Trabalho prático N.º 3

Objetivos

- Conhecer a estrutura básica e o modo de configuração de um porto de I/O no microcontrolador PIC32.
- Configurar em *assembly* os portos de I/O do PIC32 e aceder para escrever/ler informação do exterior.

Introdução

O microcontrolador PIC32 disponibiliza vários portos de I/O, com várias dimensões (número de bits), identificados com as siglas **`RB`**, **`RC`**, **`RD`**, **`RE`**, **`RF`** e **`RG`**. Cada um dos bits de cada um destes portos pode ser configurado, por programação, como entrada ou saída. Um porto de I/O de **`n`** bits do PIC32 é então um conjunto de **`n`** portos de I/O de 1 bit, independentes. Por exemplo, o bit 0 do porto E (designado por **`RE0`**) pode ser configurado como entrada e o bit 1 do mesmo porto (**`RE1`**) ser configurado como saída.

A Figura 1 apresenta o diagrama de blocos simplificado de um porto de I/O de 1 bit, no PIC32.

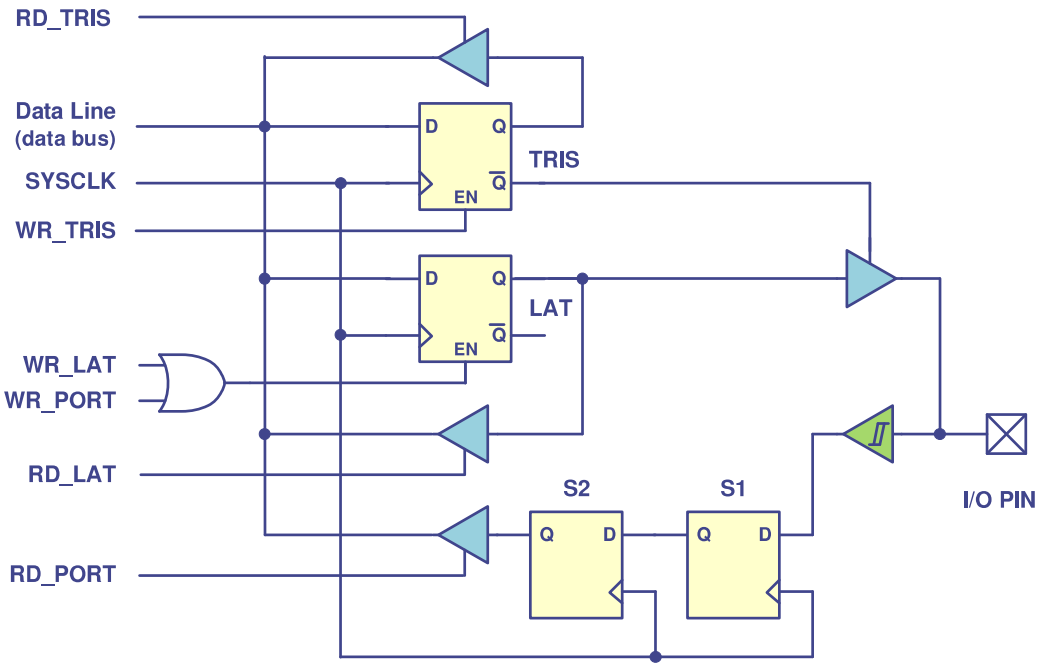

Figura 1. Diagrama de blocos simplificado de um porto de I/O de 1 bit, no PIC32.

A configuração de cada um dos bits de um porto como entrada ou saída é efetuada através dos flip-flops **`TRISxn`**, em que **`x`** é a letra identificativa do porto e **`n`** o bit desse porto que se pretende configurar. Por exemplo, para configurar o bit 0 do porto E (**`RE0`**) como entrada, o bit 0 do registo **`TRISE`** deve ser colocado a 1 (i.e. **`TRISE0=1`**); para configurar o bit 1 do porto E (**`RE1`**) como saída, o bit 1 do registo **`TRISE`** deve ser colocado a 0 (**`TRISE1=0`**).

Em termos de modelo de programação, cada porto tem associados 12 registos (4 de controlo e 8 de dados) de 32 bits em que apenas os 16 menos significativos (ou um subconjunto destes, dependendo do porto) têm informação útil. Desse conjunto de registos apenas usaremos 3: **`TRISx`**, **`PORTx`** e **`LATx`**. O registo **`TRISx`** é usado para configurar os portos como entrada ou saída, o registo **`LATx`** é usado para escrever um valor num porto configurado como saída e o **`PORTx`** para ler o valor de um porto configurado como entrada.

Os registos **TRISx**, **PORTx** e **LATx** estão mapeados no espaço de endereçamento de memória (área designada por **SFRs**), em endereços pré-definidos (disponíveis nos manuais do fabricante). O acesso para leitura e escrita desses registos é efetuado através das instruções **LW** e **SW** da arquitetura MIPS. Por exemplo, o endereço atribuído ao registo **TRISE** é **0xBF886100**, ao registo **PORTE** é **0xBF886110** e ao registo **LATE** é **0xBF886120**.

```
.equ SFR_BASE_HI, 0xBF88     # 16 MSbits of SFR area
.equ TRISE, 0x6100           # TRISE address is 0xBF886100
.equ PORTE, 0x6110           # PORTE address is 0xBF886110
.equ LATE,  0x6120           # LATE  address is 0xBF886120
```

Uma vez que um registo incorpora a informação individual de todos portos de 1 bit a que esse registo diz respeito, a modificação do valor de 1 bit (ou conjuntos de bits) tem que ser efetuada sem alterar os restantes. Ou seja, para se alterar um conjunto restrito de bits nestes registos é obrigatório usar uma sequência de instruções do tipo "*read-modify-write*". Por exemplo, se se pretender configurar os bits 0 e 3 do porto E (**RE0** e **RE3**) como saídas teremos de colocar a 0 apenas os bits 0 e 3 do registo **TRISE**, mantendo os restantes inalterados. Isso traduz-se na seguinte sequência de instruções (em conjunto com as definições anteriores):

```
lui     $t1, SFR_BASE_HI     #
lw      $t2, TRISE($t1)      # READ  (Mem_addr = 0xBF880000 + 0x6100)
andi    $t2, $t2, 0xFFF6     # MODIFY (bit0=bit3=0 (0 means OUTPUT))
sw      $t2, TRISE($t1)      # WRITE (Write TRISE register)
```

Para colocar a saída do porto **RE0** a 0 e do **RE3** a 1 (sem alterar os restantes) pode fazer-se:

```
lui     $t1, SFR_BASE_HI     #
lw      $t2, LATE($t1)       # READ (Read LATE register)
andi    $t2, $t2, 0xFFFE     # MODIFY (bit0 = 0)
ori     $t2, $t2, 8          # MODIFY (bit3 = 1)
sw      $t2, LATE($t1)       # WRITE (Write LATE register)
```

Como auxiliar de memória, note que:

- **TRIS** é relativo a *tri-state* ('0' => *tri-state off*, i.e., o porto não está no estado de alta impedância, ou seja, é um porto de saída; '1' => *tri-state on* , i.e., o porto está no estado de alta impedância, ou seja, é uma entrada);
- **PORT** diz respeito ao valor do porto de entrada;
- **LAT** refere-se a *latch*, i.e., ao registo que armazena o valor a enviar para as saídas.