



COS 101

INTRODUCTION TO COMPUTING SCIENCES



MIVA OPEN UNIVERSITY

HEADQUARTERS

PLOT 1059, O.P. FINGESI ROAD,

UTAKO, ABUJA

COURSE DEVELOPMENT TEAM

Course Developer/Writer: Dr Morolake Lawrence, PhD Computer Science

Language Editor: Udochchi Obiukwu, MA English Language

Instructional Designer: Samuel Ubaru, MSc Information Security

Table of Contents

Table of Contents	5
List of Figures	11
Course Introduction	12
Course Aims	13
Course Objectives	14
Study Session one	16
Introduction	17
1.1 Levels of Abstraction	19
1.2 Major Fields in Computing	20
1.3 Professions in Computer Science	22
1.3.1 Programmer	22
1.3.2 Software Engineer	23
1.3.3 Systems Analyst	24
1.3.4 System Manager	25
1.3.5 Network Manager	25
1.3.6 Researcher	25
1.3.7 Teacher	27
1.3.8 Chief Information Officer	27
1.4 Subject Areas in Computer Science	28
1.4.1 Artificial Intelligence	28
1.4.2 Theory of Computation	29
1.4.3 Human-Computer Interaction	29
1.4.4 Information Management	30
1.4.5 Computer Graphics	30
1.4.6 Software Engineering	31
Study Session Two	38
Introduction	39
2.1 Computer Science is all about Math	41
2.2 Men are Better Suited to Computer Science than Women	41
2.3 Computer Science is for Geniuses	41
2.4 Computer Security	42
2.5 The History of Computing	42
2.5.1 The Abacus	43

2.5.2 Jacquard's Mechanical Loom	43
2.5.3 Babbage's Counting Machine	44
2.5.4 Hollerith's Punch Cards	45
2.5.5 6BC	46
2.5.6 ENI6C	47
2.5.7 Knuth's Research	49
2.5.8 The IBM PC	49
2.5.9 Apple Macintosh	51
2.5.10 Microsoft Windows	53
2.5.11 Linux	54
Study Session Three	61
Introduction	62
3.1 Binary	65
3.1.1 Binary Basics	65
3.2 Binary in Computer Science	67
3.3 Digital Versus Analogue	68
Study Session four	77
Introduction	78
4.1 Powers of 2	80
4.2 Size Terminology	80
4.3 System Components	81
4.4 Input Devices	82
4.4.1 Keyboard	83
4.4.2 Mouse	84
4.4.3 Scanner	85
4.4.4 Digital Camera	86
4.4.5 Gamepad	86
4.5 Output Devices	86
4.5.1 Monitor	86
4.5.2 Printer	88
4.5.3 Speaker	91
4.6 Storage Devices	91
4.6.1 Tape Drive	91
4.6.2 Floppy Drive	93
4.6.3 Hard Drive	94

4.6.4 Optical Drive	96
4.6.5 Disk or Disc?	99
Study Session Five	104
Introduction	105
5.1 Overview	106
5.2 Main Memory	107
5.2.1 RAM	108
5.3 Computer Operation	109
5.3.1 Booting	110
5.3.2 CPU	111
5.2 Physical Characteristics	114
5.2.1. Cache	115
Study Session Six	122
Introduction	123
6.1 Challenges of Modern CPUs	125
6.1.1 Backwards Compatibility	125
6.1.2 Moore's Law	126
6.1.3 Pipelining	127
6.1.4 Heat Dissipation	130
6.1.5 Multiprocessing	131
6.1.6 The Problem of Comparing CPUs	131
6.1.7 CISC vs RISC	132
6.1.8 Clock Speed	133
6.1.9 Benchmarking	134
6.2 Hardware Devices	135
6.2.1 Motherboard	135
6.2.2 Graphics Card	135
6.2.3 Sound Card	136
6.2.4 Device Interfaces	136
6.2.5 USB	137
6.2.6 Firewire	139
6.2.7 IDE	139
6.2.8 SCSI	140
Study Session Seven	146
Introduction	147

7.1 Types of Software	148
7.1.1 System Software	148
7.1.2 Application Software	148
7.2 Utility Software	150
7.2.1 Malware	150
7.2.2 Viruses	150
7.2.3 Trojan Horses	151
7.2.4 Spyware	151
7.3 Avoiding Malware	152
Study Session Eight	159
Introduction	160
8.1 Functions of an Operating System	178
8.1.1 Process Management	178
8.1.2 File Management	181
8.1.3 Software Hooks	183
8.1.4 Memory Management	183
8.1.5 Event Management	187
8.1.6 Output Device Management	188
8.1.7 Security	190
8.1.8 Application and System Data Management	191
8.2 Current Operating Systems	192
8.2.1 Windows	192
8.2.2 UNIX	194
8.2.3 Mac OS	195
8.2.4 Palm OS	196
8.3 Suites and Components	197
8.3.1 Suites	197
8.3.2 Components	198
Study Session Nine	207
Introduction	208
9.1 Program Logic	209
9.2 Sequential Execution	209
9.3 Conditional Execution	211
9.4 Repetitive Execution	212
9.4.1 Counting Loop	212

9.4.2 Conditional Loop	213
9.4.3 Procedures	213
9.5 Exploration: Finding Control Flow	216
Study Session Ten	222
Introduction	223
10.1 Phases of Software Development	224
10.1.1 Specification	224
10.1.2 Design	225
10.1.3 Implementation	225
10.1.4 Testing	225
10.5 Maintenance	227
10.6 Development Paradigms	227
10.6.1 Waterfall	228
10.6.2 Rapid Prototyping	228
10.6.3 Spiral	230
10.6.4 Extreme Programming	231
10.7 Languages	231
10.7.1 Compilation	231
10.7.2 Programming Paradigms	233
10.8 Popular Programming Languages	236
10.8.1 C	236
10.8.2 C++	236
10.8.3 Visual Basic	237
10.8.4 Java	238
10.8.5 Perl	240
10.6 Language Names	240
Study Session Eleven	247
Introduction	248
11.1 Network Overview	249
11.2 Basic Parts of a Transmission	249
11.3 Network Sizes	250
11.4 Network Responsibilities	252
11.4.1 Delivery	252
11.4.2 Reliability	252
11.4.3 Performance	253

11.4.4 Security	254
Study Session Twelve	260
Introduction	261
12.1 Transmission Media	262
12.1.1 Twisted Pair	262
12.1.2 Coaxial Cable	263
12.1.3 Optical Fibre	264
12.1.4 Wireless	266
12.2 Protocols	267
12.2.1 Data Linking	267
12.2.2 Error Detection	268
12.2.3 Routing	270
12.2.4. Bridging	272
12.3. Flow Control	273
12.4 Encryption and Authentication	274
12.5 Compression	278
Study Session Thirteen	285
Introduction	286
13.1 Ethernet	287
13.2. Frame Relay	289
13.3 Bluetooth	290
Study Session Fourteen	297
Introduction	298
14.1. ARPANET	299
14.2 Commercial Internet	300
14.3 Early Applications for the Internet	301
14.4 The Future of the Internet	306
Study Session Fifteen	313
Introduction	314
15.1 T-lines	316
15.2 Dial-up Connections	316
15.3 DSL	318
15.4 Cable	319
Study Session Sixteen	327
Introduction	328

16.1 Internet Protocols	329
16.2 World Wide Web	330
16.3 HTML	331
16.3.1 A Note on Capitalisation	331
16.3.2 Character Formatting Tags	332
16.3.3 Layout Tags	333
16.3.4 Link Tags	333
16.3.5 Special Tags	334
16.3.6 HTML Tools	335
16.4 Browsers	336
16.5 Exact layout vs. Guidelines	337
16.6 Web Programming	338
16.7 Email	340
16.8 Chat and Instant Messaging	342
References	349
Glossary	351

List of Figures

Figure 1.1: Levels of Abstraction	20
Figure 2.1 Punch Card	45
Figure 2.2: €1500 IBM PC and specifications in 1980	52
Figure 3.1: Using lights to send signals	67
Figure 3.2: Todd and His Christmas Lights	67
Figure 3.3: Sampling	73
Figure 4.1: Metric size designation	82
Figure 4.2: 6 Personal Computer System	83
Figure 5.1: Simplified Computer Architecture	108
Figure 5.2: CPU Organisation	113
Figure 5.3: Example of assembly instructions	114
Figure 6.1: Pipelining	130
Figure 7.1: Know Who You're Trusting	154
Figure 8.1: TypeEdit program showing a virtual address.	185
Figure 8.2: TypeEdit program showing two virtual addresses	186
Figure 8.3: Chain of actions	190
Figure 9.1: Todd's Grooming Ritual	210

Figure 9.2: Todd Makes Two Martinis	215
Figure 11.1: Types of Networks	251
Figure 12.1: Types of wires	264
Figure 12.2: Routing	271
Figure 12.3: Secure and authenticated transmissions	277
Figure 14.1: FTP, Gopher, World Wide Web	304

Course Introduction

This course provides a comprehensive introduction to the fundamental concepts, tools, and applications of computer science. You will explore how computers work, from hardware and software components to the principles of programming, networking, and operating systems. Through various study sessions, you will gain insight into the history and evolution of computing, binary logic, digital systems, and data communication.

In this course, you will also examine the major fields and professions in computer science, such as software engineering, systems analysis, and artificial intelligence. By engaging with hands-on examples, discussions, and problem-solving exercises, you will develop the analytical and technical thinking required to understand and apply computer science principles in real-world contexts.

By the end of this course, you will have built a solid foundation to pursue advanced studies or a career in computing, programming, systems management, or any technology-driven field.

Course Aims

This course aims to provide learners with a broad and practical understanding of computer science as both a discipline and a profession. It aims to:

- 1. Build Foundational Knowledge of Computer Science Principles:** This aim focuses on introducing learners to the structure, logic, and major domains of computer science. You will explore how computers process

information, perform operations, and support innovation in various sectors.

- 2. Develop Understanding of Computer Systems and their Components:** Learners will gain knowledge of the essential hardware and software components that make up a computer system, including input, output, and storage devices, as well as operating systems and applications.
- 3. Expose Learners to Core Computing Concepts and Tools:** This aim is to familiarise learners with key ideas such as binary representation, programming logic, and the operation of networks and the Internet. These concepts provide the groundwork for more advanced computing studies.
- 4. Introduce Learners to Career Paths and Professional Roles in Computer Science:** The course aims to help learners understand the diverse professional opportunities in computing – including roles like software developer, system analyst, network manager, and researcher – and what each entails in practice.
- 5. Strengthen Analytical, Logical, and Problem-solving Skills:** Learners will be encouraged to think critically, apply logical reasoning, and use computational thinking to approach challenges, design solutions, and make informed decisions in technology environments.

Course Objectives

After studying this course, you should be able to:

1. Explain the meaning, scope, and relevance of computer science in today's world.
2. Identify the major fields and professions within computer science and describe their core functions.
3. Distinguish between various computer system components, including input, output, storage, and processing devices.
4. Describe the historical development of computing and its influence on modern technology.
5. Explain basic binary concepts and their application in computer operations.
6. Discuss different types of software and their functions in managing and running computer systems.
7. Explain the structure and functions of operating systems and their role in computer management.

STUDY SESSION ONE

INTRODUCTION TO COMPUTING

Introduction

Computers have become a ubiquitous feature of modern life. It would be difficult to get through a day without some activity involving a computer, be it composing email on a computer sitting on a desk, using a computer hidden inside a cell phone, or receiving a bill generated by a computer at the power company. Computer science allows all these activities to happen.

But what is computer science? It sounds simple enough—computer science is a branch of science that studies computers. But not everyone who works with computers is a computer scientist. The use and development of computers comprise a number of overlapping disciplines.

Before these disciplines are discussed, you need to understand a few terms.

A program is a series of steps to accomplish a given task. In general usage, a programme might refer to everyday instructions, written in English, such as instructions to change a tyre or register for a college class. In computer science, however, the term “program” refers to a series of steps given to a computer.

Learning Outcomes for Study Session One

When you have studied this session, you should be able to:

1. Explain the levels of abstraction.
2. Discuss the major fields in computing.
3. Describe the professions in computer science.
4. Differentiate between the system analyst and software engineer.
5. Explain the roles of system managers and network managers.
6. Describe the subject areas in computer science.

1.1 Levels of Abstraction

The level of abstraction in computer science refers to the distance between a particular view of a situation and its concrete reality. It is a key concept, and perhaps sounds more complicated than it is. In general, the term describes whether someone has a “big picture” view or is focused on details.

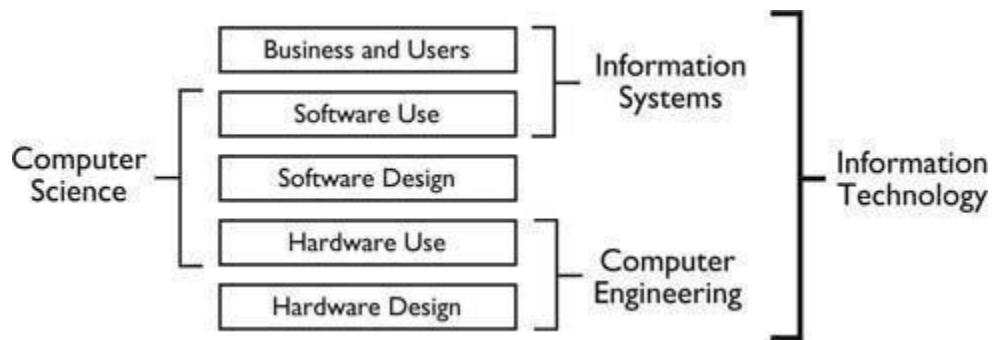


Figure 1.1: Levels of Abstraction

A person driving a car, for example, doesn't often think about what goes on under the hood. A mechanic does, but doesn't worry about the inside of most parts: if the part is broken, it gets replaced, not taken apart. On the other hand, a car designer must understand the engineering details of every part in the car. The driver, mechanic, and designer can view the same car at different levels of abstraction.

Figure 1.1 shows one way to divide computing into levels of abstraction.

The bottom level is hardware design. People working at this level understand how electrical properties within a computer work, and they can design new computing devices.

At the next level, hardware use, people understand how the computer works conceptually but have no concern over electrical properties or any knowledge of how to make a physical circuit. The level above that is software design, where people create new programmes for computers to execute. And then comes software use, the level at which people use existing programmes to perform tasks, not create new ones. Someone who is an expert at Microsoft Word works on this level.

The top level isn't about computing at all but about the people and businesses that use computers. Those who understand how a business works, or how users interact with programmes, operate at this level.

1.2 Major Fields in Computing

You may have heard terms like “computer engineering”, “computer science”, “information systems”, and “information technology” and wondered if they were synonymous. They are not, but they’re related through the levels of abstraction. You need to understand these terms too.

Computer engineering focuses on the bottom levels of abstraction: hardware design and use. It is an extension of electrical engineering, covering the design and analysis of computer hardware. In fact, at a university, computer engineering is most often taught in the school of engineering. While a computer engineer understands programming, most of his or her focus is on hardware, not software.

Computer science, the subject of this book, is the systematic study of computing processes. It is concerned with the middle levels of abstraction,

from hardware use to software use. Computer scientists work primarily in the development of software, either at the practical level (improving the speed at which a web page performs a search) or the theoretical level (exploring the limits of computers' recognition of human speech). The hope is that every subject that begins as theoretical will end as practical. To better understand software design and development, the computer scientist must understand how a computer works, even if he or she can't create one and is thus also an experienced computer user. In a university, computer science is most often taught in the school of science and mathematics.

Covering the top levels of abstraction is information systems, which is the study of how computing technology is used in business. Someone schooled in information systems has knowledge of business operations and programming, but is more interested in solving a business problem with existing solutions than in trying to invent novel solutions. In a university, information systems may be taught in the same department as computer science or in the business school.

Finally, information technology is a broad term that does not refer to a particular field but instead covers all levels of abstraction. While often restricted to mean the use of technology in business (similar to information systems), in general, the term encompasses the design, development, and implementation of computer hardware and software.

1.3 Professions in Computer Science

What, then, do computer scientists do? There are many professions for people with degrees and a background in computer science. Consider a few of the most common professional jobs.

1.3.1 Programmer

A computer scientist's natural activity is in front of a computer, writing a program, so of course it stands to reason that his or her general job title is 'programmer'. While the basic concepts of programming never change, two programmers can have jobs that are very different from each other. Writing a computer game, for example, is different from writing tax-preparation software.

Two broad subcategories of programming are applications and systems. An application programmer writes programs directly for users. Someone who helped create Microsoft Word is an application programmer. A systems programmer writes programs that operate computers behind the scenes. For example, a systems programmer would create the software that directs traffic around the Internet.

Even within these categories, programmers have different specialised knowledge. A programmer writing software for a bank needs some knowledge of finance, while one developing software to display molecules needs some background in chemistry.

It should be noted that while "programmer" is still used as a general term, it has fallen out of favour as a job title. This is because the work of developing

software involves creating specifications and designs; that is, it's now more than just programming. In response, a new term has arisen to reflect the total process: software engineer.

1.3.2 Software Engineer

A software engineer is involved in all stages of software development, from the initial meeting with prospective clients to installing updates to a program years after it was first developed. In truth, most programmers are actually software engineers, since it would be difficult for anyone to do a good job of writing a program if they hadn't been involved in the early stages of design.

The term "software engineer" arose not only to better reflect a total process, which includes programming, as we noted above, but also because of the growing importance of correct software in the computer industry. Computers are increasingly involved in our lives and are often critical to what we do, so software failure could be catastrophic.

Consider a construction analogy in which a 100-storey tower has been designed and tested by someone who is called a "builder". If you worked there, you'd no doubt prefer that the building be designed by a "structural engineer", someone who rigorously tested the design in accordance with accepted principles. Similarly, an astronaut boarding a space shuttle doesn't want to rely on a computer program written by a mere "programmer" to bring the craft safely back home but instead wants the software written by a "software engineer". Again, these activities imply that a formal, proven process was used

to create the software. It gives greater assurance that the software is free of serious defects.

1.3.3 Systems Analyst

A systems analyst makes decisions when whole systems must be introduced, upgraded, or replaced. If a chain of grocery stores determines that its current inventory control system is inadequate, for instance, a systems analyst—or team of analysts—would decide what the best solution is, taking into account all the costs involved, including purchasing new hardware, developing new software, training employees to use the new system, and so on. The best solution for the grocery store chain could involve replacing all the computers at the checkout counters or writing new software for the existing hardware.

You can see in the above example that the term “system” encompasses not only computers and software but everything that interacts with them, including the people who use them. A good systems analyst must take the skills, needs, and wants of other employees into account when making decisions.

Note that many people with the title “systems analyst” do not analyse systems exclusively, especially in smaller organisations. They are also involved in the development of the software once the system plan has been approved.

Thus, systems analysts are often software engineers as well.

1.3.4 System Manager

Once a new system is in place, someone must ensure that it continues to work.

A person responsible for maintaining an existing computer system is a system manager. He or she monitors the hardware and software and, if the needed use of the system outstrips its capacity, prioritises requests.

The system manager also supervises day-to-day tasks with the system, such as the replacement or repair of malfunctioning equipment, and is involved in the same kind of high-level decisions as systems analysts. At some organisations, both roles may be combined into one position, which is given an omnibus title such as “information technology manager”.

1.3.5 Network Manager

A network is a set of computers connected together so they can share data. A network manager is a kind of system manager who specialises in network operations: keeping the network operational, connecting new computers to the network as new employees are hired, upgrading the networking technology as needs change, and performing similar tasks.

This position is fraught with peril because, at many offices, all work must cease when the network is “down” or non-operational.

1.3.6 Researcher

A computer science researcher is involved in a formal investigation of the computer science field, which is a little different from research in other sciences.

A researcher in chemistry, for example, might mix several chemicals together as an experiment, observe the results, determine the properties of the existing compound, and compare this result with the result that was expected—the hypothesis. A computer science researcher, in contrast, does not generally conduct experiments. Since it is exactly known how a computer will interpret a given instruction or set of instructions, the researcher will know, or can test, whether a given idea will work before it is actually implemented as a program. Indeed, much research can be done without using a computer at all.

Research in this field can be practical or theoretical. Practical research has a known application already, such as an improvement to an existing process, for instance, a method to search Web pages faster or better. Theoretical research is meant to advance the discipline, without a specific practical goal in mind. Of course, today's theoretical solution may turn out to have practical ramifications tomorrow.

At one time, faculty members at colleges and universities conducted the majority of the research in computer science with grants from public and private research organisations funding at least a portion of their salaries. Further research came from candidates for doctoral degrees whose research topics related to their faculty mentor. While academic research still takes place, an increasing amount of research is done by private companies. Because the software industry is so lucrative, market forces can often drive research faster than academic concerns. While an exceptional computer science department in a university might have an annual research budget of

\$5 million, Microsoft (the world's largest software company) has an annual research budget of over \$5 billion.

1.3.7 Teacher

Like all disciplines, computer science needs people in the current generation to teach those in the next generation and pass along the accumulated knowledge and experience of the field.

At one time, most teachers of computer science were college professors. Now, computer science is often taught in high school as well. And because the industry is advancing so fast and there's a need to teach workers who are already in the field, companies employ teachers as well. Commonly, they give seminars to keep a company's employees up-to-date with the latest technologies.

1.3.8 Chief Information Officer

Not many people have the title Chief Information Officer, but that this title even exists is testament to the importance of computing in the business world. A Chief Information Officer, or CIO, is at the highest level of management, involved with all the central decisions affecting the company.

This constitutes a historical change in modern companies. Before there were CIOs, computing was considered an appendage of business, rather than an integral part of it. Like a service department, it was called when something specific was needed. Of course, computers now help guide a company's direction. Information technology can no longer be considered an

afterthought after the strategic plans have been made. The use of technology must be part of the plans themselves.

Chief Information Officers come from a variety of backgrounds but tend to have education and experience both in computer science and business.

In-Text Question and Answer

1. **ITQ:** What does the term "level of abstraction" refer to in computing?
ITA: The level of abstraction in computing refers to the degree of detail and complexity at which a system or problem is viewed or understood.
2. **ITQ:** How is the concept of abstraction demonstrated in the example of a driver, mechanic, and car designer?
ITA: The driver sees the car in terms of operation, the mechanic understands its internal components, and the car designer focuses on the engineering details, each representing a different level of abstraction.

1.4 Subject Areas in Computer Science

Within the computer science field, computer scientists can work in many areas. Depending on the profession, some computer scientists may need to know a little about each area, while others may need deep knowledge of one or two areas.

1.4.1 Artificial Intelligence

Artificial intelligence can be described as programming computers to perform tasks that would require intelligence if humans were performing the tasks. This is not the only definition, though, and of all the areas in computer science, this one has perhaps the most contentious boundaries.

Some researchers believe artificial intelligence must mimic the processes of the human brain; others are interested only in solving problems that seem to require intelligence, like understanding a request written in English.

1.4.2 Theory of Computation

The theory of computation puts limits on what can be computed. Some limits are practical. It may be shown, for instance, that a computer could solve a certain problem, but it would take hundreds of years to get the result. Other limits are absolute. Strange as it may seem, some questions have a fixed, numerical answer that cannot be computed.

Scientists in this area also compare programming solutions to specific tasks in a formal way. For example, a common computing task is sorting, which just means to put items in some order (like alphabetising a list of student records by last name). Countless ways can be used to approach the sorting problem, each with advantages and disadvantages. Computational theory is used to determine which situations are most suited to a particular approach.

1.4.3 Human-Computer Interaction

The computer scientist working in human-computer interaction investigates how people use computers now and how people and computers can work together better in the future.

This research is similar to graphic design. A graphic designer is a specialist who knows how the colours, fonts, arrangement of text, pictures, and other elements make a book, magazine, or advertisement easier for the viewer to

understand. Now that computer interfaces are increasingly graphical, the same kinds of ideas are used, except that a computer is interactive.

For example, many programmes now have a “toolbar”, which is a row of pictures that allow the user to select commonly used operations without navigating the entire menu of options. This kind of design innovation is the result of studies in human-computer interaction.

1.4.4 Information Management

A database, in general usage, is any organised collection of data. In computer science, a database specifically means a collection of data that is stored in a computer-readable form. Examples include an online book catalogue at the library or the account information for each person who has a VISA card.

The information management area is concerned with how databases are created, stored, accessed, shared, updated, and secured.

1.4.5 Computer Graphics

Computer graphics is the generation of images through computers. It includes simple text displays as well as images that appear to be in three dimensions.

An important part of computer graphics is computer visualisation, which attempts to visually display data in a way that is most understandable for the user. For instance, visualisation can allow surgeons to preview a surgical procedure before actually performing it. Other forms of visualisation involve data that has no natural pictorial form. As a result, these must be displayed in a form that tells a story or makes a point. If you've seen a graph or chart

generated with a computer program that seemed to have no clear meaning, you know why this area is important.

Like many areas in computer science, computer visualisation is as much human psychology as machine capability. The computer visualisation expert asks, “How do we as human beings process visuals?”

As computer graphics become more advanced, they terminate at a point called virtual reality, in which graphics and sensory feedback are all-encompassing. This can be seen, for example, in a room in which every surface is covered with synchronised computer displays. It’s important to note that virtual reality does not promise an experience indistinguishable from the “real world”, although that may be a goal for some researchers. Rather, it is an experience in which the outside world is temporarily blocked from our senses.

1.4.6 Software Engineering

As previously discussed, a software engineer is involved with the entire process of a program’s development, not just programming. Software engineering is concerned with improving the process of making software. This means creating new processes for software engineers to follow, new techniques for project management, new methods to test software to ensure its quality, and new metrics for measuring how effective any of the other new ideas have been.

Conclusion

A computer is any programmable electronic device. To use a computer, one

must have hardware (all the physical parts of the computer itself) and software (the programs that execute on the computer). Programmers write programs, and users use them.

Computer science is the systematic study of computing processes. It is not directly involved with the design of new hardware, but rather with the use and improvement of hardware that currently exists. Computer science is not directly concerned with business operations, but it wants to provide the tools businesses and others can use.

Summary

In this session, you have learnt to:

1. Explain the levels of abstraction.
2. Discuss the major fields in computing.
3. Describe the professions in computer science.
4. Differentiate between the system analyst and the software engineer.
5. Explain the roles of system managers and network managers.
6. Describe the subject areas in computer science.

Self-Assessment Questions and Answers (SAQs and SAAs)

SAQ 1.1. What is the level of abstraction in computer science?

SAA 1.1. The level of abstraction in computer science refers to the degree of detail and complexity at which a system or problem is viewed or understood. It

represents a way to simplify or generalise the representation of a concept or system.

SAQ 1.2. Give an example of different levels of abstraction in the context of driving a car.

SAA 1.2. Different levels of abstraction in driving a car can include:

- **The Driver's Perspective:** The driver focuses on the car's steering, acceleration, and braking, with less concern for the mechanical details.
- **The Mechanic's Perspective:** The mechanic understands the car's internal components, such as the engine and transmission, and is concerned with maintenance.
- **Car Designer's Perspective:** The car designer is involved in the engineering and design aspects, considering aerodynamics, structural integrity, and safety features.

SAQ 1.3. What are the different levels of abstraction in computing?

SAA 1.3. The different levels of abstraction in computing can be categorised into three main levels:

- **Hardware Level:** This level deals with the physical components of a computer system, such as electronic circuits, processors, memory, and storage devices.
- **Software Level:** This level focuses on the programs and instructions that control the hardware, including operating systems, programming languages, and application software.

- **Problem-solving Level:** This level involves abstracting real-world problems into computational models, algorithms, and data structures to develop solutions using software.

SAQ 1.4. What is the role of a programmer in computer science, and how does it differ from that of a software engineer?

SAA 1.4. A programmer is responsible for writing code and creating software applications based on specifications. They focus on the implementation and coding aspects. A software engineer has a broader role, involved in the entire software development lifecycle, including requirements gathering, system design, testing, deployment, and maintenance, with a comprehensive understanding of development processes and principles.

Multiple-Choice Questions and Answers (MCQs and MCAs)

1. **Which of the following best describes the concept of “levels of abstraction” in computing?**
 - A) The speed at which computers process data
 - B) The distance between a system’s conceptual view and its physical implementation
 - C) The number of software layers on a computer
 - D) The graphical representation of hardware components

Correct Answer: B) The distance between a system’s conceptual view and its physical implementation

Explanation: Levels of abstraction represent how closely one's view is to the underlying details of a system, ranging from high-level concepts to low-level technical implementations.

2. Which discipline primarily focuses on hardware design and use?

- A) Computer Science
- B) Computer Engineering
- C) Information Systems
- D) Information Technology

Correct Answer: B) Computer Engineering

Explanation: Computer engineering deals mainly with designing and analysing hardware, making it the discipline focused on the lowest levels of abstraction.

3. Which of the following best defines Computer Science?

- A) The study of computers limited to business applications
- B) The study of how computers are manufactured
- C) The systematic study of computing processes from hardware use to software development
- D) The exclusive study of software marketing and management

Correct Answer: C) The systematic study of computing processes from hardware use to software development

Explanation: Computer science bridges hardware understanding and

software design, focusing on computing processes and problem-solving methods.

4. **What is the main responsibility of a systems analyst?**

- A) Writing software code for mobile applications
- B) Designing graphics for software interfaces
- C) Evaluating and deciding on suitable technology solutions for organisations
- D) Monitoring hardware performance on local machines

Correct Answer: C) Evaluating and deciding on suitable technology solutions for organisations

Explanation: Systems analysts determine how to introduce, upgrade, or replace systems by assessing organisational needs, costs, and technical feasibility.

5. **Which of the following roles focuses on maintaining computer systems after implementation?**

- A) Network Manager
- B) Software Engineer
- C) Systems Analyst
- D) System Manager

Correct Answer: D) System Manager

Explanation: A system manager ensures that existing systems continue to

function efficiently and oversees maintenance, repair, and performance optimisation.

Tutor-Marked Assessments (TMAs)

- TMA 1.1.** The system manager and network manager share overlapping responsibilities but work in distinct contexts. Discuss this difference and its implications for organisational performance.
- TMA 1.2.** Examine the responsibilities of a systems analyst and discuss how human factors influence system design and implementation.
- TMA 1.3.** In what ways has the rise of the Chief Information Officer (CIO) transformed the relationship between computing and corporate strategy?
- TMA 1.4.** Write short notes on the roles of the following professionals:
- Programmer
 - Software Engineer
 - Systems Analyst
 - System Manager
 - Network Manager
 - Researcher

STUDY SESSION TWO MYTHS OF COMPUTER SCIENCE

Introduction

A computer is an electronic device for performing logical and mathematical operations based on its programs. The term includes not only the obvious electronic devices that have a screen, keyboard, printer, and so on, but also computers that are embedded into devices like those at supermarket checkout counters or in DVD players. What makes computers interesting and powerful is that they can be given arbitrary sets of instructions to perform.

Hardware refers to all the physical devices that make up a computer system, both those inside the computer “case” and those outside the case, like the monitor, keyboard, and mouse.

Software refers to the programs the computer executes. For example, the word processor Microsoft Word, or the computer game “Half-Life”, is software, as is a program that enables a cell phone display so the user can select a new ringtone. By analogy, when you play a movie on a DVD player, the movie is the software and the player is the hardware.

A programmer is someone who creates programs. 'User' refers to a person who uses a software program or computer.

Learning Outcomes for Study Session Two

When you have studied this session, you should be able to:

1. Explain the role of maths in computer science and its varying degrees of involvement depending on the area of specialisation..
2. Identify that interest and success in computer science are not determined by gender or ethnicity, emphasising that anyone can excel in the field.
3. Differentiate between genius and proficiency in computer science, acknowledging that extraordinary abilities are not the sole requirement for success.
4. Highlight the significance of computer security in protecting sensitive data from unauthorised access and the growing demand for professionals in this field.

5. Explore the history of computing, from the abacus and Jacquard's mechanical loom to Babbage's counting machine and Hollerith's punch cards, to gain a broader understanding of the evolution of computers and computer science.
6. Appreciate the contributions of pioneers like Atanasio, Berry, Mauchly, Eckert, and Knuth and the development of the ABC, ENIAC, and IBM PC, which marked significant milestones in the advancement of computer technology.
7. Highlight the transformative impact of personal computers, such as the IBM PC and Apple Macintosh, in bringing computing power and usability to individuals' homes and workplaces.

2.1 Computer Science is all about Math

The kind and degree of maths involved with computer science depend on what area one works in. Most programming involves maths no more advanced than high school algebra, but some specialities require more. Someone who writes a mortgage interest calculator would need to understand financial calculations. Someone who writes a program to plot the trajectory of a satellite through space needs to understand trigonometry and calculus. Most programs, though, are built upon basic operations like addition and multiplication.

2.2 Men are Better Suited to Computer Science than Women

Judging by the number of men and women working in the field, one could say that men as a group are more interested in computer science than women. But there's nothing to suggest that men are better at it. Women may have shied away from computer science due to a dislike of maths (which is probably the result of another myth) and media portrayals of computer scientists as awkward, pasty-faced "geeks". Computer science is a field that rewards excellence, regardless of gender or ethnicity, and all those interested should apply.

2.3 Computer Science is for Geniuses

Genius never hurt anyone in the sciences, but having a high IQ and having a knack for programming and other computer science concepts are two different things. While the people at the top of any profession usually have

extraordinary abilities (that's why they're at the top), plenty of "ordinary" people have excelled in this field.

2.4 Computer Security

Much sensitive data is stored on computers, including tax records, credit card bills, bank accounts, and medical histories. And with computers becoming increasingly interconnected, it's become easier for data to be stolen. The old adage, "A chain is only as strong as its weakest link," shows its truth here in the information age, where every computer is a link to another. So, it's no surprise that computer security is a rapidly growing field.

Computer security involves finding ways to protect data from unauthorised access. This includes installing software to limit intrusions to a network, instructing employees on safe habits, and analysing the aftermath of a system break-in to learn how to prevent a recurrence.

A related field is computer forensics, though in fact this is almost the reverse of computer security since it involves breaking through security to retrieve partially deleted files. The purpose of this "break-in" is to obtain and analyse evidence to be used in a court trial.

2.5 The History of Computing

The presence of computers on desks in homes and businesses is a recent idea, having only flowered in the past twenty-five years, but the idea of a machine to perform computations is a surprisingly old one. In this section, we'll explore a few of the highlights in the history of computers and computer science.

2.5.1 The Abacus

The first efforts towards mechanical assistance aided in counting, not computation. An abacus is a mechanical device with beads sliding on rods that is used as a counting device. It dates to at least the Roman Empire, and its ancestor, the counting board, was in use as far back as 500 B.C. The abacus is considered a counting device because all the computation is still done by the person using it. The abacus did show, however, that a machine could be used to store numbers.

2.5.2 Jacquard's Mechanical Loom

A loom is a machine for weaving a pattern into fabric. Early loom designs were operated by hand. For each “line” in the design, certain threads were “pulled” by an experienced weaver (or a poor worker under the direction of a weaver) to get the finished pattern. As you might guess, this process was slow and tedious.

In 1801, Frenchman Joseph-Marie Jacquard invented a mechanical loom in which threads had to be pulled at each stage in a pattern that was stored in a series of punch cards. A punch card encodes data with holes in specific locations (Figure 2.1). In the case of weaving, every thread that could be pulled had a location on the card. If there was a hole in that location, the thread was pulled. Jacquard’s loom used a series of these punch cards on a belt. The loom would weave the line dictated by the current card, then automatically advance to the next card.

1	2	3	4	5	6	7	8	9	10
■ 0	0	0	0	■ 0	0	0	0	0	0
1 1	■ 1	1	1	1 1	1	1	■ 1	1	
2 2	2	2	2	2 2	2	2	2 2	2	2
3 ■ 3	3	3	■ 3	3 3	3	3	3 3	3	3
4 4 4	4	4	4 4	4 4	4	4 4	4 4	4	4
5 5 5	5	5	5 5	5 5	■ 5	5 5	5 5	5	5
6 6 6	6	6	6 6	6 6	6 6	6 6	6 6	6	6
7 7 7	7	7	■ 7	7 7	7 7	7 7	7 7	7	7
8 8 8	8	8	8 8	8 8	8 8	8 8	8 8	8	■ 9
9 9 9	9	9	9 9	9 9	9 9	9 9	9 9	9	9

Figure 2.1 Punch Card

Jacquard's loom is not necessarily a computer because it does no mathematical calculations, but it introduced the important idea of a programmable machine. The loom is a “universal weaver” that processes different sets of punch cards to make different woven fabrics.

2.5.3 Babbage's Counting Machine

Eventually someone put together a machine that could count and execute a program and wondered if a machine could be made to compute numbers. Charles Babbage, an English researcher, spent much of the 1800s trying to develop just such a machine.

One of Babbage's early designs was for a device he called the “Difference Engine”, which produced successive terms in a mathematical series while an operator turned a crank. This may not seem a dramatic development, but at the time, mathematicians relied on tables of mathematical functions in which each value had been painstakingly calculated by hand. Thus, the Difference Engine was revolutionary.

Its success led Babbage to a more ambitious design: the Analytical Engine. Rather than being tied to a specific task like the Difference Engine, the Analytical Engine was conceived as a general-purpose computing device. Different programmes would be fed to the machine using a belt of punch cards, just as in Jacquard's loom.

The Analytical Engine was never built because Babbage ran out of money. Like many researchers today, he was dependent on government grants to continue his work. In addition, his design may not have been possible to implement using the technology of the day. He was undoubtedly a man ahead of his time.

2.5.4 Hollerith's Punch Cards

Under its Constitution, the U.S. government is required every ten years to count how many people reside in each state, a process known as the census. These numbers are used to determine the proportion of representatives each state receives in the House of Representatives.

Originally, this process was done entirely by hand. Census takers would fill out forms for each household, and then the results of these forms would be tabulated by state. This method was so onerous that by the late 1800s, the 1880 census took more than ten years to complete, which meant that the next census was starting before the results of the previous one were known. Clearly, something had to be done.

The government created a contest to find the best solution to the problem. In 1890, it was won by a census agent named William Hollerith. In his design, each

census form was encoded into a punch card. Machines called “tabulators” could rapidly process stacks of these cards.

This method was not only dramatically faster than manual tabulation, but also allowed the government to track demographics as never before, ask more questions of each citizen, and break up data into multiple categories. For example, rather than counting men who were above a certain age or were veterans, the tabulators could count the men who were in both categories, which allowed the government to better anticipate the funds that would be needed for veterans’ pensions.

The system was a resounding success and led to Hollerith’s founding of the Tabulating Machine Company, which, several mergers later, became International Business Machines, or IBM, a company that would later dominate the world of computers for decades.

2.5.5 6BC

In the period from 1939 to 1942, John Atanasoff, a professor at Iowa State University, and Clifford Berry, a graduate student at the same school, created what is now considered the first modern computer. Their machine, which they called the Atanasoff-Berry Computer, or ABC, weighed about 700 pounds and had to be housed in the basement of the physics department. By current standards, it was a terribly slow machine, reportedly capable of only a single calculation every fifteen seconds. In contrast, a computer today can perform billions of calculations per second.

Atanasoff and Berry never completed the patent process on their work, and the machine itself was dismantled a few years after it was built, when the physics department needed its basement space back. This was unfortunate, as their pioneering work in the field was underappreciated. Credit for the first modern computer was instead bestowed on a more famous project: ENIAC

2.5.6 ENIAC

Like William Hollerith's punched cards, the ENIAC story is driven by governmental need. When World War II began, the United States was woefully underprepared for military operations. The army needed to develop and test a large number of weapons in a short period of time. In particular, it had to perform a number of ballistics tests to create artillery tables—in essence, a book showing how far an artillery shell would fly from a specific gun, given wind conditions, the angle of the gun barrel, and so on.

Like the mathematical tables of Babbage's time, these artillery tables had been created by hand, but by now the army already had some devices for assisting in calculation. Called differential analysers, they operated on mechanical principles (much like Babbage's machines), not electronics. But something better was needed, in aid of which the army hired John Mauchly and J. Presper Eckert, computer scientists at the University of Pennsylvania. In 1946, the machine they proposed was called ENIAC, which stands for Electronic Numerical Integrator and Computer. Like the ABC, it was truly a modern computer.

The term “modern” might seem too strong if you actually saw this machine. Computers of that era relied on the vacuum tube, a device that resembled a lightbulb through which one electrical current could control another. This controlling aspect was used to build logical circuits because, by itself, one vacuum tube doesn’t do much. Indeed, ENIAC required about 19,000 vacuum tubes to do its work, filled an entire room, weighed thirty tonnes, and drew about 200 kilowatts (that is, 200,000 watts) of power. In comparison, a desktop computer purchased today would draw about 400 watts of power, which means ENIAC drew about 500 times more current, even though its actual ability to compute is dwarfed by the most inexpensive desktop computers of today.

