# Avaliação de Desempenho: *Benchmarks*

Arquitetura de Computadores

Lic. em Engenharia Informática

João Luis Sobral

# Avaliação do Desempenho

| | |
|---|---|
| **Conteúdos** | 1.1 – “benchmarks” |
| **Resultados de Aprendizagem** | R1.2 – Avaliar diferentes tipos de benchmarks relativamente à qualidade e tipo de informação produzida |
| | R1.3 – Selecionar as métricas e testes mais adequados à caracterização do desempenho em diferentes situações |

# Avaliação de desempenho

- **Desempenho o que é?**
  - Métrica que nos permite comparar sistemas de computação, permitindo:
    - selecionar o *melhor* sistema
    - optimizar software
    - desenhar sistemas de computação/processadores

- **Existem várias métricas consoante o sistema que se pretende avaliar:**
  - Tempo de execução (tempo de resposta)
  - Débito (quantidade de trabalho(s) realizado por unidade de tempo)
  - Consumo

- **Comparação de desempenho com base no tempo de execução de programas**
  - Uma máquina X é *n vezes* mais rápida que outra Y se:

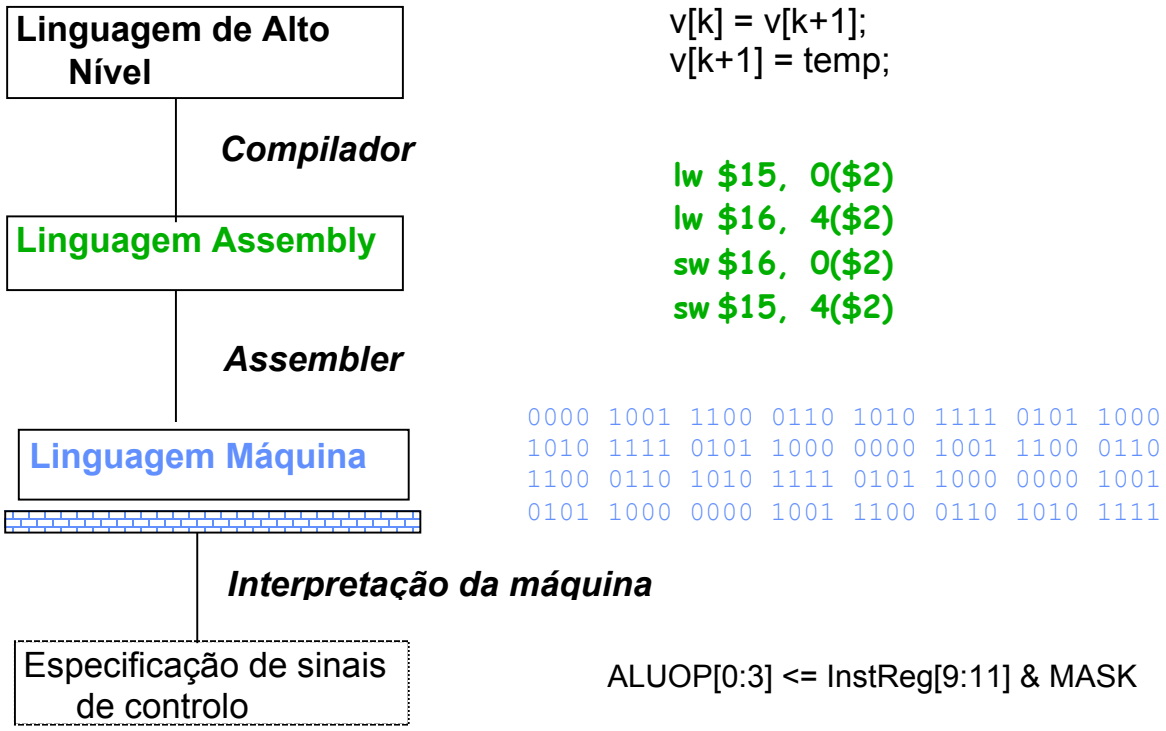
$$n = \frac{\text{Tempo de execução em Y}}{\text{Tempo de execução em X}}$$

