CBCS SCHEME

USN □□□□□□□□□□ 15EC62

Sixth Semester B.E. Degree Examination, June/July 2019
ARM Micro Controller and Embedded Systems

Time: 3 hrs. Max. Marks: 80

Note: *Answer any FIVE full questions, choosing ONE full question from each module.*

Module-1

1 a. Explain the architecture of ARM cortex – M3 processor with neat diagram. (08 Marks)
b. With neat diagram, explain operation mode and privilege levels in cortex M3. (08 Marks)

OR

2 a. What is stack? Explain push and pop operation. With the help of a neat diagram. (07 Marks)
b. Explain in detail special registers used in ARM cortex M3 processor. (09 Marks)

Module-2

3 a. Write an ALP to calculate the sum of 1 to 10 numbers. (08 Marks)
b. Explain the following instruction set : i) BFC ii) SBFX iii)ASR iv) MRS. (04 Marks)
c. Explain how CMSIS provides standard access, Interface for Embedded software. (04 Marks)

OR

4 a. Write a program to blink a LED using 'C' language. (08 Marks)
b. Explain the following assembler directives AREA, ENTRY, DCB, ALIGN. (04 Marks)
c. Explain different bus interfaces supported by cortex M3. (04 Marks)

Module-3

5 a. Explain how embedded system are classified. (08 marks)
b. With neat block diagram, explain the element of embedded system. (08 Marks)

OR

6 a. Differentiate between RISC and CISC. (04 Marks)
b. Explain how program memory are classified. (08 Marks)
c. Explain how brown-out protection circuits works. (04 Marks)

Module-4

7 a. What are the operational and nonoperational attributes of an embedded systems. (10 Marks)
b. Explain different types of serial interface bus used in automotive communication. (06 Marks)

OR

8 a. Explain fundamental issues in hardware software co-design. (06 Marks)
b. Explain with a neat block diagram how source file to object file translation take place. (06 marks)
c. Explain super loop based approach of embedded firmware design. (04 Marks)

Important Note : 1. On completing your answers, compulsorily draw diagonal cross lines on the remaining blank pages.
2. Any revealing of identification, appeal to evaluator and /or equations written eg, 42+8 = 50, will be treated as malpractice.

Scheme & Solutions Prepared by

Prof. Nikhil. Kulkarni

Head of the Department
Dept. of Electronic & Communication Engg.
K.L.S VDIT, HALIYAL (U.K.)

Module-5

9 a. With neat diagram explain operating system architecture. (08 marks)

b. Explain how operating systems are classified. (04 marks)

c. Differentiate between hard real time system and soft real time system with an example for each. (04 Marks)

OR

10 a. With neat diagram, explain embedded system development environment (08 marks)

b. For the following jobs calculate the turnaround time, waiting time using preemptive SJF scheduling algorithm (04 Marks)

Jobs	CPU bust time	Arrival time
1	10	0.0
2	2	3.0
3	1	4.0
4	4	5.0

c. Write a note on IAP [In Application Programming] and in system programming. (04 Marks)

