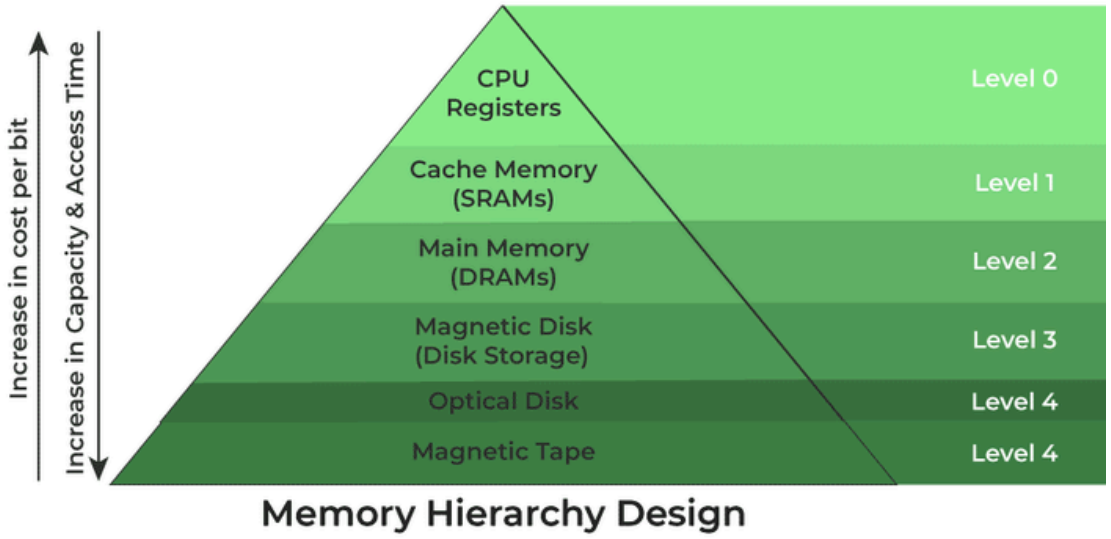
# Memória

Características conflitantes:

- Capacidade
- Acesso rápido
- Custo baixo

## Hierarquia de memória




A sarita fez um triângulo com 3 andares:
Andar 0:

- Na placa mãe
    - Registrador
    - Cache
    - Mem drive (memória ram)

Andar 1:

- Fora da placa mãe
    - HD
    - ssd
    - flash

Andar 2:

- Armazenamento externo
    - fita magnética

Do topo para a base da pirâmide:

- Velocidade diminui
- Capacidade de armazenamento aumenta
- custo por bit diminui

Os três níveis da pirâmide que a Sarita fez:
Nível 0

- Cache
    - Memória interna a CPU

Nível 1

- Memória primária (ram)

Nível 2

- Memória secundária (hd ssd)

## Características de um sistema de memória

Capacidade

- Tamanho da palavra
- Número de palavras
    - Definição de palavra:
        - Número de bits para representar um inteiro ou a instrução
- Unidade de endereçamento
    - O endereçamento pode ser a Byte ou a palavra
    - O RISC-V era endereçado a byte

Unidade de transferência

- Memória principal (ram)
    - número de bits lidos ou escrito de uma vez na memória
    - Usualmente é a palavra
- Memória secundária (hd)
    - Unidades chamadas de blocos (conjunto de palavras)

Formas de acesso

- Sequencial
    - Os dados são organizados em registros
    - O tempo de acesso varia depende do dado que você quer pegar (é sequencial)
    - Ex: fita magnética

- Direto
    - Dados organizados em blocos
    - Tempo de acesso varia
        - Por causa do local da cabeça de leitura
    - Cada bloco possui um endereço
    - Ex: disco
- Aleatório
    - Cada posição possui um endereço único
    - Tempo de acesso constante
        - constante porque é elétrico
    - É chamado de "acesso aleatório" porque qualquer posição de memória pode ser acessada de forma direta e imediata, sem a necessidade de seguir uma ordem específica, proporcionando tempos de acesso uniformemente rápidos e previsíveis.

    - Ex: memória principal
- Associativa
    - Compara bits com conteúdo das posições da memória
    - Busca por conteúdo e não por endereço
    - Em vez de fornecer um endereço para acessar os dados, você fornece um padrão de dados que está procurando. A memória associativa compara simultaneamente o padrão fornecido com o conteúdo armazenado em todas as suas posições de memória.
    - Ex: cache

Desempenho:
- Tempo de acesso (latência)
    - Memórias de acesso aleatório
        - Tempo gasto para fazer a leitura e/ou escrita
    - Memória de acesso não aleatório
        - Tempo para posicionar o mecanismo de leitura/escrita
- Tempo de ciclo de memória
    - Em memórias de acesso aleatório, é o tempo necessário para que o sistema esteja pronto para se realizar um próximo acesso
- Taxa de transferência
    - Auto explicativo

ROM
Só lê, já vem de fábrica.

Aplicações:
- Unidade de controle
- Bios
- Implementação de certos circuitos

Acesso rápido

ROM programáveis
- PROM
    - É igual a rom, mas pode ser programado uma única vez

| Tipo | Categoria | Apagar | Escrita | Volatilidade |
|---|---|---|---|---|
| ROM | Somente leitura | Não | Mascar. | Não |
| PROM | | | Elétrica | |
| EPROM | Mais R Escrita difícil | U.V. | | |
| $E^2$PROM | | Elétr.(byte) | | |
| FLASH | | Elétr.(bloco) | | |
| RAM | RW | Elétr. (byte) | | Sim |

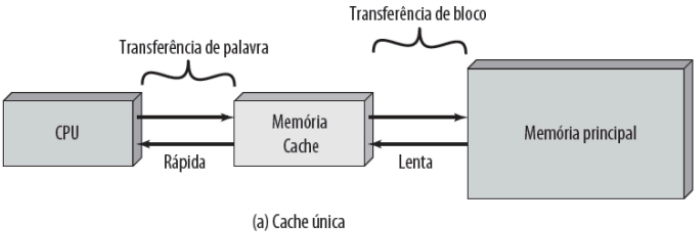
## Memória cache

Método associativo

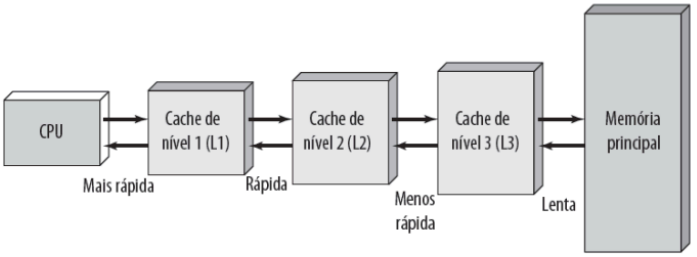
- Localiza os dados por conteúdo e não por endereço

Hierarquia:

- A memória cache é dividido em níveis
- Um nível da cache que está mais próximo do processador é normalmente um subconjunto de qualquer nível mais distante
- Todos os dados só ficam armazenados no nível mais baixo (ram)



(a) Cache única



(b) Organização de cache em três níveis

Princípio da Localidade

- Temporal
  - Se um dado na memória foi utilizado em um instante, existe a probabilidade dele ser utilizado novamente
- Espacial
  - Se um dado na memória foi utilizado existe a probabilidade das posições que estão perto serem utilizadas também

