EMBEDDED SYSTEMS
ARM processors

1. **Describe brief History of ARM processors.**

Ans. ARM Holdings has grown from a small Acorn in Cambridge - maker of some of the earliest home computers - into one of the world's most important designers of semiconductors, providing the brains for Apple's must-have iPhones and iPads.

ARM designs and licenses semiconductors to power devices such as Apple iPhones, and Microsoft have announced it will also base the next Windows system on ARM's chips. It does not actually make or sell semiconductors themselves.

1978 - Acorn Computers is established in Cambridge, and produces computers which are particularly successful in the UK. Acorn's BBC Micro computer was the most widely-used computer in school in the 1980s.

1985 - Acorn Computer Group develops the world's first commercial RISC processor - enabling a computer system which uses simpler commands in order to operate faster, an advance on the early computer systems which were created using machine code and tried to pack as many actions into each command as possible.

The company was taken over by Olivetti, after coming close to bankruptcy following the collapse of the UK home computer market the previous year.

1987 - Acorn's ARM processor is the first RISC processor available in a low-cost PC.

1990 - ARM is founded as a spin-off from Acorn and Apple, after the two companies started collaborating on the ARM processor as part of the development of Apple's new Newton computer system.

1992 - Sharp and GEC Plessy both sign up to use ARM's technology.

1998 - ARM Holdings floats on the London Stock Exchange and the Nasdaq.

1999 - ARM becomes a member of the FTSE 100.

2001 - ARM has 76.8pc share of the 32-bit embedded RISC chip market.

2007 - About 98pc of the more than 1bn mobile phones sold each year use at least one ARM processor.

The RISC processors first conceived by Acorn back in the 1980s were designed to be simple, and this made them relatively low-power. That has made ARM's chip designs dominant in the mobile device market, because they use less battery and don't take up much space.

2008 - The 10 billionth processor chip based on ARM's designs is shipped.

2011 - Steve Ballmer, Microsoft's chief executive, announces plans to base the next generation of Microsoft's Windows operating system on microchips designed by ARM. Until now Microsoft's PC software has been based on chips design by Intel of the US.

2. **What do you mean by ARM family?**

Ans.

ARM Processor Families

ARM has designed a number of processors that are grouped into different families according to the core they use. The families are based on the ARM7, ARM9, ARM10, and ARM11 cores. The postfix numbers 7, 9, 10, and 11 indicate different core designs. The ascending number equates to an increase in performance and sophistication. ARM8 was developed but was soon superseded. Table 2.9 shows a rough comparison of attributes between the ARM7, ARM9, ARM10, and ARM11 cores. The numbers quoted can vary greatly and are directly dependent upon the type and geometry of the manufacturing process, which has a direct effect on the frequency (MHz) and power consumption (watts).

Table 2.7 Revision history.

Revision	Example core implementation	ISA enhancement
ARMv1	ARM1	First ARM processor 26-bit addressing
ARMv2	ARM2	32-bit multiplier 32-bit coprocessor support
ARMv2a	ARM3	On-chip cache Atomic swap instruction Coprocessor 15 for cache management
ARMv3	ARM6 and ARM7DI	32-bit addressing Separate *cpsr* and *spsr* New modes—*undefined instruction* and *abort* MMU support—virtual memory
ARMv3M	ARM7M	Signed and unsigned long multiply instructions
ARMv4	StrongARM	Load-store instructions for signed and unsigned halfwords/bytes New mode—*system* Reserve SWI space for architecturally defined operations 26-bit addressing mode no longer supported
ARMv4T	ARM7TDMI and ARM9T	Thumb
ARMv5TE	ARM9E and ARM10E	Superset of the ARMv4T Extra instructions added for changing state between ARM and Thumb Enhanced multiply instructions Extra DSP-type instructions Faster multiply accumulate
ARMv5TEJ	ARM7EJ and ARM926EJ	Java acceleration
ARMv6	ARM11	Improved multiprocessor instructions Unaligned and mixed endian data handling New multimedia instructions

Within each ARM family, there are a number of variations of memory management, cache, and TCM processor extensions. ARM continues to expand both the number of families available and the different variations within each family. You can find other processors that execute the ARM ISA such as StrongARM and XScale. These processors are unique to a particular semiconductor company, in this case Intel. Table 2.10 summarizes the different features of the various processors. The next subsections describe the ARM families in more detail, starting with the ARM7 family.

Table 2.8 Description of the *cpsr*.