*****

Q1a] Explain the architecture of ARM cortex-M3 processor with neat diagram 8M

Soln: Here Block diagram is awarded with 4to5M

Explanation of architecture is awarded in the range of 4to5M

Brief about the architecture

i) It follows Harvard architecture

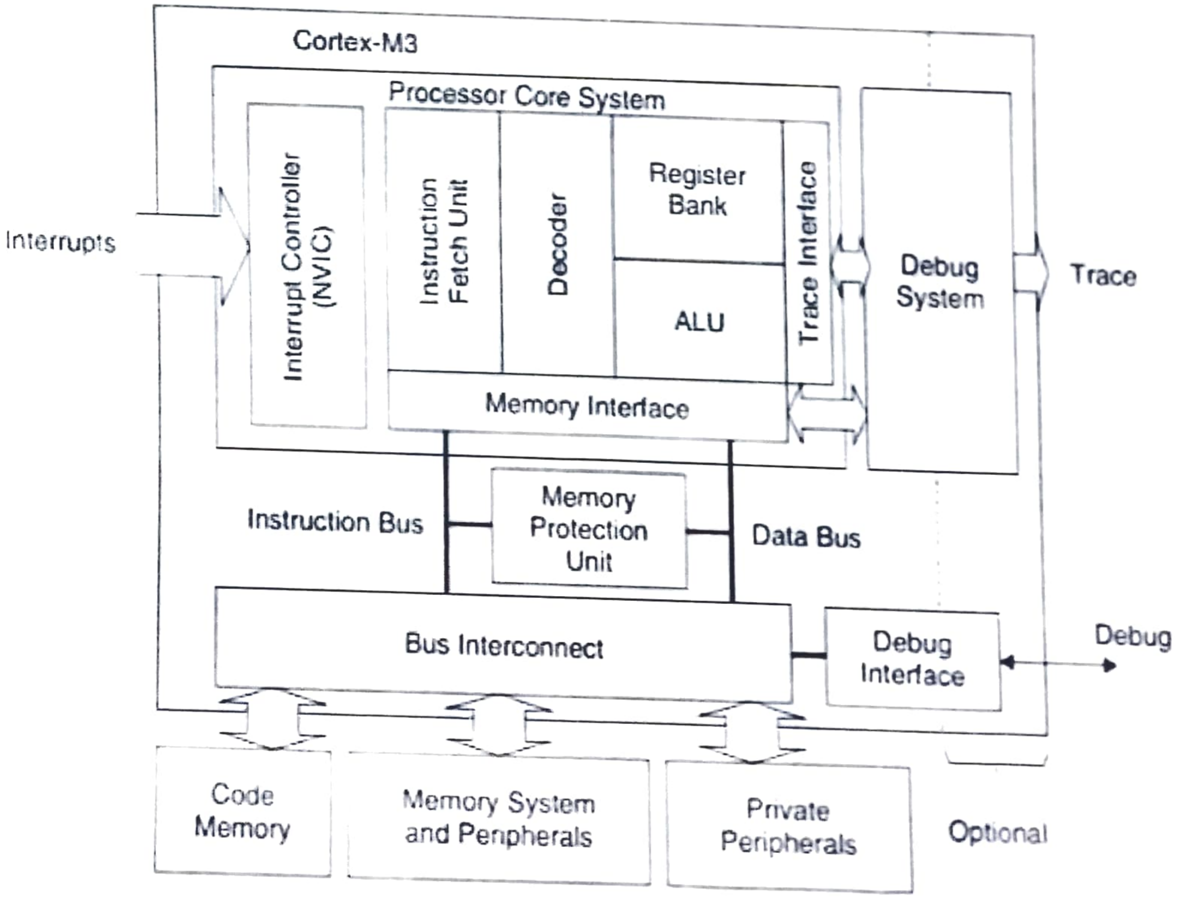
ii) It is a 32-bit up

iii) Registers: It has 13, 32-bit GPR's.

iv) MPU: Memory protection unit, used for data management

v) NVIC: used to configure up to 240 ext. interrupts

vi) Bus interconnects: are used to provide communication b/w processor and int/ext peripherals.



Q.no] 1b With neat-diagram explain the operation mode and privilage levels in cortex m3.

8 Marks

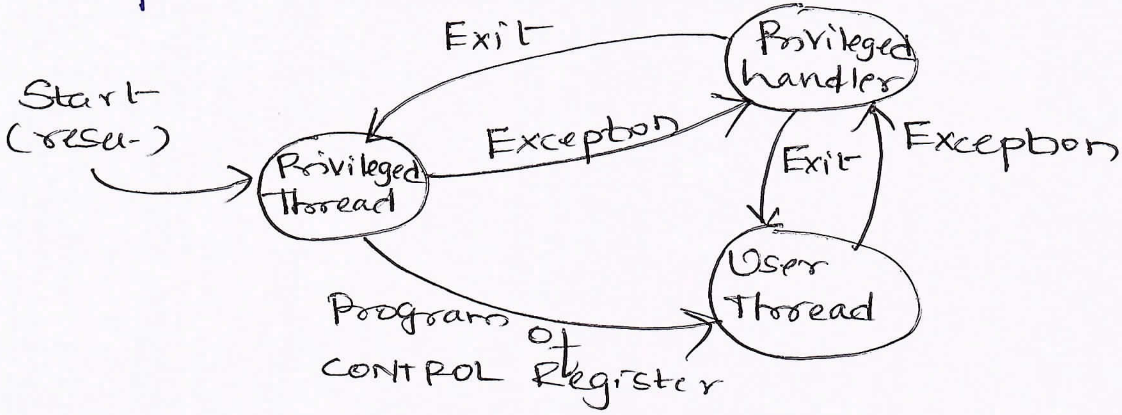
diagram: 4 to 6 marks
explanation: 2 to 4 marks.

Diagram

i) operation modes & privilege Levels in Cortex M-3

	Privilege	User
when running an exception handler	Handler mode	▨
when not running exception handler	Thread mode	Thread Mode

ii) Operation Mode Transition



*The 2 modes of operations are thread mode and handler mode indicates that processor is running a normal/ exception handler.

* privilege levels are privilege level [which supports both modes of operation] and user level [which operates only in thread mode].

* ~~After reset~~, processor will be in Privileged thread mode, from here it can be in user thread or in privileged handler handler mode [normally this state is attained for exception handling], For any exception s only, processor can shift from user thread to Privileged handler, then after finishing it, it should exit and finish the works in user thread.

