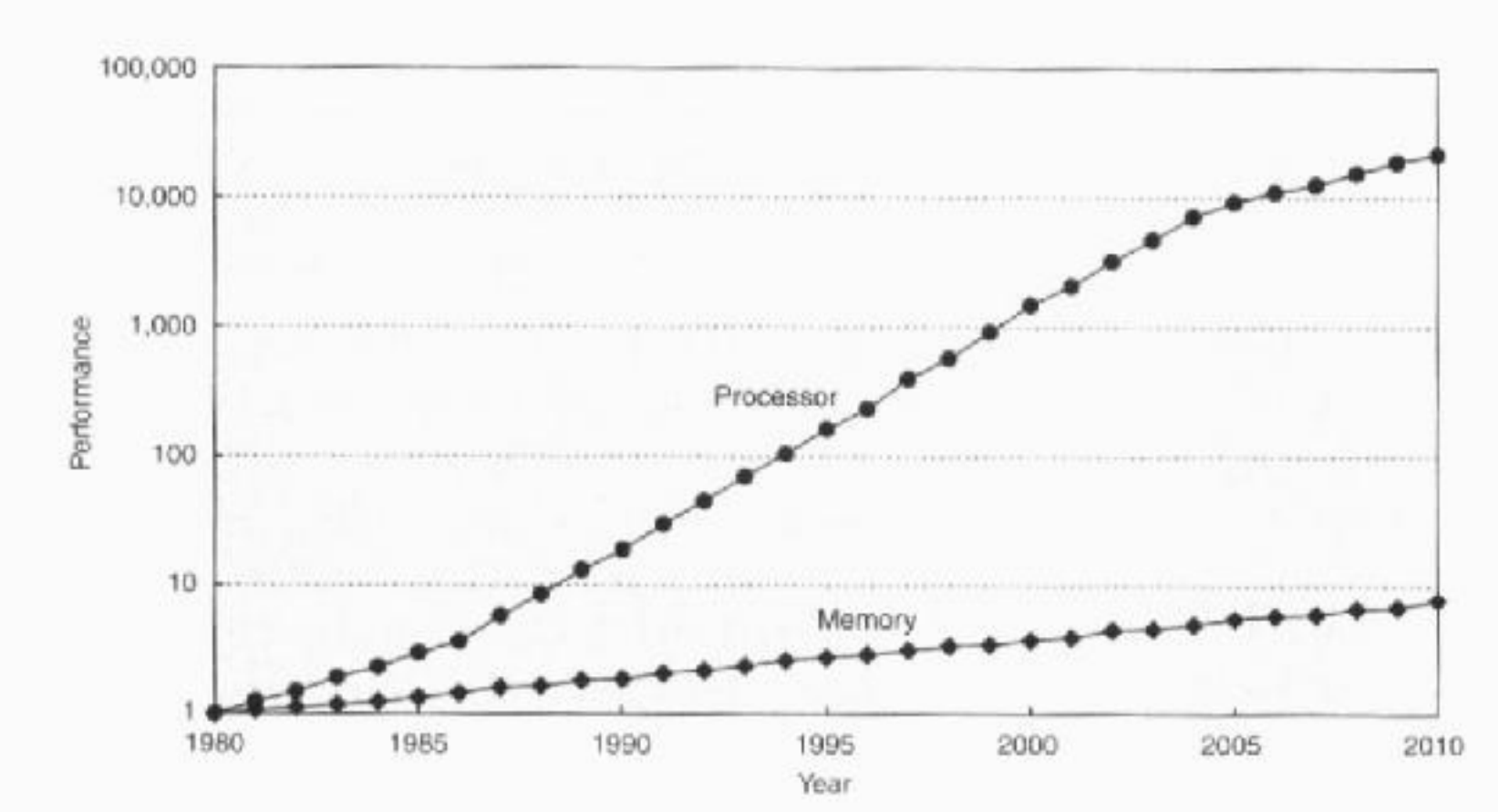
Sistemas de Memória

CES-25 – Arquiteturas para Alto Desmpenho
Prof. Paulo André Castro
pauloac@ita.br
Sala 110 – Prédio da Computação
www.comp.ita.br/~pauloac
IEC - ITA

Memória: O Gargalo de Von Neuman

- **Memória principal:** considerada como sendo o **gargalo** da *arquitetura de Von Neumann*:
- O tempo de resposta da memória principal pode ser de **muito maior** que o tempo de execução do processador
- Se o processador não recebe instruções tão rapidamente quanto ele pode processar, o que ele faz?
- Esse desnível de desempenho está diminuindo? Ou aumentando?

Desnível de Desempenho Processador x Memória



Sistema de Memória

- Porque toda a memória de um computador não pode ser feita da mesma tecnologia da CPU ?
- Seu custo seria elevadíssimo, pois o volume de informações é muito grande.
- A tecnologia da CPU tipicamente produz memórias voláteis.

Sistema de Memória

- Muitas informações devem permanecer, mesmo após a máquina ser desligada.
  - Memória não-volátil
- Volatilidade versus Desempenho
  - Memórias voláteis tender a ser sempre mais rápidas que memórias não-voláteis

- Importante: Grande parte das informações não é utilizada considerando pequenos intervalos de tempo.

