

Anotações

Exceções, Interrupções e I/O

Yuri Kaszubowski Lopes

UDESC

Exceções e Interrupções

- Eventos inesperados, que da mesma forma que branches e jumps, podem mudar a o fluxo normal do programa
- No MIPS:
 - ▶ **Exceção:** Algum evento interno inesperado que ocorreu dentro do processador
 - * e.g., overflow
 - ▶ **Interrupção:** Algum evento inesperado gerado externamente
 - * e.g., O usuário digitou algo no teclado

Anotações

Exceções e Interrupções

- Em muitas arquiteturas, como no x86, o termo interrupção é utilizado para definir tanto exceções quanto interrupções
- MIPS faz o contrário:
 - ▶ Tudo é uma exceção
 - ▶ Interrupção é um tipo especial de exceção, que foi gerada externamente
 - * Na prática, isso não nos afeta (são só terminologias diferentes)
 - ▶ Vamos utilizar a terminologia do MIPS, já que ela é usada no livro base da disciplina

Anotações

Exemplos

| Evento | Fonte | Terminologia MIPS |
|-----------------------------------|---------|-------------------|
| Solicitação de E/S | Externa | Interrupção |
| O programa chama o S.O. (syscall) | Interna | Exceção |
| Overflow aritmético | Interna | Exceção |
| Instrução inválida | Interna | Exceção |

Anotações

Tratando uma Exceção: Sistema Operacional

- Vamos considerar o caso de um overflow aritmético
 - `add $1, $2, $1`
 - o resultado não cabe nos 32 bits de `$1`

Passos básicos

- Salvar o endereço da instrução que causou a exceção
 - Salvar o endereço de PC em um registrador especial (EPC: Exception PC)
- Salvar algum código informando o que causou a exceção
 - Registrador **cause** no MIPS
 - e.g., código 1 em **cause** indica overflow, 2 significa instrução inválida
- Transferir a execução para o Sistema Operacional
 - As rotinas do kernel para o tratamento básico de exceções no MIPS iniciam no endereço 0x80000180
- O S.O. verifica o registrador cause para definir o que houve
 - Agora o S.O. tem a opção de tratar a exceção e retornar a execução do programa a partir do ponto salvo em EPC, ou terminar o programa
 - Caso o S.O. opte por terminar o programa, o EPC ainda pode servir para o debug do programa

Anotações

Tratando uma Exceção: Interrupções Vetorizadas

- Outra forma de tratar exceções é eliminar o registrador cause, e transferir o controle para um endereço específico, dependendo da fonte da exceção

| Tipo da exceção | Endereço de desvio |
|------------------------------|--------------------|
| Instrução indefinida | 0x80000000 |
| Overflow aritmético | 0x80000180 |
| Requisição de E/S do usuário | 0x80000360 |
| ... | ... |

Anotações

• Quais as vantagens e desvantagens de interrupções vetorizadas?

- ▶ - Precisamos de mais hardware
 - ★ A tabela de desvios precisa ser salva em algum lugar!
- ▶ - Menor flexibilidade
 - ★ Se um novo tipo de exceção precisar ser tratado em nosso hardware, precisamos criar uma nova entrada na tabela, ou fazer algum enjambre
- ▶ + Tratamento mais rápido
 - ★ Agora o S.O. não precisa ler cause e definir o que fazer: O fluxo do programa já é redirecionado para o endereço de tratamento específico

Tratando uma Exceção: Interrupções Vetorizadas

• x86:

- ▶ Utiliza interrupções vetorizadas

• MIPS:

- ▶ Utiliza registrador cause
- ▶ Um único cenário é tratado como uma interrupção vetorizada:
 - ★ Falta de páginas
 - ★ Tão comum que vale a pena otimizar esse tratamento através de vetorização
 - ★ Você vai aprender o que é uma falta de páginas em Sistemas Operacionais

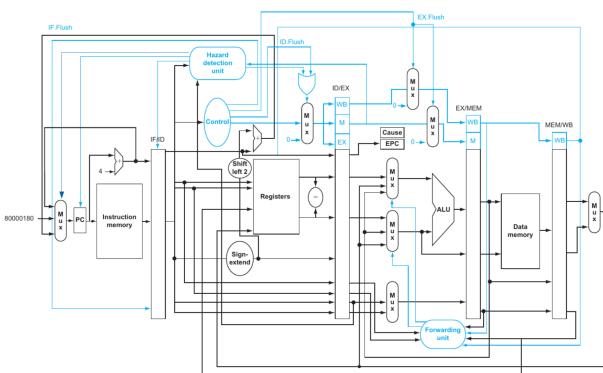
Anotações

Tratando exceções no pipeline

- Uma exceção é tratada como um hazard de controle
- Se a exceção foi causada internamente, devemos parar a execução imediatamente
 - ▶ Salvamos as informações
 - ▶ Descartamos as instruções anteriores que já estão no pipeline
 - ▶ Redirecionamos para o S.O. (ou uma rotina de tratamento qualquer)
- Note que as ações são muito similares as tomadas em um desvio condicional, para o qual a previsão de desvio estava incorreta
 - ▶ A diferença é que precisamos salvar algumas informações, e o endereço de desvio é fixo (0x80000180)

