



Infraestrutura de Software

Introdução



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO



Introdução

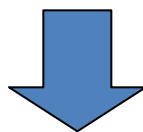
- Sistema Operacional
 - Mecanismo de abstração dos dispositivos subjacentes
 - Gerenciador de recursos (ex. processador, memória, impressora)
- Middleware
 - Plataforma de suporte de valor agregado a sistemas distribuídos



Computador Típico

- Componentes físicos (**hardware**)
 - Um ou mais processadores
 - Memória
 - Discos
 - Impressoras
 - Vários outros dispositivos de E/S (tela, mouse...)

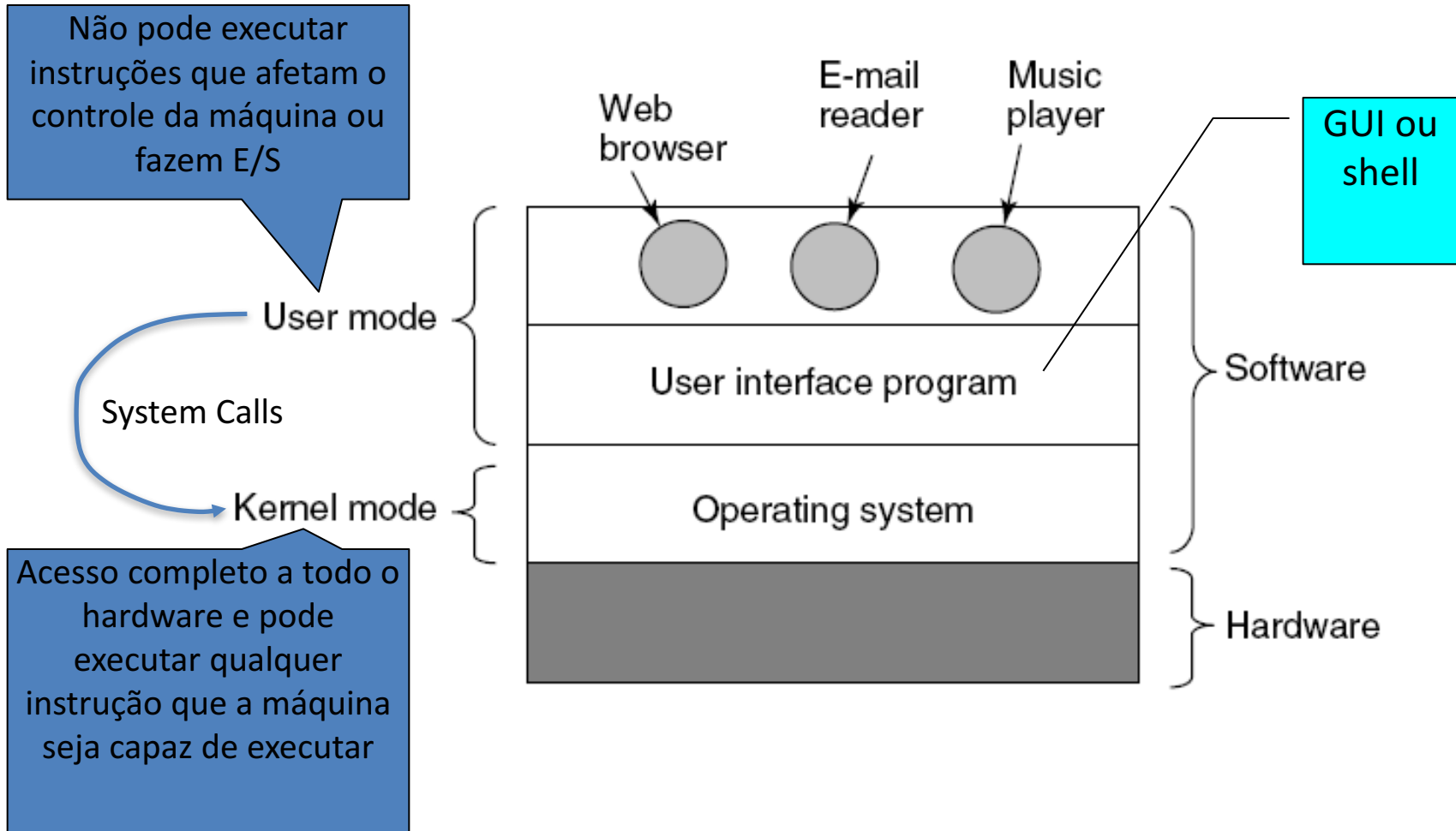
**Um Sistema
Complexo!!!**



- Gerenciar todos estes componentes requer **abstração** – *um modelo mais simples do computador*
- É isso que é o **sistema operacional**



Sistema Computacional em Camadas





Gerenciador de Recursos

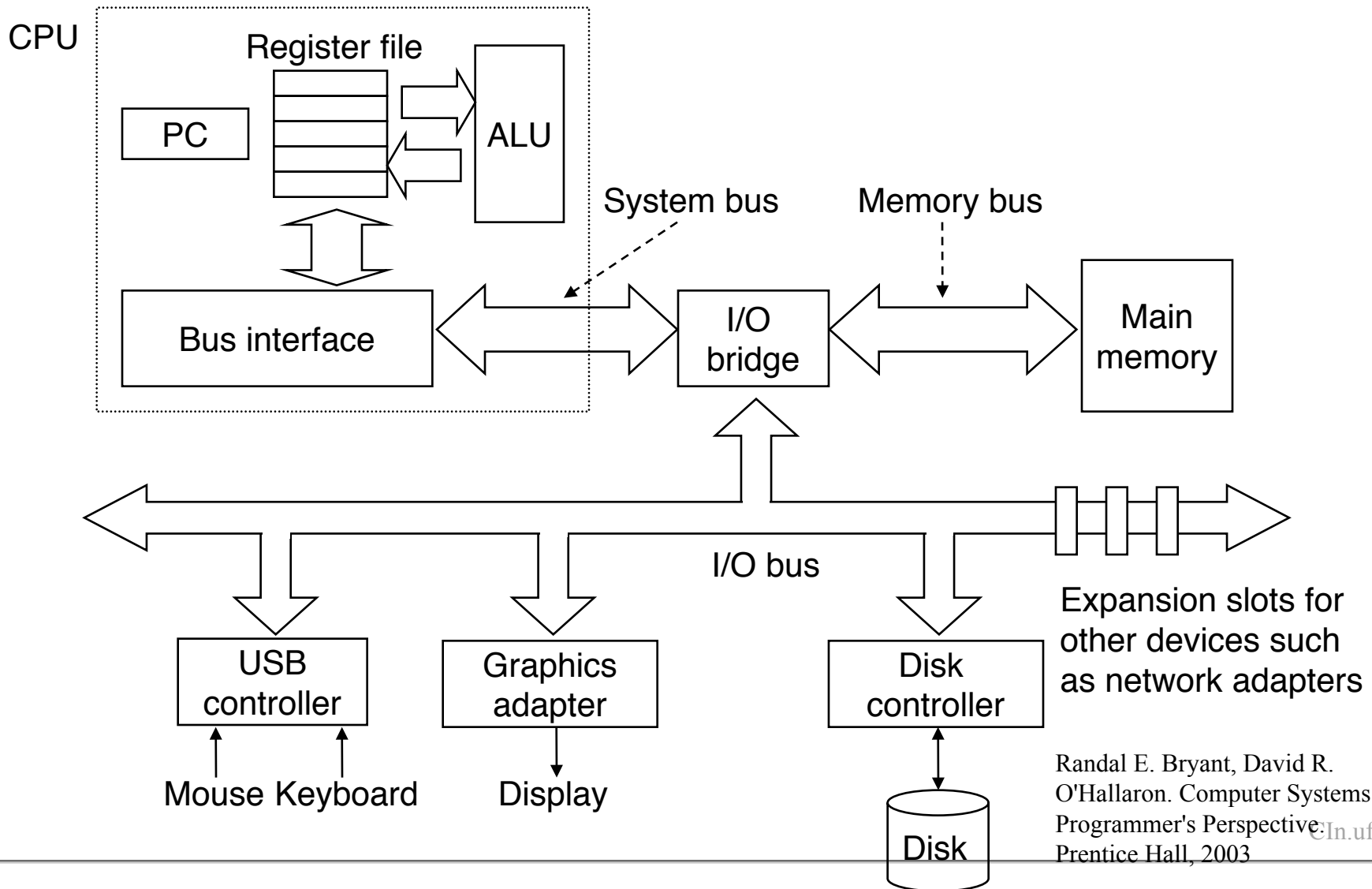
- **Gerencia** e **protege** memória, dispositivos de E/S e outros recursos (hardware)
- Permite o compartilhamento (multiplexação) de recursos
 - no tempo (time-sharing)
 - Ex.: múltiplos programas compartilham o processador (executam) ao “mesmo” tempo
 - no espaço
 - Ex.: dados de diferentes usuários/arquivos compartilham o espaço em disco

