Aula 01

Apresentação da Disciplina;

Introdução aos Sistemas Operacionais;

Histórico.

Definição

"... um conjunto de rotinas executadas pelo processador, de forma semelhante aos programas dos usuários."

"... O SO tem por objetivo funcionar como uma interface entre o usuário e o computador, tornando sua utilização mais simples, rápida e segura".

Francis Machado e Luiz Paulo Maia

Definição

"... é um programa que atua como intermediário entre o usuário e o hardware de um computador."

"... deve propiciar um ambiente no qual o usuário possa executar programas de forma conveniente e eficiente".

Silberschatz, Galvin e Gagne

Características de um SO

Transparência

- Simplifica as Atividades para o usuário

Gerência

- Compartilha e Otimiza os recursos computacionais

Encapsulamento

- Esconde os detalhes da implementação;
- “Padroniza" a interface, muitas vezes mesmo entre dispositivos diferentes.

Evolução dos SOs

Nos primeiros dispositivos computacionais:

- Operação complexa
  - Já ouviram falar do cargo de “Operador de Computador” ?
- Programação ainda mais complexa
  - Exigia grande conhecimento do *hardware* , inclusive da "linguagem de máquina”

Atualmente, SOs tendem à linguagem natural

Filme de 1986 (a 35 anos atrás)

Prof. Marco Câmara

Em 2021 …

Exemplos SO (computadores pessoais)

Windows (Windows 11) 69,52%

MacOS (Versão atual: Ventura; breve “Sonoma") 20,42%

”Não sei" 3,69%

Linux (MX Linux, Debian, Manjaro, Ubuntu) 3,07%

ChromeOS 4,13% -> **Total > 100% ?**

Unix (FreeBSD, SCOUnix, HP-UX, SunOS) 0% ?

https://gs.statcounter.com/os-market-share/mobile/worldwide

Exemplos SO (*smartphones*)

Android (70,9%)

iOS (28,36%)

Samsung (0,38%)

KaiOS (0,15%)

Não sei (0,19%)

Windows (0,02%)

https://gs.statcounter.com/os-market-share/mobile/worldwide

Exemplos SO (outros ambientes)

Workstations de aplicação específica

Servidores

Eletrodomésticos (SO embarcados)

Computadores de grande porte (*mainframes*)

Exemplos SO

Quais as principais diferenças?

- Quem conhece mais de um sistema operacional?
- Apresentar diferenças.

Aula 02

Tipos de SO;

Histórico dos SOs;

Máquina de Níveis;

Conceitos de *Hardware.*

Tipos de Sistemas Operacionais

Monoprogramáveis / Monotarefa

Multiprogramáveis / Multitarefa

Multiprocessados

Tipos de Sistemas Operacionais

- Sistemas Monoprogramáveis / Monotarefa
  - Todos recursos do sistema dedicados a uma tarefa
  - Execução de programas seqüencialmente
  - Usado nos primeiros computadores de grande porte
  - Usado nos primeiros computadores de pequeno porte
    - Sistemas Monousuário

Tipos de Sistemas Operacionais

- Sistemas Monoprogramáveis / Monotarefa
  - Sub-utilização dos recursos do sistema
    - Processador X Operações de E/S
    - Memória
    - Dispositivos de E/S
  - Implementação simples
    - Sem recursos de proteção

Tipos de Sistemas Operacionais

- Sistemas Multiprogramáveis / Multitarefa
  - Recursos do sistema compartilhados por diversas tarefas
  - Execução de programas concorrentemente
    - Aumento da produtividade
    - Redução de custos
    - Suporte a Sistemas Multiusuário
  - Compartilhamento na utilização dos recursos do sistema
    - Processador X Operações de E/S
    - Memória
    - Dispositivos de E/S

Tipos de Sistemas Operacionais

- Sistemas Multiprogramáveis / Multitarefa
  - Implementação complexa
    - Gerenciamento dos acessos concorrentes aos recursos
    - Recursos de proteção
  - Sistemas Monousuário X Multiusuário
    - Mainframes
    - Computadores Pessoais e Estações de Trabalho

Tipos de Sistemas Operacionais

- Sistemas Multiprogramáveis / Multitarefa
  - Sistemas de Tempo Real
    - Semelhante aos Sistemas de Tempo Compartilhado
    - Limites rígidos para tempo de resposta
    - Sem fatia de tempo
    - Níveis de prioridade
    - Aplicações multimídia interativas

Tipos de Sistemas Operacionais

- Multitarefa
  - Colaborativa
    - Windows 95, 98
    - Não existe fatia de tempo
  - Preemptiva
    - Windows 10 e Server, Linux, Mac OS, ...
    - Existe fatia de tempo
    - Preempção

Tipos de Sistemas Operacionais

- Sistemas Multiprocessados
  - Sistemas com mais de uma CPU interligada
    - Execução simultânea de programas
    - Supre dificuldade no desenvolvimento de processadores mais rápidos
    - Ideal para sistemas que necessitam uso intensivo de CPU
      - Processamento científico
    - ... ou para uso diferenciado de CPU
      - Baixo consumo, ou processamento avançado? (celulares com múltiplas câmeras ?)

Tipos de Sistemas Operacionais

- Sistemas Multiprocessados
  - Características
    - Multiprogramação
      - Aplicada a cada processador
    - Escalabilidade
      - Aumento da capacidade computacional
    - Reconfiguração
      - Tolerância à falha em algum processador
    - Balanceamento
      - Distribuição de carga de processamento

Tipos de Sistemas Operacionais

- Sistemas Multiprocessados
  - Classificação
    - Em função:
      - da forma de comunicação entre CPUs
      - do grau de compartilhamento da memória e E/S
    - Sistemas Fortemente Acoplados
      - Processadores com múltiplos núcleos, por exemplo.
    - Sistemas Fracamente Acoplados
      - *Server Cluster*, por exemplo.

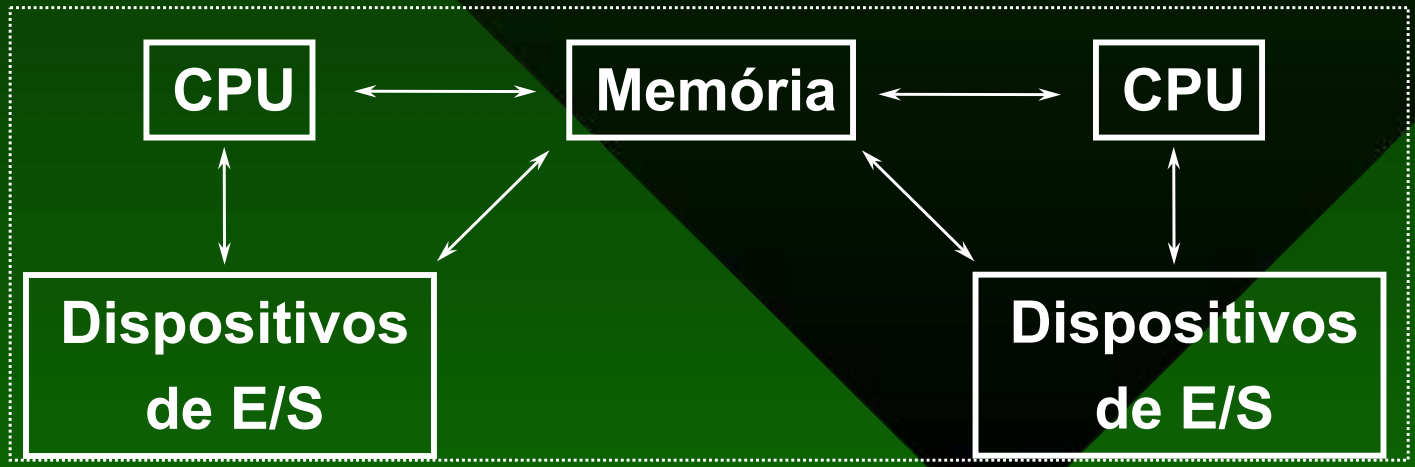
Tipos de Sistemas Operacionais

Sistemas Multiprocessados

Sistemas Fortemente Acoplados

Processadores compartilham um única memória

Espaço de Endereçamento Único
Único SO



Tipos de Sistemas Operacionais

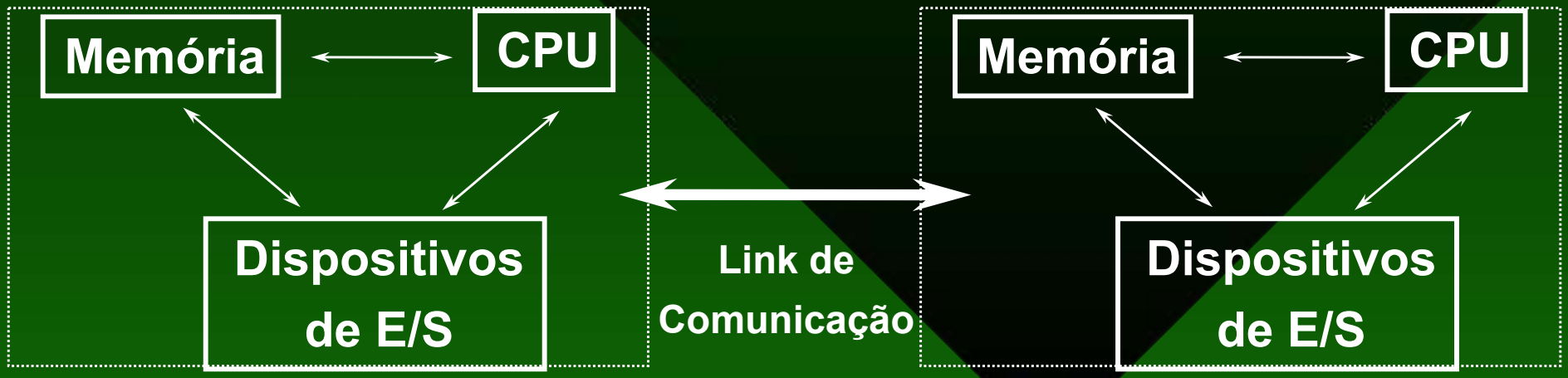
Sistemas Multiprocessados

Sistemas Fracamente Acoplados

Sistemas de Computação independentes, mas conectados (multicomputadores)

Processamento Distribuído

SO de Rede (SOR) X SO Distribuído (SOD)



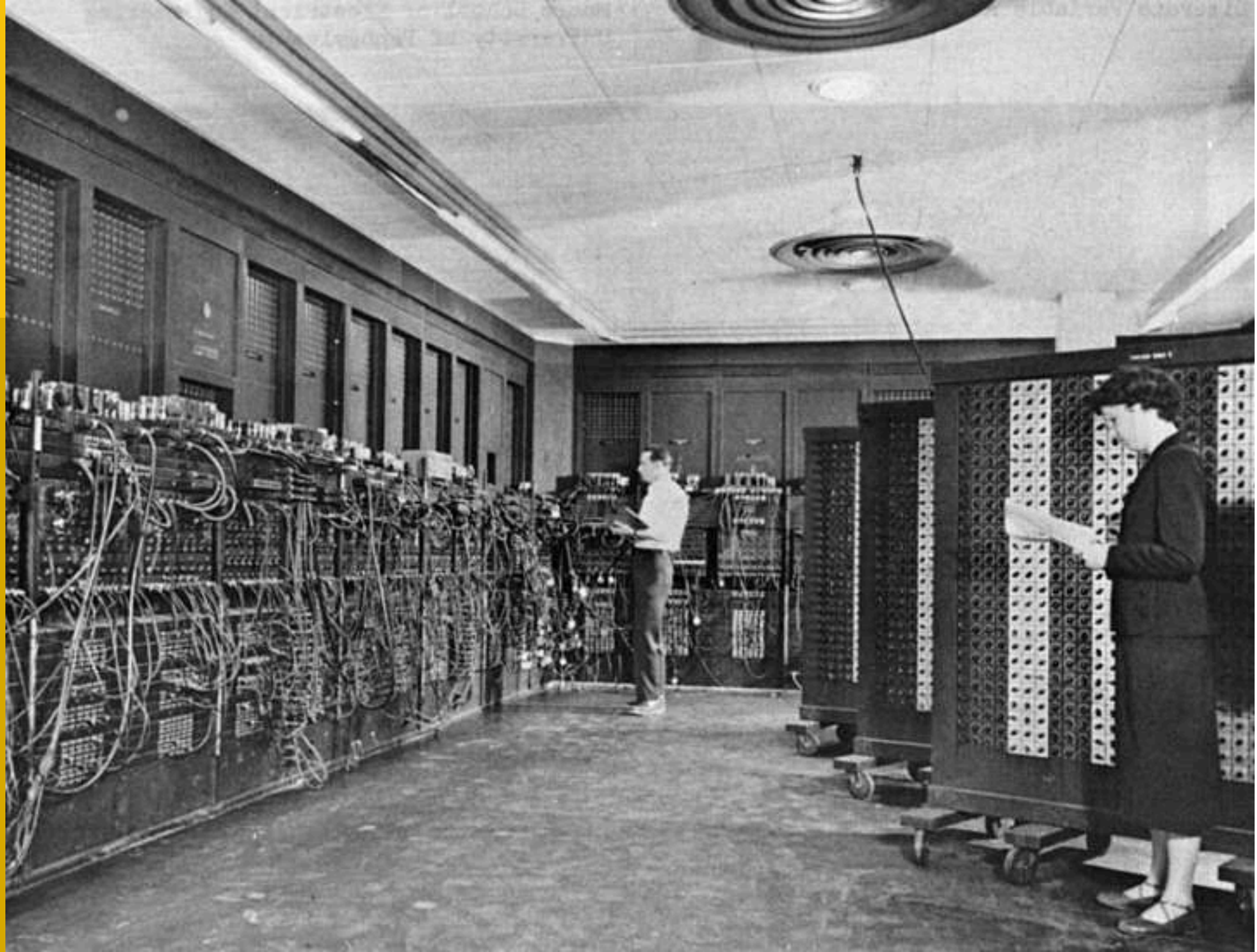
Tipos de Sistemas Operacionais

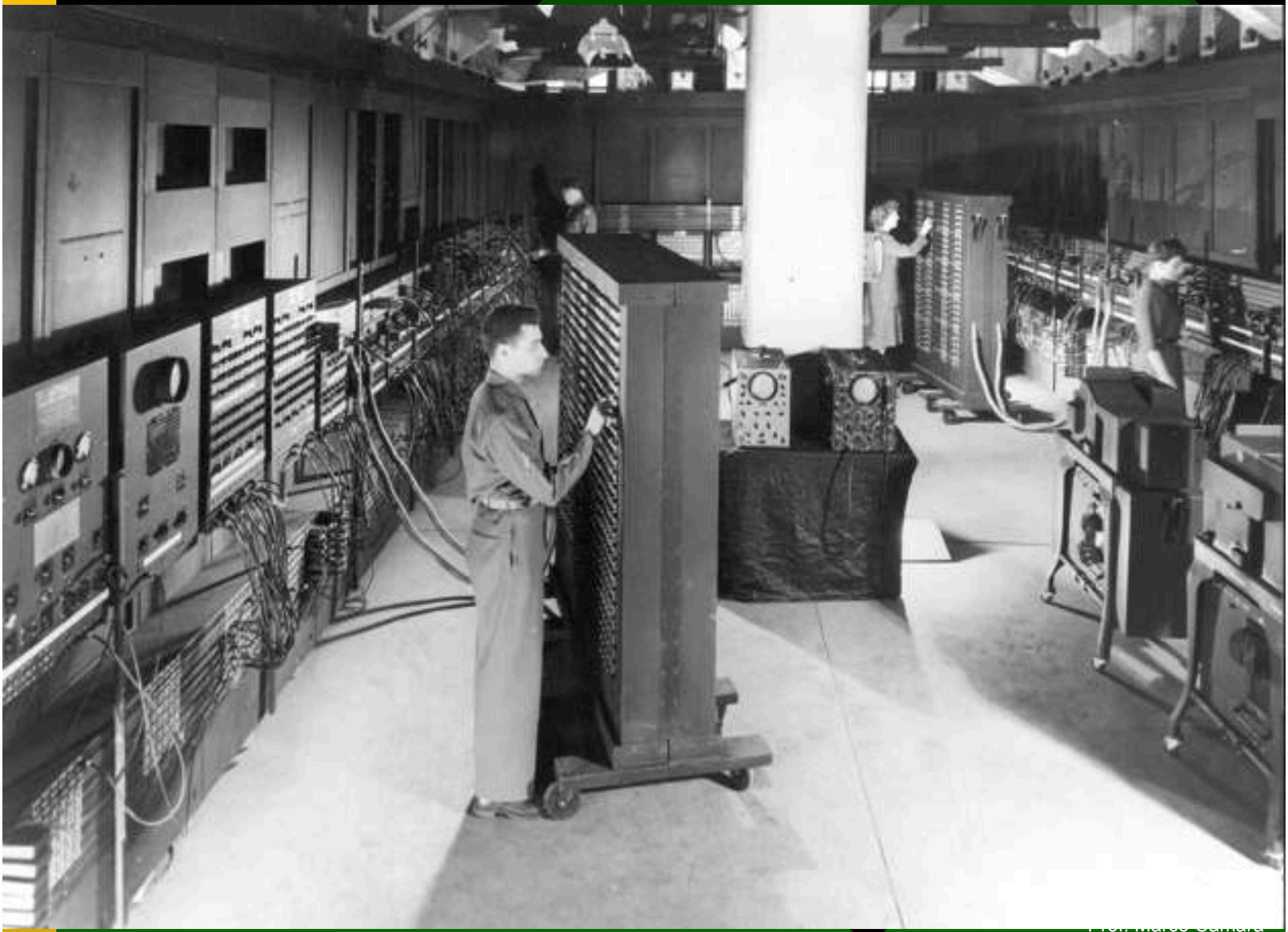
- Multiprocessamento
  - Computadores vistos originalmente como máquinas seqüenciais
    - Execução sequencial das instruções do programa
  - Sistemas Multiprocessados
    - Paralelismo - Simultaneidade
    - Execução de várias tarefas ou sub-tarefas

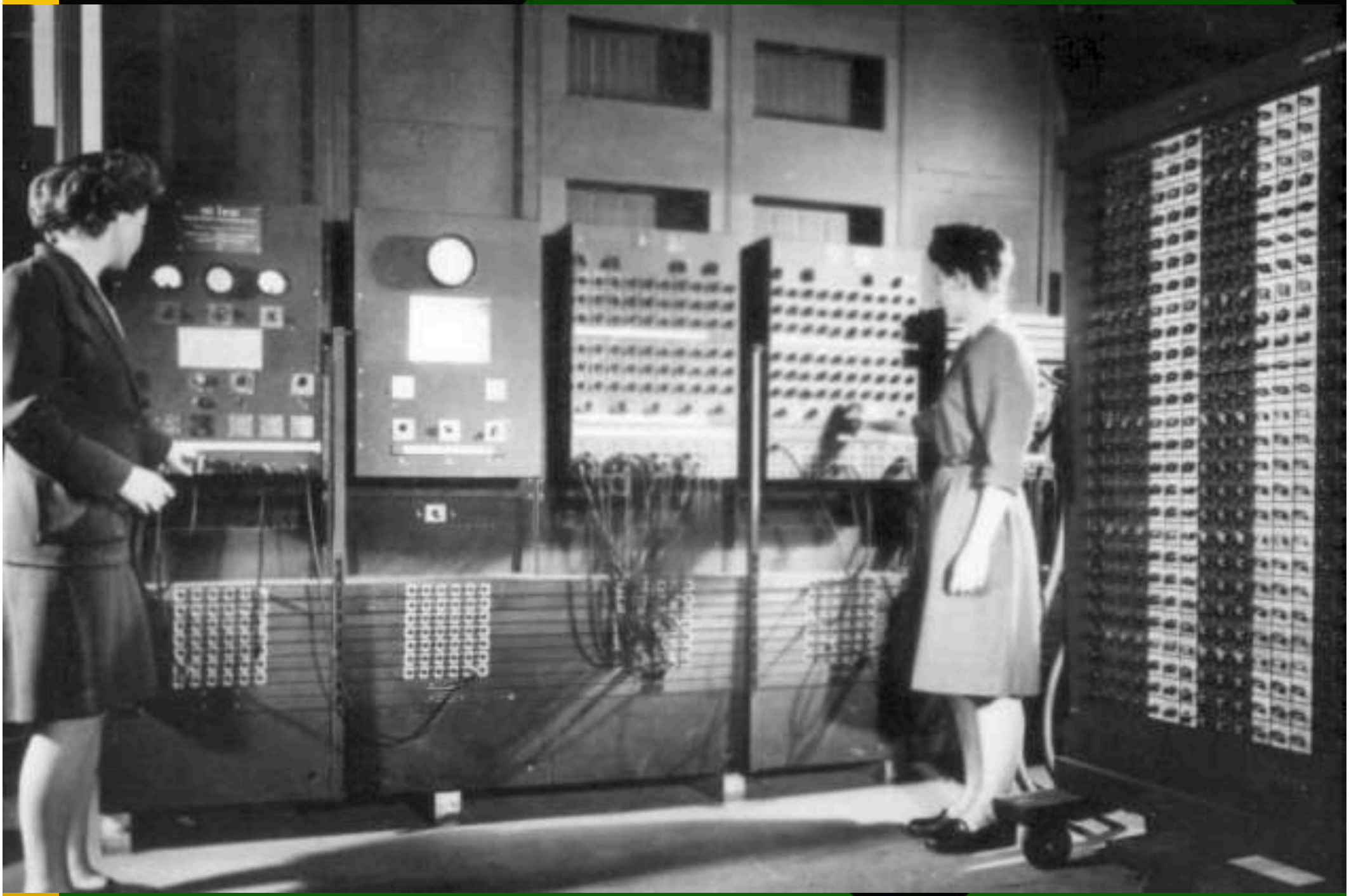
Histórico - 1ª Geração

(1945 - 1955)

- Primeiros computadores
- ENIAC (Electronic Numerical Integrator and Computer) - cálculos balísticos
- Utilização de válvulas (ENIAC possuía 18.000 válvulas)
- Programação em linguagem de máquina
- Não havia Sistema Operacional








THE ENIAC STORY
Celebrating six decades of electronic computing

arco Câmara

Histórico - 1ª Geração

(1945 - 1955)

- Outro exemplo: UNIVAC (*Universal Automatic Computer*)
  - Censo americano de 1950


Remington Rand


UNIVAC

Histórico - 2ª Geração

(1956 - 1965)

- Criação dos transistores
  - Aumento da confiabilidade (sem *bugs* ?)
  - Dimensão e consumo de energia bem menor
- Criação das memórias magnéticas
  - Acesso mais rápido aos dados
- Primeiras linguagens de programação:
  - Assembly e FORTRAN

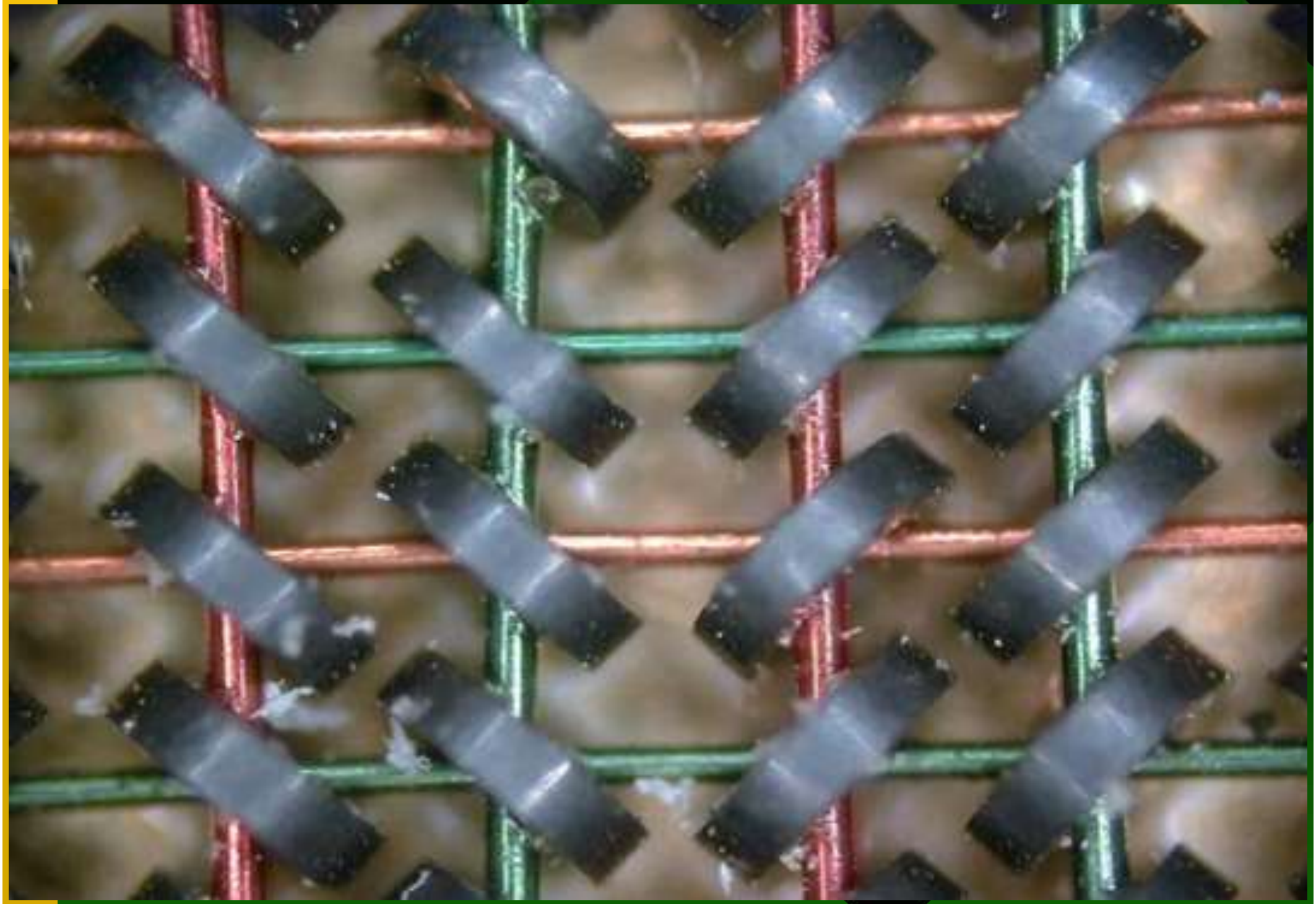
...ção

...*gs* ?)

...m menor

...icas

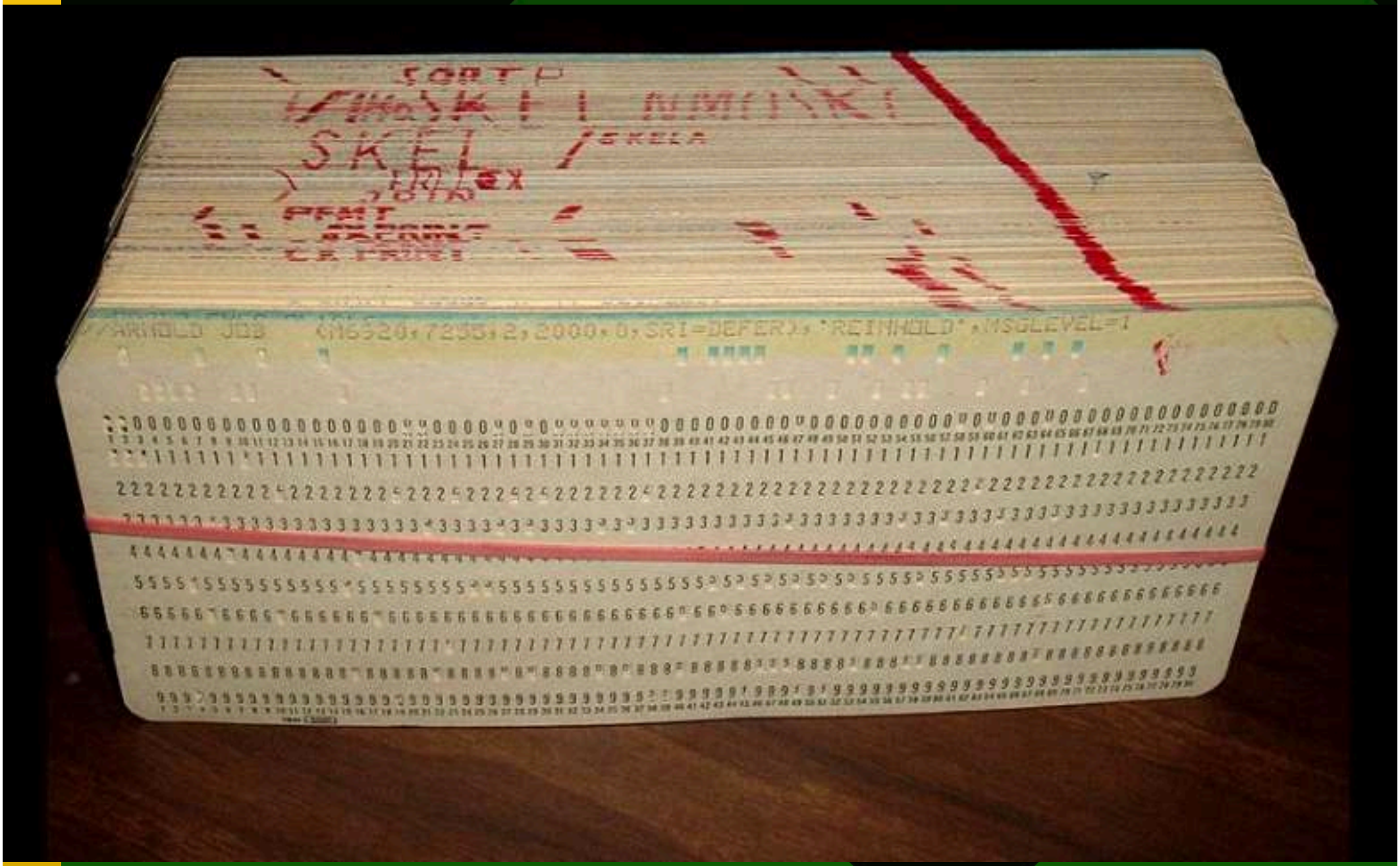
...amação:



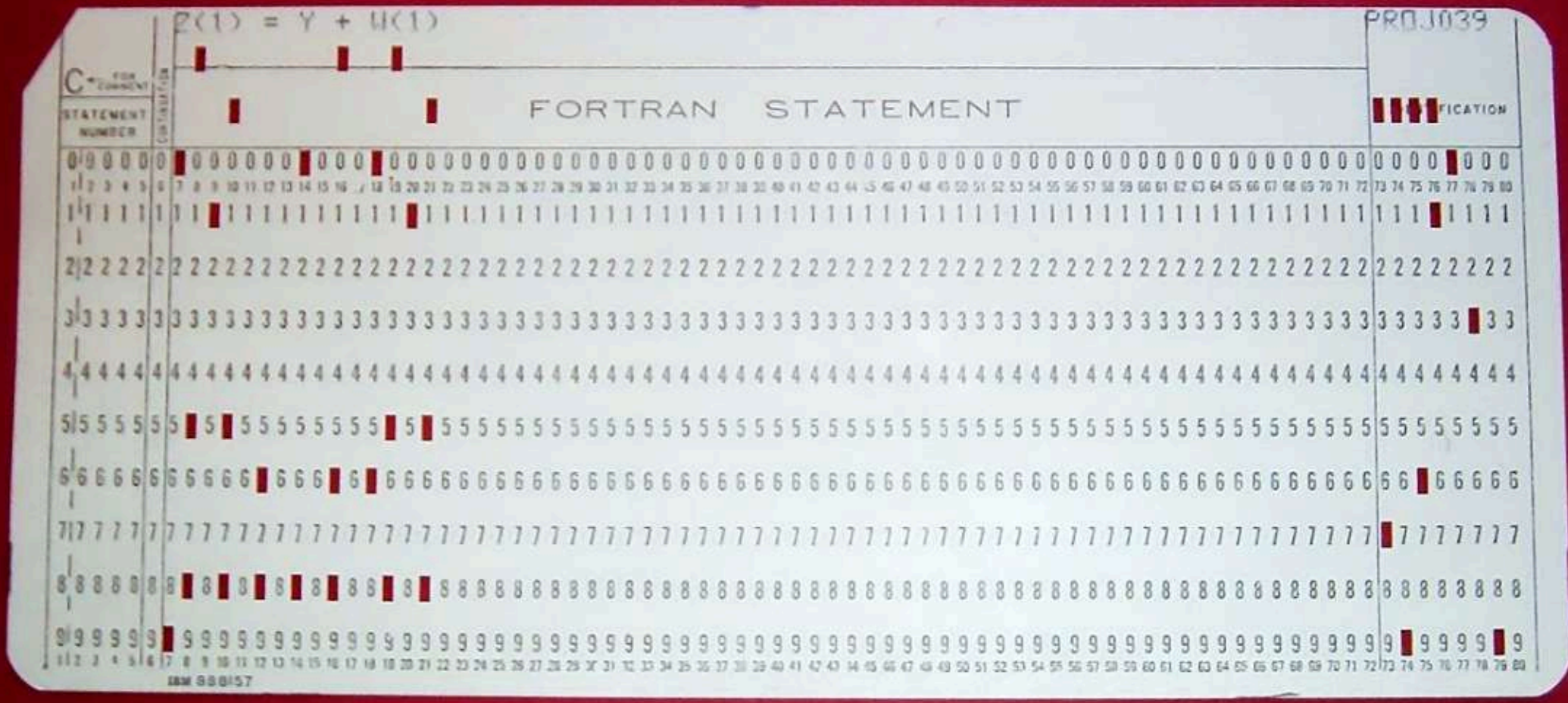
Histórico - 2ª Geração

(1956 - 1965)

- Cartão perfurado
- Processamento *batch*
- Rotinas para operações de Entrada e Saída (IOCS)
- Conceito de independência dos dispositivos


SORTP
SKEL
SKELA
PRNT
PRINT
//ARNOLD JOB (M6520,7235,2,2000,0,SRI=DEFER),'REINHOLD',MSGLEVEL=1

Histórico - 2ª Geração



Históricos - 2ª Geração

Histórico - 3ª Geração

(1966 - 1980)

- Monitor de vídeo e teclado (interação)
- *Time Sharing*
- Sistema Operacional UNIX
- Linguagem C
- Primeiros microcomputadores

Histórico - 3ª Geração

(1966 - 1980)

- Introdução dos Circuitos Integrados
  - Redução de custo e dimensões (+complexidade)
- Multitarefa / Multiprogramação
- Compartilhamento da memória
  - Primitivas com bloqueio
  - Sinais e interrupções

Histórico - 4ª Geração

(1981 - 1990)

- Aperfeiçoamento dos circuitos integrados
- Surgimento dos PC's e do DOS
- Estações de trabalho (monousuárias)
- Multiprocessadores
- Sistemas operacionais de rede e distribuídos.

