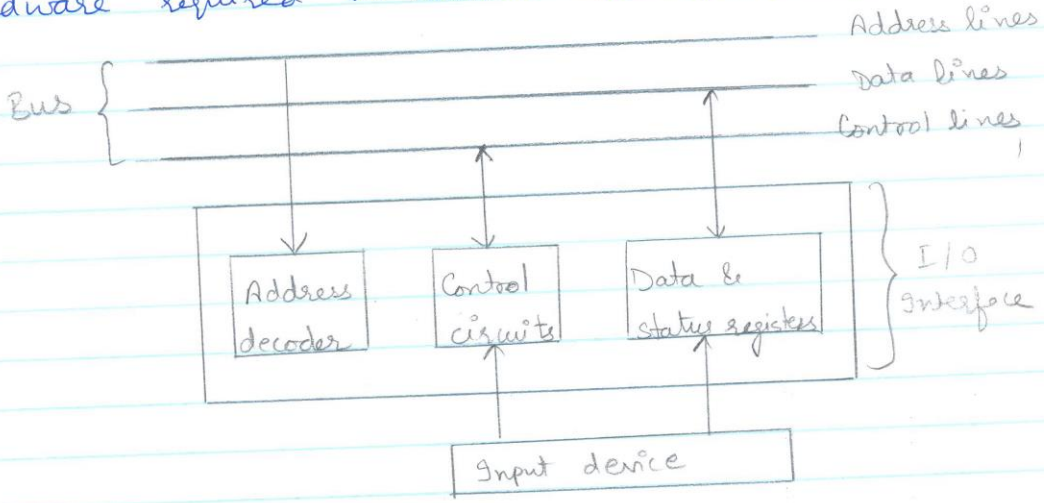
MODULE 2- IMPORTANT QUESTIONS

Q1 I/O device interface or hardware for I/O interface with registers?

Hardware required to connect an I/O device to the bus.



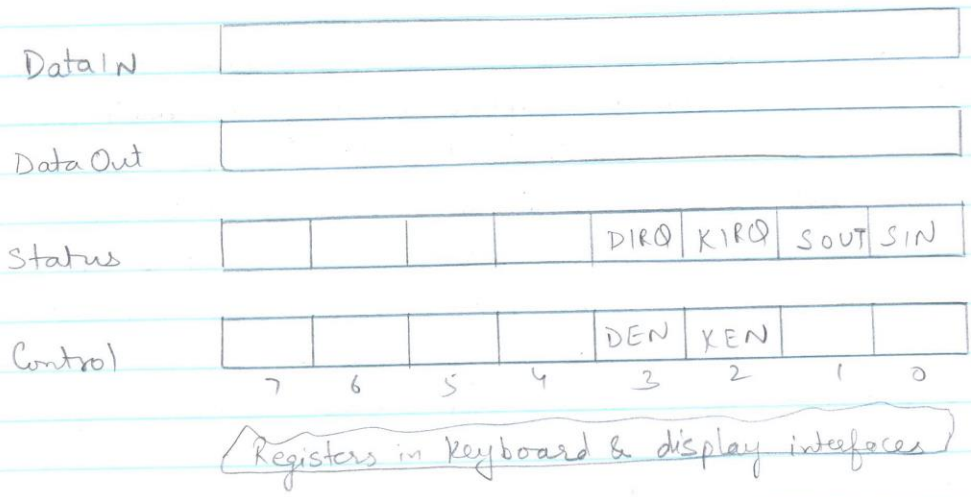
I/O interface for an input device

Address decoder - Enables the device to recognize its address when this address appears on address lines

Data register - Holds the data being transferred to or from the processor.

Status register - Contains information relevant to the operation of the I/O device.

The address decoder, the data & status registers and control circuitry required to coordinate I/O transfers constitute the device's interface circuit



Registers in keyboard & display interfaces

Registers: DATAIN, DATAOUT, STATUS, CONTROL

Flags :- SIN, SOUT — Provide status information for keyboard & display unit.

KIRQ, DIRQ — Keyboard, Display Interrupt bits

DEN, KEN — Keyboard, Display Enable bits

Eg. (You can refer to Page-42 for similar example)

```
        Move    #LINE, R0       Initialize memory pointer
WAITK   TestBit #0, STATUS      Test SIN
        Branch=0 WAITK          Wait for character to be entered
        Move    DATAIN, R1      Read character.
WAITD   TestBit #1, STATUS      Test SOUT.
        Branch=0 WAITD          Wait for display to become ready
        Move    R1, DATAOUT     Send character to display.
        Move    R1, (R0)+       Store character & advance pointer
        Compare #$0D, R1        Check if Carriage Return
        Branch≠0 WAITK          If not, get another character
        Move    #$0A, DATAOUT   otherwise send line feed.
        Call    PROCESS         Call a subroutine to process the
                                input line.
```

Q2 Importance of interrupts with example

MODULE 2- IMPORTANT QUESTIONS

INTERRUPTS

In program-controlled I/O processor repeatedly tests the device status. During this wait loop, processor is not performing any useful computation. There are many situations where other tasks can be performed while waiting for the I/O device to become ready. I/O device may alert the processor when it becomes ready. This alert may be sent using a hardware signal called an interrupt to the processor. Generally, at least one of the bus control lines, called an interrupt

Program : It consists of 2 routines.
COMPUTE – produces a set a n lines of output.
PRINT – send lines of output to printer, one line at a time.