Q2a] What is stack?, Explain PUSH and POP operation, with the help of a neat diagram [7]

Defn: — 1M, PUSH & POP operation — 4M, Diagram - 2M

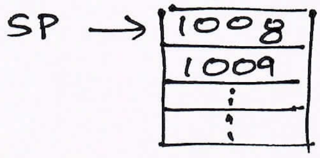
* Stack is the part of RAM, which is used to Store (write) and read data in/from it temporarly during the exceptions, and it can be used for normal operation, The address of stacs is pointe by STACK POINTER Register (SP)

* PUSH and POP are the two instructions which are exclusively used for performing read/write operation w.r.t Stack memory, The PUSH is used to transfer data from memory address to SP address and pop is used to copy data back to the original/Specified memory address from Stack (read)

* For each PUSH SP decrements by 4 and increments by 4 for every POP instruction

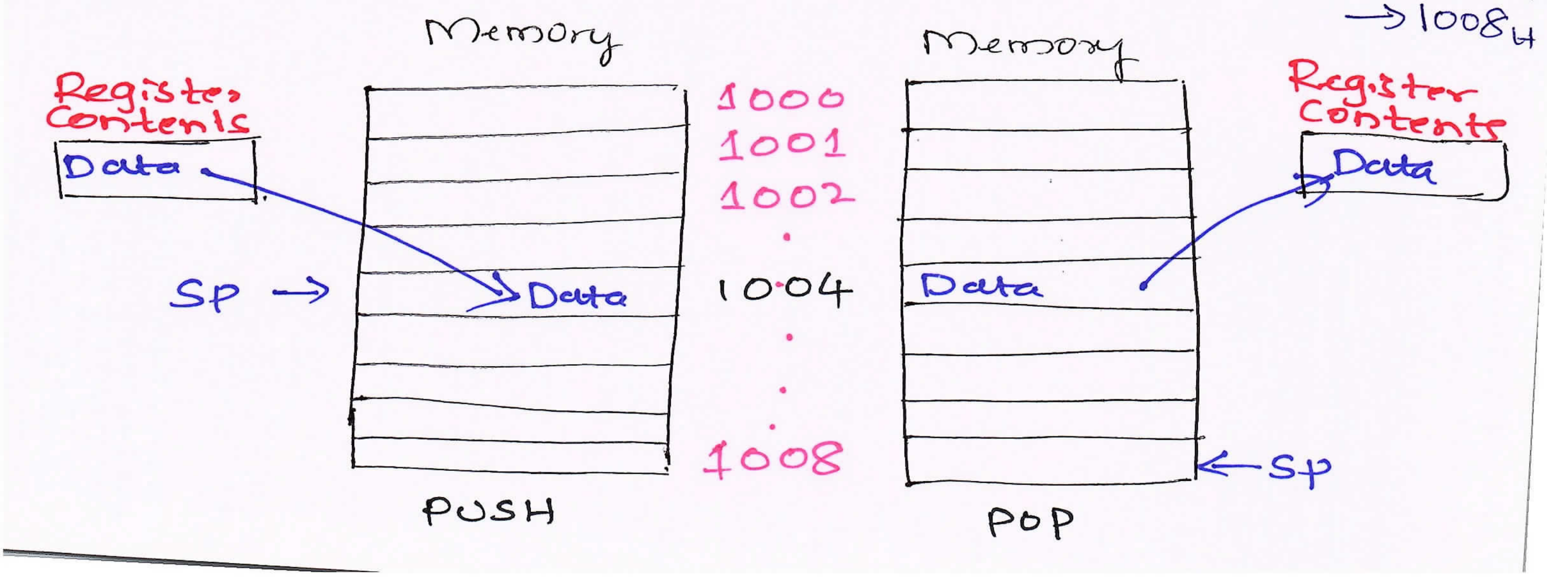
* Stack PUSH and POP operation.

Let us assume SP is pointing at 1008_H address.



* After one PUSH operation SP → SP-4 i.e. SP → 1008-4 → 1004_H.

* After one POP operation SP → SP+4 i.e. SP → 1004+4 → 1008_H



Q.no] 2b: Explain indetail about special registers used in ARM cortex M3 processor. [9M]

Sol^n: The cortex m-3 has 3 main Special Registers. they are

i) Program Status Register (P~~S~~Rs).

→ It is a 32 bit register, divided in to 3 parts namely APSR (Application PSR), which is from bit no 27 to 31; (It can allow both Read/Write) operations; IPSR (Interrupt PSR), which is from bit no 0 to 8 and it is read only, and EPSR (Execution PSR). which is from bit no 10 to 15, 24 and 25 to 26 and it is also READ only.

→ The 3 PSRs can be accessed by special instructions MSR and MRS instructions.

Example: MRS, R0, APSR ; Read flag into R0
MSR APSR, R0; Write flag from R0
MRS R0, IPSR }
MRS R0, EPSR } Read operation

→ when used ^are^ combined one, it is refered as xPSR, but in instruction PSR is valid

→ Structure of PSRs

	31	30	29	28	27	26:25	24	23:16	15:10	9	8	7	6	5	4:0
APSR	N	Z	C	V	Q	Reserved									
IPSR	Reserved										Exception No.				
EPSR	Reserved					ICI/IT	T	Res.	ICI/IT	Reserved					
xPSR	←→														

→ Bit Function

N → Negative
Z → Zero
C → Carry
V → Overflow
Q → Stick saturation flag

} It will be 1, if an operation results MSB=1, entire result zero, A carry from 32nd bit and if result exceeds 32 bit res 4 for N, Z, C and V; else they are 0

ii] Interrupt Mask Registers.

