UNIT I

1. Define Embedded System. Compare Embedded Systems & General purpose computer system.

Ans

An embedded system is a combination of 3 things:
a. Hardware
b. Software
c. Mechanical Components
And it is supposed to do one specific task only.

Example 1: Washing Machine
A washing machine from an embedded systems point of view has:
1. Hardware: Buttons, Display & buzzer, electronic circuitry.
2. Software: It has a chip on the circuit that holds the software which drives controls & monitors the various operations possible.
3. Mechanical Components: the internals of a washing machine which actually wash the clothes control the input and output of water, the chassis itself.

Example 2: Air Conditioner
An Air Conditioner from an embedded systems point of view has:
1. Hardware: Remote, Display & buzzer, Infrared Sensors, electronic circuitry.
2. Software: It has a chip on the circuit that holds the software which drives controls & monitors the various operations possible. The software monitors the external temperature through the sensors and then releases the coolant or suppresses it.
3. Mechanical Components: the internals of an air conditioner the motor, the chassis, the outlet, etc

- An embedded system is designed to do a specific job only.
  Example: a washing machine can only wash clothes, an air conditioner can control the temperature in the room in which it is placed.
- The hardware & mechanical components will consist all the physically visible things that are used for input, output, etc.
- An embedded system will always have a chip (either microprocessor or microcontroller) that has the code or software which drives the system.

The Embedded System and the General purpose computer are at two extremes. The embedded system is designed to perform a specific task whereas as per definition the general purpose computer is meant for general use. It can be used for playing games, watching movies, creating software, work on documents or spreadsheets etc.
Following are certain specific points of difference between embedded systems and general purpose computers:

Criteria	General Purpose Computer	Embedded system
Contents	It is combination of generic hardware and a general	It is combination of special purpose hardware and

	purpose OS for executing a variety of applications.	embedded OS for executing specific set of applications
Operating Systems	It contains general purpose operating system	It may or may not contain operating system.
Alterations	Applications are alterable by the user.	Applications are non-alterable by the user.
Key Factor	Performance" is key factor.	Application specific requirements are key factors.
Power Consumption	More	Less
Response Time	Not Critical	Critical for some applications

2. **Write a short note on classification of Embedded systems**

Ans The classification of embedded system is based on following criteria's:

- On generation
- On complexity & performance
- On deterministic behaviour
- On triggering

On generation

1. First generation(1G):
   a. Built around 8bit microprocessor & microcontroller.
   b. Simple in hardware circuit & firmware developed.
   c. Examples: Digital telephone keypads.

2. Second generation(2G):
   a. Built around 16-bit µp & 8-bit µc.
   b. They are more complex & powerful than 1G µp & µc.
   c. Examples: SCADA systems

3. Third generation(3G):
   a. Built around 32-bit µp & 16-bit µc.
   b. Concepts like Digital Signal Processors(DSPs),
   c. Application Specific Integrated Circuits(ASICs) evolved.
   Examples: Robotics, Media, etc.

4. Fourth generation:
   a. Built around 64-bit µp & 32-bit µc.
   b. The concept of System on Chips (SoC), Multicore
Processors evolved.
   a. Highly complex & very powerful.
   b. Examples: Smart Phones.

On complexity & performance

1. Small-scale:
   a. Simple in application need

b. Performance not time-critical.
c. Built around low performance & low cost 8 or 16 bit µp/µc.
d. Example: an electronic toy

2. Medium-scale:

a. Slightly complex in hardware & firmware requirement.
b. Built around medium performance & low cost 16 or 32 bit µp/µc.
c. Usually contain operating system.
d. Examples: Industrial machines.

3. Large-scale:

a. Highly complex hardware & firmware.
b. Built around 32 or 64 bit RISC µp/µc or PLDs or Multicore Processors.
c. Response is time-critical.
d. Examples: Mission critical applications.

On deterministic behaviour

a. This classification is applicable for "Real Time" systems.
b. The task execution behaviour for an embedded system may be deterministic or non-deterministic.
c. Based on execution behaviour Real Time embedded systems are divided into Hard and Soft.

On triggering

a. Embedded systems which are "Reactive" in nature can be based on triggering.
b. Reactive systems can be:
   - Event triggered
   - Time triggered

3. Differentiate between a RISC & CISC

Ans

RISC	CISC
1. Reduced Instruction Set Computing	1. Complex Instruction Set Computing
2. It contains lesser number of instructions.	2. It contains greater number of instructions.
3. Instruction pipelining and increased execution speed.	3. Instruction pipelining feature does not exist.
4. Orthogonal instruction set (allows each instruction to operate on any register and use any addressing mode.	4. Non-orthogonal set(all instructions are not allowed to operate on any register and use any addressing mode.
5. Operations are performed on registers only, only memory operations are load and store.	5. Operations are performed either on registers or memory depending on instruction.
6. A larger number of registers are available.	6. The number of general purpose registers are very limited.
7. Programmer needs to write more code to execute a task since	7. Instructions are like macros in C language. A programmer can

instructions are simpler ones.	achieve the desired functionality with a single instruction which in turn provides the effect of using more simpler single instruction in RISC.
8. It is single, fixed length instruction.	8. It is variable length instruction.
9. Less silicon usage and pin count.	9. More silicon usage since more additional decoder logic is required to implement the complex instruction decoding.
10. With Harvard Architecture.	10. Can be Harvard or Von-Neumann Architecture.

4. Write a short note on COTs.

Ans **Commercial off-the-shelf components(COTs)**

1) A Commercial off the Shelf product is one which is used 'asis'.
2) The COTS components itself may be develop around a general purpose or domain specific processor or an ASICs or a PLDs.
3) The major advantage of using COTS is that they are readily available in the market, are chip and a developer can cut down his/her development time to a great extent
4) The major drawback of using COTS components in embedded design is that the manufacturer of the COTS component may withdraw the product or discontinue the production of the COTS at any time if rapid change in technology occurs.
5) **Advantages of COTS:**
   a. Ready to use
   b. Easy to integrate
   c. Reduces development time
6) **Disadvantages of COTS:**
   a. No operational or manufacturing standard (all proprietary)
   b. Vendor or manufacturer may discontinue production of a particular COTS product.

5. Explain application specific IC & programmable logic devices

Ans **Application Specific Integrated Circuits. (ASIC)**

- ASICs is a microchip design to perform a specific and unique application.
- Because of using single chip for integrates several functions there by reduces the system development cost.
- Most of the ASICs are proprietary (which having some trade name) products, it is referred as Application Specific Standard Products(ASSP).
- As a single chip ASIC consumes a very small area in the total system. Thereby helps in the design of smaller system with high capabilities or functionalities.
- The developers of such chips may not be interested in revealing the internal detail of it.

Programmable logic devices(PLD's)

- A PLD is an electronic component. It used to build digital circuits which are reconfigurable.
- A logic gate has a fixed function but a PLD does not have a defined function at the time of manufacture.
- PLDs offer customers a wide range of logic capacity, features, speed, voltage characteristics.
- PLDs can be reconfigured to perform any number of functions at any time.
- A variety of tools are available for the designers of PLDs which are inexpensive and help to develop, simulate and test the designs.
- PLDs having following two major types.
  1) CPLD(Complex Programmable Logic Device):
  CPLDs offer much smaller amount of logic up to 1000 gates.
  2) FPGAs(Field Programmable Gate Arrays):
  It offers highest amount of performance as well as highest logic density, the most features.

Advantages of PLDs :-

- PLDs offer customer much more flexibility during the design cycle.
- PLDs do not require long lead times for prototypes or production parts because PLDs are already on a distributors shelf and ready for shipment.
- PLDs can be reprogrammed even after a piece of equipment is shipped to a customer.

6. **Explain various operational as well as non operational quality attributes of an Embedded System.**

Ans There are two types of quality attributes are:-

1. Operational Quality Attributes.
 These are attributes related to operation or functioning of an embedded system. The way an embedded system operates affects its overall quality.

2. Non-Operational Quality Attributes.
 These are attributes **not** related to operation or functioning of an embedded system. The way an embedded system operates affects its overall quality.
 These are the attributes that are associated with the embedded system before it can be put in operation.

