

Technische Grundlagen der Informatik 2

Rechnerorganisation

Kapitel 4: Rechenleistung

Prof. Dr. Ben Juurlink

Fachgebiet: Architektur eingebetteter Systeme
Institut für Technische Informatik und Mikroelektronik
Fak. IV – Elektrotechnik und Informatik

SS 2014

- Nach diesem Kapitel sollten Sie in der Lage sein...
 - Rechenleistung zu messen, zu protokollieren und zusammenzufassen.
 - Intelligente Entscheidungen zu treffen.
 - den Marketing-Hype zu durchblicken.
 - Ausführungszeit bei gegebener Anzahl der Befehle, **Cycles Per Instruction** (CPI) und Taktfrequenz zu berechnen.
 - Gesetz von Amdahl anzuwenden.
 - die weitverbreitetste Benchmark-Suite benennen zu können.



- Rechenleistung ist der Schlüssel der zugrundeliegenden organisatorischen Motivation.
 - *Warum ist manche Hardware für bestimmte Programme besser als andere?*
 - *Welche Faktoren der Systemleistung sind hardwarebezogen? (Brauchen wir eine neue Maschine oder ein neues Betriebssystem?)*
 - *Wie beeinflusst der Maschinenbefehlssatz die Rechenleistung?*

- Rechenleistung definieren
- Antwortzeit versus Durchsatz
- CPU-Zeit
- CPU-Leistungsgleichung
- Rechenleistung vergleichen
- Benchmarks
 - SPEC
 - Intel Pentium SPEC performance
- Gesetz von Amdahl
- MIPS und MFLOPS

- Welches Flugzeug hat die beste Leistung?



Flugzeug	Passagiere	Reichweite (Meilen)	Geschwindigkeit (Miles Per Hour)
Boeing 737-100	101	630	598
Boeing 747	470	4150	610
Concorde	132	4000	1350
Douglas DC-8-50	146	8720	544

- Abhängig von der Bewertung
 - Welcher ist der schnellste Flug nach New York? [→ **Latenz (Latency)**]
 - Wie bekommt man am schnellsten 300 Menschen nach New York?
[→ **Durchsatz (Throughput)**]

- Antwortzeit (*response time, latency*)
 - Wie lange dauert es, bis mein *Job* ausgeführt wird?
 - Wie lange dauert es, einen *Job* auszuführen?
 - Wie lange muss auf eine Datenbankabfrage gewartet werden?
- Durchsatz (*throughput*)
 - Wie viele *Jobs* können gleichzeitig auf der Maschine ausgeführt werden?
 - Wie viel Arbeit wird erledigt?
- *Wenn eine Maschine einen neuen Prozessor bekommt: Was erhöht sich?*
- *Wenn wir eine neue Maschine ins Labor stellen: Was erhöhen wir?*

- **Verstrichene Zeit (*elapsed time*)**
 - Zählt alles (*Platten- und Speicherzugriffe, I/O , etc.*)
 - Eine brauchbare Zahl, aber selten gut zum Vergleichen
- **CPU-Zeit**
 - Zählt weder I/O, noch die Ausführungszeit anderer Programme
 - Kann in System- und Benutzer-CPU-Zeit zerlegt werden
- Unser Fokus: **Benutzer-CPU-Zeit**
 - Zeit, die zur Ausführung von Code-Zeilen, die „in“ unserem Programm sind, verwendet wird
- Ausgabe des Unix-Kommandos `time`:

90.7u 12.9s 2:39 65%

Benutzer-CPU-Zeit System-CPU-Zeit Verstrichene Zeit % CPU-Zeit

$$T = N_{instr} \cdot CPI \cdot t_{cycle} = \frac{N_{instr} \cdot CPI}{f}$$

wobei

- N_{instr} = Anzahl vom Programm benötigter Maschinenbefehle
- CPI = (durchschnittliche) Taktzyklen pro Maschinenbefehl (*Cycles Per Instruction*)
- t_{cycle} = Taktzykluszeit (*cycle time*)
- f = Taktfrequenz = $1/t_{cycle}$
- Ein 4-GHz Takt hat eine Taktzykluszeit von

$$\frac{1}{4 \cdot 10^9} = 250 \cdot 10^{-12} \text{ s} = 250 \text{ Picosekunden (ps)}$$