Three registers namely PRIMASK, FAULTMASK and BASEPRI registers are used to disable the exceptions.

Register Name   Function

i) PRIMASK : It is a 1 bit register
: When 1 : masks (disables) all interrupts, except NMI and hard fault.
0 : Masking disabled

ii) FAULTMASK : It is a 1 bit register
when 1: It disables all interrupt except NMI
0 : masking disabled

iii) BASEPRI : It can be up to 8 bits.
When SET, it can disable (mask) the interrupts of the same or different level.

iii] CONTROL Register:

It is of 2 bits

→ CONTROL [1]: used to access SP.
when 1 : PSP is selected
when 0: MSP " " "

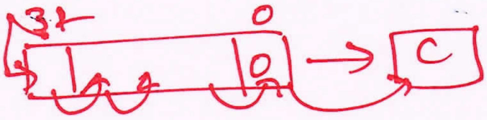
→ CONTROL [0]: used for mode selection
when 1: User state in Thread mode enabled
when 0: Privileged in thread mode enabled

Q3a] Write an ALP to calculate sum of 1 to 10 numbers.

```
          Area   Sumten   Code
          DCD    Stack_Top   ; Define Stack
          DCD    ~~Stack~~ Start ; Reset address
          Entry  ; Execution Starts here
Start                        ; Start of main program
          mov    R0, #10     ; Counter
          MOV    R1, #0      ; Sum = 0
BACK      ADD    R1,R0       ; $\overset{R_1}{sum} = \overset{R_1}{sum} + R0$
          SUBS   R0, #1      ; decrement R0 by 1, update
                               flag
          BNE    BACK     ->; If count not zero continu
                               addition.
BACK1     B  BACK1        -> ; infinite loop
          END. ;          -> end of program
```

Q 3b] Explain the following instruction sets 4M

i) BFC, ii) SBFX, iii) ASR, iv) MRS.

Sl. No	Instruction,	Operation,	Example
①	BFC; Bit Field Clear Syntax: BFC{cond} Rd, #lsb, #Width	It clears width bits in Rd, starting at the low bit position LSB other bits in Rd are unchanged.	BFC R4, #8, #12. ; Clear bit 8 to 19 (12 bits) of R4 to 0.
②	SBFX; Signed Bit Field Extract Syntax: SBFX {cond} Rd, Rn, #lsb, #width	It extracts a bitfield from one register, sign extends it to 32 bits and writes the result to the dest reg	SBFX R0, R1, #20, #4 Extract bit 20 to bit 23 (4 bits) from R1 and sign extend to 32 bits and then write the result to R0
③	ASR Arithmetic Shift Register Syntax ASR Rd, Rm, count	It moves the bits from Rm to right by no of places specified by count	ASR R7, R8, #9 ; Arithmetic Shifted by 9 bits. 
④	MRS Move the contents of special registers to GPR's Syntax MRS Rd, Spe.Reg	Its used for reading contents of a special register and write it on general registers	MRS R0, PRIMASK; Read PRIMASK value and write it to R0

Q. no 3C] Explain how CMSIS provides Standard access. Interface for Embedded Software [Not in new Syllabus].

Q 4a] Write a program to blink a LED using 'c' language.

→
```
#define LED *((volatile unsigned int*)
                  (0xDFFF000C)).

int main (void)
{
  int i;
  volatile int j;
  while (1)
 { LED = 0x00;  /* LED OFF */
  for (i=0; i<10; i++); /* delay */
   { j =0;
    }
  LED = 0xFF or 0x01. ; /* LED ON */
  for (i=0; i<10; i++) / * delay */
   { j = 0;
     }}.
 return 0;
  }.
```

4b] Explain the following assembler directives AREA, ENTRY, DCB, ALIGN

Directives	Descriptions
① AREA	It instructs the assembler to assemble or arrange a new code (ROM) or data (RAM) Segments.
② ENTRY	It is the initialization memory reference address for a program
③ DCB	Define constant Byte, It allocates one or more byte of memory and defines the run time contents of the memory
④ ALIGN.	It helps in aligning current memory location to Specified boundary by padding with zeros.

4c]. Explain different bus interfaces supported by cortex M3. (NOT in New syllabus).

Module-3

Q5 a) Explain how embedded system are clussif

Embedded Systems are clussified as

(i) Generation based.

