MC542

Organização de Computadores
Teoria e Prática

2006

Prof. Paulo Cesar Centoducatte

ducatte@ic.unicamp.br

www.ic.unicamp.br/~ducatte

MC542

Arquitetura de Computadores

Introdução

"Computer Organization and Design:
The Hardware/Software Interface"
(Capítulo 1)

Sumário

- Introdução
- Tarefas do Projetista
- Tecnologia e tendências na computação
- Custo, Preço e suas tendências
- Outros Aspectos

O que é Arquitetura de Computadores (AC)?

- 1950s a 1960s: Cursos de AC

  Aritmética Computacional

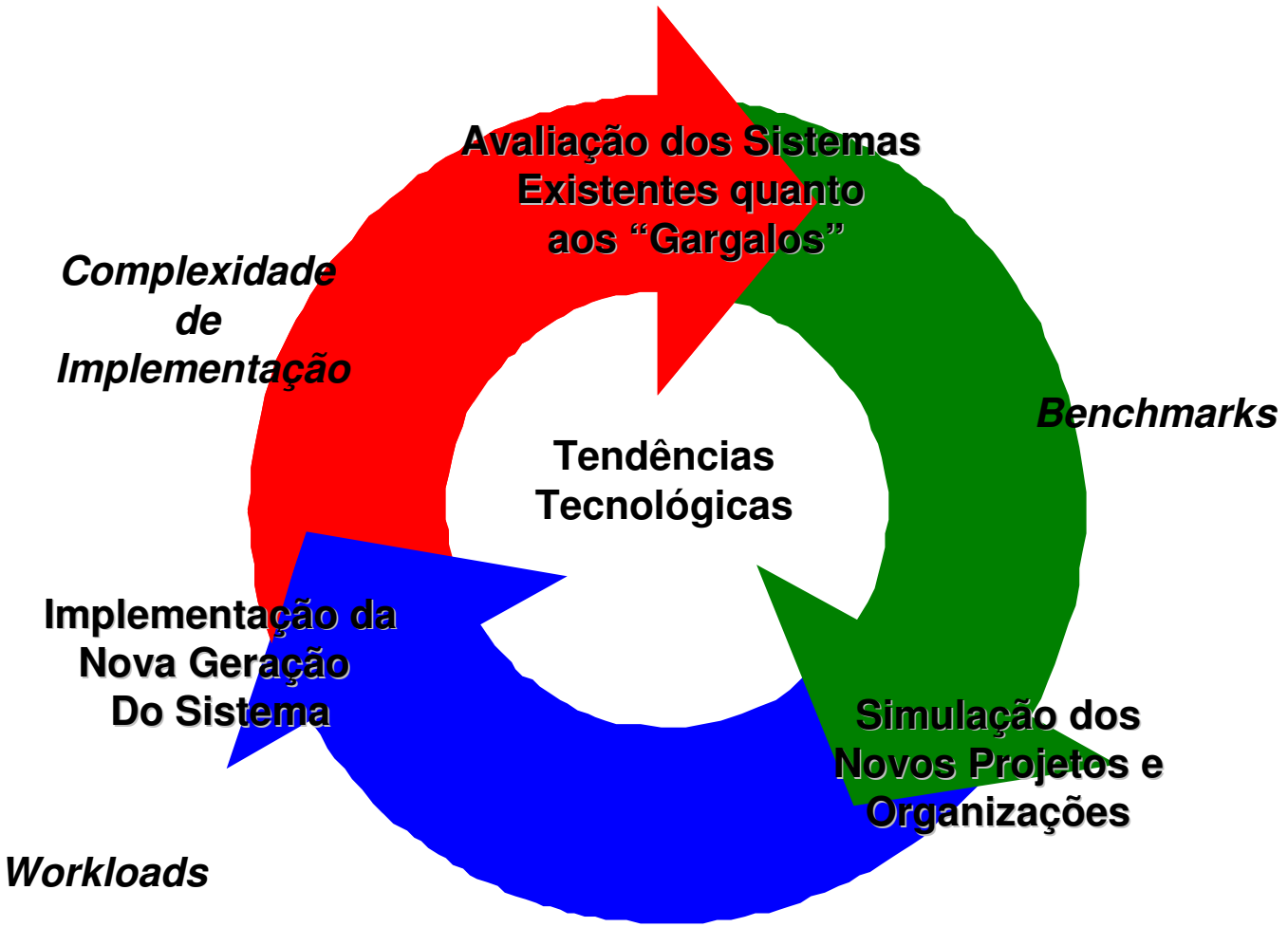
- 1970s a meados dos anos 1980s: Cursos de AC

  Projeto do Conjunto de Instruções (ISA), especialmente voltado para compiladores

- 1990s a 2000s: Cursos de AC

  Projeto de CPU, Sistemas de Memórias, Sistemas de I/O, Multiprocessadores.