What makes ENIAC so important is its reliance on electronics to solve a real-world problem. There were as few mechanical parts as possible, although some mechanics were inevitable. For example, ENIAC still used punch cards for input and output, and the parts that read and produced these cards were mechanical. The vacuum tubes were built into mini circuits that performed elementary logical functions and were built into larger circuits. Those circuits were built into even larger circuits, a design idea that is still used today.

For decades, the ENIAC was considered the first computer, but in the 1970s, a judge in a patent infringement case determined that the ENIAC was based on the designs of the ABC. Other claims were also made, including those of Konrad Zuse, whose work in wartime Germany wasn’t known to the rest of the world for decades, and the Mark I, a computer developed around the same

time at Harvard. The question of what the first modern computer was may never be settled.

2.5.7 Knuth's Research

To this point, computers were seen as increasingly useful tools, but computer science was not considered a serious discipline, separate from mathematics. One of the leading figures who changed this was Donald Knuth.

As an undergraduate studying physics and mathematics at the Case Institute of Technology in Cleveland in the 1950s, Knuth had his first contact with the school's IBM computer. From then on, computers and programmes were his obsession. He wrote programs for the IBM computer to analyse the college's basketball statistics and published research papers while still an undergraduate. When he completed the work for his bachelor's degree, the college was so impressed with his computer work that he was awarded a master's at the same time. His most famous accomplishment is *The Art of Computer Programming*, a proposed masterwork of seven volumes, of which three have been completed. It's no exaggeration to say that Donald Knuth's writings are to computer science what those of Albert Einstein are to physics.

2.5.8 The IBM PC

For decades after World War II, computers were shared. Though they had grown smaller since the days of ENIAC, they were still very large and expensive. As a consequence, entire universities or companies, or even groups of schools and companies, would share the use of a single, large, and—for the time—powerful computer.

The people who used this mainframe, as these computers were called, would often never see it. The computer was generally locked in a secure location, and users would connect to it through phone lines or other wiring. At first, all the user saw was a teletype, which is like a combination of an electric typewriter and a printer. But later, video screens were introduced, which showed green or orange text on a black background.

By the late 1970s, computer scientists realised that smaller, less powerful computers could be placed right on users' desks. "Garage" companies, so named because they were so small they might literally consist of one or two people working out of a garage, began making small computers for individuals, not whole companies. Sometimes these computers came in the form of kits.

IBM, the company that grew out of Hollerith's census tabulators, was at this time very successful in making mainframes. At first, IBM was sceptical that a market even existed for smaller computers, but eventually it decided to act. In 1981, the company introduced the IBM PC. "PC" stands for "personal computer", which is where that term for a single-user computer originates.

The price was over \$1,500, which in 1980 was a lot of money, but it was still remarkably inexpensive compared to mainframes. In addition, the IBM name gave the personal computer instant legitimacy. Yet, the IBM PC did not mark a revolution in technology. The machine had been built almost entirely from off-the-shelf parts. What was revolutionary was the concept: A computer would appear on our desks at work, would come into our homes, and would become an ordinary appliance like the telephone.

EXPLORATION: WHAT DOES \$1,500 BUY?

Find a computer for sale for around \$1,500, which was the cost of the original IBM PC. A good place to shop is Dell® (www.dell.com). Compare what you find to the features of the original home computer.

	IBM PC	Today's Computer
Processor speed (in gigahertz, or GHz)	about 0.005	_____
Main memory (RAM) (in megabytes, or MB)	about 0.02	_____
Size of disk drive (in gigabytes, or GB)	only had “floppy” drives; held less than 0.001 GB	_____
Input devices	keyboard only	_____
Output devices	screen only	_____

Figure 2.2: €1500 IBM PC and specifications in 1980

At the start of 1983, when Time magazine would normally select its “Man of the Year” for 1982, the editors instead selected “The Computer” as its “Machine of the Year”. The computer had come of age.

As you will find, today’s computers are not simply a little faster than the original desktops. A gymnasium full of IBM PCs would not equal the power of a single system today. You may not know what all these terms mean, but you will before you finish this book. For now, just marvel at how little \$1,500 once bought (or, to be optimistic, how much \$1,500 buys now).

2.5.9 Apple Macintosh

Although a great improvement over punch cards, some computer scientists saw limitations in a computer with only a keyboard for input and text for output. It was fine for researchers and computer experts to interact with the machine through obscure commands and oblique text messages, but if the

computer was going into every home, it needed to interact with users in a different way.

In the early 1970s, researchers at Xerox developed a series of computers that communicated with the user through pictures, not just words. The culmination of their early efforts was the Xerox Star. It had “windows”, a “mouse”, and many other elements you would recognise today. Eventually, this method of computer use that is mostly visual with little text would be called a graphical user interface (or GUI), and every computer would have one. Unfortunately for the executives at Xerox, they proved to be better at funding interesting projects than marketing the results.

Steve Jobs, the president of Apple Computers, toured the Xerox research facility in 1979, having traded some Apple stock for Xerox stock. He'd been told about this new interface and wanted to see it. He left impressed and decided that Apple's new computer, the “Apple Lisa”, would be the first mass-produced computer with a graphical user interface. Many of the Xerox researchers would soon be working at Apple.

Not many Apple Lisas were sold. It was an expensive computer, costing \$10,000 when it debuted in 1983. But because Jobs was convinced that the GUI was the model for the future, he tried again.

During the 1984 Super Bowl, Apple ran one of the most famous commercials in history to introduce their next computer, the Apple Macintosh. Directed by Ridley Scott, director of the movie Blade Runner, it depicted an Orwellian future of grey-clad workers who mindlessly pay homage to a “Big Brother” figure on a huge video screen until an athletic woman in running clothes smashes the

screen with a flying hammer. What this had to do with the Macintosh was never clear, but the commercial was widely discussed around office water coolers and was repeated on news programmes. Soon everyone had heard of the “Mac”.

The new computer was cheaper than the Lisa but less powerful. As with the Lisa, it was a slow seller at first, but the company stuck with it. There was no turning back to text-based computers.

2.5.10 Microsoft Windows

When IBM was readying its PC for launch, they needed an operating system, which is the core program that allows a computer to function. This is the program that starts to run when you turn on a computer, before you touch the keyboard or the mouse. Microsoft purchased an operating system from another company and adapted it for use on the IBM PC, calling its software MS-DOS, for Microsoft Disc Operating System.

In 1985, Microsoft's initial fortunes were made with MS-DOS, but once Microsoft chairman Bill Gates saw the Apple Macintosh, he knew the days of MS-DOS were numbered. Microsoft developed its own GUI, a program that would run on top of MS-DOS, and called it “Windows”.

Few people remember the original Windows today, or its sequel, Windows 2.0. They were crude interfaces by today's standards. They are most famous for starting a long legal battle with Apple, which claimed that the “look and feel” of Windows legally infringed on the designs of Apple computers. Microsoft, in

turn, claimed that Apple had stolen the important ideas from Xerox anyway. Microsoft eventually won.

In 1990, Windows 3.0 was released, the first version that was truly popular with users and developers. In 1995, Windows 95 was released, which, unlike the previous versions, was an entire operating system and interface in one and did not require that MS-DOS be installed beforehand on the system. Soon, Windows 98, Windows NT, Windows 2000, and Windows XP were developed. These products dominate the industry as of this book's first printing, and the vast majority of computers run some form of Windows.

As Microsoft became ubiquitous in the computing world, a backlash developed. Critics claimed the company's software presence on almost every computer in the world gave them an unfair competitive advantage. The company became entangled in antitrust lawsuits from rival companies and governments around the globe.

2.5.11 Linux

Some thought that the problem with Windows wasn't that it was nearly universal—there are advantages for everyone as far as that's concerned—but that it was proprietary. That is, it was owned or controlled by one company, which means that no one outside of that company knows exactly how the product works. In computer science, the opposite of 'proprietary' is 'open-source'. That is, no one company owns or controls the product, and anyone can look at how the product works and even make suggestions on how to improve it.

In 1991, Linus Torvalds, a student in Finland, made an innocent post to an email group. He wrote that he was working on a free version of UNIX, an operating system that had been in use for twenty years. The Torvalds version, which came to be called Linux (Linus UNIX), was released under an agreement called the General Public License. In essence, anyone could use Linux in any way and modify it in any way, as long as any changes made were not hidden or charged for. This ensured that Linux would remain “open”.

“Windows versus Linux” has become the main battleground in the larger war of “proprietary versus open-source”. Adherents on both sides believe their approach will lead to more effective software.

In-Text Questions and Answers (ITQs and ITAs)

ITQ 2.1: Why is computer science not solely about mathematics?

Answer: While mathematics is important in some areas of computer science, most programming uses basic operations like addition and multiplication. Advanced math such as trigonometry and calculus is only required for specific fields like space trajectory analysis.

ITQ 2.2: What factors contribute to the under-representation of women in computer science?

Answer: Media portrayals of computer scientists as “geeks” and a stereotype of computer science as a field dominated by math may discourage women. However, there is no inherent reason that women are less capable in computer science than men.

Conclusion

Computer scientists work in a variety of different jobs. Some work directly in the creation of new programmes and are called programmers or software engineers. Others work in research and teaching. Still others neither programme nor research but maintain existing computer systems.

Research areas in computer science include artificial intelligence, human-computer interaction, information management, computer graphics, software engineering, and computer security.

Human beings have always sought new tools to make life easier. Early tools eased our physical labour, and later tools, like the abacus, eased our mental labour. The idea of a mechanical tool to help with computation led to Babbage's work and eventually culminated in the ABC and ENIAC, ushering in the computer age. With the release of the IBM PC, computing came into the home. Other developments led to the Apple Macintosh, Microsoft Windows, and Linux.

Summary

In this session, you have learnt to:

1. Explain the history of computing.
2. Describe the abacus system with the aid of a diagram.
3. Examine the Jacquard's mechanical loom.
4. Explain Babbage's counting machine.
5. Describe Hollerith's punch cards.

Self-Assessment Questions and Answers (SAQs and SAs)

SAQ 2.1. What is an abacus, and what was its purpose?

SAA 2.1. An abacus is a simple mechanical device used for performing arithmetic calculations. It consists of a wooden frame with rods or wires, on which beads or stones can slide back and forth. By manipulating the beads, users can represent and manipulate numbers and perform addition, subtraction, multiplication, and division operations. The abacus was widely used in many ancient civilisations as a primary calculating tool.

SAQ 2.2. What was Joseph-Marie Jacquard's contribution to computing?

SAA 2.2. Joseph-Marie Jacquard was a French weaver and inventor who developed the Jacquard loom in the early 19th century. The Jacquard loom used punched cards to control the weaving of intricate patterns automatically. This punched card mechanism provided a programmable control system, which later influenced the development of early computing devices that used punched cards for data input and program control.

SAQ 2.3. What were Charles Babbage's inventions and their significance?

SAA 2.3. Charles Babbage was an English mathematician and inventor known for his work on mechanical computing devices. His most significant inventions were the Difference Engine and the Analytical Engine. The Difference Engine was designed to perform polynomial calculations, while the Analytical Engine was a more general-purpose machine that included concepts such as loops,

conditional branching, and memory. Although Babbage's machines were never fully constructed during his lifetime, they laid the foundation for modern computing and are considered precursors to the modern computer.

Multiple-Choice Questions and Answers (MCQs and MCAs)

1. Which statement best distinguishes hardware from software?

- A) Hardware is user input; software is output.
- B) Hardware is the physical device; software is the set of programmes it runs.
- C) Hardware is optional; software is mandatory.
- D) Hardware includes apps; software includes screens and keyboards.

Correct Answer: B) Hardware is the physical device; software is the set of programmes it runs.

Explanation: Hardware is the tangible equipment; software comprises the instructions/programs executed by the hardware.

2. Which of the following best defines a programmer?

- A) A person who uses applications
- B) A person who repairs computer hardware
- C) A person who creates programs
- D) A person who manages corporate networks

Correct Answer: C) A person who creates programs

Explanation: A programmer writes the instructions (software) a computer executes.

3. Which myth is directly addressed in the section “Computer Science Is all about Math”?

- A) Computer science requires calculus for all tasks.
- B) Only physicists can do computer science.
- C) All programming needs advanced mathematics.
- D) Most programming uses math no more advanced than high-school algebra.

Correct Answer: D) Most programming uses math no more advanced than high-school algebra.

Explanation: Advanced maths is needed in some specialties, but most programming relies on basic arithmetic/algebra.

4. What does the text say regarding gender and success in computer science?

- A) Men are inherently better than women.
- B) Women lack the capability for CS roles.
- C) Interest levels may differ, but excellence is independent of gender.
- D) Only men can handle advanced math.

Correct Answer: C) Interest levels may differ, but excellence is independent of gender.

Explanation: The field rewards excellence regardless of gender; disparities are not evidence of innate ability differences.

5. Which activity belongs to computer security rather than computer forensics?

- A) Recovering deleted files for a trial

- B) Analysing a breached system to prevent recurrence
- C) Extracting evidence from encrypted storage
- D) Reconstructing user actions from logs for court

Correct Answer: B) Analysing a breached system to prevent recurrence

Explanation: Security focuses on protection and prevention; forensics focuses on evidence recovery/analysis after incidents.

Tutor-Marked Assessments
(TMAs)

- TMA 2.1.** Describe a few of the highlights in the history of computers and computer science.
- TMA 2.2.** Jacquard's loom is not necessarily a computer, because it does no mathematical calculations. Explain this phenomenon.
- TMA 2.3.** Explain, using examples, the relationship between hardware and software, and discuss how they depend on each other to perform computational tasks.
- TMA 2.4.** Discuss societal and cultural factors that may contribute to the underrepresentation of women in computer science, and suggest how this perception might be changed.

STUDY SESSION

THREE

KEY TERMINOLOGY AND CONCEPTS

Introduction

Computer scientists classify computers based on their power and their use. A personal computer is designed for use by one person at a time. This is what most of us think about when someone says “computer”, because it’s the kind of computer we are most likely to see. The computers people use at home or students use in school labs are personal computers.

A laptop is a compact personal computer with all the devices shrunk into one container to make the computer easily portable. The general term “device” in computer science refers to any hardware component. Thus, a laptop is any computer where all the hardware is inside a single, readily carried container.

A workstation is also a computer designed for use by one person at a time, but it is connected to a network—a set of computers connected together to share data. At an insurance company, for example, all the employees may have computers on their desks, with the computers connected to a network so they can share customer data. While computers intended as workstations may be more powerful than personal computers, this is not always the case.

A mainframe is a powerful computer that is shared by multiple users at once. Each user has a terminal, which in this context is simply a keyboard and screen combination used to access the mainframe. If the terminal is nothing more than a screen and keyboard, it’s called a “dumb terminal” because it does no processing of its own. However, a workstation can also be used, in which case it’s called a “smart terminal.”

A minicomputer is any computer powerful enough to be used by multiple people but not powerful enough to be considered a mainframe. This term is fading from use.

A supercomputer is among the fastest of current computers. Because this classification is based on performance, no computer can stay in this category forever. Systems that were considered supercomputers ten years ago are ordinary in their abilities today.

A server is a computer on a network that provides a service to other computers. Mainframes are often used as servers, but not every server is a mainframe. Even a personal computer is powerful enough to be a server.

A client is a computer that uses a server for some service. An example of the client-server relationship is an automated teller machine. The ATM is a computer, but all it really knows is how much money it has left inside. When you request money from your account at an ATM, the ATM must consult a central computer at your bank to determine if your account has a high enough balance to grant your request and to debit your account once you have been given your money. The ATM is the client in this relationship, and the central computer at the bank is the server

Learning Outcomes for Study Session Three

When you have studied this session, you should be able to:

1. Explain the binary system.
2. Describe the binary basics.
3. Describe binary in computer science.
4. Highlight the concepts of digital and analogue systems.

3.1 Binary

Computers are electronic machines. Essentially, everything they do involves turning electrical switches off or on. It's difficult at first to imagine that everything computers are capable of grows from this basic principle. To see how, you need to understand the concept of a binary system.

3.1.1 Binary Basics

"Binary" means having two states. In the computer's case, the two states are off or on. The trick to understanding binary is seeing how a series of these "off" and "on" states can represent any data you can think of.

Suppose a young man called Todd is a senior computer science student living off-campus in an apartment with his roommate, Stu. Todd's friend Marta has asked him to a movie this weekend, and Todd has promised to let her know if he can go after checking his schedule. Unfortunately, Todd knows that Stu will be on the phone all night with his long-distance girlfriend.

Todd and Marta solve this problem with an idea borrowed from Paul Revere. Todd has two windows in his bedroom with a lamp in front of each. By turning on one lamp, or the other, or both, or neither, he can set up one of four signals that Marta can see from her dorm down the hill.

If Todd had more lamps, he could send a greater variety of signals, as seen in figure 3.1. To send any letter of the alphabet, for example, Todd would need the ability to send twenty-six different signals. This would require five lights.

Message number	Left lamp	Right lamp	Meaning
1	off	off	Can't go, too busy
2	on	off	Can go but not until Saturday night
3	off	on	Can go to Sunday matinee
4	on	on	Can go Friday night

Lamp Signal System for Movies

off	off	off	off	off	I	A
off	off	off	off	on	2	B
off	off	off	on	off	3	C
off	off	off	on	on	4	D
off	off	on	off	off	5	E
...						
on	on	off	off	off	25	Y
on	on	off	off	on	26	Z

Lamp Signal System for Letters

Figure 3.1: Using lights to send signals



Figure 3.2: Todd and His Christmas Lights

For Christmas, Todd buys strands of outdoor lights to hang on his apartment balcony (Figure 3.2). Each strand has five lights, and he can pull them out to turn them off. Thus, each strand can be one letter in a message using the

previous letter system. If he hangs fourteen strands from his balcony, he can spell out MERRYCHRISTMAS to Marta. A few days later, Todd can change the lights to spell out another message: HAVEAGOODNYEVE.

3.2 Binary in Computer Science

If you type “Merry Christmas” into a word processing program and then erase that and type “Have a good New Year’s Eve,” the electrical patterns inside the computer are changing just like the lights on Todd’s balcony.

In computer science, the off signals are written as the number 0, and the on signals are written as the number 1. Thus, the E row in the chart above is written as 00100, because the first two and last lamps are off, and the one in the middle is on. A single on/off indicator, written as a 0 or 1, is called a “bit,” which comes from the term binary digit.

Bits are formed into bytes, which are groups of eight bits. Why eight? This number of bits allows one byte to store one character, which is any letter, digit, or other symbol that can be displayed by typing on a keyboard. Eight bits allow 256 combinations, which is more than enough for all the lowercase letters (twenty-six), uppercase letters (another twenty-six), digits 0 through 9, and special symbols like \$, %, and the quotation marks. Even the space between words is a character. When you read that some computer devices can store as much data as a novel or an encyclopaedia, that piece of information is relying on the idea that one byte equals one character. If the average word length is five characters (four letters plus the space before the next word) and a typical novel has 80,000 words, then it takes 400,000 bytes to store a novel.

Any data can be stored as binary, as long as every possible value can be matched with a whole number, which can then be turned into a binary value. Data that is already numeric, like students' test grades, inventory counts, or account balances, is easy. Textual data can be stored easily as well, as you've just seen. The process becomes tricky, though, without an obvious one-to-one correspondence between the original data and the set of binary values.

3.3 Digital Versus Analogue

Consider a room with two lamps, one on the ceiling and the other on a table. The ceiling lamp is controlled by a variable dimmer switch on the wall. When this knob is rotated clockwise, the light gets brighter. When the knob is rotated anticlockwise, the light gets dimmer. The lamp on the table also has a knob, but when it turns, it clicks into distinct positions. If the lamp is off, rotating the knob clockwise one step makes a dim light, rotating it another step makes a medium light, rotating it again makes it fully bright, and then rotating it once more turns the lamp off again.

The table lamp has four distinct positions and could easily be mimicked in binary. The number of positions for the ceiling lamp, though, is countless because the dimmer switch doesn't have distinct steps. Put another way, the table lamp has no choice between medium-bright and fully bright, while the ceiling lamp has many possibilities between any two levels of brightness.

Data with distinct values (like the table lamp) is called discrete data. Data with continuous values (like the ceiling lamp) is called analogue data.

Discrete data stored in binary form is digital. Just think of “digital” as numeric, and then “digital data” becomes “data stored as numbers”. Digital data is stored as binary patterns, but from our level of abstraction, think of those patterns as numbers.

With a set of distinct values, a binary encoding can be easily assigned to each one. With analogue data, though, the data first has to be turned into discrete data.

This process is easiest to understand when there's only one value. Consider an ordinary mercury thermometer. To read the temperature, you compare the end of the line of mercury to a scale printed on the outside of the glass. This reading is an analogue measurement. The mercury line can be any length and often doesn't line up with a mark on the temperature scale.

Now consider a thermometer with a numeric display, like a calculator. To display the temperature as a number, the thermometer has to convert the temperature from some kind of analogue measurement to a discrete one. The process of converting an analogue measurement to a number is called 'quantisation'.

This process loses some of the original data. Suppose you have a thermometer with a digital display hanging on your wall. It shows 72 degrees, but a minute later it changes to 73 degrees. You would not think that the temperature of the room changed one degree in an instant. Instead, the temperature was slowly rising, and at a point when it became closer to 73 than 72, the display changed. analogue data has a continuous range of values. When they are replaced by a finite set of values, some rounding off must occur. The difference between

analogue data and its discrete representation is quantisation error. Thus, if the temperature is actually 72.423 and the display reads 72, the 0.423 is the quantisation error. This quantisation error can be reduced by having more possible digital values. If the temperature could be displayed as 72.4, for example, then the quantisation error would only be 0.023.

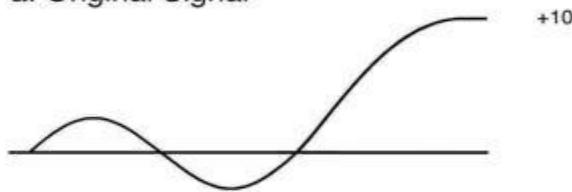
When a signal varies continuously over time, more sacrifices must be made to produce a digital version. Consider music that is changed into “digital audio”. When music plays through a speaker, the speaker’s cone moves in and out as the flow of current in the wire from the stereo goes up and down. A plotted line showing how strong the current is over time would be a continuous, if jagged, line.

The music example is a more complicated situation than the one dealing with temperature. The current has not only a continuous range of values (like the temperature) but also a changing value. To convert this kind of signal into discrete form, it requires a process called sampling. In statistics, sampling is used when an entire population cannot be queried. For instance, when 10,000 voters are asked who they will vote for in the presidential election, these 10,000 people represent, or stand in for, the whole country.

In computer science, sampling means taking analogue measurements at intervals and then quantising each measurement. With the music example, rather than trying to capture the entire original signal, the audio is sampled many times per second, and whatever signal strength is recorded at that point is turned into a number.

Just as with quantisation, the original signal is reproduced better when more samples are allowed per second. Whenever you see digital audio mentioned, it has two numbers that indicate its quality. One is the sample rate, which is the number of samples taken per second. The other is the number of bits per sample, which indicates how large a range is given to each quantised value. As these values increase, the reproduction becomes more accurate, but the resulting digital data also takes up more space.

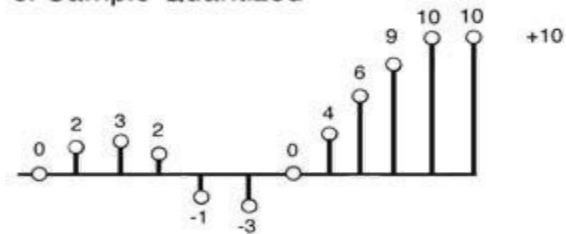
a. Original Signal



b. Sample Points



c. Sample Quantized



d. Original signal now list of numbers

0, 2, 3, 2, -1, -3, 0, 4, 6, 9, 10, 10

Figure 3.3: Sampling

In-Text Questions and Answers (ITQs and ITAs)

ITQ 3.1: How do personal computers differ from mainframes in terms of use and capacity?

Answer 3.1: Personal computers are designed for single-user tasks, while mainframes are shared by multiple users simultaneously, offering greater processing power and capacity for large-scale data handling.

ITQ 3.2: What is the significance of binary in computer operations?

Answer: Binary is the foundational language of computers, using two states (0 and 1) to represent all data and operations. Every process, from text display to complex computation, relies on these binary signals

Conclusion

Computers come in all sizes. On one end, personal computers are used by one person. At the other end, mainframes and supercomputers are used by whole organisations. Computers store data as bits, 0's and 1's. Inside the computer, bits have either high or low voltages. On a hard drive or floppy drive, there are spots on the media that are magnetised in different ways. On CDs and DVDs, there are spots that reflect light differently.

Summary

In this session, you have learnt to:

1. Explain the binary system.

2. Examine the binary basics.
3. Describe binary in computer science.
4. Differentiate between the digital and analogue systems.

Self-Assessment Questions (SAQs)

SAQ 3.1. What are the two states in a binary system?

SAA 3.1. The two states in the binary system are off (0) and on (1).

SAQ 3.2. How many lights does Todd need to send any letter of the alphabet?

SAA 3.2. Todd would need five lights to send any letter of the alphabet.

SAQ 3.3. What is a bit in computer science?

SAA 3.3. In computer science, a bit is a single on/off indicator.

represented as either 0 or 1, and it stands for binary digit.

Multiple-Choice Questions and Answers (MCQs and MCAs)

1. Which of the following is NOT a characteristic of a workstation?

- A) Used by one person at a time
- B) Connected to a network
- C) More powerful than a personal computer
- D) Always used by multiple users simultaneously

Answer: D) Always used by multiple users simultaneously

Explanation: A workstation, though powerful, is designed for single-user use, not multiple simultaneous users like a mainframe.

2. What does a binary system use to represent data?

- A) Digits 0 and 9
- B) Signals of varying frequency
- C) Two states: 0 and 1
- D) Colour-coded pulses

Answer: C) Two states: 0 and 1

Explanation: Binary uses only two states—off (0) and on (1)—to encode all computer data.

3. What is the smallest unit of data storage in a computer?

- A) Byte
- B) Word
- C) Bit
- D) Field

Answer: C) Bit

Explanation: A bit (binary digit) is the smallest unit, representing a single binary value (0 or 1).

4. Which of the following best describes quantisation error?

- A) Loss of data during compression
- B) Difference between analogue data and its digital form
- C) A system failure caused by noise
- D) Inaccurate binary representation of characters

Answer: B) Difference between analogue data and its digital form

Explanation: Quantisation error arises when converting continuous analogue signals to discrete digital values, leading to rounding differences.

5. What is the process of converting analogue data to digital form called?

- A) Encoding
- B) Quantisation
- C) Encryption
- D) Sampling

Answer: B) Quantisation

Explanation: Quantisation is the conversion of continuous analogue data into discrete digital values.

Tutor-Marked Assessments (TMAs)

- TMA 3.1.** Explain the terms mainframe, minicomputer, and supercomputer.
- TMA 3.2.** Differentiate between the digital and analogue systems.
- TMA 3.3.** Using clear examples, discuss how computers are classified according to their power and use, and analyse how these classifications influence their roles in business, education, and scientific research.
- TMA 3.4.** Explain the concept of sampling as used in computer science when converting analogue data to digital form.

STUDY SESSION

FOUR

INPUT AND OUTPUT DEVICES

Introduction

Advertisements often tout that something is “digital.” Satellite television companies, for example, offer “120 channels, all with digital-quality picture and sound!”

The truth is that digital is not always better than analogue. One main advantage of digital signals, besides being the only thing a computer can work with, is perfect reproduction. Because everything is stored as a bit, the same series of bits can be reproduced elsewhere for an exact copy of the original data. A copy of a VHS tape, which is analogue, is a little “fuzzier” than the original, while a copy of a DVD is the same as the original.

Beyond that, though, DVD has no inherent advantage. DVD recordings of movies tend to use high sample and bit rates and therefore look better than VHS tapes of the same movie. But a DVD that used a low sample (perhaps to fit a very long movie on a single DVD) might look worse than a well-made VHS tape of the same movie. Similarly, when audio CDs were first introduced to the market, the record companies would sometimes go back to the decades-old original tape masters and transfer them to CD without thinking about the process. This process meant that noise and hiss on the original tape, which weren’t audible on analogue recordings, were quite loud on the CDs. Far from sounding better, they sounded much worse than the records they replaced. Don’t assume that everything that’s digital is of high quality.

Learning Outcomes for Study Session four

When you have studied this session, you should be able to:

1. Explain the powers of 2.
2. Describe the system components.
3. Identify the input devices and output devices.
4. Explain storage devices.

4.1 Powers of 2

Because we originally counted on our fingers, we humans have a system based on the number 10. That means we work naturally with numbers like 10, 1,000, and 1,000,000. Also, that means that adding a possible digit to a number multiplies the maximum value by 10. For example, if a car has an odometer with five digits, it can display mileage up to 99,999. If we add a digit, we can display mileage up to 999,999, which is ten times as much.

Since computers use the binary system, they are naturally suited to working with numbers that are powers of 2. While adding a digit to a number allows for ten times the maximum value, adding a bit to a binary number only doubles the maximum value. That means that two bits can store four values (remember Todd's lamps), three bits can store eight values, and so on. Computers thus work best with values in the series 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, and so on, where each number is twice the previous number. These numbers are called “powers of 2.” As a consequence, computers are not described as having 250 or 500 megabytes of storage. Instead, it’s always a number like 256 or 512.

4.2 Size Terminology

Most of the world relies on the metric system, but the system hasn’t caught on in the United States except in scientific fields. The following chart lists the metric sizes that come up most often in computer science.

The first column is the prefix that’s placed in front of the unit, like “kilo” in “kilobyte.” The second column is the abbreviation for the prefix. For example, the short form of kilobyte is “kb.” Note that the capitalisation is important: M

means “mega” but m means “milli.” The third column is the general value of that prefix, which is how it is used outside of computer science. The fourth column is the value commonly used in computer science, and the fifth column provides some examples. These values are not the same because of the powers of 2. A kilobyte isn’t 1,000 bytes; it’s 1,024 bytes, because 1,024 is a power of 2. In general, though, the difference between the two values is not important. Because 1,000,000,000 is a lot easier to remember than 1,073,741,824, one can dispense with the fourth column, except when an exact value is absolutely necessary.

Prefix	Abbreviation	General Value	Computer Science Value	Sample Use
Giga	G	1,000,000,000	1,073,741,824	2.6 gigahertz
Mega	M	1,000,000	1,048,576	512 megabits
Kilo	k	1,000	1,024	256 kilobytes
Milli	m	1/1000		30 milliseconds

Metric Size Designations

Figure 4.1: Metric size designation

4.3 System Components

A computer system is more than just a computer. It's a set of components that work together to make a computer usable. Suppose you purchased a computer system today. What you would get is a collection of devices closely resembling those depicted in Figure 4.2.

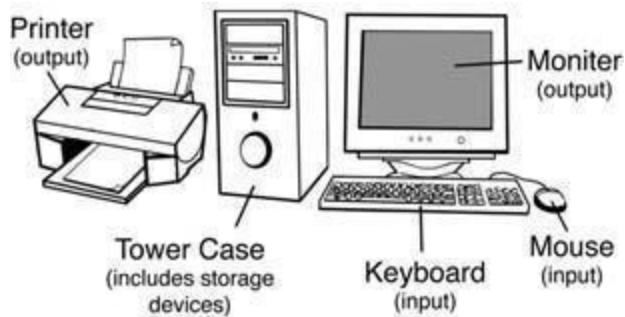


Figure 4.2: 6 Personal Computer System

The case is the box that contains the circuitry of the actual computer, along with some devices and connections for others. Some cases are wider than they are tall and are intended to sit on desktops. Others are taller than they are wide and sit on the floor. These are called towers.

In this chapter, stay at the “computer user” level of abstraction and don’t worry too much about what you can’t see inside the case. At the moment, it’s more important how the parts on the outside of the case interact with the other components.

The other components are divided into several categories: input devices, output devices, storage devices, and utility devices. An input device is used to give data to the computer. An output device transmits data from the computer back to the user. A storage device stores data for later use.

4.4 Input Devices

Common input devices include a keyboard, mouse, scanner, digital camera, and gamepad. Learn how each device works.

4.4.1 Keyboard

The most common input device is a keyboard, which is like a typewriter. The earliest keyboards had only the same keys that a typewriter does, just the letters, numbers, and other printable symbols. Over time, keyboards have gained more functionality. Most have an area on the right called a numeric keypad, which puts all the number keys together to allow fast entry of numeric data. In addition, many keyboards now have additional keys for common tasks, such as playing music or opening the user's favourite program.

While keyboards have grown more complex, the technology behind them works on simple principles. Each key is just an electrical switch. The keyboard can signal which key is pressed by assigning each key a binary pattern and transmitting that pattern to the computer.

Keyboard and Keypad Design

Have you ever wondered why the keys on a typewriter or keyboard are in an order where the letters on the top row spell QWERTY? Early typewriters would easily jam when operators typed letters next to each other too quickly. As a result, the keys were laid out so that those keys often struck in succession were not next to each other. In essence, the keys are arranged for slow typists. You can purchase keyboards with other layouts, like the “Dvorak” layout, which is intended to speed typing, not slow it down.

Take a look at a computer keyboard's numeric keypad, and then look at how the number keys are laid out on a touch-tone phone. They are opposites. On a phone, the 123 row is on the top, but on a keyboard, it's on the bottom. Why?

The phone, like the typewriter, was designed to slow down its operator because early phone systems couldn't handle a fast dialler. The numeric keypad, however, is built for speed, which is the whole point of it being there.

4.4.2 Mouse

The second most common input device is the mouse, which is a device that moves a cursor—a small arrow or other pointing shape—on the screen as it is moved. It also has one or more buttons to allow further input. Newer designs may also have a wheel that can be rolled up and down. In most programs that display pages of text, rolling the wheel moves the text up and down in unison.

Mice track their positions in multiple ways. The most common design has a rubber-covered ball on the bottom. As the mouse is slid across the desk, the ball rolls. Inside the mouse, the ball is rubbing against two rollers. If you open the compartment at the bottom of a mouse, you will see them. One roller tracks the up-down movement of the mouse; the other tracks the left-right movement. After the mouse transmits the movement of these rollers to the computer as a binary number, the computer uses this data to determine where the mouse's cursor should appear on the screen.

This design is simple but has some problems. First, the ball doesn't roll very well on some surfaces. The surface needs to be smooth enough so the ball doesn't "skip," but not so smooth that the ball slides without turning. Most users employ a square of specially designed material called a mouse pad for this reason. Second, the rolling ball tends to pick up all the dirt and lint on the

desk or pad and carry it inside, where it lodges on the rollers. The mouse's tracking gets steadily worse until it is cleaned.

An improved design is the optical mouse, which tracks the mouse position using an optical sensor, not a rolling ball. Hundreds of times per second, the mouse takes a picture of what's beneath it and, by comparing this picture to the previous one, determines how far it has moved and in what direction. Because an optical mouse has no ball, it rarely needs to be cleaned and can be used without a mouse pad.

4.4.3 Scanner

A scanner converts printed images into computer images. Some scanners can only produce black-and-white images. Their operation is similar to a photocopier. The paper to be scanned is placed on a clear sheet of glass or plastic. A bar underneath this sheet slides from one end of the paper to the other. This bar contains a bright white light and a line of optical sensors. By checking how much the light reflects back into each sensor as the bar moves, the image can be built up as a series of "grayscale" dots. Each dot is given a number. For example, 0 could be totally white, 15 could be pitch-black, and 8 could be neutral grey. These numbers are easily encoded into binary.

Colour scanners are a little more complicated but work on the same principle. Some designs have different sensors for each of the three primary colours: blue, green, and red. Other designs have only one set of sensors but make three passes across the paper. For the first pass, a red light is turned on, which makes the red in the image turn dark. Then the next pass uses a green light,

and the last pass uses a blue light. Storing a colour image takes a lot more space than a greyscale image. For each dot in the image, its level of red, green, and blue must be stored, which means that each dot is three numbers, not just one.

4.4.4 Digital Camera

A digital camera uses optical sensors to capture a photo directly in digital form. It uses the same kinds of sensors that a one-pass colour scanner uses, but instead of a line, a grid is formed. The light from the scene hits the grid, and the image is recorded.

4.4.5 Gamepad

Personal computers are often used to play games. All video game systems (like the XBox) are just single-purpose computers, anyway. A gamepad is an input device specialised for controlling games. The technology behind the gamepad is simple since it is made of a number of buttons, which are just switches.

4.5 Output Devices

Common output devices include a monitor, printer, and speaker.

4.5.1 Monitor

A monitor, the most important output device, is a computer display screen. Early monitors displayed white or a single colour on a black background, but current monitors can display as many colours as the human eye can distinguish. There are two kinds of monitors: CRT and LCD.

A CRT is a cathode ray tube monitor, which means it works like a television. At one end of the tube is a device that produces a stream of electrons. At the other end is a screen that has been coated with substances that briefly glow different colours when struck by electrons. A device that aims the electron beam scans the screen from left to right, top to bottom, and the beam is turned on or off to make some of the screen glow or not. CRT screens are inexpensive, but because the electron beam has to be shot from well behind the screen, monitors of this type tend to be very deep and thus take a lot of space on the desktop. Also, CRT screens are gently curved instead of flat, which means that the images they display appear to bend at the edges.

An LCD is a liquid crystal display. Liquid crystal, as the name implies, has properties of both a liquid and a solid. The important feature for its use in monitors is that it turns solid when exposed to an electrical current, and it is opaque when in the solid state but clear when in the liquid state. A two-colour LCD display uses a grid of liquid crystal cells in front of a bright light source, and current is sent to the cells that should be darker. More advanced displays can vary the level of transparency to make a greyscale image. For a colour display, filters are placed over the display to divide it into red, green, and blue columns.

LCD monitors were originally designed for laptop computers, where a CRT was too big to be practical. Now, they are gaining popularity with desktop computers because they take up less space and the screens are perfectly flat.

4.5.2 Printer

A printer outputs text or images on paper. Different types of printers exist for different needs. A printer used for a term paper should be fast, while a printer for photographs must print in high-quality colour.

A dot-matrix printer works by pressing an inked ribbon against the paper with a set of pins. The ribbon used is like that on a typewriter. As the print head moves across the paper, pins on the head fire out, pressing the ribbon against the paper to make a dot. These dots make up the image on the paper. As the print head reaches one side of the paper, the paper is advanced to the height of the print head, and the print head continues in the opposite direction. Dot-matrix printers, which are inexpensive to purchase and operate, are very fast, but the output quality is poor because it's easy to see the dots in the image. They are often used by companies for items like billing statements, which are printed in bulk.

An inkjet printer works by spraying ink at the paper. It is much slower than a dot-matrix printer because it takes more passes to produce the same size image, but it has several advantages. It produces a better-looking image than a dot-matrix printer because the image is not made of distinct dots. Also, some inkjet printers can print in colour, with red, yellow, and blue inks in separate reservoirs and a separate black ink reservoir, because producing black by mixing all three colours wastes a lot of ink.

Inkjet printers are inexpensive to purchase, but their cost per page of use is higher than a dot-matrix. Also, the images they produce can smear if the paper

is handled soon after printing because the ink is still wet. These are the default printers for home users and for business users who only print occasionally.

A laser printer works using electrostatic principles to transfer ink to the paper. It's not as complicated as it sounds. A drum inside the printer rotates past a laser beam that is scanning back and forth. The drum is made of a material that becomes electrostatically charged when it is struck by the scanning laser. Anyone who has taken clothes from a hot dryer knows that static electricity can make things stick together. That's how it's used here. The drum is sprayed with toner, which is a powder form of ink that has been given the opposite charge of the drum. The toner sticks where the laser hits the drum.

A piece of paper is then rolled against the drum, transferring the toner to it. If the process stopped at this point, the image wouldn't stay on the paper long because there's only static electricity holding the toner in place. The last stage in the cycle is the fuser, which is a set of very hot rollers that melt the toner into the paper. If you've ever wondered what a laser printer is doing when you turn it on and a blinking light indicates it isn't ready yet, it is waiting for the fuser to reach the right temperature.

