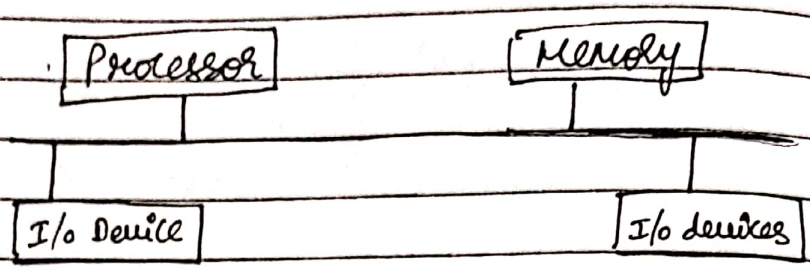
10/10/19

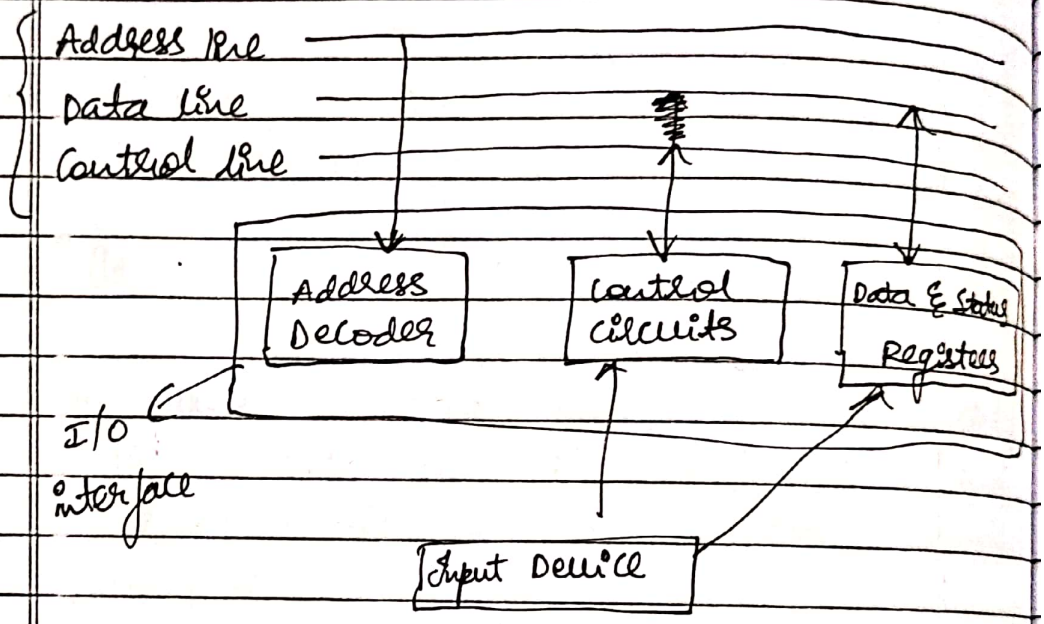
# UNIT 4

## I/O ORGANISATION & ARITHMETIC

### ① ACCESSING I/O DEVICES




Single-Bus structure




I/o interface for an I/o device

* I/o device examines the lower order bits of address bus to determine whether they should respond.

2 M/p

Q. What is Memory-Mapped I/o.

* Nothing but memory and I/o device share same address space.

Ans: Memory-mapped I/O is an arrangement where memory and I/O device share the same address space.

eg:-

```
MOVE   DATAIN , R0
MOVE   R0 , DATAOUT
```

DATAIN → I/P Buffer of input device.
DATAOUT → O/P Buffer of output device.

Q. What is advantage of seperate i/o address space over Memory mapped I/O?

⇒ That I/O devices deal with fewer address lines.

PROGRAM CONTROLLED I/O

* Status Register for keyboard

| |
|---|
| SIN = 1 |
| SIN = 0 |

→ when a character is entered ~~through keyboard~~

→ When the processor reads the character.

* The Program controlled I/O repeatedly reads the status register which leads to slow execution.

