SRIMAAN

PG-TRB

COMPUTER SCIENCE

(NEW SYLLABUS 2025-2026)

POST GRADUATE ASSISTANTS (P.G ASST.)

UNIT-3- SYSTEM SOFTWARE & OPERATING SYSTEM

TRB-ASST.PROF. STUDY MATERIALS AVAILABLE

TAMIL/ ENGLISH / MATHS/PHYSICS /CHEMISTRY /COMMERCE /BIO-CHEMISTRY/ BOTANY / ZOOLOGY/ ECONOMICS /HISTORY / GEOGRAPHY/COMPUTER SCIENCE & APPLICATION / IT/ EEE / ECE/ GEOLOGY/ BUSINESS ADMINISTRATION /HRD / MICRO-BIOLOGY / ENVIRONMENTAL SCIENCE / EDUCATION AVAILABLE.

PG-TRB STUDY MATERIALS:–TAMIL/ENGLISH/ MATHEMATICS/PHYSICS CHEMISTRY/COMMERCE (T/M & E/M)/BOTANY (T/M & E/M)/ ZOOLOGY HISTORY (T/E)/ECONOMICS (T/E)/ GEOGRAPHY / COMPUTER SCIENCE PHYSICAL EDUCATION/ POLITICAL SCIENCE TO CONTACT +91 8072230063.

SRIMAAN COACHING CENTRE-TRICHY–PG-TRB-COMPUTER SCIENCE-UNIT-3: SYSTEM SOFTWARE AND OPERATING SYSTEM STUDY MATERIAL -TO CONTACT: +91 8072230063.

2025-26 SRIMAAN

TRB-POLYTECHNIC LECTURER MATERIALS:

MATHEMATICS / ENGLISH /PHYSICS / CHEMISTRY/COMPUTER SCIENCE/ IT / EEE / ECE / EIE/ ICE/ MECHANICAL/ CIVIL/ MOP AVAILABLE.

UG-TRB & SGT: ALL SUBJECT STUDY MATERIALS AVAILABLE.

SCERT/DIET/GTTI STUDY MATERIAL AVAILABLE.

DEO & BEO (T/M & E/M) STUDY MATERIALS AVAILABLE.

TRB-ASST.PROFESSOR STUDY MATERIAL AVAILABLE.

PG-TRB: COMPUTER INSTRUCTOR GRADE-1–FULL STUDY MATERIAL WITH QUESTION BANK AVAILABLE

TNPSC-ASSISTANT DIRECTOR OF CO-OPERATIVE AUDIT STUDY MATERIAL AVAILABLE.

TNEB-(ASSESSOR/AE/JA) MATERIALS WITH QUESTION BANK AVAILABLE

UG-TRB/PG-TRB /TRB-ASST.PROF./ DEO & BEO MATERIALS ARE SENDING THROUGH COURIER.

TO CONTACT +91

8072230063

PG-TRB STUDY MATERIALS:–TAMIL/ENGLISH/ MATHEMATICS/PHYSICS CHEMISTRY/COMMERCE (T/M & E/M)/BOTANY (T/M & E/M)/ ZOOLOGY HISTORY (T/E)/ECONOMICS (T/E)/ GEOGRAPHY / COMPUTER SCIENCE PHYSICAL EDUCATION/ POLITICAL SCIENCE TO CONTACT +91 8072230063.

System Software: Machine, Assembly and High-Level Languages; Compilers and Interpreters; Loading, Linking and Relocation; Macros, Debuggers.

System software

System software is computer software designed to operate the computer hardware and to provide a platform for running application software.

The most basic types of system software are:

- **The computer BIOS and device firmware, provides basic functionality to operate and control the hardware connected to or built into the computer.**
- **The operating system (prominent examples being Microsoft Windows, Mac OS X and Linux), which allows the parts of a computer to work together by performing tasks like transferring data between memory and disks or rendering output onto a display device. It also provides a platform to run high-level system software and application software.**
- **Utility software, which helps to analyze, configure, optimize and maintain the computer.**

compiler, linker or debugger used to designate software development tools

SOFTWARE

- Set of instructions given to the computer.
- We cannot touch and feel it.
- Developed by writing instructions in programming language.
- Operations of computer are controlled via this.
- If damaged or corrupted, back up copy can be installed again.
- Eg:- Antivirus, Microsoft Office Tools.

