

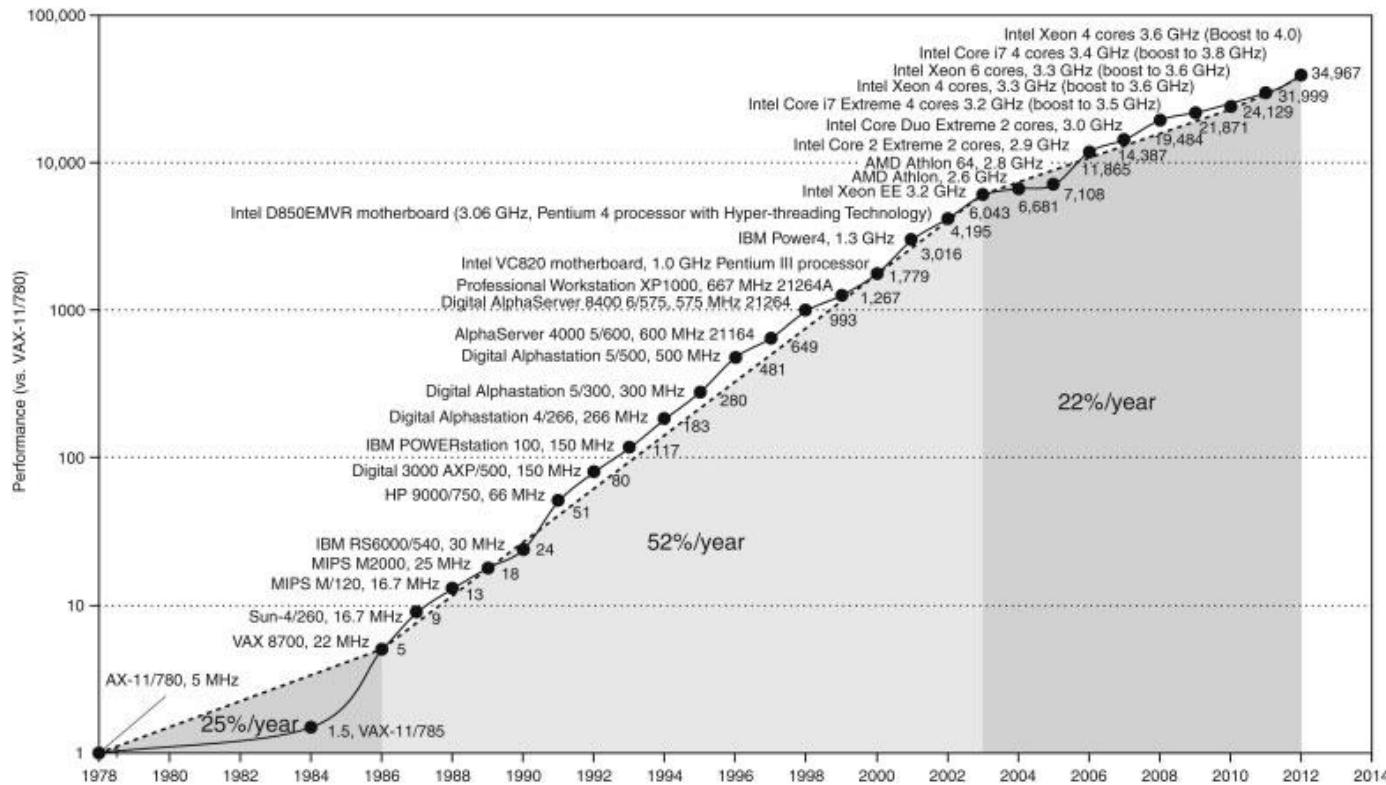
# *Arquitecturas MultiCore*

Arquitetura de Computadores  
Licenciatura em Engenharia Informática  
Luís Paulo Santos

# Material de Apoio

- “*Computer Systems: a Programmer's Perspective*”; Randal E. Bryant, David R. O'Hallaron--Pearson (2nd ed., 2011)
  - Secção 1.9.1 - Concurrency and Parallelism
  - Aside: page 584
- “*Computer Organization and Design: The Hardware / Software Interface*” David A. Patterson, John L. Hennessy; 5th Edition, 2013
  - Secção 1.8
  - Secções 6.1 e 6.2
  - Secções 6.4 e 6.5

# Evolução Uni Processador: frequência



[Computer Organization Design: The Hardware / Software Interface ; Patterson & Henessey ]

FIGURE 1.17 Growth in processor performance since the mid-1980s. This chart plots performance relative to the VAX 11/780 as measured by the SPECint benchmarks (see Section 1.10). Prior to the mid-1980s, processor performance growth was largely technology-driven and averaged about 25% per year. The increase in growth to about 52% since then is attributable to more advanced architectural and organizational ideas. The higher annual performance improvement of 52% since the mid-1980s meant performance was about a factor of seven higher in 2002 than it would have been had it stayed at 25%. Since 2002, the limits of power, available instruction-level parallelism, and long memory latency have slowed uniprocessor performance recently, to about 22% per year.

# Evolução MicroProcessadores: ILP

- *Instruction Level Parallelism (ILP)*

Os processadores incluem mecanismos para **executar várias instruções do mesmo programa** em paralelo

- **Pipelining** (até 32 estágios)  
profundidade óptima (potência vs. desempenho) ~ 7 estágios
- **Superescalaridade**  
Múltiplos *pipelines* permitem a execução simultânea de múltiplas instruções
- **Execução fora de ordem**  
Maximização do número de instruções em execução
- **Execução especulativa**  
Resolução de dependências de controlo

# Evolução MicroProcessadores: *Moore's Law*

- Lei de Moore:

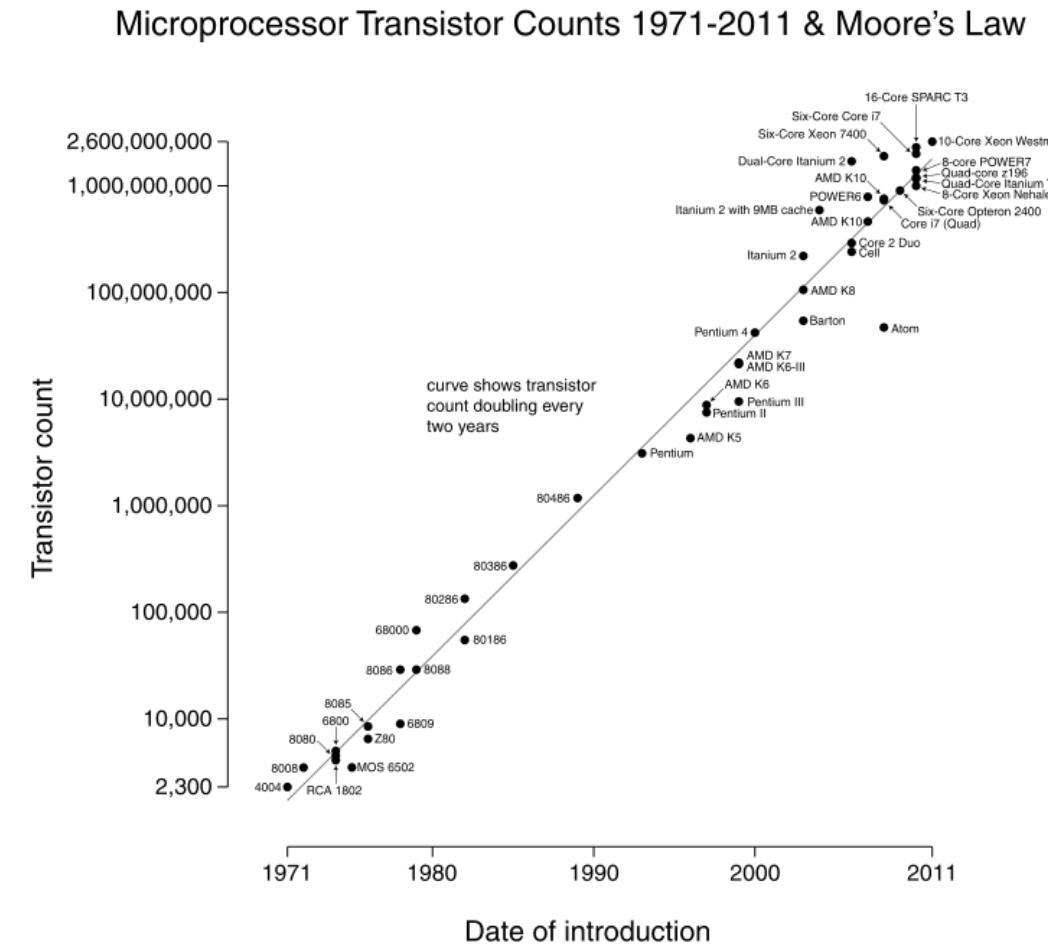
*"The number of transistors on a chip doubles every two years"*