Parts	Bits	Architectures	Description
Mode	4:0	all	processor mode
T	5	ARMv4T	Thumb state
I & *F*	7:6	all	interrupt masks
J	24	ARMv5TEJ	Jazelle state
Q	27	ARMv5TE	condition flag
V	28	all	condition flag
C	29	all	condition flag
Z	30	all	condition flag
N	31	all	condition flag

Table 2.9 ARM family attribute comparison.

	ARM7	ARM9	ARM10	ARM11
Pipeline depth	three-stage	five-stage	six-stage	eight-stage
Typical MHz	80	150	260	335
mW/MHz[a]	0.06 mW/MHz	0.19 mW/MHz (+ cache)	0.5 mW/MHz (+ cache)	0.4 mW/MHz (+ cache)
MIPS[b]/MHz	0.97	1.1	1.3	1.2
Architecture	Von Neumann	Harvard	Harvard	Harvard
Multiplier	8 × 32	8 × 32	16 × 32	16 × 32

ARM7 Family

The ARM7 core has a Von Neumann–style architecture, where both data and instructions use the same bus. The core has a three-stage pipeline and executes the architecture ARMv4T instruction set.

The ARM7TDMI was the first of a new range of processors introduced in 1995 by ARM. It is currently a very popular core and is used in many 32-bit embedded processors. It provides a very good performance-to-power ratio. The ARM7TDMI processor core has been licensed by many of the top semiconductor companies around the world and is the first core to include the Thumb instruction set, a fast multiply instruction, and the Embedded ICE debug technology.

Table 2.10 ARM processor variants.

CPU core	MMU/MPU	Cache	Jazelle	Thumb	ISA	E[a]
ARM7TDMI	none	none	no	yes	v4T	no
ARM7EJ-S	none	none	yes	yes	v5TEJ	yes
ARM720T	MMU	unified—8K cache	no	yes	v4T	no
ARM920T	MMU	separate—16K /16K $D + I$ cache	no	yes	v4T	no
ARM922T	MMU	separate—8K/8K $D + I$ cache	no	yes	v4T	no
ARM926EJ-S	MMU	separate—cache and TCMs configurable	yes	yes	v5TEJ	yes
ARM940T	MPU	separate—4K/4K $D + I$ cache	no	yes	v4T	no
ARM946E-S	MPU	separate—cache and TCMs configurable	no	yes	v5TE	yes
ARM966E-S	none	separate—TCMs configurable	no	yes	v5TE	yes
ARM1020E	MMU	separate—32K/32K $D + I$ cache	no	yes	v5TE	yes
ARM1022E	MMU	separate—16K/16K $D + I$ cache	no	yes	v5TE	yes
ARM1026EJ-S	MMU and MPU	separate—cache and TCMs configurable	yes	yes	v5TE	yes
ARM1136J-S	MMU	separate—cache and TCMs configurable	yes	yes	v6	yes
ARM1136JF-S	MMU	separate—cache and TCMs configurable	yes	yes	v6	yes

One significant variation in the ARM7 family is the ARM7TDMI-S. The ARM7TDMI-S has the same operating characteristics as a standard ARM7TDMI but is also synthesizable. ARM720T is the most flexible member of the ARM7 family because it includes an MMU. The presence of the MMU means the ARM720T is capable of handling the Linux and Microsoft embedded platform operating systems. The processor also includes a unified 8K cache. The vector table can be relocated to a higher address by setting a coprocessor 15 register. Another variation is the ARM7EJ-S processor, also synthesizable. ARM7EJ-S is quite different since it includes a five-stage pipeline and executes ARMv5TEJ instructions. This version of the ARM7 is the only one that provides both Java acceleration and the enhanced instructions but without any memory protection.

ARM9 Family

The ARM9 family was announced in 1997. Because of its five-stage pipeline, the ARM9 processor can run at higher clock frequencies than the ARM7 family. The extra stages improve the overall performance of the processor. The memory system has been redesigned to follow the Harvard architecture, which separates the dataD and instructionI buses.

