Computer Architecture

Lecture-03

Md. Biplob Hosen

Lecturer, IIT-JU.

Email: biplob.hosen@juniv.edu

Reference

- “Computer Organization and Architecture” by William Stallings; 8th Edition (Chapter-02).
  - Any later edition is fine.

PERFORMANCE ASSESSMENT

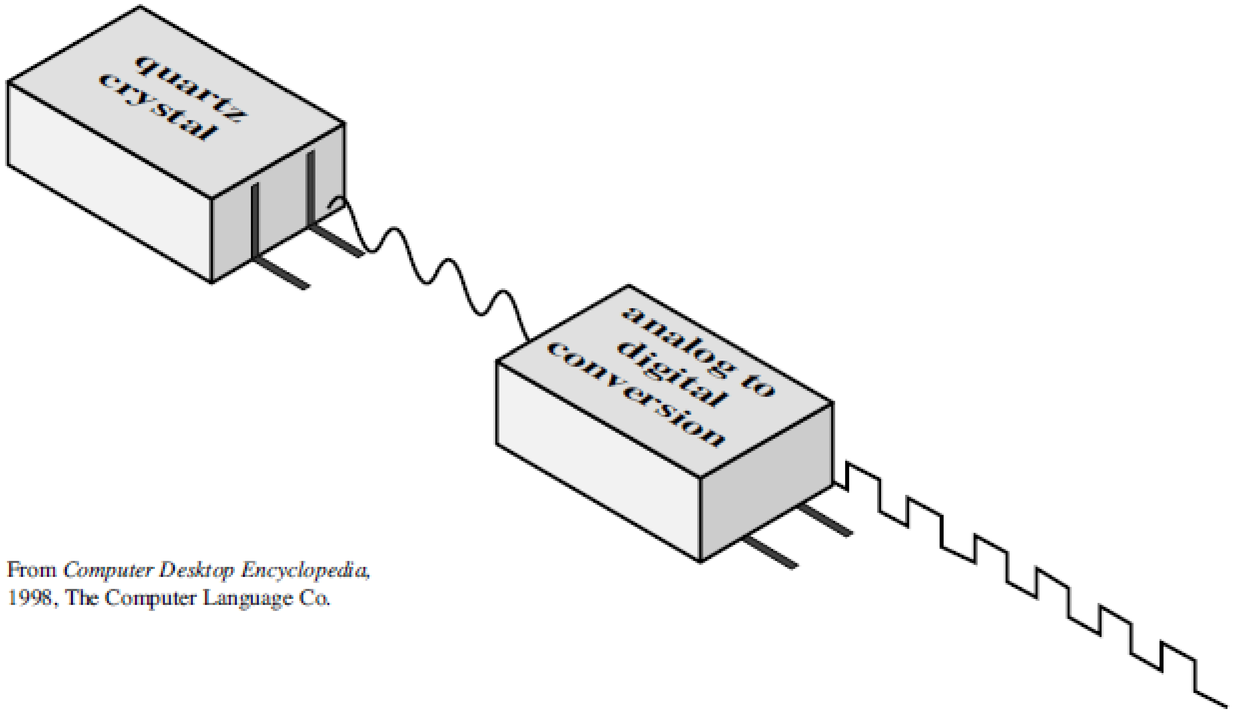
- In evaluating processor hardware and setting requirements for new systems, following parameters are important.
  - Performance (key parameter)
  - Cost
  - Size
  - Security
  - Reliability
  - Power consumption
- Difficult to make meaningful performance comparisons among different processors.
  - Raw processor speed is not sufficient to say how a processor will perform while executing a given application.
  - Application speed depends on instruction set, implementation language, compiler efficiency and programming skill along with raw processor speed.

Clock Speed and Instructions per Second

THE SYSTEM CLOCK

- Operations performed by a processor, such as fetching an instruction, decoding the instruction, performing an arithmetic operation, and so on, are governed by a system clock.
- Typically, all operations begin with the pulse of the clock.
- The speed of a processor is dictated by the pulse frequency produced by the clock, measured in cycles per second, or Hertz (Hz).
- Typically, clock signals are generated by a quartz crystal, which generates a constant signal wave while power is applied.
- This wave is converted into a digital voltage pulse stream that is provided in a constant flow to the processor circuitry.
- For example, a 1-GHz processor receives 1 billion pulses per second.
- The rate of pulses is known as the clock rate, or clock speed.
- One increment, or pulse, of the clock is referred to as a clock cycle, or a clock tick.
  - The time between pulses is the cycle time.

Continue…



From *Computer Desktop Encyclopedia,*
1998, The Computer Language Co.

Continue…

- The execution of an instruction involves a number of discrete steps, such as-
  - fetching the instruction from memory, decoding the various portions of the instruction, loading and storing data, and performing arithmetic and logical operations.
- Thus, most instructions on most processors require multiple clock cycles to complete.
- Some instructions may take only a few cycles, while others require dozens.
  - A straight comparison of clock speeds on different processors does not tell the whole story about performance.

