

Digital Electronics
ECE-204 E

L	T	P	Sessional:	50	Marks
3	1	-	Exam:	100	Marks

Unit-1 : Fundamentals of Digital Techniques : Digital signal, logic gates: AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR, Boolean algebra. Review of Number systems. Binary codes: BCD, Excess-3, Gray, EBCDIC, ASCII, Error detection and correction codes.

Unit-2 : Combinational Design Using Gates : Design using gates, Karnaugh map and Quine McCluskey methods of simplification.

Unit-3 : Combinational Design Using MSI Devices: Multiplexers and Demultiplexers and their use as logic elements, Decoders, Adders / Subtractors, BCD arithmetic circuits, Encoders, Decoders / Drivers for display devices.

Unit-4 : Sequential Circuits : Flip Flops : S-R, J-K, T, D, master-slave, edge triggered, shift registers, sequence generators, Counters, Asynchronous and Synchronous Ring counters and Johnson Counter, Design of Synchronous and Asynchronous sequential circuits.

Unit-5 : Digital Logic Families : Switching mode operation of p-n junction, bipolar and MOS devices. Bipolar logic families: RTL, DTL, DCTL, HTL, TTL, ECL, MOS, and CMOS logic families. Tristate logic, Interfacing of CMOS and TTL families.

Unit-6 : A/D and D/A Converters: Sample and hold circuit, weighted resistor and R -2 R ladder D/A Converters, specifications for D/A converters. A/D converters : Quantization, parallel -comparator, successive approximation, counting type, dual-slope ADC, specifications of ADCs.

Unit-7 : Programmable Logic Devices : ROM, PLA, PAL, FPGA and CPLDs.

Text Book :

- Modern Digital Electronics(Edition III) : R. P. Jain; TMH

Reference Book :

- Digital Integrated Electronics : Taub & Schilling; MGH
- Digital Principles and Applications : Malvino & Leach; McGraw Hill.
- Digital Design : Morris Mano; PHI.

Note:: Eight questions are to be set in all by the examiner taking at least one question from each unit. Students will be required to attempt five questions in all.

Programming Languages CSE-204 E

L **T** **P**
3 **1** **-**

Sessional : 50
Exam. : 100

Unit-1: Introduction: A brief history, Characteristics of a good programming language, Programming language translators compiler & interpreters , Elementary data types – data objects, variable & constants, data types, Specification & implementation of elementary data types, Declarations ,type checking & type conversions , Assignment & initialization, Numeric data types, enumerations, Booleans & characters.

Syntax & Semantics : Introduction, general problem of describing syntax, formal method of describing syntax, attribute grammar dynamic semantic.

Unit-2: Structured data objects : Structured data objects & data types , specification & implementation of structured data types, Declaration & type checking of data structure ,vector & arrays, records Character strings, variable size data structures , Union, pointer & programmer defined data objects, sets, files.

Subprograms and Programmer Defined Data Types: Evolution of data type concept abstraction, encapsulation & information hiding , Subprograms ,type definitions, abstract data types, over loaded subprograms, generic subprograms.

Unit-3: Sequence Control: Implicit & explicit sequence control, sequence control within expressions, sequence control within statement, Subprogram sequence control: simple call return, recursive subprograms, Exception & exception handlers, co routines, sequence control. Concurrency – subprogram level concurrency, synchronization through semaphores, monitors & message passing

Data Control: Names & referencing environment, static & dynamic scope, block structure, Local data & local referencing environment, Shared data: dynamic & static scope. Parameter & parameter transmission schemes.

Unit-4: Storage Management: Major run time elements requiring storage, programmer and system controlled storage management & phases , Static storage management , Stack based storage management, Heap storage management ,variable & fixed size elements.

Programming Languages: Introduction to procedural, non-procedural, structured, logical, functional and object oriented programming language, Comparison of C & C++ programming languages.

