Name of the faculty : Prof. Rohini Kallur
Prof. Nikhil Kulkarni

Name of Institution : KLS VDIT, Haliyal

Department : Electronics and Communication Engineering

Sem : 6th Semester B.E Degree Examination
July / August 2022

Subject : Embedded Systems
Scheme and Solution

Subject code : 18EC62

Prof. Rohini Kallur

Prof. Nikhil Kulkarni

Subject Teacher

14.09.2022
Head of the Department
Dept. of Electronic & Communication Engg.
KLS V.D.I.T., HALIYAL (U.K.)

CBCS SCHEME

USN | 2 | V | D | 1 | 9 | E | C | 0 | 1 | 5 |

18EC62

Sixth Semester B.E. Degree Examination, July/August 2022
Embedded Systems

Time: 3 hrs. Max. Marks: 100

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

Module-1

1 a. With a block Schematic, explain the function of various units in ARM cortex M3 processor architecture, in brief. **(10 Marks)**

b. Explain any 5 application of ARM cortex M3 based on its features. **(05 Marks)**

c. With diagram, explain 2 operation modes and 2 privilege levels in cortex M3, when exceptions are to be handled. **(05 Marks)**

OR

2 a. With tables, describe the various interrupts and exception along with the vector addresses. **(10 Marks)**

b. Explain Program Status Registers (PSRs) in cortex M3 along with the 2 instructions used for accessing PSRS, with a diagram. **(05 Marks)**

c. Describe the reset sequence with a diagram. **(05 Marks)**

Module-2

3 a. Explain the 16 bit instructions: CMP, ASR, SBC and LDMIA, with an example for each. **(08 Marks)**

b. Describe signed and unsigned saturation instructions with diagram and examples. **(08 Marks)**

c. Explain IT instruction with an example to convert a High level language instruction to its equivalent assembly instructions in cortex M3. **(04 Marks)**

OR

4 a. Explain the following 32 bit instructions with an example for each : ADC, BFC, LSL and PUSH. **(08 Marks)**

b. Describe CMSIS with diagram and its functions, organization and scope. **(08 Marks)**

c. Write an ALP to add the first 10 integer numbers using cortex M3 processor. **(04 Marks)**

Module-3

5 a. Describe the elements of an embedded system with a block diagram. **(10 Marks)**

b. Classify the embedded systems based on the complexities and give 2 examples for each category. **(06 Marks)**

c. Differentiate between RISC and CISC architectures. **(04 Marks)**

OR

6 a. Describe the functions of Optocoupler, I2C and IrDA for embedded system. **(10 Marks)**

b. Explain EPROM, EEPROM, FLASH, DRAM, NVRAM and Sensors required for embedded systems. **(06 Marks)**

c. Differentiate between Embedded and general computing systems. **(04 Marks)**

Module-4

7 a. Describe coin operated telephone system with a FSM, function of states and state transition diagram. **(08 Marks)**

b. Explain any 5 characteristics of embedded systems. **(05 Marks)**

c. With a block schematic, explain the ALP based embedded firmware design with its disadvantages. **(07 Marks)**

OR

8 a. Describe the sequential program model for seat belt warning system along with the operation of the system. **(08 Marks)**

b. Explain any 5 operational quality attributes of embedded systems. **(05 Marks)**

c. With a functional block diagram, explain the working of a washing machine. **(07 Marks)**

Module-5

9 a. With the state transition diagram, structure of a process and memory organization, explain the functions of status and the scheduler function for process management. **(10 Marks)**

b. With an example, describe preemptive SJF scheduling and calculate all the performance factors. **(10 Marks)**

OR

10 a. Describe out-of-circuit programming and In-system-programming. **(10 Marks)**

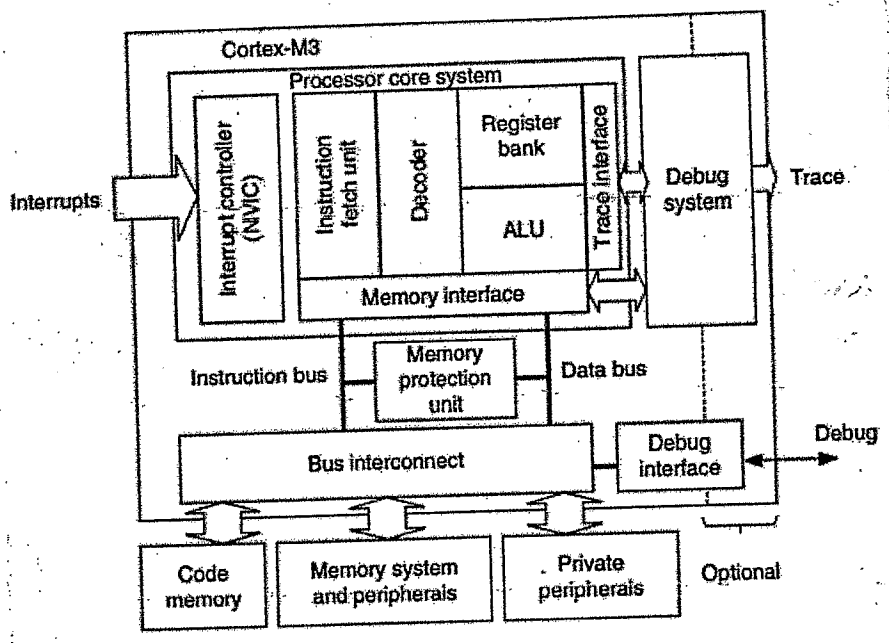
b. With a block diagram, explain the embedded system development environment with the functions of the components used in brief. **(10 Marks)**

* * * * *

Module 1

1.a. With a block schematic explain the various units in ARM Cortex M_3 processor architecture, in Brief. [10 Marks]

Ans:



Block dgm — 5M

Explanation — 5M

Fig 1: Architecture of ARM Cortex M_3

Cortex M_3 is a 32-bit Microcontroller. It has 32-bit data path, a 32-bit register bank and 32-bi

memory interface:

The processor has a Harvard Architecture, wh means that it has a separate instruction bus and data bus. This allows instructions and accesses to place at the bus time as a result of this, the perf -mance of the processor increases because data acce do not affect the instruction pipeline.

ARM Cortex has many special features: option Memory protection unit, Internal debugging compone break points and watch points, Register set, Interface Trace unit and so on.

ARM cortex M_3 has Registers R_0 through R_{15}. In which $R_0 - R_{12}$ are general purpose registers R_{13} is the Stack pointer, R_{14} is the Link register and R_{15} is the program counter.

ARM cortex M_3 also has special registers such as Program Status Register (PSR), Interrupt Mask registers [PRIMASK, FAULTMASK & BASEPRI] and a control register of 2 bits.

ARM cortex has a built in NVIC which provides multiple features to handle the interrupts. ARM Cortex M_3 also has predefined memory map. The total memory of 4GB is divided into multiple sections and allocated as code Region, RAM Region, Peripheral & System level region.

The processor also has bus interface, in which we have code memory bus, system bus and private peripheral bus. ARM Cortex M_3 has an optional MPU. This unit allows access rules to be set up for privileged access and user program access.

ARM Cortex M_3 supports Thumb-2 instruction set. It allows 255 exceptions which includes external interrupts, exception and NMI, Reset.

The cortex M_3 processor includes a number of debugging features such as program execution control including halting & stepping, instruction breakpoints, data watch points, registers and memory access, Profiling and traces.

1.b. Explain any 5 applications of ARM cortex M_3 based on its features. [05 Marks]

Ans: Applications of ARM cortex M_3 are:

i) Low cost Micro controllers —

These microcontrollers can be used in consumer products, from toys to electrical appliances or even specialized products for IT, Industrial or even medical systems.

ii) Automotive applications

As these processors offer great performance, very high energy efficiency and low interrupt latency, the processors are best suited for automotive applications.

iii) Data communications

The processors low power and high efficiency makes it ideal for many communication applications such as Bluetooth and ZigBee.

iv) Industrial control

In Industrial control applications, Simplicity fast response and reliability are key factors.

v) Consumer products

In many consumer products, a high performance microprocessor is used. The cortex M_3 processors are small, efficient and low in power consumption. So they are used in almost all consumer products.

