

Infraestrutura de Hardware

Desempenho

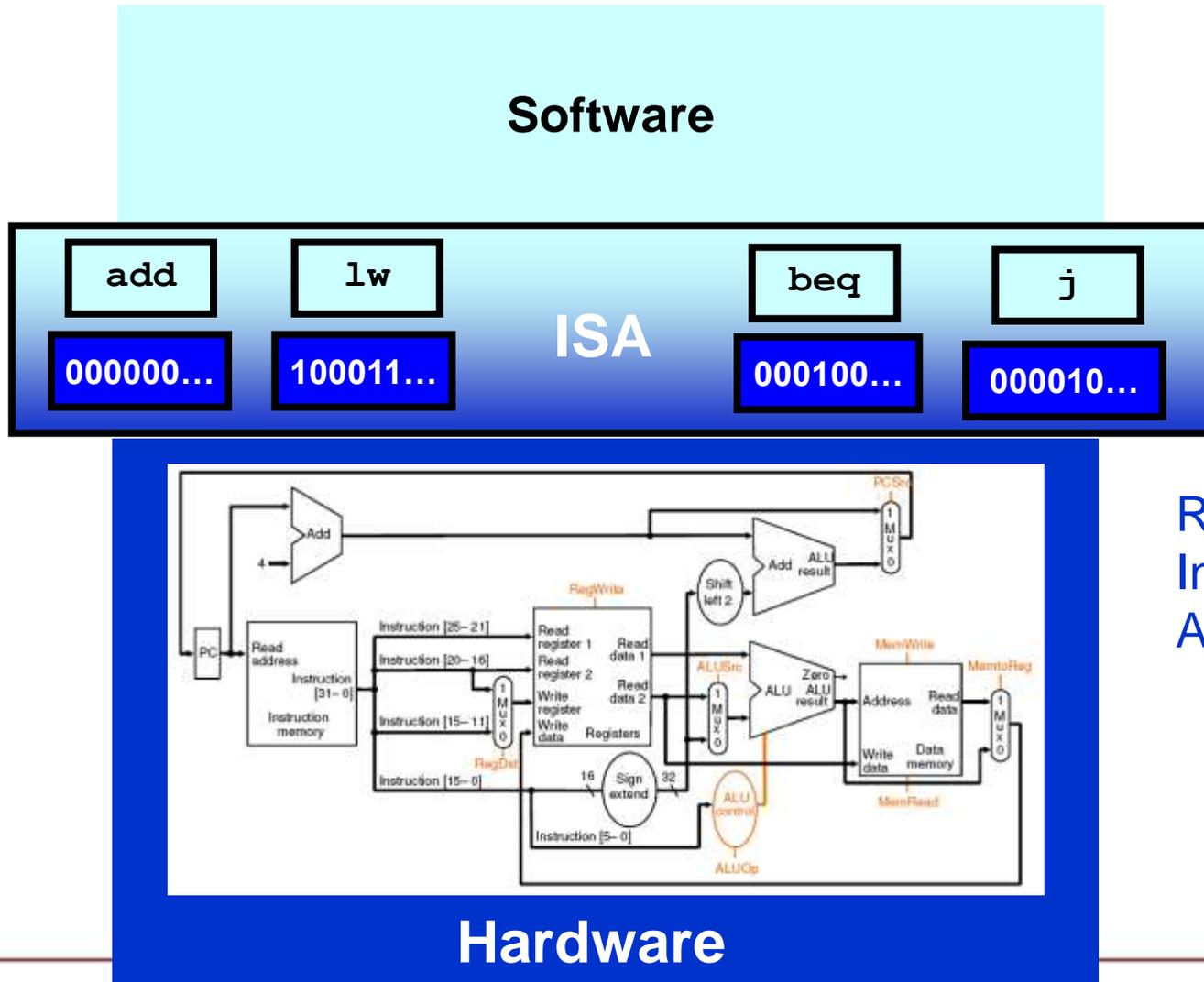


UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Perguntas que Devem ser Respondidas ao Final do Curso

- Como um programa escrito em uma linguagem de alto nível é entendido e executado pelo HW?
- Qual é a interface entre SW e HW e como o SW instrui o HW a executar o que foi planejado?
- **O que determina o desempenho de um programa e como ele pode ser melhorado?**
- Que técnicas um projetista de HW pode utilizar para melhorar o desempenho?

