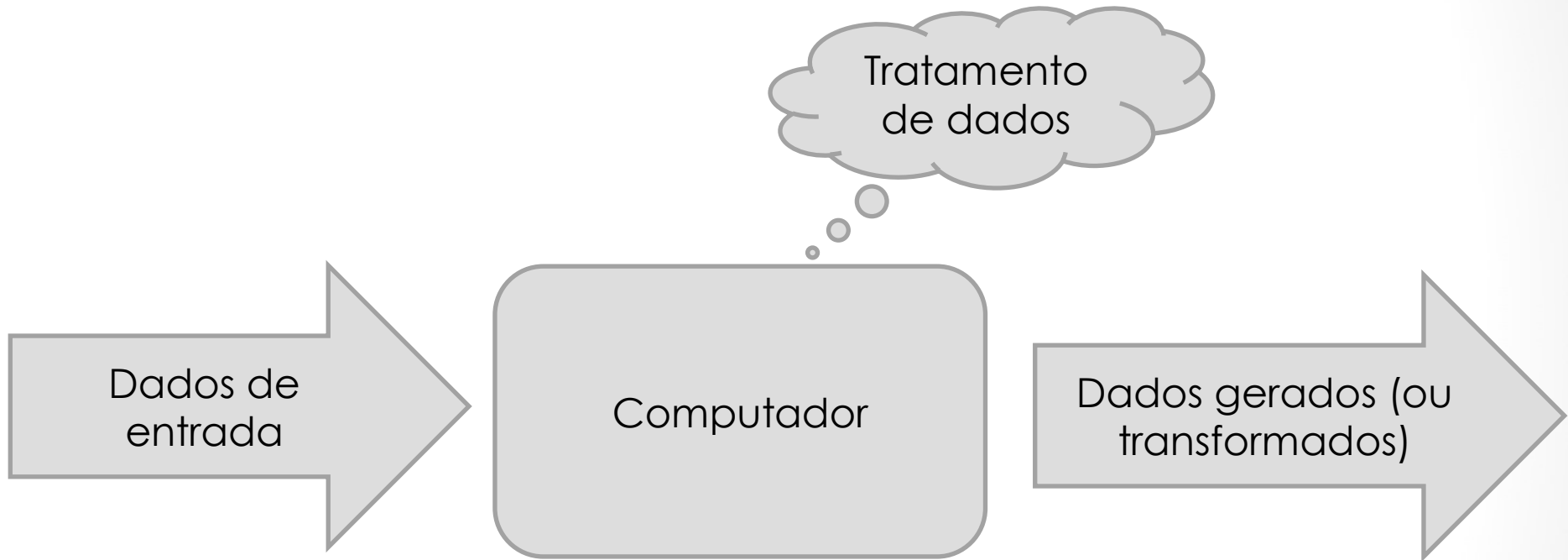


Juntando Todas as Partes

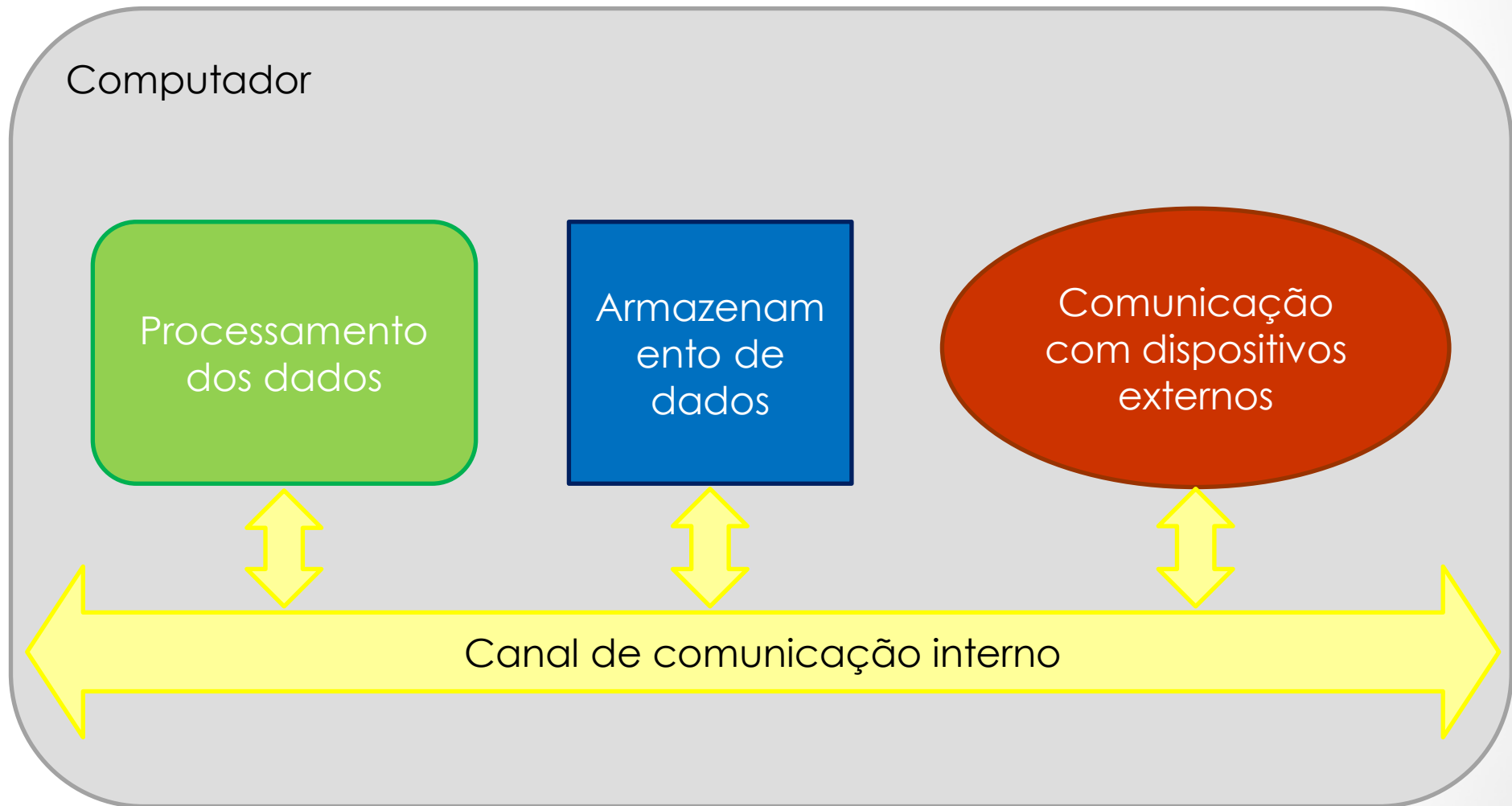
Gustavo Girão

Princípios de Funcionamento

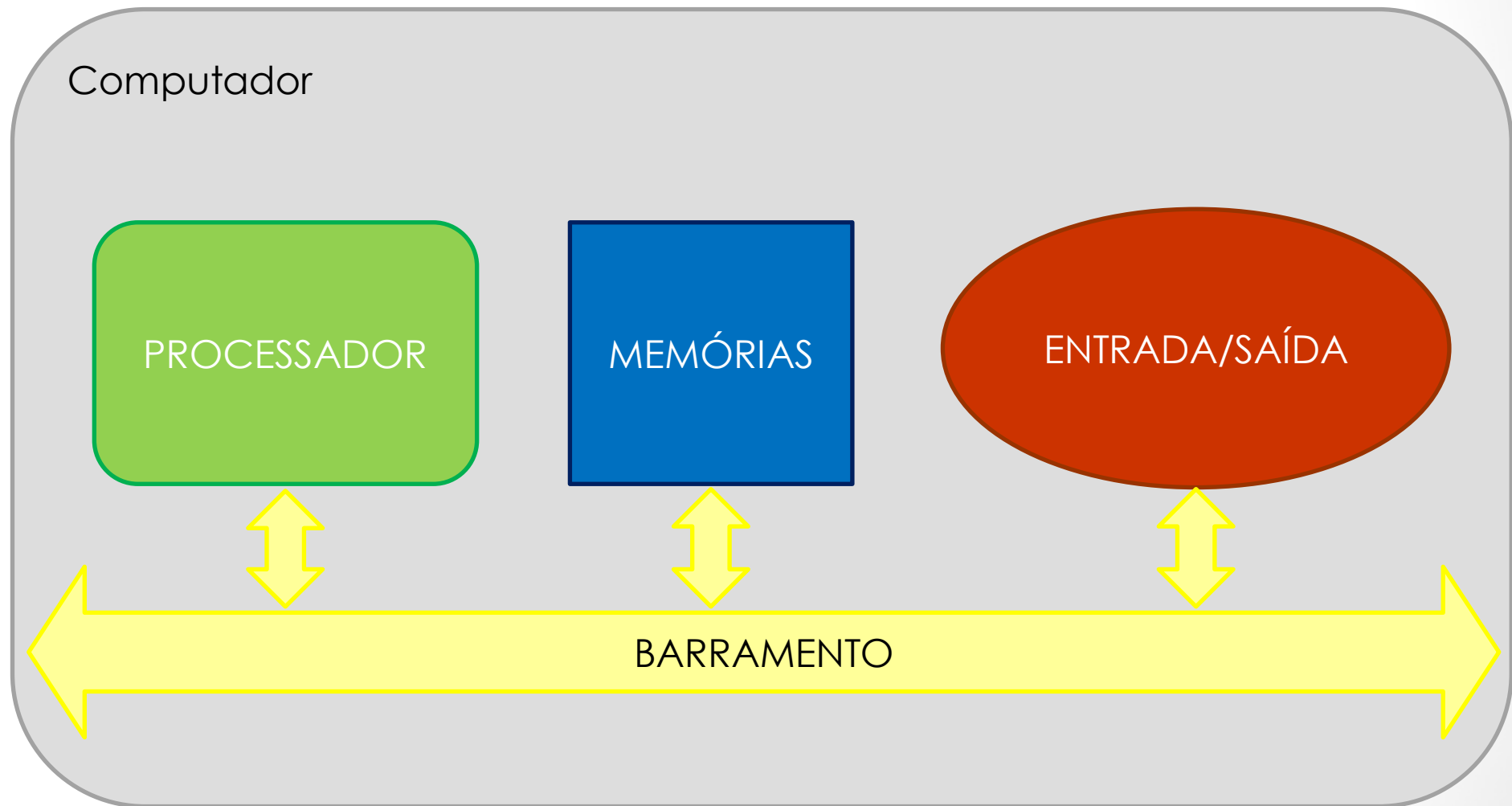
O que existe em um computador?



