Sistemas Operacionais

Revisão de conceitos de HW

Chamadas ao Sistema

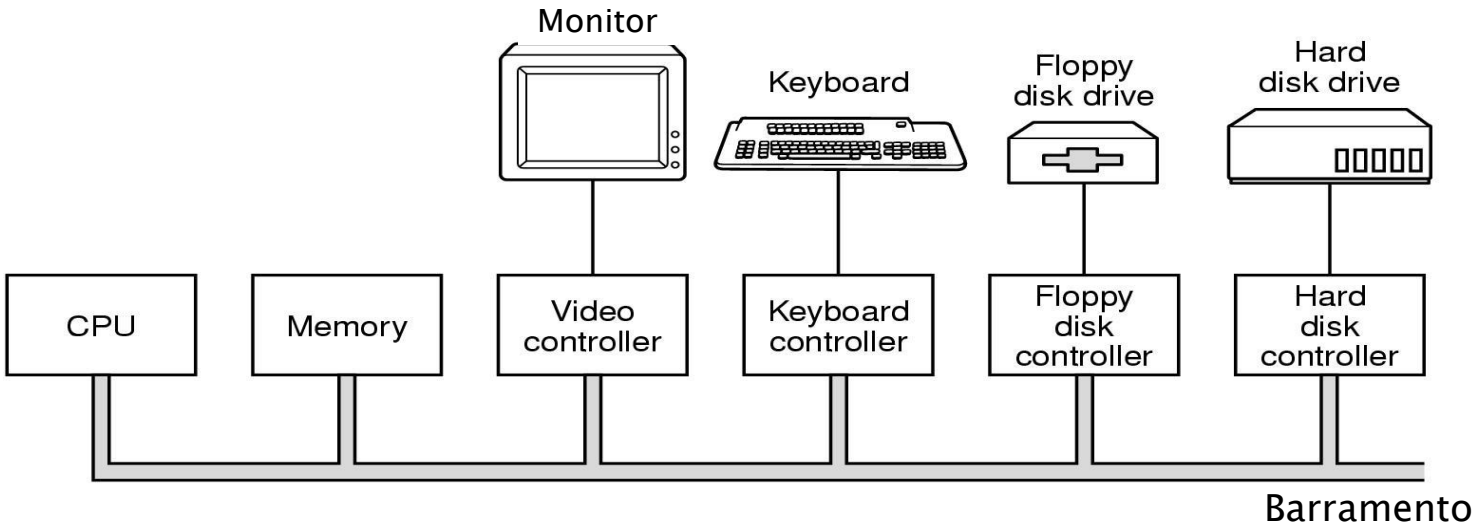
Estrutura do SO

Introdução

- O sistema operacional está intimamente ligado ao HW do computador no qual ele é executado
- O SO explora os recursos de hardware fornecendo um conjunto de serviços para os usuários do sistema
- O SO gerencia processadores, memória, dispositivos de E/S entre os vários usuários
- É importante ter algum conhecimento do hardware antes de entrar em detalhes sobre como funcionam os sistemas operacionais

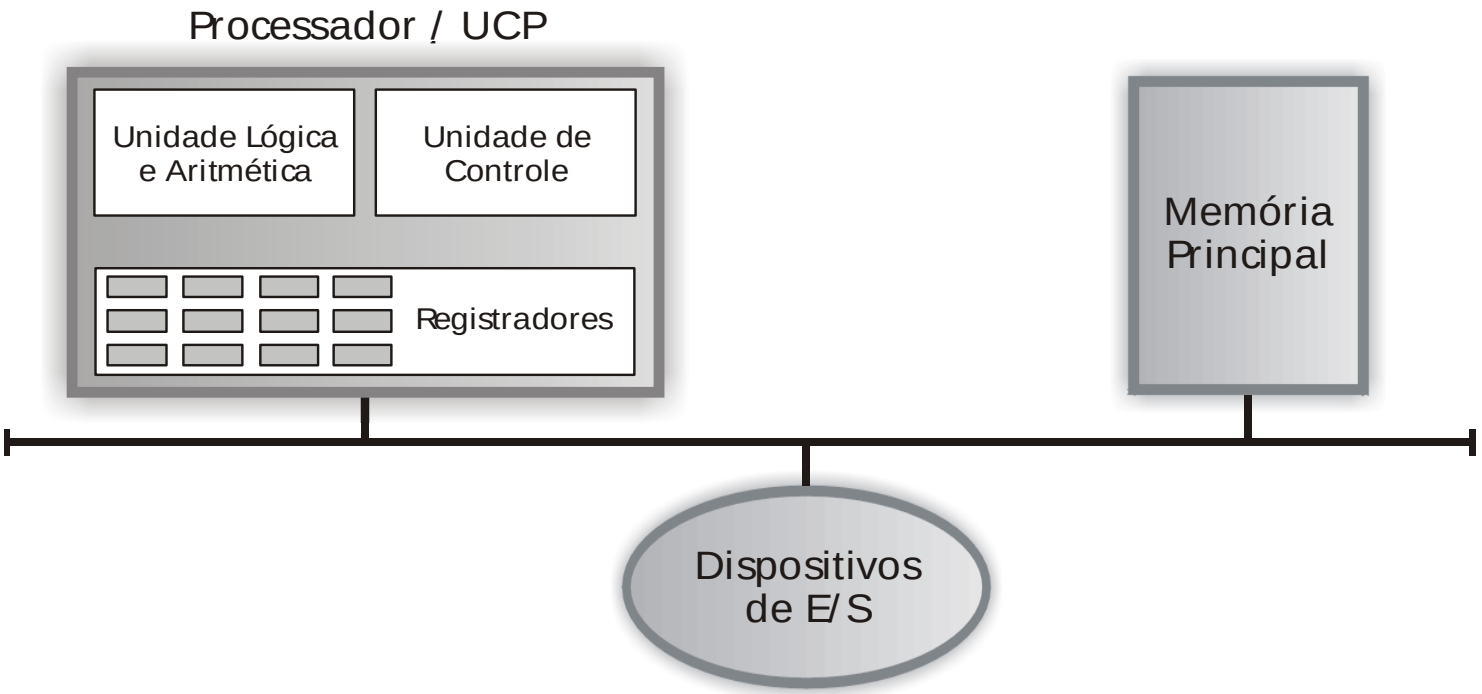
Introdução

- Um sistema computacional é um conjunto de circuitos eletrônicos integrados, formado por processadores, memórias, registradores, barramentos, monitores de vídeo, impressoras, etc
  - Todos os dispositivos manipulam dados na forma digital, o que proporciona uma maneira confiável de representação de dados



Introdução

- Todos os componentes do sistema computacional são agrupados em três subsistemas básicos: processador, memória principal e dispositivos de E/S

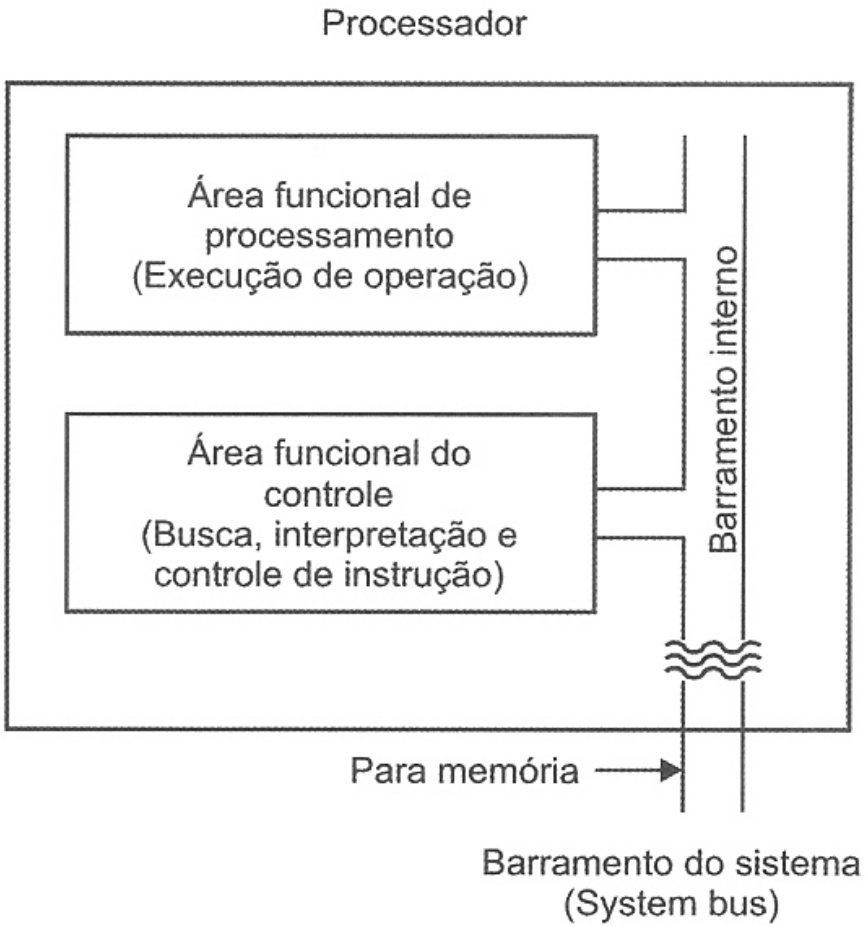


Processador

- A principal função do processador é controlar e executar as instruções presentes na memória principal
- A CPU
  - é o cérebro do computador
  - possui um conjunto específicos de instruções que ela é capaz de executar
  - gerencia todo o sistema computacional controlando as operações de cada unidade funcional

Funções do Processador

- Áreas funcionais de um processador



Processador

- *Unidade Lógica e Aritmética*: é responsável pela realização de operações lógicas e aritméticas
  - Área Funcional de Processamento
- *Unidade de controle*: gerencia as atividades de todos os componentes do computador, como gravação de dados e busca de instruções na memória principal
  - Área Funcional de Controle
- *Registradores*: armazenam dados temporariamente. É uma memória de alta velocidade, com capacidade reduzida

Área Funcional de Processamento

- **Unidade Lógica e Aritmética**
- Registradores de dados
- Registradores especiais que guardam informações decorrentes das operações realizadas

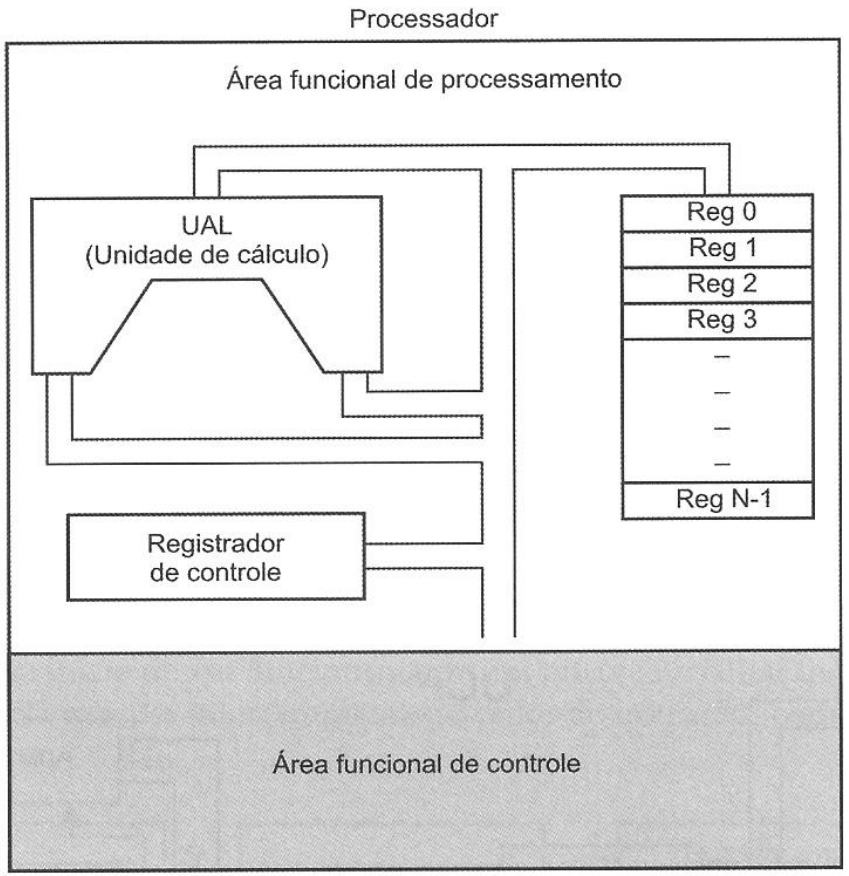
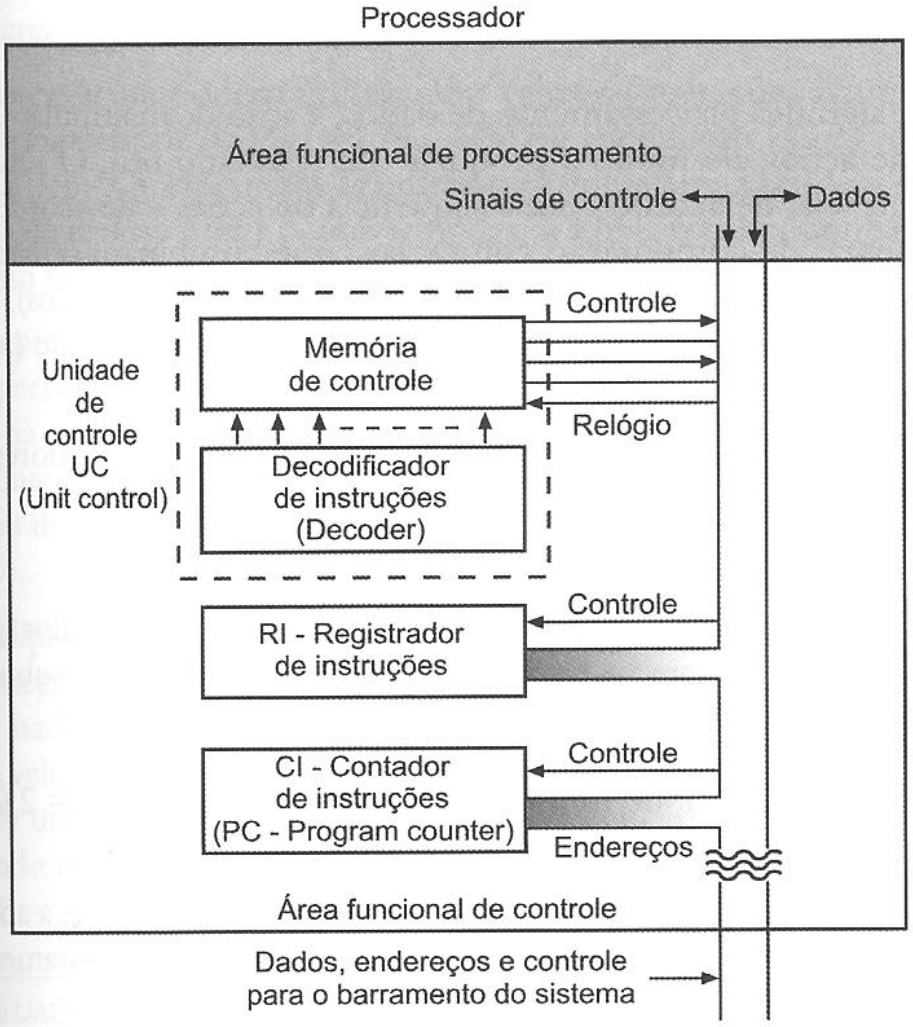


Figura 6.6 Componentes da área funcional de processamento.

