K.S.R. COLLEGE OF ENGINEERING (Autonomous)
SEMESTER - VII

		L	T	P	C
12EC3702	**EMBEDDED SYSTEM DESIGN** (Common to EC & IT)	3	0	0	3

Objective(s):

- *To learn the basics of embedded computing platform, its hardware and software.*
- *To understand the concepts of processes and operating systems, for embedded system design.*
- *To gain knowledge about the concepts of hardware accelerators, networks and system design techniques in embedded system.*

UNIT - I INTRODUCTION TO EMBEDDED ARCHITECTURE 09 Hrs

Complex systems and Microprocessors - Embedded system design process - Formalisms for system design -Design example: Model train controller - Instruction sets preliminaries - ARM processor - CPU: Programming input and output - Supervisor mode, Exception and traps – Co-processors- Memory system mechanism – CPU performance - CPU power consumption.

UNIT - II COMPUTING PLATFORM AND DESIGN ANALYSIS 09 Hrs

CPU buses - Memory devices - I/O devices - Component interfacing - Design with microprocessors - Development and debugging -Components for embedded programs- Model of programs - Assembly, Linking and loading - Basic compilation techniques -Program optimization - Program validation and testing.

UNIT - III PROCESSES AND OPERATING SYSTEMS 09 Hrs

Multiple tasks and multi processes -Preemptive real time operating systems - Priority based scheduling - Inter process communication mechanisms - Evaluating operating system performance - Power management and optimization for processes.

UNIT - IV HARDWARE ACCELERATORS & NETWORKS 09 Hrs

CPUs and accelerators -Multiprocessor performance analysis-Consumer electronics architecture - Distributed embedded architecture - Networks for embedded systems - Network based design - Internet enabled systems - Vehicles as networks - Sensor networks.

UNIT - V SYSTEM DESIGN TECHNIQUES 09 Hrs

Design methodologies - Requirement analysis - Specifications - System analysis and architecture design - Quality assurance - Software tools for embedded system development- Design example: Alarm clock, Software modem, Elevator controller.

Total hours: 45

Text Books:

1. Wayne Wolf, “Computers as Components - Principles of Embedded Computing System Design”, Morgan Kaufmann Publisher,2nd Edition, 2011.
2. Raj Kamal, “Embedded Systems - Architecture, Programming and Design”, Tata McGraw Hill, 2nd Edition, 2008.

References:

1. David E Simon, “An Embedded Software Primer”, Pearson Education, 2007.
2. K.V.K.K.Prasad, “Embedded Real-Time Systems: Concepts, Design & Programming”, Dreamtech Press, 2005.
3. Sriram V Iyer, Pankaj Gupta, “Embedded Real Time Systems Programming”, TMG, 2004.
4. Tammy Noergaard, “Embedded Systems Architecture”, Elsevier, 2006.

K.S.R.COLLEGE OF ENGINEERING (AUTONOMOUS), TIRUCHENGODE-637215
DEPARTMENT OF ELECTRONICS COMMUNICATION AND ENGINEERING

COURSE / LESSION PLAN SCHEDULE

NAME	**: C.ARUNPRASATH, R.MAHENDRAN**	**CLASS**	**:**	**FINAL ECE**
SUBJECT	**: 12EC3702 / EMBEDDED SYSTEMS DESIGN**	**SEM**	**:**	**VII**

A.TEXT BOOKS:

1. Wayne Wolf, "Computers as Components; Principles of Embedded Computing System Design", Morgan Kaufmann Publisher, 2nd Edition, 2011.
2. Raj Kamal, "Embedded Systems Architecture Programming and Design" 2nd Edition TMH, 2008.

B. REFERENCE:

1. David E Simon "An Embedded Software Primer " Pearson Education ,2007.
2. K.V.K.K.Prasad "Embedded Real-Time Systems: Concepts, Design and Programming" Dreamtech press 2005.
3. Sriram V Iyer, Pankaj Gupta, "Embedded Real Time Systems Programming", TMG, 2004.
4. Tammy Noergaard, "Embedded System Architecture", Elsevier, 2006.

C. LEGEND:

L	-	Lecture	BB	-	Block Board
OHP	-	Over Head Projector	Tx	-	Text Book
pp	-	Pages	Rx	-	Reference Book
LCDP	-	Liquid Crystal Display Projector			

S.No	Lecture Hour	Topics to be covered	Teaching Aid Required	Book No /Page No
UNIT I - INTRODUCTION TO EMBEDDED ARCHITECTURE				
1	L1	Complex systems and Microprocessors	BB	Tx1-pp(2-9)
2.	L2	Embedded system design process	BB, LCDP	Tx1-pp(10-21)
3.	L3	Formalisms for system design	BB	Tx1-pp(21-32)
4.	L4	Design example: Model train controller	BB	Tx1-pp(32-46)
5	L5	Instruction sets preliminaries	BB	Tx1-pp(57-62)
6	L6	ARM Processor	BB , LCDP	Tx1-pp(62-82)
7	L7	CPU: Programming input and output	BB	Tx1-pp(105-128)
8	L8	Supervisor modes, Exception and traps-co-processors-Memory system mechanism	BB	Tx1-pp(128-144)
9	L9	CPU Performance-CPU power consumption, **Testing tools**	**BB**	Tx1-pp(144-158) **Rx2-pp-66**
UNIT II - COMPUTING PLATFORM AND DESIGN ANALYSIS				
10	L10	CPU Buses	BB, LCDP	Tx1-pp(178-193)

11	L11	Memory devices	BB, LCDP	Tx1-pp(193-201)
				Rx2-pp(39-42)

12	L12	I/O Devices, Component interfacing	BB	Tx1-pp(202-212)
				Rx2-pp(42-48)

13	L13	Design with microprocessors	BB	Tx1-pp(212-220)
14	L14	Development and debugging	BB	Tx1-pp(220-228)

15	L15	Components for embedded programs, Model of programs	BB	Tx1-pp(252-257)
				Tx2-pp(223-227)

16	L16	Assembly, linking and loading	BB	Tx1-pp(258-265)
17	L17	Basic compilation techniques	BB	Tx1-pp(265-290)
18	L18	Program optimization, Program validation and testing, **Productivity tools**	**BB**	Tx1-pp(305-325)

UNIT III - PROCESSES AND OPERATING SYSTEMS				
19	L19	Multiple tasks and multi processes	BB, LCDP	Tx1-pp(343-351)
20	L20	Preemptive real time operating systems	BB	Tx1-pp-356
21	L21	Priority based scheduling	BB	Tx1-pp-358
22	L22, L23	Inter process communication mechanisms	BB	Tx1-pp(387-393)
23	L24	Evaluating operating system performance	BB	Tx1-pp(393-396)
24	L25	Power management	BB	Tx1-pp(396-400)
25	L26	Optimization for processes	BB	Tx1-pp-(262-267)
26	L27	**Assembly language programming.**	BB	Tx2-pp-147
UNIT IV - HARDWARE ACCELERATORS AND NETWORKS				
27	L28	CPUs and accelerators	BB	Tx1-pp(420-423)
28	L29	Multiprocessor performance analysis	BB	Tx1-pp(424-437)
29	L30	Consumer electronics architecture	BB, LCDP	Tx1-pp-(369-373)
30	L31	Distributed embedded architecture	BB	Tx1-pp(450-458)

31	L32	Network for embedded systems	BB	Tx1-pp(458-475)
				Tx2-pp(114-118)

32	L33	Network based design	BB	Tx1-pp(475-484)
33	L34	Internet enabled systems	BB	Tx1-pp(484-486)
34	L35	Vehicles as networks	BB, LCDP	Tx1-pp-(421-426)
35	L36	Sensor networks, **Video accelerator**	BB	Tx1-pp(437-445)
UNIT V - SYSTEM DESIGN TECHNIQUES				

36	L37	Design methodologies	BB, LCDP	Tx1-pp(498-507)
				Rx2-pp(99-102)

37	L38	Requirement analysis	BB	Tx1-pp(507-508)
				Rx2-pp(102-104)

38	L39	Specifications	BB	Tx1-pp(509-515)
39	L40	System analysis and architecture design	BB	Tx1-pp(515-520)
40	L41	Quality assurance	BB	Tx1-pp(520-534)
41	L42	Software tools for embedded system development	BB	Tx2-pp(34-36)
42	L43	Design example: Alarm clock	BB, LCDP	Tx1-pp(233-241)
43	L44	Software Modem	BB, LCDP	Tx1-pp(325-330)
44	L45	Elevator Controller, **Telephone PBX**	BB, LCDP	Tx1-pp(486-493) (534-539)

K.S.R.COLLEGE OF ENGINEERING (AUTONOMOUS), TIRUCHENGODE-637215

DEPARTMENT OF ELECTRONICS COMMUNICATION AND ENGINEERING

12EC3702 - EMBEDDED SYSTEM DESIGN

QUESTION BANK

UNIT - I

INTRODUCTION TO EMBEDDED ARCHITECTURE

2 MARKS:

1. What is an Embedded System? (Remembering) (CO1)

An embedded system can be defined as an computing device that does a specific focused job. E.g: Air-conditioner, VCD payer, DVD player, printer, fax machine, mobile phone,etc.

2. List some application area of embedded systems? (Analyzing) (CO1)

Consumer appliances, industrial automation, medical electronics, computer networking, telecommunications, wireless technologies, instrumentations, security, finance.

3. How an embedded system is categorized? (Remembering) (CO1)

Based on the functionality and performance requirements, embedded system can be categorized as: Stand-alone system, Real-time system, Networked information application, Mobile devices.

4. What is Stand-alone embedded systems? (Remembering) (CO1)

Stand-alone systems works in Stand-alone mode. They take inputs, process them and produced the desired output, Example LCD and LED Display.

5. What is real time systems? (Remembering) (CO1)

Embedded systems in which some specific work has to be done in a specific time period is called real-time systems. They are of two types,

a. Hard real time systems, b. Soft real time systems.

6. What is networked information appliances? (Remembering) (CO1)

Embedded systems that are provided with network interfaces and accessed by networks such as local area network or the internet are called networked information appliances.

7. What are the typical characteristics of an embedded system? (Remembering) (CO1)

- Sophisticated functionality.
- Real-time operation.
- Low manufacturing cost.
- Low power.
- Designed to tight deadlines by small teams.

8. What are the real-time requirements of an embedded system? (Remembering) (CO1)

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 real time 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.

9. Compare top-down and bottom-up design. (Analyzing)(AU-April 2014) (CO1)

Top-Down:

- Top down design proceeds from the abstract entity to get to the concrete design.
- It is most often used in designing brand new systems.

Bottom-Up:

- Bottom-up design proceeds from the concrete design to get to the abstract entity.
- It is sometimes used when one is reverse engineering a design, (i.e) when one is trying to figure out what somebody else designed in an existing design.

10. What are the ways to design a digital system in embedded computers? (Remembering) (CO1)

There are many ways to design a digital system. They are,

(i). Custom logic (ii). Field Programmable Gate Array (FPGA)..Etc

11. Discuss various issues in real time computing. (Creating)(AU-Nov/Dec 2013) (CO1)

The various issues in RT computing is :

- Real -time Response
- Recovering from failures
- Working with distributed architecture
- Asynchronous communication
- Race condition and timing.

12. Define microprocessor. (Remembering) (CO1)

A microprocessor fetches and processes the set of general-purpose instructions such as data transfer, ALU operations, stack operations, I/O operations and other program control operations.

13. Why we use microprocessor in embedded computing process? (Remembering) (CO1)

(i). Microprocessor are a very efficient way to implement digital systems.

(ii). Microprocessor makes it easier to design families of products that can be built to provide various feature sets at different price points and can be extended to provide new features to keep up with rapidly changing networks.

14. List the challenges faced in embedded computing system design? (Analyzing) (June 2016) (CO1)

(i). How much hardware do we need?

(ii). How do we meet our deadlines?

(iii). How do we minimize power consumption?

(iv). How do we design for upgradeability?

(v). Does it really work?

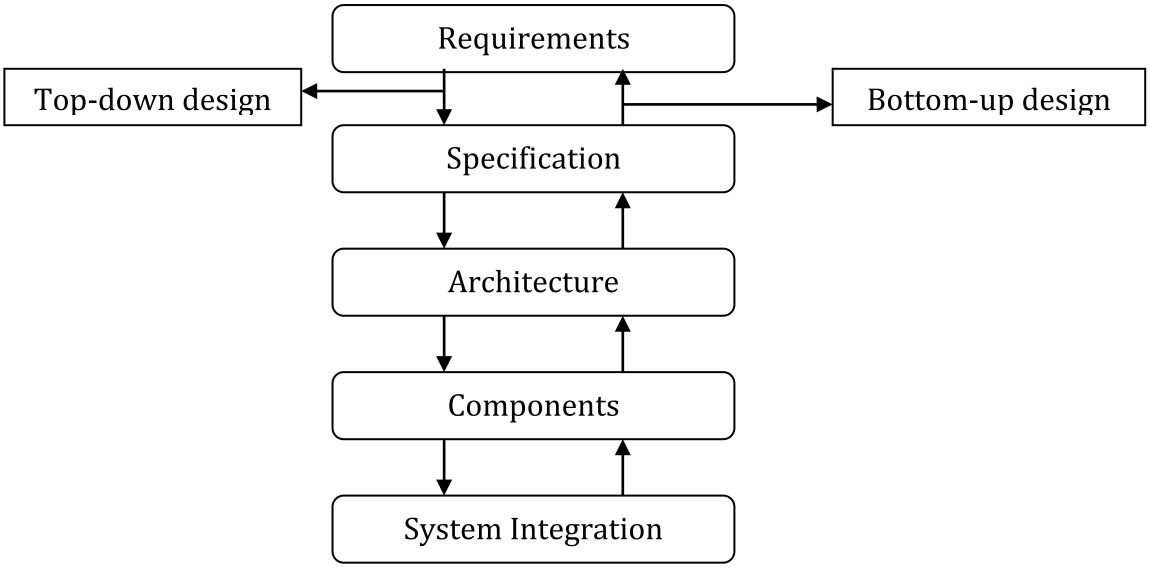
15. What are major goals of the embedded system design process? (Remembering) (CO1)

- Performance (both Overall speed and deadlines)
- Functionality and user interface.
- Manufacturing cost.
- Power consumption.
- Other requirements (physical size, etc.)

16. Show some examples of functional requirements? (Understanding) (CO1)

- Performance
- Cost
- Physical size and weight
- Power consumption

17. What are the steps or levels of abstraction involved in the embedded system design process? (Remembering) (CO1)



18. **What is mean by requirements and its types? (Remembering) (CO1)**

Before designing a system, it must to understand what has to be designed. This can be known from the starting steps of a design process.

It can be classified into two types. They are

(i). Functional requirements - It says the fundamental functions of a system.

(ii). Non functional requirements – It deals with the certain parameters to be considered in an embedded systems.

19. **How validating requirements are used in embedded system design process? (Remembering) (CO1)**

Validating is a set of requirements is ultimately a psychological task since it requires understanding both what people want and how they communicate those needs. One good way to refine at least the user interface portion of a system's requirements is to build a mock-up.

The mock-up may use canned data to simulate functionality in a restricted demonstration, and it may be executed on a PC or a workstation.

20. **What is the use of requirements form? (Remembering) (CO1)**

It is used as a checklist in the requirements analysis. From this the fundamental properties of a system came to be known.

21. **What are the entries of a requirement form? (Remembering) (CO1)**

- Name
- Purpose
- Inputs and outputs
- Functions
- Performance
- Manufacturing cost
- Power
- Physical size and weight

22. **What is meant by specification? (Remembering) (CO1)**

This is a contract between the Customer and the Architect. It conveys the customer's needs. These needs are properly used in the design process.

23. **What are the various components considered in the specification of GPS system? (Remembering) (CO1)**

- Data received from the GPS satellite constellation
- Map data
- User interface
- Operations that must be performed to satisfy customer requests
- Background actions required to keep system running, such as operating the GPS receiver.

24. **What is architecture design? (Remembering) (CO1)**

It says the way of implementing functions by a system. Actually architecture is a plan for the overall structure of the system that will be used later to design the components that make up the architecture.

25. **What are the major components to satisfy the specification? (Remembering) (CO1)**

There are two components are used to satisfy the specification. They are,

(i). **Hardware components:** It consists of CPU surrounded by memory and I/O devices. It may choose to use two memories. A frame buffer for the pixels to be displayed and a separate program/data memory for general use by the CPU.

(ii). **Software component:** It consists of timer to control when we read the buttons on the user interface and render data onto the screen.

26. **Define UML (Unified Modeling Language). (Remembering) (CO1)**

UML is an object oriented modeling language. It is a visual language that can be used to capture all these design tasks. It was designed to be useful at many levels of abstraction in the design process and it encourages design by successive refinement & progressively adding detail to the design.

27. **Define system integration? (Remembering) (CO1)**

It is a processor of combining the components into one system.

28. **What is the importance of object oriented system design? (Remembering) (CO1)**

(i). It encourages the design to be described as a number of interacting objects.

(ii). Design in terms of actual objects helps to understand the natural structure of the system.

29. **Define class (Remembering) (CO1)**

It is defined as the operation that determines how the object interacts with the rest of the world. All objects are derived from the same class have the same characteristics, although their attributes may have different values.

30. **Explain the types of relationship between objects and classes. (Understanding) (CO1)**

Association: objects communicate but one does not own the other.

Aggregation: a complex object is made of several smaller objects.

Composition: type of aggregation in which owner does not allow access to its components.

Generalization: define one class in terms of another.

31. **What is derived class? (Remembering) (CO1)**

A derived class inherits all the attributes and operations from its **base class.**

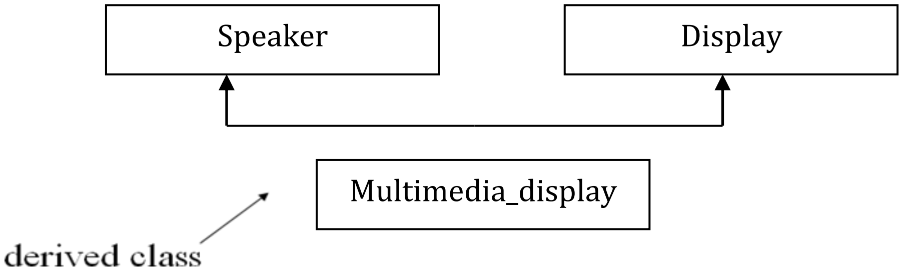
32. **What is mean by link and association? (Remembering) (CO1)**

Link - describe a relationship between objects

Association - describes relationship between classes.

33. **Define multiple inheritances. (Remembering) (CO1)**

It is defined as in which a class is derived from more than one base class.



34. **What is mean by stereo type in UML language? (Remembering) (CO1)**

It is a combination of elements in an object or class many times; we can give these patterns names, which are called stereotype in UML.

The stereotype name is written in the form **<<signal>>**

35. **Classify the types of events in behavioral description in UML language. (Understanding) (CO1)**

There are three events in behavioral description. They are,

(i). **Signal**-asynchronous event.

(ii). **Call**-follows the model of a procedure call in a programming language (Synchronized communication).

(iii). **Time-out**-causes the machine to leave a state after a certain amount of time.

36. **What is the relationship between an object-oriented specification and an object oriented programming language? (Remembering) (CO1)**

A specification language may not be executable. But both object-oriented specification and programming languages provide similar basic methods for structuring large systems.

37. **Explain sequence diagram. (Understanding) (CO1)**

A sequence diagram is somewhat similar to a hardware timing diagram, although the time flows vertically in a sequence diagram, whereas time typically flows horizontally in a timing diagram. The sequence diagram is designed to show a particular scenario or choice of events-it is not convenient for showing a number of mutually exclusive possibilities. In this case, the sequence shows what happens when a mouse click is on the menu region.

38. List out the basic requirements of Model Train controller. (Analyzing) (CO1)

- The console shall be able to control up to eight trains on a single track.
- The speed of each train shall be controllable by a throttle to at least 63 different levels in each direction (forward and reverse).
- There shall be an inertia control that shall allow the user to adjust the responsiveness of the train to commanded changes in speed. Higher inertia means that the train responds more slowly to a change in the throttle, simulating the inertia of a large train. The inertia control will provide at least eight different levels.
- There shall be an emergency stop button.
- An error detection scheme will be used to transmit messages.

39. List the requirements model for Model train controller model. (Analyzing) (CO1)

Name	-	Model train controller
Purpose	-	Control speed of up to eight model trains
Inputs	-	Throttle, inertia setting, emergency stop, train number
Outputs	-	Train control signals
Functions	-	Set engine speed based upon inertia settings; respond to emergency stop
Performance	-	Can update train speed at least 10 times per second
Manufacturing cost	-	$50
Power	-	10W (plugs into wall)
Physical size and weight	-	Console should be comfortable for two hands, approximate size of standard keyboard; weight <2 pounds.