without interrupts

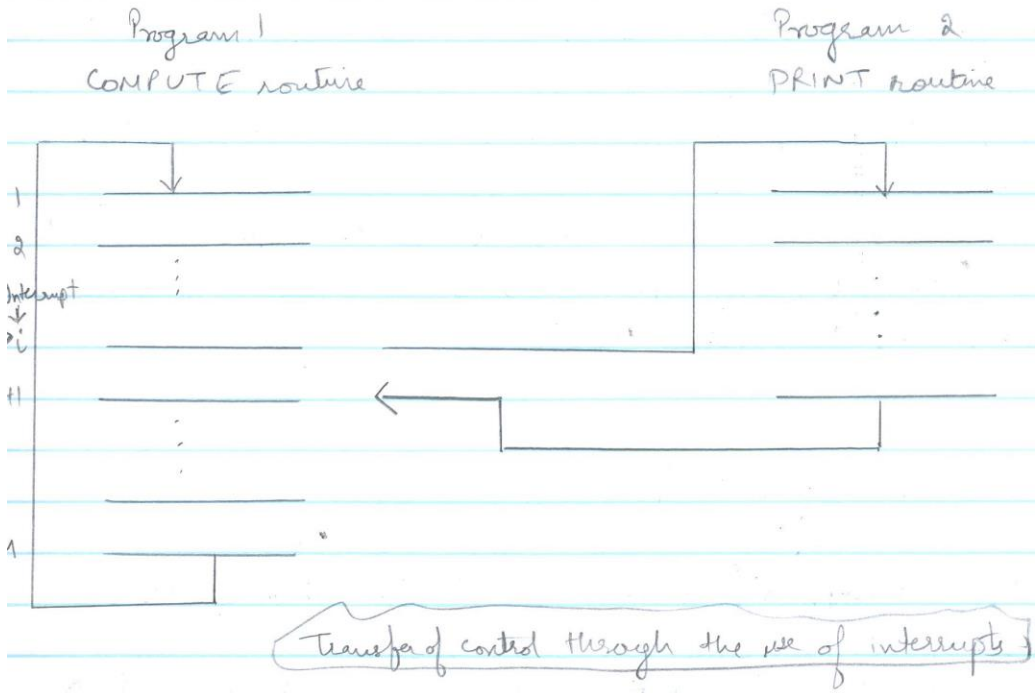
COMPUTE produces n lines.
PRINT sends 1st line
(wait for it to be printed)
send 2nd line
(wait for it to be printed)
⋮
so send n^{th} line
(wait for it to be printed)
COMPUTE produces next n lines.
PRINT sends 1st line
(wait)
send 2nd line.
(wait)
⋮
send n^{th} line
(wait)
⋮
so on.

isadvantage – Processor spends a considerable amount of time waiting for printer to become ready.

→

With interrupts Overlapping printing and computation.
i.e., execute COMPUTE routine while printing is in progress.

COMPUTE produces n lines
PRINT send 1st line
(suspend PRINT)
COMPUTE next n lines, printer printing
If printer ready, send ISR.
COMPUTE interrupted.
PRINT send 2nd line.
(suspend print)
COMPUTE next n lines, printer printing
.
.
.
So on



Q3 Handling multiple devices/interrupts (polling, vectored interrupts, interrupt nesting, daisy chaining)

MODULE 2- IMPORTANT QUESTIONS

Handling multiple devices

- Multiple devices can initiate interrupts. They use common interrupt request line.

Following techniques may be used :-

- Polling
- Vectored Interrupts
- Interrupt nesting
- Daisy chaining.

① POLLING

- The IRQ (interrupt request) bit in the status register of device is set to when a device is requesting an interrupt.
- The Interrupt service routine polls the I/O devices connected to the bus.
- The first device encountered with the IRQ bit set is serviced and ISR is invoked.
- It is easy to implement, but too much time is spent on checking the IRQ of all devices, though some devices may not be requesting service.

② VECTORED INTERRUPTS

- Device requesting an interrupt identifies itself directly to the processor.

(4 to 8 bits)

The device sends a special code to the processor over the bus.

- The code contains :
  identification of the device,
  starting address of ISR,
  address of the branch to ISR (if ISR not at that location).

The location pointed to by the interrupting device is used to store the starting address of the interrupt service routine. This address is called interrupt vector. Processor reads it and loads it into PC.

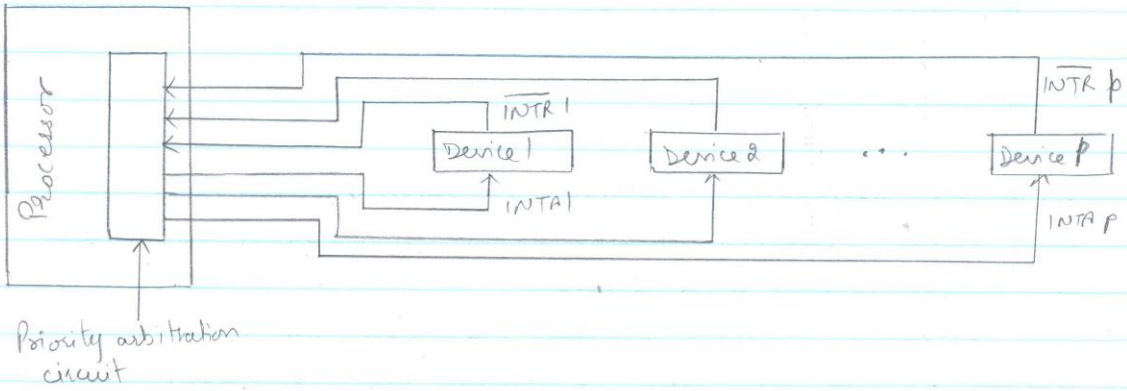
* When the processor is ready to receive interrupt-vector code, it may activate interrupt-acknowledge line, INTA. The I/O device responds by sending its interrupt-vector code and turning off the INTR signal.

② INTERRUPT NESTING

- Disabling Interrupts during execution of the ISR may not favor devices which need immediate attention. eg, keeping track of time of day.
- Pre-emption of low priority interrupt by another higher priority interrupt is known as Interrupt nesting.
- Only interrupts requests of higher priority will be accepted during execution of ISR of lower priority interrupt.
- A priority level is assigned to processor which is the priority of the program that is currently being executed. Only higher priority interrupts than this are accepted.
- Processor's priority is encoded in a few bits of processor

status word which can be changed by program instructions called privileged instructions, which can be executed only while processor is running in supervisor mode (i.e. while executing OS routines).

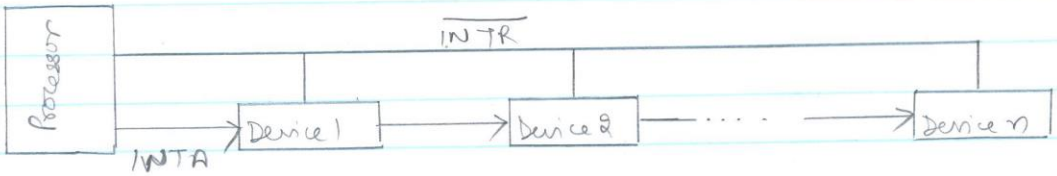
- An attempt to execute a privileged instruction while in user mode leads to a special type of interrupt called a privilege exception.
- A multiple-priority scheme can be implemented by using separate interrupt request and interrupt-acknowledge lines for each device. Each interrupt-request lines is assigned a different priority level. Interrupt requests received over these lines are sent to a priority ~~arbi~~ arbitration circuit in the processor. A request is accepted only if it has a higher priority level than that currently assigned to the processor.



