# **Unidade 7**: Dispositivos de Entrada e Saída

Prof. Daniel Caetano

**Objetivo**: Conhecer alguns dos dispositivos de entrada e saída mais comuns, além de compreender a lógica de comunicação entre os dispositivos de E/S mais comuns.

**Bibliografia**:

- MURDOCCA, M. J; HEURING, V.P. **Introdução à arquitetura de computadores**. S.I.: Ed. Campus, 2000.
- STALLINGS, W. **Arquitetura e organização de computadores**. 5ed. São Paulo: Ed. Pearson Prentice Hall, 2003.

## INTRODUÇÃO

Grande parte da funcionalidade de um computador se deve à sua capacidade de se comunicar com dispositivos de entrada e saída. De fato, é usual que se dê o nome de "computador" apenas a dispositivos que realizem operações e que possuam pelo menos uma unidade de entrada e uma de saída.

Uma vez que os dispositivos existentes são os mais variados e, em geral, possuem uma velocidade de comunicação **muito inferior** à da memória, com **tamanhos de palavras** usualmente **distintos** daqueles trabalhados pelo computador, não é praticável instalá-los no mesmo barramento de alta velocidade da memória, o que faz com que normalmente possuam um barramento diferenciado, acessado através de uma das pontes, como visto em aulas anteriores.

Além disso, a maneira com que as informações destes dispositivos são transferidas para a memória pode variar em nível de complexidade, do mais simples e lento ao mais complexo e eficiente. Nesta aula serão apresentados os três modos de comunicação existentes nos computadores modernos. A maior diferença entre os três é o nível de interferência da CPU no processo de comunicação de um dado dispositivo com a memória.

No caso de maior intervenção da CPU no processo, é dado o nome de **Entrada e Saída Programada** (ou ***polling***). No caso de menor intervenção, é dado o nome de **Acesso Direto à Memória** (ou ***Direct Memory Access***, **DMA**). No caso intermediário, é dado o nome de **Entrada e Saída Controlada por Interrupção**.

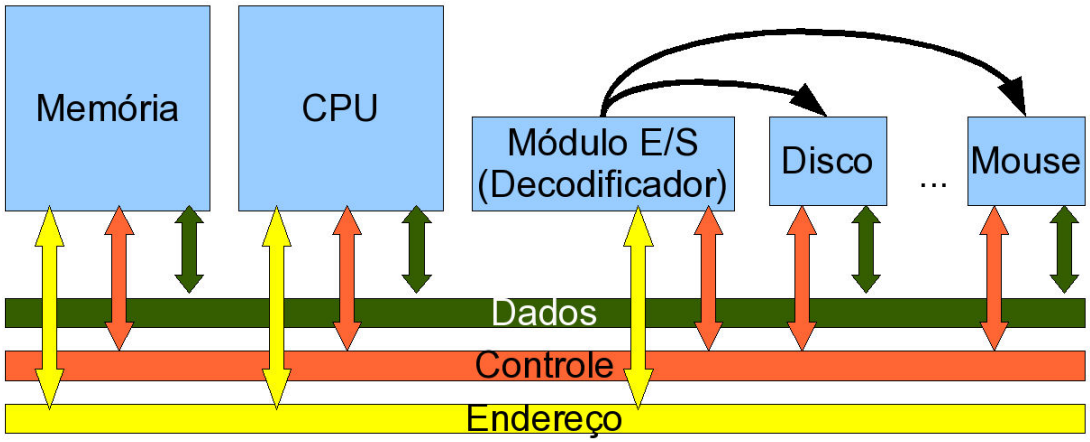
Após a visão geral sobre os métodos de transferência de dados entre dispositivos e memória, serão comentados alguns dos dispositivos mais comuns atualmente.

## 1. COMO SE ATIVA UM DISPOSITIVO DE E/S?

Um dispositivo de E/S, em certo aspecto, é bastante difernente da memória. Enquanto a memória recebe um sinal de ativação e ela é preparada para responder a todos os endereços (recebe os sinais A0 a AN), os dispositivos de E/S tem um funcionamento diferenciado, usualmente tendo apenas um (ou alguns poucos) sinal de ativação (A0).

