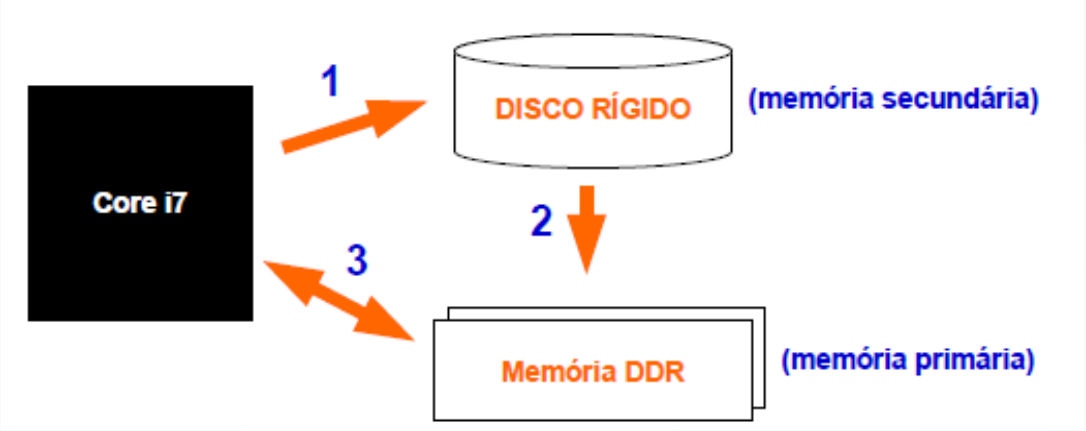
Memórias Cache
Parte II

Tópicos da aula

- **Múltiplas memórias Cache (Cache Multinível)**
- **Paralelismo e hierarquias de memória: coerência de cache**

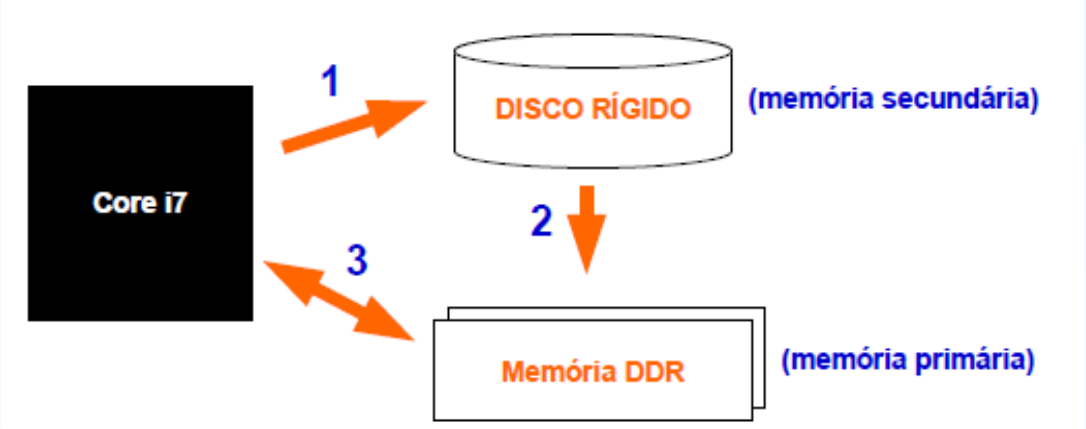
Sistema hierárquico de memórias

- Relembrando: historicamente os sistemas computacionais sempre trabalharam com uma **memória secundária** para programas (cartões perfurados, fitas magnéticas, HD, etc) e uma **memória primária** (memórias voláteis, como SDR, DDR, etc)



Sistema hierárquico de memórias

- Relembrando: historicamente os sistemas computacionais sempre trabalharam com uma **memória secundária** para programas (cartões perfurados, fitas magnéticas, HD, etc) e uma **memória primária** (memórias voláteis, como SDR, DDR, etc)
- Programas são transferidos das memórias **não voláteis** (mais lentas) para serem executados em **memórias voláteis** (mais rápidas)



Por quê?