The print quality of laser printers is excellent. Although they are more expensive than ink-jet printers for an initial purchase, the cost of ink can be less because expensive toner cartridges last much longer than ink-jet cartridges. Also, because the ink is fused into the paper, rather than sprayed on wet like an ink-jet, the image doesn't smear. In addition, laser printers are fast. They are often used for business communications, where print quality

counts, although prices have gone down enough that home users who want higher quality also purchase them.

While most laser printers produce black-and-white output, some can produce colour. These printers work by running the paper through the whole process four times, one pass each for red, yellow, blue, and black toner. Colour laser printers are very expensive. For high-quality colour printing, though, few printers can match a dye sublimation printer. “Sublimation” means going from a solid directly to a gas, such as a block of dry ice turning into a fog without melting into a puddle first. A dye sublimation printer works by heating a ribbon so that the solid ink inside turns into a gas and then seeps into the paper. The ribbon has strips of different colour inks in it to produce the different colours and has a back-and-forth print head like a dot-matrix or ink-jet printer.

Print quality is excellent with dye sublimation printers. In many cases, the result is indistinguishable from a traditional photograph. Because the ink seeps into the paper as a gas and then dries, it tends to spread out a little so that the adjacent colours mix together. In contrast, a colour inkjet printer makes little dots of different colours. If you stand away from the image, the colours mix together, but move too close, and the painting becomes just a bunch of dots. Dye sublimation produces a smoothly coloured image.

However, dye sublimation printers are very expensive. They are only needed for professional-quality photographic printing.

4.5.3 Speaker

Not all computer output is visual. Some of it is aural. Most computer systems are capable of producing sound through speakers. Computer speakers are no different in construction from those in a normal stereo system. Computers use sound to enhance or reinforce what's happening on the monitor. For example, most email programs make a little "ding" sound when a new piece of mail arrives. This sound gives the user a cue without having to interrupt the user's work. Computer games also use sound to more fully envelop the user in the game's reality.

4.6 Storage Devices

Common storage devices include a tape drive, a floppy drive, a hard drive, and an optical drive. The purpose of a storage device is to allow the writing and reading of data organised into collections called 'files'.

4.6.1 Tape Drive

The earliest computer storage devices were tape drives. A tape drive records information on tape the same way an audio or video cassette does, by creating a pattern of magnetic impulses on it. To create this pattern, the tape is dragged over an electromagnet called the "record head" or "write head," which is switched on or off to produce the desired pattern. This process leads to easy schemes for storing digital data. In the simplest arrangement, it is turned on to create a 1 bit and off to create a 0 bit. When the magnetised tape is dragged across a "read head," it reproduces the original signal in the head. In general,

the term read/write heads, or just heads, refers to the small electromagnets that produce or retrieve signals from magnetic media.

The main problem with a tape drive is that it is a sequential access device, which means it must be processed in order. There's no easy way to jump immediately to a specific piece of data. The situation is analogous to the difference between a videocassette and a DVD. If you have a favourite scene in a videocassette movie you want to watch, you have to fast-forward the tape from the beginning to that scene. You'll end up starting and stopping the tape a lot, playing a second or two of the movies to see how far you've gotten.

Most DVDs, though, have a menu that allows you to jump immediately to a particular scene. A DVD player is known as a random-access device, which means you can jump to any point without going through the previous points first. Random access is also known as direct access, which probably makes more sense.

At one time, tape drives were used as a computer's primary storage device, but that is no longer the case. Now they are used primarily for creating backups. A backup is a copy of all the data on a computer's primary storage devices, or at least all the data that cannot be easily replaced. Backups are used in case the computer's primary storage devices have a catastrophic failure or the computer is destroyed, such as in a fire. Backups are the perfect application for tape drives because tapes are cheap for the amount of storage they hold and because sequential access is not an issue.

4.6.2 Floppy Drive

A floppy disc is a flexible circle of plastic that stores data magnetically (like a tape drive). Today's floppy discs are stored in rigid shells; it's the disc inside that's "floppy," not the casing. The earliest floppy discs were ten inches in diameter, while current discs are three and a half inches; yet because they pack the data more densely, they store more overall. A floppy drive is a device to read and write floppy discs. Like a tape drive, a floppy drive has electromagnetic "heads," but in a tape drive, the head stays still and the tape is pulled past it. In a floppy drive, the disc spins, but the heads move too. The heads are on an arm that hovers over the disc and can move the heads from the inside edge to the outside edge of the disc, something like the needle arm of a phonograph, except that the heads on a floppy drive do not touch the disc. A special electric engine called a "stepper motor" is used because it can be instructed to rotate an exact number of times. This method allows the heads to be precisely placed.

A disc is organised into tracks, which are concentric circles that separate the data, like the lanes on a running track. Each track is divided into segments called "sectors." To read a particular sector, the disc is spun and the arm is moved to place the heads over the sector's track. Then the drive waits until the spinning of the disc has brought the desired sector under the head.

Floppy discs are an example of removable media (or removable storage), which refers to storage devices that can be easily taken out of the computer and taken elsewhere. Because it can go immediately to a desired sector of data, the floppy disc is a random access device. Commonly, it has been used to back

up small amounts of data and to transfer data from one computer to another. Its time may have passed, however, because storage demands have risen and networks provide easier ways to transfer data between computers. Floppy drives, which were once ubiquitous on personal computers, are no longer a standard feature.

4.6.3 Hard Drive

Hard drives are very similar to floppy drives. A hard drive, as its name implies, stores data on a rigid, magnetised disc or set of discs. Hard discs pack data much more densely than floppy discs. Because a single hard drive unit may contain multiple discs, the storage capacity is much greater. Where a single floppy disc may contain several megabytes of data, a hard drive can store 100 gigabytes of data or more, which means one hard drive is equivalent to tens of thousands of floppy discs. Hard drives are also much faster at writing and retrieving data.

To achieve this higher density and performance, hard drives spin thousands of times per minute, and use a faster and more precise mechanism for controlling the arm the heads ride on. One measure of a particular hard drive's performance is seek time, which is the length of time needed to move the heads over a specified track. Current hard drives have seek times under ten milliseconds, or, put another way, they can find a track in less than 1/100 of a second.

Although technically a "hard disc" would be a disc inside the hard drive, in common use, "hard disc" and "hard drive" are used interchangeably to refer to

the entire device, not a particular disc inside the device. This is probably because hard drives are sealed. The media is not only not removable but also can't be seen without opening the drive casing. Hard drives are sealed because the data is so precisely aligned that any dust that fell on a disc could ruin the data stored there.

Hard drives are the central storage device on computers today, and their capacity seems to double every couple of years. One problem larger drives have exacerbated is fragmentation, which is the state where files have been split up into small pieces across the disc. When a hard drive is new, the disc is one big block of unused storage, and files can be stored concurrently on the disc, perhaps all on the same track. Over time, as files are stored and erased, they split up the leftover space into smaller and smaller chunks. Eventually, individual files are too big for any one chunk and have to be split up and spread across multiple tracks. Although seek times for drives are very fast, a file can still be read much faster when it is on the same track. Having files spread all over the disc is analogous to a messy desk: although you can find what you need eventually, you have to sift through the piles to find it.

Fragmentation is one reason why computers that have been in use for a long time seem to have slowed down since they were bought. The closer a hard drive gets to capacity, the more fragmentation occurs. A program called a defragmenter carefully shuffles the file fragments around to make them all contiguous again, improving performance.

4.6.4 Optical Drive

An optical drive reads data using a laser and a light sensor. Bits are represented on an optical disc by placing bumps on the surface. The laser light reflects differently off the bumps and smooth parts of the disc to produce the binary 1 and 0 signals.

The first consumer use of optical drives was not for personal computers but for music. The availability of compact discs, or CDs, enabled consumers to purchase music in digital form, rather than analogue form as on vinyl records. Although the CD was eventually co-opted by the computer industry, its specifications were dictated by the music industry.

A music CD can hold about seventy-four minutes of music. This time was selected so it would hold as much as any single vinyl record. Because of the sample rate and bit rate chosen by the industry, this works out to a data capacity of about 780 megabytes. Unlike floppy and hard discs, a CD doesn't have concentric tracks. It contains one long spiral track, just like a vinyl record. A single track is used because a music CD needs to continuously read the disc to play the music. If it had to jump from one track to another, small gaps would appear in the music playback. The single-track causes trouble when applied to computers, though, because the CD is used as a random-access device. Finding the right spot to start reading on a CD is much slower than finding a track on a floppy or hard disc. It can take about ten times as long.

If you've ever been on a merry-go-round, you know that you go faster on the edge than you do towards the middle. To counter this effect, a CD player spins the disc slower as the laser moves to the outer part of the disc. Again, this is

done because the data needs to be read and turned into music at a constant rate.

An optical disc used for storing computer data is called a CD-ROM. A CD reader in a computer system is known as a CD-ROM drive. In both cases, ROM stands for read-only memory, which means storage data that can be read but not altered. CD-ROM drives can read and play music CDs but are mainly used to install software on CD-ROMs. Installation refers to a process where software is copied onto the hard drive and registration steps are taken to connect the software to the operating system, including making the new software appear in the operating system's "list of programs."

Early CD-ROM drives read data at the same speed as a music CD player, which, by computer standards, is very slow. Logically, this meant that reading the entire disc would take seventy-four minutes! Current CD-ROM drives can read about fifty times faster than a CD player and are able to read the outer part of the disc at about the same speed as the inner part.

While CDs and CD-ROMs are manufactured with the bumps in place, another technology allows users to make discs with their own computers. A CD-RW drive, or computer disc read/write drive, can read discs and create discs using specially made blanks. The blank discs are quite different from ordinary CDs. They contain a special dye that is initially clear but darkens when heated. A CD-RW contains a more powerful laser that is used to heat up the dye in different spots to make reflecting and nonreflecting areas that can be read like a regular CD. Because of the heat involved in the dye process, creating a CD in this way is known as burning.

Some blanks are called CD-Ws (W for write-only) and can be used only once because, after the dye has darkened, there's no way to make it clear again. Other blanks are called CD-RWs (RW for read/write), which use a substance that is heated to one of two temperatures. Heated to one temperature, that spot on the CD is reflective; when it cools; heated to another, it is opaque.

The DVD, a more recent invention, works along the same principles. A DVD is an optical disc of much greater capacity than a CD-ROM, and a DVD drive is a device in a computer system for reading DVDs. The name DVD originally stood for Digital Video.

Disc because it was developed to store movies digitally, but later the name was changed to Digital Versatile Disc because any kind of data can be stored on a DVD, just like a CD.

A basic DVD can hold about 4.7 gigabytes of data, which is about six times as much as a CD-ROM. Some DVDs have two layers of data. The bottom layer is like that on a CD-ROM—a shiny surface with smooth spots and bumps. Above that is a semi-transparent layer that's more like a CD-RW, with lighter and darker spots. Having both layers allows the capacity to almost double to around 8 gigabytes. You will sometimes see a 17 gigabyte capacity listed for DVDs—for a double-sided DVD that uses two layers on each side. Because most DVD readers can only read one side, the user has to eject the DVD, flip it over, and put it back in the reader. From a user's point of view, having a double-sided DVD isn't better than having two single-sided DVDs.

As with CDs, blank DVDs and DVD burners are also available and work on the same principles.

4.6.5 Disk or Disc?

For some reason, optical media, such as CDs and DVDs, are usually referred to as discs, with a c, although you will sometimes see the term “optical disc.” Magnetic media such as floppies and hard drives, however, are almost always referred to as discs, with a k. This is nothing more than a spelling difference reinforced by tradition. Both “*disk*” and “*disc*” simply indicate that the media are round.

In-Text Questions and Answers (ITQs and ITAs)

ITQ1.1: Why isn’t “digital” automatically better than “analog”?

ITA 1.1: Because overall quality depends on sampling and bit rates (capture/encoding choices), not merely on the signal being digital. Digital mainly guarantees **perfect copying** of the stored bits.

ITQ2.1: What key advantage does an **optical mouse** have over a ball mouse?

ITA 2.1: It uses an image sensor (no moving tracking parts), so it needs **less cleaning** and tracking more reliably on varied surfaces.

Conclusion

All data can be stored as bits, which are then called “digital data.” Some data is inherently numeric, which is easy to make digital. Other data is analogue, reflecting some continuous range of values found in the real world. Such data can be made digital, but it must be sampled and quantized first. A computer system is more than just a computer. It contains input devices (such as a

keyboard and a mouse), output devices (such as a monitor and printer), and storage devices (such as a hard drive and a CD-ROM drive).

Summary

In this session, you have learnt:

1. The size terminology of the system.
2. The powers of 2.
3. The system components.
4. The input devices and output devices.
5. The differences between keyboard and keypad designs.
6. The different types of storage devices.

Self-Assessment Questions and Answers (SAQs and SAAs)

SAQ 4.1. How does adding a bit to a binary number affect its maximum value?

SAA 4.1. Adding a bit to a binary number increases its maximum value exponentially. In binary representation, each bit has a place value that doubles as you move from right to left. The rightmost bit represents 2^0 (1), the next bit to the left represents 2^1 (2), the next represents 2^2 (4), and so on. When you add an additional bit to a binary number, you effectively increase the available place values by one. This means you can represent larger numbers because you have more combinations of 0s and 1s. Adding a bit essentially doubles the maximum value that can be represented.

SAQ 4.2. What is the relationship between the number of bits and the number of values that can be stored in a binary number?

SAA 4.2. The relationship between the number of bits and the number of values that can be stored in a binary number is exponential. Specifically, the number of values that can be represented in a binary number is equal to 2 raised to the power of the number of bits.

SAQ 4.3. Why do computers work best with numbers that are powers of 2?

SAA 4.3. Computers working with numbers that are powers of 2 can easily represent and manipulate these numbers using binary digits (bits). Binary representation aligns well with the digital circuitry in computers, allowing for efficient arithmetic operations and simpler hardware implementations.

Multiple-Choice Questions and Answers (MCQs and MCAs)

1) Which statement best explains why “digital” isn’t automatically superior to “analog”?

- A) Analog cannot be copied at all.
- B) Digital always has higher dynamic range.
- C) Quality depends on sampling/bit rates, not just being digital.
- D) Analog signals cannot store colour.

Correct Answer: C) Quality depends on sampling/bit rates, not just being digital.

Explanation: Digital enables perfect duplication of bits, but capture/encoding choices determine perceived quality.

2) Why do capacities like 256 MB or 512 MB occur frequently?

- A) They're cheap to manufacture in decimal sizes.
- B) They are powers of 10.
- C) They align with powers of 2 used in binary addressing.
- D) They reduce heat output.

Correct Answer: C) They align with powers of 2 used in binary addressing.

Explanation: Each bit doubles addressable space; capacities often land on 2^n .

3) The QWERTY layout originated primarily to:

- A) Maximise typing speed
- B) Minimise finger travel for vowels
- C) Prevent jams on early typewriters
- D) Match telephone keypad layout

Correct Answer: C) Prevent jams on early typewriters

Explanation: Frequently paired letters were separated to reduce mechanical jams.

4) Which input device converts printed images to digital form?

- A) Printer
- B) Scanner
- C) Speaker
- D) Monitor

Correct Answer: B) Scanner

Explanation: Scanners digitise documents/images via optical sensors.

5) An optical mouse tracks movement by:

- A) Measuring ball rotation
- B) Scanning surface images rapidly and comparing frames
- C) Detecting desk vibrations
- D) Using a gyroscope only

Correct Answer: B) Scanning surface images rapidly and comparing frames

Explanation: Image sensors compute delta movement without a rolling ball.

**Tutor-Marked Assessments
(TMAs)**

- TMA 1.1.** The purpose of a storage device is to allow the writing and reading of data organised into collections called files.
- TMA 1.2.** Describe four storage devices that are used for writing and reading data.
- TMA 1.3.** Critically discuss the misconception that digital signals are always superior to analogue signals.
- TMA 1.4.** Using examples of the keyboard and mouse, discuss how the design of input devices has evolved from mechanical to electronic systems. How have user needs and technological limitations influenced these design decisions over time?

STUDY

SESSION FIVE

HARDWARE

Introduction

Computer hardware consists of the physical components required for a computer system to function. It includes the motherboard, graphics card, Central Processing Unit (CPU), ventilation fans, webcam, power supply, and other components. Having looked at all the parts of a computer system, now open the case of the computer to see what's inside and to learn how it works.

Learning Outcomes for Study Session Five

When you have studied this session, you should be able to:

1. Explain the computer's main memory.
2. Describe the organisation of the hardware components.
3. Explain the process of computer operation.
4. Describe the physical characteristics of a system.

5.1 Overview

The diagram in Figure 5.1 shows the major internal components of a computer. Each of these components is discussed in detail in this study session, but for now, look at the big picture.

On the left of the diagram is the central processing unit, or CPU. This is the device that actually performs computing tasks. In a sense, the CPU is the computer. Some people even refer to the entire computer case and everything inside of it as the CPU, but that's not accurate. CPUs are often simply called processors, although processors can be used outside of the “central” role of a computer.

The processor occupies a tiny amount of space inside a computer’s case. A modern CPU sits inside a flat package about the size and shape of a saltine cracker. That’s just the package; the actual processor inside is even smaller. Because of their small size and flat, square shape, the CPU and other packaged circuits inside computers are called chips.

Next to the CPU is the main memory, which is where the computer stores programs and data while it is on.

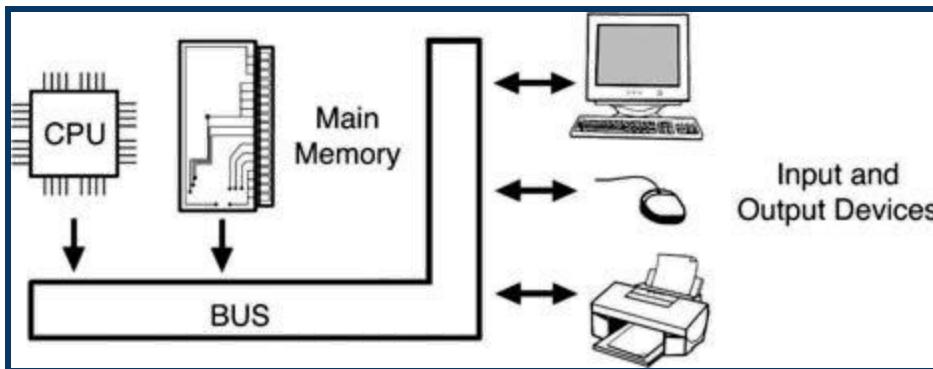


Figure 5.1: Simplified Computer Architecture

On the right of the diagram, you'll see the kinds of devices discussed in Study Session 4: keyboard, mouse, monitor, printer, hard drive, CD-ROM drive, and so on. The CPU, main memory, and all the other devices are connected by an electrical path called the bus. Like a city bus, this bus is a shared resource, too. If you were to get on a bus at First Avenue and plan to get off at Tenth Avenue, you'd have to sit through all the stops in between. Likewise, a computer bus carries traffic to and from all the devices that are connected. Mostly, this means data travelling to and from the CPU, but in some cases, data travels directly from one device to another without involving the CPU.

All this activity means that the bus can get congested, which erodes performance. Because of this, computers have multiple buses, which eases the congestion. For example, because the CPU and main memory have to communicate so frequently, there's often a bus just between them.

5.2 Main Memory

Main memory consists of RAM, which is supplemented by virtual memory.

5.2.1 RAM

The main memory in a computer is made of RAM, which stands for Random Access Memory. The term “random access” means the same thing here as in the previous chapter: that the data in the memory can be accessed in any order.

RAM is made of capacitors, which are devices that store small electrical charges. Each capacitor stores one bit. If the capacitor is mostly charged, it's considered a 1 bit, and if it's close to empty, it's a 0. There are two categories of RAM. Dynamic RAM uses capacitors that must be recharged periodically. Static RAM uses capacitors that hold their charge indefinitely. Static RAM is faster because it doesn't have to waste time with periodic recharges, but dynamic RAM is much cheaper to make. Because main memories are so large, they use dynamic RAM.

RAM is divided into cells of memory called words, which may be a single byte (eight bits) or multiple bytes. Each word's location in RAM is specified by a unique address, which is just a whole number that starts from 0. For example, if a computer system has 256 megabytes of RAM, then it has 268,435,456 bytes (remember from Study Session 4 that “mega” in this context is 1,048,576, not 1,000,000). Thus, on this system, the addresses range from 0 to 268,435,455. If a word on this system is two bytes, for instance, then only the even numbers in that range are legitimate addresses.

The capacitors in RAM are wired into grids. Think of a piece of graph paper with a capacitor wherever the lines cross. The lines on the graph would be wires called “control lines.” Although every control line has many capacitors on it, if

one vertical line and one horizontal line are selected, the lines cross in only one place. In the same way, the capacitors are wired to respond only when both of their control lines are activated.

5.3 Computer Operation

When a user requests the execution of a program, the computer brings the program from where it is stored (usually the hard drive) into main memory. Any temporary data the program needs is also stored in main memory. Storing the programs and data in main memory is necessary because storage devices like hard drives are very slow compared to the CPU. The main memory acts as a temporary “scratch pad” where the currently active program and data can be kept for quick access.

As the program executes, the CPU may need to access the devices previously discussed. For example, suppose a user opens a word processing program, such as Microsoft Word. This action brings the program into main memory. Throughout the execution of the program, data is sent to the monitor for display. Also, throughout the execution, the CPU must read the instructions of the program from main memory. Suppose the user opens an existing document for editing with Word. To do so, he or she clicks the mouse on a menu item that reads “Open,” then types the name of the file using the keyboard or clicks the file name from a list of Word files. Now the computer must access the hard drive again to load the file into main memory. The user edits the file, by using the keyboard and mouse, then prints it by accessing the printer, and then saves the file, which stores it back on the hard drive.

5.3.1 Booting

The previous discussion assumed the operating system was already running so that the user had a way to select a program to run. But what starts the operating system? At first, it seems like a chicken-and-egg problem. If the operating system is used to tell the computer what program to run next, and the operating system itself is a program, what tells the computer to run anything when it's first turned on?

The process by which a computer starts and executes the operating system is called booting. This is short for “bootstrapping,” which means to lift oneself up by one’s own bootstraps, as the expression goes, which is analogous to the seemingly impossible task the computer has when it starts.

The secret ingredient is called the BIOS, which stands for Basic Input/Output System. This is a set of small programs stored in ROM. Recall that ROM, as in CD-ROM, means read-only memory. Here it refers to memory that is accessed like RAM but is “hard-wired”; that is, it can’t be changed.

When a computer’s power is turned on, the BIOS acts like a drill sergeant, waking up the troops. It initialises all the devices in the system and performs some self-diagnostic tests. Then it performs its most important task, which is loading the operating system.

Actually, the BIOS isn’t capable of loading the operating system on its own. Instead, it retrieves a program from a special location on the primary hard drive called the boot sector. This short program is part of the operating system, and its job is to retrieve the rest of the operating system into memory.

When you turn on a computer, you often see a few seconds of plain-looking text, perhaps with a simple logo for the manufacturer of the computer. During this time, the BIOS is running. A few seconds later, the screen displays the logo of the operating system (like “Windows XP”). That’s the program from the boot sector. Finally, the full operating system is loaded, and you can begin running other programs.

5.3.2 CPU

Since the CPU is the heart of a computer, it deserves a closer look.

CPU Organisation

A CPU has three essential components: an ALU, registers, and a CU, as shown in the diagram in Figure 5.2.

The ALU, or arithmetic logic unit, performs mathematical calculations and logical operations. To the ALU, all the data in a computer is considered a number. An example of an ALU’s mathematical operation is multiplying two numbers together. An example of a logical operation is comparing two numbers to determine which one is larger.

A register is temporary storage for data. Different CPUs have different numbers of registers. While computers store most of their active data in RAM, the ALU can only directly access the registers. Thus, having more registers is an advantage.

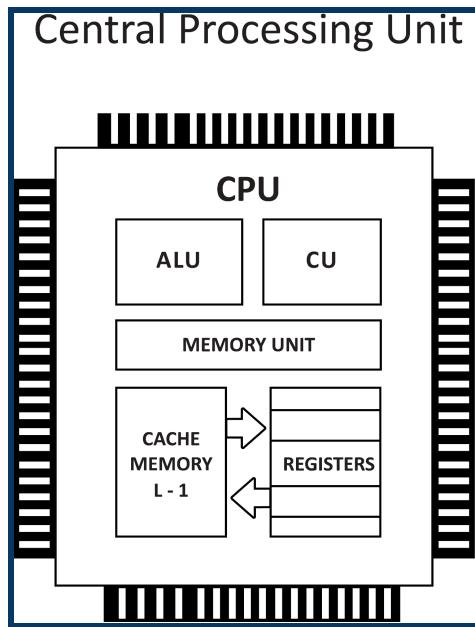


Figure 5.2: CPU Organisation

The CU, or control unit, controls the movement of data inside the CPU. It can be thought of as the CPU's traffic cop. It determines, for example, which registers are sent to the ALU for computation.

The CU is itself controlled by the program the CPU is executing. A program is made up of a series of instructions. The format of these instructions is specific to the particular model of CPU and is known as the machine language of the CPU. A single instruction either retrieves data from an address in RAM into a register, stores data from a register to an address in RAM, or sends values from two registers to the ALU to perform a computation or comparison.

The instructions themselves are stored in RAM along with the data. This means that to the CPU, an instruction is just another set of bits, another binary pattern. Having programs stored in the same memory as data is known as the stored program concept. Because the concept was popularised in a well-known article by mathematician John von Neumann, the stored program

concept was once referred to as von Neumann architecture. However, since this term gives the mistaken impression that von Neumann invented the concept, it is rarely used now, a situation similar to the ENIAC/ABC controversy. Because binary patterns are hard for the human mind to follow, when computer scientists talk about a machine language, they do so at an elevated level of abstraction called assembly language, which is a machine language written in a human-readable form. Each assembly language instruction corresponds exactly to one machine language instruction; it's just written with words and numbers instead of a bit pattern. Here's an example of assembly instructions:

move	10000, register1
move	10004, register2
add	register1, register2

Figure 5.3: Example of assembly instructions

The first instruction copies the data from address 10000 in RAM to a register in the CPU. The second instruction copies data from address 10004 in RAM to another register. The third instruction adds the contents of the two registers, and the resulting sum is stored in register 2.

When you read that a computer can perform about a billion instructions per second (as current CPUs can), keep in mind how small these instructions are. Simply retrieving two pieces of data and adding them together takes three machine language instructions, as shown above. Actions that seem trivial to the user, such as opening a window in the middle of the screen to say, “Are you sure you want to quit without saving your work?” require the execution of countless instructions on the CPU.

The heart of CPU operation is the fetch-execute cycle. The CPU endlessly repeats the following steps:

1. First, the next instruction is retrieved from memory.
2. Then, the control unit decodes the instruction, which means breaking apart the binary pattern of the instruction into pieces to determine how to execute the instruction.
3. Finally, the instruction is executed

5.2 Physical Characteristics

How does a CPU actually compute? Although from our point of view the CPU is capable of adding, multiplying, and so on, from the CPU's point of view it is not manipulating numbers, but rather patterns of bits (on/off values) that represent numbers.

The basic bit manipulation control is called the transistor, which is a device that allows one flow of current to control another. Again, using lamps as an example, imagine that you have a lamp with its own on/off switch plugged into a power socket, and that socket is controlled by another switch on the wall. If the lamp switch is “on,” you can turn the lamp on or off by using the switch on the wall. If the lamp switch is “off”, it doesn’t matter how the switch on the wall is set; the lamp will be off.

This arrangement produces what’s called the “And operation”: the lamp is on only if both switches, the wall switch “and” the lamp switch, are on. By wiring multiple transistors together in different combinations, more complicated logical operations can be performed. A particular way to connect a group of

transistors is called a logic circuit, which can be combined to form even more complicated logic circuits. As you might expect, it takes a lot of transistors to accomplish useful tasks. Simply adding two bytes representing two whole numbers together requires over 100 transistors. A modern processor contains millions of transistors. Early computers used vacuum tubes as transistors. Now, transistors are made of silicon and metal. A modern CPU is built in layers, like a very flat wedding cake. Different layers have different patterns of metal running through them. The way the layers are put together, every line of metal in one layer that crosses over a line of metal in another layer results in a transistor, with the flow of power in one line controlling the flow in the other. The design of CPUs is so complicated that it takes a powerful computer to help design them.

5.2.1. Cache

While all that is necessary to make a CPU are the ALU, CU, and registers, most CPUs devote a large percentage of their tiny real estate to another set of components, called the cache. This is fast-access memory that is used for faster retrieval than is possible from main memory. The cache would be equivalent to a file you keep on your desk, rather than putting it back in the filing cabinet, because you think you'll use it again soon.

Cache access is faster than main memory access for two reasons. First, because the cache memory is inside the CPU, the data doesn't have as far to travel as data from main memory. And second, cache memory is made from faster static RAM, not dynamic RAM.

Cache is essential to the performance of current CPUs. Retrieval from main memory is too slow relative to the CPU's clock speed. If the CPU had to wait for every piece of data to come from RAM, it would spend most of its time idling. Because early CPUs did not have caches, the speed of early computers was limited by the speed of main memory access, not the power of the CPU. This weakness was called the von Neumann bottleneck, because it was an inevitable consequence of the von Neumann (stored program) architecture.

For cache to be effective, however, the right data has to be in the cache when the CPU needs it. Because cache can only hold a fraction of the data in main memory, the CPU has rules to determine which items in RAM are to be held in cache at any given time. These rules are based on the principle of locality, which says that when a particular address in memory is accessed, it's likely that nearby addresses will be accessed soon. Using this principle, when an address is requested and the data is not currently in cache; the CPU retrieves the requested address and all the addresses around it in a block and puts them all in cache.

In-Text Questions and Answers (ITQs and ITAs)

ITQ 5.1: Why is the CPU often referred to as the “heart” of the computer system?

Answer: The CPU performs all computing tasks by executing instructions and processing data. It coordinates all components, making it central to the computer’s operations—hence its description as the system’s “heart.”

ITQ 5.2: What distinguishes dynamic RAM from static RAM?

Answer: Dynamic RAM must be refreshed periodically because its capacitors lose charge, whereas static RAM retains data without refreshing. Static RAM is faster but more expensive, while dynamic RAM is cheaper and used in main memory.

Conclusion

Internally, a computer has a CPU, main memory, and connections to devices, all linked by a set of buses. Programs and data are stored in the main memory, which is made of RAM. The CPU can access any bytes of data inside RAM using a unique numerical address.

When a computer is turned on, it goes through a process known as booting, in which the BIOS reads a small part of the operating system from the boot sector of the hard drive. A small part of the operating system loads the rest of the operating system into memory.

The basic components of a CPU are the ALU, the CU, and registers. The CU's job is to execute the instructions of a program, which involve moving data into and out of registers and sending data to the ALU for mathematical operations and comparisons. CPUs also contain a cache, which is high-speed memory that holds a subset of RAM so it can be accessed quicker.

According to Moore's Law, CPUs get twice as complex every two years. Having more transistors means more heat, but computers have increasingly elaborate schemes for dissipating heat away from the CPU.

Summary

In this session, you have learnt to:

1. Explain the computer's main memory.
2. Describe the organisation of the hardware components.
3. Explain the process of computer operation.
4. Describe the physical characteristics of a system.

Self-Assessment Questions and Answers (SAQs and SAAs)

SAQ5.1. What is the purpose of a cache in a CPU?

SAA 5.1. The cache in a CPU is a small, high-speed memory that stores frequently accessed data and instructions. It acts as a buffer between the CPU and main memory, providing faster access to frequently used data. The cache helps improve CPU performance by reducing the time required to fetch data from slower main memory.

SAQ5.2. Why is cache memory faster than main memory?

SAA 5.2. Cache memory is faster than main memory because it is built using faster and more expensive memory technologies, such as SRAM (Static Random-Access Memory). SRAM can access data much more quickly than the dynamic RAM (DRAM) used in main memory. Additionally, cache memory is physically closer to the CPU, reducing the time required for data to travel between the CPU and memory.

SAQ 5.3. How are instructions stored in RAM, and what is the advantage of the stored program concept?

SAA 5.3. Instructions are stored in RAM as binary codes that represent specific operations to be performed by the CPU. They are typically organised sequentially, with each instruction occupying a specific memory location. The advantage of the stored program concept is that it allows instructions to be stored in memory alongside data, enabling a computer to execute a wide range of programs by simply loading different sets of instructions.

Multiple-Choice Questions and Answers (MCQs and MCAs)

1. Which component is responsible for carrying out arithmetic and logical operations in the CPU?

- A) Control Unit
- B) Main Memory
- C) Arithmetic Logic Unit
- D) Cache

Answer: C) Arithmetic Logic Unit

Explanation: The Arithmetic Logic Unit (ALU) performs all mathematical and logical computations within the CPU.

2. What is the main function of the control unit in a CPU?

- A) Storing instructions
- B) Controlling data movement
- C) Performing arithmetic operations

D) Increasing cache size

Answer: B) Controlling data movement

Explanation: The Control Unit directs the flow of data and instructions within the CPU, coordinating the actions of other components.

3. Which of the following best describes virtual memory?

- A) Temporary memory within the CPU
- B) Extra memory created on a storage device to supplement RAM
- C) Permanent storage of data
- D) Memory used only for BIOS operations

Answer: B) Extra memory created on a storage device to supplement RAM

Explanation: Virtual memory extends physical RAM by using part of a storage device to simulate additional memory capacity.

4. During booting, what is the main role of the BIOS?

- A) Execute user programs
- B) Manage virtual memory
- C) Initialise hardware and load the operating system.
- D) Connect devices to the bus.

Answer: C) Initialise hardware and load the operating system.

Explanation: The BIOS initiates system checks, initialises devices, and loads the operating system from the boot sector into memory.

5. Which statement correctly differentiates cache memory from main memory?

- A) Cache is slower than RAM

- B) Cache stores permanent data
- C) Cache provides faster access and is located within the CPU
- D) Cache is used for long-term storage

Answer: C) Cache provides faster access and is located within the CPU

Explanation: Cache memory is high-speed storage inside the CPU that enables quicker data access compared to main memory (RAM).

Tutor-Marked Assessments
(TMAs)

- TMA 5.1.** A CPU has three essential components. Describe these three essential components.
- TMA 5.2.** Explain the concept of Von Neumann bottleneck.
- TMA 5.3.** Describe the principle of locality of reference.
- TMA 5.4.** Describe the **fetch-execute cycle** of a CPU and evaluate how the **control unit (CU)** and **arithmetic logic unit (ALU)** cooperate to perform program instructions.

STUDY SESSION

SIX

CHALLENGES OF MODERN CPU

Introduction

The central processing unit (CPU) is the most important component of a digital computer system. It consists of the main memory, the control unit, and the arithmetic-logic unit. It is the physical core of the entire computer system, to which various peripheral devices, such as input/output devices and secondary storage units, are connected. In modern computers, the CPU resides on a microprocessor-containing integrated circuit chip. The control unit of the central processing unit coordinates the computer's operations. It selects and retrieves instructions from the main memory in the correct order and interprets them so that the other functional components of the system are activated at the proper time to perform their respective operations. All input data are transferred from the main memory to the arithmetic-logic unit for processing, which includes the four basic arithmetic operations (addition, subtraction, multiplication, and division) and certain logic operations, such as data comparison and the selection of the desired problem-solving procedure or a viable alternative based on predetermined decision criteria.

Learning Outcomes for Study Session Six

When you have studied this session, you should be able to:

1. Explain the challenges of modern CPUs.
2. Describe the problem of comparing CPUs.
3. Explain the hardware devices.
4. Differentiate between the motherboard, graphics card, and sound card.
5. Explain various types of device interfaces.

6.1 Challenges of Modern CPUs

In addition to cache being essential to the performance of current CPUs, designers of modern CPUs face challenges that weren't faced in the early days of computing, as explained below.

6.1.1 Backwards Compatibility

Most personal computers sold today are descendants of the IBM PC and probably contain a CPU called a Pentium IV, which is a specific make of CPU produced by Intel Corporation. However, if you purchased a personal computer with an Intel CPU two years ago, it would have contained a Pentium III; before that, a Pentium II; before that, the original Pentium; and before that, another processor called the 486; and before that, the 386 processor; and so on.

When Intel introduced its Pentium IV CPU, it was sold mainly to people who already owned a computer with a previous Intel CPU, but now wanted one that was faster. They were willing to purchase a new computer, but only if their existing software would work on the new CPU. Intel, therefore, had to ensure that all programs that would execute on the previous processors would execute on the new one. This ability is known as backward compatibility.

Without backwards compatibility, computer progress would be difficult because the cost of upgrading a system would be very high. While this makes the user's life much easier, it restricts the CPU designer's freedom and makes his or her job much harder. For example, the registers on early Intel CPUs contained 16 bits, or 2 bytes, while later Intel CPUs required 32 bit registers.

The solution was to design the registers on later CPUs to act as both 16 bit and 32 bit registers, depending on what size data was sent to them.

6.1.2 Moore's Law

Back in 1965, researcher Gordon Moore wrote an article predicting that the number of transistors in a typical processor would double every two years. The unstated implication behind this prediction, which became known as Moore's Law, is that performance would also double every two years. To Moore's credit, this prediction has been proven correct over the last thirty years.

But this increase cannot continue forever. To add more transistors, either the CPU must get bigger or everything on it must get smaller. Increasing the size of the CPU is not desirable because the larger the CPU is, the more it costs to manufacture, and larger CPUs make the current flow a longer distance, which decreases performance when the goal of adding transistors is to increase performance.

As a result, the circuits have become smaller, with narrowing circuit lines approaching 0.09 microns, or 0.0000035 inches. At this level, the slightest imperfection in materials during the processor's creation ruins it. If you read about a CPU maker's problems with "yield," this term refers to the number of processors that are usable versus the number that have to be thrown away because of these flaws.

Soon the point will be reached where the circuit lines are only a few atoms across, but technology can only go so far in this direction.

Some researchers are working on designs based on quantum physics, where computers will store information in the quantum state of the atoms themselves. Some amazing properties of quantum mechanics, if harnessed, would allow computations to be performed much faster than with current designs.

6.1.3 Pipelining

The description of the basic fetch/execute cycle implies that the CPU fetches one instruction, decodes it, executes it completely, and then goes back for another. At one time, this basic cycle was the norm, but eventually another scheme had to be employed to allow the execution of instructions to overlap.

Remember Todd, the college student sending messages with Christmas lights? Let's assume that Todd has had enough of Stu, his phone-hogging roommate, and decides to get an apartment of his own. He enlists a bunch of his friends to help move him into his new place.

At first, the move isn't well organised. He backs a truck up to the front door of the apartment building and asks his friends to grab stuff out of the truck and carry it inside. However, the building is old, with hallways and staircases so narrow that Todd's friends are running into each other as they dart from the truck to the apartment and back. They decide to take turns carrying, rather than everyone unloading at once.

The problem is, if they take turns carrying items into the apartment, the move will take all day. One person can make the round trip only so fast. So, Todd devises a method to utilise his free labour more effectively. Instead of asking

any one person to carry something all the way from the truck to the apartment, he staggers them twenty feet apart along that path. Each person now only has to carry an item twenty feet, and no one crosses paths with anyone else.

In other words, instead of only one item being moved at once, as many items can be moved as there are people to move them. Although the first item on the truck doesn't get to the apartment any faster, the second item comes right behind it, and the third right behind the second, so the overall speed of the move is greatly increased.

In computer science, this idea of breaking up instructions into smaller pieces so that their execution can overlap is called pipelining. Just as with Todd's movers, there is a limit to how fast one instruction can be executed on a CPU. Because the logic circuits are long chains of transistors, it takes time for the power to flow from one end to the other. Using pipelining, one large circuit is broken into different stages, and once a result comes out of one stage, that stage can be used for another instruction. CPUs today divide execution into a dozen or more stages.

But although pipelining boosts performance, it does have drawbacks. Consider Todd's situation again. Suppose, in addition to his own stuff, he's got several boxes of things that belong to his friend Marta. He intends to leave these in the truck and drive them over to her place when he's done moving. But he forgot to mention this to the friends helping him move. At some point, one of Marta's boxes arrives at the front of the line. Once Todd recognises it, the progress of the move comes to a halt while that box is returned to the truck.

A similar problem occurs in pipelined CPUs. The CPU executes some instructions conditionally. That is, it executes an instruction comparing two numbers, and then the result of that comparison determines which of two different sets of instructions is executed next. When a pipelined CPU reaches a comparison, it has to guess how the results will go because the result won't be known until the comparison instruction reaches the front of the pipe. If it guesses wrong, it will have a pipeline full of partially executed instructions that must now be discarded. Then, the correct sequence of instructions is begun at the back end of the pipeline. If this happens too often, performance is degraded.

