CMSC 611: Advanced Computer Architecture

Performance

Some material adapted from Mohamed Younis, UMBC CMSC 611 Spr 2003 course slides

Important Equations (so far)

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

$$\text{Speedup} = \frac{\text{Performance (B)}}{\text{Performance (A)}} = \frac{\text{Time (A)}}{\text{Time (B)}}$$

$$\text{CPU time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

$$\text{CPU clock cycles} = \sum_{i=1}^{n} \text{CPI}_i \times \text{Instructions}_i$$

Amdahl's Law

The performance enhancement possible with a given improvement is limited by the amount that the improved feature is used

$$\text{Execution time after improvement} = \frac{\text{Execution time affected by the improvement}}{\text{Amount of improvement}} + \text{Execution time unaffected}$$

- A common theme in Hardware design is to *make the common case fast*
- Increasing the clock rate would not affect memory access time
- Using a floating point processing unit does not speed integer ALU operations

Example: Floating point instructions improved to run 2X; but only 10% of actual instructions are floating point

$$Exec\text{-}Time_{new} = Exec\text{-}Time_{old} \times (0.9 + .1/2) = 0.95 \times Exec\text{-}Time_{old}$$

$$Speedup_{overall} = Exec\text{-}Time_{new} \,/\, Exec\text{-}Time_{old} = 1/0.95 = 1.053$$

Ahmdal's Law for Speedup

$$Time_{old} = Time_{old} * \left(Fraction_{unchanged} + Fraction_{enhanced}\right)$$

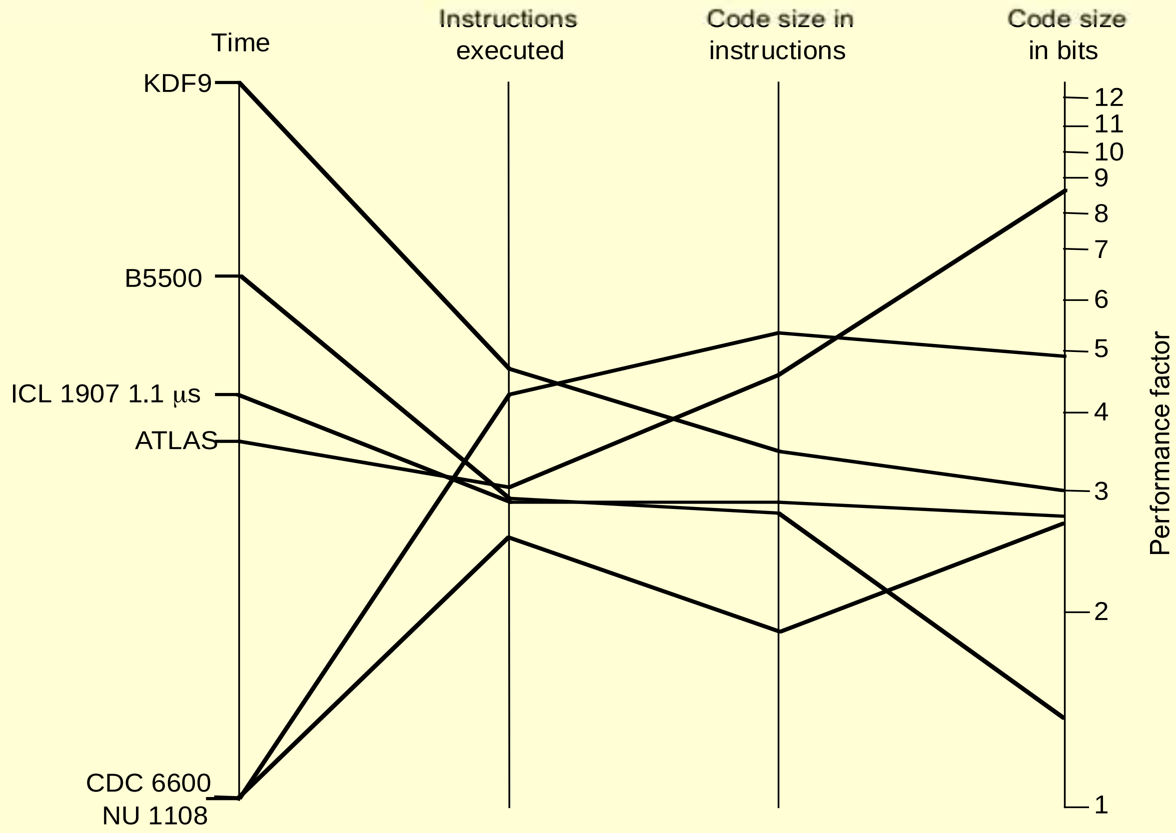
$$Time_{new} = Time_{old} * \left(Fraction_{unchanged} + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}\right)$$

