S.Y. B.Sc. (IT) : Sem. IV

Embedded Systems

Time : 2½ Hrs.] Prelim Question Paper Solution **[Marks : 75**

Q.1 Attempt the following (any THREE) [15]

Q.1(a) Compare Embedded Systems and general purpose computer systems. [5]

Ans.:

Criteria	Embedded System	General purpose computer system
Contents	It is combination of special purpose hardware and embedded OS for executing specific set of applications	It is combination of generic hardware and a general purpose OS for executing a variety of applications.
Operating System	It may or may not contain operating system	It contains general purpose operating system
Alterations	Applications are non-alterable by the user.	Applications are alterable by the user.
Key factor	Application specific requirements are key factors.	Performance" is key factor
Power Consumption	Less	More
Response Time	Critical for some applications	Not Critical

Q.1(b) Give the classifications of Embedded Systems. [5]

Ans.: Classifications of Embedded Systems.

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

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

- **On generation**

  First generation (1G):
  - Built around 8bit microprocessor & microcontroller.
  - Simple in hardware circuit developed.

  Second generation (2G):
  - Built around 16-bit µp & 8-bit µc.
  - They are more complex & powerful.

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

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

- **On complexity & performance**

  Small-scale:
  - Simple in application need
  - Performance not time-critical.
  - Built around low performance & low cost 8 or 16 bit µp/µc.
  - Example: an electronic toy

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

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

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

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

Q.1(c) Explain cots. [5]

Ans.:
- A Commercial off the Shelf product is one which is used 'asis'.
- The COTS components itself may be develop around a general purpose or domain specific processor or an ASICs or a PLDs.
- 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
- 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.
- TCP/IP plug in module is an example of COTS products.
- This network plug-in module gives TCP/IP connectivity to the system you are developing.
- There is no need to design this module yourself and write a firmware for it.
- Everything will be readily supplied by the COTS manufacturer.
- What you need to do is identify the COTS for your system and give plug-in option on your board according to the hardware plug-in connections given in the specifications of the COTS.

Advantages of COTS:
- Ready to use
- Easy to integrate
- Reduces development time

Disadvantages of COTS:
- No operational or manufacturing standard (all proprietary)
- Vendor or manufacturer may discontinue production of a particular COTS product

Q.1(d) Write a note on application specific ICs. (IC technology)/Programmable logic devices. [5]

Ans.:
- Processors vary in their customization for the problem at hand
- Depending on that it can be general purpose processor, application specific processor or special purpose processor.

- The manner in which a digital (gate-level) implementation is mapped onto an IC
  - IC: Integrated circuit, or "chip"
  - IC technologies differ in their customization to a design
  - IC's consist of numerous layers
- Three types of IC technologies
  - Full-custom/VLSI
  - Semi-custom ASIC (gate array and standard cell)
  - PLD (Programmable Logic Device)
- All layers are optimized for an embedded system's particular digital implementation
  - Placing transistors
  - Sizing transistors
  - Routing wires
- Benefits
  - Excellent performance, small size, low power
- Drawbacks
  - High NRE cost, long time-to-market
- Lower layers are fully or partially built
  - Designers are left with routing of wires and maybe placing some blocks
- Benefits
  - Good performance, good size, less NRE cost than a full-custom implementation Drawbacks
  - Still require weeks to months to develop

Programmable logic devices.

- 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.

CPLD (Complex Programmable Logic Device):

- CPLDs offer much smaller amount of logic up to 1000 gates.

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
- All layers already exist
  - Designers can purchase an IC
  - Connections on the IC are either created or destroyed to implement desired functionality
- Drawbacks
  - Bigger, more power, slower

Q.1(e) Explain the non quality attributes of an embedded system. **[5]**

Ans.: (i) 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

(ii) 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.

(iii) 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.

(iv) 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.

(v) 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.

Q.1(f) Compare RISC and CISC controllers **[5]**

Ans.:

	RISC	**CISC**
Meaning	Reduced Instruction Set Computing	Complex Instruction Set Computing
Number of instructions	Lesser	Greater
Instruction pipelining	Exist	Does not exist
Number of registers	Large	Very limited
Length of instruction	Fixed	Variable
Instructions for one task	Needs many	Needs few

Architecture used	Harvard	Harvard or von-neumann
Silicon usage	Less	More
Operations performed on	Registers only. (only load and store works with memory)	Registers or memory.