Implementation of interrupt priority using individual interrupt-request and acknowledge lines

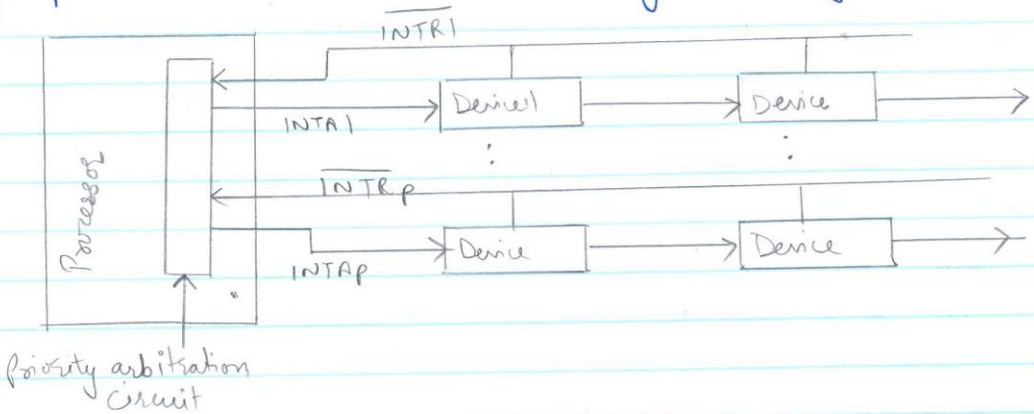
(4) DAISY CHAINING

- The interrupt request line INTR is common to all the devices.
- The interrupt acknowledgement line INTA is connected to devices in a daisy chain way.
- INTA propagates serially through the devices.
- Device that is electrically closest to the processor gets high priority.
- Low priority device may have a danger of starvation.



Daisy chaining with priority group :-

- Combining daisy chaining and interrupt nesting to form priority group.
- Each group has different priority levels and within each group devices are connected in daisy chain way.



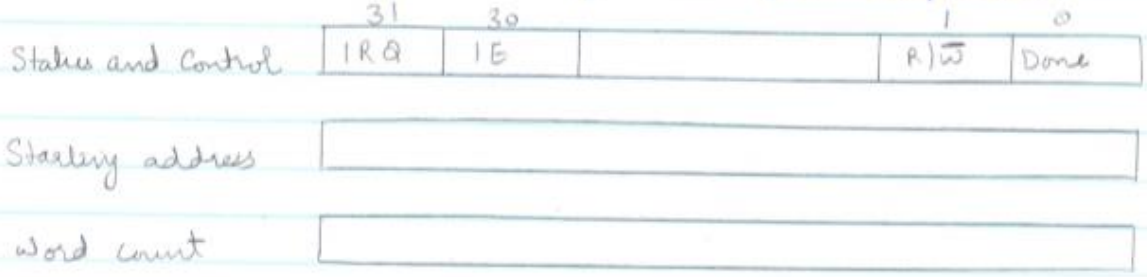
Arrangement of priority groups

Q4 DMA, DMA controller registers with bus arbitration?

DIRECT MEMORY ACCESS

- To transfer large blocks of data at high speed, between external devices and main memory, DMA (Direct Memory Access) approach is often used.
- DMA controller (control circuit in I/O device interface) allows data transfer directly between I/O device and memory, with minimal intervention of processor.
- DMA controller acts as a processor, but it is controlled by CPU.
- To initiate transfer of a block of words, the processor sends the following data to controller:
  (i) The starting address of the memory block
  (ii) The word count
  (iii) Control - to specify the mode of transfer such as read or write.
  (iv) A control to start the DMA transfer.

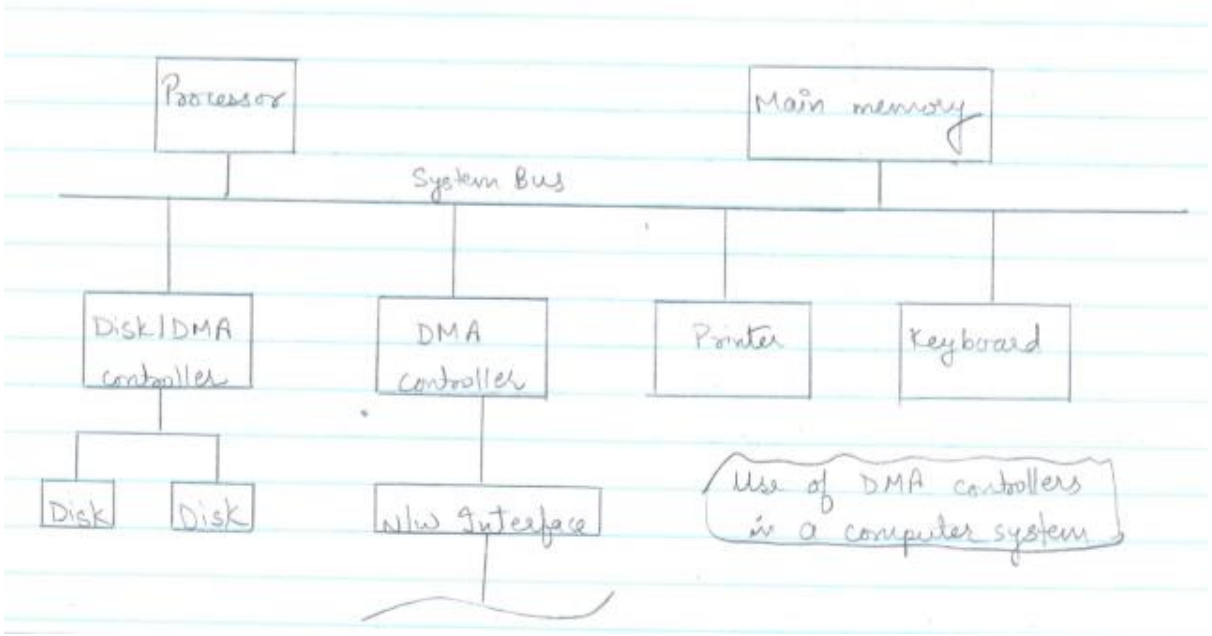
- DMA controller performs the requested I/O operation and sends an interrupt to the processor upon completion.



Registers in a DMA interface

Bits & Flags	1	0
R/W	READ	WRITE
Done	Data transfer finishes	
IRQ	Interrupt request	
IE	Raise interrupt (enable) after Data Transfer.	

MODULE 2- IMPORTANT QUESTIONS



- Memory accesses by the processor and DMA Controller are interwoven. DMA devices have <u>higher priority</u> than processor over bus control.

→ <u>Cycle Stealing</u> - DMA controller 'steals' memory cycles from processor, though processor originates most memory access. For each byte to be transferred, DMA interrupts processor.

→ <u>Block or Burst mode</u> - The DMA controller given exclusive access to the main memory to transfer a block of data without ~~its~~ interruption.

