PAGE NO.

Unit 5

Computer Arithmetic :- Introduction, Addition and subtraction, Multiplication Algorithms (Booth Multiplication Algorithm), Division Algorithm Floating point Arithmetic operations, Decimal Arithmetic Unit, Input-Output organization, Input-Output Interface, Asynchronous data transfer, Modes of transfers, Priority Interrupt, DMA, Input-Output processor (IOP), CPU IOP communication, serial communication.

Introduction:- The solution to any problem that is stated by a finite number of well-defined procedural steps is called algorithm. We contide addition, substraction, multiplication and division for the following types of data.

1. Fixed point-binary data in signed-magnitude representation
2. fixed point binary data in signed 2^{11} complement representation.
3. Floating-point binary data.
4. Binary coded decimal - (BCD) data.

Addition and substraction:-

1. **Addition and substraction with signed magnitude data:-**

In both operations the first bit is considered as Addition/substracti

1100 ↑ Negative Number

0100 ↑ Positive Number

Addition and substraction of signed-Magnitude Numbers

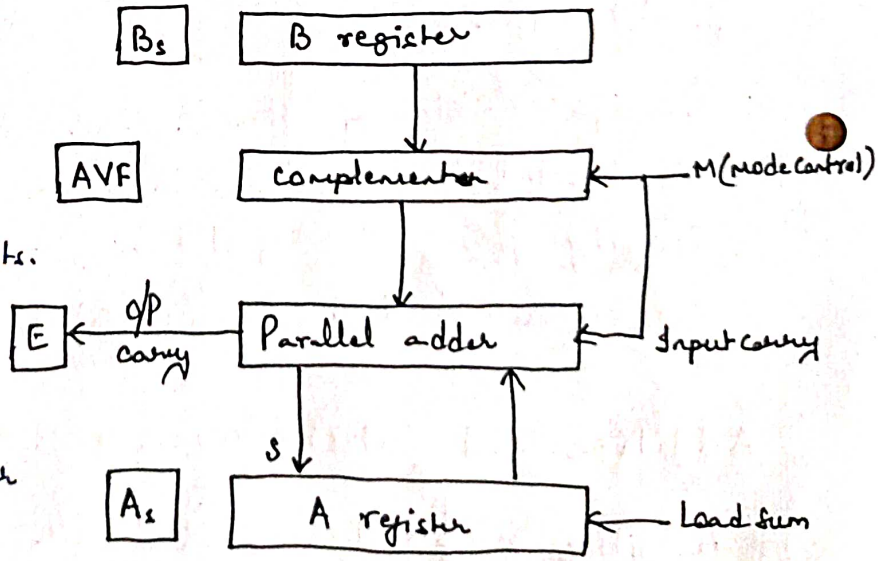
Operation	Add Magnitudes	Substract Magnitudes		
		When A>B	When A<B	When A=B
(+A)+(+B)	+(A+B)			
(+A)+(-B)		+(A-B)	-(B-A)	+(A-B)
(-A)+(+B)		-(A-B)	+(B-A)	+(A-B)
(-A)+(-B)	-(A+B)			
(+A)-(+B)		+(A-B)	-(B-A)	+(A-B)
(+A)-(-B)	+(A+B)			
(-A)-(+B)	-(A+B)			
(-A)-(-B)		-(A-B)	+(B-A)	+(A-B)

ADDition:-

1. if the signs of both the numbers are same, then we have to add the magnitudes and the resultant sign of A is consider overall.

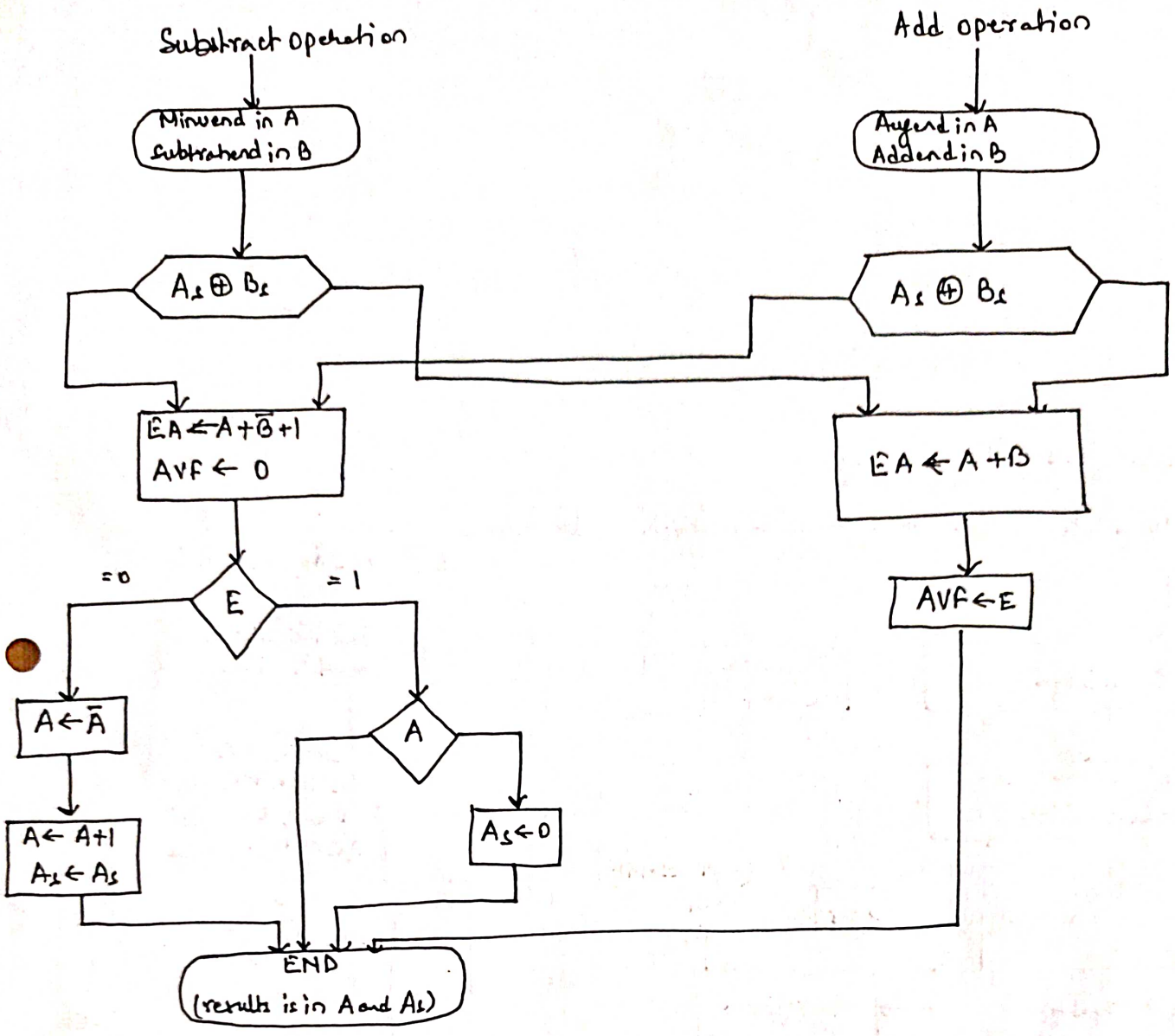
2. if the signs of both the numbers are different then we substract the magnitudes

- Here A_s is flipflop to store signed bit of first number.
- B_s stored signed bit of B-register using flipflop.
- Parallel adder → it adds parallel two bits.
- mode bit → if m=0 then parallel adder performs $B+0+A = A+B$ (mode bit from B-register, A-register)



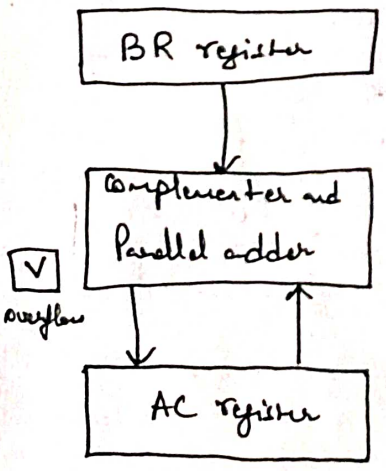
- Sum S transferred to A-register and o/p carry transferred to E-flipflop Hardware for signed magnitude addition and substraction or Extended flipflop.
- AVF → Add overflow flipflop (After Performing any addition operation, if there is any overflow then it is stored in AVF.