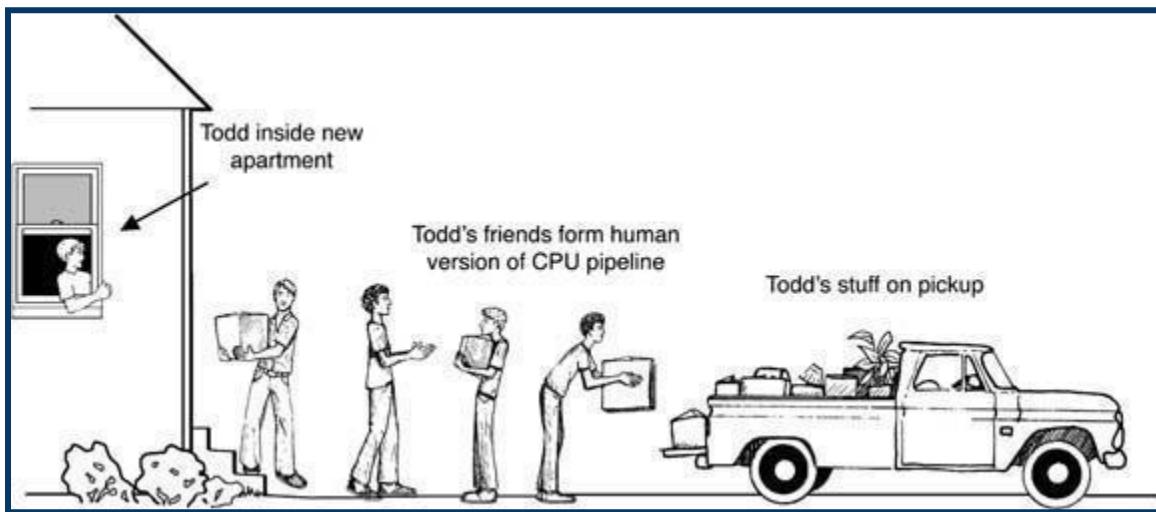


Figure 6.1: Pipelining

CPUs with pipelines, therefore, have sophisticated logic, called branch prediction, to help the CPU guess how comparisons turn out. One technique used is to track how the comparison turns out, and if the same comparison comes up again soon, the CPU guesses it will turn out the same way. But the

technique means the CPU must somehow temporarily store the results of comparisons, which further complicates its design.

6.1.4 Heat Dissipation

If you opened the case of a computer system, you probably would not be able to see the CPU package. That's because the CPU is covered by two other devices. Immediately on top is the heatsink, which is just a metal block with fins that draw away heat from the CPU. Between the heatsink and the CPU, there is a special paste that helps fill the gap between them completely and creates effective heat transfer. And on top of that is a small electric fan.

Why so much trouble? CPUs generate an enormous amount of heat. As they have become more complicated, all the components on a CPU have had to shrink, and that shrinkage causes more heat—similar to how water flows harder through a narrower pipe. If the heat wasn't conducted away from the CPU, it would quickly melt itself. In addition to the heatsink and fan, the CPU commonly has a thermal sensor attached. If the computer senses that the CPU is about to overheat, it can shut down the system.

With some computers, even this arrangement is not enough. The hottest CPUs use a liquid-cooling system. Some work like a radiator in a car, with a pump to run coolant past the CPU to collect the heat and then through a series of channels where the heat is released into the air. Other systems immerse the entire circuit board in a non-conductive liquid.

6.1.5 Multiprocessing

One way computer designers avoid these problems is to divide the work among multiple processors. Multiprocessing is the use of multiple CPUs on the same computer.

Originally, multiprocessing meant two separate physical CPUs, which were actually two different chips. Now, CPU designers are developing the dual-core processor, which combines two logically independent processors on the same chip. The advantage of multiprocessing is that it increases the performance of a computer system without increasing the speed at which one CPU executes.

The performance increases without the increase in heat.

However, multiprocessing provides only a limited benefit. One would hope that two processors would double the performance of one processor, but this is rarely the case because not all computer tasks are easily divided.

For example, if you were asked to bake ten dozen cookies for a bake sale, you could call a friend and ask him or her to make five dozen cookies, and you would make the other five dozen. Having two cooks in two kitchens would allow the cookies to be baked in half the time. But if you were asked to make a single, giant lasagna, there would be no benefit in asking for help. In the same way, some computer tasks can be easily shared among multiple processors, but some cannot.

6.1.6 The Problem of Comparing CPUs

Different makes and models of CPUs have different characteristics, but it is not easy to find the right criteria for comparing them.

6.1.7 CISC vs RISC

Early CPUs had machine languages with very simple instructions. They might not even be able to multiply two numbers as a single instruction. Instead, many “add” instructions would have to be executed. Over time, as CPUs became more complicated, so did their machine languages. Single instructions were added to do the work of multiple instructions in previous CPUs. For example, retrieving two numbers from memory and multiplying them could be one instruction.

These CPUs with large numbers of more powerful instructions become known as CISCs, or Complex Instruction Set Computers. Having more powerful instructions would seem like an advantage, and it is, but it has strong disadvantages as well. The main problem is that the logic circuits for executing an instruction must be more complicated as well, which means they take up more space on the chip. This in turn leaves less space on the chip for other things, like registers and cache, which improve the CPU’s performance.

Because of these problems, CPU designers started making instructions as simple as they could. With the saved space, they added large register sets and caches. This meant that programs had to execute more instructions to accomplish the same tasks as CISCs, but because each instruction could execute faster, the overall performance was better. CPUs that use a small set of fast-executing instructions are called RISCs, or Reduced Instruction Set Computers.

The RISC idea can be said to have won for now, and most processors can be said to be more RISC-like than CISC-like. Those that retain CISC features do so

mainly for backwards compatibility. Why is RISC faster? Because the design executes instructions faster, the only way for CISC to keep up is if the number of instructions per program can be dramatically reduced. This isn't possible because the most common instructions used in programs are the simplest ones. The most complicated instructions on a CISC machine—the ones that do the work of many RISC instructions—are so rarely used in actual programs that they make little difference.

6.1.8 Clock Speed

When a computer is advertised, a number like 2.8 GHz is often displayed prominently at the top of the feature list. This number is the clock speed, which is the number of CPU cycles per second. The Hz indicates "Hertz," which means cycles per second. The G means "Giga," which, as discussed in Chapter 1, means 1 billion. Thus, a 2.8 GHz processor has an internal clock that pulses nearly three billion times a second, and the actions of the CPU are triggered by those pulses.

If a CPU executed one machine language instruction per cycle, the clock speed would directly give the number of instructions per second, but because of pipelining and other complications, these numbers are never the same. The clock speed is most useful when comparing two CPUs of the same make and model. An Intel Pentium IV executing at 2.8 GHz is a little faster than the same Pentium IV executing at 2.6 GHz.

Clock speed is not reliable, however, when comparing different model CPUs. AMD, another chip maker, has a CPU called the Athlon XP. When this chip is

clocked at 2.2 GHz, it executes about the same number of instructions per second as a 2.8 GHz Pentium IV.

AMD chooses not to identify their processors by clock speed because their processors would appear to the consumer as slower than they really are. Intel is planning to follow suit because it is offering many variants of the same chip now. Having multiple Pentiums with the same clock speed available won't help consumers pick the right one for their needs. Instead, both companies will use artificial model numbers that have nothing to do with clock speed.

6.1.9 Benchmarking

Still, the question remains open: How does one compare one CPU's performance against another if clock speed isn't the answer? The best solution is benchmarking, which in computer science means running the same program on multiple computer systems and timing the results. If both computers are as identical as possible except for the different CPUs, then the difference in overall performance of the system should be attributable to the difference in processors.

Unfortunately, even benchmarking is not an absolute guide. CPUs have different strengths and weaknesses. Just because CPU A is ten percent faster than CPU B using one particular program doesn't mean it will be ten percent faster on all programs. CPU A may even be slower on other programs. Still, benchmarking is the most reliable tool for comparing real-world performance, especially if the benchmarks are performed using the kind of software the user is interested in running.

6.2 Hardware Devices

Now that you've seen how the computer functions, take a closer look at some of the other internal components.

6.2.1 Motherboard

The components inside a computer are located on a large circuit board called the motherboard. If you looked at a motherboard, you would see the CPU, main memory, RAM, and connections for other devices like the hard drive and CD-ROM drive. You would also see circuit traces connecting these components. These are the buses.

You would also see other chips soldered onto the board, resembling mini CPUs. They might even have their own heatsink and fan. These other chips are device interfaces.

6.2.2 Graphics Card

The CPU does not directly generate the images that are displayed on the monitor. Instead, a specialised device called a graphics card or display adapter produces the images. These are small circuit boards that fit into a slot on the motherboard.

Because a video screen is constantly refreshed, the image displayed must be stored in RAM. Originally, graphics cards were just the RAM to hold the image and a device that constantly scanned through the RAM and converted the image stored there into the signals the monitor needed for display. Now, graphic cards are much more complicated.

Computers, especially those used for entertainment, have to generate incredibly complicated images and do so many times a second. Today, graphics cards are almost complete computers in miniature. They have their own large banks of specialised RAM, sometimes almost as much as main memory. They have their own processors, which are generally more complicated, in terms of the number of transistors, than the CPUs in the same system. Because these processors get very hot, they too have heatsinks and fans. Some even draw so much power that they need a special direct connection to the computer's power supply.

6.2.3 Sound Card

As with video, the CPU doesn't generate sound directly. Instead, the audio you hear is produced by a sound card, which is a circuit board that fits into a slot on the motherboard. Because users are often less picky about sound than video, some motherboards have the sound-producing circuits directly hardwired onto the motherboard.

The heart of a sound card is a digital-to-analogue converter. This converter performs the task of modulation, which is the opposite of sampling. It takes numbers and converts them back into continuous signals. These are the signals sent to your speakers or headphones.

6.2.4 Device Interfaces

Different devices have different methods of communication with the computer. A particular device's method of communication is called its interface. Many of the "extra" chips on a motherboard are actually translators

for a certain kind of interface. It's as if the CPU only speaks English, but the hard drive and CD-ROM speak Russian, the keyboard speaks German, and the mouse speaks a dialect of Spanish.

Having different interfaces may seem like an inefficient way to make a computer. Why not have everyone agree on the same language? There are two reasons why this hasn't happened. First, the interfaces are optimised for the needs of the devices that use them. A hard drive has to transmit a tremendous amount of data in a short time, but the keyboard sends only a handful of bytes to the computer every second, even for the fastest typists. Any interface fast enough for the hard drive would be serious overkill for a keyboard. Second, just like with CPUs, designers have to be aware of backwards compatibility. When someone introduces a new keyboard interface that requires users to purchase new keyboards, even though the old interface and keyboard worked fine, users will complain.

Some common interfaces used in computers today include USB, Firewire, IDE, and SCSI.

6.2.5 USB

USB, which stands for Universal Serial Bus, is an interface for external components, like mice, keyboards, and printers.

The word "serial" means that USB uses serial transmission, in which bits are sent one at a time along the same wire. In contrast, parallel transmission sends bits in groups, like one byte at a time, by having a wire for each bit that is sent.

Parallel transmission would seem to be faster, but because it is more complicated, the overhead often outweighs the benefits.

The USB interface can transmit at a rate of 12 Mbps. M means “Mega,” bps means “bits per second,” and thus 12 Mbps means about 12 million bits per second, or 1.5 million bytes per second.

USB allows chaining, which means multiple devices can share the same interface. In the case of USB, up to 127 devices can be chained to one interface. That is, if a computer has one USB socket on the back of the case, up to 127 devices can communicate with the computer through that one socket. This chaining means the devices form a bus, just like the ones on the motherboard, only one that can change over time.

To allow for the physical connection of so many devices, some larger devices, like printers, may have both an outgoing socket (to connect to the computer) and an incoming socket (for another device to plug into and continue the chain). Because some devices, like mice, won’t have any incoming sockets, users can purchase USB “hubs,” which are nothing more than a set of extra incoming sockets.

One special feature of USB is that it allows hot-swapping, which means devices can be safely plugged in or unplugged while the computer is running. The USB interface can alert the computer’s operating system when a device is connected or disconnected from the system.

6.2.6 Firewire

Firewire is another type of high-speed serial bus. Developed by Apple, it is also known as IEEE 1394, after its official standard designation.

It's much faster than USB, transmitting data up to 400 Mbps. This speed is necessary because Firewire's intended use is connecting computers with video devices like camcorders and digital cameras, where video data is large. The speed of the interface is also good for devices like portable hard drives.

Like USB, Firewire allows hot-swapping and can chain up to sixty-three devices.

6.2.7 IDE

IDE, or Integrated Drive Electronics, is a standard interface for connecting hard drives and optical drives. The name was chosen originally because the hard drive contains its own interface hardware, whereas previous hard drives had separate interface "cards" that had to be installed in the computer first.

IDE has been superseded by EIDE, or Enhanced IDE, which allows for faster data transmission and larger hard drives.

A single IDE interface on a motherboard, known as a port, can support two drives. If two drives are on the same port, one must be designated a "master" and the other a "slave." Usually, this is done by setting switches on the drives themselves. Because many computers have more than two drives, motherboards using IDE have at least two ports.

6.2.8 SCSI

SCSI (pronounced “scuzzy”) stands for Small Computer System Interface and is another standard interface for hard drives and optical drives. Note that drives are built for particular interfaces. A hard drive made for IDE will not work on a SCSI interface.

In general, SCSI interfaces and drives are more expensive than their IDE counterparts, but they perform better. SCSI interfaces can support up to seven devices.

Both SCSI and IDE use parallel transmission. If you look inside a computer case, you will probably find that the hard drives are connected to the motherboard using a ribbon cable, which, like it sounds, is flat like a ribbon. If you look closely at a ribbon cable, you will see that it is ribbed like corduroy. Each of those ribs contains a wire to allow the parallel transmission of data.

In-Text Questions and Answers (ITQs and ITAs)

ITQ 6.1: Why is backward compatibility important for modern CPUs?

Answer: Backward compatibility allows new CPUs to run software designed for older CPUs. This reduces the cost of upgrading systems and ensures continuity in using existing programs, which makes it easier for consumers to upgrade their hardware without losing access to old software.

ITQ 6.2: What is Moore's Law, and why does it present challenges for modern CPUs?

Answer: Moore's Law predicts that the number of transistors in a processor

will double every two years. While this has been true for decades, the challenge lies in the physical limits of shrinking circuits as they approach atomic scales, which may cause manufacturing difficulties and limit further progress.

Conclusion

Different makes and models of CPUs are difficult to compare. The listed clock speed for a CPU may be misleading. The best bet is to run the same programs on computers with different CPUs and record the results.

Computer components are linked together on a motherboard. In addition to the basics of CPU and RAM, the motherboard provides interfaces for all the other devices in the computer system. Each type of device uses a different interface. External devices like a mouse or a digital camera might use USB or Firewire. Storage devices commonly use the IDE or SCSI interface.

Summary

In this session, you have learnt to:

1. Explain the challenges of modern CPUs.
2. Describe the problem of comparing CPUs.
3. Explain the hardware devices.
4. Differentiate between the motherboard, graphics card, and sound card.
5. Explain the device interfaces.

Self-Assessment Questions and Answers (SAQs and SAs)

SAQ 6.1. What is the concept of backwards compatibility, and why is it important for CPU designers?

SAA 6.1. Backwards compatibility refers to the ability of a newer system or component to work with or support older versions or systems. In the context of CPU designers, it means that a new CPU should be able to run software or applications designed for older CPUs without any compatibility issues. This is important because it allows users to upgrade their hardware while still being able to run their existing software, ensuring a smooth transition and preserving software compatibility.

SAQ 6.2. How does Moore's Law relate to the advancement of CPUs over time?

SAA 6.2. Moore's Law is an observation made by Gordon Moore, co-founder of Intel, which states that the number of transistors on a microchip doubles approximately every two years, leading to an exponential growth in computing power. This observation has been used as a guideline for the advancement of CPUs over time. As transistor counts increase, more complex and powerful CPUs can be designed, resulting in improved performance, increased capabilities, and enhanced efficiency.

SAQ 6.3. What is pipelining, and how does it improve CPU performance? What are the drawbacks of pipelining?

SAA 6.3. Pipelining is a technique used in CPU design to improve performance by overlapping the execution of multiple instructions. It breaks down the execution of instructions into a series of sequential stages, and multiple instructions are executed simultaneously but at different stages of the pipeline. This allows for higher throughput and increased instruction-level parallelism. The main benefit of pipelining is improved CPU performance and efficiency. However, there are some drawbacks, such as the possibility of pipeline stalls, dependencies between instructions, and branch prediction errors, which can reduce the overall performance gain.

Multiple-Choice Questions and Answers (MCQs and MCAs)

1. What is the primary challenge that backward compatibility presents to CPU designers?

- A) Designing new CPUs to be slower than older models
- B) Maintaining compatibility with old software while improving performance
- C) Creating larger CPUs
- D) Designing CPUs that do not require upgrades

Answer: B) Maintaining compatibility with old software while improving performance

Explanation: Ensuring that new CPUs can run older software while improving

performance restricts the design freedom of CPU manufacturers, making the task more complex.

2. What is Moore's Law?

- A) The number of transistors on a CPU will double every year.
- B) The processing speed of CPUs will double every year.
- C) The number of transistors on a CPU will double every two years.
- D) The physical size of CPUs will double every two years.

Answer: C) The number of transistors on a CPU will double every two years.

Explanation: Moore's Law predicts that the number of transistors on a chip doubles approximately every two years, leading to increased computing power.

3. Which of the following is a disadvantage of pipelining in CPUs?

- A) It speeds up execution by executing multiple instructions at once.
- B) It requires more space on the CPU.
- C) It can lead to a performance drop if instructions are mispredicted.
- D) It allows for single-instruction execution.

Answer: C) It can lead to a performance drop if instructions are mispredicted.

Explanation: Pipelining improves performance but may suffer when the CPU incorrectly guesses the outcome of conditional instructions, resulting in wasted resources.

4. Why is heat dissipation crucial for modern CPUs?

- A) To improve processing speed
- B) To prevent the CPU from overheating and causing damage

- C) To decrease the power consumption of the CPU
- D) To make the CPU more portable

Answer: B) To prevent the CPU from overheating and causing damage

Explanation: Modern CPUs generate a significant amount of heat, and without proper heat dissipation, they would overheat, potentially damaging the system.

5. What is the role of the heatsink in a CPU?

- A) To add extra power to the CPU
- B) To convert electrical energy into heat
- C) To draw heat away from the CPU to prevent overheating
- D) To make the CPU cooler in temperature

Answer: C) To draw heat away from the CPU to prevent overheating

Explanation: The heatsink is a metal block designed to absorb and dissipate heat away from the CPU, preventing it from reaching temperatures that could damage it.

Tutor-Marked Assessments (TMAs)

- TMA 1.1.** Explain Moore's law.
- TMA 1.2.** Differentiate between CISC and RISC.
- TMA 1.3.** Explain the concept of pipelining technique with the aid of a diagram.
- TMA 1.4.** Explain how backward compatibility both supports technological continuity and limits CPU design innovation.

STUDY SESSION

SEVEN

SOFTWARE

Introduction

Software is a collection of instructions, data, or programs used to operate and execute specific tasks on a computer. It is the opposite of hardware, which describes a computer's physical components. Software is an umbrella term for applications, scripts, and programs that operate on a device. Software is the variable component of a computer, while hardware is the fixed component.

Learning Outcomes for Study Session Seven

When you have studied this session, you should be able to:

1. Explain the definition of software.
2. Describe the types of software.
3. Differentiate between the application software and system software.
4. Explain the meaning of Spyware and trojan horses.
5. Highlight the process of avoiding malware.

7.1 Types of Software

Software can be categorised in many different ways. One way is to divide all programs into a few broad categories: system software, application software, utility software, and malicious software.

7.1.1 System Software

System software includes all the programs necessary to run a computer. Chief among these programs is the operating system itself, the most important piece of software running on a computer. As explained in study session 3, the operating system runs all the other programs. Without the operating system, the computer is useless.

Also included under system software are programs necessary for particular pieces of hardware. When you purchase a printer, for example, it often comes with a CD of software that must be installed for the printer to function.

7.1.2 Application Software

Application software provides specific services to the user, such as programs for word processing, email, computer games, financial management, spreadsheets, and image manipulation.

A word processor allows a user to create, edit, and format textual documents. Early word processors were very crude—“cut” and “paste” were once new ideas—but even then, these programs were often what sold people on computers. Word processors today are so advanced that they have more features than professional magazine layout programs had a few years ago.

Email clients allow the user to receive electronic messages, compose and send messages, and organise messages in folders. Recall from study session 3 that a client is a program that interacts with a central computer called a server. Email programs are called 'clients' because they retrieve and send mail through servers.

Computer game software allows a user to play a game on a computer. These games can run from very simple ("Solitaire") to incredibly complex ("Deus Ex: Invisible War"). Many computer games now have development budgets like those of Hollywood films.

Financial software tracks a user's financial accounts or prepares finance-related paperwork, such as tax forms. Examples in this category are Intuit's Quicken, Microsoft's Money, and Kiplinger's TaxCut.

Spreadsheet software provides a matrix of cells, in which each cell can be a number, a line of text, or a calculation involving the values in other cells. The power of a spreadsheet lies in being able to change a single value and have all the related results recalculate automatically. For example, if you create a spreadsheet that shows how much money you will have in five, ten, and fifteen years based on your current income, expenses, and return on your investments, you can change any of the input values and instantly see the results on your future earnings. Spreadsheet programs were among the first popular applications on computers. The idea of laying out calculations into cells like this had been used, on paper, for years, but of course, performing the calculations by hand was tedious. A computer spreadsheet saves all that tedious work.

Image manipulation was once reserved for photography professionals. But as digital photography has grown more popular and easier to use, even amateur shutterbugs want to modify their photographs and other images. The king of image manipulation software is Adobe's Photoshop. In fact, the word "photoshop" has become a verb meaning "manipulate the image" in the same way that "Xerox" has come to mean "copy."

7.2 Utility Software

Utility software enhances a user's computer experience. An example of a utility software is "anti-virus software," programs that clean unneeded files off the hard drive, screensavers, and so on. One might describe these programs as extensions of system software—features the user wishes were part of the operating system but aren't. In fact, many operating system features start off as utilities. The following are examples of what an antivirus program protects against:

7.2.1 Malware

Malware, a contraction of the words "malicious software," is a new term, but it is convenient and growing in use. Malware includes all programs that users don't want on their systems. Such software includes viruses, Trojan horses, and spyware.

7.2.2 Viruses

A computer virus is a short piece of programming code that attaches itself to a legitimate program and attempts to replicate itself by copying itself into other

programs. For example, if a virus attaches itself to your word processor every time you run it, the virus will execute first. Some viruses do nothing but replicate or display joke messages at certain times, but more malevolent viruses erase files on the hard drive. Even a virus that only replicates, however, can seriously damage the performance of a computer or prevent its proper function.

A worm is a type of virus that spreads not only to other programs on the computer it has infected, but also across network links. For example, some worms find the email program on a computer and then email themselves to every entry in the address book.

7.2.3 Trojan Horses

A Trojan horse is a program that masquerades as a legitimate piece of software but has a sinister ulterior function. For example, a program on a website may be advertised as a game. It will even run as a game, but when it runs, it signals back to the program's developer, who can then use the program to surreptitiously pilfer files from the user's computer.

7.2.4 Spyware

Spyware is hidden software that tracks user activity and reports it back to the program's developer. The most common type of spyware is used by advertisers to track websites a user visits so that tailored advertising can be later sent to the user. Spyware is often introduced to a computer system through a Trojan horse.

7.3 Avoiding Malware

Malware makes its way into a computer system because of weaknesses in system software design, user gullibility, or a combination of both. Keeping a system completely free of malware may not always be possible, but you can dramatically improve your odds by staying alert.

You can't fix problems with your computer's operating system, but you should always install any security updates as soon as they are available.

Avoiding gullibility, though, is entirely in your hands. One key is to never install any software unless you know and trust the original source of the software. Original is emphasised because if our friend Todd sends Marta a program as an email attachment, Marta must decide if she trusts the person who wrote the program, not Todd, who could be passing along a virus-infested program or a Trojan horse without realising it (Figure 7.1). The email may have been generated by a worm and not written by Todd at all. If you don't know what company created a program, don't install it.

Finally, don't open any email file attachment unless you know what kind of file it is. Learn to recognise the file types of the programs you work with: doc for Microsoft Word, for example, or movies for Apple's QuickTime movies. Certain file types that appear to be just data files can also contain program code and thus contain malware. If you are unsure, don't open the file.

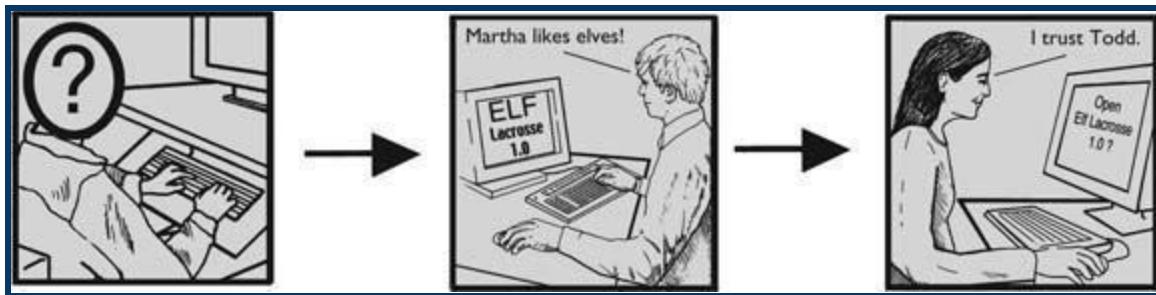


Figure 7.1: Know Who You're Trusting

In-Text Questions and Answers (ITQs and ITAs)

ITQ 7.1: What is the difference between system software and application software?

Answer: System software includes programs that are necessary for the computer to operate, such as the operating system, while application software refers to programs designed to perform specific tasks for the user, such as word processors or spreadsheets.

ITQ 7.2: Why is malware considered a serious threat to computer systems?

Answer: Malware, such as viruses, Trojan horses, and spyware, can damage files, steal sensitive data, and disrupt system performance. Its malicious nature and ability to spread quickly make it a significant security concern.

Conclusion

Three main kinds of software are system software, which includes all the programs necessary to run a computer; application software, which provides specific services to the user; and utility software, which enhances a user's computer experience. Malware, or malicious software, includes viruses and Trojan horses. The user must take care to avoid them.

The most important piece of software on any computer is its operating system. Without it, the computer ceases to function, and no other programs can be executed.

Summary

In this session, you have learnt to:

1. Explain the definition of software.
2. Describe the types of software.
3. Differentiate between the application software and system software.
4. Explain the meaning of Spyware and Trojan horses.
5. Highlight the process of avoiding malware.

Self-Assessment Questions and Answers (SAQs and SAs)

SAQ 7.1. What services does application software provide to users?

SAA 7.1. Application software provides specific services to users. It includes programs for word processing, email, computer games, financial management, spreadsheets, and image manipulation.

SAQ 7.2. How would you define computer game software?

SAA 7.2. Computer game software allows users to play games on a computer, ranging from simple ones like "Solitaire" to complex ones like "Deus Ex: Invisible War." Some computer games now have budgets comparable to Hollywood films.

SAQ 7.3. What is the role of financial software?

SAA 7.3. The role of financial software is to track a user's financial accounts or assist in preparing finance-related paperwork, such as tax forms. Examples include Quicken, Money, and TaxCut.

Multiple-Choice Questions and Answers (MCQs and MCAs)

1. Which of the following is an example of system software?

- A) Microsoft Word
- B) Antivirus software
- C) The operating system

D) A video game

Answer: C) The operating system

Explanation: The operating system is a crucial part of system software, as it manages hardware and software resources to run the computer.

2. What is the primary function of application software?

- A) To run the computer's hardware
- B) To provide tools for system maintenance
- C) To perform specific tasks for the user
- D) To protect against viruses

Answer: C) To perform specific tasks for the user

Explanation: Application software is designed to help users perform specific tasks such as word processing, gaming, or financial management.

3. Which of the following is a characteristic of malicious software (malware)?

- A) It speeds up computer performance.
- B) It is intentionally designed to damage or disrupt systems.
- C) It enhances system functionality.
- D) It automatically updates the operating system.

Answer: B) It is intentionally designed to damage or disrupt systems.

Explanation: Malware includes viruses, worms, Trojan horses, and spyware, all of which aim to harm the system, steal data, or disrupt operations.

4. What type of software is used to clean unnecessary files from a computer?

- A) System software
- B) Application software
- C) Utility software
- D) Malicious software

Answer: C) Utility software

Explanation: Utility software includes programs like disk cleaners and antivirus tools that help maintain system health and performance.

5. What is the main purpose of spyware?

- A) To enhance system performance
- B) To track user activity and send data to a third party
- C) To repair corrupt files
- D) To encrypt system data

Answer: B) To track user activity and send data to a third party

Explanation: Spyware secretly collects data on a user's activities and sends it to the software's developer, often for advertising or malicious purposes.

**Tutor-Marked Assessments
(TMAs)**

- TMA 8.1.** Differentiate between the system software and application software.
- TMA 8.2.** Explain the role of the operating system within system software.
- TMA 8.3.** Utility software is described as an extension of system software. Discuss its role in improving the computer experience, particularly in maintaining system performance and security.
- TMA 8.4.** Write short notes on the following:
- Malware
 - Viruses
 - Trojan Horses
 - Spyware

STUDY SESSION

EIGHT

OPERATING SYSTEMS

Introduction

The computer science view of an operating system is much deeper than that of the typical user. A user only thinks about the part of the operating system that can be seen and interacted with, which is known as the shell. Users often refer to the shell as the desktop. If you have used a recent computer with the Windows operating system, then the shell is in fact its desktop, with the start menu and the current time displayed in the corners along with everything else you see when the computer is first turned on.

The shell is an important part of an operating system, but it's a very small part. Most of what an operating system does happens "behind the curtain." In this section, you can look behind that curtain.

Learning Outcomes for Study Session Eight

When you have studied this session, you should be able to:

1. Explain the functions of an operating system.
2. Describe the process management.
3. Explain file management.
4. Explain the concept of memory management.
5. Describe the applications and system data management.
6. List different types of operating systems.
7. Explain the concept of suites and components.

8.1 Functions of an Operating System

An operating system exists for many kinds of management: process, file, memory, event, output device, security, and application and system data.

8.1.1 Process Management

The earliest computer systems allowed only batch processing, which means they executed only one program at a time. A school or company would have one computer (a mainframe), and the users would have to schedule time on it, the same way they would schedule the use of a conference room or an overhead projector. Modern operating systems allow multitasking, which means several programs can be running at once. Originally, this was seen as a way to avoid scheduling conflicts on a shared computer, but the capability has proven useful for personal computers as well. Multitasking has changed the way people use computers and the way they work in general. It's common for a user to have many programs open at once: an email program, an instant messenger, a word processor, and a web browser, for instance, and actively switch from one to the other as the work demands.

Even when the user is only running one program, though, the modern operating system depends on multitasking. That's because the operating system runs many background tasks, which are programs that receive no direct user interaction. An example of a background task is a print spooler. This program accepts print data to be printed from any application, like a word processor or email program, and then slowly sends the data to the printer. The print spooler is needed because printers can only accept a certain amount of

data at a time. If a word processor were responsible for directly sending data to the printer, the word processor would be busy, and thus unusable, until the print job was finished. By handing the data off to a “middleman,” the user can continue working immediately after starting the print job.

The operating system is responsible for keeping all the programs and background tasks running, which is known as process management because the operating system deals with “processes” instead of “programs.” A single program may have several instances running at the same time, and each instance is a separate process.

For example, it’s common for users to have multiple web browser windows open at once, and many nefarious websites will open a second window for you—usually containing an advertisement. Each of these windows is running the same program, but with different processes. Each window represents another copy of the program in memory, and to the operating system, it doesn’t matter that they are copies of the same program.

Because most computers have only a single CPU, only one process can actually be running at any given time. The operating system achieves multitasking through a technique called time-slicing. Simply put, this means that the operating system runs a process for a short period of time (a fraction of a second), then sets it aside and runs another process for a while, and so on. A program that seems to be running continuously is actually being constantly started and stopped.

The operating system must decide how long and how often to run each process before moving on to the next one. A process can be in one of several

states. A running process is currently executing on the CPU. A ready process is waiting for its turn to execute on the CPU. A blocked process is waiting for some event to happen to make it ready. For example, if the user has requested to save a file with the same name as an existing file, the program may put up an “Are you sure you want to overwrite the existing file?” message and wait for a response from the user. Because the process cannot continue until the user has made a decision, the operating system won’t waste the CPU’s time on it. The length of a time slice must be determined carefully. Task switching, which is the term for pausing one process and starting another, takes time. If you recall the explanation in study session 6 about pipelining, the pipeline must begin anew when the CPU switches to a different process. Also, all the values in the CPU’s registers must be stored in main memory so that when the process gets its next time slice, it can pick up right where it left off.

In general, then, the performance of the system is improved by having as few task switches as possible, which means having long time slices. The problem is, even though long slices keep the CPU doing as much useful work as possible, they lower the perceived performance of the system. That is, the user’s experience with the system suffers because there’s often a greater delay between the user’s action and the program’s reaction on the monitor. Thus, the operating system must carefully manage the time slices to achieve the best balance between overall performance and user experience.

The operating system tries to schedule the use of the CPU to give more of its time to the processes that need it. This is not easy for the operating system to arrange because there’s no way to predict which programs will be the busiest.

Instead, the operating system monitors the use of the CPU and assumes that past use is a predictor of future use. The processes are in a queue, like a line at a ticket window. When they get to the front, they use the CPU for a moment before they have to get back in line. If the process doesn't need all the CPU time it is offered, such as a background process, it is moved all the way to the back of the line, which means it takes a long time to get to the front again. However, a process still executing when its time slice runs out is assumed to be busy and is only sent back to a point in the middle of the line. This way, the busy processes get to the front of the CPU queue more often.

8.1.2 File Management

File management refers to the ability to read and modify files on storage devices and to create and delete files.

In study session 4, we said that storage devices like hard drives are divided into tracks and sectors, and a single file that occupies many sectors could be spread all over a disc. This fragmentation makes accessing a file tricky. Fortunately for application programmers, locating the fragments of a file is the responsibility of the operating system.

Each storage device has both a logical and physical structure. The logical structure is what the user sees. Consider a small file stored on a Windows operating system under the name C:\MyDocuments\myfile.txt. Here, myfile.txt is the name of the file, and C:\MyDocuments is the file's logical location. On a Windows system, it means that the file is on a hard drive

indicated by the letter C and that on that drive it is stored in a folder called "My Documents."

Remember the physical structure that was discussed in study session 4: each file is stored across various sectors on a disc. The logical structure and the physical structure are not related. Just because two files are located in the same folder, for example, does not mean they are anywhere near each other on the physical storage device.

Thus, a primary job of file management is mapping logical file locations to physical locations. To do this, the operating system maintains a directory on each storage device. The directory lists which physical locations go with which files and also keeps track of which locations on the storage device are free, that is, which are not currently used by any file. As files grow and shrink, and are created and deleted, the operating system changes the directory to reflect how the sectors are currently used. This file management allows all the application programs to deal exclusively with the logical file structure.

Note that the operating system will generally cache files. If you recall from study session 5, the CPU has a cache, which is a small amount of RAM that can hold recently used data close by, so if it's needed again soon, the request doesn't have to go all the way to main memory. Similarly, the operating system may hold recently accessed file data in main memory, so if it's needed again, the request doesn't have to go back to the storage device. Or, if a program has requested the first fifty bytes from a file, the operating system may go ahead and retrieve several kilobytes or more from the file in anticipation of a later request. This process is another form of caching.

8.1.3 Software Hooks

Certain utility programs must integrate themselves with the operating system to function. A good example is a program that scans files for viruses before they are used. If you are in a word processing program and ask the program to open a file, that file must pass through the virus scanner on its way from the operating system to the word processor.

To allow these kinds of utilities to function, the operating system must provide “hooks.” A hook is a request for the operating system to invoke another program for a certain operation. The virus scanning program establishes a hook that says, “Whenever you are opening a file, call me first.”

A few years ago, many were complaining that Microsoft’s dominance of the operating system market gave it an unfair advantage when it came to writing applications. One of the accusations was that because Microsoft had more intimate knowledge of the operating system, Microsoft developers could write software that relied on undocumented hooks, giving their programs abilities others could not match.

8.1.4 Memory Management

Another important function of the operating system is memory management, which is the service associated with the allocation of main memory. Essentially, the memory manager decides which main memory addresses are associated with which processes at any given time. All the processes currently running, and the operating system itself, have main memory needs that must be met. When a program is first begun, it is given enough memory to hold some

parts of the program and data. Over time, the program may request additional main memory space. A request for main memory that occurs during a program's execution is known as a dynamic memory allocation and is a function provided by the operating system.

One problem with dynamic memory allocations is that a program must remember to deallocate the memory when it is no longer needed so the memory can be used by another program, and all programs are expected to return all dynamic memory before closing. If a program neglects to deallocate its dynamic memory when it no longer needs it, the result is a memory leak—a block of memory that is allocated but unused. Memory leaks degrade system performance because, in essence, the system now has less main memory.

Another complication of dynamic memory is that some memory is shared between processes. When this happens, the memory manager's job is more difficult because it must track each use of the allocated memory. Just because one process has signalled it is done with the memory doesn't mean the memory can be deallocated. Instead, the operating system must wait until all processes have signalled.

Although today's computers have large main memories, they are often not large enough. As memory sizes have grown, the size of the average program has grown even faster. Consider that some software packages come in multiple CD-ROM sets, and a single CD-ROM can hold more data than the main memory on most computers. In addition, because of multitasking, many users are running multiple programs at once.

One way to never run out of RAM is through a technique called virtual memory, in which a larger main memory is simulated through the use of storage on the hard drive. The operating system maintains a table that maps each program's simulated address with the physical addresses in main memory.

As an example of how this works, let's say that the user requests to start TypeEdit, a word processing program whose size is fifty kilobytes. The operating system retrieves the first portion of the program, say 1,000 bytes, and decides to store it in main memory starting at address 100,000,000. The operating system notes this in its virtual memory table. The program itself is unaware of where it is located in physical memory. It is written as if it were stored at location 0. The entry in the table would look something like this:

Program	Virtual Address Range	Physical Address
TypeEdit	0–999	100,000,000– 100,000,999

Figure 8.1: TypeEdit program showing a virtual address.

If the program executed an instruction that said, "Store this value at location 45," the operating system, using this table, would translate that instruction to: "Store this value at location 100,000,045."

Now suppose the program needed to execute the portion of the program at locations 1,000–1,999. Because this part of the program isn't in memory, the operating system retrieves it and stores it at 75,000. Now the table looks like this:

Program	Virtual Address Range	Physical Address
TypeEdit	0–999	100,000,000– 100,000,999
TypeEdit	1,000–1,999	75,000–75,999

Figure 8.2: TypeEdit program showing two virtual addresses

Thus, each currently running process has entries in this table, and this way, each process can act as if it has the entire main memory to itself, letting the operating system handle the details.

The blocks of physical memory that the operating system hands out are called pages.

When a process requests a virtual memory range that is not currently in main memory (as TypeEdit did in the second part of the example), it's known as a page fault.

Pages may also need to be written to the hard drive. If a page contains nothing but program instructions, it can simply be written over by another page because the operating system can always retrieve that part of the program again. But if a page contains a program's current data, that data needs to be kept for when the program needs it again. The operating system maintains a special area on the hard drive, called the swap file, for the temporary storage of pages. For the best performance, this file is usually allocated as one contiguous block in the middle of the disc so that whatever other hard drive access is going on, the needed page is never far away from the read/write head's current position.

Over time, a process may gain pages or lose pages. The more active a program is, the more pages it is awarded. The goal of the virtual memory manager is to minimise the number of page faults because every page fault is a slow trip back to the hard drive. If you've ever had a large number of programs open at once and then switched from a program you've been using a lot to one you haven't used in a few minutes, you may have noticed a temporary slowdown. That's because, over time, the active program grabbed most of the pages in memory. When you switched to the inactive program, it suddenly needed a lot of pages it hadn't needed in minutes, and this generated a large number of page faults. In extreme cases, there are so many active processes competing for main memory that the computer spends most of its time swapping pages, a phenomenon known as thrashing. In general, when you see the hard drive light flash and you are not opening or saving a file, it's a good bet you're seeing the virtual memory manager at work.