The first processor in the ARM9 family was the ARM920T, which includes a separate D+I cache and an MMU. This processor can be used by operating systems requiring virtual memory support. ARM922T is a variation on the ARM920T but with half the D+I cache size. The ARM940T includes a smaller D+I cache and an MPU. The ARM940T is designed for applications that do not require a platform operating system. Both ARM920T and ARM940T

execute the architecture v4T instructions. The next processors in the ARM9 family were based on the ARM9E-S core. This core is a synthesizable version of the ARM9 core with the E extensions. There are two variations: the ARM946E-S and the ARM966E-S. Both execute architecture v5TE instructions. They also support the optional embedded trace macrocell (ETM), which allows a developer to trace instruction and data execution in real time on the processor. This is important when debugging applications with time-critical segments. The ARM946E-S includes TCM, cache, and an MPU. The sizes of the TCM and caches are configurable. This processor is designed for use in embedded control applications that require deterministic real-time response. In contrast, the ARM966E does not have the MPU and cache extensions but does have configurable TCMs. The latest core in the ARM9 product line is the ARM926EJ-S synthesizable processor core, announced in 2000. It is designed for use in small portable Java-enabled devices such as 3G phones and personal digital assistants (PDAs). The ARM926EJ-S is the first ARM processor core to include the Jazelle technology, which accelerates Java byte code execution. It features an MMU, configurable TCMs, and D+I caches with zero or nonzero wait state memories.

ARM10 Family

The ARM10, announced in 1999, was designed for performance. It extends the ARM9 pipeline to six stages. It also supports an optional vector floating-point (VFP) unit, which adds a seventh stage to the ARM10 pipeline. The VFP significantly increases floating-point performance and is compliant with the IEEE 754.1985 floating-point standard. The ARM1020E is the first processor to use an ARM10E core. Like the ARM9E, it includes the enhanced E instructions. It has separate 32K D+I caches, optional vector floating-point unit, and an MMU. The ARM1020E also has a dual 64-bit bus interface for increased performance. ARM1026EJ-S is very similar to the ARM926EJ-S but with both MPU and MMU. This processor has the performance of the ARM10 with the flexibility of an ARM926EJ-S.

ARM11 Family

The ARM1136J-S, announced in 2003, was designed for high performance and power efficient applications. ARM1136J-S was the first processor implementation to execute architecture ARMv6 instructions. It incorporates an eight-stage pipeline with separate load store and arithmetic pipelines. Included in the ARMv6 instructions are single instruction multiple data (SIMD) extensions for media processing, specifically designed to increase video processing performance. The ARM1136JF-S is an ARM1136J-S with the addition of the vector floating-point unit for fast floating-point operations.

3. **What are the latest most popular ARM processors?**

Ans. Cortex-A Series

The ARM® Cortex®-A series of applications processors provide an entire range of solutions for devices hosting a rich OS platform and user applications ranging from ultra-low-cost handset through smartphones, mobile computing platforms, digital TV and set-top boxes to enterprise networking, printers and server solutions.

Cortex-A devices provide a scalable range of power-efficient performance points for their target applications. Some illustrative examples are:

- Cortex-A17 processor most efficient mid-range solutions, delivering a new mid-range performance point for any screen - from Mobiles to Smart-TVs

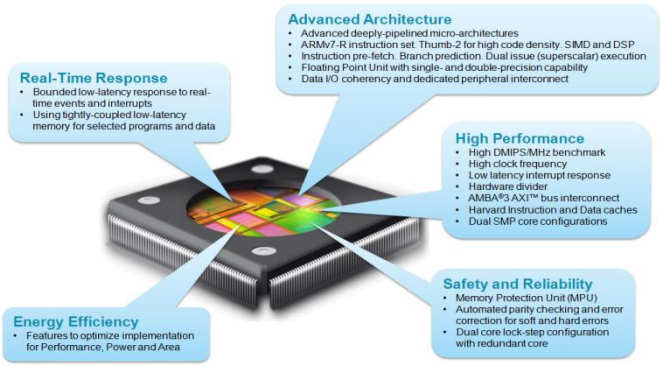
- Cortex-A15 processor highest performance solutions for current mobile and demanding wireless infrastructure applications.
- Cortex-A12 processor high performance mid-range solutions targeted at mobile smartphone and tablet devices.
- Cortex-A9 processor implementations delivering 800 MHz - 2 GHz typical frequency and delivering 5000 DMIPS of performance per core.
- Cortex-A7 processor implementations in standalone, multi-core configurations delivering 800 MHz - 1.2 GHz typical frequency, or in combination with Cortex-A15 processor for big.LITTLE processing.
- Cortex-A5 processor in low-cost implementations from 400- 800 MHz delivering over 1200 DMIPS of performance.

1. Cortex-R Series

The ARM® Cortex®-R real-time processors offer high-performance computing solutions for embedded systems where reliability, high availability, fault tolerance, maintainability and real-time responses are required.

