K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BANGALORE - 560109
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
SESSION: 2021-2022 (EVEN SEMESTER)
I SESSIONAL TEST QUESTION PAPER
SET-B

USN | | | | | | | | | | |

Degree	:	B.E	Semester	:	VI A & B
Branch	:	Electronics and Communication Engineering	Course Code	:	18EC62
Course Title	:	Embedded Systems	Date	:	16/05/2022
Duration	:	90 Minutes	Max Marks	:	30

Note: Answer ONE full question from each part.

Q No.	Question	Marks	K-Level	CO mapping
	PART-A			
1(a)	**Build** the operational modes and privilege levels in Cortex M3 processor.	5	Applying (K3)	CO1
(b)	**Explain** Bus interface of ARM Cortex M3.	5	Understanding (K2)	CO1
(c)	**Explain** any 2 load and store instructions with example.	5	Understanding (K2)	CO2
	OR			
2(a)	**Construct** two stack model and reset sequence in ARM Cortex M3.	5	Applying (K3)	CO1
(b)	**Explain** MPU of ARM Cortex M3.	5	Understanding (K2)	CO1
(c)	**Explain** shift and rotate instructions available in ARM Cortex M3 processors.	5	Understanding (K2)	CO2
	PART-B			
3(a)	**Make use of** architectural block diagram of ARM-Cortex M3 processor and explain briefly.	5	Applying (K3)	CO1
(b)	**Explain** memory mapping with diagram.	5	Understanding (K2)	CO1
(c)	**Make use of** instruction set of ARM and explain the following with example: MRS, PUSH, REV16	5	Applying (K3)	CO2
	OR			
4(a)	**Make use of** diagrams to explain register organization of Cortex M3.	5	Applying (K3)	CO1
(b)	**Explain** any 5 applications of ARM Cortex M3 processors.	5	Understanding (K2)	CO1
(c)	**Analyze** the following instructions and write the contents of the registers after the execution of each instruction: Assume R8=0x00000088, R9=0x00000006 and R3=0x00001111 I. RSB.W R8, R9, #0x10 II. ADD R8,R9,R3 III. ORR R8,R9	5	Applying (K3)	CO2

Course Incharge

HOD ECE

Professor & Head
Dept. of Electronics & Communication Engg.
K. S. School of Engineering & Management
Bangalore-560 109

IQAC- Coordinator

Principal

Principal / Director
K.S. School of Engineering & Management
Bangalore-560 062

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BANGALORE - 560109
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
SESSION: 2021-2022 (EVEN SEMESTER)
I SESSIONAL TEST QUESTION PAPER
SET-A

USN | | | | | | | | | | |

Degree	:	B.E	Semester	:	VI A & B
Branch	:	Electronics and communication Engineering	Course Code	:	18EC62
Course Title	:	Embedded Systems	Date	:	16/05/2022
Duration	:	90 Minutes	Max Marks	:	30

Note: Answer ONE full question from each part.

Q No.	Question	Marks	K-Level	CO mapping
	PART-A			
1(a)	**Make use of** architectural block diagram of ARM-Cortex M3 processor and explain briefly.	5	Applying (K3)	CO1
(b)	**Explain** any 5 applications of ARM Cortex M3 processors.	5	Understanding (K2)	CO1
(c)	**Explain** shift and rotate instructions available in ARM Cortex M3 processors.	5	Understanding (K2)	CO2
	OR			
2(a)	**Build** the operational modes and privilege levels in Cortex M3 processor.	5	Applying (K3)	CO1
(b)	**Explain** register organization of Cortex M3.	5	Understanding (K2)	CO1
(c)	**Explain** any 2 load and store instructions with example.	5	Understanding (K2)	CO2
	PART-B			
3(a)	**Construct** two stack model and reset sequence in ARM Cortex M3.	5	Applying (K3)	CO1
(b)	**Explain** any 5 advantages of ARM Cortex M3 processors.	5	Understanding (K2)	CO1
(c)	**Analyze** the following instructions and write the contents of the registers after the execution of each instruction: Assume R8=0x00000088, R9=0x00000006 and R3=0x00001111 I. RSB.W R8, R9, #0x10 II. ADD R8,R9,R3 III. ORR R8,R9	5	Applying (K3)	CO2
	OR			
4(a)	**Construct** features of NVIC and explain briefly.	5	Applying (K3)	CO1
(b)	Mention the instructions used for accessing the special registers. **Explain** the same using suitable examples.	5	Understanding (K2)	CO1
(c)	**Make use of** instruction set of ARM and explain the following with example: MRS, PUSH, REV16	5	Applying (K3)	CO2

Course Incharge | HOD ECE | IQAC- Coordinator | Principal

Professor & Head
Dept. of Electronics & Communication Engineering
K. S. School of Engineering & Management
Bangalore-560 109

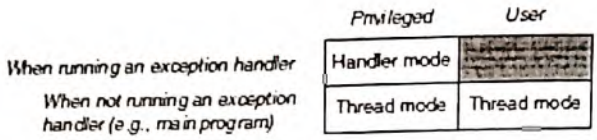
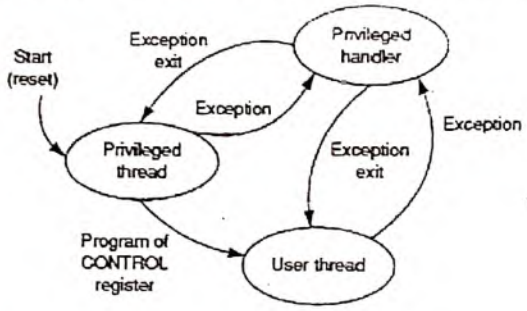
Principal / Director
K.S. School of Engineering & Management
Bangalore-560 062

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BENGALURU-560109
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
I – SESSION: 2021-2022 (EVEN SEMESTER)
SCHEME OF VALUATION
SET-B

Degree	:	B.E	Semester	:	VI A & B
Branch	:	Electronics & Communication Engg.	Date	:	16 -05-2022
Course Title	:	Embedded Systems	Course Code	:	18EC62
Duration	:	90 Minutes	Max Marks	:	30

Note: Answer ONE full question from each part

Q. No.	Scheme and Solution	Marks
1(a)	**Build** the operational modes and privilege levels in Cortex M3 processor. • The Cortex-M3 processor has two operation modes and two privilege levels. • The two operation modes are: 1. Thread mode 2. Handler mode • The operation modes determine whether the processor is running a normal program or running an exception handler like an interrupt handler or system exception handler. • There are 2 privilege levels: 1. Privilege level 2. User level • The privilege levels provide a mechanism for safeguarding memory accesses to critical regions as well as providing a basic security model. • When the processor is running a main program (thread mode), it can be either in a privileged state or a user state, but exception handlers can only be in a privileged state. • In the privileged state, a program has access to all memory ranges and can use all supported instructions. • Software in the privileged access level can switch the program into the user access level using the control register. When an exception takes place, the processor will always switch back to the privileged state and return to the previous state when exiting the exception handler.	2 marks for diagram 3 marks for explanation

	 **Figure 1.8 Operation Modes and Privilege Levels in Cortex-M3.** • A user program cannot change back to the privileged stare by writing to the control register. • It has to go through an exception handler that programs the control register to switch the processor back into the privileged access level when returning to thread mode. • The separation of privileged and user mode improves system reliability by preventing system configuration registers from being accessed or changed by some untrusted programs. 	
(b)	**Explain** Bus interface of ARM Cortex M3. The bus interface allows the Cortex-M3 to carry instruction fetches and data accesses at the same time. The main bus interfaces are as follows: 1. Code memory buses 2. System buses 3. Private peripheral bus The code memory region access is carried out on the code memory buses, which physically consist of two buses, one called I-Code and other called D-Code. These are optimized for instruction fetches for best instruction execution speed. The system bus is used to access memory and peripherals. This provides access to the Static Random Access Memory (SRAM), peripherals, external RAM, external devices, and part of the system level memory regions. The private peripheral bus provides access to a part of the system-level memory dedicated to private peripherals, such as debugging components.	**5 marks for explanation**
(c)	**Explain** any 2 load and store instructions with example. • The basic instructions for accessing memory are Load and Store.	

- Load (LDR) transfers data from memory to registers.
- Store transfers data from registers to memory.

The transfers can be in different data sizes (byte, half word, word, and double word)

2.5 marks for each

`LDRB Rd, [Rn, #offset]`	Read byte from memory location Rn + offset
`LDRH Rd, [Rn, #offset]`	Read half word from memory location Rn + offset
`LDR Rd, [Rn, #offset]`	Read word from memory location Rn + offset
`LDRD Rd1,Rd2, [Rn, #offset]`	Read double word from memory location Rn + offset
`STRB Rd, [Rn, #offset]`	Store byte to memory location Rn + offset
`STRH Rd, [Rn, #offset]`	Store half word to memory location Rn + offset
`STR Rd, [Rn, #offset]`	Store word to memory location Rn + offset
`STRD Rd1,Rd2, [Rn, #offset]`	Store double word to memory location Rn + offset

Examples on LDR instruction

Before Execution

R1 →

Address	Data
1000002A	0xABCDEF54H
1000002E	0x12345678H
10000032	0x24567893H
10000036	0x88564478H

Example 1	Example 2	Example 3
LDRB R2, [R1,#4]	LDRH R2, [R1, #4]	LDR R2, [R1, #4]
Operation: R2=mem8[R1+4]	***Operation***: R2= mem16[R1+4]	***Operation***: R2= mem32 [R1+4]
After Execution: R2=93H	***After Execution:*** R2=7893H	***After Execution:*** R2=0x24567893H

Examples on STR instruction

Before Execution

R1 →

Address	Data
1000002A	0xABCDEF54
1000002E	0x12345678
10000032	0x24567893
10000036	0x88564478

Example 1	Example 2
STRB R2, [R1, #4]	STRH R2, [R1,#4]
Before Execution: R2=0x324565CDH	***Before Execution:*** R2=0x324565CDH
Operation: mem8[R1+4]=R2	***Operation:*** Mem16[R1+4]= R2
After Execution: R1 →	***After Execution:*** R1 →

Example 1 — After Execution:

Address	Data
1000002A	0xABCDEF54H
1000002E	0x12345678H
10000032	0x245678***CDH***
10000036	0x88564478H

Example 2 — After Execution:

Address	Data
1000002A	0xABCDEF54H
1000002E	0x12345678H
10000032	0x2456***65CDH***
10000036	0x88564478H

2(a)

Construct two stack model and reset sequence in ARM Cortex M3. — **3 marks for two stack model**

- The Cortex-M3 has two SPs: the MSP and the PSP
- The SP register to be used is controlled by the control register bit 1 (CONTROL [1]).
- When CONTROL [1] is 0, the MSP is used for both thread mode and handler mode.
  1. In this arrangement, the main program and the exception handlers share the same stack memory region.
  2. This is the default setting after power-up.

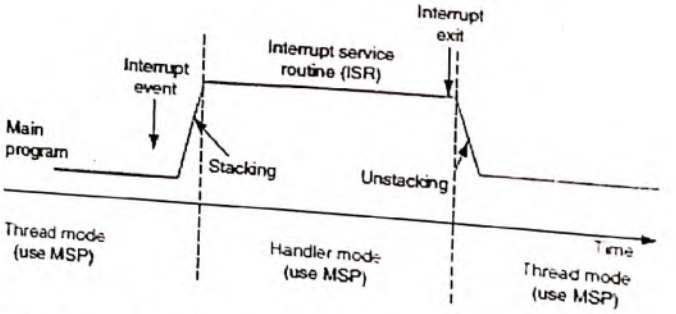


Figure 1.21 Cortex CONTROL [1]=0: Both Thread Level and Handler Use Main Stack.