for substraction operation :→ $1+\overline{B}+A = A-B$

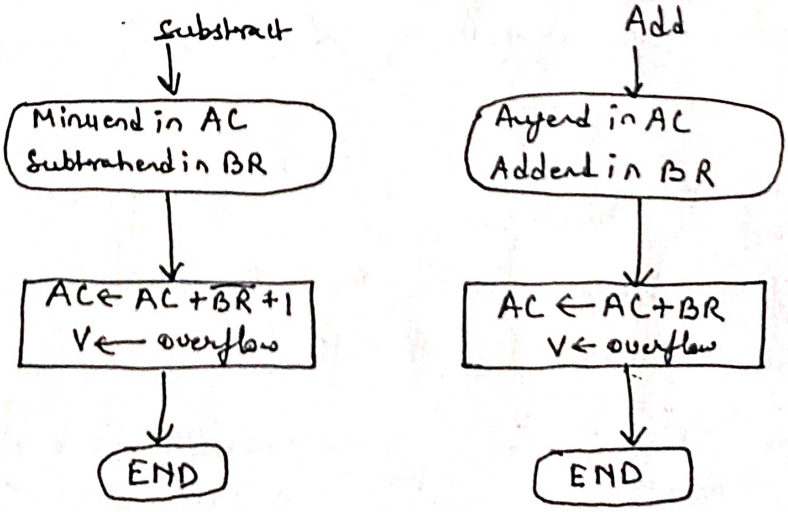


Flowchart for add and subtract operations

Addition and subtraction with signed 2's complement Data



Hardware for signed 2's complement addition and subtraction

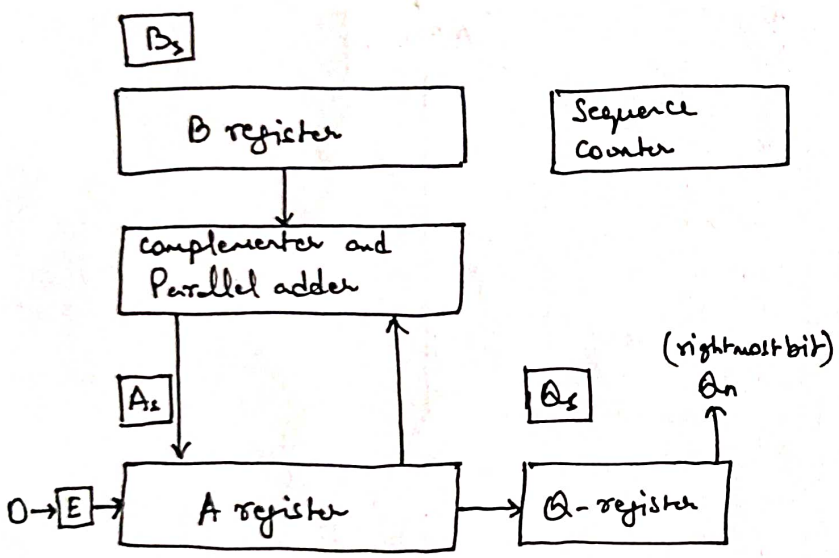


Algorithm for adding and subtracting numbers in signed-2's complement representation

PAGE NO

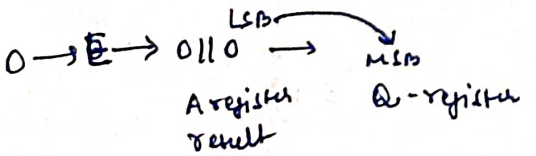
Multiplication Algorithms:-

1. Multiplication Algorithm Implementation for signed magnitude data:-



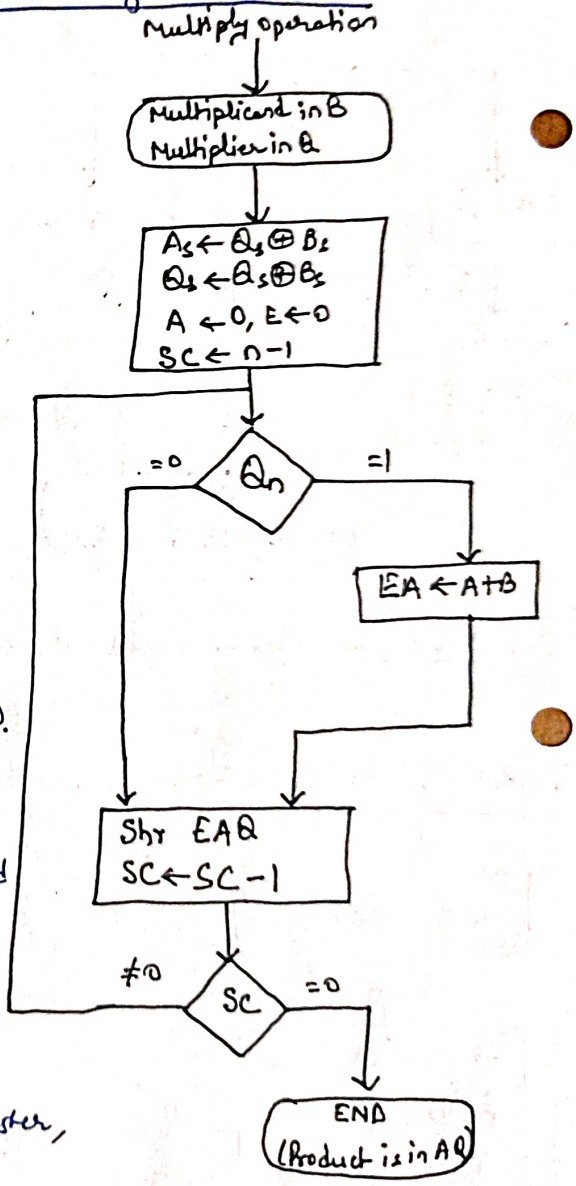
Hardware for multiply operation

- if signed bit is -ve then B_s register contains 1 otherwise 0.
- multiplier in Q register and multiplicand in B-register.
- B_s is contain signed bit of B-register and Q_s is contain signed bit of Q-register.
- Initially A is zero, it is also called source and destination register.
- After performing operation the result is stored in A register, the shift operation performs like.



and so on. so all bits of A Register shifted to Q register but one by one

- first 0 shifted to E and E bit shifted to A register and LSB bit of A shifted to Q register. and so on.



Flow chart for multiply operation

PAGE NO.

- Signed if $+ + = +1$, $- - = +1$ } + ; $+ = 0$, $- = 1$
  $+ - = - 0$, $- + = - 0$ } −

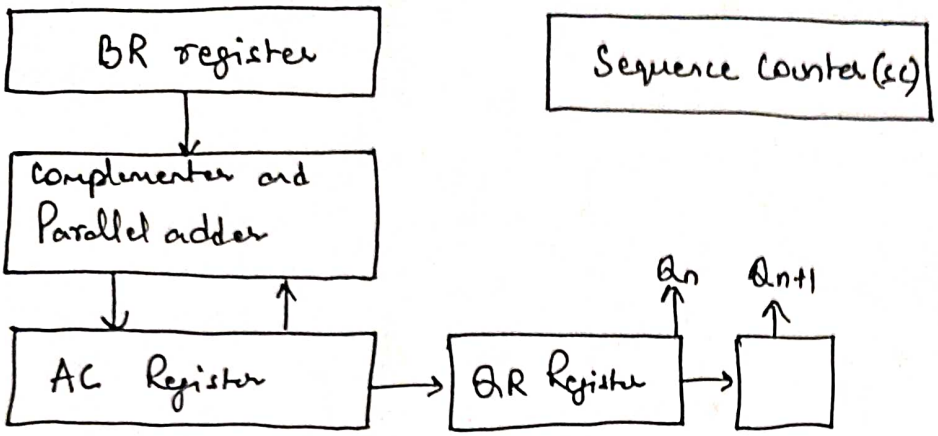
Multiplication Algorithm with signed Magnitude Data:-

