DHANALAKSHMI COLLEGE OF ENGINEERING, CHENNAI

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

EE8691 - EMBEDDED SYSTEMS

UNIT – 1: INTRODUCTION TO EMBEDDED SYSTEMS

Part - A (2 Marks)

1. What is an embedded system?

   An embedded system employs a combination of hardware & software (a “computational engine”) to perform a specific function; is part of a larger system that may not be a “computer”; works in a reactive and time-constrained environment.

2. What are the typical characteristics of an embedded system?

   Typical characteristics:
   Perform a single or tightly knit set of functions; increasingly high-performance & real-time constrained; power, cost and reliability are often important attributes that influence design; Application specific processor design can be a significant component of some embedded systems.

3. What are the advantages of embedded system

   Advantages:
   Customization yields lower area, power, cost.

4. What are the disadvantages of embedded system?

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

5. What are the applications of an embedded system?

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

6. What are the various embedded system designs?

Modelling
Refining (or "partitioning")
HW-SW partitioning

7. What are the complicating factors in embedded design? (A-11)

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

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

Hard-real time systems: where there is a high penalty for missing a deadline
e.g., control systems for aircraft/space probes/nuclear reactors; refresh rates for video, or DRAM.
Soft 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. What are the characteristics of an embedded system?
Characteristics of Embedded Systems
• Application specific
• Digital signal processing in ECS
• Reactive
• Real-time
• Distributed

10. Explain digital signal processing in embedded system Continued digitization of signals increasing the role of DSP in ES

• Signals are represented digitally as sequence of "samples"
• ADC" s are moving closer to signals

11. Give the reactivities in embedded system

Closed systems

• Execution indeterminacy confined to one source
• Causal relations are easily established.
• Open systems

• Indeterminacy from multiple sources, not controllable or observable by the programmer not possible to infer causal relations.

12. Explain distributed systems

• Consist of components that may necessarily be physically distributed.
• Consist of communicating processes on multiple processors and/or dedicated hardware connected by Communication links.
• Motivation:

- economical
- multiple processors to handle multiple time-critical tasks o physically distributed
- devices under control may be physically distributed.

13. What are the functional requirements of embedded system

Data Collection
• Sensor requirements
• Signal conditioning
• Alarm monitoring
Direct Digital Control
• Actuators
Man-Machine Interaction
• informs the operator of the current state of the controlled object
• assists the operator in controlling the system.

14. What are the various embedded system requirements? (N-13)

Types of requirements imposed by embedded applications:
• R1 Functional requirements
• R2 Temporal requirements
• R3 Dependability requirements

15. What are the temporal requirements?

Tasks may have deadlines
• Minimal latency jitter
• Minimal error detection latency
• Timing requirements due to tight software control loops
• Human interface requirements.

16. Give the classification of embedded system (A-13)

• Multi-dimensional classifications
• Hard versus software systems
• Fail-safe versus fail-operational systems
• Guaranteed-response versus best-effort
• Resource-adequate versus resource-inadequate

• Event-triggered versus time-triggered.

17. Explain the VLSI design Flow.

The design process, at various levels, is usually evolutionary in nature. It starts with a given set of requirements. Initial design is developed and tested against the requirements. When requirements are not met, the design has to be improved. If such improvement is either not possible or too costly, then the revision of requirements and its impact analysis must be considered. The Y-chart (firstintroduced by D. Gajski) shown in Fig. 1.4 illustrates a design flow for most logic chips, using design activities on three different axes (domains) which resemble the letter Y.

18. Give the VLSI design hierarchy

The use of hierarchy, or “divide and conquer” technique involves dividing a module into sub-modules and then repeating this operation on the sub-modules until the complexity of the smaller parts becomes manageable. This approach is very similar to the software case where large programs are split into smaller and smaller sections until simple subroutines.

19. What are embedded cores?

More and more vendors are selling or giving away their processors and peripherals in a form that is ready to be integrated into a programmable logic-based design. They either recognize the potential for growth in the system-on-a-chip area and want a piece of the royalties or want to promote the use of their particular FPGA or CPLD by providing libraries of ready-to-use building blocks. Either way, you will gain with lower system costs and faster time-to-market.

20. What are hybrid chips?

The vendors of hybrid chips are betting that a processor core embedded within a programmable logic device will require far too many gates for typical applications. So they\'ve created hybrid chips that are part fixed logic and part programmable logic. The fixed logic contains a fully functional processor and perhaps even some on-chip memory. This part of the chip also interfaces to dedicated address and data bus pins on the outside of the chip. Application-specific peripherals can be inserted into the programmable logic portion of the chip, either from a library of IP cores or the customer\'s own designs.

