CURRICULUM REVISION PROJECT

2012

TEACHER GUIDE FOR

(System Programming -17634)

SIXTH SEMESTER
COMPUTER SCIENCE AND ENGINEERING GROUP

Dec 2014

MAHARASHTRA STATE
BOARD OF TECHNICAL EDUCATION, Mumbai
(Autonomous) (ISO 9001:2008) (ISO/IEC 27001:2005)

INDEX

1. 0 APPROACH TO CURRICULUM DESIGN

1.1 Background:

MSBTE is introducing the revised curriculum under ‘G’ scheme from the academic year 2012-13.

There are many institutions in the state running different diploma courses. In order to ensure uniform and effective implementation of the curriculum it is necessary that every teacher is aware of approach for curriculum design, educational principles to be adopted, learning resources to be used and evaluation methods. The teacher guide prepared for each subject will provide the inputs related to above mentioned aspects to achieve uniform and effective implementation of curriculum of various subjects.

1.2 CURRICULUM PHILOSOPHY

MSBTE has adopted systems approach while designing the scientific based curriculum since 1995. The same approach has been adopted while revising the curriculum in semester pattern.

Fig. No. 1 shows the systems diagram. This diagram provides the holistic view for curriculum designing, development, implementation and evaluation

The input to polytechnic education system is the students having 10+ qualifications. The teaching learning process occurs in the institution for six/eight semesters. The output of the system i. e. Diploma pass out is normally the input to industries. (Some students do go for higher education). While designing the curriculum the expectations of the industries play a major role. Due to globalization and competition the industries expect that pass outs have generic and technological skills along with right attitude.

To fulfill the needs derived from systems approach following conceptual framework is considered:

1.3 Curriculum:

“Curriculum is an educational program designed and implemented to achieve specified educational objectives”

This definition takes into account the fact that

- Education is purposeful
- There is an organized plan of action contemplated
- Such a plan is translated into action through appropriate strategies of implementation.

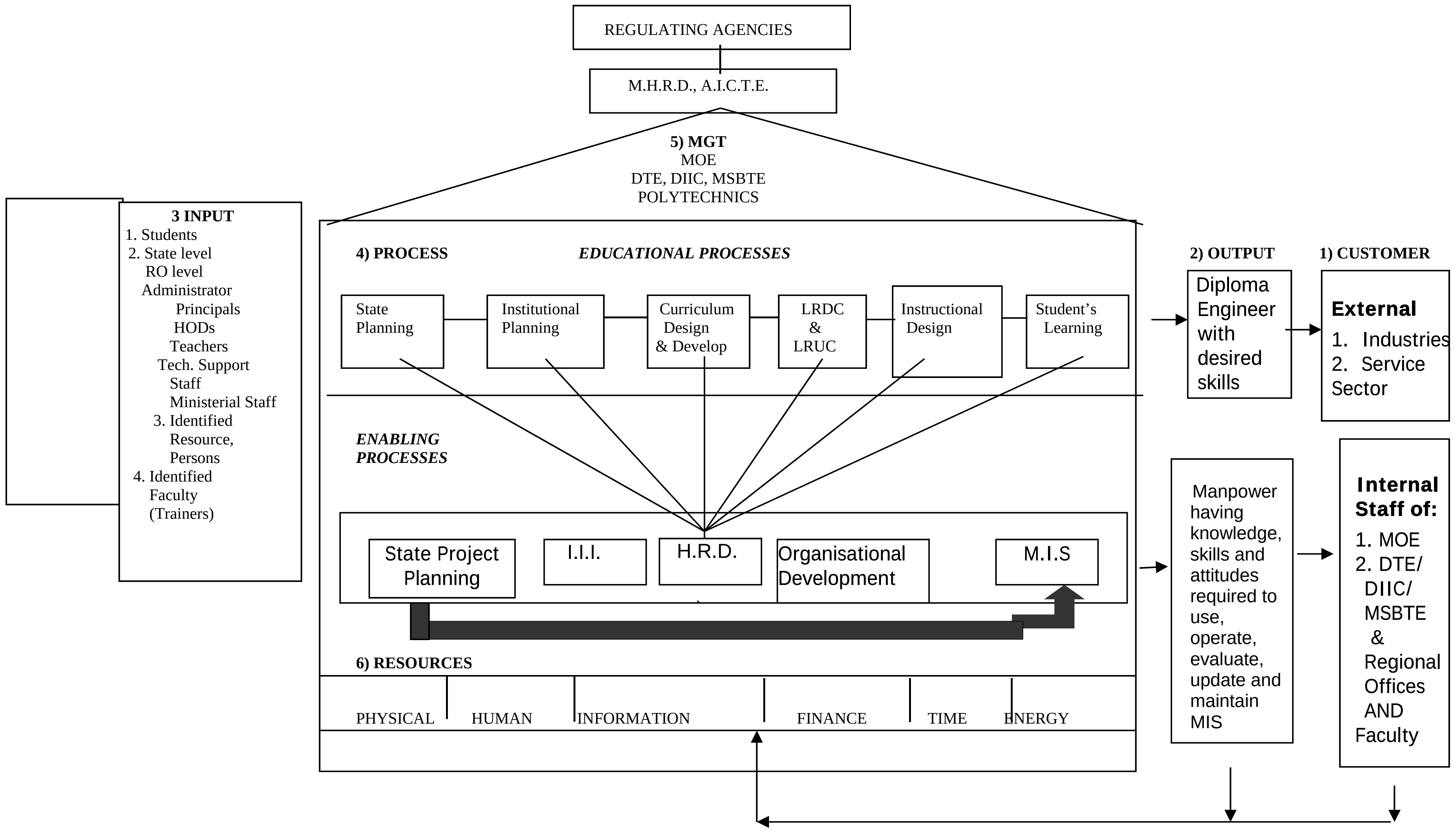


Fig 1 Systems Approach

1.4 Curriculum goals

1. To develop confidence in students by providing more exposure to industry experience and world of work at global level
2. To provide conceptual knowledge and develop analytical ability
3. To develop communication skill with good English by providing sufficient practice
4. To enhance latest technical knowledge industry interaction and media
5. To develop learning to learn skills and life skills to cope up with industrial culture
6. To impart managerial skills by providing appropriate theoretical inputs
7. To develop problem solving ability through technical projects.

1.5 DESIRED SKILLS

Industries expect from the diploma engineer the abilities and skills of general nature and specific to the job performance. The curriculum aims at developing life skills and technological skills so that the diploma pass outs would be suitable for industry. The skills are listed below:

Life Skills:

- Search information from various sources
- Develop communication ability
- Develop Presentation skill
- Work as a member of a team/group and as leader
- Collect field data
- Develop Learning to learn
- Write report for given task/work/project
- Develop computer proficiency
- Develop observation skills

Technological Skills:

Diploma engineers should possess following intellectual and motor skills in order to satisfactorily perform duties assigned to them:

A) Intellectual skills.

1. Identify the problem
2. Prepare the algorithms
3. Analyze the problem
4. Prepare the flowchart/model
5. Select hardware and software tools and technologies
6. Use of appropriate programming languages
7. Write programs
8. Test and debug computer Program
9. Diagnose the hardware faults
10. Prepare and interpret software documentation

B) Motor Skills.

1. Handle the Computer system
2. Handling trouble shooting tools
3. Assemble and disassemble computer system
4. Install hardware devices
5. Install network

1.6 Salient Changes in the curriculum:

- ❖ For First Semester Basic Science is divided into two parts- Basic Physics and Basic Chemistry. Theory examination of both parts as well as practical examination of both parts will be conducted on separate days. Sum of theory marks of both parts shall be considered for passing theory examination of Basic Science. Similarly it is applicable to practical examination. It is mandatory to appear for theory and practical examination of both parts. Candidate remaining absent in any examination of any section will not be declared successful for that exam head.
- ❖ For second semester Applied Science is divided into two sections- Applied Physics and Applied Chemistry where the theory examination of 50 marks each and practical examination of 25 Marks each will be conducted separately and the minimum passing marks for Applied Science will be the combination of both the sections. . It is mandatory

to appear for theory and practical examination of both parts. Candidate remaining absent in any examination of any section will not be declared successful for that exam head.

- ❖ The components of Development of Life Skills were taught in two semesters. In Development of Life Skills –I the topics related to personal development, such as Learning to Learn Skills, personality development, presentation skills etc. were included. In Development of Life Skills – II the topics related to Team Building, Leadership, group behavior etc. were covered. In the revised curriculum the scope of development of life skills has been broaden to include behavioral science component. Therefore the subject Development of Life Skills – II has been renamed and it is now included at Vth Semester in the revised curriculum under the title Behavioral Science.
- ❖ The subject of Professional Practices was introduced to integrate the skills acquired in Development of Life Skills, through technical subjects from second to sixth semester. The experience in implementing the contents of the subject shows that there are limited activities possible in second semester as the technical knowledge given to the students is very limited. Also at sixth semester the student are doing projects in which they are performing many activities included in the Professional Practices and therefore it is proposed that the subject of Professional Practices be prescribed only for three semesters vis. Third, fourth and fifth semesters.
- ❖ Introduction of Environment Studies at fourth Semester for all courses
- ❖ From the experience of implementation of Elective Subjects at V and VI semesters in last five years, it is proposed to have only one elective at the sixth semester for all courses. However the specialized courses like Medical Electronics, Electronics and Video Engineering will not have provision for electives. For elective, student will have to choose one from the given two/three subjects.
- ❖ While revising the curriculum redundant /obsolete topics/sub topics are being replaced by new/advance technology topics/sub topics.
- ❖ In Computer Engineering Group, for fourth Semester IF Computer Networks (CON) is replaced with Data Communication and Networking.
- ❖ For Fourth Semester IF, Applied Multimedia Technology Theory subject is changed to Practical.
- ❖ For Fifth semester CO, System Programming subject is included. For IF course, Information Security subject is included.

- ❖ In Sixth Semester , elective subjects have been included.
- ❖ In order to satisfy the course objectives online examination has been introduced for the subjects Management (for all branches) and advanced Java Programming for Computer Group .
- ❖ Linux Programming has been included as a practical subject for CO /CM branch and Scripting Technology has been included as a practical subject for IF.
- ❖ Mobile Computing has been introduced for IF branch.

2. 0 OBJECTIVES

2.1 Introduction

Objectives are the statements which describe the expected learning outcome. Such statements enable teachers to plan instructional process with appropriate resources. These objectives also provide a direction to frame proper questions to assess the learning outcome. During last decade there has been research on cognitive approach in psychology. This approach is based on biological structure of brain and meta-cognitive knowledge dimension. Important elements of this approach which form basics of learning are explained below.

2.2 Domains of Learning:

Learning is a process by which students develop relatively permanent change in mental associations through experience. This is how learning is defined by cognitive psychologists. Behavioral; psychologists define learning as a relatively permanent change in behavior.

There are following domains of learning:

A: Cognitive Domain relates to intellectual skills or abilities

B: Affective Domain relates to emotions, feelings, likes, dislikes etc.

C: Psychomotor Domain relates to manipulative skills of hands, legs. Eye-hand coordination in Engineering & Technology courses, endeavor is made to design curriculum with a focus on development of cognitive skills through classroom teaching. Whereas manipulative (psychomotor) skills are developed in workshops, laboratories & seminars where students work individually or in a group. Development of affective skills attitudes and value is supposed to be acquired through projects and co-curricular activities. These are also developed from the work culture or institutions.

How far a student has developed these abilities/skills especially from cognitive and psychomotor domains is assessed on the basis of suitable examinations. When classroom and laboratory teaching is viewed in this light, evaluation becomes an integral part of teaching – learning process.

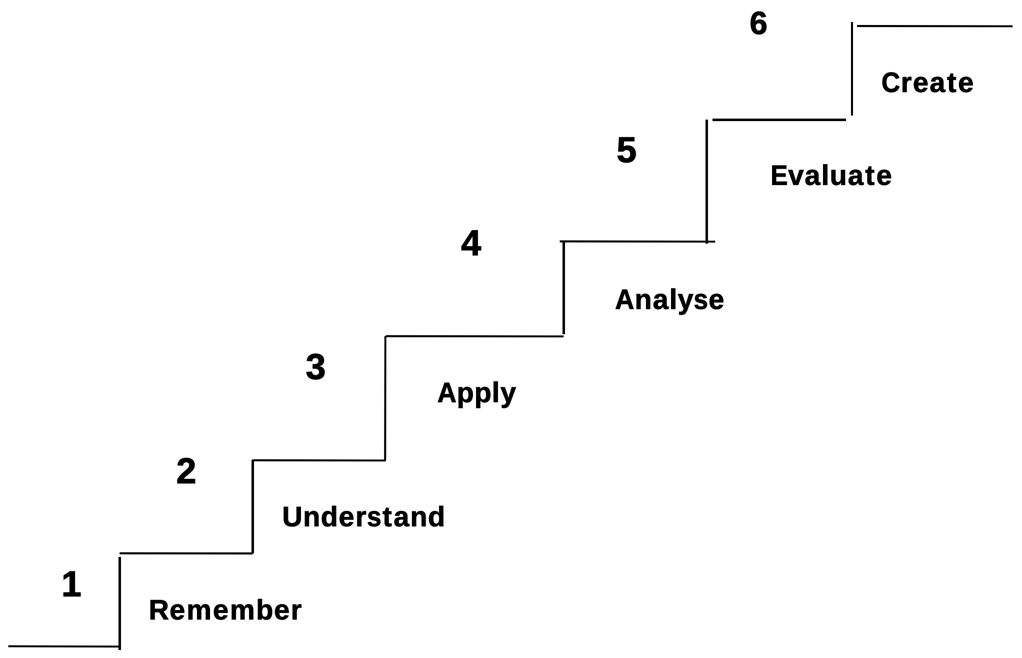
2.3 LEVELS OF LEARNING:

Question paper is a tool/ instrument designed to test the extent of learning of the student. Various questions set in a question paper should assess the abilities of students to respond to level of learning. Dr. Bloom a German educationist classified levels of learning in cognitive domain for the purpose of writing objectives and assessment. Dr. Bloom's revised taxonomy is based on cognitive psychology and is two dimensional. First dimension is cognitive process dimension ad other is knowledge dimension. Details of these two dimensions are given below.

2.4.1 Cognitive Domain:

Dr. Benjamin Bloom (1956) analysed questions asked in various examinations in American situation and proposed a hierarchical arrangement of instructional objectives (Intellectual abilities) tested by these questions.

The lowest level of cognitive learning achieved by a student is demonstrated by the recall of information that the student retrieves from his long term memory. So, the storage and retrieval of specific facts, concepts, principles, laws, definitions, properties, procedures etc. directly from memory was classified as a knowledge level objective. Thus questions testing memory of students were treated as at the lowest level of the hierarchy of intellectual abilities. The other levels of hierarchy proposed by Dr. Bloom in 1956 relate to the degree of information processing required in the brain needed to provide answer to a question. The various levels in the cognitive hierarchy proposed by Dr. Bloom in 1956 and further revised in 2001 are given below in the diagrammatic form.