8.1.5 Event Management

Another important function of a modern operating system is event management. In computer science, an event is a specific action that produces a reaction in some program. If you click the mouse in a certain spot on a program's window, for instance, the program may respond by displaying a menu. Mouse clicks and keystrokes are user-initiated events. Other events may be generated by the operating system. For example, a program that displays the current time in a corner of the screen might request a timer event to occur every minute so it can update its display.

It is the operating system's responsibility to route events to the appropriate programs. The programs do not directly communicate with the mouse or keyboard. Instead, the operating system collects all the events and sends messages to the programs that need them.

This separation between application programs and hardware is another example of abstraction, the concept introduced in study session 1. The programs are one level of abstraction removed from the mouse and keyboard. Different keyboards and mice may have different characteristics, but only the operating system needs to deal with that. It handles this with another layer of software between the operating system and devices. Software that acts as a middleman between the operating system and a device is called a device driver. These drivers are provided by the manufacturer of the device.

For output devices, the operating system expects input in a standard format. The job of an input device's driver is to provide the input in that format. This way, the programs that use the input devices can treat them generically, without worrying about who made them or what specific features they may have.

8.1.6 Output Device Management

The reverse of event management is output device management. Just as the operating system acts as a middleman between programs and input devices, it also acts as a middleman between programs and the monitor or printer.

Device drivers exist for output devices too. For output devices, the operating system provides a generic set of commands that programs can use to display

or print, and, on the other end, issues display or print commands in a standard format. The device drivers translate this standard output into the commands that the specific printer or graphics card expects.

Writing device drivers is a critical programming job. The graphics card's driver, for example, probably executes more than any other program in the computer. An excellently designed piece of hardware can still produce a poor user experience if the driver is flawed or does not take advantage of all the device's strengths.

Look at the chain of actions that would occur when a user types a letter in a word processing program (Figure 8.3). The user presses the S key on the keyboard. The keyboard's device driver communicates this key press to the operating system. The operating system determines that the key press belongs to the word processing program and sends the key press to it. The word processor determines that an S must appear in the display at the current cursor location. Now, it must update its display. To do so, it communicates back to the operating system. The operating system communicates with the device driver for the system's graphics card. The device driver communicates with the graphics card, which generates the display for the monitor. Thus, even a simple word processor's keystroke involves the interoperation of many pieces of software.

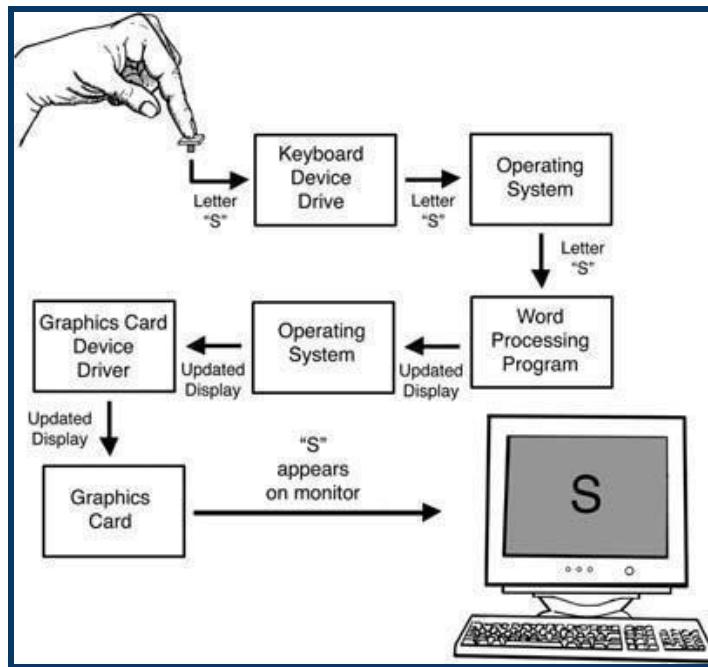


Figure 8.3: Chain of actions

8.1.7 Security

The operating system is involved in many aspects of computer security as well. Perhaps the most obvious aspect of this responsibility is user authentication, which just means positively identifying a user, usually through a username and password. Some operating systems, like Windows XP, allow multiple user accounts on a single personal computer, and each user can have private folders and files that other users cannot view, modify, or delete. If a computer is connected to a network, the operating system may be required to authenticate the user before the computer can access files across that network.

The operating system also provides security in less obvious ways. As stated earlier, in a multitasking operating system, a single computer can have multiple programs running. A flawed or malicious program could alter the instructions or data of another program, corrupting the data or crashing the program. The operating system enforces rules to keep each program separate. This separation doesn't prevent a program from crashing or corrupting its own data, but it helps prevent a single program's malfunction from creating system-wide havoc.

8.1.8 Application and System Data Management

Operating systems must store system data. An operating system is installed on a wide variety of computers with different configurations of hardware and software, and these configurations must be tracked. The operating system stores data detailing the kind of CPU in the system, the size of the hard drive, the kind of graphics card, the name of the device driver for each device in the system, and other hardware-related data. On a system with user accounts, all the account information and user preferences must be stored. The operating system may also offer choices in display resolution, organisation of menus, whether certain user actions generate a sound, and so on.

Operating systems also store some application data. Most programs allow the user to customise the interface to some extent by adding or removing a certain “toolbar,” changing the colours used, and so on. This data is stored with the operating system as well

8.2 Current Operating Systems

Current operating systems include Windows, UNIX, Mac OS, and Palm OS.

8.2.1 Windows

As mentioned in study session 2, Microsoft's Windows series of operating systems, which currently dominate the computer industry, have a long and colourful history. Microsoft gained prominence with its first operating system, DOS ("disc operating system"), which was a text-based system. If you wanted to run a program, you typed the name of the program at the "command line," and DOS would run it for you. If you wanted to delete a file, you typed "del" and the name of the file to delete it. Although primitive by today's standards, DOS was the operating system for the original IBM PC, and when the latter began selling in huge numbers, it established Microsoft's position.

The first release of Windows, in 1985, can be seen as a reaction to the introduction of the Apple Macintosh the year before. The first Windows was not a whole operating system so much as just a shell that provided a graphical interface, a mouse-based way to run programs while DOS, behind the scenes, still did all the work. While this first version contained many of the features taken for granted now, it had a long way to go. For example, multiple programs could be running in separate windows, but the windows could not overlap.

Another problem with early versions of Windows was the use of non-preemptive multitasking, which meant the applications were responsible for giving up the CPU when their time slice was over. In contrast, most operating systems, including current versions of Windows, use preemptive

multitasking, in which the CPU control automatically returns to the operating system after an application's time is up.

With non-preemptive multitasking, an application is expected to “check in” with the operating system every so often, so the operating system can regain control. The problem arises when the program “crashes” or is tied up waiting for some event that isn’t occurring. If the application never checks in, the operating system never gets control back, and the whole system is frozen. With preemptive multitasking, a misbehaving application won’t keep the operating system from maintaining control of the computer. Other applications are allowed to continue.

Windows really took off with the release of Windows 95. This version was the first that did not require DOS to be installed on the computer first. Parallel with the development of Windows 95, however, Microsoft had developed a completely new operating system, called NT, for “new technology.” Originally, NT was supposed to have been a text-based operating system like DOS, but during development, Microsoft decided it would have a graphical interface and be released under the name Windows NT.

Now, Microsoft had two diverging operating systems: Windows 95, an evolutionary development from the days of DOS, and Windows NT, a new operating system with a more modern design and all the interface features of Windows. Microsoft developed both because of backwards compatibility. Just like Intel and their CPU development, Microsoft worried about forcing everyone to switch to a completely new operating system. Although Microsoft took pains to allow older software and hardware to work with NT, the system was so

different that some incompatibility was inevitable. Thus, Microsoft hedged its bets and kept the old operating system alive, reasoning that home users would use Windows 95 and business users would switch to Windows NT.

Eventually, though, the “new technology” won. Windows 95 became Windows 98 and then Windows ME (“millennium edition”). Windows NT became Windows 2000, and then Windows XP. With Windows XP, Microsoft is only producing a single operating system again. Windows XP is used by home and business users alike.

8.2.2 UNIX

The UNIX operating system, developed at AT&T’s Bell Laboratories, has been around for over thirty years. Where operating systems like Windows tend to emphasise features that help novice users, UNIX emphasises features that help programmers and expert users. For example, UNIX has a feature called a pipe, which is a mechanism for setting the output of one program as the input of another program. If Program A corrects the spelling of a text document, Program B formats a document in a three-column layout, and Program C prints a document, then a user can pass a document through A and B to C to spell-check, format, and print the document in a single command. Though UNIX is old, it has all the features of a modern operating system, including multitasking. One of its strengths is its simplicity, which allows it to be easily modified for use on newer computers. By itself, though, UNIX has no graphical interface; it is purely a text-based operating system. This problem is quickly remedied through the use of X-Windows, a separate graphical user interface

system that runs on top of UNIX. Although UNIX and X-Windows are two different things, and X-Windows could be run on other operating systems, X-Windows can be considered the de facto interface for UNIX, except on the Macintosh.

UNIX has surged in popularity in the last few years because of Linux, an open-source version of UNIX discussed in study session 2.

UNIX offers users a great deal of low-level control and is therefore popular with expert users. Home users, though, have been slower to accept this operating system. Many popular programs are not available for UNIX, and it is more difficult for the typical home user to learn. Some UNIX programmers almost seem to take pride in offering obscure commands and interfaces.

8.2.3 Mac OS

Mac OS is the operating system for the Apple Macintosh computer; the OS stands for “operating system.” Unlike UNIX, the Mac OS developers strove to make their operating system as painless to use as possible. Many of the features that are common to all graphical interfaces were introduced in Mac OS. The first versions of Mac OS, simply called “System,” as in “System 1.0,” only supported a black-and-white display and could only run a single program at a time. However, users could switch from one program to another, giving the appearance of multitasking.

By version 8.0, the name changed to Mac OS, and the operating system supported multitasking, albeit non-preemptive multitasking.

A current version is Mac OS X, where the X indicates the roman number for 10. This version actually uses Unix as the base operating system, while providing a polished graphical interface (called “Aqua”) that is as easy to use as Macintosh owners expect. While the Mac OS holds only a tiny percentage of the marketplace, those who use the Macintosh are a devoted bunch, almost fanatical in their appreciation of the operating system’s elegance.

8.2.4 Palm OS

The above-mentioned operating systems are designed for use on personal or larger computers. The Palm OS operating system is for small computers. It was designed for the Palm Pilot, one of the original personal digital assistants. A PDA, as it’s called, is a handheld computer that functions as an appointment calendar, notepad, and calculator and can perform other basic computing tasks.

Over time, PDAs and cell phones have merged, creating “smart phones” with email and web browsing capability in addition to basic PDA and cell phone features.

The Palm OS is a tiny operating system compared to the other systems, because the storage capabilities of handheld devices are so small. This small storage space is not a problem because Palm OS only needs to support a fraction of the features of the others. It doesn’t need complicated file management, for example.

8.3 Suites and Components

In recent years, attention has shifted away from independent application programs towards programs that work together as a team. Interoperability refers to the ability of different pieces of hardware and software to communicate with each other. Software interoperability is an important trend.

8.3.1 Suites

One way to get programs to work together is to write them together. A suite is a set of applications that are sold as one package, and it offers several advantages to the consumer. The price of the suite, for instance, is usually much lower than the price of the individual applications, and because the programs are produced by the same developer, they can be made to interoperate.

Consider a suite of standard office applications, which include word processing, spreadsheets, and email programs. Interoperability could allow the email program to use the word processor program for editing. In other words, while you were composing a new email message, the email program's editing window would display the word processing program.

If the programs didn't communicate in this way, then the email program would need its own programming code for editing text. This capability would be redundant, and the email program would probably have fewer editing features. Similarly, the word processor might allow a spreadsheet to be displayed inside a document. When editing the spreadsheet, the user is actually using the spreadsheet program, even though the spreadsheet is inside a word

processing document. Or, the word processor might allow a graph, generated from spreadsheet data, to be displayed in a document. If the user later edited the spreadsheet data, the graph would automatically be updated the next time the document was opened in the word processor.

Another advantage of a suite is that each program can have a similar interface style. Developers talk about a program's "look and feel." When a program looks and feels like another program that the user is already comfortable with, the user can learn the program more easily.

Different suites exist for different kinds of users. The most common suite, as already mentioned, is the "office" suite for use in business and home offices. It usually includes a word processor, spreadsheet, email, presentation software, and sometimes other components.

Other common suites are used by web page designers. They include programs to edit web pages, organise entire Web sites, manipulate images, and so on.

Still another suite highlights graphic design and is used for the design and layout of books, magazines, advertisements, and other materials. Typical programs include a photograph manipulator, an illustration program, a document layout program, and so on. Because so much content is directed towards the Web, many graphic designer suites also include the features of a web page design suite.

8.3.2 Components

Interoperability among programs from the same developer isn't simple, but interoperability between programs from different developers is even more

complicated. What's needed is a standard way of communicating. Programs that conform to some standard interface to facilitate interoperability are called components.

There are many advantages to components for developers. For one, using components can decrease development time significantly. Consider how many programs need to let the user edit text like a word processor, such as email programs, presentation software, and data entry programs. Rather than develop that functionality themselves, a programming team can purchase a component with that capability and simply plug it into their developing application.

Components can also lead to fewer software errors. If one company makes a word processor component that hundreds of other companies will use, the component is rigorously tested and so widely used that any flaws are soon uncovered and the component is repaired and redistributed. Now, suppose instead that 100 companies each develop their own word processing capability from the ground up. It's much more likely that some of the applications will have errors in the text editor because fewer people will have seen and tested each application.

A couple of specific implementations include COM and CORBA.

COM

Microsoft's solution for interoperability is called COM, which stands for Component Object Model, and it allows the building of applications using

components from different developers. In this context, an “object” is just a program that exists to provide a service to applications.

COM works well, but because it is backed by Microsoft, it is viewed as a Windows-centric technology, even though Microsoft makes COM technology work under other operating systems as well. Some developers are nervous about conforming to a standard that is owned by a single company.

CORB6

CORBA stands for Common Object Request Broker Architecture. Unlike COM, CORBA is an open standard, which means it is supported by a non profit organisation (in this case, the OMG, or Object Management Group) that any programmer or company may join.

This support carries two benefits for programmers. First, it gives them a voice in the development of the standard. And second, it keeps the development transparent so that programmers won’t be given any surprises. Actually, the owner of a closed standard (like Microsoft with COM) would probably never deliberately make things difficult for programmers because that would be self-defeating. But everyone feels better when they have more direct control over the future.

In-Text Question and Answer (ITQ and ITA)

ITQ 8.1: What is the primary function of the operating system’s process management?

Answer: The primary function of process management is to manage and

coordinate the execution of processes, ensuring that multiple programs can run concurrently without interfering with each other. This involves managing process states and allocating CPU time effectively.

ITQ 8.2: Explain the concept of multitasking in modern operating systems.

Answer: Multitasking allows a computer to run several programs simultaneously. The operating system achieves this by dividing CPU time into short slices, allowing different processes to execute in a way that appears simultaneous to the user, even on systems with a single CPU.

Conclusion

The operating system has a number of responsibilities. First, it must keep all the programs and background tasks executing, which is known as process management. Second, it must provide file services such as reading, creating, and modifying files on storage devices. Third, it must allocate and deallocate blocks of main memory, which is memory management. A key part of memory management is virtual memory, in which a larger main memory is simulated through the use of the hard drive. Fourth, it must route events, such as user actions, to the appropriate program and then route program actions to the proper output device. In general, this routing forms the connection between application programs and the devices in a computer system. Fifth, it must protect programs from interfering with each other or with files they should not access. Sixth, it must store key data about the system and the applications that run on it. A suite is a collection of applications sold together. Suites save

the consumer money, and the applications they contain can benefit from each other's features.

A trend in software is component-based programming. Components are programs with standard interfaces that can communicate with each other, even though they may have been developed by different programmers.

Summary

In this session, you have learnt to:

1. Explain the functions of an operating system.
2. Describe the process management.
3. Discuss file management.
4. Explain the concept of memory management.
5. Describe the applications and system data management.
6. List the different types of operating systems.
7. Explain the concept of suites and components.

Self-Assessment Questions and Answers (SAQs and SAAs)

SAQ 8.1. What are the different kinds of management handled by an operating system?

SAA 8.1. The different kinds of management handled by an operating system are: Process, File, Memory, Event, Output device, and Security.

SAQ 8.2. How does multitasking change the way people use computers?

SAA 8.2. Multitasking changes the way people use computers by allowing them to run multiple programs or tasks simultaneously. Instead of being limited to running a single application at a time, multitasking enables users to switch between different programs quickly. This means users can perform multiple activities concurrently, such as listening to music while browsing the web, running a virus scan in the background while working on a document, or downloading files while editing images. Multitasking enhances productivity and efficiency by utilising the computer's resources more effectively.

SAQ 8.3. What is a background task in an operating system? Provide an example.

SAA 8.3. A background task in an operating system is a task or process that runs "in the background" without requiring user interaction or attention. These tasks typically perform non-essential operations or tasks that can be executed independently of the user's primary activities. Examples of background tasks include:

- **System Updates:** When an operating system automatically downloads and installs updates while the user continues to work on other tasks, the update process runs in the background.
- **File Indexing:** Some operating systems index files to enable faster searching. This indexing process often occurs in the background, allowing users to access and search files without significant delays.

- **Automatic Backups:** Operating systems may perform automated backups of files or system data in the background, ensuring data safety without interrupting the user's work.
- **Antivirus Scans:** Antivirus software often performs regular scans for malware or viruses in the background, protecting the system without disrupting the user's activities.

Multiple-Choice Questions and Answers (MCQs and MCAs)

1. What is the main purpose of an operating system's process management?

- A) To manage the physical hardware of the computer
- B) To control how programs access memory
- C) To allow multitasking and manage the execution of processes
- D) To prevent software from running

Answer: C) To allow multitasking and manage the execution of processes

Explanation: The operating system manages processes by allocating CPU time, allowing multiple processes to run simultaneously.

2. How does an operating system use virtual memory?

- A) It increases the amount of RAM available
- B) It stores program data in the CPU registers.
- C) It simulates a larger memory space using storage devices.
- D) It automatically allocates more memory to active programs.

Answer: C) It simulates a larger memory space using storage devices.

Explanation: Virtual memory allows the operating system to use hard drive space to simulate more RAM, enabling larger programs to run.

3. Which of the following is an example of a background task managed by the operating system?

- A) Email program
- B) Web browser
- C) Print spooler
- D) Word processor

Answer: C) Print spooler

Explanation: A print spooler is a background task that manages print jobs, allowing the user to continue working while the document is being printed.

4. What is the function of a device driver in an operating system?

- A) To control the system's security features
- B) To enable communication between the operating system and hardware
- C) To schedule tasks and manage CPU time
- D) To manage the storage of files on the hard drive

Answer: B) To enable communication between the operating system and hardware

Explanation: Device drivers act as intermediaries between the operating system and hardware devices, ensuring they work together smoothly.

5. What is the key feature of preemptive multitasking in modern operating systems?

- A) The CPU runs one program continuously until it completes.

- B) The operating system assigns a fixed time slice to each program.
- C) The operating system waits for user input before switching tasks.
- D) Programs are allowed to run indefinitely without interruption.

Answer: B) The operating system assigns a fixed time slice to each program.

Explanation: Preemptive multitasking ensures that each process gets a fair share of CPU time, allowing the system to remain responsive.

**Tutor-Marked Assessments
(TMAs)**

- TMA 9.1.** Describe how an operating system manages **file storage and retrieval**. Include the distinction between **logical** and **physical file structures**, and explain the role of **caching**.
- TMA 9.2.** Discuss the **security responsibilities of an operating system**, including user authentication, process isolation, and file access control.
- TMA 9.3.** Briefly describe the purpose of **device drivers** in an operating system.
- TMA 9.4.** Describe the following operating system management in great details
 - Process management
 - Memory management
 - File management

STUDY SESSION

NINE

PROGRAMMING

Introduction

The central task in computer science is the development of new programs. These programs are created through software engineering, a process similar to that of engineering for building programs. The specifications of the program are laid out to determine what the program is going to do. Then the program is designed to determine how the program will achieve its goals. Only then is the program written.

At the design stage, the programmer determines the program logic using an algorithm, which is a series of instructions that define how to solve a problem. This definition may sound very similar to that of a computer program, but there's an important distinction. An algorithm is described "in English" rather than written in a computer programming language. Thus, an algorithm is written to be "human readable," whereas a program is written to be "machine readable." Put another way, an algorithm is the idea behind the solution, while the program is the actual implementation of that solution.

Learning Outcomes for Study Session Nine

When you have studied this session, you should be able to:

1. Describe the program logic in program design.
2. Explain the sequential execution of a program.
3. Perform the conditional execution of a program.
4. Discuss the concept of repetitive execution.
5. Describe the phase of control flow.

9.1 Program Logic

Computer scientists prefer to discuss algorithms when determining the best solution to a problem. Because programs are written by programmers who have individual strengths and weaknesses and are executed on computers, which may execute some kinds of instructions faster than others, a program can provide a misleading picture. Algorithms are a neutral proving ground for possible solutions.

While modern programming languages are widely varied, they all share the basic concept of control flow, which is the order in which the program instructions are executed. Control flow is determined by the control structures, which are the traffic cops on the program highway.

Although computers are capable of astoundingly complex tasks, even the most complicated computer programs are built using simple arithmetic operations and a few elementary control structures, coming together in nearly infinite combinations. These elementary control structures are discussed in this section using algorithms. While many algorithms involve the manipulation of numeric data, the algorithms below involve our computer science student Todd, a young man with a very systematic approach to fun.

9.2 Sequential Execution

The sequence is the default control flow; it results from the lack of any control structure and is the easiest to understand. In a computer program, the instructions are executed as they are listed, from top to bottom, unless the

computer is told otherwise. Executing instructions in the order listed is called sequential execution.

For example, Todd uses a sequential execution algorithm when he grooms himself before going to a party.

“Pre-Party Grooming” algorithm

Wash hair. Shave face. Brush teeth

Gargle with mouthwash. Apply “product” to hair, sculpt

When Todd executes this algorithm, he simply executes the instructions in order. The order is important to Todd. For example, he washes his hair first so it has time to naturally dry during the other steps before he applies his “product”.

Todd always grooms himself the same way. Sequential execution is all that is needed when there are no choices to be made. In most cases, however, more is needed.



Figure 9.1: Todd's Grooming Ritual

9.3 Conditional Execution

In the previous example, all the instructions had to be executed every time the algorithm was used (every time Todd grooms himself for a party). In other cases, an instruction or group of instructions may be optional. With conditional execution, an instruction is executed or skipped based on a condition tested when the instruction is reached. In most algorithms and programming languages, conditional execution can be spotted by looking for the word “if,” as in, “if this condition is true, then do this step.” For that reason, programmers often refer to conditionally executed instructions simply as “if” statements.”

Todd uses conditional execution when he executes his algorithm to dress himself for a party. Todd takes his clothing choices seriously, altering his outfit to match the party’s attendees and locale.

“Getting Dressed for Party” algorithm

Put on boxers

If party is downtown,

Then put on dress shirt and khakis

Otherwise,

Put on a black sweater and jeans. Put on a leather belt and dark shoes. If Marta is at party,

Then put on the Movado watch she gave me. Put the cell phone in the pocket.

Note first that sequential execution is still the rule. This algorithm is executed from the first instruction to the last, except for the “if” statements.

This example demonstrates two kinds of conditional execution. The first is a “two way” conditional. Todd dresses upscale for downtown parties but is relaxed when partying in the suburbs. Because he will always do one or the other—he’s not going to attend a party in his underwear; he’s choosing from two paths. The second conditional is “one way.” If his friend Marta will be at the party, he’ll wear the watch she gave him. There’s no alternative here. He either wears the watch or he doesn’t.

9.4 Repetitive Execution

While conditionals are used to execute or skip an instruction, repetitive execution is used when the same instructions are executed more than once. In normal discourse, most programmers refer to repetitive execution as looping, from the idea of circling around to a previous position. Two ways to loop are counter-based and conditional.

9.4.1 Counting Loop

A counting loop is used when the number of times to repeat the code is known before the loop starts. Todd uses a counter-based loop when he attends parties where he will meet his neurotic, platonic girlfriend, Marta. If he doesn’t talk to Marta at least three times, she will become moody and bitter for weeks. After fulfilling his duty with Marta, however, and being in no mood for further conversation, Todd turns up the music volume so that everyone can dance instead of talk.

“Mingle When Marta’s Round” Algorithm

Repeat (three times):

Locate Marta

Ask her how she is doing. Listen to the response.

Spend ten minutes talking to other guests. End Repeat

Crank up stereo to drown out conversation

Note the use of the “End Repeat” to give an unambiguous ending point to the counting loop. All programming languages have something similar, so that even with all the extra spaces removed, there would be no doubt which instructions are “in” the loop and which are “out.”

9.4.2 Conditional Loop

With conditional looping, a set of instructions is executed until a specified condition is met. Conditional loops do not require knowing how many times the loop will execute. Todd uses a conditional loop at parties when Marta is absent. Freed from any conversational obligations, Todd moves from group to group at the party, settling down with the first group that is not talking about politics.

“Mingle When Marta’s Absent” Algorithm

Repeat these steps:

- Sidle up to new group. Introduce yourself.
- Listen to conversation for five minutes. Until conversation is not about politics. Spend hours chatting.

9.4.3 Procedures

Using loops allows programmers to avoid having to repeat the same block of instructions several times in a row. In some cases, however, a program calls for the same process to be repeated, but now the repetitions are separated by

other tasks. Or, two processes will be almost the same, but not exactly the same.

When Todd meets a young woman at a party (and the young woman, like Todd, is of legal drinking age), he likes to “break the ice” by making two martinis: a vodka martini for himself and a gin martini for the lady.

“Breaking the Ice” Algorithm (Original)

Fill shaker with $\frac{1}{2}$ cup ice. Add 2 ounces vodka

Add $\frac{1}{4}$ ounce dry vermouth Shake

Strain contents into martini glass. Add olive

Dump ice from the shaker and rinse the shaker. Fill shaker with $\frac{1}{2}$ cup ice

Add 2 ounces gin

Add $\frac{1}{4}$ ounce dry vermouth. Shake

Strain contents into martini glass. Add olive

This algorithm contains two sets of instructions that are very similar except that in the second step of making the first drink, Todd uses vodka, and in the second step of making the second drink, he uses gin. Because the same exact series of instructions is not repeated, a loop cannot be used to simplify this algorithm. In a case such as this, the programmer (or Todd in this case) uses a procedure, which is a named block of instructions that, when referenced, results in the execution of those instructions. Procedures allow the use of parameters. A parameter is a placeholder for an actual value in a procedure; the parameter’s value is specified when the procedure is used. This allows the

creation of generic sets of instructions for doing related, specific tasks. Here is Todd's same martini algorithm using a procedure:

“Breaking the Ice” Algorithm (Using Procedure)

Perform martini procedure (alcohol is vodka). Dump ice from shaker and rinse shaker Perform martini procedure (alcohol is gin) Martini procedure (has alcohol as a parameter): Fill shaker with 1/2 cup ice

Add 2 ounces alcohol

Add 1/4 ounce dry vermouth. Shake

Strain contents into martini glass. Add olive



Figure 9.2: Todd Makes Two Martinis

In this version, the instructions for making a martini are only given once. For the martini procedure, the type of alcohol is specified. The first use of the procedure adds two ounces of vodka in the second step; the second use of the procedure adds two ounces of gin (see Figure 9.2). Besides avoiding duplication of instructions, using procedures allows programmers to break up

complex tasks into smaller, named functional units, which makes the algorithms easier to follow. Look again at both versions of the algorithm. In the second version, it's very easy to see that the overall purpose is to make two martinis. In the first version, this is obscured.

9.5 Exploration: Finding Control Flow

In the modern world, everyone is always trying to follow printed instructions. Some common examples are tax forms, DVD player instructions, build-it-yourself furniture, and forms on websites. These instructions are, unavoidably, kinds of algorithms. Try to find the sequences, conditions, loops, and procedures in the instructions that you see every day in these forms and instructions. Which control flow elements are common? Which are rare? Would the instructions be clearer with the use of a different control flow?

In-Text Question and Answer (ITQ and ITA)

ITQ 9.1: What is the difference between an algorithm and a program?

Answer: An algorithm is a human-readable description of how to solve a problem, often written in plain language, while a program is the actual implementation of that algorithm in a machine-readable programming language.

ITQ 9.2: What is sequential execution in programming?

Answer: Sequential execution means the instructions in a program are executed in the order in which they are written, one after another, with no branching or skipping.

Conclusion

The software development process is a journey from idea to implementation. Along the way, the programmer develops an algorithm or uses an existing algorithm.

Algorithms and programs use control structures to define the order in which to execute the instructions. The common control structures are sequential (execute instructions in order), conditional (execute instructions only if some test is met), repetitive (execute instructions a specified number of times or repeat until some condition is met), and procedure (a block of instructions referenced from elsewhere in the algorithm or program). The programmer must use these control structures to define solutions to problems. Tasks that are intuitive when discussed generally are often difficult to describe in a formal algorithm. The best programmers are those who can quickly see a good solution to a problem in terms of the control structures and instructions of the specified programming language.

Summary

In this session, you have learnt to:

1. Describe the program logic in program design.
2. Explain the sequential execution of a program.
3. Perform the conditional execution of a program.
4. Discuss the concept of repetitive execution.
5. Describe the phase of control flow.

Self-Assessment Questions and Answers (SAQs and SAs)

SAQ 9.1. What is sequential execution in computer programming?

SAA 9.1. Sequential execution in computer programming refers to the default control flow where instructions are executed in the order they are listed, from top to bottom. It is a straightforward and linear execution model without any branching or repetition. Each instruction is executed one after the other, without deviation, unless the program specifies otherwise.

SAQ 9.2. How does Todd's sequential execution algorithm for pre-party grooming work?

SAA 9.2. Todd's sequential execution algorithm for pre-party grooming works by executing each instruction in the given order. He follows a predetermined sequence of actions, such as washing his hair, shaving his face, brushing his teeth, gargling with mouthwash, and applying "product" to his hair. Todd ensures that each step is performed in the specified order, allowing his hair to naturally dry during the other steps before applying the product.

SAQ 9.3. Why is the order of instructions important in Todd's pre-party grooming algorithm?

SAA 9.3. The order of instructions is important in Todd's pre-party grooming algorithm because it determines the desired outcome and ensures the of the grooming process. For example, washing his hair first allows it to have sufficient time to dry naturally while he proceeds with the remaining steps. If

the order is changed, it may impact the desired result or disrupt the overall grooming routine.

Multiple-Choice Questions and Answers (MCQs and MCAs)

1. What distinguishes an algorithm from a computer program?

- A) An algorithm is machine-readable, while a program is human-readable.
- B) An algorithm is a general solution, while a program is the specific implementation of that solution.
- C) An algorithm is written in a programming language, while a program is written in plain English.
- D) An algorithm only solves math problems, while a program can solve any type of problem.

Answer: B) An algorithm is a general solution, while a program is the specific implementation of that

Explanation: An algorithm provides the steps for solving a problem, while a program is the actual code written to implement the algorithm.

2. What is the primary role of conditional execution in a computer program?

- A) To execute instructions in a fixed order
- B) To skip instructions based on a specific condition
- C) To repeat instructions a set number of times
- D) To define the logic of a program

Answer: B) To skip instructions based on a specific condition

Explanation: Conditional execution allows instructions to be executed or skipped depending on whether a given condition is true or false.

3. In a counting loop, when is the number of repetitions determined?

- A) During the execution of the loop
- B) Before the loop starts
- C) After the loop completes
- D) After each iteration of the loop

Answer: B) Before the loop starts

Explanation: A counting loop is used when the number of repetitions is known in advance, and the program executes the loop that many times.

4. What does the concept of “procedure” in programming allow?

- A) To repeat a block of instructions with no variations
- B) To simplify complex tasks by breaking them into smaller, named units
- C) To execute conditional instructions based on multiple choices
- D) To assign a set of instructions to a single variable

Answer: B) To simplify complex tasks by breaking them into smaller, named units

Explanation: Procedures allow a programmer to break up a task into reusable, named units, which helps in managing complex tasks and improving program clarity.

5. Which control flow structure is used when a program executes the same block of instructions multiple times?

- A) Sequential execution

- B) Conditional execution
- C) Repetitive execution
- D) Event-driven execution

Answer: C) Repetitive execution

Explanation: Repetitive execution, or looping, involves repeating a set of instructions multiple times, either with a counter or until a condition is met.

Tutor-Marked Assessments
(TMAs)

- TMA 9.1.** What do you understand by program logic?
- TMA 9.2.** Differentiate between the conditional execution and sequential execution of a program.
- TMA 9.3.** Explain the difference between a **two-way conditional** and a **one-way conditional** in a program.
- TMA 9.4.** What is a **counting loop**? How does it differ from a conditional loop?

STUDY

SESSION TEN

SOFTWARE ENGINEERING

Introduction

Software engineering is the application of theory, knowledge, and practise to the development of reliable software systems that meet the computing needs of customers and users. It is applicable to small, medium, and large organisations and computing systems. Software engineering consists of several phases for the development of the software, phases that can be combined into development paradigms.

Learning Outcomes for Study Session Ten

When you have studied this session, you should be able to:

1. Explain the phases of software development.
2. Describe the specifications of a software.
3. Discuss the design phase.
4. Perform the software implementation.
5. Explain the testing phase in software.
6. Describe the maintenance phase in software.
7. Describe various software development methodologies such as waterfall, rapid prototyping, and spiral model.
8. Mention various programming languages.
9. Evaluate the purpose of compilation and programming paradigms.

10.1 Phases of Software Development

As discussed in study session 1, software engineering means using a specific method to create a program. The method includes five main tasks: specification, design, implementation, testing, and maintenance.

10.1.1 Specification

The first step in any software development project is specification, which determines exactly what abilities the finished software will have. In this phase, one asks the question, “What will the program do?”

This step is accomplished in different ways, depending on the destination of the software. Some software is created for use by a particular company or client. For example, a bank might have its own programmers develop new software for calculating mortgage amortisation tables. In this case, the programming team can talk directly with the users to see what their needs and desires are.

Other software is called “shrink-wrapped,” which literally means it’s wrapped in plastic that shrinks to the size of the box and, in general, means it’s sold commercially. Here, the programming team can’t simply ask the users what they need because they won’t know for sure who the users are until the software is available on the market. Instead, they have to rely on market research and focus groups to see the program from the user’s point of view. Software specifications detail not only what features the program will have but also what the program’s interface might look like.

10.1.2 Design

In the specification phase, one asks, “What will the program do?” In the next phase, design, one asks, “How will the program do it?” The design phase creates the blueprint for the software’s creation.

In the design phase, the programmers choose a programming language to work in, choose algorithms for different functions of the program, decide which members of the team will do what, make charts describing how the parts of the program will fit together, and so on.

10.1.3 Implementation

In the implementation phase, the program is actually written. The phase is so named because the programmers are implementing the design that was made in the previous step. Most people—beginning programmers included—assume that the majority of software development is in implementation, but it’s actually just a fraction.

10.1.4 Testing

In the testing phase, the programming team determines if the software meets the specifications and the design.

In fact, these are two different concerns. One concern is that the program works without error. The programmers do not want the program to halt prematurely (known as a crash) or to display erroneous results. Another concern is whether the program is what the users want. A program could never

crash and always produce correct output, but if it's missing a key feature, it's not a finished program.

Programmers talk of alpha testing and beta testing. Alpha testing is done by members of the programming team. Beta testing is done by users or potential users.

Many different methods are available for alpha testing. It's not as simple as just running the program and seeing how it works. Consider a program that computes mortgage loan amortisation tables. The inputs to this program are the type of interest accrued, the interest rate, the date the loan starts, and so on. With so many different permutations of inputs, it would be impossible to test them all. Because most programs have more possible inputs than can ever be tested, systematic approaches to testing must be followed.

Two main categories of testing are white box testing and black box testing.

With white box testing, the programmer uses his or her knowledge of the program to pinpoint locations of possible failure. For example, suppose the programmer knows that one of two large blocks of the program will be executed based on some condition. In this case, the programmer contrives different sets of test inputs to force execution on both paths.

With black box testing, the testers do not have any knowledge of the inner workings of the program. This kind of testing relies purely on the specifications of the program. If the program's specification says it should do X when given Y, give it X and make sure it returns Y.

At first, it might seem that white box testing is automatically better than black box testing, but this is not always the case. While having knowledge of the

program's inner workings is useful, it can lead to unwise assumptions. The part of the program that originally appeared "easiest" might have been written less carefully, and, if it is tested less carefully as well, a problem could easily slip through.

10.5 Maintenance

Most programs need support and additional development after they have been released, which is known as the maintenance phase. Support can include training clients on the software, installing the software on a client's computers, or developing additional product documentation. Additional development is needed when errors are encountered. If an error is extensive, an entirely new version may need to be installed. Small errors can be fixed with a patch, which is a section of code that replaces a part of an already installed program and is usually downloaded from the software developer's Website.

Even when no errors arise, the software may still need additional development. If a client purchases new computers with a new operating system, for example, the software developers may need to make modifications to the software. Or the business needs may change, meaning the original specifications are no longer exactly what the client needs. In some cases, changes are so extensive that development must start again in the specification phase.

10.6 Development Paradigms

The phases of development listed in the previous section can be combined in different ways, or paradigms.

10.6.1 Waterfall

This first development paradigm is the most direct. In the waterfall paradigm, the phases are performed in order. This does not mean that one phase must be finished before the next one begins, but that once a phase is finished, it is not returned to.

This paradigm is called the waterfall model because it is like having a fountain with a series of basins at different levels. Water that falls out of one basin fills the one below. Similarly, when one has enough specifications together, even if they are not finished, one can begin design work; once enough of the design is complete, the implementation can begin.

The strength of this paradigm is its simplicity. Its weakness is its assumption that development can proceed in such an orderly and linear manner. For example, specifications can often change in the middle of a project because the user was misunderstood, the implementation of the original specifications would be too expensive, or for any number of other reasons. The waterfall paradigm doesn't easily accept such changes.

10.6.2 Rapid Prototyping

A prototype, in engineering, is a model of a device that can be used in tests before production of the device begins. For instance, car makers often design a prototype of a new car that can be driven at test tracks and displayed at auto shows. The information gathered from these can be used to improve the design before the car is produced on the assembly line.

In computer science, a prototype is a quickly created version of a program that approximates the appearance of the final program but without all the final program's functions. The rapid prototyping paradigm has the programming team create a prototype as early in the process as possible.

This paradigm attempts to overcome some of the weaknesses of the waterfall model. One problem with the waterfall is that it is difficult to get the specifications correct when the user has nothing to look at. Most users are unable to visualise exactly what they want in a program interface, for example, but they will recognise a good interface when they see it. Often, the programming team spends considerable time creating detailed specification documents only to discover some fundamental misunderstanding has rendered much of the work useless.

With rapid prototyping, the team is not obsessive about capturing every detail in the initial specifications. Instead, the specifications and design are painted with a broad brush to create a quick prototype. The users are shown the prototype, and their reactions to it help clear up any misunderstandings they may have had. Then more detailed specifications are created, and development proceeds from there. Thus, this paradigm progresses through the first three phases twice, once quickly and then again more slowly and carefully.

Prototyping also helps clarify the design. For example, if the program will use a technique the team hasn't used before, the prototype gives the team experience with the technique in a "throwaway" program. The team may then

decide they need more information on the technique and additional training, or they may decide to return to another technique they know better.

One drawback to this paradigm is that creating the prototype adds to the development time. If the prototype uncovers problems in the specifications or design, the time spent on the prototype easily saves time elsewhere. But if the prototype uncovers no major issues, the team may believe the time was wasted. Also, if the prototype is too good—too close to the user's expectations—the users may press to have development continue using the prototype as a base, which horrifies the programming team because the prototype was not carefully built.

10.6.3 Spiral

In rapid prototyping, the first three phases are performed twice. In the spiral paradigm, the phases of development are repeated over and over. Initially, the specifications and design are broad, and the resulting implementation is mostly shown, much like a prototype. As development progresses, the documentation becomes more specific and the program more detailed and functional. The term “spiral” is appropriate because it implies that the team circles around the program, coming closer to the desired result with each pass.

