7

ENTRADA/SAÍDA

7.1 Dispositivos externos
Teclado/monitor
Drive de disco

7.2 Módulos de E/S
Função do módulo
Estrutura do módulo de E/S

7.3 E/S programada
Visão geral da E/S programada
Comando de E/S
Instruções de E/S

7.4 E/S controlada por interrupção
Processamento de interrupção
Aspectos de projeto
Controlador de interrupção Intel 82C59A
A interface de periférico programável Intel 8255A

7.5 Acesso direto à memória
Desvantagens da E/S programada e controlada por interrupção
Função do DMA
Controlador de DMA Intel 8237A

7.6 Acesso direto à cache
DMA usando cache compartilhada de último nível
Aspectos do desempenho relacionado à cache
Estratégias de acesso direto à cache
E/S de Dados Diretos

7.7 Processadores e canais de E/S
A evolução da função de E/S
Características dos canais de E/S

7.8 Padrões de interconexão externa
Barramento serial universal (USB)
Barramento serial FireWire
Small Computer System Interface (SCSI)
Thunderbolt
InfiniBand
PCI Express
SATA
Ethernet
Wi-Fi

7.9 Estrutura de E/S do zEnterprise EC12 da IBM
Estrutura do canal
Organização de sistema de E/S

7.10 Termos-chave, questões de revisão e problemas

OBJETIVOS DE APRENDIZAGEM

Após ler este capítulo, você será capaz de:

- Explicar o uso dos módulos de E/S como parte da organização de computador.
- Entender a diferença entre E/S programada e E/S controlada por interrupção e discutir suas vantagens relativas.
- Apresentar uma visão geral da operação do acesso direto à memória.
- Apresentar uma visão geral do acesso direto à cache.
- Explicar a função e o uso dos canais de E/S.

Além do processador e um conjunto de módulos de memória, o terceiro elemento-chave de um sistema de computação é um conjunto de módulos de E/S. Cada módulo tem interface com o barramento do sistema ou com o comutador central e controla um ou mais dispositivos periféricos. Um módulo de E/S não é simplesmente um conjunto de conectores mecânicos que liga um dispositivo fisicamente ao barramento do sistema. Em vez disso, o módulo de E/S contém uma lógica para realizar a função de comunicação entre o periférico e o barramento.

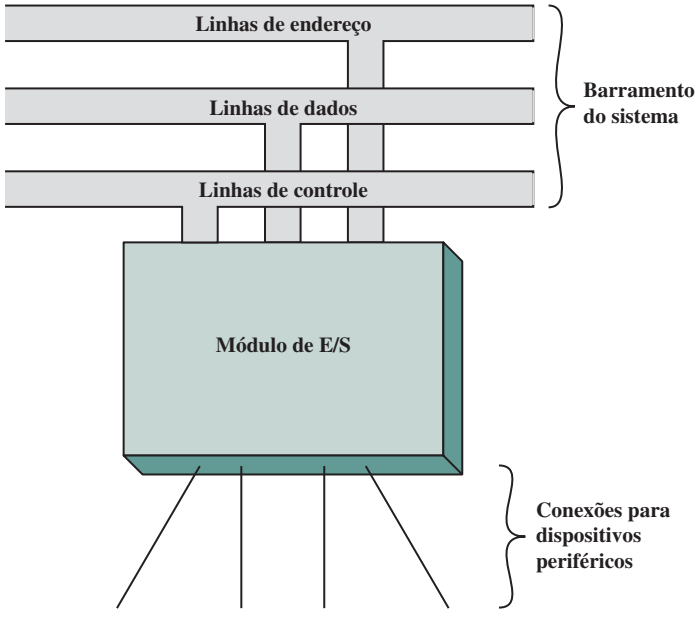
O leitor poderá se perguntar por que os periféricos não são conectados diretamente no barramento do sistema. As razões são as seguintes:

- Existe uma grande variedade de periféricos, com diversos métodos de operação. Seria impraticável incorporar a lógica necessária dentro do processador para controlar todos os tipos de dispositivos.
- A taxa de transferência de dados dos periféricos frequentemente é muito mais lenta do que a da memória ou do processador. Desse modo, é impraticável usar o barramento de alta velocidade do sistema para se comunicar diretamente com um periférico.
- Contudo, a taxa de transferência de dados de alguns periféricos é maior do que a da memória ou do processador. Novamente, uma diferença levaria a ineficiências se não fosse controlada corretamente.
- Os periféricos muitas vezes utilizam formatos de dados e extensões de palavras diferentes do que é usado pelo computador ao qual estão conectados.
- Assim, um módulo de E/S é necessário. Esse módulo tem duas funções principais (Figura 7.1):
  - Interface com o processador e a memória via barramento do sistema ou comutador central.
  - Interface com um ou mais dispositivos periféricos por conexões de dados adequadas.

Vamos começar este capítulo com uma breve discussão sobre os dispositivos externos, seguida por uma visão geral da estrutura e função de um módulo de E/S. Depois, veremos as diversas maneiras como a função de E/S pode ser realizada em cooperação com o processador e a memória: a interface de E/S interna. Em seguida, examinaremos em alguns detalhes o acesso direto à memória e as mais recentes inovações do acesso direto à cache. Por fim, examinaremos a interface de E/S externa, entre o módulo de E/S e o mundo exterior.

Figura 7.1

Modelo genérico de um módulo de E/S.



7.1 DISPOSITIVOS EXTERNOS

As operações de E/S são realizadas por meio de uma grande variedade de dispositivos externos, que oferecem um meio de trocar dados entre o ambiente externo e o computador. Um dispositivo externo conecta-se ao computador por uma conexão com um módulo de E/S (Figura 7.1). A conexão é usada para trocar sinais de controle, estado e dados entre os módulos de E/S e o dispositivo externo. Um dispositivo externo conectado a um módulo de E/S costuma ser chamado de *dispositivo periférico* ou, simplesmente, de *periférico*.

Podemos classificar os dispositivos externos em geral em três categorias:

- **Inteligíveis ao ser humano**: adequados para a comunicação com usuários de computador.
- **Inteligíveis à máquina**: adequados para a comunicação com equipamentos.
- **Comunicação**: adequados para a comunicação com dispositivos remotos.

Alguns exemplos de dispositivos inteligíveis ao ser humano são monitores de vídeo e impressoras. Alguns exemplos de dispositivos inteligíveis à máquina são sistemas de disco e de fita magnética, e sensores e atuadores, como aqueles usados em uma aplicação de robótica. Observe que estamos vendo os sistemas de disco e fita como dispositivos de E/S neste capítulo, enquanto, no Capítulo 6, eles são vistos como dispositivos de memória. Do ponto de vista funcional, esses dispositivos fazem parte da hierarquia de memória, e seu uso é discutido apropriadamente no Capítulo 6. Do ponto de vista estrutural, esses dispositivos são controlados por módulos de E/S e, por isso, devem ser considerados neste capítulo.

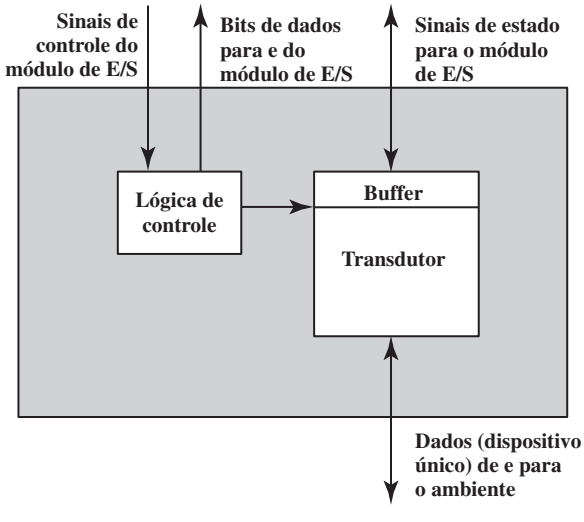
Dispositivos de comunicação permitem que um computador troque dados com um dispositivo remoto, que pode ser um dispositivo inteligível ao ser humano, como um terminal, um dispositivo inteligível à máquina ou até mesmo outro computador.

Em termos muito gerais, a natureza de um dispositivo externo é indicada na Figura 7.2. A interface com o módulo de E/S ocorre na forma de sinais de controle, de dados e de estado. Os *sinais de controle* determinam a função que o dispositivo realizará, como enviar dados ao módulo de E/S (INPUT ou READ), aceitar dados do módulo de E/S (OUTPUT ou WRITE), informar o estado ou realizar alguma função de controle particular ao dispositivo (por exemplo, posicionar uma cabeça de disco). Os *dados* estão na forma de um conjunto de bits a serem enviados ou recebidos do módulo de E/S. Os *sinais de estado* indicam o estado do dispositivo. Alguns exemplos são READY/NOT-READY, para indicar se o dispositivo está pronto para uma transferência de dados.

A *lógica de controle*, associada ao dispositivo, controla a operação do dispositivo em resposta ao módulo de E/S. O *transdutor* converte dados elétricos para outras formas de energia durante a saída e de outras formas para elétrico durante a entrada. Em geral, um buffer é associado ao transdutor para manter temporariamente os dados sendo transferidos entre o módulo de E/S e o ambiente externo. Um tamanho de buffer de 8 a 16 bits

Figura 7.2

Diagrama em blocos de um dispositivo externo.



é comum para dispositivos seriais, enquanto os dispositivos orientados por blocos, como controladores de drives de disco, podem ter buffers muito maiores.

A interface entre o módulo de E/S e o dispositivo externo será examinada na Seção 7.7. A interface entre o dispositivo externo e o ambiente está fora do escopo deste livro, mas vamos mostrar alguns exemplos rapidamente.

Teclado/monitor

O meio mais comum de interação entre computador/usuário é o conjunto teclado/monitor. O usuário fornece entrada pelo teclado. Essa entrada é, então, transmitida ao computador e também pode ser exibida no monitor. Além disso, o monitor exibe dados fornecidos pelo computador.

A unidade de troca básica é o caractere. Associado a cada caractere existe um código, em geral com tamanho de 7 ou 8 bits. O código de texto mais utilizado é o International Reference Alphabet (IRA).[1] Cada caractere nesse código é representado por um código binário exclusivo com 7 bits; desse modo, 128 caracteres diferentes podem ser representados. Os caracteres são de dois tipos: imprimíveis e de controle. Os caracteres imprimíveis são os caracteres alfabéticos, numéricos e especiais, que podem ser impressos em papel ou exibidos em um monitor. Alguns dos caracteres de controle têm a ver com o controle da impressão ou exibição de caracteres; um exemplo é o *carriage return*. Outros caracteres de controle tratam dos procedimentos de comunicação. Veja mais detalhes no Apêndice H (disponível em inglês na Sala Virtual).

Para a entrada do teclado, quando o usuário pressiona uma tecla, isso gera um sinal eletrônico que é interpretado pelo transdutor no teclado e traduzido para o padrão de bits do código IRA correspondente. Esse padrão de bits é, então, transmitido ao módulo de E/S no computador, onde o texto pode ser armazenado no mesmo código IRA. Na saída, os caracteres do código IRA são transmitidos para um dispositivo externo do módulo de E/S. O transdutor no dispositivo interpreta esse código e envia os sinais eletrônicos exigidos ao dispositivo de saída, ou para exibir o caractere indicado, ou para realizar a função de controle solicitada.

Drive de disco

Uma unidade de disco contém a eletrônica para trocar sinais de dados, controle e estado com um módulo de E/S mais a eletrônica para controlar os mecanismos de leitura/gravação de disco. Em um disco de cabeça fixa, o transdutor é capaz de converter os padrões magnéticos na superfície do disco móvel em bits no buffer do dispositivo (Figura 7.2). Um disco com cabeça móvel também deve ser capaz de fazer o braço do disco se mover radialmente para dentro e fora pela superfície do disco.

7.2 MÓDULOS DE E/S

Função do módulo

As principais funções ou requisitos para um módulo de E/S encontram-se nas seguintes categorias:

- Controle e temporização.
- Comunicação com o processador.
- Comunicação com o dispositivo.
- Buffering de dados.
- Detecção de erro.

Durante qualquer período, o processador pode comunicar-se com um ou mais dispositivos externos em padrões imprevisíveis, dependendo da necessidade de E/S do programa. Os recursos internos, como a memória principal e o barramento do sistema, precisam ser compartilhados entre uma série de atividades, incluindo E/S de dados. Assim, a função de E/S inclui um requisito de **controle e temporização** para coordenar o fluxo de tráfego entre os recursos internos e dispositivos externos. Por exemplo, o controle da transferência de dados de um dispositivo externo ao processador poderia envolver a seguinte sequência de etapas:

1 IRA é definido na ITU-T Recommendation T.50, e era conhecido originalmente como International Alphabet Number 5 (IA5). A versão do IRA para os Estados Unidos é conhecida como American Standard Code for Information Interchange (ASCII).

1. O processador interroga o módulo de E/S para verificar o estado do dispositivo conectado.
2. O módulo de E/S retorna o estado do dispositivo.
3. Se o dispositivo estiver operacional e pronto para transmitir, o processador solicita a transferência de dados por meio de um comando ao módulo de E/S.
4. O módulo de E/S obtém uma unidade de dados (por exemplo, 8 ou 16 bits) do dispositivo externo.
5. Os dados são transferidos do módulo de E/S ao processador.

Se o sistema emprega um barramento, então cada uma das interações entre o processador e o módulo de E/S envolve uma ou mais arbitrações de barramento.

Esse cenário simplificado também ilustra que o módulo de E/S precisa se comunicar com o processador e com o dispositivo externo. A **comunicação do processador** envolve o seguinte:

- **Decodificação de comando**: o módulo de E/S aceita comandos do processador, em geral enviados como sinais no barramento de controle. Por exemplo, um módulo de E/S para uma unidade de disco deve aceitar os seguintes comandos: READ SECTOR, WRITE SECTOR, SEEK número de trilha e SCAN ID de registro. Cada um dos dois últimos comandos inclui um parâmetro que é enviado no barramento de dados.
- **Dados**: os dados são trocados entre o processador e o módulo de E/S pelo barramento de dados.
- **Informação de estado**: como os periféricos são muito lentos, é importante conhecer o estado do módulo de E/S. Por exemplo, se um módulo de E/S tiver de enviar dados ao processador (leitura), ele pode não ser capaz de fazer isso porque ainda está trabalhando no comando de E/S anterior. Esse fato pode ser informado por meio de um sinal de estado, sendo os mais comuns BUSY e READY. Também pode haver sinais para informar diversas condições de erro.
- **Reconhecimento de endereço**: assim como cada palavra de memória tem um endereço, cada dispositivo de E/S também tem. Desse modo, um módulo de E/S deve reconhecer um endereço exclusivo para cada periférico que controla.

Por outro lado, o módulo de E/S também deve ser capaz de realizar **comunicação com o dispositivo**. Essa comunicação envolve comandos, informação de estado e dados (Figura 7.2).

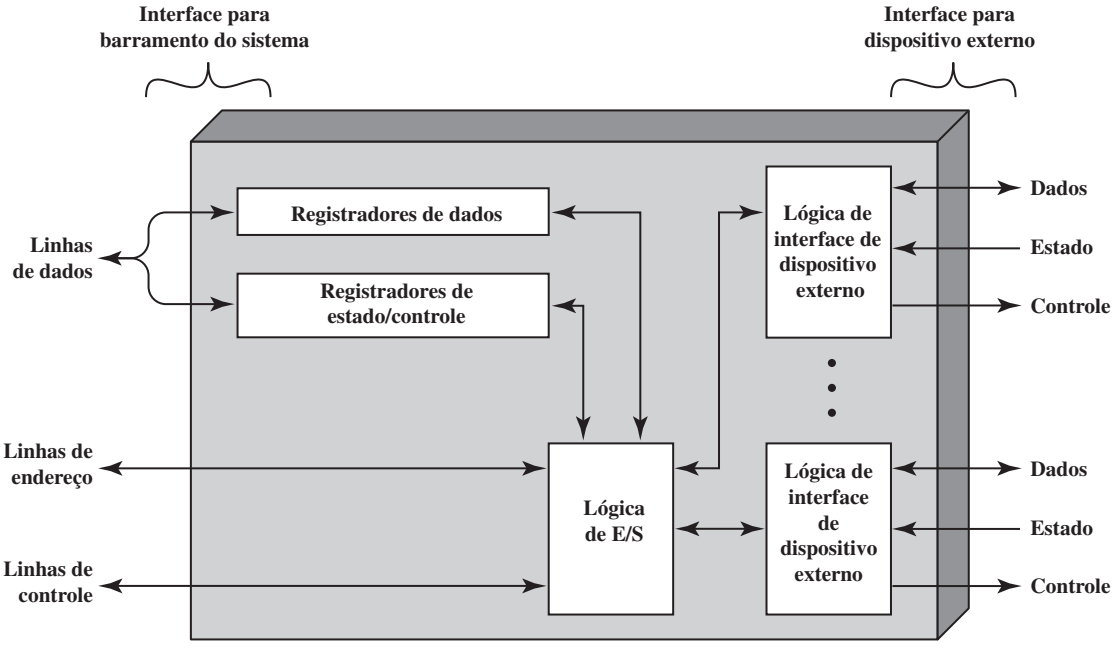
Uma tarefa essencial de um módulo de E/S é o **buffering de dados**. A necessidade dessa função fica evidente na Figura 2.1. Enquanto a taxa de transferência para entrada e saída na memória principal ou no processador é muito alta, as taxas da maioria dos dispositivos periféricos compreendem uma grande faixa. Os dados vindos da memória principal são enviados para um módulo de E/S em uma rajada rápida. Os dados são mantidos em um buffer no módulo de E/S e depois enviados ao dispositivo periférico em sua taxa de dados. Na direção oposta, os dados são mantidos em buffer para não reterem a memória com uma operação de transferência lenta. Desse modo, o módulo de E/S deve ser capaz de operar em ambas as velocidades do dispositivo e da memória. De modo semelhante, se o dispositivo de E/S opera em uma taxa mais alta que a taxa de acesso à memória, então o módulo de E/S realiza a operação de buffering necessária.