- When the CONTROL[1] is 1, the PSP is used in thread mode
  1. In this arrangement, the main program and the exception handler can have separate stack memory regions.

2 marks for

2. This can prevent a stack error in a user application from damaging the stack used by the OS.
3. The automatic stacking and unstacking mechanism will use PSP, whereas stack operations inside the handler will use MSP.

reset sequence

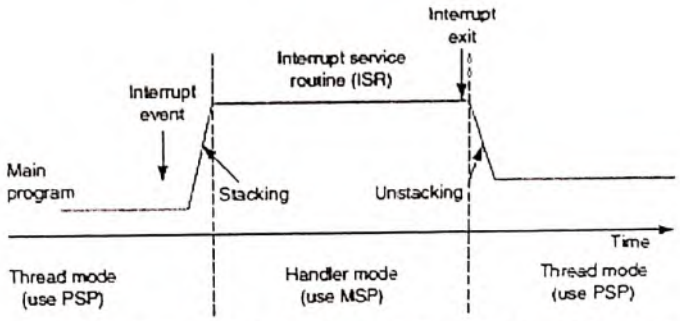


Figure 1.22 CONTROL[1]=1: Thread Level Uses Process Stack and Handler Uses Main Stack.

- It is possible to perform read/write operations directly to the MSP and PSP, without any confusion of which R13 you are referring to.
- Provided that you are in privileged level, you can access MSP and PSP values:

```
x = __get_MSP(); // Read the value of MSP
__set_MSP(x); // Set the value of MSP
x = __get_PSP(); // Read the value of PSP
__set_PSP(x); // Set the value of PSP
```

- In general, it is not recommended to change current selected SP values in a C function, as the stack memory could be used for storing local variables.
- To access the SPs in assembly, you can use the MRS and MSR instructions:

```
MRS R0, MSP  ; Read Main Stack Pointer to R0
MSR MSP, R0  ; Write R0 to Main Stack Pointer
MRS R0, PSP  ; Read Process Stack Pointer to R0
MSR PSP, R0  ; Write R0 to Process Stack Pointer
```

- By reading the PSP value using an MRS instruction, the OS can read data stacked by the user application (such as register contents before SVC).
- In addition, the OS can change the PSP pointer value—for example, during context switching in multitasking systems.

Reset Sequence

After the processor exits reset, it will read two words from memory (see Figure 3.18):

- Address 0x00000000: Starting value of R13 (the SP)
- Address 0x00000004: Reset vector (the starting address of program execution; LSB should be set to 1 to indicate Thumb state)

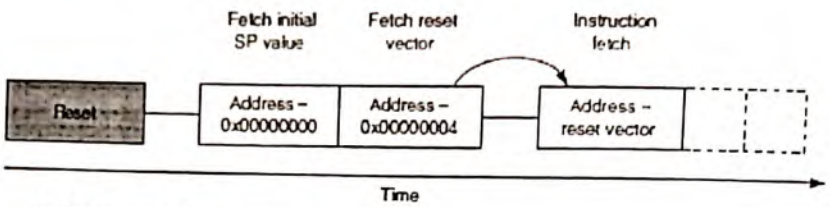


Figure 1.23 Reset Sequence.

- In the Cortex-M3, the initial value for the MSP is put at the beginning of the memory map, followed by the vector table, which contains vector address values.
- Because the stack operation in the Cortex-M3 is a full descending stack (SP decrement before store), the initial SP value should be set to the first memory after the top of the stack region.
- For example, if you have a stack memory range from 0x20007C00 to 0x20007FFF (1 KB), the initial stack value should be set to 0x20008000.
- The vector table starts after the initial SP value.
- The first vector is the reset vector.
- In the Cortex-M3, vector addresses in the vector table should have their LSB set to 1 to indicate that they are Thumb code.
- For that reason, the previous example has 0x101 in the reset vector, whereas the boot code starts at address 0x100 in figure1.24

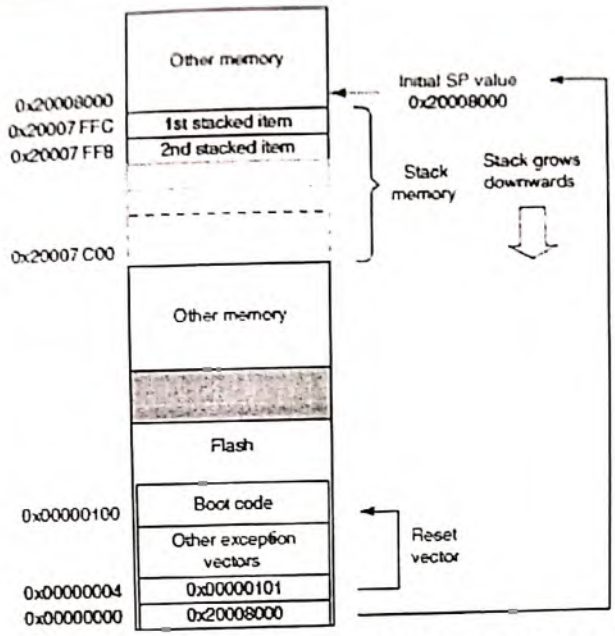
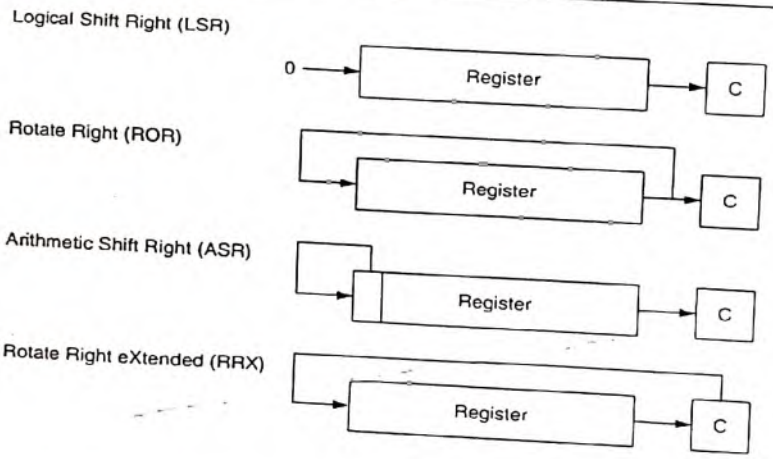


Figure 1.24 Initial Stack Pointer Value and Initial Program Counter Value Example.

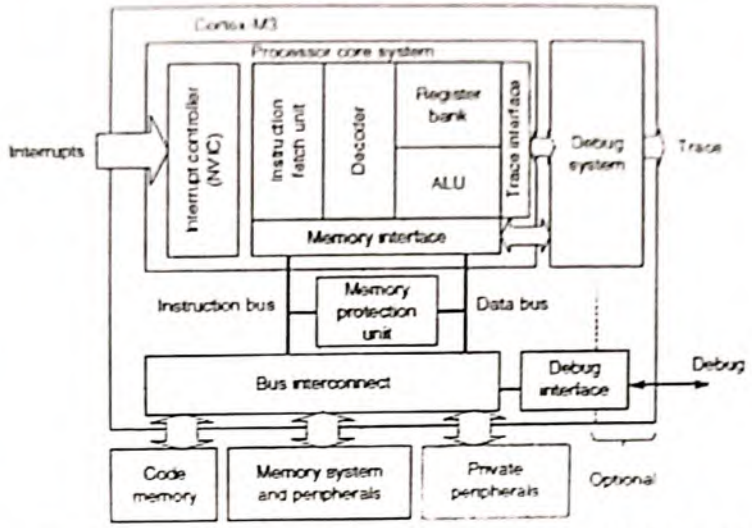
- After the reset vector is fetched, the Cortex-M3 can then start to execute the program from the reset vector address and begin normal operations.
- It is necessary to have the SP initialized, because some of the exceptions (such as NMI) can happen right after reset, and the stack memory could be required for the handler of those exceptions.

(b)	**Explain** MPU of ARM Cortex M3. • The Cortex-M3 has an optional MPU. • This unit allows access rules to be set up for privileged access and user program access. • The MPU can be used in various ways. In common scenarios, the OS can set up the MPU to protect data use by the OS kernel and other privileged processes to be protected from untrusted user programs. • The MPU can also be used to make memory regions read-only, to prevent accidental erasing of data or to isolate memory regions between different tasks in a multitasking system. • It makes embedded systems more robust and reliable. • The MPU feature is optional and is determined during the implementation stage of the microcontroller SoC design.	**5 marks for explanation**
(c)	**Explain** shift and rotate instructions available in ARM Cortex M3 processors.	

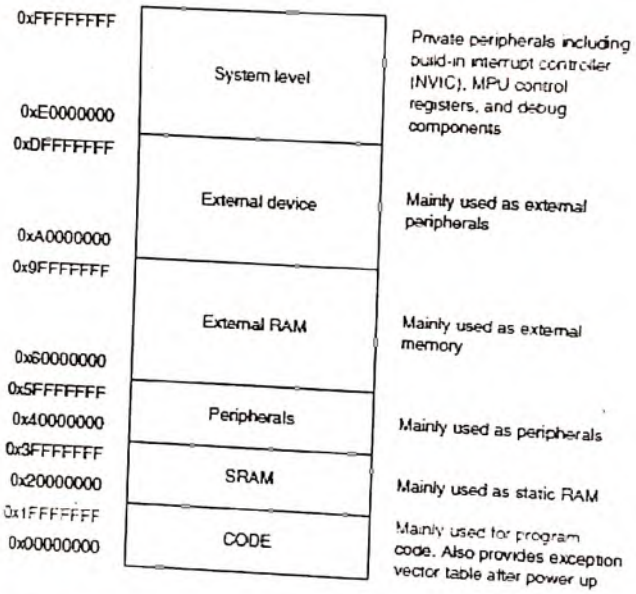
Instruction	Operation	Description	
ASR Rd, Rn, #immd	Rd = Rn >> immd	Arithmetic shift right	1 mark for each
ASR Rd, Rn	Rd = Rd >> Rn		
ASR.W Rd, Rn, Rm	Rd = Rn >> Rm		
LSL Rd, Rn, #immd	Rd = Rn << immd	Logical shift left	
LSL Rd, Rn	Rd = Rd << Rn		
LSL.W Rd, Rn, Rm	Rd = Rn << Rm		
LSR Rd, Rn, #immd	Rd = Rn >> immd	Logical shift right	
LSR Rd, Rn	Rd = Rd >> Rn		
LSR.W Rd, Rn, Rm	Rd = Rn >> Rm		
ROR Rd, Rn	Rd rot by Rn	Rotate right	
ROR.W Rd, Rn, #immd	Rd = Rn rot by immd		
ROR.W Rd, Rn. Rm	Rd = Rn rot by Rm		
RRX,W Rd, Rn	{C, Rd} = {Rn, C}	Rotate right extended	



3(a)	**Make use of** architectural block diagram of ARM-Cortex M3 processor and explain briefly.	3 marks for diagram