40. Classify the different types of documents used in DCC in controller. (Analyzing) (CO1)

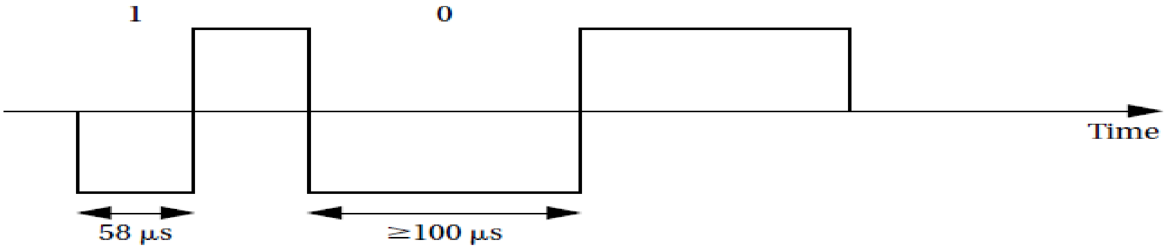
The DCC standard is given in two documents:

(i). Standard S-9.1, the DCC Electrical Standard, defines how bits are encoded on the rails for transmission.

(ii).Standard S-9.2, the DCC Communication Standard, defines the packets that carry information.

41. How the voltage and power can be take part in the DCC electrical standard? (Remembering) (CO1)

The Electrical Standard deals with voltages and currents on the track. The data signal swings between two voltages around the power supply voltage. The bits are encoded in the time between transitions, not by voltage levels. A 0 is at least 100 µs while a 1 is nominally 58 µs.



The durations of the high (above nominal voltage) and low (below nominal voltage) parts of a bit are equal to keep the DC value constant. The specification also gives the allowable variations in bit times that a conforming DCC receiver must be able to tolerate. The standard also describes other electrical properties of the system, such as allowable transition times for signals.

42. Define DCC (Digital Command Control) in controller. (Remembering) (CO1)

The Digital Command Control (DCC) standard was created by the National Model Railroad Association to support interoperable digitally-controlled model trains. Hobbyists started building homebrew digital control systems in the 1970s and Marklin developed its own digital control system in the 1980s. DCC was created to provide a standard that could be built by any manufacturer so that hobbyists could mix and match components from multiple vendors.

43. Inspect the basic packet format for DCC communication standards. (Analyzing) (CO1)

We can write the basic packet format as a regular expression:

PSA (sD) + E

P is the preamble, which is a sequence of at least 10 1 bits. The command station should send at least 14 of these 1 bits, some of which may be corrupted during transmission.

S is the packet start bit. It is a 0 bit.

A is an address data byte that gives the address of the unit, with the most significant bit of the address transmitted first. An address is eight bits long.

The addresses 00000000, 11111110, and 11111111 are reserved.

s is the data byte start bit, which, like the packet start bit, is a 0.

D is the data byte, which includes eight bits. A data byte may contain an address, instruction, data, or error correction information.

E is a packet end bit, which is a 1 bit.

44. What type of packets used in Digital Command Control? (Remembering) (CO1)

Baseline packet:

A ***baseline packet*** is the minimum packet that must be accepted by all DCC implementations. More complex packets are given in a Recommended Practice document.

A baseline packet has three data bytes:

- an address data byte that gives the intended receiver of the packet;
- the instruction data byte provides a basic instruction; and
- an error correction data byte is used to detect and correct transmission errors.

45. Discuss about Von Neumann and Harvard Architecture. (Creating) (CO1)

Von Neumann architecture:

- A computer whose memory holds both data and instructions is known as a ***von Neumann*** machine.
- The ***program counters (PC)***, which holds the address in memory of an instruction.
- The CPU fetches the instruction from memory, decodes the instruction, and executes it.

Harvard Architecture:

- Harvard machine has separate memories for data and program.
- The program counter points to program memory, not data memory. As a result, it is harder to write self-modifying programs (programs that write data values, and then use those values as instructions) on Harvard machines.

46. Explain RISC and CISC. (Understanding) (CO1)

RISC (reduced instruction set computers):

These computers tended to provide somewhat fewer and simpler instructions. The instructions were also chosen so that they could be efficiently executed in ***pipelined*** processors. Early RISC designs substantially outperformed CISC designs of the period. As it turns out, we can use RISC techniques to efficiently execute at least a common subset of CISC instruction sets, so the performance gap between RISC-like and CISC-like instruction sets has narrowed somewhat.

CISC (complex instruction set computers):

These machines provided a variety of instructions that may perform very complex tasks, such as string searching; they also generally used a number of different instruction formats of varying lengths.

47. List out the variety of characteristics used in instruction set preliminaries. (Analyzing) (CO1)

Instructions can have a variety of characteristics, including:

- Fixed versus variable length.
- Addressing modes.
- Numbers of operands.
- Types of operations supported.

48. **Define Programming Model. (Remembering) (CO1)**

The set of registers available for use by programs is called the programming model, also known as the programmer model. (The CPU has many other registers that are used for internal operations and are unavailable to programmers.)

49. **Analyze some features or parameter used in assembly language. (Analyzing) (CO1)**

Assembly languages usually share the same basic features:

- One instruction appears per line.
- ***Labels***, which give names to memory locations, start in the first column.
- Instructions must start in the second column or after to distinguish them from labels.
- Comments run from some designated comment character (; in the case of ARM) to the end of the line.

50. **Explain about ARM processor. (Understanding) (AU-Nov/Dec 2013) (CO1)**

ARM – Advanced RISC Machine. (It is an 32-bit Microprocessor)

ARM is actually a family of RISC architectures that have been developed over many years. The ARM is a 32-bit Reduced Instruction Set Computer (RISC) instruction set architecture developed by Arm holdings. ARM processor is mad suitable for Low power application.

ARM is actually a family of RISC architectures that have been developed over many years. ARM does not manufacture its own VLSI devices; rather, it licenses its architecture to companies who either manufacture the CPU itself or integrate the ARM processor into a larger system.

51. **List the functions of ARM processor in supervisor mode. (Analyzing)(AU-April 2014) (CO1)**

The various functions of ARM processor in supervisor modes are:

- Exception
- Prioritization
- Vectoring
- Traps.

52. **Define Assembly language. (Remembering) (CO1)**

The textual description of instructions, as opposed to their binary representation, is called an assembly language.

53. **List some example program for ARM processor using assembly language. (Analyzing) (CO1)**

ARM instructions are written one per line, starting after the first column. Comments begin with a semicolon and continue to the end of the line. A label, which gives a name to a memory location, comes at the beginning of the line, starting in the first column.

Here is an example:

```
          LDR r0,[r8]; a comment
label     ADD r4,r0,r1
```

54. **Identify the versions used in ARM processor with different architecture. (Applying) (CO1)**

- ARM7 is a von Neumann architecture machine
- ARM9 uses a Harvard architecture.
- However, this difference is invisible to the assembly language programmer, except for possible performance differences.

55. **Classify the different types of data used in ARM architecture. (Understanding) (CO1)**

The ARM architecture supports two basic types of data:

- The standard ARM word is 32 bits long.
- The word may be divided into four 8-bit bytes.

ARM7 allows addresses up to 32 bits long. An address refers to a byte, not a word. Therefore, the word 0 in the ARM address space is at location 0, the word 1 is at 4, the word 2 is at 8, and so on. (As a result, the PC is incremented by 4 in the absence of a branch.)

56. **What is the Instruction set features useful for embedded programming? (Remembering) (AU-May/June 2013) (CO1)**

Instruction Sets can have a variety of characteristics/features including:

- Fixed versus variable length
- Addressing modes
- Number of operands
- Types of operations supported.

57. What are the different types of data operations used in ARM Processor? (Remembering) (CO1)

- Arithmetic and logical operations (ADD-Add, ADC-Add with carry, SUBC-Subtract with carry, etc.,)
- Load and store instructions (LDR-Load, STR-Store, LDRH- Load half word, etc.,)
- Shift/rotate instruction (LSL-Logical shift left, LSR-Logical shift right, etc.,)
- ARM comparison instruction (CMP-Compare, CMN-Negated compare, TST-Bit wise test, TEQ-Bit wise negated test)
- Move instruction (MOV-Move, MVN-Move negated)

58. What is meant by current program status register (CPSR) and represent its bits? (Remembering) (CO1)

The other important basic register in the programming model is the current program status register (CPSR). This register is set automatically during every arithmetic, logical, or shifting operation.

The top four bits of the CPSR hold the following useful information about the results of that arithmetic/logical operation:

- The negative (N) bit is set when the result is negative in two's-complement arithmetic.
- The zero (Z) bit is set when every bit of the result is zero.
- The carry (C) bit is set when there is a carry out of the operation.
- The overflow (V) bit is set when an arithmetic operation results in an overflow.

59. Define branch instructions. (Remembering) (CO1)

The B (branch) instruction is the basic mechanism in ARM for changing the flow of control. The address that is the destination of the branch is often called the branch target. Branches are PC-relative—the branch specifies the offset from the current PC value to the branch target. The offset is in words, but because the ARM is byte addressable, the offset is multiplied by four (shifted left two bits, actually) to form a byte address. Thus, the instruction

B #100

Will add 400 to the current PC value.

60. What is programmable input and output devices and also mention the registers in I/P & O/P devices? (Remembering) (CO1)

Input and output devices usually have some analog or non electronic component-for instance, a disk drive has a rotating disk and analog read/write electronics. But the digital logic in the device that is most closely connected to the CPU very strongly resembles the logic you would expect in any computer system.

The interface between the CPU and the device's internals (e.g., the rotating disk and read/write electronics in a disk drive) is a set of registers. The CPU talks to the device by reading and writing the registers. Devices typically have several registers:

- **Data registers** hold values that are treated as data by the device, such as the data read or written by a disk.
- **Status registers** provide information about the device's operation, such as whether the current transaction has completed.

61. How Microprocessors can provide programming support for input and output devices? (Remembering) (CO1)

Microprocessors can provide programming support for input and output in two ways:

- I/O instructions
- memory-mapped I/O

Some architectures, such as the Intel x86, provide special instructions (in and out in the case of the Intel x86) for input and output. These instructions provide a separate address space for I/O devices.

But the most common way to implement I/O is by memory mapping-even CPUs that provide I/O instructions can also implement memory-mapped I/O. As the name implies, memory-mapped I/O provides addresses for the registers in each I/O device.

62. Define polling or Busy-Wait I/O (Remembering) (June 2016) (CO1)

The most basic way to use devices in a program is ***busy-wait I/O***. Devices are typically slower than the CPU and may require many cycles to complete an operation.

If the CPU is performing multiple operations on a single device, such as writing several characters to an output device, then it must wait for one operation to complete before starting the next one. (If we try to start writing the second character before the device has finished with the first one, for example, the device will probably never print the first character.) Asking an I/O device whether it is finished by reading its status register is often called ***polling***.

63. What is meant by Interrupt and interrupt handler or device driver? (Remembering) (CO1)

Interrupt:

The interrupt mechanism allows devices to signal the CPU and to force execution of a particular piece of code.

Interrupt handler or device driver:

When an interrupt occurs, the program counter's value is changed to point to an interrupt handler routine (also commonly known as a device driver) that takes care of the device: writing the next data, reading data that have just become ready, and so on. The interrupt mechanism of course saves the value of the PC at the interruption so that the CPU can return to the program that was interrupted. Interrupts therefore allow the flow of control in the CPU to change easily between different contexts, such as a foreground computation and multiple I/O devices.

64. List out the interface between the CPU and I/O device includes the following signals for interrupting. (Analyzing) (CO1)

- The I/O device asserts the ***interrupt request*** signal when it wants service from the CPU; and
- The CPU asserts the ***interrupt acknowledge*** signal when it is ready to handle the I/O device's request.

65. Define foreground program. (Remembering) (CO1)

The program that runs when no interrupt is being handled is often called the foreground program; when the interrupt handler finishes, it returns to the foreground program, wherever processing was interrupted.

66. Explain about priorities and vectors. (Understanding) (CO1)

Providing a practical interrupt system requires having more than a simple interrupt request line. Most systems have more than one I/O device, so there must be some mechanism for allowing multiple devices to interrupt. We also want to have flexibility in the locations of the interrupt handling routines, the addresses for devices, and so on.

There are two ways in which interrupts can be generalized to handle multiple devices and to provide more flexible definitions for the associated hardware and software:

- ***Interrupt priorities*** allow the CPU to recognize some interrupts as more important than others, and
- ***Interrupt vectors*** allow the interrupting device to specify its handler.

67. Define masking. (Remembering) (CO1)

The priority mechanism must ensure that a lower-priority interrupt does not occur when a higher-priority interrupt is being handled. The decision process is known as ***masking***.

The highest-priority interrupt is normally called the ***non maskable interrupt (NMI)***.The NMI cannot be turned off and is usually reserved for interrupts caused by power failures-a simple circuit can be used to detect a dangerously low power supply, and the NMI interrupt handler can be used to save critical state in nonvolatile memory, turn off I/O devices to eliminate spurious device operation during power down, and so on.

68. What is meant by Interrupt overhead and determine the steps involved in the process? (Remembering) (CO1)

Now that we have a basic understanding of the interrupt mechanism, we can consider the complete interrupt handling process. Once a device requests an interrupt, some steps are performed by the CPU, some by the device, and others by software. Here are the major steps in the process:

- ***CPU:*** The CPU checks for pending interrupts at the beginning of an instruction. It answers the highest-priority interrupt, which has a higher priority than that given in the interrupt priority register.
- ***Device:*** The device receives the acknowledgment and sends the CPU its interrupt vector.
- ***CPU:*** The CPU looks up the device handler address in the interrupt vector table using the vector as an index. A subroutine-like mechanism is used to save the current value of the PC and possibly other internal CPU state, such as general-purpose registers.
- ***Software:*** The device driver may save additional CPU state. It then performs the required operations on the device. It then restores any saved state and executes the interrupt return instruction.
- ***CPU:*** The interrupt return instruction restores the PC and other automatically saved states to return execution to the code that was interrupted.

69. Describe about the interrupt used in ARM7. (CO1)

ARM7 supports two types of interrupts:

- Fast interrupt requests (FIQs) and
- Interrupt requests (IRQs).

An FIQ takes priority over an IRQ. The interrupt table is always kept in the bottom memory addresses, starting at location 0.The entries in the table typically contain subroutine calls to the appropriate handler.

70. Develop the steps involved in the interrupt in ARM7 processor. (Applying) (CO1)

The ARM7 performs the following steps when responding to an interrupt

- Saves the appropriate value of the PC to be used to return,
- Copies the CPSR into a saved program status register (SPSR),
- Forces bits in the CPSR to note the interrupt, and
- Forces the PC to the appropriate interrupt vector.

When leaving the interrupt handler, the handler should:

- Restore the proper PC value,
- Restore the CPSR from the SPSR, and
- Clear interrupt disable flags.

71. List out the interrupt used in C55xprocessor. (Analyzing) (CO1)

A maskable interrupt is processed in several steps once the interrupt request is sent to the CPU:

- The interrupt flag register (IFR) corresponding to the interrupt is set.
- The interrupt enable register (IER) is checked to ensure that the interrupt is enabled.
- The interrupt mask register (INTM) is checked to be sure that the interrupt is not masked.
- The interrupt flag register (IFR) corresponding to the flag is cleared.
- Appropriate registers are saved as context.
- INTM is set to 1 to disable maskable interrupts.
- DGBM is set to 1 to disable debug events.
- EALLOW is set to 0 to disable access to non-CPU emulation registers.
- A branch is performed to the interrupt service routine (ISR).

72. Explain the mechanism used in C55x processor. (Understanding) (CO1)

The C55x provides two mechanisms

- fast-return and
- slow-return

To save and restore registers for interrupts and other context switches. Both processes save the return address and loop context registers.

The fast-return mode uses RETA to save the return address and CFCT for the loop context bits.

The slow return mode, in contrast, saves the return address and loop context bits on the stack.

73. What is meant by Supervisor mode, Exception and traps? (Remembering) (AU-Nov/Dec 2012) (AUT-Dec 2015) (CO1)

Supervisor mode:

In such cases it is often useful to have a **supervisor mode** provided by the CPU. Normal programs run in **user mode**. The supervisor mode has privileges that user modes do not.

Exception:

An **exception** is an internally detected error. A simple example is division by zero. One way to handle this problem would be to check every divisor before division to be sure it is not zero, but this would both substantially increase the size of numerical programs and cost a great deal of CPU time evaluating the divisor’s value. The CPU can more efficiently check the divisor’s value during execution. Since the time at which a zero divisor will be found is not known in advance, this event is similar to an interrupt except that it is generated inside the CPU. The exception mechanism provides a way for the program to react to such unexpected events.

Traps:

A **trap**, also known as a **software interrupt**, is an instruction that explicitly generates an exception condition. The most common use of a trap is to enter supervisor mode.

74. What is the function of exception? (Remembering) (AU-Nov/Dec 2012) (CO1)

- The main function of exception is to detect the error internally.
- It requires both prioritization and vectoring.

75. Define co-processors. (Remembering) (CO1)

CPU architects often want to provide flexibility in what features are implemented in the CPU. One way to provide such flexibility at the instruction set level is to allow co-processors, which are attached to the CPU and implement some of the instructions.

For example, floating-point arithmetic was introduced into the Intel architecture by providing separate chips that implemented the floating-point instructions.

76. What is meant by cache in memory system techniques and its conditions? (Remembering) (CO1)

Caches are widely used to speed up memory system performance. A cache is a small, fast memory that holds copies of some of the contents of main memory.

A ***cache controller*** mediates between the CPU and the memory system comprised of the main memory. The cache controller sends a memory request to the cache and main memory.

If the requested location is in the cache, the cache controller forwards the location’s contents to the CPU and aborts the main memory request; this condition is known as a ***cache hit***.

If the location is not in the cache, the controller waits for the value from main memory and forwards it to the CPU; this situation is known as a ***cache miss***.

77. Classify the cache misses in memory system mechanisms. (Understanding) (CO1)

We can classify cache misses into several types depending on the situation that generated them:

- a **compulsory miss** (also known as a **cold miss**) occurs the first time an location is used,
- a **capacity miss** is caused by a too-large working set, and
- a **conflict miss** happens when two locations map to the same location in the cache.

78. Define bit rate and miss rate. (Remembering) (CO1)

Let *h* be the ***hit rate***, the probability that a given memory location is in the cache. It follows that 1-*h* is the ***miss rate***, or the probability that the location is not in the cache. Then we can compute the average memory access time as

$$t_{\text{av}} = ht_{\text{cache}} + (1-h)t_{\text{main}}.$$

where t_{cache} is the access time of the cache and t_{main} is the main memory access time.

79. Classify the different types of levels in cache memory. (Understanding) (CO1)

- The **first-level cache** (commonly known as **L1 cache**) is closest to the CPU, the
- **second-level cache (L2 cache)** feeds the first-level cache, and so on.

80. Explain the term write through and write back in cache techniques. (Understanding) (CO1)

Write through:

Every write changes both the cache and the corresponding main memory location (usually through a write buffer). This scheme ensures that the cache and main memory are consistent, but may generate some additional main memory traffic.

Write back:

We can reduce the number of times we write to main memory by using a ***write-back*** policy: If we write only when we remove a location from the cache, we eliminate the writes when a location is written several times before it is removed from the cache.

81. Define unified cache (Remembering) (CO1)

A cache that holds both instructions and data is called a unified cache.

82. Define page fault (Remembering) (CO1)

When the CPU requests an address that is not in main memory, the MMU generates an exception called a page fault.

The program that generated the page fault is restarted by the handler only after,

- The required memory has been read back into main memory, and
- The MMU's tables have been updated to reflect the changes.

83. What is meant by address translation and its styles? (Remembering) (CO1)

A MMU translates addresses between the CPU and physical memory. This translation process is often known as ***memory mapping*** since addresses are mapped from a logical space into a physical space.

There are two styles of address translation:

- segmented and
- paged

Each has advantages and the two can be combined to form a segmented, paged addressing scheme. A segment is usually described by its start address and size, allowing different segments to be of different sizes. Pages are of uniform size, which simplifies the hardware required for address translation.

84. Define translation lookaside buffer (TLB) (Remembering) (CO1)

The efficiency of paged address translation may be increased by caching page translation information. A cache for address translation is known as a translation lookaside buffer (TLB).The MMU reads the TLB to check whether a page number is currently in the TLB cache and, if so, uses that value rather than reading from memory.

85. What are memory regions for address translation in ARM MMU Techniques? (Remembering) (CO1)

ARM MMU supports the following types of memory regions for address translation:

- Section is a 1-MB block of memory,
- Large page is 64 KB, and
- Small page is 4 KB.

86. Define pipelining and list the different types of stages in this process. (Remembering) (CO1)

Modern CPUs are designed as pipelined machines in which several instructions are executed in parallel. Pipelining greatly increases the efficiency of the CPU. But like any pipeline, a CPU pipeline works best when its contents flow smoothly. Some sequences of instructions can disrupt the flow of information in the pipeline and, temporarily at least, slow down the operation of the CPU.

The ARM7 has a three-stage pipeline:

- **Fetch** the instruction is fetched from memory.
- **Decode** the instruction's opcode and operands are decoded to determine what function to perform.