Por fim, um módulo de E/S em geral é responsável pela **detecção de erro** e, subsequentemente, por relatar erros ao processador. Uma classe de erros inclui defeitos mecânicos e elétricos relatados pelo dispositivo (por exemplo, papel emperrado, trilha de disco com defeito). Outra classe consiste em mudanças não intencionais no padrão de bits quando são transmitidos do dispositivo ao módulo de E/S. Alguma forma de código de detecção de erro normalmente é usada para detectar erros de transmissão. Um exemplo simples é o uso de um bit de paridade em cada caractere de dados. Por exemplo, o código de caracteres IRA ocupa 7 bits de um byte. O oitavo bit é definido a fim de que o número total de 1s no byte seja par (paridade par) ou ímpar (paridade ímpar). Quando um byte é recebido, o módulo de E/S verifica a paridade para determinar se ocorreu um erro.

Estrutura do módulo de E/S

Os módulos de E/S variam de modo considerável em complexidade e em número de dispositivos externos controlados por eles. Aqui, tentaremos apenas dar uma descrição. (Um dispositivo específico, o Intel 8255A, é descrito na Seção 7.4.) A Figura 7.3 oferece um diagrama de blocos geral de um módulo de E/S. O módulo conecta-se ao restante do computador por meio de um conjunto de linhas de sinal (por exemplo, linhas de barramento do sistema). Os dados transferidos de e para o módulo são armazenados em um ou mais registradores de dados. Também pode haver um ou mais registradores de estado que oferecem informações do estado atual. Um registrador de estado também pode funcionar como um registrador de controle, para aceitar informações

Figura 7.3
Diagrama do bloco de um módulo de E/S.



de controle detalhadas do processador. A lógica dentro do módulo interage com o processador por meio de um conjunto de linhas de controle. O processador usa as linhas de controle para enviar comandos ao módulo de E/S. Algumas das linhas de controle podem ser usadas pelo módulo de E/S (por exemplo, para sinais de arbitração e de estado). O módulo também precisa ser capaz de reconhecer e gerar endereços associados aos dispositivos que ele controla. Cada módulo de E/S tem um endereço exclusivo ou, se controlar mais de um dispositivo externo, um conjunto exclusivo de endereços. Por fim, o módulo de E/S contém uma lógica específica à interface com cada dispositivo que ele controla.

Um módulo de E/S funciona para permitir que o processador veja uma grande variedade de dispositivos de uma maneira simples. Existe um espectro de capacidades que podem ser oferecidas. O módulo de E/S pode ocultar os detalhes de temporização, formatos e eletromecânica de um dispositivo externo, de modo que o processador pode funcionar em termos de comandos simples de leitura e escrita, e possivelmente comandos para abrir e fechar arquivo. Em sua forma mais simples, o módulo de E/S ainda pode ter grande parte do trabalho de controlar um dispositivo (por exemplo, rebobinar uma fita) visível ao processador.

Um módulo de E/S, que assume a maior parte do processamento, apresentando uma interface de alto nível ao processador, é geralmente conhecido como canal de E/S ou *processador de E/S*. Um módulo de E/S que é muito primitivo e requer controle específico, normalmente é conhecido como *controlador de E/S* ou *controlador de dispositivo*. Os controladores de E/S em geral são vistos nos microcomputadores, enquanto os canais de E/S são usados em mainframes.

A seguir, usaremos o termo genérico *módulo de E/S* quando não houver confusão, e usaremos termos mais específicos onde for necessário.

7.3 E/S PROGRAMADA

Três técnicas são possíveis para operações de E/S. Com a *E/S programada*, os dados são trocados entre o processador e o módulo de E/S. O processador executa um programa que lhe dá controle direto da operação de E/S, incluindo verificação do estado de dispositivo, envio de um comando de leitura ou escrita e transferência dos dados. Quando o processador emite um comando ao módulo de E/S, ele deve esperar até que a operação de E/S termine. Se o processador for mais rápido que o módulo de E/S, isso desperdiça o tempo do processador. Com a E/S controlada por interrupção, o processador emite um *comando de E/S*, continua a executar outras

instruções e é interrompido pelo módulo de E/S quando o último tiver completado seu trabalho. Com a E/S programada e por *interrupção*, o processador é responsável por obter dados da memória principal para saída e por armazenar dados na memória principal da entrada. A alternativa a estes modos é conhecida como **acesso direto à memória (DMA)**. Nesse modo, o módulo de E/S e a memória principal trocam dados diretamente, sem envolvimento do processador.

A Tabela 7.1 indica a relação entre essas três técnicas. Nesta seção, exploramos a E/S programada. A E/S por interrupção e DMA são exploradas nas duas seções seguintes, respectivamente.

Tabela 7.1

Técnicas de E/S.

	Sem interrupções	Uso de interrupções
Transferência de E/S para memória via processador	E/S programada	E/S controlada por interrupção
Transferência direta de E/S para memória		Acesso direto à memória (DMA)

Visão geral da E/S programada

Quando o processador está executando um programa e encontra uma instrução relacionada a E/S, ele executa essa instrução emitindo um comando ao módulo de E/S apropriado. Com a E/S programada, o módulo de E/S realizará a ação exigida e depois definirá os bits apropriados no registrador de estado de E/S (Figura 7.3). O módulo de E/S não toma outra ação para alertar o processador. Em particular, ele não interrompe o processador. Desse modo, é responsabilidade do processador verificar periodicamente o estado do módulo de E/S até descobrir se a operação terminou.

Para explicar a técnica de E/S programada, primeiro a veremos do ponto de vista dos comandos de E/S enviados pelo processador ao módulo de E/S, e depois do ponto de vista das instruções de E/S executadas pelo processador.

Comando de E/S

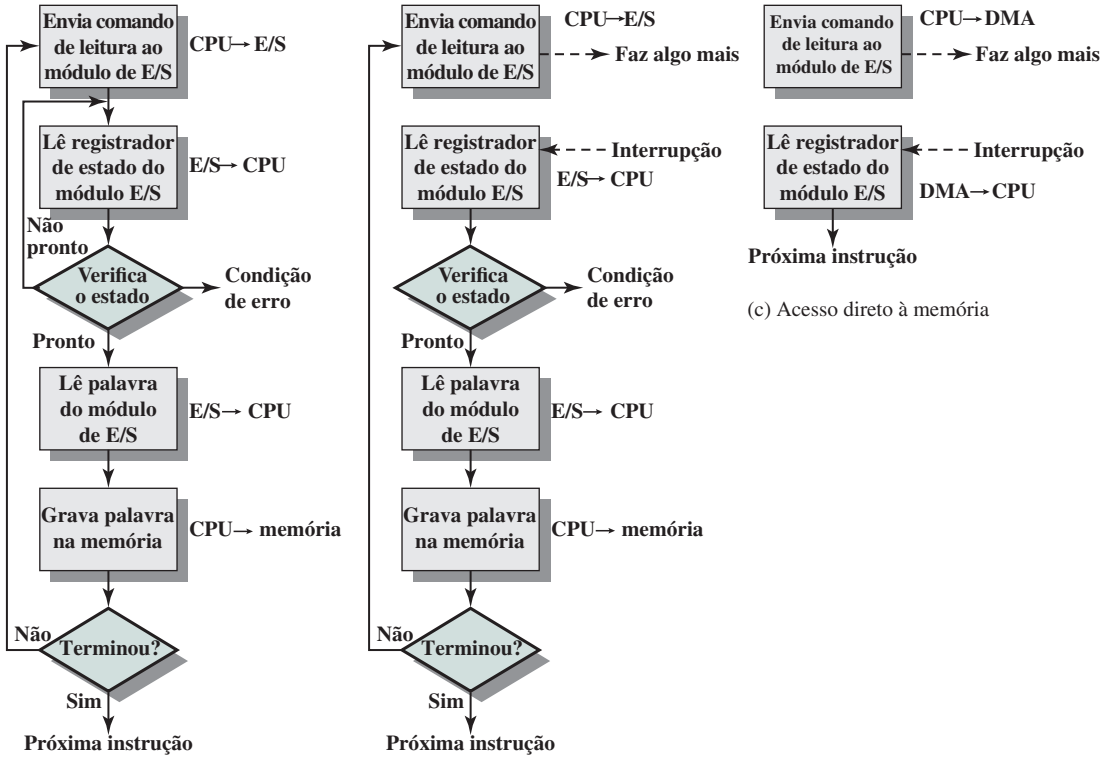
Para executar uma instrução relacionada a E/S, o processador envia um endereço, especificando um determinado módulo de E/S e dispositivo externo, e um comando de E/S. Existem quatro tipos de comandos de E/S que um módulo de E/S pode receber quando é endereçado por um processador:

- **Controle**: usado para ativar um periférico e dizer-lhe o que fazer. Por exemplo, uma unidade de fita magnética pode ser instruída a rebobinar ou mover para um registro à frente. Esses comandos são ajustados ao tipo específico de dispositivo periférico.
- **Teste**: usado para testar diversas condições de estado associadas a um módulo de E/S e seus periféricos. O processador precisará saber se o periférico de interesse está ligado e disponível para uso. Ele também precisará saber se a operação de E/S mais recente terminou e se houve algum erro.
- **Leitura**: faz com que o módulo de E/S obtenha um item de dados do periférico e o coloque em um buffer interno (representado como um registrador de dado na Figura 7.3). O processador pode obter o item de dado solicitando que o módulo de E/S o coloque no barramento de dados.
- **Escrita**: faz com que o módulo de E/S apanhe um item de dado (byte ou palavra) do barramento de dados e depois transmita esse item de dado ao periférico.

A Figura 7.4a dá um exemplo do uso da E/S programada para ler um bloco de dados de um dispositivo periférico (por exemplo, um registro da fita) para a memória. Os dados são lidos em uma palavra (por exemplo, 16 bits) de cada vez. Para cada palavra lida, o processador deve permanecer em um ciclo de verificação de estado até que determine que a palavra está disponível no registrador de dados do módulo de E/S. Esse fluxograma destaca a principal desvantagem dessa técnica: é um processo demorado, que mantém o processador ocupado desnecessariamente.

Figura 7.4

Três técnicas para entrada de um bloco de dados.



Instruções de E/S

Com a E/S programada, existe uma correspondência próxima entre as instruções relacionadas à E/S que o processador busca na memória e os comandos de E/S que o processador envia a um módulo de E/S para executar as instruções. Ou seja, as instruções são facilmente mapeadas em comandos de E/S, e em geral existe uma simples relação um para um. A forma da instrução depende do modo como os dispositivos externos são endereçados.

Em geral, haverá muitos dispositivos de E/S conectados, por meio dos módulos de E/S, ao sistema. Cada dispositivo recebe um identificador ou endereço exclusivo. Quando o processador emite um comando de E/S, o comando contém o endereço do dispositivo desejado. Desse modo, cada módulo de E/S deve interpretar as linhas de endereço para determinar se o comando é para ele mesmo.

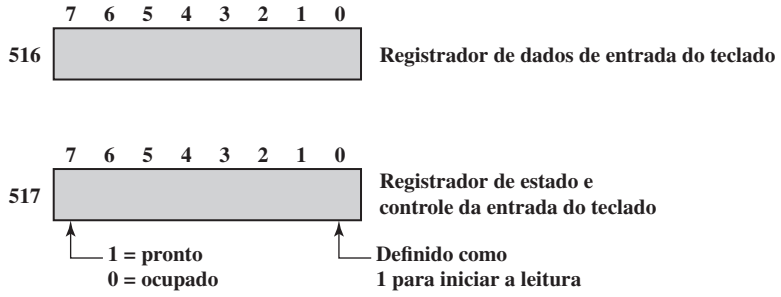
Quando o processador, a memória principal e a E/S compartilham um barramento comum, dois modos de endereçamento são possíveis: mapeado na memória e independente. Com a **E/S mapeada na memória**, existe um único espaço de endereço para locais de memória e dispositivos de E/S. O processador trata os registradores de estado e dados dos módulos de E/S como locais de memória e usa as mesmas instruções de máquina para acessar a memória e os dispositivos de E/S. Assim, por exemplo, com dez linhas de endereço, um total combinado de $2^{10} = 1.024$ locais de memória e endereços de E/S podem ser aceitos, em qualquer combinação.

Com a E/S mapeada na memória, uma única linha de leitura e uma única linha de escrita são necessárias no barramento. Como alternativa, o barramento pode ter linhas de leitura e escrita de memória além das linhas de comando de entrada e saída. A linha de comando especifica se o endereço se refere a um local de memória ou a um dispositivo de E/S. A faixa completa de endereços pode estar disponível para ambos. Novamente, com dez linhas de endereço, o sistema agora pode aceitar 1.024 locais de memória e 1.024 endereços de E/S. Como o espaço de endereço para E/S é independente do espaço da memória, isso é chamado de **E/S independente**.

A Figura 7.5 compara essas duas técnicas de E/S programada. A Figura 7.5a mostra como a interface para um dispositivo de entrada simples, como um teclado de um terminal, poderia aparecer a um programador usando a E/S mapeada na memória. Suponha um endereço de 10 bits, com uma memória de 512 bits (locais 0–511) e até 512 endereços de E/S (locais 512–1.023). Dois endereços são dedicados à entrada do teclado de um terminal em

Figura 7.5

E/S mapeada na memória e isolada.



ENDEREÇO	INSTRUÇÃO	OPERANDO	COMENTÁRIO
200	Carrega AC	"1"	Carrega acumulador
	Armazena AC	517	Inicia a leitura do teclado
202	Carrega AC	517	Colhe byte de estado
	Desvia, se sinal = 0	202	Loop até estar pronto
	Carrega AC	516	Carrega byte de dados

(a) E/S mapeada na memória

ENDEREÇO	INSTRUÇÃO	OPERANDO	COMENTÁRIO
200	Carrega E/S	5	Inicia a leitura do teclado
201	Testa E/S	5	Checa término
	Desvia se não pronto	201	Loop até estar pronto
	Entrada	5	Carrega byte de dados

(b) E/S independente

particular. O endereço 516 refere-se ao registrador de dados e o endereço 517 refere-se ao registrador de estado, que também funciona como um registrador de controle para receber comandos do processador. O programa mostrado lerá 1 byte de dados do teclado para um registrador acumulador no processador. Observe que o processador entra em um loop até que o byte de dados esteja disponível.

Com a E/S independente (Figura 7.5b), as portas de E/S são acessíveis apenas por comandos de E/S especiais, que ativam as linhas de comando de E/S no barramento.

Para a maioria dos tipos de processadores, existe um conjunto relativamente grande de diferentes instruções para referenciar a memória. Se a E/S independente for usada, haverá apenas algumas instruções de E/S. Desse modo, uma vantagem da E/S mapeada na memória é que esse grande repertório de instruções pode ser usado, permitindo uma programação mais eficiente. Uma desvantagem é que um espaço de endereços de memória valioso é utilizado. Tanto a E/S mapeada na memória quanto a E/S independente são comumente utilizadas.

7.4 E/S CONTROLADA POR INTERRUPÇÃO

O problema com a E/S programada é que o processador tem de esperar muito tempo para que o módulo de E/S de interesse esteja pronto para recepção ou transmissão de dados. O processador, enquanto espera, deve verificar repetidamente o estado do módulo de E/S. Como resultado, o nível de desempenho do sistema inteiro é bastante degradado.

Uma alternativa é que o processador envie um comando de E/S para um módulo e depois continue realizando algum outro trabalho útil. O módulo de E/S, então, interromperá o processador para solicitar atendimento quando estiver pronto para trocar dados com o processador. O processador, então, executará a transferência de dados, como antes, e depois retomará seu processamento anterior.

Vamos considerar como isso funciona, primeiro do ponto de vista do módulo de E/S. Para a entrada, o módulo de E/S recebe um comando READ do processador. O módulo de E/S, então, prossegue para ler dados de um periférico associado. Quando os dados estão no registrador de dados do módulo, o módulo envia um sinal de interrupção ao processador por uma linha de controle. O módulo, então, espera até que seus dados sejam solicitados pelo processador. Quando a solicitação é feita, o módulo coloca seus dados no barramento de dados e, então, está pronto para outra operação de E/S.

Do ponto de vista do processador, a ação para entrada é a seguinte. O processador emite um comando READ. Depois, ele prossegue com outras tarefas (por exemplo, o processador pode estar trabalhando em vários programas diferentes ao mesmo tempo). Ao final de cada ciclo de instrução, o processador checa se há interrupções (Figura 3.9). Quando ocorre uma interrupção do módulo de E/S, o processador salva o contexto (por exemplo, o contador de programa e os registradores do processador) do programa atual e processa a interrupção. Nesse caso, o processador lê a palavra de dados do módulo de E/S e a armazena na memória. Depois, ele restaura o contexto do programa em que estava trabalhando (ou de algum outro programa) e retoma a execução.

A Figura 7.4b mostra o uso da E/S por interrupção para leitura de um bloco de dados. Compare isso com a Figura 7.4a. A E/S por interrupção é mais eficiente do que a E/S programada, pois elimina a espera desnecessária. Contudo, a E/S por interrupção ainda consome muito tempo do processador, pois cada palavra de dados que vem da memória para o módulo de E/S ou do módulo de E/S para a memória deve passar pelo processador.