Para diferenciar entre esses comportamentos, o processador tem, usualmente, formas alternativas para ativar o uso da a memória ou dos dispositivos de E/S. Em geral, o processador central tem um pino chamado MREQ (do barramento de controle) que é ligado diretamente ao seletor de ativação da memória e, no caso da memória, o mesmo ocorre com o barramento de endereços inteiro.

No caso dos dispositivos de E/S, a CPU os ativa por meio de um pino chamado IORQ (também do barramento de controle), que junto com o barramento de endereços aciona um (ou vários) dispositivo chamado **decodificador** que, dependendo da configuração do endereço, acionará um dispositivos adequado. Observe a figura abaixo, simplificada com um único barramento:




Note que este desenho é um exemplo, apenas para indicar como o **módulo de E/S** "interfere" na ativação do dispositivo, em uma funcionalidade chamada de **decodificação de endereço de E/S**.

Alguns dispositivos, como o vídeo e o *hard disk*, não possuem qualquer contato direto com o barramento do sistema, ficando completamente interligados ao módulo de E/S, que assume a responsabilidade de gereciar os três barramentos. Esse tipo de móldulo de E/S é chamado **interface controladora de dispositivo**. Cada dispositivo pode ter seu próprio controlador ou é possível ter um controlador para vários dispositivos.

Como exemplo desses módulos E/S temos a placa de vídeo ou a placa SCSI/SATA. Em alguns casos, o módulo decodificador está fisicamente integrado ao dispositivo, de maneira que não é possível observá-lo.

Note que o que se chama de **interface controladora** não é apenas um seletor e decodificador de endereços; usualmente ela tem muito mais recursos, permitindo, por exemplo, que um determinado dispositivo "tome o controle" dos barramentos temporariamente, permitindo que o dispositivo converse diretamente com a memória (comum) ou com outros dispositivos (incomum).

A partir da próxima seção estudaremos quais são as formas com que a comunicação com o dispositivo pode ocorrer.

## 2. ENTRADA E SAÍDA POR *POLLING*

No esquema chamado *polling*, a CPU é responsável por todo o controle de transferências de dados de dispositivos. Isso significa que ela é responsável não só pela transferência de informações em si, mas também pela verificação constante dos dispositivos, para saber se algum deles tem dados a serem transferidos.

Isso significa que, de tempos em tempos, a CPU faz a seguinte 'pergunta', seqüencialmente, a todos os dispositivos conectados: "Você tem dados para serem transferidos para a memória?".

Quando algum dispositivo responder "sim", a CPU faz a transferência e continua perguntando aos outros dispositivos em seguida. Quando nenhum dispositivo necessitar de transferências, a CPU volta a fazer o que estava fazendo antes: executar um (ou mais) programas. Depois de algum tempo, ela volta a realizar a pergunta para todos os dispositivos novamente.

> Uma analogia que costuma ajudar a compreender a situação é a do garçom e a do cliente em um restaurante. No sistema de *polling* o garçom é a CPU e o cliente é o dispositivo. O cliente tem que esperar pacientemente até o garçom resolver atendê-lo e, enquanto o garçom atende a um cliente, ele não pode realizar qualquer outra tarefa. Depois que o cliente fez o pedido, ele ainda tem que esperar, pacientemente, o garçom trazer a comida.

Não é difícil ver que este sistema tem **três** problemas fundamentais:

a) A CPU gasta uma parcela considerável de tempo de processamento só para verificar se algum dispositivos tem dados a serem transferidos para a memória.

b) A CPU gasta uma parcela considerável de tempo de processamento apenas para transferir dados de um dispositivo para a memória.

c) Se um dispositivo precisar de um atendimento "urgente" (porque vai perder dados se a CPU não fizer a transferência imediata para a memória, para liberar espaço no dispositivo), não necessariamente ele terá.

A primeira questão é considerada um problema porque muitas vezes a CPU perde tempo perguntando para todos os dispositivos e nenhum deles tem qualquer dado a ser transferido. É tempo de processamento totalmente disperdiçado.

A segunda questão é considerada um problema porque cópia de dados é uma tarefa que dispensa totalmente a capacidade de processamento: é uma tarefa que mobiliza toda a CPU mas apenas a Unidade de Controle estará trabalhando - e realizando um trabalho menos nobre. Adicionalmente, para a transferência de um dado do dispositivo para a memória, tendo que passar pela CPU, o barramento é ocupado duas vezes pelo mesmo dado, ou seja, são realizadas duas transferências: dispositivo=>CPU e depois CPU=>Memória. Isso acaba sendo um "retrocesso", transformando o barramento em um modelo de Von Neumann simples.

