COMPUTER ARITHMETIC

Introduction ⇒.

An arithmetic processor is the part of a processor unit that executes arithmetic operations. The Four basic arithmetic operations are addition, substraction, multiplication and division.

The solution to any problem that is stated by a finite number of well-defined procedural steps is called an algorithm

Addition and substraction

(i) Addition and substraction with signed-magnitude data.

(ii). Addition and substraction with signed-2's complement data.

(i) Addition and substraction with signed-magnitude data

Add ⇒ we designate the magnitude of the two numbers A and B. when the signs of A and B are identical add the two magnitude and attach the sign of A to the Result.

For substraction when the signs of A and B are different, add the two magnitude and attach the sign of A to the result.

Add → when the sign of A and B are different, compare the magnitude. and substract the smaller number from the larger. choose the sign of the result to be the same as A if A>B or the complement of the sign of A if A<B. If the two magnitudes are equal, substract B from A and make the sign of the result positive

Sub → when the sign of A and B are different identical, compare the magnitudes and substract the smaller number from the larges.

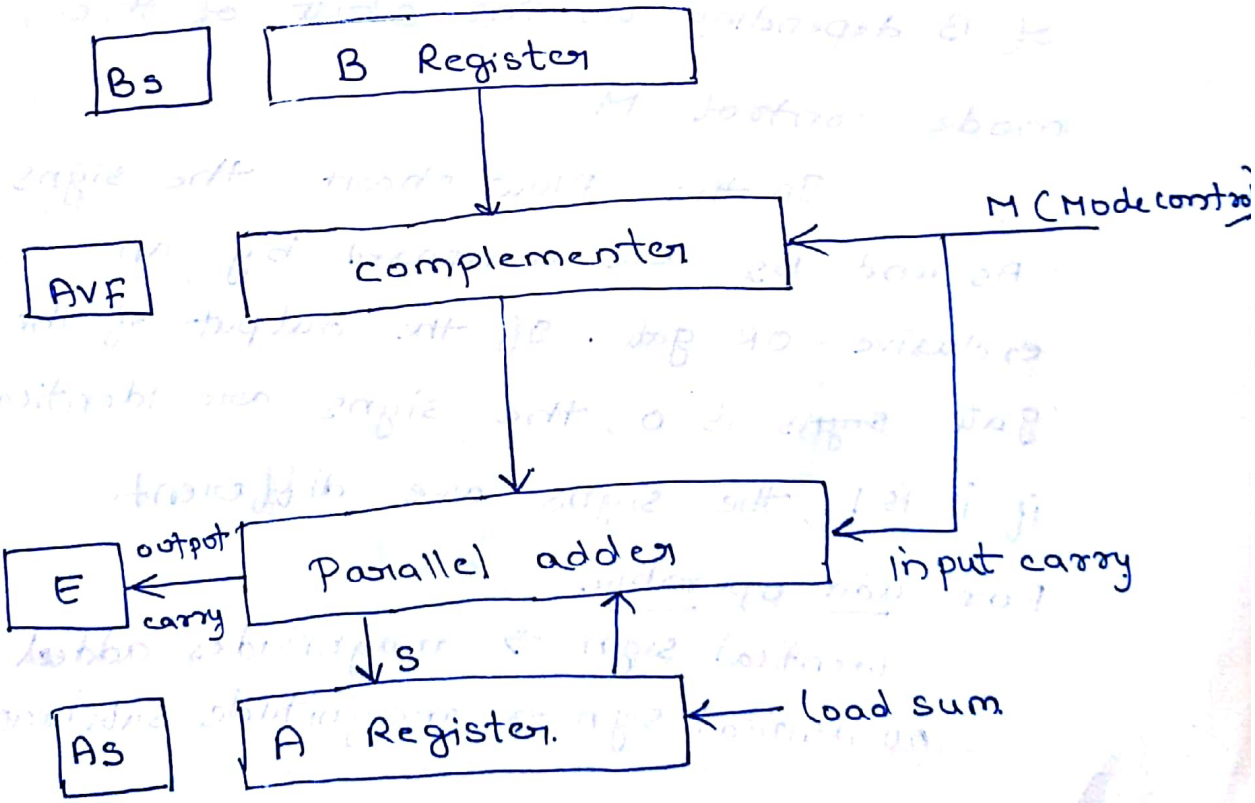
Addition and substraction of signed-magnitude number.

operation	Add magnitude	substract magnitude		
		when A>B	when A<B	when A=B
(+A)+(+B)	+(A+B)			
(+A)+(−B)		+(A−B)	−(B−A)	+(A−B)
(−A)+(+B)		−(A−B)	+(B−A)	+(A−B)
(−A)+(−B)	−(A+B)			
(+A)−(+B)		+(A−B)	−(B−A)	+(A−B)
(+A)−(−B)	+(A+B)			
(−A)−(+B)	−(A+B)			
(−A)−(−B)		−(A−B)	+(B−A)	+(A−B)

Hardware Implementation

Let A and B be two registers that hold the magnitude of the number, and As and Bs two flip-flop that hold the corresponding sign.

1) a parallel adder is needed to perform the microoperation A+B.
2) comparator to see if A>B, A=B, A<B.
3) two parallel substractor circuits are need to perform the microoperation A-B and B-A
4) OR gate for sign



Hardware for signed-magnitude addition and substraction

(4)

It consists of registers A and B and sign flip-flops As and Bs. subtraction is done by adding A to the 2's complement of B. The output carry is transferred to flip-flop E, The add-overflow flip-flop AVF holds the overflow bit when A and B are added.

The addition A plus B is done through parallel adder. The S (sum) output of the adder is applied to the input of the A register. The complementer provides an output B or the complement of B depending on the state of the mode control M.

In the Flow chart the signs As and Bs are compared by an exclusive-OR gate. If the output of the gate ~~signs~~ is 0, the signs are identical if it is 1, the signs are different.

For <u>add</u> operation

identical sign → magnitudes added

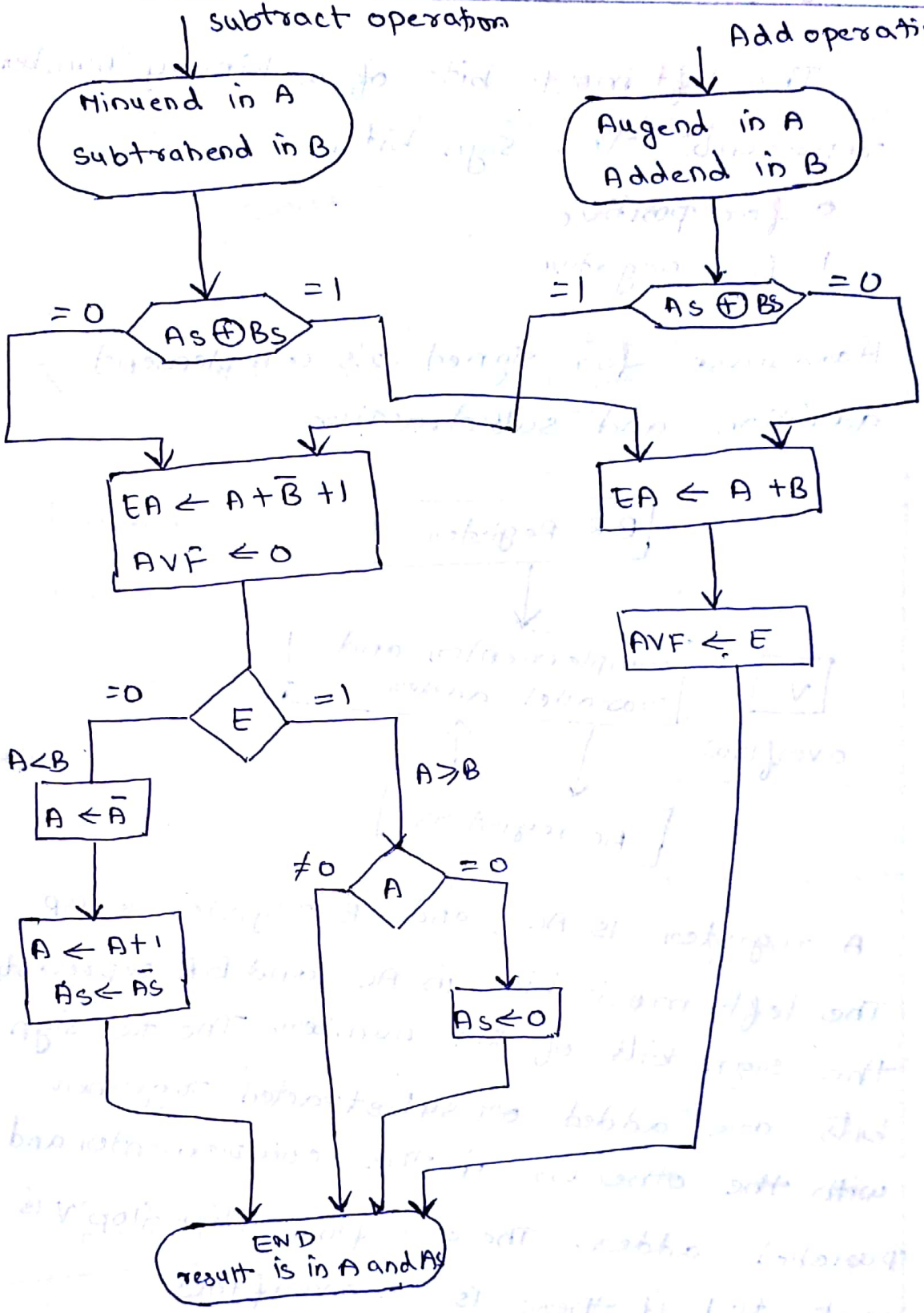
no identical sign → magnitude subtracted

For sub operation

different sign → magnitude added

identical sign → magnitude subtracted

(5)



The value of E determines which number is greater

if E = 1 $A \geq B$.

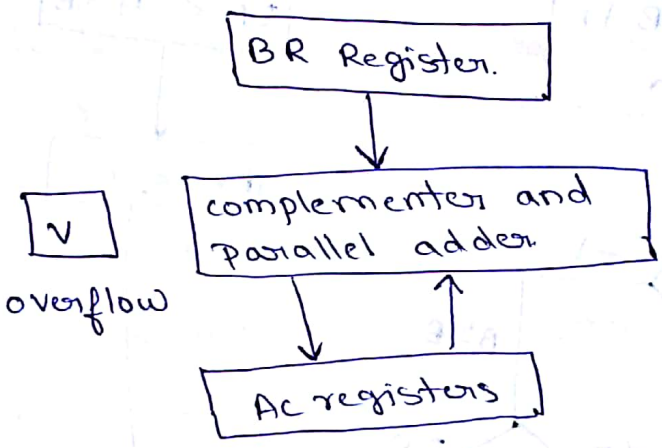
E = 0 $A < B$

Addition and subtraction with 2's Signed - 2's complement data.

The left most bit of a binary number represents the sign bit.

- 0 for positive.
- 1 for negative.

Hardware for signed - 2's complement addition and subtraction.

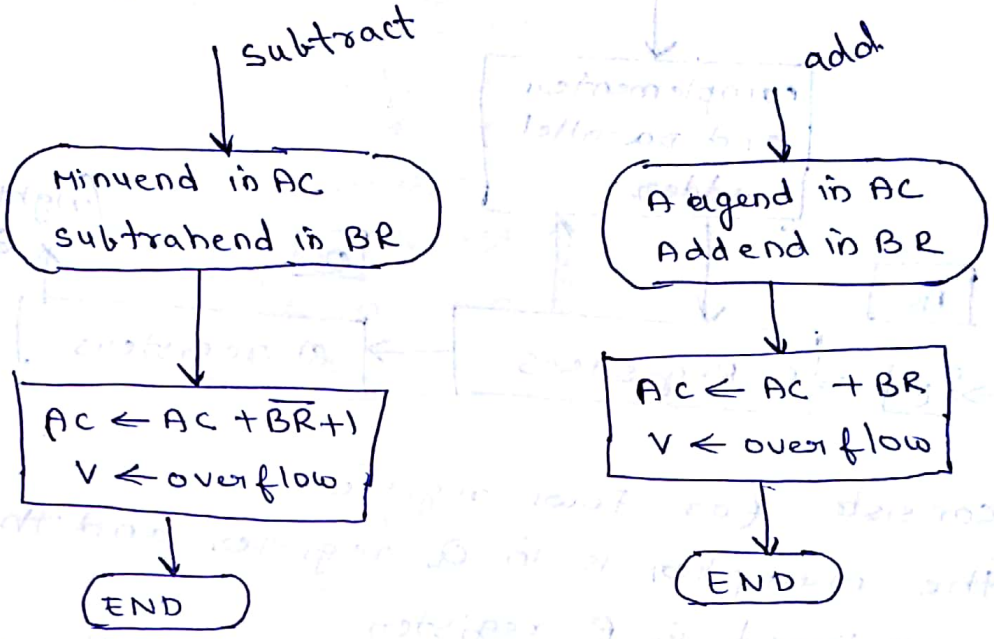


A register is AC, and B register is BR. The left most bit in AC and BR represents the sign bits of the number. The two sign bits are added or substracted together with the other bit in the complementer and parallel adder. The overflow flip-flop V is set to 1 if there is an overflow

In the algorithm the sum is obtained by adding AC and BR including their sign bits. The substraction operation is obtained by adding AC to the 2's complement of BR

(7)

Algorithm for adding and substracting numbers in signed 2's complement representation



Multiplication Algorithm

If we multiply 23 and 19 first we have to convert them to binary.

```
23 - 10111
19 - 10011
-----------
   1 0 1 1 1
  1 0 1 1 1
 0 0 0 0 0
0 0 0 0 0
1 0 1 1 1
-----------
1 1 0 1 1 0 1 0 1  → 437
```

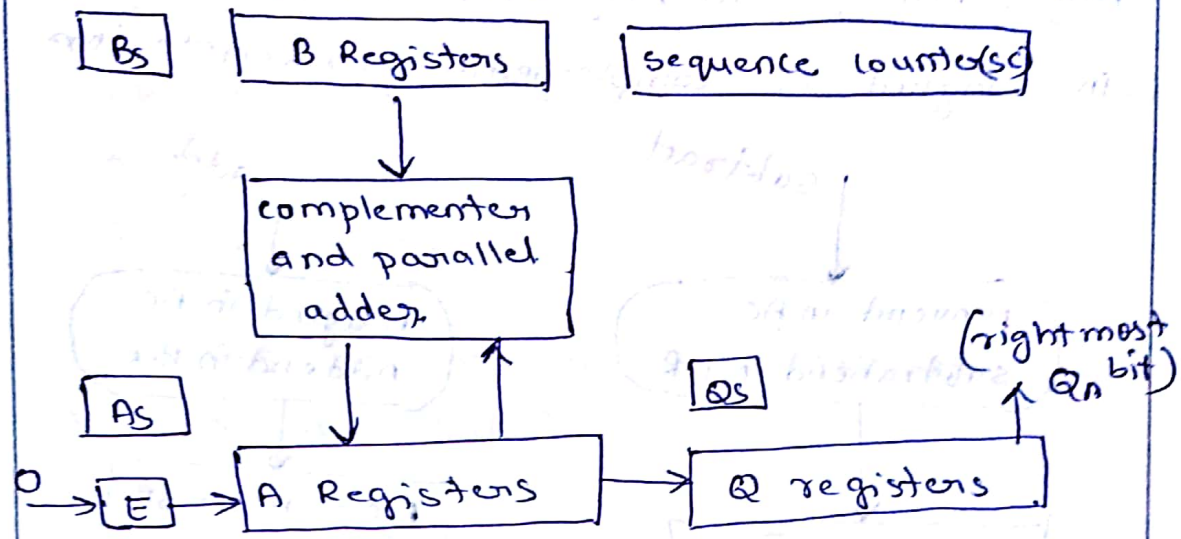
when a computer has to do multiplication it can be done in two ways