Processamento de interrupção

Vamos examinar com mais detalhes o papel do processador na E/S controlada por interrupção. O surgimento de uma interrupção dispara uma série de eventos, tanto no hardware do processador quanto no software. A Figura 7.6 mostra uma sequência típica. Quando o dispositivo de E/S completa uma operação de E/S, ocorre a seguinte sequência de eventos de hardware:

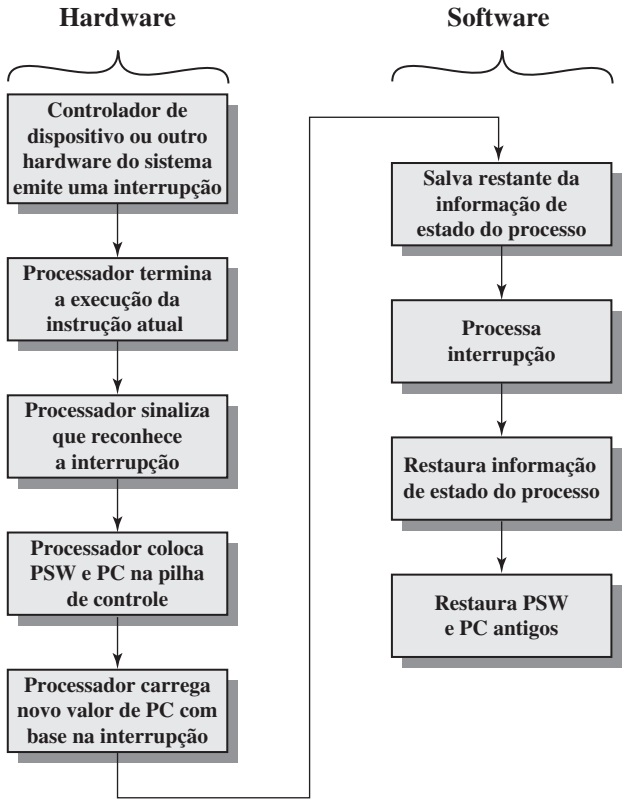
1. O dispositivo emite um sinal de interrupção ao processador.
2. O processador termina a execução da instrução atual antes de responder à interrupção, conforme indicado na Figura 3.9.
3. O processador checa a existência de uma interrupção, constata que existe interrupção e envia um sinal de confirmação ao dispositivo que a emitiu. A confirmação possibilita que o dispositivo remova seu sinal de interrupção.
4. O processador agora precisa se preparar para transferir o controle à rotina de interrupção. Para começar, ele precisa salvar as informações necessárias para retornar ao programa atual no ponto da interrupção. As informações mínimas exigidas são (a) o estado do processador, que está contido em um registrador chamado **palavra de estado do programa (PSW — *Program Status Word*)**, e (b) o local da próxima instrução a ser executada, que está contida no contador de programa. Estas podem ser colocadas na pilha de controle do sistema.[2]
5. O processador agora carrega o contador de programa com o local de endereço inicial da rotina de tratamento de interrupção que responderá a essa interrupção. Dependendo da arquitetura do computador e do projeto do sistema operacional, pode haver uma única rotina, uma rotina para cada tipo de interrupção ou uma rotina para cada dispositivo e cada tipo de interrupção. Se houver mais de uma rotina de tratamento de interrupção, o processador deve determinar qual chamará. Essa informação pode ter sido incluída no sinal de interrupção original, ou o processador pode ter de emitir uma solicitação ao dispositivo que emitiu a interrupção, para obter uma resposta que contenha a informação necessária.

Quando o contador de programa tiver sido carregado, o processador segue para o próximo ciclo de instrução, que começa com uma busca de instrução. Como a busca de instrução é determinada pelo conteúdo do contador de programa, o resultado é que o controle é transferido para o programa de tratamento de interrupção. A execução desse programa resulta nas seguintes operações:

6. Nesse ponto, o contador de programa e PSW relacionados ao programa interrompido foram salvos na pilha do sistema. Todavia, existe outra informação que é considerada parte do "estado" do programa em execução. Em particular, os conteúdos dos registradores do processador precisam ser salvos, pois esses registradores podem ser usados pela rotina de manipulação de interrupção. Assim, todos os valores, mais

2 Veja no Apêndice I (disponível em inglês na Sala Virtual) uma discussão a respeito da operação da pilha.

Figura 7.6
Processamento de interrupção simples.



qualquer outra informação de estado, devem ser salvos. Em geral, a rotina de tratamento de interrupção começará salvando o conteúdo dos registradores na pilha. A Figura 7.7a mostra um exemplo simples. Nesse caso, um programa do usuário é interrompido após a instrução no local *N*. O conteúdo de todos os registradores mais o endereço da próxima instrução ($N + 1$) são colocados na pilha. O ponteiro de pilha é atualizado para apontar para o novo topo da pilha, e o contador de programa é atualizado para apontar para o início da rotina de serviço de interrupção.

7. A rotina de tratamento de interrupção em seguida processa a interrupção. Isso inclui uma verificação da informação de estado relacionada à operação de E/S ou outro evento que causou uma interrupção. Ele também pode envolver o envio de comandos ou confirmações adicionais ao dispositivo de E/S.
8. Quando o processamento da interrupção termina, os valores dos registradores salvos são recuperados da pilha e restaurados aos registradores (por exemplo, ver Figura 7.7b).
9. O ato final é restaurar os valores do PSW e do contador de programa da pilha. Como resultado, a próxima instrução a ser executada será do programa previamente interrompido.

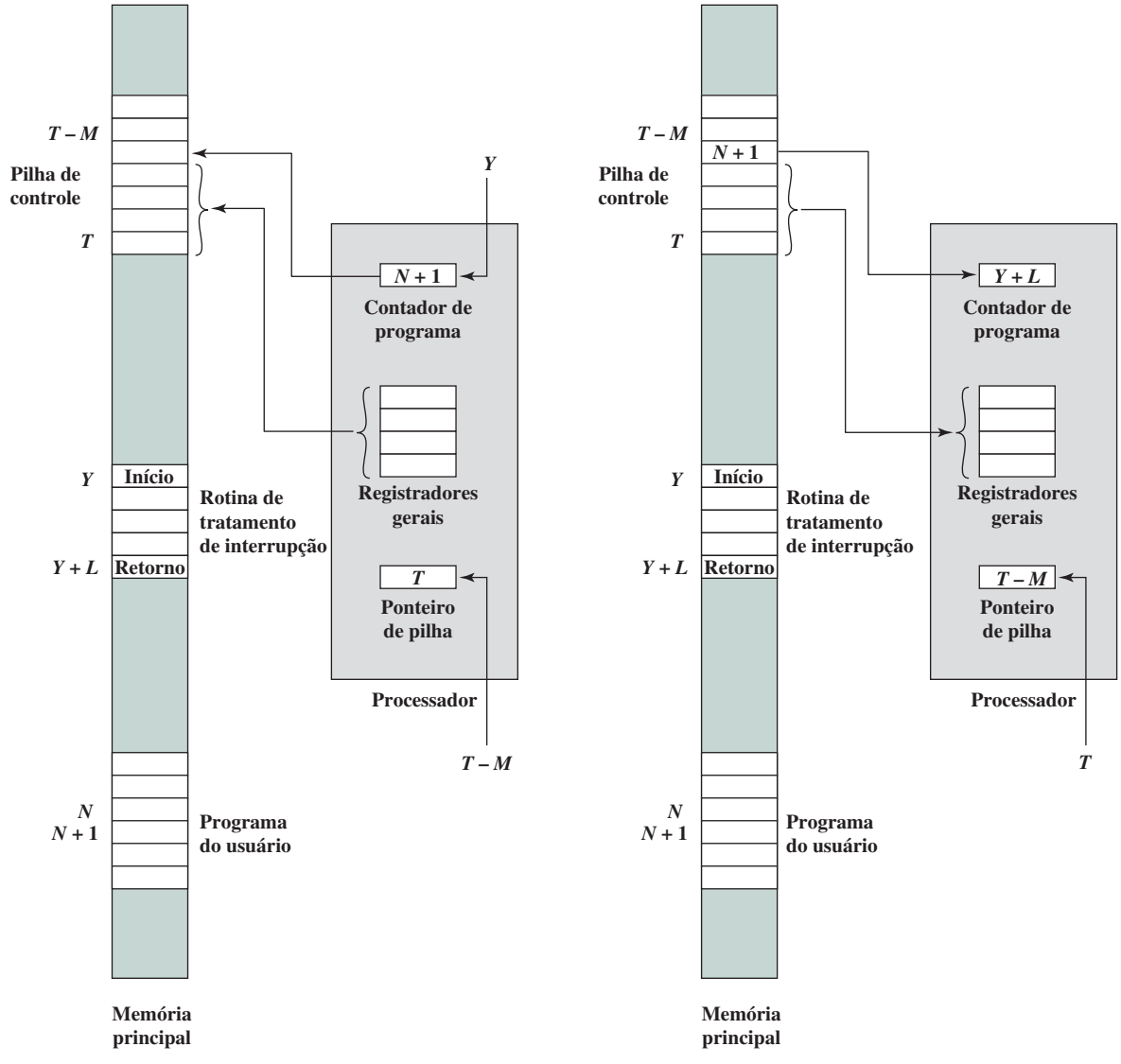
Observe que é importante salvar todas as informações de estado do programa interrompido para a retomada posterior, pois a interrupção não é uma rotina chamada pelo programa. Em vez disso, ela pode ocorrer a qualquer momento e, portanto, em qualquer ponto na execução de um programa do usuário. Sua ocorrência é imprevisível. Na verdade, conforme veremos no próximo capítulo, os dois programas podem não ter algo em comum e podem pertencer a dois usuários diferentes.

Aspectos de projeto

Dois aspectos de projeto surgem na implementação da E/S por interrupção. O primeiro é, considerando que quase sempre haverá vários módulos de E/S, como o processador determina qual dispositivo emitiu a interrupção? E o segundo é: se houver várias interrupções, como o processador decide qual deverá processar?

Figura 7.7

Mudanças na memória e registradores para uma interrupção.



(a) Interrupção ocorre após instrução no local

(b) Retorno da interrupção

Vamos considerar primeiro a identificação do dispositivo. Quatro categorias gerais de técnicas são comumente utilizadas:

- Múltiplas linhas de interrupção.
- Verificação por software.
- *Daisy chain* (verificação por hardware, vetorizado).
- Arbitração de barramento (vetorizado).

A técnica mais simples para o problema é oferecer **múltiplas linhas de interrupção** entre o processador e os módulos de E/S. Todavia, é impraticável dedicar mais do que algumas poucas linhas de barramento ou pinos de processador às linhas de interrupção. Consequentemente, mesmo que várias linhas sejam usadas, é provável que cada uma terá múltiplos módulos de E/S conectados a ela. Assim, uma das outras três técnicas deve ser usada em cada linha.

Uma alternativa é a **verificação por software**. Quando o processador detecta uma interrupção, ele desvia para uma rotina de tratamento de interrupção cuja tarefa é verificar cada módulo de E/S para determinar qual módulo causou a interrupção. A verificação poderia ser por uma linha de comando separada (por exemplo, TESTI/O). Nesse caso, o processador levanta TESTI/O e coloca o endereço de um módulo de E/S em particular nas linhas de endereço. O módulo de E/S responde positivamente se solicitou a interrupção. Como alternativa, cada módulo de E/S poderia conter um registrador de estado endereçável. O processador, então, lê o registrador de estado de cada módulo de E/S para identificar o módulo que gerou a interrupção. Quando o módulo é identificado, o processador inicia a execução da rotina de tratamento de interrupção para este dispositivo.

A desvantagem da verificação por software é que ele é demorado. Uma técnica mais eficiente é usar uma configuração ***daisy chain***, que oferece uma verificação por hardware. Um exemplo de uma configuração *daisy chain* aparece na Figura 3.26. Para interrupções, todos os módulos de E/S compartilham uma linha de requisição de interrupção comum. A linha de confirmação de interrupção é estruturada em forma de uma cadeia circular através dos módulos. Quando o processador reconhece uma interrupção, ele envia uma confirmação de interrupção. Esse sinal se propaga por uma série de módulos de E/S até ser recebido pelo módulo requisitante. O módulo requisitante normalmente responde colocando uma palavra das linhas de dados. Essa palavra é conhecida como *vetor* e é o endereço do módulo de E/S ou algum outro identificador exclusivo. De qualquer forma, o processador usa o vetor como um ponteiro para a rotina apropriada de serviço de dispositivo. Isso evita a necessidade de executar uma rotina de tratamento de interrupção geral primeiro. Essa técnica é chamada de *interrupção vetorada.*

Existe outra técnica que utiliza interrupções *vetoradas*, que é a **arbitração de barramento**. Com a arbitração de barramento, um módulo de E/S deve primeiro ganhar o controle do barramento, antes de poder ativar a requisição de interrupção. Assim, somente um módulo pode ativar a linha de cada vez. Quando o processador detecta a interrupção, ele responde na linha de reconhecimento de interrupção. O módulo requisitante, então, coloca seu vetor nas linhas de dados.

As técnicas que mencionamos servem para identificar o módulo de E/S requisitante. Elas também oferecem um modo de atribuir prioridades quando mais de um dispositivo está requisitando serviço de interrupção. Com múltiplas linhas, o processador apenas apanha a linha de interrupção com a prioridade mais alta. Com a verificação por software, a ordem em que os módulos são verificados determina suas prioridades. De modo semelhante, a ordem dos módulos em uma *daisy chain* determina suas prioridades. Por fim, a arbitração de barramento pode empregar um esquema de prioridade, conforme discutimos na Seção 3.4.

Agora, vamos examinar dois exemplos de estruturas de interrupção.

Controlador de interrupção Intel 82C59A

O Intel 80386 oferece uma única linha para *Interrupt Request* (INTR) e uma única linha para *Interrupt Acknowledge* (INTA). Para permitir que o 80386 trate de diversos dispositivos e estruturas de prioridade, ele costuma ser configurado com um arbitrador externo de interrupção, o 82C59A. Os dispositivos externos são conectados ao 82C59A, que, por sua vez, se conecta ao 80386.

A Figura 7.8 mostra o uso do 82C59A para conectar múltiplos módulos de E/S para o 80386. Um único 82C59A pode tratar de até oito módulos. Se for preciso controlar mais de oito módulos, um arranjo em cascata pode ser usado, para tratar até 64 módulos.

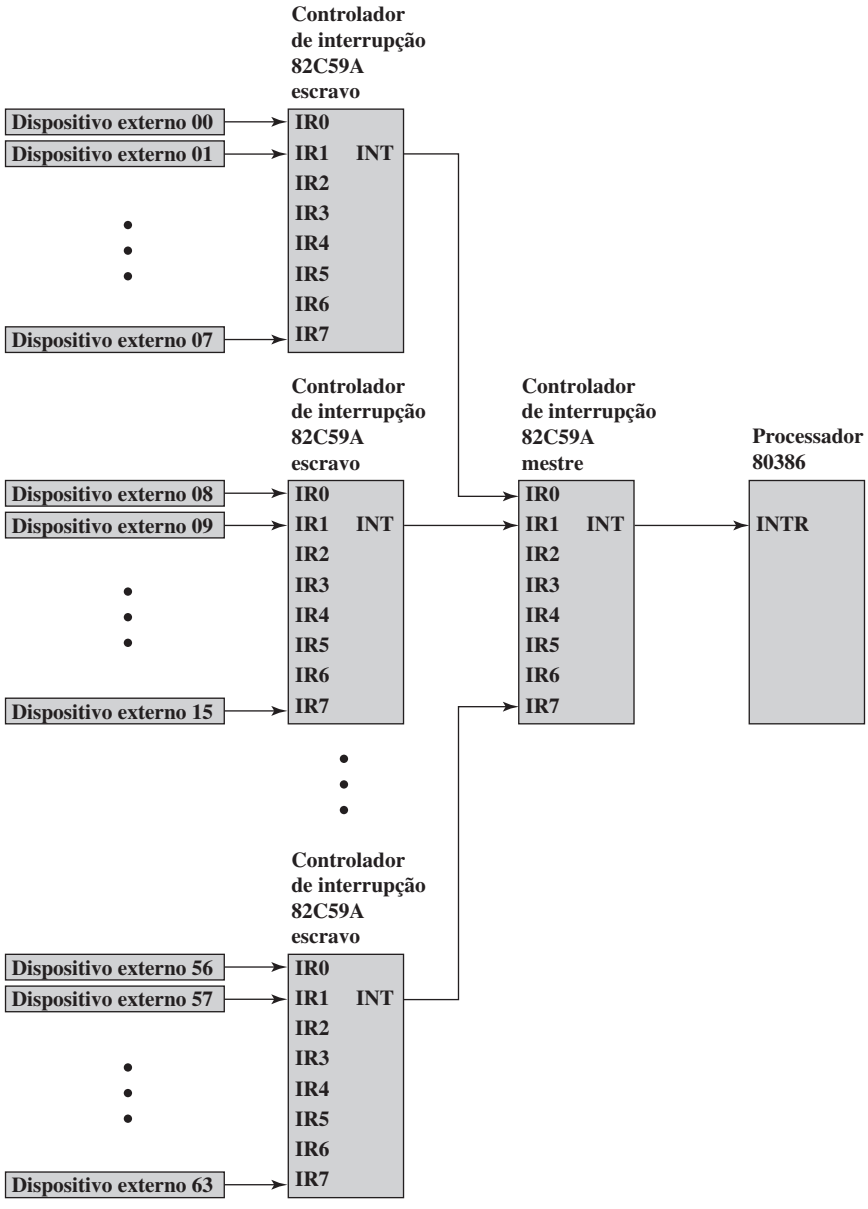
A única responsabilidade do 82C59A é o gerenciamento de interrupções. Ele aceita requisições de interrupção dos módulos conectados, determina qual interrupção tem a maior prioridade e depois sinaliza o processador levantando a linha INTR. O processador confirma por meio da linha INTA. Isso leva o 82C59A a colocar a informação apropriada do vetor no barramento de dados. O processador pode, então, prosseguir para processar a interrupção e se comunicar diretamente com o módulo de E/S para ler ou gravar dados.

O 82C59A é programável. O 80386 determina o esquema de prioridade a ser usado definindo uma palavra de controle no 82C59A. Os seguintes modos de interrupção são possíveis:

- **Totalmente aninhado**: as requisições de interrupção são ordenadas na prioridade de 0 (IR0) até 7 (IR7).
- **Rotação**: em algumas aplicações, diversos dispositivos que geram interrupções têm a mesma prioridade. Nesse modo, um dispositivo, depois de ser atendido, recebe a menor prioridade no grupo.
- **Máscara especial**: isso permite que o processador iniba interrupções de certos dispositivos.

Figura 7.8

Uso do controlador de interrupção 82C59A.



