Suruchi
1219316

Ques: 1

If the registers in the I/O interface share a common clock with CPU registers, then transfer between the two units is said to be synchronous. But in most cases, the internal timing in each unit is independent of each other, so each uses its private clock for its internal registers. In this case the two units are said to be asynchronous to each other, and if data transfer occurs between them, this data transfer is called Asynchronous Data Transfer.

The two methods can achieve this asynchronous way of data transfer.

1. Strobe control Method :-
   - → Source initiated strobe.
   - → Destination initiated strobe.

2. Handshaking method :-
   - → Source initiated handshaking
   - → Destination initiated handshaking.

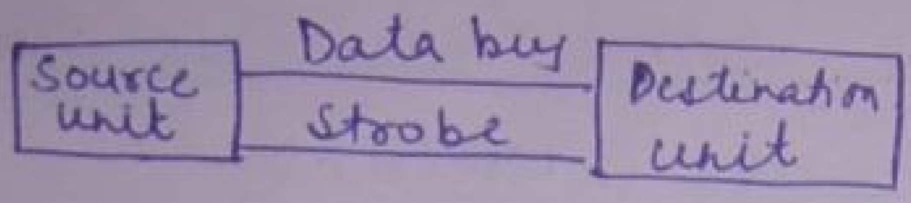
Strobe Method.

(a) Source initiated :- The source unit that initiates the transfer has no way of knowing whether the destination unit has actually received data.

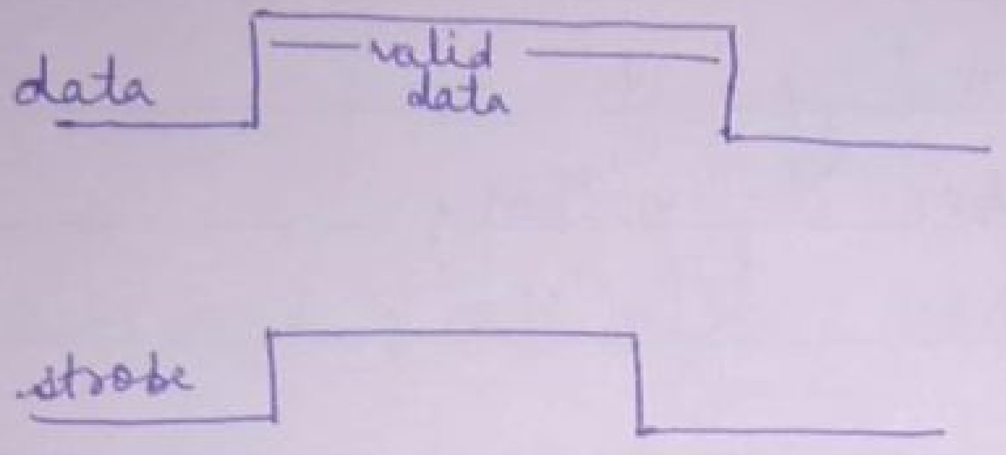
Suruchi
12193l6

Source initiated strobe

Block diagram :-

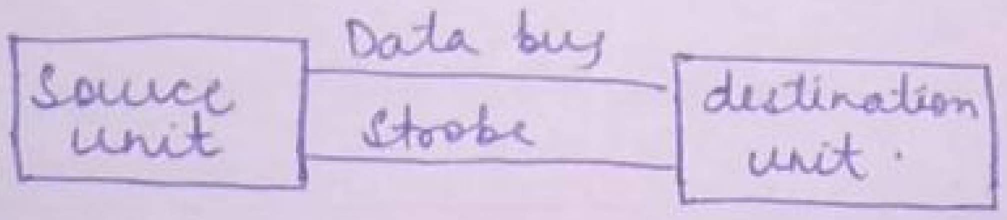


timing diagram :-

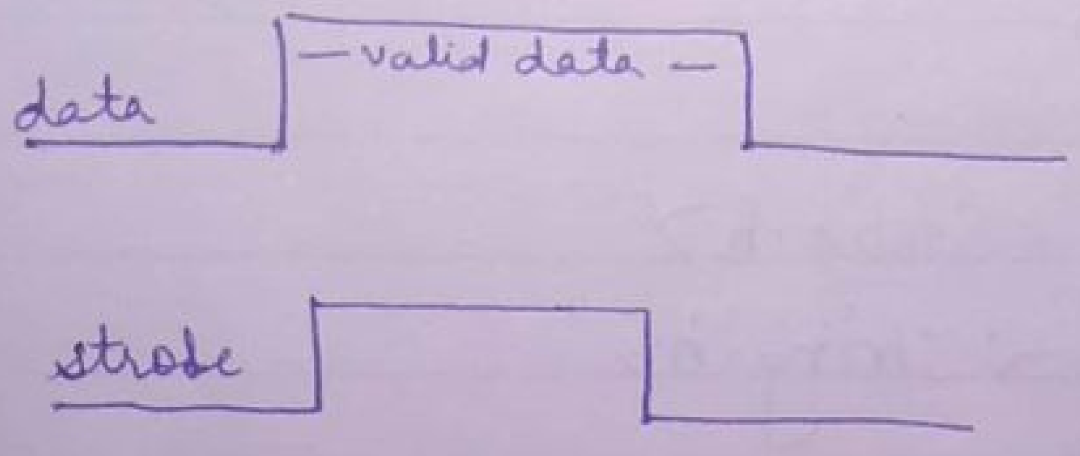


Destination initiated strobe

Block diagram :



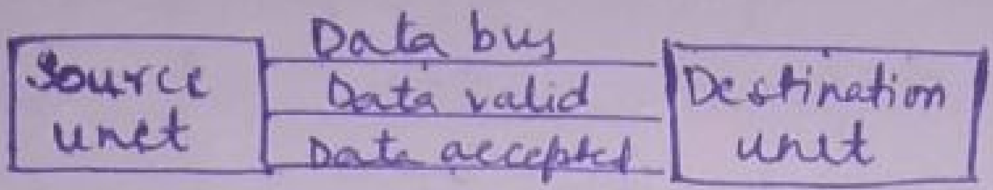
timing diagram :



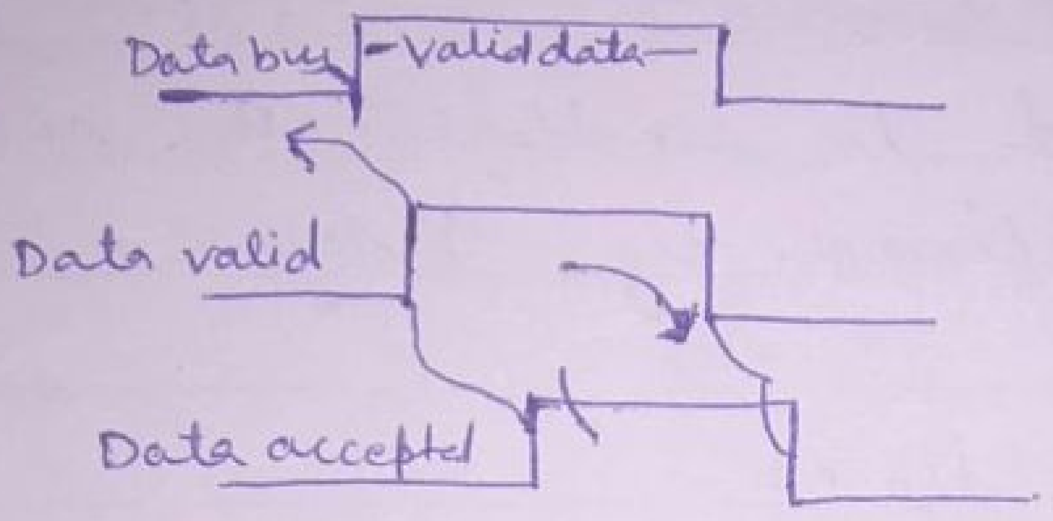
Handshaking method.

