Capítulo 4

Exceções e Interrupções

Exceções são os eventos que alteram a execução normal de um programa. **Processamento de exceção** refere à forma como um processador responde a uma exceção; enquanto **tratamento de exceção** refere à ação tomada pela rotina de serviço.

Classificam-se os eventos que alteram o ciclo normal de execução de um processador em:

internos: quando os sinais não são provenientes do meio externo. Estes podem, por sua vez, ser diferenciados em duas categorias: os executados deliberadamente pelo programador e os decorrentes de erros. Na primeira categoria, temos *tracing* e *trap* e na segunda, erro de endereçamento, instrução inválida ou não existente, violação de privilégio, *overflow*, divisão por zero, etc.

externos: quando os sinais são provenientes do meio externo, como interrupção e inicialização (*reset*).

Fig. 6.19 do livro-texto (p. 461) mostra uma classificação das exceções suportadas pelo MC68000 em as geradas por *hardware* e as geradas por *software*. Para captar os eventos externos, MC68000 provém os pinos RESET*, BERR* e IPL0*–IPL2*.

É interessante ressaltar que uma exceção nem sempre corresponde a uma interrupção. Chamamos de **interrupção** somente exceções geradas pelos periféricos, externos ao processador, para **solicitar a sua atenção**. Isso permite tratar de forma mais eficiente as transações de entrada/saída. Diferentemente do mecanismo de **polling**, que é de natureza síncrona em relação

à sequência de operação de um processador, os pedidos de interrupção tem natureza assíncrona. Usamos a expressão "solicitar a sua atenção", porque um processador pode negar ou mascarar este tipo de exceção através de **flag de interrupção**, ou por **mascaramento de linhas de interrupção**, ou através do **marcaramento por nível de prioridade**.

Outra peculiaridade de uma interrupção é que uma linha de interrupção pode ser utilizada por vários periféricos, como ilustra Fig. 6.1 do livro-texto (p. 436). Em outras palavras, nem sempre é 1:1 a correspondência entre sinais de interrupção e os periféricos ligados ao computador. Quando este é o caso, circuitos adicionais são necessários para identificar a fonte do sinal de interrupção. Isso pode ser feito de forma síncrona com uso do mecanismo de polling (*software*) – o registrador de estado de cada periférico ligado à linha ativada é consultado sequencialmente para identificar o responsável pela ativação da linha, ou por *hardware* – o sinal de IACK∗ é transferido para os periféricos ligados à mesma linha, um por um até chegar no periférico solicitante.

Cada tipo de exceção é associado a uma rotina de serviço, cujos endereços podem ser organizados na Unidade de Memória em:

- posições não contíguas de memória;
- posições contíguas como um vetor de endereços; e
- posições contíguas como um vetor de pares compostos por endereço e modo de operação.

Essencialmente, a seguinte sequência de ações é tomada pela UCP quando uma exceção é atendida:

1. terminar o seu ciclo de instrução corrente.
2. o conteúdo de PC é armazenado numa pilha.
3. o conteúdo do registrador de estado SR é armazenado numa pilha.
4. desviar para rotina de serviço. Após a execução da rotina, tanto o conteúdo do SR quanto o conteúdo do PC são recuperados para retornar o controle ao programa que foi interrompido.

Com exceção das interrupções, a identificação da rotina de serviço de cada exceção é feita internamente pela UCP. MC68000 oferece dois esquemas para tratar as interrupções: **vetorizado** e **auto-vetorizado**. No vetorizado os periféricos identificam as suas rotinas de serviço no ciclo de reconhecimento de interrupção (**ciclo de IACK**). Este esquema é compatível com os periféricos (modernos) da série 68000. No auto-vetorizado, isso é deixado a cargo da UCP, o que facilita o interfaceamento com os periféricos de 8-bits da série 6800.

O fato de poder interceptar a operação sequencial de um programa amplia a capacidade de um processador (1) no tratamento de erros gerados durante a execução de uma instrução protegendo-o de "cair" num estado indefinido e (2) na interação assíncrona com o mundo externo. Nese capítulo, utilizaremos o processador MC68000 para demonstrar isso. Mais especificamente, tentaremos responder as seguintes questões:

- como as exceções são processadas em MC68000?
- qual é o mecanismo de interrupção suportada pelo MC68000? e quais são os sinais utilizados para sincronizar as microoperações entre MC68000 e os periféricos em cada ciclo de IACK?
- quais são as exceções suportadas pelo MC68000?
- além do mecanismo de processamento de exceção provido pelo MC68000, quais recursos adicionais são necessários para tratar corretamente uma exceção?
- como um sistema operacional (*software*) pode se beneficiar destas exceções?
- como um sistema de memória mais sofisticado, como memória virtual, pode ser implementado com uso destas exceções?
- como as funcionalidades do processador MC68000 podem ser estendidas com uso destas exceções?

Exercícios de Revisão

1. O que é um evento externo e um evento interno?
2. Quais são os pinos no MC68000 que captam os eventos externos assincronamente?

3. Resolver os exercícios 1 e 2 do capítulo 6 do livro-texto.
4. O que é uma exceção? E uma interrupção?

4.1 Processamento de Exceções em MC68000

MC68000 distingue as exceções em três grupos de acordo com a sua prioridade e o tipo:

GRUPO	Exceção	Características
0	Erro no barramento Erro de endereçamento	Processamento de exceção começa assim que o evento for reconhecido (no espaço de 2 ciclos de relógio)
1	Trace Interrupção Instrução inválida Violação de privilégio	Processamento de exceção começa antes da próxima instrução
2	TRAP, TRAPV CHK DIVZ	Processamento de exceção começa na execução normal de instrução

Os vetores de exceção são instalados numa **tabela** de 256 palavras longas ocupando os endereços de \$00 0000 a \$00 03FF. Os números de vetores de exceção de 0 a 63 são reservados a exceções e de 64 a 255 a interrupções (Tab. 6.2 do livro-texto, p. 448). Este números, multiplicados por 4, correspondem aos endereços da Unidade de Memória onde estão armazenados os endereços da respectiva rotina de serviço.