- **Execute** the decoded instruction is executed.

87. **What are the parameters used to evaluate the CPU performance? (Remembering)(AU-May/June 2013) (CO1)**
    - Pipelining and
    - Caching.

88. **Define latency and throughput. (Remembering) (CO1)**

    A normal instruction requires three clock cycles to completely execute, known as the **latency** of instruction execution.

    But since the pipeline has three stages, an instruction is completed in every clock cycle. In other words, the pipeline has a **throughput** of one instruction per cycle.

89. **List the seven stages in C55x processor. (Analyzing) (CO1)**

    The C55x includes a seven-stage pipeline:
    - Fetch.
    - Decode.
    - Address computes data and branch addresses.
    - Access 1 reads data.
    - Access 2 finishes data read.
    - Read stage puts operands onto internal busses.
    - Execute performs operations.

90. **Explain the concept of CPU power consumption and its basic sources. (Understanding) (CO1)**

    The high-level power consumption characteristics of CPUs and other system components are derived from the circuits used to build those components. Today, virtually all digital systems are built with complementary metal oxide semiconductor (CMOS) circuitry.

    The basic sources of CMOS power consumption is,
    - **Voltage drops:** The dynamic power consumption of a CMOS circuit is proportional to the square of the power supply voltage (V2).
    - **Togging:** A CMOS circuit uses most of its power when it is changing its output value.
    - **Leakage:** Even when a CMOS circuit is not active, some charge leaks out of the circuit's nodes through the substrate.

91. **Classify the different types of power management provided by CPU. (Understanding) (CO1)**

    There are two types of power management features provided by CPUs.

    A **static power management** mechanism is invoked by the user but does not otherwise depend on CPU activities. An example of a static mechanism is a power down mode intended to save energy. This mode provides a high-level way to reduce unnecessary power consumption.

    A **dynamic power management** mechanism takes actions to control power based upon the dynamic activity in the CPU. For example, the CPU may turn off certain sections of the CPU when the instructions being executed do not need them.

92. **How is ARM processor different from other processors? (Remembering)(AU-Nov/Dec 2012) (CO1)**
    - ARM is a RISC (Reduced Instruction Set Computing) architecture while other processor being a CISC (Computer Instruction Set Computing) one.
    - In the ARM processor, arithmetic and logical operations cannot be perform directly on memory locations, while other processors allow such operations to directly reference main memory.

93. **When is application specific system processor (ASSPs) used in embedded systems? (Remembering) (AU-May/June 2012) (CO1)**

    ASSP is a processing unit for specific task and for specific application. In embedded system for example image compression and that is integrated through the buses with the main processor in an embedded system.

94. **What are the various in embedded system designs modeling refining (or) partitioning? (Remembering) (AU-May/June 2012) (CO1)**
    - Structural modeling

- Behavior modeling
- State machine modeling
- Process algebra modeling
- Logic based modeling
- Petri-nets modeling.

95. State the difference between top down and bottom up design approach in embedded system. (Evaluating) (Dec 2015) (CO1)

TOP DOWN DESIGN	BOTTOM UP DESIGN
It will begin with the most abstract description of the systems and conclude with details.	It starts with components to build a system decisions at one stage of design are based upon estimates of what will happen later.

BIG QUESTIONS:

1. Discuss in detail about complex systems and microprocessors**. (Creating) (CO1)**
2. Discuss about the requirements, specification and architectural design in the process of embedded system design. **(Creating) (CO1) OR**

   Discuss the top down method embedded system design development with suitable example. **(Creating) (June 2016) (CO1)**
3. Explain in detail about various mechanisms in formalisms for system design. **(Understanding) (CO1) OR**

   Discuss in detail about Structural and Behavioral description of methods used for designing an embedded system. **(Creating) (AU-May/June 2013) (CO1)**
4. List the description method in UML and explain any one method. **(Analyzing) (June 2016) (CO1)**
5. Classify the various stages involved in the design of model train controller**. (Analyzing) (AU-Nov/Dec 2012, May/June 2013, April 2014) (CO1) OR**

   Draw a model train control system and elaborate about the specification using the class diagram. **(Creating) (Dec 2015) (CO1)**
6. Explain the design process details of an embedded system with some specific example. (**Evaluating) (AU-Nov/Dec 2013) (CO1)**
7. Explain in detail about instruction set preliminaries with examples. **(Understanding) (AU-Nov/Dec 2013,Nov/Dec 2014) (CO1)**
8. Discuss in detail the embedded system design by considering ARM processor and its memory organization as a design examples. **(Creating) (AU-May/June 2013,Nov/Dec 2014) (CO1)**
9. Explain in detail about different types of data operations in ARM Processor with examples. **(Understanding)(AU-May/June 2013) (CO1)**
10. Summarize the concept about programming input and output devices in CPU and its primitives. **(Understanding) (CO1)**
11. Explain the terms: Supervisor mode, Exceptions and Traps in detail. **(Understanding) (AU-Nov/Dec 2013) (CO1)**
12. What is Coprocessor? Write about its significance. **(Remembering) (Dec 2015) (CO1)**
13. Elaborate in detail about the Memory system mechanisms and its types. **(Creating) (CO1)**
14. How to evaluate CPU Performance? **(Remembering) (AU-Nov/Dec 2014) (CO1)**
15. Justify that what type of power can be consumed in CPU with different processors. **(Evaluating) (CO1)**
16. What are the various goals and tasks of the embedded system design process? Illustrate with an example. **(Understanding) (Dec 2015) (CO1)**
17. Describe the direct mapped and set associative cache map with examples. **(Understanding) (June 2016) (CO1)**

UNIT - II

COMPUTING PLATFORM AND DESIGN ANALYSIS

1. What is CPU Bus? (Remembering) (CO2)

A computer system encompasses much more than the CPU; it also includes memory and I/O devices. The bus is the mechanism by which the CPU communicates with memory and devices. A bus is, at a minimum, a collection of wires, but the bus also defines a protocol by which the CPU, memory, and devices communicate.

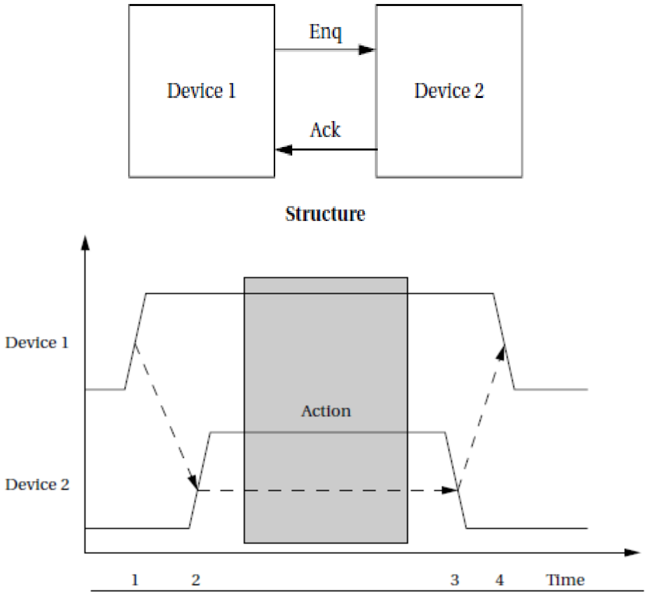
2. What is the major role of bus? (Remembering) (CO2)

One of the major roles of the bus is to provide an interface to memory.

3. Construct basic building block of most bus protocols. (Applying) (CO2)

The basic building block of most bus protocols is the **four-cycle handshake**

The handshake uses a pair of wires dedicated to the handshake: ***enq*** (meaning enquiry) and ***ack*** (meaning acknowledge). Extra wires are used for the data transmitted during the handshake.



Structure

Behavior

4. What is four-cycle handshake? (Remembering) (June 2016) (CO2)

The four cycles are described below.

1. *Device 1* raises its output to signal an enquiry, which tells *device 2* that it should get ready to listen for data.
2. When *device 2* is ready to receive, it raises its output to signal an acknowledgment. At this point, *devices 1* and *2* can transmit or receive.
3. Once the data transfer is complete, *device 2* lowers its output, signaling that it has received the data.
4. After seeing that *ack* has been released, *device 1* lowers its output.

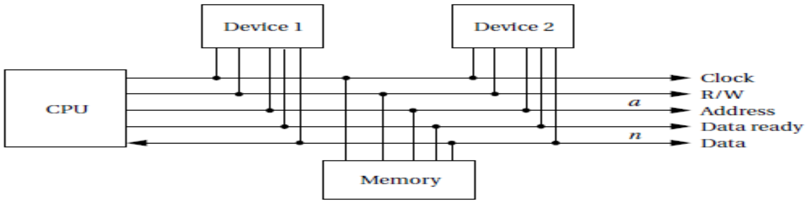
5. List the fundamental operation of bus. (Analyzing) (CO2)

The fundamental bus operations are reading and writing.

6. Define bus. (Remembering) (CO2)

The term *bus* is used in two ways. The most basic use is as a set of related wires, such as address wires. However, the term may also mean a protocol for communicating between components. To avoid confusion, we will use the term ***bundle*** to refer to a set of related signals.

7. **Construct typical microprocessor bus. (Creating) (CO2)**



8. **What are the major components in microprocessor bus? (Remembering) (CO2)**

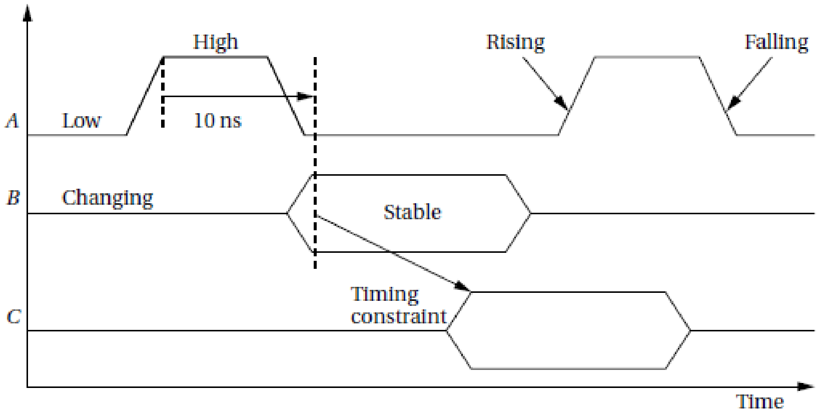
The major components follow:

- *Clock* provides synchronization to the bus components,
- *R/W* is true when the bus is reading and false when the bus is writing,
- *Address* is an *a*-bit bundle of signals that transmits the address for an access,
- *Data* is an *n*-bit bundle of signals that can carry data to or from the CPU, and
- *Data ready* signals when the values on the data bundle are valid.

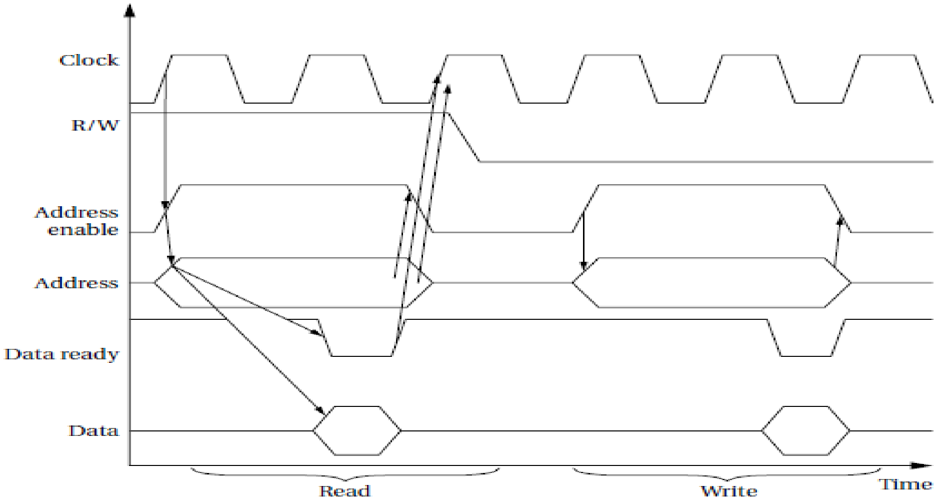
9. **Infer timing diagram. (Understanding) (CO2)**

The behavior of a bus is most often specified as a *timing diagram*. A timing diagram shows how the signals on a bus vary over time, but since values like the address and data can take on many values, some standard notation is used to describe signals.

10. **Show the timing diagram notation. (Understanding) (CO2)**



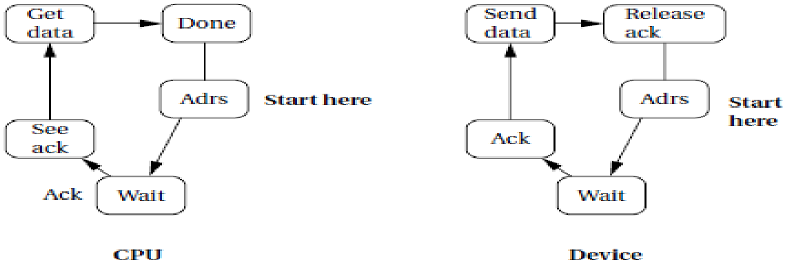
11. **Show the timing diagram for the example bus. (Understanding) (CO2)**



12. **Examine data ready signal. (Analyzing) (CO2)**

The data ready signal allows the bus to be connected to devices that are slower than the bus.

13. Demonstrate the State diagrams for the bus read transaction. (Understanding) (CO2)



14. What are wait states? (Remembering) (CO2)

The cycles between the minimum time at which data can be asserted and when it is actually asserted are known as **wait states**. Wait states are commonly used to connect slow, inexpensive memories to buses.

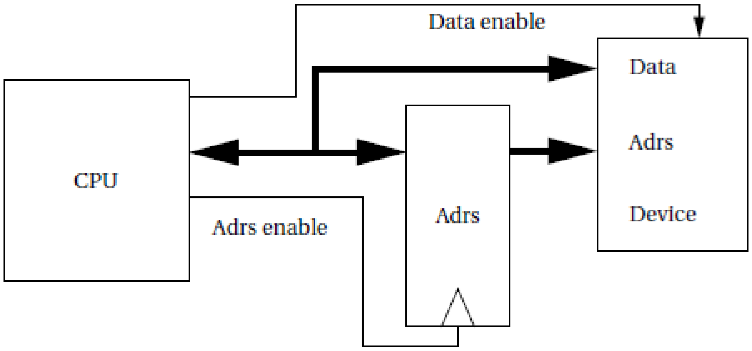
15. What is Direct Memory Access (DMA)? (Remembering) (CO2)

Direct memory access (DMA) is a bus operation that allows reads and writes not controlled by the CPU.

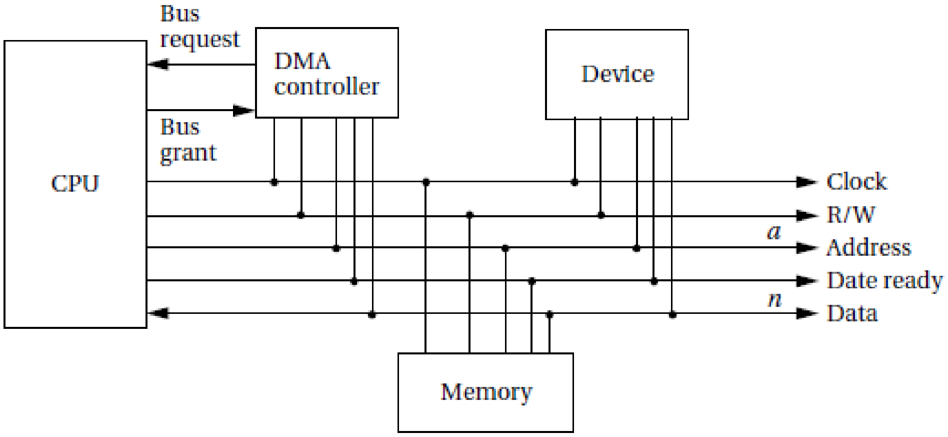
16. Explain about DMA controller. (Evaluating) (CO2)

A DMA transfer is controlled by a DMA controller, which requests control of the bus from the CPU.After gaining control, the DMA controller performs read and write operations directly between devices and memory.

17. Examine bus signals for multiplexing address and data. (Analyzing) (CO2)



18. Show the configuration of a bus with a DMA controller. (Remembering) (CO2)



19. What are two additional bus signals that DMA requires the CPU to provide? (Remembering) (CO2)

- The **bus request** is an input to the CPU through which DMA controllers ask for ownership of the bus.
- The **bus grant** signals that the bus has been granted to the DMA controller.

20. Summarize bus master. (Understanding) (CO2)

A device that can initiate its own bus transfer is known as a bus master. Devices that do not have the capability to be bus masters do not need to connect to a bus request and bus grant.

21. List the registers that are required in DMA controller for DMA operation. (Analyzing) (CO2)

The CPU controls the DMA operation through registers in the DMA controller. A typical DMA controller includes the following three registers:

1. A starting address register specifies where the transfer is to begin.
2. A length register specifies the number of words to be transferred.
3. A status register allows the DMA controller to be operated by the CPU.

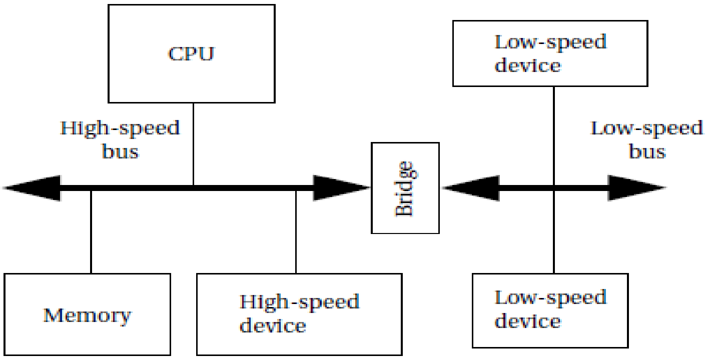
22. Define bridge. (Remembering) (CO2)

A small block of logic known as a bridge, allows the buses to connect to each other.

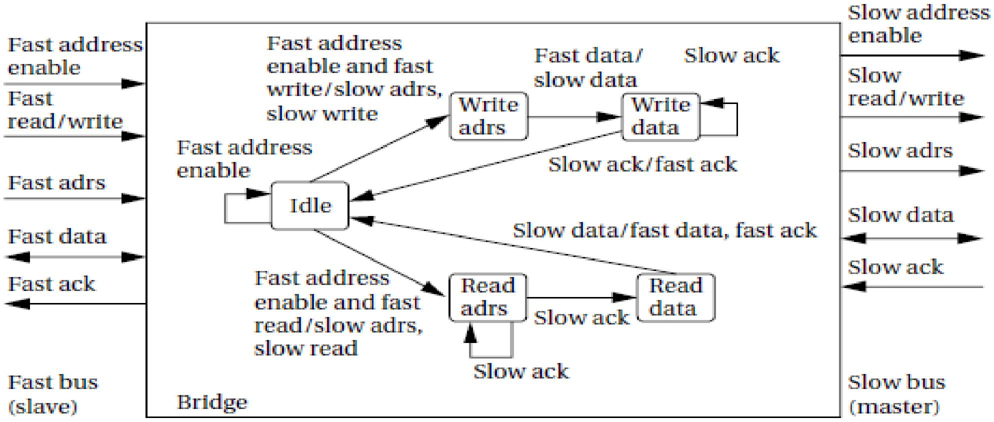
23. What are the reasons to use multiple buses and bridges? (Remembering) (CO2)

- Higher-speed buses may provide wider data connections.
- A high-speed bus usually requires more expensive circuits and connectors. The cost of low-speed devices can be held down by using a lower-speed, lower-cost bus.
- The bridge may allow the buses to operate independently, thereby providing some parallelism in I/O operations.

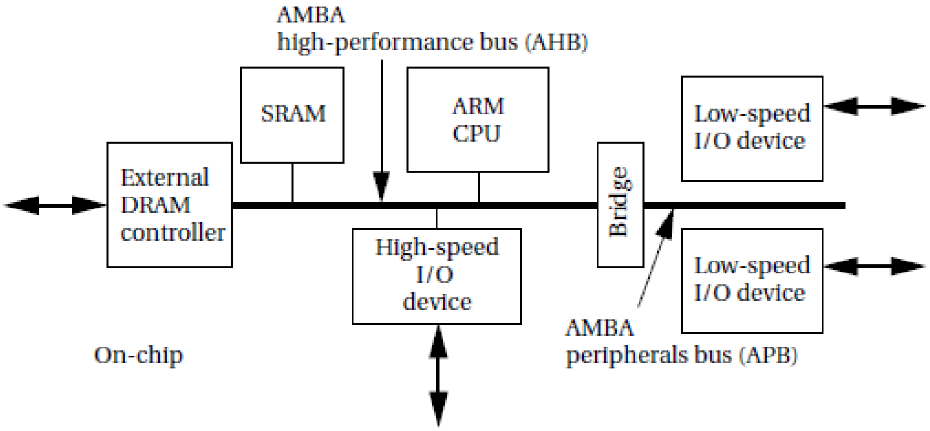
24. Build a multiple bus system.(Creating) (CO2)



25. Outline UML state diagram of bus bridge operation. (Understanding) (CO2)



26. Show the elements of the ARM AMBA bus system. (Remembering) (CO2)



27. **What is aspect ratio in memory? (Remembering) (CO2)**

The height/width ratio of a memory is known as its aspect ratio. The best aspect ratio depends on the amount of memory required.

