Organização de Computadores e Sistemas Operacionais

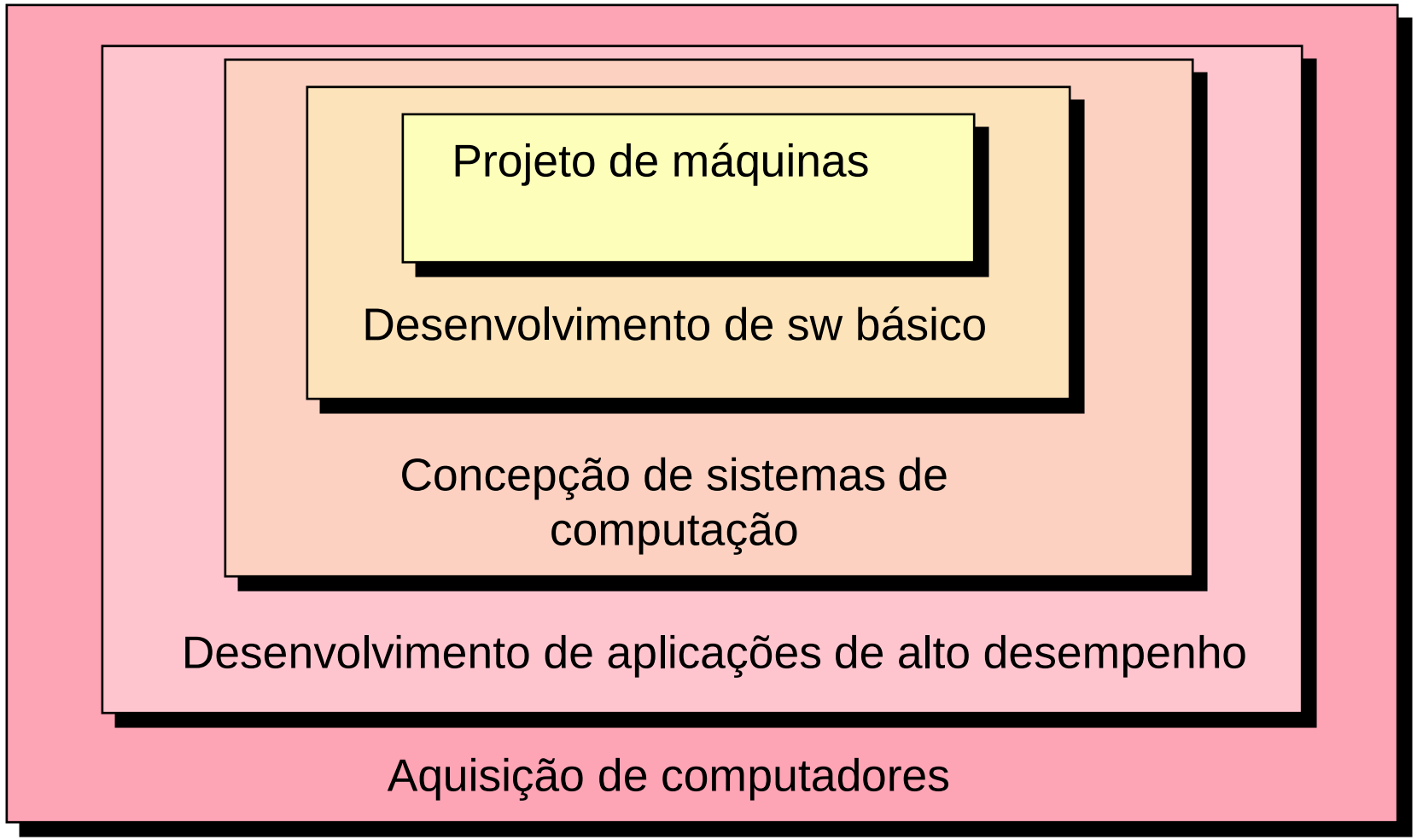
Sérgio Cavalcante
svc@cin.ufpe.br: Usem assunto com [ocso]
http://www.cin.ufpe.br/~svc/ocso

98835.0950

Motivação

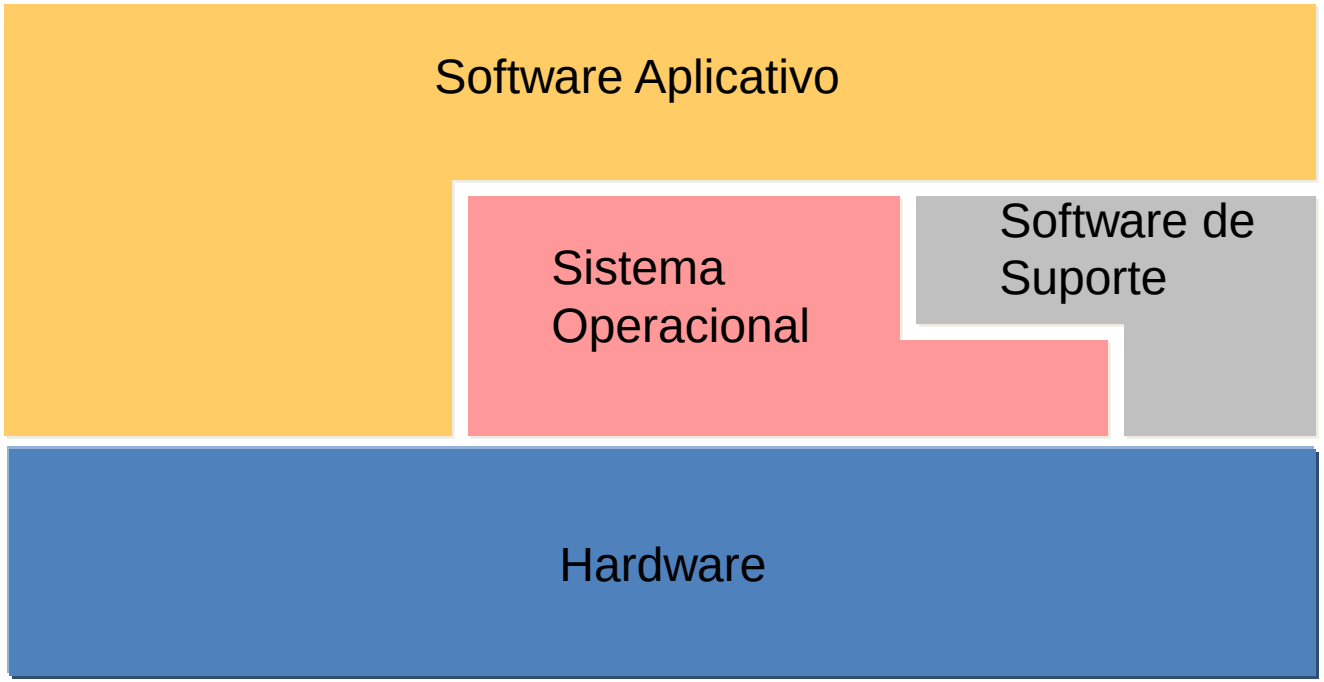
- O que é este curso ?
- Porque é importante saber conceitos de arquitetura/organização de computadores e sistemas operacionais?

Público alvo



Concreto	**Abstrato**
Tangível	**Intangível**
Hardware	**Software**

Hardware e Software



Arquitetura X Organização

Arquitetura

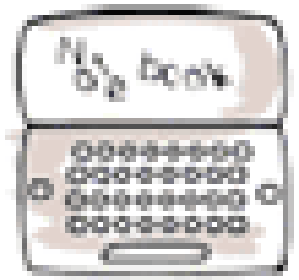
- Repertório de instruções
- Tipos de Dados
- Modos de endereçamento
- Conjunto de registradores

Organização

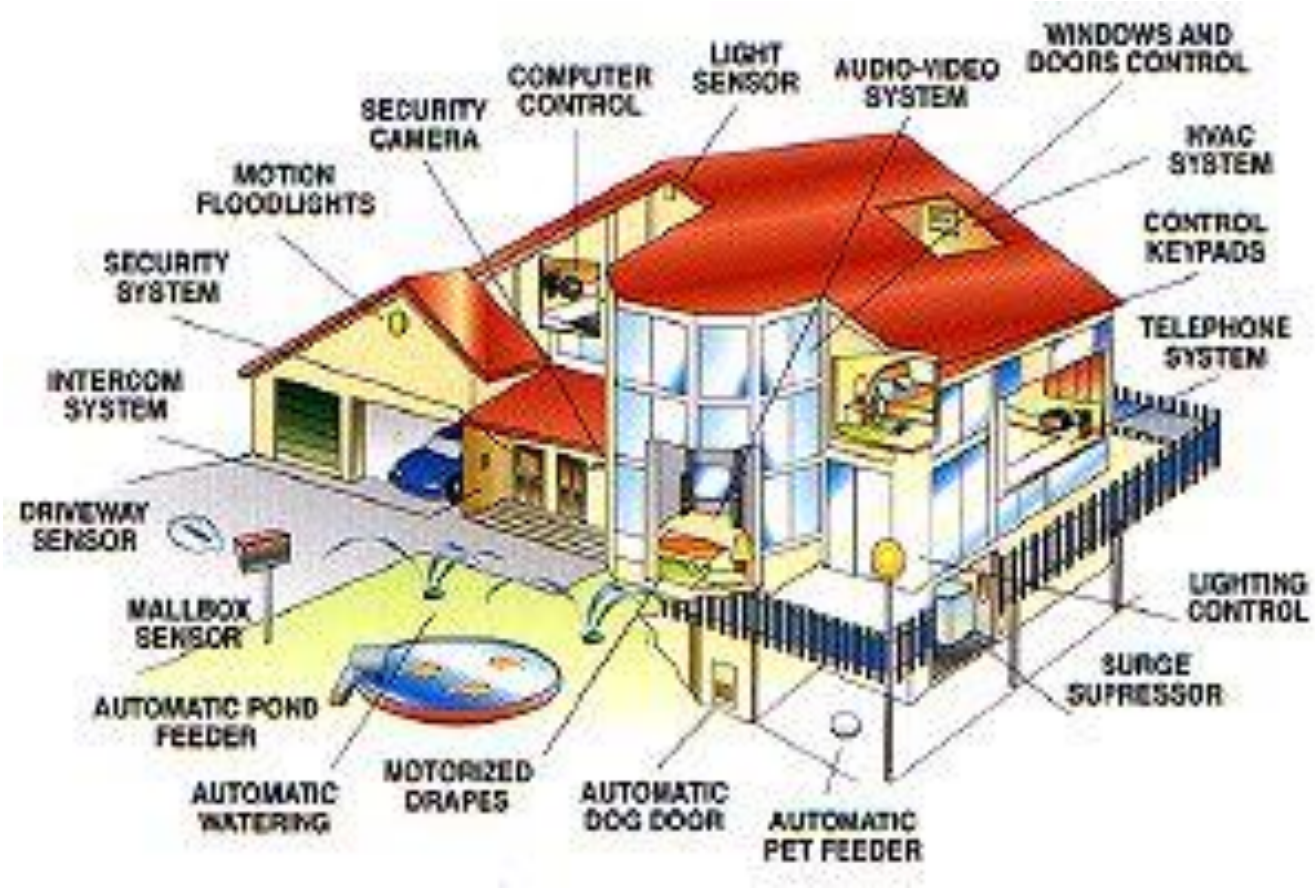
- Tecnologia de memória
- Interfaces
- Implementação das instruções
- interconexões