Bloco:

- Menor unidade que pode ser armazenada ou recuperada da cache.
  - Um bloco pode conter várias palavras da memória
  - A cache lida com um bloco inteiro por vez

Acerto (hit)

- Dados solicitados pelo processador estão na cache

Falha (miss):

- Os dados solicitados pelo processador não estão na cache

Tempo de acerto:

- Tempo para acessar um nível da hierarquia de memória incluindo o tempo necessário para determinar se é um acerto ou uma falha
- Tempo para ver se o dado está ou não na cache

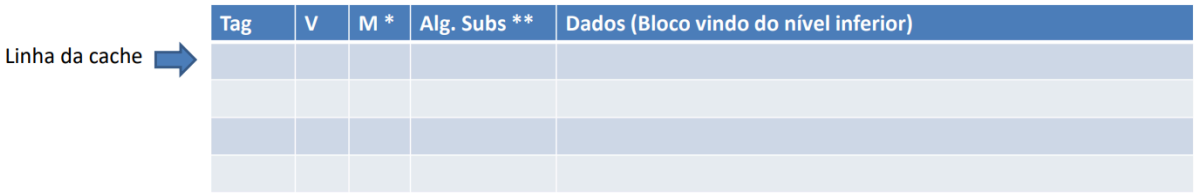
Penalidade por falha:

- Tempo necessário para levar e buscar um bloco de um nível inferior para um superior incluindo o tempo para acessar o bloco

Tamanho do bloco

- Ter um bloco grande é bom para o princípio da localidade espacial (até um certo ponto), porém é pior para o princípio da localidade temporal, logo tem que ter um equilíbrio no tamanho do bloco.
    - Se um bloco for muito grande pode ser ruim para a temporal, pois como cabe poucos blocos o bloco que você tinha colocado irá sumir, fazendo com que acabe com o princípio da  localidade temporal.
- Vai ter menos diversidade de memória
- Ele é ruim quando a probabilidade de utilizar alguém que já está no bloco é menor do que a probabilidade de ficar substituindo os blocos.

## Estrutura da cache




Tag: identificador único da linha
V: bit de validade (se está removido ou não)
M: bit de modificação

- só é necessário em write-back
- Indica se os dados que estão armazenados naquele bloco foram modificados ou não

Alg. sub: necessários para implementar os algoritmos de substituição

- necessário se o mapeamento for associativo

## Função de mapeamento

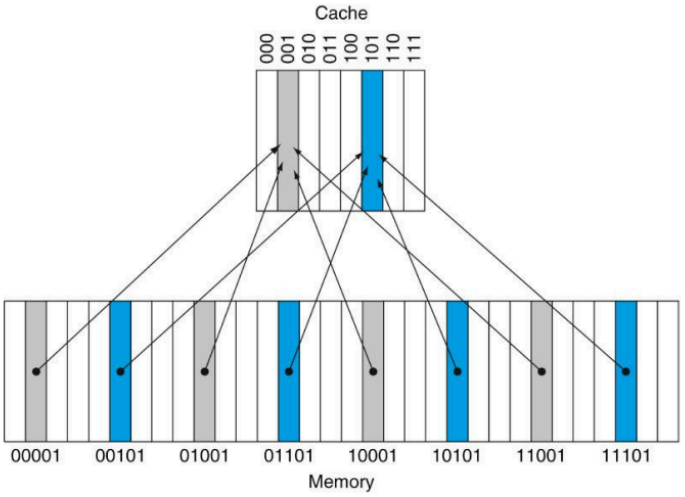
Mapeamento direto

- cada bloco na memória principal é mapeado em uma única linha da cache
- Equação: i = j módulo m

***i* = número da linha na memória cache**
***j* = número do bloco da memória principal**
***m* = número de linhas na memória cache**

- Hash

Exemplo:




Vários blocos da memória principal podem ser mapeados na mesma linha da memória cache

O tag de um linha armazena a parte mais significativa do endereço do bloco da memória principal. Ou seja, para diferentes blocos vai ter diferentes tags. Portanto, dado um bloco você sempre vai saber em qual linha ele deveria estar, logo, é só ir lá e verificar se ele está lá.

Quando a CPU quer acessar um endereço de memória, ela calcula a linha da cache usando o módulo (ou outra função de mapeamento). Em seguida, compara a tag armazenada nessa linha com a tag do endereço de memória desejado.

ByteOffset:

- Quando a arquitetura é endereçada a byte e um bloco tem mais de um byte o Byte Offset serve para identificar qual o byte dentro do bloco
- Por exemplo, em uma cache onde cada linha armazena 16 bytes, o byte offset precisaria de 4 bits para endereçar cada byte dentro da linha (pois 2^4 = 16).

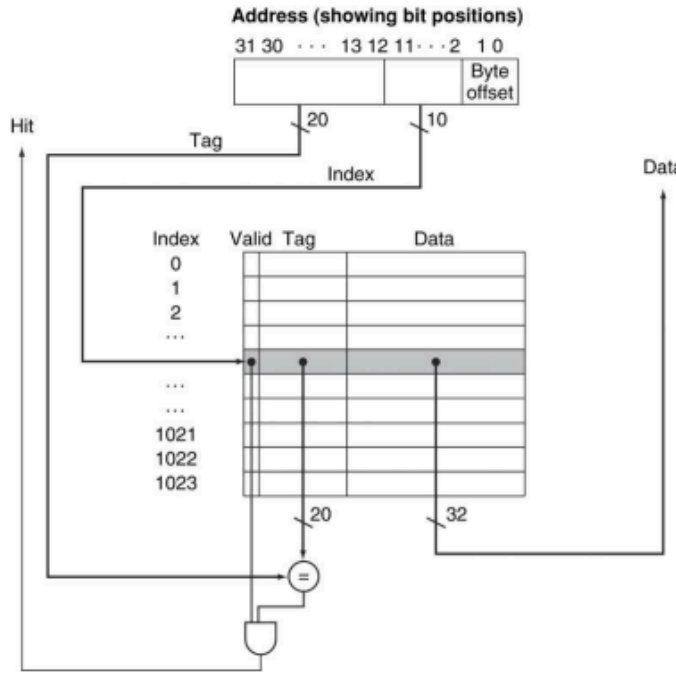
Word Offset:

- Quando a arquitetura é endereçada a palavra ou um bloco tem mais de uma palavra, logo o Word Offset serve para identificar a palavra

Index é o I (resultado do mod)

Tag:

- Serve para identificar unicamente o bloco




Cache com:
- 1024 Blocos
- 1 Palavra por Bloco
- Palavras de 4 Bytes

Fonte: Patterson & Henessy – 5ª edição

9

Vantagens:

- técnica simples
- baixo custo de implementação

Desvantagens:

