INE5607 – Organização e Arquitetura de Computadores

Hierarquia e Gerência de Memória

Aula 23: Princípios de localidade e hierarquia de memória

Prof. Laércio Lima Pilla

laercio.pilla@ufsc.br

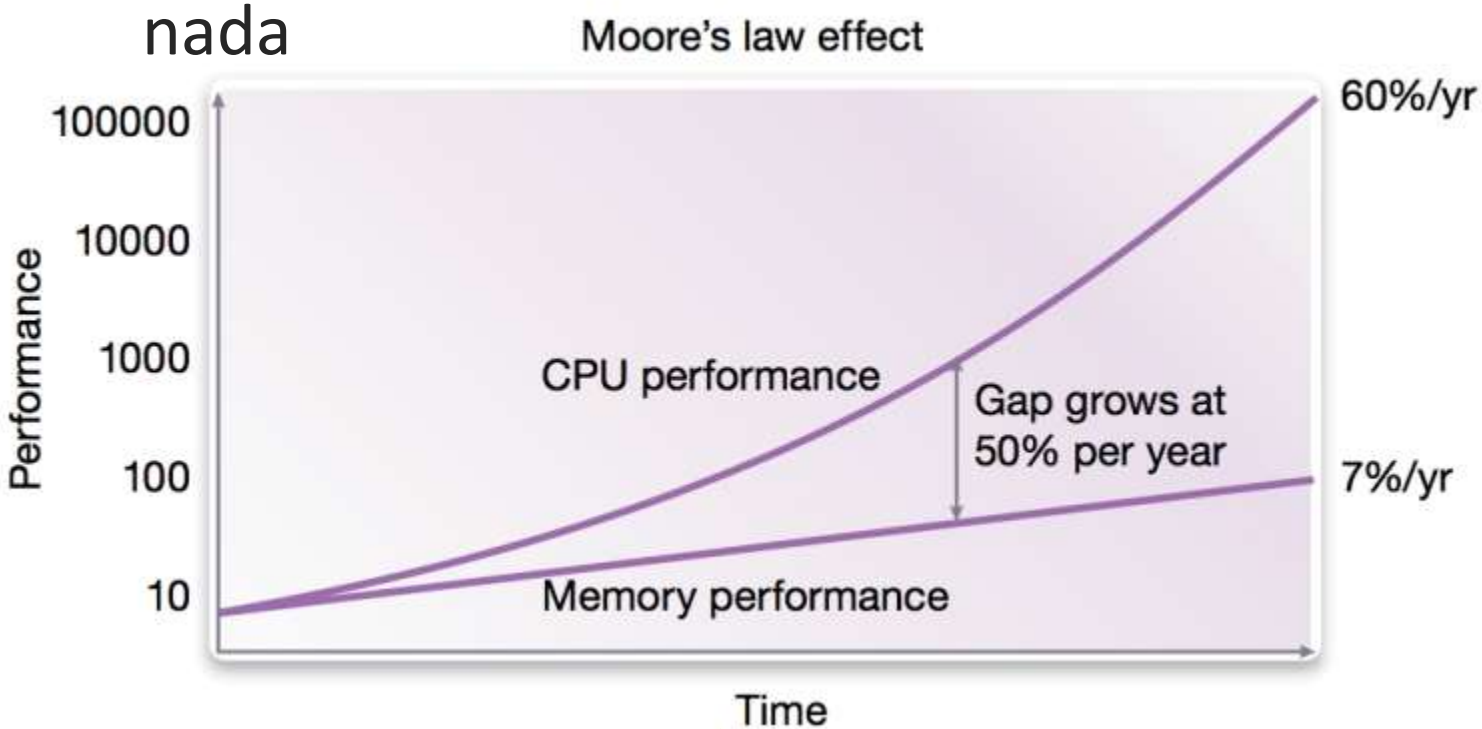
Sumário

- *Memory Wall*
- Hierarquia de memória
- Princípio da localidade
- Funcionamento
- Desempenho
- Considerações finais

MEMORY WALL

Memory Wall

- **Disparidade de desempenho** entre processadores e memória
  - **Diferença crescente**
    - Ponto onde melhorias na CPU não adiantariam de nada



Memory Wall

- Exemplo
  - Processador de 2 GHz executa uma instrução por ciclo
    - 0,5 ns /ciclo
  - 20% das instruções acessam a memória
  - Cada acesso à memória leva 100 ciclos
  - Quanto tempo cada instrução deveria levar?
    - Resposta: 0,5 ns
  - Qual o tempo médio das instruções?
    - Resposta: 0,5 ns + 0,2*50 ns = 10,5 ns (~95MHz)

Memory Wall

- Como amenizar o efeito da *Memory Wall*?
  - E se nós tivéssemos uma **memória intermediária** que **reduzisse o tempo de acesso**?
  - Exemplo: se 95% dos acessos levassem 5 ciclos na memória intermediária (os outros 5% precisam de mais 100 ciclos)

    = 0,5 + 0,2*(2,5 + 0,05*50)

    = 0,5 + 0,2*(2,5 + 2,5)

    = 0,5 + 1 = 1,5 ns

    666 MHz ou ~7x melhor do que antes

Memory Wall

- Como amenizar o efeito da *Memory Wall*?
  - E se, ao invés de uma memória intermediária, nós tivéssemos **várias memórias**?
    - Uma memória depois da outra
      - Primeiro tenta na mais rápida
      - Depois na segunda mais rápida
      - Depois na terceira mais rápida ...
    - Algo como uma **hierarquia de memórias** \o/