Histórico - 5ª Geração ?

(1991 -)

- Arquitetura cliente-servidor
- Processamento distribuído
- Linguagem natural
- Segurança, gerência e desempenho do SO e da rede
- Consolidação dos sistemas de interfaces gráficas

Máquina de Níveis

Computador como máquina de níveis ou camadas:

- Nível 2 - Aplicações;
- Nível 1 – Sistema Operacional;
- Nível 0 – *Hardware.*

Nível 0 (*Hardware*)

Dispositivos Físicos;

Microprogramação;

Linguagem de Máquina.

Níveis 1 e 2 - *Software*

Sistema Operacional;

Linguagens de Programação

- Compiladores e Interpretadores;
- Máquinas Virtuais (Java);
- Ambientes de Desenvolvimento.

Aplicações.

Aula 03

Conceitos de *Hardware*

Definição de Processo

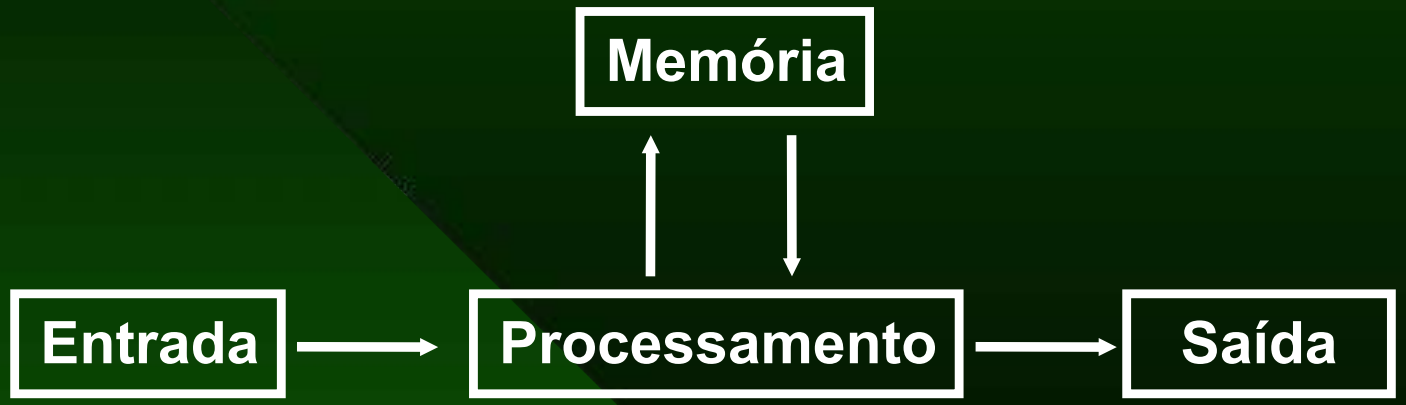
Escalonamento de processos - introdução

Conceitos

Hardware

- Três subsistemas básicos:
  - Unidade Central de Processamento;
  - Memória principal;
  - Dispositivos de entrada e saída.
- Subsistemas são também chamados de unidades funcionais;
- Implementações podem variar a depender da arquitetura.

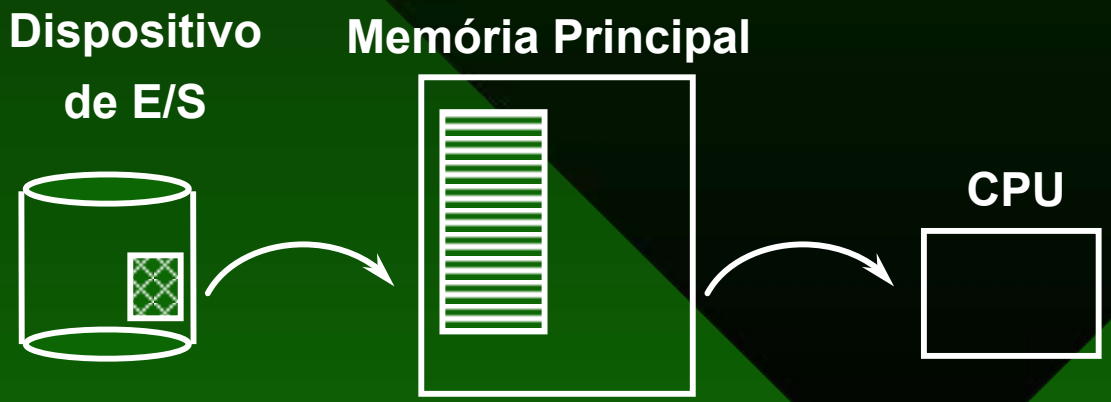
Modelo de Computador



Modelo de Computador

Execução de programas

- Programa armazenado em dispositivo de E/S
- Carga do programa na memória (SO)
- Execução das instruções (uma a uma) na CPU



CPU

Controla cada unidade funcional do sistema

Execução das instruções dos programas

- Aritméticas
- Comparação
- Movimentação

CPU

Conjunto de Instruções

- Instruções de máquinas disponíveis na CPU
- Tipos e quantidade variam de acordo com a CPU
- Compatibilidade de programas
- RISC x CISC

CPU

Desempenho de processadores

- Depende de:
  - Conjunto de instruções disponível
  - *clock*
  - Número de núcleos
- *Benchmark*
  - Usado para comparar processadores diferentes
  - Códigos específicos podem privilegiar um determinado produto !

Memória (definição)

- Externa (*storage*)
  - Armazena arquivos do SO, Aplicações e Dados;
  - Estante?
- Principal ou Interna
  - Provê o espaço de trabalho (mesa?)

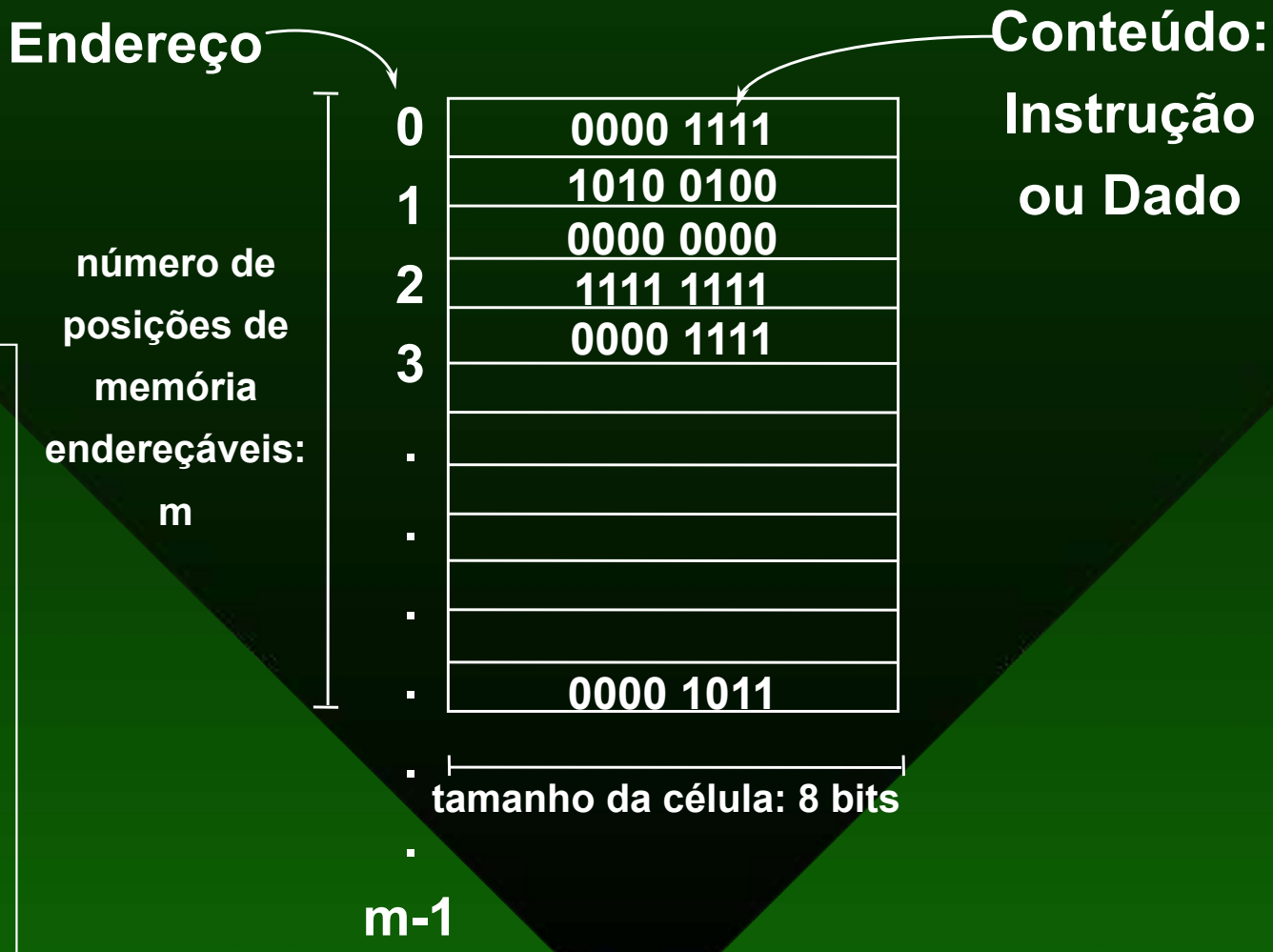
Memória Principal

- Armazena informações (*bits*)
  - Programas (instruções)
  - Dados
- Composta por várias unidades de acesso
  - Bit, Byte e Palavra;
  - Posições de memória.

Memória - Endereçamento

Cada posição de memória tem um endereço físico

Referência única de uma posição de memória



Memória - Tipos

Fisicamente, existem dois tipos:

- RAM, ou memória de leitura e escrita
  - Estática X Dinâmica;
  - A mais comumente especificada e manipulada;
  - Definida basicamente pela capacidade e barramento (performance).
- ROM
  - EPROM, EEPROM, Flash

Tipos de RAM

SRAM (*Static* RAM)

Construída com portas lógicas;

- Usa circuitos flip-flop para armazenar cada bit de memória, logo mantém as informações enquanto existir fonte de energia;
- Muito rápida;
- Cara e complexa (6 transistores), logo não é usada em grandes volumes.

DRAM (*Dynamic* RAM)

Construída c/componentes discretos

- Armazena cada bit em um capacitor conectado a UM transistor;
- O capacitor perde carga rapidamente, logo a memoria precisa ser “lembrada”;
- Simples e de baixo custo, compõe o maior volume da memorial usada nos computadores atuais.

Memória - Operação

Interligação Memória - Processador

Registradores de uso específico:

Memory Address Register - MAR - REM

Memory Data Register - MDR - RDM

Memória - Operação

Operação de Leitura da Memória

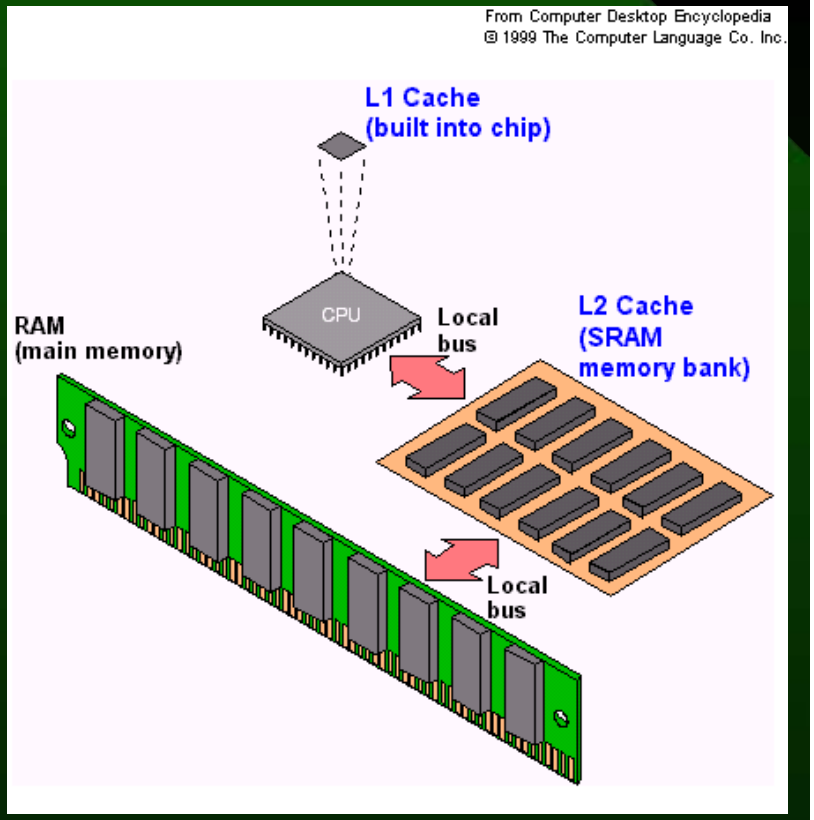
- CPU armazena endereço da célula a ser lida no MAR
- CPU gera sinal de controle indicando que a operação é de leitura da memória
- Memória recupera informação armazenada na posição endereçada e coloca no barramento de dados, chegando ao MDR

Memória - Operação

Operação de Gravação na Memória

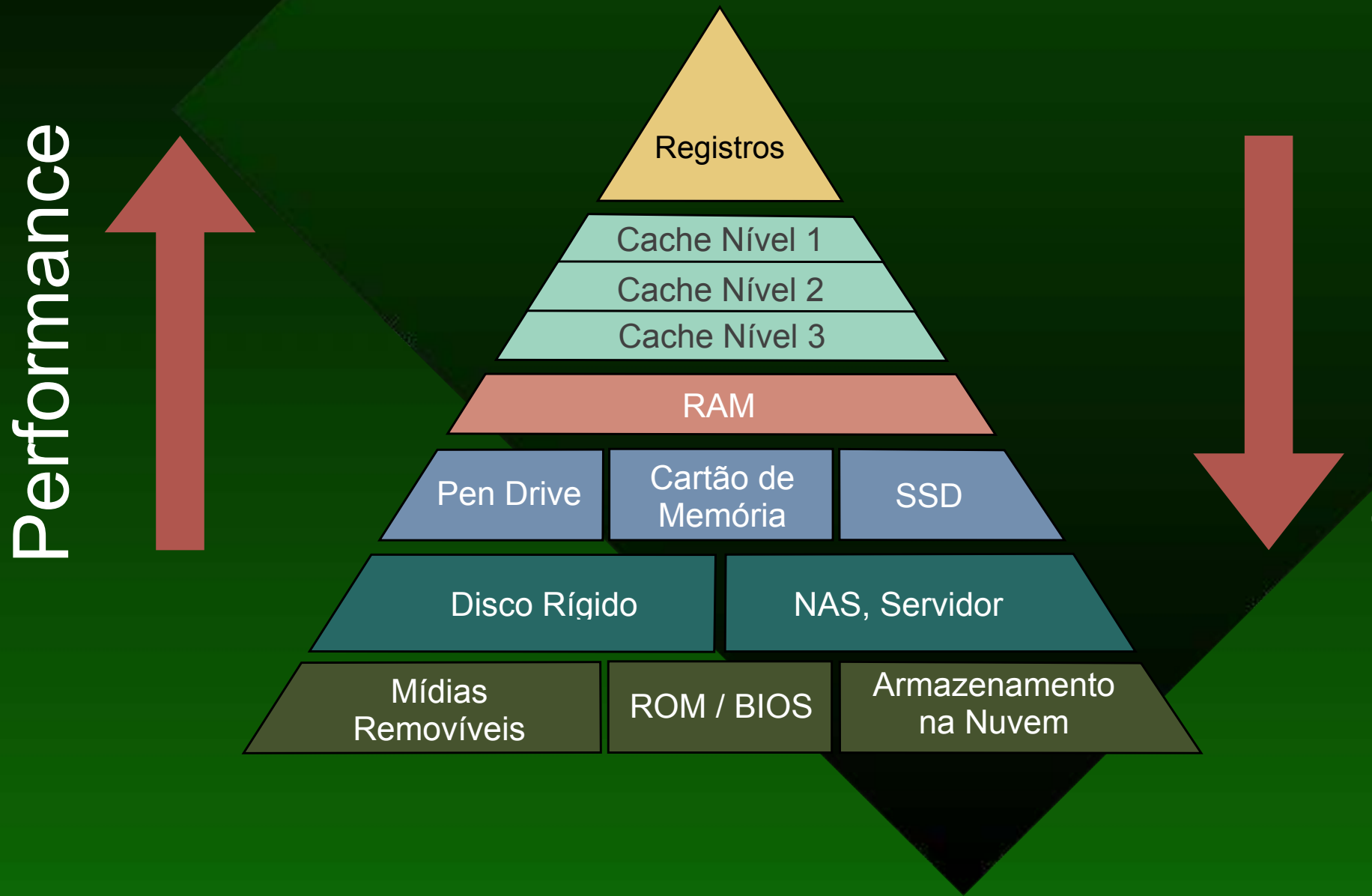
- CPU armazena endereço da célula a ser gravada no MAR
- CPU armazena a informação a ser gravada no MDR
- CPU gera sinal de controle indicando que a operação é de gravação na memória
- Memória armazena informação do barramento de dados na posição endereçada

Memória Cache



- Memória Cache
  - Memória de alta velocidade
  - Localizada entre CPU e Memória Principal
  - Aumento de desempenho com custo razoável
  - Algoritmo busca melhorar *Hit rate (*taxa de acertos)
  - Interna (L1,L2 … , ou primária) x Externa (Ln, ou secundária)

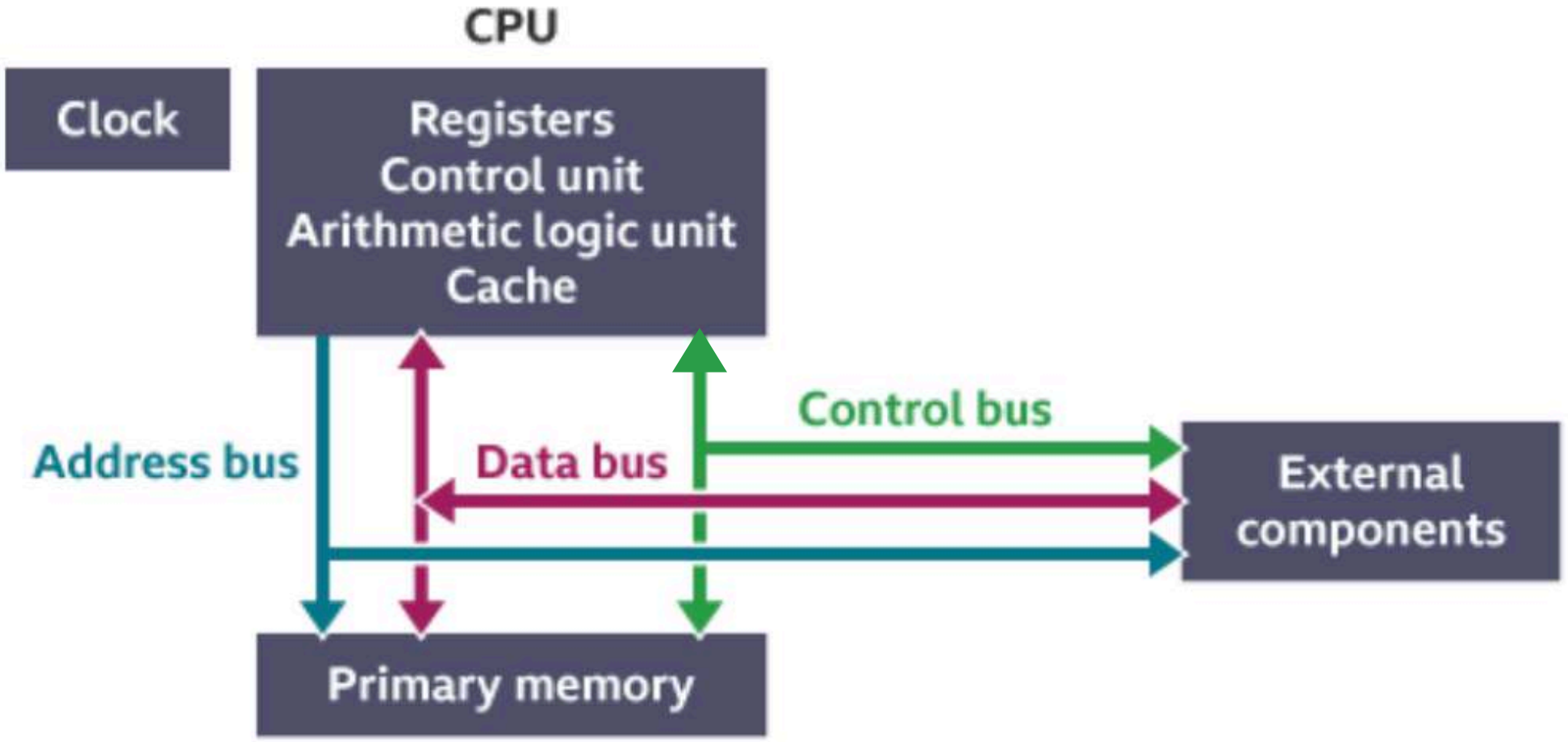
Hierarquia de Memória



Dispositivos de E/S

- Funções
  - Comunicação com meio externo
  - Memória Secundária e Interface Homem-Máquina
- Comunicação acontece através dos barramentos
  - Conjunto de vias paralelas para condução de sinais
  - Interligam CPU, Periféricos e Memória Principal

Barramentos



Barramentos

Barramento de Dados

- Número de *bits* por operação de E/S ou acesso à Memória
- Processadores de 8, 16, 32 ou 64 bits (palavra)
- Fluxo Bi-direcional (leitura / gravação)

Barramento de Endereços

- Capacidade de Armazenamento = 2^n
- Fluxo Uni-direcional (CPU ® MEM ou E/S)

Barramentos

- Barramento de Controle
  - Alguns exemplos de sinais de controle
    - *Clock;*
    - $\overline{Read}$ / *Write* ; $\overline{Write}$ / *Read;*
    - Controle de Interrupções;
  - Fluxo Uni ou Bi-direcional (depende do sinal de controle)

Definição de Processo

- Um programa em execução
- Não é o mesmo que Programa (entidade estática)
  - Programa é o código executável
  - Processo é o código executando
- Entidade dinâmica
  - Ex.: A execução de *prog.exe* 3 vezes gera 3 processos distintos do mesmo programa

Ambiente de um Processo

Todo processo precisa ter:

- Seção de texto
  - Código executável
- Seção de dados
  - Variáveis, estruturas
- Pilha do processo
  - Parâmetros etc
- Registradores, incluindo PC, SP, BP etc.

Conceitos

Para que os processos executem em ambiente multiprogramado, existe a gerência de:

- Compartilhamento da CPU, de memória, E/S etc

Nomenclatura

- Sistemas Batch
  - *Job*
- Sistemas de Tempo Compartilhado
  - Tarefa (*on-line*)

Estados do Processo

Mudanças de estado durante a execução:

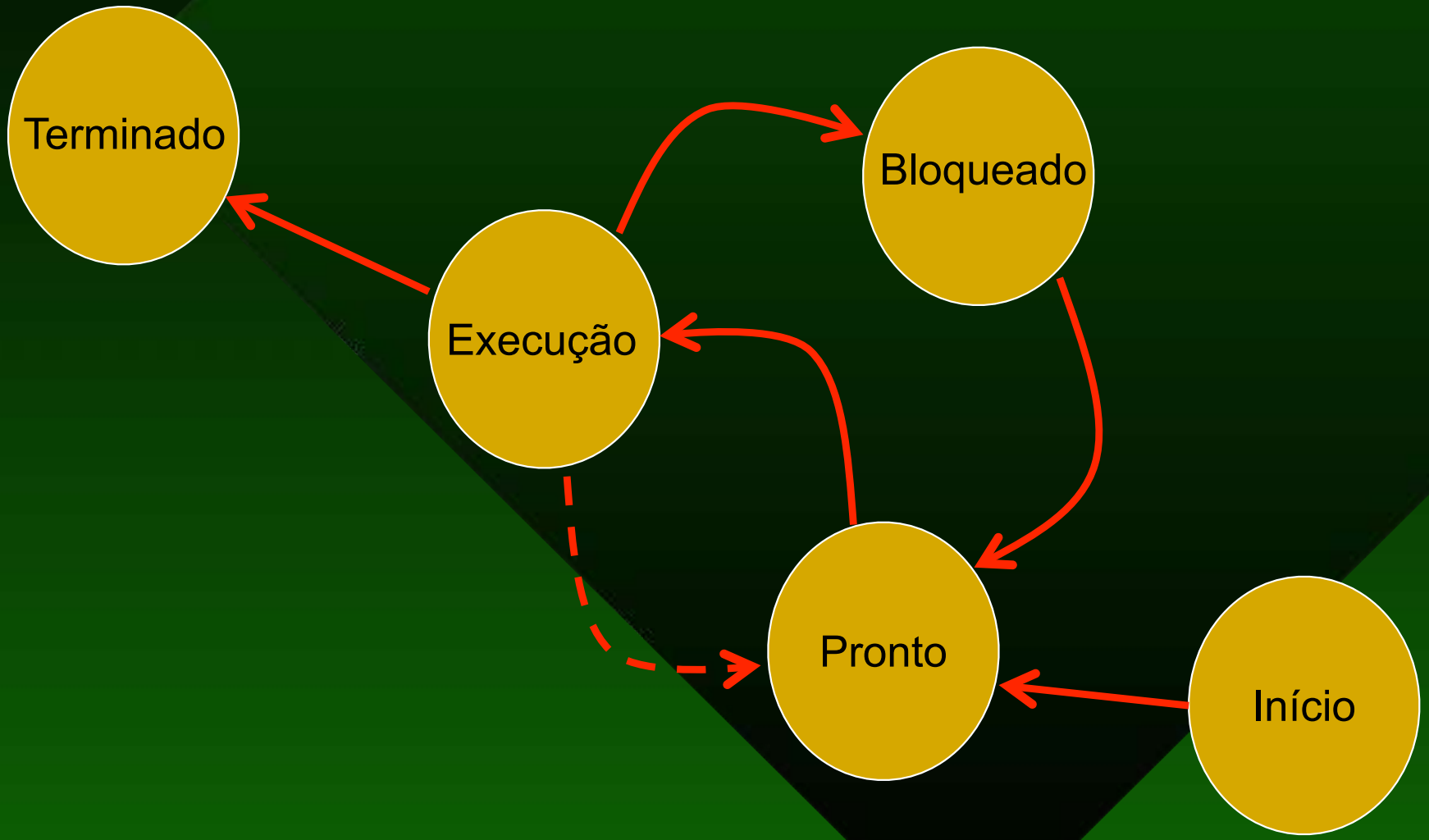
- Iniciando
  - O processo está sendo criado
    - SO Aloca Memória, Contexto (BCP)
      - Ex.: O usuário *clica* num arquivo executável no Explorer ou executa via CMD
- Executando
  - Instruções do processo estão sendo executadas
- Bloqueado
  - O processo está esperando algum evento externo

Estados do Processo

Mudanças de estado durante a execução:

- Pronto
  - O processo está aguardando chance de ser executado no processador
- Terminando
  - O processo está finalizando sua execução

Escalonamento de Processos



Aula 04

Escalonamento de Processos

Controle de Processo

- Bloco de Controle de Processo (BCP)
  - Área de Memória alocada pelo SO para gerenciar processos
  - Mantém informações sobre o processo
    - Estado do processo
    - Registradores da CPU, incluindo o PC
    - Informações para escalonamento
    - Informações para gerenciamento de memória
    - Informações de contabilização
    - Informações sobre operações de E/S
    - Ponteiros para Arquivos, *Socket* etc

Troca de Contexto

- Preempção
  - Ação de retirar um processo da CPU a qualquer instante e restaurá-lo como se nada tivesse ocorrido
    - Suportado por Interrupções (como pelo CLOCK) e pelo BCP
      - Ex.: Preempção por tempo de Quantum
        Preempção por I/O, Prioridade etc.
- Tempo da troca é considerado *overhead*
- Tempo depende de suporte de *hardware* e complexidade do SO

Escalonamento de Processos

- Manutenção de filas para controle dos processos
  - Fila de Processos – todos processos no sistema
  - Fila de Pronto – processos prontos
  - Filas de Bloqueado - processos aguardando E/S de dispositivo (interrupção) ou sinal
    - Múltiplas Filas – uma por dispositivo
      - Fila de Disco, Fila de CD, Fila de Teclado, de Rede
  - BCP efetua o encadeamento nas filas
  - Processos passam por várias filas

Escalonadores

Classificação de processos:

- I/O-BOUND – Intensivamente consumidor de E/S
  - Passa mais tempo fazendo operações de E/S do que utilizando a CPU
    - Processos Comerciais
- CPU-BOUND – Intensivamente consumidor de CPU
  - Passa mais tempo efetuando cálculos do que E/S
    - Processos Científicos/Matemáticos
- Híbrido

Escalonadores

Combinação adequada de processos CPU e I/O BOUND melhora o desempenho e flexibilidade do sistema

- Processo interativo (Ex.: Internet Explorer)
  - Tipicamente I/O Bound

Algoritmo de Escalonamento de CPU

Algoritmo do S.O. que determina qual o próximo processo a ocupar a CPU

- Executado quando ocorre estouro de Quantum ou interrupção do processo (I/O, Evento, Sinal etc.) ou o processo acaba;

Critérios mudam com características dos Processos

- *Batch, CPU Bound, I/O Bound*, Interativos

Metas do Escalonamento

Eficiência

- Manter a CPU ocupada 100% do tempo

Throughput

- Maximizar o número de processos (tarefas, jobs) executados em um dado intervalo de tempo

Turnaround

- Minimizar o tempo de um processo no sistema, desde seu início até o término
  - Tempo médio de execução
  - Fundamental a processos Batch

Metas do Escalonamento

Igualdade

Todo Processo tem direito de ocupar a CPU

Tempo de resposta

Minimizar o tempo decorrido entre a submissão de um pedido e a resposta produzida num processo interativo

Conflito entra Metas

- Atender a uma meta pode prejudicar outra
  - Qualquer algoritmo de escalonamento favorecerá um tipo de processo (*CPU Bound, I/O Bound*, Tempo Real, etc) em detrimento de outros
  - O propósito precisa ser geral

Tipos de Escalonamento

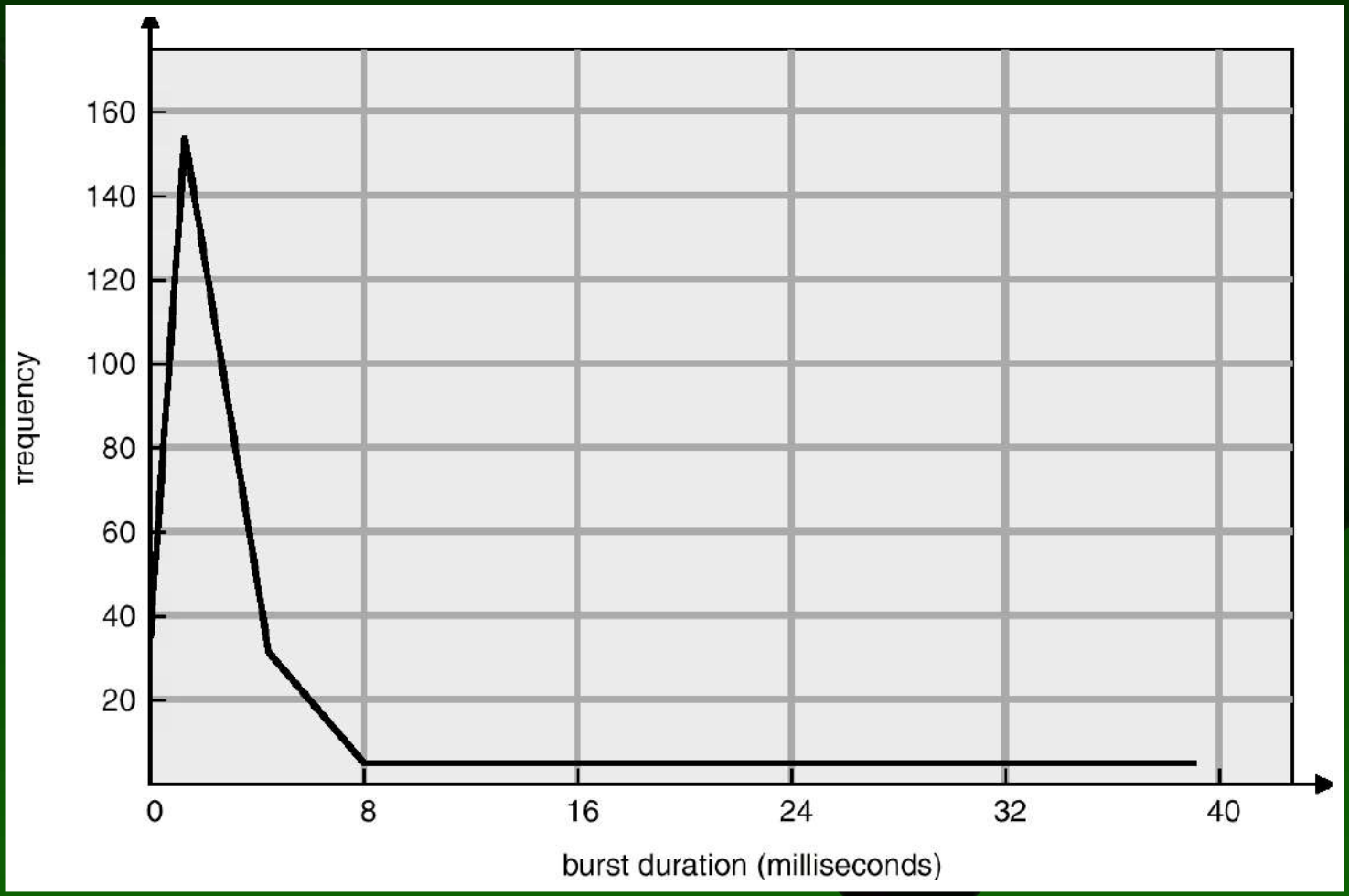
Escalonamento não-preemptivo

- Escalonamento Cooperativo;
- Processo mantém a CPU até terminar ou E/S;
- Não requer recursos especiais de hardware
  - Não existe Quantum (devolução voluntária do controle ao S.O.)
- No caso do Windows, utilizado até a versão 3.x.

