Third Semester

BTCS 301 Computer Architecture

Objectives: This course offers a good understanding of the various functional units of a computer system and prepares the student to be in a position to design a basic computer system.

1. **Register Transfer and Microoperations:** Register transfer language & operations, arithmetic microoperations, logic microoperations, shift microoperations, arithmetic logic shift unit. Design of a complete basic computer and its working. **[5]**
2. **Basic Computer Organisation and Design:** Instruction codes, Computer registers, Computer Instructions, Timing and control, Instruction Cycle, Memory reference instructions, Input/ Output and Interrupt, Design of basic Computer, Design of Accumulator Logic. **[6]**
3. **Design of Control Unit:** Control memory, design of control unit – microprogrammed, hardwired, and their comparative study. **[3]**
4. **Central Processing Unit:** General Register Organisation, Stack Organisation, Instruction formats, Addressing Modes, Data transfer and manipulations, Program control, RISC and CISC architecture. **[6]**
5. **Input-Output Organisation:** Peripheral devices, I/O Interface, asynchronous data transfer, modes of transfer, priority interrupt, DMA, I/O processor, serial communication. **[5]**
6. **Memory Organisation:** Memory hierarchy, main memory, auxiliary memory, associative memory, cache memory, virtual memory, memory management hardware. **[6]**
7. **Advanced concepts of Computer Architecture:** Concept of pipeline, Arithmetic pipeline, Instruction , vector processors and array processors. Introduction to parallel processing, Interprocessor communication & synchronization. **[5]**

Suggested Readings/ Books:

1. M. Moris Mano, **Computer System Architecture**, Pearson Education.
2. William Stallings, **Computer Organisation and Architecture**, Pearson Education.
3. David A Patterson, **Computer Architecture**, Pearson Education.
4. P. Pal Choudhri, **Computer Organisation and Design**, PHI.
5. J. P. Hayes, **Computer System Architecture**, Pearson Education.
6. Kai Hawang, **Advanced Computer Architecture**, Tata McGraw Hill.
7. Riess. **Assembly Language and Computer Architecture and using C++ and JAVA,** Cengage Learning**.**

BTAM302 Mathematics-III

Objective/s and Expected Outcome: To teach computer based Engineering Mathematics to students. After this course the student will be able to solve complex computer oriented problems.

1. **Fourier series**: Periodic Functions, Euler's Formula. Even and odd Functions, Half range expansions,

Fourier series of different waveforms. **[4]**

2. **Laplace transformations:** Laplace transforms of various standard functions, properties of Laplace transform. **[4]**
3. **Partial Differential Equations:** Formation of Partial Differential Equations, linear Partial Differential Equations, Homogeneous Partial Differential Equations with constant coefficients. **[5]**
4. **Functions of complex variables:** Limits, continuity and derivatives of the function of complex variables, Analytic function, Cauchy- Riemann equations, conjugate functions. **[5]**
5. **Linear Systems and Eigen- Values:** Gauss – elimination method, gauss- Jordan method, Gauss- Seidel iteration method, Rayleigh's Power method for Eigen values and Eigenvectors. **[4]**
6. **Differential Equations:** Solutions of Initial values problems using Eulers, modified Eulers method and Runge- kutta (upto fourth order) methods. **[4]**
7. **Probability distribution:** Binomial, Poisson and Normal distribution. **[4]**
8. **Sampling Distribution & testing of Hypothesis:** Sampling, Distribution of means and variance, Chi-Square distribution, t- distribution, F- distribution. General concepts of hypothesis, Testing a statistical Hypothesis, One and two tailed tests, critical region, Confidence interval estimation. Single and two sample tests on proportion, mean and variance. **[5]**

Suggested Readings/ Books:

1. E. Kreyszig,**" Advanced Engineering Mathematics"**, 5th Edition, Wiley Enstern 1985.
2. P. E. Danko, A. G. Popov, T. Y. A. Kaznevnikova, **"Higher Mathematics in Problems and Exercise"**, Part 2, Mir Publishers, 1983.
3. Bali, N. P., **"A Text Book on Engineering Mathematics"**, Luxmi Pub., New Delhi.
4. Peter V.O'Neil,**" Advanced Engineering Mathematics"**, Cengage Learning