Organizações Diferentes

Computadores estão presentes nos mais diversos equipamentos

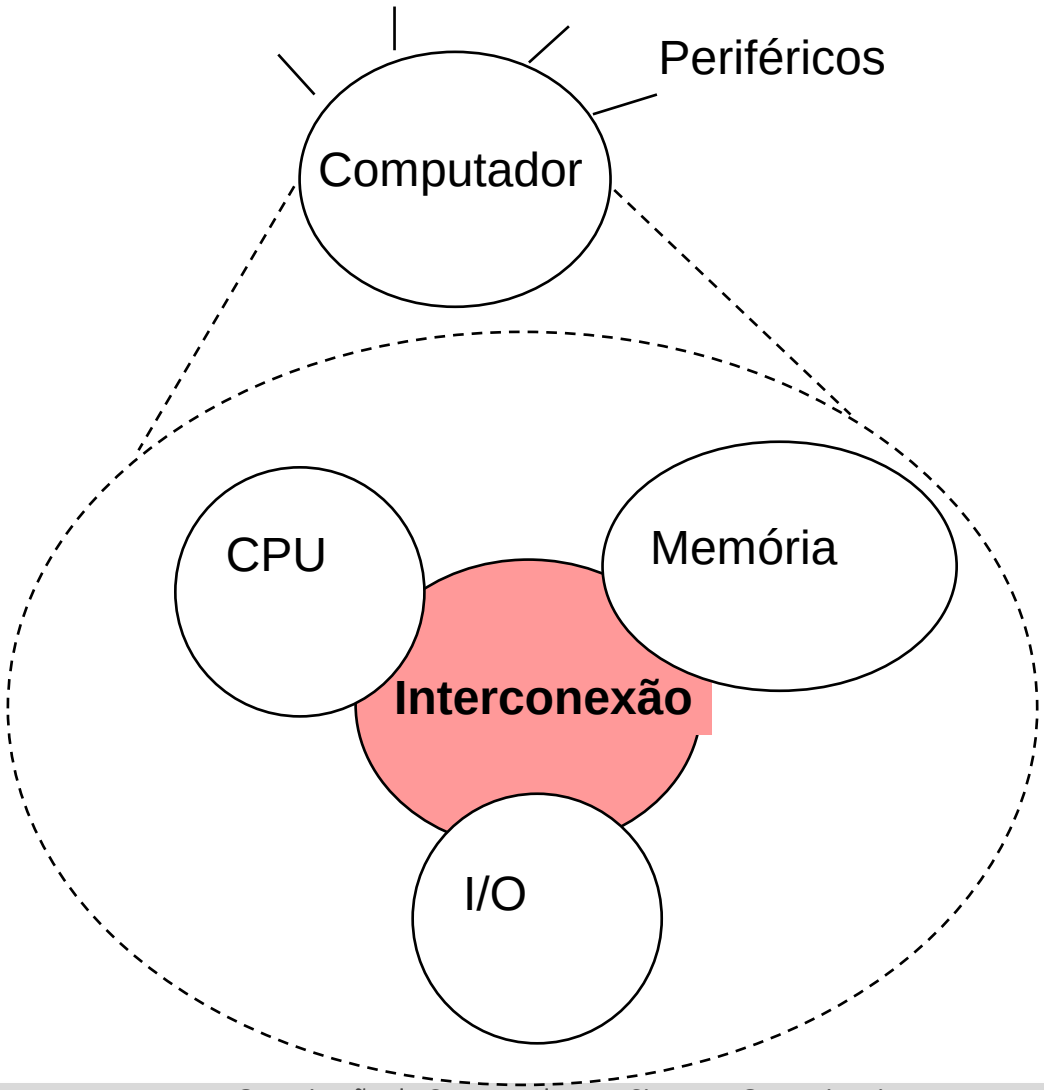


Organizações Diferentes

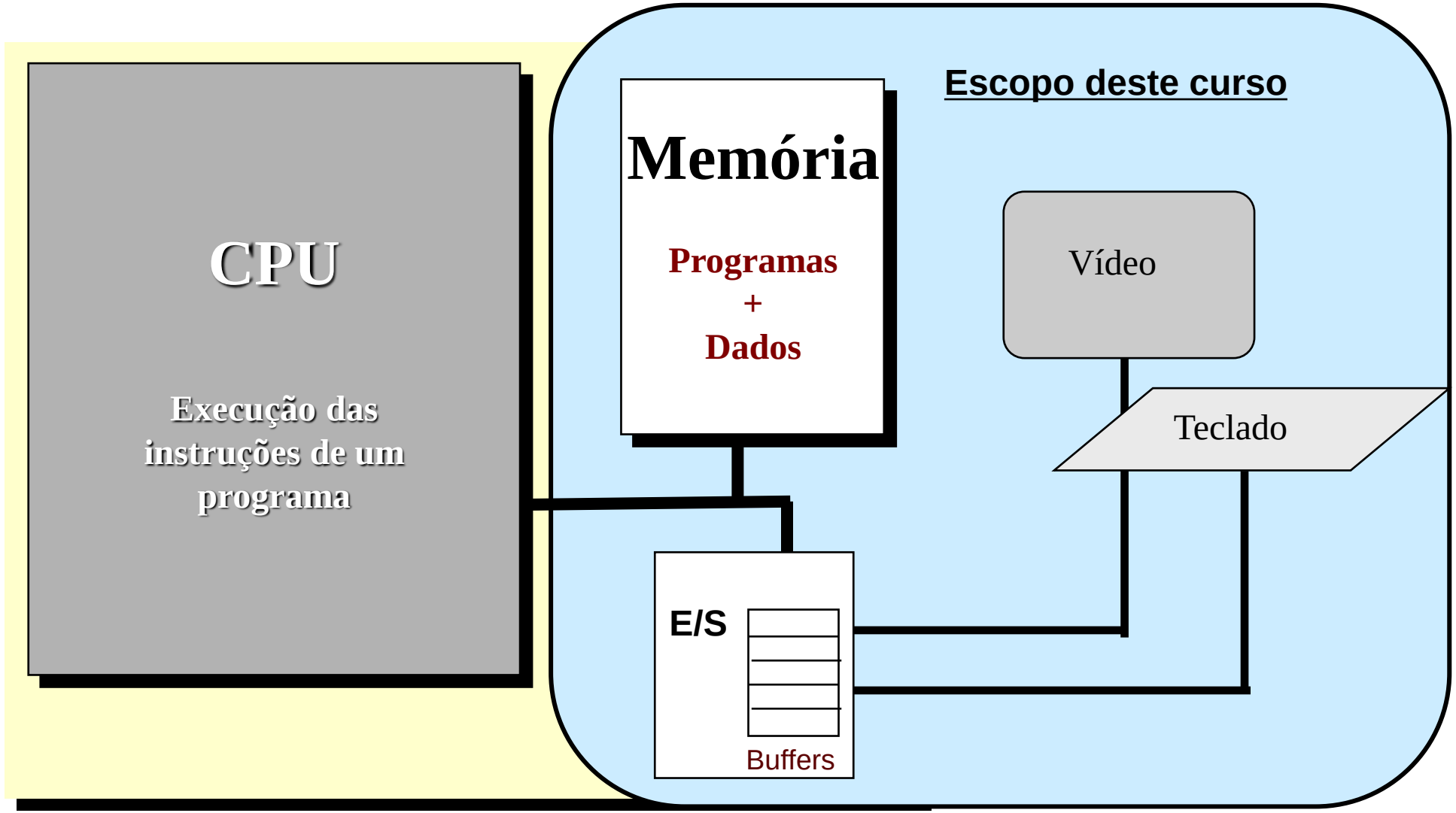


Computadores no nosso dia a dia ....