Hardware

Hardware



Um pouco de um computador típico

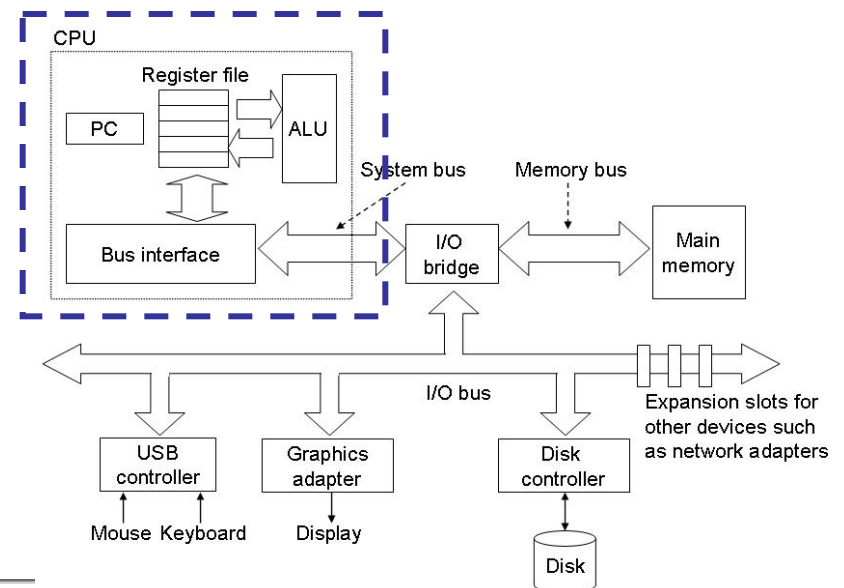




CPU: Central Processing Unit

- Unidade de Controle
- ALU: Unidade Aritmética e Lógica
- Registradores
 - Funcionam como **memória** de acesso **extremamente rápida**
 - **Baixa capacidade** de armazenamento
 - Funções específicas
 - Exemplos de registradores
 - PC (program counter): contém o endereço da próxima instrução a ser executada
 - Instruction register: onde é copiada cada instrução a ser executada

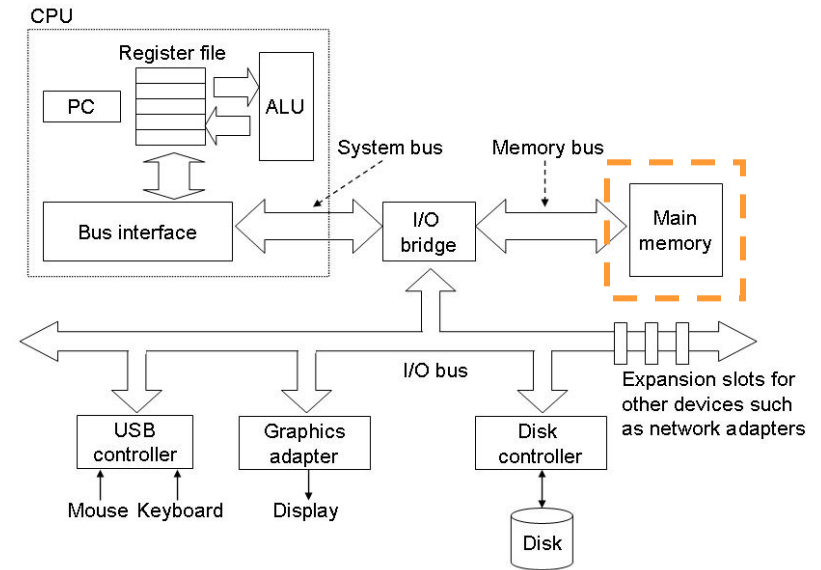
- A CPU, seguidamente, executa instruções requisitadas à memória
 - Ciclo *fetch-decode-execute*:
 1. busca instrução na memória
 2. atualiza PC
 3. decodifica instrução
 4. executa instrução





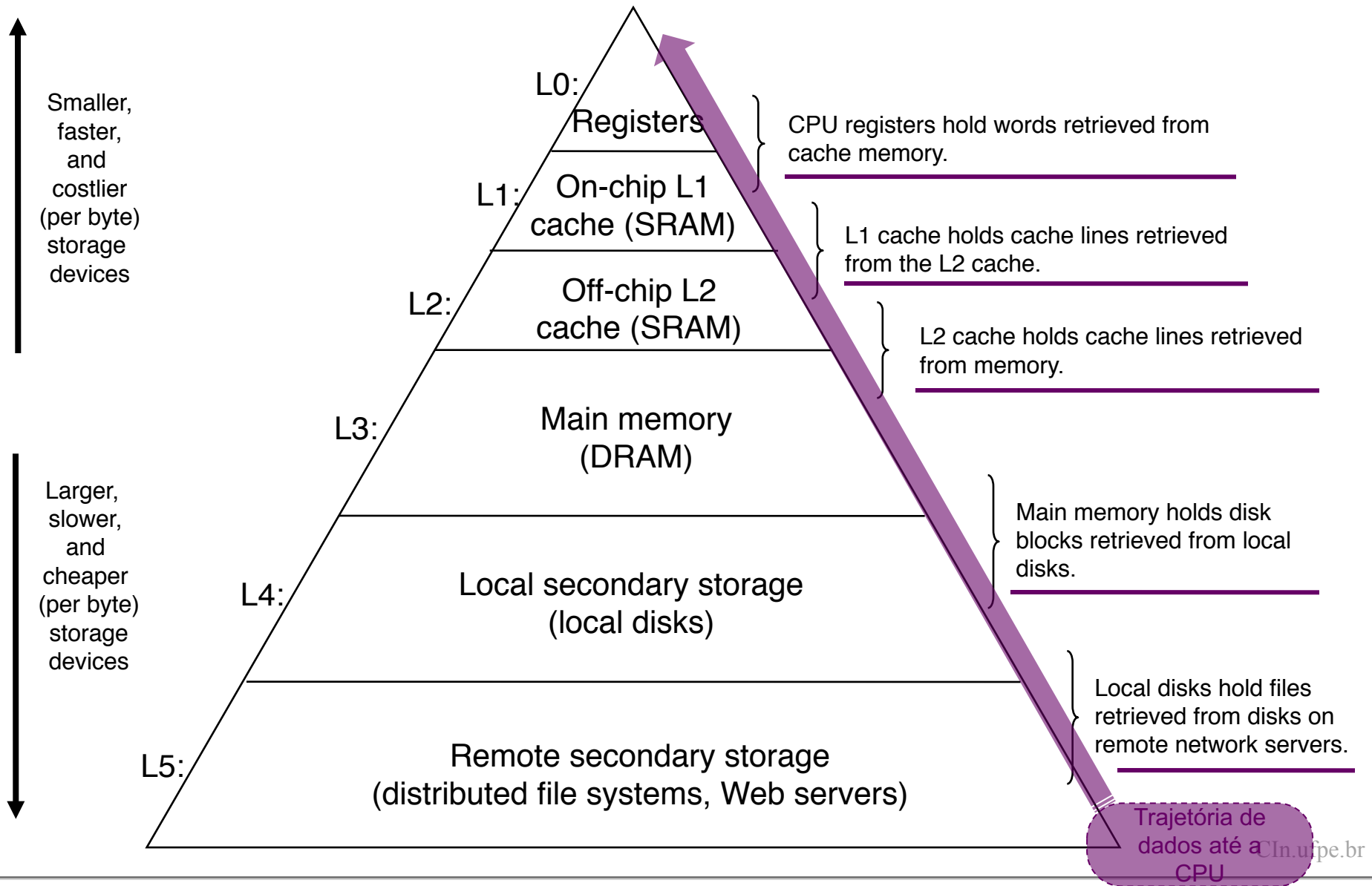
Memória

- Logicamente, a memória principal corresponde a um enorme vetor (array) de bytes
 - cada posição tem um endereço único (índice do vetor)
- Os registradores da CPU muitas **vezes** são usados para armazenar endereços de memória
 - Assim, o número de bits em cada registrador **limita** o número de **posições de memória endereçáveis**
 - Ex.: 8 bits → 256 posições...





