Model Q.P Scheme & Solution.

Subject: Embedded Systems

Code : 18EC62

Prepared by: Prof. Nikhil A. Kulkarni.

NOTE: As, the model Q.Ps for 18EC62 are not available, Q.P has been compiled by referring to the previous scheme/year VTU Q.P. for the mentioned Subject

Prepared and compiled By.

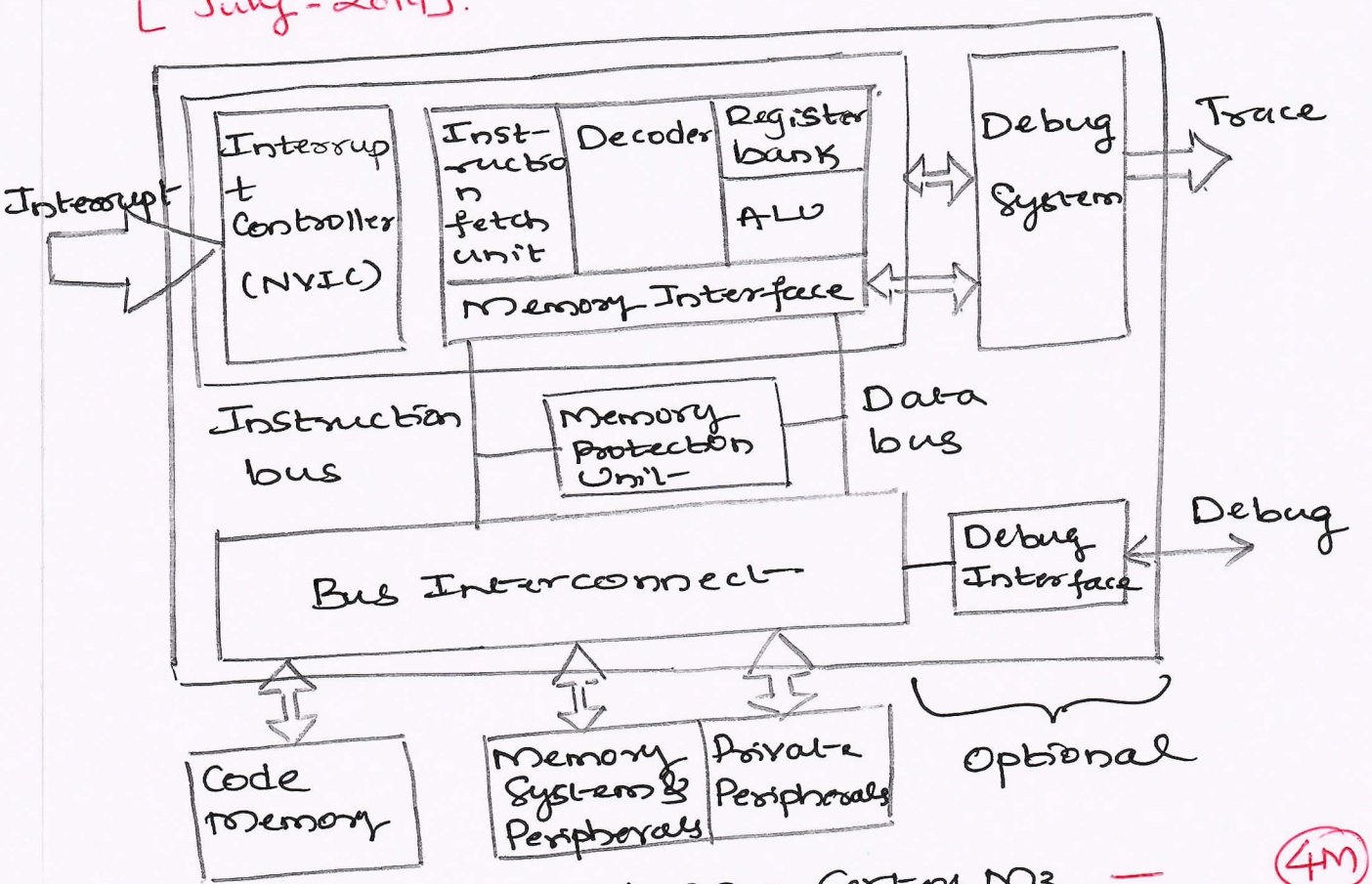
Prof. Nikhil Kulkarni

24.07.2021
Head of the Department
Dept. of Electronic & Communication Engg.
KLS V.D.I.T., HALIYAL (U.K.)

Prof. Nikhil Kulkarni

Module-1

Q 1a] Explain the architecture of ARM Cortex-M3 processor with neat diagram. — 8M [July-2019].



Architecture of ARM Cortex M3 — 4M

— 4M

Explanation

* It is a 32-bit microprocessor
* All registers and even all instructions are 32 bits
* The processor is built (designed) as per Harvard Architecture [It has seperate instruction and data bus]
* It supports both little endian and Big endian memory system.
* It has totally 13, 32-bit general purpose registers, (R0-R12)

Q1a] Contd...

* R13 is the Stack pointer (32 bit register), totally two SP are available,
  i) MSP (Main Stack Pointer)
  ii) PSP (Process Stack Pointer).
* R14: It is also a 32 bit register, refered as Link register
* R15: Program counter, which is also a 32bit register, Generally PC is used by the CPU to point to the address of the next instruction to be executed
* It supports Privileged and User levels for handler and Thread modes.
* The Interrupts are managed by a feature called NVIC [Nested Vector Interrupt Controller]
* 4 GB of memory space, i.e 2^{32}.
* It support 256 Exception types.
* All instructions can be used w.r.t all the registers.

Q1b] With Neat Diagram, Explain the Operation mode and privilege levels in cortex M3. -SM [July-2019].

P.T.O

Q1b] Contd...

