

Interrupções e DMA

Interfaces e Periféricos 2001/2

António de Brito Ferrari

ferrari@det.ua.pt

Exceções e Interrupções

- Eventos que provocam a alteração do fluxo normal de execução das instruções do programa em execução
- Eventos causadores:
 - **Internos**: ocorrência de overflow, tentativa de executar instrução não definida, chamada ao sistema de operação (*system call*)
 - **Externos**: pedidos dos dispositivos de I/O
 - **Assíncronos** relativamente ao processador – sem qualquer relação temporal fixa com este

Excepções e Interrupções

- MIPS, Motorola e outros:
 - **Excepções** – designam os eventos internos
 - **Interrupções** - designam os eventos externos (pedidos dos dispositivos de I/O)
 - Excepções, em sentido lato, designam ambos
- Intel:
 - **Interrupções** é a designação geral (eventos tanto internos como externos)
 - Chamadas ao sistema: *software interrupts* ou **traps**

I/O Programado

1. Processador envia ao periférico um comando de leitura
 - 2. Ler *status* do periférico**
 - 3. *Status ready?* No: goto 2**
 4. Ler informação do periférico
 5. Escrever a informação lida na memória
 6. Fim? No: goto 1
- **Maior parte do tempo à espera (ciclo 2, 3)**

I/O sob interrupção (ex.: leitura)

1. Processador envia ao periférico um comando de leitura
2. Processador executa outras tarefas

Periférico, quando termina execução do comando, gera ***interrupção***. (desaparece ***busy waiting***)

Processador (ISR – *Interrupt Service Routine*):

3. Verifica *Status* do periférico
4. Lê informação do periférico
5. Armazena a informação lida na memória
6. Fim? Não: goto 1

Processamento de uma Interrupção

- Análogo à invocação de uma subrotina - controlo passa do programa que está a ser executado para um outro que vai atender o pedido de interrupção - a subrotina de tratamento da interrupção (***interrupt handler*** ou **ISR - Interrupt Service Routine**)
- Diferença: a execução da ISR é desencadeada por um evento externo, não por uma instrução de invocação de subrotina do programa em execução

Processamento de Interrupção (2)

1. Periférico gera interrupção (***Int. Request***)
2. Processador:
 - termina instrução que está a ser executada,
 - se as interrupções estiverem *enabled* reconhece o pedido (***Int. Acknowledge***)
 - sistema de interrupções é desactivado (***Int. disable***)
 - salvaguarda contexto (PC, ...)
 - PC := &ISR (endereço de entrada da subrotina de tratamento da interrupção)

Processamento de Interrupção (3)

➤ **Questão 1:** como obter o endereço de entrada da ISR?

- Duas alternativas:

- Endereço único, comum a todas as interrupções - a cargo do software (*ISR*) identificar o periférico que gerou a interrupção - MIPS
- Endereço determinado por *hardware* - periférico que interrompeu identifica-se através de um **vector de interrupção** que é lido pelo processador e utilizado por este para obter o endereço da ISR correspondente – **interrupções vectorizadas** (*vectored interrupts*)
 - Intel, Motorola 68000

Processamento de Interrupção (4)

- **Questão 2:** quando ocorrem em simultâneo vários pedidos de interrupção, qual a ordem de atendimento?
 - Necessário atribuir **prioridade** diferente a cada fonte de interrupções
- **Exigência:** rapidez de processamento
 - Sistemas Tempo Real: interrupções são eventos críticos que exigem garantias em termos de tempo de resposta máximo

Processamento de exceções no MIPS

- O coprocessador CP0 e os seus registos é que definem como são processadas as exceções e interrupções
- Quando ocorre uma exceção o conteúdo do Program Counter é guardado no **EPC** (Exception Program Counter) e é iniciada a execução de uma rotina geral de tratamento de exceções

&ISR (*General Exception Vector*) = 0x80000080

interrupções não-vectorizadas

Exceções no MIPS

- Registos de CP0:
 - **EPC** – endereço de retorno
 - **Cause Register** – permite identificar exceção. Campos:
 - **Excode** – identifica o tipo de exceção. Interrupção: Excode = 0
 - **IP** (*Interrupt Pending*, 8 bits) – 6 bits para correspondem a pedidos de interrupção de fontes externas e 2 a “software interrupts”
 - **Status Register (SR)** – determina quais as interrupções *enabled* (IM – **Interrupt Mask field**) e qual o nível de privilégio do CPU (kernel/user). Tem ainda um bit (**IEc** – Interrupt Enable current) que faz o *enable/disable* geral das interrupções

Exceções no MIPS (2)