A conflict may arise if both the processor and a DMA controller or two DMA controllers try to use the bus at the same time to access the main memory. To resolve this, bus arbitration methods are required.

MODULE 2- IMPORTANT QUESTIONS

Bus Arbitration

- **Bus master**: device that initiates data transfers on the bus. The next device can take control of the bus after the current master relinquishes control.
- **Bus Arbitration**: process by which the next device to become master is selected.

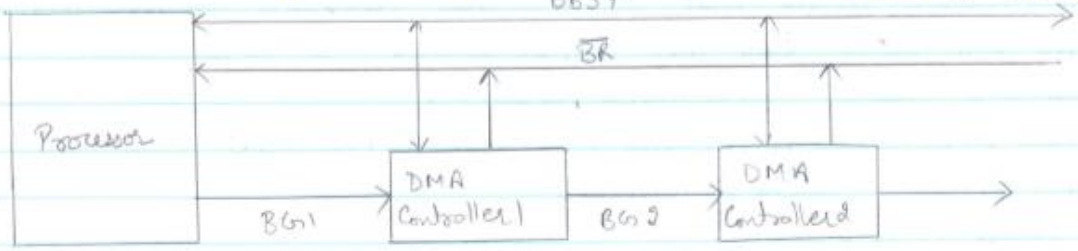
2 main types :—

1. Centralized Arbitration
2. Distributed Arbitration.

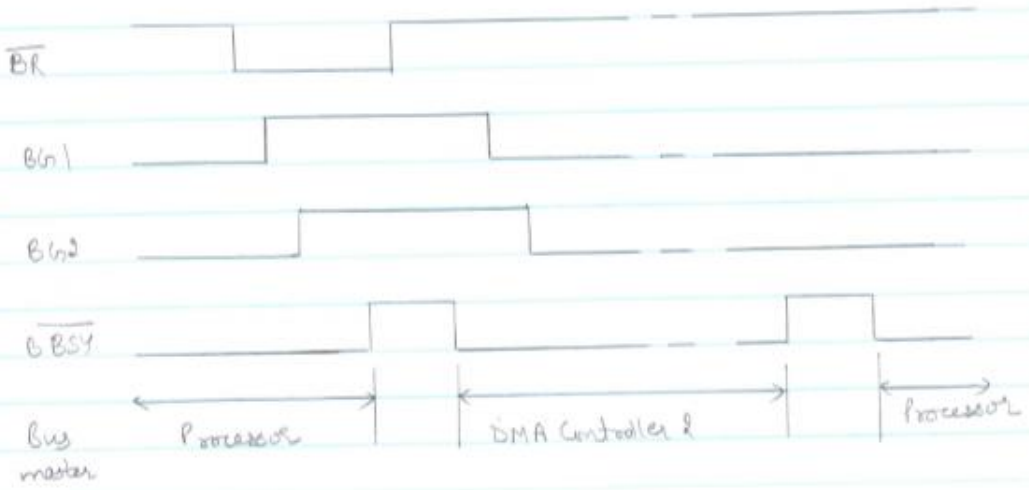
MODULE 2- IMPORTANT QUESTIONS

- Centralized Bus Arbitration Technique

→ A 'single bus arbiter' performs the required arbitration.
→ Bus arbiter may be the processor or a separate device connected to the bus (eg. DMA controller having highest priority).
→ Initially the processor will act as 'bus master'.
→ Whenever a DMA controller generates a request ($\overline{BR}$), by activating Bus Request line, processor activates Bus-Grant (BG1) signal indicating DMA controller can become a master
→ BG line is connected to all DMA devices using a daisy chain link.
→ If DMA controller 1 has enabled the request, it blocks the signal (BG1) and acts as a master for bus arbitration, thereby disabling the bus for other device use.
→ If DMA controller 1 has not enabled bus arbitration request, it simply forwards BG1 signal to its downstream neighbour DMA controllers by asserting BG2. New bus master activates $\overline{BBSY}$ (Bus Busy) line.



Bus arbitration using Daisy chain

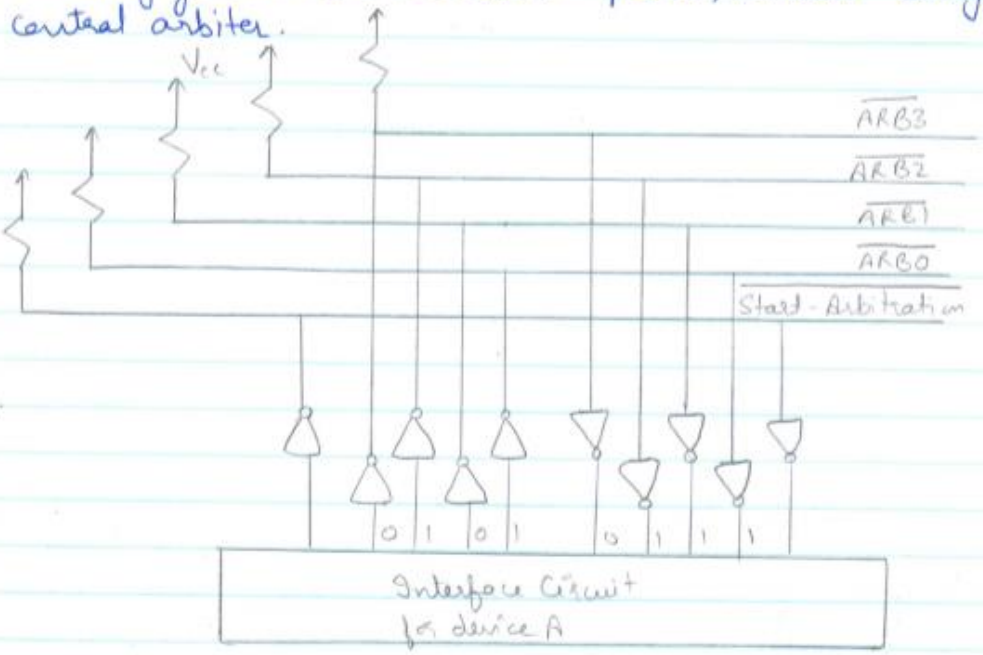


Sequence of signals for Transfer of bus mastership from processor to DMA controller 2

DMA controller 2 requests and acquires the bus mastership of the bus. At this time, it activates the bus busy line to prevent other devices from using the bus at same time. After it finishes its task, when bus is free again, processor acquires the bus if no other DMA controller request is present and activates Bus busy line.

Distributed Bus Arbitration

- All devices waiting to use the bus have equal responsibility in carrying out the arbitration process, without using a central arbiter.