---

BTCS303 Digital Circuits & Logic Design

Objective/s and Expected outcome: Demonstrate the operation of simple digital gates, identify the symbols, develop the truth table for those gates; combine simple gates into more complex circuits; change binary, hexadecimal, octal numbers to their decimal equivalent an vice versa, demonstrate the operation of a flip-flop. Design counters and clear the concept of shift resisters. Study different types of memories and their applications. Convert digital into analog and vice versa.

1. **Number Systems:** Binary, Octal, Decimal, Hexadecimal. Number base conversions, 1's, 2's, rth's complements, signed Binary numbers. Binary Arithmetic, Binary codes: Weighted BCD, Gray code, Excess 3 code, ASCII – conversion from one code to another. **[5]**
2. **Boolean Algebra:** Boolean postulates and laws – De-Morgan's Theorem, Principle of Duality, Boolean

expression – Boolean function, Minimization of Boolean expressions – Sum of Products (SOP), Product of Sums (POS), Minterm, Maxterm, Canonical forms, Conversion between canonical forms, Karnaugh map Minimization, Quine-McCluskey method - Don't care conditions. **[5]**

3. **Logic GATES:** AND, OR, NOT, NAND, NOR, Exclusive-OR and Exclusive-NOR. Implementations of Logic Functions using gates, NAND-NOR implementations. Study of logic families like RTL, DTL, DCTL, TTL, MOS, CMOS, ECL and their characteristics. **[5]**
4. **Combinational Circuits:** Design procedure – Adders, Subtractors, Serial adder/Subtractor, Parallel adder/ Subtractor Carry look ahead adder, BCD adder, Magnitude Comparator, Multiplexer/Demultiplexer, encoder/decoder, parity checker, code converters. Implementation of combinational logic using MUX.

5. **Sequential Circuits:** Flip flops SR, JK, T, D and Master slave, Excitation table, Edge triggering, Level Triggering, Realization of one flip flop using other flip flops. Asynchronous/Ripple counters, Synchronous counters, Modulo-n counter, Ring Counters. Classification of sequential circuits-Moore and Mealy, Design of Synchronous counters: state diagram, Circuit implementation. Shift registers. **[4]**
6. **Memory Devices:** Classification of memories, RAM organization, Write operation, Read operation, Memory cycle. Static RAM Cell-Bipolar, RAM cell, MOSFET RAM cell, Dynamic RAM cell. ROM organization, PROM, EPROM, EEPROM, Field Programmable Gate Arrays (FPGA). **[4]**
7. **Signal Conversions:** Analog & Digital signals. A/D and D/A conversion techniques (Weighted type, R-2R Ladder type, Counter Type, Dual Slope type, Successive Approximation type). **[5]**

Suggested Readings/ Books:

1. Morris Mano, **Digital Design**, Prentice Hall of India Pvt. Ltd

2. Donald P.Leach and Albert Paul Malvino, **Digital Principles and Applications**, 5 ed., Tata McGraw Hill Publishing Company

Limited, New Delhi, 2003.

3. R.P.Jain, Modern **Digital Electronics**, 3 ed., Tata McGraw–Hill publishing company limited, New Delhi, 2003.

4. Thomas L. Floyd, **Digital Fundamentals**, Pearson Education, Inc, New Delhi, 2003

5. Ronald J. Tocci, Neal S. Widmer, Gregory L. Moss, **Digital System -Principles and Applications**, Pearson Education.
6. Ghosal , **Digital Electronics,** Cengage Learning.

BTCS 304 Data Structures

Objectives: This course should provide the students with a fairly good concept of the fundamentals of different types of data structures and also the ways to implement them. Algorithm for solving problems like sorting, searching, insertion & deletion of data etc. related to data structures should also be discussed. After completion of this subject student should be able to choose an appropriate data structure for a particular problem.

