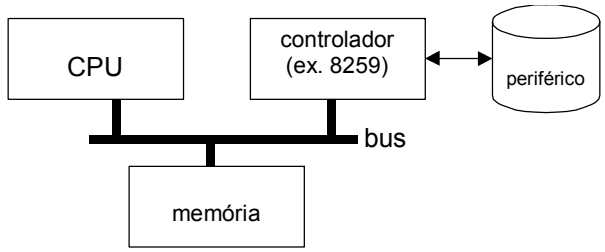
ARQUITETURA DE COMPUTADORES II

UNIDADE 1: ENTRADA E SAÍDA EM SISTEMAS DIGITAIS

Conteúdo:

1 INTRODUÇÃO

- Bibliografia: Herzog Cap.10 / H/S Interface Cap. 8
- Tipos de periféricos (parceiros da CPU na comunicação)
  - Entrada: teclado, mouse, caneta ótica, scanner, microfone, joystick, sensores, …
  - Saída: vídeo, impressora, plotter, auto-falante, atudadores, …
  - Armazenamento: discos, fitas, CD-ROM, ...
  - Comunicação: modem, placa de rede, ...
- Problema: Comunicação entre CPU e Periféricos
- Causas:
  - Diferentes características físicas (conexão, pinagem)
  - Diferente sentido da comunicação (só vai, só vem)
  - Diferentes protocolos (linguagem, formato dos dados)
  - Diferentes velocidades (regulares ou não – tempo de resposta variável)
  - Diferente freqüência (cíclico ou não)
  - Diferentes fabricantes
- Ex:
  - Mouse (sentido único – leitura dos botões e da posição x e y, freqüente, cíclico, tempo de resposta constante)
  - Teclado (sentido único – leitura de teclas, menos freqüente, aleatório, tempo de resposta variável – tecla disponível?)
  - Monitor (sentido único – escrita de dados na memória de vídeo, freqüente)
  - Disco rígido (dois sentidos – ler e escrever setor, não tão freqüente, tempo de reposta variável, rajadas – burst)
  - Impressora (dois sentidos – imprimir e códigos de controle de fluxo e erro, rajadas)
- Solução:
- 1º passo Hardware: inserir circuitos entre a CPU e os periféricos para **interface** e **controle** (chamado de controladora)



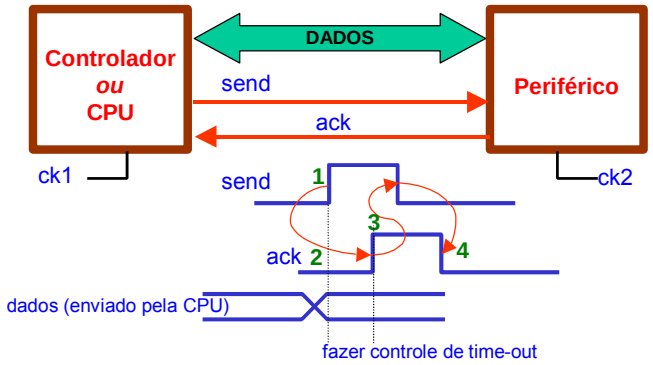
- Funções da controladora
  - Conexão física
  - Conversão de protocolos (tradução)
  - Conversão de tipos de dados (Ex: palavra → setor)
  - Armazenamento temporário (amortização das diferentes velocidades)
  - Abstração do hardware
- Funcionamento
  - CPU programa controladora
  - Comunicação com a controladora através de software (driver, gerenciador de dispositivo - S.O.)
  - CPU não conhece detalhes do hardware do dispositivo (facilita a expansão da máquina)
- Exemplos de circuitos controladores:
  - 8237: responsável pelo DMA (principalmente refresh da memória)
  - 8259: interrupções
  - 16550: UART (universal, assynchrnous receiver transmiter) – serial
  - 8253 para timer, 6845 para vídeo…
- Tenho algum destes no meu PC hoje? *Não nesta forma*
- Hoje estes controladores estão integrados nos ***chip-sets***
- Isto basta? Abstração facilitou, mas como converso com a controladora?
  - Enviar dados e comandos para os registradores (como endereçar)?
  - Controlar respostas?
  - Como trocar dados?

2 MODELOS DE TROCA DE DADOS

- Assíncrono / síncrono / semi-síncrono
- Responsável pela interação entre o periférico e controlador: protocolos
- Como transferir a Informação em nível físico?

2.1 Assíncrono

- Dois sistemas digitais autônomos
- Clocks não sincronizados e com freqüências diferentes



- Procedimento:

CPU	Periférico
1) Fornece dados e send ↑	1) Monitorando subida do sinal send
2) Fica a esperar subida do sinal ack	2) Quando send ↑ armazena os dados
	3) Uma vez armazenados os dados ack ↑
3) Quando ack ↑ remove send (↓) e os dados	4) Fica a esperar descida do sinal send
	5) Quando send é removido remove-se o ack (↓)
4) Fica a esperar descida do sinal ack (↓)	

Perguntas

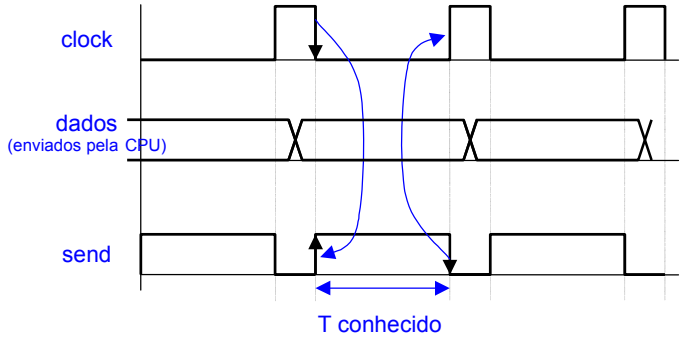
1. Deve haver algum mecanismo de controle entre os passos 1 e 2 da CPU? *(em relação à figura)*
   (resposta: sim, controle de time-out)

2. Porque os passos 3 e 4 da CPU são necessários?
   (resposta: o passo 3 permite reinicializar a CPU para nova emissão e o passo 4 aguarda a reinicialização do periférico)

2.2 Transferência Síncrona

- Mesmo clock para CPU e periférico (não adianta a mesma freqüência!)
- Clock sincroniza os sinais de controle (explicar com analogia da visão)
- Tempo para a transferência é conhecido – sem sinal de *ack* (podem ser vários ciclos de relógio)

- Procedimento:



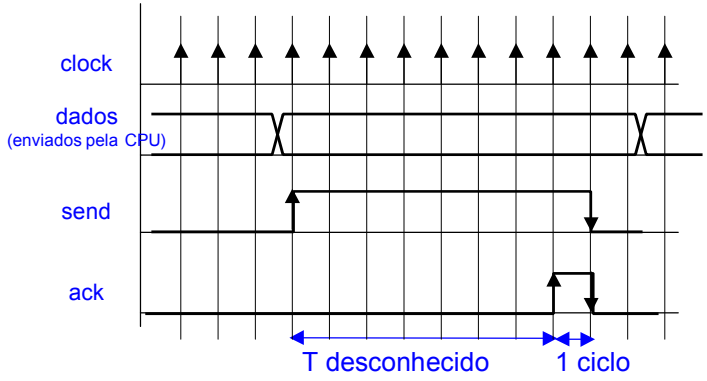
1. Quando ck ↑, CPU coloca dados no barramento
2. Quando ck ↓, send ↑ significando que há dado a ser transferido
3. Quando ck ↑ novamente terminou a transferência (send ↓)

Perguntas

- Porque o send é necessário?
  (*porque há períodos sem transmissão de dados)*

2.3 Transferência Semi-Síncrona

- Mesmo clock para CPU e periférico (não adianta a mesma freqüência !!!)



- Clock sincroniza os sinais de controle
- Tempo para a transferência é **desconhecido**

Procedimento

1. CPU disponibiliza dados e **sincronamente** ativa o sinal **send** (este sinal pode ficar ativo ***n*** ciclos, até periférico respoder)
2. Na primeira transição de subida do clock, após o periférico ter armazenado os dados, o **ack** é ativado pelo periférico

3. No ciclo de clock seguinte remove-se o **send** (CPU) e **ack** (periférico)

Perguntas

1. De um exemplo onde **time-out** é necessário.
   (*no caso de um falha entre 1 e 2*)
2. O periférico depois de subir o **ack** o abaixa novamente. Como ele sabe que a CPU já detectou o sinal?
   (*sincronismo de clocks*)
3. Qual a diferença para a transferência assíncrona?
   (*mesmo clock, eliminação do último laço de controle*)

3 MAPEAMENTO DE ENTRADA/SAÍDA

- Como endereçar as portas dos dispositivos?

3.1 Em memória (memory mapped)

- Um único espaço de endereçamento
- Destina-se um conjunto de endereços aos periféricos
- Instruções à memória podem ser tanto memória quanto operações de entrada/saída
- O mapeamento em memória tem a **vantagem** de permitir uma maior proteção ao acesso direto a dispositivos, pois os endereços de E/S podem ser controlados pelo sistema operacional.
- Exemplo: processadores da Motorola

3.2 Em portas de entrada/saída

- Dois espaços distintos de endereçamento
- Entrada saída acessada por instruções específicas (IN, OUT)
- O µp dispõe ou de um pino IO/M (High→IO, Low→M) ou 2 conjuntos de pinos independentes
- Havendo 2 conjuntos pode-se sobrepor com o acesso à memória
- Exemplo: µp da INTEL
  IN AL, porta
  OUT porta, AL
  (64Kb portas E/S de 8 bits, 32 Kb com 16 bits)
- Exemplo de programação entre os dois métodos:
  Supor que os endereços dos registradores do controlador de impressão sejam 2F8H (caracter) e 2F9H (status). Dois bits de status: s1=1 indica erro e s0=1 indica caracter sendo impresso. Trecho do programa para imprimir um caracter:

Mapeado em memória

```
le_status   mov AL, 2F9H
            or AL, 00
            jnz le_status
            mov AL, 65H
            mov 2F8, AL
```

Mapeado com portas

```
le_status   in AL, 2F9H
            or AL, 00
            jnz le_status
            mov AL, 65H  (caracter "A")
            out 2F8, AL
```

- O mapeamento em **portas** permite um acesso direto aos dispositivos de E/S, oferecendo um **maior desempenho**. A proteção por parte do S.O. fica dificultada.

4 MODOS DE TRANSFERÊNCIA

- Quando efetuar a transferência?

