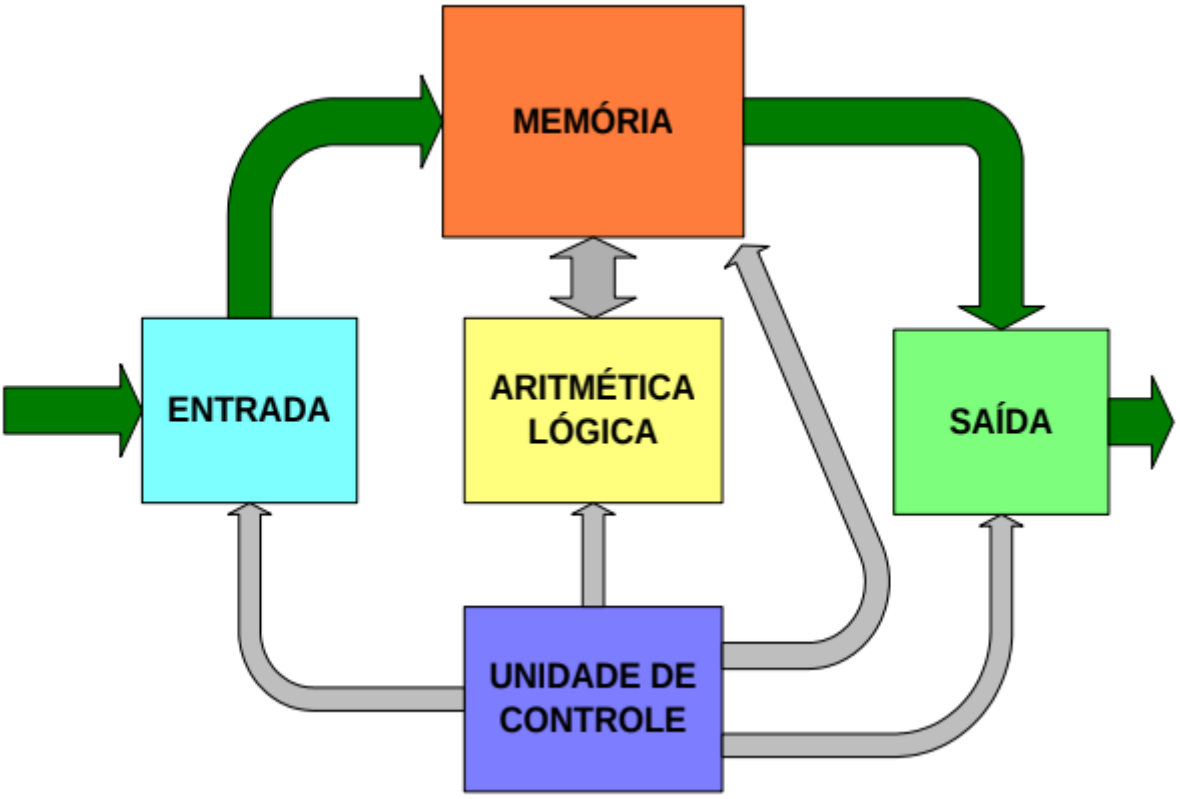
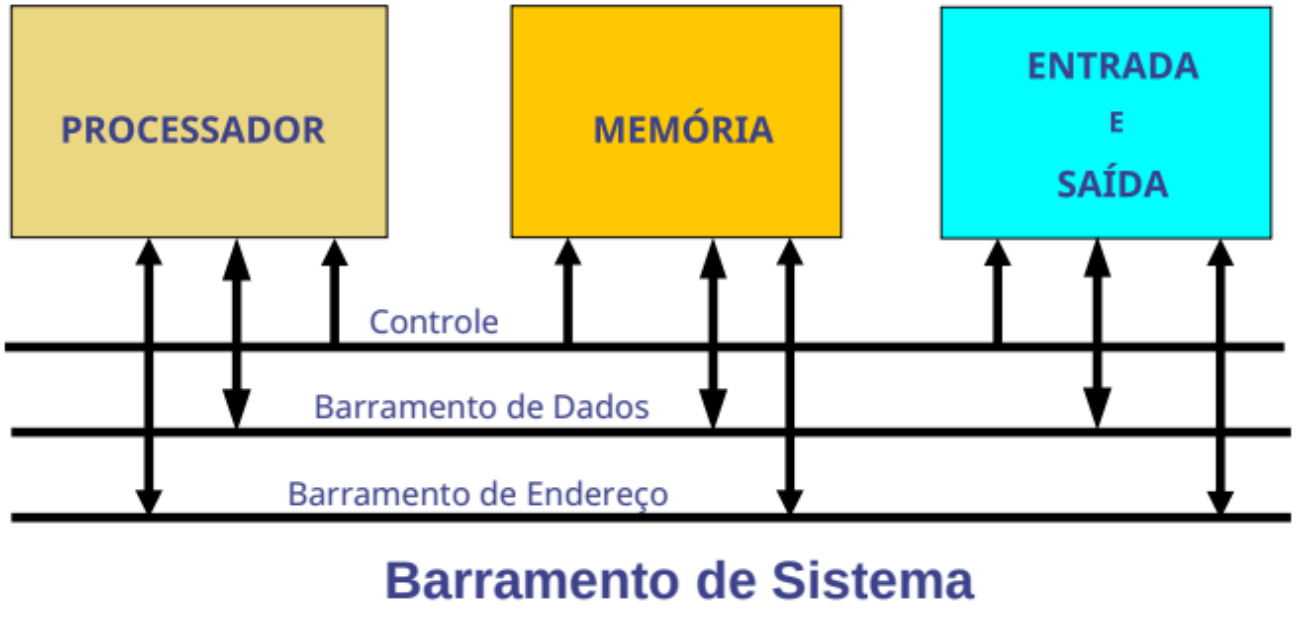
Resumo ArqCompSO

Aula 1

Modelo Von Neumann



Modelo de Barramento



Memória Principal e Secundária

- Memória Principal

    - Volátil (RAM)
      As informações armazenadas na memória volátil podem ser alteradas durante a execução de um programa. São também usadas para armazenar os resultados intermediários e finais das operações realizadas pelo processador.
    - Não Volátil (BIOS)
      A memória não volátil é usada para armazenar informações que não necessitam ser alteradas no decorrer do processamento. É utilizada para iniciar o funcionamento do computador, realizando os testes iniciais e cópia do sistema operacional para a memória.
- Memória Secundária
  A memória secundária é onde os programas e dados, incluindo aqueles do sistema operacional, são armazenados de uma forma persistente no computador. Hoje em dia é constituída, principalmente, pelo conjunto de discos magnéticos (HDs) do computador e também, cada vez mais, pelos discos de estado sólido (SSDs).

A principal característica da memória secundária é o armazenamento da informação de uma forma permanente, mesmo quando o computador é desligado.

Uma das características da memória secundária é o alto volume de dados e o baixo custo de armazenamento por byte quando comparado com a memória principal.

Entrada/Saída

A unidade de entrada e saída é necessária para prover a comunicação entre os dispositivos de ENTRADA e SAÍDA com as demais partes
do computador.

- Toda a informação é convertida de/para o formato binário pela unidade de entrada/saída. Exemplos de dispositivos de entrada/saída: são o disco rígido, teclado, terminal de vídeo, mouse, impressora, entre outros.

Processador

A UPC (Unidade de Processamento Central - CPU) é o conjunto da unidade lógica e aritmética, registradores e da unidade de controle.

Sua função é executar os programas armazenados na memória principal, buscando suas instruções, examinando as, e então executando uma após a outra.
O processador é responsável pela realização de uma série de funções:

- Busca de instruções e dados na memória.
- Programa a transferência de dados entre a memória e os dispositivos de entrada/saída.
- Decodifica as instruções.
- Realiza as operações lógica e aritméticas.
- Responde a sinais enviados por dispositivos de entrada/saída como RESET ou interrupções.

Arquitetura do Processador

Unidade Aritmética e Lógica (UAL)

A largura da arquitetura de um processador (8, 16, 32 ou 64 bits) é definida pela largura em bits do maior operando inteiro que pode ser utilizado em uma única operação pela UAL.

Como consequência direta, a largura em bits do maior operando admitido pela UAL irá determinar, normalmente, a largura em bits do acumulador e dos registradores de uso geral do processador. Não há sentido para que sejam maiores ou menores do que isso.

Registradores

O processador contém elementos de memória, de pequena capacidade mas de alta velocidade, usados para armazenar resultados temporários, chamados de **registradores**.
O conjunto desses registradores é denominado banco de **registradores**.
Existe um registrador invisível ao programador, chamado de registrador de instrução (**RI**), que armazena a instrução que está sendo executada.
Existe um registrador especial denominado apontador de instruções (**PC**) , que contém o endereço da próxima instrução que vai ser executada.

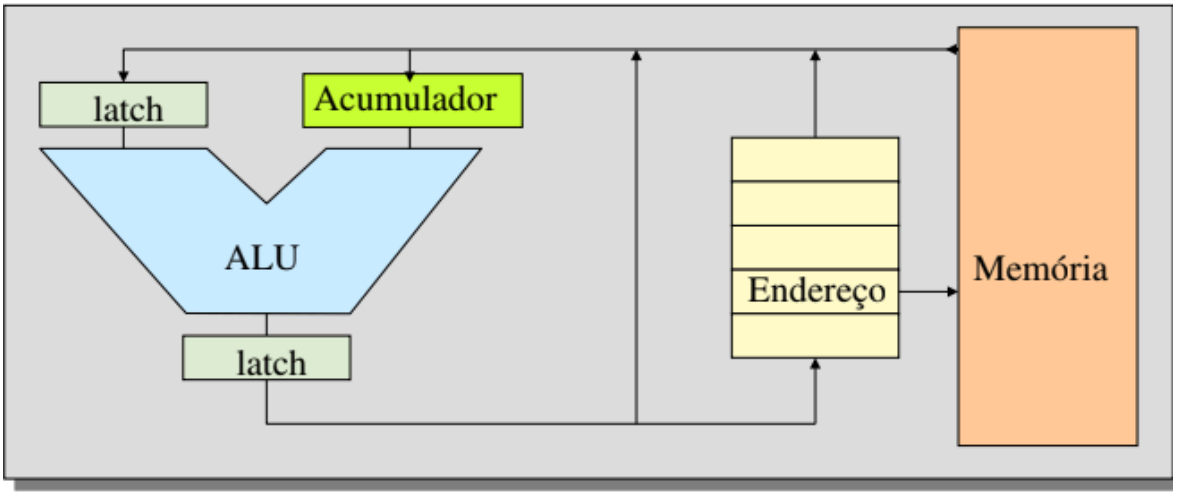
Unidade de Controle

A unidade de controle é responsável pela coordenação da atividade de todos os componentes do processador.

- Ela busca a instrução na memória e coloca no registrador de instruções (RI).
  A unidade de controle faz a decodificação da instrução que está no RI:
- Determina qual o tipo de operação vai ser realizada pela UAL
- Determina quantos e quais são os operandos de leitura, e qual o registrador de destino, se houver.
- Lê os operandos necessários para a execução da instrução e os coloca na entrada da UAL.
  A unidade de controle lê o resultado da saída da UAL e envia para o destino correto.
  Há duas formas de se implementar a unidade de controle:
- Através de microprogramação
  Arquiteturas do tipo **CISC** - (Complex Instruction Set Computers)
- Controle direto pelo hardware (PLA, ROM)
  Arquiteturas do tipo **RISC** - (Reduced Instruction Set Computers).

Tipos de Arquitetura

- Arquitetura de Acumulador
  - Um operando (em registrador ou memória), o acumulador é usado como operando implícito a maioria das vezes

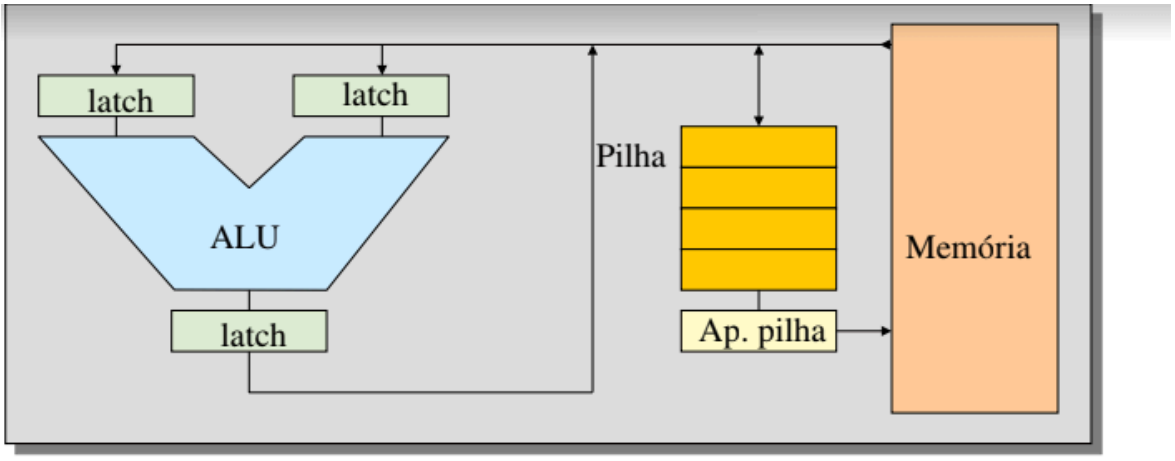


Código exemplo: `c = b+a;`

```
load  a;    // acumulador é operando implícito
add   b;
store c;
```

34

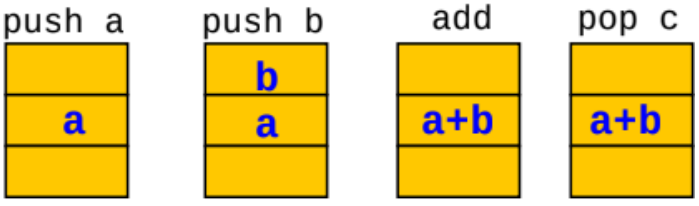
- Arquitetura de Pilha
  - Nenhum operando: todos operandos são implícitos no topo da pilha



Exemplo: `c = b+a;`

```
push a;
push b;
add  ;
pop  c;
```

Pilha:



- Arquitetura de Registrador (load / store)
  - Três operandos, todos nos registradores
  - loads e stores são as únicas instruções que fazem acesso à memória
- Arquitetura Registrador-Memória
  - Dois operandos, um em memória
- Arquitetura Memória-Memória
  - Três operandos, podem todos estar na memória

Modos de Enderençamento

Modo	Exemplo	Significado (RTL)
Imediato	`add r4, r4, #3`	R4 ← R4 + 3
Registrador	`add r4, r4, r3`	R4 ← R4 + R3
Direto ou Absoluto	`add r1, (1001)`	R1 ← R1 + M[1001]
Indireto Reg.	`add r4, (r1)`	R4 ← R4 + M[R1]
Deslocamento	`ld r4, 100(r1)`	R4 ← MEM[100 + R1]
Indexado	`add r3, (r1+r2)`	R3 ← R3 + M[R1 + R2]
Indireto Mem.	`add r1, @(r3)`	R1 ← R1 + M[M[R3]]
Pilha	`pop r1`	R1 ← M[SP]

Sinal de Relógio (Clock)

O processador tem seu funcionamento sincronizado por um sinal elétrico periódico denominado relógio. O relógio cadencia a execução das instruções em suas diversas fases. *Quanto mais rápido (maior a frequência)* for o sinal de relógio, mais rápido as instruções, e por consequência os programas, serão executados. O atraso dos componentes básicos do processador (portas lógicas, flip-flops, etc.) limitam a frequência máxima que o relógio pode ter.

