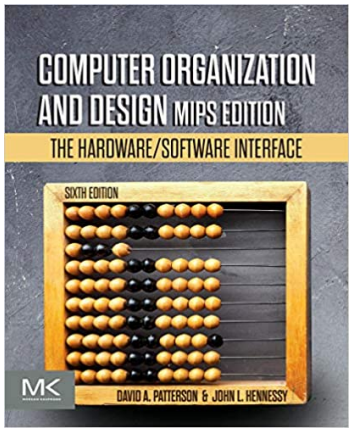
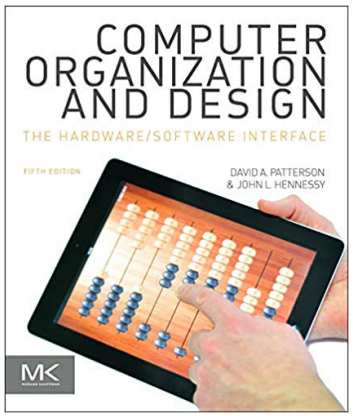
Aula 12 –Desempenho

Prof. Dr. Clodoaldo A. de Moraes Lima

Material baseado no livro "Patterson, David A., Hennessy, J. L. - Computer Organization And Design: The Hardware/Software Interface"





Sumário

- Como arquiteturas são geralmente avaliadas
- Como arquiteturas obedecem a restrições de projeto
- Métricas de desempenho
- Combinando desempenho e resultados
- Lei de Amdahl

Avaliação de uma arquitetura

Permite pontuar e avaliar de maneira qualitativa e quantitativa diferentes arquiteturas.

É essencial. Usada desde o projetista até o vendedor de computadores.

Como fazer uma avaliação precisa?

Definições

Airplane	Passenger capacity	Cruising range (miles)	Cruising speed (m.p.h.)	Passenger throughput (passengers × m.p.h.)
Boeing 777	375	4630	610	228,750
Boeing 747	470	4150	610	286,700
BAC/Sud Concorde	132	4000	1350	178,200
Douglas DC-8-50	146	8720	544	79,424

Qual é o melhor?

Qual o de melhor desempenho?

Definições

Airplane	Passenger capacity	Cruising range (miles)	Cruising speed (m.p.h.)	Passenger throughput (passengers × m.p.h.)
Boeing 777	375	4630	610	228,750
Boeing 747	470	4150	610	286,700
BAC/Sud Concorde	132	4000	1350	178,200
Douglas DC-8-50	146	8720	544	79,424

Alcance?

Velocidade?

Capacide de passageiros?

Definições

Desktop

Computador mais rápido é aquele que termina o mais rápido possível o processamento e te devolve a resposta.

Servidor

O mais rápido é aquele termina uma maior quantidade de tarefas em um único dia.

Definições

Usuário individual

Interesse no tempo de resposta para cada tarefa (tempo de execução).

Usuário de um datacenter

Interesse na vazão de processamento - i.e.: Quantos trabalhos foram concluídos em determinado período de tempo.

Definições

Como melhorar o tempo de resposta e vazão?

Trocar o processador por um mais rápido? ou

Acréscimo de mais processadores ao sistema?

Definições

Troca do processador

Diminuir o tempo de resposta sempre aumenta a vazão. Então, ambos são melhorados.

Acréscimo de mais processadores

- Somente a vazão é aumentada.
- Entretanto, diminuição no tempo de resposta pode ocorrer caso hajam processos aguardando por processamento.

Avaliando desempenho de arquiteturas

- Um workload é uma coleção de programas
- Um workload de usuário são os programas que o usuário executa no dia a dia
- Idealmente, os usuários deveriam avaliar o desempenho das arquiteturas com seus workloads antes de comprá-las
  - Ninguém, ou quase, trabalha desta maneira
- Como os usuários podem então tomar decisões de compra baseadas em desempenho?

Benchmarks

- Benchmarks são programas especificamente escolhidos para medir desempenho
- Benchmark tentam imitar workloads de comunidades de usuários em particular
  - Benchmarks científicos, benchmarks comerciais, benchmarks multimídia, etc.
- Os fabricantes geralmente informam o desempenho de suas máquinas com base na execução de Benchmarks
  - A maioria dos workloads podem ser representados por benchmarks com um conjunto relativamente pequeno de programas

O benchmarks SPEC

SPEC = System Performance Evaluation Cooperative

- Criado em 1989 por fabricantes de computadores para avaliar de forma "isenta" seus projetos.
- SPECint um conjunto de programa baseados em operações com inteiros apenas
- SPECfp um conjunto de programas baseados em operações com ponto flutuante
- Resultados divulgados por empresas que usam SPEC
  - Resultados individuais dos benchmarks
  - Resultados composto de SPECint
  - Resultados compostos de SPECfp
  - Vazão (Throughput) obtido com a execução simultánea de múltiplas instâncias do mesmo benchmark
  - SPEC também inclui java, web, e outros benchmarks
  - www.spec.org

Riscos do uso de Benchmarks

Como muitas decisões de compra são feitas com base em benchmark, os projetistas são pressionados a otimizarem seus projetos para o benchmark

Isso é realmente bom?

Novas arquiteturas

O problema do ovo e da galinha:

Projetistas precisam comparar diferentes opções de projeto antes de tê-los implementado para executar os benchmarks

Solução: Prototipagem de hardware

Projeta, gera protótipo, avalia, Projeta, gera protótipo, avalia, ...