Booth algorithm gives a Procedure

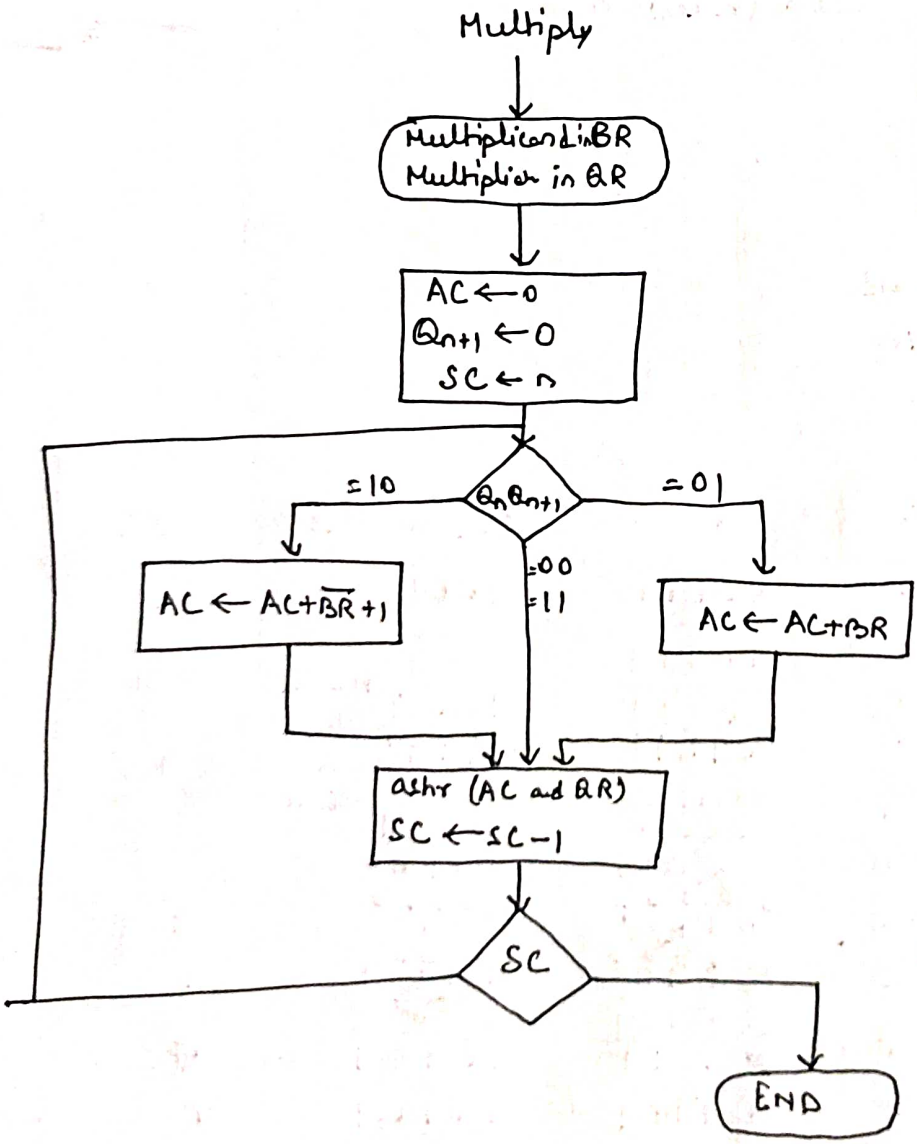
$23 = (10111)$ Multiplicand
$\times 19 = (10011)$ Multiplier
$437 = (110110101)$

Multiplicand B = 10111	E	A	Q (multiplier)	SC
	0	00000	10011	5
$Q_n=1$, add B		10111		
	0	10111	10011 (1) → Discard	
Shr EAQ	0	01011	11001	4
$Q_n=1$, add B	1	10111		
		00010	11001 (1) → Discard	
Shr EAQ	0	10001	01100	3
$Q_n=0$, Shr EAQ	0	01000	10110	2
$Q_n=0$, Shr EAQ	0	00100	01011	1
$Q_n=1$, Add B		10111		
	0	11011	10101	
Shr EAQ	0	01101	10101	0

Booth multiplication Algorithm using signed 2's complement.



Hardware



Booth Algorithm for multiplication of signed 2's complement.

Booths Multiplication Algorithm:-

multiplication with 2's complement data

Ex:- $-9 \cdot 10111$, $-13 \quad 10011$, $117 \quad (0001110101)$

Register size → 5 bits

Q_n	Q_{n+1}	BR = 10111, BR+1 = 01001	AC	QR	Q_{n+1}	SC
		Initial	00000	10011	0	5
1	0	Substract BR	01001			
			01001	10011	0	
		ashr (AC+BR)	00100	11001	1	4
1	1	ashr	00010	01100	1	3
0	1		10111			
			11001	01100	1	
		ashr	11100	10110	0	2
0	0	ashr	11110	01011	0	1
1	0	Sub BR ($\overline{BR}$+1)	01001			
			00111	01011		
		ashr	00011	10101	1	0

$9 \rightarrow 01001$

2's $-9 \rightarrow 10110$

$+ \quad 1$

2's comp. 10111

$13 \rightarrow 01101$

$-13 \rightarrow 10010$

$+ \quad 1$

2's comp. 10011

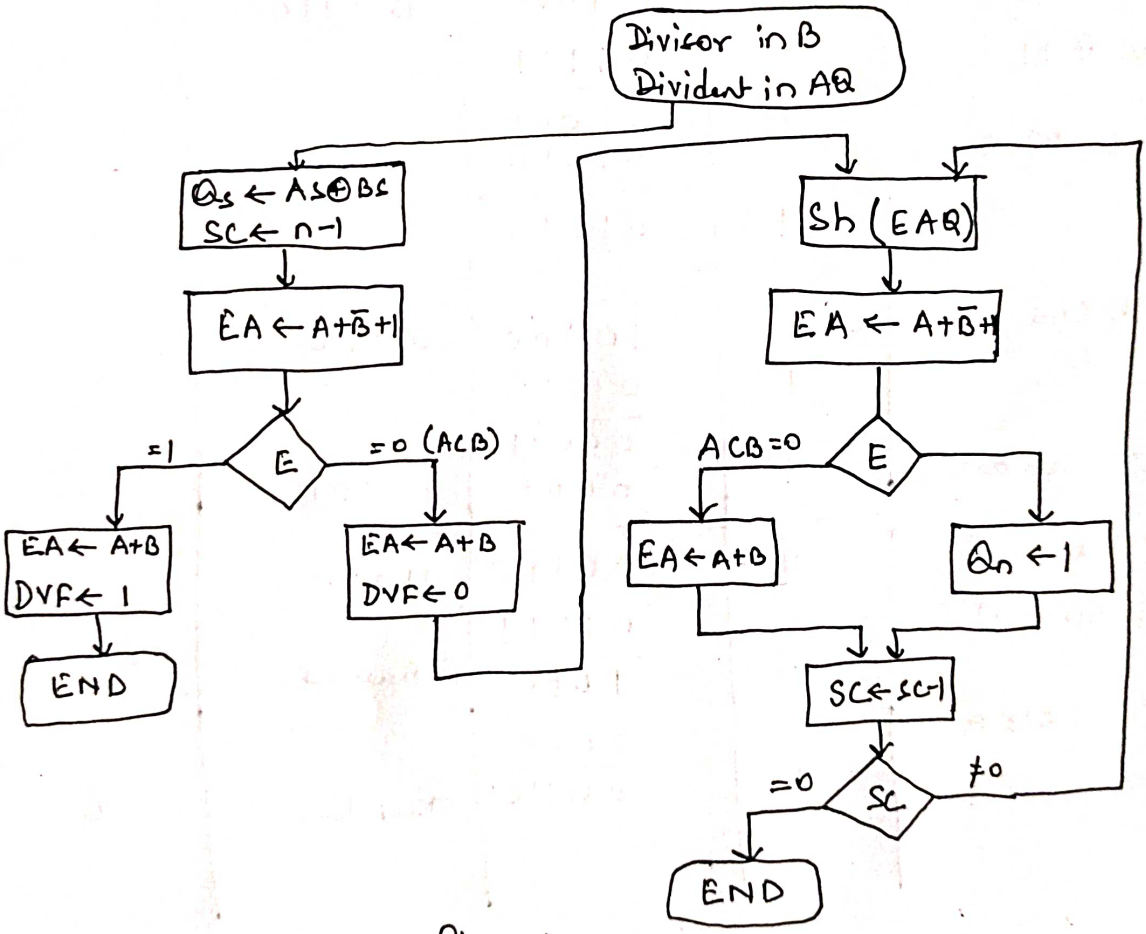
00 } Arithmetic shift right operation
11 } AC+QR

01 } ADD AC+BR

10 } SUB AC+$\overline{BR}$+1

PAGE NO

Division Algorithm for signed Magnitude Data:-



flow chart

Divisor $B = 10001$, $\overline{B}+1 = 01111$

	E	A	Q	SC
		01110	00000	5
Shl EAQ	0	11100	00000	
ADD $\overline{B}+1$		01111		
E=1, Sub $Q_n=1$	1	01011	00000	
	1	01011	00001	4
Shl EAQ	0	10110	00010	
ADD $\overline{B}+1$		01111		
E=1, Sub $Q_n=1$	1	00101	00011	3
Shl EAQ	0	01010	00110	
ADD $\overline{B}+1$		01111		
E=0, ADD B	0	11001		
		01010		
	1	01010		2
Shl EAQ		10100	01100	
	1	01111		
		00011		
E=1 Sub $Q_n=1$	1	00011	01101	1
Shl EAQ	0	00110	11010	
ADD B+1		01111	↑	
	0	10101	Quotient.	
E=0 ADD B		10001		
	1	00110 ← Remainder		0

