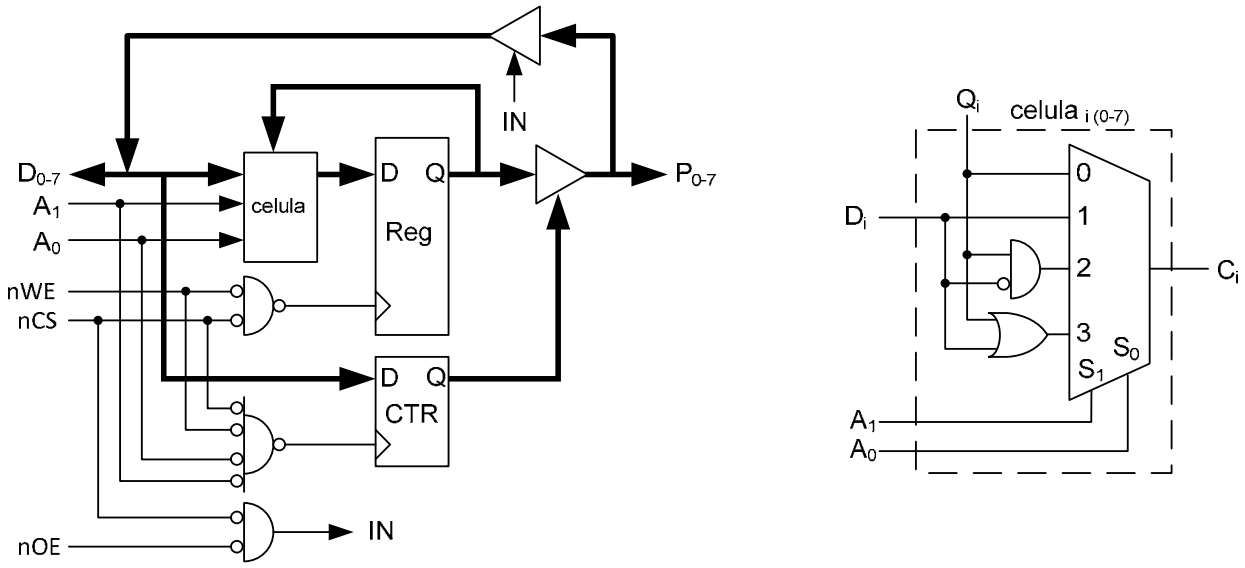

Figura 17-5 – Implementação síncrona do GPIO_V1

17.2.1 Exercício

Sobre uma fila de 4 LEDs, deslocar para a direita ou para a esquerda um LED aceso ao ritmo das transições ascendentes do sinal CLK. O sentido de deslocação é estabelecido pelo sinal DIR que, quando a zero, indica deslocamento para a direita. Existe ainda um LED denominado MAX que, quando aceso, indica que a fila já se deslocou consecutivamente na presente direcção mais que quatro vezes. O MAX apaga quando existe mudança de direcção. Utilizaremos para a fila de LEDs os bits de menor peso do GPIO (P0 a P3), o LED MAX em P4, o sinal CLK em P5 e o sinal DIR em P6.

```
.equ  gpio_base_addr,0xFF
.equ  GPIO_CTR_ADDR,0
.equ  GPIO_WIDE_ADDR,1
.equ  GPIO_SET_ADDR,2
.equ  GPIO_CLR_ADDR,3

.equ  MAX_POS,5
.equ  CLK_POS,6
.equ  DIR_POS,7
.equ  DIR_MASK,0x40
.equ  MAX_MASK,0x20

.section start
.org  0
jmp   main
```

```
      .section main
      .org  16
;void initGPIO(char * ptr); inicia o GPIO com os bits de 0 a 4 output e 5 a 7
input
initGPIO:
      ldi   r1,#0xff
      stb   r1,[r0,#GPIO_WIDE_ADDR]
      ldi   r1,#00011111b                ;programar P0-4 output, P5-7 input
      stb   r1,[r0,#GPIO_CTR_ADDR]
      ret
; void getClkAscTrans(char * ptr); detecta transição ascendete do CLK
getClkAscTrans:
      ldb   r1,[r0,#GPIO_WIDE_ADDR]      ;input P0-7
      shr   r1,r1,#CLK_POS,0        ;cy=CLK
      jc    getClkAscTrans               ; while(CLK)
clkT_1:
      ldb   r1,[r0,#GPIO_WIDE_ADDR]      ;input P0-7
      shr   r1,r1,#CLK_POS,0        ;cy=CLK
      jnc   clkT_1                       ; while(!CLK)
      ret
; void setBitMax(char * ptr); set do bit MAX
setBitMax:
      ldi   r1,#MAX_MASK
      stb   r1,[r0,#GPIO_SET_ADDR]
      ret

; void clrBitMax(char * ptr); clear do bit MAX
clrBitMax:
      ldi   r1,#MAX_MASK
      st    r1,[r0,#GPIO_CLR_ADDR]
      ret
```

```
;void roda(char port_in,char * ptr);roda esquerda direita (esquerda se DIR=0)
roda: shr   r1,r1,#DIR_POS,0
      jc    esquerda
      ld    r2,roda_image     ;roda para adireita
      rrl   r2,r2,#1
      st    r2,roda_image
roda_1:
      shl   r2,r2,#12,0 ; roda_image & 0x0f
      shr   r2,r2,#12,0
      ldi   r3,#MAX_MASK      ; port & MAX_MASK
      anl   r1,r1,r3
      orl   r2,r2,r1
      st    r2,[r0,#GPIO_WIDE_ADDR] ;  port=  (roda_image  &  0x0f)  |  (port  &
MAX_MASK)
      ret
esquerda:
      ld    r2,roda_image
      rrl   r2,r2,#15         ; equivale a rodar para a esquerda uma vez
      st    r2,roda_image
      jmp   roda_1

main: ldih  r0,#gpio_base_addr
      jmpl  initGPIO
      ldi   r1,#00010001b
      shl   r2,r1,#8,0
      orl   r1,r1,r2
      st    r1,roda_image                ;roda_image=0001000100010001b
      ldi   r1,#1
      stb   r1,[r0,#GPIO_WIDE_ADDR] ;LED da direita aceso
      ldi   r1,#0                   ;iniciar contador de deslocamentos
      st    r1,cont_max
      ld    r1,[r0,#GPIO_WIDE_ADDR]
      st    r1,old_dir
main_1:
      jmpl  getClkAscTrans
      ld    r1,[r0,#GPIO_WIDE_ADDR]
      ld    r2,old_dir
      st    r1,old_dir        ;old_dir=dir
      xrl   r1,r2,r1
      shr   r1,r1,#DIR_POS,0
      jc    diferente
      ldi   r4,#0
      st    r4,cont_max
      jmpl  clrBitMax
main_2:
      ld    r1,old_dir
      jmpl  roda
      jmp   main_1
diferente:
      ld    r1,cont_max
      inc   r1
      st    r1,cont_max
      ldi   r2,#4
      sub   r1,r2,r1
      jb    main_2
      jmpl  setBitMax
```

```
      jmp   main_2

      .section direct_data
      .org  8
old_dir:
      .space      1
cont_max:
      .space      1
roda_image:
      .space      1

      .end
```