Área Funcional de Controle



- **Unidade de Controle**
- Relógio
- Decodificador de Instrução
- Registrador de Instrução (RI)
- Contador de Instrução (CI)
- RDM e REM

Figura 6.8 Componentes básicos da área funcional de controle.

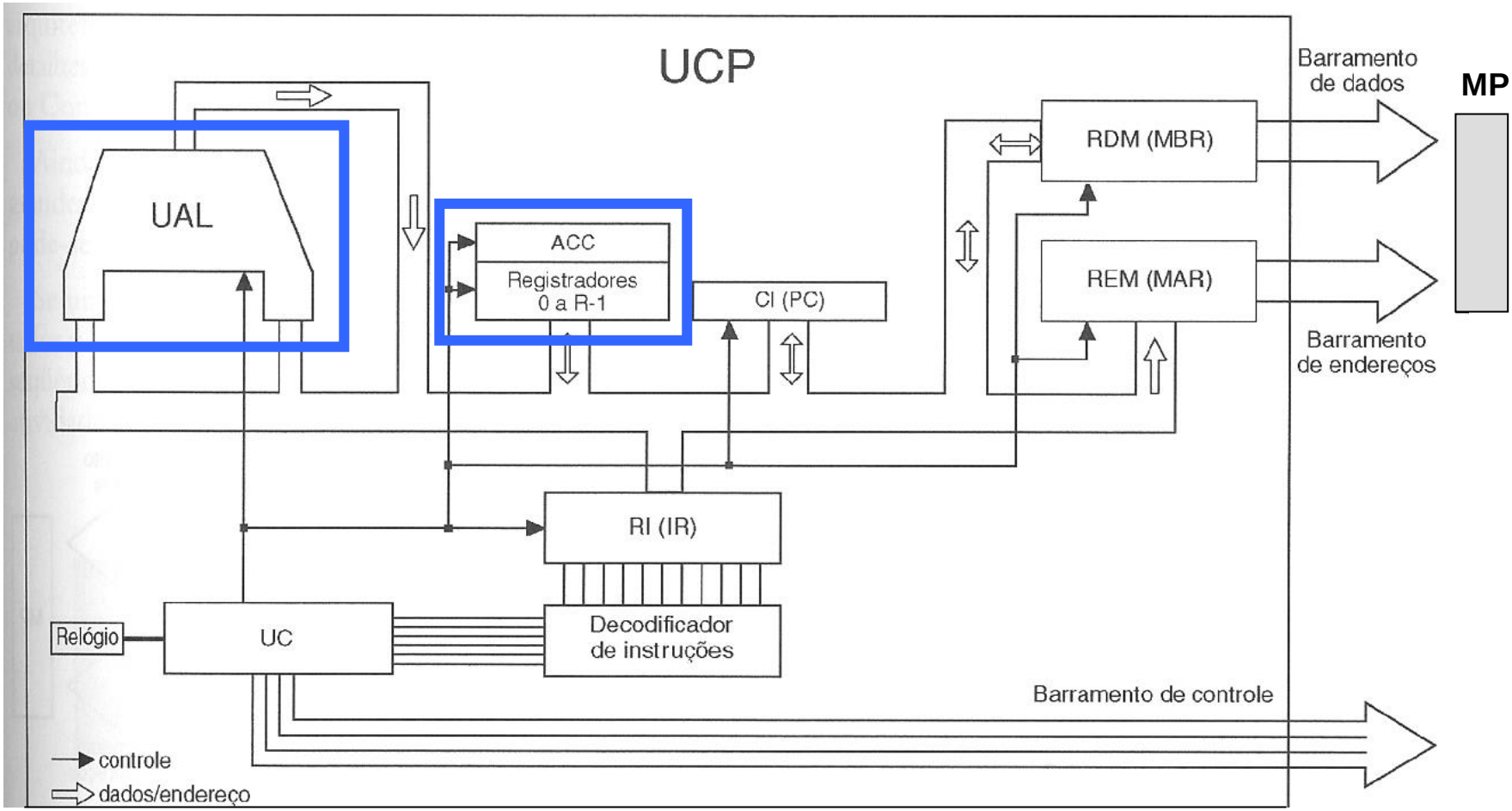
Registradores

- O sistema operacional deve estar ciente de todos os registradores
  - Quando o SO interrompe a execução de um programa ele deve salvar o estado de todos os registradores para que eles possam ser restaurados quando o programa for executado novamente
- Existem registradores
  - de uso geral (manipulados diretamente por instruções)
  - de uso específico (armazena informações de controle do processador e do sistema operacional)

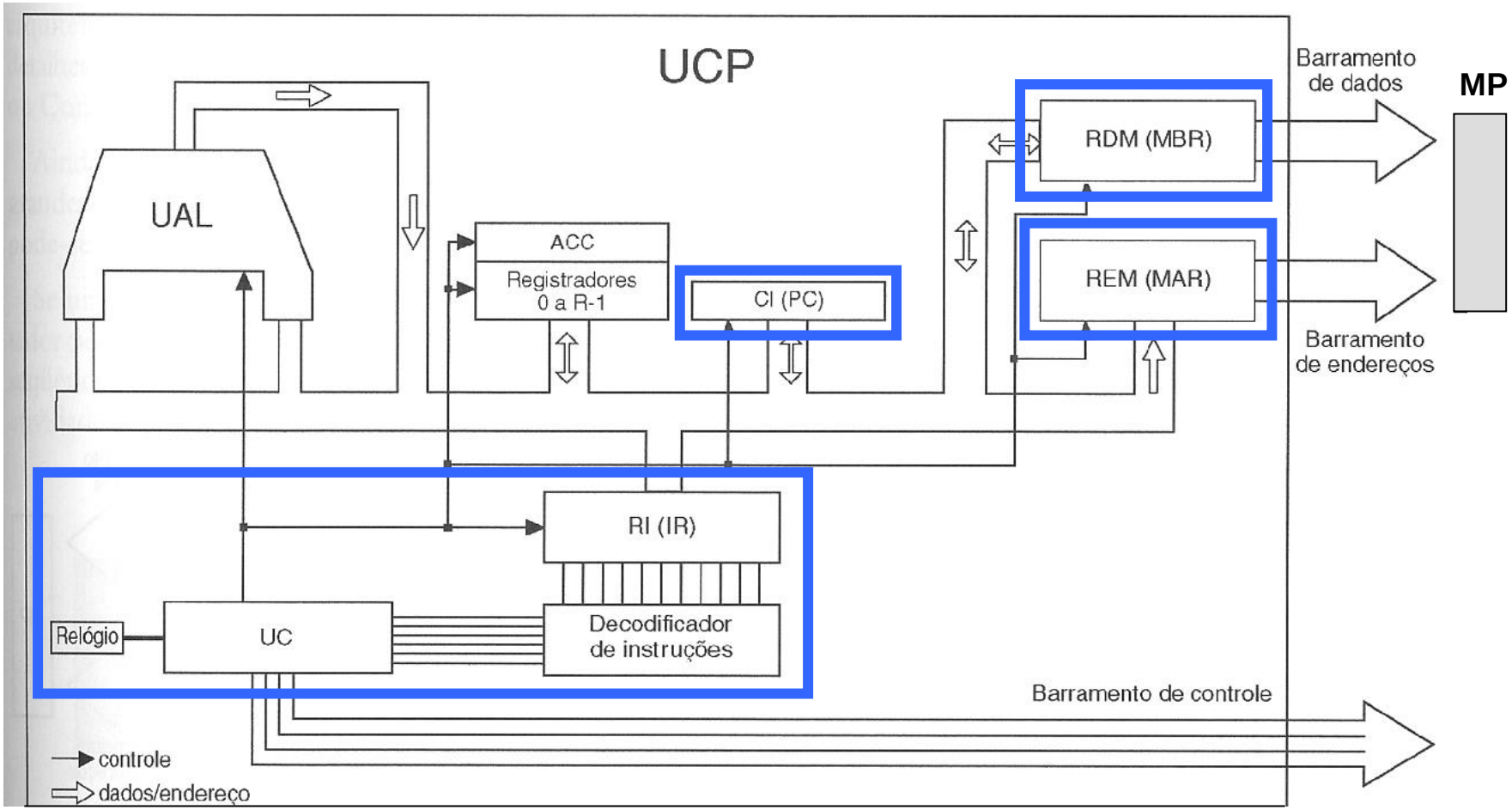
Registradores de uso específico

- *PC (program counter):* contém o endereço da próxima instrução que o processador deve buscar e executar
- *SP (stack pointer):* contém o endereço do topo da pilha que é a estrutura onde o sistema mantém informações sobre programas que estão sendo executados
- *IR (instruction register):* contém o endereço da última instrução buscada
- *PSW (program status word):* armazena informações sobre a execução de instruções, como ocorrência de overflow

Componentes da Área Funcional de Processamento

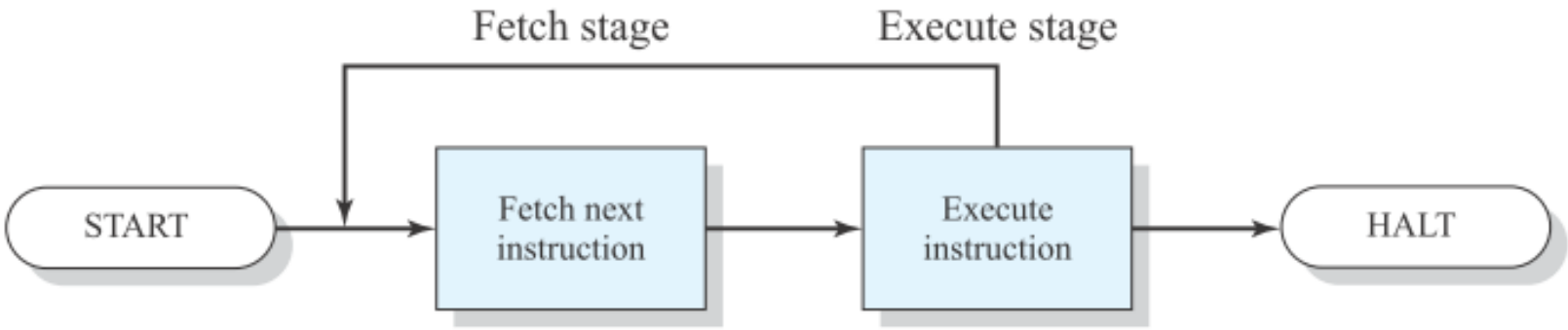


Componentes da Área Funcional de Controle



Ciclo de busca e de execução

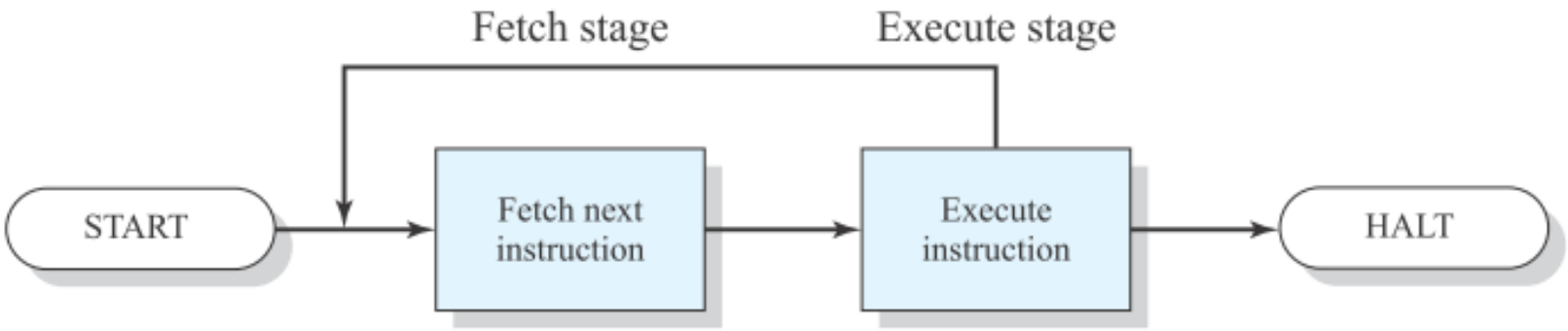
- O processamento de uma instrução consiste em dois passos: **busca** da instrução na memória e **execução**



