

TÉCNICAS DE I/O

ABF - AC II_I/O_2016

12

Técnicas de Entrada/Saída

| | No Interrupts | Use of Interrupts |
|--|----------------|----------------------------|
| I/O-to-memory transfer through processor | Programmed I/O | Interrupt-driven I/O |
| Direct I/O-to-memory transfer | | Direct memory access (DMA) |

ABF - AC II_I/O_2016

13

Técnicas de Entrada/Saída (I/O)

- **I/O Programado**

– A entrada e saída de dados é feita sob controlo do programa, que interroga o módulo de I/O (**polling**)

- **Sob Interrupção (Interrupt-driven)**

– O periférico, quando necessita de atenção, gera um pedido de **interrupção**. O processador interrompe o programa em execução para executar uma sub-rotina que faz a transferência de dados de/para o periférico

- **Acesso direto à memória (DMA)**

– Os dados são transferidos diretamente entre o periférico e a memória sem intervenção do processador

ABF - AC II_I/O_2016

14

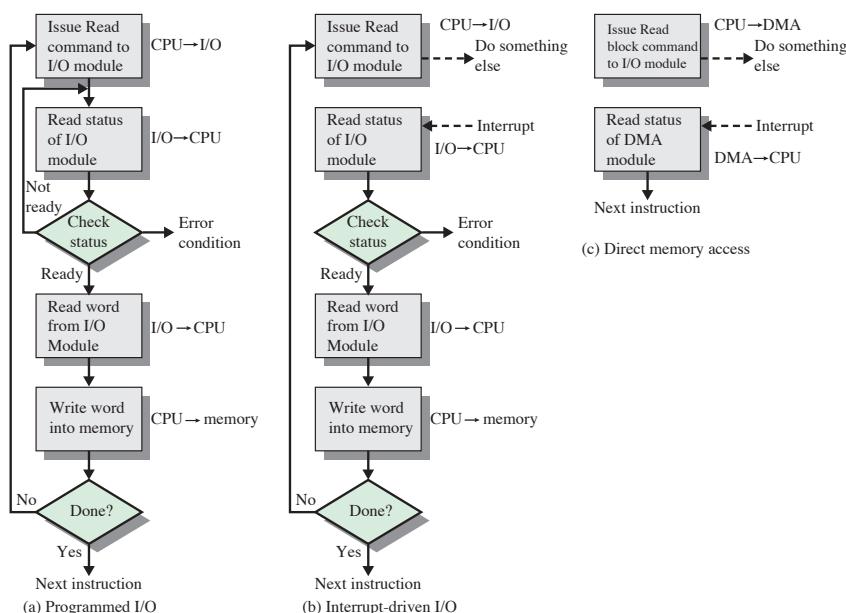


Figure 7.4 Three Techniques for Input of a Block of Data

ABF - AC II_I/O_2016

15

I/O Programado

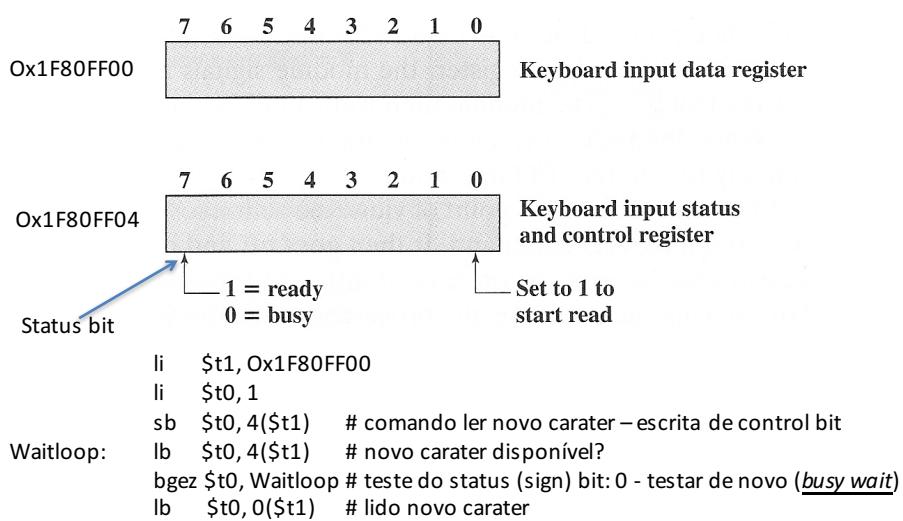
- Quando o processador está a executar um programa e encontra uma instrução relacionada com um módulo de I/O:
 - Processador envia um comando ao módulo de I/O
 - Módulo de I/O executa a ação especificada no comando e coloca nos seus bits de status os valores apropriados (sem tomar qualquer ação para alertar o processador)
 - O processador lê periodicamente o estado do módulo de I/O até verificar que a operação terminou
- Se o processador é mais rápido do que o periférico, “**busy waiting**” – processador à espera do periférico sem fazer processamento util

ABF - AC II_I/O_2016

16

I/O Programado

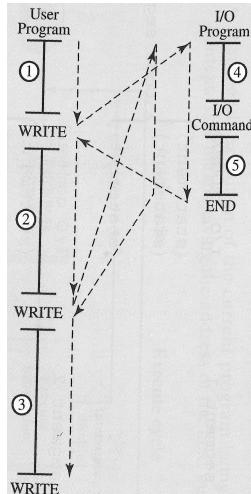
Exemplo: Leitura do teclado



ABF - AC II_I/O_2016

17

I/O programado – invocação pelo programa da rotina de comunicação com o periférico