Interface HW/SW: ISA



Repertório de Instruções da Arquitetura

Analizando Desempenho

- Desempenho é um importante fator de qualidade de um sistema
- Análise difícil
 - Tamanho e complexidade de softwares atuais
 - Técnicas sofisticadas utilizadas no projeto do hardware para aumento de desempenho
- Diferentes métricas podem ser utilizadas dependendo do tipo de aplicação
- Diferentes aspectos do sistema podem ter impacto maior no desempenho
 - Entendimento de como os diferentes aspectos afetam o sistema é essencial para desenvolver software ou projetar hardware com melhor desempenho

Fatores Que Influenciam Desempenho

- **Algoritmo**

Determina número de operações executadas

- **Linguagem de Programação, compilador, arquitetura**

Determina número de instruções de máquina executadas por operação

- **Processador e estrutura de memória**

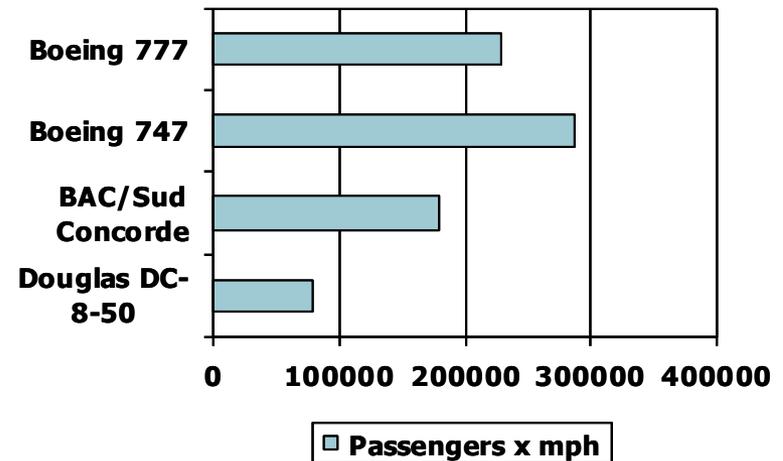
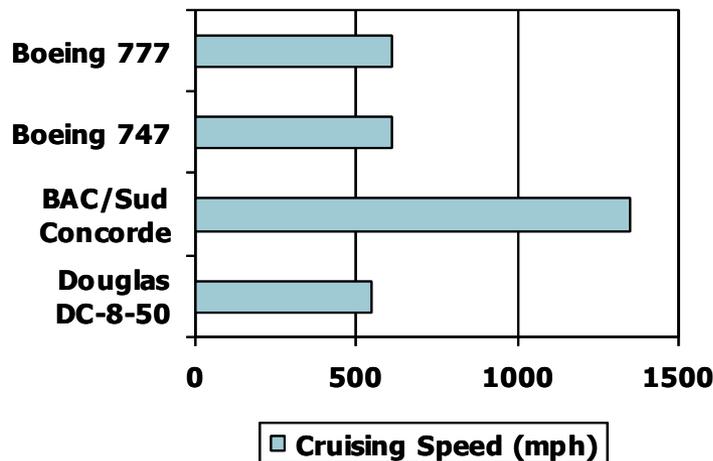
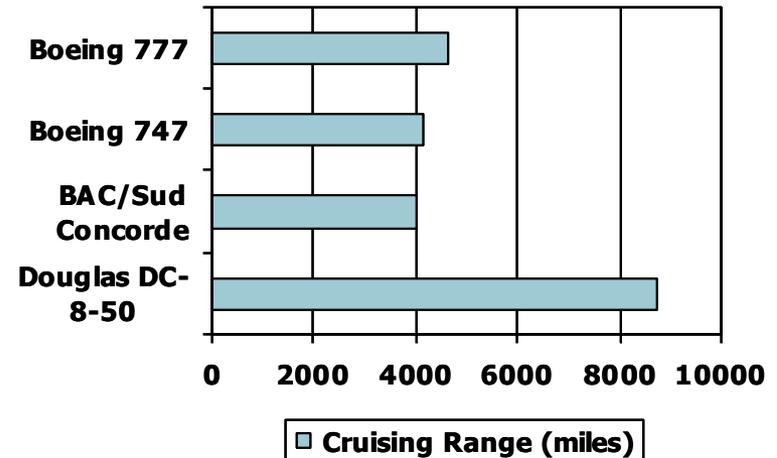
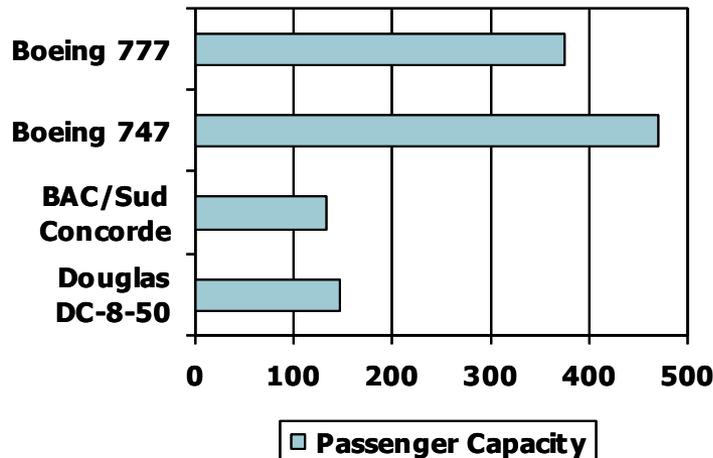
Determina velocidade em que instruções são executadas