A frequência e o tempo do ciclo do relógio estão relacionados pela seguinte equação:

$$T_c = \frac{1}{f}$$

Quanto maior a frequência de relógio maior é o consumo de energia e dissipação de calor do processador.
A dissipação de calor, além do atraso dos componentes, também impõe limites práticos sobre a maior frequência que um processador pode ter.

Iniciando um Computador

BIOS - Basic Input-Output System

Responsável por ativar os componentes de hardware do seu computador, garantir que eles estejam funcionando corretamente e, em seguida, executar o gerenciador de partida que vai iniciar o sistema operacional que você tenha instalado. Então, quando você salva uma configuração, ela é armazenada em uma pequena memória CMOS, que é alimentada por uma bateria à parte, da própria placa-mãe e permanece ativa enquanto essa bateria estiver com carga. Essa bateria também é responsável por guardar a hora do computador atualizada, alimentando o RTC (Real Time Clock) quando computador é desligado. Quando você liga o computador, a BIOS irá testar e configurar o seu computador e recuperar a hora atual a partir dessas configurações salvas.

Esse teste inicial é conhecido pelo nome de POST (Power- On Self Test), e serve para verificar diversos componentes, tais como: fonte de alimentação; adaptador de vídeo; memória principal (RAM); temporizador; teclado e mouse; etc.
Após esses testes iniciais, se tudo estiver em ordem, o dispositivo de boot, que pode ser um disco rígido, um pendrive ou mesmo a ethernet, deve ser acessado para que o processo de carga do sistema operacional seja iniciada.

UEFI

A UEFI substitui o BIOS tradicional nos computadores pessoais e não há como mudar de BIOS para UEFI em um computador já existente.
Ao invés do MBR, o UEFI utiliza uma nova de particionamento do disco, chamada de GUID Partition Table (GPT) que permite superar muitas limitações do antigo padrão BIOS/MBR, com partições maiores e redundância para a tabela de partição.
Em síntese, o UEFI é essencialmente um mini sistema operacional executando direto no firmware do processador, podendo ser carregado da memória FLASH da placa mãe, carregado do disco rígido ou mesmo através da rede.
O UEFI também definiu um formato padrão para os seus programas executáveis, além de definir uma extensão do formato FAT32 para ser utilizado nas partições que armazenam esses programas.
Ou seja, o UEFI carrega programas executáveis, compilados com um formato definido na especificação do padrão, que estão armazenados em partições de sistema destinadas exclusivamente para o UEFI, formatadas também com um padrão descrito na sua especificação do padrão.
O UEFI possui também um modo de compatibilidade com o padrão BIOS, configurável na interface de usuário.

Aula 2

Arquiteturas CISC x RISC

O tempo de execução de um programa pode ser definido pela seguinte equação:

$$T_p = C_i \times T_c \times N_i$$

Onde:

- T_p = tempo de execução do programa
- C_i = ciclos por instrução
- T_c = tempo de cada ciclo
- N_i = número de instruções

Exemplos de arquitetura CISC eram então os processadores x86 da Intel.
Já os processadores SPARC, MIPS e ARM são exemplos de arquiteturas RISC.

CISC (Memória-Memória)

Características:

- Instruções complexas demandando um número grande e variável de ciclos de máquina para sua
  execução.
- Uso de diversos modos de endereçamento de operandos.
- Instruções com formato muito variável.
- Diferentes tipos de instruções podem referenciar operandos na memória principal.
- Cada fase do processamento da instrução pode ter duração variável em função da complexidade.

Consequências:

- Implementação com uso de pipeline é difícil.
- A taxa média de execução das instruções por ciclo tende a ser bastante superior a 1 CPI.
- A unidade de controle é em geral microprogramada.
- Códigos compactos podem ser gerados pelos compiladores.

RISC (Registrador)

Características:

- Instruções mais simples demandando um número fixo de ciclos de máquina para sua execução;
- Uso de poucos modos simples de endereçamento de operandos;
- Poucos formatos diferentes de instruções
- Apenas as instruções de “load” e “store” referenciam operandos na memória principal;
- Cada fase de processamento da instrução tem a duração fixa igual a um ciclo de máquina.

Consequências:

- Implementadas com o uso do pipeline;
- A taxa média de execução de instruções por ciclo de máquina é próxima de 1 CPI;
- A unidade de controle é em geral “hardwired”;
- Processo de compilação é complexo e requer cuidados especiais para otimização do desempenho do código gerado.

Aula 3

Processadores

O que é?

- O *microprocessador*, ou comumente chamado de processador;
- É uma espécie de microchip especializado;

- Um circuito integrado que realiza as funções de cálculo e tomada de decisão de um computador, parecida com a função cérebro humano;
- Também pode ser chamado de *Unidade Central de Processamento* (UCP) (Em inglês *CPU*: Central Processing Unit);

Função

- Realiza cálculos de *operações aritméticas* e *comparações lógicas*;
- Mantem o funcionamento de todos os equipamentos e programas, pois a unidade de controle interpreta e *gerencia a execução de cada instrução do programa*;
- Administra na memória central (principal) além do programa submetido, os dados transferidos de um elemento ao outro da máquina visando o seu processamento;
- Recebe dados e comandos do usuário administra-as e as processa de acordo com as instruções armazenadas em sua memória, e fornece resultados como saída;
- Microprocessadores operam com números e símbolos representados no sistema binário;
- Ele também transmite estas informações para a placa mãe, que por sua vez as transmite para onde é necessário (como o monitor, impressora, outros dispositivos). A placa mãe serve de ponte entre o processador e os outros componentes de hardware da máquina.

Características

- Processadores geralmente possuem uma pequena memória interna, portas de entrada e de saída, e são geralmente ligados a outros circuitos digitais como memórias; multiplexadores e circuitos lógicos;
- Muitas vezes também um processador possui uma porta de entrada de instruções, que determinam a tarefa a ser realizada por ele. Estas sequências de instruções geralmente estão armazenadas em memórias, e formam o programa a ser executado pelo processador.

OBS: Os bytes são agrupados em palavras e a maioria das instruções operam sobre palavras. Assim, os registradores da CPU geralmente são do tamanho de uma palavra, então, se for de 32 bits, são 4 células que podem ser operadas a cada instrução. E, o tamanho da palavra, define normalmente a largura do processador.

Unidade de Aritmética e Lógica

Circuito que se encarrega de realizar as operações matemáticas requisitadas por um determinado programa;
A Unidade de Controle é o que há de mais próximo a um cérebro dentro do processador. Esse controlador define o regime de funcionamento e da ordem às diversas tarefas do processador;

Registradores

Os registradores são pequenas memórias velozes que armazenam comandos ou valores que são utilizados no controle e processamento de cada instrução.
Os registradores mais importantes são:

- Contador de Programa (PC) – Sinaliza para a próxima instrução a ser executada;
- Registrador de Instrução (IR) – Registra a execução da instrução;

Unidade Ponto Flutuante

Processadores atuais possuem outra unidade para cálculos, conhecida como Unidade de Ponto Flutuante. Essa, por sua vez, serve para trabalhar com números enormes, de 64, 128 bits, por exemplo;

Unidade de Gerenciamento de Memória

A MMU (em inglês: Memory Management Unit) é um dispositivo de hardware que transforma endereços virtuais em endereços físicos e administra a memória principal do computador.

Aula 4

Memórias

- Visão Geral:
  - Manipula Bit
  - Unidade de informação a ser armazenada, recuperada ou transferida (célula) - Grupo de n bits (n = 8) $\rightarrow$ 1 Byte
  - ENDEREÇO: é o código de identificação da localização das células (informações).
  Operações:
  - ESCRITA: transferência de informações de outro componente do sistema de computação para a memória (CPU $\rightarrow$ memória)
  - LEITURA: transferência de bits da memória para a CPU, disco.
  Em um sistema de computação não é possível construir e utilizar apenas um tipo de memória. Para certas atividades, por exemplo, é fundamental que a transferência de informações seja a mais rápida possível.
  Outras atividades é preferido que os dados sejam armazenados por períodos mais longos.
  Memória de um computador $\rightarrow$ subsistema - construída de vários componentes (vários tipos diferentes de memória) interligados e integrados, com o objetivo de armazenar e recuperar informações.

Tempo de Acesso

Indica quanto tempo a memória gasta para colocar uma informação no barramento de dados após uma determinada posição ter sido endereçada.
É um dos parâmetros que pode medir o desempenho da memória.
Também chamado de *latência*, se mede em números de clock necessários.
Denominação: tempo de acesso para leitura (ou tempo de leitura).

Dependente do modo como o sistema de memória é construído e da velocidade dos seus circuitos.
Memórias eletrônicas - igual, independentemente da distância física entre o local de um acesso e o local do próximo acesso - *acesso aleatório (direto)*.

Dispositivos eletromecânicos (discos, fitas, ..) - tempo de acesso varia conforme a distância física entre dois acessos consecutivos - *acesso sequencial*.

Capacidade

Quantidade de informação que pode ser armazenada em uma
memória;
Unidade de medida mais comum - byte, podem ser usadas outras unidades como células (no caso de memória principal ou cache), setores (no caso de discos) e bits (no caso de registradores).
Dependendo do tamanho da memória, isto é, de sua capacidade, indica-se o valor numérico total de elementos de forma simplificada, através da inclusão de K (kilo), M (mega), G (giga) ou T (tera).

Símbolo	Nome	Valor (em potência de 2)	Valor Decimal
K	Kilo	(2^{10})	1.024
M	Mega	(2^{20})	1.048.576
G	Giga	(2^{30})	1.073.741.824
T	Tera	(2^{40})	1.099.511.627.776
P	Peta	(2^{50})	1.125.899.906.842.624
E	Exa	(2^{60})	1.152.921.504.606.846.976
Z	Zetta	(2^{70})	1.180.591.620.717.411.303.424
Y	Yotta	(2^{80})	1.208.925.819.614.629.174.706.176

Volatilidade

Memórias podem ser do tipo **volátil** ou **não volátil**.

- *Volátil* : Perde a informação armazenada na ausência de energia elétrica. Ex.: Registradores, Memória Principal.
- *Não Volátil*: Retém a informação armazenada quando a energia elétrica é desligada. Ex.: Discos, Fitas.
  É possível manter a energia em uma memória originalmente não volátil - uso de baterias.

Tecnologia de Fabricação

Memórias de meio magnético

Fabricadas de modo a armazenar informações sob a forma de campos magnéticos.
Método de acesso às informações - *seqüencial*.
Exemplos: disquetes, discos rígidos e fitas magnéticas (de carretel ou de cartucho).

Memórias de meio óptico

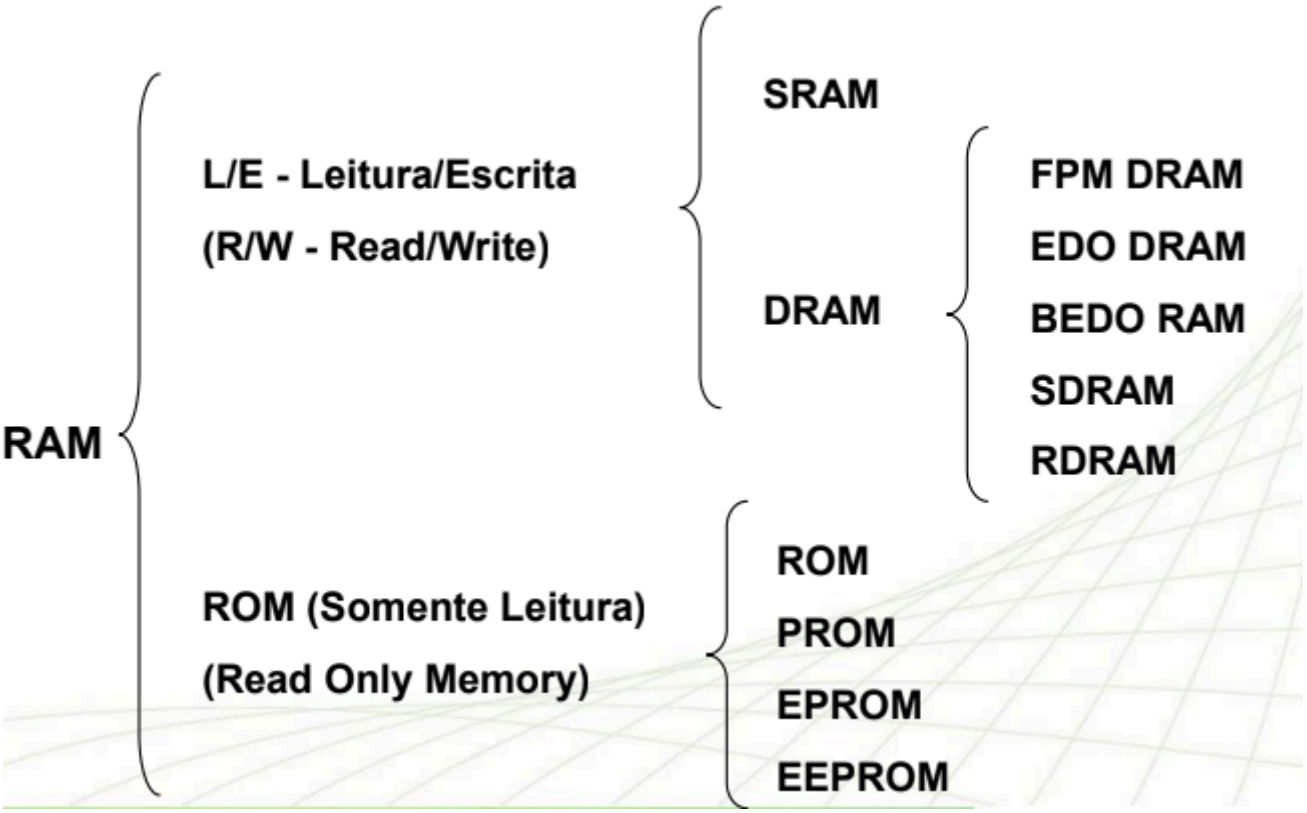
Dispositivos que utilizam um feixe de luz para “marcar” o valor (0 ou 1) de cada dado em sua superfície.