1.1 Operational Attributes
a) Response
Response is a measure of quickness of the system.
It gives you an idea about how fast your system is tracking the input variables.
Most of the embedded system demand fast response which should be real-time.

	b) Throughput Throughput deals with the efficiency of system. It can be defined as rate of production or process of a defined process over a stated period of time. In case of card reader like the ones used in buses, throughput means how much transaction the reader can perform in a minute or hour or day. **c) Reliability** Reliability is a measure of how much percentage you rely upon the proper functioning of the system. Mean Time between failures and Mean Time To Repair are terms used in defining system reliability. Mean Time between failures can be defined as the average time the system is functioning before a failure occurs. Mean time to repair can be defined as the average time the system has spent in repairs. **d) Maintainability** Maintainability deals with support and maintenance to the end user or a client in case of technical issues and product failures or on the basis of a routine system checkup It can be classified into two types :- **1. Scheduled or Periodic Maintenance** • This is the maintenance that is required regularly after a periodic time interval. • Example : Periodic Cleaning of Air Conditioners Refilling of printer cartridges. **2. Maintenance to unexpected failure** • This involves the maintenance due to a sudden breakdown in the functioning of the system. • Example: Air conditioner not powering on Printer not taking paper in spite of a full paper stack **e) Security** Confidentiality, Integrity and Availability are three corner stones of information security. Confidentiality deals with protection data from unauthorized disclosure. Integrity gives protection from unauthorized modification. Availability gives protection from unauthorized user Certain Embedded systems have to make sure they conform to the security measures. Ex. An Electronic Safety Deposit Locker can be used only with a pin number like a password. **f) Safety** Safety deals with the possible damage that can happen to the operating

person and environment due to the breakdown of an embedded system or due to the emission of hazardous materials from the embedded products.
A safety analysis is a must in product engineering to evaluate the anticipated damage and determine the best course of action to bring down the consequence of damages to an acceptable level.

Non Operational Attributes
a) Testability and Debug-ability
It deals with how easily one can test his/her design, application and by which mean he/she can test it.
In hardware testing the peripherals and total hardware function in designed manner
Firmware testing is functioning in expected way
Debug-ability is means of debugging the product as such for figuring out the probable sources that create unexpected behavior in the total system

b) Evolvability
For embedded system, the qualitative attribute "Evolvability" refer to ease with which the embedded product can be modified to take advantage of new firmware or hardware technology.

c) Portability
Portability is measured of "system Independence".
An embedded product can be called portable if it is capable of performing its operation as it is intended to do in various environments irrespective of different processor and or controller and embedded operating systems.

d) Time to prototype and market
Time to Market is the time elapsed between the conceptualization of a product and time at which the product is ready for selling or use
Product prototyping help in reducing time to market.
Prototyping is an informal kind of rapid product development in which important feature of the under consider are develop.
In order to shorten the time to prototype, make use of all possible option like use of reuse, off the self component etc**.**

e) Per unit and total cost
Cost is an important factor which needs to be carefully monitored. Proper market study and cost benefit analysis should be carried out before taking decision on the per unit cost of the embedded product.
When the product is introduced in the market, for the initial period the sales and revenue will be low
There won't be much competition when the product sales and revenue increase.
During the maturing phase, the growth will be steady and revenue reaches highest point and at retirement time there will be a drop in sales volume.

7. **Write a short note on Communication Interface**

Ans **COMMUNICATION INTERFACES**

For any embedded system, the communication interfaces can broadly classified into:

1. Onboard Communication Interfaces

These are used for internal communication of the embedded system i.e: communication between different components present on the system.

Common examples of onboard interfaces are:

- Inter Integrated Circuit (I2C)
- Serial Peripheral Interface (SPI)
- Universal Asynchronous Receiver Transmitter (UART)
- 1-Wire Interface
- Parallel Interface

Example :**Inter Integrated Circuit (I2C)**

- It is synchronous
- Bi-directional, half duplex , two wire serial interface bus
- Developed by Phillips semiconductors in 1980
- It comprises of two buses :
  1. Serial clock –SCL
  2. Serial Data – SDA
- SCL generates synchronization clock pulses
- SDA transmits data serially across devices
- I2C is a shared bus system to which many devices can be connected
- Devices connected by I2C can act as either master or slave
- The master device is responsible for controlling communication by initiating/ terminating data transfer.
- Devices acting as slave wait for commands from the master and respond to those commands.

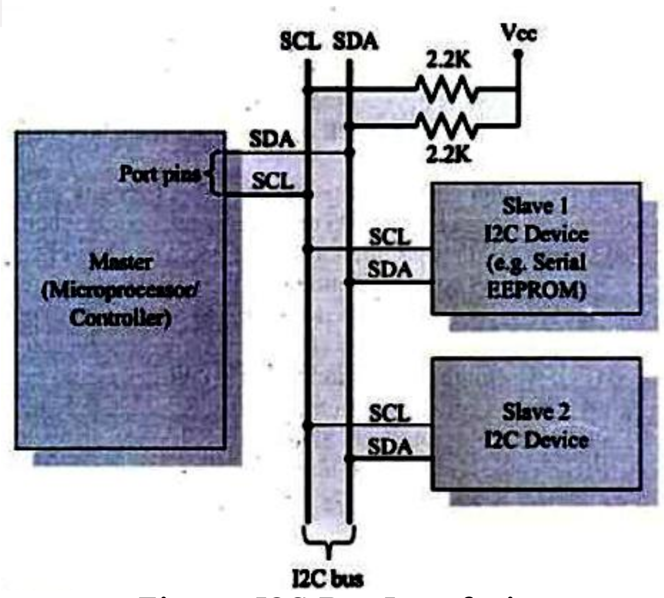

SCL SDA
Vcc
2.2K
2.2K
SDA
Port pins
SCL
Master (Microprocessor/ Controller)
SCL
SDA
Slave 1 I2C Device (e.g. Serial EEPROM)
SCL
SDA
Slave 2 I2C Device
I2C bus

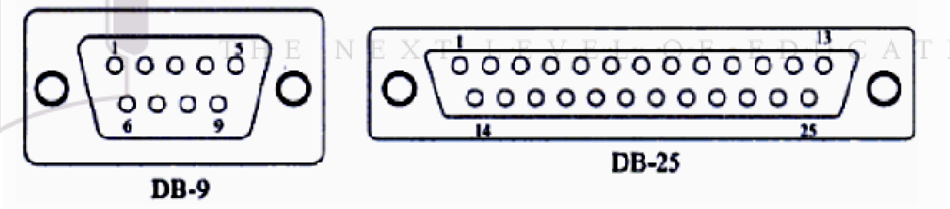
Figure: I2C Bus Interfacing

2. External or Peripheral Communication Interfaces
These are used for external communication of the embedded system i.e: communication of different components present on the system with external or peripheral components/devices.

Common examples of external interfaces are:

RS-232 C & RS-485
Universal Serial Bus (USB)
IEEE 1394 (Firewire)
Infrared (IrDA)
Bluetooth
Wi-Fi
Zig Bee
General Packet Radio Service (GPRS)
Example: RS-232 C & RS-485

It is wired, asynchronous, serial, full duplex communication
RS 232 interface was developed by EIA (Electronic Industries Associates) In early 1960s
RS 232 is the extension to UART for external communications
RS-232 logic levels use:
+3 to +25 volts to signify a "Space" (Logic 0) and
-3 to -25 volts to signify a "Mark" (logic 1).
RS 232 supports two different types of connectors :
DB 9 and DB 25 as shown in figure below



RS 232 interface is a point to point communication interface and the devices involved are called as Data Terminating Equipment (DTE) And Data Communications Terminating Equipment (DCE)
Embedded devices contain UART for serial transmission and generate signal levels as per TTL/CMOS logic.
A level translator IC (like Max 232) is used for converting the signal lines from UART to RS 232 signal lines for communication.
The vice versa is performed on the receiving side.
Converter chips contain converters for both transmitters and receivers
RS 232 is used only for point to point connections
It is susceptible to noise and hence is limited to short distances only
RS 422 is another serial interface from EIA.
It supports multipoint connections with 1 transmitter and 10 receivers.
It supports data rates up to 100Kbps and distance up to 400 ft
RS 485 is enhanced version of RS 422 and supports up to 32

	transmitters and 32 receivers
8.	**What do you mean by Sensor & Explain its various types**
Ans	**Sensor** • A Sensor is used for taking Input • It is a transducer that converts energy from one form to another for any measurement or control purpose • In the broadest definition, a sensor is a device, module, or subsystem whose purpose is to detect events or changes in its environment and send the information to other electronics, frequently a computer processor. • A sensor is always used with other electronics, whether as simple as a light or as complex as a computer. **Different Types of Sensors** Ex. • Temperature Sensor. • Proximity Sensor. • Accelerometer. • IR Sensor (Infrared Sensor) • Pressure Sensor. • Light Sensor. • Ultrasonic Sensor. • Smoke, Gas and Alcohol Sensor
	UNIT II
1.	**Define application specific embedded system**
Ans	An embedded system is a combination of 3 things: a. Hardware b. Software c. Mechanical Components And it is supposed to do one specific task only. The application areas and the products in the embedded domain are countless. 1. Consumer Electronics: Camcorders, Cameras. 2. Household appliances: Washing machine, Refrigerator. 3. Automotive industry: Anti-lock breaking system(ABS), engine control. 4. Home automation & security systems: Air conditioners, sprinklers, fire alarms. 5. Telecom: Cellular phones, telephone switches. 6. Computer peripherals: Printers, scanners. 7. Computer networking systems: Network routers and switches. 8. Healthcare: EEG, ECG machines. 9. Banking & Retail: Automatic teller machines, point of sales. 10. Card Readers: Barcode, smart card readers

Example 1: Washing Machine
A washing machine from an embedded systems point of view has:

a. Hardware: Buttons, Display & buzzer, electronic circuitry.
b. Software: It has a chip on the circuit that holds the software which drives controls & monitors the various operations possible.
c. Mechanical Components: the internals of a washing machine which actually wash the clothes control the input and output of water, the chassis itself.

