Performance of a Processor - Instruction Execution Rate

- Processor's *throughput*—the amount of work the CPU completes in a given period of time.
- A processor is driven by a clock with a constant frequency f or, equivalently, a constant cycle time τ , where $\tau = 1/f$.
- Instruction count – I_c – The number of machine instructions executed for that program until it runs to completion or for some defined time interval
- CPI- 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.
- However, on any give processor, the number of clock cycles required varies for different types of instructions, such as load, store, branch, and so on.

Instruction Execution Rate

- i – Instruction type
- CPI_i - The number of cycles required
- I_i – The number of executed instructions of type i for a given program.
- Then we can calculate an overall CPI as follows

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

Processor Time

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

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

- 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

$$T = I_c \times [p + (m \times k)] \times \tau$$

- Where,
- p – the number of processor cycles needed to decode and execute the instruction,
- m – the number of memory references needed
- k – The ratio between memory cycle time and processor cycle time

MIPS

- 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:

$$\text{MIPS rate} = \frac{I_c}{T \times 10^6} = \frac{f}{CPI \times 10^6}$$

MIPS Example

- Consider the execution of a program which results in the execution of 2 million instructions on a 400-MHz processor.
- The instruction mix and the CPI for each instruction type are given below based on the result of a program trace experiment:

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%

$$CPI = 0.6 + (2 \times 0.18) + (4 \times 0.12) + (8 \times 0.1) = 2.24.$$

$$\text{MIPS rate is } (400 \times 10^6)/(2.24 \times 10^6) \approx 178.$$

MFLOPS

- 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 evaluate 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.
- For example, consider this high-level language statement:

$$A = B + C$$

- With the traditional instruction set architecture, referred to as a complex instruction set computer (CISC), this instruction can be compiled into one processor instruction:

add mem(B), mem(C), mem (A)

Benchmarks

- On a typical RISC machine, the compilation would look something like this:

  load mem(B), reg(1);

  load mem(C), reg(2);

  add reg(1), reg(2), reg(3);

  store reg(3), mem (A)

- Because of the nature of the RISC architecture, both machines may execute the original high-level language instruction in about the same time.
- If this example is representative of the two machines, then if the CISC machine is rated at 1 MIPS, the RISC machine would be rated at 4 MIPS.
- But both do the same amount of high-level language work in the same amount of time.

Benchmarks

- Measure the performance of systems using a set of benchmark programs.
- The same set of programs can be run on different machines and the execution times compared.
- Collection of benchmark suites is defined and maintained by the System Performance Evaluation Corporation (SPEC), an industry consortium.
- Ex: SPECCPU2006, SPECjvm98, SPECweb99

Amdahl's Law

- Amdahl's law was first proposed by Gene Amdahl in [AMDA67] and 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.

Speedup

- Then the speedup using a parallel processor with N processors that fully exploits the parallel portion of the program is as follows:

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

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

Speedup

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

Speedup

- Consider any enhancement to a feature of a system that results in a speedup.
- The speedup can be expressed as

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

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

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

Speedup – Example

- 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.
- Then the overall speedup is:

$$\text{Speedup} = \frac{1}{0.6 + \frac{0.4}{K}}$$

- Thus, independent of K, the maximum speedup is 1.67.

References

- Text Book – William Stallings, “Computer Organization and Architecture, Designing for performance”, 8th edition, Prentice Hall.
- Internet Sources
- https://www.coursehero.com/file/pna2ec/D3-resides-in-bit-position-6-Dr-VSaritha-Associate-Professor-SCOPE-VIT/