- Ein 2-GHz Prozessor führt ein Programm mit 5 Millionen Befehlen aus:
 - 52% sind arithmetische Befehle, die jeweils 4 Taktzyklen benötigen
 - 25% sind Ladebefehle, die jeweils 5 Taktzyklen benötigen
 - 10% sind Speicherbefehle, zu je 4 Taktzyklen
 - 11% sind Verzweigungen, zu je 3 Taktzyklen
 - 2% sind Sprünge, zu je 3 Taktzyklen
- Wie ist der CPI-Wert des Programms?
 - $0.62 \cdot 4 + 0.25 \cdot 5 + 0.13 \cdot 3 = 4.12$
- Wie ist die Ausführungszeit?
 - $5 \cdot 10^6 \cdot 4.12 / 2 \cdot 10^9 = 10.3 \text{ ms}$

- Die CPU-Leistungsgleichung $T = N_{instr} \cdot CPI \cdot t_{cycle}$ zeigt, wie wir die Leistungsfähigkeit steigern können.
- Zur Erhöhung der Leistungsfähigkeit (alles andere bleibt gleich) können wir ... (*erhöhen* oder *verringern*?)
 - Die Zahl der nötigen Taktzyklen eines Programms _____, oder
 - Die Taktzykluszeit _____ oder, anders gesagt,
 - Die Taktfrequenz _____.
- Wie können wir was verbessern?

	Befehlsanzahl	CPI	Zykluszeit
Programm	X	(X)	
Compiler	X	(X)	
Befehlssatz	X	X	(X)

- Wenn es nur ein Programm gibt, ist es klar, welcher Computer schneller ist.
- Wenn es mehrere Programme gibt, wird es knifflig!
- Beispiel:

	Computer A	Computer B	Computer C
Programm P_1 (sek)	1	10	20
Programm P_2 (sek)	1000	100	20
Total (sek)	1001	110	40

- Totale Ausführungszeit ist ein konsistentes Auswertungsmaß.
 - B ist 9,1-mal schneller als A für die Programme P_1 und P_2
 - C ist 25-mal schneller als A für die Programme P_1 und P_2

- Wir können auch das **arithmetische Mittel** verwenden:

$$\frac{1}{n} \sum_{i=1}^n Time_i$$

- $Time_i$ Ausführungszeit des Programms P_i
 - n Anzahl Programme
- Was ist, wenn die Programme nicht gleich häufig laufen?
- Zwei Herangehensweisen:
 - Gewichtete Ausführungszeit
 - Normalisieren der Ausführungszeiten auf eine Referenzmaschine und Durchschnittermittlung

- Gewichtete Ausführungszeit:
 - Jedes Programm bekommt einen Gewichtungsfaktor, der relative Ausführungsfrequenz anzeigt.
 - Gewichtungsfaktoren summieren sich zu 1.
- Gewichtetes arithmetisches Mittel: $\sum_{i=1}^n Weight_i \times Time_i$
- Annahme: $Weight_1 (P_1)$ ist 0,8 und $Weight_2 (P_2)$ ist 0,2.

	Computer A	Computer B	Computer C
Programm P_1 (sek)	1	10	20
Programm P_2 (sek)	1000	100	20
Arithmetisches Mittel (sek)	500,5	55	20
Gewichtetes Mittel (sek)	200,8	28	20



- Ausführungszeit auf eine **Referenzmaschine** normalisieren
- Durchschnitt ermitteln
- Aber **nicht** das arithmetische Mittel verwenden um den Durchschnitt der normalisierten Ausführungszeiten zu ermitteln.
 - Kann, abhängig von Referenzmaschine, zu anderen Ergebnissen führen!

	A	B	C
P ₁ (sek)	1	10	20
P ₂ (sek)	1000	100	20

	Normalisiert auf A			Normalisiert auf B			Normalisiert auf C		
	A	B	C	A	B	C	A	B	C
P ₁ (sek)	1.0	10.0	20.0	0.1	1.0	2.0	0.05	0.5	1.0
P ₂ (sek)	1.0	0.1	0.02	10.0	1.0	0.2	50	5.0	1.0
Arith. Mittel (sek)	1.0	5.05	10.01	5.05	1.0	1.1	25.025	2.75	1.0



- Benutzen Sie stattdessen das **geometrische Mittel**:

$$\sqrt[n]{\prod_{i=1}^n Ratio_i}$$

- $Ratio_i$ = Ausführungszeit von Programm P_i auf der zu bewertende Maschine normalisiert zur Ausführungszeit auf der Referenzmaschine
- dieser Wert hat jedoch keine physikalische Bedeutung!