① The sign bit of the two number are handled separately

② The sign bit is handled in the Question itself (BOOTH ALGORITHM)

⑥

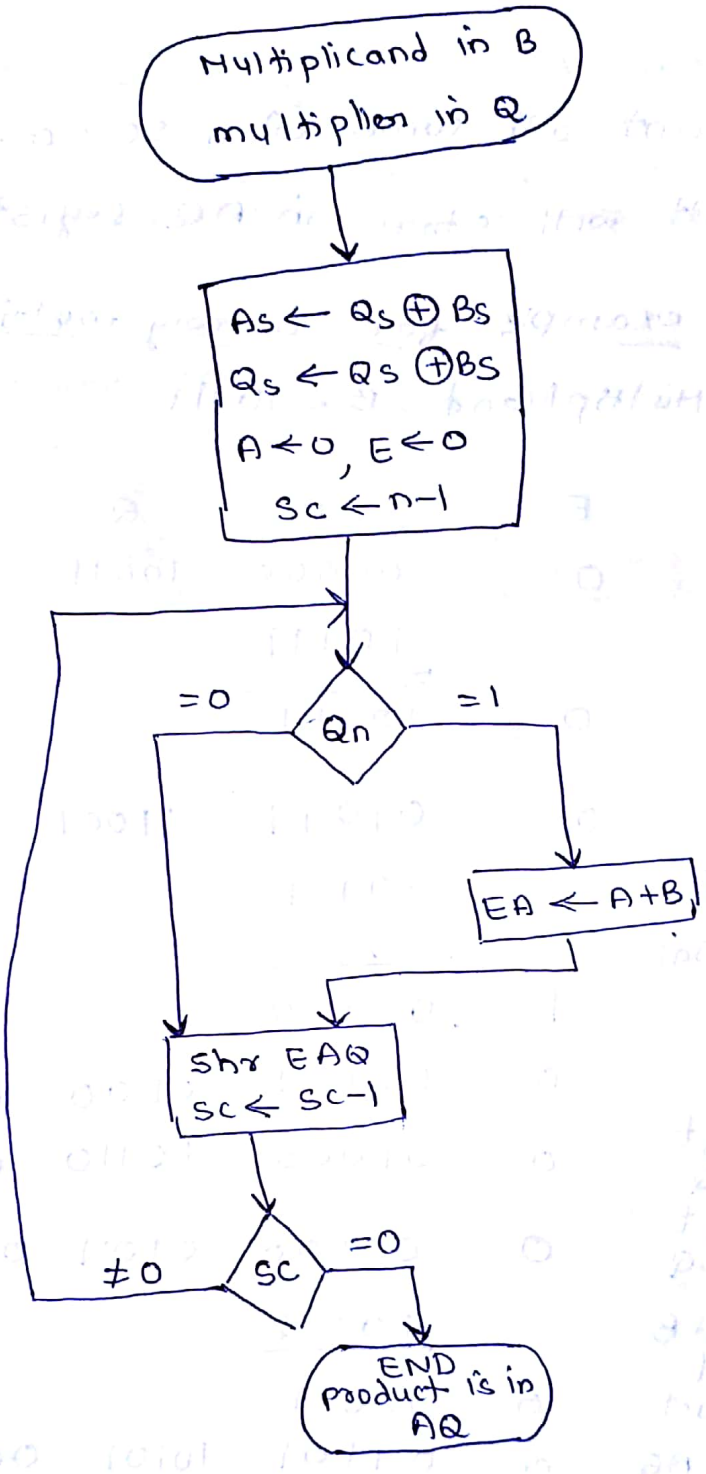
Hardware for multiply operation



consists for Four registers
the multiplier is in Q register and the multiplicand in B register
The sequence counter sc is initially set to a number equal to the number of bits in the multiplier. The counter is decremented by 1 after forming each Partial product. when the content of the counter reaches zero, the product is formed and the process stops.

Sign bit are handled separately so first finding the XOR of sign bit As, and Qs
sequence counter sc is set to a number equal to the number of bits of the multiplier. operand is stored with its sign so n-1 bits

(9)



Qn is the last bit of Q. If it is 1 the value of B will be added to A and the result will be in EA. The shift right will be performed If the last bit Qn is 0 shift right will be performed in EAQ.

After shift right (shr) the SC is decremented by 1.

once a point will come when SC = 0. The Final result will store in AQ register.

Numerical example for Binary multiplier

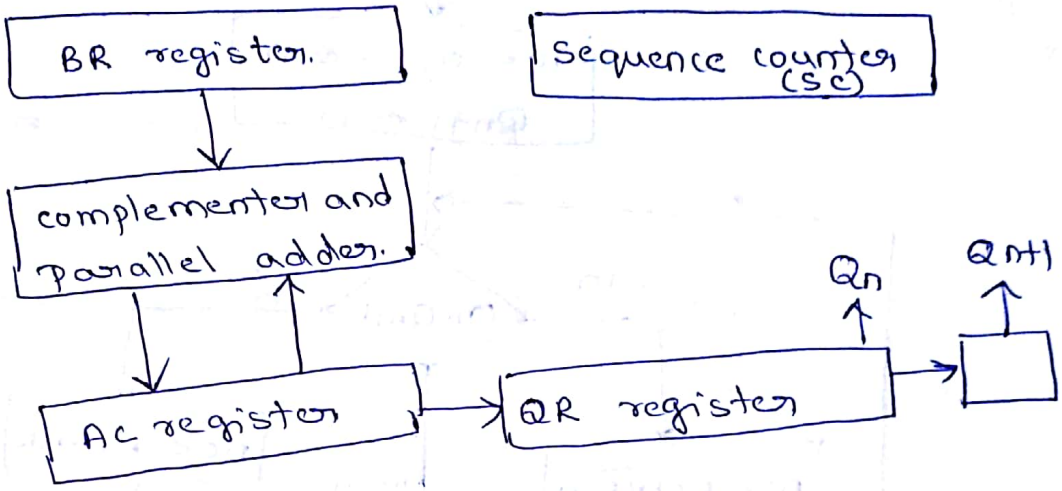
Multiplicand B = 10111

	E	A	Q	SC
Multiplier in Q	0	00000	10011	101
Qn = 1; add B		10111		
first partial Product	0	10111		
shift right EAQ	0	01011	11001	100
Qn = 1, add B		10111		
second partial Product	1	00010		
shift right EAQ	0	10001	01100	011
Qn = 0, shift right EAQ	0	01000	10110	010
Qn = 0, shift right EAQ	0	00100	01011	001
Qn = 1, add B		10111		
Fifth partial Product	0	11011		
shift right EAQ	0	01101	10101	000

Final product in AQ = 0110110101

(11) Booth Algorithm for multiplication of signed -2's complement number.

Hard ware for Booth Algorithm



Booth algorithm gives a procedure for multiplying binary integers in signed -2's complement representation. The algorithm works for positive or negative multipliers in 2's complement representation.

The registers are AC, BR and QR.

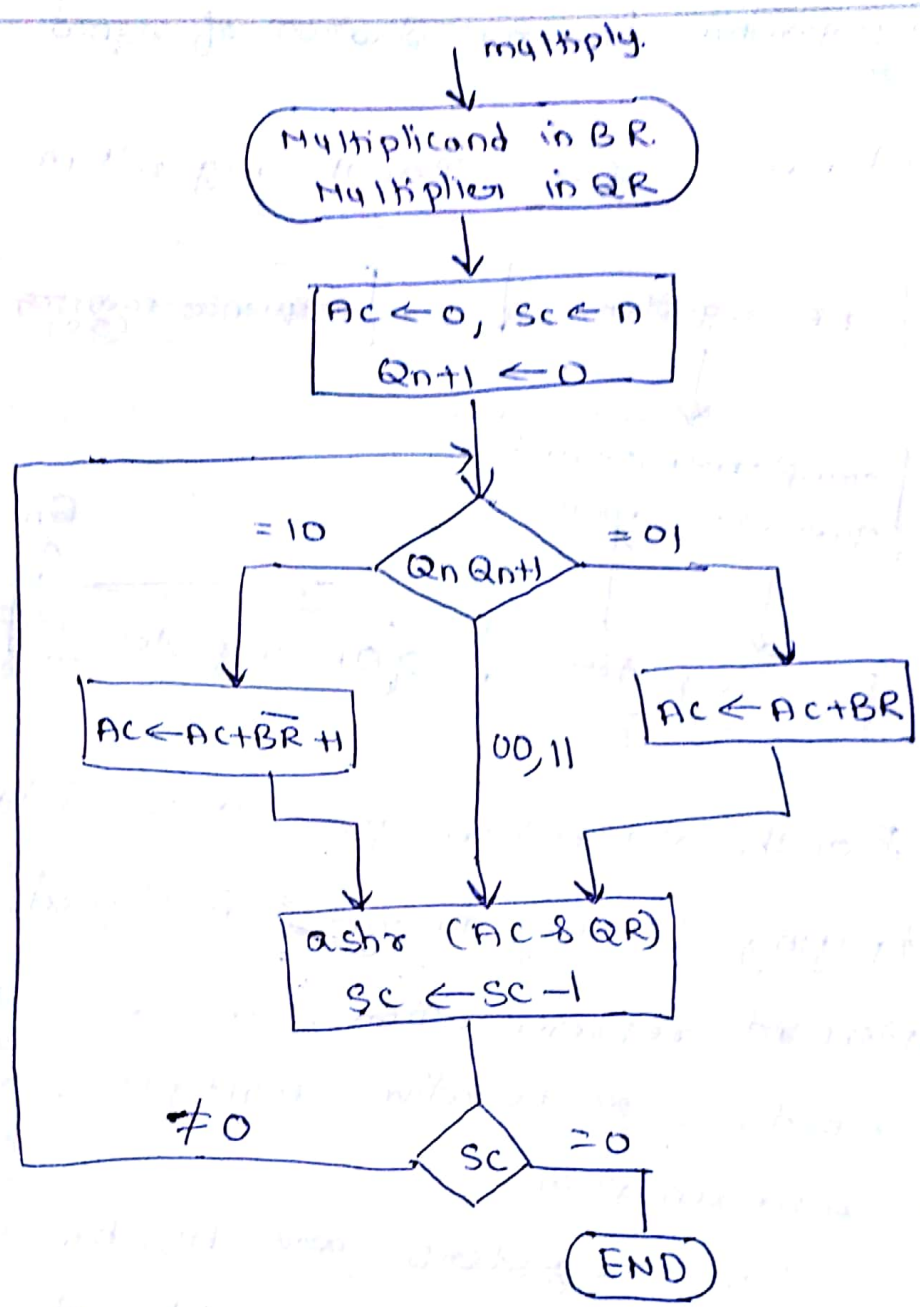
Qn designates the least significant bit of the multiplier in register QR. An extra flip-flop Qn+1 is appended to QR

numbers are represented by BR and QR because B and Q have the sign bit also.

Ac is initially set to 0

Qn+1 is zero.

SC is set to n



Here we have to check the last bits of the numbers Qn and Qn+1

If $Q_n Q_{n+1} = 01$ then the addition is performed

If $Q_n Q_{n+1} = 10$
then subtraction is performed.

for 00 and 01 arithmetic shift is performed.

(13)

step by step multiplication of $(-9) \times (-13) = +117$

$+9 \rightarrow 1001 \rightarrow 0110 + 1 = 0111$

$-9 \Rightarrow 10111$

$13 \Rightarrow 1101 \rightarrow 0010 + 1 = 0011$

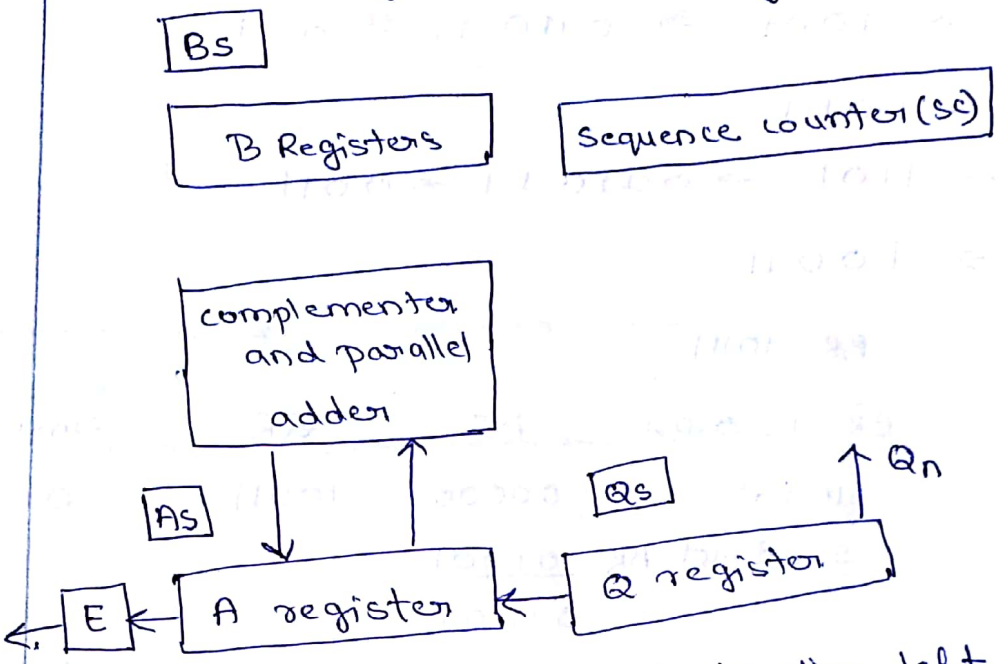
$-13 \Rightarrow 10011$

BR = 10111

QnQn+1	$\overline{BR} + 1 = 01001$	AC	QR	Qn+1	SC
	initial	00000	10011	0	101
1 0	substract BR	01001			
		01001			
	ashr	00100	11001	1	100
1 1	ashr	00010	01100	1	011
0 1	Add BR	10111			
		11001			
	ashr	11100	10110	0	010
0 0	ashr	11110	01011	0	001
1 0	substract BR	01001			
		00111			
	ashr	00011	10101	1	000

(14) DIVISION ALGORITHMS

Hardware for Division algorithm



Register EAQ is shifted to the left with 0 inserted into Qn and the previous value of E lost.

