From: Vinicius Carlos <vcarlos@gmail.com>

Date: Tue, 20 Nov 2012 14:00:37
Subject: Uma boa noticia
To: santos@inf.ufsc.

Ola Luiz, tudo bem?

Eu to te escrevendo pra te contar uma novidade! Recebi hoje uma oferta de trabalho da ARM, em Cambridge, pra comecar em Setembro do ano que vem, depois que eu terminar o mestrado.

O cargo eh na divisao de processadores, Hardware Design and Verification Graduate Engineer! To muito contente. Foi basicamente pra esse tipo de trabalho que eu resolvi vir fazer o mestrado.

Era isso! Obrigado de novo pelo apoio e ajuda!

Abracos

Vinicius

From: Vinicius Carlos <vcarlos@gmail.com>

Date: Tue, 20 Nov 2012 14:00:37
Subject: Uma boa noticia
To: santos@inf.ufsc.

ARM Ltd.
Microprocessor's Division
110 Fulbourn Road
Cambridge
GB-CB1 9NJ Great Britain

Ola Luiz, tudo bem?

(Aluno desta disciplina em 2002.2 P1=10,0; P2=10,0; P3=8,0; NF = 9,5)

Eu to te escrevendo pra te contar uma novidade! Recebi hoje uma oferta de trabalho da ARM, em Cambridge, pra comecar em Setembro do ano que vem, depois que eu terminar o mestrado.

O cargo eh na divisao de processadores, Hardware Design and Verification Graduate Engineer! To muito contente. Foi basicamente pra esse tipo de trabalho que eu resolvi vir fazer o mestrado.

Era isso! Obrigado de novo pelo apoio e ajuda!

Abracos

Vinicius

Desempenho: benchmarks

Estudo de caso 1: SPEC CPU

- **Tempos de execução normalizados**
  - **Relativos a uma máquina de referência**

$$\text{Razão SPEC} = \frac{\text{TempoEx}_{\text{SUN UltraSparc@296MHz}}}{\text{TempoEx}_{\text{medido}}}$$

- **Para os 12 programas inteiros**
  - **CINT2006 = média geométrica das razões SPEC**
- **Para os 17 programas de ponto flutuante**
  - **CFP2006 = média geométrica das razões SPEC**

Exemplo: CINT2006 para Intel Core i7 920

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

Exemplo: CINT2006 para Intel Core i7 920

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

$f = 2.66\ GHz \Rightarrow T = 0.376\ 10^{-9} = 0.376\ ns$

Exemplo: CINT2006 para Intel Core i7 920

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

$$I \times CPI \times T = t_{ex}$$

Exemplo: CINT2006 para Intel Core i7 920

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

$t_{referência}/t_{ex}$ = SPEC ratio

Exemplo: CINT2006 para Intel Core i7 920

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

$$\text{CINT2006} = (r_1 \times r_2 \times r_3 \times \ldots \times r_{11} \times r_{12})^{1/12}$$

Exemplo: CINT2006 para Intel Core i7 920

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

Pior CPI **Melhor CPI**

Exemplo: CINT2006 para Intel Core i7 920

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

Pior/melhor = 6 vezes

Discussão

- **Dada ISA, como melhorar o desempenho ?**
  - **Maior frequência de relógio (f)**
    - » **Melhor implementação do sistema digital**
    - » **Melhor tecnologia de fabricação do circuito integrado**
  - **Menor número de instruções (I)**
    - » **Melhor escolha de algoritmo, linguagem e opções de otimização no compilador**
  - **Menor número de ciclos por instrução (CPI)**
    - » **Melhor organização do HW (*pipeline, cache, multicore*)**
    - » **Melhor escolha de opções de otimização no compilador**
      - **Seleção de instruções (*mix*)**
      - **Ordenamento de instruções (exploração de paralelismo)**