Following are the details of each level which indicate the general and specific objectives. Further appropriate verbs are given which are useful in setting good questions. In this table only four levels are considered for diploma students.

Description of the Major Levels in the cognitive Domain (Bloom's Taxonomy)	Illustrative General Instructional Objectives	Illustrative verbs for stating specific learning outcomes
Remember – Knowledge is defined as the remembering of previously learned material. This may involve the recall of a wide range of material, from specific facts to complete theories, but all that is required to mind of the appropriate information. This represents the lowest level of learning outcomes in the cognitive domain	Knows common terms, specific facts, basic concepts, principles, methods & procedures	Define, describe, identify label, list, match, name, outline, reproduce, select, state
Understand – This is defined as the ability to grasp the meaning of material. This may be shown by translating material from one form to another (words or numbers) by interpreting material (explaining or summarizing), and by estimating future trends (predicting consequences or effects). Draw sketches these	Understands fact, principles Interprets verbal material, Interprets charts, tables, graphs. Translates verbal material to mathematical	Convert, distinguish estimate, explain, extend, generalize, give examples; infer, paraphrase, predict, rewrite, summarize, draw labeled sketches.

learning outcomes go one step beyond the simple remembering of material and represent the lowest level of understanding.	formula. Estimates consequences implied in data. Justifies methods & procedures.	
Apply – Application refers to the ability to use learned material in new and concrete situations. This may include the application of such things as concepts, principles, rules, methods, laws and theories. Learning outcomes in this area require a higher level of understanding than those under the level described earlier.	Applies principles to new situations. Applies theories to practical situations. Solves mathematical problem. Construct charts, graphs Demonstrates correct usage of a procedure	Change, compile, demonstrate, discover manipulate, modify operate, predict, prepare, produce, show, solve, use.
Analyze – Analysis refers to the ability to break down material into its component parts so that its organizational structure may be understood. This may include the identification of the parts, analysis of the relationship between parts, and recognition of the organizational principles involved. Learning outcomes here represent a higher intellectual level than "understand" and apply because they require an understanding of both the content and the structural form of the material.	Recognizes unstated assumptions and logical fallacies in reasoning. Distinguishes between facts and inferences. Evaluates relevance/ adequacy of data.	Breakdown, diagram, differentiate, discriminate, distinguish, identify illustrate, infer, outline, point out, relate, select, separate, subdivide.

2.4.2 Categories of Knowledge Dimension

After considering the various designations of knowledge types, especially developments in cognitive psychology that have taken place since the original framework of Bloom's taxonomy, knowledge is categorised in 4 types – Factual , Conceptual, Procedural and Meta-cognitive.

Factual Knowledge (A) is knowledge of discrete, isolated content elements. It includes knowledge of terminology and knowledge of specific details and elements. In contrast,

Conceptual Knowledge (B) is knowledge of "more complex, organised knowledge form". It includes knowledge of classifications and categories, principles and generalizations and theories, models and structures.

Procedural Knowledge (C) is "knowledge of how to do something". It includes knowledge of skills and algorithms, techniques and methods, as well as knowledge of criteria used to determine and/or justify "when to do what" within specific fields and disciplines.

Meta-cognitive knowledge (D) is “knowledge about cognition in general as well as awareness of and knowledge about one’s own cognition. It encompasses strategic knowledge, knowledge about cognitive tasks, including contextual and conditional knowledge; and self-knowledge”.
Assessment is required to be done on the basis of categories of knowledge and levels of learning. Table below indicates the two dimensional grid based on Blooms Taxonomy for setting questions.

Knowledge Dimension	COGNITIVE PROCESS DIMENSION			
	1 Remember	2 Understand	3 Apply	4 Analyze
A. Factual Knowledge				
B. Conceptual Knowledge				
C. Procedural Knowledge				
D. Meta-cognitive Knowledge				

2.5 Components of Curriculum:

2.5.1 Rationale: It indicates the logical basis for the inclusion of the subject in the curriculum It also indicates the importance of the subject related to entire curriculum.

Rationale tells the students the connection of subjects related to study of higher level subjects and also the use in their job/profession.

2.5.2 Objectives: Objectives indicate what the student will be able to do/perform after he/she completes the study of the subject. It also in other words indicates the scope of the subject.

Objectives indicate what is achievable and hence give direction to the student about how to study the subject, what important things are to be observed and performed during practicals.

Just as rationale indicates the use of the knowledge gained while studying the subject, objectives indicate how efficiently and effectively one can work if the objectives are fulfilled while studying the subject.

2.5.3 Learning Structure: It graphically/pictorially indicates the content of the curriculum of the subject and what is to be learnt in the subject. As you know that Cognitive Domain knowledge is divided in four components as mentioned in the Two dimensional grid. Of this Factual, Conceptual and Procedural knowledge components are identified in the curriculum of the subject along with the applications.
Facts, Concepts, Principles are used in developing procedures and applications. So these are given sequentially below procedure as Principles, Concepts and Facts in their order. Learning structure also provide an idea about how to develop the subject logically to achieve the objectives.

2.5.4 Contents: List of topics and subtopics to be included in the curriculum of the subject is given in the contents. This helps in achieving the rationale and objectives identified. Contents indicate the importance of the topics, sub topics in development of the subject and accordingly weightages in terms of Hours required to teach the subject components, so that the desired learning takes place. Marks to be allotted while testing the knowledge gained by the student are also indicated.

2.5.5 Practicals: While designing the curriculum the objectives are identified. To achieve these objectives students have to develop certain intellectual and motor skills. These skills are developed through well designed Practicals. So in the curriculum the list of the skills to be developed through Practicals is given. The list of Practicals is so developed that after performing the Practicals identified skills will be developed. Here it is necessary that the teacher gives enough opportunity to all the students to perform the practical properly to develop the skills in each one of them.

The skills will be developed if the students actually perform certain activities or tasks. Therefore it is necessary that any practical included in the curriculum necessarily involve some activities to be done by the students. So one has to think and innovate to modify the study experiments so that students will be asked to perform some activity. It could be in terms of identifying components, listing of materials used for manufacturing the components, stating importance of use of certain materials etc.

So any curriculum of a subject is so designed that it achieves the objectives of that subject as well as fulfill the objectives of the entire curriculum

3.0 CONTENT ANALYSIS

3.1 Components of Content Analysis:

As we have discussed earlier, any curriculum or syllabus of a SUBJECT given to the teacher is organised in terms of UNITS which include TOPICS or SUB-TOPICS as the case may be indicating the TIME in which it is expected to be taught to the students. Components of a topic or part thereof are analysed here at a micro level.

Before we begin actual teaching of any topic (lesson), we must carefully and critically analyse it so that we can plan for teaching - select appropriate media, methods and techniques of teaching and arrange the suitable resources to be required. This analysis of the content of a Topic results in identification of the following components of the content:

1. Facts
2. Concepts
3. Principles (rules, laws, theories)
4. Applications
5. Procedures
6. Skills (Psychomotor Skills), and
7. Attitudes (underlying affective behaviors as quite often these are not specifically mentioned in the curriculum, still they are to be developed lesson after lesson gradually).

When we undertake the exercise of content analysis, we ourselves understand the subject fully well and at the same time we become clear as to what we are going to teach. It also gives us an idea as to which methods of teaching and media of instruction we should prepare and use and also what resources including time we will require. This analysis will also enable us to design assignments as well as how we are going to assess students learning.

Since the nature of the components of content (1 to 7) differs from one another. These are learned by the students differently as different mental processes are involved in learning these components. The immediate implication of this varying nature of components is that these need to be taught differently and assessed differently. For example, if you look at components I to 5 all of which belong to Cognitive Domain of Learning; Component 6 belongs to Psychomotor Domain and Component 7 belongs to Affective Domain (cannot be taught as these attitudes are caught), you will find that these differ from one another. The classification of human behaviors (activities) into the above three domains of learning entails the use of entirely different methods and media of instruction. Different locations of learning (classroom, laboratories, workshops, field visits) need to be selected.

Now we will discuss these components in some detail and see how each one of these should be taught and assessed differently.

3.1.1 FACTS:

These are universally accepted and commonly understood items about which there cannot be much argument and discussion. These are required only to be informed. For example: The sun rises in east and sets in the west; names of scientists and the year in which their theories were propounded; the rules and regulations of admission and examination prescribed by the University are some of the examples of facts. Sometimes, they need not be emphasised in the class as the students already know them. But information can be passed on by word of mouth, if deemed necessary.

3.1.2 CONCEPTS:

A concept is an abstraction or an idea that permits the learner to classify a variety of related phenomena into a convenient and meaningful category. Concept of something is like a picture formation of that thing which helps in conceptualizing it. Gagne says that concept learning produces a certain fundamental change in human performance that is independent of subject or content. Concepts can be divided into the following two categories:

1. **Concrete Concepts:** those which can be seen, touched and manipulated e.g. house, book, table, chair, cat, dog, any machine or apparatus, overhead projector, chalkboard and duster.

2. **Abstract Concepts:** those which cannot be seen and touched and handled but can only be imagined e.g. force, work, fractions, decimal, bending moment, moment of

inertia, friction, heat, and induction. Teaching of concrete concepts is not that difficult because the teacher can show the object physically or its picture. On the contrary, teaching of an abstract concept offers difficulty to the teacher as well as for students to understand. These concepts can be learned by heart without understanding as children mug up Nursery Rhymes without understanding even a single word. But at the stage of higher tearing, this type of rote learning is not desirable. Adolescents (teenagers) and adults do not accept things without understanding.

3.1.3 Concept Attributes:

We identify a concept and understand it, once we are told about its qualities characteristics, and features. They are technically called concept attributes. While teaching a concept to our students we must spell out as many attributes as possible for better understanding of the concept.

Example: The Concept **of Friction**

Attributes:

1. Friction is a resistive force.
2. Frictional force acts in the direction opposite to the direction of the applied force.
3. Frictional force is more when the surfaces in contact are rough.
4. Smooth surfaces (perfect) have zero friction.
5. Frictional force is self-adjusting to a limit.

Towards the end of this Theme Paper a number of examples of concept attributes are given for your guidance.

The following questions pertaining to a concept (object or process) will be helpful in writing concept attributes:

1. What it is.
2. What are its constituent parts.
3. How it works.
4. How it is similar to and different from other known concepts.
5. What are its uses?

3.1.4 PRINCIPLES:

A principle is a statement of relationship between two or more concepts. Principles are sometimes called rules, laws or generalizations. In others words, relationship between two or more concepts which is scientific and universally true is called a Principle.

For Example: (related concepts are underlined)

1. Actions and reactions are equal and opposite.
2. Ohm's law I = V/R is a principle, where I (Current), V (Voltage), and R (Resistance) are the concepts. While teaching a principle we must recall the concepts which it involves. These concepts might have been taught in the previous lesson. As you already know, concept learning is a prerequisite to Principle learning. Thus we recall the concepts of current, voltage and resistance by asking questions to the students. Only after that we must tell the relationship among these i.e. Ohm's Law.

3.1.5 APPLICATIONS:

Whatever principles, laws and theories have been learned are only academic exercises unless these are applied to solve a practical problem. In other words, we call this application transfer of learning to a new situation. If you recall, the process of learning dealt with in Theme Paper 2, you will appreciate that the litmus test of learning having occurred is its application in a new situation or solving a new problem.

For example:

1. Ohm's law can be applied to find out the unknown quantity (voltage, current, and resistance).

2. Design of a structure can be made based on related principles and theories.

3. Principles of learning and events of instruction can be applied in ‘Designing a lesson Plan' and 'Presenting the lesson in the classroom".

4, The above principles can also be applied while preparing textbooks, workbooks, learning packages and laboratory manuals to be used by the students.

3.1.6 PROCEDURES:

While analysing the content of a topic you might come across certain standard procedures which are prescribed to perform an operation or a given task. These procedures should be clearly identified and taught accordingly not to be left to chance. We should not pre-suppose that the students understand them. We cannot afford to take these things for granted.

For Example:

1. Procedure of setting up of an apparatus.
2. Procedure to start an engine.
3. Procedure to operate a machine (a lathe).

3.1.7 SKILLS (PSYCHOMOTOR):

A skill is an ability to perform a task expertly and well. The skilled performance; must meet a pre-specified standard of acceptable performance. A skill has the following three characteristics:

1. It represents a chain of motor responses;
2. It involves the co-ordination of hand and eye movements, and
3. It requires the organization of chains into complex response patterns.

Skills could be intellectual (thinking, understanding); interactive (communication skills) and social (socialising, mixing up with others) also. But normally when we use the word skills, it refers to psychomotor skills.

For Example:

1. Welding a butt joint,
2. Setting a theodolite at a station,
3. Making proper circuit connections, and
4. Turning a job on a lathe machine.

Laboratories and workshops of Polytechnics are the locations where these skills are developed among the students under the guidance of expert instructors *of* operators. Drill and practice are the main methods of teaching and learning these skills through model demonstrations and careful observations thereof.

Alongside developing these skills, desirable attitudes like cooperation, team work, leadership, safety, cost consciousness are also developed.

3.2 TEACHING OF CONCEPTS;

In order to teach concepts effectively the following steps have been suggested by De Cecco & Crawford (1974).

Steps Suggested:

1. Describe the performance expected of the student after he has learned the concept.
2. Reduce the number of attributes to be learned in complex concepts and make important attributes dominant.
3, Provide the student with verbal indicators (explanation).
4. Provide positive and negative examples (non-examples) of the concept.
5. Present the examples in close succession or simultaneously.
6. Provide occasions for student responses and the reinforcement of these responses, and
7. Assess the learning of the concept.

3.3 TEACHING OF PRINCIPLES:

De Cecco & Crawford (1974) has suggested the following steps for teaching principles effectively.

Steps:

1. Describe the performance expected of the student after he has learned the principle.
2. Decide and indicate which concepts or principles the students must recall in learning the new principle.
3. Assist the student in the recall of component concepts.
4. Help the student in the recall of component concepts.
5. Help the student to combine the concepts and put them in a proper order.
6. Provide for practice of the principle and for reinforcement of student responses.
7. Assess the learning of the principle.

3.4 CONCLUSION:

To sum up, it can be said that. it is essential for the teachers to develop the skills of 'Content Analysis' of their subjects. It brings content clarity amongst the teachers themselves. More importantly, Content Analysis will be a pre-requisite for writing Instructional Objectives of the topic to be taught. Teaching and learning process is bound to be effective once these crucial academic activities are undertaken.