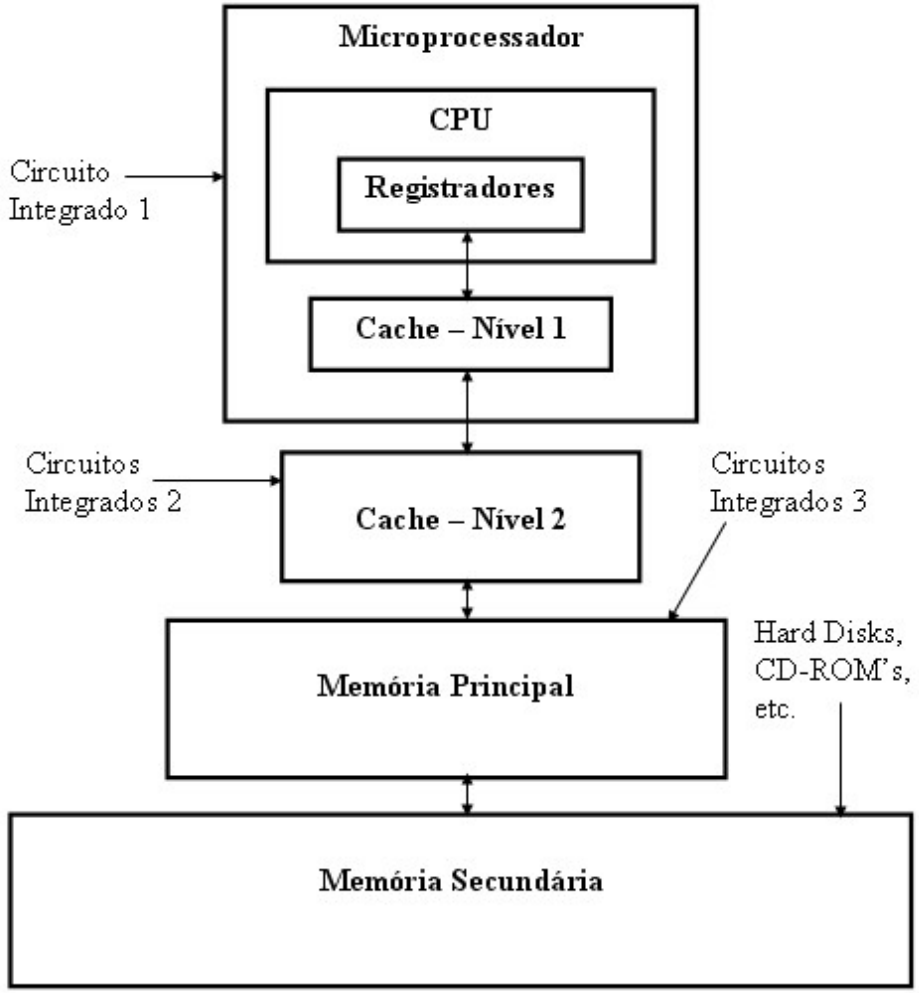
Princípio da Localidade

- “Os programas tendem a reutilizar dados e instruções que usaram recentemente.”
  - “Regra prática”: um programa passa 90% de seu tempo de execução em 10% de seu código.
  - Também aplica-se a dados mas não tão fortemente quanto em instruções.

- Dois Tipos de localidade:
  - Localidade espacial
  - Localidade Temporal

- Como tirar vantagem dessa característica para obter mais desempenho?

Hierarquia de Memória



Perguntas

- Os níveis mais altos devem ser sempre mais rápidos e os mais baixos maiores. Por quê?
- Em que situação é vantajoso introduzir novos níveis na hierarquia?
- Ao colocar informações nos níveis mais altos, deve-se apagar dos níveis mais baixos?

Níveis de Memória Visíveis

- Registradores
  - Memória de trabalho da CPU para armazenamento temporário de instruções e dados;
  - registradores de propósitos gerais, vários registradores de propósitos específicos (AR, PC, IR, SP, Flags, etc.);

