Universidade Federal de Minas Gerais
Escola de Engenharia – Departamento de Engenharia Eletrônica
Laboratório de Arquitetura e Organização de Computadores
Assunto: Projeto de uma Controladora de Interrupções
Autoria: Prof. Ricardo de Oliveira Duarte

Objetivos:

Explicar o funcionamento de uma controladora de interrupções.
Apresentar os passos necessários ao projeto de uma controladora de interrupções.

Seção 1: Necessidade de uma controladora de interrupções para uma CPU:

1) Habilitar ou desabilitar a atenção da CPU às exceções (eventos internos).
2) Habilitar ou desabilitar individualmente, a atenção da CPU aos eventos de periféricos (eventos externos).
3) Registrar para a CPU a ocorrência de alguma exceção.
4) Priorizar para a CPU o atendimento de uma determinada exceção à outra.
5) Registrar para a CPU o endereço da memória onde se iniciará a rotina de tratamento da exceção.
6) Registrar para a CPU o endereço de retorno da memória de programa, após o término da execução da rotina de tratamento de exceção.

Seção 2: Projeto da controladora de interrupções:

Passo 1: Determinar quais elementos da via de dados que proporcionarão o armazenamento do estado em caso de ocorrência de interrupção.

Para atender os itens 1 e 2 da seção 1 será necessário um registrador para armazenar o estado (habilitado/desabilitado). Chamaremos esse registrador de IER (*Interrupt Enable Register*). Devemos reservar 1 bit desse registrador para habilitar/desabilitar qualquer tipo de interrupção (esse bit funciona como uma chave geral de um quadro de luz). Os demais bits do registrador IER estarão associados, cada um, a uma interrupção de um periférico ou tipo de exceção especifica. Gravar "1" em um determinado bit do registrador IER, significa habilitar o tipo de exceção correspondente a aquele bit do IER. Para habilitar/desabilitar qualquer tipo de exceção, devemos primeiramente enviar um comando de desativação geral de qualquer tipo de exceção, usando a chave geral. Ao término da configuração desejada, voltamos a habilitar o sistema para atenção a qualquer tipo de exceção (habilitar a chave geral).

Para atender o item 3 da seção 1 será necessário um outro registrador, o qual chamaremos de registrador IFR (*Interrupt Flag Register*). Esse registrador será responsável por registrar a ocorrência de alguma exceção, enquanto a CPU estiver processando qualquer passo de qualquer instrução. Cada bit do registrador IFR está também associado a um tipo de exceção. É desejável que haja uma correspondência direta com as posições do registrador IER, somente por razão de clareza. Gravar "1" em um determinado bit do registrador IFR significa que o periférico ou a exceção correspondente solicitou a atenção da CPU.

Para atender o item 4 pode-se organizar dentro do registrador IFR a ordem de atendimento das exceções, no caso de haver mais de uma exceção ocorrida (marcada). A ordem é você quem decide, do bit mais significativo para o menos significativo, ou o contrário. No caso de uma CPU comercial há um registrador especifico para armazenar a ordem de prioridade de tratamento de um certo periférico. Esse registrador se chama IPR (*Interrupt Priority Register*) e normalmente está associado a um periférico especifico. No nosso caso, não haverá necessidade de um IPR para cada periférico.

Para atender o item 5 da seção 1, ou seja, para registrar o endereço da rotina de tratamento de interrupções (ISR – *Interrupt Service Routine*), armazenada na memória de programa, destina-se uma região continua de memória reservada para armazenar tais endereços. Essa região contínua pode ficar na

região da memória de programa, na memória de dados ou em uma região de memória interna dedicada para armazenar tais endereços (chamaremos de *scratchpad memory*). Denominamos essa forma de tratamento de exceções/interrupções de interrupção vetorada. Cada elemento desse vetor será um endereço da memória de programa para o início da ISR correspondente. E o índice desse vetor determinará o tipo de interrupção ocorrida, ou seja, a causa (registrador *CAUSE*) da ocorrência da interrupção.

Por fim, para atender o item 6 da seção 1, para registrar o endereço de retorno da memória de programa, antes da chamada da ISR, há a necessidade de se destinar um registrador especifico para tal registro. Trata-se do registrador EPC (*Exception Program Counter*) ou RAR (*Return Adress Register*) em outras arquiteturas. O endereço da próxima instrução da memória de programa a ser executada pela CPU, antes da execução da primeira instrução da ISR deverá ficar salvo dentre desse registrador, para que, ao término da execução da ISR, a CPU possa retornar ao processamento do programa principal de onde parou, tudo isso antes de começar a tratar a interrupção solicitada.

Algoritmo da rotina de tratamento de qualquer exceção (a ser programado quando vocês escreverem um código fonte para testes com *assembly* da sua CPU):

1) Desabilitar o bit responsável para atenção geral às exceções.
2) Limpar o bit do registrador IFR correspondente a exceção que originou a execução da ISR em questão.
3) Realizar a sequência de ações desejada para o tratamento da exceção.
4) Voltar a habilitar o bit responsável para atenção geral às exceções.
5) Retornar ao ponto de execução apontado pelo conteúdo armazenado em EPC ou RAR.

Passo 2: Disponha os elementos de estado na via de dados da sua CPU e interligue-os de maneira a permitir que uma ISR seja executada (ou tratada) por sua CPU.

Passo 3: Determine se alguma instrução extra no conjunto de instruções de sua CPU deverá ser implementada, seja para executar qualquer ação no processo de tratamento de interrupção ou no processo de configuração de qualquer tipo de interrupção.

Passo 4: Escreva um programa em linguagem C que configura à atenção da CPU um determinado tipo de exceção/interrupção e sua ISR correspondente.

Passo 5: Converta o programa que você escreveu no passo anterior para a linguagem *assembly*.

Passo 6: Converta o programa que você escreveu no passo anterior para a linguagem de máquina.

Passo 7: Modifique a unidade de controle da sua CPU, ou implemente a unidade de controle da sua controladora de interrupções.

Passo 8: Crie um *testbench* para simular a sua CPU com a ocorrência e o tratamento de um tipo de interrupção. (Ex.: por ora, o tratamento de overflow será uma boa escolha como exceção, pois não temos outros periféricos implementados e associados a endereços de memória).

Passo 9: Compile a sua CPU com a controladora de interrupções e corrigir eventuais erros de compilação.

Passo 10: Simule o seu sistema com o *testbench* criado no passo.