Secções 1, 2 e 3 do programa não envolvem I/O
 4 – preparação da operação de I/O:
 verificar status do dispositivo
 transferir o dado para o módulo de I/O
 preparar os parâmetros para o comando de I/O
 Emitir o comando de I/O e
 aguardar que o dispositivo o execute (testar ***status***)
 5 – completar a operação

ABF - AC II_I/O_2016

18

I/O SOB INTERRUPÇÃO

ABF - AC II_I/O_2016

19

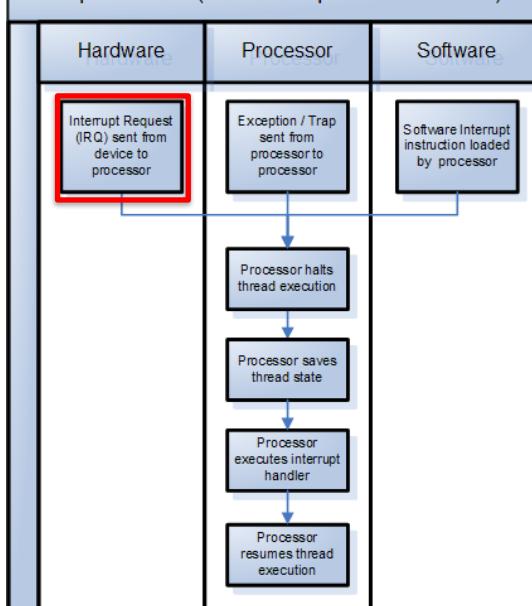
Interrupções e Exceções

- **Interrupção** – resultado de uma operação normal (pedido de um módulo de I/O) – única condição de exceção que ocorre independentemente da execução da sequência de instruções pelo CPU
 - **Exceção** – resultado de uma condição de erro
 - Erro de endereço de memória
 - Instrução ilegal (não existente ou privilegiada)
 - Erro de paridade na leitura da memória
 - *System Calls e Traps* (e.g. instruções ***break*** e ***syscall*** causam exceções)
- Resposta semelhante em ambos os casos:
- A sequência das instruções do programa em execução é interrompida para executar a rotina que trata da interrupção ou exceção em causa (*interrupt/exception handler*).

ABF - AC II_I/O_2016

20

Interrupt Process (from three potential sources)

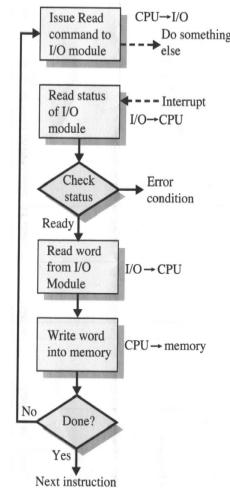


ABF - AC II_I/O_2016

21

Entradas/Saídas sob Interrupção

- Problema com i/o sob controlo do programa: processador tem de esperar muito tempo para que o módulo de i/o esteja pronto a receber ou transmitir um dado
- Alternativa:
 1. processador envia comando de i/o ao módulo e continua a fazer trabalho útil
 2. O módulo de i/o **interrompe** o processador quando está pronto para trocar dados com o processador
 3. O processador executa a transferência de dados e retoma o que estava a fazer



ABF - AC II_I/O_2016

22

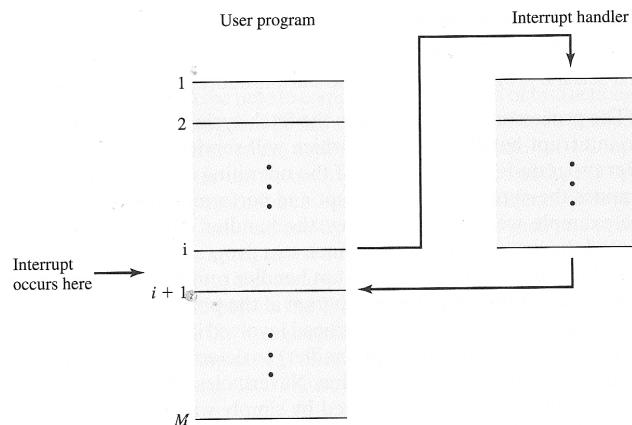
I/O sob interrupção (ex.: leitura)

1. Processador envia ao periférico um comando de leitura
2. Processador prossegue execução
Quando termina execução do comando periférico gera interrupção
3. Processador termina a execução da instrução atual e executa rotina de tratamento de interrupções (*Interrupt Service Routine* ou *Interrupt Handler*):
 1. Identifica a origem da interrupção
 2. Verifica *status* do periférico
 3. Lê o novo dado
 4. Armazena-o na memória
3. Processador retoma execução do programa

