COM 314 1st NOTE

Peter Abtsham

COM 314 – COMPUTER ARCHITECTURE

1.0 INTRODUCTION TO COMPUTER ARCHITECTURE

1.1 INTRODUCTION TO COMPUTER SYSTEMS

Computer systems have conventionally been defined through their interfaces at a number of layered abstraction levels, each providing functional support to its predecessor. Included among the levels are the application programs, the high-level languages, and the set of machine instructions. Based on the interface between different levels of the system, a number of computer architectures can be defined.
The interface between the application programs and a high-level language is referred to as a language architecture. A different definition of computer architecture is built on four basic viewpoints. These are the structure, the organization, the implementation, and the performance. In this definition, the structure defines the interconnection of various hardware components, the organization defines the dynamic interplay and management of the various components, the implementation defines the detailed design of hardware components, and the performance specifies the behaviour of the computer system.

1.2 OVERVIEW OF COMPUTER ORGANISATION AND ARCHITECTURE

Computer architecture deals with the functional behaviour of a computer system as viewed by a programmer. This view includes aspects such as the sizes of data types (e.g. using 16 binary digits to represent an integer), and the types of operations that are supported (like addition, subtraction, and subroutine calls). Computer organization deals with structural relationships that are not visible to the programmer, such as interfaces to peripheral devices, the clock frequency, and the technology used for the memory.
Computer Architecture refers to those attributes of a system visible to a programmer or those attributes that have a direct impact on the logical execution of a program. Computer Organisation refers to the operational units and their interconnections that release the architectural specifications. Examples of architectural attributes includes the instruction set, the number of bits used to represent various data types (e.g. numbers, characters), I/O mechanisms and techniques for addressing memory. Organisational attributes include those hardware details transparent to the programmer, such as control signals, interfaces between the computer and peripherals, and the memory technology used.
As an example, it is an architectural design issue whether a computer will have a multiply instruction. It is an organisational issue whether that instruction will be implemented by a special multiply unit or by a mechanism that makes repeated use of the add unit of the system.

1.3 PERFORMANCE MEASURES

There are various facets to the performance of a computer. For example, a user of a computer measures its performance based on the time taken to execute a given job (program). On the other hand, a laboratory engineer measures the performance of his system by the total amount of work done in a given time. While the user considers the program execution time a measure for performance, the laboratory engineer considers the throughput a more important measure for performance
Performance analysis should help answering questions such as how fast can a program be executed using a given computer? In order to answer such a question, we need to determine the time taken by a computer to execute a given job. We define the clock cycle time as the time between two consecutive rising (trailing) edges of a periodic clock signal (Fig. 1.1). Clock cycles allow counting unit computations, because the storage of computation results is synchronized with rising (trailing) clock edges. The time required to execute a job by a computer is often expressed in terms of clock cycles.

execute a job can be expressed as

$$CPU\ time = CC \times CT = CC/f$$

It may be easier to count the number of instructions executed in a given program as compared to counting the number of CPU clock cycles needed for executing that program.

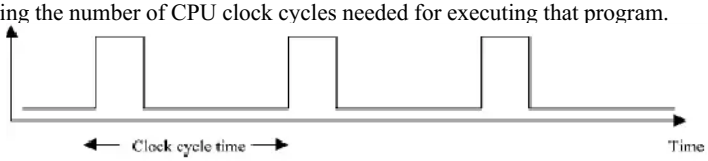

Figure 1.1 Clock signal

Therefore, the average number of clock cycles per instruction (CPI) has been used as an alternate performance measure. The following equation shows how to compute the CPI.

$$CPI = \frac{CPU\ clock\ cycles\ for\ the\ program}{Instruction\ count}$$

$$CPU\ time = Instruction\ count \times CPI \times Clock\ cycle\ time$$

$$= \frac{Instruction\ count \times CPI}{Clock\ rate}$$

It is known that the instruction set of a given machine consists of a number of instruction categories: ALU (simple assignment and arithmetic and logic instructions), *load*, *store*, *branch*, and so on. In the case that the CPI for each instruction category is known, the overall CPI can be computed as

$$\text{CPI} = \frac{\sum_{i=1}^{n} CPIi \times Ii}{Instruction\ count}$$

where Ii is the number of times an instruction of type i is executed in the program and CPIi is the average number of clock cycles needed to execute such instruction.

Example 1: Moore's law, which is attributed to Intel founder Gordon Moore, states that computing power doubles every 18 months for the same price. An unrelated observation is that floating point instructions are executed 100 times faster in hardware than via emulation. Using Moore's law as a guide, how long will it take for computing power to improve to the point that floating point instructions are emulated as quickly as their (earlier) hardware counterparts?

SOLUTION: Computing power increases by a factor of 2 every 18 months, which generalizes to a factor of 2^x every $18x$ months. If we want to figure the time at which computing power increases by a factor of 100, we need to solve $2^x = 100$, which reduces to $x = 6.644$. We thus have $18x = 18 \times (6.644\ \text{months}) = 120$ months, which is 10 years.