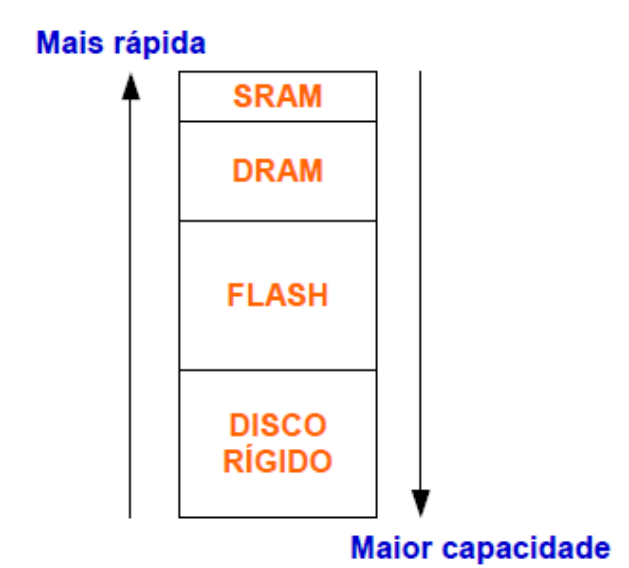
Porque...

- O objetivo (sonho) sempre foi:

memória infinita + memória muito rápida

Evolução das tecnologias

- A velocidade dos processadores **aumentou** significativamente nas últimas décadas, porém **o mesmo não aconteceu** com as memórias dinâmicas
- A memória cache é uma alternativa para melhorar o desempenho geral das máquinas



Impacto no aumento da frequência

- O que acontece se o processador opera numa frequência muito maior que a memória?

Impacto no aumento da frequência

- O que acontece se o processador opera numa frequência muito maior que a memória?
- Ocorrência de uma parada (***stall***) no processador...

Impacto no aumento da frequência

- O que acontece se o processador opera numa frequência muito maior que a memória?
- Ocorrência de uma parada (***stall***) no processador...
- Dobrar a frequência do processador não implica necessariamente em dobrar o desempenho. Por quê?

Impacto no aumento da frequência

- O que acontece se o processador opera numa frequência muito maior que a memória?
- Ocorrência de uma parada (***stall***) no processador...
- Dobrar a frequência do processador não implica necessariamente em dobrar o desempenho. Por quê?
  - Faltas na cache
  - Há uma relação de compromisso entre pipeline e cache para melhorar o desempenho

Impacto no aumento da frequência

- O que acontece se o processador opera numa frequência muito maior que a memória?
- Ocorrência de uma parada (***stall***) no processador...
- Dobrar a frequência do processador não implica necessariamente em dobrar o desempenho. Por quê?
  - Faltas na cache
  - Há uma relação de compromisso entre pipeline e cache para melhorar o desempenho
- Como minimizar esse problema?

Impacto no aumento da frequência

- O que acontece se o processador opera numa frequência muito maior que a memória?
- Ocorrência de uma parada (***stall***) no processador...
- Dobrar a frequência do processador não implica necessariamente em dobrar o desempenho. Por quê?
  - Faltas na cache
  - Há uma relação de compromisso entre pipeline e cache para melhorar o desempenho
- Como minimizar esse problema?
  - **Aumentar os níveis de cache**

Procedimentos durante uma falta

1) Comandar leitura da Memória Principal
2) Esperar que acesso se complete
   - Requer múltiplos ciclos
3) Atualizar a cache
   - O **bloco** é buscado na Memória Principal (campo de dados)
   - MSBs do endereço são usados como **tag**
   - **Bit de validade** é ativado
4) O acesso à cache é **reiniciado**

Manipulando faltas na Cache

- Há uma **Unidade de Controle** que monitora entrada - ***hit***
  - Detecta falta ou acerto
- No caso de ocorrência de **Acerto**
  - Continua execução normal
- No caso de ocorrência de **Falta**
  - Pausa da CPU (***stall on miss***)
    - Unidade de controle da CPU **congela** registradores
    - Controlador dedicado copia item requisitado p/ cache

Diferenças no *Stall*

- Pausa devido a **falta na cache**
  - Toda a CPU sofre pausa
    - Neste caso, o conteúdo dos registradores é **congelado**
- Pausa devido a ***hazard***
  - Nem toda a CPU sofre pausa
    - Algumas instruções continuam execução para resolver o *hazard*

Métricas de Desempenho

- Taxa de sucesso ou de acertos (***hit rate***)
- Taxa de fracasso ou de faltas (***miss rate***)
- Consequência: **h = 1 - m**
- Tempo de acerto (***hit time***)
  - tempo p/ acessar nível superior
- Penalidade de falta (***miss penalty***)
  - tempo para substituir bloco no nível superior
  - tempo de acesso ao nível inferior é dominante