The Cortex-R series processors provide fast time-to-market through proven technology shipped in hundreds of millions of products and leverages the vast ARM Ecosystem and global, local language, 24/7 support services to ensure rapid and low risk development.

Cortex-R Series Features:



- Deeply pipelined micro-architecture
- Performance enhancing technologies such instruction pre-fetch, branch prediction and superscalar execution
- Hardware divider, Floating Point Unit (FPU) option
- Hardware SIMD DSP
- ARMv7-R architecture with Thumb-2 instructions for high code density without sacrificing performance
- Harvard architecture with Instruction and Data cache controllers
- Tightly-Coupled Memories (TCM) local to the processor for fast-responding code and data such as interrupt handlers
- High performance AMBA®3 AXI™ bus interfaces

2. Cortex-M Series

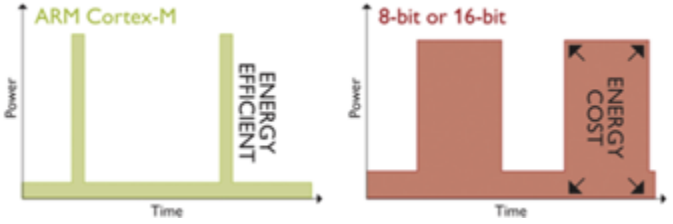
The ARM® Cortex®-M processor family is an upwards compatible range of energy-efficient, easy to use processors designed to help developers meet the needs of tomorrow's embedded applications. Those demands include delivering more features at a lower cost, increasing connectivity, better code reuse and improved energy efficiency.

Cortex-M Series Features:

Industry standard

ARM Cortex-M processors is a global microcontroller standard, having been licensed to over 40 ARM partners including leading vendors such as Freescale, NXP Semiconductors, STMicroelectronics, Texas Instruments, and Toshiba. Using a standard processor allows ARM partners to create devices with a consistent architecture while enabling them to focus on creating superior device implementations.

Energy efficiency



Lower energy costs, longer battery life

- Run at lower MHz or with shorter activity periods
- Architected support for sleep modes
- Work smarter, sleep longer than 8/16-bit

Smaller code

Lower silicon costs

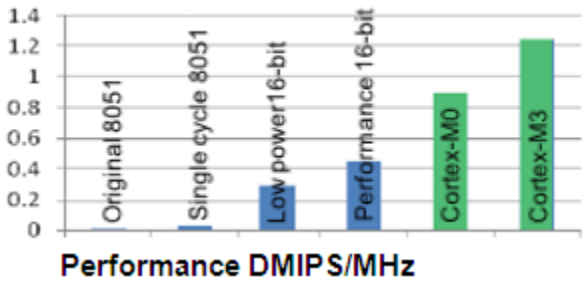
- High density instruction set
- Achieve more per byte than 8/16-bit devices
- Smaller RAM, ROM or Flash requirement

Ease of use

Faster software development and reuse

- Global standard across multiple vendors
- Code compatibility
- Unified tools and OS support

High performance



More competitive products

- Powerful Cortex-M processor
- Delivers more performance per MHz
- Enables richer features at lower power

4. Describe ARM Instruction Set in detail.

Ans. Instruction set overview

All ARM instructions are 32 bits long. Instructions are stored word-aligned, so the least significant two bits of instruction addresses are always zero in ARM state.

Thumb and Thumb EE instructions are either 16 or 32 bits long. Instructions are stored half-word aligned. Some instructions use the least significant bit of the address to determine whether the code being branched to is Thumb code or ARM code.

Before the introduction of Thumb-2 technology, the Thumb instruction set was limited to a restricted subset of the functionality of the ARM instruction set. Almost all Thumb instructions were 16-bit. Together, the 32-bit and 16-bit Thumb instruction set functionality is almost identical to that of the ARM instruction set.Table 5 describes some of the functional groupings of the available instructions.

Table 5. Instruction groups

Instruction Group	Description
Branch and control	These instructions are used to: • branch to subroutines • branch backwards to form loops • branch forward in conditional structures • make following instructions conditional without branching • change the processor between ARM state and Thumb state.
Data processing	These instructions operate on the general-purpose registers. They can perform operations such as addition, subtraction, or bitwise logic on the contents of two registers and place the result in a third register. They can also operate on the value in a single register, or on a value in a register and an immediate value supplied within the instruction. Long multiply instructions give a 64-bit result in two registers.
Register load and store	These instructions load or store the value of a single register from or to memory. They can load or store a 32-bit word, a 16-bit halfword, or an 8-bit unsigned byte. Byte and halfword loads can either be sign extended or zero extended to fill the 32-bit register. A few instructions are also defined that can load or store 64-bit doubleword values into two 32-bit registers.
Multiple register load and store	These instructions load or store any subset of the general-purpose registers from or to memory.