HARDWARE

- Physical parts of a computer.
- We can touch and feel it.
- Constructed using physical components.
- Operates under control of software.
- If damaged, can be replaced.
- Eg:- Ke

SOFTWARE vs HARDWARE

SOFTWARE	HARDWARE
1. Collection of instructions that tells computer what to do	1. Physical elements of computer
2. Divided in to a. System Software b. Application Software	2. Categories a. Input Devices. b. Output Devices
c. Utility Software	c. Storage Devices
3. Should be installed in to computer	3. Once software is loaded these can be used.
4. Prone to viruses 5. If damaged/ corrupted reinstallation is possible	4. No virus attacks 5. If damaged, can be replaced.
Eg:- Microsoft Office, Adobe	Eg:- Mouse, Monitor, Keyboard

TYPES OF SOFTWARE

1. System Software:

- Contains collection of programs that support operation of computer.
- Helps to run computer hardware and computer system.
- Handles running of computer hardware.
- These are of different types”
  a) Operating System
  b) Language Translators
     i. Compiler
     ii. Assembler
     iii. Interpreter
     iv. Macro Processor
  c) Loader
  d) Linker
  e) Debugger
  f) Text Editor

2. Application Software:
    - It allows end users to accomplish one or more specific tasks.
    - Focus on application or problem to be solved.

Operating System

- Acts as interface between user and system.
- Provide user friendly interface.
- Functions:
    a) Process Management
    b) Memory Management
    c) Resource Management
    d) I/O Operations
    e) Data Management
    f) Provide Security for job.

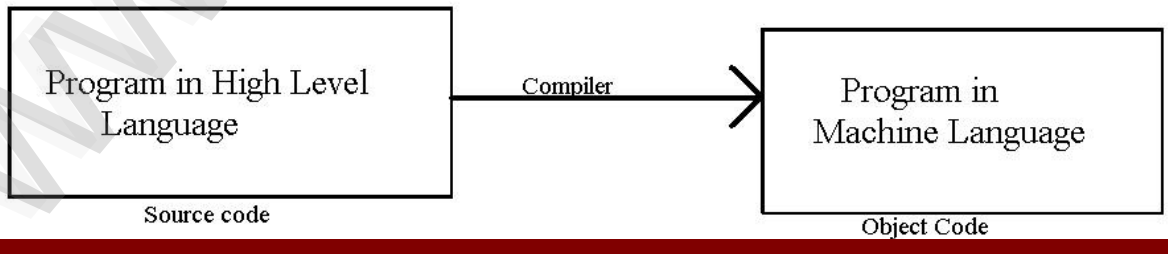
High-Level Languages

Language Translators

- Program that takes input program in one language and produces an output in another language.

I. Compilers

- Translates program in high level language in to machine level language.
- Conversion or translation is taking place by taking program as whole.
- Bridges the semantic gap between language domain and execution domain.
- Perform syntax analysis, semantics analysis and intermediate code generation.



II. Interpreters

- Translates statement of high level language in to machine level language by taking the program line by line.
- Interpretation cycle includes:

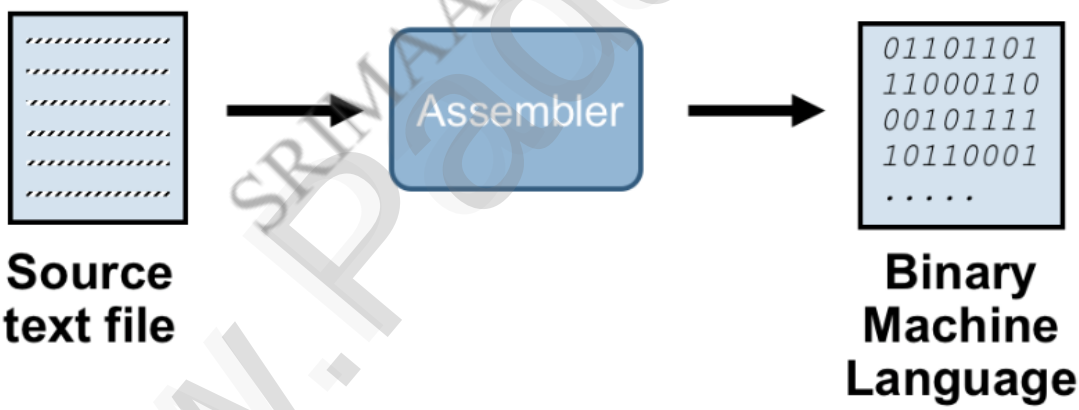
  i) Fetch the statement.

  ii) Analyze the statement and determine its meaning.

  iii) Execute the meaning of statement.