(a) Source - initiated transfer :-

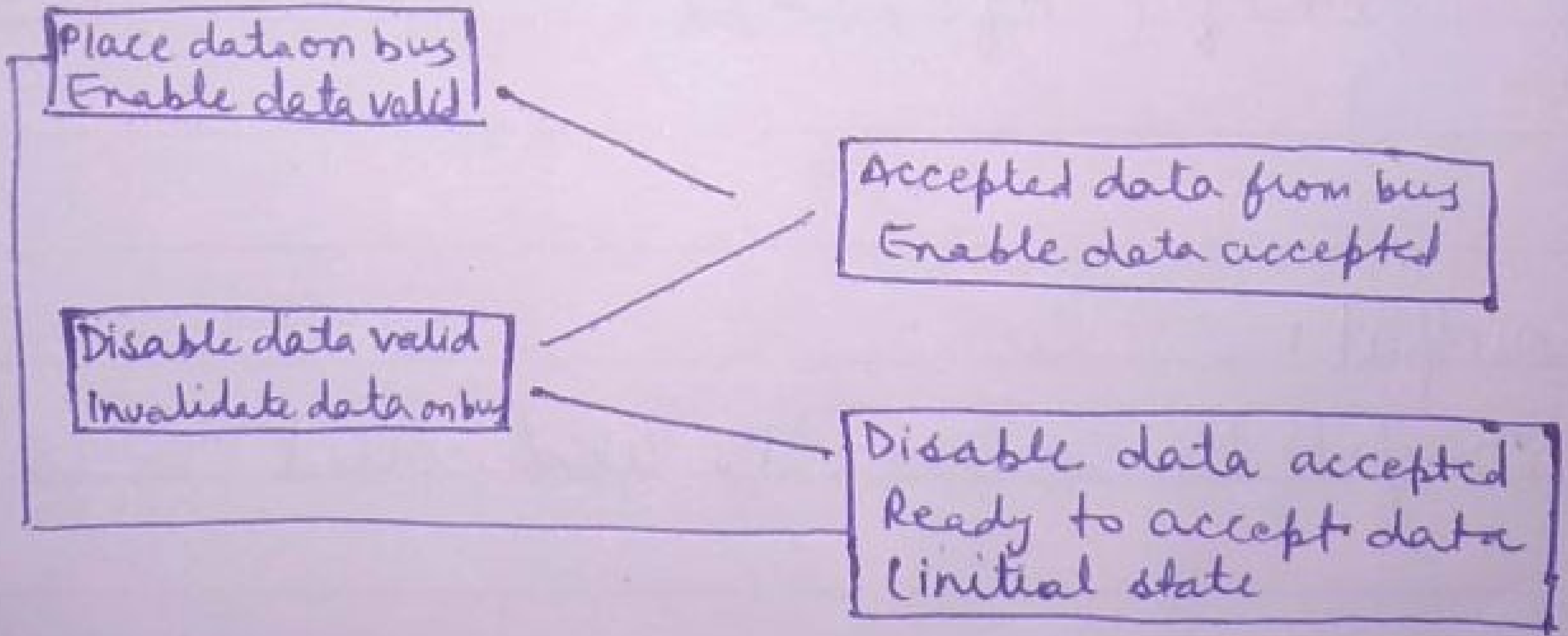
Block diagram:



timing diagram:



sequence of events:

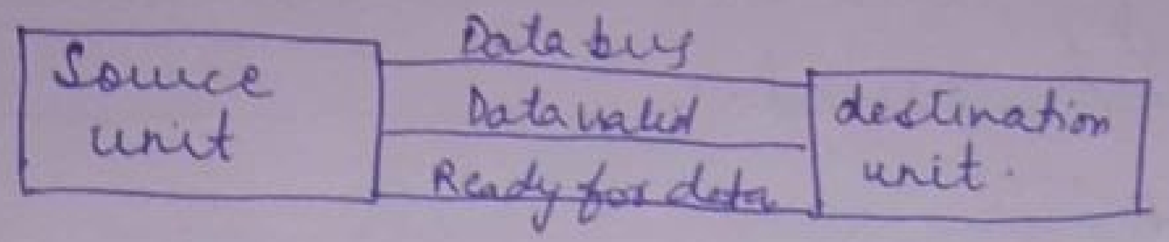


- Allows arbitary delays from one state to the next.
- Permits each unit to respond at its own data transfer rate.
- The rate of transfer is determined by the slower unit.

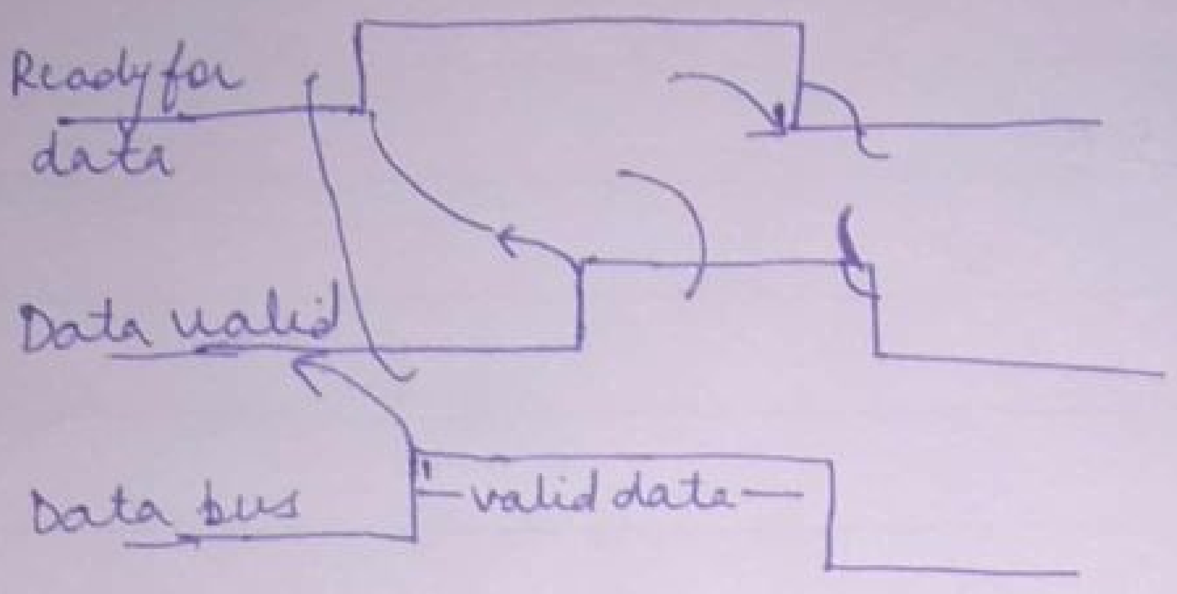
Suruchi
1219316

(b) destination initiated:

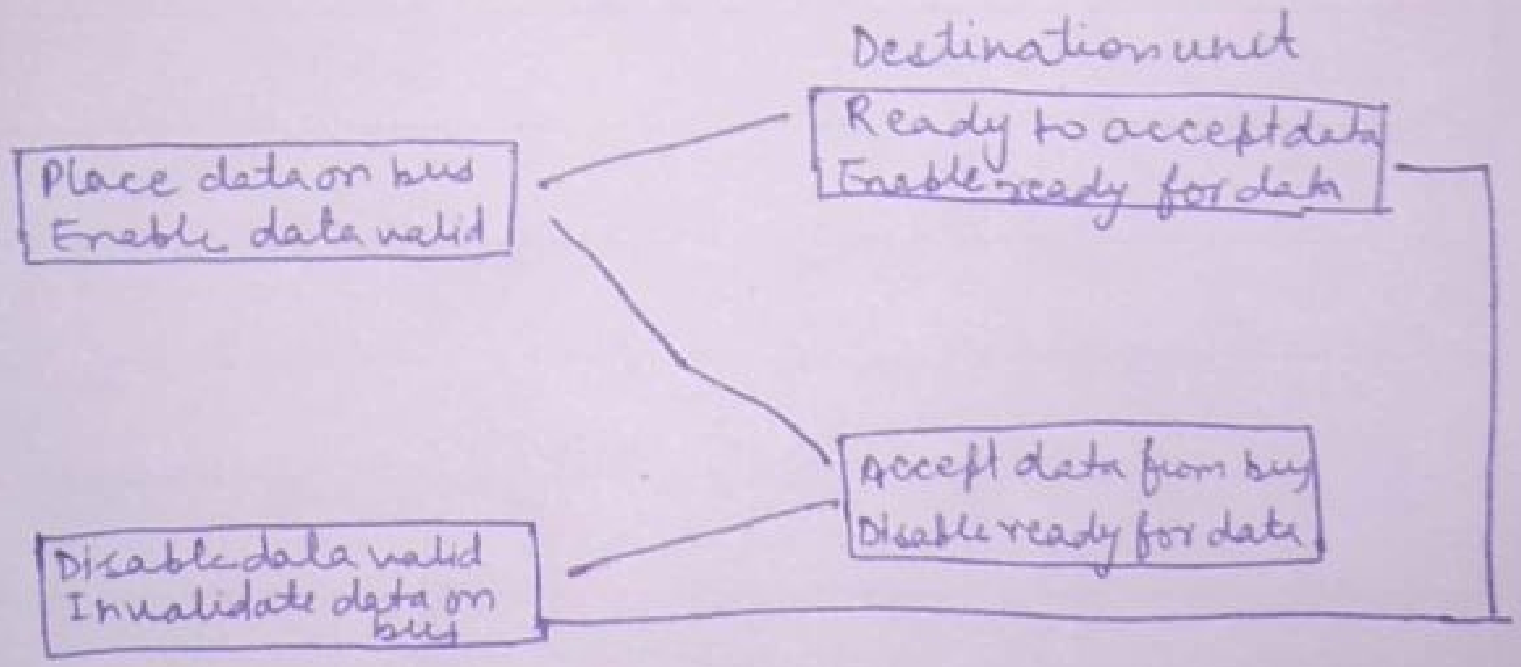
Block diagram.



Timing Diagram



sequence of events



- Handshaking provides a high degree of flexibility and reliability because the successful completion of a data transfer relies on active participation by both units.
- if one unit is faulty, data transfer will not be completed.

→ can be detected by means of a timeout mechanism.

Suruchi

1219316

Ques-2

Daisy-chaining Priority :-

The Daisy-chaining method of establishing priority consists of a serial connection of all devices that request an interrupt. The device with the highest priority is placed in the first position, followed by lower-priority devices up to the device with lowest priority, which is placed last in the chain.

→ The interrupt request line is common to all devices & forms a wired logic connection.

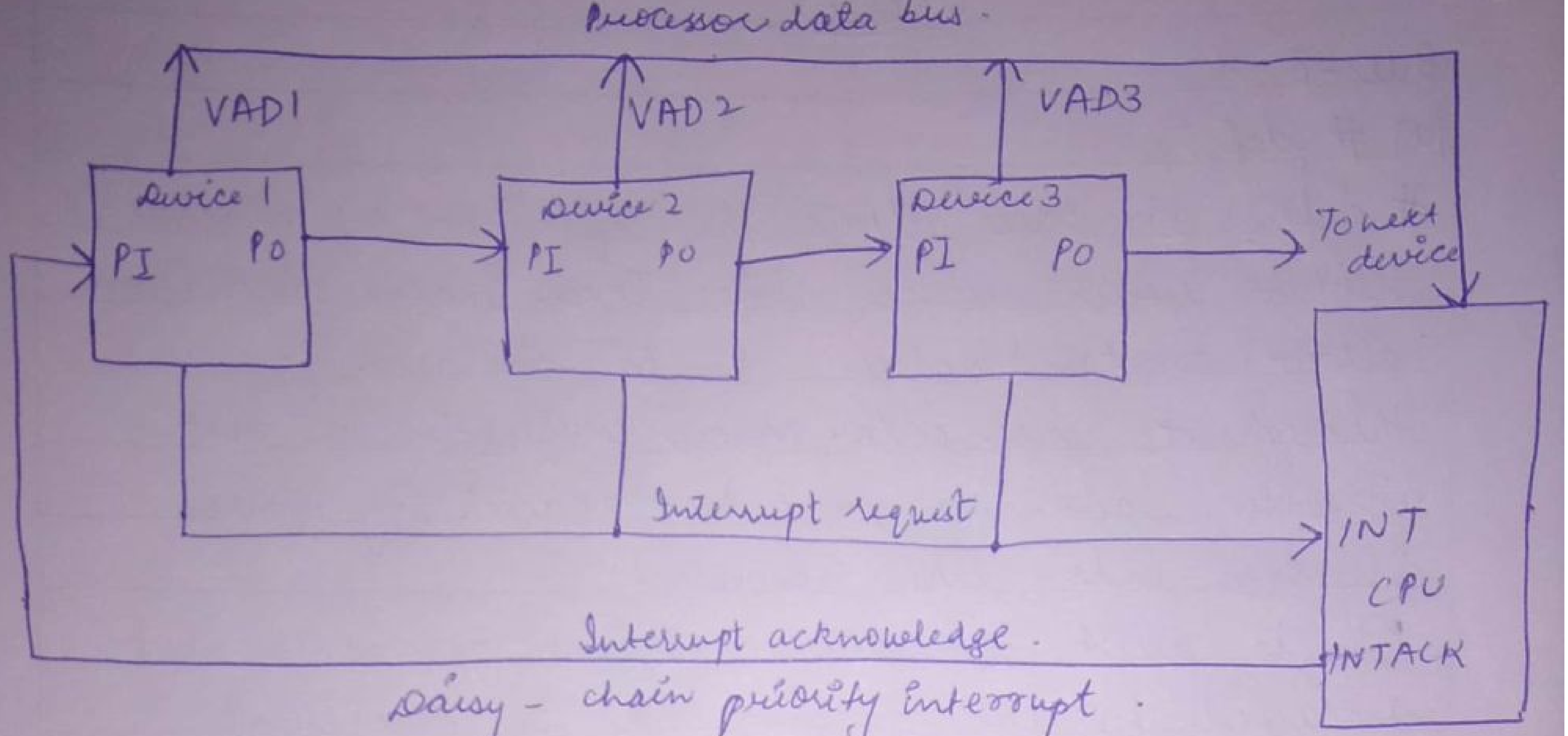
→ If any device has its interrupt signal in the low level state, the interrupt ~~Input~~ line goes to the low-level state & enables the interrupt input in the CPU.