- **E/S (incluindo Sistema Operacional)**

Determina velocidade em que operações de E/S são executadas

Definindo Desempenho

Qual avião tem melhor desempenho?



Diferentes Métricas

- **Tempo de Execução ou Resposta**

Tempo que leva para completar uma tarefa

Importante para o usuário comum de um sistema

- **Throughput**

Trabalho total feito por unidade de tempo

- Exemplos: tarefas ou transações por segundo ou hora

Importante para análise de máquinas servidoras

Frequentemente afetado pelo tempo de execução

Analisaremos fatores que afetam tempo de execução

Desempenho Relativo

- Desempenho = $1/\text{Tempo de Execução}$
- “X é n vezes mais rápido que Y”

$$\text{Desempenho}_x / \text{Desempenho}_y = \text{Tempo de Execução}_y / \text{Tempo de Execução}_x = n$$

- **Exemplo: tempo para rodar um programa**
10s em A, 15s em B
 $\text{Tempo de Execução}_B / \text{Tempo de Execução}_A$
 $= 15s / 10s = 1.5$
Então A é 1.5 vezes mais rápido que B

Medindo Tempo de Execução

■ Tempo Gasto (Elapsed time)

Tempo total de resposta, incluindo todos os aspectos

- Processamento, E/S, overhead de sistema operacional, tempo esperando algum evento

Determina desempenho do sistema

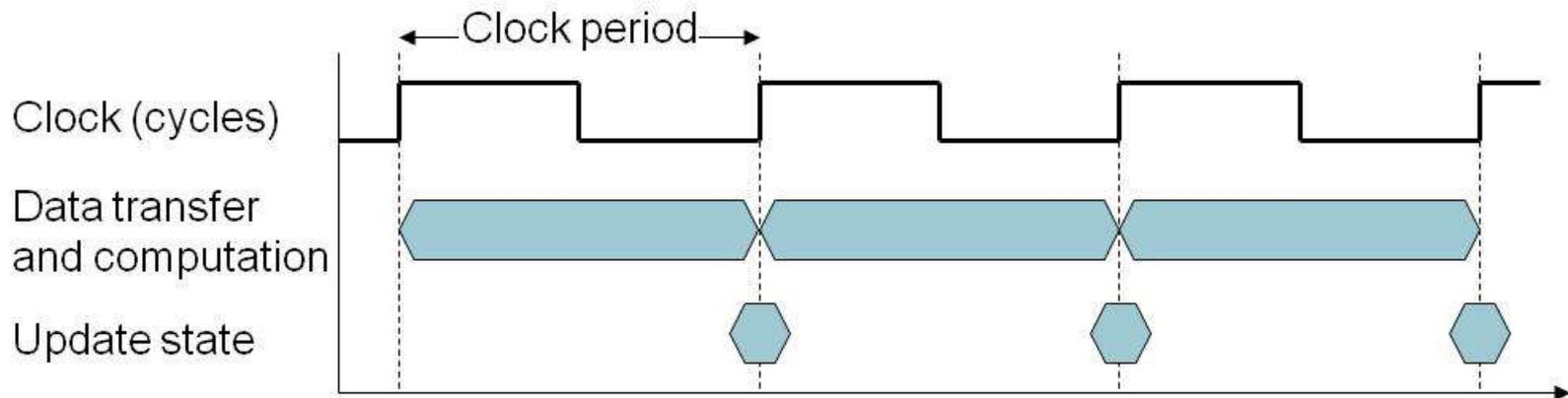
■ Tempo de CPU (CPU Time)