Continue…

INSTRUCTION EXECUTION RATE

- A processor is driven by a clock with a constant frequency f or, equivalently, a constant cycle time τ, where $\tau = 1/f$.
- Define the instruction count, I_c, for a program as the number of machine instructions executed for that program until it runs to completion or for some defined time interval.
- An important parameter is the average cycles per instruction CPI for a program.
- If all instructions required the same number of clock cycles, then CPI would be a constant value for a processor.
- On any given processor, the number of clock cycles required varies for different types of instructions, such as load, store, branch, and so on.

Continue…

- Let CPI_i be the number of cycles required for instruction type i and I_i be the number of executed instructions of type i for a given program.
- Then we can calculate an overall CPI as follows:

$$CPI = \frac{\sum_{i=1}^{n} CPI_i \times I_i}{I_c}$$

Calculate CPI.

Instruction Type	CPI	Instruction Mix
Arithmetic and logic	1	60%
Load/store with cache hit	2	18%
Branch	4	12%
Memory reference with cache miss	8	10%

Continue…

- The processor time T needed to execute a given program can be expressed as:

$$T = I_c \times CPI \times \tau$$

- **Calculate processor time for previous example [with 100 MHz processor].**
- We can refine this formulation by recognizing that during the execution of an instruction, part of the work is done by the processor, and part of the time a word is being transferred to or from memory.
- In this latter case, the time to transfer depends on the memory cycle time, which may be greater than the processor cycle time.
- We can rewrite the preceding equation as :

$$T = I_C \times \left[p + (m \times k)\right] \times \tau$$

- where p is the number of processor cycles needed to decode and execute the instruction, m is the number of memory references needed, and k is the ratio between memory cycle time and processor cycle time.
- The five performance factors in the preceding equation (I_c, p, m, k, τ) are influenced by four system attributes: the design of the instruction set (known as instruction set architecture), compiler technology (how effective the compiler is in producing an efficient machine language program from a high-level language program), processor implementation, and cache & memory hierarchy.

Continue…

- A common measure of performance for a processor is the rate at which instructions are executed, expressed as millions of instructions per second (MIPS), referred to as the MIPS rate.
- We can express the MIPS rate in terms of the clock rate and CPI as follows:

$$MIPS\_rate = \frac{I_C}{T \times 10^6} = \frac{f}{CPI * 10^6}$$

- Consider the execution of a program which results in the execution of 2 million instructions on a 400-MHz processor.
- **Calculate MIPS_rate.**

Continue…

Instruction Type	CPI	Instruction Mix
Arithmetic and logic	1	60%
Load/store with cache hit	2	18%
Branch	4	12%
Memory reference with cache miss	8	10%

- The average CPI when the program is executed on a uniprocessor with the above trace results is CPI = $0.6 + (2 \times 0.18) + (4 \times 0.12) + (8 \times 0.1) = 2.24$.
- The corresponding MIPS rate is $(400 \times 10^6) / (2.24 \times 10^6) = 178$.

Continue…

- Another common performance measure deals only with floating-point instructions.
- These are common in many scientific and game applications.
- Floating point performance is expressed as millions of floating-point operations per second (MFLOPS), defined as follows:

$$\text{MFLOPS rate} = \frac{\textit{Number of executed floating-point operations in a program}}{\textit{Execution time} \times 10^6}$$

Benchmarks

- Measures such as MIPS and MFLOPS have proven inadequate to evaluating the performance of processors.
- Because of differences in instruction sets, the instruction execution rate is not a valid means of comparing the performance of different/ architectures.
- RISC and CISC machines may not be rated with same MIPS rating but may do same amount of work at the same time.
- A = B + C; //High-level language statement
- Add mem (B), mem (C), mem (A) //one instruction for CISC computer
- For RISC computer, it will be executed with four instructions, but possibly with same time.
  - Load mem (B), reg (1);
  - Load mem (C), reg (2);
  - Add reg (1), reg (2), reg (3);
  - Store reg (3), reg (A)
- Moreover, the performance of a given processor on a given program may not be useful in determining how that processor will perform on a very different type of application.

Benchmarks

- In early 1990s, interest shifted to measuring performance of systems using a set of benchmark programs.
- The same set of programs can be run to different machines and the execution times are compared.
- Characteristics of a benchmark program:
  - Written in high level language.
  - Representative of particular programming style, such as system programming, numerical programming, or commercial programming.
  - Can be measured easily.
  - Has wide distribution.