Tarefas do Projetista



Tendências

- Gordon Moore (fundador da Intel), em 1965 observou que o número de transistores em um chip dobrava a cada ano (Lei de Moore)

  Continua valida até os dias de hoje

- O desempenho dos processadores, medidos por diversos benchmarks, também tem crescido de forma acelerada.

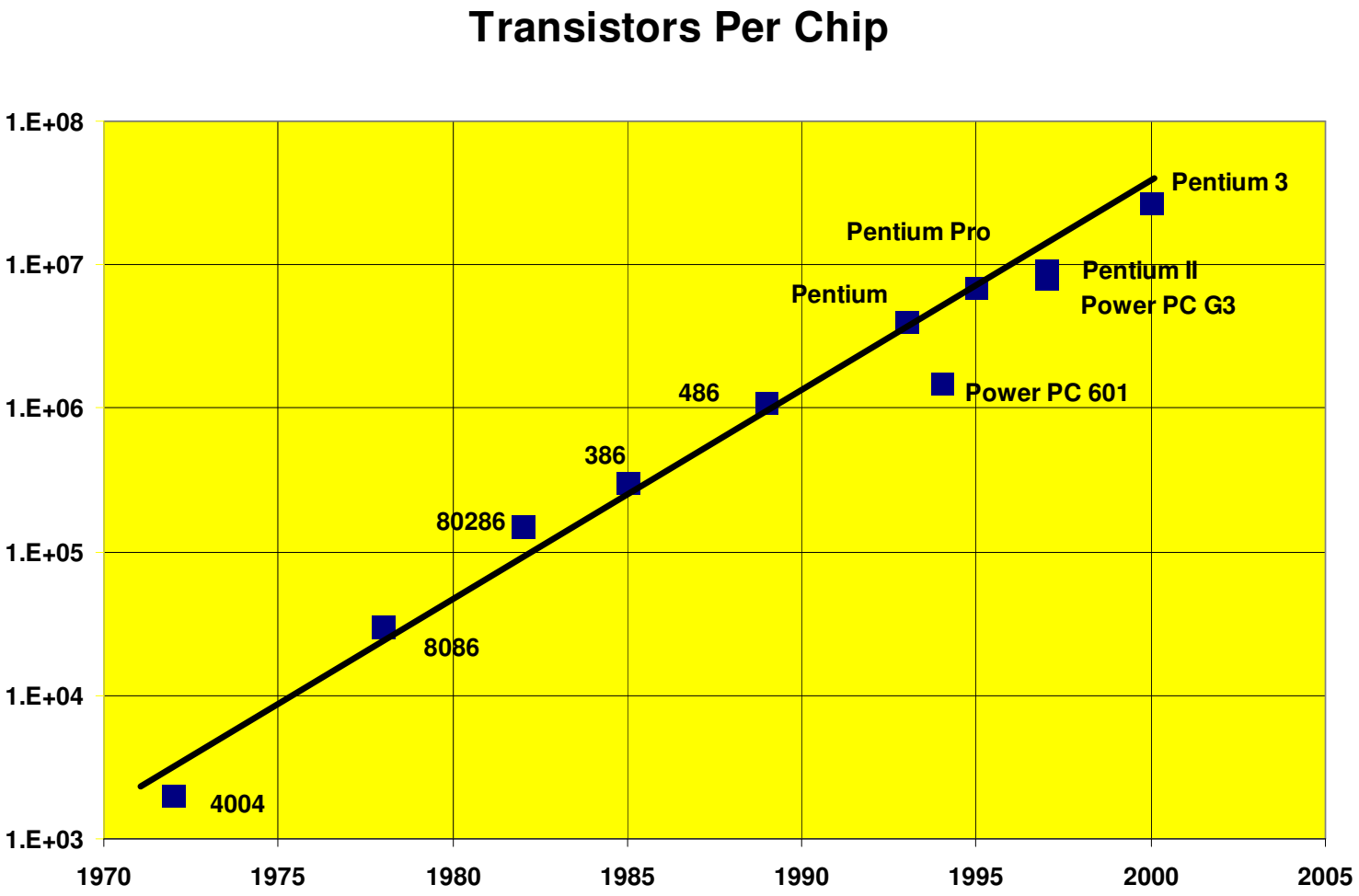
- A capacidade das memórias tem aumentado significativamente nos últimos 20 anos

  (E o custo reduzido)