4.0 CURRICULUM

Course Name: Computer Science and Engineering
Course Code : CW
Semester : Sixth
Subject Title : System Programming
Subject Code : 17634

Teaching and Examination Scheme:

Teaching Scheme			**Examination Scheme**					
TH	TU	PR	PAPER HRS	TH	PR	OR	TW	TOTAL
03	--	02	03	100	50#	--	25@	175

NOTE:

- **Two tests each of 25 marks to be conducted as per the schedule given by MSBTE.**
- **Total of tests marks for all theory subjects are to be converted out of 50 and to be entered in mark sheet under the head Sessional Work (SW).**

Rationale:

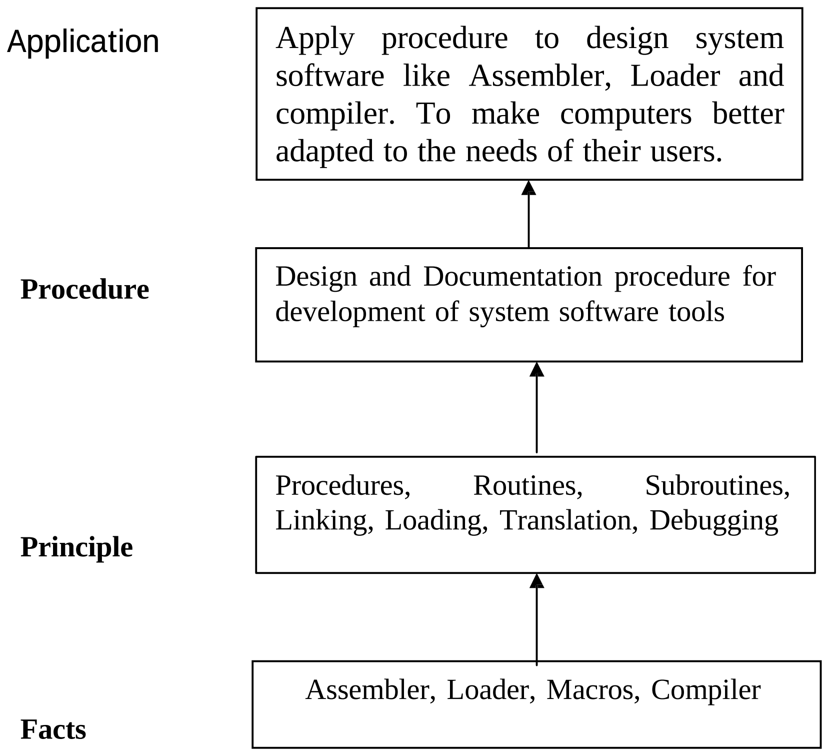
A modern computer has powerful capabilities such as fast CPU, large memory, Input- Output devices and networking support. However, It has to be instructed through the machine language. A common user does not wish to interact with the computer at this level. The System programs are the collection of programs that bridge the gap between the users and the operating system. The main aim of System programming is to understand designing and implementation of software's like assemblers, loaders and compilers. Using system programming students will have an idea about how the system tools coordinates with operating system.

General objectives:
Students will be able to:

- Understand the concept of machine structure, machine language and assembly language.
- Define symbols, literals, instructions, and assign addresses.
- Understand the concept of lexical, syntax and semantic analysis.
- Differentiate between procedures and subroutines.

- Understand macros, macro call and expansion.
- Understand the concepts of Memory allocation, loading and linking.
- Understand design of compiler, loader, linker and assembler.

Learning Structure:


Application
Apply procedure to design system software like Assembler, Loader and compiler. To make computers better adapted to the needs of their users.
Procedure
Design and Documentation procedure for development of system software tools
Principle
Procedures, Routines, Subroutines, Linking, Loading, Translation, Debugging
Facts
Assembler, Loader, Macros, Compiler

Theory

Topic No	Contents	Hours	Marks
1	**INTRODUCTION TO SYSTEM PROGRAMMING** **Objectives:** ➢ Recognize the need of system programming. ➢ Understand the role of language processors. **1.1** What is System Software? Goals of System Software. **1.2** Components of System Software : Assemblers, Loader, compiler, Macro processor **1.3** Evolution of System Software and Operating System **1.4** Foundations of system Programming, Machine Structure	04	10
2	**ASSEMBLER** **Objectives:** ➢ Introduce Single pass and Two-Pass assembler ➢ Understand the general assembly scheme **2.1** General design procedure **2.2** Design of the assembler :Statement of the problem; Data Structure; Format of databases; Algorithm; Look for modularity **2.3** Table Processing: Searching and Sorting- Linear Search; Binary Search Sorting: Interchange sort; Shell sort; Bucket sort; Radix exchange sort; Address calculation sort; Comparisons of sort; Hash or Random entry searching	10	20
3	**MACRO LANGUAGE AND MACRO PROCESSORS** **Objectives:** ➢ Comprehend the definition and expansion of macros instructions ➢ Gain insight into design of macro preprocessor **3.1** Macro Instructions **3.2** Features of a Macro facility - Macro Instruction Arguments; Conditional macro expansion; Macro call within Macros; Macro Instruction defining Macros **3.3** Implementation - Implementation of restricted faculty: Two Pass Algorithm, A Single Pass Algorithm, Implementation of macro calls within Macros, Implementation within an assembler	10	20

4	**LOADERS AND LINKING** **Objectives:** ➢ Understand the concepts and requirements of loading and linking ➢ Gain insight into the design of linker **4.1** Loaders Schemes : “Compile and go” loaders; General Loader Schemes; Absolute Loaders; Subroutine linkages; Relocating loaders; Direct linking loaders; Other loaders scheme: Binders, Linking loaders Overlays, Dynamic Binders **4.2** Design of Absolute loaders **4.3** Design of Direct Linking Loaders: Specification Problem; Specification of data structures; Format of database; Algorithm	10	20

5	**COMPILER** **Objectives:** ➢ Understand the aspects of compilation of high-level languages. ➢ Describe the various phases of compilers. ➢ Discuss about memory allocation scheme used in compilers. **5.1** Statement of a problem: Recognizing basic elements; Recognizing Syntactic units and Interpreting meaning; Intermediate from: Arithmetic statements, Non-Arithmetic statement, Non-executable statements; Storage Allocation; Code Generation: Optimization(M/c independent), Optimization (M/c dependent); Assembly Phase; General Model of Compiler. **5.2** Phases of Compiler : Lexical Phase: Tasks, Databases, Algorithm; Syntax Phase: Databases, Algorithm; Interpretation Phase: Databases, Algorithm; Optimization: Databases, Algorithm; Storage Assignment: Databases, Algorithm; Code Generation: Databases, Algorithm; Assembly Phase: Databases, Algorithm; Passes of a Compiler	12	24
6	**PARSING** **Objectives:** ➢ Identify and understand the role of a lexical and syntax analyzer. ➢ Understand the top-down and bottom-up parsing techniques. **6.1** Top down parser **6.2** Bottom up parser	02	06
	Total	**48**	**100**

Skills to be developed: Intellectual Skills:

- Use of programming language constructs in program implementation.
- To be able to apply different logics to solve given problem.
- To be able to write program using different implementations for the same problem

- Study different types of errors as syntax semantic, fatal, linker & logical
- Debugging of programs
- Understanding different steps to develop program such as
    - Problem definition
    - Analysis
    - Design of logic
    - Coding
    - Testing
    - Maintenance (Modifications, error corrections, making changes etc.)

Motor Skills:

Proper handling of Computer System.

Practicals:

List of Practical:

Sr. No.	Title of the Experiment
1.	Understand and Develop a C-Program to implement interchange sort on 8 numbers.
2.	Understand and develop implement a C-program to implement a bucket sort for three digit numbers.
3.	Understand and Develop a C-Program to implement a Radix interchange sort for binary numbers.
4.	Understand and develop a C-program for address calculation sort. algorithm.
5.	Understand a program for generating a symbol table in C or using LEX/YACC.
6.	Develop different kinds of macro subroutine in an assembly language..
7.	Understand different Loader Schemes.
8.	Develop a program to read tokens and print its type using LEX/YACC.
9.	Develop a program for code generation using LEX/YACC.
10.	Develop a program for identifying loop variant using LEX/YACC.
11.	Develop a LEX program to parse input to check that if belongs to given syntax of language.

NOTE: All Practical to be performed on Linux OS using gcc, Lex and Yacc

Learning Resources

1.Books

Sr. No.	**Author**	**Title**	**Publisher**
1	John J. Donovan	System Programming	Tata McGraw-Hill Edition
2	D.M. Dhamdhere	System Programming and Operating System	Tata McGraw-Hill Edition
3	G.Sudha Sadashiv	Compiler Design	SciTech

4	Rajesh K. Maurya	System Programming	Dreamtech

2. Websites:

en.wikipedia.org/wiki/System_programming
in1.csie.ncu.edu.tw/~chia/Course/Assembly/macro.

5.0 IMPLEMENTATION STRATEGY:

5.1 Planning of Lectures for a Semester with Content Detailing:

Teacher shall implement the methodology/ techniques mentioned in the following table while teaching the topics. Along with this teacher may use additional/alternative methods to make students learning more meaningful.

Topic 1: Introduction **to System Programming** **10M**

Specific Objective (As given in the Curriculum)

1. **Recognize the need of system programming.**
2. **Understand the role of language processor.**

Additional Specific Objective

- **List the system hardware and software components.**
- **State the need of assembler, loader, compiler, macro assembler, and operating system.**

Knowledge Category	**Example/s of category**	**Teaching methodology**
FACT	System, Software. Machine Structure Operating System	PPT's and Video's to show the hardware and software components.
CONCEPT	Assembler. Loader Compiler Macro Processor	The teacher should be able to demonstrate the actual processing of source code in any HLL to go through the process of assembler , loader, compiler to get the final output
PRINCIPLE	Working with Turbo C compiler	To demonstrate the execution of a simple C program using F7 Step mode by adding a watch for a variable.
PROCEDURE	LINUX OS	To install LINUX OS, and to get familiar with gcc environment.
APPLICATION	Hello World program	To implement the program in gcc.

Learning Resources:

Books:

Title: System Programming by J. Donovan.

1) System Programming and Operating System by D.M. Dhamdhere

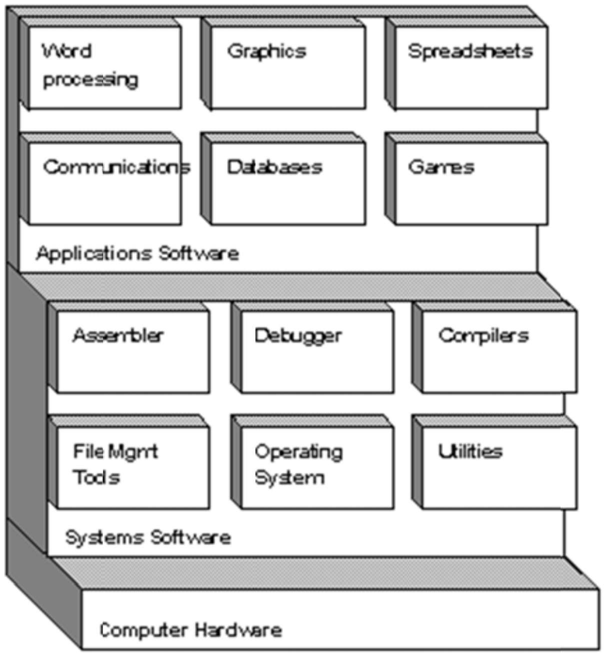
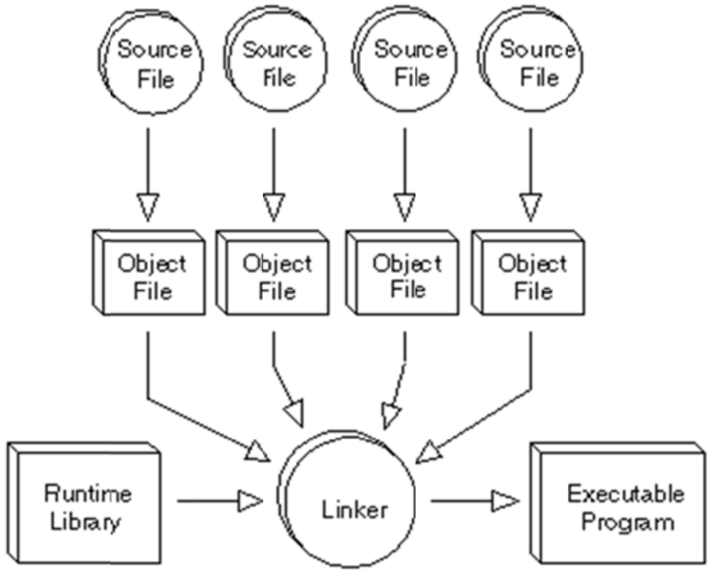
Teaching Aids:

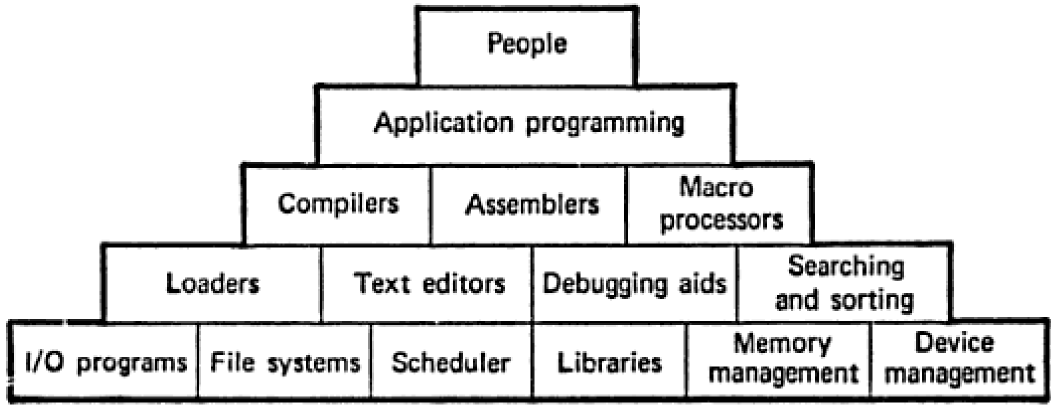
Video Lectures:

1. www.youtube.com/watch?v=GNu5YMEhIdg

Websites :

1. www.linux.com › Learn Linux › Linux New User Guides

Lecture No.	Topic/ Subtopic to be covered
1	What is system software? Goals of system software. :explanation of the following diagram with respect to its components is expected. Goals of system software can be explained by taking the real life examples.  Word processing Graphics Spreadsheets Communications Databases Games Applications Software Assembler Debugger Compilers File Mgmt Tools Operating System Utilities Systems Software Computer Hardware
2	Components of System software: teacher shall explain the components like assemblers, compilers ,loaders, linkers , macros and formal system by explaining their significance. Book : System programming by Donovan Page no.: 4,5,6
3	Evolution of System Software.  Source File Source File Source File Source File Object File Object File Object File Object File Runtime Library Linker Executable Program