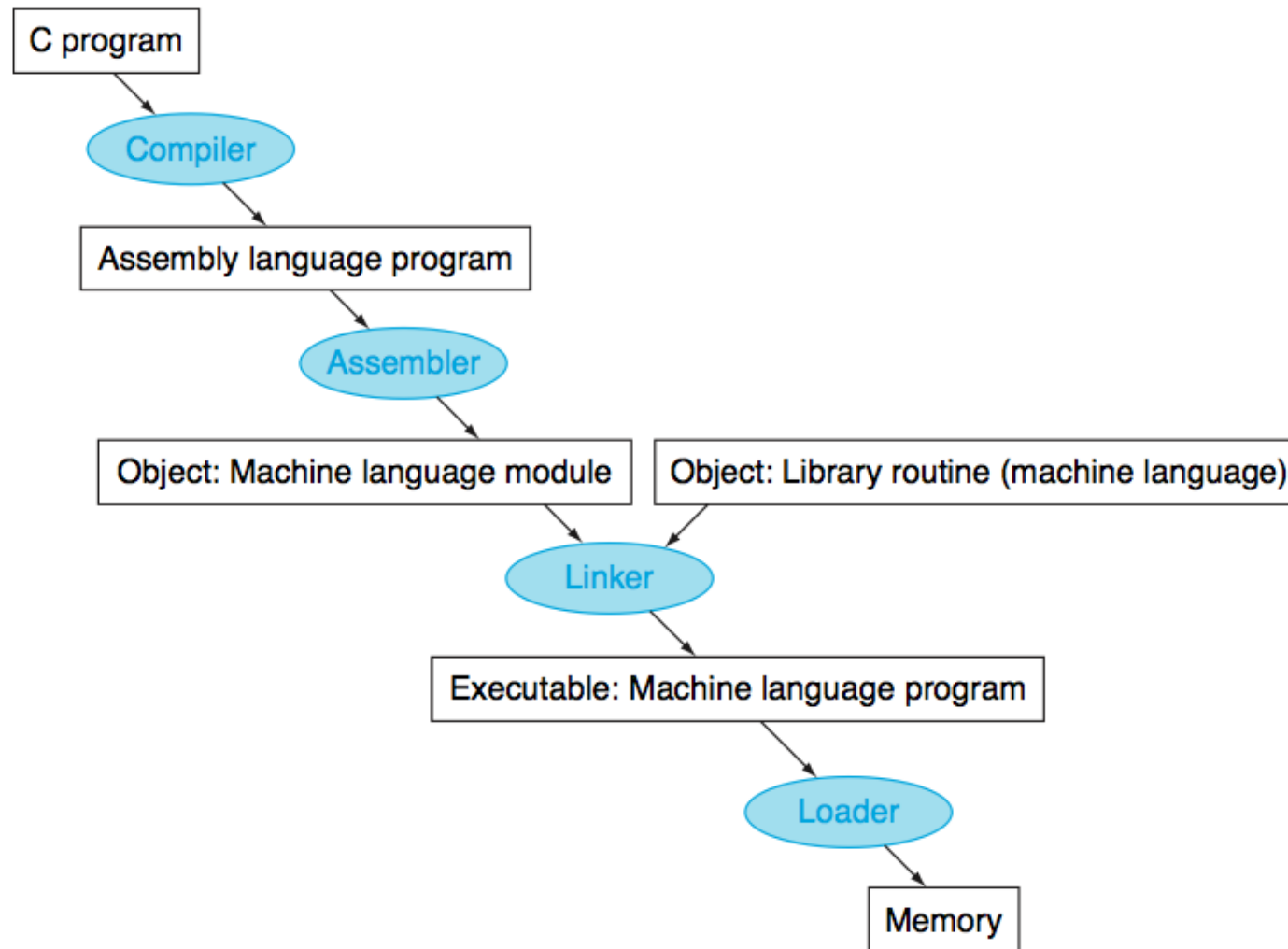
O que existe em um computador?



O que existe em um computador?



O que um computador precisa para executar?



O que precisa ser definido em projeto?

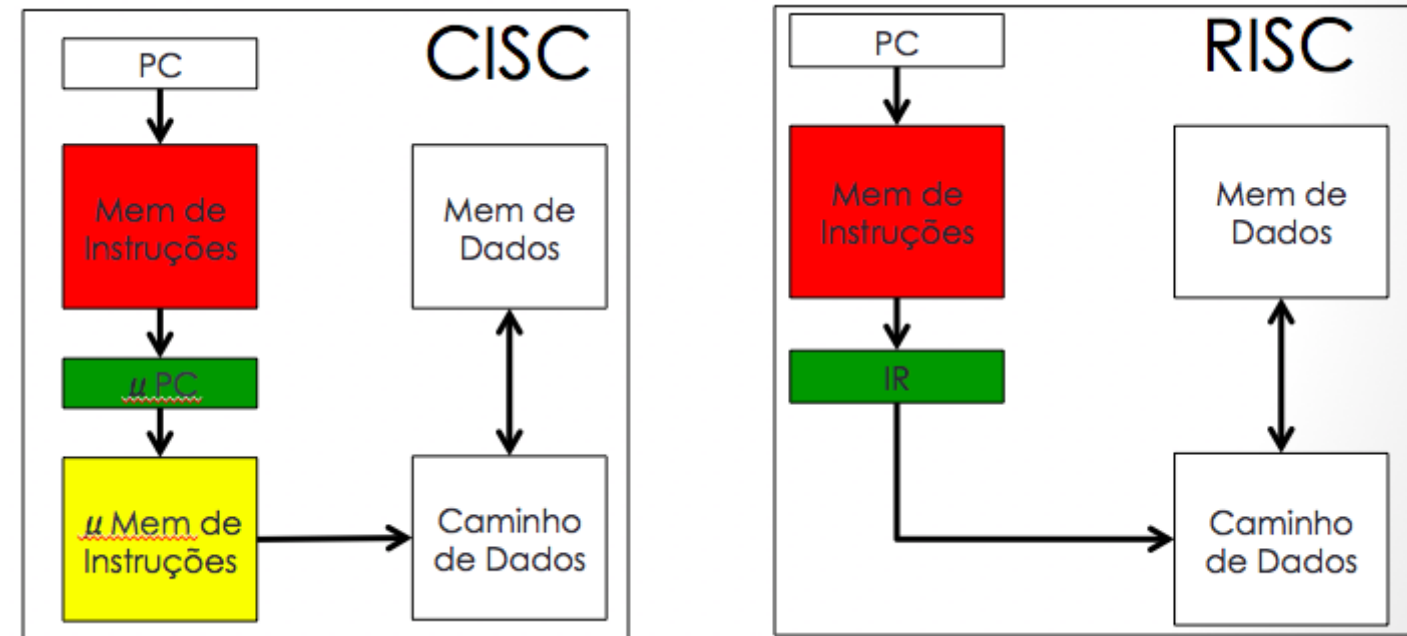
Formato de Instrução

opcode	Referência Operando Destino	Referência Operando Fonte 1	Referência Operando Fonte 2
x bits	y bits	z bits	k bits

- Modos de Ender.

- Direto
- Indireto
- Imediato
- Registrador
- Indireto de registrador
- Entre outros

Tipos de Processadores pelo Conjunto de Instr



SEQUÊNCIA DE FUNCIONAMENTO



Boot

- BIOS – Basic Input/Output System
 - Conjunto de programas embutidos em um ou mais chips
 - Controla o hardware e faz a interface entre o SO e hardware
 - Está armazenado em um memória ROM (Read Only Mem)
 - ✧ Memória não volátil!!!
- Ao ligar o computador, a (o) BIOS é executada (o):
 1. POST (Power on Self Test), que realiza testes no sistema
 2. SETUP, que habilita a configuração da placa-mãe e dos chipsets, como: data, hora, senha, drives, preferências de dispositivo de boot, sequência de boot, gerenciamento de energia, etc
 3. BOOTSTRAP LOADER, que lê o setor físico dos dispositivos (HD, CD, DVD, etc) em busca do MBR (*master boot record*).



Inicializando o SO

4. BOOTSTRAP LOADER (Bootloader) é carregado na memória principal
5. BIOS entrega controle de execução ao bootloader
6. Bootloader manda carregar SO na memória principal
7. Controle de execução passa para o SO

Observem que tudo que será executado precisa ser carregado na memória

Bootloader

É código!!!

Precisa estar na memória e ser executado pelo processador

```
.globl begtext, begdata, begbss, endtext, enddata, endbss
.text
begtext:
.data
begdata:
.bss
begbss:
.text

BOOTSEG = 0x07c0
INITSEG = 0x9000
SYSSEG  = 0x1000      | system loaded at 0x10000 (65536).
ENDSEG  = SYSSEG + SYSSIZE

entry start
start:
    mov ax, #BOOTSEG
    mov ds, ax
    mov ax, #INITSEG
    mov es, ax
    mov cx, #256
    sub si, si
    sub di, di
    rep
    movw
    jmpigo, INITSEG
go: mov ax, cs
    mov ds, ax
    mov es, ax
    mov ss, ax
    mov sp, #0x400      | arbitrary value >>512

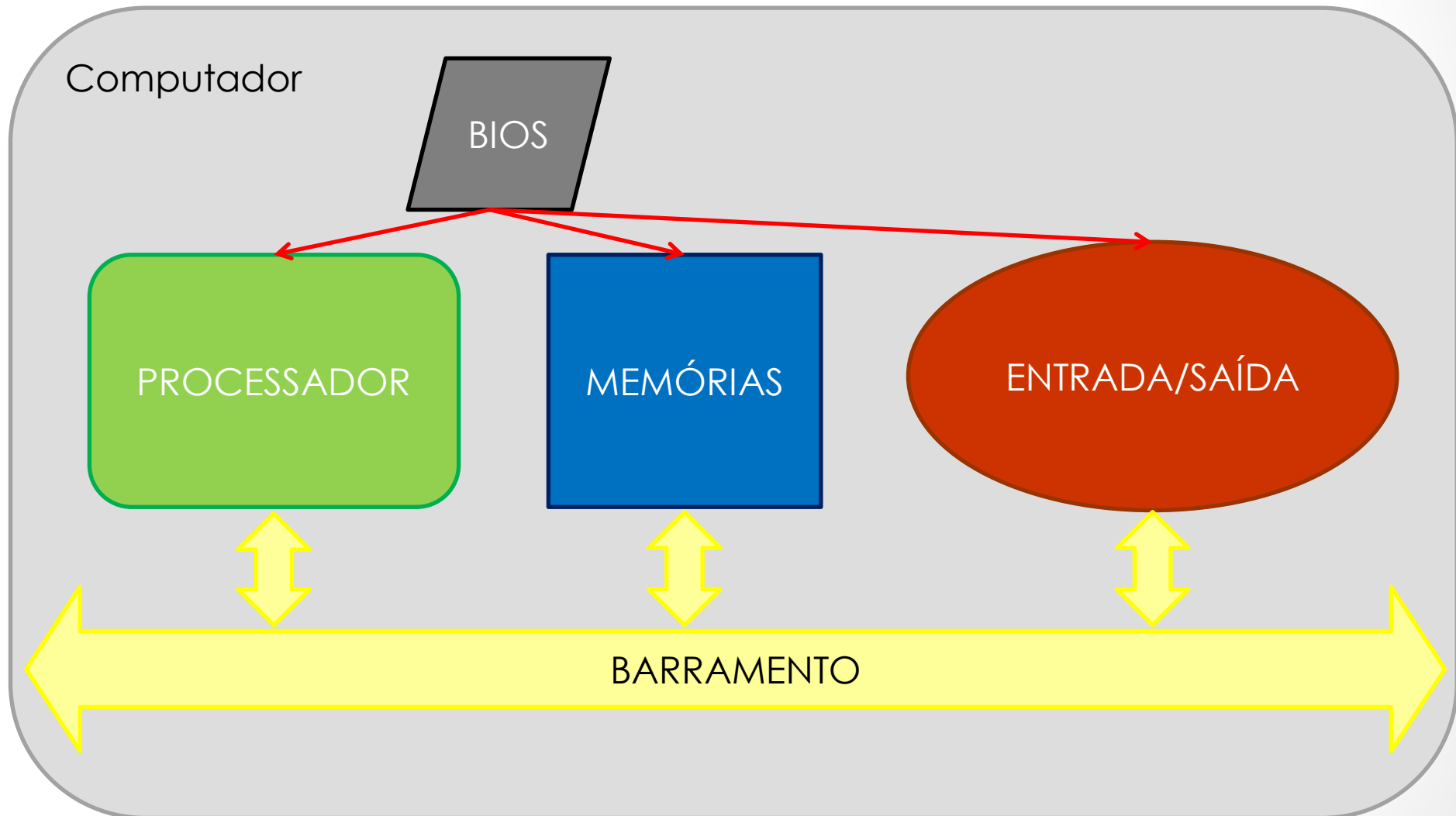
    mov ah, #0x03 | read cursor pos
    xor bh, bh
    int 0x10

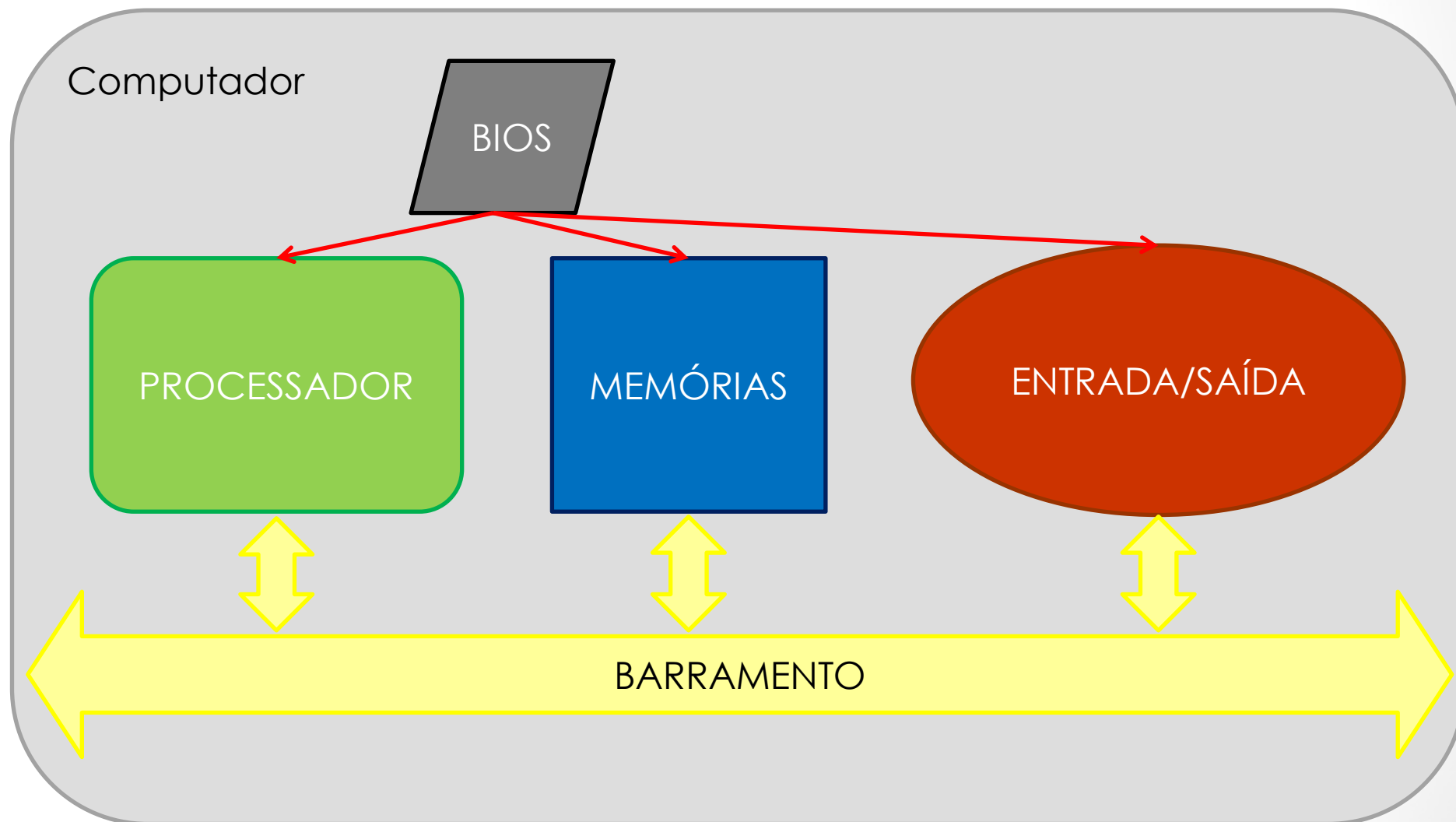
    mov cx, #24
    mov bx, #0x0007 | page 0, attribute 7 (normal)
```

