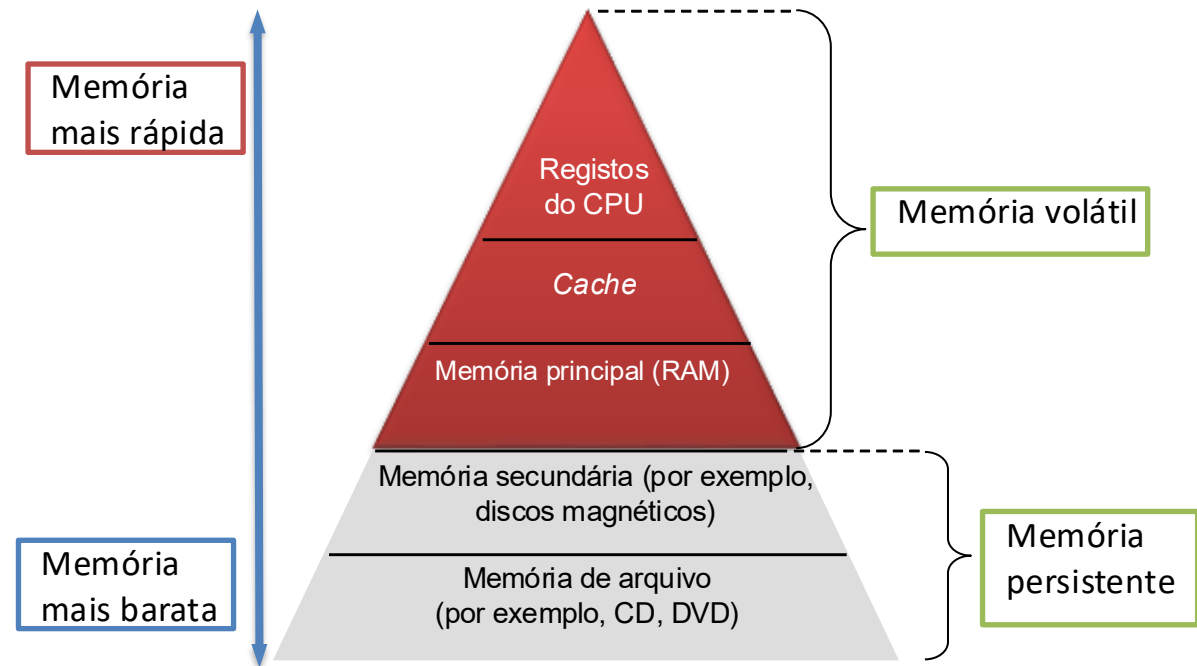


# Objetivos da Gestão de Memória

# Hierarquia de Memória

- A hierarquia de memória representa normalmente a relação entre memória de acesso rápido, mas de custo elevado versus memórias de custo muito mais reduzido, mas mais lentas
- Outras características importantes:
  - Persistência
  - Acesso aleatório vs por blocos
- Tirar partido destas diferenças é um exercício de engenharia e constitui um dos problemas que os sistemas operativos tentam resolver