Escalonamento preemptivo

- Requer temporizador na CPU (fatia de *quantum* ou Uso do *clock*);
- Requer suporte do SO para coordenar acesso a dados compartilhados de forma consistente (proteção).

Frequência de processos por duração de surto de CPU



Escalonamento FIFO

First Come First Served (FCFS, FIFO, PEPS)

Não preemptivo

Processo	Início	Duração (ut)
P_1	0	24
P_2	0	3
P_3	0	3

Escalonamento FIFO

Ordem de chegada dos processos:

$$P_1, P_2, P_3$$

Diagrama de Gantt

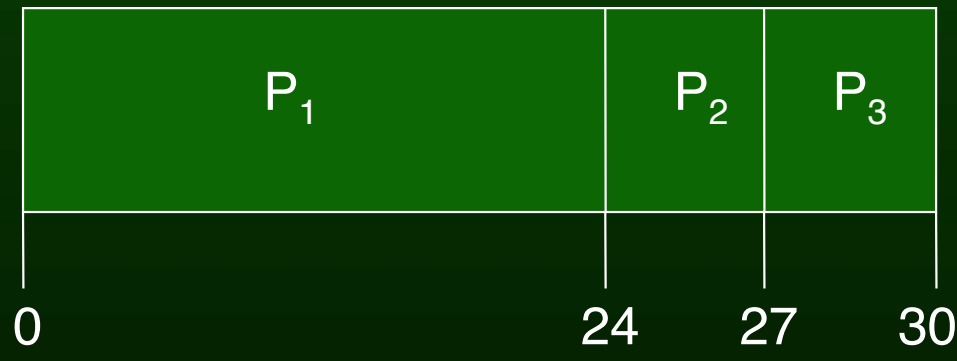
Escalonamento FIFO

Tempos de espera

$P_1 = 0$

$P_2 = 24$

$P_3 = 27$



Throughput = 0,1 (3/30)

Dica: Tempo de Espera = tempo no estado de Pronto.

Tempo médio de espera

(0 + 24 + 27) / 3 = 17

Escalonamento FIFO

Tempos de saída

$P_1 = 24$

$P_2 = 27$

$P_3 = 30$

Tempo médio de saída

(24 + 27 + 30) / 3 = 27

Escalonamento SJF

Shortest-Job-First (Menor Job Primeiro)

- Melhor: “próximo surto de CPU menor primeiro”

Usado para Processos *batch*.

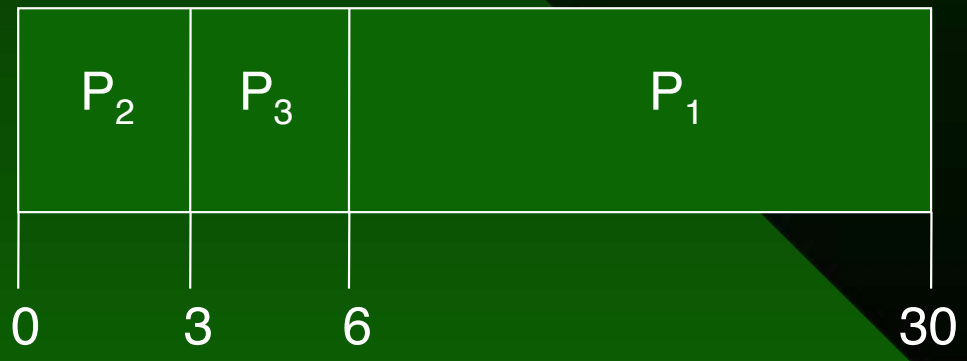
Execução diária permite determinar seu tempo total.

Escalonamento SJF (no mesmo exemplo)

Outra ordem de chegada

P_2 , P_3 , P_1

Diagrama de Gantt

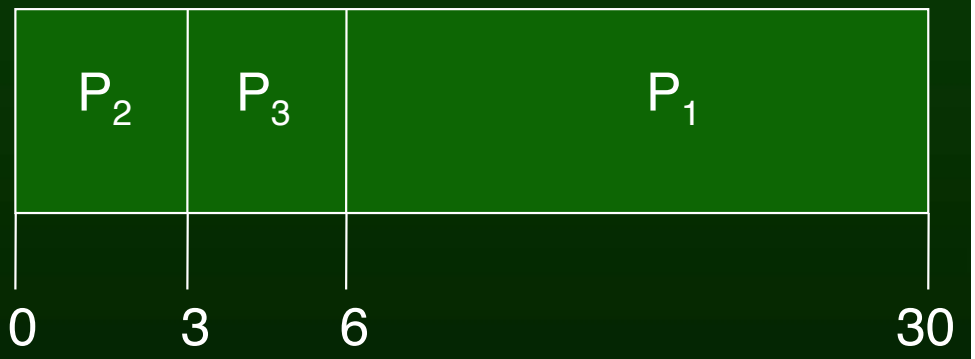


Escalonamento SJF (no mesmo exemplo)

FIFO ordenado (SJF / MPP)

Menor Processo Primeiro

Menor tempo de execução



Tempos de espera

$TEP_1 = 6$; $TEP_2 = 0$; $TEP_3 = 3$

Tempo médio de espera melhora

(6 + 0 + 3) / 3 = **3** (era **17** no FIFO)

Tempo médio de espera não é mínimo

Pode variar muito (com os surtos de CPU)

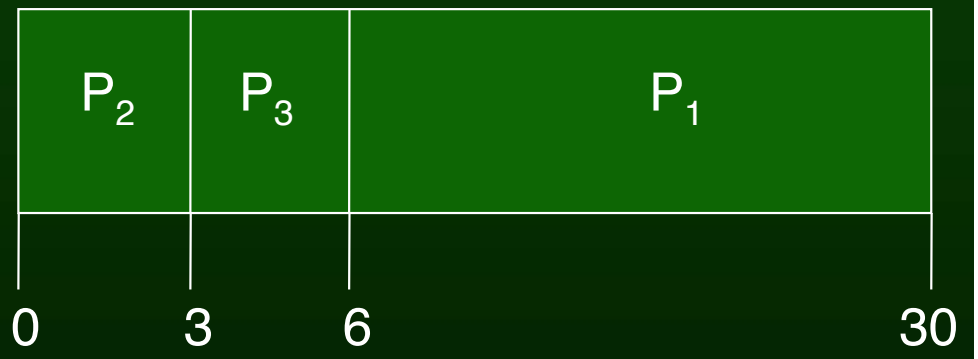
Efeito Comboio

Processos I/O bound esperam por CPU bound

Escalonamento SJF (no mesmo exemplo)

- Tempos de saída

  $P_1 = 30$; $P_2 = 3$; $P_3 = 6$



- Tempo médio de saída melhora

  (30 + 3 + 6) / 3 = **13** (era **27** no FIFO)

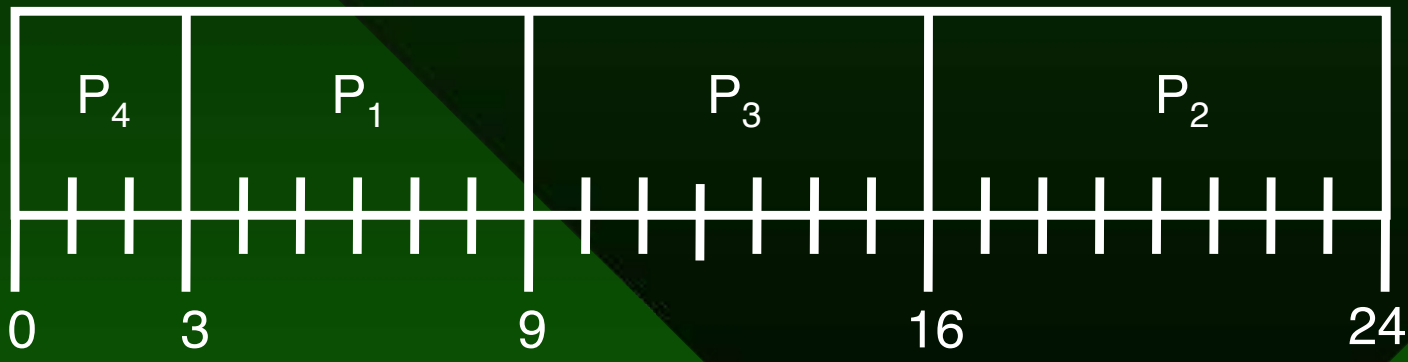
- Tempo médio de saída não é mínimo
  - Pode variar muito (com os surtos de CPU)

Escalonamento SJF (outro exemplo)

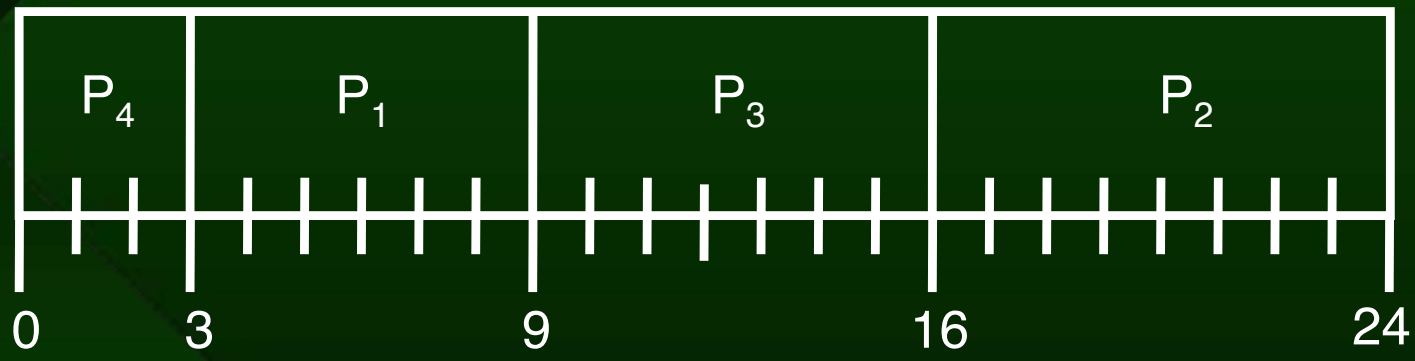
PID	Início	Duração surto
P_1	0	6
P_2	0	8
P_3	0	7
P_4	0	3

Tempos de espera

$P_1 = 3$; $P_2 = 16$; $P_3 = 9$; $P_4 = 0$



Escalonamento SJF (outro exemplo)



Tempo médio de espera melhora

(3 + 16 + 9 + 0) / 4 = 7

Para FIFO, nesta situação, seria 10,25 = (0+ 6+14+21)/4

Tempo médio de espera é mínimo

Algoritmo considerado ***ótimo***

Escalonamento SJF

- Problema: determinação exata da duração do próximo surto de CPU é impossível
  - SJF é usado para escalonamento de jobs em sistemas batch
    - Usuário especifica o tempo de CPU do job
  - Em escalonamento de CPU é usada estimativa
    - Baseada na duração dos surtos anteriores
      - Média exponencial

SJF com preempção

- Não preemptivo
  - Processo usa CPU até completar surto
- Preemptivo
  - Novo processo pronto com surto previsto (T_A)
  - Tempo restante previsto para o processo em execução (T_B)
  - Se $T_A < T_B \Rightarrow$ preempção por prioridade
  - *Shortest-Remaining-Time-First* (SRTF)

SJF com preempção

Processo	Instante de chegada	Duração de surto
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

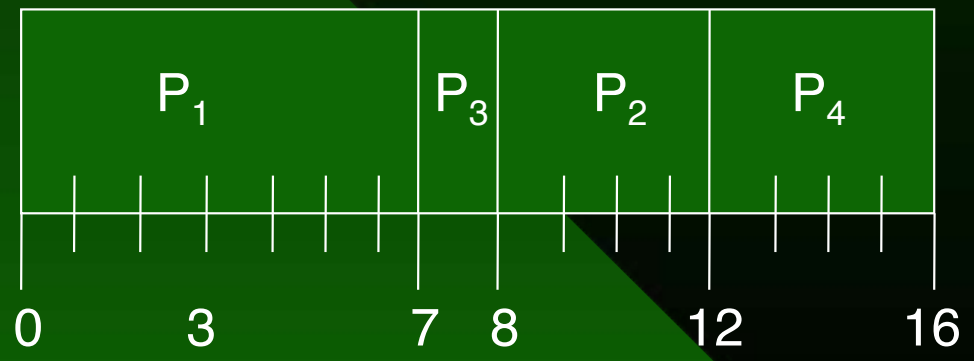
SJF com preempção

SJF não preemptivo

Tempo de espera médio = (0 + 6 + 3 + 7) / 4 = 4

$TEP_1 = 0$ $TEP_2 = 6$ (8 - 2)

$TEP_3 = 3$ $TEP_4 = 7$



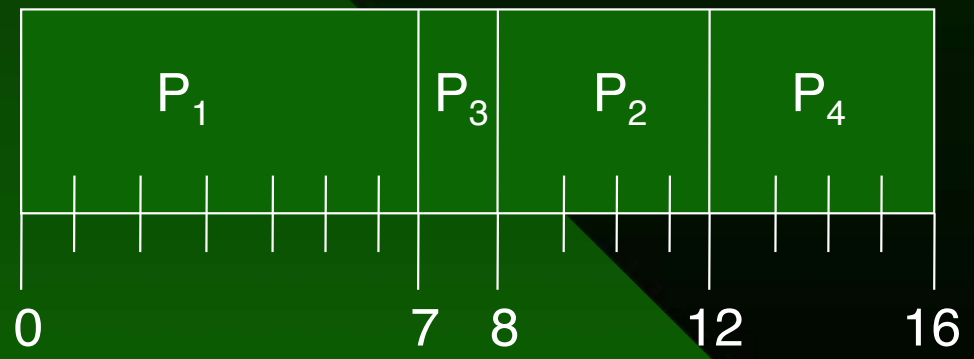
SJF com preempção

SJF não preemptivo

Tempo de saída médio = (7 + 10 +4 + 11) / 4 = 8

$TSP_1 = 7$ $TSP_2 = 10$

$TSP_3 = 4$ $TSP_4 = 11$



Tempo	Pronto	Execução	Term.	Observações
0	P1 (7)			
0		P1 (7)		
2	P2 (4)	P1 (5)		t(P2) < t (P1)
2	P1 (5)	P2 (4)		
4	P1(5), P3(1)	P2 (2)		t(P3) < t(P2)
4	P1(5), P2(2)	P3 (1)		
5	P1(5), P2(2), P4(4)		P3	
5	P1(5), P4(4)	P2 (2)		
7	P1(5), P4(4)		P2	
7	P1(5)	P4 (4)		
11	P1(5)		P4	
11		P1 (5)		
16			P1	

Proc.	Cheg.	Surto
P1	0	7
P2	2	4
P3	4	1
P4	5	4

SJF Preemptivo

Qual o tempo médio de espera ?

Qual o tempo médio de saída ?

SJF Preemptivo

Qual o tempo médio de espera ?

P1=0; P2=0; P3=0; P4=2

Média = 0,5

Qual o tempo médio de saída ?

P1=16; P2=5 (7-2); P3=1 (5-4); P4=6 (11-5);

Média = 7

Aula 05

Escalonamento de Processos (Tempo Real)

Escalonamento de Tempo Real

- Sistemas de tempo real crítico
  - Limites rígidos de tempo
  - SO garante execução no tempo ou rejeita
  - Exige software especial e hardware dedicado

Escalonamento de Tempo Real

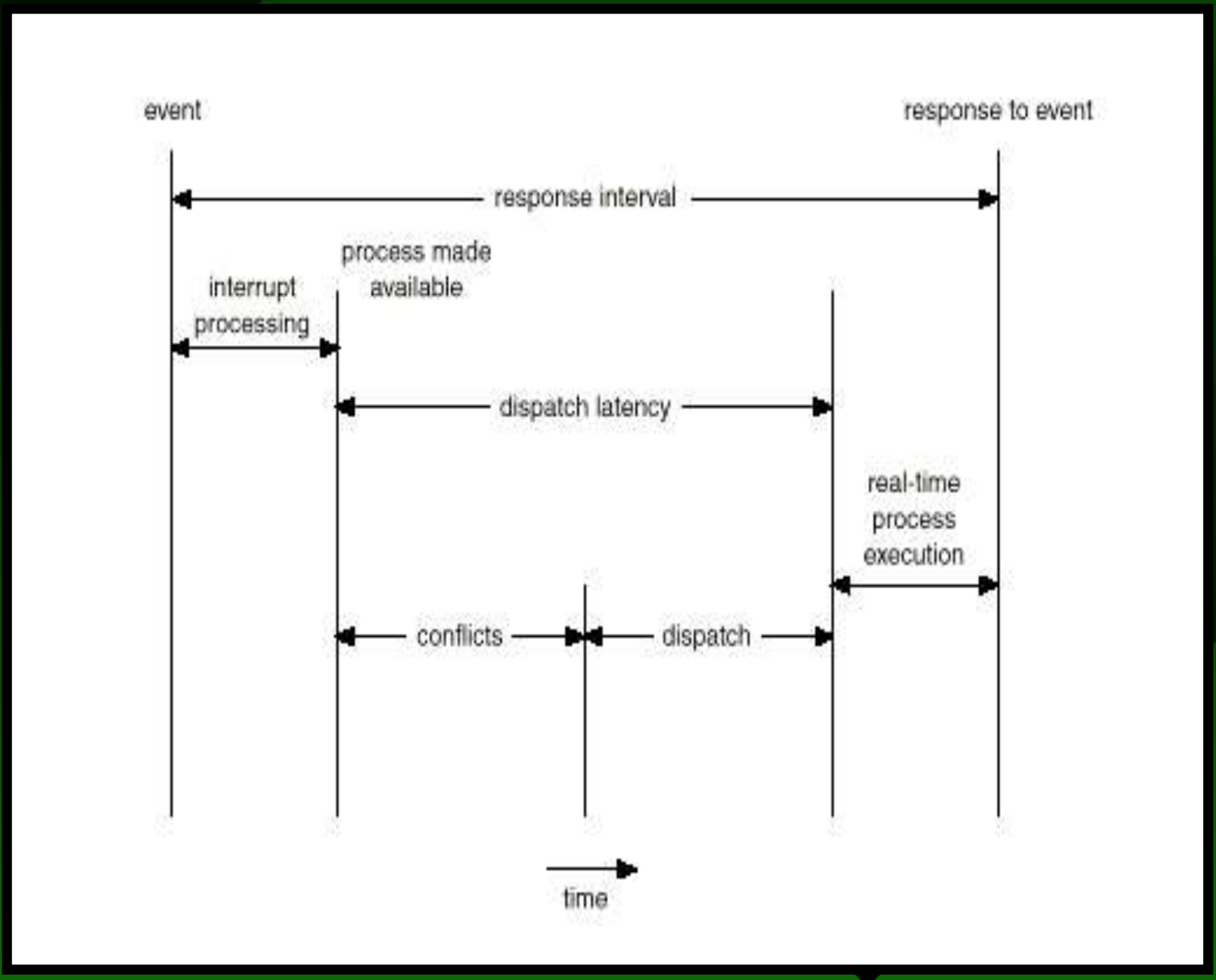
- Sistemas de tempo real não-crítico
  - Processos críticos com prioridade
    - Gera desbalanceamento do sistema
  - Suporte do SO
    - Escalonamento com prioridade
    - Não degradação da prioridade dos processos críticos
    - Latência de carga pequena
      - Chamadas ao sistema e operações de E/S
      - Pontos de preempção seguros
      - Todo Kernel preemptível (sincronização)
      - Protocolo de herança de prioridade

Escalonamento de Tempo Real

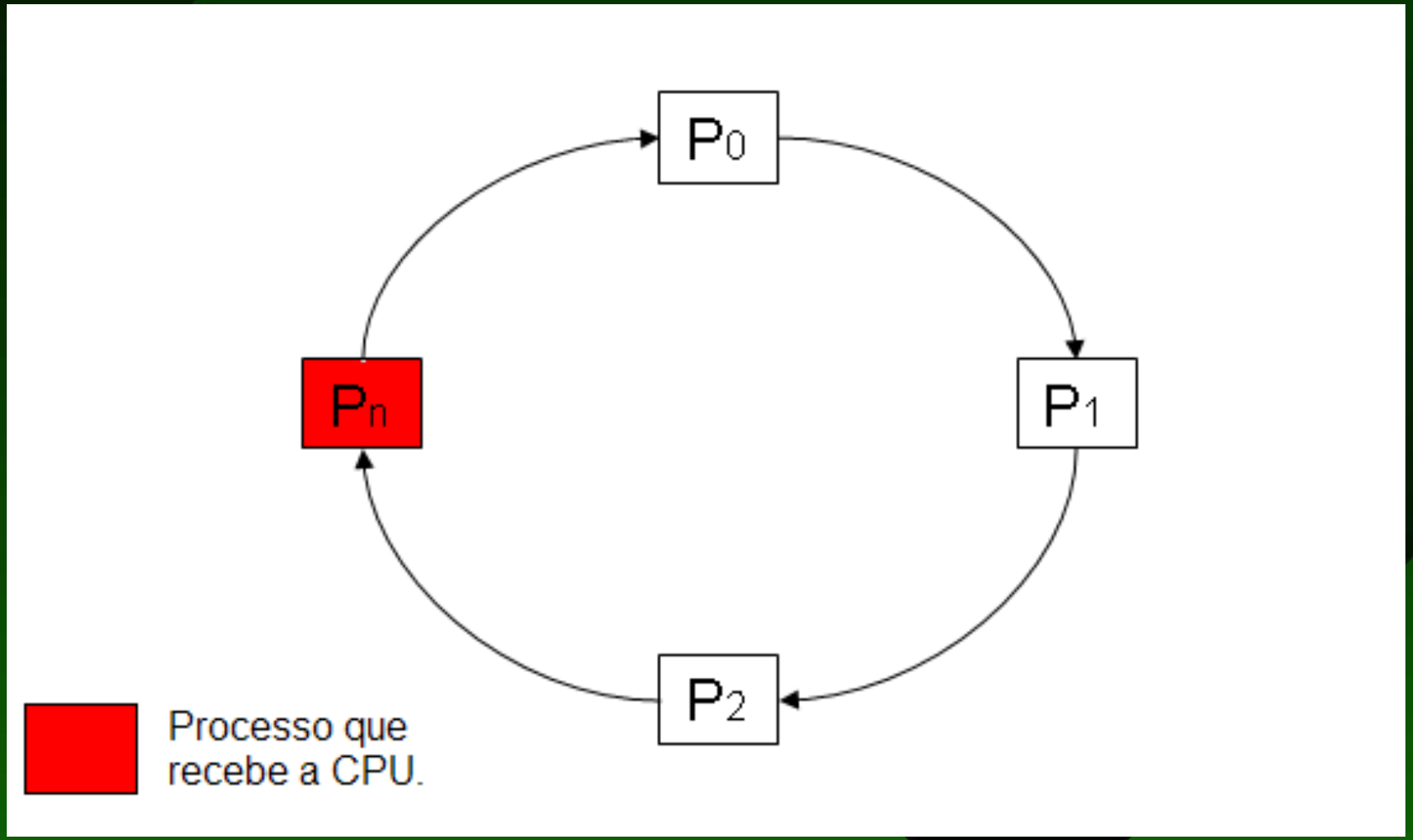
Dispatch Latency

- Descreve a quantidade de tempo que um sistema gasta para responder à requisição de um processo
- O tempo de resposta (TR) total consiste em:
  - TR de Interrupção
  - Dispatch latency
  - TR da aplicação

Escalonamento de Tempo Real



Escalonamento *Round Robin*



Escalonamento *Round Robin*

Round-Robin (revezamento circular)

- Método antigo e muito simples;
- Cada processo recebe uma fatia de tempo (Quantum), e é escalonado seguindo a ordem de uma fila circular;
- Sistema Preemptivo - Interrupção do Clock
- Se o Quantum acabar antes do término do processo (preempção), ou se o mesmo for bloqueado, o processo vai para o final da fila.

Escalonamento *Round Robin*

Resultados típicos:

- Elimina problemas de *starvation*;
- Tempo de espera médio é longo;
- Tempo de saída maior que SJF;
- Tempo de resposta melhor que SJF;
- Quantum grande -> FIFO;
- Quantum pequeno -> Baixa eficiência devido ao excesso de trocas de contexto.

Escalonamento *Round Robin*

- Análise da eficiência
  - Com quantum **q** e **n+1** processos prontos:
    - Tempo máximo de espera: n*q
- Um exemplo
  - Suponha uma fila de pronto com 100 processos, Quantum de 100 ms (valor típico);
  - Um processo interativo executa, faz uma requisição, vai para bloqueado e de lá para o fim da fila
    - Quando a resposta será entregue ao usuário do processo interativo?

Escalonamento por Prioridade

Cada processo tem uma prioridade

- Número inteiro dentro de limites (0 a 7 / 0 a 4095)
- Menor (ou maior) número ⇒ maior prioridade
  - Empate ⇒ Round Robin, FCFS

Starvation – Estagnação

- Bloqueio por tempo indefinido
- Soluções:
  - Decrementa prioridade a cada ciclo em execução;
  - *aging* (envelhecimento)

Escalonamento por Prioridade

Prioridade

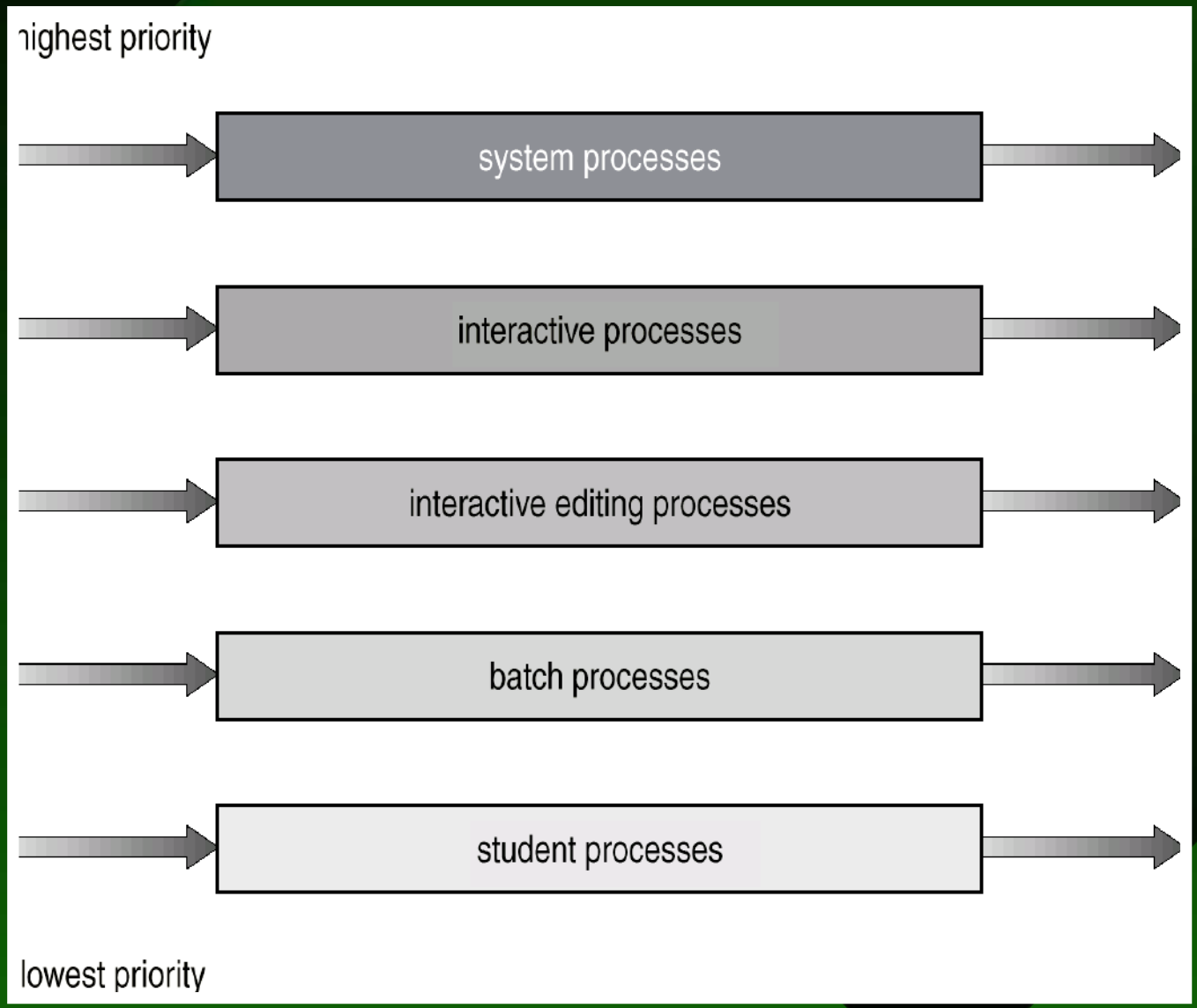
- Definida interna ou externamente;
- Estática ou Dinâmica;
- Alguns SO (Unix) permitem definir prioridade a um processo (comando *nice*);
- SO pode, por exemplo, aumentar prioridade de processos *I/O Bound*;
  - Ex: Prioridade = 1 / fração Quantum utilizada
- SJF: caso especial de prioridade ?

Pode-se classificar processos por classes de prioridade, e, dentro de cada classe, usar outro algoritmo de

Escalonamento por Múltiplas Filas

- Escalonamento preemptivo entre filas
  - Prioridade fixa: só atende filas menos prioritárias se as demais estiverem vazias
  - *Time slice* 80% para *foreground* com RR e 20% para *background* com FIFO

Escalonamento por Múltiplas Filas



Escalonamento por Múltiplas Filas

- Filas caracterizadas pelos surtos de CPU dos processos
  - *I/O bound* e interativos com mais prioridade
    - Passam a maior parte do tempo Bloqueados
  - Processos podem mudar de fila
  - *Aging* pode ser facilmente implementado
- Algoritmo preemptivo

Escalonamento com Múltiplos Processadores

- Escalonamento de CPU mais complexo
  - Existem sistemas com barramento de E/S privativo de determinado processador
- Várias filas de processos prontos
  - Possibilidade de desperdício de recursos

Escalonamento com Múltiplos Processadores

- Única fila de processos prontos
  - *Symmetric Multiprocessing* (SMP)
    - Cada processador faz seu escalonamento
    - Compartilhamento de estruturas de dados do SO
      - Sincronização
  - *Assymmetric Multiprocessing*
    - Escalonamento no processador mestre

Sincronismo de Processos

Sincronismo de Processos

É um problema inerente à gestão de recursos;

- Acesso simultâneo a recursos que precisam ser compartilhados;
- Pode gerar *dead-lock*, *starvation* ou inconsistência.

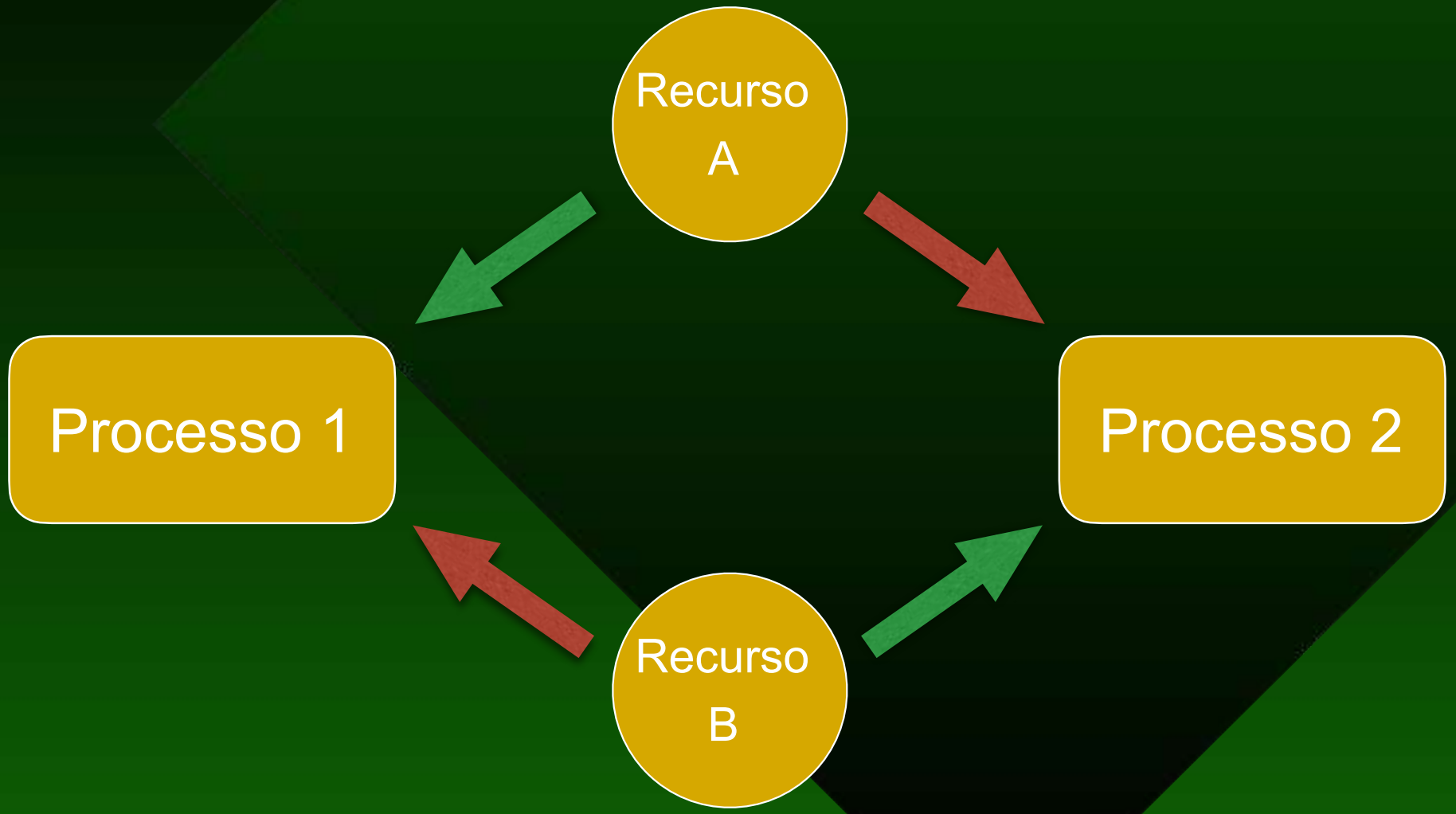
Deadlock

Sincronização de Processos

Dead-Lock

- Processo P1 bloqueia recurso R1 para seu uso;
- Processo P2 bloqueia recurso R2 para seu uso, e é bloqueado aguardando liberação de R1;
- Processo P1 é bloqueado aguardando R2.