* operation mode and allowed Privilege levels in cortex - M3. (2M)

When running an exception	Privileged: Handler Mode	User:
When not running an exception	Thread Mode	Thread Mode

Figure 1

* Allowed operation mode transitions (2M)

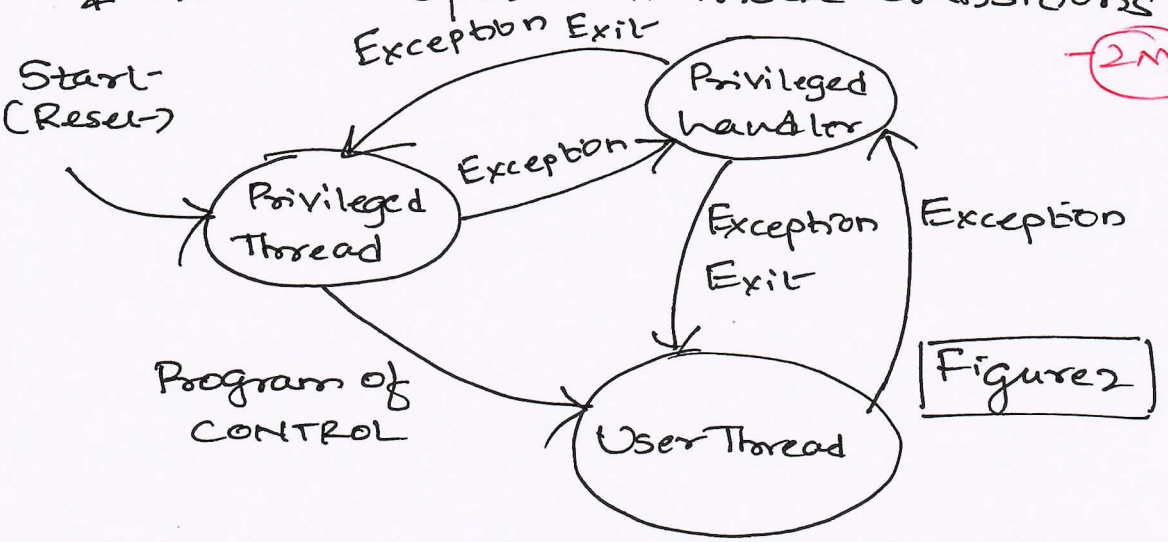


Figure 2

→ Above two figures shows the various modes of operations with levels and their transitions

→ From Fig 1, we can notice that - , If the processor is in Privileged level mode, then it can handle the exceptions {refered as Handler mode}, if not handling {it is refered as thread mode}. (4M)

→ If it is in User Level, then it cannot handle any exceptions, {Thread mode}.

→ Figure 2, shows various transitions between the modes and levels

Q.2a] What is Stack? Explain PUSH and POP Operation with the help of a near- diagram [7M] [July 2019].

* Stack: It is the part of the memory in any µp/µc, which is used to store the data temporarily, In particular during an exception handling process. — 1M

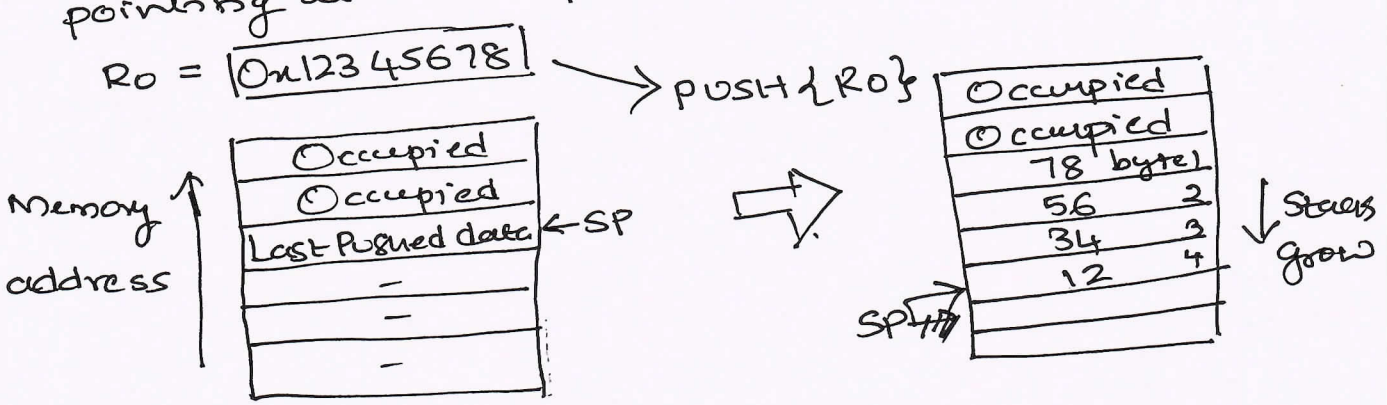
* The instructions PUSH and POP are used along with LDM/STM to access the SP, where PUSH is used for transfering data from memory/registers to the stack, and POP is used for retriving the contents from STACK to respective register/location

* In, Cortex M-3 the Stack is fully descending one, i.e it grows towards the lower address of the memory with 4 byte space for each PUSH and grows upward by 4bytes for each POP instruction 2M

* Example: [You, can give any relavent examples of your choice].

PUSH operation: [2M]

Assume that- R0 = 0x12345678, SP is pointing at- last- pushed data.

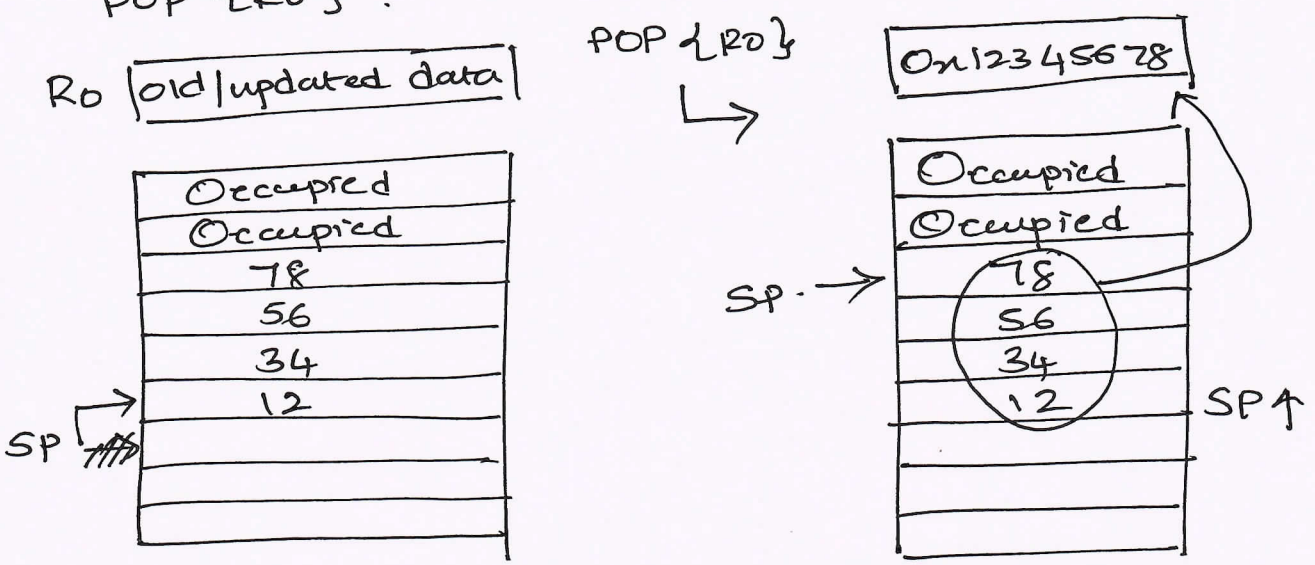


Q.2a] Contd..

