# Desempenho

# Desempenho: uma analogia

- **O que é desempenho ?**

| Airplane | Passenger capacity | Cruising range (miles) | Cruising speed (m.p.h) | Passenger throughput (passenger x m.p.h) |
|---|---|---|---|---|
| Boeing 777 | 375 | 4630 | 610 | 228.750 |
| Boeing 747 | 470 | 4150 | 610 | 286.700 |
| BAC/Sud Concord | 132 | 4000 | 1350 | 178.200 |
| Douglas DC-8-50 | 146 | 8720 | 544 | 79.424 |

# Desempenho: métricas

- **Comparação entre PCs**
  - **Qual deles termina a tarefa primeiro ?**
  - **Tempo para realizar uma tarefa**
    - » **Tempo de execução ou tempo de resposta**
- **Comparação entre servidores**
  - **Qual deles completou mais tarefas ?**
  - **Número de tarefas na unidade de tempo**
    - » **Vazão (*throughput*)**

# Desempenho relativo

- **"X é n vezes mais rápido que Y" significa:**

$$n = \frac{\text{TempoEx}(Y)}{\text{TempoEx}(X)} = \frac{\text{Desempenho}(X)}{\text{Desempenho}(Y)}$$

# Medida de desempenho

- **Tempo de execução de um programa**
  - **Medido em segundos**
- **Tempo de resposta**
  - **Tempo total para completar uma tarefa**
    - » **CPU + memória + HD + E/S**
    - » **Compartilhamento: programas simultâneos**
- **Tempo de execução de CPU**
    - » $\text{tempo}_{\text{CPU}} = \text{tempo}_{\text{usuário}} + \text{tempo}_{\text{sistema}}$

# Medida de desempenho

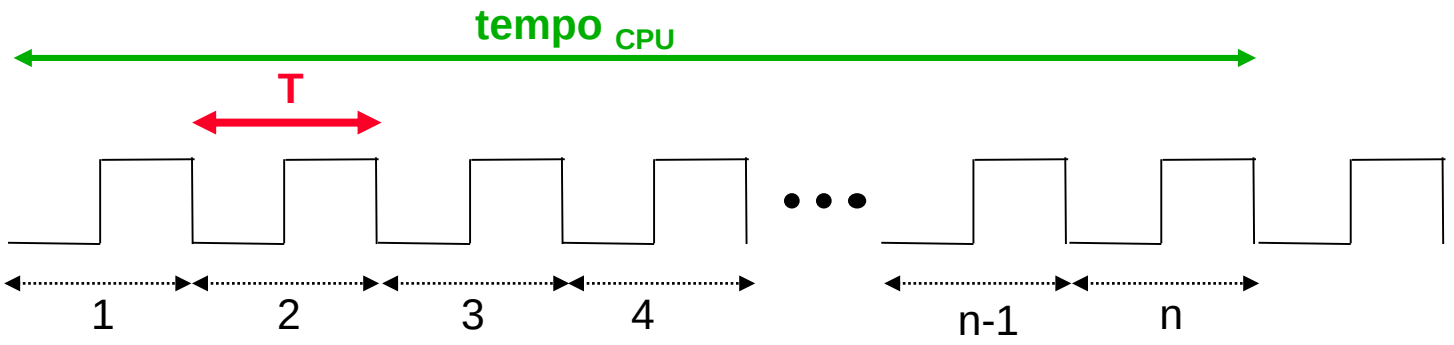
- **Desempenho do sistema**
  - **Tempo de resposta de um sistema sem carga**
- **Desempenho de CPU**
  - **Tempo de CPU dedicado a um dado usuário**
  - **Foco deste curso**

# Desempenho: perspectivas

- **Perspectiva do usuário**
  - **Quão rápido um programa executa ?**
  - **Métrica: tempo**
- **Perspectiva do projetista**
  - **Quão rápido o HW executa funções básicas**
  - **Métrica: ciclos de relógio**

# O relógio

- **Computador é um sistema digital síncrono**
- **Relógio**
  - **Determina quando ocorrem eventos no HW**
  - **Período (T) ou frequência (f)**
    - » **Exemplo: T = 0.25ns = 250 ps ou f = 4 GHz**




$$\text{tempo}_{\text{CPU}} = n \times T = \frac{n}{f}$$

# Relacionando as perspectivas

$$\text{tempo}_{\text{CPU}} = \text{ciclos}_{\text{CPU}} \times T = \frac{\text{ciclos}_{\text{CPU}}}{f}$$

# Exemplo

- **Problema**
  - **10s (comp. A, 4 GHz)**
  - **6s (comp. B, ?), mas $ciclos_B = 1{,}2 \times ciclos_A$**
- **Solução**

$$tempo_{CPU} = ciclos_{CPU} \times T = \frac{ciclos_{CPU}}{f}$$

# Exemplo

- **Problema**
  - **10s (comp. A, 4 GHz)**
  - **6s (comp. B, ?), mas $ciclos_B = 1{,}2 \times ciclos_A$**
- **Solução**