POORNIMA
COLLEGE OF ENGINEERING
DETAILED LECTURE NOTES

Floating Point Arithmetic Operations:- (Addition and Subtraction)

If AC ≠ 0, BR ≠ 0 then perform steps

1. Check for zeros
2. Alignment of mantissas
3. Add or Subtract mantissas
4. Normalized the result.

exponent

mantissa × base

0.23456

msb

Normalized form

* If msb is not zero then it is normalized form. 0.23456

Non-zero msb

* if msb is zero then it is not normalized form 0.0123

Zero msb (not Normalized form)

$0.23456 \times 10^{-5} \Rightarrow$

For eg:-

$AC = 0.538123 \times 10^{3}$

$BR = 0.123000 \times 10^{-1}$

- for add or sub exponents must be equal either both 10^{3} or both 10^{-1}.

- In first approach we make 10^{3} to 10^{-1}.

$AC = 0.230000 \times 10^{-1}$

$BR = 0.123000 \times 10^{-1}$

0.353000×10^{-1}

- In second approach we make 10^{-1} to 10^{3}

$AC = 0.538123 \times 10^{3}$

$BR = 0.000012 \times 10^{3}$

AC = 0

BR = 0

AC = 0

AC = 0

BR = 10

— 10

For Addition

AC = 0.000

BR = 0123

AC = 0.123

AC ← BR

for eg:- 0.532×10^3
0.712×10^3
} if both the sign and powers are same then we perform operation. Algorithm

1.244×10^3 ← overflow

- After adding both registers if overflow comes shift right operations perform.

0.124×10^4 (After Performing shift right operation answer changed).

- Register cannot flow overflow.

Substraction Operation:-

for eg:-

$AC = 0.532 \times 10^3$
$BR = 0.521 \times 10^3$

0.011×10^3 ← Underflow

we perform shift left operation 0.110×10^3

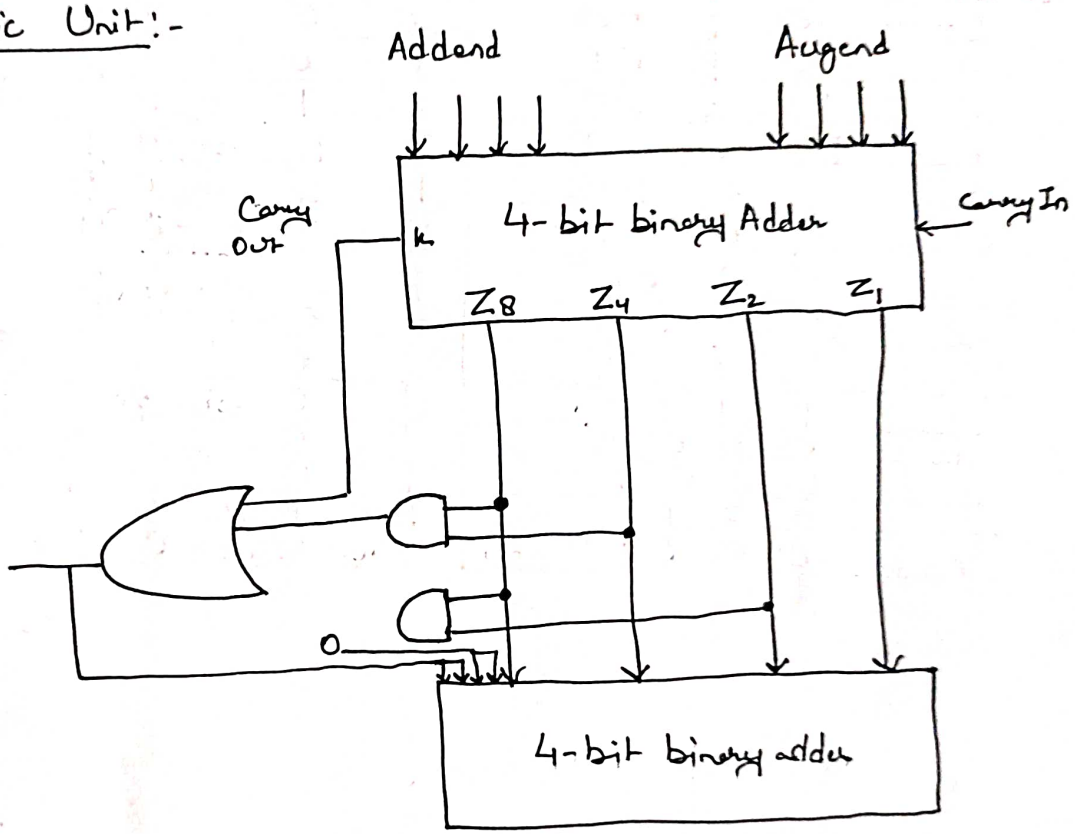
- After Performing the substraction operation, if we get the most significant bit is zero 0.011×10^3 (msb) then we can say that it is comes in underflow condition.

So for that underflow we perform shift left operation. 0.110×10^2

Last

- Normalized the result. (Addition overflow, Substraction underflow) to get Normalized result.

Decimal Arithmetic Unit:-



Block diagram of BCD Adder

A decimal Arithmetic unit is a digital function that performs decimal micro operations. it mainly perform two operations. Addition and subtraction.

The addition operation is performed with BCD Adder whereas subtraction is Performed with BCD subtractor. Here, BCD stands for binary-coded decimal.

BCD code . In this code each decimal digit (0-9) is represented by a 4-digit binary number. for eg. 374 is a decimal number whereas 3, 7 and 4 are the decimal digits. Hence forth, by using BCD we can represent decimal digits by a 4-bit binary number.

- Positional weights are 8-4-2-1. And sometimes BCD is known as 8-4-2-1 code.

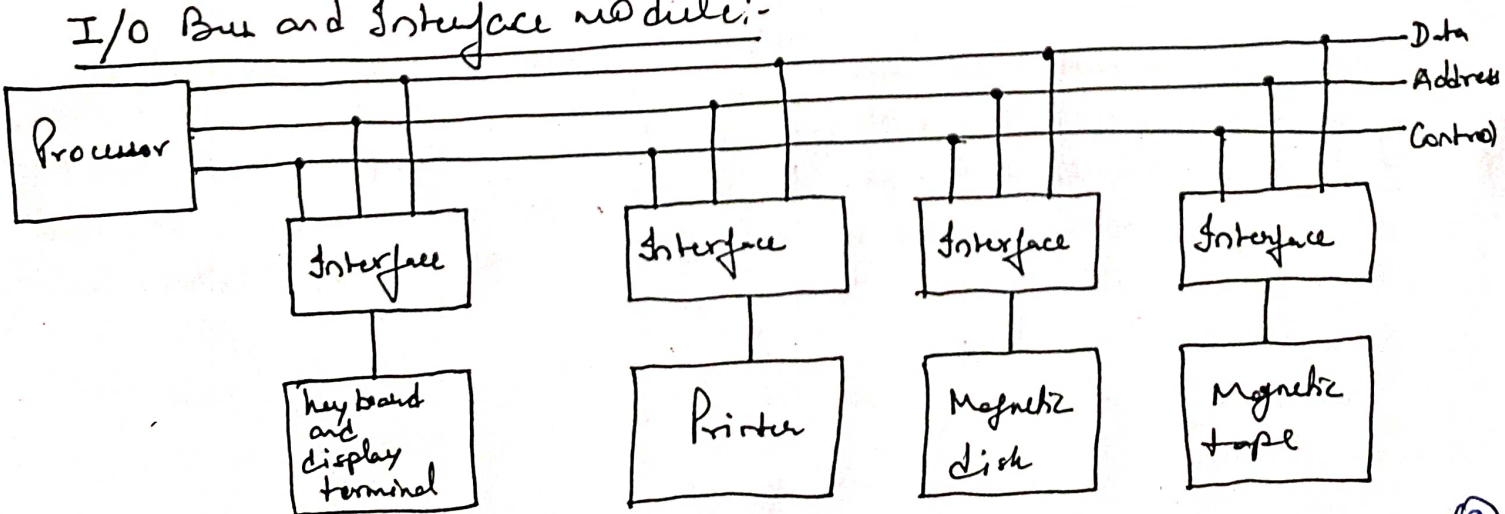
Derivation of BCD Adder :-

Binary Sum					BCD sum					Decimal
k	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