Text Book:

- Programming languages Design & implementation by T.W. .Pratt, 1996, Prentice Hall Pub.
- Programming Languages – Principles and Paradigms by Allen Tucker & Robert Noonan, 2002, TMH,

Reference Books:

- Fundamentals of Programming languages by Ellis Horowitz, 1984, Galgotia publications (Springer Verlag),
- Programming languages concepts by C. Ghezzi, 1989, Wiley Publications.,
- Programming Languages – Principles and Pradigms Allen Tucker , Robert Noonan 2002, T.M.H.

Object Oriented Programming Using C++
IT-252 E

L **T** **P**
3 **1** **-**

Sessional : 50
Exam. : 100

Unit-1: Introduction to C++, C++ Standard Library, Basics of a Typical C++ Environment, Pre-processors Directives, Illustrative Simple C++ Programs. Header Files and Namespaces, library files. Concept of objects, basic of object modeling, object classes, associations, behaviors, description, Object Oriented Analysis & Object Modeling techniques,

Object Oriented Concepts : Introduction to Objects and Object Oriented Programming, Encapsulation (Information Hiding), Access Modifiers: Controlling access to a class, method, or variable (public, protected, private, package), Other Modifiers, Polymorphism: Overloading,, Inheritance, Overriding Methods, Abstract Classes, Reusability, Class's Behaviors.

Classes and Data Abstraction: Introduction, Structure Definitions, Accessing Members of Structures, Class Scope and Accessing Class Members, Separating Interface from Implementation, Controlling Access Function And Utility Functions, Initializing Class Objects: Constructors, Using Default Arguments With Constructors, Using Destructors, Classes : Const(Constant) Object And Const Member Functions, Object as Member of Classes, Friend Function and Friend Classes, Using This Pointer, Dynamic Memory Allocation with New and Delete, Static Class Members, Container Classes And Integrators, Proxy Classes, Function overloading.

Unit-2: Operator Overloading: Introduction, Fundamentals of Operator Overloading, Restrictions On Operators Overloading, Operator Functions as Class Members vs. as Friend Functions, Overloading, <<, >> Overloading Unary Operators, Overloading Binary Operators.

Inheritance: Introduction, Inheritance: Base Classes And Derived Classes, Protected Members, Casting Base- Class Pointers to Derived- Class Pointers, Using Member Functions, Overriding Base -Class Members in a Derived Class, Public, Protected and Private Inheritance, Using Constructors and Destructors in derived Classes, Implicit Derived -Class Object To Base- Class Object Conversion, Composition Vs. Inheritance.

Unit-3: Virtual Functions and Polymorphism: Introduction to Virtual Functions, Abstract Base Classes And Concrete Classes, Polymorphism, New Classes And Dynamic Binding, Virtual Destructors, Polymorphism, Dynamic Binding.

Files and I/O Streams: Files and Streams, Creating a Sequential Access File, Reading Data From A Sequential Access File, Updating Sequential Access Files, Random Access Files, Creating A Random Access File, Writing Data Randomly To a Random Access File, Reading Data Sequentially from a Random Access File. Stream Input/Output Classes and Objects, Stream Output, Stream Input, Unformatted I/O (with read and write), Stream Manipulators, Stream Format States, Stream Error States.

Unit-4: Templates & Exception Handling: Function Templates, Overloading Template Functions, Class Template, Class Templates and Non-Type Parameters, Templates and Inheritance, Templates and Friends, Templates and Static Members. Introduction, Basics of C++ Exception Handling: Try Throw, Catch, Throwing an Exception, Catching an Exception, Re-throwing an Exception, Exception specifications, Processing Unexpected Exceptions, Stack Unwinding, Constructors, Destructors and Exception Handling, Exceptions and Inheritance.

Text Books:

- C++ How to Program by H M Deitel and P J Deitel, 1998, Prentice Hall
- Object Oriented Programming in Turbo C++ by Robert Lafore ,1994, The WAITE Group Press.
- Programming with C++ By D Ravichandran, 2003, T.M.H

Reference books:

- Object oriented Programming with C++ by E Balagurusamy, 2001, Tata McGraw-Hill
- Computing Concepts with C++ Essentials by Horstmann, 2003, John Wiley,
- The Complete Reference in C++ By Herbert Schildt, 2002, TMH.
- C++ Programming Fundamentals by Chuck Easttom, Firewall Media.