POP operation [2M]

—X—

POP {R0}.



Q2b] Explain in detail about the Special registers used in ARM Cortex M3 processor. [9M] [July 2019].

There are 3 special Registers available in Cortex-M3 Processor, They are

i) Program Status Register (PSRs)

ii) Interrupt Mask Registers (PRIMASK, FAULT MASK and BASEPRI)

iii) Control Register (CONTROL).

All, these special registers can be accessed through MSR and MRS Instructions

Consider

i) Program Status Register (PSRs). — 3M for explanation

The PSRs is subdivided into 3 Registers,

a) APSR (Application Program Status Register)

b) IPSR (Interrupt Program Status Register)

c) EPSR (Execution Program Status Register)

Q2b] Contd...

* Out of these 3 PSRs, only APSR can be READ/WRITE, but IPSR and EPSR are READ only.
* Instructions used for READ/WRITE are

  MRS R0, APSR or IPSR or EPSR ; Read.

  MSR APSR, R0 ; Write.

* PSRs is a 32 bit register, which is shown below-

	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																ExceptionNo
EPSR						ICI/IT	T			ICI/IT						

* Combinedly, It is refered as xPSR

* Descriptions of the flags

N → Negative Flag: It is 1, if MSB of result is 1 else 0

Z → Zero Flag: It is 1, if result is 0, else 0

C → Carry/borrow: It is 1, when an overflow occurs i.e (>32bits) Or, when a lower number subtracts a higher number.

V → Overflow: It is 1, when result exceeds 32 bits, else it is 0.

Q → Saturation flag

ICI/IT → Interrupt and IF-THEN instruction Status bit.

T → Thumb state

Exception number: Indicates the exception number handled by processor

Q.no2b] Contd...

ii] PRIMASK, FAULTMASK and BASEPRI: -3M

These registers are used to disable few/all exceptions

→ PRIMASK: It is a 1-bit register, If SET, It allows NMI and hard fault-exception. Default value is '0', which indicates no masking is possible

→ FAULTMASK: It is a 1 bit register, If SET, It allows only NMI, Default-value is '0', which indicates masking is not-possible.

→ BASEPRI: It is a register which can be up to 8bits. If SET it can disable an interrupts of the same or lower levels, with disturbing higher level interrupts.

iii] CONTROL Register: -3M

It is used to define privilege levels and SP selection. It is a 2 bit register

CONTROL [1]: It provides the status about the STACKie. If 1 - Alternate Stack is used and, if 0 → Default-[MSP] is used

CONTROL [0]: If 1 indicates User State in thread mode and 0 indicates Privilege in thread mode

Prof Nikhil Kulkarni

Module-2

Q3a] Explain Shift and rotate instructions available in ARM Cortex M3. Why is there rotate right instruction but no rotate left instruction. 8M [July 2018].

* Cortex M3 supports various Shift and rotate instructions. In Shift operation, the bits are Shifted to right/left w.r.t the value Specified by the source or immediate operand and in rotate instruction the bits are rotated to the right also.

* Shift and Rotate Instructions

[At least you need to describe 4 instructions in each type]

Shift operation [

1] Arithemetic Shift instructions:

a) ASR [Arithemetic Shift Right].

ASR Rd, Rn, #imm ; here Rd is loaded with the right shifted version of Rn Specified by imm value

2] Logical Shift right

a) LSR Rd, Rn; Rd ← Rd >> Rn i.e Rd is loaded by logically shifted version of Rd by the value specified in Rn.

3] Logical Shift Left:

a) LSL Rd, Rn; Rd ← Rd << Rn, here it is logically Shifted left

4] Rotate instruction

a) Rotate Right [ROR]; ROR Rd, Rn; here Rd is rotated by number mentioned by Rn

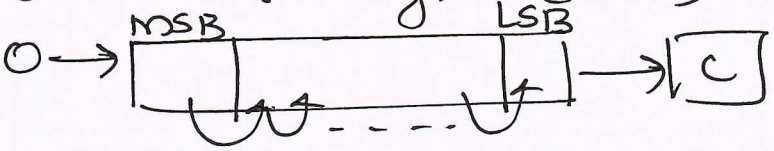
2M

Q3c] Contd...

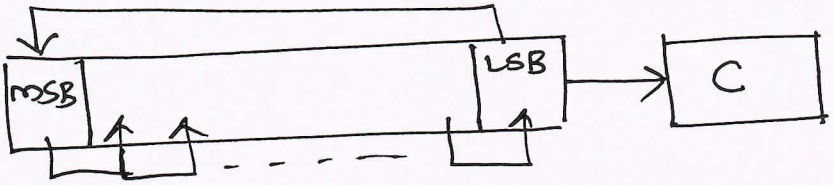
Shift and rotate instruction method.

1. Logical Shift Left (LSL)

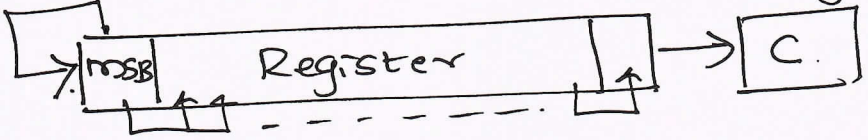
2. Logical Shift right (LSR)



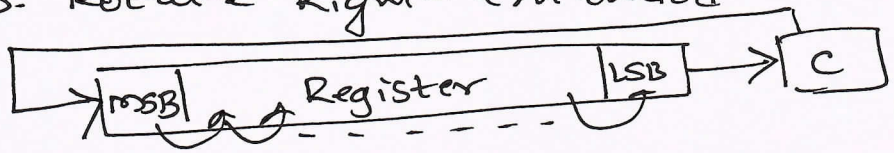
3. ROR {Rotate Right}.



