Dr.K.Gopi
Department of ECE

UNIT-3

EMBEDDED FIRMWARE DESIGN

Topics:

Embedded Firmware design approaches, Embedded Firmware development languages, Program Models, DFG Models, state Machine Programming Models for Event-controlled Program Flow, Device driver programming, Concepts of C versus Embedded C and Compiler versus Cross-compiler.

Embedded Firmware Design approaches:

- In the traditional embedded system development approach, the Hardware and the software partitioning is done at an early stage. The engineers from software group take care of the software architecture, whereas the engineers from the hardware group are responsible for building the hardware required for the product.
- The firmware design approaches for embedded product is purely dependent on the complexity of the functions to be performed, the speed of operation required, etc.

Two basic approaches are used for Embedded firmware design.

1) Conventional Procedural Based Firmware Design'
2) Embedded Operating System (OS) Based Design'.

The conventional procedural based design is also known as 'Super Loop Model'

The Super Loop Based Approach: The Super Loop based firmware development approach is adopted for applications that are not time critical and where the response time is not so important (embedded systems where missing deadlines are acceptable).

- It is very similar to a conventional procedural programming where the code is executed task by task. The task listed at the top of the program code is executed first and the tasks just below the top are executed after completing the first task.
- In a multiple task based system, each task is executed in serial in this approach. The firmware execution flow for this will be

  1. Configure the common parameters and perform initialization for various hardware components memory, registers, etc.
  2. Start the first task and execute it
  3. Execute the second task

4. Execute the next task
5. Execute the last defined task
6. Jump back to the first task and follow the same flow .

From the firmware execution sequence, it is obvious that the order in which the tasks to be executed are fixed and they are hard coded in the code itself.

The Embedded Operating System (OS) Based Approach:

- The Operating System (OS) based approach contains operating systems, which can be either a General Purpose Operating System (GPOS) or a Real Time Operating System (RTOS) to host the user written application firmware. The General Purpose OS (GPOS) based design is very similar to a conventional PC based application development where the device contains an operating system (Windows/Unix/ Linux, etc. for Desktop PCs) .

  Examples of Embedded products using Microsoft® Windows XP OS are Personal Digital Assistants ; (PDAs), Hand held devices/Portable devices
- Real Time Operating System (RTOS) based design approach is employed in embedded products demanding Real-time response. RTOS respond in a timely and predictable manner to events.
- Real Time operating system contains a Real Time kernel responsible for performing pre-emptive multitasking, scheduler for scheduling tasks etc. A Real Time Operating System (RTOS) allows flexible scheduling of system resources like the CPU and memory and offers some way to communicate between task.

  Examples of RTOS: Windows CE, VxWorks, Embedded Linux', 'Symbian' etc

COMPUTATIONAL MODELS IN EMBEDDED DESIGN

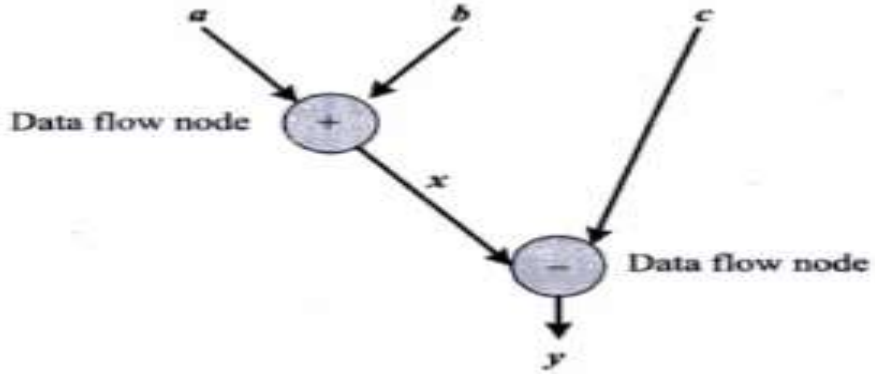
Data Flow Graph (DFG) model, State Machine model, Concurrent Process model, Sequential Program model are the commonly used computational models in embedded system design.