Melhoria de Desempenho

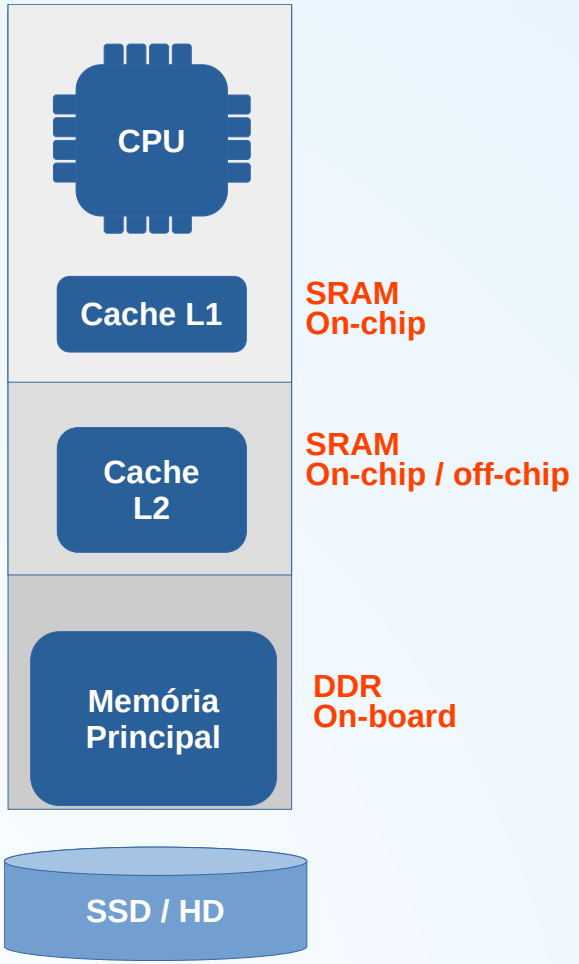
- Ciclos gastos com paradas devidas a faltas

$$\frac{\text{acessos}}{\text{programa}} \times \text{mr} \times \text{penalidade}$$

- Como reduzir a **taxa de faltas** (**mr**)?
  - Posicionamento mais flexível via associatividade
    - Mapeamento direto
    - Memória associativa por conjunto
    - Memória totalmente associativa
- Redução da penalidade
  - **Múltiplos níveis** de cache