Explaination of each application - 1 Mark

1.C. With diagram, explain 2 operation modes and 2 privilege levels in Cortex M_3, when exceptions are to be handled. [05 Marks]

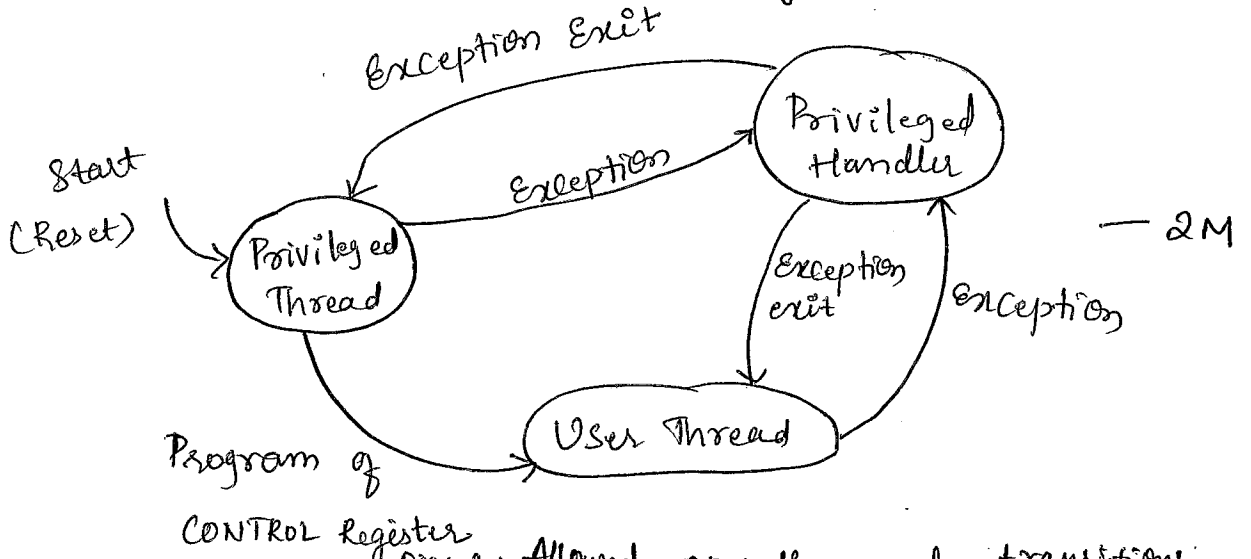
Ans: Cortex M_3 has two modes and two privilege levels.

The operation modes: Thread mode and handler mode determine whether the processor is running a normal code or running an exception handler like an interrupt handler.

When Running	Privileged	User
exception handler	Handler mode	
when not running an exception handler	Thread mode	Thread mode

— 1M

Fig 1: Operation modes & Privilege levels



— 2M

Fig. 2: Allowed operation mode transitions

When the processor is running a main program, it can be either in a privileged mode or user state. But exception handler can only be in a privileged state.

When processor exits reset, it is in thread mode with privileged access rights.

Explanation – 2M

In the privileged state, the program has access to all memory ranges and can use all supported instructions.

As shown in figure 2, 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.

A user program cannot change back to the privileged state 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 returning to thread mode.

2.a. With tables, describe the various interrupts and exceptions along with the Vector addresses. [10 Marks]

Table 3.4 Exception Types in Cortex-M3

Exception Number	Exception Type	Priority	Function
1	Reset	–3 (Highest)	Reset
2	NMI	–2	Nonmaskable interrupt
3	Hard fault	–1	All classes of fault, when the corresponding fault handler cannot be activated because it is currently disabled or masked by exception masking
4	MemManage	Settable	Memory management fault; caused by MPU violation or invalid accesses (such as an instruction fetch from a nonexecutable region)
5	Bus fault	Settable	Error response received from the bus system; caused by an instruction prefetch abort or data access error
6	Usage fault	Settable	Usage fault; typical causes are invalid instructions or invalid state transition attempts (such as trying to switch to ARM state in the Cortex-M3)
7–10	—	—	Reserved
11	SVC	Settable	Supervisor call via SVC instruction
12	Debug monitor	Settable	Debug monitor
13	—	—	Reserved
14	PendSV	Settable	Pendable request for system service
15	SYSTICK	Settable	System tick timer
16–255	IRQ	Settable	IRQ input #0–239

— 3M

Table 3.5 Vector Table Definition after Reset

Exception Type	Address Offset	Exception Vector
18-255	0x48-0x3FF	IRQ #2-239
17	0x44	IRQ #1
16	0x40	IRQ #0
15	0x3C	SYSTICK
14	0x38	PendSV
13	0x34	Reserved
12	0x30	Debug monitor
11	0x2C	SVC
7-10	0x1C-0x28	Reserved
6	0x18	Usage fault
5	0x14	Bus fault
4	0x10	MemManage fault
3	0x0C	Hard fault
2	0x08	NMI
1	0x04	Reset
0	0x00	Starting value of the MSP

-2M

When an exception event takes place on the Cortex M_3 and is accepted by the processor core, the corresponding exception handler is executed. To determine the starting address of the ~~execution~~ exception handler, a vector table mechanism is used.

The vector table is an array of word data inside the system memory, each representing the starting address of one exception type. ~~The~~

If the reset is exception type 1, the address of the reset vector is 1 times 4, which equals 0x00000004 and NMI Vector (type 2) is located in 2x4 = 0x00000008. The address 0x00000000 is used to store the starting value for the Msp.

Explaination - 5M

2.b. Explain Program status Register (PSR) in cortex M_3 along with the 2 instructions used for accessing PSRs, with a diagram. [05 MARKS]

Ans:

	31	30	29	28	27	26:25	24	23:20	19:16	15:10	9	8	7	6	5	4:0
APSR	N	Z	C	V	Q											
IPSR													Exception number			
EPSR							ICI/IT	T				ICI/IT				

— 3 M
explanation — 2 M

Program Status Registers are subdivided into three Status registers.

i) Application Program status Register
ii) Interrupt Program Status Register
iii) Execution Program status Register

Three PSRs can be accessed together or seper using the special register access instructions MSR a MRS. when they are accessed as a collective item, th name xPSR is used.

We can read PSR using MRS instruction We can also change the APSR using MSR instruct but EPSR and IPSR are read-only.

For Eg:

```
MRS   r0, APSR ; Read flag state into R0
MRS   r0, IPSR ; Read Exception/Interrupt
MRS   r0, EPSR ; Read execution State
MSR   APSR, r0 ; write flag state
```

If all the PSR is accessed at once, we can use

```
MRS   r0, PSR ; Read PSR
MSR   PSR, r0 ; write to PSR.
```

2. c. Describe the reset sequence with a diagram.

[05 Marks]

Ans:

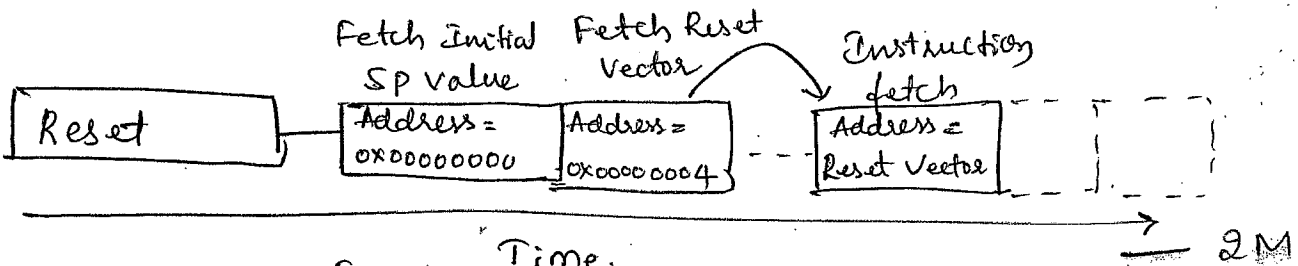
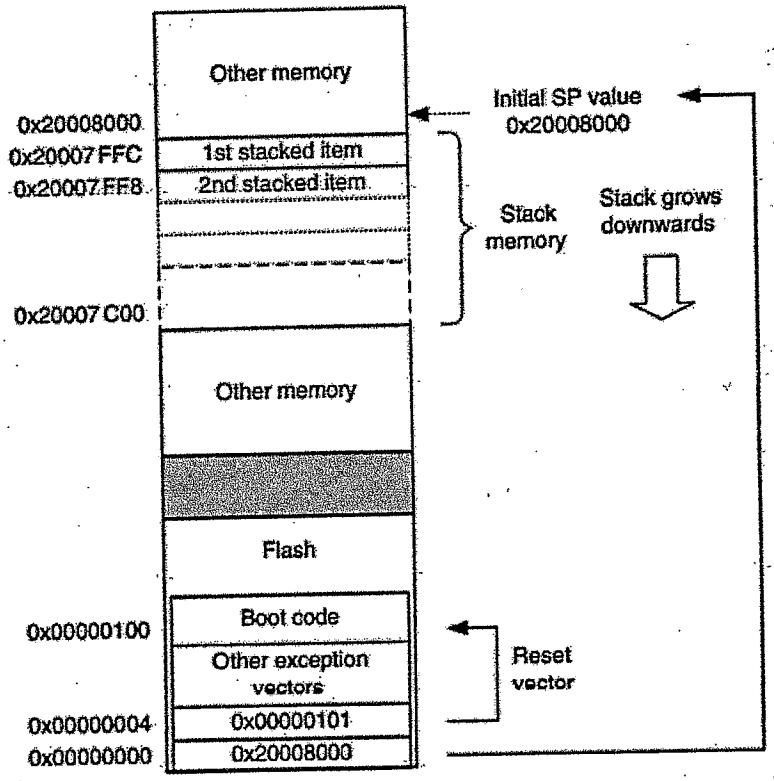