Discussão

- **Dada ISA, como melhorar o desempenho ?**
  - **Maior frequência de relógio (f)**
    - » **Melhor implementação do sistema digital**
    - » **Melhor tecnologia de fabricação do circuito integrado**
  - **Menor número de instruções (I)**
    - » **Melhor escolha de algoritmo, linguagem e opções de otimização no compilador**
  - **Menor número de ciclos por instrução (CPI)**
    - » **Melhor organização do HW (*pipeline, cache, multicore*)**
    - » **Melhor escolha de opções de otimização no compilador**
      - **Seleção de instruções (*mix*)**
      - **Ordenamento de instruções (exploração de paralelismo)**

Discussão

- **É um erro comum assumir que:**

  “A introdução de uma melhoria afetando apenas parte de uma máquina aumente o desempenho total proporcionalmente à melhoria.”

Discussão

- **Exemplo:**
  - **Um programa executa em 100ns.**
  - **As multiplicações contribuem com 80ns.**
  - **De quanto preciso acelerar a multiplicação para que o programa execute 5× mais rápido ?**

Discussão

- **Exemplo:**
  - **Um programa executa em 100ns.**
  - **As multiplicações contribuem com 80ns.**
  - **De quanto preciso acelerar a multiplicação para que o programa execute 5× mais rápido ?**

$$\text{TempoEx}_{\text{c/melhoria}} = \frac{\text{TempoEx}_{\text{afetado p/melhoria}}}{\text{proporção de melhoria}} + \text{TempoEx}_{\text{não afetado p/melhoria}}$$

Discussão

- **Exemplo:**
  - **Um programa executa em 100ns.**
  - **As multiplicações contribuem com 80ns.**
  - **De quanto preciso acelerar a multiplicação para que o programa execute 5× mais rápido ?**

$$\text{TempoEx}_{\text{c/melhoria}} = \frac{\text{TempoEx}_{\text{afetado p/melhoria}}}{\text{proporção de melhoria}} + \text{TempoEx}_{\text{não afetado p/melhoria}}$$

$$\frac{100}{5} = \frac{80}{n} + (100 - 80)$$

Discussão

- **Exemplo:**
  - **Um programa executa em 100ns.**
  - **As multiplicações contribuem com 80ns.**
  - **De quanto preciso acelerar a multiplicação para que o programa execute 5× mais rápido ?**

$$\text{TempoEx}_{\text{c/melhoria}} = \frac{\text{TempoEx}_{\text{afetado p/melhoria}}}{\text{proporção de melhoria}} + \text{TempoEx}_{\text{não afetado p/melhoria}}$$

$$\frac{100}{5} = \frac{80}{n} + (100 - 80) \Leftrightarrow 20 = \frac{80}{n} + 20$$

Discussão

- **Exemplo:**
  - **Um programa executa em 100ns.**
  - **As multiplicações contribuem com 80ns.**
  - **De quanto preciso acelerar a multiplicação para que o programa execute 5× mais rápido ?**

$$\text{TempoEx}_{\text{c/melhoria}} = \frac{\text{TempoEx}_{\text{afetado p/melhoria}}}{\text{proporção de melhoria}} + \text{TempoEx}_{\text{não afetado p/melhoria}}$$

$$\frac{100}{5} = \frac{80}{n} + (100 - 80) \Leftrightarrow 20 = \frac{80}{n} + 20 \quad \Leftrightarrow 0 = \frac{80}{n} \quad !?$$

Discussão

- **É um erro comum:**

“Utilizar apenas parte da equação de desempenho como métrica.”

Discussão

- **É um erro comum:**

  "Utilizar apenas parte da equação de desempenho como métrica."