[Adaptado de G. Moore, 1965]

Também formulada como:

*"O desempenho dos processadores dobra cada 18 meses"*

[David House, Intel]



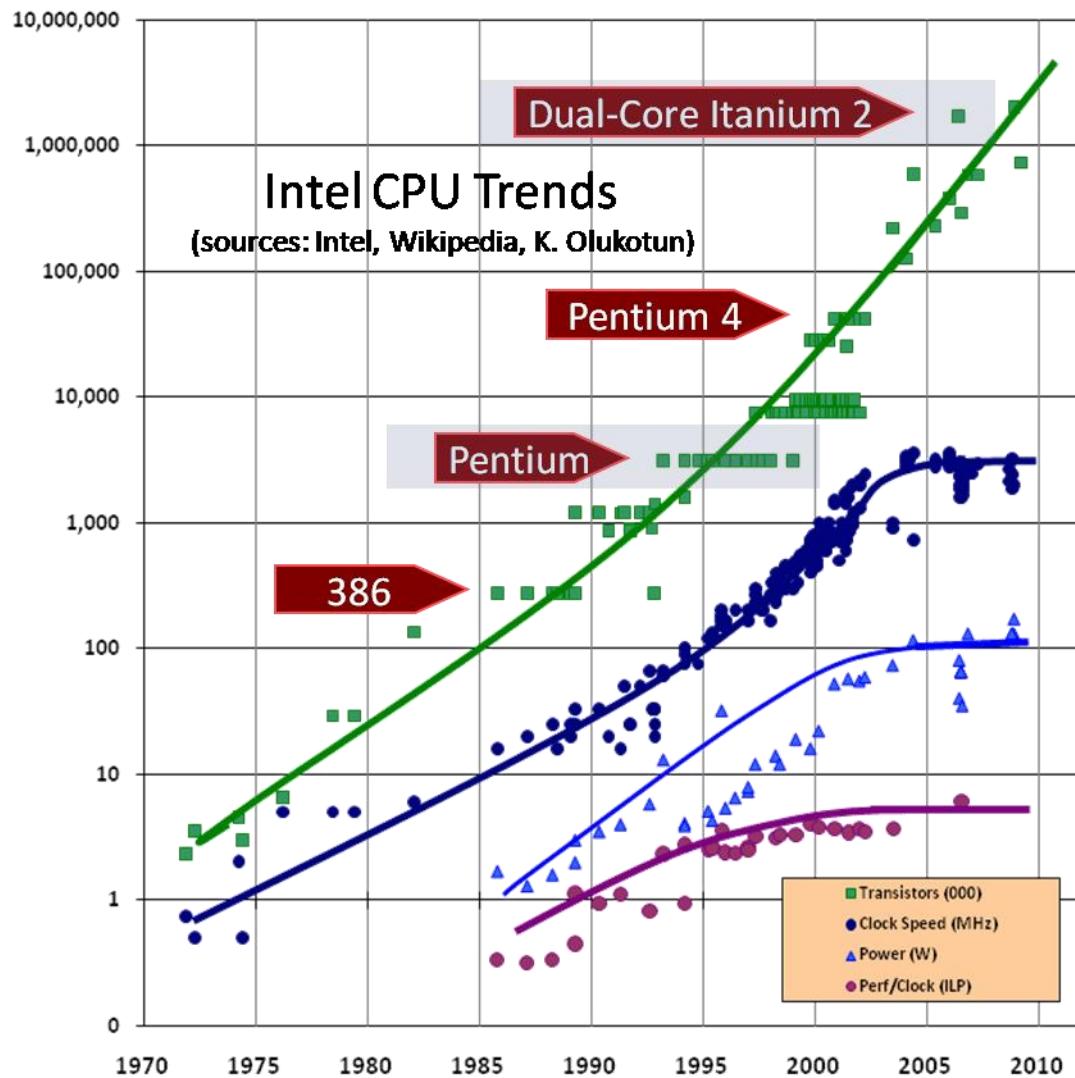
# Evolução MicroProcessadores: *power & ILP walls*

- O aumento da frequência do relógio aumenta a potência dissipada

$$P = CV^2f \quad \begin{array}{l} P - \text{potência dinâmica} \\ C - \text{capacidade} \end{array} \quad \begin{array}{l} V - \text{tensão} \\ f - \text{frequência} \end{array}$$

- O *hardware* adicional necessário para explorar ILP e controlar *pipelines* profundos e execução fora de ordem é muito complexo, resultando em:
  - Elevado consumo de potência associado a este controlo
  - Desenho lógico extremamente complexo, cuja correcção é difícil de verificar
- O grau de ILP disponível nos programas é limitado (sendo o caso comum cerca de 3 instruções por ciclo)

# Evolução MicroProcessadores: *power wall*



# Evolução MicroProcessadores: *power wall*

- Em 2004 a Intel lança o Prescott, processador *single core*, com pipelines muito profundos e anuncia que esta arquitectura poderá ir até aos 10 GHz
- O consumo de potência e o calor dissipado nunca permitiram que este processador fosse além dos 3.8 GHz
- Em 2005 é lançado o Prescott 2M, com *hyper threading*, prenunciando que algo estaria a mudar na forma como os processadores Intel iriam evoluir  
(outros fabricantes já tinham seguido esta linha desde 2001)

# *Thread Level Parallelism*

- Paralelismo a um nível mais grosso do que as instruções de um programa:
  - explorar o paralelismo entre diferentes **fios de execução (*threads*)** de um mesmo programa ou de programas diferentes
  - as *threads* podem ser partes de um programa paralelo ou mesmo programas diferentes (processos)
  - cada *thread* tem o seu estado (instruções, dados, contexto ( conteúdo de registos, *instruction pointer*, etc.)) para que possa executar independentemente

thread 0

```
for (i=0; i < S/2; i++)  
{ processa a[i]; }
```

thread 1

```
for (i=S/2; i< S ; i++)  
{ processa a[i]; }
```

# *Thread Level Parallelism*

- ILP explora **paralelismo implícito** numa **única sequência de instruções**
- TLP explora **paralelismo explícito** entre **múltiplas sequências de instruções**