Solução muito cara e demorada devido ao custo em tempo e recursos para projetar e gerar protótipo

Novas arquiteturas

- Simulações podem ser uma alternativa adequada
  - Exploração do espaço de projeto
- Para avaliar o desempenho de diferentes projetos bastaria
  - O número de ciclos de relógio necessário para executar cada benchmark
  - A frequência de relógio de cada opção de projeto
  - Tempo de execução= Número de ciclos/frequência
- Não tão fácil quanto parece

Novas Arquiteturas

- Um modelo descrito em linguagem de alto nível que represente adequadamente a arquitetura deve ser usado
  - Muitos níveis de abstração podem ser escolhidos
  - Cada nível de abstração oculta ou explicita detalhes do modelo real
- O modelo executa código de máquina e coleta métricas de desempenho
  - Dissipação de potência também pode ser avaliada
  - Corretude funcional e não funcional não é importante neste momento, só o suficiente para confiar no modelo
- Vários parâmetros da arquitetura podem e devem ser alterados para avaliar as opções de projeto

Métricas de desempenho

- Suponha a execução de um programa em 2 estações de trabalho diferentes
  - A estação mais rápida é aquela que termina a execução primeiro
- Suponha a execução de programas em 2 estações de trabalho compartilhadas
  - A estação mais rápida é aquela que completa a execução de mais programas durante um mesmo intervalo de tempo
- Usuário tradicional: reduzir o tempo de resposta
  - Tempo entre o início e o fim de uma tarefa, também chamado tempo de execução
- Usuário tradicional: aumentar a vazão(throughput)
  - Quantidade de trabalho realizado em um dado intervalo de tempo

Métricas de desempenho

TEMPO, TEMPO, TEMPO!!!!

- Tempo de resposta (latência)
  - Quanto tempo leva para meu trabalho ser realizado?
  - Quanto tempo leva para realizar um trabalho específico?
  - Quanto tempo preciso esperar para finalizar minha simulação?
- Vazão (throughput)
  - Quantos trabalhos a máquina pode realizar em um intervalo de tempo?
  - Qual é a velocidade média de execução ?
  - Quanto trabalho está sendo feito?
- Tempo de reposta vs Vazão
  - Se atualizarmos uma máquina com um novo processador, em que melhoramos?
  - Se acrescentarmos uma máquina ao laboratório, em que melhoramos?

Tempo de execução

Tempo decorrido (real time)

- Conta tudo (acessos a disco e a memória, E/S etc.)
- Um número útil, mas normalmente não é ideal para fins de comparação

Tempo de CPU (usertime + system time)

- Não conta E/S ou tempo gasto executando outros programas pode ser dividido em tempo de sistema e tempo de usuário

Nosso foco: tempo de CPU do usuário (usertime)

- Tempo gasto executando as linhas de código que estão em nosso programa desconsiderando chamadas de sistema e tratamento por parte do SO

Métricas de desempenho

Problema

Diferente métricas podem ser utilizadas: usuários vs projetistas

Solução

Correlacionar as métricas como forma de inferir quais impactos as mudanças de hardware tem sobre o usuário.

Métricas de desempenho

Para melhorar o desempenho, o tempo de resposta deve diminuir

Dado um computador X, tem-se que:

$$Performance_X = \frac{1}{Executiontime_x}$$

Métricas de desempenho

Na comparação entre dois computadores (X e Y):

$$Performance_X > Performance_Y$$

$$\frac{1}{Executiontime_X} > \frac{1}{Executiontime_Y}$$

$$Executiontime_Y > Executiontime_X$$

$$\frac{Performance_X}{Performance_Y} = \frac{Executiontime_Y}{Executiontime_X} = n$$

Exemplo

Exercício 1

Se um computador A demora 27.5s para executar uma tarefa e o computador B demora 32.1 segundos para a mesma, quantas vezes A é mais rápido que B?

A = 27.5 s
B = 32.1 s

$$\frac{Performance_A}{Performance_B} = \frac{ExecutionTime_B}{ExecutionTime_A} = \frac{32.1}{27.5} = 1.167$$

Definições

- Como visto, o tempo é a medida de desempenho utilizada.
- Quem efetua a mesma tarefa em um menor espaço de tempo é o mais rápido

- Entretanto, conforme já mencionado, o tempo de cálculo de uma tarefa não deve considerar: Acesso à disco, memória, I/0, etc.

Métricas de desempenho

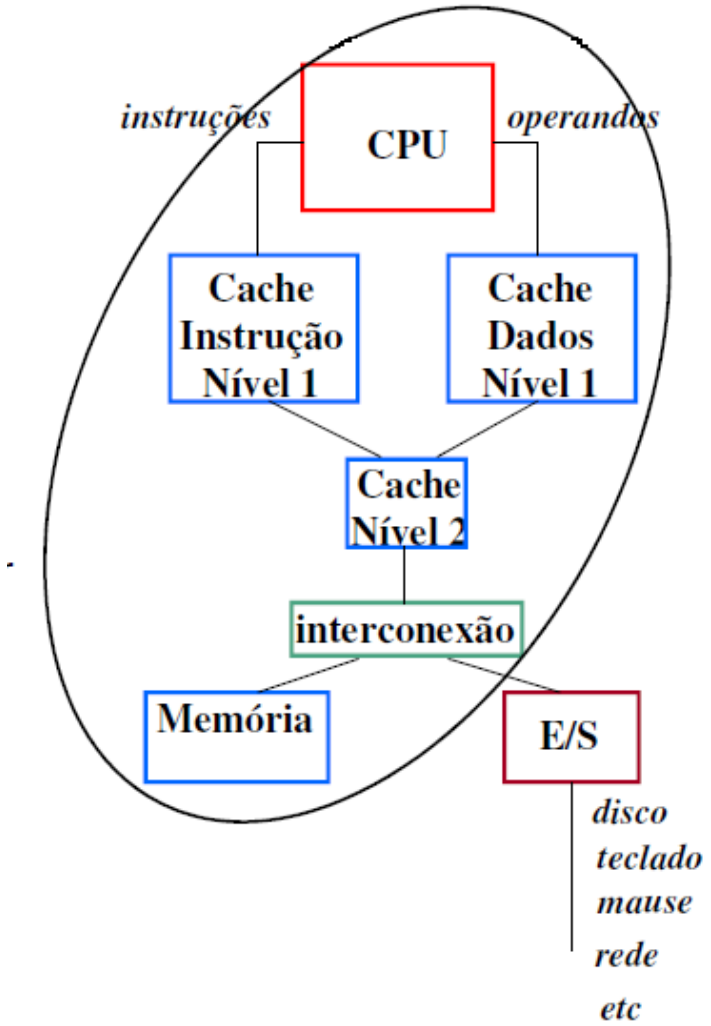
- Medida precisa de tempo requer a identificação dos momentos em que a tarefa está realmente sendo executada no processador (CPU time).
- Diferentes aplicações possuem diferentes características comportamentais:
  - CPU bound
  - I/O bound