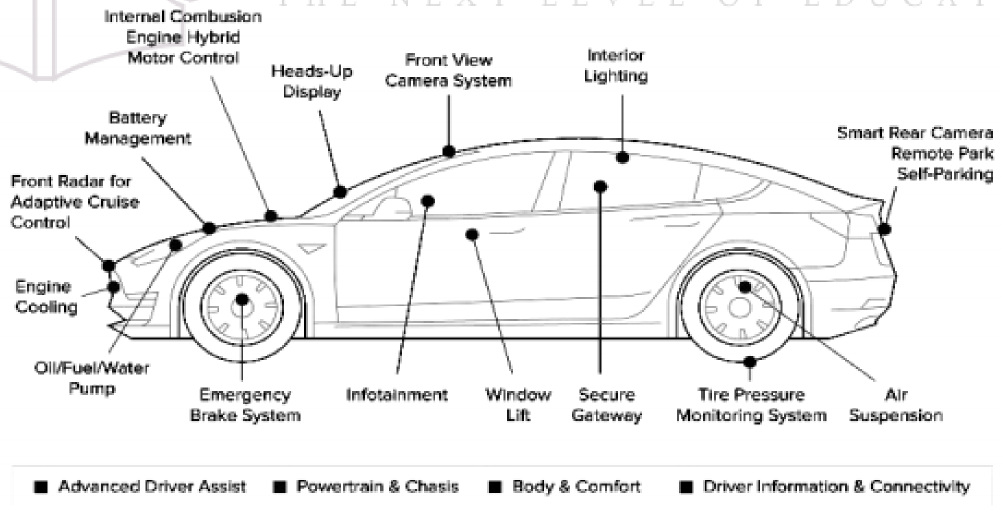
Example 2: Air Conditioner
An Air Conditioner from an embedded systems point of view has:

a. Hardware: Remote, Display & buzzer, Infrared Sensors, electronic circuitry.
b. Software: It has a chip on the circuit that holds the software which drives controls & monitors the various operations possible. The software monitors the external temperature through the sensors and then releases the coolant or suppresses it.
c. Mechanical Components: the internals of an air conditioner the motor, the chassis, the outlet, etc

An embedded system is designed to do a specific job only.
Example: a washing machine can only wash clothes, an air conditioner can control the temperature in the room in which it is placed.
The hardware & mechanical components will consist all the physically visible things that are used for input, output, etc.
An embedded system will always have a chip (either microprocessor or microcontroller) that has the code or software which drives the system.

2. **Write a short note on Domain specific Embedded system**

Ans



Automotive Embedded System (AES)
The Automotive industry is one of the major application domains of embedded systems.

	Automotive embedded systems are the one where electronics take control over the mechanical system. Ex. Simple viper control. The number of embedded controllers in a normal vehicle varies somewhere between 20 to 40 and can easily be between 75 to 100 for more sophisticated vehicles. One of the first and very popular use of embedded system in automotive industry was microprocessor based fuel injection. Some of the other uses of embedded controllers in a vehicle are listed below: 1. Air Conditioner 2. Engine Control 3. Fan Control 4. Headlamp Control 5. Automatic break system control 6. Wiper control 7. Air bag control 8. Power Windows AES are normally built around microcontrollers or DSPs or a hybrid of the two and are generally known as Electronic Control Units (ECUs)
3.	**Explain Memory used in Embedded system and also explain memory map**
Ans	Types of memory There are three main types of memories, they are RAM, ROM, Hybrid **RAM (Random Access Memory)** It is read write memory. Data at any memory location can be read or written. It is volatile memory, i.e. retains the contents as long as electricity is supplied. Data access to RAM is very fast. **Types of RAM** There are 2 important memory device in the RAM family. SRAM (Static RAM) DRAM (Dynamic RAM) **SRAM (Static RAM)** It retains the content as long as the power is applied to the chip. If the power is turned off then its contents will be lost forever. **DRAM (Dynamic RAM)** DRAM has extremely short Data lifetime (usually less than a quarter of second). This is true even when power is applied constantly. A DRAM controller is used to make DRAM behave more like SRAM The DRAM controller periodically refreshes the data stored in the DRAM. By refreshing the data several times a second, the DRAM controller keeps the contents of memory alive for a long time. **ROM (Read Only Memory)** It is read only memory. Data at any memory location can be only read.

It is non-volatile memory, i.e. the contents are retained even after electricity is switched off and available after it is switched on.
Data access to ROM is slow compared to RAM

Types of ROM
There are three types of ROM described as follows:
Masked ROM
These are hardwired memory devices found on system.
It contains pre-programmed set of instruction and data and it cannot be modified or appended in any way. (It is just like an Audio CD that contains songs pre-written on it and does not allow to write any other data)
The main advantage of masked ROM is low cost of production.
Prom (Programmable ROM)
This memory device comes in an un-programmed state i.e. at the time of purchased it is in an un-programmed state and it allows the user to write his/her own program or code into this ROM.

In the un-programmed state the data is entirely made up of 1's.

PROMs are also known as one-time-programmable (OTP) device because any data can be written on it only once. If the data on the chip has some error and needs to be modified this memory chip has to be discarded and the modified data has to be written to another new PROM.

Eprom (Erasable-And-Programable ROM)
It is same as PROM and is programmed in same manner as a PROM.
It can be erased and reprogrammed repeatedly as the name suggests.
The erase operation in case of an EPROM is performed by exposing the chip to a source of ultraviolet light.
The reprogramming ability makes EPROM as essential part of software development and testing process.

Hybrid
It is combination of RAM as well as ROM
It has certain features of RAM and some of ROM
Like RAM the contents to hybrid memory can be read and written
Like ROM the contents of hybrid memory are non volatile

Types of Hybrid Memory
There are three types of Hybrid memory devices:
EEPROMs
a. EEPROMs stand for Electrically Erasable and Programmable ROM.
b. It is same as EPROM, but the erase operation is performed electrically. Any byte in EEPROM can be erased and rewritten as desired
Flash
a. Flash memory is the most recent advancement in memory technology.
b. Flash memory devices are high density, low cost, nonvolatile, fast (to read, but not to write), and electrically reprogrammable.

Flash is much more popular than EEPROM and is rapidly displacing many of the ROM devices. Flash devices can be erased only one sector at a time, not byte by byte.

NVRAM
NVRAM is usually just a SRAM with battery backup.
When power is turned on, the NVRAM operates just like any other SRAM but when power is off, the NVRAM draws enough electrical power from the battery to retain its content.
NVRAM is fairly common in embedded systems.
It is more expensive than SRAM.

Memory Map
A Memory Map is the processor's "address book." It shows what these devices look like to the processor. The memory map contains one entry for each of the memories and peripherals that are accessible from the processor's memory space.
All processors store their programs and data in memory.
These chips are located in the processor's memory space, and the processor communicates with them by way of two sets of electrical wires called the address bus and the data bus. To read or write a particular location in memory, the processor first writes the desired address onto the address bus. The data is then transferred over the data bus.

A memory map is a table that shows the name and address range of each memory device and peripheral that is located in the memory space.

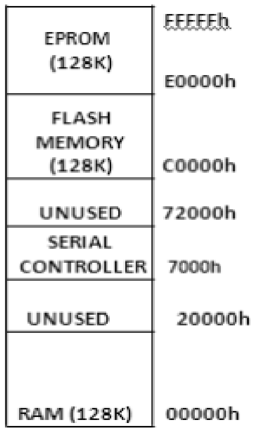
Organize the table such that the lowest address is at the bottom and the highest address is at the top. Each time a new device is added, add it to the memory map, place it in its approximate location in memory and label the starting and ending addresses, in hexadecimal. After inserting all of the devices into the memory map, be sure to label any unused memory regions as such.