Computer Architecture and organization

CSE – 202 E

L T P
3 1 -

Sessional : 50
Exam. : 100

Unit-1: General System Architecture: Store program control concept, Flynn's classification of computers (SISD, MISD, MIMD); Multilevel viewpoint of a machine: digital logic, micro architecture, ISA, operating systems, high level language; structured organization; CPU, caches, main memory, secondary memory units & I/O; Performance metrics; MIPS, MFLOPS.

Instruction Set Architecture: Instruction set based classification of processors (RISC, CISC, and their comparison); addressing modes: register, immediate, direct, indirect, indexed; Operations in the instruction set; Arithmetic and Logical, Data Transfer, Machine Control Flow; Instruction set formats (fixed, variable, hybrid); Language of the machine: 8086 ; simulation using MASM

Unit-2: Basic non pipelined CPU Architecture: CPU Architecture types (accumulator, register, stack, memory/ register) detailed data path of a typical register based CPU, Fetch-Decode-Execute cycle (typically 3 to 5 stage); microinstruction sequencing, implementation of control unit, Enhancing performance with pipelining. Hardwired control design method, Micro programmed control unit.

Unit-3: Memory Hierarchy & I/O Techniques: The need for a memory hierarchy (Locality of reference principle, Memory hierarchy in practice: Cache, main memory and secondary memory, Memory parameters: access/ cycle time, cost per bit); Main memory (Semiconductor RAM & ROM organization, memory expansion, Static & dynamic memory types); Cache memory (Associative & direct mapped cache organizations. Allocation & replacement policies, segments, pages & file organization, virtual memory.

Unit-4: Introduction to Parallelism: Goals of parallelism (Exploitation of concurrency, throughput enhancement); Amdahl's law; Instruction level parallelism (pipelining, super scaling –basic features); Processor level parallelism (Multiprocessor systems overview).

Computer Organization [80x86]: Instruction codes, computer register, computer instructions, timing and control, instruction cycle, type of instructions, memory reference, register reference. I/O reference, Basics of Logic Design, accumulator logic, Control memory, address sequencing, micro-instruction formats, micro-program sequencer, Stack Organization, Instruction Formats, Types of interrupts; Memory Hierarchy. Programmed I/O, DMA & Interrupts.

Text Books:

- Computer Organization and Design, 2nd Ed., by David A. Patterson and John L. Hennessy, Morgan 1997, Kauffmann.
- Computer Architecture and Organization, 3rd Ed., by John P. Hayes, 1998, TMH.

Reference Books:

- Operating Systems Internals and Design Principles by William Stallings, 4th edition, 2001, Prentice-Hall Upper Saddle River, New Jersey
- Computer Organization, 5th Ed., by Carl Hamacher, Zvonko Vranesic, 2002, Safwat Zaky.

- Structured Computer Organisation by A.S. Tanenbaum, 4th edition, Prentice-Hall of India, 1999, Eastern Economic Edition.
- Computer Organisation & Architecture: Designing for performance by W. Stallings, 4th edition, 1996, Prentice-Hall International edition.
- Computer Architecture & Organisation by M. Mano, 1990, Prentice-Hall.
- Computer Architecture- Nicholas Carter, 2002, T.M.H.

Note: Eight questions will be set in all by the examiners taking at least two questions from each unit .Students will be required to attempt five questions in all at least one from each unit.

Microprocessors And Interfacing

ECE-216 E

L T P
3 1 -

Sessional : 50

Exam. : 100

Unit-1:THE 8085 PROCESSOR : Introduction to microprocessor, 8085 microprocessor : Architecture, instruction set, interrupt structure, and assembly language programming.

MEMORY INTERFACING: Semiconductor memory and its types- Static and dynamic RAM, ROM, EPROM, EEROM and NOVRAM- Interfacing memory- Interfacing SRAM, DRAM, EPROM etc. Timing of RAM and ROM signals.

