

revisão – Redes e Barramentos

● Redes

- * protocolos – nomes e endereçamento, proteção, confiabilidade
- * **latência total**
 - = **overhead_transmissão** proc 100% ocupado com envio
 - + **tempo_de_vôo** veloc da luz + atrasos hw
 - + $|mensagem| / largura_de_banda$ [bits / bits/s]=[s]
 - + **overhead_recepção** proc 100% ocupado recebendo
- * topologias
 - estáticas: ponto-a-ponto, estrela, anel, malha
 - dinâmicas: crossbar, barramento, redes Delta & família

● Barramentos

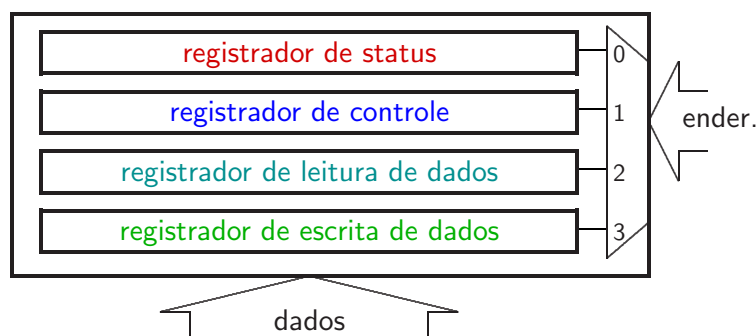
características de vazão e latência

- * síncronos **cmd** ; *proc* ; **rsp**
- * assíncronos **cmd** ; **act** ; *proc* ; **rsp** ; **act**

Entrada e Saída

- Tipos e Características de Dispositivos
- Arquitetura do Sistema de E/S
- Discos
- Redes, barramentos, vazão e latência
- Dispositivos, Interfaces com CPU e com Sist Operacional
 - * Modelo de dispositivo
 - * Espaços de endereçamento e hierarquia de barramentos
 - * Modos de acesso – por programa, interrupção, ADM
 - * E/S e hierarquia de memória
- Desempenho e projeto

Organização de um periférico



- status** estado do periférico: dados prontos, erro, interr pendente
- controle** modo de operação, comandos, formatação dos dados
- entrada** leitura pelo processador
- saída** escrito pelo processador

Organização de um periférico (ii)

```
typedef struct Perif {    // modelo de periférico
    int    status ;
    int    controle ;
    char    leit ;
    char    escr ;
} Perif;

Perif.controle = MODO_DE_OPERACAO ;

Perif.escr = c ;          // envia um caracter

// recebe bloco de caracteres do periférico
for (i=0; i < TAM; i++) {
    while (Perif.status != PRONTO) { }; // busy wait
    recebidos[i] = Perif.leit;
}
```

Espaços de Endereçamento (i)

Periféricos mapeados em espaço disjunto (Intel x86)

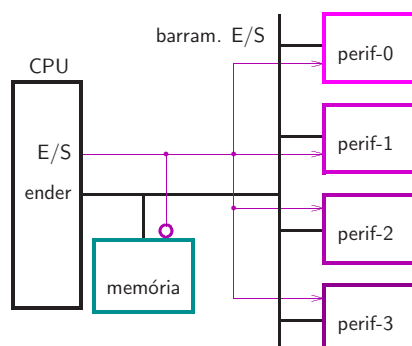
signal do barramento indica se referência é E/S ou memória
 necessita instruções especiais para efetuar operações de E/S
 operações de E/S com instruções específicas `in` e `out`

Periféricos mapeados como memória

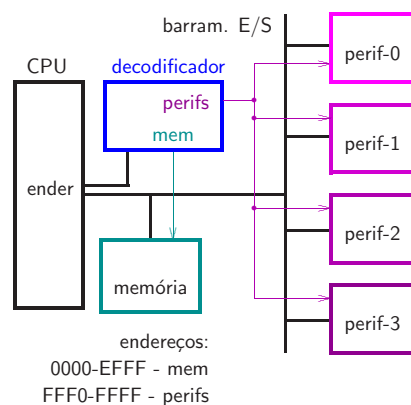
faixa do espaço de endereçamento reservada para E/S
 endereço diferencia E/S de memória
 operações de E/S com instruções 'comuns' `lw` e `sw`

Espaços de Endereçamento (ii)

E/S em espaço de endereçam disjunto



Periféricos mapeados como memória



Classes de Periféricos

- Lentos e Preguiçosos:
 - * teclado – 10 caracteres por segundo
 - * mouse – 30 caracteres por segundo
- Rápidos e Gulosos:
 - * disco rígido – 512 bytes em 0.1ms (≈ 4 Mbytes/s)
 - * interface de rede rápida – 1 Mbytes em 0.1ms (≈ 100 Mbytes/s)
 - * controlador de vídeo – 30 Kbytes em 1ms (≈ 30 Mbytes/s)