Deadlock



Sincronização de Processos

Starvation

Política de escalonamento isola um ou mais processos, que nunca são escalonados, ou nunca conseguem ter acesso aos recursos necessários;

É necessário estudar algoritmos específicos para tratar este tipo de falha.

Sincronização de Processos

Inconsistência:

- Dois ou mais processos acessam a mesma base de dados, e alteram dados simultaneamente;
- Exemplos:
  - Processo A totaliza o campo X de todos os registros, enquanto que o Processo B altera o conteúdo de registros já contabilizados pelo Processo A -> Soma inconsistente?
  - Processos que manipulam informações de forma coordenada são interrompidos no meio de uma atualização. Dados inconsistentes?

Sincronização de Processos

Problemas clássicos

- Produtor / Consumidor;
- O jantar dos filósofos.

Sincronização de Processos

- Produtor / Consumidor
  - Dois processos compartilham um *buffer* de tamanho limitado
    - O processo produtor:
      - Produz um dado
      - Insere no *buffer*
      - Volta a gerar um dado
    - O processo consumidor:
      - Consome um dado do buffer (um por vez)

Sincronização de Processos

Sincronização de Processos

Produtor / Consumidor

O problema é:

Como garantir que o produtor não adicionará dados no *buffer* se este estiver cheio?

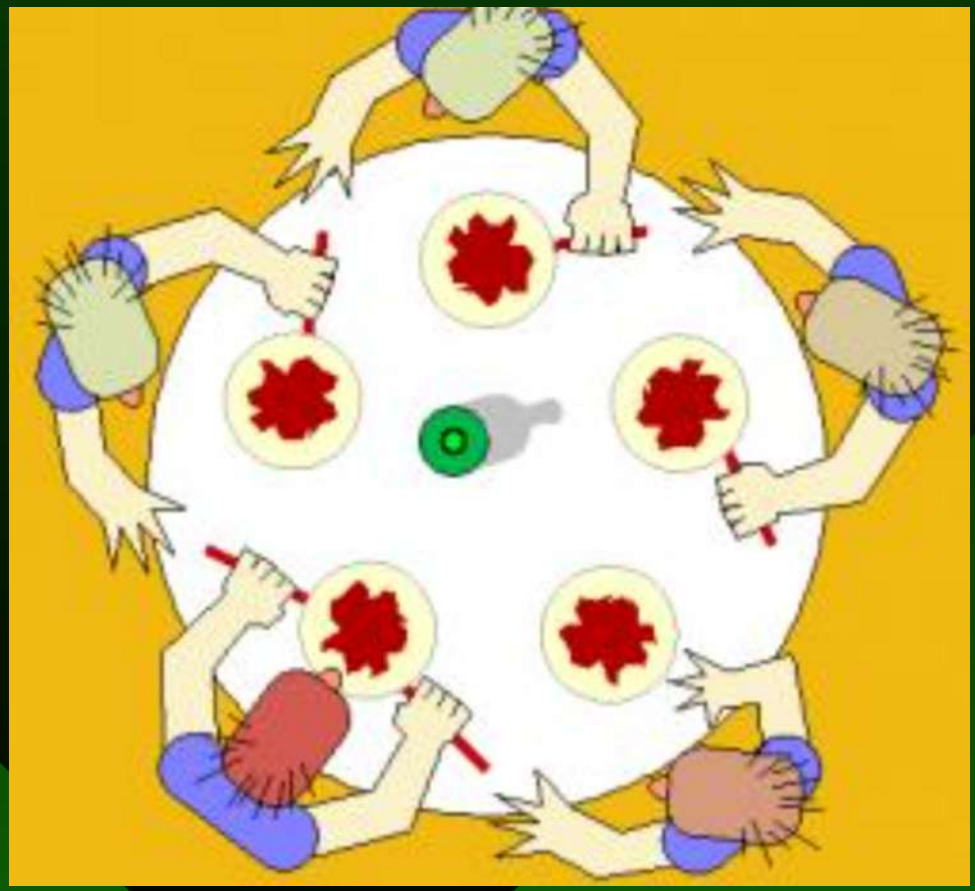
Como garantir que o consumidor não vai remover dados de um *buffer* vazio?

A solução não é trivial, pois os processos podem ser interrompidos! O que aconteceria se, antes de concluir o incremento de um contador, o processo fosse interrompido, e o contador fosse decrementado pelo outro processo?

Sincronização de Processos

O jantar dos filósofos

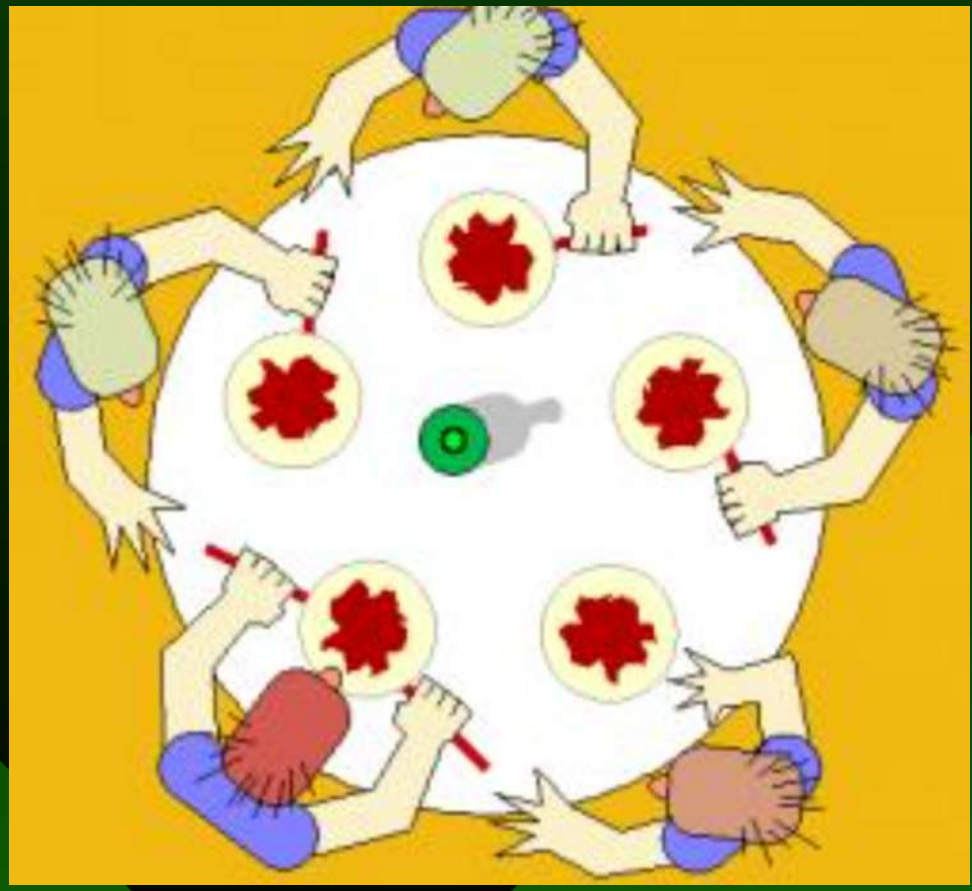
- Há cinco filósofos em torno de uma mesa;
- Um garfo é colocado entre cada filósofo;
- Cada filósofo deve, alternadamente, refletir e comer;
- Para que um filósofo coma, ele deve possuir dois garfos



Sincronização de Processos

O jantar dos filósofos

- O garfo somente pode ser pego se não estiver em uso por nenhum outro filósofo;
- Após comer, o filósofo deve liberar o garfo que utilizou;
- Um filósofo pode segurar o garfo da sua direita ou o da sua esquerda assim que estiverem disponíveis, mas só pode começar a comer quando ambos estiverem sob sua posse.



Aula 06

Sincronismo de Processos (continuação)

Resolvendo o sincronismo

Existem soluções de *hardware* e *software*;

Definição da "Região Crítica"

- Exclusão Mútua controlada por variável?
- Desabilitar interrupções?

Processos / *threads* que aguardam

- "Espera Ocupada";
- Bloqueio?

Região Crítica e Exclusão Mútua

Como tratar o acesso a recursos globais compartilhados, com instruções intercaladas pelo escalonador, fora do controle do programador?

Acesso realizado de forma atômica

Modelo de Código:

```
{ ***
SeçãoNãoCrítica;
ProtocoloDeEntrada;
SeçãoCrítica;
ProtocoloDeSaída;
****
}
```

Região Crítica e Exclusão Mútua

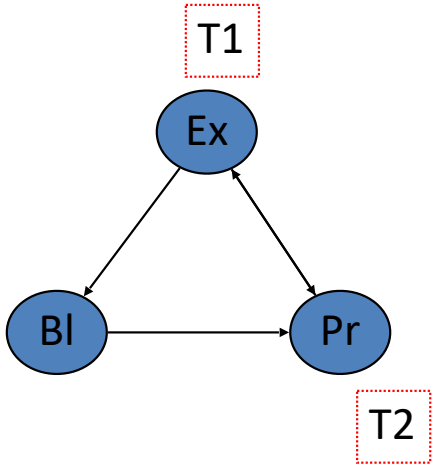
A região crítica é um trecho do código que deve ser executado de forma atômica, sem a intercalação com outro processo;

Metas:

- APENAS UM processo/*thread* estará na sua sessão crítica de cada vez, ou seja, deverá ser executado sob exclusão mútua;
- Um processo/*thread* não pode ser interrompido ou entrar em loop dentro dos protocolos de entrada/saída ou na sessão crítica;
- Nenhum processo/*thread* fora da sessão crítica pode bloquear outro processo/*thread*;

Threads Concorrentes

Chave = 1



- Thread 1

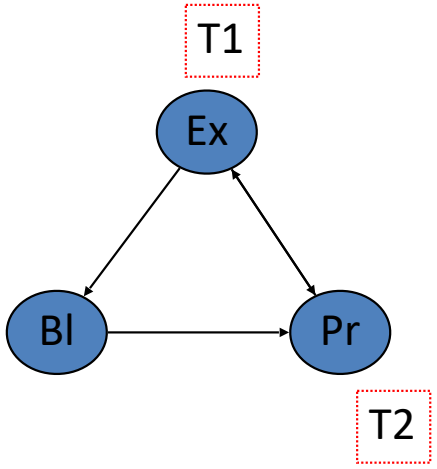
```
...
while (chave = = 0);
chave = 0;
RC
chave = 1;
...
```

- Thread 2

```
...
while (chave = = 0);
chave = 0;
RC
chave = 1;
...
```

Threads Concorrentes

Chave = 0



- Thread 1

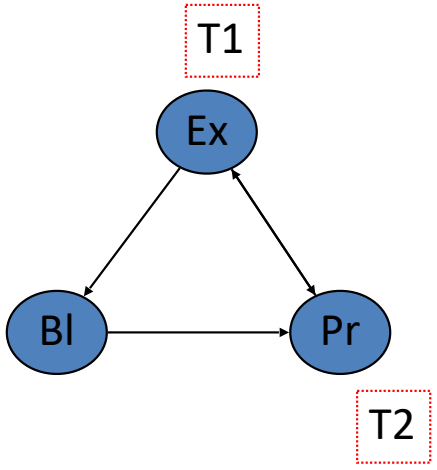
```
...
    while (chave = = 0);
    chave = 0;
     RC
    chave = 1;
...
```

- Thread 2

```
...
    while (chave = = 0);
    chave = 0;
     RC
    chave = 1;
...
```

Threads Concorrentes

Chave = 0



- Thread 1

```
...
   while (chave = = 0);
  chave = 0;
   RC
  chave = 1;
...
```

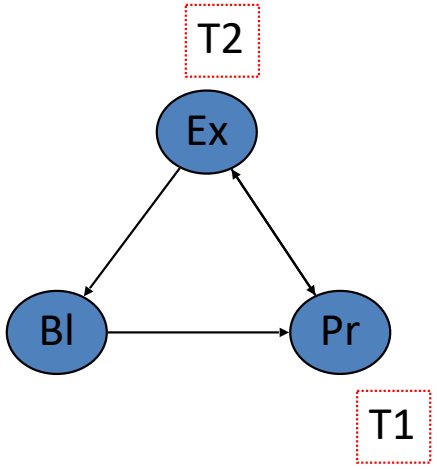
Preempção
Por tempo

- Thread 2

```
...
   while (chave = = 0);
  chave = 0;
   RC
  chave = 1;
...
```

Threads Concorrentes

Chave = 0



- Thread 1

```
...
   while (chave = = 0);
  chave = 0;
   RC
  chave = 1;
...
```

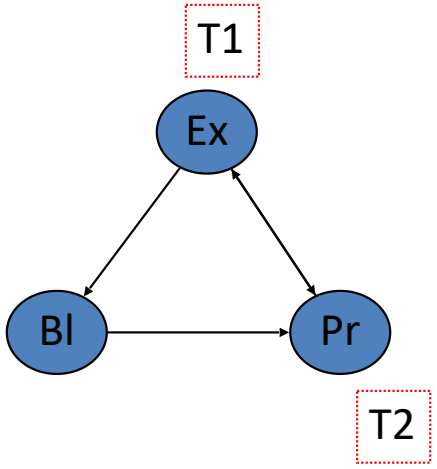
- Thread 2

```
...
   while (chave = = 0);
  chave = 0;
   RC
  chave = 1;
...
```

A thread 2 vai passar todo o tempo de quantum testando se chave = 0.

Threads Concorrentes

Chave = 0



- Thread 1

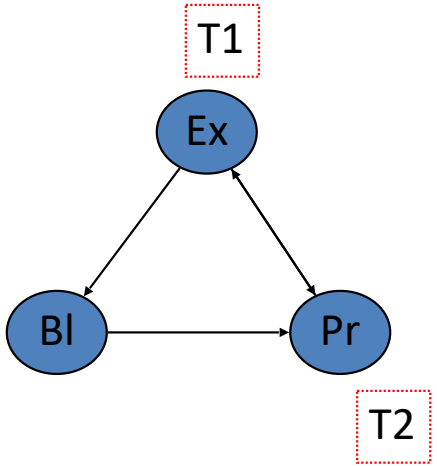
```
...
   while (chave = = 0);
  chave = 0;
   RC
  chave = 1;
...
```

- Thread 2

```
...
   while (chave = = 0);
  chave = 0;
   RC
  chave = 1;
...
```

Threads Concorrentes

Chave = 1



- Thread 1

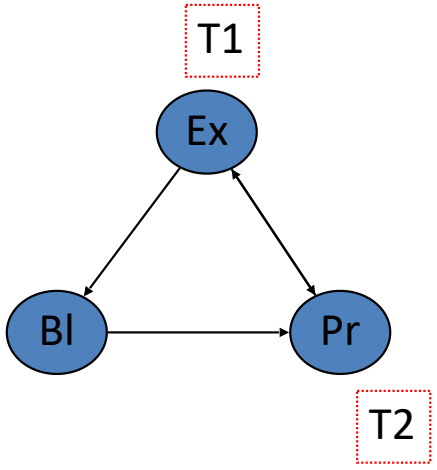
```
...
    while (chave = = 0);
   chave = 0;
    RC
→ chave = 1;
...
```

◆ Thread 2

```
...
→  while (chave = = 0);
   chave = 0;
    RC
   chave = 1;
...
```

Threads Concorrentes

Chave = 1



- Thread 1

```
...
   while (chave = = 0);
  chave = 0;
    RC
  chave = 1;
...
```

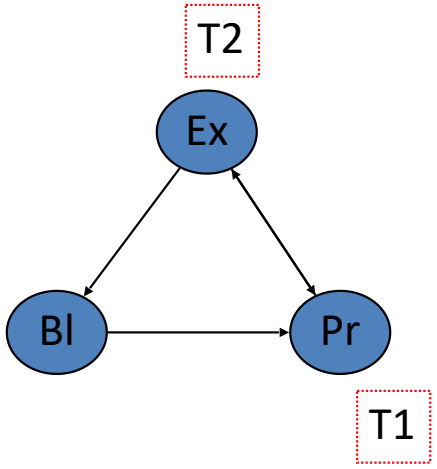
Preempção
Por tempo

◆ Thread 2

```
...
   while (chave = = 0);
  chave = 0;
    RC
  chave = 1;
...
```

Threads Concorrentes

Chave = 1



- Thread 1

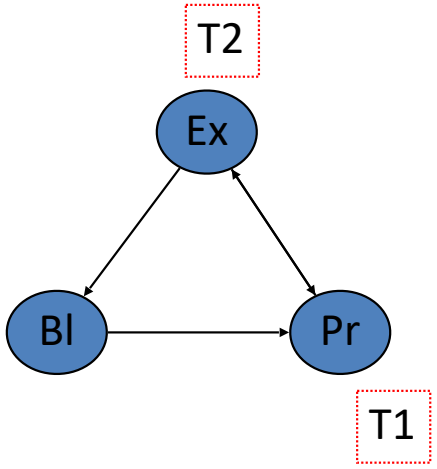
```
...
    while (chave = = 0);
   chave = 0;
     RC
   chave = 1;
...
```

- Thread 2

```
...
    while (chave = = 0);
   chave = 0;
    RC
   chave = 1;
...
```

Threads Concorrentes

Chave = 0



- Thread 1

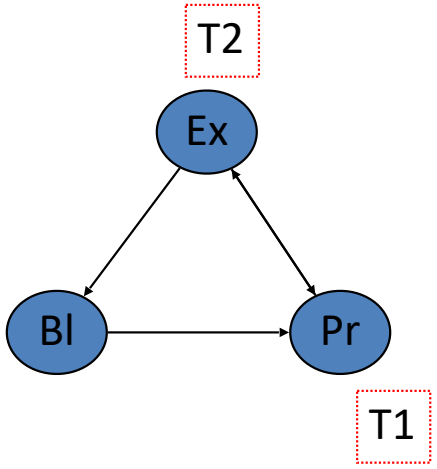
```
...
    while (chave = = 0);
   chave = 0;
    RC
   chave = 1;
...
```

◆ Thread 2

```
...
    while (chave = = 0);
   chave = 0;
    RC
   chave = 1;
...
```

Threads Concorrentes

Chave = 0



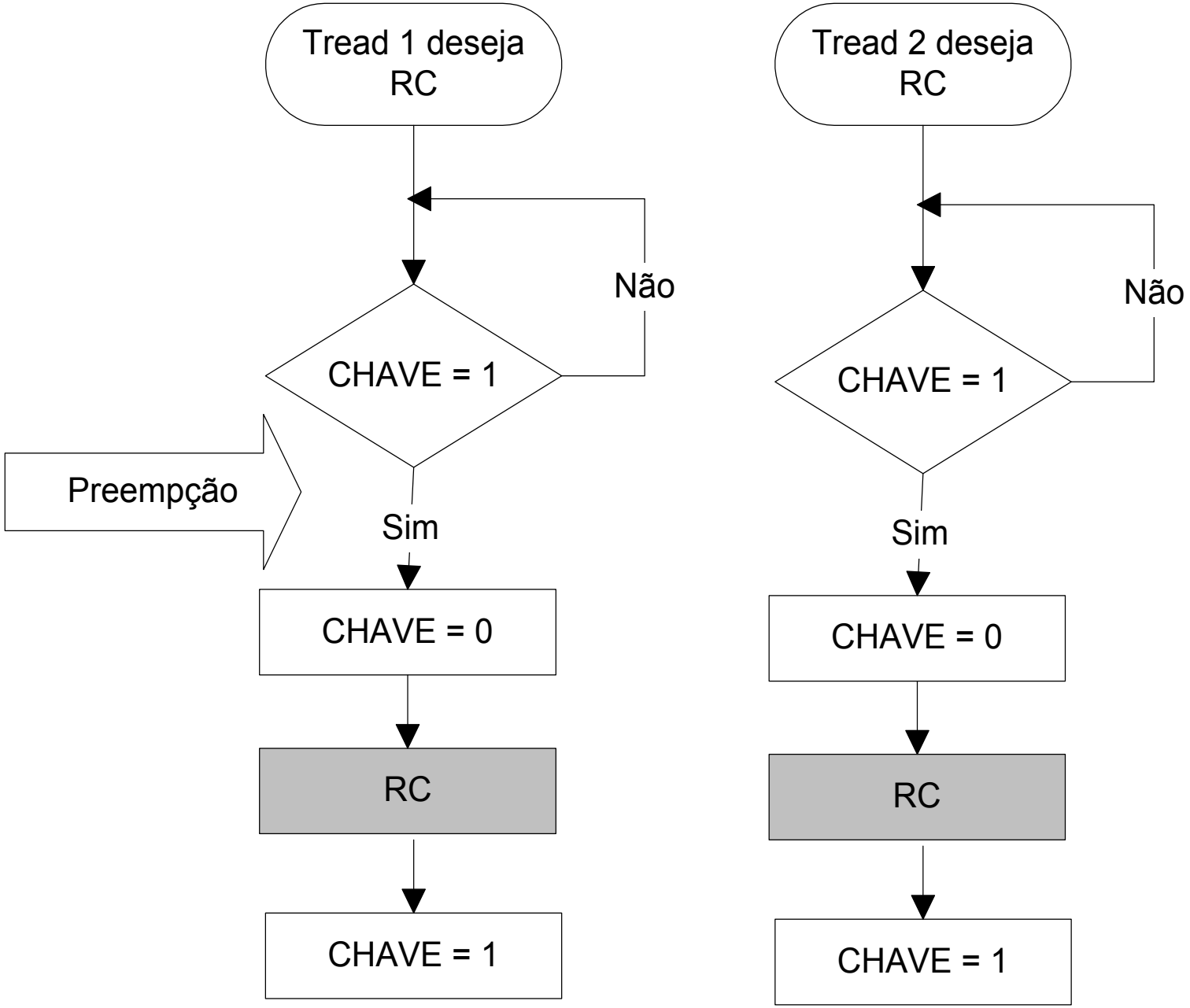
- Thread 1

```
...
    while (chave = = 0);
   chave = 0;
     RC
   chave = 1;
...
```

◆ Thread 2

```
...
    while (chave = = 0);
   chave = 0;
    RC
   chave = 1;
...
```

Existem problemas? Sim !



O ponto de falha é o teste !

Entre o teste e a atribuição de valor para o semáforo pode haver preempção?

- Sim ! A interrupção aguarda apenas o final da instrução corrente.

Soluções de *hardware*:

- Desabilitar interrupções;
- Usar uma nova instrução.

O ponto de falha é o teste !

Desabilitar Interrupções

- Pode comprometer o desempenho e integridade do sistema se não for reabilitada;
- Difícil em ambiente multiprocessado
  - Desabilita interrupção em 1 processador, e ... os outros continuam a executar processos/*threads*
- *Clock* é atualizado através de interrupções;
- Sem interrupções não há troca de processos por estouro de tempo.

O ponto de falha é o teste !

- Instrução *Test-And-Set* (TSL)
  - Uma das instruções do processador
  - Em uma única instrução:
    - Lê variável
    - Armazena valor num registrador
    - Atribui novo valor à variável
  - Execução atômica
    - Não pode ser interrompida

Sleep-WakeUp

A solução vista até agora envolve a chamada "espera ocupada", ou seja, perde-se tempo valioso do processador em um "*loop*" até o estouro do *Quantum*;

O uso do recurso de "*Sleep-WakeUp*" poderia bloquear o processo que encontra a região crítica "fechada".

Condições para o *Deadlock*

A preocupação com a execução simultânea de processos fez, na prática, que um processo aguardasse o término da execução de outro;

Mas o que acontece quando dois ou mais processos aguardam um pelos outros?

Condições para o *Deadlock*

Condições necessárias:

1. Exclusão Mútua
   Apenas um processo por vez pode alocar e manipular um recurso
2. Posse e Espera;
3. Não Preempção;
4. Espera Circular.

Ocorrência precisa ser simultânea

Se ao menos uma não ocorrer, não haverá *DeadLock*

Condições para o *Deadlock*

Condições necessárias:

1. Exclusão Mútua
2. Posse e Espera

   Um processo, de posse de um recurso, pode solicitar novos recursos
3. Não Preempção;
4. Espera Circular.

Ocorrência precisa ser simultânea

Se ao menos uma não ocorrer, não haverá *DeadLock*

Condições para o *Deadlock*

Condições necessárias:

1. Exclusão Mútua
2. Posse e Espera
3. Não Preempção

   Um recurso não pode ser removido explicitamente do processo
4. Espera Circular.

Ocorrência precisa ser simultânea

Se ao menos uma não ocorrer, não haverá *DeadLock*

Condições para o *Deadlock*

Condições necessárias:

1. Exclusão Mútua
2. Posse e Espera
3. Não Preempção
   Um recurso não pode ser removido explicitamente do processo
4. Espera Circular
   Ocorre CICLO no GRAFO de alocação

Ocorrência precisa ser simultânea

Se ao menos uma não ocorrer, não haverá *DeadLock*

Aula 07

Condições para o Dead-Lock;

Introdução à Gerência de Memória

Condições para o *Deadlock*

Como usar recursos:

- Solicitar Recurso ao S.O.
- Se o Recurso estiver LIVRE
  - ALOCAR RECURSO
- Senão:
  - BLOQUEAR PROCESSO
- Após Uso:
  - LIBERA PROCESSO

Soluções

Ignorar o problema (“Algoritmo do Avestruz”);

- Linux, Unix, Windows

Detectar e Recuperar;

Evitar dinamicamente sua ocorrência, com alocação cuidadosa de recursos;

Impedir o *Deadlock* através da negação de pelo menos uma das 4 condições necessárias para sua ocorrência.

Detectar e Recuperar

- Detecção com um recurso de cada tipo
  - Montar Grafo
    - Se houver ciclo -> DEADLOCK
  - Analisar grau de multiprogramação
    - Taxa de bloqueio e tempo de espera
- Como recuperar?
  - Extermínio;
  - Volta ao Passado.
- Detecção X Recuperação

Detectar e Recuperar

- Extermínio
  - Mata um processo no ciclo;
  - Critério de escolha do processo:
    - Aleatório;
    - Prioridade;
    - Tempo de CPU;
    - Número de Recursos Alocados.

Detectar e Recuperar

Volta ao Passado

Rollback em processo do ciclo até liberar o recurso

P1 ---|R2|------|------|R1|------|-------|----

P2 ---|------|R1|------|------|R2|-------|----

P2 é alvo

Desfazer P2 até antes de alocar R1

Custo?

Quem implementa? Ex. SGBD

Como Evitar Deadlock

- Algoritmo do Banqueiro
  - Empréstimo X Reserva
  - Qual o $ que o banco empresta?
  - Estado Seguro
    - Probabilidade de Deadlock = 0%
- Vetores e Matrizes
  - Vetor de Recursos Existentes (VRE);
  - Vetor de Recursos Disponíveis (VRD);
  - Matriz de Recursos Alocados (MRA);
  - Matriz de Recursos Requisitados (MRR);
  - Matriz de Recursos Possivelmente Necessários. (MRPN).

Impedir Deadlock

- Exclusão Mútua
  - Impressora
    - Fila de Impressão - Finita
    - Problema: e se a fila encher?
  - Retardar o problema
- Posse e Espera
  - Requisitar todos os recursos possivelmente necessários num só passo e antecipadamente
    - Atomicidade: ou aloca todos ou nenhum
  - Problemas:
    - Pode alocar recursos que não serão utilizados;
    - Pode atrasar em função da demora para liberação de recursos que não serão utilizados de imediato;
    - S.O. precisa manter quantidade alta de recursos do mesmo tipo para manter concorrência.

Impedir Deadlock

- Posse e Espera
  - Priorizar recursos, numerando-os;
  - Requisitar recursos por ordem de prioridade;
  - *Wait-Die* (WD) não preemptivo
    - Quando P quer recurso alocado a Q, ele espera se for mais velho. Caso contrário, P morre;
  - *Wound-Wait* (WW) preemptivo
    - Quando P quer recurso alocado a Q, ele aguarda apenas se for mais novo. Caso contrário, Q morre;
  - Problemas:
    - Pode alocar recursos que não serão utilizados;
    - Pode atrasar em função da demora para liberação de recursos que não serão utilizados de imediato.

Gerência de Memória

Gerência de Memória

- Programas só executam se estiverem na memória principal;
- Funções do Gerenciador de Memória:
  - Controlar alocação de processos;
    - Novos processos;
    - Múltiplos processos;
    - Término de processo;
    - Crescimento e diminuição
      - Dados e Pilha

Gerência de Memória

- Modelos
  - Partições
    - Estáticas (Fixas)
    - Dinâmicas (Variáveis)
  - *Swapping*
  - Paginação
  - Segmentação

Alocação de Memória

- Alocação Contígua
  - Divisão da memória
    - SO residente
      - Código e dados (tabela de vetores, buffers, etc)
    - Processos de usuários
  - Proteção
    - Registrador de relocação: menor endereço
    - Registrador de limite: maior deslocamento
    - Cada processo tem seus valores (troca de contexto)

Alocação de Memória

- Alocação Contígua Simples
  - Partição Fixa
  - Implementada nos primeiros sistemas e ainda usada nos monoprogramáveis (monotarefa);
  - Memória é dividida em duas áreas:
    - Sistema Operacional e processo do usuário;
  - Usuário não pode usar uma área maior do que a disponível;
  - Sem proteção.

Alocação de Memória

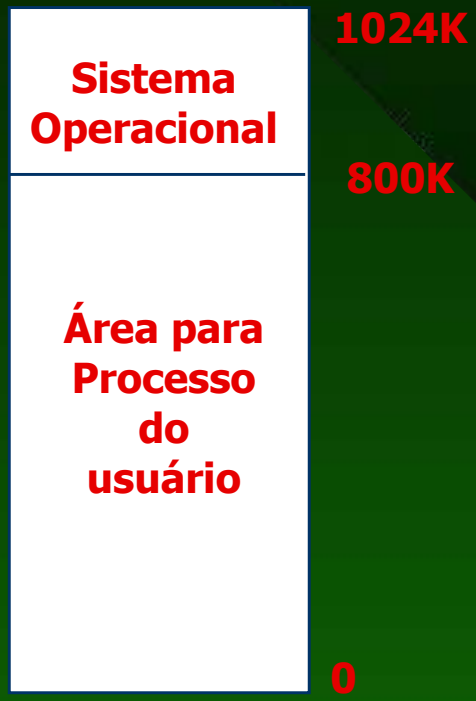
Alocação Contígua Simples

Memória Principal

Registrador de proteção delimita as áreas do sistema operacional e do usuário;
Sistema verifica acessos à memória em relação ao endereço do registrador.

Alocação de Memória

Alocação Contígua Simples



Nesse modelo, o SO foi carregado na **memória alta**.
Se o processo faz referência ao endereço real 2050, ele lhe pertence.
Registrador Base = 0K
Registrador Limite = 800K

Suponha que prog1.c executando referencie o endereço lógico 100. Qual o endereço físico?
Resp: 0K + 100

Nesse caso, o endereço lógico coincide com o físico.

Alocação de Memória

Alocação Contígua Simples

O maior endereço lógico referenciável por PROG1.C é 800K -1, pois além disso a aplicação invadiria o espaço reservado para o SO.

O espaço de endereços de PROG1.C é **[0, 800K)**

Alocação de Memória

Alocação Contígua Simples

Nesse modelo, o SO foi carregado na memória baixa.

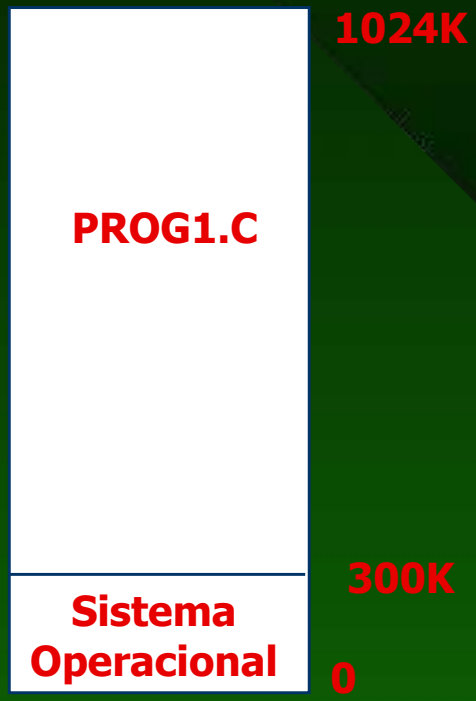
Registrador Base = 300K
Registrador Limite = 724K

Suponha que PROG1.C referencie o endereço lógico 100, por exemplo. Nesse caso, o endereço físico seria 300K + 100.