- **Exemplo: MIPS para medir desempenho.**
  - **Milhões de instruções por segundo**

$$\text{MIPS} = \frac{I}{\text{TempoEx} \times 10^6}$$

  - **Noção intuitiva, mas:**
    - » **Não leva em conta diferenças no conj. de instruções**
    - » **Varia entre programas num mesmo computador**
    - » **Pode variar inversamente com o desempenho!?**

Discussão

- **Se f = 4 GHz, qual código é mais rápido ?**

classe	CPI
A	1
B	2
C	3

(em bilhões de instruções)

	A	B	C
Compilador 1	5	1	1
Compilador 2	10	1	1

$$\text{TempoEx} = \frac{\text{ciclos}_{CPU}}{f} = \frac{\sum_{i=1}^{n} (CPI_i \times C_i)}{f}$$

$$\text{MIPS} = \frac{I}{\text{TempoEx} \times 10^6}$$

$$\text{TempoEx}_1 = \frac{(5\times1+1\times2+1\times3)\times10^9}{4\times10^9} = 2{,}5\ \text{s}$$

$$\text{MIPS}_1 = \frac{(5+1+1)\times10^9}{2{,}5\times10^6} = 2800$$

$$\text{TempoEx}_2 = \frac{(10\times1+1\times2+1\times3)\times10^9}{4\times10^9} = 3{,}75\ \text{s}$$

$$\text{MIPS}_2 = \frac{(10+1+1)\times10^9}{3{,}75\times10^6} = 3200$$

Cálculo do MIPS (alternativo)

$$\frac{\text{instruções}}{\text{segundo}} = \frac{\text{instruções}}{\text{ciclo}} \times \frac{\text{ciclos}}{\text{segundo}} = \frac{1}{\text{CPI}} \times f = \frac{f}{\text{CPI}}$$

$$\text{MIPS} = \frac{\text{instruções} / \text{s}}{10^6} = \frac{\text{f}}{10^6 \times \text{CPI}}$$

Cálculo do MIPS (alternativo)

$$\frac{\text{instruções}}{\text{segundo}} = \frac{\text{instruções}}{\text{ciclo}} \times \frac{\text{ciclos}}{\text{segundo}} = \frac{1}{\text{CPI}} \times f = \frac{f}{\text{CPI}}$$

$$\text{MIPS} = \frac{\text{instruções} / \text{s}}{10^6} = \frac{\text{f}}{10^6 \times \text{CPI}}$$

Classe	CPI	Fração
A	5	0,33
B	2	0,50
C	4	0,10
D	4	0,07

f = 200 MHz

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

$$\text{MIPS} = \frac{200 \times 10^6}{10^6 \times 3,33} = 60$$

Consumo de energia e potência

- **Na era do PC**
  - ***Notebooks***
    - » **Maximizar tempo de bateria ⇒**
    - » **Minimizar consumo de energia**
  - ***Desktops* e *servers***
    - » **Viabilizar resfriamento ⇒**
    - » **Minimizar consumo de potência**
- **Na era pós-PC**
  - **Dispositivos pessoais móveis**
    - » **Maximizar tempo de bateria ⇒**
    - » **Minimizar consumo de energia**
  - **Cloud (*warehouse scale computing*)**
    - » **Maximizar a eficiência energética (operações/J)**

Estudo de caso 2: SPEC Power

- **Começou com**
  - ***SPEC benchmark for Java business applications***
- **Consumo de potência de servidores sob diferentes níveis de carga**
  - **Desempenho: operações/s (ssj_ops)**
  - **Potência: Watts (Joules/s)**
- **Métrica de eficiência energética**
  - **Quantas operações por Joule?**

Exemplo: SPECpower_ssj2008 p/ Xeon X5650

Target Load %	Performance (ssj_ops)	Average Power (Watts)
100%	865,618	258
90%	786,688	242
80%	698,051	224
70%	607,826	204
60%	521,391	185
50%	436,757	170
40%	345,919	157
30%	262,071	146
20%	176,061	135
10%	86,784	121
0%	0	80
Overall Sum	4,787,166	1,922
Σssj_ops/Σpower =		2,490

$$\text{Overall ssj\_ops per Watt} = \left(\sum_{i=0}^{10} \text{ssj\_ops}_i\right) / \left(\sum_{i=0}^{10} \text{power}_i\right)$$

Mais informações?

- **Acesso aos SPEC benchmarks:**
  - **http://www.spec.org/cpu2006/ (CPU)**
  - **http://www.spec.org/power_ssj2008/ (Power)**