	Normalisiert auf A			Normalisiert auf B			Normalisiert auf C		
	A	B	C	A	B	C	A	B	C
P_1 (sek)	1.0	10.0	20.0	0.1	1.0	2.0	0.05	0.5	1.0
P_2 (sek)	1.0	0.1	0.02	10.0	1.0	0.2	50	5.0	1.0
Geom. Mittel (sek)	1.0	1.0	0.63	1.0	1.0	0.63	1.58	1.58	1.0



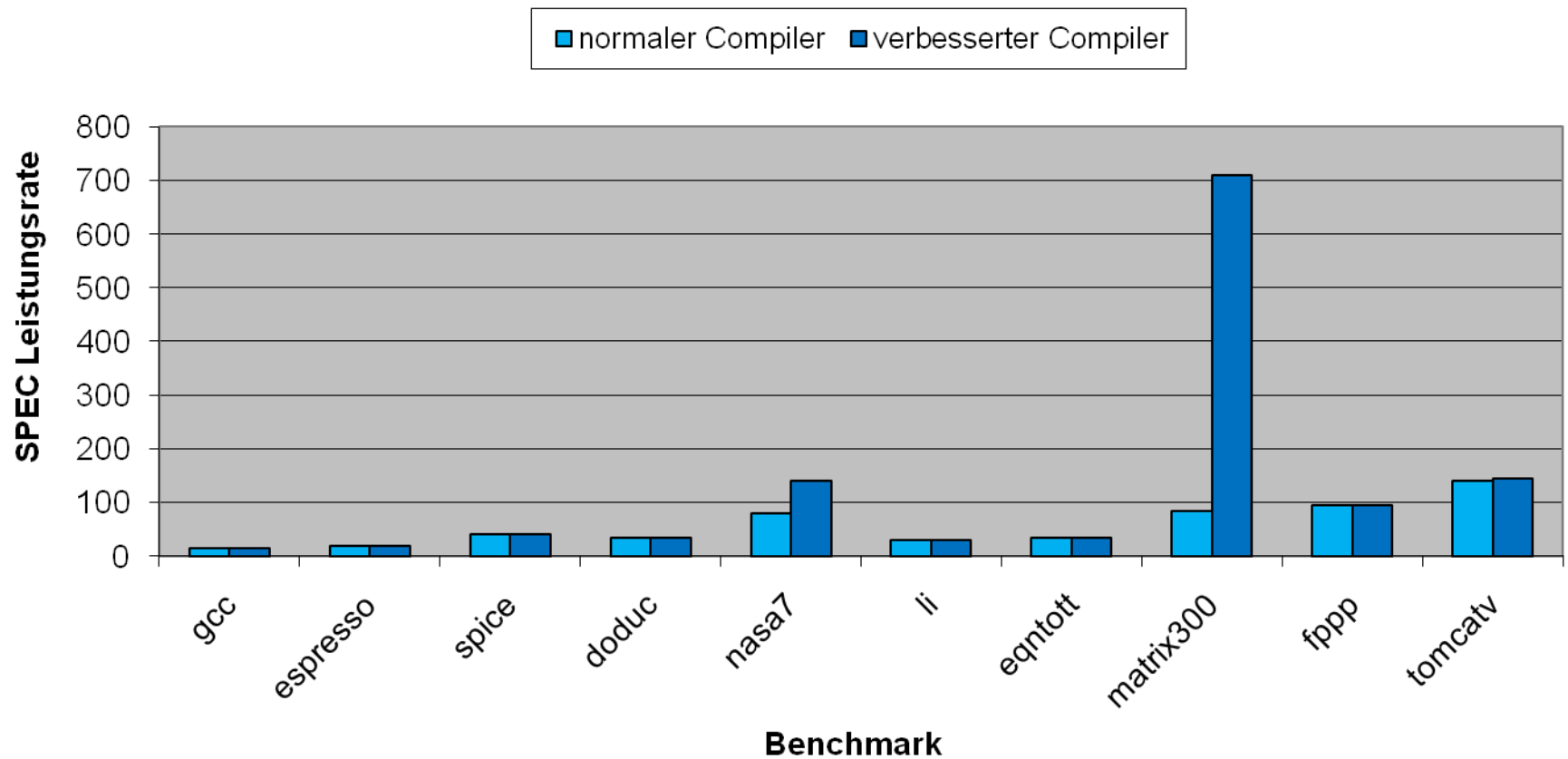
- Rechenleistung wird am besten durch das Ausführen echter Applikationen bestimmt.
 - Typische Programme für den erwarteten *Workload* benutzen, oder
 - typische Applikationen der erwarteten Klasse, z.B.: Compiler/Editoren, Wissenschaftliche Anwendungen, Grafik, etc.
- Kleine Benchmarks
 - Schön für Architekten und Designer
 - Leicht zu standardisieren
 - Können missbraucht werden
- SPEC (*Standard Performance Evaluation Corporation*)
 - Unternehmen haben sich auf Set realer Programme und Eingaben geeinigt.
 - Wertvoller Indikator für Rechenleistung (und Compilertechnologie)
 - Können immer noch missbraucht werden.