→ When no interrupts are pending, the interrupt line stays in the high-level state and no interrupts are recognized by the CPU. This equivalent to a negative-logic OR operation.

→ The signal is received by the device 1 at the PI input. The acknowledge signal passes on to the next device through the PO output only if device 1 is not requesting an interrupt.

→ If device 1 has a pending interrupt, it blocks the acknowledge signal from the next device by placing a 0 in the PO output. It then proceeds to insert its own interrupt vector address (VAD) into the data bus for the CPU to use during the interrupt cycle.

Suruchi
1219316



Daisy - chain priority interrupt.

→ A device with a 0 in its PI input generates a 0 in its PO output to inform the next-lower-priority device that the acknowledge signal has been blocked.

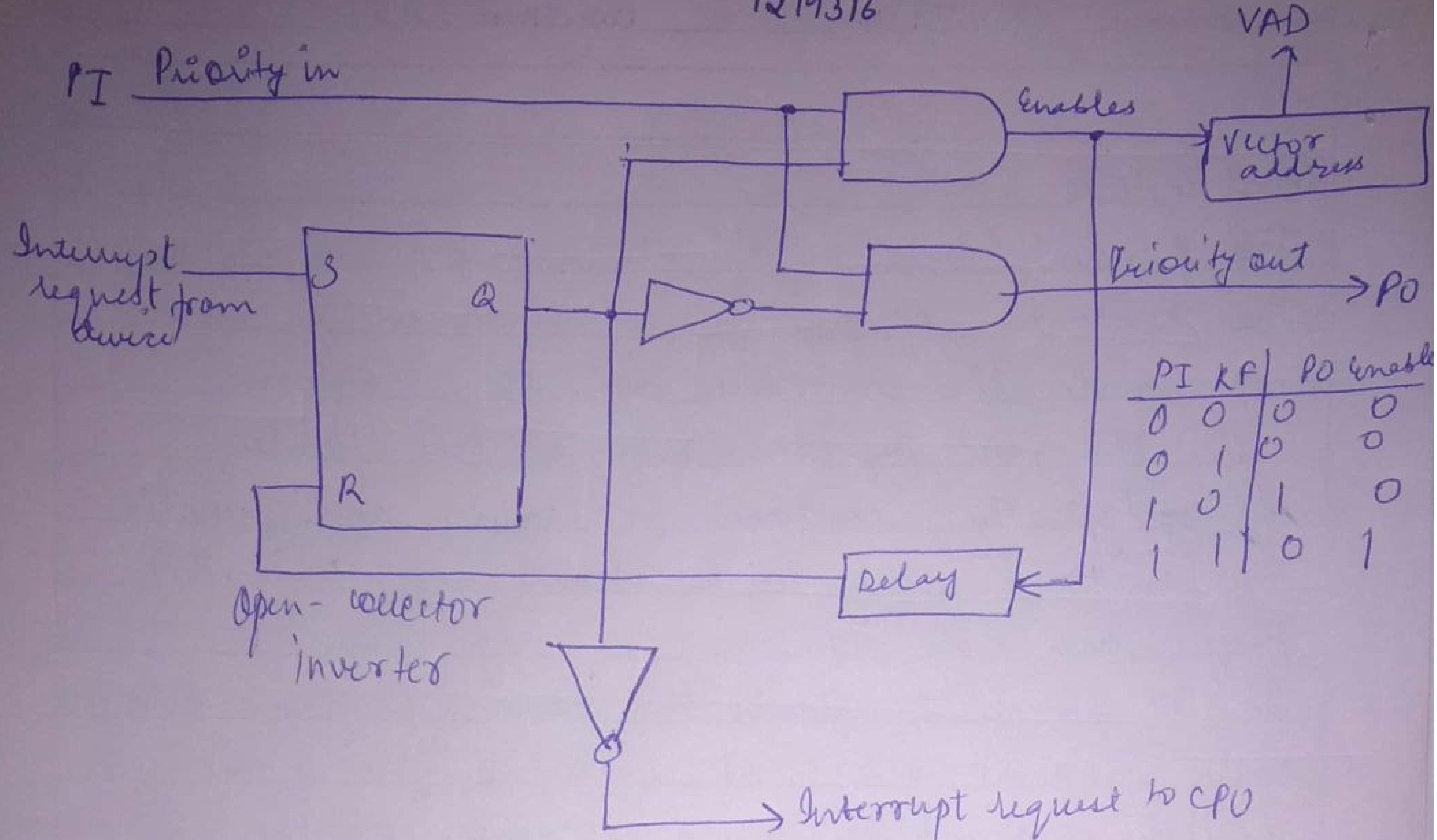
→ A device that is requesting an interrupt & has a 1 in its PI input will intercept the ~~acknowledge signal~~ and has a 1.

→ Thus the device with PI = 1 and PO = 0 is the one with the highest priority that is requesting an interrupt & this device places its VAD on the data bus.

→ The daisy chain arrangement gives the highest priority to the device that receives the Interrupt acknowledge signal from the CPU.

→ The father the device is from the 1st position, the lower is its priority.

Suruchi
1219316



PI	RF	PO	Enable
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

one stage of daisy chain priority arrangement

Parallel Priority Interrupt

The parallel priority interrupt method uses a register whose bits are set separately by interrupt signal from each device. Priority is established according to the position of the bits in the register.
In addition to the interrupt register, the circuit may include a mask register whose purpose is to control the status of each interrupt request. The mask register can be programmed to disable the lower-priority interrupt while a higher-priority device is being serviced.