28. **What is a RAM? (Remembering) (CO2)**

Random Access Memories (RAM's) can be both read and written. They are called random access because, unlike magnetic disks, addresses can be read in any order.

29. **What is DRAM? (Remembering) (CO2)**

Most bulk memory in modern systems is dynamic RAM (DRAM). DRAM is very dense; it does, however, require that its values be refreshed periodically since the values inside the memory cells decay over time.

30. **What is SRAM? (Remembering) (CO2)**

The dominant form of dynamic RAM today is the Synchronous DRAMs (SDRAMs), which use clocks to improve DRAM performance. SDRAMs use Row Address Select (RAS) and Column Address Select (CAS) signals to break the address into two parts, which select the proper row and column in the RAM array. Signal transitions are relative to the SDRAM clock, which allows the internal SDRAM operations to be pipelined.

31. **Write characteristics of SRAM and DRAM. (Remembering) (Dec 2015) (CO2)**

- SRAM is faster than DRAM.
- SRAM consumes more power than DRAM.
- More DRAMs can be put on a single chip.
- DRAM values must be periodically refreshed.

32. **Elaborate SIMMs and DIMMs. (Creating) (CO2)**

Memory for PCs is generally purchased as Single In Line Memory Modules (SIMMs) or Double In Line Memory Modules (DIMMs). A SIMM or DIMM is a small circuit board that fits into a standard memory socket. A DIMM has two sets of leads compared to the SIMM's one. Memory chips are soldered to the circuit board to supply the desired memory.

33. **What is ROM? (Remembering) (CO2)**

Read Only Memories (ROMs) are preprogrammed with fixed data. They are very useful in embedded systems since a great deal of the code, and perhaps some data, does not change over time. Read-only memories are also less sensitive to radiation induced errors.

34. **Distinguish factory programmed ROM or mask and field programmable ROM.(Evaluating) (June 2016) (CO2)**

Factory programmed ROMs or mask are ordered from the factory with particular programming. ROMs can typically be ordered in lots of a few thousand, but clearly factory programming is useful only when the ROMs are to be installed in some quantity. They are sometimes called mask programmed ROM.

Field-programmable ROMs, on the other hand, can be programmed in the lab. Flash memory is the dominant form of field programmable ROM and is electrically erasable.

35. **What is flash memory? (Remembering) (CO2)**

Flash memory is the dominant form of field-programmable ROM and is electrically erasable. Flash memory uses standard system voltage for erasing and programming, allowing it to be reprogrammed inside a typical system. This allows applications such as automatic distribution of upgrades - the flash memory can be reprogrammed while downloading the new memory contents from a telephone line.

36. **What is boot - block flash? (Remembering) (CO2)**

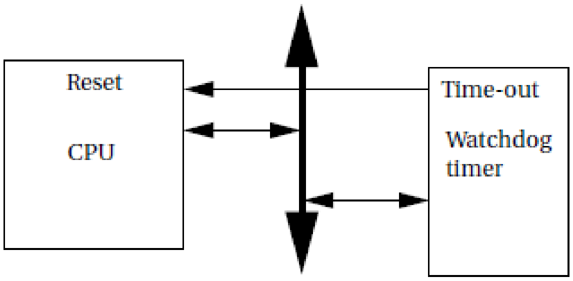
Early flash memories had to be erased in their entirety; modern devices allow memory to be erased in blocks. Most flash memories today allow certain blocks to be protected. A common application is to keep the boot-up code in a protected block but allow updates to other memory blocks on the device. As a result, this form of flash is commonly known as boot-block flash.

37. **Distinguish Timers and counters (Evaluating). (CO2)**

Timers and **counters** are distinguished from one another largely by their use, not their logic. Both are built from adder logic with registers to hold the current value, with an increment input that adds one to the current register value. However, a timer has its count connected to a periodic clock signal to measure time intervals, while a counter has its count input connected to an aperiodic signal in order to count the number of occurrences of some external event. Because the same logic can be used for either purpose, the device is often called a **counter/timer**.

38. **What is watchdog timer? (Remembering) (CO2)**

A ***watchdog timer*** is an I/O device that is used for internal operation of a system. As shown in Figure,the watchdog timer is connected into the CPU bus and also to the CPU's reset line. The CPU's software is designed to periodically reset the watchdog timer, before the timer ever reaches its time-out limit. If the watchdog timer ever does reach that limit, its time-out action is to reset the processor. In that case, the presumption is that either a software flaw or hardware problem has caused the CPU to misbehave. Rather than diagnose the problem, the system is reset to get it operational as quickly as possible.



39. **What is the use of A/D and D/Converters? (Remembering) (CO2)**

Analog/digital (A/D) and **digital/analog (D/A)** converters (typically known as **ADCs** and **DACs**, respectively) are often used to interface nondigital devices to embedded systems.

40. **What is the use of keyboard? (Remembering) (CO2)**

A keyboard is basically an array of switches, but it may include some internal logic to help simplify the interface to the microprocessor.

41. **What is the use of LED's? (Remembering) (CO2)**

Light-emitting diodes (LEDs) are often used as simple displays by themselves, and arrays of LEDs may form the basis of more complex displays.

42. **What is framed buffer? (Remembering) (CO2)**

A **frame buffer** is a RAM that is attached to the system bus. The microprocessor writes values into the frame buffer in whatever order is desired. The pixels in the frame buffer are generally written to the display in **raster order** by reading pixels sequentially.

43. **Distinguish passive matrix and active matrix. (Analyzing) (CO2)**

Early LCD panels were called **passive matrix** because they relied on a two-dimensional grid of wires to address the pixels. Modern LCD panels use an **active matrix** system that puts a transistor at each pixel to control access to the LCD. Active matrix displays provide higher contrast and a higher-quality display.

44. **What is touchscreen? (Remembering) (CO2)**

A **touchscreen** is an input device overlaid on an output device. The touchscreen registers the position of a touch to its surface. By overlaying this on a display, the user can react to information shown on the display.

45. **List the types of touchscreen. (Remembering) (CO2)**

The two most common types of touchscreens are resistive and capacitive.

46. **What are the types of component interfacing? (Remembering) (CO2)**

Memory Interfacing, Device Interfacing.

47. **How I/O devices are designed in device interfacing?(Remembering) (CO2)**

Some I/O devices are designed to interface directly to a particular bus, forming ***glueless interfaces***. But ***glue logic*** is required when a device is connected to a bus for which it is not designed.

48. **List out the elements which included in the architecture of an embedded computing system.(Remembering) (CO2)**

Hardware elements, Software elements.

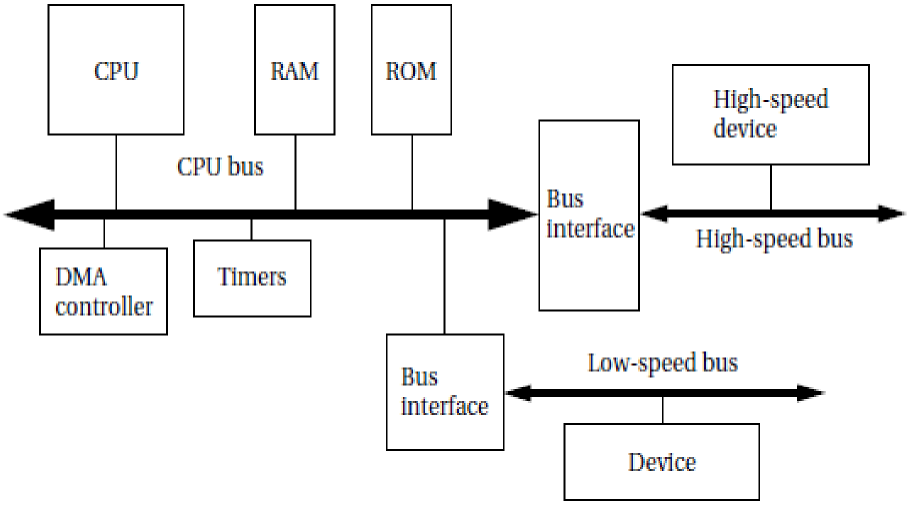
49. What are the elements included in hardware architecture?(Remembering) (CO2)

CPU,Bus,Memory,Input and Output devices

50. Define Evaluation boards.(Remembering) (CO2)

At the board level, the first step is to consider ***evaluation boards*** supplied by the microprocessor manufacturer or another company working in collaboration with the manufacturer. Evaluation boards are sold for many microprocessor systems; they typically include the CPU, some memory, a serial link for downloading programs, and some minimal number of I/O devices.

51. Show the Hardware architecture of a typical PC.(Remembering) (CO2)



52. Define Peripheral Component Interconnect(PCI) (Remembering) (CO2)

PCI (***Peripheral Component Interconnect***) is the dominant high-performance system bus . PCI uses high-speed data transmission techniques and efficient protocols to achieve high throughput. PCI uses wide buses with many data and address bits along with multiple control bits.

53. Define Target(Remembering) (CO2)

The hardware on which the code will finally run is known as the ***target***. The host and target are frequently connected by a USB link, but a higher-speed link such as Ethernet can also be used. The target must include a small amount of software to talk to the host system.

54. Explain about Cross compiler? (Understanding)(CO2)

A ***cross-compiler*** is a compiler that runs on one type of machine but generates code for another. After compilation, the executable code is downloaded to the embedded system by a serial link or perhaps burned in a PROM and plugged in and also often make use of host-target debuggers, in which the basic hooks for debugging are provided by the target and a more sophisticated user interface is created by the host.

55. What is meant by test bench program?(Remembering) (CO2)

A **test bench program** is built to help debug the embedded code. The **test bench** generates inputs to simulate the actions of the input devices; it may also take the output values and compare them against expected values.

56. Define Breakpoint(Remembering) (CO2)

Breakpoint is a debugging tool,which helps the user to specify an address at which the program's execution is to break. When the PC reaches that address, control is returned to the monitor program. From the monitor program, the user can examine and/or modify CPU registers, after which execution can be continued. Implementing breakpoints does not require using exceptions or external devices.

57. What is Logical analyzer?(Remembering) (CO2)

- The logic analyzer records the values on the signals into an internal memory and then displays the results on a display once the memory is full or the run is aborted. The logic analyzer can capture thousands or even millions of samples of data on all of these channels, providing a much larger time window into the operation of the machine than is possible with a conventional oscilloscope.

- A typical logic analyzer can acquire data in either of two modes that are typically called **state** and **timing modes**.

58. What are the debugging challengers?(Remembering) (CO2)

The debugging challengers includes,

- Logical errors in software can be hard to track down, but errors in real-time code can create problems that are even harder to diagnose.
- Real-time programs are required to finish their work within a certain amount of time; if they run too long, they can create very unexpected behavior.

59. List and explain the components used in embedded software.(Remembering) (CO2)

State machine, Circular buffer and queues.

Circular Buffer: The circular buffer is a data structure that lets us handle streaming data in an efficient way.

Queues: Queues are used whenever data may arrive and depart at somewhat unpredictable times or when variable amounts of data may arrive. A queue is often referred to as an **elastic buffer.**

60. List out the types of models of programs.(Remembering) (CO2)

- Data flow graphs
- Control/Data flow graphs

61. What are the nodes present in control/data flow graphs?(Remembering) (CO2)

- Decision nodes,
- Dataflow nodes.

62. What is object code?(Remembering) (CO2)

The process of translating the symbolic assembly language statements into bit level representations of instructions known as object code.

63. What does a linker do? (Remembering) (Nov/Dec 2012) (CO2)

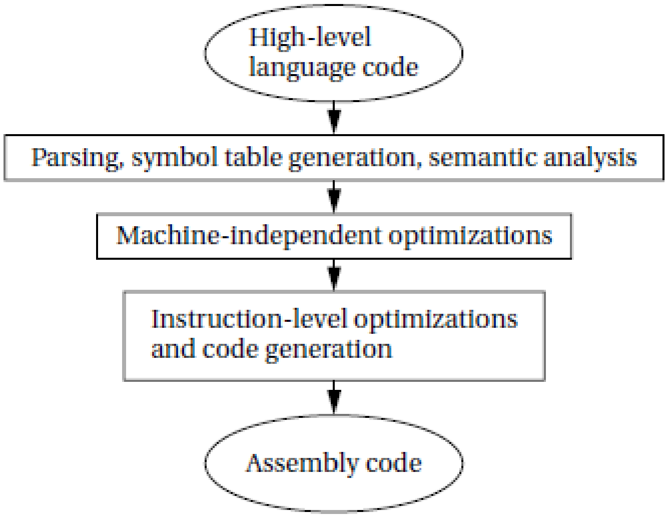
A program that combines multiple object program units, resolving references between them.

64. Distinguish Entry point and External reference?(Analyze) (CO2)

Entry point: The place in the file where a label is defined.

External reference: The place in the file where the label is used.

65. Construct the steps for compilation process. (Creating) (CO2)

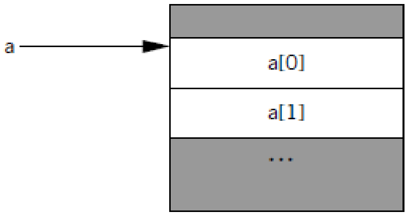

High-level language code
Parsing, symbol table generation, semantic analysis
Machine-independent optimizations
Instruction-level optimizations and code generation
Assembly code

66. Compare stack pointer (sp) and frame pointer (fp). (Analyzing)(CO2)

Stack pointer: Defines the end of the current frame.

Frame pointer: Defines the end of the last frame.

67. **Show the layout of one dimensional array in memory.(Understanding) (CO2)**



68. **Mention the different types of loop transformations. (CO2)**

Loop fusion, Loop distribution, Loop tiling.

69. **How loop fusion is differ from the loop distribution? (Remembering) (CO2)**

Loop fusion combines two or more loops into a single loop whereas the loop distribution, decomposing a single loop into multiple loops.

70. **State the conditions must be satisfied for transformation in loop fusion. (CO2)**

1. Loops must iterate over the same values.
2. The loop bodies must not have dependencies that would be violated if they are executed together.

71. **Write short notes on Loop tiling and Array padding. (CO2)**

Loop tiling: It breaks up a loop into a set of nested loops, with each inner loop performing the operations on a subset of the data.

Array padding*:* It adds dummy data elements to a loop in order to change the layout of the array in the cache.

72. **Write about two types of testing strategies. (CO2)**

Black-box methods generate tests without looking at the internal structure of the program.

Clear-box (also known as **white-box**) methods generate tests based on the program structure.

73. **What is meant by branch testing?(Remembering) (CO2)**

Branch testing is a simple testing strategy, which requires the true and false branches of a conditional and every simple condition in the conditional's expression to be tested at least once.

74. **What are all the tests available in black-box-testing?(Remembering) (CO2)**

Random tests, Regression tests.

75. **State the function of an assembler and linker. (Remembering) (Dec 2015) (CO2)**

When translating assembly code into object code, the assembler, must translate opcodes and format the bits in each instruction and translate labels into addresses.

A linker allows a program to be stitched together out of several smaller pieces. The linker operates on the object files created by the assembler and modifies the assembled code to make the necessary links between files.

BIG QUESTIONS

1. Explain in detail about various bus protocols in CPU processor. **(Evaluating) (CO2)**
2. Discuss the system bus configuration and explain the bus protocol. **(Creating)(AU-May/June 2013) (CO2)**
3. Explain briefly about various types of memory devices interface with a processor. **(Understanding) (AU-April 2014, Nov/Dec 2014) (CO2)**
4. How I/O devices are interface with a processor? **(Remembering) (AU-Nov/Dec 2012) (Dec 2015)(CO2)**
5. Explain the concepts of component interfacing. **(Evaluating) (AU-April 2014) (CO2)**
6. Explain the design of microprocessor based on the hardware architecture of a PC. **(Evaluating) (Dec 2015) (CO2)**
7. With a suitable example explain how debugging is carried out using debugger and compilers? **(Understanding) (AU-Nov/Dec 2012, April 2014) (CO2)**
8. Explain in detail about components for embedded programs. **(Understanding) (CO2)**
9. Discuss in detail the fundamental model used for program development. **(Creating)(AU-May/June 2013) (CO2)**

10. Discuss in detail about assembly, loading and linking with examples. **(Creating) (AU-Nov/Dec 2013,2014) (CO2)**
11. Explain in detail about basic Compilation Techniques. **(Evaluating) (CO2) OR**
    Draw the basic compilation process and illustrate the functional mechanism of compilation task. **(Creating) (AU-April 2014) (Dec 2015) (CO2)**
12. Analyze the optimization of programs and explain how to validate and test the programs? **(Analyzing) (AU- Nov/Dec 2012,May/June 2013,Nov/Dec 2014) (Dec 2015) (CO2)**
13. Draw the three structures commonly used in embedded software with programming and elaborate with an example. **(Creating) (June 2016) (CO2)**
14. What are the various debug techniques and challenges in embedded system design? Illustrate in detail. **(Understanding) (AU-May/June 2013) (June 2016) (CO2)**
15. Explain about how assembler helps in the development of program design. **(Understanding) (AU-Nov/Dec 2012) (CO2)**
16.

UNIT - III

PROCESSES AND OPERATING SYSTEMS

1. **What are the two fundamental abstractions that allow us to build complex applications on microprocessors? (Remembering) (CO3)**

   The process and the operating system (OS)

2. **What is RTOS? (Remembering) (CO3)**

   Real Time Operating Systems (RTOSs), which are OS's that provide facilities for satisfying real-time requirements

3. **Compare RTOS and General purpose OS. (Understanding) (April/May 2015) (CO3)**

   A RTOS allocates resources using algorithms that take real time into account. General-purpose OS's, in contrast, generally allocate resources using other criteria like fairness. Trying to allocate the CPU equally to all processes without regard to time can easily cause processes to miss their deadlines.

4. **Explain multi tasks. (Understanding) (Nov/Dec 2014) (CO3)**

   Most embedded systems require functionality and timing that is too complex to embody in a single program. We break the system into multiple tasks in order to manage when things happen

5. **Motivate Tasks. (Analyzing) (CO3)**
   - Many (if not most) embedded computing systems do more than one thing that is, the environment can cause mode changes that in turn cause the embedded system to behave quite differently.
   - For example, when designing a telephone answering machine, we can define recording a phone call and operating the user's control panel as distinct tasks, because they perform logically distinct operations and they must be performed at very different rates.
   - These different tasks are part of the system's functionality, but that application-level organization of functionality is often reflected in the structure of the program as well.

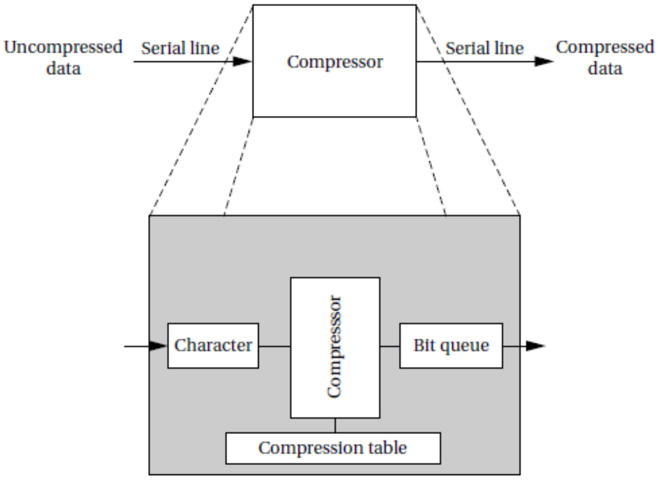
6. **Recall Process. (Remembering) (Nov/Dec 2013) (Dec 2015) (CO3)**

   A process is a single execution of a program. If we run the same program two different times, we have created two different processes. Each process has its own state that includes not only its registers but all of its memory. In some OSs, the memory management unit is used to keep each process in a separate address space. In others, particularly lightweight RTOSs, the processes run in the same address space.

7. **What are threads? (Remembering) (Dec 2015) (June 2016) (CO3)**

   Processes that share the same address space are often called threads.

8. Build an on-the-fly compression box. (Creating) (CO3)



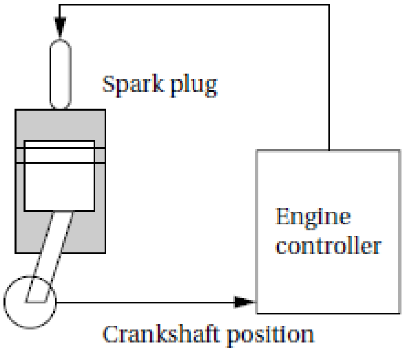
9. What are Multirate systems? (Remembering) (CO3)

Implementing code that satisfies timing requirements is even more complex when multiple rates of computation must be handled. Multirate embedded computing systems are very common, including automobile engines, printers, and cell phones. In all these systems, certain operations must be executed periodically, and each operation is executed at its own rate.