Unit-2 : THE 8086 MICROPROCESSOR ARCHITECTURE : Architecture, block diagram of 8086, details of sub-blocks such as EU, BIU; memory segmentation and physical address computations, program relocation, addressing modes, instruction formats, pin diagram and description of various signals.

INSTRUCTION SET OF 8086 : Instruction execution timing, assembler instruction format, data transfer instructions, arithmetic instructions, branch instructions, looping instructions, NOP and HLT instructions, flag manipulation instructions, logical instructions, shift and rotate instructions, directives and operators, programming examples.

Unit-3 : INTERFACING DEVICE : The 8255 PPI chip: Architecture, control words, modes and examples. Interfacing D/A and A/D converters

Unit-4: DMA : Introduction to DMA process, 8237 DMA controller,

INTERRUPT AND TIMER : 8259 Programmable interrupt controller, Programmable interval timer chips.

Text Books :

- Microprocessor Architecture, Programming & Applications with 8085 : Ramesh S Gaonkar; Wiley Eastern Ltd.
- The Intel Microprocessors 8086- Pentium processor : Brey; PHI

Reference Books :

- Microprocessors and interfacing : Hall; TMH
- The 8088 & 8086 Microprocessors-Programming, interfacing, Hardware & Applications :Triebel & Singh; PHI
- Microcomputer systems: the 8086/8088 Family: architecture, Programming & Design : Yu-Chang Liu & Glenn A Gibson; PHI.
- Advanced Microprocessors and Interfacing : Badri Ram; TMH

Note: Eight questions will be set in all by the examiners taking at least two questions from each unit .Students will be required to attempt five questions in all at least one from each unit.

C ++ Programming Lab.

IT-256E

L T P
- - 3

Sessional : 50
Practical : 25

Q1. Raising a number n to a power p is the same as multiplying n by itself p times. Write a function called `power()` that takes a double value for n and an int value for p , and returns the result as double value. Use a default argument of 2 for p , so that if this argument is omitted, the number will be squared. Write a `main()` function that gets values from the user to test this function.

Q2. A point on the two dimensional plane can be represented by two numbers: an X coordinate and a Y coordinate. For example, (4,5) represents a point 4 units to the right of the origin along the X axis and 5 units up the Y axis. The sum of two points can be defined as a new point whose X coordinate is the sum of the X coordinates of the points and whose Y coordinate is the sum of their Y coordinates.

Write a program that uses a structure called `point` to model a point. Define three points, and have the user input values to two of them. Then set the third point equal to the sum of the other two, and display the value of the new point. Interaction with the program might look like this:

```
Enter coordinates for P1: 3 4
Enter coordinates for P2: 5 7
Coordinates of P1 + P2 are : 8, 11
```

Q 3. Create the equivalent of a four function calculator. The program should request the user to enter a number, an operator, and another number. It should then carry out the specified arithmetical operation: adding, subtracting, multiplying, or dividing the two numbers. (It should use a `switch` statement to select the operation). Finally it should display the result.

When it finishes the calculation, the program should ask if the user wants to do another calculation. The response can be 'Y' or 'N'. Some sample interaction with the program might look like this.

```
Enter first number, operator, second number: 10/ 3
Answer = 3.333333
Do another (Y/ N)? Y
Enter first number, operator, second number 12 + 100
Answer = 112
Do another (Y/ N) ? N
```

Q4. A phone number, such as (212) 767-8900, can be thought of as having three parts: the area code (212), the exchange (767) and the number (8900). Write a program that uses a structure to store these three parts of a phone number separately. Call the structure `phone`. Create two structure. Variables of type `phone`. Initialize one, and have the user input a number for the other one. Then display both numbers. The interchange might look like this:

```
Enter your area code, exchange, and number: 415 555 1212
My number is (212) 767-8900
Your number is (415) 555-1212
```

Q 5. Create two classes `DM` and `DB` which store the value of distances. `DM` stores distances in meters and centimeters and `DB` in feet and inches. Write a program that can read values for the class objects and add one object of `DM` with another object of `DB`. Use a friend function to carry out

the addition operation. The object that stores the results maybe a DM object or DB object, depending on the units in which the results are required. The display should be in the format of feet and inches or meters and centimeters depending on the object on display.