*“An embarrassed Intel Corp. acknowledged Friday that a **bug in a software program known as a compiler** had led the company to overstate the speed of its microprocessor chips on an industry benchmark by 10 percent. However, industry analysts said the coding error...was a sad commentary on a common industry practice of “**cheating**” on standardized performance tests...The error was pointed out to Intel two days ago by a competitor, Motorola ...came in a test known as SPECint92...Intel acknowledged that it had “**optimized**” **its compiler** to improve its test scores. The company had also said that it did not like the practice but felt to compelled to make the optimizations because **its competitors were doing the same thing**... At the heart of Intel’s problem is the practice of “**tuning**” compiler programs to recognize certain computing problems in the test and then substituting special handwritten pieces of code...”*

Saturday, January 6, 1996 New York Times

- Compiler-„Verbesserungen“ und Rechenleistung





- SPEC Mitglieder:
 - Acer Inc. * Action S.A. * **Advanced Micro Devices** * **Apple Inc.** * ASUSTeK Computer Inc. * Avere Systems * BlueArc * Bull S.A. * **Cisco Systems, Inc.** * Citrix Online * CommuniGate Systems * Dell * E4 Computer Engineering SPA * EMC * FORMAT Sp. z o.o. * Fujitsu * Hitachi Data Systems * Hitachi Ltd. * HP * Huawei Technologies * **IBM** * Incom S.A. * **Intel** * Itautec S/A * Lenovo * Microsoft * NEC - Japan * Neptun * NetApp * Novell * NTT System * **NVIDIA** * **Oracle** * Parallels * Platform Computing Inc. * Principled Technologies * The Portland Group * QLogic Corporation * **Red Hat** * SAP AG * SGI * **Sun Microsystems** * Super Micro Computer, Inc. * Symantec Corporation * Unisys * Via Technologies * Virtustream Ltd. * **VMware** * WSO2
- Derzeitige Haupt-Benchmark-Suite: SPEC CPU2006
 - CINT2006: Integer Benchmarks
 - CFP2006: Gleitkomma Benchmarks
- Webseite: www.spec.org



Integer Benchmarks (C und C++)

Name	Beschreibung
gzip	Komprimierung
vpr	FPGA Schaltkreis- und Leiterbahnanordnung
gcc	GNU C-Compiler
mcf	Kombinatorische Optimierung
crafty	Schachprogramm
parser	Textverarbeitungsprogramm
eon	Computervisualisierung
perlbmk	Perl-Anwendung
gap	Gruppentheorie, Interpreter
vortex	Objektorientierte Datenbank
bzip2	Komprimierung
twolf	Ort- und Routensimulation