Exemplos:

- CD-ROM (leitura)
- CD-RW (leitura e escrita)

Memórias de semicondutores

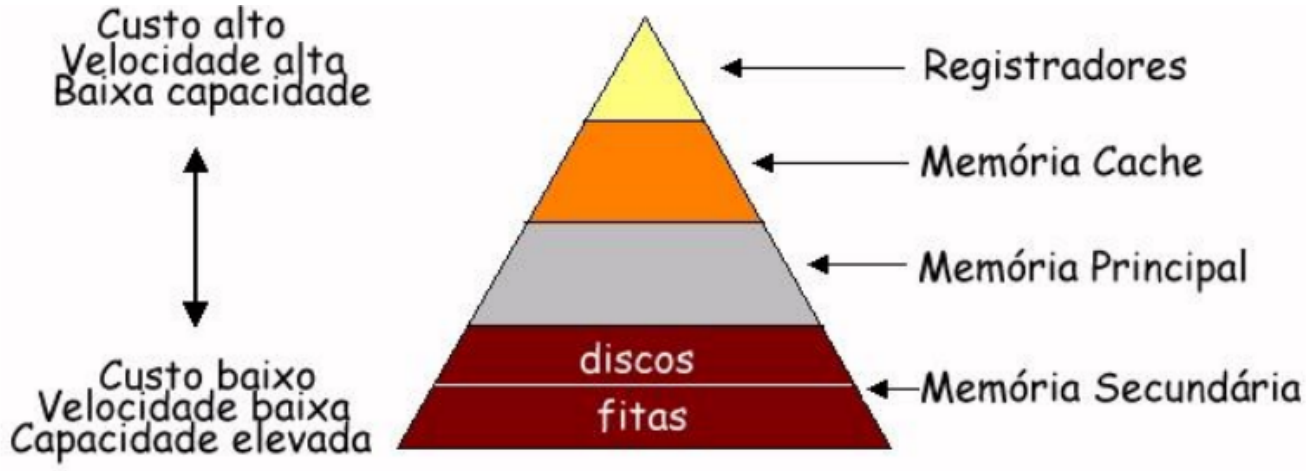
Rápidas e relativamente caras, se comparadas com outros tipos.
Exemplos: Registradores, Memória Principal, Memória Cache e SSD.



- R/W - *Read and Write*
  - Memória de leitura e escrita, de acesso aleatório e volátil.
  - Pode ser estática (SRAM) ou dinâmica (DRAM).
    - SRAM - uso de circuitos transistorizados (flip-flops)
    - DRAM - uso de capacitores (1 transistor e 1 capacitor por bit, não usa flip-flops), necessita de refresh
    - DDR ou SDRAM-II (Double Data Rate SDRAM)
    - RDRAM (Rambus DRAM)
- ROM - *Read Only Memory*
  - Memória apenas de leitura. Uma vez gravada não pode mais ser alterada. De acesso aleatório, não é volátil.
  - Mais lenta que a R/W e mais barata. • Pode ser programada por máscara ("mask programmed“- MROM) em fábrica.
  - Utilizada geralmente para gravar programas que não se deseja permitir que o usuário possa alterar ou apagar (Ex.: o BIOS - Basic Input Output System e Microprogramas de Memórias de Controle).
- PROM - *Programmable Read Only Memory*

    - Memória apenas de leitura, programável.
    - ROM programável com máquinas adequadas (chamadas queimadores de PROM).
    - Geralmente é comprada "virgem" (sem nada gravado), sendo muito utilizada no processo de testar programas no lugar da ROM, ou sempre que se queira produzir ROM em quantidades pequenas.
    - Uma vez programada (em fábrica ou não), não pode mais ser alterada.
- EPROM - *Erasable Programmable Read Only Memory*
    - Memória apenas de leitura, programável (com queimadores de PROM) e apagável (com máquinas adequadas, à base de raios ultra-violeta).
- EEPROM (ou E2PROM) - *Electrically Erasable Programmable Read Only Memory*
    - Memória apenas de leitura, programável e eletronicamente alterável. Também chamada EAROM (Electrically Alterable ROM).
    - EPROM apagável - processo eletrônico, sob controle da UCP (equipamento e programas adequados), menor e mais rápida que a EPROM.
- Flash
    - Funcionamento similar ao da EEPROM – conteúdo total ou parcial da memória pode ser apagado normalmente por um processo de escrita.
    - Apagadas e regravadas por blocos (o apagamento não pode ser efetuado ao nível de byte como na EEPROM), alta capacidade de armazenamento
    - O termo flash foi imaginado devido à elevada velocidade de apagamento dessas memórias em comparação com as antigas EPROM e EEPROM.
- Memória CMOS - (*Complementary Metal Oxide Semiconductor*)
    - Tipo especial de memória para armazenamento das opções essenciais de configuração de inicialização → quantidade de memória instalada, data, hora.
    - Alimentação via bateria.

Hierarquia de Memória



A HIERARQUIA DA MEMÓRIA ESTÁ BASEADA NAS SEGUINTES CARACTERÍSTICAS:

1. Custo
2. Tamanho

3. Velocidade

   Obs: Quanto maior for a velocidade, maior o custo e consequentemente menor o tamanho.

Tipos de Memórias

- Registradores (Internos a CPU)
- Cache
  - São medidas conforme a sua latência e dividem-se em alguns casos em L1, L2 e L3;
  - São dispositivos de armazenamento que seguem uma hierarquia de tamanho, velocidade e custo. Todas são voláteis.
    - *Cache L1* (Primária) - Interna ao processador.
    - *Cache L2* (Secundária) - Atualmente: localizada no interior da pastilha do processador
    - *Cache L3* - localizada externamente ao processador (mas acompanha ele).
  - Quanto mais próxima do processador, melhor será o desempenho do mesmo.
- Memória Principal (RAM)
  - Há normalmente uma pequena quantidade de memória não volátil fazendo parte da memória principal (contém o BIOS).
  - Cada posição da memória principal tem um endereço único
  - Geralmente é combinada com uma memória CACHE menor e mais veloz
  - Endereçamento
    - A memória principal é organizada como um conjunto de células(ou posições) capazes de armazenar, cada uma, 8 bits (1 byte);
    - Existe 1 endereço para cada célula de memória, portanto, a célula é a menor unidade de memória endereçável;
    - Bytes são agrupados em PALAVRAS;
    - A maioria das instruções opera sobre palavras;
    - Registradores da CPU geralmente são do tamanho de uma palavra:
      - 32 bits = 4 células;
      - 64 bits = 8 células;
- Memória Secundária (CD, DVD, Pen Drive, ...)
  - É a memória mais barata, com mais espaço e comum nos computadores
  - São as mais lentas unidades de armazenamento de um sistema computacional.

Aula 5

Programação em Linguagem de Montagem

Sem Anotações

Aula 6

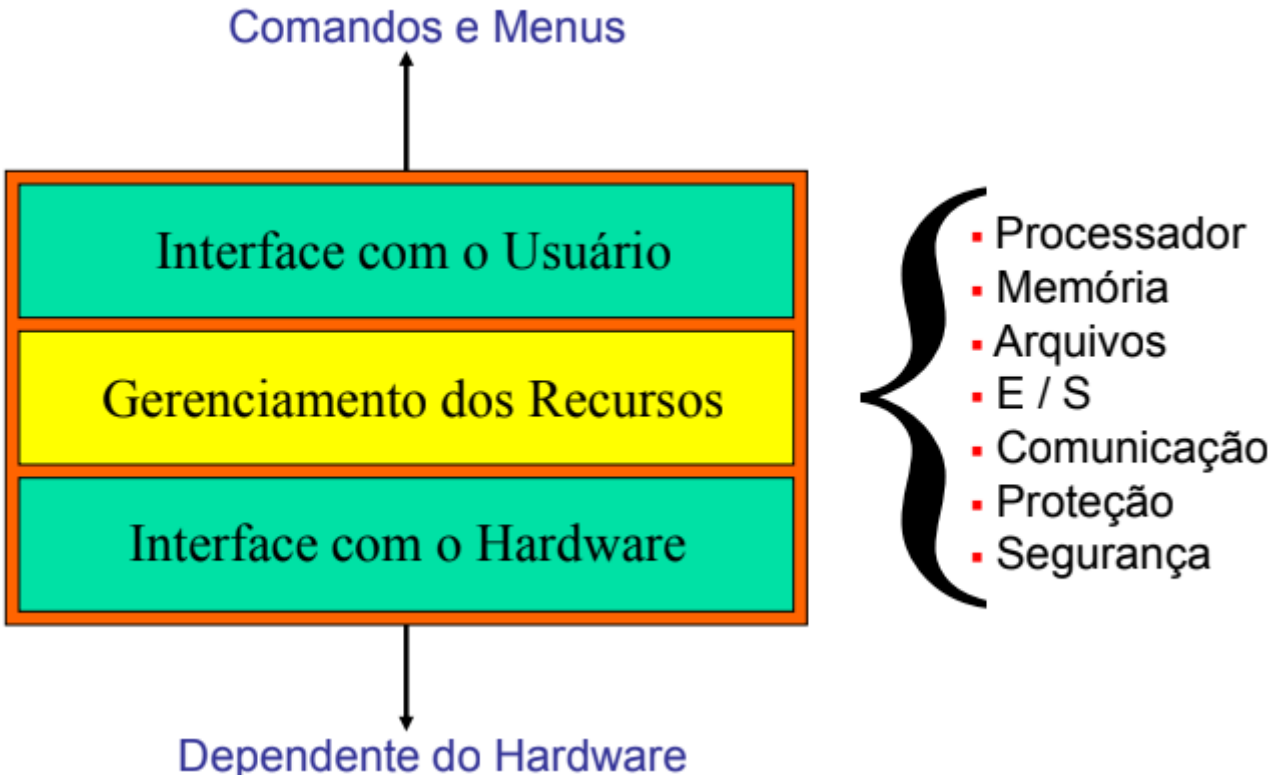
Introdução a Sistemas Operacionais

O que é?

- É um software (programa)
- Atua como intermediário entre o Usuário e o Hardware
- Fornece um ambiente onde o usuário possa executar programas
- Garante uma utilização eficiente do Hardware
- Protege o Sistema de Computação e os usuários

São recursos de hardware	São recursos de software
Tempo de Processador (CPU)	Programas Utilitários
Espaço em Memória	Bibliotecas de Funções – DLLs
Espaço para armazenamento de arquivos	Rotinas de Serviço
Dispositivos de Entrada e de Saída	Programas Aplicativos
Dispositivos de Comunicação de Dados	Programas de Interface com Dispositivos - Drivers

Visão Simplificada de um SO



Interface com o usuário

- Acessar o Sistema – segurança de acesso
- Criar e Gerir Diretórios / Arquivos e Programas
- Executar Programas
- Acessar Dispositivos de E / S

- Acessar conteúdo de Arquivos
- Detectar Erros de execução
- Contabilizar o Uso do sistema

Classificação de SO

Sistema Monotarefa: Admite e gerencia apenas uma tarefa em execução por vez. Ex: DOS
Sistema Multitarefa: Admite e gerencia vários tarefas em processamento concorrente. Ex: Windows 98,Windows 2000/NT/XP, Linux ...

Sistema Monousuário: Admite e gerencia apenas um usuário – não permite que mais de um usuário esteja "logado" simultaneamente
Ex: Windows 98, Windows NT (exceto versão com Terminal Server)

Sistema Multiusuário: Admite e gerencia vários usuários – permite que mais de um usuário esteja "logado" no sistema simultaneamente.
Ex: Linux, Windows 2000, VMS

Sistemas Monoprocessados
Somente reconhece uma única CPU
Multitarefa ou monotarefa
Ex: Windows 98

Sistemas Multiprocessados
Reconhece mais de uma CPU
execução simultânea
Ex:Windows 2000/NT/XP, Linux

Sistemas Batch
Os programas são processados em Lote, um de cada vez, não havendo interação com o usuário.

Sistemas Time Sharing
Os usuários compartilham o tempo de uso do computador que, em seqüência, dedica uma fatia do tempo de processamento para cada usuário.

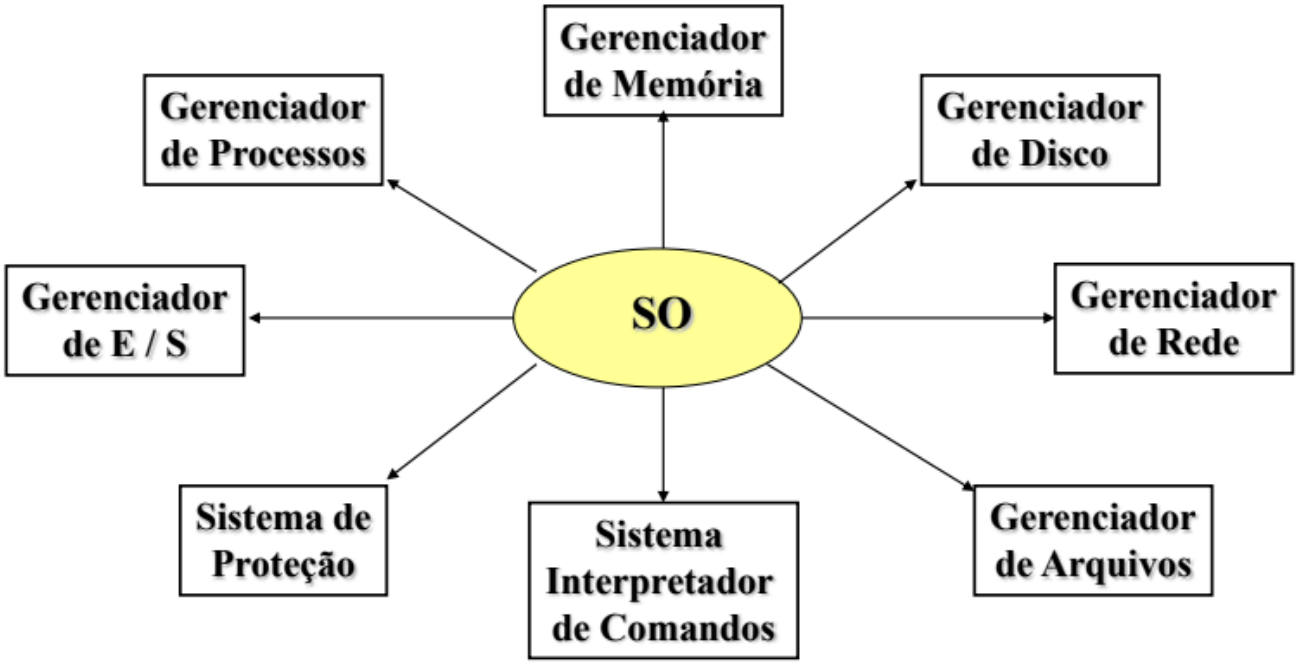
Sistemas de Tempo Real
Sistemas que possuem um forte vínculo com o tempo.
O resultado correto deve ser dado no tempo previsto.
Sistemas Embarcado
Sistemas inseridos em produtos com funções específicas como forno de microondas, VCR, equipamentos bélicos etc.