HIERARQUIA DE MEMÓRIA

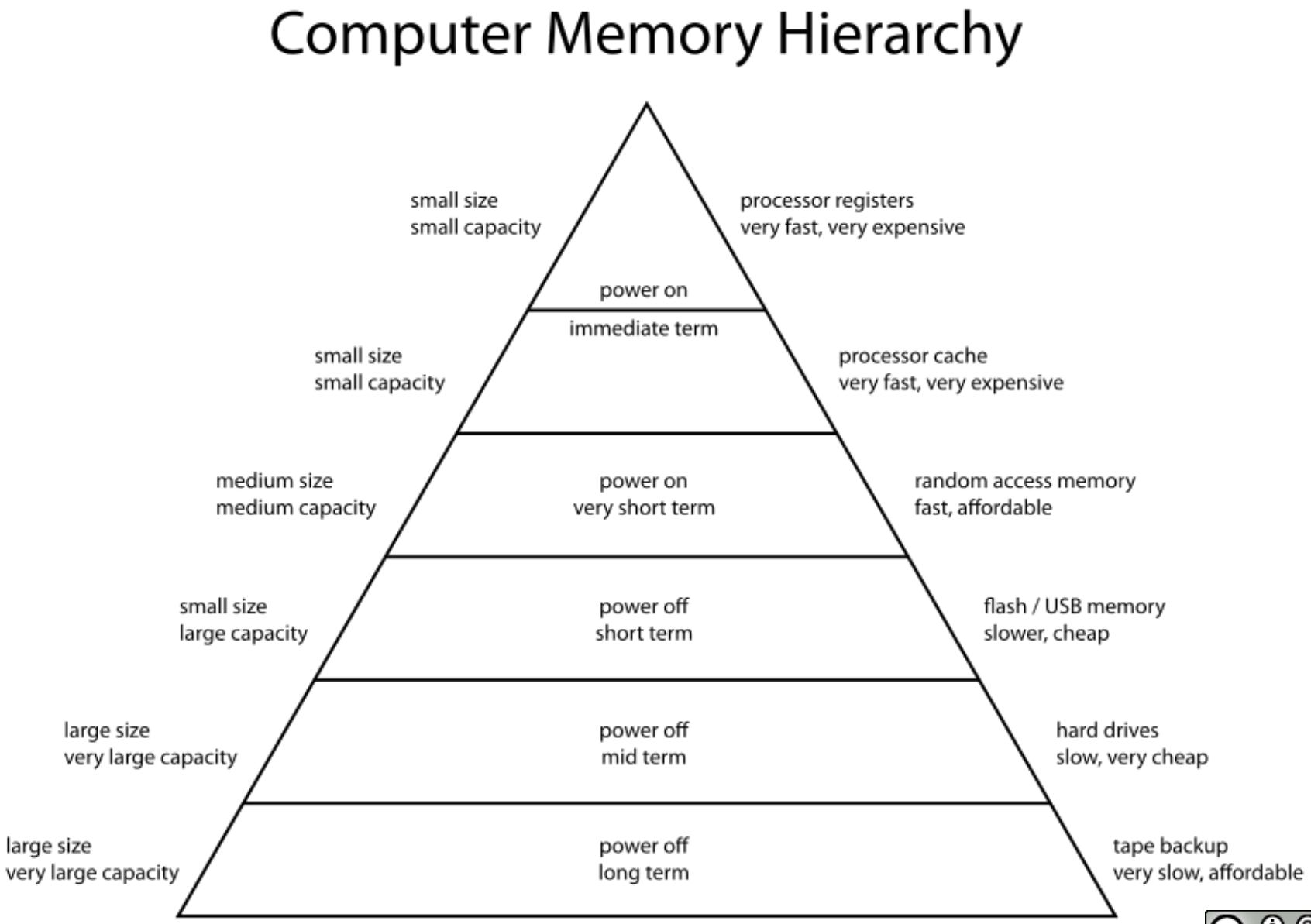
Hierarquia de memória

- Estrutura que usa múltiplos níveis de memória
  - Quanto **mais longe da CPU**,
    - **Maior** a memória,
    - **Mais lenta** a memória e
    - **Mais barata** a memória

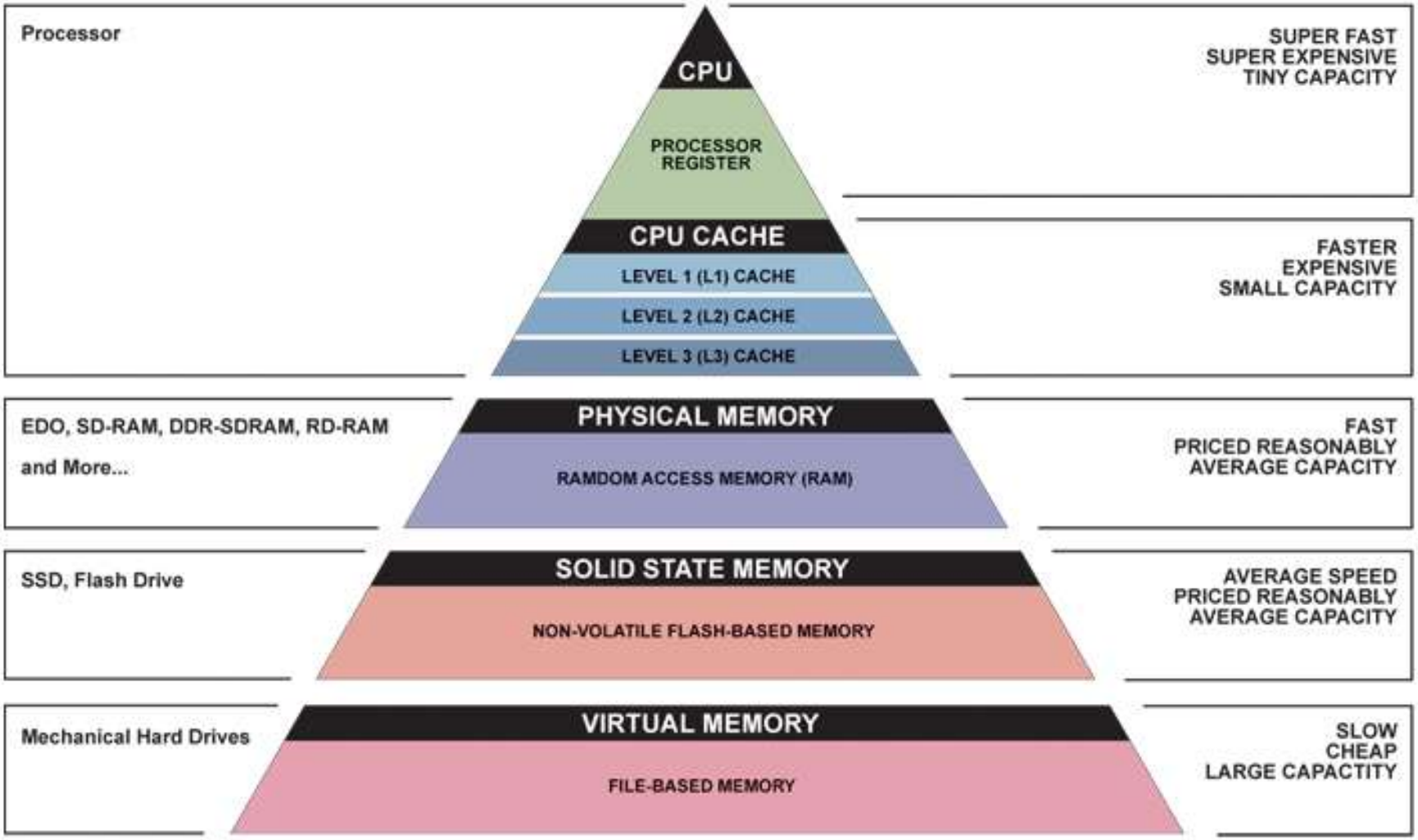
Hierarquia de memória

- Exemplos
  - **Caches**
    - SRAM: *Static Random Access Memory*
    - Dois ou três níveis internos ao processador, MBs de capacidade
  - **Memória principal**
    - DRAM: *Dynamic Random Access Memory*
    - Alguns pentes de memória com GBs de capacidade
  - **Memória secundária**
    - Disco rígido com TBs de capacidade

Hierarquia de memória



Hierarquia de memória



▲ Simplified Computer Memory Hierarchy
Illustration: Ryan J. Leng

Hierarquia de memória

- Outra visão
  - *"O ideal seria ter uma capacidade de memória infinitamente grande a ponto de qualquer word específica [...] estar imediatamente disponível. [...] Somos [...] forçados a reconhecer a possibilidade de construir uma hierarquia de memórias, cada uma com uma capacidade maior do que a anterior, mas com acessibilidade menos rápida."*
    - A. W. Burks, H. H. Goldstine e J. von Neumann, 1946

Hierarquia de memória

- Tempo de acesso à memória
  - O que acontece se os dados estão sempre na memória principal e nunca na cache?
  - Qual seria o **caso ideal**?
    - O que nos indica que podemos chegar perto do ideal?