$$Speedup_{overall} = \frac{Time_{old}}{Time_{new}} = \frac{Time_{old}}{Time_{old} * \left(Fraction_{unchanged} + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}\right)}$$

$$= \frac{1}{Fraction_{unchanged} + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$$

$$Speedup_{overall} = \frac{1}{\left(1 - Fraction_{enhanced}\right) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$$

Can Hardware-Independent Metrics Predict Performance?



- The Burroughs B5500 machine is designed specifically for Algol 60 programs
- Although CDC 6600's programs are over 3 times as big as those of B5500, yet the CDC machine runs them almost 6 times faster
- Code size cannot be used as an indication for performance

Comparing & Summarizing Performance

	Computer A	Computer B
Program 1 (seconds)	1	10
Program 2 (seconds)	1000	100
Total time (seconds)	1001	110

- Wrong summary can present a confusing picture
  - A is 10 times faster than B for program 1
  - B is 10 times faster than A for program 2
- Total execution time is a consistent summary measure
- Relative execution times for the same workload
  - Assuming that programs 1 and 2 are executing for the same number of times on computers A and B

$$\frac{\text{CPU Performance (B)}}{\text{CPU Performance (A)}} = \frac{\text{Total execution time (A)}}{\text{Total execution time (B)}} = \frac{1001}{110} = 9.1$$

Execution time is the only valid and unimpeachable measure of performance

Performance Summary (Cont.)

$$\text{Arithmetic Mean (AM)} = \frac{1}{n}\sum_{i=1}^{n}\text{Execution\_Time}_i$$

$$\text{Weighted Arithmetic Mean (WAM)} = \sum_{i=1}^{n} w_i \times \text{Execution\_Time}_i$$

Where: n is the number of programs executed

w_i is a weighting factor that indicates the frequency of executing program i

with $\sum_{i=1}^{n} w_i = 1$ *and* $0 \le w_i \le 1$

- Weighted arithmetic means summarize performance while tracking exec. time
- Never use AM for normalizing time relative to a reference machine

	Time on A	Time on B	Norm. to A		Norm. to B	
			A	B	A	B
Program 1	1	10	1	10	0.1	1
Program 2	1000	100	1	0.1	10	1
AM of normalized time			1	5.05	5.05	1
AM of time	500.5	55	1	0.11	9.1	1

Performance Summary (Cont.)

$$\text{Geometric Mean (GM)} = \sqrt[n]{\prod_{i=1}^{n} \text{Execution\_Time\_ratio}_i}$$

Where: *n* is the number of programs executed

With $$\frac{\text{Geometric Mean }(X_i)}{\text{Geometric Mean }(Y_i)} = \text{Geometric Mean}\left(\frac{X_i}{Y_i}\right)$$

