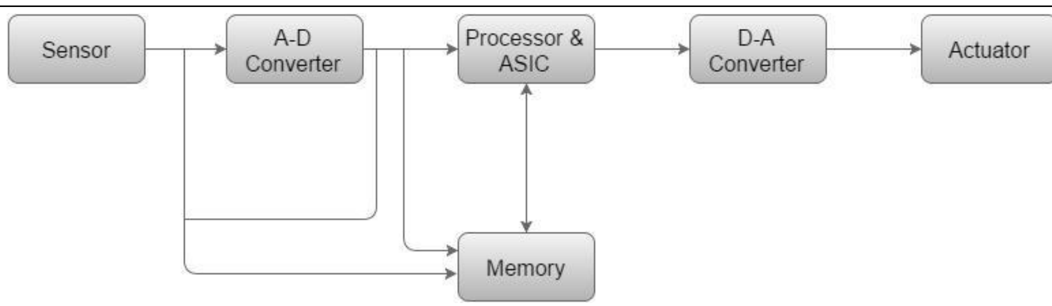
<table>
<tr><td>Q.1</td><td>What is System? Explain with an example</td></tr>
<tr><td></td><td>A system is an arrangement in which all its unit assemble work together according to a set of rules. It can also be defined as a way of working, organizing or doing one or many tasks according to a fixed plan. For example, a watch is a time displaying system. Its components follow a set of rules to show time. If one of its parts fails, the watch will stop working. So we can say, in a system, all its subcomponents depend on each other.</td></tr>
<tr><td>Q.2</td><td>Define Embedded System.</td></tr>
<tr><td></td><td>As its name suggests, Embedded means something that is attached to another thing. An embedded system can be thought of as a computer hardware system having software embedded in it. An embedded system can be an independent system or it can be a part of a large system. An embedded system is a microcontroller or microprocessor based system which is designed to perform a specific task. For example, a fire alarm is an embedded system; it will sense only smoke.</td></tr>
<tr><td>Q.3</td><td>Explain 3 components of Embedded System.</td></tr>
<tr><td></td><td>An embedded system has three components −
• It has hardware.
• It has application software.
• It has Real Time Operating system (RTOS) that supervises the application software and provide mechanism to let the processor run a process as per scheduling by following a plan to control the latencies. RTOS defines the way the system works. It sets the rules during the execution of application program. A small scale embedded system may not have RTOS.</td></tr>
<tr><td>Q.4</td><td>Basic Structure of an Embedded System</td></tr>
<tr><td></td><td>

The following illustration shows the basic structure of an embedded system –
• Sensor − It measures the physical quantity and converts it to an electrical signal which can be read by an observer or by any electronic instrument like an A2D converter. A sensor stores the measured quantity to the memory.</td></tr>
</table>

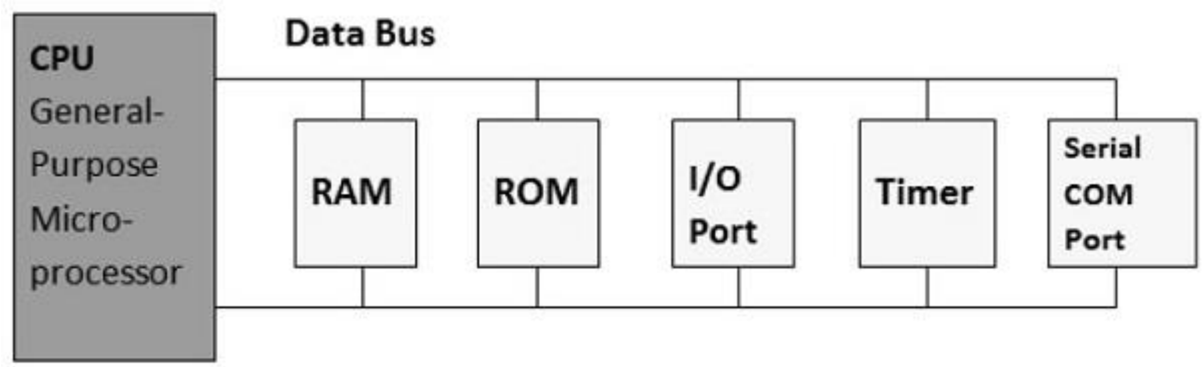
	• **A-D Converter** − An analog-to-digital converter converts the analog signal sent by the sensor into a digital signal. • **Processor & ASICs** − Processors process the data to measure the output and store it to the memory. • **D-A Converter** − A digital-to-analog converter converts the digital data fed by the processor to analog data • **Actuator** − An actuator compares the output given by the D-A Converter to the actual (expected) output stored in it and stores the approved output.
Q.5	**Four General Embedded System Types** ◆ **General Computing** • Applications similar to desktop computing, but in an embedded package • Video games, set-top boxes, wearable computers, automatic tellers ◆ **Control Systems** • Closed-loop feedback control of real-time system • Vehicle engines, chemical processes, nuclear power, flight control ◆ **Signal Processing** • Computations involving large data streams • Radar, Sonar, video compression ◆ **Communication & Networking** • Switching and information transmission • Telephone system, Internet
Q.6	Processor and its' units
	Processor is the heart of an embedded system. It is the basic unit that takes inputs and produces an output after processing the data. For an embedded system designer, it is necessary to have the knowledge of both microprocessors and microcontrollers. **Processors in a System**