PRINCÍPIO DA LOCALIDADE

Princípio da localidade

- Analogia da biblioteca

Princípio da localidade

- Analogia da biblioteca
  - Livros na estante
    - Muitos livros, acesso lento
  - Livros na mesa
    - Poucos livros, acesso rápido
  - Livro na mão
    - Um livro, acesso muito rápido

Princípio da localidade

- Analogia da noite de filmes (para quem não gosta de estudar)
  - Comida na geladeira
    - Muita comida, acesso lento
  - Comida na mesa
    - Pouca comida, acesso rápido
  - Comida na mão
    - Nom nom nom nom

Princípio da localidade

- **Princípio da localidade**
  - Visão de programas
    - **Programas acessam uma parte relativamente pequena de seu espaço de endereçamento em um dado instante de tempo**
  - Visão de estudantes
    - Estudantes usam uma parte relativamente pequena da biblioteca em um dado instante de tempo

Princípio da localidade

- **Dois tipos de localidade**
  - **Localidade temporal**
    - Se um endereço é referenciado, é provável que ele seja referenciado em breve
  - **Localidade espacial**
    - Se um endereço é referenciado, é provável que endereços próximos sejam referenciados em breve

Princípio da localidade

- **Exemplos**
  - **Alta localidade temporal**
    - Instruções em loop
  - **Baixa localidade temporal**
    - *Loop unrolling*
  - **Alta localidade espacial**
    - Procedimento sem testes (branches)
  - **Baixa localidade espacial**
    - Estruturas indexadas

Princípio da localidade

- Exemplos
  - **Alta localidade temporal** e **baixa localidade espacial**
    - Acesso aleatório em um vetor enquanto se reutiliza cada dado várias vezes
  - **Baixa localidade temporal** e **alta localidade espacial**
    - Leitura de vetor em ordem com acesso a cada dado apenas uma vez

FUNCIONAMENTO

Funcionamento

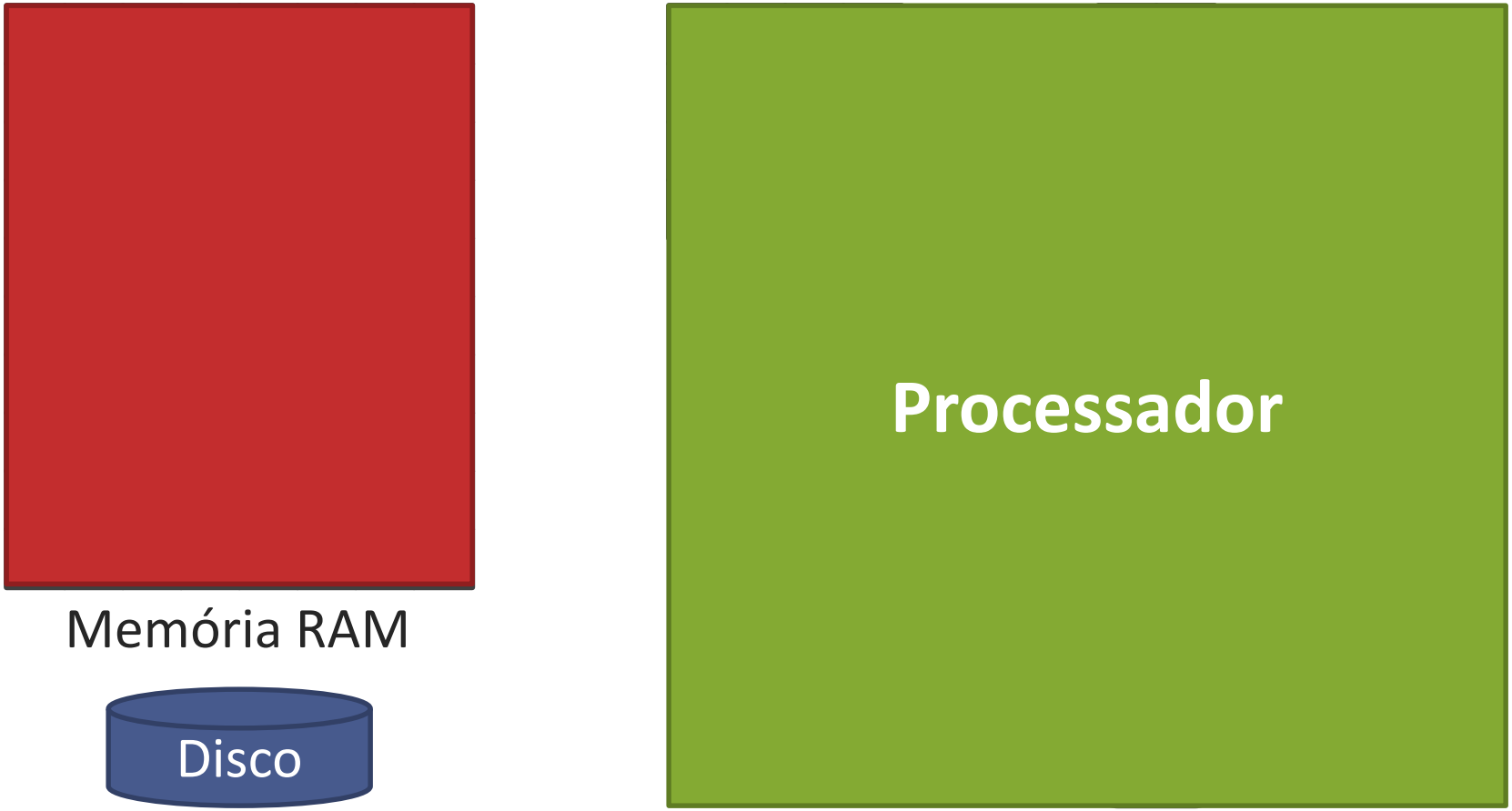
- Acessos à cache
  - **Acerto (*cache hit*)**
    - Endereço acessado se encontra na cache
    - Tudo ok
  - **Falha (*cache miss*)**
    - Endereço acessado não se encontra na cache
    - Pede para o próximo nível na hierarquia pelo dado

