Sistemas Operacionais

4º período

Professora: Michelle Hanne

PUC Minas

Gerência de Memória

Gerenciamento de Memória

PUC Minas

A Memória Principal (RAM) é um recurso de suma importância, pois é na memória RAM que estarão armazenados os programas (instruções e dados) que serão processados.

Portanto é de suma importância que os sistemas operacionais façam com muito cuidado, o gerenciamento desta memória.

(Tanenbaum, 2016)

Embora os computadores atuais possuam quantidades de memórias muitas vezes maiores do que os modelos mais antigos, de 20, 10 anos atrás, os programas também se tornaram maiores, e muito mais rapidamente do que a memória.

Pode-se afirmar que “programas tendem a se expandir a fim de ocupar toda a memória disponível”, lei de Parkinson.

(Tanenbaum, 2016)

Gerenciamento de Memória

PUC Minas

O que o programador deseja ?

- Dispor de uma memória infinitamente Grande.
- Que a memória seja rápida e não volátil.
- Tudo isso a baixo custo.

Infelizmente ainda não temos tecnologia para isto...

O que fazer então?

Ao longo dos anos foi descoberto o conceito de hierarquia de memórias, os computadores tem :

- Alguns megabytes de memória cache muito rápida, de custo alto e volátil.
- Alguns gigabytes de memória principal, volátil e custo e velocidade médios.
- Alguns terabytes de armazenamento secundário, em disco, não volátil e velocidade e custo baixos.
- Armazenamento removível: DVDs, dispositivos USB e etc.... .

O objetivo do sistema operacional é abstrair essa hierarquia em um modelo útil e, então gerenciar a abstração.

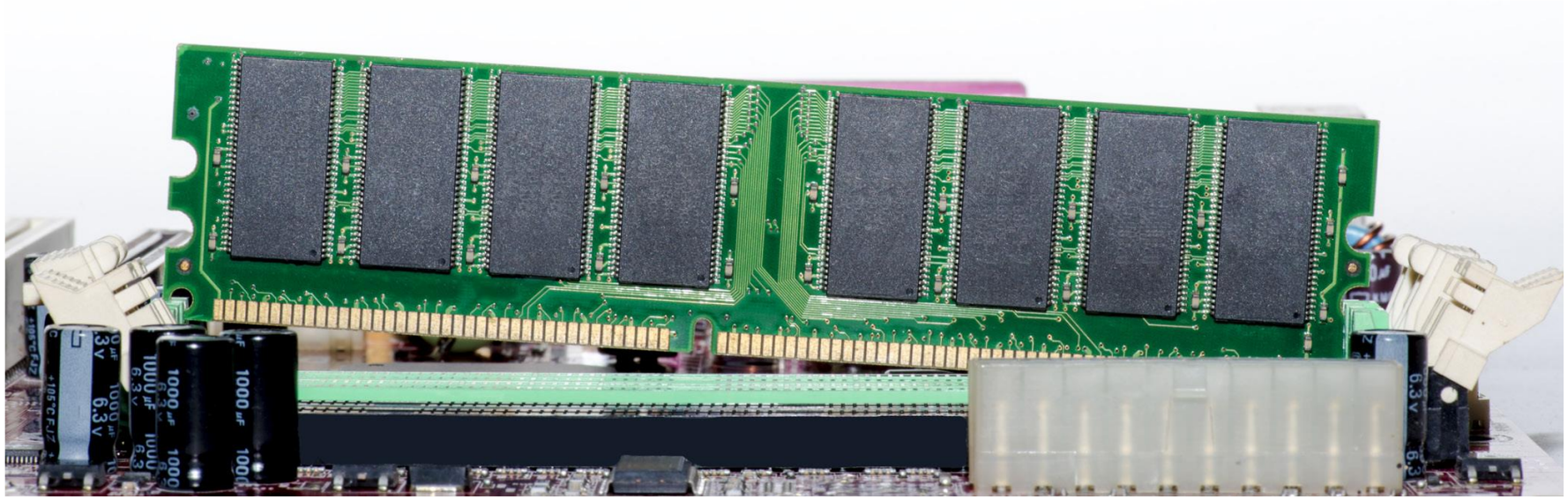
(Tanenbaum, 2016)

Memória Física ou Principal

PUC Minas

- Também chamada de memória principal ou **random access memory (RAM**), a memória física é um dispositivo **semicondutor** que armazena os dados, **de forma aleatória, em matrizes (arrays).**
- Uma **memória volátil só permanece com informações quando energizada**. Uma memória de acesso aleatório, diferentemente de acessos sequenciais de discos, identifica a capacidade de acesso a qualquer posição e em qualquer momento.
- Sua localização é próxima do processador, região mais rápida dentro da arquitetura de qualquer dispositivo eletrônico.

Fonte: SHUTTERSTOCK, 2018.



Memória Física ou Principal

PUC Minas

Tanto a memória quanto o processador na arquitetura de um computador são dispositivos de alta velocidade. **Na computação, velocidade é sinônimo de alto consumo de energia.** Diferentes tendem a minimizar o consumo de energia pelas memórias do processador.

- Exemplos que incorporam elementos de armazenamento não voláteis:
- Memória de acesso aleatório resistiva (**RRAM**) (TOSSON, 2018)
- Memória de acesso aleatório magneto resistiva (**MRAM**) (KHVALKOVSKIY, 2013), para alcançar potência, mantendo baixa energia de operação e velocidade de acesso rápido no processamento de dados voláteis.

Características como baixo consumo e velocidade mais rápida, permitem o uso desse hardware em dispositivos menores, como relógios e sensores. Essa geração de hardwares menores, com mais capacidade e com integração com sistemas de informação, modelo denominado IoT (internet of things)

Memória Cache

PUC Minas

- Na computação, memória cache é um dispositivo de acesso rápido que está localizado dentro de um dispositivo de hardware, como um processador ou um disco rígido. A finalidade dessa memória é melhorar o desempenho desse dispositivo em um processo como o de leitura de dados (STUART, 2011).

- No disco rígido, por exemplo, a **memória cache armazena** as últimas trilhas percorridas pelo leitor de disco, evitando, assim, a repetição do leitor em percorrer as mesmas trilhas várias vezes, o que torna o desempenho do disco mais rápido. A memória cache está armazenada na **placa lógica do disco rígido**. **Nos processadores**, essas memórias são representadas pelas siglas **L1, L2 e L3**.

Cache (Motivação)

PUC Minas

- Processador precisa ler e escrever dados na memória principal.
- Mas, tipicamente, o processador é muito mais rápido que a MP.
- Exemplo (em um computador hipotético):
  - Uma soma, tempo de leitura dos operandos a partir da memória é de 60 ns.
  - Os registradores, o tempo efetivo de soma é 0,3 ns.
- Processador precisa aguardar enquanto dados são carregados os dados da Memória RAM para os registradores.
  - Tempo chamado de **wait state**.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Cache – Princípio da Localidade

PUC Minas

Na década de 1960, pesquisadores começaram a estudar a execução de programas, sobre os padrões de acesso à memória.

Notaram que o acesso à memória não é totalmente aleatório.

Quando um determinado endereço é acessado, há maior probabilidade de acesso à endereços **próximos**.

Princípio da Localidade.

- Foram notados dois tipos de localidade distintos:
- A **localidade espacial**.
- E a **localidade temporal**.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Cache – Princípio da Localidade Espacial