	 • The Cortex-M3 is a 32-bit microprocessor. It has a 32-bit data path, a 32-bit register bank, and 32- bit memory interfaces. • The processor has a Harvard architecture, which means that it has a separate instruction bus and data bus. This allows instructions and data accesses to take place at the same time, and as a result of this, the performance of the processor increases because data accesses do not affect the instruction pipeline. • The instruction and data buses share the same memory space (a unified memory system). In other words, you cannot get 8 GB of memory space just because you have separate bus interfaces. • For complex applications that require more memory system features, the Cortex-M3 processor has an optional Memory Protection Unit (MPU), and it is possible to use an external cache if it's required. Both little endian and big endian memory systems are supported.	**2 marks for explanation**
(b)	**Explain** memory mapping with diagram. The Cortex-M3 has a predefined memory map. This allows the built-in peripherals, such as the interrupt controller and the debug components, to be accessed by simple memory access instructions. Thus, most system features are accessible in C program code. The predefined memory map also allows the Cortex-M3 processor to be highly optimized for speed and ease of integration in system-on-a-chip (SoC) designs. The Cortex-M3 design has an internal bus infrastructure optimized for this memory usage. In addition, the design allows these regions to be used differently. For example, data memory can still be put into the CODE region, and program code can be executed from an external	**5 marks for explanation**

Random Access Memory
(RAM) region.
System level memory contains the interrupt controller and the debug components.
By having fixed addresses for these peripherals, you can port applications between different Cortex-M3 products more easily.
Overall, the 4 GB memory space can be divided into ranges as shown below:



(c) **Make use of** instruction set of ARM and explain the following with example:
MRS, PUSH, REV16
MRS <general purpose reg>,<special register>
MSR < special register >,< general purpose reg >
Example:
MRS R0, PSR ; Read Processor status word into R0.
MSR CONTROL, R1 ; Write value of R1 into control register

PUSH<reg> ; Decrement and store
POP <reg> ; Get and increment
Multiple register PUSH and POP operation
PUSH {R0, R4-R7, R9}; Push R0, R4, R5, R6, R7, R9 into stack memory
POP {R2,R3}; Pop R2 and R3 from stack
PUSH {R0-R3, LR}; Save register contents at beginning of subroutine processing.
POP {R0-R3, PC}; restore registers and return

REV16 Rd, Rn
Rd = rev16(Rn)
Reverse bytes in each half word

2 marks for MRS and PUSH

1 mark for REV16

Before Execution: R0 = 0x1234789AH, R1 = 0x9A23C8ABH
Instruction: REV16 R0, R1
After Execution:
R1 = 0x9A23C8ABH R0 = 0x 239AABC8H

Make use of diagrams to explain register organization of Cortex M3.

4(a)

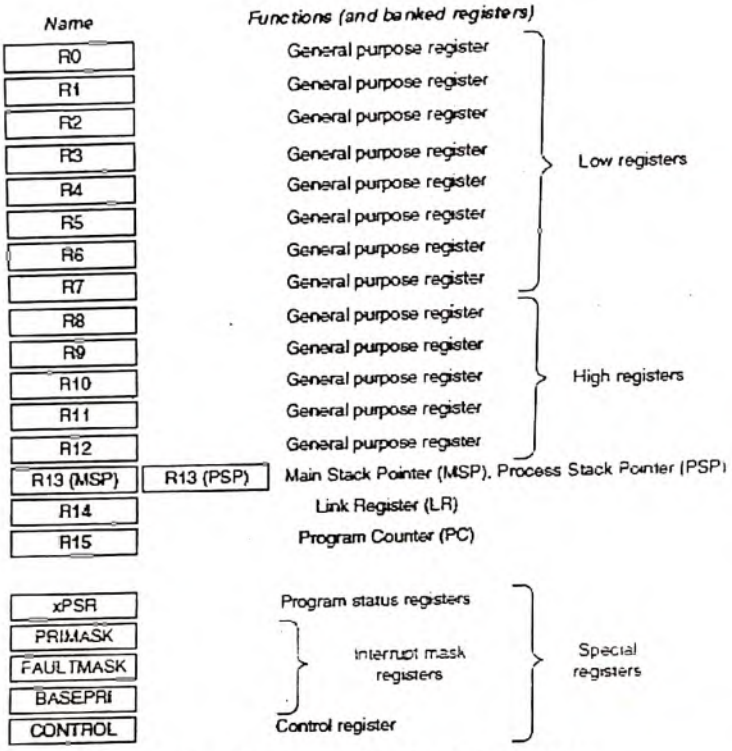


Figure 1.6 Registers in the Cortex M3.

- The Cortex-M3 processor has registers R0 through R15. R13 (the stack pointer) is banked, with only one copy of the R13 visible at a time.
- R0–R12: General-Purpose Registers: R0–R12 is 32-bit general-purpose registers for data operations.
    1. Low Registers, R0-R7: Some 16-bit Thumb instructions can only access a subset of these registers. They can be accessed by all 16-bit Thumb instructions and all 32-bit Thumb-2 instructions. The reset value is unpredictable.
    2. High Registers, R8-R12: They can be accessed by all 32-bit Thumb-2 instructions but not by all 16-bit Thumb instructions. These registers are 32 bits; the reset value is unpredictable.
- R13: Stack Pointers: The Cortex-M3 contains two stack pointers (R13). They are banked so that only one is visible at a time.

3 marks for general purpose register

2 marks for special registers

They have two separate memories. When using the register name R13, you can only access the current SP; the other one is inaccessible unless you use special instructions to move to special register from general-purpose register (MSR) and move special register to general purpose register (MRS).

- The two stack pointers are as follows:
  1. Main Stack Pointer (MSP) or SP_main: The default stack pointer, used by the operating system (OS) kernel and exception handlers and all application codes that require privileged access.
  2. Process Stack Pointer (PSP) or SP_Proces: This is used by the base-level application code (when not running an exception handler).
- R14: The Link Register (LR): Inside an assembly program, you can write it as either R14 or LR. When a subroutine is called, the return address is stored in the link register.
- R15: The Program Counter: You can access it in assembler code by either R15 or PC. The program counter is the current program address. This register can be written to control the program flow.

Because of the pipelined nature of the Cortex-M3processor, when you read this register, you will find that the value is different than the location of the executing instruction, normally by 4.

For example,

0x1000: mov R0, PC; // R0=0x1004

- Special Registers: The Cortex-M3 processor also has a number of special registers:
  1. Program Status registers (PSRs)
  2. Interrupt Mask registers (PRIMASK, FAULTMASK, and BASEPRI)
  3. Control register (CONTROL)

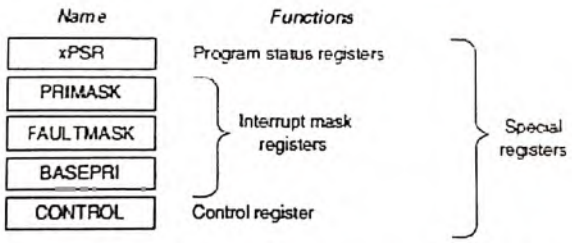


Figure 1.7 Special Registers in the Cortex M3.

These registers have special functions and can be accessed only by special instructions. They cannot be used for normal data processing.

Register	Function
xPSR	Provide arithmetic and logic processing flags (zero flag and carry flag), execution status, and current executing interrupt number
PRIMASK	Disable all interrupts except the nonmaskable interrupt (NMI) and hard fault
FAULTMASK	Disable all interrupts except the NMI
BASEPRI	Disable all interrupts of specific priority level or lower priority level
CONTROL	Define privileged status and stack pointer selection

(b)	**Explain** any 5 applications of ARM Cortex M3 processors. 1. Low-cost microcontrollers: The Cortex-M3 processor is ideally suited for low-cost microcontrollers, which are commonly used in consumer products, from toys to electrical appliances. It is a highly competitive market due to the many well-known 8-bit and 16-bit microcontroller products on the market. Its lower power, high performance, and ease-of- use advantages enable embedded developers to migrate to 32-bit systems and develop products with the ARM architecture. 2. Automotive: It has an ideal application in automotive industry. It has a very high performance efficiency and low interrupt latency, allowing it to be used in real time systems. 3. Data communications: The processor has low power and high efficiency, along with many instructions in Thumb-2, so it is ideal for many communication applications like Bluetooth and ZigBee. 4. Industrial control: In industrial control applications, simplicity, fast response, and reliability are key factors. Because of the features of cortex M3, like, low latency and enhanced fault handling features, it is ideal for industry applications. 5. Consumer products: In many consumer products, a high-performance microprocessor is used. Because of small processor, low power consumption, high efficiency, and robust memory protection, Cortex M3 is a best choice for consumer products.	**1 mark for each**

(c)	**Analyze** the following instructions and write the contents of the registers after the execution of each instruction: Assume R8=0x00000088, R9=0x00000006 and R3=0x00001111 I. RSB.W R8, R9, #0x10 II. ADD R8,R9,R3 III. ORR R8,R9 I. RSB.W R8,R9,#0x10 Operation: R8= 0x10-R9 0x00000010-0x00000006= 0x0000000A R8=0x0000000A II. ADD R8,R9,R3 Operation: R8=R9+R3 R8=0x00000006+0x00001111 R8=0x00001117 III. ORR R8,R9 Operation: R8=R8||R9 R8=0x00000088||0x00000006 R8=0x0000008E	2 marks for RSB and ADD 1 mark for ORR

Course Incharge

HOD ECE

Professor & Head
Dept. of Electronics & Communication Engineering
K. S. School of Engineering & Management
Bangalore-560 109

Principal

CO3

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BANGALORE - 560109
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
SESSION: 2021-2022 (EVEN SEMESTER)
II SESSIONAL TEST QUESTION PAPER
SET-A

USN | | | | | | | | | | |

Degree : B.E
Branch : Electronics and communication Engineering
Course Title : Embedded Systems
Duration : 90 Minutes

Semester : VI A & B
Course Code : 18EC62
Date : 16/06/2022
Max Marks : 30

Note: Answer ONE full question from each part.

Q No.	Question	Marks	K-Level	CO mapping
	PART-A			
1(a)	**Make use of** ARM instructions to calculate the sum of 1 to 10 numbers.	5	Applying (K3)	CO2
(b)	**Explain** the 6 purpose of embedded systems with an example for each.	5	Understanding (K2)	CO3
(c)	Define Embedded systems. **Make use of** block diagram to explain the elements of embedded systems.	5	Applying (K3)	CO3
	OR			
2(a)	**Make use of** neat diagram to explain the organization of CMSIS and its benefits.	5	Applying (K3)	CO2
(b)	**Differentiate** between i) RISC and CISC ii) Big Endian and Little Endian architecture	5	Understanding (K2)	CO3
(c)	**Construct** the following: I2C bus and IrDA	5	Applying (K3)	CO3
	PART-B			
3(a)	**Make use of** ARM instructions to blink LED using 'C' Language.	5	Applying (K3)	CO2
(b)	**Illustrate** different types of ROMs.	5	Understanding (K2)	CO3
(c)	**Build** the classification of embedded system based on Generation and triggering.	5	Applying (K3)	CO3
	OR			
4(a)	**Discover** how CMSIS provides standard access interface for embedded software.	5	Applying (K3)	CO2
(b)	**Explain** ZigBee, Bluetooth communication interface.	5	Understanding (K2)	CO3
(c)	**Construct** the application areas of Embedded system.	5	Applying (K3)	CO3

Course Incharge | HOD ECE | IQAC- Coordinator | Principal

Professor & Head
Dept. of Electronics & Communication Engineering
K. S. School of Engineering & Management
Bengalore-560 109

Dr. K. RAMA NARASIMHA
Principal/Director
K S School of Engineering and Management
Bengaluru - 560 109

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BENGALURU-560109
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
-SESSION: 2021-2022 (EVEN SEMESTER)
SCHEME OF VALUATION
SET-A

Degree	:	B.E	Semester	:	VI A & B
Branch	:	Electronics & Communication Engg.	Date	:	16 -05-2022
Course Title	:	Embedded Systems	Course Code	:	18EC62
Duration	:	90 Minutes	Max Marks	:	30

Note: Answer ONE full question from each part

Q. No.	Scheme and Solution	Marks

1(a)	**Make use of** ARM instructions to calculate the sum of 1 to 10 numbers.
	AREA RESET, DATA, READONLY
	EXPORT __Vectors
	__Vectors
	DCD 0x20001000 ; stack pointer value when stack is empty
	DCD Reset_Handler ; reset vector
	ALIGN
	AREA MYCODE, CODE, READONLY
	ENTRY
	EXPORT Reset_Handler
	Reset_Handler
	MOV R0, #0 ; R0 accumulates total
	MOV R1, #10 ; R1 counts from 10 down to 1
	again ADD R0, R0, R1
	SUBS R1, R1, #1
	BNE again
	halt B halt ; infinite loop to stop computation
		5 marks for code

(b)	**Explain** the 6 purpose of embedded systems with an example for each. Each embedded system is designed to serve the purpose of any one or a combination of the following tasks: 1. Data Collection/Storage/Representation 2. Data Communication 3. Data (Signal) Processing 4. Monitoring 5. Control 6. Application Specific User Interface Data Collection/Storage/Representation □ An embedded system designed for the purpose of data collection performs acquisition of data from the external world. □ Data collection is usually done for storage, analysis, manipulation and transmission. □ The term "data" refers all kinds of information, viz. text, voice, image, video, electrical signals and any other measurable quantities.	**1 mark for each**

- Data can be either analog (continuous) or digital (discrete).
- The collected data may be stored or transmitted or it may be processed or it may be deleted instantly after giving a meaningful representation.
- A digital camera is a typical example of an embedded system with data collection/storage/representation of data.