Hierarquia de Memória



Software Básico

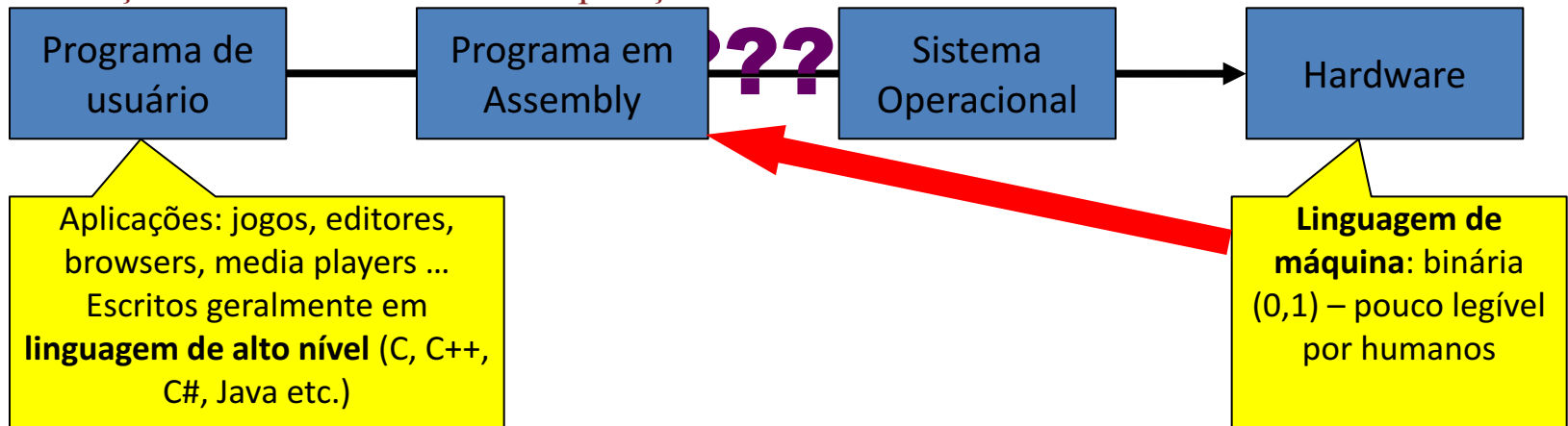
SOFTWARE BÁSICO



Software Básico

[A. Raposo e M. Endler, PUC-Rio, 2008]

- “Conhecendo mais sobre o que está ‘por baixo’ do programa, você pode escrever programas mais eficientes e confiáveis”
- Abstrações em um sistema de computação:



- A linguagem de montagem (Assembly) é um mapeamento direto da linguagem de máquina, mas que introduz várias “facilidades” (ou “menos dificuldades”) para o programador
 - usa “apelidos” das instruções de máquina, mais fáceis de lembrar do que seu valor hexadecimal exigido pelo processador
 - Ex.: `mov eax, edx`

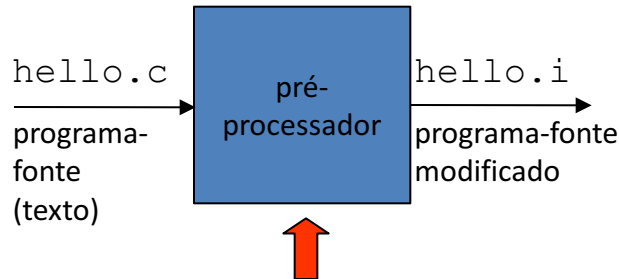
└─ move o que está no registrador de dados para o acumulador



Gerando um executável

- `unix> gcc -o hello hello.c`

```
1. #include <stdio.h>
2. int main()
3. {
4.     printf("hello, world\n");
5. }
```



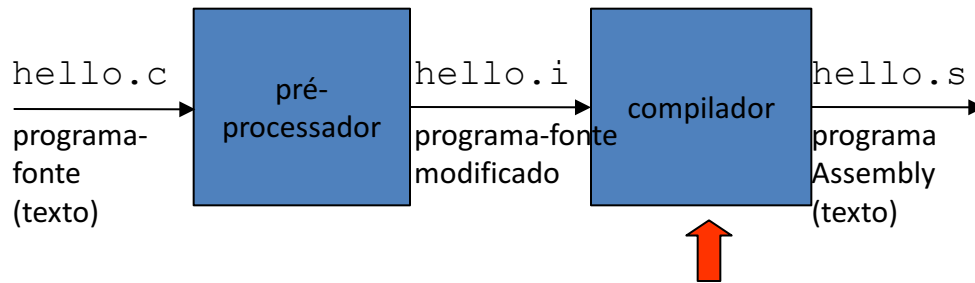
- Modifica o programa em C de acordo com diretivas começadas com #
 - Ex.: `#include <stdio.h>` diz ao pré-processador para ler o arquivo `stdio.h` e inseri-lo no programa fonte
- O resultado é um programa expandido em C, normalmente com extensão `.i`, em Unix



Gerando um executável

- `unix> gcc -o hello hello.c`

```
1. #include <stdio.h>
2. int main()
3. {
4.     printf("hello, world\n");
5. }
```



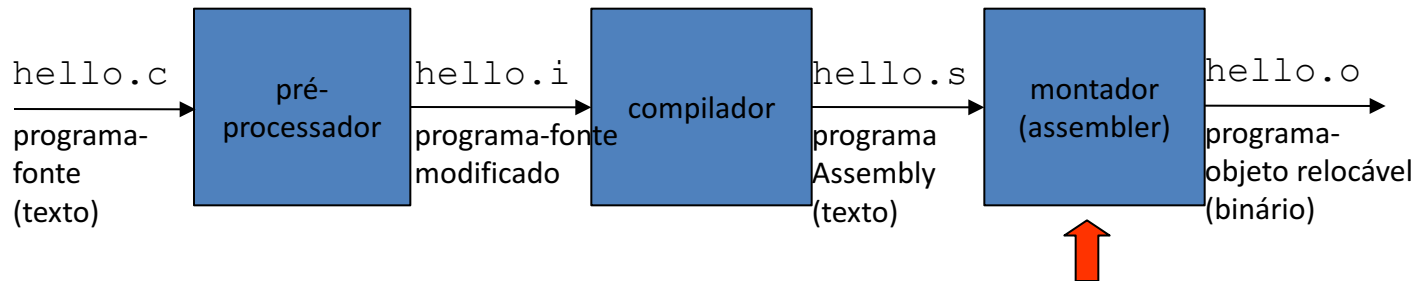
- Compilador traduz o programa `.i` em um programa em Assembly
 - É o formato de saída comum para os compiladores nas várias linguagens de programação de alto nível
 - i.e., programas em C, Java, Fortran, etc vão ser traduzidos para a mesma linguagem Assembly



Gerando um executável

- `unix> gcc -o hello hello.c`

```
1. #include <stdio.h>
2. int main()
3. {
4.     printf("hello, world\n");
5. }
```