4. ASR {Arithmetic Shift right}.



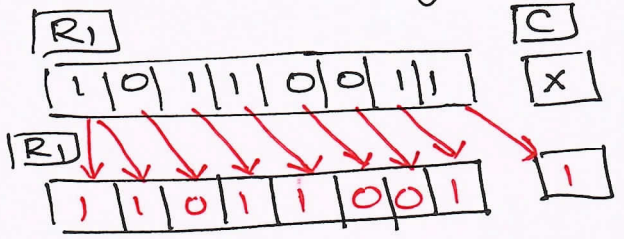
5. Rotate Right extended



2M

Example

Consider ASR instruction, with $R_1 = 10110011$ and $R_2 = 1$ then after performing ASR R_1, R_2 we get -



2M

*The rotate left operation can be replaced by a rotate right operation with different offset. i.e. A rotate left by 4-bit operation can be written as a rotate right by 28 bits. which gives the same result with same execution time

2M

Q3b] Explain the following instructions with suitable examples i) BFC, ii) SXTH, iii) UBFX and iv) RBIT →[8M] [July 2018]

← 2 marks For each instruction explanation with example →

Consider

iv] RBIT: Reverse bit order in 32 bit word

example: RBIT R7, R8; Reverse bit order of value in R8 and write the result to R7. 2M

Assume that R8 = 11 00 33 22

After execution R7 = 44 CC 00 88

31 16 15 0

i] Bit Field Clear BFC:

Syntax: BFC {cod} Rd, #lsb, #width., It clears a bit-field in a register. It clears width bits in Rd, starting at low bit position LSB and other bits in Rd are unchanged. 2M

Example: BFC R4, #4, #12; Clear bit 4 to 15 (12 bits) of R4 to zero

ii] SXTH: Sign extend a half word.. i.e. First, it rotates the value from Rm right by 16 bits and extracts bits from [15:0] and sign extends to 32 bits 2M

Example: SXTH R4, R6, if R6 = 0x55AA8765 then after execution R4 = 0xFFFF8765

iii] UBFX: It is Unsigned bit field extract instn.

Syntax: UBFX.W <Rd>, <Rn>, <#LSB>, <#width>

It extracts a bit field from a register starting from any location mentioned by LSB with width Zero extends it and results in destination 2M

Ex: LDR R0, =0x5678ABCD AFTER EXE R1=0x000000BC

UBFX.W R1, R0, #4, #8

Q4a] Write the memory map and explain memory access attributes in Cortex M3 [8M] [July 2018] [Not in New Syllabus]

Q4b] Analyse the following instructions and write the content of the registers after the execution of each instructions.

Assume: R8 = 0x00000088, R9 = 0x00000006 and R3 = 0x00001111

i) RSB.W R8, R9, #0x10; ii) ADD R8, R9, R3;

iii) BIC.W R6, R8, #0x06; iv) ORR R8, R9

[8M] [July 2018].

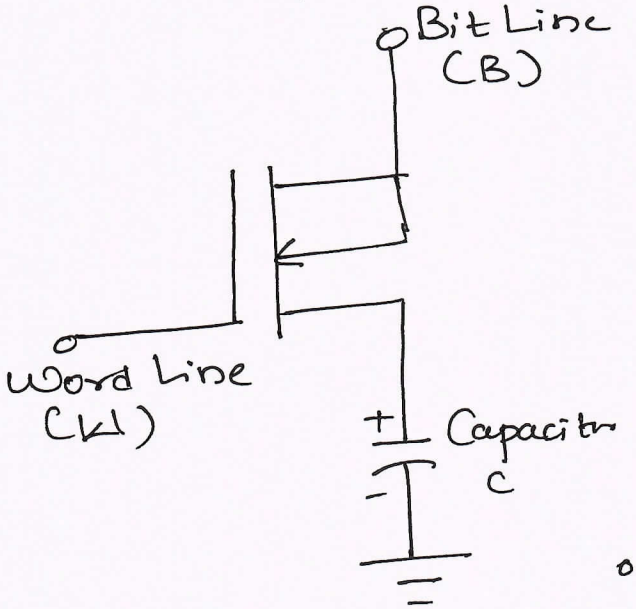
Instruction | Register Contents & Analysis

Instruction	Before	After	
① RSB.W R8, R9, #0x10	R8 = 0x00000088 R9 = 0x00000006	R8 = 0x0000000A	R8 ← 10 − R9 } 2M
② ADD R8, R9, R3	R8 = 0x00000088 R9 = 0x00000006 R3 = 0x00001111	R8 = 00001117 R8 ← R9 + R3	} 2M
③ BIC.W R6, R8, #0x06	R6 = ? R8 = 0x00000088	R6 ← (R8) AND (~0x06) R8 = 0x00000088 ~0x06 = 0xFFFFFFF9 ∴ R6 = 0x00000088	} 2M
④ ORR R8, R9	R8 = 0x00000088 R9 = 0x00000006	R8 ← R8 OR R9 R8 ← 0x0000008E	} 2M

1000
0110
OR = 1110 = E

Q5a] Define the term RAM. Mention different types of RAM and explain any one with neat circuit diagram. [6M] [Sept-2020].

* RAM stands for Random Access Memory, It is also known as Data Memory, This particular memory allows both Read/Write operation. 1M
* SRAM, DRAM and NVRAM are the different RAM types.
* Let us consider the operation of D-RAM 1M
* D-RAM stands for Dynamic RAM. A DRAM Cell consists of a MOSFET and a Capacitor
* Following Figure shows the D-RAM Cell Set-up/Circuit



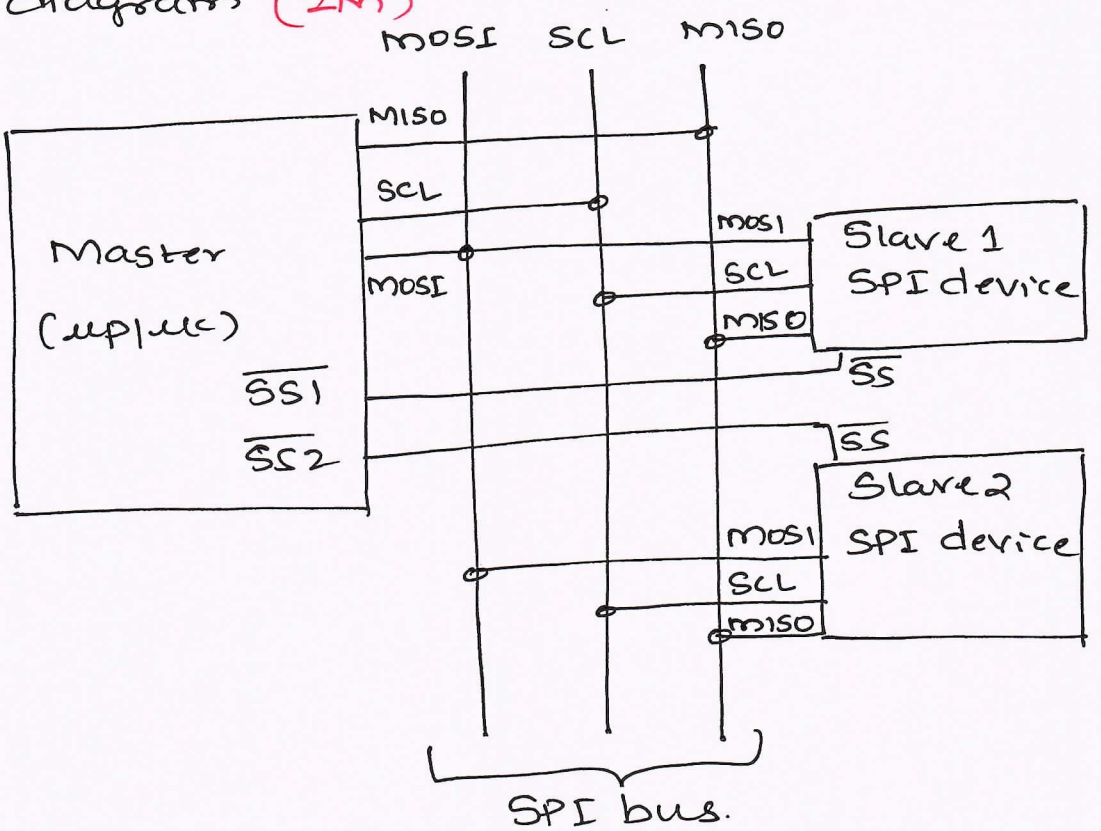
→ Write Operation (2M)