Características desejadas:

- Rápida resposta a eventos críticos: eventos de segurança requerem resposta imediata
- Não sobrecarregar a CPU com atividades de E/S: algumas atividades como acesso a disco e *refresh* de memória exigem altas taxas de E/S
- Três modos: **programado**, **interrupção** e **DMA**

4.1 Entrada/saída Programada

- E/S é controlada pela CPU (registradores e instruções específicas).
- O processador pergunta para cada periférico se este está apto a receber ou transmitir uma unidade de informação (geralmente um byte), e em caso afirmativo realiza a transferência.
- Tipos: modo bloqueado (busy wait), polling (inquisição) e interjeição

4.1.1 Modo Bloqueado

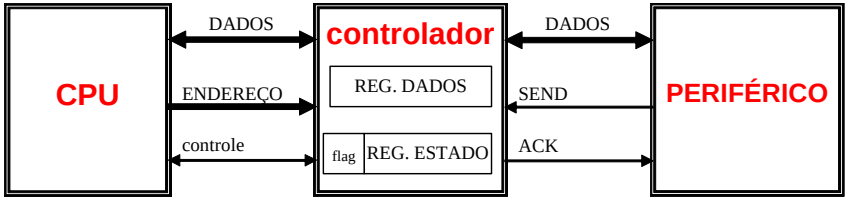
- Uma vez iniciada a comunicação, a CPU fica "ocupada" (escrava) até o término da operação.
- CPU é subutilizada, pois periféricos são muito mais lentos que a CPU.

- Exemplo de utilização: computador dedicado a, por exemplo, uma balança. A única atividade da máquina é pesar.

4.1.2 Polling

- *POOL – piscina, POLL – apuração de votos*
- A CPU periodicamente testa o conteúdo dos bits do(s) registrador (es) de estado existente(s) nos controladores de E/S
- Em cada controlador há registradores que indicam transferencia a ser realizada (*status*).

- No modo polling a CPU possui controle total, realizando todo o trabalho.
- O que faço com o tempo se não tenho a resposta para continuar o programa que espera por E/S?



- Protocolo assíncrono entre **controlador** e **periférico**:
  1. Periférico transfere dados para o controlador, ativando o sinal "send";
  2. Controlador armazena o dado em um registrador de dados, enviando o sinal "ack", ligando um *flag* de "dado presente" no registrador de estado;
  3. Controlador remove o "ack" *quando* o flag for zero (CPU já buscou)
  4. Periférico remove o "send" quando ack for zero, habilitando novo envio de dados;
- **CPU ⇔ Controlador**
  1. CPU tem um programa que é executado periodicamente (*polling*) para o teste do flag de "dado presente" no registrador de estado;
  2. Quando flag=1, CPU lê o dado do controlador, armazenando em um registrador;
  3. Após ler o dado: flag←0.
  4. Periférico pode enviar novo dado.

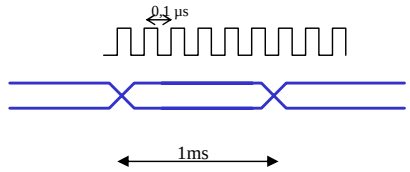
- Observação: **overrun error**
  - No protocolo acima só são enviados dados quando *ack*=0. Uma variação do protocolo acima é permitir que o periférico envie dados assim que receber o *ack*. Se a CPU for mais lenta que o periférico (muito difícil) pode acontecer de dados novos sobre-escreverem dados que ainda não foram lidos, ocasionando erro

Cálculo de tempo desperdiçado – exemplo 1

- CPU lê o *flag* de status a cada 0,1 µs (10 MHz)
- E/S transfere dados a 1 Kb/seg – largura do bus: 1 Byte
- Pergunta-se: quantas leituras "desnecessárias" são feitas ao *flag* de status para 1 transferência?

R: taxa de leitura do flag: 0.1 µs

taxa de transferência: 1 Byte a cada 1ms



$$\frac{1ms}{0{,}1\mu s} = \frac{10^{-3}}{10^{-7}} = 10^{4}$$

ou seja 9999 leituras sem sucesso para 1 com sucesso

Cálculo de tempo desperdiçado – exemplo 2 – H/S interface p. 568

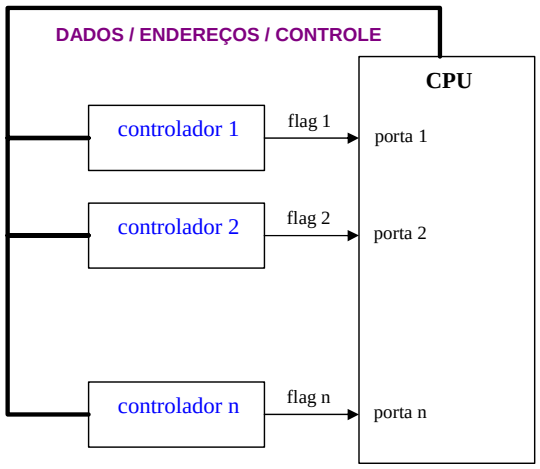
- Supor uma CPU operando a 50 MHz, executando uma rotina de polling que consuma 100 ciclos. Pede-se:

a) Tempo de CPU consumido para verificar o mouse, sabendo-se que este deve ser verificado 30 vezes/seg.
   - Ciclos gastos pelo mouse: 30*100 = 3000 ciclos/seg
   - % de tempo CPU: $3000*100/50*10^6$ = 0,006 % do tempo de CPU
   - conclusão: mouse não incomoda

b) Tempo para transferir dados de um disquete, com taxa de 50 KB/ seg, transferindo-se 2 Bytes por vez.
   - Ciclos gastos pelo disquete: ($50*2^{10}$ Bytes/ 2 Bytes) * 100 = 2560000 ciclos/seg
   - % de tempo CPU: $2560000*100/50*10^6$ = 5,12 % do tempo de CPU
   - conclusão: tolerável

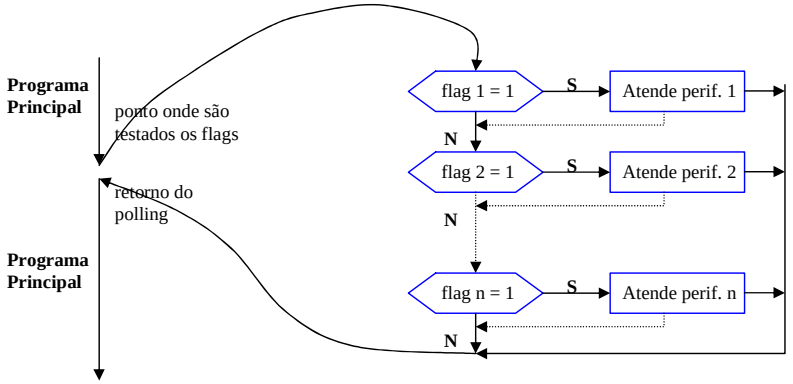
c) Tempo para transferir dados de um winchester, com taxa de 2 MB/ seg, transferindo-se 4 Bytes por vez.
   - Ciclos gastos pelo HD: ($2*2^{20}$ Bytes/ 4 Bytes) * 100 = $52,46*10^6$ ciclos/seg
   - % de tempo CPU: $(52,4*10^6/50*10^6)*100$ = 105% do tempo de CPU
   - conclusão: impossível !!!

- **CPU com várias portas de E/S (ex: 8051 possui 4 portas de 8 bits)**



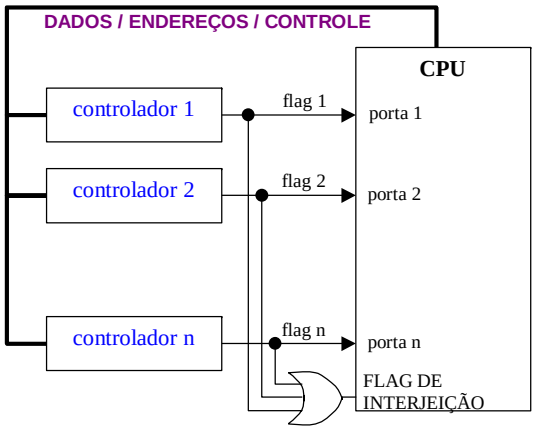
- Verificação dos *flags*:
  - Prioridade implícita
  - Normalmente atende o primeiro periférico com flag=1 e termina a rotina de polling
  - Pode acontecer de ter periférico "esquecido"
  - Linha tracejada: opcional, que significa teste dos demais periféricos, evitando o "esquecimento" de periféricos
  - Esta solução pode causar muito atraso para a CPU
- Projetista decide em função de:
  - Número de periféricos
  - Freqüência estimada dos pedidos de atendimento
  - Freqüência de teste dentro do programa principal

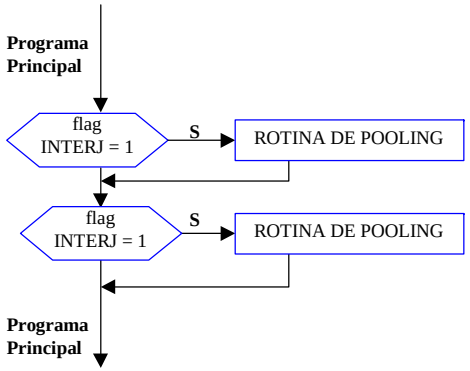
- Polling é muito utilizado em ambientes industriais, para monitorar, por exemplo, grandezas físicas



4.1.3 Interjeição

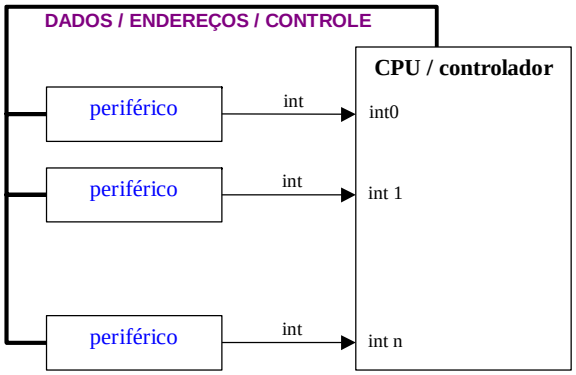
- Otimização do polling
- Ou lógico de todos os flags de stats ➔ menor tempo com o teste
- CPU só testa um bit, independentemente do número de periféricos





4.2 Interrupção (H/S 569)

- Interrupção foi criada para solucionar o problema do tempo desperdiçado com múltiplos testes que é inerente ao polling
- CPU/controlador é avisado pelo periférico que este está pronto para transmitir/receber dados
- Principais características de interrupções de E/S:
  - Assincronismo em relação a qualquer instrução, ocorrendo a qualquer instante
  - Na ocorrência de uma interrupção, a mesma é atendida após o término da instrução corrente. O teste de interrupção é feito depois da execução da instrução
  - Diferenciar interrupção de hardware de exceções (interrupções do hardware interno, exemplo, divisão por zero)
  - Dispositivos podem ter diferentes prioridades
- Havendo um periférico, ao menos, pedindo interrupção, inicia-se o ciclo de tratamento de E/S (varia de µp para µp):