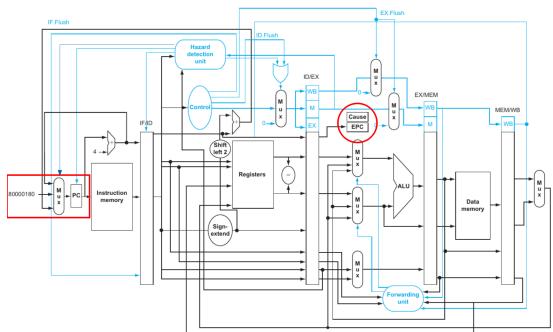
Anotações

Tratando exceções no pipeline



Anotações

Tratando exceções no pipeline



Anotações

- Note que estamos salvando PC +4 em EPC. O S.O. deve levar isso em consideração ao tratar uma exceção nesse processador MIPS.
- A unidade de detecção de hazard pode redirecionar para 0x80000180

Comunicando-se com dispositivos

Atenção

Esse trecho foi baseado no livro de Tanenbaum, Bos (2016), que tem um foco em S.O., e utiliza instruções similares a do x86

Anotações

- Todo dispositivo de E/S é composto por pelo menos dois componentes:
 - Parte mecânica: e.g., HD, teclado
 - Parte eletrônica: Possui pelo menos alguns registradores para indicar o que o componente está fazendo, ou precisa fazer
 - ★ registrador de 8 bits em uma impressora simples, onde escrevemos o próximo caractere que a impressora deve imprimir

Espaço de E/S

Anotações

- Controladores especializados (que podem ser encontrados na placa mãe ou dentro da própria CPU) podem dar um número único para cada registrador de cada dispositivo de E/S conectado
 - Número de Porta de E/S
- Podemos então desenvolver instruções para nossa CPU, que recebe um número de Porta de E/S e o valor a ser escrito
- Exemplos (instruções estilo x86):
 - `in REG, PORTA`: lê o conteúdo da porta especificada e armazena em REG
 - `out PORTA, REG`: escreve o conteúdo de REG para a porta especificada
 - Em ambos os casos, REG é um registrador da CPU, e PORTA é o identificador de um registrador de um dispositivo de E/S qualquer

E/S Mapeada em Memória

- Outra opção é o controlador dar um endereço de memória para cada registrador de E/S
 - ▶ Ao Ler/escrever nesse endereço de memória, o controlador detecta, e redireciona os dados para o registrador do dispositivo
 - * Nada é realmente lido/escrito na memória principal da máquina
- Chamamos de E/S mapeada em memória
 - ▶ Agora instruções lw e sw podem ser usadas para escrever na memória, e também para nos comunicar com os demais dispositivos de E/S
 - ▶ Precisamos “apenas” de um controlador que intercepta o sinal da CPU, e redireciona para o local correto

Anotações

E/S

- MIPS utiliza E/S mapeada em memória
 - ▶ E/S mapeada em memória é também comum em microcontroladores
 - * Família de microcontroladores PIC e ATMega
- X86 utiliza ambas técnicas
 - ▶ Possui instruções especializadas para lidar com portas de E/S
 - ▶ Memória entre [0-(64K-1)] reservada para portas de E/S

Anotações

Enviando sinais ao processador

- Um dispositivo de E/S pode enviar um sinal ao processador
- E/S baseada em interrupção
- Precisamos de sinais de controle externos em nosso processador
- Quando o processador recebe um sinal externo, ele gera uma **interrupção** (exceção gerada por um dispositivo externo)
- O processador pode também ler do barramento externo informações sobre a interrupção
 - ▶ e.g., no barramento podemos incluir informações sobre o hardware que solicitou a interrupção

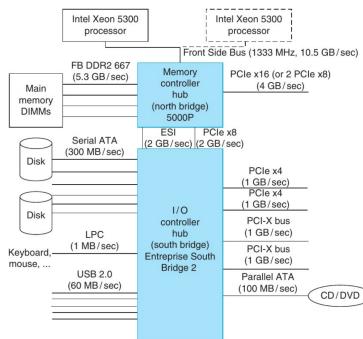
Anotações

Enviando sinais ao processador

- Quando uma **interrupção** é gerada, o processador pode terminar de executar as instruções que já estão no pipeline, antes de redirecionar para o S.O.
 - ▶ A exceção não é interna, e não tem **nada que impeça as instruções** que estão na metade do caminho de **terminarem**
 - ▶ Evitamos stalls
 - ▶ A interrupção pode ser assíncrona com relação as instruções que estão dentro da CPU
- O processador armazena as informações sobre a interrupção, redireciona para o S.O., e o restante do tratamento é o mesmo de uma exceção qualquer
 - ▶ O problema agora é do S.O. ou da rotina de tratamento