O endereço lógico **NÃO** coincide com o físico.

Alocação de Memória

Alocação Contígua Simples

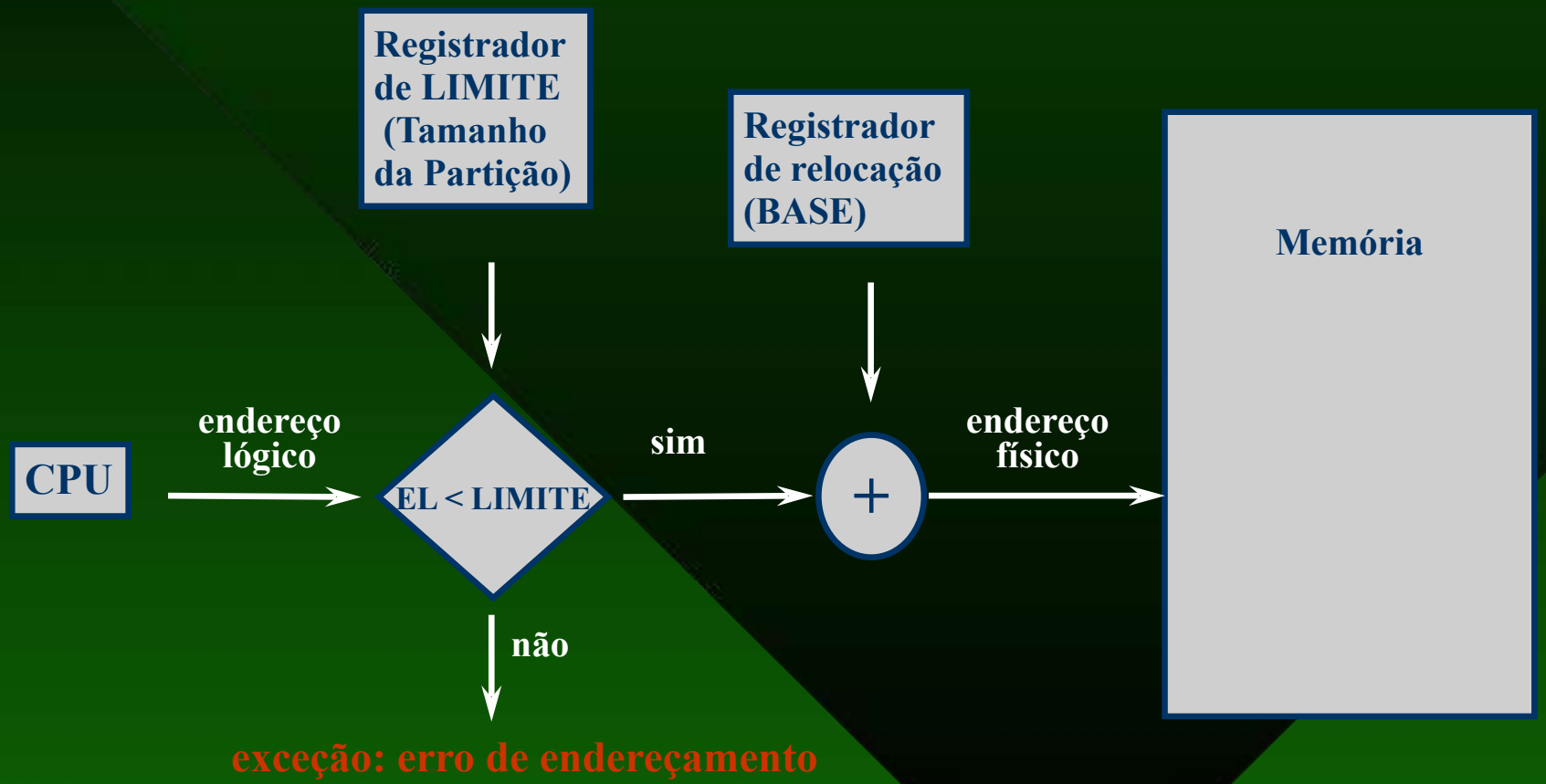


O maior endereço lógico referenciável por PROG1.C é [724K -1], pois além disso ele sairia da memória.

O espaço de endereços físicos de PROG1.C é **[300K, 1024K)**

O espaço de endereços físicos do SO é [0, 300K)

Alocação de Memória



Alocação Contígua Simples

Processos de usuário limitados pelo tamanho da memória principal disponível.

Solução:

- Dividir o programa em módulos ou partes;
- Permitir execução independente de cada módulo, usando a mesma área de memória;
- Técnica: *Overlay* (sobreposição);

Alocação Contígua Simples

Permite ao programador “expandir” os limites da memória principal;

Overlay – Área de memória comum onde módulos compartilham mesmo espaço;

Técnicas de Gerenciamento

Overlays

- Sobreposição de dados e instruções desnecessários
- Permite que um processo seja maior que o espaço alocado a ele
- Complexa responsabilidade passada ao programador
  - Eventual suporte de compiladores

Partições Fixas

Evolução dos sistemas operacionais demandou uso da memória por vários usuários simultaneamente;

Memória foi dividida em áreas de tamanho fixo: partições;

- Tamanho das partições era estabelecido no *boot*, em função do tamanho dos programas;
- Reparticionamento demandava novo *boot* com a nova configuração.

Tipos de Alocação Particionada:

- Alocação Particionada Estática:
  - Absoluta e Relocável;
- Alocação Particionada Dinâmica.

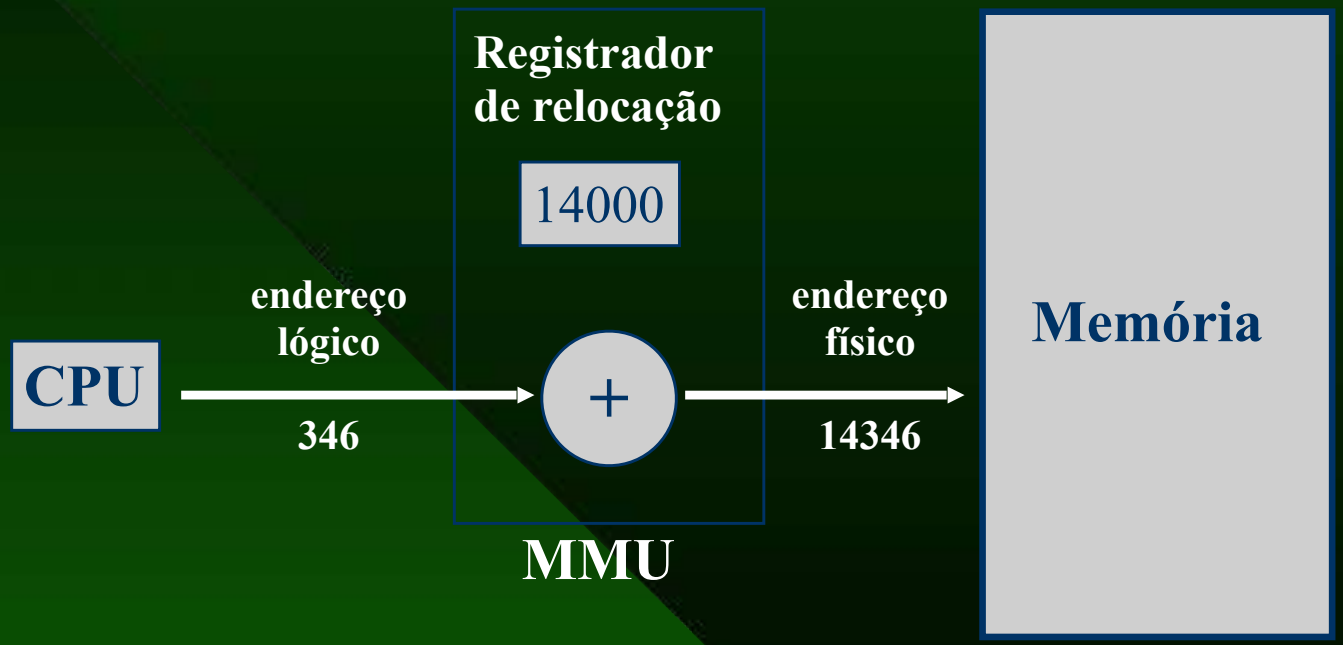
Espaço de Endereçamento

- Endereço Lógico ou Virtual
  - Gerado pela CPU
  - Espaço de endereçamento lógico
- Endereço Físico
  - Enviado à memória
    - Carregado no REM (Registrador de Endereço de Memória)
  - Espaço de endereçamento físico

Espaço de Endereçamento

- Mapeamento do Espaço Virtual para Físico
  - Só é necessário se a associação é feita em tempo de execução
  - Usuário trata endereços lógicos, nunca físicos
  - Suporte de hardware: MMU
    - *Memory Management Unit*
    - Exemplo: Registrador de relocação

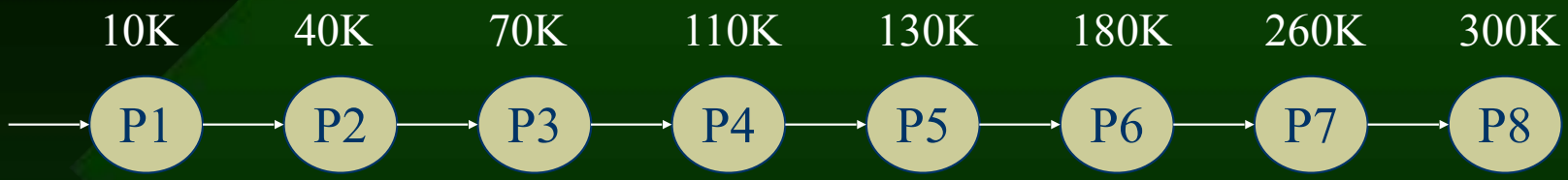
Espaço de Endereçamento



Alocação de Memória

- Alocação Particionada Estática
  - Partições Fixas
  - SOs multiprogramáveis – Ambiente *batch*
  - Partições
    - Porções de memória de tamanho fixo
    - Controla o grau de multiprogramação
  - Tamanho da partição estabelecido na fase de inicialização do sistema (*boot*)
  - SO mantém Fila de Entrada e Tabela de Partições

Alocação de Memória



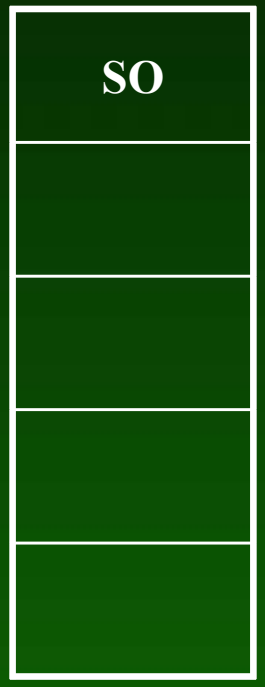


Tabela de Partições

Partição	Base	Limite	Processo ID	Tamanho Processo
0	0K	124K	SO	124K
1	124K	50K		
2	174K	150K		
3	324K	250K		
4	574K	450K		

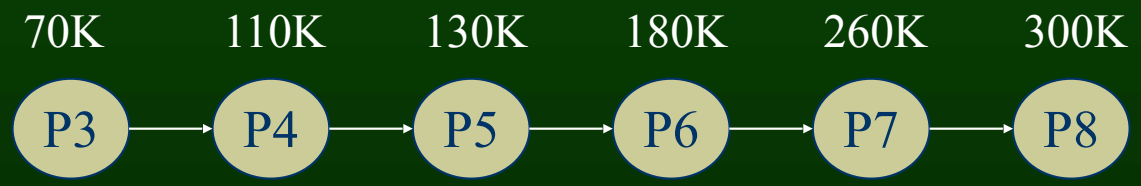
Alocação de Memória

SO
P1

Tabela de Partições

Partição	Base	Limite	Processo ID	Tamanho Processo
0	0K	124K	SO	124K
1	124K	50K	P1	10K
2	174K	150K		
3	324K	250K		
4	574K	450K		

Alocação de Memória

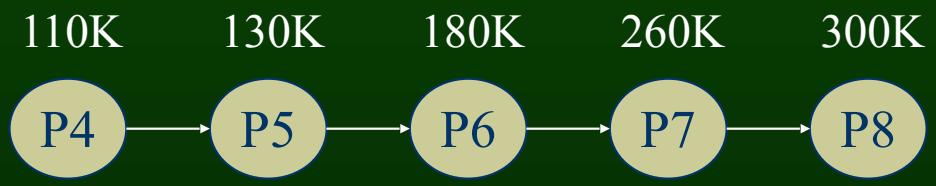


SO
P1
P2

Tabela de Partições

Partição	Base	Limite	Processo ID	Tamanho Processo
0	0K	124K	SO	124K
1	124K	50K	P1	10K
2	174K	150K	P2	40K
3	324K	250K		
4	574K	450K		

Alocação de Memória

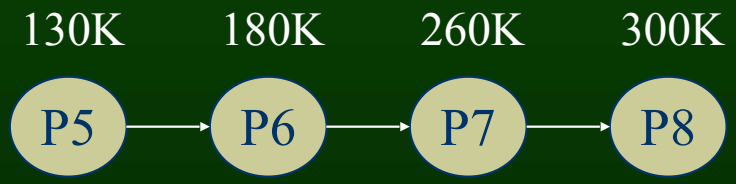


SO
P1
P2
P3

Tabela de Partições

Partição	Base	Limite	Processo ID	Tamanho Processo
0	0K	124K	SO	124K
1	124K	50K	P1	10K
2	174K	150K	P2	40K
3	324K	250K	P3	70K
4	574K	450K		

Alocação de Memória



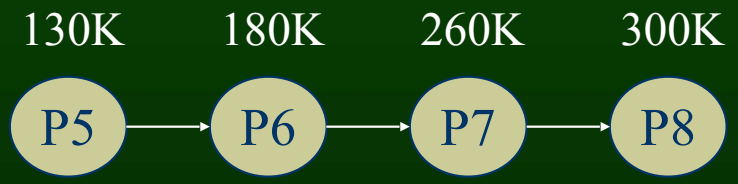
SO
P1
P2
P3
P4

Tabela de Partições

Partição	Base	Limite	Processo ID	Tamanho Processo
0	0K	124K	SO	124K
1	124K	50K	P1	10K
2	174K	150K	P2	40K
3	324K	250K	P3	70K
4	574K	450K	P4	110K

Alocação de Memória

A alocação ocorreu por ordem do Escalonador, sem avaliação da ocupação de memória.

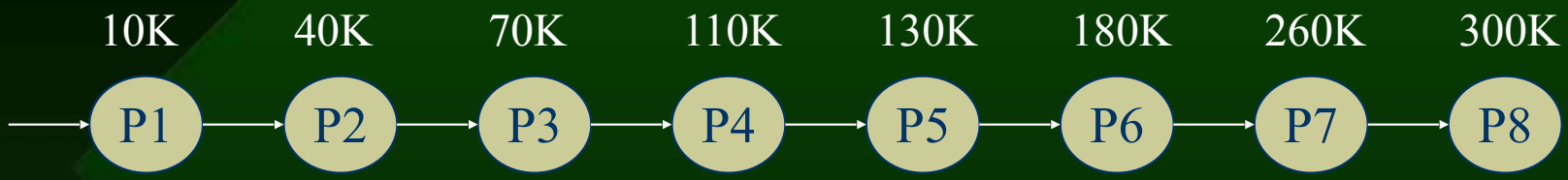


SO
P1
P2
P3
P4

Tabela de Partições

Partição	Base	Limite	Processo ID	Tamanho Processo
0	0K	124K	SO	124K
1	124K	50K	P1	10K
2	174K	150K	P2	40K
3	324K	250K	P3	70K
4	574K	450K	P4	110K

Alocação de Memória



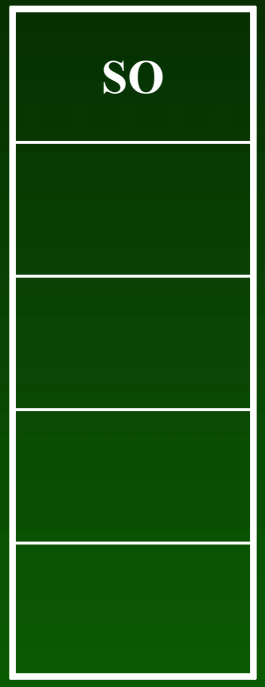
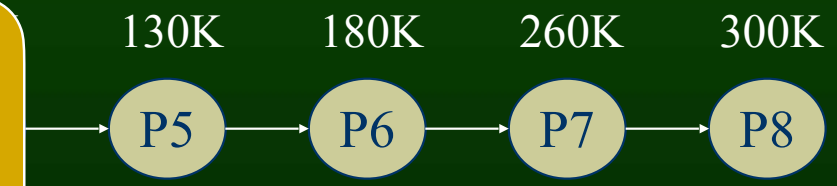


Tabela de Partições

Partição	Base	Limite	Processo ID	Tamanho Processo
0	0K	124K	SO	124K
1	124K	50K		
2	174K	150K		
3	324K	250K		
4	574K	450K		

Alocação de Memória

Como seria a alocação pelo Gerenciador de Memória?



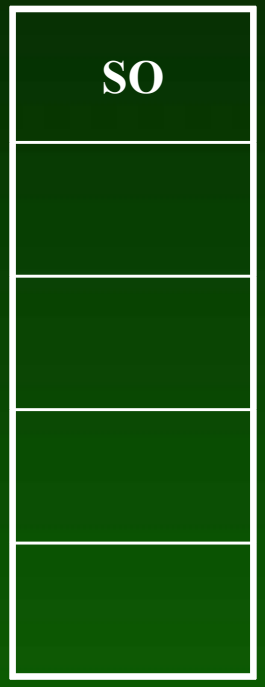
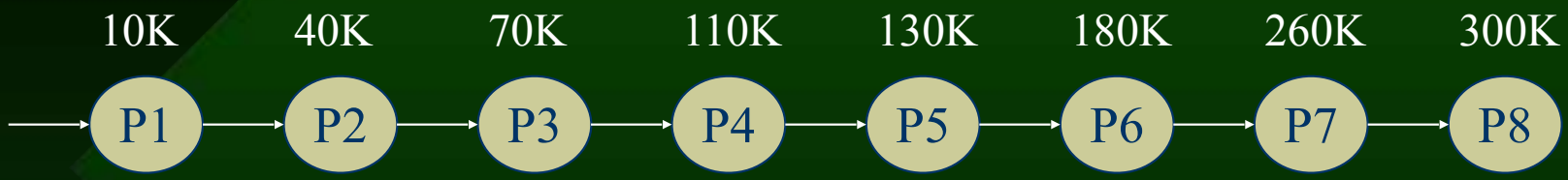


Tabela de Partições

Partição	Base	Limite	Processo ID	Tamanho Processo
0	0K	124K	SO	124K
1	124K	50K		
2	174K	150K		
3	324K	250K		
4	574K	450K		

Alocação de Memória



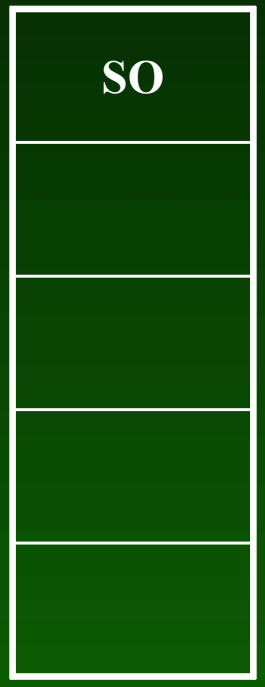
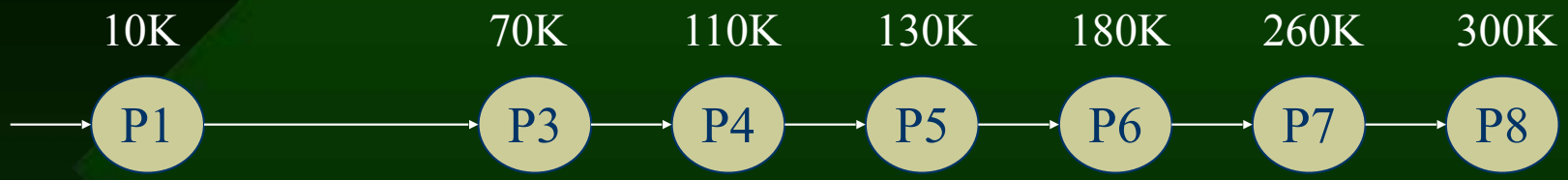


Tabela de Partições

Partição	Base	Limite	Processo ID	Tamanho Processo
0	0K	124K	SO	124K
1	124K	50K		
2	174K	150K		
3	324K	250K		
4	574K	450K		

Alocação de Memória

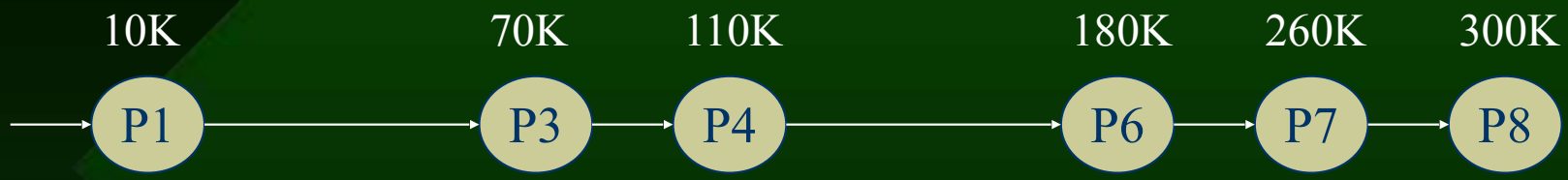


SO
P2

Tabela de Partições

Partição	Base	Limite	Processo ID	Tamanho Processo
0	0K	124K	SO	124K
1	124K	50K	P2	40K
2	174K	150K		
3	324K	250K		
4	574K	450K		

Alocação de Memória

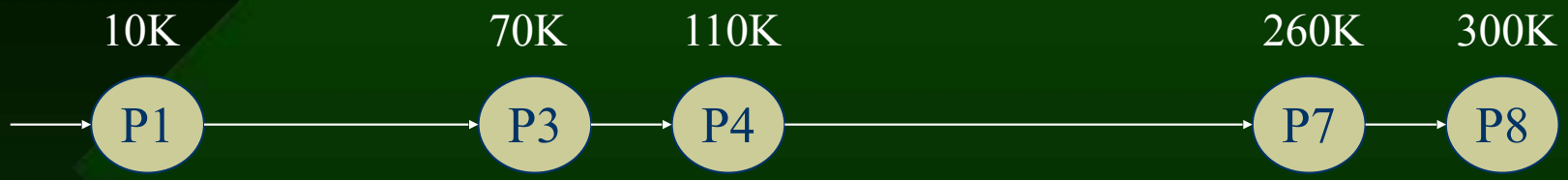


SO
P2
P5

Tabela de Partições

Partição	Base	Limite	Processo ID	Tamanho Processo
0	0K	124K	SO	124K
1	124K	50K	P2	40K
2	174K	150K	P5	130K
3	324K	250K		
4	574K	450K		

Alocação de Memória

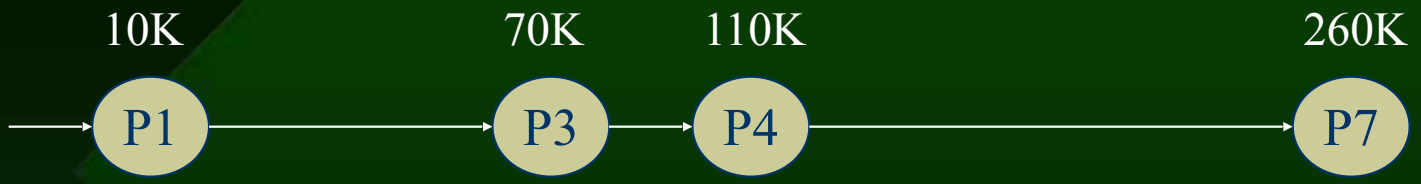


SO
P2
P5
P6

Tabela de Partições

Partição	Base	Limite	Processo ID	Tamanho Processo
0	0K	124K	SO	124K
1	124K	50K	P2	40K
2	174K	150K	P5	130K
3	324K	250K	P6	180K
4	574K	450K		

Alocação de Memória



SO
P2
P5
P6
P8

Tabela de Partições

Partição	Base	Limite	Processo ID	Tamanho Processo
0	0K	124K	SO	124K
1	124K	50K	P2	40K
2	174K	150K	P5	130K
3	324K	250K	P6	180K
4	574K	450K	P8	300K

Alocação de Memória

A alocação ocorreu por ordem do Gerenciador de Memória, sem avaliação da ocupação do Escalonador.

SO
P2
P5
P6
P8

Tabela de Partições

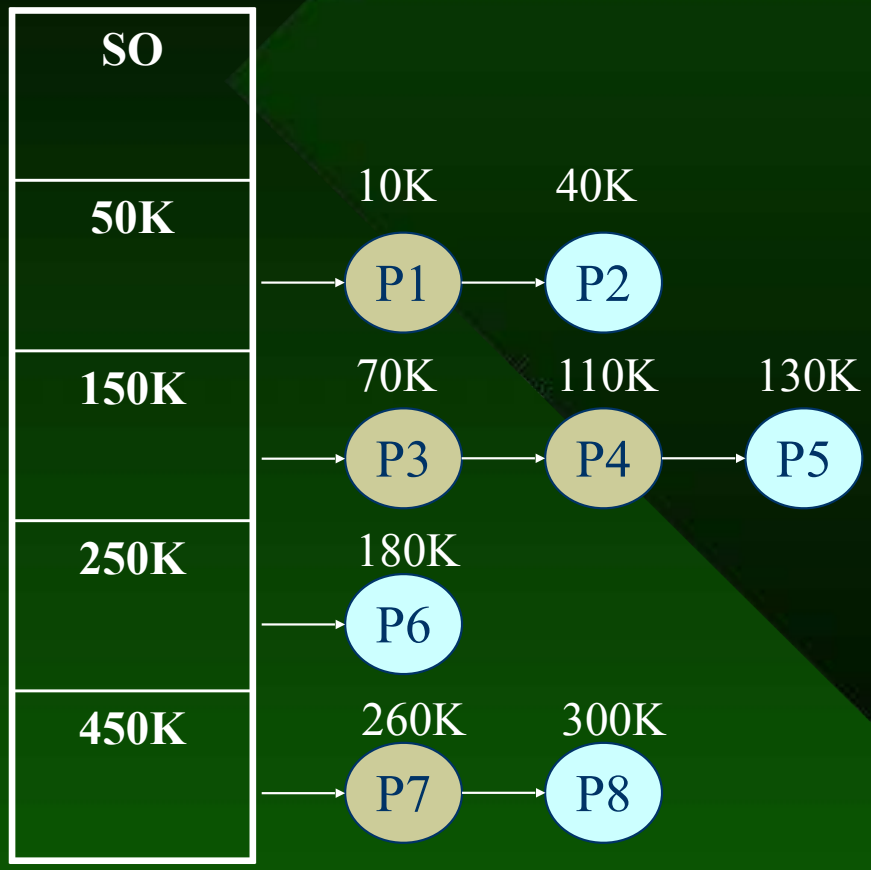
Partição	Base	Limite	Processo ID	Tamanho Processo
0	0K	124K	SO	124K
1	124K	50K	P2	40K
2	174K	150K	P5	130K
3	324K	250K	P6	180K
4	574K	450K	P8	300K

Alocação de Memória

Os problemas se repetem:

- Se o gerenciador de memória prioriza a melhor alocação da partição, o que acontecerá com processos muito pequenos, dado que na fila sempre existem maiores?
  - Inanição
- Quais seriam as estratégias para solução da Inanição?

Alocação de Memória



Múltiplas Filas

Quando P6 acabar, a partição ficará ociosa.
Pode ocorrer de uma partição ficar livre, enquanto outra tem muitos processos.

Alocação de Memória

- Fila única com *aging*
  - Determina-se um contador de saltos K_i (para cada processo *i*) e uma constante N (idade máxima);
  - Toda vez que uma partição ficar livre
    - Procura-se pelo processo que deixa o menor espaço desperdiçado;
    - Se na fila, antes dele, houver um processo que caiba na mesma partição, incrementamos K_i ;
    - Se K_i = N, P_i deve ser executado, independente do tamanho da partição.

Estratégias de Alocação

Escolha do bloco que deve ser alocado ao processo

Vários algoritmos possíveis

- *First-fit* (Performance?)
  - Aloca primeiro bloco suficientemente grande
  - Não precisa pesquisar todos blocos
- *Best-fit* (Ocupação de Memória?)
  - Aloca o menor bloco suficientemente grande
  - Precisa pesquisar todos blocos
    - Alternativa: ordenação dos blocos por tamanho
  - Gera o menor bloco de memória restante
- *Worst-fit* (Realocação?)
  - Aloca o maior bloco
  - Precisa pesquisar todos blocos
    - Alternativa: ordenação dos blocos por tamanho
  - Gera o maior bloco de memória restante

Estratégias de Alocação

Como avaliar?

Simulação !