# Gestão de Memória

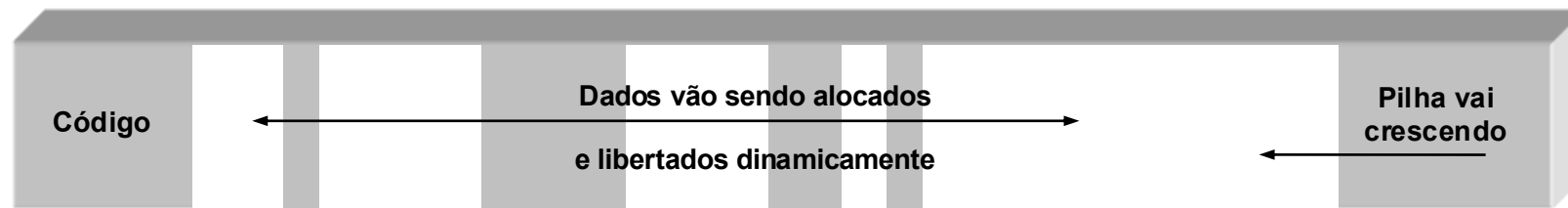
## Objetivos do Sistema Operativo

- Gerir o espaço de endereçamento dos processos
  - Assegurar que cada processo dispõe da memória que necessita
  - Garantir que cada processo só acede à memória a que tem direito (proteção)
  - Optimizar o desempenho dos acessos a memória, tendo em conta os mecanismos necessários aos requisitos acima

# Espaço de Endereçamento - Utilizador

- Espaço de endereçamento de um processo é o conjunto de posições de memória que o respetivo programa pode referenciar

Espaço de endereçamento do processo em modo utilizador  
(blocos cinzentos estão ocupados; blocos brancos estão livres)



- Código e bibliotecas
- Dados inicializados ou dados declarados estaticamente
- *Heap* - dados alocados dinamicamente
- Zonas de memória partilhadas
- Bibliotecas partilhadas
- Pilha utilizador (*stack*)

# Dimensão do Espaço de Endereçamento

- Gostaríamos que os processos tivessem toda a memória que necessitam, ou seja, que os programadores se pudessem abstrair da realidade de quanta memória física está disponível
- O espaço de endereçamento está sempre limitado pelo endereçamento do processador
  - Processadores de 32 bits –  $2^{32}$  bytes = 4 Gbytes (4 000 000 000)
  - Processadores de 64 bits -  $2^{64}$  bytes 16 Exabytes ( $16 * 10^{18}$ )
  - Mas é evidente que não é esta a limitação, nas arquiteturas mais recentes

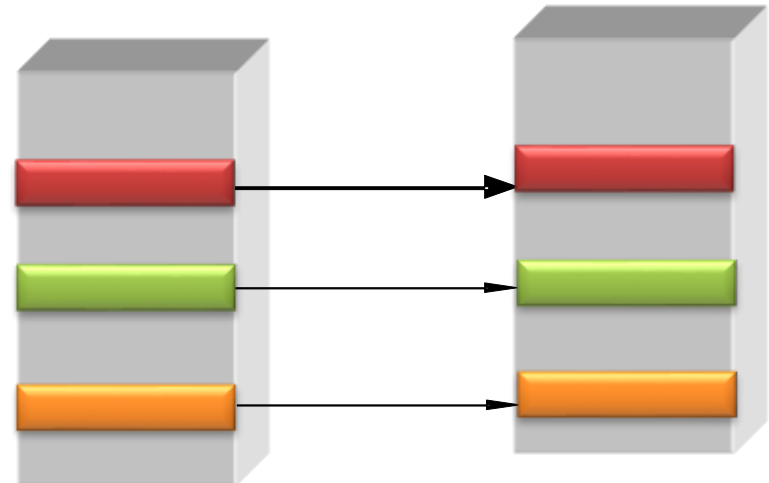
# Endereçamento Real

- Importante perceber como os endereços existentes nos programas se relacionam com os endereços na memória
- Primeira solução mais simples, o endereço indicado no programa é aquele que é acedido na memória principal
- Os mapas de memória são iguais e os endereços no programa são os mesmos que na memória primária

## Mapeamento direto

Espaço de endereçamento visto pelo programa

Memória Primária



endereço → valor

# Endereçamento Real

## Limitações?

- Dimensão dos programas?
- Multiprogramação?
- Capacidade de ser executado em diferentes máquinas?
- A multiprogramação seria impossível se não estiverem vários programas/processos carregados simultaneamente na memória primária
- Sem multiprogramação não há processos...