# Avaliação de desempenho

- **Tipos de sistemas de computação**
  - Computadores de secretária
    - Mercado de maior dimensão em termos monetários
    - Abrange sistemas de €1000 e €10.000
    - A tendência é para uma optimização da relação custo/desempenho
    - *Bechmarks* para este tipo de mercado: SPEC, Winstone, etc.
  - Servidores
    - Centrado em arquiteturas para fornecimento de serviços em larga escala, com elevada disponibilidade (ex. servidores web)
    - A escalabilidade é um factor importante
    - Os servidores são projetados para maximizarem o débito (ex. número de páginas web servidas por minuto)
    - As *bechmarks* são orientadas para a medição do número de transações por unidade de tempo: SPECfs, SPECWeb, TPC-H, TPC-W.
  - Computadores embebidos
    - Computadores incluídos em periféricos, cuja presença do computador não é evidente
    - Mercado de maior crescimento (ex. Telemóveis)
    - Grande variedade de poder de processamento e de preço
    - O preço e o consumo energético são factores chave
    - Frequentemente estão sujeitos a requisitos de tempo real (ex. compressão de voz)

# Avaliação de desempenho

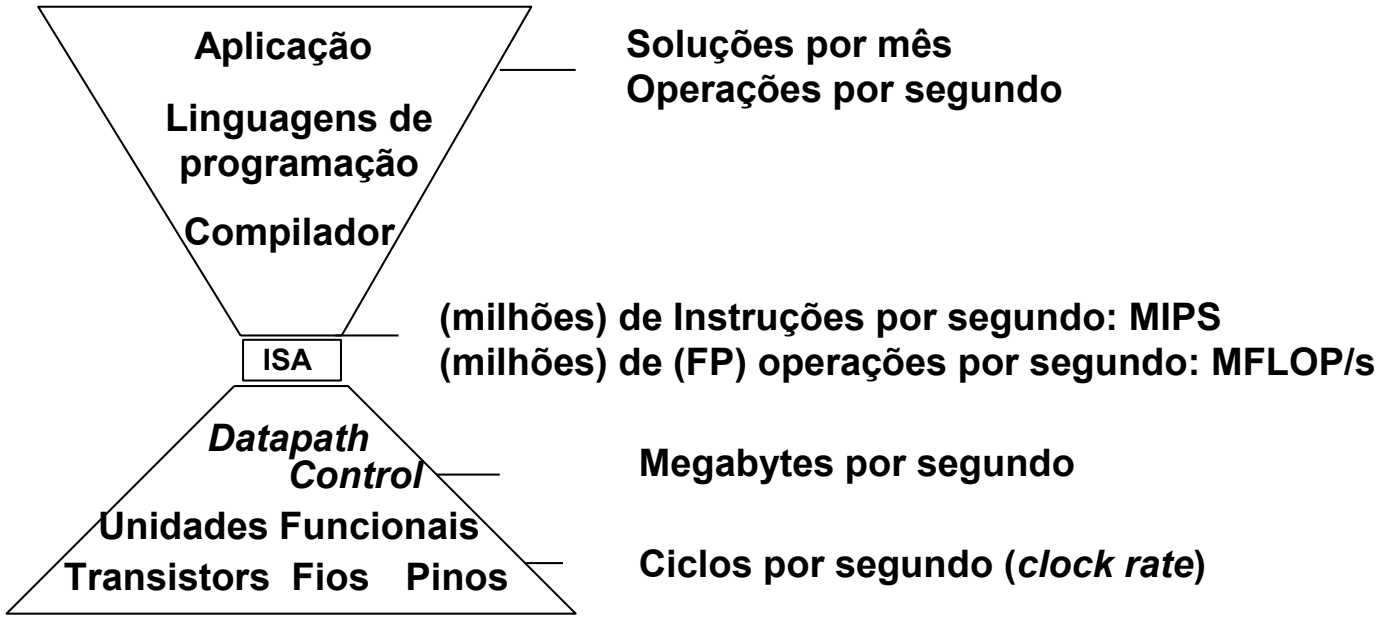
- **Níveis de abstração**



# Avaliação de desempenho

- **Tipos de métricas de desempenho**

$$\text{Desempenho} = \frac{1}{\text{Tempo de execução}}$$



# Benchmark

- Entende-se por *benchmark* um teste, ou conjunto de testes, que, quando executados num determinado sistema de computação, dão alguma medida do **desempenho** de um(s) determinado(s) **componente(s) do sistema** na execução de uma determinada **tarefa**

