COL216
Computer Architecture

Performance

3rd March, 2022

Comparing Computers

- Functionality
- Performance/speed
- Power/energy consumption
- Cost
- Reliability
- Size/weight

Different Perspectives

- Individual user's perspective
- Service provider's perspective
- CPU designer's perspective
- Embedded system designer's perspective

Multiple parameters may be important

What are implications of architectural choices on these parameters?

Questions related to performance

- Can one system be better for one user whereas other system be better for another user ?
- Is the system performance only dependent on hardware? What about software?
- How does the machine's instruction set affect performance?

Defining Performance

- How do you define performance?
- Is there a unique way of defining it?

Two notions of "performance"

- Time (response time, execution time, latency)
  - How long does it take for my job to run?
  - How long must I wait for my query?
- Throughput (tasks per unit time)
  - How many jobs can the machine run at once?
  - What is the average execution rate?

Latency vs. Throughput

- Latency and throughput may often go together, but some time they may be in opposition - why?
- When is throughput more important than execution time?
- When is execution time more important than throughput?

Understanding Performance

- How are the following likely to effect response time and throughput?
  - Upgrade a machine with a new processor with faster clock.
  - Increasing the number of jobs (e.g., having a single computer service multiple users).
  - Adding a new machine in the lab.
  - Increasing the number of processors (e.g., in a network of ATM machines).

Performance Definitions

- Performance is in units of things-per-second
  - bigger is better
- If we are primarily concerned with response time

$$\text{performance} = \frac{1}{\text{execution\_time}}$$

Relative Performance

" X is n times faster than Y" means

$$n = \frac{\text{performance}(X)}{\text{performance}(Y)} = \frac{\text{execution\_time}(Y)}{\text{execution\_time}(X)}$$

Comparing Performance

- If an Intel processor runs a program in 8 seconds and an ARM processor runs the same program in 10 seconds, how many times faster is the Intel processor?
- n = 10 / 8 = 1.25 times faster (or 25% faster)
- Why might someone choose to buy the ARM processor in this case?

Definitions of Time

- Defined in different ways:
  - Response time : total time = CPU exec time + disk access time + waiting time.
  - CPU execution time : total time spent by CPU
  - User CPU time : time spent in user program
  - System CPU execution time : time spent by OS.
- For example, a program may have a system CPU time of 22 sec., a user CPU time of 90 sec., a CPU execution time of 112 sec., and a response time of 162 sec..

Computing CPU time

- The time to execute a given program can be computed as

  CPU time = CPU clock cycles x clock cycle time or

  CPU time = CPU clock cycles / clock rate

- CPU clock cycles = (instr/program) x (clock cycles/instruction)

  = Instruction count x CPI

  which gives

  CPU time = Instruction count x CPI x clock cycle time

  CPU time = Instruction count x CPI / clock rate

- The units for this are

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

Example of computing CPU time

- If clock rate = 500 MHz, find execute time for a program that executes 1,000 instructions, if the CPI for the program = 3.5?

CPU time = instruction count x CPI / clock rate

CPU time = 1000 x 3.5 / (500 x 10^6) sec = 7 μs

- If clock rate increases from 500 MHz to 600 MHz and the other factors remain the same, how many times faster will the computer be?

$$\frac{\text{CPU time old}}{\text{CPU time new}} = \frac{\text{clock rate new}}{\text{clock rate old}} = \frac{600\ \text{MHz}}{500\ \text{MHz}} = 1.2$$

Computing CPI

- CPI = avg. no. of cycles per instruction.

