# Lógica digital e organização de computadores

Memória Cache

TADS
Tecnologia em Análise e Desenvolvimento de Sistemas

Campus de Alegre

# Introdução

## Desafio

- A Diferença de Velocidade entre Processador e Memória Principal (MP).
  - A velocidade de processamento da UCP (Unidade Central de Processamento) tem crescido muito mais rápido (cerca de 50% a.a.) do que o tempo de acesso da MP (RAM) (cerca de 10% a.a.).
- **Problema:** O processador passa a maior parte do tempo ocioso (esperando) que os dados cheguem da MP.
- **Solução:** Inserir um nível de memória intermediário (Cache) para reduzir o tempo médio de acesso.

# O princípio da localidade

- O subsistema de cache é eficaz graças ao Princípio da Localidade (ou Princípio de Localização).
- Conceito: Durante a execução de um programa, os acessos à memória não são aleatórios; eles tendem a se concentrar em regiões específicas e por curtos períodos de tempo.
- Se os dados usados recentemente estiverem na Cache, a UCP os encontrará rapidamente.

# Tipos de localidade

- **Localidade Temporal:** Se um item de dado ou instrução é referenciado, ele provavelmente será referenciado novamente em breve.
  - Exemplo: Variáveis em loops (laços de repetição).
- **Localidade Espacial:** Se um item de dado ou instrução é referenciado, itens com endereços próximos a ele provavelmente serão referenciados em breve.
  - Exemplo: Acesso sequencial a vetores ou execução de instruções adjacentes.

## Termos chave da cache

- Bloco (Block): A unidade básica de transferência entre a MP e a Cache (e vice-versa). Contém dados ou instruções.
- Linha (Line): O local de armazenamento na memória cache. Cada linha da cache comporta exatamente um bloco de memória principal.
- Hit (Acerto): O dado solicitado pela UCP está presente na Cache. É um acesso rápido.
- Miss (Falta): O dado solicitado pela UCP não está presente na Cache. Acesso lento, pois exige busca na MP.

# Termos chave da cache

- **Taxa de Acerto** (Hit Ratio - H): A porcentagem de acessos à memória que resultam em Hits
  - $$H = \frac{Total\ de\ Acessos}{Acessos\ Acertados}$$
- **Taxa de Falha** (Miss Ratio - F): A porcentagem de acessos que resultam em Misses. ***F = 1 - H.***
- **Tempo Médio de Acesso** ($T_M$): O principal indicador de desempenho.
  - $T_M = (H \times T_{Cache}) + (F \times T_{MP})$
- O objetivo do projeto é maximizar $H$ e minimizar $T_M$.

# Tipos de uso da cache

- Cache Unificada: Armazena instruções e dados em um único espaço.
  - Vantagem: Uso mais eficiente do espaço de cache (dinâmico).
  - Desvantagem: Acesso simultâneo de instruções e dados pode gerar conflitos.
- Cache Separada (Split Cache): Separa a cache em Cache de Instruções ($C_I$) e Cache de Dados ($C_D$).
  - Vantagem: Reduz o conflito de acesso.
  - Uso Comum: Usualmente no **nível L1**.

# Níveis de Cache (L1, L2, L3)

- A maioria dos sistemas utiliza múltiplos níveis de cache para otimizar o desempenho e o custo.
  - L1 (Level 1): O nível mais próximo e rápido do processador.
  - Velocidade: Mais rápida (<2ns).
  - Capacidade: Menor.
  - Localização: Geralmente no chip da UCP (On-Chip).
- L2 (Level 2): Nível intermediário entre L1 e a MP.
  - Velocidade: Mais lenta que L1.
  - Capacidade: Maior.
  - Localização: Pode ser no chip ou em um módulo separado

# Níveis de Cache (L1, L2, L3)

- L3 (Level 3): O nível de cache mais lento e maior.
  - Velocidade: Mais lenta que L2.
  - Capacidade: Maior.
  - Localização: Geralmente no chip ou fora dele, compartilhada por múltiplos núcleos.
  - Regra Geral: A busca por um dado ocorre sequencialmente:
    - L1 → L2 → L3 → MP.

## Elementos de projeto da cache

- Fatores Críticos: O desempenho da Cache depende de decisões de projeto:
  a. Mapeamento de Dados MP/Cache: Como determinar onde um bloco da MP será armazenado na Cache.
  b. Algoritmos de Substituição: Qual bloco remover em caso de miss (falha).
  c. Política de Escrita: Como garantir a coerência dos dados entre Cache e MP.