10. Analyze the task of automotive engine control. (Analyzing) (CO3)

The simplest automotive engine controllers, such as the ignition controller for a basic motorcycle engine, perform only one task - timing the firing of the spark plug, which takes the place of a mechanical distributor.

11. Develop automotive engine control system. (Applying) (CO3)



12. Infer timing requirements on processes. (Understanding) (CO3)

- Processes can have several different types of timing requirements imposed on them by the application.
- The timing requirements on a set of processes strongly influence the type of scheduling that is appropriate.
- A scheduling policy must define the timing requirements that it uses to determine whether a schedule is valid.
- Before studying scheduling proper, we outline the types of process timing requirements that are useful in embedded system design.

13. What is release time? (Remembering) (CO3)

- The release time is the time at which the process becomes ready to execute; this is not necessarily the time at which it actually takes control of the CPU and starts to run.
- The release time is generally measured from that event, although the system may want to make the process ready at some interval after the event itself.
- In simpler systems, the process may become ready at the beginning of the period.

- More sophisticated systems, such as those with data dependencies between processes, may set the release time at the arrival time of certain data, at a time after the start of the period.

14. **What is deadline? (Remembering) (CO3)**

A deadline specifies when a computation must be finished. The deadline for an aperiodic process is generally measured from the release time, since that is the only reasonable time reference. The deadline for a periodic process may in general occur at some time other than the end of the period.

15. **Define task graph. (Remembering) (CO3)**

A set of processes with data dependencies is known as a **task graph**.

16. **What is directed acyclic graph? (Remembering) (CO3)**

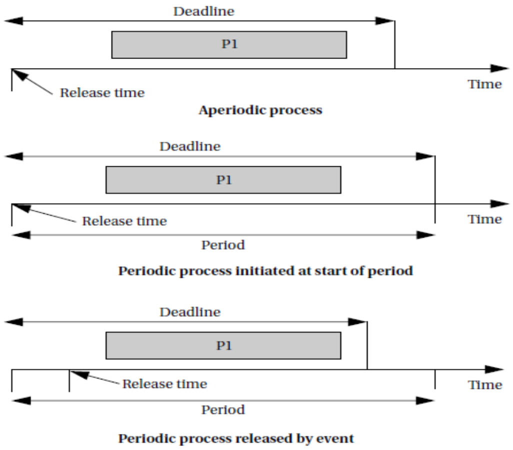
The data dependencies must form a directed acyclic graph (DAG) - a cycle in the data dependencies is difficult to interpret in a periodically executed system.

17. **Compare initiation time and completion time. (Evaluating) (CO3)**

The **initiation time** is the time at which a process actually starts executing on the CPU.

The **completion time** is the time at which the process finishes its work.

18. **Outline release time and deadline. (Understanding) (CO3)**



Aperiodic process

Periodic process initiated at start of period

Periodic process released by event

19. **Examine CPU time of process. (Analyzing) (CO3)**

The most basic measure of work is the amount of CPU time expended by a process. The CPU time of process ‘i’ is called $C_{i.}$ Note that the CPU time is not equal to the completion time minus initiation time; several other processes may interrupt execution. The total CPU time consumed by a set of processes is

$$T = \sum_{1 \le i \le n} T_i.$$

20. **What is meant by utilization? (Remembering) (CO3)**

Utilization is the ratio of the CPU time that is being used for useful computations to the total available CPU time.

$$U = \frac{\text{CPU time for useful work}}{\text{total available CPU time}}. \qquad U = \frac{T}{t}.$$

21. **Recall scheduling. (Remembering) (CO3)**

The first job of the OS is to determine that process runs next. The work of choosing the order of running processes is known as scheduling.

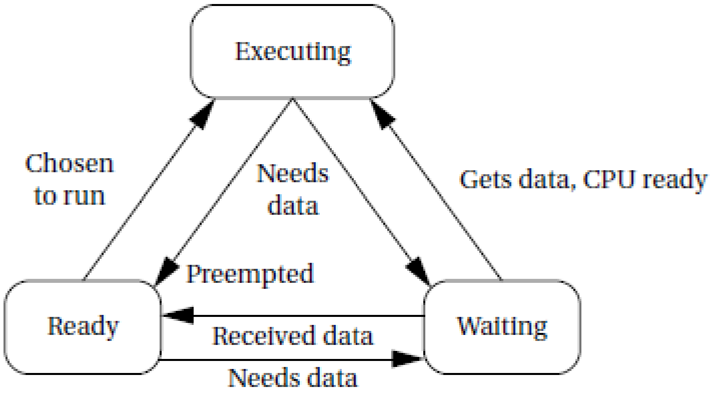
22. **List the three basic scheduling states. (Analyzing) (CO3)**

The OS considers a process to be in one of three basic scheduling states are **waiting**, **ready**, and **executing**.

23. **Explain scheduling policy. (Evaluating) (May/June 2013) (CO3)**

- A **scheduling policy** defines how processes are selected for promotion from the ready state to the running state.
- Every multitasking OS implements some type of scheduling policy. Choosing the right scheduling policy not only ensures that the system will meet all its timing requirements, but it also has a profound influence on the CPU horsepower required to implement the system’s functionality.

24. Construct Scheduling states of a process. (Creating) (CO3)



25. What is meant by Schedulability? (Remembering) (CO3)

- Schedulability means whether there exists a schedule of execution for the processes in a system that satisfies all their timing requirements.
- In general, we must construct a schedule to show schedulability, but in some cases we can eliminate some sets of processes as unschedulable using some very simple tests.
- Utilization is one of the key metrics in evaluating a scheduling policy.

26. What is hyperperiod? (Remembering) (CO3)

For periodic processes, the length of time that must be considered is the hyperperiod, which is the least-common multiple of the periods of all the processes.

27. Define unrolled schedule. (Remembering) (CO3)

The complete schedule for the least-common multiple of the periods is sometimes called the unrolled schedule.

28. What are preemptive real-time operating systems? (Remembering) (CO3)

A RTOS executes processes based upon timing constraints provided by the system designer. The most reliable way to meet timing constraints accurately is to build a preemptive OS and to use priorities to control what process runs at any given time.

29. What is Preemption? (Remembering) (CO3)

Preemption is an alternative to the C function call as a way to control execution. To be able to take full advantage of the timer, we must change our notion of a process as something more than a function call.

30. Recall kernel. (Remembering) (CO3)

- The kernel is the part of the OS that determines what process is running.
- The kernel is activated periodically by the timer.
- The kernel determines what process will run next and causes that process to run.
- On the next timer interrupt, the kernel may pick the same process or another process to run.

31. What is time quantum? (Remembering) (CO3)

The kernel is activated periodically by the timer. The length of the timer period is known as the time quantum because it is the smallest increment in which we can control CPU activity.

32. Compare context and context switching. (Evaluating) (CO3)

The set of registers that define a process are known as its **context** and switching from one process's register set to another is known as **context switching**.

33. What is process control block? (Remembering) (CO3)

The data structure that holds the state of the process is known as the process control block.

34. How does the kernel determine what process will run next? (Remembering) (CO3)

We want a mechanism that executes quickly so that we don't spend all our time in the kernel and starve out the processes that do the useful work. If we assign each task a numerical priority, then the kernel can simply look at the processes and their priorities, see which ones actually want to execute (some may be wailing for data or for some event),and select the highest priority process that is ready to run. This mechanism is both flexible and fast. The priority is a non-negative integer value. The exact value of the priority is not as important as the relative priority of different processes.

35. Distinguish active objects and active class. (Analyzing) (CO3)

UML often refers to processes as **active objects**, that is, objects that have independent threads of control. The class that defines an active object is known as an **active class**.

36. Elaborate two major ways to assign priorities. (Creating) (CO3)

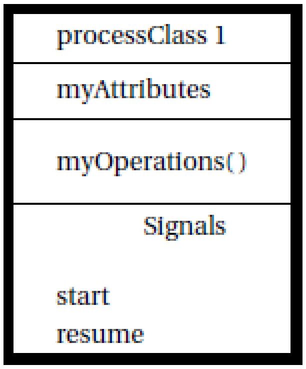
Static priorities that do not change during execution.

Dynamic priorities that do change during execution.

37. What is Rate Monotonic Scheduling (RMS)? (Remembering) (CO3)

Rate Monotonic Scheduling (RMS), introduced by Liu and Layland, was one of the first scheduling policies developed for real-time systems and is still very widely used. RMS is a static scheduling policy. It turns out that these fixed priorities are sufficient to efficiently schedule the processes in many situations. The theory underlying RMS is known as **Rate Monotonic Analysis (RMA).**

38. Show the UML active class structure. (Understanding) (CO3)



39. Summarize Rate Monotonic Analysis (RMA) theory. (Understanding) (CO3)

- All processes run periodically on a single CPU.
- Context switching time is ignored.
- There are no data dependencies between processes.
- The execution time for a process is constant.
- All deadlines are at the ends of their periods.
- The highest-priority ready process is always selected for execution.

40. What is response time and critical instant? (Remembering) (CO3)

We define the **response time** of a process as the time at which the process finishes. The **critical** instant for a process is defined as the instant during execution at which the task has the largest response time.

41. Interpret Earliest Deadline First (EDF) scheduling. (Understanding) (CO3)

Earliest Deadline First (EDF) is another well known scheduling policy. It is a dynamic priority scheme - it changes process priorities during execution based on initiation times. As a result, it can achieve higher CPU utilizations than RMS.

42. Explain Earliest Deadline First (EDF). (Understanding) (CO3)

The EDF policy is also very simple: It assigns priorities in order of deadline. The highest-priority process is the one whose deadline is nearest in time, and the lowest priority process is the one whose deadline is farthest away. Clearly, priorities must be recalculated at every completion of a process. However, the final step of the OS during the scheduling procedure is the same as for RMS - the highest priority ready process is chosen for execution.

43. Compare RMS and EDF. (Evaluating) (CO3)

EDF can extract higher utilization out of the CPU, but it may be difficult to diagnose the possibility of an imminent overload. Because the scheduler does take some overhead to make scheduling decisions, a factor that is ignored in the schedulability analysis of both EDF and RMS, running a scheduler at very high utilizations is somewhat problematic.

RMS achieves lower CPU utilization but is easier to ensure that all deadlines will be satisfied. In some applications, it may be acceptable for some processes to occasionally miss deadlines.

44. What if your set of processes is unschedulable and you need to guarantee that they complete their deadlines? (Remembering) (CO3)

There are several possible ways to solve this problem:

- **Get a faster CPU**. That will reduce execution times without changing the periods, giving you lower utilization. This will require you to redesign the hardware, but this is often feasible because you are rarely using the fastest CPU available.
- **Redesign the processes to take less execution time**. This requires knowledge of the code and may or may not be possible.
- **Rewrite the specification to change the deadlines**. This is unlikely to be feasible, but may be in a few cases where some of the deadlines were initially made tighter than necessary.

45. Explain priority inversion. (Understanding) (CO3)

In all cases we assume that each process is totally self-contained. However, that is not always the case for instance, a process may need a system resource, such as an I/O device or the bus, to complete its work. Scheduling the processes without considering the resources those processes require can cause **priority inversion**, in which a low-priority process blocks execution of a higher priority process by keeping hold of its resource.

46. How process can send a communication? (Remembering) (CO3)

A process can send a communication in one of two ways: **blocking** or **nonblocking**.

After sending a blocking communication, the process goes into the waiting state until it receives a response. Nonblocking communication allows the process to continue execution after sending the communication. Both types of communication are useful.

47. List the different styles in inter process communication. (Analyzing) (April 2014) (June 2016) (CO3)

Shared memory-It is a bus based system and it consists of CPU and I/O device.

Message passing-It complements the shared memory model and it communicating entry has its own message send/receive unit.

48. What are the assumptions made in evaluating operating system performance? (Remembering) (CO3)

- We have assumed that context switches require zero time.
- We have assumed that we know the execution time of the processes.
- We probably determined worst-case or best-case times for the processes in isolation.

49. How system's power consumption is done in RTOS? (Remembering) (CO3)

The RTOS and system architecture can use static and dynamic power management mechanisms to help manage the system's power consumption.

50. Determine when to switch into and out of a power-up mode. (Evaluating) (CO3)

Determining when to switch into and out of a power-up mode requires an analysis of the overall system activity.

- Avoiding a power-down mode can cost unnecessary power.
- Powering down too soon can cause severe performance penalties.

51. What is the goal of predictive shutdown? (Remembering) (CO3)

The goal is to predict when the next request will be made and to start the system just before that time, saving the requestor the start-up time. In general, predictive shutdown techniques are probabilistic - they make guesses about activity patterns based on a probabilistic model of expected behavior. Because they rely on statistics, they may not always correctly guess the time of the next activity.

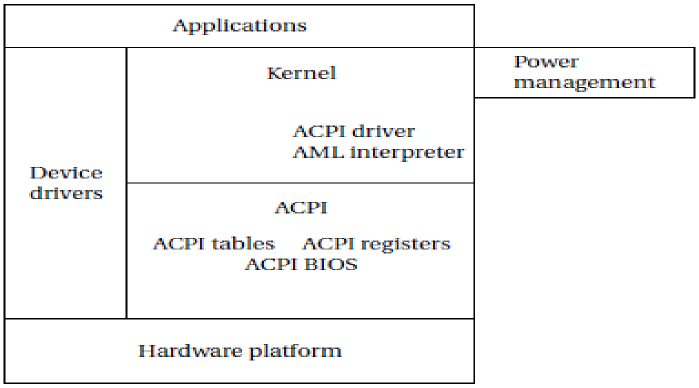
52. List the problems that are caused by predictive shutdown.(Analyzing) (CO3)

- The requestor may have to wait for an activity period. In the worst case, the requestor may not make a deadline due to the delay incurred by system start-up.
- The system may restart itself when no activity is imminent. As a result, the system will waste power.

53. What is Advanced Configuration and Power Interface (ACPI)? (Remembering) (CO3)

Advanced Configuration and Power Interface (ACPI) is an open industry standard for power management services. It is designed to be compatible with a wide variety of OSs. It was targeted initially to PCs.

54. Illustrated role of ACPI in the system. (Understanding) (CO3)



55. List the basic global power states supported by ACPI. (Analyzing) (CO3)

1. G3, the mechanical off state, in which the system consumes no power.
2. G2, the soft off state, which requires a full OS reboot to restore the machine to working condition. This state has four substates:
   - S1, a low wake-up latency state with no loss of system context;
   - S2, a low wake-up latency state with a loss of CPU and system cache state;
   - S3, a low wake-up latency state in which all system state except for main memory is lost; and
   - S4, the lowest-power sleeping state, in which all devices are turned off.
3. G1, the sleeping state, in which the system appears to be off and the time required to return to working condition is inversely proportional to power consumption.
4. G0, the working state, in which the system is fully usable.
5. The legacy state, in which the system does not comply with ACPI.

56. Define Semaphore. (Remembering) (CO3)

Semaphore is a special variable or function that is used to take note of certain actions to prevent another task or process from proceeding.

57. What is context switching? (Remembering) (Nov/Dec 2012) (Nov/Dec 2013) (April 2014) (April/May 2015) (CO3)

Saving the contents of the CPU registers and loading the new task parameters is called context switching.

58. How is Real Time Operating System uniquely different than a general purpose OS? (Remembering) (April/May 2015) (Nov/Dec 2014) (CO3)

A **Real Time Operating System (RTOS)** is a multitasking operating system intended for real-time applications. An RTOS will typically use specialized scheduling algorithms in order to provide the real-time developer with the tools necessary to produce deterministic behavior in the final system. Key factors in an RTOS are therefore minimal interrupt latency and a minimal thread switching latency.

An **Operating System (OS)** is the software component of a computer system that is responsible for the management and coordination of activities and the sharing of the limited resources of the computer. The operating system acts as a host for applications that are run on the machine

59. What does a scheduler do in an operating system environment? (Remembering) (Nov/Dec 2012) (CO3)

A process can go into the executing state only when it has all its data, is ready to run, and the scheduler selects the process as the next process to run.

60. What are the three states of task or process? (Remembering) (CO3)

(i) Running, (ii) Ready to run, (iii) Waiting

61. State the importance of power management and optimization. (Analyzing) (Dec 2015) (CO2)

A power management is a strategy for determining when to perform certain power management operations. It examines the state of the system to determine when to take actions. The overall strategy is based on the characteristics of the static and dynamic power management mechanisms.

BIG QUESTIONS

1. Discuss in detail about multi tasks and multi processes. **(Creating) (AU-Nov/Dec 2013) (CO3)**
2. Explain the services of the operating system in handling multi process scheduling and communication. **(Understanding) (AU-April 2014) (CO3) OR**

   Explain any two scheduling policies used in multi process environment. **(Understanding) (AU-Nov/Dec 2012,2014) (CO3)**
3. State the function of a scheduler. Also illustrate the basic mechanism of preemptive scheduling. **(Understanding) (Dec 2015) (CO3)**
4. Explain the principle of priority based context switching mechanism. Discuss about various priority based scheduling algorithms with an example. **(Understanding) (AU-May/June 2013) (Dec 2015) (June 2016) (CO3)**
5. Discuss about multi process and inter process communication mechanism. **(Creating) (AU-Nov/Dec 2012, Nov/Dec 2013) (CO3) OR**

   Explain in detail how shared memory and message passing mechanisms are used for inter process communication. **(Understanding) (AU-May/June 2013) (CO3) OR**

   Explain in detail about inter process communication mechanism provided by OS to transfer data. **(Understanding) (AU-April/May 2015) (Dec 2015) (CO3)**
6. How to assess the operating system performance? Explain **(Remembering) (AU-Nov/Dec 2014) (June 2016) (CO3) OR**

   How is the performance of the operating system evaluated? **(Remembering) (Dec 2015) (CO3)**
7. Elaborate the power management optimization in embedded system design with example. **(Creating) (AU-April 2014) (June 2016)(CO3)**
8. Discuss in detail about power optimization strategies for CPU operation. **(Creating) (AU-April/May 2015) (CO3)**
9. Outline short notes on Co-operative scheduling. **(Understanding) (AU-April/May 2015) (CO3)**

UNIT-IV

HARDWARE ACCELARATORS AND NETWORKS

1. What is mean by accelerators/hardware accelerator and give one example? (Remembering) (Nov/Dec 2013) (CO4)

An accelerator is one important category of processing element for embedded Multiprocessor. Accelerators can provide large performance increases for applications with computational kernels that spend a great deal of time in a small section of code. Accelerators can also provide critical speedups for low-latency I/O functions.

Example: Hardware/software co-design.

2. Define host. (Remembering) (CO4)

A CPU accelerator is attached to the CPU bus; the CPU is often called the HOST. The CPU talks to the accelerator through data and control registers in the accelerator. These registers allow the CPU to monitor the accelerator’s operation and to give the accelerator commands.

3. Analyze the important needs of ASIC in hardware accelerators. (Analyzing) (CO4)

(i). we could, in fact, find the standard component in a catalog that does what we want, though we may have to design interface logic to mate it to our CPU.

(ii). we could design a custom integrated circuit. This requires considerable design and manufacturing time, but may be worth the effort for high- volume designs.

(iii). we could use a programmable logic technology such as a Field Programmable Gate Array (FPGA).

4. Define FPGAs. (Remembering) (CO4)

FPGS is a programmable logic chip that can quickly be customized to a particular function. FPGAs come in two basic varieties. One time programmable devices can be programmed once, while reprogrammable devices can be programmed many times with new logic designs.

5. List the main role of accelerators for designing embedded software. (Analyzing) (CO4)

(i).For building distributed embedded systems is that they offer significantly better cost/performance.

(ii).The processing element purchase price is a nonlinear function of performance increases.

6. Classify the several types of applications used in the accelerators. (Understanding) (CO4)

The several types of applications that are well suited to accelerators are,

(i).Some functions require operations that do not map onto a CPU's data operations. The mismatch may be due to several reasons.