Q.2 Attempt the following (any THREE) [15]

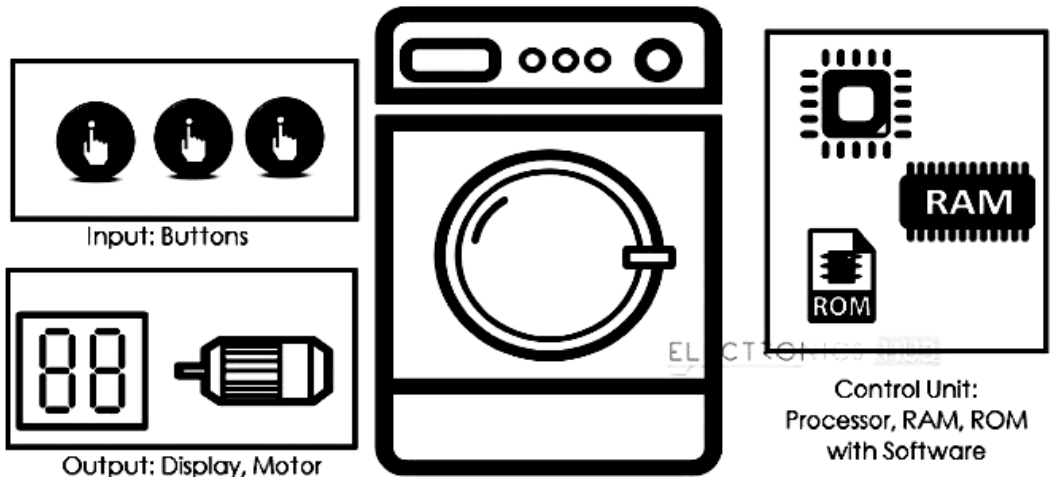
Q.2(a) Explain application specific embedded system. [5]

Ans.: Washing Machine

Let us see the important parts of the washing machine; this will also help us understand the working of the washing machine:

- **Water inlet control valve**: Near the water inlet point of the washing there is water inlet control valve. When you load the clothes in washing machine, this valve gets opened automatically and it closes automatically depending on the total quantity of the water required. The water control valve is actually the solenoid valve.
- **Water pump**: The water pump circulates water through the washing machine. It works in two directions, re-circulating the water during wash cycle and draining the water during the spin cycle.

Embedded System Example : Washing Machine



Pasts of a Washing Machine

- **Tub**: There are two types of tubs in the washing washing machine: inner and outer. The clothes are loaded in the inner tub, where the clothes are washed, rinsed and dried. The inner tub has small holes for draining the water. The external tub covers theinner tub and supports it during various cycles of clothes washing.
- ***Agitator or rotating disc***: The agitator is located inside the tub of the washing machine. It is the important part of the washing machine that actually performs the cleaning operation of the clothes. During the wash cycle the agitator rotates continuously and produces strong rotating currents within the water due to which the clothes also rotate inside the tub. The rotation of the clothes within water containing the detergent enables the removal of the dirt particles from the fabric of the clothes. Thus the agitator produces most important function of rubbing the clothes with each other as well as with water.

  In some washing machines, instead of the long agitator, there is a disc that contains blades on its upper side. The rotation of the disc and the blades produce strong currents within the water and the rubbing of clothes that helps in removing the dirt from clothes.