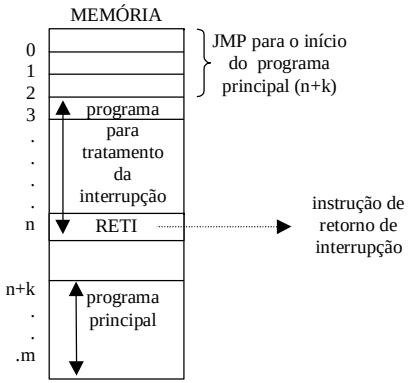


1. Interrupções são temporariamente desabilitadas
2. Salva-se o contexto (PC, registradores) em um pilha
3. Identifica-se o periférico, colocando no PC o endereço da rotina de tratamento da interrupção
4. Executa-se o programa de interrupção
5. Recupera-se o contexto
6. Interrupções são novamente habilitadas

- Procedimento análogo à chamada de subrotina

Exemplo: microcontrolador 8051 (Intel)

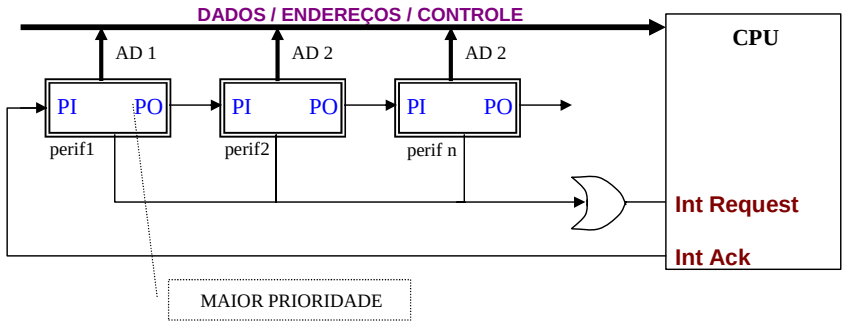
- 4 pinos dedicados à interrupção, a qual pode ser habilitada ou não por software (INT0, INT1, TIMER0, TIMER1).
- Quando há pedido de interrupção no pino INT0, o PC é posto em uma pilha, e PC recebe valor 3.



TRATAMENTO DE DIVERSAS INTERRUPÇÕES SIMULTÂNEAS

- INTERRUPÇÃO COM POLLING
  - Opera do mesmo modo que na interjeição: ou lógico de todos os pedidos de interrupção é enviado ao processador/controlador
  - Lento, pois são feitos testes seqüenciais para atendimento dos periféricos

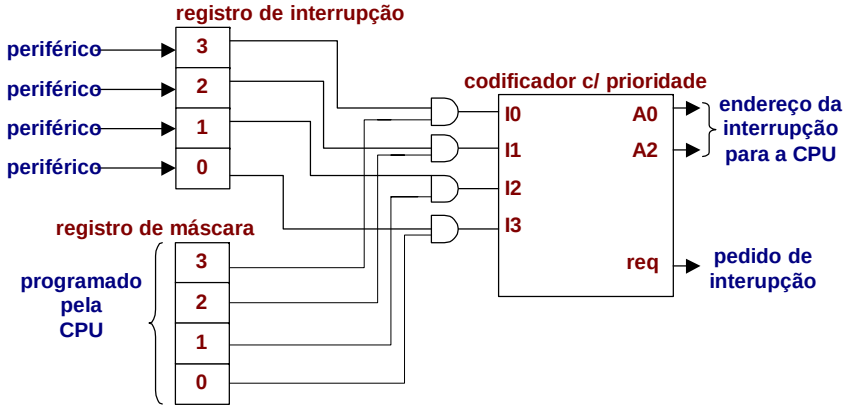
- INTERRUPÇÃO DAISY CHAIN, ou serial
  - Conexão em série dos dispositivos de E/S (prioridade implícita)
  - Exemplo: SCSI
  - Funcionamento:
    1. Quando há um pedido de interrupção, a CPU após aceitá-lo habilita o sinal "int ack"
    2. O primeiro dispositivo que tiver o pedido de interrupção ativo, envia o seu endereço para a CPU, desabilitando os demais periféricos (propagando PO com 0).



legenda: PI: prioridade IN, PO: prioridade OUT, AD: endereço

  - Método simples, pois um dispositivo se conecta ao outro, simplificando a fiação

- INTERRUPÇÃO PARALELA
  - Vários periféricos podem solicitar simultaneamente interrupções, como nos métodos anteriores, porém um codificador de prioridade selecionará qual interrupção atender.
  - Organização do controlador de interrupção:



  - Mascaramento: possibilidade de habilitar ou não determinada interrupção
  - O controlador de interrupções do PC (8259A) opera desta forma. As prioridades são fixas, apesar do hardware original prever alterações de prioridades

INTERRUPÇÕES NO x86 (INTEL)

- Três tipos de interrupções são utilizados:

a) Externas (HARDWARE)
  - Teclado, impressora, placa comunicação, …
  - São ativadas através de pinos do µp: NMI (não mascarável) e INTR
  - Exemplo de não mascarável: falta de luz ou paridade em memória
  - Controlador 8259A (1 XT, 2 AT)

b) Software (traps)
  - Não são verdadeiras interrupções, pois são chamadas por software
  - Instrução típica: INT xx
  - Exemplo:

```
ASSEMBLY:
mov  AH, 09H
mov  DX, offset ´alo mamãe´
int 21H

c:
#include <dos.h>

void main( void )
{
 union REGS pregs;
 struct SREGS sregs;
 char Message[20] = "Fernando$";   /* $ fim de string */

 pregs.h.ah = 0x09;
 sregs.ds = FP_SEG( Message );     /* Get the var. */
 pregs.x.dx = FP_OFF( Message );   /* addresses    */
 intdosx( &pregs, &pregs, &sregs );
}
```

c) Internas ao processador
  - Também denominadas **exceções**, exemplo: divisão por 0

VETOR DE INTERRUPÇÕES

- Primeiro 1k de memória, sendo reservado 4 bytes por interrupção, resultando em 256 endereços para interrupções
- Cada endereço aponta para uma função armazenada na BIOS (ou definida pelo programador). Este endereço para a função é denominado de "entry point"
- Estes endereços podem ser modificados, permitindo utilizar um outro programa para tratar determinada interrupção

BIOS

- Programa residente que tem por principal função ativar as interrupções (dispositivos de E/S)
- Garante a **compatibilidade** entre os diferentes PCs, pois apesar do hardware de cada um ser diferente, a BIOS se encarrega de realizar a interface entre o software e a máquina
- Quando o PC é ligado, há um salto para a posição de memória FFF0 (16 posições abaixo de 1M), verificando-se todo o hardware.

TABELA DAS INTERRUPÇÕES

Absolute address (hex)	Interrupt value	Function	Hardware interrupt
internas ao processador			
0000:0000	00H	Divide by Zero Interrupt	
0000:0004	01H	Single Step Interrupt	
0000:000C	03H	Breakpoint	
0000:0010	04H	Arithmetic Overflow Handler	
não mascarável			
0000:0008	02H	Non-Maskable Interrupt	
mascarável			
0000:0028	0AH	VGA Retrace (AT Slave)	IRQ2
0000:002C	0BH	Serial Port 2	IRQ3
0000:0030	0CH	Serial Port 1	IRQ4
0000:0034	0DH	Hard Disk	IRQ5
0000:0038	0EH	Floppy disk	IRQ6
0000:003C	0FH	Parallel Port	IRQ7
0000:01C0	70H	Real Time Clock	IRQ8
0000:01C4	71H	LAN Adapter	IRQ9
0000:01C8	72H	Reserved	IRQ10
0000:01CC	73H	Reserved	IRQ11
0000:01D0	74H	Mouse	IRQ12
0000:01D4	75H	80287 NMI Error	IRQ13
0000:01D8	76H	Hard disk controller	IRQ14
0000:01DC	77H	Reserved	IRQ15

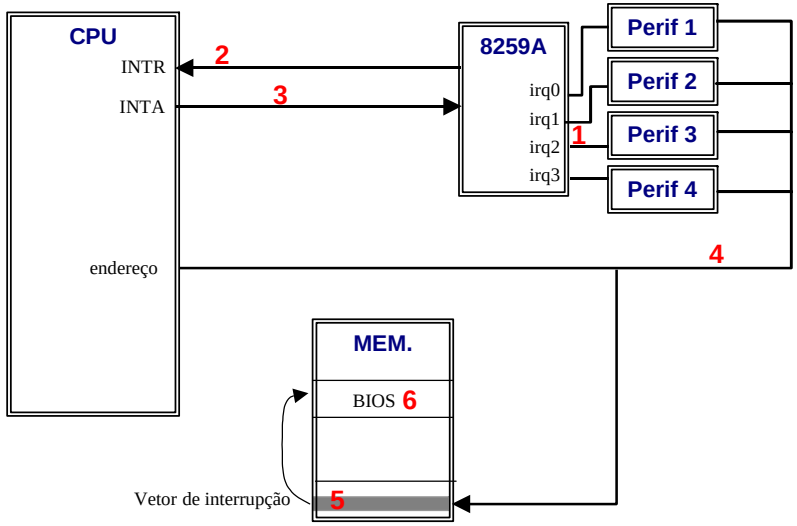
- No windows pode-se exibir como estão atribuídas as interrupções:

TRATAMENTO DAS INTERRUPÇÕES

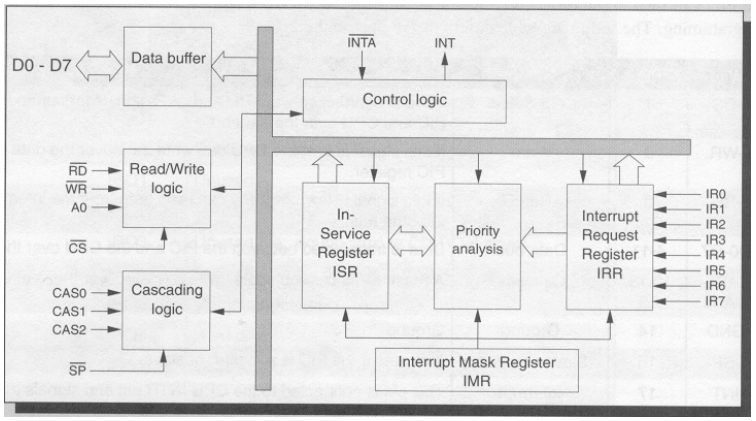
- O procedimento de tratamento das interrupções em PC's segue o seguinte procedimento:

1. Periférico faz requisição ao controlador de interrupção, através das linhas IRQ. Se a interrupção está habilitada, o controlador enviara um sinal ao processador
2. Controlador de interrupção repassa pedido de interrupção ao processador (INTR)
3. Processador envia confirmação de aceite da interrupção (INTA) que a controladora de interrupção repassa ao periférico em questão