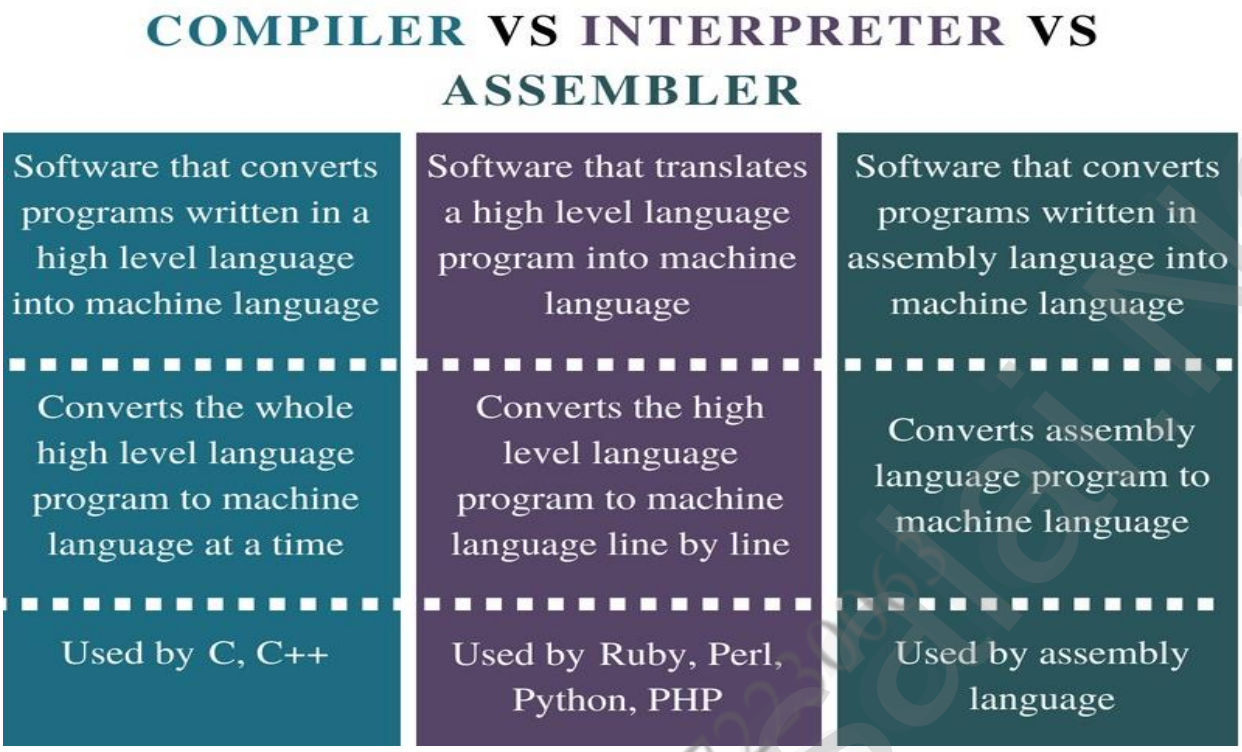
III. Assemblers

- Programmers found it difficult to read or write programs in machine language, so for convenience they used mnemonic symbols for each instruction which is translated to machine language.
- Assemblers translate assembly language to machine language.
- Translate mnemonic code to machine language equivalents.
- Assign machine address to symbol table.



Working:

- Find the required information to perform task.
- Analyze and design suitable data structures to hold and manipulate information.
- Find the process or steps needed to gather information and maintain it.
- Determine processing step required to execute each identified task.



Linker

- Process of collecting and combining various pieces of code and data in to single file that can be loaded in to memory and executed.
- Linking performed a compile time, when source code is translated to machine code, at load time, when program is loaded in to memory and executed by loader and at run time by application programs.