Tempo de processamento de um programa

- Não leva em conta tempo de E/S, execução de outros programas, etc

Clock

- Operações de uma CPU são governadas pelo clock



- Período do clock: duração de um ciclo de clock
Ex: $250\text{ps} = 0.25\text{ns} = 250 \times 10^{-12}\text{s}$
- Frequência do Clock : ciclos por segundo
Ex: $4.0\text{GHz} = 4000\text{MHz} = 4.0 \times 10^9\text{Hz}$

Tempo de CPU

$$\begin{aligned}\text{Tempo de CPU} &= \text{Ciclos de Clock (utilizados pela CPU)} \times \\ &\quad \text{Período de Clock} \\ &= \frac{\text{Ciclos de Clock}}{\text{Frequência}}\end{aligned}$$

- Desempenho pode ser melhorado
 - Reduzindo número de ciclos de clock
 - Aumentando frequência do clock
- Projetista de Hardware deve frequentemente fazer o *trade off* entre frequência e número de ciclos

Tempo de CPU: Exemplo

- Computador A

Frequência de clock: 2GHz, Tempo de CPU: 10s

- Projetando Computador B

Objetivo: Tempo de CPU = 6s

Maior frequência do clock possível, mas isto causa um aumento no número de ciclos em 1,2x

- **Qual deve ser então a frequência do clock para atingir o tempo de CPU necessário?**

$$\text{Frequência}_B = \frac{\text{Ciclos de Clock}_B}{\text{Tempo de CPU}_B} = \frac{1,2 \times \text{Ciclos de Clock}_A}{6s}$$

$$\begin{aligned} \text{Ciclos de Clock}_A &= \text{Tempo de CPU}_A \times \text{Frequência}_A \\ &= 10s \times 2\text{GHz} = 20 \times 10^9 \end{aligned}$$

$$\text{Frequência}_B = \frac{1,2 \times 20 \times 10^9}{6s} = \frac{24 \times 10^9}{6s} = 4\text{GHz}$$

Número de Instruções e CPI

Ciclos de Clock = N° de Instruções x Ciclos por Instrução (CPI)

Tempo de CPU = N° de Instruções x CPI x Período de Clock

$$= \frac{\text{N° de Instruções} \times \text{CPI}}{\text{Frequência}}$$

- **N° de Instruções por um programa**
Determinado pelo algoritmo, ISA e compilador
- **Quantidade média de ciclos por instrução**
Determinado pelo hardware da CPU
Se diferentes instruções têm diferentes CPIs
 - Depende de como as diferentes instruções foram usadas no programa

CPI : Exemplo

- Computador A: Tempo do ciclo = 250ps, CPI = 2,0
- Computador B: Tempo do ciclo = 500ps, CPI = 1,2
- Mesma ISA
- Qual é o **mais** rápido e **quão** mais rápido?

$$\begin{aligned}\text{Tempo de CPU}_A &= N^{\circ} \text{ de Instruções} \times \text{CPI}_A \times \text{Tempo do Ciclo}_A \\ &= N \times 2,0 \times 250\text{ps} = N \times 500\text{ps}\end{aligned}$$

A é mais rápido

$$\begin{aligned}\text{Tempo de CPU}_B &= N^{\circ} \text{ de Instruções} \times \text{CPI}_B \times \text{Tempo do Ciclo}_B \\ &= N \times 1,2 \times 500\text{ps} = N \times 600\text{ps}\end{aligned}$$

$$\frac{\text{Tempo de CPU}_B}{\text{Tempo de CPU}_A} = \frac{N \times 600\text{ps}}{N \times 500\text{ps}} = 1,2$$

1,2x mais rápido

Mais Detalhes sobre CPI

- Se diferentes classes de instruções levam diferentes números de ciclos

$$\text{Ciclos de Clock} = \sum_{i=1}^n (\text{CPI}_i \times \text{N}^\circ \text{ de instruções}_i)$$

- Média ponderada da CPI

$$\text{CPI} = \frac{\text{Ciclos de Clock}}{\text{N}^\circ \text{ de Instruções}} = \sum_{i=1}^n \left(\text{CPI}_i \times \frac{\text{N}^\circ \text{ de Instruções}_i}{\text{N}^\circ \text{ de Instruções}} \right)$$