Funcionamento

- Aula 2

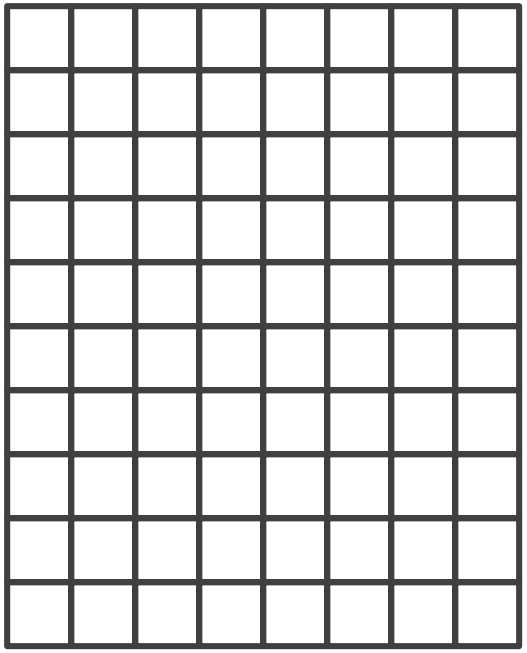
Funcionamento

- Plataforma hipotética

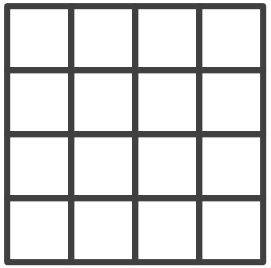


Funcionamento

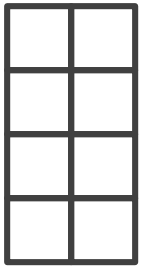
- Plataforma hipotética



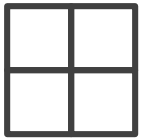
Memória RAM



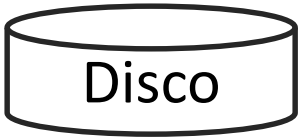
Cache L3



Cache L2



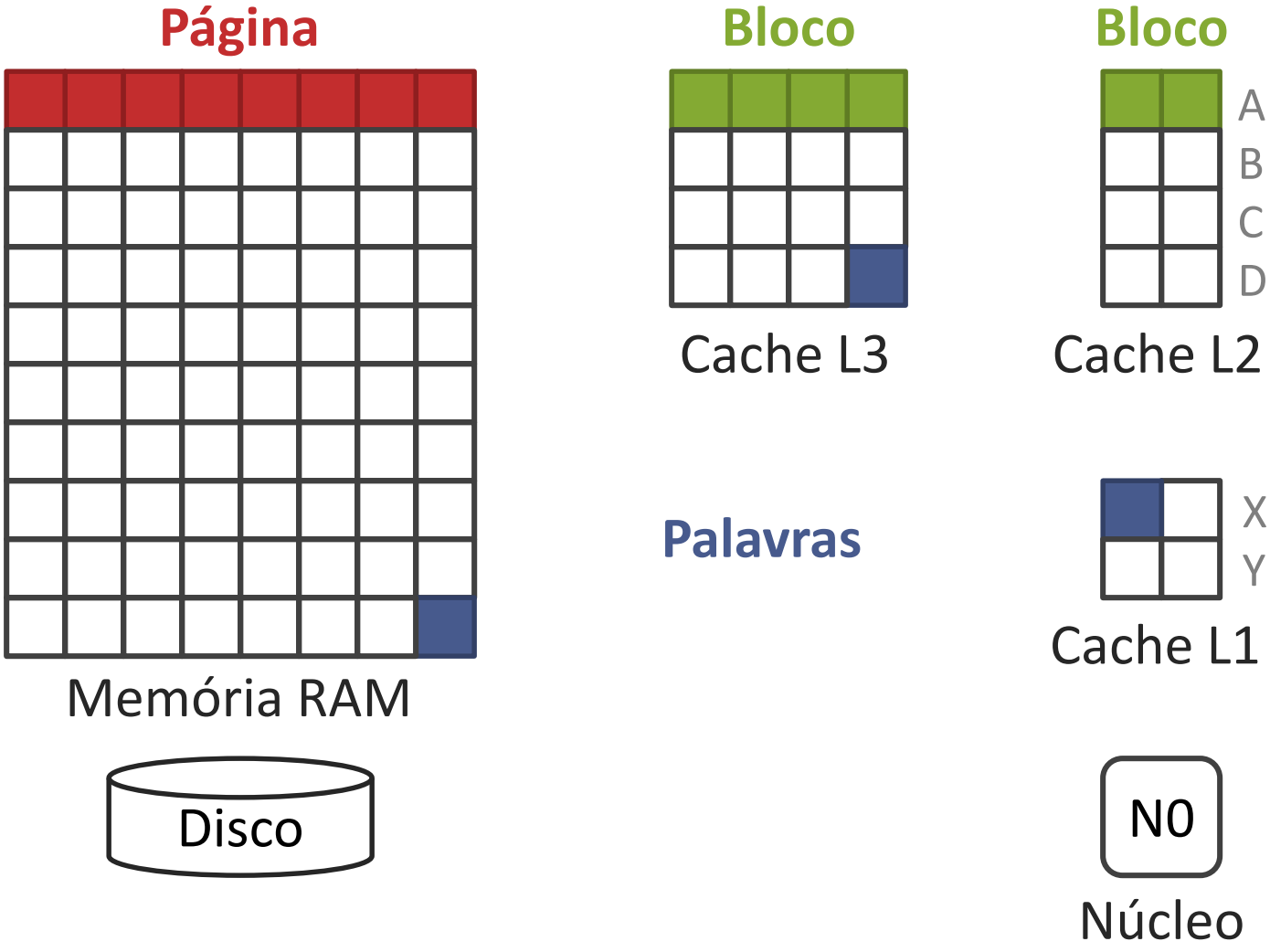
Cache L1



Núcleo

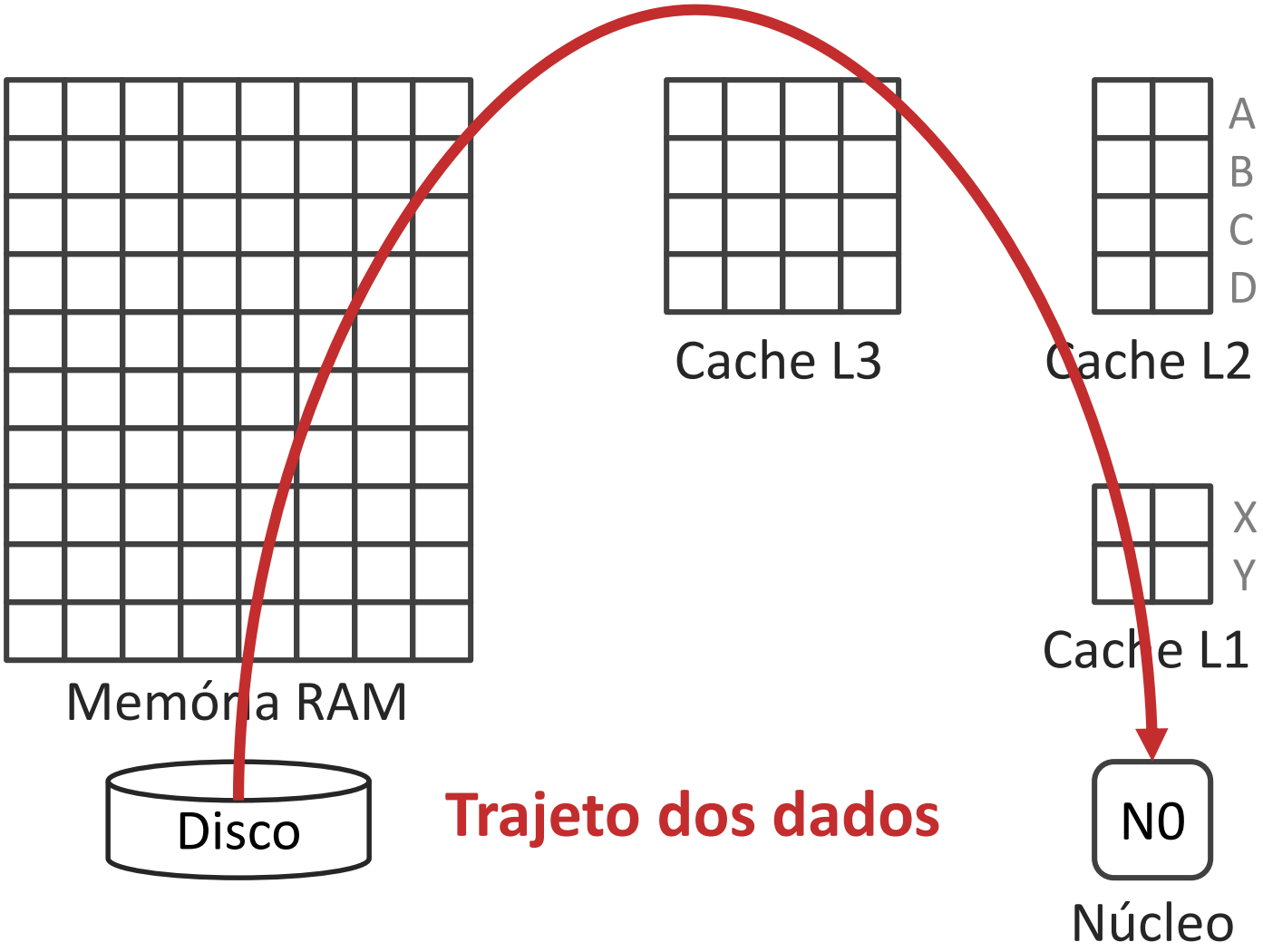
Exemplo de uso

- Plataforma hipotética