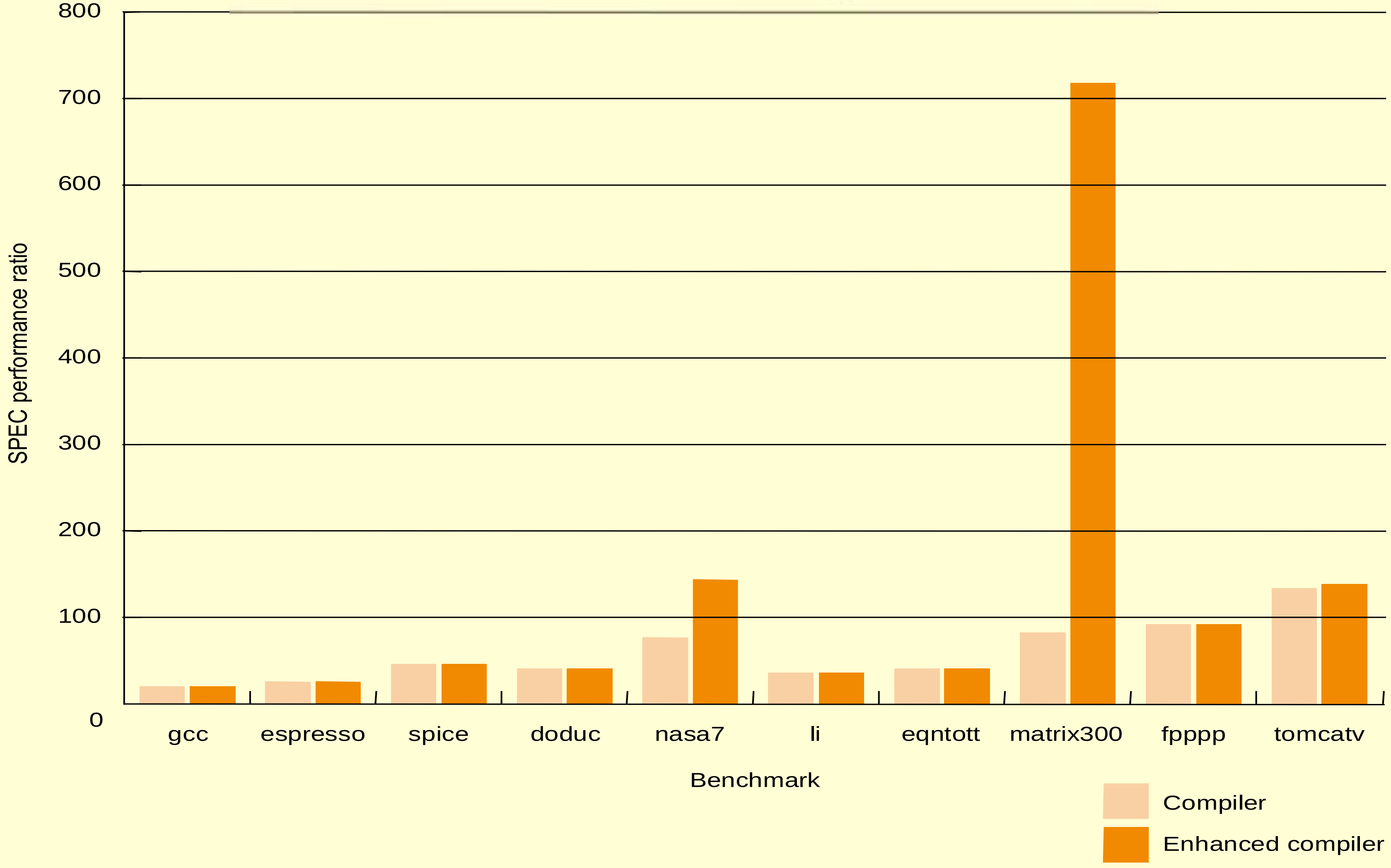
➨ Geometric mean is suitable for reporting average normalized execution time

	Time on A	Time on B	Norm. to A		Norm. to B	
			A	B	A	B
Program 1	1	10	1	10	0.1	1
Program 2	1000	100	1	0.1	10	1
GM of time or normalized time	31.62	31.62	1	1	1	1

