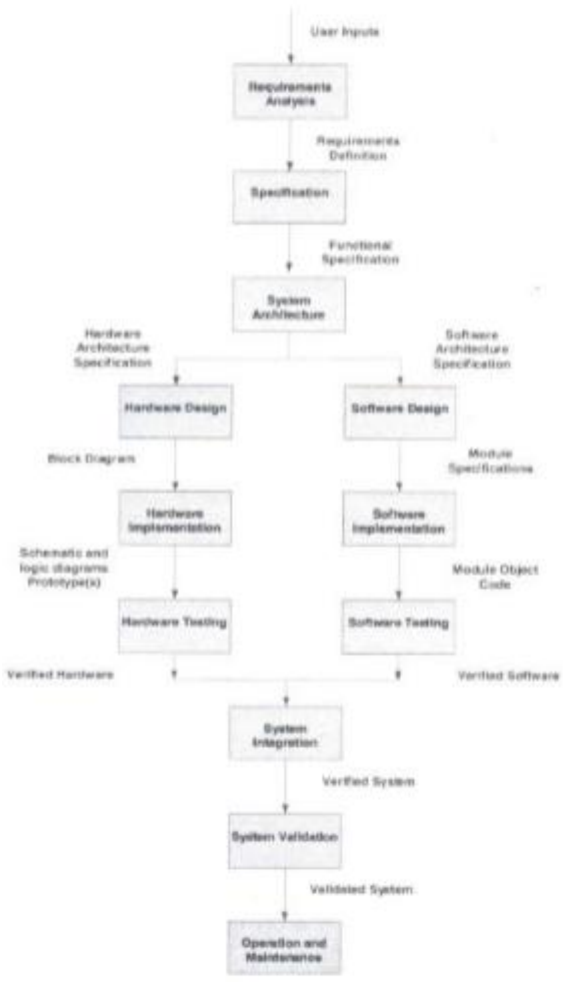
IAT-1 Solution

1 a) Major elements of an Embedded system design and development process.



The traditional design approach has been traverse the two sides of the accompanying diagram separately, that is,

- Design the hardware components
- Design the software components.
- Bring the two together.
- Spend time testing and debugging the system.

The major areas of the design process are:

- Ensuring a sound software and hardware specification.
- Formulating the architecture for the system to be designed.
- Partitioning the h/w and s/w.
- Providing an iterative approach to the design of h/w and s/w.

The important steps in developing an embedded system are:

- Requirement definition.
- System specification.
- Functional design
- Architectural design
- Prototyping.

The major aspects in the development of embedded applications are :

- Digital hardware and software architecture
- Formal design , development, and optimization process.
- Safety and reliability.
- Digital hardware and software/firmware design.
- The interface to physical world analog and digital signals.
- Debug, troubleshooting and test of our design.

1 b)

Purpose of a watchdog timer: It is a special type of timer used in high end embedded applications. Periodically the microprocessor must reset the timer at definite intervals of time. If due to any fault in the system, if the timer is not reset by the microprocessor, then the timer resets the entire system, thereby solving the problem.

CISC: Large number of complex instructions. Instructions are of variable number of bytes. It is prominent on Hardware. It has high cycles per second. Less registers, more addressing modes.

RISC: Smaller number of instructions. Instructions are of fixed number of bytes. Prominent on software. Low cycles per second .Uses more registers, fewer addressing modes.

2a)

Steps involved in the implementation of microprocessor based embedded system:

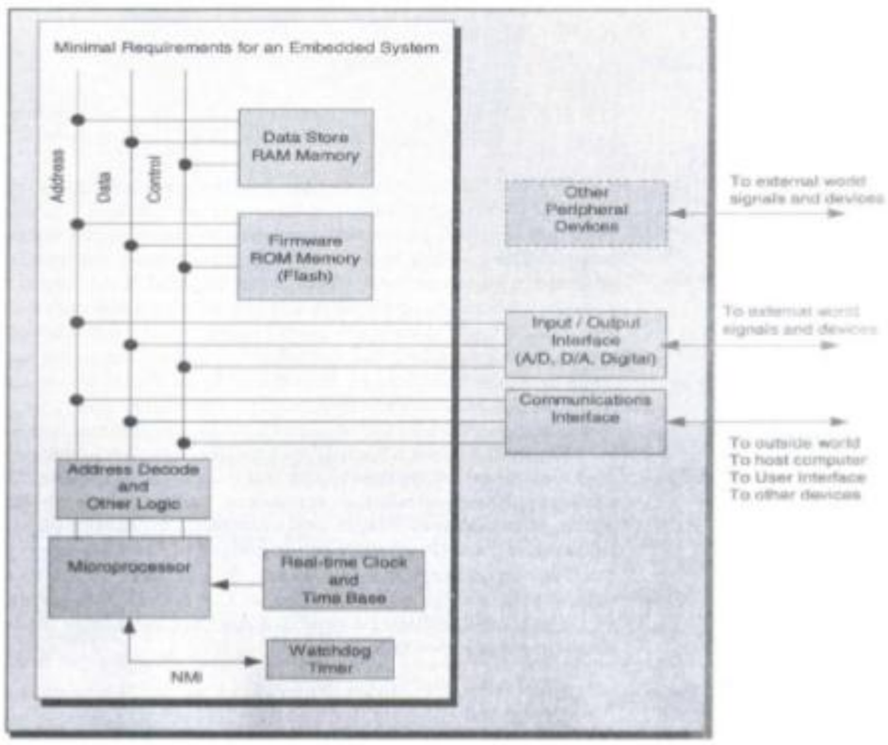
The microprocessor controls the whole system by executing a set of instructions call firmware that is stored in ROM.

An instruction set, or instruction set architecture (ISA), is the part of the computer architecture related to programming, including the native data types, instructions, registers, addressing modes, memory architecture, interrupt and exception handling, and external I/O.

An ISA includes a specification of the set of op-codes (machine language), and the native commands implemented by a particular processor.

To run the application, when power is first turned ON, the microprocessor addresses a predefined location and fetches, decodes, and executes the instruction one after the other.

The implementation of a microprocessor based embedded system combines the individual pieces into an integrated whole as shown in Figure, which represents the architecture for a typical embedded system and identifies the minimal set of necessary components.


Minimal Requirements for an Embedded System
Address
Data
Control
Data Store RAM Memory
Firmware ROM Memory (Flash)
Other Peripheral Devices
To external world signals and devices
Input / Output Interface (A/D, D/A, Digital)
To external world signals and devices
Communications Interface
To outside world
To host computer
To User Interface
To other devices
Address Decode and Other Logic
Microprocessor
Real-time Clock and Time Base
Watchdog Timer
NMI

2 b)

Embedded system: It is an electro-mechanical system designed to perform a specific function. It is a combination of embedded software and Computer hardware.

Embedded system techniques allow us to make products that are smaller, faster , more reliable and cheaper.

Hard real-time systems: Missing a single deadline leads to a complete or catastrophic system failure. Eg : Atomic reactor, Automobiles.

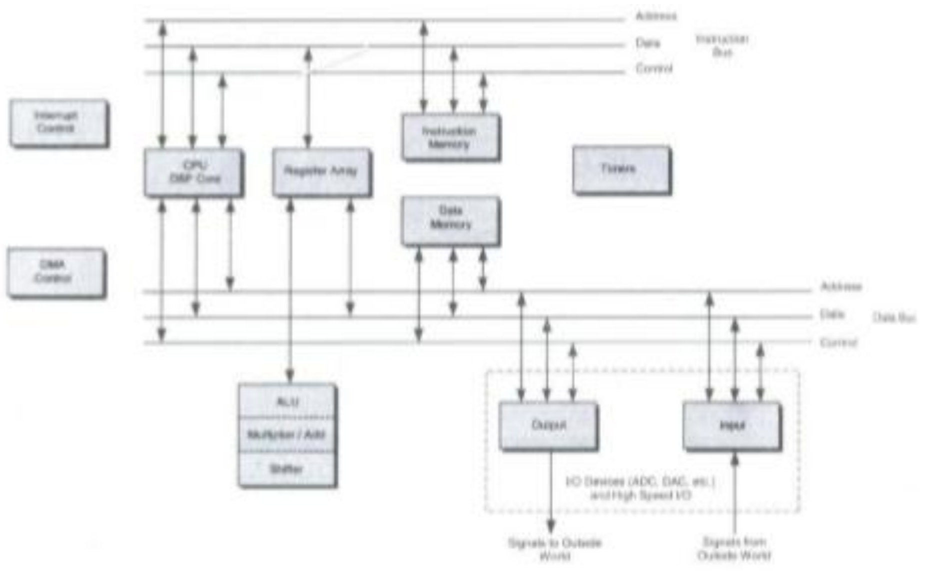
Soft real-time systems: Performance is degraded, but system is not destroyed if deadlines are not met sometimes. Eg : Video games.