- Ciclo de busca:
  - PC contém o endereço da próxima instrução a ser buscada
  - Instrução é buscada e colocada no IR para ser interpretada
  - PC é incrementado

Ciclo de busca e de execução

- O processamento de uma instrução consiste em dois passos: **busca** da instrução na memória e **execução**

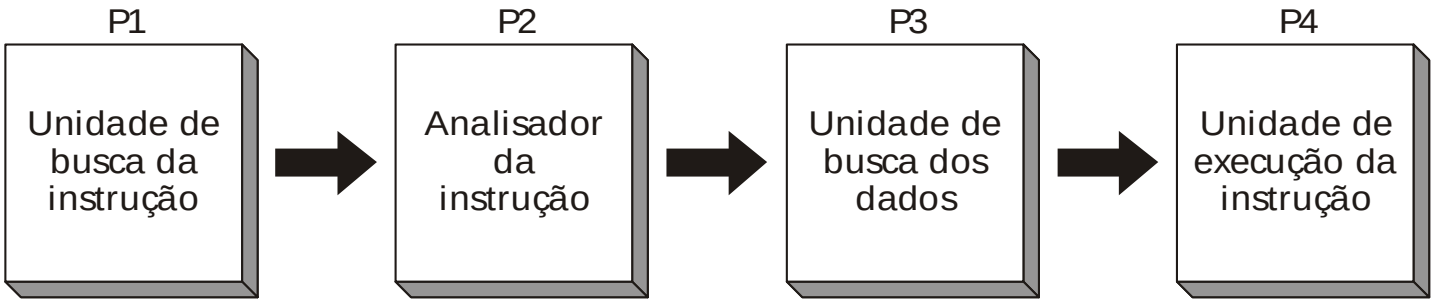


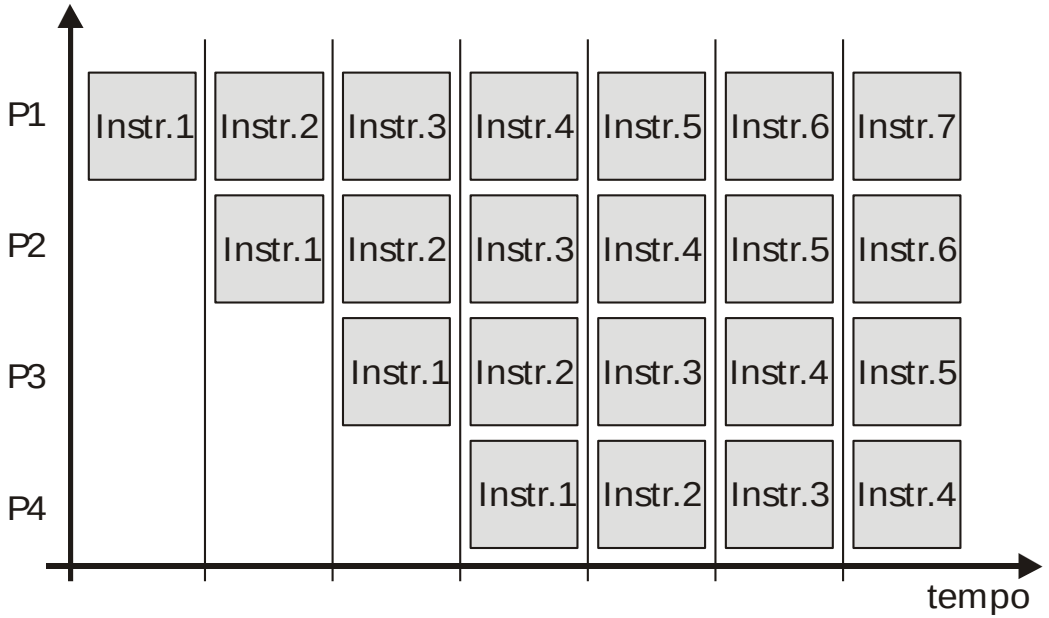
- Ciclo de execução (A CPU interpreta a instrução e executa ação):
  - CPU ↔ memória: transferência de dados
  - CPU ↔ E/S: transferência de dados
  - Processamento de dados: operação lógica ou aritmética

Pipeline

- Para melhorar o desempenho os projetistas de CPU abandonaram o modelo simples de busca, decodificação e execução de uma instrução por vez
- As CPUs tem unidades separadas de busca, decodificação e execução
  - Enquanto executa a instrução *n*, busca a instrução *n+1*
- Esta técnica é denominada pipeline e permite que várias instruções sejam processadas em estágios diferentes, ao mesmo tempo
- O conceito se assemelha a uma linha de montagem

Pipeline com 4 estágios





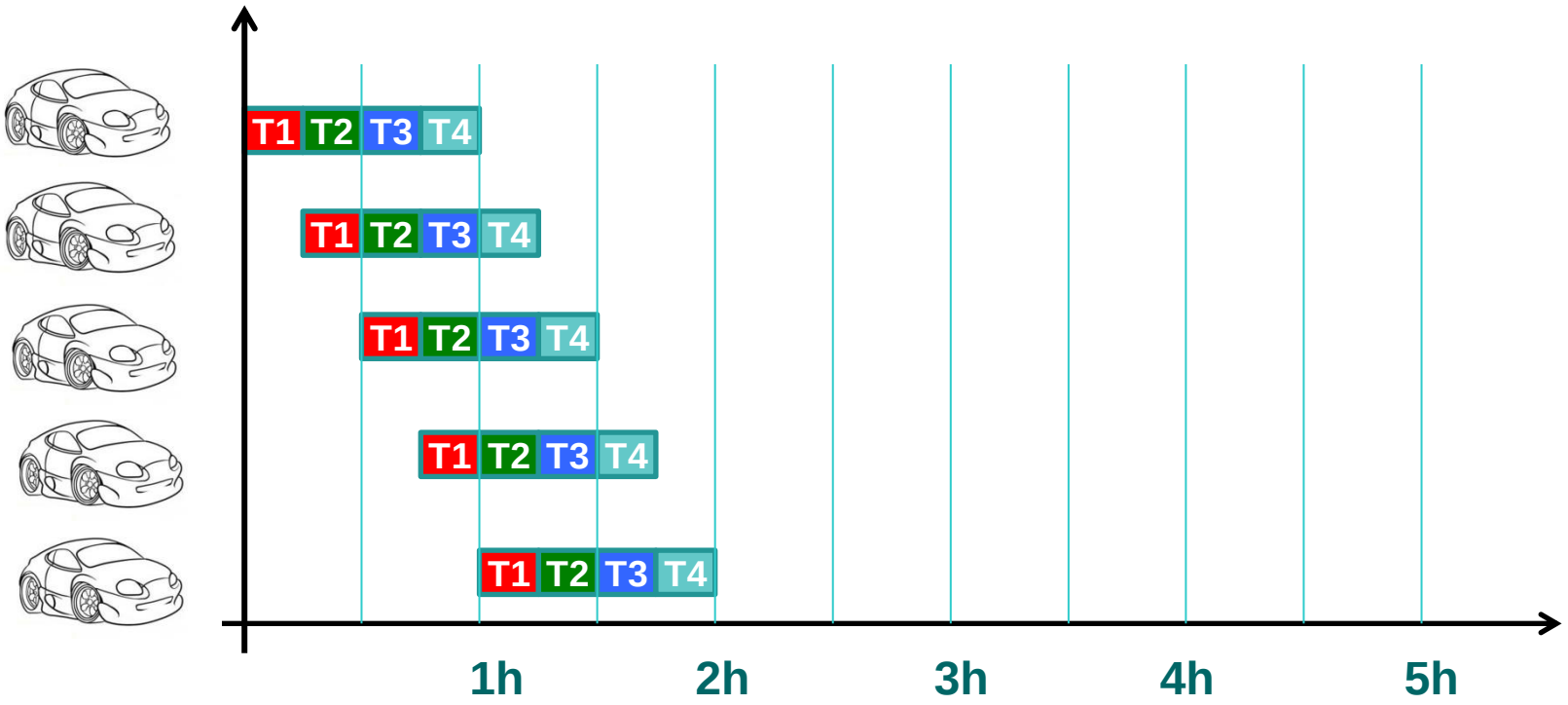
Exemplo de uma linha de Montagem

- Fábrica de automóveis
  - T1 – Montagem da carroceria
  - T2 – Montagem do motor na carroceria
  - T3 – Montagem das demais partes (pneus, bancos ,vidros, fiação)
  - T4 – Pintura
  - Supondo que cada etapa gaste 15 minutos, o tempo para se montar um carro seria de 1 hora

Exemplo de uma linha de Montagem

- Criando setores com os 4 estágios de montagem independentes.

- Usando *pipelining*



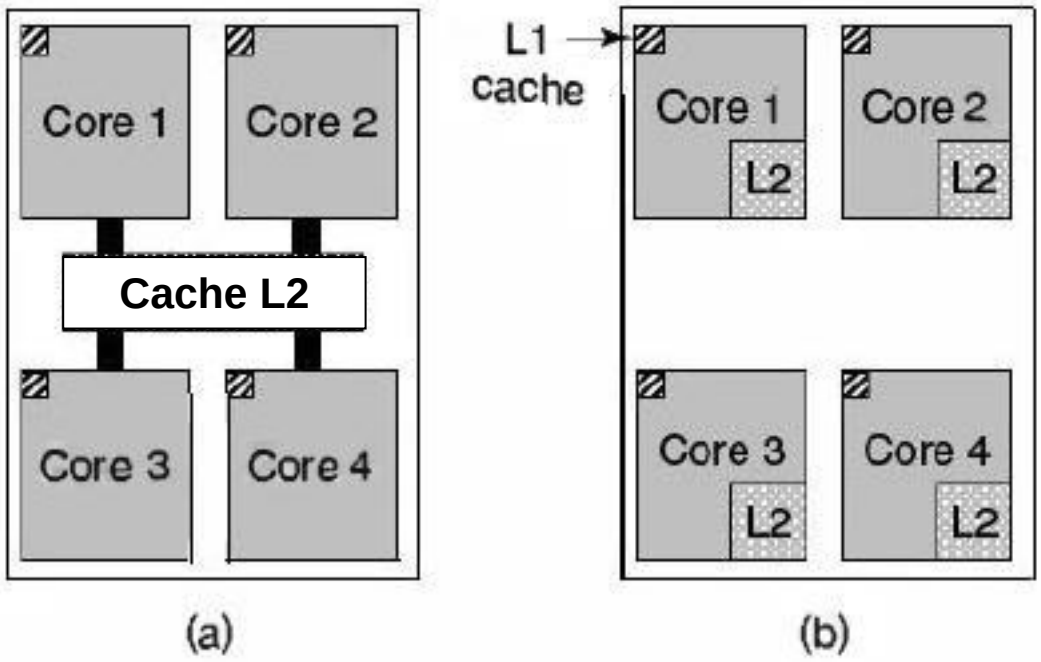
Multithreading ou Hyperthreading

- O Pentium 4 da Intel introduziu a propriedade de multithreading (hyperthreading)
  - Permite que a CPU mantenha o estado de duas threads diferentes e faça o chaveamento entre elas em uma escala de tempo de nanossegundos
  - Não proporciona paralelismo real
- O multithreading tem implicações para o Sistema Operacional
  - Cada thread aparece para o SO como uma CPU em separado
  - 2 CPUs com duas threads cada → O SO vê 4 CPUs!

Multithreading ou Hyperthreading

- Considere um sistema com duas CPUs e cada uma com duas threads (hyperthreading)
- Suponha que sejam iniciados 3 programas 100% CPU-bound:
  - P1 – 5 ns
  - P2 – 10ns
  - P3 – 20 ns
  - Quanto tempo é necessário para concluir a execução destes programas?
    - Os programas não bloqueiam durante a execução e não mudam de CPU após a atribuição inicial

Processadores Multinúcleo



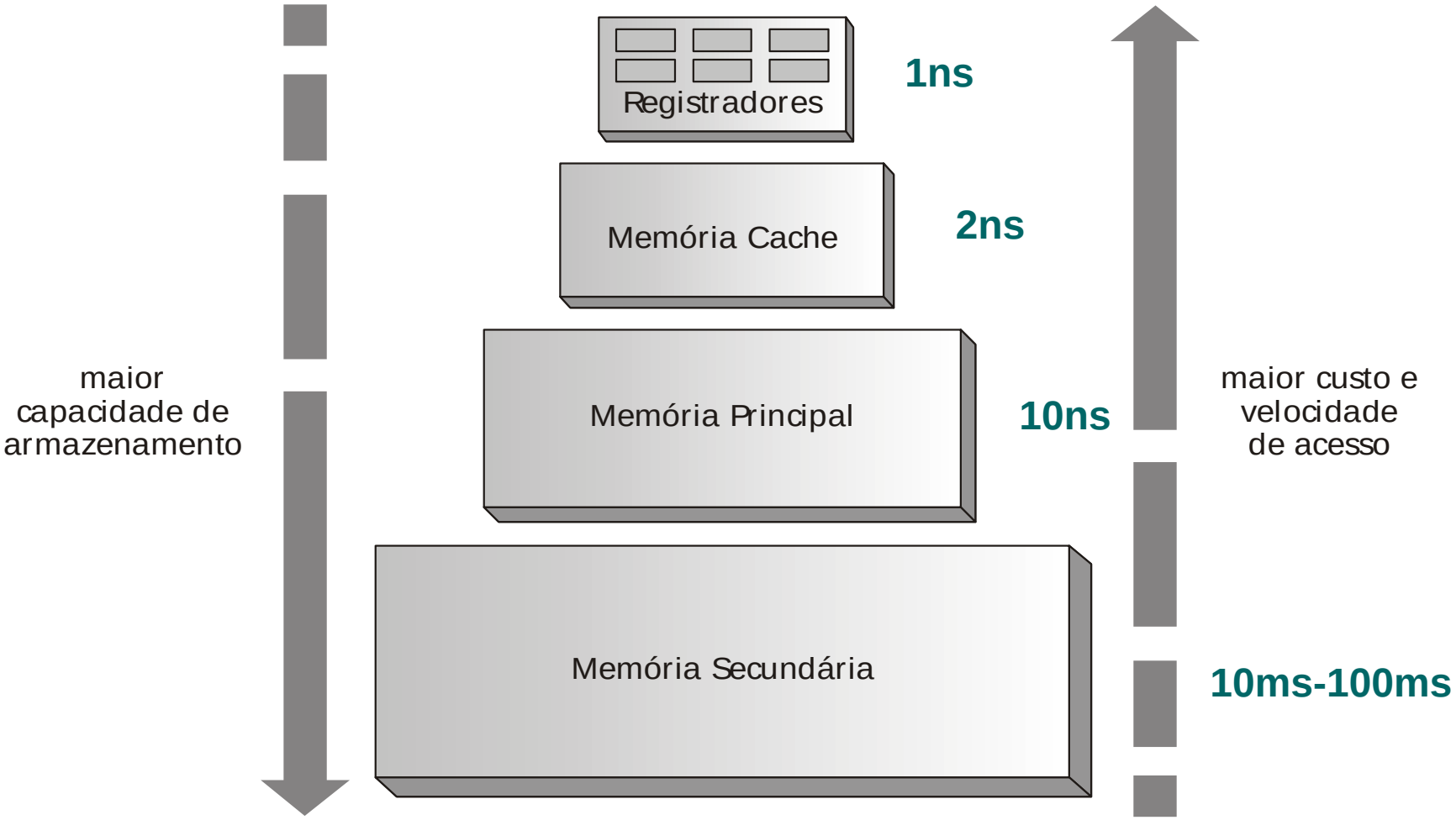
- Para fazer uso de chips multinúcleos de maneira eficiente são necessários sistemas operacionais de multiprocessadores