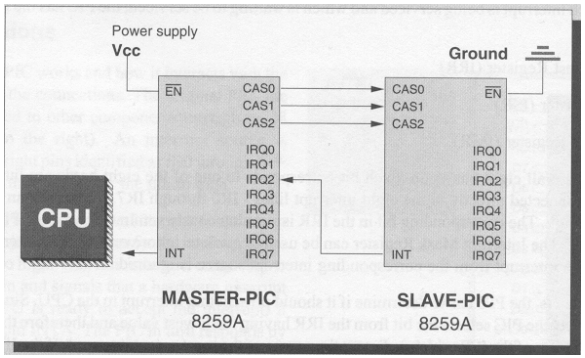
4. Periférico coloca seu endereço/dados no barramento
5. O endereço referencia uma posição de memória no vetor de interrupção (primeiro 1K de memória)
6. Esta posição de memória apontará para o endereço da rotina de tratamento de interrupção, situada normalmente na BIOS



- **Controlador de Interrupção (PIC)**



- **Cascateamento de dois PIC´s**



- Registradores:
  - ISR: interrupção que está sendo atendida
  - IRR: quem está pedindo interrupção
  - IMR: máscara de interrupção
- Acesso aos registradores:
  - Endereços do Master: 20H e 21H
  - Endereços do Slave: A0H e A1H
- Leitura dos registradores:
  - Para ler o IRR e o ISR primeiro envia-se uma palavra de controle para a porta e depois lê-se o estado da porta.
  - A rotina de tratamento de interrupção **deve** avisar ao PIC que a interrupção terminou. Se a interrupção pertence ao SLAVE, avisar os 2 PICs. EOI = end of interruption
  - A máscara de interrupção encontra-se no endereço + 1. Por *default* todas as interrupções estão habilitadas (iguais a zero). Pode-se ler ou escrever neste registrador para habilitar ou não as interrupções

COMO UTILIZAR INTERRUPÇÕES:

1. TECLADO: Interceptar o teclado, por exemplo, para reconhecer hot-keys

```
#include <stdio.h>
#include <dos.h>

#define INTKEY  0x09

int cont;

void interrupt New_Key_Int();    /* interrupt prototype */
void interrupt (*Old_teclado)(); /* interrupt function pointer */

void interrupt New_Key_Int()
{ cont ++;
  printf("Tecla %d\n", cont);
  Old_teclado();
}

int main(void)
{
  cont =0;

  Old_teclado =_dos_getvect(INTKEY);

  _dos_setvect(INTKEY, New_Key_Int);

  for( ; cont < 10; ); /* espera 10 teclas */

  _dos_setvect(INTKEY, Old_teclado);

  puts("OK");
  return(1);
}
```

ARMAZENA O ENDEREÇO DA ROTINA ORIGINAL DO TECLADO

ATRIBUI NOVA FUNÇÃO

RECUPERA FUNÇÃO ORIGINAL

2. Shift PrtSc:

```
#include <stdio.h>
#include <dos.h>

void interrupt get_out();    /* interrupt prototype */
void interrupt (*oldfunc)(); /* interrupt function pointer */

int looping = 1;

int main(void)
{
  puts("Press <Shift><PrtSc> to terminate");

  oldfunc  = _dos_getvect(5);          /* save the old interrupt */

  _dos_setvect(5,get_out);             /* install interrupt handler */

  while (looping);                     /* do nothing */

  _dos_setvect(5,oldfunc);             /* restore to original interrupt routine */

  puts("Success");
  return 0;
}

void interrupt get_out() {
  looping = 0;             /* change global var to get out of loop */
}
```

4.3 DMA – transferência direta à memória

- Interrupção libera a CPU da tarefa de aguardar por ocorrência de evento de E/S, porém a CPU continua sendo o elemento ativo (responsável por realizar a transferência de dados)
- Solução: utilização de DMA. Dispositivo controlador (pode ser outro processador) é o responsável pela transferência de dados, ficando a CPU livre para realizar outras operações
- **Importante**: mecanismos de interrupção continuam sendo utilizados pelo controlador para comunicação com o processador, mas apenas no término de um evento de E/S, ou na ocorrência de erros
- Durante a operação o controlador de DMA se torna o mestre do barramento e controla a transferência E/S ⇔ Memória
- Existem 3 passos em uma transferência DMA:

1. CPU programa a controladora de DMA com:
   - Identificação do dispositivo que solicitou DMA
   - Operação a ser realizada no dispositivo (escrita/leitura)
   - Endereço origem e destino (na memória principal e no periférico)
   - Número de bytes a serem transmitidos
2. O controlador de DMA assume o barramento, iniciando a operação. A transferência inicia quando o dado está disponível (no dispositivo ou memória). O controlador de DMA fornece o endereço de leitura ou escrita na memória. Se a requisição necessita mais de uma transferência no barramento, o DMA gera o próximo endereço de memória. Por meio deste mecanismo o DMA pode transferir milhares de bytes do disco para a memória, sem interromper a CPU
3. Concluída a transferência de DMA, o controlador interrompe o processador, que poderá verificar se a operação foi realizada com sucesso examinando a memória ou interrogando o controlador de DMA

"uma vez o DMAC programado, a CPU e o DMAC operam em paralelo"

- Em um sistema computacional podem existir diversos controladores de DMA. Em um sistema do tipo processador, memória com mesmo barramento, e diversos dispositivos de E/S, poderá existir um controlador para cada dispositivo.
- Exemplo – Supor uma CPU operando a 50 MHz, com HD transferindo 2MB/seg, utilizando DMA.
  - Duração da programação do controlador de DMA pela CPU: 1000 ciclos de clock
  - Duração da rotina de interrupção para tratar término do DMA: 500 ciclos de relógio
  - Na média, cada operação de transferência do disco manipula 4KB (algumas 10K, outras 1K, …)
  - Supondo que o disco estará transferindo dados para a memória continuamente (100%), qual a fração de tempo da CPU é consumida para a operação?
- Resposta:

  tempo necessário para transferir os 4 Kbytes de uma operação de DMA:

  se o disco produz $2*10^6$ Kbytes (2 MB) a cada segundo preciso de $2*10^6/4*10^3$= **500 transferências** de 4 Kbytes por segundo para tratar os dados

  a cada transferência preciso 1500 ciclos de CPU

  portanto em um segundo são gastos 500 * 1500 ciclos = $75*10^4$ ciclos/s

  consequentemente: % de CPU = $75*10^4$ * 100 / $50*10^6$ = 1,5 %
- Comentários
  - Comparando com polling e interrupção, DMA deve ser utilizado para interface de periféricos que necessitem de grande vazão (por exemplo, disco rígido)
  - Se a CPU precisar do barramento durante o DMA, ocorrerá atraso, uma vez que a memória está ocupada
  - Utilizando caches no processador o desempenho irá melhorar, uma vez que o processador não precisará aguardar pela liberação do barramento

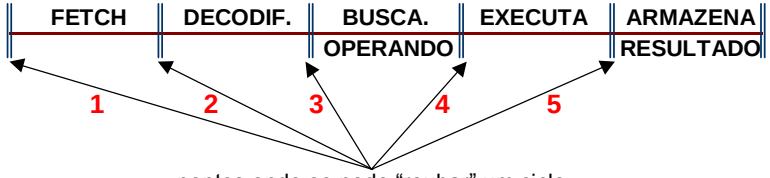
- Problema do DMA: conflito de barramento, pois a CPU também precisa do barramento para buscar as instruções.
- Soluções:
  - HALT (transferência de bloco)
  - Roubo de Ciclos (*cycle stealing*)
  - Entrelaçado (*interlevead*)

4.3.1 Halt

- Método simples, onde DMAC suspende o processador (ganha o barramento)
- Pino de HOLD
- Vantagem? CPU pode realizar operações que não fazem acesso ao barramento
- Passos:
  1. Uma vez programado o DMAC, este envia sinal de HOLD para a CPU
  2. CPU conclui instrução corrente e envia HOLDACK
  3. DMA recebendo HOLDACK tem aceso total ao barramento
  4. Ao término da transferência o DMA remove o sinal HOLD

4.3.2 Roubo de Ciclo (*cycle stealing*)

- envio palavra a palavra (ou em blocos)
- DMA pede uso do barramento, e CPU o entrega **entre** ciclos de máquina.
- Passos:
  1. DMAC pede bus, assincronamente ➔ DMAREQ
  2. CPU termina ciclo corrente e envia DMAACK, liberando o barramento
  3. DMAC faz uma transferência e remove DMAREQ
  4. Terminada TODA a transferência, DMAC avisa a CPU por interrupção



pontos onde se pode "roubar" um ciclo

- Obs: a quantidade a transferir em cada "roubo" é programada pela CPU

4.3.3 Entrelaçado (*interlevead*)

- DMAC disputa o barramento com a CPU para fazer acesso à memória
- Controladora do barramento pode dar prioridade a CPU para evitar atrasos

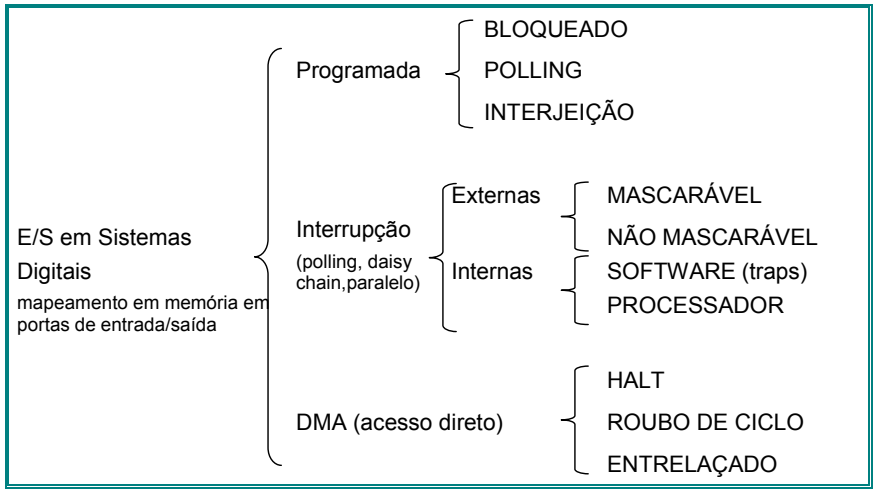
Estudo de Casos - DMA no PC

- Utiliza o controlador 8237A
- Exemplo de uso: HD e "*sound blaster*"
- Na "*soud blaster*"
  - CPU programa DMAC
  - DMAC recebe sinal "go" da CPU
  - DMA realiza leitura na RAM e envia para a placa
  - CPU enquanto isto pode realizar outras operações que não envolvam barramento
- Antes de iniciar uma transferência é necessário definir:
  - Página da memória a ser utilizada
  - Deslocamento na página
  - Quantidade de dados a ser transferida
- Limitações do DMA no PC:
  - Máximo 64 KB por vez (razão: o contador de bytes transferidos do DMAC contém 16 bits).
  - Só pode transferir dentro da mesma página.
  - Velocidade: apenas 6 MHz (para a época do XT era OK), logo para um processador de 200 MHz não faz muito sentido utilizar DMA
- HDs/placas de som/refresh (15 µs) de memória ainda utilizam DMA
- *Conclusão: novos padrões de DMA no PC, afim de aumentar o desempenho:*