21. What do you meant by gate counts? (N-14)

The gate count by itself is almost useless. Different vendors use different measures: number of available gates, equivalent number of NAND gates, equivalent number of gates in a PLD, equivalent number of gates in an ASIC, etc. You simply can\'t compare these numbers across vendors. A better comparison can be made in terms of numbers of registers (flip-flops) and I/O pins

22. What is prototyping

Many times a CPLD or FPGA will be used in a prototype system. A small device may be present to allow the designers to change a board\'s glue logic more easily during product development and testing. Or a large device may be included to allow prototyping of a system-on-a-chip design that will eventually find its way into an ASIC. Either way, the basic idea is the same: allow the hardware to be flexible during product development. When the product is ready to ship in large quantities, the programmable device will be replaced with a less expensive, though functionally equivalent, hard-wired alternative

23. Give the internal structure of FPGA

The development of the FPGA was distinct from the PLD/CPLD evolution just described. This is apparent when you look at the structures inside. Figure 2 illustrates a typical FPGA architecture. There are three key parts of its structure: logic blocks, interconnect, and I/O blocks. The I/O blocks form a ring around the outer edge of the part. Each of these provides individually selectable input, output, or bi-directional access to one of the general-purpose I/O pins on the exterior of the FPGA package. Inside the ring of I/O blocks lies a rectangular array of logic blocks. And connecting logic blocks to logic blocks and I/O blocks to logic blocks is the programmable interconnect wiring.

24. Define FPGAs

Field Programmable Gate Arrays (FPGAs) can be used to implement just about any hardware design. One common use is to prototype a lump of hardware that will eventually find its way into an ASIC. However, there is nothing to say that the FPGA can\'t remain in the final product. Whether or not it does will depend on the relative weights of development cost and production cost for a particular project. (It costs significantly more to develop an ASIC, but the cost per chip may be lower in the long run.

26. Define PLDs

At the low end of the spectrum are the original Programmable Logic Devices (PLDs). These were the first chips that could be used to implement a flexible digital logic design in hardware. In other words, you could remove a couple of the 7400-series TTL parts (ANDs, ORs, and NOTs) from your board and replace them with a single PLD. Other names you might encounter for this class of device are Programmable Logic Array (PLA), Programmable Array Logic (PAL), and Generic Array Logic (GAL).

27. What are dependability requirements of an embedded system? (N-12)

Safety critical failure modes certification Maintainability
MTTR in terms of repairs per hour Availability
A = MTTF / (MTTF + MTTR) Security

28. Give the diversity of embedded computing

Diversity in Embedded Computing

Pocket remote control RF transmitter
100 KIPS, crush-proof, long battery life
Software optimized for size
Industrial equipment controller
1 MIPS, safety-critical, 1 MB memory
Software control loops
Military signal processing
1 GFLOPS, 1 GB/sec IO, 32 MB

Part - B (16 Marks)

1. What is the need for IDE in an Embedded Architecture? Discuss
2. Explain the VLSI design Circuits
3. Explain the software embedded systems
4. Explain the components of exemplary embedded systems
5. Describe the architecture of a typical micro controller with a neat diagram
6. Explain the basic processors and hardware units in the embedded system
7. Explain how software is embedded into a system
8. With a neat diagram, explain the architecture of Motorola, highlighting their addressing modes
9. Discuss the various addressing modes and the instruction set of a macro-controller.
10. Explain the methods used in the embedded system on a chip
11. Explain in detail about embedded system building process. (A-13)
12. Explain in detail about structural units in embedded system. (A-14)
13. What are the factors are considered in processor selection & memory device. (A-12)
14. Define following
    - Timer counter
    - Watch dog timer
    - Real time clock
    - In circuit emulator (A-13)

•

Unit – 2: EMBEDDED NETWORKING

Part A (2 Marks)

1. Give the summary of I/O devices used in embedded system

Program, data and stack memories occupy the same memory space. The total addressable memory size is 64 KB.
Program memory - program can be located anywhere in memory. Jump, branch and call instructions use 16-bit addresses, i.e. they can be used to jump/branch anywhere within 64 KB. All jump/branch instructions use absolute addressing.
Data memory - the processor always uses 16-bit addresses so that data can be placed anywhere.
Stack memory is limited only by the size of memory. Stack grows downward.
First 64 bytes in a zero memory page should be reserved for vectors used by RST instructions. I/O ports -256 Input ports -256 Output ports
Registers- Accumulator or A register is an 8-bit register used for arithmetic, logic, I/O and load/store operations.

