Interrupções de Hardware (e software)

Introdução

O mecanismo de INTERRUPÇÃO é utilizado em todos os processadores comerciais e existe em uma forma simplificada no CESAR16i. Seu uso é essencial para permitir a operação dos sistemas operacionais modernos. Sem esse mecanismo seria muito difícil realizar as tarefas comuns nos sistemas computacionais atuais.

No texto que segue são apresentados os princípios das interrupções, como elas existem nos processadores comerciais e como são as interrupções no CESAR16i.

Mecanismo de Chamada de Subrotinas

Para entender como é a operação desse importante mecanismo, vamos utilizar um mecanismo já conhecido: a "**chamada de subrotinas**". Para isso, é necessário entender os elementos componentes de sua operação. Os elementos presentes na realização da chamada de uma subrotina são os seguintes:

- Mecanismo de acionamento da subrotina: é a chamada da subrotina, propriamente dita;
- Endereço de destino: endereço onde inicia o código da subrotina;
- Salvamento de dados de retorno: endereço(s) na memória onde será salvo o endereço de retorno da interrupção assim como a forma de realizar esse salvamento. Essa informação será utilizada ao encerrar a subrotina.

Cada processador realiza a chamada de subrotinas de forma diferente. Portanto, implementam os elementos listados acima de forma diferente. Vamos analisar cada um desses elementos.

Mecanismo de acionamento: trata da forma como a subrotina é acionada. Em geral, os processadores utilizam uma instrução específica para essa finalidade. No caso do processador RAMSES assim como no CESAR16 utiliza-se uma instrução "JSR", que é responsável por salvar os dados que serão usados no final da subrotina, e então realiza um "salto". Entretanto, devido à diferença existente nos outros dois elementos (endereço de destino e forma e local de salvamento dos dados de retorno), a forma como se dá esse processo de chamada é diferente em cada processador.

Endereço de destino: a chamada da subrotina envolve um "salto" para o início da subrotina. Esse endereço está definido como um operando da instrução. No caso do RAMSES esse é o único operando da instrução; no caso do CESAR16 esse endereço é o segundo operando da instrução. Portanto, o endereço de retorno é indicado, implicitamente, pelo programador, na própria instrução que realiza a chamada da subrotina.

Salvamento de dados de retorno: conforme explicado no mecanismo de acionamento, a chamada de subrotina envolve o salvamento de dados que serão usados quando esta terminar, no "retorno da subrotina". Os dados de retorno podem ser os mais variados. Mas, sem dúvida alguma, aquele que sempre está presente é o endereço de retorno. Esse endereço corresponde ao endereço da instrução imediatamente posterior àqueles usados pela instrução de chamada da subrotina. Dessa forma, no final da execução da subrotina o processador deverá realizar um "salto" para o endereço salvo. Esse elemento é totalmente diferente quando se compara o processador RAMSES e o processador CESAR16. Recomenda-se ler a documentação dessa instrução nos processadores RAMSES e CESAR16.

Cabe ressaltar que o mecanismo de “pilha” implementador CESAR16 é aquele amplamente utilizado nos processadores atuais por possibilitar a programação com reentrância, recorrência e número infinito de níveis de aninhamento (na realidade, é limitado pelo tamanho da memória). Em termos gerais, o endereço onde se deve retomar a execução é aquele que está armazenado no “topo da pilha” (vide mecanismo de pilha do processador CESAR16).

Mecanismo de Reconhecimento de Interrupção

O “**mecanismo de interrupção de hardware**” também apresenta os mesmos três elementos. Esses elementos são os seguintes:

Mecanismo de acionamento: a interrupção é ativada através do acionamento de um sinal de hardware disponibilizado pelo processador (“INT” na Figura 1), onde se deve conectar um sinal de saída do periférico (“OUT” na Figura 1). Sempre que o periférico acionar esse sinal de saída, o processador perceberá esse fato e poderá “reconhecer a interrupção”, iniciando o processo de chamada da “subrotina” de interrupção. Essa subrotina é chamada de “tratador de interrupção” (*Interrupt Service Routine* ou *Interrupt Handle*).

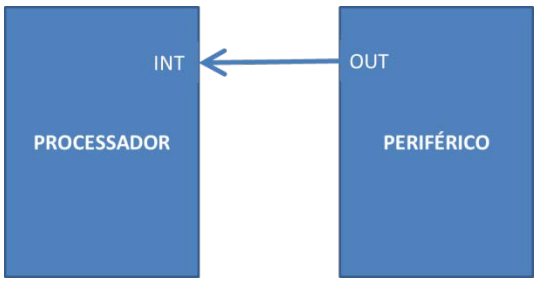

Figura 1:Conexão do sinal do periférico à entrada de interrupção

É importante salientar que o processador só pode “aceitar” um pedido de interrupção entre duas instruções[1], mais precisamente após o término da execução de uma instrução e imediatamente antes da busca (“*fetch*”) da instrução seguinte. Na Figura 2 (a) é apresentado o ciclo de operação dos processadores, composto por “busca”, “decodificação” e “execução”. Na Figura 2 (b) é apresentado esse ciclo quando se acrescenta o reconhecimento da interrupção, que só pode acontecer após o término das instruções. Observar, também, que o bloco de reconhecimento da interrupção da Figura 2 (b) representa a ativação da interrupção, devendo o processador retomar o ciclo padrão de operação logo após essa ativação.