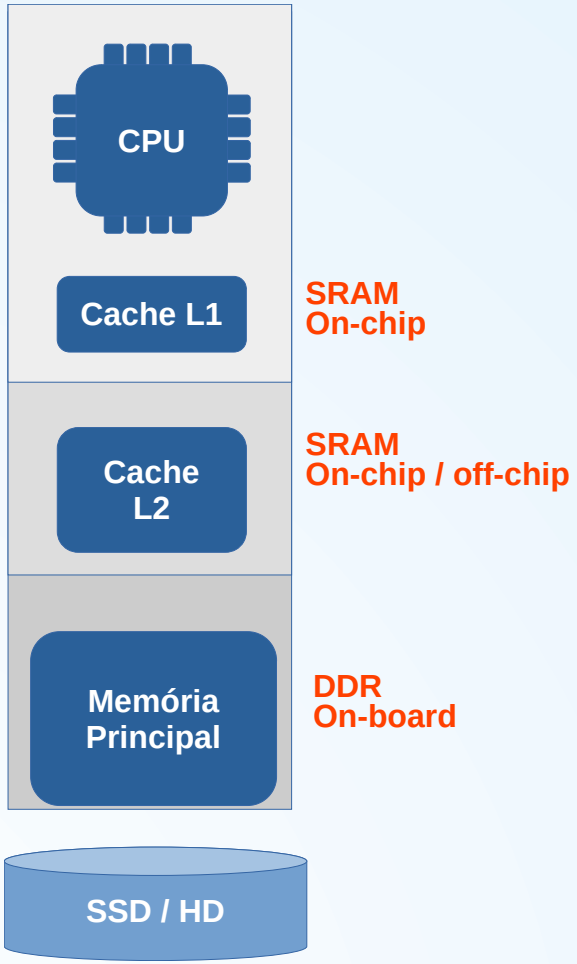
Múltiplas memórias Cache

- Os níveis com **números mais baixos** correspondem a **caches pequenas**



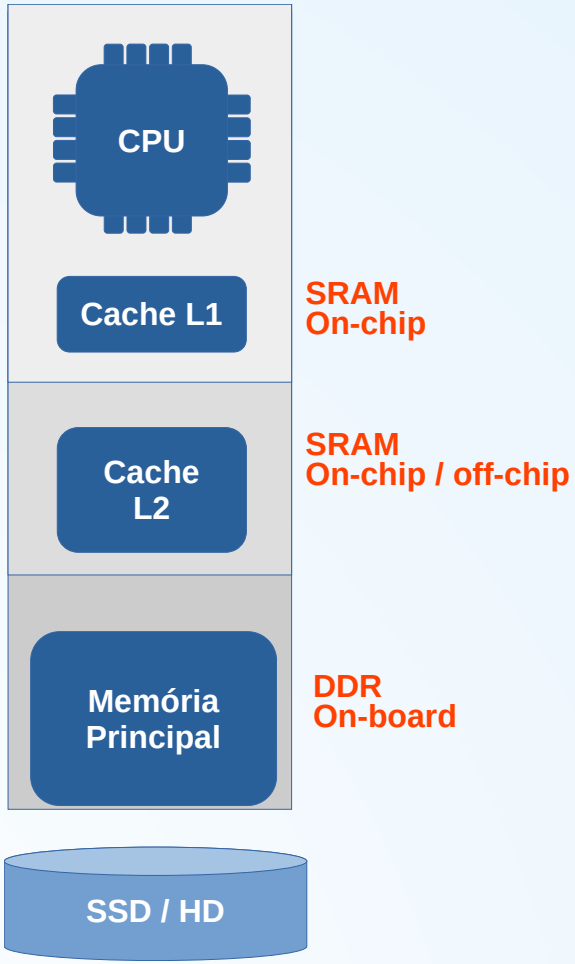
Múltiplas memórias Cache

- Os níveis com **números mais baixos** correspondem a **caches pequenas**
- Quanto **mais alto o número**, tende a ser **mais lenta a cache**, mas também é **maior sua capacidade de armazenamento**



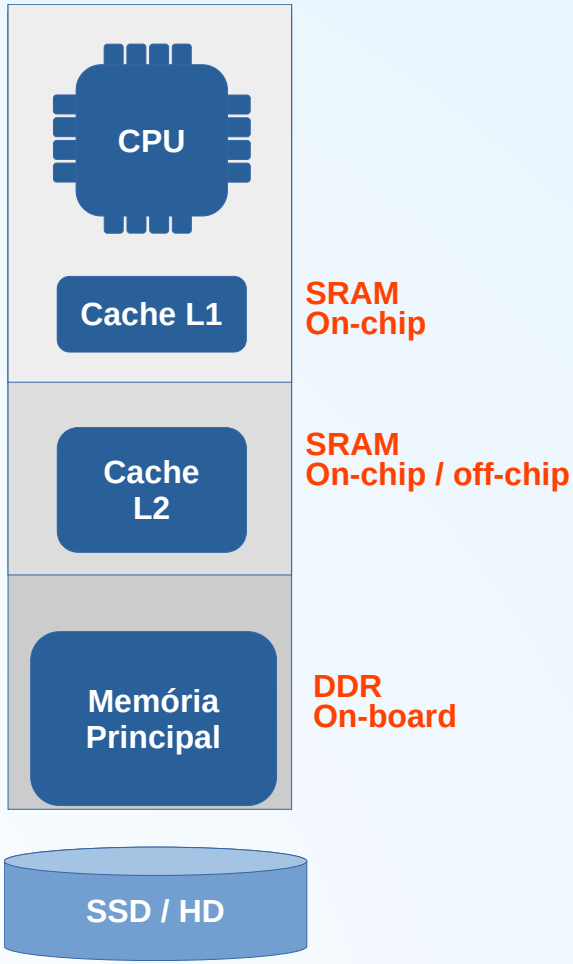
Múltiplas memórias Cache

- Os níveis com **números mais baixos** correspondem a **caches pequenas**
- Quanto **mais alto o número**, tende a ser **mais lenta a cache**, mas também é **maior sua capacidade de armazenamento**
- Apesar das diferentes velocidade de acesso, as memórias cache **ainda são muito mais rápidas** que a memória **principal**