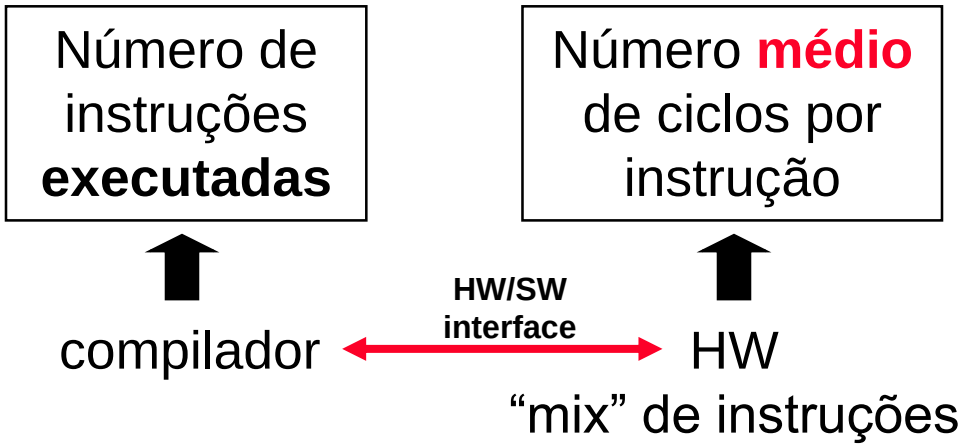
$$ciclos_{CPU}(A) = tempo_{CPU}(A) \times f(A) = 40 \times 10^9$$

$$tempo_{CPU}(B) = \frac{1.2 \times ciclos_{CPU}(A)}{f(B)} = \frac{1.2 \times 40 \times 10^9}{f(B)} = 6$$

$$f(B) = \frac{1.2 \times 40 \times 10^9}{6} = 8\,GHz$$

# Refinando a modelagem

$$ciclos_{CPU} = I \times CPI$$




$$tempo_{CPU} = I \times CPI \times T = \frac{I \times CPI}{f}$$

# Como determinar cada fator ?

$$\text{tempo}_{CPU} = I \times CPI \times T$$

SW: “profiler”, simulador
HW: contador

Simulação da implementação

manual do processador

$$\text{ciclos}_{CPU} = \sum_{i=1}^{n} (I_i \times CPI_i)$$

(Exemplo: p. 35-36)

# Cálculo do CPI médio

$$CPI = \frac{\sum_{i=1}^{n} I_i \times CPI_i}{I}$$

(média ponderada dos CPIs)

$$CPI = \sum_{i=1}^{n} \left(\frac{I_i}{I} \times CPI_i\right) = \sum_{i=1}^{n} (F_i \times CPI_i)$$

| Classe | CPI | Qde. | Fração |
|---|---|---|---|
| A | 5 | 20M | 0,33 |
| B | 2 | 30M | 0,50 |
| C | 4 | 6M | 0,10 |
| D | 4 | 4M | 0,07 |

$$CPI = \frac{5 \times 20 + 2 \times 30 + 4 \times 6 + 4 \times 4}{60} = \frac{10}{3}$$

$$CPI = 5 \times 0{,}33 + 2 \times 0{,}50 + 4 \times 0{,}10 + 4 \times 0{,}07 = 3{,}33$$

# Impacto no desempenho

- **Algoritmo**
  - **Afeta o número de instruções executadas**
    - » **Determina o número de instruções do programa**
    - » **Exemplo: mais passos, mais instruções**
  - **Pode afetar o CPI**
    - » **Pode favorecer instruções mais lentas ou rápidas**
    - » **Exemplo: uso de instruções de ponto flutuante ao invés de instruções inteiras**

# Impacto no desempenho

- **Linguagem**
  - **Afeta o número de instruções**
    - » **Determina as instruções-fonte a serem traduzidas em instruções do processador-alvo**
    - » **Exemplo: teste automático de limites de arranjo (Lab 03)**
  - **Afeta o CPI**
    - » **Forte suporte a abstrações de dados requer chamadas indiretas, que deterioram o CPI**
  - **Exemplo: Invocação de métodos em Java**
    - » **Seção 2.15 do livro-texto (CD):**
      - **Teste se ponteiro nulo (beq, bne)**
      - **Carga do endereço da tabela de métodos (lw)**
      - **Carga do endereço do método apropriado (lw)**
      - **Desvio para endereço em registrador (jral)**

# Impacto no desempenho

- **Compilador**
  - **Afeta o número de instruções executadas**
    - » **Determina a tradução de comandos-fonte em instruções do processador-alvo**
    - » **Exemplo: eliminação de redundâncias, eliminação de instruções de desvio**
  - **Pode afetar o CPI**
    - » **Determina a proporção de instruções de cada tipo**
    - » **Determina a ordem das instruções (*pipeline*)**
    - » **Influencia a localidade de acesso à memória (cache)**
    - » **Exemplo: escalonamento de código, *loop unrolling***