Distributed arbitration scheme

→ Each device is identified by using a 4bit identification number.

→ Devices start contending for bus by enabling 'Start-arbitration' signal and place their 4-bit identification ~~number~~ on the bus (4 lines $ARB_0 - ARB_3$).

→ Device having ~~has~~ highest identification number is selected to get the granted service.

→ Selection procedure :

MODULE 2- IMPORTANT QUESTIONS

- Assume that two devices A and B have their identification numbers 5 and 6 respectively.

  i.e id of A = 0101

  id of B = 0110.

  Device A transmits the pattern 0101 on arbitration lines & device B sends the pattern 0110 on arbitration lines.

- A code value is calculated applying 'Logical OR' on identification numbers of contending devices.

  i.e. 0101

  + 0110

  0111 ← code generated.

  This code generated by 'OR' operation is sent back to all the contending devices.

- Each contending device, compares its own id on arbitration lines with code value bit by bit starting from MSB. that bit &
- When it finds a mismatch in any bit place, the remaining lower order bits of that device id are disabled to '0'.

  device A 0101 Code 0111

  mismatch

  So now device A shows 0100.

  device B 0110 code 0111

  mismatch

  · 0110 code.

  now → 'OR' for new codes 0100
  + 0110
  0110

  This means device B wins the election and is chosen as the bus master

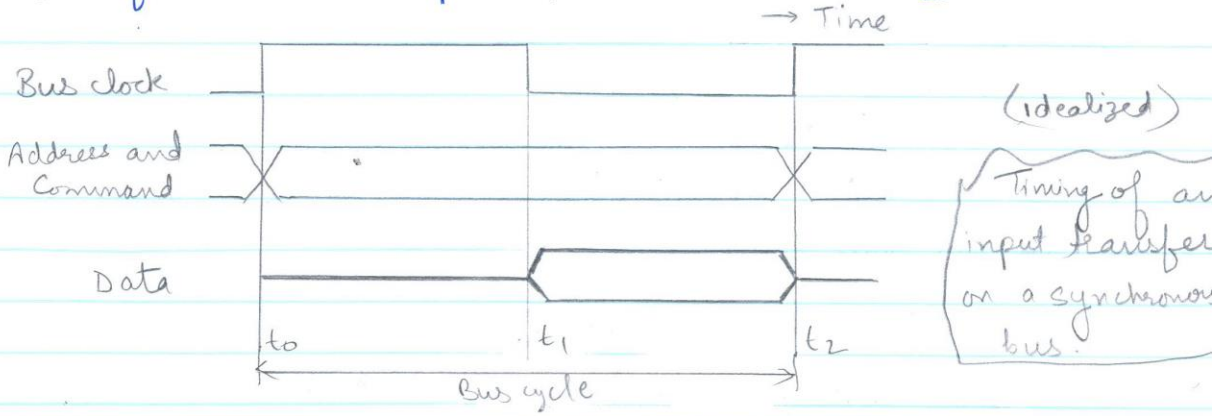
Q5 BUS –Synchronous and Asynchronous bus

BUSES

The processor, main memory and I/O devices can be inter-connected by means of a common bus. A Bus is a set of lines designed to transfer all bits of a word from source to destination. Protocol defines certain set of rules that govern the behavior of various devices connected to the bus. It defines when to place information on the bus and when to assert control signals and so The data transfer can be carried out to in two ways
→ Synchronous bus
→ Asynchronous bus.

(a) Synchronous bus.

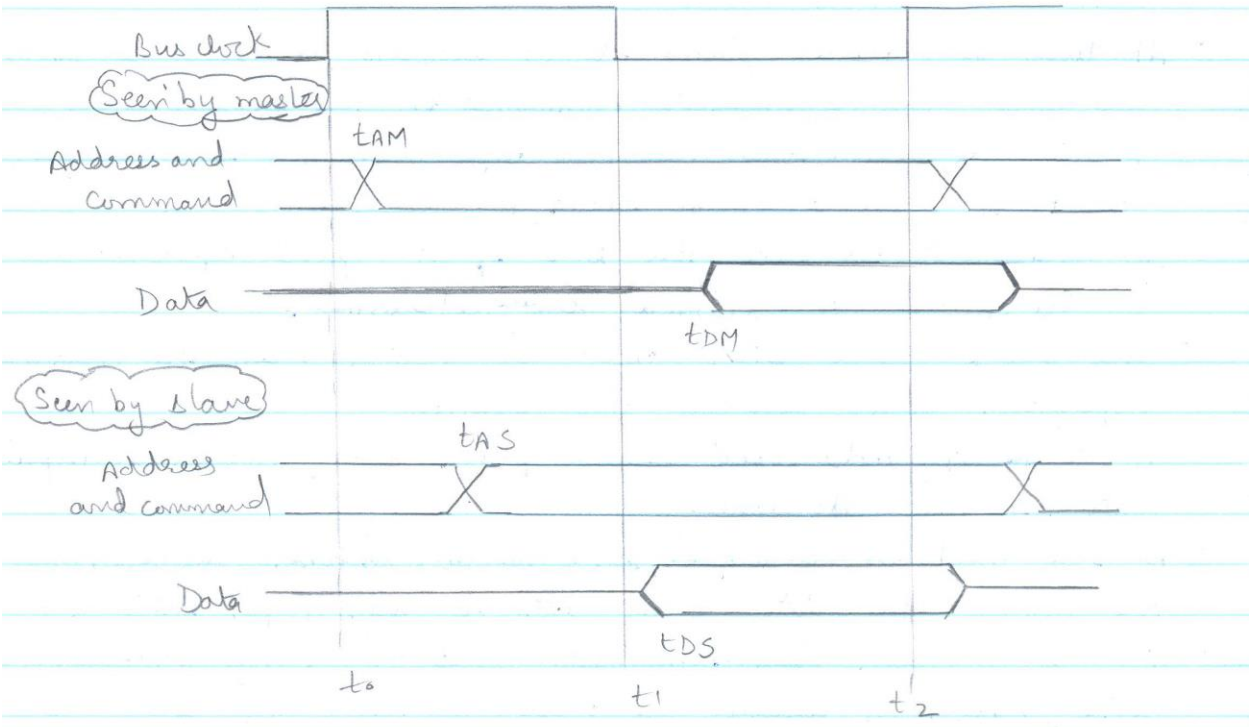
- In synchronous bus, all devices derive timing information f a common clock signal.
- Equally spaced pulses on the line define equal time int These pulses constitute a bus cycle during which one o transfer can take place. Scheme is illustrated below



Observation

- The exact time at which signals actually change state are somewhat different from the idealized signal diagram. This is because of the propagation delays on the bus wires and circuits of devices.
- The below figure shows realistic two views of the signal each of i.e, signal as seen by master and other seen by slave. Assume clock changes are seen at the same time by all the devices on the bus.