ABF - AC II_I/O_2016

23

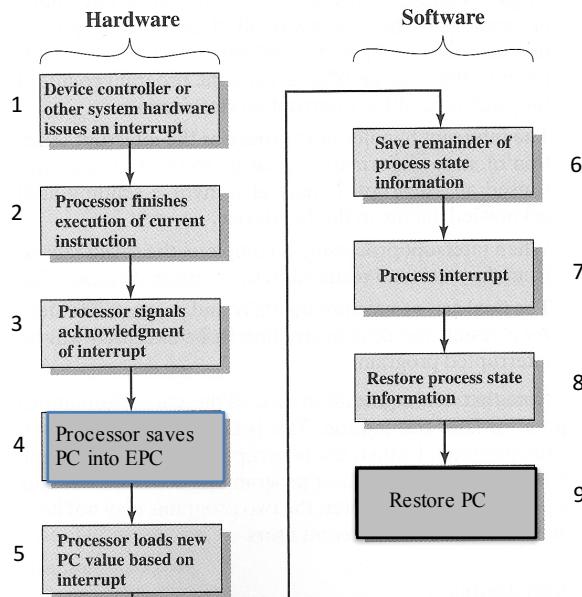
Interrupções – transferência de controle



ABF - AC II_I/O_2016

24

Processamento de uma Interrupção



ABF - AC II_I/O_2016

25

Questões:

1. Como determina o processador qual o dispositivo que formulou o pedido de interrupção?
2. Quando existirem vários pedidos de interrupção em simultâneo como decide o processador qual atender primeiro?

ABF - AC II_I/O_2016

26

Identificação do dispositivo - alternativas

1. Cada dispositivo com uma linha de pedido de interrupção própria – **inviável dedicar mais do que um nº restrito de linhas do bus ou pins do processador a interrupções**
 - Mesmo quando existe mais do que uma linha de pedido de interrupção, cada uma tem vários módulos de I/O a ela ligados
2. Identificação por s/w (**S/W poll**) – a ISR testa cada módulo de I/O para identificar qual pediu a interrupção. Uma vez identificado o processador salta para a rotina específica de tratamento das interrupções desse módulo
3. **Daisy Chain** (H/W poll, vetorizada)

ABF - AC II_I/O_2016

27

Prioridades no atendimento de pedidos de interrupção

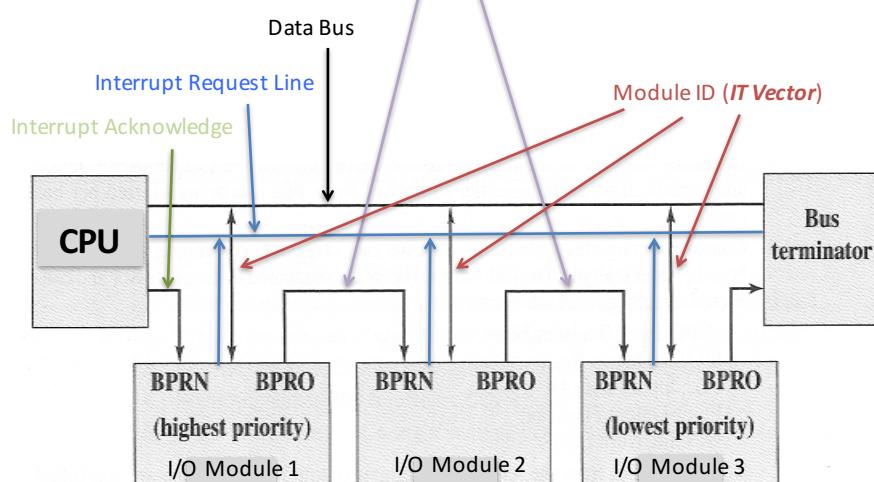
Múltiplas fontes de interrupção - duas soluções possíveis:

1. Impedir as interrupções (**interrupt disable**) enquanto está a ser processada uma interrupção
➤ Não tem em conta a necessidade de garantir tempos de resposta
2. Definir prioridades para as interrupções, permitindo que a rotina de tratamento de uma interrupção seja interrompida para atender pedido de interrupção de um dispositivo com maior prioridade.