A interface de periférico programável Intel 8255A

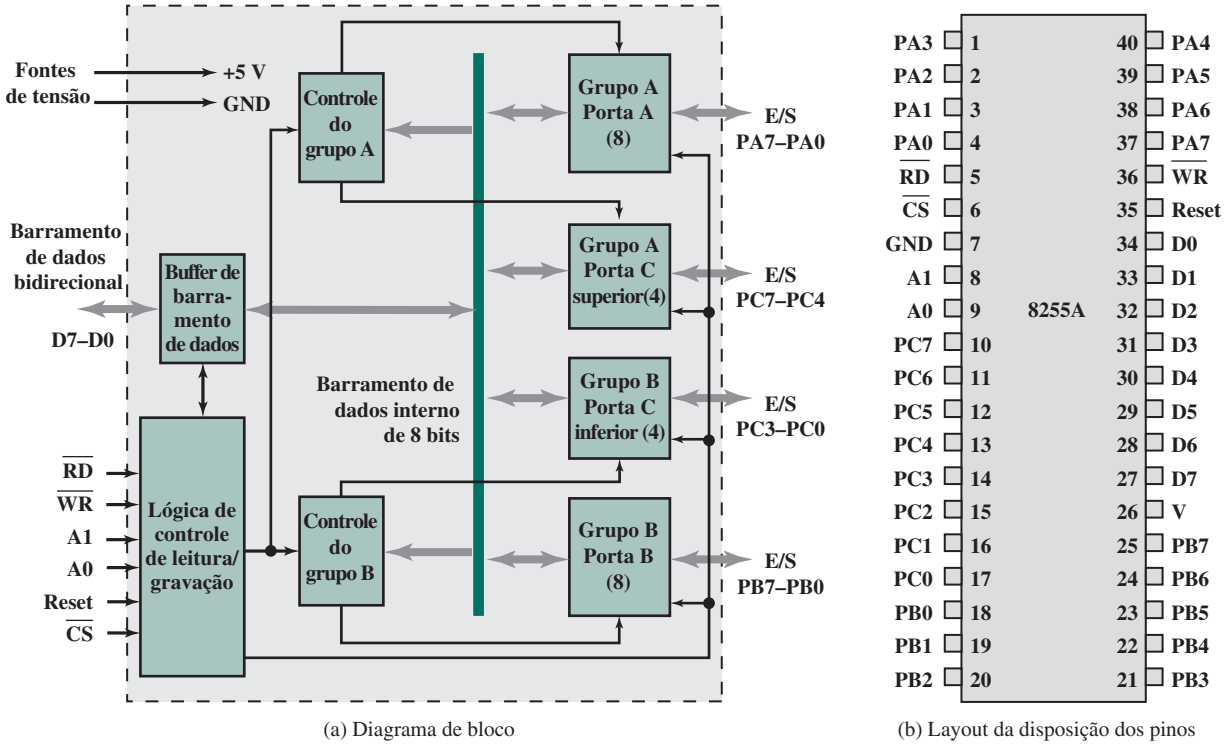
Como um exemplo de um módulo de E/S usado para a E/S programada e E/S controlada por interrupção, consideramos o módulo Intel 8255A Programmable Peripheral Interface. O 8255A é um módulo de E/S de uso geral em um único chip, projetado para uso com o processador Intel 80386. Tem sido desde então clonado por outros fabricantes e é um chip de controlador periférico amplamente usado. Seu uso inclui um controlador para dispositivos simples de E/S para microprocessadores e em sistemas embarcados, incluindo sistemas com microcontroladores.

ARQUITETURA E OPERAÇÃO A Figura 7.9 mostra um diagrama de bloco geral e o assinalamento dos pinos para uma cápsula de 40 pinos, onde o chip é montado. Conforme mostrado no layout da disposição dos pinos, o 8255A inclui as seguintes definições:

- **D0–D7**: essas são as linhas de E/S de dados para o dispositivo. Toda a informação lida e gravada no 8255A ocorre por meio dessas oito linhas.

Figura 7.9

Interface de periférico programável Intel 8255A.



(a) Diagrama de bloco

(b) Layout da disposição dos pinos

- **$\overline{\text{CS}}$ (entrada de seleção de chip)**: se essa linha é um 0 lógico, o microprocessador pode ler e escrever no 8255A.
- **$\overline{\text{RD}}$ (entrada de leitura)**: se essa linha é um 0 lógico e a entrada de CS é um 0 lógico, as saídas de dados do 8255A são habilitadas no barramento dos dados do sistema.
- **$\overline{\text{WR}}$ (entrada de gravação)**: se essa linha de entrada é um 0 lógico e a entrada de CS é um 0 lógico, os dados são escritos no 8255A a partir do barramento de dados do sistema.
- **RESET**: o 8255A é colocado em seu estado de reset se a linha de entrada é um 1 lógico. Todas as portas periféricas são inicializadas no modo de entrada.
- **PA0–PA7, PB0–PB7, PC0–PC7**: as linhas de sinal são usadas como portas de E/S de 8 bits. Elas podem ser conectadas aos dispositivos periféricos.
- **A0, A1**: a combinação lógica dessas duas linhas de entrada determina em qual registrador interno do 8255A os dados são escritos ou lidos.

O lado direito do diagrama de blocos da Figura 7.9a é a interface externa do 8255A. As 24 linhas de E/S são divididas em três grupos de 8 bits (A, B, C). Cada grupo pode funcionar como uma porta de E/S de 8 bits, proporcionando, assim, conexão de dispositivos periféricos. Além disso, o grupo C é subdividido em grupos de 4 bits (C_A e C_B), que podem ser usados em conjunto com as portas de E/S A e B. Configuradas dessa forma, as linhas do grupo C são utilizadas para sinais de controle e de estado.

O lado esquerdo do diagrama em blocos é a interface interna para o barramento do sistema do microprocessador. Ele inclui um barramento de dados bidirecional de 8 bits (D0 até D7), usado para transferir dados entre o microprocessador e as portas de E/S e para transferir informações de controle.

O processador controla o 8255A por meio do registrador de controle de 8 bits no processador. O processador pode definir o valor do registrador de controle para especificar a variedade dos modos e configuração de operação. Do ponto de vista do processador, há uma porta de controle, e os bits do registrador de controle são definidos no processador e, então, enviados para a porta de controle por meio das linhas D0–D7. As duas linhas de endereço especificam uma das três portas de E/S ou o registrador de controle, do seguinte modo:

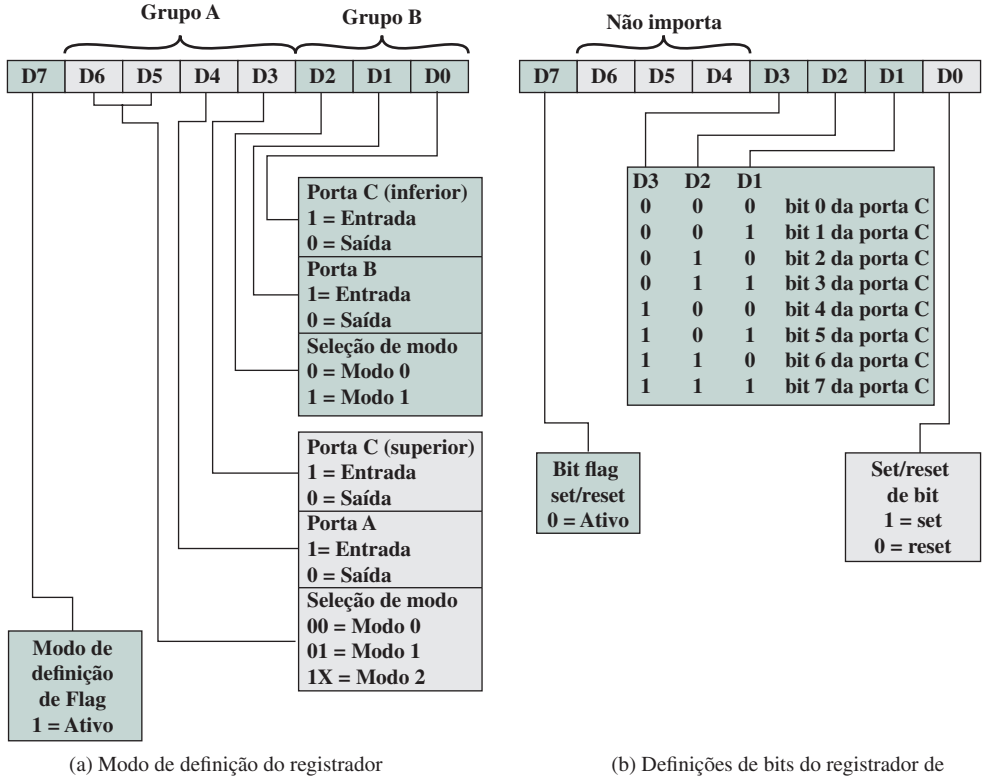
A1	A2	Seleciona
0	0	Porta A
0	1	Porta B
1	0	Porta C
1	1	Registrador de controle

Desse modo, quando o processador define tanto A1 como A2 como 1, o 8255A interpreta o valor de 8 bits no barramento de dados como uma palavra de controle. Quando o processador transfere a palavra de controle de 8 bits com a linha D7 definida a 1 (Figura 7.10a), a palavra de controle é usada para configurar o modo de operação das 24 linhas de E/S. Os três modos são:

- **Modo 0**: esse é o modo de E/S básico. Os três grupos de oito linhas externas funcionam como três portas de E/S de 8 bits. Cada porta pode ser projetada como entrada ou saída. Os dados só podem ser enviados para uma porta se ela for definida como saída, e os dados só podem ser lidos a partir da porta se ela for definida como entrada.
- **Modo 1**: nesse modo, as portas A e B podem ser configuradas como entrada ou saída, e as linhas da porta C servem como linhas de controle para A e B. Os sinais de controle têm dois propósitos principais: *handshaking* e requisição de interrupção. O *handshaking* é um mecanismo de temporização simples. Uma linha de controle é usada pelo emissor como uma linha DATA READY, para indicar quando os dados estão presentes nas linhas de dados de E/S. Outra linha é usada pelo receptor como um ACKNOWLEDGE, indicando que os dados foram lidos e as linhas de dados podem ser apagadas. Outra linha pode ser designada como uma linha INTERRUPT REQUEST e ligada de volta ao barramento do sistema.
- **Modo 2**: esse é o modo bidirecional. Nesse modo, uma porta A pode ser configurada tanto como linhas de entrada como de saída para o tráfego bidirecional, com as linhas da porta B sendo definidas na direção oposta. De novo, as linhas da porta C são usadas para a sinalização de controle.

Figura 7.10

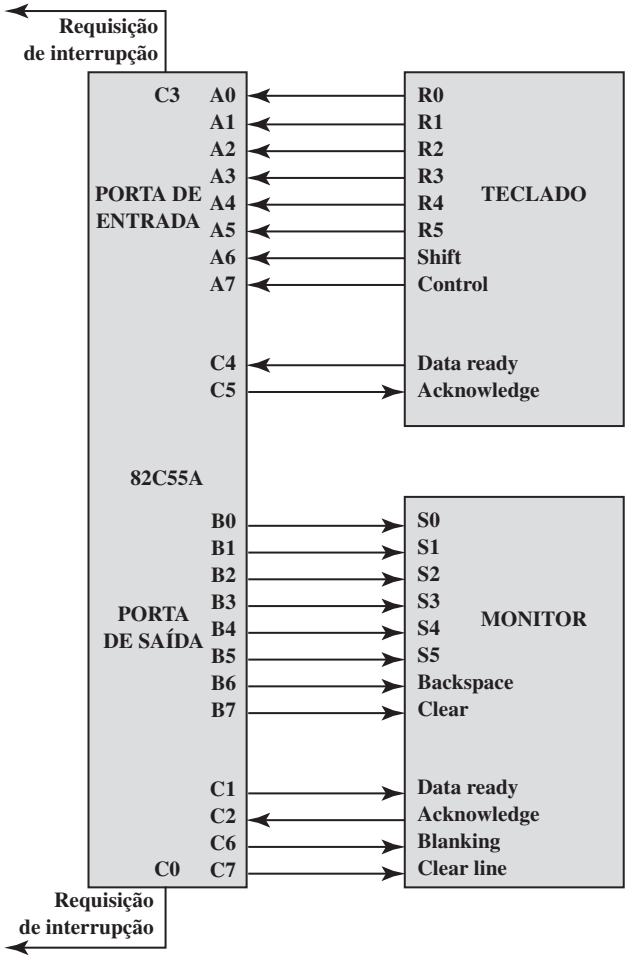
Palavra de controle do Intel 8255A.



(a) Modo de definição do registrador de controle 8255 para configurar o 8255

(b) Definições de bits do registrador de controle 8255 para modificar os bits da porta C

Figura 7.11
Interface de teclado/monitor para o 8255A.



Quando o processador define D7 igual a 0 (Figura 7.10b), a palavra de controle é usada para programar os valores de bit da porta C de modo individual. Essa característica é raramente usada.

EXEMPLO DE TECLADO/MONITOR DE VÍDEO Por conta de o 8255A ser programável pelo registrador de controle, pode ser usado para controlar uma série de dispositivos periféricos simples. A Figura 7.11 ilustra seu uso para controlar um terminal com teclado e monitor de vídeo. O teclado oferece 8 bits de entrada. Dois desses bits, SHIFT e CONTROL, possuem significado especial ao programa de tratamento de teclado executado pelo processador. Contudo, essa interpretação é transparente ao 8255A, que simplesmente aceita os 8 bits de dados e os apresenta no barramento de dados do sistema. Duas linhas de controle de *handshaking* são fornecidas para uso com o teclado.

O monitor também é ligado por uma porta de dados de 8 bits. Novamente, dois dos bits possuem significados especiais, que são transparentes ao 8255A. Além das duas linhas de *handshaking*, duas linhas oferecem funções de controle adicionais.

7.5 ACESSO DIRETO À MEMÓRIA

Desvantagens da E/S programada e controlada por interrupção

A E/S controlada por interrupção, embora mais eficiente que a E/S programada, ainda requer a intervenção ativa do processador para transferir dados entre a memória e um módulo de E/S. Além disso, quaisquer

transferências de dados precisam atravessar um caminho passando pelo processador. Assim, essas duas formas de E/S têm duas desvantagens inerentes:

1. A taxa de transferência de E/S é limitada pela velocidade com a qual o processador pode testar e atender a um dispositivo.
2. O processador fica ocupado no gerenciamento de uma transferência de E/S; diversas instruções precisam ser executadas para cada transferência de E/S (como um exemplo, veja a Figura 7.5).

Existe uma espécie de escolha entre essas duas desvantagens. Considere a transferência de um bloco de dados. Usando a E/S programada simples, o processador é dedicado à tarefa de E/S e pode mover dados em uma taxa relativamente alta, à custa de não fazer mais nada. A E/S por interrupção libera o processador até certo ponto, mas depende da taxa de transferência de E/S. Apesar disso, os dois métodos possuem um impacto negativo sobre a atividade do processador e a taxa de transferência de E/S.

Quando grandes volumes de dados precisam ser movidos, uma técnica mais eficiente é necessária: acesso direto à memória (DMA).

Função do DMA

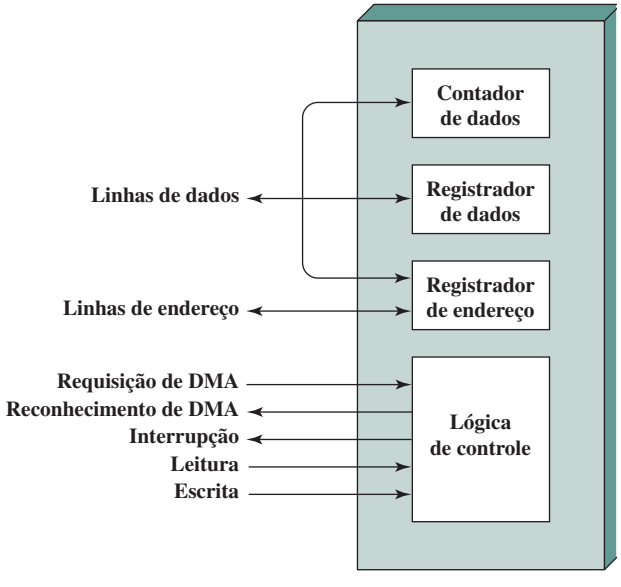
O DMA envolve um módulo adicional no barramento do sistema. O módulo de DMA (Figura 7.12) é capaz de imitar o processador e, na realidade, assumir o controle do sistema do processador. Ele precisa fazer isso para transferir dados de e para a memória pelo barramento do sistema. Para esse propósito, o módulo de DMA deve usar o barramento apenas quando o processador não precisa dele, ou então deve forçar o processador a suspender a operação temporariamente. Essa última técnica é mais comum e é conhecida como *roubo de ciclo* (*cycle stealing*), pois o módulo de DMA efetivamente rouba um ciclo do barramento.

Quando o processador deseja ler ou escrever um bloco de dados, ele envia um comando ao módulo de DMA com as seguintes informações:

- Se uma leitura ou escrita é solicitada, usando o controle de leitura e escrita entre o processador e o módulo de DMA.
- O endereço do dispositivo de E/S envolvido, comunicado nas linhas de dados.
- O local inicial na memória para ler ou escrever, comunicado nas linhas de dados e armazenado pelo módulo de DMA em seu registrador de endereço.
- O número de palavras a serem lidas ou gravadas, novamente comunicado por meio das linhas de dados e armazenado no contador de dados.

Figura 7.12

Diagrama em blocos típico do DMA.



O processador, então, continua com outro trabalho. Ele delegou essa operação de E/S a um módulo de DMA. O módulo de DMA transfere o bloco de dados inteiro, uma palavra de cada vez, diretamente de ou para a memória, sem passar pelo processador. Quando a transferência termina, o módulo de DMA envia um sinal de interrupção ao processador. Desse modo, o processador é envolvido apenas no início e no final da transferência (Figura 7.4c).

A Figura 7.13 mostra onde, no ciclo de instrução, o processador pode ser suspenso. Em cada caso, o processador é suspenso exatamente antes de precisar usar o barramento. O módulo de DMA, então, transfere uma palavra e retorna o controle ao processador. Observe que isso não é uma interrupção; o processador não salva o contexto e faz algo mais. Em vez disso, o processador é interrompido por um ciclo do barramento. O efeito geral é fazer com que o processador execute mais lentamente. Apesar disso, para uma transferência de E/S de múltiplas palavras, o DMA é muito mais eficiente do que a E/S controlada por interrupção ou programada.