**“O código deve ser escrito explicitamente para expor TLP”**  
***“the free lunch is over”***

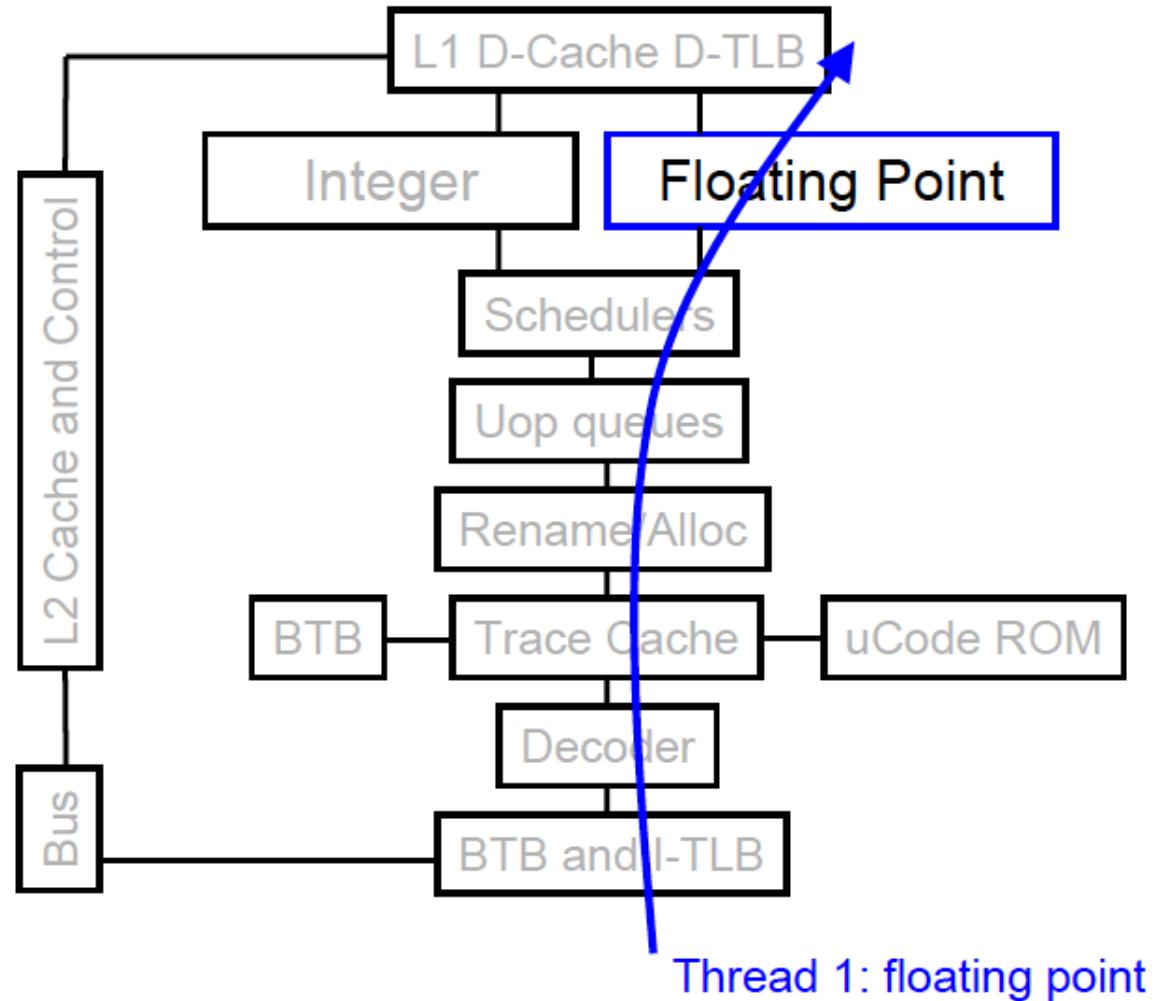
- Objectivo:
  - aumentar o débito em computadores que executam vários programas
  - diminuir o tempo de execução de programas paralelos (*multithreaded*)

# Simultaneous Multi Threading

- Permitir a execução simultânea de múltiplas *threads* no mesmo processador (*core* ou núcleo)
- Ideia base: as diferentes unidades funcionais podem ser partilhadas por diferentes *threads*
- Exemplo:  
Se uma *thread* está à espera que uma unidade complete uma operação de vírgula flutuante, outra *thread* pode usar a unidade de operações inteiras

