Performance Metrics

- 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 is the average for a program
  - and yields the “*native MIPS*”

Performance Metrics

- MIPS can be misleading
  - Varies between programs on the same machine
  - Does not reflect the instruction set complexity
  - May vary inversely with performance

- The following example illustrates this last point:

MIPS Example

Assume a program is translated by two different compilers for the same machine:

$$\text{MIPS} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

	Instruction counts (in millions) for each instruction class		
Instruction Class	A	B	C
Code from compiler 1	4	2	1
Code from compiler 2	9	1	1

$CPI_1 = ((4*1 + 2*2 + 1*3)*10^6) / ((4+2+1)*10^6) = 1.57$

Thus $MIPS_1 = 100\text{ MHz} / 1.57*10^6 = 63.64$

CPU $time_1 = ((4+2+1)*10^6 * 1.57) / 100 * 10^6 = 0.11$ sec

MIPS Example

Assume a program is translated by two different compilers for the same machine:

	Instruction counts (in millions) for each instruction class		
Instruction Class	A	B	C
Code from compiler 1	4	2	1
Code from compiler 2	9	1	1

CPI_2 = ((9*1 + 1*2 + 1*3)*10^6) / ((9+1+1)*10^6) = 1.27

Thus $MIPS_2$ = 100 MHz / 1.27*10^6 = 78.57

CPU $time_2$ = ((9+1+1)*10^6* 1.27) /100 * 10^6 = 0.14 sec

Yields a higher MIPS rating but takes longer to execute

Performance Metrics

- *Native MIPS* was defined above
- *Peak MIPS* is based on minimum CPI
  - Assumes all instructions in program have minimum CPI
  - Minimum possible CPI = 1
  - All instructions require at least 1 clock cycle
- Relative MIPS is based on a standard machine

$$MIPS_{relative} = \frac{Time_{reference}}{Time_{unrated}} \times MIPS_{reference}$$

Bench Marks

- *Benchmarks are suites of programs*
  - Chosen to be typical of workloads
  - Systems are rated based on benchmark execution times
  - Average time is used as a measure of performance
- Arithmetic mean is not used as average
  - Outliers can have undue influence on result
  - Geometric mean is used instead (defined below)

Performance Metrics

Geometric Mean → $\sqrt[n]{\prod_{i=1}^{n} \text{Execution time}}$

- Standard Performance Evaluation Corp (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)

Performance Metrics

- SPEC CPU2017
  - Comprised of a group of 43 programs
  - Organized into 4 suites

- Another possibility is the Harmonic Mean
  - More appropriate for averaging rates such as MIPS
  - Average MIPS for suite gives a single performance measure

Harmonic Mean → $\overline{S_H} = \frac{N}{\sum_{1}^{N} \frac{1}{S_i}}$

Amdahl's Law

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

$$T_{\text{improved}} = \frac{T_{\text{affected}}}{\text{improvement factor}} + T_{\text{unaffected}}$$

- Example: multiply accounts for 80s/100s
  - How much must multiply be improved to get 5× the overall performance?

$$20 = \frac{80}{n} + 20$$

  - Can't be done!
- Corollary: make the common case fast

Floating Point Performance

MIPS measures integer performance

Engineering Applications employ floating point

- So the appropriate metric is Mega-flops (MFLOPS)
- Floating point support varies even more
- trig and exponentiation functions may be available
- In other cases, only simple arithmetic is supported

MFLOPS can be very unreliable in predicting performance

It is best to use execution time to assess performance