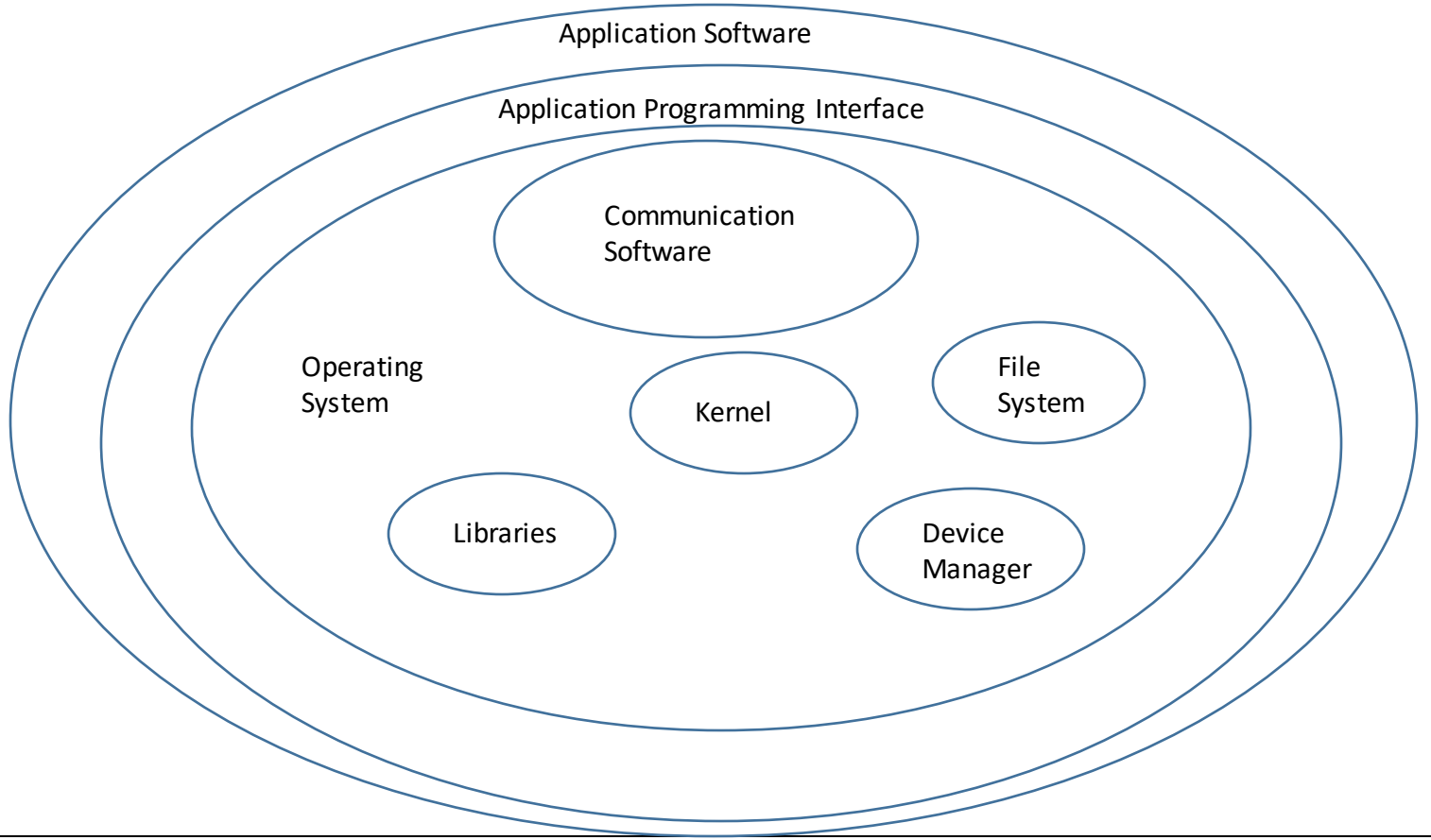
Software Architecture

-Software Architecture of an Embedded System



Software Architecture

- Operating System consist of
  - Kernel
  - Device Manager
  - Communication Protocol Software
  - Libraries
  - File System

Software Architecture

- Operating System consist of
  - Kernel
    - It is the central module of operating system.
    - It manages the tasks to achieve the desired performance of the embedded system. The two important elements needed to manage the tasks are,
      - Task Scheduling & Inter Task Communication.
    - Kernel also provides the following services:
      - Memory Management
      - Interrupt Handling
      - Device Management

Software Architecture

- Operating System consist of
  - Device Manager
    - It provides the necessary interface between the application and the hardware.
    - It manages the I/O devices through interrupts and device drivers.
  - Communication Protocol Software
    - If the embedded system has communication interfaces (USB, Ethernet), the upper layer protocol such as TCP/IP need to be interfaced with OS , to get network enabled.
  - Libraries
    - The operating system may have some library file in object code, which can be used through the API calls.
  - File System
    - OS contains the code to support different file structure. File type refers to the ability of the operating system to distinguish between different types of file such as text files, source files and binary files etc.