Mais um Exemplo de CPI

- Uma mesma sequência de código foi compilado de 2 formas diferentes usando instruções das classes A, B, C

Classe	A	B	C
CPI da classe	1	2	3
Nº de Instruções na sequência 1	2	1	2
Nº de Instruções na sequência 2	4	1	1

- **Nº de Instruções na sequência 1 = 5**

$$\begin{aligned} &\text{Ciclos de Clock} \\ &= 2 \times 1 + 1 \times 2 + 2 \times 3 \\ &= 10 \end{aligned}$$

$$\text{Média CPI} = 10/5 = 2,0$$

- **Nº de Instruções na sequência 2 = 6**

$$\begin{aligned} &\text{Ciclos de Clock} \\ &= 4 \times 1 + 1 \times 2 + 1 \times 3 \\ &= 9 \end{aligned}$$

$$\text{Média CPI} = 9/6 = 1,5$$

Desempenho: Resumindo

$$\text{Tempo de CPU} = \frac{\text{Instruções}}{\text{Programa}} \times \frac{\text{Ciclos de Clock}}{\text{Instrução}} \times \frac{\text{Segundos}}{\text{Ciclo de Clock}}$$

- **Desempenho depende de:**

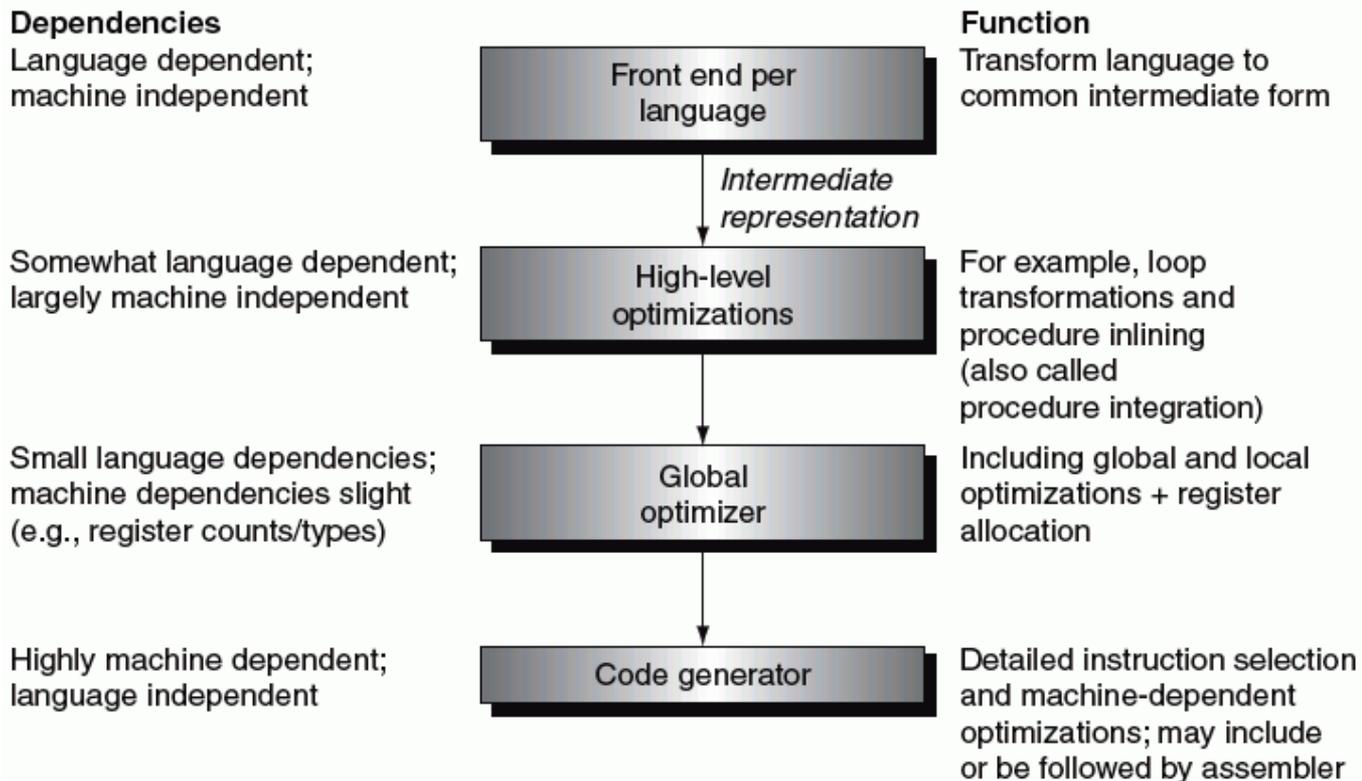
Algoritmo: afeta nº de instruções, possivelmente CPI