Data Communication

- Embedded data communication systems are deployed in applications ranging from complex satellite communication systems to simple home networking systems.
- The transmission is achieved either by a wire- line medium or by a wireless medium. Data can either be transmitted by analog means or digital means.
- The data collecting embedded terminal itself can incorporate data communication units like wireless modules (Bluetooth, ZigBee, Wi-Fi, EDGE, GPRS, etc.) or wire-line modules (RS- 232C, USB, TCP/IP, PS2, etc.).

Data (Signal) Processing

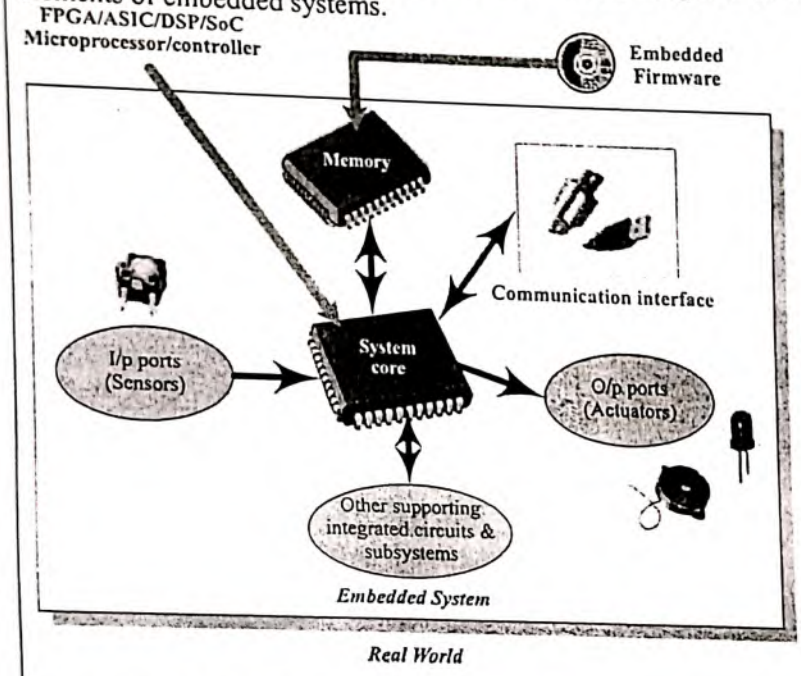
- The data (voice, image, video, electrical signals and other measurable quantities) collected by embedded systems may be used for various kinds of data processing.
- Embedded systems with signal processing functionalities are employed in applications demanding signal processing like speech coding, synthesis, audio video codec, transmission applications, etc.
- A digital hearing aid is a typical example of an embedded system employing data processing.

Monitoring

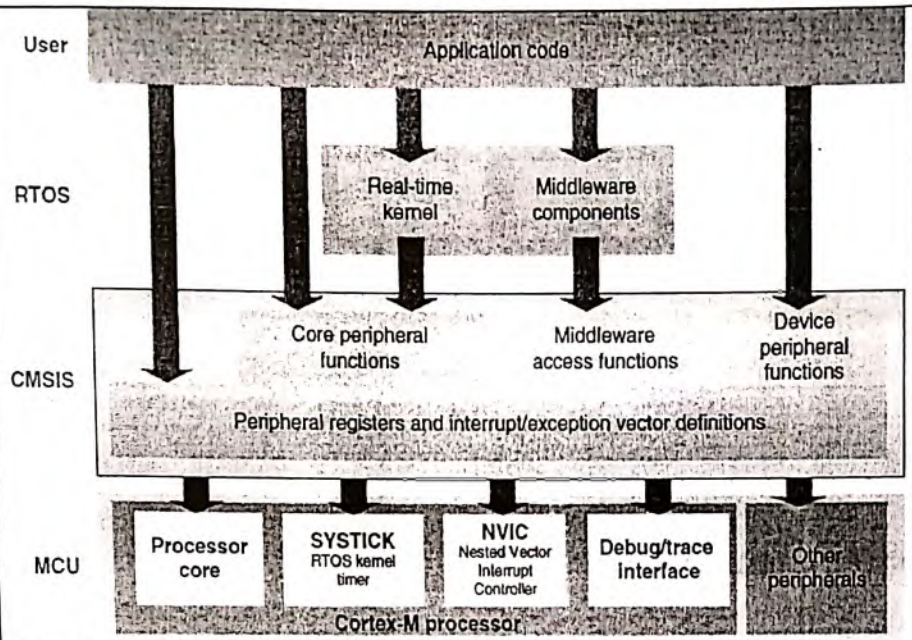
- Almost embedded products coming under the medical domain are used for monitoring.
- A very good example is the electro cardiogram (ECG) machine for monitoring the heartbeat of a patient.
- The machine is intended to do the monitoring of the heartbeat.
- It cannot impose control over the heartbeat.

Control

- Embedded systems with control functionalities impose control over some variables according to the changes in input variables.
- A system with control functionality contains both sensors and actuators.

	□ Sensors are connected to the input port for capturing the changes in environmental variable or measuring variable. □ The actuators connected to the output port are controlled according to the changes in input variable to put an impact on the controlling variable to bring the controlled variable to the specified range. Application Specific User Interface □ These are embedded systems with application-specific user interfaces like buttons, switches, keypad, lights, bells, display units, etc. □ Mobile phone is an example for this.	
(c)	Define Embedded systems. Make use of block diagram to explain the elements of embedded systems.  It contains a single chip controller, which acts as the master brain of the system. □ The controller can be 1. A microprocessor or 2. A microcontroller or 3. A Field Programmable Gate Array (FPGA) device or 4. A Digital Signal Processor (DSP) or 5. An Application Specific Integrated Circuit (ASIC)/Application Specific Standard Product (ASSP) □ An embedded system can be viewed as a reactive system. □ The control is achieved by processing the information coming from the sensors and user interfaces, and controlling some actuators that regulate the physical variable. □ Key boards, push button switches, etc. are examples for common user interface input devices.	**2 marks for block diagram** **3 marks for explanation**

	LEDs, liquid crystal displays, piezoelectric buzzers, etc. are examples for common user interface output devices for a typical embedded system. □ The memory of the system is responsible for holding the control algorithm and other important configuration details. □ For most of embedded systems, the memory for storing the algorithm or configuration data is of fixed type, which is a kind of Read Only Memory (ROM). □ It is not available for the end user for modifications □ The memory is protected from unwanted user interaction by implementing some kind of memory protection mechanism. □ The most common types of memories used in embedded systems for control algorithm storage are OTP, PROM, UVEPROM, EEPROM and FLASH. □ Sometimes the system requires temporary memory for performing arithmetic operations or control algorithm execution and this type of memory is known as "working memory". □ Random Access Memory (RAM) is used in most of the systems as the working memory. □ Various types of RAM like SRAM, DRAM and NVRAM are used for this purpose. □ Apart from these, communication interface is essential for communicating with various subsystems of the embedded system and with the external world. □ The communication interfaces may be used to achieve onboard (I2C, SPI, UART, parallel bus interface, etc.) or external communication (wireless interfaces like Infrared, Bluetooth, Wi-Fi, etc.)	
2(a)	**Make use of** neat diagram to explain the organization of CMSIS and its benefits. The CMSIS is divided into multiple layers as follows: □ Core Peripheral Access Layer: Name definitions, address definitions, and helper functions to access core registers and core peripherals. □ Middleware Access Layer: Common method to access peripherals for the software industry. Targeted communication interfaces include Ethernet, UART and SPI. Allows portable software to perform communication tasks on any Cortex microcontrollers that support the required communication interface. □ Device Peripheral Access Layer (MCU specific): Name definitions, address definitions, and driver code to access peripherals Access Functions for Peripherals (MCU specific): Optional additional helper functions for peripherals	**2 marks for diagram**



3 marks for explanation

Benefits of CMSIS

- Better software portability and reusability.
- Allows software to be quickly ported between Cortex-M3 and other Cortex-M processors, reducing time to market. For embedded OS vendors and middleware providers, the advantages of the CMSIS are significant.
- By using the CMSIS, their software products can become compatible with device drivers from multiple microcontroller vendors.

Without the CMSIS, the software vendors either have to include a small library for Cortex-M3 core functions or develop multiple configurations of their product so that it can work with device libraries from different microcontroller vendors.

(b) **Differentiate** between

I. RISC and CISC

II. Big Endian and Little Endian architecture

RISC	CISC
1. Lesser number of instructions	1. Greater number of instructions
2. Instruction pipelining and increasedexecution speed	Generally no instruction pipelining feature
3. Orthogonal instruction set (Allows each instruction to operate on any register and use any addressing mode)	Non-orthogonal instruction set (All instructions are not allowed to operate on any register and use any addressing mode. It is instruction-specific)
4. Operations are performed on registersonly, the only memory operations are	Operations are performed on registersor memory depending on

2.5 marks for each

load and store	instruction
5. A large numb o registe ar available er f rs e	5. Limited number of general purpose registers
6. Programmer needs to write more code to execute a task since the instructions are simpler ones	Instructions are like macros in C language. A programmer can achieve the desired functionality with a single instruction which in turn provides the effect of using more simpler single instructions in RISC
7. Single, fixed length instructions	7. Variable length instructions
8. Less silicon usage and pin count	More silicon usage since more additional decoder logic is required to implement the complex instructiondecoding
9. With Harvard Architecture	9. Can be Harvard or Von-Neumann Architecture

Big-Endian vs. Little-Endian Processors/Controllers

- Endianness specifies the order in which the data is stored in the memory by processor operations in a multi byte system.
- Suppose the word length is two byte then data can be stored in memory in two different ways:
    1. Higher order of data byte at the higher memory and lower order of data byte at location just below the higher memory - Little-Endian

E.g.: Intel x86 Processors

2. Lower order of data byte at the higher memory and higher order of data byte at location just below the higher memory - Big-Endian

E.g.: Motorola 68000 Series Processors

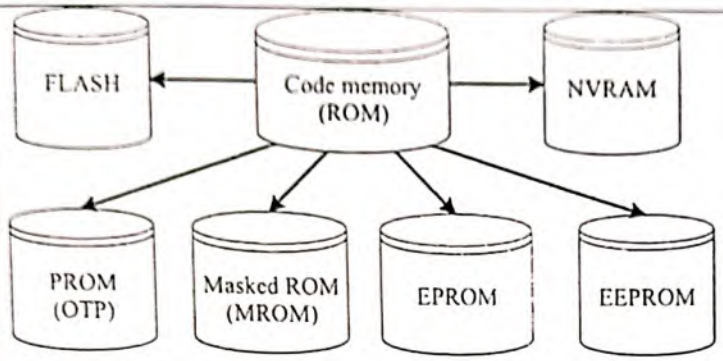
- Little-endian means the lower-order byte of the data is stored in memory at the lowest address, and the higher-order byte at the highest address. (The little end comes first.)