Instruction Group	Description
Status register access	These instructions move the contents of a status register to or from a general-purpose register.
Coprocessor	These instructions support a general way to extend the ARM architecture. They also enable the control of the CP15 System Control coprocessor registers.

Q1. Explain JTAG.

Solution

Introduction

JTAG is an established technology (and industry standard) with a potential that is only now becoming fully realised. Connection testing and In System Programming (ISP) are the two applications most often associated with JTAG, but it has far more to offer.

Though we don't highlight XJTAG products here, they are designed to harnesses the full power of JTAG and are able to implement all of the functionality described in this document.

Background

JTAG was conceived to address difficulties in testing circuits using the traditional 'bed-of-nails' approach. Modern packaging technologies such as BGA and Chip Scale Packaging limit and in some cases eliminate physical access to pins*.

JTAG overcomes this problem, by placing cells between the external connections and the internal logic of the device, see Figure 1.With the cells configured as a shift register, JTAG can be used set and retrieve the values of pins (and the nets connected to them) without direct physical access.

There's also an option to sample the data values as they pass between the core logic and the pins during the normal operation of the device.

The JTAG interface adds four extra pins on each device: TDI to input data to the device, TDO to output data from the device, TMS to control what should be done with the data and a clock signal, TCK to synchronise everything.

For a device to be **JTAG-compliant** its manufacturer must provide a BSDL (Boundary Scan Description Language) file, which describes how the JTAG aspects of the device work.

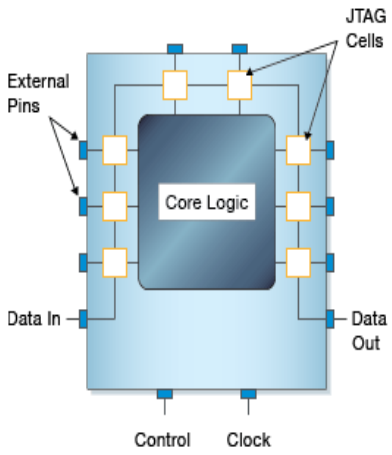


Figure 1 - Simple JTAG device

If a circuit contains more than one JTAG compliant device, these can be linked together to form a **JTAG chain**. In a JTAG chain the data output from the first device becomes the data input to the second device; the control and clock signals are common to all devices in the chain. Figure 2 provides a representation of a simple JTAG chain containing three devices.

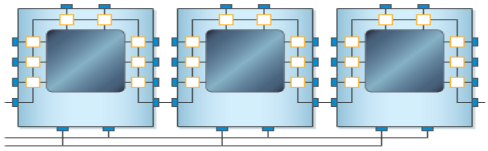


Figure 2 - Simple JTAG chain

*The word pins will be used to describe both leaded device connections and BGA solder ball connections.

Ball Grid Array (BGA)

A BGA package, such as that shown in Figure 3, differs from earlier package technologies in that all of its external connections are made through balls of solder between the bottom face of the device and the circuit board rather than through pins protruding from the side of the device.

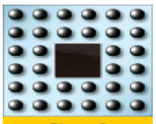


Figure 3
BGA device

The testing of circuits containing BGA packaged devices has been one of the driving forces in popularising JTAG testing. Because the connections between the device and the circuit board are inaccessible both physically and for visual inspection, the only alternative to JTAG for monitoring manufacturing integrity is X-ray inspection. This costly and time-consuming process requires that each board be X-rayed and the images inspected to check each solder ball has been correctly placed, makes contact with the board, but has not spread to cause short circuits. X-Ray provides useful information, but still relies on visual inspection, manual or automated, so cannot be fully relied on to locate all errors.

Against this background JTAG Boundary Scan is more than a useful alternative to bed-of-nails testing; it is a significant money saving tool that can eliminate the need for costly and possibly inconclusive X-ray technology.