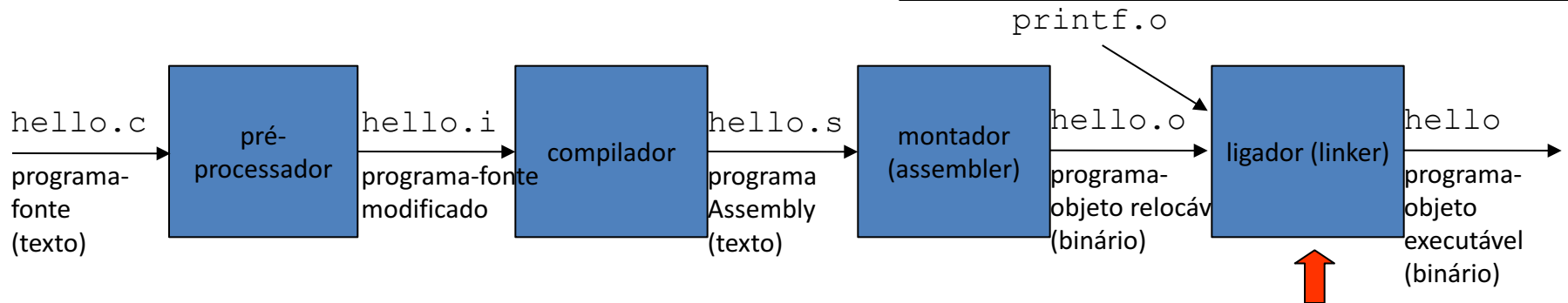
- Montador (Assembler) transforma o programa em Assembly em um programa binário em linguagem de máquina (chamado programa-objeto)
 - Os módulos de programas, compilados ou montados, são armazenados em um formato intermediário (“*Programa-Objeto Relocável*” - extensão .o)
- Endereços de acesso e a posição do programa na memória ficam **indefinidos**



Gerando um executável

- `unix> gcc -o hello hello.c`

```
1. #include <stdio.h>
2. int main()
3. {
4.     printf("hello, world\n");
5. }
```



- O ligador (linker) gera o programa executável a partir do .o gerado pelo assembler
 - No entanto, pode haver funções-padrão da linguagem (ex., `printf`) que não estão definidas no programa, mas em outro arquivo .o pré-compilado (`printf.o`)
 - O ligador faz a junção dos programas-objeto necessários para gerar o executável

Execução



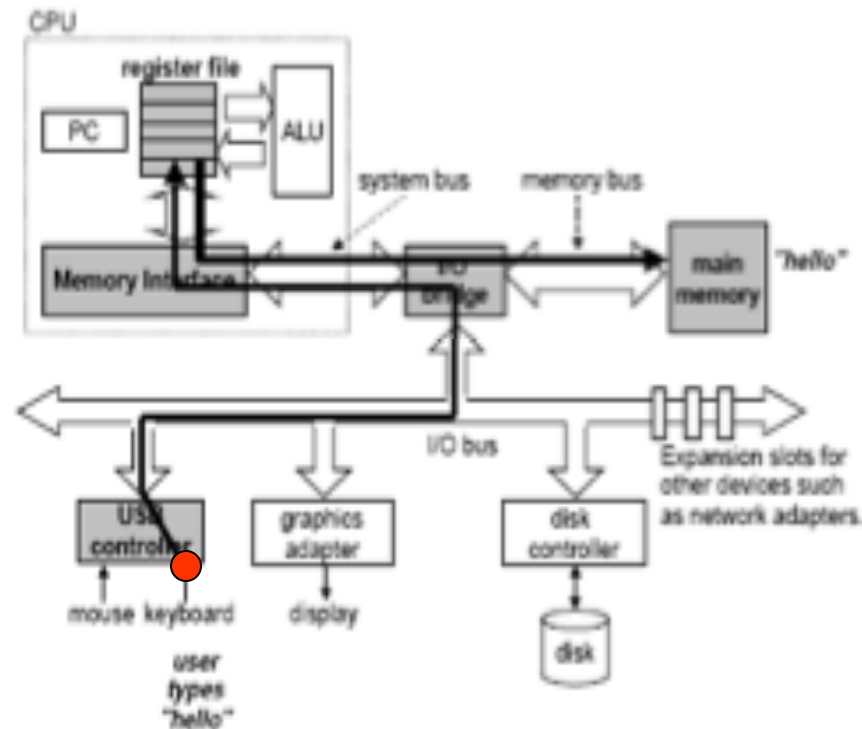
Como acontece...



Processo

Conceito: Um programa em execução

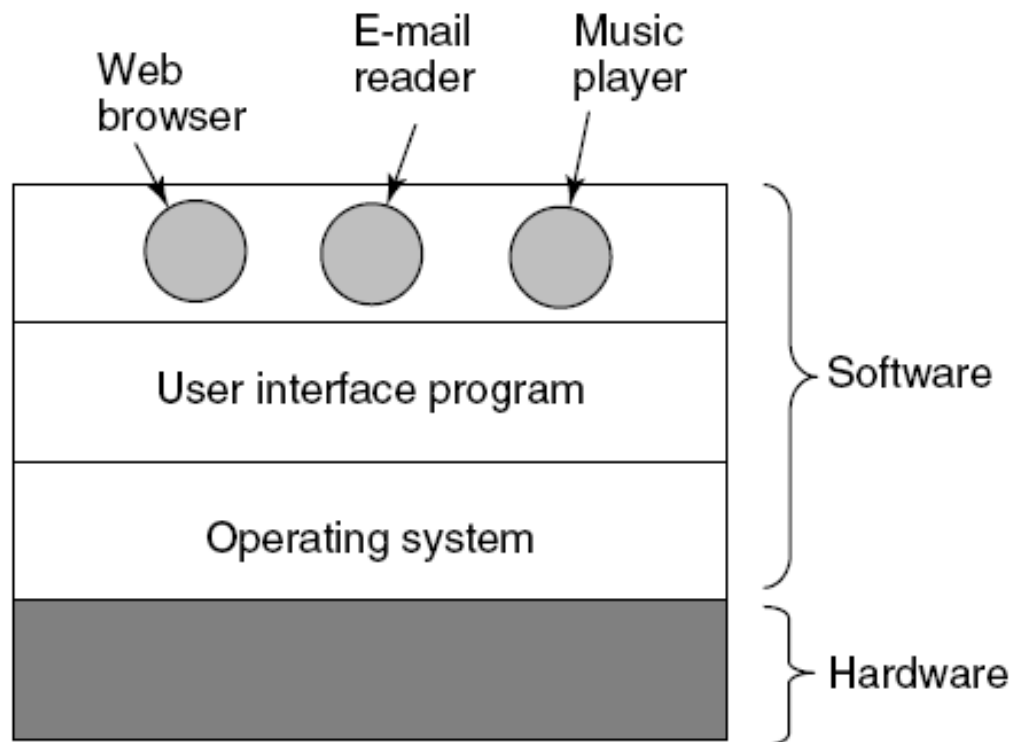
Ao digitar “hello”, os caracteres são passados para um registrador e depois para memória principal





Mais de um programa em execução

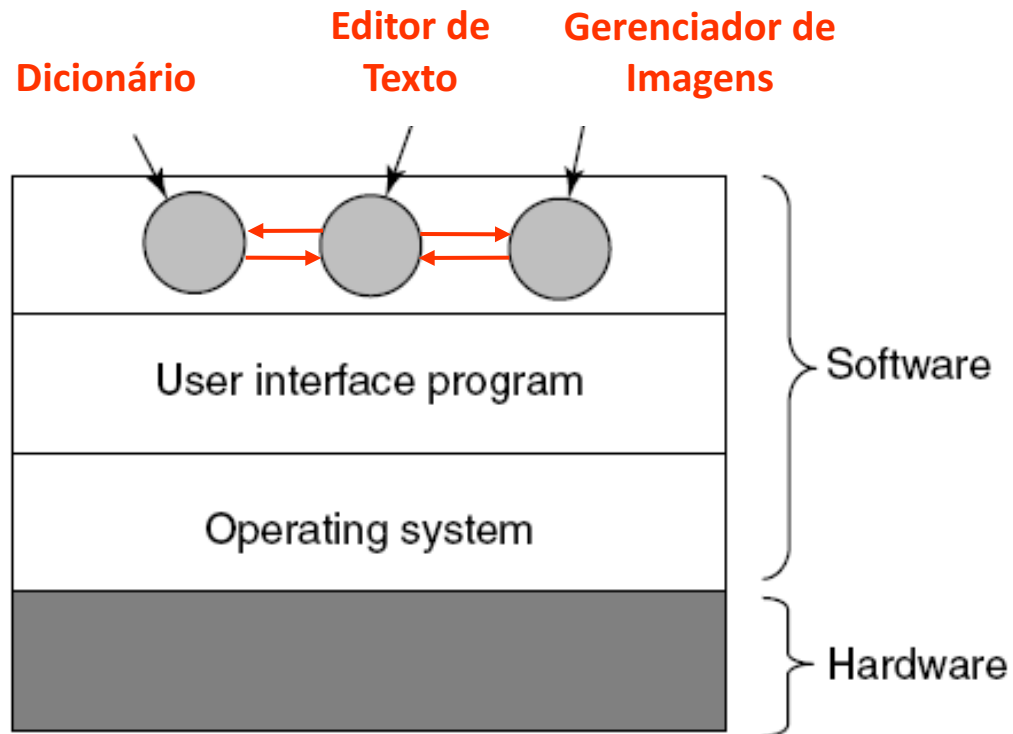
- Múltiplos processos *vs.* um (ou [poucos] mais) processador(es) \Rightarrow **como pode???**