Types:

a) Linking Loader: Performs all linking and relocation operations directly in to main memory for execution.

b) Linkage Editor: Produce a linked version of program called as load module or executable image. This load module is written in to file or library for later execution.

c) Dynamic Linker: This linking postpones the linking function until execution time. Also called as dynamic loading.

Loader

- Utility of an operating system.
- Copies program from a storage device to computer's main memory.
- They can replace virtual address with real address.
- They are invisible to user.

Debugger

- An Interactive debugging system provides programmers with facilities that aid in testing and debugging of programs.
- Debugging means locating bugs or faults in program.
- Helps in fixing error.
- Determination of exact nature and location of error in the program.

Device Driver

- It is a software module which manages the communication and control of specific I/O device on type of device.
- Convert logical requests from the user in to specific commands directed to device itself.

Macro Processor

- Macro is the unit of specification of program generation through expansion.
- Macros are special code fragments that are defined once in the program and used by calling them from various places within the program.
- Macro processor is a program that copies stream of text from one place to another, making a systematic set of replacements as it does so.
- They are often embedded in other programs such as assemblers and compilers.

Application software

In contrast to system software, software that allows users to do things like create text documents, play games, listen to music, or surf the web is called application software. Thus **Application software**, is computer software designed to help the user to perform specific tasks.

Examples include enterprise software, accounting software, office suites, graphics software and media players.

The system software serves the application, which in turn serves the user.

ASSEMBLERS

Assembler is system software which is used to convert an assembly language program to its equivalent object code. The input to the assembler is a source code written in assembly language (using mnemonics) and the output is the object code. The design of an assembler depends upon the machine architecture as the language used is mnemonic language.

BASIC ASSEMBLER FUNCTIONS: The basic assembler functions are:

1) Translating mnemonic language code to its equivalent object code.
2) Assigning machine addresses to symbolic labels.

The design of assembler:

The design of assembler can be to perform the following:

1) Scanning (tokenizing)
2) Parsing (validating the instructions)
3) Creating the symbol table
4) Resolving the forward references
5) Converting into the machine language

It also includes-

□□□Convert mnemonic operation codes to their machine language equivalents

- Convert symbolic operands to their equivalent machine addresses
- Decide the proper instruction format Convert the data constants to internal machine representations
- Write the object program and the assembly listing

assembler directives

these do not generate the object code but directs the assembler to perform certain operation. These directives are:

START: Specify name & starting address.
END: End of the program, specify the first execution instruction. BYTE, WORD, RESB, RESW
End of record: a null char(00) End of file: a zero length record

The assembler design can be done in two ways:

1. Single pass assembler
2. Multi-pass assembler

Single-pass Assembler: In this case the whole process of scanning, parsing, and object code conversion is done in single pass. The only problem with this method is resolving forward reference.

Two-Pass assembler: (multi-pass assembler) It resolves the forward references. The process of the multi-pass assembler can be as follows:

Pass-1

a) Assign addresses to all the statements
b) Save the addresses assigned to all labels to be used in *Pass-2*
c) Perform some processing of assembler directives such as RESW, RESB to find the length of data areas for assigning the address values.
d) Defines the symbols in the symbol table(generate the symbol table)

Pass-2

a) Assemble the instructions (translating operation codes and looking up addresses).
b) Generate data values defined by BYTE, WORD etc.
c) Perform the processing of the assembler directives not done during *pass-1*.
d) Write the object program and assembler listing.

Loading, Linking and Relocation

LOADERS AND LINKERS

The Source Program written in assembly language or high level language will be converted to object program, which is in the machine language form for execution. This conversion either from assembler or from compiler, contains translated instructions and data values from the source program, or specifies addresses in primary memory where these items are to be loaded for execution. This contains the following three processes, and they are,