Linguagem: afeta nº de instruções, CPI

Compilador: afeta nº de instruções, CPI

ISA: afeta nº de instruções, CPI, período do clock

Otimizando Desempenho Através do Compilador



■ Diferentes tipos de otimização:

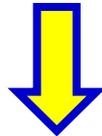
Alto nível, local, global

Exemplo de Otimização de Alto Nível

■ Loop Unrolling

Consiste em replicar corpo do laço para reduzir número de iterações, evitando testes de fim de laço e jumps

```
for( i = 0; i < 10; i++){  
    n = n + 5;  
}
```



```
for( i = 0; i < 5; i++){  
    n = n + 5;  
    n = n + 5;  
}
```

Outro Exemplo de Otimização de Alto Nível

■ Procedure Inlining

Consiste em substituir chamadas a procedimento pelo corpo do procedimento, para reduzir overhead da chamada

```
void imprimeSituacao(float m) {
    if (m >= 7)
        printf("\nAprovado");
    else
        printf("\nReprovado");
}
int main() {
    float media;
    printf("Digite media: ");
    scanf("%f", &media);
    imprimeSituacao(media);
}
```



```
int main() {
    float media;
    printf("Digite media: ");
    scanf("%f", &media);
    if (media >= 7)
        printf("\nAprovado");
    else
        printf("\nReprovado");
}
```

Otimização Local/Global

- Otimizações locais trabalham em cima de blocos simples individualmente

Ex: branches, expressões, loops, etc

- Otimizações globais trabalham sobre múltiplos blocos simples espalhados pelo código

- Exemplos de técnicas:

Strength Reduction

- Substitui operações complexas por operações mais simples
- Ex: Substitui instruções mult em assembly por sll quando for possível

Dead code elimination

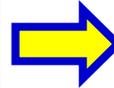
- Elimina código que não afeta resultado final do programa
- Exs: variáveis não utilizadas, condições inalcançáveis de ifs

Exemplo de Otimização Local/Global

■ Code Motion

Consiste em identificar trechos de código dentro de um loop que calculam sempre o mesmo valor em todas as iterações (invariante do loop) e colocar estes trechos para fora do loop

```
int y;  
int n = 0, x = 5;  
for( i = 0; i < 10; i++) {  
    y = x + 10;  
    n = n + 5;  
}
```



```
int y;  
int n = 0, x = 5;  
y = x + 10;  
for( i = 0; i < 10; i++) {  
    n = n + 5;  
}
```