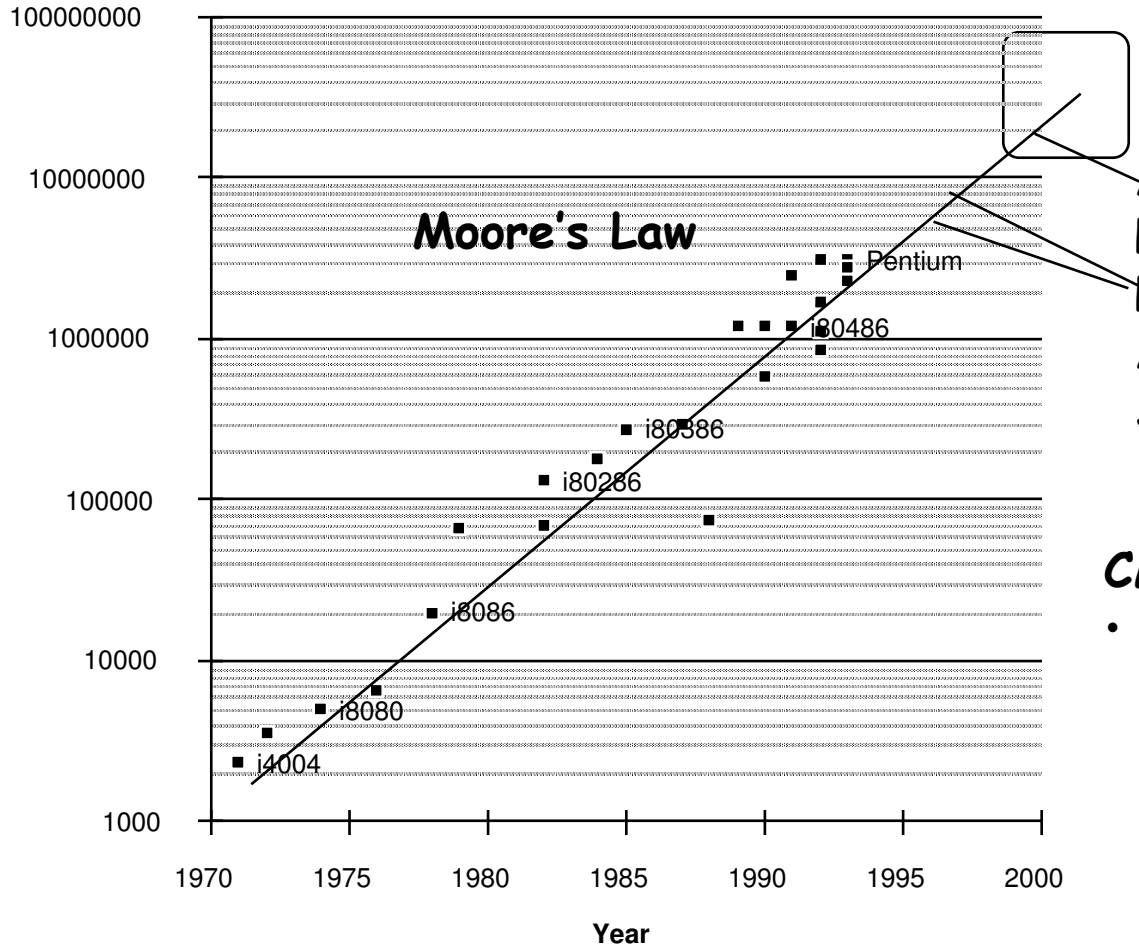
Qual a Razão desta evolução nos últimos anos?

- Desempenho
  - Avanços tecnológicos
    - » Domínio de CMOS sobre as tecnologias mais antigas (TTL, ECL) em custo e desempenho
  - Avanços nas arquiteturas
    - » RISC, superscalar, VLIW, RAID, …
- Preço: Baixo custo devido
  - Desenvolvimento mais simples
    - » CMOS VLSI: sistemas menores, menos componentes
  - Alto volume (escala)
- .....