Observe que o vetor de exceção da (exceção) RESET é o único que ocupa 2 palavras longas e está armazenado no espaço de programa do supervisor (durante o seu acesso o código de função, FC2 FC1 FC0, é igual a 110). A primeira palavra é reservada para apontador de pilha do supervisor, SSP, e a segunda para o endereço da rotina de serviço que reinicializa (*reset* ou *reboot*) o sistema. Isso é porque MC68000 precisa ter um apontador de pilha bem definido para processar qualquer exceção.

O mecanismo para deferir ou indeferir uma solicitação de interrupção é por nível de prioridade. Três bits do registrador de estado (Fig. 6.13 do livro-texto, p. 454) suportam este controle. Exceto uma interrupção do nível 7, que é indeferível, qualquer interrupção só é atendida pela UCP se tiver um nível de prioridade maior do que o nível indicado no SR. Em outras palavras, MC68000 suporta **interrupção por nível**.

O processamento de exceção em MC68000 ocorre em quatro fases (Fig 6.18 do livro-texto, p. 460):

Fase 1: Inicialização da exceção

- uma cópia do registrador de estado SR é feita internamente no MC68000;
- o modo de operação é setado para o modo de supervisor com execução contínua, ou seja setar os bits do SR S=1 e T=0; e
- se for a exceção RESET, os bits do nível de interrupção no SR são setados 111 (nível máximo) e se for uma interrupção, os bits são setados no nível da interrupção que está sendo atendida.

Fase 2: Determinação do número de vetor de exceção. Com exceção da interrupção vetorizada, o número de vetor é gerado internamente pelo processador e a partir dele é determinado o endereço da rotina de serviço.

Fase 3: Armazenamento do contexto corrente (porção volátil) do processador na pilha do supervisor, de forma a permitir que este contexto seja recuperado após o tratamento da exceção. A quantidade de informação a ser armazenada depende da natureza da exceção. Para exceções do grupo 1 e 2 é guardada uma quantidade mínima de informação: o conteúdo do PC e o conteúdo do SR (que foi salvo na fase 1!); e para as do grupo 0, dados adicionais são retidos para facilitar o diagnóstico da causa da exceção e ajudar na decisão por ações convenientes. Mais especificamente, estes dados adicionais são: a instrução, o endereço, o tipo de ciclo de acesso (R/W*), o código de função do ciclo de instrução corrente e indicação sobre o que está sendo processado (instrução ou não).

Fase 4: Execução da rotina de serviço

- o endereço da rotina de serviço é carregado no PC; e

- a UCP executa sequencialmente as instruções a partir deste endereço até a instrução RTE, quando as informações de PC e SR são lidos da pilha do supervisor e restaurados.

Observações:

- MC68000 tem dois modos de operação: supervisor e usuário. O processamento de exceção sempre ocorre no modo de supervisor (Fig. 6.14 do livro-texto, p. 455).
- Na fase 1 o conteúdo do SR é copiado temporariamente em algum registrador de trabalho interno da UCP. Esta cópia é que é guardada na pilha durante a fase 3.
- A fase 2 do processamento de uma interrupção é diferente da fase 2 de outras exceções. Numa interrupção, a fase 2 inicia com a geração do sinal IACK* que pode ser derivado dos sinais AS*, FC0–FC2 e A_{01}–A_{03} (Figs. 6.7 e 6.9 do livro-texto, pp. 445 e 447).
  - Quando se trata de uma interrupação vetorizada, o periférico solicitante coloca no barramento o número do vetor e ativa o sinal DTACK* em resposta ao sinal IACK*. Este ciclo de leitura é conhecido como o ciclo de conhecimento de interrupção (**ciclo de IACK**). Ele ocorre exatamente entre a escrita da palavra menos significativa do PC e a escrita da palavra mais significativa do PC na pilha do supervisor (Figs. 6.5 e 6.6 do livro-texto, p. 443).
  - Quando se trata de uma interrupção auto-vetorizada, o sinal VPA* deve ser ativado externamente em resposta ao sinal IACK* para indicar que a própria UCP deve providenciar o número de vetor. MC68000 sempre responde ao sinal VPA* ativando o sinal VMA* e o ciclo de IACK é então sincronizado com o sinal VMA* (Fig. 6.8 do livro-texto, p. 446).
- Embora 68000 não consiga retornar de uma exceção do grupo 0, é possível “enganar” o processador gerando um novo conjunto de informação na pilha e tentar retornar ao ponto de interrupção a partir deste novo conjunto. Este procedimento não é recomendável. O mais seguro é utilizar processadores que consigam retornar deste grupo de exceção.