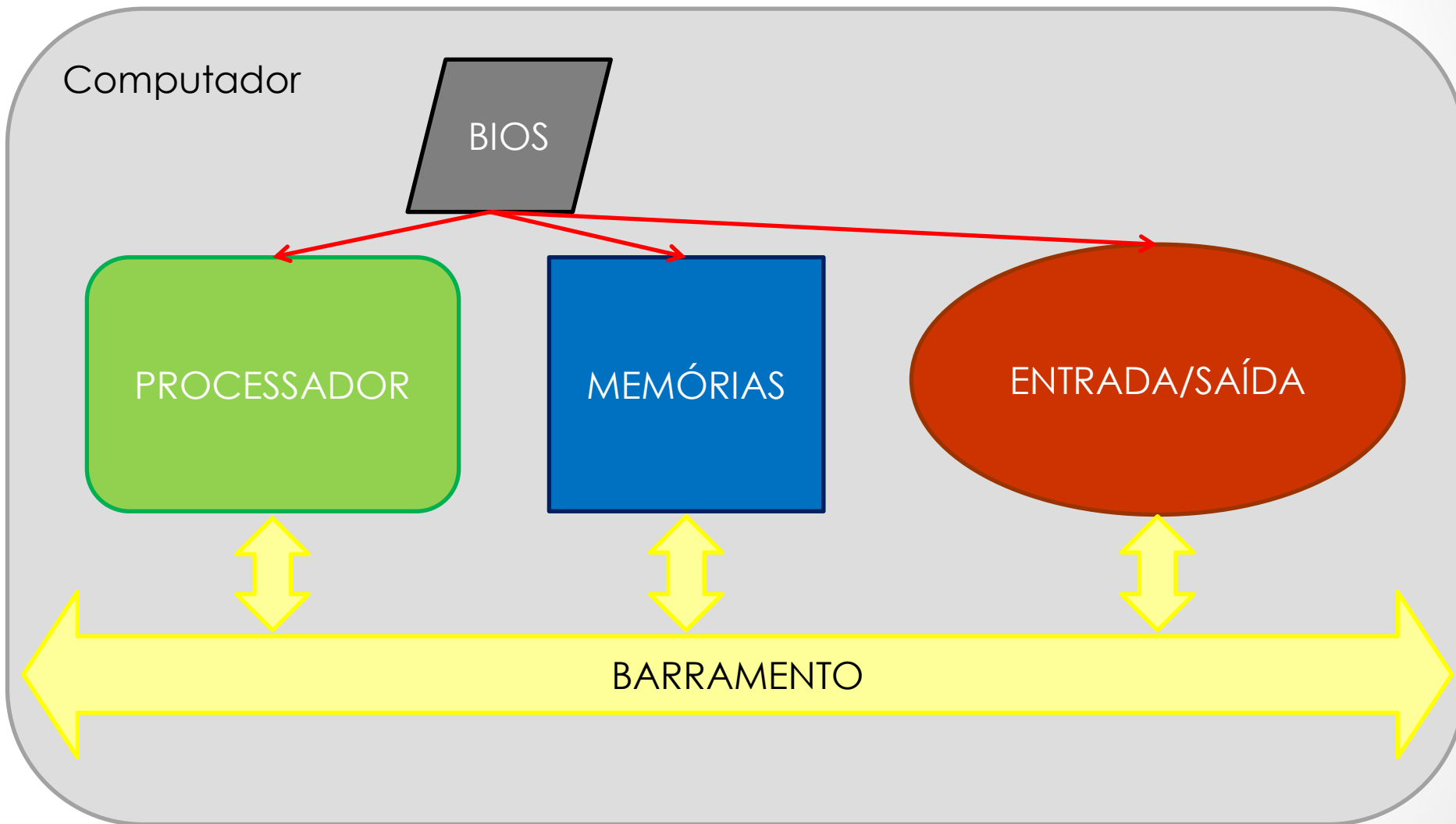
Executando Programas

- Após inicializado, programas de usuários, utilitários, sistema operacional são executados seguindo ordem de prioridade, justiça, políticas de substituição, etc.