Tendências
Lei de Moore



Tendência Tecnológica: Capacidade Microprocessadores

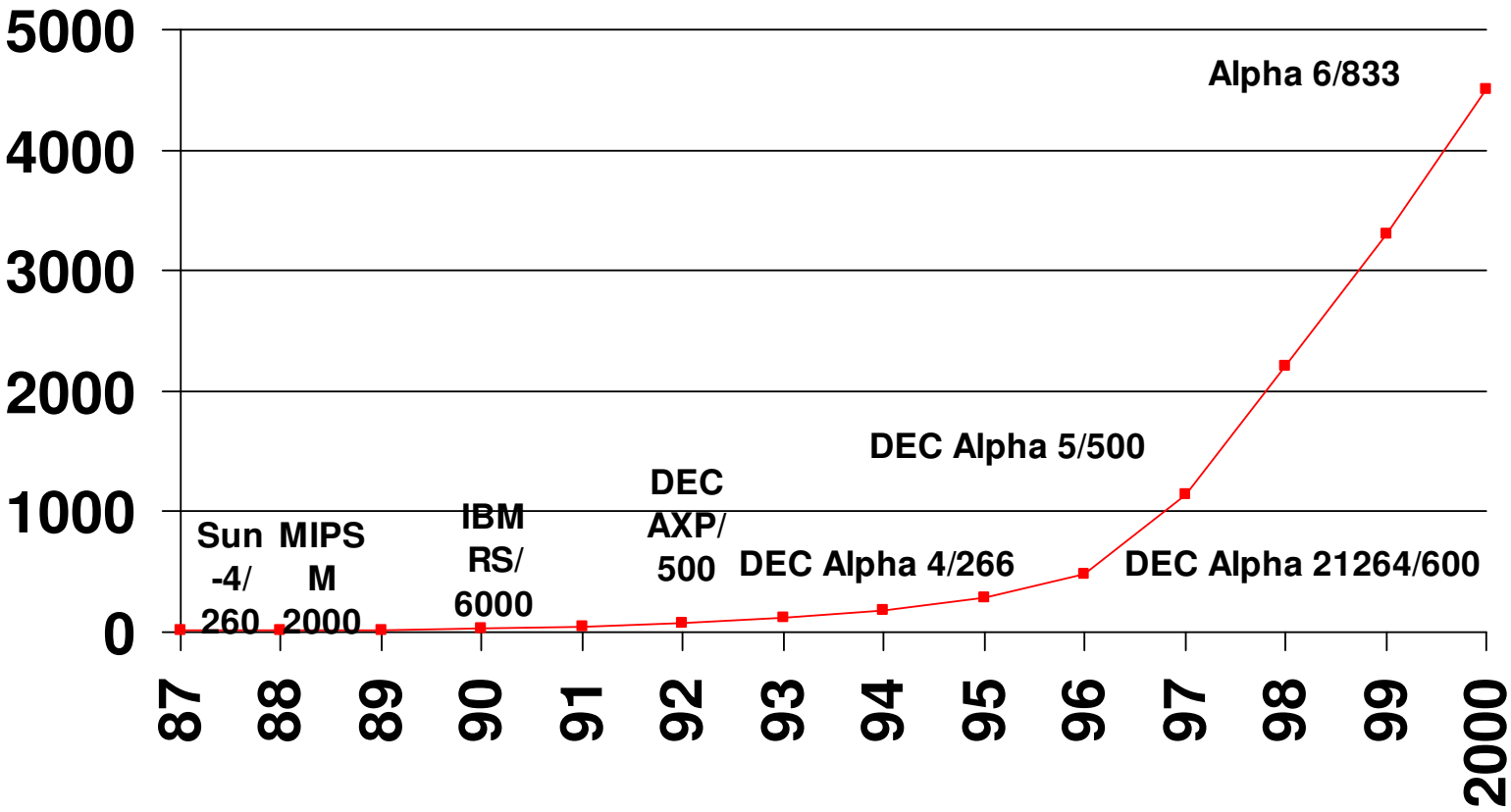


Alpha 21264: 15 million
Pentium Pro: 5.5 million
PowerPC 620: 6.9 million
Alpha 21164: 9.3 million
Sparc Ultra: 5.2 million