- *First-fit* e *best-fit* utilizam melhor a memória
- *First-fit* é mais rápida

Estratégias de Alocação - *Best Fit*

Processos	Tamanho	Turnaround
Processo 1	100k	3
Processo 2	10k	1
Processo 3	35k	2
Processo 4	15k	1
Processo 5	23k	2
Processo 6	6k	1
Processo 7	25k	1
Processo 8	55k	2
Processo 9	88k	3
Processo 10	100k	3

Bloco Memória	Processo
Bloco 1 (50k)	
Bloco 2 (200K)	
Bloco 3 (70K)	
Bloco 4 (115K)	
Bloco 5 (15K)	

Tempo: -1

Estratégias de Alocação - *Best Fit*

Processos	Tamanho	Turnaround	
Processo 1	100k	3	Alocado
Processo 2	10k	1	
Processo 3	35k	2	
Processo 4	15k	1	
Processo 5	23k	2	
Processo 6	6k	1	
Processo 7	25k	1	
Processo 8	55k	2	
Processo 9	88k	3	
Processo 10	100k	3	

Bloco Memória	Processo
Bloco 1 (50k)	
Bloco 2 (200K)	
Bloco 3 (70K)	
Bloco 4 (115K)	Processo 1
Bloco 5 (15K)	

Tempo: 0

Estratégias de Alocação - *Best Fit*

Processos	Tamanho	Turnaround	
Processo 1	100k	3	Alocado
Processo 2	10k	1	Alocado
Processo 3	35k	2	
Processo 4	15k	1	
Processo 5	23k	2	
Processo 6	6k	1	
Processo 7	25k	1	
Processo 8	55k	2	
Processo 9	88k	3	
Processo 10	100k	3	

Bloco Memória	Processo
Bloco 1 (50k)	
Bloco 2 (200K)	
Bloco 3 (70K)	
Bloco 4 (115K)	Processo 1
Bloco 5 (15K)	Processo 2

Tempo: 0

Estratégias de Alocação - *Best Fit*

Processos	Tamanho	Turnaround	
Processo 1	100k	3	Alocado
Processo 2	10k	1	Alocado
Processo 3	35k	2	Alocado
Processo 4	15k	1	
Processo 5	23k	2	
Processo 6	6k	1	
Processo 7	25k	1	
Processo 8	55k	2	
Processo 9	88k	3	
Processo 10	100k	3	

Bloco Memória	Processo
Bloco 1 (50k)	Processo 3
Bloco 2 (200K)	
Bloco 3 (70K)	
Bloco 4 (115K)	Processo 1
Bloco 5 (15K)	Processo 2

Tempo: 0

Estratégias de Alocação - *Best Fit*

Processos	Tamanho	Turnaround	
Processo 1	100k	3	Alocado
Processo 2	10k	1	Alocado
Processo 3	35k	2	Alocado
Processo 4	15k	1	Alocado – **15K em 70K !**
Processo 5	23k	2	
Processo 6	6k	1	
Processo 7	25k	1	
Processo 8	55k	2	
Processo 9	88k	3	
Processo 10	100k	3	

Bloco Memória	Processo
Bloco 1 (50k)	Processo 3
Bloco 2 (200K)	
Bloco 3 (70K)	Processo 4
Bloco 4 (115K)	Processo 1
Bloco 5 (15K)	Processo 2

Tempo: 0

Estratégias de Alocação - *Best Fit*

Processos	Tamanho	Turnaround	
Processo 1	100k	3	Alocado
Processo 2	10k	1	Alocado
Processo 3	35k	2	Alocado
Processo 4	15k	1	Alocado – **15K em 70K !**
Processo 5	23k	2	Alocado – **23K em 200K !**
Processo 6	6k	1	
Processo 7	25k	1	
Processo 8	55k	2	
Processo 9	88k	3	
Processo 10	100k	3	

Bloco Memória	Processo
Bloco 1 (50k)	Processo 3
Bloco 2 (200K)	Processo 5
Bloco 3 (70K)	Processo 4
Bloco 4 (115K)	Processo 1
Bloco 5 (15K)	Processo 2

Tempo: 0

Estratégias de Alocação - *Best Fit*

Processos	Tamanho	Turnaround	
Processo 1	100k	3	Alocado
Processo 2	10k	1	Concluído !
Processo 3	35k	2	Alocado
Processo 4	15k	1	Concluído !
Processo 5	23k	2	Alocado
Processo 6	6k	1	
Processo 7	25k	1	
Processo 8	55k	2	
Processo 9	88k	3	
Processo 10	100k	3	

Bloco Memória	Processo
Bloco 1 (50k)	Processo 3
Bloco 2 (200K)	Processo 5
Bloco 3 (70K)	
Bloco 4 (115K)	Processo 1
Bloco 5 (15K)	

Tempo: 1

Estratégias de Alocação - *Best Fit*

Processos	Tamanho	Turnaround	
Processo 1	100k	3	Alocado
Processo 2	10k	1	Concluído !
Processo 3	35k	2	Alocado
Processo 4	15k	1	Concluído !
Processo 5	23k	2	Alocado
Processo 6	6k	1	Alocado
Processo 7	25k	1	
Processo 8	55k	2	
Processo 9	88k	3	
Processo 10	100k	3	

Bloco Memória	Processo
Bloco 1 (50k)	Processo 3
Bloco 2 (200K)	Processo 5
Bloco 3 (70K)	
Bloco 4 (115K)	Processo 1
Bloco 5 (15K)	Processo 6

Tempo: 1

Estratégias de Alocação - *Best Fit*

Processos	Tamanho	Turnaround	
Processo 1	100k	3	Alocado
Processo 2	10k	1	Concluído !
Processo 3	35k	2	Alocado
Processo 4	15k	1	Concluído !
Processo 5	23k	2	Alocado
Processo 6	6k	1	Alocado
Processo 7	25k	1	Alocado
Processo 8	55k	2	
Processo 9	88k	3	
Processo 10	100k	3	

Bloco Memória	Processo
Bloco 1 (50k)	Processo 3
Bloco 2 (200K)	Processo 5
Bloco 3 (70K)	Processo 7
Bloco 4 (115K)	Processo 1
Bloco 5 (15K)	Processo 6

Tempo: 1

Estratégias de Alocação - *Best Fit*

Processos	Tamanho	Turnaround	
Processo 1	100k	3	Alocado
Processo 2	10k	1	Concluído !
Processo 3	35k	2	Concluído !
Processo 4	15k	1	Concluído !
Processo 5	23k	2	Concluído !
Processo 6	6k	1	Concluído !
Processo 7	25k	1	Concluído !
Processo 8	55k	2	
Processo 9	88k	3	
Processo 10	100k	3	

Bloco Memória	Processo
Bloco 1 (50k)	
Bloco 2 (200K)	
Bloco 3 (70K)	
Bloco 4 (115K)	Processo 1
Bloco 5 (15K)	

Tempo: 2

Estratégias de Alocação - *Best Fit*

Processos	Tamanho	Turnaround	
Processo 1	100k	3	Alocado
Processo 2	10k	1	Concluído !
Processo 3	35k	2	Concluído !
Processo 4	15k	1	Concluído !
Processo 5	23k	2	Concluído !
Processo 6	6k	1	Concluído !
Processo 7	25k	1	Concluído !
Processo 8	55k	2	Alocado
Processo 9	88k	3	
Processo 10	100k	3	

Bloco Memória	Processo
Bloco 1 (50k)	
Bloco 2 (200K)	
Bloco 3 (70K)	Processo 8
Bloco 4 (115K)	Processo 1
Bloco 5 (15K)	

Tempo: 2

Estratégias de Alocação - *Best Fit*

Processos	Tamanho	Turnaround	
Processo 1	100k	3	Alocado
Processo 2	10k	1	Concluído !
Processo 3	35k	2	Concluído !
Processo 4	15k	1	Concluído !
Processo 5	23k	2	Concluído !
Processo 6	6k	1	Concluído !
Processo 7	25k	1	Concluído !
Processo 8	55k	2	Alocado
Processo 9	88k	3	Alocado
Processo 10	100k	3	

Bloco Memória	Processo
Bloco 1 (50k)	
Bloco 2 (200K)	Processo 9
Bloco 3 (70K)	Processo 8
Bloco 4 (115K)	Processo 1
Bloco 5 (15K)	

Tempo: 2

Estratégias de Alocação - *Best Fit*

Processos	Tamanho	Turnaround	
Processo 1	100k	3	Concluído !
Processo 2	10k	1	Concluído !
Processo 3	35k	2	Concluído !
Processo 4	15k	1	Concluído !
Processo 5	23k	2	Concluído !
Processo 6	6k	1	Concluído !
Processo 7	25k	1	Concluído !
Processo 8	55k	2	Alocado
Processo 9	88k	3	Alocado
Processo 10	100k	3	

Bloco Memória	Processo
Bloco 1 (50k)	
Bloco 2 (200K)	Processo 9
Bloco 3 (70K)	Processo 8
Bloco 4 (115K)	
Bloco 5 (15K)	

Tempo: 3

Estratégias de Alocação - *Best Fit*

Processos	Tamanho	Turnaround	
Processo 1	100k	3	Concluído !
Processo 2	10k	1	Concluído !
Processo 3	35k	2	Concluído !
Processo 4	15k	1	Concluído !
Processo 5	23k	2	Concluído !
Processo 6	6k	1	Concluído !
Processo 7	25k	1	Concluído !
Processo 8	55k	2	Alocado
Processo 9	88k	3	Alocado
Processo 10	100k	3	Alocado

Bloco Memória	Processo
Bloco 1 (50k)	
Bloco 2 (200K)	Processo 9
Bloco 3 (70K)	Processo 8
Bloco 4 (115K)	Processo 10
Bloco 5 (15K)	

Tempo: 3

Estratégias de Alocação - *Best Fit*

Processos	Tamanho	Turnaround	
Processo 1	100k	3	Concluído !
Processo 2	10k	1	Concluído !
Processo 3	35k	2	Concluído !
Processo 4	15k	1	Concluído !
Processo 5	23k	2	Concluído !
Processo 6	6k	1	Concluído !
Processo 7	25k	1	Concluído !
Processo 8	55k	2	Concluído !
Processo 9	88k	3	Alocado
Processo 10	100k	3	Alocado

Bloco Memória	Processo
Bloco 1 (50k)	
Bloco 2 (200K)	Processo 9
Bloco 3 (70K)	
Bloco 4 (115K)	Processo 10
Bloco 5 (15K)	

Tempo: 4

Estratégias de Alocação - *Best Fit*

Processos	Tamanho	Turnaround	
Processo 1	100k	3	Concluído !
Processo 2	10k	1	Concluído !
Processo 3	35k	2	Concluído !
Processo 4	15k	1	Concluído !
Processo 5	23k	2	Concluído !
Processo 6	6k	1	Concluído !
Processo 7	25k	1	Concluído !
Processo 8	55k	2	Concluído !
Processo 9	88k	3	Concluído !
Processo 10	100k	3	Alocado

Bloco Memória	Processo
Bloco 1 (50k)	
Bloco 2 (200K)	
Bloco 3 (70K)	
Bloco 4 (115K)	Processo 10
Bloco 5 (15K)	

Tempo: 5

Estratégias de Alocação - *Best Fit*

Processos	Tamanho	Turnaround	
Processo 1	100k	3	Concluído !
Processo 2	10k	1	Concluído !
Processo 3	35k	2	Concluído !
Processo 4	15k	1	Concluído !
Processo 5	23k	2	Concluído !
Processo 6	6k	1	Concluído !
Processo 7	25k	1	Concluído !
Processo 8	55k	2	Concluído !
Processo 9	88k	3	Concluído !
Processo 10	100k	3	Concluído !

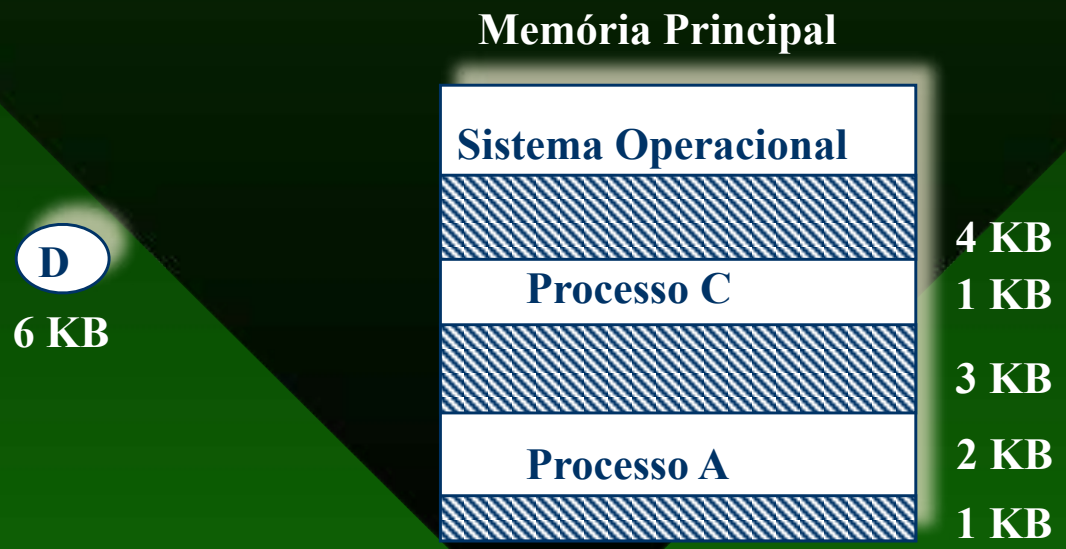
Bloco Memória	Processo
Bloco 1 (50k)	
Bloco 2 (200K)	
Bloco 3 (70K)	
Bloco 4 (115K)	
Bloco 5 (15K)	

Tempo: 6

Alocação Contígua

Fragmentação Externa

- Ocorre quando os processos terminam, deixando espaços de memória livre fragmentados;
- Não há espaço livre contíguo suficiente



Alocação Contígua

- Fragmentação Externa - soluções:
  - Aguardar término de processos
  - Desfragmentação
    - Reloca partições ocupadas para criar área livre única
      - CONTÍGUA
    - Exige relocação dinâmica – em tempo de execução
      - MGM traduzindo endereços
    - *Overhead*
  - Permitir alocação não contígua
    - Paginação e segmentação

Aula 09

Swapping

Programas ficavam na memória principal, mesmo se bloqueados, a espera de um evento (I/O);

Solução encontrada: *Swapping*;

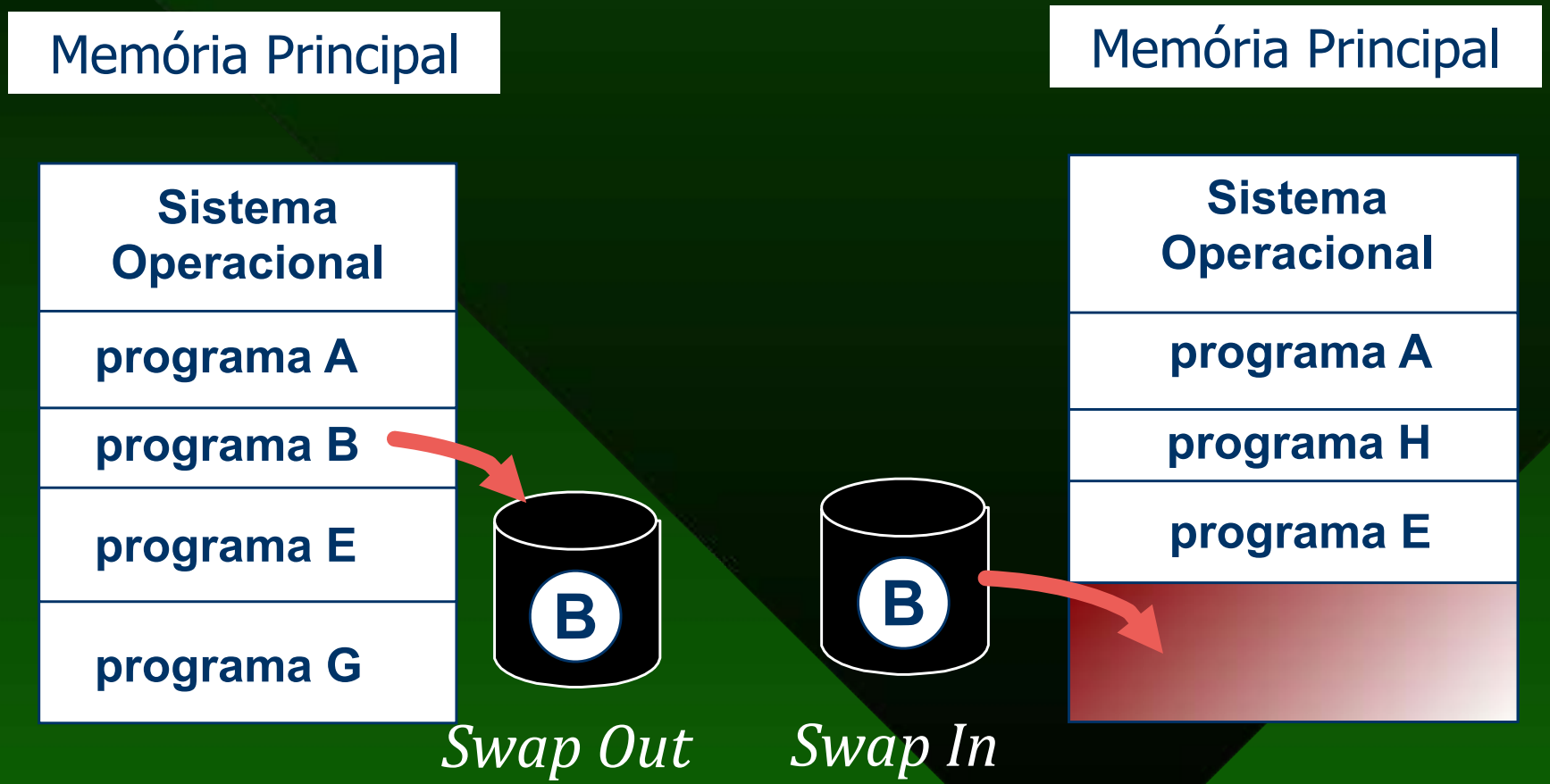
- Processos não ficam mais na memória o tempo todo;
- Resolve problema de processos que aguardam por espaço livre adequado.

Swapping

Processo residente na memória é escolhido e levado para o disco (*Swapped-Out*), dando lugar a outro;

Processo *Swapped-Out* retorna posteriormente à memória (*Swapped-In*), sem "*perceber*" o que ocorreu.

Swapping


Memória Principal
Sistema Operacional
programa A
programa B
programa E
programa G
B
Swap Out
B
Swap In
Memória Principal
Sistema Operacional
programa A
programa H
programa E

Swapping

Problema gerado pelo Swapping:

- Relocação dos programas
  - Se processos “saem e voltam” muitas vezes, tempo gasto com relocação é alto (*overhead*);
- Solução
  - Mecanismo de Relocação Dinâmica:
- Quando ocorrer referência a algum endereço, o endereço da instrução será somado ao valor do registrador:

Endereço Real = Endereço Base da Partição + Deslocamento.

Deslocamento = Endereço Referenciado no Processo

Swapping

Vantagens:

- Permite maior compartilhamento da memória;
- Aumento no *throughput*

Eficiente para sistemas com poucos usuários e pequenas aplicações;

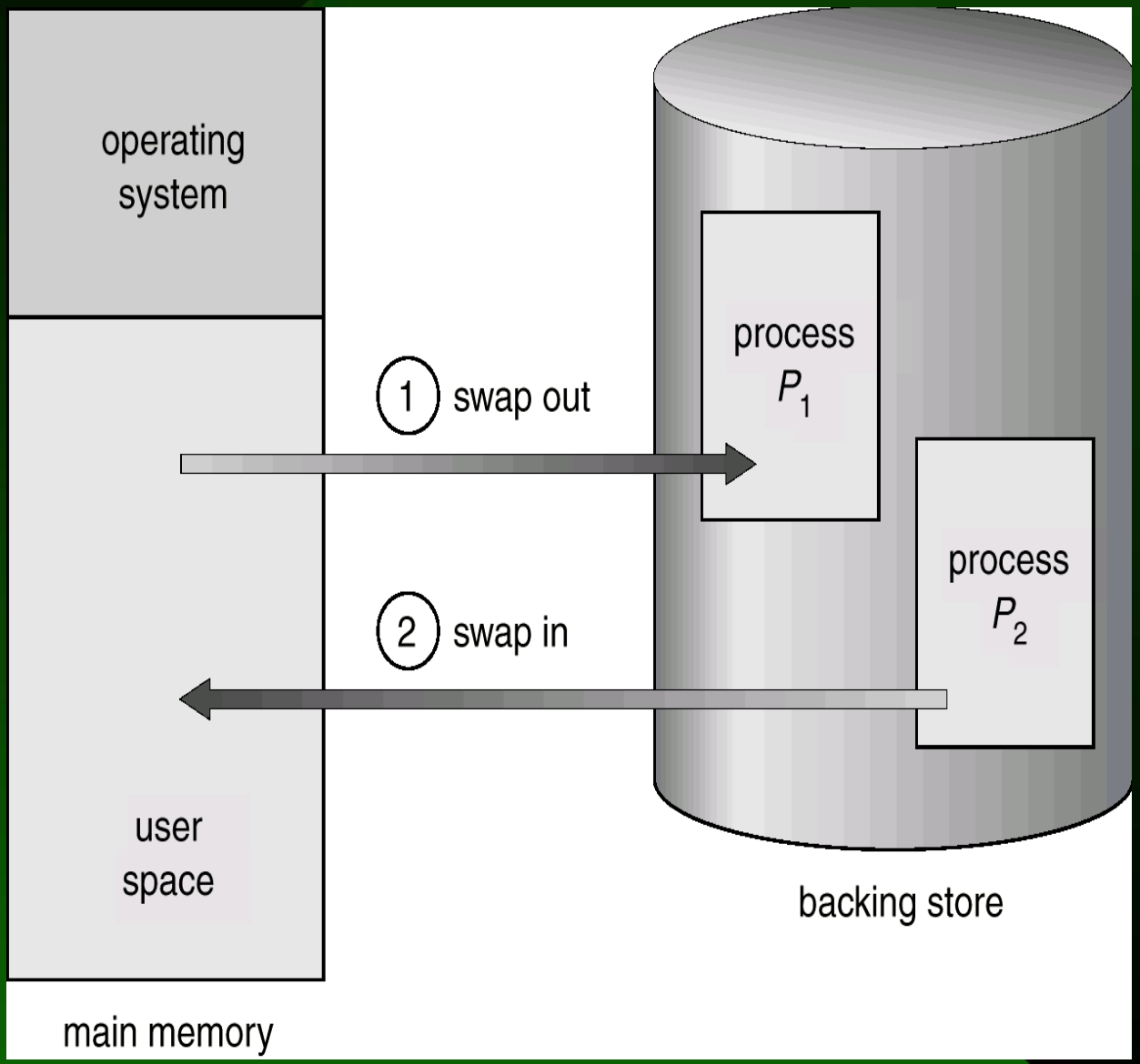
Problema:

- custo do *Swapping* (*in*/*out*).

Swapping

- *Swap-out*
  - Processos são removidos temporariamente
- *Swap-in*
  - Processos precisam voltar para o mesmo espaço
    - Resolução de endereço em tempo de compilação ou carga
  - Processos não precisam voltar para o mesmo espaço
    - Resolução de endereço em tempo de execução
    - Endereços físicos recalculados na execução

Swapping


operating system
user space
main memory
1 swap out
2 swap in
process P_1
process P_2
backing store

Swapping

- Dispositivo auxiliar em disco
  - Deve ser rápido
  - Proporcional à quantidade de dados movidos
  - Escalonador de Alto Nível verifica processo selecionado
    - Pode exigir *swap-in* (e *swap-out* para criar espaço)

Swapping

- *Swap-out* de processos com I/O pendente
  - Acesso à área do processo
    - Pode gerar inconsistências
  - Soluções
    - Não retirar processo com I/O pendente
    - Sempre usar buffers do SO

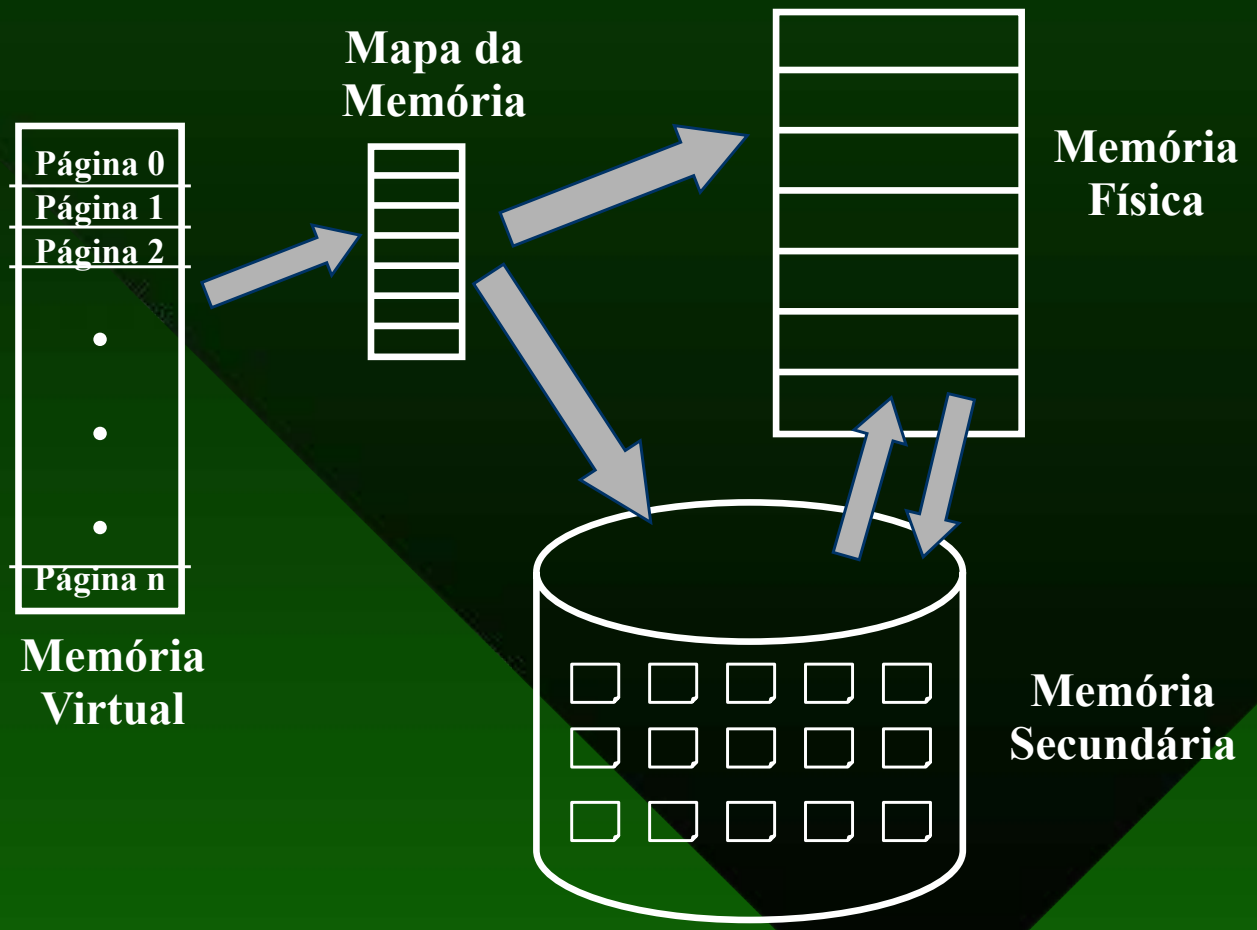
Memória Virtual

- Técnica de Gerenciamento de Memória
- Permite execução de processos que não estejam totalmente na memória
  - A parte sendo executada precisa estar em memória
- Programas não precisam ser carregados totalmente na memória
  - Rotinas de exceção
  - Rotinas necessárias apenas na inicialização
  - Matrizes e tabelas super-dimensionadas

Memória Virtual

- Processos podem ser maiores que a memória física
- Economia permite execução de mais processos
- Implementação complexa
- Transparência para usuário e programador
  - Diferente da técnica de ***overlay***
- *Overhead* de traduções e *swap*
- Implementações
  - Paginação sob demanda
    - Referência a página fora da memória
  - Segmentação sob demanda

Memória Virtual



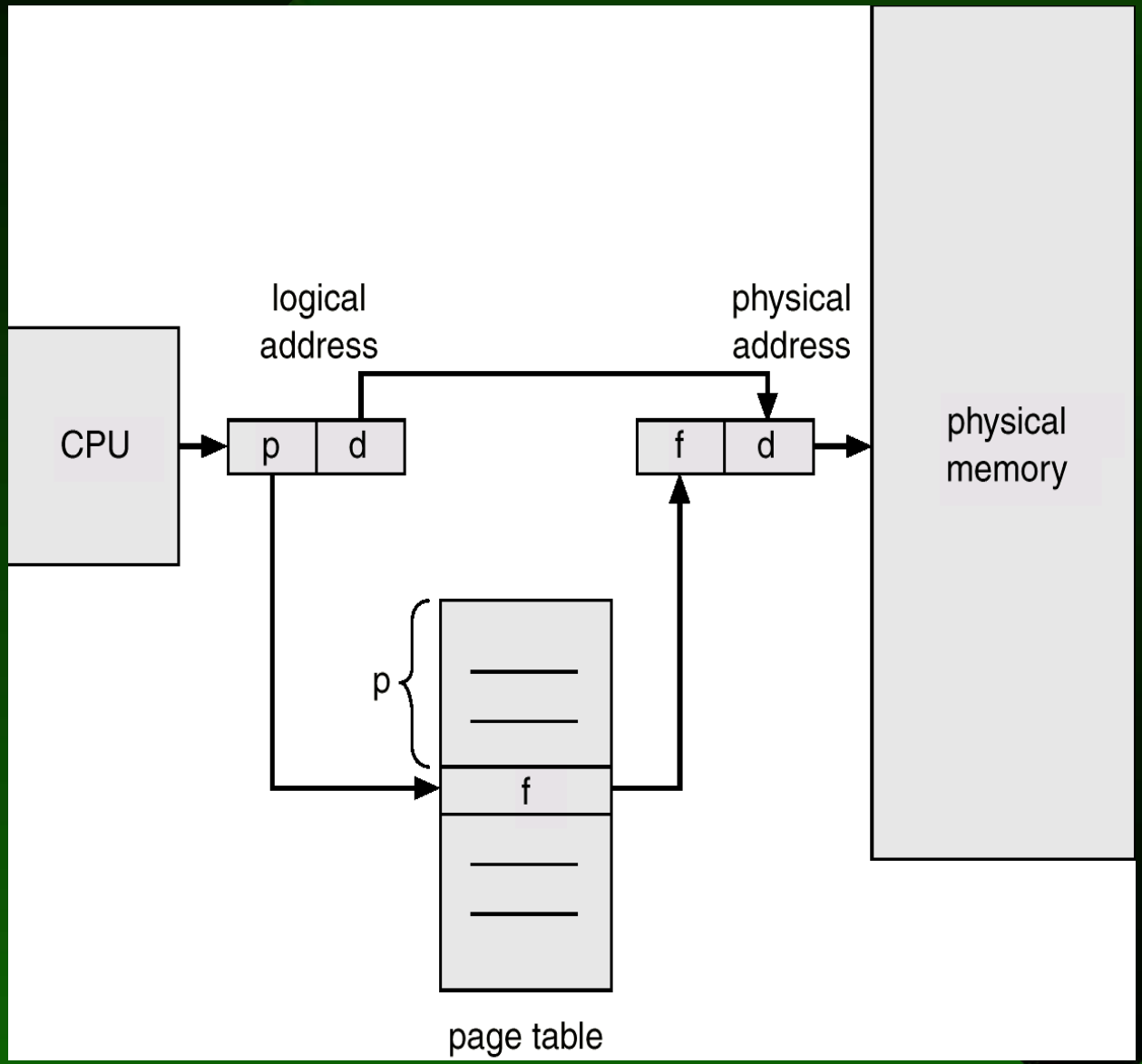
Paginação

- Permite espaço de endereçamento não contíguo
- Memória física dividida em quadros (*frames*)
- Memória lógica dividida em páginas (*pages*)
- Quadros e páginas têm o mesmo tamanho
  - Normalmente entre 512 bytes e 16MB
- Páginas dos processos podem ocupar qualquer quadro disponível

Paginação

- Endereço lógico
  - Número da página
  - Deslocamento dentro da página (offset)
- Tabela de página do processo
  - Número da página serve como índice da tabela
  - Associa páginas a quadros

Paginação


logical
address
physical
address
CPU
p
d
f
d
physical
memory
p
f
page table

Paginação

- Fragmentação
  - Apenas interna
    - Apenas no último quadro
    - Pior caso: desperdício de quase 1 quadro

Paginação

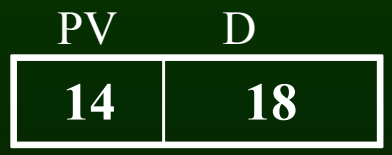
Tamanho da Página (pequeno)

- Diminui perda com fragmentação
- Tabela de Páginas Grande
  - Custo (*overhead*) para tradução
  - Consumo de memória com tabela
- Exemplo:
  - Sistema com RE = 32 bits
  - Páginas com 256 bytes = 2^8
  - Número de linhas da TAP? 2^{24} linhas

Paginação

Tamanho da Página (grande)

- Diminui overhead de controle
- I/O mais eficiente – menos SWAP
- Maior desperdício interno
- Exemplo:
  - Sistema com 32 bits
  - Páginas com 256 Kbytes = 2^{18}
  - Número de linhas da TAP? 2^{14} linhas
- Comparação
  - TAP 1024 X menor; Página 1024 X maior
    - Menos SWAP
    - Maior desperdício dentro da página
      - Alocar um processo de 10 K necessita de, no mínimo, 256 K



Aula 10

Sistema de Arquivos

Sistema de Arquivos

Uma das principais partes de um SO

- Sigla DOS (*Disk Operational System*)

Gerencia o armazenamento e acesso às informações nos dispositivos de memória secundária (aplicações e dados);

Define uma interface para a manipulação dos arquivos nos diversos dispositivos de memória secundária, mantendo homogeneidade e transparência.

Sistema de Arquivos (partes)

- Arquivos propriamente ditos
  - Informações logicamente relacionadas (*bits*, *bytes*, linhas e registros), que representam aplicações ou dados em diversos formatos;
  - Informações e Dados para uso do próprio SO (arquivos de sistema);
- Estrutura de Pastas ou Diretórios
  - Organização hierárquica dos arquivos
  - Atributos e Metadados

Organização e hierarquia

Unidades de memória secundária (discos) podem ser divididas em partições ou volumes;

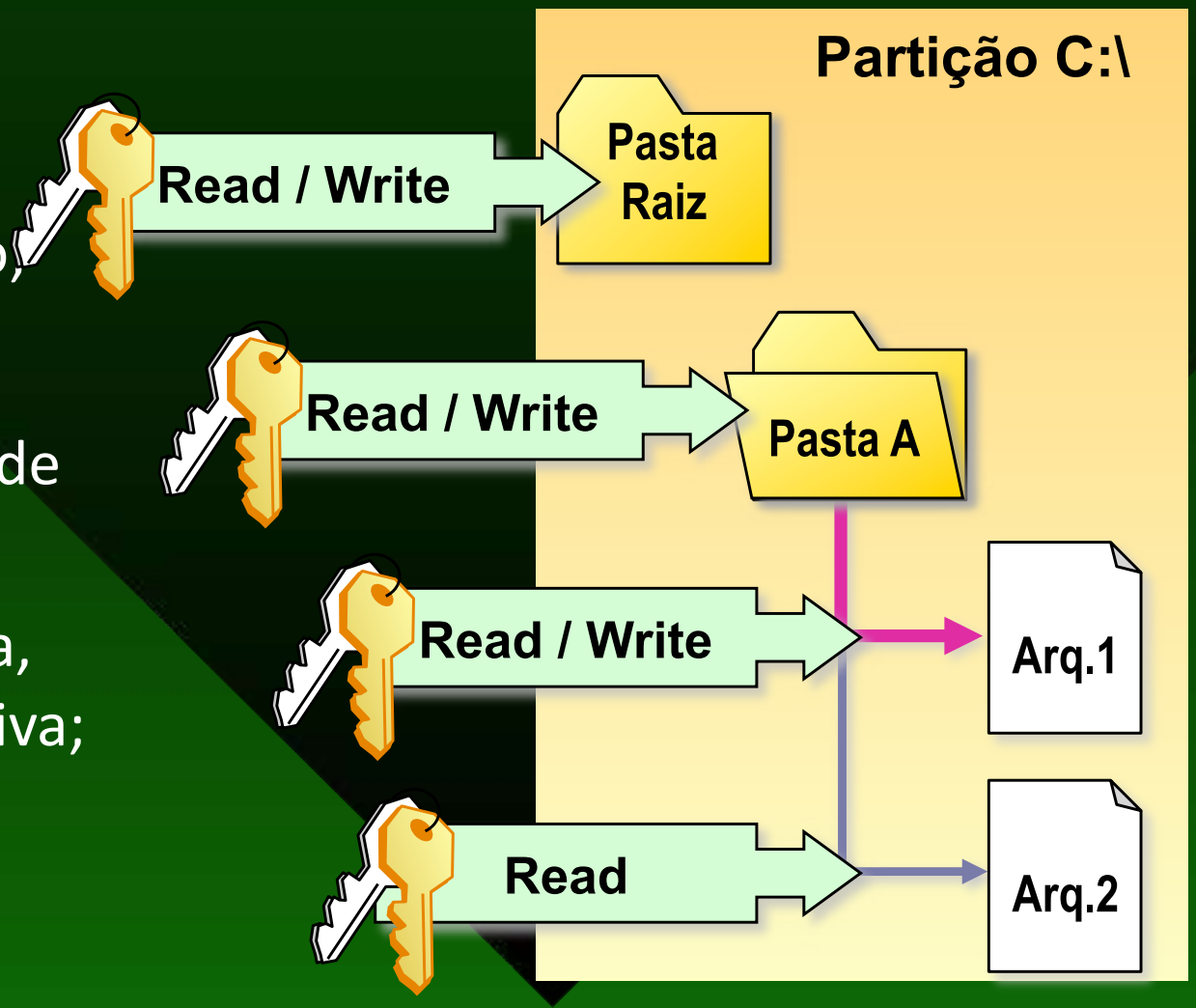
Cada disco contém pelo menos uma partição onde estão localizados os arquivos e as pastas/diretórios;

Pasta / Diretório:

- Estrutura de dados que mantém informações sobre a coleção dos arquivos contidos na memória secundária (um ou mais dispositivos);
- Também possui atributos;
- Pode ser vista como um conjunto de tabelas que associa nomes a arquivos.