## Mapeamento Direto (Direct Mapping)

- Conceito: Cada bloco da MP possui apenas um local possível (uma linha) na Cache para ser armazenado.
- Simplicidade: É o método mais simples de implementar e de hardware mais rápido (não requer busca por similaridade).
- Fórmula: Linha da Cache =
  - (Endereço do Bloco na MP) *MOD* (Número de Linhas da Cache).

# Mapeamento Direto: Estrutura

- Estrutura: A Cache é dividida em linhas, e cada linha armazena:
  - TAG: Identificador único do bloco da MP.
  - Dados (Bloco): O conteúdo da memória principal.
  - Bit Válido (V): Indica se o bloco contém dados válidos (V=1) ou se a linha está vazia (V=0).

# Mapeamento Direto: Formato do Endereço

- O endereço do bloco na MP é dividido em três campos:
  - TAG (Etiqueta): Usado para comparação. Identifica unicamente o bloco da MP que está na linha.
  - Linha (Índice): Usado para endereçamento. Determina qual linha da Cache deve ser acessada.
  - Word/Byte (Palavra/Byte): Usado para seleção interna do dado dentro do bloco.

# Mapeamento Direto: Vantagens e Desvantagens

- Vantagens:
  - Fácil de implementar.
  - Não necessita de complexos algoritmos de substituição (a substituição é automática).
- Desvantagens:
  - Alta taxa de conflito. Se dois blocos da MP mapeiam para a mesma linha da Cache, eles se substituem constantemente, mesmo que haja espaço livre em outras linhas.

# Mapeamento Direto: Vantagens e Desvantagens

- Exemplo:
  - Cache com $2^4$ = 16 linhas.
- Blocos da MP 0 e 16 mapeiam para a mesma Linha:
  - Bloco 0: 0(mod 16) = 0 (Linha 0)
  - Bloco 16: 16(mod 16) = 0 (Linha 0)
- Se o programa acessa sequencialmente Bloco 0 e Bloco 16, ocorrerá Miss em todos os acessos (o Miss de Conflito).

## Mapeamento Associativo (Fully Associative Mapping)

- **Conceito:** Um bloco da MP pode ser alocado em qualquer linha livre da Cache.
- **Flexibilidade Máxima:** Elimina o Miss de Conflito.
- **Complexidade:** Requer busca em **todas as linhas** da Cache simultaneamente para verificar se o bloco está presente (busca pela **TAG**).

## Mapeamento Associativo (Fully Associative Mapping)

- **Conceito:** Um bloco da MP pode ser alocado em qualquer linha livre da Cache.
- **Flexibilidade Máxima:** Elimina o Miss de Conflito.
- **Complexidade:** Requer busca em **todas as linhas** da Cache simultaneamente para verificar se o bloco está presente (busca pela **TAG**).

# Mapeamento Associativo: Estrutura

- Estrutura:
  - Não há campo de índice de "Linha".
  - O endereço da Cache consiste apenas na **TAG** e no **Dado**.
  - Necessita de hardware comparador para **todas as linhas** ao mesmo tempo.
- Vantagem: A Cache só falha por **capacidade** (Miss de Capacidade) ou **compulsório** (Miss Compulsório).

# Mapeamento Associativo: Formato do Endereço

- O endereço da MP é dividido em apenas dois campos:
  - **TAG (Etiqueta):** Usado para **comparação**. É muito maior que no Mapeamento Direto, pois precisa identificar unicamente o bloco em toda a MP.
  - **Word/Byte (Palavra/Byte):** Usado para **seleção** interna do dado dentro do bloco.

# Mapeamento Associativo: Vantagens e Desvantagens

- **Vantagens:**
  - Maior Taxa de Acerto (H) possível.
  - Flexibilidade total na alocação dos blocos.
- **Desvantagens:**
  - Alto Custo de Hardware: Requer circuitos complexos de comparação paralela e lógica para escolha do bloco a ser substituído.
  - Lógica de substituição mais complexa (e lenta).

## Mapeamento Associativo por Conjuntos

- Conceito: É uma combinação dos dois métodos anteriores para equilibrar velocidade e flexibilidade.
- Organização: A Cache é dividida em Conjuntos (Sets), e cada conjunto contém K linhas.
- Um bloco da MP mapeia para apenas um conjunto, mas pode ser alocado em qualquer das K linhas daquele conjunto.

# Mapeamento por Conjuntos: Estrutura

- Fórmula:
  - Conjunto da Cache = (Endereço do Bloco na MP) *MOD* (Número de Conjuntos).
- Mapeamento Direto é um Associativo por Conjuntos com K=1 (1-way).
- Mapeamento Associativo Completo é um Associativo por Conjuntos com K = Número de Linhas (N-way).