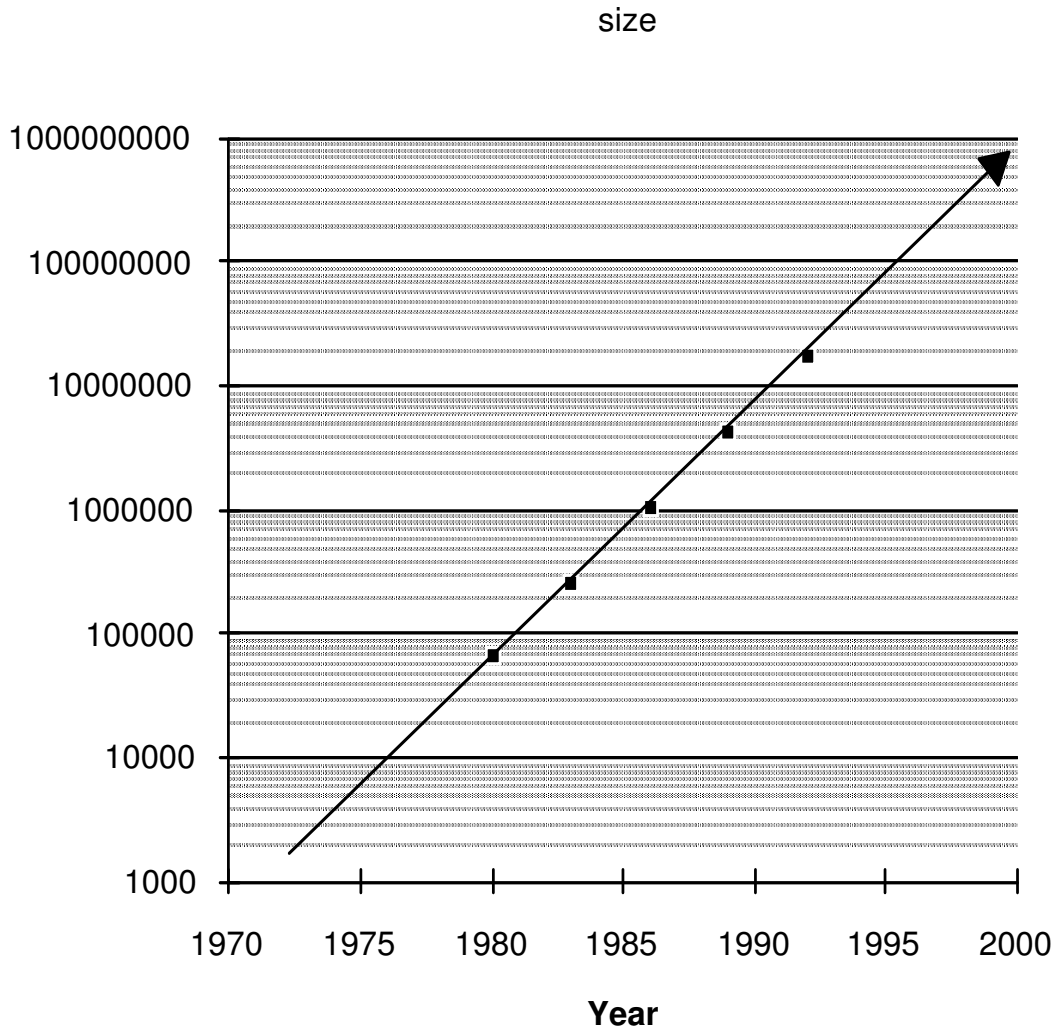
CMOS:

- **Die size: 2X a cada 3 anos**

Tendências
Desempenho dos processadores



Tendências
Capacidade das Memórias



ano	Mbyte	cycle time
1980	0.0625	250 ns
1983	0.25	220 ns
1986	1	190 ns
1989	4	165 ns
1992	16	145 ns
1996	64	120 ns
2000	256	100 ns

Tendências
Velocidade

- Para a CPU o crescimento da velocidade tem sido muito acelerado
- Para Memória e disco o crescimento da velocidade tem sido modesto

Isto tem levado a mudanças significativas nas arquiteturas, SO e mesmo nas práticas de programação.

	Capacidade	Speed (latency)
Lógica	2x em 3 anos	2x em 3 anos
DRAM	4x em 3 anos	2x em 10 anos
Disco	4x em 3 anos	2x em 10 anos

Medidas
?

- Como descrever em forma numérica o desempenho dos computadores?

- Quais ferramentas (ou qual ferramental) usar para realizar as medidas?

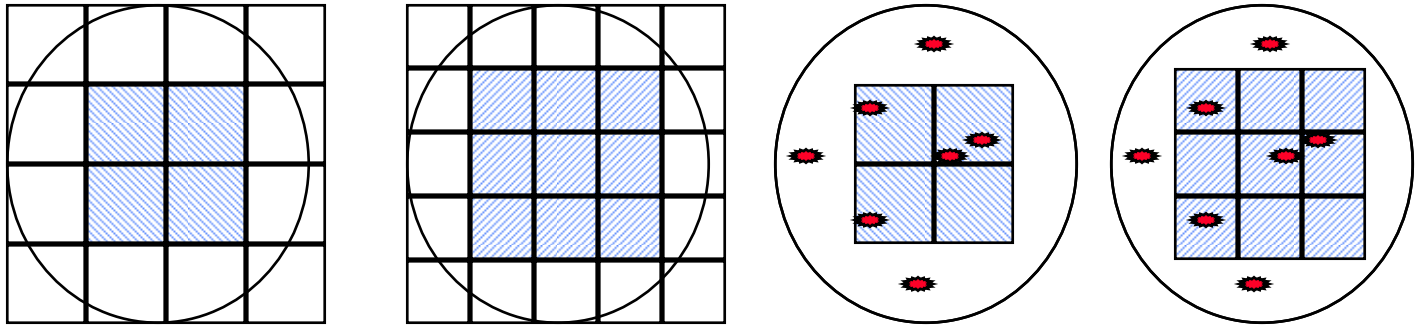
Custo, Preço e suas tendências

Custo de Circuito Integrado (IC)

$$\text{IC cost} = \frac{\text{Die cost} + \text{Testing cost} + \text{Packaging cost}}{\text{Final test yield}}$$

$$\text{Die cost} = \frac{\text{Wafer cost}}{\text{Dies per Wafer} \times \text{Die yield}}$$