2. Give the general registers used in embedded system.

General registers:

• 8-bit B and 8-bit C registers can be used as one 16-bit BC register pair. When used as a pair the C register contains low-order byte. Some instructions may use BC register as a data pointer.
• 8-bit D and 8-bit E registers can be used as one 16-bit DE register pair. When used as a pair the E register contains low-order byte. Some instructions may use DE register as a data pointer.
• 8-bit H and 8-bit L registers can be used as one 16-bit HL register pair. When used as a pair the L register contains low-order byte. HL register usually contains a data pointer used to reference memory addresses.
Stack pointer is a 16 bit register. This register is always incremented/decremented by 2.

3. Give the instruction sets of 8080 processor (A-12)

8080 instruction set consists of the following instructions:
• Data moving instructions.
• Arithmetic - add, subtract, increment and decrement.
• Logic - AND, OR, XOR and rotate.
• Control transfer - conditional, unconditional, call subroutine, return from subroutine and restarts.
• Input/Output instructions.
Other - setting/clearing flag bits, enabling/disabling interrupts, stack operations.

4. Give the addressing modes of 8080. Addressing modes (N-14)

Register - references the data in a register or in a register pair.

Register indirect - instruction specifies register pair containing address, where the data is located.
Direct.
Immediate - 8 or 16-bit data.

5. Define bus.

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

6. Define interrupts.

The processor supports maskable interrupts. When an interrupt occurs the processor fetches from the bus one instruction, usually one of these instructions:
• One of the 8 RST instructions (RST0 - RST7). The processor saves current program counter into stack and branches to memory location N * 8 (where N is a 3-bit number from 0 to 7 supplied with the RST instruction).
CALL instruction (3 byte instruction). The processor calls the subroutine, address of which is specified in the second and third bytes of the instruction

7. What are maskable interrupts?

Maskable Interrupts:
The processor can inhibit certain types of interrupts by use of a special interrupt mask bit. This mask bit is part of the condition code register, or a special interrupt register. If this bit is set, and an interrupt request occurs on the Interrupt Request input, it is ignored

8. What are non maskable interrupts

Non-Maskable Interrupts:
There are some interrupts which cannot be masked out or ignored by the processor. These are associated with high priority tasks which cannot be ignored (like memory parity or bus faults). In general, most processors support the Non-Maskable Interrupt (NMI). This interrupt has absolute priority, and when it occurs, the processor will finish the current memory cycle, then branch to a special routine written to handle the interrupt request.

9. Give the advantages of interrupts (N-11)

The Use of Interrupts
The use of interrupts generally falls into the following categories,
• Input/Output data transfers for peripheral devices
• Input signals to be used for timing purposes

- Emergency situations (power-down)
- Real-Time-executives/Multitasking
- Event driven programs

10. What are the steps involved in interrupt processing?

The Basic steps involved in interrupt processing are

1. Enable interrupts
2. Request an interrupt
3. Call and perform the interrupt service routine
4. Return to the previous program

11. Give the steps for accomplishing input output data transfer

ACCOMPLISHING INPUT/OUTPUT DATA TRANSFER
There are three main methods used to perform/control input/output data transfers. They are,
- software programming (scanning or polling)
- interrupt controlled
- direct memory access (DMA)

12. Give the limitations of polling technique.

The polling technique, however, has limitations.

- It is wasteful of the processors time, as it needlessly checks the status of all devices all the time.
- It is inherently slow, as it checks the status of all I/O devices before it comes back to check any given one again.
- when fast devices are connected to a system, polling may simply not be fast enough to satisfy the minimum service requirements. priority of the device is determined

13. What do you meant by bus arbitration?

Most processors use special control lines for bus arbitration, ie, controlling the use of the address and data bus,
- an input which the DMAC uses to request the bus
- an output(s) indicating the bus status
- an output indicating acceptance of the DMAC\'s bus request

14. What do you mean by asynchronous communication? (N-12)

The most basic way of sharing data is by copying the data in question to each server. Of course, this will only work if the data is changed infrequently, and always by someone with administrative access to all the servers in the cluster.