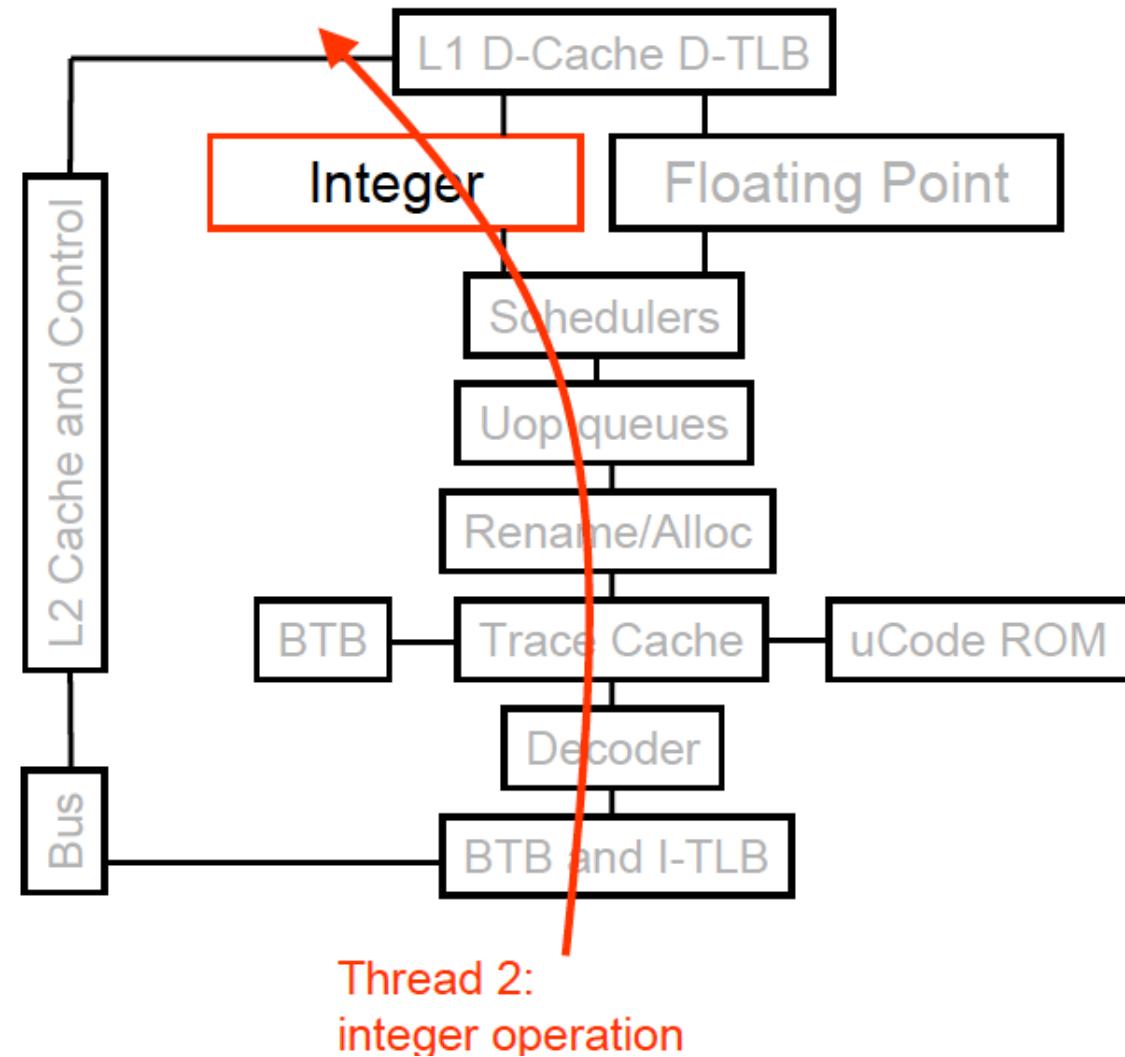
# Simultaneous Multi Threading

- Sem SMT apenas uma thread usa o núcleo em cada instante



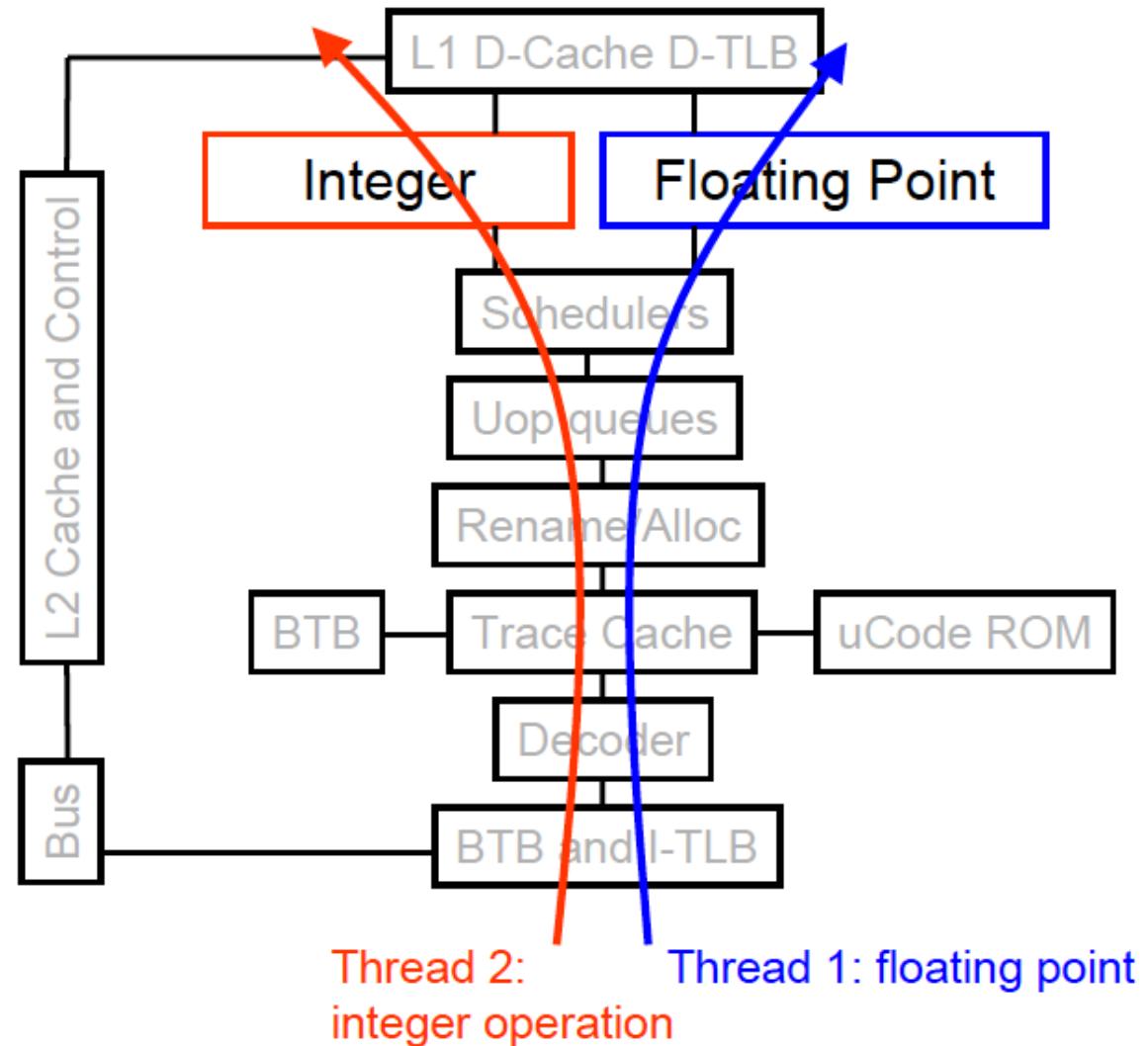
# Simultaneous Multi Threading

- Sem SMT apenas uma thread usa o núcleo em cada instante



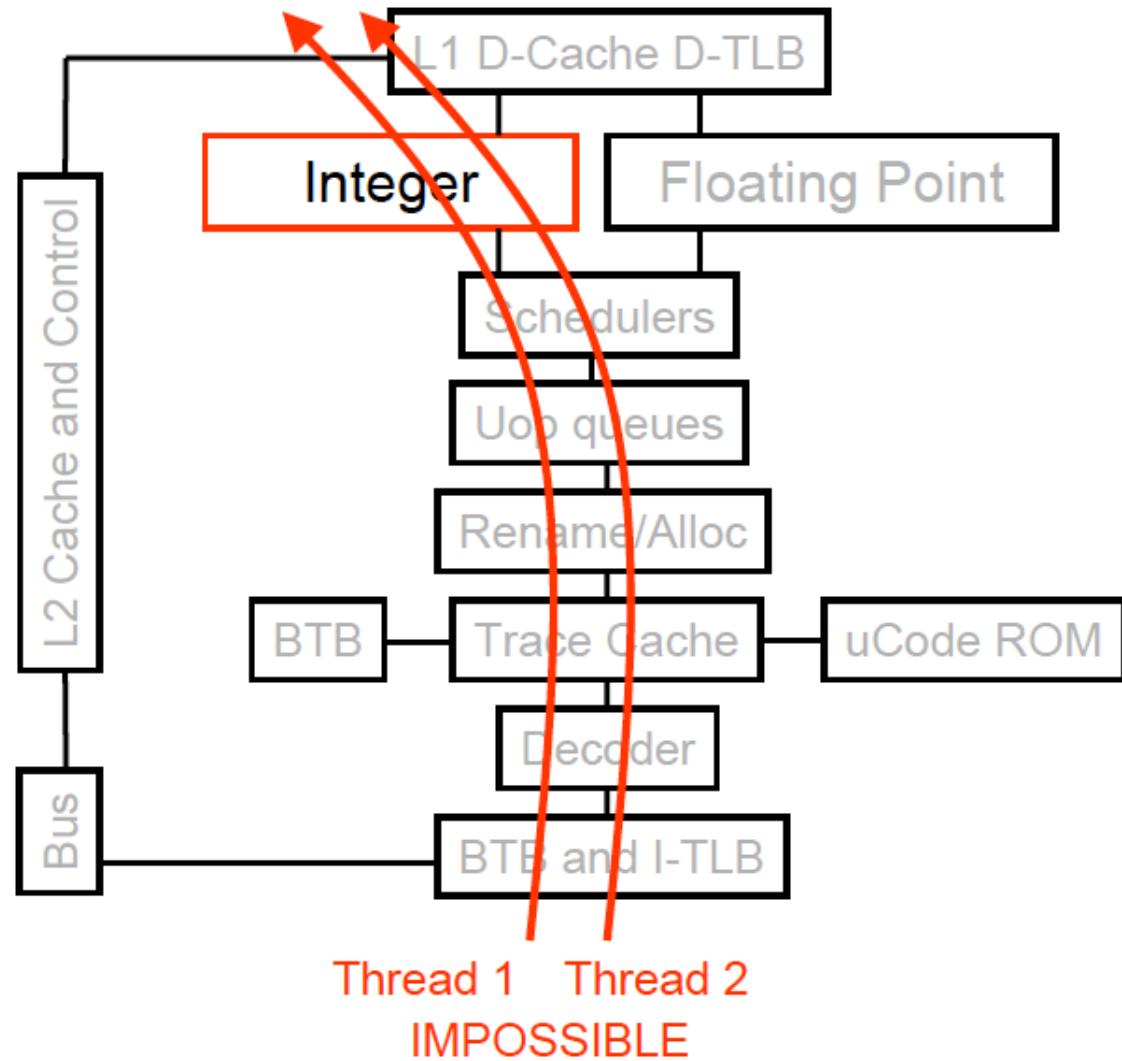
# Simultaneous Multi Threading

- Com SMT ambas as *threads* podem executar simultaneamente



# Simultaneous Multi Threading

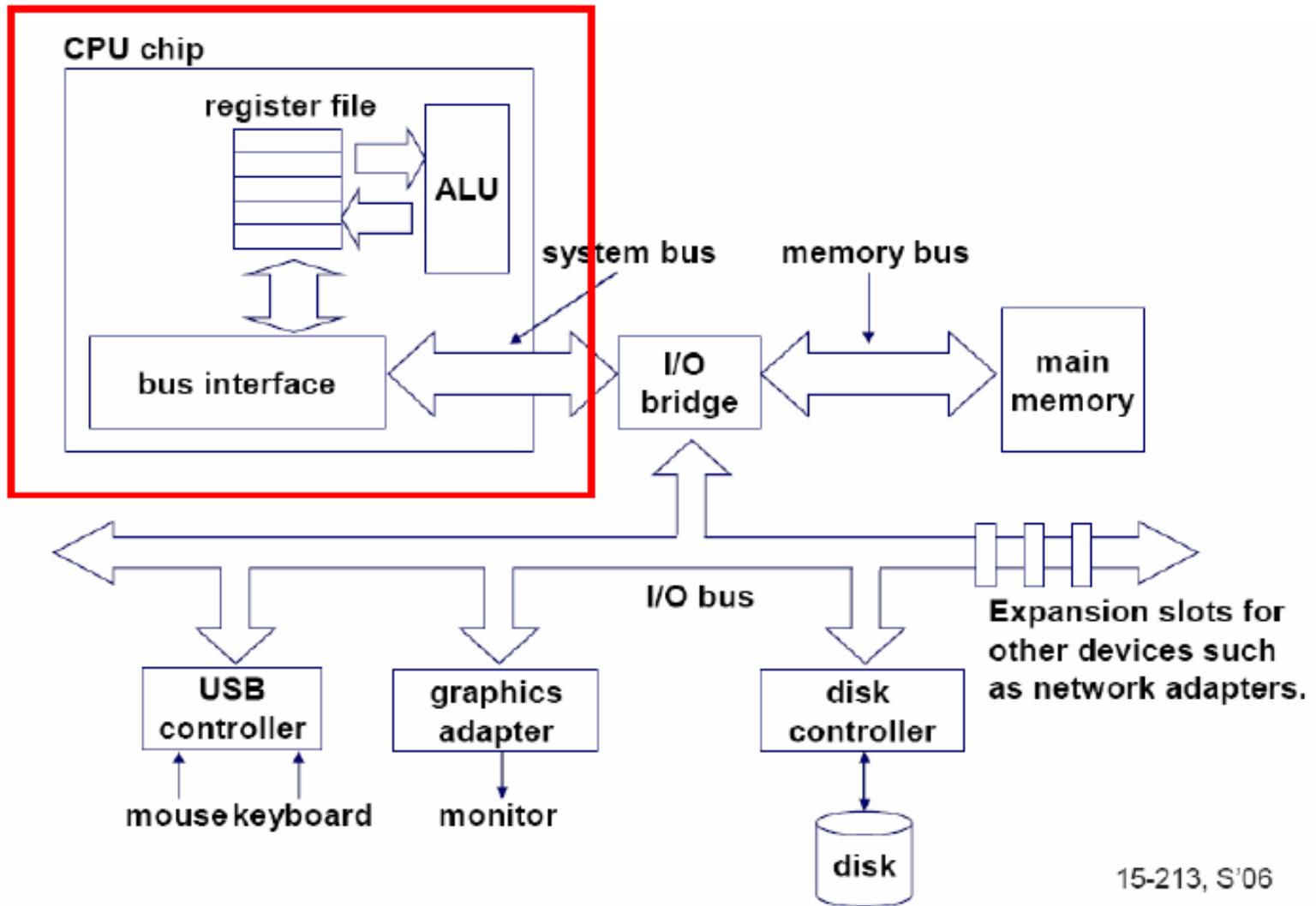
