Department of Electrical Engineering, Feng-Chia University

Chapter 1
Computer Abstractions and Technology (Part 2)

王振傑 (Chen-Chieh Wang)
ccwang@mail.ee.ncku.edu.tw

Department of Electrical Engineering, Feng-Chia University

Outline

- Introduction
- CPU Performance and Its Factors
- Evaluating Performance & Benchmarks
- Fallacies and Pitfalls

Performance

- Measure, Report, and Summarize
- Make intelligent choices
- See through the marketing hype
- Key to understanding underlying organizational motivation
- Power, area, and performance tradeoffs

Why is some hardware better than others for different programs?

What factors of system performance are hardware related?
(e.g., Do we need a new machine, or a new operating system?)

How does the machine's instruction set affect performance?

What is performance?

Text Book : P27 ~ P28

Which of these airplanes has the best performance?

Airplane	Passengers	Range (mi)	Speed (mph)	Throughput (Passenger x mph)
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

Boeing 777

Boeing 747

Concorde

DC-8

Text Book : P28

Response Time and Throughput

- Response Time (Execution Time)
  - How long does it take for my job to run?
  - How long does it take to execute a job?
  - How long must I wait for the database query?
- Throughput
  - How many jobs can the machine run at once?
  - What is the average execution rate?
  - How much work is getting done?

If we upgrade a machine with a new processor what do we increase?
If we add a new machine to the lab what do we increase?

Text Book : P30

Execution Time

- Elapsed Time (Wall-Clock Time)
  - Counts everything *(disk and memory accesses, I/O , etc.)*
  - A useful number, but often not good for comparison purposes
- CPU time
  - Doesn't count I/O or time spent running other programs
  - Can be broken up into **system time**, and **user time**
- Our focus: user CPU time
  - Time spent executing the lines of code that are "in" our program

Book's Definition of Performance

- For some program running on machine X,

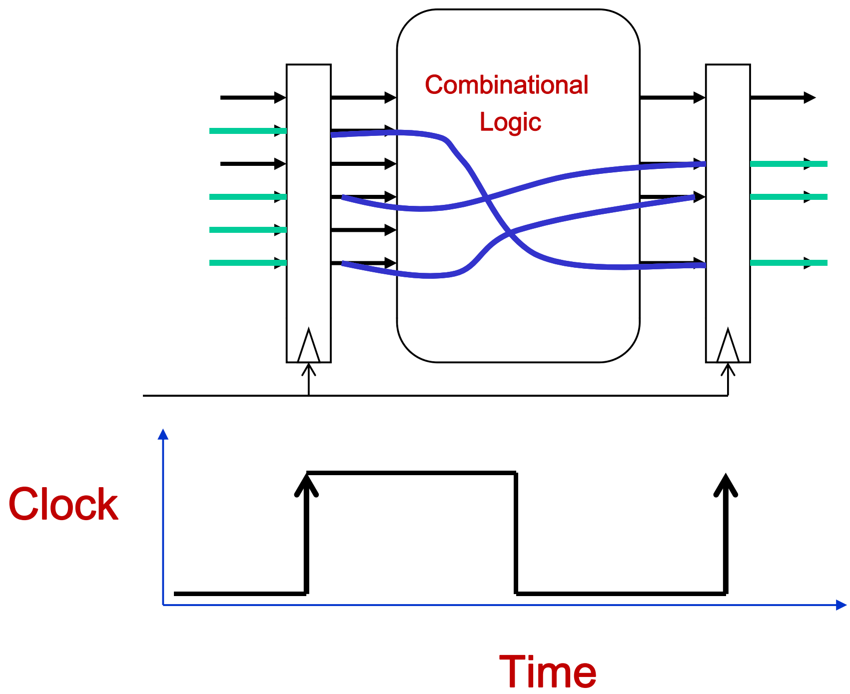
$$\text{Performance}_X = \frac{1}{\text{Execution time}_X}$$

- "X is n times faster than Y"