ABF - AC II_I/O_2016

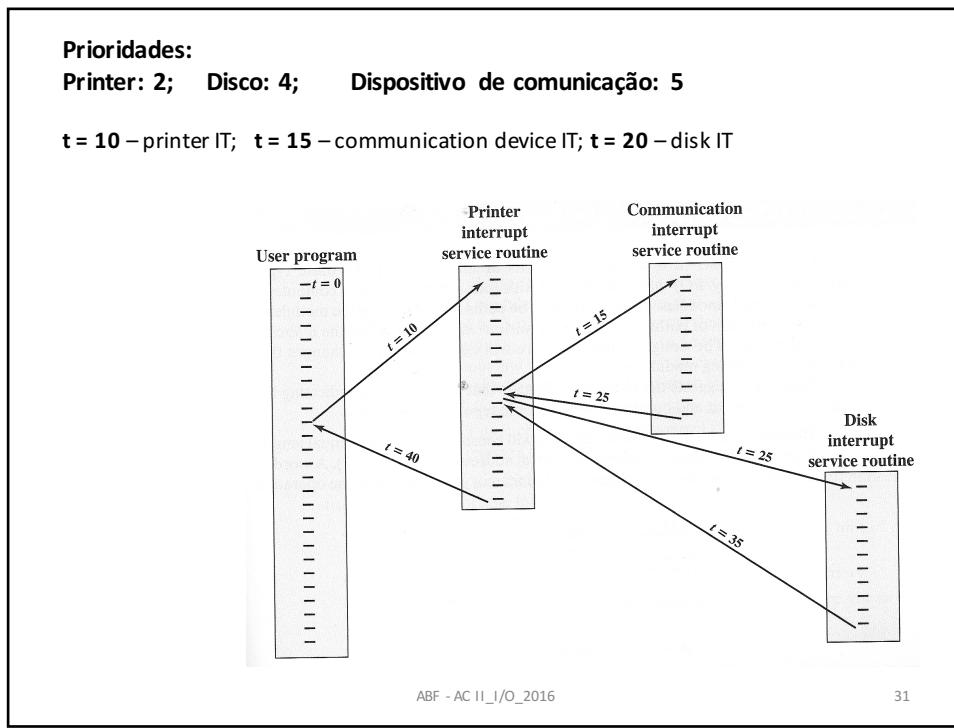
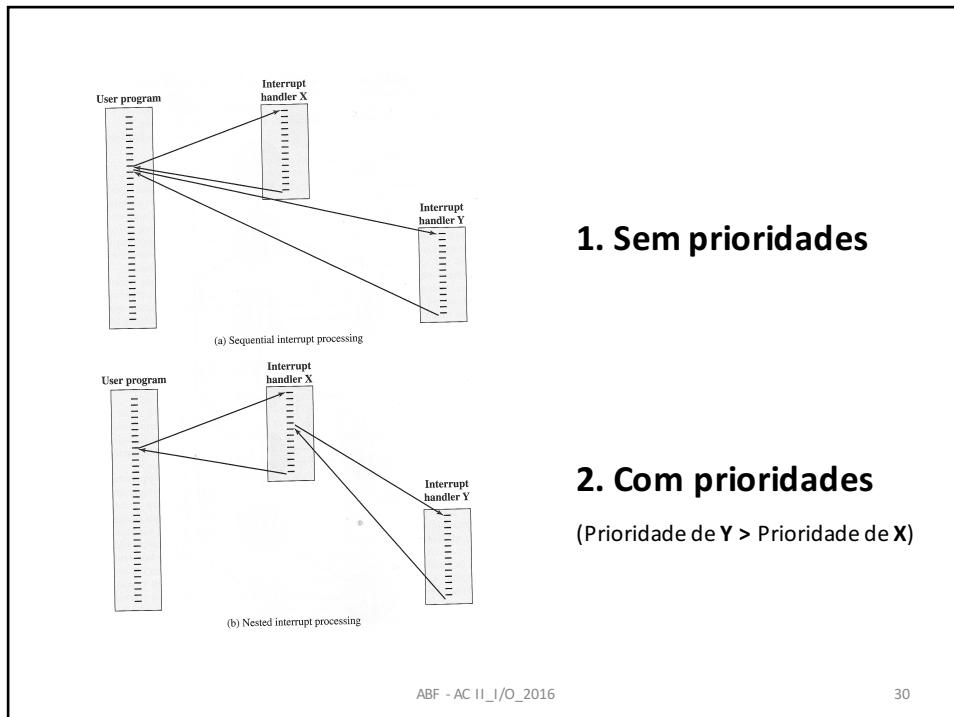
28

Multiplas fontes de interrupção: Daisy Chain



ABF - AC II_I/O_2016

29



Controlador de Interrupções

- Programação

- Enable/Disable dos pedidos de interrupção de cada periférico
- Definir os níveis de prioridade de atendimento dos pedidos de interrupção dos vários periféricos

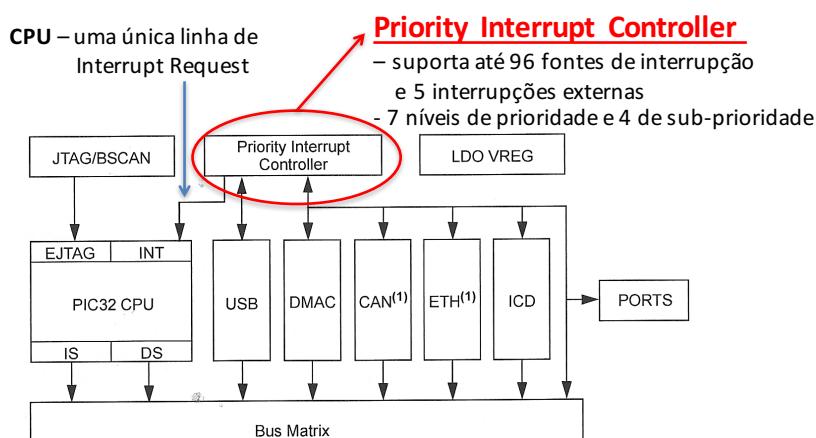
- Operação

1. Periférico x pede interrupção
2. Se (ITx enabled) Controlador de IT transmite pedido de interrupção ao CPU
3. CPU termina instrução em curso e
se prioridade de ITReq > nível de prioridade atual **IT acknowledge**
4. Salvaguarda do PC e transferência de controle para **IT Handler**
Non-vectored IT – IT Handler identifica periférico que pediu IT
Vectored IT – Controlador de IT fornece **IT vector** – controle transferido diretamente para o IT Handler do periférico que pediu IT