Hardware Algorithm

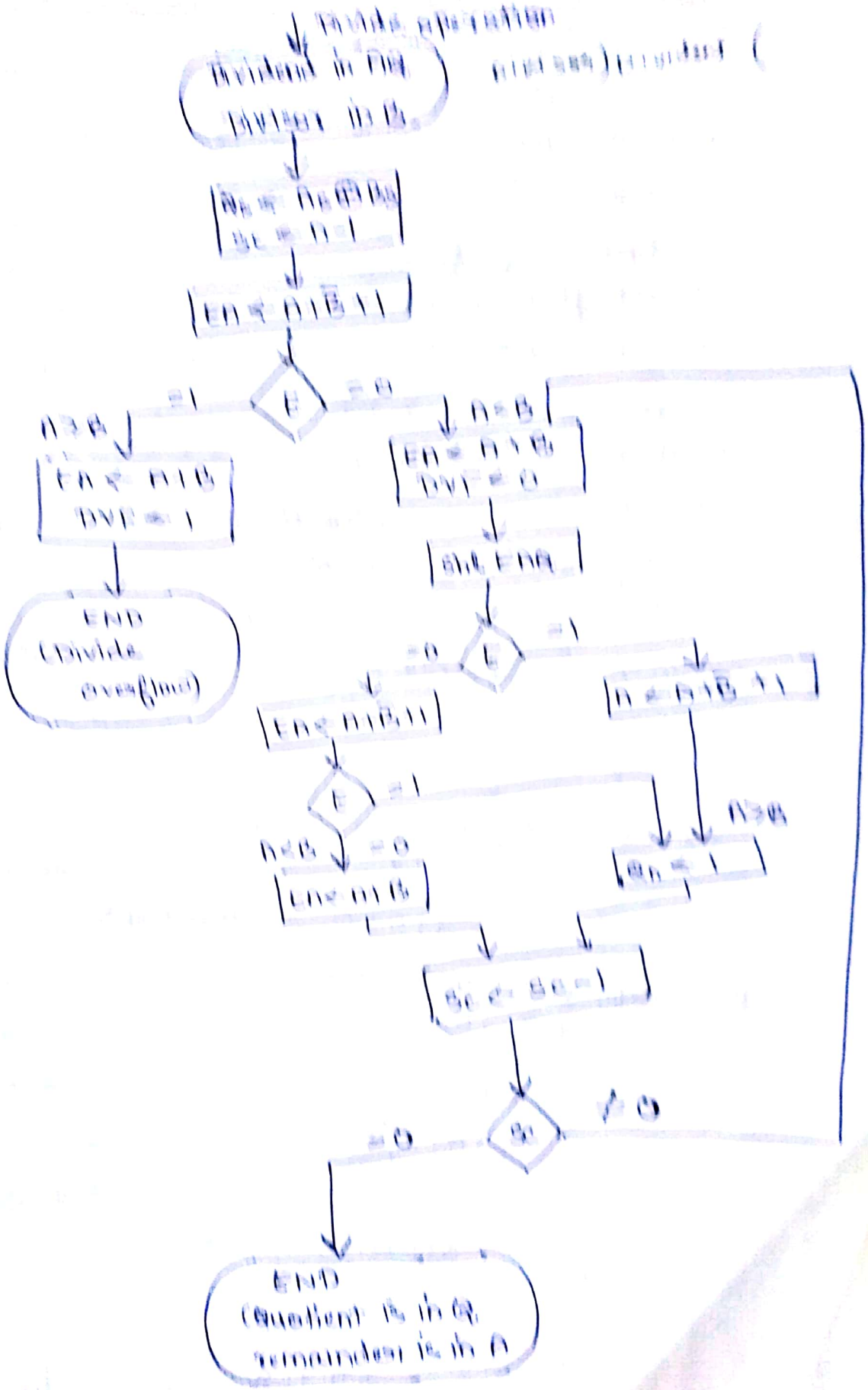
The divident is in A Q and the divisor in B sign bits are determined by XOR gates

$$Qs \leftarrow As \oplus Bs$$

To compare the ~~first~~ bit of the dividend and divisor, substraction is performed

ie $EA \leftarrow A + \bar{B} + 1$

$E = 1 \rightarrow$ means an overflow ie dividend is greator than Divisor and it will result in divide overflow condition and it will end.



E = 0 means no overflow.

means the dividend first bit are smaller than divisor and the divison process will continue.

→ Shl EAQ is performed

→ check if E=0 or E=1

If E=0 substraction is performed and result is stored in EA. again E value is checked if E=1, Qn value is set to 1

If E=0 then the value of B is added to A ie to reset the value of A which was changed after substraction

Then the sequence counter is decremented by 1.

Example of binary Division.

```
                  11010  → quotient
              ------------
B = 10001) 0 11100 00000
           @ 1110
             01110 0
           - 10001
           ---------
             01011 0
           -  10001
           ---------
             001010
           -- 10001
           ---------
              01001 0
           --- 10001
           ---------
               00011 0
                00110  - remainder
```

(17)

Divisor B = 10001 $\quad \bar{B}+1 = 01111$

	E	A	Q	SC
Dividend:		01110	00000	5
Shl EAQ	0	11100	00000	
add $\bar{B}+1$		01111		
E = 1	1	01011		
set Qn = 1	1	01011	00001	4
Shl EAQ	0	10110	00010	
Add $\bar{B}+1$		01111		
E = 1	1	00101		
set Qn = 1	1	00101	00011	3
Shl EAQ	0	01010	00110	
ADD $\bar{B}+1$		01111		
E = 0	0	11001		
leave Qn = 0	0	11001	00110	
ADD B		10001		2
↑ Restore remainder.	1	01010		
Shl EAQ	0	10100	01100	
ADD $\bar{B}+1$		01111		
E = 1	1	00011		
Set Qn = 1	1	00011	01101	1
Shl EAQ	0	00110	11010	
ADD $\bar{B}+1$		01111		
E = 0 ; leave Qn = 0	0	10101	11010	
Add B		10001		
Restore remainder	1	00110	11010	0

Neglect E

Remainder in A → 00110

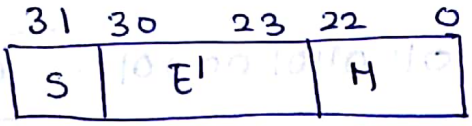
Quotient in Q → 11010

FLOATING POINT ARITHMETIC OPERATION

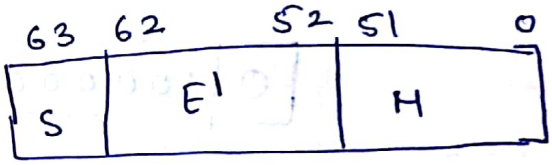
The standard for representing Floating point is 32bit and 64 bit Developed by IEEE Based on the total number of bits they are. single precision or double precision

Floating point representation

Single precision
32bits



Double precisio
64 bits



M = mantissa, s = sign
E' = E+bias.
bias is a number that is added in both single and double precision.

Single - 127
double - 1023

exg → $(1460.125)_{10}$
converting in binary
$= (10110110100.001)_2$.
Normalize it
$1.011\,0110100 \times 2^{10}$
number is positive.
S = 0
E = 10
M = 01101100001
$E' = 10 + 127 = (137)_{10}$

$(137)_{10} = (10001001)_2$

0	10001001	01101100001 --- 0

Double precision

$S = 10$

$E = 10$

$E' = 10 + 1023 = 1033$

$(1033)_{10} = (1\ 0000001001)_2$

0	10000001001	01101100001 --- 0

Floating point number in computer register consists of two parts a mantissa m and exponent e $\rightarrow m \times r^e$

The Floating point number has a 0 in the most significant position of the mantissa is said to have an UNDERFLOW

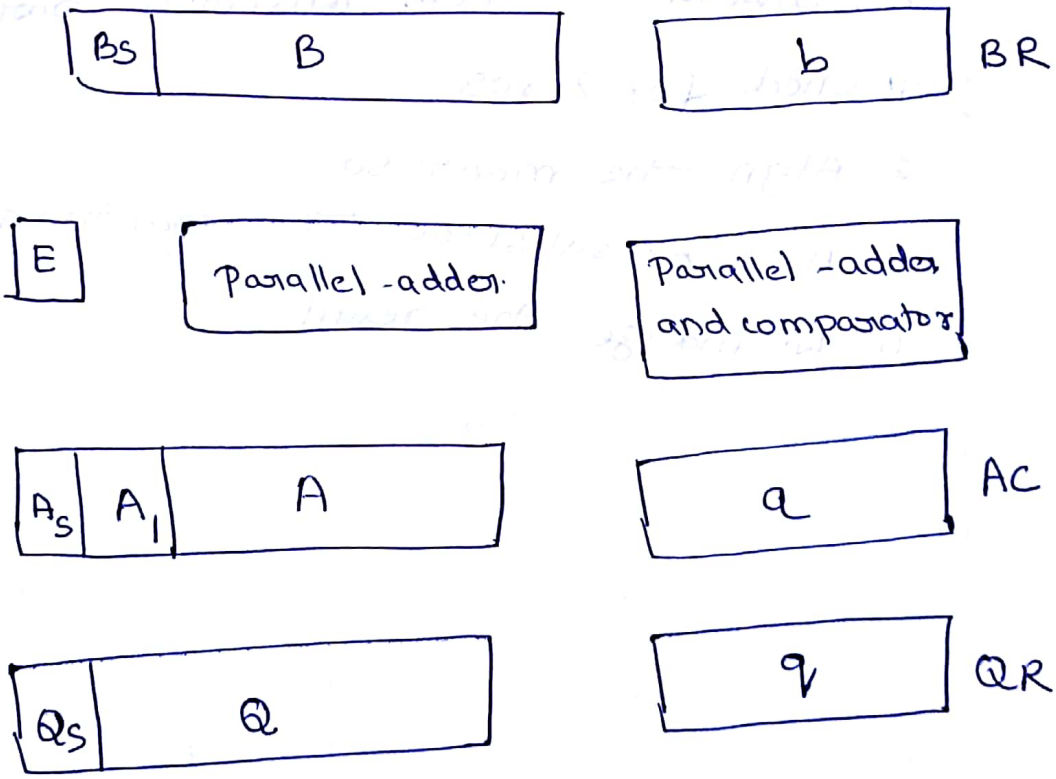
To normalize a number than contains an underflow, it is necessary to shift the mantissa to the left and decrement the exponent until a non zero digit appears in the first position

Register configuration

The register configuration for Floating point operation is quite similar to the layout for Fixed point operation. As a general rule. the same register and adder used for fixed point arithmetic are used for processing the. mantissa. The difference lies in the way the. exponents are handled.

Register organization

There are three registers BR, AC and QR. Each register is sub divided into two parts



The mantissa part has same upper case. letters., the exponent part uses the corresponding lower case letters.

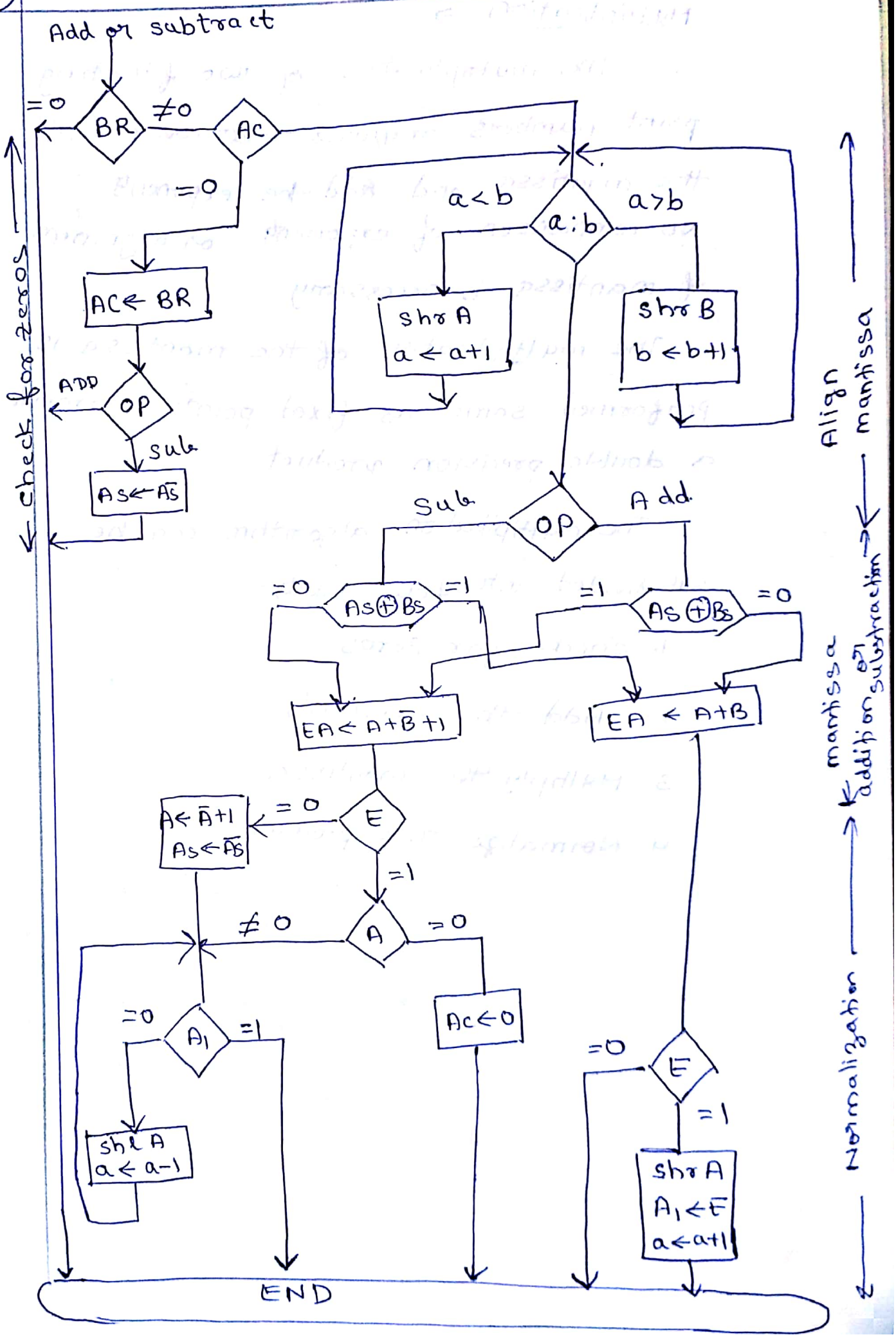
The parallel adder adds the two mantissas and transfers the sum into A and the carry into E.

A separate parallel adder is used for the exponents since the exponents are biased.