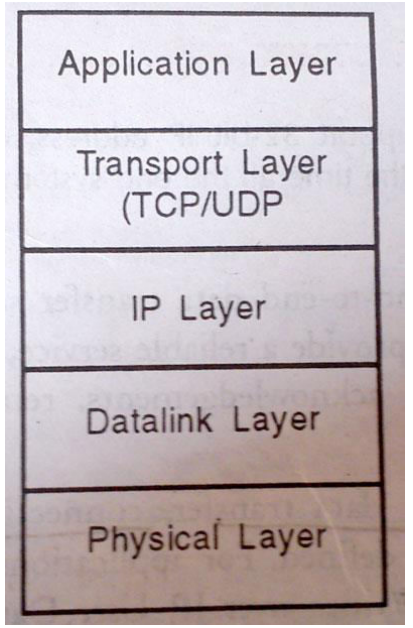
Communication Software

- Most of the embedded systems need a communication interface to interact with the external world.
- Communication software needs to be integrated with the firmware.
- To make Ethernet interface, TCP/IP protocol stack has to run on it.

TCP/IP Protocol Suite

- The TCP/IP protocol suite led the most important concept of packet switching.
- In Packet switching, the data to be transmitted is divided into small packets and each packet is transmitted from the source to the destination.
- During the transmission some packets may be lost and packets may not be received in sequence.
- Using TCP/IP protocols take care of the packets through several process.

- It consists of 5 layers
  - Physical Layer
  - Data Link Layer (referred also as Network Layer)
  - Internet Protocol (IP) layer
  - Transport layer (TCP layer and UDP layer)
  - Application layer



- **Physical Layer**
  - This layer defines the characteristics of the transmission such as data rate and signal encoding scheme
  - Establishment and termination of a connection to a communication medium.
  - Defines the relationship between a device and a physical medium.
  - Includes a layout of pins, voltage, cable specifications etc.

- **Data Link Layer:** This layer defines the protocols to manage the links,establishing a link, transferring the data received from the upper layers, and disconnecting the link.

- Internet Protocol (IP) layer
  - The two important function of this layer are addressing and routing.
  - IP layer software runs on every end system and router connected the internet.
  - The presently running IP layer software are IPV4(32 bits), IPv6(128 bits).

- Transport Layer
  - The TCP/IP protocol used in Transport Layer are listed below.
    - Transmission Control Protocol (TCP)
      - Ensure that the data is delivered to application layer without any error.
      - Checks whether packets received are in sequence or not. If they are not in sequence, they have to be arranged in sequence.
      - Check received packets contains error or not. If packets are received in error, TCP layer has to ask for retransmission.
      - Flow control mechanism informs the other system not to send any more packets till further information(Controls the mismatch in speeds).
      - Connection oriented protocol, 3 way handshake is used to establish connection.

- Transport Layer
  - The TCP/IP protocol used in Transport Layer are listed below.
    - User Datagram Protocol (UDP)
      - Although, TCP provides reliable service by taking care of error control and flow control. Processing required for the TCP layer is very high. Hence, it is called ‘heavy-weight’ protocol.
      - In some real time applications such as video communication and network management, such high requirement processing may create problems. So another transport protocol is used, i.e User Datagram Protocol

- Transport Layer
  - The TCP/IP protocol used in Transport Layer are listed below.
    - User Datagram Protocol (UDP)
      - It sends the packets to the destination one after the other, without bothering whether they are being received correctly or not.
      - Connectionless protocol, Used in multicast communication.

- TCP/IP Protocol Suite
  - Application Layer
    - The application layer enables the user, whether human or software, to access the network.
    - It Provides user interfaces and support for services such as electronic mail, remote file access and transfer.
    - Applications on that layer (E-mail clients, web browsers, Chats, etc.) – top-stack applications (As people are on the top of the stack).
    - Applications provide people with a way to create message.
    - Application layer services establish an interface to the network.
    - Protocols provide the rules and formats that govern how data is treated.