ABF - AC II_I/O_2016

32

Multiplas fontes de interrupção: PIC32



ABF - AC II_I/O_2016

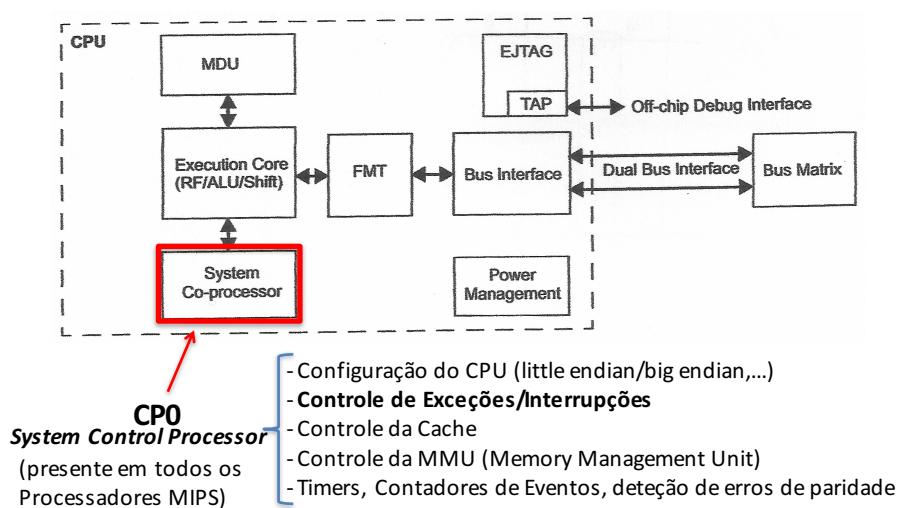
33

INTERRUPÇÕES NO MIPS

ABF - AC II_I/O_2016

34

Exceções: o CoProcessador 0 (CP0)



ABF - AC II_I/O_2016

35

Exceções

- Causadas por:
 - Hardware: **Interrupções** – causadas por um periférico (p.ex. o teclado)
 - Software, tambem designadas por **Traps** (tentativa de execução de instrução indefinida, *breakpoint, system call*)
- Quando ocorre uma exceção:
 - Causa da exceção é registada no **Cause Register**
 - Salto para a rotina de tratamento de exceções (**Exception handler** – endereço 0x80000180)
 - Retorno ao programa em execução quando ocorreu a exceção

ABF - AC II_I/O_2016

36

Registros das Exceções

- Registros de **CP0** – não são parte do banco de registo do processador
 - **SR (Status Register)** – control/status register
 - **Cause** – regista a causa da exceção
 - **EPC (Exception PC)**
- Para ler e escrever registo de CP0, duas instruções: **mfc0** e **mtc0**
 - Exemplo:

```

mfc0    $t0, SR
andi    $t0, <complemento dos bits que se pretendem colocar a 0>
ori     $t0, <bits que se pretendem colocar a 1>
mtc0    $t0, SR
  
```

ABF - AC II_I/O_2016

37

MIPS CP0 – instruções (só executáveis em modo kernel)

- **mtc0 s, <n>** # move to cp0 – coloca o conteúdo do registo **s** do processador no registo **n** de CP0
- **mfc0 d, <n>** # move from cp0 – transfere o conteúdo do registo **n** de CP0 para o registo **d** do processador – única maneira de inspecionar o conteúdo de um registo de CP0
- **eret** # return from exception (**rfe** nas primeiras versões do MIPS) – **PC = EPC**

ABF - AC II_I/O_2016

38

Processamento das Exceções

1. O PC (endereço da instrução a executar quando o *handler* termina) é salvaguardado no registo **EPC** (*Exception Program Counter*) de C0
 2. No registo **Cause** (C0) é escrita a causa da interrupção/exceção
 3. Processador passa a **modo kernel** e **disables ITs**
 4. Processador salta para Exception Handler
- **Exception Handler:**
 1. Salvaguarda os registo no stack (prólogo)
 2. Lê o Cause Register ***mfc0 \$k0, Cause***
 3. Processa a exceção
 4. Restaura os registo (epílogo)
 5. Retorna ao programa
 - **\$k0, \$k1** – registo do CPU cujo uso está reservado para o núcleo do sistema de operação (registo 26 e 27)

ABF - AC II_I/O_2016

39

CP0 - Registros relacionados com Exceções/Interrupções

| Reg. No. | Nome | Função |
|-------------------|-------|--|
| Core Timer | 8 | BadVAddr |
| | 9 | Count |
| | 11 | Compare |
| | 12 | Status (SR) |
| | | IntCtl |
| | | SRSCtl |
| | | SRSMMap |
| | 13 | Cause |
| 14 | EPC | Conteúdo do PC na exceção mais recente |
| PIC32 ————— 15 | Ebase | Endereço base dos vetores de interrupção |

ABF - AC II_I/O_2016

40

Status (!) Register (SR) bits

- **IE** – Interrupt Enable (bit 0)
 - permite (= 1)/inibe (= 0) globalmente as interrupções
- **EXL** (Exception Level – bit 1) e **ERL** (ERror Level – bit 2) – estes bits inibem as interrupções quando estão **set**. Quando ocorre uma interrupção (exceção) um deles é automaticamente **set** – novas interrupções ficam inibidas
- **IPL<2:0>** - Interrupt Priority Level (bits 12-10)
 - O processador só é interrompido se a prioridade do pedido de interrupção for superior ao valor de IPL
- **NMI** (bit 19) – Non-Maskable Interrupt
 - ❖ **Interrupções enabled sse IE = 1, EXL = 0, ERL = 0**

ABF - AC II_I/O_2016

41

Cause Register

- **IP7-0 – Interrupt Pending** – pedidos de interrupção pendentes. **Cause(IP7-2)** (bits 15-10) indica os pedidos externos de interrupção. O **AND** com a **Interrupt Mask** (SR) define os pedidos de interrupção ativos
- **TI – Timer Interrupt** (bit 30) – interrupção causada pelo *timer* do CPU (*core timer*)