ADDITION AND SUBTRACTION

During addition and substraction, the two floating point operands are in AC and BR. The sum of difference is formed in the AC. The algorithm can be divided into four consecutive parts

1. Check for zeros
2. Align the mantissa
3. Add or substract the mantissa.
4. Normalize the result



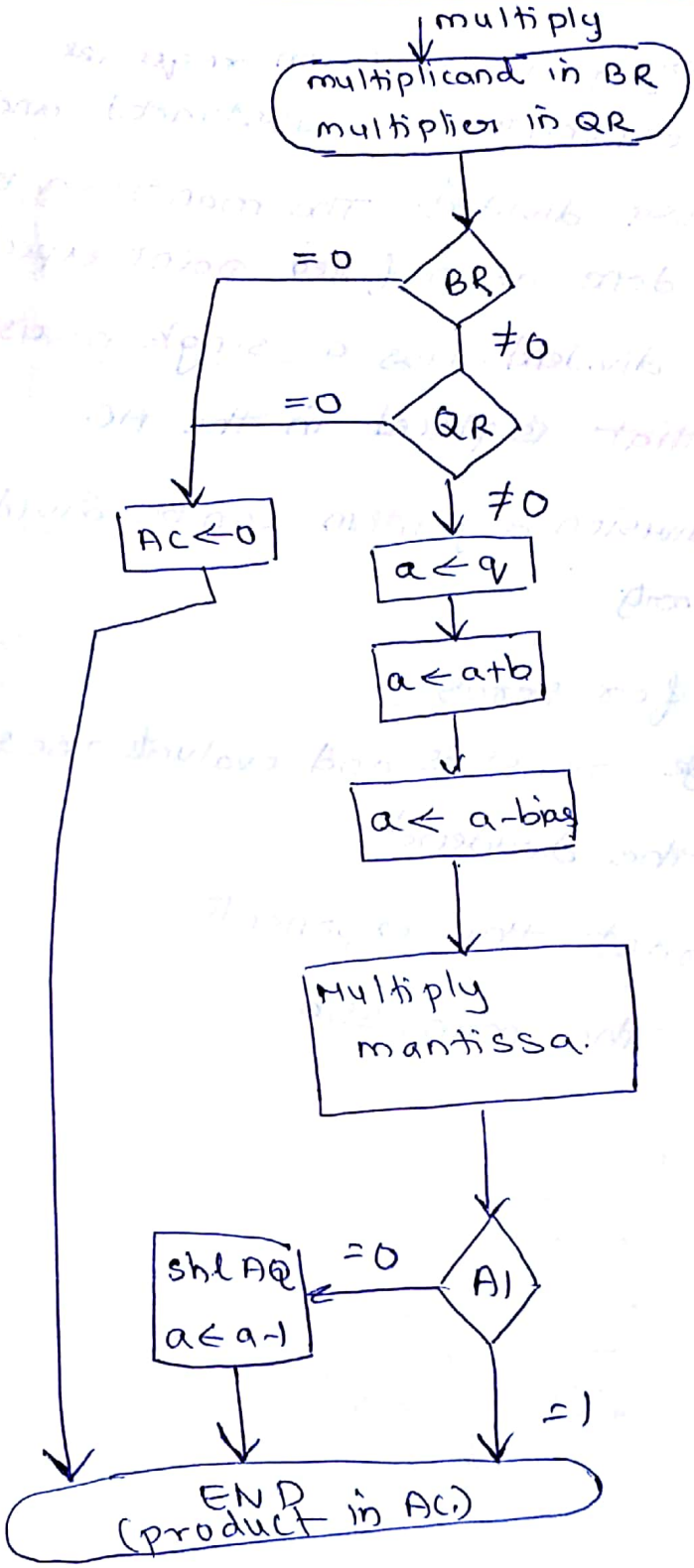
Multiplication ⇒

The multiplication of two floating point numbers requires that we multiply the mantissa and add the exponents.
No comparison of exponents or alignment of mantissa is necessary

The multiplication of the mantissa is performed same as fixed point to provide a double precision product.

The multiplication algorithm can be subdivided into four parts.

1. checks for zeros
2. Add the exponents
3. Multiply the mantissa.
4. Normalize the product.

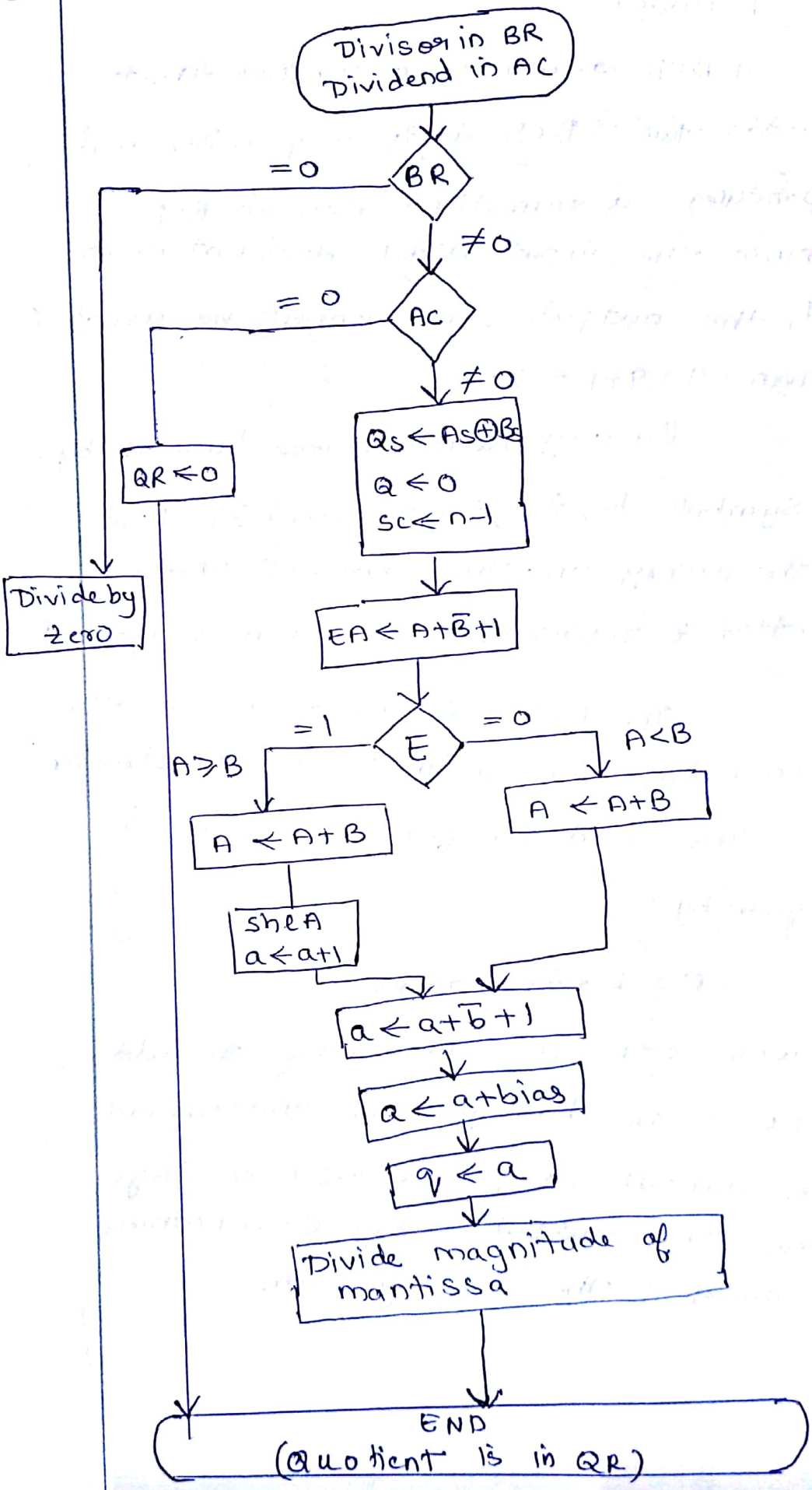


DIVISION

Floating point division requires that the exponents be substracted and the mantissa divided. The mantissa is division is done as in fixed point expect that the dividend has a singk precision mantissa that is placed in the AC.

The division algorithm can be divided into five parts

1. check for zeros
2. intialize registers and evaluate the sign
3. Align the dividend
4. substract the exponents.
5. Divide the mantissa.



INPUT - OUTPUT ORGANIZATION

~~Pheripheral Devices~~

Peripheral devices

A computer device, such as a CD-ROM drive or printer, that is not part of the essential computer ie, the memory and microprocessor.

Peripheral devices are external. such as a mouse, keyboard, printer., monitor external zip drive or scanner. or. internal such as CD-ROM drive. CD-R drive. or internal modem.

Internal peripheral devices are often referred to as integrated peripherals

Types of peripheral devices

There are three types of peripheral devices

→ input devices
→ output devices
→ storage devices

Input devices →

electronic devices that are connected to a computer and produce. input signals

Keyboard
scanner.
Microphone.
Mouse

output devices ⇒.

output device is the result of data processi activity when it is presented externally to the system.

The output from a computer can be printed or displayed form

An output device is hardware that is capable. of delivering or showing information to one more users

An output device shows, prints and presents the result of a computer's work.

Monitor.
Printer.
speaker
Projector.

storage device

Information and documents are stored a computer storage so that it can be retriev whenever they are needed later on

computer storage means holding the data in electromagnetic form. for access by a computer processor.

Hark Disk, ROM, thumb drive.

Mouse and Keyboard

→ A characteristic feature of display devices is a cursor that marks the position in the screen where next character will be inserted

→ The display terminal can operate in a single-character mode where all characters entered on the screen through keyboard are transmitted to computer.

Printer

Printer provide a permanent record on paper of computer output data or text

There are three basic types of character printer

daisy wheel
dot matrix.
laser printers

The daisywheel printer contains a wheel with the characters placed along the circumference. To print a character, the wheel rotates to proper position

The dot matrix printer contains a set of dots along the printing mechanism

The laser printer uses a rotating photographic drum that is used to imprint the character images

The pattern is then transferred onto paper in the same manner as a copying machine.

Magnetic tape-

→ Magnetic tapes are used mostly for storing files of data.

→ Access is sequential and consists of record that can be accessed one after another as the tape moves along a stationary read-write mechanism.

→ It is one of the cheapest and slowest method for storage and has the advantage that tape can be removed when not in use.

Magnetic disk

magnetic disks have high speed rotational surfaces coated with magnetic material. Access is achieved by moving a read-write mechanism to a track in the magnetized surface. Disks are used mostly for bulk storage of programs and data.

INPUT- OUTPUT Interfaces

Input - output interface provides a method for transferring information between internal storage and external I/O devices

Peripherals connected to computer need special communication links for interfacing them with the central processing unit

The purpose of the communication link is to resolve the differences that exist between the central computer and each peripheral.

Peripherals are electromechanical devices. But CPU and memory are electronic devices. Therefore conversion of signal values may be required

Data codes and formats in the peripherals differ from the word format in CPU and memory

Data transfer rate of peripherals are slower than CPU., so synchronization may be needed.

The operating modes of peripherals are different. so they must be controlled so as not to disturb the operation of other pheripherals that are connected to CPU

I/o bus and interface modules

The I/o bus consists of data lines, address lines and control lines.

The I/o bus from the processor is attached to all peripherals interface.

To communicate with a particular device, the processor places a device address on address lines.

Each interface decodes the address and control received from the I/o bus, interprets them for peripherals and provides signals for the peripheral controller.

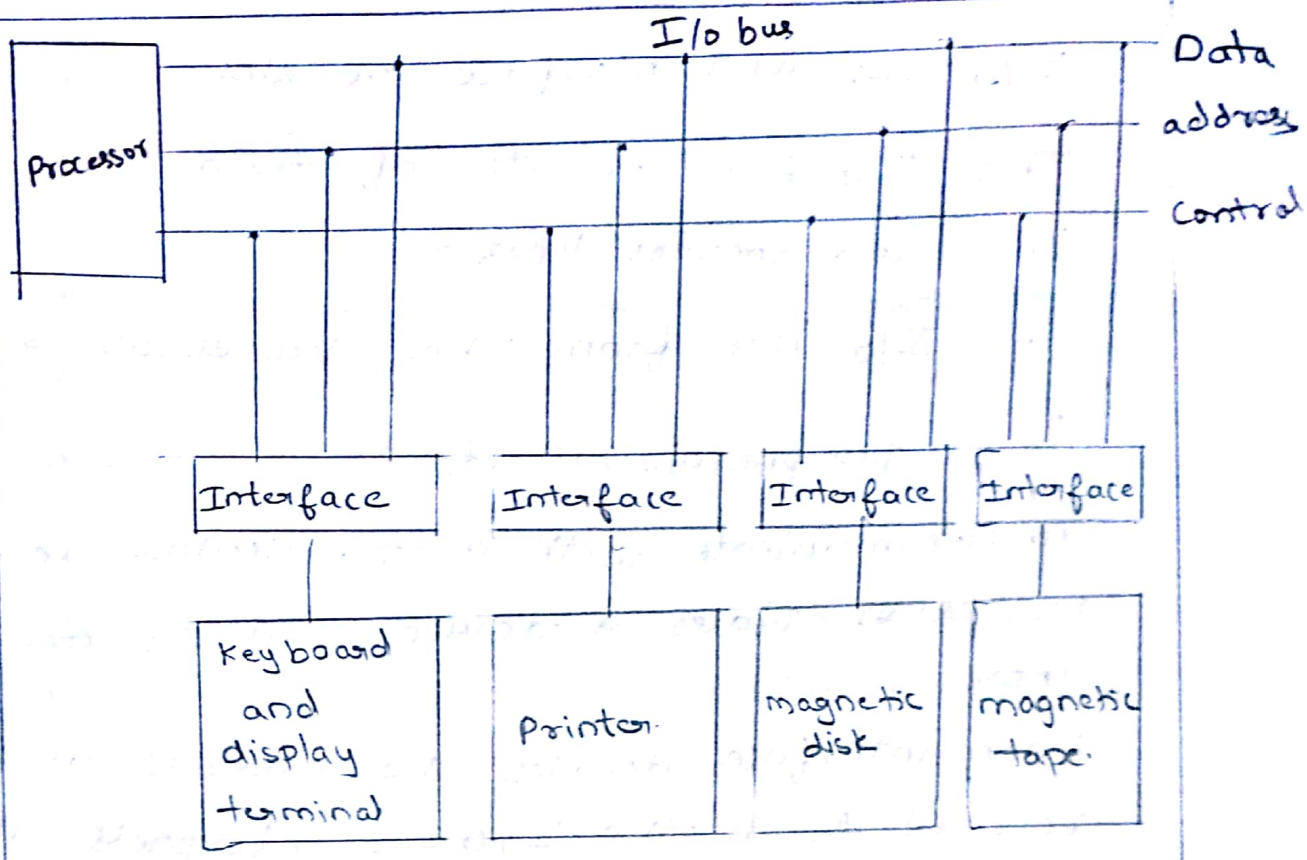
It also synchronizes the data flow and supervises the transfer between peripheral and processor. Each peripheral has its own controller.