Fig. 1: Reset Sequence

After the processor exits reset, it will read two words from memory

i) Address 0x00000000 : Starting value of R_{13}

ii) Address 0x00000004 : Reset vector



— 2M

explaination — 1

As shown in the vector table, first vector is the reset vector, which is the second piece of data fetched by the processor after reset.

— 2M

explanation — 1

3.a. Explain the 16-bit instructions : CMP, ASR, SBC and LDMIA with an example for each. [08 marks]

Explanation of each instruction - 2M

Ans: i) CMP : compare instruction

The compare instruction subtracts two values and updates the flags, but the result is not stored in any registers.

Ex: CMP R_0, R_1 ; calculate $R_0 - R_1$ & update flag.

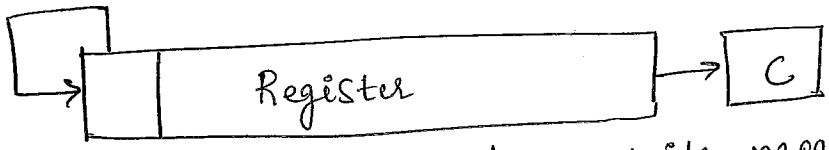
CMP R_0, #0x12; calculate $R_0 - 0x12$ & update flag

ii) ASR : Arithmetic Shift right.

ASR R_d, R_n, #immed

ASR R_d, R_n

ASR.W R_d, R_n, R_m



Arithmetic shift right by n bits means the data will be shifted right by n bits and LSB is moved to carry flag and MSB bit is remained as it is in the MSB.

Ex: ASR R_3, R_3, #1 ; R_3 is arithmetically right shifted once and stored back in R_3.

iii) SBC : Subtract with carry

SBC R_d, R_m ; $R_d = R_d - R_m$ - borrow

SBC.W R_d, R_n, #immed ; $R_d = R_n$ - #immed - borrow

SBC.W R_d, R_n, R_m ; $R_d = R_n - R_m$ - borrow

Ex:

SBC R_2, R_1 ; $R_2 = R_2 - R_1$ - borrow.

Subtracts R_1 from R_2 with borrow and store the result in R_2.

iv) LDMIA: Load multiple increment after.

LDMIA $R_d!$, <reg list>

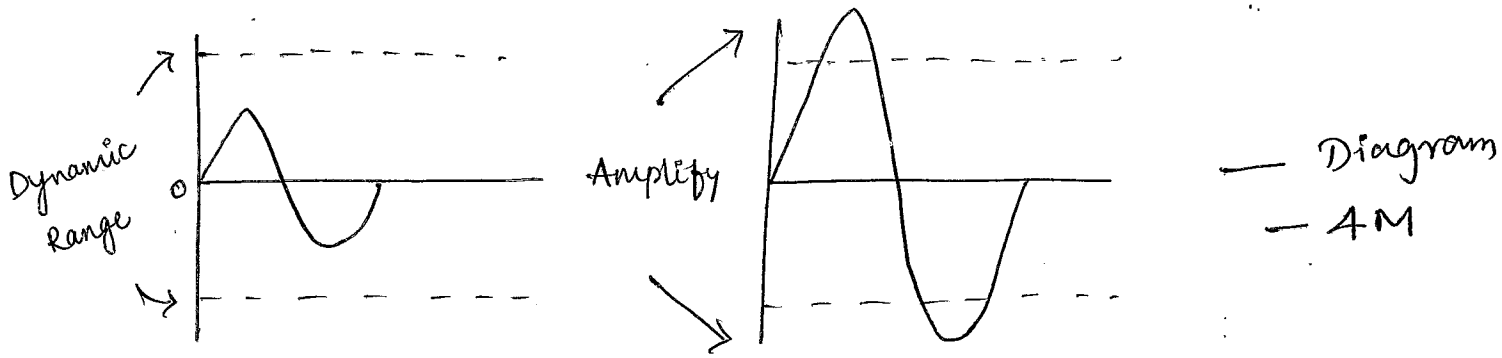
Read multiple words from memory location specified by R_d; address increment after each transfer

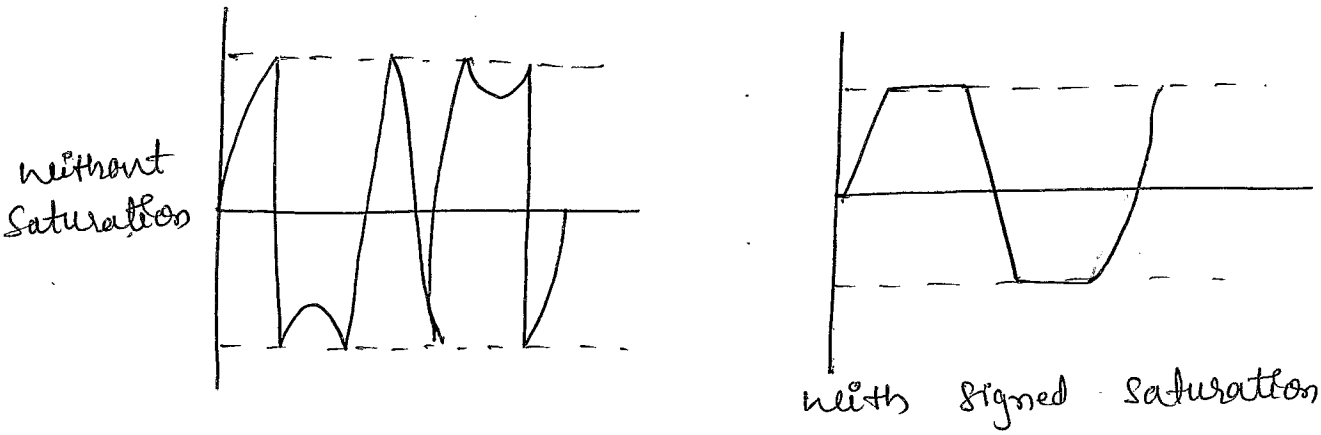
Eg: LDMIA $R_4!$, {R_0-R_1}; Read two registers from table and store in the register.

3.6. Describe Signed and unsigned saturation instructions with diagram and examples. [08 Marks]

Ans: Cortex M_3 supports two instructions that provide signed and unsigned saturation operations: SSAT and USAT for signed data type and unsigned data type respectively. Saturation is commonly used in signal processing.

* when an input signal is amplified, there is a chance that the o/p will be larger than the allowed o/p range.





The saturation operation does not prevent the distortion of the signal, but at least the amount of distortion is greatly reduced in the signal waveform.

SSAT.W <Rd>, #imme, <Rn>, {, <shift>}

USAT.W <Rd>, #imme, <Rn>, {, <shift>}

Eg: SSAT.W R1, #16, R0

USAT.W R1, #16, R0 Explaination – 4M

3.c. Explain IT instruction with an example to convert a High level language instruction to its equivalent assembly instructions in cortex M3. [04 Marks]

Ans: IT instruction is If – then instruction. It helps porting of assembly application codes from ARM cortex M3. When ARM assembler is used, and if a conditional execution instruction is used in assembly code without IT instruction, the assembler can insert the required IT instruction automatically.