Question Bank Based on PTU Paper Pattern (Unit Wise)

UNIT –1

1. Define a System.
A way of doing one or more tasks according to a program.
2. What is an embedded system?
An embedded system employs a combination of hardware & software (a "computational engine") to perform a specific function; is part of a larger system that may not be a "computer"; works in a reactive and time-constrained environment
3. **What are the typical characteristics of an embedded system?**
Typical characteristics:
Perform a single or tightly knit set of functions;
Increasingly high-performance & real-time constrained;
Power, cost and reliability are often important attributes
That influence design;
Application specific processor design can be a significant component of some embedded systems.
Other characteristics:
• Application specific
• Digital signal processing in ECS
• Reactive
• Real-time
• Distributed
4. What are the advantages of embedded system?
Advantages:

Customization yields lower area, power, cost, etc.,

5. What are the disadvantages of embedded system?

Disadvantages:

Higher HW/software development overhead design, compilers, debuggers, etc., may result in delayed time to market!

6. What are the applications of an embedded system?

Embedded Systems: Applications:

• Consumer electronics, e.g., cameras, camcorders, etc.,
• Consumer products, e.g., washers, microwave ovens, etc.,
• Automobiles (anti-lock braking, engine control, etc.,)
• Industrial process controllers & avionics/defense applications
• Computer/Communication products, e.g., printers, FAX machines, etc.,
• Emerging multimedia applications & consumer electronics

7. What are the various embedded system designs Modeling Refining (or "partitioning")

HW-SW partitioning

8. What are the complicating factors in embedded design?

Complicating factors in the design of embedded systems

• Many of the subtasks in design are intertwined.
• Allocation depends on the partitioning, and scheduling presumes a certain allocation.
• Predicting the time for implementing the modules in hardware or software is not very easy, particularly for tasks that have not been performed before.

9. What are the real-time requirements of an embedded system?

Hard-real time systems: where there is a high penalty for missing a deadline e.g., control systems for aircraft/space probes/nuclear reactors; refresh rates for video, or DRAM. Soft realtime systems: where there is a steadily increasing penalty if a deadline is missed. e.g., laser printer: rated by pages-per-minute, but can take differing times to print a page (depending on the complexity of the page) without harming the machine or the customer.

10. What are the functional requirements of embedded system?

Data Collection

• Sensor requirements
• Signal conditioning
• Alarm monitoring

Direct Digital Control

• Actuators

Man-Machine Interaction

• informs the operator of the current state of the controlled object
• assists the operator in controlling the system.

11. What are the main components of an embedded system?

Three main components of embedded systems:

1. The Hardware
2. Application Software
3. RTOS

12. Define embedded microcontroller.

An embedded microcontroller is particularly suited for embedded applications to perform dedicated task or operation.

Example: 68HC11xx, 8051, PIC, 16F877, etc.,

6. What are the applications of an embedded system?

Embedded Systems: Applications:

• Consumer electronics, e.g., cameras, camcorders, etc.,
• Consumer products, e.g., washers, microwave ovens, etc.,
• Automobiles (anti-lock braking, engine control, etc.,)
• Industrial process controllers & avionics/defense applications
• Computer/Communication products, e.g., printers, FAX machines, etc.,
• Emerging multimedia applications & consumer electronics

7. What are the various embedded system designs Modeling Refining (or "partitioning")

HW-SW partitioning

8. What are the complicating factors in embedded design?

Complicating factors in the design of embedded systems

• Many of the subtasks in design are intertwined.
• Allocation depends on the partitioning, and scheduling presumes a certain allocation.
• Predicting the time for implementing the modules in hardware or software is not very easy, particularly for tasks that have not been performed before.

9. What are the real-time requirements of an embedded system?

Hard-real time systems: where there is a high penalty for missing a deadline e.g., control systems for aircraft/space probes/nuclear reactors; refresh rates for video, or DRAM. Soft realtime systems: where there is a steadily increasing penalty if a deadline is missed. e.g., laser printer: rated by pages-per-minute, but can take differing times to print a page (depending on the \"complexity\" of the page) without harming the machine or the customer.

10. What are the functional requirements of embedded system?

Data Collection

• Sensor requirements
• Signal conditioning
• Alarm monitoring

Direct Digital Control

• Actuators

Man-Machine Interaction

• informs the operator of the current state of the controlled object
• assists the operator in controlling the system.

11. What are the main components of an embedded system?

Three main components of embedded systems:

1. The Hardware
2. Application Software
3. RTOS

12. Define embedded microcontroller.

An embedded microcontroller is particularly suited for embedded applications to perform dedicated task or operation. Example: 68HC11xx, 8051, PIC, 16F877, etc.,