Processos Comunicantes

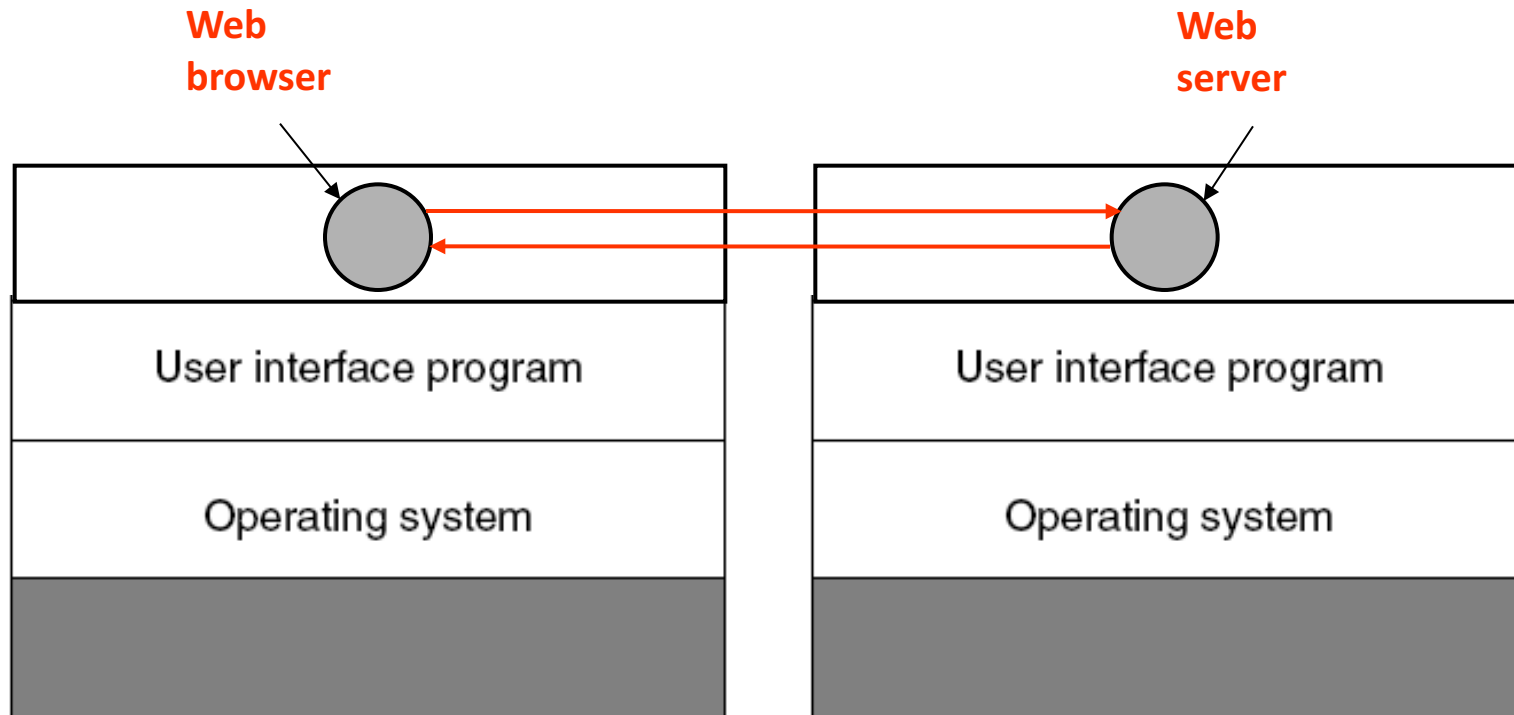
- Como pode???





Sistemas Distribuídos

- Processos em máquinas distintas e que se comunicam





Sistemas Distribuídos

Como fazer funcionar aplicações distribuídas que usam diferentes sistemas de computador (hardware), sistemas operacionais e software de aplicação (ex. linguagens de programação), interconectadas por diferentes redes?

heterogeneidade

O problema da **interoperabilidade**

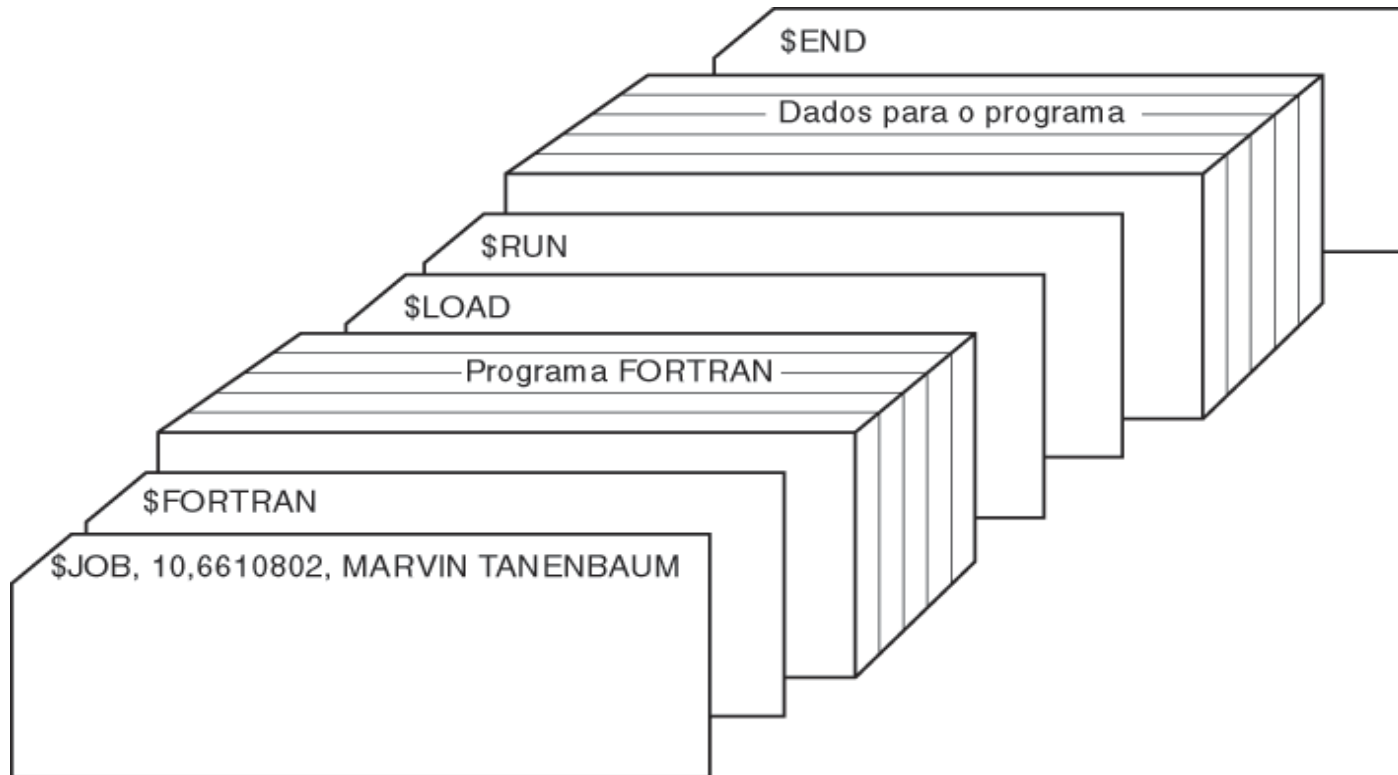


História dos Sistemas Operacionais

- Primeira geração: 1945 - 1955
 - Válvulas, painéis de programação
- Segunda geração: 1955 - 1965
 - transistores, **sistemas em lote**
- Terceira geração: 1965 – 1980
 - CIs (circuitos integrados) e **multiprogramação**
- Quarta geração: 1980 – presente
 - Computadores pessoais
- Hoje: onipresença – computação ubíqua