15. Give the conventions used on schematic.

• Signals are not always shown as continous lines
Each signal is given a name; if two lines on the schematic have the same name, they r connected, even though it is not explicitly shown
For eg: if one of the address line coming out of the micro proceesor is labeled “A15” then every other line labeled A15 is that same address signal..
• The actual pin numbers on the parts that will be used in the finished circuit are shown next to each signal coming out of each part.

16. What do you meant by high speed device interface?

Fail-over clustering would not be practical without some way for the redundant servers to access remote storage devices without taking a large performance hit, as would occur if these devices were simply living on the local network. Two common solutions to this problem are double-ended SCSI and fibre-channel.

17. What do you meant by timer frequency? (A-14)

A timer counts up a value every two clock cycles, and can be setup to count from an arbitrary value (Time) up to it\'s maximum value (256 in an 8-bit timer). The timer input clock is prescaled by N from the System Clock, meaning that N system cycles correspond to one timer clock cycle. Thus, the formula for base timer frequency in an 8-bit timer is Clk/2*N*(256-Time).

18. What is an IDE?

An integrated development environment (IDE) (also known as an integrated design environment and integrated debugging environment) is computer software to help computer programmers develop software.

19. Define Object oriented interfacing

In general, an interface is a device or a system that unrelated entities use to interact. According to this definition, a remote control is an interface between you and a television set, the English language is an interface between two people, and for eg protocol of behavior enforced in the military is the interface between people of different ranks.

20. What are software interrupts?

Software interrupts are still widely used by protected-mode operating systems in order to publish their internal services to the external world. However, most of these interfaces are now wrapped by functions and cannot be accessed directly (although it\'s possible, it\'s not recommended). For instance, Windows NT uses the undocumented interrupt 2Eh to allow control transfers between its user mode and kernel mode components. Whenever a non-privileged Windows application issues a call to an API function, the system routes the call until it reaches an INT 2Eh instruction.

21. What are hardware interrupts?

Hardware interrupts in protected-mode behave in the same manner as described in the real-mode section. The only major difference is that when executing in protected-mode, the processor consults its IDT rather than searching the real-mode IVT for the address of the interrupt handler.

22. What is a decoder? (N-14)

Basically, the opposite of a mux.
• Note the ``3\'\' with a slash, which signifies a three bit input. This notation represents three (1-bit) wires.
• A decoder with n input bits, produces 2^n output bits.
• View the input as ``k written an n-bit binary number\'\' and view the output as 2^n bits with the k-th bit set and all the other bits clear.
• Implement on board with AND/OR.

Part - B (16 Marks)

1. Describe the functions of a typical parallel I/O interface with a neat diagram
2. Explain high speed I/O interfacing in detail
3. Write short notes on
   (i) Analog to digital converter
   (ii) UART
4. Explain the functions of various buses used during transfer
5. Explain the synchronous and asynchronous communications from serial devices
6. Explain the various timer and counting devices
7. Explain the various bus structures used in embedded systems
8. Explain the sophisticated interfacing features in devices /ports
9. (i) Why device driver needed in system design?
   (ii) Explain in detail about CAN bus. (A-12)
10.(i) Explain in detail about SPI
   (ii) Explain in detail about I^2C. (A-15)
11. Explain in detail about Rs 232 Standards and also Rs22 & Rs 485. (N-11)
12. What are Different buses in embedded system? Explain them. (A-13)

Unit – 3: EMBEDDED FIRMWARE DEVELOPMENT ENVIRONMENT

Part - A (2 Marks)

1. What is a cross compiler? (A-13)

A Cross-Compiler is a computer program that translates a computer program written in one computer language (called the source language) into an equivalent program written in another computer language (called the output or the target language) as in a normal compiler.
A cross-compiler is distinguished by producing output in a form to be used by a Computer architecture other than the one hosting the cross-compiler. The output of a cross-compiler is usually the native machine language of the target system

2. Give the recursion concept in embedded C.

Recursion simply means applying a function as a part of the definition of that same function. Thus the definition of GNU (the source of much free software) is said to be recursive..
The key to making this work is that there must be a terminating condition such that the function branches to a non-recursive solution at some point.
A simple example. The mathematical factorial function is defined as being the product of all the numbers up to and including the argument, and the factorial of 1 is 1. Thinking about this, we see that another way to express this is that the factorial of N is equal to N times the factorial of (N-1).