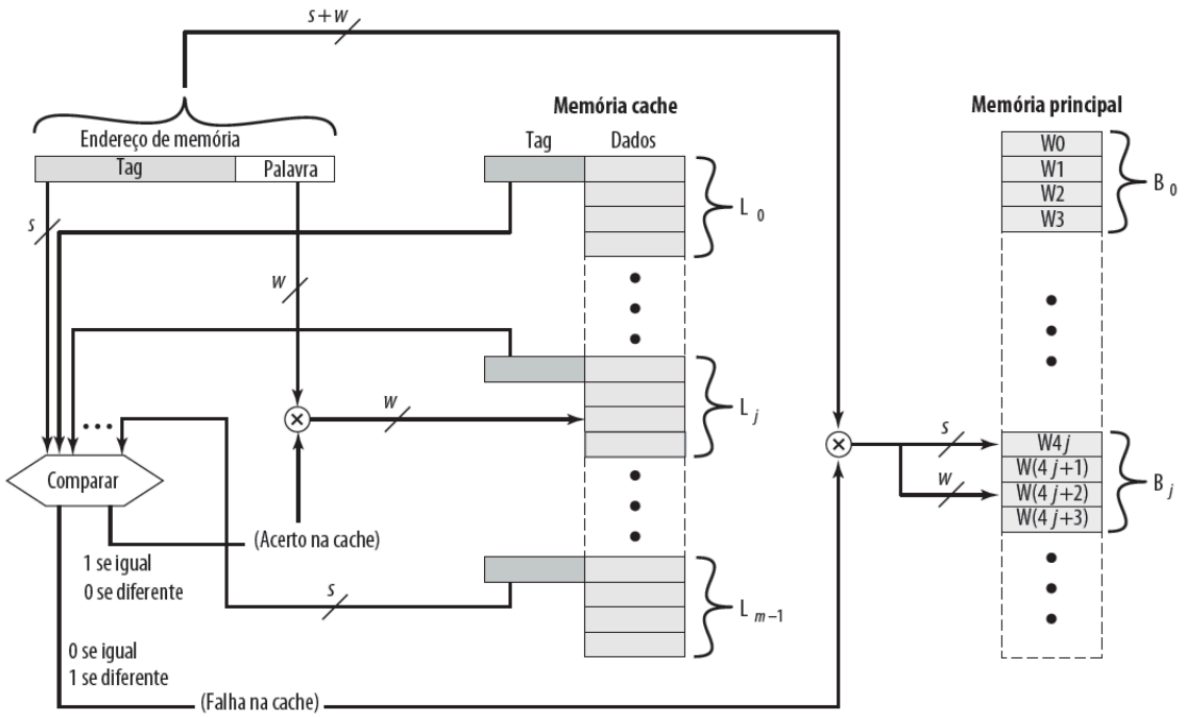
- tem muito conflito, algo que pode diminuir a taxa de acertos

Mapeamento associativo

- Evita desvantagens do mapeamento direto
- Um bloco da memória principal pode ir para qualquer linha
- Tem Byte Offset, Word Offset e tag

Para saber se está na cache:

- Compara simultaneamente o campo de rótulo do endereço do bloco acessado com o rótulo de todos os blocos da cache.
    - Ele faz isso através de um XOR




A escolha de onde um bloco vai ir pra cache é feita através de políticas de substituição.

Vantagem:

- Maior flexibilidade para a escolha da linha a ser substituída quando um novo bloco é trazido

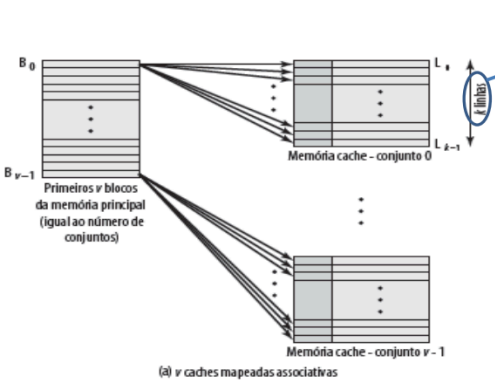
Desvantagem:

- Circuito complexo para fazer a comparação simultânea.

Mapeamento associativo por conjunto

- Combina o melhor dos dois mundos.
- A memória cache é dividida em v conjunto de k linhas(bloco) cada
- Agora além de Byte Offset, Word Offset e a tag também tem set (conjunto) que serve para identificar cada conjunto.

# Memória cache dividida em *v* conjuntos de *k* linhas cada:




**Linha: nomenclatura do Stallings**
**Bloco: nomenclatura do Patterson**

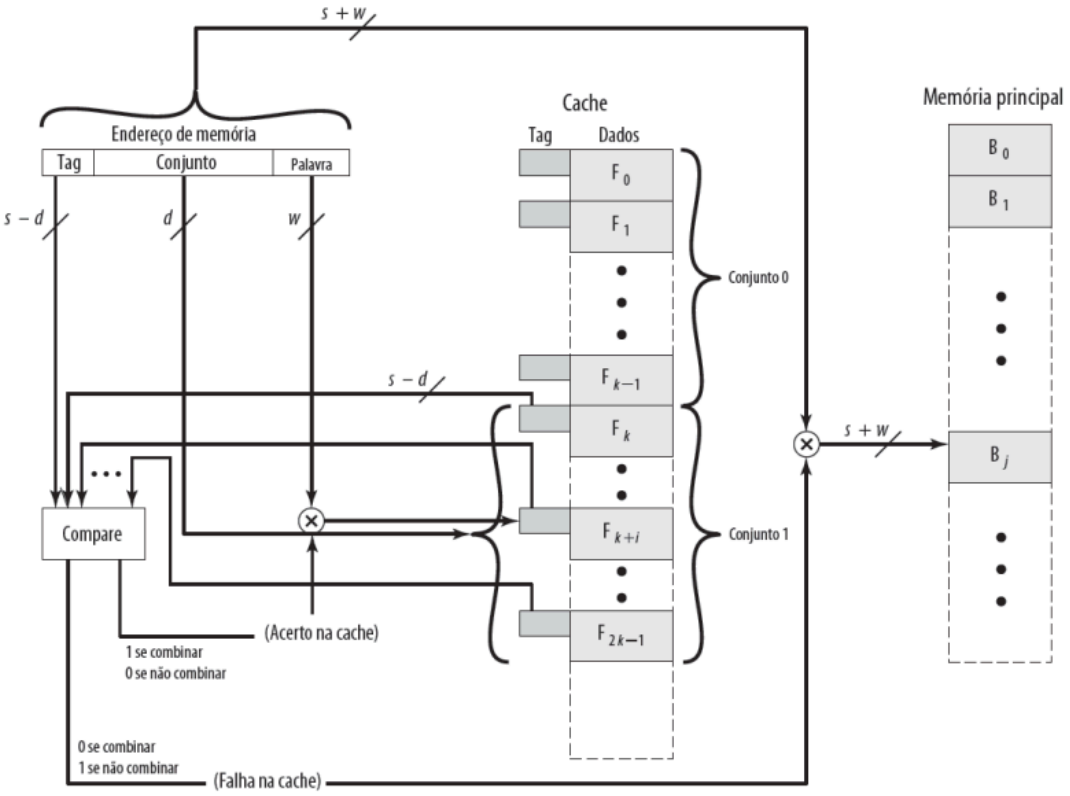
***m* = *v* x *k***
***i* = *j* módulo *v***