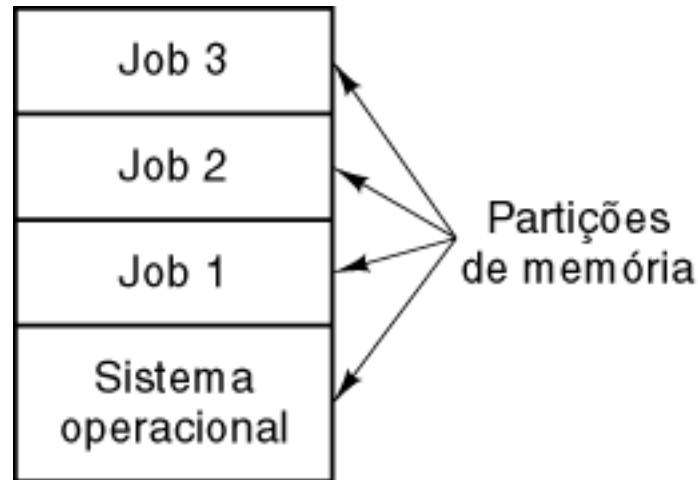
História dos Sistemas Operacionais



- Estrutura de um job típico (lote de cartões) – 2a. geração



História dos Sistemas Operacionais



- Sistema de **multiprogramação**
 - Três jobs na memória – **3a. geração**



Diversidade de Sistemas Operacionais

- Sistemas operacionais de **computadores de grande porte** (*mainframe*)
- Sistemas operacionais de servidores / **redes**
- Sistemas operacionais de **multiprocessadores** (paralelismo)
- Sistemas operacionais de computadores pessoais
- Sistemas operacionais de dispositivos portáteis/ **móveis** (ex. celulares)
- Sistemas operacionais de **tempo-real**
- Sistemas operacionais **embarcados**
- Sistemas operacionais de cartões inteligentes
- Sistemas operacionais de sensores



Supercomputação

O Caso do BAM (Brazilian Global Atmospheric Model)

- Pronto para rodar com melhor resolução espacial, de 10 km, mas...
 - Uma das limitações para rodar o modelo com essa resolução é a falta de capacidade computacional
- Mesmo utilizando toda a capacidade de processamento do supercomputador Tupã (30 mil processadores) ao longo de duas horas, foi possível gerar previsões para apenas 24 horas
 - Com um computador novo 28 vezes mais potente, por exemplo, demoraria aproximadamente uma hora para fazer previsões com até sete dias de antecedência
- O máximo que o Tupã consegue hoje é rodar o BAM com resolução espacial de 20 km
- Quanto melhor (menor) a resolução espacial do modelo, melhor é a capacidade de representar a topografia, como vales e montanhas, e o contraste entre mar e continente
 - Com resolução espacial de 180 km, estes contrastes não são bem definidos, e a praia pode parecer terra
 - EUA: resolução espacial de 13 km, usando supercomputadores entre 30 e 50 vezes mais velozes que o Tupã – na ordem de PetaFlops ou 10^{15} operações de ponto flutuante por segundo