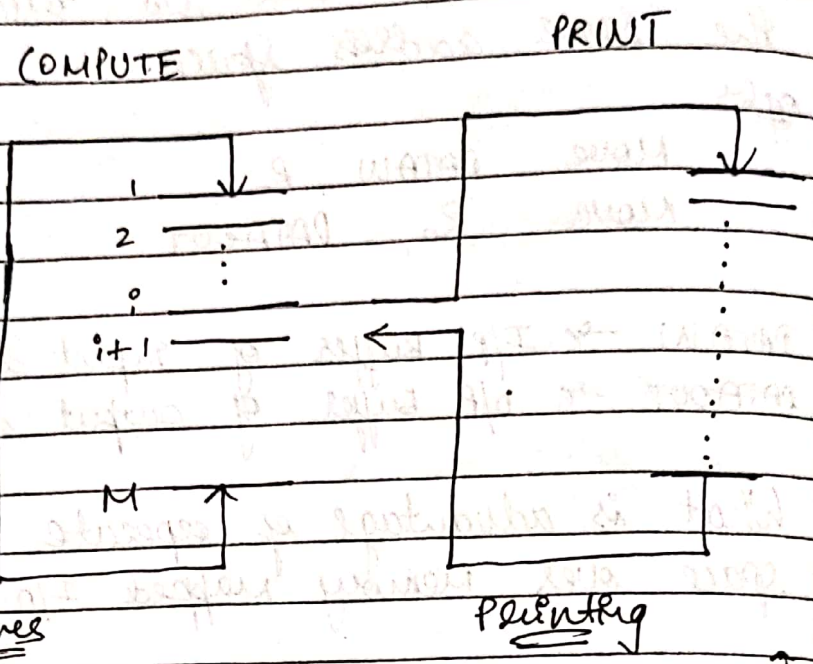
To overcome this Interrupts are used in which the I/O device sends a special signal when it is ready for a data transfer.

INTERRUPTS

Printer ⇄ Processor

Date :
Page No :

Example



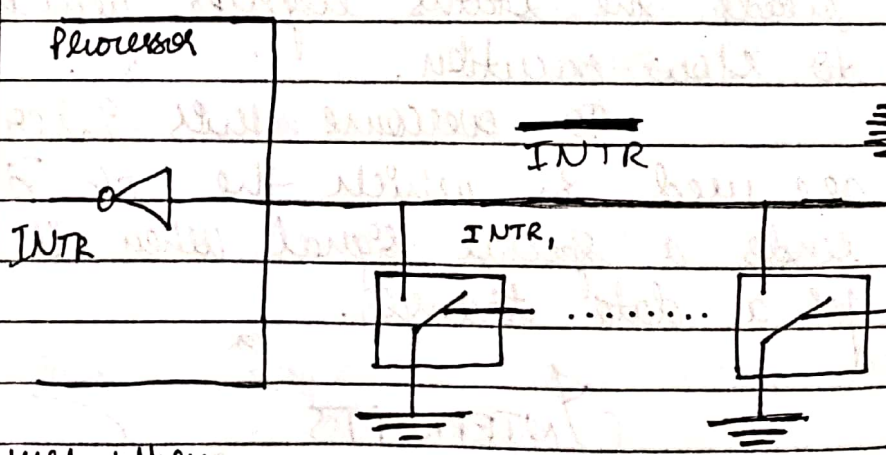

☆ ISR (Interrupt Service Routine)

It is a program that is executed when interrupt occurs.

☆ Interrupt Latency

It is the delay b/w time when interrupt request is received and the start of execution of ISR.

☆ Interrupt Hardware




Here, when
INTR = 0 or INTR = 1 an interrupt occurs.

11/10/19

Date :
Page No :

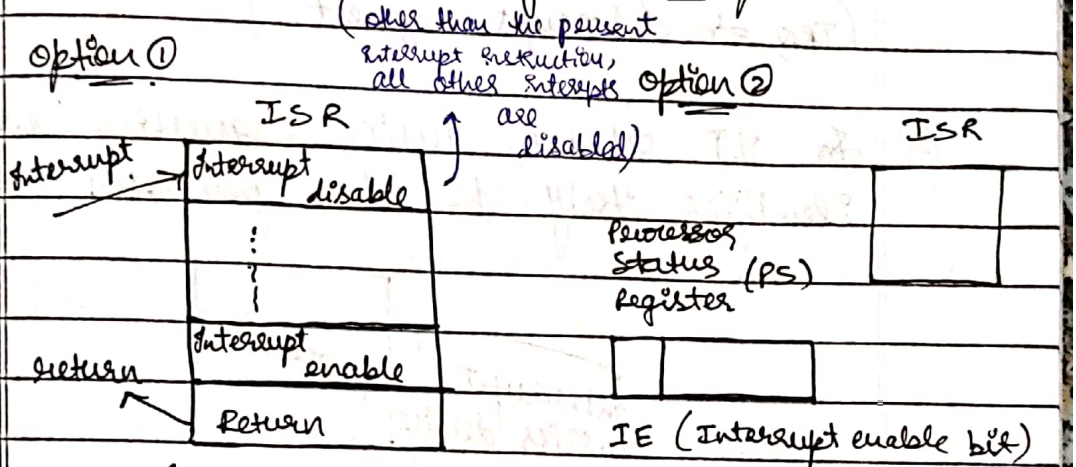
## Interrupt request line implementation using open.

$$INTR = INTR_1 + INTR_2 + \dots + INTR_n$$

* All devices are connected to the interrupt request line through switches to the ground.

When device requests an interrupt, it closes its switch and ~~voltage~~ INTR becomes 0, and interrupt occurs.

## Enabling and Disabling Interrupts




(Should explain theoretically also)

*. In option 1, processor uses interrupt disable instruction as 1st inst in the ISR.

