

Arquitetura de Computadores II

1. O sistema de Entradas/Saídas (1)

António de Brito Ferrari
ferrari@ua.pt

Ligação de Dispositivos Externos (Periféricos) a um sistema de computação

- **Periféricos:**

- Grande variedade de tipos de operação
 - Impraticável incorporar no processador a lógica necessária para controlar uma gama diversificada de periféricos
- Velocidade de transferência de dados frequentemente muito menor do que a do processador e da memória
 - Impraticável usar o processador para comunicar diretamente com o periférico
- Usam frequentemente formatos de dados e comprimentos de palavra diferentes dos do processador

➤ Necessário **I/O Modules (I/O Adapters)**

Processador \longleftrightarrow Módulo de I/O

- **Descodificação dos comandos**
 - O módulo de I/O recebe comandos do processador através do Barramento de Controle (***Control Bus***)
- **Dados**
 - Transferidos através do Barramento de Dados (***Data Bus***)
- **Status**
 - Status bits – BUSY, READY, condições de erro, ...
- **Descodificação de endereço**
 - Cada I/O Module tem de reconhecer os endereços dos periféricos que controla

ABF - ASCII_I/O_1

3

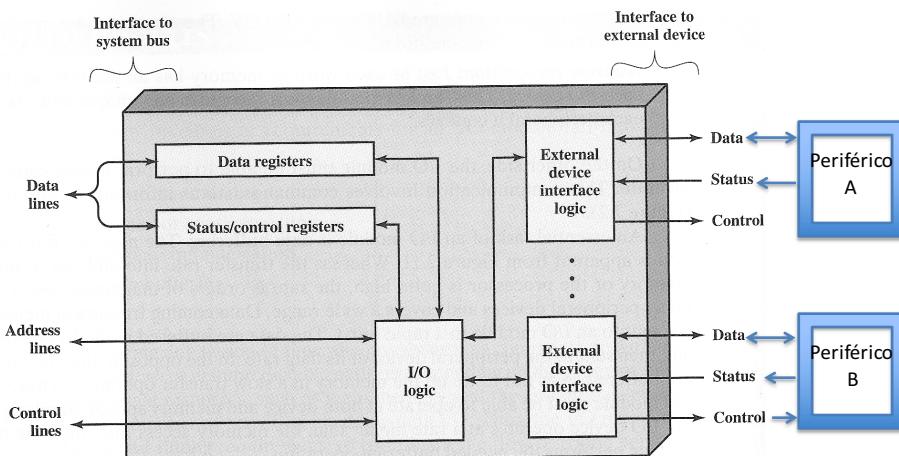
Funções dos módulos de I/O

- Um módulo de I/O funciona para permitir ao processador ver um modelo simplificado de um conjunto de periféricos (i.e. abstrair dos detalhes de funcionamento de cada periférico específico)
- O módulo de I/O pode esconder do processador detalhes de temporização, formatos e do controle eletromecânico do periférico, permitindo ao processador comunicar com o periférico por simples comandos de escrita e de leitura, e eventualmente comandos de *open* e *close* file.
- Os módulos de I/O mais simples podem deixar ainda para o processador grande parte do trabalho de controlo do periférico (p.ex. rebobinar uma unidade de fita magnética)

ABF - ASCII_I/O_1

4

Estrutura dos Módulos de I/O



ABF - AC II_I/O_1

5

Dispositivos Externos (Periféricos)

- **Periférico**
 - Um dispositivo externo ligado a um módulo de entrada/saída
- Fornecem o meio de transferir informação entre o ambiente exterior e o computador
- Ligam-se ao computador através de um módulo de entrada/saída
 - A ligação é usada para trocar informação de controle, status e dados entre o módulo de I/O e o dispositivo
- 3 categorias:
- Legíveis pelas pessoas
 - Adequados para comunicar com o utilizador do computador
 - Terminais Video (VDTs), teclado, impressoras
- Legíveis pela máquina
 - Adequados para comunicar com equipamento
 - Unidades de Disco e Banda magnéticos, pens, sensores e atuadores
- Comunicação
 - Adequados para comunicar com dispositivos remotos (um terminal, outro computador, ...)