	System programming by Donovan Page no.: 4,5,6
4	Foundation of system Programming. System programming by Donovan Page no.: 2 Teacher shall explain the foundation of system programming with the help of the diagram : 2 MACHINE STRUCTUR  People Application programming Compilers Assemblers Macro processors Loaders Text editors Debugging aids Searching and sorting I/O programs File systems Scheduler Libraries Memory management Device management **FIGURE 1.1** Foundations of systems programming

Topic 2: **Assembler** **20M**

Specific Objective (As given in the Curriculum)

1. Introduce single pass and two pass assembler.
2. Understand the general assembly scheme.

Additional Specific Objectives

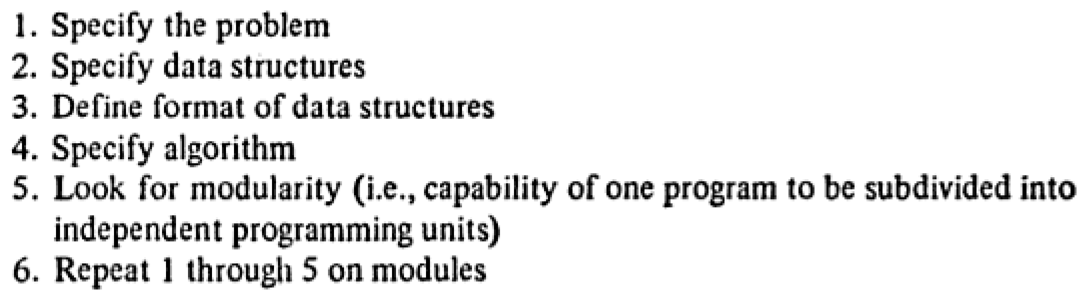
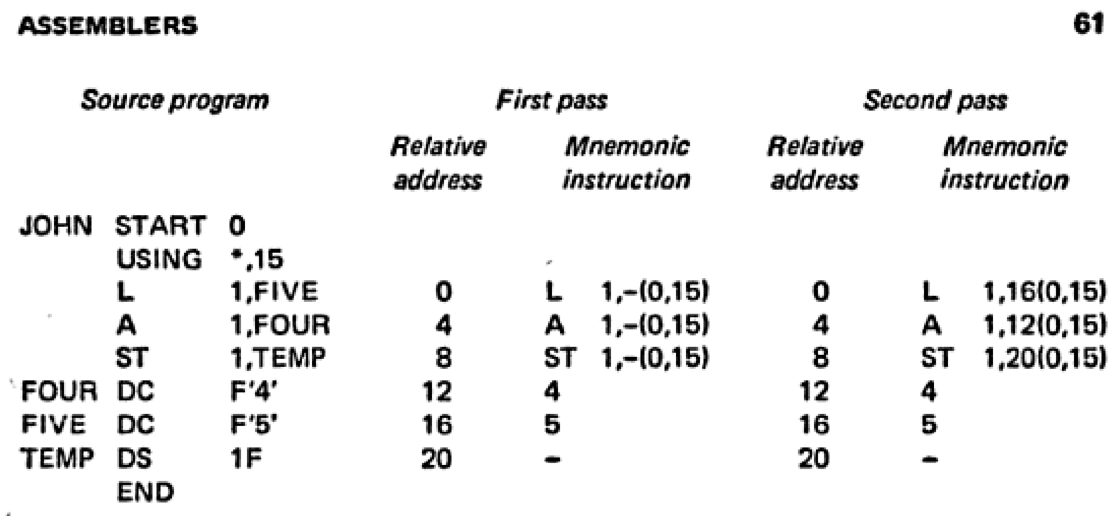
- ➢ State the different searching and sorting algorithms.
- ➢ Describe the procedure for searching and sorting algorithms.

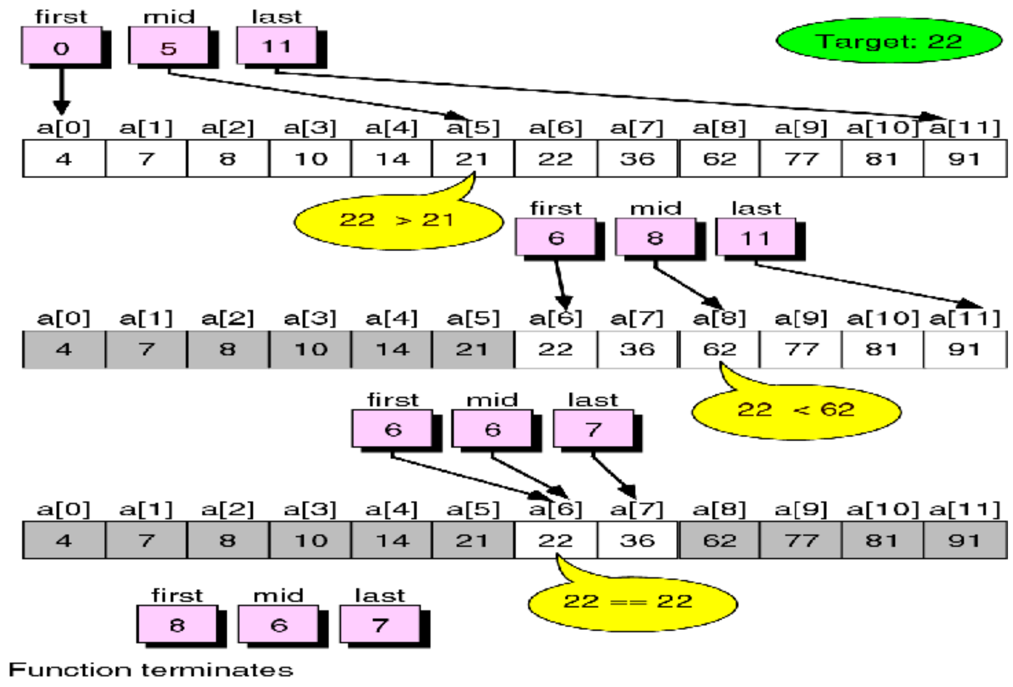
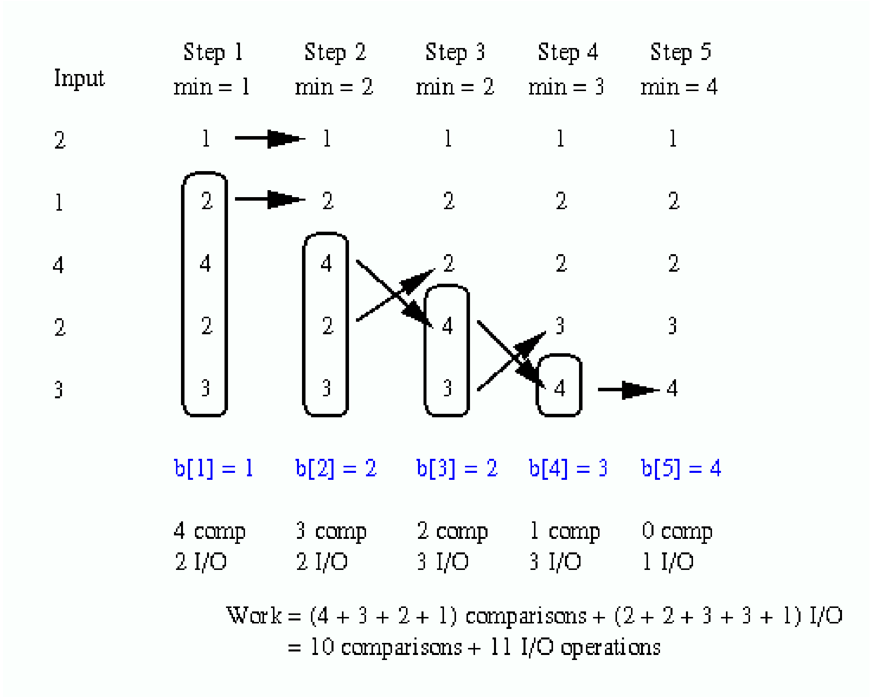
Knowledge Category	**Example/s of category**	**Teaching methodology**
FACT	Algorithm Modularity Searching Sorting.	Write an algorithm to sort three integer numbers.
CONCEPT	i. Assembler (Pass 1 & Pass 2) ii. Format of databases. iii. Hashing.	Flowcharts to explain the Pass 1 & Pass 2 of an assembler with supporting presentations.
PRINCIPLE	Identify the databases, symbols and literals.	On the board design and develop the symbol table and the literal table with student interaction
PROCEDURE	Algorithms for various searching and sorting methods.	Through PowerPoint presentations and videos.
APPLICATION	Implementation of various sorting and searching methods.	Show the real time application of mobile phone contacts on searching and sorting.

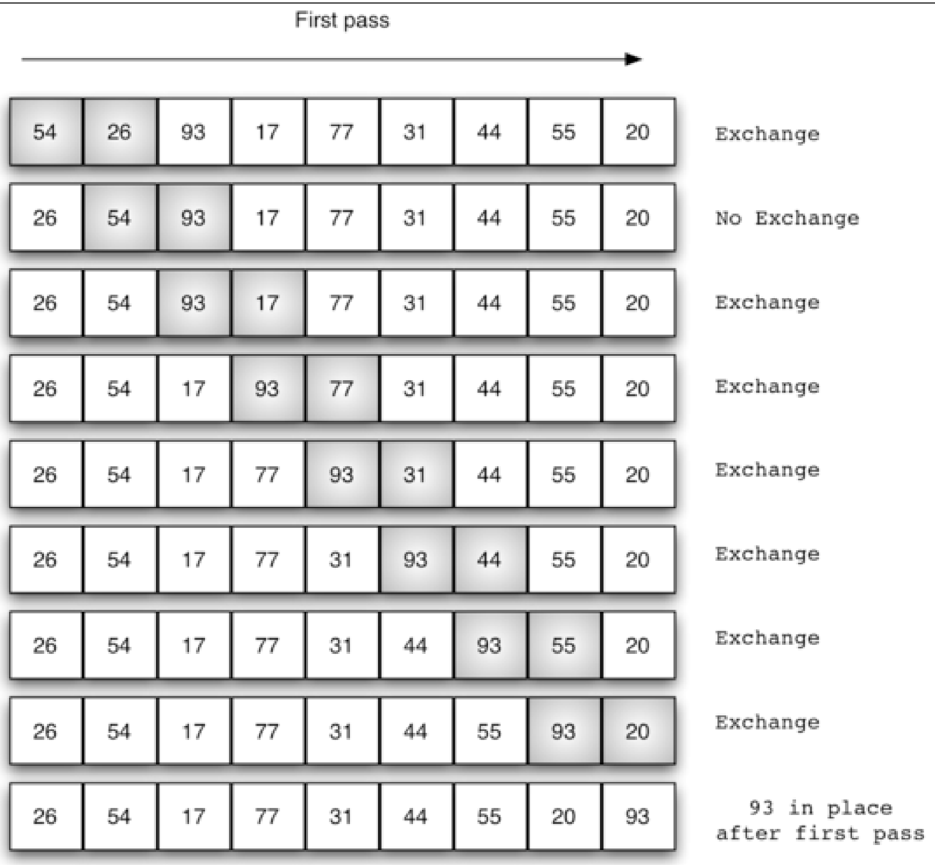
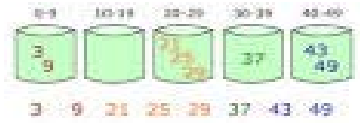
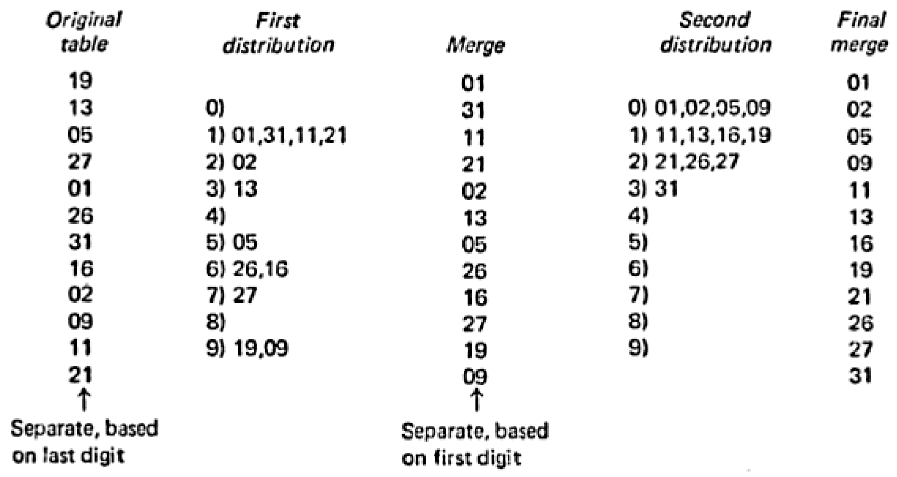
Learning Resources:
Books:
Title:

2) System Programming by J. Donovan.
3) System Programming and Operating System by D.M. Dhamdhere

Teaching Aids:
Video Lectures:
www.youtube.com/watch?v=GNu5YMEhIdg
Websites :
www.linux.com › Learn Linux › Linux New User Guides

Lecture No.	Topic/ Subtopic to be covered
1.	General design procedure of an assembler : teacher shall explain the following points with respect to their functions and significance.  1. Specify the problem 2. Specify data structures 3. Define format of data structures 4. Specify algorithm 5. Look for modularity (i.e., capability of one program to be subdivided into independent programming units) 6. Repeat 1 through 5 on modules Book : John Donovan page number :60
2.	Design of the assembler :Statement of the problem Following example shall be explained by the teacher :  ASSEMBLERS 61 Source program | First pass: Relative address, Mnemonic instruction | Second pass: Relative address, Mnemonic instruction JOHN START 0 USING *,15 L 1,FIVE | 0 | L 1,-(0,15) | 0 | L 1,16(0,15) A 1,FOUR | 4 | A 1,-(0,15) | 4 | A 1,12(0,15) ST 1,TEMP | 8 | ST 1,-(0,15) | 8 | ST 1,20(0,15) FOUR DC F'4' | 12 | 4 | 12 | 4 FIVE DC F'5' | 16 | 5 | 16 | 5 TEMP DS 1F | 20 | - | 20 | - END FIGURE 3.2 Intermediate steps in assembling a program Book : John Donovan page number :61
3.	Data Structure; Format of databases Book : John Donovan page number :66 Formats of the tables like POT, MOT ,PSEUDO OP,symbol table, literal table, shall be explained by the teacher by taking an example of any simple program and formulating the tables on the board by interacting with the students.
4.	Algorithm; Look for modularity Book : John Donovan page number :77 The functions shall be explained to the students considering the two categories as multiuse and unique for pass1 and pass2 of an assembler.

