UNIT 14: CISC AND RISC ARCHITECTURE

UNIT STRUCTURE

10.1 Learning Objectives
10.2 Introduction
10.3 Complex Instruction Set Computers (CISC)
    10.3.1 Performance Analysis of CISC
10.4 Reduced Instruction Set Computers (RISC)
    10.4.1 Performance Analysis of RISC
10.5 Let Us Sum Up
10.6 Further Readings
10.7 Answers to Check Your Progress
10.8 Model Questions

14.1 LEARNING OBJECTIVES

After going through this unit you will be able to:

- describe the basic concepts CISC architecture and its performance issues
- describe the basic concepts RISC architecture and its performance issues
- differentiate between these two types of architectures

14.2 INTRODUCTION

The development and evolution of high level languages has led to a semantic gap, between the kind of operations allowed by these high level languages and the kind of operations allowed by the computer architecture. To meet the requirements of complex high level languages, complex architectures were designed that offered larger number of basic operations. On the other hand, a reduced instruction set computer focuses on a simpler set of instructions and fewer basic operations that can be executed in one clock cycle.

In this unit, we will explore CISC and RISC architectures. We shall also discuss the problems associated with complex architectures and how reducing the complexity of the architectures may lead to improved performances.

14.3 COMPLEX INSTRUCTION SET COMPUTERS (CISC)

High level languages (HLL) often include complex operations to make it easier for the programmer to express the algorithms. There is a semantic gap between the complex operations provided by the HLLs and the operations provided by the architecture. The translation of these complex operations into the basic operations that the architecture provides is left to the compiler. As a result, the programs generated by the compilers are large and not efficient enough. To close this semantic gap, complex architectures with large instructions sets, several addressing modes, and more number of basic operations were designed to meet the requirements of the high level languages. Computers with such architectures are called Complex Instruction Set Computers (CISC).

Such complex instruction sets are expected to reduce the complexity of compilers and improve the efficiency of the code generated by the compilers. CISC architectures aim to complete any task in as few number of machine instructions as possible. To accomplish that, a large set of complex instructions are directly incorporated into hardware. Such a complex architecture also provides support for even more sophisticated HLLs.

The compilers in CISC architectures are much simpler and the machine code generated is also much smaller in size. As a result, another advantage to CISC architecture is that programs require lesser memory. Further, smaller programs have the advantage that the number of *instruction fetch* operations during the execution of the program will be much smaller. This is expected to reduce execution time of the program.

14.3.1 Performance Analysis of CISC

CISC architectures were originally designed to simplify compilers for high level language and to enable compilers to generate more concise machine

code. In an attempt to do so, complex operations are incorporated into the hardware. However, it has been noted that it is often difficult to exploit such a complex set of machine instructions. In order to use the complex instruction set properly the compiler has to look for the machine language translation that exactly fits a HLL instruction. This becomes difficult and time consuming when the number of choices that the compiler has for translating some HLL instruction is very large.

Further, CISC architecture compilers were expected to reduce the size of machine code generated. However, with the advances in technology, memory has become much cheaper and therefore is no longer a constraint. Additionally, it is not always true that CISC architectures generate smaller size machine code because the length of a single machine instruction is much larger. As the number of opcodes increase with the number basic machine operations available, the number of bits required to specify the opcodes and the various addressing modes also increase.

In CISC architectures instruction execution was expected to be faster. This is because in order to execute a single HLL instruction much fewer number of machine instructions need to fetched and executed. However, in practice this is not always so. To allow for such a sophisticated set of instructions, either the control unit needs to be much more complex or in case of micro programmed control unit, the number of stored micro programs will be much larger. In both of these cases executing a single machine instruction would require more number of clock cycles. This is true not only for the complex operations but also for the simpler operations that are more frequently used during program execution.

14.4 REDUCED INSTRUCTION SET COMPUTERS (RISC)

While CISC architectures were initially designed to reduce memory requirements, complexity of compilers and memory requirements, researchers have later recognized that there were several disadvantages to the approach. Because of these disadvantages, several groups have

decided to pursue a completely opposite approach to CISC architecture. The Reduced Instruction Set Computers (RISC) have characteristics that are completely opposite of CISCs. Instead of having complex hardware to simplify the software (viz. compilers), RISC architecture aims to simplify the hardware at the cost of complex software. At the hardware level only the most basic and most frequently used operations are incorporated, in order to keep the number of basic operations as small as possible. As a result of this, even though the compilers are more complex and the number of machine instructions generated for a HLL program is much larger, the time required to execute a single machine instruction is constant. This enables the processor to execute programs at the rate of *one instruction per cycle.* With such a simple set of machine instructions, it becomes possible to have simple hardwired control unit. As result, a single machine instruction can be executed much faster.