Examinando Como Diferentes Fatores Podem Influenciar Desempenho

- **Exemplo: Ordenação**

Algoritmos : Bubble sort e Quicksort

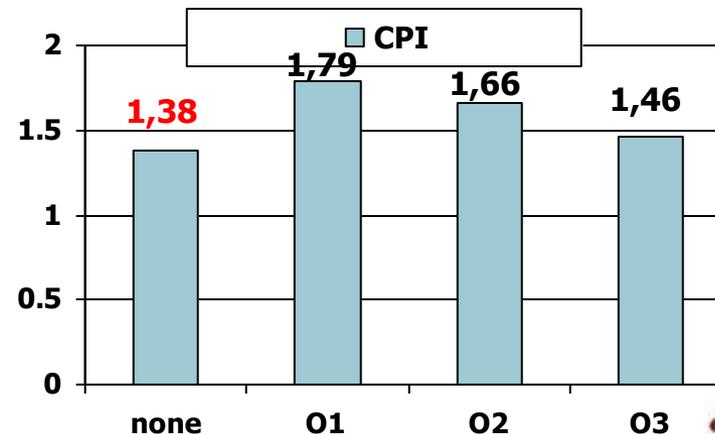
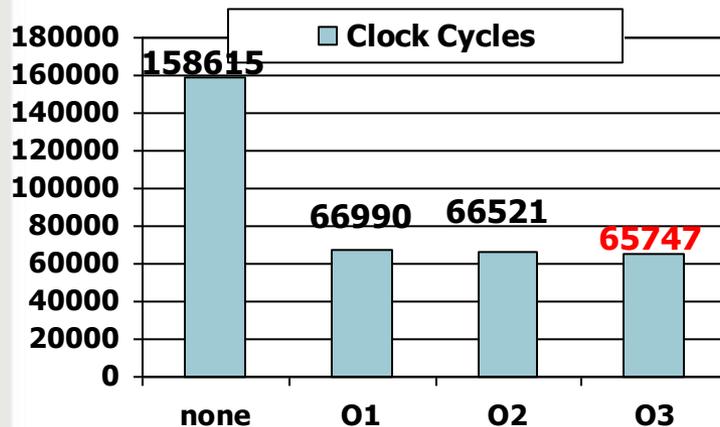
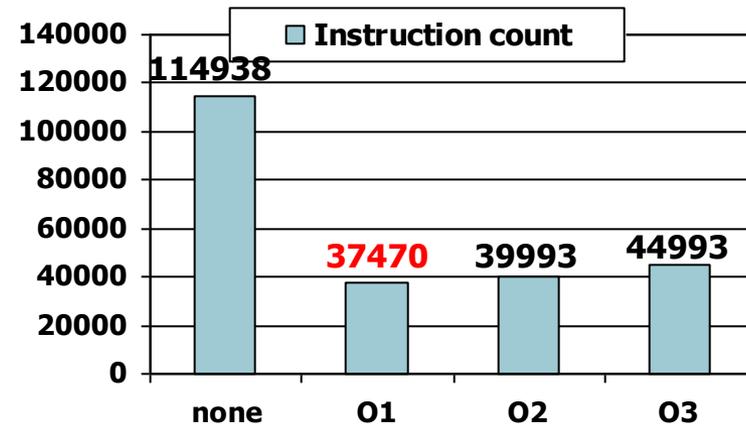
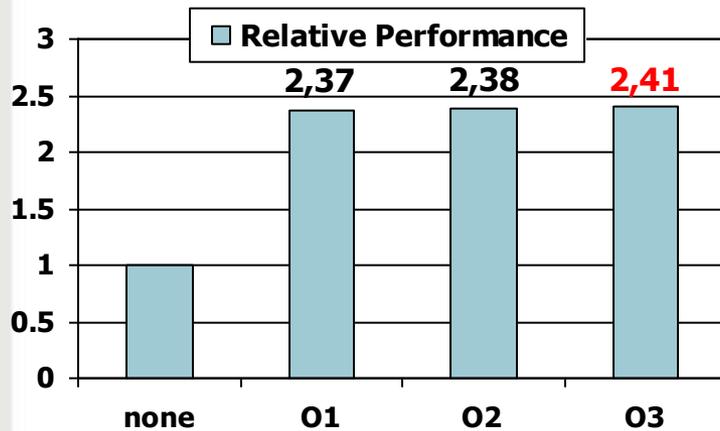
Linguagem: C e Java

Compilador: gcc com 3 níveis de otimização para C (O1, O2 e O3) e interpretador e JIT para Java