The block diagram of the Printer sharing device shown above contains three devices attached to the address and data buses. These devices are the RAM and ROM and a Serial Controller.

Let us assume that the RAM is located at the bottom of memory and extends upward for the first 128 KB of the memory space.

The ROM is located at the top of memory and extends downward for 256 KB. But considering the ROM contains two ROMs-an EPROM and a Flash memory device-each of size 128 KB.

The third device, the Serial Controller, is a memory-mapped peripheral whose registers are accessible between the addresses say 70000h and 72000h. The diagram below shows the memory map for the printer sharing device.



For every embedded system, a header file should be created that describes these important features and provides an abstract interface to the hardware. It allows the programmer to refer to the various devices on the board by name, rather than by address.

The part of the header file below describes the memory map

```
#define RAM_BASE (void *) 0x00000000
#define SC_BASE (void *) 0x70000000
#define SC_INTACK (void *) 0x70001000
#define FLASH_BASE (void *) 0xC0000000
#define EPROM_BASE (void *) 0xE0000000
```

Drain pipe: The drain pipe enables removing the dirty water from the washing that has been used for the washing purpose.

4. **What is memory testing & its purpose & control status register**

Ans The purpose of a memory test is to confirm that each storage location in a memory device is working.

Memory Testing is performed when prototype hardware is ready and the designer needs to verify that address and data lines are correctly wired and memory chips are working properly. Basic idea implement in testing can be understood by this simple task:
Write some set of Data values to each Address in Memory and Read it back to verify.
Ex. If number '50' is stored at a particular Address it is expected to be there unless rewritten or erased.

If all values are verified by reading back then Memory device passes the test. Only through careful selection of data values can make sure passing result to be meaningful.

Difficulties involved in memory testing:
It can be difficult to detect all memory problems with a simple test.
Many Embedded Systems include Memory Tests only to detect catastrophic

memory failures which might not even notice memory chips removal.

A Strategy for memory testing
For memory testing the strategy adopted should be effective and efficient. Ideally there should be multiple small tests instead of one large test.

It would be best to have three individual memory tests:
A data bus test: Checks electrical wiring problems
An address bus test: Checks improperly inserted chips
A device test: Checks to detect missing chips and catastrophic failures and problems with the control bus wiring

These tests have to be executed in a proper order which is: data bus test first, followed by the address bus test, and then the device test. That's because the address bus test assumes a working data bus, and the device test results are meaningless unless both the address and data buses are known to be good.

5. **Write a short note on Watch Dog Timer**

Ans **WATCHDOG TIMER**
It is hardware equipment.
It is special purpose hardware that protects the system from software hangs.
Watchdog timer always counts down from some large number to zero
This process takes a few seconds to reset, in the meantime, it is possible for embedded software to "kick" the watchdog timer, to reset its counter to the original large number.
If the timer expires i.e. counter reaches zero, the watchdog timer will assume that the system has entered a state of software hang, then resets the embedded processor and restarts the software
It is a common way to recover from unexpected software hangs
The figure below diagrammatically represents the working of the watchdog timer

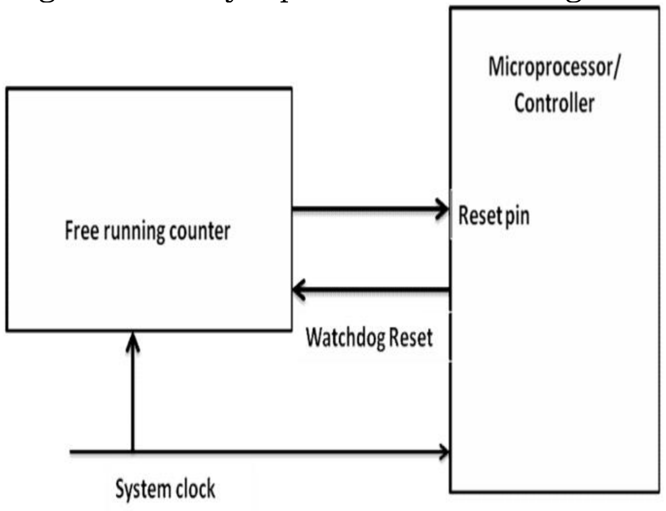


Figure: Watchdog Timer

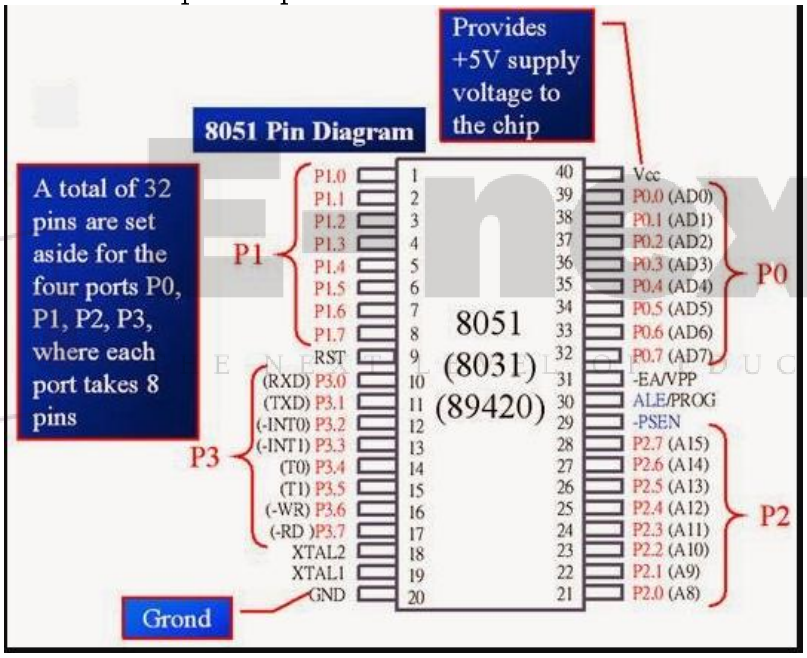
UNIT III

1 Explain Different types of Micro Controller

2. Explain Pin diagram of 8051

Ans **Pin Diagram of 8051 Microcontroller**

8051 microcontroller families (89C51, 8751, DS89C4xO, 89C52) come in different packages like quad-flat package, leadless chip carrier and dual-in-line package. These all packages consist of 40 pins which are dedicated to several functions such as I/O, address, RD, WR, data and interrupts. But, some companies offer a 20-pin version of the microcontrollers for less demanding applications by reducing the number of I/O ports. Nevertheless, a vast majority of developers use the 40-pin chip.



The pin diagram of 8051 microcontroller consists of 40 pins as shown below. A total of 32 pins are set away into four Ports such as P0, P1, P2 and P3. Where, each port contains 8 pins. Therefore, the microcontroller 8051's pin diagram and explanation is given below.

- **Port1 (Pin1 to Pin8):** Port1 includes pin1.0 to pin1.7 and these pins can be configured as input or output pins.
- **Pin 9 (RST):** Reset pin is used to Reset 8051 Microcontroller by giving a positive pulse to this Pin.
- **Port3 (Pin 10 to 17):** The Port3 Pins are similar to port1 pins and can be used as universal Input or output pins. These pins dual-function Pins and