Aula 7

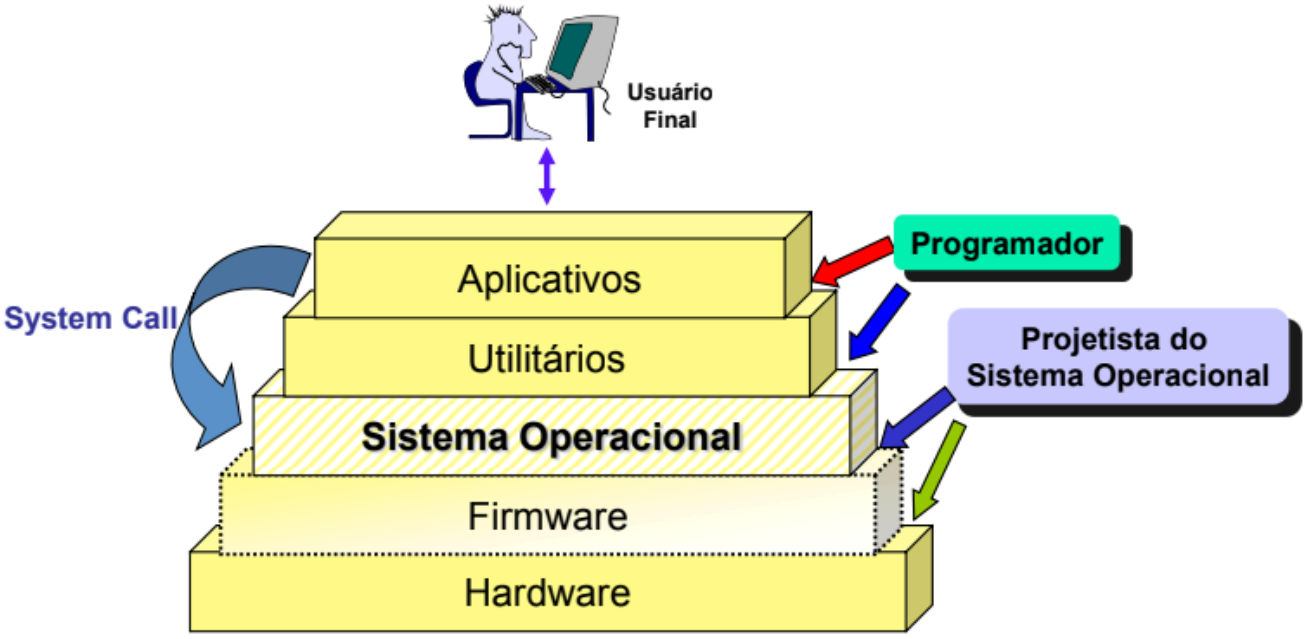
Sem anotações

Aula 8

Componentes de SO

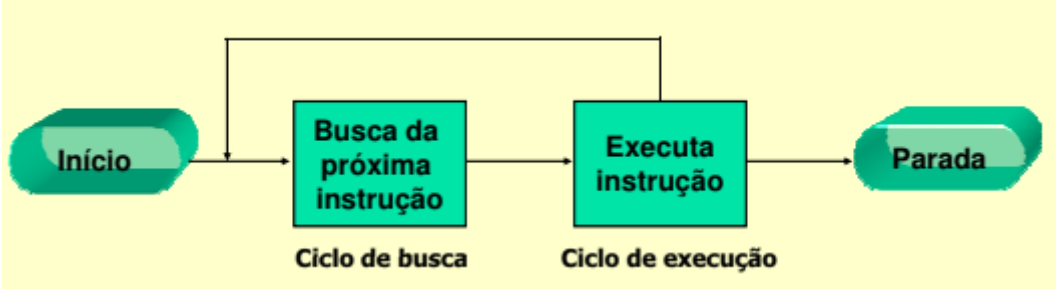


Posicionamento em Camadas



Instruções

Sequência de bits que são interpretados pela UC e que disparam operações lógicas ou aritméticas a serem executadas pelos circuitos do hardware. (dependente do hardware dependente do hardware)



Tipos de Instruções

- Acesso à memória
  - Transferência de dados entre o processador e a memória
- Entrada / saída
  - Transferência de dados entre o processador e o dispositivo
- Tratamento de dados
  - Operações aritméticas ou lógicas
- Controle (desvios)
  - Alteração da seqüência de execução de instruções

Arquiteturas de SO

1. Sistema Monolítico
   1. Dominou até os primeiros grandes sistemas para Mainframes.
   2. Problemas:
      1. Bugs
      2. Memórias
      3. Complexidade
2. Sistema Modular (Camadas)
   1. O sistema é dividido em níveis sobrepostos. Cada nível oferece funções que só podem ser utilizadas pelas camadas mais externas.
3. Cliente / Servidor

Interrupções

- Suspendem a tarefa em execução pela ocorrência de um evento externo (interrupção)
- Permitem a execução de outras instruções enquanto uma operação de E/S está sendo executada
- Melhoram a eficiência do processador
- Acionam uma Rotina do SO chamada de Tratador de
- Interrupções – “Interrupt Handler”
  Observação:
  Após o término da interrupção, a tarefa suspensa pode retornar à execução ou uma outra ser selecionada.

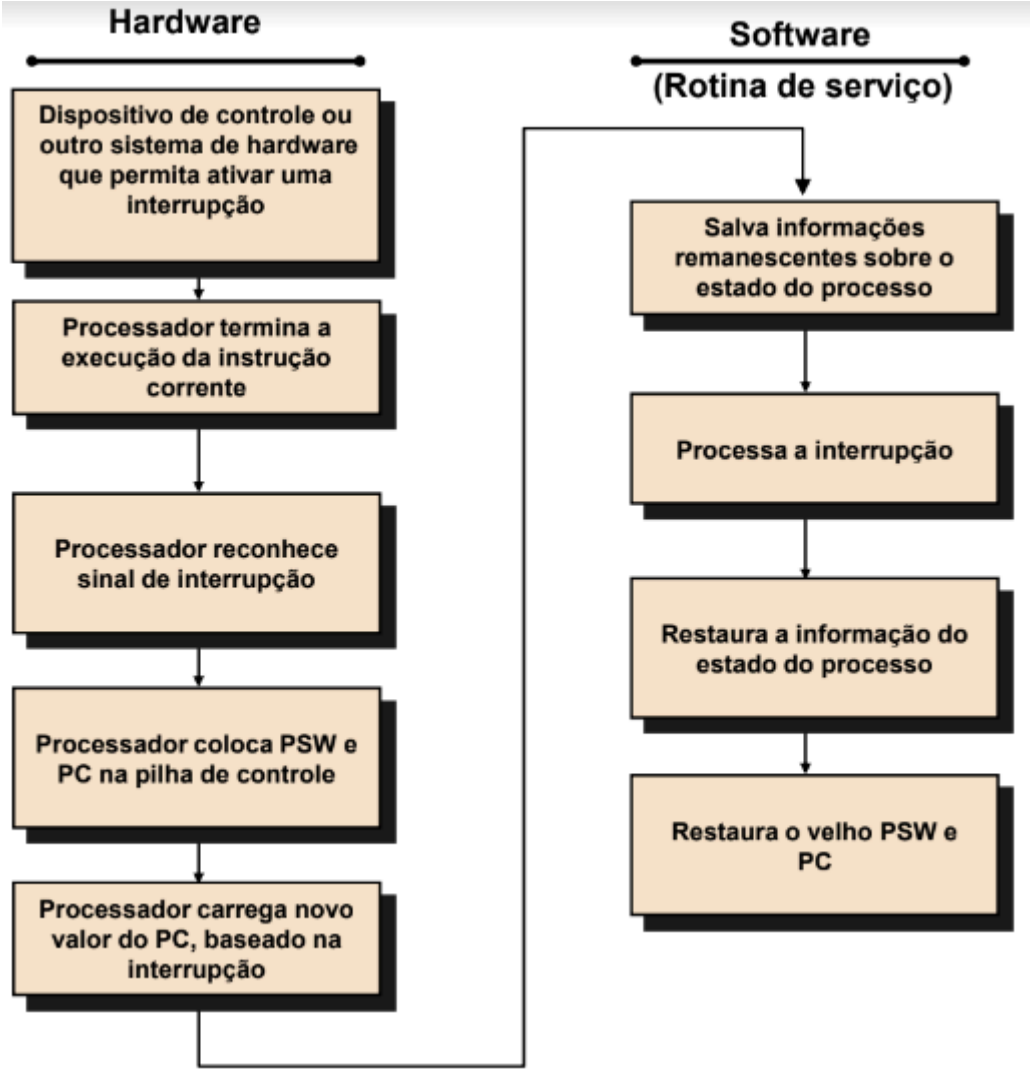
Tratamento de Interrupções

É feito pelo SO, que determina a natureza da interrupção e dispara a Rotina de Serviço adequada para executar as ações que forem necessárias.

A execução do programa corrente é suspensa
O endereço da Rotina de Serviço é localizado na tabela de interrupções
O status do programa corrente é salvo (conteúdo do PC, PSW, registradores, ...)
O controle do processador é transferido para a rotina de serviço

O ponto de interrupção pode ocorrer em qualquer ponto do programa
As rotinas de serviço normalmente fazem parte do Sistema Operacional
Overhead adicional para ativar e executar a rotina de serviço.

Fluxo de Interrupção



Tipos de Interrupções

Sincrona

- Estados de Exceção (trap)
  - estouro aritmético
  - divisão por zero
  - instrução ilegal
  - acesso não permitido
- Interrupção de software
  - chamada de sistema (system call)
- Relógio (temporizador)
  - usado pelo programa
  - usado pelo SO (time slice)

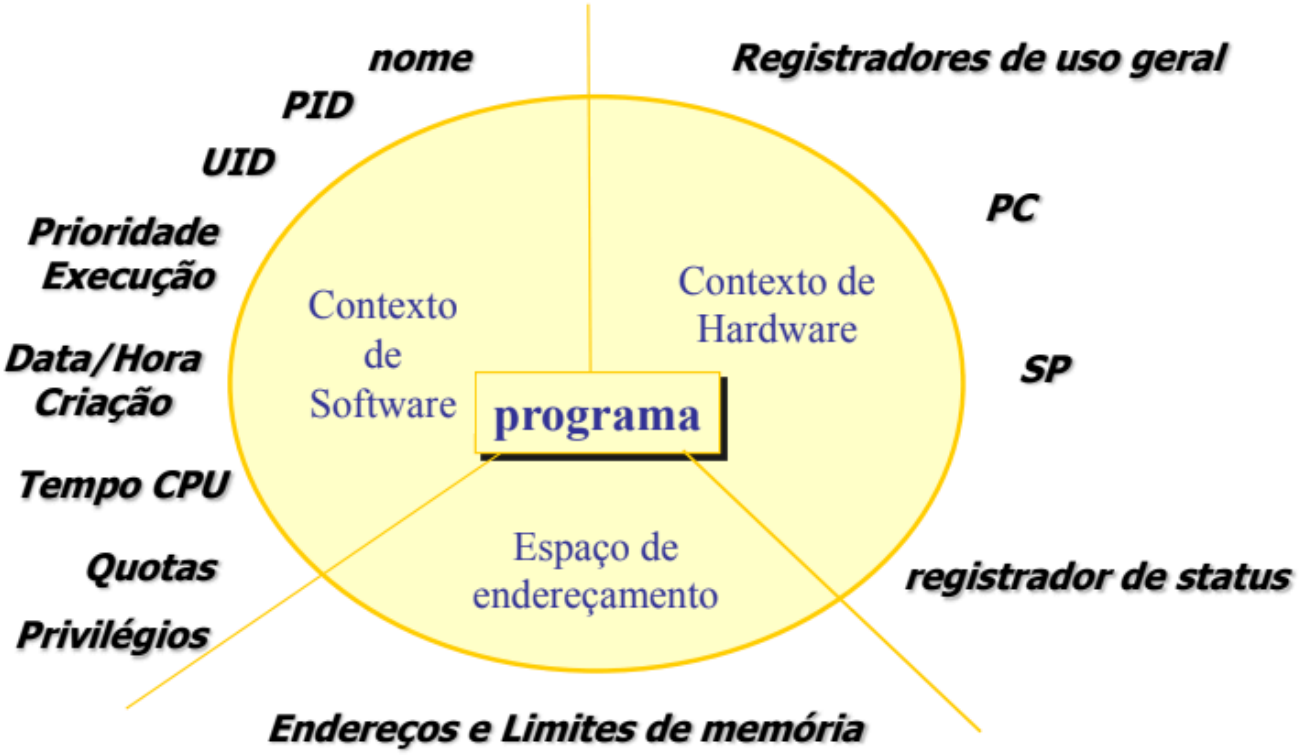
Assíncrona

- Falha de Hardware
  - Erro de paridade Memória
  - Falha no disco, etc
- Entrada e Saída
  - Sinalização de conclusão

Aula 9

Processos

Estrutura de um processo



Estrutura de Controle

identificação	estado
registrador SP	
registrador PC	
registradores de uso geral	
informações de escalonamento	
limites de memória	
privilégios	
relação de arquivos abertos	

Modos de Execução de um SO

Modo usuário → instruções associadas ao uso não privilegiado
Modo kernel → instruções associadas ao uso privilegiado

Etapas de criação de um processo

1. Atribui um identificador único (PID)
2. Aloca uma entrada na tabela de processos
3. Aloca espaço para o processo
4. Inicializa o PCB (Process Control Block)
5. Coloca o processo na fila apropriada
6. Cria estruturas auxiliares

Execução

A execução de um processo leva a as possíveis situações

Trocas de Contexto

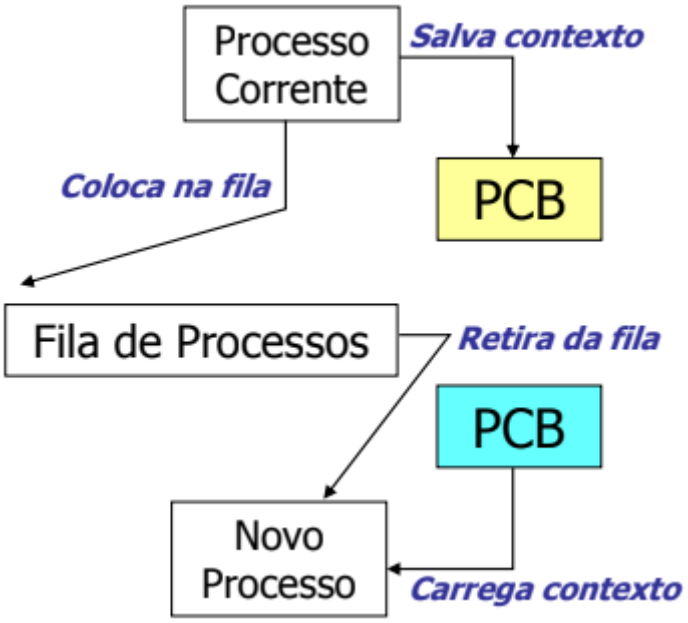
Causas

- Interrupção : Reação a um evento assíncrono
- Trap : Associado a erro na execução de uma instrução
- System Call : Requisição explícita.