Sequência de execução





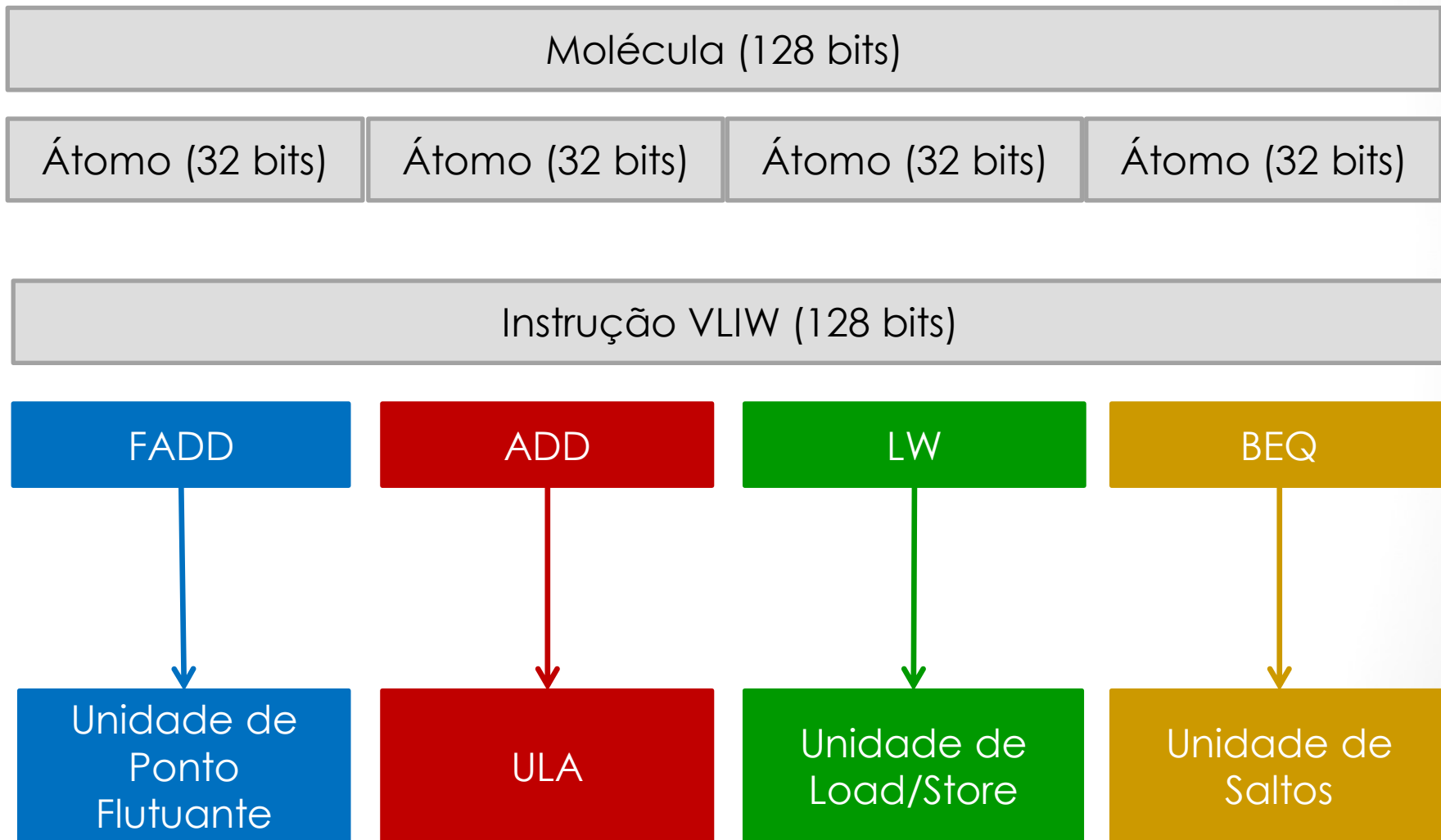


CONCEITOS AVANÇADOS DE PROCESSADORES

Conceitos avançados

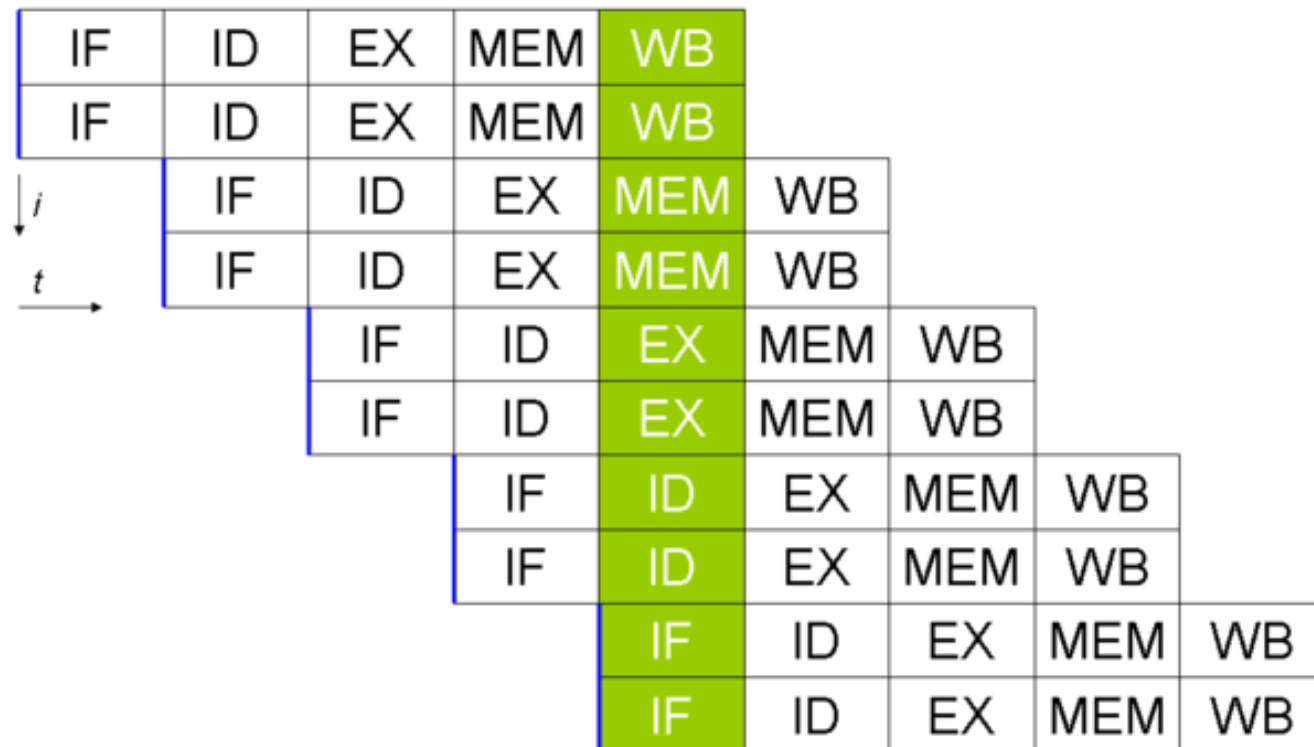
- Podemos obter $CPI < 1$?
 - Conceito de IPC – Instruções por ciclo
 - Como fazer isso?
 - ✧ Executando mais de uma instrução (despacho)
- Despacho de instruções
 - Estático
 - ✧ Superescalar
 - Dinâmico
 - ✧ VLIW
- Ações
 - Definir quais instruções serão despachadas
 - Garantir corretude

Very Large Instruction Word (VLIW)



Princípios da super-escalaridade

- Várias unidades de execução
- Várias instruções completadas simultaneamente em cada ciclo de relógio
- **Hardware** é responsável pela extração de paralelismo



VLIW vs Superscalar

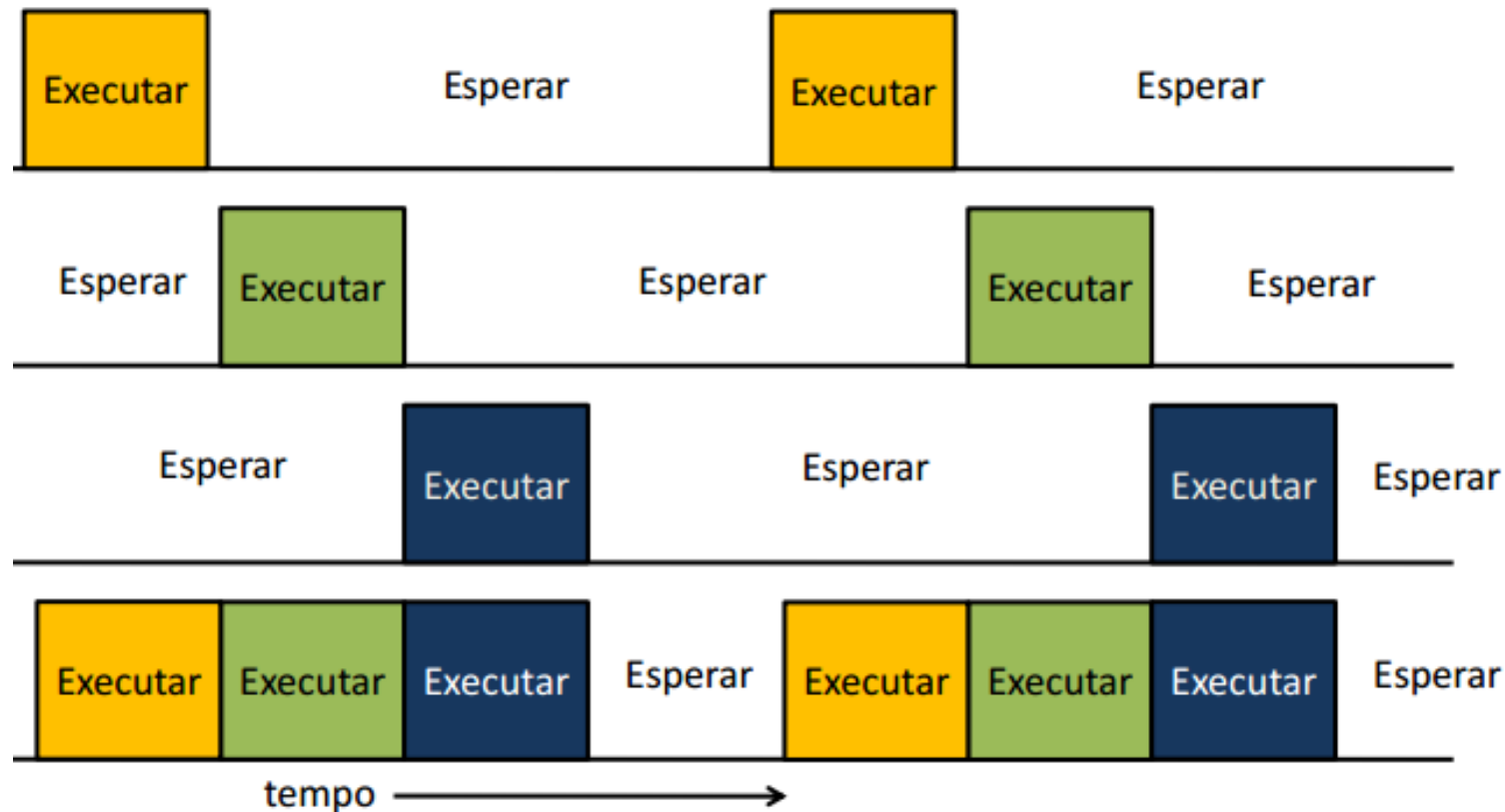
- Complexidade do hardware
 - VLIW é mais simples e mais fácil de escalar (decodificação)
 - Superscalares precisam encontrar paralelismo em tempo de execução
- Programação e complexidade do compilador
 - VLIW é mais complexo
 - ✧ Compiladores precisam encontrar todo o paralelismo
 - ✧ Lock step: conflitos podem fazer com que outras instruções fiquem paradas
 - Superscalares utilizam compilação comum

VLIW vs Superescalar

- Compatibilidade de software
 - VLIW precisa de recompilação
- Espaço em memória
 - VLIW precisa de maior largura de banda de instruções
 - Técnicas para evitar conflitos podem ser despendiosas
 - ✧ NOPs são um desperdício
 - ✧ Loop unrolling utiliza uma quantidade ainda maior de espaço em memória

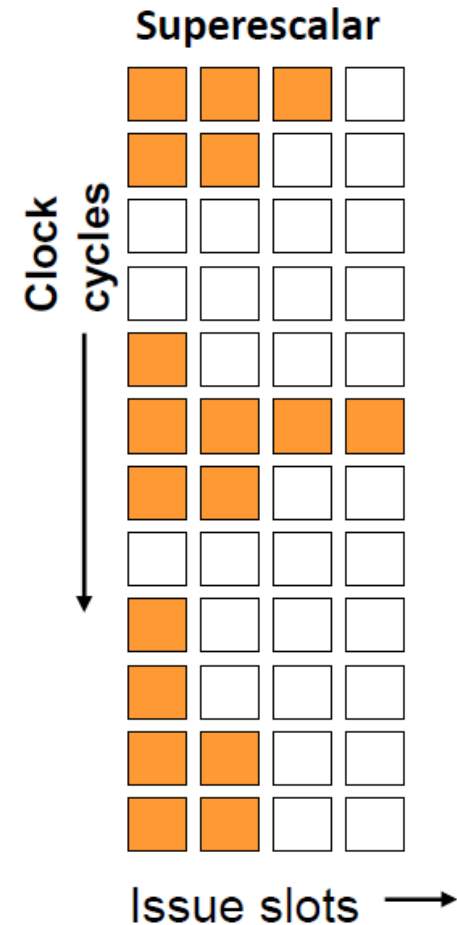
Multithreading?

- Multiprogramação!!
 - Thread Level Parallelism



Problemas com ILP

- Mesmo processadores superescalares não conseguem obter desempenho máximo
 - Dependências de dados
 - Acesso a memória
 - Latência de E/S
- Desperdício horizontal e vertical



Multithreading

- Abordagem Multithreading
 - Duas ou mais threads podem executar **virtualmente** de forma simultânea no mesmo processador
 - **Replicação** do estado que mantém as threads, porém com **compartilhamento** da maior parte dos recursos de hardware
 - Hierarquia de memória compartilhada

Multithreading

- Objetivos
 - Melhor utilização de **recursos** (cache e unidades de execução são compartilhadas entre threads)
 - Baixo consumo de área ($< 5\%$ da área do chip)
 - Redução do tempo de execução da aplicação

Onde é utilizado?