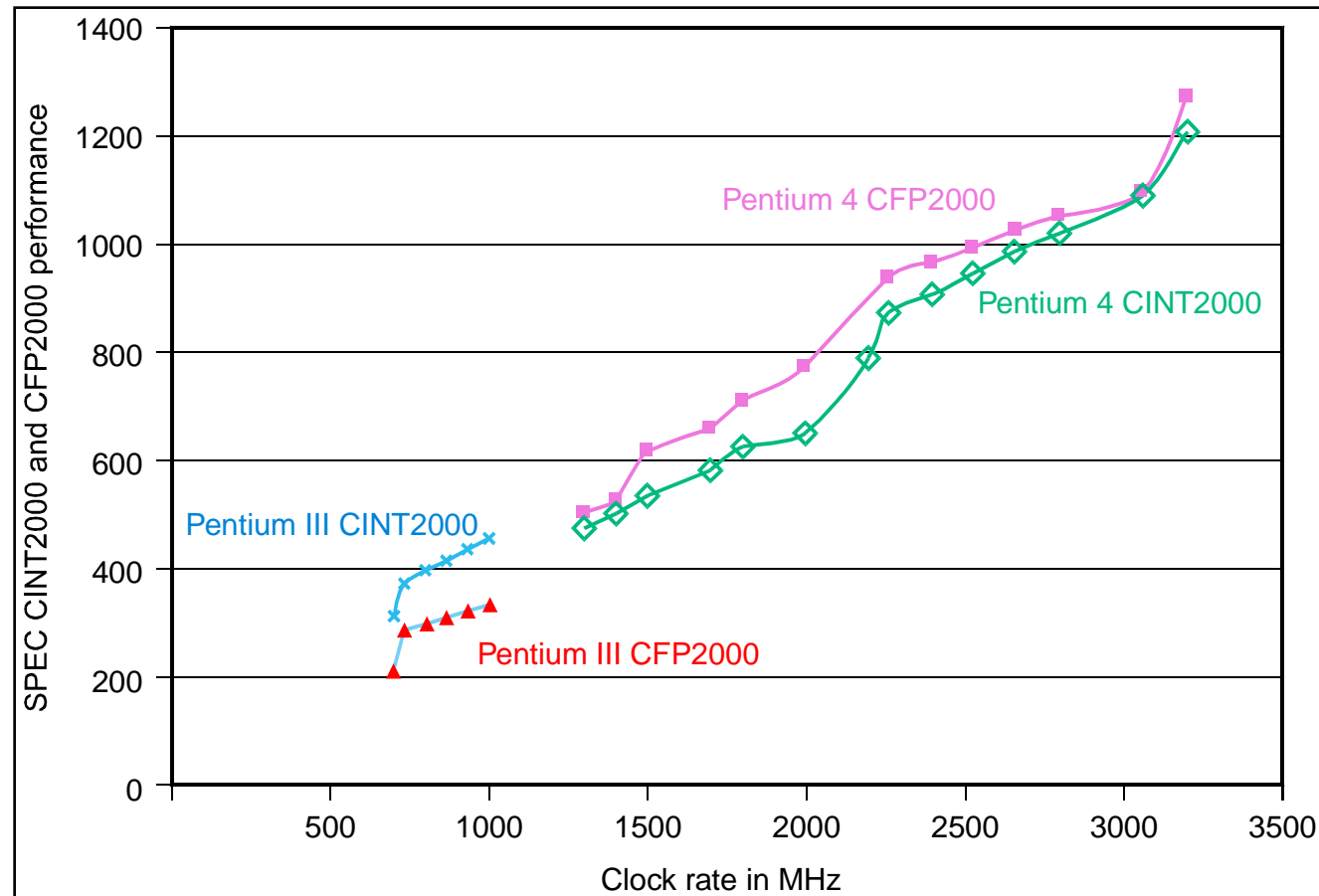
Gleitkomma-Benchmarks (Fortran und C)

Name	Beschreibung
wupwise	Quanten-Chromodynamik
applu	Parabolische/elliptische partielle Differentialgleichungen
mgrid	3D-Mehrgitterverfahren: Potenzialfeld
swim	Flachwassersimulation
mesa	3D-Grafikbibliothek
galgel	Strömungsmechanik-Simulationen
art	Bilderkennung mit neuronalen Netzen
equake	Simulation Ausbreitung seismischer Wellen
facerec	Bilderkennung von Gesichtern
ammp	Chemische Simulationen
lucas	Primzahlentests
fma3d	Crash-Simulation mit finiten Elementen
sixtrack	Beschleunigerkonstruktion für hochenergetische Atomphysik
apsi	Meteorologie: Schadstoffausbreitung