- Initially (WL) line is kept at HIGH Level, this will turn on the MOSFET (nMOS).
- Next, The value to be written is carried out by pulling (B) Line to either HIGH or LOW
- For, this particular value Capacitor gets charged, hence write operation is carried out.

→ Read Operation (2M): Initially Word Line is pulled HIGH, then the Charge on Capacitor reflects (can be Read through the Bit Line). The value depends on the previous charge.

Q5b] With a neat Interfacing diagram explain the SPI bus [6M] [Sept-2020]

* Consider the following SPI bus Interfacing diagram (2M)



* SPI Stands for (Serial Peripheral Bus), It is a synchronous bi directional full duplex four wire serial interface bus.

* SPI uses four signals lines for communication.

MOSI: Master Out- Slave In, It carries the data from master to slave (1M)

MISO: Master In Slave Out, It carries the data from Slave to Master. (1M)

SCLK: (Serial Clock) which carries the clock signal

SS: Slave Select, It is an active low signal, (1M) which is used by the master to select a particular Slave for communication. (1M)

* Communication is initially established by MASTER through SS line, with MOSI HIGH, and If MISO is HIGH SS line is D.

Q5] Bring out the difference between FPGA and CPLD [4M] [Sept-2020]

1-Marks For each difference

CPLD	FPGA
1) Supports less amount of Gate density / Logic density	1) Supports highest amount of logic density
2) Not Supported	2) Supports built in hard-wired support for processor
3) It offers very predictable timing Characteristics	3) It is less supported for time critical features
4) Inexpensive	4) Expensive.
5) Ex: Xilinx CoolRunner	5) Xilinx Virtex.

Q.6a] Differentiate between RISC & CISC -4M [July2019]

RISC	CISC
1) Supports Orthogonal type Instruction set	1) Non-orthogonal instruction set.
2) Supports Harvard Architecture	2) Supports both Harvard and Von-Neuman Architecture
3) Less number of instructions are present	3) More number of instructions are present
4) Single, Fixed length instructions	4) Variable length instructions.

1M for each difference.

Prof. Nikhil Kunkeri

Q6b] Explain the different 'on board' communic-ation interfaces in brief. [8M] [Jan-2020]. 15

* On board communication interfaces are used to connect and communicate among the different peripherals present on the same board / circuit.

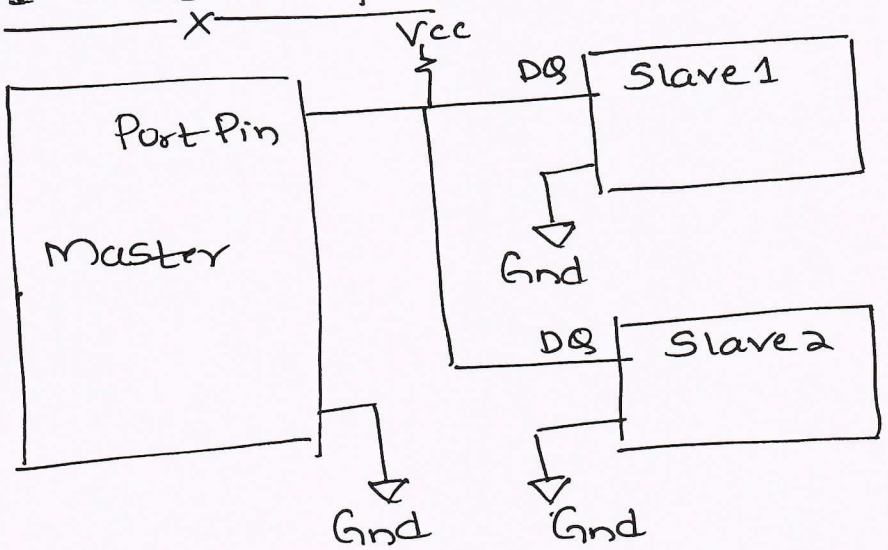
* There are 5 on board communication interfaces available, which are I^2C, SPI, UART, 1-wire interface and parallel interface.

Here you can explain any 2 interface in detail, each carries 4 M.

Consider

① SPI {ref Q.no 5b} 4M

② 1-Wire Interface:



* It is an asynchronous half duplex communication protocol developed Maxim Dallas Semiconductor

* It is also known as Dallas 1-Wire protocol.

Q6b contd....

The sequence of operation for communicating with a 1-wire slave device is listed below.

1. The master device sends a 'Reset' pulse on the 1-wire bus
2. The slave device present on the bus responds with a presence pulse
3. The master device sends a ROM command. This addresses the slave device to which it wants to initialize the communication. 4M
4. The master device sends a read/write function command to read/write internal memory or register of the slave device.
5. The master initiates a Read data/Write data from the device or to the device.

Q6c] Write a note on embedded firmware 4M [Sept 2020]

* Embedded Firmware is the pre-installed program or OS which is not alterable.
* There are two approaches for this
  i) The super loop based approach:

This is used for the applications, where time is not the constraint to complete the given task. The firmware execution flow is 2M

a) Initialize the common parameters and perform initialisation.
b) Start the task1
c) execute task2
d) ....
:
execute last task
Jump back to the task1 and repeat

Q.6C] contd....