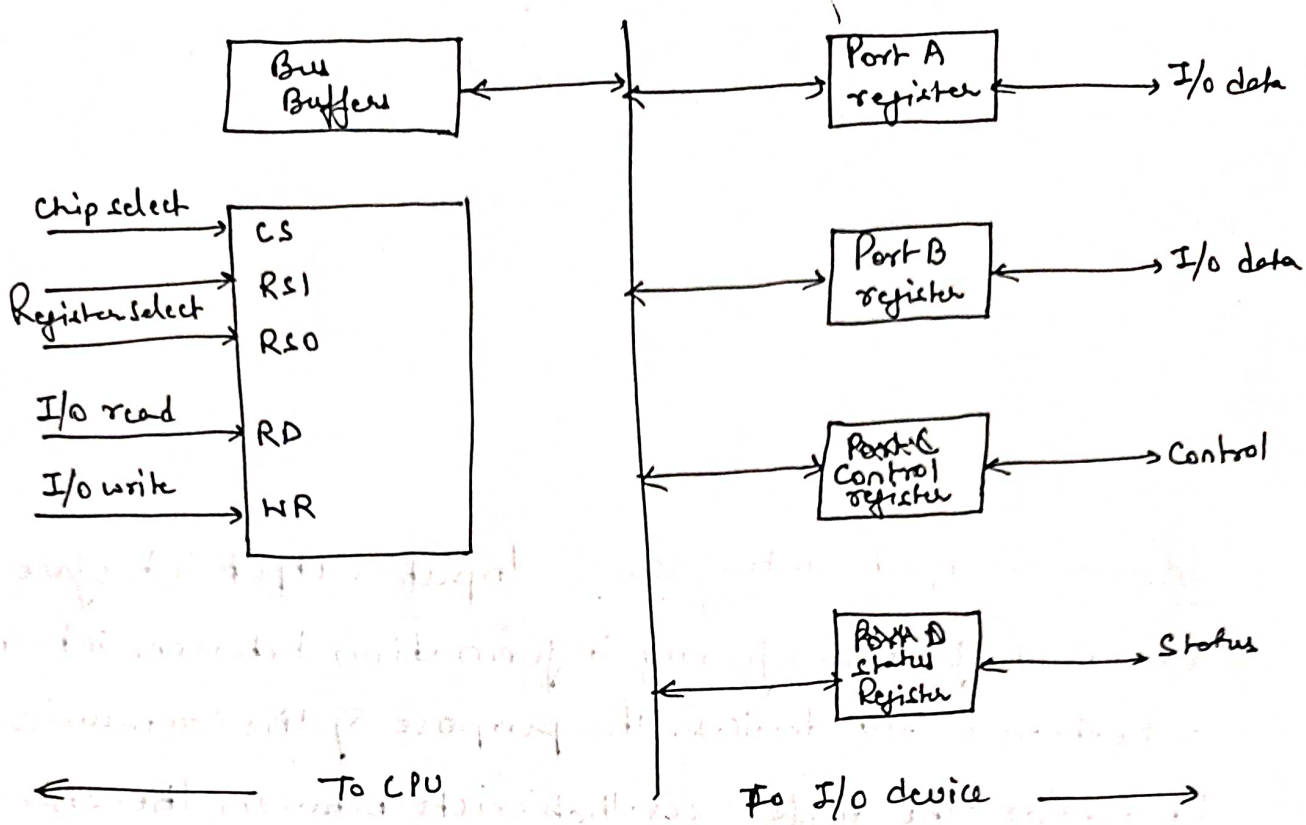
Input-Output Interface:- Input-output interface provides a method for transferring information between internal storage and external I/O devices. The purpose of the communication link is to resolve the differences that exists between the central computer and each peripheral. The major differences are:-

1. Peripherals are electromechanical and electromagnetic devices and their manner of operation is different from the operation of CPU and memory which are electronic devices.
2. The data transfer rate of peripherals is usually slower than the transfer rate of CPU.
3. Data codes and formats in perpherals differ from the word format in the CPU and memory.
4. The operating modes of peripherals are different from each other and each must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

I/O Bus and Interface Module:-



Connection of I/O Bus to I/O devices



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

Example) I/O Interface unit

Asynchronous Data Transfer :- 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.

In most cases the internal timing of each units are said to be independent to 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 b/w them, this data transfer is called Asynchronous data transfer

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

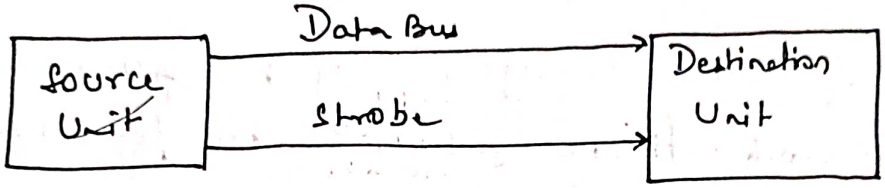
- **Strobe Control :-** A strobe pulse is supplied by one unit to indicate to the other unit when the transfer has to occur.
- **Handshaking :-** This method is commonly used to accompany each data item being transferred with a control signal that indicates data in the bus. The unit receiving the data item responds with another signal to acknowledge receipt of the data.

The strobe pulse and handshaking method of asynchronous data transfer is not restricted to I/O transfer.

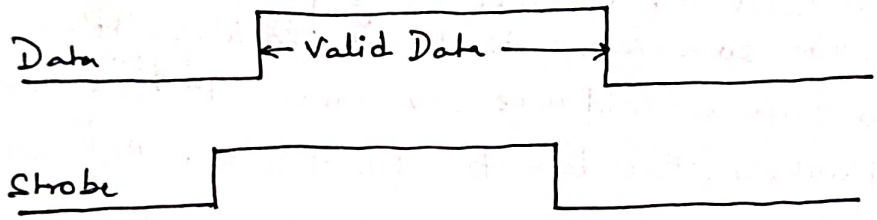
Asynchronous data transfer Methods :- The asynchronous data transfer between two independent units requires that control signals be transmitted b/w the communicating units to indicate when they send the data. Thus, the two methods can achieve the asynchronous way of data transfer.

1. Strobe control methods:- The strobe control method of asynchronous data transfer employs a single control line to time each transfer. This control line is also known as a strobe, and it may be achieved either by source or destination, depending on which initiate the transfer.

(a) Source initiated strobe:- In the below block diagram, you can see that strobe is initiated by strobe, and as shown in timing diagram, the source unit first places the data on the data bus.

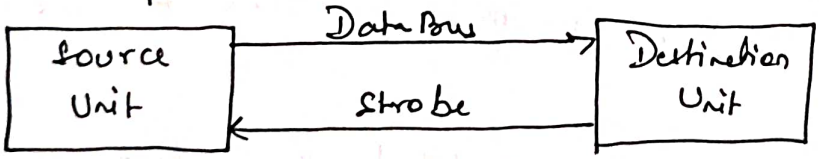


(a) Block Diagram

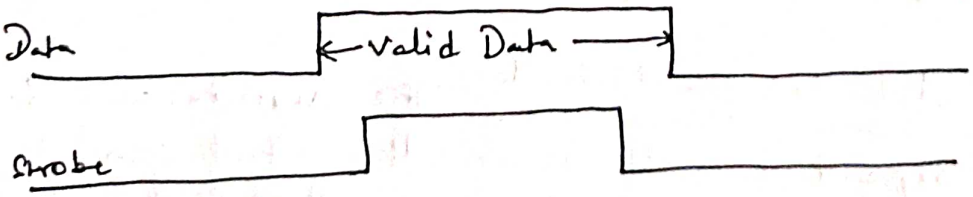


(b) Timing Diagram.

(b) Destination initiated strobe:- In the below block diagram, you see that the strobe initiated by destination, and in the timing diagram, the destination unit first activates the strobe pulse, informing the source to provide the data.



(a) Block Diagram



(b) Timing Diagram

The source unit responds by placing the requested binary information on the data Bus. the data Must be valid and remain on the bus long enough for the destination unit to accept it.

The falling of the strobe pulse can be initiated use again to trigger a destination unit to accept it/register.