Generation	Features	Examples
1st Gen.	*Built with 8bit µp and 4-bit µc * hardware is simple * Assembly language is used for development	* Digital Tele phone Keypa
2nd Gen	*Used 16bit µp/8 bit µc * More powerful than 1st Gen	*ADC.
3rd Gen	* Uses 32 bit µp/16bit µc * Domain Specific processor and controllers are used	*Robobics
4th Gen	* 64 bit µp/32 bit µc are used * SoC chip. * Complex, but powerful	* Smart phone

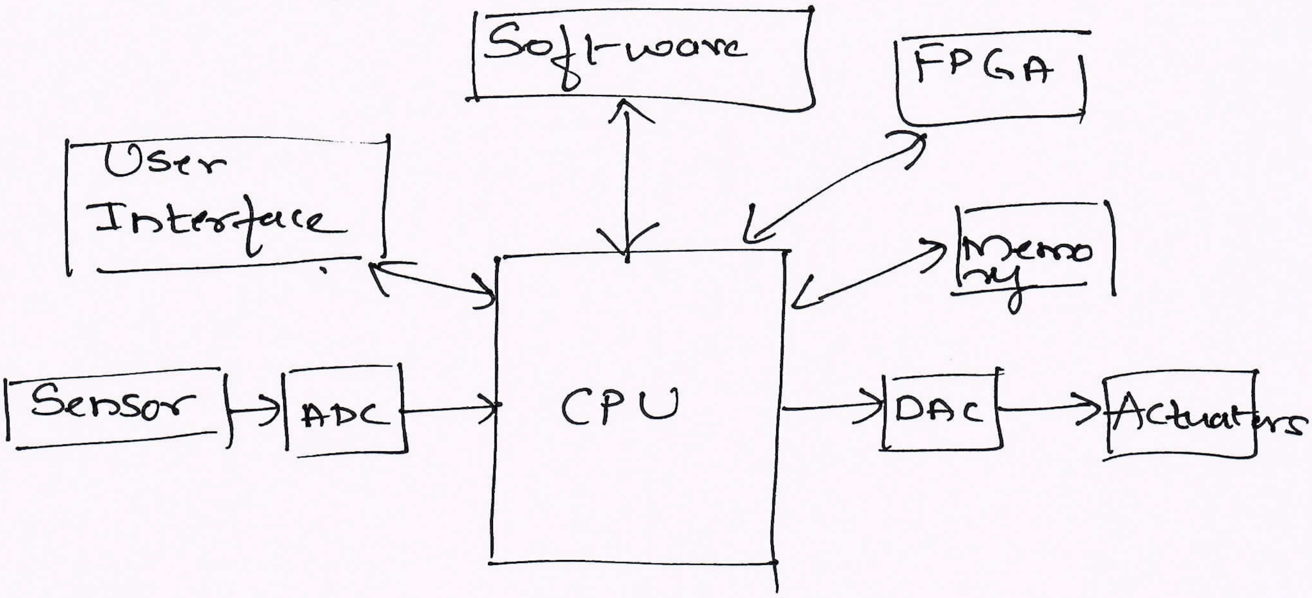
ii) Performance and Complexity based.
Classifications are

a) Small Scale: Uses 8/16 bit µc for design, Toys are the examples, .asm is used

b) Medium Scale: uses single or 16 or 32 bit µc, RISCs or DSPs. .asm / c/c++ are used for development

Q 5b] With a neat block diagram, explain elements of embedded system.

Block Diagram



Explanation

1) An Embedded System has 3 main components hardware, soft-ware and RTOS.

2) hardware includeds processor, memory, ports Sensors, actuators etc

3) Software includes : Languag for interface, application soft-ware etc

4) RTOS: It helps processor to run an application or set of applications on/for Specifed amount/defined time

Example: SMART Phone

h/w – processor, RAM, ROM, etc

s/w – UI, OS, applications

RTOS – UI, OS.

Q.no : 6A] Differentiate between RISC and CISC

RISC	CISC
1) Stands for Reduced Instruction Set Computer	1) Stands for Complex Instruction Set Computer
2) Execution speed is fast	2) Slow compared to RISC
3) Less/Limited no of instructions	3) More no of instructions
4) Support orthogonal execution	4) All instructions are non-orthogonal.

Q.no 6B) Explain how program memory are classified

Program memory or Rom are classified as

i) MROM (Masked ROM):
- It is one time programmable ROM
- It is programmed at vendor end only
- Less Flexible

ii) Programmable Read only Memory (PROM).
- It is programmable at user end also
- It uses FUSE mechanism for programming and its only one time programmable
- It uses AOI logic

iii) EPROM (Erasable PROM):
- It can be reprogrammed by erasing it to the UV lights,
- It uses mos and floating gate concept
- A quartz crystal is used for allowing UV light

iv) EEPROM (Electrically Erasable PROM):
- It can be erased/reprogrammed electrically
- life cycle is limited

Q.no 6C] Explain how brownout- protection Circuit works [NOT in new Scheme]

Module-4

Q.7a] What are the operational and non opera -nal attributes of an embedded system

Sol^n: operational quality attributes

These are the way in which overall impact of operation of an embedded system They are

1. Response: It is the quickness of an e/s output for a given i/p. w.r.t time
2. Throughput: It is the efficiency of the syste It is no of events/work that takes place for a given amount of time
   Ex: Card reader read/write capacity
3. Reliability: It is the measure of dependancy on an E/S, which is decided w.r.t MTBF (Mean Time Between Failures) & MTTR (Mean Time To Repair).
   Ex: Disk Recovery / OS performance
4. Maintainability: It is related to the frequency of recovery time after a failure event
5. Security: Three terms Confidentiality, (protection of data from unknown access); Integrity (Unauthorized modification is not allowed) and Availability
6. Safer: It deals with the possible damage that can be prevented from hazardus conditions.

ii) Non-operational quality attributes.