- TCP/IP Protocol Suite
  - Application Layer

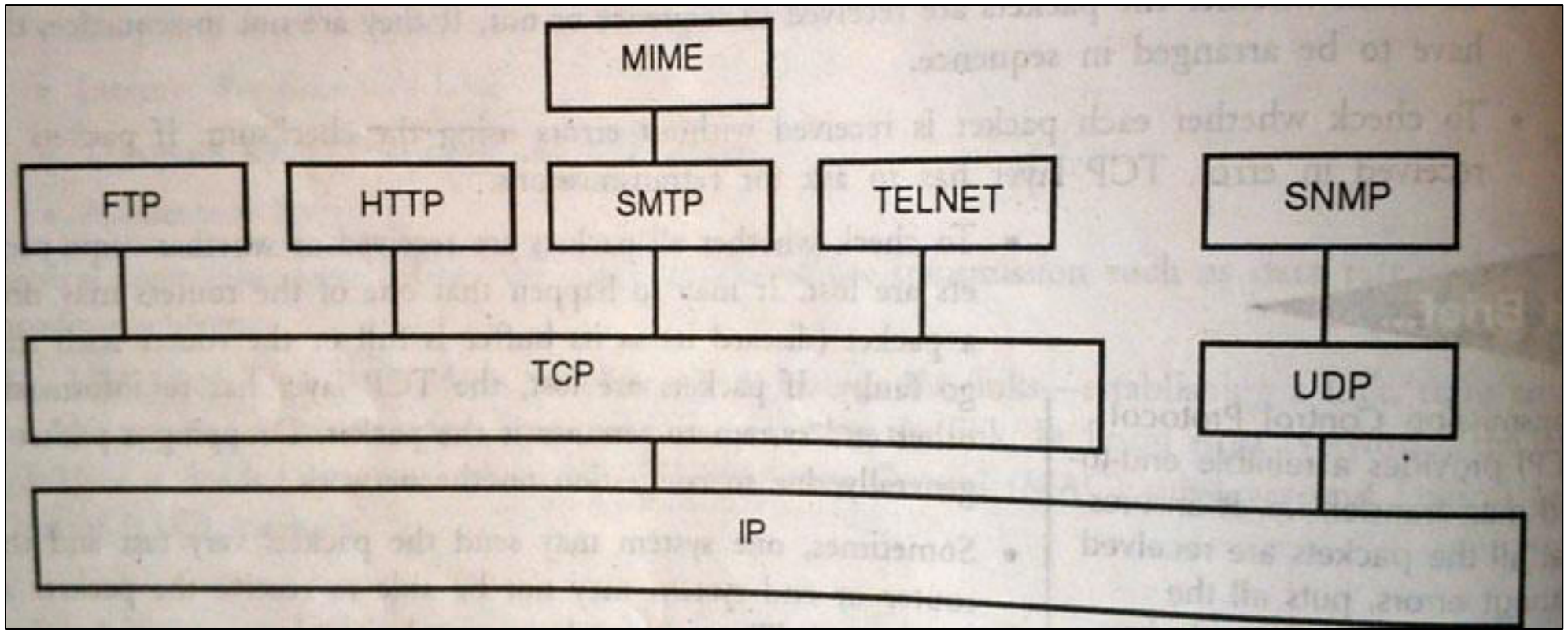


Fig : Application Layer Protocols in TCP/IP protocol stack

- TCP/IP Protocol Suite
  - Application Layer
    - The various application layer protocols are listed below.
      - Simple Mail Transfer Protocol (**SMTP**), for electronic mail containing ASCII Text.
      - Multimedia Internet Mail Extension (**MIME**), for electronic mail with multi-media content.
      - File Transfer Protocol **(FTP**) for file transfer.
      - **TELNET** for remote login
      - Hyper Text Transfer Protocol (**HTTP**) for World Wide Web Service.
      - Simple Network Management Protocol (**SNMP**) stack is used for network management. It runs above the UDP layer.

Process of Generating Executable Image

- Application development on desktop computers is called **native development** as the execution are done on the same hardware platform.