O mecanismo de DMA pode ser configurado de diversas maneiras. Algumas possibilidades aparecem na Figura 7.14. No primeiro exemplo, todos os módulos compartilham o mesmo barramento do sistema. O módulo de DMA, atuando como um processador substituto, utiliza E/S programada para trocar dados entre a memória e um módulo de E/S por meio do módulo de DMA. Essa configuração, embora possa ser pouco dispendiosa, certamente é ineficaz. Assim como a E/S programada controlada pelo processador, cada transferência de uma palavra consome dois ciclos de barramento.

O número de ciclos de barramento exigidos pode ser reduzido substancialmente integrando as funções de DMA e E/S. Como a Figura 7.14b indica, isso significa que existe um caminho entre o módulo de DMA e um ou mais módulos de E/S, que não inclui o barramento do sistema. A lógica de DMA pode realmente fazer parte de um módulo de E/S, ou pode ser um módulo separado que controla um ou mais módulos de E/S. Esse conceito pode ser levado um passo adiante conectando módulos de E/S a um módulo de DMA, usando um barramento de E/S (Figura 7.14c). Isso reduz o número de interfaces de E/S no módulo de DMA para um e oferece uma configuração facilmente expansível. Nesses dois casos (figuras 7.14b e c), o barramento do sistema, que o módulo de DMA compartilha com o processador e a memória, é usado pelo módulo de DMA somente para trocar dados com a memória. A troca de dados entre os módulos de DMA e E/S ocorre fora do barramento do sistema.

Figura 7.13

DMA e pontos de interrupção durante um ciclo de instrução.

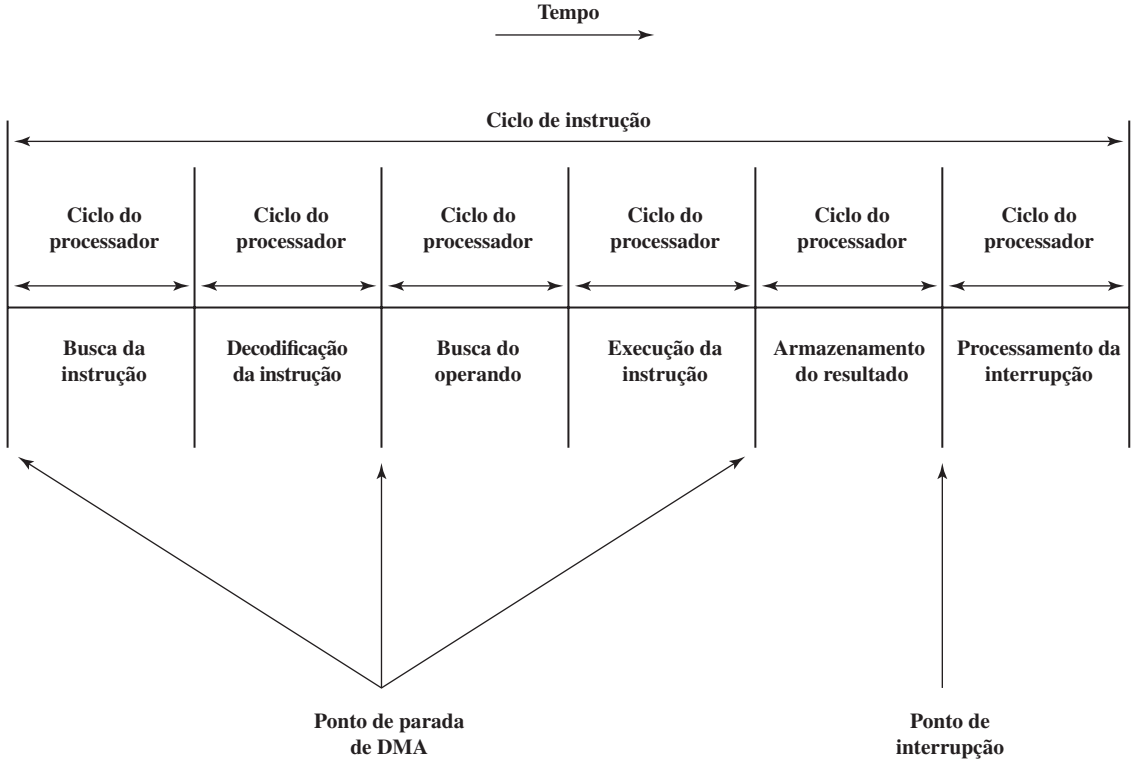
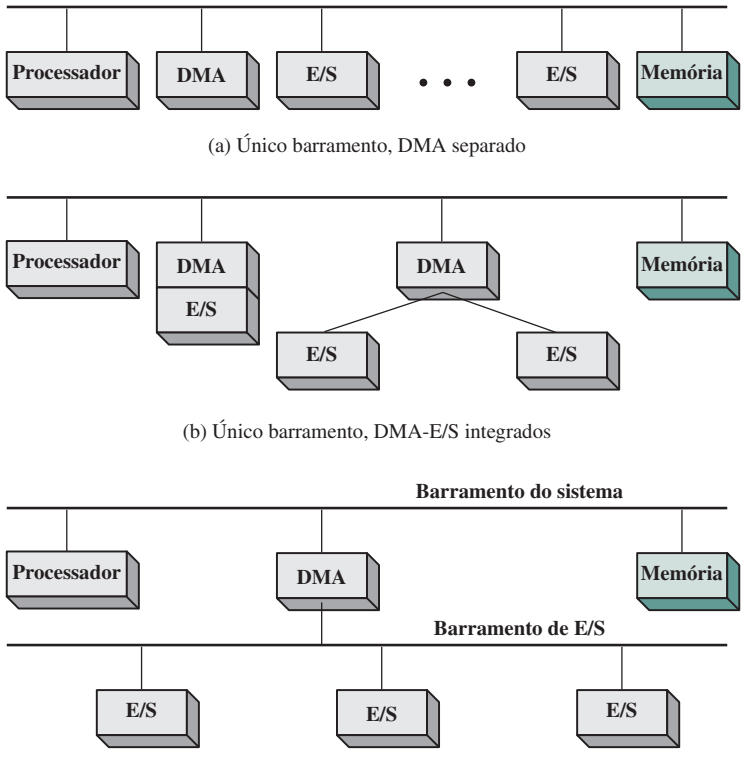


Figura 7.14
Configurações de DMA alternativas.



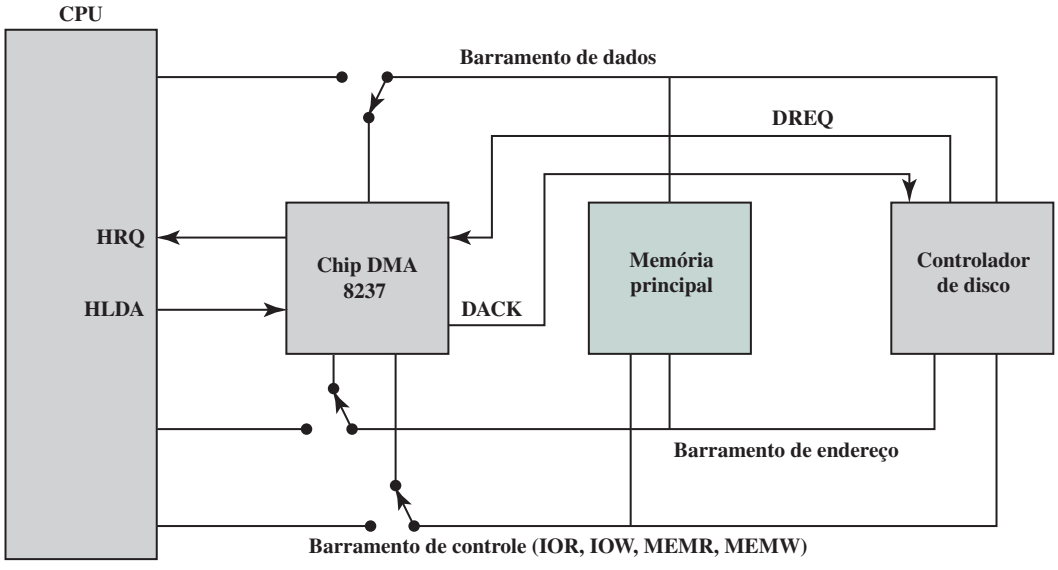
Controlador de DMA Intel 8237A

O controlador de DMA Intel 8237A realiza a interface com a família de processadores 80x86 e com uma memória DRAM, para oferecer uma capacidade de DMA. A Figura 7.15 indica o local do módulo de DMA. Quando o módulo de DMA precisa usar os barramentos do sistema (dados, endereço e controle) para transferir dados, ele envia um sinal denominado HOLD ao processador. O processador responde com o sinal HLDA (*hold acknowledge*), indicando que o módulo de DMA pode usar os barramentos. Por exemplo, se o módulo de DMA tiver que transferir um bloco de dados da memória ao disco, ele fará o seguinte:

1. O dispositivo periférico (como o controlador de disco) requisitará o serviço de DMA levantando o sinal DREQ (requisição de DMA).
2. O DMA levantará sua linha HRQ (requisição de HOLD), sinalizando à CPU através de seu pino HOLD que ele precisa usar os barramentos.
3. A CPU terminará o ciclo de barramento atual (não necessariamente a instrução atual) e responderá à solicitação de DMA levantando sua linha HDLA (confirmação de HOLD), dizendo, assim, ao DMA 8237 que ele pode seguir em frente e usar os barramentos para realizar sua tarefa. A linha de HOLD deve permanecer ativa enquanto o DMA estiver realizando sua tarefa.
4. O DMA ativará a linha DACK (confirmação de DMA), que diz ao dispositivo periférico que ele começará a transferir os dados.
5. O DMA começa a transferir os dados da memória para o periférico, colocando o endereço do primeiro byte do bloco no barramento de endereço e ativando MEMR, lendo, assim, o byte da memória para o barramento de dados; depois, ele ativa IOW para gravá-lo no periférico. Em seguida, o DMA decrementa o contador e incrementa o ponteiro de endereço, repetindo esse processo até que a contagem chegue a zero e a tarefa esteja encerrada.
6. Depois que o DMA terminar seu trabalho, ele desativará HRQ, sinalizando à CPU que ela pode retomar o controle de seus barramentos.

Figura 7.15

Uso do barramento do sistema pelo controlador de DMA 8237.



DACK = DMA *acknowledge* (reconhecimento de DMA)
DREQ = DMA *request* (requisição de DMA)
HLDA = HOLD *acknowledge* (reconhecimento de HOLD)
HRQ = HOLD *request* (requisição de HOLD)

Enquanto o DMA está usando os barramentos para transferir dados, o processador fica ocioso. De modo semelhante, quando o processador está usando o barramento, o DMA fica ocioso. O DMA 8237 é conhecido como um controlador de DMA flutuante. Isso significa que os dados movidos de um local para outro não passam pelo chip de DMA e não são armazenados nele. Portanto, o DMA só pode transferir dados entre uma porta de E/S e um endereço de memória, mas não entre duas portas de E/S ou dois locais de memória. Todavia, conforme explicamos mais adiante, o chip de DMA pode realizar uma transferência de memória a memória por meio de um registrador.

O 8237 contém quatro canais de DMA, que podem ser programados independentemente, e qualquer um deles pode estar ativo a qualquer momento. Esses canais são numerados com 0, 1, 2 e 3.

O 8237 tem um conjunto de cinco registradores de controle/comando para programar e controlar a operação de DMA por um de seus canais (Tabela 7.2):

- **Comando**: o processador carrega esse registrador para controlar a operação do DMA. D0 habilita uma transferência de memória para memória, em que o canal 0 é usado para transferir um byte para um registrador temporário do 8237 e o canal 1 é usado para transferir o byte do registrador para a memória. Quando a transferência de memória para memória está habilitada, D1 pode ser usado para desativar o incremento/decremento no canal 0, de modo que um valor fixo pode ser gravado em um bloco de memória. D2 habilita ou desabilita o DMA.
- **Estado**: o processador lê esse registrador para determinar o estado do DMA. Os bits D0–D3 são usados para indicar se os canais 0–3 atingiram sua contagem final TC (do inglês, *Terminal Count*). Os bits D4–D7 são usados pelo processador para determinar se algum canal possui uma requisição de DMA pendente.
- **Modo**: o processador define esse registrador para determinar o modo de operação do DMA. Os bits D0 e D1 são usados para selecionar um canal. Os outros bits selecionam diversos modos de operação para o canal selecionado. Os bits D2 e D3 determinam se a transferência é de um dispositivo de E/S para a memória (escrita) ou da memória para a E/S (leitura), ou uma operação de verificação. Se D4 estiver definido em 1, então o registrador de endereço de memória e o registrador contador são recarregados com seus valores originais ao final de uma transferência de dados por DMA. Os bits D6 e D7 determinam o modo como o 8237 é utilizado. No modo único (*single*), um único byte de dados é transferido. Os modos bloco (*block*) e demanda (*demand*) são usados para uma transferência em bloco, com o modo demanda

permitindo o término prematuro da transferência. O modo cascata (*cascade*) permite que vários 8237s sejam dispostos em cascata, expandindo o número de canais para mais de 4.

- **Máscara única**: o processador define esse registrador. Os bits D0 e D1 selecionam o canal. O bit D2 limpa ou define em 1 o bit de máscara para esse canal. É através desse registrador que a entrada DREQ de um canal específico pode ser mascarada (desabilitada) ou desmascarada (habilitada). Enquanto o registrador *comando* pode ser usado para desabilitar o chip de DMA inteiro, o registrador *máscara única* permite que o programador desabilite ou habilite um canal específico.
- **Máscara total**: esse registrador é semelhante ao registrador máscara única, exceto que todos os canais podem ser mascarados ou desmascarados com uma operação de escrita.

Além disso, o 8237A tem oito registradores de dados: um registrador de endereço de memória e um contador para cada canal. O processador define esses registradores para indicar o local da memória principal a ser afetado pelas transferências.

Tabela 7.2

Registradores do Intel 8237A.

Bit	Comando	Estado	Modo	Máscara única	Máscara total
D0	H/D memória para memória	Canal 0 atingiu TC	Seleção de canal	Seleção de canal	Limpa/define em 1 o bit de máscara do canal 0
D1	H/D manutenção de endereço do canal 0	Canal 1 atingiu TC			Limpa/define em 1 o bit de máscara do canal 1
D2	H/D controlador	Canal 2 atingiu TC	Verificar/escrever/ler transferência	Limpa/define em 1 o bit de máscara	Limpa/define em 1 o bit de máscara do canal 2
D3	Temporização normal/ comprimida	Canal 3 atingiu TC		Não usado	Limpa/define em 1 o bit de máscara do canal 3
D4	Prioridade fixa/ rotativa	Requisição do canal 0	H/D de autoinicialização		Não usado
D5	Seleção de gravação adiada/ estendida	Requisição do canal 1	Seleção de incremento/ decremento de endereço		
D6	Percepção de DREQ ativo alto/ baixo	Requisição do canal 2			
D7	Percepção de DACK ativo alto/ baixo	Requisição do canal 3	Seleção de modo demanda/único/bloco/ cascata		

H/D = habilita/desabilita

TC = contagem final

7.6 ACESSO DIRETO À CACHE

O DMA provou ser um meio efetivo de aprimorar o desempenho da E/S com dispositivos periféricos e tráfego de E/S em redes. Contudo, para os aumentos significativos nas taxas de dados para a E/S em redes, ele não está apto a fazer escala para atender à crescente demanda. Essa demanda vem sobretudo da implantação generalizada de comutadores Ethernet de 10 Gbps e 100 Gbps para lidar com grandes quantidades de transferência de dados para e dos servidores de banco de dados e outros sistemas de alto desempenho (STALLINGS, 2014a).

Uma fonte secundária de tráfego, mas cada vez mais importante, é decorrente do WiFi com velocidade de gigabits. Os dispositivos de rede WiFi que lidam com 3,2 Gbps e 6,76 Gbps estão se tornando bastante disponíveis e produzindo demanda nos sistemas empresariais (STALLINGS, 2014b).

Nesta seção, vamos mostrar como habilitar a função de E/S para ter acesso direto à cache pode aprimorar o desempenho, uma técnica chamada de **acesso direto à cache (DCA)**. Durante esta seção, estamos preocupados somente com a cache que está mais próxima da memória principal, chamada de **cache de último nível**. Em alguns sistemas, será uma cache L2, em outros, uma cache L3.

Para começar, descrevemos o modo como sistemas multicore modernos usam cache compartilhada no chip para aprimorar o desempenho do DMA. Essa técnica envolve habilitar a função do DMA a fim de ter acesso direto à cache de último nível. A seguir, examinamos os aspectos de desempenho relacionados à cache que se manifestam quando o tráfego de rede de alta velocidade é processado. A partir daí, consideraremos algumas estratégias diferentes para o DCA que são designadas para melhorar o desempenho de processamento de protocolo de rede. Por fim, esta seção descreve uma técnica de DCA implementada pela Intel, conhecida como E/S de Dados Diretos.

DMA usando cache compartilhada de último nível

Como discutido no Capítulo 1 (veja Figura 1.2), os sistemas multicore modernos incluem tanto a cache dedicada a cada *core* como um nível adicional da cache compartilhada, seja L2 ou L3. Com o tamanho crescente da cache de último nível disponível, os desenvolvedores de sistema aprimoraram a função do DMA, de modo que o controlador de DMA tem acesso à cache compartilhada em uma maneira semelhante à dos *cores*. Para esclarecer a interação do DMA e da cache, será útil primeiro descrever uma arquitetura de sistema específica. Para esse propósito, apresentamos a seguir uma visão geral do sistema Xeon da Intel.

PROCESSADOR MULTICORE XEON Trata-se de uma família de processadores de tecnologia de ponta e de alto desempenho usada nos servidores, estações de trabalho de alto desempenho e supercomputadores. Alguns dos membros da família Xeon usam o sistema de interconexão em anel, como ilustrados para o Xeon E5-2600/4600 na Figura 7.16.