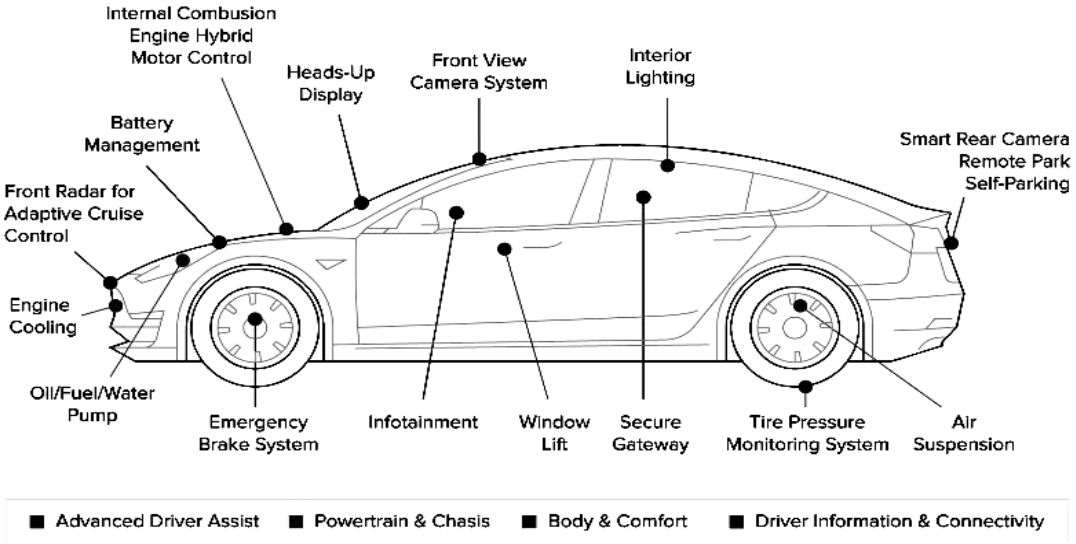
Motor of the washing machine: The motor is coupled to the agitator or the disc and produces it rotator motion. These are multispeed motors, whose speed can be changed as per the requirement. In the fully automatic washing machine the speed of the motor i.e. the agitator changes automatically as per the load on the washing machine.

Timer: The timer helps setting the wash time for the clothes manually. In the automatic mode the time is set automatically depending upon the number of clothes inside the washing machine.

Printed circuit board (PCB): The PCB comprises of the various electronic components and circuits, which are programmed to perform in unique ways depending on the load conditions (the condition and the amount of clothes loaded in the washing machine). They are sort of artificial intelligence devices that sense the various external conditions and take the decisions accordingly. These are also called as fuzzy logic systems. Thus the PCB will calculate the total weight of the clothes, and find out the quantity of water and detergent required, and the total time required for washing the clothes. Then they will decide the time required for washing and rinsing. The entire processing is done on a kind of processor which may be a microprocessor or microcontroller.

Q.2(b) Explain domain specific embedded system. [5]

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)

Q.2(c) What is memory map. **[5]**

Ans.: 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.

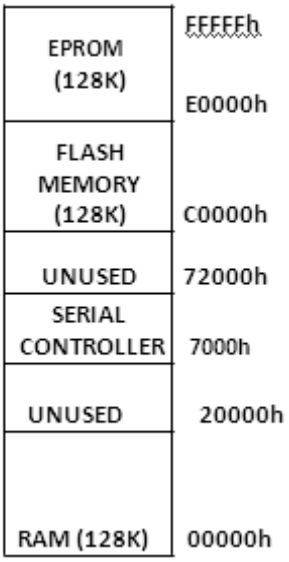
Organize the table such that the lowest address is at the bottom and the highest address is at the top. Each time a new device is added, add it to the memory map, place it in its approximate location in memory and label the starting and ending addresses, in hexadecimal. After inserting all of the devices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

Q.2(d) Short note on memory used in embedded system. **[5]**

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.

Q.2(e) What is memory testing and its purpose and control and status registers. [5]

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.

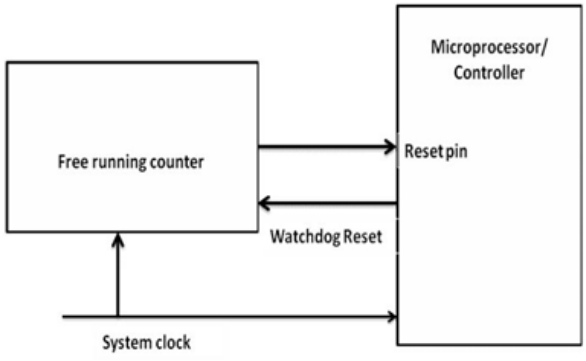
Q.2(f) What is watchdog timer and control and status registers. [5]

Ans.: 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



Control and status registers
Control and status registers are the basic interface between and embedded processor and peripheral device.