For example the printer controller controls the paper motion, the print timing

The control lines are referred as an I/o command. The commands are as follow

control command → A control command is issued to activate the peripheral and to inform it what to do

status command → A status command is used to test various status conditions in the interface and the peripheral.



output data command → A data out put command causes the interface to respond by transferring data from the bus into one of its registers

Input data command → The data input command is the opposite of the data output. In this case the ~~peripheral~~ interface receives an item of data from the peripheral and places it in its buffer region

I/O versus memory bus.

The processor must communicate with the memory unit like the I/O bus, the memory bus contains data, address, and read/write control lines.

There are three ways that computer buses can be used to communicate with memory and I/O

uses use two separate buses, one for memory and the other for I/O

use one common bus for both memory and I/O but have separate control lines for each isolated I/O or mapped I/O

use one common bus for memory and I/O with common control lines

Isolated versus memory-mapped I/O

Isolated I/O →

The isolated I/O configuration separates all I/O interfaces addresses from the memory addresses.

~~it has~~ CPU has distinct input and output instruction

the memory address and I/O address have its own address space

If the address of interface registers are placed on the address lines the I/O read or I/O write control lines are enabled

If the memory address is placed on the address lines the memory read and memory write control lines are enabled

Memory - mapped I/O →

→ In this configuration same address space is used for both memory and I/O. There are no specific I/O instruction

It allows the computer to use the same instructions for both I/O transfers and memory transfers

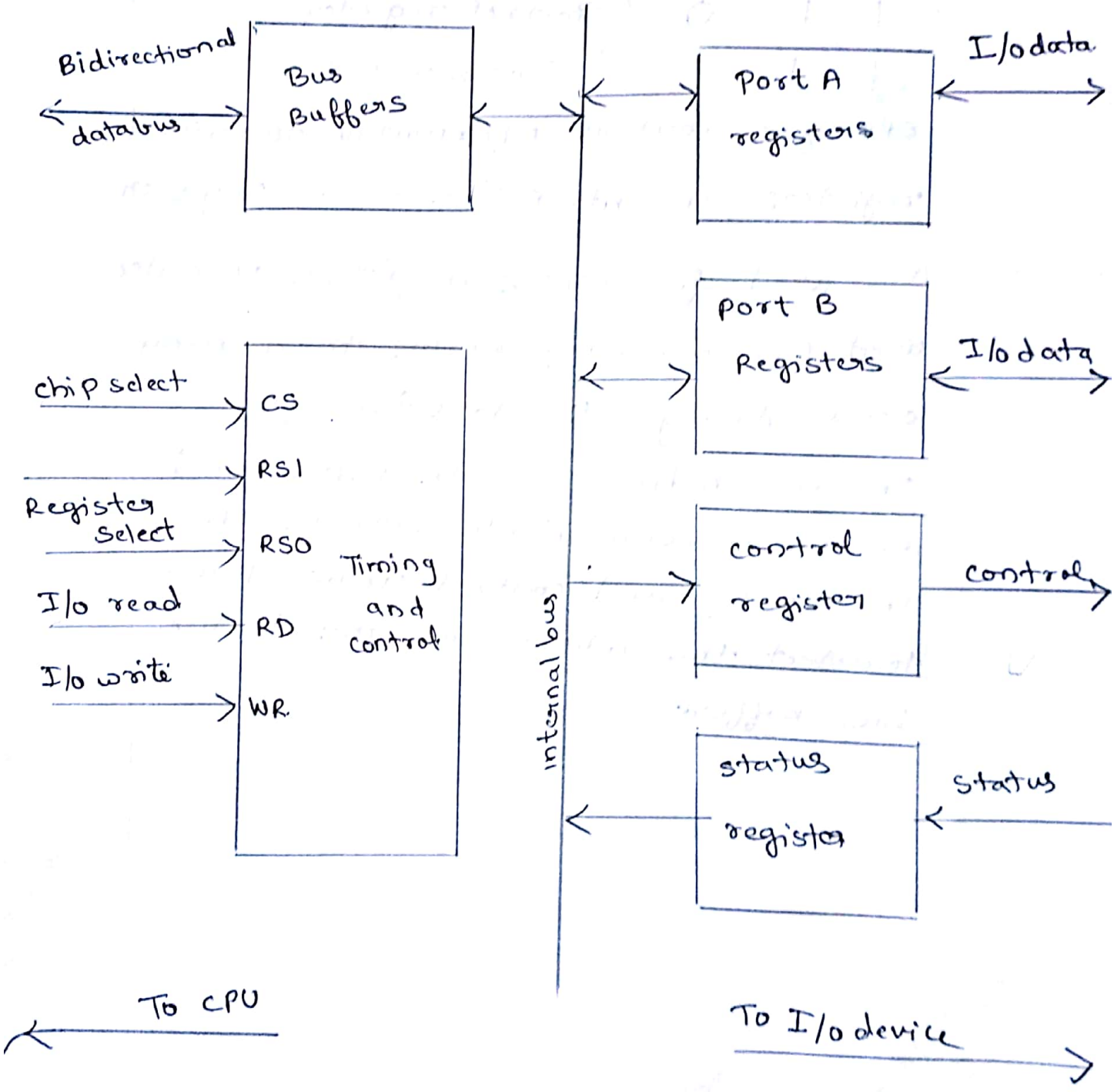
Example of I/O interface.

The four registers communicate directly with the I/O device attached to the interface. The I/O data to and from the device can be transferred into either port A and B

Port A may be defined as an input port and port B may be defined as

an output port

The output device such as magnetic disk transfer data in both direction. so bidirectional data bus is used



CS	RS1	RS0	Register selected.
0	X	X	None: data bus in high-impedance.
1	0	0	Port A register
1	0	1	Port B register
1	1	0	control register
1	1	1	Status register.

CPU gives control information to control registers. The bits in the status register are used for status conditions. It is also used for recording errors than may occur during the data transfer.

The bus buffers use the bidirectional data bus to communicate with the CPU The timing and control circuit is used to detect the address assigned to the bus buffers

ASYNCHRONOUS DATA TRANSFER

In digital system, the internal operations are synchronized by means of clock pulses supplied by a common pulse generator.

In a computer, CPU and ~~I/O~~ an I/O interface are designed independently of each other If the register in the interface share a common clock with the CPU register, the data transfer between the two units are said to be synchronous

when internal timing in each unit is independent from the other and when registers in interface and registers of CPU uses its own private clock the two units are said to be asynchronous to each other

CPU and I/O devices must coordinate for data transfer.

Methods used in Asynchronous data transfer

strobe control.
Handshaking.
are Two ways of data transfer.

STROBE CONTROL

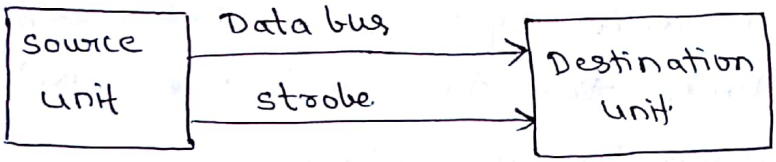
It employs a single control line to time each transfer.

Strobe: control method of data transfer. Uses a single control signal for each transfer. The strobe may be activated by either the source unit or the destination unit

→ source initiated strobe.

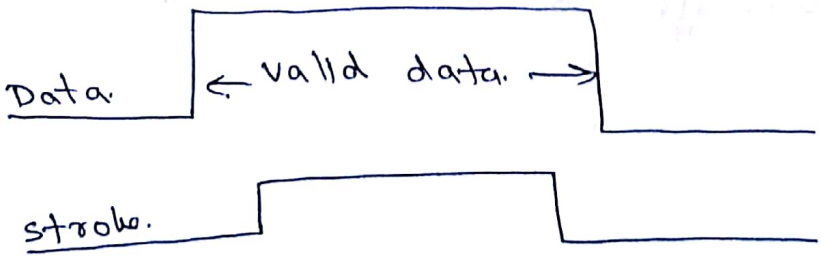
→ Destination initiated strobe

Source initiated strobe.



The data bus carries the binary information from source unit to the destination unit.

The strobe is a single line that informs the destination unit when a valid data word is available in the bus

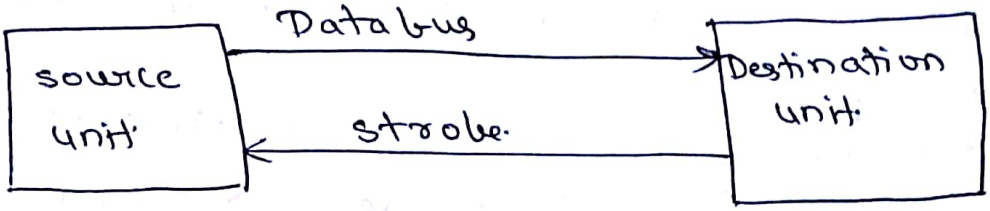


Timing diagram

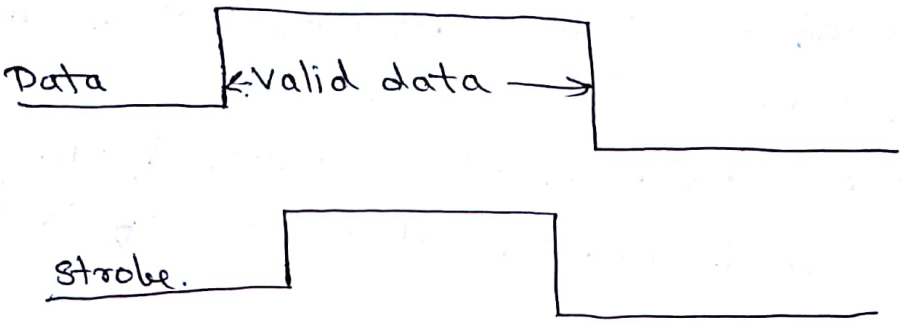
The source unit first places the data on the bus.
After a brief delay to ensure that the data settle to a steady value., the source activities the strobe pulse.
The information of the data bus and the strobe signal remain in the active state for a sufficient time period to allow the destination unit to receive the data.
The source removes the data from the bus for a brief period of time after it disables its strobe pulse.

Destination initiated strobe.

→ first the destination unit activates the strobe pulse, informing the source to provide the data.
→ The source unit responds by placing the required binary information on the unit to accept it
→ The data must be valid and remain in the bus long enough for the destination unit to accept it.
→ The falling edge of the strobe pulse can be used again to trigger a destination register
→ The destination unit then disable the strobe.
The source removes the data from the bus after a predetermined time interval



Block diagram



Timing diagram

HANDSHAKING.

→ In case of source initiated data transfer under strobe control method, the source unit has no way of knowing whether destination unit has received the data or not

→ similarly, destination initiated transfer has no method of knowing whether the source unit has placed the data on the databus

→ Handshaking mechanism solves this problem by introducing a second control signal that provides a reply to the unit that initiat the transfer.

→ There are two control lines in hanshaking tcchniques

1. source to destination unit.
2. Destination to source unit

Source initiated Transfer

Hand shaking signals are used to synchronize the bus activities

The two hand shaking lines are.

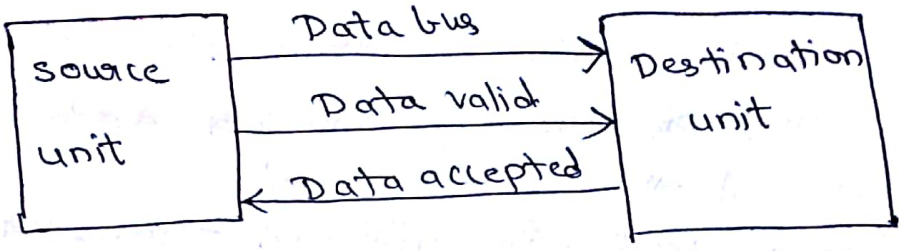
① → data valid.

② data accepted

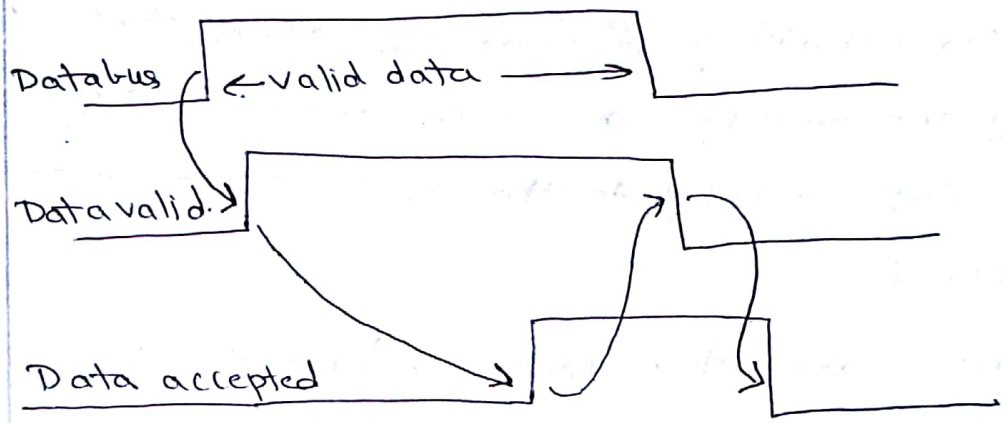
data valid → generated by the source unit

data accepted → generated by the destination unit.