- Embedded software cannot be developed directly on the embedded system. Initially, the development is done on a desktop computer and then the software is transferred to the embedded system. This is known as **cross-platform development.**

- The procedure for creating and executing an applications in an embedded system is different for the following reasons

  - There is a distinction between the operating system and the application software, whereas in an embedded system(generally) everything is a single piece of code.

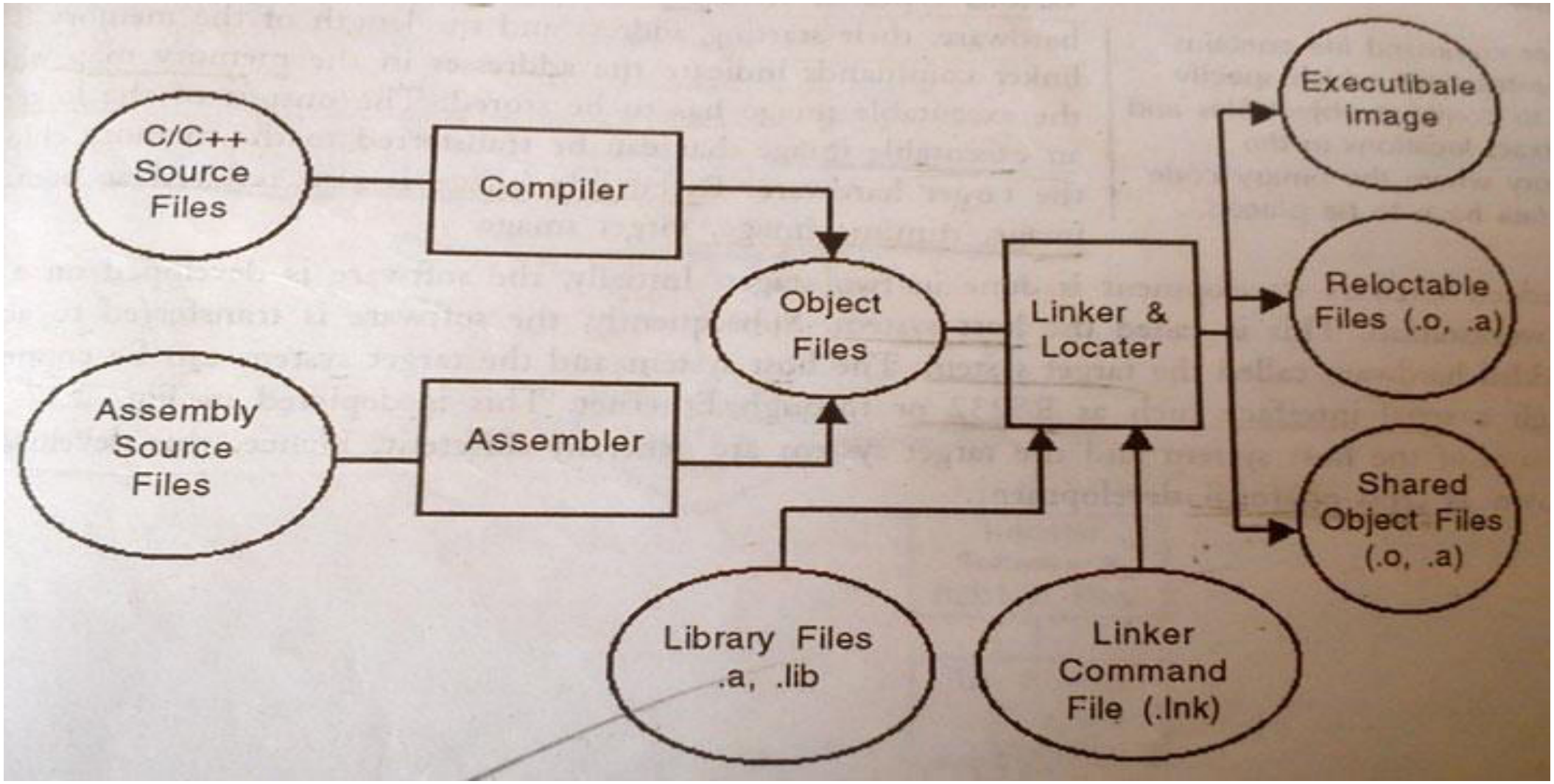
- In an embedded system, there is only one application that needs to run continuously. Multiple applications running in embedded system is very rare.

- On a desktop computer, in which part of the memory the application is loaded is not that important; but in embedded systems you need to decide where the code will reside so that processor executes the instructions from that memory location.