Métricas de desempenho

- Consideraremos somente o CPU time.
- Entretanto, "na prática, a teoria é outra" :)
- Sistema operacional introduz abstrações para representação das tarefas
- CPU time pode ainda ser subdivido em tarefas menores, dificultando ainda mais sua aferição.

Avaliando só o necessário

- Decompondo a execução em componentes permite focar em pontos importantes da arquitetura
  - Operações de E/S são freqüentemente sobrepostas por outras tarefas na CPU
  - O subsistema de E/S pode ser desenvolvido de forma quase independente do resto
- Tempo de CPU ignora o tempo de execução do componente de E/S



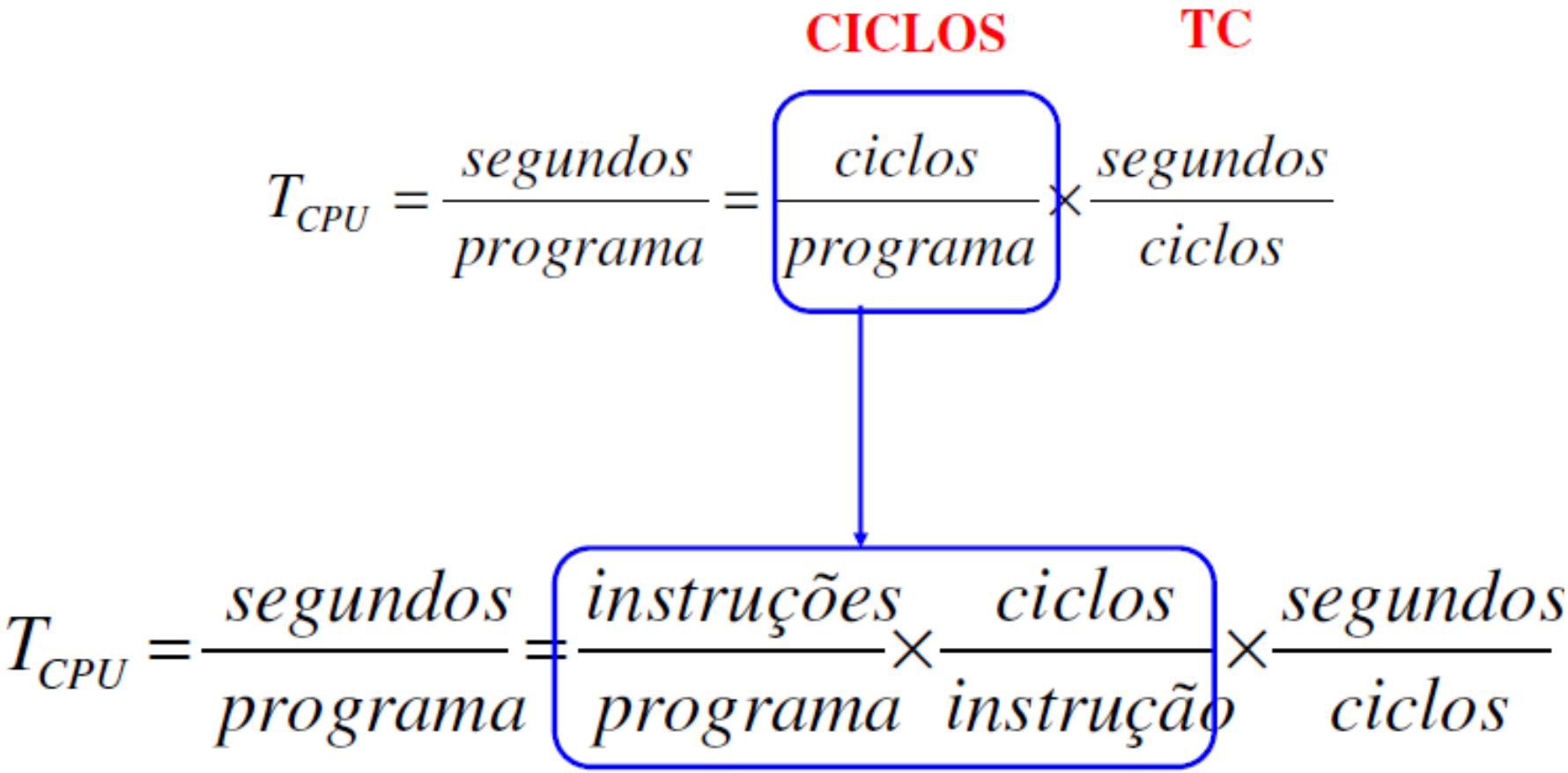
Tempo de CPU como métrica

- É o tempo gasto pela CPU, cache (eventualmente mais de uma) e a memória para a execução de uma tarefa
- Ignora tempo requerido por operações de E/S
- Composto de:
  - Tempo de CPU do usuário: O tempo gasto pelo programa do usuário
  - Tempo de CPU do sistema: O tempo gasto pelo sistema operacional
- Tempo de CPU do usuário é normalmente avaliado ...
  - Vários benchmarks padrão, como SPEC com baixa atividade do SO
  - Vários simuladores de arquitetura não suportam SO
    - Muitas vezes a programação requer inclusão de bibliotecas
  - O código do SO freqüentemente não está disponível para avaliação