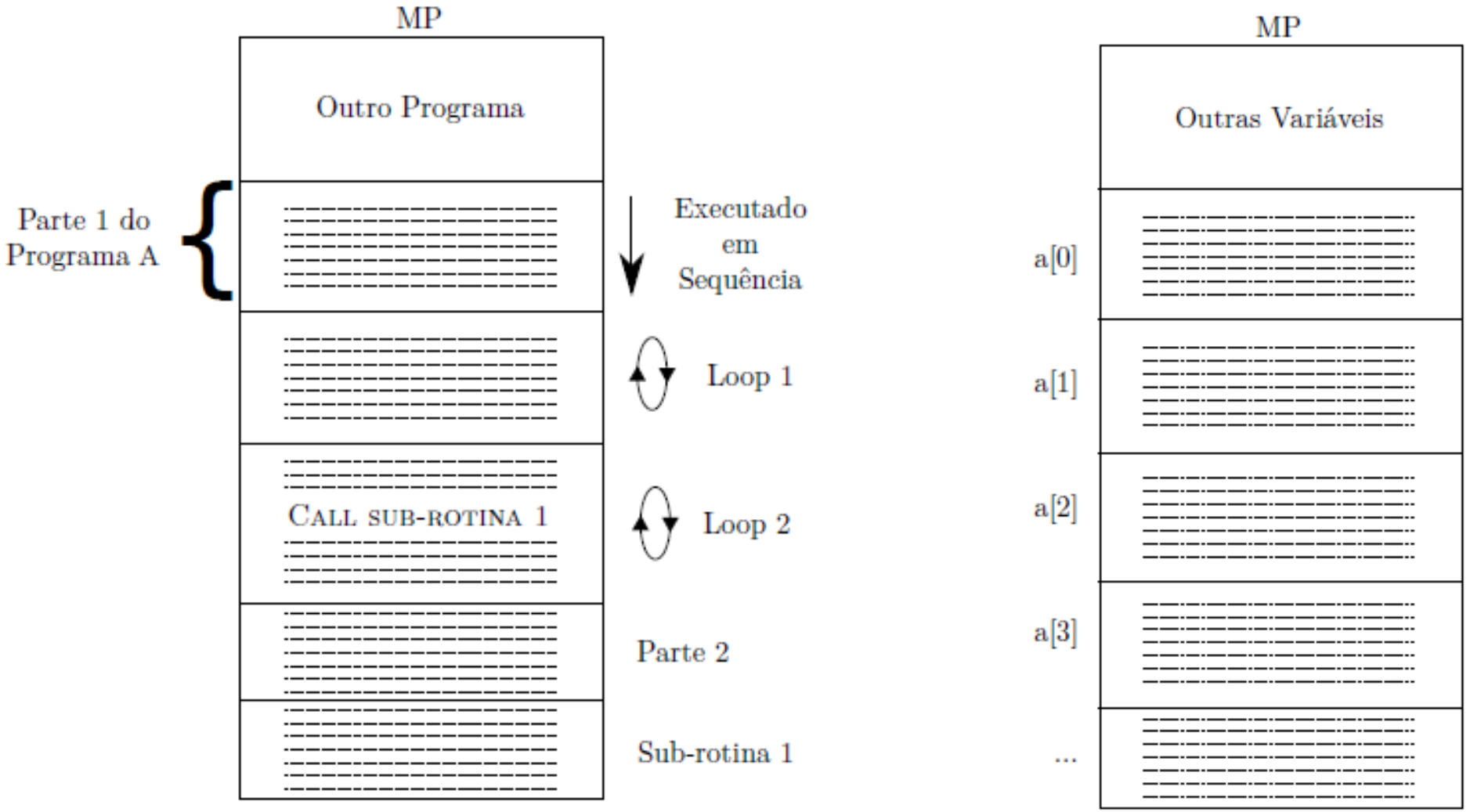
PUC Minas

- Princípio da localidade espacial:
- Se um endereço x *é* acessado, é provável que endereços $x + 1, x + 2, x + 3, \ldots$ Também sejam em um **futuro próximo**.
- Este princípio se aplica à execução de programas por dois motivos principais:
  - A organização do código executável em memória.
  - A existência de estruturas de dados na forma de coleções de itens.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Cache – Princípio da Localidade Espacial

PUC Minas



Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Cache – Princípio da Localidade Temporal

PUC Minas

- Se um endereço *x é* acessado, é provável que ele seja acessado novamente num **futuro próximo**.
- Este princípio se aplica à execução de programas por três motivos principais:
  - A existência de variáveis que são lidas/modificadas frequentemente em uma mesma região do código.
  - A existência de repetições com variáveis acumuladoras.
  - A existência de repetições que executam as mesmas instruções várias vezes seguidas.

```
int a[10], b = 0, i;
for (i = 0; i < 10; i++)
    a[i] = i;
for (i = 1; i < 10; i++)
    b += a[i-1] * i + a[i];
```

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Cache

PUC Minas

O termo *cache é* usado em vários contextos em computação.

- Cache de um *browser*.
- Cache de arquivos em memória
- Memória cache.

Em todos estes contextos, a ideia é armazenar informações de interesse em um local de **acesso mais rápido**.

- É impossível armazenar tudo, mas se algo é encontrado na cache, ganhamos em desempenho.

Note que a consulta à cache consome algum tempo.

Mas supõe-se que esse tempo é pequeno em relação ao ganho que ocorre quando achamos o dado.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Funcionamento da Memória Cache

PUC Minas

A memória cache explora o princípio da localidade.

- Tanto temporal, quanto espacial.

Ela é uma memória relativamente pequena, mas bastante rápida.

Toda vez que o processador tenta acessar uma posição de memória, ele primeiro verifica a existência do dado na cache.

Quando um dado é acessado na MP, ele é armazenado na cache.

- Princípio da localidade temporal: ele poderá ser usado novamente em breve.

Além do dado em si, armazena-se um **bloco** de dados próximos.

- Princípio da localidade espacial: dados próximos podem ser usados em breve.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Operação de Leitura em Memória Cache

PUC Minas

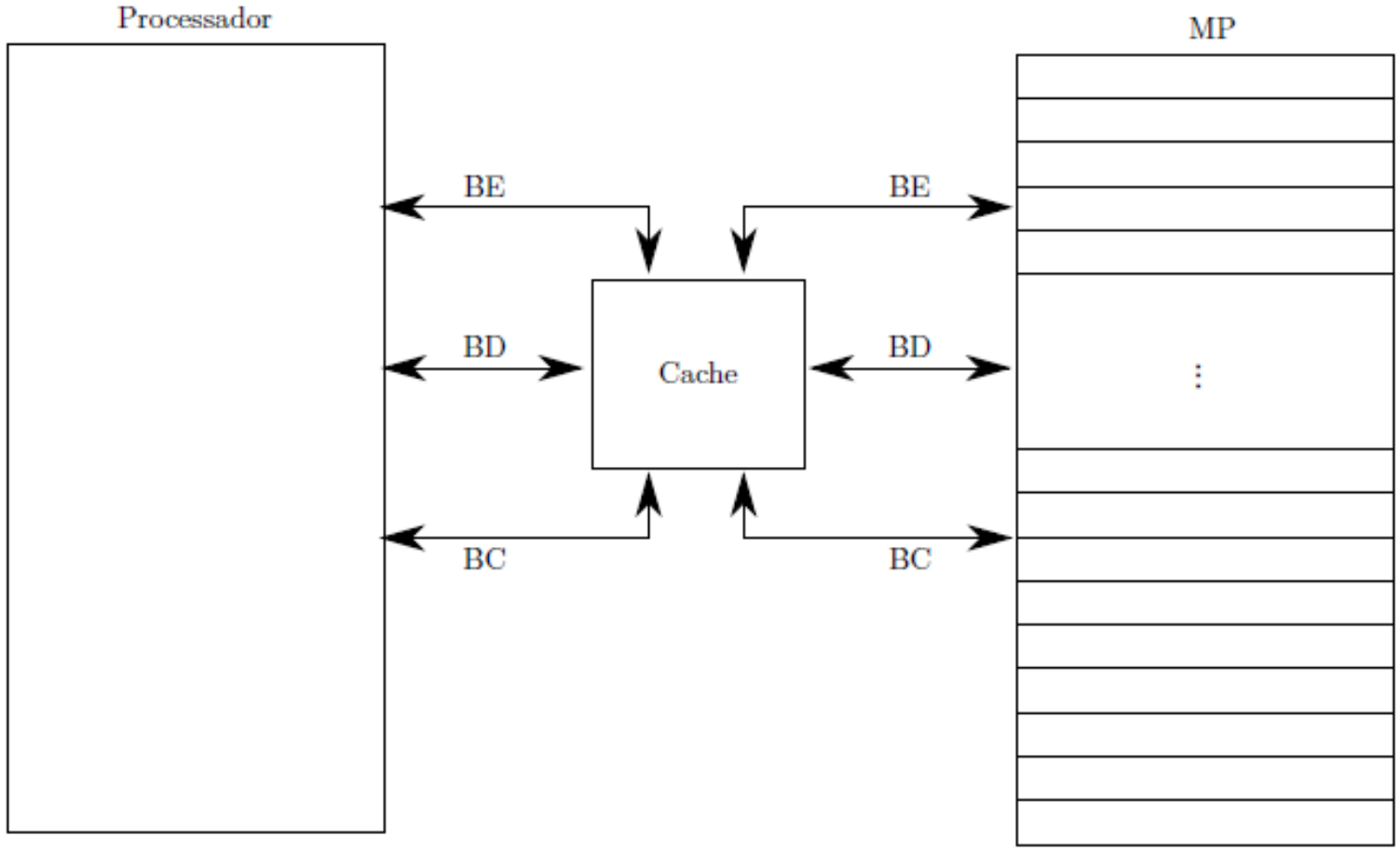
- A existência da memória cache altera o processo de leitura de dados da MP.
- Quando a CPU deseja ler uma posição de memória, os seguintes eventos ocorrem:
  - Processador escreve o endereço a ser lido no BE de acesso à MP.
  - Controlador da cache intercepta pedido e verifica se o dado está em cache.
    - Se sim, temos um acerto (**hit**): dado é copiado para o processador.
    - Se não, temos um falta (**miss**): cache pede o dado `a MP, o armazena e repassa para a CPU.
- Note que, no caso de um **miss**, cache solicita um bloco inteiro `a MP.
  - Conjunto de dados maior que o dado a ser lido.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Operação de Leitura em Memória Cache