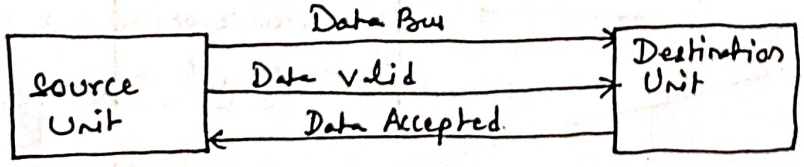
The destination unit then disables the strobe. finally, and source removes the data from the data bus after a determined time interval.

In this case, the strobe may be a memory read control from the CPU to a memory unit. The CPU initiates the read operation to inform the memory, which is a source unit, to Place the selected word into the data bus.

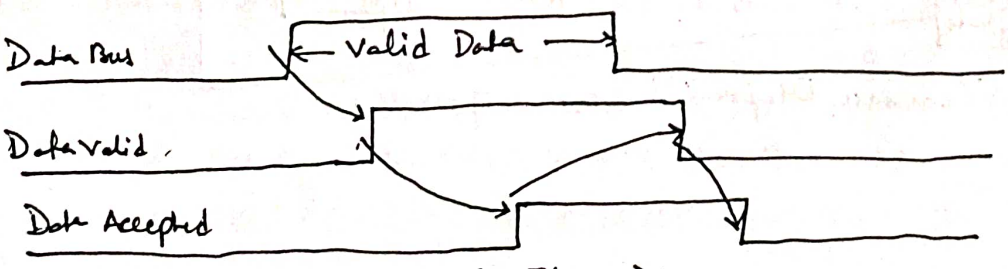
2. Handshaking Method:- The strobe method has the disadvantage that the source unit that initiates the transfer has no way of knowing whether the destination has received the data that was placed in the bus. Similarly a destination unit that initiates the transfer has no way of knowing whether the source unit has placed data on the bus.

So this problem is solved by handshaking method. The handshaking method introduces a second control signal line that replays the unit that initiates the transfer.

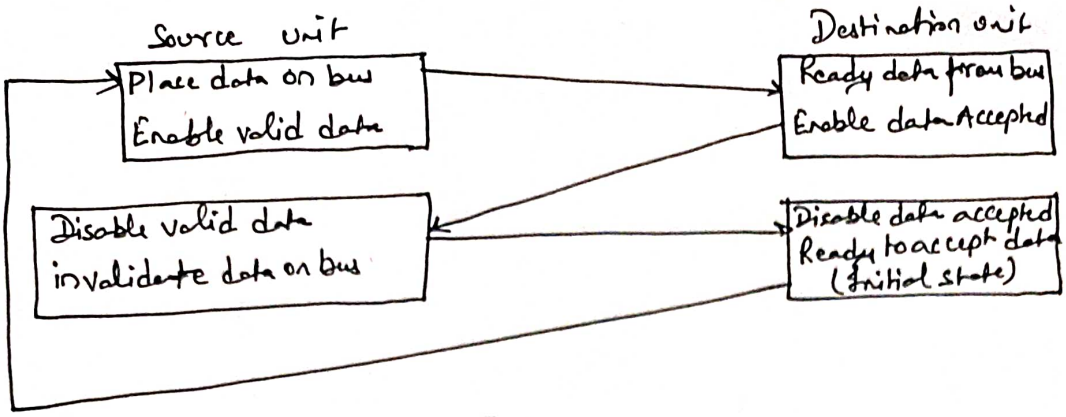
- Source initiated handshaking:- In the below block diagram, you can see that two handshaking lines are "data valid", which is generated by the source unit, and "data accepted", generated by the destination unit.



(a) Block Diagram



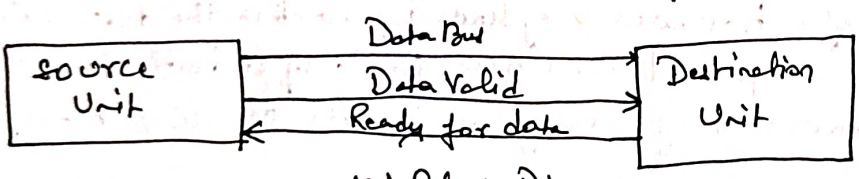
(b) Timing Diagram



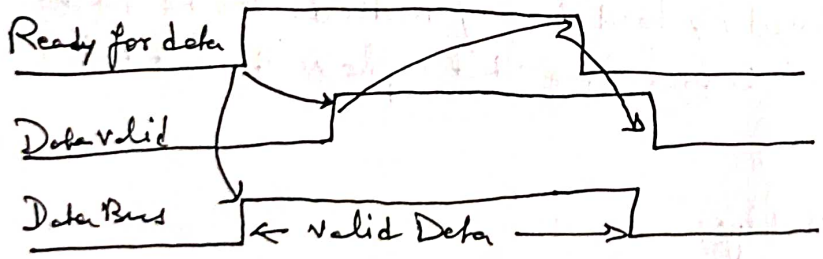
(c) Sequence Diagram.

source initiates a transfer by placing data on the bus and enabling its data valid signal. The destination unit then activates the data accepted signal after it accepts the data from the bus.

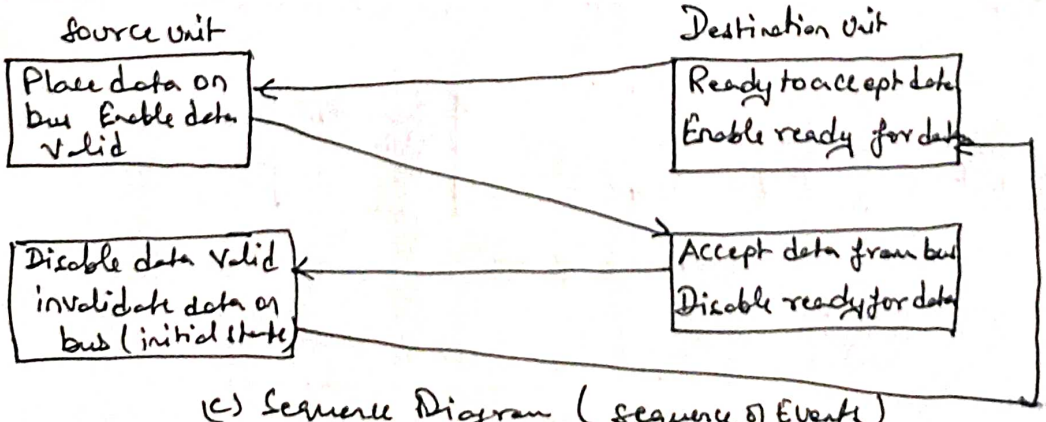
• Destination Initiated handshaking:- In the below block diagram, you see that the two handshaking lines are "data valid", generated by the source unit, and "ready for data" generated by the destination unit.
Note that the name of signal data accepted generated by the destination unit has been changed to ready for data to reflects its new meaning.



(a) Block Diagram



(b) Timing Diagram



(c) Sequence Diagram (sequence of Events)

The destination transfer is initiated, so the source unit not place data bus until it receives a ready data signal from the destination unit. After that, the handshaking process is the same as that of the source initiated.

The sequence of events is shown in its sequence digram, and the timing relationship b/w signals is shown in its timing digram. Therefore, the sequence of events in both cases would be identical.

Modes of Transfer:- The binary information that is received from an external device is usually stored in the memory unit. The information that is transferred from the CPU to the external device is originated from the memory unit. CPU merely Processes the information but the source and target is always the memory unit. Data transfer between CPU and the I/O devices may be done in different modes.

Data transfer to and from the peripherals may be done in any of the three Possible ways.

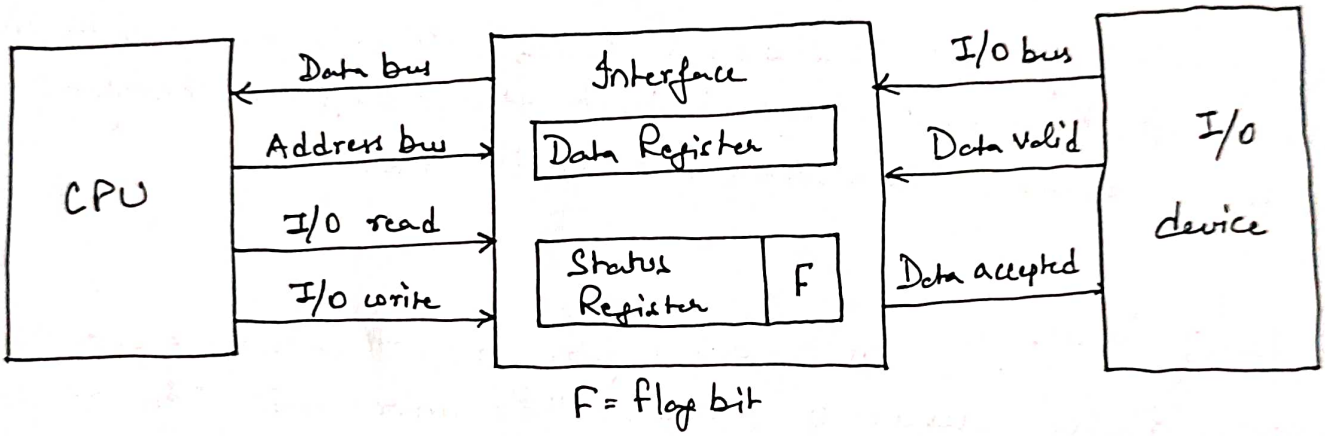
1. Programmed I/O.
2. Interrupt Initiated I/O.
3. Direct memory Access.