Tratamento diferente para as duas classes:

periféricos lentos podem esperar;

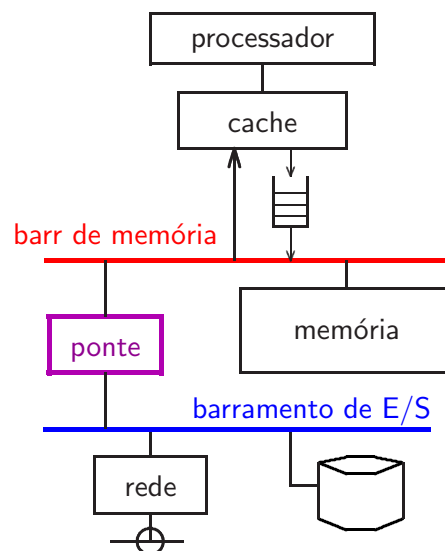
periféricos rápidos devem ser prontamente atendidos;

tratamento de grandes volumes é mais complexo

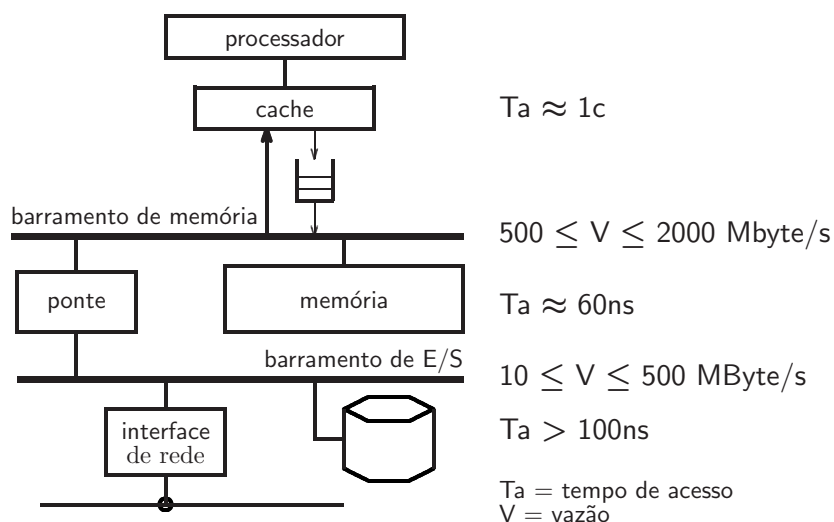
que o de caracteres individuais

+ detalhes em SO

Hierarquia de Barramentos (i)



Hierarquia de Barramentos (ii)

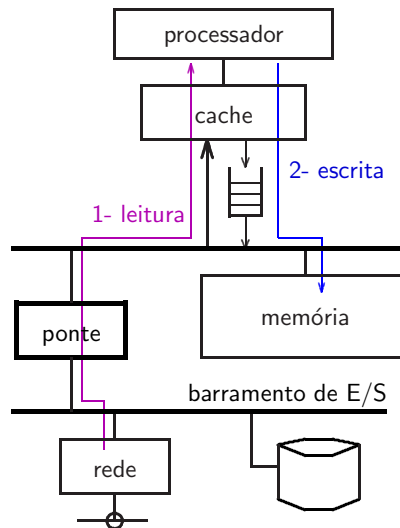


Entrada e Saída Programada (i)

Transferências entre periférico e memória executadas por programa, palavra a palavra
→ dispositivos lentos
e/ou poucos dados

```
for (i=0; i < TAM; i++)
    M[i] = Perif.dados;

...
1  ld r1, DADOS(rP)
2  st r1, 0(rM)
...
```



Entrada e Saída Programada (ii)

Eficiente para mensagens pequenas

espera ocupada → processador não faz trabalho útil,
só efetua cópias entre periféricos e memória

```
for (i=0; i < TAM; i++) {
    while (Perif.status != PRONTO)
        { }; // busy wait
    M[i] = Perif.dados;
}
```