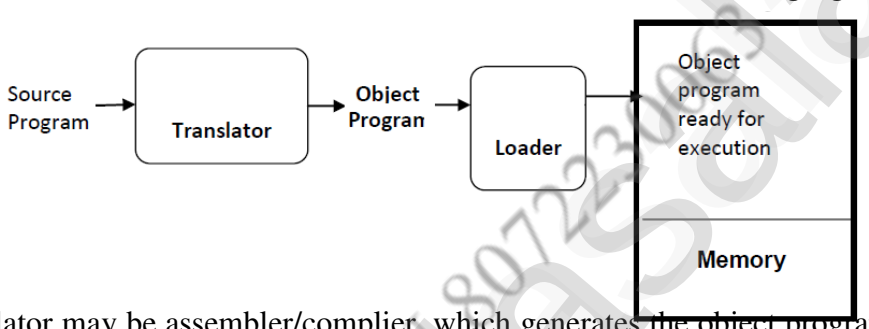
Loading - which allocates memory location and brings the object program into memory for execution - (Loader)

Linking- which combines two or more separate object programs and supplies the information needed to allow references between them - (Linker)

Relocation - which modifies the object program so that it can be loaded at an address different from the location originally specified - (Linking Loader)

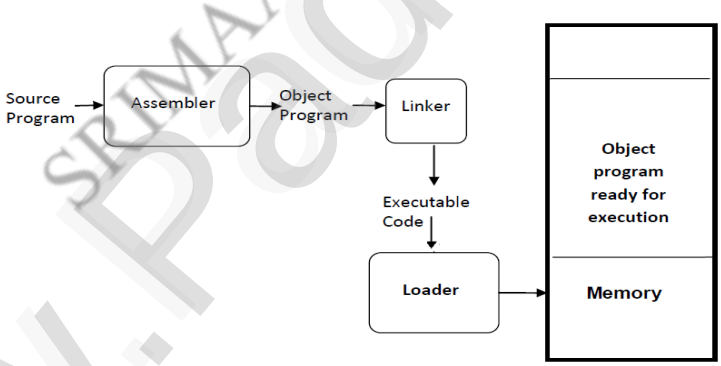
Basic Loader Functions

A loader is a system program that performs the loading function. It brings object program into memory and starts its execution. The role of loader is as shown in the follwing figure-



In figure translator may be assembler/complier, which generates the object program and later loaded to the memory by the loader for execution.

Role of both loader and linker:



Type of Loaders

The different types of loaders are, absolute loader, bootstrap loader, relocating loader (relative loader), and, direct linking loader.

1) **Absolute Loader** The object code is loaded to specified locations in the memory. At the end the loader jumps to the specified address to begin execution of the loaded program.

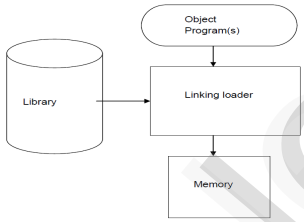
2) **A Bootstrap Loader** When a computer is first turned on or restarted, a special type of absolute loader, called bootstrap loader is executed. This bootstrap loads the first program to be run by the computer -- usually an operating system.

3) **Relocating loader** The object program is loaded into memory wherever there is room for it. The actual starting address of the object program is not known until load time.Loaders that allow for program relocation are called relocating loaders or relative loaders.

4) **Direct Linking Loader** Linking Loader uses two-passes logic.

Pass 1: Assign addresses to all external symbols

Pass 2: Perform the actual loading, relocation, and linking.



Linking Options

There are some common alternatives for organizing the loading functions, including relocation and linking are-

Linking Loaders – Perform all linking and relocation at load time.

The above diagram shows the processing of an object program using Linking Loader. The source program is first assembled or compiled, producing an object program. A linking loader performs all linking and loading operations, and loads the program into memory for execution.

Linkage editors - A linkage editor produces a load module or an executable image – which is written to a file or library for later execution. Some useful functions of Linkage editor are, an absolute object program can be created, if starting address is already known.

Dynamic linking, in which linking function is performed at execution time. A subroutine is loaded and linked to the rest of the program when it is first called – usually called dynamic linking, dynamic loading or load on call. The advantages of dynamic linking are, it allow several executing programs to share one copy of a subroutine or library.

Relocation

It modifies the object program so that it can be loaded at an address different from the location originally specified .**Two different techniques used for relocation.**

1) Modification record method
2) Relocation bit method.