Tempo de CPU

- Tempo de CPU = CICLOS x TC = INST x CPI x TC
- CICLOS
  - Número total de ciclos para executar um programa
- TC
  - Tempo do ciclo do relógio (Período do relógio)
  - 1/frequência do relógio
- INST
  - Número total de instruções assembly executadas
- CPI
  - Número médio de cíclos do relógio por instrução executada
  - Total de ciclos do relógio / Total de instruções executadas (CICLOS/INST)
  - Diferentes tipos de instruções (add, divide, etc.) podem ter diferentes números de ciclos de relógio para executar

Desempenho


CICLOS
TC
$T_{CPU} = \frac{segundos}{programa} = \frac{ciclos}{programa} \times \frac{segundos}{ciclos}$
$T_{CPU} = \frac{segundos}{programa} = \frac{instruções}{programa} \times \frac{ciclos}{instrução} \times \frac{segundos}{ciclos}$

Desempenho - Terminologia

$$T_{CPU} = \frac{segundos}{programa} = \frac{instruções}{programa} \times \frac{ciclos}{instrução} \times \frac{segundos}{ciclos}$$

Nº de instruções por programa

Ciclos de clock por instrução (CPI)

Período de clock

$$T_{exec} = INST \times CPI \times TC$$

Exemplo de tempo de CPU

lw $4, 0($2)	**2 cycles**
lw $6, 4($2)	**2 cycles**
add $4, $4, $6	**1 cycle**
sw $4, 0($2)	**1 cycle**

- CICLOS = 6
- INST = 4
- CPI = 6/4 = 1.5
- TC = 1 ns
- CPU Time = CICLOS x TC = 6 x 1ns = 6ns

Exemplo de tempo de CPU

Considere duas implementações do mesmo conjunto de instruções (Instruções Set Architecture - ISA)

Para um mesmo programa

- Máquina A: TC = 0.25 ns e CPI = 2.0
- Máquina B: TC = 0.5 ns e CPI = 1.2

- Qual a máquina é mais rápida para estes programas?
- Quanto mais rápida?

Exemplo de tempo de CPU

Ambas máquinas executam o mesmo número de instruções (INST), pois executam o mesmo programa.

Primeiro Passo

Encontrar o tempo de execução do programa para cada máquina:

$$T_{CPU\_A} = INST \times CPI_A \times TC \quad \Rightarrow \quad T_{CPU\_A} = INST \times 2{,}0 \times 250ps$$

$$T_{CPU\_B} = INST \times CPI_B \times TC \quad \Rightarrow \quad T_{CPU\_B} = INST \times 1{,}2 \times 500ps$$

Exemplo de tempo de CPU

Segundo Passo

Encontrar a razão entre os desempenhos das máquinas

$$\frac{Desempenho_A}{Desempenho_B} = \frac{T_{CPU\_B}}{T_{CPU\_A}} = \frac{INST \times 1{,}2 \times 500}{INST \times 2{,}0 \times 250} = 1{,}2$$

Portanto, a máquina A é 1.2 vez mais rápida que B

Qual(is) parte(s) do tempo de CPU (INST, CPI, TC)

São influenciadas pelo projeto da ISA?

São influenciadas pelo desenvolvedor do compilador?

São influenciadas pelo projetista da microarquitetura?

O que é importante quando...

- Os programas já estão compilados e você é o projetista da microarquitetura?
- A ISA e microarquitetura estão definidas e você é o desenvolver dor do compilador?
- Você está comparando duas máquinas que têm diferentes ISAs?

