Capítulo 5

Gerenciamento de I/O

O acrônimo I/O (*Input/Output*) ou E/S (Entrada/Saída) representa toda a sorte de dispositivos eletrônicos, eletromecânicos e ópticos que são integrados a um sistema computacional com o propósito de realizar a comunicação do processador e memória com o meio externo. De certa forma, o computador seria uma máquina inútil caso não existissem meios de realizar-se as operações de entrada e saída.

Os dispositivos de I/O, dispositivos periféricos ou apenas periféricos podem ser classificados de forma ampla em três categorias [STA96, p. 179]:

Human-Readable Dispositivos apropriados para serem utilizados por usuários do computador tais como o teclado, o *mouse* ou o monitor de vídeo.

Machine-Readable São aqueles projetados para interligação do computador com outros equipamentos, tais como unidades de disco, CD-ROM ou fita magnética.

Communication Destinados a realizar a comunicação do computador com outros dispositivos remotos, tais como placas de rede ou *modems*.

5.1 Módulos de I/O

A arquitetura de Von Neumann (Figura 1.2) define o computador como o conjunto de processador, memória e dispositivos de entrada e saída interconectados através vários barramentos (*buses*) especializados: o barramento de dados (*data bus*), o barramento de endereços (*address bus*) e o barramento de controle (*control bus*).

Mas diferentemente do que poderíamos pensar, os dispositivos periféricos em si não são conectados diretamente à tais barramentos, mas, ao invés disso, os periféricos são conectados a **módulos de I/O** que por sua vez são ligados aos barramentos do sistema. As razões para isto são várias:

- Existem inúmeros tipos de periféricos, com diferentes formas de operação, sendo impraticável implantar no computador uma lógica que permitisse a operação com todos ou mesmo uma grande parte destes dispositivos.

- Como visto, a velocidade de operação dos dispositivos periféricos é muito menor que a da memória ou do processador.

Desta forma é muito mais conveniente implementar-se **módulos de I/O** que atuem como conexões mais genéricas para os diferentes periféricos, possibilitando o uso de estruturas padronizadas para ligação com a memória e processador. Por essa razão, os módulos de IO são freqüentemente chamados de **interfaces**.

Um módulo de I/O geralmente possui algumas linhas de controle internas que servem para determinar as ações que devem ser executadas pelo módulo, linhas de *status* que servem para indicar o estado de funcionamento do módulo ou resultado de alguma operação, linhas de dados para conexão do módulo de I/O com o barramento de dados do sistema e um conjunto particular de linhas para conexão com o periférico que efetivamente será controlado pelo módulo de I/O. Na Figura 5.1 temos um representação genérica do que pode ser um módulo de I/O.

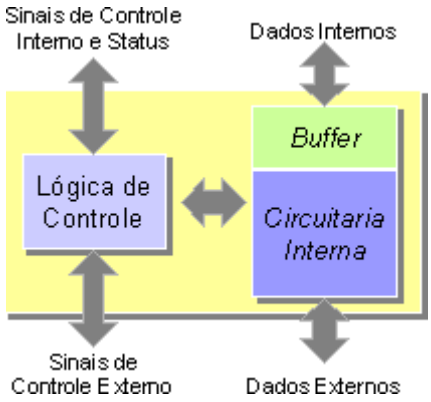

Figura 5.1: Estrutura genérica de módulo de I/O

A conexão com o barramento do processador geralmente tende a ser uma interface de alto nível, isto é, uma conexão orientada à comandos, mais adaptada à operação com um processador e, portanto, dirigida ao trabalho de programação de alto nível. Já a conexão com o periférico propriamente dito é usualmente uma interface de baixo nível, contendo inúmeros sinais elétricos e um protocolo dedicado próprio, que exige tratamento mais especializado. Desta forma, as particularidades de cada tipo de periférico ficam isoladas do sistema principal, facilitando o desenvolvimento dos programas que utilizarão estes dispositivos.

Esta estrutura permite que um único módulo de I/O controle mais de um periférico, geralmente do mesmo tipo, tal como nos controladores de unidades de disco IDE [1]que podem administrar de uma até quatro unidades de disco deste padrão.