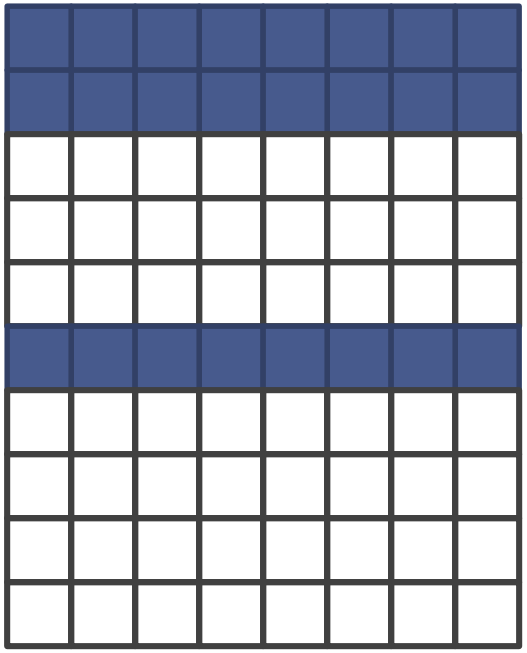
Funcionamento

- Plataforma hipotética

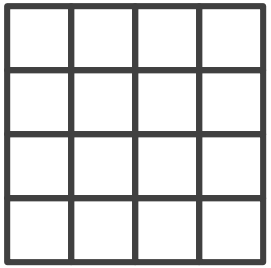


Funcionamento

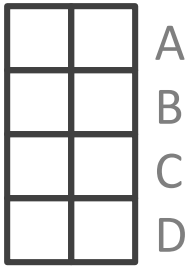
- Carregamento de um programa para a memória



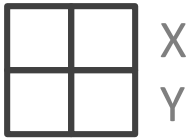
Memória RAM



Cache L3



Cache L2

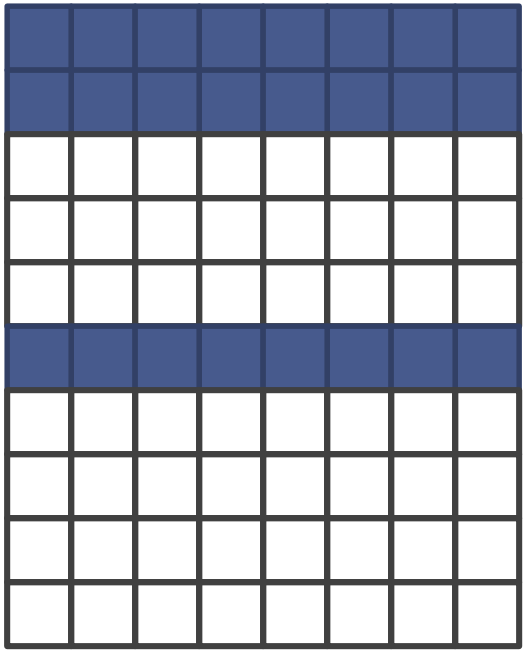


Cache L1

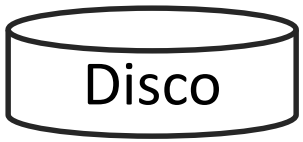
Núcleo

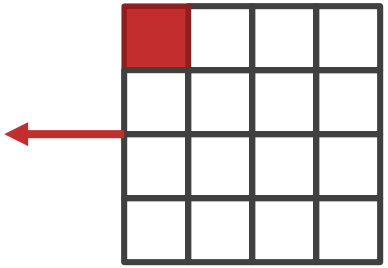
Funcionamento

- Início da execução do programa
  - Cache miss!