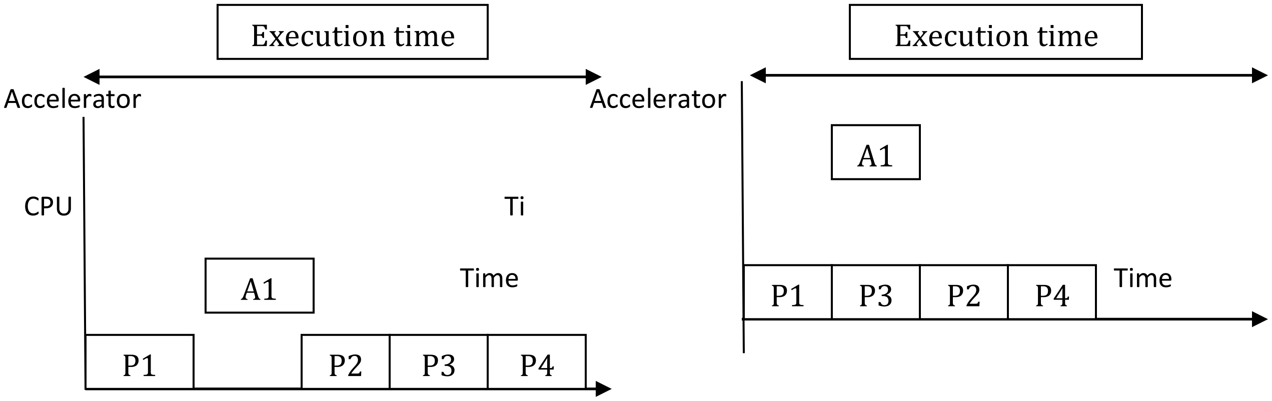
(ii).Highly responsive input and output operations may be best performed by an accelerator with an attached I/O unit. If data must be read, processed, and written to meet a very tight deadline.

(iii). Applications with streaming data, such as wireless and multimedia, provide special challenges to CPUs. Streaming data are not well suited to CPU caches since each element of the stream has a limited lifetime. An accelerator with special purpose data fetch logic may be an efficient way to perform operations on the data stream.

7. What is mean by single and multi threaded system? (Remembering) (CO4)

Single threaded: Whether the CPU sits idle while the accelerator runs in the single-threaded case.

Multi threaded: CPU can do useful work in parallel with a accelerator.



SINGLE THREADED **MULTI THREADED**

8. What are the factors that influence a t_{in} and t_{out} in bus transaction? (Remembering) (CO4)

The values for t_{in} and t_{out} must reflect the time required for the bus transactions, including the following factors are,

(i).The time required to flush any register or cache values to main memory. If those values are needed in main memory to communicate with the accelerator.

(ii).The time required for transfer of control between the CPU and accelerator.

9. Classify the components or units present in the accelerator.(Analyzing) (CO4)

Register file: The accelerator acts as a buffer between main memory and the accelerator core.

Read unit: It can read ahead of the accelerators requirements and load the registers with the next required data.

Write unit: It can send recently completed values to main memory while the core works with other values.

10. What is meant by caching problem in accelerator? (Remembering) (CO4)

In order to avoid tying up the CPU, the data transfers can be performed in DMA mode, which means that the accelerator must have the requires logic to become a bus master and perform DMA operations.

11. List out the steps or sequence of operations involved during the CPU cache problem. (Analyzing) (CO4)

The sequence of operations to avoid the CPU cache problem is,

(i). The CPU reads location S

(ii). The accelerator writes S

(iii). The CPU again reads S

12. How will avoid cache problem in accelerator? (Remembering) (CO4)

If the CPU has cached location S, the program will not see the values of S written by the accelerator. It will instead get the old value of S stored in the cache. To avoid this problem, the CPU's cache must be updated to reflect the fact that this cache entry is invalid. Your CPU may provide cache invalidation instructions; you can also remove the location from the cache by reading another location that is mapped to the same cache line.

13. Define Synchronization process. (Remembering) (CO4)

If the CPU and accelerator operate concurrently and communicate via shared memory, it is possible that similar problems will occur in main memory, not just in the cache. If one PE reads a value and then updates it, the other PE may change the value, causing the first PE's update to be invalid. In some cases, it may be possible to use a very simple synchronization scheme for communication

Synchronization process: The CPU writes data into a memory buffer, starts the accelerator, and waits for the accelerator to finish, and then reads the shared memory area.

14. What is meant by Scheduling and allocation? (Remembering) (CO4)

Scheduling: The scheduling of operations on the PEs and the communications between the PEs are linked. If one PE finishes its computations too late, it may interfere with another communication on the network as it tries to send its result to the PE that needs it.

Allocation: The allocation of computations to the PEs determines what communications are required-if a value computed on one PE is needed on another PE, it must be transmitted over the network.

15. What is distributed embedded system? (Remembering) (April 2014) (June 2016) (CO4)

In distributed embedded system, several processing elements (either microprocessor or ASICs) are connected by a network that allows them to communicate. The application is distributed over the processing elements, and some of the work is done at each node in the network.

More than one computer or group of computer and PEs are connected via network that forms distributed embedded systems.

16. Why embedded system is distributed in various applications? (Remembering) (CO4)

- Higher performance at lower cost.
- Physically distributed activities---time constants may not allow transmission to central site.
- Improved debugging a prototype or diagnose a problem in the field--use one CPU in network to debug others.
- May buy subsystems that have embedded processors.

17. What is meant by network abstraction? (Remembering) (CO4)

The International Standards Organization (ISO) has developed a seven layer model for networks known as Open Systems Interconnection (OSI) models or network abstractions. It provides a standard way to classify network components and operations.

18. Mention the reason for building distributed embedded systems. (Remembering) (Dec 2015) (CO4)

The first is that they offer significantly better cost/performance. As the faster PEs are very costly, splitting the application so that it can be performed on several smaller PEs in much cheaper.

Multiple PEs can also help with real time performance we can often meet deadlines and be responsible to interaction much more easily when we put time critical processes on separate PEs.

19. Define point-to-point communication link. (Remembering) (CO4)

A point-to-point link establishes a connection between exactly two processing elements. Point-to-point links are simple to design precisely because they deal with only two components.

20. List the OSI layers from lowest to highest level of abstraction. (Analyzing) (April 2014) (CO4)

Application	End-use interface
Presentation	Data format
Session	Application dialog control
Transport	Connections
Network	End-to-end service
Data link	Reliable data transport
Physical	Mechanical,electrical

Physical: defines the basic properties of the interface between systems, including connectors, bit formats, etc.

Data link: error detection and control across a single link (single hop).

Network: end-to-end multi-hop data communication.

Transport: provides connections; may optimize network resources.

Session: services for end-user applications: data grouping check pointing, etc.

Presentation: data formats, transformation services.

Application: interface between network and end-user programs.

21. Define half duplex and full duplex communications. (Remembering) (CO4)

Half duplex communication: It allows for only one-way communication.

Full duplex communication: Point-to-point connection that can be used for simultaneous communication in both directions between the two PEs.

22. What is meant by bus? (Remembering) (CO4)

Buses: The exchange of information.

Information is transferred between units of the microcomputer by collections of conductors called buses.

There will be one conductor for each bit of information to be passed, e.g., 16 lines for a 16 bit address bus. There will be address, control, and data buses.

23. What are the different types of bus arbitration in distributed network? (Remembering) (CO4)

There are two types of bus arbitration scheme used in distributed networks are given by,

(i). Fixed-priority arbitration: It gives priority to competing devices in the same way. If a high priority and a low priority device both have long transmissions ready at the same time, it is quite possible that the low priority device will not be able to transmit anything until the high priority device has sent all its data packets.

(ii). Fair arbitration: These schemes make sure that no device is starved. **Round robin arbitration** is the most commonly used of the fair arbitration schemes. The PCI bus requires that the arbitration scheme used on the bus must be fair, although it does not specify a particular arbitration scheme. Most implementations of PCI use round robin arbitration.

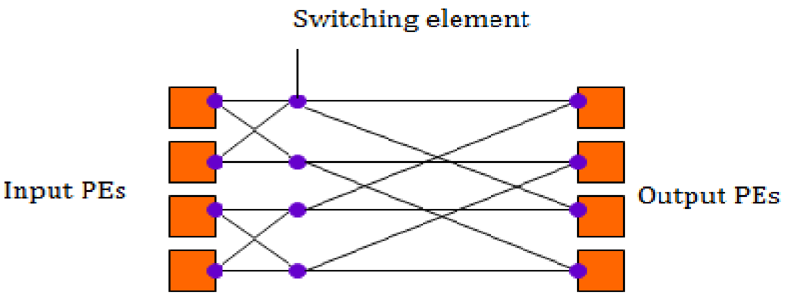
24. Define crossbar and cross point network. (Remembering) (CO4)

Crossbar: Crossbar not only allows any input to be connected to any output, it also allows all combinations of input/output connections to be made.

Cross point: It is a switch that connects an input to an output.

25. Explain multistage networks. (Understanding) (CO4)

Many other networks have been designed that provide varying amounts of parallel communication at varying hardware costs.



26. What are different types of buses in networks for embedded systems? (Remembering) (CO4)

There are four types of buses in networks for embedded systems. They are,

(i). **Multibus and VME:** It is developed by Intel and Motorola, respectively, for multicard computer systems and has been widely used in industrial applications.

(ii). **ISA Bus:** It has been used to support many I/O cards for PC- based embedded systems.

(iii). **PCI Bus:** It has replaced ISA for high speed interfaces in PC based applications.

27. List the interconnect networks has been developed for distributed embedded computing. (Analyzing) (CO4)

(i). **I^{2}C Bus:** It is used in microcontroller based systems.

ii). **Controller Area Network (CAN) Bus:** It was developed for automotive electronics. It provides megabit rates and can handle large numbers of devices.

(iii). **Echelon LON:** It was developed for home and industrial automation.

(iv). Many DSPs supply their own interconnect structures for multiprocessing.

28. What is I^{2}C Bus? (Remembering) (CO4)

The I^{2}C Bus is a well known bus commonly used to link microcontrollers into systems. It has even been used for the command interface in an MPEG-2 video chip; while a separate bus was used for high speed video data, setup information was transmitted to the on-chip controller through an I^{2}C Bus interface**.**

29. List out the characteristics and advantages of I^{2}C Bus. (Analyzing) (CO4)

Characteristics

- It is serial in nature
- Multiple master, fixed priority arbitration

Advantages:

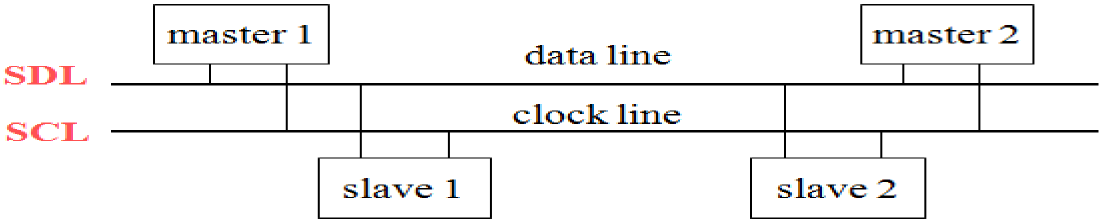
- Designed to be low cost
- Easy to implement
- Moderate speed (up to 100 kilobits per second for the standard bus and up to 400jbits/sec for the extended bus)

30. Explain two lines used in I^{2}C bus physical layer. (Understanding) (CO4)

There are two lines are used in I^{2}C bus

(i). Serial Data Line (SDL): For data

(ii). Serial Clock Line (SCL): which indicates when valid data are on the data line.



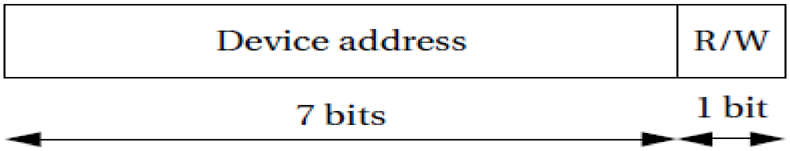
31. Explain Bus transaction in I^{2}C bus? (Understanding) (CO4)

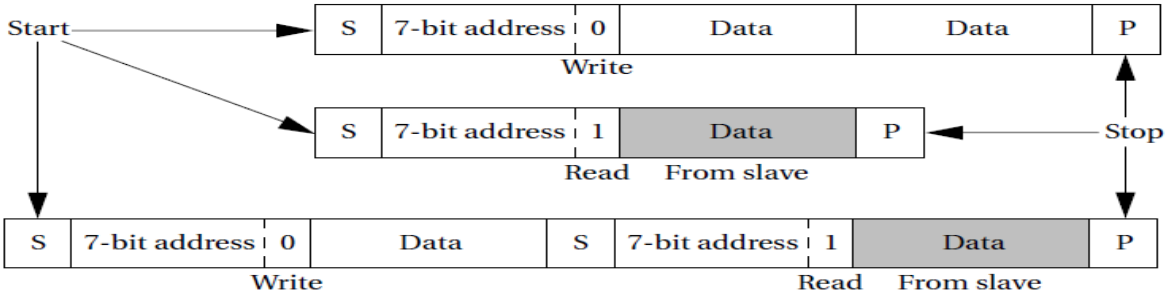
A bus transaction comprised a series of 1-byte transmissions and an address followed by one or more data bytes. I2C encourages a data-push programming style. When a master wants to write a slave, it transmits the slave’s address followed by the data.

A bus transaction is initiated by a start signal and completed with an end signal as follows:

- A start is signaled by leaving the SCL high and sending a 1 to 0 transition on SDL.
- A stop is signaled by setting the SCL high and sending a 0 to 1 transition on SDL.

32. Illustrate the Format of an I^{2}C address transmission and bus transaction. (Understanding) (CO4)





In figure, the master writes 2 bytes to the addressed slave. In the second, the master requests a read from a slave. In the third, the master writes 1 byte to the slave, and then sends another start to initiate a read from the slave

33. What is meant by Ethernet? (Remembering) (CO4)

Ethernet is very widely used as a local area network for general-purpose computing. Because of its ubiquity and the low cost of Ethernet interfaces, it has seen significant use as a network for embedded computing. Ethernet is particularly useful when PCs are used as platforms, making it possible to use standard components, and when the network does not have to meet rigorous real-time requirements.

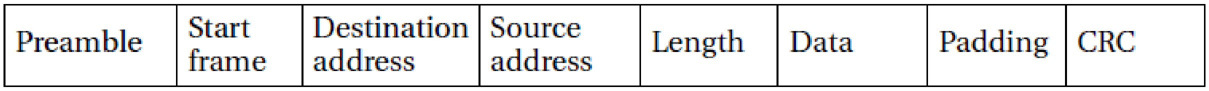
34. Explain CDMA Techniques and exponential backoff used in Ethernet. (Understanding) (CO4)

The Ethernet arbitration scheme is known as *Carrier Sense Multiple Access* with *Collision Detection (CSMA/CD)*.

In this technique, a node that has a message waits for the bus to become silent and then starts transmitting. It simultaneously listens, and if it hears another transmission that interferes with its transmission, it stops transmitting and waits to retransmit. The waiting time is random, but weighted by an exponential function of the number of times the message has been aborted.

A message may be interfered with several times before it is successfully transmitted; the ***exponential backoff*** technique helps to ensure that the network does not become overloaded at high demand factors.

35. Build the basic format for Ethernet packets. (Applying) (CO4)



Preamble	Start frame	Destination address	Source address	Length	Data	Padding	CRC

36. Define field bus. (Remembering) (CO4)

Fieldbus is a set of standards for industrial control and instrumentation systems. The H1 standard uses a twisted-pair physical layer that runs at 31.25 MB/s. It is designed for device integration and process control.

The High Speed Ethernet standard is used for backbone networks in industrial plants. It is based on the 100 MB/s Ethernet standard. It can integrate devices and subsystems.

37. Define message delay in network based design. (Remembering) (CO4)

The message delay for a single message with no contention (as would be the case in a point-to-point connection) can be modeled as

$$t_m = t_x + t_n + t_r$$

Where t_x is the transmitter-side overhead, t_n is the network transmission time, and t_r is the receiver-side overhead.

38. What is meant by single hop and multi hop networks? (Remembering) (CO4)

Singlehop network: A message is received at its intended destination directly from the source, without going through any other network node.

Multihop networks: A messages are routed through network nodes to get to their destinations. (Using a multistage network does not necessarily mean using a multihop network—the stages in a multistage network are generally much smaller than the network PEs.)

39. Define Internet Protocol (IP). (Remembering) (CO4)

The Internet Protocol (IP) is the fundamental protocol on the Internet. It provides connectionless, packet-based communication. Industrial automation has long been a good application area for Internet-based embedded systems. Information appliances that use the Internet are rapidly becoming another use of IP in embedded computing.

40. How router is used to transmit data in Internet Protocol? (Remembering) (CO4)

A node that transmits data among different types of networks is known as a router. The router’s functionality must go up to the IP layer, but since it is not running applications, it does not need to go to higher levels of the OSI model.

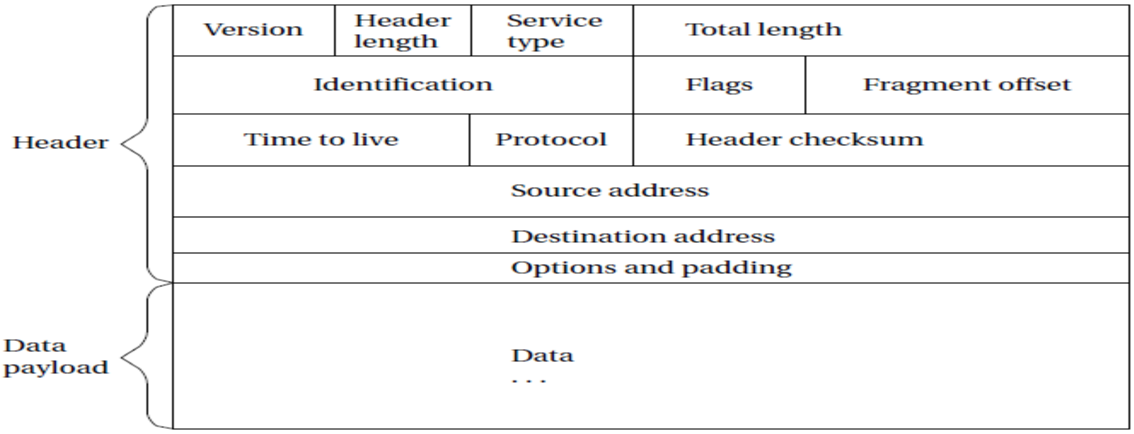
In general, a packet may go through several routers to get to its destination. At the destination, the IP layer provides data to the transport layer and ultimately the receiving application. As the data pass through several layers of the protocol stack, the IP packet data are encapsulated in packet formats appropriate to each layer.

41. Explain best-effort routing. (Understanding) (C48)

The fact that IP works at the network layer tells us that it does not guarantee that a packet is delivered to its destination. Furthermore, packets that do arrive may come out of order. This is referred to as ***best-effort routing***.

Since routes for data may change quickly with subsequent packets being routed along very different paths with different delays, real-time performance of IP can be hard to predict.

42. Show the basic format or structure of an IP packet. (Understanding) (CO4)



43. Explain the concept about TCP. (Evaluating) (CO4)

The Internet also provides higher-level services built on top of IP. The ***Transmission Control Protocol (TCP)*** is one such example. It provides a connection oriented service that ensures that data arrive in the appropriate order, and it uses an acknowledgment protocol to ensure that packets arrive. Because many higher level services are built on top of TCP, the basic protocol is often referred to as TCP/IP.

44. Show the important protocol used in IP. (Understanding) (CO4)

TCP is used to provide

- ***File Transport Protocol*** for batch file transfers,
- ***Hypertext Transport Protocol (HTTP)*** for World Wide Web service,
- ***Simple Mail Transfer Protocol*** for email, and Telnet for virtual terminals. A separate transport protocol,
- ***User Datagram Protocol*** is used as the basis for the network management services provided by the ***Simple Network Management Protocol***.

45. List out the applications of Internet Protocol (IP). (Analyzing) (CO4)

The Internet provides a standard way for an embedded system to act in concert with other devices and with users, such as:

- One of the earliest Internet-enabled embedded systems was the laser printer. High-end laser printers often use IP to receive print jobs from host machines.
- Portable Internet devices can display Web pages, read email, and synchronize calendar information with remote computers.
- A home control system allows the homeowner to remotely monitor and control home cameras, lights, and so on.

46. How the data is protected in IP and mention the types of attack in IP? Explain. (Remembering) (CO4)

Connecting an embedded system to the Internet opens up the system to the same sorts of attacks that are made on PCs and servers every day. However, attacks on embedded systems can destroy not only information but also the physical devices connected to the embedded processor. listed several example attacks that caused significant damage:

- A work infected the computer network of the CSX railway, causing all trains in the Washington, DC area to be shut down for a half day.
- A worm disabled the computer-based safety monitoring system at the Davis- Besse nuclear power plant in Ohio.
- A former consultant to a waste water plant in Australia used its computers to release one million liters of sewage into the area waterways.

47. Determine the requirements of network used for vehicles. (Evaluating) (CO4)

Networks are used for a variety of purposes in vehicles, with varying requirements on reliability and performance:

- Vehicle control (steering and brakes in cars, flight control surfaces in airplanes) is the most critical operation in the vehicle since it determines vehicle stability.
- Instruments for the driver or pilot must be reliable but often operate at higher data rates than do the vehicle control systems.
- Crew information systems may provide intercom functions, etc.
- Passenger systems provide entertainment, Internet access, etc.

48. What is meant by CAN Bus or Automotive networks? (Remembering) (CO4)

The CAN bus was designed for automotive electronics and was first used in production cars in 1991. CAN is very widely used in cars as well as in other applications.