Memória

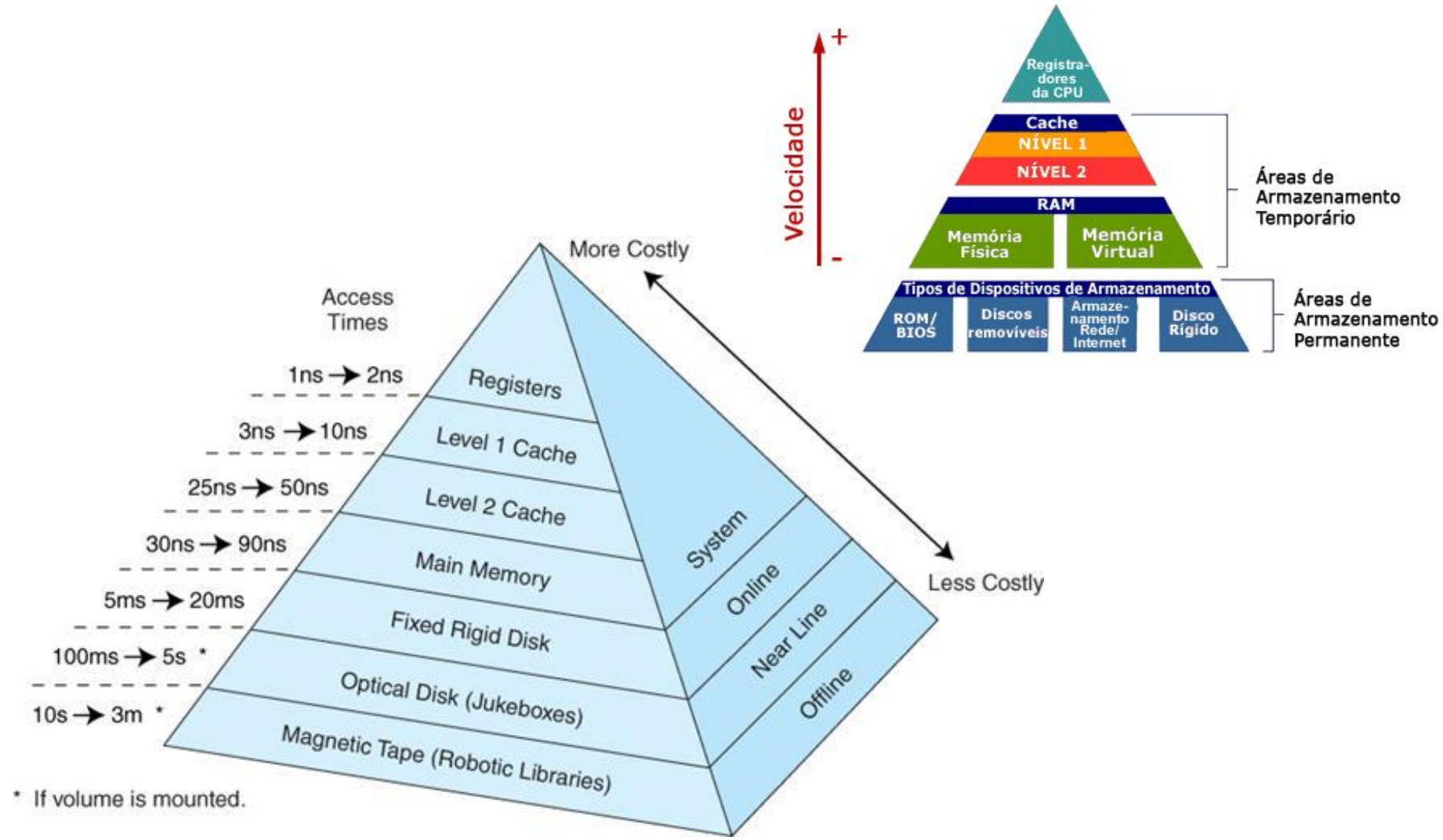
- Algumas questões devem ser consideradas ao se projetar uma memória: capacidade, velocidade e custo
  - Memória mais rápida, possui maior custo
  - Memória com maior capacidade, possui menor custo
  - Memória com maior capacidade, possui menor velocidade
- O projetista quer uma memória de baixo custo e com grande capacidade de armazenamento, porém para aumentar o desempenho a memória deve ser rápida

Tempo de Acesso X Tamanho X Custo

Memória


Registradores
1ns
Memória Cache
2ns
Memória Principal
10ns
Memória Secundária
10ms-100ms
maior capacidade de armazenamento
maior custo e velocidade de acesso

Hierarquia de Memória



Hierarquia de Memória

- Cada componente possui características próprias
- Para definir a função e as diferenças de cada componente, as seguintes características são analisadas:
  - Tempo de acesso
  - Capacidade
  - Volatilidade
  - Custo

Memória Secundária

- Meio não volátil de armazenamento de programas e dados
- Grande capacidade de armazenamento (TB)
- Acesso é bem mais lento do que a memória principal
  - HDs – 8 a 30 ms
- O disco também pode ser usado como uma extensão da memória principal $\rightarrow$ Memória Virtual

Memória Principal

- É a memória básica de um sistema de computação, onde são armazenadas as instruções e os dados
- Tempo de acesso: um ciclo de memória (10 a 50 ns)
- Capacidade superior à memória cache (GB)
- Volátil
  - Existe porém uma pequena parte da Memória Principal que é constituída de memória não-volátil (ROM) – instruções que são executadas quando computador é ligado
- Custo mais baixo que o da memória cache

Memória Principal

- A memória é composta por unidades de acesso chamadas células

- O acesso a uma célula é especificado por um endereço

  - A especificação do endereço é realizada através do registrador REM (MAR - memory address register). Assim, a unidade de controle sabe qual célula de memória será acessada.

  - O número de células endereçáveis é limitado pelo tamanho do REM (*n* bits endereça 2^n células)

Exercício de MP

- Considere uma MP com 32K células de 1 byte cada
- Perguntas:
  - Qual o número de células endereçáveis?
  - Qual o tamanho em bits do registrador REM?
  - Qual o tamanho em bits da MP?

Memória Cache

- Podem ser internas ou externas ao processador (L1, L2, L3)
- Tempo de acesso: 5 a 20 ns
- Capacidade: deve ser suficiente para evitar que a informação buscada não esteja presente e que assim o sistema sofra um atraso para transferir a informação da MP para a cache
  - Valores típicos: cache L1 (32 a 256KB) e cache L2 (4MB)
- Voláteis e alto custo

Memória Cache

- *O propósito do uso da memória cache é minimizar a disparidade existente entre a velocidade com que a CPU executa instruções e a velocidade com que os dados são lidos e gravados na MP*

Memória Cache

- Diferença de velocidade entre Processador e MP
  - O processador executa uma operação rapidamente e fica em estado de espera para receber dados da memória
  - O processador é muito mais rápido que a MP
  - O problema de diferença de velocidade se torna difícil de solucionar apenas com melhoras no desempenho da MP, devido a fatores de custo e tecnologia
  - Enquanto o desempenho dos processadores dobra a cada 18/24 meses, o tempo de acesso a MP aumenta bem menos (cerca de 10% por ano)
    - O GAP de velocidade entre processador e MP vem aumentando!

Memória Cache

- Na década de 60, diversos pesquisadores, principalmente da IBM, se reuniram para analisar o comportamento dos programas
  - Foi verificado que em cada período de tempo (relativamente curto) só uma fração da MP era acessada
  - Desta análise surgiu um princípio de funcionamento dos programas, chamado genericamente de princípio da localidade
- Princípio da localidade:

É a tendência do processador referenciar instruções e dados na memória principal localizados em endereços próximos.

Memória Cache

- Princípio da localidade:

  É a tendência do processador referenciar instruções e dados na memória principal localizados em endereços próximos.

  Por exemplo, esta tendência pode ser justificada pela alta incidência de estruturas de repetição e acessos a vetores.

- Assim o princípio da localidade garante que após a transferência de um bloco da MP para a cache, haverá alta probabilidade de cache hits em futuras referências

Conceito de Localidade

- Princípio de localidade
  - Localidade espacial: endereços próximos tendem a ser acessados em curtos períodos de tempo
  - Localidade temporal: um mesmo endereço tende a ser acessado novamente em um futuro próximo

Localidade Temporal

- Programas tendem a usar o mesmo endereço em um curto espaço de tempo
  - Loops repetem um mesmo conjunto de instruções

Cálculo da média de 2 turmas, A e B

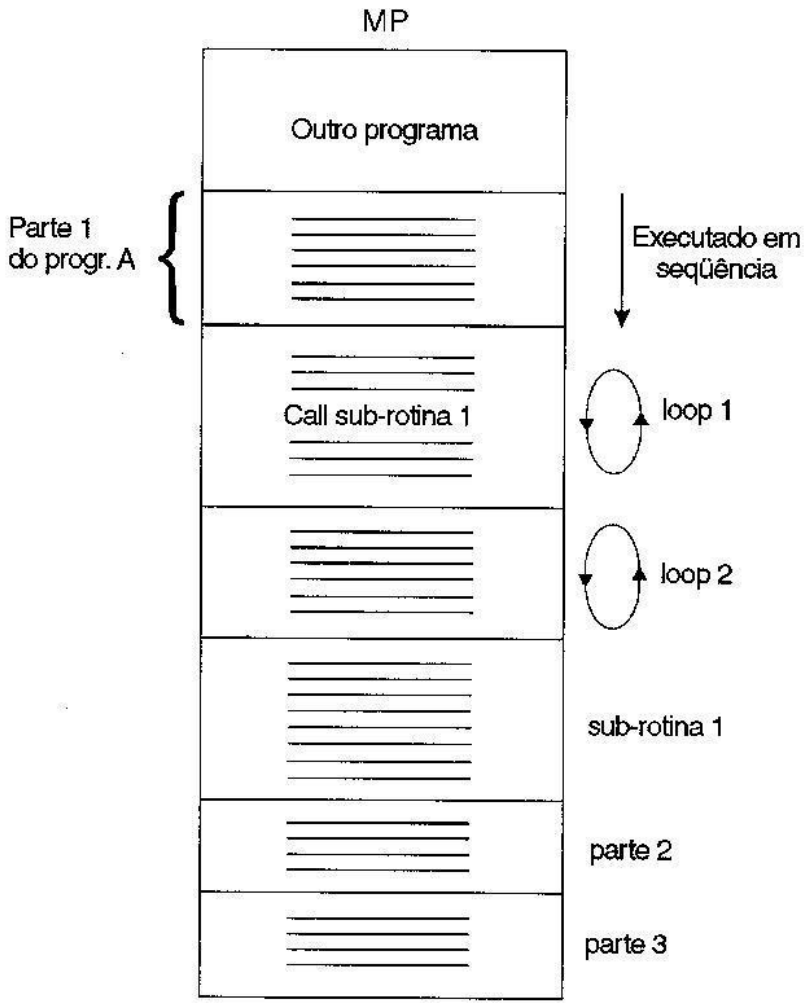
```
void main ()
{
printf ("Número de alunos da turma A: ");          início de execução em seqüência
scanf ("%d", &quant_A);
maior_nota_A = -1;
soma_nota_A = 0;                                   término execução em seqüência
for (i=0;i < quant_A; i++)
{                                                  loop  - início
  printf  ("Informe a matrícula do aluno: ");
  scanf  ("%d",&matr[0][i]);
  printf  ("Informe a nota: ");
  scanf  ("%f",&nota[0][i]);
  if (nota[0][i] > maior_nota_A )
     maior_nota_A = nota[0][i];
  soma_nota_A = soma_nota_A + nota[0][i];
}                                                  loop - término
media_nota_A = soma_nota_A / quant_A;

clrscr ();                                         sub-rotina (limpa tela)

printf  ("Número de alunos da turma B: ");         início de exec. em seq.
scanf  ("%d", &quant_B);
total = 0;
soma_nota_B = 0;                                   término exec. seq.
for (i=0;i < quant_B; i++)
{                                                  loop - início
  printf  ("Informe a matrícula do aluno: ");
  scanf  ("%d",&matr[1][i]);
  printf  ("Informe a nota: ");
  scanf  ("%f",&nota[1][i]);
  if (nota[1][i] > maior_nota_A)
     total++;
  soma_nota_B = soma_nota_B + nota[1][i];
}                                                  término - loop
media_nota_B = soma_nota_B / quant_B;              cálculo média turma B
printf  ("A média dos alunos da turma A foi: %4.2f", media_nota_A);
printf  ("A média dos alunos da turma B foi: %4.2f", media_nota_B);
printf  ("A nota mais alta da turma A foi: %4.2f", maior_nota_A);
printf  ("%d alunos da turma B obtiveram nota superior à maior nota da turma A", total);
}
```

Figura 5.3 Exemplo de programa para demonstração de localidades na sua execução.