Ações Tomadas

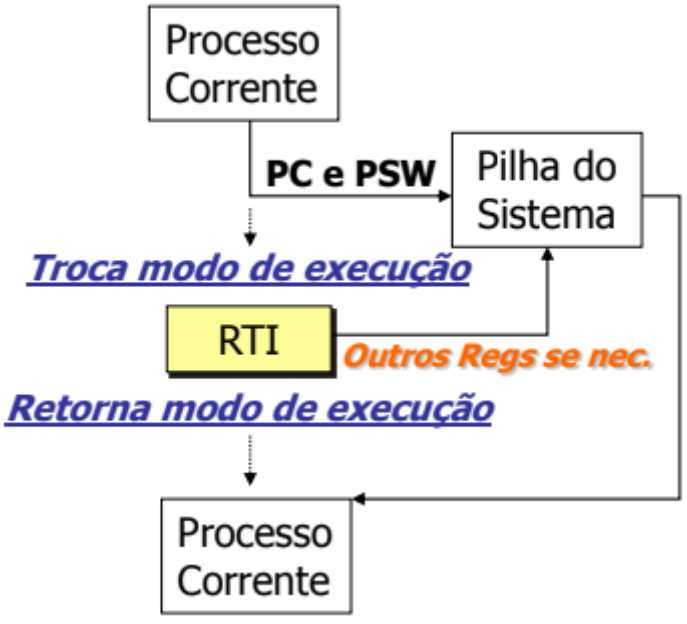
- Salva o estado do processador
- Muda o estado do processo

- Muda o processo para a fila apropriada
- Seleciona o novo processo
- Atualiza o PCB do novo processo
- Modifica os mapeamentos de memória
- Restaura o estado do processador



Trocas de Modo de Execução

É uma troca menor e mais rápida que a troca de contexto;
O estado do processo corrente não é alterado;
Ocorre geralmente quando o processador ao final de um ciclo de instrução detecta a existência de uma interrupção pendente. Nestes casos o processador realiza os seguintes passos:
• Salva o PC e a PSW do processo em execução na pilha;
• Carrega o PC com o endereço inicial da rotina de interrupção;
• Troca o modo de execução de usuário para kernel (privilegiado) para que instruções privilegiadas do tratador de interrupções possam ser executas.



Formas de Execução de SO

• Como Kernel separado
Nesta abordagem as rotinas do SO sempre são executadas como entidades separadas que operam no modo privilegiado e no espaço de endereçamento do Kernel.
• Dentro do processo usuário
Nesta abordagem as rotinas do SO são executadas dentro dos processos usuários, que apenas mudam de modo de execução.
• Como processos separados
Nesta abordagem as rotinas do SO são executadas como processos no modo usuário, trocando o modo de execução quando necessário.

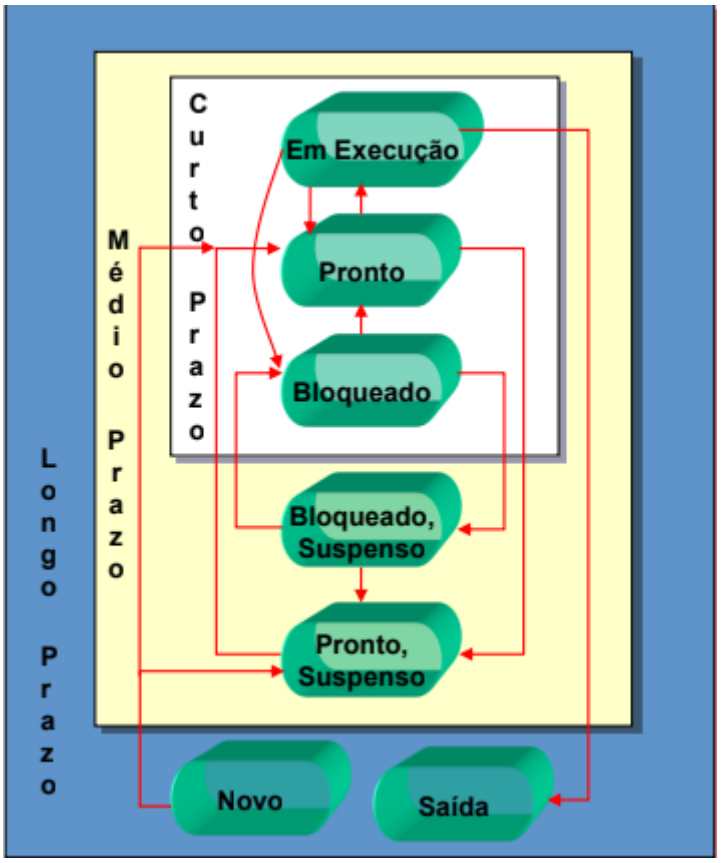
Aula 10

Escalonamento de Processo

Escalonar é uma função do SO que consiste em escolher (determinar) dentre os processos candidatos aquele que:
a) Ganhará acesso ao ambiente de processamento
b) Será retirado do ambiente de processamento
c) Ganhará a posse da CPU
As estratégias de escalonamento adotadas em um SO têm por principais objetivos:

- Manter o processador ocupado a maior parte do tempo (reduzir o idle time)
- Balancear o uso da CPU pelos processos ativos;
- Privilegiar a execução de aplicações críticas;
- Maximizar o throughput do sistema;
- Proporcionar tempos de resposta razoáveis para usuários interativos.

Níveis de Escalonamento



Utilização do Processador:
Usuário → Tempo gasto desde o submissão do requerimento até o início da resposta (**Tempo de Resposta**)
Sistema → Número de processos completados por unidade de tempo (**Throughput**)

Longo prazo

Trata da admissão de novos processos

- Batch: escolhe o próximo processo a ser executado
- Usuário interativo: recusa ou não a sessão

Determina quais e quantos processos são aceitos para execução
O escalonamento de Longo Prazo pode tentar manter uma mistura de processos tipo CPU-bounded e I/O bounded
Controla o grau de multiprogramação

Se mais processos são aceitos:

- É menos provável que todos os processos estejam bloqueados em um determinado instante de tempo, aumentando assim a concorrência (melhor uso da CPU);
- Cada processo terá uma fração menor do tempo de CPU;
- Se o número de processos ativos for muito alto, o overhead causado pela troca de processos será tão grande que o tempo útil de utilização da CPU cairá.

Médio prazo

Trata da admissão de processos que estão na condição de suspensos (completamente fora da memória principal).

- Implementa o swapping
  A decisão de swapping é baseada na necessidade do gerenciamento da multiprogramação
  Feito pelo software de gerenciamento de memória

Determina qual será o próximo processo a ser executado (também chamado de escalonamento de CPU)
Conduz a escolha de outro processo para execução: (Scheduler)

- Interrupção de clock
- Interrupção de E/S
- System Calls e traps
- Sinais

Curto prazo

Trata da execução dos processos que estão na condição de prontos.

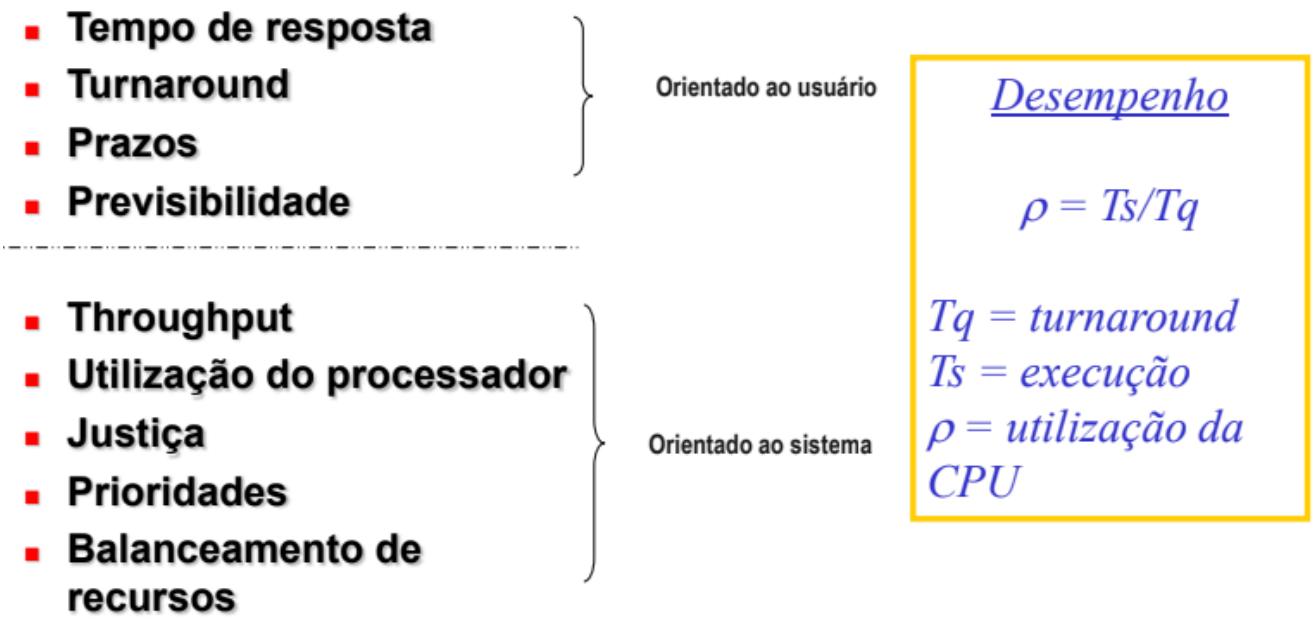
Preempção:
As políticas de escalonamento podem ser classificadas segundo a possibilidade do SO interromper ou não um processo em execução e substituí-lo por outro.

- Sem Preempção
  O processo fica executando até terminar ou até ser bloqueado em consequência a uma chamada ao sistema (I/O ou pedido de recurso do S.O.)
- Com Preempção
  O processo em execução pode ser interrompido:
  - Quando chega um novo processo
  - Se um outro processo de maior prioridade fica pronto
  - Quando interrompido pelo clock (timeslice ou quantum)
    Evita que um processo monopolize o processador,oferecendo um melhor serviço
    Quantum $\rightarrow$ Tempo máximo que um processo pode deter o controle da CPU.
- Escalonamento Não-Preemptivo:
  Foi o primeiro tipo de escalonamento implementado nos sistema multiprogramáveis, onde predominava o processamento batch.
- Escalonamento Preemptivo:
  Caracterizado pela possibilidade do SO suspender a execução de um processo e substituí-lo por outro que esteja no estado de pronto.

Escalonamento de E/S

Trata da requisição de dispositivo de E/S pelos processos com E/S pendentes.

Parâmetros de Avaliação



$\rho \rightarrow$ Número de CPU
$turnaround \rightarrow$ Tempo total de execução

Algoritmos de Escalonamento de Curto Prazo

Por Prioridade

Cada processo recebe uma prioridade ao ser iniciado
O escalonador seleciona o processo pronto de maior prioridade
Existem várias filas “pronto”, uma para cada nível de prioridade

Problema: Processos de baixa prioridade podem sofrer Starvation
Solução: Mudar dinamicamente a prioridade de acordo com a “idade” ou o histórico de execução do processo

First-Come First-Served (FCFS)

Sem preempção
Processos são executados na ordem de chegada
Processos curtos podem esperar em demasia
Favorece processo CPU-bound

- processos I/O-bound esperam processo CPU-bound terminar ou ficar bloqueado
- quando este fica bloqueado, a CPU fica ociosa

Round Robin

Usa preempção baseado num “quantum”(time slice, fatia de tempo)
Quantum: tempo máximo que um processo pode manter o controle da CPU.

- muito curto: maior o overhead de escalonamento
- muito longo: degenera em FCFS
  Busca democratizar a distribuição do tempo da CPU

Shortest Process Next (SPN)

Política sem preempção
Processo com o tempo esperado de serviço (T_s) menor é selecionado primeiro
Favorece processos curtos em detrimento dos longos
Utilizado em sistemas batch

Shortest Remaining Time (SRT)

Versão com preempção (na ativação) da política do processo mais curto
Quando o processo chega na CPU e ele tem um temo de execução menor do que o processo que já está em execução, o processo em execução sofre preempção e então a CPU passa a executar o novo processo.
O tempo de execução deve ser estimado
Favorece mais ainda processos curtos

Highest Response Ratio Next (HRRN)

Escolhe o processo com o maior valor para:

$$\frac{\text{tempo em espera} \ + \ \text{tempo de execução estimado}}{\text{tempo de execução estimado}}$$

Busca privilegiar o balanceamento

Feedback

Penaliza os processos executados há mais tempo

- não é necessário saber o tempo de execução
  Utiliza múltiplas filas com prioridades dinâmicas
- a cada ciclo de execução a prioridade diminui
- starvation: prioridade menor implica em quantum maior ou muita espera aumenta a prioridade
- Favorece processos curtos e I/O bounded

Escalonamento em Multiprocessadores

Formas de Acoplamento

- Fracamente acoplados (ou cluster):
  Uma coleção de sistemas relativamente autônomos onde cada processador tem sua própria memória principal e canais de entrada e saída (rede de computadores).
- Fortemente acoplados:
  Uma coleção de processadores que compartilham memória principal e estão sob controle

integrado de um sistema operacional
    - Mestre / Escravo
    - SMP (Multiprocessamento Simétrico)

Granularidade da Sincronização

- Independente
  Processos independentes. Não é preciso sincronização.
- Muito grossa
  Processos em diferentes nós de uma rede (fracamente acoplados) – intervalo de 2000 a 1 milhão de instruções
- Grossa
  Processos em multiprocessamento – intervalo de 200 a 2000
- Média
  Rotinas de um mesmo processo – intervalo de 20 a 200
- Fina
  Instruções de um mesmo processo – intervalo < 20

Estratégias

Processo:
Quanto mais processadores, menos importante é o escalonamento. Uso do FCFS
Threads:
• Load Sharing
Garantia de compartilhamento dos recursos e uso do processador. Fila única de processos

• Gang Scheduling:
Conjunto de threads relacionadas é escalonado para trabalhar num conjunto de processadores ao mesmo tempo, numa base de um-para- um

• Dedicated Processor Assignment:
Para cada programa é alocado um número de processadores igual ao número de threads, para o tempo total de execução do programa

• Dynamic Scheduling
O número de threads no programa pode ser alterado durante o curso de execução

Sistemas de Tempo Real

Sistema de tempo real pode ser definido como aquele em que o funcionamento correto da aplicação não depende apenas do resultado gerado mas também do tempo em que o mesmo é gerado.