These registers are a part of peripheral hardware and their location size and individual meanings are feature of the peripheral.

For example, The registers vary from device to device: example the registers within a serial controller are very different from those in a timer.

Depending upon the design of the processor and target board, peripheral devices are located either in the processor's memory space or within the I/O space.

It is common for Embedded Systems to include some peripherals of each type. These are called Memory-Mapped and I/O-mapped peripherals.

Of the two types, memory-mapped peripherals are generally easier to work with and are increasingly popular.

Memory-mapped control and status registers can be used just like ordinary variables.

Q.3 Attempt the following (any THREE) [15]

Q.3(a)List down features of 8051 microcontroller and compare 8051 family members. [5]

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.

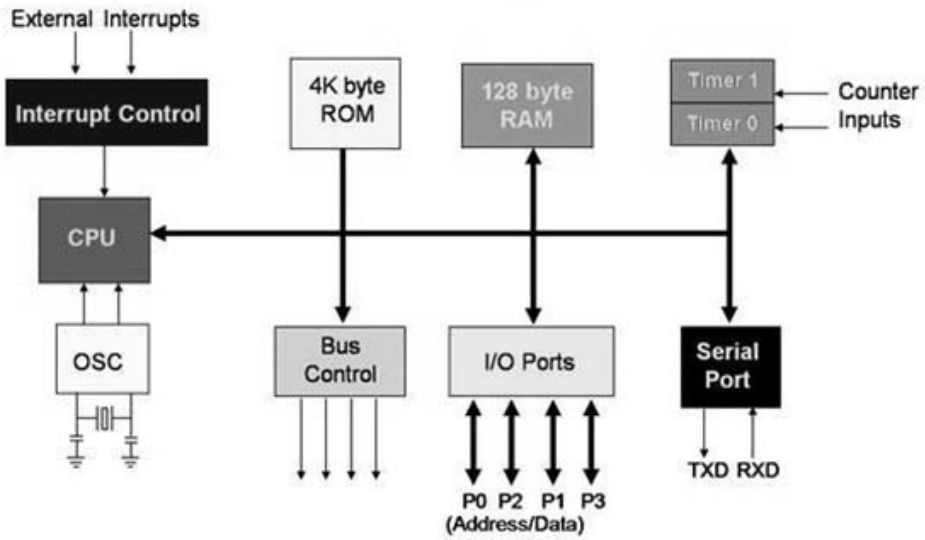
Comparison between 8051 family members

The following table compares the features available in 8051, 8052, and 8031.

Feature	8051	8052	8031
ROM(bytes)	4K	8K	0K
RAM(bytes)	128	256	128
Timers	2	3	2
I/O pins	32	32	32
Serial port	1	1	1
Interrupt sources	6	8	6

Q.3(b)Draw block diagram of 8051 microcontroller. [5]

Ans.: The following illustration shows the block diagram of an 8051 microcontroller



Q.3(c)Explain flag bits and PSW register. **[5]**

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.

Q.3(d)Write an 8051 C program to toggle bits of P1 ports continuously with a 250 ms. **[5]**

Ans.:

```
#include <reg51.h>
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<itime;i++)
for (j=0;j<1275;j++);
}
```

Q.3(e) 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. **[5]**

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);
}
}
}
```

Q.3(f) Write an 8051 C program to convert packed BCD 0×29 to ASCII and display the bytes on P1 and P2. **[5]**

Ans.:

```
#include <reg51.h>
void main(void)
{
unsigned char x,y,z;
unsigned char mybyte=0x29;
x=mybyte&0x0F;
P1=x|0x30;
y=mybyte&0xF0;
y=y>>4;
P2=y|0x30;
}
```

Q.4 Attempt the following (any THREE) **[15]**

Q.4(a) What are the selection criteria for controller. **[5]**

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.

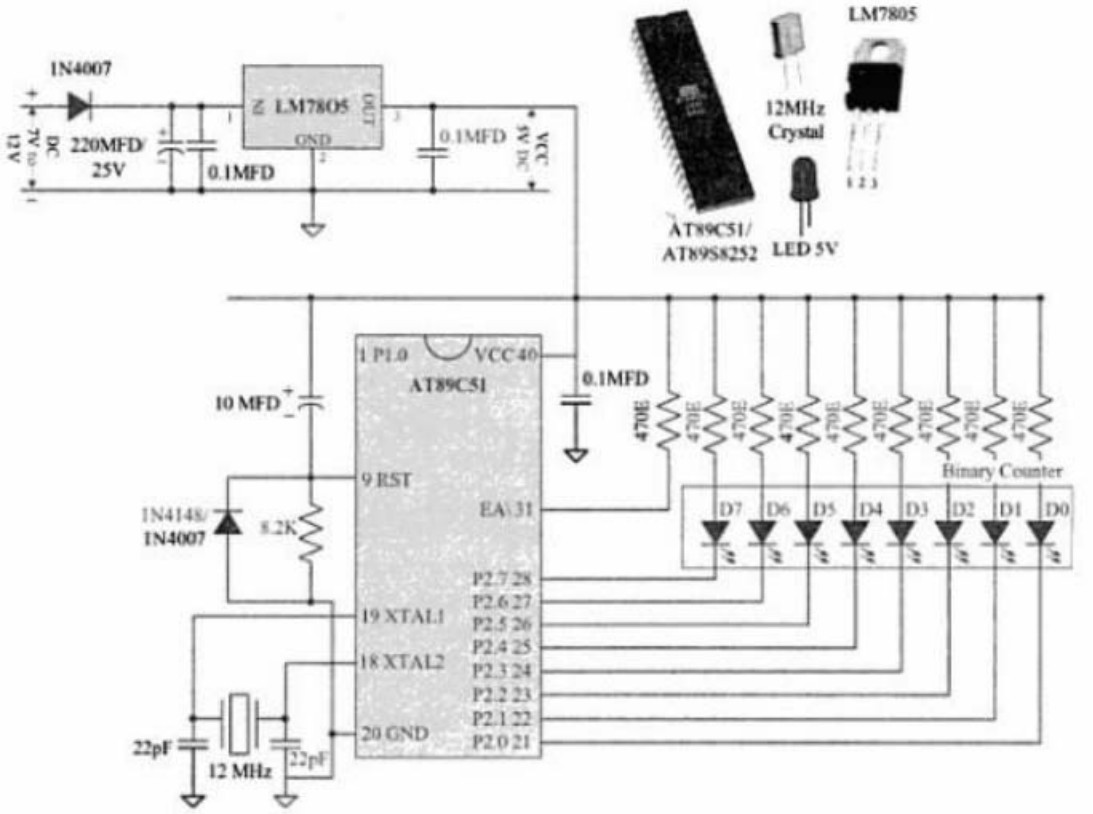
- **Features Set:** The important queries related to the feature set are: Does the microcontroller support all the peripherals required by the application, say serial interface, parallel interface, etc.? Does it satisfy the general I/O port requirements by the application? Does the controller support 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 consideration. It deals with–Does the controller manufacture provide cost-effective 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.

Q.4(b)Why to select 8051 microcontroller. **[5]**

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.

Q.4(c)Draw 8051 microcontroller with interface of 8 LED. **[5]**

Ans.:



Q.4(d)What is Debugging the Embedded Software and infinite loop. **[5]**

Ans.: 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;

- Remote debugger
- Emulator
- Simulato

Q.4(e)What is Build process in embedded system. **[5]**

Ans.: **Definition:** The process which converts source code to executable code is called as the build process.

The build process for embedded systems is different. This is because the code to be run on an embedded system is written one platform i.e. general purpose computer and executed on another platform i.e. the target hardware.

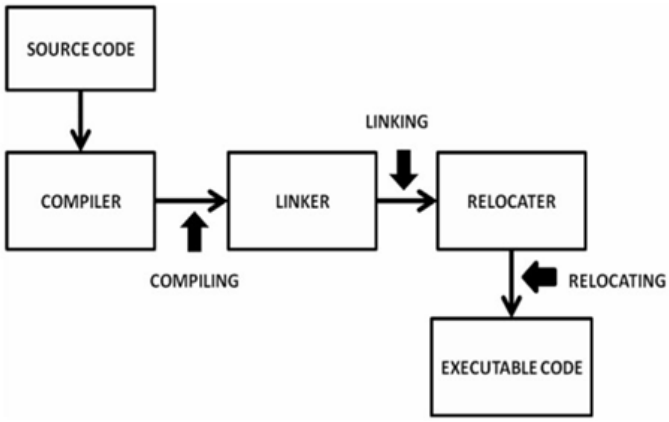
An Embedded system would also use tools such as a Compiler, Linker, Locater and Debugger to perform the entire build process. These tools would be a part of a larger IDE.

A compiler which produces the executable code to be run on a different platform is called a cross-compiler; else it is called a native compiler.

Example: Turbo C++ is a native compiler. The compiler in case of embedded systems development is a cross compiler.

The build process involves three steps:

Compiling
Linking
Locating



Q.4(f) What is infinite loop, compiling and linking. **[5]**

Ans.: 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.

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 cross-compilers. 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.

Linking

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.

Q.5 Attempt the following (any THREE) [15]