$$CPI = \sum_{i=1}^{n} CPI_i \times F_i$$

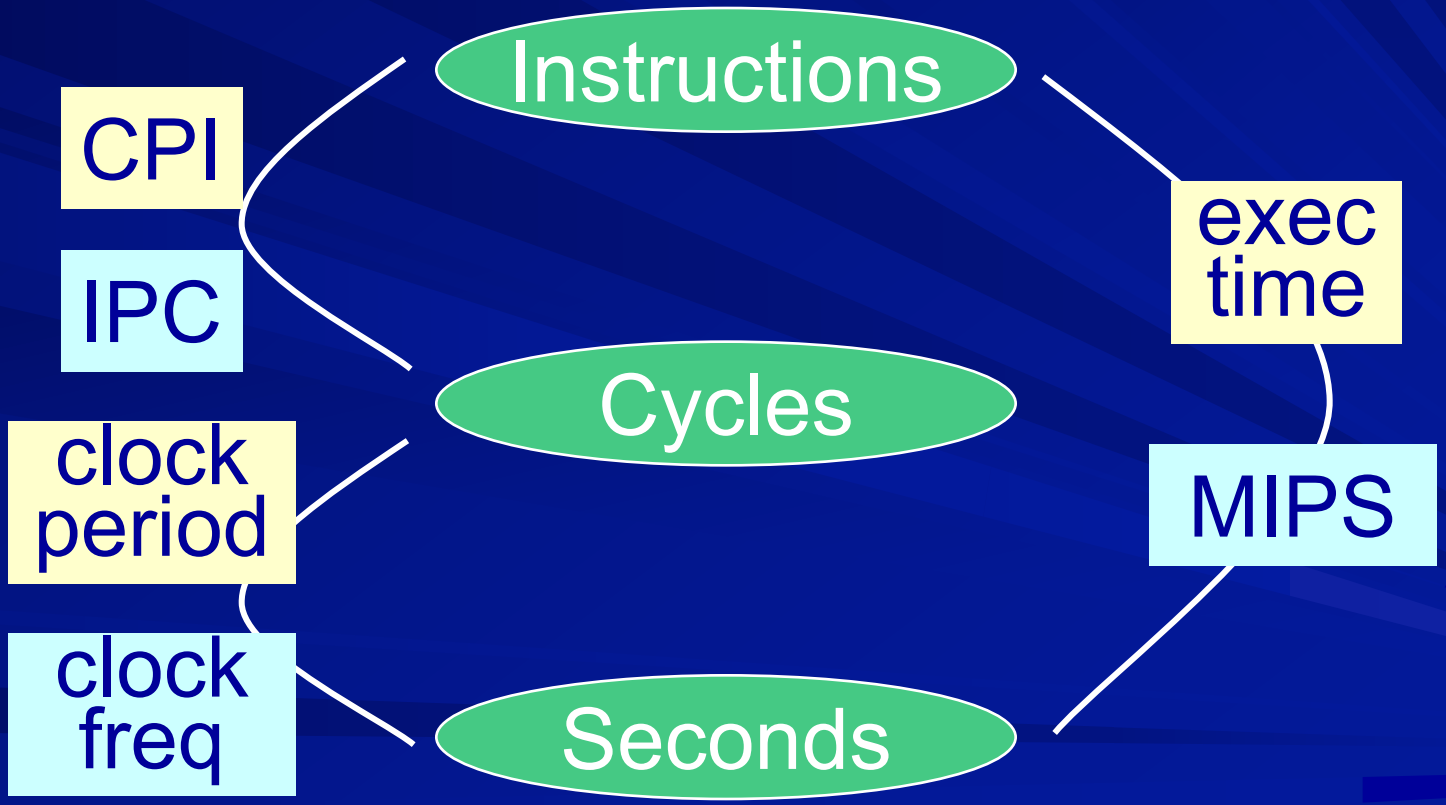
Example:

Op	F_i	CPI_i	CPI_ix F_i	% Time
DP	50%	4	2.0	50%
Load	20%	5	1.0	25%
Store	10%	4	0.4	10%
Branch	20%	3	0.6	15%
Total	100%		4.0	100%

Different numbers of cycles for different instructions

- Multiplication takes longer than addition
- Floating point operations take longer than integer ones
- Accessing memory takes more time than accessing registers
- Changing cycle time often changes number of cycles required for various instructions

Instructions↔Cycles↔Seconds



Performance

- Performance is determined by execution time
- Common pitfall: Use one of these variables as indicator of performance
  - # of cycles to execute program?
  - # of instructions in program?
  - # of cycles per second?
  - average # of cycles per instruction?
  - average # of instructions per second?

Example

- Computer A, (400 MHz clock), a program runs in 10 sec.
- Target: build a new machine B, the program runs in 6 sec.
- New (more expensive!) technology available to substantially increase the clock rate.
- This change will affect the rest of the CPU design, causing B to require 1.2 times as many clock cycles as A for the same program.
- What clock rate will make it possible for the designer to achieve the target?

Solution

- Two implementations (A and B) of the same instruction set architecture (ISA).

- If two machines have the same ISA which of our quantities (e.g., clock rate, CPI, execution time, # of instructions, MIPS) will always be identical?

Solution continued

A has clock cycle time = 2.5 ns
B has clock cycle time = *T* ns

- For given program,
  A has CPI = *c* , execution time = 10 s
  B has CPI = 1.2 *c* , execution time = 6 s

No. of instructions executed = *N* 10^9 for both

10 = *N* * *c* * 2.5 6 = *N* * 1.2 * *c* * *T*

T^{-1} = 1.2 *10 / (2.5 * 6) = .8 GHz = 800 MHz

Compiler factor

A compiler designer is to decide between two code sequences.

- First code sequence:
  - 2 of A, 1 of B, 2 of C
- Second code sequence:
  - 4 of A, 1 of B, 1 of C

3 classes of instructions:

- A : 1 cycle
- B : 2 cycles
- C : 3 cycles

- Which sequence will be faster?
  - How much?
  - CPI for each?