**PROBLEMA:** O componente do sistema medido por um determinado teste é importante para a aplicação em causa?

**PROBLEMA:** Como garantir que a carga (*workload,* tarefa) a que a máquina é sujeita durante a medição é equivalente à carga (*workload,* tarefa) a que o utilizador normalmente a sujeita?

**PROBLEMA:** Como garantir que os fabricantes e/ou vendedores não manipulam os testes (ou condições de realização dos mesmos) no sentido de melhorarem os resultados?

# Benchmarks Sintéticos

Pequenos programas desenvolvidos especificamente para medir alguma característica específica da máquina. Normalmente não realizam nenhuma tarefa específica.

**Desvantagens**

1. Não reflectem a carga que um utilizador aplica à sua máquina;
2. Programas pequenos que utilizam apenas a *cache*;
3. Alguns compiladores geram código optimizado para estes testes. Estas optimizações não podem depois ser usadas em aplicações reais.

**Vantagens**

1. Na fase inicial de desenho de um sistema estes testes são muito úteis, pois são fáceis de compilar e mesmo de simular.

**Exemplos:** Dhrystone (int) e Whetstone (FP)

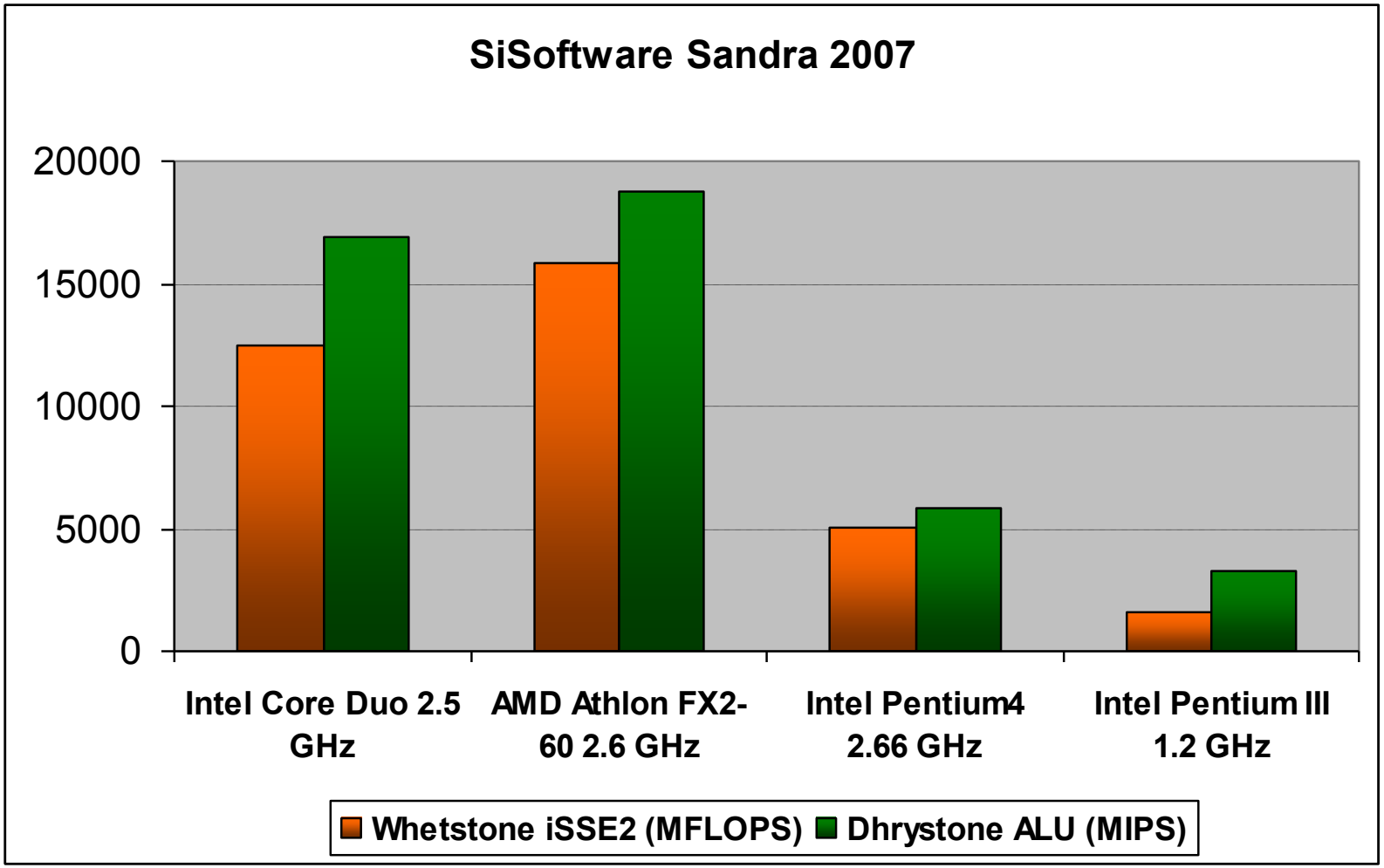
# Dhrystone (inteiros)