- Mas com SMT muitos recursos são partilhados e não podem ser usados simultaneamente por diferentes *threads*



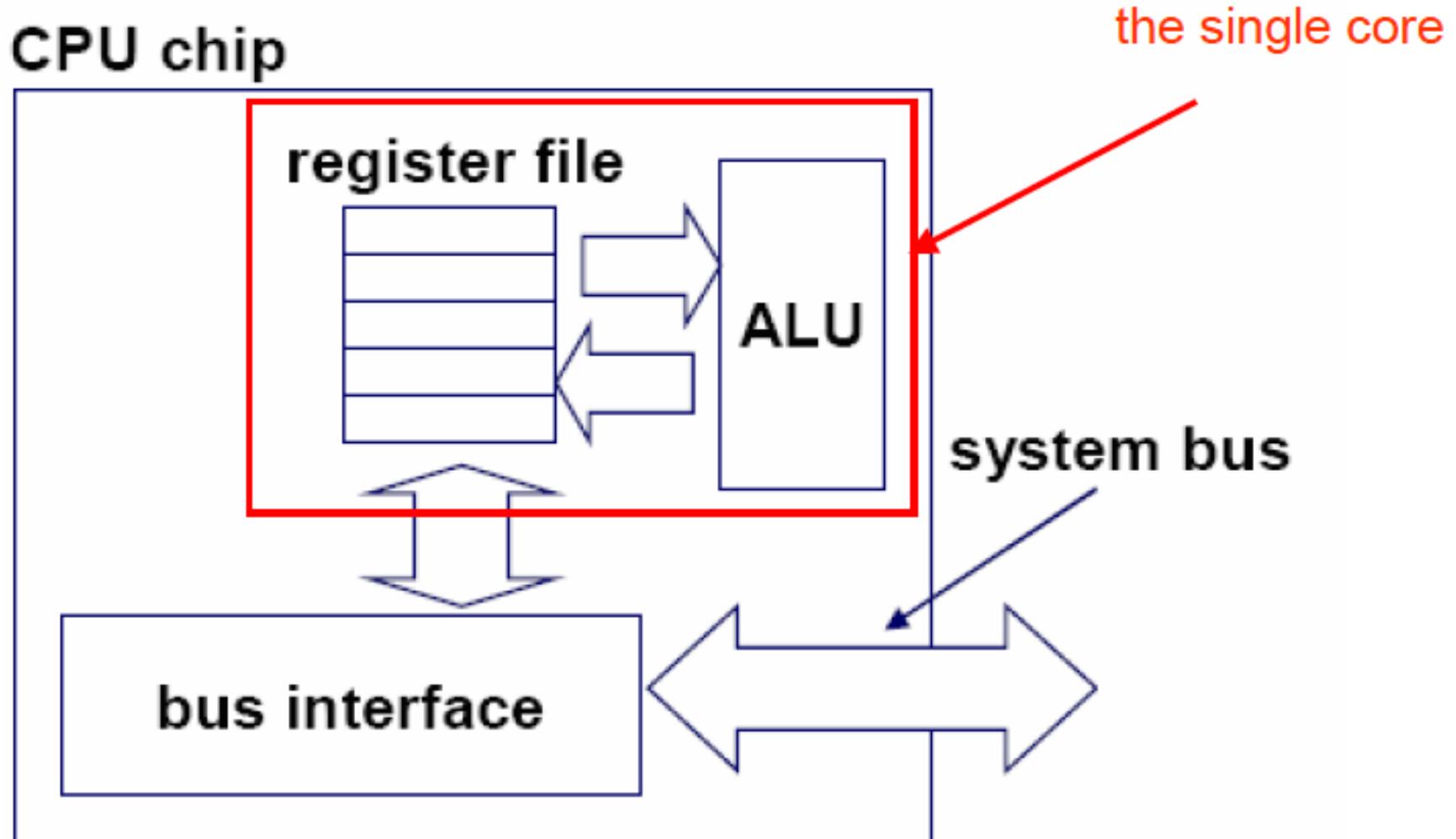
# Simultaneous Multi Threading

- As threads partilham recursos entre si resultando num ganho marginal
- A Intel designa SMT por HyperThreading (HT)
- Nos processadores hyperthreaded duas *threads* podem usar diferentes recursos do mesmo processador
- HT resulta normalmente num ganho de 30% em tempo de execução
- Diz-se que um processador HT tem dois cores (núcleos) lógicos