A terceira questão é considerada um problema porque, eventualmente, dados serão perdidos. Adicionalmente, se o dispositivo em questão precisar **receber** dados para tomar alguma atitude, problemas mais sérios podem ocorrer (como uma prensa hidráulica controlada por computador causar a morte de uma pessoa por falta de ordens em tempo hábil).

Por outro lado, o sistema de transferência por *polling* é de implementação muito simples, o que faz com que muitas vezes ele seja usado em aplicações onde os problemas anteriormente citados não são totalmente relevantes.

## 3. ENTRADA E SAÍDA POR INTERRUPÇÃO

No esquema chamado de entrada e saída por interrupção, a CPU fica responsável apenas pelas transferências em si. Isso significa que ela não tem que verificar os dispositivos, para saber se há dados a serem transferidos.

Mas se a CPU não faz a verificação, como ela vai perceber quando uma transferência precisa ser feita? Simples: o dispositivo dispara um sinal do barramento de controle chamado "Interrupção" (chamado de IRQ - *Interrupt ReQuest*). Quando a CPU percebe este sinal, ela sabe que algo precisa ser feito com algum dispositivo; normalmente uma transferência de dados (seja de entrada ou saída).

> Voltando a analogia do restaurante, o sistema com interrupções seria o fato de o cliente possuir uma sineta que, ao tocar, o garçom viria o mais rapidamente possível para atender ao cliente

Há sistemas em que há mais dispositivos que interrupções. Neste caso, o sistema ainda terá que fazer *polling* para saber qual foi o dispositivo que solicitou atenção; entretanto, o polling será feito somente quando ***certamente*** um dispositivo precisar de atenção da CPU (uma entrada ou saída de dados). Desta forma, elimina-se o problema de tempo perdido fazendo *pollings* quando nenhum dispositivo precisa de transferências.

A forma mais eficiente, entretanto, é quando temos pelo menos uma interrupção por dispositivo, de forma que a CPU saiba sempre, exatamente, qual é o dispositivo que está solicitando atenção e nenhum tipo de *polling* precise ser feito.

Essa característica resolve os problemas a) e c) existentes no sistema de *polling* puro, embora o problema b), relativo ao tempo de CPU gasto com as transferências em si, ainda esteja presente.

Entretanto, sempre que lidamos com sinais no barramento, temos que levantar uma questão: e se dois ou mais dispositivos solicitarem uma interrupção ao mesmo tempo? Normalmente os sistemas com várias interrupções possuem vários **níveis de prioridade de interrupção** (*interrupt level*). A CPU sempre atenderá a interrupção de maior prioridade primeiro (normalmente de "número" menor: IRQ0 tem mais prioridade que IRQ5).

Existe um outro problema também: quando a CPU recebe uma IRQ, ela **sempre** para o que está fazendo, momentaneamente, para realizar outra atividade qualquer. Ao finalizar esta atividade, ela volta ao que estava fazendo antes da IRQ ocorrer. Entretanto, há alguns processos em que talvez o programador não queria interrupções - em aplicações onde o controle de tempo é crítico, as interrupções podem causar problemas graves. Nestes casos, o programador pode usar uma instrução em linguagem de máquina que desliga as interrupções (normalmente chamada DI, de *Disable Interrupts*). Obviamente ele precisa ligá-las novamente depois (usando uma instrução normalmente chamada EI, de *Enable Interrupts*).

Entretanto, ao desligar as interrupções, o programador pode, potencialmente, causar um dano grave ao funcionamento do sistema operacional, por exemplo. Os sistemas operacionais modernos usam a interrupção para realizar a troca de aplicativos em execução, na chamada "multitarefa preemptiva". Por esta razão, as arquiteturas modernas possuem pelo menos uma interrupção chamada de Interrupção Não Mascarável (NMI, de *Non Maskable Interrupt*), que não pode ser desligada, nunca.

## 4. ENTRADA E SAÍDA POR DMA

No esquema chamado de entrada e saída por DMA (Acesso Direto à Memória), a CPU fica responsável apenas por coordenar as transferências. Isso significa que ela não tem que verificar os dispositivos, para saber se há dados a serem transferidos e nem mesmo transferir estes dados.