	the function of each Pin is given as: • **Pin 10 (RXD):** RXD pin is a Serial Asynchronous Communication Input or Serial synchronous Communication Output. • **Pin 11 (TXD):** Serial Asynchronous Communication Output or Serial Synchronous Communication clock Output. • **Pin 12 (INT0):** Input of Interrupt 0 • **Pin 13 (INT1):** Input of Interrupt 1 • **Pin 14 (T0):** Input of Counter 0 clock • **Pin 15 (T1):** Input of Counter 1 clock • **Pin 16 (WR):** Writing Signal to write content on external RAM. • **Pin 17 (RD):** Reading Signal to read contents of external RAM. • **Pin 18 and 19 (XTAL2, XTAL1):** X2 and X1 pins are input output pins for the oscillator. These pins are used to connect an internal oscillator to the microcontroller. • **Pin 20 (GND):** Pin 20 is a ground pin. • **Port2 (Pin 21 to Pin28):** Port2 includes pin21 to pin28 which can be configured as Input Output Pins. But, this is only possible when we don't use any external memory. If we use external memory, then these pins will work as high order address bus (A8 to A15). • **Pin 29 (PSEN):** This pin is used to enable external program memory. If we use an external ROM for storing the program, then logic 0 appears on it, which indicates Micro controller to read data from the memory. • **Pin 30 (ALE):** Address Latch Enable pin is an active high-output signal. If we use multiple memory chips, then this pin is used to distinguish between them. This Pin also gives program pulse input during programming of EPROM. • **Pin 31 (EA):** If we have to use multiple memories then the application of logic 1 to this pin instructs the Microcontroller to read data from both memories: first internal and then external. • **Port 0 (Pin 32 to 39):** Similar to the port 2 and 3 pins, these pins can be used as input output pins when we don't use any external memory. When ALE or Pin 30 is at 1, then this port is used as data bus: when the ALE pin is at 0, then this port is used as a lower order address bus (A0 to A7) • **Pin40 (VCC):** This VCC pin is used for power supply.
3.	**List features of 8051**
Ans	An 8051 microcontroller comes bundled with the following features: • 64K bytes on-chip program memory (ROM) • 128 bytes on-chip data memory (RAM) • Four register banks • 128 user defined software flags • 8-bit bidirectional data bus • 16-bit unidirectional address bus • 32 general purpose registers each of 8-bit • 16 bit Timers (usually 2, but may have more or less) • Three internal and two external Interrupts • Four 8-bit ports,(short model have two 8-bit ports)

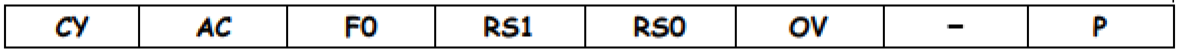
- 16-bit program counter and data pointer
- 8051 may also have a number of special features such as UARTs, ADC, Op-amp, etc

4. **Explain flag bits & PSW register**

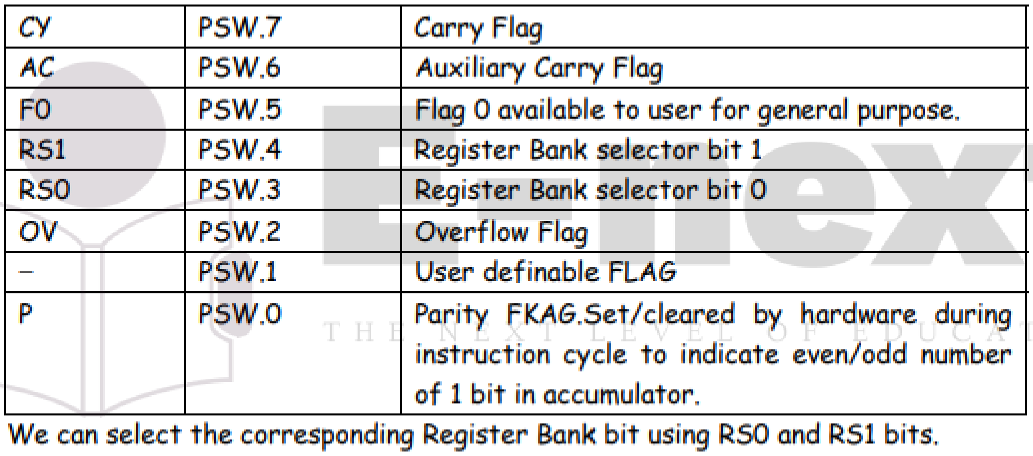
Ans

The program status word (PSW) register is an 8-bit register, also known as flag register. It is of 8-bit wide but only 6-bit of it is used. The two unused bits are user-defined flags. Four of the flags are called conditional flags, which means that they indicate a condition which results after an instruction is executed. These four are CY (Carry), AC (auxiliary carry), P (parity), and OV (overflow). The bits RS0 and RS1 are used to change the bank registers. The following figure shows the program status word register.

The PSW Register contains that status bits that reflect the current status of the CPU.

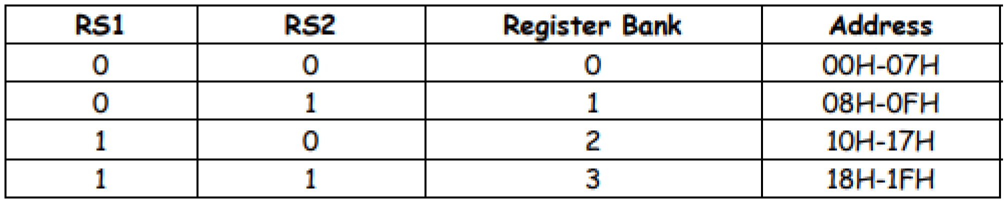


CY	AC	F0	RS1	RS0	OV	–	P



CY	PSW.7	Carry Flag
AC	PSW.6	Auxiliary Carry Flag
F0	PSW.5	Flag 0 available to user for general purpose.
RS1	PSW.4	Register Bank selector bit 1
RS0	PSW.3	Register Bank selector bit 0
OV	PSW.2	Overflow Flag
–	PSW.1	User definable FLAG
P	PSW.0	Parity FKAG.Set/cleared by hardware during instruction cycle to indicate even/odd number of 1 bit in accumulator.

We can select the corresponding Register Bank bit using RS0 and RS1 bits.



RS1	RS2	Register Bank	Address
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

- **CY, the carry flag** – This carry flag is set (1) whenever there is a carry out from the D7 bit. It is affected after an 8-bit addition or subtraction operation. It can also be reset to 1 or 0 directly by an instruction such as "SETB C" and "CLR C" where "SETB" stands for set bit carry and "CLR" stands for clear carry.

- **AC, auxiliary carry flag** – If there is a carry from D3 and D4 during an ADD or SUB• operation, the AC bit is set; otherwise, it is cleared. It is used for the instruction to perform binary coded decimal arithmetic.

	• **P, the parity flag** – The parity flag represents the number of 1's in the accumulator• register only. If the A register contains odd number of 1's, then P = 1; and for even number of 1's, P = 0. • **OV, the overflow flag** – This flag is set whenever the result of a signed number• operation is too large causing the high-order bit to overflow into the sign bit. It is used only to detect errors in signed arithmetic operations.
5.	**Write a note on data types of 8052 C**
Prepare for few programs 1 or 2 questions may be programming	
6.	**Write an 8051 C program to toggle bits of P1 ports continuously with a 250 ms.**

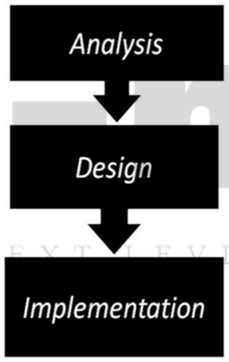
Ans	`#include void MSDelay(unsigned int);`
	`void main(void)`
	` {`
	`while (1) //repeat forever`
	`{`
	` p1=0x55;`
	`MSDelay(250);`
	` p1=0xAA;`
	`MSDelay(250);`
	` }`
	`}`
	` void MSDelay(unsigned int itime)`
	` {`
	` unsigned int i,j;`
	`for (i=0;i <time; i++)`
	`for (j=0;j<1275;j++)`
	`}`

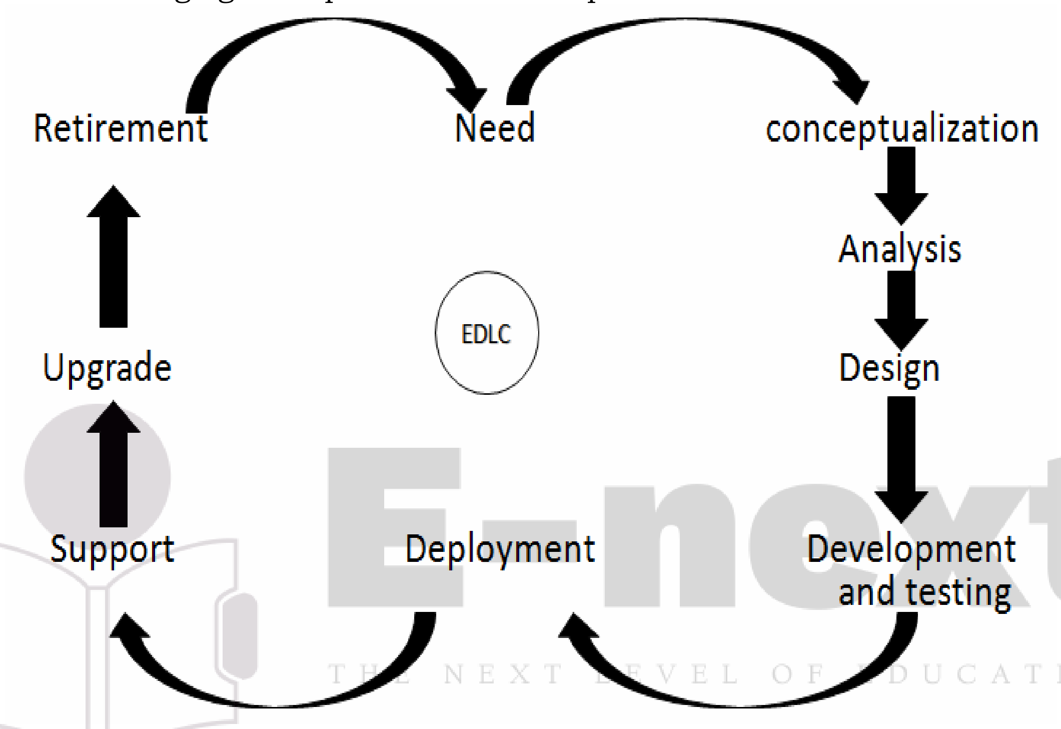
7.	**A door sensor is connected to the P1.1 pin, and a buzzer is connected to P1.7. Write an 8051 C program to monitor the door sensor, and when it opens, sound the buzzer. You can sound the buzzer by sending a square wave of a few hundred Hz.**