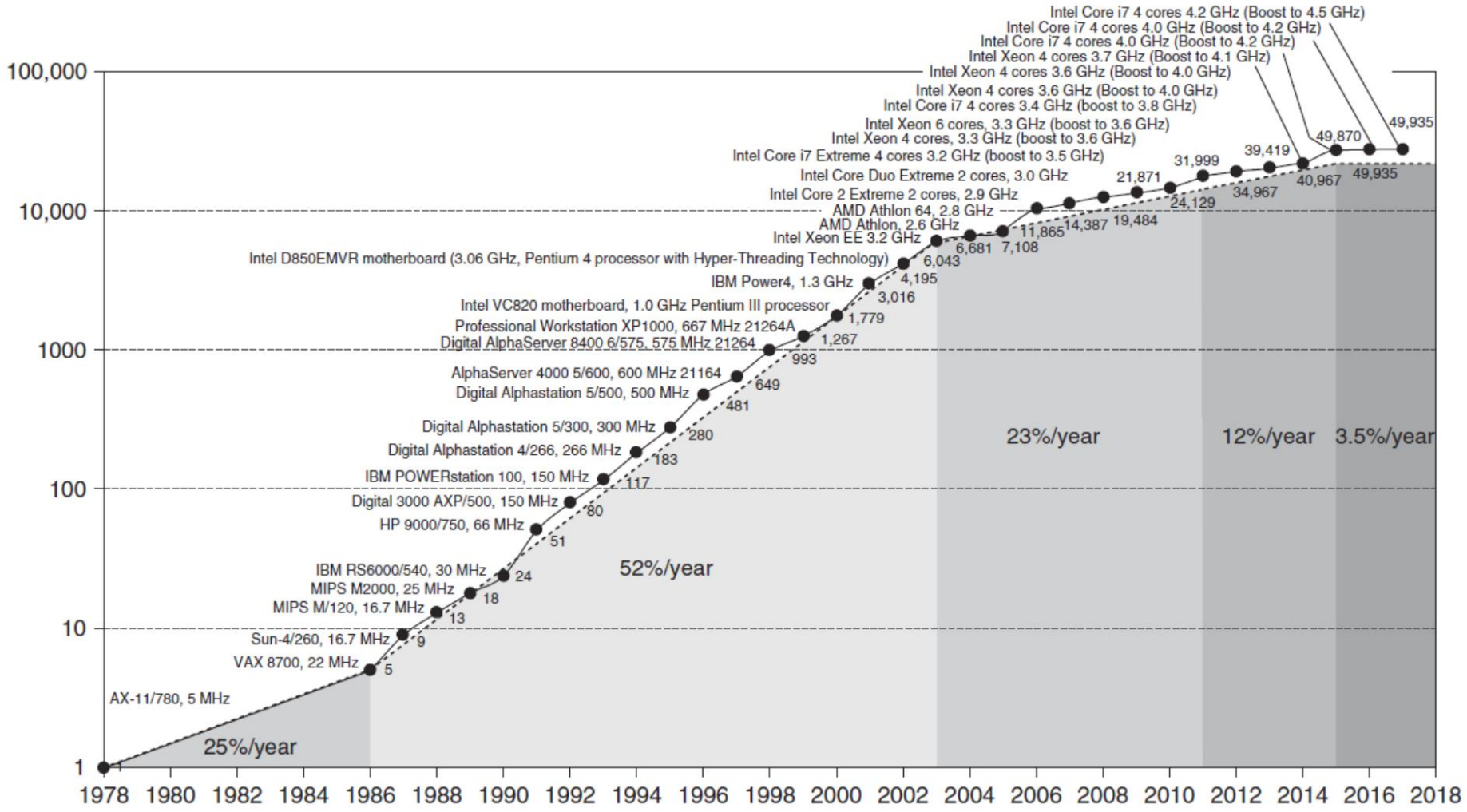
# Impacto no desempenho

- **ISA**
  - **Afeta o número de instruções executadas**
    - » **Afeta a seleção de instruções pelo compilador**
    - » **Que instruções do processador-alvo são necessárias para executar uma dada função ?**
    - » **Exemplos: push/pop; a = a + A[i]**
  - **Afeta o CPI**
    - » **Determina o custo em ciclos de cada instrução**
    - » **Exemplo: modos de endereçamento complexos**
  - **Afeta a frequência**
    - » **A simplicidade das instruções permite organizar o sistema digital com menores período de relógio, para uma dada tecnologia de fabricação**
    - » **Exemplo: níveis de lógica na decodificação de instruções**