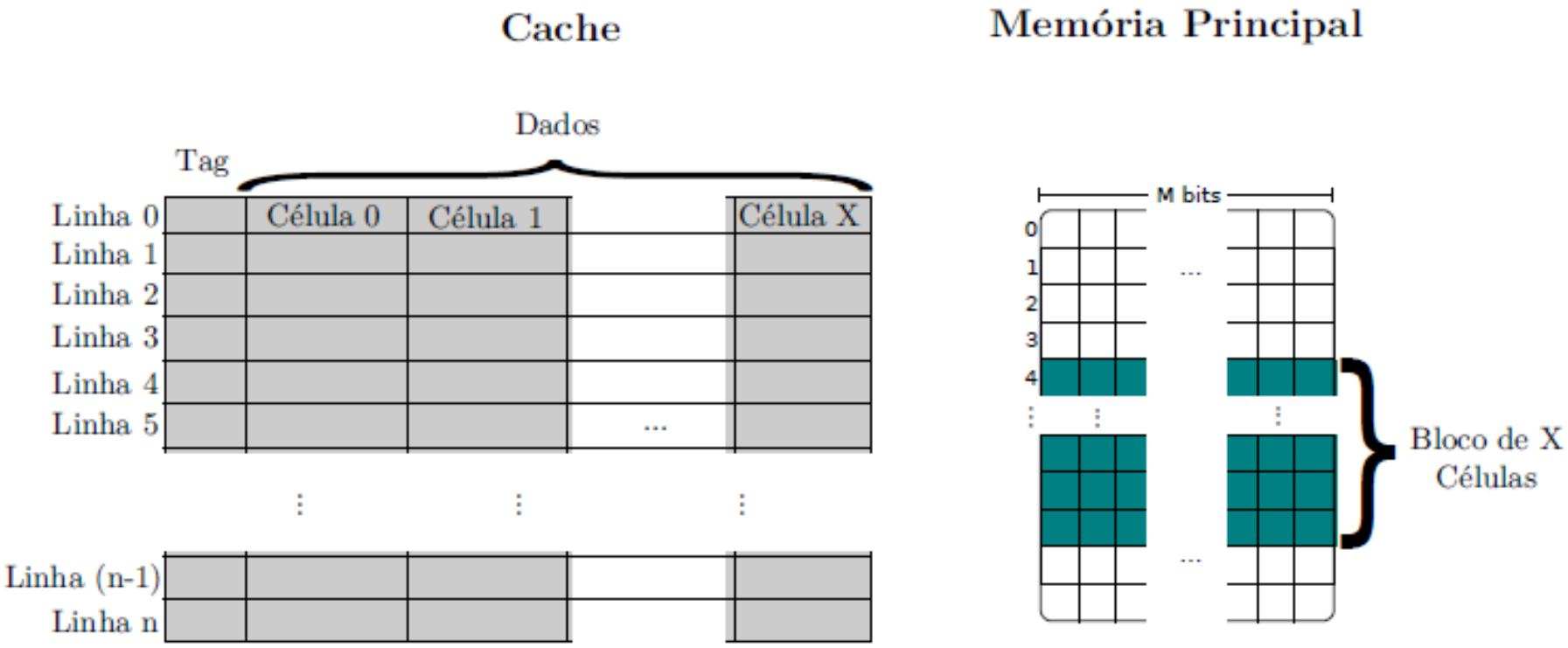
PUC Minas

- Possível arquitetura com a presença da memória cache.



Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Estrutura de uma Memória Cache



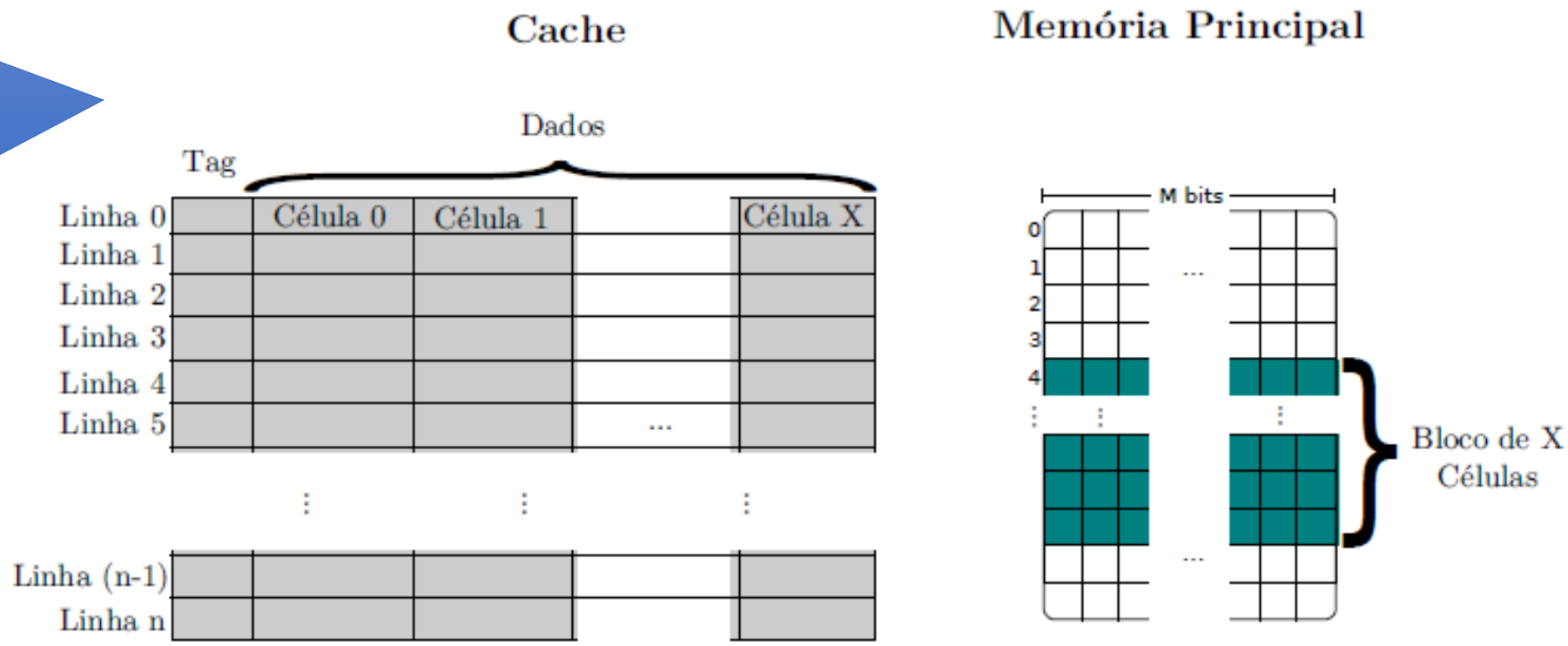
- A memória cache é organizada em **linhas**.
  - Cada linha contém X células.
  - E um campo *tag*.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Estrutura de uma Memória Cache

PUC Minas

Há ainda o campo tag. Indica, de alguma forma, o endereço do bloco na MP. Lembre-se: a MP é muito maior que a cache.