Compiler and MIPS

- Two compilers being tested for 100 MHz machine with 3 classes of instructions:
  - A (1 cycle), B (2 cycles), and C (3 cycles)
  - Compiler 1: 5M A, 1M B, 1M C instructions
  - Compiler 2: 10M A, 1M B, 1M C instructions
- Which sequence has higher MIPS?
- Which sequence has lower execution time?

Programs to test performance

- 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.

Computer Benchmarks

- Benchmark = program (s) used to evaluate computer performance.
- Allow a user to make performance comparisons
- Benchmarks should
  - Be representative of the type of applications
  - Not be overly dependent on one or two features
- Benchmarks can vary greatly in terms of their complexity and their usefulness.

Sources of Benchmarks

- Synthetic benchmarks
  - small benchmarks can be easily written
  - nice for architects and designers
  - easy to use, but can be abused
- SPEC (**S**tandard **P**erformance **E**valuation **C**orporation)
  http://www.spec.org

  "An ounce of honest data is worth a pound of marketing hype"
  - benchmarks derived from real programs

SPEC Benchmark Categories

- Cloud
- CPU
- Graphics/Workstations
- ACCEL/MPI/OMP
- Java Client/Server
- Mail Servers
- Storage
- Power
- Virtualization
- Web Servers

SPEC CPU Benchmarks

- SPEC CPU 92 => SPEC CPU 95 => SPEC CPU 2000 => SPEC CPU 2006 => SPEC CPU 2017
- SPEC CPU 2017: 43 benchmarks organized into 4 suites
  - SPECspeed 2017 Integer
  - SPECspeed 2017 Floating Point
  - SPECrate 2017 Integer
  - SPECrate 2017 Floating Point

Integer benchmarks

- Perl interpreter, GNU C compiler
- Route planning
- Network simulation
- Video compression, Data compression
- Tree search in games like chess
- Recursive solution generator (Sudoku)

Floating point benchmarks

- Modeling of physical phenomena like explosion, molecular dynamics, ocean, atmosphere etc.
- Optical tomography
- Ray tracing, 3-d rendering, animation
- Computational electromagnetics
- Image manupulation

Amdahl's Law

$$\text{Speedup} = \frac{\text{ExTime old}}{\text{ExTime new}} = \frac{\text{Performance new}}{\text{Performance old}}$$

Suppose that an enhancement accelerates a fraction $\text{Fraction}_{\text{enhanced}}$ of the task by a factor $\text{Speedup}_{\text{enhanced}}$

Execution Time After Improvement =
time unaffected + time affected / improvement

Amdahl's Law

$$ExTime_{new} = ExTime_{old} \times \left[(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}} \right]$$

$$Speedup = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$$

Example:

A program runs in 100 seconds on a machine, with multiply responsible for 80%. 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?

Example

- Suppose we enhance a machine making all floating-point instructions run five times faster.
- Let the execution time of some benchmark before the floating-point enhancement be 10 seconds.
- What will the speedup be if half of the 10 seconds is spent executing floating-point instructions?

$Speedup_{enhanced} = 5$

$ExTime_{old} = 10$

$Fraction_{enhanced} = 0.5$

$ExTime_{new} =$

$10\ (1-.5\ +.5/5) = 6$

$Speedup = 10/6 = 1.67$

Example continued

- We are looking for a benchmark to show off the new FP unit.
- Want the overall benchmark to show a speedup of 3.
- Benchmark under consideration runs for 100 seconds with the old floating-point hardware.
- How much of the execution time would FP instructions have to account for in this program in order to yield our desired speedup on this benchmark?