Eficiência de E/S Programada

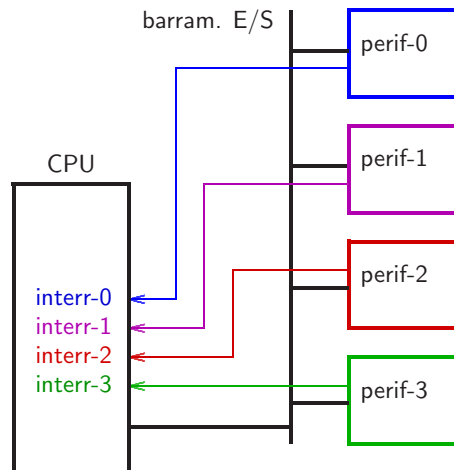
- Periférico endereçado como E/S
 - * recepção: periférico → memória
 - leituras na velocidade do barramento de E/S
 - escritas na velocidade da memória (fila de escrita cheia)
 - * transmissão: memória → periférico
 - leituras na velocidade da memória (faltas na cache)
 - escritas na velocidade do barramento de E/S
- Periférico endereçado como memória
 - * acessos desviam cache: referências na velocidade da memória
 - * acessos através da cache: após ler registradores do periférico, invalidar bloco na cache (consistência com memória)
- Frequentemente acessos de E/S desviam a cache por problemas de consistência entre o conteúdo dos registradores de status e as cópias na cache que são caducas

E/S por Interrupção (i)

CPU programa registrador de controle com operação a ser efetuada pelo periférico;

quando operação completa, periférico interrompe CPU

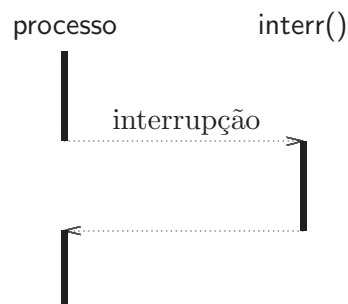
Enquanto periférico não está pronto processador fica livre para efetuar outras tarefas



E/S por Interrupção (ii)

Quando detecta interrupção, processador salta para função que trata do evento sinalizado pela interrupção

Após tratar evento, processador busca instrução que teria sido executada, não fosse a interrupção



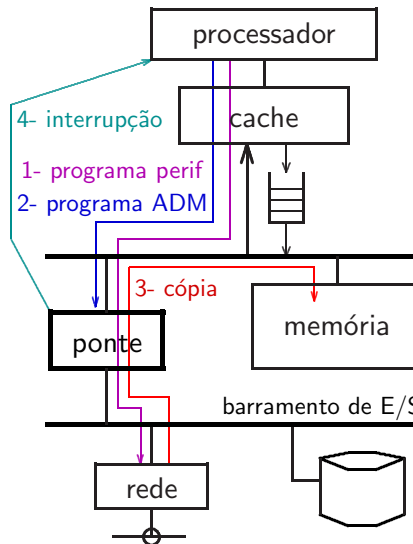
E/S por Interrupção (iii)

- Operações numa função de tratamento de interrupções:
 - * desabilita novas interrupções de prioridade < esta
 - * salva registradores
 - * habilita interrupções de prioridade ≤ esta
 - * efetua tratamento do evento causador
 - * recompõe registradores
 - * habilita demais interrupções
 - * retorna
- código do tratador deve ser **reentrante**
tratador de interrupção pode ser interrompido

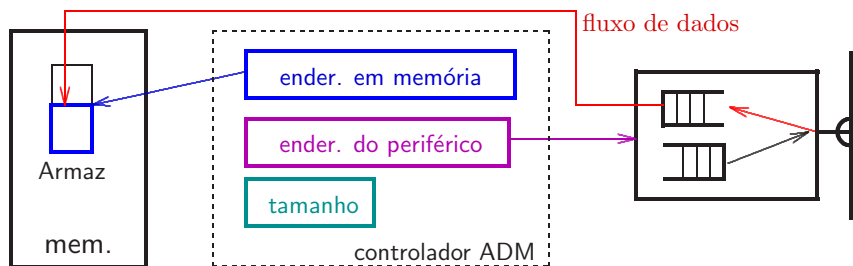
Acesso Direto à Memória (i)

CPU programa periférico e controlador de ADM;
c-ADM efetua transferências entre periférico e memória; quando transferência completa, c-ADM interrompe CPU

ADM é bom para grandes volumes de dados



Acesso Direto à Memória (ii)



inicialização pela CPU:

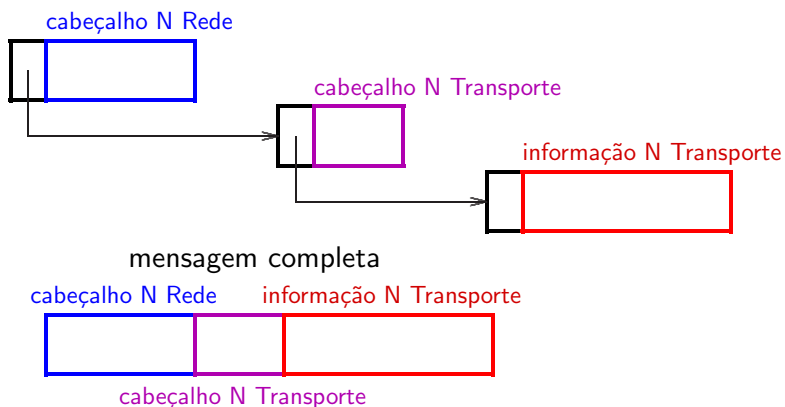
```
ADM → mem = &(Armaz[0]);
ADM → perif = (Interf[rxREDE]);
ADM → tam = Interf.tam;
```

transfer pelo c-ADM:

```
for ( ; ADM → tam > 0; ADM → tam-- )
    (ADM → mem)++ = ADM → perif;

/* interrompe processador */
```