Mas se a CPU não faz a verificação, como ela vai perceber quando uma transferência precisa ser feita? Como já foi visto, pela interrupção (assim, DMA pressupõe interrupções). Mas se a CPU não faz a transferência, como os dados vão parar na memória? Simples: a CPU comanda um dispositivo responsável pela transferência, normalmente chamado simplesmente de DMA. Quando a CPU perceber o sinal de IRQ, ela verifica qual a transferência a ser feita e comanda o DMA, indicando o dispositivo origem, a posição origem dos dados, a posição

destino dos dados e o número de bytes a transferir. O circuito do DMA fará o resto. Quando ele acabar, uma outra interrupção será disparada, informando que a cópia foi finalizada.

> Voltando a analogia do restaurante, o sistema com DMA seria como se, ao cliente tocar a sineta, o garçom (CPU) mandasse um outro garçom auxiliar (DMA) para fazer o que precisa ser feito, e o garçom "chefe" (CPU) continuasse a fazer o que estava fazendo antes. Ao terminar seu serviço, o garçom auxiliar (DMA) avisa ao garçom chefe (CPU) que está disponível novamente.

Este sistema resolve todos os problemas a), b) e c) apresentados anteriormente.

## 5. DISPOSITIVOS PRINCIPAIS DE ENTRADA E SAÍDA

Os dispositivos de entrada e saída dos computadores atuais são de conhecimento geral. Entretanto, veremos alguns detalhes de seu funcionamento interno.

### 5.1. HardDisks

Os harddisks são, em essência, muito similares aos disk-drives e disquetes comuns; entretanto, existem diferenças.

Dentro de um harddisk existem, normalmente, vários discos (de alumínio ou vidro) fixos a um mesmo eixo, que giram em rotações que variam de 3000 a 1000 rpm (rotações por minuto). Estes discos são, assim como os disquetes, cobertos por óxido de ferro, que pode ser regionalmente magnetizado em duas direções (indicando 0 ou 1). Estas regiões são acessíveis na forma de setores de uma trilha do disco. O tamanho de bits disponíveis em cada setor varia de caso para caso, mas 512 bytes é um valor comum.

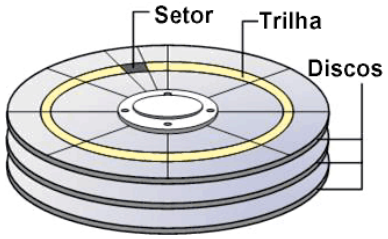



**Figura 1**: Discos, trilhas e setores

Para ler estes bits de um setor, existe normalmente uma cabeça de leitura para cada face de disco. No caso acima, seriam, normalmente, 6 cabeças de leitura, presas a um mesmo eixo. Apenas uma funciona por vez.

Quando é necessário ler um dado do disco, é preciso descobrir em que setor(es) ele está e ler este(s) setor(es). Um sistema de arquivos é exatamente uma maneira ordenada de guardar arquivos em diversos setores e o sistema operacional faz a parte mais chata de

localização, fazendo com que os programadores possam lidar diretamente com o conceito lógico de "arquivos".

## 5.2. Discos Ópticos

Os discos ópticos são discos plásticos cobertos de uma camada de alumínio que é "entalhado" para representar os 0s e 1s: 1 quando não houver uma concavidade e 0 quando houver. Para a leitura há uma cabeça que é capaz de disparar um feixe de laser e também é capaz de recebê-lo de volta. O feixe de laser é disparado contra a superfície "entalhada" do disco; se encontrar uma concavidade, sofre um tipo de deflexão e a lente detecta esta deflexão como sendo um 0. Se não encontrar a concavidade, a deflexão será diferente, e a lente detectará esta outra deflexão como 1.

Os dados não são colocados em um disco óptico em um formato de trilhas e setores. Os dados em um disco óptico são colocados em forma espiral, que vai do interior para a borda do disco. O disco **não** gira com velocidade constante: gira mais lentamente à medida em que a cabeça se afasta do centro do disco.

## 5.3. Teclados

Independente da posição das teclas de um teclado, em geral elas são compreendidas pelo computador na forma de uma matriz com um determinado número de linhas e colunas ou simplesmente numeradas (juntando o número da linha e da coluna). Por exemplo: se a tecla "A" fica na linha 4 e coluna 2, então o número da tecla poderia ser 402, por exemplo.