Os módulos de I/O geralmente executam algumas das seguintes funções: controle e temporização, comunicação com o processador, comunicação com periférico, armazenamento temporário de dados e detecção de erros.

Uma outra estrutura possível para os módulos de I/O são os **canais de I/O** (*I/O channels*). Os canais de I/O são sistemas computacionais de propósito especial destinados ao tratamento de entrada e saída de forma independente do processador do sistema computacional [DEI92, p. 27]. Esta alternativa estrutural, usada tipicamente em computadores de grande porte (*mainframes*), opera com múltiplos barramentos de alta velocidade, podendo acessar o armazenamento primário de forma independente, proporcionando grande desempenho, pois as operações de I/O são realizadas paralelamente ao processamento.

Micro e minicomputadores utilizam geralmente um modelo de barramento interno simples para comunicação entre processador, memória e os demais dispositivos do sistema. Compartilhando este barramento encontram-se dispositivos especializados nas em funções mais importantes (unidades de disco e monitor de vídeo) chamados controladores, proporcionando considerável ganho de performance e ainda assim utilizando uma arquitetura mais simples que os canais de I/O [TAN92, p. 207].

5.2 Operação de Módulos de I/O

Os módulos de I/O podem ser operados de três maneiras básicas:

- I/O Programado,
- I/O com Interrupções e
- I/O com Acesso Direto à Memória (DMA)

De forma geral, o que distingue estas três formas de operação é a participação do processador e a utilização do mecanismo de interrupções, conduzindo a resultados bastante distintos.

5.2.1 I/O Programado

Com o I/O programado (*programmed I/O*) os dados são trocados diretamente entre o processador e o módulo de I/O, ou seja, o processador deve

[1]O acrônimo IDE significa *integrated device eletronics* ou dispositivo eletrônico integrado.

executar um programa que verifique o estado do módulo de I/O, preparando-o para a operação se necessário, enviando o comando que deve ser executado e aguardando o resultado do comando para então efetuar a transferência entre o módulo de I/O e algum registrador do processador.

Portanto é responsabilidade do processador verificar o estado do módulo de I/O em todas as situações, inclusive quando aguarda dados. Isto significa que o processador não pode executar outras tarefas enquanto aguarda a operação do módulo de I/O. Veja na Figura 5.2 um esquema indicando o funcionamento das operações de I/O programadas.

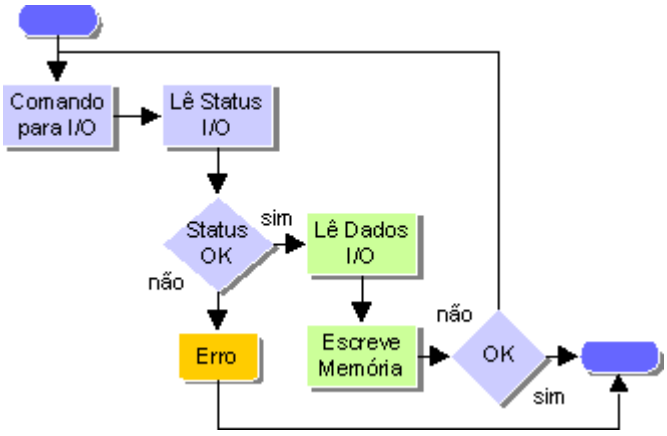

Figura 5.2: Funcionamento do I/O programado

Como os comandos enviados ao módulo de I/O podem significar uma operação com um dispositivo periférico lento, digamos uma unidade de disco, então o processador deverá permanecer em espera durante toda a operação executada no periférico, por mais lento que possa ser, significando um sério comprometimento da performance geral do sistema, dado que o processador fica ocupado com a monitoração da operação em andamento (veja Figura 5.3).

O módulo de I/O se relaciona com o processador através de comandos, ou seja, códigos de controle transformados em sinais enviados ao módulo que indica qual operação deverá ser realizada, tais como controle (*control*), teste (*test*), escrita (*write*) ou leitura (*read*). Geralmente os comandos do módulo de I/O tem equivalência direta com as instruções do processador, facilitando sua operação integrada.