Firm real time : Few missed deadlines will not lead to a total failure, but missing more than a few deadlines may lead to complete system failure. Eg: weather forecast

3 a)

Architecture of a DSP: A digital signal processor (DSP) is a specialized microprocessor with an architecture optimized for the operational needs of digital signal processing. A DSP provides fast, discrete time, signal-processing instructions. It has Very Large Instruction Word (VLIW) processing capabilities; it processes Single Instruction Multiple Data (SIMD) instructions fast; it processes Discrete Cosine Transformations (DCT) and inverse DCT

(IDCT) functions fast. The latter are a must for fast execution of the algorithms for signal analyzing, coding, filtering, noise cancellation, echo-elimination, compressing and decompressing, etc. To support high speed arithematic, the device will often implement a multiply-accumulate (MAC) primitive m which a multiply and add to the accumulator is performed in a single operation, which is useful in matrix operations.



3 b)

Four major categories of execution flow:

i) **Sequential** : It described the fundamental movement through a program. Each instruction contained in the program is executed in sequence, one after another.
A significant amount of the total code in an application is evaluated and executed in a sequential order.
Eg: a=10;
    b=20;
    c=a+b;

ii) **Branch** : This construct terminates a sequential flow of control with a decision point. At such a point, one of the several alternate paths for continued execution is taken based on the outcome of a test on some condition.
The branch construct is used to implement an *if else*, *switch* or *case* statement.
The conditional information is temporarily held as a collection of bits in a *flag register* or *condition code* register.
Eg: if (a==b)
        c=d+e;
else
        c=d-e;

iii) **Loop** : This construct permits the designer to repeatedly execute a set of instructions either forever or until some condition is met. The decision to evaluate the loop can be made before the loop is entered (entry conditionloop) or after the body of the loop is evaluated (exit condition loop).
The loop type of construct is seen in *do, repeat, while* or *for* statements.

```
Eg:  While (myVar <10)
       {
          index = index +2;
          myVar ++;
        }
```

iv) **Procedure or function call**: It is the most complex of the flow of control constructs. Such invocation requires that the control flow leave the current context, execute a set of instructions, and then return to the original context.
CALL and RET instructions are used to implement a subroutine or a function call.

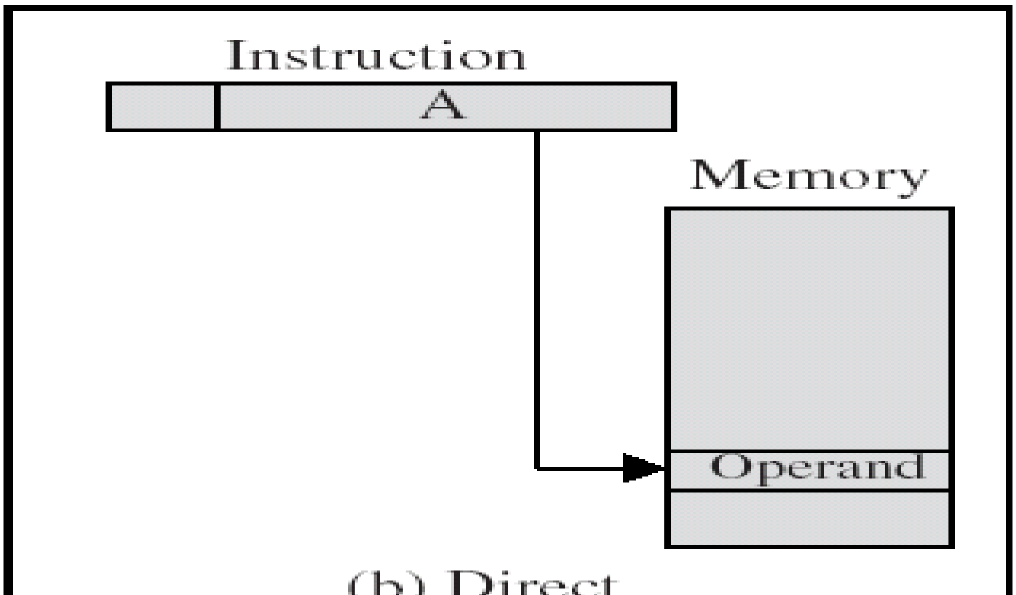
4 a)