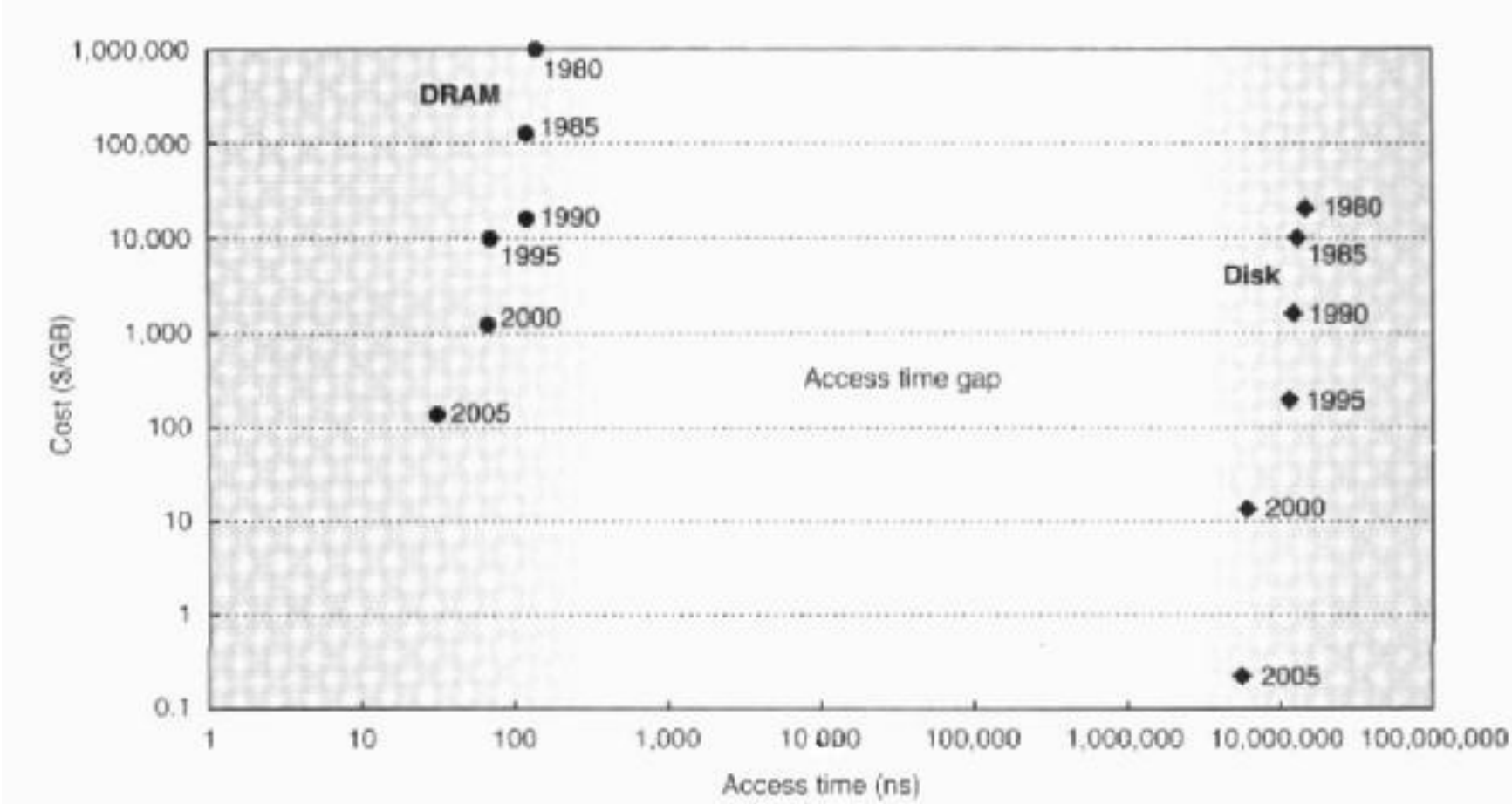
- Memória Principal
  - Em comparação com os registradores da CPU, sua tecnologia costuma ser **inferior** e sua **capacidade de armazenamento é bem maior**
  - Deve ser suficientemente grande para armazenar os programas em execução e seus respectivos dados
  - Até 70 vezes mais lenta que os registradores
- Memória Secundária
  - Não volátil
  - Memória de overflow
  - Grande desnível em relação a memória principal

Hierarquia de Memória

Memória secundária

- Armazena os dados não voláteis
  - programas do sistema de software do computador, arquivos de dados, etc;
- Funciona como memória de ***overflow***, quando a capacidade da memória principal é excedida (***memória virtual***);
- É acessada pela CPU por meio de controladores específicos e programas de **I/O** que transferem dados entre ela e a memória principal;
- Grande desnível em relação a memória principal em termos de desempenho e tamanho

Memória vs Disco
Custo/GB x Tempo de Acesso



Memória Cache

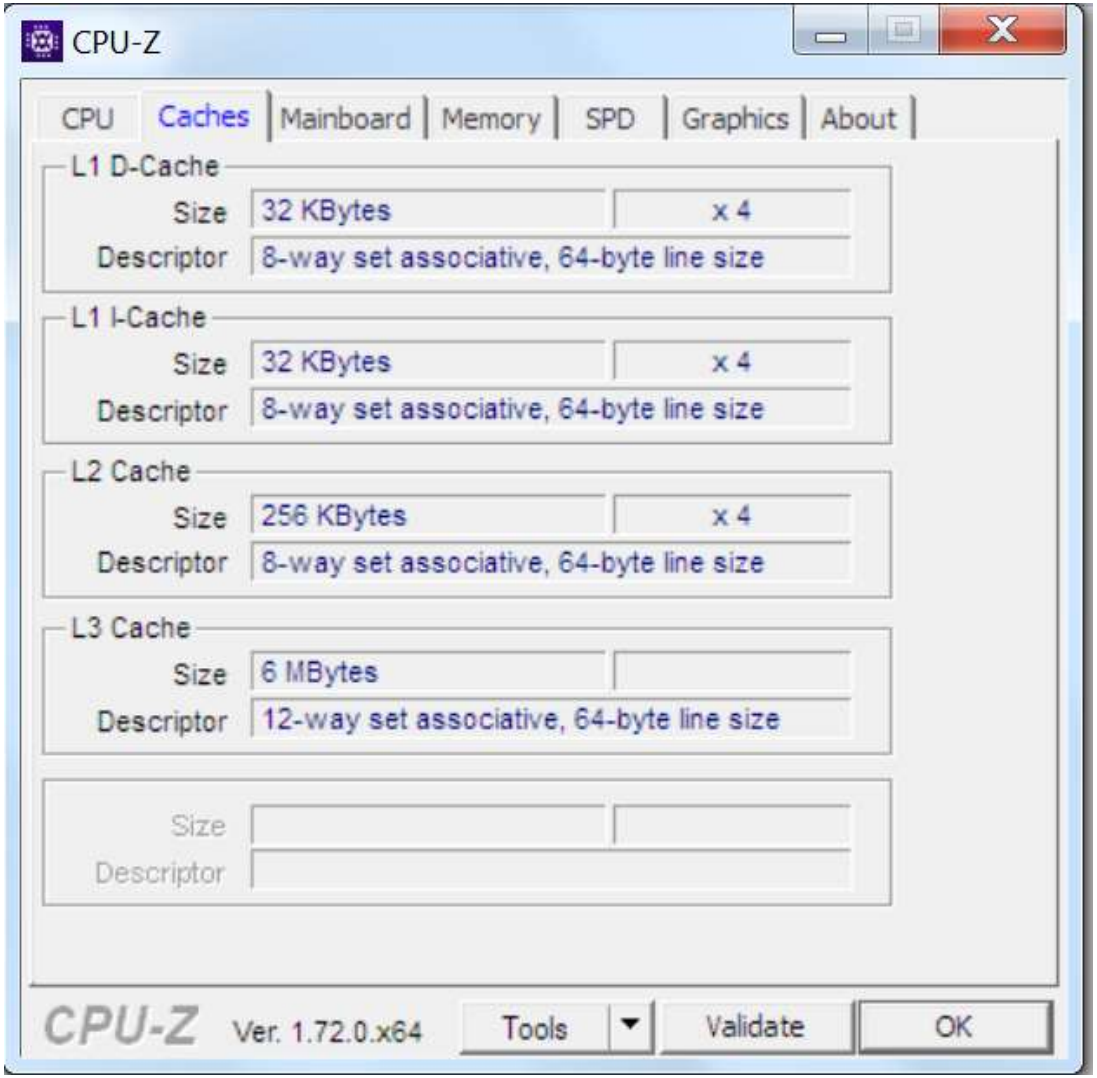
- Armazena cópia das regiões da memória principal mais freqüentemente usadas pela CPU, num curto intervalo de tempo;
- Diferente das outras memórias caches são transparentes aos programadores (mesmo em assembly);
- Computadores com vários níveis hierárquicos de cache são comuns atualmente, tipicamente os níveis mais altos ficam no mesmo circuito integrado da CPU;