ABF - AC II_I/O

6

Periféricos (Dispositivos Externos)

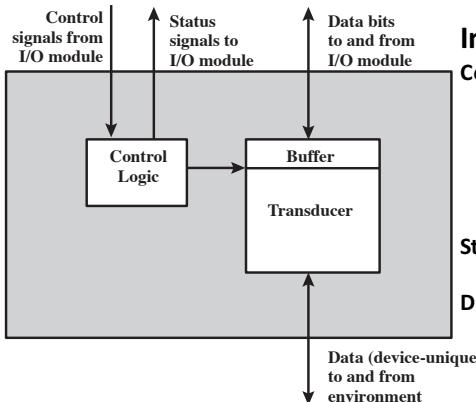


Figure 7.2 Block Diagram of an External Device

Interface do Módulo de I/O :

Control signals – determinam a função a executar pelo periférico

I/P (Read) – enviar dado para I/O Module

O/P (Write) – receber dado do I/O Module

Status signals – indicam o estado do periférico

Data – conjunto de bits a ser recebidos ou enviados pelo Módulo de I/O

Periférico:

Control Logic – controla a operação do periférico segundo as diretrivas do I/O Module

Buffer – armazena temporariamente os dados transferidos entre o I/O Module e o exterior
Transducer – conversão dos sinais elétricos em outras formas

ABF - AC II_I/O

7

Módulo de I/O – modelo de programação

- **Data Register** – registo onde o processador coloca o dado a ser escrito no periférico (*output*) e onde o periférico coloca o dado a ser lido pelo processador (*input*)
- **Status Register (*read-only*)** – bits que indicam o estado do periférico
- **Control Register (*write-only*)** – registo onde o processador escreve os comandos/modo de operação para o periférico

ABF - ACII_I/O_1

8

Comandos de Entrada/Saída

- 4 tipos de comandos que um módulo de I/O pode receber quando é endereçado pelo processador:
 1. **Control** – usado para ativar o periférico e indicar-lhe o que fazer
 2. **Teste** – usado para testar as várias condições de estado associadas ao módulo de I/O e aos periféricos a ele ligados
 3. **Input** – faz com que o módulo de I/O obtenha um dado do periférico e o coloque no seu buffer interno
 4. **Output** – faz com que o módulo de I/O leia um dado do *data bus* e o transmita ao periférico

Processador ↔ Módulo de I/O (Exemplo: Input)

1. O processador interroga o I/O module para verificar o status do periférico.
 2. O I/O module indica o status do periférico.
 3. Se o periférico está operacional e pronto a transmitir, o processador pede a transferência dos dados ao I/O module através de um comando.
 4. O I/O module obtém um dado (e.g., 8 or 16 bits) do periférico
 5. O dado é transferido do I/O module para o processador.
- Cada uma destas transações envolve uma ou mais operações de arbitragem do bus que liga o processador ao módulo de I/O

Como identificar os registos de I/O?

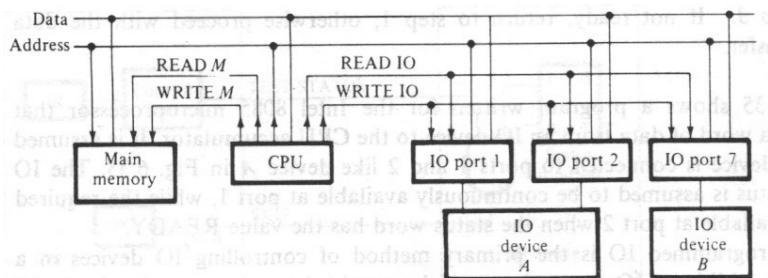
Alternativas:

- Agrupados num espaço de endereçamento próprio, distinto do espaço de endereçamento de memória – **I/O isolado**
 - Reportório de instruções inclui instruções específicas para I/O (Intel x86: instruções **in** e **out**; espaço de endereçamento de 64kB (endereços de I/O de 16-bits))
- Registos de I/O mapeados em memória (**Memory-mapped I/O**)

Organização do Sistema de Entradas/Saídas

- **Memory mapped I/O**
 - Os Periféricos e a Memória partilham o mesmo *address space*
 - I/O aparece como leitura/escrita na memória (*loads* e *stores*)
 - Ausência de comandos específicos para I/O
 - **MIPS**: instruções de **load** (input) e **store** (output) para comunicar com os periféricos
- **I/O Isolado**
 - Espaço de endereçamento separado, específico dos dispositivos de I/O
 - Necessárias linhas de seleção distintas para a memória e os dispositivos de I/O
 - Comandos especiais para I/O (instruções **In** e **Out**)