i) Direct addressing mode: In this mode, the address of the operand is given in the instruction itself. Address field contains address of operand. Single memory reference to access data. No additional calculations to work out effective address. Limited address space

Effective address EA = address field (A)

Eg: ADD A, 05h

Look into the address 05h, fetch the value present in that address and add it to the accumulator.



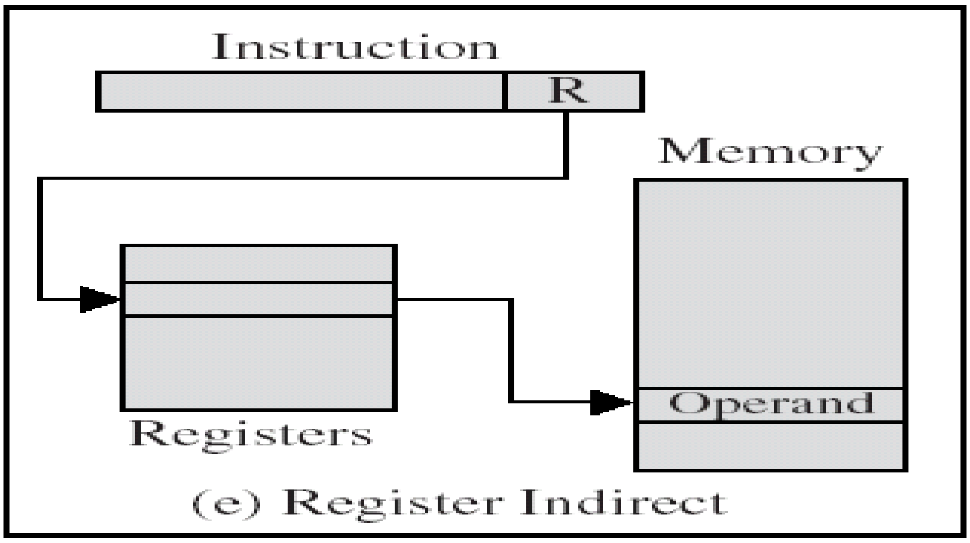
ii) Register indirect addressing mode: In this mode, the register contains the address of the operand. It provides the means to easily implement pointer-type operations that are commonly used in C and C++.

The major disadvantage of indirect addressing is that an additional memory access is necessary to retrieve the operand's value

EA = (R) : The effective **address** of the operand is the contents of a register or main memory location, location whose **address** appears in the instruction. Indirection is noted by placing the name of the register or the memory **address** given in the instruction in parentheses.

Eg: MOV R0, @R1

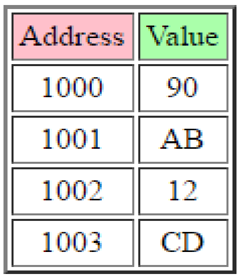
@ acts as a pointer to memory locations



Timing diagram for serial write operation using a register: Refer to figure 1.49 from text book

4 b) **Big Endian**: In big endian, the most significant byte is stored in the smallest address.

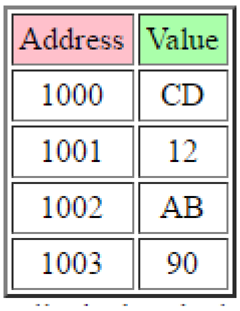
Eg: 90AB12CD



Address	Value
1000	90
1001	AB
1002	12
1003	CD

Little Endian : In little endian, the *least* significant byte is stored in the smallest address.