Performance Benchmarks

- Many widely-used benchmarks are small programs that have significant locality of instruction and data reference
- Universal benchmarks can be misleading since hardware and compiler vendors do optimize their design for these programs
- The best types of benchmarks are real applications since they reflect the end-user interest
- Architectures might perform well for some applications and poorly for others
- Compilation can boost performance by taking advantage of architecture-specific features
- Application-specific compiler optimization are becoming more popular

Effect of Compilation



App. and arch. specific optimization can dramatically impact performance

The SPEC Benchmarks

- SPEC stands for System Performance Evaluation Cooperative suite of benchmarks
  - Created by a set of companies to improve the measurement and reporting of CPU performance
- SPEC2000 is the latest suite that consists of 12 integer (written in C) and 14 floating-point (in Fortran 77) programs
  - Customized SPEC suites have been recently introduced to assess performance of graphics and transaction systems.
- Since SPEC requires running applications on real hardware, the memory system has a significant effect on performance

Performance Reports

Hardware	
Model number	Powerstation 550
CPU	41.67-MHz POWER 4164
FPU (floating point)	Integrated
Number of CPU	1
Cache size per CPU	64K data/8k instruction
Memory	64 MB
Disk subsystem	2 400-MB SCSI
Network interface	N/A
Software	
OS type and revision	AIX Ver. 3.1.5
Compiler revision	AIX XL C/6000 Ver. 1.1.5 AIX XL Fortran Ver. 2.2
Other software	None
File system type	AIX
Firmware level	N/A
System	
Tuning parameters	None
Background load	None
System state	Multi-user (single-user login)

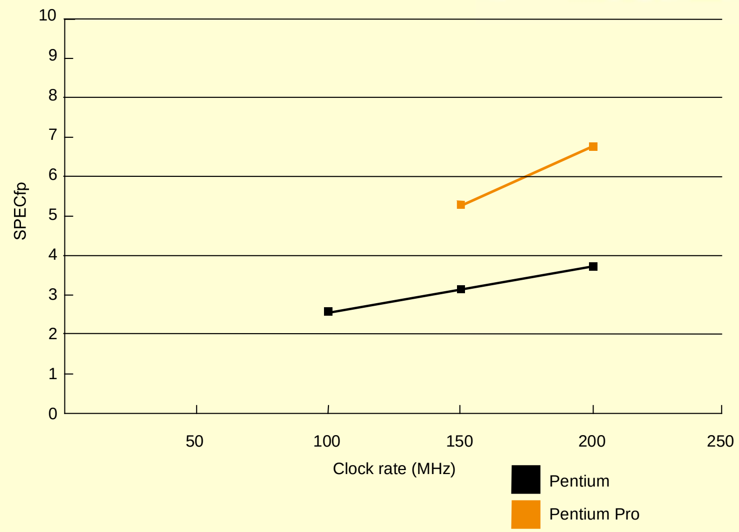
Guiding principle is *reproducibility* (report environment & experiments setup)