**mapeamento associativo por conjuntos de *k* blocos**
**Conhecido como *k-way set***

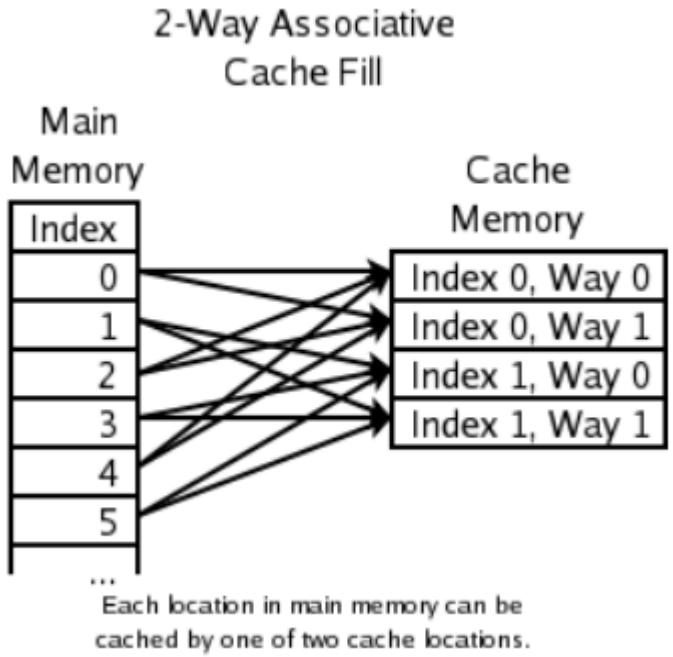
***i* = número da linha da memória cache**
***j* = número do bloco da memória principal**
***m* = número de linhas na memória cache**

Basicamente a memória cache é dividida em V conjuntos cada um com K blocos. É utilizado o método direto para decidir a qual conjunto um bloco vai, e dentro do bloco é utilizado o método associativo.

Diminui muito a colisão e também a + para fazer as comparações.



O rótulo é bem menor.




Isso daqui é quando em cada conjunto de 2 blocos.

## Algoritmo de substituição

- FIFO
  - Primeiro a entrar, primeiro a sair
- LRU
  - menos recente utilizado
- LFU
  - menos frequentemente utilizado
- Aleatório
  - Com certeza é o melhor (não, contém ironia)

LRU

- a linha a ser substituída é aquela que está há mais tempo sem ser utilizada
- Implementação
  - Contador associado a cada linha
  - Acerto: contador vira 0 e os demais contadores são incrementados
- Problema: o número de linhas torna-se grande
  - Porque vai ter que fazer um sort em muitos caras (se for bubble fudeu tlg)

FIFO - First In, First Out

- Remove a linha que está há mais tempo na cache (fila)

LFU - Least Frequently Used

- Remove a linha que foi utilizado menos vezes em um determinado tempo
- Frequência = utilização / tempo
- Usa contador

Aleatório:

- Tem um contador que é incrementado a cada ciclo, quando a substituição é necessária escolhe-se a linha cujo endereço é igual ao valor atual do contador.

## Política de escrita

- Escrita na cache
    - Se algo que você vai fazer uma escrita está na cache, você inicialmente escreve na cache e ai vai ter que garantir que essa alteração ocorra na memória principal.
- Os valores devem ficar iguais por motivos de:
    - Acesso de entrada e saída
        - Operações feitas diretamente na memória principal, logo tem que garantir que esteja igual na cache
    - Acesso por múltiplos processadores
        - Vários processadores de caches diferentes podem acessar a mesma memória, logo os valores têm que estar coerentes.
- Write-Hit
    - Quando os dados a serem escritos estão na cache
    - Write-through
        - Os dados são escritos na cache e imediatamente na memória principal
        - Causa gargalo por sempre ter que escrever na memória
    - Write-back
        - Os dados são escritos inicialmente somente na cache e só depois na memória principal. É mais eficiente, porém tem que ter um mecanismo para saber quando atualizar a memória principal
        - Ele só escreve na memória quando vai ser substituído
            - Estratégia simples:
                - Toda vez que o bloco é retirado da cache, seu valor é escrito na memória principal, estando alterado ou não
            - Estratégia complexa:
                - Quando o bloco for retirado da cache, seu valor só é escrito na memória principal se o bit de MODIFICADO estiver como 1.
        - Problema: acesso à memória principal pelos módulos de E/S deve ser feito através da memória cache
            - Isso faz com que o circuito fique mais complexo e de um potencial gargalo.
- Write-Miss:
    - Quando os dados a serem escritos não estão na cache
    - Write allocate

        - O bloco de dados é carregado na cache primeiro e a escrita é feita na cache
        - Depois disso é escolhido utilizar o Write-through ou Write-Back para atualizar a memória principal
    - No allocate
        - Os dados são escritos diretamente na memória principal

Dirty bitch é uma puta suja. Brincadeira, é quando o bloco está modificado e precisa fazer write back.

Origem das falhas (misses):

- Compulsória
    - É o miss causado pelo primeiro acesso a um bloco, pois como é o primeiro acesso ele com certeza não vai estar na cache
- Capacidade
    - É a falha causada quando a cache está cheia, ou seja, ela não tem capacidade de conter todos os blocos necessários durante a execução de um programa, fazendo com que os blocos sejam substituídos e depois recuperados.
- Conflito
    - Causado pela colisão no mapeamento direto ou no mapeamento associativo por conjunto.
    - A taxa de falha depende do tipo do mapeamento

Número de caches

- L1
    - Interna à pastilha do processador
- L2
    - externa (quando não tem l3)/interna (quando tem l3) à pastilha do processador
- L3
    - externa a à pastilha do processador. Sua existência fez a L2 ir para dentro do processador

Memória cache dentro do processador (on-chip)

- Reduz a atividade do processador no barramento externo
    - Diminui o tempo de execução e aumenta o desempenho
- Enquanto o processador faz acessos à cache interna o barramento fica livre para efetuar outras transferências

Memória cache fora do processador (off-chip)

- Dados que normalmente dão falha na cache on-chip frequentemente estão na cache off-chip
- Se não existir cache off-chip
    - Dado uma falha vai direto para a MP
    - Ai a velocidade vai depender do barramento e do acesso à MP causando um baixo desempenho
- Se existir cache off-chip

    - Economia de tempo de acesso depende das taxas de acerto nas caches on-chip
    - usar cache off-chip vai melhorar muito o desempenho

Cache inclusiva

- O dado que está na L1 também está na L2
- Vantagem:
    - Quando os dispositivos de E/S desejam acessar a cache, eles só precisam ver a L2
    - Mais simples de implementar

Cache exclusiva

- Se um dado está na cache é garantido estar em pelo menos uma das caches (L1 ou L2)
    - Não pode estar em ambas simultaneamente
- Quando da miss na L1 o dado é encontrado na L2 e é trocado com um bloco da L1
- Vantagem:
    - Maior capacidade

Memória cache unificada

- Uma única memória cache para armazenar dados e instruções
- Taxa de acerto maior que a separada
    - Balanceamento automático entre dados e instruções
- Utilizada nos níveis L2 e L3

Memória cache separada

- Tem uma memória cache para dados e uma memória cache para instruções
- A política de escrita só precisa ser aplicada à cache de dados
- Tem estratégias diferentes para cada cache
    - tamanho total, tamanho do bloco, organização
- É utilizada no nível L1