Ans	`#include <reg51.h>`
	`void MSDelay(unsigned int);`
	` sbit Dsensor=P1^1; sbit Buzzer=P1^7;`
	`void main(void)`
	`{`
	`Dsensor=1; //make P1.1 an input`
	`while (1)`
	`{`
	` while (Dsensor==1)//while it opens`
	`{`
	` Buzzer=0;`
	`MSDelay(200);`

	Buzzer=1;
	MSDelay(200);
	}
	}
	}

UNIT IV	
1.	**What are the Selection criteria for the Controller**
Ans	Selection of a microcontroller for any application depends on some design factors. A good designer finalizes his selection based on a comparative study of the design factors. The important factors to be considered in the selection process of a microcontroller are listed below. **Features Set:** The important queries related to the feature set are: Does the microcontroller support all the peripherals required by the application, say serial interface, parallel interface, etc.? Does it satisfy the general I/O port requirements by the application? Does the controller support sufficient number of timers and counters? Does the controller support built-in ADC/DAC hardware in case of signal processing applications? Does the controller provide the required performance? **Speed of Operation:** Speed of operation of performance of the controller is another important design factor. The number of clocks required per instruction cycle and the maximum operating clock frequency supported by the processor greatly affects the speed of operation of the controller. The speed of operation of the controller is usually expressed in terms of million instructions per second (MIPS). **Code Memory Space:** If the target processor/controller application is written in C or any other high level language, does the controller support sufficient code memory space to hold the compiled hex code (In case of controllers with internal code memory)? Data Memory Space: Does the controller support sufficient internal data memory (on chip RAM) to hold run time variables and data structures? **Development Support**: Development support is another important factor for Does the controller manufacture provide cost-effective–consideration. It deals with development tools? Does the manufacture provide product samples for prototyping and sample development stuffs to alleviate the development pains? Does the controller support third party development tools? Does the manufacture provide technical support if necessary? **Availability:** Availability is another important factor that should be taken into account for the selection process. Since the product is entirely dependent on the controller, the product development time and time to market the product solely depends on its availability. By technical terms it is referred to as Lead time. Lead

	time is the time elapsed between the purchase order approval and the supply of the product. **Power Consumption**: The power consumption of the controller should be minimal. It is a crucial factor since high power requirement leads to bulky power supply designs. The high power dissipation also demands for cooling fans and it will make the overall system messy and expensive. Controllers should support idle and power down modes of operation to reduce power consumption. **Cost:** Last but not least, cost is a big deciding factor in selecting a controller. The cost should be within the reachable limit of the end user and the targeted user should not be high tech. Remember the ultimate aim of a product is to gain marginal benefit.
2.	**Write a short note on Embedded Development cycle**
Ans	**EMBEDDED PRODUCT DEVELOPMENT LIFE CYCLE (EDLC)** EDLC is Embedded Product Development Life Cycle It is an Analysis – Design – Implementation based problem solving approach for embedded systems development. There are three phases to Product development:  Analysis involves understanding what product needs to be developed Design involves what approach to be used to build the product Implementation is developing the product by realizing the design. **Need for EDLC** EDLC is essential for understanding the scope and complexity of the work involved in embedded systems development. It can be used in any developing any embedded product EDLC defines the interaction and activities among various groups of a product development phase. Example: project management, system design **Objectives of EDLC** The ultimate aim of any embedded product in a commercial production setup is to produce Marginal benefit. Marginal is usually expressed in terms of Return On Investment. The investment for product development includes initial investment, manpower, infrastructure investment etc.

	EDLC has three primary objectives are: 1. Ensure that high quality products are delivered to user 2. Risk minimization defect prevention in product development through project management 3. Maximize the productivity DIFFERENT PHASES OF EDLC The following figure depicts the different phases in EDLC:  Retirement Need conceptualization Analysis EDLC Upgrade Design Support Deployment Development and testing
3.	**Why 8051 is most widely used in Embedded system**
Ans	8051 is a very versatile microcontroller featuring powerful Boolean processor which supports bit manipulation instructions for real time industrial control applications. The standard 8051 architecture supports 6 interrupts (2 external interrupts, 2 timer interrupts and 2 serial interrupts), two 16 bit timers/counters, 32 I/O lines and a porgammble full duplex serial interface. Another fascinating feature of 8051 is the way it handles interrupts. The interrupts have two priodity leavels and each interrupt is allocated fixed 8 bytes of code memory. This approach is very efficient in real time application. Though 8051 is invented by Intel, today it is available in the market from more than 20 vendors and with more than 100 varients of the original 8051 flavour, supporiting CAN, USB, SPI and TCP/IP interfaces, integrated ADC/DAC, LCD Controller and extended number of I/O ports. Another remarkable feautre of 8051 is its low cost. The 8051 flash microcontroller (AT89C51) from Atmel is available in the market for less than 1USS per piece. So imagine its cost for high volume purchases.

4. Write a short note on Compiling & also explain Cross Compilers

Ans

Compiling

- The process of compiling is done by the compiler.
- The compiler takes input as source code files and gives output as multiple object files.
- Compilers for embedded systems are essentially crosscompilers.

For example while compiling the programmer has to select the target processor for which the code has to be generated.

- The contents of the object files depend on its format.
- Two commonly used formats are:

1. Common Object file format (COFF)
2. Extended file format (ELF)

- Object files generally have the following structure:

header	It describes the sections that will be contained in the object file.
text	It contains all code blocks
data	It contains all initialized global variables and their values
bss	It contains all initialized global variables.

Cross-compiler

A cross-compiler is a compiler that runs on one machine and produces object code for another machine. The cross-compiler is used to implement the compiler, which is characterized by three languages:

1. The source language,
2. The object language, and
3. The language in which it is written.

If a compiler has been implemented in its own language, then this arrangement is called a "bootstrap" arrangement

5. What is Debugging? Also explain infinite loop

Ans

DEBUGGING THE EMBEDDED SOFTWARE

Debugging is the process of eliminating the bugs/errors in software.

The software written to run on embedded systems may contain errors and hence needs debugging.

However, the difficulty in case of embedded systems is to find out the bug/ error itself. This is because the binary image you downloaded on the target board was free of syntax errors but still if the embedded system does not function the way it was supposed to be then it can be either because of a hardware problem or a software problem. Assuming that the hardware is perfect all that remains to check is the software.

The difficult part here is that once the embedded system starts functioning there is no way for the user or programmer to know the internal state of the components on the target board.

The most primitive method of debugging is using LEDs. This is similar to using a printf or a cout statement in c/c++ programs to test if the control enters the loop or not. Similarly an LED blind or a pattern of LED blinks can be used to check if the control enters a particular piece of code.
There are other advanced debugging tools like;
a. Remote debugger
b. Emulator
c. Simulator

1 Remote Debuggers

a. Remote Debugger is a tool that can be commonly used for:
   - Downloading
   - Executing and
   - Debugging embedded software
b. A Remote Debugger contains a hardware interface between the host computer and the target embedded system.