I/O isolado

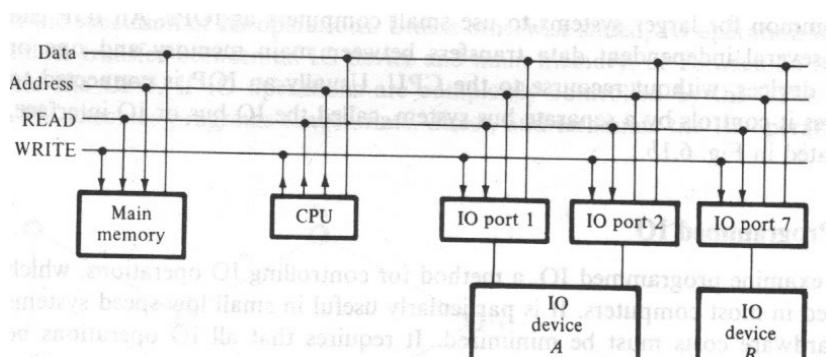


- Comandos separados de READ e WRITE para memória (READ M e WRITE M) e para as portas de Entrada/Saída (READ IO e WRITE IO)
- Espaço de endereçamento de IO muito mais reduzido que espaço de endereçamento da memória

ABF - ACII_I/O_1

13

I/O mapeado na memória

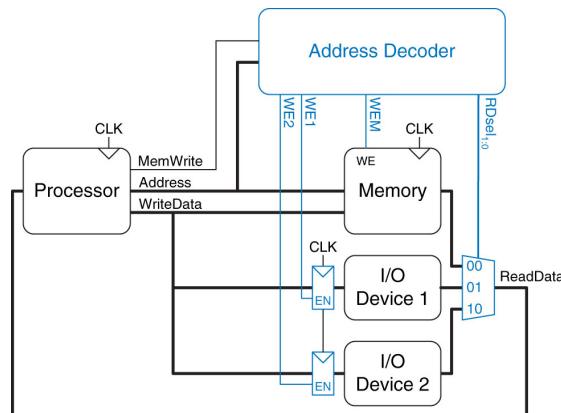


- Espaço de endereçamento único para memória e dispositivos de entrada e saída de dados

ABF - ACII_I/O_1

14

Descodificação dos Endereços para Memory-mapped I/O



ABF - ACII_I/O_1

15

PIC32 Memory Map

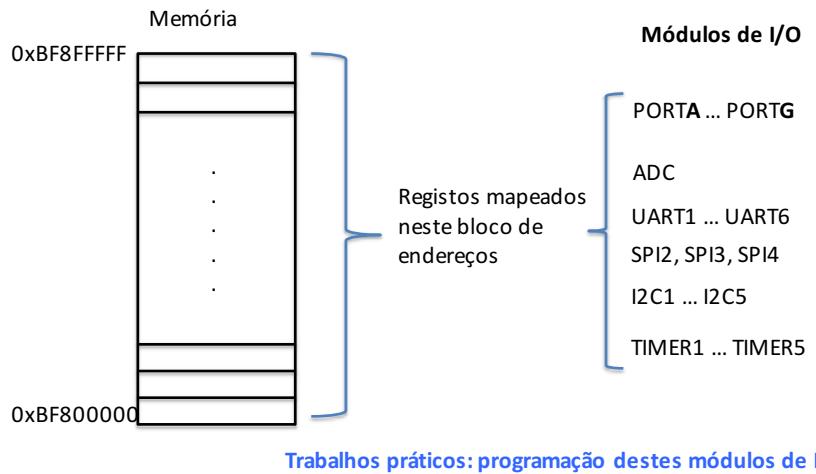
**Endereços dos
registos dos módulos de I/O
SFRs – Special Function Registers
0xBF800000 a 0xBFxFFFF**