- No fluxograma da Fig. 6.18 do livro-texto (p. 460) o teste pelo erro no barramento BERR* aparece em vários pontos. Isso é porque BERR* é um sinal gerado externamente em resposta a uma falha no ciclo de barramento (p. ex., ler o número de vetor de exceção, escrever o conteúdo do PC e SR na pilha e buscar o endereço da rotina de serviço).
- A falha dupla no barramento (*double bus fault*) que aparece no fluxograma da Fig. 6.18 não corresponde exatamente a uma exceção e sim, à ocorrência sucessiva de duas exceções em um intervalo de tempo muito curto, insuficiente para que o tratamento da primeira exceção seja concluída. Neste caso, o sinal HALT* é ativado automaticamente.
- A sequência de processamento da exceção RESET é um pouco diferente da de outras exceções. Como já mencionamos, o processador precisa ter (1) o apontador da pilha do sistema para poder tratar uma exceção, e (2) o endereço inicial de execução a ser carregado no PC (Fig. 6.29 do livro-texto, p. 482). Observe que estes dois endereços devem ser mantidos nas partes não voláteis do Sistema de Memória.

Além das exceções já mencionadas, podem ocorrer ainda no MC68000 as seguintes exceções internas, algumas delas iniciadas de forma síncrona pelo programador (Fig. 6.19 do livro-texto, p. 461) e outras geradas automaticamente pelo processador:

erro no endereçamento: quando o endereço no barramento é inválido;

instrução ilegal: quando o código de operação da instrução não corresponde a nenhum padrão de bits no repertório de instruções válidas do MC68000;

divisão por zero: quando o valor do denominador de uma divisão é "zero";

validar o conteúdo do registrador: quando a instrução CHK é executada;

violação de privilégio: quando uma instrução privilegiada é executada no modo de usuário;

TRAP: quando a instrução TRAP é executada (esta instrução suporta até 16 distintos valores no seu campo de argumento);

TRAPV: quando a instrução TRAPV é executada;

Line 1010: quando uma instrução que contém 1010 nos seus bits mais significativos é executada;

Line 1111: quando uma instrução que contém 1111 nos seus bits mais significativos é executada;

Trace: quando o bit T do registrador de estado SR estiver com o valor 1, ou seja o processador estiver no modo de execução passo-a-passo. Esta exceção é gerada automaticamente após a execução de cada instrução.

Exercícios de Revisão

1. Resolver os exercícios 3, 4, 14, 21, 31, 35, 37, 40 e 44 do capítulo 6 do livro-texto.
2. Resolver os exercícios 19, 20 e 22 do capítulo 6 do livro-texto.

4.2 Implementação do Processamento de Exceções em MC68000

O tratamento correto de uma exceção em MC68000 requer a implementação de recursos complementares ao mecanismo de processamento de exceção provido pelo mesmo, a saber:

- instalação de **tabela de vetores** na Unidade de Memória;
- instalação de **rotinas de serviço** (programas) nos endereços especificados na tabela de vetores;
- implementação em circuitos as lógicas de ativação de sinais que correspondem aos eventos externos, RESET*, BERR* e interrupções IP0*–IP2*; e
- no caso de interrupções, implementação em circuitos a lógica de ativação do sinal IACK* requerido pelos periféricos.

Através da **tabela de vetores** (de 256 palavras longas ocupando os endereços de \$00 0100 até \$00 03FF) o processador determina o endereço da rotina de serviço de cada exceção. Portanto, a primeira idéia que pode aparecer é implementar esta tabela nas memória não-voláteis da Unidade de Memória, de forma que os seus dados sejam sempre preservados. Porém, surge um outro problema: como se altera o endereço de uma rotina de serviço? Os endereços das rotinas de serviço devem ser manipuláveis pelos projetistas de sistema/aplicativos, uma vez que os programas (*software*) para tratamento das exceções (*hardware*) são desenvolvidos e instalados por estes últimos. Como se resolve este dilema?

Na prática este dilema é resolvido com a distinção entre o vetor (0) da exceção RESET e os vetores de outras exceções. O conteúdo do vetor 0 é fundamental para iniciar um sistema, pois ele contém o apontador da pilha do supervisor (necessário para executar uma exceção) e o endereço da rotina (de serviço) que reinicializa (*reset*) o sistema. Em outras palavras, somente o conteúdo do vetor 0 precisa ficar armazenado fisicamente em uma memória não-volátil (ROM ou *battery-baked* RAM). Com isso, o nosso problema se reduz então à seguinte pergunta: qual lógica deve ser implementada para mapear parte do conteúdo de uma memória não-volátil e o conteúdo de uma memória volátil em um espaço de memória contíguo do processador?

Duas técnicas são muito utilizadas:

Sobreposição de memórias: os endereços \$00 0100–\$00 03FF são mapeados em uma memória volátil. Porém, uma lógica adicional remapeia os endereços \$00 0000–\$00 0007 (dos dados do vetor 0) para um outro conjunto de endereços nos quais uma memória não-volátil esteja mapeada. Esta idéia pode ser implementada de duas formas:

1. o decodificador de endereços, ao detectar os endereços \$00 0000–\$00 0007, seleciona a pastilha de memória não-volátil no lugar da memória volátil (Figs. 6.11 e 6.12. do livro-texto, pp. 451 e 452).
2. o decodificador de endereços utiliza o fato de que na inicialização as primeiras operações executadas pelo processador é ler o endereço da pilha do supervisor (dois ciclos de barramento) e ler o endereço do programa de "reset" (dois ciclo de barramento), ele sempre acessa a memória não-volátil (onde estão os dados do vetor 0) nos primeiros quatro ciclos de barramento. Depois disso, volta-se ao controle usual (Fig. 6.31 do livro-texto, p. 484).

Shadow ROM: ambos os blocos de memória volátil e não-volátil são mapeados no mesmo espaço de endereços \$00 0100–\$00 03FF. Durante a inicialização, a memória volátil é acessada no ciclo de escrita e a memória não-volátil no ciclo de leitura, até a completa instalação da tabela de vetores. A partir de então, somente a memória volátil é acessada (Fig. 6.32 do livro-texto, p. 486).