- Desktop computers use virtual memory. In a multi-tasking system, when a new process has to be executed, the presently running process is transferred to the virtual memory (which can be on the hard disk). In embedded systems, the secondary storage is not available.

- In case of embedded system, communication software and application software have to be converted into a single executable image and transferred to the memory of the embedded system.

Process of Creating an Executable Image.



Process of Creating an Executable Image.

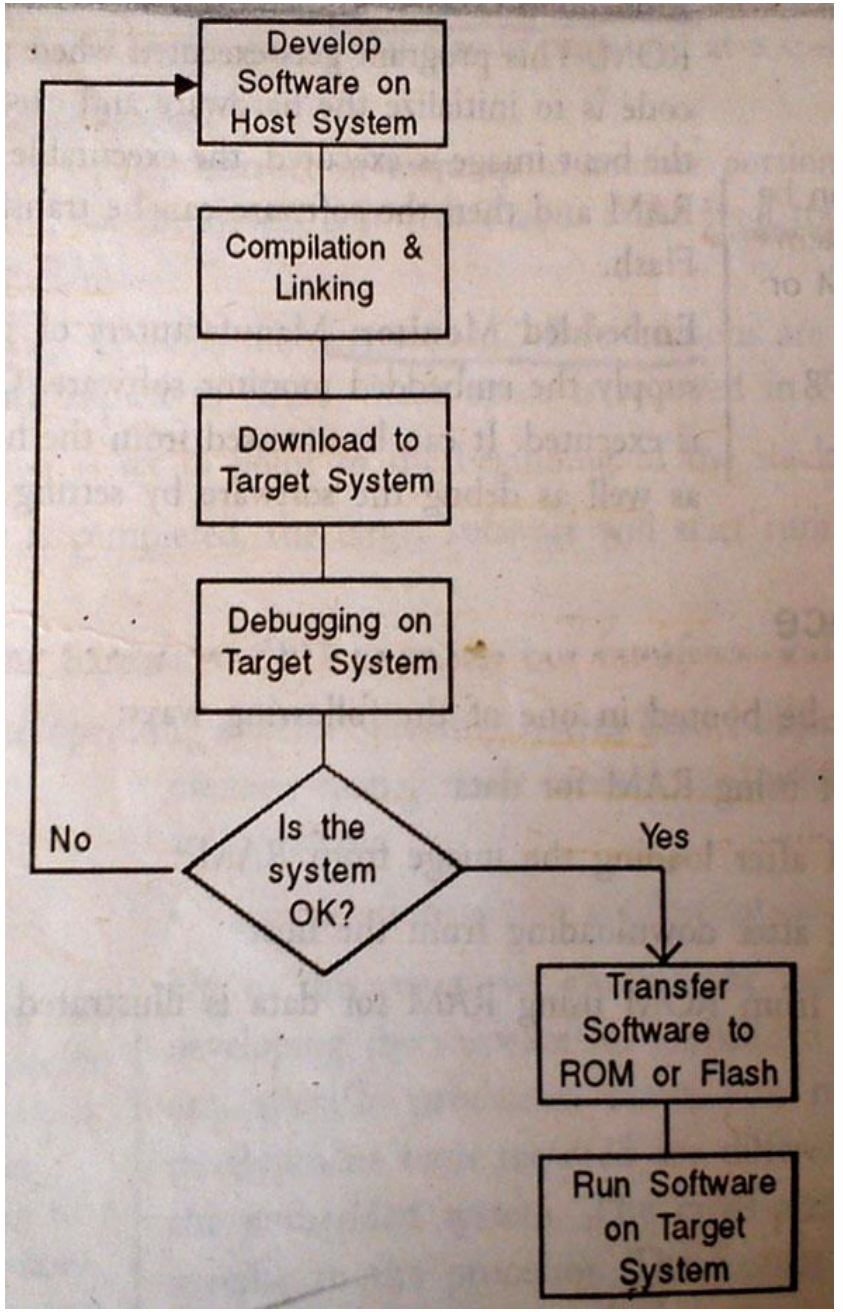
- The source files, written in C or C++ are converted into object files using the compiler.
- The source files written in the assembly language of the target processor are converted into object files using the assembler.
- Each object file contains the binary code (instructions) and program data.
- Each object file created in the above process will have the following information:
  - Name of the source file.
  - Size of the file, size of the binary instructions and size of the data
  - Processor-specific binary instructions and data
  - Symbol table that contains details of variables, their data types and addresses.
  - Debugging information if the compilation is done with the debug option.