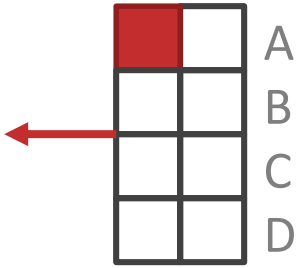


Memória RAM



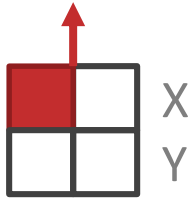


Cache L3

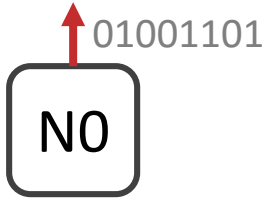


Cache L2

Dados não encontrados nas caches no início do programa, aka falha compulsória



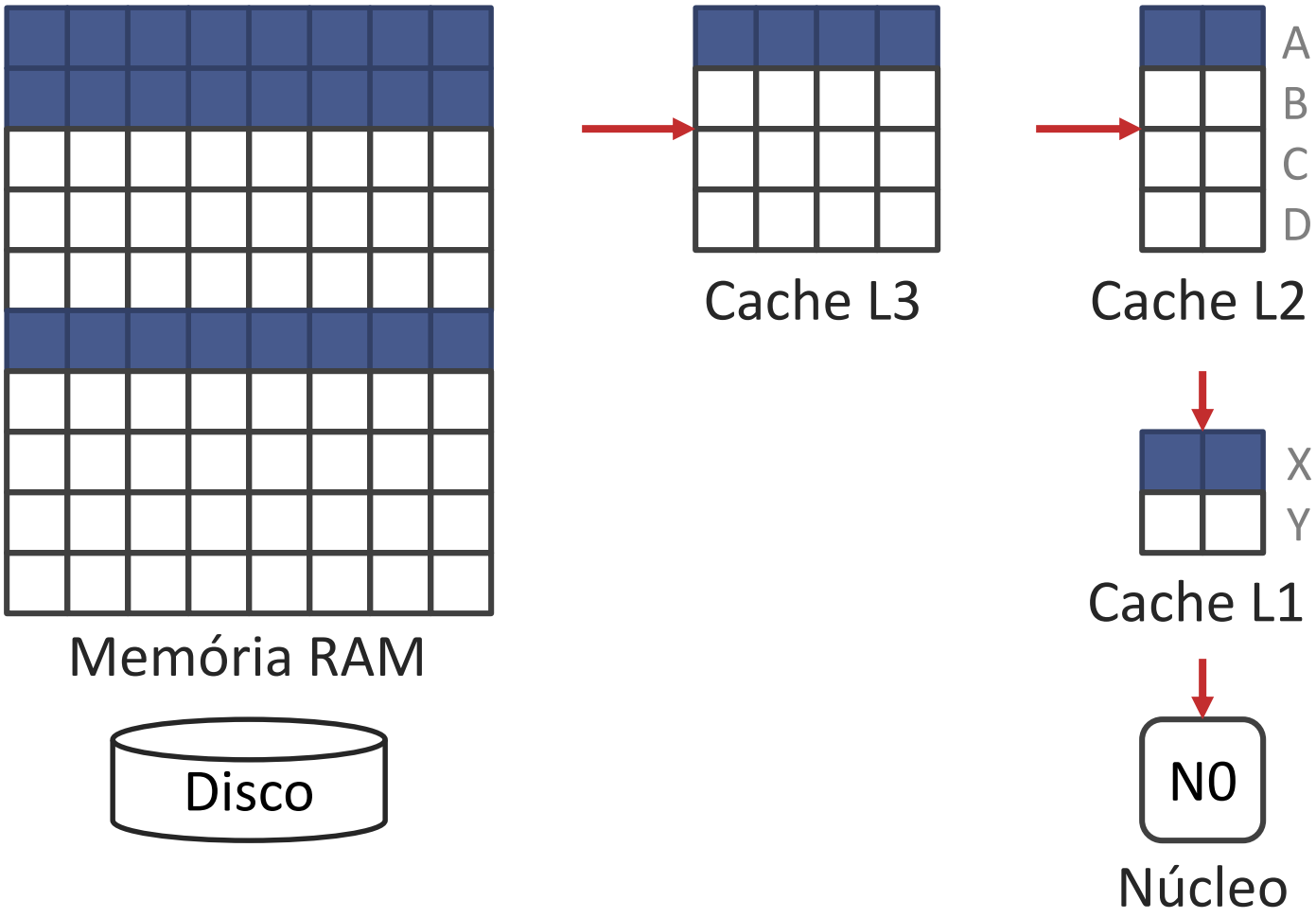
Cache L1



Núcleo

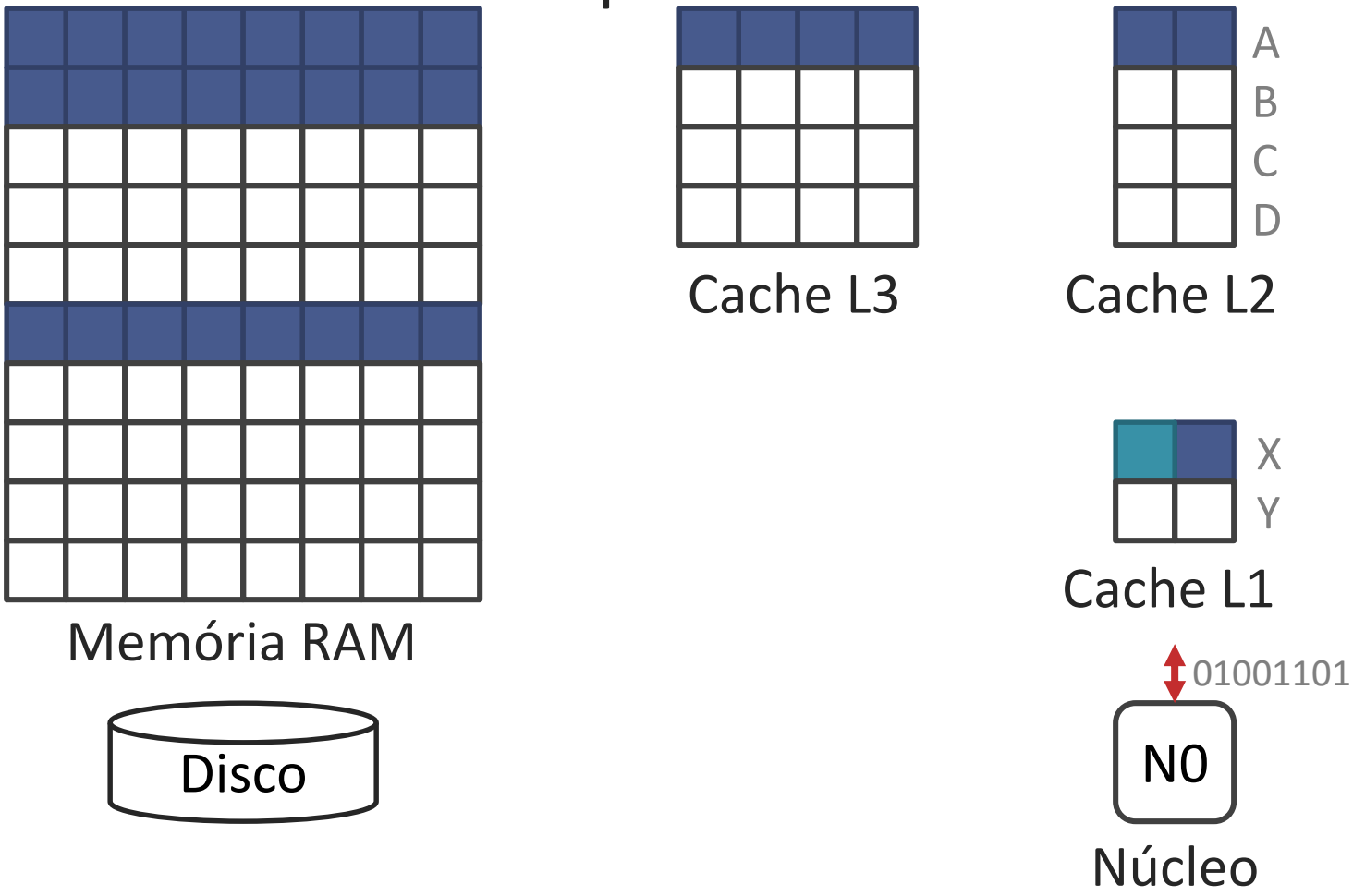
Funcionamento

- Início da execução do programa



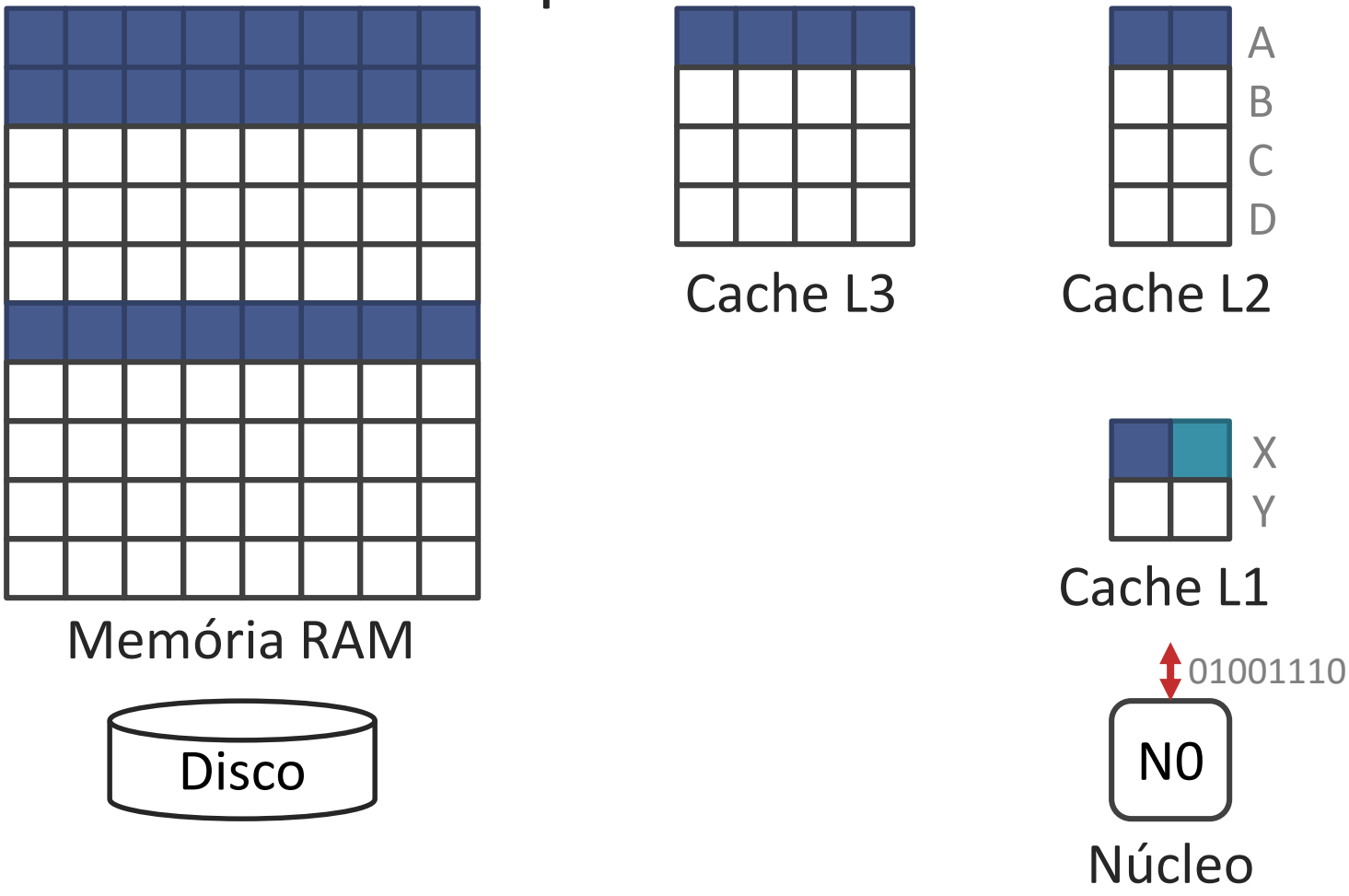
Funcionamento

- Acesso ao dado novamente: Cache hit
  - Localidade temporal



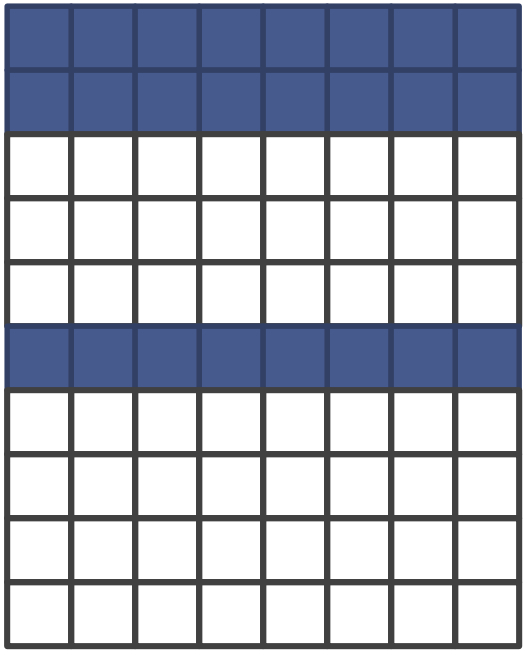
Funcionamento

- Acesso ao próximo dado: Cache hit
  - Localidade espacial



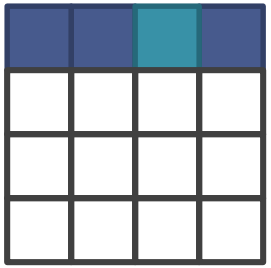
Funcionamento

- Acesso ao dado seguinte: Cache miss na L1 e L2

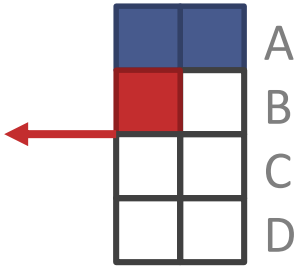


Memória RAM

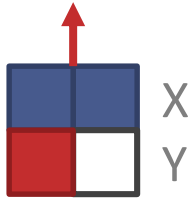
Disco



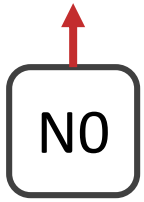
Cache L3



Cache L2



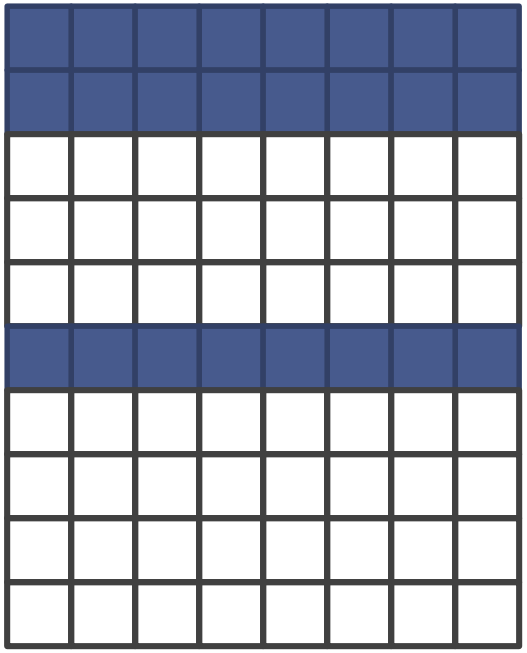
Cache L1



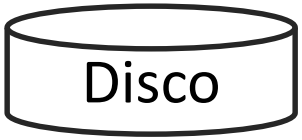
Núcleo

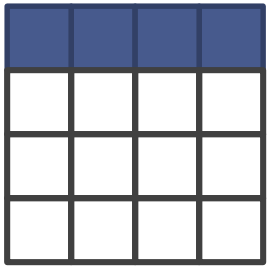
Funcionamento

- Acesso ao dado seguinte: Cache miss na L1 e L2

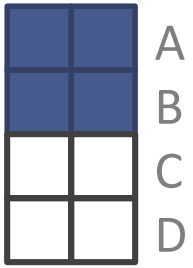


Memória RAM

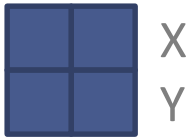




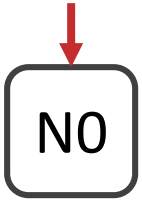
Cache L3



Cache L2



Cache L1



Núcleo

DESEMPENHO

Desempenho

- Desempenho depende de vários parâmetros
  - **Taxa de acerto** (*Hit rate*)
    - Percentual de acesso à cache que resultam em acertos
  - **Tempo de acerto** (*Hit time*)
    - Número de ciclos para acessar a cache

Desempenho

- Desempenho depende de vários parâmetros
  - **Taxa de falta** (*Miss rate*)
    - Miss rate = 1 – hit rate
  - **Penalidade de falta** (*Miss penalty*)
    - Número de ciclos para acessar o próximo nível na hierarquia
      - L1 p/ L2, L2 p/ L3, L3 p/ RAM, RAM p/ disco

Desempenho

- Número de ciclos médio para acesso à memória
  - AMAT: *Average Memory Access Time*
  - **#Ciclos = tempo de acerto + (taxa de falta * penalidade de falta)**