Memória Cache 2

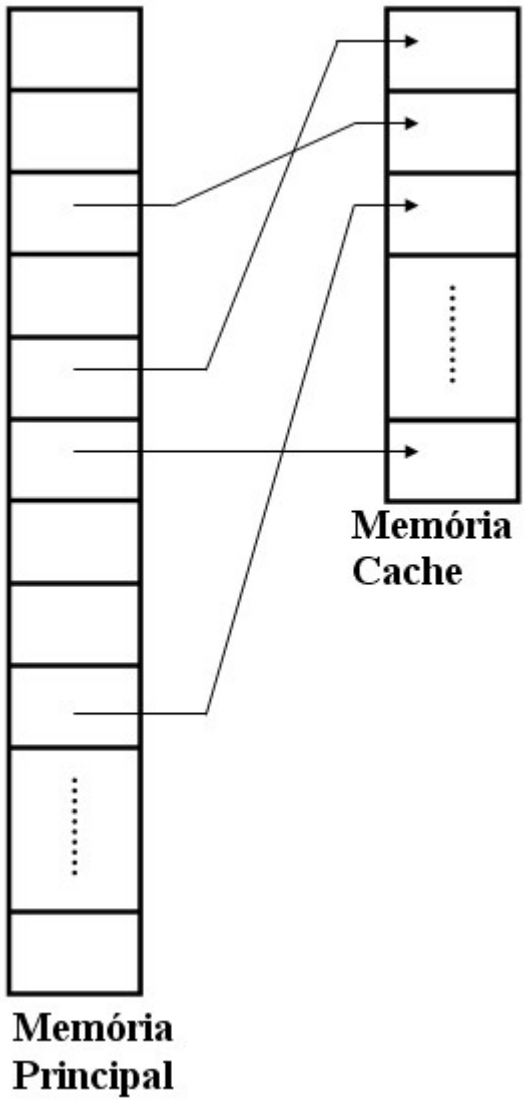
- Em multiprocessadores,
  - Várias processadores demandam informações de uma memória, logo a cache também é fundamental
  - há complexidade adicional para controlar o acesso a cache de modo coerente

- Muitas vezes é dividida em Cache de dados e cache de instruções
- Cache de Instruções
  - A **próxima instrução** a ser executada é retirada do cache de instruções;
  - não estando ali presente, **nova seqüência** é carregada no CI:
  - Caso o CI contenha **aninhamentos inteiros de laços** com grande número de repetições, serão evitados inúmeros **acessos à memória**

Exemplo Sistema de Cache



Funcionamento da Hierarquia de Memória



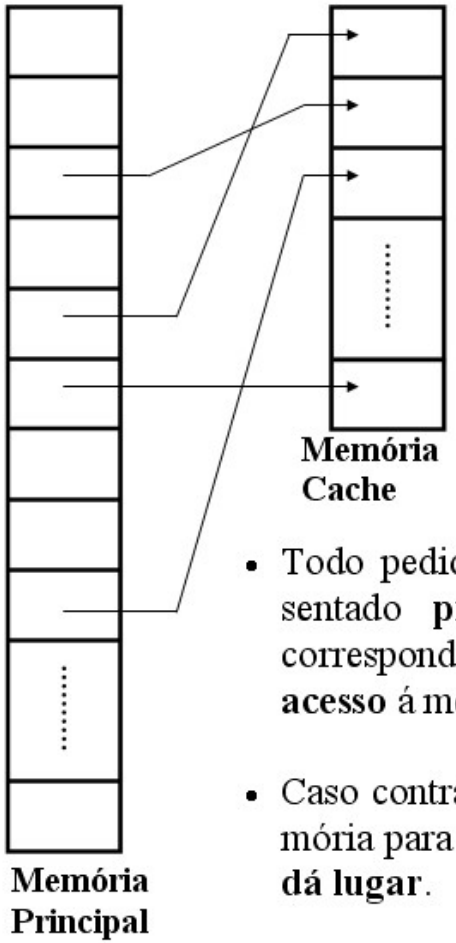
- Os níveis de hierarquia de memória (caches, principal, secundária) são divididos em blocos de palavras chamadas linhas
- Em dado momento, algumas linhas (blocos) tem cópia dois ou mais níveis
- Todo pedido de acesso é apresentado primeiro ao nível mais alto, se está lá evita-se o acesso ao nível inferior
- Caso contrário, tal linha é trazida do nível inferior para o nível superior

Perguntas

- Porque os blocos de transferência mais baixos são iguais ou maiores?