# Computador Single Core

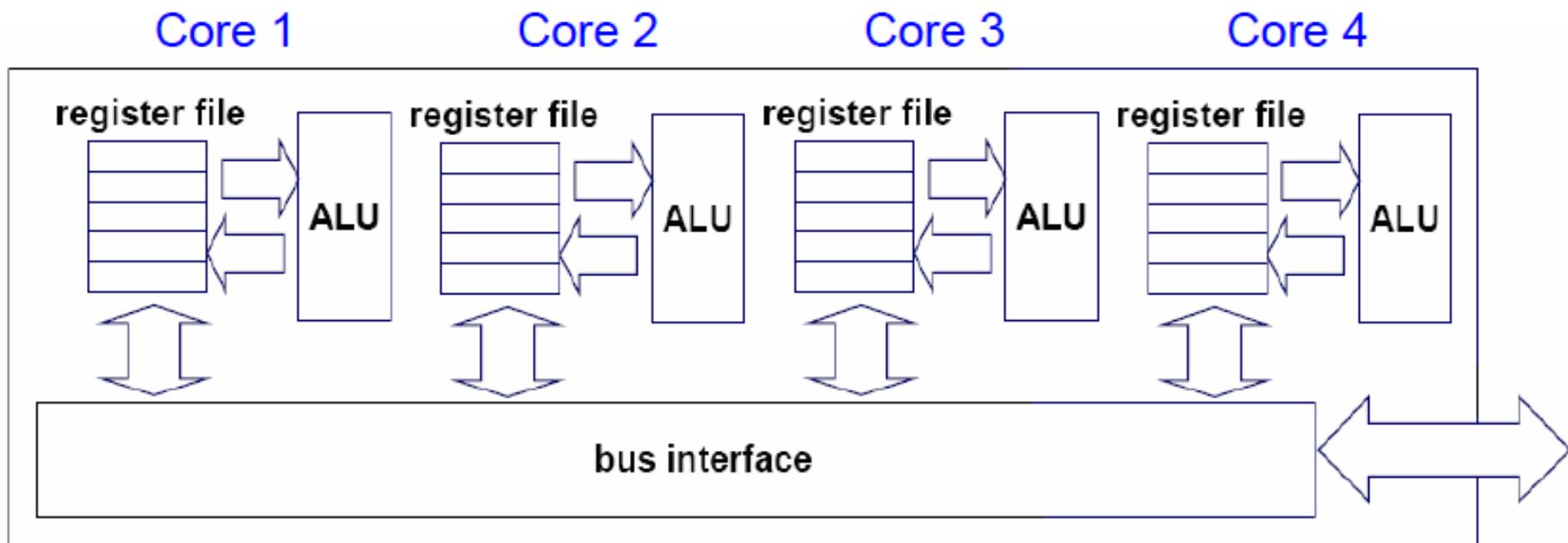


# *Single Core Chip*



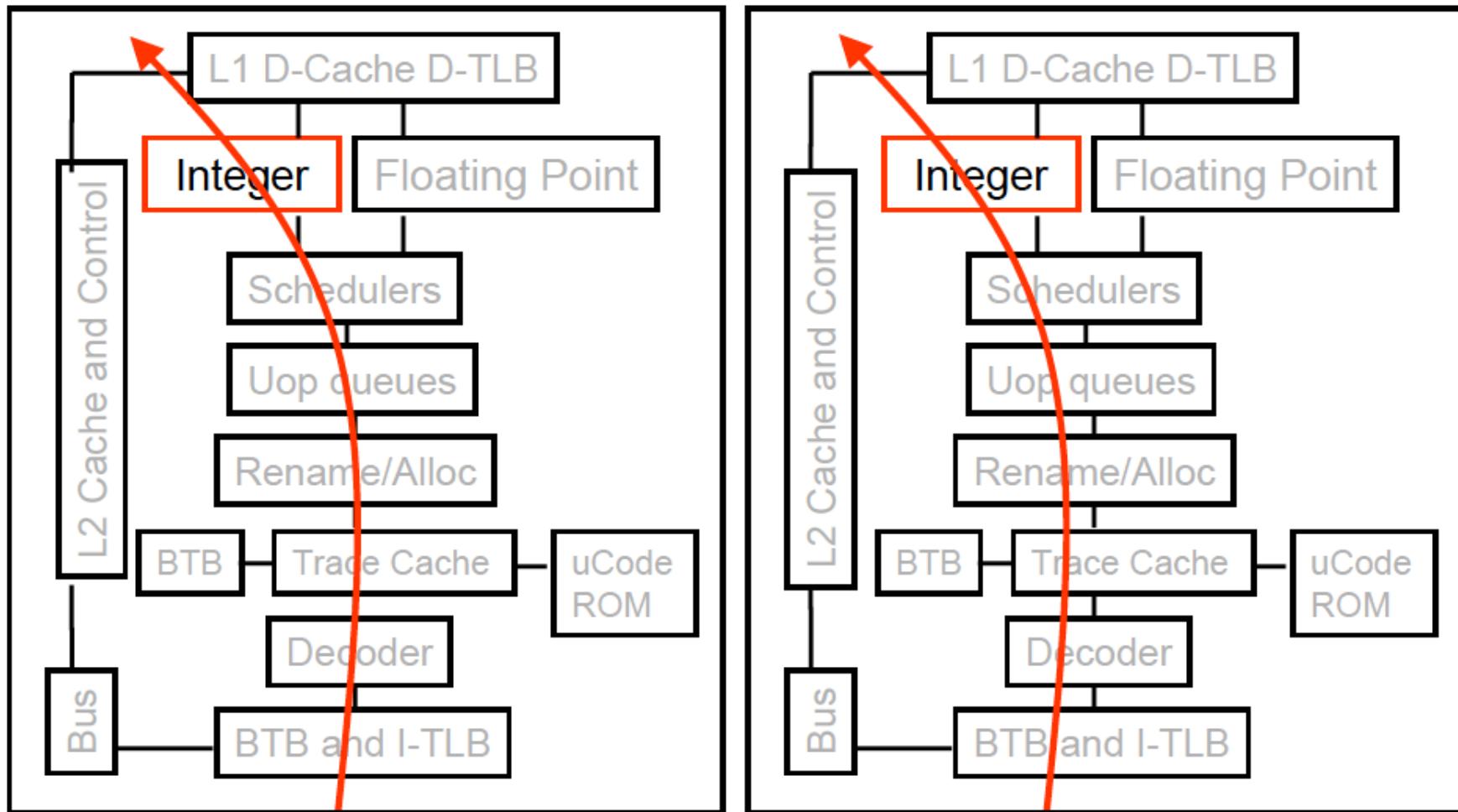
# *Multi Core Chip*

- Replicar múltiplos núcleos num único *chip*



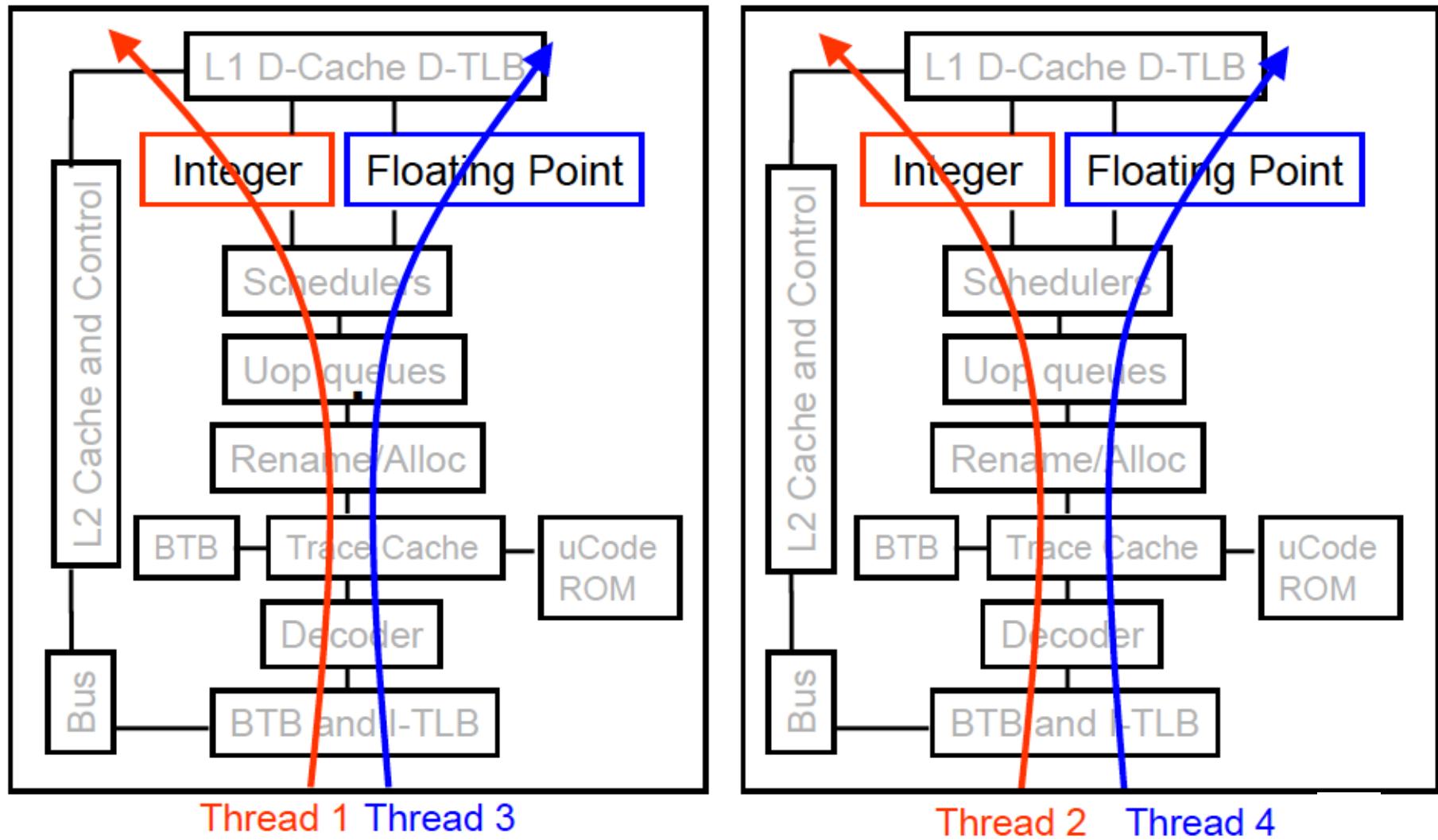
# Multi Core Chip

- As *threads* podem correr em diferentes cores



# *Multi Core Chip e SMT*

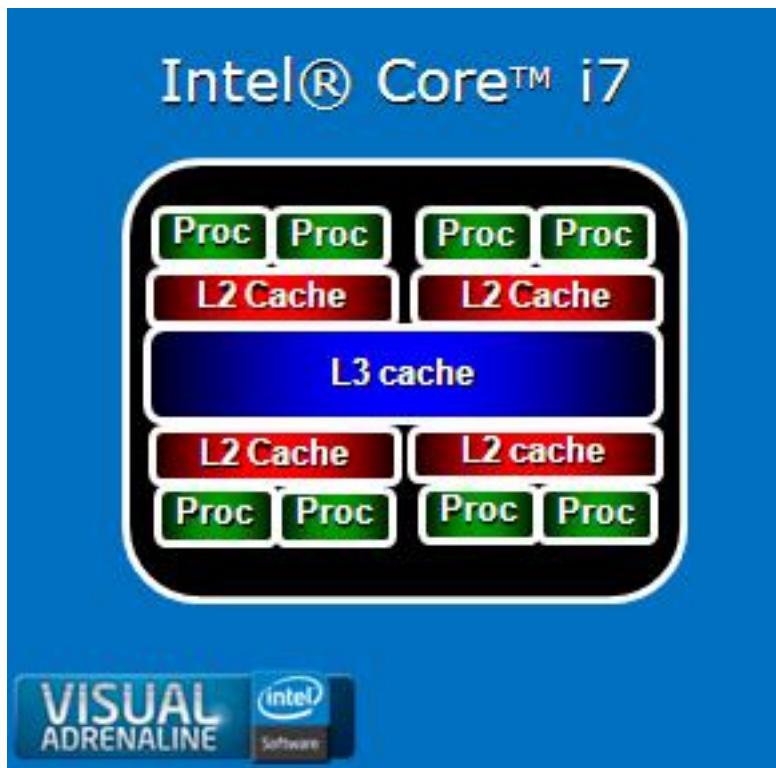
- Os cores podem ou não suportar SMT



# *Multi Core*

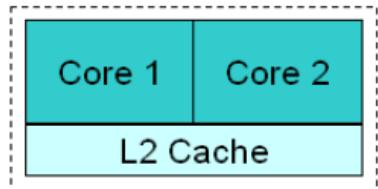
- Actualmente:
  - computadores domésticos : processadores com 2 a 4 cores
  - servidores : 8 a 64 cores
- O Sistema Operativo vê cada core como um processador independente
- O ganho com cada core adicional diminui e os cores competem por recursos tais como barramentos e memória