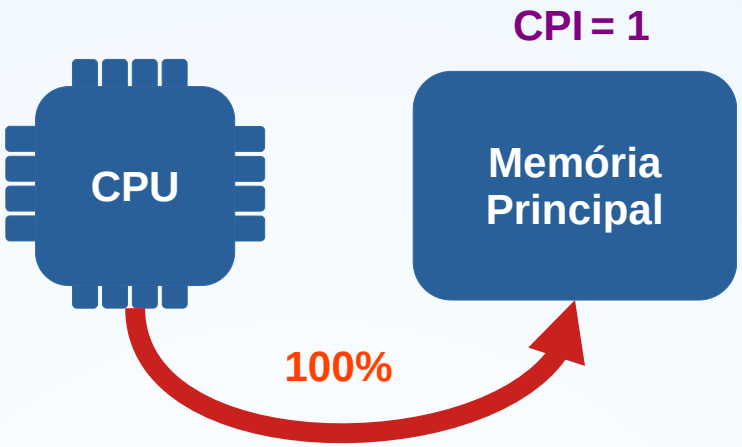
Múltiplas memórias Cache

- Os níveis com **números mais baixos** correspondem a **caches pequenas**
- Quanto **mais alto o número**, tende a ser **mais lenta a cache**, mas também é **maior sua capacidade de armazenamento**
- Apesar das diferentes velocidade de acesso, as memórias cache **ainda são muito mais rápidas** que a memória **principal**
- Os fabricantes adotaram a estratégia de mais de um nível de memória cache pelo mesmo motivo do **conceito de cache** em si: com níveis mais rápidos, a taxa de **acertos (hit) é maior**, melhorando o desempenho geral do sistema computacional



Justificando mais níveis de cache...

- Cenário ideal (pipeline cheio) : $CPI = CPI_{ideal} = 1$

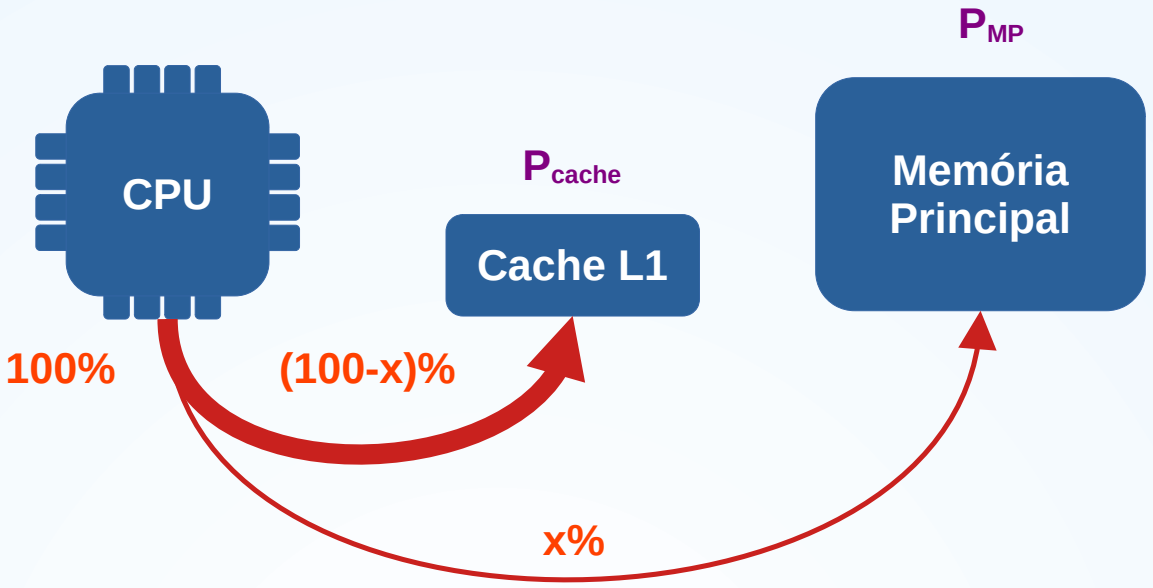


Justificando mais níveis de cache...