3. Give the debugging strategies.

Integration and testing of software is difficult, and embedded systems provide the additional challenges of limited manipulation and visibility of the system through a small number of inputs and outputs. Abnormal system states, in particular, are difficult to test, because the system must be driven into the state before its behavior in that state can be determined.
This introduces the idea of instrumentation code injected into the implementation of UML models for the purposes of increasing the controllability, observability, and testability of the system. The instrumentation is used in both the development and the target environments, and allows interactive system debugging at the model level. In batch mode, the instrumentation serves as the basis for data collection, initialization, and test automation.

4. Give the embedded concept in UML.

The effective application of UML models to software engineering for challenging applications-especially in the embedded context-requires a development process that will ensure:
• Models are rigorous and complete
• The resulting system implementation can be optimized without impacting the models
• The overall architecture of the system is maintained by the process through multiple releases and requirement evolution

5. What are hybrid chips?

The vendors of hybrid chips are betting that a processor core embedded within a programmable logic device will require far too many gates for typical applications. So they\'ve created hybrid chips that are part fixed logic and part programmable logic. The fixed logic contains a fully functional processor and perhaps even some on-chip memory. This part of the chip also interfaces to dedicated address and data bus pins on the outside of the chip. Application-specific peripherals can be inserted into the programmable logic portion of the chip, either from a library of IP cores or the customer\'s own designs.

6. Explain interrupt handling. (N-15)

Connect one of the STK500 buttons to one of the AVR\'s external interrupt pins and write a simple routine to handle that IRQ. You must configure this external interrupt in order that a button release (a rising edge on the interrupt pin) triggers an event (e.g. prints a bit pattern in the STK500\'s leds).

```
void main(void)
{
int_handler(BUTTON, &button_handler);
int_enable();
while(1);
}
void button_handler(void) ISR
{
leds_on(buttons_chk());
}
```

7. What is an assembler?

An assembler is a computer program for translating assembly language — essentially, a mnemonic representation of machine language — into object code. A cross assembler (eg: cross compiler) produces code for one type of processor, but runs on another.

8. Define task scheduling. (A-15)

Most RTOSs do their scheduling of tasks using a scheme called \"priority-based preemptive scheduling.\" Each task in a software application must be assigned a priority, with higher priority values representing the need for quicker responsiveness. Very quick responsiveness is made possible by the \"preemptive\" nature of the task scheduling. \"Preemptive\" means that the scheduler is allowed to stop any task at any point in its execution, if it determines that another task needs to run immediately

9. Explain Boolean algebra.

Certain logic functions (i.e. truth tables) are quite common and familiar.
We use a notation that looks like algebra to express logic functions and expressions involving them. The notation is called Boolean algebra in honor of George Boole.
A Boolean value is a 1 or a 0.
A Boolean variable takes on Boolean values.
A Boolean function takes in boolean variables and produces boolean values.

10. Give the various simulators.

• Scheduling Simulator
• Deadlocking Simulator
• Memory Management Simulator
• File System Simulator

11. What are scheduling simulator? (A-14)

The scheduling simulator illustrates the behavior of scheduling algorithms against a simulated mix of process loads. The user can specify the number of processes, the mean and standard deviation for compute time and I/O blocking time for each process, and the duration of the simulation. At the end of the simulation a statistical summary is presented.

12. What are the various embedded system requirements?

Types of requirements imposed by embedded applications:
• R1 Functional requirements
• R2 Temporal requirements
• R3 Dependability requirements

13. What are the temporal requirements?

Tasks may have deadlines
• Minimal latency jitter
• Minimal error detection latency
• Timing requirements due to tight software control loops
• Human interface requirements.

14. Give the classification of embedded system.

• Multi-dimensional classifications
• Hard versus software systems
• Fail-safe versus fail-operational systems

- Guaranteed-response versus best-effort
- Resource-adequate versus resource-inadequate
- Event-triggered versus time-triggered.

15. Define Object oriented interfacing.

In general, an interface is a device or a system that unrelated entities use to interact. According to this definition, a remote control is an interface between you and a television set, the English language is an interface between two people, and for eg protocol of behavior enforced in the military is the interface between people of different ranks.