Uma vez instalada a tabela de vetores, é necessário instalar as rotinas de serviço para tratamento das exceções. Em computadores de uso genérico, muitas destas rotinas são inclusas no sistema operacional e são instaladas automaticamente durante a inicialização; e outras, como as rotinas de serviço para tratamento de interrupções, são comumente instaladas pelos programadores de aplicativos.

O interfaceamento dos seguintes pinos com o mundo externo para captar corretamente os eventos é fundamental nos sistemas baseados em MC68000:

RESET*: o sinal neste sinal é gerado propositalmente pelo usuário, ao ligar o sistema (computacional) a uma fonte de alimentação ou ao "resetar" manualmente o sistema. No caso de conexão à fonte, um circuito de atraso (na ordem de grandeza de 100ms) deve ser incluído para garantir uma correta operação dos componentes semicondutores; e no caso de "reset" manual um circuito *debouncer* deve ser incluído para eliminar pulsos espúrios de chaveamento (Fig. 6.30 do livro-texto, p. 484).

IP0*–IP2*: os sinais nestes 3 pinos definem o nível de prioridade do sinal solicitante de interrupção e é utilizado pelo processador para decidir se deve deferir ou indeferir a solicitação. No lado dos periféricos, entretanto, há somente um pino de saída para o sinal de solicitação de interrupção, IRQ*. Portanto, uma lógica adicional de codificação dos sinais IRQ* em níveis de prioridade é necessária para interfacear o pino IRQ* e os pinos IP0*–IP2* (Figs. 6.7 e 6.9 do livro-texto, pp. 445 e 447).

BERR*: A deteção de um erro no barramento é deixado a cargo do projetista do sistema. Para garantir que o sistema não fique travado em um ciclo de barramento, é interessante incluir uma lógica externa que ative o sinal BERR* depois de ter esgotado um certo tempo de espera pela ativação do sinal DTACK* ou do sinal VPA*. Usualmente, o circuito é um temporizador. Vale ressaltar aqui que a ativação do sinal BERR* no ciclo de leitura/escrita gera a exceção *bus error*, e no ciclo de IACK gera a exceção *spurious interrupt*.

Especificamente no caso de interrupções, os periféricos sempre aguardam a ativação do seu sinal IACK* para iniciar a fase 2 do seu processamento. Em relação a MC68000 os sinais IACK* podem ser derivados a partir dos sinais A_{01}–A_{03}, FC0–FC2 e AS*, conforme ilustram Figs. 6.7 e 6.9 do livro-texto, pp. 445 e 447.

Observações

- Mesmo que não utilize todos os vetores, é de boa prática deixar os endereços \$00 0000–\$00 03FF só para a tabela de vetores de exceção.
- Para evitar processamento de números de vetores inválidos, os registradores dos periféricos devem ser inicializados (normalmente, pela rotina que "reseta" o sistema) com o número \$0F (15) que corresponde à exceção *uninitialized interrupt vector*. Isso garante que mesmo que um programa tenha deixado de carregar o número de vetor no registrador de um periférico, ações totalmente inesperadas sejam evitadas.
- MC68000 provê facilidades para reexecutar um ciclo de barramento que falhou, considerando a possibilidade de que a falha ter sido resultado de algum ruído no barramento e de que na nova tentativa o ciclo completará com sucesso. Uma reexecução é determinada pela ativação simultânea dos sinais RESET* e HALT* como ilustra a Fig. 6.34 do livro-texto (p. 489). Note a restrição de que o sinal HALT* deve ser negado no mínimo um ciclo de relógio após a negação do BERR*. Fig. 6.35 do livro-texto (p. 490) apresenta o esquema de um circuito que implementa a geração destes sinais a partir dos sinais AS* e ERROR (sinal externo). O problema deste circuito é que se um erro for persistente, o processador tentará novas reexecuções sem sucesso. Alguma lógica adicional deve ser incluída para controlar o número de tentativas. Fig. 6.36 do livro-texto (p. 491) sugere uma lógica que permite apenas uma tentativa de reexecução.
- O nível de interrupção 7 é sensível a borda (de transição alto–baixo) e não a nível como os outros níveis de interrupção (1 a 6). Portanto, ele é não-mascarável, ou seja, não pode ser desabilitado por *software*.

Exercícios de Revisão

1. Resolver o exercício 15 do capítulo 6 do livro-texto.
2. Para utilizar todas as facilidades do mecanismo de interrupção oferecidas pelo MC68000, quais são os circuitos adicionais necessários para interfaceá-lo com os periféricos?
3. Explique a lógica do circuito na Fig. 6.7 do livro-texto (p. 445).
4. Explique a lógica do circuito na Fig. 6.9 do livro-texto (p. 447).
5. Altere o circuito da Fig. 6.36 do livro-texto de forma que sejam feitas três tentativas de reexecução no lugar de uma tentativa.

Informações adicionais:

- As UCPs da família 80X86 (Intel) oferecem somente dois pinos de entrada para interrupção: uma é mascarável (INTR) e a outra é não-mascarável (NMI). Para expandir a entrada de mascarável para 8 ou 16 entradas (com prioridades programáveis), faz-se uso de um controlador de interrupções, como 8259.
- As CPUs da família 80X86 (Intel) respondem a um sinal de interrupção com dois pulsos INTA* que são utilizados pelo 8259 para colocar no barramento de dados o vetor de interrupção.

4.3 Aplicações das Exceções

A capacidade de processamento de exceções dos processadores permite desenvolver sistemas operacionais mais sofisticados ou ferramentas de apoio ao desenvolvimento de *software/hardware*. Nesta seção veremos o papel das exceções em algumas importantes aplicações, algumas das quais são particulares do MC68000.