Block diagram

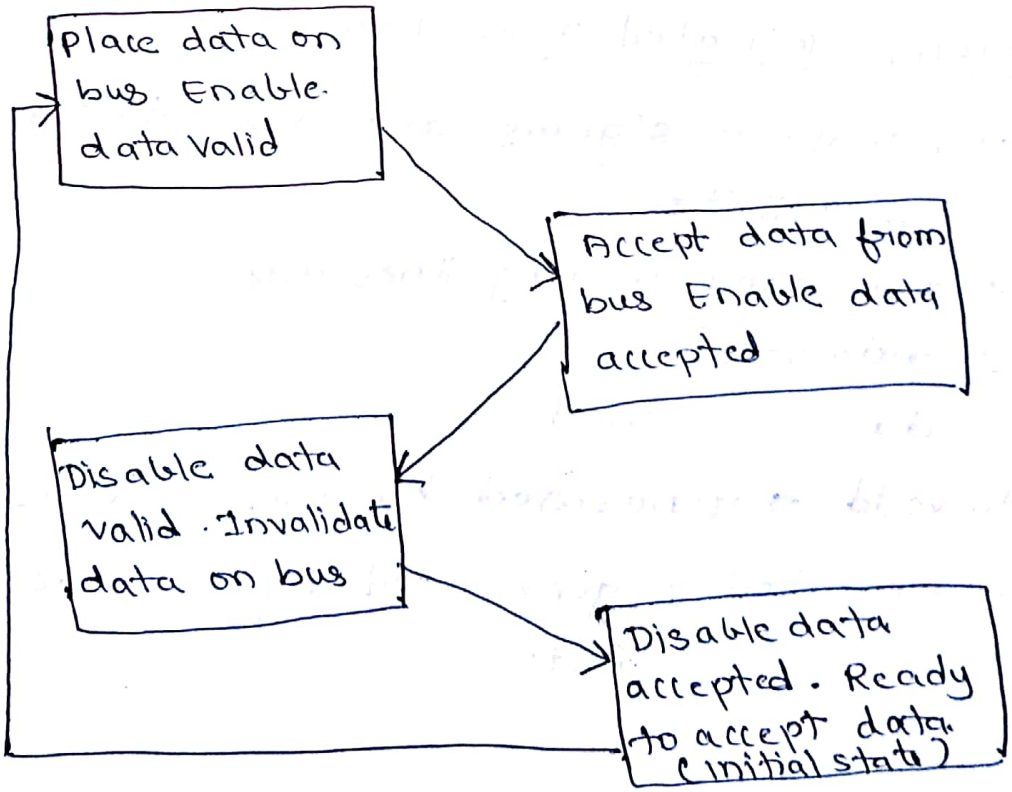


Timing diagram



Sequence of Events

Source unit — Destination unit



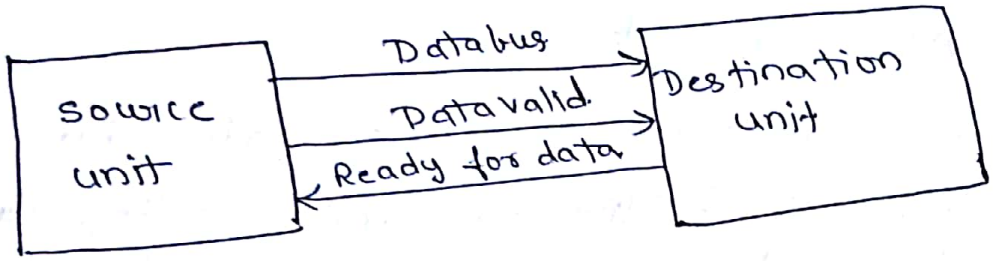
The sequence of events

→ The source unit initiates the transfer by placing the data on the bus and enabling its data valid signal.
→ The data accepted signals is activated by the destination unit after it accepts the data from the bus.
The source ~~unitth~~ unit then disables its data valid signal, which invalidates the data on the bus.
The destination unit then disables its data accepted signal and the system goes into its initial state

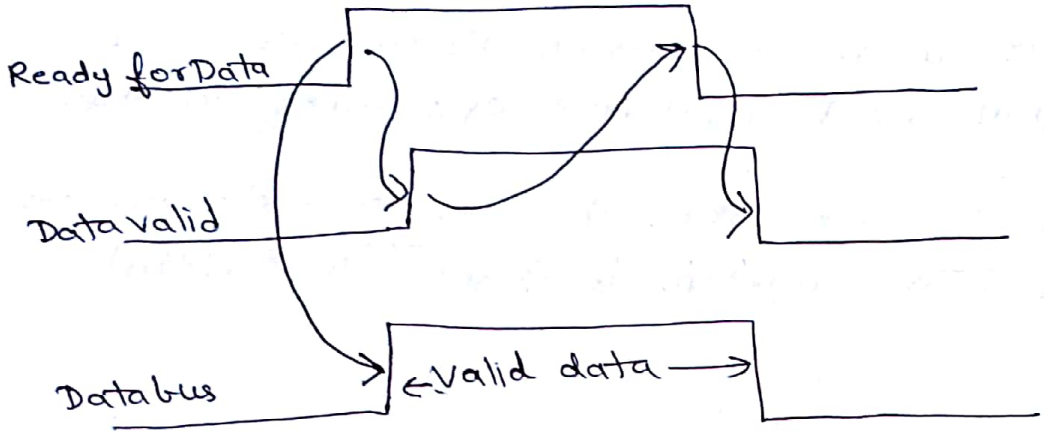
DESTINATION INITIATED TRANSFER USING HANDSHAKING

In this case the name of the signal generated by the destination unit is ready for data
The source unit does not place the data on the bus until it receives the ready for data signals from the destination unit.
The handshaking procedure follows the same pattern as in source initiated case. The sequence of events in both the cases is almost same except except the ready for signal has been converted from data accepted in case of source initiated

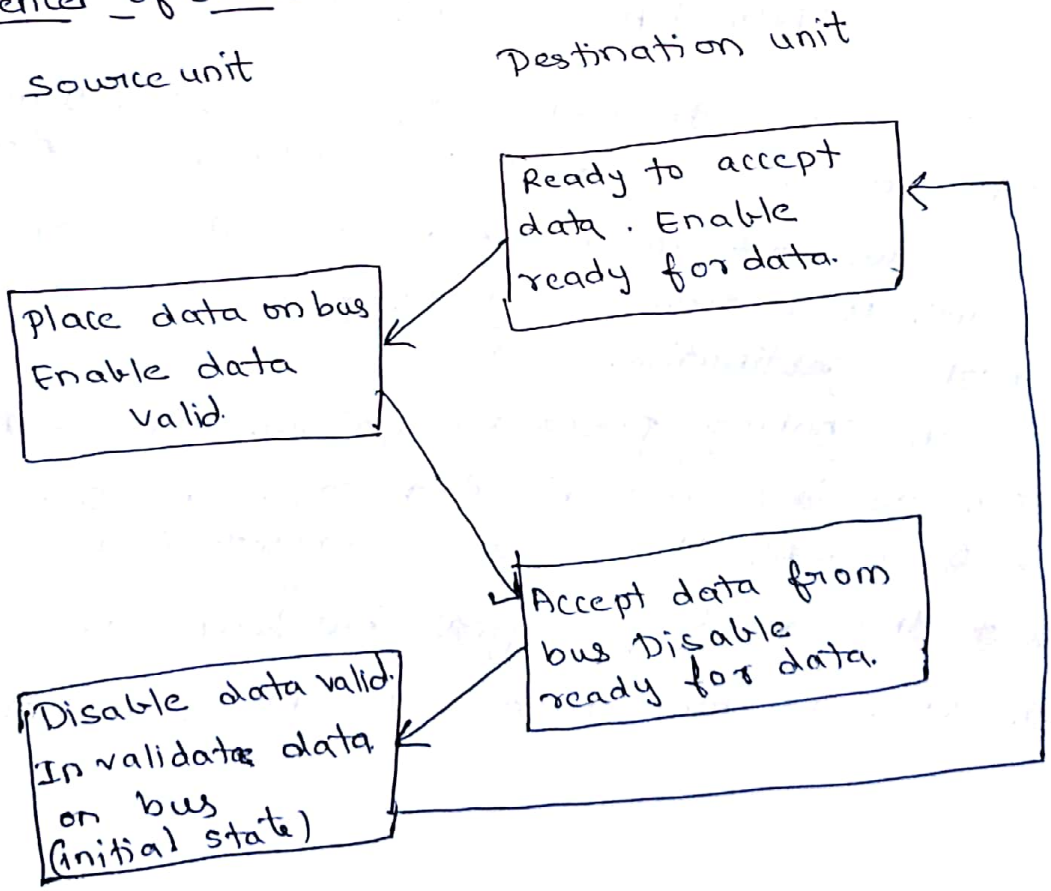
Block diagram



Timing diagram



Sequence of events



ASYNCHRONOUS SERIAL TRANSFER

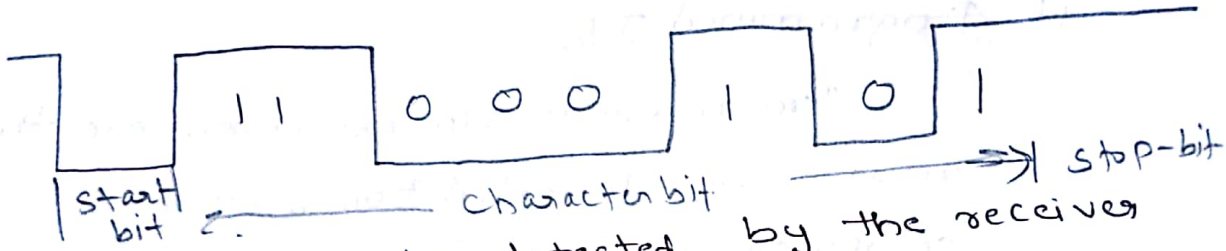
Asynchronous serial Transfer

- employs special bits which are inserted at both ends of the character code
- each character consists of three parts

1 → start bit

2 ⇒ Data bit

3 ⇒ stop bit.



A character can be detected by the receiver from the knowledge of 4 rules

→ when data are not being sent, the line is kept in the 1-state (idle state)

→ The initiation of a character transmission is detected by a start bit, which is always a 0

→ The character bits always Follow the start bit

→ After the last character, a stop bit is detected when the line returns to the 1-state for at least 1 bit time.

The receiver knows in advance the transfer rate of the bits and the number of information bits to expect.

(47) Modes of Transfer.

Data Transfer between CPU and I/O devices ~~are 3 types~~ can take place in a variety of modes

1. programmed I/O
2. Interrupt-initiated I/O
3. Direct memory access (DMA)

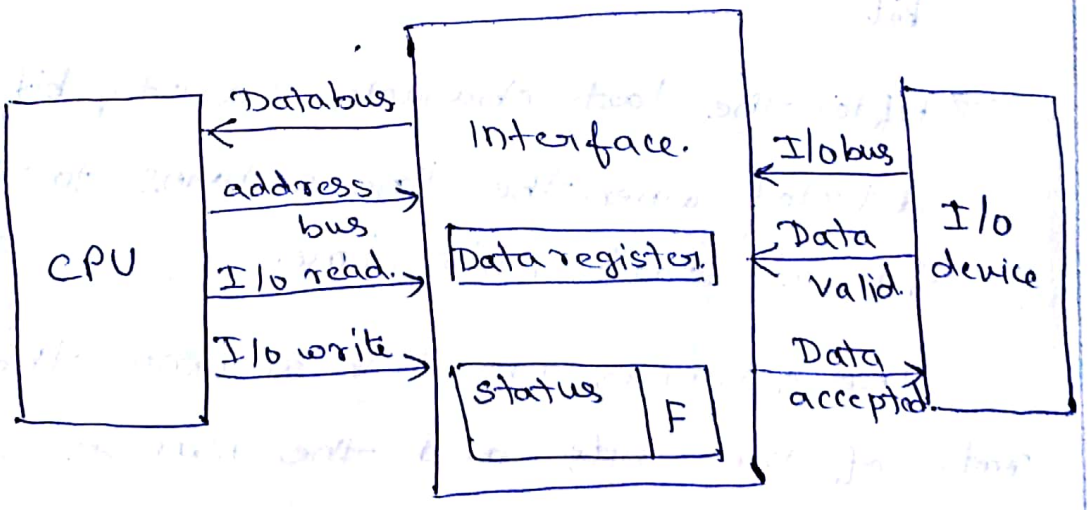
① Programmed I/O

Programmed I/O instruction are the result of I/O instruction written in computer program

Each data item transfer is initiated by the instruction in the program.

The program control data transfer to and from CPU and peripherals

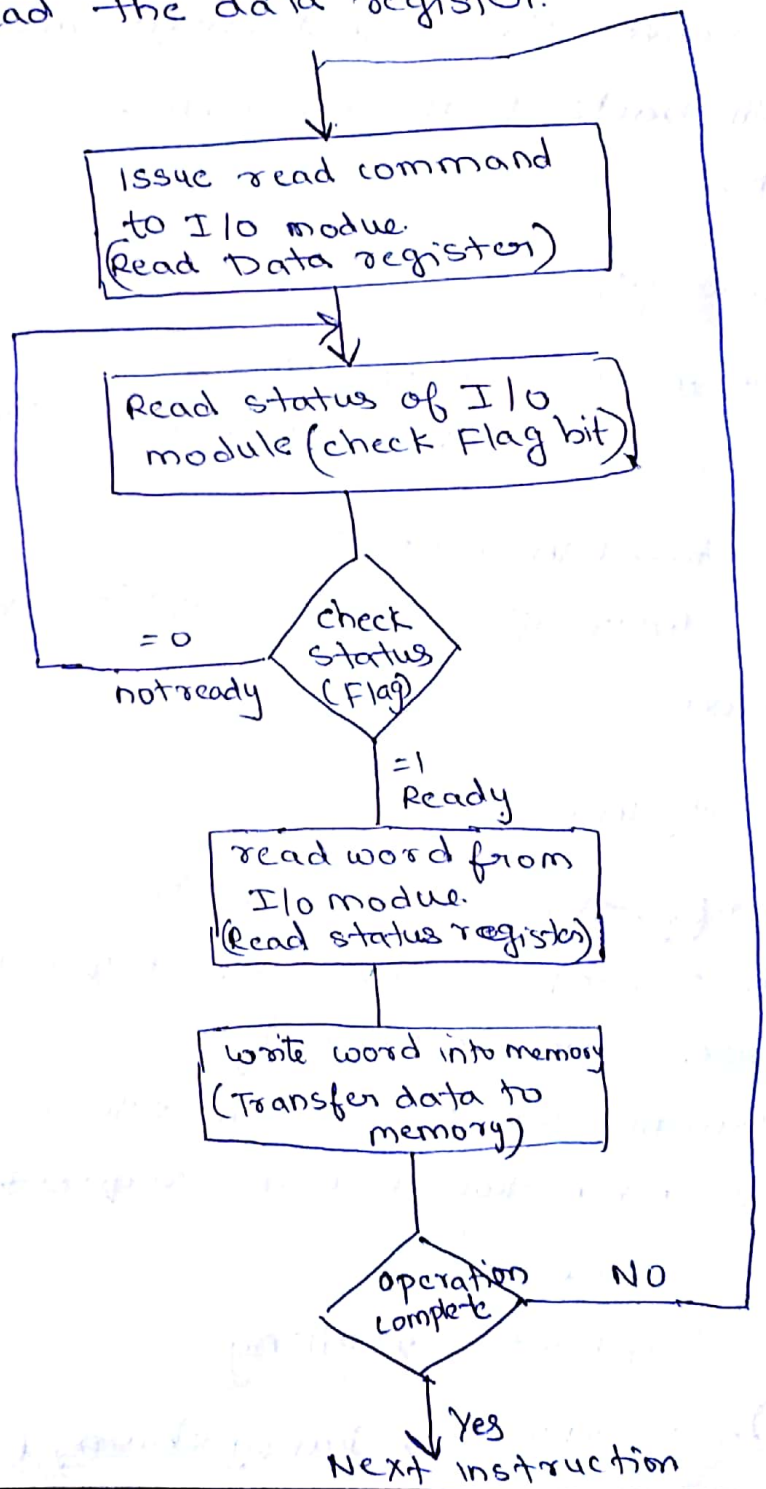
Transferring data under programmed I/O requires constant monitoring of the Peripherals by the CPU.



F = Flag bit

The transfer of each byte requires three instruction

1. Read the status registers
2. check the status of the flag bit and branch to step 1 if not set or to step 3 if set
3. Read the data register.



Interrupt-Initiated I/O

when the interface determines that the Peripheral is ready for data transfer, it generates an interrupt.
After receiving the interrupt signal the CPU stops the task which it is processing and service the I/O transfer. and then return backs to its previous processing task.

Two ways.