Benchmarks

SPEC Benchmarks

- Standardized benchmark suite are developed to measure and compare computer performance.
- Benchmark suite is a collection of programs, defined in high-level language, that attempt together to provide a representative test of a computer in a particular application or system programming area.
- System Performance Evaluation Corporation (SPEC) defines and maintains best known collection of benchmark suite.
- SPEC performance measurements are widely used for comparison.
- SPEC CPU2006 is a well-known benchmark suite.
- It consists of 17 floating-point programs written in C, C++ & Fortran, and 12 integer program written in C & C++.
- This suite contains over 3 million lines of code.
- SPECjvm98, SPECjbb2000, SPECweb99, SPECmail2001 are some SPEC suites.

Benchmarks

AVERAGING RESULTS

- To obtain a reliable comparison of the performance of various computers, it is preferable to run a number of different benchmark programs on each machine and then average the results.
- For example, if *m* different benchmark program, then a simple arithmetic mean can be calculated as follows:

$$R_A = \frac{1}{m}\sum_{i=1}^{m} R_i$$

- where R_i is the high-level language instruction execution rate for the ith benchmark program.
- An alternative is to take the harmonic mean.

Benchmarks

- SPEC benchmarks concern themselves with two fundamental metrics : **a speed metric** and a **rate metric**.
- The **speed metric** measures the ability of a computer to complete a single task.
- SPEC defines a base runtime for each benchmark program using a reference machine.
- Results for a system under test are reported as the ratio of the reference run time to the system run time.
- The ratio is calculated as follows:

$$r_i = \frac{Tref_i}{Tsut_i}$$

- Where, $Tref_i$ is the execution time of benchmark program i on the reference system and $Tsut_i$ is the execution time of benchmark program i on the system under test.

Benchmarks

- The larger the ratio, the higher the speed.
- An overall performance measure for the system under test is calculated by averaging the values for the ratios for all 12 integer benchmarks.
- SPEC specifies the use of a geometric mean, defined as follows:

$$r_G = \left(\prod_{i=1}^{n} r_i \right)^{1/n}$$

- where r_i is the ratio for the i^{th} benchmark program.
- The speed metric is calculated by taking the twelfth root of the product of the ratios.

Benchmarks

- The **rate metric** measures the throughput or rate of a machine carrying out a number of tasks.
- For the rate metrics, multiple copies of the benchmarks are run simultaneously.
- Typically, the number of copies is the same as the number of processors on the machine.
- Again, a ratio is used to report results, although the calculation is more complex
- The ratio is calculated as follows:

$$r_i = \frac{N \times Tref_i}{Tsut_i}$$

- where $Tref_i$ is the reference execution time for benchmark i, N is the number of copies of the program that are run simultaneously, and $Tsut_i$ is the elapsed time from the start of the execution of the program on all N processors of the system under test until the completion of all the copies of the program.
- A geometric mean is calculated to determine the overall performance measure.

Amdahl’s Law

- First proposed by Gene Amdahl.
- Deals with the potential speedup of a program using multiple processors compared to a single processor.
- Consider a program running on a single processor such that a fraction *(1 - f)* of the execution time involves code that is inherently serial and a fraction f that involves code that is infinitely parallelizable with no scheduling overhead.
- Let *T* be the total execution time of the program using a single processor.
- Then the speedup using a parallel processor with *N* processors that fully exploits the parallel portion of the program is as follows:

$$\text{Speedup} = \frac{\text{time to execute program on a single processor}}{\text{time to execute program on N parallel processors}}$$

Continue…

$$= \frac{T(1-f)+Tf}{T(1-f)+\frac{Tf}{n}} \qquad = \frac{1}{(1-f)+\frac{f}{N}}$$

- Two important conclusions can be drawn:
  1. When f is small, the use of parallel processors has little effect.
  2. As N approaches infinity, speedup is bound by 1/(1 - f), so that there are diminishing returns for using more processors.

Continue…

- Amdahl’s law can be generalized to evaluate any design or technical improvement in a computer system.
- Consider any enhancement to a feature of a system that results in a speedup.
- The speedup can be expressed as:

$$\text{Speedup} = \frac{\text{Performance after enhancement}}{\text{Performance before enhancement}}$$

Continue…

$$\text{Speedup} = \frac{\text{Execution time before enhancement}}{\text{Execution time after enhancement}}$$

- Suppose that a feature of the system is used during execution a fraction of the time f, before enhancement. The speedup of the feature after enhancement is SU_f. Then the overall speedup of the system is:

  - Speedup = $\dfrac{1}{(1-f) + \dfrac{f}{SU_f}}$

Continue…

- Suppose that a task makes extensive use of floating-point operations, with 40% of the time is consumed by floating-point operations. With a new hardware design, the floating-point module is speeded up by a factor of K.
- What is the overall speedup?
- What is the maximum speedup?

  - Speedup = $\frac{1}{0.6 + \frac{0.4}{\mathrm{k}}}$
  - Maximum Speedup = 1.67 (How is it possible?)

Assignment

- Solve the exercise problems: 2.10, 2.11, 2.12, 2.13, 2.14, 2.15, 2.16 & 2.17, and submit the scanned copy of the solution within 10/10/2021 in the Google classroom.

Thank You ☺