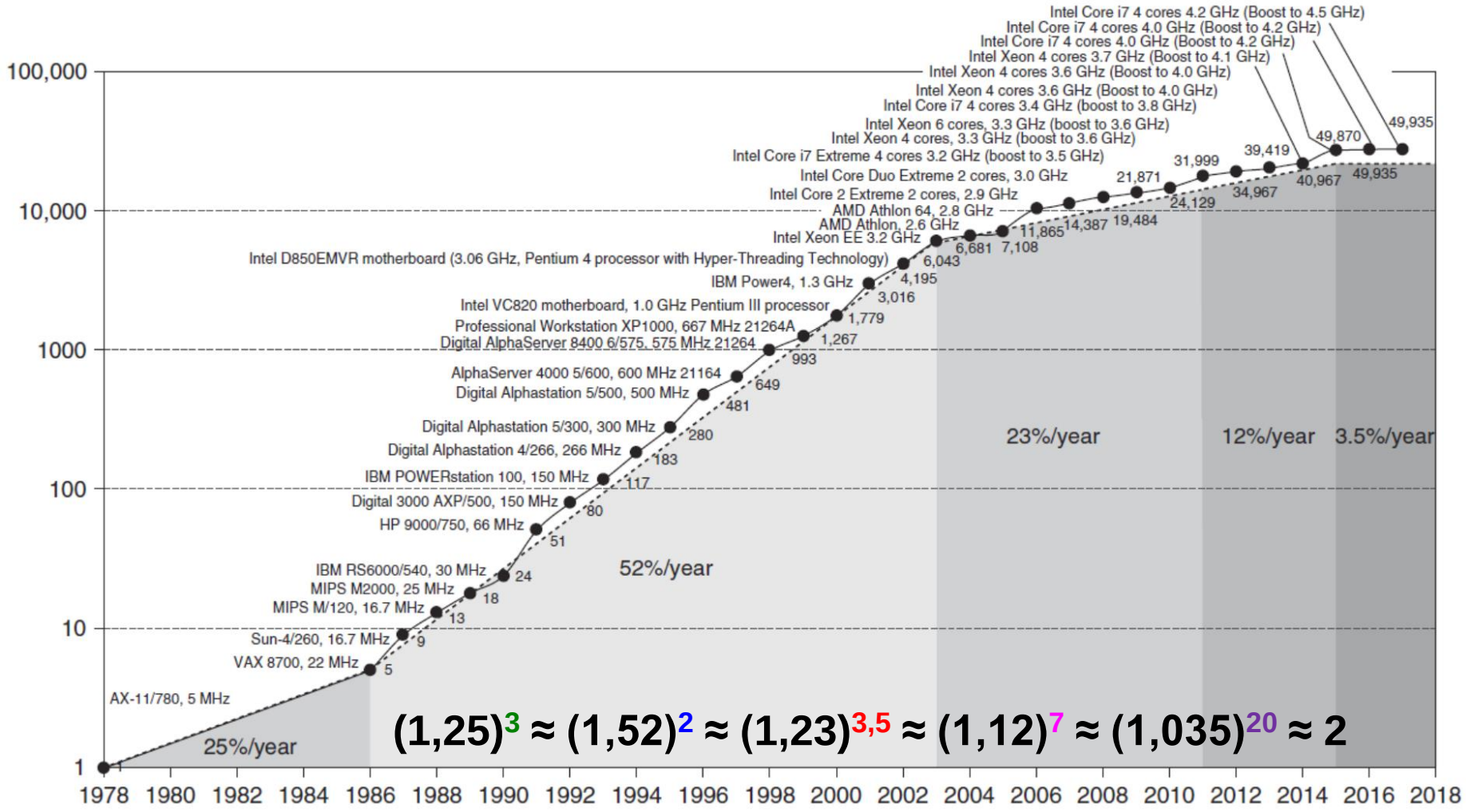
# Discussão

- **Você está satisfeita/satisfeito com o crescimento do desempenho dos PCs nos últimos 5 anos?**
- **O mercado está satisfeito com o crescimento do desempenho nos últimos 5 anos?**
- **O desempenho em processadores *multicore* é satisfatório?**
- **Os crescimento do desempenho em processadores *multicore* é satisfatório ?**