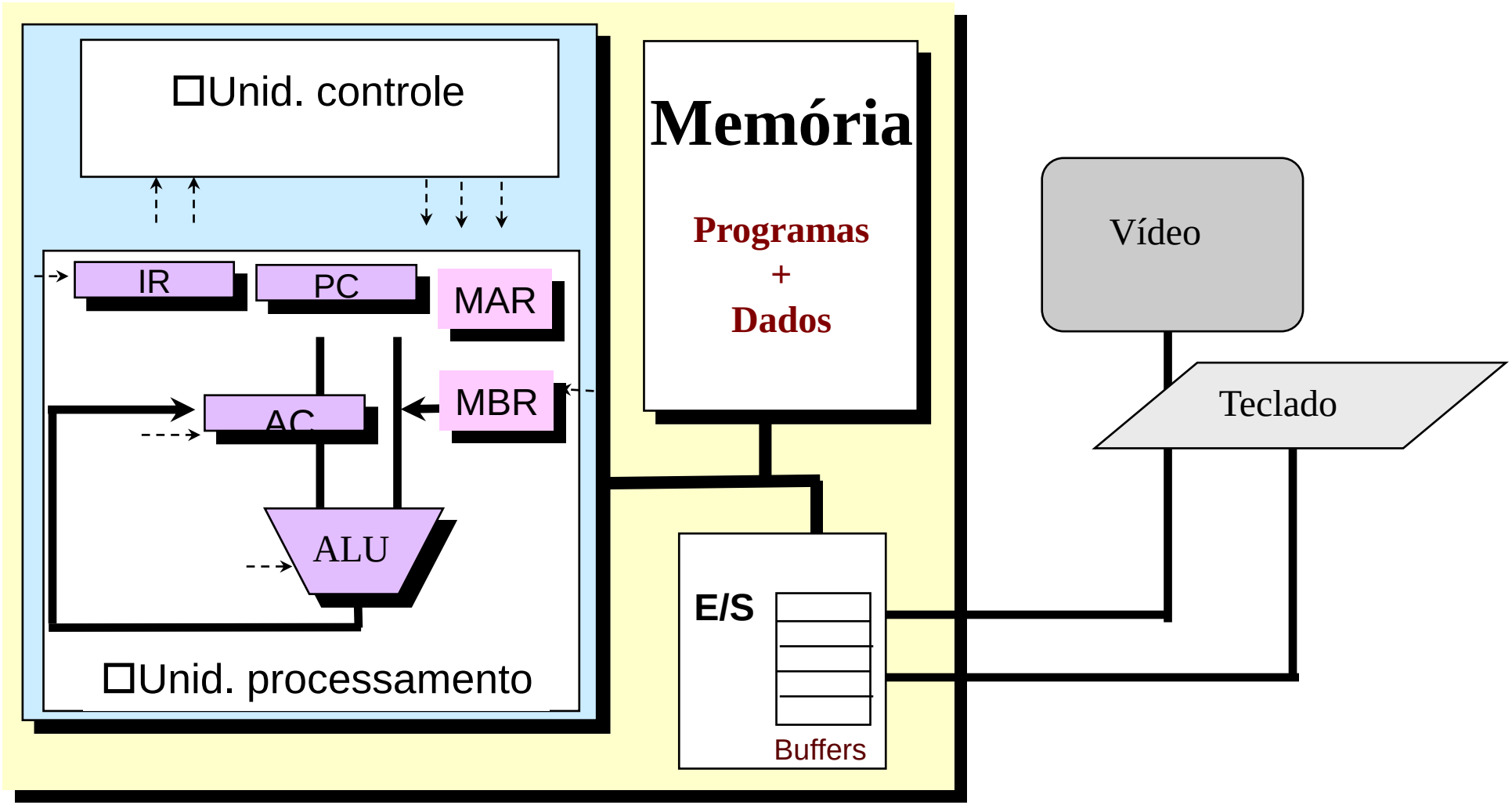
Estrutura de um computador



Componentes de um computador

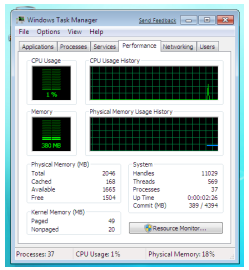


Componentes de um computador



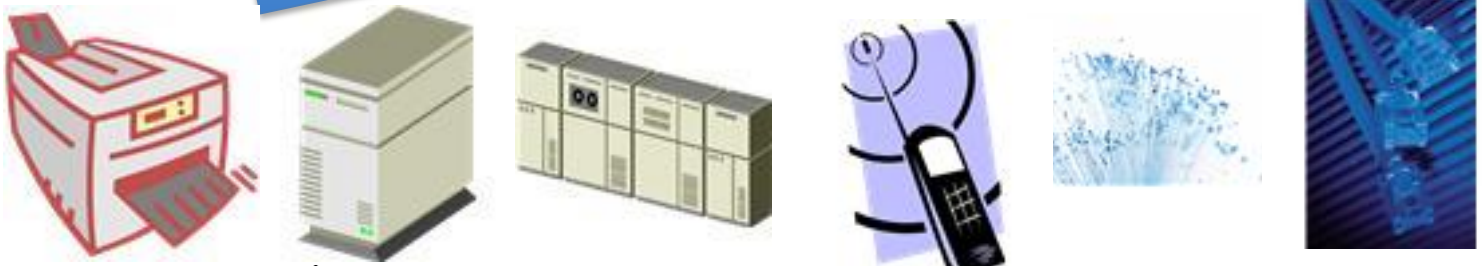
Mundo real

Pessoas e aplicativos

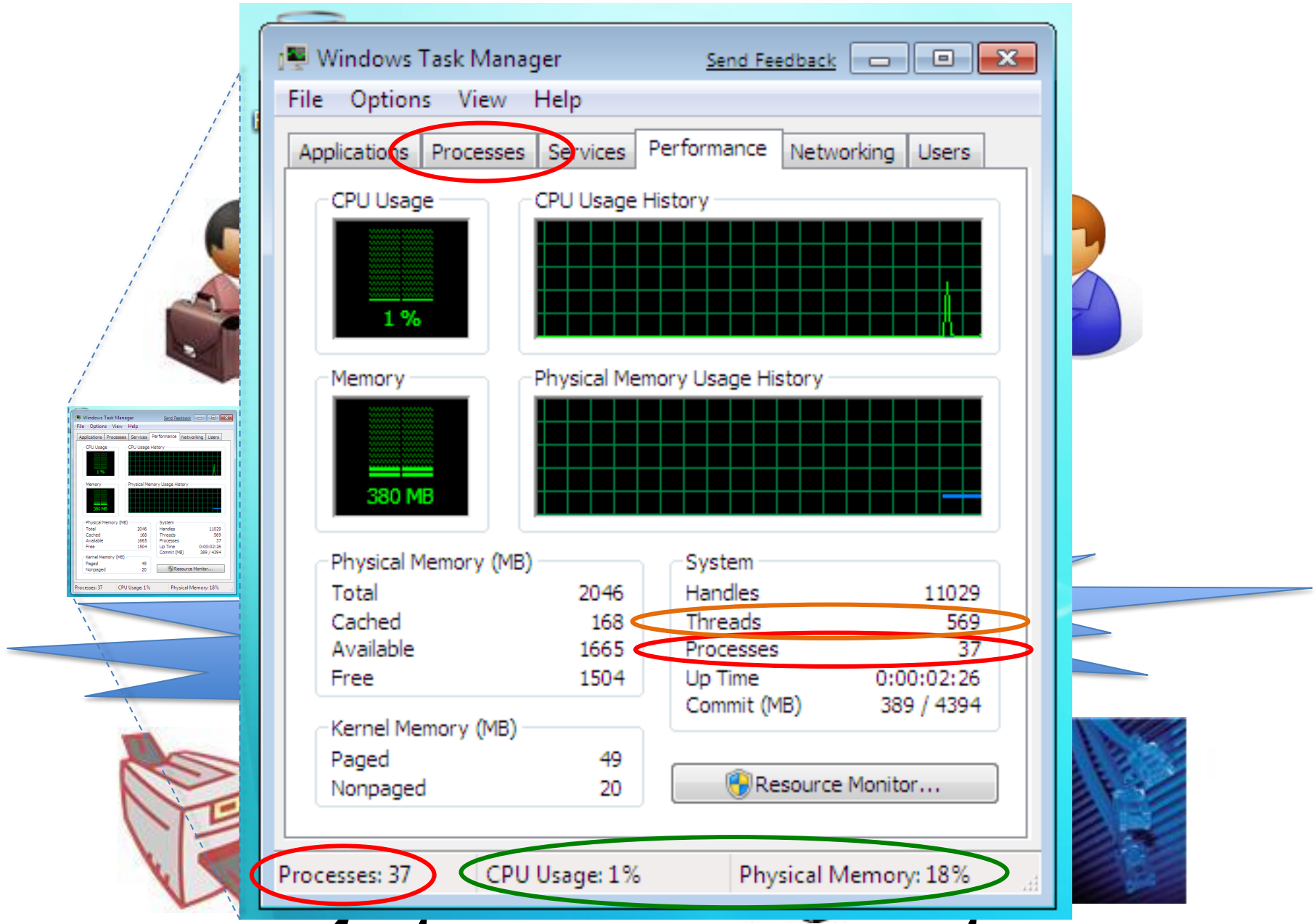


sistemas operacionais, middleware

uso, gerenciamento, compartilhamento
Complexidade



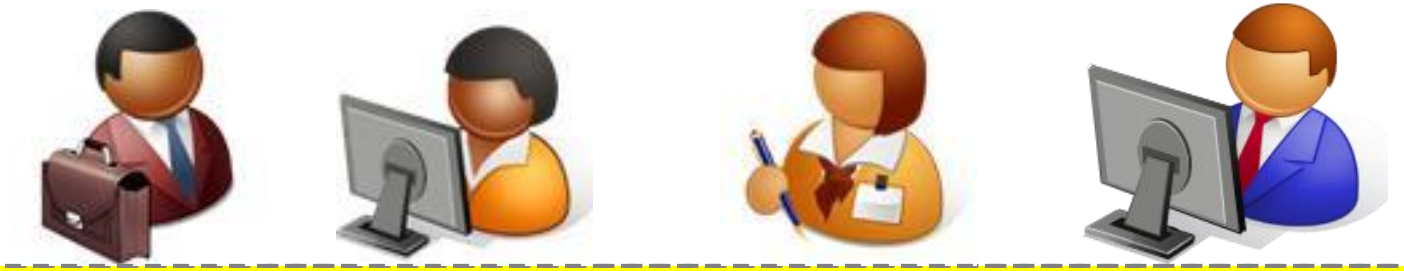
hardware *redes*


Windows Task Manager
Send Feedback
File Options View Help
Applications Processes Services Performance Networking Users
CPU Usage
1 %
CPU Usage History
Memory
380 MB
Physical Memory Usage History
Physical Memory (MB)
Total 2046
Cached 168
Available 1665
Free 1504
System
Handles 11029
Threads 569
Processes 37
Up Time 0:00:02:26
Commit (MB) 389 / 4394
Kernel Memory (MB)
Paged 49
Nonpaged 20
Resource Monitor...
Processes: 37
CPU Usage: 1%
Physical Memory: 18%