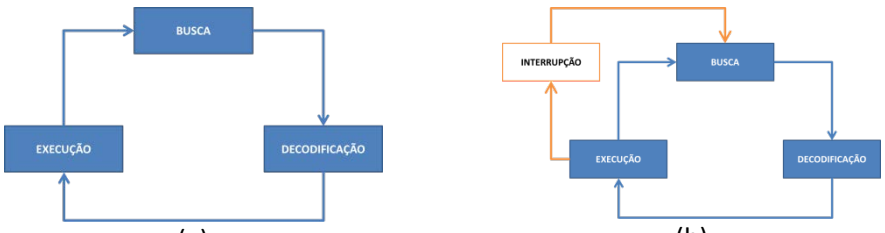

Figura 2 :Ciclos de operação dos processadores: (a) sem interrupção e (b) com interrupção

[1] Alguns processadores são capazes de reconhecer a solicitação de interrupção durante a execução das instruções. Essas são interrupções especiais e não é o caso do CESAR16i. Um exemplo desse tipo de interrupção é aquele usado pelos mecanismos de memória virtual.

Endereço de destino: a chamada de uma subrotina é realizada pela execução de uma instrução específica onde um dos operandos indica o endereço da subrotina. No caso da interrupção não há como saber quando ocorrerá a solicitação da interrupção, uma vez que esse evento dependerá apenas do periférico. Portanto, é necessário definir o endereço do tratador de interrupção antes que ela possa acontecer. Existem várias formas de fazer isso. Cada processador pode definir seu modo de operação e alguns deles, inclusive, podem oferecer mais de uma forma para realizar esse reconhecimento (vide expressões como "*Nonmaskable Interrupts*" e "*Vectored Interrupts*"). No processador CESAR16i o endereço do tratador de interrupção deve ser escrito em dois bytes chamados de "vetor de interrupção", localizados a partir de 0FFBEH (65470). Como já explicado, a escrita do valor adequado no vetor de interrupção deve ser feita antes que as interrupções possam ser reconhecidas.

Salvamento de dados de retorno: durante o reconhecimento da interrupção o processador deve salvar o endereço de retorno de maneira que, ao final da execução do tratador de interrupção, o processador possa continuar a partir do endereço salvo. No caso do processador CESAR16i, além do endereço do retorno, o processador salva na "pilha" os códigos de condição (também chamados de "*flags*"). O motivo disso é que, diferentemente de uma chamada de subrotina, não é possível prever onde vai ocorrer o reconhecimento da interrupção e, portanto, não é possível salvar o conteúdo de códigos de condição que venham a ser alterados durante a execução do tratador de interrupção.

Então, ao ser reconhecida uma interrupção, são salvos na pilha os códigos de condição e o endereço de retorno. Assim, ao encerrar o tratador de interrupção, também se deve realizar um retorno que retire da pilha o endereço de retorno assim como os códigos de condição. Portanto, o retorno do tratador de interrupção deve ser realizado por uma instrução diferente daquela usada no retorno de uma subrotina: o retorno de uma subrotina utiliza uma instrução RTS enquanto que o retorno do tratador de interrupção deve utilizar uma instrução RTI (*Return From Interrupt*).

Portanto, o mecanismo completo de reconhecimento, execução e retorno da interrupção é o seguinte:

- Um periférico aciona um sinal que está conectado à entrada de interrupção do processador;
- O processador, sempre após a execução das instruções, verifica se esse sinal foi acionado;
- Se foi acionado, o processador salva na pilha o conteúdo dos códigos de condição e o valor do "*Program Counter*", que é o endereço da próxima instrução a ser executada (é o endereço de retorno) e executa um "salto" para o endereço escrito no vetor de interrupção;
- O processador executa o tratador de interrupção até que encontre uma instrução RTI, que retorna o conteúdo dos códigos de condição da pilha assim como o endereço de retorno, que é escrito no "*Program Counter*", realizando um "salto" para o endereço de retorno.

Interrupções no CESAR16i

O processador CESAR16i disponibiliza duas entradas de interrupção, que estão conectadas às saídas dos periféricos teclado e temporizador, conforme representado na Figura 3. Essas entradas possuem circuitos do tipo "*flip-flop*" capazes de armazenar uma solicitação de interrupção e evitando, assim, que sinais muito rápidos fornecidos pelos periféricos venham a ser perdidos.