ABF - AC II_I/O_2016

42

PIC - Exceções e o Cause Register

Table 2-10: Cause Register EXCCODE<4:0> Bits

| Exception Code Value | | Mnemonic | Description |
|----------------------|-----------|----------|---|
| Decimal | Hex | | |
| 0 | 0x00 | Int | Interrupt |
| 1-3 | 0x01 | — | Reserved |
| 4 | 0x04 | AdEL | Address error exception (load or instruction fetch) |
| 5 | 0x05 | AdES | Address error exception (store) |
| 6 | 0x06 | IBE | Bus error exception (instruction fetch) |
| 7 | 0x07 | DBE | Bus error exception (data reference: load or store) |
| 8 | 0x08 | Sys | Syscall exception |
| 9 | 0x09 | Bp | Breakpoint exception |
| 10 | 0x0A | RI | Reserved instruction exception |
| 11 | 0x0B | CPU | Coprocessor Unusable exception |
| 12 | 0x0C | Ov | Arithmetic Overflow exception |
| 13 | 0x0D | Tr | Trap exception |
| 14-31 | 0x0E-0x1F | — | Reserved |

ABF - AC II_I/O_2016

43

Count e Compare (PIC32 Core Timer)

- **Count** – incrementado a cada 2 ciclos de relógio.
- **Compare** – em conjunto com Count permite implementar um *timer*: **core timer**. Mantem um valor estável que só muda quando é escrito. Quando *Count* = *Compare* gera uma interrupção

ABF - AC II_I/O_2016

44

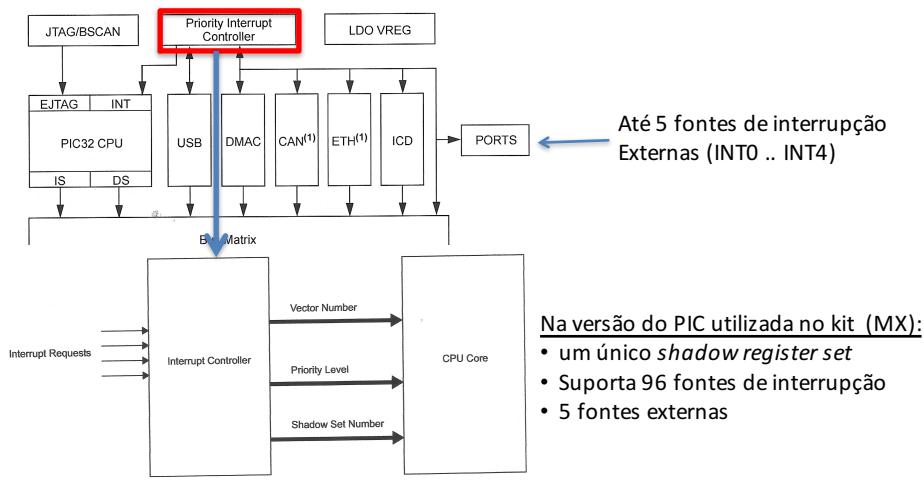
Interrupts PIC32 I/O

ABF - AC II_I/O_2016

45

PIC32 Priority Interrupt Controller

Responsável pelo pré-processamento dos pedidos de IT (IRQ) dos periféricos e por os apresentar ao CPU por ordem de prioridade



ABF - AC II_I/O_2016

46

PIC32 - EBase Register

- **ExceptionBase** (bits29-12) – endereço base dos vetores de interrupção

Prioridade das Interrupções no MIPS

- A arquitetura MIPS deixa para um controlador de interrupções externo (**EIC**) a definição de prioridades para as diferentes fontes de interrupção
- **PIC32** – prioridades definidas pelo **Priority Interrupt Controller**

ABF - AC II_I/O_2016

47

Sistema de Interrupções do PIC32MX

- 96 fontes de interrupção
- Suporta 5 fontes de interrupção externas
- 7 níveis de prioridade
- Possível definir 4 níveis de sub-prioridade dentro de cada nível de prioridade
- Segundo exemplar (cópia) dos registos gerais \$r0,...,\$r31 (***Shadow Register Set***) – quando ocorre uma interrupção com a mais alta prioridade (nível 7) o processador passa automaticamente a usar o *Shadow Register Set*
 - dispensa salvaguarda do conteúdo dos registos gerais no stack (que constitui o prólogo das ISR das restantes interrupções).

ABF - AC II_I/O_2016

48

Bits de Controle das Interrupções

- Cada fonte de interrupção tem associados 7 bits de controle, agrupados em vários SFRs:
 - *Interrupt Enable (_IE)* – registos **IECx** (Interrupt Enable Control Register): cada bit do registo enables/disables uma das interrupções de um dispositivo específico
 - = 0 - o evento associado é impedido de causar interrupções
 - = 1 - o pedido de interrupção é processado
 - *Interrupt Flag (_IF)* – registos **IFSx** (Interrupt Flag Status Register): cada bit regista os pedidos de interrupção de um dispositivo específico
 - Ativada (=1) de cada vez que há um pedido de interrupção
 - *Interrupt Priority* – registos **IPCx** (Interrupt Priority Control Register) – 2 níveis de prioridade:
 - Priority Level (_IP) – 3 bits que definem o nível de prioridade da fonte de interrupções
 - Se $IP_x < CO_StatusReg_IPL$ os pedidos de interrupção são ignorados
 - Subpriority Level (_IS) – 2 bits (**ISx**) que definem 4 níveis secundários de prioridade num mesmo grupo de prioridade (mesmo _IP)