1. **Dynamic Memory Management:** Understanding pointers, usage of pointers, arithmetic on pointers, memory allocation, memory management functions and operators, debugging pointers - dangling pointers, memory leaks, etc. **[2]**

2. **Introduction:** Concept of data type, definition and brief description of various data structures, data structures versus data types, operations on data structures, algorithm complexity, Big O notation. **[2]**
3. **Arrays:** Linear and multi-dimensional arrays and their representation, operations on arrays, sparse matrices and their storage. **[3]**
4. **Linked List:** Linear linked list, operations on linear linked list, doubly linked list, operations on doubly linked list, application of linked lists. **[4]**
5. **Stacks:** Sequential and linked representations, operations on stacks, application of stacks such as parenthesis checker, evaluation of postfix expressions, conversion from infix to postfix representation, implementing recursive functions. **[4]**
6. **Queues:** Sequential representation of queue, linear queue, circular queue, operations on linear and circular queue, linked representation of a queue and operations on it, deque, priority queue, applications of queues.

7. **Trees:** Basic terminology, sequential and linked representations of trees, traversing a binary tree using recursive and non-recursive procedures, inserting a node, deleting a node, brief introduction to threaded binary trees, AVL trees and B-trees. **[4]**
8. **Heaps:** Representing a heap in memory, operations on heaps, application of heap in implementing priority queue and heap sort algorithm. **[2]**
9. **Graphs:** Basic terminology, representation of graphs (adjacency matrix, adjacency list), traversal of a graph (breadth-first search and depth-first search), and applications of graphs. **[3]**
10. **Hashing & Hash Tables:** Comparing direct address tables with hash tables, hash functions, concept of collision and its resolution using open addressing and separate chaining, double hashing, rehashing. **[3]**
11. **Searching & Sorting:** Searching an element using linear search and binary search techniques, Sorting arrays using bubble sort, selection sort, insertion sort, quick sort, merge sort, heap sort, shell sort and radix sort, complexities of searching & sorting algorithms. **[5]**

Suggested Readings/ Books:

1. Sartaj Sahni, **Data Structures, Algorithms and Applications in C++,** Tata McGraw Hill.
2. Tenenbaum, Augenstein, & Langsam, **Data Structures using C and C++**, Prentice Hall of India.
3. R. S. Salaria, **Data Structures & Algorithms Using C++,** Khanna Book Publishing Co. (P) Ltd.
4. Seymour Lipschutz, **Data Structures**, Schaum's Outline Series, Tata McGraw Hill
5. Kruse, **Data Structures & Program Design**, Prentice Hall of India.
6. Michael T. Goodrich, Roberto Tamassia, & David Mount, **Data Structures and Algorithms in C++** (Wiley India)

7. Thomas H Cormen, Charles E Leiserson, Ronald L Rivest , and Clifford Stein, Introduction to Algorithms.
8. Ellis Horowitz, Sartaj Sahni, & Dinesh Mehta, Fundamentals of Data Structures in C++.
9. Malik , **Data Structures using C++**, Cengage Learning.

BTCS 305 Object Oriented Programming Using C++

Objectives: To understand the basic concepts of object oriented programming languages and to learn the techniques of software development in C++.