(c)	**Construct** the following: I2C bus and IrDA Inter Integrated Circuit (I2C) Bus □ The Inter Integrated Circuit Bus (I2C or I2C Pronounced 'I square C') is a synchronous bi-directional half duplex two wire serial interface bus. □ (Half duplex - one-directional communication at a given point of	**2.5 marks for each**

	very familiar with it. □ E.g.: The remote control of TV, VCD player, etc. works on Infrared. □ Infrared communication technique uses infrared waves of the electromagnetic spectrum for transmitting the data. □ It supports point-point and point-to-multipoint communication, provided all devices involved in the communication are within the line of sight. □ The typical communication range for IrDA lies in the range 10 cm to 1 m. □ The range can be increased by increasing the transmitting power of the IR device. □ IR supports data rates ranging from 9600bits/second to 16Mbps. □ Depending on the speed of data transmission IR is classified into: □ Serial IR (SIR) – supports data rates ranging from 9600bps to 115.2kbps. □ Medium IR (MIR) – supports data rates of 0.576Mbps and 1.152Mbps. □ Fast IR (FIR) – supports data rates up to 4Mbps. □ Very Fast IR (VFIR) – supports data rates up to 16Mbps. Ultra Fast IR (UFIR) – supports data rates up to 96Mbps. □ GigaIR – supports data rates 512 Mbps to 1 Gbps. □ IrDA communication involves a transmitter unit for transmitting the data over IR and a receiver for receiving the data. □ Infrared Light Emitting Diode (LED) is the IR source for transmitter and at the receiving end a photodiode acts as the receiver. □ Both transmitter and receiver unit will be present in each device supporting IrDA communication for bidirectional data transfer. Such IR units are known as 'Transceiver'.	

3(a)	**Make use of** ARM instructions to blink LED using 'C' Language.
	#define LED *((volatile unsigned int *)(0xDFFF000C)) int main (void)
	{
	int i; /* loop counter for delay function */
	volatile int j; /* dummy volatile variable to prevent C compiler from optimize the delay away */
	while (1) {
	LED = 0x00; /* toogle LED */
	for (i=0;i<10;i++) {j=0;} /* delay */ LED = 0x01; /* toogle LED */
	for (i=0;i<10;i++) {j=0;} /* delay */
	}
	return 0;
	}
		5 marks for each

(b)	**Illustrate** different types of ROMs.	



2 marks for diagram

3 marks for explanation

Masked ROM (MROM)

- Masked ROM is a one-time programmable device.
- Masked ROM makes use of the hardwired technology for storing data.
- The device is factory programmed by masking and metallisation process at the time of production itself, according to the data provided by the end user.

Programmable Read Only Memory (PROM) / (OTP)

- One Time Programmable Memory (OTP) or PROM is not pre-programmed by the manufacturer.
- The end user is responsible for programming these devices.
- This memory has nichrome or polysilicon wires arranged in a matrix. These wires can be functionally viewed as fuses.
- It is programmed by a PROM programmer which selectively burns the fuses according to the bit pattern to be stored.

Erasable Programmable Read Only Memory (EPROM)

- Erasable Programmable Read Only Memory (EPROM) gives the flexibility to reprogram the same chip.
- EPROM stores the bit information by charging the floating gate of an FET.
- Bit information is stored by using an EPROM programmer, which applies high voltage to charge the floating gate.
- EPROM contains a quartz crystal window for erasing the stored information.

Electrically Erasable Programmable Read only Memory (EEPROM)

- The information contained in the EEPROM memory can be altered by using electrical signals at the register/byte level.
- They can be erased and reprogrammed in-circuit.
- These chips include a chip erase mode and in this mode they can be erased in a few milliseconds.
- It provides greater flexibility for system design.
- The only limitation is their capacity is limited (a few kilobytes) when compared with the standard ROM.

	FLASH □ FLASH memory is a variation of EEPROM technology – It combines the reprogram ability of EEPROM and the high capacity of standard ROMs. □ FLASH is the latest ROM technology. □ Most popular ROM technology used in today's embedded designs. □ FLASH memory is organised as sectors (blocks) or pages. □ FLASH memory stores information in an array of floating gate MOSFET transistors. Non-Volatile RAM (NVRAM) □ Non-volatile RAM is a random access memory with battery backup. □ It contains static RAM based memory and a minute battery for providing supply to the memory in the absence of external power supply.	
(c)	**Build** the classification of embedded system based on Generation and triggering. **Classification Based on Generation** 1. First Generation □ Early embedded systems were built around 8-bit microprocessors like 8085 and Z80 and 4-bit microcontrollers. □ Simple in hardware circuits with firmware developed in assembly code □ E.g.: Digital telephone keypads, stepper motor control units, etc. 2. Second Generation □ Embedded systems built around 16-bit microprocessors and 8-bit or 16-bit microcontrollers. □ Instruction set were much more complex and powerful than the first generation. □ Some of the second generation embedded systems contained embedded operating systems for their operation. □ E.g.: Data acquisition systems, SCADA systems, etc. 3. Third Generation □ Embedded systems built around 32-bit microprocessors and 16-bit microcontrollers. □ Application and domain specific processors/controllers like Digital Signal Processors (DSP) and Application Specific Integrated Circuits (ASICs) came into picture. □ The instruction set of processors became more complex and powerful and the concept of instruction pipelining also evolved. □ Dedicated embedded real time and general purpose operating systems entered into the embedded market. □ Embedded systems spread its ground to areas like robotics, media, industrial process control, networking, etc. 4. Fourth Generation □ The advent of System on Chips (SoC), reconfigurable processors and multicore processors are bringing high performance, tight integration	**3 marks for genera tion** **2 marks for trigger ing**

	and miniaturisation into the embedded device market. □ The SoC technique implements a total system on a chip by implementing different functionalities with a processor core on an integrated circuit. □ They make use of high performance real time embedded operating systems for their functioning. □ E.g.: Smart phone devices, Mobile Internet Devices (MIDs), etc. 5. Next Generation □ The processor and embedded market is highly dynamic and demanding. □ The next generation embedded systems are expected to meet growing demands in the market. Classification Based on Triggering □ Embedded systems which are 'Reactive' in nature (like process control systems in industrial control applications) can be classified based on the trigger. □ Reactive systems can be either event-triggered or time-triggered.	
4(a)	**Discover** how CMSIS provides standard access interface for embedded software. **Areas of standardization** □ **Hardware Abstraction Layer (HAL) for Cortex-M processor registers:** This includes standardized register definitions for NVIC, System Control Block registers, SYSTICK register, MPU registers, and a number of NVIC and core feature access functions. □ **Standardized system exception names:** This allows OS and middleware to use system exceptions easily without compatibility issues. □ **Standardized method of header file organization**: This makes it easier for users to learn new cortex microcontroller products and improve software portability. □ **Common method for system initialization**: Each Microcontroller Unit (MCU) vendor provides a SystemInit() function in their device driver library for essential setup and configuration, such as initialization of clocks. □ **Standardized intrinsic functions**: By having standardized intrinsic functions, software reusability and portability are considerably improved. □ **Common access functions for communication**: This provides a set of software interface functions for common communication interfaces including universal asynchronous receiver/transmitter (UART), Ethernet, and Serial Peripheral Interface (SPI). □ **Standardized way for embedded software to determine system clock frequency**: A software variable called System Frequency is defined in device driver code. This allows embedded OS to set up the	**1 mark for each**

	SYSTICK unit based on the system clock frequency.	
(b)	**Explain** ZigBee, Bluetooth communication interface. ZigBee is a low power, low cost, wireless network communication protocol based on the IEEE 802.15.4-2006 standard. □ ZigBee is targeted for low power, low data rate and secure applications for Wireless Personal Area Networking (WPAN). □ The ZigBee specifications support a robust mesh network containing multiple nodes. □ This networking strategy makes the network reliable by permitting messages to travel through a number of different paths to get from one node to another. □ ZigBee operates worldwide at the unlicensed bands of Radio spectrum, mainly at 2.400 to 2.484 GHz, 902 to 928 MHz and 868.0 to 868.6 MHz. □ ZigBee supports an operating distance of up to 100 metres and a data rate of 20 to 250 Kbps. □ In the ZigBee terminology, each ZigBee device falls under any one of the following ZigBee device category: □ ZigBee Coordinator (ZC)/Network Coordinator □ The ZigBee coordinator acts as the root of the ZigBee network. □ The ZC is responsible for initiating the ZigBee network and it has the capability to store information about the network. □ ZigBee Router (ZR)/Full function Device (FFD) □ Responsible for passing information from device to another device or to another ZR. □ ZigBee End Device (ZED)/Reduced Function Device (RFD): □ End device containing ZigBee functionality for data communication. **Bluetooth (BT)** □ Bluetooth is a low cost, low power, short range wireless technology for data and voice communication. □ Bluetooth was first proposed by Ericsson in 1994. Bluetooth operates at 2.4GHz of the Radio Frequency spectrum and uses the Frequency Hopping Spread Spectrum (FHSS) technique for communication. □ Bluetooth supports a data rate of up to 1Mbps to 24Mbps and a range of approximately 30 to 100 feet for data communication (depending on the version) □ v1.2 supports data rate up to 1Mbps □ v2.0 + EDR supports data rate up to 3Mbps □ v3.0 + HS and v4.0 supports data rate up to 24Mbps □ Bluetooth communication has two essential parts – a physical link part and a protocol part. □ The physical link is responsible for the physical transmission of data between devices supporting Bluetooth communication □ The protocol part is responsible for defining the rules of communication.	**2.5 marks for each**

	□ The physical link works on the wireless principle making use of RF waves for communication. □ Bluetooth enabled devices essentially contain a Bluetooth wireless radio for the transmission and reception of data. □ The rules governing the Bluetooth communication is implemented in the 'Bluetooth protocol stack'. □ The Bluetooth communication IC holds the stack. □ Each Bluetooth device will have a 48 bit unique identification number. □ Bluetooth communication follows packet based data transfer. □ Bluetooth supports point-to-point (device to device) and point-to-multipoint (device to multiple device broadcasting) wireless communication. □ The point-to-point communication follows the master-slave relationship. □ A Bluetooth device can function as either master or slave. □ When a network is formed with one Bluetooth device as master and more than one device as slaves, it is called a Piconet. □ A Piconet supports a maximum of seven slave devices. □ Bluetooth is the favourite choice for short range data communication in handheld embedded devices. □ Bluetooth technology is very popular among cell phone users as they are the easiest communication channel for transferring ringtones, music files.	
(c)	**Construct** the application areas of Embedded system. 1. Consumer electronics: Camcorders, cameras, etc. 2. Household appliances: Television, DVD players, washing machine, refrigerators, microwave oven, etc. 3. Home automation and security systems: Air conditioners, sprinklers, intruder detection alarms, closed circuit television (CCTV) cameras, fire alarms, etc. 4. Automotive industry: Anti-lock braking systems (ABS), engine control, ignition systems, automatic navigation systems, etc. 5. Telecom: Cellular telephones, telephone switches, handset multimedia applications, etc. 6. Computer peripherals: Printers, scanners, fax machines, etc. 7. Computer networking systems: Network routers, switches, hubs, firewalls, etc. 8. Healthcare: Different kinds of scanners, EEG, ECG machines, etc.	**1 mark for each**

	9. Measurements & Instrumentation: Digital multimeters, digital CROs, logic analyzers, PLC systems, etc. 10. Banking & Retail: Automated teller machines (ATM) and currency counters, point of sales (POS), etc. 11. Card readers: Barcode, smart card readers, hand held devices, etc. 12. Wearable Devices: Health and fitness trackers, Smartphone screen extension for notifications, etc. 13. Cloud Computing and Internet of Things (IoT)	

Course Incharge

HOD ECE

Professor & Head
Dept. of Electronics & Communication Engineering
K. S. School of Engineering & Management
Bangalore-560 109

Principal

Principal / Director
K.S. School of Engineering & Management
Bangalore-560 062

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BANGALORE - 560109
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
SESSION: 2021-2022 (EVEN SEMESTER)
III SESSIONAL TEST QUESTION PAPER
SET-B

USN | | | | | | | | | | |

Degree	:	B.E	Semester	:	VI A & B
Branch	:	Electronics and Communication Engineering	Date	:	14/07/2022
Course Title	:	Embedded Systems	Course Code	:	18EC62
Duration	:	90 Minutes	Max Marks	:	30

Note: Answer ONE full question from each part.