Entretanto, a maioria dos teclados consegue enviar apenas um número de 7 ou 8 bits para o computador (0 a 127 ou 0 a 255), fazendo com que uma numeração tão simples não seja possível, mas a idéia permanece. Ainda assim, as teclas especiais que modificam o significado das teclas (como shift, alt, ctrl, etc) fariam com que 7 (ou 8) bits jamais fossem suficientes para transmitir todas as possíveis combinações. Por esta razão, alguns dos valores de 7 (ou 8) bits são separados para indicar o pressionamento de teclas especiais, de forma que os valores são transmitidos separadamente.

A função do "mapa de teclado" que é selecionado ao configurar um sistema operacional é justamente a de "mapear" cada número que o teclado irá enviar ao computador a uma letra correta. É por esta razão que se um teclado US-International for configurado com um mapa ABNT-2 as teclas não se comportarão como o esperado.

Mais uma vez o sistema operacional faz a maior parte do trabalho para o programador, que recebe diretamente o valor correto da tecla (letra ou número), ao invés de ter que lidar com números de teclas no teclado.

## 5.4. Mouses Ópticos

Os mouses ópticos são dispositvos extremamente complexos. Sua função é emitir uma luz sobre uma superfície e digitalizar a imagem da superfície onde ele se encontra. O mouse realiza essa operação milhares de vezes por segundo, comparando as imagens coletadas para detectar qual foi o **deslocamento** entre elas.

Depois de calculado este deslocamento, o mouse converte esse valor em um deslocamento proporcional que será enviado ao computador, que moverá o ponteiro do mouse na tela. É importante notar que o deslocamento na tela será tão maior quanto mais brusco for o movimento do mouse (um deslocamento rápido do mouse causa maior deslocamento na tela que o mesmo deslocamento de mouse feito lentamente).

## 5.5. Monitores de Vídeo LCD

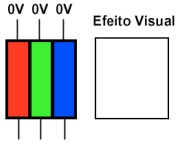
Os monitores de vídeo do tipo LCD são compostos de 3 partes:

a) um papel de fundo, onde são pintados todos os pixels, cada um deles divididos verticalmente em 3 cores: vermelho, verde e azul (RGB, Red Green Blue).
b) um sistema de iluminação (que deve iluminar uniformemente o papel de fundo)
c) uma tela de cristal líquido

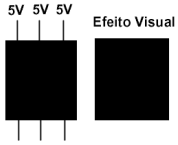
A tela de cristal líquido é composta por uma matriz de elementos de cristal líquido. O cristal líquido tem uma propriedade que é a de impedir que a luz passe em uma dada direção quando colocado sobre uma diferença de potencial suficientemente alta. A quantidade de luz que o cristal líquido deixa passar naquela direção é proporcional à diferença de potencial (campo elétrico). Pequena diferença de potencial faz com que muita luz passe; grande diferença de potencial faz com que pouca luz passe.

Cada pixel tem, então, três células de cristal líquido: uma para a região vermelha do papel, outra para a região verde do papel e outra para a região azul do papel. As cores são compostas variando o campo elétrico em cada célula de cristal líquido. Exemplos:

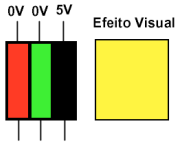
1) Campo baixo em R, G e B: cor branca (todas as cores "acesas")



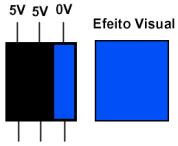

2) Campo alto em R, G, B: cor preta (todas as cores "apagadas")



3) Campo baixo em R e G e alto em B: cor amarela (só vermelho e verde "acesos")




4) Campo baixo em B e alto em R e G: cor azul (só azul "aceso")




As variações de intensidade e matiz das cores são obtidas variando a diferença de potencial em cada uma das células de cristal líquido.

## 6. BIBLIOGRAFIA

MURDOCCA, M. J; HEURING, V.P. **Introdução à arquitetura de computadores**. S.I.: Ed. Campus, 2000.

STALLINGS, W. **Arquitetura e organização de computadores**. 5ed. São Paulo: Ed. Pearson Prentice Hall, 2003.