The CAN bus uses bit-serial transmission. CAN runs at rates of 1 MB/s over a twisted pair connection of 40 m. An optical link can also be used. The bus protocol supports multiple masters on the bus. Many of the details of the CAN and I2C buses are similar, but there are also significant differences.

49. Classify the different types of terminology used in CAN bus. (Analyzing) (CO4)

Each node in the CAN bus has its own electrical drivers and receivers that connect the node to the bus in wired-AND fashion.

In CAN bus there are two terminologies, a logical 1 on the bus is called ***recessive*** and a logical 0 is ***dominant***.

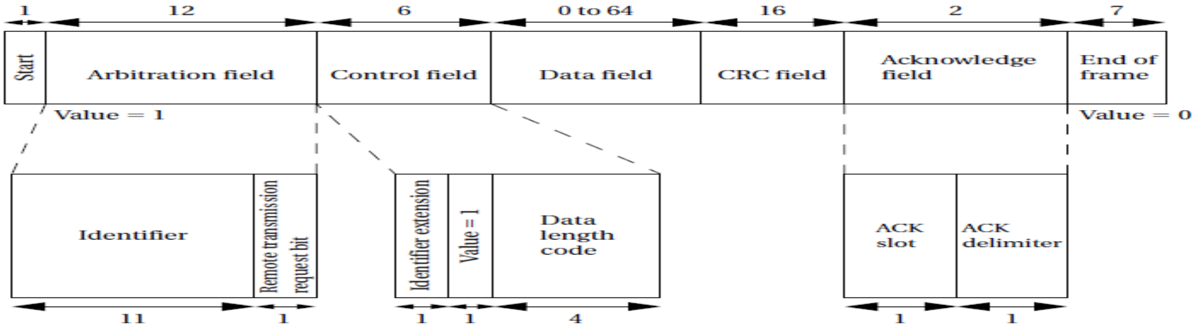
50. Explain the concept about data frame in CAN Bus. (Understanding) (CO4)

The driving circuits on the bus cause the bus to be pulled down to 0 if any node on the bus pulls the bus down (making 0 dominant over 1). When all nodes are transmitting 1s, the bus is said to be in the recessive state; when a node transmits a 0, the bus is in the dominant state. Data are sent on the network in packets known as ***data frames***.

51. Draw the CAN data frame format and Explain each term. (Understanding) (CO4)

A data frame starts with a 1 and ends with a string of seven zeroes. (There are at least three bit fields between data frames.)

- The first field in the packet contains the packet's destination address and is known as the arbitration field. The destination identifier is 11 bits long.
- The trailing remote transmission request (RTR) bit is set to 0 if the data frame is used to request data from the device specified by the identifier. When RTR_1, the packet is used to write data to the destination identifier.
- The control field provides an identifier extension and a 4-bit length for the data field with a 1 in between.
- The data field is from 0 to 64 bytes, depending on the value given in the control field. A cyclic redundancy check (CRC) is sent after the data field for error detection.
- The acknowledge field is used to let the identifier signal whether the frame was correctly received: The sender puts a recessive bit (1) in the ACK slot of the acknowledge field; if the receiver detected an error, it forces the value to a dominant (0) value. If the sender sees a 0 on the bus in the ACK slot, it knows that it must retransmit. The ACK slot is followed by a single bit delimiter followed by the end-of-frame field.



52. What is meant by FlexRay network? (Remembering) (CO4)

The FlexRay network has been designed as the next generation of system buses for cars. FlexRay provides high data rates-up to 10 MB/s-with deterministic communication. It is also designed to be fault-tolerant.

53. Explain about LIN bus and MOST bus. (Understanding) (CO4)

<u>LIN (Local Interconnect Network):</u>

The Local Interconnect Network (LIN) bus was created to connect components in a small area, such as a single door. The physical medium is a single wire that provides data rates of up to 20 KB/s for up to 16 bus subscribers. All transactions are initiated by the master and responded to by a frame. The software for the network is often generated from a LIN description file that describes the network subscribers, the signals to be generated, and the frames.

<u>MOST (Media Oriented Systems Transport):</u>

The Media Oriented Systems Transport (MOST) bus was designed for entertainment and multimedia information. The basic MOST bus runs at 24.8 MB/s and is known as MOST 25; 50 and 150 MB/s versions have also been developed. MOST can support up to 64 devices. The network is organized as a ring.

54. What are different types of channel data used in CAN bus? (Remembering) (CO4)

Data transmission is divided into channels.

- A **control channel** transfers control and system management data.
- **Synchronous channels** are used to transmit multimedia data; MOST 25 provides up to 15 audio channels.

- An **asynchronous channel** provides high data rates but without the quality-of-service guarantees of the synchronous channels.

55. Define certification and type certification in avionics. (Remembering) (CO4)

The most fundamental difference between avionics and automotive electronics is ***certification***.

Anything that is permanently attached to the aircraft must be certified. The certification process for production aircraft is twofold: first, the design is certified in a process known as ***type certification***; then, the manufacture of each aircraft is certified during production.

56. Explain line replaceable units. (Understanding) (CO4)

The certification process is a prime reason why avionics architectures are more conservative than automotive electronics systems. The traditional architecture for an avionics system has a separate unit for each function: artificial horizon, engine control, flight surfaces, etc. These units are known as ***line replaceable units*** and are designed to be easily plugged and unplugged into the aircraft during maintenance.

57. What is meant by sensor network? (Remembering) (CO4)

Sensor networks are large-scale embedded systems that may contain tens of thousands or millions of nodes. Sensors are used in a wide variety of applications: manufacturing plants, weather reporting, etc. Traditional sensor systems use custom wiring to bring data to centralized computers for analysis. Sensor networks use standardized platforms to transport data either for analysis at a server or for computing directly in the network. Sensor networks generally rely on battery-operated nodes and wireless data communication.

58. Explain the advantages of network based design. (Analyzing) (AU-Nov/Dec 2013) (CO4)

- Designed for low cost
- Provide excessively high communication speed
- To avoid bottle neck problem
- No need of load balancing

59. What is the use of attached accelerator to CPU? (Remembering) (AU-May/June 2013) (CO4)

- The CPU accelerator is attached to the CPU bus. The CPU is also called the host. The CPU talks to the accelerator through the data and control registers in the accelerator.
- Control register allow the CPU to monitor the accelerator's operation and to give the accelerator commands.
- The CPU and accelerator will communicate via shared memory. The accelerator perform read and write operation directly.
- An accelerator interface is functionally equal to an I/O device but it does not perform input or output. CPUs and accelerators perform computations for specification.

60. What are the merits of embedded distributed architecture? (Remembering) (AU-Nov/Dec 2012) (CO4)

- Error identification is easier.
- It has more cost effective performance.
- Deadliness for processing the data is short.

61. What is the role played by the accelerator in the design of embedded system? (Remembering) (AU-Nov/Dec 2012) (CO4)

- One important category of PE for embedded multiprocessor is the accelerator.
- An accelerator is attached to CPU buses to quickly execute certain key functions. It can provide large performance increase, for many applications with computational kernels.
- It can also provide critical speedups for low latency I/O functions.

62. What is priority inheritance? (Remembering) (AU-May/June 2012) (CO4)

Priority inheritance is a method of eliminating priority inversion, using this process scheduling algorithm will increase the priority of a process to the maximum priority of any process waiting for any resource on which the process has a resource lock.

63. **Justify the need of multiprocessor in embedded system design. (Evaluating) (June 2016) (CO4)**

A multiprocessor is, in general, any computer system with two or more processors coupled together. Multiprocessors used for scientific or business applications tend to have regular architectures: several identical processors that can access a uniform memory space. We use the term processing element (PE) to mean any unit responsible for computation, whether it is programmable or not.

64. **List down the various interconnect networks used for distributed embedded computing. (Analyzing) (Dec 2015) (CO4)**

- I^2C bus us used in microcontroller based systems.
- CAN bus is developed for automotive electronics.
- SHARC link ports for high speed links used on many modern DSPs.

BIG QUESTIONS

1. Discuss in detail about CPU and types of accelerators in a system. **(Creating) (AU-April 2014, Nov/Dec 2014) (CO4)**
2. Elaborate in detail about accelerated system design and its components in a system. **(Creating)(AU-April 2014, Nov/Dec 2014, Dec 2015) (CO4)**
3. Discuss in briefly about consumer electronics architecture in detail. **(Creating) (CO4)**
4. Explain how the performance of Distributed embedded network systems is analyzed. **(Understanding) (AU-Nov/Dec 2012,2013,2014) (CO4)**
5. Discuss proper examples show how communication analysis is carried out in Network for embedded Systems. **(Creating) (AU-Nov/Dec 2013,April 2014) (CO4)**
6. Discuss the share link parts and Ethernet. **(Creating) (AU-Nov/Dec 2013,April 2014) (CO4)**
7. Explain the communication protocol of I^2C bus with necessary diagrams. **(Understanding) (AU-Nov/Dec 2013,April 2014) (Dec 2015) (CO4) OR**

   Elucidate the I^2C with neat example. **(Understanding) (June 2016) (CO4)**
8. Explain the application of CAN bus in embedded networking. **(Understanding) (AU-Nov/Dec 2013,April 2014) (CO4) OR**

   Explain in detail the features, bus structure, function and applications of CAN bus.**(Understanding) (AU-April 2014) (CO4)**
9. Explain the structure and organization of CAN Bus and I^2C Bus. **(Evaluating) (AU-Nov/Dec 2013,April 2014) (CO4)**
10. Explain about accelerators and issues related network based design of embedded systems. **(Evaluating) (AU-Nov/Dec 2012, April 2014) (CO4)**
11. Discuss in detail about internet enabled system and architecture of Distributed Embedded system. **(Creating) (AU-Nov/Dec 2012, April 2014) (CO4) OR**

    Why internet enabled system is preferred for non real time interaction? Explain. **(Remembering) (Dec 2015) (CO4)**
12. Explain the following terms vehicles in networks and sensor networks in a systems. **(Understanding) (CO4)**

    Discuss the CAN bus in automotive electronics and CAN controller with example. **(Creating) (June 2016) (CO4)**
13. Explain in detail about various categories of sensor networks. **(Understanding) (CO4)**
14. Assume that I^2C bus runs at the rate of 100kbps and we need to send one 8-bit byte. How to compute the number of bits in the complete packet and find the time required to transmit, over head delay and message delay considering 20 instructions are executed. **(Applying) (Dec 2015) (CO4)**
15. Explain in detail about various performance analyses in multiprocessor system. **(Evaluating) (CO4)**
16. Discuss in detail about Ethernet and Myrinet. **(Creating) (AU-Nov/Dec 2013) (CO4)**

UNIT-V

SYSTEM DESIGN TECHNIQUES

1. Define design flow. (Remembering) (CO5)

A design flow is a sequence of steps to be followed during a design. Some of the steps can be performed by tools, such as compilers or CAD systems; other steps can be performed by hand.

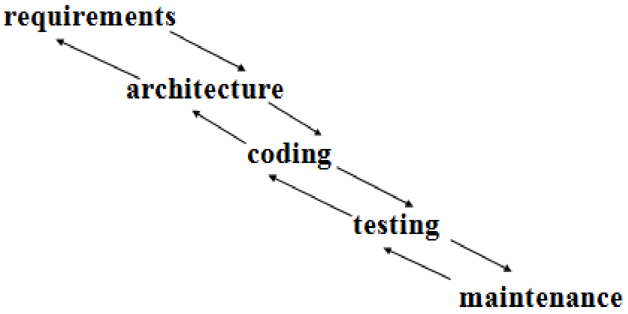
2. What are the various models in system design? (Remembering) (CO5)

- Waterfall model
- Spiral model
- Successive refinement model
- Hardware/Software design model

3. List the major level of abstractions in the embedded system design process. (Analyzing) (CO5)

- Requirements
- Specification
- Architecture
- Components
- System integration

4. List the phases in Waterfall model. (Analyzing) (CO5)



5. Compare spiral and successive refinement model in system design. (Evaluating) (CO5)

Spiral Model	Successive refinement model
The spiral model assumes that several versions of the system will be built.	In this approach, the system is built several times.
At each levels of design, the designers go through requirements, construction and testing phases.	It Provides bottom-up feedback from previous stages.
It takes less time when compared to refinement model.	It takes too much time to complete each stage.

6. List the major goals of design process. (Analyzing) (June 2016) (CO5)

- Functionality, performance
- Time-to-market
- Design cost
- Manufacturing cost, power consumption
- Quality

7. What is mean by Concurrent Engineering? (Remembering) (CO5)

Concurrent Engineering attempts to take a broader approach and optimize the total flow. Reduced design time is an important goal for concurrent engineering, but it can help with any aspect of the design that cuts across the design flow, such as reliability, performance, power consumption and so on.

8. What are the elements of concurrent engineering? (Remembering) (CO5)

- Cross-functional teams
- Concurrent product realization
- Incremental information
- Integrated project management

- Early and continual supplier involvement
- Early and continual customer focus

9. Construct the steps involved in the Concurrent Engineering applied to Telephone Systems? (Applying) (CO5)

- Benchmarking
- Breakthrough improvement
- Characterization of the current process
- Create the target process
- Verify the new process
- Implement across the product line
- Measure results and improve

10. Define Requirements. (Remembering) (CO5)

Requirements are informal descriptions of what the customer wants. It is used to creating a requirements document is effective communication between the customers and the designers.

11. What are the different types of requirements in design process? (Remembering) (CO5)

There are two types of requirements in design process. They are

(i). **Functional requirements**-It states what the system must do, such as compute an FFT.

(ii). **Non-Functional requirement**-It can be any number of other attributes, including physical size, cost, power consumption, design time, reliability, and so on.

12. Define Specification. (Remembering) (CO5)

Specifications are more detailed, precise, and consistent descriptions of the system that can be used to create the architecture.

13. List the test and requirements for the design process. (Analyzing) (CO5)

The several tests to meet the requirements are,

- Correctness
- Unambiguousness
- Completeness
- Verifiability
- Consistency
- Modifiability and Traceability

14. Choose the ways involved in traceability among requirements of design process. (Creating) (CO5)

- Able to trace backward from the requirements to know why each requirement exists.
- Able to trace forward from documents created before the requirements.
- Able to trace forward to understand how each requirement is satisfied in the implementation.
- Able to trace backward from the implementation to know which requirements they were intended to satisfy.

15. How do you determine the settings for requirements? (Remembering) (CO5)

- Customer interviews.
- Comparison with competitors.
- Sales feedback.
- Mock-ups, prototypes.
- Next-bench syndrome (HP): design a product for someone like you.

16. What type of UML languages used in specification in design process? (Remembering) (CO5)

An effective UML language used in state machine specification is **SDL Language**. It is developed by the communication industry for specifying communication protocols, telephone systems. SDL is an event-oriented state machine model since transitions between states are caused by internal and external events.

17. Choose the SDL specifications used in state machine. (Creating) (CO5)

SDL Specifications includes,

- States

- Actions
- Both conditional and unconditional transitions between states.

18. Define State chart. (Remembering) (CO5)

The State chart is one well-known technique for state-based specification that introduced some important concepts. The state chart notation uses an event-driven model. State charts allow states to be grouped together to show common functionality.

19. Identify the two basic grouping involved in state chart. (Applying) (CO5)

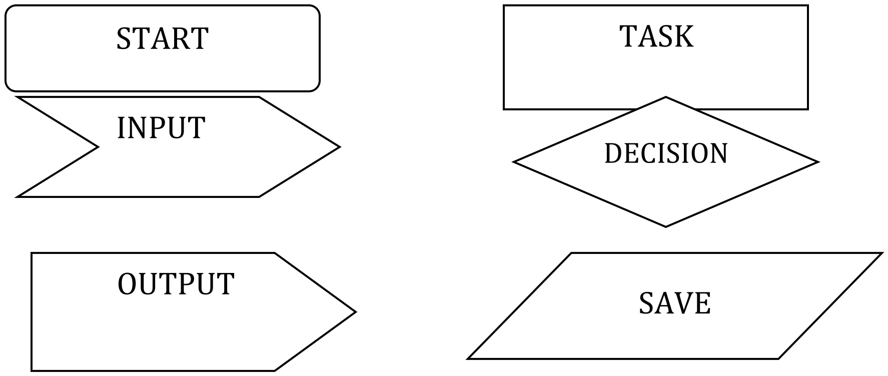
There are two basic groupings in state chart. They are

(i). OR state
(ii). AND state

20. Define CRC cards (Remembering) (CO5)

The CRC card methodology is a well-known and useful way to help analyze a system structure. It is particularly well suited to object-oriented design since it encourages the encapsulation of data and functions.

21. List the language symbols used in the SDL Specification languages. (Analyzing) (CO5)



22. Mention the use of CRC cards for system analysis. (Understanding) (Dec 2015) (CO5) OR List out the major items involved in the CRC methodology. (Analyzing) (CO5)

CRC card methodology for system analysis is useful method for understanding the overall structure of a complex system. The CRC used to identify:

Classes-define the logical groupings of data and functionality
Responsibilities-describe what the classes do
Collaborators-define other classes with which a given class works.

23. Construct the front and back layout format of a CRC Card. (Creating) (CO5)

Class name: Superclasses: Subclasses:	Class name: Class's function: Attributes:

24. Identify the steps in CRC card methodology to analyze a system. (Applying) (CO5)

- Develop an initial list of classes
- Write an initial list of responsibilities and collaborators
- Create some usage scenarios
- Walk through the scenarios
- Refine the classes, responsibilities and collaborators
- Add class relationships

25. What are the basic sets of classes in CRC card analysis of elevator system? (Remembering) (CO5)

Real-world classes: elevator car, passenger, floor control, car control, and car sensor.
Architectural classes: Car state, floor control reader, car control reader, car control sender, scheduler.

26. **List the elevator responsibilities and collaborators in CRC Card analysis. (Analyzing) (CO5)**

Class	Responsibilities	Collaborators
Elevator car*	Move up and down	Car control, car sensor, car control sender
Car control*	Transmits car requests	Passenger, floor control reader
Car state	Reads current position of car	Scheduler, car sensor

27. **Define Quality Assurance (QA). (Remembering) (CO5)**
    The quality assurance process is vital for the delivery of a satisfactory system. Its main aim is to improving the quality of the resulting system. It makes sure that all stages of the design process help to deliver a quality product.
28. **Illustrate the critical task involved in the Therac-25 medical imaging system. (Understanding) (CO5)**
    - A treatment monitor controls and monitors the setup and delivery of the treatment in eight phases.
    - A Servo task controls the radiation gun, machine motions, and so on.
    - A housekeeper task takes care of system status interlocks and limit checks.
29. **List the four major software components in Therac-25 medical imaging system. (Analyzing) (CO5)**
    - stored data
    - scheduler
    - set of tasks
    - Interrupt services.
30. **What is mean by ISO 9000? (Remembering) (CO5)**
    The International Standards Organization (ISO) has created a set of quality standards known as ISO 9000. ISO 9000 was created to apply to a broad range of industries, including but not limited to embedded hardware and software. It concentrates on processes used to create the product or service.
31. **What are the important observations to be considered in quality management based in ISO 9000? (Remembering) (CO5)**
    - Process is crucial
    - Documentation is important
    - Communication is important
32. **List out the techniques used to verify system design in ISO 9000. (Analyzing) (CO5)**
    (i). Manual based-Many of the software testing techniques can be applied manually by racing through the program to determine the required tests.
    (ii). Tool based-Tool based verification helps considerably in managing large quantities of information that may be generated in a complex design.
33. **Define CMM (Capability Maturity Model). (Remembering) (CO5)**
    One well-known way of measuring the quality of an organization's software development process is the Capability Maturity Model (CMM) developed by Carnegie Mellon University's Software Engineering Institute. The CMM provides a model for judging an organization.
34. **What are the different levels of maturity in CMM? (Remembering) (CO5)**
    Five levels of organizational maturity in CMM are,
    - **Initial**: poorly organized process depends on individuals.
    - **Repeatable**: basic tracking mechanisms.
    - **Defined**: processes documented and standardized.
    - **Managed**: makes detailed measurements.
    - **Optimizing**: measurements used for improvement.
    -

35. **What are the important requirements considered in Quality Assurance techniques? (Remembering) (CO5)**

    (i).**Prototypes-**It is a very useful tool when dealing with end users with numerous functional and nonfunctional requirements such as data display, speed of operation, size, and weight and so on.

    (ii).**Prototyping language or specification language-**It is well suited for prototyping and very high-level languages may be able to perform functional attributes, such as the mathematical function to be performed, but not nonfunctional attributes such as the speed of execution.

    (iii).**Preexisting system**-It can be used to help the end user articulate his or her needs. In some cases, it may be possible to construct a prototype of the new system from the Preexisting system.

36. **Elaborate the specification validation among Quality Assurance techniques. (Creating) (CO5)**

    (i).**Usage scenarios**-It helps designers fill out the details of a specification and ensure its completeness and correctness.

    (ii).**Formal techniques**-It makes use of mathematical proofs. Proofs may be done either manually or automatically.