Q No.	Question	Marks	K-Level	CO mapping
	PART-A			
1(a)	**Construct** super loop based approach for Embedded firmware design.	5	Applying (K3)	CO4
(b)	**Explain** different types of serial interface bus used in automotive communication.	5	Understanding (K2)	CO4
(c)	**Explain** how operating systems are classified.	5	Understanding (K2)	CO5
	OR			
2(a)	**Make use of** block diagram to explain the working of washing machine.	5	Applying (K3)	CO4
(b)	**Explain** the different characteristics of Embedded system.	5	Understanding (K2)	CO4
(c)	**Differentiate** between Hard Real time system and Soft Real time system with an example for each.	5	Understanding (K2)	CO5
	PART-B			
3(a)	**Construct** fundamental design approaches in hardware and software co-design.	5	Applying (K3)	CO4
(b)	**Explain** operational quality attributes of Embedded system.	5	Understanding (K2)	CO4
(c)	Three processes with process IDs P1, P2, P3 with estimated completion time 10, 5, 7 milliseconds respectively enter the ready queue together. A new process P4 with estimated completion time 2 ms enters the 'Ready' queue after 2 ms. Assume all the processes contain only CPU operation and no I/O operations are involved. **Calculate** the waiting time and Turn Around Time (TAT) for each process and the average waiting time and Turn Around Time in the SRT scheduling.	5	Applying (K3)	CO5
	OR			
4(a)	**Build** a coin operated public telephone unit based on FSM model.	5	Applying (K3)	CO4
(b)	List all computational models and **Explain** DFG & concurrent process model.	5	Understanding (K2)	CO4
(c)	**Make use of** neat diagram to explain operating system architecture.	5	Applying (K3)	CO5

Course Incharge | HOD ECE | IQAC- Coordinator | Principal

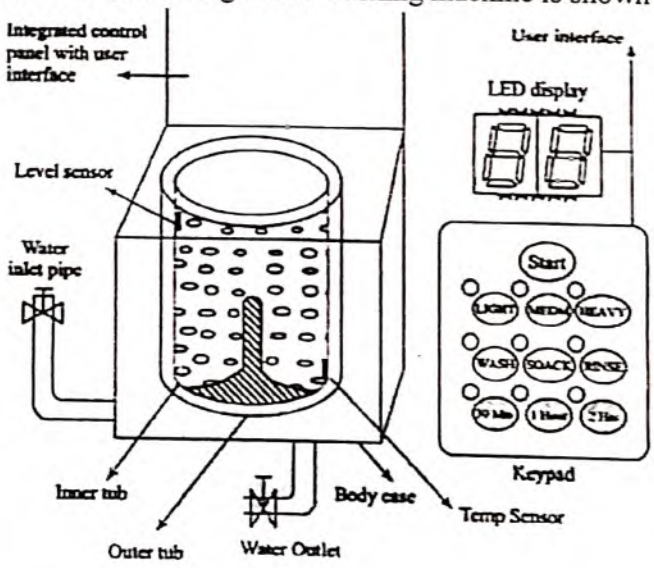
Professor & Head
Dept. of Electronics & Communication Engineering
K. S. School of Engineering & Management
Bangalore-560 109

Dr. K. RAMA NARASIMHA
Principal/Director
K S School of Engineering and Management
Bengaluru - 560 109

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BENGALURU-560109
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
III SESSION: 2021-2022 (EVEN SEMESTER)
SCHEME OF VALUATION
SET-A

Degree	:	B.E	Semester	:	VI A & B
Branch	:	Electronics & Communication Engg.	Date	:	14/07/22
Course Title	:	Embedded Systems	Course Code	:	18EC62
Duration	:	90 Minutes	Max Marks	:	30

Note: Answer ONE full question from each part

Q. No.	Scheme and Solution	Marks
1(a)	Make use of block diagram to explain the working of washing machine. The actuator part of washing machine consists of motorised agitator, tumbler tub, water drawing pump and inlet valve to control the flow of water into the unit. The sensor part consists of the water temperature sensor, level sensor etc. The control part consists of microcontroller or microprocessor controlled based board with interfaces to the sensors and actuators. The sensor data is fed back to the control unit and the control unit also provides connectivity to user interfaces like keypad for setting washing machine time, selecting the type of material to be washed like light, medium, heavy etc. User feedback is reflected through the display unit and LEDs connected to the control board. The functional block diagram of washing machine is shown below:  ☐ The mechanism includes the motor, transmission, clutch, pump, agitator, inner tub, outer tub and water inlet valve.	2 marks for diagram 3 marks for explanation

	□ Water inlet valve connects to the water supply line using at home and regulates the flow of water into the tub. □ The integrated control panel consists of a microprocessor/controller based board with I/O interfaces and a control algorithm running in it. □ Input interface includes the keyboard which consists of wash type selector namely Wash, Spin and Rinse, cloth type selector namely Light, Medium, Heavy duty and washing time setting, etc. □ The output interface consists of LED/LCD displays, status indication LEDs, etc. connected to the I/O bus of the controller. □ The other types of I/O interfaces which are invisible to the end user are different kinds of sensor interfaces, namely, water temperature sensor, water level sensor, etc. and actuator interface including motor control for agitator and tub movement control, inlet water flow control, etc.	
(b)	**Explain** the different characteristics of Embedded system. Some of the important characteristics are: 1. Application and domain specific 2. Reactive and real time 3. Operates in harsh environments 4. Distributed systems 5. Small size and weight 6. Power concerns 1. **Application and Domain specific** □ Each embedded system has certain functions to perform and they are developed in such a manner to do the intended functions only. □ They cannot be used for any other purpose. □ This is the major criterion which distinguishes an embedded system from a general purpose system. □ For example, the embedded control unit of a microwave oven cannot be replaced with an air conditioners embedded control unit, because the embedded control units of microwave oven and air conditioner are specifically designed to perform certain specific tasks. □ Also an embedded control unit developed for a particular domain, say telecom, cannot be replaced with another control unit designed to serve another domain like consumer electronics. 2. **Reactive and Real time** □ Embedded systems are in constant interaction with the real world through sensors and user-defined input devices which are connected to the input port of the system. □ Any changes happening in the real world (which is called an Event) are captured by the sensors or input devices in Real Time and the control algorithm running inside the unit reacts in a designed manner to	**1 mark for each**

bring the controlled output variables to the desired level.

- The event may be periodic one or unpredicted one.
- If an event is unpredicted one then such system should be designed in such a way that it should be scheduled to capture the events without missing them.
- Embedded systems produce changes in output in response to the changes in the output and they are generally referred as **Reactive systems.**
- Real Time System operation means the timing behaviour of the system should be deterministic.
- The system should respond to requests or tasks in a known amount of time.
- A Real Time system should not miss any deadlines for tasks or operations.
- It is not necessary that all embedded systems should be Real Time in operations.
- Embedded applications or systems which are mission critical, like flight control systems, Antilock Brake Systems (ABS), etc. are examples of Real Time systems.

3. **Operation in harsh environment**

- Certain embedded systems are designed to operate in harsh environments may be a dusty or a high temperature zone or a area subject to vibrations and shock.
- Systems placed in such areas should be capable to with stand all these adverse operating conditions.
- The design should take care of the operating conditions of the area where the system is going to implement.
- For example, if the system needs to be deployed in a high temperature zone, then all the components used in the system should be of high temperature grade.
- Also proper shock absorption techniques should be provided to systems which are going to be commissioned in places subject to high shock.

4. **Distributed systems**

- The term ***distributed*** means that embedded systems may be a part of larger systems.
- Many numbers of such distributed embedded systems form a single large embedded control unit.

For example,

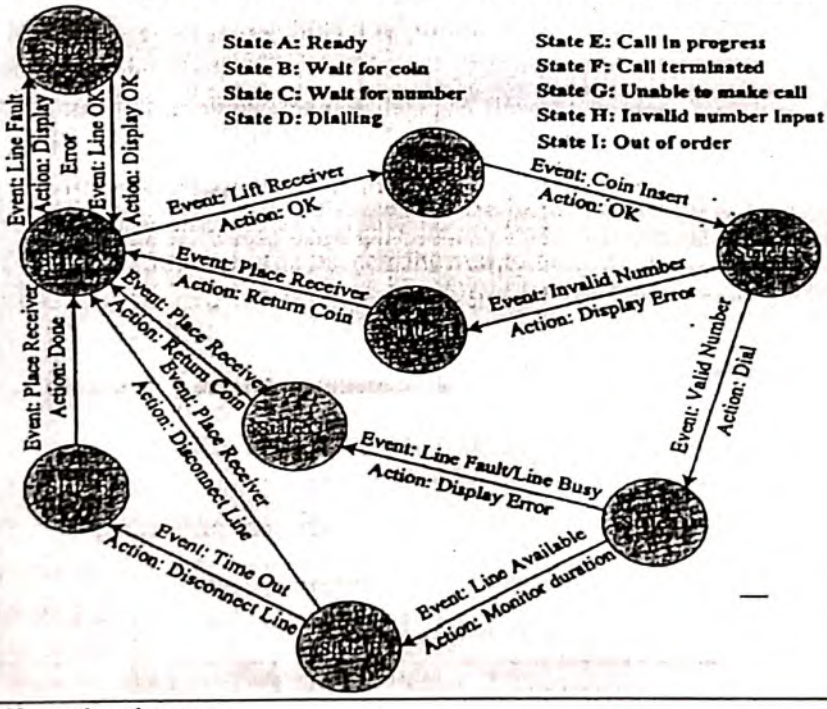
- An automatic vending machine. It contains a card reader (for pre-paid vending systems), a vending unit, etc.
- Each of them are independent embedded units but they work together

	to perform the overall vending function. Another example is the Automated Teller Machine (ATM). **5. Small size and weight** □ An embedded system that is compact in size and has light weight will be desirable or more popular than one that is bulky and heavy. □ The product aesthetics (size, weight, shape, style etc) will be one of the deciding factors to choose a product. □ Ex. Currently available cell phones. The cell phones that have the maximum features are popular but also their size and weight is an important characteristic. **6. Power concerns** □ Power management is another important factor that needs to be considered in designing embedded systems. □ Embedded systems should be designed in such a way as to minimise the heat dissipation by the system. □ The production of high amount of heat demands cooling requirements like cooling fans which in turn occupies additional space and make the system bulky. □ Select the design according to the low power components like low dropout regulators, and controllers/processors with power saving modes. □ Also power management is a critical constraint in battery operated application. □ The more the power consumption the less is the battery life.	
(c)	**Differentiate** between Hard Real time system and Soft Real time system with an example for each. **Hard Real-Time** •Real-Time Operating Systems that strictly adhere to the timing constraints for a task are referred as *'Hard Real-Time'* systems. •They must meet the deadlines for a task without any slippage. •Missing any deadline may produce catastrophic results for Hard Real-Time Systems, including permanent data loss and irrecoverable damages to the system/users. •Hard Real-Time systems emphasise the principle '*A late answer is a wrong answer*'. •Air bag control systems and Anti-lock Brake Systems (ABS) of vehicles are typical examples for Hard Real-Time Systems. •Any delay in the deployment of the air bags makes the life of the passengers under threat. Hard Real-Time Systems does not implement the virtual memory model for handling the memory. •This eliminates the delay in swapping in and out the code corresponding to the task to and from the primary memory. •Most of the Hard Real-Time Systems are automatic and does not contain a *Human in the Loop (HITL)*.	**2.5 marks for each**