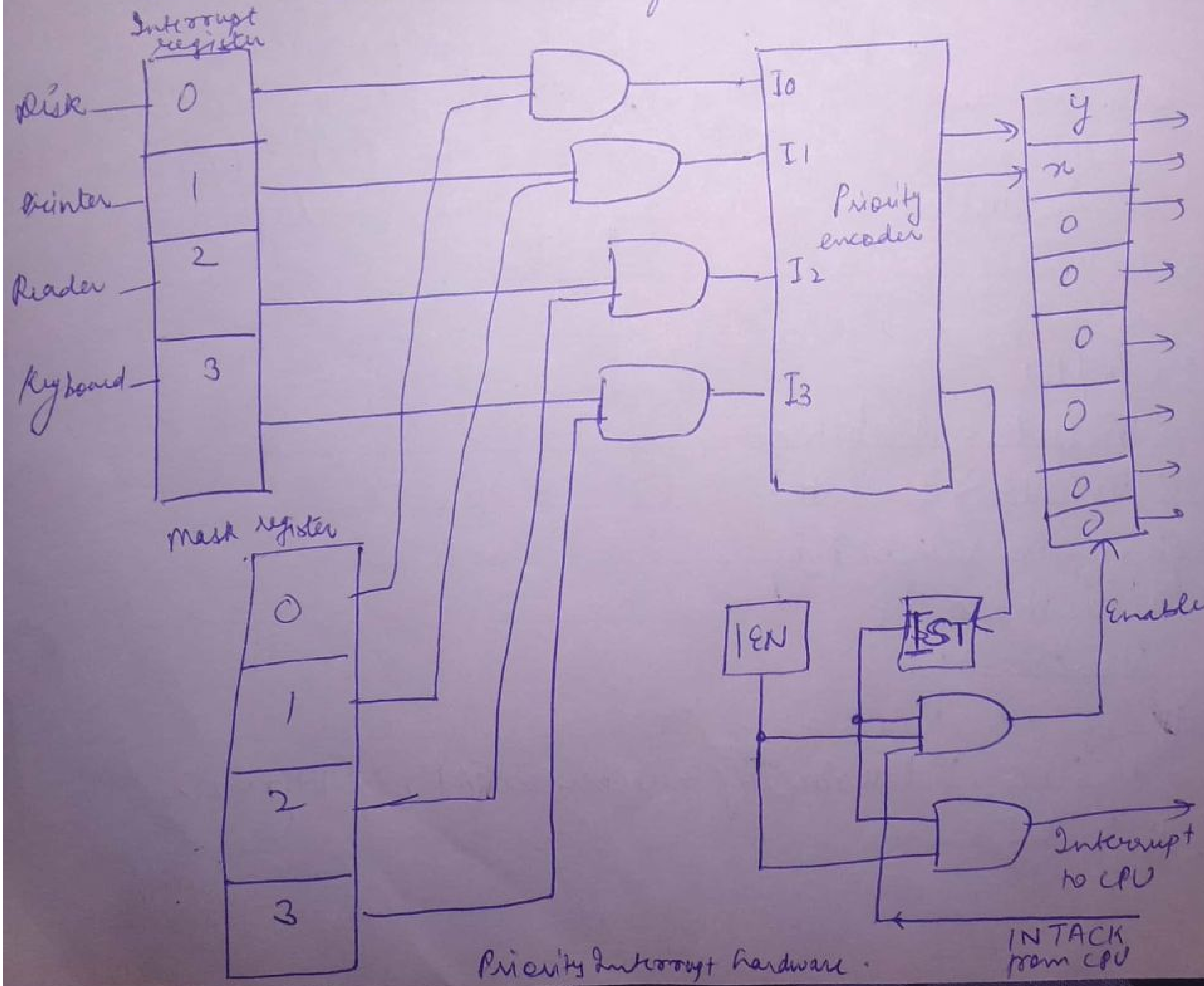
Suruchi
1219316

→ The priority logic for a system of four interrupts source ~~[illegible]~~. It consist of an interrupts register whose individual bits are set by external conditions and cleared by program instructions.

→ The magnetic disk, being a high-speed device, is given the highest priority.

→ In this way an interrupt is recognized only if its corresponding mask bit is set to 1 by the program.

→ The priority encoder generates two bits of vector address which is transferred to the CPU.



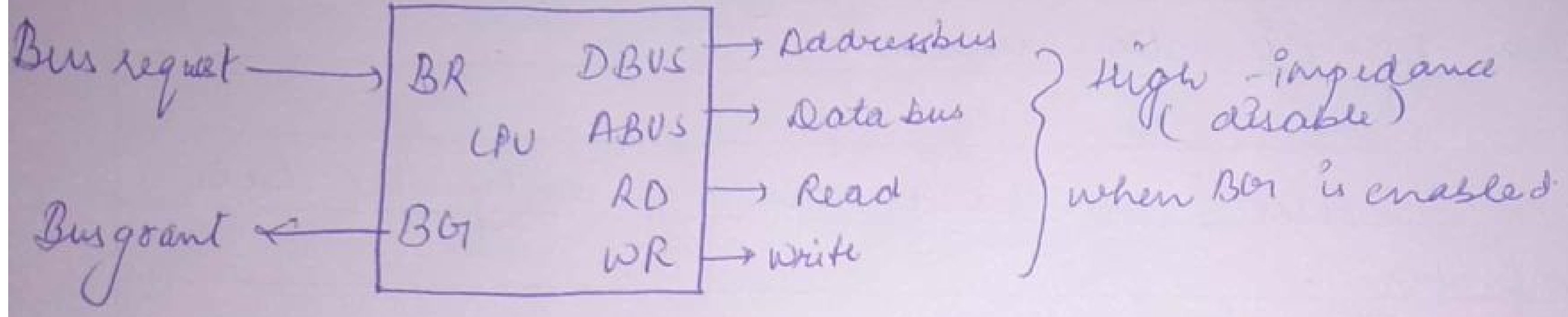
Priority Interrupt hardware.