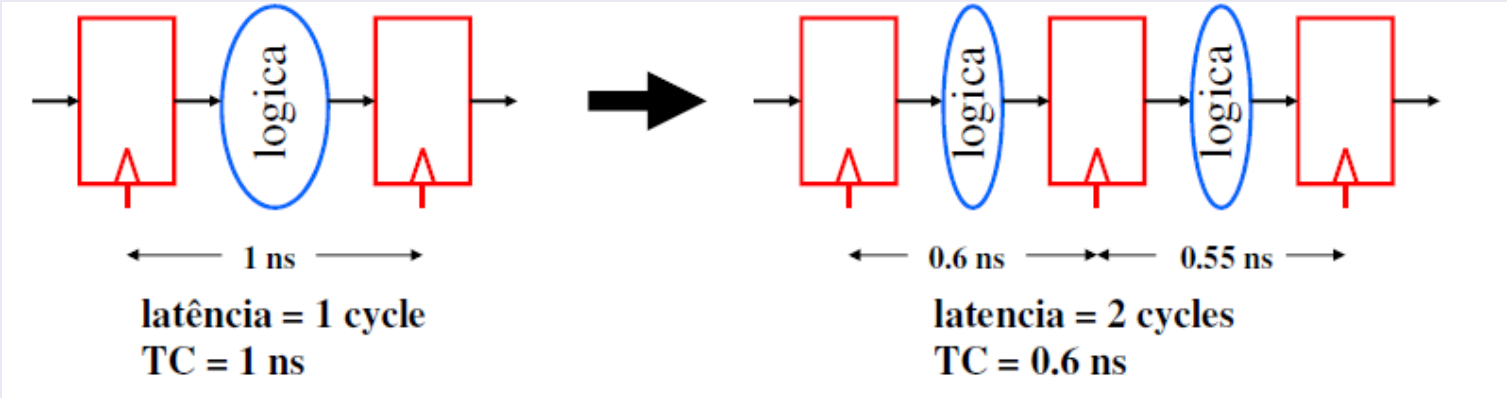
Resumo Geral

Componente de hardware ou software	Afeta o quê?	Como?
Algoritmo	Contagem de instruções, possivelmente o CPI	O algoritmo determina o número de instruções do programa fonte executadas e, portanto, o número de instruções do processador executadas. O algoritmo também pode afetar o CPI, favorecendo instruções mais lentas ou mais rápidas. Por exemplo, se o algoritmo usar mais operações de ponto flutuante, ele tenderá a ter um CPI mais alto.
Linguagem de programação	Contagem de instruções, CPI	A linguagem de programação certamente afeta a contagem de instruções, já que as instruções na linguagem são traduzidas em instruções do processador, que determinam a contagem de instruções. A linguagem também pode afetar o CPI devido aos seus recursos; por exemplo, uma linguagem com pesado suporte para abstração de dados (como Java) exigirá chamadas indiretas, que usarão instruções de CPI mais altos.
Compilador	Contagem de instruções, CPI	A eficiência do compilador afeta a contagem de instruções e a média de ciclos por instrução, já que o compilador determina a tradução das instruções da linguagem fonte para instruções do computador. O papel do compilador pode ser bastante complexo e afeta o CPI de maneiras complexas.
Conjunto de instruções	Contagem de instruções, velocidade de clock, CPI	O conjunto de instruções afeta os três aspectos do desempenho da CPU, uma vez que ela afeta as instruções necessárias para uma função, o custo em ciclos de cada instrução e a velocidade de clock geral do processador.

Latência

- Latência = número de ciclos de relógio requeridos para fazer algo
  - Acesso a cache, execução de uma instrução, etc.
  - Outra definição: Quantidade de tempo (ns) para fazer algo

- Projetistas podem aumentar a latência com vistas a diminuir o TC



  - Por que a opção de maior latência tem melhor desempenho?

Compromisso CICLOS - TC

- Melhorias que afetam positivamente CICLOS ou TC frequentemente afetam negativamente o outro (CICLOS ou TC)
- Exemplos
  - Aumento da cache para reduzir o CICLOS de leitura aumenta o TC de leitura
    - CICLOS diminui porque a memória principal é acessada menos frequentemente
    - Caches maiores operam mais lentamente, aumentando o TC
  - Aumento do paralelismo reduz CICLOS e aumenta TC
    - Paralelismo resulta em execução simultânea de instruções, o que reduz o CICLOS,
    - mas torna o controle mais complexo, aumentando o TC

Compromisso INST - CPI

- Quando um programa em linguagem de alto nível é traduzido para assembly o desenvolvedor do compilador tem várias opções que afetam INST e CPI
  - A melhor solução pode envolver mais instruções simples
- Exemplo: multiplicação por 5

Métricas de desempenho

Um desenvolvedor está decidindo sobre duas sequências de códigos a serem utilizados. O mesmo possui as seguintes informações:

	CPI for each instruction class		
	A	B	C
CPI	1	2	3

Está considerando duas sequências de código que possuem o seguinte número de instruções:

Code sequence	Instruction counts for each instruction class		
	A	B	C
1	2	1	2
2	4	1	1

Métricas de desempenho

- a) Qual das sequências executa o maior número de instruções?
- b) Qual será a mais rápida?
- c) Qual o CPI para cada sequência?

Resposta

- a) Qual das sequências executa o maior número de instruções?
  Sequência 1: $2 + 1 + 2 = 5$ instruções.
  Sequência 2: $4 + 1 + 1 = 6$ instruções
- b) Qual será a mais rápida?
  Calculando o número de ciclos de clock CPU para cada sequência:

$$\text{CPU clock cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{C}_i)$$

$$\text{CPU clock cycles}_1 = (2 \times 1) + (1 \times 2) + (2 \times 3) = 2 + 2 + 6 = 10 \text{ cycles}$$

$$\text{CPU clock cycles}_2 = (4 \times 1) + (1 \times 2) + (1 \times 3) = 4 + 2 + 3 = 9 \text{ cycles}$$

Resposta

- c) Qual o CPI para cada sequência?

$$\text{CPI} = \frac{\text{CPU clock cycles}}{\text{Instruction count}}$$

$$\text{CPI}_1 = \frac{\text{CPU clock cycles}_1}{\text{Instruction count}_1} = \frac{10}{5} = 2.0$$

$$\text{CPI}_2 = \frac{\text{CPU clock cycles}_2}{\text{Instruction count}_2} = \frac{9}{6} = 1.5$$

Resumindo os resultados de desempenho

- Útil para gerar uma médida de desempenho para múltiplos resultados de benchmark
- Para tempo de execução (e métricas relacionadas)
  - O tempo total de execução de n programas
  - Tambem chamado de Média Aritmética (MA)

$$MA = \frac{1}{n} \sum_{i=1}^{n} Tempo_i$$

onde $Tempo_i$ é o tempo de execução do i-ésimo programa

Resumindo os resultados de desempenho

A MA ponderada define peso para cada programa

$$AM_p = \frac{1}{n} \sum_{i=1}^{n} Peso_i * Tempo_i$$

- onde $Peso_i$ é peso associado ao i-ésimo programa
- A soma de todos os pesos resulta em 1

Resumindo os resultados de desempenho

- Para medidas de desempenho dadas em taxa (razão), por exemplo, instruções/sec
  - É comum usar a Média Harmônica (HM)

$$HM = \frac{n}{\sum_{i=1}^{n} Taxa_i}$$

  - Onde $Taxa_i$ é a taxa do i-ésimo programa
  - n número de programas
  - Média Harmônica Ponderada (HMp) pode ser definida da mesma forma que MAp

Métricas de desempenho

MIPS - Milhões de instruções por segundo

- Usar somente o clock é uma métrica ruim de comparação.
- Outra métrica ruim é o MIPS: Milhões de instruções por segundo.
- Fácil de compreender: Computador mais rápido possui MIPS maior.