Example 2: Consider computing the overall CPI for a machine A for which the following performance measures were recorded when executing a set of benchmark programs. Assume that the clock rate of the CPU is 200 MHz.

Instruction Category	Percentage of occurrence	No. of cycles per instruction
ALU	38	1
Load & Store	15	3
Branch	42	4
Others	5	5

Assuming the execution of 100 instructions, the overall CPI can be computed as

$$\text{CPIa} = \frac{\sum_{i=1}^{n} CPIi \times Ii}{Instruction\ count} = \frac{38 \times 1 + 15 \times 3 + 42 \times 4 + 5 \times 5}{100} = 2.76$$

It should be noted that the CPI reflects the organization and the instruction set architecture of the processor while the instruction count reflects the instruction set architecture and compiler technology used.

A different performance measure that has been given a lot of attention in recent years is MIPS (million instructions-per-second (the rate of instruction execution per unit time)), which is defined as:

$$\text{MIPS} = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6} = \frac{\text{Clock Rate}}{\text{CPI} \times 10^6}$$

Example 3: Suppose that the same set of benchmark programs considered above were executed on another machine, call it machine B, for which the following measures were recorded.

Instruction Category	Percentage of occurrence	No. of cycles per instruction

Branch	15	3
Others	20	5

What is the MIPS rating for the machine considered in the previous example (machine A) and machine B assuming a clock rate of 200 MHz?

$$\text{CPI}^a = \frac{\sum_{i=1}^{n} CPIi \times Ii}{Instruction\ count} = \frac{38 \times 1 + 15 \times 3 + 42 \times 4 + 5 \times 5}{100} = 2.76$$

$$\text{MIPS}_a = \frac{\text{Clock Rate}}{\text{CPIa} \times 10^6} = \frac{200 \times 10^6}{2.76 \times 10^6} = 70.24$$

$$\text{CPI}_b = \frac{\sum_{i=1}^{n} CPIi \times Ii}{Instruction\ count} = \frac{35 \times 1 + 30 \times 2 + 20 \times 5 + 15 \times 3}{100} = 2.4$$

$$\text{MIPS}_b = \frac{\text{Clock Rate}}{\text{CPI}_b \times 10^6} = \frac{200 \times 10^6}{2.4 \times 10^6} = 83.67$$

Thus $\text{MIPS}_b > \text{MIPS}_a$.

It is interesting to note here that although MIPS has been used as a performance measure for machines, one has to be careful in using it to compare machines having different instruction sets. This is because MIPS do not track execution time. Consider, for example, the following measurement made on two different machines running a given set of benchmark programs. Assume that the clock rate is 200MHz.

Instruction Category	No. of instructions (in millions)	No. of cycles per instruction
Machine (A)		
ALU	8	1
Load & Store	4	3
Branch	2	4
Others	4	3
Machine (B)		
ALU	10	1
Load & Store	8	2
Branch	2	4
Others	4	3

$$\text{CPI}_a = \frac{\sum_{i=1}^{n} CPIi \times Ii}{Instruction\ count} = \frac{(8 \times 1 + 4 \times 3 + 2 \times 4 + 4 \times 3) \times 10^6}{(8+4+2+4) \times 10^6} = 2.2$$

$$\text{MIPS}_a = \frac{\text{Clock Rate}}{\text{CPI}_a \times 10^6} = \frac{200 \times 10^6}{2.2 \times 10^6} = 90.9$$

$$CPU\ time_a = \frac{Instruction\ count \times CPI_a}{\text{Clock rate}} = \frac{18 \times 10^6 \times 2.2}{200 \times 10^6} = 0.198s$$

$$\text{CPI}_b = \frac{\sum_{i=1}^{n} CPIi \times Ii}{Instruction\ count} = \frac{(10 \times 1 + 8 \times 2 + 2 \times 4 + 4 \times 3) \times 10^6}{(8+4+2+4) \times 10^6} = 1.9$$

$$\text{MIPS}_b = \frac{\text{Clock Rate}}{\text{CPI}_b \times 10^6} = \frac{200 \times 10^6}{1.9 \times 10^6} = 105.3$$

$$CPU\ time_b = \frac{Instruction\ count \times CPI_b}{\text{Clock rate}} = \frac{24 \times 10^6 \times 1.9}{200 \times 10^6} = 0.228s$$

$\text{MIPS}_b > \text{MIPS}_a$ and $\text{CPU}_b > \text{CPU}_a$

The example shows that although machine B has a higher MIPS compared to machine A, it requires longer CPU time to execute the same set of benchmark programs.

Million floating-point instructions per second, MFLOP (rate of floating-point instruction execution per unit time) has also been used as a measure for machines' performance. It is defined as

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

1.4 BASIC PROCESSOR ARCHITECTURE