Eg: High level language Code

```
if (R0 equal R1) then
{
    R3 = R4 + R5
    R3 = R3/2
}
else
{
    R3 = R6 + R7
    R3 = R3/2
}
```

Eg: Assembly code

```
CMP R0, R1
ITTEE EQ
ADDEQ R3, R4, R5
ASREQ R3, R3, #1
ADDNE R3, R6, R7
ASRNE R3, R3, #1
```

Assembly code – 2M

High level code – 2M

4. a) Explain the following 32-bit instructions with an example for each: ADC, BFC, LSL and PUSH.

[08 Marks]

Ans:- i) ADC : Add with carry

Explanation of each instruction - 2M

ADC R_d, R_n, R_m; $R_d = R_n + R_m$ + carry

ADC R_d, #immed; $R_d = R_d$ + #immed + carry

The instruction adds the values in R_n & operand2 or immed data together with carry flag.

Ex: ADC R_0, R_1, R_2; $R_0 = R_1 + R_2$ + carry

ADC R_0, #012H; $R_0 = R_0$ + 12H + carry

ii) BFC : Bit field clear

BFC.W <R_d>, <#LSB>, <#width>

Bit field clear instruction clears 1-31 adjacent bits in any position of a register.

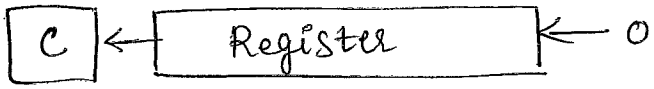
For example;

LDR R_0, = 0x1234FFFF

BFC.W R_0, #4, #8.

This will give R_0 = 0x1234000F.

iii) LSL : Logical shift Left



LSL R_d, R_n, #immed ; $R_d = R_n$ << immed

LSL R_d, R_n ; ; $R_d = R_d$ << R_n

LSL.W R_d, R_n, R_m; ; $R_d = R_n$ << R_m

The content of Register will be left shifted and MSB will be moved to carry flag and the LSB will be filled with 0s.

Ex: LSL $R_0, R_0, \#2$; $R_0 = R_0 << 2$

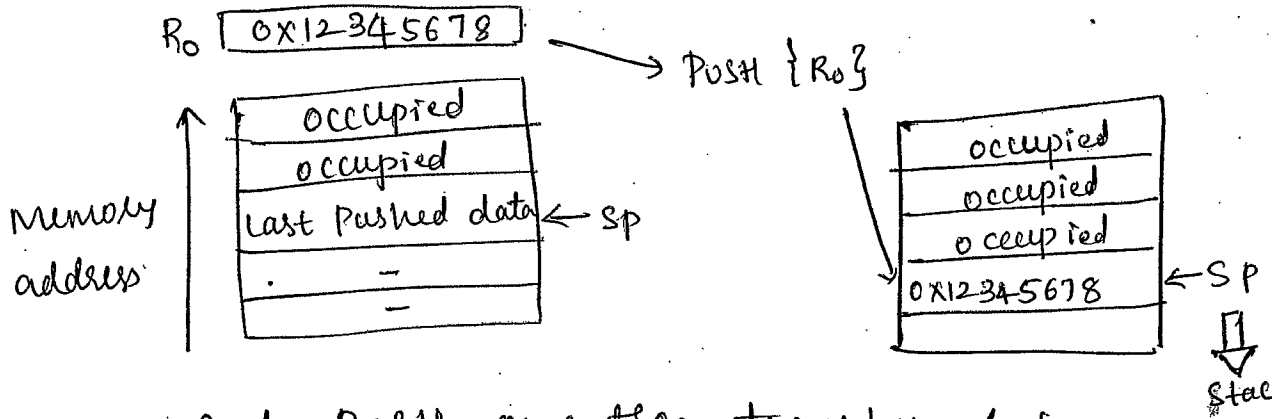
LSL R_2, R_2, R_1; $R_2 = R_2 << R_1$

iv) PUSH :

PUSH {R_0}; Store R_0 to stack

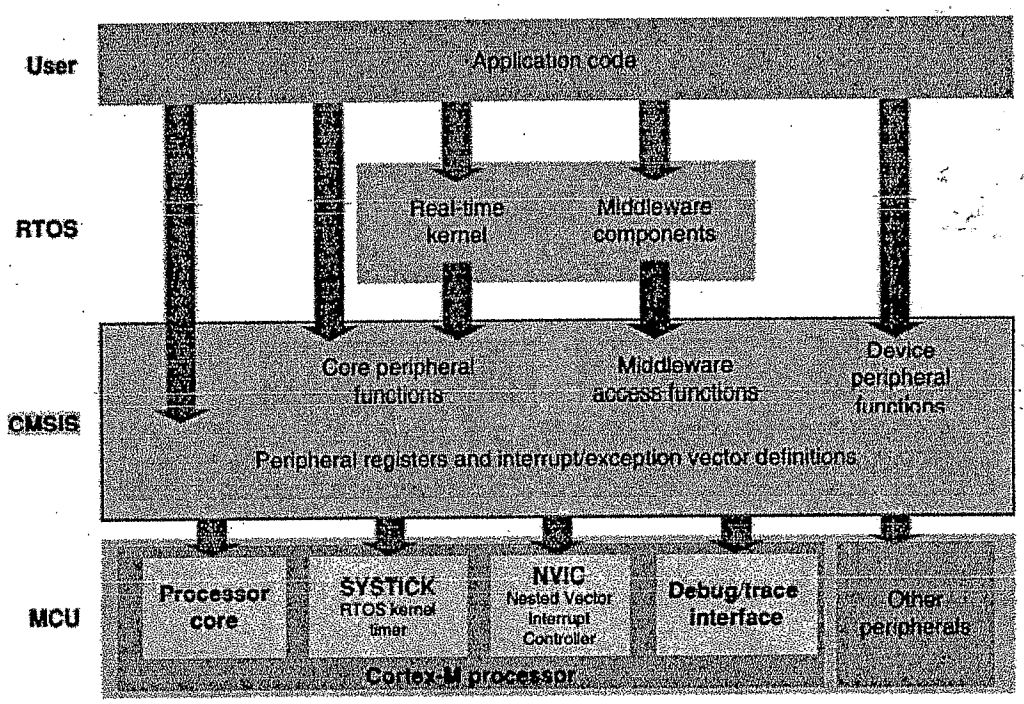
PUSH {R_1}; Store R_1 to stack

PUSH {R_2}; Store R_2 to stack



Each PUSH operation transfers 4 bytes of data, the sp decrements by 4 at a time or a multiple of 4 if more than 1 register is pushed.

4:b) Describe CMSIS with diagram and its functions, organization and scope. [08 Marks]



— 4M
Explanation
— 4M

fig: CMSIS Architecture

CMSIS defines the basic requirements to achieve Software reusability and portability.

The organization of CMSIS can be shown as in figure(1).

CMSIS is divided into multiple layers as explained below

i) Core peripheral Access layer
ii) Middle ware Access layer
iii) Device peripheral Access layer
iv) Access functions for peripherals.

The Scope of CMSIS

i) Hardware Abstraction layer for cortex M processor regi
ii) Standardised System exception names
iii) Standarised method of header file organization
iv) Common method for System initialization
v) Stardardised intrinsic functions
vi) Common action functions for communication
vii) Standardised way for embedded Software to determine System clock frequency.

Benefits of CMSIS :-

The main advantage of CMSIS is much better portability and reusability. It allows Software to be quickly ported between cortex M_3 and other cortex-M processors, reducing time to market.