O E5-2600/4600 pode ser configurado com até oito *cores* em um chip único. Cada *core* tem caches dedicadas L1 e L2. Existe uma cache L3 de até 20 MB. A cache L3 é dividida em faixas, cada uma associada com cada *core*, embora cada um possa se referir a toda a cache. Além do mais, cada faixa tem seu próprio pipeline de cache, de tal maneira que os pedidos podem ser enviados em paralelo a essas faixas.

A interconexão em anel bidirecional de alta velocidade liga os *cores*, cache de último nível, PCIe e controlador integrado de memória (IMC).

Em essência, o anel opera da seguinte forma:

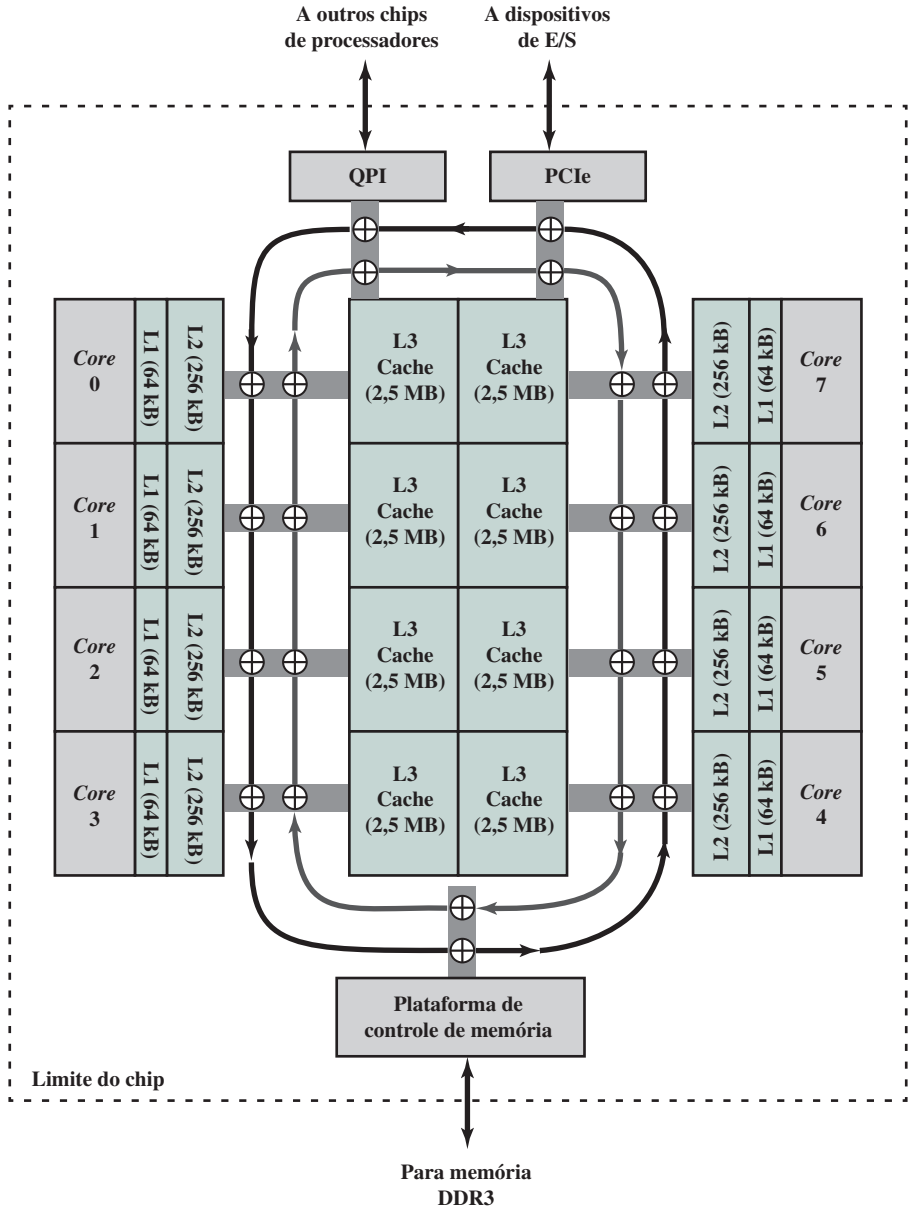
1. Cada componente que se fixa ao anel bidirecional (QPI, PCIe, cache L3 e cache L2) é considerado um agente de anel, e implementa a lógica do agente de anel.
2. Os agentes de anel cooperam por meio de um protocolo distribuído para solicitar e alocar acesso ao anel, na forma de slots de tempo.
3. Quando um agente tem dados a enviar, ele escolhe uma direção de anel que resulte em um caminho mais curto ao destino e transmite quando um escalonamento de um slot estiver disponível.

A arquitetura em anel proporciona bom desempenho e tem boa escala para os diversos *cores*, até um ponto. Para sistemas com grande número de *cores*, diversos anéis são usados, com cada anel suportando alguns *cores*.

USO DE DMA COM A CACHE Na operação tradicional de DMA, os dados são trocados entre a memória principal e o dispositivo de E/S por meio da estrutura de interconexão de sistema, como um barramento, anel ou matriz QPI ponto a ponto. Então, por exemplo, se o Xeon E5-2600/4600 tiver usado uma técnica de DMA tradicional, a saída se processaria da seguinte maneira. Um driver de E/S que está sendo executado em um *core* enviaria um comando de E/S ao controlador de E/S (PCIe na Figura 7.16) com o local e o tamanho do buffer na memória principal contendo os dados a serem transferidos. O controlador de E/S envia uma solicitação de leitura que é roteada à plataforma do controlador de memória (MCH), a qual acessa os dados na memória DDR3 e os coloca no anel de sistema para entrega ao controlador de E/S. A cache L3 não está envolvida nessa transação, e uma ou mais leituras de memória off-chip são requisitadas. De modo semelhante, para entrada, os dados chegam a partir do controlador de E/S e são entregues pelo anel de sistema à MCH e escritos na memória principal. A MCH deve também invalidar quaisquer linhas

Figura 7.16

Arquitetura de chip Xeon E5-2600/4600.



de cache L3 que correspondam aos locais de memória atualizados. Nesse caso, uma ou mais gravações de memórias off-chip são necessárias. Além disso, se uma aplicação quiser acessar os novos dados, uma leitura de memória principal é necessária.

Com a disponibilidade de grandes quantidades de caches de último nível, uma técnica mais eficiente é possível, e é usada pelo Xeon E5-2600/4600. Para saída, quando o controlador de E/S emite uma requisição de leitura, a MCH primeiro checa para ver se os dados estão na cache L3. Esse provavelmente é o caso se uma aplicação tiver recentemente escrito dados no bloco de memória que seria saída. Nesse caso, a MCH direciona os dados a partir da cache L3 ao controlador de E/S; não são necessários acessos à memória principal. No entanto, isso também faz com que os dados sejam retirados da cache, ou seja, o ato de ler por um dispositivo de E/S ocasiona que os dados sejam retirados. Desse modo, a operação de E/S procede de modo eficiente porque ela não requer acesso à memória principal. Todavia, se uma aplicação de fato precisar desses dados no futuro, eles devem ser lidos de volta na cache L3 a partir da memória principal. A operação de entrada no E5-2600/4600 opera como descrito no parágrafo anterior; a cache L3 não é envolvida. De tal maneira, o aprimoramento do desempenho envolve somente as operações de saída.

Um ponto final. Embora a transferência de saída seja diretamente da cache para o controlador de E/S, o termo *acesso direto à cache* não é usado para esse aspecto. Ainda, o termo é reservado para a aplicação de protocolo de E/S, como descrito no restante desta seção.

Aspectos do desempenho relacionado à cache

O tráfego de rede é transmitido na forma de uma sequência de blocos de protocolo, chamados de pacotes ou unidades de dados de protocolo. O menor protocolo de nível, ou de ligação, é geralmente o Ethernet, de modo que cada bloco de dados que chegue ou parta consiste em um pacote de Ethernet que contém o *payload* do pacote de protocolo de mais alto nível. Os protocolos de mais alto nível são em geral o protocolo de internet (IP — do inglês, *Internet Protocol*), que opera no topo da Ethernet, e o protocolo de controle de transmissão (TCP — em inglês, *Transmission Control Protocol*), que opera no topo do IP. Por conseguinte, o bloco da Ethernet consiste em um bloco de dados com um cabeçalho de TCP e um cabeçalho de IP. Para os dados de saída, os pacotes de Ethernet são formados em um componente periférico, como um controlador de E/S ou um controlador de interface de rede (NIC — em inglês, *Network Interface Controller*). De maneira similar, para o tráfego de entrada, o controlador de E/S retira da informação da Ethernet e entrega o pacote de TCP/IP à CPU hospedeira.

Tanto para o tráfego de saída como de entrada, o *core*, a memória principal e a cache estão envolvidos. Em um esquema de DMA, quando uma aplicação deseja transmitir dados, ela os coloca em um buffer de aplicação específica na memória principal. O *core* transfere isso para um buffer de sistema em uma memória principal e cria o TCP necessário e os cabeçalhos de IP, que também são bufferizados na memória de sistema. O pacote é, então, apanhado por DMA para transferir por NIC. Essa atividade não se dedica somente à memória principal, mas também à cache. Para o tráfego de entrada, as transferências similares entre o sistema e os buffers de aplicação são necessárias.

Quando grandes volumes de tráfego de protocolo são processados, dois fatores nesse cenário degradam o desempenho. Primeiro, o *core* consome ciclos de clock valiosos copiando dados entre o sistema e os buffers de aplicação. Em segundo lugar, por conta de as velocidades de memória não serem mantidas nas velocidades de CPU, o *core* perde tempo esperando por leituras e escritas de memória. Nessa via tradicional de processar o tráfego de protocolo, a cache não ajuda porque os dados e os cabeçalhos de protocolo estão constantemente mudando e, assim, a cache deve ser sempre atualizada.

Para esclarecer a questão do desempenho e explicar o benefício de DCA como uma maneira de aprimorar o desempenho, vamos analisar o processo de tráfego de protocolo em mais detalhes para o tráfego de entrada. Em termos gerais, ocorrem as seguintes etapas:

1. **Chegada de pacote**: o NIC recebe como entrada um pacote Ethernet. Ele processa e extrai informações de controle da Ethernet. Isso inclui fazer um cálculo de detecção de erro. O pacote restante de TCP/IP é, então, transferido ao módulo do DMA do sistema, que, em geral, é parte do NIC. O NIC também cria um descritor de pacote com informação acerca do pacote, como seu local de buffer na memória.
2. **DMA**: o módulo de DMA transfere dados, inclusive o descritor de pacote, para a memória principal. Ele deve também invalidar as linhas de cache correspondentes, se houve alguma.
3. **NIC interrompe o host**: depois que vários pacotes tiverem sido transferidos, o NIC emite uma interrupção ao processador do host.
4. **Cabeçalhos e descritores de recuperação**: o *core* processa a interrupção, invocando um processo de tratamento de interrupção, o qual lê o descritor e o cabeçalho dos pacotes recebidos.
5. **Ocorrências de falha de cache**: por ser a chegada de novos dados, as linhas de cache correspondentes ao sistema de buffer que contém novos dados são invalidadas. De tal maneira, o *core* deve parar de ler os dados a partir da memória principal na cache e, então, aos registradores de *core*.
6. **O cabeçalho é processado**: o software de protocolo é executado no *core* para analisar os conteúdos de TCP e de cabeçalho de IP. Isso provavelmente vai incluir o acesso ao bloco de controle de transporte (TCB), que contém informação de conteúdo relacionada ao TCP. O acesso ao TCB pode ou não causar uma falha de cache, necessitando de acesso de memória principal.
7. **Transferência do bloco**: a porção de dados do pacote é transferida a partir do buffer de sistema para o buffer de aplicação apropriado.

Uma sequência similar de etapas ocorre para o tráfego de saída de pacote, mas há algumas diferenças que afetam o modo como a cache é gerenciada. Para o tráfego de saída, ocorrem as seguintes etapas:

1. **Solicitação de transferência de pacote**: quando uma aplicação tem um bloco de dados para transferir a um sistema remoto, ela coloca os dados em um buffer de aplicação e alerta o SO com algum tipo de chamada do sistema.

2. **Criação do pacote**: o SO invoca um processo de TCP/IP a fim de criar o pacote de TCP/IP para transmissão. O processo do TCP/IP acessa o TCB (que pode envolver uma falha de cache) e cria cabeçalhos apropriados. Também lê dados a partir do buffer da aplicação e, então, coloca o pacote completo (cabeçalhos mais dados) em um buffer do sistema. Observe que os dados que são escritos no buffer do sistema também existem na cache. O processo de TCP/IP também cria um descritor de pacote que é colocado na memória compartilhada com o módulo de DMA.
3. **Chamada de uma operação de saída**: usa um programa de driver de dispositivo para sinalizar ao módulo de DMA que a saída está pronta para o NIC.
4. **Transferência de DMA**: o módulo de transferência de DMA lê o descritor de pacote, então a transferência de dados é realizada a partir da memória principal ou da cache de último nível ao NIC. Observe que as transferências de DMA invalidam a linha de cache em uma cache, mesmo no caso de uma leitura (pelo módulo de DMA). Se a linha estiver modificada, ela causa um *write back*. O *core* não faz as validações. As validações acontecem quando o módulo de DMA lê os dados.
5. **Sinais de NIC de finalização**: depois que uma transferência estiver completa, o NIC sinaliza ao driver no *core* que originou o sinal de envio.
6. **O driver libera buffer**: uma vez que o driver receba o aviso de finalização, ele libera espaço do buffer para reúso. O *core* deve também invalidar as linhas de cache que contêm dados de buffer.

Como pode ser visto, a E/S de rede envolve uma série de acessos à memória principal e à memória cache, além de movimentar os dados entre um buffer de aplicação e um sistema de buffer. O grande envolvimento da memória principal torna-se um gargalo, à medida que o desempenho tanto do *core* como da rede aceleram os ganhos nos tempos de acesso à memória.

Estratégias de acesso direto à cache

Várias estratégias têm sido propostas para fazer uso mais eficiente das caches para E/S de rede, com o termo geral *acesso direto à cache* aplicado para todas elas.

A mais simples é uma que foi implementada como um protótipo em uma série de processadores Xeon da Intel entre 2006 e 2010 (KUMAR, 2007, e INTEL CORP., 2008). Essa forma de DCA aplica-se apenas ao tráfego proveniente de rede. A função DCA no controlador de memória envia uma dica de pré-busca ao *core* tão logo os dados estejam disponíveis na memória do sistema. Isso habilita o *core* a fazer pré-busca de pacote de dados a partir do buffer do sistema, evitando, assim, falhas na cache e desperdícios relacionados aos ciclos do *core*.

Enquanto essa forma simples de DCA proporciona alguma melhoria, ganhos muito mais substanciais podem ser obtidos ao se evitar completamente o sistema de buffer na memória principal. Para a função específica do processamento do protocolo, observe que o pacote e a informação de descritor de pacote são acessados somente uma vez no buffer do sistema pelo *core*. Para pacotes recebidos, o *core* lê os dados a partir do buffer e transfere a carga útil do pacote a um buffer de aplicação. Ele não tem necessidade de acessar dados no buffer de sistema de novo. De maneira similar, para os pacotes de saída, uma vez que o *core* tenha colocado os dados no buffer de sistema, ele não precisa acessar os dados novamente. Supõe-se, portanto, que o sistema de E/S foi equipado não somente com uma capacidade de acessar diretamente a memória principal, mas também de acessar a cache, tanto para operações de entrada como de saída. Então, seria possível usar a cache de último nível em vez da memória principal para buffer dos pacotes e descritores de pacotes de entrada e de saída.

Essa última técnica, que é um DCA verdadeiro, foi proposta por Huggahalli (2005). Também tem sido descrita como **injeção de cache** (LEONARD, 2007). Uma versão dessa forma mais completa de DCA é implementada na linha de processador Xeon da Intel, conhecida como **E/S de Dados Diretos** (INTEL..., 2012).

E/S de Dados Diretos

A E/S de Dados Diretos da Intel (DDIO — em inglês, *Intel Direct Data I/O*) é implementada em toda a família Xeon E5 de processadores. Sua operação é mais bem explicada com a comparação lado a lado de transferências com e sem DDIO.

PACOTE DE ENTRADA Primeiro, analisamos o caso de um pacote que chega ao NIC a partir da rede. A Figura 7.17a mostra as etapas envolvidas na operação de DMA. A NIC inicia uma gravação na memória (1).

Então, o NIC invalida as linhas de cache que correspondem ao buffer de sistema (2). A seguir, a operação de DMA é realizada, depositando o pacote diretamente na memória principal (3). Por fim, depois de o *core* apropriado receber um sinal de interrupção de DMA, o *core* pode ler os pacotes de dados a partir da memória por meio da cache (4).

Antes de discutir o processamento do pacote de entrada usando DDIO, é preciso resumir a discussão da política de gravação de cache a partir do Capítulo 4, e apresentar uma nova técnica. Para a discussão a seguir, há aspectos relacionados à coerência da cache que surgem em um ambiente de multiprocessador ou multicore. Eles são discutidos no Capítulo 17, mas seus detalhes não precisam causar preocupação por ora. Vale lembrar que existem duas técnicas para lidar com a atualização para uma linha de cache:

- ***Write through***: todas as operações de gravação são feitas para a memória principal e também para a cache, assegurando que a memória principal seja sempre válida. Qualquer outro módulo *core*-cache pode monitorar o tráfego para a memória principal e manter a consistência dentro de sua própria cache local.
- ***Write back***: atualizações são feitas somente na cache. Quando ocorre uma atualização, um bit de atualização (*dirty bit*) associado à linha é definido. Então, quando um bloco é substituído, é feito um *write back* para a memória principal se, e somente se, o bit de atualização for definido.

A DDIO usa a estratégia de *write back* na cache L3.