Part - B (16 Marks)

1. Explain the concepts of embedded programming
2. Explain the features of assemblers, combilers and cross combilers used in embedded systems
3. Explain Interrupt service routine Queues
4. What is an interrupt? What are the types of interrupts? Explain
5. What is interrupt latency? Explain
6. Explain the C.Program elements, macros and functions used in embedded systems..
7. Explain the use of function calls in embedded programming
8. Explain how to optimize memory codes
9. Explain the various multiple function calls in Embedded C
10. Discuss the scheduling architecture and the algorithms used in embedded software development
11. Give the difference between programming in assembly language (ALP) and high level language
12. What are the different phases of EDLC and explain them. (N-15)
13. Explain in detail about modeling of EDLC. (N-15)
14. (i)Issues in hardware-software Co-design.
    (ii) Explain in detail about data flow graph? (N-14)
15. Explain in detail about state machine model. (N-14)
16. How the embedded system designed by using sequential program model? (A-12)

Unit – 4: RTOS BASED EMBEDDED SYSTEM DESIGN

Part A (2 Marks)

1. What is a thread?

   A thread otherwise called a lightweight process (LWP) is a basic unit of CPU utilization, it comprises of a thread id, a program counter, a register set and a stack. It shares with other threads belonging to the same process its code section, data section, and operating system resources such as open files and signals.

2. What are the benefits of multithreaded programming? (A-15)

   The benefits of multithreaded programming can be broken down into four major categories:
   Responsiveness
   Resource sharing
   Economy
   Utilization of multiprocessor architectures

3. Compare user threads and kernel threads. User threads Kernel threads

   User threads are supported above the kernel and are implemented by a thread library at the user level
   Kernel threads are supported directly by the operating system
   Thread creation & scheduling are done in the user space, without kernel intervention. Therefore they are fast to create and manage Thread creation, scheduling and management are done by the operating system. Therefore they are slower to create & manage compared to user threads
   Blocking system call will cause the entire process to block If the thread performs a blocking system call, the kernel can schedule another thread in the application for execution

4. Define thread cancellation & target thread. (N-12)

   The thread cancellation is the task of terminating a thread before it has completed. A thread that is to be cancelled is often referred to as the target thread. For example, if multiple threads are concurrently searching through a database and one thread returns the result, the remaining threads might be cancelled.

5. What are the different ways in which a thread can be cancelled?

   Cancellation of a target thread may occur in two different scenarios:

   Asynchronous cancellation: One thread immediately terminates the target thread is called asynchronous cancellation.

Deferred cancellation: The target thread can periodically check if it should terminate, allowing t he target thread an opportunity to terminate itself in an orderly fashion.

6. Define CPU scheduling.

CPU scheduling is the process of switching the CPU among various processes. CPU scheduling is the basis of multiprogrammed operating systems. By switching the CPU among processes, the operating system can make the computer more productive.

7. What is preemptive and non preemptive scheduling?

Under non preemptive scheduling once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or switching to the waiting state. Preemptive scheduling can preempt a process which is utilizing the CPU in between its execution and give the CPU to another process.

8. What is a Dispatcher? (A-14)

The dispatcher is the module that gives control of the CPU to the process selected by the short-term scheduler. This function involves:
Switching context
Switching to user mode
Jumping to the proper location in the user program to restart that program.

9. What is dispatch latency?

The time taken by the dispatcher to stop one process and start another running is known as dispatch latency.

10. What are the various scheduling criteria for CPU scheduling?

The various scheduling criteria are

CPU utilization Throughput Turnaround time Waiting time Response time

11. Define throughput? (A-15)

Throughput in CPU scheduling is the number of processes that are completed per unit time. For long processes, this rate may be one process per hour; for short transactions, throughput might be 10 processes per second.

12. What is turnaround time?

Turnaround time is the interval from the time of submission to the time of completion of a process. It is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.

13. Define race condition.

When several process access and manipulate same data concurrently, then the outcome of the execution depends on particular order in which the access takes place is called race condition. To avoid race condition, only one process at a time can manipulate the shared variable.

14. What is critical section problem?

Consider a system consists of „n„ processes. Each process has segment of code called a critical section, in which the process may be changing common variables, updating a table, writing a file. When one process is executing in its critical section, no other process can allowed to execute in its critical section.

15. What are the requirements that a solution to the critical section problem must satisfy?

The three requirements are

Mutual exclusion
Progress
Bounded waiting

16. Define entry section and exit section.

The critical section problem is to design a protocol that the processes can use to cooperate. Each process must request permission to enter its critical section. The section of the code implementing this request is the entry section. The critical section is followed by an exit section. The remaining code is the remainder section.