	•The presence of *human in the loop* for tasks introduces unexpected delays in the task execution. Soft Real-Time Real-Time Operating Systems that do not guarantee meeting deadlines, but offer the best effort to meet the deadline are referred as *'Soft Real-Time'*systems. •Missing deadlines for tasks are acceptable for a Soft Real-time system if the frequency of deadline missing is within the compliance limit of the Quality of Service (QoS). •A Soft Real-Time system emphasises the principle *'A late answer is an acceptable answer, but it could have done bit faster'.* •Soft Real-Time systems most often have a *human in the loop (HITL).* •Automated Teller Machine (ATM) is a typical example for Soft-Real-Time System. •If the ATM takes a few seconds more than the ideal operation time, nothing fatal happens. •An audio-video playback system is another example for Soft Real-Time system. •No potential damage arises if a sample comes late by fraction of a second, for playback.	
2(a)	List all computational models and **Construct** CDFG & sequential program model. The commonly used computational models in embedded system design are: □ Data Flow Graph Model □ Control Data Flow Graph Model □ State Machine Model □ Sequential Program Model □ Concurrent/Communicating Process Model □ Object-Oriented Model **Control Data Flow Graph/Diagram (CDFG) Model** □ The DFG model is a data driven model in which the execution is controlled by data and it doesn't involve any control operations (conditionals). □ The Control DFG (CDFG) model is used for modelling applications involving conditional program execution. □ CDFG models contains both data operations and control operations. The CDFG uses Data Flow Graph (DFG) as element and conditional (constructs) as decision makers. □ CDFG contains both data flow nodes and decision nodes, whereas DFG contains only data flow nodes. □ Consider the implementation of the CDFG for the following requirement. □ This requirement contains a decision making process. □ The CDFG model for the same is given in the figure. □ The control node is represented by a 'Diamond' block which is the decision making element in a normal flow chart based design. □ CDFG translates the requirement, which is modelled to a concurrent	**1 mark for listing** **1.5 mark for CDFG** **2.5 marks for sequential model**

(b)	**Explain** operational quality attributes of Embedded system. **Operational Quality Attributes** The operational quality attributes represent the relevant quality attributes related to the embedded system when it is in the operational mode or 'online' mode. The important operational quality attributes are: 1. Response 2. Throughput 3. Reliability 4. Maintainability 5. Security 6. Safety 1. Response: □ Response is a measure of quickness of the system. □ It gives you an idea about how fast your system is tracking the input variables. □ Most of the embedded system demand fast response which should be real-time. 2. Throughput □ Throughput deals with the efficiency of system. □ It can be defined as rate of production or process of a defined process over a stated period of time. □ The rates can be expressed in terms of units of products, batches produced or any other meaningful measurements. 3. Reliability □ Reliability is a measure of how much percentage you rely upon the proper functioning of the system or what is the % susceptibility of the system failures. □ Mean Time between failures (MTBF) and Mean Time To Repair (MTTR) are terms used in defining system reliability. □ MTBF gives the frequency in hours/weeks/months. □ Mean Time between failures can be defined as the average time the system is functioning before a failure occurs. 4. Maintainability □ Maintainability deals with support and maintenance to the end user or a client in case of technical issues and product failures or on the basis of a routine system check-up □ Reliability and maintainability are considered as two complementary disciplines. □ It can be classified into two types :- **a) Scheduled or Periodic Maintenance (Preventive maintenance)** This is the maintenance that is required regularly after a periodic time	**1 mark for each**

	interval. Example: Periodic Cleaning of Air Conditioners Refilling of printer cartridges. b) **Maintenance to unexpected failure** This involves the maintenance due to a sudden breakdown in the functioning of the system. 5. Security □ Confidentiality, Integrity and Availability are three major measures of information security. □ Confidentiality deals with protection data from unauthorized disclosure. 6. **Safety** □ Safety deals with the possible damage that can happen to the operating person and environment due to the breakdown of an embedded system or due to the emission of hazardous materials from the embedded products. □ The breakdown of an embedded system may occur due to a hardware failure or a firmware failure.	
(c)	**Explain** how operating systems are classified. Depending on the type of kernel and kernel services, purpose and type of computing systems where the OS is deployed and the responsiveness to applications, Operating Systems are classified into different types. □ General Purpose Operating System (GPOS) □ Real-Time Operating System (RTOS) General Purpose Operating System (GPOS) • The operating systems which are deployed in general computing systems are referred as General Purpose Operating Systems (GPOS). • The kernel of such a GPOS is more generalised and it contains all kinds of services required for executing generic applications. • General purpose operating systems are often quite non-deterministic in behaviour. • Their services can inject random delays into application software and may cause slow responsiveness of an application at unexpected times. • GPOS are usually deployed in computing systems where deterministic behaviour is not an important criterion. • Personal Computer/Desktop system is a typical example for a system where GPOSs are deployed. • Windows XP/MS-DOS etc. are examples for General Purpose Operating Systems. Real-Time Operating System (RTOS) 'Real-Time' implies deterministic timing behaviour. • Deterministic timing behaviour in RTOS context means the OS services consumes only known and expected amounts of time regardless the number of services. • A Real-Time Operating System or RTOS implements policies and	**2.5 marks for each**

	rules concerning time-critical allocation of a system's resources. • The RTOS decides which applications should run in which order and how much time needs to be allocated for each application. • Predictable performance is the hallmark of a well-designed RTOS. • This is best achieved by the consistent application of policies and rules. • Policies guide the design of an RTOS. • Rules implement those policies and resolve policy conflicts. • Windows CE, QNX, VxWorks, MicroC/OS-II, etc. are examples of Real-Time Operating Systems (RTOS).	
3(a)	**Build** a coin operated public telephone unit based on FSM model. • The calling process is initiated by lifting the receiver (off-hook) of the telephone unit. • After lifting the phone the user needs to insert a 1 rupee coin to make the call. • If the line is busy, the coin is returned on placing the receiver back on the hook (on-hook). • If the line is through, the user is allowed to talk till 60 seconds and at the end of 45th second, prompt for inserting another 1 rupee coin for continuing the call is initiated. • If the user doesn't insert another 1 rupee coin, the call is terminated on completing the 60 seconds time slot. • The system is ready to accept new call request when the receiver is placed back on the hook (on-hook). • The system goes to the 'Out of Order' state when there is a line fault.  	1 mark for explanation 4 marks model
(b)	Differentiate between 1. Compiler vs Cross compiler	

2. C vs Embedded C

'C' vs. 'Embedded C'

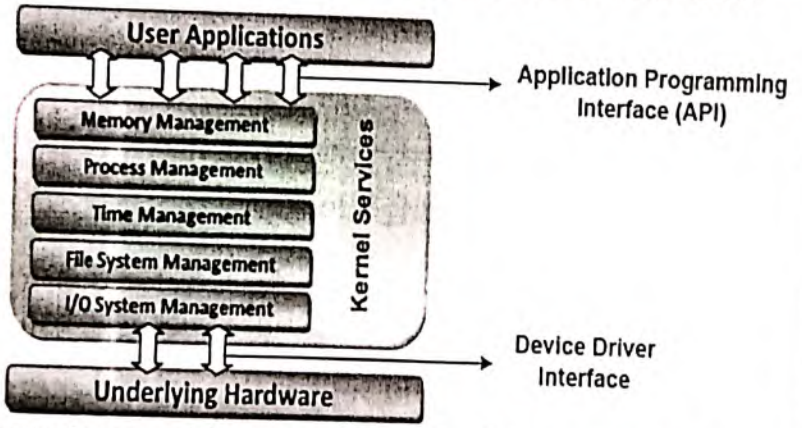
- 'C' is a well structured, well defined and standardised general purpose programming language with extensive bit manipulation support.
- 'C' offers a combination of the features of high level language and assembly and helps in hardware access programming (system level programming) as well as business package developments (Application developments like pay roll systems, banking applications, etc).
- The conventional 'C' language follows ANSI standard and it incorporates various library files for different operating systems.
- A platform (operating system) specific application, known as, *compiler* is used for the conversion of programs written in 'C' to the target processor (on which the OS is running) specific binary files.
  - Hence it is a platform specific development.
- Embedded 'C' can be considered as a subset of conventional 'C' language.
- Embedded 'C' supports all 'C' instructions and incorporates a few target processor specific functions/instructions.
- The standard ANSI 'C' library implementation is always tailored to the target processor/controller library files in Embedded 'C'.
- The implementation of target processor/controller specific functions/instructions depends upon the processor/controller as well as the supported cross-compiler for the particular Embedded 'C' language.
- A software program called 'Cross-compiler' is used for the conversion of programs written in Embedded 'C' to target processor/controller specific instructions (machine language).

Compiler vs. Cross-Compiler

- Compiler is a software tool that converts a source code written in a high level language on top of a particular operating system running on a specific target processor architecture (e.g. Intel x86/Pentium).
- Here the operating system, the compiler program and the application making use of the source code run on the same target processor.
- The source code is converted to the target processor specific machine instructions.
- The development is platform specific (OS as well as target processor on which the OS is running).
- Compilers are generally termed as 'Native Compilers'.
  - A native compiler generates machine code for the same machine (processor) on which it is running.
- Cross-compilers are the software tools used in cross-platform development applications.
  - In cross-platform development, the compiler running on a particular target processor/OS converts the source code to machine code for a target processor whose architecture and instruction set is different from the processor on which the compiler is running or for an operating system which is different from the current development environment OS.
- Embedded system development is a typical example for cross-platform development.
  - Embedded firmware is developed on a machine with Intel/AMD or any other target processors and the same is converted into machine code for any other target processor architecture (e.g. 8051, PIC, ARM etc).
- Keil C51 is an example for cross-compiler.
- In embedded firmware application, whenever we use the term 'Compiler' it normally refers to the cross-compiler.

2.5 marks for each

(c) **Make use of** neat diagram to explain operating system architecture.