- Gastos com paradas devido à faltas no acesso à cache

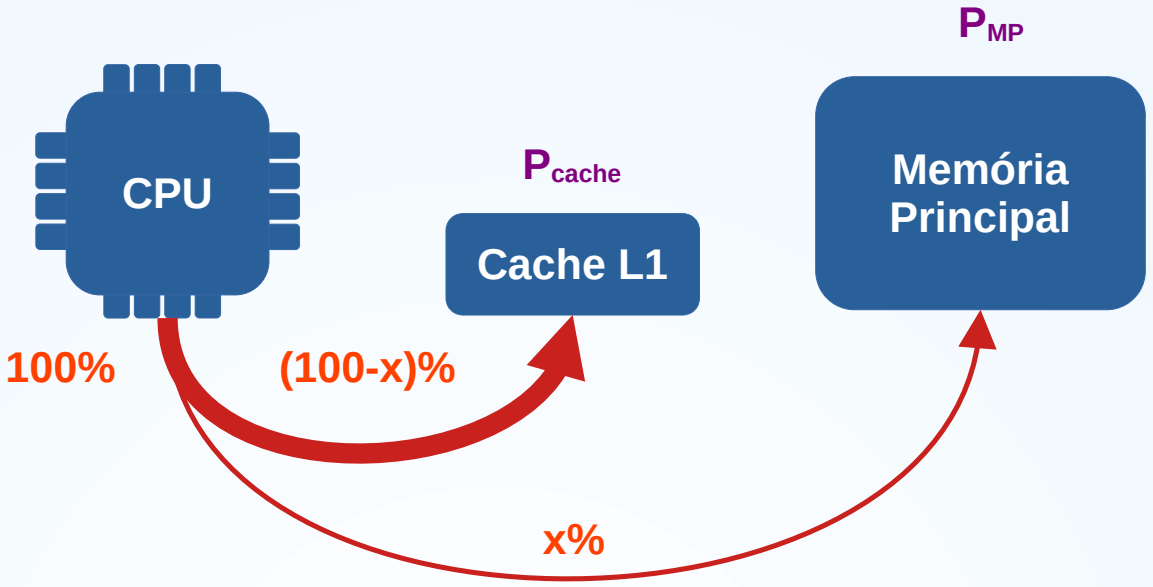
$$CPI = CPI_{ideal} + CPI_{stall}$$

- Cenário com uma cache:



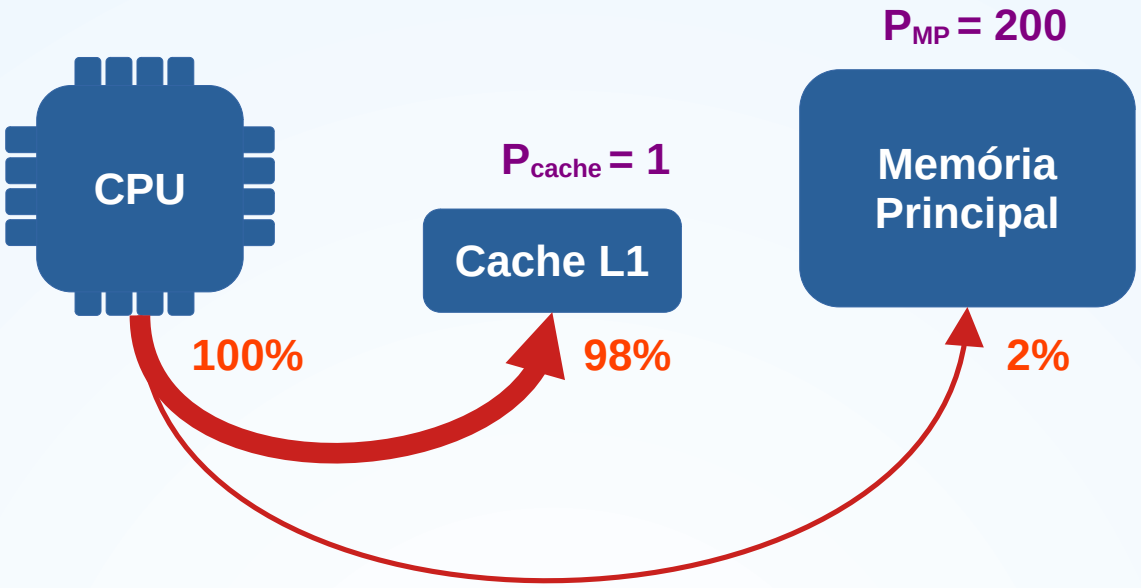
Justificando mais níveis de cache...