37. **Explain design reviews. (Evaluating) (CO5)**

    The design review is a critical component of any quality assurance process. The design review is a simple, low cost way to catch bugs early in the design process. A design review is simply a meeting in which team members discuss a design, reviewing how a component of the system works.

38. **Classify the components available in the design review process. (Analyzing) (CO5)**

    Designers: present design to rest of team, make changes.

    Review leader: coordinates premeeting activities, the design review itself.

    Review scribe: records the minutes of meeting so that designers and other know which problems need to be fixed.

    Review audience: Audience members will naturally include other members of the project for which this component is being designed.

39. **List the requirements of alarm clock model. (Analyzing) (CO5)**

Name	-	alarm clock
Purpose	-	24-hour digital clock with one alarm
Inputs	-	set time, set alarm, hour, minute, alarm on/off
Outputs	-	four-digit display, PM indicator, alarm ready, buzzer
Functions	-	keep time, set time, set alarm, turn alarm on/off, activate buzzer by alarm
Performance	-	hours and digits, no seconds; not high precision
Manufacturing cost	-	consumer product
Power	-	AC
Physical size & weight	-	fits on stand

40. **Explain the system architecture or hardware/ software components used in alarm clock model. (Understanding) (CO5)**

    <u>Hardware components:</u>

    (i). Periodic (clock) (ii). Aperiodic (buttons, buzzer activation).

    <u>Software components:</u>

    (i). Interrupt-driven routine – can updates the current time. The current time will be kept in a variable in memory.

    (ii). Foreground program – poll the buttons and execute their commands.

41. **Discuss about the concept of Interrupt driven routine. (Creating) (CO5)**

    The interrupt-driven current time handler is how often the timer interrupts occur. A one minute interval would be very convenient for the software, but a one –minute timer would require a large number of counter bits. Use software variable to convert interrupt frequency to seconds.

42. **Identify the testing methods used in alarm clock model. (Applying) (CO5)**

<u>**Component testing:**</u>

(i). test interrupt code on the platform

(ii). can test foreground program using a mock-up.

<u>**System testing:**</u>

(i).The clock's accuracy can be checked against a reference clock

(ii).The commands can be exercised from the buttons.

(iii).The buzzer's functionality should be verified.

43. **Explain the concept and operation of Software modem. (Understanding) (CO5)**

The modem will use Frequency Shift Keying (FSK), a technique used in 1200 baud modems. The FSK scheme transmits sinusoidal tones, with 0 and 1 assigned to different frequencies. The scheme used to translate the audio input into a bit stream and analog input is sampled and the resulting stream is sent to two digital filters (such as an FIR filter). One filter passes frequencies in the range that represents a 0 and rejects the 1-band frequencies, and the other filter does the converse. The outputs of the filters are sent to detectors, which compute the average value of the signal over the past *n* samples. When the energy goes above a threshold value, the appropriate bit is detected.

44. **List the requirements of software modem model. (Analyzing) (CO5)**

Name	- Modem
Purpose	- fixed baud rate frequency-shift keyed modem
Inputs	- analog sound input, reset button
Outputs	- analog sound output, LED bit display
Functions	-**Transmitter:** sends data stored in microprocessor memory in 8-bit bytes. Sends start bit for each byte equal in length to one bit. **Receiver:** Automatically detects bytes and stores results in Main memory. Displays currently received bit on LED.
Performance	- 1200 baud
Manufacturing cost	- dominated by microprocessor and analog I/O
Power	- AC
Physical size & weight	- small and light enough to fit on a desktop.

45. **What is look back testing in software modem model? (Remembering) (CO5)**

Look-back testing in the telecommunication industry, is simpler and a good first step. Look back can be performed in two ways.

(i).A shared variable can be used to directly pass data from transmitter to the receiver.

(ii).An audio cable can be used to plug the analog output to the analog input. In this case it is also possible to inject analog noise to test the resiliency of the detection algorithm.

46. **Explain the operations of Elevator controller. (Evaluating) (CO5)**

The elevator car is the unit that runs up and down the hoistway carrying passengers; we will use N to represent the number of hoistways. Each car runs in a hoistway and can stop at any F floors. Every elevator car has a car control panel that allows the passenger to select floors to stop at. Each floor has a single floor control panel that calls for an elevator. Each floor also has a set of displays to show the current state of the elevator systems.

47. **List the components available in an Elevator controller. (Analyzing) (CO5)**

1. **Car control panel:** It have F buttons to request the floors plus an emergency stop button.
2. **Floor control panel:** It has an up button and a down button that request an elevator going in the chosen direction.
3. **Display:** It has an up light and a down light; if the elevator is idle, neither light is on.
4. **Elevator control panel:** It consists of two components.

(i). single master controller: It governs the overall behavior of all elevators

(ii). each elevator a car controller runs everything that must be done within the car.

48. **Classify the various indicator types used in the sensing elevator positions. (Understanding) (CO5)**

There are two types of indicator used in elevator control positions. They are.

(i). **coarse indicators**: It runs the entire length of the hoistway and a sensor determines when the elevator passes each one.

(ii). **Fine indicators:** They are located only around the stopping point for each floor.

49. **List the main role for master controller in Elevator controller. (Remembering) (CO5)**

(i). It must read inputs from the floor control panels.
(ii). Send signals to the lights on the floor displays.
(iii). Read floor requests from the car controllers
(iv). Take inputs from the car sensors.

50. **List the requirements of software modem model. (Analyzing) (CO5)**

Name	- Elevator systems
Inputs	- *F* floor control inputs, *N* position sensors, *N* car control Panels, one master control panel
Outputs	- *F* displays, *N* motor controllers
Functions	- Responds to floor, car, and master control panels; Operates cars safely
Performance	- Control of elevators is time critical
Manufacturing cost	- cost of electronics is small compared to mechanical systems
Power	- Not important
Physical size & weight	- Cabling is the major concern

51. **Why most designers use FOSS tools in embedded system development? (Remembering) (Nov/Dec 2012) (CO5)**

Because,

- It makes software portable.
- It speeds up the development process
- It provides good foundation for system development activities

52. **What are FOSS tools for embedded systems? (Remembering) (Nov/Dec 2013) , (May/June 2013) (CO5)**

GNU Compiler Collection (gcc) and GNU debugger (gdb) are the most popular FOSS (Free and open source) tools used in embedded systems.

53. **Discuss about H/W and S/W co-design. (Creating) (Nov/Dec 2013) (CO5)**

Embedded systems architecture design is the task of selecting and programming a suitable configuration of components for a required system application. Building an embedded system is not an easy task. Every embedded system consists of an embedded hardware and embedded software.

So software and hardware plays a main role in design of embedded system architecture.

Need For Co-Design:

- Co-design refers to parallel or concurrent development of hardware and software for an embedded system.
- Co-design reduces the overall design and development cycle of the embedded system.
- It helps the designer to find the bugs at early stage.
- It also reduces the number of errors, particularly at the hardware-software interface level.

54. **Write the relative merits of quality assurance. (Remembering) (Dec 2015) (CO5)**

The quality of a product is judged by how well it satisfies its intended function. The quality assurance processes in vital for the delivery of a satisfactory system. For example consider medical equipment like aviation electronics is a safety critical application, this medical equipment caused deeths before its design errors were properly understood.

BIG QUESTIONS

1. Explain in detail about various design methodologies. **(Understanding) (CO5)**
2. Illustrate in detail about the requirement analysis to design an embedded application. **(Creating) (Dec 2015) (CO5)**
3. Why do we need to verify the specification? Explain. **(Remembering) (Dec 2015) (CO5)**
4. Explain the system analysis and architecture design of an embedded system design. **(Understanding) (CO5)**
5. Explain the various models used in the design flow. **(Analyzing)(CO5)**
6. Draw the waterfall model of the software development process and explain. **(Analyzing) (Dec 2015) (CO5)**
7. How the software tools are implemented for design for embedded system development? **(Remembering) (CO5)**
8. Explain in detail about the design of alarm clock in detail. **(Evaluating) (June 2016) (CO5)**
9. Discuss in detail about different design process in software modem. **(Creating) (AU-Nov/Dec 2012,2013, April 2014) (CO5) OR**

   Explain in detail about the principle of operation of Software modem. **(Understanding) (Nov/Dec 2014) (June 2016) (CO5)**
10. Expalin in detail about design steps of Elevator controller with suitable diagrams**. (Evaluating) (Dec 2015) (CO5)**
11. Explain measurement driven quality assurance. **(Understanding) (CO5)**
12. Discuss the importance of Quality assurance in the development of embedded system. **(Understanding) (CO5)**

Reg. No. : □□□□□□□□□□□□

Question Paper Code : 11253

B.E./B.Tech. DEGREE EXAMINATION, MAY/JUNE 2014.

Seventh Semester

080280071 — EMBEDDED SYSTEMS

(Common to B.E. Part-Time, Sixth Semester, Electrical and Electronics Engineering)

(Regulation 2008)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. What are the applications of the embedded system?

2. What are the main components of embedded system?

3. Why is an embedded processor preferred over a microprocessor or microcontroller in an embedded system?

4. What is dynamic memory allocation?

5. List out the some applications of timer.

6. Give the significant features of PCI/X compared to PCI bus.

7. Differentiate the binary and counting semaphore.

8. Draw the process state transitions.

9. Mention the features of RTOS.

10. What is the use of action plan in the embedded system development process?

PART B — (5 × 16 = 80 marks)

11. (a) Explain in detail the various functional building blocks of embedded system. (16)

Or

(b) (i) Explain the role of timers in embedded system. (6)

(ii) Explain the operation of Interrupt controllers in embedded system. (10)

12. (a) (i) Explain the functions of various structural units of a processor in an embedded system. (12)

(ii) What is the problems of shared memory and how to overcome it? (4)

Or

(b) Explain the case study of requirement of memory devices for automatic washing machine. (16)

13. (a) (i) Explain the timer and counting devices in detail. (8)

(ii) Explain the 12C bus. (8)

Or

(b) Describe the parallel port and serial port device drivers in embedded systems. (16)

14. (a) Write short notes on :

(i) Non maskable interrupt. (8)

(ii) Writing interrupt service routine in c and assembly languages. (8)

Or

(b) (i) Discuss about the deadlock. (4)

(ii) Explain the any two types of scheduling algorithms in detail. (12)

15. (a) Discuss the role of RTOS in Interrupt handling and Task scheduling. (16)

Or

(b) (i) How to use of Target system in the embedded systems development? (6)

(ii) Explain the embedded system design issues in system development process. (10)

———————

Reg. No. □□□□□□□□□□□□

Question Paper Code : 91384

B.E./B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2014.

Eighth Semester

Electronics and Communication Engineering

EC 2042/EC 801 – EMBEDDED AND REAL TIME SYSTEMS

(Regulation 2008)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. What is the purpose of supervisor mode?

2. State the functions of coprocessor.

3. What do you mean by control bus in a CPU?

4. Can latches be used to construct input ports? Justify.

5. Define: Multitasking.

6. Give any two properties of Operating Systems.

7. What are hardware accelerators?

8. State some of the networks dedicated for embedded systems.

9. What do you mean by co-design?

10. List out the advantages of-set-top-box.

PART B — (5 × 16 = 80 marks)

11. (a) Draw the architecture of an ARM processor. Explain about the various blocks in detail. (16)

Or

(b) (i) How to evaluate CPU performance? (6)

(ii) State and explain various Instruction sets preliminaries. (10)

12. (a) Explain memory system interface with CPU, with examples. (16)

Or

(b) (i) Discuss briefly about Assembly and Linking. (10)

(ii) What are program validation and testing? (6)

13. (a) Describe in detail about the Scheduling policies with suitable examples. (16)

Or

(b) How to evaluate operating system performance? Explain. (16)

14. (a) Explain about accelerated system design with suitable diagrams. (16)

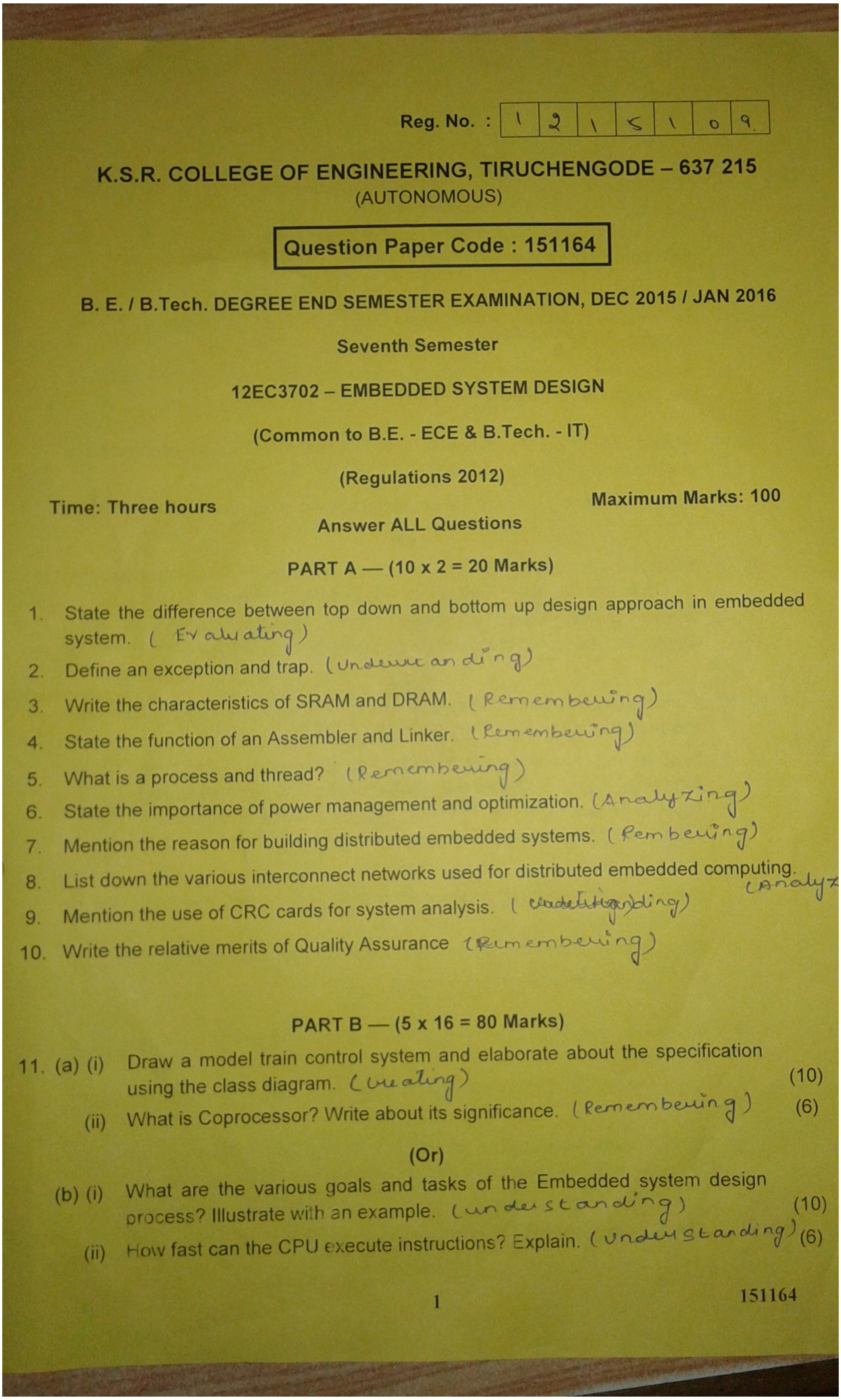
Or

(b) Discuss in detail about the distributed embedded architecture. (16)

15. (a) Describe in detail about the principle of operation of software modem. (16)

Or

(b) What are FOSS tools for embedded system development? Explain the tools in detail. (16)



Reg. No. : 1 2 1 5 1 0 9

K.S.R. COLLEGE OF ENGINEERING, TIRUCHENGODE – 637 215

(AUTONOMOUS)

Question Paper Code : 151164

B. E. / B.Tech. DEGREE END SEMESTER EXAMINATION, DEC 2015 / JAN 2016

Seventh Semester

12EC3702 – EMBEDDED SYSTEM DESIGN

(Common to B.E. - ECE & B.Tech. - IT)

(Regulations 2012)

Time: Three hours **Maximum Marks: 100**

Answer ALL Questions

PART A — (10 x 2 = 20 Marks)

1. State the difference between top down and bottom up design approach in embedded system. (Evaluating)
2. Define an exception and trap. (Understanding)
3. Write the characteristics of SRAM and DRAM. (Remembering)
4. State the function of an Assembler and Linker. (Remembering)
5. What is a process and thread? (Remembering)
6. State the importance of power management and optimization. (Analyzing)
7. Mention the reason for building distributed embedded systems. (Rembering)
8. List down the various interconnect networks used for distributed embedded computing. (Analyz
9. Mention the use of CRC cards for system analysis. (understanding)
10. Write the relative merits of Quality Assurance (Remembering)

PART B — (5 x 16 = 80 Marks)

11. (a) (i) Draw a model train control system and elaborate about the specification using the class diagram. (creating) (10)

(ii) What is Coprocessor? Write about its significance. (Remembering) (6)

(Or)

(b) (i) What are the various goals and tasks of the Embedded system design process? Illustrate with an example. (understanding) (10)

(ii) How fast can the CPU execute instructions? Explain. (understanding) (6)

12. (a) (i) Elaborate the functions of various I/O devices used in Embedded Computing systems. (creating) (10)

(ii) Draw the basic compilation process and illustrate the functional mechanism of compilation task. (understanding) (6)

(Or)

(b) (i) Explain the design of microprocessor based on the Hardware architecture of a PC. (Evaluating) (10)

(ii) Analyze the optimization of programs and explain how to validate and test the programs? (Analyzing) (6)

13. (a) (i) State the function of a scheduler. Also illustrate the basic mechanism of preemptive scheduling (understanding) (10)

(ii) How is the performance of the operating system evaluated? (Evaluating) (6)

(Or)

(b) (i) What are the various Inter process Communication mechanism provided by OS to transfer data? Explain any two in detail. (Remembering) (10)

(ii) State the advantages of priority based scheduling and how this scheduling meets out the deadline of a task or process? (Remembering) (6)

14. (a) (i) State the main function of an accelerator and illustrate how to interface this accelerator to CPU? (understanding) (10)

(ii) Assume that I^2C bus runs at the rate of 100kbps and we need to send one 8-bit byte. How to compute the number of bits in the complete packet and find the time required to transmit, overhead delay and message delay considering 20 instructions are executed. (Applying. (6)

(Or)

(b) (i) Draw the basic structure and electrical interface of an I^2C bus and explain about its mechanism. (Evaluating) (10)

(ii) Why internet enabled system is preferred for Non real time interaction? Explain. (Remembering) (6)

15. (a) (i) Illustrate in detail about the requirement analysis to design an Embedded application. (understanding) (10)

(ii) Why do we need to verify the Specification? Explain. (Remembering (6)

(Or)

(b) (i) Starting from Specification explain the design mechanism in Elevator Controller with their hardware design. (Evaluating) (10)

(ii) Draw the waterfall model of the software development process and explain (Analyzing) (6)

****

Reg. No. : | 1 | 2 | 1 | 5 | 5 | 3 | 8 |

K.S.R. COLLEGE OF ENGINEERING, TIRUCHENGODE – 637 215

(AUTONOMOUS)

Question Paper Code :152181

B. E. / B.Tech. DEGREE END SEMESTER EXAMINATION, JUNE 2016

Seventh Semester

12EC3702 – EMBEDDED SYSTEM DESIGN

(Common to B.E. - ECE & B.Tech. - IT)

(Regulations 2012)

Time: Three hours **Maximum Marks: 100**

Answer ALL Questions

PART A — (10 x 2 = 20 Marks)

1. List the challenges faced in embedded computing system design.

2. What is busy wait I/O?

3. What is four cycle handshake?

4. Differentiate mask and field programmable ROM.

5. What is Threads?

6. List the different styles in inter process communication.

7. Justify the need of multiprocessor in embedded system design.

8. What is distributed embedded system?

9. List out the design process goals.

10. Sketch the embedded system design flow.

PART B — (5 x 16 = 80 Marks)

11. (a) Discuss the top down method embedded system design development with suitable example. (16)

(Or)

(b) (i) Describe the direct mapped and set associative cache map with example. (8)

(ii) List the description method in UML and explain any one method. (8)

12. (a) What are the various debug techniques and challenges in embedded system design? Illustrate in detail. (16)

(Or)

(b) Draw the three structures commonly used in embedded software with programming and elaborate with an example. (16)

13. (a) What is priority based scheduling and explain the rate monotonic and EDF with suitable example. (16)

(Or)

(b) (i) How to assess the operating system performance? Explain. (8)

(ii) Elaborate the power management optimization in embedded system design with example (8)

14. (a) Elucidate the I^{2}C with neat example. (16)

(Or)

(b) Discuss the CAN bus in automotive electronics and CAN controller with example. (16)

15. (a) Create a microprocessor based alarm clock with neat sketch. (16)

(Or)

(b) Design a software modem with neat sketch. (16)

****