ABF - AC II_I/O_2016

49

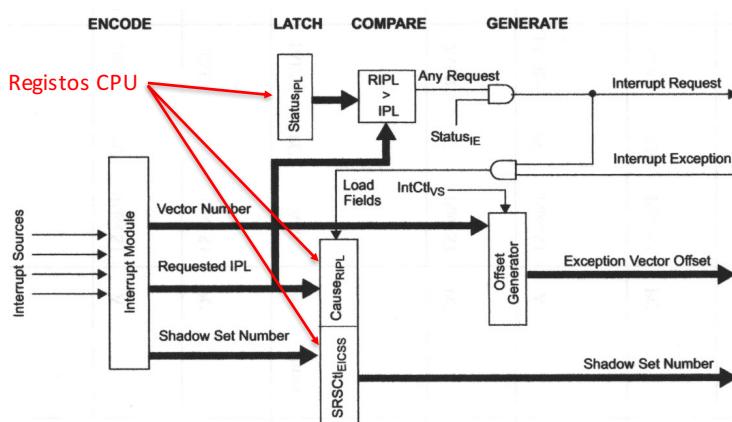
Funcionamento do Controlador de ITs

- Todos os pedidos de interrupção (**IRQs**) são testados na transição positiva de SYSCLK e registados nos registos **IFSx**
- O pedido de interrupção é ignorado se o correspondente bit em **IECx** = 0 (o bit atua como máscara da *flag* de interrupção)
- Se as interrupções estiverem *enabled* os **IRQs** são codificados num no. de vetor. A cada um destes está associado um nível de prioridade determinado pelo valor do campo **IPx** do respetivo registo **IPC**
 - 2 modos de funcionamento: **Single Vector** e **Multi-Vector**
- O controlador de Interrupções seleciona o IRQ de maior prioridade entre os IRQ pendentes e apresenta ao CPU o no. do vetor associado e o nível de prioridade do pedido

ABF - AC II_I/O_2016

50

PIC -> CPU



ABF - AC II_I/O_2016

51

PIC IT Controller – Registros (Registros mapeados na memória)

- **INTCON** – INTerrupt CONtrol Register
- **INTSTAT** - INTerrupt STATus Register
- **IFSx** – Interrupt Flag Status Registers
- **IECx** – Interrupt Enable Control Registers
- **IPCx** – Interrupt Priority Control Registers

ABF - AC II_I/O_2016

52

Interrupt Flag Status Regs.: **IFS0, IFS1, IFS2**

- 3 registos de 32 bits ($3 * 32 = 96$) com as *interrupt flags* dos periféricos: **1** – ocorreu IT; **0** – não ocorreu

Interrupt Enable Control Regs.: **IEC0, IEC1, IEC2**

- 3 registos de 32 bits indicando quais as fontes de interrupção autorizadas: **1** – IT enabled; **0** – IT disabled

Interrupt Priority Control Regs.: **IPC3, ..., IPC7**

- **IPx<0:2>** - Interrupt Priority bits (Priority Levels 0 ... 7)
- **ISx<1:0>** - Interrupt Subpriority bits (Subpriority Levels 0 ... 3)

ABF - AC II_I/O_2016

53

INTCON – INTerrupt CONtrol Register

- **SS0** – Single Vector SHadow Register Set (bit 16)
- **MVEC** – Multi-VECtor Configuration bit (bit 12)
 - = 0 – single vector (IT não vetorizadas);
 - = **1** – **IT vetorizadas (modo usado em AC II)**
- **TPC<2:0>** - Interrupt Proximity Timer Control bits
 - Definem o nível máximo de prioridade para arrancar com o *timer*
 - **000** - Disables Interrupt Proximity Timer

ABF - AC II_I/O_2016

54

INTSTAT - INTerrupt STATus Register

- **SRIPL <2:0>** - nível de prioridade do mais recente ITReq(em Single Vector mode) apresentado ao CPU
- **VEC<5:0>** - IT_Vector apresentado ao CPU

IPTMR - Interrupt Proximity TiMeR Register

- **IPTMR<31:0>** - valor inicial do *timer* quando é disparado por uma interrupção

