1.Introdução

Uma das principais funções de um sistema operacional é controlar todos os dispositivos de E/S do computador. Ele precisa enviar comandos para os dispositivos, capturar interrupções e tratar de erros. Também deve fornecer uma interface simples e fácil de usar entre os dispositivos e o restante do sistema.

2.Princípios do hardware de E/S

Os dispositivos de entrada e saída (E/S) podem ser divididos em duas categorias que são ***dispositivos de bloco*** e ***dispositivos de caractere***. Um dispositivo de bloco armazena informações em blocos de tamanho fixo, cada um com seus próprios endereços. Os tamanhos de bloco comuns variam de 512 até 32.768 bytes. A propriedade fundamental de um dispositivo de bloco é que ele pode ler ou escrever cada bloco independentemente de todos os outros. Os discos são os dispositivos de bloco mais comuns.

Um dispositivo de caractere envia ou aceita um fluxo de caracteres, sem considerar nenhuma estrutura de bloco. Ele não é endereçável e não tem nenhuma operação de busca. As impressoras, interfaces de rede, mouses e a maioria dos outros dispositivos que não são do tipo disco podem ser vistos como dispositivos de caractere.

Essa classificação não é perfeita, alguns dispositivos simplesmente não se encaixam. Os relógios, por exemplo, não são endereçáveis por bloco. Tampouco eles geram ou aceitam fluxos de caracteres. Tudo que eles fazem é causar interrupções em intervalos bem definidos.

O sistema de arquivos, por exemplo, trata somente com dispositivos de bloco abstratos e deixa a parte dependente de dispositivo para um software de nível mais baixo, chamado ***driver de dispositivo***.

Os dispositivos de E/S têm uma variação enorme em suas velocidades, o que impõe uma pressão considerável no software para funcionar bem com diferentes taxas de dados.

3.Controladoras de dispositivos

Normalmente, as unidades de E/S consistem em um componente mecânico e um componente eletrônico. Frequentemente é possível separar as duas partes para fornecer um projeto mais modular e geral. O componente eletrônico é chamado de ***controladora de dispositivos*** ou ***adaptador***. Nos computadores pessoais, ele frequentemente assume a forma de uma placa de circuito impresso que pode ser inserida em um *slot* de expansão.

Muitas controladoras podem controlar 4, 8 ou até mais dispositivos idênticos. Se a interface entre a controladora e o dispositivo for padronizada, como uma interface ANSI, IEEE ou um padrão ISO oficial, ou de fato, então as empresas poderão fazer controladores ou dispositivos que se encaixem nessa interface.

Frequentemente, a interface entre a controladora e o dispositivo é de baixo nível. O que sai da unidade de disco é um fluxo serial de bits, começando com um ***preâmbulo***, seguido dos 4096 bits de um setor (512 x 8) e, finalmente, uma soma de verificação, também chamado de Código de Correção de Erros (ECC). O preâmbulo é gravado quando o disco é formatado e contém o número do cilindro e do setor, o tamanho do setor e dados similares.

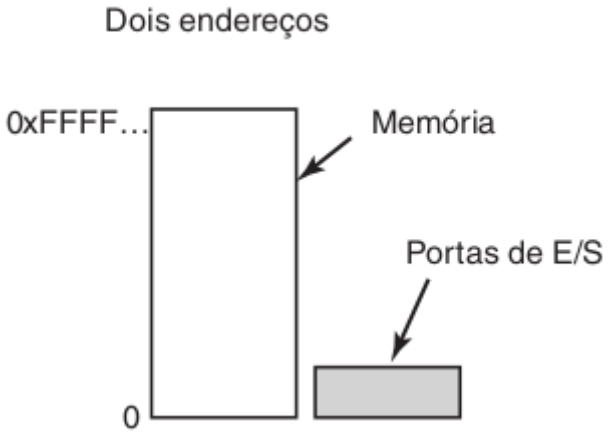
A tarefa da controladora é converter o fluxo serial de bits em um bloco de bytes e realizar toda a correção de erro necessária. Normalmente, o bloco de bytes é primeiramente montado, bit a bit, em um *buffer* dentro da controladora. Depois que sua soma de verificação tiver sido verificada e o bloco declarado como livre de erros, ele poderá então ser copiado na memória principal.

A controladora de um monitor também funciona como um dispositivo serial de bits, em um nível igualmente baixo. Ela lê na memória os bytes que contêm os caracteres a serem exibidos e gera os sinais usados para modular o feixe de elétrons do tubo de raios catódicos (CRT). A controladora também gera os sinais para fazer um feixe CRT realizar o retraço horizontal, após a tela inteira ter sido varrida. Com ela, o sistema operacional inicializa a controladora com alguns parâmetros, como o número de caracteres ou pixels por linha e o número de linhas por tela, e deixa que ela se encarregue de fazer a exibição.