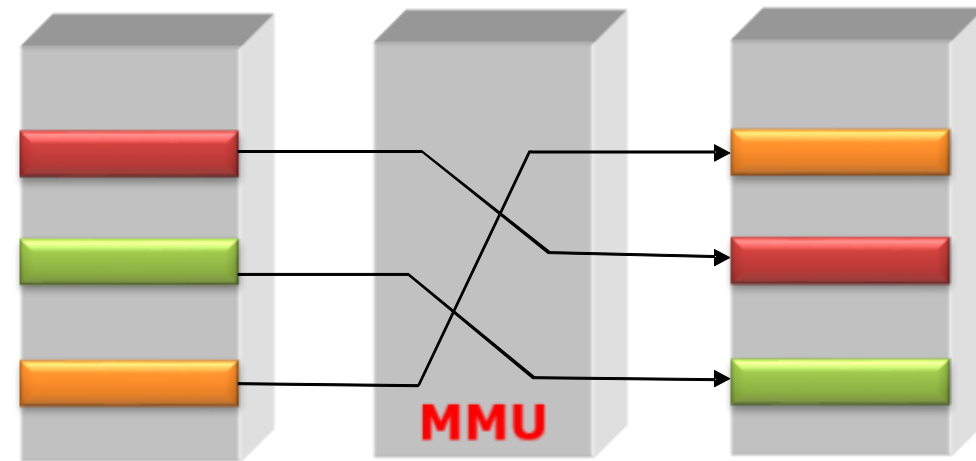
# Endereçamento Virtual

## Eliminar a restrição

- Espaço de endereçamento dos processos não tem relação com a memória física
- Os programas usam endereços virtuais que em tempo de execução são traduzidos para os endereços reais
- Endereços virtuais são convertidos pela Unidade de Gestão de Memória (MMU – *Memory Management Unit*) que faz parte do processador