Virtual Memory Map	
0xFFFFFFF	Reserved
0xBFC03000	Device Configuration Registers
0xBFC02FFF	
0xBFC02FF0	
0xBFC02FEF	
0xBFC00000	Boot Flash
0xBF900000	Reserved
0xBF8FFFFF	SFRs
0xBF800000	Reserved
0xBD080000	Reserved
0xBD07FFFF	Program Flash ⁽²⁾
0xBD000000	Reserved
0xA0020000	
0xA001FFFF	RAM ⁽²⁾
0xA0000000	Reserved
0x9FC03000	
0x9FC02FFF	Device Configuration Registers
0x9FC02FF0	
0x9FC02FEF	
0x9FC00000	Boot Flash
0x9D080000	Reserved
0x9D07FFFF	Program Flash ⁽²⁾
0x9D000000	Reserved
0x80020000	
0x8001FFFF	RAM ⁽²⁾
0x80000000	Reserved
0x00000000	

ABF - ACII_I/O_1

16

PIC32 Memory-mapped I/O Registers (SFRs – Special Function Registers)



ABF - ASCII_I/O_1

17

Comandos de Entrada/Saída

- **Control** (*swcontrol register*)
 - Usados para ativar um dispositivo e indicar-lhe o que fazer
- **Test** (*lw status register*)
 - Usados para testar várias condições do estado (*status*) do dispositivo
- **Read (Input)** (*lw data register*)
 - O módulo de I/O obtém um dado do periférico e coloca-o num buffer interno. O processador obtém o dado indicando ao módulo de I/O que o coloque no Data Bus
- **Write (Output)** (*sw data register*)
 - O módulo de I/O recebe o dado (byte ou word) do data bus e transmite-o posteriormente ao periférico

ABF - ASCII_I/O_1

18

Programação de I/O Especificidades

- Manipulação do conteúdo dos registos de controle e status é feita ao nível do bit – diferentes bits desses registos controlam diferentes modos operação do módulo de I/O
- Utilização intensiva do uso de máscaras para configurar, definir a operação a executar ou testar o estado do módulo.

ABF - ASCII_I/O_1

19

Operações Lógicas

- Instruções para manipulação de bits

Operação	C	Java	MIPS
Shift left	<<	<<	sll
Shift right	>>	>>>	srl
AND bit-a-bit	&	&	and, andi
OR bit-a-bit			or, ori
NOT bit-a-bit	~	~	nor

- Uteis para extrair e inserir grupos de bits numa *word*

ABF - AC I - MIPS IS_2

20

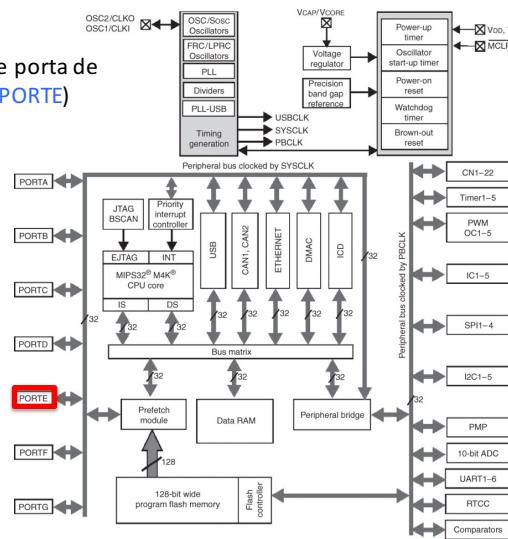
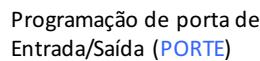
Selecionar bits e escrever bits

- Selecionar bits numa palavra
 - ***andi*** – valor do imediato com 1s nos bits que se pretende selecionar e com os restantes bits 0
- Impor valores para alguns bits de uma palavra
 - **Colocar bits a 0** -> ***andi*** com 0s nessas posições (e 1s nas que posições que não se pretendem alterar)
 - **Colocar bits a 1** -> ***ori*** com 1s nessas posições (e 0s nas que posições que não se pretendem alterar)

Operações Lógicas

Source Registers								
\$s1	1111	1111	1111	1111	0000	0000	0000	0000
\$s2	0100	0110	1010	0001	1111	0000	1011	0111
Assembly Code								Result
and \$s3, \$s1, \$s2	\$s3	0100	0110	1010	0001	0000	0000	0000
or \$s4, \$s1, \$s2	\$s4	1111	1111	1111	1111	1111	0000	1011
xor \$s5, \$s1, \$s2	\$s5	1011	1001	0101	1110	1111	0000	1011
nor \$s6, \$s1, \$s2	\$s6	0000	0000	0000	0000	0000	1111	0100