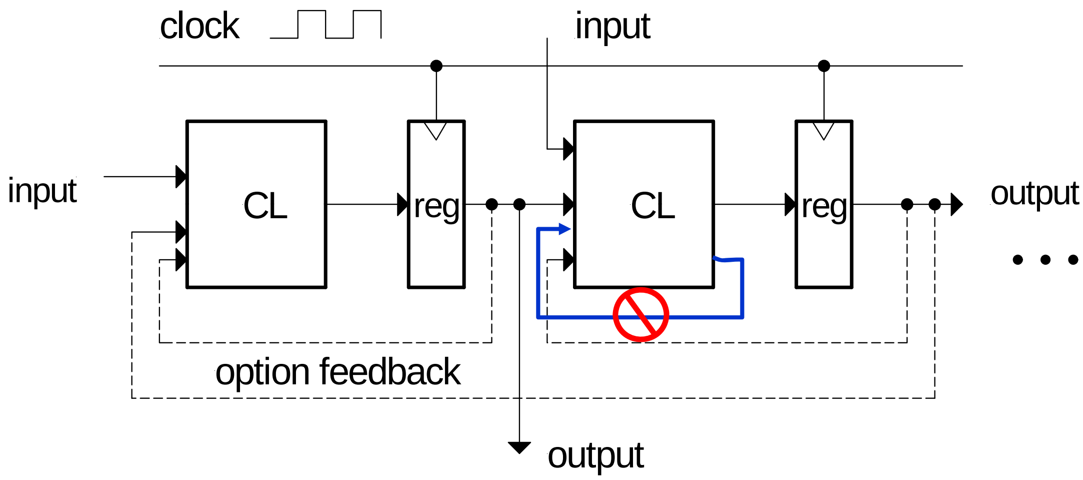
$$\frac{\text{Performance}_X}{\text{Performance}_Y} = n$$

- Problem:
  - machine A runs a program in 10 seconds
  - machine B runs the same program in 15 seconds

What makes Digital Systems tick?



Synchronous Circuit Design

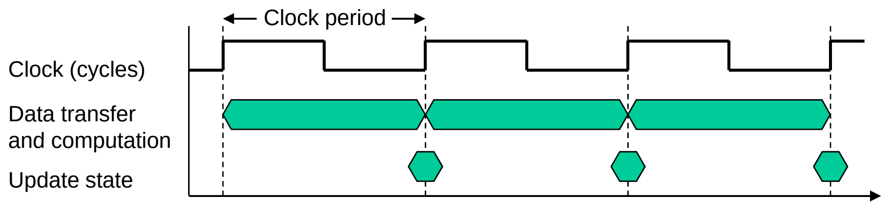


- Combinational Logic Blocks (CL)
  - Acyclic
  - no internal state (no feedback)
  - output only a function of inputs
- Registers (reg)
  - collections of flip-flops
- Clock
  - distributed to all flip-flops
- ALL CYCLES GO THROUGH A REG!

Text Book : P31

Clock Cycles

- Operation of digital hardware governed by a constant-rate clock



- Instead of reporting execution time in seconds, we often use cycles

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

- Cycle time = time between ticks = seconds per cycle
- Clock rate (frequency) = cycles per second (1 Hz. = 1 cycle/sec)

A 4GHz. clock has a $\frac{1}{4\times10^{9}}\times10^{12} = 250\ \text{picoseconds (ps)}$ cycle time

Outline

- Introduction
- CPU Performance and Its Factors
- Evaluating Performance & Benchmarks
- Fallacies and Pitfalls

Text Book : P32

How to Improve Performance

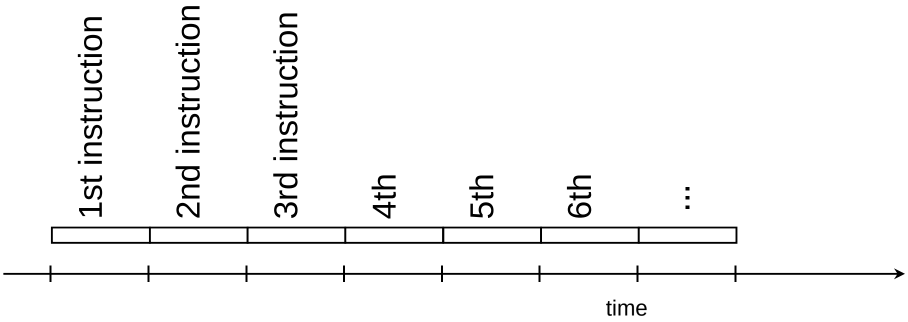
$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

So, to improve performance (everything else being equal) you can either (increase or decrease?)