- A hierarquia de memória é um aspecto fundamental no desempenho do sistema
- Para tirar partido de arquitecturas multicore os programas devem ser desenvolvidos especificamente para esse efeito

# *MultiCore*: Exemplos



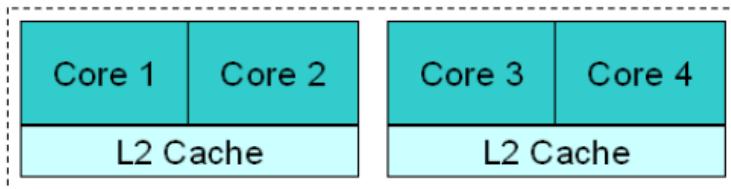
- 4 cores (8 lógicos com HyperThreading)
- cache L1 privada para cada core físico
- cache L2 partilhada por pares de cores físicos
- cache L3 partilhada

# *MultiCore: Exemplos*



Shared Cache

- Core Duo
- Core 2 Duo

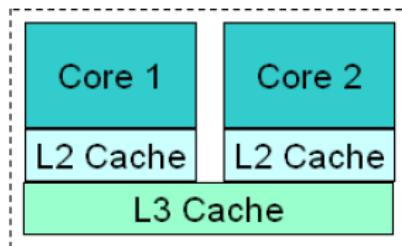


Current Intel Quad-Core CPUs

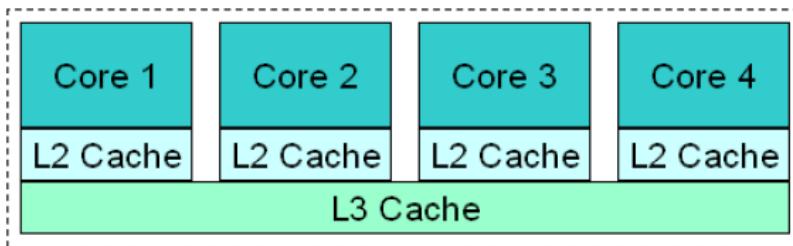
- Core 2 Quad
- Core 2 Extreme QX

L1 = 32KB+32KB (/Core)