Infra-estruturas de Suporte

a

Usuários / Programas de Usuários

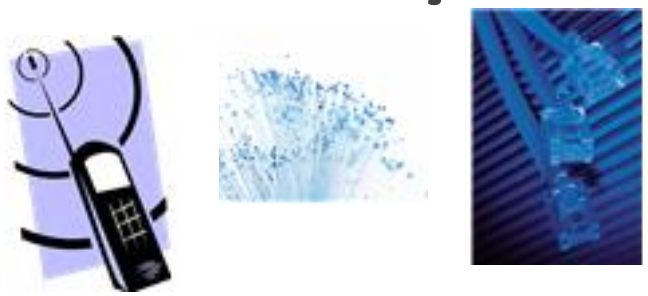


Infra-estrutura de Software

sistemas operacionais, middleware

Infra-estrutura de Hardware

Infra-estrutura de Comunicação



Computadores de grande porte, desktops, tablets, celulares, TV etc.

Sistemas operacionais visam gerenciar a operação de computadores de modo a oferecer flexibilidade, eficiência, segurança, transparência e compartilhamento de recursos

Quatro grupos básicos: processos, memória, armazenamento (arquivos), entrada e saída

Pra quê software básico?

- O que acontece quando ligamos o computador?
- E quando “clicamos” num ícone?
- Como funcionam dois programas ao mesmo tempo?
- Como ocorre o mapeamento de discos?
- E se dois programas quiserem usar o mesmo recurso?

- Existe aqui um programa (PowerPoint) rodando,
  - usando o processador da máquina,
  - …a memória,
  - …manipulando um arquivo armazenado no disco,
  - …aparecendo na tela,
  - …recebendo comandos, via teclado

Como se faz?

Como isso está acontecendo ao mesmo tempo?

Um Sistema Operacional

- Uma máquina abstrata
  - Esconde detalhes através de uma máquina virtual, mais fácil de usar

- Um gerenciador de recurso
  - Cada programa tem um **tempo com o recurso**
    - **Ex.: CPU, Impressora,**
  - Cada programa tem um **espaço no recurso**
    - **Ex.: Memória**

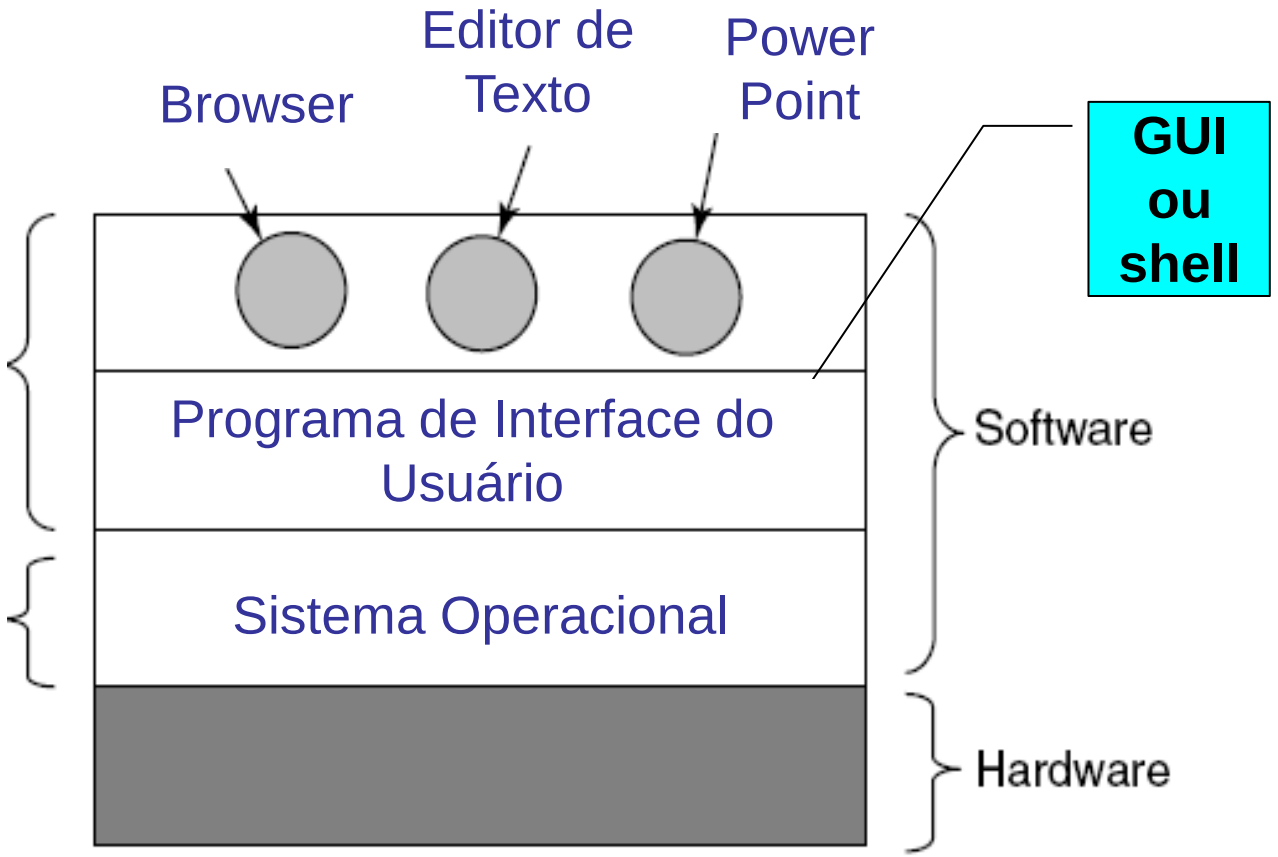
Sistema Computacional em Camadas

Não pode executar instruções que afetam o controle da máquina ou fazem E/S

Modo Usuário

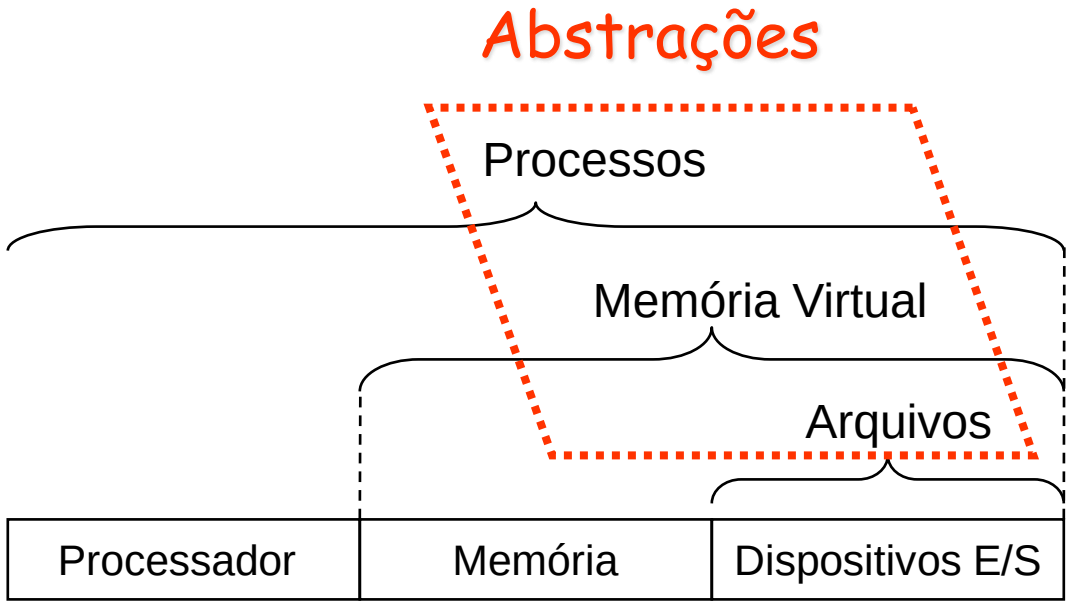
Modo Kernel

Acesso completo a todo o hardware e pode executar qualquer instrução que a máquina seja capaz de executar