- **Generating Executable Image**
  - The format for this object file is called the Object File Format. The format has been standardized into two standards
    - Common Object File Format (COFF)
    - Executable and Linking Format (ELF)
  - The standard is maintained to ensure that the object files generated by different compilers can be combined together.
  - Linker combines the various object files including the library files used in the program and creates an executable image, or a single relocatable object file or a shared library file.
  - Linker command file contains the instruction which specify how to combine object files and the exact locations in the memory where the binary code and data have to be placed.
  - The output of the linker is an executable image that can be transferred to the memory chip of the target hardware.
  - Executable image is also referred as bootable image, runtime image, target image.

- Embedded software development is done in two stages.
  - First software is developed on a PC or a workstation. This is called the **host system**.
  - After that software is transferred to actual embedded hardware called the **target system**.
  - The host system and the target system can be connected through a serial interface such as RS232 or through Ethernet or USB.
- The processors of the host system and the target system are generally different. Hence, this development is known as **cross-platform development**.

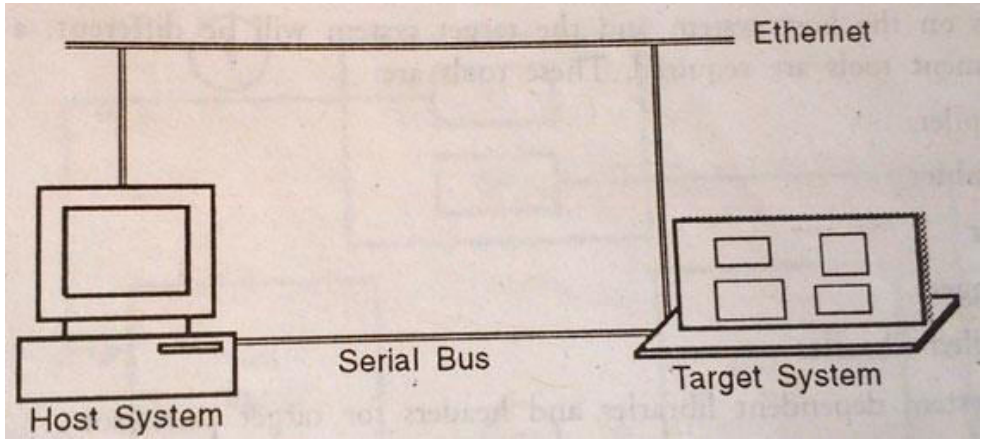
Cross-Platform Development

- The Process of cross-platform development is shown in the flowchart.
- The source code is written on the host system, compiled and linked using cross-platform development tools and then downloaded onto the target and tested.
- If the software is not working as per requirements, it can be debugged on the target itself.
- After ensuring everything is OK, the executable image is transferred to ROM or Flash Memory. Then Embedded system can run on its own.



- As the processors on the host system and the target system will be different, a number of cross-platform development tools are required. These tools are
  - Cross-compiler
  - Cross-assembler
  - Cross-linker
  - Cross-debugger
  - Cross-compiled libraries
  - Operating system dependent libraries and headers for target processor.

- The executable image can be transferred to the target hardware by one of the following mechanisms:
  - Programming the EEPROM or Flash
  - Downloading the image through a communication interface which requires a file transfer utility and an embedded loader or an embedded monitor on the embedded system.
  - Downloading through JTAG port.



An embedded loader or an embedded monitor is used to do the hardware initialization and run the initial bootup code.

- **Embedded Loader**: Embedded loader is a program that resides in the ROM. This program gets executed when power is switched on. This code is to initializes the hardware and execute the boot image. After the boot image is executed, the executable image is transferred to the RAM and then the software can be transferred to the EEPROM or FLASH.

- **Embedded Monitor:** On power-on, this software is executed. It can be accessed from the host to download the image as well as debug the software by setting breakpoints.