Organização e hierarquia

- Localização
  - Identificação do dispositivo de armazenamento;
  - Ponteiro para o local dentro do dispositivo;
- Hierarquia
  - Estrutura em árvore de pastas / diretórios;
  - Interface homogênea, transparente e intuitiva;



Atributos

Os atributos que um arquivo pode receber no FAT32 são:

- Somente Leitura
- Oculto
- Sistema
- ID do Volume
- Diretório
- Arquivo

Atributos e Metadados

Básicos

- Nome (segue regras específicas, e pode estar dividido em nome e extensão);
- Data e Hora (criação, última modificação, acesso etc);
- Tamanho;

Avançados

- Usuários (proprietário, criador etc);
- Segurança: ACL (*Access Control List*), atributos de segurança (*Read-Only*, *Execute*, *Hidden* etc), filtros de direitos etc;
- Informações diversas (classificação, migração etc).

Operações sobre arquivos

Rotinas de Sistema permitem às aplicações realizar operações de E/S como tradução de nomes em endereços, leitura e gravação de dados, criação e eliminação de arquivos;

Operações sobre arquivos mais comuns são:

- Criar;
- Abrir;
- Ler;
- Gravar;
- Fechar;
- Renomear;
- Mover;
- Apagar / Destruir.

Operações sobre pastas

Uma pasta também é um arquivo, com vários atributos, mas tratada de forma diferenciada pelo SO (tipicamente um bit define se é arquivo ou pasta;

Operações sobre pastas mais comuns são:

- Procurar, apagar, renomear, mover e copiar;
- Mostrar uma lista com o conteúdo da pasta;

Estruturas mais comuns são:

- Nível Único (*Single-Level Folder*);
- Dois Níveis (*Two-Level Folder*);
- Árvore;
- Grafo Acíclico;
- dGrafo.

Tab. Alocação de Arquivos (FAT)

Dividida em quatro regiões:

- Reservada (ou setor de boot)
- Tabela de Arquivos (FAT)
- Arquivos e diretórios

Desfragmentação

A FAT não possui nenhum mecanismo que impeça, ou pelo menos diminua a fragmentação, daí a necessidade deste trabalho periódico;

A função é mover os arquivos, de forma que eles fiquem gravados em *clusters* sequenciais;

O processo é demorado, e além disto, precisa ser repetido regularmente;

A manipulação pode ocorrer apenas nos apontadores da lista encadeada para unidades SSD, ou similares.

Compartilhamento de Arquivos

- Arquivos e pastas podem ter permissões atribuídas individualmente.
  - Podem ser atribuídas permissões a um grupo de usuários ou a um único usuário.
  - Pode definir o nível de acesso.
  - Permissões para arquivos tem prioridade sobre permissões para pastas.
  - Negar permissão tem prioridade sobre Permitir.

ACL

ACL (*Access Control Lists*)

- Esta lista é utilizada como um método adicional para prover permissões para usuários a determinados arquivos e pastas;

Cada entrada de uma lista ACL contém:

- Qual(is) usuários(s) tem acesso aquela entrada;
- Se a entrada é responsável por negar ou permitir o acesso;
- Quais direitos serão permitidos ou negados:

(R)leitura, (W)escrita, (X)execução, (D)apagar,
(A)modificação de atributos, (P)modificação de permissões.

ACL - Herança

O ACL permite 4 tipos de herança para arquivos e sub-pastas de uma determinada pasta:

- Herança para todos os arquivos;
- Herança para todas as pastas;
- Herança limitada apenas aos arquivos da pasta;
- Herança somente para sub-pastas e arquivos dentro da pasta mas não a ela.

O sistema tem que possuir um registro de quais Entradas da ACL são herdadas e quais não são.

Métodos de Acesso

Métodos de Acesso

Forma de acesso da informação no arquivo;

Existem diversas formas, e estas determinam a organização dos arquivos, performance e até segurança;

Tipos de acesso:

- Acesso sequencial;
- Acesso direto;
- Acesso indexado.

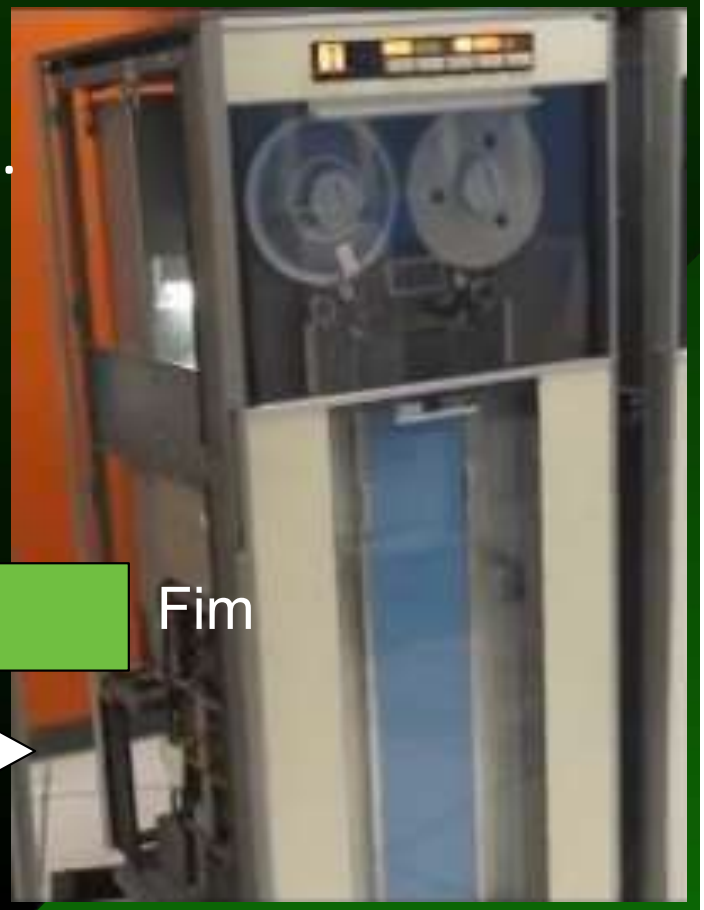
É possível suportar mais de um método de acesso.

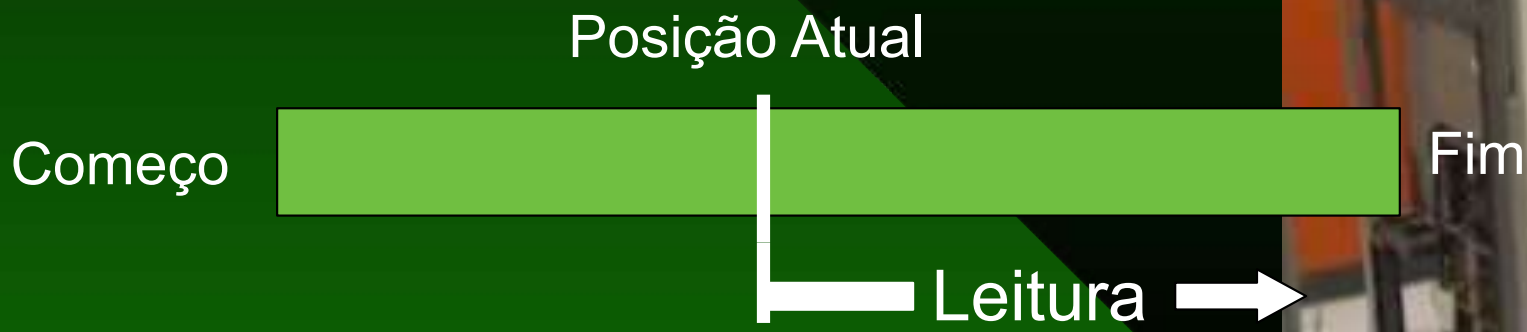
Acesso Sequencial

Primeiro método de acesso (usado nas fitas);

Informações no arquivo são processadas na ordem que foram gravadas;

Gravação só é possível no final do arquivo.





Sequencial / Alocação Contígua

Informações do Arquivo:

- Início e Tamanho

Problemas

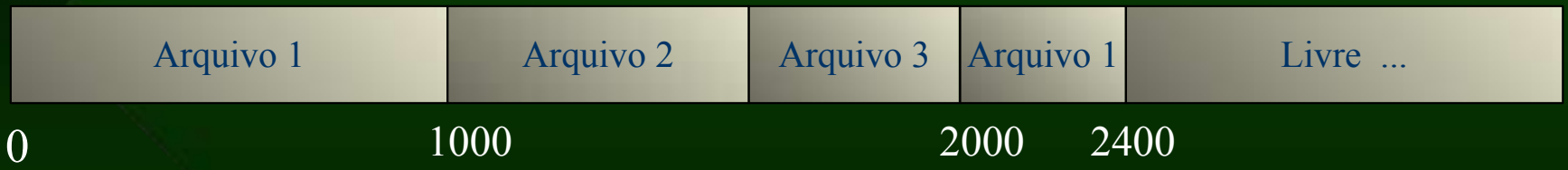
- Sistema trabalha com pré-alocação
- Usuário precisa definir antecipadamente o tamanho do arquivo

Problemas na Alocação

- Maior que a necessária = Desperdício
- Menor que a necessária = Inviável

Sequencial / Alocação Segmentada

Como o Arquivo1 pode crescer?



Informações do Arquivo

- Início, Tamanho, número de segmentos
- Segmento 0 ➔ 0, 1000
- Segmento 1 ➔ 2000, 400

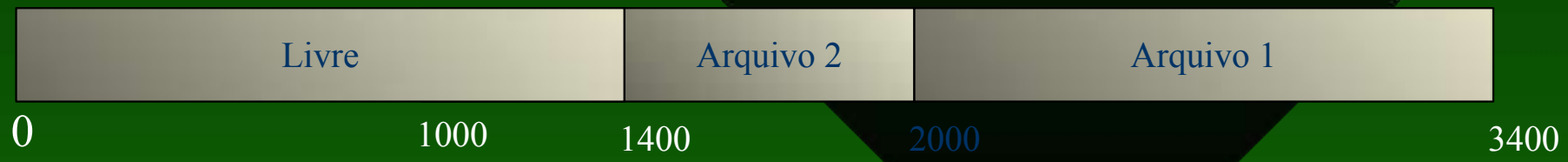
Com muitos segmentos, perdemos desempenho

- Solução? Desfragmentação (arquivo com um único segmento)

Sequencial / Alocação Contígua

Apagando o Arquivo 3

Como alocar um arquivo com 1200 bytes?



Acesso Direto

Arquivo é dividido em blocos que podem ser acessados em qualquer ordem;

Ideal para grande quantidade de informação, como as bases de dados;

Leitura baseada na especificação do número do bloco;

É possível combinar o acesso direto com o acesso sequencial.

Alocação Contígua

Armazenagem do arquivo no disco em blocos sequenciais;

Informações: Nome, Endereço de início e Tamanho;

Problemas:

- O tamanho do arquivo precisa ser definido no instante da sua criação, e de forma definitiva;
- A reserva de espaço “extra” (pré-alocação) pode gerar ociosidade do espaço alocado;
- Alocação para novos arquivos depende da disponibilidade de n blocos no disco, em sequência.

Temos uma situação crítica quando existem espaços livres, mas nenhum deles suporta um novo arquivo.

Uso de Blocos de disco

Definido como a menor unidade de alocação;

Diminui a fragmentação a um único bloco por arquivo (o último);

Perda média = 50% do tamanho do bloco por arquivo (maior para arquivos pequenos, menor para maiores);

Implementações Típicas

Deve haver controle dos espaços livres e dos espaços alocados aos arquivos e diretórios;

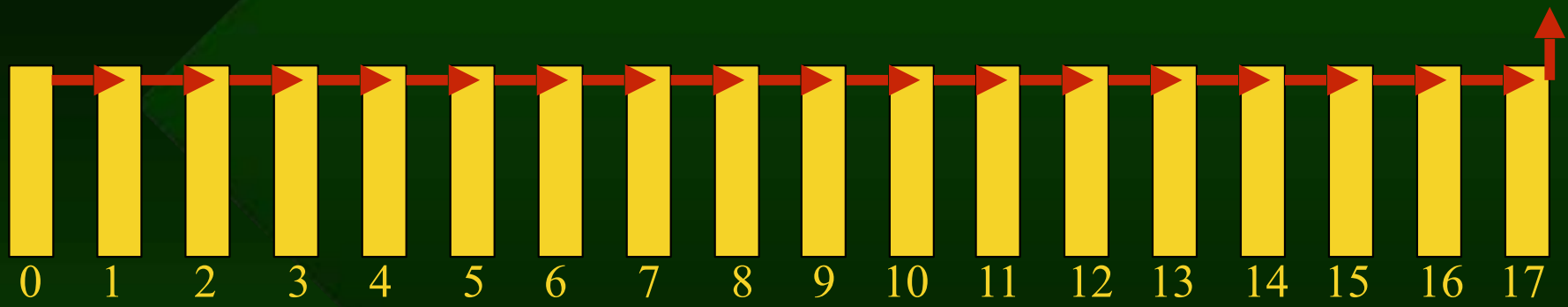
O espaço livre é gerenciado por:

- **Mapa de bits**: cada entrada na tabela é associada a um bit que indica se o bloco está livre (bit=0) ou ocupado (bit=1);

Na ligação encadeada, cada bloco guarda o endereço do próximo bloco, ou aponta para [NULL];

- Uma tabela mantém o endereço do 1º bloco de cada arquivo, ou do espaço livre, e o número de blocos contíguos;

Ligação encadeada

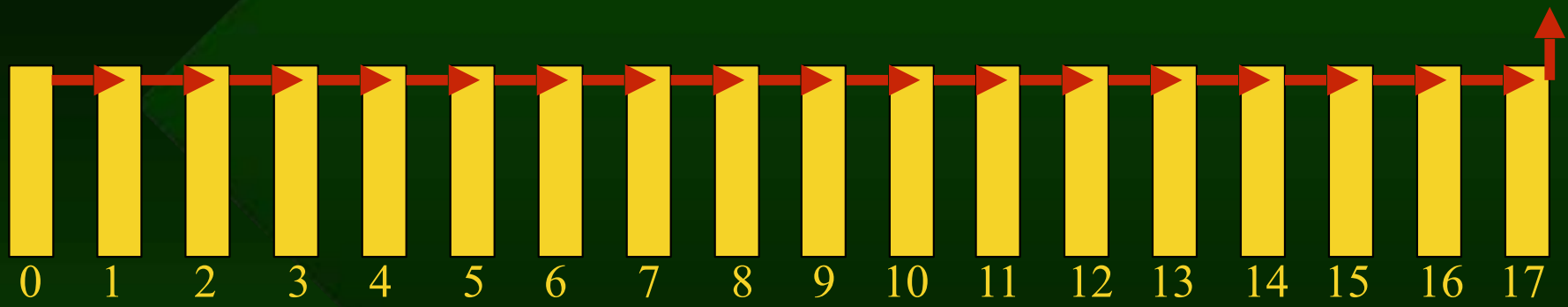


LBL: **L**ista de **B**locos **L**ivres (FBL - ***F****ree* ***B****lock* ***L****ist*)

No exemplo: LBL vai do bloco 0 ao 17

Bloco padrão com 1 KB.

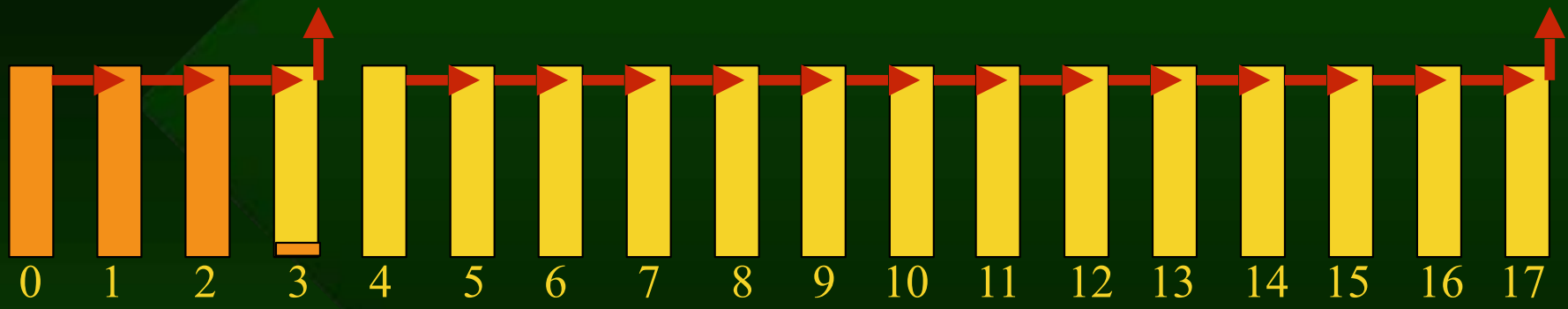
Ligação encadeada



Vamos alocar um arquivo inicial de 3KB;

NOME	Bloco Inicial	Tamanho
LBL	0	18K

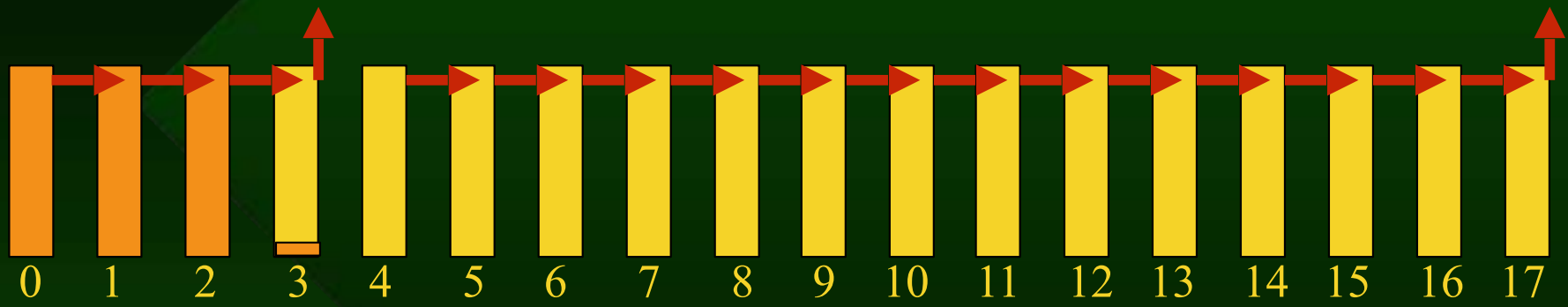
Ligação encadeada



3 blocos ocupados + 12B;

NOME	Bloco Inicial	Tamanho
LBL	4	14K
Arq01	0	3072

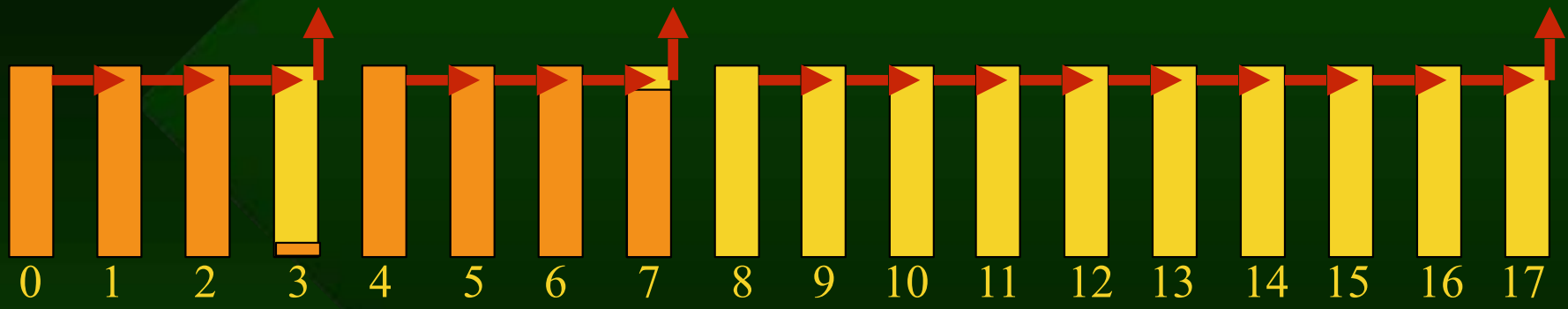
Ligação encadeada



Vamos alocar mais um arquivo com 4000B;

NOME	Bloco Inicial	Tamanho
LBL	4	14K
Arq01	0	3072

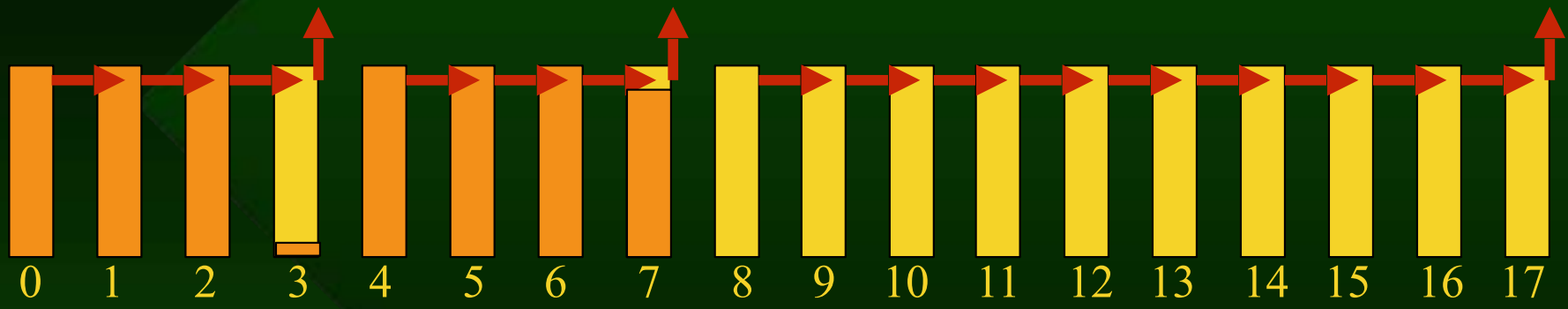
Ligação encadeada



Quase 4 blocos ocupados;

NOME	Bloco Inicial	Tamanho
LBL	8	10K
Arq01	0	3072
Arq02	4	4000

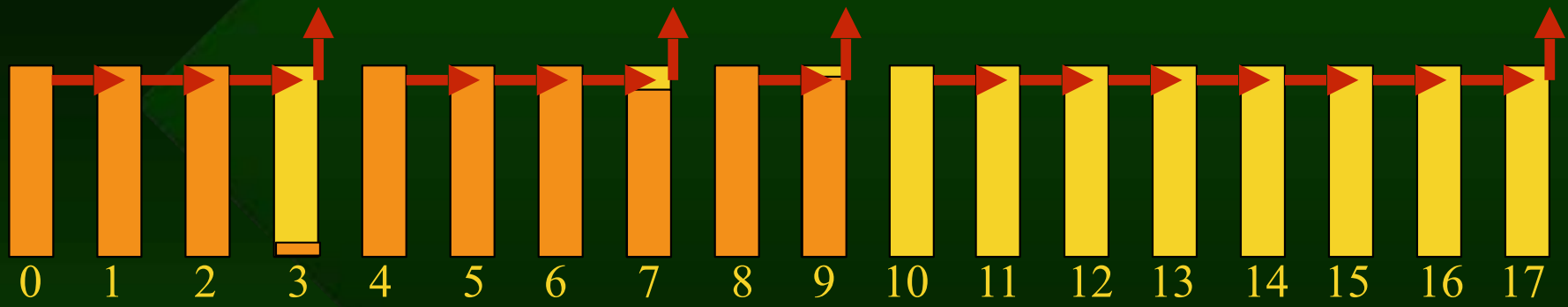
Ligação encadeada



Mais um arquivo com 2000B;

NOME	Bloco Inicial	Tamanho
LBL	8	10K
Arq01	0	3072
Arq02	4	4000

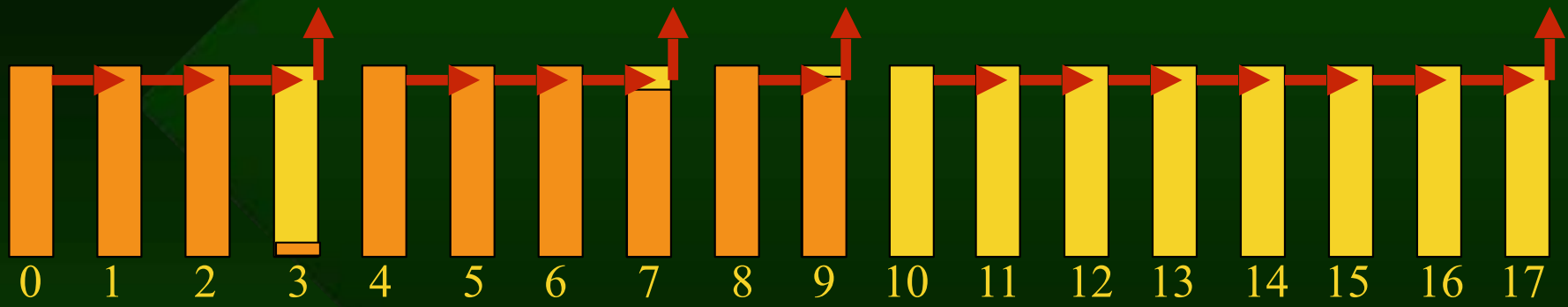
Ligação encadeada



Quase 2 blocos ocupados;

NOME	Bloco Inicial	Tamanho
LBL	10	8K
Arq01	0	3072
Arq02	4	4000
Arq03	8	2000

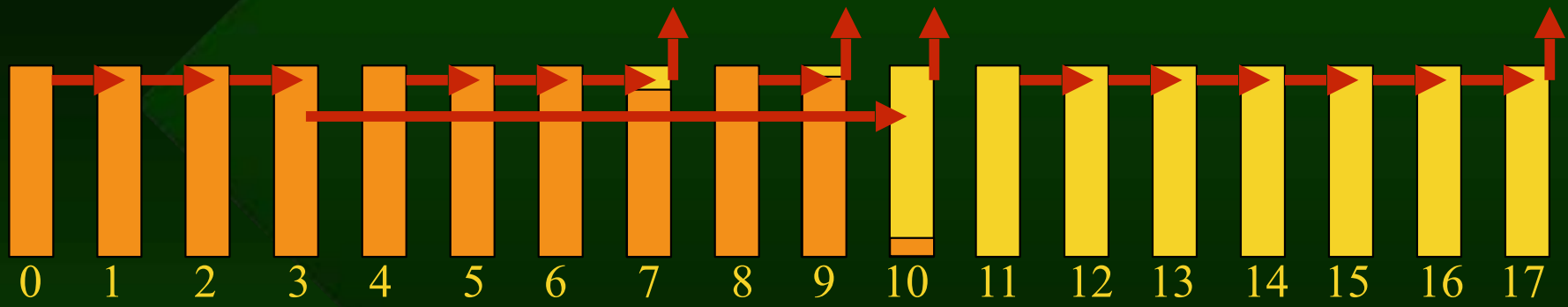
Ligação encadeada



O arquivo 1 cresce de 3072 para 4200 Bytes;

NOME	Bloco Inicial	Tamanho
LBL	10	8K
Arq01	0	3072
Arq02	4	4000
Arq03	8	2000

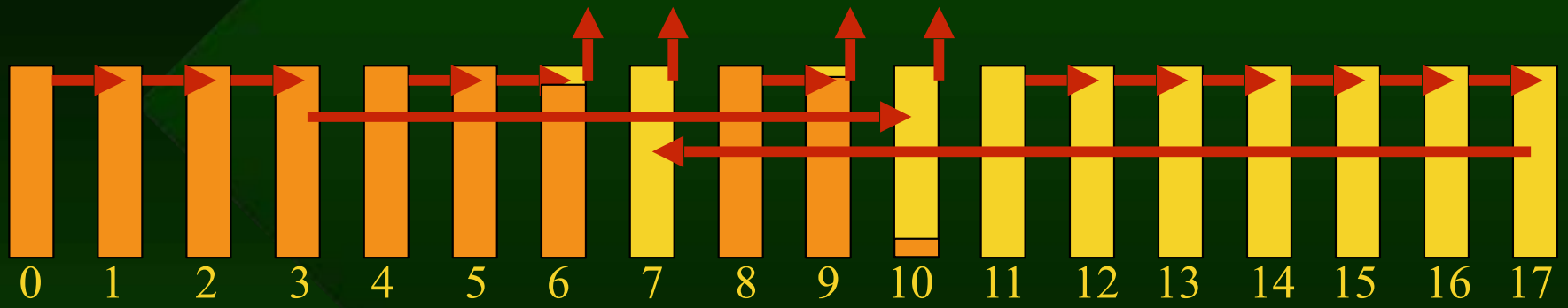
Ligação encadeada



O arquivo 2 diminui de 4000 para 3000 Bytes;

NOME	Bloco Inicial	Tamanho
LBL	11	7K
Arq01	0	4200
Arq02	4	4000
Arq03	8	2000

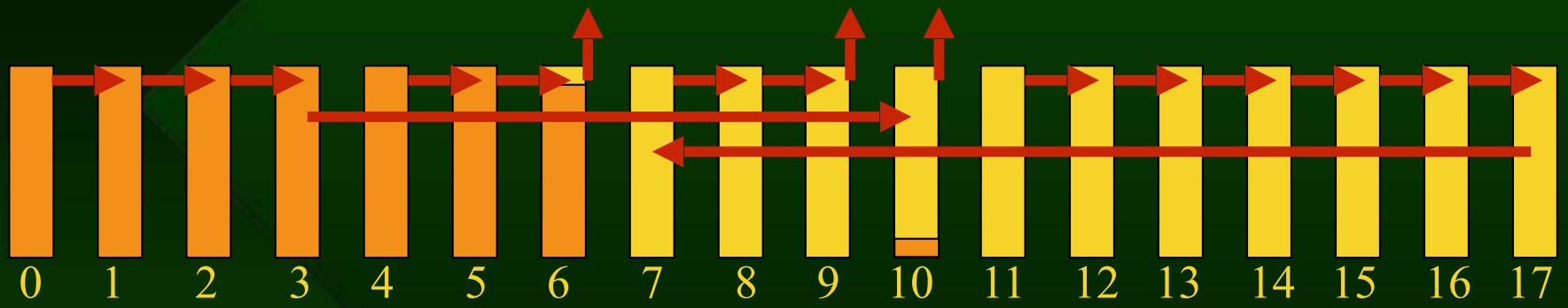
Ligação encadeada



O arquivo 3 é apagado;

NOME	Bloco Inicial	Tamanho
LBL	11	8K
Arq01	0	4200
Arq02	4	3000
Arq03	8	2000

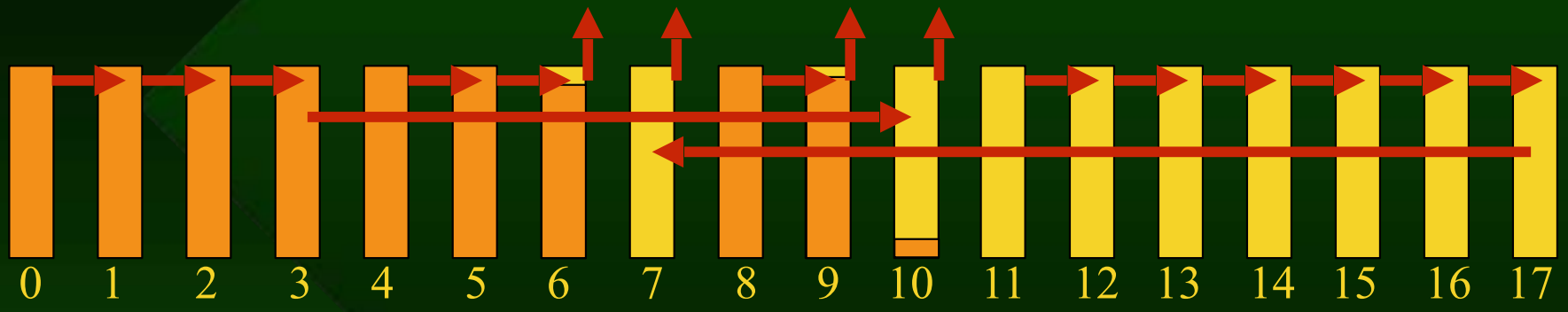
Ligação encadeada



Seria possível "recuperar" o arquivo 3 ?

NOME	Bloco Inicial	Tamanho
LBL	11	10K
Arq01	0	4200
Arq02	4	3000
?rq03	8	2000

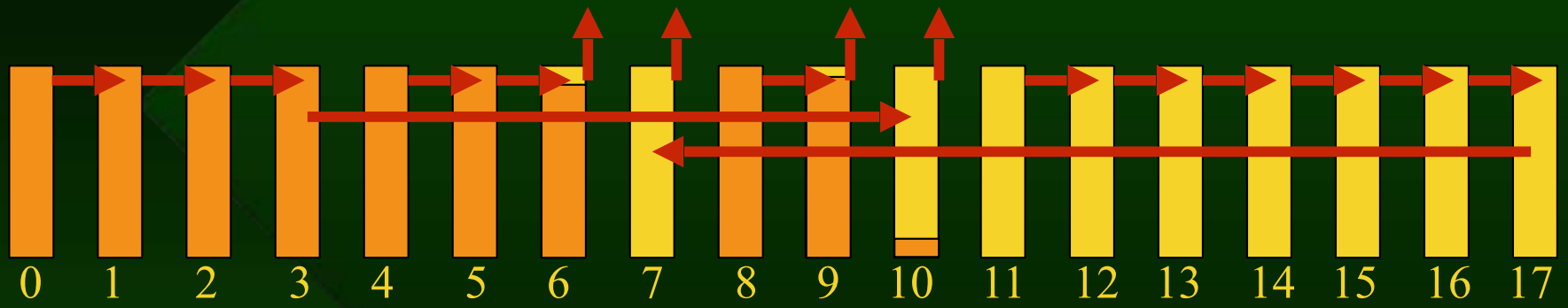
Ligação encadeada



Seria possível "recuperar" o arquivo 3 ?