Eg: 90AB12CD



Address	Value
1000	CD
1001	12
1002	AB
1003	90

Truncation error: Truncation errors are those that result from using an approximation in place of an exact mathematical procedure. Range of truncation error is $-2^{-n} < E_t \leq 0$.

Example: Approximation to a derivative using a finite difference equation, Taylor's series

Rounding error: Rounding is an alternative to truncation, where the last digit is "adjusted" to give a more accurate representation of the number. Range of Rounding error is

$$-1/2\ 2^{-n} < E_R \leq ½\ 2^{-n}\ .$$

- An example of rounding error, consider the speed of light in a vacuum. The official value is 299,792,458 meters per second.
- In scientific (power-of-10) notation, that quantity is expressed as 2.99792458 x 10^8. Rounding it to three decimal places yields 2.998 x 10^8.
- The rounding error is the difference between the actual value and the rounded value, in this case (2.998 - 2.99792458) x 10^8, which works out to 0.00007542 x 10^8.
- Expressed in the correct scientific notation format, that value is 7.542 x 10^3, which equals 7542 in plain decimal notation.

5 a)

Arity: The number of operands that an instruction operates on at any time is called the **Arity** of an instruction.

- **One address instruction**: If the arity of an instruction or the number of operands in an instruction is one, it is called One address/operand instruction. Eg : ++x, x--.
- **Two address instruction** : If the arity of an instruction or the number of operands in an instruction is two, it is called Two address/operand instruction. Eg : x=y, x=x+y
- **Three address instruction** : If the arity of an instruction or the number of operands in an instruction is three, it is called Three address/operand instruction. Eg : z=x+y.

5 b) Propogation of Error: Refer page no- 11 to 12 along with the example.

6 a)

A **stack** is a data structure that occupies an area in memory. Its size is fixed and supports several operations. Individual items can be added and stored in a **stack** using a **push** operation. Objects can be retrieved using a **pop** operation, which removes an item from the **stack**.

Different operations that can be performed with stack are PUSH and POP

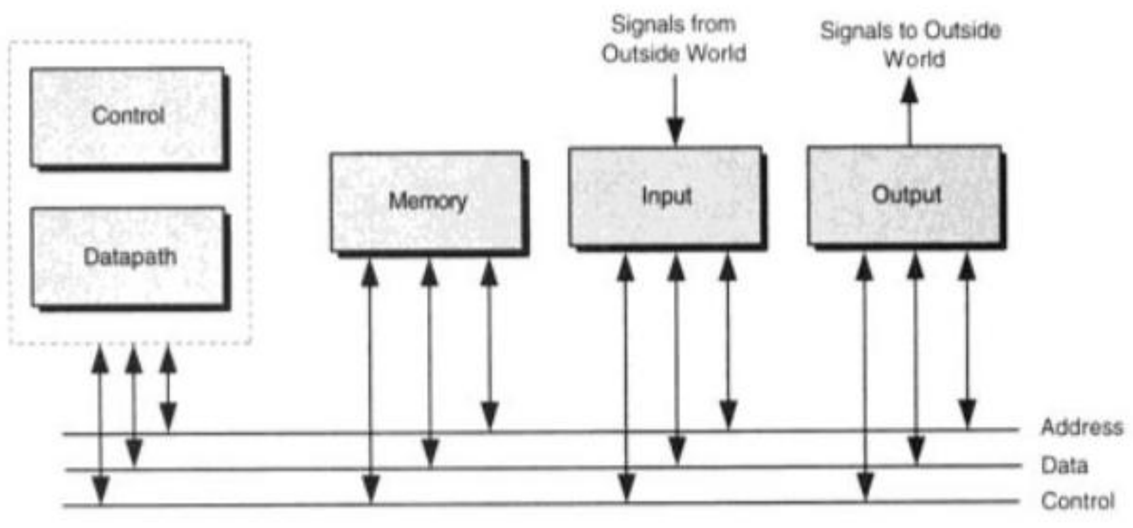
PUSH - Instruction used to enter data into the stack. It increments the address that is held by the stack pointer to refer to the next empty spot and then writes the data to be stored into the address in memory designated by that address.

POP - Instruction used to remove data from the stack. It decrements the address that is held by the stack pointer to refer to the next lower address. The retrieved value is returned as the result of the pop operation.