17. What is semaphores? (N-12)

A semaphore „S" is a synchronization tool which is an integer value that, apart from initialization, is accessed only through two standard atomic operations; wait and signal. Semaphores can be used to deal with the n-process critical section problem. It can be also used to solve various synchronization problems.
The classic definition of „wait"

```
wait (S)
{
while (S<=0)
;
S--;
```

}
The classic definition of „signal"
signal (S)
{ S++;
}

18. When the error will occur when we use the semaphore?

i. When the process interchanges the order in which the wait and signal operations on the semaphore mutex.
ii. When a process replaces a signal (mutex) with wait (mutex).
iii. When a process omits the wait (mutex), or the signal (mutex), or both.

19. Define deadlock.

A process requests resources; if the resources are not available at that time, the process enters a wait state. Waiting processes may never again change state, because the resources they have requested are held by other waiting processes. This situation is called a deadlock.

20. What is the sequence in which resources may be utilized?

Under normal mode of operation, a process may utilize a resource in the following sequence: Request: If the request cannot be granted immediately, then the requesting process must wait until it can acquire the resource.
Use: The process can operate on the resource. Release: The process releases the resource.

21. What are conditions under which a deadlock situation may arise?

A deadlock situation can arise if the following four conditions hold simultaneously in a system:
1. Mutual exclusion
2. Hold and wait
3. No pre-emption
4. Circular wait

Part - B (16 Marks)

1. Explain how thread an process are used in embedded system.
2. Explain process management amd memory management in embedded system
3. Explain file system organization and implementation
4. Explain how interrupt routines handled in embedded system
5. Explain round robin scheduling
6. Explain the real time operating systems
7. Explain cyclic scheduling with time slicing
8. Explain how critical section in handled by a pre-emptinve scheduler
9. Explain interprocess communication ams synchronization
10. Explain interprocess communications using signals
11. Give the functions for message Queues and mail boxes
12. Explain remote procedure call with an example
13. Explain the various RTOS Task scheduling models.

14.(i)Explain in detail about internet routines in RTOS.

(ii) Explain Multi processing &Multi Tasking. (A-13)

15.(i) Compare preemptive & Non preemptive scheduling

(ii) Define massage passing. Explain different massage passing technique in embedded System. (N-14)

16. (i)Explain in detail inter process communication.

(ii) Comparisons of VXWORK, MC/DOS, μc OS II, RT Linux. (N-15)

17. Define Following

- Mallbox
- Pipes
- Priority Inversion
- Priority Inheritance (N-11)

Unit – 5: EMBEDDED SYSTEM APPLICATION DEVELOPMENT

Part - A (2 Marks)

1. Define RTOS. (N-15)

A real-time operating system (RTOS) is an operating system that has been developed for real-time applications. It is typically used for embedded applications, such as mobile telephones, industrial robots, or scientific research equipment.

2. Define task and task rates. (A-14)

An RTOS facilitates the creation of real-time systems, but does not guarantee that they are real-time; this requires correct development of the system level software. Nor does an RTOS necessarily have high throughput — rather they allow, through specialized scheduling algorithms and deterministic behavior, the guarantee that system deadlines can be met. That is, an RTOS is valued more for how quickly it can respond to an event than for the total amount of work it can do. Key factors in evaluating an RTOS are therefore maximal interrupt and thread latency

3. Explain memory allocation in embedded system.

Memory allocation is even more critical in a RTOS than in other operating systems. Firstly, speed of allocation is important. A standard memory allocation scheme scans a linked list of indeterminate length to find a suitable free memory block; however, this is unacceptable as memory allocation has to occur in a fixed time in a RTOS.
Secondly, memory can become fragmented as free regions become separated by regions that are in use. This can cause a program to stall, unable to get memory, even though there is theoretically enough available. Memory allocation algorithms that slowly accumulate fragmentation may work fine for desktop machines—when rebooted every month or so—but are unacceptable for embedded systems that often run for years without rebooting

4. Explain multi task and their functions in embedded system. (A-13)

This system implements cooperative and time-sliced multitasking, provides resource locking and mailbox services, implements an efficient paged memory manager, traps and reports errors, handles interrupts, and auto starts your application at system startup. By following some simple coding practices as shown in the documented coding examples, you can take advantage of these sophisticated features without having to worry about the implementation details.