- Células em uma linha representam células sequenciais na MP.
  - Um **bloco**.
  - Quando um dado é copiado da MP para a cache, **todo o seu bloco é trazido juntamente**

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

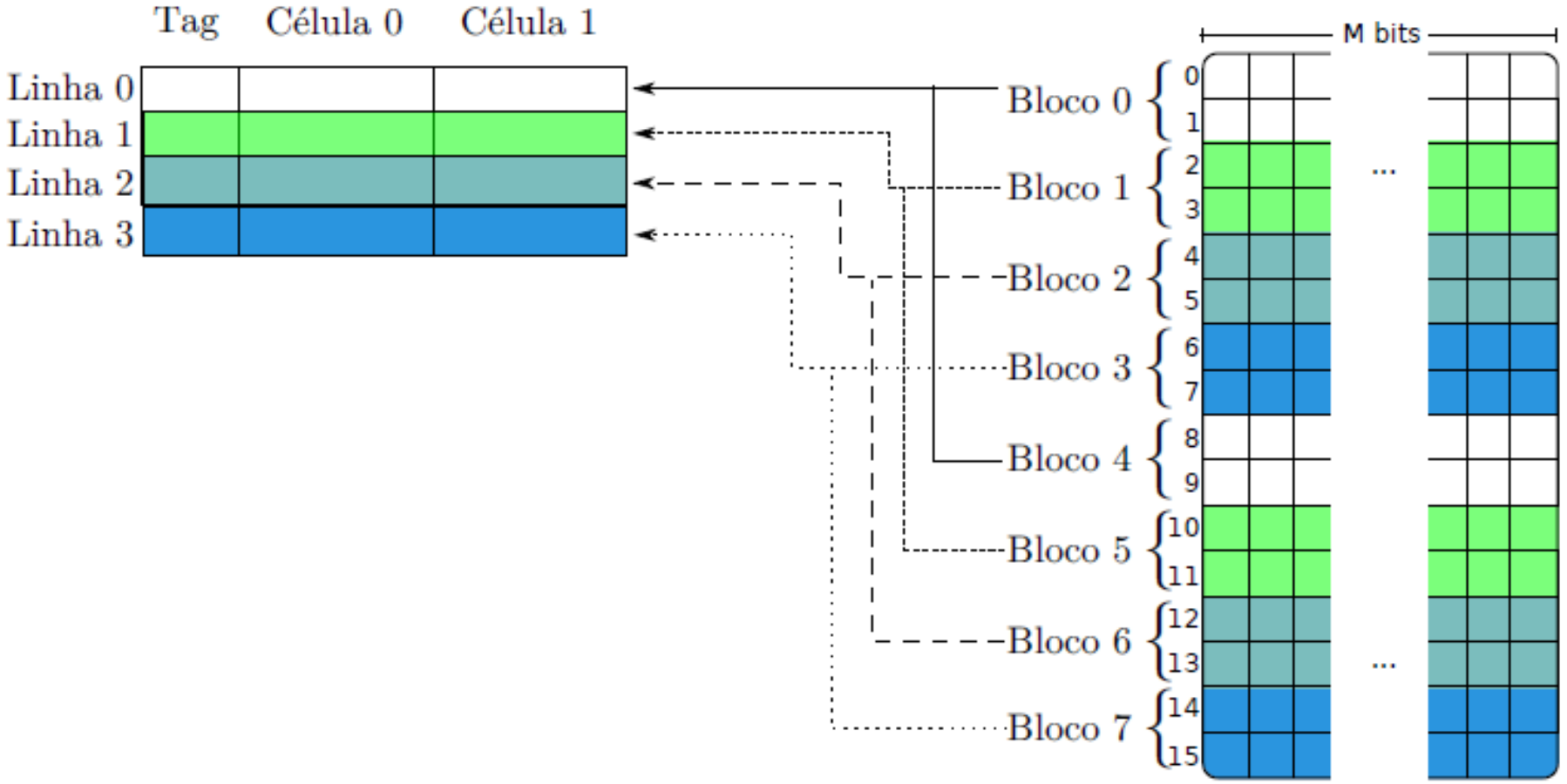
Memória Cache vs Memória Principal

PUC Minas

- A memória cache é muito menor que a MP.
- Não podemos guardar todos o conteúdo da MP na cache.
- Só podemos armazenar um **subconjunto dos blocos da MP**.
- Por isso, algumas **questões fundamentais**:
  - Quais blocos **manter** na cache?
  - Como determinar, **de forma eficiente**, se um dado está em cache?
  - O que fazer quando um dado em cache é **alterado**?

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Cache – Mapeamento Direto



Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Cache – Problema de Mapeamento

PUC Minas

Como a cache guarda um subconjunto pequeno dos blocos da MP, a primeira tarefa é determinar se o dado está na cache.

Sabemos que o campo tag é usado para, de alguma forma, indicar o campo de endereço do bloco na MP.

- Busca tem que ser eficiente.
- Caso contrário, tempo de acesso à cache fica alto

Se usarmos o campo tag para armazenar o endereço na MP da primeira célula do bloco, podemos fazer uma **busca sequencial**:

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Cache – Problema de Mapeamento

PUC Minas

Na prática, problema da busca em cache é resolvido através de tipos de mapeamento.

Maneiras de se determinar em qual linha da cache cada bloco da MP pode ser inserido.

Há três tipos de mapeamento:

- O mapeamento direto.
- O mapeamento associativo.
- O mapeamento associativo por conjunto.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Cache –Mapeamento Associativo

PUC Minas

Compara as tags em paralelo

- Exemplo: compara todas as tags na cache ao endereço ao mesmo tempo.

A comparação em paralelo é possível em hardware.

Cada linha da cache estaria ligada a um circuito comparador cuja entrada seria o endereço a ser acessado.

As saídas dos comparadores seriam combinadas através de uma função lógica OU.

Se uma das linhas tivesse a tag adequada, a saída seria verdadeira.

Este tipo de solução recebe o nome de mapeamento associativo.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Cache –Mapeamento Associativo (Vantagens e Desvantagens)

PUC Minas

O grande benefício do mapeamento associativo é flexibilizar a alocação de blocos em linhas da cache.

- Podemos ter blocos em quaisquer linhas.
- O que tende a aumentar a taxa de cache hit.

Como desvantagem, temos a maior complexidade do circuito da cache.

- Principalmente em termos econômicos.
- Cache associativa é mais cara que a cache com mapeamento direto.
- Principalmente para caches grandes.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Cache –Mapeamento Associativo por Conjunto

PUC Minas

O meio termo entre o mapeamento direto e o associativo é o mapeamento por conjunto.

A cache é dividida em N conjuntos de linhas.

Assim como no mapeamento direto, cada bloco da MP tem um mapeamento fixo para um conjunto específico.

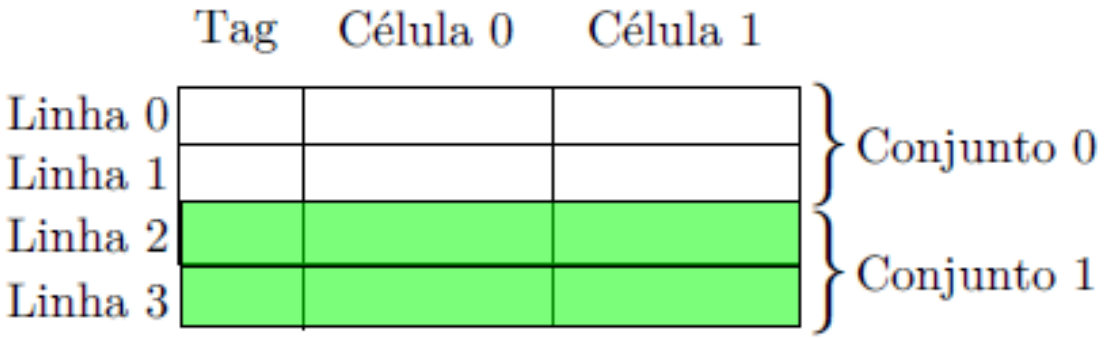
Mas dentro do seu conjunto, bloco pode ser armazenado em qualquer linha.

- Como no mapeamento associativo.

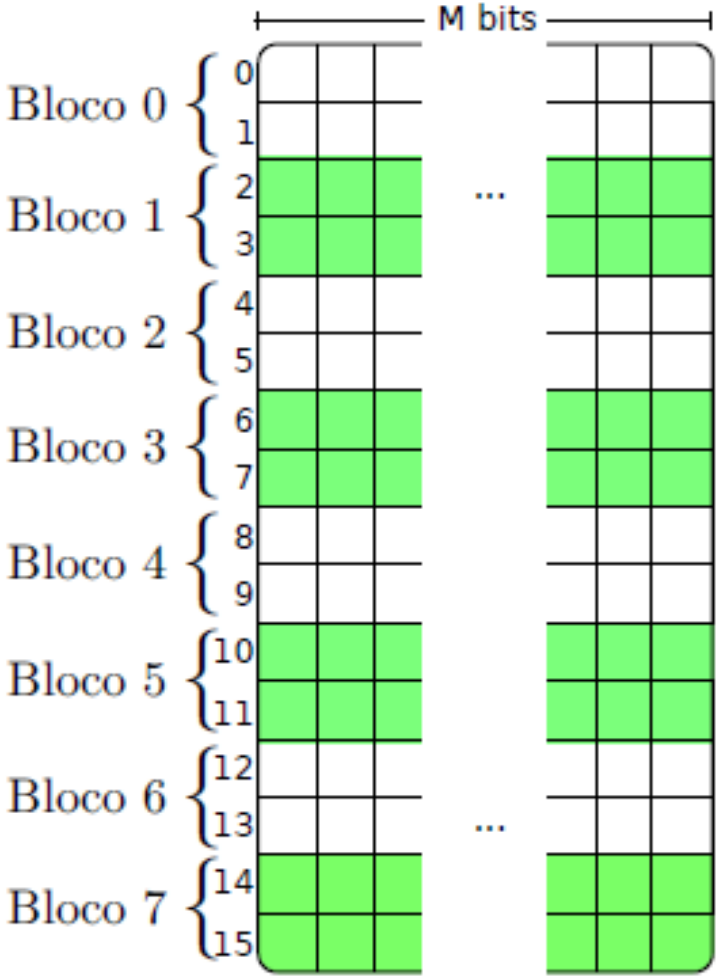
Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Cache –Mapeamento Associativo por Conjunto

PUC Minas



Mapeamento de Endereços	
Conjunto 0	Conjunto 1
0000	0010
0001	0011
0100	0110
0101	0111
1000	1010
1001	1011
1100	1110
1101	1111



Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Cache – Níveis

PUC Minas

Computadores modernos geralmente têm vários nvíeis de cache.