4.c) Write an ALP to add the first 10 integer numbers using cortex M_3 processor. [04 Marks]

Ans:

```
        MOV  R1, #10
        MOV  R2, #0
Loop:   ADD  R2, R1, R1
        SUBS R1, #0X01
        BNE  loop
```

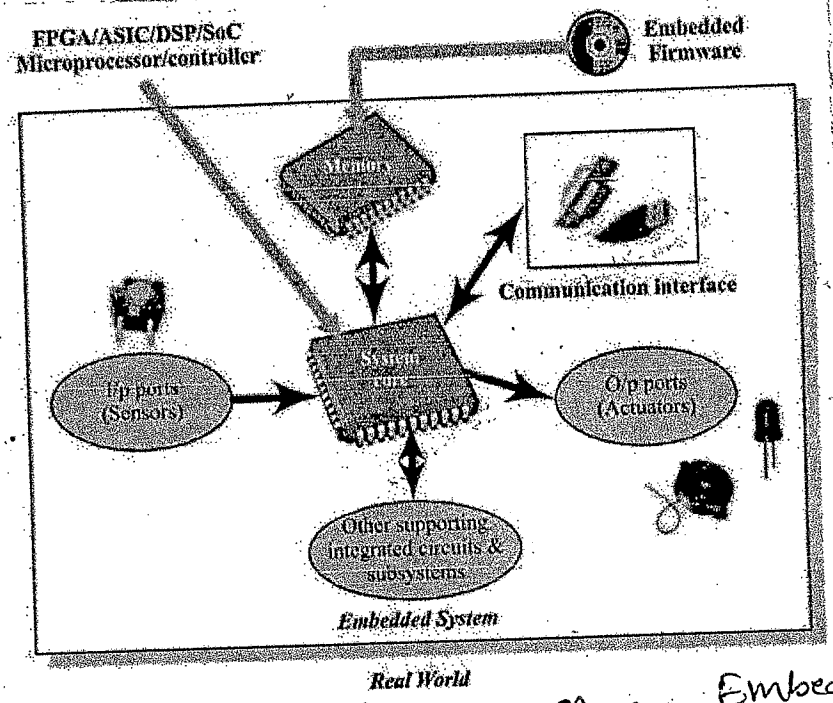
Program – 4 Marks

STR R_2, [R_0]

BX LR

5.a. Describe the elements of an embedded System with a block diagram. [10 Marks]

Ans.



— 4M

Explaination = 6M

fig: Block diagram of an Embedded System

A typical Embedded System contains a single chip controller, which acts as the master brain of the System.

The elements of an embedded System are

i) System core ii) I/p ports iii) O/p ports iv) Memory

v) other supporting IC and subsystems.

Core of an Embedded System

Embedded Systems are domain and Application Specific and built around a central core. The core of the Embedded System falls into any one of the following categories

i) General purpose and Domain Specific processors

ii) ASICs

iii) Programmable logic devices (PLD)

iv) Commercial off the Shelf components (COTS)

Memory:

Memory is an important part of a processor or controller based embedded systems. Some of the Processors / controllers contain built in memory or on-chip memory. Other donot contain any memory inside the chip, hence require off-chip memory

Sensors and Actuators:

A sensor is a transducer device that converts energy from one form to another for any measurem or control phase. Sensor is an input device.

Eg: Hall effect sensor, temperature sensor

Actuator is a form of Transducer device which conve Signals to corresponding physical action. Actuator is an o/p device.

Eg: Relay

Other Supporting ICs and Subsystems:

1) Reset circuit —

Reset circuit brings the internal registers and hardware systems of a processor/controller to a known state and starts the firmware execution from the Reset vector.

2) Real time clock and watchdog timer:

RTC is a s/m component responsible for keeping track of time. RTC is intended to function even in the absence of power.

watchdog timer is a hardware timer for monitoring the firmware execution.

5.b. Classify the embedded systems based on the complexities and give 2 examples for each category. [06 Marks]

Ans: According to the complexity of embedded systems, the Embedded Systems are classified in to the following categories. Each classification – 2M

i) Small scale Embedded Systems

Embedded systems which are simple in application needs and where the performance requirements are not time critical fall under this category.

Ex: Electronic toys, watches

ii) Medium-scale Embedded Systems

Embedded systems which are slightly complex in hardware and firmware requirements fall under this category. These systems usually contain operating System.

Ex: Industrial machines

iii) Large scale Embedded Systems

Embedded systems which involve highly complex hardware and firmware requirements fall under this category.

Ex: Mission critical applications

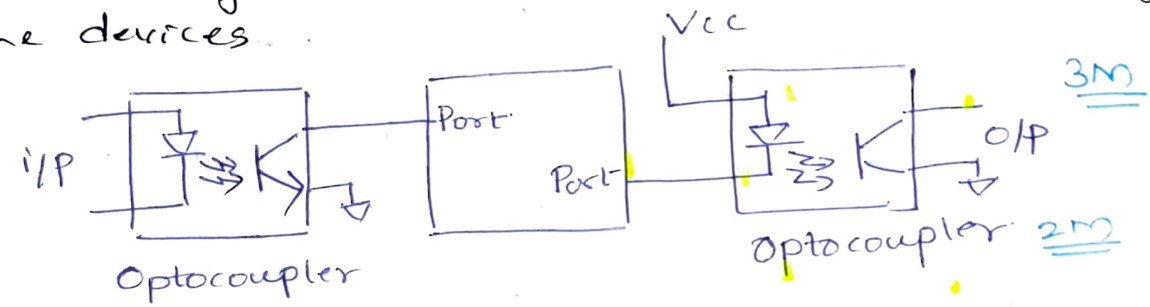
5.c. Differentiate between RISC and CISC architectures. [04 marks]
Each difference – 1M.

Ans:

RISC	CISC
① Lesser number of instructions	① Greater number of instructions
② Instructions pipelining & increased execution speed.	② Generally no instructions pipelining feature.
③ Orthogonal instruction set	③ Non-orthogonal instruction set
④ Operations are performed on registers only	④ operations are performed on registers or memory
⑤ Large no of registers are available	⑤ Limited number of general pur registers

Q No 6a) Functions of Optocoupler, I2C & IrDA - 10M

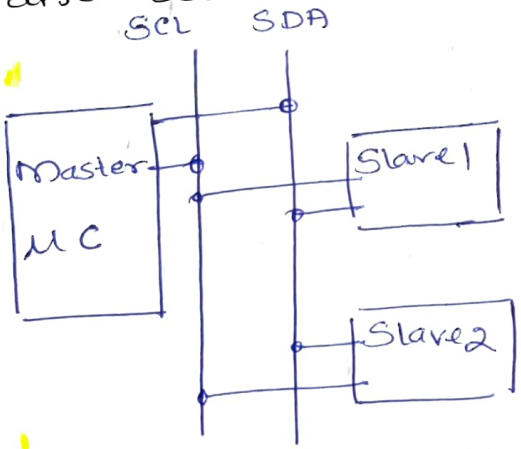
Optocoupler: It is a solid state device used to isolate two ports of a CKt. It contains a LED and a photo transistor in a single package, where it allows it to be used at both i/p side for sensing and at the o/p side for actuating the devices.



3M

2M

I2C: It stands for Inter Integrated CKt Bus, founded in 1980's by Philips Semiconductor. It consists of two BUS Lines Serial CLOCK - SCL and Serial Data - SDA.



Following sequence is followed during the communication

1. Master devices makes SCL to 1 & SDA to 0
2. It send 7 to 10 bit address of the slave device
3. The MSB of data transmitted first followed by till LSB. [During SCL is at 1]
4. For read/write Master device sends [1/0 bcs 4]
5. Slave device sends an ACK signal to master device for read/write operation [ACK = 1]
6. The communication is terminates with SDA = 1 and SCL = 0

4M

IrDA: It is a line of sight, half duplex. wifi technique used for data communication b/w the devices, Typical range is from 10 cms to 1m, data rates are 9600 Bps to 16 Mbps

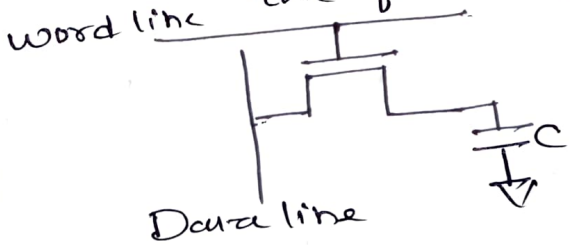
1M

Q.No 6B: EPROM, EEPROM, FLASH, DRAM, NVRAM and Sensor

1 X 6 = 6M

6M

Parameter	Description
① EPROM	Erasable Programmable ROM, stores the data bit by charging the floating gate of an FET, A quartz crystal window is used to program the memory by exposing it to UV rays for 20 to 30 mins
② EEPROM	It is Electrically racable Programmable ROM which can be programmed at register/Byte level. These can be erased/reprogrammed at ckt level.
③ FLASH	It uses a Floating Gate MOSFET. The earasing of the memory can be done at the Sector level/Page level without effecting the other sector/pages. Typical erasable cycle is 1000.
④ NVRAM	Non-Volatile RAM. which will be having a battery backup. It contains a SRAM cell and small battery ckt. The Life SPAN of NVRAM is 10 years
⑤ Sensor	It is a kind of transducer used to convert analog signal at the i/p side of an embedded System to required electorcal level.
⑥ DRAM	It stores the Data in the form of charge These are makeup of mos gates. It contains a typical MOSFET and capacitor for a one memory cell, where capacitor is used to store the data in the form of charge