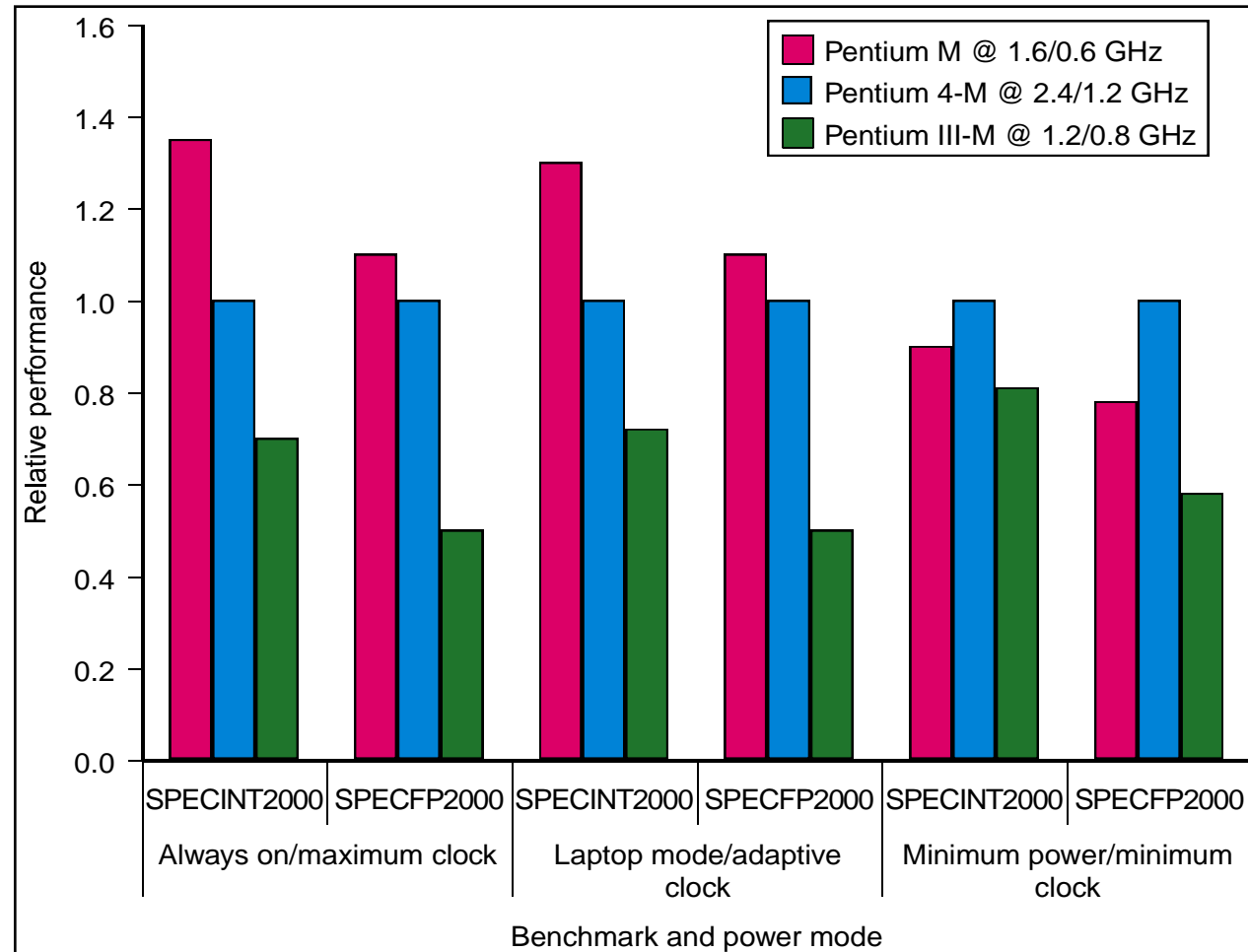
*Führt Verdopplung
der Taktfrequenz
zur doppelten
Rechenleistung?*

*Kann Maschine mit
geringerer Takt-
frequenz über
bessere
Rechenleistung
verfügen?*



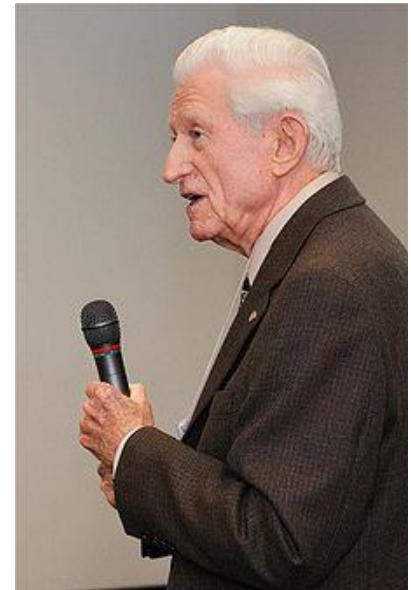
Stromverbrauch ist entscheidende Begrenzung der Leistungsfähigkeit.