	Teacher shall List the function in pass1 and pass2 of assembler to clarify the modularity concept of the students.
5.	Linear Search; Binary Search : Teacher shall take the following example on the board and show the workout from beginning to the end of the searching process of an element.  Target: 22 first mid last 0 5 11 a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11] 4 7 8 10 14 21 22 36 62 77 81 91 22 > 21 first mid last 6 8 11 22 < 62 first mid last 6 6 7 22 == 22 first mid last 8 6 7 Function terminates
6.	Interchange sort  Input 2 1 4 2 3 Step 1 min = 1 Step 2 min = 2 Step 3 min = 2 Step 4 min = 3 Step 5 min = 4 b[1] = 1 b[2] = 2 b[3] = 2 b[4] = 3 b[5] = 4 4 comp 2 I/O 3 comp 2 I/O 2 comp 3 I/O 1 comp 3 I/O 0 comp 1 I/O Work = (4 + 3 + 2 + 1) comparisons + (2 + 2 + 3 + 3 + 1) I/O = 10 comparisons + 11 I/O operations
7.	Shell sort

	 First pass 54 26 93 17 77 31 44 55 20 Exchange 26 54 93 17 77 31 44 55 20 No Exchange 26 54 93 17 77 31 44 55 20 Exchange 26 54 17 93 77 31 44 55 20 Exchange 26 54 17 77 93 31 44 55 20 Exchange 26 54 17 77 31 93 44 55 20 Exchange 26 54 17 77 31 44 93 55 20 Exchange 26 54 17 77 31 44 55 93 20 Exchange 26 54 17 77 31 44 55 20 93 93 in place after first pass
8.	Bucket sort  0-9 10-19 20-29 30-39 40-49 3 9 21 25 29 37 43 49 3 9 21 25 29 37 43 49 Book : John Donovan page number :86 Example from the book may be solved by the teacher in the class on the board.  *Original table* | *First distribution* | *Merge* | *Second distribution* | *Final merge* 19 | | 01 | | 01 13 | 0) | 31 | 0) 01,02,05,09 | 02 05 | 1) 01,31,11,21 | 11 | 1) 11,13,16,19 | 05 27 | 2) 02 | 21 | 2) 21,26,27 | 09 01 | 3) 13 | 02 | 3) 31 | 11 26 | 4) | 13 | 4) | 13 31 | 5) 05 | 05 | 5) | 16 16 | 6) 26,16 | 26 | 6) | 19 02 | 7) 27 | 16 | 7) | 21 09 | 8) | 27 | 8) | 26 11 | 9) 19,09 | 19 | 9) | 27 21 | | 09 | | 31 ↑ Separate, based on last digit | | ↑ Separate, based on first digit Radix exchange sort

Book :system programming by Donovan page number :88
Teacher shall show the demonstration of sorting the numbers on the board or with the help of presentation.

9. Address calculation sort
*www.c-program-**example**.com/.../c-program-to-implement-**address**.html*
Book :system programming by Donovan page number :89,90
Teacher can use the charts of presentations to show the students how address calculation sort works on numbers.

Hash or Random entry searching
*www.cs.cmu.edu/~adamchik/15-121/lectures/**Hashing/hashing**.html*
Book :system programming by Donovan page number :91
Teacher shall explain the searching with the help of solving the example with the interaction with the students.

10. Comparisons of sort

	Time					
Sort	Average	Best	Worst	Space	Stability	Remarks
Bubble sort	O(n^2)	O(n^2)	O(n^2)	Constant	Stable	Always use a modified bubble sort
Modified Bubble sort	O(n^2)	O(n)	O(n^2)	Constant	Stable	Stops after reaching a sorted array
Selection Sort	O(n^2)	O(n^2)	O(n^2)	Constant	Stable	Even a perfectly sorted input requires scanning the entire array
Insertion Sort	O(n^2)	O(n)	O(n^2)	Constant	Stable	In the best case (already sorted), every insert requires constant time
Heap Sort	O(n*log(n))	O(n*log(n))	O(n*log(n))	Constant	Instable	By using input array as storage for the heap, it is possible to achieve constant space

Merge Sort	O(n*log(n))	O(n*log(n))	O(n*log(n))	Depends	Stable	On arrays, merge sort requires O(n) space; on linked lists, merge sort requires constant space
Quicksort	O(n*log(n))	O(n*log(n))	O(n^2)	Constant	Stable	Randomly picking a pivot value (or shuffling the array prior to sorting) can help avoid worst case scenarios such as a perfectly sorted array.

Topic 3: **Macro Language and Macro processor** **20M**

Specific objectives (As given in curriculum)

1. Comprehend the definition and expansion of macro instructions.
2. Gain insight into design of macro preprocessor.

Additional specific objectives :

- State the need of macros and macro assembler.
- List the operation that can be performed on macros.

Knowledge Category	**Example/s of category**	**Teaching methodology**
FACT	Assembler, symbols and mnemonics	Through presentation and recollecting previous topic
CONCEPT	Macro , macro calls, implementation of macro with assembler	Flowcharts to explain the Pass 1 & Pass 2 of a macro with supporting presentations.
PRINCIPLE	Operation (working) of macro with macro expansion.	On the board design and develop the MOT and the POT with student interaction
PROCEDURE	-----------	-------------
APPLICATION	To identify addressing modes in each instruction and use the instructions in programs.	Explain the usage of instructions in sample programs, Given the set of instructions, determine the output

References material:
Learning Resources:
Books:
Title:

1) System Programming by J. Donovan.
2) System Programming and Operating System by D.M. Dhamdhere

Lecture No.	**Topic / Subject to be covered**
1	Macro Instructions **Difference between macro and function**

No	**Macro**	**Function**
1	Macro is **Preprocessed**	Function is **Compiled**
2	**No Type Checking**	**Type Checking** is Done
3	**Code** Length **Increases**	**Code** Length remains **Same**

4	Use of macro can lead to **side effect**	No **side Effect**
-		
5	Speed of Execution is **Faster**	Speed of Execution is **Slower**
6	Before Compilation macro name is replaced by macro value	During function call , Transfer of Control takes place
7	Useful where small code appears many time	Useful where large code appears many time
8	Generally Macros do not extend beyond one line	Function can be of any number of lines
9	Macro does not Check **Compile Errors**	Function Checks **Compile Errors**

2

Function consumes less memory:

Prior to compilation, all the macro-presences are replaced by their corresponding macro expansions, which consumes considerable memory. On the other hand, even if a function is invoked 100 times, it still occupies the same space. Hence function consumes less memory.

Example for function:

In calling program function call would appear as,

```
c=SUM(x,y);
```

Where sum is the name of the function. And x, y are the arguments passed from calling program to called program.
This statement will invoke the following function SUM. Hence x, y are copied to dummy variables a and b. And then function computes the sum and stores the value in c, which will be returned back to the calling program.

```
int sum(int a, int b)

{int c;c=a+b;
return c;}
```

3

Features of a Macro facility –following features shall be explained by the teacher :

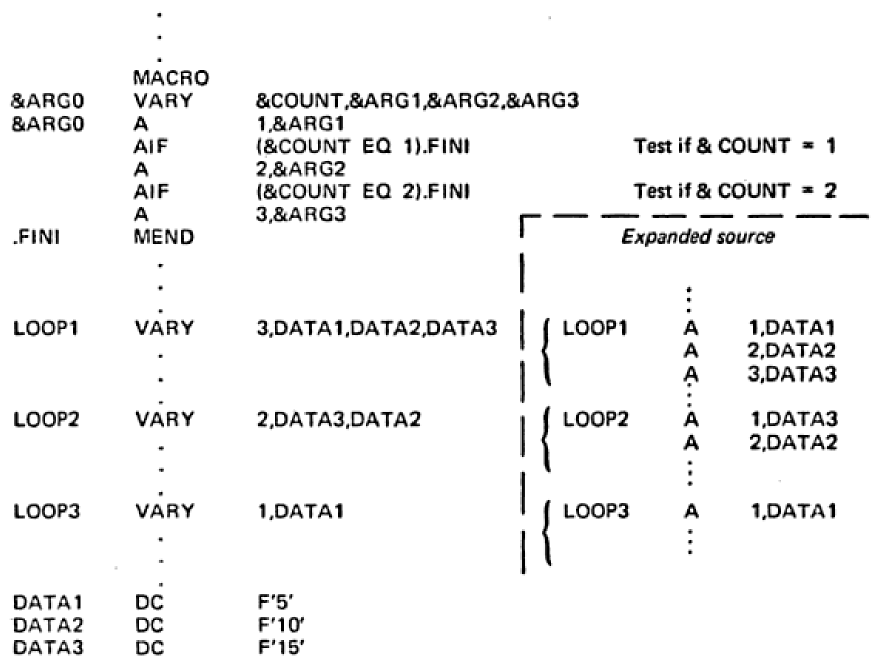
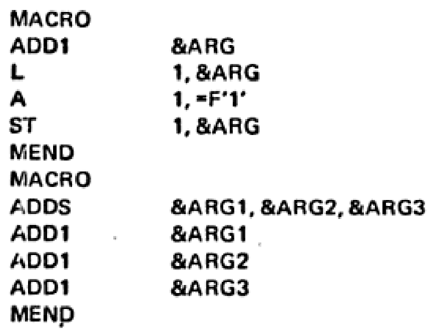
1. Macro Instruction Arguments;
   page no. 88 system programming by John Donovan

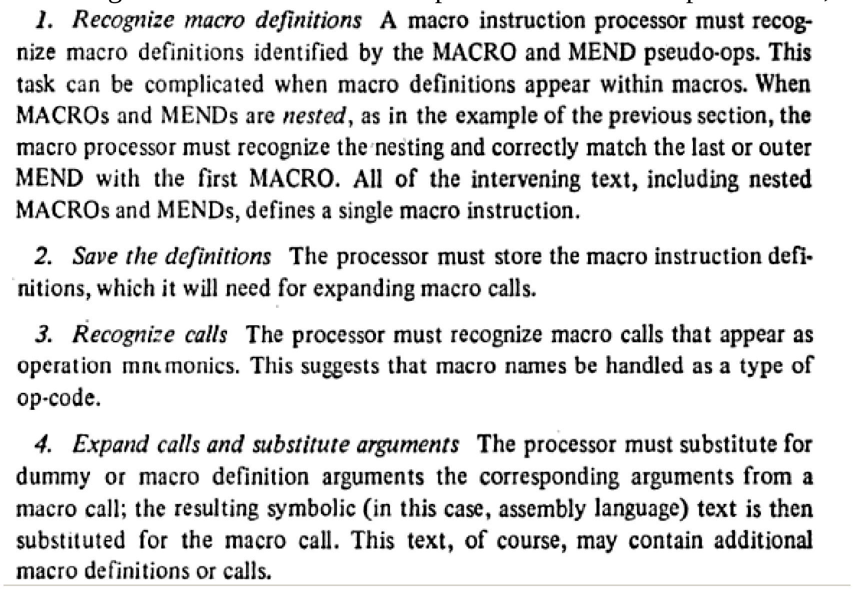
```
                 :
        A        1, DATA 1
        A        2, DATA 1
        A        3, DATA 1
                 :
        A        1, DATA 2
        A        2, DATA 2
        A        3, DATA 2
                 :
DATA 1  DC       F'5'
DATA 2  DC       F'10'
```

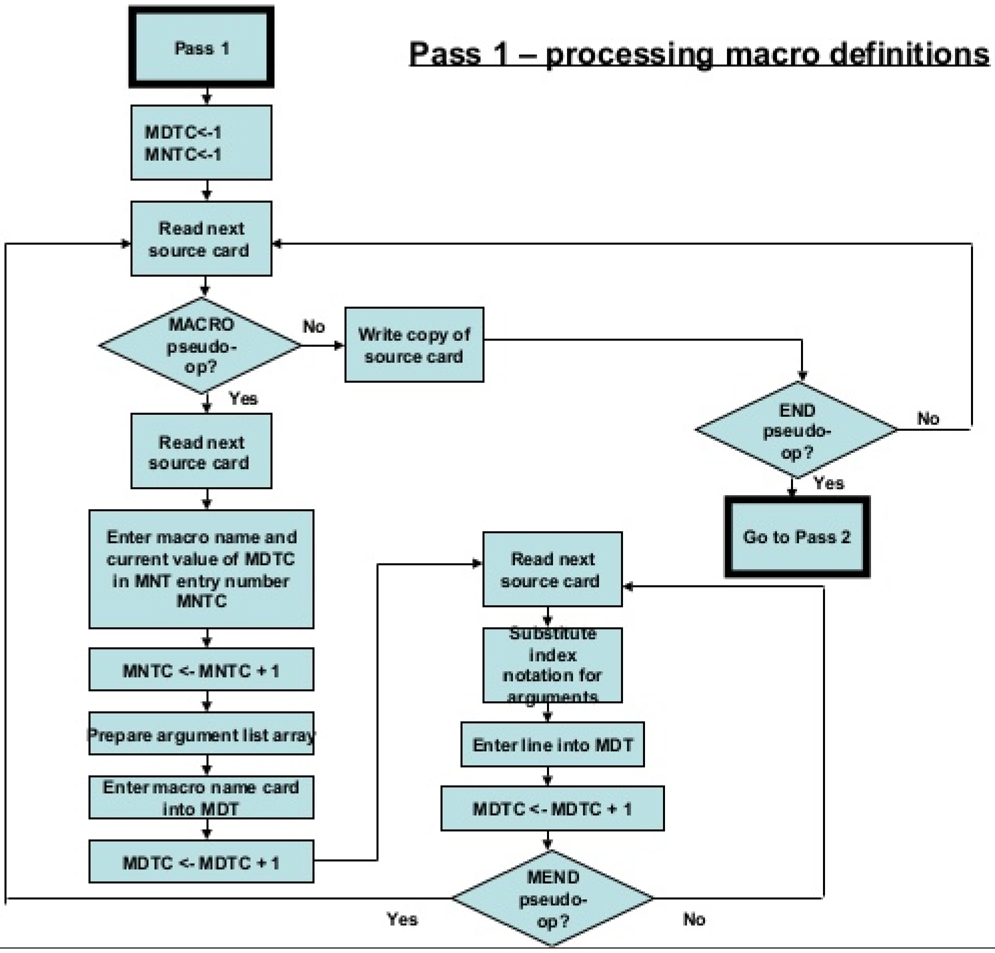
This module can be written as :