- Em que $\mathbf{P_{MP}}$ é a penalidade (em ciclos por instrução) no acesso a memória principal e buscar o dado faltante e $\mathbf{P_{cache}}$ é a penalidade devido à falha (*miss*) em cada nível de cache
- Cenário com uma cache:



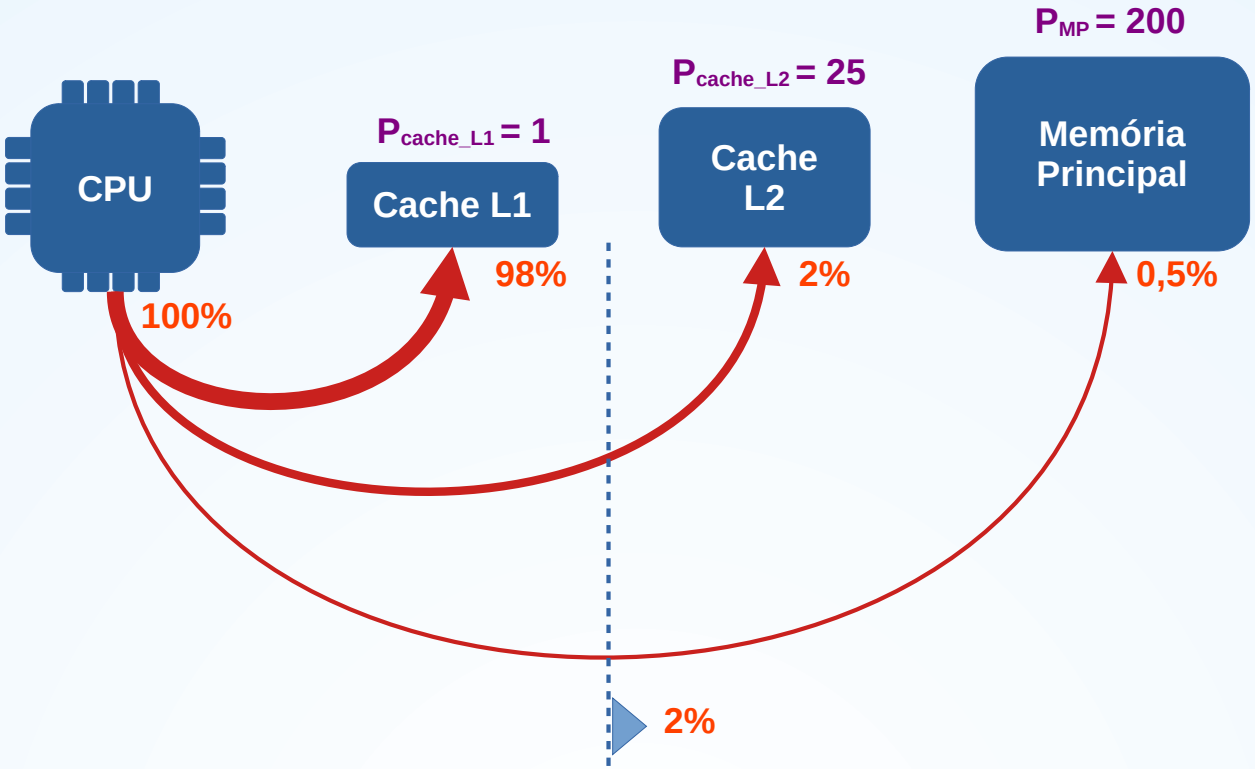
Justificando mais níveis de cache...

- Considerando valores hipotéticos para a análise com **um** nível de cache (ver figura):
- CPI = 1 + 0,02 x 200 = 5



Justificando mais níveis de cache...

- Considerando agora **dois** níveis de cache:
- CPI = 1 + 0,02 x 25 + 0,005 x 200 = 2,5



Impacto da cache no desempenho

- Instruções de LOAD e STORE tem muito **impacto** na taxa de faltas
- Considerando um **tamanho fixo** de cache
  - **Mais palavras** por bloco:

    Maior captura da localidade espacial
  - **Mais associatividade**:

    Maior captura da localidade temporal

Aumentar desempenho via cache

- **Ponto de vista do Desenvolvedor de SW**
  - Conhecimento da estrutura de cache
  - Implementação de código considerando a estrutura da cache
    - Exemplo: tratamento de matrizes (linhas ; colunas)
  - Otimizações de compilador