Desempenho

- Exemplo
  - Taxa de acerto (hit rate): **80%**
  - Tempo de acerto (hit time): **5 ciclos**
  - Penalidade de falta (miss penalty): **20 ciclos**

  C_m = **5** + **0,2*20** = 5 + 4 = **9 ciclos**

  - Taxa de acerto (hit rate): **95%**

  C_m = **5** + **0,05*20** = 5+1 = **6 ciclos**

Desempenho

- **Cálculos podem ser encadeados**
  - Cache L1
    - Miss rate 50%, Hit time 2 ciclos
  - Cache L2
    - Miss rate 10%, Hit time 10 ciclos, Miss penalty 100
  - Ciclos médios

$C_m = ht_{L1} + (1-hr_{L1})*C_{L2}$

$C_{L2} = ht_{L2} + (1-hr_{L2})*mp_{L2}$

$C_m = 2 + 0,5*(10 + 0,1*100)$

$C_m = 2 + 0,5*(10 + 10) = 2 + 0,5*20 = 11$

Desempenho

- **Como o acesso à memória afeta o desempenho**

  Tempo de CPU = (Ciclos de CPU + Ciclos em stalls de memória)*Tempo do ciclo de relógio

  - Em geral, assume-se que os ciclos de CPU já contam com o uso de uma cache de mais alto nível perfeita
    - Sem falhas -> Sem penalidades por falhas
    - Ciclos em stalls de memória precisam ser contados com as faltas nas caches

Desempenho

- **Como o acesso à memória afeta o desempenho**

  Tempo de CPU = (Ciclos de CPU + Ciclos em stalls de memória)*Tempo do ciclo de relógio

  Ciclos em stalls de memória = Acessos à memória por programa* taxa de faltas * penalidade por falta

  ou

  Ciclos em stalls de memória = Instruções por programa* faltas por instruções* penalidade por falta

Desempenho

- **Exemplo 1**
  - Um programa leva a 5% de falhas na cache de instruções e 10% de falhas na cache de dados. Sabendo que 20% de suas instruções são load/store, como o CPI do processador abaixo é afetado pela carga de trabalho do programa?
    - L1 instruções ou dados: tempo de acesso de 5 ciclos, penalidade de falha de 50 ciclos
    - CPI do processador sem stalls de memória: 3

Desempenho

- **Exemplo 1**
  - CPI do processador sem stalls de memória: 3
  - L1 instruções
    - Usada por 100% das instruções
    - Hit rate 95%, hit time 5, miss penalty 50