- Praticamente todos os processadores modernos de propósito geral utilizam
 - Na descrição nominal do processador destaca-se o número de threads suportadas.
 - **HyperThreading** é um caso de Multithreading proposto pela **Intel**
 - ✧ Disponível em processadores como Xeon, Pentium 4, Atom
 - ✧ Benchmarks da Intel apontam ganho médio de 30% de desempenho

Multithreading

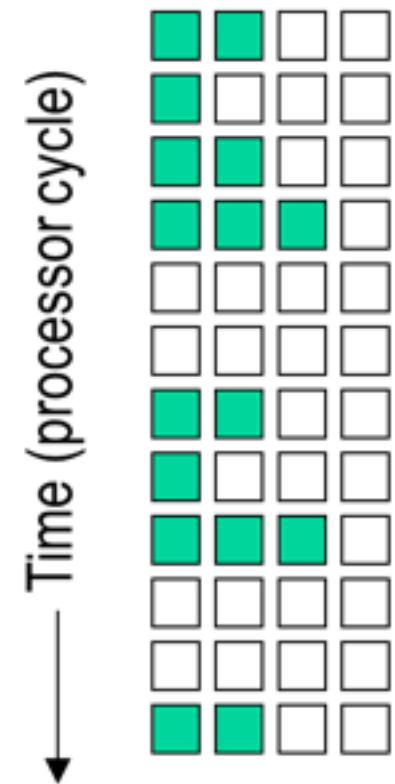
- Vantagens da abordagem
 - Quando uma thread interrompe muito (periférico, memória principal, HD), outras threads podem usar os recursos do processador. Reduzindo a ociosidade do processador.
 - Quando mais de uma thread executa sobre a mesma área de dados, elas podem compartilhar a cache, reduzindo o tempo de execução global do sistema.
- Desvantagens da abordagem
 - As threads podem interferir umas com as outras no compartilhamento de recursos de hardware como cache.
 - Complexidade de gerenciamento de execução

Multithreading

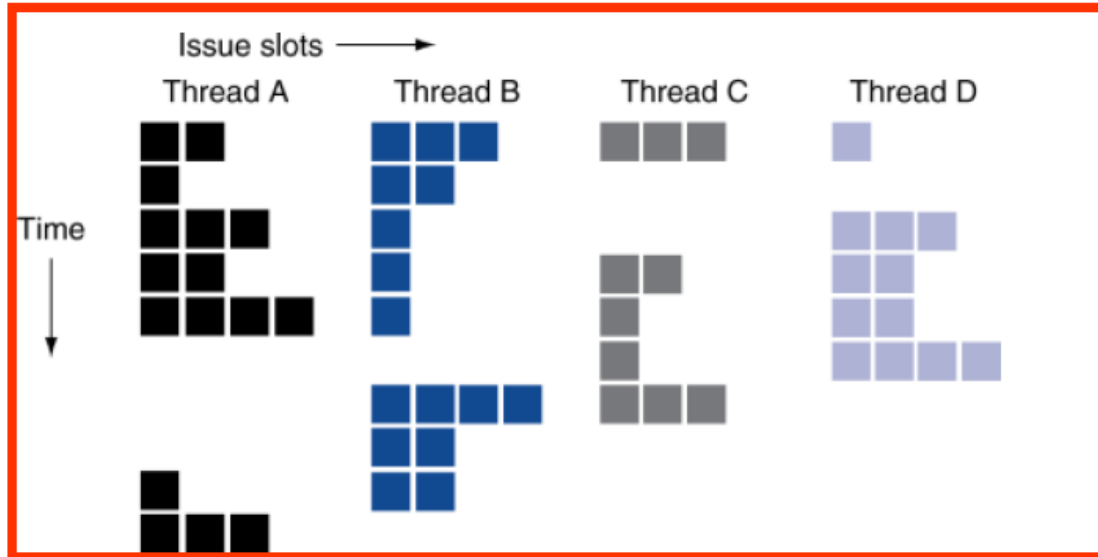
- Tipos de Multithreading
 - Granularidade grossa (coarse grained)
 - ✧ Block multithreading
 - Granularidade fina (fine grained)
 - ✧ Interleaved multithreading
 - Simultaneous multithreading (SMT)
 - ✧ Multithreading simultâneo

Sem Multithreading

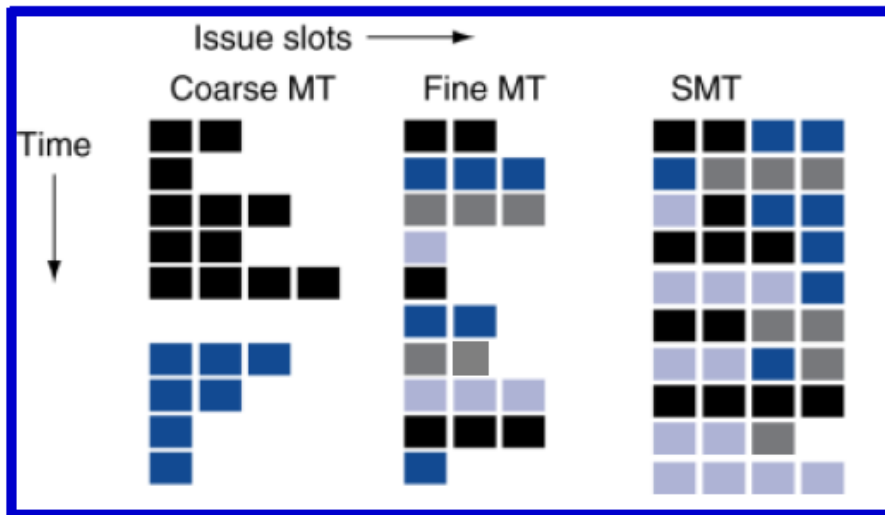
Superscalar



Exemplo



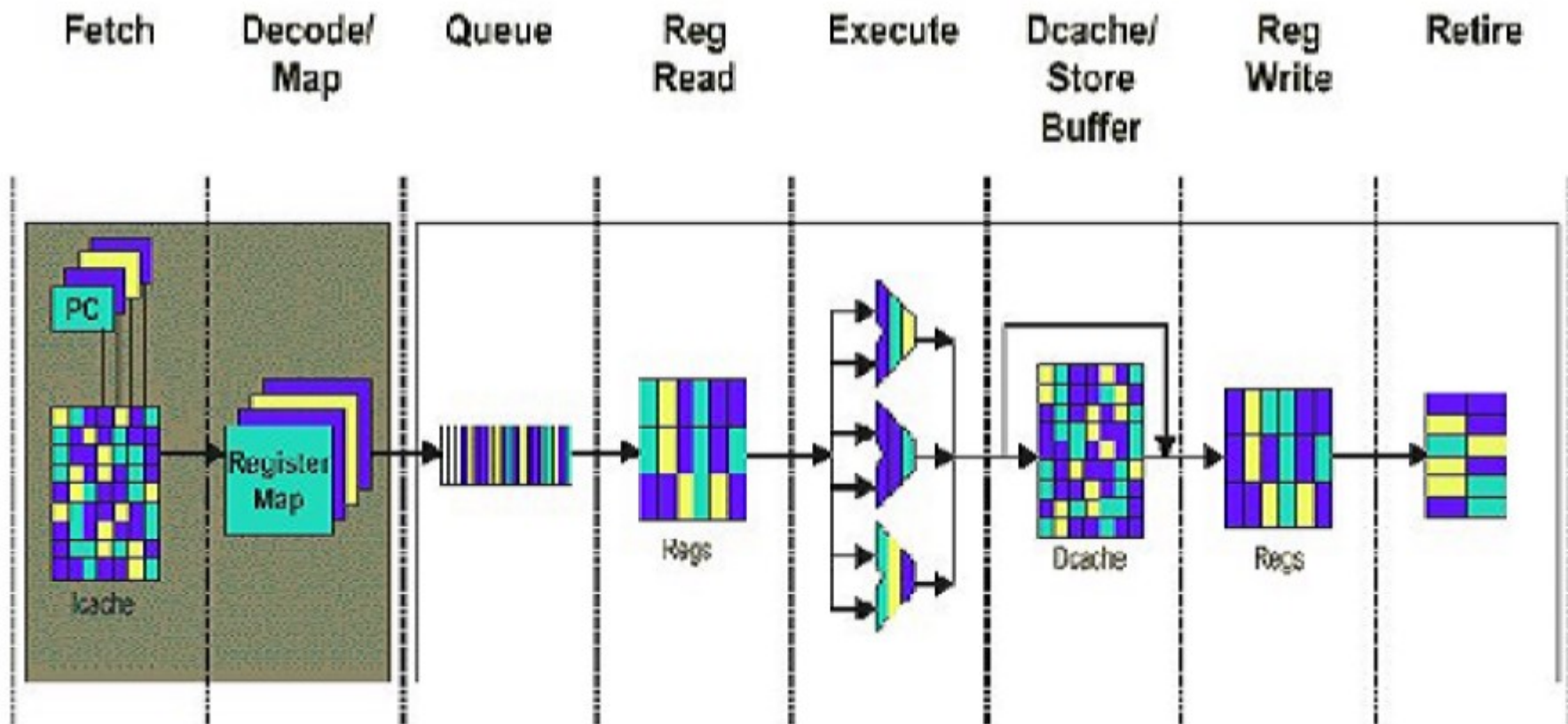
4 threads executando **isoladamente** em um processador superescalar de grau 4 sem multi-threading



4 threads executando conjuntamente em um processador superescalar de grau 4 com multi-threading utilizando diferentes estratégias de multi-threading