NOME	Bloco Inicial	Tamanho
LBL	11	8K
Arq01	0	4200
Arq02	4	3000
Arq03	8	2000

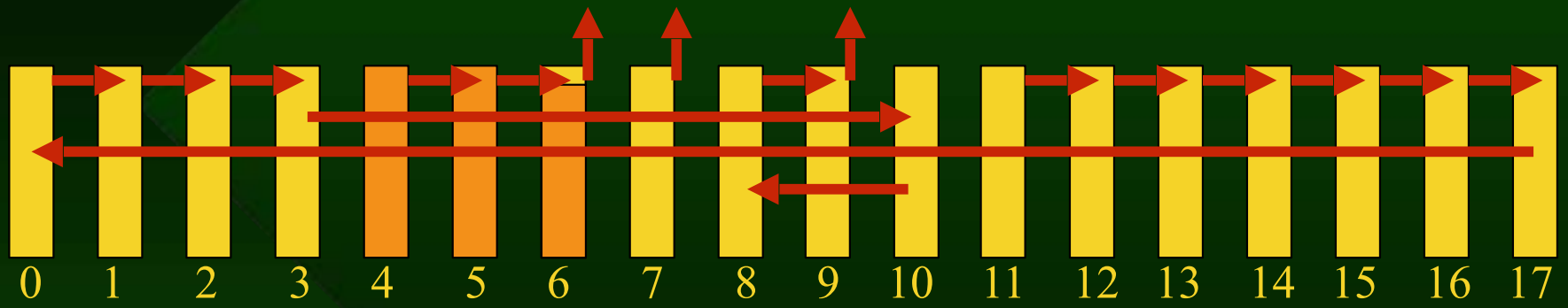
Ligação encadeada



Os arquivos 3 e 1 são apagados;

NOME	Bloco Inicial	Tamanho
LBL	11	8K
Arq01	0	4200
Arq02	4	3000
Arq03	8	2000

Ligação encadeada



Os arquivos 1 e 3 são apagados;

NOME	Bloco Inicial	Tamanho
LBL	11	15K
?rq01	0	4200
Arq02	4	3000
?rq03	8	2000

Alocação Encadeada - Problemas

Acesso aos blocos dos arquivos só pode ser sequencial. Para ler o bloco “n" de um arquivo, “n” blocos precisam ser lidos;

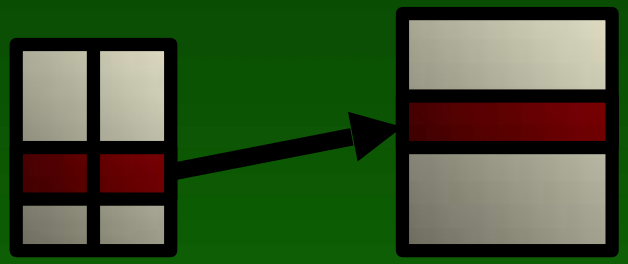
Espaço desperdiçado nos blocos para o armazenamento dos ponteiros.

Se o arquivo ficar fragmentado, a leitura sequencial é pior do que na alocação contígua;

A vantagem sobre a alocação contígua resume-se ao desperdício devido à fragmentação interna.

Métodos de Acesso ao Arquivo

- Acesso indexado
  - Sofisticação do acesso direto;
  - Chamado de acesso indexado ou por chave;
  - Arquivo deve possuir uma área de índice onde existam ponteiros para os diversos registros.



Alocação Indexada

Informações: Nome, endereço do índice + índice;

Ponteiros para os blocos do arquivo são mantidos numa estrutura chamada **bloco de índice**;

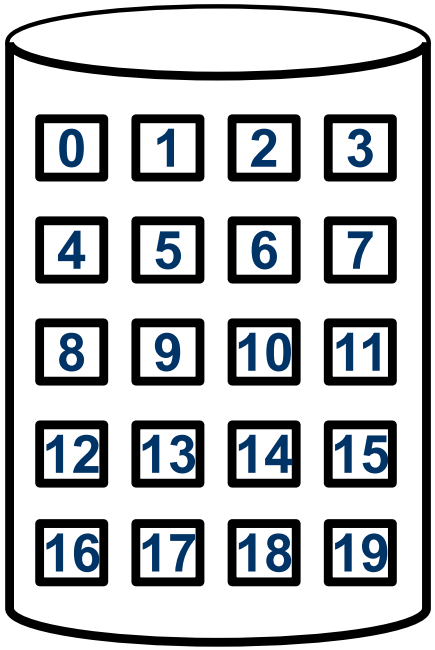
Permite acesso direto sem fragmentação;

Não utiliza informação de controle nos blocos como na alocação encadeada.

Alocação Indexada de Blocos

Linha	Valor
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	10
10	11
11	12
12	13
13	14
14	15
15	16
16	17
17	18
18	19
19	999

FAT



Diretório

Arquivo	1o. Bloco	Tamanho
LBL	0	20480

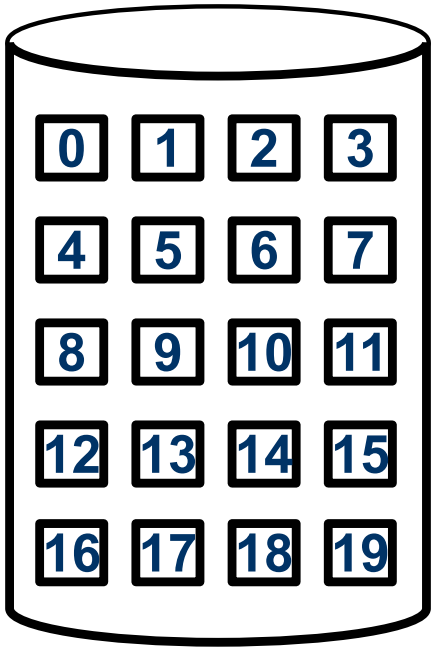
Bloco = 1 KB = 1024 bytes

A primeira coluna da tabela [linha] é virtual, pois identifica a linha.

Alocação Indexada de Blocos

Linha	Valor
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	10
10	11
11	12
12	13
13	14
14	15
15	16
16	17
17	18
18	19
19	999

FAT



Diretório

Arquivo	1o. Bloco	Tamanho
LBL	0	20480

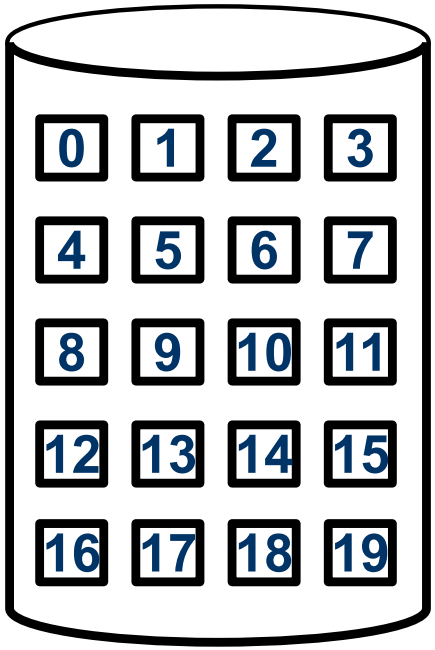
Qual o primeiro bloco da LBL?
Resp: 0
Qual o segundo bloco da LBL?
Resp: Na FAT, para onde o bloco 0 aponta? Busca-se na linha 0.

O último bloco deve apontar para EOF.
O que é EOF? Um valor predeterminado.
No exemplo assumiu-se 999.

Alocação Indexada de Blocos

Linha	Valor
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	10
10	11
11	12
12	13
13	14
14	15
15	16
16	17
17	18
18	19
19	999

FAT



Diretório

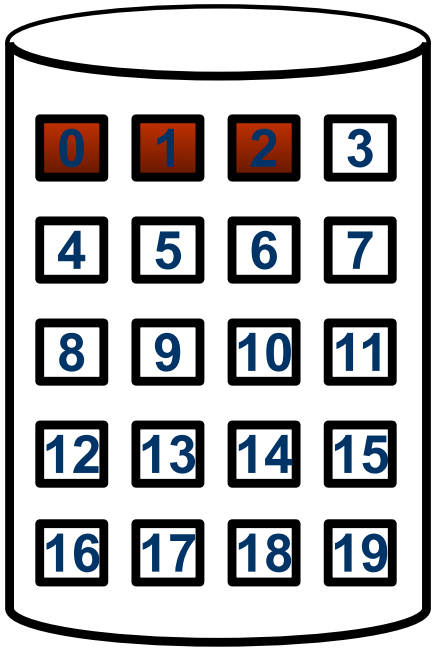
Arquivo	1o. Bloco	Tamanho
LBL	0	20480

Alocar o arquivo 1 com 2872 bytes
Precisamos de 3 blocos livres.

Alocação Indexada de Blocos

Linha	Valor
0	1
1	2
2	999
3	4
4	5
5	6
6	7
7	8
8	9
9	10
10	11
11	12
12	13
13	14
14	15
15	16
16	17
17	18
18	19
19	999

FAT



Alocar o arquivo 2 com 4020 bytes

Diretório

Arquivo	1o. Bloco	Tamanho
LBL	3	17408
Arquivo 1	0	2872

Alocar o arquivo 1 com 2872 bytes
Precisamos de 3 blocos livres.

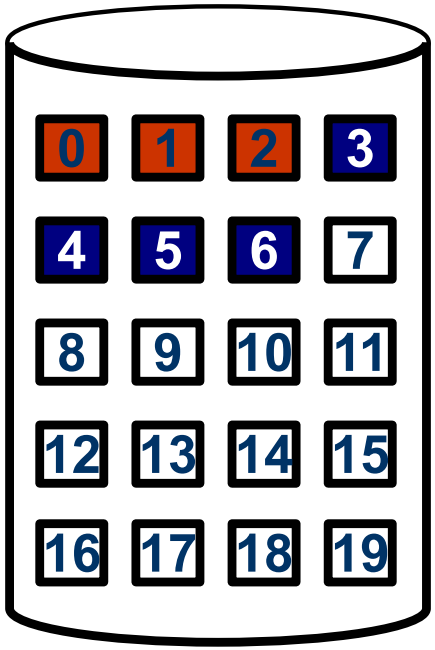
Bloco = 1 KB = 1024 Bytes

Blocos do Arquivo 1 = [0,1, 2]

Alocação Indexada de Blocos

Linha	Valor
0	1
1	2
2	999
3	4
4	5
5	6
6	999
7	8
8	9
9	10
10	11
11	12
12	13
13	14
14	15
15	16
16	17
17	18
18	19
19	999

FAT



Aumentar o arquivo 1 em 1030 bytes

Diretório

Arquivo	1o. Bloco	Tamanho
LBL	7	13312
Arquivo 1	0	2872
Arquivo 2	3	4020

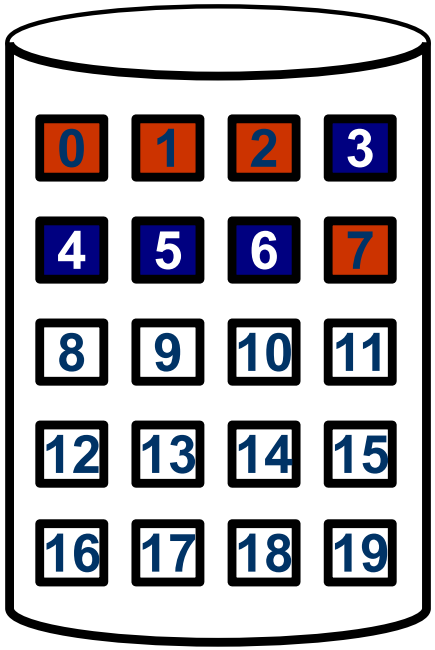
Alocar o arquivo 1 com 2872 bytes
Alocar o arquivo 2 com 4020 bytes

Bloco = 1 KB = 1024 Bytes
Blocos do Arquivo 1 = [0,1, 2]
Blocos do Arquivo 2 = [3, 4, 5, 6]

Alocação Indexada de Blocos

Linha	Valor
0	1
1	2
2	7
3	4
4	5
5	6
6	999
7	999
8	9
9	10
10	11
11	12
12	13
13	14
14	15
15	16
16	17
17	18
18	19
19	999

FAT



Criar o arquivo 3 com 5211 bytes

Diretório

Arquivo	1o. Bloco	Tamanho
LBL	8	12288
Arquivo 1	0	3902
Arquivo 2	3	4020

Aumentar o arquivo 1 em 1030 bytes

Bloco = 1 KB = 1024 Bytes

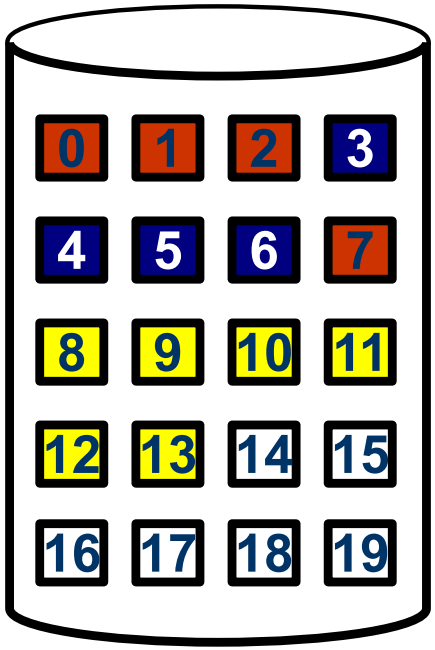
Blocos do Arquivo 1 = [0,1, 2, 7]

Blocos do Arquivo 2 = [3, 4, 5, 6]

Alocação Indexada de Blocos

Linha	Valor
0	1
1	2
2	7
3	4
4	5
5	6
6	999
7	999
8	9
9	10
10	11
11	12
12	13
13	999
14	15
15	16
16	17
17	18
18	19
19	999

FAT



Aumentar o arquivo 1 em 2405 bytes

Diretório

Arquivo	1o. Bloco	Tamanho
LBL	14	6144
Arquivo 1	0	3902
Arquivo 2	3	4020
Arquivo 3	8	5211

Criar o arquivo 3 com 5211 bytes

Bloco = 1 KB = 1024 Bytes

Blocos do Arquivo 1 = [0,1, 2, 7]

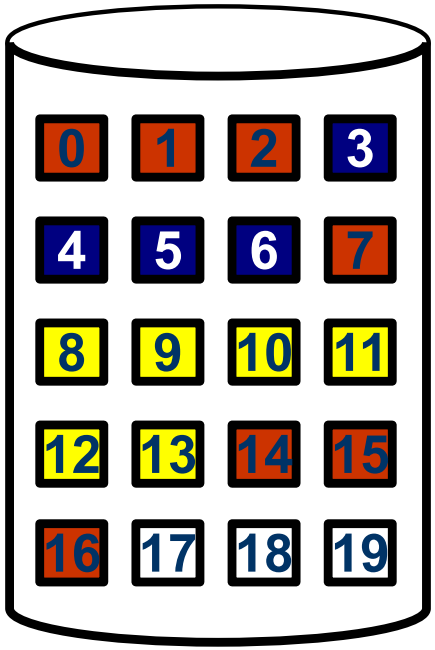
Blocos do Arquivo 2 = [3, 4, 5, 6]

Blocos do Arquivo 3 = [8, 9, 10, 11, 12, 13]

Alocação Indexada de Blocos

Linha	Valor
0	1
1	2
2	7
3	4
4	5
5	6
6	999
7	14
8	9
9	10
10	11
11	12
12	13
13	999
14	15
15	16
16	999
17	18
18	19
19	999

FAT



Apagar o arquivo 1

Diretório

Arquivo	1o. Bloco	Tamanho
LBL	17	3072
Arquivo 1	0	6307
Arquivo 2	3	4020
Arquivo 3	8	5211

Aumentar o arquivo 1 em 2405 bytes

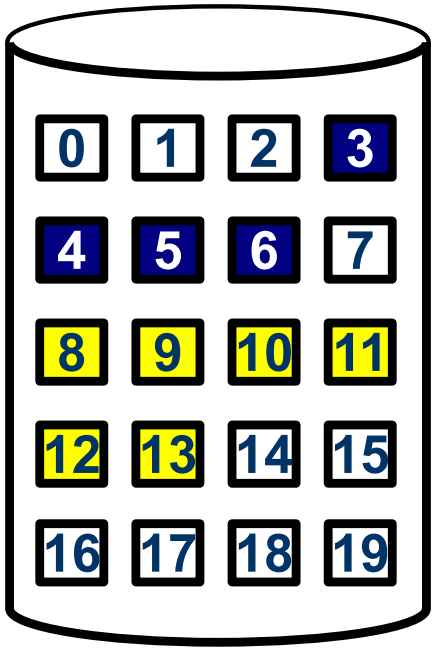
Bloco = 1 KB = 1024 Bytes
LBL = [17, 18, 19]

Blocos do Arquivo 1 = [0,1, 2, 7, 14, 15, 16]
Blocos do Arquivo 2 = [3, 4, 5, 6]
Blocos do Arquivo 3 = [8, 9, 10, 11, 12, 13]

Alocação Indexada de Blocos

Linha	Valor
0	1
1	2
2	7
3	4
4	5
5	6
6	999
7	14
8	9
9	10
10	11
11	12
12	13
13	999
14	15
15	16
16	999
17	18
18	19
19	0

FAT



Recuperar o arquivo 1

Diretório

Arquivo	1o. Bloco	Tamanho
LBL	17	10240
Arquivo 1	0	6307
Arquivo 2	3	4020
Arquivo 3	8	5211

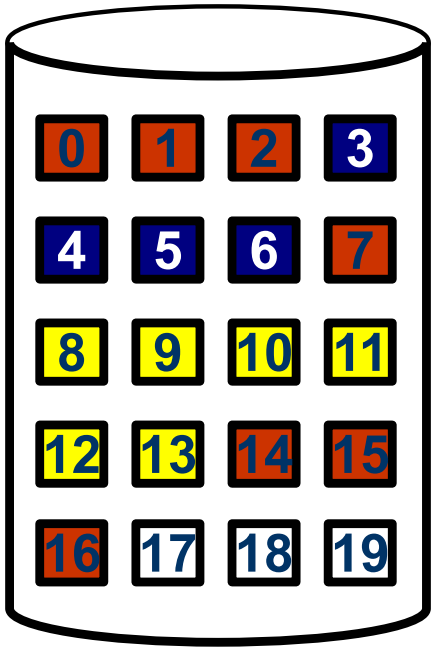
Bloco = 1 KB = 1024 Bytes
LBL = [17, 18, 19, 0, 1, 2, 7, 14, 15, 16]

Blocos do Arquivo 1 = [0,1, 2, 7, 14, 15, 16]
Blocos do Arquivo 2 = [3, 4, 5, 6]
Blocos do Arquivo 3 = [8, 9, 10, 11, 12, 13]

Alocação Indexada de Blocos

Linha	Valor
0	1
1	2
2	7
3	4
4	5
5	6
6	999
7	14
8	9
9	10
10	11
11	12
12	13
13	999
14	15
15	16
16	999
17	18
18	19
19	999

FAT



Apagar o arquivo 1 e o arquivo 2

Diretório

Arquivo	1o. Bloco	Tamanho
LBL	17	3072
Arquivo 1	0	6307
Arquivo 2	3	4020
Arquivo 3	8	5211

Recuperar o arquivo 1

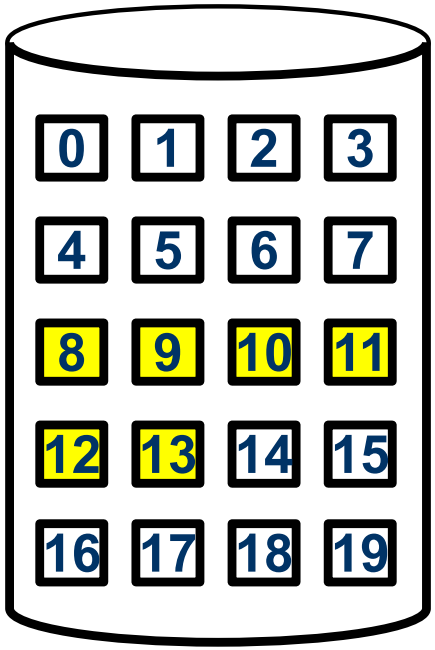
Bloco = 1 KB = 1024 Bytes
LBL = [17, 18, 19]

Blocos do Arquivo 1 = [0,1, 2, 7, 14, 15, 16]
Blocos do Arquivo 2 = [3, 4, 5, 6]
Blocos do Arquivo 3 = [8, 9, 10, 11, 12, 13]

Alocação Indexada de Blocos

Linha	Valor
0	1
1	2
2	7
3	4
4	5
5	6
6	999
7	14
8	9
9	10
10	11
11	12
12	13
13	999
14	15
15	16
16	3
17	18
18	19
19	0

FAT



Recuperar o arquivo 2

Diretório

Arquivo	1o. Bloco	Tamanho
LBL	17	14336
Arquivo 1	0	6307
Arquivo 2	3	4020
Arquivo 3	8	5211

Bloco = 1 KB = 1024 Bytes

LBL = [17, 18, 19, 0, 1, 2, 7, 14, 15, 16, 3, 4, 5, 6]

Blocos do Arquivo 1 = [0,1, 2, 7, 14, 15, 16]

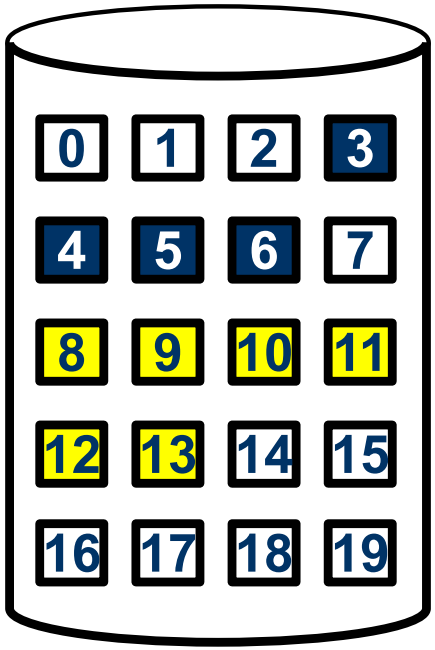
Blocos do Arquivo 2 = [3, 4, 5, 6]

Blocos do Arquivo 3 = [8, 9, 10, 11, 12, 13]

Alocação Indexada de Blocos

Linha	Valor
0	1
1	2
2	7
3	4
4	5
5	6
6	999
7	14
8	9
9	10
10	11
11	12
12	13
13	999
14	15
15	16
16	999
17	18
18	19
19	0

FAT



Apagar o arquivo 2

Diretório

Arquivo	1o. Bloco	Tamanho
LBL	17	10240
Arquivo 1	0	6307
Arquivo 2	3	4020
Arquivo 3	8	5211

Recuperar o arquivo 2

Bloco = 1 KB = 1024 Bytes

Blocos do Arquivo 1 = [0,1, 2, 7, 14, 15, 16]

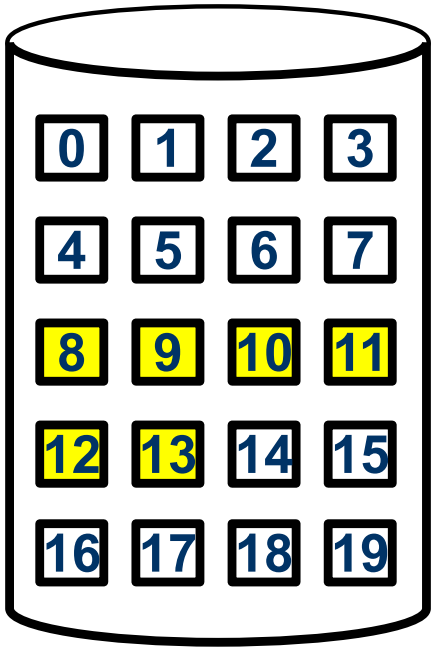
Blocos do Arquivo 2 = [3, 4, 5, 6]

Blocos do Arquivo 3 = [8, 9, 10, 11, 12, 13]

Alocação Indexada de Blocos

Linha	Valor
0	1
1	2
2	7
3	4
4	5
5	6
6	999
7	14
8	9
9	10
10	11
11	12
12	13
13	999
14	15
15	16
16	3
17	18
18	19
19	0

FAT



Recuperar o arquivo 1

Diretório

Arquivo	1o. Bloco	Tamanho
LBL	17	14336
Arquivo 1	0	6307
Arquivo 2	3	4020
Arquivo 3	8	5211

Apagar o arquivo 2

Bloco = 1 KB = 1024 Bytes

Blocos do Arquivo 1 = [0, 1, 2, 7, 14, 15, 16]

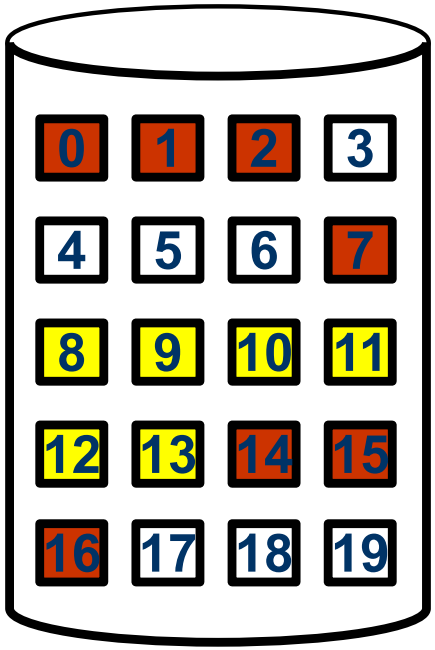
Blocos do Arquivo 2 = [3, 4, 5, 6]

Blocos do Arquivo 3 = [8, 9, 10, 11, 12, 13]

Alocação Indexada de Blocos

Linha	Valor
0	1
1	2
2	7
3	4
4	5
5	6
6	999
7	14
8	9
9	10
10	11
11	12
12	13
13	999
14	15
15	16
16	999
17	18
18	19
19	3

FAT



Diretório

Arquivo	1o. Bloco	Tamanho
LBL	17	7168
Arquivo 1	0	6307
Arquivo 2	3	4020
Arquivo 3	8	5211

Recuperar o arquivo 1

Bloco = 1 KB = 1024 Bytes

Blocos do Arquivo 1 = [0,1, 2, 7, 14, 15, 16]

Blocos do Arquivo 2 = [3, 4, 5, 6]

Blocos do Arquivo 3 = [8, 9, 10, 11, 12, 13]

FAT de 16 bits

Valor
1
2
3
4
5
6
7
8
9
10
11
...
1000
1002
...
...
65532
65533
65534
65535

FAT de 16 bits → 2^{16} = 65.536 linhas → 65536 blocos

Cada linha da FAT16 possui 16 bits

Tamanho da FAT16 na memória = 2^{16} * 16 bits
= 64K * 2 bytes
= 128 K bytes

Os 128 KB têm que ser espaço de memória principal !

Como precisamos de valores específicos para marcar o fim de arquivo, e eventuais blocos defeituosos, nem todos os blocos podem ser usados, e alguns são desperdiçados.

FAT de 16 bits

Valor
1
2
3
4
5
6
7
8
9
10
11
...
1000
1002
...
...
65532
65533
65534
65535

Tamanho do bloco = Tamanho da unidade / Número de linhas

Formatando Unidades de Armazenamento
256 GB → 256GB / 64K = 4MB, ou seja, bloco de 4MB
1 TB → Bloco: 16 MB

Com a fragmentação interna, a perda será de 8MB x nº arquivos.

Cache

Cache

- Maior desempenho para acessos repetidos a um mesmo bloco de disco;
  - Cache de Blocos na memória;
  - Blocos mais recentemente referenciados do disco;
  - Substitui os LRU - *Least Recently Used*
- Maior desempenho, Menor confiabilidade
  - Lê e grava apenas no cache
  - Em caso de falha (energia?), todas as alterações são perdidas;
  - Risco de falha de consistência.
- Priorizar gravação de itens essenciais?

Cache: Desempenho X Confiabilidade

- I/O de bloco essencial
  - Lê no Cache e grava no Cache e no Disco
- I/O de bloco não essencial
  - Lê e Grava no Cache
- O que acontece se faltar energia ao sistema?
  - O sistema fica íntegro e as alterações do usuário são perdidas.

Segurança em Sistemas Operacionais

Segurança em Sist.Operacionais

- Envolve o ambiente externo;
- Envolve riscos de ...
  - acesso não autorizado
  - destruição ou alteração maliciosa;
  - introdução de inconsistências.
- Alguns conceitos:
  - Vulnerabilidade, Ameaça, Ataque, invasor/cracker
- Proteção total é impossível?
  - Ao menos devemos tornar o ataque muito caro

Ataques típicos

Brecha de sigilo

Acesso não autorizado a dados sigilosos. Ex: Info de C.C.

Brecha de integridade

Modificação não autorizada de dados ou aplicações

Brecha de disponibilidade

Destruição de dados, ou de infraestrutura

Roubo de serviços

Uso não autorizado de serviços

Recusa de serviços

Conhecido como DOS (Denial of Service), impede o uso legítimo de sistemas de informação

Métodos típicos

- Mascaramento
  - Violação da autenticação de usuários legítimos
- Reexecução
  - Repetição fraudulenta de uma transação
- Modificação de mensagens
- Ataque do Intermediário
  - Invasor se instala no fluxo de dados
  - Sequestro de sessão

Níveis de Proteção (externos ao SO)

- Físico
  - Acesso físico aos dispositivos deve ser controlado;
  - Acesso forçado ou furtivo deve ser impedido.
- Humano
  - Autorização e autenticação cuidadosa. Ex: Autenticação forte;
  - Treinamento; abordar técnicas de engenharia social e *phishing*.

Níveis de Proteção (internos ao SO)

- SO
  - Falha de processos podem provocar DOS, por exemplo;
  - Troca de mensagens pode revelar dados de autenticação;
  - Estouro de pilha pode liberar processos não autorizados;
  - E muitas outras falhas ...
- Rede
  - Quando o acesso físico não é possível, a rede é a opção do *cracker*;
  - Interceptação de mensagens não criptografadas;
  - Mascaramento;
  - Negação de serviços de rede.

Recursos internos do SO

Autenticação de usuários

Atributos de Segurança no Sist.de Arquivos

LOG de atividades de sistema

Proteção de *hardware*. Ex.: memória

Ameaças de Aplicações

Cavalo de Tróia

- Programas construídos para usuários com privilégios restritos de segurança não devem ser executados por usuários menos restritos;
- Emulação de login. Ex: Ctrl+Alt+Del;
- *Spyware* e os canais ocultos;
- Princípio do privilégio mínimo.

Alçapão

- O código pode tratar de forma diferenciada um usuário específico;
- Compiladores poderiam incluir trechos de código independente do código-fonte? Quem verificaria?

Ameaças de Aplicações

- Bomba Lógica
  - Brechas de segurança ativadas por condições específicas
    Ex.: demissão do programador?
- Estouro de pilha
  - Programador substitui endereço de retorno da pilha para apontar código malicioso.
- Vírus

Ameaças de Redes

Worms (vermes)

- Explora vulnerabilidades das redes e serviços para inserir códigos maliciosos nos dispositivos conectados.

Varredura de Portas

- Não é um ataque, mas identifica as vulnerabilidades para ataques futuros;
- Normalmente são facilmente detectados, e por isso são disparados a partir de máquinas “zumbis”.

DOS

- Normalmente não podem ser evitados;
- A variação DDOS (DOS distribuído) é ainda mais complexa de evitar.

Entrada e Saída (I/O)

Processar ou Comunicar?

Muitas vezes enviar e ler informações é até mais importante do que processar

- Página WEB;
- Manipulação de Bases de Dados;
- Edição de Arquivos.

Comunicar pode ser bem complexo

- Enorme variedade de *hardwares*;
- Barramentos, portas e controladoras;
- *Drivers* de dispositivos.

Desafios de *Hardware* e *Software*.

Hardware de Comunicação

Ponto a ponto

- Cabos interligam "portas";
- UART (*Universal Asynchronous Receiver/Transmitter*);
- USB (*Universal Serial Bus*);
- HDMI (*High Definition Multimedia Interface*);
- SATA (*Serial Advanced Technology Attachment*).

Multiponto

- Barramentos interligam diversos dispositivos;
- Barramentos internos (endereços, dados e controle);
- PCI (*Peripheral Component Interconnect*);
- i2C (*Inter-Integrated Circuit*);

Transferência de Informações

- Leitura e Escrita de Registradores
  - Processador lê ou escreve em I/O;
  - Tipicamente ocorre manipulação de registros internos do periférico;
  - Técnica tem conectividade simples, porém performance baixa.
- Mapeamento de Memória
  - Processador compartilha bloco de memória com periférico a ser lido/escrito;
  - A leitura e a escrita ocorrem na memória principal pelo processador.
- Duas técnicas
  - Alguns dispositivos usam as 2 técnicas (placas gráficas).

Execução do Processo

- Polling
  - Periféricos são acessados individualmente;
  - Interatividade pode ser acessada;
  - Parte do tempo de processamento é perdida.
- Interrupção
  - Interatividade preservada;
  - Habilita a resposta a eventos assíncronos (por demanda);

Tipos de Interrupção

- Mascaráveis
  - Tipicamente utilizada para tratar eventos disparados por dispositivos de I/O;
  - Pode ser adiada em função da execução de trechos de código críticos;
  - Normalmente obedecem a um esquema de prioridade.
- Não mascaráveis
  - Trata eventos emergenciais, como erros de alocação de memória, invasão de áreas protegidas, acesso privilegiado de usuários não autorizados etc;
  - Trata as chamadas “exceções”.

DMA (*Direct Memory Access*)

- Processador especializado
  - Processadores de uso geral tipicamente não são eficientes na manipulação de grandes quantidades de memória.
- Requisição de Barramento
  - Para operar, o DMA não permite que a CPU acesse a memória;
  - A CPU, no entanto, continua tendo acesso aos seus registradores e cache;
  - A política de escrita no cache pode trazer problemas