1). Non vectored interrupt →
fixed branch address

2). vectored interrupt →
interrupt source supplies the branch address

Priority interrupt

Identify the source of the interrupt when several sources will request service simultaneously
Determine which condition is to be served first when two or more request arrive simultaneously

1) software → polling
2). Hardware → Daisy chain, parallel priority

Polling

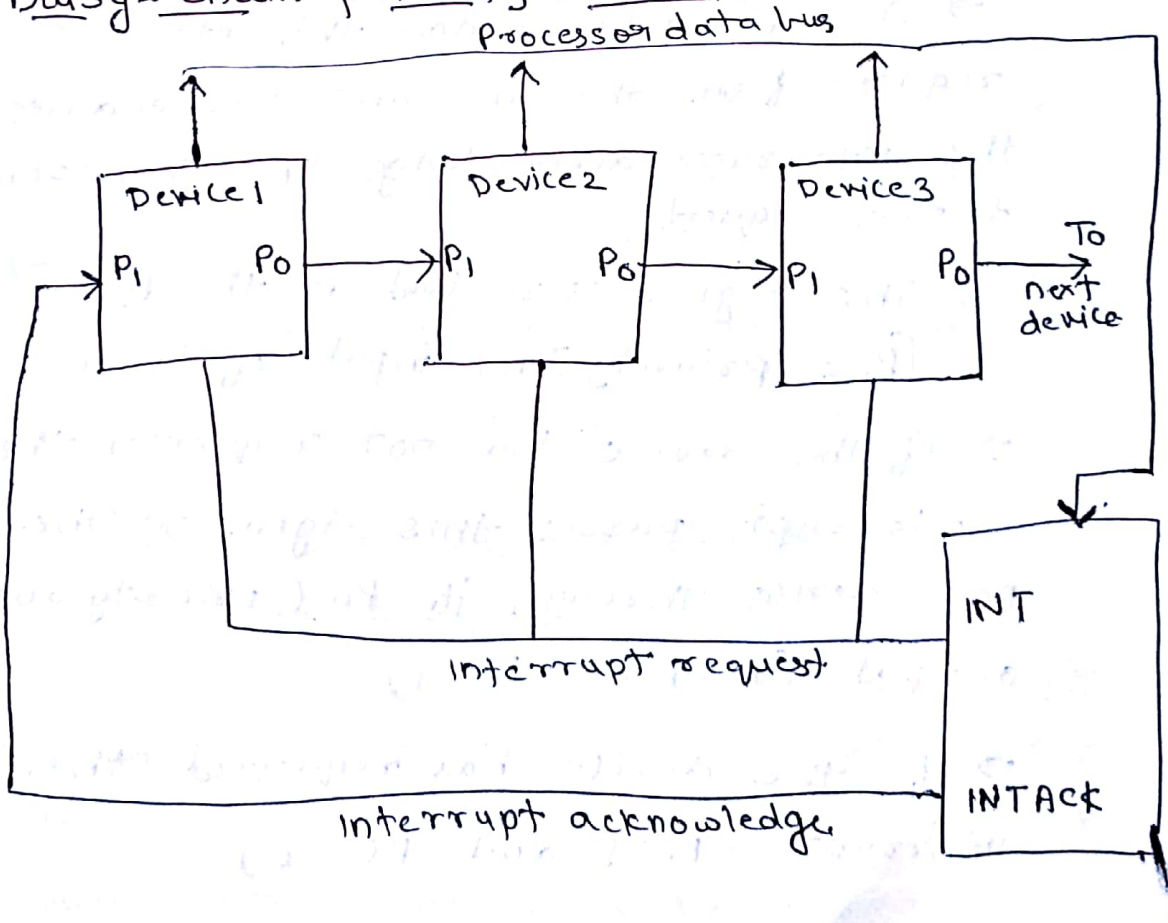
→ Identify the highest priority source by software means

- one common branch address is used for all interrupts
- program polls the interrupt sources in sequence
- The highest-priority source is tested first

Polling priority interrupt

If there are many interrupt sources, the time required to poll them can exceed the time available to service the I/o device. means it is slow

Daisy-chain priority interrupt



(5)

The daisy-chaining method involves connecting all the devices that can request an interrupt in a serial manner. This configuration is governed by the priority of the devices. The device with the highest priority is placed first followed by the second highest priority device and so on.

There is an interrupt request line which is common to all the device and goes into the CPU.

→ when no interrupts are pending, the line is in High state. But if any of the devices raises an interrupt, it places the interrupt request line in the Low state.

→ The CPU acknowledges this interrupt request from the line and then enables the interrupt acknowledge line in responce to the request.

→ This signal is received at the P_1 (Priority In) input of device 1

→ if the device has not requested the interrupt passes this signal to the next device through its Po (priority out) output ($P_1 = 1$ and $P_0 = 1$)

→ if the device has requested the interrupt ($P_1 = 1$ and $P_0 = 0$)

The device consumes the acknowledge signal and blocks its further use by placing 0 at its Po (priority out) output.

The device then proceeds to place its interrupt vector address (VAD) into the data bus of CPU.

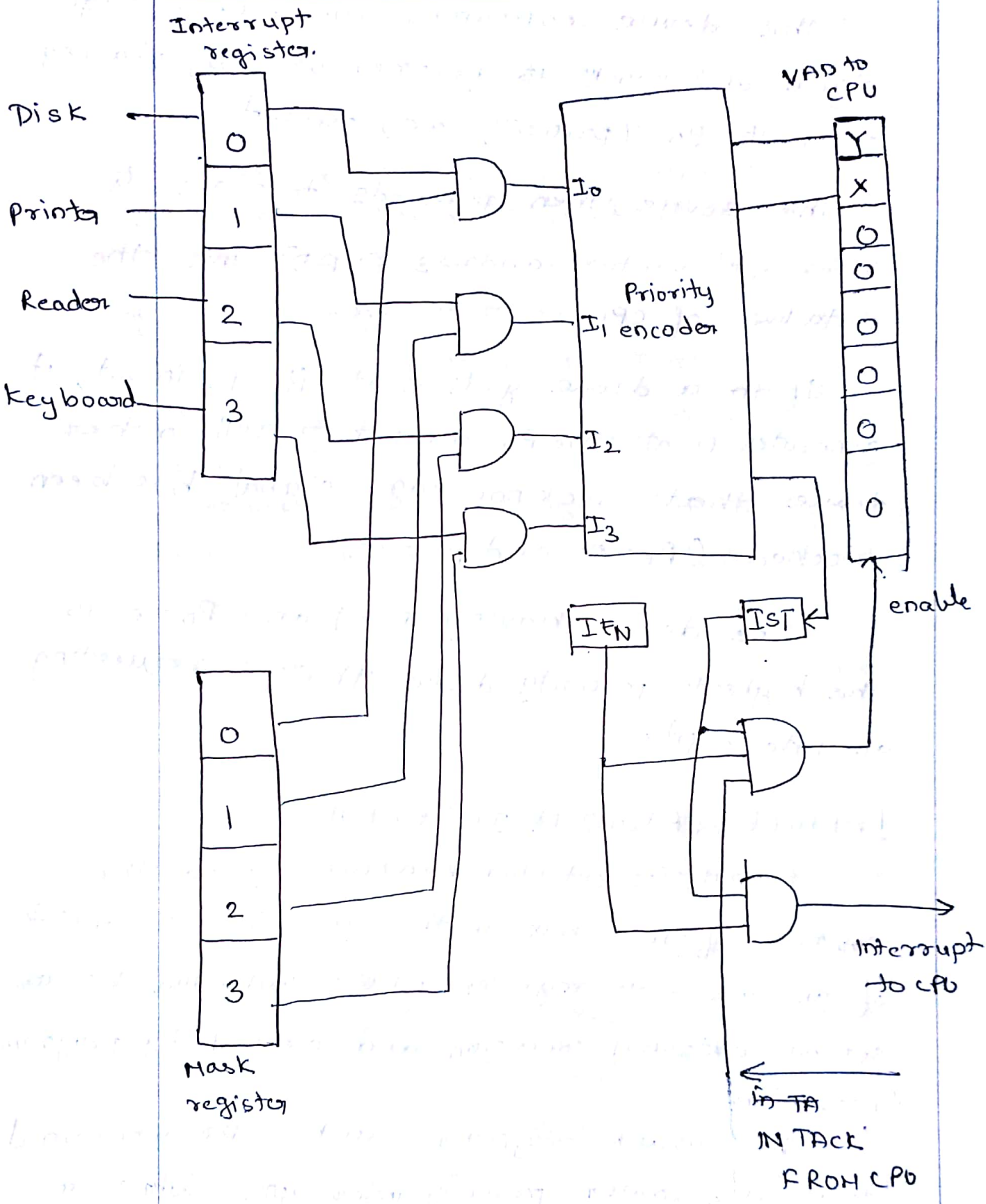
If a device gets 0 at its P_I input, it generates 0 at the Po output to tell other devices that acknowledge signal has been blocked ($P_I = 0$ and $P_O = 0$)

The device having $P_I = 1$ and $P_O = 0$ is the highest priority device that is requesting an interrupt.

PARALLEL PRIORITY INTERRUPT

Priority is established according to the position of the bits in the register. It consists of an interrupt register whose individual bit are set by external condition and cleared by program instruction.

The mask register can be programmed to disable lower-priority interrupts while a higher-priority device is being serviced. It can also provide a facility that allows a high-priority device to interrupt the CPU while a lower-priority device is being serviced.



gt consists of an interrupt register whose individual bits are set by external condition and cleared by program instruction.

→ The mask register has the same number of bits as the interrupt register

→ By means of program instruction, it is possible to set or reset any bit in the mask register.

→ Each interrupt bit and its corresponding mask bit are applied to an AND gate to produce inputs to the priority encoder.

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 the vector address, which is transferred to the CPU.

IST → interrupt status flip-flop → gt is set when an interrupt that is not masked occurs

IEN → interrupt enable flip-flop → can be set or cleared by the program to provide an overall control over the interrupt system

The outputs of IST ANDed with IEN provide a common interrupt signal for the CPU.

The interrupt acknowledge INTACK signal from the CPU enables the bus buffers in

in the output register and a vector address VAD is placed into the data bus.

Priority Encoder →

The logic of the priority encoder is such that if two or more inputs arrive at the same time the input having the highest priority will take precedence

Inputs				outputs			
I_0	I_1	I_2	I_3	x	y	IST	Boolean Function
1	x	x	x	0	0	1	
0	1	x	x	0	1	1	$x = I_0' I_1'$
0	0	1	x	1	0	1	$y = I_0' I_1 + I_0' I_2'$
0	0	0	1	1	1	1	$(IST) = I_0 + I_1 + I_2 + I_3$
0	0	0	0	x	x	0	

The output of the priority encoder is used to form part of the vector address for each interrupt source.

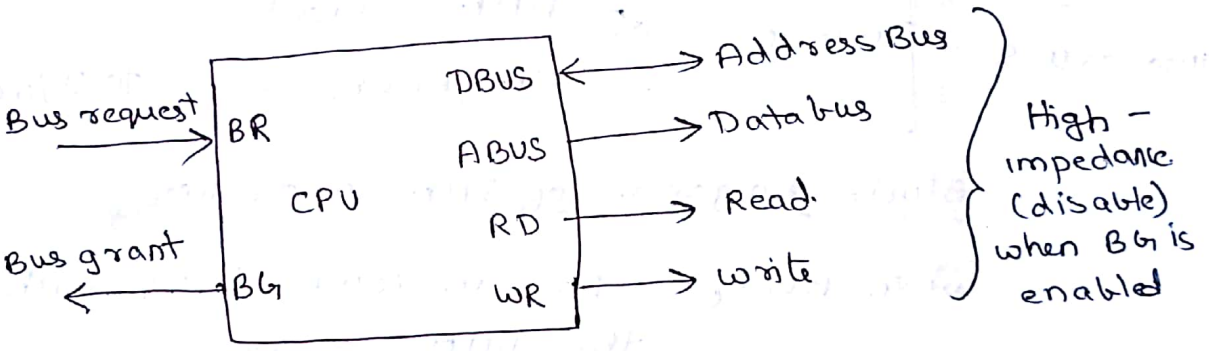
DIRECT MEMORY ACCESS (DMA)

To transfer large blocks of data at high speed between external device and main memory, DMA approach is often used.

DMA controller allows data transfer directly between I/O device and memory, with minimal invention of processor.

DMA controller acts as a processor, but it is controlled by CPU.

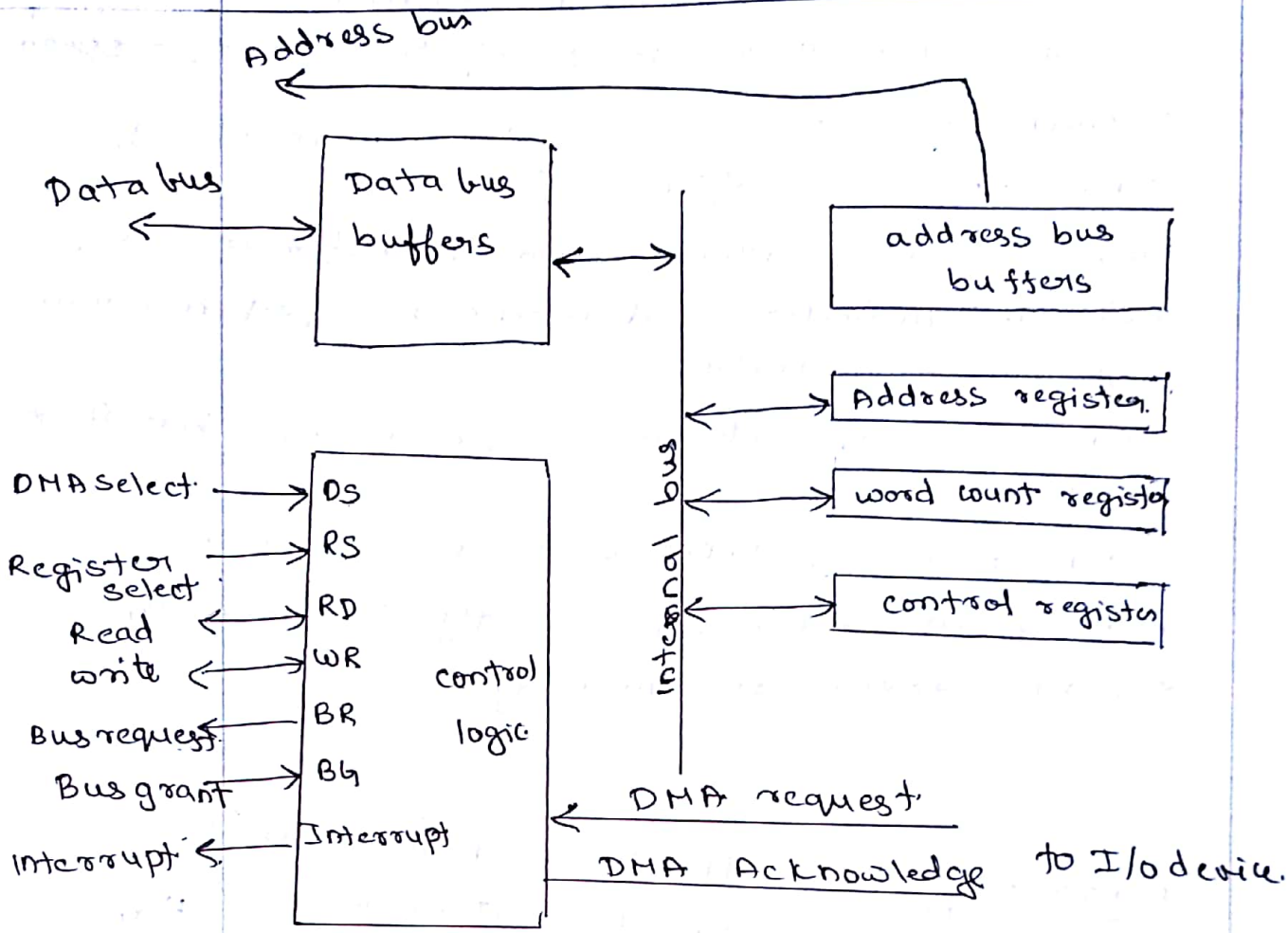
DMA controller takes over the buses to manage the transfer directly between the the I/O device and memory



Two control signals in the CPU that facilitate the DMA transfer.