Dado que é possível para um módulo de I/O controlar mais de um dispositivo periférico, então é necessário que sejam associados endereços a cada um destes periféricos, de forma a permitir sua operação individualizada. Existem duas formas para a interpretação destes endereços por parte dos módulos de I/O quando existe o compartilhamento de barramentos do sistema:

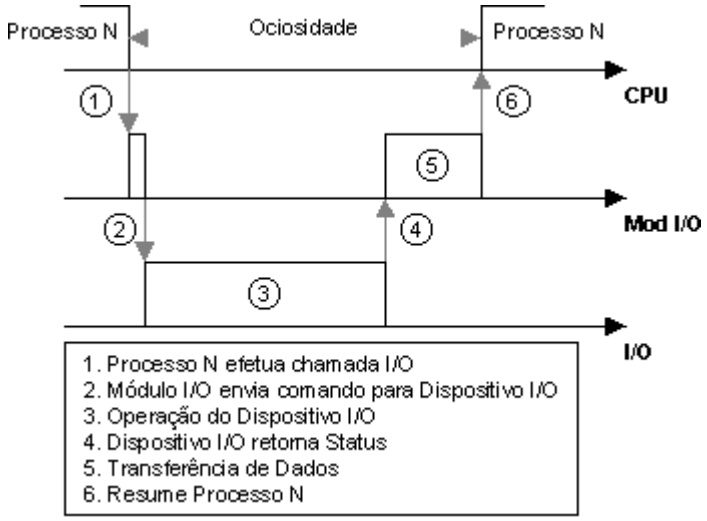

Figura 5.3: Temporização do I/O programado

- **Mapeada em Memória** (*memory-mapped*)
  Onde o módulo de I/O opera dentro do espaço de endereçamento da memória, usando um conjunto de endereços reservados. Desta forma o processador trata os registradores de *status* e dados do módulo de I/O como posições ordinárias de memória utilizando operações comuns de leitura e escrita. Para o funcionamento neste modo o processador deve dispor de uma linha individual de leitura e outra para escrita.

- **Mapeada em I/O** (*I/O mapped*)
  Também chamada de I/O isolado (*isolated I/O*), onde existe um espaço de endereçamento independente para os dispositivos de I/O. Para tanto o processador deve dispor de uma linha de leitura/escrita e outra de entrada/saída. As portas de I/O passam a ser acessíveis apenas por operações especiais de I/O.

Utilizando a forma de I/O mapeado em memória temos uma maior simplicidade e a disponibilidade de um maior conjunto de instruções embora reduzido espaço em memória devido a reserva de endereços para portas de I/O. Os computadores baseados nos processadores Motorola 680x0 utilizam este método.

Com o I/O isolado temos maior segurança nas operações envolvendo memória ou I/O e um maior espaço de endereçamento as custas de uma organização ligeiramente mais complexa e um reduzido número de instruções dedicadas. Os microcomputadores IBM PC compatíveis são um exemplo desta utilização. No caso, para enviar-se dados ao monitor de vídeo padrão do sistema devem ser utilizados os endereços de I/O na faixa de `03D0h` a `03DFh`, enquanto que o acesso às placas de som tipo SoundBlaster devem

usar o endereço `0220h`.

5.2.2 I/O com interrupções

Para superar o problema da espera do processador por operações nos dispositivos periféricos, pode ser utilizado o mecanismo das interrupções, ou seja o I/O através de interrupções (*interrupt driven I/O*). Tecnicamente falando, uma interrupção permite que uma unidade ganhe a atenção imediata de outra, de forma que a primeira unidade possa finalizar sua tarefa [DEI92, p. 25].

Assim sendo, quando o processador envia um comando para o módulo de I/O, o mesmo pode passar executar uma outra tarefa, sem a necessidade de monitorar o módulo acionado. Quando a operação for concluída, o módulo de I/O interrompe o processador, isto é, aciona uma interrupção para requisitar o processamento dos dados (a troca de dados com o processador). O processador executa a troca de dados, liberando o módulo de I/O e retomando o processamento anterior.

Conforme podemos observar na Figura 5.4, a operação de I/O com interrupções é a seguinte:

1. O processador envia um comando ao módulo de I/O (por exemplo, uma operação *read*), que a realiza de modo independente (i.e., em paralelo a atividade do processador).