MIPS como medida de desempenho

- Considere o computador com três classes de instrução e medições de CPI.

	CPI para esta classe de instrução		
	A	B	C
CPI	1	2	3

- Suponha que medindo o código gerado por dois computadores diferentes para o mesmo programa, obtivemos os seguintes dados

Código do	Contagens de instruções (em bilhões) para cada classe de instrução		
	A	B	C
Compilador 1	5	1	1
Compilador 2	10	1	1

- Considere que a velocidade de clock do computador seja 4GHz. Que sequência de código será executada mais rápido de acordo com o MIPS? E de acordo com o tempo de execução?

MIPS como medida de desempenho

- Primeiro, encontramos o tempo de execução para dois compiladores diferentes usando a seguinte equação:

$$\text{Tempo de execução} = \frac{\text{Ciclos de clock da CPU}}{\text{Velocidade de clock}}$$

Podemos usar a fórmula anterior para os ciclos de clock da CPU

$$\text{Ciclos de clock da CPU} = \sum_{i=1}^{n} (CPI_i \times C_i)$$

$$\text{Ciclos de clock da CPU}_1 = (5 \times 1 + 1 \times 2 + 1 \times 3) \times 10^9 = 10 \times 10^9$$

$$\text{Ciclos de clock da CPU}_2 = (10 \times 1 + 1 \times 2 + 1 \times 3) \times 10^9 = 15 \times 10^9$$

MIPS como medida de desempenho

- Calculando o tempo de execução para os dois compiladores

$$\text{Tempo de execução}_1 = \frac{10 \times 10^9}{4 \times 10^9} = 2.5 \text{ segundos}$$

$$\text{Tempo de execução}_2 = \frac{15 \times 10^9}{4 \times 10^9} = 3.75 \text{ segundos}$$

Portanto, concluímos que o compilador 1 gera o programa mais rápido, de acordo com o tempos de execução

MIPS como medida de desempenho

- Vamos calcular o índice de MIPS para versão do programa, usando

$$MIPS = \frac{\text{Contragem de Instruções}}{\text{Tempo de execução} \times 10^6}$$

$$MIPS_1 = \frac{(5+1+1) \times 10^9}{2.5 \times 10^6} = 2800$$

$$MIPS_2 = \frac{(10+1+1) \times 10^9}{3.75 \times 10^6} = 3200$$

- Assim, o código do compilador 2 possui um índice de MIPS mais alto, mas o código do compilador 1 é executado mais rápido
- MIPS pode falhar em fornecer um quadro verdadeiro do desempenho

MIPS: Problemas

- Não leva em conta a capacidade das instruções. Dois computadores diferentes não podem ser comparados pelo MIPS.
- MIPS pode variar com programas diferentes dentro de um mesmo computador. Um programa pode usar instruções que tem maior ou menor CPI.

	Clock rate	CPI Class A	CPI Class B	CPI Class C	CPI Class D
P1	1.5 GHz	1	2	3	4
P2	2 GHz	2	2	2	2

Métricas de desempenho

- Componentes da medida de desempenho:

Componentes de desempenho	Unidades de Medida
Tempo de execução na CPU	Segundos para o programa.
Contagem de Instruções	Instruções executadas no programa.
CPI (Ciclos por Instrução)	Número médio de ciclos de *clock* por instrução.
Tempo de ciclo de *clock*	Segundos por ciclo de *clock*