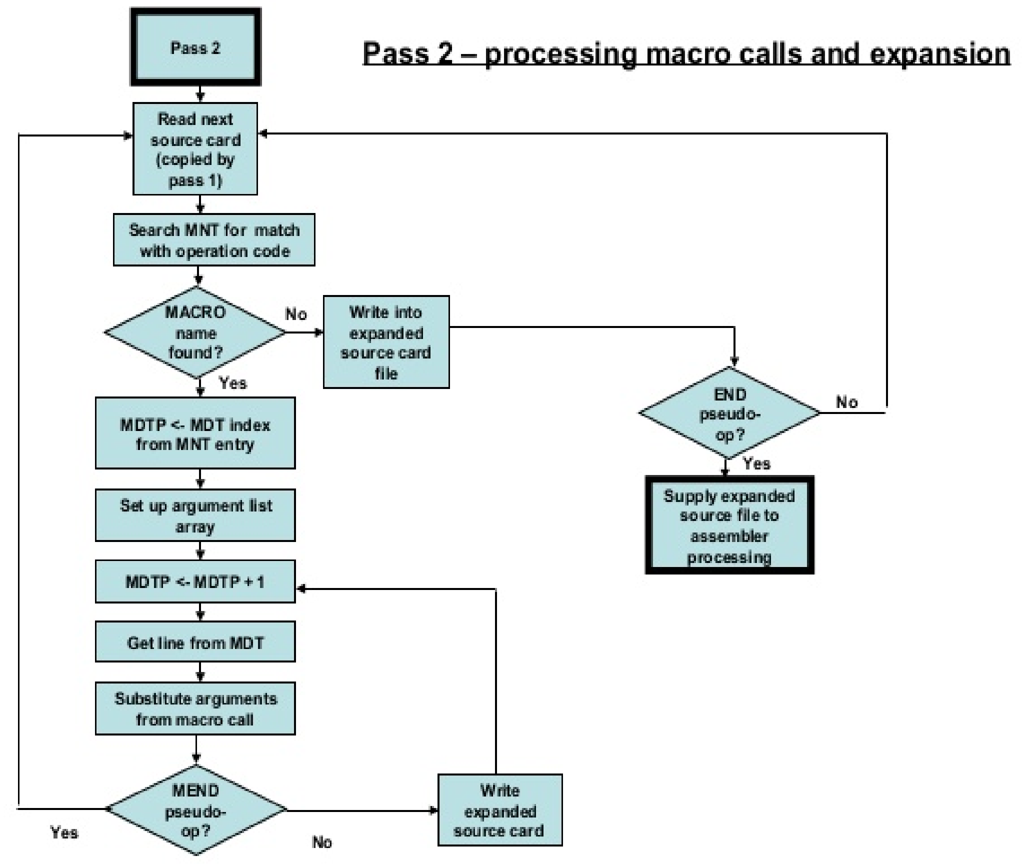
```
                 Source                                  |        Expanded source
        MACRO                     Macro INCR has         |
                                   one argument          |
        INCR      &ARG                                   |
        A         1,&ARG                                 |
        A         2,&ARG                                 |
        A         3,&ARG                                 |
        MEND                                             |
          .                                              |
          .                                              |             :
          .                                              |
        INCR      DATA1           Use DATA1 as           |      { A     1,DATA1
                                  operand                |      { A     2,DATA1
          .                                              |      { A     3,DATA1
          .                                              |             :
          .                                              |
        INCR      DATA2           Use DATA2 as           |      { A     1,DATA2
                                  operand                |      { A     2,DATA2
          .                                              |      { A     3,DATA2
          .                                              |
          .                                              |
DATA1   DC        F'5'                                   | DATA1     DC    F'5'
DATA2   DC        F'10'                                  | DATA2     DC    F'10'
          .                                              |
          .                                              |
          .                                              |
```

2. Conditional macro expansion : page no. 88 system programming by John Donovan
   First the teacher shall explain the macro instruction AGO and AIF which permit the conditional reordering of the sequencing of the macros.
   Then by taking the following example conditional expansion can be explained.

	 . . . MACRO &ARG0 VARY &COUNT,&ARG1,&ARG2,&ARG3 &ARG0 A 1,&ARG1 AIF (&COUNT EQ 1).FINI Test if & COUNT = 1 A 2,&ARG2 AIF (&COUNT EQ 2).FINI Test if & COUNT = 2 A 3,&ARG3 .FINI MEND . . . LOOP1 VARY 3,DATA1,DATA2,DATA3 . . . LOOP2 VARY 2,DATA3,DATA2 . . . LOOP3 VARY 1,DATA1 . . . DATA1 DC F'5' DATA2 DC F'10' DATA3 DC F'15' Expanded source : LOOP1 A 1,DATA1 A 2,DATA2 A 3,DATA3 : LOOP2 A 1,DATA3 A 2,DATA2 : LOOP3 A 1,DATA1 : *www.sas.com/storefront/aux/en/sp**macro**easy/60560_excerpt.pdf*
3	2. Implementation of macro calls within Macros: Designing and implementation of the macro calls within macros shall be explained by recalling the macro calls within macros features and its expansion. Implementation of the macro call within macros must be explained by taking an example in the class. Databases and the flowchart shall be explained by the teacher with the help of the chart of presentations. 1. More information at : site : www.csie.ntnu.edu.tw/~ghhwang/course_slices/.../Chapter4.pdf
4	2 . Macro call within Macros; Page no 119 system programming by John Donovan For the given module below :  MACRO ADD1 &ARG L 1, &ARG A 1, =F'1' ST 1, &ARG MEND MACRO ADDS &ARG1, &ARG2, &ARG3 ADD1 &ARG1 ADD1 &ARG2 ADD1 &ARG3 MEND Within the definition of macros ADDS are 3 separate calls to the previously defined macro ADD1. Teacher shall explain the students how to call the macros within macros by taking the above

	example and show it expanded with the help of board by actually solving the problem. 3. Macro Instruction defining Macros Page no 121 ,system programming by Donovan Various instructions defining macros shall be explained by the teacher as MACRO which specifies the macros, DEFINE which defines the macros, MEND which ends the macro. Further students should be able to identify the macro calls and the PPTS ON *: www.ai.mit.edu/projects/dynlangs/oodl-course/.../jb-**macros.ppt*** PPTS ON : *www2.kuas.edu.tw/prof/me06/part-2/2_10-**macro.ppt***
5	Implementation of restricted faculty: Two Pass Algorithm Following 4 main tasks of the Macro processors shall be explained first,  *1. Recognize macro definitions* A macro instruction processor must recognize macro definitions identified by the MACRO and MEND pseudo-ops. This task can be complicated when macro definitions appear within macros. When MACROs and MENDs are *nested*, as in the example of the previous section, the macro processor must recognize the nesting and correctly match the last or outer MEND with the first MACRO. All of the intervening text, including nested MACROs and MENDs, defines a single macro instruction. *2. Save the definitions* The processor must store the macro instruction definitions, which it will need for expanding macro calls. *3. Recognize calls* The processor must recognize macro calls that appear as operation mnemonics. This suggests that macro names be handled as a type of op-code. *4. Expand calls and substitute arguments* The processor must substitute for dummy or macro definition arguments the corresponding arguments from a macro call; the resulting symbolic (in this case, assembly language) text is then substituted for the macro call. This text, of course, may contain additional macro definitions or calls. Page no. 123, System Programming by Donovan Teacher shall also explain the specification of databases used in algorithm such as MNT,MDT,MDTC,MNTC,MLA,MDTP. www.csie.ntnu.edu.tw/~ghhwang/course_slices/.../Chapter4.pdf ppts on *:* *in1.csie.ncu.edu.tw/~chia/Course/Assembly/**macro**.ppt*
6	1. A Single Pass Algorithm PAGE NO 127 System programming by Donovan The tasks to be performed while designing of a single pass algorithm shall be explained first. The databases are required to be explained. Flowchart of the single pass algorithm shall be explained with the help of chart or presentation.
7	Implementation of macro calls within Macros:

	Designing and implementation of the macro calls within macros shall be explained by recalling the macro calls within macros features and its expansion. Databases and the flowchart shall be explained by the teacher with the help of the chart of presentations. More information at : site : *www.csie.ntnu.edu.tw/~ghhwang/course_slices/.../Chapter4.pdf*
8	Implementation within an assembler: Page no 140 System programming by Donovan Flowchart of the designing shall be explained with the help of chart by the teacher. Handouts are available on : *www.cs.tufts.edu/comp/40/handouts/um**asm**.pdf*
9	 Pass 1 – processing macro definitions Pass 1 MDTC<-1 MNTC<-1 Read next source card MACRO pseudo-op? No Write copy of source card Yes END pseudo-op? No Yes Go to Pass 2 Read next source card Enter macro name and current value of MDTC in MNT entry number MNTC MNTC <- MNTC + 1 Prepare argument list array Enter macro name card into MDT MDTC <- MDTC + 1 Read next source card Substitute index notation for arguments Enter line into MDT MDTC <- MDTC + 1 MEND pseudo-op? Yes No

10	Pass 2 Pass 2 – processing macro calls and expansion Read next source card (copied by pass 1) Search MNT for match with operation code MACRO name found? No Write into expanded source card file Yes MDTP <- MDT index from MNT entry Set up argument list array MDTP <- MDTP + 1 Get line from MDT Substitute arguments from macro call MEND pseudo-op? Yes No Write expanded source card END pseudo-op? No Yes Supply expanded source file to assembler processing

Topic 4**: LOADERS AND LINKING** **20M**

Objective(As per curriculum)

1. Understand the concepts and requirements of loading and linking.
2. Gain insite into the design of linker.

Additional Specific Objectives

- State the importance of overlays, dynamic binder.
- Explain the format of data bases used in direct linking loader.

Knowledge Category	**Example/s of category**	**Teaching methodology**
FACT	Loaders and linkers	Through presentations
CONCEPT	Loader, linkers, loader schemes, designing of loaders	Charts to explain the functions of loaders and linkers with supporting presentations.
PRINCIPLE	Process of loading and linking process	Example of a program getting loaded in main memory and linking with various other files for its execution. Practically demo can be shown. Data structures and databases.
PROCEDURE	Designing of a loader and linkers	Explain the various design methodologies with the help of charts or diagrams and compare.
APPLICATION	To identify addressing modes in each instruction and use the instructions in programs.	Explain the usage of loaders and linkers for programming and execution. Implementation of loaders and linkers.

References material:

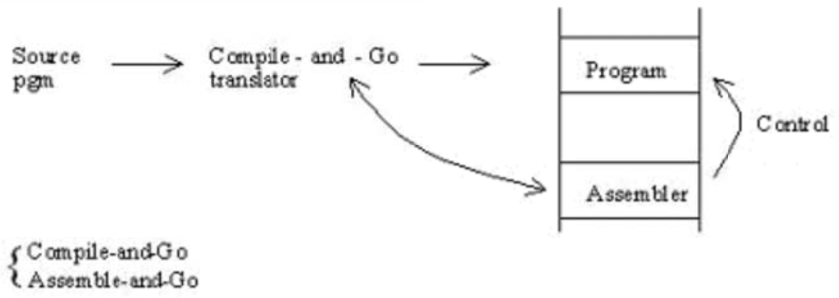
Learning Resources:

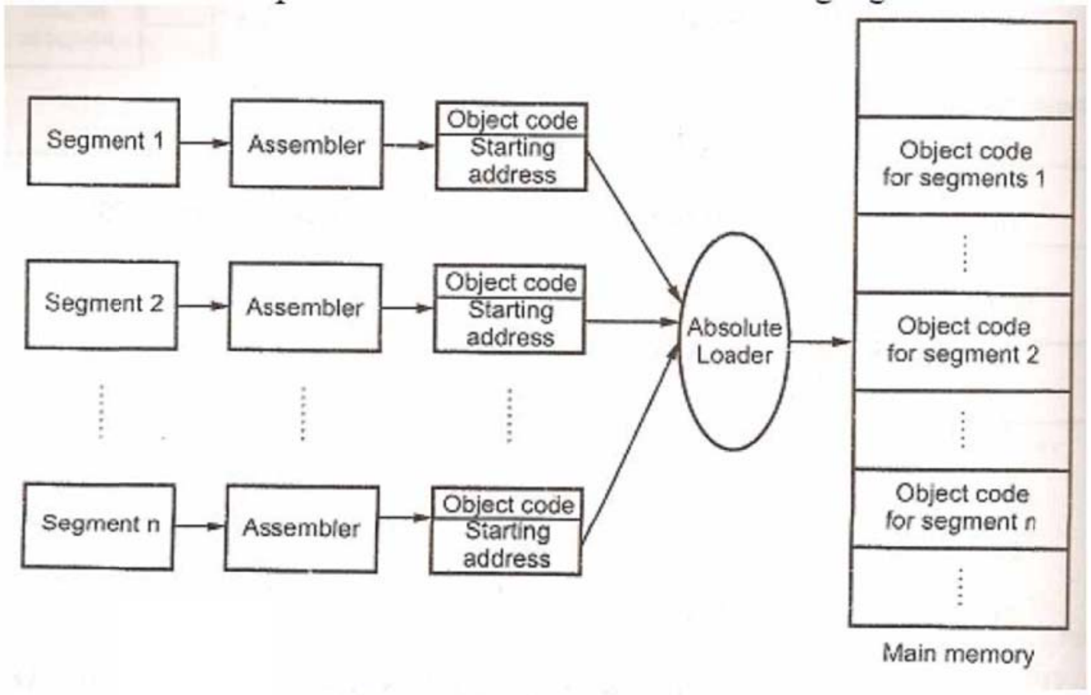
Books:

Title:

1) System Programming by J. Donovan.
2) System Programming and Operating System by D.M. Dhamdhere

Lecture No.	**Topic / Subject to be covered**
1	Compile and go" loaders scheme Page no. 150 system programming by Dononvan Teacher shall explain the following diagram with respect to the blocks :

	 Source pgm Compile - and - Go translator Program Assembler Control Compile-and-Go Assemble-and-Go Also the teacher shall explain the working of the same with the help of presentations. www.just.edu.jo/~munzer/Courses/CPE471/Lec/**loader/loaders.ppt**
2	General Loader Schemes; Page no. 151 system programming by Dononvan In this the loader accepts the assembled machine instructions, data & other information present in the object format & places machine instructions & data in core in an executable format. The loader is smaller than assembler so more memory is available to the user. Now we don't need to retranslate the program. If all the source program translators produce compatible object programs & use compatible linkage conventions, it is possible to write subroutines in different languages. Absolute Loaders In this scheme the object program is placed in secondary devices. The loader only accepts the machine language text & places into core at the location prescribed by the assembler. Disadvantage is that the programmer must specify the load address in the program & also if there are multiple subroutines, the programmer must remember the address of each & use that in other subroutines to perform subroutine linkages. *in1.csie.ncu.edu.tw/~chia/Course/SP/**loader**.ppt*
3	Subroutine linkages: Page no. 154 system programming by Dononvan Relocating loaders : In this the assembler assembles each procedure segment independently & passes onto the loader the text & information as to relocation & intersegment references. For each source program the assembler outputs a text prefixed by the transfer vector that consists of addresses containing the names of subroutines referenced by the source program. The assembler would also provide to the loader the length of the program & length of the transfer vector. The loader will load each subroutine identified in the Transfer vector. It will then place a transfer instruction to the corresponding subroutine in each entry in the TV. Direct linking loaders The direct linking loader is the most common type of loader. This type of loader is a relocatable loader. The loader cannot have the direct access to the source code. And to place

	the object code in the memory there are two situations: either the address of the object code could be absolute which then can be directly placed at the specified location or the address can be relative. If at all the address is relative then it is the assembler who informs the loader about the relative addresses. The assembler should give the following information to the loader 1)The length of the object code segment 2) The list of all the symbols which are not defined 111 the current segment but can be used in the current segment. 3) The list of all the symbols which are defined in the current segment but can be referred by the other segments. *bit.kuas.edu.tw/~csshieh/teach/93A/sp/note/sp07.**ppt***
4	1. Binders and Linking loaders : teacher shall explain the binders and linking loaders with the help of program execution process. 2. Teacher shall actually demonstrate the process of binding and loading while execution of a program is on. 3. Overlays and Dynamic Binders: Sometimes a program may require more storage space than the available one Execution of such program can be possible if all the segments are not required simultaneously to be present in the main memory. In such situations only those segments are resident in the memory that area ctually needed at the time of execution *www.just.edu.jo/~munzer/Courses/CPE471/Lec/**loader/loaders.ppt***
5	Design of Absolute loaders : Page no. 167 system programming by Dononvan Process of absolute loading shall be explained with the help of the diagram given below. Teacher shall explain the functions of the components shown in the diagram.  Segment 1 Assembler Object code Starting address Segment 2 Assembler Object code Starting address Segment n Assembler Object code Starting address Absolute Loader Object code for segments 1 Object code for segment 2 Object code for segment n Main memory **Process of absolute loading** *www.cs.thu.edu.tw/files/sp_chap3.**ppt***