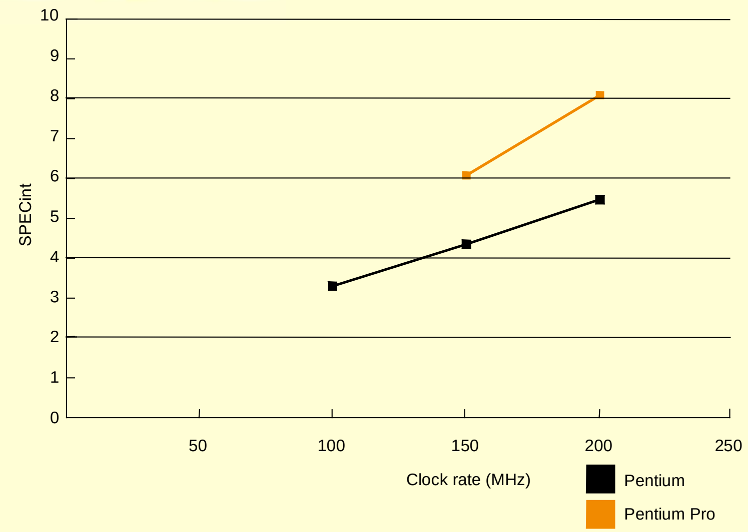
The SPEC Benchmarks

$$\text{SPEC ratio} = \frac{\text{Execution time on SUN SPARCstation 10/40}}{\text{Execution time on the measure machine}}$$