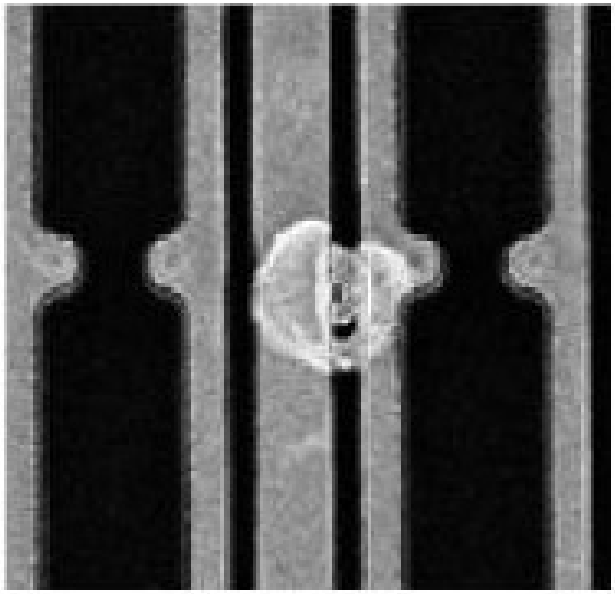
$$\text{Dies per wafer} = \frac{\pi\,(\text{Wafer\_diam}/2)^2}{\text{Die\_Area}} - \frac{\pi \times \text{Wafer\_diam}}{\sqrt{2 \cdot \text{Die\_Area}}} - \text{Test\_Die}$$

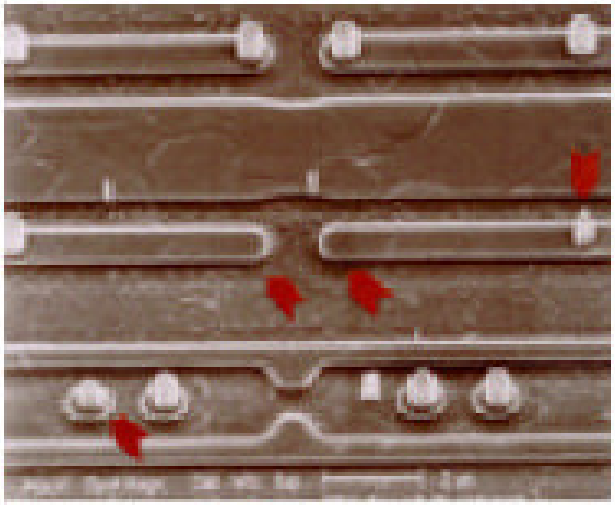


$$\text{Die Yield} = \text{Wafer\_yield} \times \left\{1 + \left(\frac{\text{Defect\_Density} \times \text{Die\_area}}{\alpha}\right)^{-\alpha}\right\}$$

Custo do Die é proporcional à (área do die)^4

Integrated Circuits Yield -Defects





Examplos Reais

Chip	Metal layers	Line width	Wafer cost	Defect /cm^2	Area mm^2	Dies/ wafer	Yield	Die Cost
386DX	2	0.90	$900	1.0	43	360	71%	$4
486DX2	3	0.80	$1200	1.0	81	181	54%	$12
PowerPC 601	4	0.80	$1700	1.3	121	115	28%	$53
HP PA 7100	3	0.80	$1300	1.0	196	66	27%	$73
DEC Alpha	3	0.70	$1500	1.2	234	53	19%	$149
SuperSPARC	3	0.70	$1700	1.6	256	48	13%	$272
Pentium	3	0.80	$1500	1.5	296	40	9%	$417

– From "Estimating IC Manufacturing Costs," by Linley Gwennap, *Microprocessor Report*, August 2, 1993, p. 15