The following sections give an overview of these models**:**

Data Flow Graph/Diagram (DFG) Model:

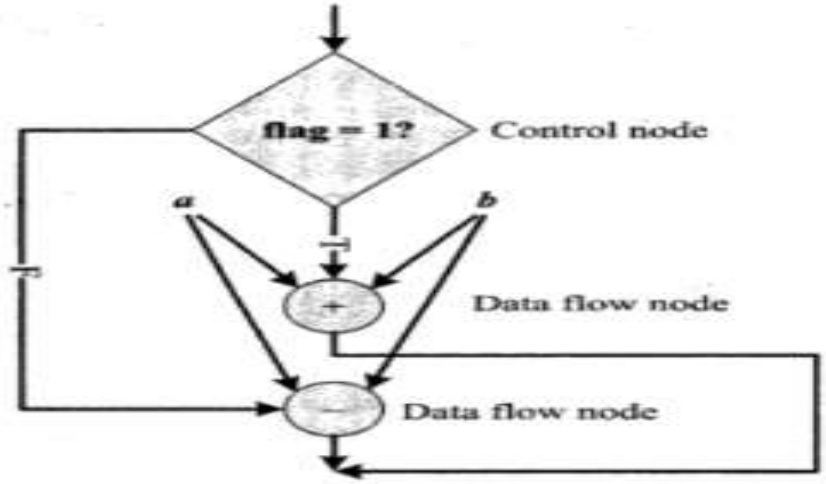
- The Data Flow Graph (DFG) model is a data driven model in which the program execution is determined by data. This model emphasises on the data and operations on the data which transforms the input data to output data.
- Indeed Data Flow Graph (DFG) is a visual model in which the operation on the data (process) is represented using a block (circle) and data flow is represented using arrows.

- An inward arrow to the process (circle) represents input data and an outward arrow from the process (circle) represents output data in DFG notation.DSP applications are typical examples for it (DCT,FFT etc).
- Now let's have a look at the implementation of a DFG. Suppose one of the functions in our application contains the computational requirement x = a + b; and y = x - c.
- In a DFG model, a data path is the data flow path from input to output. A DFG model translates the program as a single sequential process execution. The implementation of the DFG model for the above requirement as shown in the figure.



Control Data Flow Graph/Diagram (CDFG}:

- The DFG model is a data driven model in which the execution is controlled by data and it doesn't involve any control operations (conditionals).
- The Control DFG (CDFG) model is used for modeling applications involving conditional program execution. CDFG models contains both data operations and control operations. Let us have a look at the implementation of the CDFG for the following requirement.
- If flag = 1 , x = a + b; else y = a-b; This requirement contains a decision making process. The control node is represented by a 'Diamond' block which is the decision making element in a normal flow chart based design.



- A real world example for modeling the embedded application using CDFG is the capturing and saving of the image to a format set by the user.
- In a digital still camera where everything is data driven starting from the Analog Front End which converts the CCD sensor generated analog signal to Digital Signal and the task which stores the data from ADC to a frame buffer for the use of a media processor which performs various operations like, auto correction, white balance adjusting, etc.
- The decision on, in which format the image is stored (formats like JPEG, TIFF, BMP, etc.) is controlled by the camera settings, configured by the user.

State Machine Model:

- The State Machine model is used for modeling reactive or event-driven embedded systems whose processing behaviour are dependent on state transitions. Embedded systems used in the control and industrial applications are typical examples for event driven systems.
- The State Machine model describes the system behaviour with 'States', 'Events', 'Actions' and 'Transitions'.
    - **State** is a representation of a current situation.
    - An **event** is an input to the state. The event acts as stimuli for state transition.
    - **Action** is an activity to be performed by the state machine
    - **Transition** is the movement from one state to another state

Example 1 of state machine model (SEAT BELT MONITORING SYSTEM)

- As an example let us consider the design of an embedded system for driver/passenger 'Seat Belt Warning' in an automotive using the FSM model. The system requirements are captured as.