L2 = 2MB/4MB/6MB



K10-based dual-core CPU



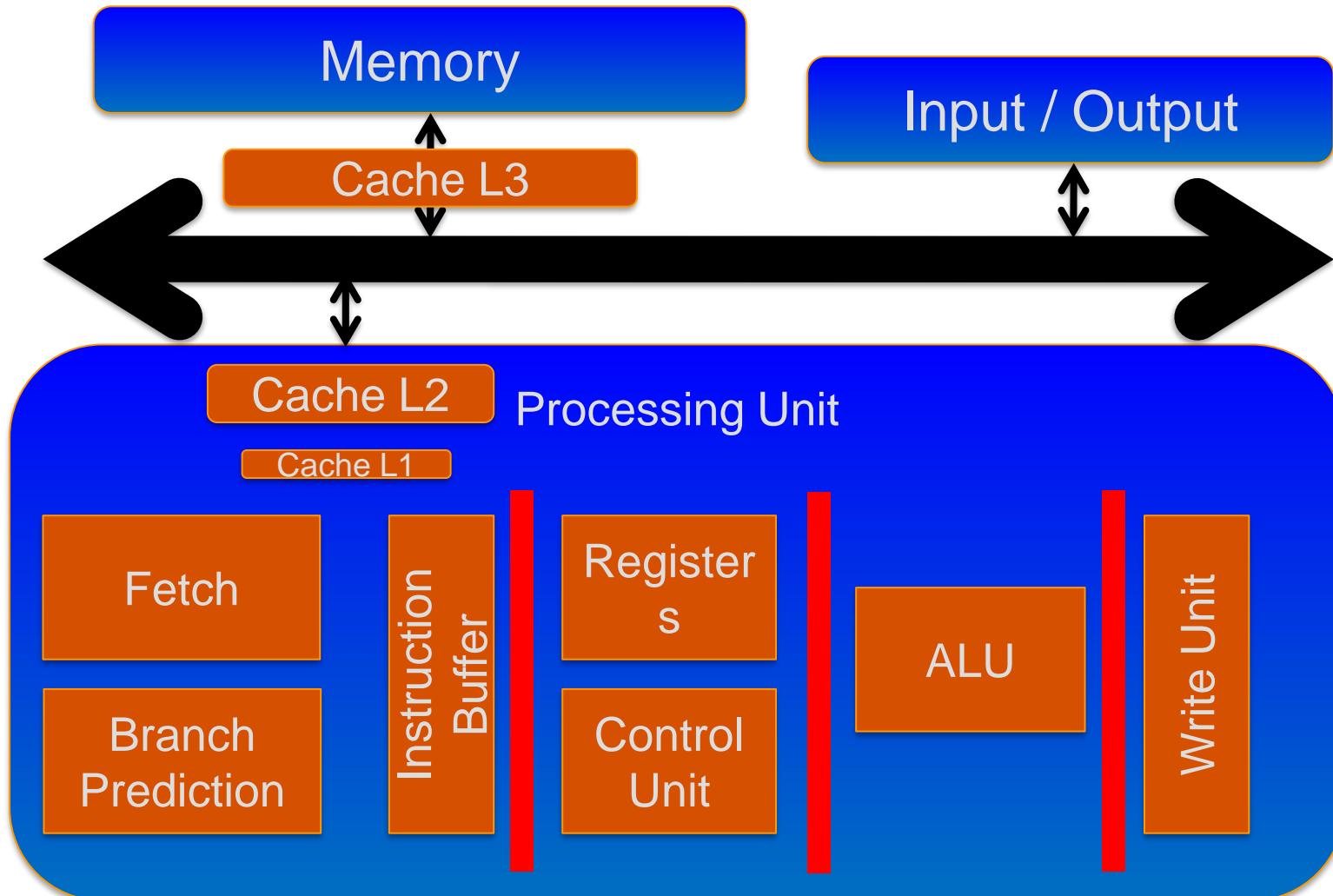
K10-based quad-core CPU

L1 = 64KB+64KB (/Core)

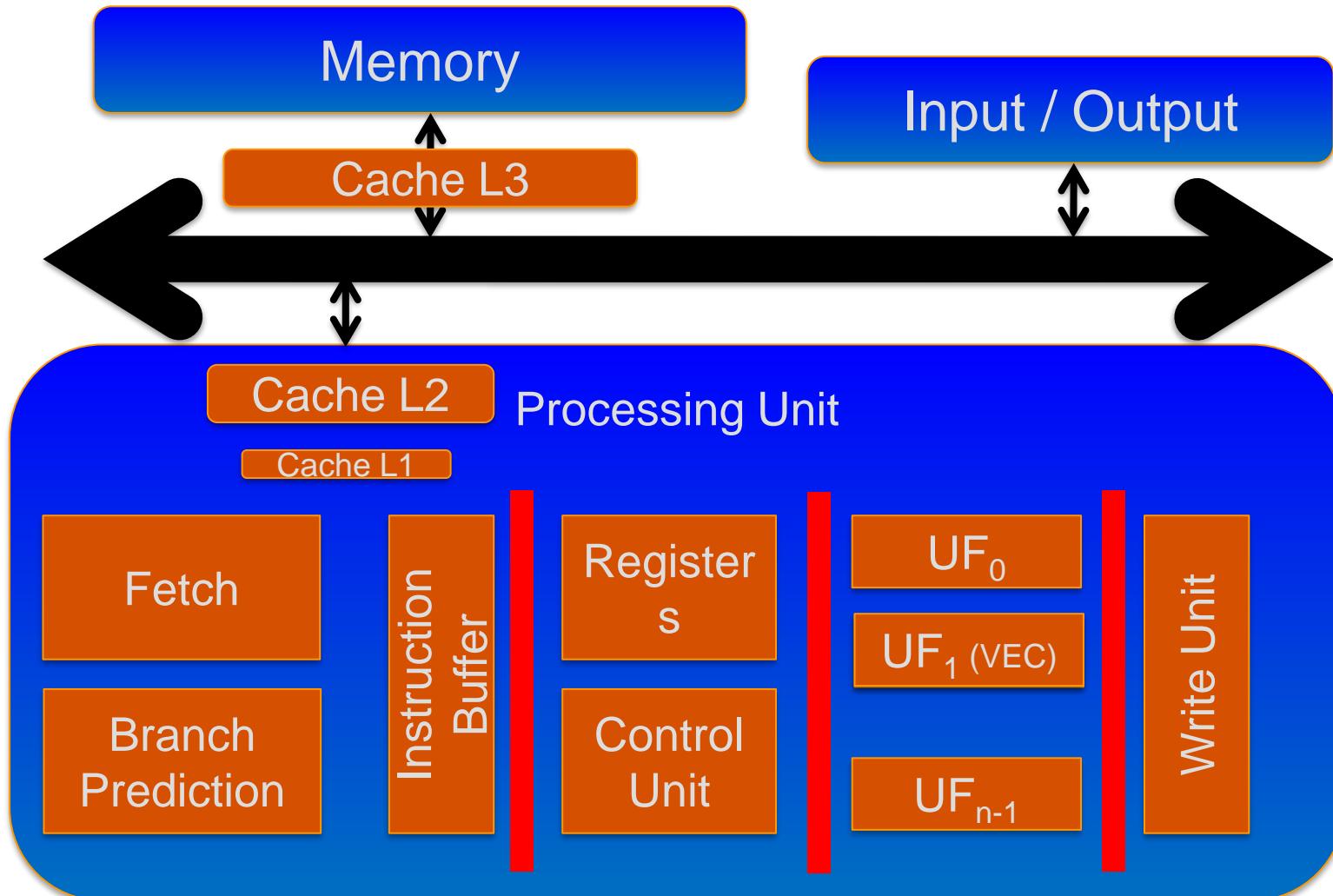
L2 = 512K (/Core)

L3 = 2MB/6MB

# Evolução dos Processadores



# Evolução dos Processadores



# Evolução dos Processadores