Suruchi
1219316

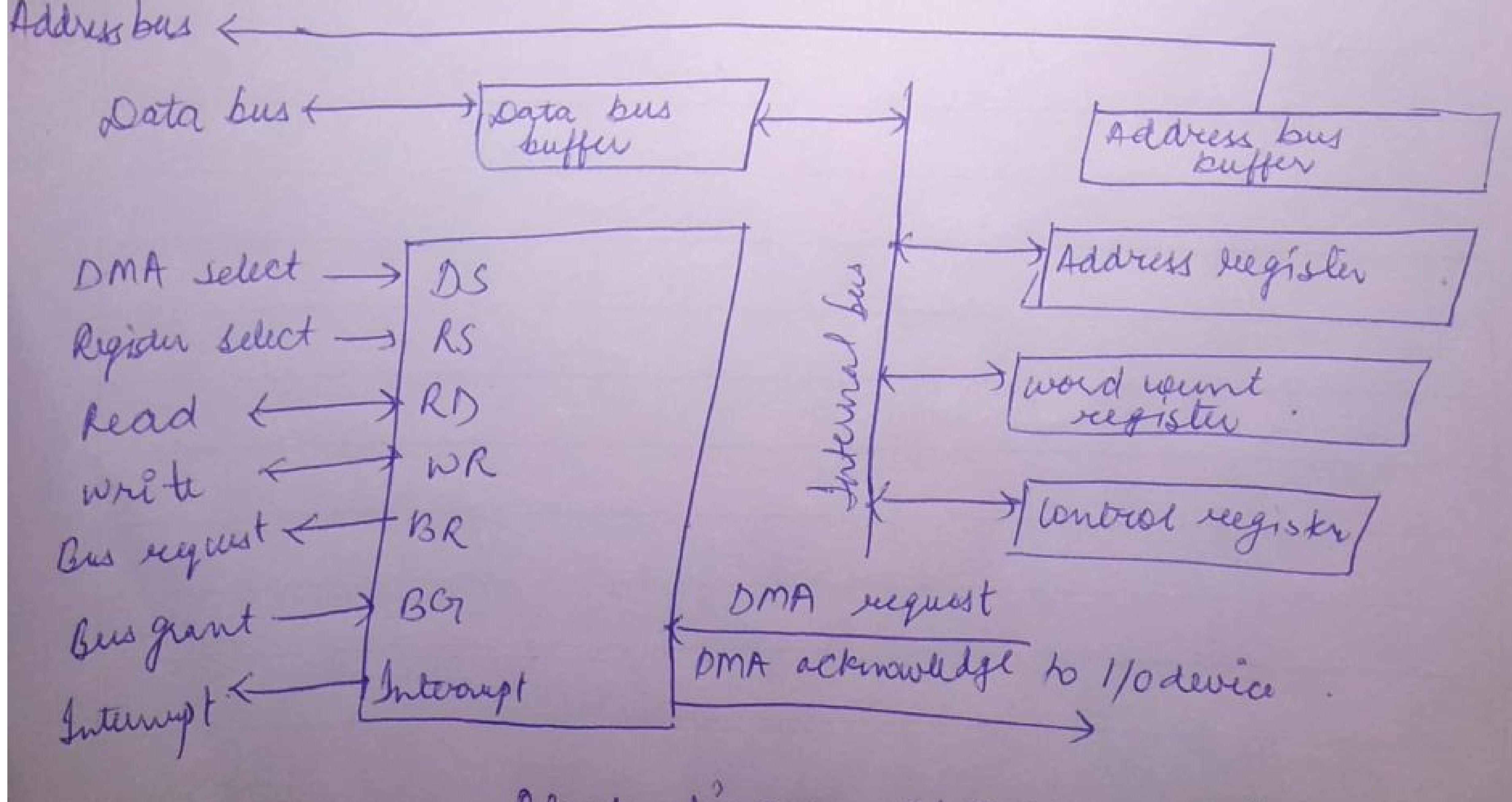
Ques-3

DMA controller.

The DMA controller needs the usual circuit of an interface to communicate with the CPU and I/O device.
In addition, It needs an address register, a word count register, and a set of address lines.
The address register and address lines are used for direct communication with the memory



The word count register specifies the number of words that must be transferred. The data transfer may be done directly between the device and memory under the control of DMA.



Block diagram of DMA controller.

Suruchi
1219316

→ The unit communicates with the CPU via the data bus and control line. The register in DMA are selected by CPU through the address bus by enabling the DS & RS inputs.

→ The RD & WR inputs are bidirectional. When BG input is 0 the CPU can communicate with DMA register through the data bus to ~~by~~ CPU read from or write to DMA register.

→ When BG = 1 the CPU has relinquished the Buses & the DMA can communicate directly with the memory by specifying an address in the address bus & activating the RD or WR control.

→ The DMA controller has three register :-

→ An address register.
→ a word count register.
→ a control register.

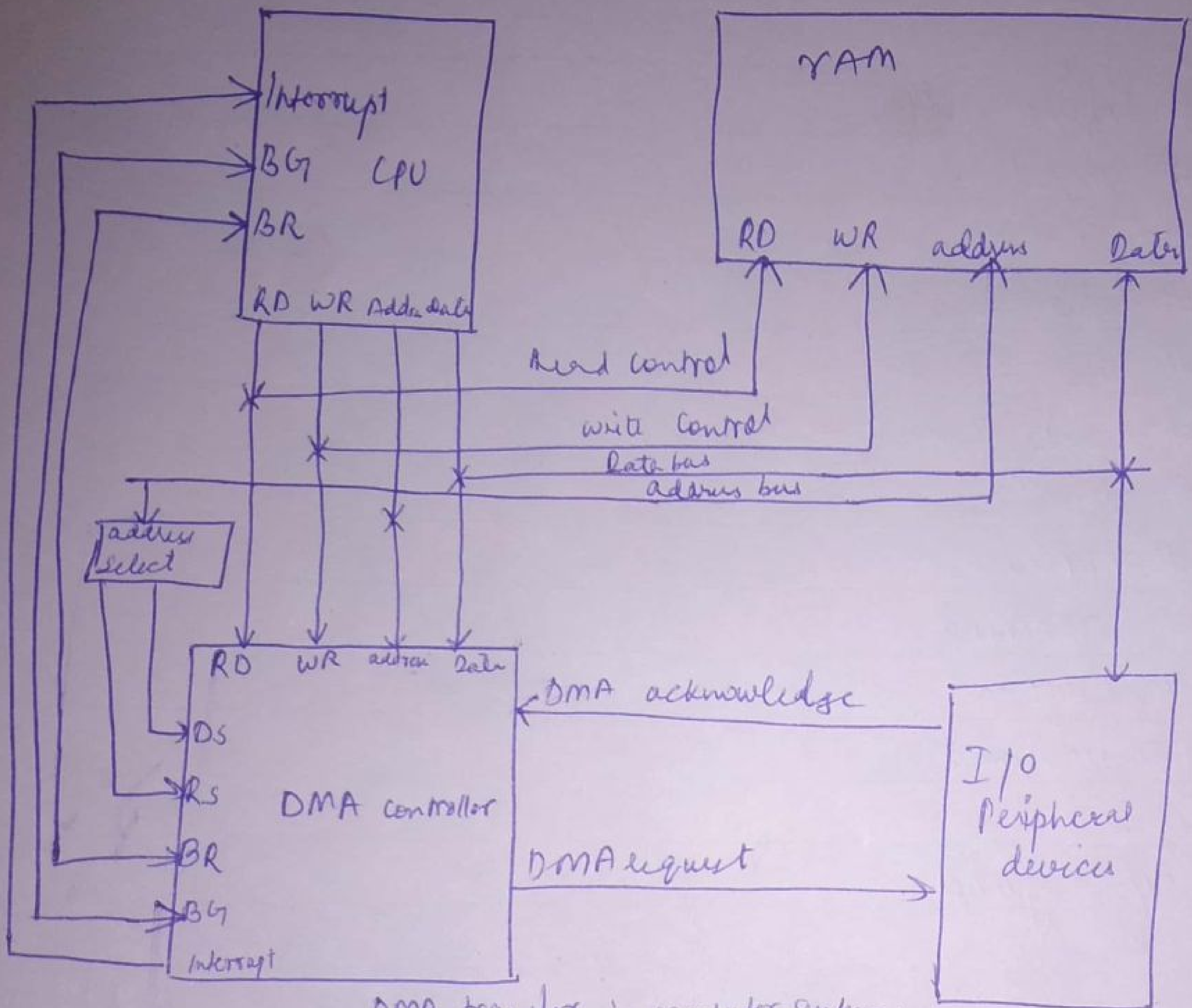
The address register contains an address to specify the desired location in memory. The address bits go through the bus buffers into the address bus.
The word count register holds the number of words to be transferred.