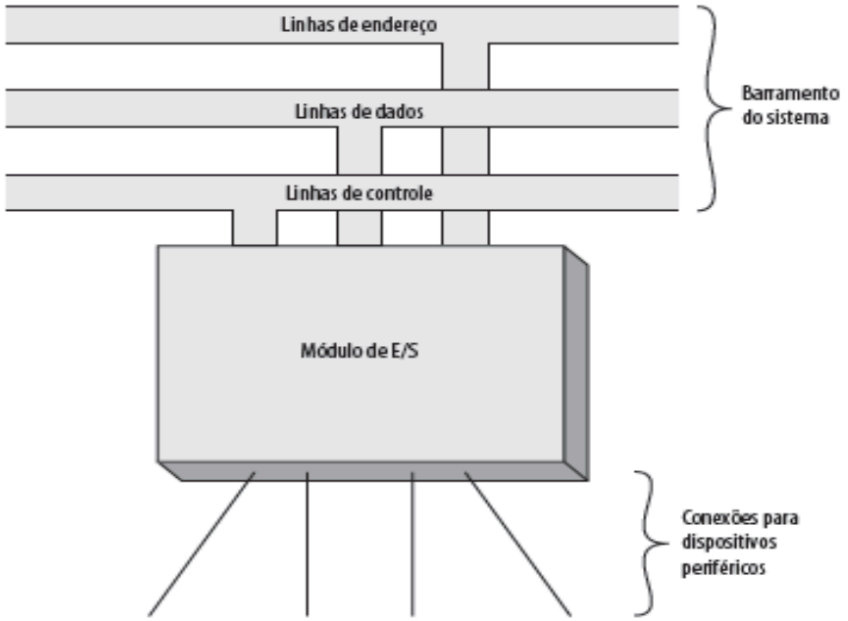
Desempenho das caches

| Design change | Effect on miss rate | Possible negative performance effect |
|---|---|---|
| Increases cache size | Decreases capacity misses | May increase access time |
| Increases associativity | Decreases miss rate due to conflict misses | May increase access time |
| Increases block size | Decreases miss rate for a wide range of block sizes due to spatial locality | Increases miss penalty. Very large block could increase miss rate |

# Entrada e saída

Existe a necessidade de um Módulo (ou controlador) de E/S, pois:

- A taxa de transferência é menor do que CPU ou RAM
- Formatos de dados e tamanhos de palavras diferentes




Módulo de E/S:

- Faz a comunicação dos dispositivos de E/S com o processador e a memória
- Livra a CPU do gerenciamento dos dispositivos periféricos de E/S

Funções do Módulo de E/S

- Controle e temporização
- Comunicação com o processador
- Comunicação com dispositivos
- Área de armazenamento temporário de dados
- Detecção de erros

5 Coisas, lembra disso e sai fácil.
CCCAD
Mnemônica: Cuidado: cães distraídos, agitados e desterros
“ : Carne, comer, comida, ai que DELÍCIA!

## Controle e temporização

Controla o fluxo de dados entre os recursos internos e os dispositivos externos.

Passos para o controle:

- Módulo de E/S retorna o estado do dispositivo para o processador
- Processador verifica estado do dispositivo

- Se estiver pronto para transmitir o processador envia um comando para o módulo de E/S
- O módulo de E/S obtém dados do dispositivo externo
- Os dados são transferidos do módulo de E/S para o processador.

## Comunicação com o processador

Decodificação do comando
- Módulo de E/S aceita e codifica comandos do processador
    - Esses comandos são sinais no barramento de controle

Dados:
- Os dados são trocados via barramento de dados

Informação de estado:
- Como existem periféricos lentos (tipo o teclado) é importante saber seu estado
- Ex: ready/busy
- Também contém sinais de erro
- Via barramento de controle

Reconhecimento de endereço:
- Cada dispositivo de E/S tem um endereço único
- O módulo de E/S deve ser capaz de reconhecer esses endereços
- Via linha de endereço

## Comunicação com os dispositivos externos
- Os dispositivos externos se comunicam com o módulo de E/S através de sinais de controle, dados e estado
    - Controle
        - Função a ser executada pelo dispositivo (input ou read, output ou write)
    - Estado:
        - ready/not ready
    - Dados
        - conjunto de bits a serem enviados ou recebidos do módulo de E/S

## Área de armazenamento temporário de dados

Data buffering
- Taxa de transferência dos dispositivos externos são baixas e variam de um dispositivo para o outro.
- Memória -> módulo de E/S = transferência rápida
- Módulo de E/S -> memória
    - Armazena os dados que veio do dispositivo no buffer e quando o buffer estiver lotado você manda os dados para a memória

## Controle de erros

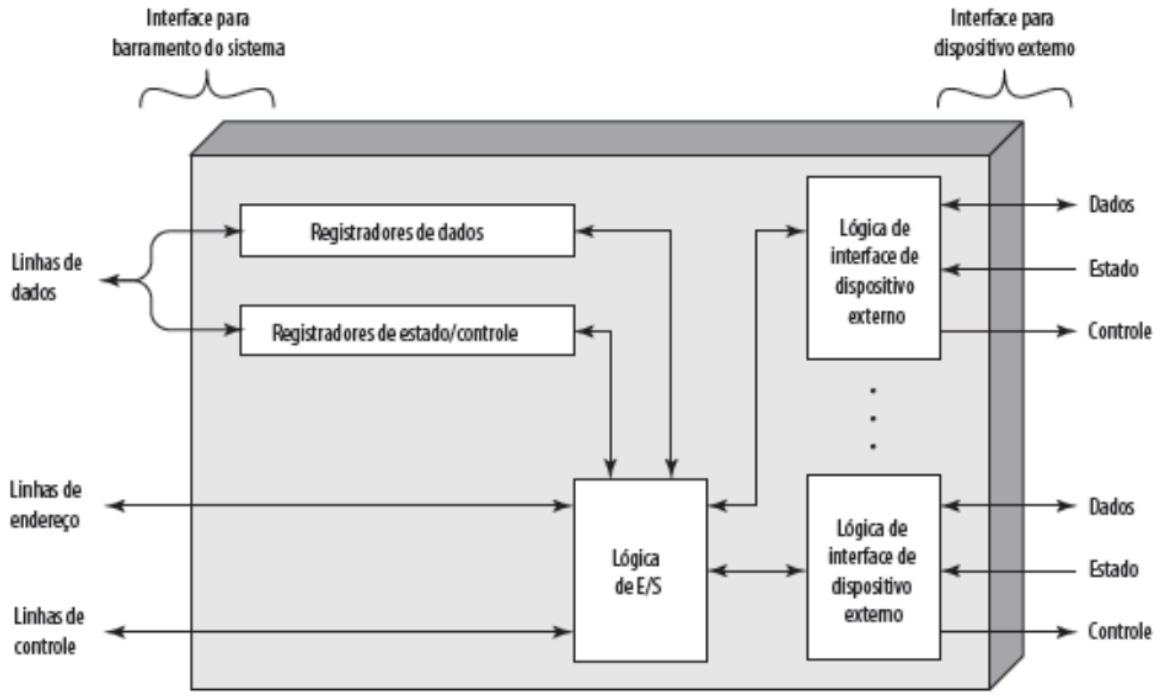
Possíveis erros:

- Mau funcionamento mecânica ou elétrico, alteração nos padrões de bits
    - Erro grotesco né, lab de física nos ensinou que não iremos considerá-lo