4.3.1 Depuração

Uma das ferramentas mais importante no apoio ao desenvolvimento de *software* é o depurador. Um depurador é, em essência, um programa que chaveia o modo do processador para o modo de execução passo-a-passo (*trace mode*) e que contém um conjunto de rotinas para analisar e modificar o conteúdo dos registradores e da Unidade de Memória durante a execução de um programa. Operando no modo *trace*, o processador gera automaticamente a exceção TRACE após a execução de cada instrução e desvia o fluxo de controle para uma das rotinas providas pelo depurador para executar a operação de depuração desejada (pp. 467–469 do livro-texto).

Observação

- Aqui fica claro o motivo pelo qual na fase 1 do processamento de uma exceção o valor 0 é atribuído ao bit T do SR. Isso evita que a exceção TRACE seja gerada após a execução de cada instrução da rotina de serviço.

4.3.2 Portabilidade dos Programas

Um dos fatores importantes a se considerar no desenvolvimento de um *software* é a sua portabilidade. Para isso, procura-se evitar o uso de instruções dependentes das características de *hardware* do sistema no desenvolvimento de um programa. As 16 exceções TRAP do MC68000 permitem o projetista do sistema encapsular essas características em rotinas de serviço e oferecer aos programadores uma interface para as funcionalidades do modo supervisor.

Observações

- Na disciplina EA870 foi utilizada a instrução TRAP #14 para acessar as 255 diferentes funções implementadas no monitor TUTOR.
- Pode-se considerar a exceção TRAP uma chamada ao sistema (operacional) – *system call*.

4.3.3 Emulação de Instruções

Com uso das exceções *LINE 1010* e *LINE 1111* o projetista do sistema pode implementar no MC68000 um repertório de instruções novas por *software*.

A característica comum das novas instruções é que os quatro bits mais significativos da sua primeira palavra seja 1010 – $AXXX (daí o nome LINE 1010) ou 1111 – $FXXX (daí o nome LINE 1111). Toda vez que o processador executa uma destas instruções a exceção LINE 1010 ou LINE 1111 é gerada. O que se deve fazer dentro da rotina de serviço é identificar a instrução através do conteúdo da pilha do supervisor (Fig. 6.21 do livro-texto, p. 466) e desviar o fluxo de controle para a subrotina (também pertencente à rotina de serviço) capaz de executar a função da instrução.

Observações

- As exceções *LINE 1010* e *LINE 1111* pertencem ao grupo 2 de exceções, ou seja, o processamento inicia na execução normal de uma instrução. Portanto, o valor do PC salvo na pilha do supervisor é o endereço da instrução corrente e não da próxima instrução. Para retornar à próxima instrução, como se espera, deve-se corrigir o conteúdo do PC salvo dentro da rotina de serviço antes da chamada da instrução RTE.
- A exceção *LINE 1111* foi projetada para suportar coprocessadores. Os processadores da família 68000 (68020, 68030 ou 68040) que provêm facilidades para se comunicar com coprocessadores, ao ler uma instrução que inicia com o padrão 1111, tentam primeiro acionar o coprocessador e só no caso este não responder, gera-se uma exceção.

4.3.4 Aplicativos Interativos

Os aplicativos gráficos interativos são aqueles nos quais os usuários podem entrar os dados em qualquer instante sem se preocupar com o processamento de natureza síncrona do computador. Hoje em dia estes aplicativos podem ser desenvolvidos em cima de um *software* conhecido como **sistema de gerência de janelas**, que tem funções análogas às de um sistema operacional. Ao invés de gerenciar os recursos computacionais de forma genérica, um sistema de gerência de janelas controla o uso de recursos gráficos como as áreas de exibição (janelas) no monitor, os eventos de interação gerados a partir do teclado, do *mouse* ou de outros periféricos.

Os eventos de interação são gerados assincronamente, de forma totalmente aleatória, pelos usuários. O que se pode determinar a priori é somente a ação a ser tomada na ocorrência de um evento específico. Este tipo de controle pode ser facilmente implementado com uso do mecanismo de interrupção. Os eventos podem ser mapeados em interrupções que solicitam atenção do processador e as ações a serem tomadas correspondem a rotinas de serviço. De fato, muitos sistemas de gerência de janelas abstraem isso com uso de dois conceitos: funções de **Callback** e **MainLoop**. Os programas aplicativos interativos são inicializados com a definição da função de *callback* (rotina de serviço) para cada evento antes de entrar num laço de espera (infinito) por eventos dos usuários (*MainLoop*).

4.3.5 Sistemas em Tempo Real e Multitarefas

Definição 4.1 Processamento em tempo real *é quando o sistema responde a um evento num intervalo tolerável pela aplicação.*

Definição 4.2 Sistema de multitarefas *é quando o sistema atende vários processos concomitantemente, atribuindo a cada um uma fatia de tempo.*

Sistemas em tempo real e multitarefas são intimamente relacionados, mas não são sinônimos. Sistemas em tempo real favorecem o tempo de resposta a um evento e sistemas de multitarefas procuram otimizar o uso de recursos.

O núcleo de um sistema em tempo real distingue três estados de tarefas: pronto (para execução), em execução e bloqueados (Fig. 6.24 do livro-texto, p. 472). As informações de cada tarefa são armazenadas no bloco de controle de tarefa (TCB).