- CPU tem 6 pins para pedidos de interrupção externos (6 linhas de *Interrupt Request*) – o seu estado é dado pelos bits do campo IP do Cause Register
- Identificação da fonte da interrupção feita por software (ISR lê campo **IP** do registo **Cause**).
- Embora as prioridades das várias linhas seja em geral fixa em cada sistema ela é estabelecida por software
 - ordem pela qual a ISR geral examina os bits do campo **IP**

Interrupções vectorizadas

- “**Vector de Interrupção**” identifica interrupção - colocado no bus de dados é usado pelo processador para indexar uma tabela em memória que contém os endereços das subrotinas de tratamento de interrupções (***Interrupt Table***)
 - A cada fonte de interrupção está associada uma subrotina de tratamento específica
- **Intel:** vector de interrupção: 1 *byte*
Interrupt Table (PC): endereços 0 .. 3FF_h (256 entradas de 4 bytes) – cada entrada contém CS e IP da *ISR* correspondente
$$\&ISR = 4 * (\text{Vector de Interrupção})$$

Nota: a partir do 80286 foi introduzida a instrução *LIDT* (Load Interrupt Descriptor Table) que permite modificar o endereço-base da *Interrupt Table* e alterar a respectiva dimensão

Interrupções na arquitectura Intel (**real mode**)

- Processador tem duas linhas de pedido de interrupção, com níveis de prioridade diferentes:

NMI – Non-Maskable Interrupt (não pode ser desactivada – prioridade máxima) – vector de interrupção (= 2) gerado internamente pelo processador

INTR – pode ser desactivada pela *flag I (interrupt enable)*:

I = 0 – INTR disabled

I = 1 – INTR enabled

- **PIC** – Programmable Interrupt Controller – possibilita que o sistema suporte múltiplas fontes de interrupção

Interrupt Table (Intel)

Memory Address Mem. Contents

3FF	IP ₂₅₅
	CS ₂₅₅
	IP ₁
4	CS ₁
	IP ₀
0	CS ₀

Intel: Sequência de atendimento de IT

1. Pedido de Interrupção detectado
2. Acknowledge da interrupção; ler vector de interrupção do *Data Bus* (colocado no Data Bus pelo PIC)
3. Colocar *flags* (PSW), CS e IP no stack; interrupções automaticamente desactivadas (flags **IF** e **TF** postas a 0) quando da salvaguarda das *flags* no stack
4. Indexar a Tabela de Interrupções para ler os novos valores de CS:IP (**salto para a ISR**)
5. Executar ISR
6. ISR termina com **IRET** (instrução de **Interrupt Return**) - restaura os valores de IP, CS e PSW (regresso ao programa interrompido)

Interrupt Service Routine

- Começa por salvar (no *stack*) o conteúdo dos registos que vai utilizar (antes de entrar na ISR apenas foi salva a informação mínima que permite retomar a execução do programa interrompido)
- Antes de terminar restaura o conteúdo dos registos
- Se durante a execução da ISR se quiser permitir o atendimento de interrupções de prioridade mais elevada necessário colocar a flag $I = 1$
 - ***STI*** – instrução Set Interrupt
 - ***CLI*** – instrução Clear Interrupt

Múltiplas fontes de Interrupção

- Necessário poder ter uma multiplicidade de periféricos com possibilidade de interromperem o processador
PC: teclado, rato, porto série, porto paralelo, controlador de diskettes, relógio de tempo real, ...
- Necessário definir o que fazer quando há mais do que um pedido de interrupção em simultâneo:
 - atribuição de diferentes **níveis de prioridade** (i.e. de urgência de atendimento) aos pedidos

Interrupções Múltiplas - alternativas

- várias linhas de ***Int Req*** com diferentes níveis de prioridade - Motorola 68000 (prioridade fixa), MIPS (prioridade estabelecida pelo software)
- processador com uma só linha de ***Int Req*** – Intel
 - lógica de arbitragem entre os pedidos dos diferentes periféricos incorporada num componente externo: ***PIC*** (**Programmable Interrupt Controller**)
 - Pentium: *APIC* (Advanced *PIC*)
 - i80188 (Intel *embedded processors* em geral): *PIC* on-chip

Intel – linhas de interrupção dos CPU

- ***NMI*** – Non-Maskable Interrupt (input) – sempre activa; vector de interrupção 2
- ***INTR*** – *INT*errupt *Re*quest (input) – apenas activa quando a flag ***I = 1***
- ***~INTA*** - *INT*errupt *A*cknowledge (output) – indica que o pedido de interrupção (***INTR***) foi reconhecido pelo CPU e que o vector de interrupção deve ser colocado no Data Bus