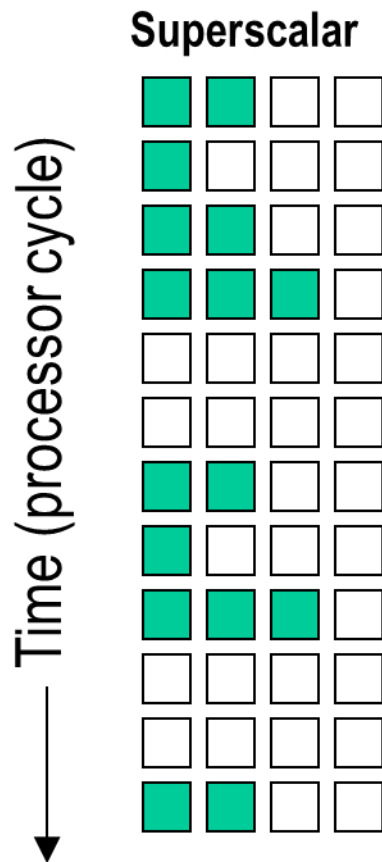
Multithreading

- Processador com Simultaneous Multithreading



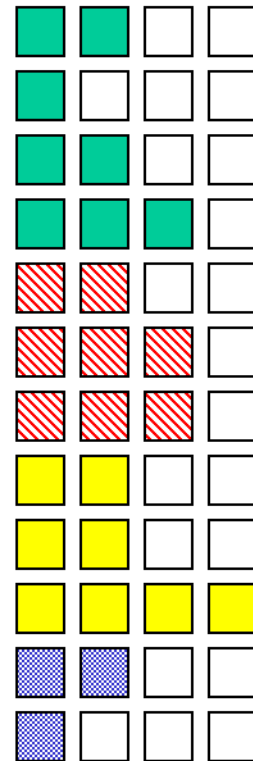
Tipos de Multithreading

Sem Multithreading

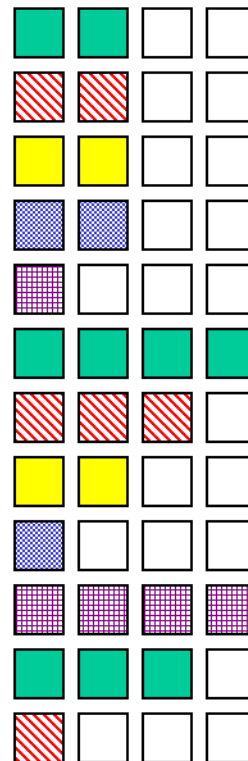


Com Multithreading

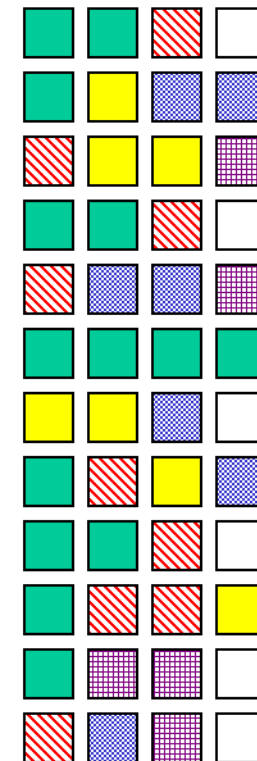
Coarse-Grained



Fine-Grained



Simultaneous Multithreading



Thread 1



Thread 2



Thread 3



Thread 4



Thread 5



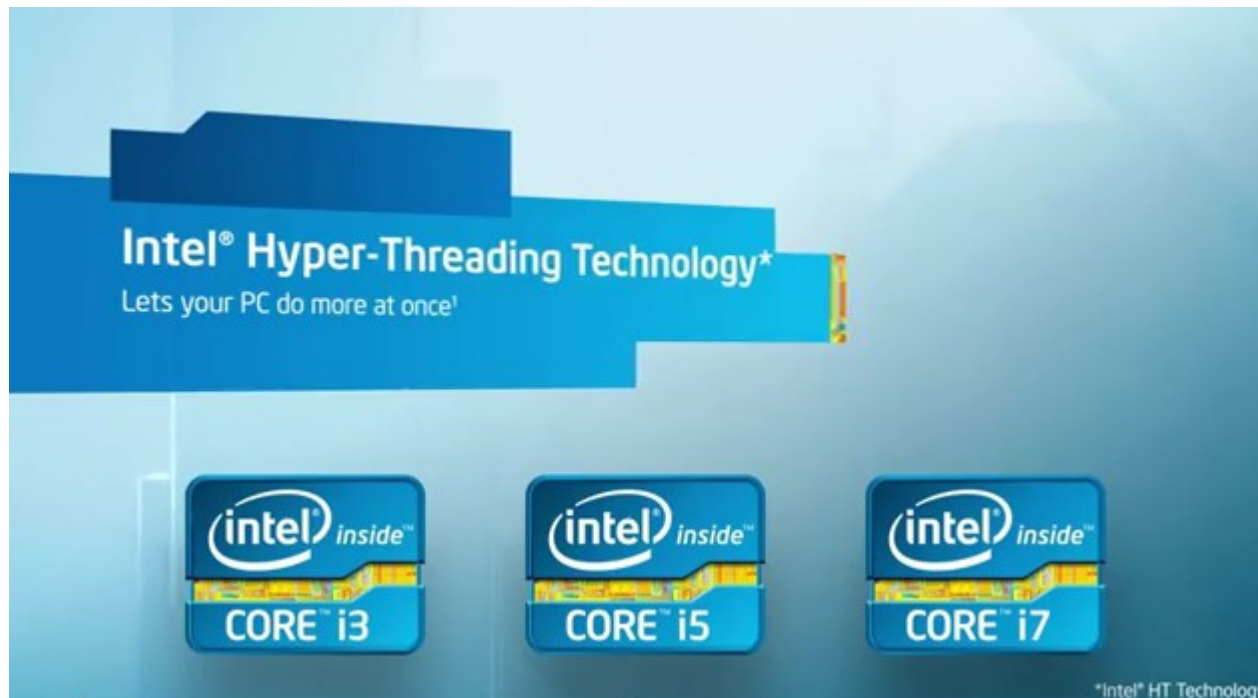
Idle slot

Simultaneous Multithreading (SMT)

- Intel – Hyper-Threading Technology

Para saber mais:

<http://www.intel.com.br/content/www/br/pt/architecture-and-technology/hyper-threading/hyper-threading-technology.html>





Intel® Core™ i7-6700K Processor (8M Cache, up to 4.20 GHz)

- Specifications ^
- Essentials
- Performance
- Supplemental Information
- Memory Specifications
- Graphics Specifications
- Expansion Options
- Package Specifications
- Advanced Technologies
- Intel® Data Protection Technology
- Intel® Platform Protection Technology
- Compatible Products ▾
- Benchmarks ▾
- Ordering / sSpecs / Steppings ▾

Specifications	
- Essentials	
Processor Number	i7-6700K
Status	Launched
Launch Date	Q3'15
Lithography	14 nm
Recommended Customer Price	\$339.00 - \$350.00
- Performance	
# of Cores	4
# of Threads	8
Processor Base Frequency	4.00 GHz
Max Turbo Frequency	4.20 GHz
Cache	8 MB SmartCache
Bus Speed	8 GT/s DMI3
# of QPI Links	0
TDP	91 W

Compare +

Related Products

6th Generation Intel® Core™ i7 Processors

Intel® Core™ i7-6700 Desktop Processor Series

Products formerly Skylake

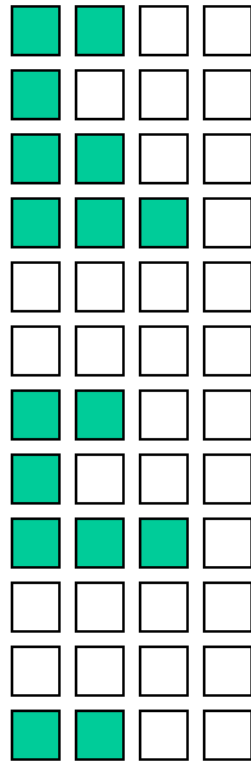


Learn how Intel is pursuing conflict-free technology. >

















































Quick Links

- Export Full Specifications >
- Find Compatible Boards >
- Optimized Game Settings >
- Support Overview >
- Search Distributors >

Time (processor cycle)



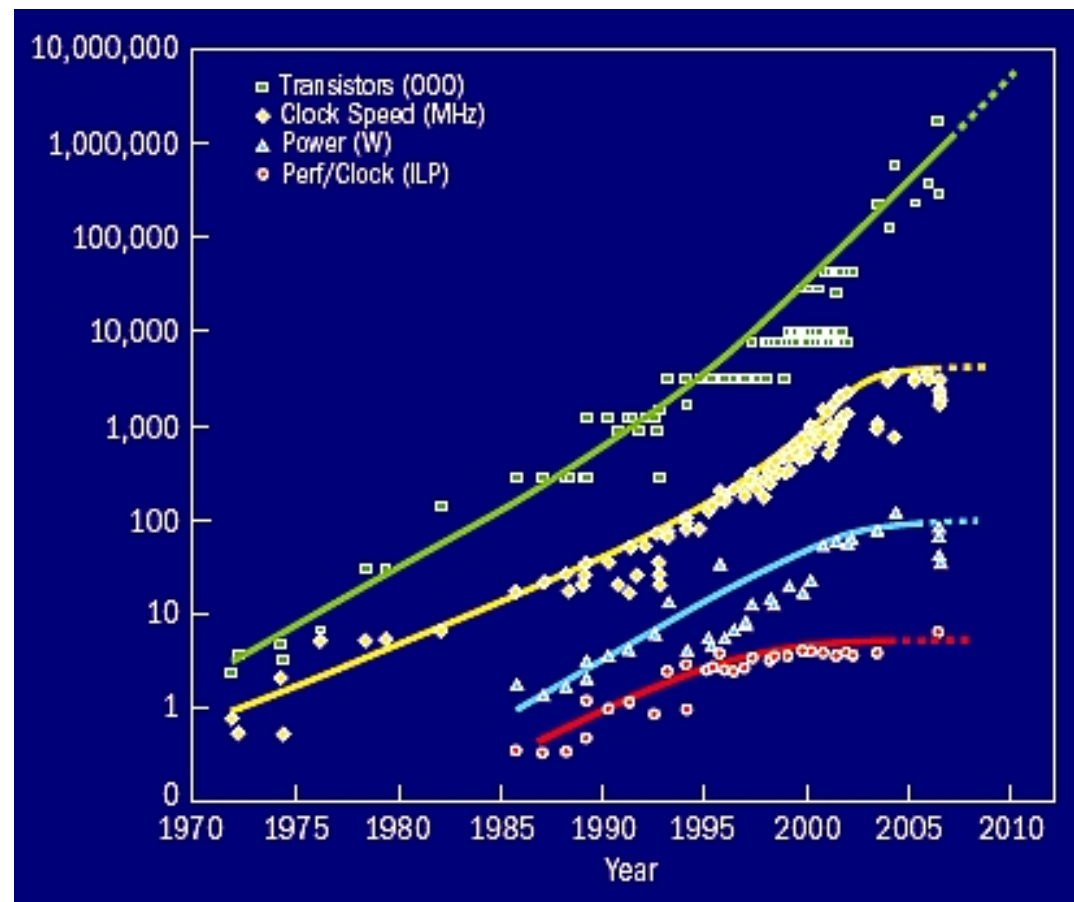
■	■	□	□
■	□	□	□
■	■	□	□
■	■	■	□
▨	▨	□	□
▨	▨	▨	□
▨	▨	▨	□
■	■	□	□
■	■	□	□
■	■	■	■
▩	▩	□	□
▩	□	□	□

□ Idle slot

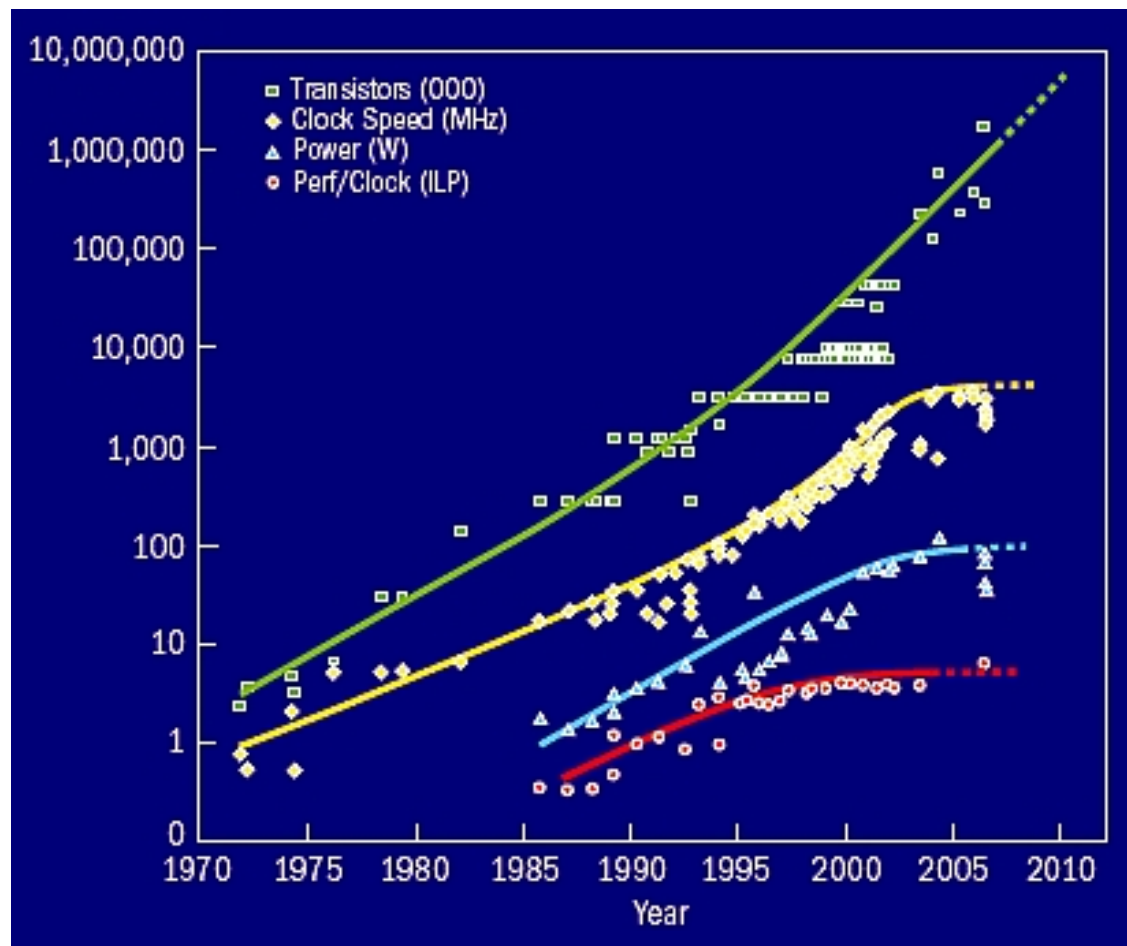
Introdução

- Limites de soluções de aumento de desempenho
 - Parede de ILP
 - ✧ Dependências de dados verdadeiras
 - ✧ Superescalar -> em tempo de execução -> Consumo de área e potência
 - ✧ VLIW -> em tempo de compilação -> Compatibilidade de software