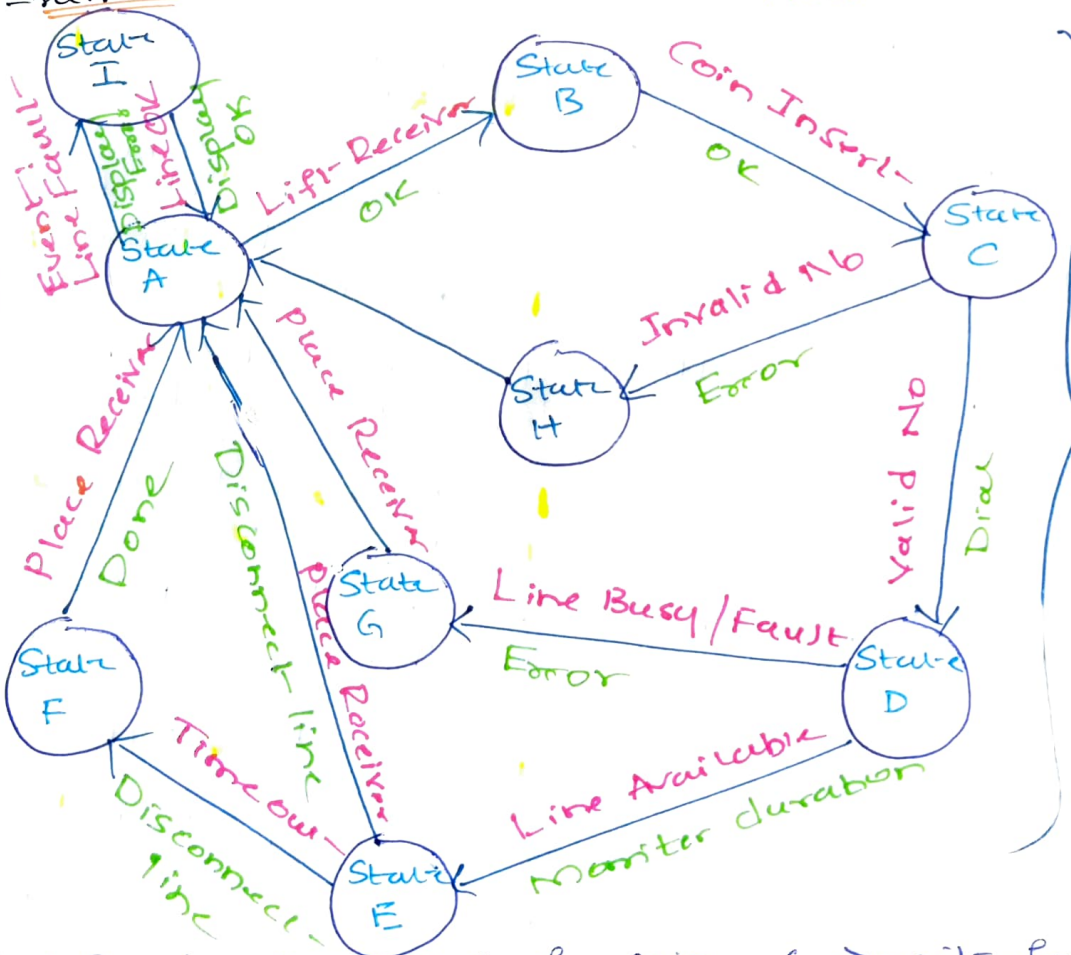
OK

Q.no 6c) Difference Between Embedded and General computing systems. 1X4=4M — 4M

General computing	Embedded Systems
① Contains a GPOS	② May or may not contain frequres OS
② Non-Time critical	② Time critical
③ Non-deterministics in execution/behaviour	③ Execution behaviour is deterministic
④ Flexible in Application /Features	④ Very Rigid

Q.no 7a) Coin Operated FSM telephone system, functions of states, and State transition diagram — 8M

EVENTS
ACTION



4M

A → Ready, B → Wait for coin C → wait for number
D → Dialing E → Call in progress F → Call Terminated
G → Unable to call H → Invalid i/p I → Out of order

OK

4M

① The calling process is initiated by lifting the receiver
② User needs to insert the coin
③ If line is busy coin is returned
④ If line connects, user has 60 sec to talk and after 45th sec before 59 sec, user need to insert one more coin for continuation
⑤ Place the receiver back
⑥ If line is FAULT, system goes out of order

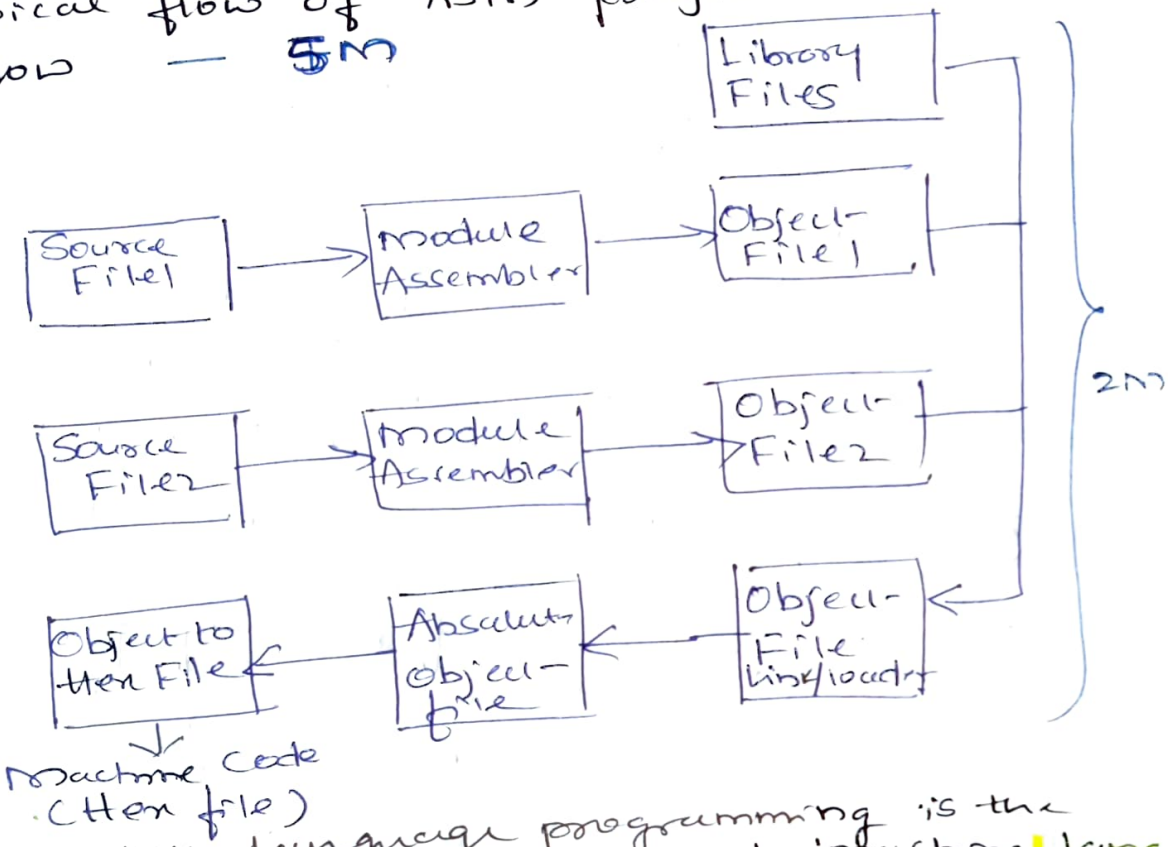
Q.no 7B : Any 5 characteristics of Embedded System — 1 X 5 = 5M

Characteristics	Description
① Application & Domain Specific	* It distinguishes ES from GPCS, Ex: Microwave oven Embedded ckt is not-replaceable by AC ckt.
② Reactive and Real Time	This is possible by using well calibrated Sensors & Actuators, Ex: Smoke sensor and Buzzer
③ Operates in harsh Environment	ES Application behaviour should not change w.r.t environmental changes, Eg: Cellphone network in Hot & cold condition should be same
④ Distributed	It should support various flexible options and results for the user, Eg Remote, Vending machine, ATM
⑤ Power Concern	It is expected to use less power / flexible in power consumption in an ES. Eg: Night mode in cellphone

OK

Q.no 7C With a Block Schematic, Explain the ALP based ES with its disadvantages 7M