Stack pointer:

Entry and removal of data from just one end of the stack is impractical because of the time burden in moving every piece of data each time a new entry is made A more practical implementation adds or removes data at the open end of the structure. The memory address reflecting the current top of the stack is remembered and modified after each addition and removal. Such an address is called stack pointer. It is a small register that stores the address of the last program request in a **stack** Refer fig 1.38 in the text.

6 b)



The datapath and control block, more commonly known as CPU coordinates the activities of the system as well as performs computations and data manipulation operations necessary to execute the application.

We move signals into, out of, or throughout the system an paths called buses. Signals flowing on the wires making up the busses are classified into three major categories: ***Address, Data and Control.***

The data are the key signals that are being moving around; the address signals identify where the data is coming from and where it is going to; the control signals specify and coordinate how the data is transported.

7 a) **Op-code**: Once the number of instructions that a microprocessor supports is determined, then each instruction is assigned a unique code, such code is called the operational code. It specifies what operation is to be performed by the central processing unit (CPU).

Entry conditional loop : The decision to execute the loop is made at the beginning of the loop. The loop is not executed at all if the condition is false.

Exit conditional loop: The decision to execute the loop is present at the end of the loop.

The loop is executed at least once even if the decision happens to be false.

7b)

RTN model for Microprocessor datapath.

The data path is a collection of registers and an associated set of micro operations on the data held in the registers. The control unit directs the ordered execution of the micro operations so as to effect the desired transformation of the data. Thus the system‘s behavior can be expressed by the movement of data among these registers, by operations and transformations performed on the register‘s contents, and by the management of how such movements and operations take place. The operations on data found at the instruction level are paralled by a similar, yet more detailed, set of operations at the register level.

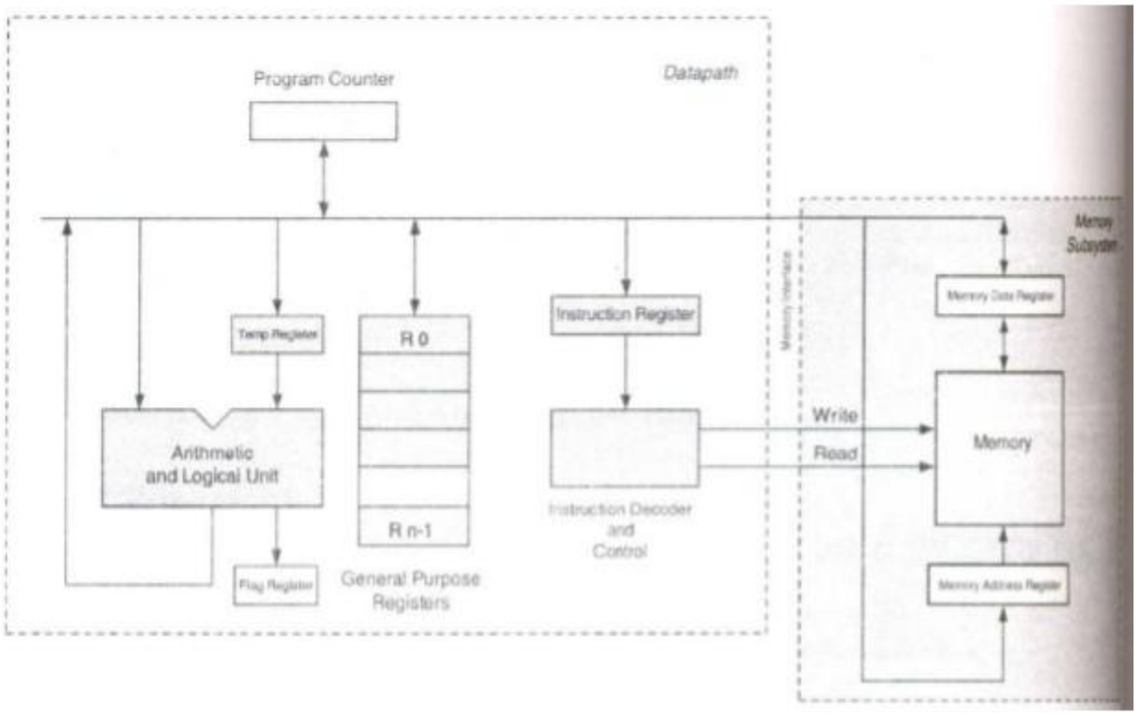
The control of the microprocessor data path comprises four fundamental operations defined as the instruction cycle:

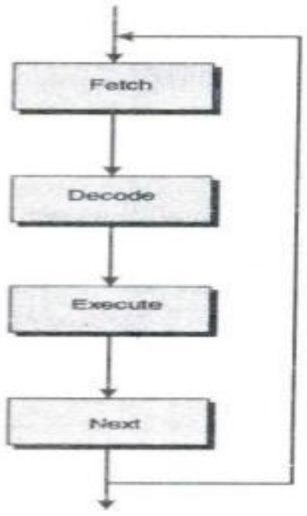
Fetch-The fetch operation retrieves an instruction from memory. That instruction is identified by its address, which is the contents of the PC.

Decode-The decode step is performed when the opcode field in the instruction is extracted from the instruction and decoded by the decoder. That information is forwarded to the control logic, which will initiate the execute portion of the instruction cycle.

Execute-Based on the value contained in the opcode field, the control logic performs the sequence of steps necessary to execute the instruction.

Next-The address of the next instruction to be executed is dependent on the type of instruction to be executed and potentially, on the state of the condition flags as modified by the recently completed instruction.





8 a) State diagram and Elements of state diagram:

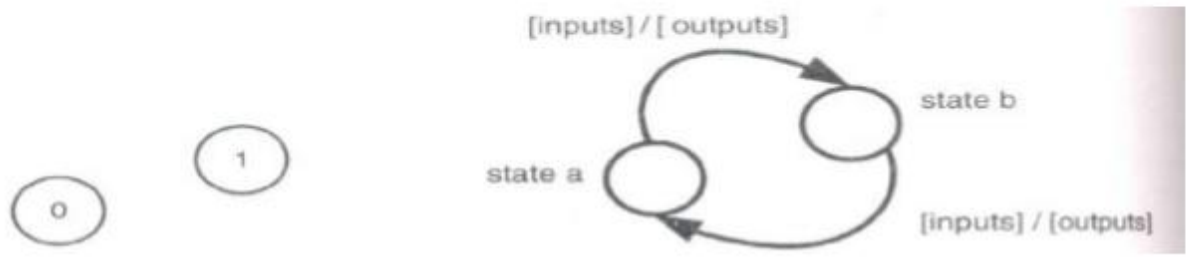
In the embedded world, the state diagram is one of the means used to capture, describe and specify the behavior of a system.

Elements of a state diagram: In a state diagram, each state is represented by a **circle, node**, or **vertex**. We label each node to identify the state.

We show the transition between two states using a labelled **directed line** or **arrow** called **arc.**

Because the line has a direction, the state diagram is referred to as a **Directed graph**. The head or **point of the arrow identifies the final state**, and the **tail or back of the arrow identifies the initial state.**

Eg:



8 b) Theoretical model of FSM:

A finite-state machine (FSM) or finite-state automaton (plural: automata), or simply a state machine, is a mathematical model of computation used to design both computer programs and sequential logic circuits.

The machine is in only one state at a time; the state it is in at any given time is called the current state. It can change from one state to another when initiated by a triggering event or condition; this is called a transition. A particular FSM is defined by a list of its states, and the triggering condition for each transition.

Simple examples are vending machines which dispense products when the proper combination of coins are deposited, elevators which drop riders off at upper floors before going down, traffic lights which change sequence when cars are waiting, and combination locks which require the input of combination numbers in the proper order.

Finite-state machines can model a large number of problems, among which are electronic design automation, communication protocol design, language parsing and other engineering applications.