Diagrama de Blocos do PIC32



ABF - AC I Intra

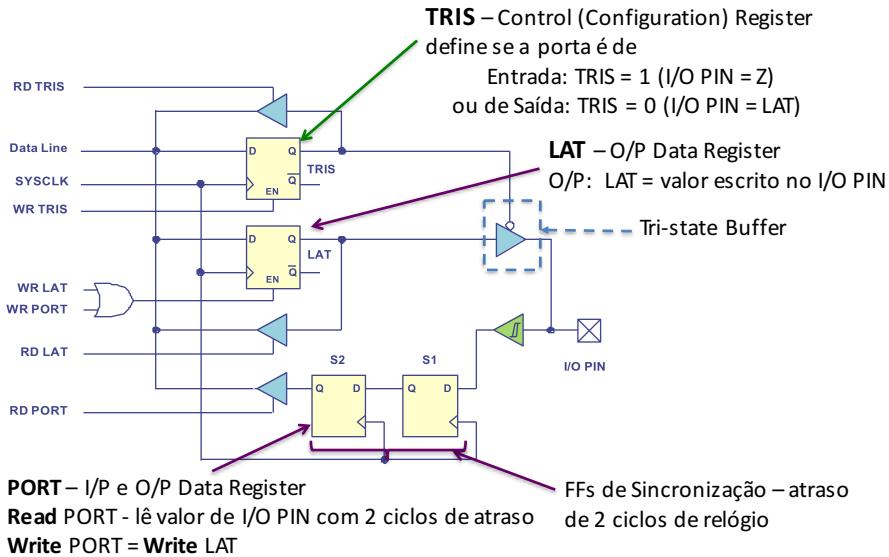
23

PIC32 I/O Ports

AC2 PIC32 I/O - ABF

24

Portas do PIC32 – Estrutura Simplificada



25

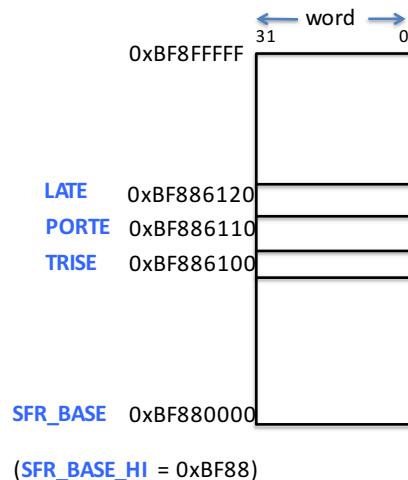
Programação de uma porta

- Designação dos registos do Port E:
 - TRISE, LATE, PORTE
- Comunicação com as portas:
 - Registros mapeados no espaço de endereçamento de memória (*memory-mapped I/O*):
 - Leitura de registo – **load** PORTE
 - Escrita em registo – **store** LATE

AC2 PIC32 I/O - ABF

26

Mapeamento em memória do PORTE



(**SFR_BASE_HI** = 0xBF88)

AC2 PIC32 I/O - ABF

27

Programação de uma porta

1. Configurar porta – escrever no registo TRISx

Ex: configurar os bits 0 e 2 de TRISE como porta de saída
(sem alterar a configuração dos outros bits)

Endereço relativo de TRISE = **0x6100**

```

lui $t1, SFR_BASE_HI
lw $t2, TRISE($t1) # ler configuração de PORTE
andi$t2, $t2, 0xFFFF # bits 0 e 2 de $t2 = 0
sw $t2, TRISE($t1) # pins 0 e 2 de PORTE configurados como
                     # de saída; restantes bits inalterados

```

2. Comunicação de dados com a porta:

ler portas de entrada
atribuir valores às portas de saída

AC2 PIC32 I/O - ABF

28

Atribuir valores às portas de Saída

Para colocar as saída RE0 e RE3 a 1 pode fazer-se:

```
lui      $t1, SFR_BASE_HI      #
lw       $t2, LATE($t1)        # Read LATE register
ori      $t2, $t2, 9            # Set bit0 and bit3
sw       $t2, LATE($t1)        # Write LATE register
```

LATE	31 X X X	3 2 1 0 1 X X 1
------	-------------------	----------------	--------------------