- **Exemplo: cache L1, L2 e L3.**

Os níveis mais baixos correspondem a caches pequenas e rápidas.

- Quanto mais alto o nível, mais lenta é a cache, mas também maior.
- Embora ainda muito mais rápidas e menores que a MP.

Se um nível é muito mais rápido que os níveis acima e conseguimos boas taxas de cache hit, então há ganho de desempenho.

Além do campo tag associado a cada linha e dos dados, uma cache pode armazenar certos bits de controle.

- Exemplo: o bit *dirty*.
- Indica que conteúdo do bloco foi alterado e precisa ser atualizado na MP.

Outro bit comum é o de validade.

- Indica se o conteúdo de uma linha da cache é válido.
- Conteúdo pode ser inválido, por exemplo, quando o processador é ligado.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Cache – Níveis

PUC Minas

- Obviamente queremos sempre a maior cache possível.
  - Mas caches muito grandes não são viáveis.
- Tamanhos típicos giram em torno de:
  - Dezenas de KB na L1.
  - Centenas de KB na L2.
  - Alguns poucos MB na L3.
- Uma questão interessante é como alocar a memória disponível.
- Exemplo: qual a largura das linhas da cache.
  - Linhas maiores favorecem a localidade espacial.
  - Mas com linhas menores, podemos ter mais linhas.
  - Favorece situações em que os acessos são mais distribuídos

Desempenho com Memória Cache

PUC Minas

- O objetivo de inserirmos a memória cache na hierarquia é **reduzir o tempo de acesso à memória**. O objetivo é alcançado?
- Isso depende de vários fatores:
  - O tempo de acesso quando há um *hit*.
  - O tempo de acesso quando há um *miss*.
  - E a **eficiência da cache**.
- Define-se a eficiência de uma memória cache como o percentual de *hits*:

$$E_c = \frac{100 \times hits}{acessos}$$

- Eficiência varia com uma série de fatores, incluindo o programa em execução.
- Espera-se algo em torno de 95% a 98%.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Cache - Desempenho

PUC Minas

Para medir o desempenho de um sistema que possui memória cache, podemos calcular o **tempo médio de acesso à memória (TMAM)**. Considera eventos de:

- acerto (cache hit),
- falta (cache miss) e
- taxa de faltas.

Fórmula para cálculo:

TMAM = tempo de acerto + taxa de faltas × penalidade por falta

- **Tempo de acerto:** tempo de descobrir que ocorreu um acerto mais o tempo de acesso à memória cache.
- **Penalidade por falta:** tempo de descoberta da ocorrência da falta adicionado ao tempo de cópia de um bloco da memória principal para a cache e o tempo de acesso à cache.
- **Taxa de faltas:** número de faltas que ocorrem por instrução.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Cache – Desempenho - Exemplo

PUC Minas

1) Considere uma máquina que possui um relógio com **um ciclo de 4 ns**, apresenta uma **penalidade de falta igual a 30 ciclos de relógio**, **uma taxa de faltas de 0.05** faltas por instrução e o **tempo de acerto igual a 2** ciclo de relógio.

a) Calcule TMAM (Tempo Médio de Acesso a Memória) para esta máquina.

TMAM = tempo de acerto + taxa de faltas × penalidade por falta

Tempo de Acerto-> 4*2 = 8 ns

Penalidade por Falta=> 30*4 = 120 ns

Taxa de Faltas é 0,05

TMAM= 8+0,05*120 = 14 ns

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Cache – Desempenho - Exemplo

PUC Minas

b) Suponha que se aumente a quantidade de memória cache desta máquina, e, com esta mudança, a taxa de faltas mude para 0.02 faltas por instrução e o tempo de acerto passa para 1,5 ciclos de relógio. O que ocorrerá com o desempenho dessa máquina?.

TMAM = tempo de acerto + taxa de faltas × penalidade por falta

Tempo de Acerto-> 4*1,5 = 6 ns

Penalidade por Falta=> 30*4 = 120 ns

Taxa de Faltas é 0,02

TMAM= 6+0,02*120 = 8,4 ns

Então, esta mudança melhorou o desempenho da máquina pois o TMAM diminuiu.

Melhorou o desempenho em 40%.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Buffer de Hard Disck

PUC Minas

Buffer é uma área temporária – geralmente localizada na memória principal. Exemplo: Na operação de copiar (Ctrl + C) e colar (Ctrl + V) em sistemas operacionais Windows ou outro com tela gráfica. Ao copiar quaisquer dados, ele fica armazenado no buffer (área temporária de armazenamento) até que a operação de colar seja executada.

No disco rígido, o buffer tem papel primordial, pois ele melhora o desempenho do disco justamente porque minimiza o tempo de acesso aos dados. **Um acesso convencional, por exemplo, a um disco rígido de operação de leitura/gravação é geralmente lento, pois há diversos aspectos, como velocidade de rotação e velocidade de deslocamento do braço e das operações gravação/leitura da trilha, que, somados, geram um pequeno atraso**.

O **acesso ao buffer é mais rápido por ser uma memória rápida**; sendo assim, ele é consultado **primeiro pelo sistema operacional antes do procedimento de acesso tradicional realizado pelo sistema operacional, ou seja, o sistema operacional consulta o buffer ao invés de fazer uma consulta tradicional no disco.** Memórias buffers podem ser encontradas intrínsecas em dispositivos como **discos rígidos, gravadores de CD/DVD e impressoras, ou seja, em qualquer dispositivo que faça operação de entrada/saída**.

Enquanto que os HDs possuem tamanho (volume) em gigabytes ou terabytes, os buffers são de tamanhos menores (megabytes).

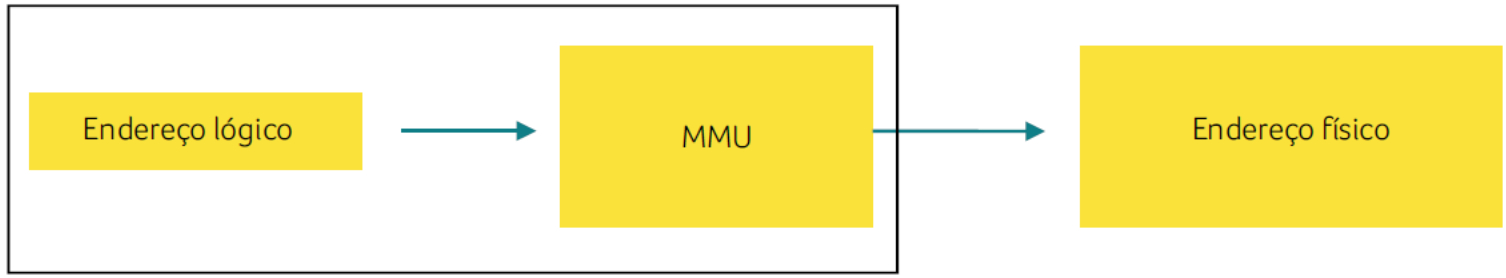
Memória lógica x memória física

PUC Minas

A memória principal de um computador possui dois tipos de memória: lógica e física. A memória lógica é aquela utilizada pelos programas sempre que precisam alocar um espaço em memória, e esse espaço será alocado nas memórias físicas.

Para que o processo se concretize, **há a necessidade de um mapeamento entre endereços de memória lógica para endereço de memória física**. Esse processo de tradução de endereços lógicos em endereços físicos é realizado por uma unidade de gerência de memória **chamada MMU (memory management unit) (FIGURA 9).**

Essa unidade pode ser formada por um único chip ou por um conjunto de chips (SILBERSCHATZ, 2015). A Unidade 5 trará mais detalhes sobre memória lógica e memória virtual.