	A processor has two essential units − • Program Flow Control Unit (CU) • Execution Unit (EU) The CU includes a fetch unit for fetching instructions from the memory. The EU has circuits that implement the instructions pertaining to data transfer operation and data conversion from one form to another. The EU includes the Arithmetic and Logical Unit (ALU) and also the circuits that execute instructions for a program control task such as interrupt, or jump to another set of instructions. A processor runs the cycles of fetch and executes the instructions in the same sequence as they are fetched from memory.
Q.7	Types of Processors

	Processors can be of the following categories −
	
	• General Purpose Processor (GPP)
	  ○ Microprocessor
	  ○ Microcontroller
	  ○ Embedded Processor
	  ○ Digital Signal Processor
	  ○ Media Processor
	• Application Specific System Processor (ASSP)
	• Application Specific Instruction Processors (ASIPs)
	• GPP core(s) or ASIP core(s) on either an Application Specific Integrated Circuit (ASIC) or a Very Large Scale Integration (VLSI) circuit.

Q.8	**Microprocessor**
	A microprocessor is a single VLSI chip having a CPU. In addition, it may also have other units such as coaches, floating point processing arithmetic unit, and pipelining units that help in faster processing of instructions.

Earlier generation microprocessors’ fetch-and-execute cycle was guided by a clock frequency of order of ~1 MHz. Processors now operate at a clock frequency of 2GHz



A SIMPLE BLOCK DIAGRAM OF A MICROPROCESSOR

Q.9

Microcontroller

A microcontroller is a single-chip VLSI unit (also called **microcomputer**) which, although having limited computational capabilities, possesses enhanced input/output capability and a number of on-chip functional units.

CPU	RAM	ROM
I/O Port	Timer	Serial COM Port

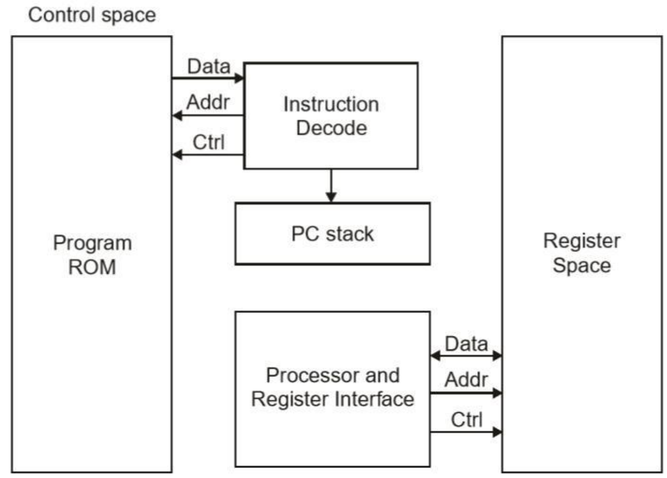
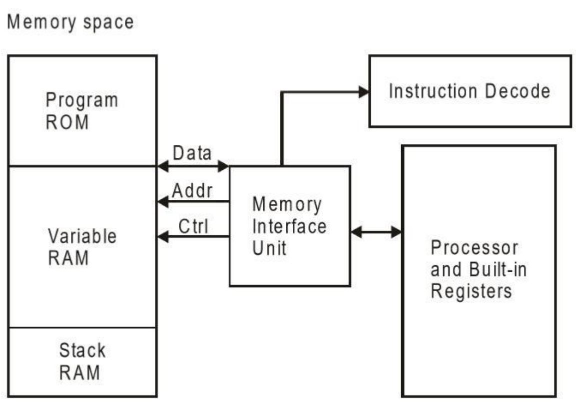
Microcontrollers are particularly used in embedded systems for real-time control applications with on-chip program memory and devices.