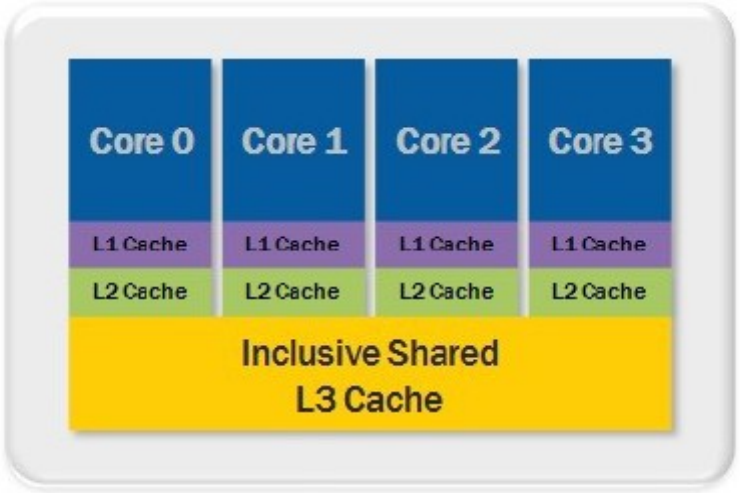
- **Ponto de vista do Projetista do sistema computacional**
  - Seleção de parâmetros da cache para aplicação
  - Capacidade, associatividade, tamanho de bloco, níveis
  - Algoritmo, implementação, compilador

Coerência x Consistência

Coerência de Cache	Consistência de Memória
Está relacionada ao comportamento de leituras e escritas em uma memória específica.	Está relacionada ao comportamento de leituras e escritas em relação a diferentes unidades de memória.
Coerência de cache é necessária para sistemas equipados com memória com função de cache.	Consistência de memória pode ser necessária em sistemas que têm ou não caches.
Assunto de INE5411	Fora do escopo da disciplina

Exemplo: múltiplas memórias Cache

- Caches L1 2 L2 são consideradas privadas e cache L3 compartilhada
- L3 pode ter ou não variáveis compartilhadas entre os núcleos



Core i7 da Intel com 3 níveis de memória cache.

Requisitos de coerência

- (1) O sistema de memória é coerente se preserva a **ordem** do programa **localmente**. Exemplo:
  - Core 0 escreve numa posição X da memória compartilhada
  - Nenhum outro Core faz escrita intermediária nesta posição X
  - Quando o Core 0 lê a posição X, o valor retornado é o próprio valor escrito por ele

Requisitos de coerência

- (2) O sistema de memória é coerente se provê uma **visão global coerente** da memória. Exemplo:
  - O Core 0 escreve na posição X
  - Nenhum outro Core faz escrita intermediária nesta posição (X)
  - O próximo acesso de leitura a X está suficientemente afastado de sua escrita
  - O Core 1 lê a posição X e o valor retornado é o valor escrito pelo Core 0

Requisitos de coerência

- (3) Se o sistema sistema de memória é coerente, então ele se provê **uma mesma ordem de escrita para todos os processadores**. Exemplo:
  - O Core 0 escreve na posição X
  - O Core 2 escreve na posição X
  - Leituras de X feitas pelo Core 0 ou Core 2 enxergam os valores escritos na mesma ordem

Quais as estratégias para garantir coerência?

- Uso de protocolos para manter o estado de compartilhamento de um bloco
  - Mais de uma cópia do bloco em caches distintas

Snooping e Diretório

- Protocolo de coerência de cache Distribuído: ***Snooping***
  - Cada controlador de cache monitora o status, verificando se a cache local tem cópia do bloco referenciado
  - Uso: pequeno número de Núcleos (Cores) (2, 4)

- Protocolo de coerência de cache Centralizado: **Diretório**
  - O status é mantido em um único lugar. Para cada bloco, um vetor de bits indica qual core tem uma cópia
  - Uso: grande número de Núcleos (Cores) (8, 16, ou mais)

Snooping

- Utiliza um meio compartilhado para fazer "*broadcasting*" de faltas na cache
  - Exemplo: barramento compartilhado
- Os controladores de cache são programados para monitorar o barramento (bisbilhotando → *snooping*)

Critério de invalidação

- Ao escrever um item na cache, o processador deve invalidar outras cópias daquele item em outras caches
  - Protocolo ***write-invalidate***
- Um processador tem exclusividade de acesso a um item antes de escrever nele
  - O processador escreve no bloco da sua cache e invalida bloco na cache de outro(s) processador(es)
  - Valor do bloco é atualizado na Memória Principal
- Protocolo de coerência implementados em HW
  - Torna a funcionalidade da cache invisível para o SW
  - Permite que programador se concentre no que importa

FIM MÓDULO 17