Example

```
Void main ()
{
configurations ();
Initialisation ();
while (1)
{
Task 1 ();
Task 2 ();
  :
Task n ();
}}
```

ii) Embedded OS based approach:

* It requires OS
* It can be either General purpose OS or a RTOS
* In case of GPOS, the applications are installed on top of OS and used, example like printer drivers
* RTOS based design approach is used in embedded products demanding RTOS.
* RTOS allows flexible scheduling of system resources like CPU and memory

2M

Prof. Nikhil Kulkarni

Module-4

(18)

Q7a] Explain the term quality attributes in an embedded system development content. What are the different quality attributes to be considered in an embedded system design. [8M] [July 2018].

→ Quality attribute (Q.A) is the performance measure of a embedded system. There are two types

I] Operational Q.A [4M] {Any 4 or all 6}

i) Response: It is the quickness of the system with which it responds to an input.

ii) Throughput: It is related to the efficiency of the system, From this the number of events for a given amount of time can be analysed

iii) Reliability: It is related to the correctness of a system, i.e. mean time between failures to the mean time to repair i.e MTBF & MTTR.

iv) Maintainability: It indicates the necessary steps/tools used to recover a system from failures.

2] Non-Operational Attributes: 4M {Any 4 or all}

i) Testability and debugability: Once the embedded system is designed, before marketing, it will go through various tests and debugging methods to ensure the quality of the product.

ii) Evolvability: The demand/design of an E/S should meet the current technology.

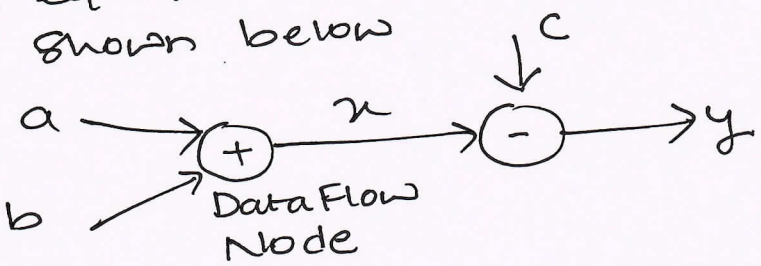
iii) Portability: It is the performance parameter which indicates the variation in performance w.r.t. different operating environment

iv) Per Unit and total cost: It is related to the cost/unit for a single and bulk orders

Q7b] Explain DFG and CDFG models in an E/S. design [8M] [July 2018]

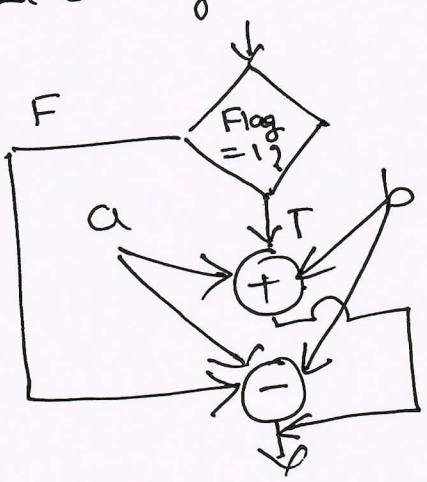
→ DFG {4M}

* It translates the data processing requirements in to data flow graph
* It is a visual model in which operations like [+, −, *, /] on the data is represented using a block [Circle] and data flow is represented using arrows. An inward arrow to the process / Circle indicates input and outward arrow indicates output.
* The eqn $x = a + b$ and $y = x - c$ is represented as shown below



→ CDFG {4M}

* A control cond^n can be added to DFG .. i.e In a basic CDFG, we have two types of nodes. decision nodes and data flow nodes
* Example: If flag=1, $x = a + b$; else $y = a - b$ can be represented by CDFG as follows..



Prof. Nikhil Kulkarni

20

Q8] What is hardware Soft-ware Codesign? Explain the fundamental design approaches in detail. [10] [Jan 2020]

→ During the design of an embedded s/m. both hardware and software should be compitable with each others performance.

→ Some of the fundamental design approaches are

i) Selecting the model 2M A model is a formal System consisting of objects and composibon rules. It may be i/p and o/p model, power con-sumpbon model etc

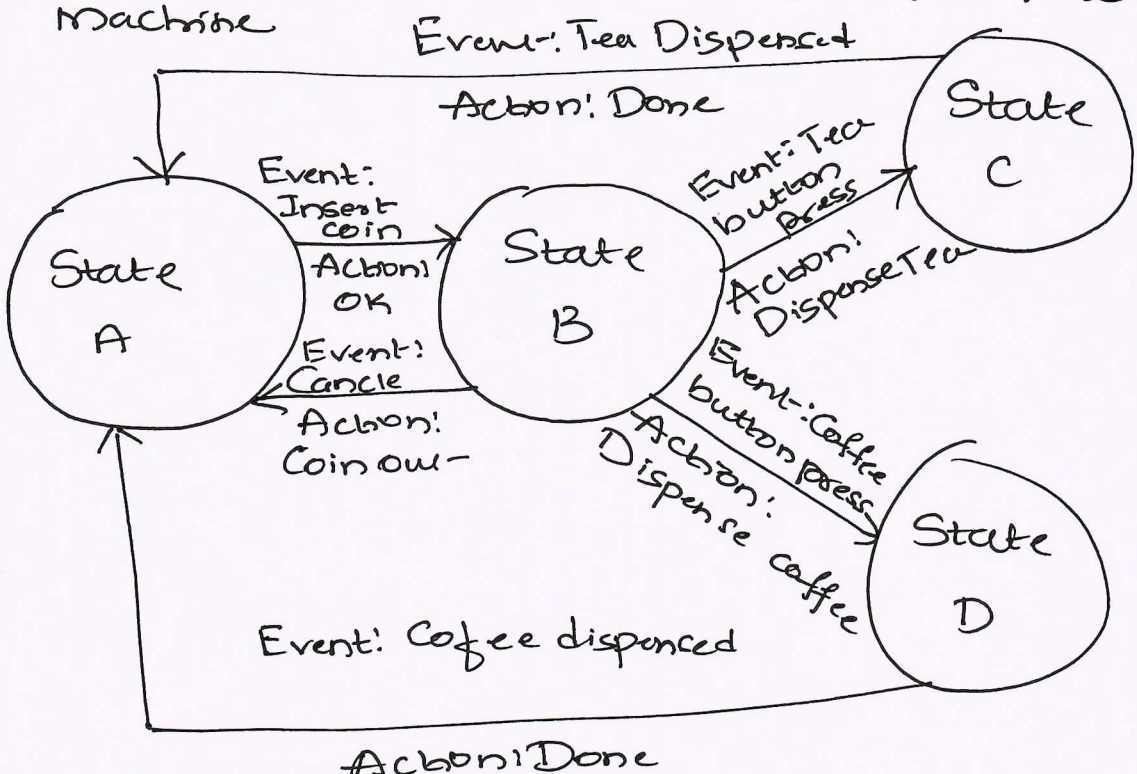
ii). Selecting the architecture 2M: It specifies, how a design is going to be implemented interms of nomber and types of components and interconnecbon among them, Few fa-mous architectures refered are RISC, CISC, VLIW etc

iii) Selecting the language 2M: A programming language captures a computabonal model and maps it into architecture, multiple programming languages like C, C++ etc are used for software implementabon.

iv) Partitioning system requirements into hardware and software: 2M
Before assembling hardware with related soft-ware, each module is tested with its related hardware and soft-ware process.