The spiral paradigm may seem very similar to rapid prototyping, but there are important differences. With rapid prototyping, the team creates a first program as a demonstration only, while in the spiral paradigm, there is no intention to throw away any of the work that is created.

The spiral paradigm, like rapid prototyping, may suffer from a perception of slow progress.

10.6.4 Extreme Programming

A more recent paradigm that is gaining popularity is extreme programming. This paradigm is built more upon specific tenets than a rigid process plan. Extreme programming requires the programmers to build testing into code modules. That is, when a programmer adds new features to a program, the programmer also adds another block of code to test the first. That way, the entire program can be tested at any time, which aids in regression testing.

Another interesting tenet is team programming. Extreme programmers work in two-person teams sharing one computer. At any given time, one person is programming and the other is checking the work of the first or checking a reference manual.

10.7 Languages

Computers understand machine language, but programmers use higher-level languages to create programs for the computer. A compiler translates from one language to another.

10.7.1 Compilation

In study session 5, we defined machine language as the set of instructions that a particular model of CPU understands. By definition, every program that a computer executes is in the form of its machine language, but that doesn't mean it was written that way. The instructions in a machine language are

trivially simple, which means machine language programs require many instructions to accomplish even a simple processing task. Programming in machine language is therefore a tedious task that should be avoided whenever possible.

You might wonder: If machine language is the language the CPU understands, how can one program any other way?

One can write in another programming language that is easier to use, then translate the program into the machine language of the CPU used to execute the program. These easier-to-use programming languages are called “high-level languages” because they are at a high level of abstraction from the actual machine hardware (see “Levels of Abstraction” in study session 1). The process of translating a high-level language into a machine language is called compilation.

For example, suppose that a programmer needs to write a loan application program for a bank. The bank will run this program on its loan officer’s computer, which uses an Intel Pentium IV processor. The programmer doesn’t want to write directly in the Pentium IV machine language, determines that the high-level language Visual Basic is the best choice for this project, and so writes the program in the Visual Basic language. When a program is in the form of a high-level language, it is called source code, because it is the source of the machine language program that is actually executed. The programmer compiles the source code using the Visual Basic compiler, which translates the program into the Pentium IV machine language. In the form of machine language, the program is known as machine code, or the executable.

The compiler is the central tool of the programmer. Having the compiler places the programmer at a higher level of abstraction. The programmer can, to a great extent, ignore the hardware and concentrate on solving the problem at hand. Before solving the problem, though, the programmer must pick a programming language in which to solve the problem.

10.7.2 Programming Paradigms

The programmer usually works with procedural programming and object-oriented programming to create programs.

Procedural Programming

The original way of making programs, procedural programming, simply defines the control flow necessary to solve a task, using procedures to divide the source code into functional units.

Because procedural programs are similar to the algorithms they implement, this style of programming is considered easier for the beginning programmer to understand. However, pure procedural programming has weaknesses. Its straightforward “list the steps to solve the problem” model worked well when most programs simply consumed input and produced output. Today, most programs are interactive; they sit waiting for the user to tell them what happens next. The procedural paradigm does not match up well with this style of program. Also, procedural programs are often “loose” with data, sharing the data among all parts of the program. Allowing unfettered access to data has problems.

Object-Oriented Programming

In object-oriented programming, instructions are grouped together with the data they operate on. To understand what this means, consider the student records office at a typical university. The registrar is the official in charge of safeguarding students' official records and making any applicable changes to those records. Students come to the registrar and fill out forms for their various requests, such as having a transcript sent to another school, updating the student's address, or changing the student's enrollment in the coming semester. The students have no direct access to the records themselves; they may only hand their forms to the registrar, who then executes the request, which results in returned copies of records, changes in the files, or whatever the request might have been.

This situation illustrates the key principle of object-oriented programming, information hiding, which occurs when data is placed out of direct reach of the data's users. Here, the users are the students, the data are the records, and the registrar lies between them.

Information hiding has many benefits, the first of which is that it maintains the integrity and security of the data. Consider the problems that would occur if the students had direct access to the file room. They might file paperwork that was incorrectly completed because it would not have been reviewed by the registrar. Furthermore, each student would have access to all student files, not just his or her own, which is a serious security and privacy concern. By requiring all requests to go through the registrar, all paperwork can be checked for accuracy before filing, and all students can be required to show

identification before they are provided copies of their records, assuring security and privacy.

Another benefit is the separation of the interface from the implementation. The forms the students must fill out for their requests are the interface of the records system; the forms are all that the students see from the records office. The implementation is the method used to store the files. The files could be stored on paper in cabinets, on microfilm, in a computer database system, or even in different formats for different records. Because the interface and implementation are separate, the students do not need to know how the records are actually stored. Furthermore, the records office can change the filing system at any time without affecting the student's services, as long as the forms the student uses do not change.

A final benefit is reuse. Suppose that instead of only one records office for the entire university, each individual school at the university had its own. If the same forms are used at all records offices throughout the campus, a student who began in the School of Humanities but later transferred to the School of Mathematics will understand how to make requests of the new school's records office because the interface is the same at both.

The object-oriented programming paradigm gives all these benefits. Instructions and data that logically belong together are kept together, which allows sections of code to be reused in other programs without a lot of trouble. Certain procedures act as gatekeepers—like our registrar—which keeps the data secure. The interface is kept separate from the implementation so that, if

a better algorithm for some problem is found, a new piece of program can be substituted without disturbing the rest.

10.8 Popular Programming Languages

Besides choosing a development paradigm, the programmer must also choose the best language for a project. Popular programming languages include C, C++, Visual Basic, Java, and Perl.

10.8.1C

C, a language with a minimal name, has been around since the early 1970s. By modern standards, it is a very low-level language; it has a smaller set of simpler instructions compared to the other languages discussed here. C encourages “terse” programs that are short and often difficult to read for anyone who is not the original programmer. This language is strictly procedural and is most often used for systems programming, creating software that directly interacts with the hardware, or any situation in which performance is paramount. By keeping everything simple, a C compiler can produce tight, efficient machine code, making C a high-performance language.

10.8.2 C++

The name C++ is pronounced as “C plus plus.” In the C language, ++ is an operator that means “to increase by one”; thus, the name can be read as “one more than C.” This reading is appropriate since C++ is an extension of the C language. Almost all of the instructions in C are retained (indeed, most C++

compilers can compile C source code), but new ones have been added to allow the use of the object-oriented paradigm. C++ is thus known as a hybrid language, one that does not enforce a single programming paradigm. The programmer using C++ can use both the procedural and object-oriented paradigms, even within the same program. C++ is a general-purpose language and is used for all types of programming.

10.8.3 Visual Basic

The Visual Basic language was developed by Microsoft, and its history is interwoven with that of the Microsoft Windows series of operating systems. Those systems provided a graphical user interface instead of plain text command entry, which made life easier for the computer user but more complicated for the programmer. Creating the user interface often required much more work than the underlying logic of the program.

Visual Basic changed that. More than just a programming language, it was a complete program design environment that itself extolled all the benefits of an intelligent user interface. It offered a “drag-and-drop” interface design, which allowed programmers to design an interface by mixing and matching parts (like the Mr. Potato Head toy). This capability meant that programmers could spend their time on the core program logic. The language itself was based on an older language called BASIC, which stands for Beginner’s All-purpose Symbolic Instruction Code. BASIC was designed as a teaching language and is easy to learn, with an instruction set resembling English more than other popular

languages like C or C++. Putting all the features together, Visual Basic allowed a programmer with ordinary skills to create a good-looking program quickly.

For these reasons, Visual Basic became very popular for business program development. The language evolved as the Windows operating system went through its revisions, always allowing (or, in some cases, forcing) the programmer to make use of the latest Windows features. Eventually, the language attracted more “hard-core” programmers, who appreciated many Visual Basic features but complained that the language was too simplistic, hid too many important details from the programmer, and lacked important features like full support for object-oriented programming.

Microsoft listened to these complaints, and now Visual Basic is no longer a simple language. In its latest revision, it’s truly a modern, object-oriented programming language. However, these changes have complicated the language, and development is not as easy as it once was. No language can please all programmers.

10.8.4 Java

Java is an object-oriented language with a set of instructions inspired by C++. It was originally developed for use in embedded systems. An embedded system is a computer and program in one self-contained package, for example, a “smart appliance” such as a DVD player. However, Java’s popularity comes not from embedded systems but from its use on the World Wide Web.

Java is popular for the Web because of its distinguishing feature: platform independence. A platform-independent language produces programs that can be executed on different processors and operating systems. Java achieves this by not fully compiling the source code. Rather than producing machine code to run on a specific processor, a Java compiler produces “intermediate code,” which is at a level between source code and machine code. On each system that runs Java programs, a special program called the Java Virtual Machine—called the VM for short—must reside that is specific to that processor and operating system. The VM interprets the instructions in the intermediate code and performs the specific instructions needed for that system.

Platform independence is the key for programs on the World Wide Web because so many different systems use it. Even outside of the Web, the ability to write one program that can run on a Windows, Macintosh, or UNIX system is very powerful, saving time for the software developer. Java’s motto is, “Write once; run anywhere.”

Platform independence comes at a price, though, which is performance. Programs executing in a processor’s machine code run faster than programs that must be interpreted on the fly. If a programmer only needs a program to work for a particular operating system, the performance loss may be enough to push development into another language.

10.8.5 Perl

One last language, Perl, is a scripting language, which refers to a language in which the source code is processed by a program called an interpreter, rather than compiled. Such languages are used for smaller tasks than compiled languages and are often used for “one-off” programs—special tasks that may only need to be done once. Perl is particularly suited for text manipulation, such as finding all the occurrences of a particular word in a document and exporting all the paragraphs in which that word appears to a second file. Perl is often used for programming tasks on a website.

Because Perl programs tend to be short but numerous, the language is very expressive. That is, the language incorporates the features of many other languages so that programs can be written quickly. This means the language doesn’t follow any particular paradigm, and Perl programs are not easy for others to read. As a general-purpose language, these problems would be serious deficiencies, but for the programs Perl is used for, they are not.

10.6 Language Names

Computer scientists enjoy creating names for new technologies and ideas in their discipline, and programming language names are no exception. Some interesting choices include the following languages:

- Ada was named after Augusta Ada Byron, daughter of poet Lord Byron. She worked with Charles Babbage to develop plans for the Analytical Engine, the rudimentary analogue computer described in Study Session 2.

- APL simply stands for “A Programming Language.”
- SNOBOL is slang for an early language for business programming that was named COBOL (Common Business Oriented Language). The name SNOBOL is a play on the name, born from the developers’ lack of faith in their language’s popularity (as in, “it doesn’t have a snowball’s chance in...”).

In-Text Questions and Answers (ITQs and ITAs)

ITQ 10.1: What are the five main tasks involved in software development?

Answer: The five main tasks in software development are specification, design, implementation, testing, and maintenance.

ITQ 10.2: Why is the specification phase important in software engineering?

Answer: The specification phase defines what the software will do, detailing the program’s features and interface, which guides the subsequent design and development processes.

Conclusion

The programmer uses the algorithm to develop an actual program, which is called the source code. The source code must be translated into machine code by a compiler before it can be executed.

Software engineering involves different phases. The first step in any software development project is specification, in which it is determined exactly what

abilities the finished software will have. The design phase creates the blueprint for the software's creation. In the implementation phase, the program is actually written. After it's written, the program must be thoroughly tested. Even once the program is released to a client, it may require modification or support. Various methodologies for software engineering exist that approach these phases in different ways.

Different programming languages exist for different types of programming. Some programming languages are procedural and mimic the structure of the algorithm. Other languages are object-oriented and place instructions and data together in a way that protects the data and encourages code reuse. Some languages are high-level and make quick work of developing the user interface. Other languages are low-level and offer higher performance but require more work from the programmer.

Summary

At the end of this session, you learnt:

1. Explain the phases of software development.
2. Describe the specifications of a software.
3. Evaluate the design phase.
4. Describe the specification of the software.
5. Describe the design phase.
6. Perform the software implementation.
7. Explain the testing phase in software.
8. Describe the maintenance phase in software.

9. Describe various software development techniques.

Self-Assessment Questions and Answers (SAQs and SAs)

SAQ 10.1. What is the first step in any software development project?

SAA 10.1. The first step in any software development project is typically the requirement gathering and analysis phase (specification). This involves identifying and understanding the needs and goals of the software, as well as determining the requirements and constraints that will guide the development process.

SAQ 10.2. How is the specification phase different for software created for a specific company versus shrink-wrapped software?

SAA 10.2. The specification phase can differ for software created for a specific company versus shrink-wrapped software. When creating software for a specific company, the specification phase involves close collaboration with the company's stakeholders to understand their specific needs and tailor the software accordingly. In contrast, shrink-wrapped software is developed for a wider market, and the specification phase focuses on creating a product that will appeal to a broader range of users, often based on market research and industry standards.

SAQ 10.3. What is the purpose of the design phase in software development?

SAA 10.3. The purpose of the design phase in software development is to create a blueprint or plan for the software solution. It involves translating the requirements into a structured architecture and detailed design that outline how the software components will interact and function. The design phase helps ensure that the software solution is well-organised, efficient, and scalable.

Multiple-Choice Questions and Answers (MCQs and MCAs)

1. Which of the following is the first phase in the software development process?

- A) Design
- B) Testing
- C) Implementation
- D) Specification

Answer: D) Specification

Explanation: The specification phase determines the features and functionality that the finished software will have.

2. What is the primary focus of the testing phase in software development?

- A) Ensuring the software has no errors
- B) Ensuring the software meets the specifications
- C) Writing the source code
- D) Providing end-user documentation

Answer: B) Ensuring the software meets the specifications

Explanation: The testing phase checks whether the software works without errors and if it satisfies the initial specifications and design.

3. In the waterfall model of software development, what happens after one phase is completed?

- A) The phase can be revisited and modified.
- B) The process moves to the next phase with no return to previous phases.
- C) The phase is evaluated for completion.
- D) The testing phase is implemented next.

Answer: B) The process moves to the next phase with no return to previous phases.

Explanation: In the waterfall model, each phase is completed sequentially without revisiting earlier phases once completed.

4. Which of the following is a major advantage of the rapid prototyping development model?

- A) It allows for detailed final specifications before development starts.
- B) It enables users to visualize the program early, helping clarify their requirements.
- C) It requires no initial feedback from users.
- D) It works best for large-scale projects.

Answer: B) It enables users to visualise the program early, helping clarify their requirements.

Explanation: Rapid prototyping helps users visualise the software early,

ensuring that misunderstandings or miscommunications are addressed early in the development process.

5. In the spiral development model, what happens as development progresses?

- A) The specifications and design become less detailed
- B) Documentation becomes less relevant
- C) The software becomes more detailed and functional with each iteration
- D) Each phase is performed sequentially with no revisions

Answer: C) The software becomes more detailed and functional with each iteration.

Explanation: The spiral model involves iterative development, where each pass through the phases adds more detail and functionality to the software.

Tutor-Marked Assessments (TMAs)

- TMA 10.1.** Explain a programming philosophy in which instructions are grouped together with the data.
- TMA 10.2.** Describe three phases of software development.
- TMA 10.3.** Explain four popular programming languages.
- TMA 10.4.** Explain the difference between **white box testing** and **black box testing**.

STUDY SESSION

ELEVEN

NETWORK OVERVIEW

Introduction

In study session 1, networks were defined as sets of computers connected together so they could share data. A company can make its current data, such as customer account information, available to all employees at any time. Networks allow resource sharing, file sharing, and enable different users to communicate with each other. Concerning resource sharing, networks can allow one resource, such as a printer, to be used by multiple users. Without a network, a printer would have to be installed for every user who needed one. Files can be shared without a network, but that means putting them on disc and walking them from one user to another (sometimes jokingly referred to as a sneakernet). But with a network, files can be shared without anyone leaving a chair. Users can then easily collaborate from their computers.

Learning Outcomes for Study Session Eleven

When you have studied this session, you should be able to:

1. Explain the network overview.
2. Outline the basic parts of a transmission.
3. Describe the network sizes.
4. Explain the responsibilities of network.

11.1 Network Overview

Networks allow communication. They allow users in different locations to communicate, either in real time (video conferencing) or not (email). In general, a network allows a group of computers to act in concert, as if they were one computer. From a user's point of view, if the network is fast and robust, the locations of data and other users are irrelevant; it's as if all the data on the network is on each user's computer.

11.2 Basic Parts of a Transmission

A network, then, is a mechanism for transmitting data from one computer to another. When data is transmitted across a network, it is referred to as a message. These messages may be fixed in size (a fixed number of bits of data) or of variable size, but there's always some upper limit. Because of this limitation, most blocks of data to be transferred are broken up into multiple messages.

In any single transmission, the computer that transmits the message is the sender, and the computer to which the message is transmitted is the receiver. This terminology refers to a single transmission only. In most cases, each computer is both sending and receiving.

Two computers sending data to each other at the same time is known as a full-duplex transmission. In some situations, both computers can send to each other, but not at the same time, which is a half-duplex transmission. A good example of this is a “push to talk” cell phone. When the operator of the phone pushes the “talk” button, the speaker on the phone cuts off for as long as the

button is pressed. When the phone is used this way, the operator can listen or talk but not do both at the same time. If only one computer can send, it is known as a simplex transmission. If you have a dish on your roof for receiving television signals, you have a simplex communications link. The television data comes from your service provider, which beams it up to a satellite orbiting the Earth, which in turn beams it down to the ground to all the rooftop dishes. However, no mechanism sends data from the dish back up to the satellite; it's a one-way street. Each computer connected to a network is known as a node. Some nodes are users' computers, some are devices (like a printer shared by users of a network), and some exist just to direct traffic to other nodes. The network medium is the physical connection the message crosses to get from sender to receiver. Some networks use multiple media (the plural of the word "medium"). With satellite television, for example, the satellite beams signals through the air to the dish, which is one medium, but then the dish is connected to the television through wires, which is another medium.

11.3 Network Sizes

Networks come in different sizes. Some networks involve only a few machines that are all in the same building. Others involve thousands of computers or more, spread across the globe. The smallest network, a point-to-point connection, connects just two computers (see Figure 11.1). A simple example of this would be someone who has two computers at home and wants to be able to easily share files between them.

A local area network, or LAN, connects computers in a single building or in adjacent buildings.

A wide area network, or WAN, connects computers that are widely separated, either in different parts of a city or in different countries.

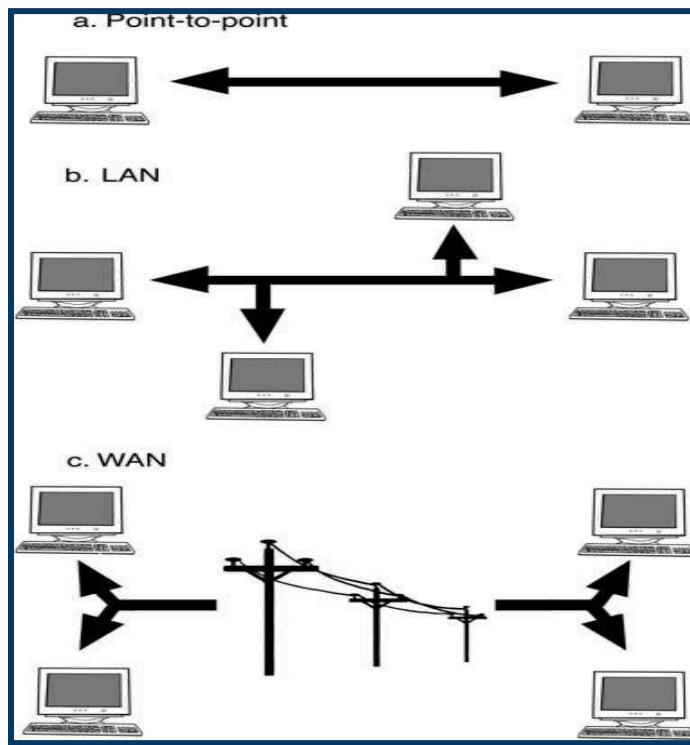


Figure 11.1: Types of Networks

The number of nodes does not determine whether a network is a LAN or a WAN. While the average WAN has more nodes than the average LAN, either could have just a few nodes or many. What makes the difference is the physical separation between the computers. In both a LAN and a WAN, the same organisation owns or controls the use of all the computers on the network. In a LAN, the organisation also controls other network components, such as the medium. In a WAN, the organisation cannot physically string cable to the other

end of a city or continent. Instead, it has to connect to some general carrier to provide the connection.

The situation is analogous to a company phone system. If an employee wants to call someone in the same office, the call can be placed only using equipment that resides in the office. If that employee wants to make a call to someone in the branch office in another state, then some long-distance company like AT&T, Sprint, or MCI provides the connection.

11.4 Network Responsibilities

To be effective, a network has to fulfil expectations related to delivery, reliability, performance, and security.

11.4.1 Delivery

The first responsibility of a network is to deliver the message to the receiver. When the sender and receiver are directly connected to each other, this is not difficult. But in most cases, there are many intermediate nodes between the sender and receiver, and routing is important. This means what it does in general usage, following a “route.” In this case, routing determines the path a message will take from the sender to the receiver, which involves determining the intermediate nodes the message passes through along the way.

11.4.2 Reliability

For a computer network, reliability means that the message that arrives at the receiver is the same message that left the sender. Unfortunately for computer networks, the world is full of things that can corrupt signals. If you have ever

moved a telephone too close to a television or other electrical device and heard static on the line, you've experienced this problem firsthand. All electrical devices emit electromagnetic radiation, which can interfere with networks. In fact, two network connections can even interfere with each other, a phenomenon known as crosstalk. When you faintly hear someone else's phone conversation during your own, you're experiencing crosstalk.

In order to ensure reliability, networks must be designed both to reduce the influence of other devices and to safely handle the situation when a signal is corrupted.

11.4.3 Performance

For networks, performance means that the message arrives at the receiver on time. Note that performance is relative to the application. For example, it's acceptable for an email to take several minutes to arrive at the receiver, but in teleconferencing, a delay of several minutes would make communication between the conference attendees so frustrating as to be unusable. Performance means getting the message there when it needs to be there, whenever that is.

Performance is a function of both the underlying physical media and the network rules for transmission. The amount of data a particular network can transfer in a given time frame is referred to as the network's bandwidth, which is usually measured in bits per second.

11.4.4 Security

Network security encompasses several points. First, a private network must prevent unauthorised access. If people could gain access to a bank's network, for example, they could potentially access the sensitive financial data of the bank's customers. Even on a public network like the Internet, security is a concern. If you do any shopping on the Internet, you know that at some point you are sending your credit card number in a message over the network. While you may trust the message's receiver—the merchant you are buying from—with that data, the message passes through many nodes along the way, nodes that you don't know or trust. Thus, a mechanism must exist for passing sensitive data through unknown hands.

In-Text Questions and Answers (ITQs and ITAs)

ITQ 11.1: What is the primary function of a network?

Answer: The primary function of a network is to allow communication between computers, enabling data and resource sharing and allowing users to collaborate without leaving their desks.

ITQ 11.2: What is the difference between full-duplex, half-duplex, and simplex transmissions?

Answer:

- **Full-duplex transmission** allows both computers to send and receive data simultaneously (e.g., phone calls).

- **Half-duplex transmission** allows data to flow in both directions, but not at the same time (e.g., push-to-talk radios).
- **Simplex transmission** allows data to flow in only one direction (e.g., satellite TV signals).

Conclusion

Networks allow users to share resources, files, and data, and to communicate—all without leaving their computer. Networks come in all sizes. A point-to-point connection connects just two computers. A local area network (LAN) connects computers in a single building or in adjacent buildings. A wide area network (WAN) connects computers that are widely separated. A transmission of data involves a sender, a message, and a receiver. Networks are commonly connected with twisted pair, coaxial cable, or optical fibre, which are called the network media. In general, the less expensive the media, the less bandwidth the network has, and the more susceptible it is to interference. Twisted pair is cheap but fairly slow, and optical fibre is faster but much more expensive. Wires are also easier to work with than optical fibre. Fibre is often used for high-speed backbones connecting sets of computers connected by wires. Other networks transmit data through the air and are called “wireless.” Such networks are useful when the devices on the network will not remain in the same location for long.

Summary

In this session, you have learnt to:

1. Explain the network overview.
2. Outline the basic parts of a transmission.
3. Describe the network sizes.
4. Explain the responsibilities of network.

Self-Assessment Questions and Answers (SAQs and SAs)

SAQ 11.1. What are the roles of the sender and receiver in a network transmission?

SAA 11.1. The sender is responsible for initiating the transmission by sending the message, while the receiver is the intended destination that receives and processes the message.

SAQ 11.2. What is the difference between full-duplex and half-duplex transmissions?

SAA 11.2. Full-duplex transmission refers to a situation where two computers can send data to each other simultaneously. In contrast, half-duplex transmission allows both computers to send data, but not at the same time. An example of half-duplex transmission is a "push-to-talk" cell phone, where the user can either listen or talk but not do both simultaneously. SAA11.3.

Networks can vary in size, ranging from small networks connecting a few

machines within a building to large-scale networks spanning across multiple buildings, cities, or even countries.

SAQ11.3. What is the difference between a LAN and a WAN?

SAA 11.3. A LAN (Local Area Network) connects computers within a single building or in adjacent buildings, typically owned and controlled by the same organisation. A WAN (Wide Area Network), on the other hand, connects computers that are widely separated, often in different parts of a city or different countries.

Multiple-Choice Questions and Answers (MCQs and MCAs)

1. Which of the following best defines a Local Area Network (LAN)?

- A) A network that connects computers across a large geographic area
- B) A network that connects computers in a single building or nearby buildings
- C) A network that connects only two computers
- D) A network that uses satellite connections for data transmission

Answer: B) A network that connects computers in a single building or nearby buildings

Explanation: A LAN connects computers within a limited physical area like a building or campus.

2. In a half-duplex transmission, what happens?

- A) Data is transmitted in both directions at the same time.
- B) Data can only be sent in one direction at a time.

- C) Data transmission is blocked.
- D) The data transmission is one-way only.

Answer: B) Data can only be sent in one direction at a time.

Explanation: In half-duplex communication, the data flows in both directions but not simultaneously.

3. Which of the following is NOT a responsibility of a network?

- A) Delivery
- B) Reliability
- C) Performance
- D) Hardware maintenance

Answer: D) Hardware maintenance

Explanation: Hardware maintenance is not a network responsibility; networks are concerned with delivery, reliability, performance, and security.

4. What type of network is used to connect computers across large distances, such as different cities or countries?

- A) LAN
- B) PAN
- C) WAN
- D) MAN

Answer: C) WAN

Explanation: A WAN (Wide Area Network) connects computers over large geographical distances.

5. What is crosstalk in networking?

- A) The process of securely transmitting sensitive data
- B) Interference caused by the electromagnetic radiation of other devices
- C) The method used for routing data through the network
- D) A communication protocol for transferring messages

Answer: B) Interference caused by the electromagnetic radiation of other devices

Explanation: Crosstalk occurs when signals from different communication lines interfere with each other due to electromagnetic radiation.

Tutor-Marked Assessments (TMAs)

- TMA 1.1.** Discuss the role of **performance** in network operations, including how bandwidth affects it.
- TMA 1.2.** Compare **wired networks** with **wireless networks**, considering cost, speed, and mobility.
- TMA 1.3.** Explain the importance of **reliability** in a network and describe the mechanisms that can ensure it.
- TMA 1.4.** Write short notes on the following:
 - Network sizes
 - Network responsibilities
 - Differentiate between reliability and performance

STUDY SESSION

TWELVE

NETWORK COMPONENTS

Introduction

The components of computer networks include both the hardware and software required for installing computer networks in businesses and homes. The server, client, peer, transmission medium, and connecting devices constitute the hardware components. The components of software are the operating system and protocols. Now that you have some understanding of the transmission of data and the expectations for networks, look at the components that make networks work effectively.

Learning Outcomes for Study Session Twelve

When you have studied this session, you should be able to:

1. Explain transmission media.
2. Describe the protocol and its functions.
3. Explain the flow control of a system.
4. Differentiate between encryption and authentication.

12.1 Transmission Media

To transmit data, computers can be connected through twisted-pair wires, coaxial cable, optical fibres, and electromagnetic waves.

12.1.1 Twisted Pair

The simplest way to connect computers is through copper wires. Two wires must be used to make a complete electrical circuit. When you see a pair of wires in a lamp cord, they run in parallel. Originally, network wires were made the same way, but not anymore. Twisted pair refers to two copper wires that are braided together instead of running in parallel.

The wires are braided to enhance reliability. As we noted before, electromagnetic interference is the bane of accurate transmission. However, it turns out that if the wires or a circuit are exposed to the same interference at the same level, the interference tends to cancel itself out. With parallel wires running past some source of interference, one wire is closer to the source than the other and gets a higher level of interference. But when the wires are braided, each wire tends to get the same overall level of interference, reducing the negative effects.

A twisted pair is sometimes covered with a metal mesh, at which point it is called a shielded twisted pair, or STP. This shield tends to reduce interference even more. If it doesn't have this additional protection, it is known as an unshielded twisted pair, or UTP. The main advantage of twisted pairs is their simplicity. It is easy for network technicians to work with because it is very

flexible, can fit in tight spaces, and can easily be cut to different lengths. The technology is also very inexpensive.

12.1.2 Coaxial Cable

Coaxial cable refers to a construction in which one wire is placed inside the other. This is the same cable used for cable television. At the end of one of these cables, you'll see a thick wire in the centre and, around that, a kind of nut that screws onto a connector on the television or cable box. If you cut the cable in the middle somewhere, you would see that the thick wire runs through the middle with some insulation, a ring of metal, and a plastic coating around it. The ring of metal connects to that nut on the end. A coaxial cable makes a complete circuit because the wire in the middle is one wire and the ring around the outside acts as the second wire.

Different grades of coaxial cable are available for different applications. They all have the same basic construction, but the width of the layers varies, or different materials might be used for one of the layers. Coaxial cable has a greater bandwidth than twisted pair. In general, the thicker the wire, the more data it can transmit at once, and cable uses thicker wire than twisted pair.

Coaxial cable has some disadvantages, though, that can offset the higher bandwidth. The cable is so thick that it cannot easily fit around tight corners; it is more expensive, and, compared to twisted pair, it is more difficult to cut and make connectors. One problem that twisted pair and coaxial cables share is that signals in copper wires tend to fade over a distance. This explains why two televisions in a house on the same cable system can have varying reception:

one television has a longer cable than the other. To combat this, network designers install repeaters, which are devices that read a signal on one wire and reproduce it on another. In this way, longer distances can be covered. The different wire types are shown in Figure 12.1.

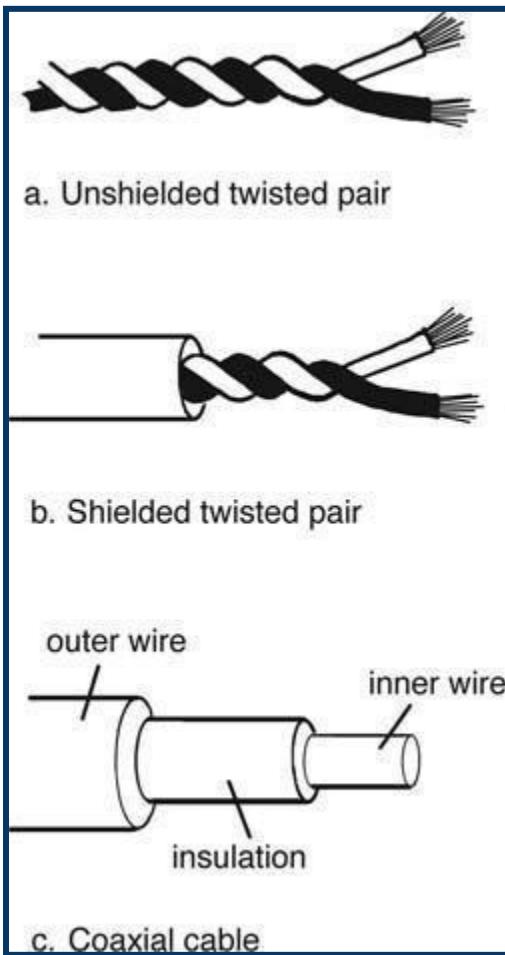


Figure 12.1: Types of wires

12.1.3 Optical Fibre

Optical fibre transmits data as light along a thin glass or plastic filament. The nature of how light refracts in glass or plastic allows this. You may have seen tabletop novelty lights that use plastic filaments. These have ordinary lightbulbs in the base, but sticking out of the base are a number of plastic

strands that look something like porcupines. The ends of these strands glow the same colour as the lightbulb. The strands guide the light from one end to the other. Optical fibre works the same way. At one end of the fibre is a light source, and at the other end is a light detector. To transmit binary data, the light can be turned on for each 1 and off for each 0.

The light source in less expensive systems is a light-emitting diode, something like the light on some keyboards that indicates whether “Caps Lock” is on. More expensive optical fibre systems use a laser as a light source. The laser is better because it produces a tightly focused beam.

Optical fibre has a number of major advantages over copper wire technology. First, because it uses light and not electricity, optical fibre is immune to electromagnetic interference. Second, it has a much higher bandwidth than copper wire. Third, it can run much longer distances than copper wire. While repeaters may still be needed, they can be spaced much farther apart.

Optical fibre does have some disadvantages, though. The main issue is cost. The fibre itself is more expensive to make than a wire, and the equipment needed at both ends of a connection is more expensive too. Also, optical fibre is much harder to work with. It is fairly stiff and a lot easier to accidentally damage. A network technician cannot simply splice two fibres together the way two wires can be spliced.

Because of these disadvantages, optical fibre is often used for a backbone, which is a term for a major pathway in a network. For example, if a company has 100 nodes split evenly between two adjacent office buildings, the connection between the two buildings might be optical fibre, while the connections within

each building could be copper wire. The backbone would be optical fibre because it gets more traffic than any other connection. But connecting all the computers in each office building through optical fibre would be expensive and unnecessary.

12.1.4 Wireless

“Wireless” refers to transmitting data via electromagnetic waves through the air. These waves are categorised based on their frequency. Lower frequency waves are called radio waves; these include the frequencies used by AM and FM radio as well as those used by broadcast television. Higher frequency waves are called microwaves.

Wireless technology is not a general replacement for wires or fibre. Any time wire or fibre is practical, it is a better choice than wireless, which is used in special situations like long-distance transmission. In fact, the long-distance telephone network relies on microwave transmission towers.

Another situation is broadcasting, which refers to sending out a single message to multiple receivers at once. Unlike wires and fibre, which direct a signal to a particular destination, wireless transmissions are unguided. They may be aimed in a specific direction, but they are still broadcast to a wide area. The advantage is that any receiver in the broadcast area can get the message. This approach is more efficient than sending the same message to each receiver individually.

A final reason to use wireless is mobility. In a wired network, nodes can only be where the wires are. In a wireless network, the nodes can be anywhere in the transmission area.

Wireless has a couple of problems that other media do not have. One problem is that air is the conduit for transmission, and the characteristics of the air are constantly changing. For example, weather, solar flares, and other natural phenomena more often affect wireless networks. The other problem relates to security. Because the messages are broadcast, they are easily intercepted. This makes the security built into the protocol especially important.

12.2 Protocols

Every network needs a set of rules, or protocols. These function for the network the way traffic rules and knowledge function for the highways. To drive from one place to another, you have to know where the roads are (using a map on paper or in your head), the rules of the road (drive on the right, a red light means stop, and so on), and be able to make decisions while on the road (if an accident stops traffic on the freeway, what is the next best way to get to your destination?). The network protocol does the same thing for the network. Protocol functions include data linking, error detection, routing, and bridging.

12.2.1 Data Linking

A data link is the direct connection between two nodes in a network. In a network with thousands of nodes, a message may go through many intermediates to get to its destination, but that trip is still managed one data link at a time.

The data link part of a protocol is concerned with reliability and ensuring that the message makes it to the other side. A common way to achieve this is through an acknowledgement, or ACK, which is a short message sent in the opposite direction of the original message. If node A sends a message to node B, when B receives it, it sends an ACK back to A. If a specified period of time passes without an ACK, node A assumes something went wrong with the original message and sends the original message again.

Such a scheme gets more complicated when the network is busy. Node A may need to send many messages to node B faster than B can reply to the first message. B must then indicate which message is acknowledged when it sends an ACK, and A must “check off” successful messages and resend the unsuccessful ones. This checking is similar to balancing a chequebook at the end of a month, checking off the cashed cheques.

12.2.2 Error Detection

When node B sends acknowledgements, it should only acknowledge a packet if it arrives in the same form that it left node A. The trick is how node B knows that a message was damaged. If a person receives a wet or smeared paper memo, the damage is easily seen. But a message in a network is just a pattern of bits—1’s and 0’s. Any damage to the message, from the receiver’s point of view, simply changes some 1’s to 0’s or vice versa. The damage is not self-evident.

Error detection refers to rules in the protocol that help identify whether some bits have been changed during transmission. A simple scheme is redundant

transmission. Send the data in the message multiple times. The receiver checks the copies to see if they match. If they do, the ACK is sent, but if the copies differ, then some of the bits must have changed. Some protocols would send a NACK, or negative acknowledgement, to indicate that the message was garbled, but it's not necessary.

Another scheme is called parity, in which the sender counts the 1's in the message, and then attaches a 1 to the end of the message if this count is odd and 0 if it is even. This means that in the final message sent, the number of 1 bits is always even. For example, if the message originally has six 1 bits, a 0 is added, which means there are still six 1 bits, an even number; but if there are seven 1 bits, a 1 is added, making the total eight, an even number. Then the receiver just has to count the number of 1 bits and see if the count is even.

More advanced schemes can even do error correction, which allows the original message to be recovered even with some corruption of bits. Redundancy is one way to do this. If the message is sent five times and four messages arrive alike and the fifth is different, it's safe to assume the fifth message is corrupted and the other four represent the correct message.

Because error correction requires sending so much extra data, it is usually not implemented. The overall performance of the network is better when the occasional corrupted message is sent again. The exceptions to this occur when the transmission time is so long that retransmission becomes a problem. The NASA Mars Rovers are examples of this. Because it takes so long for a message to get from Earth to Mars, or the reverse, it is better to correct the message than to resend it.

12.2.3 Routing

Once the network can successfully transmit between nodes that are directly connected, it needs the ability to transmit from any node in the network to any other node. One of the important details at this level is routing, which determines the path a message will take across the network. A router is a special node that forwards packets towards their destination. Networks are built by connecting users' computers to routers. A single router can support multiple users' computers and then connect routers to each other.

When a router receives a message, it must determine which connection to send it to. This is easy if the destination is a computer it is directly connected to, but if it's not directly connected, it has to make educated guesses. Most routers configure themselves as they go along and do not have a picture of the entire network. The problem is similar to deciding which plane to get on at the airport when none of the flights go directly to your destination city and you don't know the flight schedule for any other airport.

The solution is for the routers to share information with each other. Usually the information shared is what's known as a hop count, which is the number of nodes a message must pass through to reach a particular computer. For example, suppose router W is directly connected to computer A (Figure 12.2). Then W's hop count to A is only one. Suppose W is also connected to routers X and Y. Periodically, W sends all the hop counts it knows to X and Y. When X and Y find out that W's hop-count to A is one, they decide that the hop-count to that computer is two; that is, it will take one hop to get to W and then another hop to get to A.

When X sends out its own hop counts to another router, Z, the other router updates its hop count for A to three, and so on.

This system also allows the network to eventually route around routers that have malfunctioned or have been taken down, albeit slowly. Normally, the router sends the messages to the directly connected node with the lowest hop count to that destination. But if an adjacent router is down, the router picks the next best. The problem is that other routers may still be sending traffic intended to pass through the down node.