Localidade Espacial



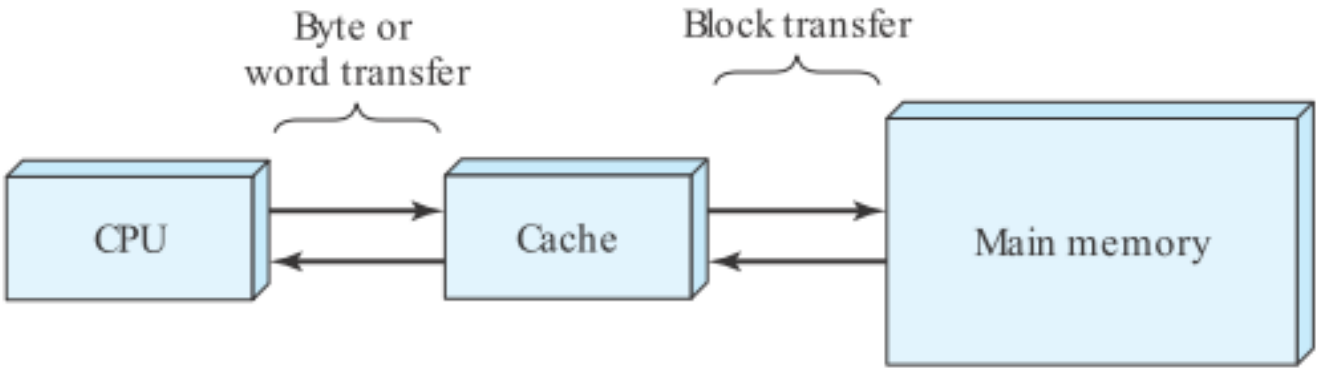
- Endereços próximos tendem a ser acessados em um curto período de tempo

Figura 5.2 Um programa em execução com várias partes (exemplo do princípio de localidade especial).

Memória Cache

- A memória cache armazena uma pequena parte do conteúdo da MP e deve tirar proveito do princípio da localidade

- Quando o processador faz referência a um dado, primeiro é verificado se ele se encontra na cache



Memória Cache

- Caso o dado esteja na cache, não há necessidade de acesso a memória principal, diminuindo o tempo de acesso (*cache hit*)
- Caso o dado não esteja na cache, o acesso a memória principal será obrigatório (*cache miss*)
  - Neste caso, é transferido um bloco de dados da memória principal para a memória cache
- O tempo adicional de busca na cache e depois na memória principal é compensado pela alta quantidade de *cache hits* $\rightarrow$ princípio da localidade
  - Taxa de acertos → 95% a 98%

Memória Cache

- O conceito de *caching* exerce um papel importante em muitas áreas da computação
- Sempre que um recurso pode ser dividido em partes, sendo algumas usadas com mais frequência que as outras → *caching* pode ser usado para melhorar o desempenho
- É um conceito muito usado pelos sistemas operacionais
  - Arquivos muitos usados podem ser mantidos em MP para evitar a busca em disco de modo repetitivo

Registradores

- São internos ao processador
  - Guardam instruções e dados que estão sendo manipulados em cada operação executada pelo processador
- Tempo de Acesso
  - Por serem construídos com a mesma tecnologia do processador tem o menor tempo de acesso do sistema (1 a 2 ns)
- Capacidade
  - São fabricados para armazenar um dado, uma instrução ou um único endereço (< 1KB)
- Voláteis

Dispositivos de E/S

- A CPU e a memória não são os únicos recursos que o sistema operacional tem que gerenciar
- Os dispositivos de E/S interagem intensivamente com o SO
- São utilizados para permitir a comunicação entre o sistema computacional e o mundo externo
- Podem ser divididos em duas categorias:
  - Dispositivos usados como memória secundária
  - Dispositivos que servem como interface homem-máquina

Dispositivos de E/S

- O controle real do dispositivo de E/S é complicado
- O controlador do dispositivo apresenta uma interface mais simples para o SO
- Cada controlador é diferente, possuindo softwares diversos para o controle dos dispositivos
  - O software que conversa com o controlador → driver do dipositivo de E/S
  - Cada fabricante de controladores oferecem drivers para cada tipo de sistema operacional
  - O driver do dispositivo tem que ser colocado dentro do núcleo do sistema operacional

Operações de E/S

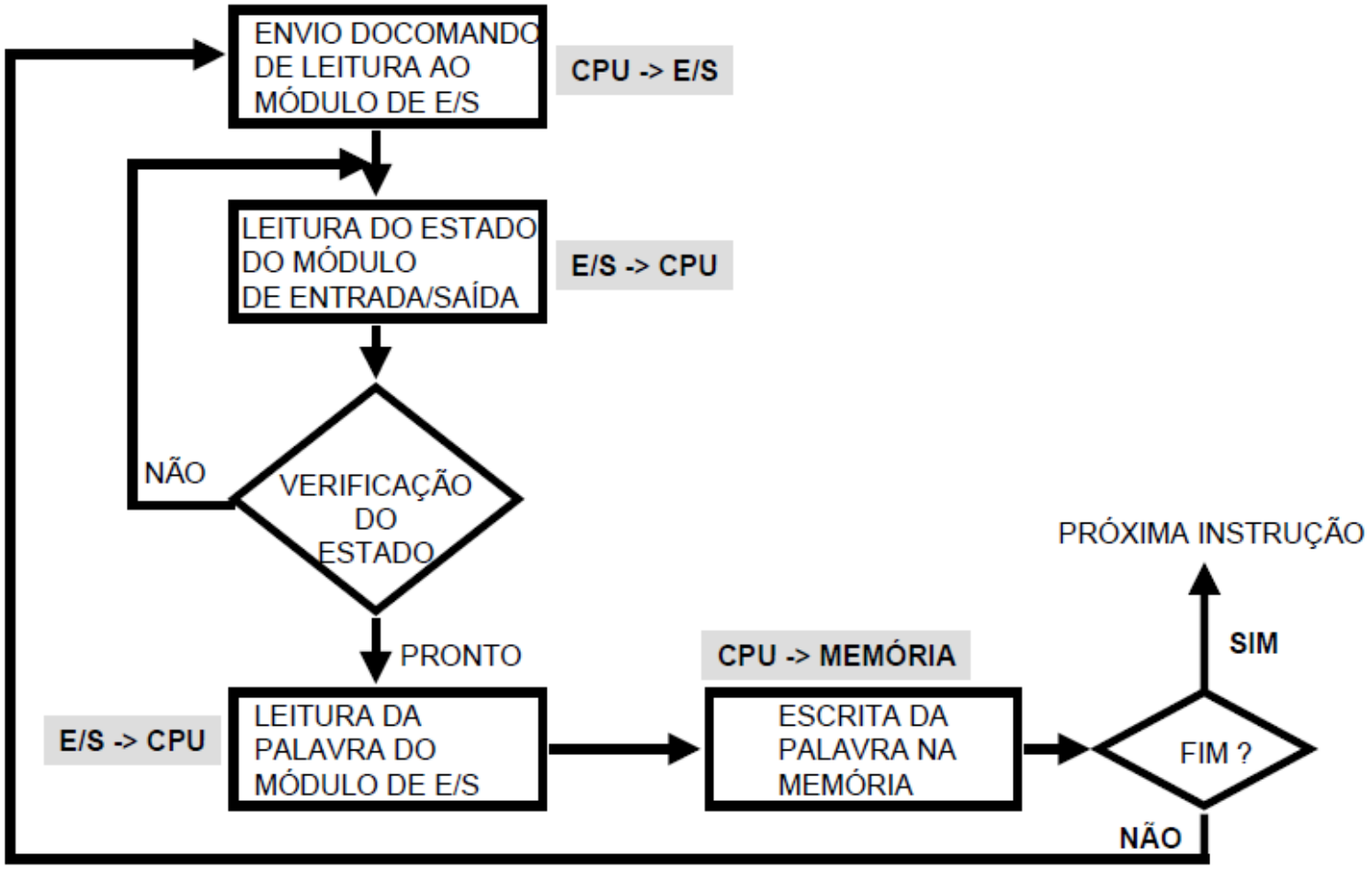
- As operações de entrada e a saída podem ser realizadas de 3 formas diferentes:
  - Controlada por programa
  - Controlada por interrupção
  - DMA (direct memory access)

Operações de E/S

- Controlada por programa
  - O programa do usuário emite uma chamada ao sistema, que o núcleo do SO traduz em uma chamada para o driver apropriado
  - O driver inicia a E/S perguntando ao dispositivo continuamente sobre o seu estado.
  - Desta forma a CPU permanece ocupada testando permanentemente o estado do periférico para saber quando a operação chega ao final → Espera ocupada
  - Quando termina a operação de E/S, o Sistema Operacional retorna o controle ao processo que o chamou

Operações de E/S

- Controlada por programa



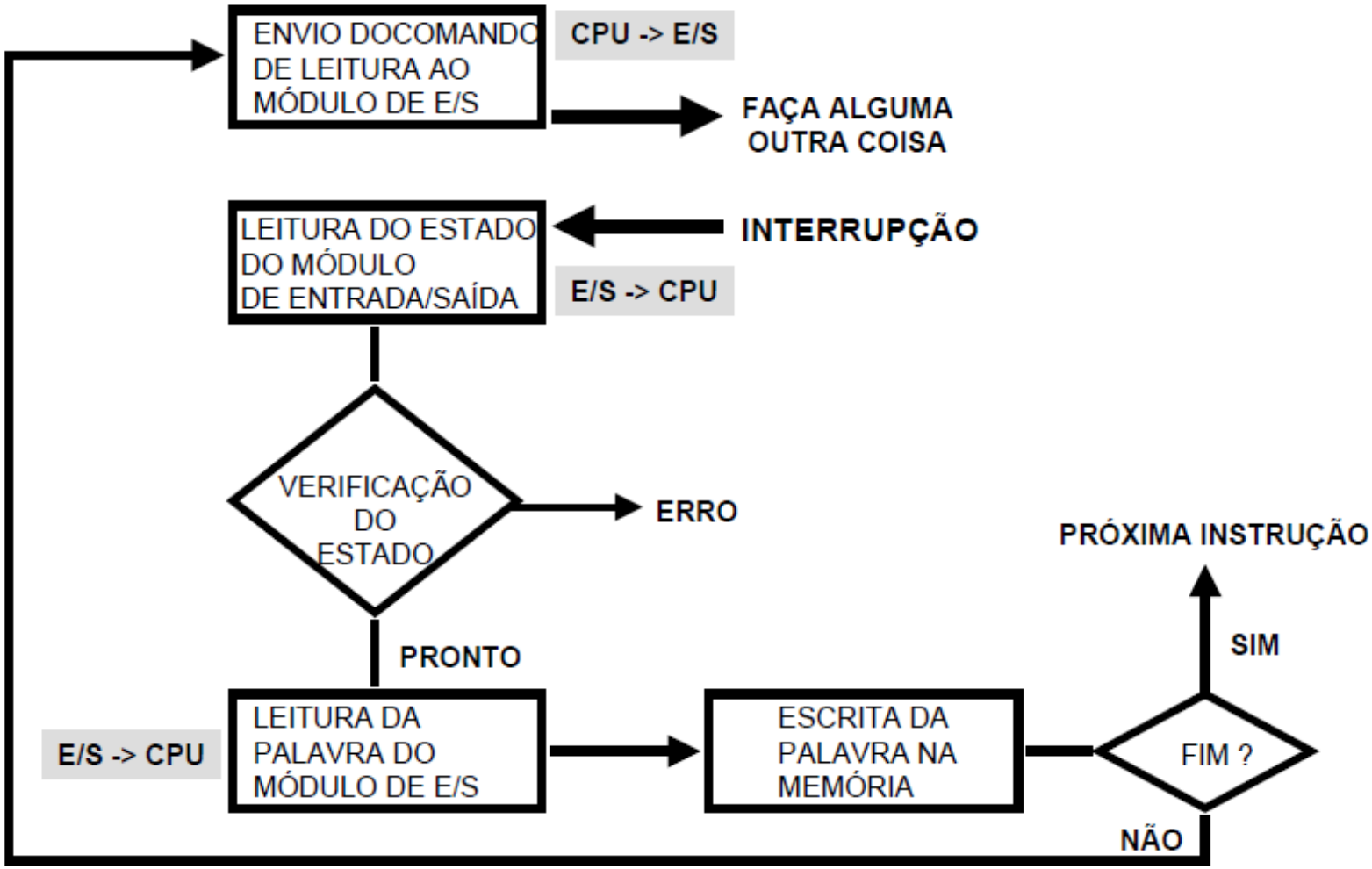
Operações de E/S

- Controlada por interrupção
  - O driver do controlador interrompe a CPU para avisar do término da operação de E/S
  - Toda a transferência entre dados e memória ainda exige a intervenção da CPU

1. Ao receber o sinal de leitura, o driver fica encarregado de ler os blocos do disco e armazená-los em registradores próprios
2. Em seguida, o driver sinaliza uma interrupção para a CPU
3. Quando a CPU atende a interrupção, os dados do controlador são transferidos para a memória principal
4. Ao final da transferência o sistema operacional devolve o controle ao programa interrompido