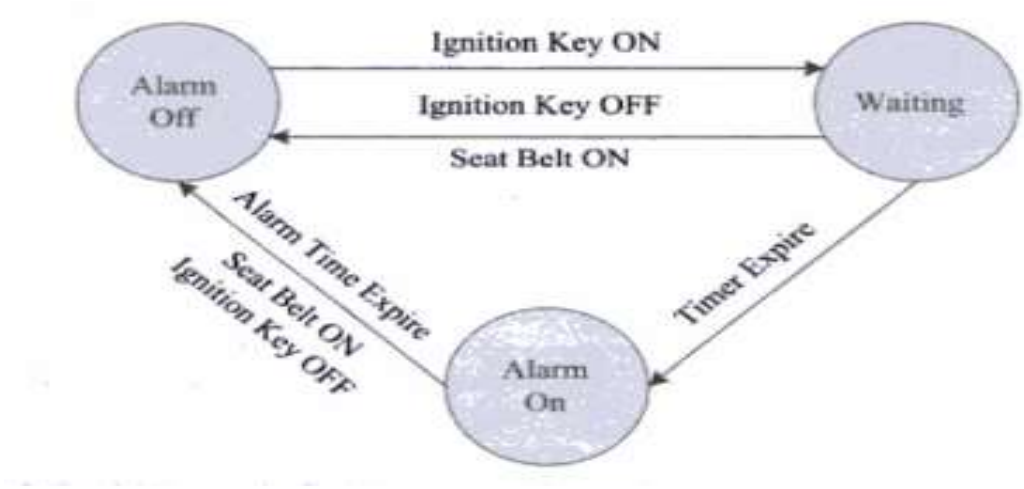
The system requirements are captured as:

1. When the vehicle ignition is turned on and the seat belt is not fastened within 10 seconds of ignition ON, the system generates an alarm signal for 5 seconds.

2. The Alarm is turned off when the alarm time (5 seconds) expires or if the driver/passenger fastens the belt or if the ignition switch is turned off, whichever happens first.

Here the states are 'Alarm Off', 'Waiting' and 'Alarm On' and the events are 'Ignition Key ON', 'Ignition Key OFF', 'Timer Expire', 'Alarm Time Expire' and 'Seat Belt ON'.

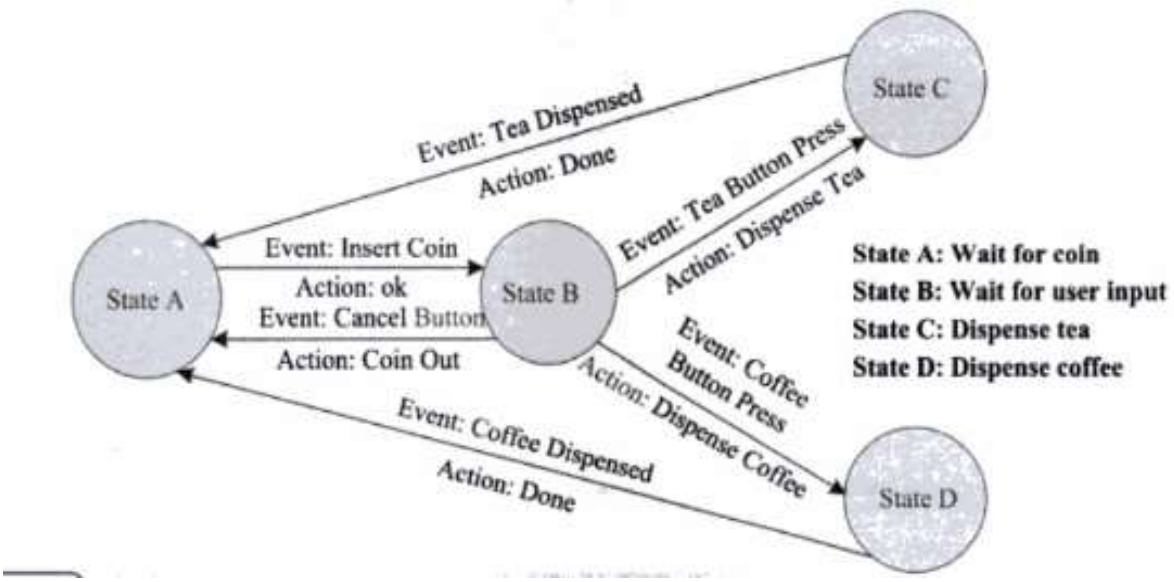
- Using the FSM, the system requirements can be modeled as given in Figure. The 'Ignition Key ON' event triggers the 10 second timer and transitions the state to 'Waiting'.
- If a 'Seat Belt ON' or 'Ignition Key OFF' event occurs during the wait state, the state transitions into 'Alarm Off'. When the wait timer expires in the waiting state, the event 'Timer Expire' is generated and it transitions the state to 'Alarm On' from the 'Waiting' state. The 'Alarm On' state continues until a 'Seat Belt ON' or 'Ignition Key OFF' event or 'Alarm Time Expire' event, whichever occurs first.
- The occurrence of any of these events transitions the state to 'Alarm Off'. The wait state is implemented using a timer.