1) Testability and debugability:

For an embedded system, there Should be a provision for checking the problems occurring w.r.t h/w or s/w through testable Tools and at the same time we need to hav sol^n to them using proper debug methods.

Ex: Detection of ports for USB, other device.

2) Evolvability: there should not be any generation gap difference, interms of working style, environment for embeded systems.
Ex: Win98, if it works now on new PC, it is having good evolvability.

3) Portability: Irrespective of different processor / controller, the performance of an application should not be affected

4) Time to prototype and market:

It is the duration required to design a new embedded product, and release of the same in to the market before a similar product gets launched

5) Per Unit and Total cost:

It is the cost of a product and total investment needed to produce in a bulk

Q no 7b] Explain the different types of Serial Interfaces bus used in automotive communications.

→ Three bus architecture/technologies are used they are CAN, LIN and MOST.

→ BUS_Name	Description
① CAN	* It stands for Control Area Network. * used in serial communication * It was released in 1986 by Rober Bosch and aliance * Mostely used in the automotive serial bus architecture * It messages with an 11 bit message ID which identifies the message type and also establishes the message priority * It uses message ID to perform bus access arbitration b/w nodes. * At a given time either it can send or receive the bits.
② LIN	* It Stands for Local Interconnect Network * Used for in-vehicle communication and n/w, with the help of sensors and actuators * It uses single master and multiple slave bus that uses a single wire for transmission of data * only master can initiate the transmission through header pull- * Example: Windows, Wipers and AC

③ MOST bus: It stands for Media Oriented System Transport, founded in 1998 by Bmw, Becker etc. It uses daisy chain or ring configuration to distribute audio, video signals, It usen FON technology and it can handle 65 devices.

Q8 a>. Explain fundamental issues in hardwar software co-design.

① Selecting the model: Model is a reference design or method used in the embedded sys

② Selecting the Architecture:

It gives the complete information about th implementation details. The various architect are

a> Controller Architecture: It uses State mach model, which uses State register and two combinational ckts.

b> Data path architecture: Used in DFG model data path is the channel through which data is transmitted

c> FSMD [Finite State Machine Datapath]. It is the combined technique of a> and b>

d> CISC and RISC: These are the architectures refered for deciding the instruction design or format

③ ~~c>~~ Selecting the Language: Here, it allows the designers to use C, C++, Java, HDL etc development languages.

④ Partitioning System Requirements into hardware and Software

H/w

* For deciding the architecture based on Harvard / Von-Neuman

S/w

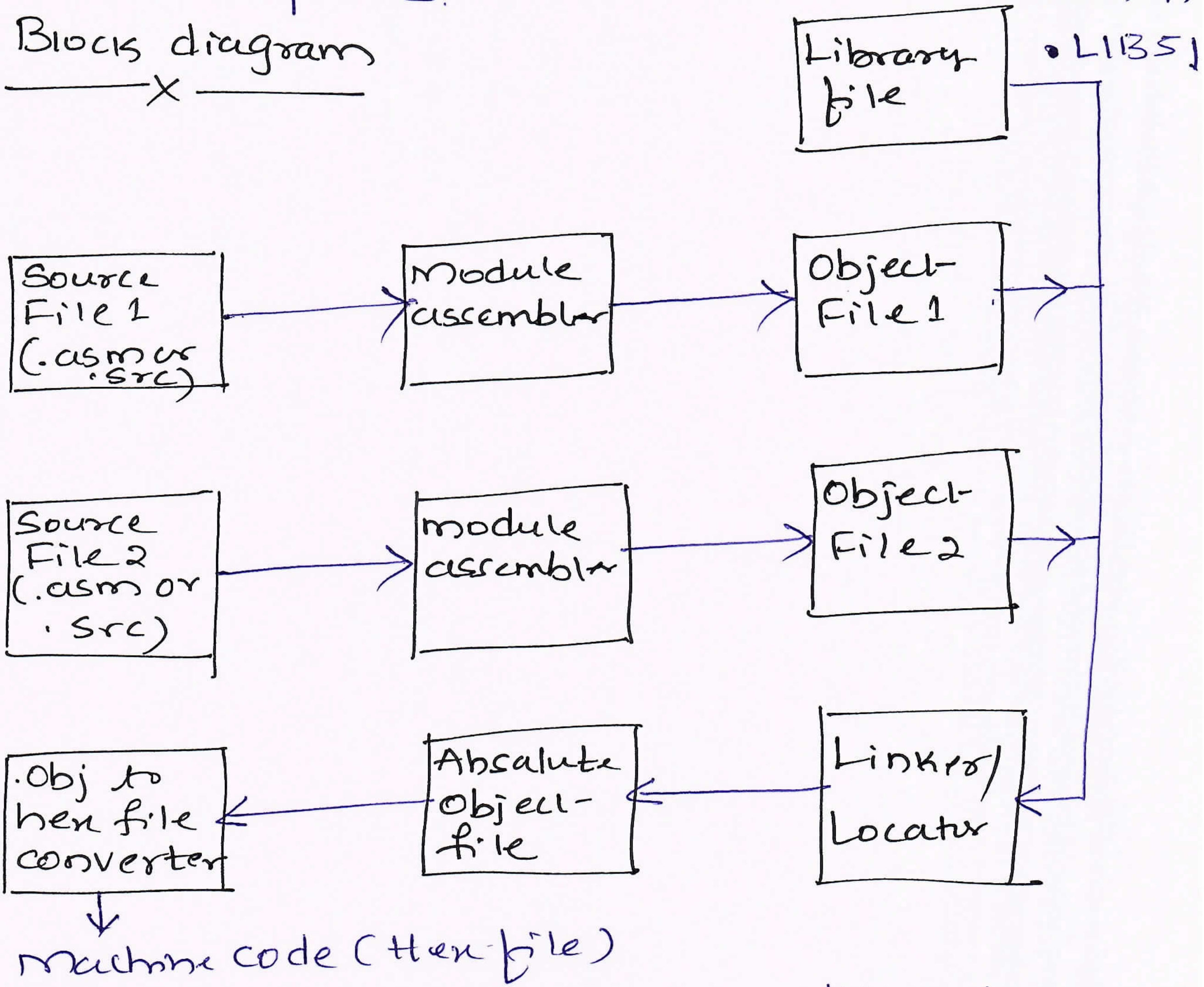
* For deciding Instruction architecture based on RISC/CISC

