# Gerenciamento de memória

## **M**emória

Uma **memória**, em computação é qualquer dispositivo que permite a um computador guardar dados(ou programas) temporariamente ou permanentemente.

### Tipos de memória

Existem 4 tipos de memória:

#### Registradores

São memórias de alta velocidade e pouca capacidade de armazenamento do processador que guardam dados para uso imediato.

#### Cache

É uma memória que faz cache de dados que foram processados e poderão ser reusados no futuro, existem vários tipos de cache, as mais comuns são **L1, L2 e L3, essas memórias são diferenciadas por sua capacidade de armazenamento e latência** (nanossegundos ou clocks da CPU em que ocorre a transferência dos dados).

É possível que ocorra cache flush, quando dados que estavam no cache são transferidos para a RAM para que possam ser armazenados de maneira definitiva.

#### Memória RAM (principal)

Pode ser **SIMM (single in-line memory module) ou DIMM(Double in-line memory module)** dependendo se os chips de memória são soldados de um só lado ou dos dois. Pode ser **SDRAM (Synchronous Dynamic Random Access Memory)** se trabalha de acordo com os clocks da CPU ou **DRAM (Dynamic Random Access Memory)** se não trabalha.

Ainda é classificada por **DDR (Double Date Rate)**, o DDR define quantos dados podem ser enviados em uma transmissão, DDR1 pode transferir 2x mais dados, DDR2 4x mais e assim vai.

#### Memória secundária (HD)

Expansão da memória secundária que é mais barata, possui mais capacidade de armazenamento e é mais lenta, tendo a latência medida em ms o que é 1mi de vezes mais lento que memórias cache.

## **Gerência de memória**

Para sua execução, todo programa precisa receber memória “infinita”, de maneira rápida, confiável e a um baixo custo, a função do **SO é abstrair os tipos e a latência das memórias e gerenciar essa abstração.**

As principais **funções de um gerenciador de memória são**: alocar memória entre os processos, liberar memória quando um processo termina e gerenciar o swapping.

As **premissas básicas** do ger. de memória são: otimizar a utilização da memória principal e garantir a segurança dos dados, isso era simples em SOs que só rodavam um programa já que bastava definir quais endereços o programa em execução poderia usar e gerenciá-los.

### **Formas de gerenciar a memória**

Em sistemas multiprogramados pode ocorrer de um programa acessar os dados do outro ou uma colisão de endereçamentos. Para resolver esses problemas existem algumas estratégias aplicadas a alocação de memória, elas são:

#### **A**locação dinâmica

Soluciona os dois problemas, na alocação dinâmica é criado um container de endereçamento para cada processo. A memória era dividida em vários pedaços iguais e cada pedaço pertencia a um processo.

Do ponto de vista do programa seu container de endereçamento era toda a memória. Cada container continha seu endereçamento de 0 até n, de modo que um endereço no container A continha dados diferentes no container B,



Os espaços são criados via hardware, existem **dois registradores**, um **registrador base** com todos endereços ocupados até a criação do processo e um **registrador limite** que define o endereço máximo que o processo vai ocupar.

Os **processos podem somar seus endereços ao registrador base e verificar se a soma ultrapassa o registrador limite.** Somente o SO pode atribuir valores para os registradores.

O problema com essa solução são os limites de memória fixos e iguais dos espaços de endereçamento, causando fragmentação da memória e limitando seu uso entre os processos ativos já que nem todo programa vai usar todo espaço disponível no container o tempo todo.

### **Paginação**

É comum que falte memória física para a demanda dos processos ativos, para contornar o problema foi criada a memória virtual, que é basicamente usar parte da memória secundária para guardar processos de maneira temporária e trocar (swapping) os programas entre a memória principal e a secundária de acordo com a necessidade.

Quando usando paginação a memória física e a memória virtual são combinadas em uma única memória, o endereçamento dessa memória é dividido em blocos com uma parte dos endereçamentos de x até x chamados de páginas.

**Cada endereço mapeado em uma página que não faz parte da memória virtual equivale diretamente a um endereço na memória física principal**. A estrutura responsável por mapear endereços físicos diretamente para endereços de páginas é chamada **tabela de páginas**.

Os endereços que um programa podem referenciar estão contidos numa estrutura chamada **espaço de endereçamento virtua**l e os **endereços que o programa ocupa** são contidos no espaço de endereçamento físico.

Quando um programa solicita dados que estão memória secundária esses dados são movidos para a memória principal e endereçados.

##### Memory Management Unit – MMU

Quando um programa solicita dados em um endereçamento, ele está solicitando dados em um endereçamento virtual, que precisa ser traduzido em um endereço físico. Essa tradução ocorre com muita frequência, e seriam gastos muitos recursos para fazê-la, é por isso que existe o MMU.

O MMU é responsável por mapear de maneira rápido endereçamentos virtuais em endereçamentos físicos. Isso é feito pelo uso do mecanismo Dynamic Address Translation – DAT.

#### Segmentação

A segmentação parte do princípio que um programa pode ser dividido em dados e instruções, e que essas partes do programa podem ser armazenadas em blocos, chamados segmentos.

Segmentos **podem ter tamanhos diferentes** e **não precisam ocupar espaço de maneira sequencial** na memória principal, de modo que **um segmento ocupará qualquer área na memória que for grande para contê-lo.**

Somente os **segmentos necessários para execução são mantidos em memória**, quando um programa referência um segmento que não existe na memória o gerenciador deve recuperar o segmento solicitado.

A tradução de endereços de segmentação pode ocorrer de diversas maneiras, dependendo de como o endereço virtual é mapeado para o endereço físico, mas uma boa estratégia é usar paginação.