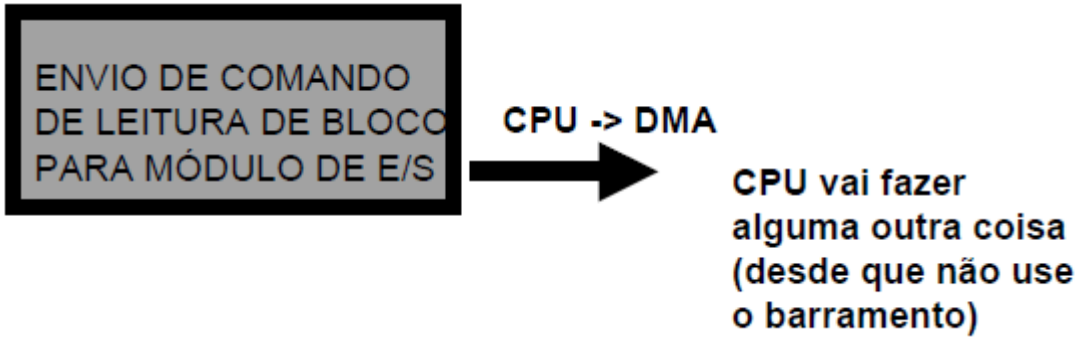
Operações de E/S

- Controlada por interrupção



Operações de E/S

- DMA (Direct Memory Access)
  - Neste método de E/S é necessário um hardware especial → módulo DMA
  - O módulo DMA pode controlar o fluxo de bits entre a memória e algum controlador sem a intervenção constante da CPU
  - A CPU só precisa intervir no início e no final da transferência



Operações de E/S

- DMA
  - Quando se deseja ler ou gravar blocos o processador envia um comando para o módulo DMA, com as seguintes informações:
    - O endereço do dispositivo de E/S
    - A posição inicial da memória onde os dados serão gravados ou lidos
    - O tamanho dos blocos
  - Com estas informações o controlador realiza a transferência entre os dispositivos de E/S e a memória principal
  - Quando a transferência estiver completa o módulo DMA envia um sinal para a CPU

DMA x Multiprogramação

Nos primeiros computadores, cada byte de dado lido ou escrito era tratado diretamente pela CPU (não existia DMA). Que implicações a ausência de DMA tem para a multiprogramação?

Interrupções

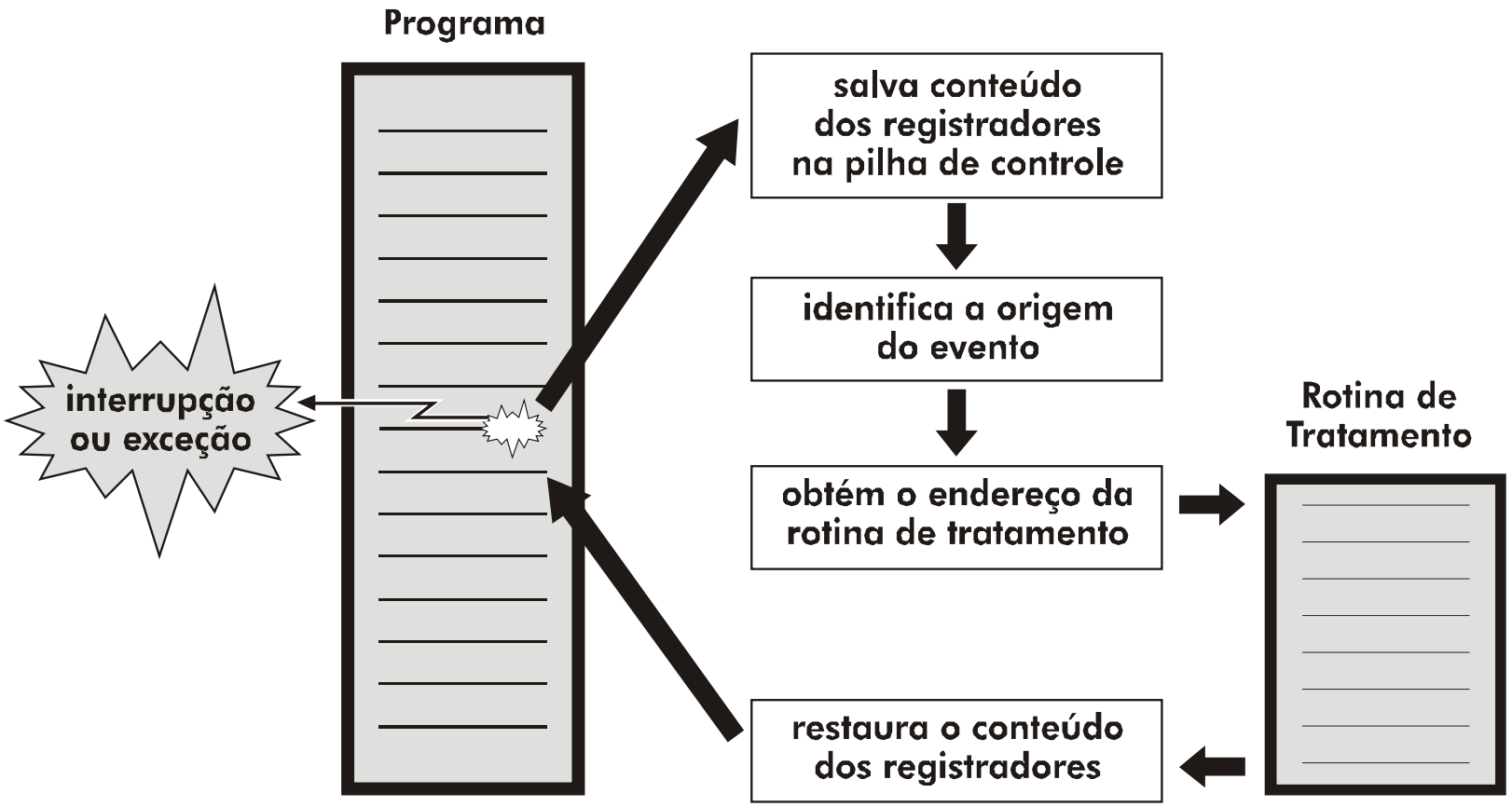
- Durante a execução de um programa podem ocorrer alguns eventos inesperados, ocasionados por um desvio forçado no fluxo de execução de um programa
  - Este tipo de evento é conhecido como interrupção
  - Pode ser consequência da sinalização de algum dispositivo de HW ou da execução de instruções do próprio programa
- A interrupção torna possível a implementação da concorrência
  - Concorrência da CPU: quando um processo perde o uso da CPU e depois retorna para continuar seu processamento, seu estado deve ser idêntico ao momento em que foi interrompido

Interrupções

- Tipos mais comuns

Programa	Causada por alguma condição gerada pelo próprio programa. Chamadas de exceções (*traps*) Ex. Divisão por zero ou overflow
Timer	Gerada pelo temporizador do processador. Permite, por exemplo, que o processador faça escalonamento de processos
E/S	Causada por um dispositivo de E/S. Quando, por exemplo, um periférico avisa a CPU que está pronto para transmitir algum dado
Falha de HW	Gerada por uma falha de hardware como falta de energia

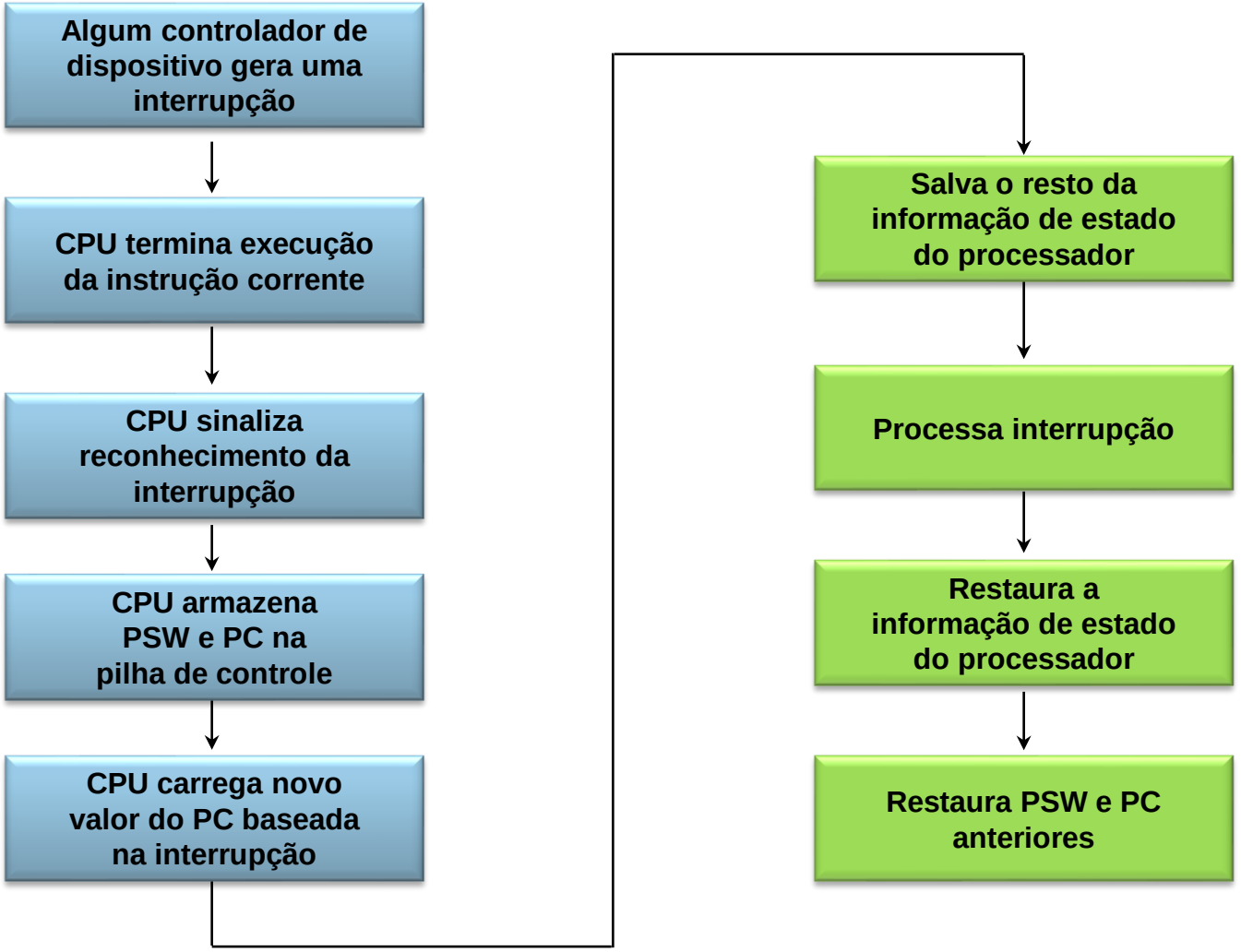
Ciclo de Tratamento de Interrupções



Interrupções X Exceções (*Traps*)

- Uma interrupção é gerada por um evento externo ao programa
- As exceções são causadas por alguma condição gerada pelo próprio programa
- Quando uma exceção é gerada, o programa é interrompido e desviado para uma rotina de tratamento de exceções
  - Para cada tipo de exceção existe uma rotina de tratamento adequada, que pode ser escrita pelo próprio programador
- Exceções são eventos síncronos e interrupções são assíncronos

Mecanismo de Interrupção



Interrupções Múltiplas

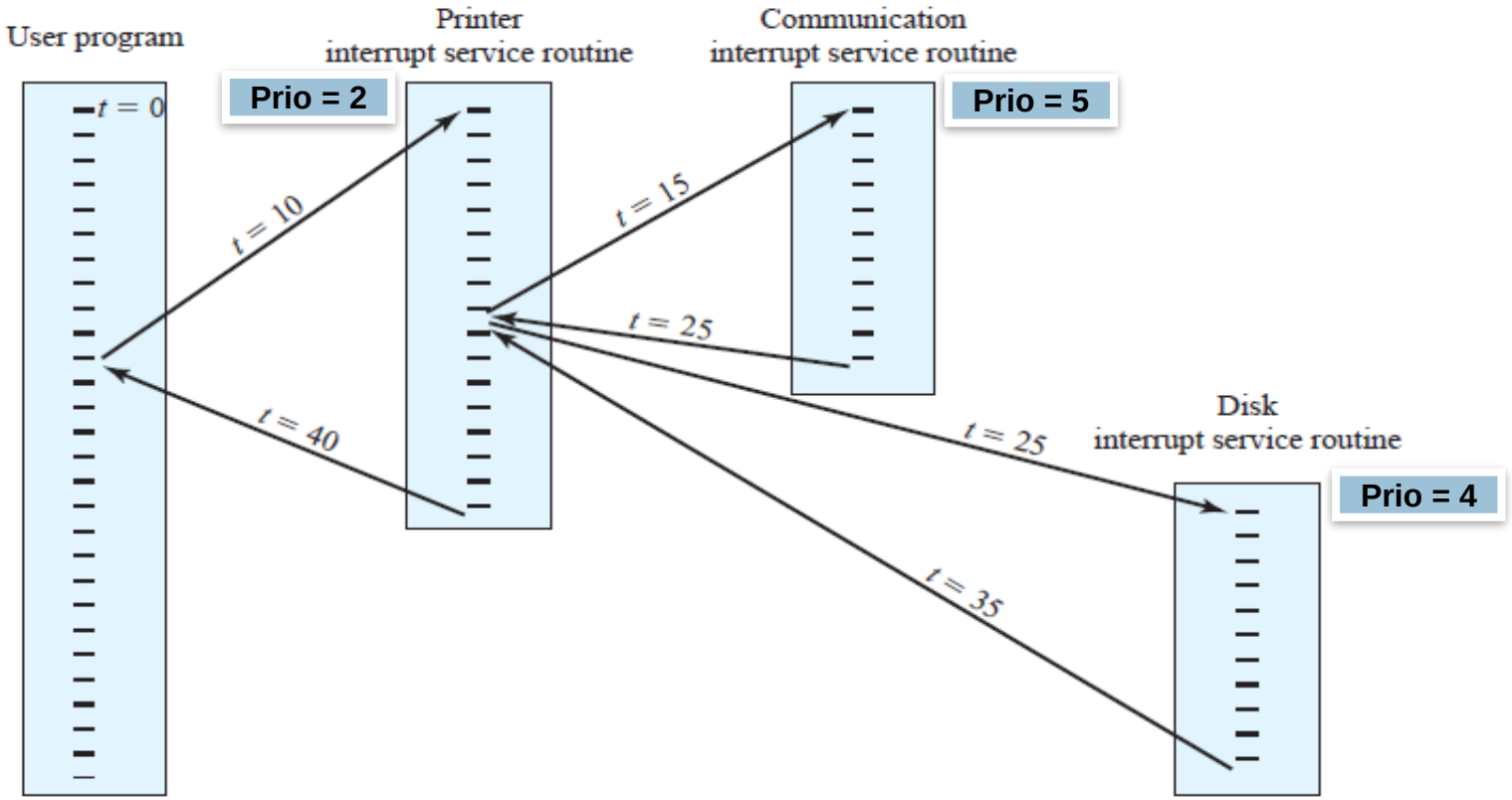
- Várias interrupções podem acontecer ao mesmo tempo
  - Por exemplo:
    - Um programa pode estar recebendo dados de uma linha de comunicação e imprimindo resultados ao mesmo tempo
    - A impressora irá gerar uma interrupção cada vez que ela completar uma operação
    - O controlador da linha de comunicação irá gerar uma interrupção cada vez que uma unidade de dado chegar
    - É possível ocorrer uma interrupção na comunicação enquanto uma interrupção da impressora está sendo processada
- O processador deve saber qual interrupção deve ser tratada primeiro