Tarefa Hard Real Time

Os deadlines são imperativos. Não sendo cumprido os resultados de nada servirão.

Tarefa Soft Real Time

Os deadlines são desejáveis porém não mandatórios.

Características

Determinismo – tempo de execução predeterminado
Tempo de resposta – atraso de tempo (latência) necessária para o sistema iniciar o tratamento de uma interrupção
Controle pelo usuário – o usuário tem que ter a possibilidade de determinar o tipo de suas aplicações (soft ou hard), bem como definir a prioridade das mesmas.
Confiabilidade – é muito mais importante em sistemas de tempo real do que em todos as demais modalidades.
Tolerância a falhas – um SO de tempo real é dito estável se na impossibilidade de atender ao deadline de todas as tarefas ativas, garante o atendimento daquelas mais críticas.

Funcionalidades

– Troca de contexto rápida
– Tamanho reduzido
– Resposta rápida à interrupções
– Multitarefa com funcionalidades de IPC e concorrência
– Escalonamento preemptivo com base em prioridade
– Minimização dos intervalos em que as interrupções estão desabilitadas
– Primitivas para atrasar por um tempo fixo, para suspender ou parar tarefas em execução
– Alarmes e timeouts especiais

Escalonamento

1. Round Robin Premptivo
2. Por Prioridade Não-Preemptivo
3. Por Prioridade Preemptivo em Determinados Pontos
4. Preempção Imediata

Aula 11

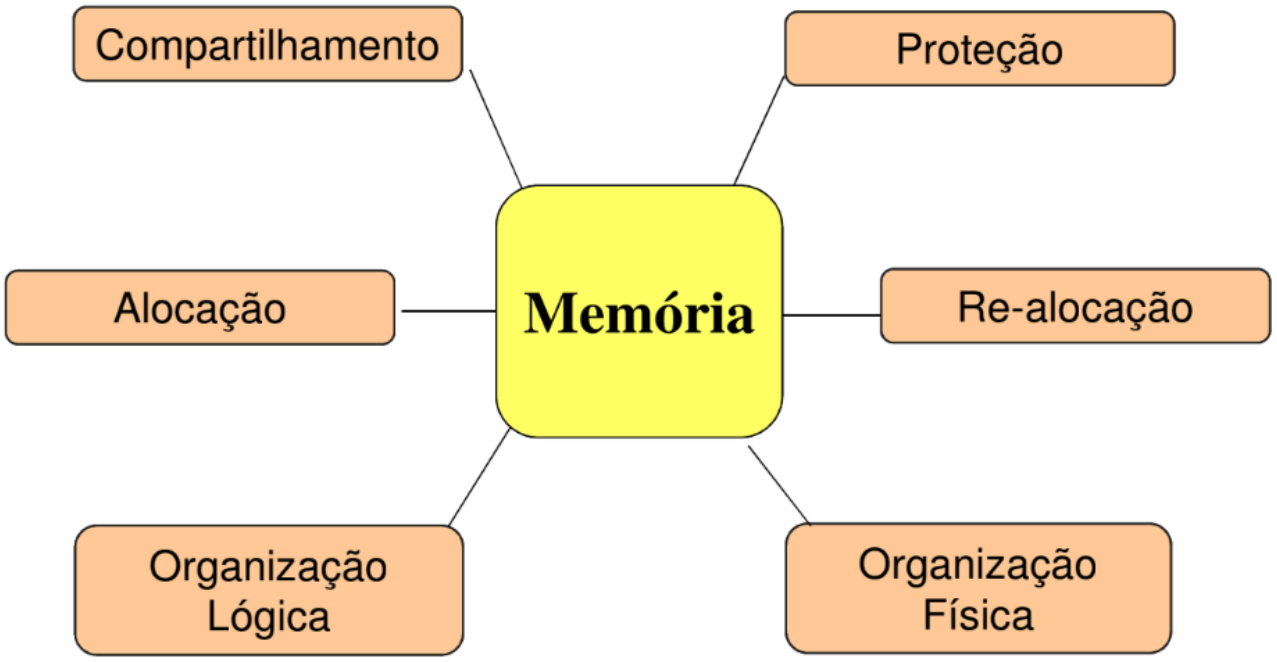
Gerenciamento de Memória

Gerenciar a memória consiste na tarefa de subdividir e alocar espaços para acomodar os processos em execução.

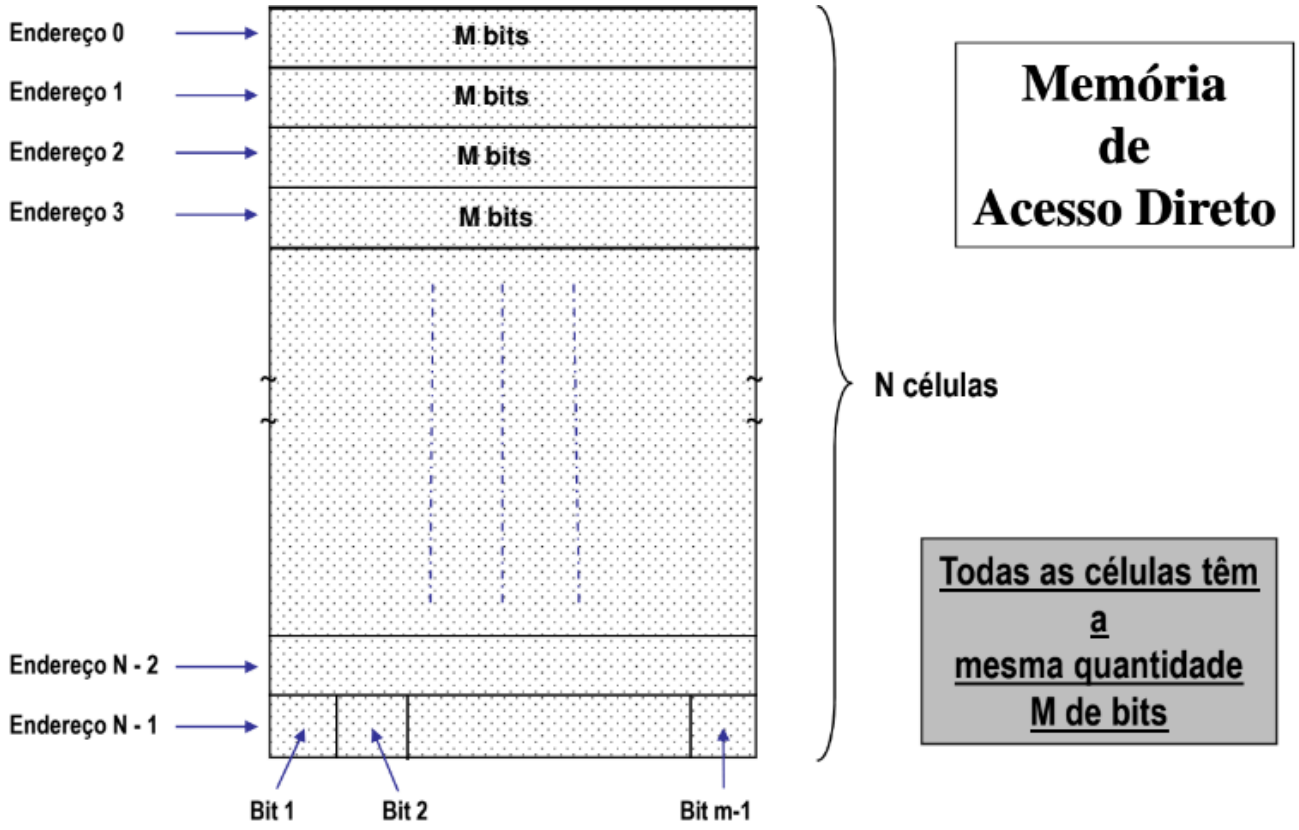
- Espaços são requeridos requeridos (alocados) e liberados, a medida que os processos são executados.
- Os espaços de memória ocupados pelos processos precisam ser preservados (protegidos)
- Os processos podem ter a necessidade de aumentar o espaços ocupado ou mesmo compartilhar espaço com outros

Organização e Funcionalidades

Funcionalidades



Organização Física



Organização Lógica (Particionamento)

Consiste na forma como a memória é vista (particionada) logicamente pelo SO.

Contíguo Simples

Estático

- Endereços absolutos são gerados de forma estática - Linkeditor ou Carregador
- Simples implementação
- Baixo desempenho
- Fragmentação interna

Estático Relocável

- Endereçamento relativo
- Endereços absolutos dinâmicos, calculados em tempo de execução
- Baixo desempenho
- Fragmentação interna

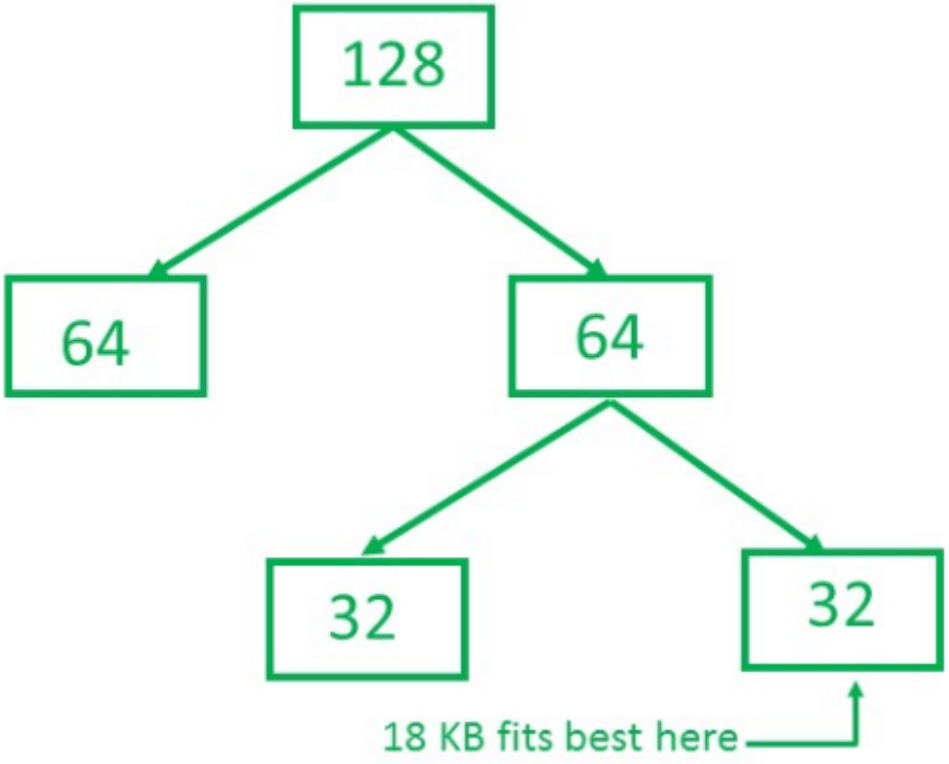
Dinâmico

- A quantidade e o tamanho das partições são variáveis
- Para cada processo, é alocado o espaço exato que for necessário (não tem fragmentação interna)
- Eventualmente, são criados “buracos” de tamanho pequeno, sem utilidade de uso (fragmentação externa)
- Os processos, de tempos em tempos, precisam ser re-alocados para eliminar os “buracos” (compactação)

Esquemas de Alocação

- **First-Fit**
  Procura a partir da memória o primeiro bloco livre que sirva
  Pode criar muitos pequenos blocos livres no início da memória
  Considerado o algoritmo mais rápido
- **Next-Fit**
  Escolhe o próximo bloco livre a partir da última alocação em que caiba o processo
  Tende a acabar com o grande bloco livre no final da memória
- **Worst-Fit**
  Escolhe o maior bloco livre a partir início da memória.
  Tende deixar buracos maiores que o Best-fit
- **Best-Fit**
  Escolhe o menor bloco que comporte o processo
  Cria muitos buracos pequenos, exigindo mais compactações
  Oferece o pior desempenho
- **Buddy System**
  A memória é organizada em blocos de 2^k
  $L \leq K \leq U$
  2^L – menor bloco que pode ser alocado

2^U – maior bloco (tamanho da memória)
reduz problema da colcha de retalhos (re-agrupamento)
insere fragmentação interna
Exemplo:



Paginado

A memória é particionada em pedaços de tamanho igual, assim como os processos;

Os pedaços que compõem os processos são chamados de páginas e os pedaços de memória são as molduras de página (frames);

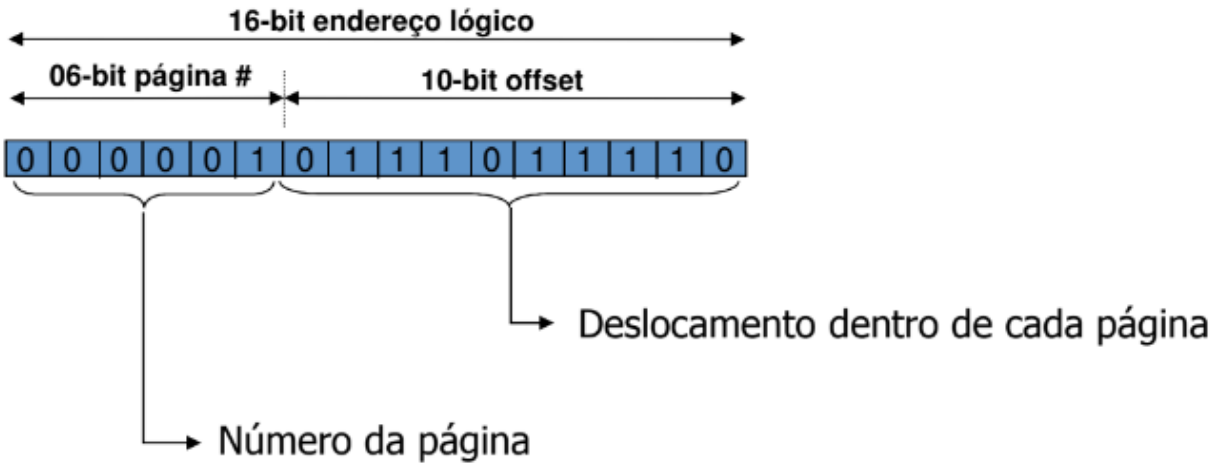
# Págs	Frames

Quando um processo é carregado, suas páginas são alocadas em quaisquer molduras disponíveis, não necessariamente contíguas;