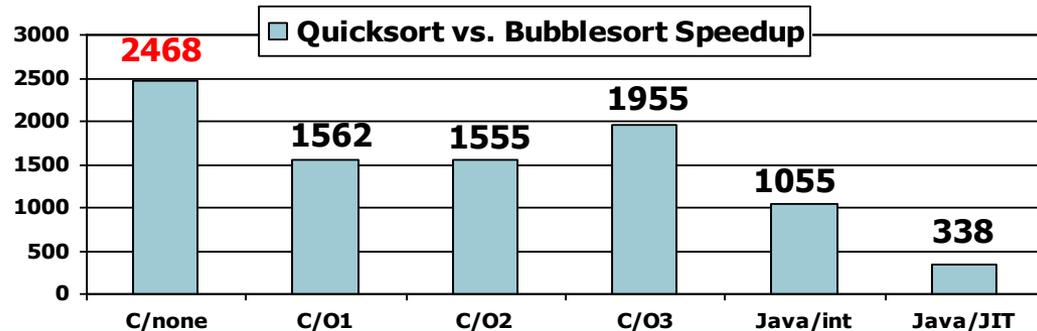
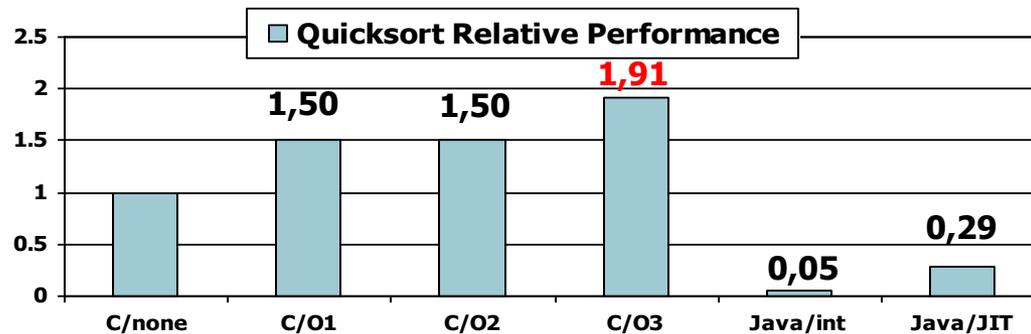
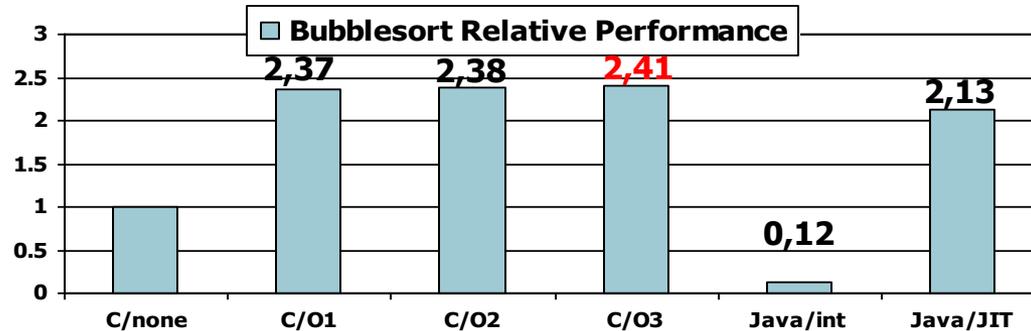
ISA: Pentium 4 (com Linux)

- **Comparação será feita em relação a códigos C não otimizados**

Efeito de Otimização do Compilador gcc



Efeito da Linguagem e Algoritmo



Analisando Resultados

- Número de instruções e CPI isoladamente não são bons indicadores de desempenho
- Otimizações de compilador são sensíveis ao algoritmo
- Otimizações nem sempre levam a um melhor desempenho
 - Pode aumentar tamanho do código, o que pode afetar desempenho, pois código pode ficar grande demais para caber em cache **(veremos depois)**
- Código compilado Java/JIT é significamente mais rápido que código interpretado pelo JVM
 - Comparável a C otimizado em alguns casos
- **Nenhuma otimização é capaz de compensar um algoritmo ruim!**

Como Podem Ser Comparados Diferentes Computadores?

■ Benchmarks

Conjunto de aplicações padrões que são utilizados para avaliar o desempenho

Vários domínios:

- Científicos
- Banco de dados
- Processamento de sinais
- Rede

■ SPEC (System Performance Evaluation Cooperative)

Iniciativa criada pela indústria para criar benchmarks para avaliar seus computadores

Benchmarks da SPEC

- Benchmarks usando inteiros
Escritos em C e C++
- Benchmarks usando ponto flutuante
Escritos em C e Fortran

Integer benchmarks		FP benchmarks	
Name	Description	Name	Type
gzip	Compression	wupwise	Quantum chromodynamics
vpr	FPGA circuit placement and routing	swim	Shallow water model
gcc	The Gnu C compiler	mgrid	Multigrid solver in 3-D potential field
mcf	Combinatorial optimization	applu	Parabolic/elliptic partial differential equation
crafty	Chess program	mesa	Three-dimensional graphics library
parser	Word processing program	galgel	Computational fluid dynamics
eon	Computer visualization	art	Image recognition using neural networks
perlbmk	perl application	equake	Seismic wave propagation simulation
gap	Group theory, interpreter	facerec	Image recognition of faces
vortex	Object-oriented database	amp	Computational chemistry
bzip2	Compression	lucas	Primality testing
twolf	Place and rote simulator	fma3d	Crash simulation using finite-element method
		sixtrack	High-energy nuclear physics accelerator design
		apsi	Meteorology: pollutant distribution