Nos sistemas preemptivos existe um relógio de tempo real (RTC) que gera eventos de interrupção periodicamente para o sistema operacional chavear o atendimento das tarefas prontas (para execução) de acordo com uma estratégia de escalonamento (Fig. 6.26 do livro-texto, p. 474). Este chaveamento é também conhecido como **chaveamento de contexto**, pois o conteúdo dos registradores, das pilhas e do contador de programa da tarefa corrente é salvo no seu TCB e o da nova tarefa recuperado antes da execução da nova tarefa (Fig. 6.25 do livro-texto, p. 473).

Vale observar que os eventos gerados pelo RTC concorrem com os outros eventos a atenção da UCP. Uma forma uniforme de tratá-los é apresentada

nas páginas 474–481 do livro-texto, na qual, exceto o escalonador, todas as tarefas que recebem eventos passam para o estado pronto. O chaveamento do estado pronto para o estado execução é gerenciado exclusivamente pelo escalonador em resposta aos eventos gerados pelo RTC.

Em um sistema de multitarefas, a execução de uma instrução por uma tarefa não deve afetar o contexto de outras tarefas. Portanto, só o mecanismo de interrupção não é suficiente para implementá-lo corretamente, uma vez que existem instruções como RESET ou STOP que afetam todo o sistema. Para evitar que uma tarefa altere inadvertida ou propositalmente o controle do sistema, é útil distinguir duas classes de instruções: as **privilegiadas** e as **não-privilegiadas**. Por segurança, todas as instruções capazes de modificar o conteúdo do registrador de estado devem ser privilegiadas.

MC68000 é um processador que suporta a implementação de um sistema de multitarefas, pois ele suporta interrupções e distingue dois modos de operação: o **modo de supervisor** (privilegiado) e o **modo de usuário** (não-privilegiado). No modo de supervisor deve operar o sistema operacional que controla as múltiplas tarefas. Estas, por sua vez, são executadas no modo de usuário. Com isso, não só garante que os efeitos de cada instrução de uma tarefa fiquem restritos a ela como também permite que o sistema operacional tenha controle sobre todas as tarefas.

Exercícios de Revisão

- Resolver os exercícios 6, 17, 23, 24 e 28 do capítulo 6 do livro-texto.

4.3.6 Unidade de Gerência de Memória

Definição 4.3 Unidade de Gerência de Memória *(MMU) é essencialmente um circuito que mapeia/traduz os endereços lógicos gerados pela UCP em endereços (globais ou físicos) da Unidade de Memória (Figs. 7.8 e 7.10 do livro-texto, pp. 532 e 533).*

Observações

- Espaço de endereços lógicos de um processador está relacionado com a capacidade deste em representá-los (quantidade de bits de endereçamento). No caso de MC68000, este espaço vai de 0 até 2^{24}-1.
- Espaço de endereços físicos de um computador está relacionado com a capacidade de armazenamento da sua Unidade de Memória, que também vai de 0 até o limite máximo.

As principais funções de uma Unidade de Gerência de Memória são:

1. traduzir os endereços lógicos em endereços físicos de forma transparente.
2. proteger a memória contra acessos indevidos.
3. permitir diferentes processos compartilhar recursos (instruções e dados).
4. facilitar a implementação de memória virtual, provendo apropriados sinais na ocorrência de falta do dado endereçado na Unidade de Memória.

Observações

- Basicamente existem duas soluções para contornar o problema de executar um programa cujo tamanho exceda o do espaço de memória do processador:
  1. Sobreposição (*overlays*): programas são divididos em pedaços modulares e armazenados em distintos **bancos de memória** selecionáveis. Fica a cargo dos **programadores** selecionar o banco correto através das instruções para garantir a correta execução de um programa (Fig. 7.11 do livro-texto, p. 534).

     – Mapeamento direto: o chaveamento entre os bancos de memória é feito diretamente pela especificação do banco de memória requerido. Neste caso, precisa-se desviar para o banco fixo, onde se encontra a instrução com o endereço de desvio, antes de chavear e desviar para o novo banco (Fig. 7.12 do livro-texto, p. 534).
     – Mapeamento indexado: A identificação de um banco a ser acessado está implícita no endereço. A seleção de um dado banco atavés do endereço lógico é feito automaticamente pelo *hardware* através de uma tabela de mapeamento. O conteúdo desta tabela (ou seja, a função de mapeamento) é definido pelo *software* (sistema operacional ou monitor). Neste caso, o desvio de um banco ao outro pode ser especificado através de um modo de endereçamento (Fig. 7.13 do livro-texto, p. 536).
  2. Memória Virtual: é um artifício para superar a capacidade de armazenamento da Unidade de Memória. Programas são divididos em pedaços modulares e armazenados em memória secundária (discos). Somente a parte em execução é garantida pelo sistema que esteja na Unidade de Memória. Com isso, consegue-se simular um grande espaço de memória (virtual) com uso de poucas memórias físicas.
     – Segmentada: o espaço de endereçamento é dividido em vários subespaços de tamanhos não necessariamente iguais e totalmente independentes entre si, denominados **segmentos**. A cada segmento é atribuída uma sequência linear de endereços lógicos a partir de 0 (Fig. 7.15 do livro-texto, p. 539).
     – Paginada: existe somente uma sequência linear de endereços lógicos dividida em blocos de mesmo tamanho, denominados **páginas** (Fig. 7.14 do livro-texto, p. 537). Quando a sequência é muito grande, é usual ainda utilizar a técnica de multinível para determinar mais eficientemente um endereço físico (Fig. 7.16 do livro-texto, p. 540).
- Comparações entre paginação e segmentação de uma memória virtual:

Considerações	Paginação	Segmentação
O programador precisa estar consciente da técnica utilizada?	Não	Sim
Quantos subespaços de memória	1	vários
Endereços virutais podem exceder endereços físicos?	Sim	Sim
Os dados e as instruções podem estar em espaços distintos?	Não	Sim
Tabelas de tamanhos distintos podem ser facilmente acomodadas?	Não	Sim
Compartilhamento de dados/instruções é facilitado?	Não	Sim
Finalidade	Estender o espaço linear de endereços	Permitir armazenar dados e programas de distintos processos em espaços lógicos separados, facilitando o mecanismo de compartilhamento e proteção.