- Bigger numeric values of the SPEC ratio indicate faster machine

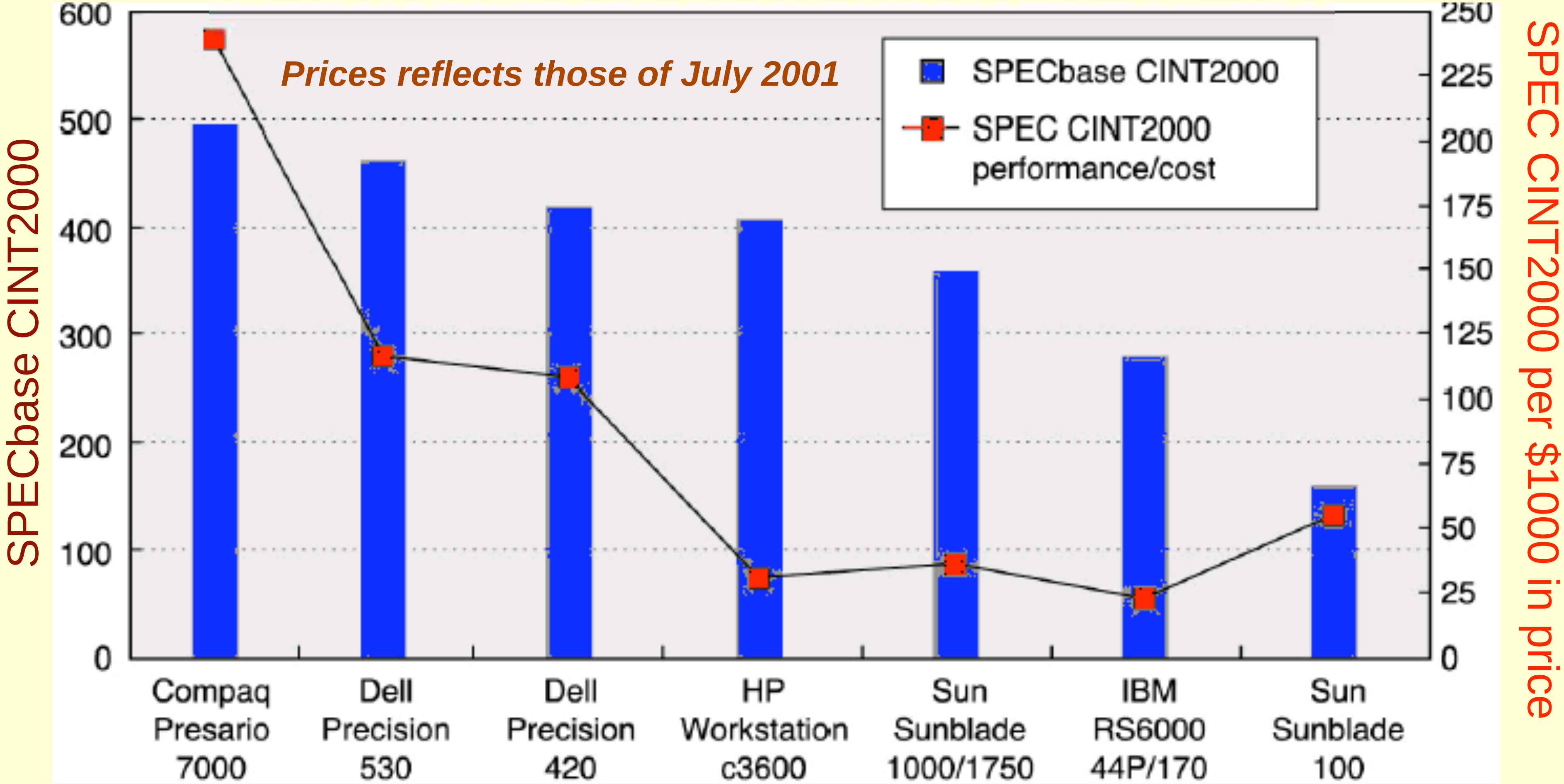
SPEC95 for Pentium and Pentium Pro





- The performance measured may be different on other Pentium-based hardware with different memory system and using different compilers
  - At the same clock rate, the SPECint95 measure shows that Pentium Pro is 1.4-1.5 times faster while the SPECfp95 shows that it is 1.7-1.8 times faster
  - When the clock rate is increased by a certain factor, the processor performance increases by a lower factor

Price-Performance Metric



- Different results are obtained for other benchmarks, e.g. SPEC CFP2000
- With the exception of the Sunblade price-performance metrics were consistent with performance

Historic Perspective

- In early computers most instructions of a machine took the same execution time
  - The measure of performance for old machines was the time required performing an individual operation (e.g. addition)
- New computers have diverse set of instructions with different execution times
  - The relative frequency of instructions across many programs was calculated
  - The average instruction execution time was measured by multiplying the time of each instruction by its frequency
- The average instruction execution time was a small step to MIPS that grew in popularity

Using MIPS

- MIPS = Million of Instructions Per Second
  - one of the simplest metrics
  - valid only in a limited context

$$\text{MIPS (native MIPS)} = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6}$$

- There are three problems with MIPS:
  - MIPS specifies the instruction execution rate but not the capabilities of the instructions
  - MIPS varies between programs on the same computer
  - MIPS can vary inversely with performance (see next example)

The use of MIPS is simple and intuitive, faster machines have bigger MIPS

Example

Consider the machine with the following three instruction classes and CPI:

Instruction class	CPI for this instruction class
A	1
B	2
C	3

Now suppose we measure the code for the same program from two different compilers and obtain the following data:

Code from	Instruction count in (billions) for each instruction class		
	A	B	C
Compiler 1	5	1	1
Compiler 2	10	1	1

Assume that the machine's clock rate is 500 MHz. Which code sequence will execute faster according to MIPS? According to execution time?

Answer:

Using the formula: $\text{CPU clock cycles} = \sum_{i=1}^{n} CPI_i \times C_i$

Sequence 1: CPU clock cycles = $(5 \times 1 + 1 \times 2 + 1 \times 3) \times 10^9 = 10 \times 10^9$ cycles

Sequence 2: CPU clock cycles = $(10 \times 1 + 1 \times 2 + 1 \times 3) \times 10^9 = 15 \times 10^9$ cycles

Example (Cont.)

Using the formula: $\text{Exection time} = \dfrac{\text{CPU clock cycles}}{\text{Clock rate}}$

Sequence 1: Execution time = $(10\times10^9)/(500\times10^6)$ = 20 seconds
Sequence 2: Execution time = $(15\times10^9)/(500\times10^6)$ = 30 seconds

Therefore compiler 1 generates a faster program

Using the formula: $\text{MIPS} = \dfrac{\text{Instruction count}}{\text{Execution time} \times 10^6}$

Sequence 1: $\text{MIPS} = \dfrac{(5+1+1)\times10^9}{20\times10^6} = 350$

Sequence 2: $\text{MIPS} = \dfrac{(10+1+1)\times10^9}{30\times10^6} = 400$

Although compiler 2 has a higher MIPS rating, the code from generated by compiler 1 runs faster