# Limites ao crescimento do desempenho



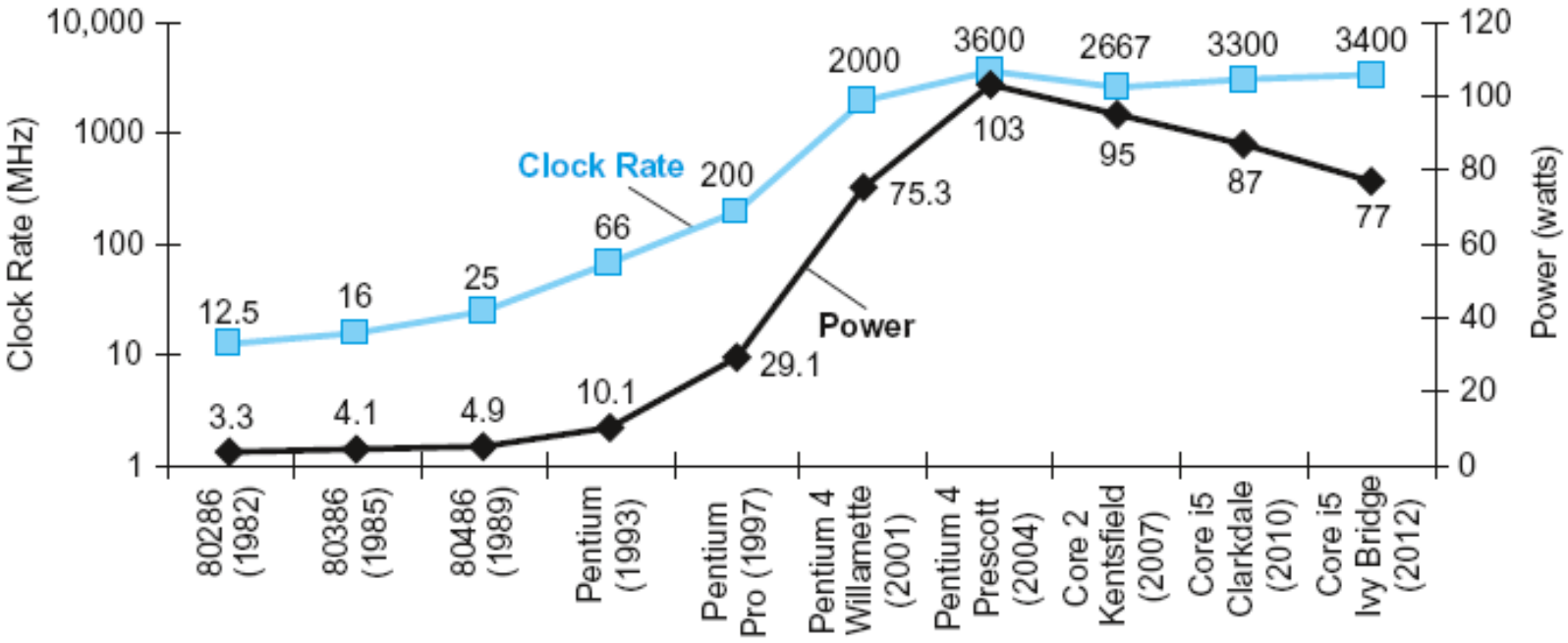
# Limites ao crescimento do desempenho




$$(1{,}25)^{3} \approx (1{,}52)^{2} \approx (1{,}23)^{3{,}5} \approx (1{,}12)^{7} \approx (1{,}035)^{20} \approx 2$$

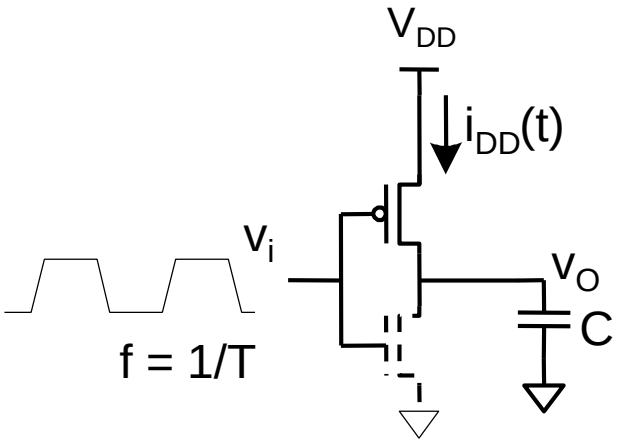
**Lei de Moore, RISC+ILP, barreira de potência (fim do Dennard Scaling) + limite de ILP, limite do número de cores úteis (serialização), fim da Lei de Moore**

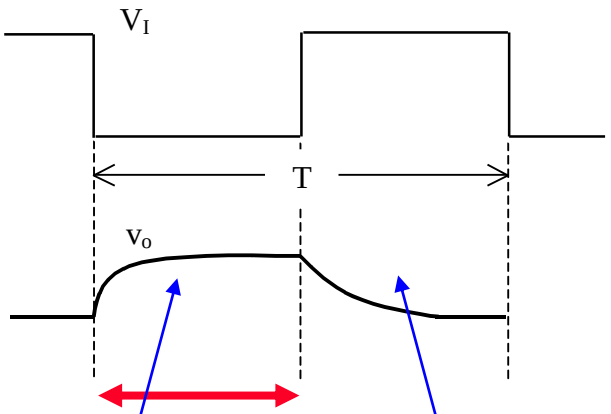
# A barreira da potência




Qual a relação entre P e f ?