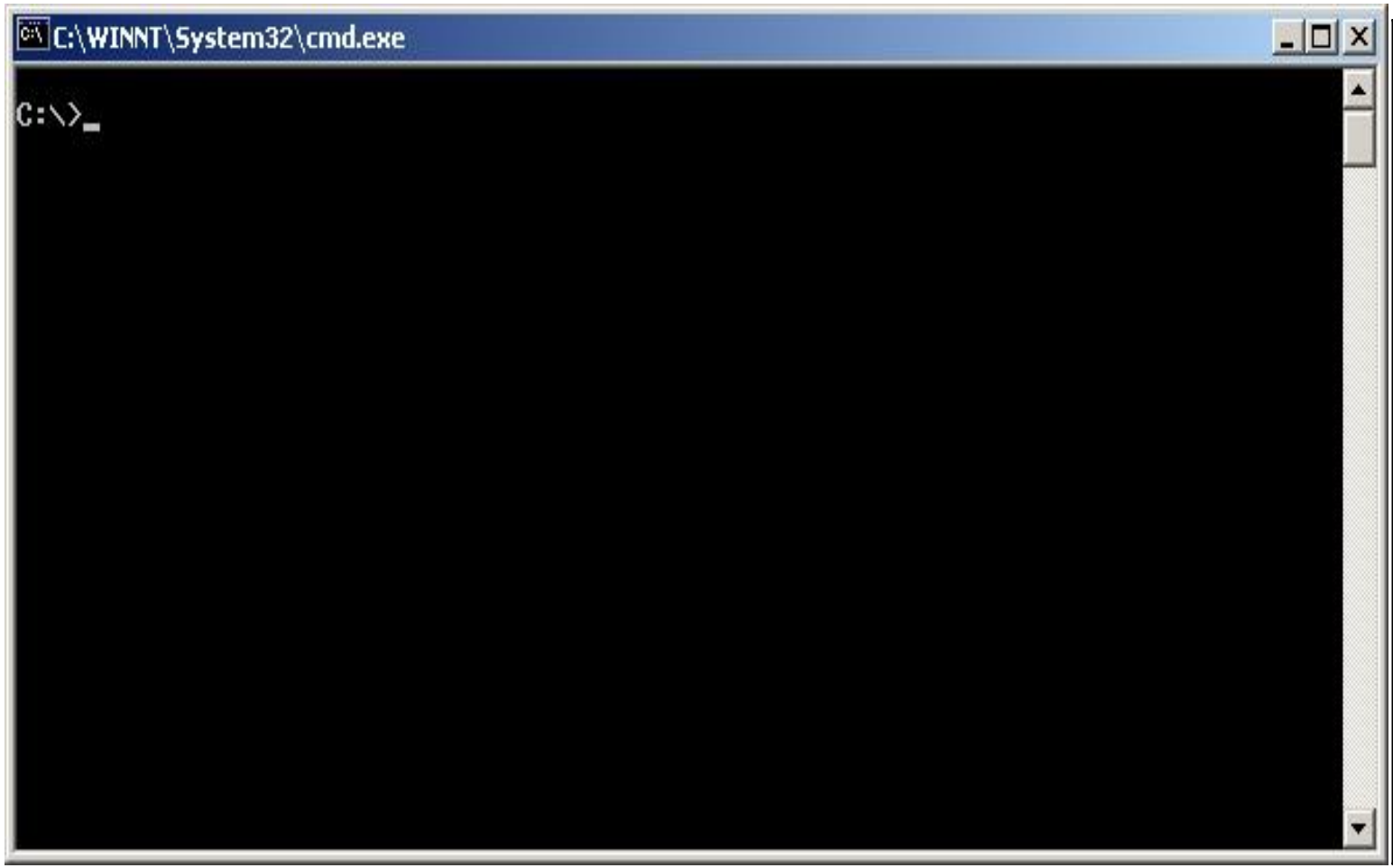


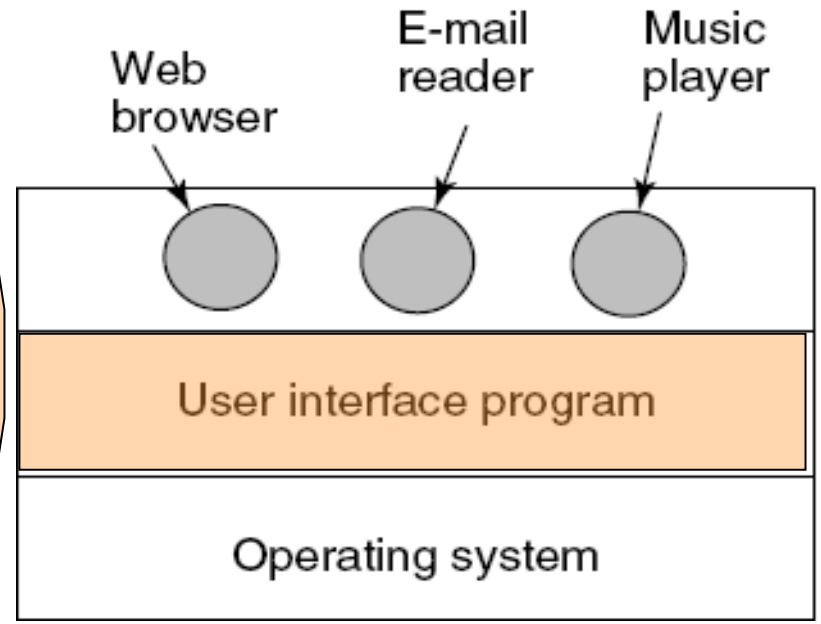
O S.O. como uma Máquina Estendida

Sistemas operacionais fazem com que o hardware, que tem interfaces difíceis, se torne mais acessível por meio de abstrações mais fáceis e simples.

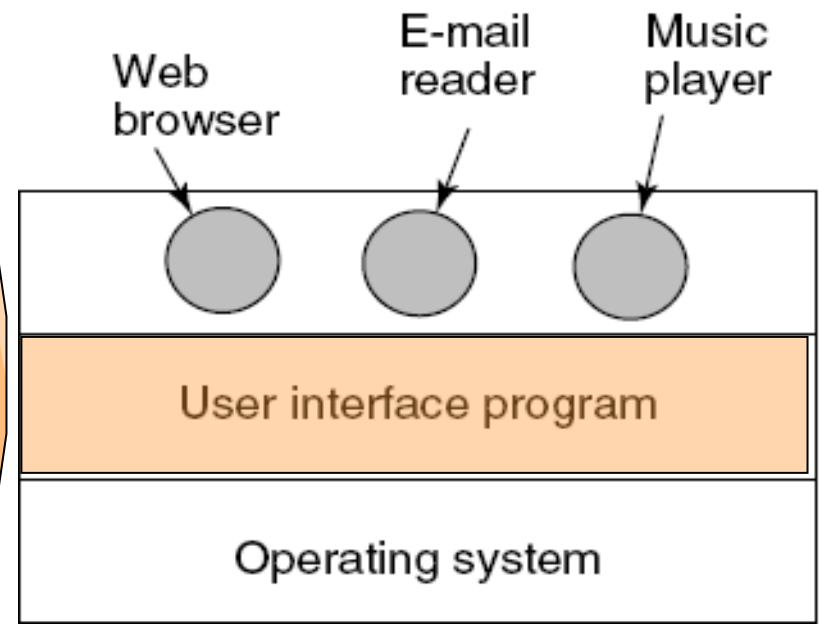


SO: Interface de Usuário
Shell





SO: Interface de Usuário
GUI – *Graphical User Interface*



O S.O. como Gerenciador de Recursos

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

Um Sistema Operacional...

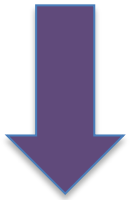
- [é um conjunto de programas que] **gerencia os recursos disponíveis**
  - processo/processador
  - memória
  - arquivos/disco
  - dispositivos de entrada/saída – teclado, tela, mouse etc.

Eficiência, compartilhamento e resolução de possíveis conflitos

- Gerência de processo
- Gerência de memória
- Gerência de disco/ armazenamento – Sistema de Arquivos
- Gerência de entrada/saída

Um Sistema Operacional...

- [é um conjunto de programas que] **visa esconder as peculiaridades do hardware**



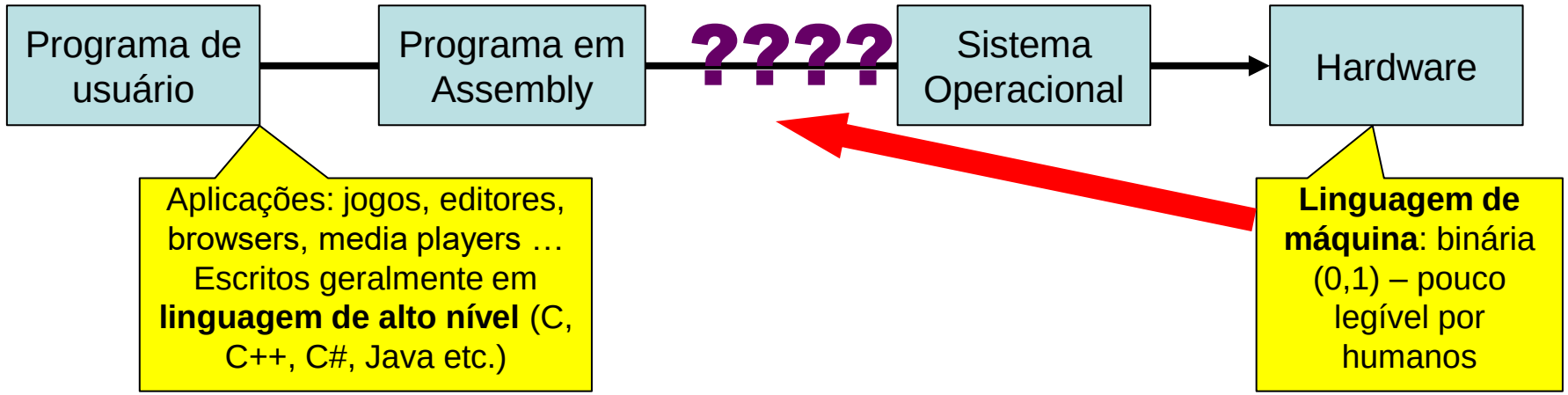
Máquina mais fácil de ser utilizada, mais amigável e mais segura

Software Executável

Software Executável

[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:

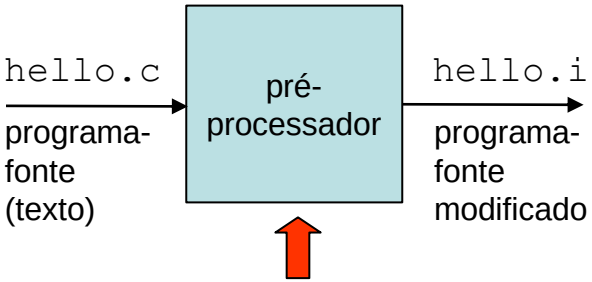


- A linguagem de montagem (**Assembly**) é um mapeamento direto da linguagem de máquina, mas que introduz várias "facilidades" (ou "menos dificuldades") para o programador
  - usa "apelidos" das instruções de máquina, mais fáceis de lembrar do que seu valor hexadecimal exigido pelo processador
    - Ex.: mov AX, DX
      - move o que está no registrador DX para o registrador AX

Gerando um executável

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

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

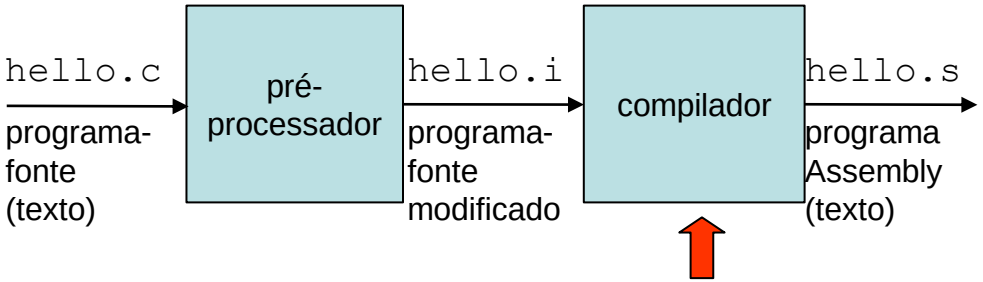


- Modifica o programa em C de acordo com diretivas começadas com #
  - Ex.: #include <stdio.h> diz ao pré-processador para ler o arquivo stdio.h e inseri-lo no programa fonte
- O resultado é um programa expandido em C, normalmente com extensão .i, em Unix

Gerando um executável

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

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

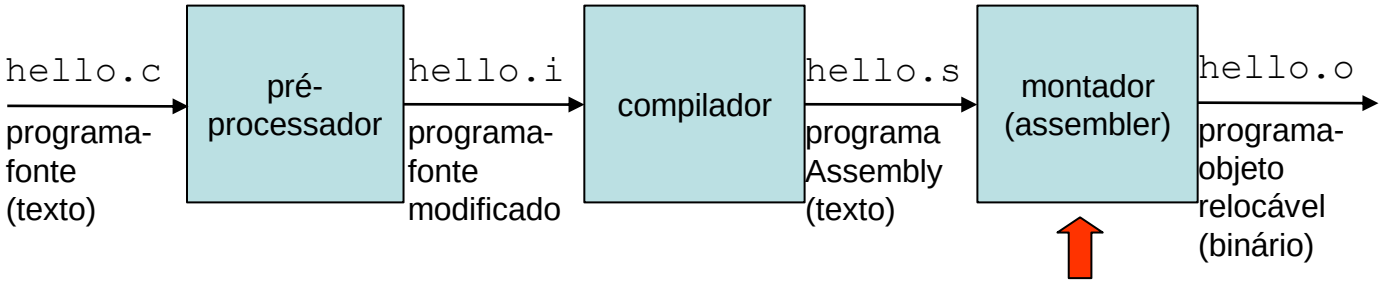


- Compilador traduz o programa .i em um programa em Assembly
  - É o formato de saída comum para os compiladores nas várias linguagens de programação de alto nível
    - i.e., programas em C, Java, Fortran, etc vão ser traduzidos para a mesma linguagem Assembly

Gerando um executável

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

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

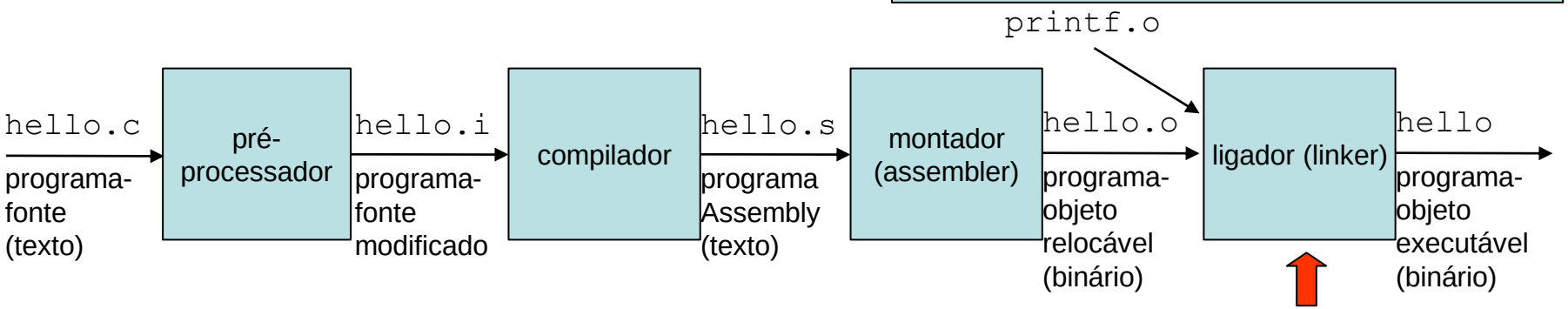


- Montador (Assembler) transforma o programa em Assembly em um programa binário em linguagem de máquina (chamado programa-objeto)
  - Os módulos de programas, compilados ou montados, são armazenados em um formato intermediário ("*Programa-Objeto Relocável*" – extensão .o)
- Endereços de acesso e a posição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

Gerando um executável

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

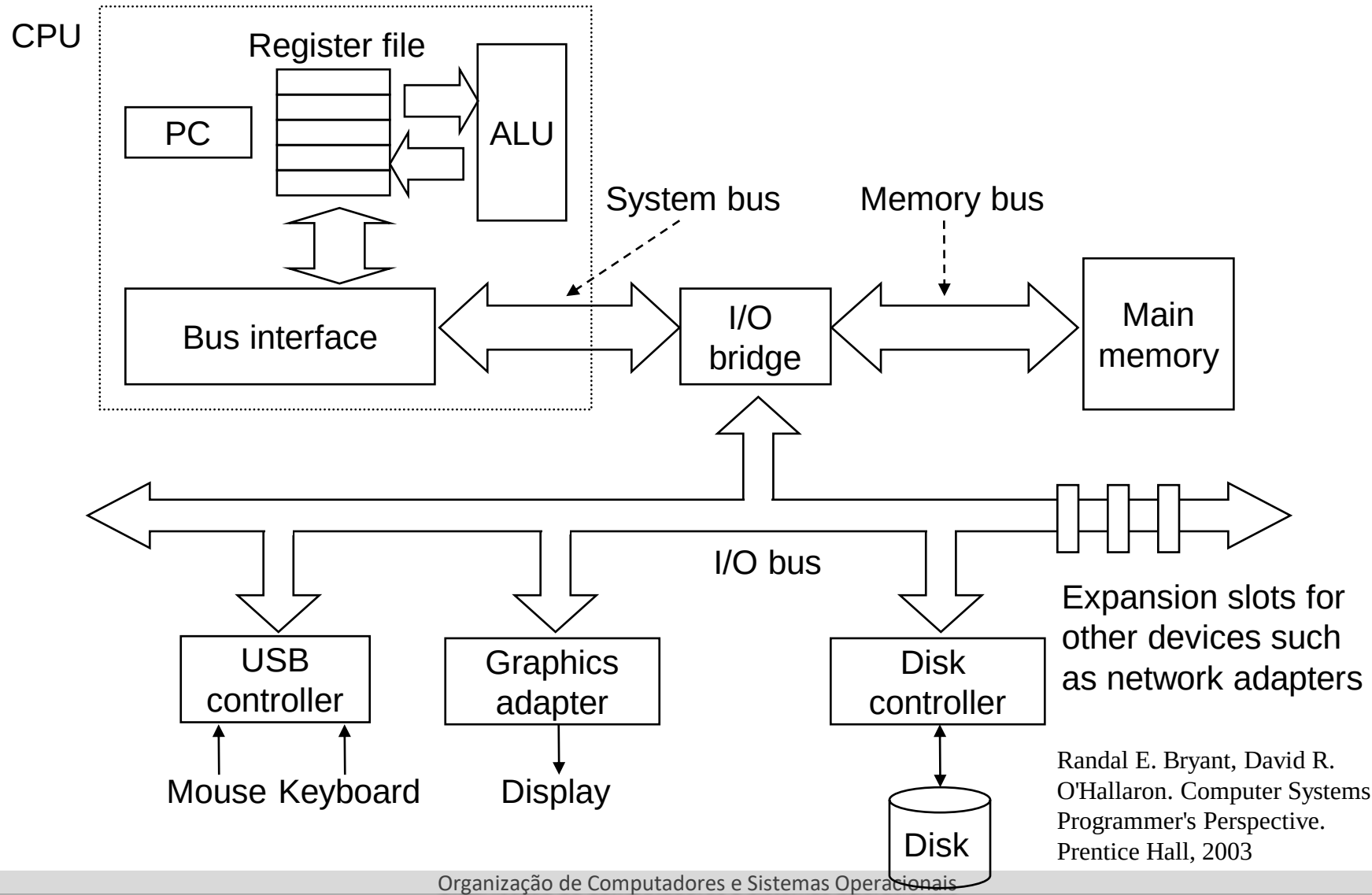
```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      printf("hello, world\n");
5.  }
```