Q.5(a) What are the selection criteria for RTOS [5]

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

Q.5(b) What are key characteristics of an RTOS [5]

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

- reliability,
- predictability,
- performance,
- 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.

Q.5(c) What operating system and it's types **[5]**

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 time-sharing, 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 multi-tasking, 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.

Q.5(d) What is Emulator/Simulator/Disassembler/Decomipiler **[5]**

Ans.: Emulators

The terms simulators and emulators are very confusing but their basic functionality is the same i.e. to debug the code. There is a difference in which this is achieved by both the tools.

A simulator is a utility program that duplicates the target CPU and simulates the features and instructions supported by target CPU whereas an Emulator is a self contained hardware device which emulates the target CPU.

The Emulator hardware contains the necessary emulation logic and is connected to the debugging application that runs on the host PC.
The Simulator 'simulates' while the Emulator 'emulates'

Simulator

Simulators are used for embedded firmware debugging.
Simulator simulates the target hardware, while the code execution can be inspected.
Simulators have the following characteristics which make them very much favorable:
Purely software based
No need of target system (hardware)
Support only for basic operations
Cannot Support or lack real time behavior

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.**

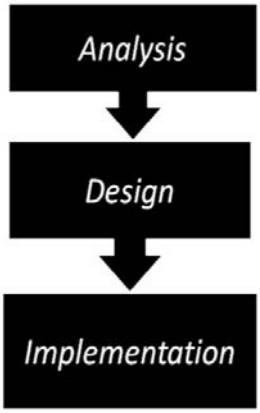
A **DECOMIPILER** is a utility program for translating machine codes into corresponding high level language instruction.

A decompiler performs the reverse operation of a compiler/cross-compiler.

Q.5(e) Short note on EDLC **[5]**

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

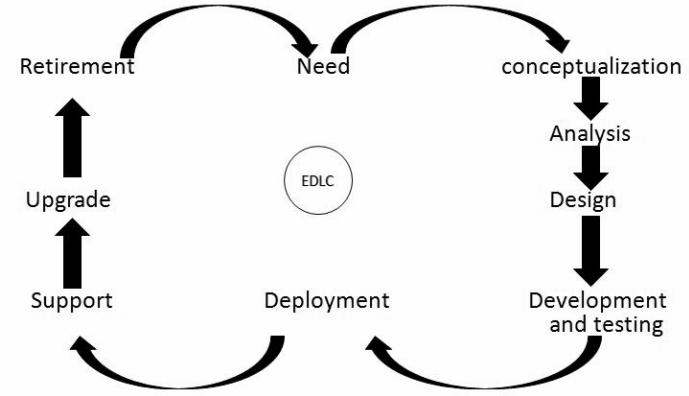
The ultimate aim of any embedded product in a commercial production setup is to produce Marginal benefit. Marginal is usually expressed in terms of Return On Investment. The investment for product development includes initial investment, manpower, infrastructure investment etc.

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:



Q.5(f) What are the treads in embedded industry. [5]

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.

❑ ❑ ❑ ❑ ❑