  - L1 dados
    - Usada por 20% das instruções
    - Hit rate 90%, hit time 5, miss penalty 50
  - $CPI_{final} = CPI_{sem\ stalls} + CPI_{inst} + CPI_{dados}$

Desempenho

- **Exemplo 1**

$CPI_{final} = CPI_{sem\ stalls} + CPI_{inst} + CPI_{dados}$

$CPI_{inst} = 1*(5 + 0,05*50) = 1*(5 + 2,5) = 7,5$

$CPI_{dados} = 0,2*(5 + 0,1*50) = 0,2*(5+5) = 0,2*10 = 2$

$CPI_{final} = 3 + 7,5 + 2 = 12,5$

Desempenho

- **Exemplo 2** (do livro-texto)
  - Cache de instruções com taxa de faltas de 2%
  - Taxa de faltas para dados de 4%
  - Frequência de acesso a dados de 36%
  - Penalidade de falta de 100 ciclos
  - CPI = 2 com sem considerar memória
  - O quão mais rápido seria o processador se não houvessem faltas de cache?

Desempenho

- **Exemplo 2**
  - Ciclos em faltas de instruções
    - I*0,02*100 = 2*I
  - Ciclos em faltas de dados
    - I*0,36*0,04*100 = 1,44*I
  - Total de ciclos em stalls de memória
    - 2*I + 1,44*I = 3,44*I
  - CPI com stalls = 3,44 + 2 = 5,44
  - CPI sem stalls = 2

Desempenho

- **Exemplo 2**
  - Diferença de desempenho
    - 5,44 / 2 = 2,72
    - O processador seria 2,72 mais rápido sem faltas de cache

Desempenho

- **Exemplo 3**
  - Dado o sistema com dois níveis de cache com as características listadas abaixo, quantos ciclos adicionais levam loads e stores em média?
    - L1: miss rate 25%, hit time 2 ciclos
    - L2: miss rate 10%, hit time 10 ciclos
    - Memória: hit time 200 ciclos

    $C_m = 2 + 0{,}25*(10 + 0{,}1*200)$

    $C_m = 2 + 0{,}25*30 = 2 + 7{,}5 = 9{,}5$ ciclos

CONSIDERAÇÕES FINAIS

Considerações finais

- *Memory Wall*
  - Disparidade de desempenho entre CPU e memória
- Hierarquia de memória
  - Como funciona
  - Como medir desempenho
- Próximo passo
  - Como mapear dados da memória para cache

INE5607 – Organização e Arquitetura de Computadores

Hierarquia e Gerência de Memória

Aula 23: Princípios de localidade e hierarquia de memória

Prof. Laércio Lima Pilla

laercio.pilla@ufsc.br