MODULE 2- IMPORTANT QUESTIONS



Detailed timing diagram for input transfer

The master sends the address and command signals on the rising edge of ~~B~~ at beginning of clock period 't_0'. But this signal actually will not appear on the bus until 't_{AM}' due to delay in the bus ~~to~~ driver circuit. A while later, at time 't_{AS}' the signal reaches the slave. The slave decodes the address at 't_1' and sends the requested data. Here also, data signals donot appear on the data lines until 't_{DS}', as data is to be taken from memory. Data travels towards master and arrives at time 't_{DM}'. At time 't_2', the master loads the data to its buffer. Here time $t_2 - t_{DM}$ is the setup time for the master's input buffer.

Multiple-cycle Transfer

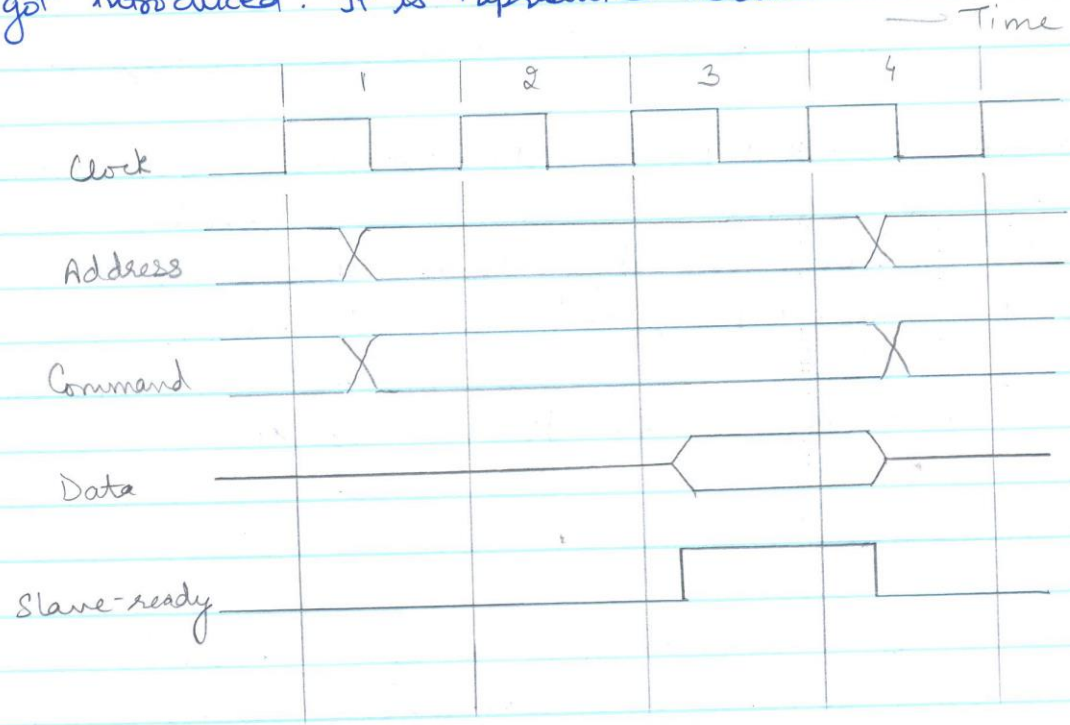
In a single clock-cycle data transfer (above mentioned scheme) there are some limitations:—

→ data transfer must take place within a single clock cycle duration

→ Because of this requirement, clock cycles are chosen to accommodate longest delays on the bus with slowest device interfaces.

→ Processor has no way to confirm whether the addressed device only has sent data.

To overcome these limitations, multiple cycle transfer got introduced. It is represented below:—



An input transfer using multiple clock cycles

Asynchronous bus

(b) Asynchronous Bus

- Data transfer is controlled by a handshake protocol.
- Instead of using common clock, it uses 2 timing control lines named master-ready and slave-ready.
- The master places the address and command information on the bus, and activates Master-ready line at t_1.
- All devices on the bus decode the address and the selected

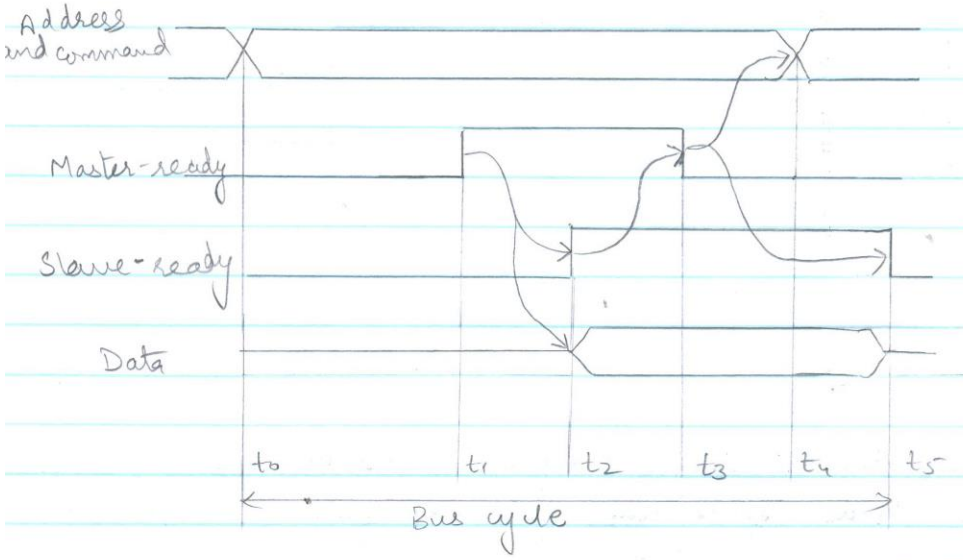
slave informs the master by setting slave - ready ~~signal~~ line after doing required operation. At t_2, selected slave does this. for input operation, it places data from its register on the data lines. It sets slave -ready signal to 1.

At t_3, slave -ready signal arrives at master indicating the availability of data on the bus. After doing some settings, master strobes the data into its input buffer and drops master - ready signal indicating it has received the data.

At t_4, master removes the address and command information from the bus.

At t_5, the device interface observes 1 to 0 transition of master ready signal and removes data and slave ready signals from the bus.

This completes the input transfer. * Delays are caused due to bus skew.