2.5.2 Boot Sequence

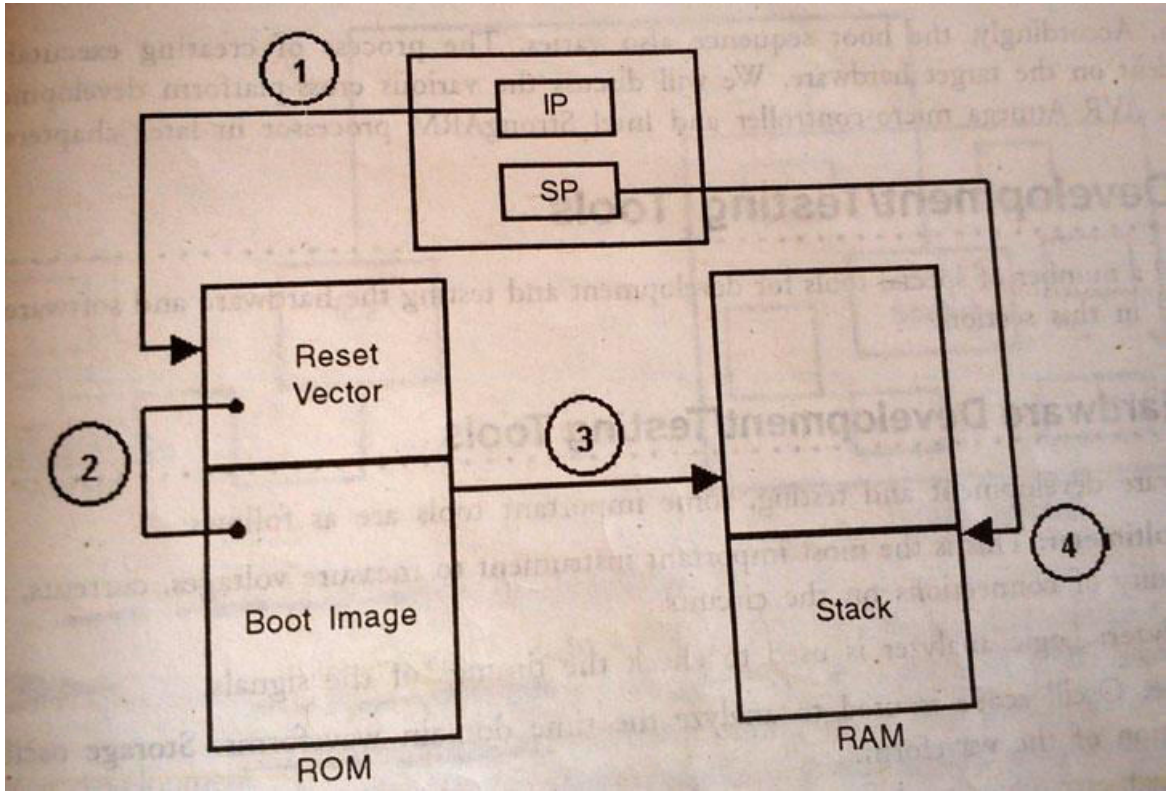
- An embedded system can be booted in one of the following ways:
  - Execute from ROM using RAM for data
  - Execute from RAM after loading the image from RAM
  - Execute from RAM after downloading from the host.

The Process of executing from ROM using RAM for data

- Some embedded devices have such limited memory resources that the program image executes directly out of the ROM. Sometimes the board vendor provides the boot ROM.

- 1. On power-on, the CPU always executes the code contained at a specific address in ROM. This is called **Reset Vector.**

- **(***The reset vector is the default location in a central processing unit which will go to find the first instruction it will execute after a reset. That is to say, the reset vector is a pointer or address where the CPU should always begin as soon as it is able to execute instructions***).**



- 2. The Reset Vector is a jumping instruction to jump to another portion of the memory where the code that resides to boot the system is stored. This is called **bootstrap code**.

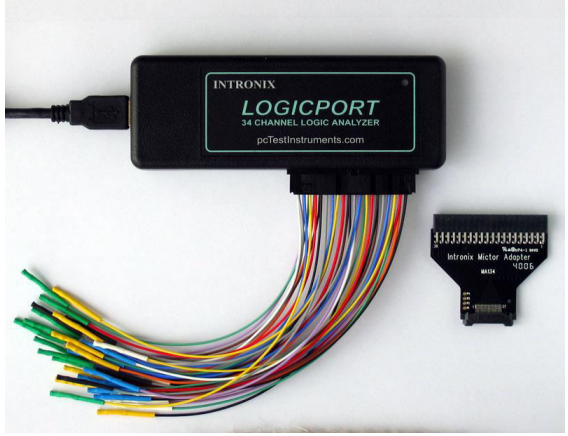
- 3. The executable image contains the data sections which are both readable and writable. These data sections are copied to RAM . Stack space is reversed in RAM.