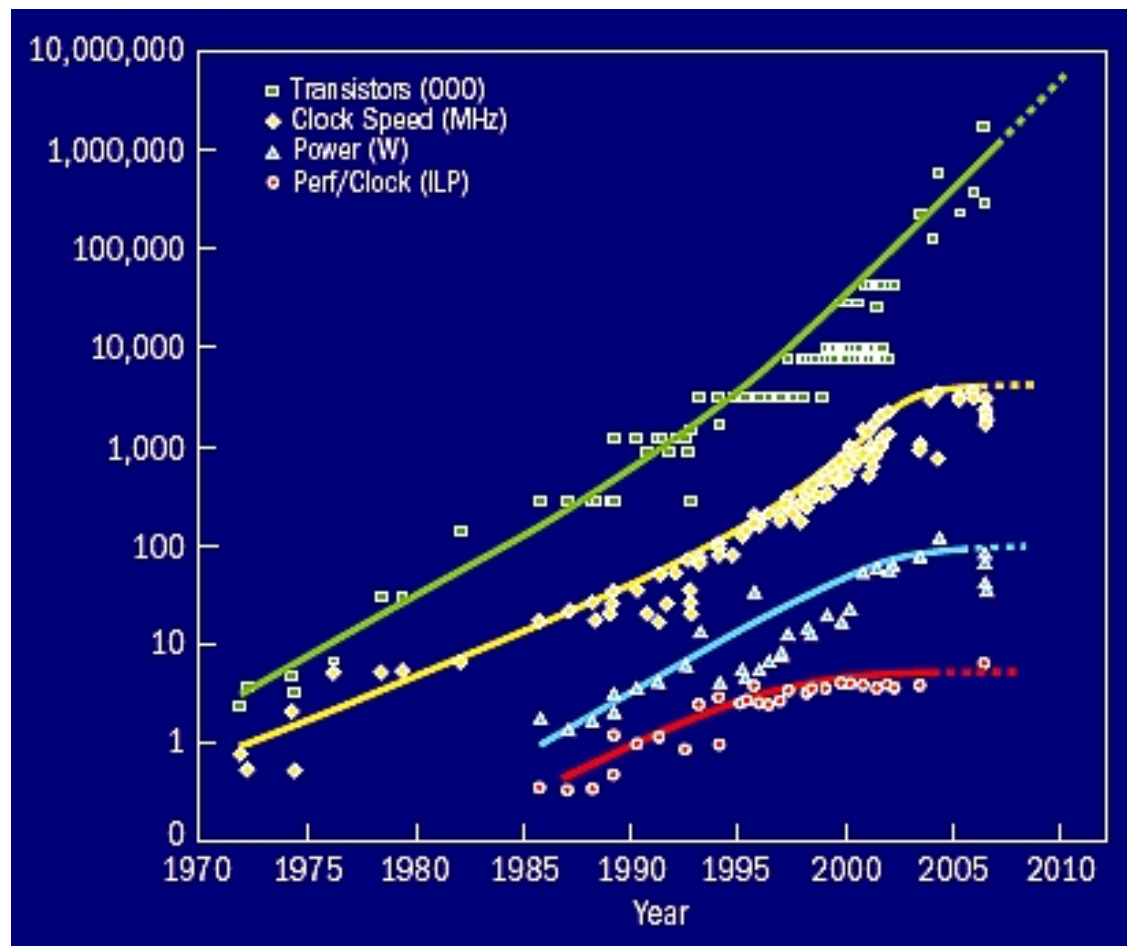
Introdução

- Limites de soluções de aumento de desempenho
 - Parede de Frequência
 - ✧ Aumento da profundidade do Pipeline é comprometido



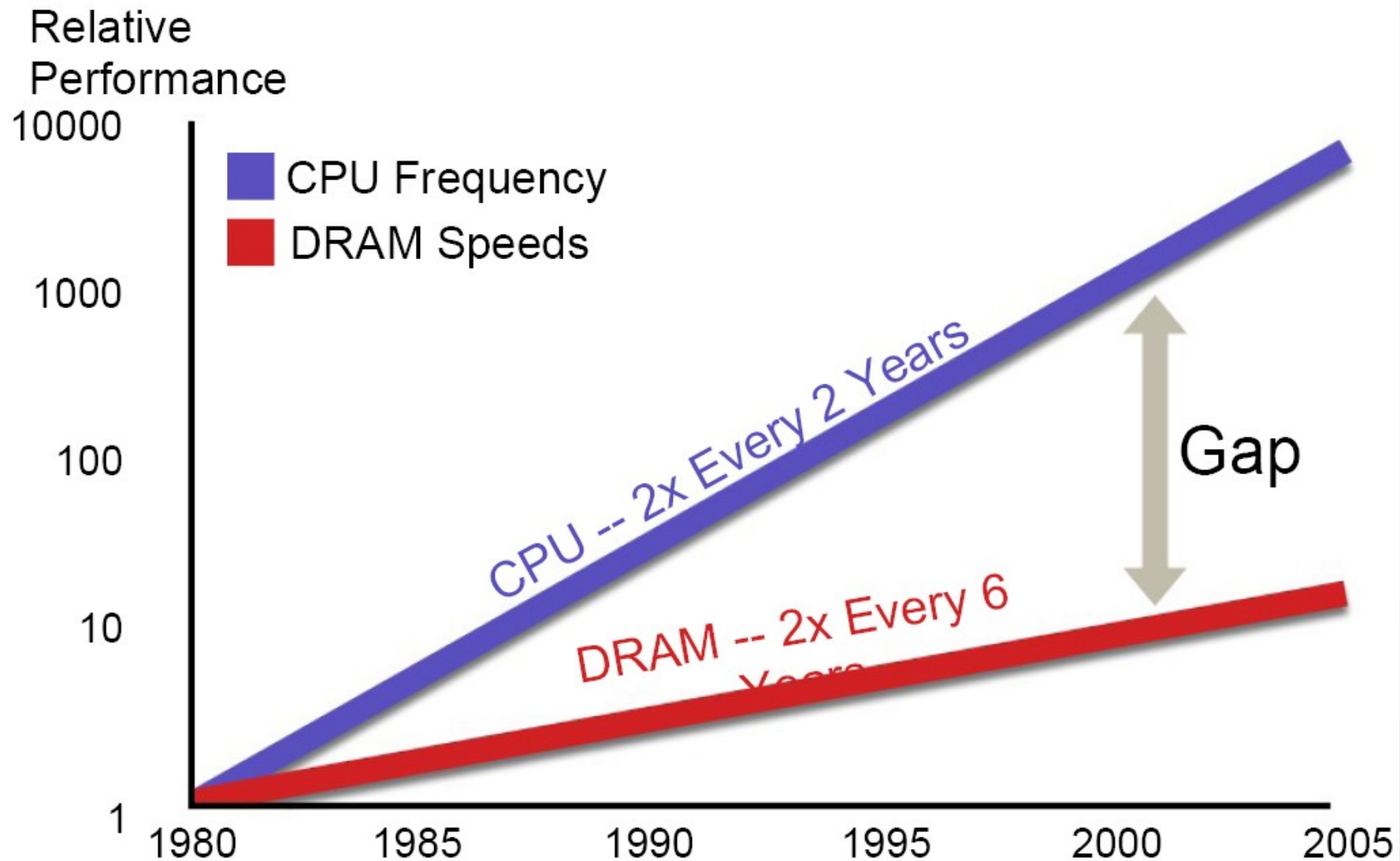
Introdução

- Limites de soluções de aumento de desempenho
 - Parede de Potência
 - ✧ Leakage aumenta 7.5x a cada nova geração



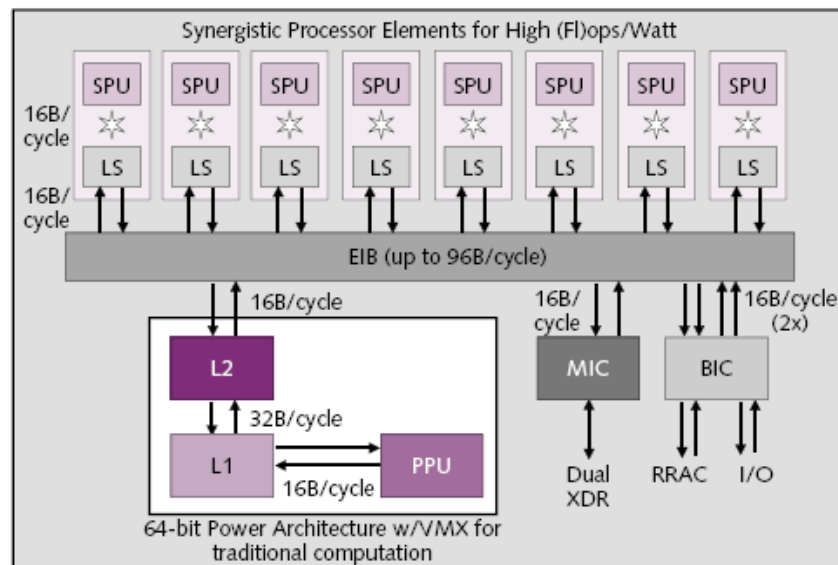
Introdução

- Gargalo da memória



Introdução

- Como utilizar os transistores adicionais que a lei de Moore nos dá a cada 18 meses?
- Uma das soluções: **Multicores**
 - Uso de paralelismo a nível de threads
 - Neste contexto, o gargalo da memória toma outra dimensão

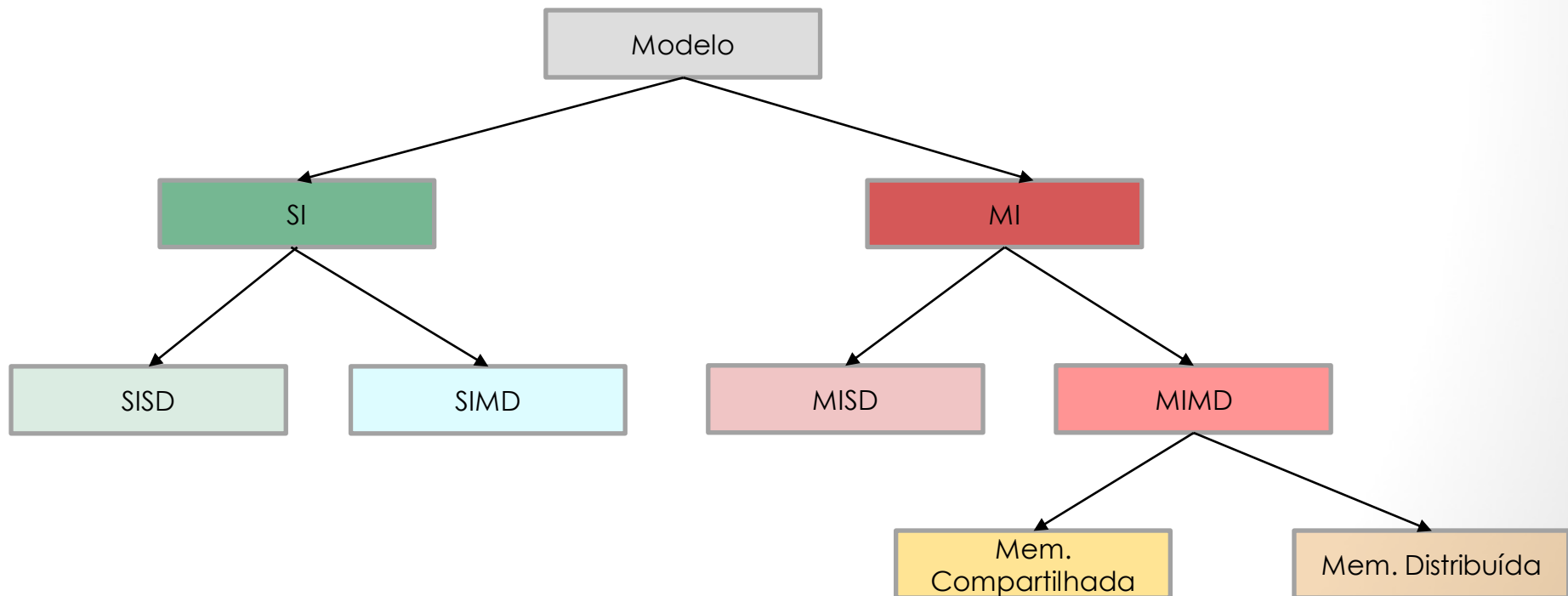


Introdução

- **Multithreading:** único core (CPU) que pode executar múltiplas threads simultaneamente.
- **Multiprocessador:** múltiplas CPUs fortemente acopladas para cooperarem na resolução de um único problema.
- **Processador Multicore** (MPSoC – *Multiprocessor System on Chip*): multiprocessador onde os cores estão em um único chip

Taxonomia de Flynn

- Modelo de classificação, proposto em 1972 e mais aceito até hoje.
- Baseia-se em dois conceitos: sequência de instruções e sequência de dados.