2. O processador passa a executar outra tarefa, ou seja, um outro processo.

3. Ao finalizar a operação requisitada, o módulo de I/O sinaliza uma interrupção para o processador.

4. Ao término da instrução corrente, o processador verifica a ocorrência de uma interrupção, determinando qual dispositivo a originou e então sinalizando conhecimento (*acknowledgment signal*).

5. O processador salva o contexto da tarefa atual na pilha (*stack*), preservando assim o conteúdo do contador de programa e demais registradores.

6. O processador carrega o endereço da rotina de serviço no contador de programa para poder iniciar a execução da rotina de tratamento da interrupção detectada. Tais endereços são armazenados numa região pré-determinada da memória, denominada vetor de interrupções.

7. A rotina de tratamento da interrupção é executada, ou seja, os dados solicitados são lidos do módulo de I/O para um registrador do processador e depois transferidos para uma área de memória apropriada.

8. Finalizada a rotina de tratamento da interrupção, o processador restaura o contexto da tarefa interrompida, lendo o conteúdo do contador de programa e demais registradores da pilha.

9. O processador retorna ao processamento da tarefa no ponto em que foi interrompida.

10. Mais um momento mais a frente, em seu *quantum*, o processo que solicitou a operação de I/O será executado com os dados obtidos a sua disposição.

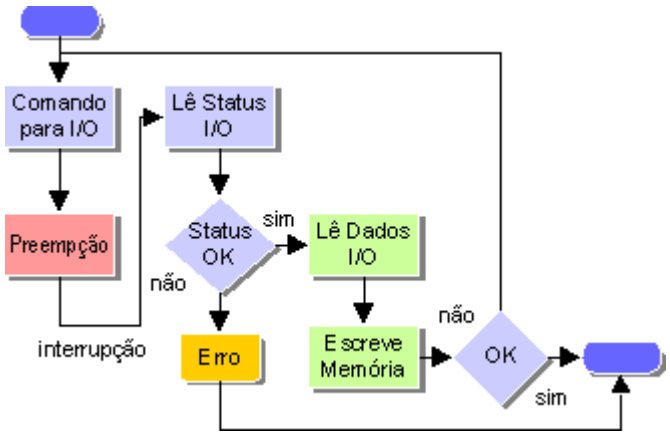

Figura 5.4: Funcionamento do I/O com interrupções

A operação de dispositivos de I/O utilizando interrupções permite que o processador permaneça trabalhando enquanto o módulo de I/O realiza a operação solicitada, melhorando o desempenho do sistema pois duas atividades são *paralelizadas*, embora os dados da operação continuem a ser manipulados pelo processador, como mostra também a Figura 5.5.

O maior problema relacionado com o uso das interrupções é que, usualmente, o processador dispõe de poucas linhas de interrupção. Desta forma surgem as seguintes questões: como o processador identificará o módulo que sinalizou uma interrupção e como serão tratadas múltiplas interrupções simultâneas?

Para resolver-se esta questões, podem ser empregadas várias diferentes técnicas [STA96, p. 194]: múltiplas linhas de interrupção, *software poll*, *hardware poll* vetorizado e *bus arbitration* vetorizada. Usualmente são assinalados números para as interrupções, onde as de menor número tem prioridade sobre as número maior. Isto significa que uma interrupção de número 4 será processada primeiro do que uma interrupção de número maior ($>= 5$), sem ser interrompida por estas, mas podendo ser interrompida por uma interrupção de número menor (< 4).