1. **Object-Oriented Programming Concepts:** Introduction, comparison between procedural programming paradigm and object-oriented programming paradigm, basic concepts of object-oriented programming — concepts of an object and a class, interface and implementation of a class, operations on objects, relationship among objects, abstraction, encapsulation, data hiding, inheritance, overloading, polymorphism, messaging. **[2]**
2. **Standard Input/Output:** Concept of streams, hierarchy of console stream classes, input/output using overloaded operators >> and << and memberv functions of i/o stream classes, formatting output, formatting using ios class functions and flags, formatting using manipulators. **[3]**
3. **Classes and Objects:** Specifying a class, creating class objects, accessing class members, access specifiers, static members, use of *const* keyword, friends of a class, empty classes, nested classes, local classes, abstract classes, container classes, bit fields and classes**.** **[4]**
4. **Pointers and Dynamic Memory Management:** Declaring and initializing pointers, accessing data through pointers, pointer arithmetic, memory allocation (static and dynamic), dynamic memory management using *new* and *delete* operators, pointer to an object, *this* pointer, pointer related problems - dangling/wild pointers, null pointer assignment, memory leak and allocation failures**.** **[5]**
5. **Constructors and Destructors:** Need for constructors and destructors, copy constructor, dynamic constructors, explicit constructors, destructors, constructors and destructors with static members, initializer lists. **[2]**
6. **Operator Overloading and Type Conversion:** Overloading operators, rules for overloading operators, overloading of various operators, type conversion - basic type to class type, class type to basic type, class type to another class type. **[4]**
7. **Inheritance:** Introduction, defining derived classes, forms of inheritance, ambiguity in multiple and multipath inheritance, virtual base class, object slicing, overriding member functions, object composition and delegation, order of execution of constructors and destructors. **[5]**

8. **Virtual functions & Polymorphism:** Concept of binding - early binding and late binding, virtual functions, pure virtual functions, abstract clasess, virtual destructors. **[3]**

9. **Exception Handling:** Review of traditional error handling, basics of exception handling, exception handling mechanism, throwing mechanism, catching mechanism, rethrowing an exception, specifying exceptions. **[2]**

10. **Templates and Generic Programming:** Template concepts, Function templates, class templates, illustrative examples. **[3]**

11. **Files:** File streams, hierarchy of file stream classes, error handling during file operations, reading/writing of files, accessing records randomly, updating files. **[3]**

Suggested Readings/ Books:

1. Lafore R., **Object Oriented Programming in C++**, Waite Group.
2. E. Balagurusamy, **Object Oriented Programming with C++**, Tata McGraw Hill.
3. R. S. Salaria, **Mastering Object-Oriented Programming with C++**, Salaria Publishing House.
4. Bjarne Stroustrup, **The C++ Programming Language**, Addison Wesley.
5. Herbert Schildt, **The Complete Reference to C++ Language**, McGraw Hill-Osborne.
6. Lippman F. B, **C++ Primer,** Addison Wesley.
7. Farrell- **Object Oriented using C++,** Cengage Learning.

BTCS306 Data Structures Lab

List of practical exercises, to be implemented using object-oriented approach in C++ Language.

1. Write a menu driven program that implements following operations (using separate functions) on a linear array:
   - Insert a new element at end as well as at a given position
   - Delete an element from a given whose value is given or whose position is given
   - To find the location of a given element
   - To display the elements of the linear array
2. Write a menu driven program that maintains a linear linked list whose elements are stored in on ascending order and implements the following operations (using separate functions):
   - Insert a new element
   - Delete an existing element
   - Search an element
   - Display all the elements

3. Write a program to demonstrate the use of stack (implemented using linear array) in converting arithmetic expression from infix notation to postfix notation.
4. Program to demonstrate the use of stack (implemented using linear linked lists) in evaluating arithmetic expression in postfix notation.
5. Program to demonstration the implementation of various operations on a linear queue represented using a linear array.
6. Program to demonstration the implementation of various operations on a circular queue represented using a linear array.
7. Program to demonstration the implementation of various operations on a queue represented using a linear linked list (linked queue).
8. Program to illustrate the implementation of different operations on a binary search tree.
9. Program to illustrate the traversal of graph using breadth-first search.
10. Program to illustrate the traversal of graph using depth-first search.
11. Program to sort an array of integers in ascending order using bubble sort.
12. Program to sort an array of integers in ascending order using selection sort.
13. Program to sort an array of integers in ascending order using insertion sort.
14. Program to sort an array of integers in ascending order using radix sort.
15. Program to sort an array of integers in ascending order using merge sort.
16. Program to sort an array of integers in ascending order using quick sort.
17. Program to sort an array of integers in ascending order using heap sort.
18. Program to sort an array of integers in ascending order using shell sort.
19. Program to demonstrate the use of linear search to search a given element in an array.
20. Program to demonstrate the use of binary search to search a given element in a sorted array in ascending order.