________ the # of required cycles for a program, or
________ the clock cycle time or, said another way,
________ the clock rate.

How many cycles are required for a program?

- Could assume that # of cycles = # of instructions

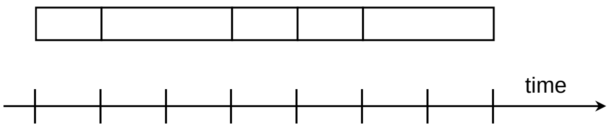


This assumption is incorrect,

different instructions take different amounts of time on different machines.

Why? hint: remember that these are machine instructions, not lines of C code

Different numbers of cycles for different instructions



- Multiplication takes more time than addition
- Floating point operations take longer than integer ones
- Accessing memory takes more time than accessing registers
- *Important point: changing the cycle time often changes the number of cycles required for various instructions (more later)*

Example

Our favorite program runs in 10 seconds on computer A, which has a 2 GHz clock. We are trying to help a computer designer build a new machine B, that will run this program in 6 seconds. The designer has determined that a substantial increase in the clock rate is possible, but this increase will affect the rest of the CPU design, causing machine B to require 1.2 times as many clock cycles as machine A for this program. What clock rate should we tell the designer to target?"

- Don't Panic, can easily work this out from basic principles

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

Now that we understand cycles

- A given program will require
  - some number of instructions (machine instructions)
  - some number of cycles
  - some number of seconds
- We have a vocabulary that relates these quantities:
  - cycle time (seconds per cycle)
  - clock rate (cycles per second)
  - CPI (cycles per instruction)

    a floating point intensive application might have a higher CPI
  - MIPS (millions of instructions per second)

    this would be higher for a program using simple instructions

Text Book : P35 ~ P38

Factors that affect performance

$$\text{Time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

➔ *CPU time = IC x CPI x Clock cycle time* ★

- IC (instruction count) :
  - ISA, compiler
- CPI (cycles per instruction) :
  - ISA, micro-architecture, cache memory, pipeline…
- Clock cycle time:
  - process technology, micro-architecture

Performance

- Performance is determined by execution time
- Do any of the other variables equal performance?
  - # of cycles to execute program?
  - # of instructions in program?
  - # of cycles per second?
  - average # of cycles per instruction?
  - average # of instructions per second?
- Common pitfall: thinking one of the variables is indicative of performance when it really isn't.

Text Book : P33

CPI in More Detail

- If different instruction classes take different numbers of cycles

$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)$$

- Weighted average CPI

$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \underbrace{\frac{\text{Instruction Count}_i}{\text{Instruction Count}}}_{\text{Relative frequency}} \right)$$

Text Book : P34

CPI Example

Suppose we have two implementations of the same instruction set architecture (ISA).
For some program,

Machine A has a clock cycle time of 250 ps and a CPI of 2.0
Machine B has a clock cycle time of 500 ps and a CPI of 1.2

What machine is faster for this program, and by how much?

CPU Time = IC x CPI x CCT

Text Book : P35 ~ P36

of Instructions Example

A compiler designer is trying to decide between two code sequences for a particular machine. Based on the hardware implementation, there are three different classes of instructions: Class A, Class B, and Class C, and they require one, two, and three cycles (respectively).

The first code sequence has 5 instructions: 2 of A, 1 of B, and 2 of C
The second sequence has 6 instructions: 4 of A, 1 of B, and 1 of C.

Which sequence will be faster? How much?
What is the CPI for each sequence?