Example 2: Design an automatic tea/coffee vending machine based on FSM model for the following requirement.

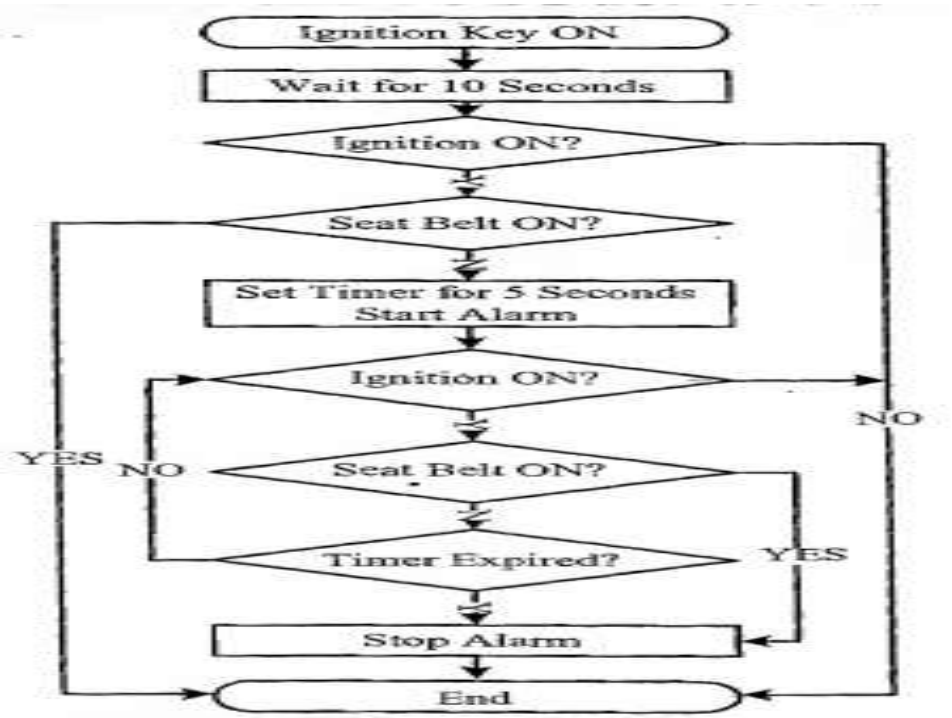
- The tea/coffee vending is initiated by user inserting a 5 rupee coin. After inserting the coin, the user can either select 'Coffee' or 'Tea' or press 'Cancel' to cancel the order and take back the coin.
- The FSM representation for the above requirement is given in Figure. In its simplest representation, it contains four states namely; 'Wait for coin' 'Wait for User Input', 'Dispense Tea' and 'Dispense Coffee'.
- The event 'Insert Coin' (5 rupee coin insertion), transitions the state to 'Wait for User Input'. The system stays in this state until a user input is received from the buttons 'Cancel', 'Tea' or 'Coffee' (Tea and Coffee are the drink select button).