# Revisão de tecnologia CMOS



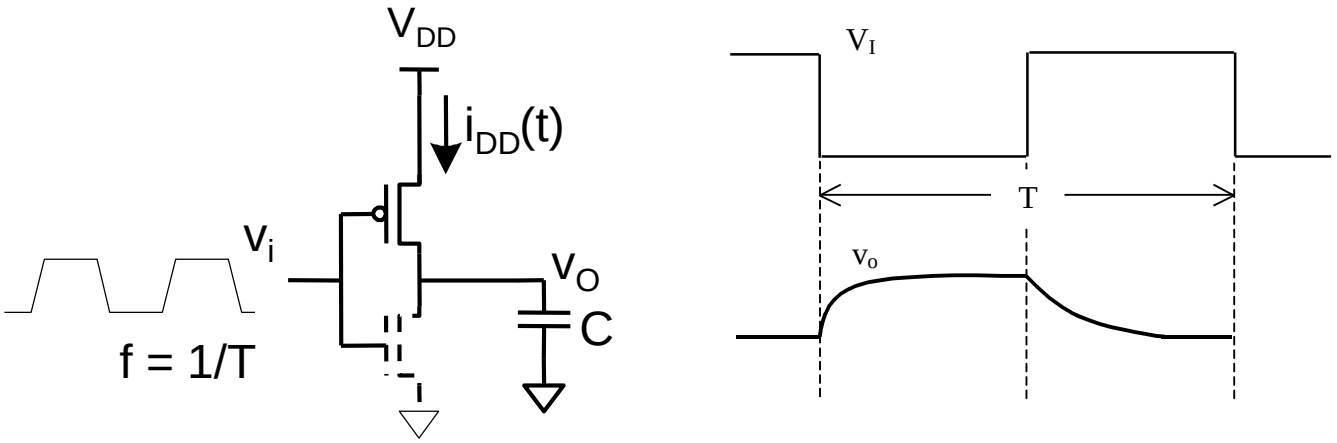




**Energia drenada da fonte quando saída 0→1**

E/2 dissipada na forma de calor através do **P**MOS

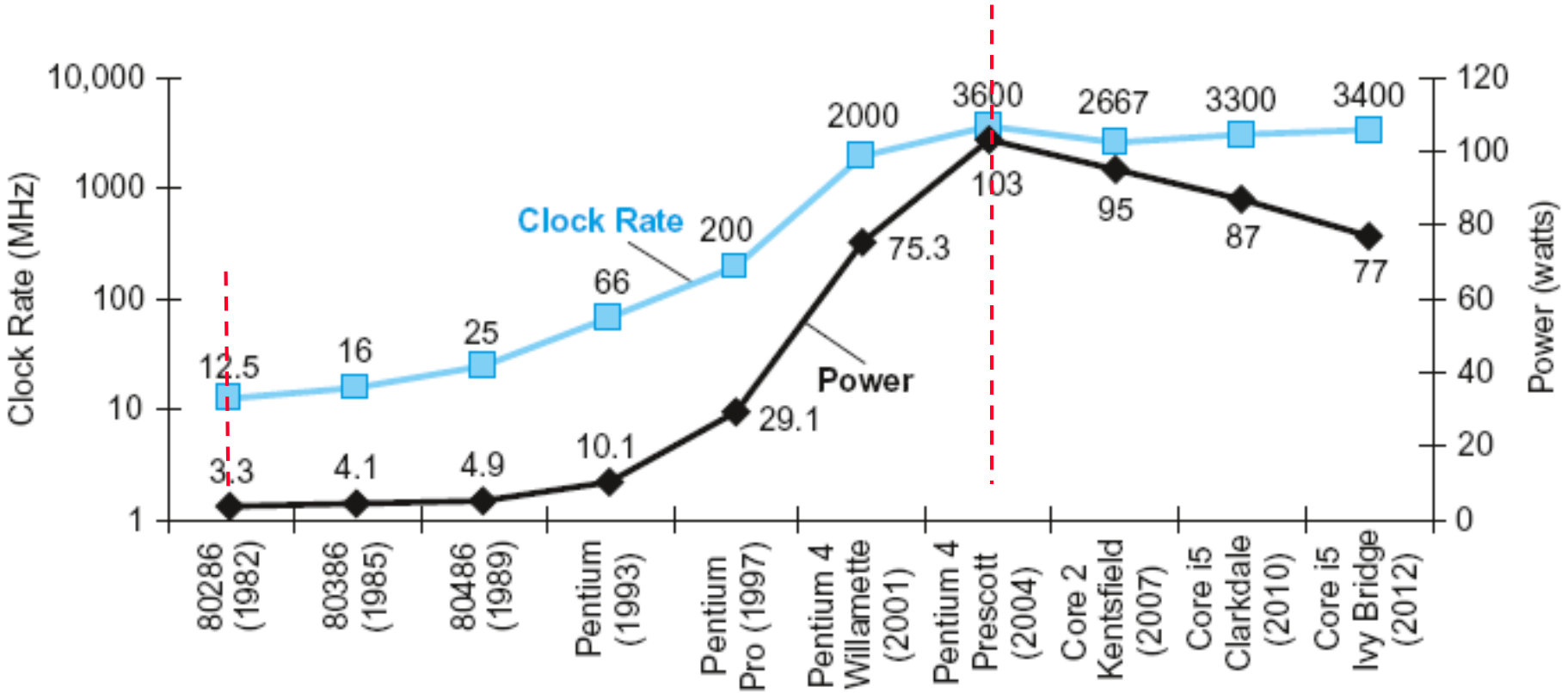
E/2 dissipada na forma de calor através do **N**MOS