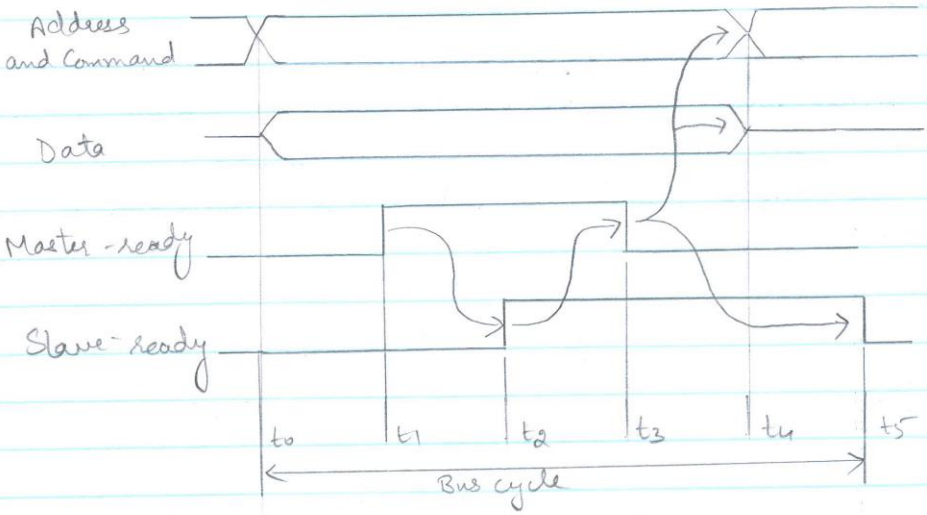


Handshake control of data transfer during an input operation

Asynchronous Bus - Input

- for output operation, master places the output data on the data lines at the same time as it puts address and command information.
- The selected slave sets slave-ready signal and strobes data into its output buffer.

The handshake signals are fully interlocked. A change of state in one signal is followed by a change in the other signal. Hence, this scheme is known as a full handshake



Handshake control of data transfer during an output operation
Asynchronous Bus- output

Choice of a particular design (synchronous or asynchronous) depends on trade-offs among following factors such as:

→ simplicity of the device interface
→ Ability to accommodate device interfaces that introduce different amounts of delay

Q6 Parallel port/interface and serial port/interface

MODULE 2- IMPORTANT QUESTIONS

Parallel Port

8-bit input port
8-bit output port

8-bit parallel port.

Assume that interface circuit is connected to a 32-bit processor that uses memory-mapped I/O and asynchronous bus protocol.

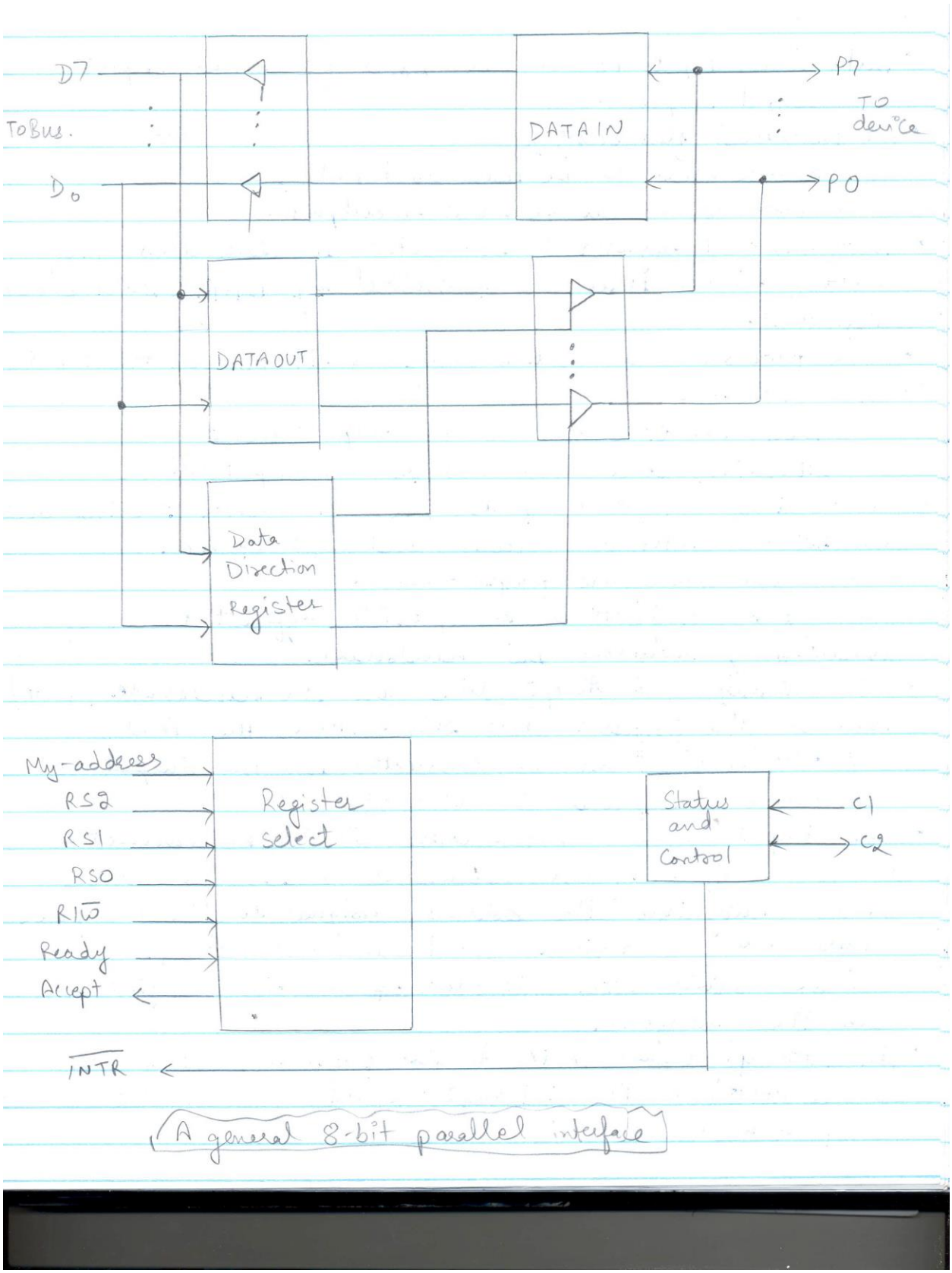
Keyboard to processor connection (Input port)

- A keyboard consists of mechanical switches that are normally open.
- When a key is pressed, its switch closes & creates path for

General 8-bit parallel interface

The input and output interfaces described above can be combined into a single interface. A more flexible parallel port is created if the data lines to I/O devices are bidirectional.