Fast Multiword DMA: *padrão de DMA permitido pelo chipset Triton da intel que utiliza um modo de transferência de dados entre disco rígido e memória, chamado PIIX (PCI-ISA IDE Xcelerator). Este modo utiliza o Fast Multiword DMA que transfere 48 bits por vez ao invés dos 16 bits originais. Com isto a transferência pode chegar à 16,6 MB/s com a vantagem de não utilizar o processador, ao contrário do esquema PIO.*

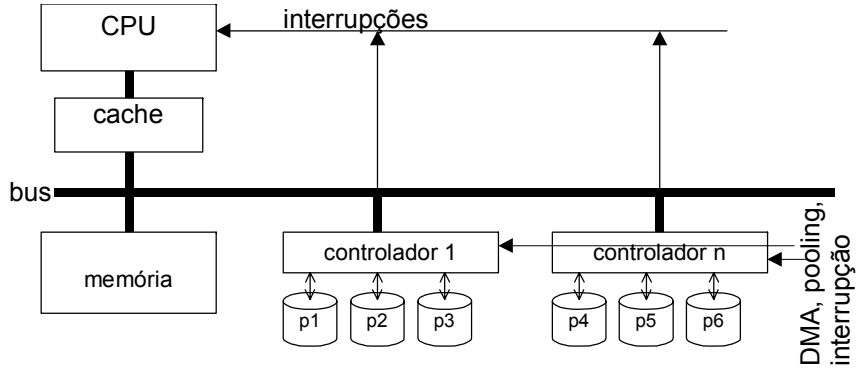
Ultra DMA ou Ultra DMA/33 : *é um protocolo para transferência de dados entre o disco rígido e a memória principal (RAM). O protocolo Ultra DMA/33 transfere dados no modo de rajadas (burst mode) a uma taxa de 33.3 Mbps (megabytes por segundo). Foi desenvolvido pela Quantum Corporation e Intel para tornar-se um padrão da industria. O Ultra DMA foi introduzido em computadores no final de 1997 com o objetivo de torná-lo um padrão de interface para disco rigido em 1998. O Ultra DMA utiliza CRC (Cyclical Redundancy Checking) para proteção dos dados.*

RESUMO E/S:



5 BARRAMENTOS

- Contexto:



- Canal de comunicação compartilhado composto por um conjunto de fios para a conexão de subsistemas.
- Utilizado para conectar CPU, memória e periféricos
- Possibilita a **expansão** do computador de forma organizada
- Vantagens da utilização: versatilidade e baixo custo
  - Versatilidade: ao se definir um sistema único para conexão (padrão), novos dispositivos podem ser adicionados/movidos entre sistemas computacionais que utilizam o mesmo padrão de barramento

  - Baixo custo: um único conjunto de fios é utilizado para conectar diversos dispositivos (compartilhamento)
- Maior desvantagem: perda de tempo na disputa (contensão)
  - O barramento é o gargalo do sistema, uma vez que **não** pode ser utilizado para realizar mais de uma transferência de forma simultânea
- Operações realizadas no barramento:
  - Entrada ou saída: (1) envio do endereço, (2) envio ou recepção do dado (ou controle)
- Restrições físicas: número de linhas (fios) e tamanho do barramento
  - Barramento multiplexado: mesmos fios para dados e endereços
  - Não multiplexado: separa linhas e endereços
  - Largura do barramento: desejável ser igual à largura da palavra
- Importância: Evolução da velocidade da CPU *versus* velocidade do barramento.
  - Supor um programa que consuma 90 seg de CPU e 10 seg para E/S. Se a velocidade da CPU melhora 50% por ano e a E/S permanece constante, qual o ganho de desempenho em 5 anos?

Ano	CPU (seg)	E/S (seg)	Total (seg)
0	90	10	**100**
1	60 (÷1,5)	10	70
2	40	10	50
3	27	10	37
4	18	10	28
5	12	10	**22**

Ganho CPU: 90/12 = 7.5

Ganho total: 100/22 = **4.5**

E/S representa inicialmente 10% do tempo total, e no quinto ano 45% (10/22).

- Padrões:
  - Simplificam a indústria, pois o fabricante produz um periférico para uma determinada norma e não para o equipamento
  - Exemplos: NuBus, Sbus, SCSI, EISA, VESA, PCI, PCMCIA, …
  - O tipo de barramento depende muitas vezes da aplicação: multimídia, vídeo, notebook, memória

5.1 Tipos de barramentos

- Processador – memória
- Entrada/saída
- Back-plane (“pano de fundo”)
- Back-side cache

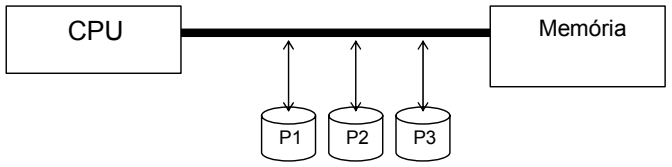
- **Processador – memória (*front-side*):**
  - Curtos, alta velocidade
  - Projetados de acordo com o sistema de memória da placa
  - Proprietários do fabricante
- **Entrada / Saída**
  - Padrão seguido por diversos fabricantes. Exemplo: SCSI, NuBus
  - Diversos dispositivos de E/S conectados
  - Normalmente não conectam diretamente periféricos ao sistema de memória
  - Taxas de transmissão variadas, longos
- **Backplane**
  - Pano de fundo para a ligação de outros barramentos (espinha dorsal – *backbone* em redes)
  - Projetados para possibilitar a ligação de vários grupos de dispositivos de E/S através de um único adaptador ao barramento P/M
  - Permite com isto maximizar a velocidade do barramento P/M
- **Back-side cache**
  - Utilizado para conectar a cache diretamente ao processador
  - Funciona muitas vezes na mesma freqüência do processador
  - Porque chamar de barramento, canal compartilhado? É uma porta

5.2 Arquiteturas de Barramento

- Os dispositivos de E/S de um sistema podem estar ligados ao processador de três formas:
  - Arquitetura de barramento único
  - Arquitetura de barramento em dois níveis
  - Arquitetura de barramento hierárquica

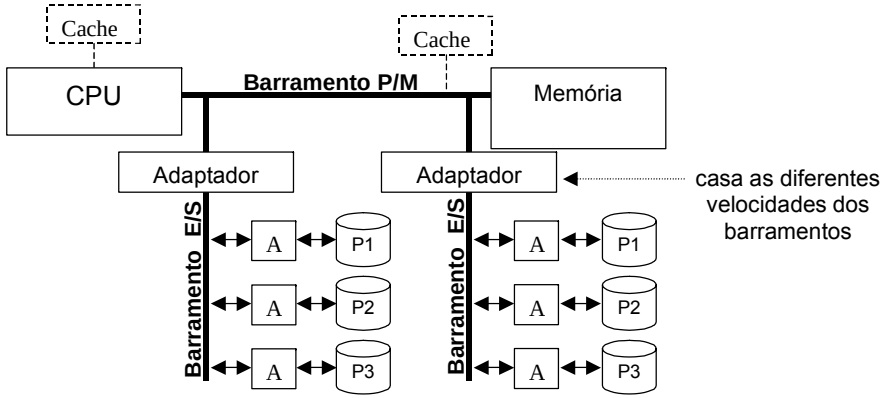
5.2.1 Barramento Único

- Memória e dispositivos de E/S estão ligados a CPU através de um único barramento
- Forma mais simples de interconexão
- Barramento tem que acomodar dispositivos com características e velocidades bem diferentes e o desempenho da comunicação cai



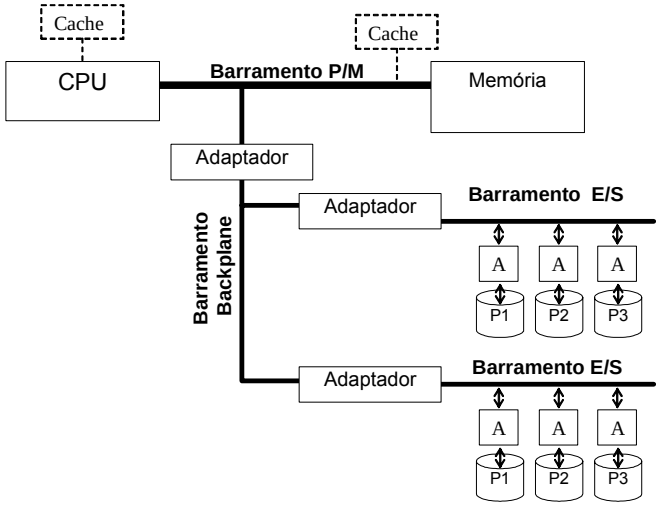
5.2.2 Dois níveis

- Processador e memória se comunicam através de um barramento principal
- Outros barramentos de E/S estão ligados ao barramento principal através de adaptadores, compondo um segundo nível na arquitetura de barramentos
- Desta forma o barramento principal pode funcionar a uma maior velocidade já que os adaptadores se encarregam da comunicação com os barramentos de E/S mais lentos



5.2.3 Hierárquica

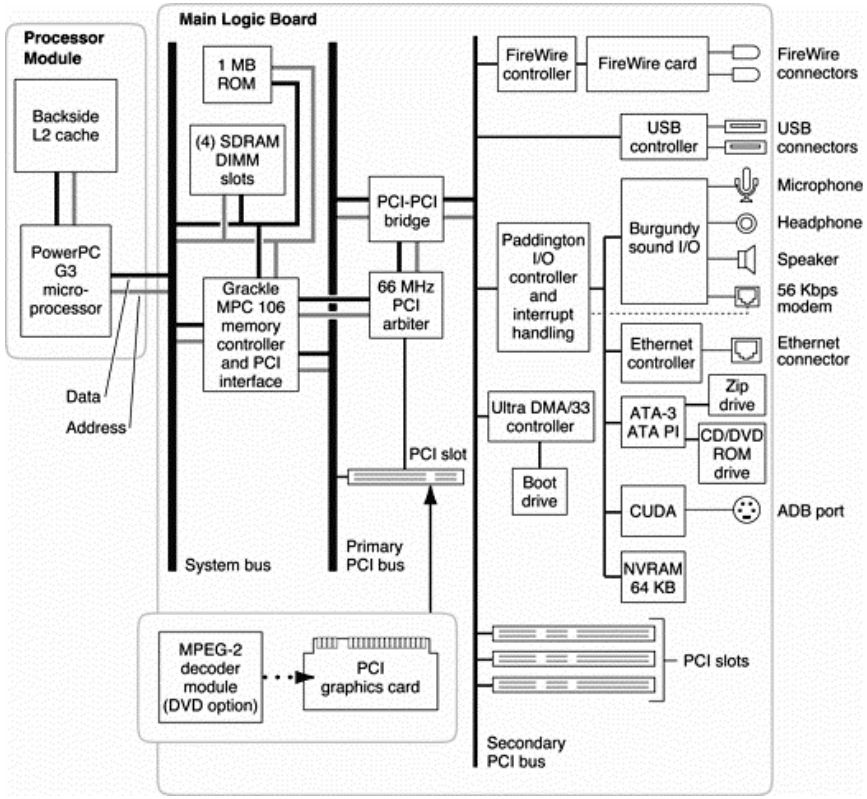
- Processador e memória se comunicam através de um barramento principal
- Um barramento backplane concentra toda E/S do sistema e é ligado ao barramento principal (só **um adaptador** é ligado ao barramento principal)
- Ao barramento backplane estão ligados diferentes barramentos de E/S atrvés de adaptadores