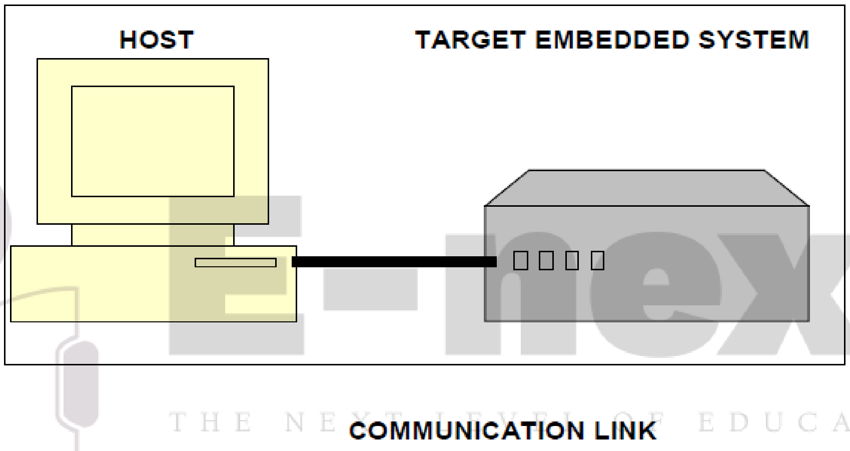


Figure: Remote Debugger

The Software interface of the remote debugger has GUI-based main window and several smaller windows for the source code, register contents and other information about the executing program.

a. It contains two pieces of software :
   - **Frontend remote debugger**
   - It runs on the host computer.
   - It provides the human interface.

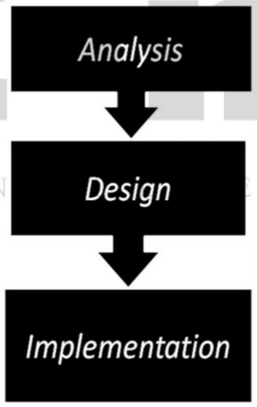
   Backend remote debugger
   - Backend remote debugger runs on the target processor.
   - It communicates with the frontend over a communications link of some sort.
   - It provides for low-level control of the target processor and is usually called the debug monitor.

   Debug monitor is a piece of software that has been designed specifically for use as a debugging tool for processors and chips.
   It is automatically started whenever the processor is reset.
   It monitors the communication link to the host computer and

	responds to requests from the remote debugger running there. One such debugger is the GNU. It was originally designed for native debugger. It performs cross-debugging. Communication between the GDB frontend and debug monitor is byte-oriented and designed for transmission over a serial connection. **Infinite Loop** • The code for every embedded program is written in an infinite loop. This is because the embedded system is supposed to run every time it is turned on till the time its power goes off or it stops functioning. • The code for blinking LED is also enclosed in an infinite loop. The functions toggle() and delay() run infinite number of times. • An application of an embedded system has an infinite loop around its code. It's just like the program you did to implement switch case where the program has to run continuously until the user selects to exit.
6.	**Explain the Linker in brief**
Ans	The process of linking is carried out by the linker The linker takes input as multiple object files and gives output as a single object file which is also called as the relocatable code The output of compiler is multiple object files. These files are incomplete in the sense that they may contain reference to variables and functions across multiple object files which need to be resolved. The job of the linker is to combine these multiple object files and resolve the unresolved symbols. The Linker does this by merging the various sections like text, data, and bss of the individual object files. The output of the linker will be a single file which contains all of the machine language code from all of the input object files that will be in the text section of this new file, and all of the initialized and uninitialized variables will reside in the new data section and bss section respectively.

UNIT V

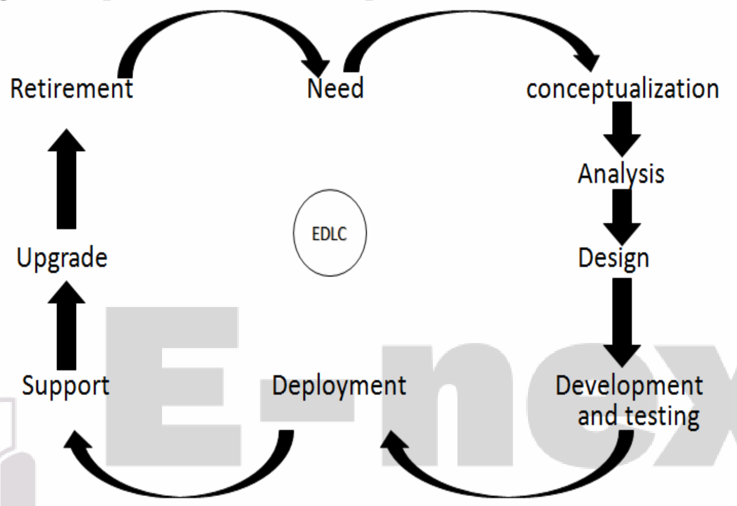
1.	**Write a short note on Kernel & also explain its types**
Ans	**What Is Kernel -** A kernel is a central component of an operating system. It acts as an interface between the user applications and the hardware. The sole aim of the kernel is to manage the communication between the software (user level applications) and the hardware (CPU, disk memory etc). The main tasks of the kernel are : • Process management • Device management • Memory management • Interrupt handling • I/O communication • File system...etc **Types Of Kernels -** **Kernels may be classified mainly in two categories** 1. Monolithic 2. Micro Kernel **1 Monolithic Kernels** Earlier in this type of kernel architecture, all the basic system services like process and memory management, interrupt handling etc were packaged into a single module in kernel space. This type of architecture led to some serious drawbacks like 1) Size of kernel, which was huge. 2)Poor maintainability, which means bug fixing or addition of new features resulted in recompilation of the whole kernel which could consume hours In a modern day approach to monolithic architecture, the kernel consists of different modules which can be dynamically loaded and un-loaded. This modular approach allows easy extension of OS's capabilities. With this approach, maintainability of kernel became very easy as only the concerned module needs to be loaded and unloaded every time there is a change or bug fix in a particular module. So, there is no need to bring down and recompile the whole kernel for a smallest bit of change. Also, stripping of kernel for various platforms (say for embedded devices etc) became very easy as we can easily unload the module that we do not want. Linux follows the monolithic modular approach **2 Microkernels** This architecture majorly caters to the problem of ever growing size of kernel code which we could not control in the monolithic approach. This architecture allows some basic services like device driver management, protocol stack, file

	system etc to run in user space. This reduces the kernel code size and also increases the security and stability of OS as we have the bare minimum code running in kernel. So, if suppose a basic service like network service crashes due to buffer overflow, then only the networking service's memory would be corrupted, leaving the rest of the system still functional. In this architecture, all the basic OS services which are made part of user space are made to run as servers which are used by other programs in the system through inter process communication (IPC). eg: we have servers for device drivers, network protocol stacks, file systems, graphics, etc. Microkernel servers are essentially daemon programs like any others, except that the kernel grants some of them privileges to interact with parts of physical memory that are otherwise off limits to most programs. This allows some servers, particularly device drivers, to interact directly with hardware. These servers are started at the system start-up.
2.	**Write a short note on EDLC**
Ans	**EMBEDDED PRODUCT DEVELOPMENT LIFE CYCLE (EDLC)** EDLC is Embedded Product Development Life Cycle It is an Analysis – Design – Implementation based problem solving approach for embedded systems development. There are three phases to Product development:  Analysis involves understanding what product needs to be developed Design involves what approach to be used to build the product Implementation is developing the product by realizing the design. **Need for EDLC** EDLC is essential for understanding the scope and complexity of the work involved in embedded systems development. It can be used in any developing any embedded product EDLC defines the interaction and activities among various groups of a product development phase. Example: project management, system design **Objectives of EDLC** The ultimate aim of any embedded product in a commercial production setup is

to produce Marginal benefit. Marginal is usually expressed in terms of Return On Investment. The investment for product development includes initial investment, manpower, infrastructure investment etc.

EDLC has three primary objectives are:
1. Ensure that high quality products are delivered to user
2. Risk minimization defect prevention in product development through project management
3. Maximize the productivity

DIFFERENT PHASES OF EDLC
The following figure depicts the different phases in EDLC:



3. **What are the selection criteria for RTOS also explain its characteristics**

Ans Different parameters for selecting the RTOS are

1. Interrupt Latency,
2. Context switching
3. Inter task Communication (Message Queue Mechanism, Signal Mechanism, Semaphores),
4. Power Management (Sleep mode, Low power mode, idle mode, Stand by mode)
5. No. of Interrupt levels
6. Kernel Size
7. Scheduling Algorithms (Round Robin Scheduling, First Come First Serve, Shortest Job First, Preemptive Scheduling etc),
8. Interrupt Levels,
9. Maintenance Fee
10. Timers
11. Priority Levels
12. Kernel Synchronization (timers, mutexes, events, semaphores etc),
13. Cost,

14. Development host,
15. Task switching time and
16. Royalty Fee

Characteristics of RTOS -
An application's requirements define the requirements of its underlying RTOS. Some of the more common attributes are

- performance,
- predictability,
- reliability,
- scalability.
- compactness, and
- scalability

These attributes are discussed next; however, the RTOS attribute an application needs depends on the type of application being built.