Q 6. Create a class rational which represents a numerical value by two double values- NUMERATOR & DENOMINATOR. Include the following public member Functions:

- constructor with no arguments (default).
 - constructor with two arguments.
 - void reduce() that reduces the rational number by eliminating the highest common factor between the numerator and denominator.
 - Overload + operator to add two rational number.
 - Overload >> operator to enable input through cin.
 - Overload << operator to enable output through cout.
- Write a main () to test all the functions in the class.

Q 7. Consider the following class definition

```
class father {
    protected : int age;
public;
    father (int x) {age = x;}
    virtual void iam ( )
    { cout << "I AM THE FATHER, my age is : "<< age<< endl;}
};
```

Derive the two classes son and daughter from the above class and for each, define iam () to write our similar but appropriate messages. You should also define suitable constructors for these classes. Now, write a main () that creates objects of the three classes and then calls iam () for them. Declare pointer to father. Successively, assign addresses of objects of the two derived classes to this pointer and in each case, call iam () through the pointer to demonstrate polymorphism in action.

Q 8. Write a program that creates a binary file by reading the data for the students from the terminal. The data of each student consist of roll no., name (a string of 30 or lesser no. of characters) and marks.

Q9. A hospital wants to create a database regarding its indoor patients. The information to store include

- a) Name of the patient
- b) Date of admission
- c) Disease
- d) Date of discharge

Create a structure to store the date (year, month and date as its members). Create a base class to store the above information. The member function should include functions to enter information and display a list of all the patients in the database. Create a derived class to store the age of the patients. List the information about all the to store the age of the patients. List the information about all the pediatric patients (less than twelve years in age).

Q 10. Make a class Employee with a name and salary. Make a class Manager inherit from Employee. Add an instance variable, named department, of type string. Supply a method toString that prints the manager's name, department and salary. Make a class Executive inherit from Manager. Supply a method toString that prints the string "Executive" followed by the information stored in the Manager superclass object. Supply a test program that tests these classes and methods.

Q11. Imagine a tollbooth with a class called toll Booth. The two data items are a type unsigned int to hold the total number of cars, and a type double to hold the total amount of money collected. A constructor initializes both these to 0. A member function called payingCar () increments the car total and adds 0.50 to the cash total. Another function, called nopayCar (), increments the car total but adds nothing to the cash total. Finally, a member function called displays the two totals. Include a program to test this class. This program should allow the user to push one key to count a paying car, and another to count a nonpaying car. Pushing the ESC key should cause the program to print out the total cars and total cash and then exit.

Q12. Write a function called reversit () that reverses a string (an array of char). Use a for loop that swaps the first and last characters, then the second and next to last characters and so on. The string should be passed to reversit () as an argument.

Write a program to exercise reversit (). The program should get a string from the user, call reversit (), and print out the result. Use an input method that allows embedded blanks. Test the program with Napoleon's famous phrase, "Able was I ere I saw Elba)".

Q13. Create some objects of the string class, and put them in a Deque-some at the head of the Deque and some at the tail. Display the contents of the Deque using the forEach () function and a user written display function. Then search the Deque for a particular string, using the first That () function and display any strings that match. Finally remove all the items from the Deque using the getLeft () function and display each item. Notice the order in which the items are displayed: Using getLeft (), those inserted on the left (head) of the Deque are removed in "last in first out" order while those put on the right side are removed in "first in first out" order. The opposite would be true if getRight () were used.

Q 14. Assume that a bank maintains two kinds of accounts for customers, one called as savings account and the other as current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed.

Create a class account that stores customer name, account number and type of account. From this derive the classes cur_acct and sav_acct to make them more specific to their requirements. Include necessary member functions in order to achieve the following tasks:

- a) Accept deposit from a customer and update the balance.
- b) Display the balance.
- c) Compute and deposit interest.
- d) Permit withdrawal and update the balance.
- e) Check for the minimum balance, impose penalty, necessary and update the balance.
- f) Do not use any constructors. Use member functions to initialize the class members.