Interrupções Múltiplas

- (1ª Abordagem) Desabilita outras interrupções
  - Quando um programa está executando e uma interrupção ocorre, as demais interrupções são desabilitadas
  - Interrupções pendentes são verificadas ao fim do tratamento da interrupção corrente, e são tratadas sequencialmente
  - Quando terminar o tratamento, as interrupções são habilitadas
- Vantagem: simples implementação
- Desvantagem: não leva em consideração prioridades, nem tempos de execução críticos

Interrupções Múltiplas

- (2ª Abordagem) Definição de prioridades
  - Interrupções de menor prioridade podem ser interrompidas por outra de maior prioridade



Chamadas ao Sistema

Núcleo do SO

- O sistema operacional não é executado como uma aplicação sequencial
- Suas rotinas são executadas sem uma ordem definida
- O SO é formado por um conjunto de rotinas que “oferece” serviços aos usuários e suas aplicações
  - O conjunto de rotinas é chamado núcleo ou *kernel* do SO
- É o *kernel* que oferece os serviços básicos para todas as partes do sistema operacional

Núcleo do SO

- As principais funções do *kernel* são:
  - Tratamento de interrupções e exceções
  - Criação e eliminação de processos
  - Sincronização e comunicação entre processos
  - Escalonamento e controle dos processos
  - Gerência de memória
  - Gerência do sistema de arquivos
  - Gerência de dispositivos de E/S
  - Suporte a redes locais e distribuídas
  - Segurança do sistema

Modos de Acesso

- Os serviços do *kernel* são requisitados através de chamadas ao sistema
- As chamadas ao sistema são a porta de entrada para se ter acesso ao núcleo do SO
- Para que aplicações do usuário não causem danos a integridade do sistema, algumas instruções não podem ser executadas diretamente
  - Instruções privilegiadas só são executadas pelo SO
- 2 modos de acesso: modo usuário e modo *kernel* (ou monitor ou supervisor)

Modos de Acesso

- Especificação do modo do processador:
  - O modo de acesso é especificado por um bit de modo do registrador PSW
  - Exemplo:
    - Quando um processo faz uma chamada ao sistema, o bit de modo é modificado para privilegiado → Modo Kernel
    - Uma rotina específica do sistema operacional modifica o bit de modo

Modos de Acesso

- Instruções privilegiadas só podem ser executadas no modo *kernel*
- Quando o programa do usuário necessita executar uma instrução privilegiada é feita uma chamada ao sistema
- A chamada ao sistema altera o modo de acesso para o modo *kernel*
- O SO executa a instrução e depois retorna ao modo usuário
- Se o programa do usuário tentar executar uma instrução privilegiada, o programa é encerrado

Instruções Privilegiadas

- Quais das instruções abaixo só podem ser executadas em modo kernel?

A. Desativar todas as interrupções

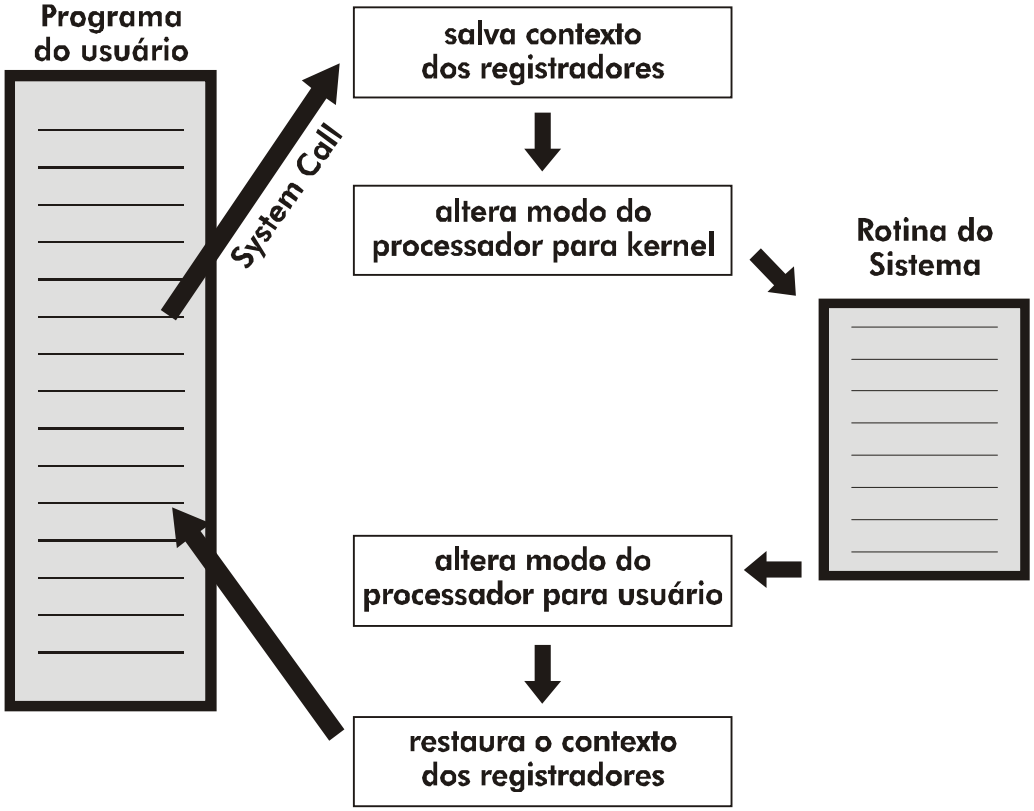
B. Ler o relógio de hora do dia

C. Configurar o relógio de hora do dia

D. Mudar o mapeamento de memória

Chamadas ao Sistema

- Para cada serviço existe uma chamada de sistema
- O SO tem seu próprio conjunto de chamadas ao sistema (nomes, parâmetros e forma de ativação específicas)



Chamadas ao Sistema

- As rotinas do sistema podem ser divididas em grupos, por exemplo:

Funções	Chamadas ao Sistema
Gerência de processos	- Criação e eliminação de processos - Sincronização e comunicação de processos - Alteração das características do processo
Gerência de Memória	- Alocação e liberação de memória
Gerência do Sistema de Arquivos	- Criação e eliminação de arquivos e diretórios - Alteração das características de arquivos e diretórios - Abrir e fechar arquivos - Leitura e gravação em arquivos
Gerência de Dispositivos	Alocação e liberação de dispositivos Operações de E/S em dispositivos

Chamadas de sistema - Exemplos

Tipo	Windows	Unix
Controle de Processos	CreateProcess()	fork()
	ExitProcess()	exit()
	WaitForSingleObject()	wait()
Manipulação de Arquivos	CreateFile()	open()
	ReadFile()	read()
	WriteFile()	write()
Manipulação de Dispositivos	SetConsoleMode()	ioclt()
	ReadConsole()	read()
	WriteConsole()	write()
Manipulação de Informações	GetCurrentProcessID()	getpid()
	SetTimer()	alarm()
	Sleep()	sleep()
Comunicação	CreatePipe()	pipe()
	CreateFileMapping()	shmget()
	MapViewOfFile()	mmap()
Proteção	SetFileSecurity()	chmod()
	InitializeSecurityDescriptor()	umask()
	SetSecurityDescriptorGroup()	chown()

Estrutura do SO

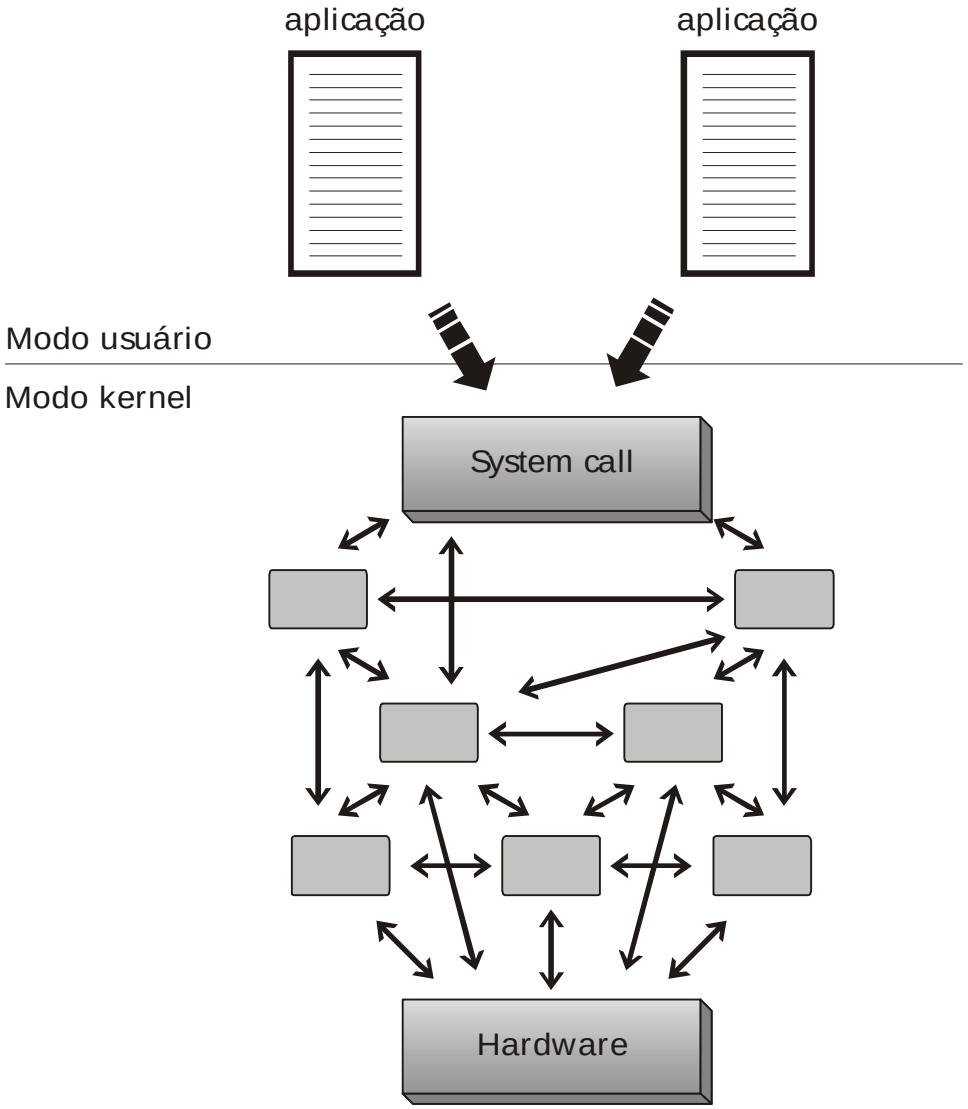
Estrutura do SO

- A estrutura do núcleo do sistema operacional e o inter-relacionamento entre os seus diversos componentes podem variar conforme a concepção do projeto
- O projeto do sistema depende muito do HW que será utilizado e do tipo de sistema que se deseja construir
- Alguns tipos de arquiteturas/estruturas:
  - Monolíticos
  - Em Camadas
  - Micronúcleo
  - Cliente-Servidor