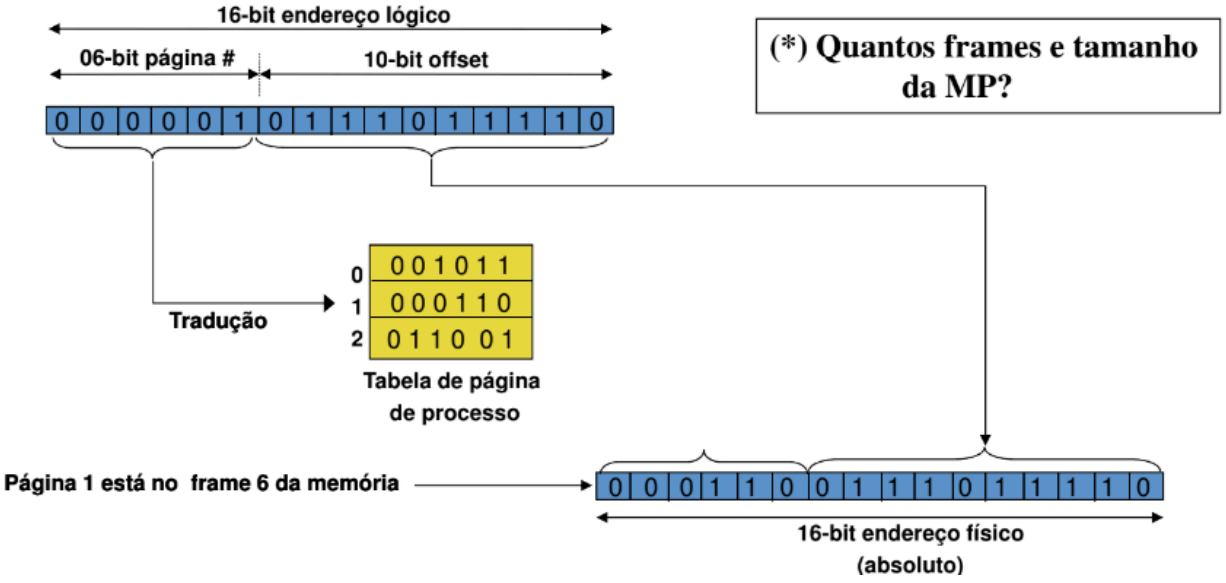
O S.O. precisa manter uma tabela de páginas por processo e uma lista de molduras disponíveis.

Tabelas de Páginas: Precisa ser mantida uma para cada processo, de forma a associar a página do processo com o frame correspondente em memória utilizado.

- Endereço Lógico



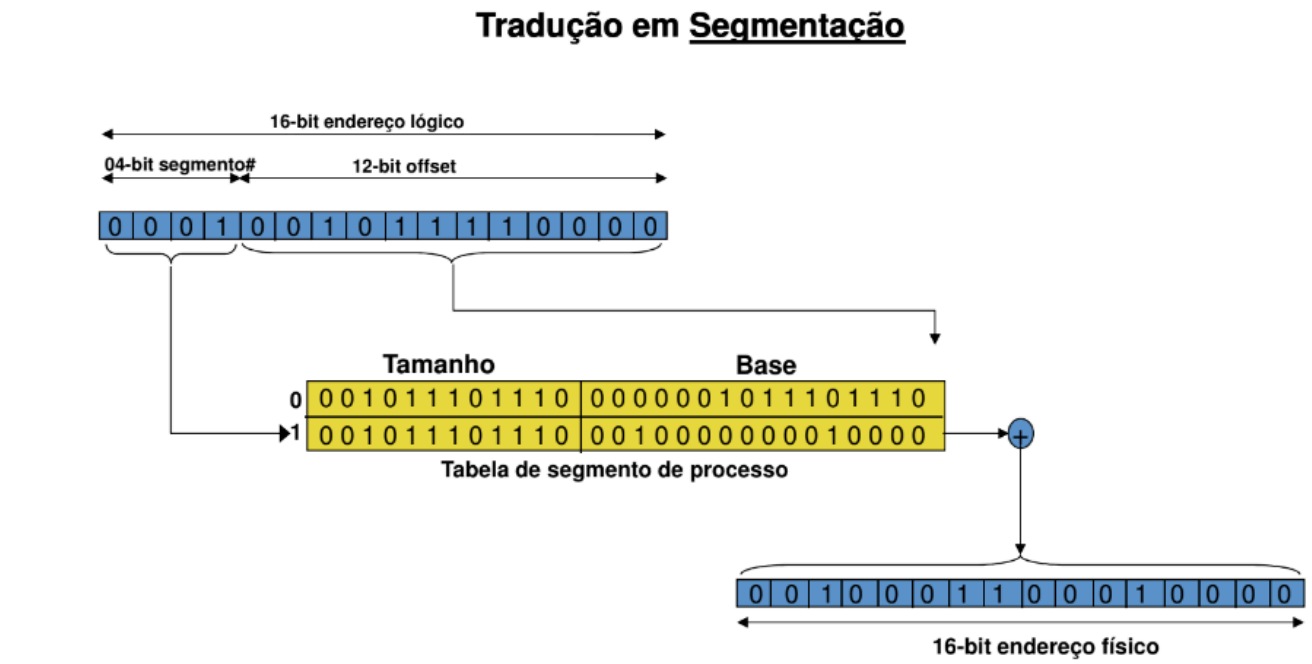
Tradução Lógico → Físico:



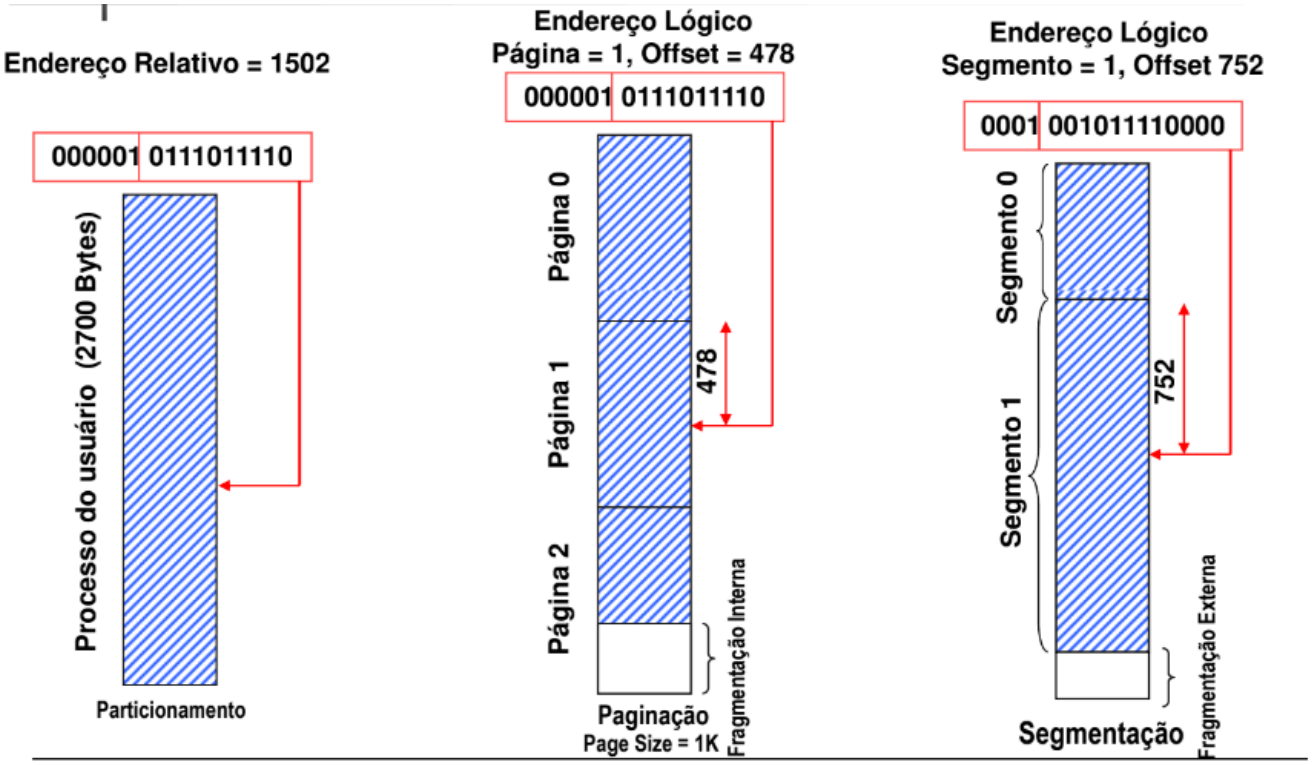
Segmentado

Cada programa é subdividido em blocos de diferentes tamanhos, chamados segmentos.
Quando um processo é carregado para a memória principal, cada segmento diferente pode ocupar qualquer lugar.
O SO mantém uma tabela de segmentos de cada processo. Cada entrada contém:
o início do endereço físico daquele segmento
o tamanho do segmento (por proteção)
Apresenta fragmentação externa

- Tradução Lógico → Físico:



Tradução de Endereços

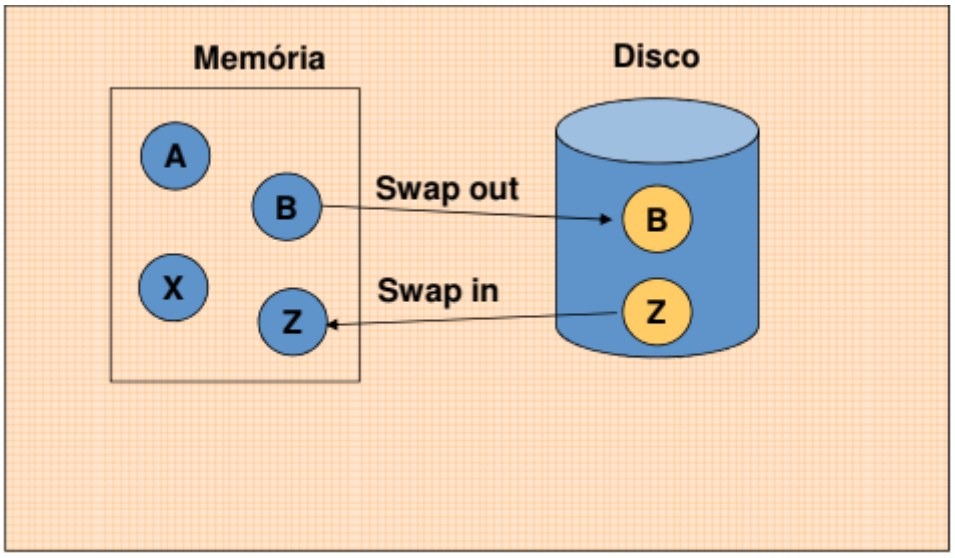


Vantagens da Paginação e Segmentação

Maior flexibilidade na alocação de espaços em memória (tabelas de páginas e de segmentos livres)
Endereçamento não contínuo em memória
Baixa fragmentação interna
Correlação com a lógica do programa