Memória Virtual

PUC Minas

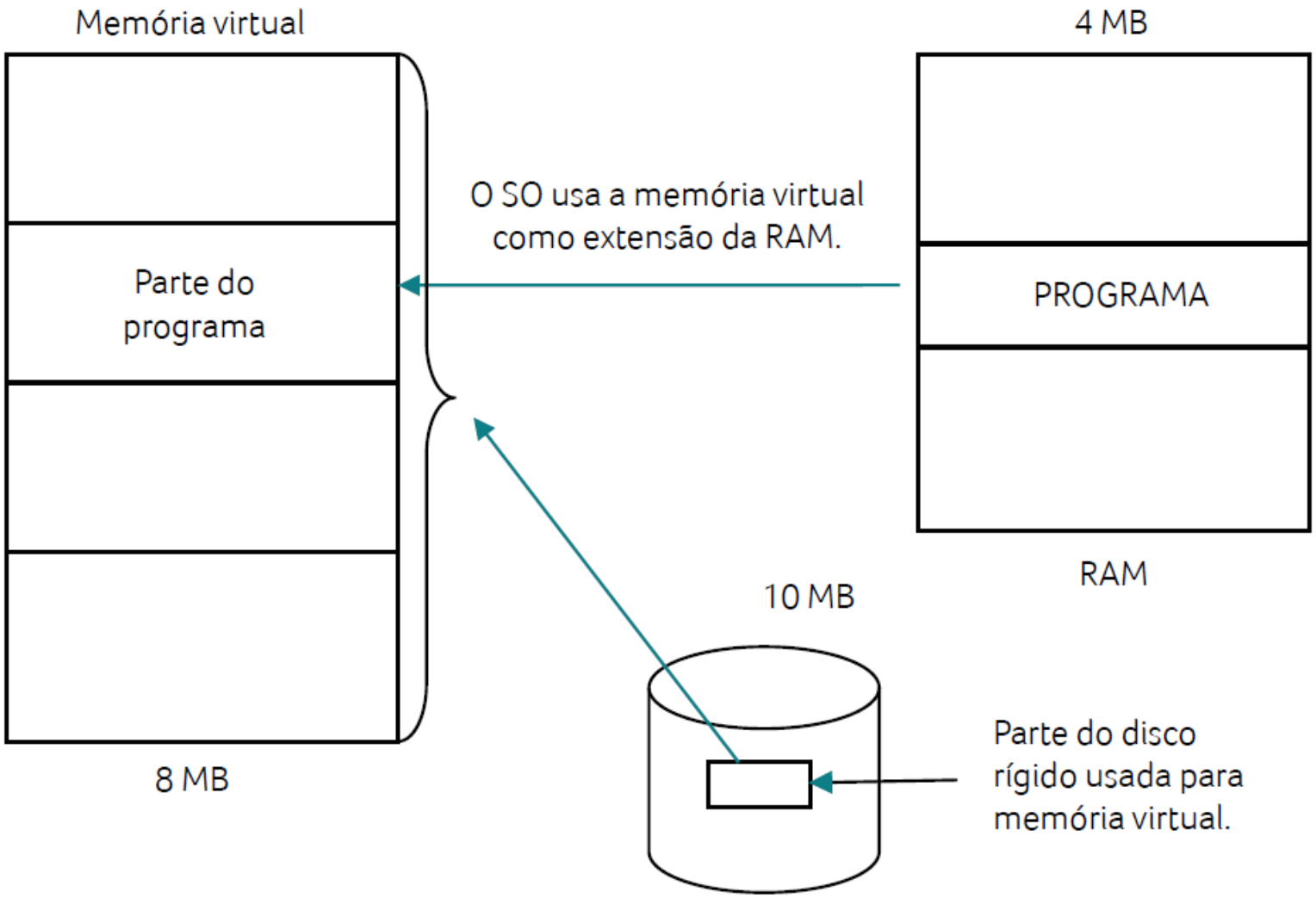
Essa memória também é chamada de **memória secundária** ou *memória de disco* Essa memória é um espaço em disco rígido utilizado pelo sistema operacional quando o espaço em memória física não é suficiente.

Uma parte do **disco rígido é usada como extensão da memória principal e possui espaços de alocação** onde as partes dos programas serão armazenadas.

A memória virtual é no mínimo duas vezes maior que a memória física.

Imagine um usuário que acaba de colocar o Windows em modo suspenso. Nesse momento, o sistema operacional armazena todos os dados em memória física em memória virtual**. Agora, se a RAM for de capacidade de 4 MB, a memória virtual deverá ser no mínimo de 8 MB em um disco de 10 GB.**

Memória Virtual



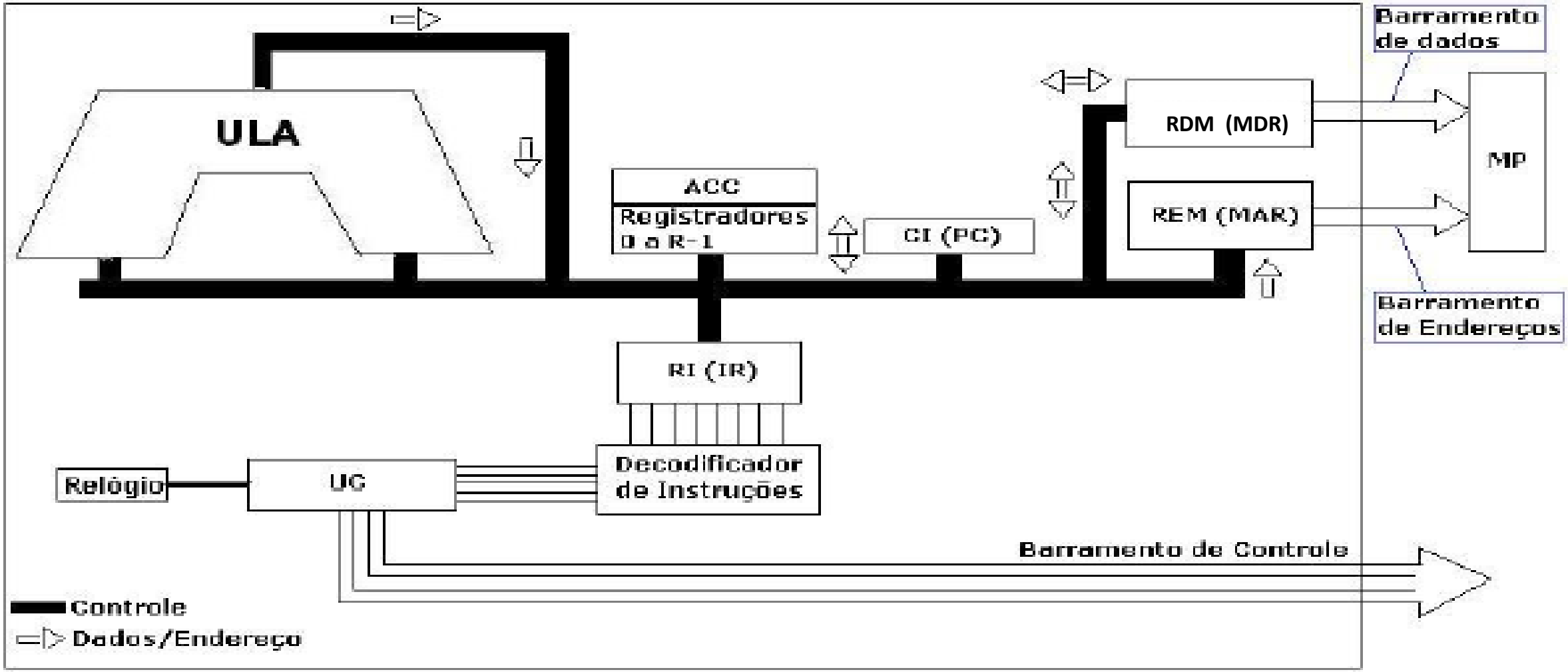
Registradores

PUC Minas

Registradores são memórias utilizadas pela **CPU para aceitar, armazenar e transferir com alta velocidade os dados e instruções que estão sendo usados pela própria CPU**.,

Há tipos variados com diversas finalidades de registradores, como: **AC ou acumulador, registrador de dados ou DR**, **registrador AR ou endereço**, **contador de programa (PC), registrador de dados de memória (MDR), registrador de índice e registrador de** buffer de memória.

Registradores



Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Registradores

PUC Minas

Registradores são memórias capazes de armazenar pequenas quantidades de bits internamente ao processador.

- **Em geral, registradores são do tamanho da palavra do processador.**
- Porém, podem ser ligeiramente maiores ou menores.

Registradores armazenam as informações importantes para a execução imediata.

- Ou em um futuro muito próximo.
- Por exemplo: soma de dois números.
- Ambos os números estarão em registradores.
- Resultado da soma será armazenado em um registrador.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Registradores

PUC Minas

Registradores não servem apenas para armazenar operandos e resultados de operações.