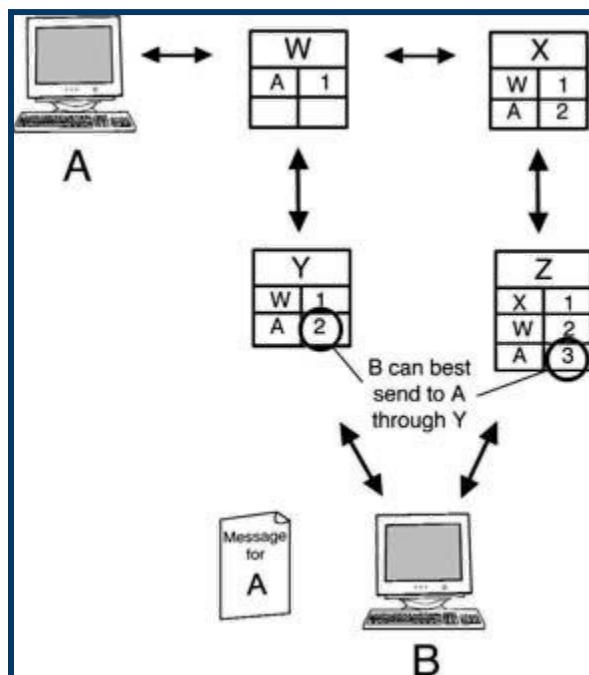


Figure 12.2: Routing

For example, if some distant router thinks the fastest way to send a message to A is to go to Z, it does so even if X is down because it is not directly connected to X and therefore has no way to know this has happened. Z routes this message as best it can, but in the worst case, it may send the message

back the way it came. Eventually, Z's routing table will reflect that it is not the best intermediary for a message going to A, but this takes time.

Because bad news travels slowly around the network, messages that come in at a time when tables are inaccurate could be sent on wayward trips or even caught in an endless loop where routers keep passing the same message among themselves. To keep a message from floating around the network indefinitely, messages generally contain their own hop count, that is, the count of nodes they have passed through already. Once that count exceeds a specified value, it's assumed the message is doomed, and it's discarded.

12.2.4. Bridging

A bridge is a node that logically divides a network into smaller subnetworks. In an office LAN with ten computers, for instance, a bridge could be used to divide the LAN into two LANs of five computers each. Although the two networks are logically separate, nodes in one LAN can send messages across the bridge to nodes in the other LAN. In some types of small networks, because smaller networks are more efficient than larger ones; the use of bridges can improve network performance.

A gateway is a kind of bridge that connects two different kinds of networks. For example, an office may use LAN technology for the interoffice network but want to allow office users to access the Internet, which uses a different protocol. In this case, the gateway translates LAN messages to the Internet protocol and vice versa.

Because gateways are often a LAN's point of connection with "the outside world," they pose special security risks. Much care must be taken to ensure that messages that are not responses to requests from LAN users are not let "into" the LAN.

12.3. Flow Control

Networks must also provide flow control, which is a method for allowing the receiver to control how often messages are sent by the sender. A good example of why flow control is needed is a printer attached to a network. The printer receives messages across the network and stores the data to be printed in RAM. As each page is printed, the space in RAM is reclaimed.

However, even a fast printer takes a few seconds to print a page. During those seconds, the data for several pages arrives. The printer cannot remove data from RAM as fast as it is coming in. Without flow control, the printer's RAM would overflow. Flow control allows the printer to tell the computer sending the print job to "hold on," and later to tell it to send more data.

Sometimes the mechanism for flow control is part of the mechanism for error control. One such mechanism is called stop-and-wait, which is when the sender sends a single message and does not send another until the first message's ACK is returned. This method is slow because the sender spends most of its time waiting, not sending.

A more advanced implementation is called a sliding window, in which the sender and receiver agree to a set number of unacknowledged messages that can be in transit at any time. For example, if that number is five, the sender

sends out the first five messages it wants to send without waiting for an ACK. When the ACK for the first message is returned, the sender sends the sixth message; when the ACK for the second message is returned, the sender sends the seventh message, and so on.

12.4 Encryption and Authentication

Messages sent across a network often contain sensitive information, such as credit card numbers. Unscrupulous people might try to intercept these messages. Preventing the interception is not always possible because the message might pass through nodes that are not under the control of the sender or the receiver. For security, some mechanism must be in place so that intercepted messages are useless to the interceptor.

Encryption refers to encoding a message so it can only be read by the intended receiver. Encryption works by processing the data with a special value called the “key,” which results in a new message. The new message cannot be returned to the original (cannot be decrypted) without the use of a key. The entire process is referred to as cryptography. There are two categories of cryptography.

In private-key cryptography, the encryption key is the same as the decryption key. This approach means that for the sender to transmit an encrypted message, the receiver must already have the sender’s secret key. The sender cannot transmit the key to the receiver because then any possible interceptor of the encrypted message could intercept the key as well. That would defeat the purpose of encryption.

In the second category, public-key cryptography, the encryption key and decryption key are different. Each receiver maintains a private key, which it does not share. Each private key generates a public key, which is then “published,” which means it’s shared with any node that asks for it—by posting it on a public website, for example. Messages that are encrypted with a public key can only be decrypted using the matching private key. For instance, if node A wants to transmit a message to node B, node A uses B’s freely available public key to encrypt the message. The resulting message can only be decrypted by B’s private key. Thus, no matter who intercepts the message, only B can decrypt it.

What makes public-key cryptography work is the mathematical relationship between a private key and a public key. The public key can easily be computed from the private key. However, the reverse computation, though possible, is so time-consuming that it is believed to be intractable even for a computer. (In the 1992, Robert Redford movie *Sneakers*, a scientist supposedly finds a computationally “easy” way to do the reverse calculation, which would make a lot of secure data suddenly insecure.)

Public-key cryptography solves the problem of safely getting the keys to the right nodes, but it introduces another problem, authentication, which means confirming the sender’s identity. Even if messages can only be read by the receiver, that doesn’t mean they were all sent by the expected sender. An imposter could hijack a conversation by sending messages using the receiver’s public key, confusing the receiver. For example, if someone could send messages to your bank that your bank thought came from an ATM you were

using, the messages could tell the bank to transfer money from your account to the imposter's.

Thankfully, authentication can be achieved in several ways (Figure 12.3). One solution is to give the sender a private key and have it publish a public key (just as the receiver must publish its public key). The sender then encrypts messages, first with its own private key and then with the public key of the receiver. Remember that a message locked with one key of a public/private pair must be unlocked with the other member of the pair. Thus, when the receiver runs the message through its private key, the resulting message must be processed again with the sender's public key. Because the message decrypts in the end with the sender's public key, it proves it was originally encrypted with the sender's private key, which only the sender has; hence, it proves the sender sent the message.

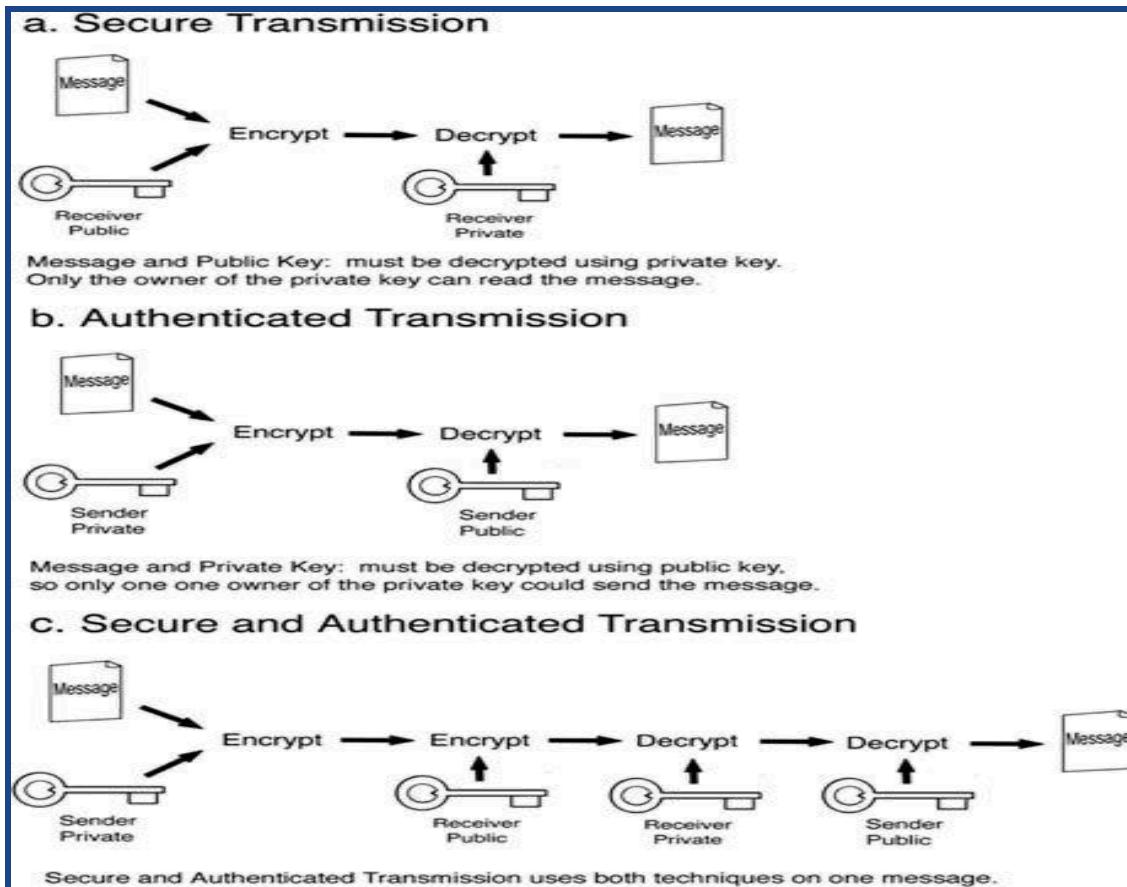


Figure 12.3: Secure and authenticated transmissions

Authentication works well when the senders and receivers all know each other and where to find each other's public keys. If you want to make a purchase over the Internet, though, the merchant can't find your public key, even if you have one. In those cases, a combination of public-key and private-key cryptography is used. A form of public-key cryptography is initially used, and through this, the sender and receiver create a temporary shared secret key. The secret key is used until the sender is done sending messages, and then it is discarded.

12.5 Compression

One way to improve network performance is to send less data over it. Like a highway, a network runs more smoothly when it isn't congested. Of course, asking users to put less traffic on the network to improve performance is self-defeating, but there is another way.

Compression means reducing the size of stored data. Decompression means returning compressed data to its original state. Compression has two types.

Lossy compression loses some part of the original data, so when the data is decompressed, it is a little different than it was originally. Compressed audio and image file formats often use this kind of compression because the loss isn't noticeable to the human ear or eye.

Lossless compression returns the data to its exact original form. Many network protocols do this automatically to reduce traffic on the network. At first, lossless compression seems like a trick. How can data be reduced and yet still keep track of the parts that are gone?

Suppose our college student, Todd, has to call his cable company with a dispute over his bill. Before the company can help him, it wants to know his account number. Unfortunately, Todd's account number is long:
940192949657005987291010220765720239075

Slowly reading this number over the phone takes Todd a long time. The next day, Todd's friend Marta notes a problem with her cable bill and has to make the same call. Marta's account number is just as long, but it has a definite pattern.
555555559999999999999000000000000000000

Instead of actually reading this number out digit by digit, Marta says, “it’s nine fives, then fifteen nines, then fifteen zeroes,” which is a lot faster. What Marta is using is a technique called run-length encoding, in which repetitions of the same number are replaced with a special notation. She could write her account number in a shorthand notation like 9:5, 15:9, 15:0, which takes up a lot less space and can be easily converted to the original form whenever she needs it.

In a computer network, the only numbers involved are 0's and 1's, but the concept is the same. Although other compression schemes may use a technique other than run-length encoding, they all work on the principle of finding some pattern in the data and replacing that data with a shorthand notation for the pattern.

In-Text Questions and Answers (ITQs and ITAs)

ITQ 12.1: What are the primary transmission media used in computer networks?

Answer: The primary transmission media used in computer networks are twisted pair wires, coaxial cables, optical fibres, and wireless electromagnetic waves.

ITQ 12.2: What is the main advantage of twisted pair cables in networking?

Answer: The main advantage of twisted pair cables is their simplicity and cost-effectiveness. They are flexible and easy to install and can fit into tight spaces.

Conclusion

The set of rules for transmission for a particular network is the network's protocol. Mechanisms in the protocol are responsible for making sure that the message actually arrives at the destination and that it hasn't been corrupted from its original content. Protocols also determine how much of the network's bandwidth is in good use and help prevent unauthorised access to network data.

Special devices on networks help direct traffic. A bridge splits a network into two logical networks. A router connects computers and decides what path a message should take to get to its destination. A gateway connects networks with different protocols. To keep messages from being read except by the intended recipient, they are encrypted by the network. In private-key cryptography, the encryption key is the same as the decryption key, so the sender and receiver must somehow be given the key without anyone else knowing what it is. In public-key cryptography, the encryption key and decryption key are different. Any receiver can safely publish its key, allowing any sender to transmit a message that only the receiver can decrypt and read. Public-key cryptography can also be used to authenticate the sender.

Summary

In this session, you have learnt to:

1. Explain transmission media.
2. Describe the protocol and its functions.

3. Explain the flow control of a system.
4. Differentiate between encryption and authentication.

Self-Assessment Questions and Answers (SAQs and SAs)

SAQ 12.1. How does braiding twisted-pair wires enhance reliability in data transmission?

SAA 12.1. Braiding twisted-pair wires enhances reliability in data transmission by providing electromagnetic interference (EMI) shielding. The braided shielding around the twisted pair wires helps to reduce the impact of external electromagnetic signals or noise on the data being transmitted. This shielding prevents signal degradation and improves the overall reliability and quality of the transmission.

SAQ 12.2. What is the difference between shielded twisted pair (STP) and unshielded twisted pair (UTP)?

SAA 12.2. The difference between shielded twisted pair (STP) and unshielded twisted pair (UTP) lies in the presence of shielding. STP has an additional outer metallic shielding layer around the twisted pairs, whereas UTP lacks this shielding. The shielding in STP provides better protection against electromagnetic interference (EMI) and crosstalk, resulting in improved signal quality and reduced data errors compared to UTP. However, STP cables are generally thicker, heavier, and more expensive than UTP cables.

SAQ 12.3. What is the purpose of protocols in a network?

SAA 12.3. The purpose of protocols in a network is to establish rules and standards for communication between devices. Protocols define the format, timing, sequencing, and error handling mechanisms for data exchange. They ensure that devices can understand and interpret the information they receive and enable successful communication within a network.

Multiple-Choice Questions and Answers (MCQs and MCAs)

1. Which of the following is a disadvantage of optical fibre compared to copper cables?

- A) It has a higher bandwidth.
- B) It is immune to electromagnetic interference.
- C) It is more difficult to work with and more expensive.
- D) It can transmit data over longer distances.

Answer: C) It is more difficult to work with and more expensive.

Explanation: While optical fibre has higher bandwidth and longer transmission distances, its cost and handling difficulty make it less practical for certain installations.

2. What is run-length encoding used for in data compression?

- A) To reduce data by encoding the data as binary
- B) To compress data by removing all non-essential data
- C) To shorten repetitive data by using a shorthand notation
- D) To encrypt data for secure transmission

Answer: C) To shorten repetitive data by using a shorthand notation

Explanation: Run-length encoding reduces the size of data by replacing

repeated characters with a shorthand notation indicating the count of occurrences.

3. What is the main function of routing in a network?

- A) To detect errors in the data transmission
- B) To determine the best path for messages to travel across the network
- C) To provide security for data transmission
- D) To manage the flow of data between users

Answer: B) To determine the best path for messages to travel across the network

Explanation: Routing is responsible for directing messages from the source to the correct destination through the most efficient path.

4. Which of the following is a characteristic of shielded twisted pair (STP) cables compared to unshielded twisted pair (UTP) cables?

- A) STP is less expensive.
- B) STP has less susceptibility to interference.
- C) STP is more flexible and easier to install.
- D) STP has a lower bandwidth.

Answer: B) STP has less susceptibility to interference.

Explanation: STP cables have an additional shield to reduce interference, making them more reliable than UTP cables, especially in environments with high electromagnetic noise.

5. What is the primary role of encryption in network security?

- A) To prevent unauthorized access to network devices
- B) To ensure that the data remains unreadable to unauthorized users
- C) To compress data before transmission
- D) To manage the flow of data between network devices

Answer: B) To ensure that the data remains unreadable to unauthorized users

Explanation: Encryption ensures that intercepted data cannot be read unless the correct decryption key is used.

Tutor-Marked Assessments (TMAs)

- TMA 12.1.** Protocol functions include data linking, error detection, routing, and bridging.
- TMA 12.2.** Compression has two types. Explain these two types.
- TMA 12.3.** Differentiate between encryption and authentication.
- TMA 12.4.** Differentiate between **shielded twisted pair (STP)** and **unshielded twisted pair (UTP)**.

STUDY SESSION

THIRTEEN

NETWORK TECHNOLOGIES

Introduction

The technology of networking enables the exchange of data between large and small information systems used primarily by businesses and schools. Network technicians, also known as network engineers or specialists, are accountable for the configuration, installation, and troubleshooting of the technology used to transmit digital data, including audio, video, and data files. Through networking, end-users can transmit files, messages, and other data via email or other channels, sharing information via Internet or Intranet connections based on an organisation's requirements.

Current networking technologies include Ethernet, Frame Relay, and Bluetooth.

Learning Outcomes for Study Session Thirteen

When you have studied this session, you should be able to:

1. Explain Ethernet and LAN.
2. Explain the frame relay for networking.
3. Describe Bluetooth technology.

13.1 Ethernet

Ethernet, one of the most popular LAN technologies, refers to a number of related LAN protocols that use different media and have different capacities. On the low end are 10BASE-T Ethernet, which uses unshielded twisted pair media, and 10BASE-2 Ethernet, which uses coaxial cable. Both support data rates of 10 megabits per second (that's what the 10 in the names refers to), which is a little more than 1 megabyte per second. To put that in perspective, this level of network could transmit the contents of a full CD-ROM in about ten minutes. On the upper end is Gigabit Ethernet, which, as the name implies, can transmit a gigabit, or 1,000 megabits, per second. That means it could transmit an entire CD-ROM in about six seconds. There is even 10 Gigabit Ethernet that would reduce that to half a second! Gigabit Ethernet can use multiple copper wires but most often uses optical fibre.

In Ethernet, nodes compete for the use of the network. As you've seen in other types of networks, each node shares a data link with adjacent nodes, and some nodes act as a message forwarding service. In Ethernet, all the nodes on the network share a common link. All the nodes share the medium, are always listening, and can send messages at any time.

You might wonder how the nodes schedule the use of the network. The answer is that they don't. Any node can use the network as long as it does not detect another message on the network. A collision could occur when two nodes try to send at the same time, resulting in garbled data. Because it takes time for a message to propagate down the line, if node A starts to send a message, there

is a short delay before node B senses this message on the line. If node B sends its own message during this delay, the collision occurs.

Fortunately, nodes can sense when a collision has occurred. The nodes that tried to send will wait a random amount of time and then resend the message if the network is clear. Because the wait is random, usually one node starts retransmitting first, and then the other detects it and waits. In this way, Ethernet allows the nodes to share the network without directly communicating with each other about the details.

If a node were to send out its entire message before a collision occurred, even though it could sense the collision, it would not realise that its own message was affected and would instead assume a message from other nodes caused the collision. Because of this erroneous assumption, it would not resend the message. If the wires connecting the nodes are short, this situation cannot happen because the first part of the message reaches every node before the sender sends the last part of the message. Any collision must occur within that time because nodes only send when they haven't sensed another message on the wire.

The collision domain is the maximum distance of wire between two nodes, so that the beginning of a message from one node reaches the other before the end of the message is sent. When Ethernets are designed, they must ensure that the wiring used doesn't exceed this length from one of the nodes to the other.

Unfortunately, if faster Ethernet versions are used, messages travel faster, and the collision domain is smaller. A smaller domain means a shorter network,

probably with fewer nodes, which sounds like a bad deal because with more nodes, one wants more speed. The solution is bridges, which we discussed previously. By strategically placing bridges throughout the Ethernet, the overall network becomes a set of mini-Ethernets, each one well within the confines of its collision domain.

13.2. Frame Relay

Frame Relay is a high-speed network protocol used for connecting LANs to form a WAN. WANs were originally created by leasing dedicated backbone lines from long-distance service providers. If a company has an office in New York and another in Albany, it would lease the exclusive use of either a bundle of wires or an optical fibre that ran between the two points. This sort of lease was wasteful, though, because the company would have to lease based on its maximum usage of the line. In other words, if it occasionally needed to send a gigabyte of information in a few seconds, the company would have to buy a line that could support that even if most of the time it didn't use the line much at all. When a sender transmits a lot more data than its average rate, on the other hand, it is known as bursty data.

Another problem is that if the company had multiple offices, it would have to lease lines to connect each office to every other office. While a company with two offices needs one line, a company with five offices needs ten.

What was needed was a WAN that could handle bursty data without requiring excess bandwidth and that could simplify connections between multiple offices. Frame Relay does both. A long-distance service provider that offers

Frame Relay is essentially allowing all the WANs it supports to share the use of all of its cables or optical fibres. There's plenty of bandwidth for bursty data as long as all the WANs are not bursty at the same time.

A disadvantage of Frame Relay is that individual messages experience variable delays in reaching their destinations. This delay makes it problematic for applications like real-time video because the image data must be chopped up into small messages to be sent, and a delay in any of the messages can prevent the video from being displayed properly.

13.3 Bluetooth

Bluetooth is a network protocol for short-range wireless communication. It is an open standard, which means any company can make Bluetooth-enabled products. It is named in honour of Harald Bluetooth, king of Denmark in the tenth century, who united Denmark and Norway a millennium ago.

Other wireless networking technologies existed before Bluetooth, but they had a different design goal, such as putting a permanent network in place in a building where it was not feasible to run a lot of cabling. Bluetooth networks are not intended to last very long instead are created spontaneously and disappear quietly.

Previously, wireless networks connected the same sorts of devices as wired networks, such as desktop computers and printers. Bluetooth, though, appears in handheld devices like cell phones, television remotes, and personal digital assistants. There is even a Bluetooth-enabled pen; when the user

writes, nothing shows up on the paper, but the writing can later be transmitted to a computer for storage.

Most networks are installed by skilled technicians, but Bluetooth can't work that way, because the whole point is for the devices to be carried around everywhere. As a consequence, Bluetooth networks have to "install" themselves.

As long as a Bluetooth device is turned on, it is constantly seeking other Bluetooth devices. Each device sends out radio waves in the same frequency range used by cellphones, garage door openers, and others. The signal uses very little power, which limits the range to about thirty feet. When two Bluetooth devices sense each other's presence, they form an ad hoc network called a piconet. One device is designated a master, and the other device is a slave. The master drives all the communication in the piconet.

If other Bluetooth devices come into range, they can be added to the piconet as additional slaves; up to seven slaves can be attached to a single master. If even more devices come into range, multiple piconets can link together, with the master of one piconet playing the role of slave in another, an arrangement known as a "scatternet." When the devices fall out of range of each other, the piconets and scatternets dissolve.

To get an idea of how useful Bluetooth could be if it were widely adopted, consider the following scenario. Marta attends a seminar on robotics given by one of her university's professors. The professor uses a Bluetooth-enabled computer in the presentation, and Marta has brought along her Bluetooth-enabled PDA. She takes no notes because her professor's computer

automatically transmits them to her PDA during the lecture. To get back to her apartment, Marta takes the subway. As she passes the station gate, her PDA receives an updated schedule from the local transit authority. When she gets home and turns on her desktop computer, the professor's lecture is copied to the "Robotics" folder on her computer. This is a scenario that imagines the future, but it's a future that may arrive soon.

In-Text Questions and Answers (ITQs and ITAs)

ITQ 13.1: What are the main advantages of Ethernet technology in networking?

Answer: Ethernet offers high data transmission speeds, flexibility in media choices (e.g., twisted pair, coaxial, optical fibre), and scalability, with versions like Gigabit Ethernet and 10 Gigabit Ethernet providing high-speed connections. It is also cost-effective and widely adopted.

ITQ 13.2: How does Frame Relay handle bursty data in WANs?

Answer: Frame Relay allows multiple WANs to share bandwidth from a service provider, handling bursty data efficiently without requiring excess bandwidth. This is achieved by using a shared infrastructure, ensuring that the network can handle large data bursts when needed, as long as not all networks are sending large amounts simultaneously.

Conclusion

Currently popular networking technologies include Ethernet, which is used for local area networks; Frame Relay, which is used to connect local area networks to form a wide-area network; and Bluetooth, which automatically creates short-range wireless networks as devices come close to each other.

Summary

In this session, you have learnt to:

1. Explain Ethernet and LAN.
2. Explain the frame relay for networking.
3. Describe Bluetooth technology.

Self-Assessment Questions and Answers (SAQs and SAAs)

SAQ 13.1. How do nodes handle collisions in Ethernet networks?

SAA 13.1. In Ethernet networks, nodes handle collisions by sensing when a collision has occurred. The nodes that tried to send will wait a random amount of time and then resend the message if the network is clear.

SAQ 13.2. What assumption does a node make if it sends out its entire message before a collision occurs?

SAA 13.2. If a node sends out its entire message before a collision occurs, it assumes that a message from other nodes caused the collision and does not

realise that its own message was affected. Therefore, it does not resend the message.

SAQ 13.3. What are the advantages and disadvantages of Frame Relay in terms of message delays?

SAA 13.3. The advantages of Frame Relay in terms of message delays include high-speed transmission and the ability to share bandwidth among multiple WANs. However, a disadvantage is that individual messages may experience variable delays in reaching their destinations, which can be problematic for real-time applications like video.

Multiple-Choice Questions and Answers (MCQs and MCAs)

1. What is the main disadvantage of using Frame Relay in a WAN?

- A) It has high latency.
- B) It causes network
- C) It can introduce variable delays in message delivery.
- D) It cannot handle bursty data.

Answer: C) It can introduce variable delays in message delivery.

Explanation: Frame Relay can cause delays in delivering messages, making it less suitable for real-time applications like video conferencing.

2. Which of the following best describes the function of a Bluetooth master in a piconet?

- A) It controls the data transmission and communicates with all devices in the

piconet.

- B) It sends data only when requested by a slave device..
- C) It coordinates the encryption of data sent between devices.
- D) It is only responsible for connecting devices to the internet.

Answer: A) It controls the data transmission and communicates with all devices in the piconet.

Explanation: The **Bluetooth master** controls communication in the piconet and ensures that all data flows through it.

3. What is the primary advantage of optical fibre over coaxial cable and twisted pair cables in networking?

- A) It is more cost-effective for long-distance transmission.
- B) It has a much higher bandwidth and is immune to electromagnetic interference.
- C) It is easier to install and maintain.
- D) It supports faster speeds over shorter distances.

Answer: B) It has a much higher bandwidth and is immune to electromagnetic interference.

Explanation: **Optical fibre** offers higher bandwidth and is immune to electromagnetic interference, making it ideal for long-distance data transmission.

4. Which Ethernet version has the fastest data transmission speed?

- A) 10BASE-T
- B) 10BASE-2

- C) Gigabit Ethernet
- D) 10 Gigabit Ethernet

Answer: D) 10 Gigabit Ethernet

Explanation: **10 Gigabit Ethernet** offers the fastest transmission speed at 10 Gbps, far surpassing other versions.

5. What is a piconet in Bluetooth technology?

- A) A single Bluetooth device connected to a server
- B) A network of multiple Bluetooth devices connected wirelessly
- C) A range of Bluetooth frequencies used for data transmission
- D) A fixed network that connects Bluetooth devices to the internet

Answer: B) A network of multiple Bluetooth devices connected wirelessly

Explanation: A **piconet** is a small network of Bluetooth devices that can communicate with each other wirelessly.

Tutor-Marked Assessments (TMAs)

- TMA 13.1.** Explain the Bluetooth technology used in networking technologies.
- TMA 13.2.** Differentiate between Ethernet and LAN.
- TMA 13.3.** What is a **collision domain** in an Ethernet network?
- TMA 13.4.** Explain what is meant by "bursty data" in Frame Relay networks.

STUDY SESSION

FOURTEEN

THE INTERNET

Introduction

The Internet is a worldwide network that connects many networks in many disciplines. Before you learn about connecting to the Internet and examine its uses, look briefly at the Internet's fascinating history.

Learning Outcomes for Study Session Fourteen

When you have studied this session, you should be able to:

1. Explain the history of ARPANET.
2. Discuss the commercial internet.
3. Explain the applications of the internet.

14.1. ARPANET

The network that became known as the Internet began as a project of the U.S. Department of Defence. In 1969, the department's Advanced Research Project Agency (ARPA) established a wide-area network with two nodes using leased lines called ARPANET, which was called an internetwork, or a network of networks, which was shortened to the term internet. The nodes on this network were universities and private research facilities working under ARPA contracts. The thought was that researchers could better share data if communication were easier.

At first, the network was a raw data transfer device without any of the applications users now take for granted. The applications came later, out of need. By 1972, for example, enough nodes existed that users needed the ability to send text messages about the state and future expansion of the network. Because of this need, someone wrote a program that allowed text messages to be sent from one node to another. Electronic mail was born.

While ARPANET quickly gained popularity in research and academic circles, many years passed before the public at large became aware of it. This delay may have been a good thing because the network was created, in part, to test the ideas of internetworking as well as to use them. The design and protocols of the network went through many iterations, some of which were so fundamental as to cause a flag day, which in computer science jargon is a change that is not backwards compatible, thus requiring all users to implement the change on the same day. Such changes were troublesome

enough when there were only a few nodes on the network, but they would be almost impossible now.

14.2 Commercial Internet

ARPANET spawned imitators, creating internetworks for those not working for ARPA. Soon, it seemed every group of researchers had its own internet. One internet site, for instance, was devoted to magnetic fusion researchers working for the U.S. Department of Energy. Another new Internet was NSFNET, founded in 1986 by the National Science Foundation (NSF). The original purpose of this network was to connect five supercomputers that the foundation had commissioned at schools across the country, but this goal was quickly amended to support all forms of scholastic use. The NSFNET used the protocols of the ARPANET but had a much larger scope. It eventually reached every major college and university and became publicly known in a way that ARPANET never was.

These developments soon attracted commercial interest. Even at this early stage, it was easy to see possibilities in the technology beyond campus use. The NSFNET charter, though, limited its use strictly to academic pursuits. In response, private companies developed their own networks to allow commercial traffic. When they arrived, the Internet, with a capital I, was born. The current Internet refers to the worldwide network that developed out of the NSFNET project. The term internet, with a small i, is a generic term for a network that connects other networks.

Because these private networks captured all of the burgeoning commercial traffic, they quickly grew in capacity, to the point where the original internets that inspired them were superfluous. Eventually, both the ARPANET and NSFNET were dissolved, leaving all traffic in the hands of commercial networks.

14.3 Early Applications for the Internet

What drove the popularity of the Internet wasn't its raw ability to transfer data across the globe but the new applications that used that ability. In technology, it's always the software that drives the hardware. Customers wouldn't buy a DVD player if some of their favourite movies weren't available on DVD. What is always needed with a new technology is a killer app, an application that convinces people to buy the technology.

When the Internet was still a cottage industry, it lacked a killer app. Email was useful, but not enough people had email accounts to make it exciting. What the Internet needed was an application that would allow users to publish documents and other files so that anyone who needed them could find them and access them. Then the Internet could become a giant repository of information.

One of the first steps in this direction was FTP, or File Transfer Protocol, which, as the name implies, is a mechanism for transferring files across the Internet. FTP allows users to publish files on a particular computer and allow others to copy these files to their own computers. Files could be made available to all or protected by a password.

However, to access an FTP file, users would have to know the address of the computer where it resided. Just as with RAM, where every byte of memory has an address, each computer on the Internet has an address. If users aren't given the address where a particular file is located, they can't find the file. The situation would be akin to a library whose books are ordered on the shelves by ISBN instead of by subject and author. No one would be able to find anything that way without using the library's catalogue.

The next step, then, was creating a catalogue for the FTP files. In 1989, McGill University in Montreal developed a tool called "Archie," the name of a play on the words "archive" and "archivist." This tool would regularly contact all the FTP file locations it knew about, request their file directories, and compose the file list into a searchable index. Thus, Archie can be considered the first Internet search engine. (A search engine is an application that allows users to find particular records or files from a large collection.)

The tool did not solve all the problems. It could only search the FTP addresses it had been given; it could not find new FTP locations on its own. Also, Archie wasn't easy for nontechnical people to use. In addition, the program was developed for UNIX systems and, like most UNIX programs of that era, depended upon a certain level of user savvy.

Another problem is that although Archie allowed searching for files, it did not allow browsing. To continue the library analogy, searching with Archie is like using the card catalogue to find titles on a specific subject, and browsing is like wandering the stacks to see if any of the titles on the spines sound interesting. Also, libraries often make displays—like "Fun Books about the

Olympics!"—but there's no equivalent way to group titles using FTP and Archie. In short, no real system of organisation existed among the FTP files.

Then, in 1991, the Gopher system was introduced, so named because it was developed at the University of Minnesota, whose mascot is a golden gopher. Instead of allowing users to simply publish files, Gopher was a mechanism for publishing both files and menus on the Internet. Rather than being presented with a list of files, the user accessing a Gopher site was given a menu with options like "1. Files on Prussian Military History." Menus could have submenus, and submenus could have further menus below them, and so on (Figure 14.1). Thus, Gopher was easily used by nontechnical people and allowed easy browsing. This system also allowed one Gopher site to reference another. Users who published a list of Prussian military history files could include menu items that would take them to other Gopher sites that had files on the same subject.

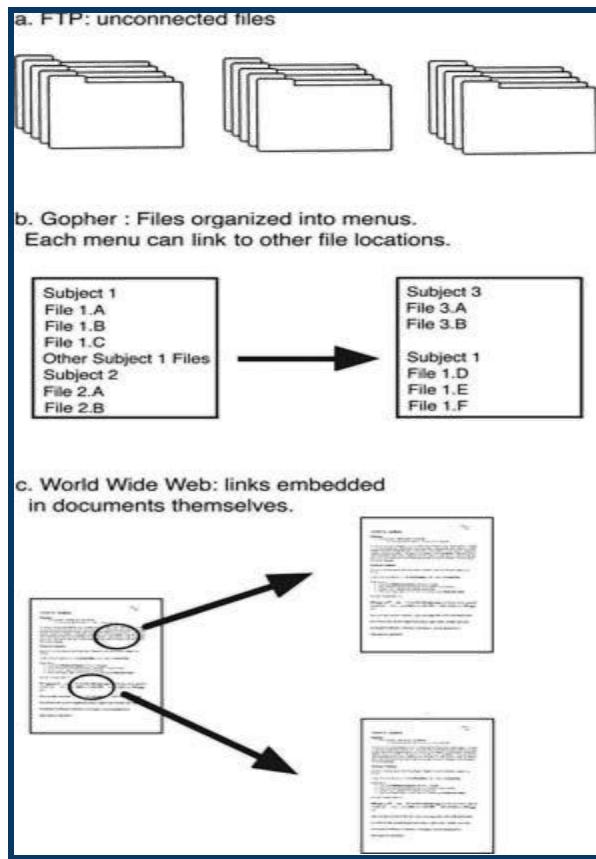


Figure 14.1: FTP, Gopher, World Wide Web

As with FTP, search engines were developed for Gopher, the first being called “Veronica.” Officially, this was an acronym for “Very Easy Rodent Oriented Netwide Index of Computerised Archives,” but the name, clearly inspired by the FTP search engine Archie, was a play on the character Veronica from the Archie comic book. (Another search engine introduced later was called “Jughead.”)

Because Gopher sites referenced each other, it allowed search engines to find new Gopher locations on their own. A spider is a program that “crawls” through references on Internet sites (like those in Gophers) to find new sites and material. Through the use of spiders, search engines could eventually index all the Gophers in existence.

All the pieces were falling into place, but the Internet still lacked its killer app. Though Gopher was easier to use than FTP, it was still far from the mouse-based graphical interfaces that users enjoyed on the Macintosh and Windows.

While Gopher was being developed, a physics researcher, Tim Berners-Lee, led a team that suggested organising documents using hypertext. A hypertext document contains embedded links to other documents. This is similar to the Gopher concept except that the documents and references to other documents are integrated, not separated into menus and files. Note that at this stage, the documents were all plain text, and the links between documents were chosen using the keyboard, not the mouse.

The final piece came into place with Mosaic in 1993, which was a hypertext program that displayed documents graphically and in which users selected links to other documents with the mouse.

Soon everyone was creating hypertext documents and linking them to each other. The links among these documents formed a kind of spiderweb, and the entire collection of interrelated hypertext documents is now called the World Wide Web, or WWW, often just called “the Web.”

The Web was the Internet’s killer app. While the Internet had long been the playground of researchers, college students, and hard-core computer enthusiasts, it now had a purpose that was so powerful and easy to use that the public couldn’t help but notice. The popularity of the Web caused explosive growth of the Internet. The number of connected computers seemed to double overnight. Even today, to many people, “the Internet” and

“the Web” are the same thing, because to them, there’s no point to the Internet without the Web.

14.4 The Future of the Internet

Where does the Internet go from here? While some researchers are looking for more killer apps for the existing Internet, others believe it’s time for a new, faster Internet. To this end, a consortium of universities and government agencies is working to develop Internet2, the next version of the Internet.

In a way, this is history repeating itself, with universities banding together to test technologies that may one day be used by everyone. The current network created by Internet2 is called “Abilene,” named after a railhead in Abilene, Texas, that, when built in the 1860s, was the gateway to the frontier. Abilene provides higher-speed transmissions with lower delays than the current Internet. In a speed trial, the network was able to send the equivalent of a dozen CD-ROMs worth of data in under a minute. While that’s in the same class as Gigabit Ethernet, consider that this wasn’t over a LAN, but over cables that stretched for thousands of miles. The Internet2 group believes important applications exist that can only work on such a network.

One bane of the current Internet is real-time video; for example, having a camera in New York transmit video to a user in Los Angeles. The current Internet is capable of transmitting video, but because of various possible delays and problems in transmission, most video data must accumulate before it can be passed on to the user. This buffering means that the video player waits until it has several seconds (or more) of received images before it begins

playback. The delay allows it to recover from small glitches in transmission without interrupting the user's video playback.

While buffering is fine in some cases, it hampers participation. If you're trying to participate in a conversation that is taking place on the other end of the network, you need the ability to respond to what's happening now, not what happened thirty seconds ago. Furthermore, to reduce problems further, most video sent across the Internet is small and uses few images per second, which results in choppy playback.

Such problems have limited the development of video applications for the current Internet. If technologies from the Abilene project are adopted by the wider community, though, the way people use the Internet could change dramatically again. No one knows what killer apps could exist for the next Internet.

In-Text Questions and Answers (ITQs and ITAs)

ITQ 14.1: How did **ARPANET** evolve into what is now known as the **Internet**?

Answer: ARPANET, established by the U.S. Department of Defence in 1969, was the first network of networks, allowing researchers to share data. Over time, ARPANET expanded, and the development of commercial networks, such as NSFNET by the National Science Foundation in 1986, led to the birth of the Internet. As commercial networks grew, the original research networks like ARPANET and NSFNET were dissolved, consolidating all traffic into the public Internet.

ITQ 14.2: What was the role of **FTP** in the early development of the Internet?

Answer: FTP (File Transfer Protocol) allowed users to transfer files across the Internet. It helped establish the Internet as a tool for data sharing, though it required users to know the exact address of the file. This method of data sharing was foundational for the Internet's growth before more advanced systems, like Gopher and the World Wide Web, emerged.

Conclusion

The Internet began with the ARPANET, a project of the U.S. Department of Defence. The NSFNET was born out of ARPANET, but because it did not permit commercial traffic, private networks using the same protocol quickly supplanted it. These private networks form the backbone of what is now called the Internet.

Early file-sharing applications, such as the file transfer protocol and Gopher, were popular on college campuses and in research centres but unknown to the general public. The World Wide Web, which connected users to a vast array of information through a graphical, mouse-based interface, changed that, and as it grew, so did the Internet.

Summary

In this session, you have learnt to:

1. Explain the history of ARPANET.
2. Discuss the commercial internet.

3. Explain the applications of the internet.

Self-Assessment Questions and Answers (SAQs and SAAs)

SAQ 14.1. What was the significance of electronic mail in the development of ARPANET?

SAA 14.1. Electronic mail (email) was significant in the development of ARPANET as it was one of the first applications that emerged on the network. It allowed users to send text messages from one node to another, and it became one of the key uses of the network.

SAQ 14.2. Why did it take many years for the public to become aware of ARPANET?

SAA 14.2. ARPANET took many years to become known to the public because, initially, its primary focus was on research and academic circles. The network was created to test ideas of internetworking, and its protocols and design went through various iterations before reaching a wider audience.

SAQ 14.3. How did ARPANET influence the development of the commercial internet?

SAA 14.3. ARPANET influenced the development of commercial internets by inspiring the creation of similar internetworks by different groups of researchers and organisations. These internetworks eventually evolved into

commercial networks that catered to broader audiences and facilitated commercial traffic.