Utilização de um código de detecção de erro (último bit do ascii)

## Estruturas dos módulos de E/S

- Quanto maior a quantidade de dispositivos a qual pode se conectar mais complexo
- O módulo é conectado ao restante do sistema através de linhas de sinais
- Registradores de dados (buffer)
    - Os dados que são transferidos são armazenados temporariamente em registradores de dados
- Registradores de estado
    - Armazenam o estado atual do dispositivo
    - Esses mesmos registradores podem ser utilizados pela CPU para enviar comandos
- A interação com o processador é através de linhas de controle
    - O processador envia os comandos através dessas linhas
- Cada dispositivo que o módulo se conecta tem um endereço
    - Tem uma lógica específica para fazer interface com




## Operações de E/S

- Podem ser realizadas por meio do processador ou diretamente entre a memória e o módulo
- Podem ser interrompidas no meio
- Três técnicas de operação
    - E/S programada

        - Com o processador e sem interrupção
    - E/S dirigida por interrupção
        - Com o processador e com interrupção
    - Acesso direto à memória (DMA - Direct Memory Access)
        - Diretamente com interrupção

Comandos de E/S

- CPU envia um dos 4 comandos
    - Control
        - ativa o módulo e fala o que deve ser feito
    - Test
        - Testar diversas condições
    - Read
        - Ler uma informação do dispositivo
    - Write
        - Escrever uma informação no dispositivo

Endereçamento dos dispositivos de E/S

- E/S mapeada na memória
    - A memória e os dispositivos de E/S compartilham o mesmo espaço de endereçamento
        - Dispositivos de E/S são tratados como se fossem partes da memória do sistema
    - Espaço do endereçamento único
        - Tanto a memória quanto os dispositivos de E/S compartilham o mesmo espaço de endereçamento.
    - Registradores de dados e estados seriam posições na sua memória ram.
    - Vantagens:
        - Simples
        - Integração de espaço (tudo é em um espaço só)
    - Desvantagens
        - Pode haver conflitos entre os endereços da memória e os endereços dos dispositivos de E/S
        - Gasto de memória

- E/S independente
    - Utiliza um espaço de endereçamento separado específico para os dispositivos de E/S
    - Endereços utilizados para acessar dispositivos de E/S são diferentes dos usados para acessar a memória principal
    - É necessário ter instruções especiais para acessar esses endereços
        - Essas instruções são: IN e OUT.
            - Utilizadas para ler e escrever em portas de E/S
    - Vantagens:
        - Isolamento
            - Como os endereços são separados diminui a possibilidade de conflito
    - Desvantagens:

        - Complexidade para acessar os endereços
        - Perda de performance

E/S programada

- Existe um programa, executado pelo processador para manipular todas as operações de E/S
- Muito tempo é perdido, pois o processador é muito mais rápido que o módulo de E/S e tem que ficar esperando pela operação de E/S
- Passo a passo
    - CPU requisita uma operação de E/S
    - Módulo de E/S realiza a operação
    - Módulo de E/S seta o bit de estado para indicar o status da operação
    - CPU fica constantemente verificando o bit de estado
        - A CPU pode ficar verificando o tempo todo até o bit ser 1, ou seja, ficar em um while true (burrice)
        - A CPU verifica, se for 0, vai fazer outra coisa e depois volta e verifica de novo se está como 1.
    - O módulo de E/S não informa a CPU diretamente
        - O módulo não manda para a cpu quando tem uma interrupção, ou seja, ou quando da pau ou quando a operação termina.
        - A cpu tem que ficar verificando o bit constantemente
- Desvantagem
    - CPU espera pelo dispositivo
        - Como os dispositivos podem ser lentos isso pode dar gargalo
    - Se a CPU iria ficar parada não tem problema
        - Caso contrário é necessário usar outra forma de E/S

E/S Dirigida por interrupção

- Tem como objetivo diminuir o tempo gasto nas operações de E/S
- Inicialização da transferência vem do dispositivo
- Interrupção
    - O programa em execução é interrompido para execução de uma tarefa mais urgente
    - Essa tarefa mais urgente seria o que o dispositivo de E/S quer fazer
        - O teclado manda um sinal de urgência para o processador dizendo que ele deve processar o que o teclado quer.
- Passo a passo
    - CPU emite comando de leitura
        - Esse comando não existe para o teclado
        - Esse comando existe para o SSD, por exemplo
    - Módulo recebe dados do dispositivo enquanto a CPU faz qualquer outra coisa
    - Módulo interrompe a CPU
    - CPU solicita dados
    - Módulo transfere dados
- O sinal de interrupção está presente no ciclo de instrução da CPU

    - Se durante o ciclo de instrução o sinal estiver ativado a CPU pausa a instrução, salva o sinal e faz a interrupção
- Vantagem:
    - A CPU não fica parada mais
- Desvantagem:
    - Ela ainda precisa gerenciar a transferência de dados durante interrupções que podem sobrecarregar o processador caso tenha muitas interrupções ou grandes volumes de dados.

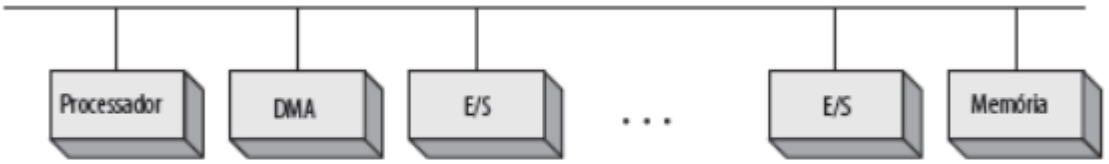
E/S com Acesso Direto à Memória (DMA)

- Grande problema da comunicação entre o módulo de E/S e o processador
    - O processador gasta muito tempo tendo que lidar com os processo de E/S
    - Se tiver muitos processos pode acabar sobrecarregando o processador ao ponto de gerar gargalo no módulo de E/S
- Os dados trocados entre a memória e o módulo não precisam passar pela CPU
- Quando a quantidade de dados é muito grande acaba tendo ganha de desempenho com DMA
- Tem um módulo adicional (hardware) conectado ao barramento
    - Tem um processador dedicado à DMA que substitui a CPU nas operações de E/S com DMA
- Passo a passo
    - CPU programa a DMA para transferir bloco de dados
        - CPU continua processamento
        - DMA executa transferência
    - DMA encerra operação
        - Envia interrupção para a CPU
    - Processador executa próxima instrução
- O controlador de DMA (“processador dedicada”) tem controle do barramento de endereços, dados e controle enquanto a CPU executa operações internas
- Duas maneiras de implementar uma DMA
    - Roubo de ciclo
        - Forçar o uso do barramento, suspendendo o processador temporariamente
        - Redução no processamento
        - Não tem interrupção
            - Quando tem uma interrupção a CPU salva o que estava fazendo e realiza a nova tarefa. No contexto do roubo de ciclo a CPU não salva nada, ela simplesmente fica parada até que o DMA pare de utilizar o barramento.
        - Atrasa a CPU, mas não tanto quanto fazer a CPU realizar a transferência
        - Se a CPU não está usando os barramentos o DMA só vai e os utiliza, agora se a CPU está usando ocorre o roubo de ciclo
    - Modo transparente
        - O controlador DMA só realiza transferências de dados durante os ciclos nos quais a CPU não está utilizando os barramentos.