Controlador de Interrupções – i8259A

- 8 linhas de pedidos de interrupção priorizadas ($IR0, \dots, IR7$) – 8259A activa a sua saída INT (ligada à entrada $INTR$ do μP) sempre que uma dessas linhas, desde que corresponda a uma interrupção não “mascarada”, é colocada a 1 pelo interface do periférico.
- Quando recebe o $INTA$ (acknowledge) o 8259A coloca nas suas saídas $D0...D7$ (ligadas ao bus de dados) o vector de interrupção correspondente ao pedido de mais alta prioridade pendente

8259A – Modelo de programação (1)

- **Status Registers:**
 - **IRR** – Interrupt Request Register – indica quais os pedidos de interrupção que estão activos
 - **ISR** – In-Service Register – indica a interrupção que está a ser servida
- **IMR** – Interrupt Mask Register (**Control/Status**) - (OCW1) – especifica quais as fontes de interrupção desactivadas (*disabled*) – bit respectivo = 1

8259A – Modelo de programação (2)

- Inicialização – **ICW1, ..., ICW4 – Initialization Command Words** – definem o modo básico de operação – programadas quando o sistema é ligado; só depois de programadas é que o 8259 pode funcionar
 - **ICW2** – define o vector de interrupção usado com cada linha de pedido de interrupção (= valor do 1º vector de interrupção)
- **Operation Command Words – OCW1, ..., OCW3** – podem ser programadas durante o funcionamento do dispositivo
 - **OCW1** – Interrupt Mask Register

O sistema de Interrupções dos PC

- PCs com bus ISA: dois i8259 ligados em cascata (*master* e *slave*) – 15 linhas IRQ0, IRQ1, IRQ3...15 (7 + 8) para interrupções de periféricos
 - Master 8259: registos acedidos pelas portas 20H e 21H (I/O)
 - Slave 8259: registos acedidos pelas portas A0H e A1H (I/O)
- Inicializados durante *POST* (Power-On Self-Test)
 - ICW1 mapeado no port 20H (master) e A0H (slave)
 - ICW2, ICW3 e ICW4 mapeados no port 21H (A1H)
 - Qualquer posterior instrução *OUT A, 21H* (A1H) escreve em OCW1 (o Interrupt Mask Register)

O sistema de Interrupções dos PC (2)

- Dois i8259A fornecem 15 linhas de interrupção:
 - IRQ0 – System Timer
 - IRQ1 – Interface Keyboard
 - IRQ4 – Porta Série 1
 - IRQ6 – Controlador Floppy
 - IRQ7 – Porta Paralelo 1
 - IRQ8 – Relógio Tempo Real
 - IRQ12 – Interface Rato
 - IRQ14 – Controlador Disco (disponível no bus ISA)
 - IRQ10, IRQ11, IRQ15 –no bus ISA (disponíveis para serem utilizados por outras cartas de interface)

DMA vs. Interrupções

- I/O por interrupção:
 1. **Processador** envia ao periférico um comando de leitura
 2. Processador executa outras tarefas
 3. Periférico, quando termina execução do comando, gera **interrupção**.
 4. **Processador** verifica *Status* do periférico
 5. **Processador** lê informação do periférico
 6. **Processador** armazena a informação lida na memória
 7. **Fim? Não: goto 1**
 - Processador ocupado com simples transferência de dados
- Alternativa: sistema de I/O encarrega-se das transferências entre o periférico e a memória, sem envolvimento do processador
 - **DMA** – Acesso Directo à Memória

DMA – Acesso Directo à Memória

1. **Processador** envia ao **controlador de DMA** um comando (e.g. ler do periférico um **bloco de informação**)
2. Processador executa outras tarefas
3. **Controlador de DMA** verifica *Status* do periférico
4. **Controlador de DMA** lê informação do periférico
5. **Controlador de DMA** armazena a informação lida na memória
6. Fim da leitura do Bloco? Não: goto 3
7. **Controlador de DMA**, quando termina execução do comando, gera *interrupção*.

DMA – Comandos

1. **Processador** envia ao **controlador de DMA** um comando (e.g. ler do periférico um **bloco de informação**)
 - Qual a informação necessária ao controlador de DMA?
 - **Endereço**-base da zona **de memória** onde deve colocar os dados
 - **No. de elementos** do bloco a transferir
 - Qual o interface do controlador de DMA com o sistema?
 - Address Bus
 - Data Bus

Em ambos capacidade de actuar como **bus master**
- i8237 – Controlador de DMA - 4 canais de DMA

Stallings – Fig. 6.12

Stallings – Fig. 6.13

Stallings – Fig. 6.14