- Pode ocorrer que um dado esteja em nível superior, mas não no nível logo inferior? Como?

Detalhamento do Funcionamento da Memória Cache



- Memória principal e cache são divididas em blocos de palavras chamadas **linhas** (normalmente uma linha tem 8 palavras).
- Num dado momento, algumas linhas da memória principal têm **duplicata** na cache.
- Todo pedido de acesso à memória é apresentado **primeiro à cache**; se a linha correspondente lá estiver, **evita-se um acesso** á memória principal.
- Caso contrário, tal linha é trazida dessa memória para a cache; outra linha da cache lhe **dá lugar**.

Quatro perguntas sobre Organização da Cache

- **P1: Onde um bloco pode ser inserido no nível superior (posicionamento do bloco) ?**
- P2: Como um bloco é encontrado se está no nível superior da hierarquia (cache) ?
- P3: Que bloco deve ser substituído ao ocorrer uma falha (substituição do bloco) ?
- P4: O que acontece em uma gravação (estratégia de gravação) ?

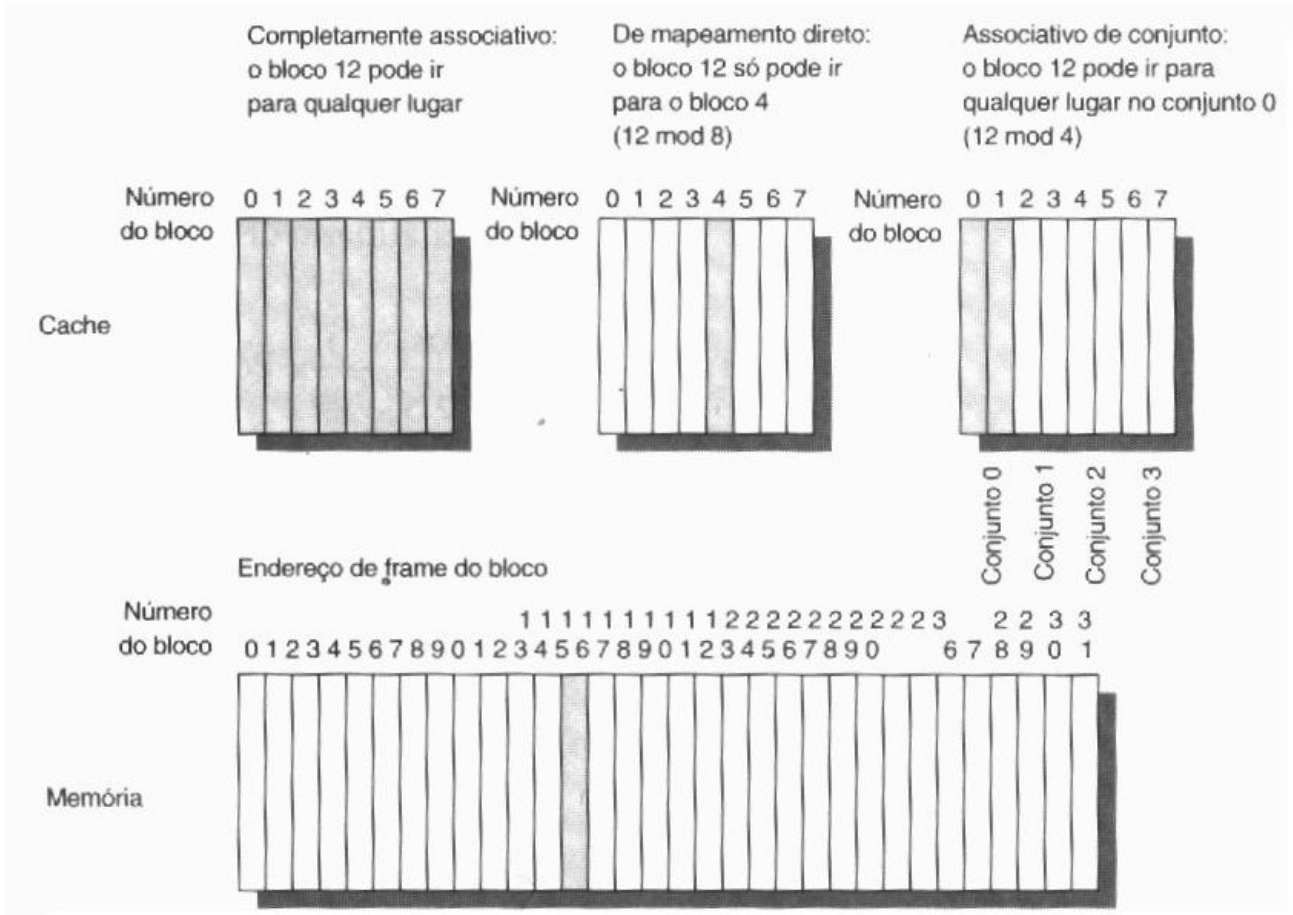
Posicionamento do Bloco

- Abordagens:
  - Cache de Mapeamento Direto
    - Cada bloco da memória só pode ser colocado em uma posição da memória
    - Ex. (Endereço do Bloco) mod (Número de blocos na cache)
  - Cache Completamente Associativo
    - Cada bloco pode ser alocado em qualquer posição da cache
    - Posição definida pela estratégia de substituição