Preparo para Próxima Aula

- Leia atentamente as páginas 542–551 do livro-texto e descreva sucintamente como a MMU 68451 mapeia um endereço lógico do processador MC68000 em um endereço físico.

- Responda as seguintes perguntas:

  1. Qual á a função dos códigos de função FC0–FC2 na tradução de endereços?
  2. Como se pode proteger uma área específica de memória?
  3. Quem é responsável pelo conteúdo de AST (*address space table*)?
  4. Como se pode compartilhar o acesso de uma área específica de memória?

  5. Como a MMU68451 pode ser utilizada no projeto de uma memória virtual?

Apesar das inúmeras facilidades providas pela MMU, estamos interessados neste capítulo em

- mostrar como se pode projetar um sistema de memória virtual com uso das facilidades providas por ela, do mecanismo de exceção de um processador (*hardware*) e de um programa de troca de páginas (*software*); e

- mostrar como se pode com uso dela e das facilidades de um processador controlar os acessos à Unidade de Memória.

Observação

- Um algoritmo de **troca de página** substitui um bloco residente na Unidade de Memória pelo bloco faltante durante um acesso à memória. As estratégias de substituição mais conhecidas são: troca ótima, *not-recently-used page*, FIFO, *least recently used*, *least frequently used*).

Uma MMU não só consegue detectar a falta de um dado endereçado na Unidade de Memória como também provê um pino para indicar esta falta. Este sinal pode ser utilizado pelo processador para gerar uma exceção e chamar um algoritmo de troca de página (rotina de serviço) para carregar o bloco faltante da memória secundária para a memória principal e retornar ao ponto interrompido, que é o acesso ao dado endereçado. No caso dos processadores da família 68000 o sinal de falta de página (ou erro no barramento) deve entrar pelo pino BERR*.

Observação

- Nesta altura é fácil inferir que o processador MC68000 não suporta a implementação de um sistema de memória virtual, uma vez que ele não consegue retornar apropriadamente de uma exceção BERR. Só os processadores de 68010 em diante conseguem tratar melhor esta exceção.

A unidade de gerência de memória MC68451 suporta segmentos de 256 bytes até 2Mbytes e contém 32 descritores de 72 bits, cada qual com as seguintes informações referentes a um segmento (Fig. 7.17 do livro-texto, p. 543):

- Endereço-base lógico (LBA, 16 bits): início de um segmento lógico;
- Endereço-base físico (PBA, 16 bits): início de um segmento físico;
- Máscara de endereço lógico (LAM, 16 bits): define indiretamete o tamanho de cada segmento, indicando os bits em LBA/PBA que são significativos para identificar um segmento.
- Número do espaço de endereço (ASN, 8 bits);
- Máscara do espaço de endereço (ASM, 8 bits);
- Registrador do espaço do segmento (SSR, 8 bits):
  - U (used): indica se o segmento tem sido usado desde a sua definição.
  - I (interrupt): quando igual a 1 força interrupção toda vez que o segmento for acessado (útil para depuração).
  - IP (interrupt pending): só é setado quando I=1 e o segmento é acessado.
  - M (modified): indica se o segmento foi modificado ou não.
  - WP (write-protect): protege o segmento da escrita.
  - E (enable): habilita o segmento para acesso.

Nas páginas 542–549 é explicado detalhadamente o mapeamento dos endereços lógicos em endereços físicos com uso da MMU 68451. Observe que os códigos de função FC0–FC2 podem ser utilizados para gerenciar os acessos à Unidade de Memória (Fig. 7.20 do livro-texto, p. 547).

Funcionalmente ele dispõe os seguintes sinais:

- sinais de endereços lógicos: 16 pinos de endereço e 4 pinos de códigos de função;
- sinais de endereço físicos/dados multiplexados: 23 pinos;

- sinais de controle de barramento: *strobes* da UCP (3 pinos), DTACK*, FAULT* e IRQ*;
- sinais de controle da pastilha: RESET*, CS*;
- sinais de controle da Unidade de Memória: HADD* (saída dos endereços), ED* (saída dos dados), strobe de endereços para Unidade de Memória (MAS*); e
- outros.

Fig. 7.21 do livro-texto (p. 550) apresenta um esquema de conexão de uma Unidade de Memória com o processador MC68000 através da MMU 68451.

4.3.7 Coprocessadores

Definição 4.4 Coprocessadores *são unidades de processamento especiais que assistem a uma unidade central de processamento a executar certas operações, estendendo as suas funcionalidades.*

Preparo para Próxima Aula

- Leia atentamente as páginas 597–603 do livro texto e descreva sucintamente a arquitetura do coprocessador de ponto-flutuante MC68882.

Sob o ponto de vista dos programadores, os coprocessadores estendem o repertório de instruções de um processador. Porisso, eles só precisam tomar conhecimento do conjunto de novas instruções e de novos registradores providos pelos mesmos. Os projetistas do sistema precisam, entretanto, saber a organização dos coprocessadores para conectá-los fisicamente a um processador e estabelecer a comunicação entre eles. Veremos aqui como o mecanismo de exceção pode ser útil na implementação desta comunicação em algumas situações.