Bei eingebetteten Systemen Stromverbrauch so wichtig wie Leistungsfähigkeit.





- Angenommen wir können
 - Anteil f ($0 \leq f \leq 1$) der Gesamtausführungszeit
 - um ein Faktor x verbessern,
 wie hoch wird der gesamte **Speedup S** sein?



$$T_{\text{new}} = (1 - f)T_{\text{old}} + \frac{fT_{\text{old}}}{x}$$

$$S = \frac{T_{\text{old}}}{T_{\text{new}}} = \frac{T_{\text{old}}}{(1 - f)T_{\text{old}} + (f / x)T_{\text{old}}} = \frac{1}{1 - f + f / x}$$

- Angenommen ein Programm läuft in 100 Sekunden, wobei Multiplikationen 80 Sekunden davon verbrauchen.
 - Um wieviel muss die Geschwindigkeit der Multiplikation erhöht werden, wenn das Programm 4-mal schneller laufen soll?

$$S = \frac{1}{1 - f + f / x} \Rightarrow 4 = \frac{1}{0.2 + 0.8 / x} \Rightarrow 0.05 = 0.8 / x \Rightarrow x = 16$$

- Wie wird es 5-mal schneller ?
- Erneut das 3. Design Prinzip:

Optimiere den häufig vorkommenden Fall



- Angenommen wir verbessern einen Prozessor, sodass alle Gleitkomma-Befehle fünfmal schneller laufen. Wenn die Ausführungszeit vom Benchmark vor der Gleitkomma-Verbesserung 10 Sekunden ist, wie wird der *Speedup* sein, wenn fünf Sekunden für Gleitkomma-Befehlen verwendet werden?
- Wir suchen einen Benchmark für die neue Gleitkomma-Einheit und wollen, dass dieser über das gesamte Programm einen *Speedup* von 3 anzeigt. Ein Benchmark den wir erwägen läuft mit der alten Gleitkomma-Hardware 100 Sekunden lang. Wie hoch muss der Anteil der Gleitkomma-Befehle an der gesamten Ausführungszeit sein, um bei diesem Benchmark den gewünschten *Speedup* hervorzurufen?



- MIPS (**M**illion **I**nstructions **p**er **S**econd)

$$\text{MIPS} = \frac{\text{Anzahl von Instruktionen}}{\text{Ausführungszeit in Sekunden} \times 10^6}$$

- Probleme bei der Verwendung von MIPS als Maß:
 - Leistungsfähigkeit der Befehle wird nicht berücksichtigt
 - Prozessoren mit unterschiedlichen Befehlssätzen können nicht verglichen werden.
 - MIPS variiert für verschiedene Programme
 - Es gibt keinen einzelnen MIPS-Wert für alle Programme
 - MIPS kann invers zur Leistung variieren
 - Programm mit 1M Befehlen und CPI 1,0 hat höheren MIPS-Wert als Programm für das selbe Problem mit 0,4M Befehlen und CPI 2.0
- MFLOPS (*mega (10⁶) floating point operations per second*) ähnlich



- Ausführungszeit ist das einzig gültige Rechenleistungsmaß.
- CPU-Leistungsgleichung: $T = N_{instr} \times CPI \times t_{cycle}$
- Wenn Programme nicht gleichhäufig:
 - Gewichtetes arithmetisches Mittel
 - Auf Referenzmaschine normalisieren und Durchschnitt bilden (aber nicht arithmetischen Durchschnitt)
- SPEC (Standard Performance Evaluation Corporation)
 - Satz von realen Programmen und Eingaben zum Leistungsvergleich
 - www.spec.org

- Amdahls Gesetz:
$$S = \frac{1}{1 - f + f / X}$$