BTCS 307 Institutional Practical Training

BTCS 308 Digital Circuits & Logic Design Lab

Implementation all experiments with help of Bread- Board.

1. Study of Logic Gates: Truth-table verification of OR, AND, NOT, XOR, NAND and NOR gates; Realization of OR, AND, NOT and XOR functions using universal gates.
2. Half Adder / Full Adder: Realization using basic and XOR gates.

3. Half Subtractor / Full Subtractor: Realization using NAND gates.
4. 4-Bit Binary-to-Gray & Gray-to-Binary Code Converter: Realization using XOR gates.
5. 4-Bit and 8-Bit Comparator: Implementation using IC7485 magnitude comparator chips.
6. Multiplexer: Truth-table verification and realization of Half adder and Full adder using IC74153 chip.
7. Demultiplexer: Truth-table verification and realization of Half subtractor and Full subtractor using IC74139 chip.
8. Flip Flops: Truth-table verification of JK Master Slave FF, T-type and D-type FF using IC7476 chip.
9. Asynchronous Counter: Realization of 4-bit up counter and Mod-N counter using IC7490 & IC7493 chip.
10. Synchronous Counter: Realization of 4-bit up/down counter and Mod-N counter using IC74192 & IC74193 chip.
11. Shift Register: Study of shift right, SIPO, SISO, PIPO, PISO & Shift left operations using IC7495 chip.
12. DAC Operation: Study of 8-bit DAC (IC 08/0800 chip), obtain staircase waveform using IC7493 chip.
13. ADC Operations: Study of 8-bit ADC.

BTCS 309 Object Oriented Programming Using C++ Lab

1. **[Classes and Objects]** Write a program that uses a class where the member functions are defined inside a class.
2. **[Classes and Objects]** Write a program that uses a class where the member functions are defined outside a class.
3. **[Classes and Objects]** Write a program to demonstrate the use of static data members.
4. **[Classes and Objects]** Write a program to demonstrate the use of const data members.
5. **[Constructors and Destructors]** Write a program to demonstrate the use of zero argument and parameterized constructors.
6. **[Constructors and Destructors]** Write a program to demonstrate the use of dynamic constructor.
7. **[Constructors and Destructors]** Write a program to demonstrate the use of explicit constructor.
8. **[Initializer Lists]** Write a program to demonstrate the use of initializer list.
9. **[Operator Overloading]** Write a program to demonstrate the overloading of increment and decrement operators.
10. **[Operator Overloading]** Write a program to demonstrate the overloading of binary arithmetic operators.
11. **[Operator Overloading]** Write a program to demonstrate the overloading of memory management operators.
12. **[Typecasting]** Write a program to demonstrate the typecasting of basic type to class type.
13. **[Typecasting]** Write a program to demonstrate the typecasting of class type to basic type.
14. **[Typecasting]** Write a program to demonstrate the typecasting of class type to class type.
15. **[Inheritance]** Write a program to demonstrate the multilevel inheritance.

16. **[Inheritance]** Write a program to demonstrate the multiple inheritance.
17. **[Inheritance]** Write a program to demonstrate the virtual derivation of a class.
18. **[Polymorphism]** Write a program to demonstrate the runtime polymorphism.
19. **[Exception Handling]** Write a program to demonstrate the exception handling.
20. **[Templates and Generic Programming]** Write a program to demonstrate the use of function template.
21. **[Templates and Generic Programming]** Write a program to demonstrate the use of class template.
22. **[File Handling]** Write a program to copy the contents of a file to another file byte by byte. The name of the source file and destination file should be taken as command-line arguments,
23. **[File Handling]** Write a program to demonstrate the reading and writing of mixed type of data.
24. **[File Handling]** Write a program to demonstrate the reading and writing of objects.