- If the event triggered in 'Wait State' is 'Cancel' button press, the coin is pushed out and the state transitions to 'Wait for Coin'.
- If the event received in the 'Wait State' is either 'Tea' button press, or 'Coffee' button press, the state changes to 'Dispense Tea' and 'Dispense Coffee' respectively. Once the coffee/tea vending is over, the respective



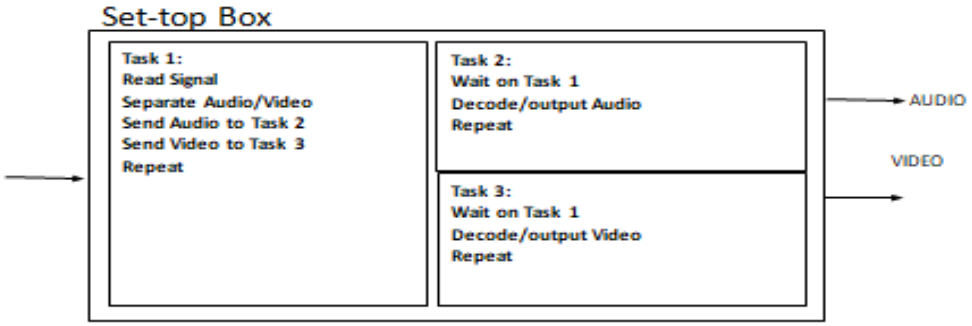
Sequential programming model:

- In the sequential programming Model, the functions or processing requirements are executed in sequence. Here the program instructions are iterated and executed conditionally and the data gets transformed through a series of operations.
- FSM (Finite state machine) are good choice for sequential program modeling. Another important tool used for modelling sequential program is Flow Charts.
- The FSM approach represents the states, events, transitions and actions, whereas the Flow Chart models the execution flow. The execution of functions in a sequential program model for the 'Seat Belt Warning' system is illustrated below.



Concurrent model:

- The concurrent or communicating process model models concurrently executing tasks/processes. It is easier to implement certain requirements in concurrent processing model than the conventional sequential execution.
- Sequential execution leads to a single sequential execution of task and thereby leads to poor processor utilisation, when the task involves I/O waiting, sleeping for specified duration etc.
- If the task is split into multiple subtasks, it is possible to tackle the CPU usage effectively, when the subtask under execution goes to a wait or sleep mode, by switching the task execution. However, concurrent processing model requires additional overheads in task scheduling, task synchronisation and communication.
- **Consider an example having separate tasks running independently but sharing data**



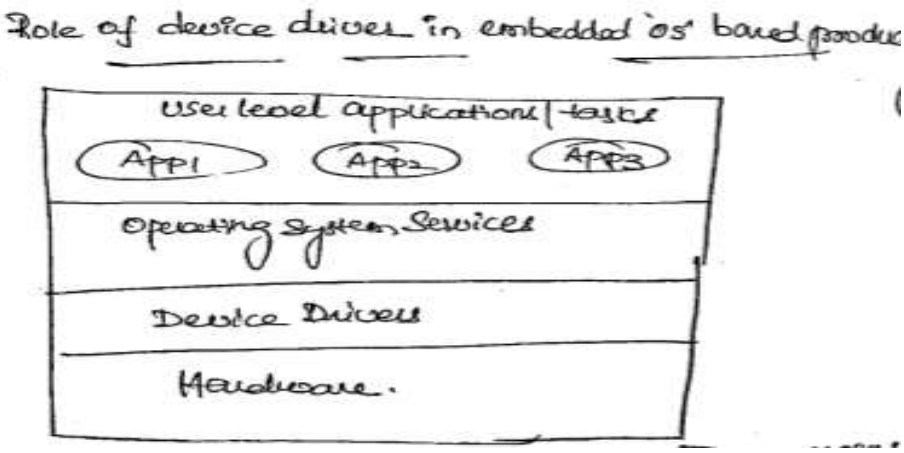
Device Drivers:

Def: Device driver is a piece of software that acts as a bridge between the operating system and the hardware.

In an operating system based product architecture, the user applications talk to the Operating System for all necessary information exchange including communication with the hardware peripherals.

- Device drivers are responsible for initiating and managing the communication with the hardware peripherals. They are responsible for establishing the connectivity, initializing the hardware (setting up various registers of the hardware device) and transferring data.
- An embedded product may contain different types of hardware components like Wi-Fi module, Storage device interface, etc. The initialization of these devices and the protocols required for communicating with these devices may be different.