2 marks for diagram

3 marks for explanation

For a general purpose OS, the kernel contains different services for handling the following:

Process management:

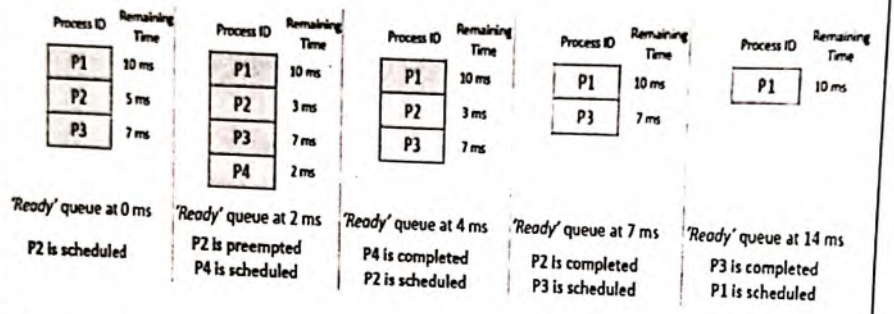
- It deals with managing processes/tasks.
- It includes setting up the memory space for the process, loading the process's code into the memory space, allocating system resources, scheduling and managing the execution of the process, setting up and

	managing the process control block (PCB), inter process communication and synchronization, process termination/deletion etc. **Primary memory management** □ The term primary memory refers to the volatile memory (RAM) where processes are loaded and variables and shared data associated with each process are stored. **File system management** □ A file is a collection of related information. □ A file could be a program (source code or executable), text files, image files, word documents, audio/video files, etc. □ Each of these files differs in the kind of information they hold and the way in which the information stored. **I/O system (device) management** Kernel is responsible for routing I/O requests coming from different user applications to the appropriate I/O devices of the system. In a well structured OS, the direct accessing of I/O devices are not allowed and the access to them are provided through a set of Application Programming interfaces (APIs) exposed by the kernel. **Secondary storage Management** □ The secondary storage management deals with managing the secondary storage memory devices, if any, connected to the system. Secondary memory is used as backup medium for programs and data since the main memory is volatile. **Protection Systems** □ Most of the modern operating systems are designed in such a way to support multiple users with different levels of access permissions (eg. Windows XP with user permissions like 'Administrator', 'standard', 'restricted' etc). □ Protection deals with implementing the security policies to restrict the access to both user and system resources by different applications or processes or users. **Interrupt Handler** □ Kernel provides handler mechanism for all external/internal interrupts generated by the system. □ These are some of the important services provided by the kernel of operating systems.	
4(a)	**Construct** fundamental design approaches in hardware and software co-design. The firmware design approaches for embedded product is purely dependent on the complexity of the functions to be performed, the speed of operation required, etc.	

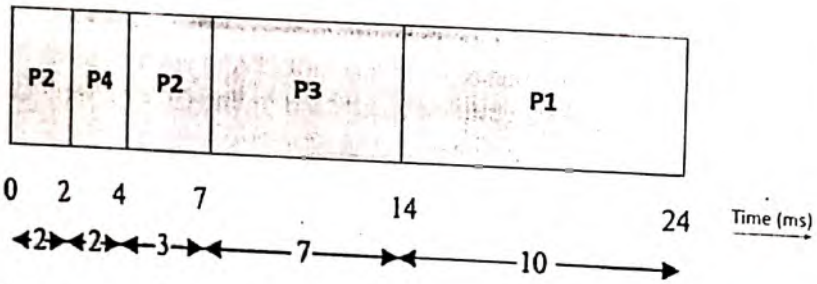
	Two basic approaches are used for embedded firmware design: • Super Loop Based Approach (Conventional Procedural Based Design) • Embedded Operating System (OS) Based Approach Super Loop Based Approach □ The Super Loop based firmware development approach is adopted for applications that are not time critical and where the response time is not so important. □ It is very similar to a conventional procedural programming where the code is executed task by task. □ The task listed at the top of the program code is executed first and the tasks just below the top are executed after completing the first task. □ In a multiple task based system, each task is executed in serial in this approach. A typical example of a 'Super loop based' product is an electronic video game toy containing keypad and display unit. • The program running inside the product may be designed in such a way that it reads the keys to detect whether the user has given any input and if any key press is detected the graphic display is updated. • The keyboard scanning and display updating happens at a reasonably high rate. **Embedded Operating System (OS) Based Approach** The Embedded Operating System (OS) based approach contains operating systems, which can be either a General Purpose Operating System (GPOS) or a Real Time Operating System (RTOS) to host the user written application firmware. The General Purpose OS (GPOS) based design is very similar to a conventional PC based application development where the device contains an operating system (Windows/Unix/Linux, etc. for Desktop PCs) and you will be creating and running user applications on top of it. • Example of a GPOS used in embedded product development is Microsoft Windows XP Embedded.	**2.5 marks for each**
(b)	**Explain** with a block diagram, how source file is converted into object file. □ Translation of assembly code to machine code is performed by assembler. • The assemblers for different target machines are different. • A51 Macro Assembler from Keil software is a popular assembler for the 8051 family microcontroller. □ The various steps involved in the conversion of a program written in assembly language to corresponding binary file/machine language are illustrated in the figure.	

(c) Three processes with process IDs P1, P2, P3 with estimated completion time 10, 5, 7 milliseconds respectively enter the ready queue together. A new process P4 with estimated completion time 2 ms enters the 'Ready' queue after 2 ms. Assume all the processes contain only CPU operation and no I/O operations are involved. **Calculate** the waiting time and Turn Around Time (TAT) for each process and the average waiting time and Turn Around Time in the SRT scheduling.

5 marks for solving



The execution sequence can be written as below:



The waiting time for all the processes are given as

- $Waiting\ time\ for\ P2 = 0\ ms + (4 - 2)\ ms = 2\ ms$
- $Waiting\ time\ for\ P4 = 0\ ms$
- $Waiting\ time\ for\ P3 = 7\ ms$
- $Waiting\ time\ for\ P1 = 14\ ms$

$$Average\ Waiting\ time = \frac{Waiting\ time\ for\ all\ the\ processes}{Number\ of\ processes}$$

$$= \frac{2+0+7+14}{4}\ ms = \frac{23}{4} ms$$

$$= 5.75\ ms$$

$Turn\ Around\ Time\ (TAT) = Time\ spent\ in\ ready\ queue\ + Execution\ Time$

- $Turn\ Around\ Time\ (TAT)\ for\ P2 = 2\ ms + 5\ ms = 7\ ms$
- $Turn\ Around\ Time\ (TAT)\ for\ P4 = 0\ ms + 2\ ms = 2\ ms$
- $Turn\ Around\ Time\ (TAT)\ for\ P3 = 7\ ms + 7\ ms = 14\ ms$
- $Turn\ Around\ Time\ (TAT)\ for\ P1 = 14\ ms + 10\ ms = 24\ ms$

$$Average\ Turn\ Around\ Time\ (TAT) = \frac{TAT\ for\ all\ the\ processes}{Number\ of\ processes}$$

$$= \frac{7+2+14+24}{4}\ ms = \frac{47}{4}ms$$

$$= 11.75\ ms$$

Course Incharge

HOD ECE

Professor & Head
Dept. of Electronics & Communication Engineering
K. S. School of Engineering & Management
Bangalore-560 109

Principal

Principal/Director
K. S. School Of Engineering & Ma
BENGALURU-560

K.S. GROUP OF INSTITUTIONS

K.S. SCHOOL OF ENGINEERING & MANAGEMENT

15, Mallasandra, Near Vajarahalli, Off. Kanakapura Road, Bengaluru- 560 109
www.kssem.edu.in

KSSEM
K S SCHOOL OF ENGINEERING AND MANAGEMENT

BLUE BOOK

Name of the Student: Kavya. C

Class / Sem : 6th sem, 'B' sec **Branch:** E.C.E

USN : | 1 | K | G | 1 | 9 | E | C | 0 | 4 | 9 |

SUBJECT : Embedded Systems **Subject Code :** 18EC62

MAXIMUM MARKS :

Test	I	II	III	Average Marks Obtained
Date	16/5/22	16/6/22	14/7/22	Avg = 24+10 = 34
Marks Obtained	22 Good	25	26	
Signature of the Student	Kavya.C	Kavya.C	Kavya.C	
Initials of Room Supervisor	Bhargavi	P		Sangam
Initials of Faculty	Bhargavi	Bhargavi	Bhargavi	Bhargavi

Verified 28

NAME OF FACULTY : Bhargavi Sangam

SIGNATURE : Bhargavi

SIGNATURE OF H.O.D.

K S SCHOOL OF ENGINEERING AND MANAGEMENT

First Internal test

Q. No	Marks	CO	Q. No	Marks	CO	CO	Total
1(a)	05	CO1	3(a)	05	CO1	CO1	16
1(b)	03	CO1	3(b)	03	CO1		
1(c)	02	CO2	3(c)	04	CO2	CO2	6
OR			OR				
2(a)			4(a)	04	CO1		
2(b)			4(b)		CO1		
2(c)			4(c)	02	CO2	Grand Total	22

Second Internal test

Q. No	Marks	CO	Q. No	Marks	CO	CO	Total
1(a)	05	CO2	3(a)	04	CO2	CO2	09
1(b)	05	CO3	3(b)	02	CO2		
1(c)	04	CO3	3(c)	05	CO3	CO3	16
OR			OR				
2(a)			4(a)				
2(b)			4(b)				
2(c)			4(c)			Grand Total	25

Third Internal test

Q. No	Marks	CO	Q. No	Marks	CO	CO	Total
1(a)	04	CO4	3(a)			CO4	09
1(b)	05	CO4	3(b)				
1(c)	04	CO5	3(c)	04	CO5	CO5	17
OR			OR				
2(a)			4(a)	05	CO4		
2(b)			4(b)	04	CO4		
2(c)			4(c)	04	CO5	Grand Total	26

Signature of the Staff

K.S. GROUP OF INSTITUTIONS

K.S. SCHOOL OF ENGINEERING & MANAGEMENT

15, Mallasandra, Near Vajarahalli, Off. Kanakapura Road, Bengaluru- 560 109
www.kssem.edu.in

KSSEM
K S SCHOOL OF ENGINEERING AND MANAGEMENT

BLUE BOOK

Name of the Student: KOVI.SHARATHCHANDRA

Class / Sem : VI-B. **Branch:** ECE

USN : 1 K G 1 9 E C O S 2.

SUBJECT : Embedded systems **Subject Code :** 18EC62

MAXIMUM MARKS :

Test	I	II	III	Average Marks Obtained
Date	16/5/22.	16/6/22	14/6/22	
Marks Obtained	17	21	21	Avg = 20+9 = 29
Signature of the Student				
Initials of Room Supervisor				
Initials of Faculty				

Verified

NAME OF FACULTY : Bhagyashri Sangam

SIGNATURE :

SIGNATURE OF H.O.D.

K S SCHOOL OF ENGINEERING AND MANAGEMENT

First Internal test

Q. No	Marks	CO	Q. No	Marks	CO	CO	Total
1(a)	05	CO1	3(a)	05	CO1	CO1	14
1(b)	04	CO1	3(b)				
1(c)	02	CO2	3(c)	01	CO2	CO2	03
OR			OR				
2(a)			4(a)				
2(b)			4(b)				
2(c)			4(c)			Grand Total	17

Second Internal test

Q. No	Marks	CO	Q. No	Marks	CO	CO	Total
1(a)	04	CO2	3(a)	05	CO2	CO2	06+3=09
1(b)	04	CO3	3(b)	05	CO3		
1(c)			3(c)	03	CO3	CO3	12
OR			OR				
2(a)			4(a)				
2(b)			4(b)				
2(c)			4(c)			Grand Total	18+3=21

Third Internal test

Q. No	Marks	CO	Q. No	Marks	CO	CO	Total
1(a)	04	CO4	3(a)			CO4	13
1(b)	05	CO4	3(b)				
1(c)	05	CO5	3(c)			CO5	08
OR			OR				
2(a)			4(a)	04	CO4		
2(b)			4(b)				
2(c)			4(c)	03	CO5	Grand Total	21

Signature of the Staff