The control register specifies the mode of transfer. Thus the CPU can read from or write into the DMA register under program control via the data bus.

Scanned by CamScanner

Suruchi
1219316

DMA Transfer



DMA transfer in computer system.

→ The CPU communicates with DMA through the address & data bus as with any ~~interfacing~~ interface unit.

→ DMA has its own address, which activates the DS & RS lines. The CPU initializes the DMA through the data bus

→ once the DMA receives the start control command, it can start the transfer between the peripheral devices & the memory

→ when the peripheral devices receives a DMA acknowledge, it puts a word in the data bus or receives a word from the data bus.

Suruchi
12193/6

Ques: 4

Associative memory :-

Many data-processing applications require the search of items in a table stored in memory. As assembler program searches the symbol address table in order to extract the symbol's binary equivalent.

The time required to find an item stored in memory can be reduced considerably if stored data can be identified for access by the content of the data itself rather than by an address.

A memory unit accessed by content is called is called an associative memory or content addressable memory (CAM).

When a word is written in an associative memory, no address is given. The memory is capable of finding an empty unused location to store the word.

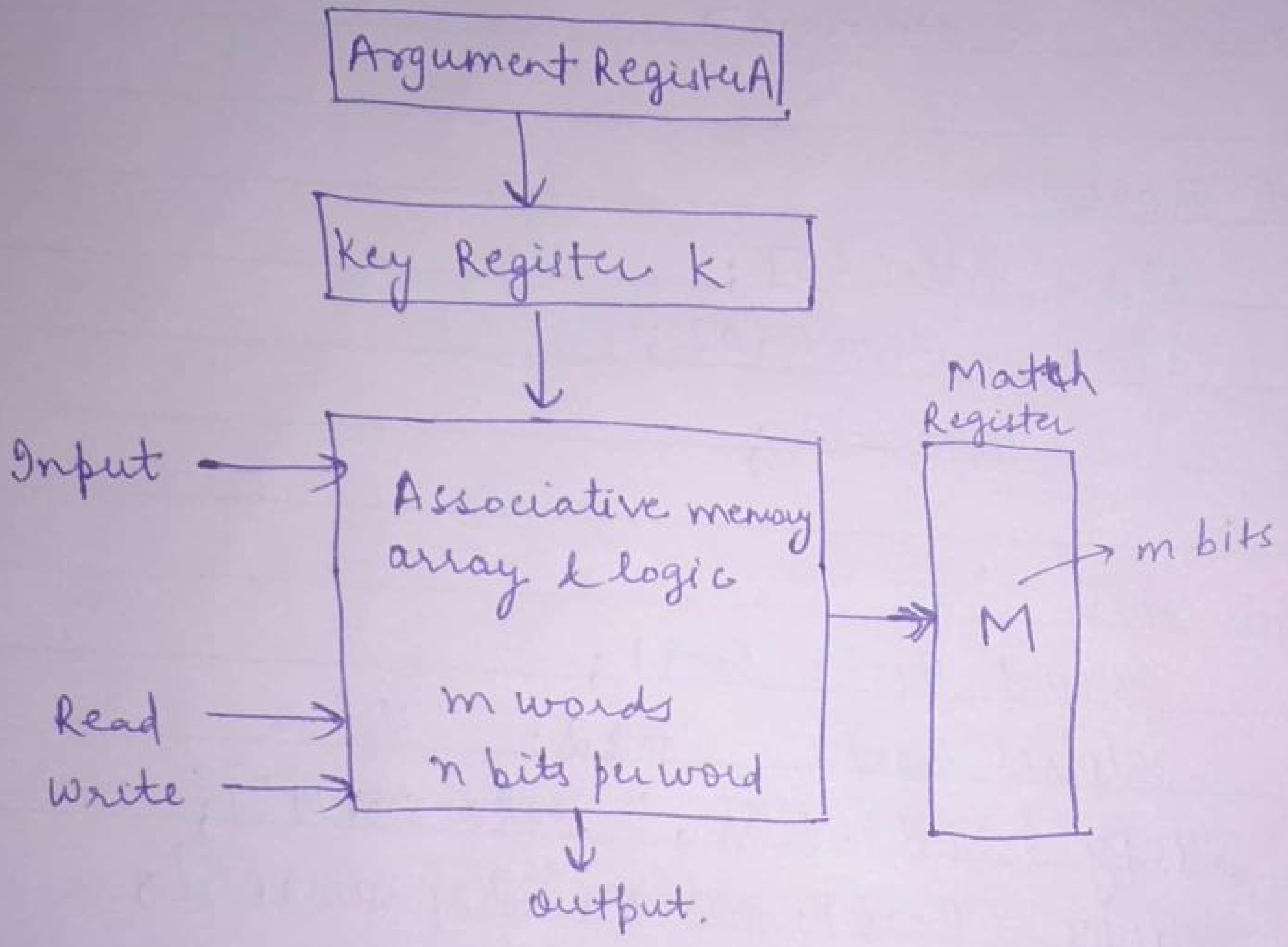
When a word is to be read from an associative memory, the content of the word, or part of the word is specified.

Hardware organization:- It consist of a memory and logic for m words with n bits per word.

The argument register A and key register K each have n bits, one for each bit of a word. The match register M has m bits, for for each memory word. Each word in memory is compared with the

Suruchi
1Q19316

content of the argument register. The words that match the bits of the argument register set a corresponding bit in the match register.