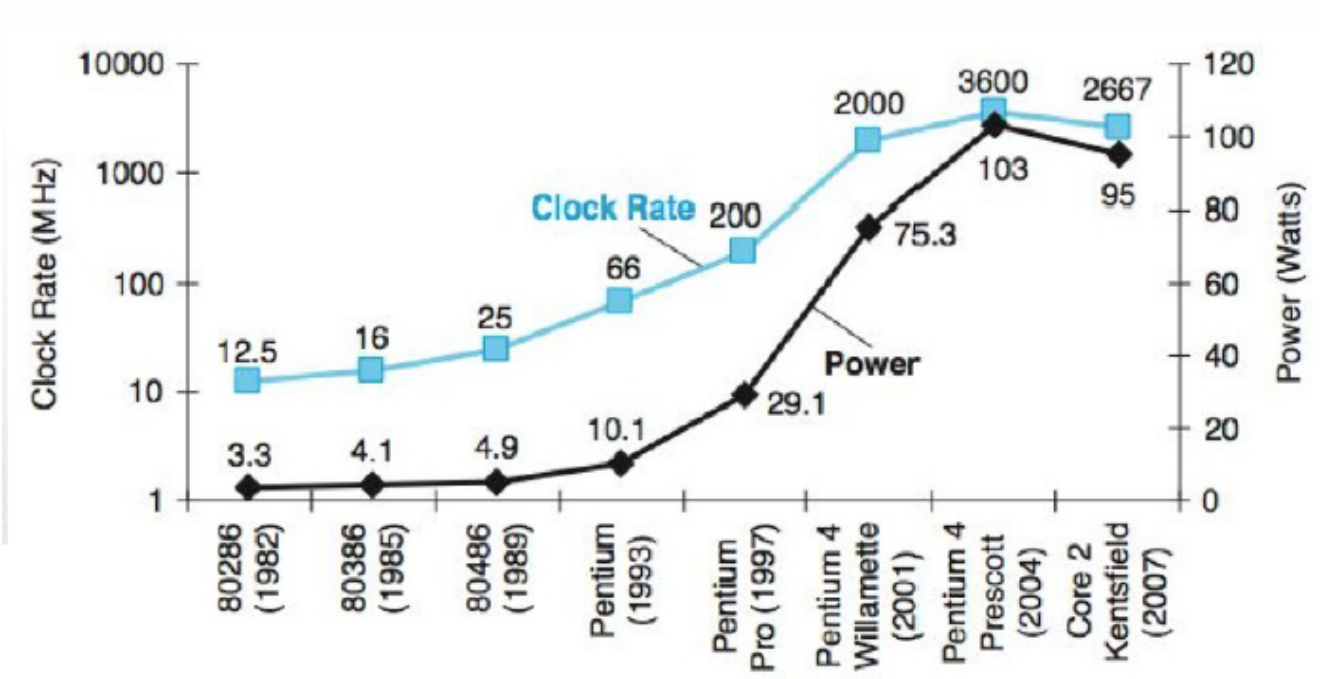
Entendendo o desempenho de um programa

- Dependente do algoritmo, linguagem, compilador, arquitetura e hardware utilizados

Componente	Área afetada	Como?
Algoritmo	Número instruções utilizadas, possivelmente CPI	Algoritmo é quem diz o número de instruções a serem executadas. O algoritmo também pode afetar o CPI através do favorecimento a instruções mais rápidas ou lentas.
Linguagem de Programação	Número instruções utilizadas, CPI	Os trechos na linguagem de programação são traduzidos para instruções do processador, o qual determina o número de instruções a serem executadas. Pela mesma razão, o CPI pode ser afetado
Compilador	Número instruções utilizadas, CPI	A eficiência do compilador afeta ambos o número de instruções a serem utilizadas e o ciclo médio por instrução uma vez que o compilador é quem traduz da linguagem de prog. para instruções da CPU.
Arquitetura de Instruções	Número instruções utilizadas, taxa de *clock*, CPI	Afeta todos os outro três fatores, uma vez que essa afeta as instruções necessárias por uma função.

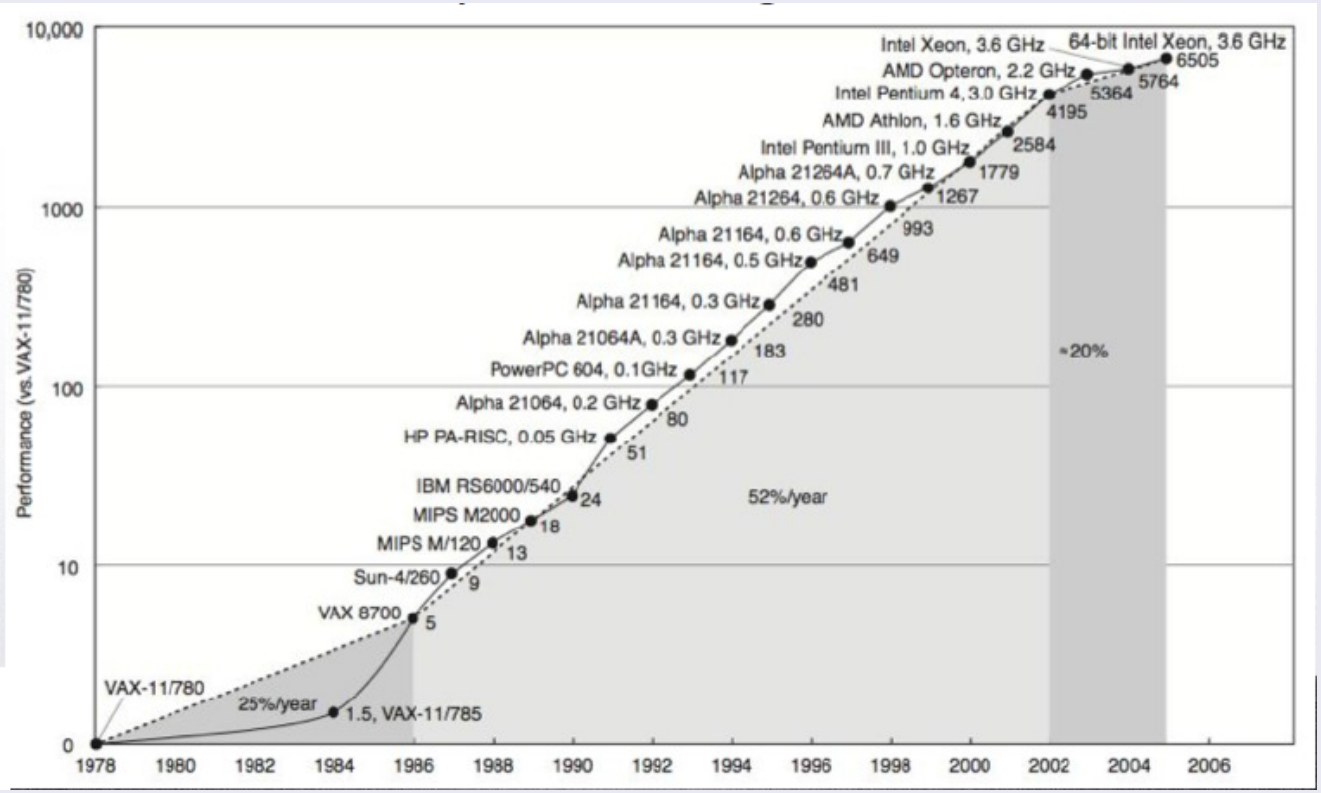
Evolução do desempenho ao longo dos anos

- O consumo energético e a taxa de clock aumentaram durante as últimas décadas. Recentemente, notou-se um recuo.