ABF - AC II_I/O_2016

55

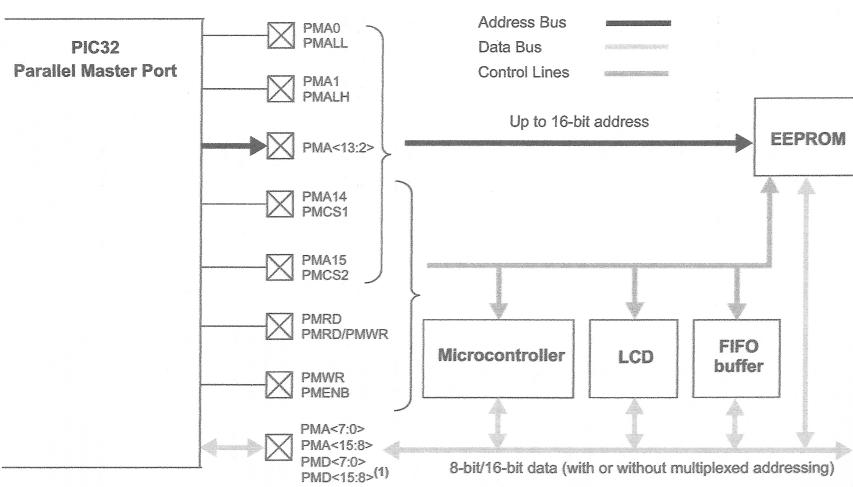
Parallel Master Port

- Exemplo de um módulo de I/O usado para I/O programado e I/O sob interrupção
- Concebido para comunicar com dispositivos paralelos muito diversos – dispositivos de comunicação, LCDs, memórias externas, microcontroladores
- Configurável para poder satisfazer os requisitos dos dispositivos muito diversos com que comunica

ABF - AC II_I/O_2016

56

PMP: ligações a dispositivos externos



ABF - AC II_I/O_2016

57

DMA

ABF - AC II_I/O_2016

58

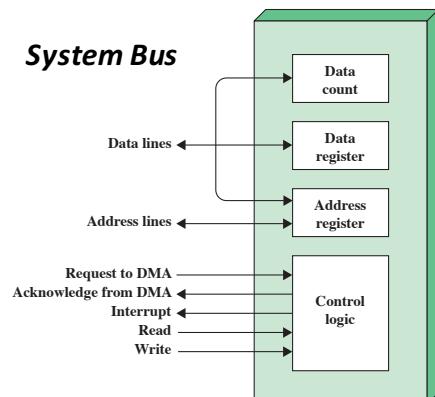
I/O sob Interrupção Limitações

- Taxas de transferência de informação de/para os periféricos limitada pela velocidade com que o processador pode atender e servir os dispositivos
- Consumo de tempo de Processador – por cada transferência o processador tem de executar um conjunto de instruções
 - Quando é necessário transferir grandes volumes de dados (e.g. HDD ou SSD) é mais eficiente usar outra técnica – Acesso Direto à Memória (**DMA**)

ABF - AC II_I/O_2016

59

Módulo DMA



- Capaz de assumir o controle do sistema em lugar do processador (necessário para transferir dados de/para a memória através do bus do sistema)
- Módulo DMA usa o bus quando o CPU não o está a utilizar, ou força o processador a suspender temporariamente a operação (*cycle stealing*)

Figure 7.11 Typical DMA Block Diagram

ABF - AC II_I/O_2016

60

Operação do Módulo DMA

1. Quando o processador quer ler ou escrever um bloco de dados, envia um comando ao módulo DMA indicando:
 - Tipo de operação (**read** ou **write**)
 - Endereço do periférico
 - Endereço base de memória para ler/escrever os dados – escrito no **Address Register**
 - Número de palavras a transferir – escrito no **Data Count Register**
2. Processador continua a executar; módulo DMA transfere o bloco, *word a word*, diretamente da memória sem intervenção do processador
3. Módulo DMA envia uma interrupção ao processador quando termina a transferência

➤ Processador envolvido apenas no início e no final da operação

ABF - AC II_I/O_2016

61

Configurações de DMA

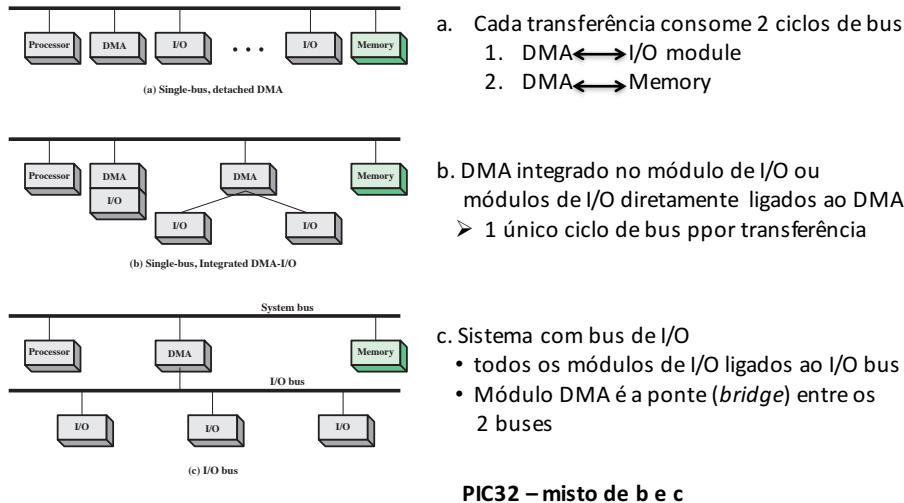


Figure 7.13 Alternative DMA Configurations ABF - AC II_I/O_2016

62

I/O - Bibliografia

- W. Stallings, "Computer Architecture and Organization", Ch. 7 (7.1–7.4)
- D.M. Harris, S.L.Harris, "Digital Design and Computer Architecture", Ch. 8 (8.5 e 8.6)
- D. Patterson, J. Hennessy, "Computer Organization and Design"
- PIC32 Family Reference Manual
 - Section 2 – CPU for devices with M4K Core
 - Section 12 – I/O Ports
 - Section 8 – Interrupts
- L. Di Jasio, "Programming 32-bit Microcontrollers in C", Day 5: Interrupts
- D. Sweetman, "See MIPS Run", Chs. 3, 5