Posicionamento do Bloco

- Abordagens (Continuação)
  - Cache Associativo de Conjunto
    - Um bloco da memória pode ser deslocado para um único determinado conjunto (o qual conta com um certo número de blocos), dentro do qual o bloco pode ser colocado em qualquer posição
    - Ex.: Conjunto Escolhido = (Endereço do Bloco) mod (Número de conjuntos na cache)

Abordagens de Posicionamento



Vantagens e Desvantagens das Abordagens

- Mapeamento Direto
  - Vantagens: Simplicidade
  - Desvantagens: Possível Ineficiência com inflexibilidade
- Completamente Associativo
  - Vantagens: Total flexibilidade
  - Desvantagens: Complexidade e custo de implementação
- Associativo de Conjunto
  - Vantagens e Desvantagens: Meio termo entre as anteriores

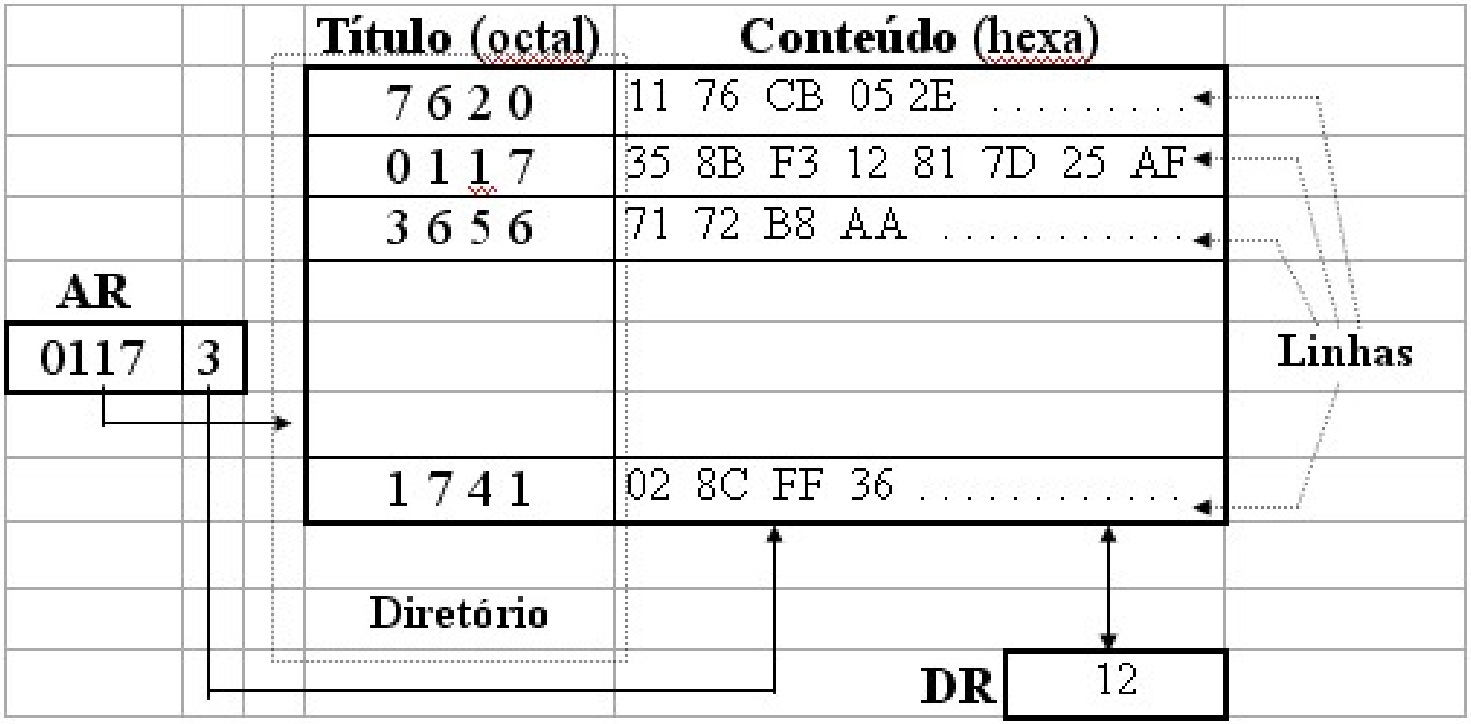
Quatro perguntas sobre Organização da Cache

- P1: Onde um bloco pode ser inserido no nível superior (posicionamento do bloco) ?
- **P2: Como um bloco é encontrado se está no nível superior da hierarquia (cache) ?**
- P3: Que bloco deve ser substituído ao ocorrer uma falha (substituição do bloco) ?
- P4: O que acontece em uma gravação (estratégia de gravação) ?

Conceitos Úteis

- A cache é divida em blocos com um número fixo de palavras de memória (nível mais alto)
- O endereço de memória é então divido em duas partes:
  - O endereço do bloco (às vezes, chamado de título, ou tag)
  - A posição da palavra dentro do bloco (offset)
- Bits de offset = $\log_2$ (Tamanho do Bloco)
- Endereço do Bloco = Endereço – bits de offset

Organização da Cache



Acessando a cache

- **Exemplo:**
  - Endereço 01173 $\rightarrow$ título 0117 $\rightarrow$ Sucesso $\rightarrow$ **DR$\leftarrow$12**
  - Endereço 01163 $\rightarrow$ título 0116 $\rightarrow$ Falta $\rightarrow$ Procura na memória principal

P2: Como um bloco é encontrado na cache ?