Métricas de desempenho

Evolução do desempenho ao longo dos anos



Evolução do desempenho ao longo dos anos

Multiprocessadores

- Múltiplos cores: Avanços na vazão em detrimento do tempo de resposta.
- No passado, programadores contavam com a Lei de Moore:
  - A velocidade de seus programas teoricamente dobraria a cada 18 meses.
  - Devido a barreira do aumento da potência, o tempo de resposta não é mais o referencial.

Evolução do desempenho ao longo dos anos

Multiprocessadores

- O paralelismo sempre foi um ponto crítico de desempenho na computação.
- Dois níveis de paralelismo:
  - Hardware: Técnicas aplicadas ao hardware para melhorar a vazão de instruções executadas.
  - Software: Técnicas aplicadas ao software para melhorar o desempenho de programas.

Evolução do desempenho ao longo dos anos

Multiprocessadores

- Desenvolvedores são forçados a considerar o paralelismo em hardware quando criando softwares.
- Por que é tão difícil desenvolver softwares que fazem uso de paralelismo:
  - a) Programação paralela é em essência a programação focada em desempenho, naturalmente mais difícil.
  - b) O software não pode somente estar correto e resolver o problema para o qual foi desenvolvido. Ele precisa ser rápido.

Evolução do desempenho ao longo dos anos

Multiprocessadores

- Por que é tão difícil desenvolver softwares que fazem uso de paralelismo:
  - c) Rapidez para o processador significa que o desenvolvedor precisa modularizar melhor seu código. De forma a aproveitar melhor os recursos do hardware.
    - Ex.: Analogia da escrita de um artigo para um jornal:
    - Escalonamento;
    - Balanceamento de carga;
    - Overhead de sincronização e comunicação

Lei de Amdahl

- Aumento de desempenho devido a uma melhora é limitado pela fração do tempo que a melhora é usada.

$$ET_{new} = ET_{old} * \left[(1 - fraction_{enhanced}) + \frac{fraction_{enhanced}}{speedup_{enhanced}} \right]$$

- onde
  - ET_{old} é o tempo de execução sem a melhora
  - $fraction_{enhanced}$ é a fração do tempo que é beneficiado pela melhora
  - $speedup_{enhanced}$ é a aceleração obtida usando a melhora

Exemplo da lei de Amdahl

- Assuma que operações de multiplicação constituem 20% do tempo de execução de um benchmark
- Qual é a redução no tempo de execução provocada por um novo multiplicador em hardware que é 10 vezes mais rápido que o anterior?

$$ET_{new} = ET_{old} * \left[(1 - 0.2) + \frac{0.2}{10}\right]$$

$$speedup = \frac{ET_{old}}{ET_{new}} = 1.22$$

Exercício Lei de Amdahl

- Suponha que uma máquina foi melhorada fazendo com que todas as instruções de ponto flutuante executassem 5 vezes mais rápido.
- Se o tempo de execução de um certo benchmark antes da melhoria era de 10 segundos.
- Qual será o ganho (speedup) se metade deste tempo é gasto executando instruções de ponto flutuante?

Exercícios Lei de Amdahl

- Suponha que devemos escolher um benchmark que demonstre a melhoria na nova unidade de ponto flutuante descrita anteriormente.
- Deseja-se mostrar um ganho geral da ordem de 3
- Considere um benchmark que roda em 100 segundos com o antigo hardware de ponto flutuante.
- Qual deve ser a proporção de operações de ponto flutuante nos programas deste benchmark de forma a se alcançar o ganho desejado?

Exercício sobre Desempenho

- Um programa roda em 10 segundos em um computador A, que tem um clock de 400 MHz.
- Está sendo projetada uma nova máquina B, que deverá rodar este programa em 6 segundos.
- No entanto, essa redução afetará o projeto do resto da CPU. Por essa razão, a máquina B requer 1,2 vezes mais ciclos de clock que a máquina A para um mesmo programa.
- Qual a freqüência de clock da nova máquina?

Exercício sobre Desempenho

- Um projetista de compilador deve decidir entre duas seqüências de código assembly para uma máquina particular.
- Existem três diferentes classes de instruções (A, B, C) que compõem cada uma das seqüências de código assembly.
- Cada Classe de instruções apresenta diferentes números de ciclos de relógio por instrução (CPI).

Classe	CPI
A	1
B	2
C	3

Exercício sobre Desempenho

Seqüência de código	Nº de instruções por classe		
	A	B	C
1	2	1	2
2	4	1	1

- Qual a seqüência de código que executa mais instruções?
- Qual a seqüência mais rápida? Quanto?
- Qual o CPI de cada seqüência?

Lembre-se que...

- Desempenho possui especificidades de um programa para outro
  - Tempo total de execução é um resumo consistente de desempenho
- Para uma das arquitetura, o aumento de desempenho vem de:
  - Aumento na taxa de clock
  - Melhorias na organização do processador de modo a diminuir a CPI
  - Melhorias no compilador que reduzem o CPI e/ou número de instruções
  - Escolhas da linguagem/algoritmo que afetam o número de instruções
- Erro comum: esperar que a melhoria de um aspecto (uma medida) do desempenho possa afetar o desempenho final de todo o sistema