Eles também armazenam informações como:

- O estado atual do processador.
- Informações sobre a última operação realizada. Ex: se o resultado foi zero ou negativo.

O ponto atual de execução de um programa.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Registradores

PUC Minas

Processadores modernos geralmente contêm vários registradores.

Um processador x86 (32 bits), por exemplo, contém cerca de 32 registradores **visíveis ao usuário**.

- Há registradores especiais, usados para controle do processador.

Mas nem todos os registradores são iguais.

- Um registrador pode servir a um propósito específico.

De maneira geral, podemos dividir os registradores em dois grupos:

- Registradores acessíveis ao usuário.
- Registradores internos.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Registradores

PUC Minas

- Dentre os Registradores acessíveis ao Usuário, podemos ter uma nova subdivisão.
  - **Registradores de dados.**
  - **Registradores de endereço.**

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Registradores de Dados

PUC Minas

Armazenam dados a serem processados.

- Ou resultados de operações.

Operações da CPU operam sobre estes dados.

Processadores modernos geralmente fazem distinção entre registradores inteiros e de ponto flutuante.

Processadores mais antigos ou mais simples normalmente contam com um registrador chamado **acumulador**.

- Usado como entrada para operações.
- Mas também acumula a saída.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Registradores de Endereço

PUC Minas

Armazenam valores numéricos que dizem respeito a endereços na memória principal. Podem ser Absolutos ou relativos.

Certas operações de uma CPU usam os valores destes registradores como referências para acessar a memória principal.

Relação direta com os **modos de endereçamento**.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Outros Registradores

PUC Minas

- Dependendo da arquitetura, **há vários outros tipos de registradores acessíveis ao usuário.**
- Algumas contêm, por exemplo, registradores constantes.
  - Não podem ser escritos.
  - Possuem sempre o mesmo valor.
  - Algum tipo de constante útil, como 0, 1, ou π
- Outros comuns são os registradores de propósito especial:
  - *Program Counter* (PC): contador de programa.
  - Palavra de *status*: informações sobre o estado atual do processador.
  - Em geral, não podem ser **diretamente** escritos.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Registradores Internos

PUC Minas

Registradores não são acessíveis ao usuário, nem para leitura, nem escrita.

- Exemplo: programa em execução do usuário.

Auxiliam a CPU nas operações a serem executadas.

Exemplo de tipos de Registradores Internos:

- **Registrador de Instrução.**
- **Memory data register (MDR).**
- **Memory address register (MAR).**

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Registrador de Instrução

PUC Minas

As instruções (operações) que devem ser executadas pelo processador são também armazenadas em memória (Registrador de Memória).

Logo, elas são representadas como conjuntos de bits.

Assim como nos esquemas de representação de números, cada bit (ou conjunto de bits) possui um significado.

Para "entender" a operação, CPU precisa analisar os bits da instrução.

Para isso, instrução é colocada em um registrador especial.

- O Registrador de Instrução.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Registradores de Memória

PUC Minas

Este conjunto de registradores é usado para auxiliar no acesso à memória principal.

Tipicamente, são dois registradores diferentes:

- *Memory data register* (MDR).
- *Memory address register* (MAR).

Estes registradores são usados como interfaces para o barramento de acesso à memória.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Registrador de Memória MAR (Memory Adress Register)

PUC Minas

O MAR é o registrador que armazena o endereço da memória principal a ser acessado.

- Tanto no caso de uma leitura, quanto no caso de uma escrita.
- Quando uma operação a ser executada precisa ler ou escrever dados na memória, processador coloca no MAR o endereço correspondente.

Este valor é transmitido pelo BE (Barramento de Endereço) para o controlador da memória.

O número de bits do MAR deve corresponder ao número de bits do Barramento de Endereço

Ou seja, ele deve um número suficiente de bits para endereçar todas as células da memória.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Registrador de Memória MDR (Memory Data Register)

PUC Minas

Quando o processador deseja escrever uma palavra na memória, ele a coloca no MDR.

Após o sinal de escrita ser colocado no barramento, conteúdo do MDR é transmitido através do BD (Barramento de Dados).

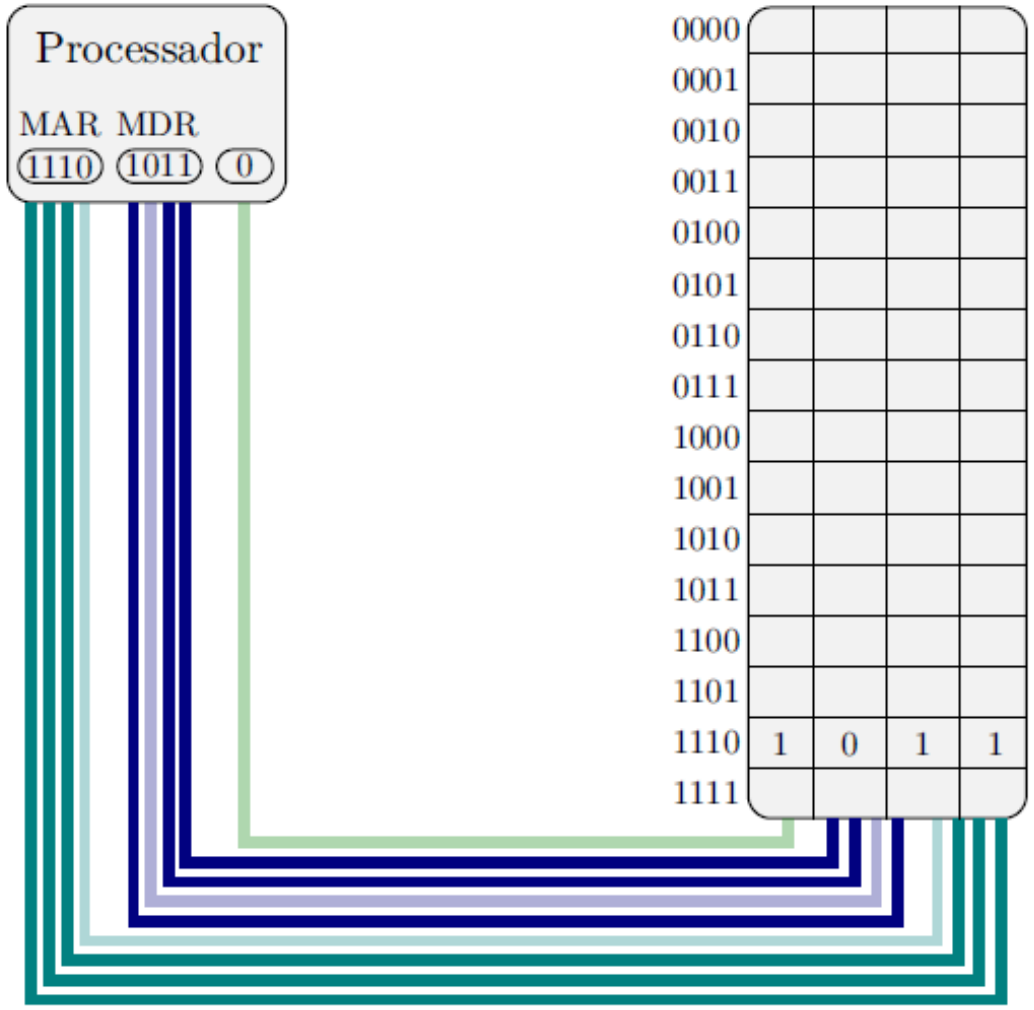
O que a MP escreve no BD, chega ao MDR no processador.

- Uma vez lá, processador pode facilmente acessar o dado.

O MDR deve ter o mesmo número de bits da palavra do processador (mesmo do Barramento de Dados).