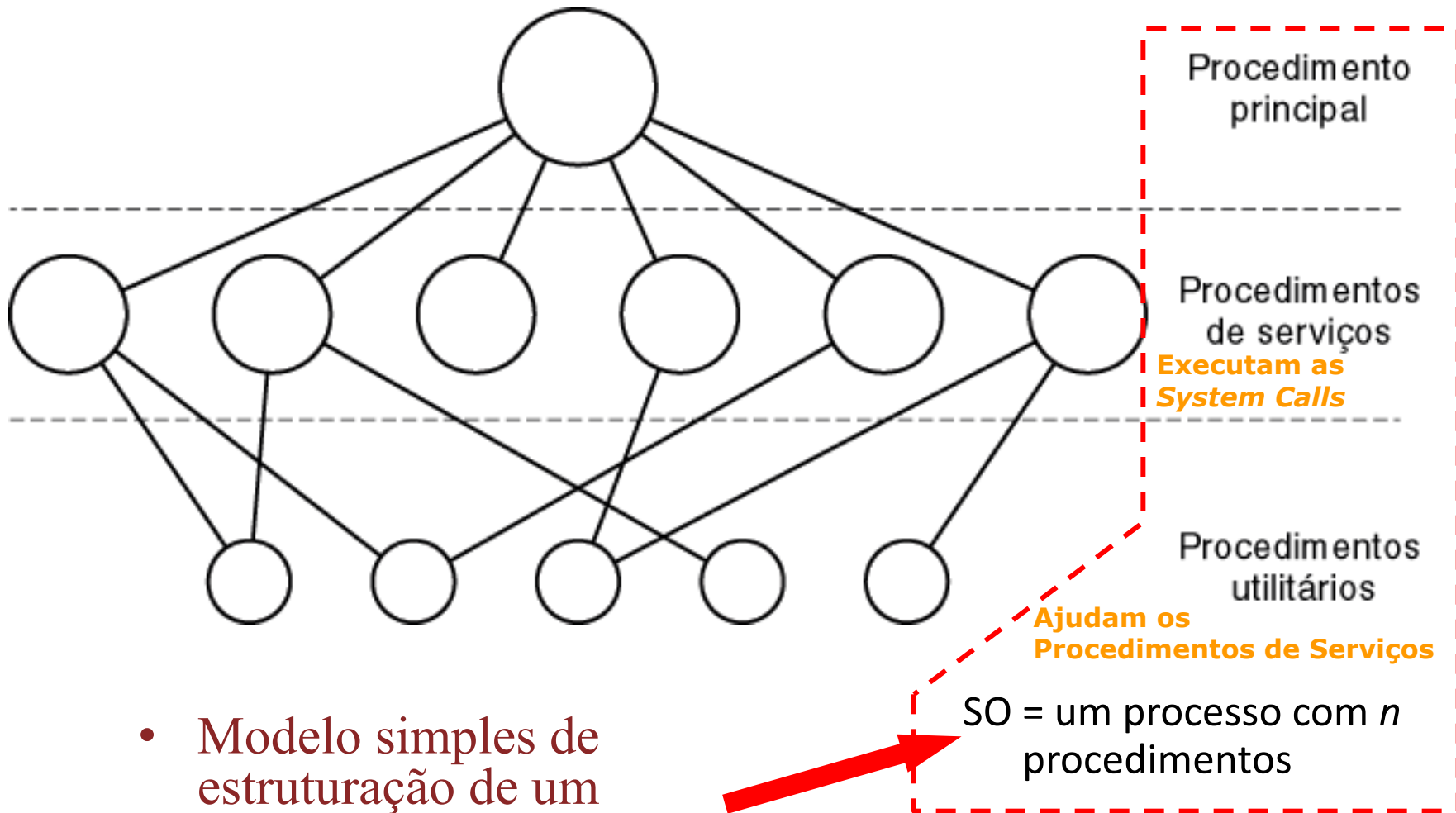
Estruturação de Sistemas Operacionais

- Monolítico
- Camadas
- Cliente-Servidor
- Virtualização



Estrutura de Sistemas

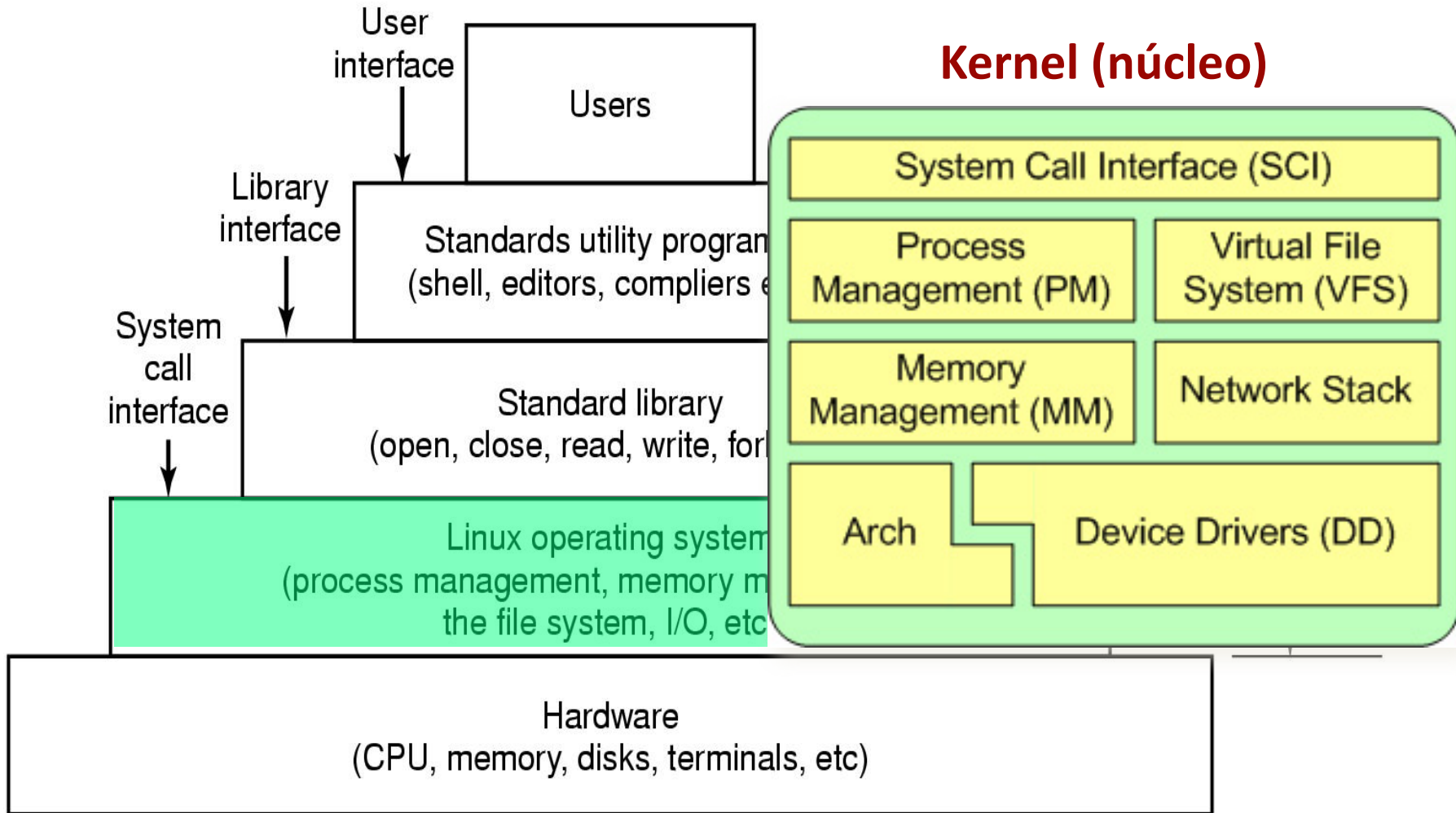
Operacionais: Sistema Monolítico



- Modelo simples de estruturação de um sistema **monolítico**

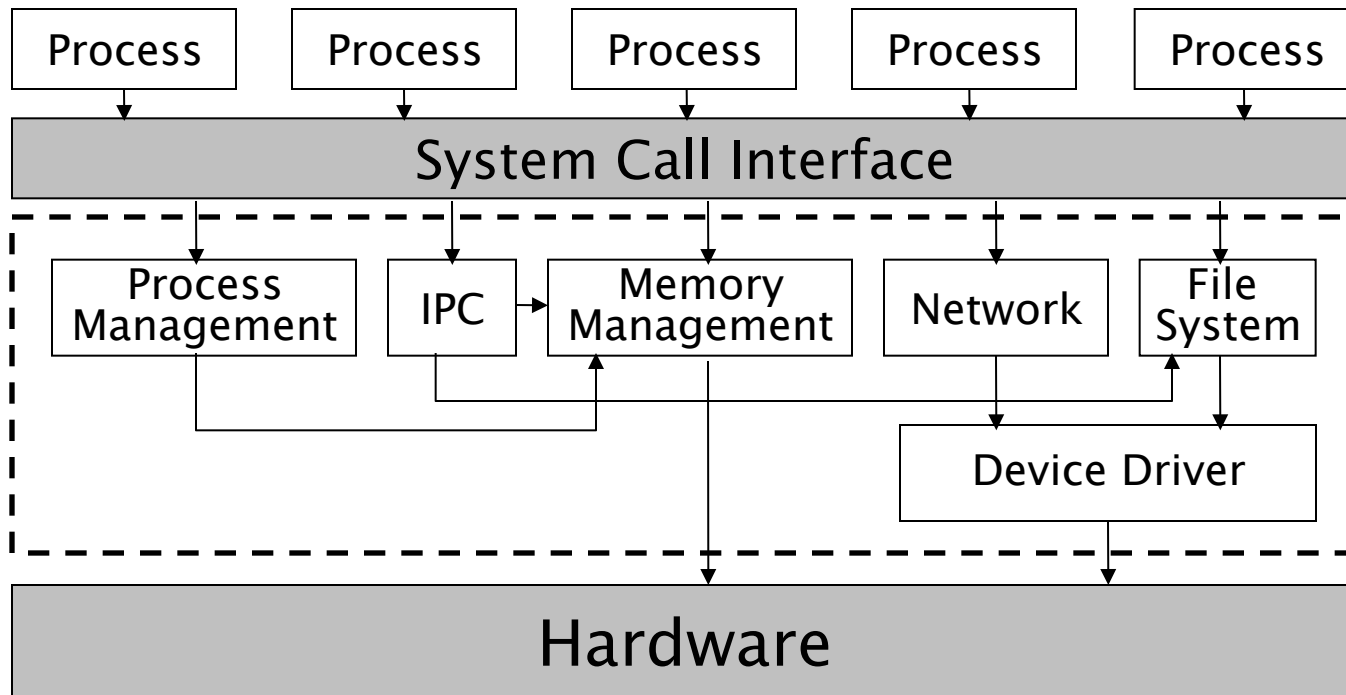


Camadas em Linux





Linux Kernel: Relacionamentos



APPLICATIONS

Home

Contacts

Phone

Browser

...

APPLICATION FRAMEWORK

Activity Manager

Window Manager

Content

View

Notification Manager

Package Manager

Telephony Manager

GTalk Service

LIBRARIES

Surface Manager

Media Framework

OpenGL |

ANDROID

SGL

SSL

ANDROID RUNTIME

LINUX KERNEL

Display Driver

Camera Driver

Bluetooth Driver

Flash Memory Driver

Binder (IPC) Driver

USB Driver

Keypad Driver

WiFi Driver

Audio Drivers

Power Management

Que estratégias você imagina para economizar energia?