- All these requirements are implemented in drivers and a single driver will not be able to satisfy all these. Hence each hardware (more specifically each class of hardware) requires a unique driver component.



- Certain drivers come as part of the OS kernel and certain drivers need to be installed on the fly.

- For example, the program storage memory for an embedded product, say Flash memory requires a Flash driver to read and write data from/to it. This driver should come as part of the OS kernel image. Device drivers which are part of the OS image are known as 'Built-in drivers' or 'On-board drivers'. These drivers are loaded by the OS at the time of booting the device and are always kept in the RAM.

Introduction C and embedded C:

- C is generally used for Desktop computers while embedded C is used for Microcontroller based applications (Embedded Systems). C uses the resources of Desktop like memory, OS (operating system) etc. while embedded c uses the limited resources like RAM, Rom, I/O etc.
- Earlier the embedded applications were developed using assembly language programming and the problem is they do not provide portability. Whereas the C is easy to understand, reliable and portable and hence it is preferred.
- Both C and Embedded C have almost same syntax, data types, functions, etc. Embedded C is basically an extension to the Standard C Programming Language with additional features like Addressing I/O, multiple memory addressing and fixed-point arithmetic, etc.

Importance of 'C' :

- C' is a well structured, well defined and standardised general purpose programming language . It supports to improve the quality of computer programming and for solving the problems in a small amount of time.
- Using complier 'c' language is converted into machine level language.
- The C language used the following keywords like, for, if/else, while, switch and do/while and large number of arithmetic and logic operators such +, = =, and ++.
- It does not care about resources of memory because the whole memory is available in CPU.

EMBEDDED 'C':

- Embedded C is most popular programming language in software field for developing electronic gadgets. Each processor used in electronic system is associated with embedded software.
- Embedded C programming plays a key role in performing specific function by the processor. In day-to-day life we used many electronic devices such as mobile phone, washing machine, digital camera, etc. These all device working is based on microcontroller that are programmed by embedded C.
- With the advancement in the technology, embedded systems were associated with the Processors & Controllers which make use of embedded software.
- The embedded C is the extension of C language that finds its application in the embedded system to write embedded software.

- Most of the syntax and some library functions used by Embedded C are same as that of C, like variable declaration, conditional statements, arrays and strings, loops, main () function.
- In daily lives we find various types of Embedded Systems, like mobiles, digital cameras, washing machines, AC etc. These all devices are micro-controller/micro-processor based and mostly embedded devices use Embedded C language to develop their applications.

Advantages of embedded c:

- It is easier to understand.
- It performs the same task all the time so there is no need of any hardware changing such as extra memory or space for storage.
- Hardware cost of embedded c systems is usually so much low.

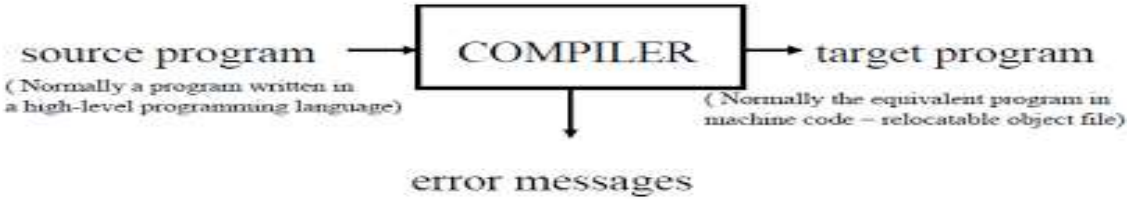
Differences between C and embedded C

C programming	Embedded C programming
C is a general purpose programming language, which can be used to design any type of desktop based applications.	Embedded C is an extension of C language (some of the features are there, which can be used to specific purposes), it is used to develop micro-controller based applications (low-level or/and application level
C language program is hardware independent	Embedded C program is hardware dependent language
Used for desktop applications	Used for microcontroller/ microprocessor
Does not bother about memory resources. i.e., use entire memory available in CPU.	Uses limited resources. Hence consider memory allocation of the embedded processor applications. ➢
Input can be given to the program while it is running	Only the pre-defined input can be given to the running program. ➢
Some of the examples of Applications: logical programs, system software programs	Some of the examples of real-time Applications: DVD, TV, Digital camera

Compiler versus Cross-compiler:

Def: Compiler is a software that transforms a program written in high-level programming language into machine language.