Typical flow of ASM program is shown below — 5M



* Assembly language programming is the human readable notation of 'machine language', whereas machine language is processor understable language
* It uses alphabets, numbers and special characters.
* Typical format of assembly language is <Lable> Opcode Operands comment. where Lable is and comments are optional, Opcode is an operational code which acts on the Operands, Example 3M

```
mov A, #30h
 ↑      operands
opcode
```

* Library files: It contains the target board selection options to set frequency, RAM, ROM area selection, predefined logic opcode
* object-files: These are the various files of a project created by users.
* linker/loader: Linker like ASL for assembly

OK

language programs to link all .obj files of a project with an Absolute Object File. Of a project file, which are converted to Hex file which can be used to execute and assemble all as individual .obj files. as executable files

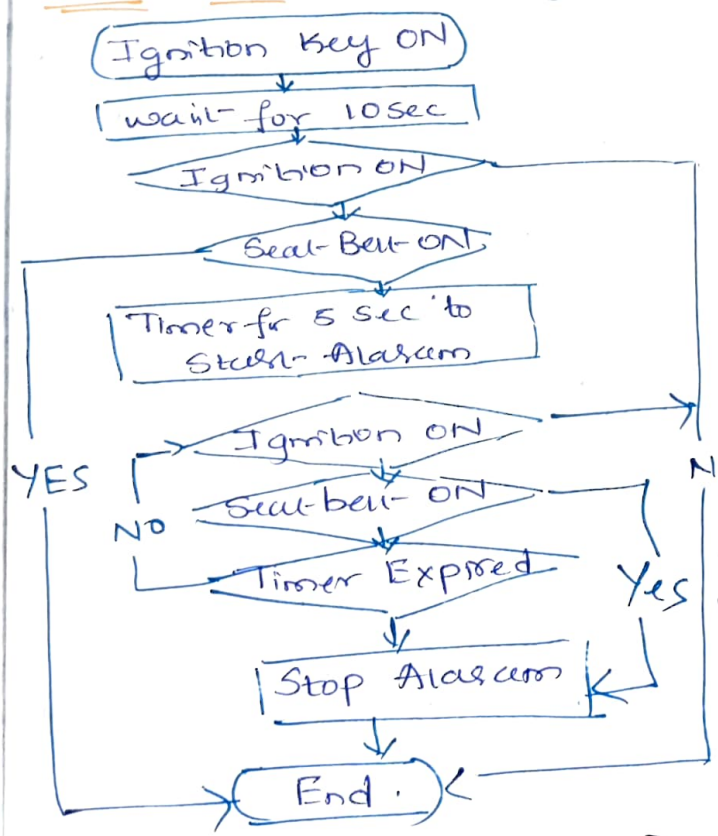
Drawbacks — 2M

① High Development Time: It requires more lines of assembly code for performing an action, which can be implemented using a single instruction in High level language like C

② Non-portable: Target applications written in assembly instructions are valid only for that particular family of processor Eg like Intel x86 which can't be used for another target Processor / controller.

Q.no 8A

Seat Belt Warning System program Model — 8M




```
#define 'ON 1
#define OFF 0'
#define YES 1
#define NO 0
void seatbelt warn()
{
wait = 10sec();
if (ignition key() = ON)
{
if (check_seat_belt_() = OFF)
{
Set timer(5);
Start alarm();
while((seat belt() = OFF) &&
      (ignition key() = OFF) &&
      timer_expire() = NO);
Stop alarm }}}
```

5 M

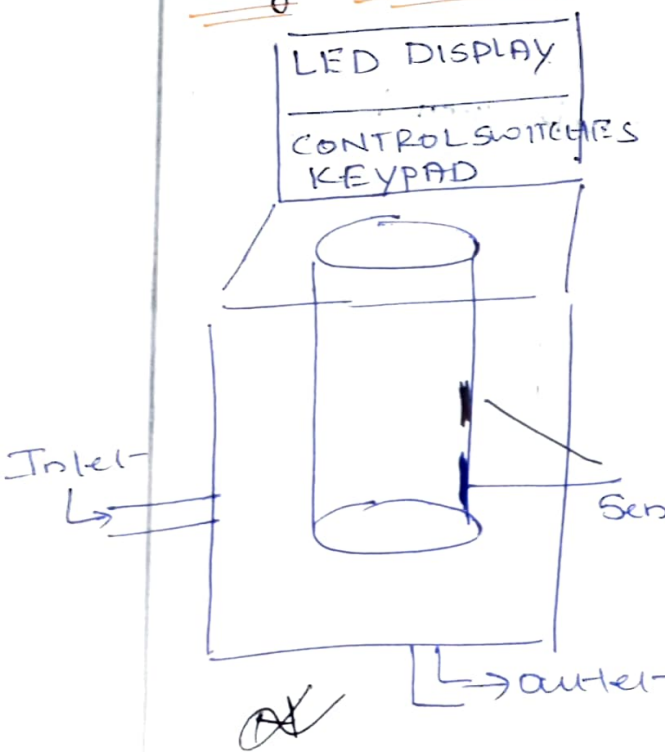
3 M

OK

Q.no 8b Five Operational Quality attributes → 5M
1 X 5 = 5M

Attributes	Description.
① Response	* measure of quickness of the system * It gives us how fast system is reading input and executing them * Ex: Sensor
② Throughput	* Deals with efficiency of a system * no of products produced per unit time defines the capacity
③ Reliability	* measure of how much we can depend on the proper function of a system or no of times system getting failed
④ Security :	* It is the measure of Confidentiality, Integrity and Availability of the system
⑤ Safety	* It deals with the possible damages that can happen to the operators, public and environment due to the emission of radioactive or hazardous like e-waste.

Q.no 8c Functional Block diagram and working of a washing machine - 7M marks



Working

① Through control switches WM is set to the different clock cycle

② Through inlet water is allowed to flow in to the machine

3M ③ Sensor: Temperature sensor senses & set the temperature depending on wash cycle, water level sensor indicates the level of water to processor

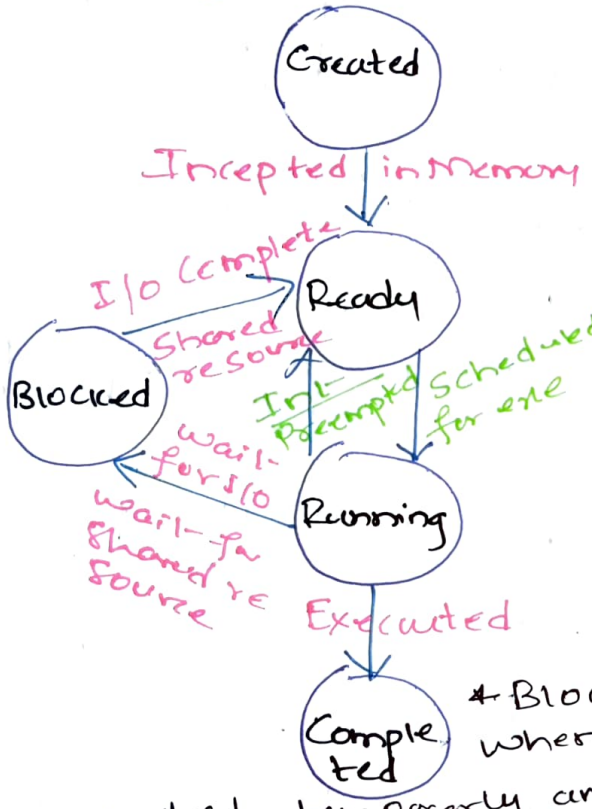
④ Once the washing is over the water is dispensed through outlet

← 4M →

Q.No 9a) State Transition Diagram, Structure of a Process, and Memory organization of a process 10M

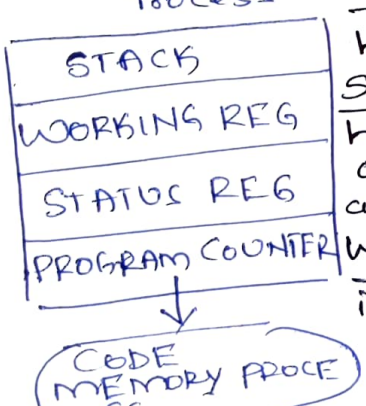
4 + 3 + 3 = 10M

* State Transition Diagram & Explanation [2+2 = 4M]



* The parameters mentioned in circle indicate various states of a process [Created Ready, etc]
* Any process / task gets initiates with created state, where they are not allocated with any of the system resources
* 'Ready' state indicates the allocation of resource like memory / IO is ready to be used by the initialed task
* 'Running' indicates the presently a task is running with resources of ES
* Blocked state is reached, when a running process gets pre-empted temporarily and once it is completed it will be taken back to Ready state followed by running state and completion.