- O Tag marca o endereço de memória ao qual corresponde o bloco de cache e também se utiliza um bit para marcar a validade ou não do bloco (bit de validade)
- Tags sempre são pesquisadas em paralelo

P3: Substituição do Bloco

- Que bloco pode ser substituído ao ocorrer uma falha?
  - •Mapeamento direto: Não há decisão a ser feita, pois cada bloco é direcionado para um bloco de cache
  - •Mapeamento Completamente associativo ou de conjunto: várias opções
    - •Aleatória: Escolhe-se aleatoriamente um bloco para a substituição
    - •Menos recentemente usado(Least Recently Used): Registra-se os acessos aos blocos e se retira do cache o menos usado recentemente. Baseia-se no passado para prever o futuro

Implementação do LRU

- Implementação com Contador
  - Guardar para cada bloco, em um campo contador o número do último acesso (em clocks)
  - Substitui-se o bloco com menor clock
- Implementação com Pilha
  - Manter em uma pilha os números de cada bloco
  - Ao ser realizada acesso a um bloco X, este passaria ao topo da pilha
  - Substitui-se o bloco cujo número está na base da pilha

Implementação Aproximada de LRU

- Exige bastante do hardware
- Criar um campo de referência com n bits
  - Em cada acesso setar para 1
  - Deslocar
- Primeiro a entrar, Primeiro a sair(First In First out, FIFO), Esta opção se aproxima ao substituir o mais antigo, no lugar do menos recentemente usado

P4: Estratégia de Gravação

- As leituras dominam as operações de memória, mas também existem escritas. Segundo algumas estimativas, aproximadamente 10%

- Estratégias de Solução:
  - Write-Through: As informações são gravadas no cache e na memória inferior
  - Write-Back: As informações são gravadas apenas no cache. A gravação na memória ocorrerá apenas quando houver substituição do bloco.

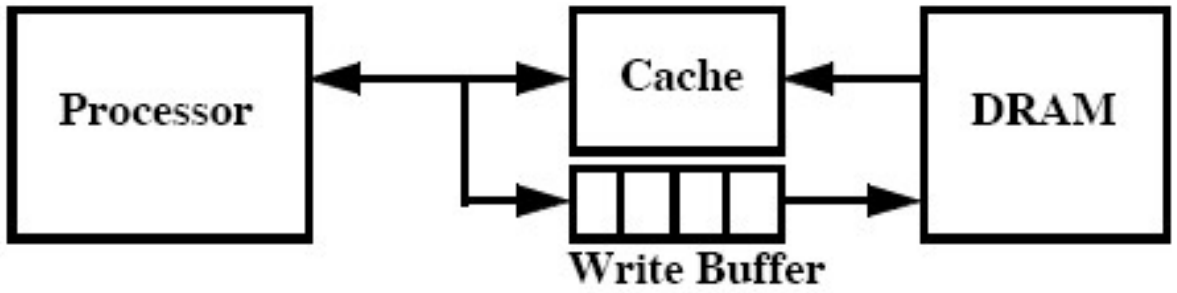
Estratégias de Gravação

- Write-Back
  - Vantagens
    - Usa menos largura de Banda da Memória (Multiprocessadores)
    - Poupa Energia por usar menos hardware
  - Desvantagens
    - Precisa controlar atualização da Memória
    - Falha de leitura pode causar gravação
    - Memória fica inconsistente com a cache (multiprocessadores)

Estratégias de Gravação

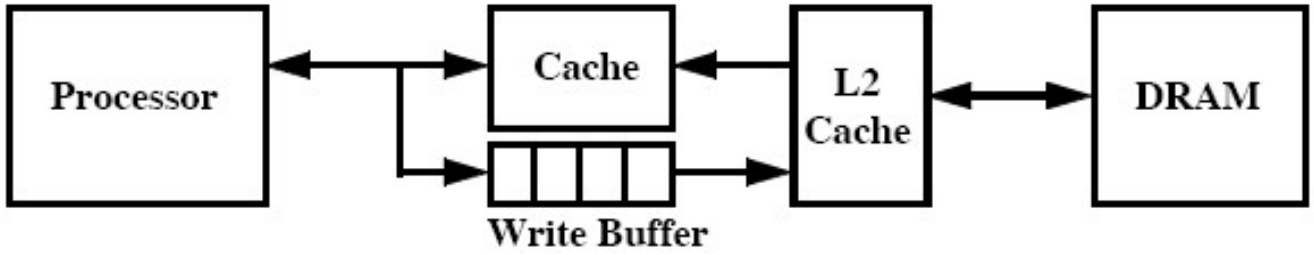
- Write-Through
  - Vantagens
    - Mais fácil de Implementar
    - Falha de leitura não causa gravação
    - Nível Inferior sempre coerente (multiprocessadores)
  - Desvantagem
    - Gravações sempre demoram o tempo do nível inferior, mesmo em caso de acerto

Buffer de Gravação



- Write-Through
  - Buffer de Gravação para evitar atrasar a CPU
  - FIFO (First In, First Out)
  - Tipicamente em torno de 4 entradas
- Problema: Frequência de Instruções de Gravação maior que frequência da DRAM (Saturação do Buffer)

Buffer de Gravação 2



- Problema: Freqüência de Instruções de Gravação maior que capacidade de atendimento da DRAM (Saturação do Buffer)

- Solução: Aumentar o desempenho das gravações através da introdução de outro nível de cache

Falha de Gravação

- Ao tentar gravar um dado que não está na cache há duas opções:

- Write Allocate (Gravação com Alocação) :
  - Traz o bloco que contém o endereço para a cache

- Write Not Allocate (Gravação sem Alocação):
  - O dado fica apenas na memória principal

Quatro perguntas sobre Organização da Cache

- P1: Onde um bloco pode ser inserido no nível superior (posicionamento do bloco) ?
- **P2: Como um bloco é encontrado se está no nível superior da hierarquia (cache) ?**
- P3: Que bloco deve ser substituído ao ocorrer uma falha (substituição do bloco) ?
- P4: O que acontece em uma gravação (estratégia de gravação) ?