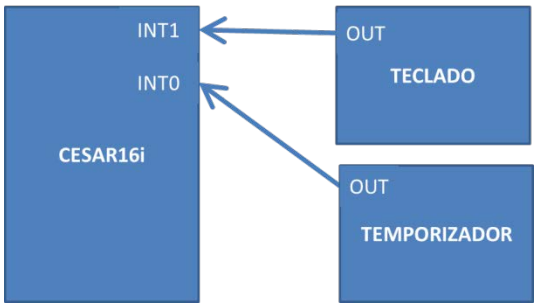

Figura 3: Interrupções no CESAR16i

Para controlar e monitorar as interrupções os programas devem escrever e/ou ler o conteúdo de alguns endereços específicos da memória. Esses endereços são o IVET (vetor de interrupção, nos endereços 0FFBEH e 0FFBFH), INTE (controle de habilitação das interrupções, no endereço 0FFD9H) e INTS (monitoramento do estado das interrupções, no endereço 0FFD8H).

O vetor de interrupção deve ser escrito com o endereço onde inicia o tratador de interrupção (ISR – *Interrupt Service Routine*). Esses endereços devem ser escritos em formato Big-endian (padrão do CESAR16i), antes que as interrupções possam ser reconhecidas.

O byte INTE (*Interrupt Enable*) deve ser utilizado pelo programa para controlar quais as interrupções que podem ser reconhecidas: aquelas que estão habilitadas. Sempre que o processador realizar a leitura ou escrita desse endereço isso será feito no formato byte (como acontece com o visor ou o teclado) e não no formato word (como acontece quando é realizada a leitura ou escrita da memória). Apenas três dos oito bits desse endereço tem significado, conforme representados na Figura 4. Os bits restantes devem ser mantidos em “0”, para garantir a compatibilidade com futuras versões do processador.

Figura 4 : Detalhamento de INTE

O bit “IE” (INTE.7) é o controle geral das interrupções. Se esse bit estiver desligado (em “0”), nenhuma interrupção será reconhecida. Caso esse bit esteja ligado (em “1”), então serão reconhecidas apenas as interrupções cujos bits específicos estiverem ligados. Os bits específicos são “IEStec” e “IEStim”: “*interrupt enable source*” 1 e 0, respectivamente.

O bit “IEStec” (INTE.1) é o controle específico da interrupção ligada à entrada 1, que está conectada à saída do periférico “teclado”. Portanto, para que uma interrupção de teclado seja reconhecida é necessário que os bits INTE.7 e INTE.1 estejam ligados.

O bit “IEStim (INTE.0) é o controle específico da interrupção ligada à entrada 0, que está conectada à saída do periférico “temporizador” (*timer*). Portanto, para que uma interrupção de temporizador seja reconhecida é necessário que os bits INTE.7 e INTE.0 estejam ligados.

Deve-se observar que existe apenas um vetor de interrupção, apesar de existirem duas fontes de interrupção. Isso significa que, qualquer que seja a fonte de interrupção, a rotina chamada será a mesma. Então, como saber qual periférico acionou a interrupção? A resposta está no byte INTS (*Interrupt Status*), que será usado para identificar qual foi a fonte da interrupção ou mesmo se houve a coincidência que os

dois periféricos geraram interrupções simultaneamente. De forma semelhante ao INTE, apenas três bits desse byte tem significado, conforme apresentado na Figura 5. Não há garantia sobre o valor dos bits restantes desse byte, para garantir a compatibilidade com futuras versões do processador.

Figura 5 : Detalhamento de INTS

O bit "IP" – *Interrupt Pending* – (INTS.7), quando ligado (em "1") indica que o processador está executando o tratador de interrupção. Esse bit é salvo na pilha junto com os códigos de condição, quando é reconhecida uma interrupção e é retornado quando executado o RTI. Esse bit tem a função de permitir o controle do tratador de interrupção, caso se deseje utilizar um tratador de interrupção reentrante.

Para identificar o periférico responsável pelo acionamento da interrupção deve-se utilizar os bits de indicação da fonte de interrupção pendente. São os bits "IPStec" e "IPStim": "*Interrupt Pendind Source*" 1 e 0, respectivamente. Esses bits correspondem às saídas dos "*flip-flops*" cujas entradas estão ligadas aos sinais de hardware de solicitação de interrupção.

O bit "IPStec" (INTS.1) indica que a interrupção pendente foi originada pelo periférico ligado à entrada 1, que é o teclado.

O bit "IPStim" (INTS.0) indica que a interrupção pendente foi originada pelo periférico ligado à entrada 0, que é o temporizador.

Portanto, no tratador de interrupção deve-se verificar qual dos bits foi acionado de maneira a decidir qual operação executar: se o tratamento do periférico teclado ou o periférico temporizador. Notar que pode ocorrer dos dois periféricos solicitarem, simultaneamente, a interrupção. Para atender a essa situação a rotina de interrupção deverá possibilitar o atendimento aos dois periféricos.