Estudo de Casos - Exemplos para a arquitetura de barramentos hierárquica

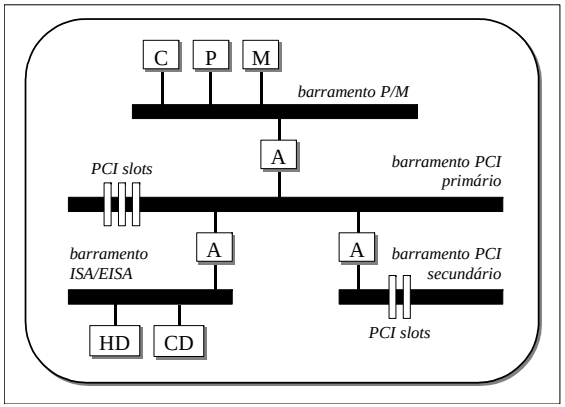
Máquina: Power Macintosh G3

- Processador: Power PC 750
- Ano de fabricação: 1999
- Características: Possui três barramentos, um barramento Processador/Memória e dois barramentos PCI (primário e secundário). O barramento PCI primário é de 32 bits 66MHz e o secundário é de 64 bits 33 MHz



Máquina: Texas Instruments PC Architecture

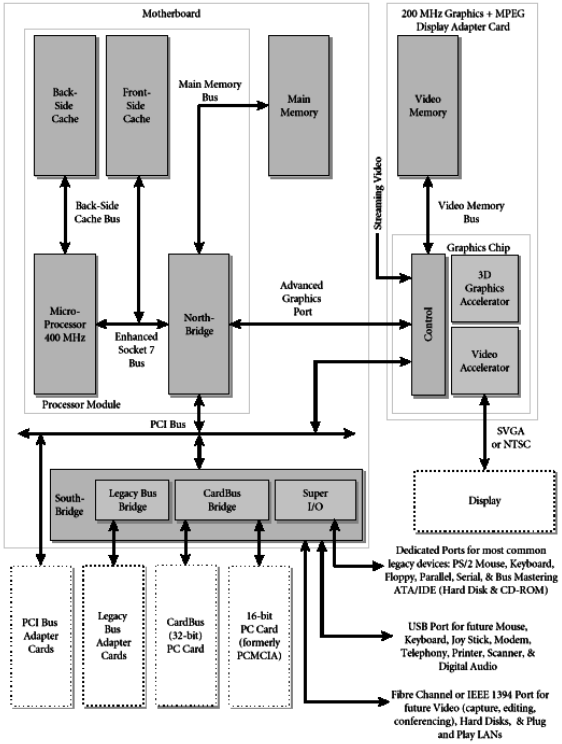
- Processador: Intel
- Ano de fabricação: 1998-1999
- Características: Possui três barramentos, um barramento Processador/Memória e dois barramentos PCI (primário e secundário). No mesmo nível do barramento PCI secundário é usado também um *legacy bus* (barramento ISA/EISA/MCA)



Estudo de Casos - Nova tendência, arquiteturas com chaveadores

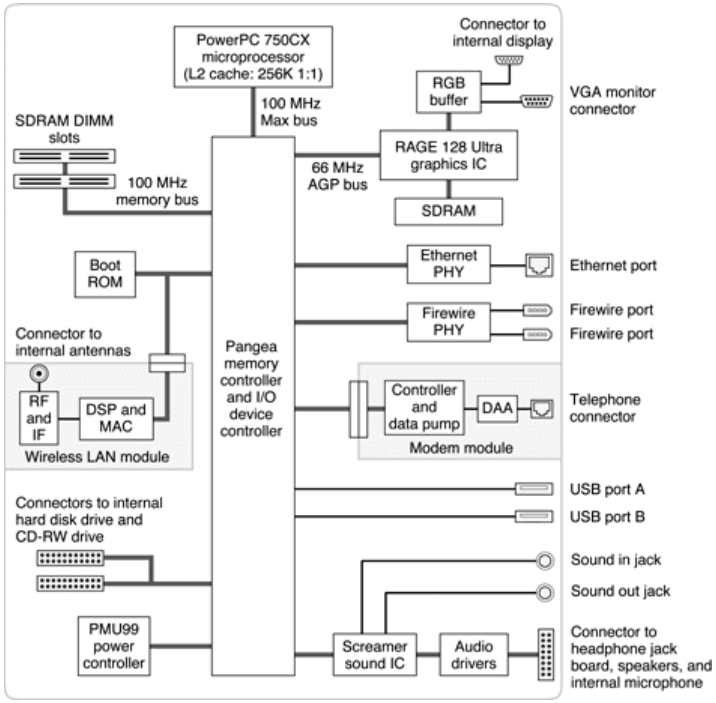
Máquina: PC Graphics platform

- Processador: Intel 400 MHz
- Ano de fabricação: 1998-1999
- Características: Utiliza um chaveador no lugar do barramento P/M (*frontbus*)



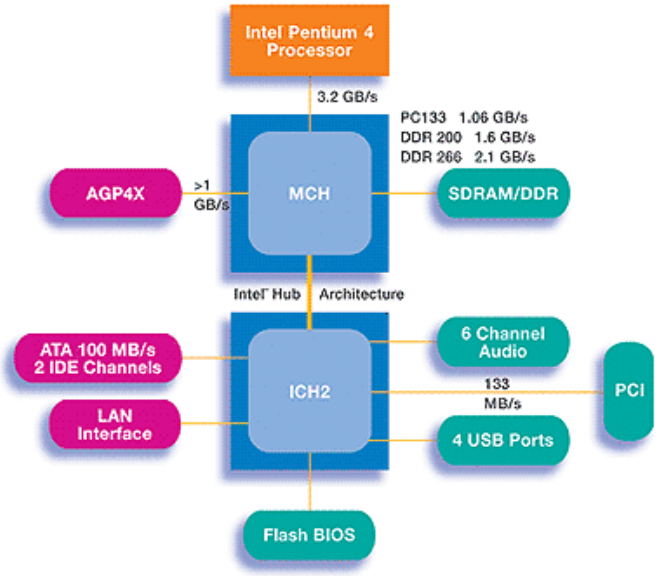
Máquina: Apple iMac

- Processador: Power PC 750-CX
- Ano de fabricação: 2002
- Características: Utiliza um único grande chaveador simplificando a arquitetura e conseqüentemente o design da placa mãe. Máquina compacta sem barramento PCI



Máquina: Arquitetura Intel para Pentium 4

- Processador: Pentium 4
- Ano de fabricação: 2003
- Características: Utiliza dois chaveadores separando os dispositivos em duas classes



5.3 AGP (*Accelerated Graphics Port*)

- Solução encontrada para aproximar a ligação da placa de vídeo do processador por causa de sua importância nas aplicações atuais (Interfaces gráficas, jogos, computação gráfica)
- Representa uma "promoção" para a placa de vídeo (sobe na hierarquia)
- É uma porta e não um barramento, ou seja, uma conexão dedicada a um único dispositivo, sem disputa
- Pode ser visto quase como uma ligação direta ao processador (quase porque normalmente esta ligada na ponte do processador)
- Introduzida no Pentium II, chipset 440LX
- Slot parecido com o PCI, um pouco menor e mais alto
- Largura de 32 bits com 66 MHz tendo uma vazão de 254.3 MBytes/s
- Melhorias no padrão indicadas por 2x (transmissão de 2 dados por ciclo), 4x (transmissão de 4 dados por ciclo) resultando em aumento de vazão
- Cuidado: não adianta acelerar só a conexão entre bridge e placa de vídeo se ambos os extremos não suportarem a mesma vazão (o barramento interno de dados do processador e a placa de vídeo ligada no AGP)

5.4 Acesso ao barramento

- Problema:
  - Como um dispositivo ganha o barramento?
  - Como evitar o caos?
- Solução: "bus master", controla o acesso ao barramento

  - Exemplo: processador controla todos os acessos ao barramento, porém perde-se muito tempo de CPU
- Alternativa: diferentes "bus master"
  - Cada "bus master" pode iniciar uma transação
  - Problema: se mais de um "bus master" pede o barramento ao mesmo tempo?
- Solução: arbitragem do barramento
  - Dispositivo envia um pedido de acesso (*request*)
  - Espera até receber o barramento (*grant*)
- A arbitragem de barramento deve balancear os seguintes critérios:
  - Dispositivos com maior prioridade devem ser atendidos primeiro
  - Não abandonar dispositivos de baixa prioridade
- Alguma soluções para a arbitragem de barramento:
  - Daisy chain: simples, mas não resolve abandono (postergação indefinida). Exemplo: SCSI
  - Arbitragem paralela (centralizada) : cada dispositivo tem sua prória linha de requisição e árbitro decide quem ganha (bus master). **gargalo:** muitos fios
  - Arbitragem distribuída: sem árbitro central, vários requests, candidatos colocam seus códigos no barramento pedindo acesso, e os próprios dispositivos avaliam prioridade. Ex: NuBus (Mac)
  - Arbitragem distribuída com detecção de colisão: como a anterior, porém em caso de colisão espera-se a liberação e tenta-se novamente. Exemplo: *ethernet*

5.5 Desempenho *versus* custo

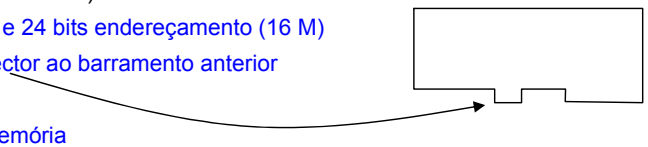
Opção	Alto Desempenho	Baixo Custo
Pinagem	Separa dados/endereço	Multiplexa
Largura dos dados	Grande (ex: 32 bits)	Menor (ex: 8 bits)
Tamanho do Bloco	Múltiplas palavras	Palavra a palavra
Arbitragem	Múltiplos mestres	Único mestre
Relógio	Síncrono	Assíncrono

- Importante: em barramentos há apenas uma transação por vez
- Ideal seria conexão ponto-a-ponto entre os dispositivos:
  - Problema para implementação: grande número de portas nos dispositivos, grande quantidade de pinos e longas distâncias na placa mãe (alto custo)

5.6 Evolução dos barramentos na família Intel

		1# ger.	segunda geração		terceira geração	
	PC-XT	ISA (AT)	MCA	EISA	VESA	PCI
Bits	8	16	32	32	32	32 (64)
Clock	4,77	8	16	16	33	33 (66)
MB/seg	2,38	8	20	33	132	132 (264) 266 (533)