Multiple-Choice Questions and Answers (MCQs and MCAs)

1. What was the original purpose of ARPANET when it was created in 1969?

- A) To connect commercial entities for business transactions
- B) To enable easy communication between researchers and universities
- C) To create a worldwide social network
- D) To develop email systems

Answer: B) To enable easy communication between researchers and universities

Explanation: ARPANET was designed by the U.S. Department of Defence to help researchers and universities share data more efficiently.

2. Which application is considered the Internet's killer app that drove its explosive growth?

- A) Gopher
- B) FTP
- C) Email
- D) World Wide Web (WWW)

Answer: D) World Wide Web (WWW)

Explanation: The WWW revolutionised the Internet by providing a graphical and easily navigable system of hypertext documents, significantly increasing user engagement.

3. What was a significant limitation of FTP in the early days of the Internet?

- A) It was too fast for the network infrastructure.
- B) It required users to know the exact address of the file.
- C) It was incompatible with commercial use.
- D) It could not transfer large files.

Answer: B) It required users to know the exact address of the file.

Explanation: **FTP** required users to have precise knowledge of the file's location on the network, making it less user-friendly than later systems like **Gopher** and the **WWW**.

4. Mosaic, created in 1993, was a significant development for the Internet because it:

- A) Enabled email communications across the Internet
- B) Introduced real-time video streaming
- C) Allowed for graphical representation and mouse-based browsing of documents
- D) Created a protocol for file sharing

Answer: C) Allowed for graphical representation and mouse-based browsing of documents

Explanation: Mosaic was the first web browser that allowed users to interact with hypertext documents via a graphical interface, a key factor in the rise of the World Wide Web.

5. Which of the following best describes Internet2?

- A) A new version of the Internet with lower speeds and higher latency
- B) A private network for researchers and universities that supports higher-speed transmissions
- C) A platform for online shopping
- D) An alternative to the **World Wide Web**

Answer: B) A private network for researchers and universities that supports higher-speed transmissions

Explanation: Internet2 is designed to provide higher-speed data transmissions, aiming to support future applications requiring low-latency and high-bandwidth connections.

Tutor-Marked Assessments
(TMAs)

- TMA 14.1.** Write short note on the historical development of ARPANET.
- TMA 14.2.** Describe the early applications of the internet.
- TMA 14.3.** What was the significance of the Gopher system in Internet history?
- TMA 14.4.** Describe the function of early Internet search engines like Archie and Veronica.
How did they solve the problem of locating files?

STUDY SESSION

FIFTEEN

CONNECTING TO THE INTERNET

Introduction

In today's digital age, connecting computers to the Internet is essential for communication, research, business, and entertainment. There are several technologies available for establishing Internet connections, each with its own characteristics, speed, and cost implications.

Common ways to connect computers to the Internet include T-lines, dial-up connections, and cable lines. These technologies represent different stages in the evolution of Internet access. Dial-up technology, which uses telephone lines, was among the earliest and slowest methods of connecting to the Internet. Cable lines, on the other hand, offer faster data transmission by utilising coaxial cables originally designed for television services. T-lines provide even greater reliability and speed, making them ideal for organisations and businesses that require consistent, high-bandwidth connections for data transfer, video conferencing, and network operations.

Exploring these different Internet connection methods offers insight into how modern communication systems operate. It also highlights how continuous technological innovation has improved connection reliability, enhanced data transmission speeds, and expanded access to information across the globe.

Learning Outcomes for Study Session Fifteen

When you have studied this session, you should be able to:

1. Explain the Dial-up connections.
2. Differentiate between T-lines and DSL.
3. Explain the Dial-up connections.

15.1 T-lines

To use the Internet, businesses, schools, and other organisations first must connect their own computers together using LAN technology, such as Ethernet. One of the nodes on the LAN is the gateway. As described in the previous chapter, a gateway translates from one kind of network to another. In this case, the gateway translates from Internet protocols to those of the LAN. Then the gateway must be connected to the Internet. Because all the computers on the LAN must communicate with the Internet through this one connection, the connection needs a lot of bandwidth. A common solution is a T-line, a line leased from a service provider that comes in different levels for different bandwidth needs. (Originally, the provider would have been AT&T, which created these designations.)

A T1 line is usually a twisted pair and provides about 1.5 megabits per second, or about 190 kilobytes per second. That means it could transmit the contents of a CD-ROM in about an hour. At one time, a T1 line was considered fast, but that is no longer the case. A T3 line is a bundle of T1 lines and provides about 44 megabits per second, or about 5 megabytes per second. That means it could transmit the contents of a CD-ROM in about two and a half minutes.

15.2 Dial-up Connections

Home users originally connected to the Internet through ordinary phone lines, known as voice lines in the industry, and often still do. The phone system works on analogue principles: As a person talks on one end, the voice vibrates a speaker coil, which produces an electrical level that varies as the vibrations

vary. On the other end, the electricity is applied to another speaker coil, which produces the same vibrations in the air at that location, reproducing the sound of the original voice. Except for using electricity, the process is the same as the childhood game where two tin cans are connected by a string. Because computers communicate digitally and phone lines are analogue, a conversion is required.

A modem is a device that can convert a digital signal to an analogue signal and vice versa. The name comes from MOdulator/DEModulator; modulation is digital-to-analogue conversion, and demodulation is analogue-to-digital conversion.

When someone uses a dial-up connection, a modem must exist in the computer system and at the other end of the connection. An Internet Service Provider, or ISP, actually provides the connection to the Internet. Each message sent over a dial-up connection is digital until it gets to the sender's modem, where it is converted to analogue, transmitted over the phone line to another modem, which converts it back to a digital message, reformats it to the Internet's protocol, and starts it on its way to its ultimate destination.

Phone lines do not support a high range of frequencies, which is why what people heard over the phone once sounded like they were in a box. This low-frequency range results in low bandwidth. Dial-up connections can only support a speed of 56 kilobits per second, which means it would take more than a day to transmit the contents of a CD-ROM. Even this speed is only available in one direction—from the Internet to the user's computer. In the other direction, the speed is only 33 kilobits per second. The term downstream

refers to the direction from the Internet to the user's computer, and upstream refers to the direction from the user's computer to the Internet. In many methods of communicating with the Internet, the downstream bandwidth is much higher than the upstream bandwidth. In many applications, this is no problem, but some applications, such as video conferencing, require the same bandwidth in both directions.

15.3 DSL

The weakness of a dial-up connection comes from its use of analogue phone equipment, not the wires used for phone lines. A DSL, or Digital Subscriber Line, transmits data digitally over normal phone lines and thus allows much higher data transfer speeds. Unlike a dial-up connection, which is available to anyone with a phone line, DSL requires that your phone company install special equipment. A phone line is a pair of wires leading from a home or office to a building known as a central office, or CO, and each pair from the CO to a phone is known as a local loop. You have probably seen these windowless one- or two-story CO buildings, either with a phone company's name on the outside or no markings whatsoever. The DSL equipment must be installed at a CO to provide DSL service to all the lines that connect to that CO, so the line never passes through analogue switches.

The performance of DSL is heavily dependent on the length of the local loop. The longer the wire, the more the signal degrades, and if the speed isn't lowered, too many errors occur for the connection to be useful. Therefore, the closer the home or business is to a CO, the better the DSL will perform. Unlike a

dial-up connection, because DSL uses a higher range of frequencies than is used for voice communication, it doesn't tie up the phone line. Someone can talk on the phone while the DSL is in use without any interference between the two.

DSL comes in different bandwidth arrangements. DSL for home use is usually ADSL, where A stands for "asynchronous," which has a much higher downstream bandwidth than upstream bandwidth. (If the bandwidth is the same in both directions, the two directions are synchronised; if they are different, they are asynchronous.) This arrangement makes sense for home users because the most common home use of the Internet is the Web, where lots of data in documents is sent down to the user's computer but little data is sent back up (often, just the name and location of the next document the user has selected). A typical ADSL service offers 1.5-megabits-per-second downstream (equivalent in that direction to a T1 line) and 256-kilobits-per-second upstream, which, although slower, is still about eight times faster than a dial-up connection's upstream. DSL services intended for business use offer even higher downstream rates, such as 3 megabits per second or more. At these speeds, they are a good alternative to a T1 line.

15.4 Cable

Another popular way to get a high-speed connection to the Internet is through the "cable company," the same company that provides a home with cable television. Access is through a box known as a cable modem, which, despite the name, is not a modem at all, because there is no digital-to-analogue

conversion. Instead, a cable modem is a gateway, converting between two different LAN protocols so a computer can communicate over the cable line.

The cable modem has two connection ports, one for the cable itself and one for the connection to the user's computer. The computer and the cable modem usually communicate through Ethernet—it's a miniature LAN. Some cable modems connect through a USB port (see study session 5) on the computer. The cable modem communicates with the cable company with a different protocol; like Ethernet, though, the connection is shared with other users. A device called a CMTS, or Cable Modem Termination System, is installed by the cable company for a set of subscribers in an area. The CMTS is another gateway, pulling data from the cable line and translating it into the Internet's protocols, or vice versa. If you've ever used a cable modem, you may have noticed an "activity" light that comes on even when you are not sending to, or receiving from, the Internet. This light indicates some activity between your CMTS and all the cable modems it serves, not just activity involving your cable modem. It is this sharing of the local cable that limits transmission speed. The cable company actually builds transfer limits into the system. If one user could use 100 percent of the cable, the speed would greatly outclass that of DSL, but because it must be shared, the speeds are "throttled" to something similar to DSL, in a range of about 700 kilobits to 2 megabits per second or more. Because the "throttle" is built into the system, the cable company can offer tiers of service. Thus, users can pay for just the bandwidth they need. Both DSL and cable modem connections to the Internet are often referred to as "broadband," but this term is more of an advertiser's talk than

anything that has technical meaning. In networking, “broadband” refers to a signal that uses a wide range (a broad band) of frequencies. In the vernacular, though, broadband simply means a connection that’s a lot faster than a dial-up connection.

In-Text Questions and Answers (ITQs and ITAs)

ITQ 15.1: What is the difference between **T1** and **T3** lines in terms of bandwidth and speed?

Answer: A **T1 line** provides a speed of about **1.5 megabits per second**, which is suitable for smaller-scale connections, while a **T3 line** is a bundle of **T1 lines** and provides about **44 megabits per second**. This makes **T3** lines significantly faster, capable of transmitting large amounts of data more efficiently than **T1** lines.

ITQ 15.2: How does **dial-up technology** work and what are its limitations in terms of speed?

Answer: **Dial-up technology** uses a **modem** to convert digital signals to analogue signals over a phone line and vice versa. It is limited to a maximum speed of **56 kilobits per second** downstream and **33 kilobits per second** upstream, making it much slower than other connection methods. It also ties up the phone line, preventing simultaneous voice calls.

Conclusion

Users connect to the Internet through an Internet Service Provider. Businesses often connect through a T1 line, which is a twisted pair, or a bundle of T1 lines called a T3 line. Business and home users both use DSL lines, which send digital signals across the existing phone lines. Dial-up connections also use the phone line but must convert the computer's digital signals to analogue and back, which limits their effectiveness. Finally, cable television providers also offer Internet connections through their coaxial cables. The Internet uses two protocols, IP and TCP, to provide data transmission services. IP provides a raw and unreliable service, while TCP offers a higher-level service with error correction and retransmission.

Summary

In this session, you have learnt to:

1. Explain the Dial-up connections.
2. Differentiate between T-lines and DSL.
3. Explain the Dial-up connections.

Self-Assessment Questions and Answers (SAQs and SAs)

SAQ 15.1. How does a modem facilitate the connection between a computer and the Internet in a dial-up connection?

SAA 15.1. In a dial-up connection, a modem acts as a device that converts digital signals from the computer into analogue signals for transmission over the analogue phone lines. It also performs the reverse conversion, converting analogue signals received from the phone line back into digital signals for the computer to process.

SAQ 15.2. What is the maximum speed supported by a dial-up connection?

SAA 15.2. The maximum speed supported by a dial-up connection is typically 56 kilobits per second (Kbps) for downstream (from the Internet to the user's computer) and 33 Kbps for upstream (from the user's computer to the Internet).

SAQ 15.3. How does DSL differ from a dial-up connection in terms of data transmission?

SAA 15.3. DSL (Digital Subscriber Line) differs from a dial-up connection in terms of data transmission by allowing digital data to be transmitted over normal phone lines, providing higher data transfer speeds compared to dial-up connections. DSL uses a higher range of frequencies than analogue voice communication, allowing for faster transmission.

Multiple-Choice Questions and Answers (MCQs and MCAs)

1. What is the typical speed of a T1 line?

- A) 44 megabits per second
- B) 1.5 megabits per second
- C) 56 kilobits per second
- D) 700 kilobits per second

Answer: B) 1.5 megabits per second

Explanation: T1 lines provide speeds of about **1.5 megabits per second**, which makes them suitable for small-scale networks.

2. Which of the following is a major limitation of dial-up technology?

- A) Requires a specialised modem for every device
- B) Can only support **56 kilobits per second** maximum speed
- C) Uses fiber optics for faster speeds
- D) Provides simultaneous voice and data transmission

Answer: B) Can only support **56 kilobits per second** maximum speed

Explanation: Dial-up technology is limited to a speed of **56 kilobits per second**, much slower than modern alternatives like **DSL** or **cable internet**.

3. What does DSL allow users to do that dial-up does not?

- A) Transmit data over the air
- B) Use the phone line and the Internet simultaneously
- C) Provide higher speeds than **T3**
- D) Use fiber optic connections

Answer: B) Use the phone line and the Internet simultaneously

Explanation: **DSL** allows users to access the Internet while using the phone line for voice calls, unlike **dial-up**, which uses the same line and blocks phone access.

4. Cable internet connections are referred to as broadband because:

- A) They use a wide range of frequencies for transmitting data
- B) They are primarily used for streaming video
- C) They can connect multiple devices without requiring extra hardware
- D) They provide the same speed as **T1** lines

Answer: A) They use a wide range of frequencies for transmitting data

Explanation: **Cable internet** is called **broadband** because it transmits data using a wide frequency range, allowing it to carry a large amount of data.

5. What is the maximum typical upstream speed for DSL?

- A) 33 kilobits per second
- B) 256 kilobits per second
- C) 1.5 megabits per second
- D) 44 megabits per second

Answer: B) 256 kilobits per second

Explanation: **DSL** typically offers speeds of **1.5 megabits per second** downstream and **256 kilobits per second** upstream.

**Tutor-Marked Assessments
(TMAs)**

- TMA 15.1.** What is the function of a **modem** in a dial-up Internet connection? Why does it limit speed?
- TMA 15.2.** Define **downstream** and **upstream** in Internet connections. Provide examples of applications that rely heavily on each.
- TMA 15.3.** How does a **cable modem** differ from a dial-up modem in terms of signal conversion and network architecture?
- TMA 15.4.** Explain the following internet terminologies:
- T-lines
 - Dial-up connections
 - DSL
 - Cable

STUDY SESSION

SIXTEEN

INTERNET PROTOCOLS

Introduction

The Internet Protocol (IP) is a protocol or set of rules for routing and addressing data packets so they can travel across networks and reach their intended destination. The data that traverses the Internet is broken down into packets. IP information is attached to each packet, allowing routers to send packets to the correct destination. Every device or domain that connects to the Internet is assigned an IP address, and as data packets are routed to the IP address associated with each device or domain, data arrives at its destination. Depending on the transport protocol used in conjunction with IP, the packets are handled differently once they reach their destination. TCP and UDP are the most common transport protocols. The protocols used by the Internet are called IP and TCP and operate at different levels. IP, which simply means Internet Protocol, is the lower-level protocol and allows raw communication between two nodes on the Internet.

Learning Outcomes for Study Session Sixteen

When you have studied this session, you should be able to:

1. Explain the Internet Protocols.
2. Explain the World Wide Web.
3. Describe the HTML web language.
4. Differentiate between HTML tags.
5. Describe web programming.

16.1 Internet Protocols

Internet nodes are identified by a numerical address known as an IP address. This address is written as four numbers in the range of 0 to 255, which makes 256 possibilities (remember “powers of 2” from study session 4) separated by periods; for example, 170.171.1.150. Each node on the Internet at a given time must have an IP address for other nodes to have access to it.

Because remembering the IP address of another computer would be difficult for users, each node is also given a unique domain name. A node cannot directly use a domain name to send data, but it can use it to acquire the IP address it needs. A special computer called a domain name server, or DNS, can return the IP address for a given domain name. For instance, when you type a name like www.broadwaybooks.com into a Web browser, before the browser can retrieve the web page, it must first check the DNS. If you type a nonexistent name into a browser, the error message displayed tells you that the DNS search failed.

Because so many computers use the Internet now, most home users are not given a permanent IP address in the way that a Website is. Instead, each time the computer is connected to the Internet, a computer at the user’s ISP provides a temporary IP from a range of addresses assigned to that ISP.

IP is a low-level protocol that only provides what is known as a best effort delivery service, which means it performs no error checking, does not track the message, and doesn’t ensure that it is delivered. Because this service is not good enough for most applications, a higher-level protocol, called TCP, or Transmission Control Protocol, is used alongside IP to provide reliable

transmission from sender to receiver. TCP provides for message tracking, retransmitting, and error checking.

16.2 World Wide Web

As stated earlier in study session 14, the World Wide Web is the Internet's killer app and what makes the Internet as popular as it is. The Web is successful for several reasons.

First, compared to other ways to share information electronically, the Web allows the publisher of new material to decide how the material is organised and displayed. Just as a phone book is organised differently and has a different style than a treatise on the mating habits of penguins, different subjects on the Web can be organised and displayed in a way most conducive to the material.

Second, Web publishing is cheap. Most ISP accounts provide some amount of Web space at no additional charge. This means that most Web users are potential Web publishers. Even a professional Web hosting contract can be had for well under \$100 a month. Because Web publishing is so inexpensive compared to other ways of disseminating information, the total content of the Web grows quickly.

Third, the Web is easy to use. No special technical knowledge is needed to find information or browse someone's material. Even young children have no problem with the Web.

16.3 HTML

The atom of the Web is a single document called a web page, which is a text document written in a special format called HTML. Groups of web pages stored under the same Internet domain are referred to as websites.

HTML (Hypertext Markup Language) is a combination of plain text and specially formatted instructions that indicate how the text should appear. The term “markup” refers to those special instructions. Just as a typesetter or layout editor may indicate with a red pen how a plain block of text is to be formatted on a page, HTML indicates how a web page is formatted on the screen when viewed through a program called a web browser, or just browser.

16.3.1 A Note on Capitalisation

The term “World Wide Web”, or “Web” for short, refers to a unique global entity and should properly be capitalised. Not every web page is on the Web, though. You could store a copy on your own hard drive for safekeeping, for example, and some companies create groups of web pages for their LAN (company policies, for example), not for publishing on the Internet.

In short, when referring to an HTML document that may or may not be available on the Internet, “web” is fine, but when referring to the World Wide Web itself or anything on it, “Web” is better.

An HTML document is “marked up” using special instructions called tags, which are distinguished in the document using the keyboard symbols and which, in this context, are called “angle brackets.” Here’s a simple example marked up in HTML:

I **< b >**must **< /b >**have doughnuts, *< i >*right now!*< /i >* When displayed, this text would appear as:

I must have doughnuts, right now!

In the HTML, the **< b >** tag indicates “start displaying the text in boldface,” and the **< /b >** indicates “stop displaying the text in boldface.” Similarly, the **< i >** and **< /i >** indicate where to start and stop the display of text in italics.

HTML tags can be divided into four categories.

16.3.2 Character Formatting Tags

Character formatting tags change the appearance of the text itself. The **< b >** and **< i >** tags shown in the above example are character formatting tags. This category includes most of the same options that would appear on the “Font” menu of a typical word processor, including font face, font size, bold and italics, and so on.

Choosing a font, though, is a more complicated decision than with a word processor. When you use a word processor, it only allows you to pick a font that is actually installed on your computer. When designing a web page, a designer cannot know for sure what fonts are installed on all the computers that will view it. If a page designates the use of a font that is not on the computer, the browser picks the font it thinks is the closest match or falls back to a default font. The text is always visible, but it may appear very differently than what the designer intended. Because of these unknowns, designers tend to stick to a few common fonts.

16.3.3 Layout Tags

Layout tags determine how the text flows across the page. These tags say where one paragraph ends and the next begins and are used to divide the page into different sections, for example, to make multicolumn layouts.

Web designers have some of the same issues with layout as they do with fonts. Users may freely resize their browser window, and, if the window is dramatically narrower or wider than the designer expects, the layout could turn out badly.

16.3.4 Link Tags

Links are what make hypertext. HTML can make two kinds of references to material outside the document. One kind is a link to another web page or a different place on the same page, which is called an anchor tag. Anchors can be used to make a link to another page or to make a place where other pages can link or to do both. For example, a page summarising the history of the Olympic Games might have a paragraph for each Olympics with an anchor tag for each so that another page could have a link directly to, say, the 1980 Olympics. Furthermore, that same page might have a series of links at the top so users could jump directly to the part of the history they're interested in. Without any anchors, other pages can only link to the page itself, which means the top of the page.

Another type of link is to other media, like images. Beyond using photographs and other graphics to better convey the material, web page designers use images to enhance the layout and make up for other shortcomings. If a

designer wants to use an unusual font for a title at the top of a page, knowing that it's unlikely that the font will be available on most users' computers, the designer can create an image of that text in the chosen font. Because images require more bytes to store than an equivalent amount of text, such techniques must be used sparingly.

Designers can also embed other media, such as sounds or videos. Finally, the page may link to other documents for which HTML is not an appropriate format. The IRS, for example, makes all the U.S. tax forms and instructions available on its site but doesn't store them in HTML. Instead, it uses the portable document format (PDF), which was developed by the Adobe Corporation to provide document designers with the kind of control unavailable in HTML. It avoids the font problem by allowing the document designer to include the font data in the document. This makes the file much larger, but it's not important what fonts are installed on the user's computer. It avoids the layout problems as well, allowing the document to be scaled to fit the user's window or zoomed in at the user's request. PDF is an excellent choice for standardised forms that may be printed by the user, such as the ones the IRS provides.

16.3.5 Special Tags

Other tags exist for items that don't affect the page itself. For example, the web page can specify what is displayed in the browser's title bar. Other tags, called meta tags, don't have anything to do with the display but provide

information about the page's content. This information can be useful for indexing the pages and may also aid browsers in displaying the page properly.

16.3.6 HTML Tools

Creating web pages is complicated work. It combines the skills of a graphic designer with those of a programmer. Early HTML documents were created using a text editor; the designer had to actually type in all those tags. This approach is troublesome for large pages because, in text form, an HTML document looks very different than when it is viewed in a browser. If you want to see how different they are, most browsers have an option—in Internet Explorer, it's under the “View” menu—to view the “source” of the page, which is the HTML document in text form. Do this for any large page on the Web, and you'll quickly see the difficulty.

Thankfully, tools now allow direct editing of HTML. These programs work like a page layout program or advanced word processor. The designer just drags the text to where it needs to be, sets the font sizes and faces, drops in the images, and so on, and the editor creates the HTML without the designer even having to see it.

Also helpful to designers is the addition of stylesheets to HTML. A stylesheet is a document that an HTML document can reference to determine how its text is displayed. For example, a web page for a news organisation could have a stylesheet that defines a “headline” as a 20-point, bold Times New Roman font; a “byline” as a 12-point Arial font, italicised; and a “story body” as a regular 12-point Arial font. Then a web page with a news story indicates that it is using

that stylesheet, declares that “Costumed Crusader Saves City” is the headline, “Jimmy Dugan, Cub Reporter” is the byline, and the rest of the page is the story body. The browser then formats the text according to the stylesheet. If a Website has only one page, a stylesheet doesn’t save any time. But when a site has dozens or hundreds of pages with similar elements, a stylesheet allows the text formatting to be designated in one place instead of over and over on each page. If the designer later decides that 20-point headlines are too big or not big enough, the only document that needs to be changed is the stylesheet. Without it, every page on the site would have to be changed.

16.4 Browsers

As mentioned earlier, web pages are displayed in an application called a browser. Internet Explorer, Netscape, Mozilla, and Opera are some of the many browsers available. Besides displaying the page, the browser is responsible for retrieving it, usually across the Internet. The browser must also sometimes send user information back across the Internet; this happens when a user has filled out a form. Browsers also contain utility functions to improve the user’s experience, such as maintaining a history of recently viewed pages. To improve performance, most browsers also have a cache. As we noted in study session 5, CPUs have caches to keep recently used data nearby so that a request doesn’t have to be made to main memory if the data is needed again soon. A Web cache employs the same principle, keeping a copy of recently viewed web pages on the user’s hard drive and displaying the copy of a page when the user requests to view it, rather than waiting for it to cross the Internet again. Still,

the most important thing the browser does is display the page. In a perfect world, all browsers would display the same page in exactly the same way. Unfortunately, this is not the case. Early in the history of the Web and HTML, the companies developing browsers were in a hurry to introduce new features, moving faster than the organisation responsible for developing HTML itself. They created their own flavours of HTML (adding their own tags), and this tactic encouraged designers to create pages that wouldn't display properly except in that company's browser.

16.5 Exact layout vs. Guidelines

When the Web and HTML were first gaining momentum, a debate raged over how specific the markup tags should be. Essentially, the debate was between those who thought the tags should specify as much as possible and those who thought they should be more like guidelines.

For example, rather than using a `<i>` tag for italics, those in the latter camp preferred to use an `` tag, which is short for “emphasis.” In most browsers, text marked with `` would still display in italics, but the browser could render it in boldface, in a bright red colour, or in any other way that would clearly emphasise the text. A tag like `` is a logical tag that describes how text relates to the document, rather than how it actually appears.

It appears that the “exact layout” camp won out over the “logical guidelines” camp. For graphic designers, exact layout is a good thing because they have more precise control over the appearance of each page, but for some users, this choice is a bad thing. Exact specifications cause problems with

accessibility, which refers to ease of use for people with disabilities. For example, a designer who uses tags to highlight important blocks of text in a different colour isn't helping a viewer with colour blindness.

The situation isn't quite as bad now. It's rare to find a web page that displays properly in one browser but is unreadable in another. Still, some differences remain, and they're enough to make a good layout turn ugly. Sometimes these differences can exist even between older and newer versions of the same browser. These differences leave the designer with two choices. One choice is to declare that the web page has been designed for a particular browser and that any problems with other browsers are the user's problem. The other choice is to check each page in a cross-section of popular browsers and avoid tags and designs that are known to cause problems.

16.6 Web Programming

Some websites have pages that change so frequently, it's not feasible to run each one through an HTML editor when the changes are necessary. A dynamic web page is a web page that is generated when the user requests it, rather than being created once and subsequently merely copied to the user. Dynamic web pages allow forms to be separated from content even more than style sheets do. For example, in a news organisation, a template for a page can be created that essentially says, "Main story goes here," "Top sports story goes here," "Advertisement goes here," and so on. When the user requests to view the page, the website's server fills in the template with the current stories, a randomly selected advertisement, and everything else.

Dynamic web pages are an example of server-side scripting, which is programming performed on a Web server using a scripting language (such as Perl, discussed in study session 10). Client-side scripting, in contrast, embeds small pieces of programming code in a web page, which is run on a user's machine in

response to user interaction. When you fill out a form on a website and click the "Submit" button, you are executing a piece of client-side scripting.

Server-side scripting is also used for handling user data once it reaches the server. For example, a Website for paying an electric bill may request that the user enter an account number. The server-side scripting can check with the utility company's database to determine if the account number is valid and, if so, retrieve the account data, at which point it generates a dynamic web page to return to the user.

Client-side scripting can perform only basic tasks. To execute a server-side script, it requires a "round trip"; that is, data must travel from the user's computer to the Web server and back again, which takes time. For faster and richer user interaction, some designers include embedded programs that run on the user's computer inside the web page. Two common mechanisms for providing embedded programs are the Java programming language (described in study session 10), which allows the creation of "applets,"—small programs with full graphical interfaces—and Macromedia's Flash software, which allows nonprogrammers to create animations and interactive programs in a style that is well suited to tutorials.

16.7 Email

Email refers to messages, most often text, sent across a network from one user to another or multiple users. On the Internet, email is usually accomplished

through the use of two protocols, SMTP and POP3. SMTP, which stands for Simple Mail Transfer Protocol, is used by the program that sends the message. POP3 is the third version of a protocol called POP, or Post Office Protocol, and is used by the program that receives the message. In addition to receiving the message, the POP3 program is responsible for storing it until the user actually retrieves it on his or her computer.

The SMTP and POP3 programs don't reside on the computers of the users who compose and receive the mail. Instead, each user is given an account on an SMTP or POP3 server provided by the user's ISP. If you look under the setup options of any email client, such as Outlook, you will see where the domains of the SMTP and POP3 servers are specified. Often, the domains have names like smtp.myISP.com, or mail.myISP.com, where "myISP" is the user's ISP.

When a user sends an email, a chain of events takes place. The user's email client communicates with the user's designated SMTP server. The SMTP program locates the IP address of the email's recipient using a DNS server and then sends the email to the POP3 server at that address. The POP3 server receives the message and then stores it on the server. Eventually, the recipient checks his or her email using an email client, and the POP3 server sends the message to that user, erasing its copy.

This mechanism may seem to have more steps and computers involved than is necessary, but email would not work as well as it does without it. Because the POP3 program is on a server that is always running, a user's account can receive an email at any time. If the POP3 program were on the recipient's computer, that user could only receive email when the computer was on. At any other time, attempts to send email to that user would fail. When an email fails to reach its destination, it's known as a bounce because one of the mail servers usually returns the email to the original sender with an explanation. Bounced emails are why the SMTP program must be on its own always-running server too. Because of transient problems in the Internet, a valid email address may be temporarily unavailable. The SMTP server can retry a troublesome email several times before calling it quits. Even after a sender has shut down his or her computer, the SMTP server may still be trying to send out the user's last batch of emails.

Email is a very inexpensive way to communicate, but like all good things, it can be abused. The most common form of abuse is spam, which is unsolicited email messages that are usually commercial in nature and sent in bulk to multiple addresses. Spammers harvest valid email addresses from various sources or even pay legitimate companies for their customer mailing lists. Spammers also "guess" email addresses, appending randomly generated names to known domain names. The spam itself is usually an offer to buy some dubious (or even illegal) product or service.

Spam is an increasing problem. Once enough spammers begin sending to a particular address, the spam will crowd out the real email, making the account almost useless. To protect themselves, users must be very careful about whom they release their addresses to, and they need to use filters designed to delete spam before it hits the user's inbox. Still, no method is completely effective. The only way spam will go away is if no user ever responds to it, which would make it unprofitable for the spammers to send it.

16.8 Chat and Instant Messaging

Chat refers to applications that allow a group of people to type text messages that are seen by everyone in the group. Chat allows clubs and other organisations to exist with members that span the globe.

Some chats are Web-based and run using an embedded program written in a language like Java. A user who has something to say types and hits the “Enter” key, and the Java applet sends the text to the Web server, which then relays it to all the members of the group.

IRC, or Internet Relay Chat, is an Internet-based protocol for chatting. IRC doesn't require the user to go to a particular website but instead to run the IRC client, which works for IRC like an email client works for email. Just as with email clients, users have a number of IRC clients to choose from.

Chatting is an example of how technology has social side effects. People can connect in ways they never could before. Suppose you enjoyed collecting ceramic figures shaped like mariachi bands. There are probably not enough like-minded people in your city, or any city, to form a sustainable Mariachi

Ceramics Club. But there are probably enough people in the world to support such a club. Through applications like Chat, the club could meet online.

A similar application is called instant messaging, in which two users can exchange text messages and have the messages arrive immediately. A single user can receive messages from several other users, but that isn't considered chat because each message only goes to one user at a time. For this reason, it doesn't require a central location to distribute messages; it only requires two users with compatible messaging clients. Instant messaging also allows users to be notified when users on their "friends list" are connected to the Internet.

In-Text Questions and Answers (ITQs and ITAs)

ITQ 16.1: Why is **IP** considered a **low-level protocol**, and how does **TCP** complement it?

Answer: **IP** is considered a **low-level protocol** because it only handles the routing and addressing of packets without checking for errors or ensuring reliable delivery. To address these limitations, **TCP (Transmission Control Protocol)** is used alongside **IP**. **TCP** provides a reliable transmission service by ensuring error-checking, message tracking, and retransmission, thus making sure that data packets reach their destination accurately.

ITQ 16.2: What are the primary advantages of the **World Wide Web (WWW)**?

Answer The **World Wide Web** offers several advantages:

1. **Customisation** of content presentation, allowing information to be organised in a way that best suits the content.

2. **Low cost** for publishing, making it accessible for both individuals and organisations to create content.
3. **Ease of use**, requiring no technical expertise for most users to access and browse information.

Conclusion

The World Wide Web consists of interlinked documents called web pages, which are text messages in a special format called HTML. The HTML format specifies tags commands in the text, enclosed in angle brackets, that indicate how the text should appear and how it should be laid out across the page. Email is sent across the Internet using the SMTP protocol for transmitting messages and the POP3 protocol for receiving them. This arrangement allows the sender and receiver of the message to be offline while the message actually transmits. Chat and instant messaging allow for real-time business collaboration and socialising

Summary

In this session, you have learnt to:

1. Explain the Internet protocols.
2. Explain the World Wide Web.
3. Describe the HTML web language.
4. Differentiate between HTML tags.

Self-Assessment Questions and Answers (SAQs and SAs)

SAQ 16.1. What is the role of a domain name server (DNS)?

SAA 16.1. Domain names are used instead of IP addresses for accessing nodes on the Internet because they are more user-friendly and easier to remember. Domain names provide a human-readable and meaningful representation of an IP address. They allow users to access websites and other resources on the internet using a recognisable name (e.g., www.example.com) instead of a series of numbers.

SAQ 16.2. How are IP addresses assigned to home users?

SAA 16.2. The role of a domain name server (DNS) is to translate domain names into their corresponding IP addresses. When a user enters a domain name in their web browser, the DNS server is responsible for resolving that domain name to the associated IP address. It acts as a directory that maps domain names to IP addresses, enabling the proper routing of network traffic.

SAQ 16.3. What is the purpose of the TCP protocol in relation to IP?

SAA 16.3. Character formatting tags in HTML change the appearance of text by applying specific styles to the text content. For example, the **** tag makes the text bold, the **<i>** tag makes it italic, the **<u>** tag underlines it, and so on. These tags allow web designers to emphasise or differentiate text within the content of a web page.

Multiple-Choice Questions and Answers (MCQs and MCAs)

1. Which of the following best describes an IP address?

- A) A numeric address that helps identify devices on the internet
- B) A protocol for transmitting data over the internet
- C) A system for managing domain names
- D) A server that stores web content

Answer: A) A numeric address that helps identify devices on the internet

Explanation: IP addresses are unique numerical identifiers assigned to devices on the internet, allowing them to communicate and exchange data.

2. Which protocol ensures reliable transmission of data on the internet?

- A) IP
- B) DNS
- C) TCP
- D) HTTP

Answer: C) TCP

Explanation: TCP (Transmission Control Protocol) ensures reliable transmission by checking for errors, managing retransmissions, and tracking message delivery, whereas IP focuses on routing the data.

3. What is the purpose of a DNS (Domain Name Server)?

- A) To host web content
- B) To translate domain names into IP addresses
- C) To send data across networks

D) To manage email services

Answer: B) To translate domain names into IP addresses

Explanation: DNS translates human-readable domain names (e.g., www.example.com) into IP addresses, allowing users to access websites without memorising complex numeric addresses.

4. Which feature of the World Wide Web makes it so widely used?

- A) Its ability to transmit data quickly
- B) Its high cost of publishing
- C) Its ease of use and ability to host information from diverse sources
- D) Its dependence on high-speed internet connections

Answer: C) Its ease of use and ability to host information from diverse sources

Explanation: The **World Wide Web** is popular because it is easy to use, inexpensive to publish on, and allows diverse types of information to be accessed globally.

5. What is HTML primarily used for?

- A) To send emails over the web
- B) To provide dynamic content on web pages
- C) To define the structure and layout of web pages
- D) To manage domain names

Answer: C) To define the structure and layout of web pages

Explanation: HTML is the standard language used for creating and structuring content on web pages, including formatting text, adding images, and creating links.

Tutor-Marked Assessments
(TMAs)

- TMA 16.1.** Describe the way Internet Protocols function.
- TMA 16.2.** Differentiate between a link and a special tag.
- TMA 16.3.** Explain the difference and similarity between World Wide web and HTML.
- TMA 16.4.** What is the purpose of a **domain name server (DNS)**, and why are domain names used instead of IP addresses?

References

- Anderson, C. (2021). *Modern operating systems* (4th ed.). Pearson.
- Bellis, M. (2018). *The history of the IBM PC*. Retrieved from <https://inventors.about.com/library/weekly/aa031599.htm>
- Clark, M. (2019). *Fundamentals of computing: The next generation*. Springer.
- Dawson, R. (2021). *Principles of computer science and technology* (3rd ed.). Oxford University Press.
- Jones, J. S., & Patel, R. (2017). *Introduction to database systems* (8th ed.). Pearson.
- Karlin, S., & Sokal, R. R. (2020). *Computer applications and statistics* (2nd ed.). Wiley.
- Larkin, P., & Fitzgerald, M. (2018). *The digital revolution: New challenges in programming*. CRC Press.
- Russell, S., & Norvig, P. (2020). *Artificial intelligence: A modern approach* (4th ed.). Pearson.
- Ullman, J. D. (2020). *Principles of database and knowledge-base systems* (2nd ed.). W. H. Freeman.

Vogel, H. R. (2021). *Entertainment industry economics* (11th ed.). Cambridge University Press.

Weik, M. H. (2021). *The ENIAC story: Recollections of early computing* (2nd ed.). MIT Press.

Watson, S. (2017). *Computer science and the theory of computation*. Routledge.

West, R. A. (2019). *Digital computing and networking technologies*. Elsevier.

Young, M. T. (2021). *Foundations of computer science*. Springer.

Glossary

Abacus - A mechanical device used for performing arithmetic calculations, consisting of rods or wires with beads that can slide back and forth.

Algorithm - A set of instructions designed to perform a specific task or solve a problem.

Application Software - Programs designed to perform specific tasks for the user, such as word processors or games.

Binary System - A system used by computers, consisting of two states, 0 and 1, to represent data.

Bit - A binary digit, representing the smallest unit of data in a computer, with values of 0 or 1.

Byte - A unit of digital information, typically consisting of 8 bits.

Cache - A small, high-speed storage area used to store frequently accessed data for quicker retrieval.

CD-ROM - A type of optical disc used to store data, which can be read by a computer but not written to.

Central Processing Unit (CPU) - The primary component of a computer that processes instructions and manages operations.

Computer Graphics - The creation, manipulation, and representation of visual images using a computer.

Computer Network - A set of computers and devices connected to share data and resources.

Computer Security - Measures taken to protect a computer system from unauthorized access or attacks.

Data Storage - The process of saving data for future use, involving devices such as hard drives, SSDs, and optical media.

Database - A structured collection of data that is stored and accessed electronically, commonly used in business and other fields.

Disk - A magnetic storage device, such as a hard disk drive, used to store data.

Digital Data - Data that is represented in binary form, suitable for computer processing.

Digital Subscriber Line (DSL) - A technology used to transmit digital data over telephone lines at higher speeds than traditional dial-up connections.

Display - The visual output of a computer system, typically shown on a monitor or screen.

Encryption - The process of converting data into a coded format to prevent unauthorized access.

Hard Drive - A storage device used in computers to store data, typically using magnetic or solid-state technology.

Human-Computer Interaction (HCI) - The study and design of how people interact with computers and to improve usability.

Input Devices - Hardware used to send data to a computer, such as keyboards, mice, and scanners.

Internet Protocol (IP) - A set of rules that govern how data is routed and addressed on the internet.

Kilobyte (KB) - A unit of digital information storage, equal to 1,024 bytes.

Laser Printer - A printer that uses a laser to create an electrostatic image on a drum to print text and images onto paper.

Mainframe - A large, powerful computer used by multiple users simultaneously, typically for enterprise or governmental tasks.

Minicomputer - A computer smaller than a mainframe but still capable of supporting multiple users simultaneously.

Monitor - A screen used to display visual output from a computer system.

Operating System (OS) - Software that manages hardware and software resources and provides services for computer programs.

Random Access Memory (RAM) - Temporary memory used by a computer to store data and instructions that are actively used.