Espaço de endereçamento visto pelo programa

Memória Primária

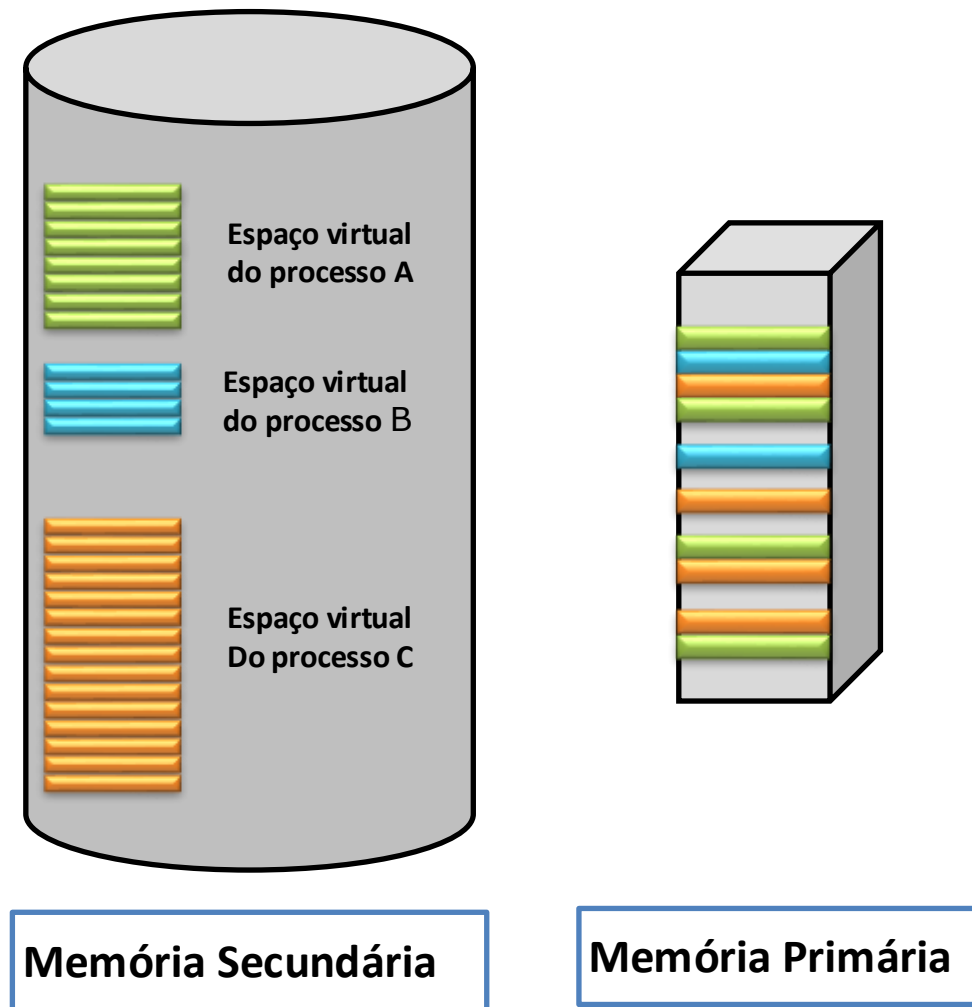


endereço virtual → *MMU* → endereço físico → valor



# Espaço de Endereçamento Virtual

- **Nova abstração:** espaço de endereçamento virtual
- O sistema operativo cria um espaço de endereçamento virtual guardando o espaço de endereçamento em memória de massa e só colocando na memória primária uma pequena fração estritamente necessária
- Como é possível? **Pelo princípio da localidade temporal e espacial da execução dos programas** que já justificou as *caches* de hardware



# Endereçamento Virtual

- Um endereço virtual não se refere nem à memória primária nem à memória secundária, mas precisamente a uma memória virtual
- É função do *hardware* de gestão de memória e do sistema operativo fazer a correspondência entre o endereço virtual e o endereço físico.
- O *byte* referenciado pelo endereço virtual pode estar em memória primária ou em memória secundária
- Se estiver em memória primária
  - a unidade de gestão de memória traduz automaticamente o endereço virtual num endereço físico e o *byte* é acedido
- Se estiver em memória secundária
  - a unidade de gestão de memória avisa o sistema operativo para este o carregar/alocar em memória primária

# Proteção do Espaço de Endereçamento

Endereço  $\rightarrow$  valor ( $\forall$  *endereço*  $\in$  *espaço de endereçamento do processo*)

- A proteção é fazer cumprir esta restrição, testando se o endereço do programa pertence ao espaço de endereçamento atual do processo
- Na proteção há ainda que assegurar que o processo tem os habituais direitos: leitura, escrita e execução quando pretende executar uma destas operações
- Se um programa referenciar uma posição de memória fora do espaço de endereçamento do processo ou a que não tem o direito de acesso correspondente, o hardware de gestão de memória provoca uma exceção que chama o sistema operativo causando, se não for possível tratá-la, a terminação do processo
- O espaço de endereçamento em modo núcleo, permite endereçar todas as posições de memória

# Desafios

- Como construir o endereçamento virtual de forma a não introduzir grande impacto no desempenho do ciclo de endereçamento
- Como gerir a transferência de blocos da memória de massa para memória primária e vice-versa
- Como suportar a dinâmica da criação dos processos e das alterações aos seus espaços de endereçamento
- Como garantir a proteção de memória

# Resumo

- O Sistema Operativo considera que cada processo tem um espaço de endereçamento virtual. Os compiladores geram os programas para esse espaço virtual
- O espaço de endereçamento virtual tem endereços diferentes dos endereços físicos e que tem de ser transformados na execução.
- O espaço de endereçamento virtual é composto por blocos na memória de massa, e por blocos na memória principal estando em memória principal os necessários para que a execução possa avançar
- O ciclo de endereçamento é executado pelo hardware, mas o sistema operativo gere todas as operações que permitem transparentemente criar, modificar, herdar/partilhar os espaços de endereçamento virtuais