Relocation bit method. If the relocation bit corresponding to a word of object code is set to 1, the program‟ s starting address is to be added to this word when the program is relocated. Bit value 0 indicates no modification is required.

bit mask. The relocation bits are gathered together following the length indicator in each text record and which is called as bit mask. For e.g. the bit mask FFC (111111111100) specifies that the first 10 words of object code are to be modified during relocation.

MACRO PROCESSORS

A *Macro* represents a group of statements in the source programming language that allows the programmer to write shorthand version of a program. The macro processor replaces each macro instruction with the corresponding group of source language statements (*expanding*). The design of a macro processor generally is *machine independent.*

Macro processor is system software that replaces each macroinstruction with the corresponding group of source language statements. This is also called as expanding of macros.

Macro expansion statements give the name of the macro instruction being invoked and the arguments to be used in expanding the macros. These statements are also known as macro call.

Directives used in macro definition is MACRO - it identifies the beginning of the macro definition and MEND - it marks the end of the macro definition

Data structures used in macro processor:

DEFTAB –it contains a macro prototype and the statements that make up the macro body. NAMTAB – it is used to store the macro names and it contains two pointers for each macro instruction which indicate the starting and end location of macro definition
ARGTAB – it is used to store the arguments during the expansion of macro invocations.

Conditional macro expansion If the macro is expanded depends upon some conditions in macro definition then it is called as conditional macro expansion. Sfollowing statements used for conditional macro expansion.
IF-ELSE-ENDIF statement
WHILE-ENDW statement

For example: Consider the macro definition
#define DISPLAY (EXPR) Printf ("EXPR = %d\n", EXPR)
Expand the macro instruction DISPLAY (ANS) by Printf ("EXPR = %d\n", ANS)

Nested macro call: The statement, in which a macro calls on another macro, is called nested macro all. In the nested macro call, the call is done by outer macro and the macro called is the inner macro.
Two passes of macro: Pass1: processing of definitions. Pass 2:actual-macro expansion.

Line by line processor: This macro processor reads the source program statements, process the statements and then the output lines are passed to the language translators as they are generated, instead of being written in an expanded file. Its **advantages are-** It avoids the extra pass over the source program during assembling
and it may use some of the utility that can be used by language translators so that can be loaded once.

General-purpose macro processors: The macro processors that are not dependent on any particular programming language, but can be used with a variety of different languages are known as general purpose macro processors. Eg. The ELENA macro processor. Its **advantages are -** programmer does not need to learn about a macro facility for each compiler and overall saving in software development cost and maintenance cost.

Execution of nested macro: The execution of nested macro call follows the LIFO rule. In case of nested macro calls the expansion of the latest macro call is completed first.

The tasks involved in macro expansion:
- identify the macro calls in the program
- the values of formal parameters are identified
- maintain the values of expansion time variables declared in a macro
- expansion time control flow is organized
- determining the values of sequencing symbols
- expansion of a model statement is performed

Compiler

A **compiler** is a computer program (or set of programs) that transforms source code written in a programming language (the *source language*) into another computer language (the *target language*, often having a binary form known as *object code*).

The name "compiler" is primarily used for programs that translate source code from a high-level programming language to a lower level language (e.g., assembly language or machine code).

If the compiled program can run on a computer whose CPU or operating system is different from the one on which the compiler runs, the compiler is known as a cross-compiler.

A program that translates from a low level language to a higher level one is a *decompiler*.

A compiler operations: lexical analysis, preprocessing, parsing, semantic analysis (Syntax-directed translation), code generation, and code optimization.

machine-independent Before the development of FORTRAN (FORmula TRANslator), the first higher-level language, in the 1950s, machine-dependent assembly language was widely used. With the advance of high-level programming languages soon followed after FORTRAN, such as COBOL, C, BASIC, programmers can write machine-independent source programs.

The structure of a compiler
A compiler requires-
1) determining the correctness of the syntax of programs,
2) generating correct and efficient object code,
3) run-time organization, and
4) formatting output according to assembler and/or linker conventions.
A compiler consists of three main parts: the frontend, the middle-end, and the backend.
The **front end** checks whether the program is correctly written in terms of the programming language syntax and semantics.
The **middle end** is where optimization takes place.
The **back end** is responsible for translating the IR from the middle-end into assembly code.