Anotações

Mundo Real

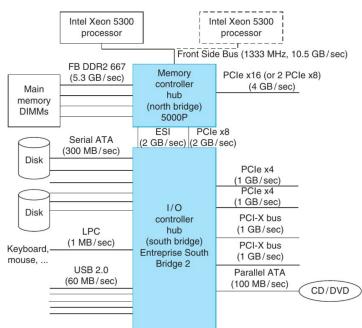


Os principais controladores que fazem as traduções e enviam as interrupções dos dispositivos de E/S para a CPU são a ponte norte e a ponte sul

Anotações

Mundo Real

- **Ponte Norte:**
 - ▶ Mais próxima da CPU
 - ▶ Conecta os dispositivos que exigem maior largura de banda
 - ▶ Memória, PCIe x16, ...
 - ▶ Caminho para a ponte sul
- **Ponte Sul:**
 - ▶ Conecta os dispositivos mais lentos
 - ▶ Mouse, teclado, placa de rede, discos, ...



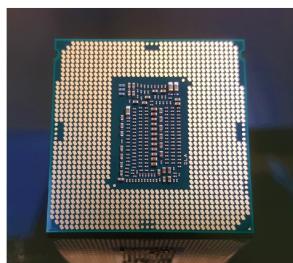
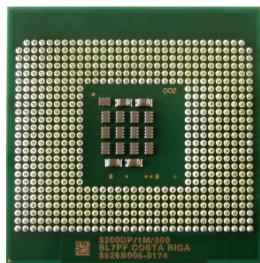
Anotações

Processadores atuais

- A maioria dos processadores x86-64 atuais incorporaram as pontes norte e sul
 - ▶ Esses controladores ainda existem, mas se encontram dentro da CPU
 - ▶ A ideia dos projetistas é reduzir os atrasos no circuito com isso, criando um controlador otimizado que está muito próximo aos núcleos de processamento
 - ▶ Problemas:
 - ★ Agora tudo deve se comunicar "diretamente com a CPU"
 - ★ Gastamos área do chip da CPU com esses controladores
 - ★ Podemos precisar de pinos extras para conectar a CPU, pois as pontes externas que faziam a "filtragem" deixam de existir
 - ★ Ainda temos pequenos "hubs" externos, mas não fazem todo o trabalho que as pontes fazem

Anotações

Exemplos



Intel Xeon para Socket PPGA604 (mesmo socket do Xeon 5300) **604 Pinos** Processador de 2006

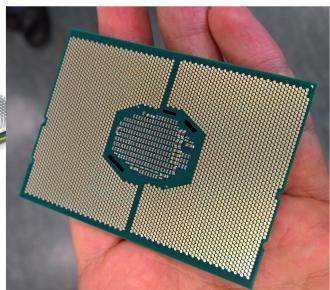
As imagens não estão em escala!

Anotações

Exemplos



Processador Xeon para socket LGA 3647 (3647 pinos para "se comunicar com o mundo")



Processador Xeon para socket LGA 3647 (**3647 pinos**)

Anotações

Exercícios

- ➊ Abra o simulador Mars e verifique que processadores MIPS realmente possuem os registradores cause e EPC. Eles ficam na aba "Coproc 0" - Coproc 0 é comumente o coprocessador de exceções do MIPS, enquanto Coproc 1 é o coprocessador de ponto flutuante.
- ➋ Note que múltiplas exceções podem ocorrer ao mesmo tempo em nosso pipeline
 - ▶ Ao mesmo tempo que uma instrução gera um overflow em EX, outra gera uma instrução inválida em ID, e outra gera um endereço inválido em IF, ...
 - ▶ Como você trataria isso? Qual exceção deve ser priorizada?
- ➌ O S.O. é o responsável por tratar as exceções. O S.O. também é um programa, que **utiliza os mesmos registradores** do seu programa na CPU. Descreva o trecho de código que o S.O. **sempre** deve executar **antes, e depois** do tratamento da exceção, **para que as informações salvas nos registradores não se percam**, e seu programa possa continuar executando normalmente após uma exceção (se o sistema operacional decidir que o seu programa pode continuar executando).
- ➍ Pesquisa: O que é DMA (Direct Memory Access)?

Anotações

Referências

- ➊ D. Patterson; J. Hennessy. **Organização e Projeto de Computadores: Interface Hardware/Software**. 5a Edição. Elsevier Brasil, 2017.
- ➋ TANENBAUM, Andrew S. Sistemas operacionais modernos. 10. ed. Pearson Education do Brasil, 2018.
- ➌ STALLINGS, William. Arquitetura e organização de computadores. 8. ed. São Paulo: Pearson Education do Brasil, 2010.
- ➍ J. Hennessy; D. Patterson. Arquitetura de computadores: Uma abordagem quantitativa. 5a Edição. Elsevier Brasil, 2014

Anotações

Anotações