Page - 1

8 b] Explain with a neat- block diagram. how source file to object- file translation takes place.

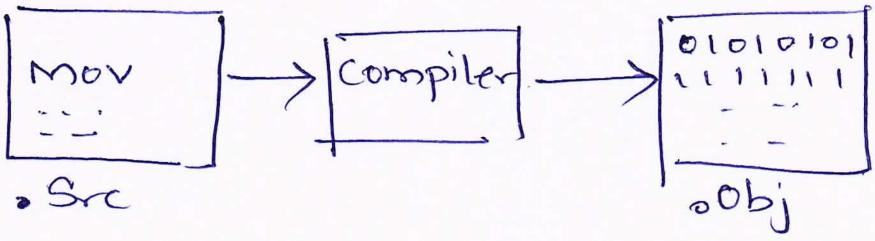
Block diagram

——x——



1) The programmer uses text editor to create the source file which can be in .c/.asm [This is high level language]

2) Then with the help of language translator, the source file can be converted to object file, which contains 0 and 1.
C-51 is a famous compiler used by keil μvision



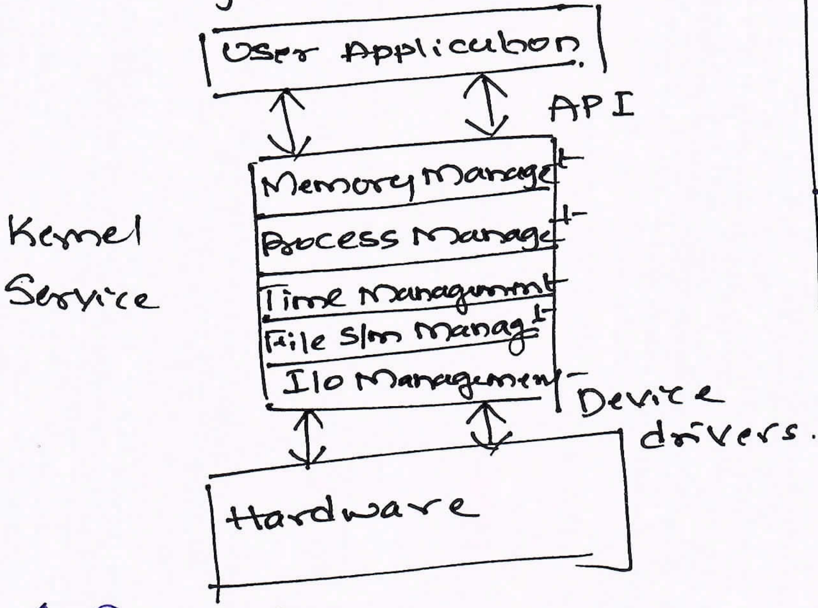
Q.no 8c] Explain super loop based approach of embedded firmware design

Sol^n:
- This type of approach is used for applications that are not time critical and where response time is not much important
- It does not requires OS.
- Examples are video games, printer, scanner
- The whole code for the system is written as one loop which executes continuosly.

Module-5

Q.no 9a] With neat diagram explain OS architecture

Diagram



→ OS acts like a bridge b/w user applications and intended hardware on which it is running

→ Kernel is a core of OS which is responsible for managing the resources

→ Overall OS architecture will have user application various management and hardware.