1. Programmed I/O:- It is due to the Result of the I/O instructions that are written in the computer program. Each data item transfer is initiated by an instruction in the program. Usually the transfer is from a CPU register and memory. In this case it requires constant monitoring by the CPU of the peripherals devices.

For eg:- it is assumed that the device is sending a sequence of bytes that must be stored in memory. The transfer of each byte requires three instructions.

1. Read the status Register.
2. Check the status of the flag bit and branch to step1 if not set or to step3 if set.
3. Read the data register.

Each byte is read into a CPU register and then transferred to memory with a store instruction. A common I/O programming task is to transfer a block of words from an I/O device and store them in a memory buffer.



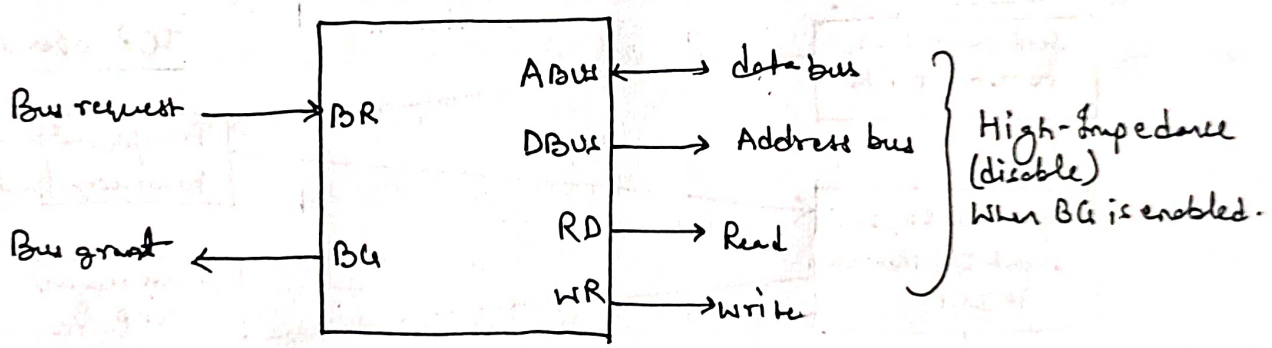
Data transfer from I/O device to CPU

2. Interrupt Initiated I/O:- An alternative to the CPU constantly monitoring the flag is to let the interface inform the computer when it is ready to transfer data. This mode of transfer uses the interrupt facility. While the CPU is running a program, it does not check the flag. However, when the flag is set, the computer is momentarily interrupted from proceeding with the current program and is informed of the fact that the flag has been set. The CPU deviates from what it is doing to take care of the input or output transfer. After the transfer is completed, the computer returns to the previous program to continue what is doing before the interrupt.

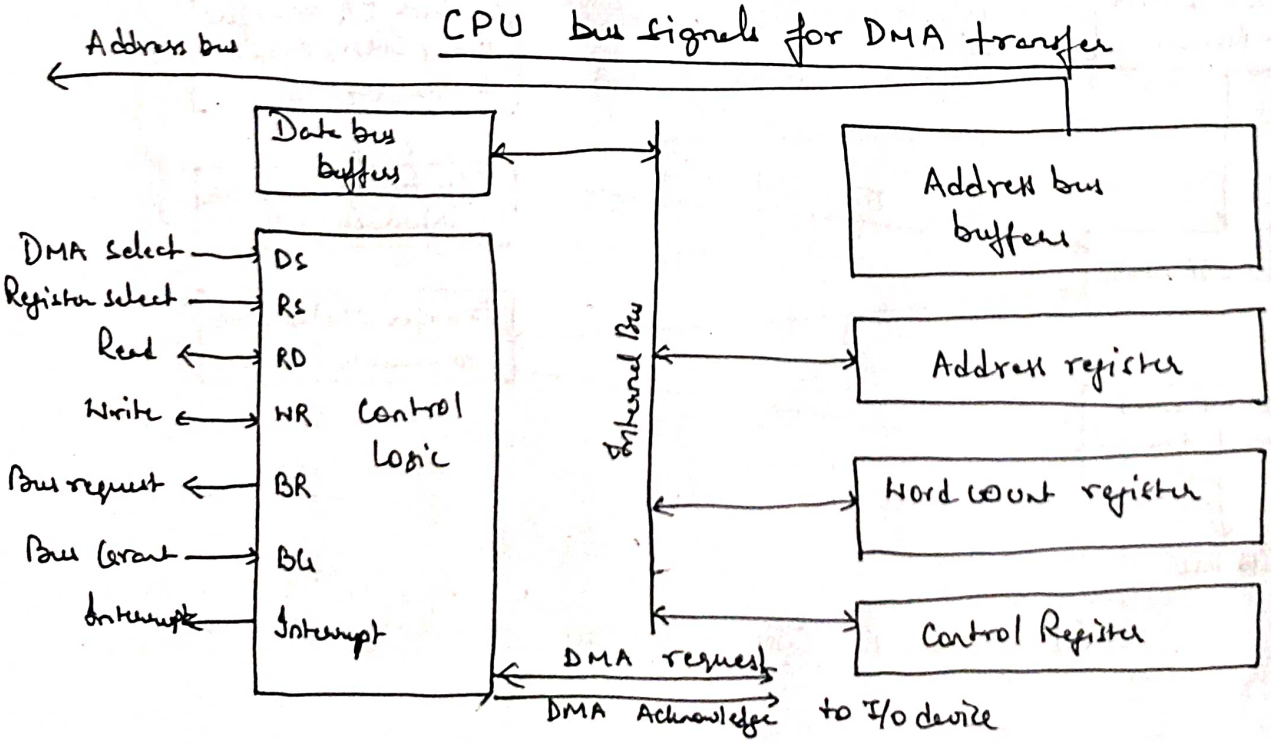
Both the methods programmed I/O and Interrupt-driven I/O suffer from two inherent drawbacks.

- The I/O transfer rate is limited by the speed with which the processor can test and service a device.
- The processor is tied up in managing an I/O transfer; a number of instructions must be executed for each I/O transfer.

3. **Direct memory Access (DMA)**:- The transfer of data b/w a fast storage device such as magnetic disk and memory is often limited by the speed of the CPU. Removing the CPU from the Path and letting the peripheral device manage the memory buses directly would improve the speed of transfer. This transfer technique is called DMA..