- 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

Hardware

Um pouco de um computador típico

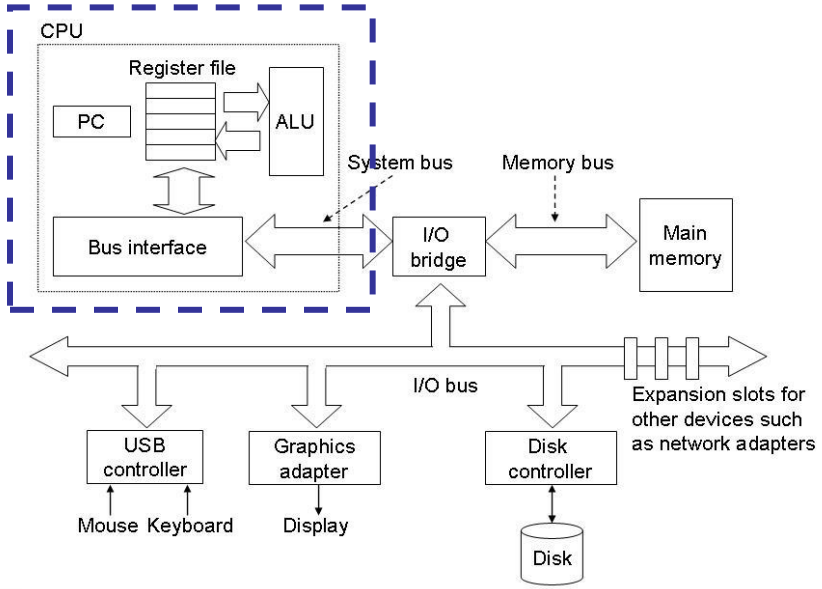


Randal E. Bryant, David R. O'Hallaron. Computer Systems: A Programmer's Perspective. Prentice Hall, 2003

CPU: Central Processing Unit

- Unidade de Controle
- ALU: Unidade Aritmética e Lógica
- Registradores
  - Funcionam como uma memória de acesso extremamente rápida
  - Baixa capacidade
  - 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

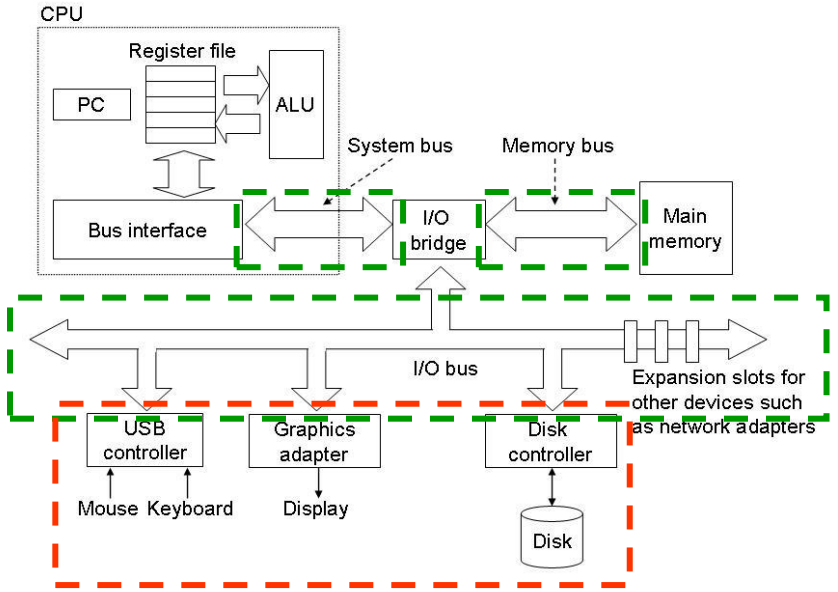
- A CPU, seguidamente, executa instruções requisitadas à memória
  - Ciclo *fetch-decode-execute*:
    1. busca instrução na memória
    2. Incrementa PC
    3. decodifica instrução
    4. executa instrução



Barramentos e Dispositivos de E/S

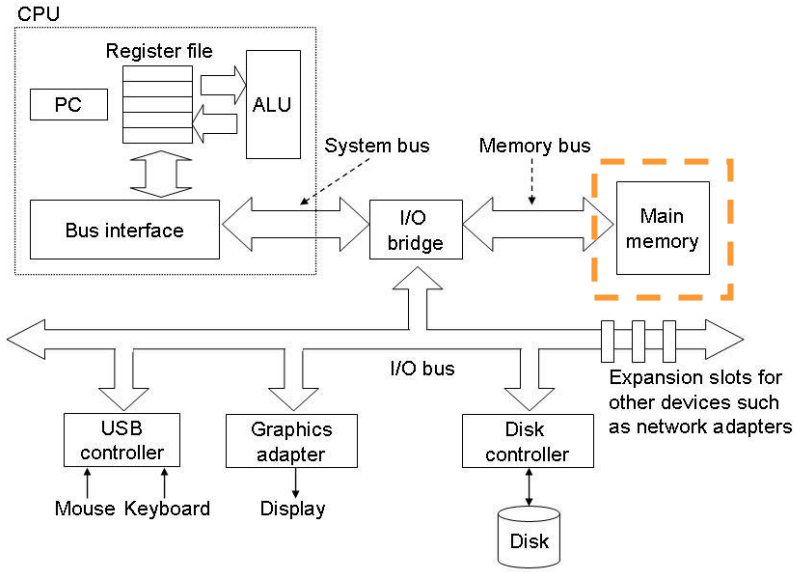
- Barramentos: conexões elétricas que carregam a informação entre os vários componentes da máquina

- Dispositivos de E/S:
  - Conexão da máquina com o mundo externo
  - Conectados ao barramento de E/S por
    - *controladores* (chips no próprio dispositivo ou na placa mãe) ou
    - *adaptadores* (quando placa separada)