Abordagem Quantitativa

- Faça o caso comum ser mais rápido

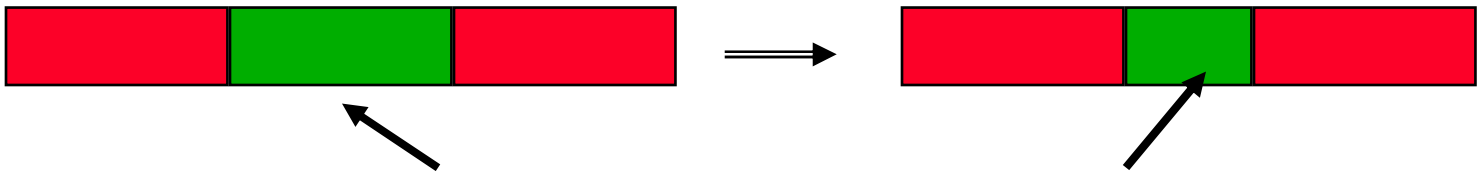
- Amdahl's Law:
  - Relaciona o speedup total de um sistema com o speedup de uma porção do sistema

O speedup no desempenho obtido por uma melhoria é limitado pela fração do tempo na qual a melhoria é utilizada

Abordagem Quantitativa
Amdahl's Law

Speedup devido a uma melhoria E:

$$Speedup\,(E) = \frac{Execution\_Time\_Without\_Enhancement}{Execution\_Time\_With\_Enhancement} = \frac{Performance\_With\_Enhancement}{Performance\_Without\_Enhancement}$$



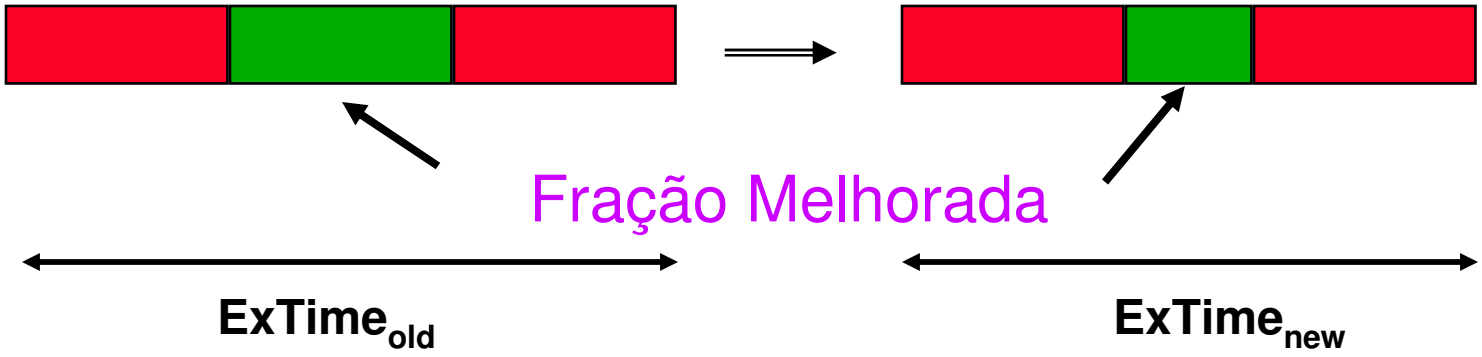
Fração melhorada

Suponha que a melhoria E acelera a execução de uma fração F da tarefa de um fator S e que o restante da tarefa não é afetado pela melhoria E.

Abordagem Quantitativa
Amdahl's Law

$$ExTime_{new} = ExTime_{old} \times \left[(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}} \right]$$

$$Speedup_{overall} = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$$



Abordagem Quantitativa
Amdahl's Law

- Exemplo: Suponha que as instruções de ponto flutuante foram melhorada e executam 2 vezes mais rápidas, porém somente 10% das instruções, em um programa, são FP