- PC-XT:
  - Barramento é extensão da pinagem do 8088 (8 bits)
  - Mesma freqüência da CPU (4,77 MHz)
  - Acesso ao barramento a cada 2 ciclos de clock (2,38 MB/seg)
- ISA (industry standard architecture) - 80286
  - 16 bits de dados (64k) e 24 bits endereçamento (16 M)
  - Acréscimo de um conector ao barramento anterior
  - 8 Mbytes/seg
  - Problema: acesso à memória
  - Solução (COMPAQ): barramento separado para a memória (dual bus)



- MCA (386, 16 MHz, 4G memória, 32 bits) – 80386
  - Micro channel architecture
  - Introduzido em 1986, tecnologia proprietária
  - Introdução de circuito “bus master” no lugar do processador
  - Assíncrono
  - Conector diferente para evitar problemas
  - 20 MB/seg
- EISA
  - 32 bits de dados
  - Criado por união de grandes empresas (evita royalties do MCA)
  - 33 MB/seg
- VESA (vide electronics standard association)
  - EISA, MCA, PCI substituem ISA
  - VESA complementa ISA
  - Alta taxa de transmissão
  - VESA 2.0 – Pentium 64 linhas de endereço
  - Até 66 MHz, ideal 33 MHz
  - 3 slots no máximo
  - OK para vídeo, discos rápidos e rede

- PCI (peripheral component interface)
  - Lançado em 1992 pela Intel
  - Rápido e independente de CPU. É possível utilizar o barramento PCI com CPUs Intel, Alpha, PowerPC, …
  - Chips seguem padrão do barramento (simplifica lógica e número de chips)
  - Apenas os drivers (software) precisa ser adaptado à CPU destino
  - Norma “aberta”
  - Endereço e dados multiplexados em 32 bits
  - PCI automaticamente sincroniza placas de expansão, endereços de portas, canais de DMA e interrupções. O usuário não mais se preocupa com estes detalhes
  - PCI-X aumentou ainda mais a velocidade do barramento até 4.3 GBytes/s
- IDE/ATA/ATAPI (integrated drive electronics, AT attachment (packet interface))
  - Padrão de interface AT (16 bits) ~1986
  - IDE/ATA originalmente só para discos rígidos, CD-ROM e fitas eram na época proprietários ou SCSI
  - Em 1990 expansão para periféricos diferentes (CD-ROM, Fitas etc.)
  - Evolução: EIDE, ATA-2, Fast ATA, ATA-3, Ultra ATA, Ultra DMA
  - Ultra DMA 80 pinos (40 adicionais são *ground* para poder amostrar o sinal mais rápido)
  - Vazão: 33 MBs, 44 MBs, 66 MBs
- PCMCIA (personal computer memory card industry association)
  - Independente de S.O.
  - Utilizado em PC, Mac, máquinas fotográficas, laptops, …
  - Configurável pro software (sem chaves ou jumpers)
  - Overhead em software
  - Simples, versátil, 26 linhas de dados (64MB)

Barramentos Padrão

	PCI *Periferial Component Interconect*	SCSI *Small Computer System Interface*	USB *Universal Serial Bus*	IEEE 1394 *Firewire*
Ano de Lançamento	1992	1986	1994	1995
Transmição	paralela	paralela	serial	serial
Sincronismo	síncrono (PC´s) ou assíncrono	síncrono	síncrono	Síncrono ou **assíncrono**
Largura (dados)	32, 64	8 (regular), 16 PC´s (wide), 32 (very-wide)	2 (Half-duplex)	2 (half-duplex)
Freqüencia	33, 66, 100 MHz	5 (regular), 10 (fast), 20 MHz (ultra)	8 MHz	**50 MHz → backplane bus 100, 200, 400 MHz → cabo**
Vazão	132, 264, 528 MB/s	5, 10, 20, 40 MB/s	12 Mb/s	50 Mb/s → backplane bus 100, 200, 400 Mb/s → cabo
Uso	barramento de E/S ou *Backplane* interno	barramento de E/S interno e externo	barramento de E/S externo	barramento de E/S ou *Backplane* externo
Linhas no cabo	---	25	4	4 (sem alimentação), 6 (com alimentação)
Comprimento do cabo	---	6 m no máximo	5 m no máximo	16 segmentos de 4.5 m (com alimentação) ou de 25m (sem alimentação)
Ligação	---	em série (linear)	árvore	árvore
Terminação	---	necessária	desnecessária	desnecessária
N° de Dispositivos	até 32	até 16	até 127	até 63
Endereçamento	automático	estático (jumpers)	dinâmico (negociado)	dinâmico (negociado)
Alimentação	no barramento	externa	no barramento	no barramento ou externa
Conexão de dispositivos	máquina desligada	máquina desligada	máquina ligada (*hot-plugable*)	máquina ligada (*hot-plugable*)
Negociação	*bus mastering* (pinos REQ, GNT)	similar a *daisy-chain*	canal virtual: *pipe* (negociado, *stream* ou *packages*)	similar a *daisy-chain* (também chamado SCSI serial)
Aplicações	Intertfaces para barramentos externos (USB, Firewire, SCSI), Placas de rede, Placas Gráficas	Fitas magnéticas, leitores de CD, Discos rígidos, Scanners, Zip	Teclados, Monitores, Mouse, Joystick, Caixas de som, Zip (mais lento)	Aparelhos MIDI, Transmissão de vídeo, (Cameras, televisão, video-cassete), Discos Rígidos

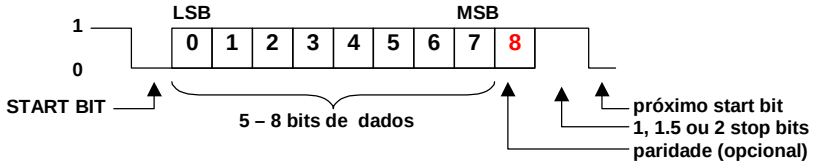
Observações:
1) Atenção para as diferenças entre o padrão e o que é na prática implementado nas máquinas pelos fabricantes
2) Padrões evoluem, dados não são definitivos

6 INTERFACE SERIAL

- Envio bit a bit
- Conectores para a porta serial do PC : 9 ou 25 pinos

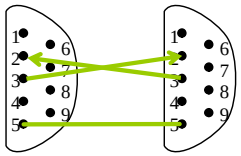
- Aplicações:
  - Modem, computador a computador, mouse, impressora serial
  - Sensores / atuadores
- Simplicidade: dois fios para uni-direcional ou três fios para bi-direcional
- Desvantagem: velocidade (em comparação com a paralela)
- Protocolo padrão: RS232 (um lógico +12V, zero lógico –12V)

  (Linha telefônica: analógico → modem significa modulador/demodulador)
- Transmissão Síncrona
  - Transmissor/receptor compartilham sinal de sincronismo comum
  - Exige mais um fio no cabo
- Transmissão Assíncrona
  - Não há sinal de sincronismo
  - Dado a ser transmitido deve ser “encapsulado”, com informação de controle
  - Transmissor e receptor devem ter mesma taxa de transmissão
- Protocolo RS-232



  - Todos os parâmetros devem ser iguais entre os dois lados
  - Taxa de transmissão ou baud rate: tempo para enviar um bit
    (1200, 2400, 4800, 9600 – PC, … 144.000) bps
    Se, p.e., a taxa é de 9600, significa que o receptor lê a linha a cada 1/9600s.
  - Start bit: para sincronismo
  - Stop bit: após o envio de uma palavra, a linha vai para 1, sinalizando final da palavra. Pode-se programar 1, 1.5 ou 2 stop bits
  - Paridade: opcional, para detecção e correção de erros (pode ser par ou ímpar)

- Conexão mínima:



  - Pino 5: ground, pino 3: transmissão, pino 2: recepção
  - Outros: protocolo com modem. Ex: pino 9 ring indicator
- Comparação entre diferentes padrões de interface serial

Standard	Data rate (current)	Medium	Devices per port
RS-232C	144000 bps	Twisted pair	1
IrDA	4 Mbps	Optical	126
USB	12 Mbps	Special 4-wire cable	127
IEEE 1394	100 Mbps	Special 6-wire cable	64

6.1 Porta Serial do PC

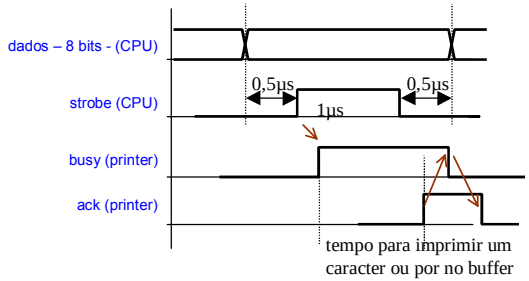
- Controlador: UART (universal asynchronous receiver transmiter), 8250 ou 16550 (digitar no PC o comando MSD – diagnóstico, que indica os controladores)
- Acesso ao registradores:
  - COM1: 3F8 – 3FF
  - COM2: 2F8 – 2FF
  - COM3: 3E8 – 3EF
  - COM4: 2E8 – 2EF

7 INTERFACE PARALELA

- Transfere byte a byte, controlador 8255
- PC:
  - Porta Paralela Padrão
  - PPP (*PS/2 Parallel Port*)
  - EPP (*Enhanced Parallel Port*)
  - ECP (Enhanced Capabilities Port)
- Registradores:
  - LPT1: 03BCH, LPT2: 0378H, LPT3: 0278H

7.1 Porta Paralela Padrão ou Standard

- Unidirecional – apenas o PC envia dados (controle é obviamente bidirecional)
- Do lado PC 25 fios, e do lado impressora 36 pinos
- Protocolo: Centronics (cabo de até 3m, devido a problemas de *crosstalk*)



  - 1 byte a cada 2 µs. Taxa? 500 KB/s
  - Busy: sinaliza dispositivo ocupado ou buffer cheio. No caso de impressão a impressora pode estar imprimido um caractere e a CPU deve esperar (atua como um "freio")
  - Overhead de CPU, barramento, protocolos, atendimento à interrupções, faz a taxa cair a 100KB/s.
  - Ainda suficiente para qualquer impressora

Registradores:

- 3 registradores de E/S:
  - Porta: registrador de dados
  - Porta+1: status (vem da impressora): (7) busy, (6) ack, (5) paper out, (4) select – ativa ou destiva impressora, (3) error, (2-0), não utilizados
  - Porta+2: controle (vai para impressora): (3) select, (2) initialize, (1) auto feed, (0) strobe

7.2 PPP (*PS/2 Parallel Port*)

- Bidirecional – HDs, SCSI, rede, comunicação, …
- Compatível com unidirecional
- Possibilidade de realizar a transferência por DMA ➔ rapidez