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Exemplo de Leitura na Memória Principal

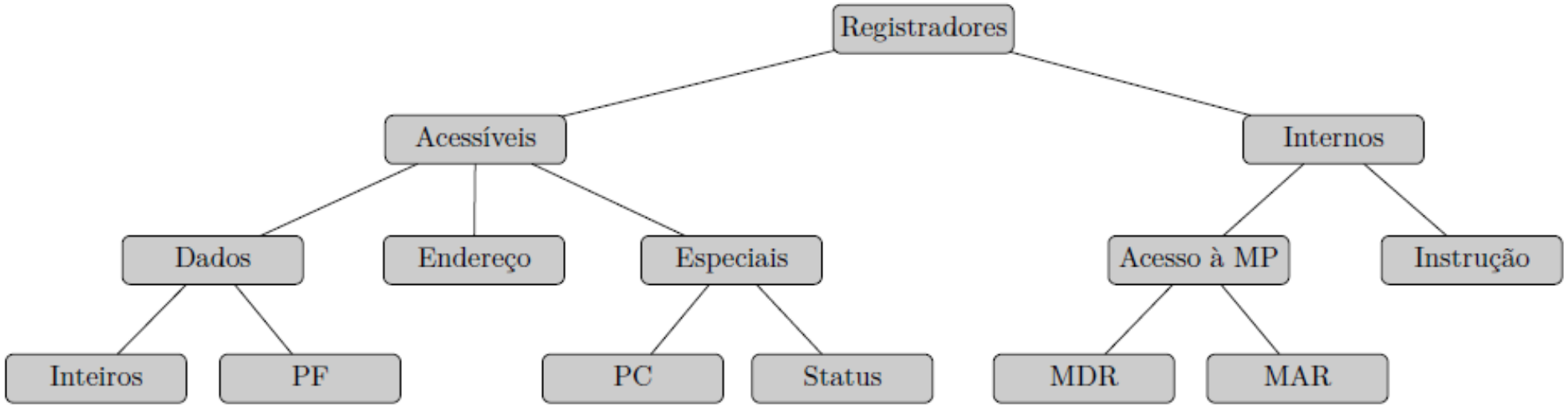


- MAR e MDR estão associados ao BE e BD, respectivamente.
- No exemplo, UCP coloca endereço no MAR.
- Após o sinal de leitura no barramento, memória transmite o valor a ser lido.
- Valor é colocado no MDR.
  - UCP pode facilmente acessá-lo.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Registradores – Resumo de Tipos

PUC Minas



Lembre-se que arquiteturas diferentes possuem registradores diferentes. E que também pode variar de acordo com a tecnologia da época.

Fonte: https://sites.google.com/a/ic.uff.br/fernanda/2017-2/fac

Exemplo

PUC Minas

- Se um computador tem uma MP com total de armazenamento de 2^{16} bits e **possui barramento de dados** com **tamanho de 16 bits**, qual o tamanho mínimo do MAR e do MDR ? (Considere que a barra de dado tem o tamanho de uma palavra)

Exemplo

PUC Minas

Tam(MDR) = 16 bits

Capacidade = Qtde * Tam

2^{16} = Qtde * 2^4

Qtde = 2^{16} / 2^4

Qtde = 2^{12}

Tam(MAR) = 12 bits

Tamanho da palavra igual ao barramento de dados

$2^4 = 16\ bits$

Fórmula:

T=N × M (em bits)

Fonte:

Exemplo

Tam(MDR) = 16 bits

Capacidade = Qtde * Tam

2^{16} = Qtde * 2^{4}

Qtde = 2^{16} / 2^{4}

Qtde = 2^{12}

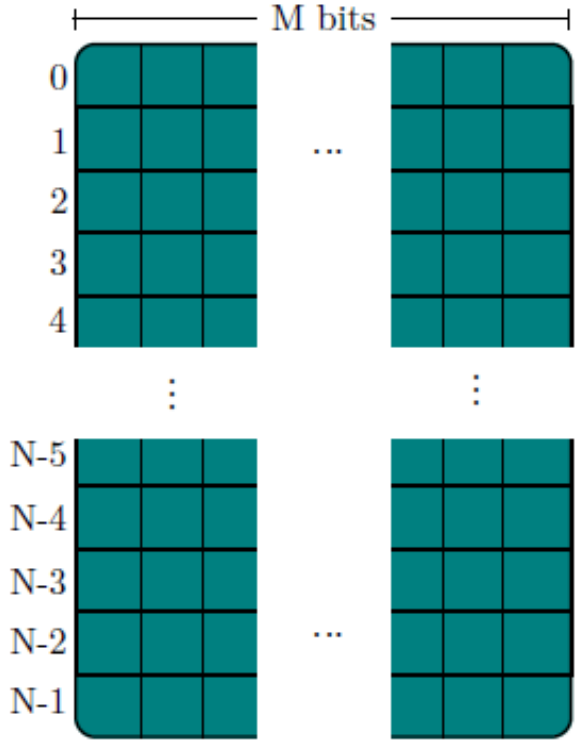
Tam(MAR) = 12 bits

Mesma Fórmula:

T=N × M (em bits)

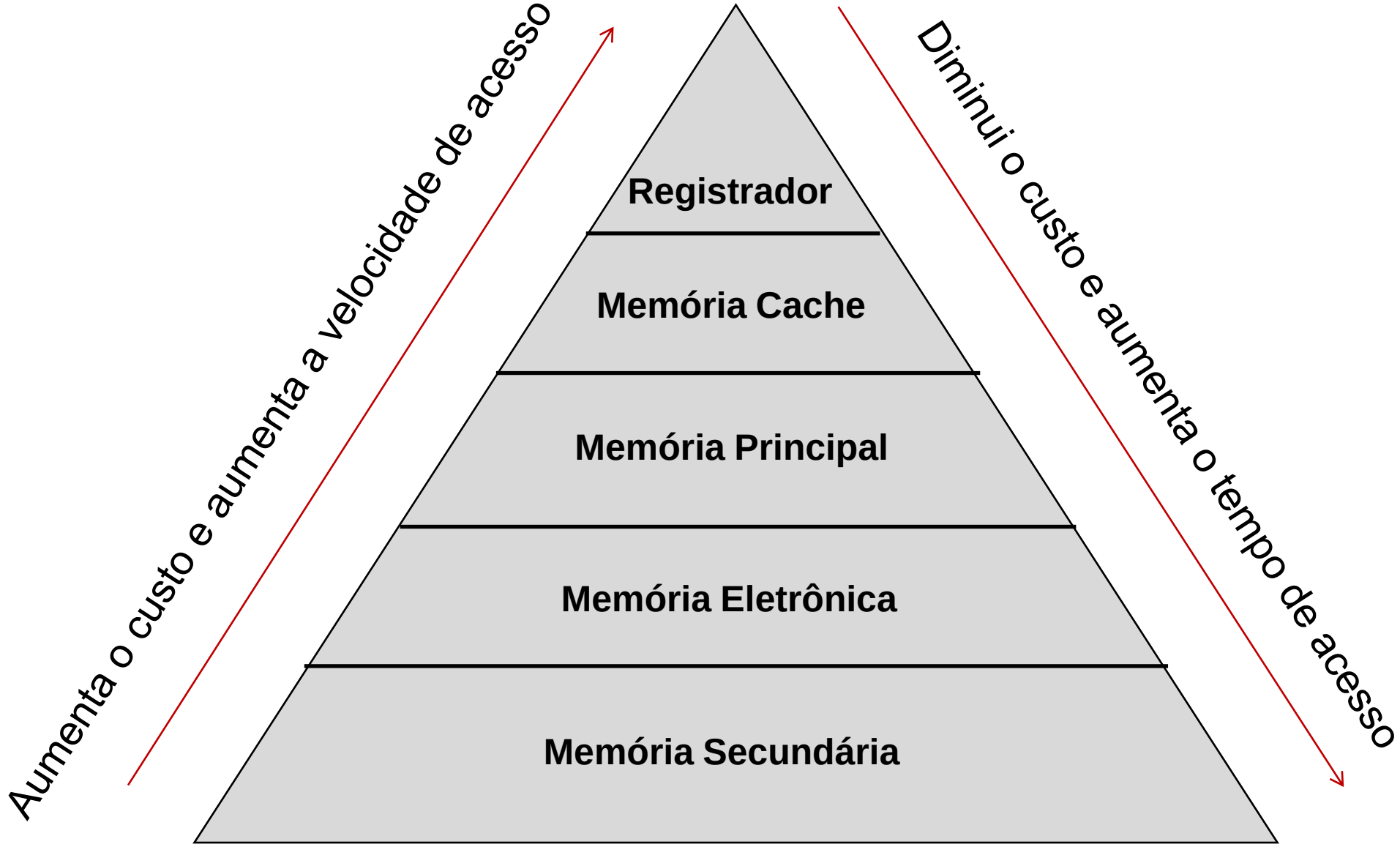
Onde

***N* linhas (células) e *M* colunas (bits)**



Fonte:

Hierarquia de Memória



Referências

PUC Minas

- STUART, Brian L.. Princípios de sistema operacionais: projetos e aplicações. São Paulo: Cenage Learning, 2011. 637 p. ISBN: 9788522107339.
- TANENBAUM, Andrew S. Sistemas Operacionais Modernos. 4a ed. São Paulo: Pearson, 2016. 864 p. ISBN: 9788543005676.
- MACHADO, Francis Berenger; MAIA, Luiz Paulo. Arquitetura de sistemas operacionais. 5a ed. Rio de Janeiro: LTC, 2013. 263 p. ISBN:9788521622109.
- SILBERSCHATZ, Abraham. Fundamentos de Sistemas Operacionais. 9ª ed. LTC, 2015. 524 p. ISBN: 9788521629399. ISBN digital 9788521630012.