- Teste sintético que contém uma mistura representativa de operações inteiras:
  - invocação de procedimentos
  - utilização de apontadores, inteiros e caracteres
  - atribuições e cálculo de expressões
- Desenvolvida em 1984 por Reinhold Weicker em Ada.
  Melhorada e convertida para C em 1989 por Weicker e Richardson
- Resultados em *Dhrystone loops per second*
  Resultados em **MIPS** (Milhões de instruções por segundo) não podem ser usados para comparar diferentes arquitecturas (ex. CISC vs. RISC)
- Muito divulgada, mas muitos compiladores incluem optimizações específicas para este teste

# Whetstone (vírgula flutuante)

- Teste sintético que contém uma mistura representativa de operações em vírgula flutuante:
  - `abs, sqrt, exp, alog, sin, cos, atan, …`
- Desenvolvida em Algol60, em 1972
- Resultados em MWIPS (Milhões de Whetstone instruções por segundo) ou em MFLOPS (Milhões de operações em vírgula flutuante por segundo)
- Muito divulgada, mas muitos compiladores incluem optimizações específicas para este teste

# Resultados


SiSoftware Sandra 2007
20000
15000
10000
5000
0
Intel Core Duo 2.5 GHz
AMD Athlon FX2-60 2.6 GHz
Intel Pentium4 2.66 GHz
Intel Pentium III 1.2 GHz
Whetstone iSSE2 (MFLOPS)
Dhrystone ALU (MIPS)

# Conjuntos de Aplicações Reais

Conjuntos de aplicações seleccionadas por representarem cargas típicas para um grande número de utilizadores

**Desvantagens**

1. Difícil seleccionar conjuntos de aplicações que garantidamente representem uma grande maioria de utilizadores;
2. Estes testes levam muito tempo a executar e as condições de teste e relatório de resultados são geralmente muito exigentes;

**Vantagens**

1. O utilizador pode geralmente confiar que os resultados reflectem com algum grau de precisão o desempenho a esperar da sua máquina;
2. Tratando-se de aplicações reais torna-se difícil aos fabricantes introduzirem características especiais no *hardware* ou nos compiladores para inflaccionar os resultados.

**Exemplo:** SPEC Benchmarks

# SPEC- Standard Performance Evaluation Corporation

A SPEC (http://www.spec.org/) é uma associação criada em 1989 por um grupo de companhias para normalizar:

- o conjunto de testes a que as máquinas devem ser submetidas;
- as condições em que estes testes devem ser realizados;
- a forma como os resultados devem ser documentados

| Benchmark | Descrição |
|---|---|
| SPEC MPI'2007 | High Performance Computing (MPI) |
| SPEC jvm'2008 | Java Virtual Machine |
| SPEC sfs'2008 | System File Server |
| SPEC viewperf'2011 | Gáficos e workstations |
| SPEC virt_sc'2010 | Servidores em ambiente virtualizado |
| SPEC power_ssj'2008 | Consumo energético de Servidores |
| SPEC CPU'2006 | Processador – memória - compilador |

# SPEC CPU2006

Conjunto de programas cuidadosamente seleccionados para representarem a carga que um utilizador "regular" impõem à sua máquina.