6	Other loader schemes Other loader schemes like subroutine linkages, linkage editors shall be explained by the teacher.
7	Other loader schemes Other loader schemes like subroutine linkages, linkage editors shall be explained by the teacher.
8	Specification of data structures Page no. 176 system programming by Dononvan The data structures used in designing of a loader shall be explained by the teacher. pass1 1. input object deck 2. Initial Program Load Address(IPLA) 3. Program Load Address(PLA) counter 4. Global External Symbol Table(GEST) 5. A copy of the input (TXT card) 6. RLD (Relocation & Linkage Directory) 7. END card pass2 copy of object program IPLA parameter (Initial Program Load Address) supplied by the OS or programmer to specify the address to load the first segment PLA counter (Program Load address) to keep track of each segment location GEST (Global external symbol table) to store each external symbol and its corresponding assigned core address Local External Symbol Array(LESA) to establish correspondence between ESD ID used in ESD & RLD cards ESD (external symbol dictionary)
9	Format of database Databases shall be explained such as : 1. ESD card Source card reference Name Type ID Relative address Length 1 John SD 01 0 64 2 Result LD 02 52 3 Sum ER 03 -- TYPE SD (Segment Definition) 01 START CSECT LD (Local Definition) 02 ENTRY ER (External Reference) 03 EXTRN It contains all symbols that are defined in this program but that may be referenced elsewhere and all symbols referenced in this program but defined elsewhere.

	2. RLD Card Source card reference ESD ID Length Flag (+/-) Relative address 14 01 4 + 48 17 03 4 + 60 Contain information about those locations whose content depend on the address at which program is placed. It contains the location& length of each constant that needs to be changed due to relocation -By what it has to be changed -Operation to be performed ESD CARD AND GST shall be explained in the same manners.
10	Algorithm Page no 181 system programming by Donovan ‰ In Pass 1, concerned only Header and Defined records. ‰ CSADDR+CSLTH = the next CSADDR. ‰ A load map is generated. ‰ In Pass 2, as each Text record is read, the object code is moved to the specified address (plus the current value of CSADDR). ‰ When a Modification record is encountered, the symbol whose value is to be used for modification is looked up in ESTAB. ‰ This value is then added to or subtracted from the indicated location in memory.

Topic 5 : Compiler **24M**

Objective as per curriculum
1. Understand the aspects of compilation of high Level language
2. Describe the various phases of compilers.
3. Discuss about memory allocation schemes used in compiler

Specific Objectives :
- ➢ State the functions of Compiler
- ➢ State the phases of compiler

Knowledge Category	Example/s of category	Teaching methodology
FACT	Compiler	Through presentations and charts
CONCEPT	Translation ,syntactic units,	Explaining how the translation

	interpretation ,storage allocation	takes place with an example and with the help of charts and presentations
PRINCIPLE	Process of translation	Explanation about how the program gets translated and compiled.
PROCEDURE	designing of various types of compilers	Explain the various phases of compiler with the help of charts and presentations.
APPLICATION	To identify various phases in compilation and procedure to design compilers and implementation of algorithms.	Explaining the algorithms with help of charts by taking valid example so that the students will understand what happens in every phase of design.

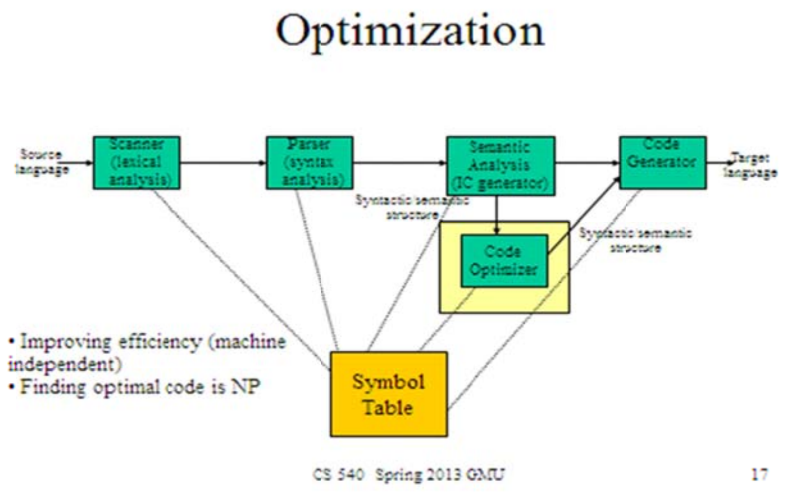
Learning Resources::
Books:

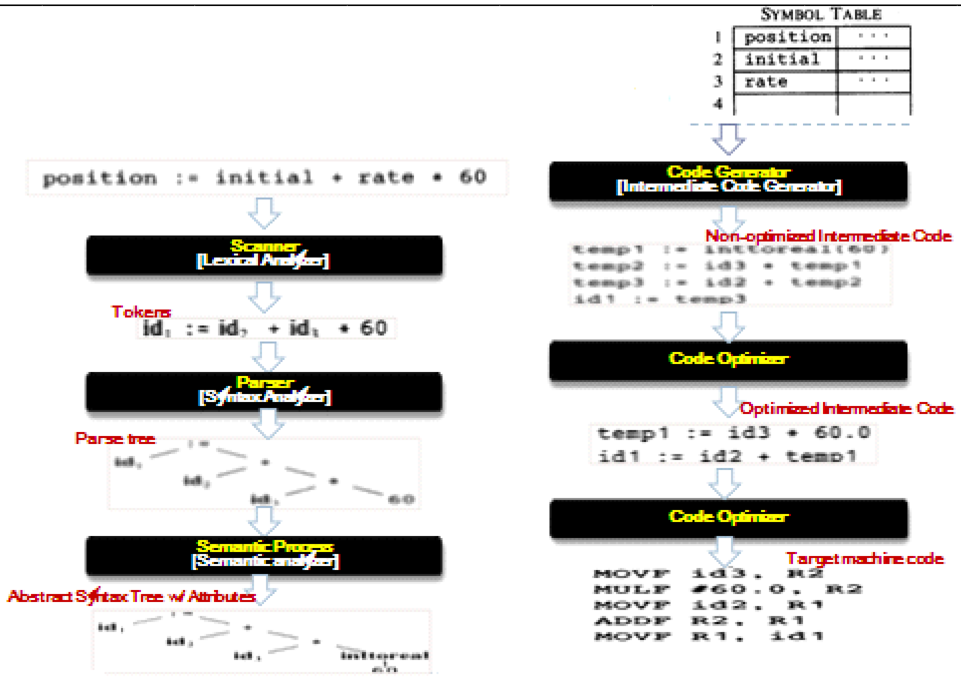
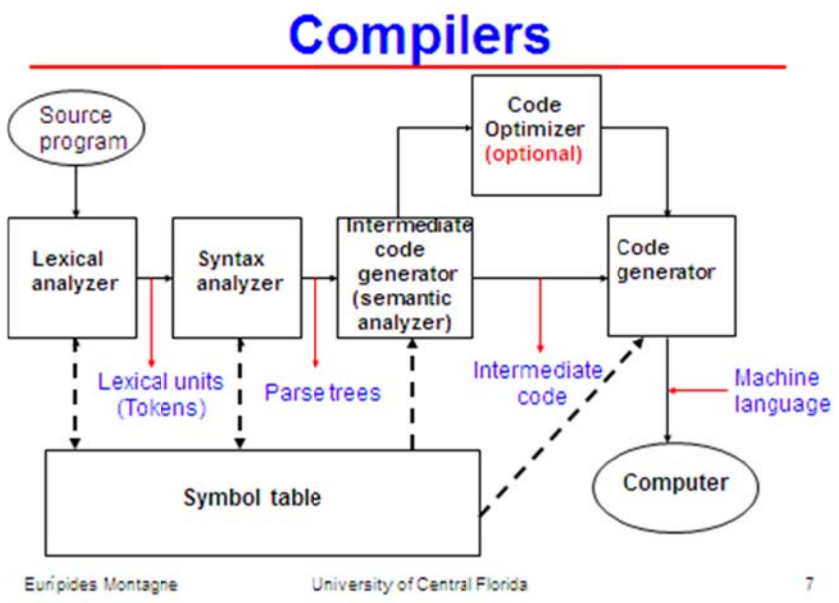
1. System Programming by J. Donovan.
2. System Programming and Operating System by D.M. Dhamdhere
3. Compiler Design by G.Sudha Sadashiv, SciTech
4. System Programming by Rakesh K Mourya ,Dreamtech

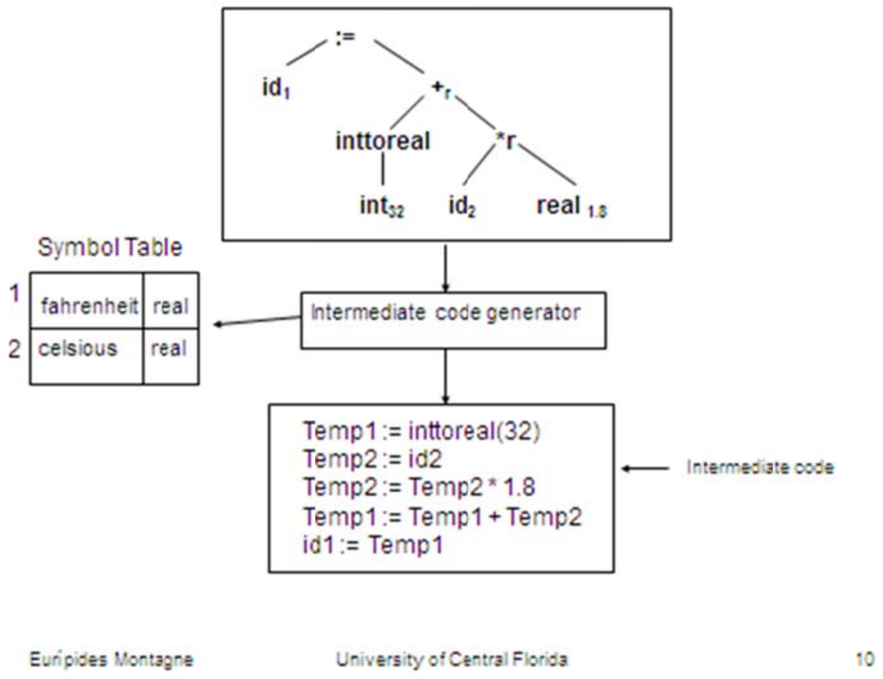
Teaching Aids: Books, CDs, PPTs, Charts etc.
Websites:

www.dreamtechpress.com (PPTs available)
www.cs.princeton.edu/~appel/modern(for compiler implementation in Java/ML/C)

Lecture No.	Topic/ Subtopic to be covered
1.	Statement of problem Page No. 265 (Donovan) ■ Compilation of a program proceeds through a fixed series of phases ☐ Each phase use an (intermediate) form of the program produced by an earlier phase ☐ Subsequent phases operate on lower-level code representations ■ Each phase may consist of a number of passes over the program representation ☐ Pascal, FORTRAN, C languages designed for one-pass compilation, which explains the need for function prototypes ☐ Single-pass compilers need less memory to operate ☐ Java and ADA are multi-pass
2.	Intermediate forms

	<u>Intermediate code:</u> Produces a program in a different language representation: Assembly language Similar to assembly language Something higher than assembly language Note: semantic analysis is an integral part of the intermediate code generator
3.	Storage allocation Page No.301(Donovan) ▶ Instruction sets for some popular architecture are highly no uniform. ▶ High-level programming language operations are not always easy to support. ▶ Ex. exceptions, threads, dynamic heap access … ▶ Exploiting architectural features such as cache, distributed processors and memory ▶ Effective use of a large number of processors
4.	Code generation :machine optimization (machine dependent and machine independent) Page No. 275 (Donovan)  Source language Scanner (lexical analysis) Parser (syntax analysis) Semantic Analysis (IC generator) Code Generator Target language Syntactic semantic structure Code Optimizer Syntactic semantic structure Symbol Table Optimization • Improving efficiency (machine independent) • Finding optimal code is NP CS 540 Spring 2013 GMU 17
5.	Assembly phase

	 SYMBOL TABLE 1 position ... 2 initial ... 3 rate ... 4 position := initial + rate * 60 Scanner [Lexical Analyzer] Tokens id1 := id2 + id3 * 60 Parser [Syntax Analyzer] Parse tree Semantic Process [Semantic analyzer] Abstract Syntax Tree w/ Attributes inttoreal 60 Code Generator [Intermediate Code Generator] Non-optimized Intermediate Code temp1 := inttoreal(60) temp2 := id3 * temp1 temp3 := id2 + temp2 id1 := temp3 Code Optimizer Optimized Intermediate Code temp1 := id3 * 60.0 id1 := id2 + temp1 Code Optimizer Target machine code MOVF id3, R2 MULF #60.0, R2 MOVF id2, R1 ADDF R2, R1 MOVF R1, id1
6	General model of a compiler : diagram and explanation  Compilers Source program Lexical analyzer Syntax analyzer Intermediate code generator (semantic analyzer) Code Optimizer (optional) Code generator Lexical units (Tokens) Parse trees Intermediate code Machine language Symbol table Computer Euripides Montagne University of Central Florida 7
7	Lexical phase : databases, algorithm Page no 281 (Donovan) **Lexical analyzer:** Gathers the characters of the source program into lexical units. Lexical units of a program are:

	identifiers special words (reserved words) operators special symbols **Comments are ignored!**
8	Syntax phase : databases, algorithm Page no. 283 (Donovan) **Syntax analyzer:** Takes lexical units from the lexical analyzer and use them to construct a hierarchical structure called **parse tree** Parse trees represent the syntactic structure of the program.
9	Interpretation phase : databases, algorithm  := id1 +r inttoreal *r int32 id2 real 1.8 Symbol Table 1 fahrenheit real 2 celsious real Intermediate code generator Intermediate code Euripides Montagne University of Central Florida 10Temp1 := inttoreal(32) Temp2 := id2 Temp2 := Temp2 * 1.8 Temp1 := Temp1 + Temp2 id1 := Temp1
10	Optimization : databases, algorithm Code Optimization Find More Efficient Ways to Execute Code Replace Code With More Optimal Statements Common optimisations include: • removing redundant identifiers, • removing unreachable sections of code, • identifying common subexpressions,

	• unfolding loops and • eliminating procedures.
11	Code generation : databases, algorithm Page No 306 (Donovan) From the machine-independent form assembly or object code is generated by the compiler MOVF id3, R2 MULF #60.0, R2 MOVF id2, R1 ADDF R2, R1 MOVF R1, id1 This machine-specific code is optimized to exploit specific hardware features
12	Assembly phase : databases, algorithm and assembly phase Page No 313 (Donovan)

Topic 6 : Parsing **10M**

Objectives as per curriculum:

- Identify and understand the role of a lexical and syntax analyzer.
- Understand the top-down and bottom-up parsing techniques**.**

Additional Specific Objectives

- State the functions of Lexical Analyzer.
- State the functions of Top Down Parser and Bottom Up Parser.