# Revisão de tecnologia CMOS




$$P_{din} = C \times V_{DD}^{\ 2} \times f_{0 \rightarrow 1}$$

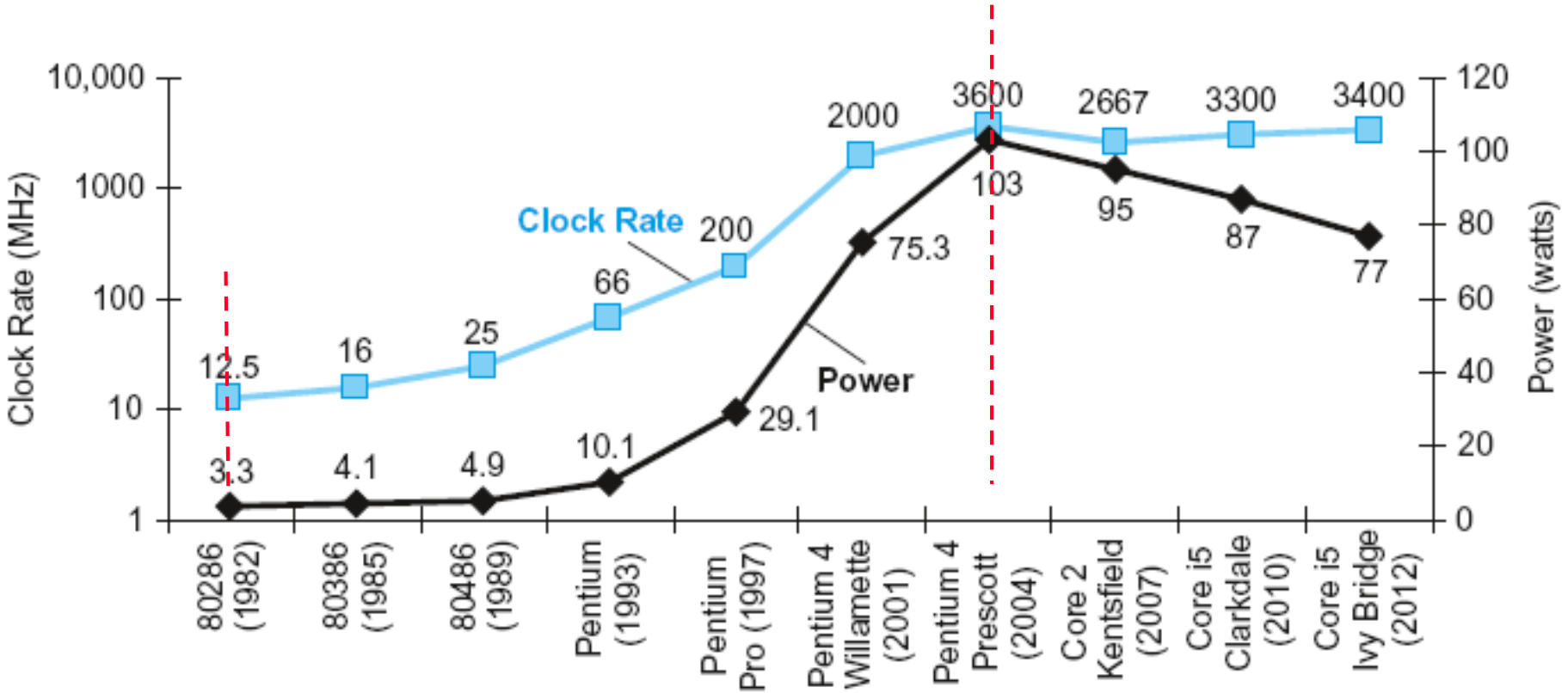
# Correlação entre frequência e potência




$$P_{din} = C \times V_{DD}^{\ 2} \times f_{0 \rightarrow 1}$$

Por que P cresceu só 30x enquanto f cresceu 300x ?

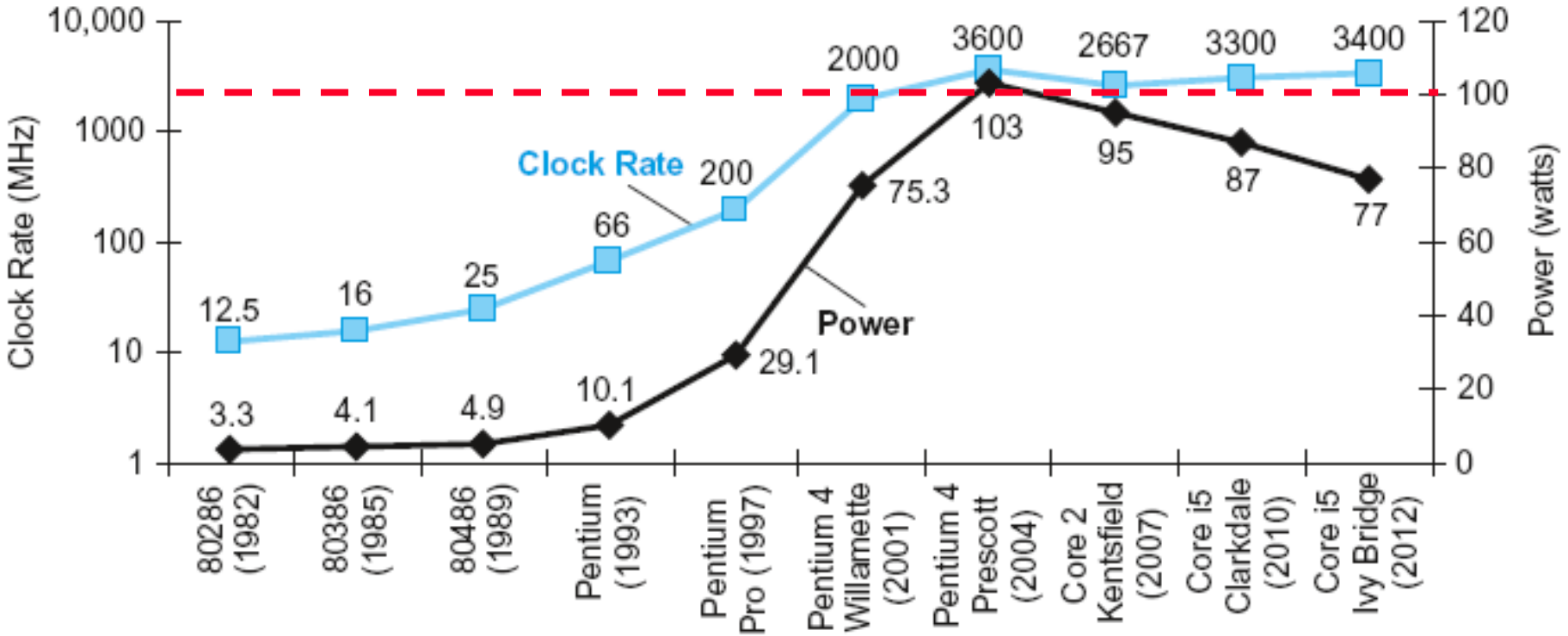
# Correlação entre frequência e potência




$$P_{din} = C \times V_{DD}^{\ 2} \times f_{0 \rightarrow 1}$$

Porque $V_{DD}$ caiu de 5V para 1V no mesmo período!