5. Explain memory allocation functions. (A-15)

Determinism of service times is also an issue in the area of dynamic allocation of RAM memory. Many general-computing non-real-time operating systems offer memory allocation services from what is termed a \"Heap.\" The famous \"malloc\" and \"free\" services known to C-language programmers work from a heap

6. What do you meant by high speed message passing?

   Intertask message communication is another area where different operating systems show different timing characteristics. Most operating systems actually copy messages twice as they transfer them from task to task via a message queue

7. What is a fixed time task switching? (A-12)

   The time it takes to do task switching is of interest when evaluating an operating system. A simple general-computing (non-preemptive) operating system might do task switching only at timer tick times, which might for example be ten milliseconds apart. Then if the need for a task switch arises anywhere within a 10-millisecond timeframe, the actual task switch would occur only at the end of the current 10-millisecond period. Such a delay would be unacceptable in most real-time embedded systems.

8. Give the queue related function for embedded system.

   To connect a message queue, or create it if it doesn\'t exist. The call to accomplish this is the msgget() system call:
   int msgget(key_t key, int msgflg);
   msgget(): returns the message queue ID on success, or -1 on failure (and it sets errno, of course.) The first, key is a system-wide unique identifier describing the queue you want to connect to (or create). Every other process that wants to connect to this queue will have to use the same key.

9. Give the fuction for sending a queue.

   Each message is made up of two parts, which are defined in the template structure struct msgbuf, as defined in sys/msg.h:
   struct msgbuf { long mtype; char mtext[1];
   };
   The field mtype is used later when retrieving messages from the queue, and can be set to any positive number. mtext is the data this will be added to the queue

10. Give a function for receiving a message from a queue.

    A call to msgrcv() that would do it looks something like this:

```
#include
key_t key;
int msqid;
struct pirate_msgbuf pmb; /* where L\'Olonais is to be kept */
key = ftok(\"/home/beej/somefile\", \'b\');
msqid = msgget(key, 0666 | IPC_CREAT);
msgrcv(msqid, &pmb, sizeof(pmb), 2, 0); /* get him off the queue! */
```

11. Give the steps to destroy a message queue. There are two ways:
    1. Use the Unix command ipcs to get a list of defined message queues, then use the command ipcrm to delete the queue.
    2.Write a program to do it for you

12. Give the needs for memory management.
    Each new model of computer seems to come with more main memory than the last, but, since the memory requirements of the software rise just as fast, memory is always a precious commodity, hence the need for memory management .
    • Memory is allocated to a process when needed
    • Memory is deallocated when no longer in use
    • Swapping allows the total memory used by all the running processes to exceed main memory
    • Virtual memory makes it possible to run a single program that uses more memory than the main memory (normally RAM) available on the system. Virtual memory is normally divided into pages .
    • Programs refer to parts of memory using addresses . In a virtual memory system, these are virtual addresses. The virtual address is mapped onto a physical addresses by a memory management unit (MMU)

13. Explain file system.
    A file system is where the user stores his/her data. The operating system aims to:
    • make storing and retrieving files as easy as possible for the user
    • provide good performance. Make efficient use of the hardware

14. What do you meant by extended memory?
    The extra memory beyond the first megabyte is known as extended memory. The primary difference between EMS and extended memory is that EMS memory will work with any Intel processor including the 8088. Extended memory is only available for computers based on the 286 and later chips. Then there is the question of usefulness. Until recently, few DOS programs used extended memory. This situation is changing, though, as more programs support extended memory through XMS (the eXtended Memory Standard). XMS allows programs to share extended memory.

15. Give the timer delay functions. (N-11)

```
typedef struct Timer Timer;
 typedef void (*TimerAction)(Timer *t);
 struct Timer {
 App * app; /* associated App */
 int milliseconds; /* interval between actions */ int remaining; /* time until action */ Timer Action
 action; /* user-defined action */ void * data; /* user-defined data */
 int value; /* user-defined value */
```

Part - B (16 Marks)

1. Give the RTOS system level functions
2. Explain the various time delay functions
3. Explain in detail about Memory allocation related functions
4. Explain briefly about mailbox related functions
5. Explain Queue related functions
6. Explain multiple tasks and their functions
7. Give the steps for creating list of tasks
8 .Give the exemplary coding techniques
9. List and explain the various task service functions
10. Explain how IPCs are used in embedded system.
11. Explain the various multiple function calls in Embedded C
12. Explain in detail about designing washing machine. (A-10)
13.Explain the various steps involved in Automatic Application. (N-09)
14.Explain in detail about one of the smart card system application. (N-14)
15.Explain in detail about various types involved in designs embedded system. (A-13)