Reliability
Embedded systems must be reliable. Depending on the application, the system might need to operate for long periods without human intervention.
Different degrees of reliability may be required. For example, a digital solar-powered calculator might reset itself if it does not get enough light, yet the calculator might still be considered acceptable. On the other hand, a telecom switch cannot reset during operation without incurring high associated costs for down time. The RTOSes in these applications require different degrees of reliability.

Predictability
Because many embedded systems are also real-time systems, meeting time requirements is key to ensuring proper operation. The RTOS used in this case needs to be predictable to a certain degree. The term deterministic describes RTOSes with predictable behavior, in which the completion of operating system calls occurs within known timeframes.

Developers can write simple benchmark programs to validate the determinism of an RTOS. The result is based on timed responses to specific RTOS calls. In a good deterministic RTOS, the variance of the response times for each type of system call is very small.

Performance
This requirement dictates that an embedded system must perform fast enough to fulfill its timing requirements. Typically, the more deadlines to be met-and the shorter the time between them-the faster the system's CPU must be. Although underlying hardware can dictate a system's processing power, its software can also contribute to system performance. Typically, the processor's performance is expressed in million instructions per second (MIPS).

Throughput also measures the overall performance of a system, with hardware and software combined. One definition of throughput is the rate at which a

	system can generate output based on the inputs coming in. Throughput also means the amount of data transferred divided by the time taken to transfer it. Data transfer throughput is typically measured in multiples of bits per second (bps). Sometimes developers measure RTOS performance on a call-by-call basis. Benchmarks are written by producing timestamps when a system call starts and when it completes. Although this step can be helpful in the analysis stages of design, true performance testing is achieved only when the system performance is measured as a whole. **Compactness** Application design constraints and cost constraints help determine how compact an embedded system can be. For example, a cell phone clearly must be small, portable, and low cost. These design requirements limit system memory, which in turn limits the size of the application and operating system. In such embedded systems, where hardware real estate is limited due to size and costs, the RTOS clearly must be small and efficient. In these cases, the RTOS memory footprint can be an important factor. To meet total system requirements, designers must understand both the static and dynamic memory consumption of the RTOS and the application that will run on it. **Scalability** Because RTOSes can be used in a wide variety of embedded systems, they must be able to scale up or down to meet application-specific requirements. Depending on how much functionality is required, an RTOS should be capable of adding or deleting modular components, including file systems and protocol stacks. If an RTOS does not scale up well, development teams might have to buy or build the missing pieces. Suppose that a development team wants to use an RTOS for the design of a cellular phone project and a base station project. If an RTOS scales well, the same RTOS can be used in both projects, instead of two different RTOSes, which saves considerable time and money.
4.	**Write a note on Operating System**
Ans	An operating system (OS) is system software that manages computer hardware and software resources and provides common services for computer programs **Types of operating systems** **Single- and multi-tasking** A single-tasking system can only run one program at a time, while a multi-tasking operating system allows more than one program to be running in concurrency. This is achieved by timesharing, dividing the available processor time between multiple processes that are each interrupted repeatedly in time slices by a task-scheduling subsystem of the operating system. Multi-tasking may be characterized in preemptive and co-operative types. In preemptive

multitasking, the operating system slices the CPU time and dedicates a slot to each of the programs. Unix-like operating systems, e.g., Solaris, Linux, as well as AmigaOS support preemptive multitasking. Cooperative multitasking is achieved by relying on each process to provide time to the other processes in a defined manner. 16- bit versions of Microsoft Windows used cooperative multi-tasking. 32-bit versions of both Windows NT and Win9x, used preemptive multi-tasking.

Single- and multi-user

Single-user operating systems have no facilities to distinguish users, but may allow multiple programs to run in tandem. A multi-user operating system extends the basic concept of multi-tasking with facilities that identify processes and resources, such as disk space, belonging to multiple users, and the system permits multiple users to interact with the system at the same time. Time-sharing operating systems schedule tasks for efficient use of the system and may also include accounting software for cost allocation of processor time, mass storage, printing, and other resources to multiple users.

Distributed

A distributed operating system manages a group of distinct computers and makes them appear to be a single computer. The development of networked computers that could be linked and communicate with each other gave rise to distributed computing. Distributed computations are carried out on more than one machine. When computers in a group work in cooperation, they form a distributed system.

Templated

In an OS, distributed and cloud computing context, templating refers to creating a single virtual machine image as a guest operating system, then saving it as a tool for multiple running virtual machines. The technique is used both in virtualization and cloud computing management, and is common in large server warehouses.

Embedded

Embedded operating systems are designed to be used in embedded computer systems. They are designed to operate on small machines like PDAs with less autonomy. They are able to operate with a limited number of resources. They are very compact and extremely efficient by design. Windows CE and Minix 3 are some examples of embedded operating systems.

Real-time

A real-time operating system is an operating system that guarantees to process events or data by a specific moment in time. A real-time operating system may be single- or multitasking, but when multitasking, it uses specialized scheduling algorithms so that a deterministic nature of behavior is achieved. An event-driven system switches between tasks based on their priorities or external events while time-sharing operating systems switch tasks based on clock interrupts

	Library A library operating system is one in which the services that a typical operating system provides, such as networking, are provided in the form of libraries and composed with the application and configuration code to construct a unikernel: a specialized, single address space, machine image that can be deployed to cloud or embedded environments.
5.	**Differentiate between a Dicompiler & Disambler**
Ans	DISASSEMBLER/DECOMIPILER A Disassembler/ Decomipiler is a reverse engineering tool. Reverse Engineering is used in embedded system to find out the secret behind the working of a proprietary product. A DISASSEMBLER is a utility program which converts machine codes into target processor specific assembly code/instruction. The process of converting machine codes to assembly code is called disassembling. A DECOMIPILER is a utility program for translating machine codes into corresponding high level language instruction. A decompiler performs the reverse operation of a compiler/cross-compiler.
6.	**Write Industry trends in Embedded Development**
Ans	**Following are the major trends in processor architecture in embedded development. System on Chip (SoC)** This concept makes it possible to integrate almost all functional systems required to build an embedded product into a single chip. SoC are now available for a wide variety of diverse applications like Set Top boxes, Media Players, PDA, etc. SoC integrate multiple functional components on the same chip thereby saving board space which helps to miniaturize the overall design. Multicore Processors/ Chiplevel Multi Processor This concept employs multiple cores on the same processor chip operating at the same clock frequency and battery. Based on the number of cores, these processors are known as: Dual Core – 2 cores Tri Core – 3 cores Quad Core – 4 cores These processors implement multiprocessing concept where each core implements pipelining and multithreading.

Reconfigurable Processors
It is a processor with reconfigurable hardware features.

Depending on the requirement, reconfigurable processors can change their functionality to adapt to the new requirement. Example: A reconfigurable processor chip can be configured as the heart of a camera or that of media player.

These processors contain an Array of Programming Elements (PE) along with a microprocessor. The PE can be used as a computational engine like ALU or a memory element.

Operating System Trends
The advancements in processor technology have caused a major change in the Embedded Operating System Industry.

There are lots of options for embedded operating system to select from which can be both commercial and proprietary or Open Source.

Virtualization concept is brought in picture in the embedded OS industry which replaces the monolithic architecture with the microkernel architecture.

This enables only essential services to be contained in the kernel and the rest are installed as services in the user space as is done in Mobile phones.

Off the shelf OS customized for specific device requirements are now becoming a major trend.

Development Language Trends
There are two aspects to Development Languages with respect to Embedded Systems Development

Embedded Firmware
It is the application that is responsible for execution of embedded system.
It is the software that performs low level hardware interaction, memory management etc on the embedded system.

Embedded Software
It is the software that runs on the host computer and is responsible for interfacing with the embedded system.

It is the user application that executes on top of the embedded system on a host computer.

Early languages available for embedded systems development were limited to C & C++ only. Now languages like Microsoft C$, ASP.NET, VB, Java, etc are available.

	Java Java is not a popular language for embedded systems development due to its nature of execution. Java programs are compiled by a compiler into bytecode. This bytecode is then converted by the JVM into processor specific object code. During runtime, this interpretation of the bytecode by the JVM makes java applications slower that other cross compiled applications. This disadvantage is overcome by providing in built hardware support for java bytecode execution.