Como melhorar o desempenho da Memória?

- Reduzir o ciclo de memória
  - usando memória cache
  - Melhorar taxa de acerto

- Aumentar o tamanho da palavra
  - Acessar várias palavras em paralelo (usando memória entrelaçada).

Definições

- **Ciclo de Memória:** tempo para devolver a CPU uma palavra
- **Palavra de Memória:** conjunto de bytes que pode ser entregue a CPU a cada requisição.
- **Taxa de Transferência:** número de bytes por unidade de tempo entregues pela memória.
- Exemplo:
  - palavra de 32 bits ou 4 bytes
  - ciclo de memória = 50 ns;
  - taxa de transferência: 640 Mbits/s ou 80 MB/s.

Desempenho da Memória

- Nem todo acesso à memória é atendido pela cache;
  - Quando é atendido → ***Sucesso*** na cache
  - Quando não → ***Falha*** na cache

- Grandeza Fundamental para o desempenho do sistema de Memória com Cache:
  - Taxa de sucesso (ou taxa de acerto)

Desempenho da Memória

- O tempo efetivo de um sistema com cache é um valor intermediário entre os tempos da cache e da memória principal.
  - $Tef = h * Tc + (1 - h) * Tm \quad (0 \leq h \leq 1)$
- **Taxa de sucesso h:** probabilidade de sucesso na cache
- **Taxa de falha (1-h):** probabilidade de falha na cache

Desempenho da Memória

- Taxa de Erro e Penalidade
  - T_{ef} = Tc + (Tx. Erro)* Penalidade
  - Tx. Erro = (1-h)
  - Penalidade = Tm-Tc
- Qual o tempo efetivo considerando dois níveis de caches ?
  - Tef = h * Tc + (1 – h) * [h_2 * Tc_2 + (1-h_2)*Tm]
  - Como se mede a taxa de acerto h_2 ?

Taxa de Acerto na Cache

- É comum a obtenção de valores altos para **h**; desse modo, o ciclo efetivo fica sensível a pequenas mudanças de **h**:
  - Caso **Tc = Tm / 10** e **h** caia de **0.99** para **0.98** (1%) então **Tef** sobe **8.3%** (quase 10%).
  - Caso **Tc = Tm / 10** e **h** caia de **0.99** para **0.89** (~10%) então **Tef** sobe **82.5%** (quase dobra).
  - Caso **Tc = Tm / 20** e **h** caia de **0.99** para **0.89** então **Tef** é multiplicado por **2.5**.

Desempenho de Memória

- **Pequenas** melhorias em **h** podem resultar em **substancial aumento** de desempenho no sistema de memória.

- Fatores que variam a razão de sucesso **h**
  - Número de palavras dos blocos e número de blocos;
  - Critério de escolha do bloco que dará lugar a um novo bloco vindo da memória principal, na ocorrência de uma falta (*Política de substituição de blocos*).

Exercício

- Considere um computador com
  - CPI(Clock por Instrução) = 1,0
  - Penalidade por erro = 25 ciclos
  - Taxa de erros = 2%
  - Acessos a memória representam 50% do total de instruções em média.
- Quão mais rápido seria este computador se não houvesse erro de cache?

Resposta

- Computador sem Erros:
  - Tempo de Execução Ideal = IC *1,0* Tempo de Clock

- Computador Real
  - Tempo de Execução Real= Tempo de Execução ideal + Tempo de Parada
  - Tempo de Parada = IC * Acesso a memória/Instrução * Taxa de Erros * Penalidade de Erro * Tempo de clock

Resposta

- Tempo de Parada:
  - TP= IC*(1+0,5)*0,02*25 = IC *0,75
  - (1+0,5) – 1 para acesso de instrução e 0,5 para acesso de dados da instrução
- Tempo de Execução Real
  - Tempo de Parada : IC*0,75 * tempo de clock
  - Tempo de Execução = 1,75*IC * tempo de clock

- Ganho = 1,75 *IC*TClock / 1,0*IC*TClock = 1,75

Medidas de Taxa de Erro de Cache

- Erro/Instrução
  - Ex.:3 falhas/1000 instruções
- Taxa de Erro /acesso a memória
  - Ex.: 2% dos acessos causam falha de cache
- Erro/Instrução = (Taxa de Erro * Acessos a Memória)/IC = Taxa de Erro * (Acesso)/Instrução

No Exemplo, anterior

- Erro/Instrução = Taxa de Erros *Acessos/Instrução
  - Erro/Instrução = 0,02* (1+0,5) = 0,03
- Recalculando o Tempo de Parada para o exemplo, obtemos o **mesmo** resultado
  - Tempo de Parada = IC * Erro/Instrução * Penalidade de Erro
  - TP = IC * 0,03 * 25 = IC * 0,75