- Programmers write programs for the computer in high-level languages. The Processor/Controllers does not understand these programs. Because processors/Controllers understands only Machine language(0's& 1's). A compiler is a translator that is capable of transforming source code into a machine code. There are various types of compilers. Compilers are generally termed as 'Native Compilers'.
- A native compiler generates machine code for the same machine (processor) on which it is running.

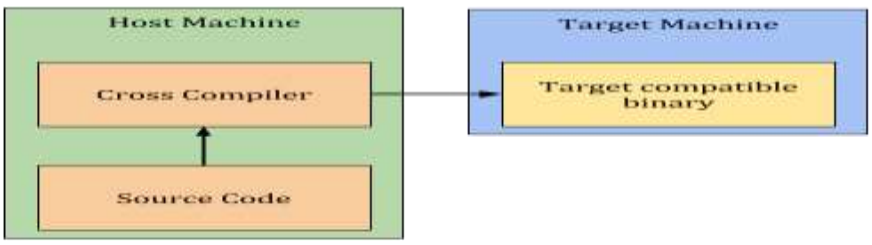


- The compilation occurs before the execution. It displays all error messages in the program . It is impossible to execute the program without fixing these errors.
- After the compilation, these programs generate an intermediate object code. Language such as C and C++ use compilers. Usually, compiler-based languages are fast in execution.

Cross compiler:

Cross compiler is a type of a compiler that can create an executable code for a platform other than the one on which the compiler is running.

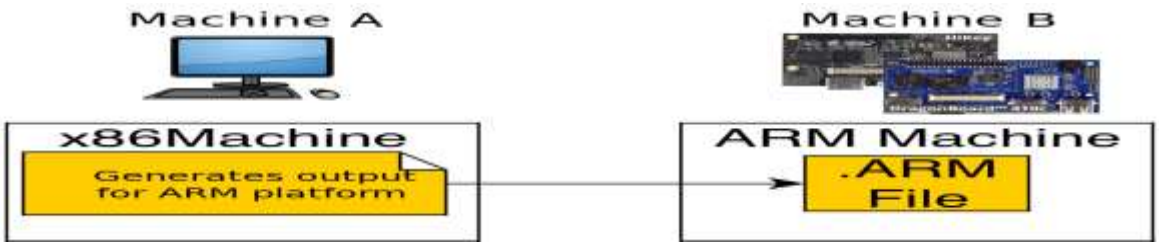
Ex: A compiler that runs on Windows Platform also generates the code that runs on the Linux platform is called as cross compiler.



Cross Compiler operation

Cross-compilers are the software tools used in cross-platform development applications.

- In cross platform development, the compiler running on a particular target processor converts the source code to machine code for a processor whose architecture and instruction set is different from the processor on which the compiler is running.



- The Keil C51 Cross Compiler for the 8051 microcontroller is the most popular 8051 C compiler in the world. It provides more features than any other 8051 C compiler available today.

Features of Cross compilers:

- Full use of the 8051 register banks,
- Use of **AJMP** and **ACALL** instructions,
- Built-in interface for the RTX51 Real-Time Kernel.
- Support for different manufactures such as Atmel, AMD, Cypress, Dallas Semiconductor, Infineon, Philips, and Triscend microcontrollers,
- Support for the Philips 8*x*C750, 8*x*C751, and 8*x*C752 limited instruction sets.
- Note: In Embedded systems where ever we see the word called as Compiler it refers to Cross compiler.

Difference between Compiler and Cross compiler:

Compiler	Cross Compiler
Translates program for same hardware/platform/machine on it is running.	Translates program for different hardware/platform/machine other than the platform which it is running.
It is used to build programs for same system/machine & OS it is installed.	It is used to build programs for other system/machine like AVR/ARM.
It is dependent on System/machine	It is independent of System/machine