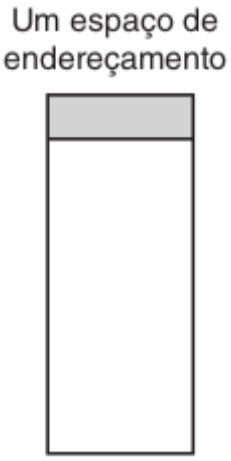
4.E/S mapeada em memória

Cada controladora tem alguns registradores utilizados para comunicação com a CPU. Escrevendo nesses registradores, o sistema operacional pode fazer o dispositivo enviar dados, aceitar dados, ligar-se ou desligar-se ou executar alguma outra ação. Lendo esses registradores, o sistema operacional pode saber qual é o estado do dispositivo, se ele está pronto para aceitar um novo comando e afins. Além dos registradores de controle, muitos dispositivos possuem um *buffer* de dados que o sistema operacional pode ser e escrever.

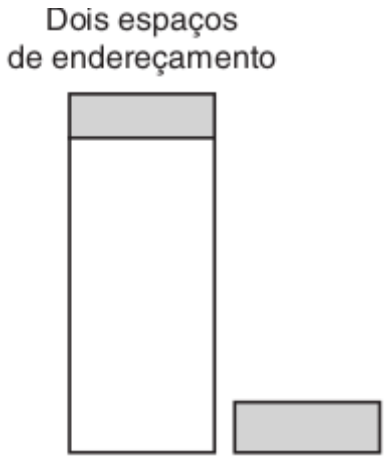
Surge o problema de como a CPU se comunica com os registradores de controle e com os *buffers* de dados dos dispositivos. Existem duas alternativas. Na primeira estratégia, cada registrador de controle recebe um número de ***porta de E/S***, um valor inteiro de 8 ou de 16 bits. Usando uma instrução de E/S especial como: IN REG, PORT, a CPU pode ler o registrador de controle PORT e armazenar o resultado no registrador REG da CPU. Analogamente, usando: OUT PORT, REG, a CPU pode escrever o conteúdo de REG em um registrador de controle.



Em outros computadores, os registradores de E/S fazem parte do espaço de endereçamento normal da memória. Esse esquema é chamado de ***E/S mapeada em memória***.



Normalmente, os endereços atribuídos estão no topo do espaço de endereçamento. Um esquema misto, com *buffers* de dados de E/S mapeados em memória e portas de E/S separados para os registradores de controle são mostrados abaixo.



Em todos os casos, quando a CPU quer ler uma palavra, ou da memória ou de uma porta de E/S, ela coloca o endereço necessário nas linhas de endereço do barramento, e então, envia um sinal READ em uma linha de controle do barramento. Uma segunda linha de sinal é usada para dizer se é necessário espaço de E/S ou espaço de memória. Se for espaço de memória, a memória responderá a requisição. Se for espaço de E/S, é o dispositivo de E/S que responderá. Se houver apenas espaços de memória, todo o módulo de memória e todo dispositivo de E/S compara as linhas de endereço com o intervalo de endereços que ele atende. Se o endereço cair em seu intervalo, ele responderá ao seu pedido.

4.Interrupções

Normalmente os registradores da controladora têm um ou mais ***bits de status*** que podem ser testados para determinar se uma operação de saída está concluída ou se novos dados estão disponíveis em um dispositivo de entrada. Uma CPU pode executar um laço, sempre testando um bit de status, até que um dispositivo esteja pronto para aceitar ou fornecer novos dados. Isso é chamado de ***consulta sequencial*** (*polling*) ou ***espera ativa*** (*busy wait*).

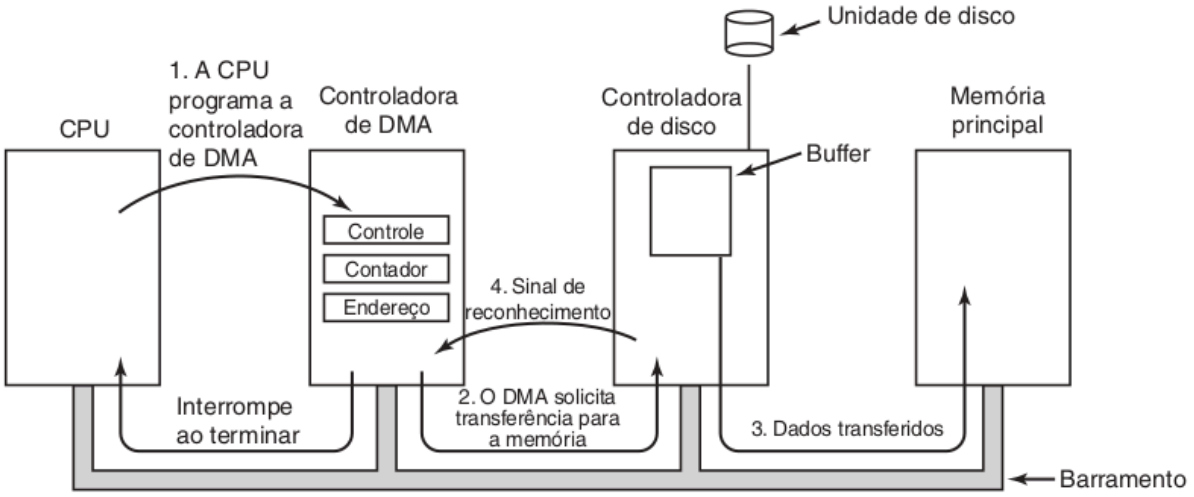
No âmbito da E/S, onde você pode ter de esperar por um tempo muito longo para que o mundo externo aceite ou produza dados, a consulta sequencial não é aceitável, exceto para sistemas dedicados muito pequenos, que não executam múltiplos processos.