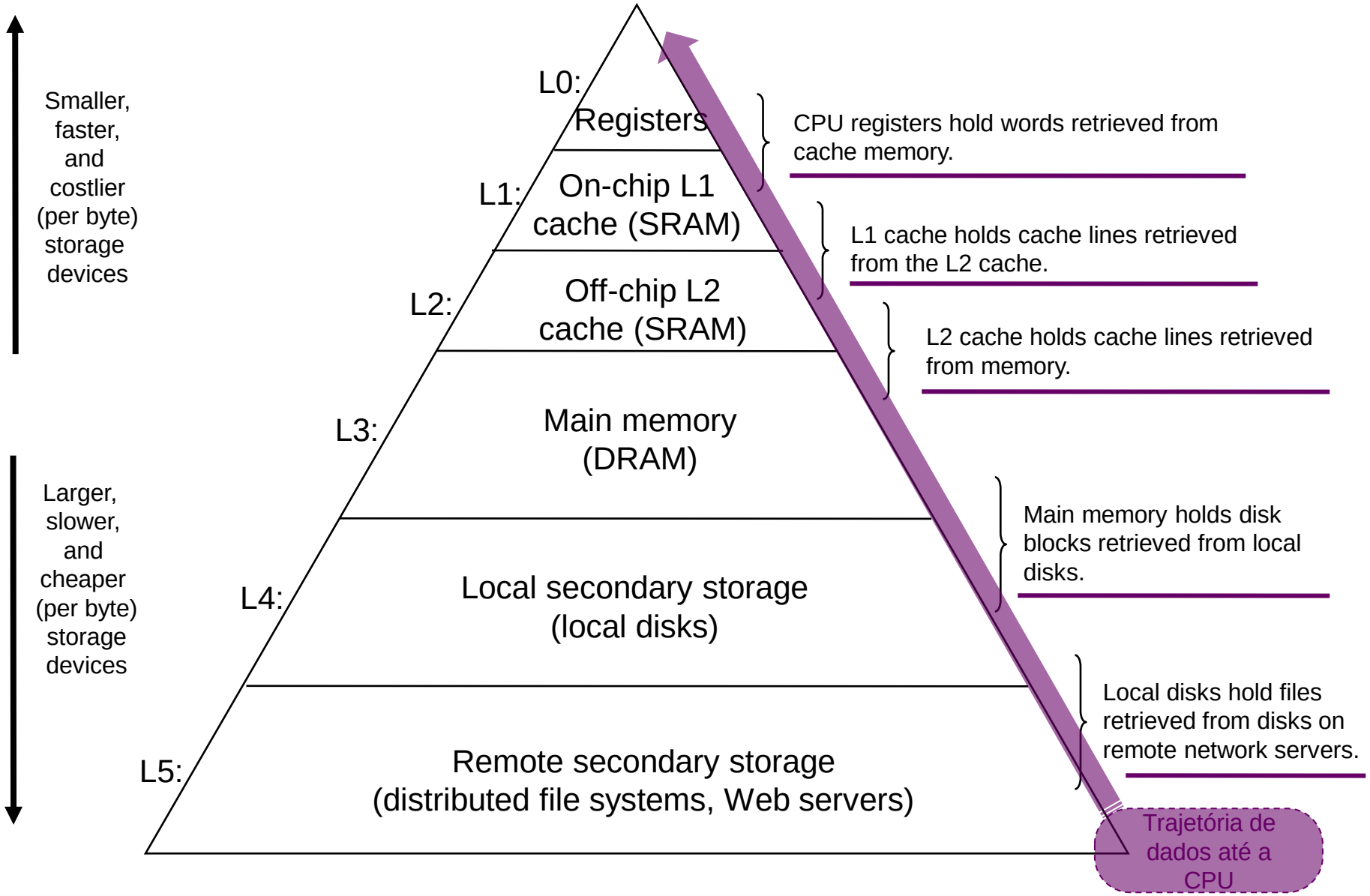


Memória

- 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



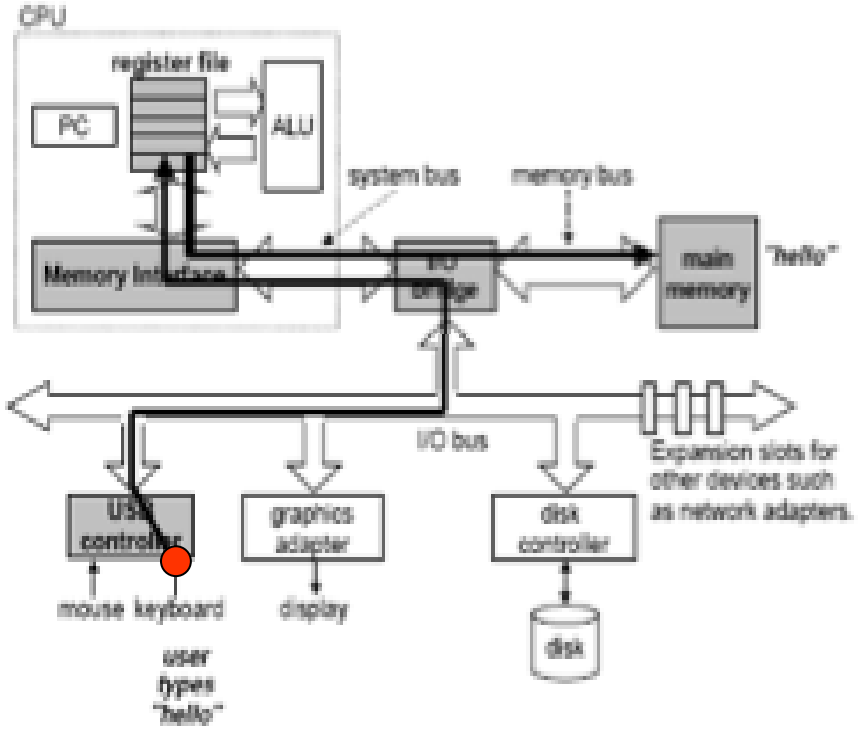
Execução

Como acontece...

Processo

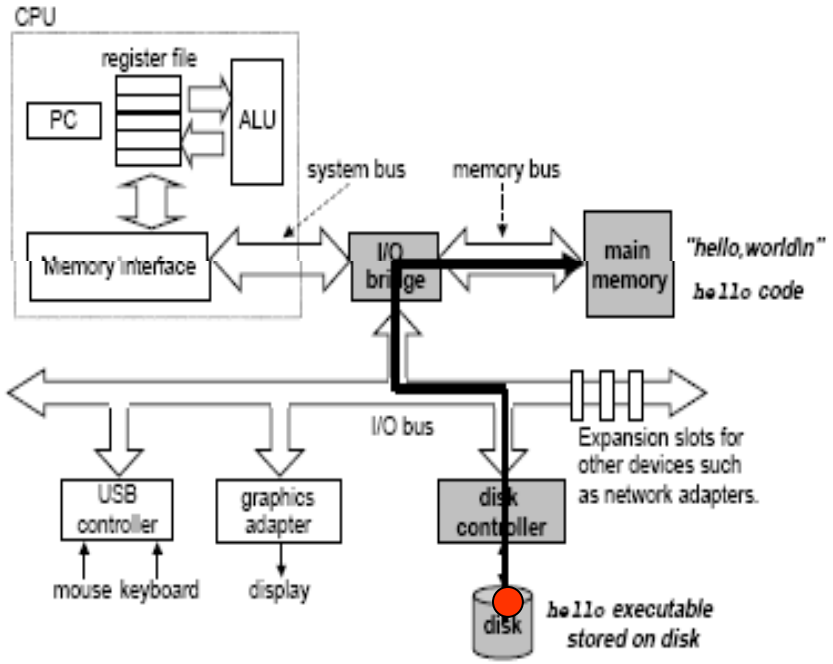
Conceito: Um programa em execução

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



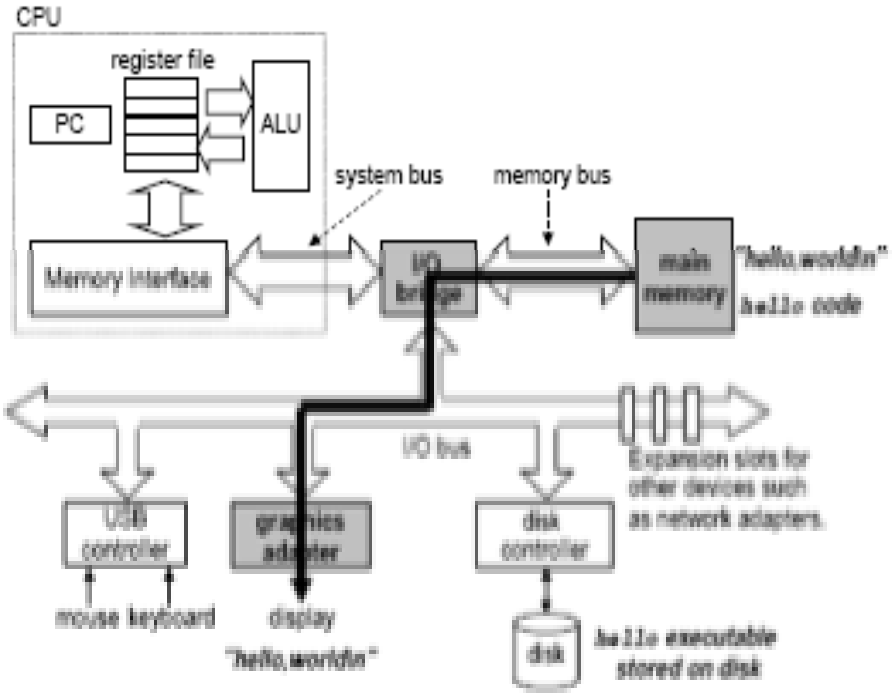
Programa em execução

2. Ao clicar “Enter”, sabe-se que acabou o comando e então é realizada uma seqüência de instruções para copiar código e dados do programa `hello` do disco para a memória principal



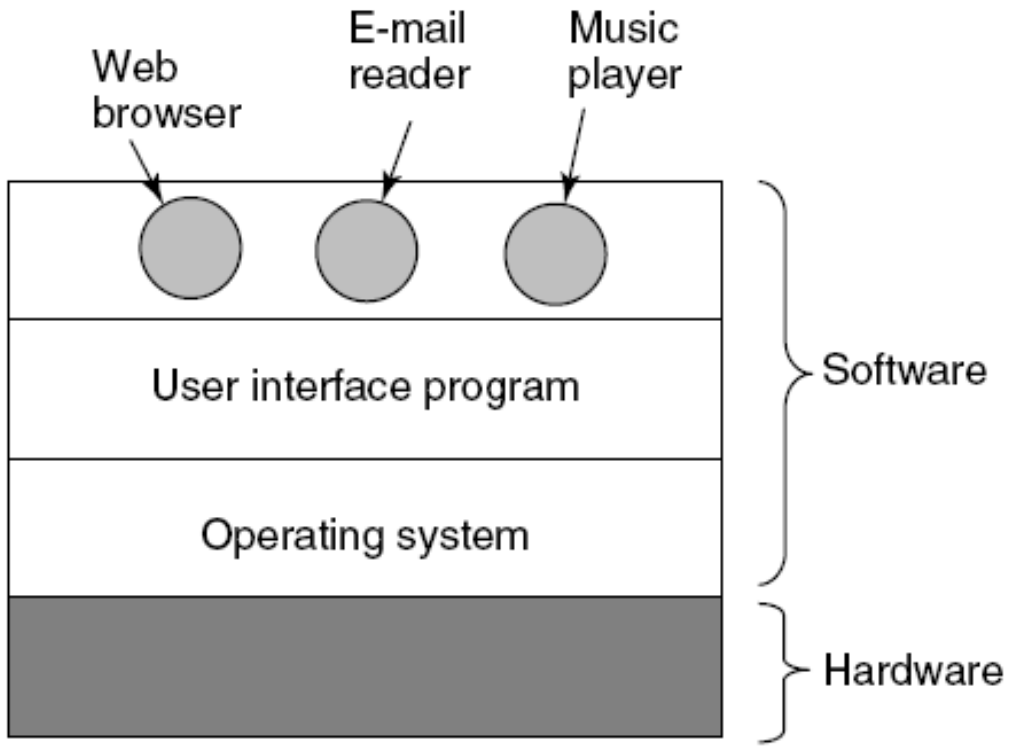
Programa em execução

3. PC aponta para o endereço de memória onde o programa `hello` foi escrito
4. Processador executa instruções em linguagem de máquina da função `main()` do programa



Mais de um programa em execução

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



Sistema Numérico

- Como contar coisas?
  - I,II, III, IIII ,IIIII, ..., IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
- De onde veio a base 10?
  - Algarismos Arábicos (séculos I a IV)
    - Sistema numérico posicional
    - O zero veio no século IX
- Sistema Romano

Ao final do curso você deverá ser capaz de...

- Explicar o funcionamento de um SO
  - Dos pontos de vista de mecanismo de abstração e gerenciamento de recursos

- Aplicar vários dos conceitos discutidos, como processos, *threads*, interrupções e escalonamento

...E não deverá ser capaz de

- Projetar um novo sistema operacional
- Implementar um novo sistema operacional
- Estender um sistema operacional existente

Material de Estudo

- Transparências das aulas
  - www.cin.ufpe.br/~svc/ocso

- Livro
  - Sistemas Operacionais Modernos – 2ª Edição. A. Tanenbaum, 2003
    - Opção: Modern Operating Systems 3 e. Prentice-Hall, 2008 (Já em Português, edição 2010)