Compiler output
One classification of compilers is by the platform on which their generated code executes. This is known as the *target platform.*
A *native* or *hosted* compiler is one which output is intended to directly run on the same type of computer and operating system that the compiler itself runs on.
The output of a cross compiler is designed to run on a different platform. Cross compilers are often used when developing software for embedded systems that are not intended to support a software development environment.
The output of a compiler that produces code for a virtual machine (VM) may or may not be executed on the same platform as the compiler that produced it. For this reason such compilers are not usually classified as native or cross compilers.

Cross compiler

A **cross compiler** is a compiler capable of creating executable code for a platform other than the one on which the compiler is run. Cross compiler tools are used to generate executables for embedded system or multiple platforms. It is used to compile for a platform upon which it is not feasible to do the compiling, like microcontrollers that don't support an operating system. It has become more common to use this tool for paravirtualization where a system may have one or more platforms in use.
Uses of cross compilers

The fundamental use of a cross compiler is to separate the build environment from target environment. This is useful in a number of situations:

- Embedded computers where a device has extremely limited resources. For example, a microwave oven.

- Compiling for multiple machines. to support several different versions of an operating system or to support several different operating systems.
- Compiling on a server farm. involves many compile operations can be executed across any machine
- Bootstrapping to a new platform.
- Compiling native code for emulators

Canadian Cross: The Canadian Cross is a technique for building cross compilers for other machines. Given three machines A, B, and C, one uses machine A to build a cross compiler that runs on machine B to create executables for machine C.

GCC and cross compilation: GCC, a free software collection of compilers, can be set up to cross compile. It supports many platforms and languages.Cross compiling GCC requires that a portion of the *target platform*'s C standard library be available on the *host platform*.

Manx Aztec C cross compilers: Manx's Aztec C86, their native mode 8086 MS DOS compiler, was also a cross compiler. It created binary executables for then-legacy operating systems for the 16 bit 8086 family of processors.

Microsoft C cross compilers: The first Microsoft C Compilers were made by the same company who made Lattice C and were rebranded by Microsoft as their own, until MSC 4 was released, which was the first version that Microsoft produced themselves.

.NET and beyond: The .NET Framework runtime and CLR provide a mapping layer to the core routines for the processor and the devices on the target computer. The command-line C compiler in Visual Studio will compile native code for a variety of processors and can be used to build the core routines themselves.

TO BE CONTINUED.....

Basics of Operating Systems: Operating System Structure, Operations and Services; System Calls, Operating-System Design and Implementation; System Boot.

Basics of Operating Systems:

OPERATING SYSTEM

What is an Operating System?

A program that acts as an intermediary between a user of a computer and the computer hardware Operating system goals:

- Execute user programs and make solving user problems easier
- Make the computer system convenient to use
- Use the computer hardware in an efficient manner

Computer System Structure

Computer system can be divided into four components

- Hardware – provides basic computing resources
  CPU, memory, I/O devices
- Operating system
  Controls and coordinates use of hardware among various applications and users
- Application programs – define the ways in which the system resources are used to solve the computing problems of the users
  Word processors, compilers, web browsers, database systems, video games

SRIMAAN COACHING CENTRE-TRICHY- TET/PG-TRB / UG-TRB BEO/ DEO/TRB-POLY/ASST.PROF/TN-MAWS /TNEB /SCERT STUDY MATERIALS AVAILABLE- CONTACT:8072230063.

2025-26 SRIMAAN

TN-MAWS-MUNICIPAL ADMINISTRATION & WATER SUPPLY DEPARTMENT-(DEGREE & DIPLOMA) STUDY MATERIALS AVAILABLE.

TRB-ASSISTANT PROFESSORS IN GOVERNMENT ARTS AND SCIENCE COLLEGES & COLLEGES OF EDUCATION STUDY MATERIALS AVAILABLE.