Q.10

Microprocessor vs Microcontroller

Microprocessor	**Microcontroller**
Microprocessors are multitasking in nature. Can perform multiple tasks at a time. For example, on computer we can play music while writing text in text editor.	Single task oriented. For example, a washing machine is designed for washing clothes only.

	RAM, ROM, I/O Ports, and Timers can be added externally and can vary in numbers.	RAM, ROM, I/O Ports, and Timers cannot be added externally. These components are to be embedded together on a chip and are fixed in numbers.
	Designers can decide the number of memory or I/O ports needed.	Fixed number for memory or I/O makes a microcontroller ideal for a limited but specific task.
	External support of external memory and I/O ports makes a microprocessor-based system heavier and costlier.	Microcontrollers are lightweight and cheaper than a microprocessor.
	External devices require more space and their power consumption is higher.	A microcontroller-based system consumes less power and takes less space.
Q.11	**Harvard architecture**.	
	The Harvard architecture offers separate storage and signal buses for instructions and data. This architecture has data storage entirely contained within the CPU, and there is no access to the instruction storage as data. Computers have separate memory areas for program instructions and data using internal data buses, allowing simultaneous access to both instructions and data. Programs needed to be loaded by an operator; the processor could not boot itself. In a Harvard architecture, there is no need to make the two memories share properties.	

<table>
<tr><td></td><td>
</td></tr>
<tr><td>Q.12</td><td>Von Neumann architecture</td></tr>
<tr><td></td><td>The Von Neumann architecture was first proposed by a computer scientist John von Neumann. In this architecture, one data path or bus exists for both instruction and data. As a result, the CPU does one operation at a time. It either fetches an instruction from memory, or performs read/write operation on data. So an instruction fetch and a data operation cannot occur simultaneously, sharing a common bus.


Von-Neumann architecture supports simple hardware. It allows the use of a single, sequential memory. Today's processing speeds vastly outpace memory access times, and we employ a very fast but small amount of memory (cache) local to the processor.</td></tr>
<tr><td>Q.13</td><td>

Von-Neumann Architecture vs Harvard Architecture

</td></tr>
</table>

	The following points distinguish the Von Neumann Architecture from the Harvard Architecture. **Von-Neumann Architecture** / **Harvard Architecture** Single memory to be shared by both code and data. / Separate memories for code and data. Processor needs to fetch code in a separate clock cycle and data in another clock cycle. So it requires two clock cycles. / Single clock cycle is sufficient, as separate buses are used to access code and data. Higher speed, thus less time consuming. / Slower in speed, thus more time-consuming. Simple in design. / Complex in design.

The following points distinguish the Von Neumann Architecture from the Harvard Architecture.

Von-Neumann Architecture	Harvard Architecture
Single memory to be shared by both code and data.	Separate memories for code and data.
Processor needs to fetch code in a separate clock cycle and data in another clock cycle. So it requires two clock cycles.	Single clock cycle is sufficient, as separate buses are used to access code and data.
Higher speed, thus less time consuming.	Slower in speed, thus more time-consuming.
Simple in design.	Complex in design.

Q.14

CISC and RISC

CISC is a Complex Instruction Set Computer. It is a computer that can address a large number of instructions.

In the early 1980s, computer designers recommended that computers should use fewer instructions with simple constructs so that they can be executed much faster within the CPU without having to use memory. Such computers are classified as Reduced Instruction Set Computer or RISC.

Q.15

CISC vs RISC

The following points differentiate a CISC from a RISC −

CISC	RISC
Larger set of instructions. Easy to program	Smaller set of Instructions. Difficult to program.

Simpler design of compiler, considering larger set of instructions.	Complex design of compiler.
Many addressing modes causing complex instruction formats.	Few addressing modes, fix instruction format.
Instruction length is variable.	Instruction length varies.
Higher clock cycles per second.	Low clock cycle per second.
Emphasis is on hardware.	Emphasis is on software.
Control unit implements large instruction set using micro-program unit.	Each instruction is to be executed by hardware.
Slower execution, as instructions are to be read from memory and decoded by the decoder unit.	Faster execution, as each instruction is to be executed by hardware.
Pipelining is not possible.	Pipelining of instructions is possible, considering single clock cycle.