Figure-1 shows a simple finite-state machine having no inputs other than a clock and have only primitive outputs. Such machines are referred to as autonomous clocks. A high level block diagram for a finite-state machine begins with the diagram in figure-1

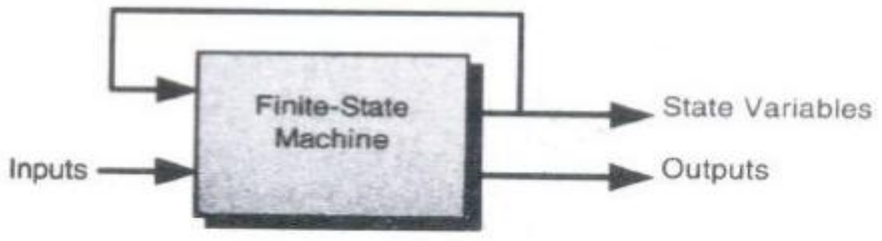


Fig-1

The output shown in the diagram may be the values of the state variables, combinations of the state variables, or combinations of the state variables and the inputs.

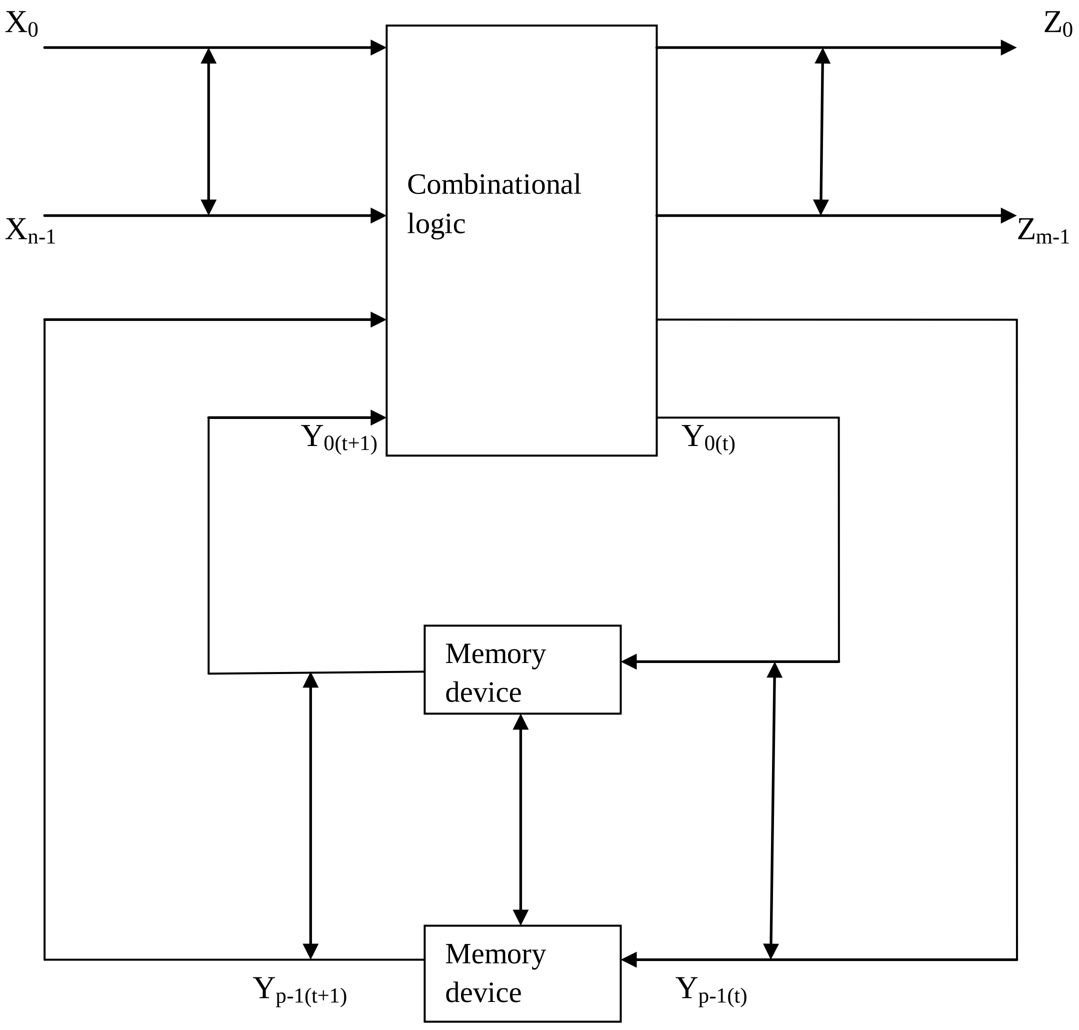


Fig-2