- 4. CPU’s Stack Pointer is set to point to the beginning of the stack

- Once the boot sequence is completed, the target software will start running. This software does the following:
  - Initialization of the hardware : CPU, memory, bus interfaces and devices are initialized.
  - Initialization of the operating system.
  - Initialization o the application code.

2.6.1 Hardware Development/Testing Tools

- For hardware development and testing, some important tools are as follows:
  - **Digital Multimeter**: This is the most important instrument to measure voltages, currents, and to check the continuity of connections on the circuits.



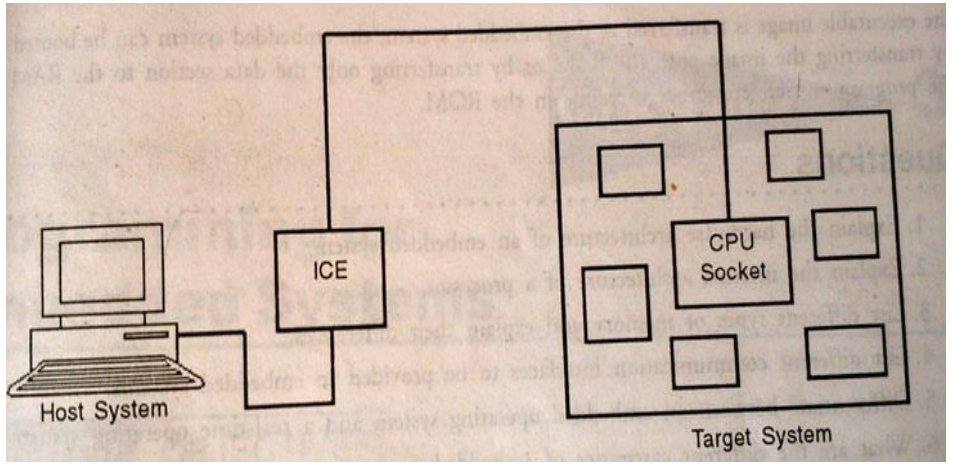
- **Logic Analyzer** :
  - Logic analyzer is used to check the timings of the signals.
  - It gives an exact reflection of what happens when a particular line of firmware is running.
  - A logical analyzer contains special connectors and clips which can be attached to the target board for capturing digital data.
  - In target board debugging applications , it captures the states of various port pins, address bus and data bus of the target processor.

- **Spectrum Analyzer:**
  - A spectrum analyzer is a laboratory instrument that displays signal magnitude (strength) as it varies by signal frequency. The frequency appears on the horizontal axis, and the amplitude is displayed on the vertical axis.
  - Spectrum analyzer is used to analyze the signals in frequency domain.

2.6.2 Software Development/Testing Tools

- For software development, the important development tools are listed below:
  - **Operating System Development Suite:**
    - If you want to make the application off-the-shelf operating system, we can obtain the development suite from the operating system vendor. This development suite contains the API calls to access the OS services. The development suite may run either on a Windows system or Unix system.
  - **Cross-platform development tools:**
    - Cross-compiler generates the object code for the given processor of the source code developed in a high level language such as C or C++. There are several GNU tools that can be downloaded for numerous processors. These cross-compilers provide an Integrated Development Environment (IDE) with editor, compiler, debugger etc. all bundled together.

- **ROM Emulator:**
  - ROM emulator is a debugging tool that helps debug a ROM chip by simulating the ROM with RAM. It is used to debug the software by setting the breakpoints in the memory.
  - More generally, a breakpoint is a means of acquiring knowledge about a program during its execution. During the interruption, the programmer inspects the test environment (general purpose registers, memory, logs, files, etc.) to find out whether the program is functioning as expected.
- **EPROM Programmer:**
  - If the hardware doesnot support in-circuit programming or if there is only EPROM for program memory, there is need of EPROM programmer. Along with it, we need EPROM eraser.
- **Instruction Set Simulator (ISS):**
  - ISS is a software utility that creates a virtual version of the processor on the PC.



- In-Circuit Emulator (ICE):
  - ICE is a device the emulates the CPU.
  - We can plug-in the device in the place of CPU on the target hardware board.
  - This device fits into the CPU socket of the target board on one side, and the other side it is connected to the host through USB port.
  - It is possible to run a debugger on the host to debug the code while it is running on the target hardware.
  - ICE is processor-specific and costly, but an excellent tool for debugging.