Inclui vários testes dos quais se destacam:

| Benchmark | Descrição |
|---|---|
| SPECint_base2006 | Programas com operações em inteiros, compilados sem optimizações. |
| SPECint2006 | Programas com operações em inteiros, compilados com optimizações. |
| SPECfp_base2006 | Programas com operações em vírgula flutuante, compilados sem optimizações. |
| SPECfp2006 | Programas com operações em vírgula flutuante, compilados com optimizações. |

# Evolução da benchmark SPEC CPU

| SPEC2006 benchmark description | SPEC2006 | SPEC2000 | SPEC95 | SPEC92 | SPEC89 |
|---|---|---|---|---|---|
| GNU C compiler | | | | | gcc |
| Interpreted string processing | | | perl | | espresso |
| Combinatorial optimization | | mcf | | | li |
| Block-sorting compression | | bzip2 | | compress | eqntott |
| Go game (AI) | go | vortex | go | sc | |
| Video compression | h264avc | gzip | ijpeg | | |
| Games/path finding | astar | eon | m88ksim | | |
| Search gene sequence | hmmer | twolf | | | |
| Quantum computer simulation | libquantum | vortex | | | |
| Discrete event simulation library | omnetpp | vpr | | | |
| Chess game (AI) | sjeng | crafty | | | |
| XML parsing | xalancbmk | parser | | | |
| CFD/blast waves | bwaves | | | | fpppp |
| Numerical relativity | cactusADM | | | | tomcatv |
| Finite element code | calculix | | | | doduc |
| Differential equation solver framework | dealII | | | | nasa7 |
| Quantum chemistry | gamess | | | | spice |
| EM solver (freq/time domain) | GemsFDTD | | | swim | matrix300 |
| Scalable molecular dynamics (~NAMD) | gromacs | | apsi | hydro2d | |
| Lattice Boltzman method (fluid/air flow) | lbm | | mgrid | su2cor | |
| Large eddie simulation/turbulent CFD | LESlie3d | wupwise | applu | wave5 | |
| Lattice quantum chromodynamics | milc | apply | turb3d | | |
| Molecular dynamics | namd | galgel | | | |
| Image ray tracing | povray | mesa | | | |
| Spare linear algebra | soplex | art | | | |
| Speech recognition | sphinx3 | equake | | | |
| Quantum chemistry/object oriented | tonto | facerec | | | |
| Weather research and forecasting | wrf | ammp | | | |
| Magneto hydrodynamics (astrophysics) | zeusmp | lucas | | | |

# SPEC int

| Teste | HLL | Descrição |
|---|---|---|
| 445.gobmk | C | Artificial Intelligence |
| 464.h264ref | C | Vídeo Compression |
| 403.gcc | C | C Programming Language Compiler |
| 429.mcf | C | Combinatorial Optimization |
| 458.sjeng | C | Game Playing: Chess |
| 471.omnetpp | C++ | Discrete Event Simulation |
| 456.hmmer | C | Search Gene Sequence |
| 400.perlbench | C | PERL Programming Language |
| 462.libquantum | C | Physics: Quantum Computing |
| 473.astar | C++ | Path finding algorithms |
| 401.bzip2 | C | Compression |
| 483.xalancbmk | C++ | XML processing |

12 programas

# SPEC CPU 2006

Como calcular os índices SPECint2006 e SPECfp2006?

1. O tempo de execução de cada teste numa máquina de referência (SUN Ultra Enterprise 2) é dividido pelo tempo de execução na máquina a testar. Chama-se a esta razão o **SPEC ratio**.

2. É calculada a média geométrica dos vários SPEC ratios.

$$SPEC = \sqrt[n]{\prod_{i}^{n} SPECratio_i}$$

# SPEC CPU 2006