The bus request (BR) input is used by the DMA controller to request CPU for the control of bus.

The CPU activates the bus grant (BG) output to inform the external DMA that the buses are in the high-impedance state.



Block diagram of DMA controller

when BG = 0 CPU can communicate with the DMA register

BG = 1 the CPU gives the control of Bus to DMA

The DMA communicates with the external peripherals through the request and acknowledge lines by using handshaking procedure.

The DMA first initialize the CPU. After that the DMA starts and continues to transfer data between memory and peripheral unit until an entire block

is transferred. The CPU initialize the DMA by sending the following information through the data bus.

1. The starting address of the memory block where data are available (for read) or where data are to be stored (for write)
2. The word count, which is the number of words in the memory block.
3. control to specify the mode of transfer. such as read or write
4. A control to start the DMA transfer.

The starting address is stored in the address register. The word count is stored in the word count register, and the computer information in the control register.

DMA Transfer

The CPU communicates with the DMA through the address and data buses as with any interface unit.

The DMA has its own address, which activates the DS and 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 device and the memory.

→ The DMA request line is used to request a DMA transfer

→ The bus request (BR) signal is used by the DMA controller to request the CPU to relinquish control of the buses

→ The CPU activates the bus grant (BG) output to inform the external DMA that its buses are in a high-impedance state (so that they can be used in the DMA transfer)

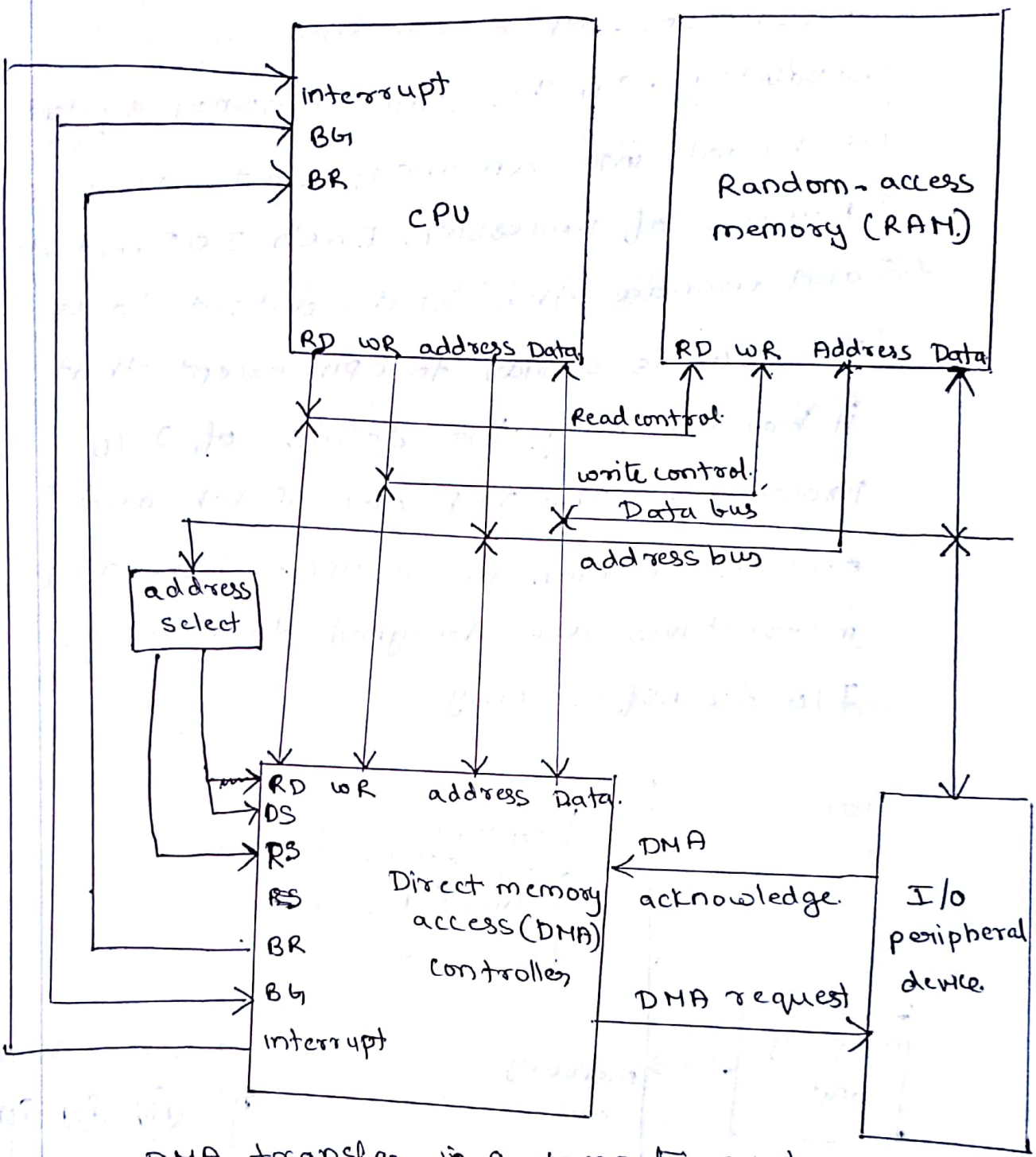
→ The address bus is used to address the DMA controller and memory at given location

→ The Device select (DS) and register select (RS) lines are activated by addressing the DMA controller.

→ The RD and WR lines are used to specify either a read (RD) or write (WR) operation on the given memory location

→ The DMA acknowledge line is set when the system is ready to initiate data transfer

→ The data bus is used to transfer data between the I/O device and memory

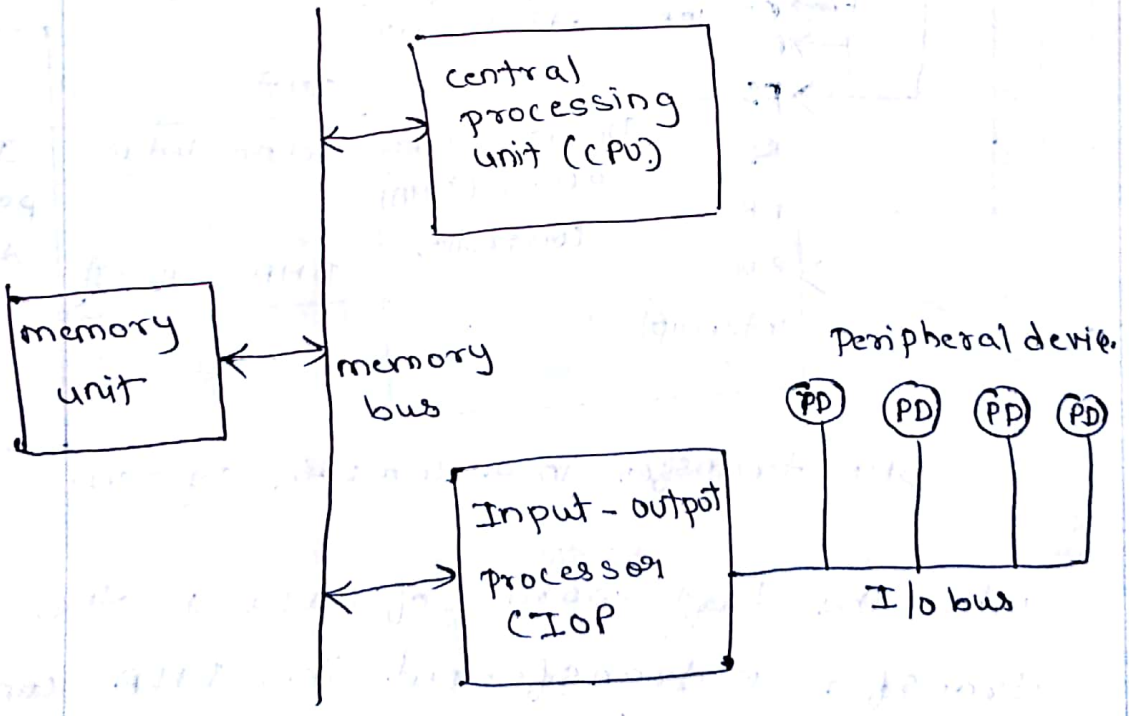


DMA transfer in a computer system

When the last word of data in the DMA transfer is transferred, the DMA controller informs the termination of the transfer to the CPU by means of the interrup line

Input - output processor (IOP)

An input - output processor (IOP) is a processor with direct memory access capability. In this, the computer system is divided into a memory unit and number of processor. Each IOP controls and manage the input - output tasks. The IOP is similar to CPU except that it handles only the details of I/o processing. The IOP can fetch and execute its own instruction. These IOP instructions are designed to manage I/o transfers only



The memory unit occupies the central position and can communicate with each processor. The CPU processes the data required for solving the computational tasks. The IOP provides a path for transfer of data between peripherals and memory. The CPU assigns the task of initiating the I/O program. The IOP operates independent from CPU and transfer data between peripherals and memory

The communication between the IOP and the devices is similar to the program control method transfer. And the communication with the memory similar to the direct memory access method

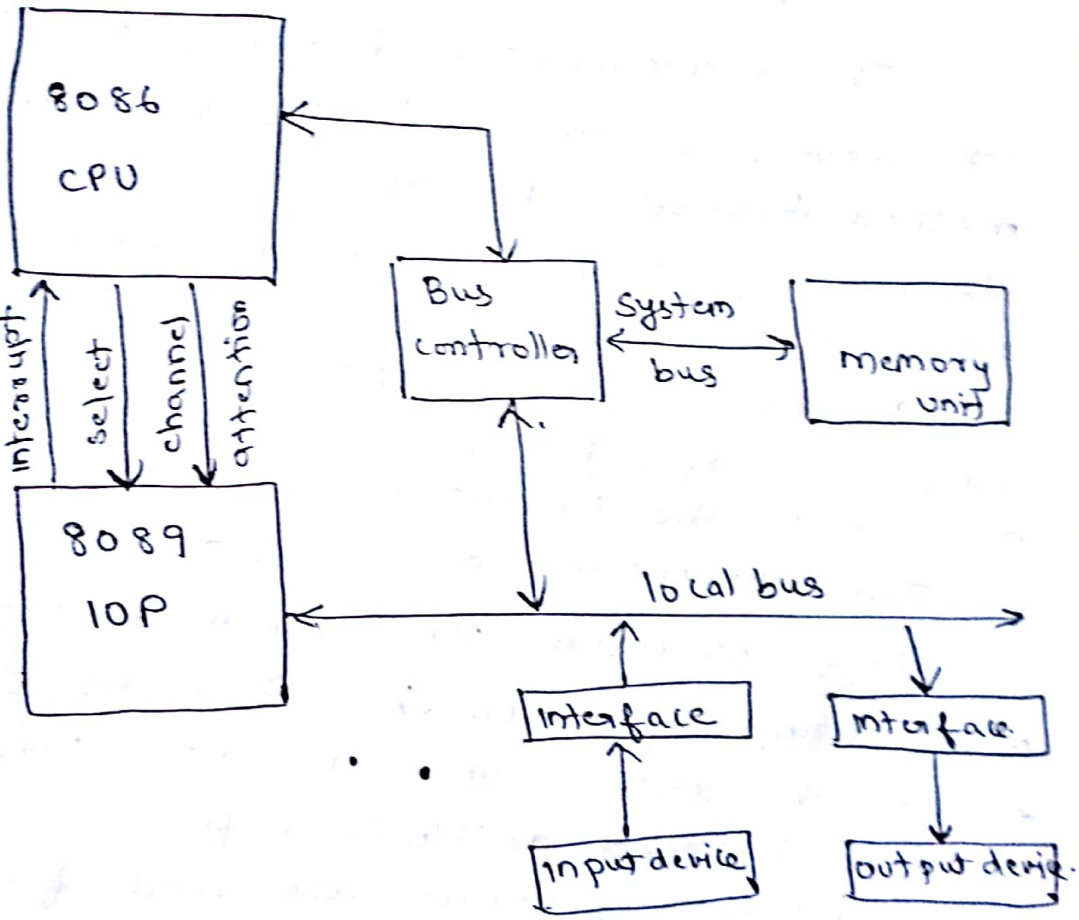
In large scale computer, each processor is independent of other processor and any processor can initiate the operation

The CPU can act as master and the IOP act as slave processor. The CPU assigns the task of initiating operation. but it is the IOP, who executes the instructions. CPU instruction provide operation to start an I/O transfer

instructions that are read from memory by an IOP are also called commands to distinguish them from instructions that are read by CPU. commands are prepared by programmers and are stored in memory. command words make the program for IOP. CPU informs the IOP where to find the command in memory

Intel 8089 IOP

The intel 8089 I/o processor is contained in a 40-pin integrated circuit package. The 8089 is designed to function as an IOP in a microcomputer system where the intel 8086 microprocessor is used as the CPU The 8089 IOP reads the message from memory, carries out the operation, and notifies the CPU when it has finished



Intel 8086/8089 microcomputer system block diagram

The 8089 IOP has 50 basic instruction than can operate on individual bits, on bytes or 16-bit words. The 8086 Function as the CPU and the 8089 as the IOP. The two units share a common memory through a bus controller connected to a system, which is called a "Multibus" by Intel. The IOP uses a local bus to communicate with various interface units connected to I/O devices. The CPU communicates with the IOP by enabling the channel attention line. The select line is used by the CPU to select one of two channels in the 8089. The IOP gets the attention of the CPU by sending an interrupt request.

The CPU and IOP communicate with each other by writing messages for one another in system memory. The CPU prepares the message area and signals the IOP by enabling the channel attention line. The IOP reads the message, performs the required I/O Function and executes the appropriate channel program. When the channel has completed its program it issues an interrupt request to the CPU