Q8b] With FSM model, explain the design and operation of automatic tea coffee machine. [6M] [Jan 2020]

→ Consider the following diagram of FSM model for Automatic Tea/Coffee Vending Machine



State A: Wait For Coin; State B: Wait for user i/p

State C: ~~Wait For~~ Dispense Tea / State D: Dispense Coffee

The FSM sequence is explained below

i) User needs to insert coin (State A), if it is cancled it returns to State A from B.

ii) State B is the option for users to select Tea/coffee, then corrosponding State C / State D is attained respectively

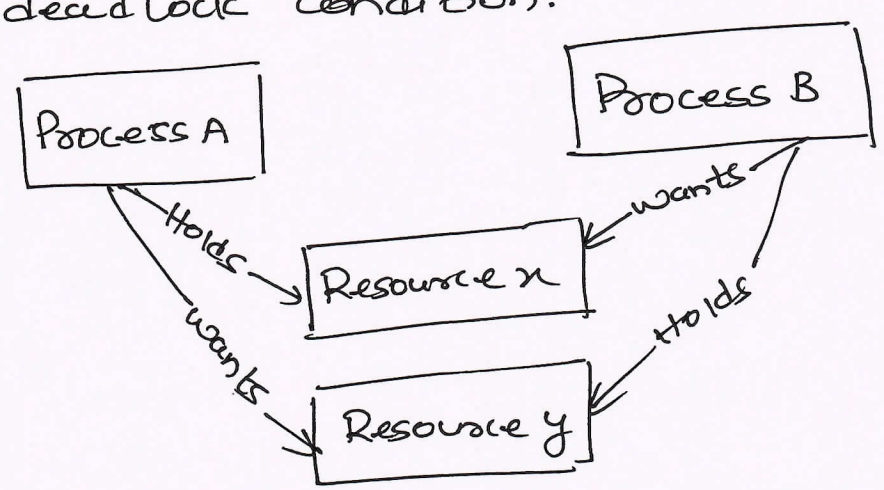
iii) Once either State C / State D transition is made it comes back to State A

Q9a) Explain the concept of 'Deadlock' with neat diagram. Mention the different condition which favours deadlock situation [8M] [July 2018]

→ Deadlock can be defined as the permanent blocking of a set of processes that either complete for the system resources. It can occur on sharable resources like files, printers, database, disks etc

→ In simplest form, Deadlock is the condition in which a process is waiting for a resource held by another process which is waiting for a resource held by the first process.

→ Consider the following figure, which depicts the deadlock condition.



Process A holds the resource x and it needs a resource y which is held by Process B. Hence the Process A is in deadlock for resource y and Process B is in deadlock for resource x.

Till here 2 to 3 marks can be alloted

Q9a] contd...

The different conditions favouring a deadlock situations are listed below.

i) Mutual Exclusion: This is the criteria that only one process can hold a resource at a time

ii) Hold and Wait: The condition in which a process holds a shared resource by acquiring the lock controlling the shared access and waiting for additional resources held by other process.

iii) No Resource Pre-emption: This is the criteria that OS cannot take back the shared resource by a process.

iv) Dead lock handling: It is decided by type of OS running on the embedded system. and it can be non-uniform.

v) Circular Wait: If a Process is waiting for a resource which is currently held by another process. which in turn is waiting for a resource held by the first process.

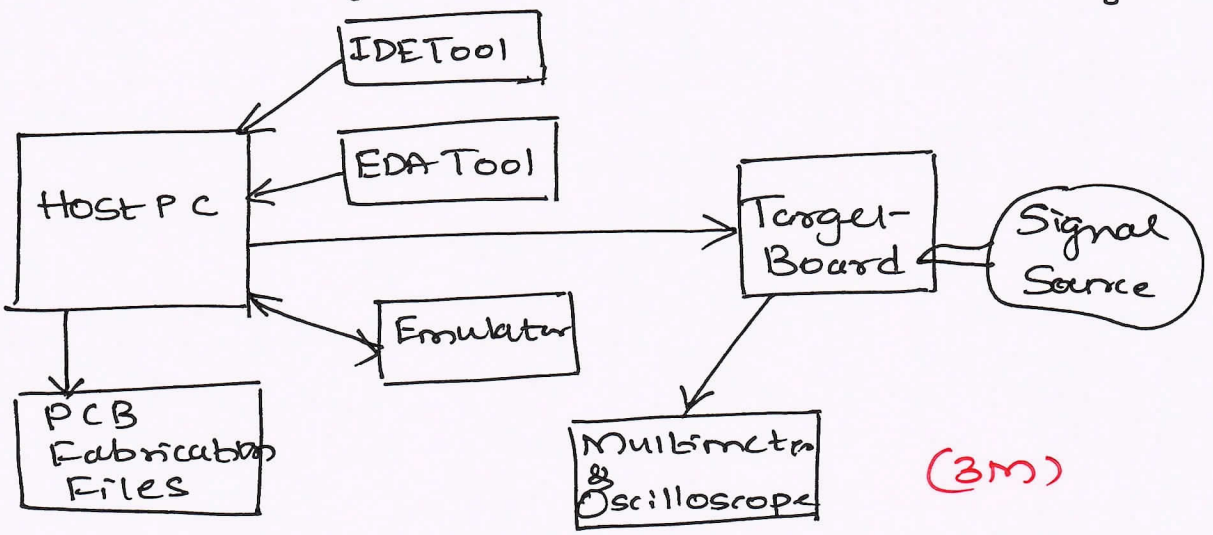
1 Mark for each

Q9b] Write a block schematic of IDE environment for embedded system design and explain their function in brief [8m] [July 2018]

P.T.O

Q9B] Contd....