Acesso Direto à Memória (iii)



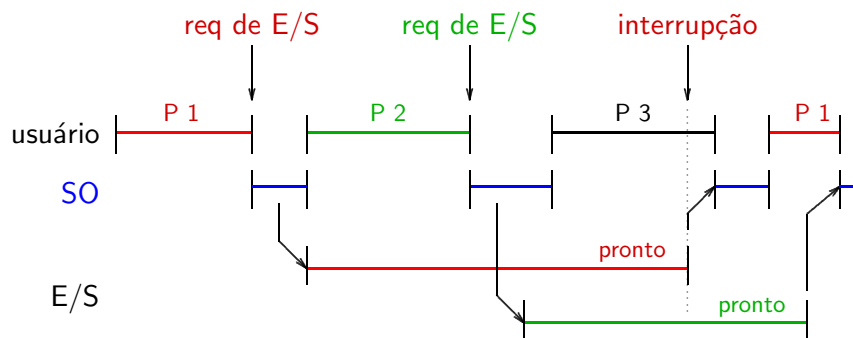
Eficiente para mensagens grandes

Se controlador de ADM suporta dispersão e coleta, porções de msgns distribuídas por vários armazenadores

canais de E/S
scatter-gather

revisão: E/S e Computação

E/S concorre com computação de maneiras complexas

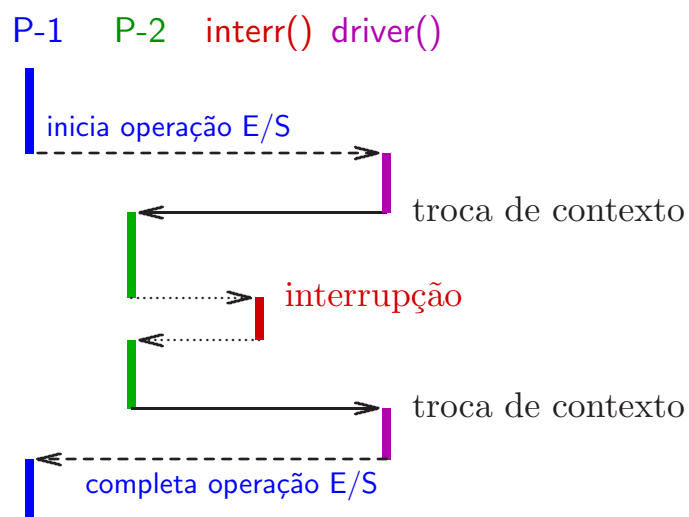


$$\mathcal{T}_{\text{tarefa}} = \mathcal{T}_{\text{cpu}} + \mathcal{T}_{\text{E/S}} - \mathcal{T}_{\text{concorr}}$$

Desempenho de Caches e E/S

- Operações de E/S são:
 - * movimentação entre periféricos e memória
 - * modificação e/ou inspeção simples (checksum)
 - * efetuadas em modo supervisor (pelo Sistema Operacional)
- Operações de E/S tem pouca localidade:
 - * cópias e movimentação de dados
 - * escalonamento de processos (interrupções, threads)
 - * envolvimento do SO em cada operação de E/S

E/S e Caches (ii)



E/S e Caches (iii)

Caches tem pouca eficácia para E/S por causa da baixa localidade e interferência do SO:

- interrupções causam poluição (expurgam blocos úteis)
- escalonamento do processador causa poluição
- cópias enchem fila de escrita e bloqueiam processador
- troca de contexto invalida conteúdo da cache
- operações nas filas de dispositivos causam re-escalonamento

resumo – Sistemas de E/S

- $\mathcal{T}_{\text{tarefa}} = \mathcal{T}_{\text{cpu}} + \mathcal{T}_{\text{ES}} - \mathcal{T}_{\text{concorr}}$
- Hierarquia de barramentos – desempenho
desemp(CPU-cache) \gg desemp(cache-mem) \gg desemp(E/S)
- Espaço de endereçamento: como memória *vs* como E/S
- Tipos de barramento:
 - * síncrono *vs* assíncrono *vs* split transaction
- Processamento de E/S
 - * por programa
 - * por interrupção
 - * acesso direto a memória
 - * canais de E/S (c-ADM programáveis)