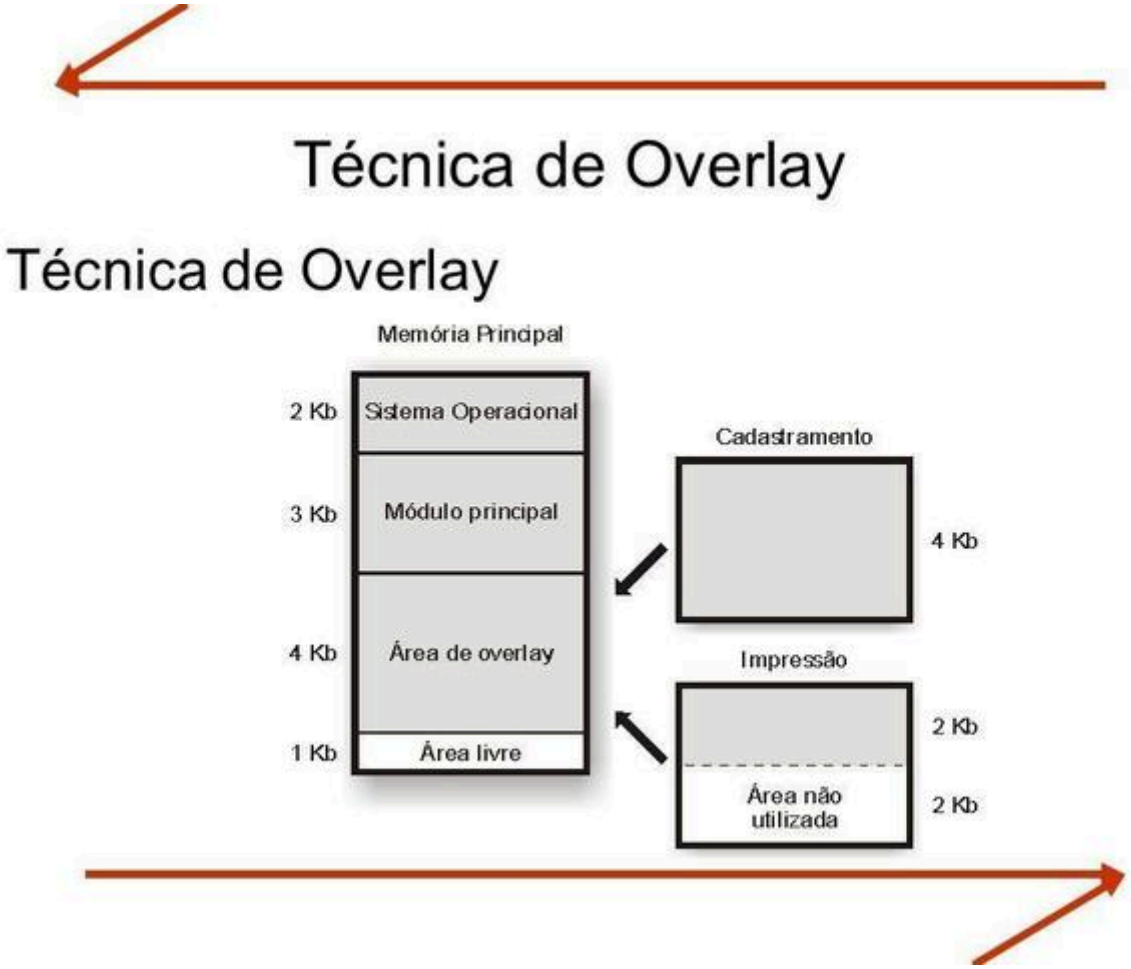
Swapping

Um processo saí memória para outro entrar

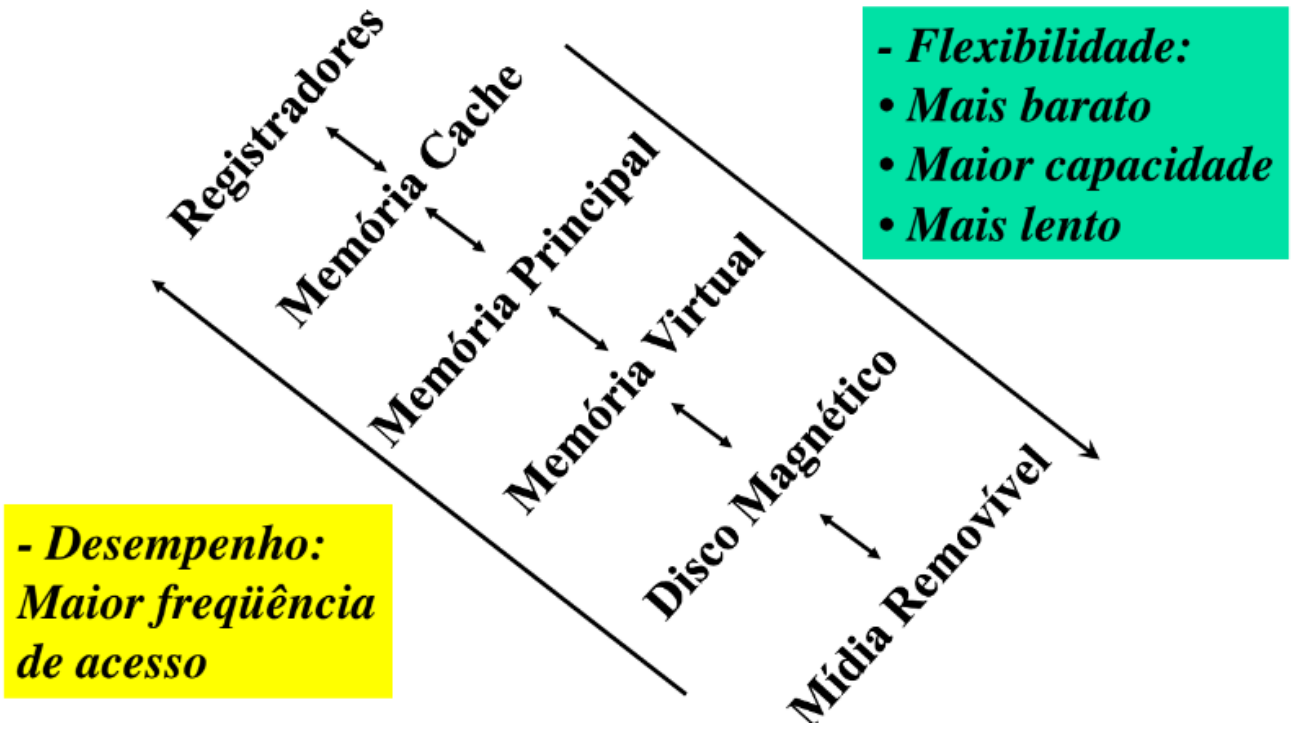


Overlay

Consiste em pegar parte da memória disponível e dividi-la em duas parte, *Módulo Principal* e *Área de Overlay*



Hierarquia de Memória



Cache

Sem anotações

Aula 11

Memória Virtual

Consiste na utilização de espaços do disco rígido como extensão lógica da memória primária.

A memória virtual é transparente para o programador e para o processador.
A memória virtual expande o tamanho da memória primária.
A memória virtual não é ilimitada.
O sistema ganha em flexibilidade e perde em termos de desempenho.

Características

Todas as referências a memória passam a ser com endereços lógicos virtuais (VA), que são traduzidos em endereços físicos, em tempo de execução.

Uma tarefa é dividida em partes (páginas ou segmentos), não necessariamente localizados em áreas contíguas na memória.

Com a memória virtual acaba a necessidade de todas as partes de uma tarefa estarem carregadas em memória primária.

Uma tarefa pode ocupar diferentes áreas de memória durante a sua execução

O uso da memória virtual é transparente ao usuário e à própria CPU

Maior tempo de resposta para as referências à memória.

Maior complexidade do hardware e do esquema de gerenciamento.

Impossibilidade de estimar de forma precisa e segura, o tempo a ser gasto em qualquer referência à memória.

Uma mesma referência à memória pode consumir tempos diferentes de execução.

Vantagens

Mais processos mantidos em MP
os processos são carregados parcialmente
maior eficiência na utilização do processador

Processos podem ser maiores que a memória principal
Todo programador tem disponível uma memória de trabalho (virtual) de tamanho igual a todo espaço de endereçamento disponível.

O SO se encarrega de trazer para a memória física as partes necessárias para a execução do programa.

Endereçamento

Cada referência virtual é convertida para o endereçamento físico em tempo de execução. Este processo de conversão é chamado de mapeamento.

Hardware:
– Tradução (mapeamento) eficiente de endereços.
– Movimentação eficaz dos trechos de informação entre a memória virtual e a física.

SO:
– Re-alocação eficiente da memória física.
– Princípio da Localidade x Trashing

OBS: Memória Virtual ≠ Swapping

Princípio da Localidade

As referências de memória tendem a ser agrupadas em termos espaciais e/ou temporais.

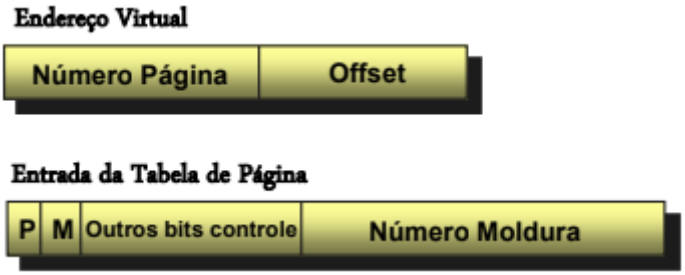
• Somente alguns trechos do código são necessários para a execução num curto espaço de tempo;
• É possível ter uma razoável noção dos trechos de código que serão utilizados num futuro próximo, reduzindo os riscos de trashing.

Trashing

Trashing é a situação onde o sistema passa a maior parte do tempo removendo e trazendo partes de processos ao invés de executar instruções dos mesmos.
A memória normalmente está toda ocupada com partes de diversos processos.
Quando o SO precisa carregar uma nova parte, um outro pode precisar ser removido para abrir espaço.
Se for removida uma parte que seja referenciada logo a seguir, esta precisará ser carregada novamente.

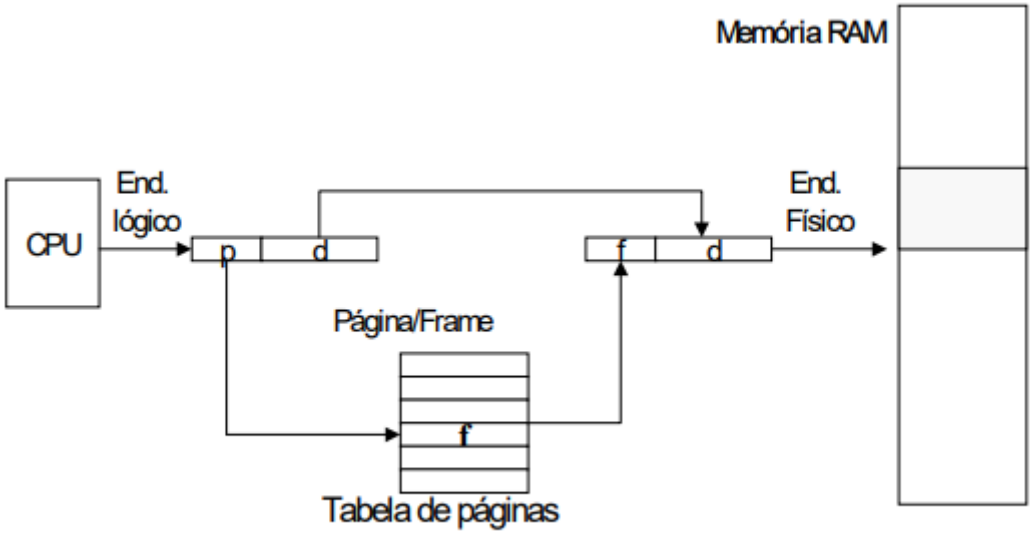
Paginação

Cada processo tem sua própria tabela de páginas
Cada entrada contém um bit de presença (P) indicando se a página se encontra na memória física ou não
Se a página está presente, a entrada contém o número da moldura da página correspondente
Cada entrada também contém um bit de modificação (M)
páginas não modificadas não precisam ser gravadas em disco quando removidas

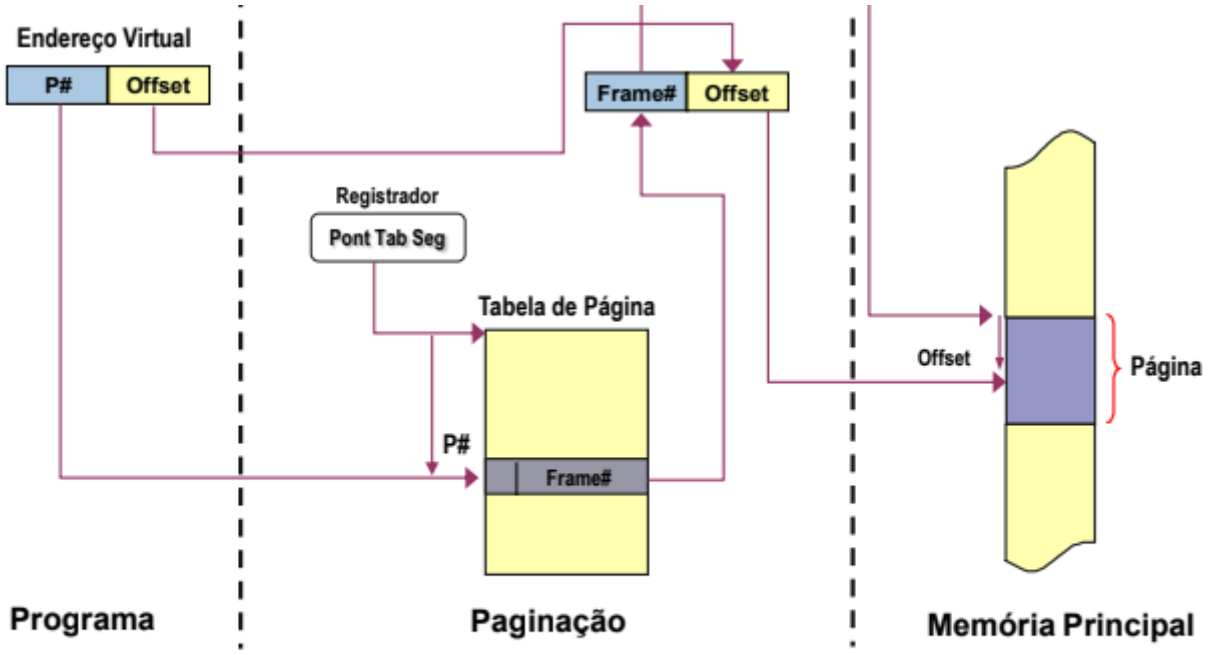


Tradução de Endereços

Exemplo 1:



Exemplo 2:



- ***Problema:*** → ***Espaço ocupado pela PT (proporcional ao tamanho da memória virtual)***
- ***Solução:*** → ***Tabela de páginas invertidas***
  Múltiplas tabelas de página

Múltiplas Tabelas

Tabelas Invertidas

Transaction Lookaside Buffer - TLB

Tamanho da Página

Quanto menor a página = Menor a fragmentação interna
Quanto menor a página = Mais páginas por processo
maior a tabela de páginas
mais tabelas na memória secundária
mais falta de páginas (page faults)
A transferência de dados com a memória secundária é mais eficiente com blocos maiores
O número de falta de páginas (page faults):
diminui à medida que o tamanho da página aumenta (até certo ponto)
depois deste ponto, começa a baixar (fenômeno da saturação)

Estratégias de SO (Segmentação Paginada)

Busca

Determina quando uma página deve ser carregada

Por demanda
Somente traz as páginas referenciadas
Existem muitas faltas de página quando o processo começa
Por carga antecipada

Traz mais páginas do que o necessário
É mais eficiente trazer várias páginas contíguas em disco do que cada uma individualmente
Se torna ineficiente se são trazidas páginas que não serão referenciadas

Alocação

Determina onde será carregada a página ou segmento na memória
Irrelevante em sistemas paginados => A eficiência do hardware é a mesma para qualquer combinação página-moldura
Em sistemas com segmentação pura deve ser usado um dos algoritmos first-fit, worst-fit, next-fit

Re-alocação

Determina a página a ser removida quando uma nova página está sendo carregada

- Algoritmos
  - *Ótimo*
    Seleciona a página cuja próxima referência será a mais distante
    - Resulta no menor número de falta de páginas
    - Irrealizável já que não é possível prever o futuro
    - Útil para avaliar a eficiência de outras políticas
  - *Least Recently Used - LRU*
    Substitui a página que não é referenciada há mais tempo (Least Recently Used)
    - Pelo princípio de localidade, esta deve ser a página com menos probabilidade de ser referenciada no futuro próximo
    - O desempenho pode ser quase tão bom quanto a política ótima
    - Implementação computacionalmente cara
  - *First in, First out - FIFO*
    Substitui a página carregada há mais tempo (First in, First out)
    - As molduras formam um buffer circular
    - Algoritmo extremamente simples
    - A página residente há mais tempo na memória não significa que não será mais utilizada
    - É possível que hajam trechos utilizados constantemente durante toda a execução do programa
  - *Do Relógios*
    Aproximação do algoritmo LRU
    - Variações conhecidas como NRU (Not Recently Used)
    - Requer um bit adicional na tabela de páginas: o bit de uso (ou referência)
    - Quando a página é carregada, o bit de uso é ligado e avança o ponteiro para o próximo frame
    - Quando a página é referenciada, o bit de uso é ligado
    - A primeira página com o bit de uso igual a zero é removida

- Durante a procura da página a ser substituída, os bits de uso das páginas pesquisadas são desligados

OBS: Tem mais matéria após esse tópico, porém como esta foi a ultima parte abordada, presumo que a matéria para prova termina aqui.