CPU clock cycles_1 =
CPU clock cycles_2 =

CPI_1 =
CPI_2 =

Outline

- Introduction
- CPU Performance and Its Factors
- Evaluating Performance & Benchmarks
- Fallacies and Pitfalls

Text Book : P48

Benchmarks

- Performance best determined by running a real application
  - Use programs typical of expected workload
  - Or, typical of expected class of applications
    e.g., compilers/editors, scientific applications, graphics, etc.
- Small benchmarks
  - nice for architects and designers
  - easy to standardize
  - can be abused
- SPEC (System Performance Evaluation Cooperative)
  - companies have agreed on a set of real program and inputs
  - can still be abused
  - valuable indicator of performance (and compiler technology)

Department of Electrical Engineering, Feng-Chia University

Text Book : P48

SPEC CPU Benchmark

- Programs used to measure performance
  - Supposedly typical of actual workload
- Standard Performance Evaluation Cooperative (SPEC)
  - Develops benchmarks for CPU, I/O, Web, …
- SPEC CPU2006
  - Elapsed time to execute a selection of programs
    - Negligible I/O, so focuses on CPU performance
  - Normalize relative to reference machine
  - Summarize as geometric mean of performance ratios
    - CINT2006 (integer) and CFP2006 (floating-point)

$$\sqrt[n]{\prod_{i=1}^{n} \text{Execution time ratio}_i}$$

Department of Electrical Engineering, Feng-Chia University

Text Book : P49

CINT2006 for Opteron X4 2356

Name	Description	IC×10^9	CPI	Tc (ns)	Exec time	Ref time	SPECratio
perl	Interpreted string processing	2,118	0.75	0.4	637	9,777	15.3
bzip2	Block-sorting compression	2,389	0.85	0.4	817	9,650	11.8
gcc	GNU C Compiler	1,050	1.72	0.4	724	8,050	11.1
mcf	Combinatorial optimization	336	10.00	0.4	1,345	9,120	6.8
go	Go game (AI)	1,658	1.09	0.4	721	10,490	14.6
hmmer	Search gene sequence	2,783	0.80	0.4	890	9,330	10.5
sjeng	Chess game (AI)	2,176	0.96	0.4	837	12,100	14.5
libquantum	Quantum computer simulation	1,623	1.61	0.4	1,047	20,720	19.8
h264avc	Video compression	3,102	0.80	0.4	993	22,130	22.3
omnetpp	Discrete event simulation	587	2.94	0.4	690	6,250	9.1
astar	Games/path finding	1,082	1.79	0.4	773	7,020	9.1
xalancbmk	XML parsing	1,058	2.70	0.4	1,143	6,900	6.0
Geometric mean							11.7

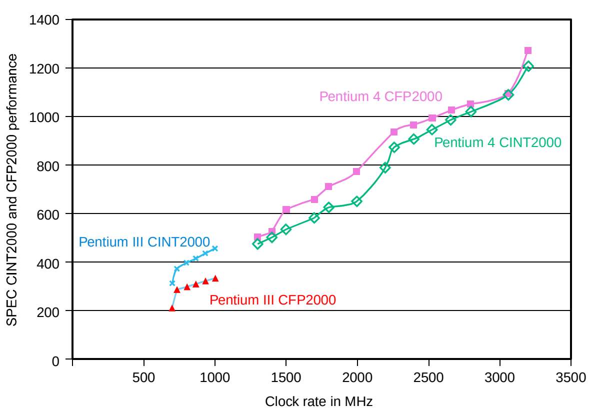
High cache miss rates

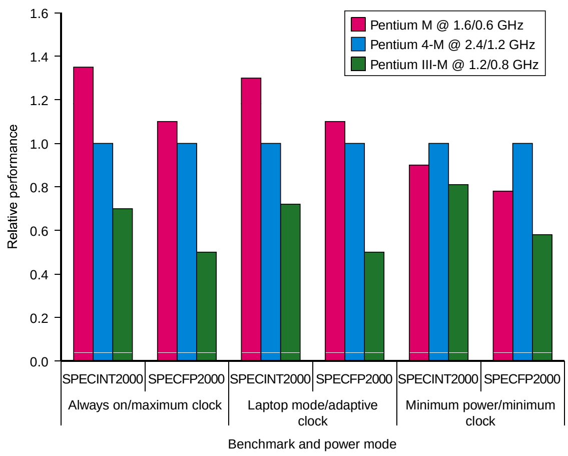
Department of Electrical Engineering, Feng-Chia University

SPEC 2000

Does doubling the clock rate double the performance?