$Speedup_{enhanced} = 5$

$Speedup = 3$

$ExTime_{old} = 100$

$ExTime_{new} = 33.3$

$Fraction_{enhanced} = f$

$33.3 = 100(1-f + f/5)$

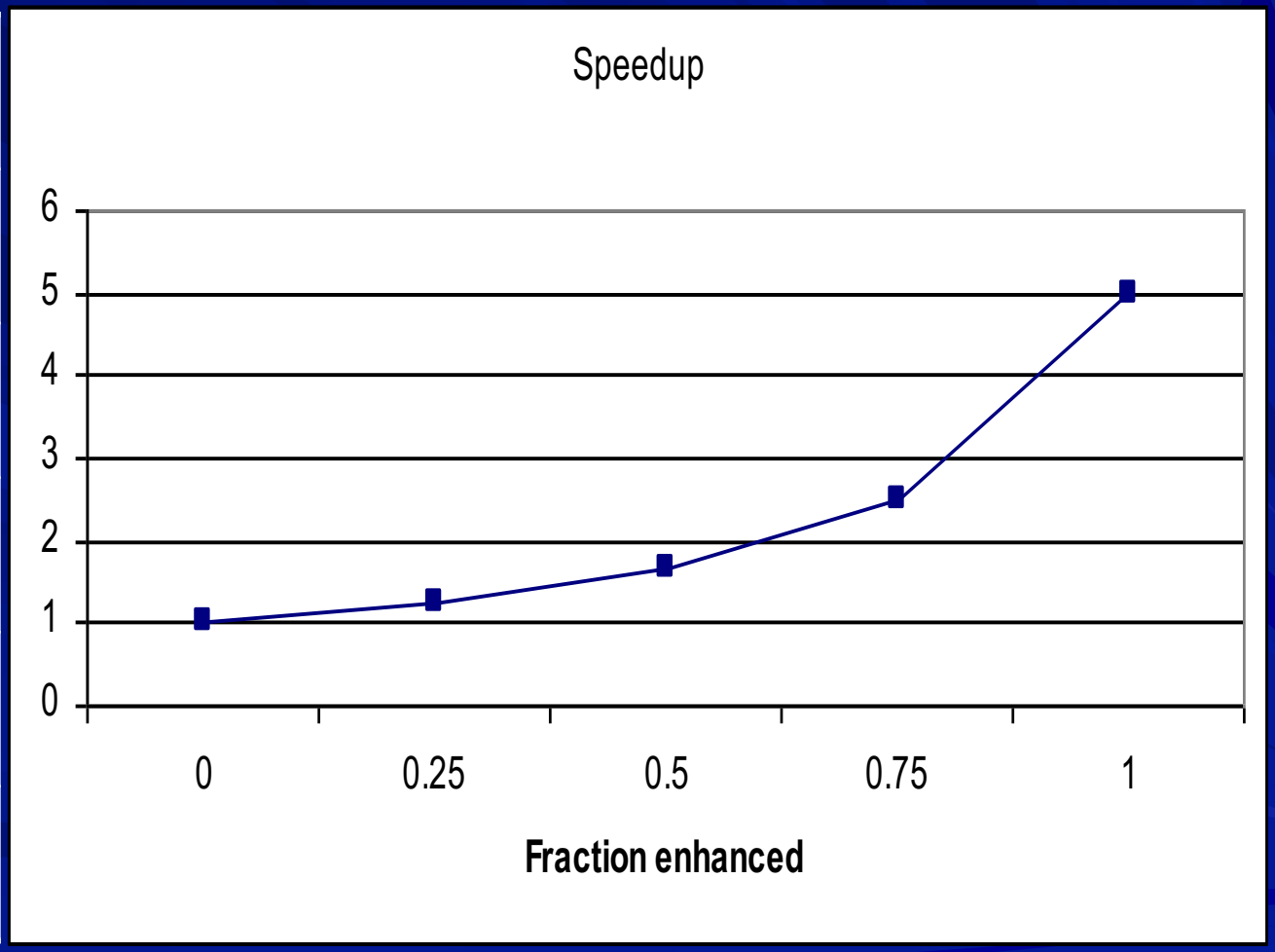
$1 = 3 (1 – 4 f / 5)$

$f = 5 / 6$

$100 * 5/6 = 83.3$

Example continued

$Frac_{enh}$	Speedup
0.00	1.00
0.25	1.25
0.50	1.67
0.75	2.50
1.00	5.00



Example: FP h/w vs s/w

C Clock freq 1000 MHz

	Instr mix in prog P	Cycles in MFP	Int instr in MNFP
FP multiply	10%	6	30
FP add	15%	4	20
FP divide	5%	20	50
Int instructions	70%	2	
CPI			

10% * 6 + 15% * 4 + 5% * 20 + 70% * 2
= 0.6 + 0.6 + 1.0 + 1.4 = 3.6

Example: FP h/w vs s/w

C Clock freq 1000 MHz

	Instr mix in prog P	Cycles in MFP	Int instr in MNFP
FP multiply	10%	6	30
FP add	15%	4	20
FP divide	5%	20	50
Int instructions	70%	2	
CPI		3.6	2.0
MIPS = C/CPI		277.8	500.0
N		300M	

300M(10% * 30 + 15% * 20 + 5% * 50 + 70% * 1)
= 300M(3 + 3 + 2.5 + 0.7) = 300M * 9.2 = 2760M

Example: FP h/w vs s/w

C Clock freq 1000 MHz

	Instr mix in prog P	Cycles in MFP	Int instr in MNFP
FP multiply	10%	6	30
FP add	15%	4	20
FP divide	5%	20	50
Int instructions	70%	2	
CPI		3.6	2.0
MIPS = C/CPI		277.8	500.0
N		300M	2760M
ExTime = N*CPI/C		1.08	5.52
MFLOPS = 30%*300M/ExTime		90/1.08	90/5.52

Thanks