- Transferência mais lenta, entretanto não atrapalha o processador

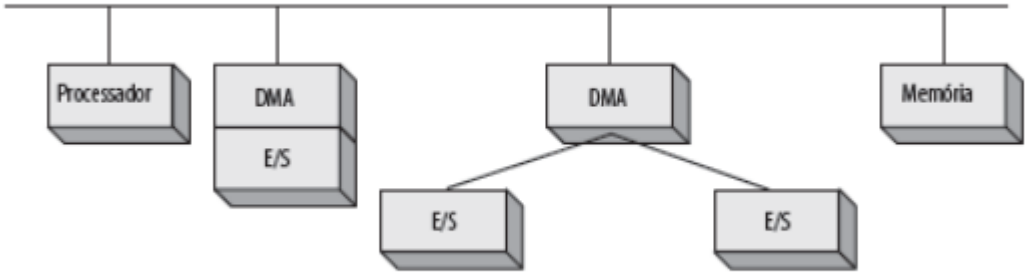
Barramento com a DMA

- Barramento único
  - Cada transferência usa o barramento duas vezes.
    - E/S para DMA
    - DMA para memória
  - Configuração mais barata e eficiente

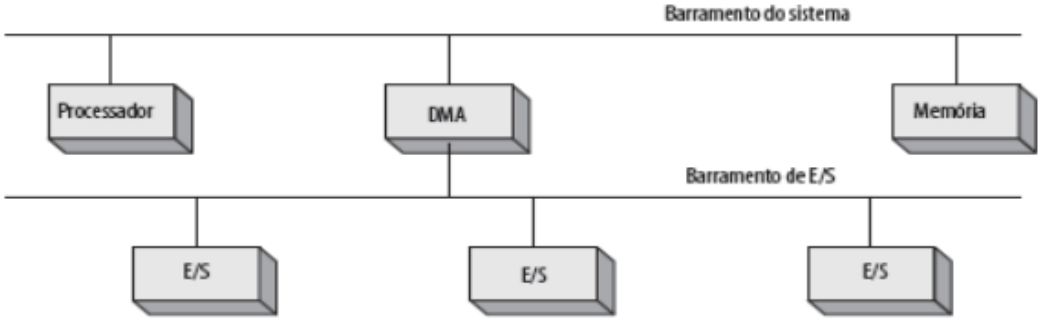



(a) Único barramento, DMA separado

- Barramento único, DMA - E/S integrados
  - Existe linhas entre a DMA e os dispositivos de E/S, fazendo com que a comunicação entre eles não tenha que passar pelo barramento
  - Cada transferência usa o barramento somente uma vez




(b) Único barramento, DMA-E/S integrados




(c) Barramento de E/S

## Interfaces de entrada e saída

Controlador

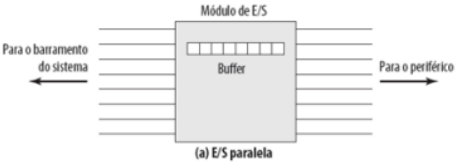
- Circuitos integrados que fornecem uma função específica na placa mãe
    - Exemplo: USB controller, SATA controller
    - O módulo de E/S contém esses controladores

Interface

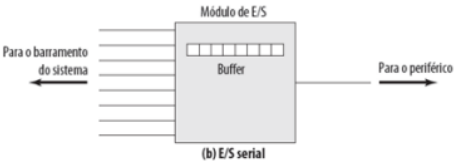
- Padrões através dos quais os dispositivos são conectados
    - HD se conecta ao controlador do HD através de uma interface SATA
    - O módulo de E/S também pode conter essas interfaces

Interfaces externas

- Modos de transferência de dados
    - Paralela
        - Várias linhas de conexão entre o módulo de E/S e o periférico
        - Vários bits são transferidos ao mesmo tempo
        - Usada em periféricos de alta velocidade




    - Serial
        - Somente uma linha de transmissão de dados
        - Bits transmitido um de cada vez
        - Comum para impressoras e terminais



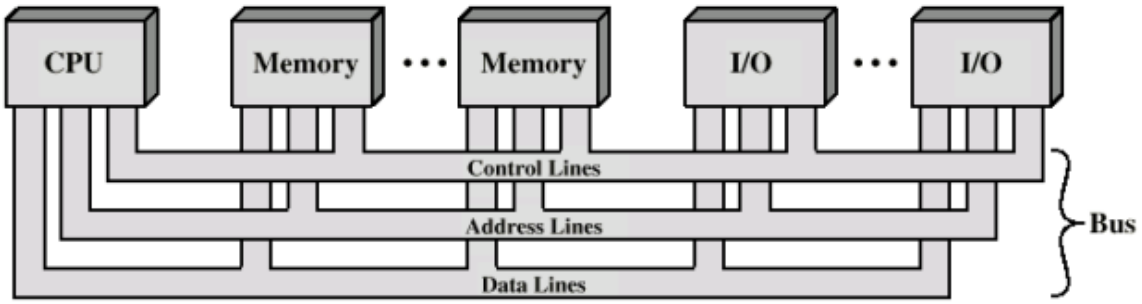

- Modos de conexão ao computador
    - Ponto a ponto
        - Linha dedicada entre o módulo de E/S e o dispositivo
        - Conecta, por exemplo, teclado, impressora e modem
        - Exemplo: RS-232
    - Multiponto
        - Barramento
            - Vários dispositivos compartilham a mesma linha de comunicação
        - Utilizada para conexão de dispositivos externos de armazenamento em massa
        - Neste tipo de conexão, dispositivos que estão conectados juntos podem trocar informações entre si sem ter que passar pelo processador
        - Exemplo: SCSI

## Barramentos

- Caminho físico entre dois ou mais dispositivos
- Somente um dispositivo pode transmitir sinal pelo barramento a cada instante

Três grupos funcionais

- Linhas de dados
- Linhas de endereço
- Linhas de controle




Barramento de dados

- Transmite: dados e instruções
- Transmite tanto dados quanto instruções da mesma maneira
    - Não existe uma distinção ao tratar de um ou de outro
- Largura
    - 8, 16, 32 ou 64 bits

Barramento de endereço

- Endereça portas de E/S e memória RAM.
- Identificam a origem e o destino dos dados transferidos pelo barramento de dados
    - O barramento de endereço indica onde na memória tem que ir para poder ler aquele endereço e transmitir os dados
- A largura deste barramento determina a capacidade máxima da memória
    - Determina quantos endereços de uma vez podem ser acessados na memória

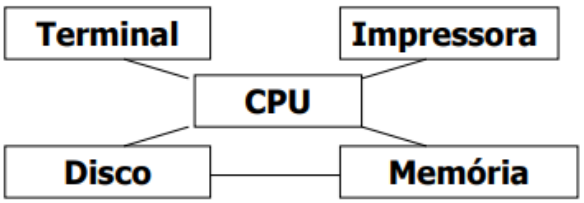
Barramento de controle

- Controla e gerencia o acesso ao barramento de dados e ao barramento de endereço
    - Garante que somente um dispositivo os utilize por vez
- Transmite comandos e sinais de temporização
    - Escrita/Leitura na memória

    - Escrita/Leitura em portas de E/S
    - Requisição, concessão do barramento etc.