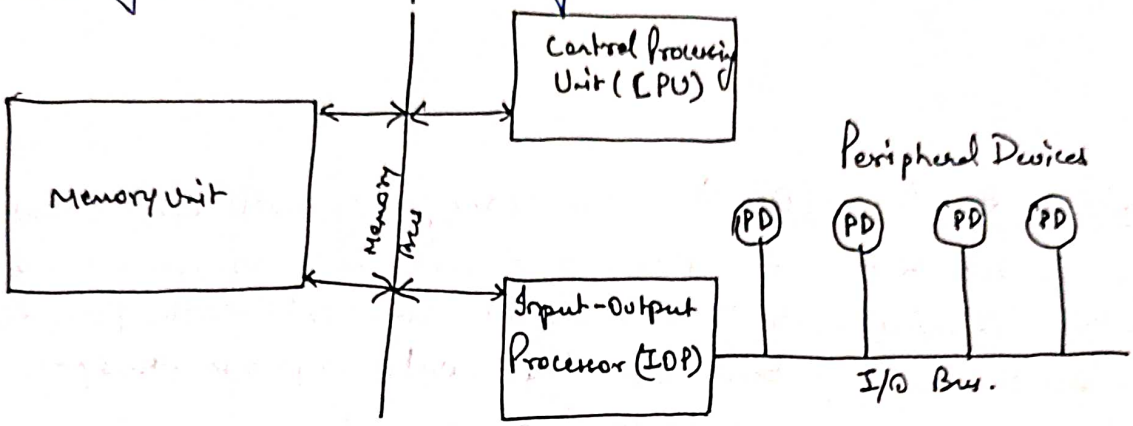
CPU bus signals for DMA transfer



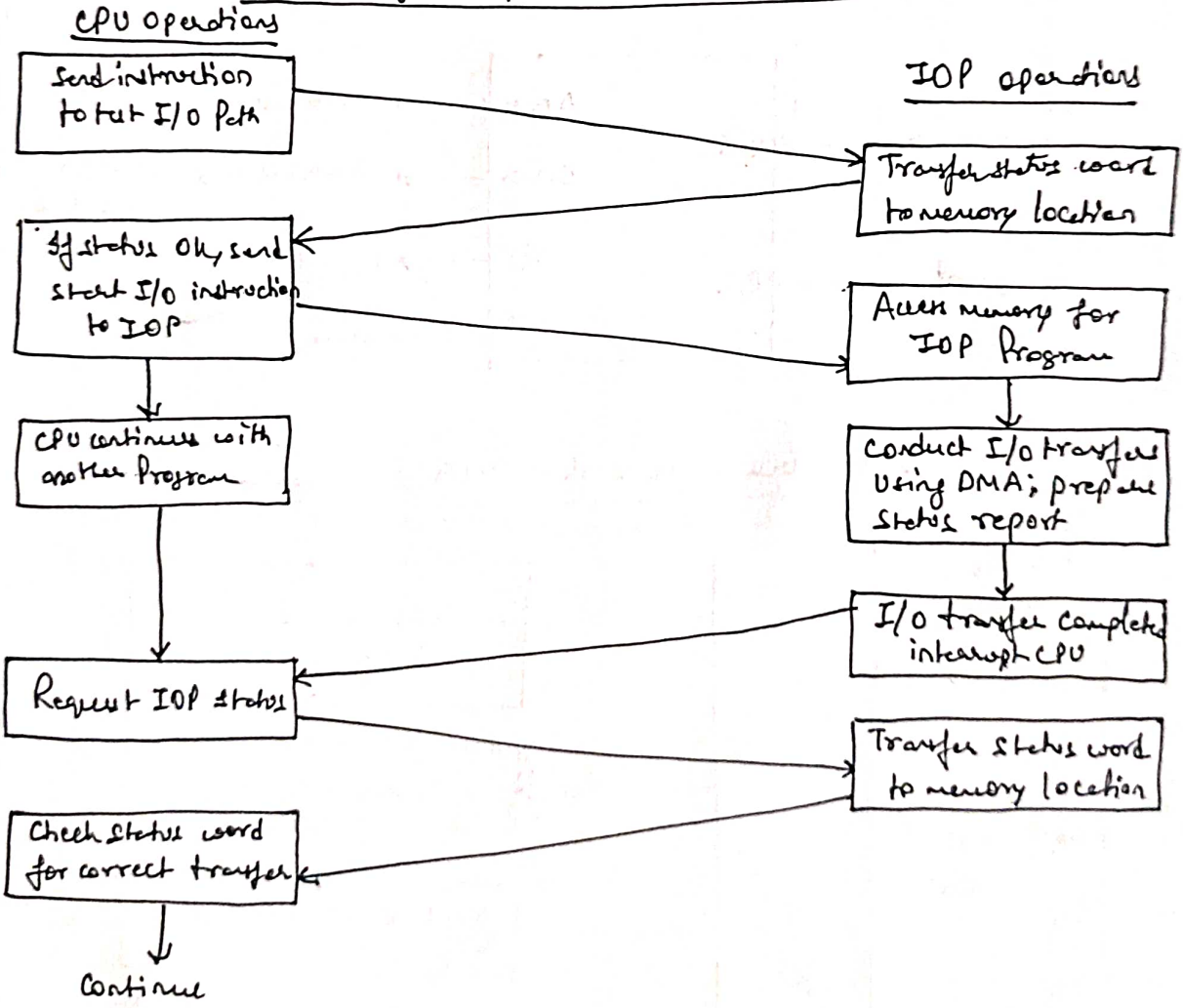
Block Diagram of DMA Controller

Input-Output Processor (IOP):-

An input-output processor (IOP) may be classified as a Processor with direct memory access capability that communicates with I/O devices. In this configuration, the computer system can be divided into a memory unit, and a number of processor comprised of the CPU and one or more IOPs.



Block diagram of a computer with I/O Processor



CPU - IOP Communication

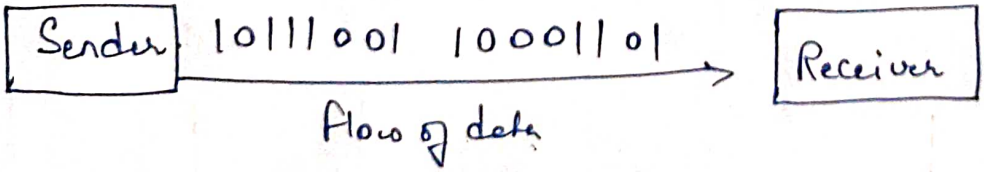
Serial Communication:-

Serial communication is the process of sequentially transferring the information/bits on the same channel. Due to this, the cost of wire will be reduced, but it slows the transmission speed. Generally, communication can be described as the process of interchanging information between individuals in the form of audio/video, verbal words and written documents. The serial Protocol is run on every device that can be our mobile, Personal computers and many more with the help of some Protocols.

The serial communication can either be asynchronous or synchronous

1. Synchronous communication:-

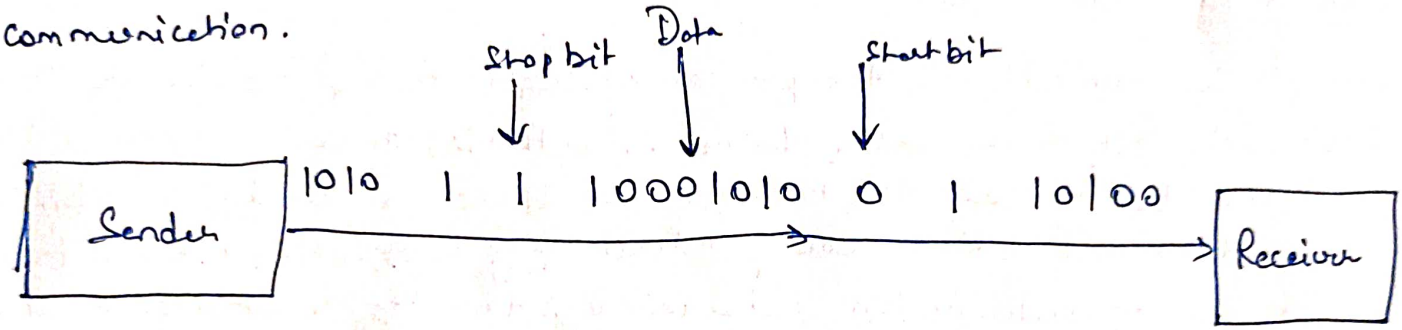
In Synchronous communication, the frames or data will be constructed with the help of combining the group of bits. That frames will be continuously sent in time with a master clock. It uses a synchronized clock frequency to operate the data of sender or receiver. In synchronous comms, there is no need to use the gaps, start bits and stop bits. The time taken by the sender and receiver is synced that's why the frequency of timing error will be less, and the data will move faster.



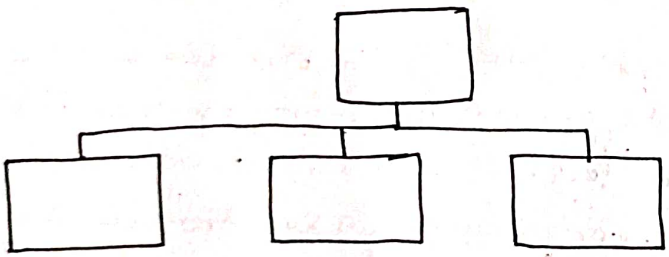
Synchronous Transmission

2. Asynchronous Communication:-

In asynchronous communication, the groups of bits will be treated as an independent unit and these data bits will be sent at any point in time. In order to make synchronization between sender and receiver, the stop bits and start bits are used between the data bytes. These bits are useful to ensure that the data is correctly sent. The time taken by data bits of sender and receiver is not constant, and the time b/w transmissions will be provided by the gaps. In asynchronous communication, we don't require synchronization b/w the sender and the receiver devices, which is the main advantage of asynchronous communication.



Transmission Modes in Serial Communication:-



Half Duplex

Simplex

Full Duplex