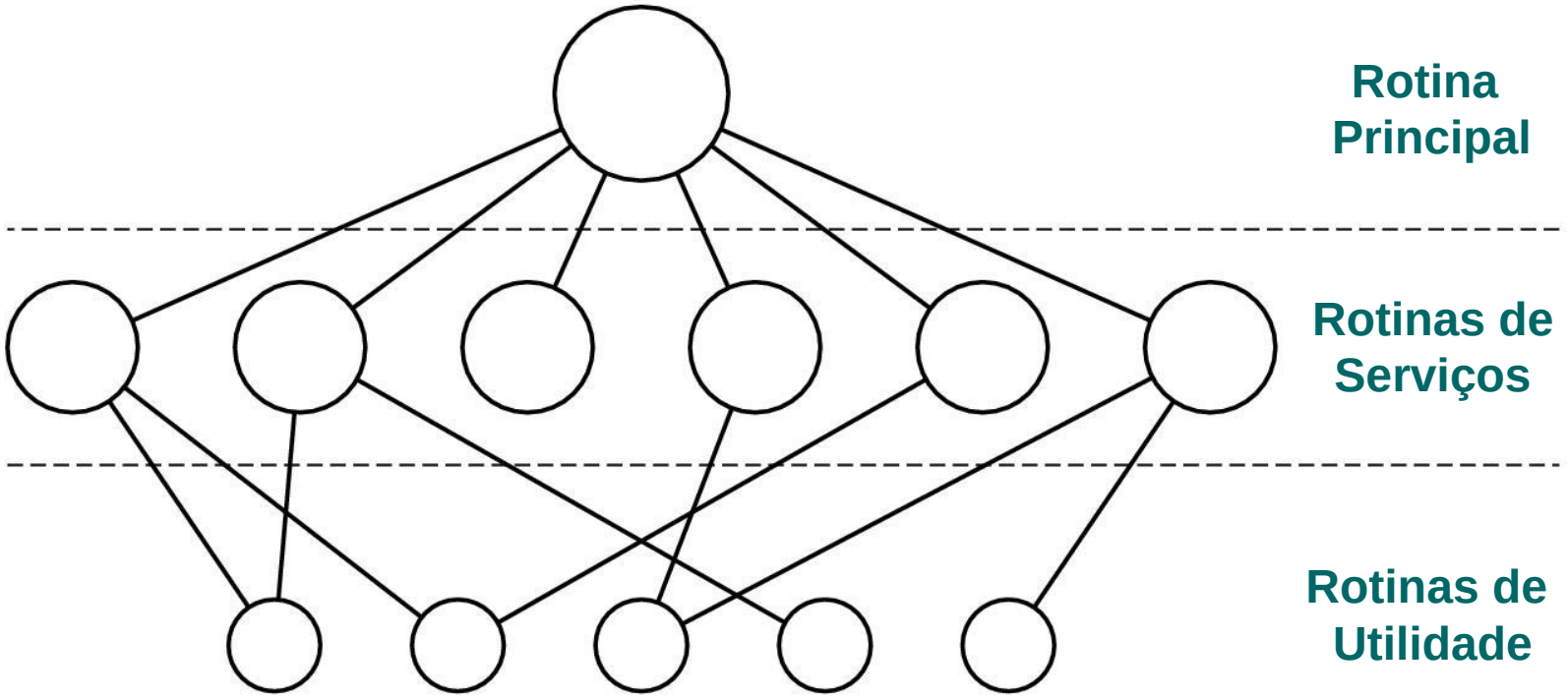
Estrutura do SO

- Sistemas Monolíticos
  - Estrutura o SO como um conjunto de políticas que podem interagir livremente umas com as outras
  - O SO é escrito como um conjunto de procedimentos, podendo cada procedimento chamar outro, sempre que necessário
  - Utilizando esta abordagem cada procedimento é compilado separadamente e depois são *linkados* em um único objeto
  - Mesmo em sistemas monolíticos pode existir um mínimo de estrutura (procedimento principal, procedimento de serviços e procedimentos utilitários)

Sistemas Monolíticos



Sistemas Monolíticos



Estrutura do SO

- Sistemas de camadas
  - Organiza o SO como uma hierarquia de camadas
  - Cada camada possui módulos que oferecem um conjunto de funções
  - Os módulos de uma camada só podem fazer referência aos módulos das camadas inferiores
  - A vantagem da estruturação em camadas é isolar as funções do SO
    - Cria uma hierarquia de níveis de modos de acesso

Sistema em Camadas

- O primeiro sistema em camadas

Camada	Função
5	Operador
4	Programas de usuário
3	Controle de Entrada e Saída
2	Comunicação operador-processo
1	Gerenciamento de memória
0	Alocação do processador e multiprogramação

Camada 0: fornecia os serviços para a multiprogramação básica da CPU. Realizava chaveamento de processos quando temporizadores expiravam ou quando ocorriam interrupções

Sistema em Camadas

- O primeiro sistema em camadas

Camada	Função
5	Operador
4	Programas de usuário
3	Controle de Entrada e Saída
2	Comunicação operador-processo
1	Gerenciamento de memória
0	Alocação do processador e multiprogramação

Camada 1: reservava espaço para os processos na memória principal. As camadas acima não precisavam se preocupar com tal tarefa. A camada 1 assegurava que as páginas eram trazidas para MP quando necessário.

Sistema em Camadas

- O primeiro sistema em camadas

Camada	Função
5	Operador
4	Programas de usuário
3	Controle de Entrada e Saída
2	Comunicação operador-processo
1	Gerenciamento de memória
0	Alocação do processador e multiprogramação

Camada 2: encarregava-se da comunicação entre cada processo e o console de operação.

Sistema em Camadas

- O primeiro sistema em camadas

Camada	Função
5	Operador
4	Programas de usuário
3	Controle de Entrada e Saída
2	Comunicação operador-processo
1	Gerenciamento de memória
0	Alocação do processador e multiprogramação

Camada 3: encarregava-se do gerenciamento dos dispositivos de E/S, armazenando temporariamente os fluxos de instrução que vinham ou iam para estes dispositivos.

Sistema em Camadas

- O primeiro sistema em camadas

Camada	Função
5	Operador
4	Programas de usuário
3	Controle de Entrada e Saída
2	Comunicação operador-processo
1	Gerenciamento de memória
0	Alocação do processador e multiprogramação

Camada 4: ficavam os programas do usuário.

Camada 5: ficava o processo operador do sistema.

Sistema em Camadas

- A vantagem principal é a simplicidade na construção e depuração
  - A primeira camada pode ser depurada sem qualquer preocupação com o restante do sistema (utiliza apenas o HW básico)
  - Quando a primeira camada é depurada, sua funcionalidade correta pode ser assumida, e assim por diante.
- A principal dificuldade é decidir em qual camada uma determinada funcionalidade estará presente.
- Cada camada acrescenta um custo adicional à chamada de sistema

Estrutura do SO

- Micronúcleo (MicroKernel)
  - A abordagem de Micronúcleo é tornar o núcleo do SO o mais simples possível
    - No sistema em camadas, muitas camadas entram no núcleo do SO
  - Estrutura o sistema operacional removendo todos os componentes não essenciais ao *kernel*, implementando-os como programas no nível de sistema e do usuário
  - Existe pouco consenso sobre quais serviços devem permanecer no *kernel*.
    - Fornece uma gerência mínima de processo e memória, além de facilidade de comunicação

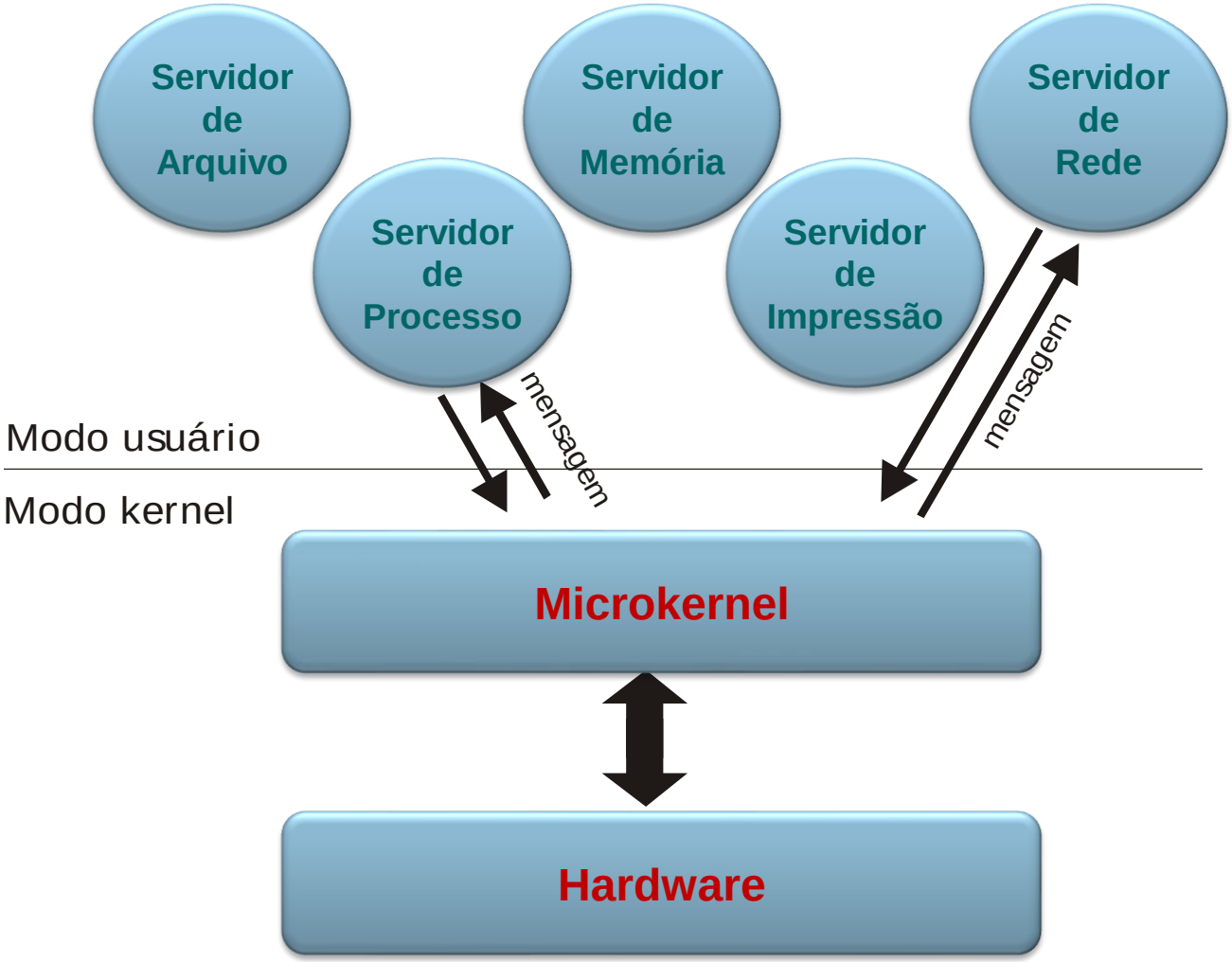
Estrutura do SO

- Micronúcleo
  - O objetivo é alcançar alta confiabilidade por meio da divisão do SO em módulos pequenos
  - Somente o micronúcleo é executado em modo kernel
  - Os demais processos são executados em modo usuário, onde um erro pode derrubar somente o processo e não todo o sistema
  - Usado em aplicações de tempo real, aviônica e industriais, por exemplo

Estrutura do SO

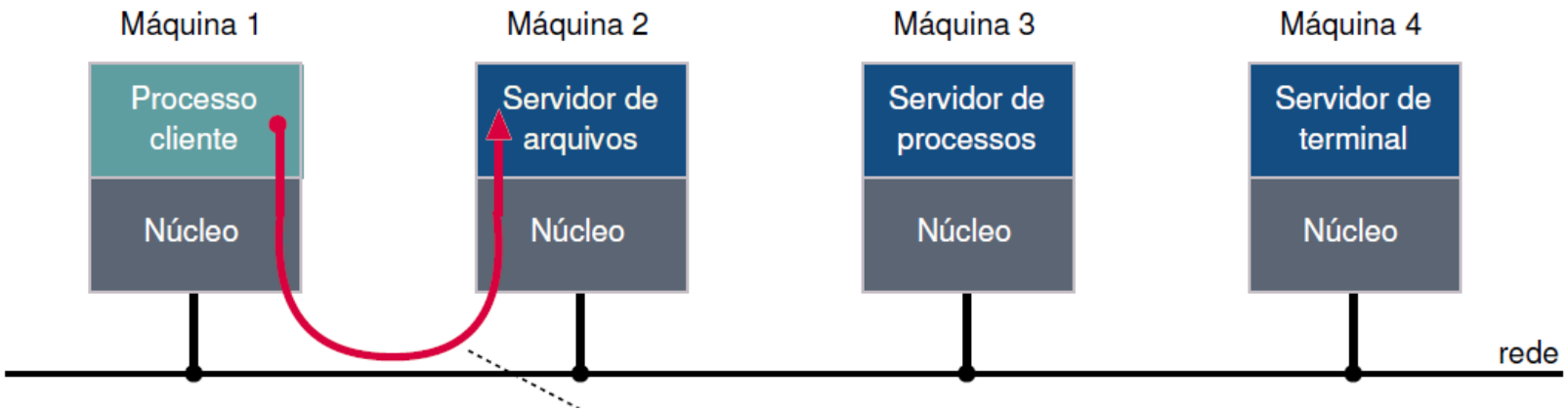
- Modelo Cliente-Servidor
  - Uma variação da idéia de micronúcleo é distinguir entre duas classes de processos: servidores e clientes
  - O sistema é dividido em processos, sendo cada um responsável por oferecer um conjunto de serviços: serviços de criação de processos, serviços de arquivos e serviços de memória.
  - No modelo cliente-servidor básico os processos são divididos em dois grupos:
    - Servidor implementa serviços específicos
    - Clientes requisitam serviços de um servidor

Modelo Cliente-Servidor



Estrutura do SO

- Modelo Cliente-Servidor
  - O serviço é solicitado pelo cliente enviando uma mensagem ao processo servidor
  - O servidor realiza o trabalho e envia a resposta através de outra mensagem
  - O núcleo do SO trata a comunicação entre clientes e servidores



Estrutura do SO

- Modelo Cliente-Servidor
  - A vantagem é a fácil adaptação para sistemas distribuídos
  - Normalmente o que se implementa é uma combinação do sistema de camadas com o modelo cliente-servidor
  - Nestes casos, o núcleo do sistema além de ser responsável pela comunicação entre processos clientes e servidores, passa a incorporar outras funções críticas:
    - gerência de E/S
    - gerência de memória
    - escalonamento de processos