UG-TRB MATERIALS

GRADUATE TEACHERS / BLOCK RESOURCE TEACHER EDUCATORS (BRTE) & SGT

- UG TRB: TAMIL MATERIAL WITH QUESTION BANK.
- UG TRB: ENGLISH STUDY MATERIAL +Q. BANK.
- UG-TRB: MATHEMATICS MATERIAL WITH Q. BANK (E/M)
- UG TRB: PHYSICS MATERIAL WITH QUESTION BANK (E/M)
- UG TRB: CHEMISTRY MATERIAL + QUESTION BANK (E/M)
- UG TRB: HISTORY MATERIAL + Q.BANK (E/M)
- UG TRB: ZOOLOGY MATERIAL + QUESTION BANK (E/M)
- UG TRB: BOTANY MATERIAL +QUESTION BANK (T/M& E/M)
- UG TRB: GEOGRAPHY STUDY MATERIAL (E/M)

SCERT/DIET/GTTI (LECTURER) STUDY MATERIAL AVAILABLE.

TNPSC-(CESE)-JSO STUDY MATERIAL AVAILABLE.

TANGEDCO (TNEB)-(T/M & E/M)

ASSESSOR/ASSISTANT ENGINEER (A.E)/JUNIOR ASSISTANT (ACCOUNTS)

SRIMAAN COACHING CENTRE-TRICHY- TET/PG-TRB / UG-TRB BEO/ DEO/TRB-POLY/ASST.PROF/TN-MAWS /TNEB /SCERT STUDY MATERIALS AVAILABLE- CONTACT:8072230063.

2025-26 SRIMAAN

PG-TRB MATERIALS

PG-TRB: COMPUTER SCIENCE (NEW SYLLABUS-2025-2026) STUDY MATERIAL WITH Q.BANK AVAILABLE

- PG TRB: TAMIL STUDY MATERIAL +QUESTION BANK (T/M)
- PG TRB: ENGLISH MATERIAL WITH QUESTION BANK.
- PG-TRB: MATHEMATICS MATERIAL WITH Q.BANK (E/M)
- PG TRB: PHYSICS MATERIAL WITH QUESTION BANK (E/M)
- PG TRB: CHEMISTRY MATERIAL + QUESTION BANK (E/M)
- PG TRB: COMMERCE MATERIAL WITH Q.BANK (T/M)&(E/M)
- PG TRB:ECONOMICS MATERIAL+Q. BANK (T/M & E/M)
- PG TRB: HISTORY MATERIAL + Q. BANK (T/M & E/M)
- PG TRB: ZOOLOGY MATERIAL + QUESTION BANK (E/M)
- PG TRB: BOTANY MATERIAL +QUESTION BANK (T/M& E/M)
- PG TRB: GEOGRAPHY STUDY MATERIAL (E/M)

TNPSC-DEO (District Educational Officer(Group – I C Services)
(TAMIL & ENGLISH MEDIUM) STUDY MATERIAL AVAILABLE.

TRB-BEO (Block Educational Officer)
(TAMIL & ENGLISH MEDIUM) STUDY MATERIAL AVAILABLE.

TRB-POLYTECHNIC LECTURER-(NEW SYLLABUS) STUDY MATERIALS AVAILABLE

- MATHEMATICS STUDY MATERIAL with Question Bank.

SRIMAAN COACHING CENTRE-TRICHY- TET/PG-TRB / UG-TRB BEO/ DEO/TRB-POLY/ASST.PROF/TN-MAWS /TNEB /SCERT STUDY MATERIALS AVAILABLE- CONTACT:8072230063.

2025-26
SRIMAAN

- ENGLISH STUDY MATERIAL with Question Bank.
- PHYSICS STUDY MATERIAL with Question Bank.
- CHEMISTRY STUDY MATERIAL with Question Bank.
- MODERN OFFICE PRACTICE STUDY MATERIAL with Q.B.
- COMPUTER SCIENCE STUDY MATERIAL with Question Bank.
- INFORMATION TECHNOLOGY STUDY MATERIAL with Q.Bank.
- ECE STUDY MATERIAL with Question Bank.
- EEE STUDY MATERIAL With Question Bank.
- MECHANICAL STUDY MATERIAL With Question Bank.
- CIVIL STUDY MATERIAL with Question Bank.
- EIE STUDY MATERIAL with Question Bank.
- ICE STUDY MATERIAL with Question Bank.

EMAIL ID:srimaanacademy@gmail.com

10% Discount for all materials. Materials are sending through

COURIER.

Youtube link: http://www.youtube.com/@srimaantetpgtrbcoachingcen9477

TO CONTACT

+91 80722 30063

SRIMAAN PUBLICATIONS