Uma operação de gravação na cache pode encontrar uma falha de cache, a qual pode ser tratada por uma destas duas estratégias:

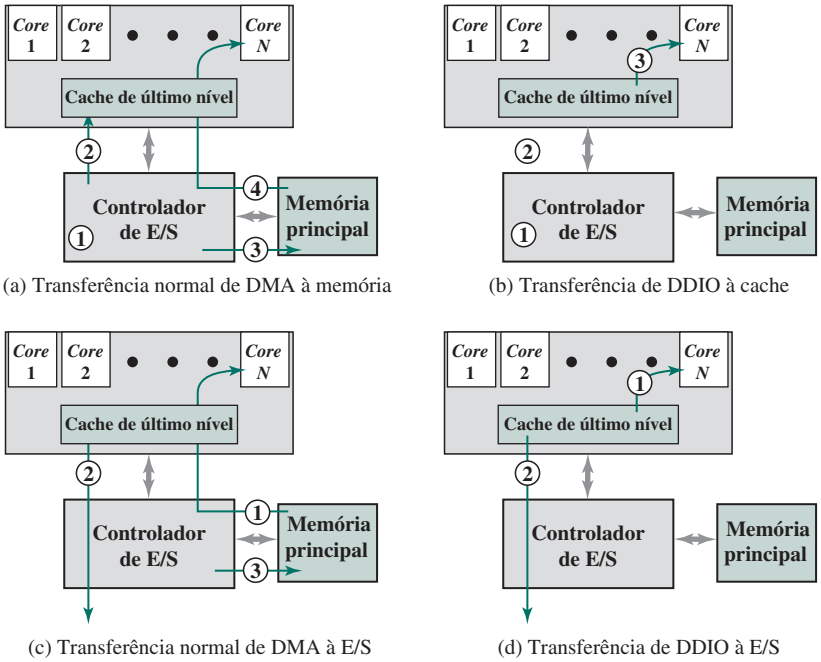
- ***Write allocate***: a linha requerida é carregada na cache a partir da memória principal. Então, a linha na cache é atualizada pela operação de escrita. Esse esquema é tipicamente usado com o método de *write back*.
- ***Non-write allocate***: o bloco é modificado na memória principal. Não há alteração feita na cache. Esse esquema é geralmente usado com o método de *write through*.

Considerando essas informações, descreveremos a estratégia de DDIO para transferências de entrada iniciadas pelo NIC.

1. Se houver um acerto de cache, a linha de cache é atualizada, mas não a memória principal; essa é simplesmente a estratégia de *write back* para o acerto de cache. A literatura da Intel refere-se a isso como **atualização de escrita**.
2. Se houver uma falha de cache, a operação de gravação ocorre para uma linha na cache que não vai fazer *write back* para a memória principal. As gravações subsequentes atualizam a linha de cache, novamente

Figura 7.17

Comparação de DMA e DDIO.



(a) Transferência normal de DMA à memória

(b) Transferência de DDIO à cache

(c) Transferência normal de DMA à E/S

(d) Transferência de DDIO à E/S

sem referência à memória principal ou sem ação futura que escreva esses dados na memória principal. A documentação da Intel (INTEL..., 2012) refere-se a isso como *write allocate*, que infelizmente não tem o mesmo significado para o termo na literatura geral da cache.

A estratégia de DDIO é efetivamente para a aplicação do protocolo de rede, porque os dados de entrada não precisam ser retidos para uso futuro. A aplicação do protocolo vai gravar os dados em um buffer de aplicação, e não há necessidade de armazenar temporariamente em um buffer de sistema.

A Figura 7.17b mostra a operação para a entrada de DDIO. O NIC inicia uma gravação na memória (1). Então, o NIC invalida as linhas de cache que correspondem a um buffer de sistema e deposita os dados de entrada na cache (2). Por fim, depois que o *core* apropriado recebe um sinal de interrupção de DCA, o *core* pode ler os dados de pacote a partir da cache (3).

PACOTE DE SAÍDA A Figura 7.17c mostra as etapas que envolvem a operação de DMA para transmissão de pacote de saída. O manuseio de protocolo de TCP/IP que é executado no *core* lê os dados a partir do buffer de aplicação e grava no buffer de sistema. Esses acessos de dados resultam em falhas de cache e ocasionam que os dados sejam lidos na memória e na cache L3 (1). Quando o NIC recebe a notificação para iniciar uma operação de transmissão, lê os dados a partir da cache e os transmite (2). O acesso da cache pelo NIC possibilita que os dados sejam desalojados a partir da cache e é feito *write back* na memória principal (3).

A Figura 7.17d mostra as etapas envolvidas em uma operação de DDIO para transmissão de pacote. O manuseio do protocolo de TCP/IP cria o pacote a ser transmitido e armazena no espaço alocado na cache L3 (1), mas não na memória principal (2). A operação de leitura iniciada pelo NIC é satisfeita pelos dados a partir da cache, sem causar retiradas na memória principal.

Deveria estar claro a partir dessas comparações lado a lado que a DDIO é mais eficiente que o DMA para pacotes tanto de entrada como de saída e, portanto, mais apta para manter a taxa de tráfego de pacote alta.

7.7 PROCESSADORES E CANAIS DE E/S

A evolução da função de E/S

Com a evolução dos sistemas de computação, tem havido um crescimento no padrão de complexidade e sofisticação dos componentes individuais. Em nenhum outro lugar isso é mais evidente do que na função de E/S. Já vimos parte dessa evolução. As etapas dessa evolução podem ser resumidas da seguinte forma:

1. A CPU controla diretamente o dispositivo periférico. Isso é visto em dispositivos simples controlados por microprocessador.
2. Um controlador ou módulo de E/S é acrescentado. A CPU usa a E/S programada sem interrupções. Com essa etapa, a CPU fica por fora dos detalhes específicos das interfaces do dispositivo externo.
3. A mesma configuração da etapa 2 é utilizada, mas agora as interrupções são empregadas. A CPU não precisa gastar tempo esperando que uma operação de E/S seja realizada, aumentando, assim, sua eficiência.
4. O módulo de E/S recebe acesso direto à memória, por meio de DMA. Ele agora pode mover um bloco de dados de ou para a memória sem envolver a CPU, exceto no início e no final da transferência.
5. O módulo de E/S é aprimorado para se tornar um processador por conta própria, com um conjunto especializado de instruções, ajustado para E/S. A CPU direciona o processador de E/S a executar um programa de E/S armazenado na memória. O processador de E/S busca e executa essas instruções sem intervenção da CPU. Isso permite que a CPU especifique uma sequência de atividades de E/S e seja interrompida somente quando a sequência inteira tiver sido executada.
6. O módulo de E/S tem uma memória local própria e, de fato, é um computador separado. Com essa arquitetura, um grande conjunto de dispositivos de E/S pode ser controlado, com o mínimo de envolvimento da CPU. Um uso comum para essa arquitetura tem sido no controle da comunicação com terminais interativos. O processador de E/S cuida da maior parte das tarefas envolvidas no controle dos terminais.

Enquanto se prossegue nesse caminho de evolução, cada vez mais a função de E/S é realizada sem envolvimento da CPU. A CPU fica cada vez mais livre das tarefas relacionadas a E/S, melhorando o desempenho. Com as duas últimas etapas (5-6), ocorre uma grande mudança com a introdução do conceito de um módulo de E/S capaz de executar um programa. Para a etapa 5, o módulo de E/S em geral é conhecido como um *canal de E/S*.

Para a etapa 6, o termo *processador de E/S* é muitas vezes usado. Contudo, os dois termos ocasionalmente são aplicados às duas situações. No texto seguinte, usaremos o termo *canal de E/S*.

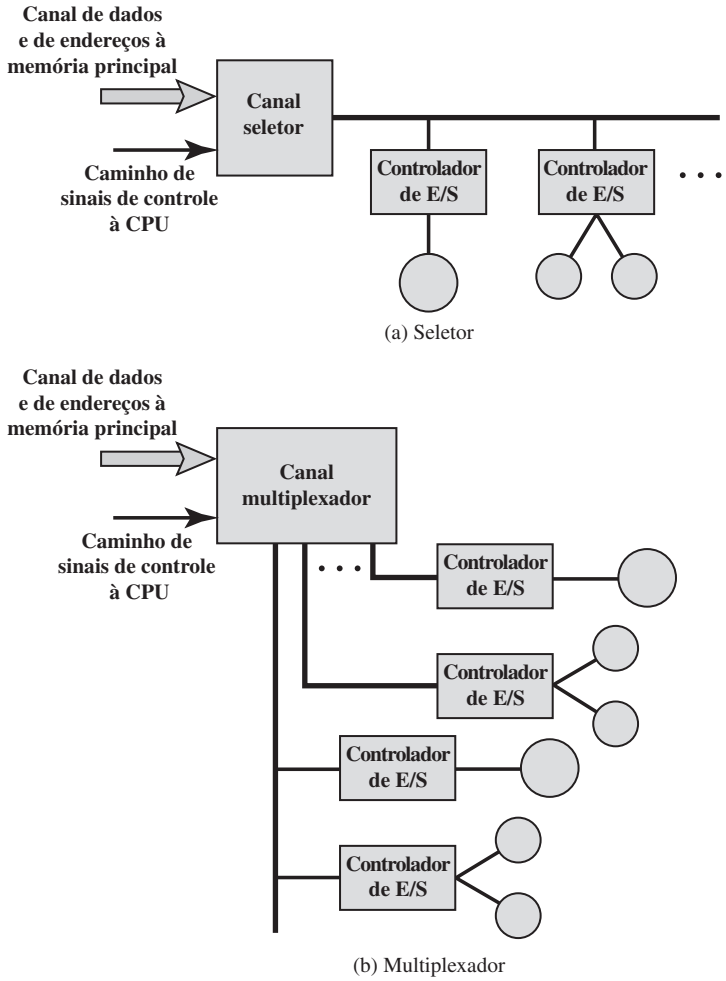
Características dos canais de E/S

O canal de E/S representa uma extensão do conceito de DMA. Um canal de E/S tem a capacidade de executar instruções de E/S, o que lhe oferece um controle completo sobre as operações de E/S. Em um sistema de computação com esses dispositivos, a CPU não executa instruções de E/S. Essas instruções são armazenadas na memória principal para serem executadas por um processador de uso específico no próprio canal de E/S. Desse modo, a CPU inicia uma transferência de E/S instruindo o canal de E/S a executar um programa na memória. O programa especificará o dispositivo ou os dispositivos, a área ou as áreas da memória para armazenamento, as prioridades e ações a serem tomadas para certas condições de erro. O canal de E/S segue essas instruções e controla a transferência de dados.

Dois tipos de canais de E/S são comuns, conforme ilustramos na Figura 7.18. Um *canal seletor* controla diversos dispositivos de alta velocidade e, a qualquer momento, é dedicado à transferência de dados com um desses dispositivos. Assim, o canal de E/S seleciona um dispositivo e efetua a transferência de dados. Cada dispositivo, ou pequeno grupo de dispositivos, é tratado por um *controlador*, ou módulo de E/S, que é semelhante aos módulos de E/S que discutimos até aqui. Desse modo, o canal de E/S atua no lugar da CPU para controlar esses controladores de E/S. Um *canal multiplexador* pode tratar da E/S com vários dispositivos ao mesmo tempo. Para dispositivos de baixa velocidade, um *multiplexador de byte* aceita ou transmite caracteres o mais rápido

Figura 7.18

Arquitetura de canal de E/S.



possível a diversos dispositivos. Por exemplo, o fluxo de caracteres resultante de três dispositivos com diferentes velocidades e fluxos individuais $A_1A_2A_3A_4$ ..., $B_1B_2B_3B_4$ ..., e $C_1C_2C_3C_4$ ... poderia ser $A_1B_1C_1A_2C_2A_3B_2C_3A_4$, e assim por diante. Para dispositivos de alta velocidade, um *multiplexador de bloco* intercala os blocos de dados de vários dispositivos.

7.8 PADRÕES DE INTERCONEXÃO EXTERNA

Nesta seção, proporcionaremos uma breve visão geral dos padrões de interface externa mais amplamente usados para dar suporte a E/S. Dois deles, Thunderbolt e InfiniBand, são examinados em detalhes no Apêndice J (disponível em inglês na Sala Virtual).

Barramento serial universal (USB)

O USB (do inglês, *Universal Serial Bus*) é bastante usado para conexões periféricas. É a interface padrão para dispositivos de velocidade mais lenta, como teclado e dispositivos apontadores, mas também é comumente usada para E/S de alta velocidade, incluindo impressoras, drives de disco e adaptadores de rede.

Ele vem de várias gerações. A primeira versão, USB 1.0, definiu uma taxa de dados de *baixa velocidade* de 1,5 Mbps e uma taxa de *velocidade completa* de 12 Mbps. O USB 2.0 proporciona uma taxa de dados de 480 Mbps. O USB 3.0 inclui um novo barramento de maior velocidade chamado de *SuperSpeed* em paralelo com o barramento do USB 2.0. A velocidade de sinalização do Super-Speed é de 5 Gbps, mas, em razão da sobrecarga de sinalização, a taxa de dados úteis é de até 4 Gbps. A mais recente especificação é USB 3.1, que inclui o modo de transferência mais rápido, chamado de *SuperSpeed+*. Esse modo de transferência atinge uma taxa de sinalização de 10 Gbps e uma taxa de dados teóricos úteis de 9,7 Gbps.

O sistema USB é controlado por um controlador host central, que é conectado aos dispositivos para criar uma rede local com uma topologia hierárquica em árvore.

Barramento serial FireWire

O FireWire foi desenvolvido como uma alternativa para a interface SCSI (do inglês, *Small Computer System Interface*), a fim de ser usado em sistemas menores, como computadores pessoais, estações de trabalho e servidores. O objetivo foi atender às demandas crescentes por taxas de E/S mais altas nesses sistemas, enquanto evita as tecnologias de canais de E/S robustas e caras desenvolvidas para os sistemas baseados em mainframes e em supercomputadores. O resultado é o padrão IEEE 1394, para um barramento serial de alto desempenho (*high performance serial bus*), em geral conhecido como FireWire.

O FireWire usa uma configuração *daisy-chain*, com até 63 dispositivos conectados em uma única porta. Além do mais, até 1.022 barramentos FireWire podem ser interconectados usando pontes, permitindo que um sistema aceite tantos periféricos quantos forem necessários.

O FireWire permite o que é conhecido como conexão a quente (*hot plugging*), que significa que é possível conectar e desconectar periféricos sem ter que desligar o sistema de computação ou reconfigurar o sistema. Além disso, FireWire permite configuração automática; não é necessário definir manualmente IDs de dispositivo ou se preocupar com a posição relativa dos dispositivos. Com FireWire, não existem terminações, e o sistema executa automaticamente uma função de configuração para atribuir endereços. Observe também que um barramento FireWire não precisa ser rigorosamente uma *daisy chain*. Em vez disso, é possível usar uma configuração estruturada em forma de árvore.

Um recurso importante do padrão FireWire é que ele especifica um conjunto de três camadas de protocolos para padronizar o modo como o sistema host interage com os dispositivos periféricos pelo barramento serial. A camada física define os meios de transmissão que são permitidos pelo FireWire e as características elétricas e de sinalização de cada um. Taxas de dados de 25 Mbps a 3,2 Gbps são definidas. A camada de ligação descreve a transmissão de dados em pacotes. A camada de transação define um protocolo de requisição e resposta que esconde os detalhes de camada mais baixa do FireWire das aplicações.

Small Computer System Interface (SCSI)

A interface SCSI é um padrão comum para conectar dispositivos periféricos (discos, modems, impressoras etc.) a computadores pequenos e médios. Embora envolva taxas de dados mais altas, ela perdeu popularidade para competidores como USB e FireWire em sistemas menores. No entanto, as versões de alta velocidade da SCSI permanecem populares para o suporte à memória de massa em sistemas empresariais. Por exemplo, o zEnterprise EC12 da IBM e outros mainframes da IBM oferecem suporte para a SCSI, e uma série de sistemas Seagate de drive de disco rígido usa SCSI.

A organização física da SCSI é um barramento compartilhado, que pode suportar até 16 ou 32 dispositivos, dependendo da geração do padrão. O barramento proporciona transmissão paralela em vez de serial, com uma largura de barramento de 16 bits nas primeiras gerações e 32 bits nas últimas gerações. A velocidade varia de 5 Mbps na especificação original de SCSI-1 a 160 Mbps na SCSI-3 U3.

Thunderbolt

A mais recente, e uma das mais rápidas, tecnologia de conexão de periféricos a se tornar disponível para uso de propósito geral é o Thunderbolt, desenvolvido pela Intel em colaboração com a Apple. Um cabo Thunderbolt pode gerenciar o trabalho anteriormente requerido de diversos cabos. A tecnologia combina dados, vídeos, áudios e energia em uma única conexão de alta velocidade para periféricos como drives de disco rígidos, conjuntos de RAID (*Redundant Array of Independent Disks*), caixas de captura de vídeo e interfaces de rede. Isso proporciona uma taxa de transferência de até 10 Gbps em cada direção e até 10 watts de potência aos periféricos conectados.

O Thunderbolt é descrito em detalhes no Apêndice J.

InfiniBand

InfiniBand é uma especificação de E/S, voltada para o mercado de servidores de ponta. A primeira versão da especificação foi lançada no início de 2001, atraindo vários fornecedores. Por exemplo, a série de mainframes zEnterprise da IBM se apoia fortemente no InfiniBand há vários anos. O padrão descreve uma arquitetura e especificações para o fluxo de dados entre os processadores e dispositivos de E/S inteligentes. InfiniBand tornou-se uma interface popular para redes de armazenamento de dados e outras configurações de armazenamento de grande capacidade. Basicamente, o InfiniBand permite que servidores, armazenamento remoto e outros dispositivos de rede sejam conectados em uma central de comutadores e links. A arquitetura baseada em comutador pode conectar até 64.000 servidores, sistemas de armazenamento e dispositivos de rede.

O InfiniBand é descrito em detalhes no Apêndice J.

PCI Express

O PCI Express é um sistema de barramento de alta velocidade que conecta periféricos de uma grande variedade de tipos e velocidades. O Capítulo 3 discute o PCI Express de modo detalhado.

SATA

Serial ATA (*Serial Advanced Technology Attachment*) é uma interface para sistemas de armazenamento de disco. Proporciona taxas de dados de até 6 Gbps, com um máximo por dispositivo de 300 Mbps. O SATA é bastante usado em computadores desktop, bem como em aplicações industriais e embarcadas.