7.3 EPP (*Enhanced Parallel Port*)

- 800 Kbytes/s
- Compatível com os anteriores, bidirecional (cabo especial, sempre o mesmo conector).
- Utiliza os bits 7 e 6 do registro de controle. C7 [1 EPP, 0 standard], C6 [1 BID, 0 UNI]
- Permite conectar periféricos em daisy-chain (exemplo: zip drive)
- Porta+3 : endereço do periférico

- Porta+[4-7]: buffer de 32 bits, CPU escreve diretamente 32 bits

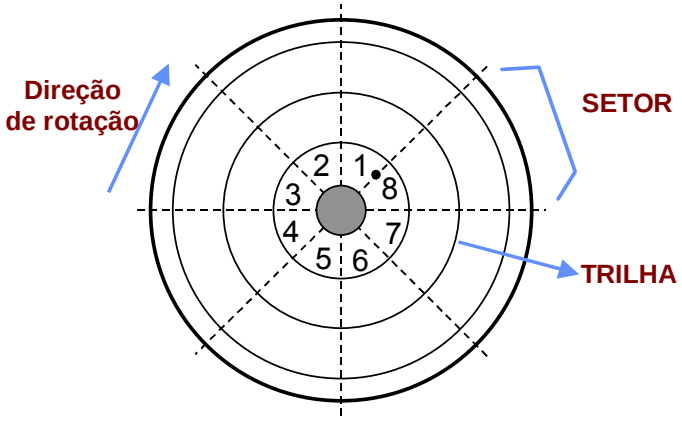
7.4 ECP (*Enhanced Capabilities Port*)

- 1 Mbyte/s, compressão de dados
- Criada em 1992 para a arquitetura do 386 (HP e Microsoft)

8 DISCOS

- Principais periféricos para armazenamento de dados
- Dispositivos de entrada/saída
- Outros meios utilizados
  - Fitas magnéticas: principalmente em workstations e máquinas de grande porte, baixo custo, utilizadas para backup ou para instalação de novos programas
  - Discos óticos, baixo custo, grande capacidade de armazenamento, grande imunidade a ruído
- Modo de armazenamento: magnético
- Organização física
  - Discos concêntricos : trilhas
  - Subdivisões das trilhas: setores
  - Podem haver várias superfícies , várias cabeças de leitura montadas sobre o mesmo braço movem-se juntas
  - Conjunto de trilhas na mesma posição: cilindro
- Formatos de disquetes (PC)
  - No PC um setor é sempre 512 bytes (0.5 Kb)

    capacidade do disco (kb) = (cilindros*setores*cabeças) * 0.5
  - Disquetes de 2.88 à partir do DOS 5.0



Disquete	Capacidade	Cilindros	Setores por trilha	Cabeças
5 ¼"	180 Kb	40	9	1
	360 Kb	40	9	2
	1.2 Mb	80	15	2
3 ½"	720 Kb	80	9	2
	1.44 Mb	80	18	2
	2.88 Mb	80	36	2

- Estrutura física:
  - Localização dentro do disco pela BIOS : cilindro, cabeça e setor
  - Localização dentro do disco pelo DOS (ocupa 32 bits):
    - setores lógicos começando em 0
    - setor lógico 0: cilindro 0, cabeça 0, setor 1 (boot)
- Conversão entre setor físico e setor lógico

  setor lógico = (setor físico-1) + cabeça * n_setores + cilindro * n_setores * n_cabeças
  - Setor físico = 1 + (setor lógico **mod** n_setores)
  - cabeça = (setor lógico **div** n_setores) **mod** n_cabeças
  - cilindro= setor lógico **div** (n_setores * n_cabeças)

    (div: divisão inteira, mod: resto de uma divisão inteira)
- Exemplo: Disco com 830 cilindros, 7 cabeças e 17 setores, dado gravado na posição 500,6,8

  capacidade = (830 * 7 * 17) / 2 = 49385 Kb = 50Mb

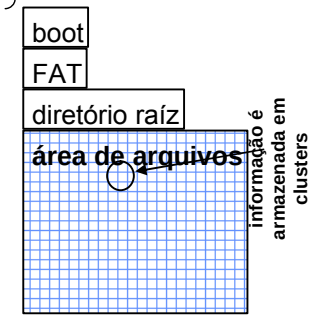
  setor lógico = 7 + 6*17 + 500*17*7 = 59609

  setor físico = 1+(59609 mod 17) = 1 + 7 = 8

  cabeça = (59609 div 17) mod 7 = 3506 mod 7 = 6

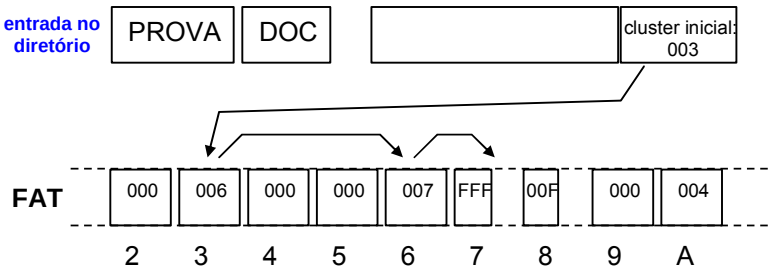
  cilindro = 59609 div (17*7) = 59609 div 119 = 500

  verificação
- Organização lógica: 4 áreas
  - Setor de boot (primeiro setor - 0 lógico ou 0,0,1 físico)
  - Tabela de alocação de arquivos (FAT) - duplicada
  - Diretório raíz - nome e informação sobre cada arquivo
  - Área de arquivos - onde ficam os arquivos
- Área alocada, (em setores):



Disquete	Capacidade	Boot	FAT	Diretório
5 ¼"	360 Kb	1	4	7
	1.2 Mb	1	14	14
3 ½"	720 Kb	1	6	7
	1.44 Mb	1	18	14
	2.88 Mb	1	18	15

- Setor de boot
  - Área reservada, todo disco contém esta área
  - Se há programa de boot, no final do setor há a "assinatura" 55H e AAH
  - Contém bloco de parâmetros para a BIOS, que descreve o disco (setores, cabeças, cilindros) ou a partição
- Diretório raiz
  - Identificação dos arquivos
  - Cada arquivo/subdiretório (entrada) ocupa 32 bytes
  - Exemplo: 14 setores (disco de 1.44 Mb), implica em 14*512 bytes (7168 bytes), considerando 32 bytes por entrada temos 224 (7168/32) possíveis entradas
  - Limite do número de arquivos é 16*14 = 224
  - Entrada contem principalmente: nome (8), extensão (3), atributo (1), hora (2), data (2), tamanho (4), cluster inicial (2)
  - Nome que inicia por 5EH é arquivo apagado
  - Atributos: read only (0), hidden (1), sistema (2), volume (3), subdiretório (4)
- FAT (file alocation table)
  - Área de arquivos é dividida em **clusters**
    - cluster pequeno = arquivos muito fragmentados
    - cluster grande = espaço ocioso
    - disco de 1.44: 1 cluster = 1 setor (512 bytes)
  - Utilizada para encadear todos os clusters



  - Existe uma entrada na FAT para cada cluster (entrada = ponteiro)

  - **Bom tamanho de cluster: 8 KB (16 setores de 512k)**
  - Versões do DOS até a 3.3 utilizavam FATs de 12 bits, resultando em 4096 clusters, multiplicando por 8k, resultava em um disco de no máximo 32MB.
  - DOS 4.0 e superiores utilizam FAT de 32 bits, resultando em 64 Kclustes. Para cluster de 8KB, implica em disco de 512Mb. O tamanho máximo de cluster sendo 32KB implica em discos de 2 GB.
  - FAT32, utiliza na verdade 28 bits para clusters, resultando em 256 Mclusters. Considerando 8KB por cluster, podemos ter discos de 2TB !
  - Valores especiais: 0 = cluster livre, FF7H ou FFF7H = indica cluster ruim, maior que FFF7H indica fim de arquivo
  - Clusters 0 e 1 não existem
  - Região muito sensível: qualquer alteração significa perda de dados ou "mistura" de arquivos"

Discos rígidos (winchester)

- Mesma organização que disquetes. Exemplos:

Cilindros	Cabeças	Setores	Capacidade
615	4	17	20
1024	8	17	67
1024	4	60	124
1599	9	35	258
1632	15	54	664
1962	19	72	1374

- Maior capacidade (6.2 Gb)
- Limitação: número de clusters é limitado a 2^{16} (65k)

  ➔ **FAT 32 resolve este problema**
- Assim *(1 setor = 512 bytes)*:
  - Cluster de 2k - capacidade máxima de 134 M
  - Cluster de 8k - capacidade máxima de 536 M
  - Cluster de 16k - capacidade máxima de 1 Gb

  (quanto maior o cluster maior o espeço perdido no disco)
- Solução: particionar o disco
- Partição
  - Cada partição é considerada um novo disco (com setor de boot próprio)
  - DOS permite até quatro partições diferentes (podendo-se ter até 4 sistemas operacionais diferentes)
  - Tabela de partição: primeiro setor físico do disco
- Tempo de acesso: inferior a 20 ms
- Comparação entre disco rígido magnético (HD) e discos óticos (CD-R, CD-RW)

	HD	CD
Rotação	Constante	Variável
Organização dos dados	Trilhas concêntricas	Espiral “boa noite”
Número de setores	Igual em todas as trilhas	Maior quanto mais longe do centro
Otimizado para	Velocidade	Capacidade

9 EXERCÍCIOS

1. Explique a diferença entre entrada/saída (**E/S**) mapeada em memória e entrada e saída mapeada em portas.
2. Explique as principais diferenças entre E/S programada e E/S por interrupção.
3. Qual o método utilizado para otimizar E/S tipo *polling*? Explique-o.
4. Mostre o diagrama de um sistema de E/S por interrupção tipo *Daisy-Chain*. Explique-o
5. Como funciona o mecanismo de mascaramento de interrupções? O que significa uma interrupção não mascarável, dando um exemplo de aplicação.
6. Como é realizado o mecanismo de tratamento de interrupção em um microprocessador? Cite as principais etapas do processo, comparando com uma chamada de subrotina.
7. Mostre como é feito o tratamento das interrupções de E/S nos computadores com microprocessadores 80x86.
8. Em que casos a utilização do método de E/S tipo DMA é recomendada?
9. Supor uma CPU operando a 40 MHz, com HD transferindo dados a 1MB/s continuamente, utilizando DMA. Se a duração da programação do controlador de DMA pela CPU necessita 500 ciclos de clock, a duração da rotina de interrupção para tratar término do DMA é de 500 ciclos de relógio, e cada operação de transferência do disco manipula 4KB, qual a fração de tempo da CPU é consumida na operação?
10. Qual a função básica dos barramentos? Qual a vantagem de normatizá-los?
11. O que é “arbitragem” de barramento? Quais os principais métodos de arbitragem, explicando-os?
12. Cite três características que afetam o desempenho de um barramento.
13. Mostre o padrão de bits utilizado na transmissão serial RS-232.
14. Por que a organização dos setores não é exatamente igual nos discos óticos e nos magnéticos?