# Mapeamento por Conjuntos: Formato do Endereço

- O endereço da MP é dividido em três campos:
  - **TAG (Etiqueta):** Usado para comparação dentro do conjunto.
  - **Conjunto (Set):** Usado para endereçamento do conjunto na Cache.
  - **Word/Byte (Palavra/Byte):** Usado para seleção interna do dado dentro do bloco.

# Mapeamento Associativo por K-Vias

- K-Vias Associativo: Termo que define a quantidade de linhas em cada conjunto.
  - Exemplos: 2-way, 4-way, 8-way, etc.
- Avanço: O aumento de K (o número de vias) reduz o Miss de Conflito, mas aumenta a complexidade de hardware.
- É o mapeamento mais utilizado nos projetos atuais.

# Comparação dos 3 Tipos de Mapeamento

| Característica | Direto | Associativo | Set-Associative |
|---|---|---|---|
| Localização | 1 linha específica | Qualquer linha | K linhas de 1 conjunto |
| Hardware | Mais Simples | Mais Complexo | Intermediário |
| TAG | Linha + TAG | TAG (Maior) | Conjunto + TAG (Interm.) |
| Flexibilidade | Baixa | Máxima | Intermediária (Ajustável por K) |

# Algoritmos de Substituição

- Tópico: Qual bloco da Cache deve ser removido (evicted) em caso de Miss?
- Contexto: Necessário para mapeamentos Associativo e Set-Associative (o Direto é automático).
- O algoritmo escolhido afeta diretamente a Taxa de Acerto.
- O objetivo é remover o bloco que tem a menor probabilidade de ser acessado novamente em breve.

# Políticas de Substituição Comuns

- **LRU (Least Recently Used)**: Substitui o bloco que foi utilizado há mais tempo.
  - Melhor desempenho, mas mais complexo de implementar (rastreamento de tempo).
- **FIFO (First-In, First-Out)**: Substitui o bloco que entrou primeiro na Cache.
  - Simples, mas não reflete bem o princípio da localidade.
- **LFU (Least Frequently Used)**: Substitui o bloco que foi usado menos vezes.
  - Complexo de implementar (requer contador de uso).

# Políticas de Substituição Comuns

- **LRU (Least Recently Used)**: Substitui o bloco que foi utilizado há mais tempo.
  - Melhor desempenho, mas mais complexo de implementar (rastreamento de tempo).
- **FIFO (First-In, First-Out)**: Substitui o bloco que entrou primeiro na Cache.
  - Simples, mas não reflete bem o princípio da localidade.
- **LFU (Least Frequently Used)**: Substitui o bloco que foi usado menos vezes.
  - Complexo de implementar (requer contador de uso).

# Políticas de Escrita

- Como garantir a Coerência dos dados entre Cache e MP quando a UCP modifica um dado na Cache?
- A escrita pela Cache deve ser tratada com cuidado para que a MP e outros dispositivos de E/S tenham sempre o valor mais atualizado.
- Dois Tipos Principais: Write-Through e Write-Back.

# Política de Escrita: Write-Through

- Conceito: Toda escrita na Cache é imediatamente propagada para a Memória Principal.
- Vantagens:
  - MP está sempre coerente (atualizada).
  - Miss de leitura simplificado.
- Desvantagens:
  - Gera tráfego constante no barramento do sistema.
  - Maior tempo de escrita, pois é limitado pela velocidade da MP.

# Política de Escrita: Write-Back

- Conceito: A escrita ocorre apenas na Cache. O bloco modificado só é escrito na MP quando for removido (evicted).
- Bit Sujo (Dirty Bit): Um bit de controle indica se o bloco na Cache foi modificado (sujo) e precisa ser escrito na MP.
- Vantagens:
  - Minimiza o tráfego no barramento (apenas blocos sujos são escritos).
  - Maior velocidade de escrita (na velocidade da Cache).
- Desvantagens: Maior complexidade e MP pode ficar incoerente por um tempo.

# Otimização e Projeto Final

- Tamanho da Cache: Quanto maior, melhor a Taxa de Acerto, mas maior o Tempo de Acesso e o Custo. É um equilíbrio.
- Largura da Linha (Tamanho do Bloco): Otimiza a Localidade Espacial.
  - Linhas muito curtas: desperdiça banda, mais Misses de Localidade Espacial.
  - Linhas muito longas: pode carregar dados desnecessários (poluição), substituindo blocos úteis e aumentando a penalidade do Miss.
- Conclusão: O projeto da Cache é uma constante busca por equilíbrio entre velocidade, capacidade e custo para maximizar o desempenho geral do sistema.