1. Process Management: Process is a program of an application which is running, here OS decides how many resources should be allocated with how much quantity, Ex: Printing cycle: Needs memory temporarily, RAM, processor CPU clock cycle.
2. Memory Management: Here OS decides, the amount of memory (RAM + Rom) needed to run a particular application.
3. Device Management: It deals with the which device (like USB, Printer, mouse), can consume the amount of RAM / Rom to be allocated
4. File Management: It is the collection of files, which can be created, deleted, modified by OS.
5. I/o System management:

Page-21

Q. 9B] Explain how OS are classified - 4M

Sol^n: They are classified as

i) General Purpose OS: It is the OS, on which any applications can be installed and used for the applications which are not time critical, example: Windows, Unix, etc.

ii) Real Time OS: These types of OS are time bounded OS, typical used for time critical appli-cations, example: Vx-works, Applications like sensor detection s/m use it

Q 9C]. Differentiate b/w hard real time s/m and soft real time s/m

Sl. No.	Hard Real Time	Soft Real Time
1)	It is used in time bounded appl^n where dead line is critical	Its ment for the applications where dead line can be flexible
2)	safety is critical	safety is not critical
3)	Data integrity time is less	It is flexible here
4)	Ex: OTP is a example	Ex: ATM transaction bw

Q.no 10] b].

Given.

Jobs	CPU bus time	Arrival Time
1	10	0.0
2	2	3.0
3	1	4.0
4	4	5.0
	17.	12

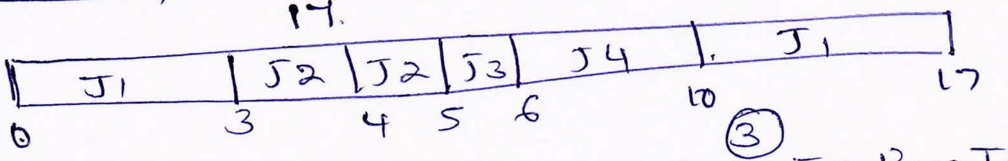
① Wait time

$J_1 = 0 - 0 + 10 - 3 = 7$

$J2 = 3 - 3 = 0$

$J3 = 5 - 4 = 1$

$J4 = 6 - 5 = 1$



② $AWT = (7+0+1+1)/4 = 2.25$;

③ TAT = Bus Time + WT

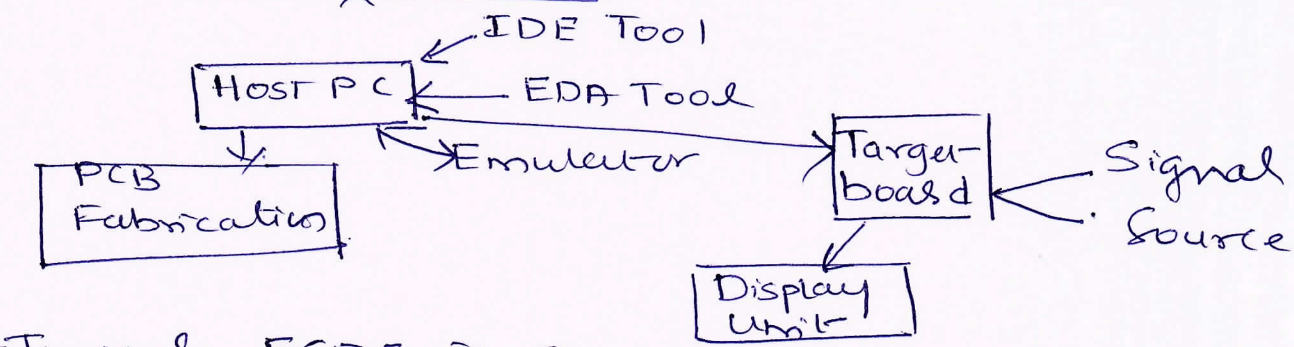
J1 =	10	+ 7	= 17
J2 =	2	+ 0	= 2
J3 =	1	+ 1	= 2
J4 =	4	+ 1	= 5

④ $ATT = (17+2+2+5)/4$

$= 6.5.$

Q.10a]. With neat diagram explain embedded System development environment

→ Block Diagram



* Typical ESDE is shown above, It can be assumed like a smart phone (HOST-PC), which is supporte by all other parameters shown.

* Consider

1) IDE: Integrated development environment, It is the s/w that assist programmers in developin software code on a h/w platform

2) EDA: Set of s/w tools used for chip design. examples: Xilinx, etc.

3) Emulator: It immitates or it is the ability to replicate same function of a device virtually.

4) Display Units: They can be like oscilloscope monitors used to study the behaviour of s/m.

Q.10c] Write a note on IAP — 4M

* Stands for In Application Programming
* It is the ability to erase and program the code memory in the end user application is "IAP"
* Examples are BIOS, System Restore in mobile
* It will be having a pre-programmed area in the memory which can run with some user interface commands.