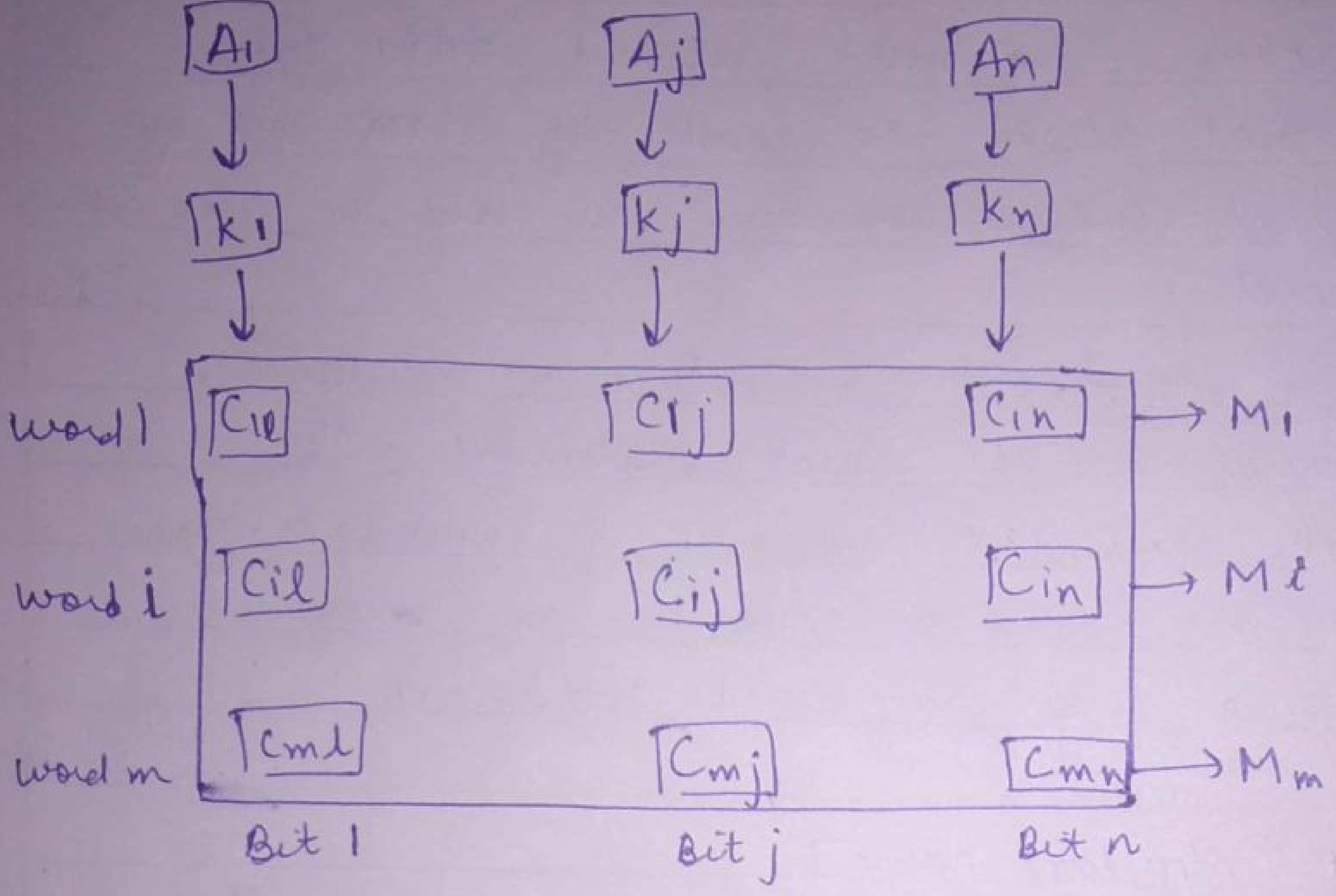
A	101	111100	
K	111	000000	
word1	100	111100	no match
word 2	101	000001	match

word 2 matches the unmasked argument field because the three leftmost bits of the argument and the word are equal.

Scanned by CamScanner

Suruchi
1219316

Associative memory of m words, n cells per word.

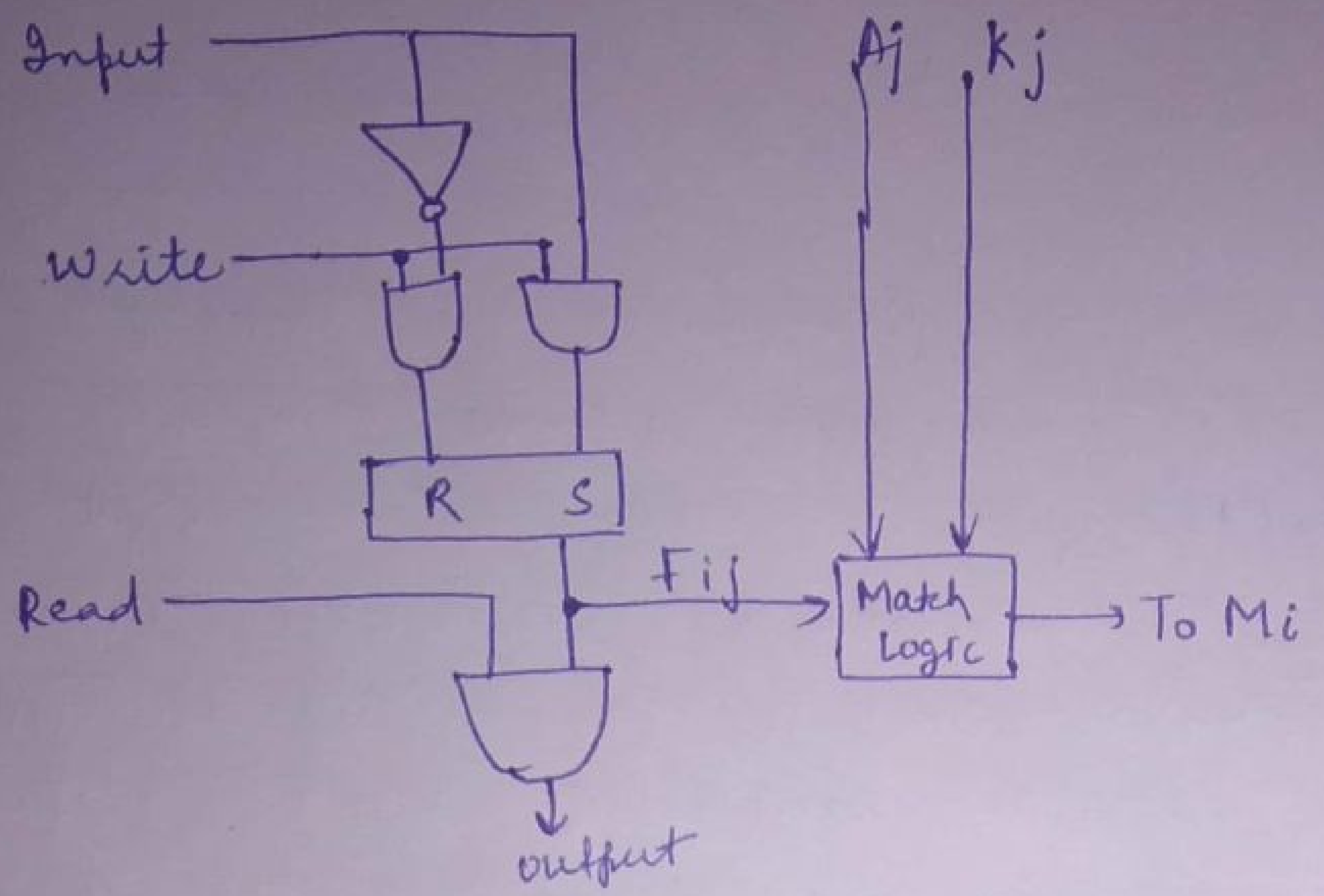


Match logic : The match logic for each word can be derived from the comparision algorithm for two binary numbers. first, we neglect the key bits and compare the argument in A with the bits stored in the cells of the words Word i is equal to the argument in ~~if~~ A if $A_j = F_{ij}$ for $j = 1, 2, \ldots n$. Two bits are equal if they are both 1 or both 0. The equality of two bits can be expressed logically by the boolean function.

$$x_j = A_j F_{ij} + A_j' + F_{ij}'$$

where $x_j = 1$. if the pair of bits in position j are equal ; otherwise, $x_j = 0$.

Suruchi
1219316.



for a word i to be equal to the argument in A, we must have all x_j variables equal to 1. This is the condition for setting the corresponding match bit Mi to 1. The boolean function for this condition is

$$M_i = x_1 x_2 x_3 \cdots x_n$$

and constitutes the AND operation of all pairs of matched bits in a word.