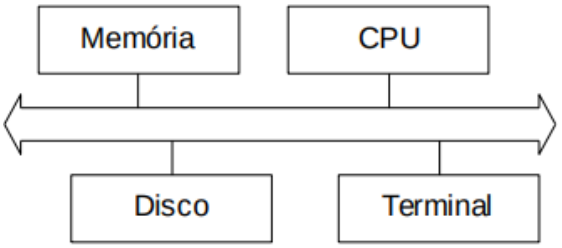
Conexão ideal: ponto a ponto

- Liga a CPU com todos os elementos de maneira única
- Caro e complicado




Barramento comum

- Todos os dispositivos ligados ao mesmo barramento
- Problemas
    - Muitos dispositivos conectados a um único barramento
    - Quanto mais dispositivos mais atraso vai ter
    - Falta de paralelismo




Hierarquia de Barramentos

- Barramento do processador
    - Utilizado pelo microprocessador para enviar dados dentro do processador
    - Leva as coisas a ULA, etc tlg.
- Barramento da cache
    - Dedicado para acesso à memória cache
- Barramento de memória (System Bus ou Barramento do Sistema)
    - Conecta a memória ao processador
- Barramento de E/S local (High speed bus)
    - Barramento de alta velocidade
    - Conecta processador, memória com dispositivos de E/S de desempenho críticos (placa de vídeos, redes, etc.)
    - PCI é um exemplo desse tipo de barramento
- Barramento de E/S padrão (Expansion bus)
    - Barramento mais lento
    - Utilizado para conectar os dispositivos de E/S mais lentos (mouse, modem, placa de som, etc.)

## Conceitos gerais

Mestres

- Ativam a transferência de informação no barramento
- Pode existir mais de um mestre
    - Tem um esquema para selecionar quem
- Somente 1 mestre pode ativar transferências no barramento em um dado instante

Escravos

- Recebem informações enviadas pelo mestre

Troca de informação (transação):

- Um ou mais escravos podem ser selecionados por um mestre
- O mestre endereça o escravo
- Os escravos comparam os endereços dos barramentos com o próprio endereço
    - Para ver se ele foi selecionado
- Se for o mesmo endereço
    - Escravo se conecta ao barramento
- durante a conexão ocorre troca de informações
- Ao final o mestre quebra a conexão com os escravos

Um subsistema pode ser tanto Mestre quanto escravo, entretanto nunca podem ser os dois ao mesmo tempo.

Transações entre Mestre-Escravo

- endereços
- dados
- controle (temporização, por exemplo)

Características de um barramento

- Tipo
    - Dedicado
    - Multiplexado
- Método de arbitragem
    - Centralizado
    - Distribuído
- Temporização
    - Síncrona
    - Assíncrona
- Largura do barramento
- Tipos de transferência de dados

Tipos de barramentos

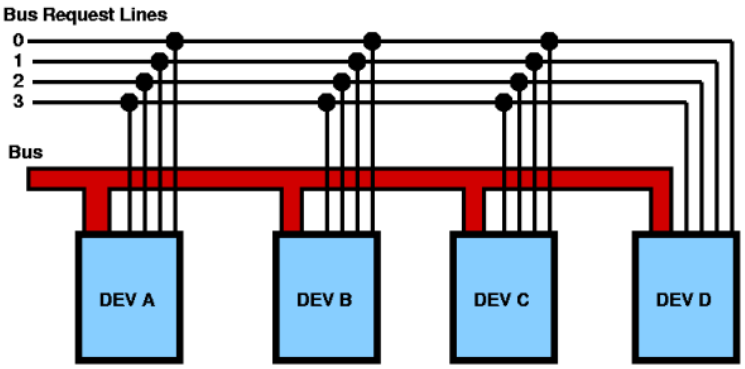
- Dedicado
    - Linhas separadas são utilizadas para transferência de dados e endereços
    - Vantagem
        - Alta taxa de transferência
    - Desvantagem
        - Maior custo e tamanho
- Multiplexado
    - As linhas são compartilhadas para transmissão de dados, endereços e controle
    - Vantagem
        - Poucas linhas
    - Desvantagens

        - Unidades mais complexas
        - Redução de desempenho

Os maiores barramentos são aqueles para fazer transferências de dados e endereços.

Método de arbitragem

- Existe mais de um módulo controlando o barramento
    - CPU, controlador de DMA, etc.
    - Somente um pode controlar por vez
- Existe a necessidade de arbitragem para que não exista colisão
- Método de arbitragem - Centralizada
    - Árbitro
        - É um componente responsável por gerenciar o acesso ao barramento
        - Determina quem vai poder usar o barramento e por quanto tempo
    - PCI tem o método de arbitragem centralizada
    - Como funciona
        - Mestre pede o uso do barramento
        - Árbitro aloca ou não o barramento, dependendo da prioridade
    - Prioridade
        - O barramento é sempre alocado ao mestre com maior prioridade
        - Pode causar monopólio
    - Round-Robin
        - O barramento é dado aos mestre de modo cíclico
            - Um de cada vez
    - Prioridade variante
        - Toda vez que um mestre tem um pedido negado sua prioridade é aumentada
        - Quando um mestre tem o pedido atendido seu valor de prioridade é reiniciado
    - Vantagem
        - Simplifica o gerenciamento
            - Único ponto de controle
    - Desvantagens
        - Possível gargalo
            - Por só ter 1 cara controlando
        - Complexo para tornar eficiente
- Método de arbitragem - Distribuída
    - Mais de um módulo pode controlar o barramento
    - A escolha do mestre é feita de maneira distribuída
    - O barramento será dado ao mestre de prioridade mais alta
        - Essa prioridade é definida através de vários módulos




## Temporização

- Modo pelo qual os eventos são coordenados no barramento
- Tem que ser sincronizado para indicar a validade da informação
- Um ciclo do barramento pode levar vários ciclos de clock
    - Em cada ciclo de clock tem que ter um sincronismo para validar as informações
- Temporização síncrona
    - Sincronização feita com base em um sinal de clock
    - A operação que o barramento vai fazer se inicia no início de um ciclo de clock
- Temporização assíncrona
    - Cada operação é iniciada e completada de maneira independentemente do clock
    - Existência de sinais de controle que indicam o início e conclusão das operações
        - Essa troca de sinais se chama handshake

## Transferência de dados

- Mestre -> Escravo
    - Escrita
- Mestre <- Escravo
    - Leitura
- Read Modify Write (RMW)
    - Primeiro se faz uma operação de leitura para aí fazer a escrita
        - A escrita deve ser feita sem ter interrupção para evitar inconsistência
- Read After Write (RAW)
    - Ocorre a escrita e logo depois a leitura
        - Essa leitura serve para garantir a integridade da escrita
- Split Transfer
    - Permite a interrupção e retomada de uma transferência de dados quando eles ainda não estão disponíveis

Largura do barramento

- Barramento de dados
    - Quanto maior for a largura mais dados conseguem ser transmitidos de uma vez

- Barramento de endereço
    - Tem um grande impacto sobre um sistema como um todo, pois quanto maior for a largura maior o número de posições de memória que podem ser endereçadas

Bandwidth do barramento

- Soma total de dados que podem ser transferidas em um barramento em uma dada unidade de tempo