$$ExTime_{new} = ExTime_{old} \times (0.9 + 0.1/2) = 0.95 \times ExTime_{old}$$

$$Speedup_{overall} = \frac{1}{0.95} = 1.053$$

Amdahl's Law

Execução de um programa em N processadores

$$\text{Fraction}_{\text{enhanced}} = \text{parallelizable part of program}$$

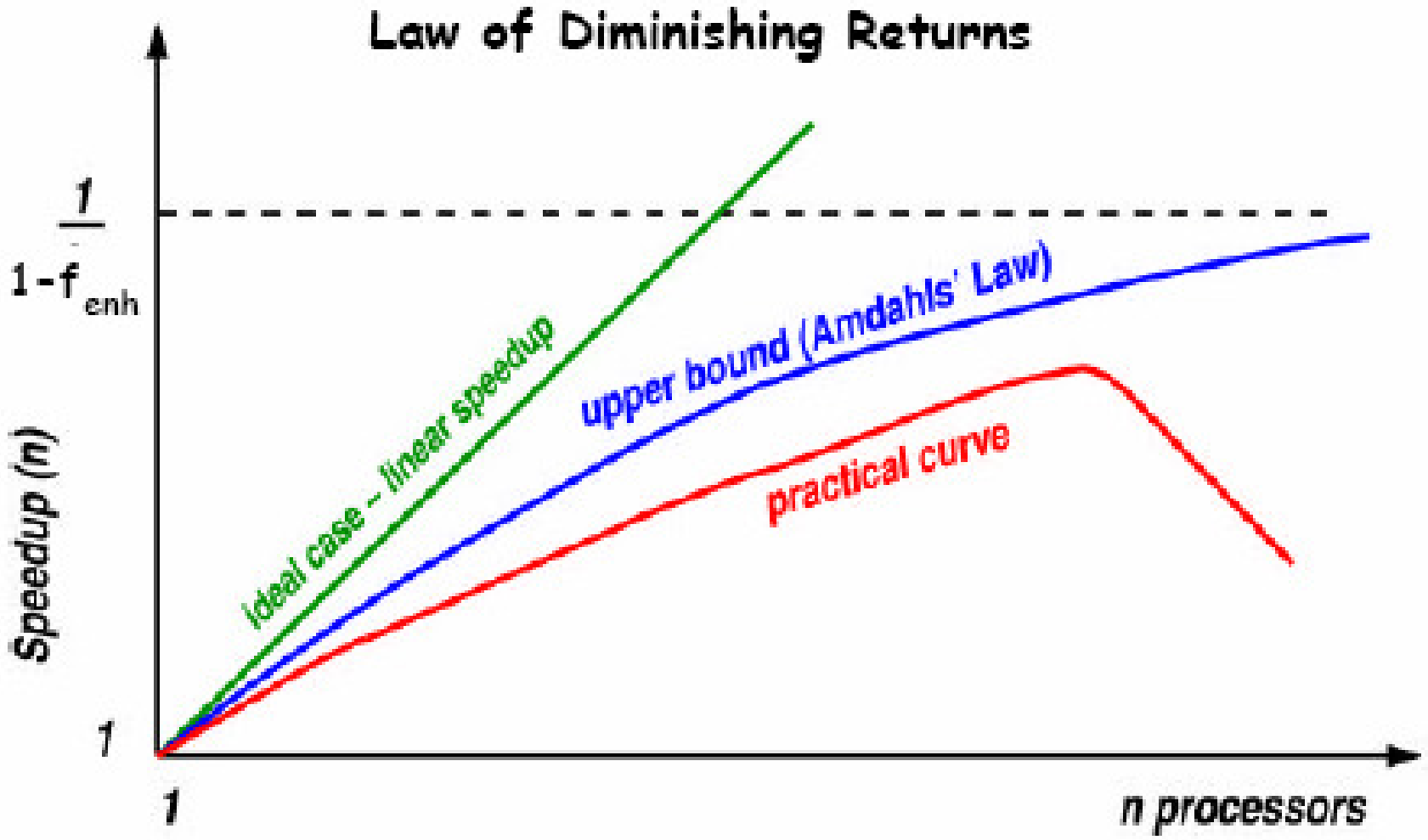
$$\text{Speedup}_{\text{enhanced}} = n$$

$$\text{ExTime}_{\text{new}} = \text{ExTime}_{\text{old}} \left(1 - \text{Fraction}_{\text{enhanced}}\right) + \frac{\text{ExTime}_{\text{old}} \times \text{Fraction}_{\text{enhanced}}}{n}$$

$$\text{Speedup}_{\text{overall}} = \frac{\text{ExTime}_{\text{old}}}{\text{ExTime}_{\text{new}}} = \frac{1}{\left(1 - \text{Fraction}_{\text{enhanced}}\right) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

$$\lim_{n \to \infty} \text{Speedup}_{\text{overall}} = 1 \,/\, \left(1 - \text{Fraction}_{\text{enhanced}}\right)$$

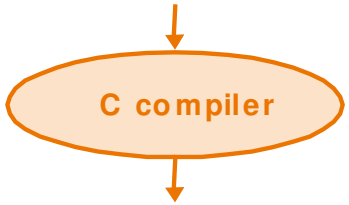
Amdahl's Law - Graph



Níveis de Abstração

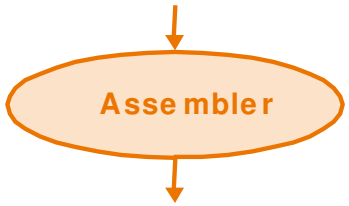
High-level language program (in C)

```
swap(int v[], int k)
{int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```



Assembly language program (for MIPS)

```
swap:
    muli $2, $5,4
    add  $2, $4,$2
    lw   $15, 0($2)
    lw   $16, 4($2)
    sw   $16, 0($2)
    sw   $15, 4($2)
    jr   $31
```



Binary machine language program (for MIPS)

```
00000000101000010000000000011000
00000000100011100001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```