Can a machine with a slower clock rate have better performance?





Text Book : P50

SPEC Power Benchmark

- Power consumption of server at different workload levels
  - Performance: ssj_ops/sec
  - Power: Watts (Joules/sec)

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

SPECpower_ssj2008 for Opteron X4

Target Load %	Performance (ssj_ops/sec)	Average Power (Watts)
100%	231,867	295
90%	211,282	286
80%	185,803	275
70%	163,427	265
60%	140,160	256
50%	118,324	246
40%	920,35	233
30%	70,500	222
20%	47,126	206
10%	23,066	180
0%	0	141
Overall sum	1,283,590	2,605
∑ssj_ops/ ∑power		493

Outline

- Introduction
- CPU Performance and Its Factors
- Evaluating Performance & Benchmarks
- Fallacies and Pitfalls

Text Book : P51

Pitfall : Amdahl's Law ★

- Improving an aspect of a computer and expecting a proportional improvement in overall performance

α: $\text{Fraction}_{\text{enhanced}}$

N: $\text{Speedup}_{\text{enhanced}}$

$$\text{Execution time}_{\text{new}} = \text{Execution time}_{\text{old}} \times \left((1-\alpha) + \frac{\alpha}{N} \right)$$

$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution time}_{\text{old}}}{\text{Execution time}_{\text{new}}} = \frac{1}{(1-\alpha) + \frac{\alpha}{N}}$$

$$\text{if } N \to \infty$$

$$\text{Speedup}_{\text{overall}} =$$

- *Corollary: Make the common case fast*

Amdahl's Law

EXAMPLE

Suppose a program runs in 100 seconds on a machine, with multiply responsible for 80 seconds of this time. How much do we have to improve the speed of multiplication if we want the program to run 4 times faster? How about making it 5 times faster?

ANSWER

Text Book : P52

Fallacy : Low Power at Idle

- Look back at X4 power benchmark
  - At 100% load: 295W
  - At 50% load: 246W (83%)
  - At 10% load: 180W (61%)
- Google data center
  - Mostly operates at 10% – 50% load
  - At 100% load less than 1% of the time
- Consider designing processors to make power proportional to load

Department of Electrical Engineering, Feng-Chia University

Pitfall : MIPS as a Performance Metric

- MIPS: Millions of Instructions Per Second
  - Doesn't account for
    - Differences in ISAs between computers
    - Differences in complexity between instructions

$$\text{MIPS} = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6}$$
$$= \frac{\text{Instruction count}}{\dfrac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

  - CPI varies between programs on a given CPU

Department of Electrical Engineering, Feng-Chia University

MIPS example (1/2)

- Two different compilers are being tested for a 4 GHz machine with three different classes of instructions: Class A, Class B, and Class C, which require one, two, and three cycles (respectively). Both compilers are used to produce code for a large piece of software.

  The first compiler's code uses 5 billion Class A instructions, 1 billion Class B instructions, and 1 billion Class C instructions.

  The second compiler's code uses 10 billion Class A instructions, 1 billion Class B instructions, and 1 billion Class C instructions.

- Which sequence will be faster according to MIPS?
- Which sequence will be faster according to execution time?

Department of Electrical Engineering, Feng-Chia University

MIPS example (2/2)

- **Answer:**

CPU clock cycles$_1$ =

CPU clock cycles$_2$ =

Execution time$_1$ =

Execution time$_2$ =

MIPS$_1$ =

MIPS$_2$ =

Department of Electrical Engineering, Feng-Chia University

Remember

- Performance is specific to a particular program/s
  - Total execution time is a consistent summary of performance
- For a given architecture performance increases come from:
  - increases in clock rate (without adverse CPI affects)
  - improvements in processor organization that lower CPI
  - compiler enhancements that lower CPI and/or instruction count
- Pitfall: expecting improvement in one aspect of a machine's performance to affect the total performance
- You should not always believe everything you read! Read carefully!