Q 15. Create a base class called shape. Use this class to store two double type values that could be used to compute the area of figures. Derive two specific classes called triangle and rectangle from the base shape. Add to the base class, a member function get_data () to initialize base class data members and another member function display_area () to compute and display the area of figures. Make display_area () as a virtual function and redefine this function in the derived classes to suit their requirements.

Using these three classes, design a program that will accept dimensions of a triangle or a rectangle interactively and display the area. Remember the two values given as input will be treated as lengths of two sides in the case of rectangles and as base and height in the case of triangles and used as follows:

$$\begin{aligned} \text{Area of rectangle} &= x * y \\ \text{Area of triangle} &= \frac{1}{2} * x * y \end{aligned}$$

**Digital Electronics Lab.
ECE-212 E**

L T P
- - 3

Sessional : 50
Practical : 25

LIST OF EXPERIMENTS

1. Study of TTL gates – AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.
2. Design & realize a given function using K-maps and verify its performance.
3. To verify the operation of multiplexer & Demultiplexer.
4. To verify the operation of comparator.
5. To verify the truth tables of S-R, J-K, T & D type flip flops.
6. To verify the operation of bi-directional shift register.
7. To design & verify the operation of 3-bit synchronous counter.
8. To design and verify the operation of synchronous UP/DOWN decade counter using J K flip-flops & drive a seven-segment display using the same.
9. To design and verify the operation of asynchronous UP/DOWN decade counter using J K flip-flops & drive a seven-segment display using the same.
10. To design & realize a sequence generator for a given sequence using J-K flip-flops.
11. Study of CMOS NAND & NOR gates and interfacing between TTL and CMOS gates.
12. Design a 4-bit shift-register and verify its operation . Verify the operation of a ring counter and a Johnson counter.

Note : At least ten experiments are to be performed, atleast seven experiments should be performed from above list. Remaining three experiments may either be performed from the above list or designed & set by the concerned institution as per the scope of the syllabus.

**Microprocessors and Interfacing Lab.
ECE-218 E**

L T P
- - 3

Sessional : 25
Practical : 25

LIST OF EXPERIMENTS

1. Study of 8085 Microprocessor kit.
2. Write a program using 8085 and verify for :
 - a. addition of two 8-bit numbers.
 - b. addition of two 8-bit numbers (with carry).
3. Write a program using 8085 and verify for :
 - a. 8-bit subtraction (display borrow)
 - b. 16-bit subtraction (display borrow)
4. Write a program using 8085 for multiplication of two 8- bit numbers by repeated addition method. Check for minimum number of additions and test for typical data.
5. Write a program using 8085 for multiplication of two 8- bit numbers by bit rotation method and verify.
6. Write a program using 8085 for division of two 8- bit numbers by repeated subtraction method and test for typical data.
7. Write a program using 8085 for dividing two 8- bit numbers by bit rotation method and test for typical data.
8. Study of 8086 microprocessor kit
9. Write a program using 8086 for division of a defined double word (stored in a data segment) by another double word division and verify.
10. Write a program using 8086 for finding the square root of a given number and verify.
11. Write a program using 8086 for copying 12 bytes of data from source to destination and verify.
12. Write a program using 8086 and verify for:
 - a. Finding the largest number from an array.
 - b. Finding the smallest number from an array.
13. Write a program using 8086 for arranging an array of numbers in descending order and verify.
14. Write a program using 8086 for arranging an array of numbers in ascending order and verify.
15. Write a program for finding square of a number using look-up table and verify. .
16. Write a program to interface a two digit number using seven-segment LEDs. Use 8085/8086 microprocessor and 8255 PPI.

17. Write a program to control the operation of stepper motor using 8085/8086 microprocessor and 8255 PPI.

NOTE: At least ten experiments have to be performed in the semester out of which seven experiments should be performed from above list. Remaining three experiments may either be performed from the above list or designed & set by the concerned institution as per the scope of the syllabus of EE-309-C.