MODULE 2- IMPORTANT QUESTIONS



A general 8-bit parallel interface

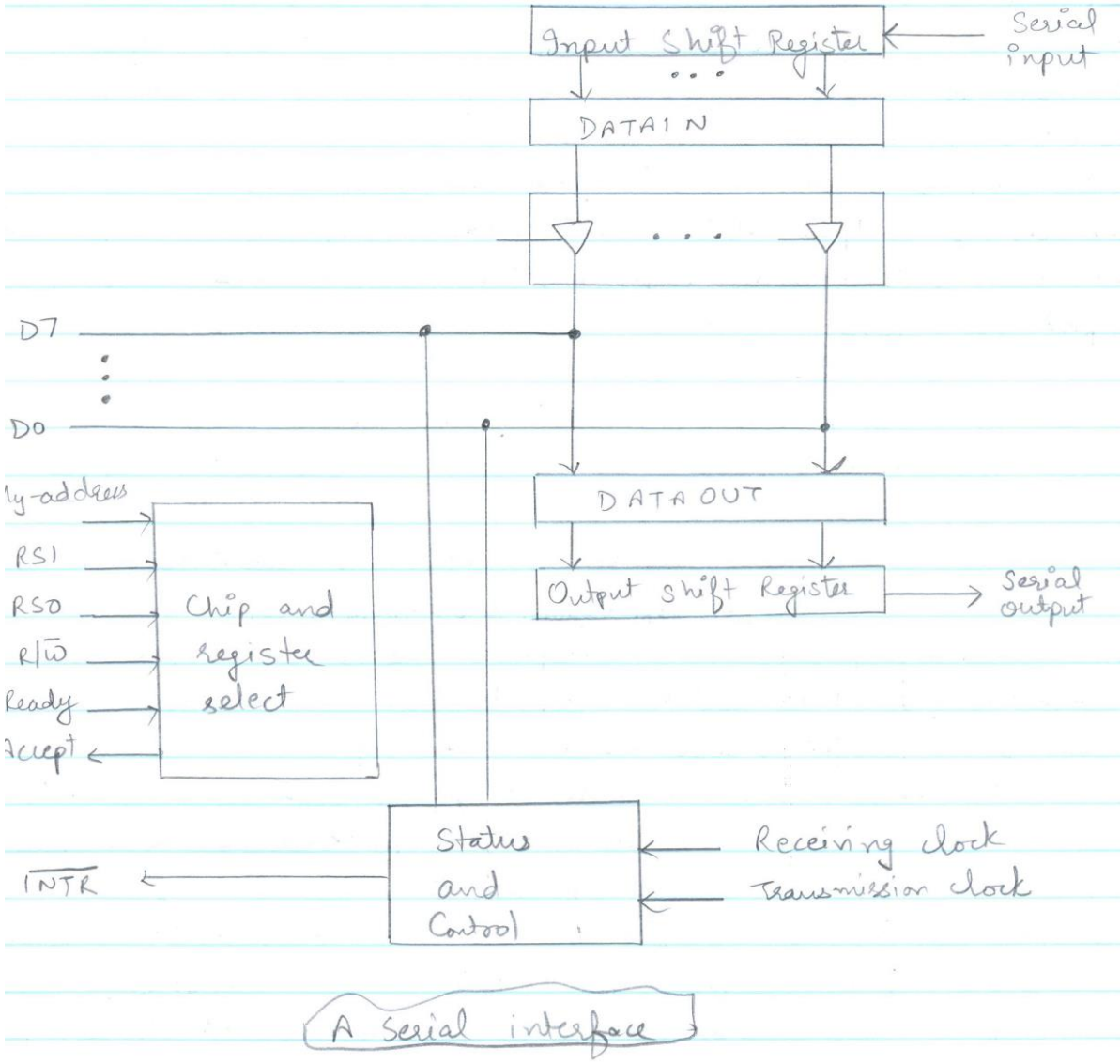
MODULE 2- IMPORTANT QUESTIONS

- Data lines P_7 to P_0 can be used for either input or output purposes.
- For increased flexibility,
  - → some lines can be used as inputs, and
  - → some lines can be used as outputs.
- The DATAOUT register is connected to data lines via 3-state drivers that are controlled by DDR (Data Direction Register).
- The processor can write any 8-bit pattern into DDR.
- If DDR = 1,
  Then, dat line acts as an output-line;
  otherwise, data-line acts as an input-line.
- Two lines, C_1 and C_2 are used to control the interaction between interface-circuit and I/O device.
  These two lines are programmable.
- Line C_2 is bidirectional to provide different modes of signaling, including the handshake.
- The Ready and Accept lines are the handshake control lines on the processor-bus side. Hence, the Ready and Accept lines can be connected to the ~~output of~~ Master-ready and Slave-ready.
- The input signal My-address should be connected to the output of an address decoder. The address decoder recognizes the address assigned to the interface.
- There are 3 register select lines: $RS_0 - RS_2$.
  Three register select lines allow up to eight registers in the interface.
- An interrupt-request INTR is also provided. It should be connected to the interrupt-request line on the computer bus.

SERIAL PORT

MODULE 2- IMPORTANT QUESTIONS

SERIAL PORT



A serial interface

- A serial port is used to connect the processor to I/O devices that require transmission of data one bit at a time.
- Serial port is capable of communicating in a bit-serial fashion on the device side and in a bit-parallel fashion on the bus side. This transformation between

parallel and serial formats is achieved with shift registers.

- The input shift register accepts bit-serial input from the I/O device. When all 8 bits of data have been received, the contents of this shift register are loaded in parallel into DATAIN register.

Similarly, output data in DATAOUT register are loaded into the output shift register, from which bits are shifted out and sent to I/O device serially.

SIN=1 when data is present in DATAIN,
as soon as processor reads it, SIN=0.

SOUT=1 when DATAOUT is available (empty) and ready to load.
as soon as processor writes in DATAOUT, SOUT becomes 0.
(i.e., shift register & DATAIN/DATAOUT)

If double buffering is not imposed, after receiving one character from the serial line, the device cannot start receiving the next character until the processor reads the contents of DATAIN. Therefore, a pause would be needed between two characters to allow the processor to read the input data. With double buffer, the transfer of the second character can begin as soon as the first character is loaded from shift register into DATAIN register.

It requires fewer wires, and thus is convenient for connecting devices that are physically far away from the computer.

Thus speed of transmission (bit rate) depends on the nature of devices connected. To accommodate a range of devices, a serial interface must be able to use a range of clock speeds. Therfore, given circuit in diagram allows

Q7 Standard I/O Interfaces- PCI, SCSI, USB

MODULE 2- IMPORTANT QUESTIONS

Refer notes(its a 10 marks detail question) anyone will be asked . USB or SCSI or PCI