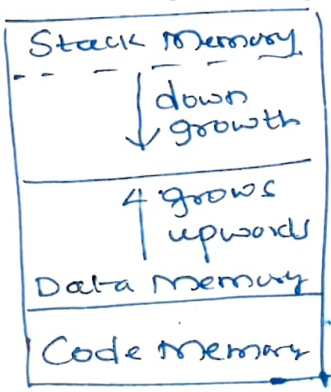
* Process Structure [1 + 2]



Let us understand this by considering an example of PDF Reader, here
STACK! It is part of memory, which holds the recently running as well as currently running address of process and gets updated,
WORKING REG! It is the local address in ROM where the actual PDF application is installed
STATUS REG! It indicates the response of an application which is selected (PDF reader)
PROGRAM COUNTER! It points to the address of the next process to be executed while present process is running

Q.no 9a) continued

Memory Organization of a process [1M+2M=3M]



Here, Stack Memory : Holds the currently running / recently used process address, as no of process increase, It gets decreases

Data Memory : It's the RAM, which is utilized by each process and it grows upwards as process increase

Code Memory : Address of the process where it is installed or runs.

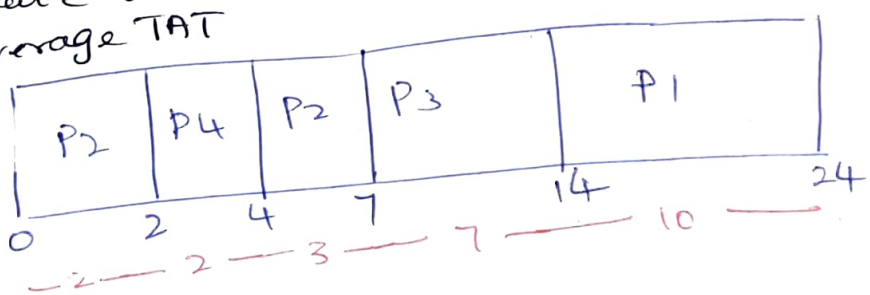
Q.no 9b) Pre-emptive SJF Scheduling / Shortest Remaining Time (SRT) with example [4+6] 10M

4M

* SJF / SRT is a kind of scheduling method
* here it allows a process to be stopped (pre-empt) by another process
* first it finds all the process starting from least completion time to maximum completion time
* It allows a preemption in the presently running process, if a new process arrives with least completion time than that of a presently running process and selects it and completes it

2M

Example: Let us consider 3 process P1, P2, P3 with estimated completion time of (10, 5, 7) msec respectively enters the ready queue. A new process P4 enters process with estimated completion time of 2msec. enters the ready queue after 2msec Calculate wait time, Average wait time, TAT and Average TAT

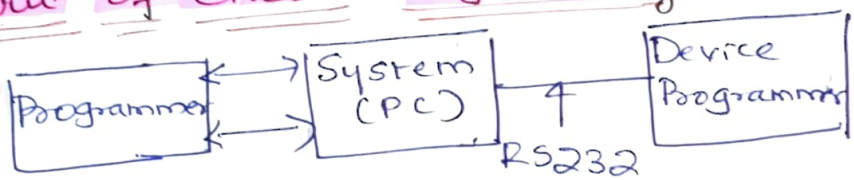


Wait Time for	Avg wait time = All wait time / no of process	TAT = Time spent in ready queue + Execution time	Avg TAT = TAT of all / no of process
$P_3 = 0ms + (4-2)ms = 2ms$	$(P_4+P_2+P_3+P_1)/4 = (0+2+7+14)/4$	$P_2 = 7ms$	$(7+2+14+24)/4 = 47/4$
$P_4 = 0ms$	$= \frac{23}{4}$	$P_4 = 2ms$	$= 11.75$ msec
$P_3 = 7ms$	$= 5.75$ msec	$P_3 = 14ms$	
$P_1 = 14ms$		$P_1 = 24ms$	

4M

Q.no 10a) Out of circuit Programming and In System Programming [ISP] [5+5] 10M

Out of Circuit Programming



1M

* This is a debugging method which is performed outside the target board.
* The CPU of target board is taken out of the CKT
* Following sequence gives the steps of debugging

3M

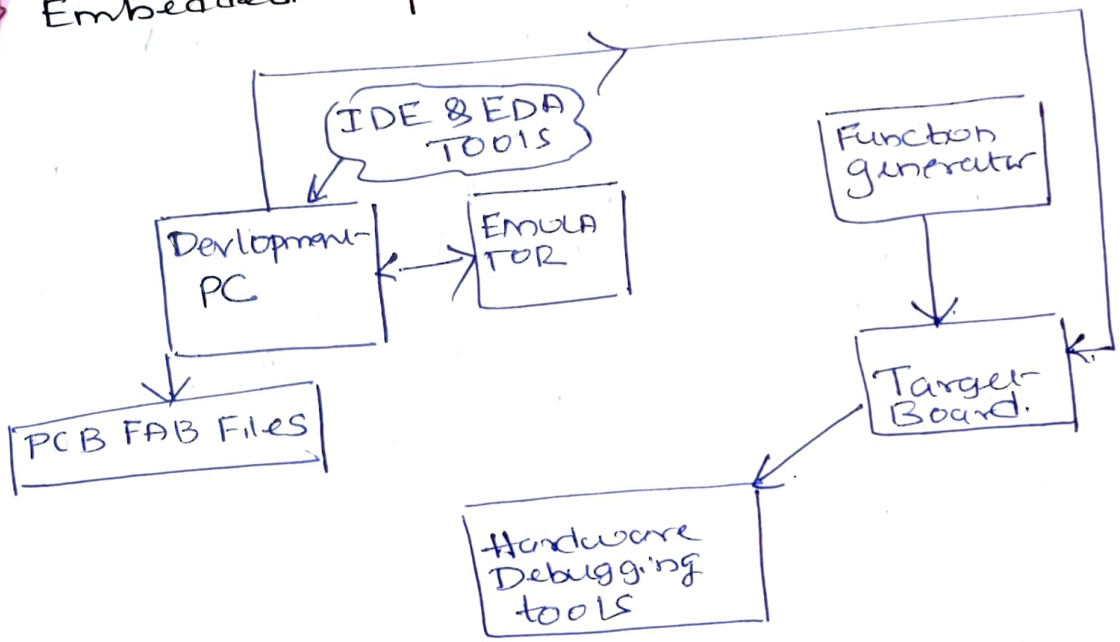
1. Connect the programming device to the specified port of PC, Power up the device
2. Execute the programming utility on the PC and ensure proper connectivity is established b/w PC and board
3. Insert the device to be programmed in to the open socket as per the insert diagram specified by the vendor
4. Wait till completion of the programming [Check LED Indicator "Green" for sucess and "RED" for Error]

In System Programming (ISP) [5M]

- It is done within the system, it means firmware is embedded in to the target device without removing it from the target board.
- Target device must support ISP feature.
- Chips supporting the ISP generates the necessary programming signal internally,
- The interfacing can be carried out by utility program running on PC through Serial/Parallel /USB port.
- Popular protocols used are JTAG and SPI.
- During the debug session target boards are configured in ISP modes.
- The device receives commands and data from the host, erases and reprograms code memory according to the received command.

5M

Q.No 10b) Embedded System Development Environment — 10M



4M

Embedded System Development Environment —

Q no 10 b continued

* The development environment consists of a PC which acts as core of development environment.
* IDE and EDA Tools are used to simulate the required programs to be run/checked with target board.
* An Emulator hardware is used to debug the Target board., Signal sources like Function generator are used with emulator to check the real time o/p on the signal generator.
* Sometimes Digital CRO, multimeter, Logic Analyser etc are also used in Emulation process
* IDE and EDA tools are selected based on the target hardware development boards. These tools can be either freeware / Licenced versions
* IDE also includes the IDE softwares (like Xilinx, Keil, etc) which will be having in built Text editors, Cross Compilers (for same platform developments) and compiler for cross platform developments).
* Linker and Debugger are also used to link files from different formats like .obj to .hex plat form and debug them properly
* GUI based IDEs provide a visual Development Environment (VDE) with user interactions.

GM