Além dos bits de status, muitas controladoras utilizam interrupções para informar à CPU quando estão prontas para ter seus registradores lidos ou escritos. No contexto da E/S, tudo que você precisa saber é que a maioria dos dispositivos de interface fornece uma saída que é logicamente igual ao bit de status de "operação completa", ou "dados prontos" de um registrador, mas que se destina a ser usada para estimular uma das linhas de pedido de interrupção. O número de entradas na controladora de interrupção podem ser limitado.

5. Acesso direto à memória

Tenha ou não E/S mapeada em memória, a CPU de um sistema precisa endereçar as controladoras de dispositivos para trocar dados com elas. A CPU pode solicitar dados de uma controladora de E/S um byte por vez, mas fazer isso para um dispositivo como um disco, que produz um bloco de dados grande, desperdiçaria tempo de CPU, portanto, é usado frequentemente um esquema diferente denominado ***Acesso Direto à Memória*** (*DMA*).

O OS pode usar apenas DMA se o hardware tiver uma controladora de DMA, o que a maioria dos sistemas possui. Mais comumente, é disponível uma única controladora de DMA para regular as transferências dos vários dispositivos de E/S, muitas vezes de forma concomitante. Não importa onde esteja localizada fisicamente, a controladora de DMA tem acesso ao barramento do sistema independente da CPU.



Primeiramente, a controladora lê o bloco da unidade de disco em śerie, bit por bit, até que o bloco inteiro esteja em seu *buffer* interno. Em seguida, ela calcula a soma de verificação para verificar se não ocorreu nenhum erro de leitura. Então, a controladora causa uma interrupção. Quando o sistema operacional começa a executar, ele pode ler o bloco de disco do *buffer* da controladora, um byte ou palavra por vez, executando um laço, com cada iteração lendo um byte ou palavra de um registrador de dispositivo da controladora, armazenando-o na memória principal, incrementando o endereço de memória e decremento a contagem de itens a serem lidos até que ela chegue a zero.

Quando o DMA é usado, o procedimento é diferente. Primeiramente, a CPU programa a controladora de DMA, configurando seus registradores para saber o que deve transferir e para onde. Ela também envia um comando para a controladora de disco, dizendo a ela para que leia dados do disco em seu *buffer* interno e confira a soma de verificação. Quando dados válidos estiverem no *buffer* da controladora de disco, o DMA poderá começar.

A controladora de DMA inicia a transferência enviando uma requisição de leitura para a controladora de disco pelo barramento (2). Essa requisição de leitura é semelhante às outras e a controladora de disco não sabe, nem se preocupa, se ele veio da CPU ou de uma controladora de DMA. Normalmente, o endereço de memória a ser escrita é posto nas linhas de endereçamento do barramento, de modo que, quando a controladora de disco busca a próxima palavra de seu *buffer* interno, ela sabe onde escrevê-la. A escrita na memória é outro ciclo de barramento padrão (3). Quando a escritura termina, a controladora de disco envia um sinal de reconhecimento (*ack*) para a controladora de DMA, também pelo barramento (4).

Então, a controladora de DMA incrementa o endereço de memória a ser usado e decrementa a contagem de bytes. Se a contagem de bytes ainda for maior do que 0, as etapas de 2 e 4 se repetem até que chegue a 0. Nesse ponto, a controladora causa uma interrupção. Quando o sistema operacional inicia, não precisa copiar o bloco na memória, o bloco já vai estar lá

O segundo motivo é que, uma vez iniciada uma transferência de disco, os bits continuarão chegando do disco a uma velocidade constante, esteja a controladora pronta para eles ou não. Se a controladora tentasse escrever os dados diretamente na memória, ela teria que usar o barramento do sistema para cada palavra transferida. Se o barramento estivesse ocupado por algum outro dispositivo que o estivesse usando, a controladora teria de esperar. Se a próxima palavra do disco chegasse antes que a anterior tivesse sido armazenada, a controladora precisaria armazená-la em algum outro lugar.

Nem todos os computadores utilizam DMA. Se não houver nenhum outro trabalho para ela, não tem sentido fazer a CPU esperar que a controladora de DMA termine. Além disso, livrar-se da controladora de DMA e fazer com que a CPU realize todo o trabalho no software significa economia de dinheiro, o que é importante em sistemas de baixo custo ou portáteis como os sistemas embarcados.