Ethernet

É uma tecnologia de rede predominantemente com fios, usada em casas, escritórios, centros de dados, empresas e redes de área ampla. Conforme evoluiu para suportar taxas de dados de até 100 Gbps e distâncias de poucos metros a dezenas de quilômetros, tornou-se essencial para computadores pessoais, estações de trabalho, servidores e dispositivos de armazenamento massivo de dados em grandes e pequenas organizações.

A Ethernet começou como um sistema experimental baseado em barramento de 3 Mbps. Com um sistema de barramento, todos os dispositivos relacionados, como PCs, conectam-se a um cabo coaxial comum, bem como sistemas residenciais de TV a cabo. A primeira Ethernet comercialmente disponível, e a primeira versão do IEEE 802.3, foram os sistemas baseados em barramento que operavam em 10 Mbps. Conforme a tecnologia avançou, a Ethernet mudou de baseada em barramento para baseada em comutação, e a taxa de dados aumentou periodicamente por uma ordem de magnitude. Com os sistemas baseados em comutação, há um comutador central, com todos os dispositivos conectados diretamente a ele. Atualmente, os sistemas de Ethernet estão disponíveis em velocidades de até 100 Gbps. Oferecemos aqui uma breve cronologia.

- 1983: 10 Mbps (megabits por segundo, milhões de bits por segundo).
- 1995: 100 Mbps.
- 1998: 1 Gbps (gigabits por segundo, bilhões de bits por segundo).
- 2003: 10 Gbps.
- 2010: 40 Gbps e 100 Gbps.

Wi-Fi

O Wi-Fi é uma tecnologia de acesso à internet predominantemente sem fio, usado em casas, escritórios e espaços públicos. O Wi-Fi em casa agora conecta computadores, tablets, smartphones e hosts de dispositivos eletrônicos, como câmeras de vídeo, TVs e termostatos. O Wi-Fi nas empresas tem se tornado um meio essencial para aumentar a produtividade dos colaboradores e a efetividade da rede. E os hotspots de Wi-Fi público expandiram-se de modo significativo para proporcionar acesso livre à internet em locais públicos.

Como a tecnologia de antenas, as técnicas de transmissão sem fio e projeto de protocolo sem fio evoluíram, o comitê IEEE 802.11 ficou apto a apresentar padrões para novas versões de Wi-Fi em velocidades ainda maiores. Uma vez que o padrão seja publicado, a indústria rapidamente desenvolve os produtos. Apresentamos aqui uma breve cronologia, começando com o padrão original, que é chamado simplesmente de IEEE 802.11, e mostrando a taxa de dados máxima para cada versão:

- 802.11 (1997): 2 Mbps (megabits por segundo, milhões de bits por segundo).
- 802.11a (1999): 54 Mbps.
- 802.11b (1999): 11 Mbps.
- 802.11n (1999): 600 Mbps.
- 802.11g (2003): 54 Mbps.
- 802.11ad (2012): 6,76 Gbps (bilhões de bits por segundo).
- 802.11ac (2014): 3,2 Gbps.

7.9 ESTRUTURA DE E/S DO zENTERPRISE EC12 DA IBM

O zEnterprise EC12 é o mais recente lançamento de computador mainframe da IBM (isto no momento em que este livro foi escrito). O sistema é baseado no uso do chip de processador zEC12, que é um chip multicore de 5,5-GHz com seis *cores*. A arquitetura do zEC12 pode ter um máximo de 101 chips de processador para um total de 606 *cores*. Nesta seção, analisaremos a estrutura de E/S do zEnterprise EC12.

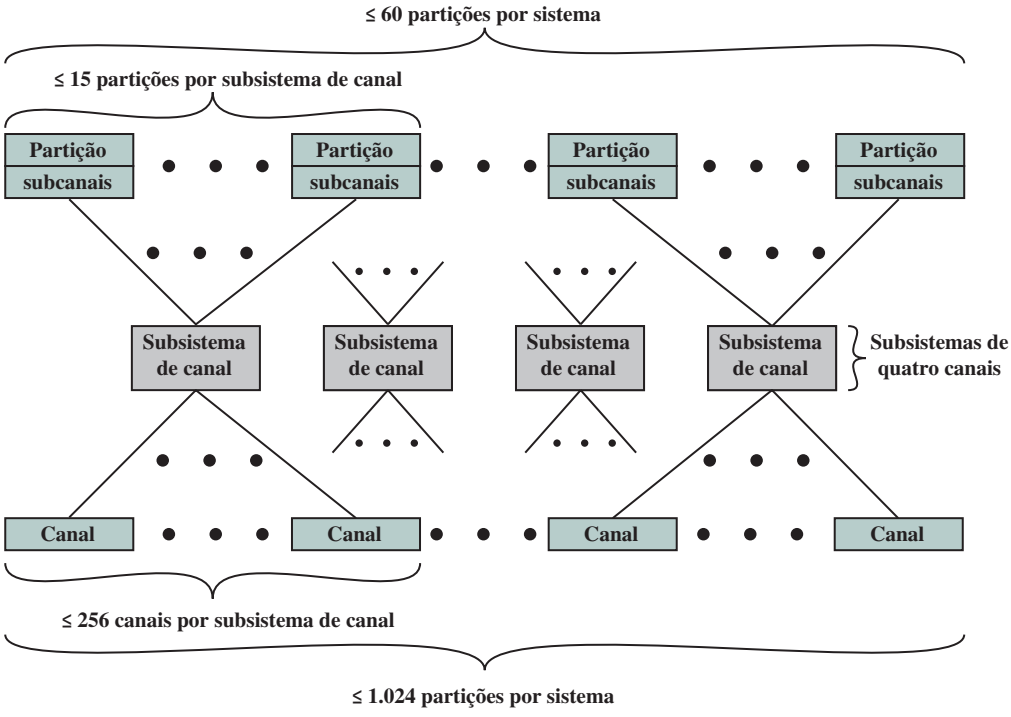
Estrutura do canal

O zEnterprise EC12 tem um subsistema de E/S dedicado que gerencia todas as operações, descarregando completamente essa carga de processamento e de memória dos processadores principais. A Figura 7.19 mostra a estrutura lógica do subsistema de E/S. Dos 96 processadores *core*, até 4 deles podem ser dedicados para o uso de E/S, criando 4 **subsistemas de canais (CSS — do inglês, *Channel Subsystems*)**. Cada CSS é constituído pelos seguintes elementos:

- **Processador de assistência de sistema (SAP — do inglês, *System Assist Processor*)**: o SAP é um processador central configurado para operação de E/S. Seu papel é descarregar as operações de E/S e gerenciar canais e filas de operações de E/S. Ele alivia os outros processadores de tarefas de E/S, possibilitando que se dediquem à lógica de aplicação.

Figura 7.19

Estrutura de subsistema de canal de E/S IBM zEC12.



- **Área de sistema de hardware (HSA — do inglês, *Hardware System Area*)**: a HSA é uma parte reservada da memória do sistema que contém a configuração de E/S. É usada pelos SAPs. Uma quantidade fixa de 32 GB é reservada, que não é parte da memória comprada pelo consumidor. Isso proporciona maior flexibilidade de configuração e maior disponibilidade para eliminar interrupções planejadas ou pré-planejadas.
- **Partições lógicas**: uma partição lógica é uma forma de máquina virtual, que é, em essência, um processador lógico definido no nível do sistema operacional.[3] Cada CSS suporta até 16 partições lógicas.
- **Subcanais**: um subcanal aparece a um programa como um dispositivo lógico e contém a informação necessária para realizar a operação de E/S. Um subcanal existe para cada dispositivo de E/S endereçável pelo CSS. Um subcanal é usado pela execução do código de subsistema de canal em uma partição para passar uma requisição de E/S ao subsistema de canal. Um subcanal é assinalado a cada dispositivo definido para a partição lógica. Até 196k subcanais são suportados pelo CSS.
- **Caminho de canal**: um caminho de canal é uma interface única entre o subsistema de canal e uma ou mais unidades de controle, por meio de um canal. Os comandos e dados são enviados através de um caminho de canal para realizar uma requisição de E/S. Cada CSS pode ter até 256 caminhos de canal.
- **Canal**: os canais são pequenos processadores que se comunicam com as unidades de controle (UCs) de E/S. Eles gerenciam a transferência de dados entre a memória e os dispositivos externos.

Essa estrutura elaborada possibilita que o mainframe gerencie um grande número de dispositivos de E/S e de ligações de comunicação. Todo o processamento de E/S é descarregado a partir da aplicação e dos processadores do servidor, melhorando o desempenho. Esses processadores de subsistema de canal são de algum modo gerais em configuração, habilitando-os a gerenciar um amplo leque de tarefas de E/S e a manter os crescentes requisitos. Esses processadores de canal são especificamente programados para as unidades de controle de E/S com as quais eles interfaceiam.

3 Uma máquina virtual é uma instância de um sistema operacional ao longo de uma ou mais aplicações executadas em uma partição de memória isolada dentro do computador. Isso possibilita que diferentes sistemas operacionais sejam executados em um mesmo computador ao mesmo tempo, bem como previne que as aplicações interfiram umas com as outras.

Organização de sistema de E/S

Para explicar a organização do sistema de E/S, é preciso primeiro explicar brevemente o layout físico do zEnterprise EC12. A Figura 7.20 apresenta uma visão frontal da versão refrigerada a água (há também uma versão refrigerada a ar). O sistema tem as seguintes características:

- Peso: 2.430 kg (5.358 lbs).
- Largura: 1,568 m (5,14 ft).
- Profundidade: 1,69 m (6,13 ft).
- Altura: 2,015 m (6,6 ft).

Não exatamente um laptop.

O sistema consiste em dois compartimentos, chamados de frames, que hospedam vários componentes da zEnterprise EC12. Um frame A do lado direito inclui duas grandes gaiolas, mais espaço para cabos e outros componentes. A gaiola superior é uma gaiola de processadores, com quatro encaixes para alojar quatro *books* (sistema de acondicionamento de unidades da IBM) de processadores que são totalmente interconectados. Cada *book* contém um módulo de multichip (MCM), cartões de memória e conexões de E/S da gaiola. Cada MCM é uma placa que aloja seis chips multicore e dois chips de controle de armazenamento.

A gaiola inferior no frame A é uma gaiola de E/S, que contém hardware de E/S, inclusive multiplexadores e canais. A gaiola de E/S é uma unidade fixa instalada pela IBM na fábrica, para as especificações do consumidor.

O frame Z, do lado esquerdo, contém baterias internas e fontes de tensão e espaço para um ou mais elementos de suporte, que são usados por um gerenciador de sistema para gerenciamento da plataforma. O frame Z também contém encaixes para duas ou mais gavetas de E/S.

Uma gaveta de E/S contém componentes similares aos da gaiola de E/S. As diferenças são que as gavetas são menores e mais fáceis de serem colocadas e retiradas no local do cliente para atender aos pedidos de alteração.

Com esse fundamento, mostraremos agora uma configuração típica da estrutura do sistema de E/S zEnterprise EC12 (Figura 7.21). Cada *book* de processadores zEC12 suporta duas infraestruturas internas de E/S (ou seja, internas aos frames A e Z): InfiniBand para gaiolas de E/S e gavetas de E/S, bem como PCI Express (PCIe) para gavetas de E/S. Esses controladores de canal são chamados de ***fanouts***.

As conexões do InfiniBand do *book* de processadores com as gaiolas de E/S são feitas por meio de um *fanout* do adaptador de canal do host (HCA — do inglês, *Host Channel Adapter*), que tem ligações de InfiniBand para multiplexadores InfiniBand na gaveta ou gaiola de E/S. Os multiplexadores de InfiniBand são usados para servidores de interconexão, equipamento de infraestrutura de comunicação, armazenamento e sistemas embarcados. Além de usar InfiniBand para sistemas de interconexão, todos os quais usam InfiniBand, o multiplexador

Figura 7.20

Frames de E/S do zEC12 da IBM — vista frontal.

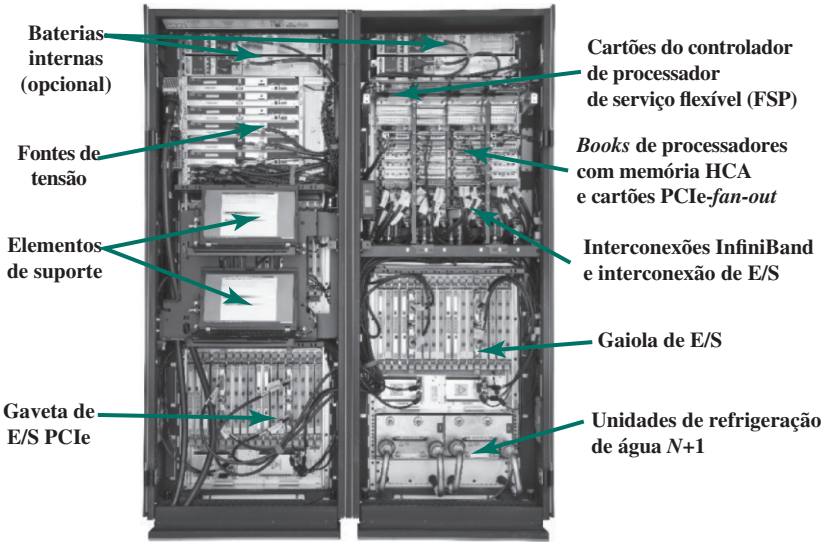
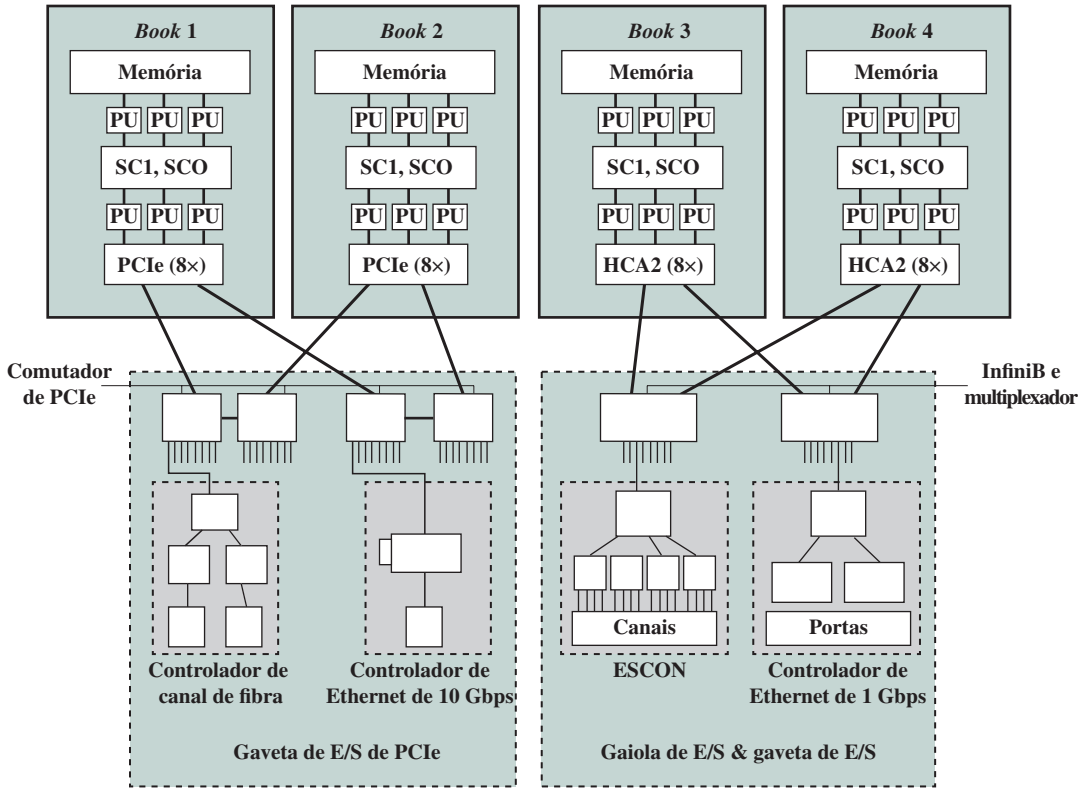


Figura 7.21
Estrutura do sistema de E/S zEC12 da IBM.



de InfiniBand suporta outras tecnologias de E/S. A ESCON (*Enterprise Systems Connection*) suporta conectividade a discos, fitas e impressoras que usam tecnologia proprietária baseada em fibras. As conexões de Ethernet proporcionam conexões de 1 e 10 Gbps para uma gama de dispositivos que suportam essa tecnologia de rede local. Um uso digno de nota da Ethernet é construir servidores locais maiores, em particular para interconectar servidores blade um com o outro e com outros mainframes.[4]

As conexões de PCIe a partir do *book* de processadores até as gavetas de E/S se dão por meio do PCIe *fanout* para os comutadores de PCIe. Os comutadores de PCIe podem conectar uma gama de controladores de dispositivos de E/S. Exemplos comuns de zEnterprise EC12 são Ethernet de 1 Gbps e 10 Gbps e canal de fibra.

Cada *book* contém uma combinação de até 8 InfiniBand HCA e PCIe *fanout*s. Cada *fanout* suporta até 32 conexões, para um total de 256 conexões por *book* de processadores, cada conexão sendo controlada por um processador de canal.

7.10 TERMOS-CHAVE, QUESTÕES DE REVISÃO E PROBLEMAS

Acesso direto à cache (DCA), 215	Canal seletor, 221	E/S mapeada na memória, 200
Acesso direto à memória (DMA), 199	Comando de E/S, 199	E/S programada, 198
Atualização de escrita, 219	Dispositivo periférico, 195	E/S serial, 222
Cache de último nível, 215	E/S controlada por interrupção, 201	InfiniBand, 223
Canal de E/S, 198	E/S de Dados Diretos, 218	Injeção de cache, 218
Canal multiplexador, 221	E/S independente, 200	Interrupção, 199

4 Um servidor blade é uma arquitetura de servidor que hospeda diversos módulos de servidor (*blades*) em um único chassi. É bastante usado em centros de dados para economizar espaço e melhorar o gerenciamento do sistema. Seja independente ou montado em rack, o chassi proporciona a fonte de tensão, e cada lâmina tem sua própria CPU, memória e disco rígido.