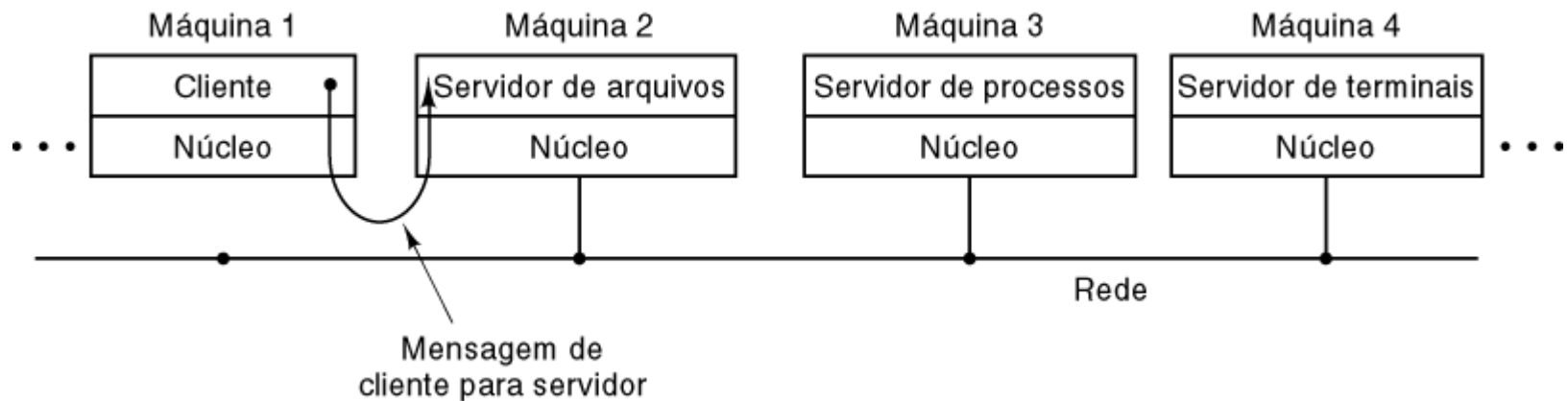


Estruturação de Sistemas Operacionais

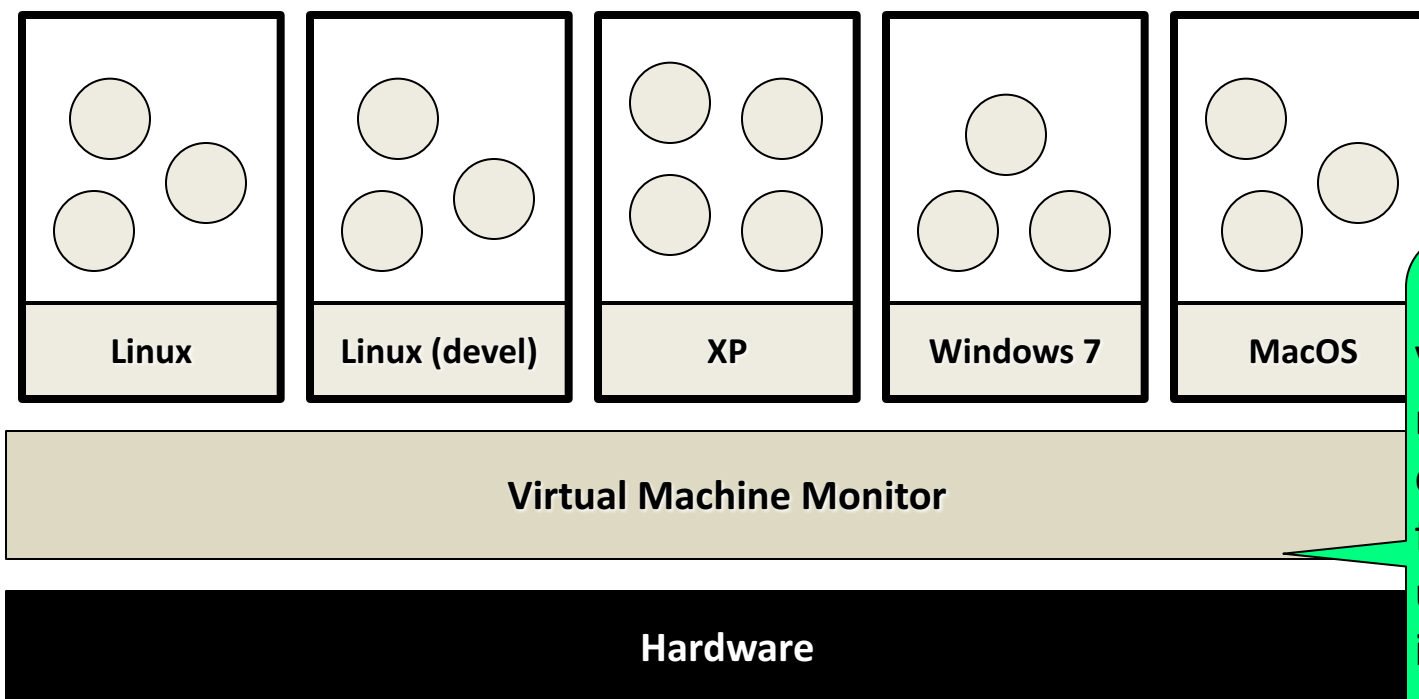
- ✓ Monolítico
- ✓ Camadas
 - Cliente-Servidor
 - Virtualização

Estrutura de Sistemas Operacionais: Cliente-Servidor

- O modelo **cliente-servidor** em um **sistema (operacional) distribuído**



Estrutura de Sistemas Operacionais: Máquina Virtual (Virtualização)

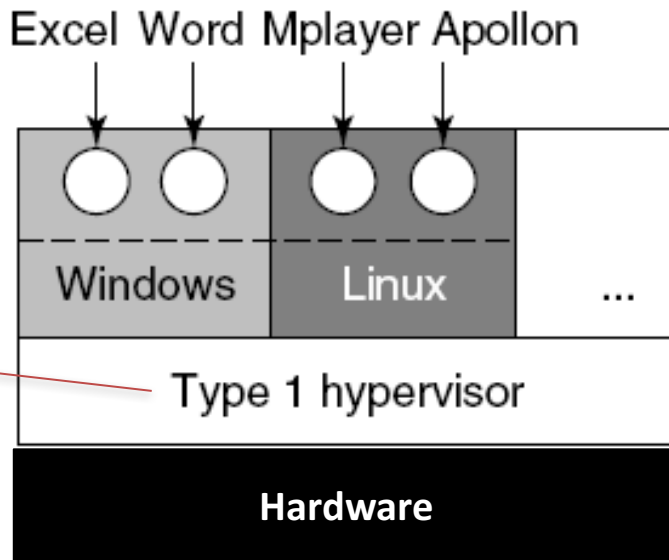


VMM opera na interface de hardware, fornecendo uma interface idêntica para os SOs acima



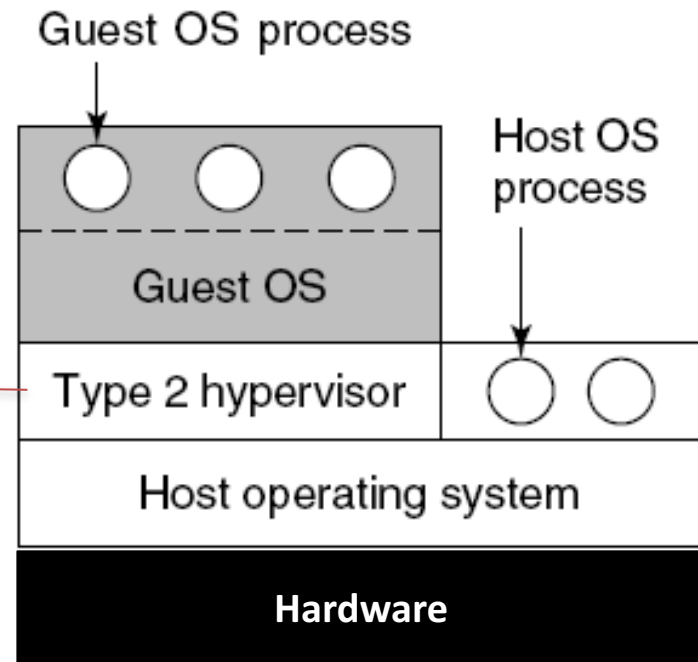
Virtual Machines: Tipos (Arquiteturas)

native, bare-metal hypervisor
Ex. Xen



Hipervisor Tipo 1

hosted hypervisor
Ex. VirtualBox



Hipervisor Tipo 2

Exemplo de virtualização na Computação em Nuvem