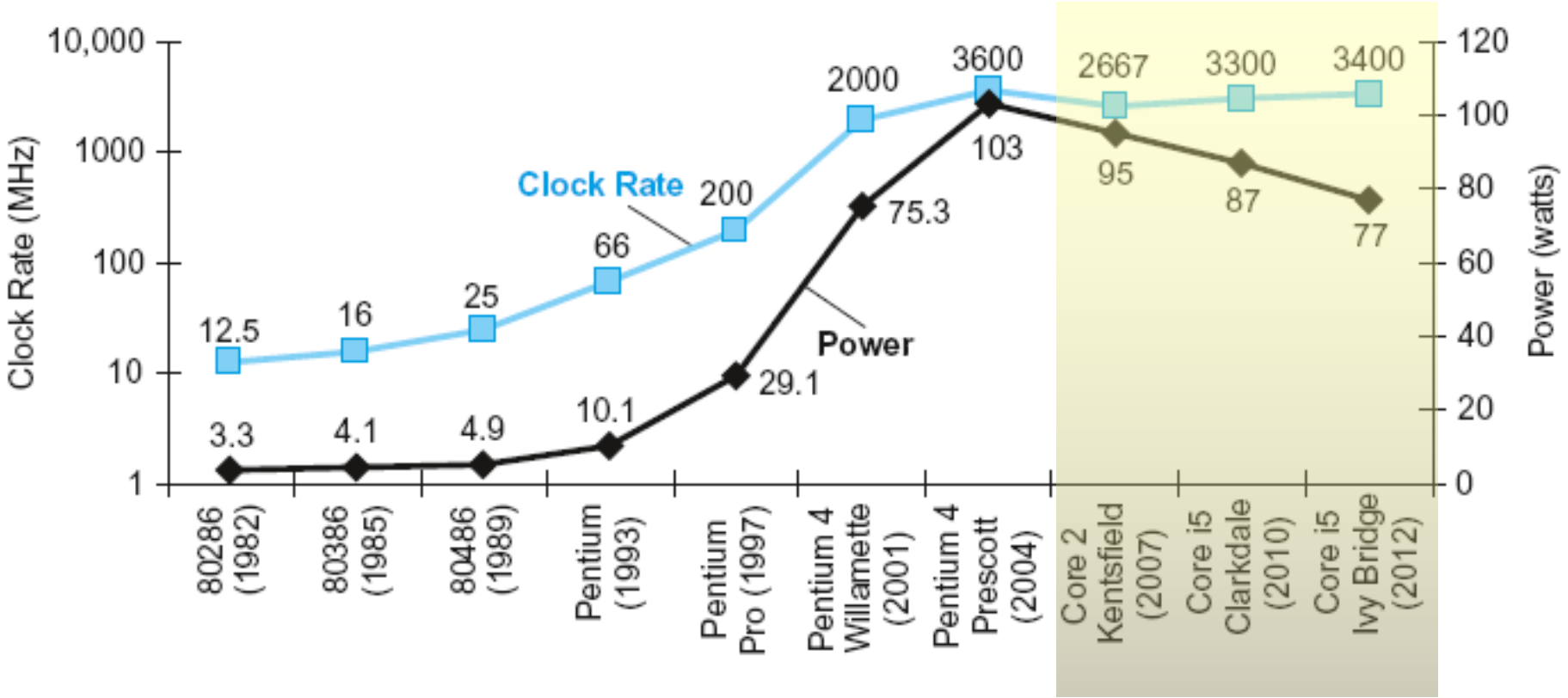
# A barreira da potência




$$P_{din} = C \times V_{DD}^{\ 2} \times f_{0 \rightarrow 1}$$

Para PCs, há um limite prático de resfriamento que restringe a potência a cerca de 100W

# Consequência: mudança de paradigma




$$P_{din} = C \times V_{DD}^{\ 2} \times f_{0 \rightarrow 1}$$

Do uniprocessador ao multiprocessador em um único chip!

(**CMP**: ***C**hip **M**ulti-**P**rocessing*)

# Consequência do novo paradigma

- **Requer programação paralela explícita**
  - **Ao contrário do paralelismo entre instruções**
    - » **Onde HW executa múltiplas instruções por vez**
    - » **Paralelismo escondido do programador**
      - **Compilador e HW fazem quase todo o trabalho**
  - **Dificuldades do novo paradigma:**
    - » **Programar para desempenho (mais difícil)**
      - **Convencional: deve-se garantir programa correto, resolução de problema importante, interface adequada**
      - **Agora tem que se programar para velocidade**
    - » **Balancear carga**
    - » **Otimizar comunicação e sincronização**

# HW/SW para exploração de paralelismo

- **Suporte a *thread-level parallelism***
  - **Instruções para sincronização entre *threads***
  - **Acesso à memória compartilhada entre *threads***
    - » **Coerência de cache**
  - **Execução paralela de múltiplas *threads* dentro de um programa**
- **Suporte a *instruction-level parallelism***
  - **Execução paralela de múltiplas instruções dentro de uma *thread***
- **Suporte a data-level parallelism**
  - **Execução paralela de múltiplas operações dentro de uma instrução**

# Desempenho