RISC architectures usually ensure faster execution of operations by ensuring that instructions are executed on operands that are stored in registers. Registers are much faster storage devices and therefore if frequently used operands remain on registers, the execution time becomes much faster. Further, this ensures that the control unit is even simpler because the only operations that require memory access are simple LOAD and STORE operations. This characteristic of RISCs necessitates a large number of registers available for the processor in order to keep the operands on the registers for a longer duration and further reduce the number of memory access operations.

Since most instructions in RISC architectures allow only register-register operations, the number of addressing modes required is also significantly reduced. This further simplifies the instruction set and the control unit. A simple instruction set in turn leads to simple instruction formats. Instruction lengths in RISC architectures are fixed and are usually aligned to word boundaries. The number of bits required for opcode is fixed and relatively small because of the small number of available basic operations. The number of bits required for specifying addressing of operands is also fixed and small, because all operands are in registers and registers have shorter address lengths. The simple instruction format requires simple hardware for decoding in the control unit and a uniform number of instructions fetch

operations. All the characteristics of RISC architectures are aimed at simplifying the control unit to improve efficiency of instruction execution.

14.4.1 Performance Analysis of RISC

The characteristics with which RISC architectures were designed allow for several performance benefits. While the complexity of compilers is increased with RISC architecture, it is also true that a lot of power to generate more efficient machine code rests with the compiler. The compilers are better enabled to optimize the machine code because the available machine instructions are more primitive. Further, the compiler is not required to find an exact match of machine instruction for every HLL instruction. Instead, depending on how well the compiler is designed, it may be possible to generate a more effective and efficient machine code for a given HLL program.

Most instructions in any HLL program involve only simple operations. As a result, the machine code generated by the compiler would involve a lot of simple operations that can be executed much faster due to the simple control unit.

The instruction format in RISC architecture is relatively simple and uniform. As a result, the various stages of execution for all instructions are also more or less uniform. This uniformity allows for pipelining to be implemented more efficiently than compared to CISC architectures.

However, most of the perceived advantages of RISC over CISC are based on a lot circumstantial evidence. It is worth noting that there are certain features of CISC architectures that can be incorporated in a RISC design and vice versa for a more well-rounded system. Table 10.1 lists some of the key differences between RISC and CISC architectures.

Table 10.1: Differences between RISC and CISC

CISC	RISC
• Emphasis on hardware • Includes multi-clock complex instructions • Small code sizes, high cycles per second • Transistors used for storing complex instructions	• Emphasis on software • Single-clock, reduced instruction only • Low cycles per second, large code sizes • Spends more transistors on storage registers



CHECK YOUR PROGRESS

Q1. Give the full form of CISC and RISC.
Q2. Give two differences between CISC and RISC.



14.5 LET US SUM UP

- CISC architectures involve incorporating complex operations at the hardware level to enable simple translation of high level language to machine code.
- CISC architectures aim towards simplifying compiler design, support for very high level languages, and reduction of memory requirements for program storage.
- CISC machine instructions allow a large number of basic operations and addressing modes.
- RISC architectures are aimed at simplifying hardware at the cost of complex compilers. Compilers in RISC architectures are expected to be able to effectively translate HLL programs.
- RISC architectures incorporate only the most basic and frequently used instructions at the hardware level. This significantly reduces the hardware complexity.
- RISC architecture allows execution of a single machine instruction in one clock cycle.
- RISC architectures have fewer operations and addressing modes. This reduces the length of a single instruction and allows for uniformity in the instruction format.



14.6 FURTHER READINGS

1) Hamacher, V. C., Vranesic, Z. G., Zaky, S. G., Vransic, Z., & Zakay, S. (1984). Computer organization (Vol. 3). New York et al.: McGraw-Hill.
2) Stallings, W. (2000). Computer organization and architecture: designing for performance. Pearson Education India.