Following Figure shows the schematic of IDE



(3M)

* The IDE is a software that assists programmers in developing software. IDEs normally consist of a source code editor, a compiler, a linker/locater and a debugger.
* Generally, an IDE is developed to one specific programming language or one (family of) specific processor or hardware. Some commonly used IDEs for embedded systems are the GNU compiler collections (GCC), Eclipse, Delphi.
* IDEs provides the tools for a computer programmer to develop a software. It allows programmer to write their high level code, test it, translate it. They consist of

1. Source code editor
2. Interpreter
3. Automation Tools
4. Debugger
5. Compiler
6. Auto-Documentation
7.

(5M)

Q10a] Bring out the difference between Simulator and Emulator [2M] [Sept-2020].

Simulator	Emulator
1. It is also a virtual device which replicates only the software functionality of the target device for testing purpose	1. It is a virtual device which replicates the hardware and software of the target device on computer.
2 Example: ModelSim	2. Example: Fritzing

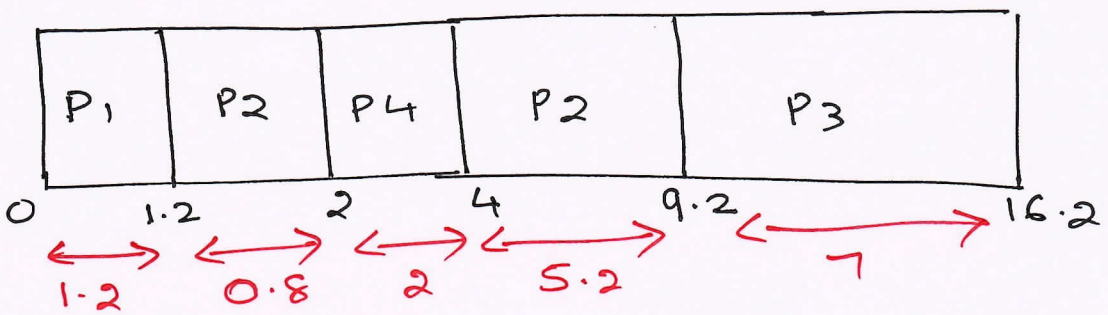
Q10b] Describe a Preemptive SJF Scheduling. Determine Avg TAT and Avg WT, If Process P1, P2 and P3 with estimated (completion time) of 1.2, 6, 7 ms. enter ready queue to gether and later P4 with a completion time of 2 ms enters the ready queue after 2 ms [7M] [Sept-2020].

Pre-emptive SJF

* In this algorithm, a task is chosen whose remaining run time is the shortest.
* If new task needs less time to complete than the current task, then the current task is blocked and the new task is run.
* Given

Process IDs	(ms) Completion Time	(ms) Arrival Time
P1	1.2	0
P2	6	0
P3	7	0
P4	2	2 ms

Q10b] Contd....



Process	Waiting Time (ms)	Turnaround Time (ms)
P_1	0	1.2
P_2	3.2	9.2
P_3	9.2	16.2
P_4	0	2

$$\text{Avg. Wait Time} = \frac{0+3.2+9.2+0}{4} = 3.10 \text{ ms}$$

$$\text{Avg TAT} = \frac{1.2+9.2+6.2+2.0}{4} = 4.65 \text{ ms}$$

Q.10c] Explain the term process, task and thread. [7M] [Sept-2020].

Consider

1) Task: These are the units of sequential code implementing the system actions and executed concurrently by an OS. [1M]

Q 10 C] Contd ...

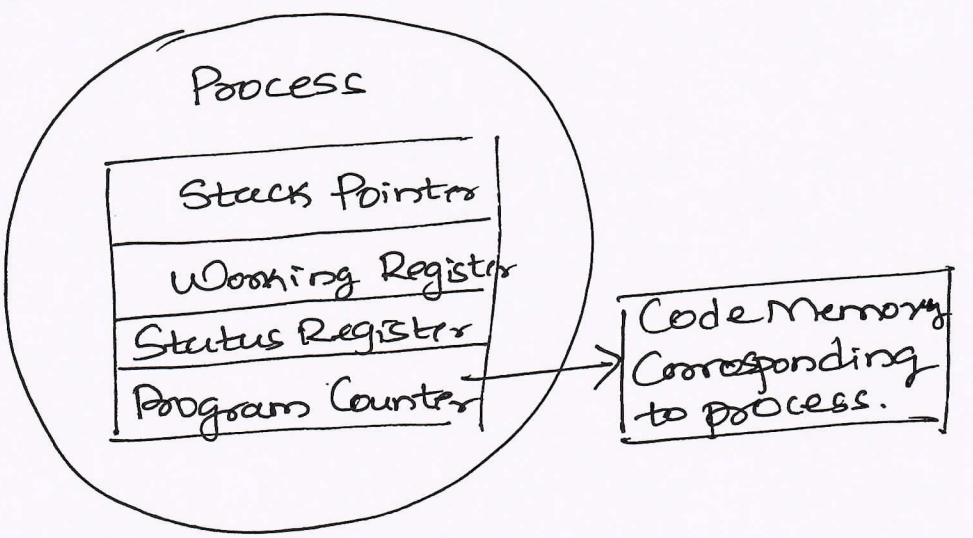
ii] Process [4M]

A process is a program, or part of it, in execution. Process is also known as an instance of a program in execution. Multiple instance of the same program can execute simultaneously.

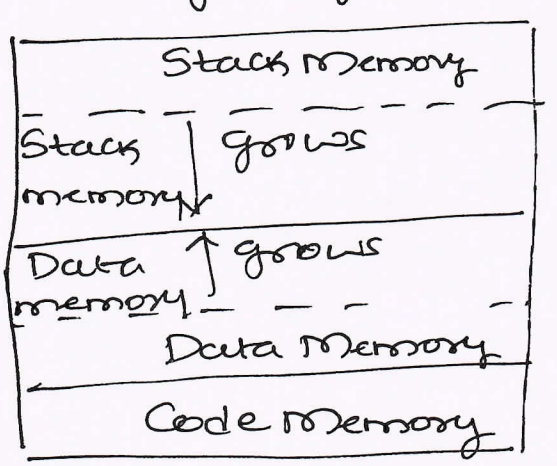
Example : Two PDF Files opened in an new tab / tabs

Following Fig shows the Structure of a Process and also the Memory Organisation.

i) Process Structure



ii) Memory Organisation



Q10] contd...

Let us take a print process of two different files,

→ here the contents / files to be printed are stored in Stack memory (file address)

→ Working registers are related to the taking the work from the related device drivers. i.e. here it checks for printer drivers)

→ Status Register: It indicates, if the given task is completed or not

→ Program Counter: It counts the no of processes involved and points to the address of the next process to be executed.

In overall process, the Stack memory increases as no of process increase and data consumption also increases by data memory

iii] Threads [2m] Threads are part of a process, A process may include no of threads.

Example of threads

i) Audio file lists in a music player.