Há várias formas para os coprocessadores interagirem com os processadores hospedeiros:

- os coprocessadores tem os seus próprios decodificadores de instrução e contador de programa,
- os coprocessadores se comunicam com os processadores através de barramentos dedicados, e
- mapear o coprocessador no espaço de endereços do microprocessador. Neste caso, o coprocessador é tratado como um periférico com o qual o processador comunica assincronamente através de um protocolo pré-definido.

Os projetistas dos coprocessadores da série 68000 optaram por esta última alternativa, na qual os coprocessadores nunca podem ser donos do barramento. Os acessos à Unidade de Memória são sempre feitos via UCP.

Como já mencionamos, os processadores da família 68000 reservam instruções que iniciam com padrão 1111 para os coprocessadores. Estas instruções contém quatro campos (Fig. 7.59 do livro-texto, p. 595):

1. 4 bits para indicar que é uma instrução do coprocessador: 1111;
2. 3 bits para indicar o tipo de coprocessador;
3. 3 bits para indicar uma das 8 possíveis classes de instruções (p. 595); e
4. 5 bits cujo conteúdo depende da classe da instrução.

Além disso, 68000 reserva um espaço de memória específico, denominado **espaço da UCP** ((FC2,FC1,FC0) assumem (1,1,1) nos acessos a este espaço), para mapear até 8 coprocessadores. A cada coprocessador é alocado um espaço contíguo de 32 bytes (Fig. 7.55 do livro-texto, p. 592). O endereço de um registrador do coprocessador é representado por uma palavra de 32 bits divididos em 5 campos (Fig. 7.55 do livro-texto, p. 591):

1. A_{31}–A_{20} e A_{12}–A_{05} iguais a 0;
2. A_{19}–A_{16} = (0,0,1,0) para indicar um acesso a coprocessador;
3. A_{15}–A_{13} para indicar o tipo de processador; e
4. A_{04}–A_{00} para identificar o registrador do processador.

Os coprocessadores da série 68000 utilizam a interface do barramento assíncrono para se comunicar com os processadores da mesma série (Fig. 7.57 do livro-texto, p. 594). A comunicação se faz através dos registradores mapeados no espaço de memória da UCP.

Quando um processador detecta uma instrução do tipo *LINE 1111*, ele a traduz em uma instrução do coprocessador e a transfere para o registrador de comando do coprocessador, fazendo um acesso ao espaço da UCP. Através de uma lógica externa o sinal CS* do coprocessador é ativado e inicia-se a transferência cujo controle é feito através da passagem de mensagens por meio dos registradores. Nos processadores 68020, 68030 ou 68040 o protocolo de transferências é gerenciado automaticamente por um *firmware* interno. Os processadores 68000, no entanto, não dispõem este protocolo. Uma forma de contornar esta deficiência é emulá-lo com uso da exceção *LINE 1111* e a sequência de transações ocorrerá dentro da rotina de serviço correspondente. A implementação desta rotina requer o conhecimento da organização interna dos registradores nos processadores (Fig. 7.60 do livro-texto, p. 596).

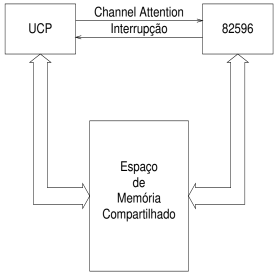
Exercícios de Revisão

1. Resolver os exercícios 13, 14, 15, 17, 18, 20, 22, 37 e 38 do capítulo 7 do livro-texto.

Informações adicionais:

- Os coprocessadores matemáticos Intel387^{TM}DX também utilizam os ciclos de barramento para se comunicar com os processadores Intel386^{TM}. A UCP automaticamente ativa o coprocessador quando uma das suas instruções for detectada (instruções ESC). Todos os acessos à memória são executados pela UCP e o coprocessador provê uma lógica de controle (*hardware*) para transferências de operandos e resultados entre ele e a UCP.
- Os coprocessadores Intel82596DX/SX realizam todas as funções de controle de acesso a uma rede local e são compatíveis com os processadores

de 32 bits. Diferentemente dos coprocessadores matemáticos, estes podem ser mestres de barramento e buscam sequencialmente as suas instruções num espaço de memória compartilhado (com a UCP) quando o sinal CA (*channel attention*) for ativado pela UCP. Quando o conteúdo do espaço compartilhado for modificado, o coprocessador utiliza o sinal INT/INT* para alertar a UCP.



4.4 Auto-avaliação

Após este capítulo, você deve ser capaz de:

- diferenciar uma exceção de uma interrupção.
- descrever sucintamente o processamento de uma exceção em 68000.
- explicar o que é uma interrupção vetorizada e uma interrupção auto-vetorizada.
- descrever o ciclo de IACK de uma interrupção vetorizada.
- descrever o ciclo de IACK de uma interrupção auto-vetorizada.
- explicar quais recursos adicionais devem ser providos pelo projetista do sistema para que o processador 68000 trate corretamente os eventos externos.
- listar as aplicações das exceções suportadas pelo MC68000.

- explicar como o mecanismo de exceção pode suportar a implementação de um sistema operacional em tempo real (multitarefas).
- listar as funções de uma unidade de gerência de memória.
- explicar como o mecanismo de exceção pode suportar a implementação de um sistema de memória virtual.
- explicar o papel de um coprocessador em um sistema computacional.
- explicar como o mecanismo de exceção pode suportar a implementação do protocolo de comunicação entre um processador e um coprocessador.

4.5 Lista de Exercícios

1. Resolver o exercício 10 do capítulo 6 do livro-texto.
2. Resolver o exercício 26 do capítulo 6 do livro-texto.
3. Resolver o exercício 19 do capítulo 7 do livro-texto.