Knowledge Category	**Example/s of category**	**Teaching methodology**
FACT	Lexical phase and syntax	Through presentations and charts and examples
CONCEPT	Syntax analysis and semantic analysis	Role of syntax analysis and semantics analysis can be explained with help of examples of programs.
PRINCIPLE	Debugging	Explaining what are the actions taken syntax and semantics of the statements are not correct while programming. Demo of execution of a program can be shown.
PROCEDURE	Design procedure for the development of parsers	Explanation with the help of charts and presentations or workouts on board with some examples.
APPLICATION	To identify the role of lexical and syntax analyzer.	Explanation of top down parsing and bottom up parsing with examples using presentations. Both the approaches can be explained on the board by taking some examples.

Learning Resources::

Books:

1. System Programming by J. Donovan.
2. System Programming and Operating System by D.M. Dhamdhere
3. Compiler Design by G.Sudha Sadashiv, SciTech
4. System Programming by Rakesh K Mourya ,Dreamtech

Teaching Aids: Books, CDs, PPTs, Charts etc.

Websites:

www.dreamtechpress.com (PPTs available)
www.cs.princeton.edu/~appel/modern (for compiler implementation in Java/ML/C)

Lecture No.	**Topic/ Subtopic to be covered**
1.	Parsing : significance and use of parsing ■ Checks whether the token stream meets the grammatical specification of the language and generates the syntax tree. ☐ A syntax error is produced by the compiler when the program does not meet the grammatical specification. ☐ For grammatically correct program, this phase generates an internal representation that is easy to manipulate in later phases ■ Typically a syntax tree (also called a parse tree). ■ A grammar of a programming language is typically described by a context free grammar, which also defines the structure of the parse tree. Parsing approaches and their significances. Discovering the derivation of a string: *If one exists* Harder than generating strings **Two major approaches** • Top-down parsing • Bottom-up parsing
2.	Top down parser LL(1), recursive descent Start at the root of the parse tree and grow toward leaves Pick a production & try to match the input Wrong “pick” → may need to backtrack Bottom up parser. LR(1), operator precedence Start at the leaves and grow toward root As input is consumed, encode possible parse trees in an internal state Bottom-up parsers handle a large class of grammars

5.2 Planning and Conduct of Test:

a) The time table and sample test paper for the test should be displayed minimum 10 days before the test.
b) Each test will be of 25 marks.
c) First test should cover about 40% of curriculum and second test should cover remaining curriculum.

Sr. No	Class Test	Marks	Topics
1	Class Test 1	25	Topic 1, Topic 2, Topic 3.1, Topic 3.2 (data transfer instructions, arithmetic instructions)
2	Class Test 2	25	Topic 3.2 remaining topics to end of the chapter. Topic 4, Topic 5, Topic 6

5.1 **Details about conduct of assignments:**

- ✓ After completion of each chapter one assignment should be given.
- ✓ Assignment question shall be given from sample question paper, old MSBTE question papers
- ✓ It shall be assessed by subject teacher before giving next Assignment.
- ✓ Evaluation of Assignment should be done effectively.
- ✓ Sample question paper of Microprocessor and Programming to be solved by every student

5.4 Strategies for Conduct of Practical:

5.4.1 Suggestions for effective conduct of practical and assessment:

- Display the Date wise schedule of the experiment to be performed in the Laboratory.
- At the beginning of the semester teacher/lab assistant should check and ensure that the Computers and Assembler software used for the experiments are installed.
- Before start of any practical Teachers should explain the specific objective of that particular practical.
- Teacher should divide total students into number of group as given in practical manual.
- Teacher should refer the guidelines given in the lab manual.
- Teacher should make the students aware of instructions given in the lab manual.
- Teacher should ensure that the activities given in the Lab Manual are performed by the student and observations should be tabulated.
- Teacher shall assess the performance of students continuously as per norms prescribed by MSBTE CIAAN norms.
- During assessment teacher is expected to ask questions to the students to tap their achievements regarding related knowledge and skills so that students can prepare while submitting record of the practical. Focus should be given on development of enlisted skills rather than theoretical / codified knowledge.

5.4.3 Preparation for conduct of practical

Sr. No.	Activity	Duration
1.	Teacher shall explain the objectives of the experiment.	10 Min.
2.	Teacher shall demonstrate the execution and the desired output.	10 Min
3.	Teacher shall make the students perform the execution of programs and check the output by changing the input.	30 Min.
4.	Record the observations in the manual and write the appropriate outputs.	30 Min.
4.	Teacher shall evaluate the students’ performance as per the CPA table.	40 Min.

6.0 Mode of assessment:

6.1 Class Test:

- There will be two tests each of 25 marks.
- The tests will be conducted as per the MSBTE schedule.
- Teacher should prepare model answer of class test question papers.
- After completion of test, subject teacher should display model answer on Department Notice Board.
- Teacher should show the answer paper of class test to the student and discuss about the mistakes.
- Teacher should maintain the record of class test as per MSBTE norms (CIAAN)
- Format for question paper should be as per the sample question paper supplied by MSBTE.
- Guidelines for Setting Class Test Question Paper:
  - ❖ Question no.1 Attempt any three out of four (3X3=9 Marks)
  - ❖ Question no.2 Attempt any two out of three (2X4=8 Marks)

Or

- ❖ Question no.2 Attempt any one out of two (1X8=8 Marks)
- ❖ Question no.3 Attempt any two out of three (2X4=8 Marks)

6.1.1 Sample Test Papers:

Sample Test Paper 1

Institute Name:

Course Name: Diploma in Computer Science and Engg Course Code: CW

Semester: SIXTH

Subject**:** System Programming

Marks: **25** Time: **1 hour**

Instructions:

1. All questions are compulsory
2. Illustrate your answers with neat sketches wherever necessary
3. Figures to the right indicate full marks
4. Assume suitable data if necessary
5. Preferably, write the answers in sequential order

Q1. Attempt any THREE **09(3*3)**

a. Define module. Give importance of modularity.
b. Draw the labelled machine structure diagram of system programming.
c. Draw the RS format of instruction and describe the fields in it.
d. Write the syntax of following instructions with suitable example :
   i. **START** **ii. USING**

Q2. Attempt any TWO **08(4*2)**

a. Draw the neat labelled diagram of foundation of system programming.
b. Give the formats of databases for the design of assembler.
c. List the steps of algorithm of shell sort and give example to sort five numbers.

Q3. Attempt any One **08(8*1)**

a. Sort the given numbers using Bucket sort technic with the pictorial representation in ascending order.
   110, 628, 576, 78, 9915,38,692,42,69,21,83,95
b. Write the binary search algorithm and give the example to search a number from the list of 10 numbers.

6.1.2 **Sample Test Paper 2**

Institute Name:

Course Name: Diploma in Computer Science and Engg Course Code: CW

Semester : SIXTH

17634

Subject**:** System Programming

Marks: **25** Time: **1 hour**

Instructions:

1. All questions are compulsory
2. Illustrate your answers with neat sketches wherever necessary
3. Figures to the right indicate full marks
4. Assume suitable data if necessary
5. Preferably, write the answers in sequential order

Q1. Attempt any THREE **09(3*3)**

a. Describe the working of compile and go loader with neat labelled sketch.
b. Draw the parse tree for the sting 'ccdeef' using bottom up parsing approach.
c. List the tasks performed by macro processor.
d. Write the four functions performed by loader.

Q2. Attempt any TWO **08(4*2)**

a. Write the expanded source code for the following code.

```
    MACRO
    Sub_div
    B       1,DATA
    B       2,DATA
    B       3,DATA
    MEND
    .
    .
    Sub_div
    .
    .
    Sub_div
DATA DC         F'10'
```

b. Describe the working of interpretation phase of compiler.

c. State the importance of overlays in linking loaders.

Q3. Attempt any one **08(8*1)**

a. Write the contents of macro name table and macro definition table for the following code.

```
Macro
A          &c_d, &e_d, &reg
Move       &reg, &c_d
Sub        &reg, &e_d
Mov        &reg, &c_d
Mend
```

b. What is done in code optimization process? Why is it required? Write one suitable example.

Scheme – G

Sample Question Paper

Course Name: - Computer Science and Engineering Group

Course Code:- CW

Semester: - SIXTH

Subject Title: - SYSTEM PROGRAMMING **17634**

Marks: - 100 Marks

Instructions

1. All questions are compulsory
2. Illustrate your answer with neat sketches wherever necessary
3. Figures to the right indicates full marks
4. Assume suitable data if necessary
5. Preferably, write the answers in sequential order

Q 1(a) Answer any **THREE**of the following (12)

i. Draw a neat labeled diagram of foundation of system software.

ii. Draw RR format of instruction. Describe the fields in it.

iii. Define Macro. State two uses of macros in a program.

iv. Write the expanded source code for the following :

```
Macro
Incr
A   1, Data
A   2, Data
A   3, Data
Mend
.
.
.
Incr
.
```

.

.

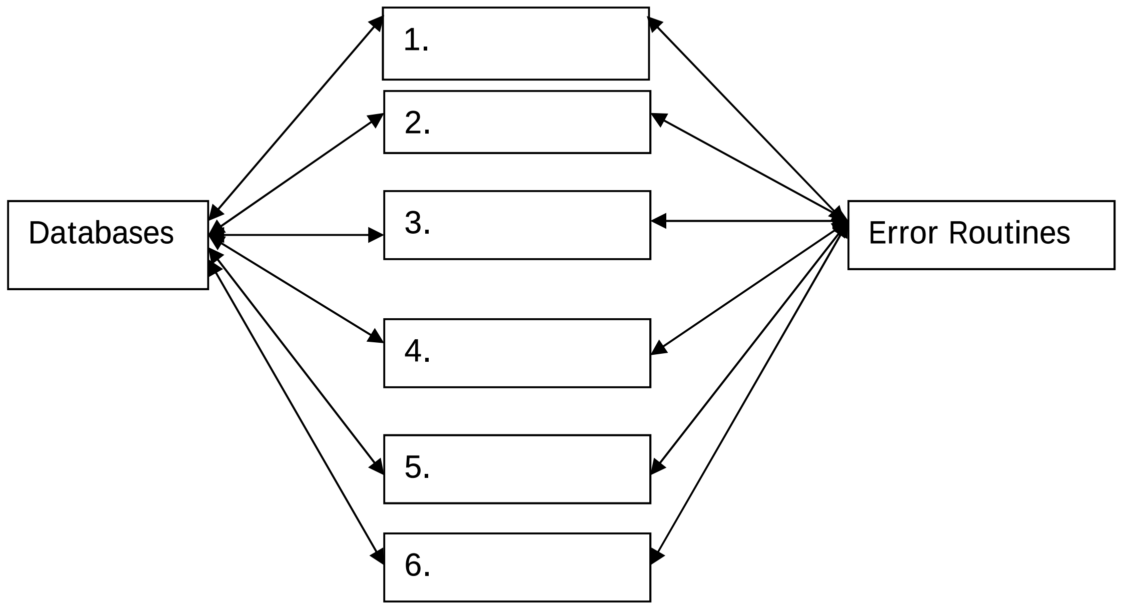
Incr

Data DC F'5'

Q 1 (b) Answer any **TWO** of the following (06)

i. List the four components of system programming. Write one function of two components.

ii. Observe the given diagram of compiler. Write the names of phases from 1 to 6.



Q 2 Answer any **TWO** of the following (16)

a) Draw the flowchart for Pass1 of a two pass assembler.

b) Write the contents of M.N.T. and M.D.T. for the following code :

```
Macro
A       &m_v, &i_v, &reg
Move    &reg, &m_v
Add     &reg, &i_v
Move    &reg, m_v
Mend
```

c) Draw the neat labeled diagram of compile and go loader. Describe its working.

Q 3Answer any **FOUR** of the following (16)

a) Draw the neat labeled diagram of intermediate phase of a compiler.

b) Describe the four tasks performed by Macro Processor.
c) Sort the following numbers in descending order using bucket sort :
   78, 354, 51, 278, 63, 89, 312, 12.
d) List the components of system software. State the functions of two components.
e) For the following sub expression, draw the table for intermediate code with optimization :

   Z = (A+B) * (C-D) + (A+B)

Q 4. A)Answer any **<u>THREE</u>**of the following (12)

a) Mention four notational shorthand for representing regular expression.
b) “Variable used before declaration leads to the problem of forward reference “Give solution to rectify the problem.
c) Give the specifications of database used in assembler design
d) List the steps for the binary search algorithm. List the best, worst and average case complexity.

Q 4. B) Answer any **<u>ONE</u>** of the following. (06)

a) Define parser. Draw the parse tree for the string ‘abccd’ using top down parser.
b) Describe token with respect to lexical analysis with a suitable example.

Q 5 Answer any **<u>TWO</u>** of the following (16)

a) Describe the designing of absolute loader with respect to its performances based upon
   1. Allocation
   2. Loading
   3. Relocation
   4. Linking
b) With the neat diagram describe the analysis and synthesis phase of general model of compiler
c) Draw neat labeled diagram of pass1 of a macro processor.

Q 6Answer any **<u>FOUR</u>** of the following (16)

a) Mention four functions of storage assignment phase of a compiler.
b) Write the necessity of overlays in linking loaders.
c) List the specification of data structures in direct linking loaders.
d) What is hashing? Write a suitable example of hashing.
e) Draw the neat labelled diagram of general loading scheme.