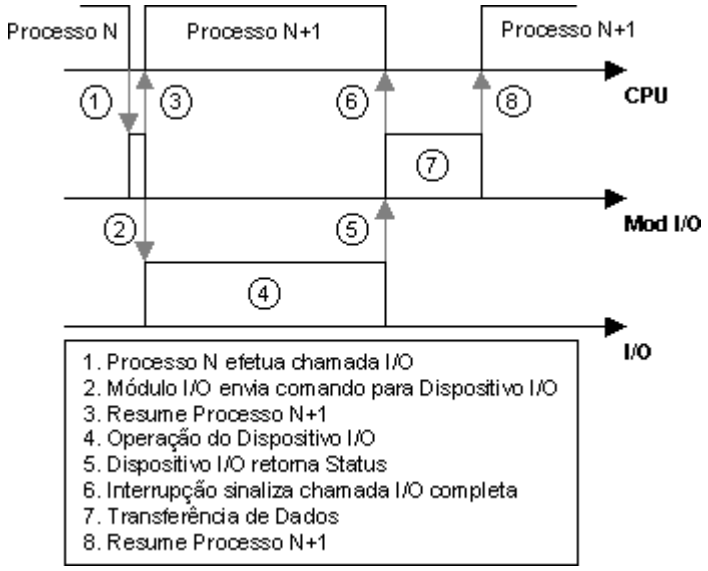

Figura 5.5: Temporização do I/O com interrupções

Como exemplo, apresentamos a Tabela 5.1 que contêm o mapeamento das interrupções num sistema IBM-PC compatível. Alguns dos valores são padronizados enquanto outros são particulares do sistema utilizado como exemplo.

Tabela 5.1: Mapa de interrupções de um IBM-PC compatível

Int	Dispositivo	Int	Dispositivo
0	Cronômetro do sistema	8	CMOS/Relógio do sistema
1	Teclado	9	Porta de comunicação COM3
2	Controlador de interrupção	10	Porta de comunicação COM2
3	Placa de rede (*)	11	Ponte PCI (*)
4	Porta de comunicação COM1	12	Mouse porta PS/2 (*)
5	Placa de som (*)	13	Coprocessador numérico
6	Controlador de disco flexível	14	Controlador IDE/ESDI
7	Porta de Impressora LPT1	15	Controlador IDE/ESDI

(*) Opções não padronizadas

5.2.3 I/O com Acesso Direto à Memória (DMA)

As técnicas de I/O programado e I/O com interrupções possuem um grande inconveniente que é a limitação da velocidade de transferência de dados a capacidade do processador em movimentar tais dados a partir do módulo de I/O para o armazenamento primário, pois isso envolve a execução repetida de várias instruções. Além disso o processador fica comprometido não

apenas com a transferência dos dados, mas com a monitoração do módulo de I/O, no caso de I/O programado, ou com a sobrecarga imposta pelas operações de interrupção, no caso de I/O via interrupção. Se um módulo de I/O for utilizado para a movimentação de uma grande quantidade de dados, ambas as formas comprometerão a performance do sistema.

Para solucionar-se este problema pode ser utilizada uma outra técnica denominada I/O através de acesso direto à memória ou DMA (*direct memory access*).

A técnica de DMA propõe utilizar uma única interrupção para efetuar a transferência de um bloco de dados diretamente entre o periférico e a memória primária, sem o envolvimento do processador e com isso reduzindo o número de operações necessárias e assim acelerando o processo.

Para tanto, torna-se necessária a existência de um módulo adicional, chamado de controlador de DMA, cuja operação, ilustrada na Figura 5.6, é descrita a seguir [STA96, p. 199]:

1. O processador envia comando (leitura ou escrita) para controlador de DMA.

2. O processador continua seu trabalho enquanto DMA efetua a transferência com o dispositivo de I/O.

3. Para acessar a memória o controlador de DMA *rouba* ciclos do processador para acessar a memória principal, atrasando-o ligeiramente.

4. Ao final da operação o controlador de DMA aciona uma interrupção para sinalizar o término da operação.

5. O processador pode executar a rotina de tratamento da interrupção processando os dados lidos ou produzindo novos dados para serem escritos.

Este método é significativamente mais rápido do que o I/O programado ou I/O via interrupções pois utiliza apenas uma única interrupção, o processador fica liberada para executar outras tarefas e a transferência dos dados ocorre em bloco (e não *byte* a *byte*) diretamente entre o periférico e a memória.

O controlador de DMA é um dispositivo especializado nesta operação, suportando tipicamente o trabalho com vários periféricos diferentes, cada um utilizando um canal de DMA (DMA *channel*).

Outra grande vantagem da técnica de DMA é que ela pode ser implementada no *hardware* de diversas formas diferentes, conforme a quantidade de dispositivos de I/O e performance pretendida, como ilustrado na Figura 5.7.