Taxonomia de Flynn

- SI – Single Instruction
 - SISD – Single Instruction Single Data
 - ✧ É o caso das máquinas clássicas de Von Neumann, com um único fluxo de instrução e um único fluxo de dados.
 - SIMD – Single Instruction Multiple Data
 - ✧ Uma mesma instrução executa sobre muitos dados.
Exemplo: arquiteturas vetoriais (Cray1) onde a mesma operação é executada sobre múltiplos operandos.

Taxonomia de Flynn

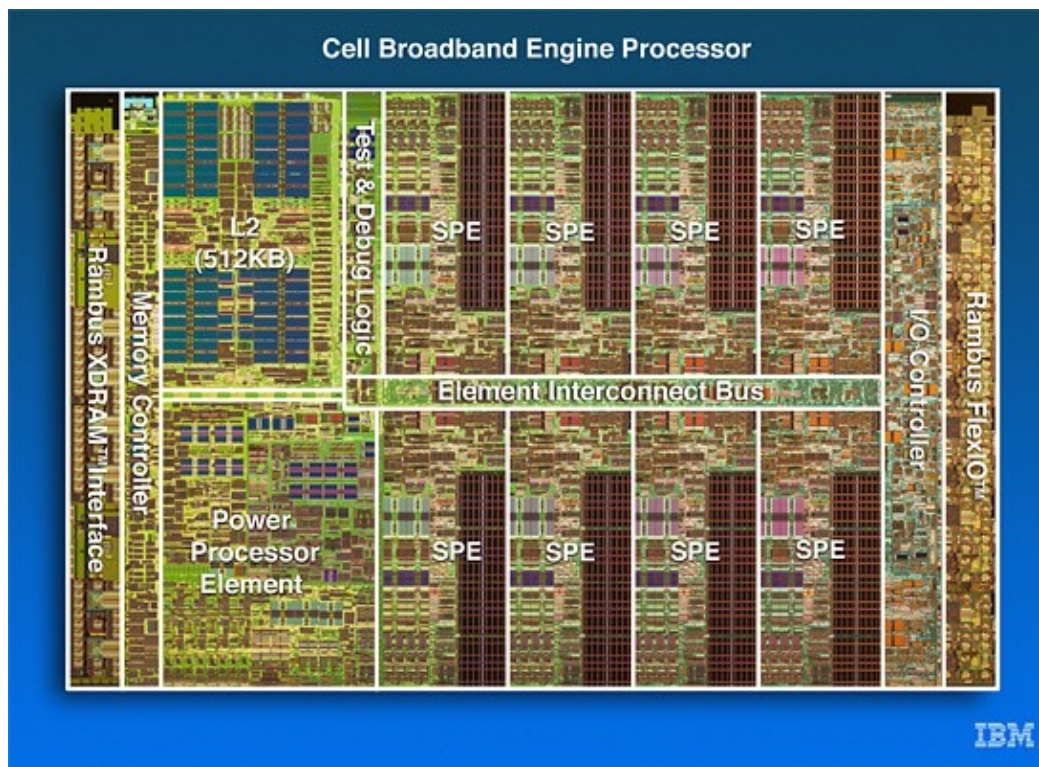
- MI – Multiple Instruction
 - MISD – Multiple Instruction Single Data
 - ✧ É o caso de um conjunto de processadores no qual os dados vão sendo processados em um processador e passados para o processador seguinte. Exemplo: a proposta que mais se aproxima desse tipo é a máquina de fluxo de dados (dataflow machine).
 - MIMD – Multiple Instruction Multiple Data
 - ✧ São os multiprocessados, com várias instruções sendo executadas sobre diferentes dados.

MIMD - Multicores

- CPU com um único processador que contém dois ou mais núcleos
- Compartilham recursos computacionais, como barramentos, cache
- Alcançam alto desempenho por dois motivos
 - Vários núcleos executando em paralelo
 - Tempo curto de comunicação, já que estão em uma mesma CPU

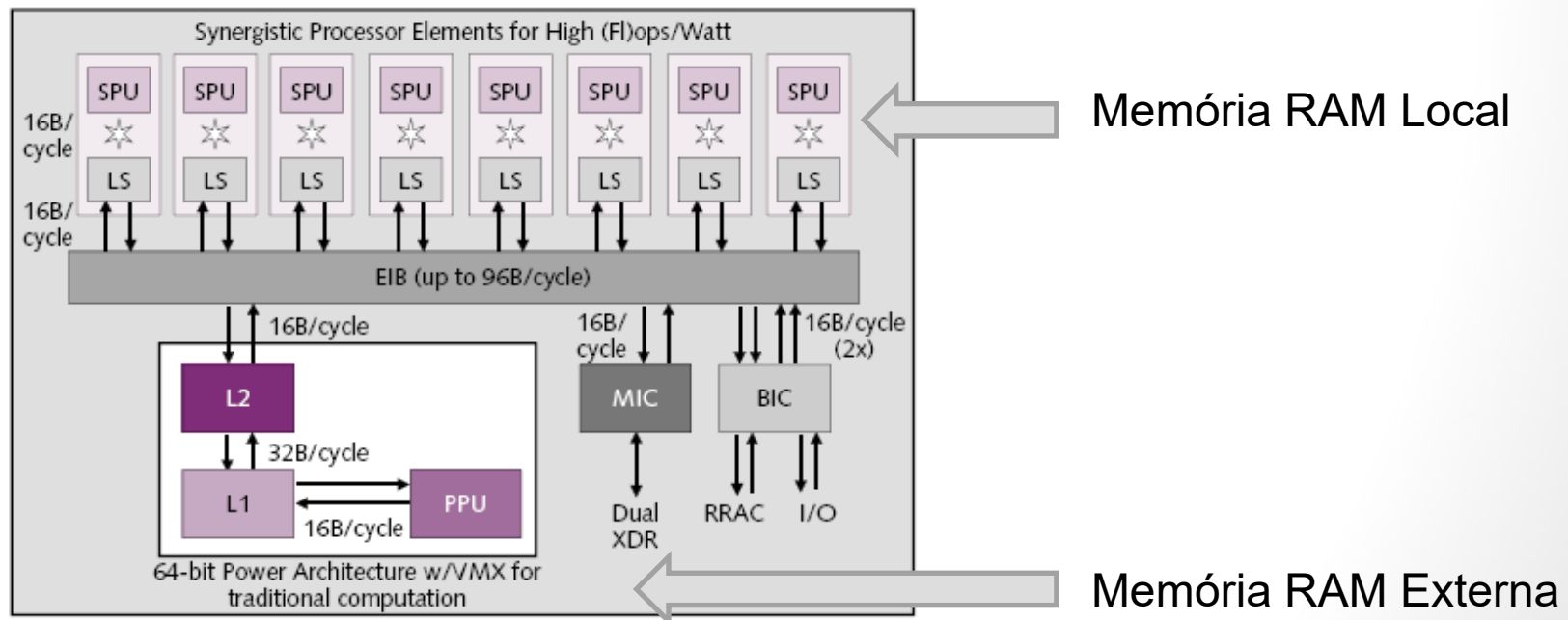
Sistemas Multiprocessados

- Cell Processor (IBM, Toshiba, Sony)
 - Atinge 204.8 Gflops por segundo com precisão simples e 14.6 Gflop por segundo com precisão dupla
 - Suporte a virtualização
 - Largura de banda da memória chega a 25.6 GB/s



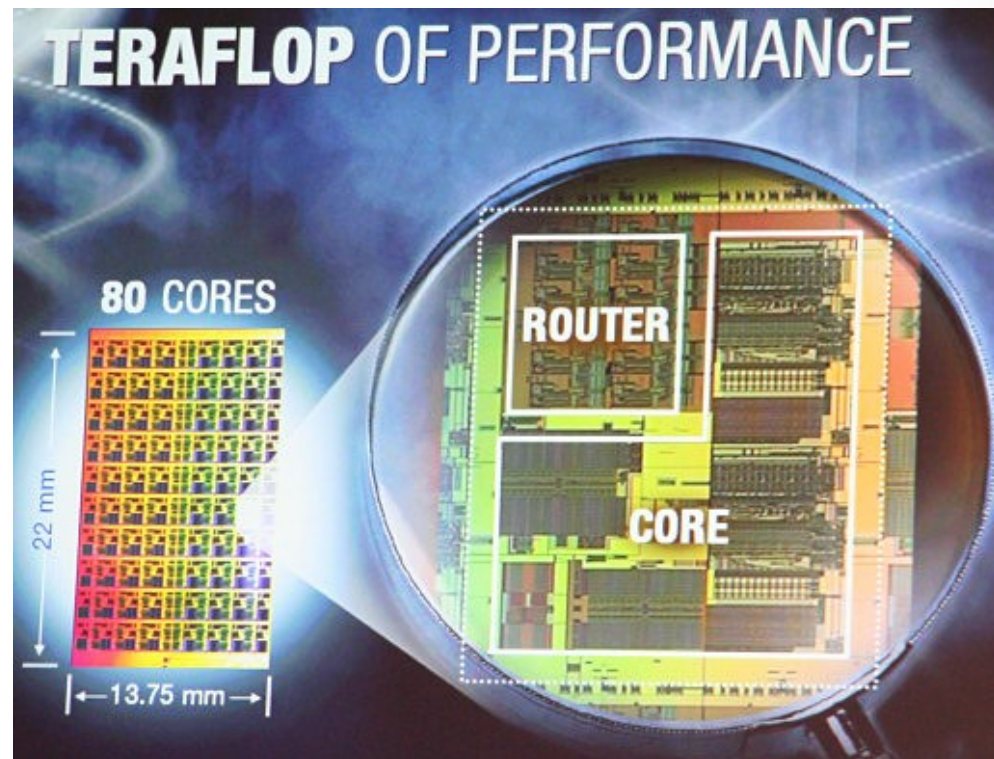
Sistemas Multiprocessados

- Cell Processor (IBM, Toshiba, Sony)
 - PowerXCell 8i
 - ✧ ICache e DCache 32KB 2-way
 - 8 Synergistic Processing Elements (SPE)
 - ✧ 256KB de memória local em cada SPE, 6 cycle load latency
 - Cache L2 de 512K interna ao chip



Sistemas Multiprocessados

- 80 Core TeraFlop Processor (Intel)
 - 80 Cores VLIW
 - Uso de uma rede-em-chip
 - Caches L1 integradas
 - Uso de Power Management
 - ✧ 62 Watts de consumo
 - ✧ Sleep Transistor
 - ✧ Clock Gating



Homogêneos vs. Heterogêneos

- Multicores homogêneos
 - Todos os processadores são iguais
 - ✧ Em algum aspecto seja ele organizacional (monociclo, multiciclo, pipeline) ou arquitetural (ISA)
 - ✧ São mais simples de **desenvolver** e **testar**
 - Arquitetura **regular**
- Multicores heterogêneos
 - Pelo menos UM processador é diferente
 - ✧ Em algum aspecto seja ele organizacional (monociclo, multiciclo, pipeline) ou arquitetural (ISA)
 - ✧ Implementação complexa
 - Testes de comunicação são necessários
 - ✧ Dão mais opções de execução
 - Processadores mais rápidos porém maiores vs. Processadores mais menores porém mais lentos



Juntando Todas as Partes

Gustavo Girão