*. In option 2, the IE bit is set to 1 in the PS register to enable further interrupts.

IE = 1 → Processor enables interrupt.

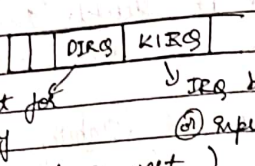
IE = 0 → " disables. "

Date :
Page No :

## Handling Multiple Devices

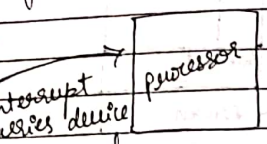
① Polling ② Vectored interrupts ③ Priority (Interrupt nesting) ④ Daisy chain (Simultaneous requests)

① In polling scheme, the first device encountered with its IRQ bit set is the device that should be serviced.

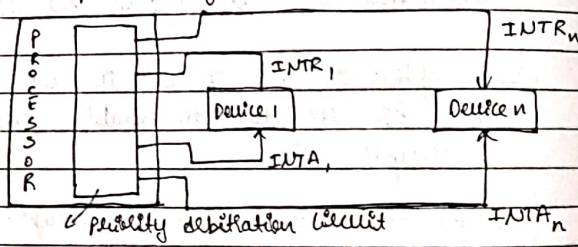



(IRQ → Interrupt request)

② In VI scheme, device requesting interrupt identifies itself to the processor.




③ Interrupt Nesting (Priority)

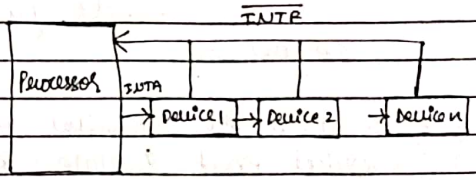



A → Acknowledgement

In multiple priority scheme, processor's priority level is changed according to the device priority that it is servicing.

In this scheme, request is accepted only if it has a higher priority level than that of which is currently assigned to the processor.

④ Daisy-Chain




Here,

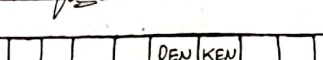
Devices that are electrically closest to the processor has the highest priority, the next devices along the chain have the next priority.

* Devices can be organised into groups and each group is connected at a different priority level. Within a group devices are connected in daisy chain.
(It is the combination of ③ & ④)

## INTERRUPTS

Controlling Device requests

Control register




Interrupt enable bit.

14/10/19

Status register | .... | DIRQ | KIRQ |

DIRQ (Interrupt request)

DEN - Display enable.
KEN - Keyboard enable.
DIRQ - Display Interrupt request.
KIRQ - Keyboard " "

Example:- If DEN=1 & DIRQ=1
Then processor accepts interrupt request from display.
(or else no).

Q. What is the mechanism adopted so that the devices don't generate Interrupt requests?

Ans → Write all the cases with the diagram.

* Two mechanisms are used for controlling Interrupt requests.
  ① Interrupt enable
  ② Interrupt request

* IE bit determines whether device is allowed to generate Interrupt requests.

* IRQ bit determines whether Interrupt is generated.

☆ EXCEPTION
It refers to an event that causes an Interruption.

Q. Differentiate b/w I/o Interrupt and Exception.

Exceptions
① Recovery from errors → memory error, program error (eg:- Division by zero) (%)
② Debugging
③ Privilage Instructions
(Exception service Routine - ESR)

| I/o Interrupt | Exception (Program generated Interrupt) |
|---|---|
| ① Finishes execution of current process and then Interruption is handled | ① Suspends execution of current instructions & Interruption is handled |

* Error checking code in main memory allows error detection in the stored data.

② ☆ Debugging
Debugger helps programmers find error in the program.
Trace Breakpoints
↳ (Memory locations & register contents)
exception occurs after

* In trace, execution of every Inst.

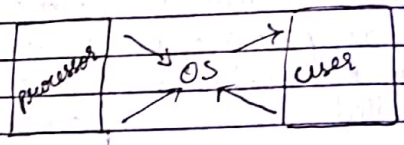
* Breakpoints are used to Interrupt the program at specific points selected by user.

15/10/19

* protect the OS from being corrupted, certain instructions called privilage inst are executed only when the processor is in supervisor mode.

* OS co-ordinates all the activities in a computer like communication b/w processes.

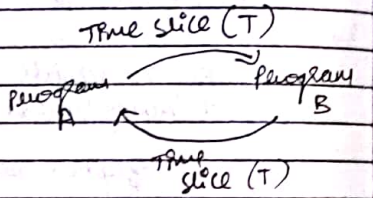
Interrupts in Operating System (OS)




process → program in execution.

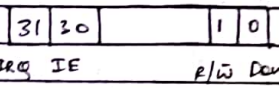
Interaction between application program & OS user Multitasking

OSINIT
↓
OSSERVICES
↓
IOINIT
↓
KBDINIT
↓
SCHDULER
↓
program
↓
KBDATA



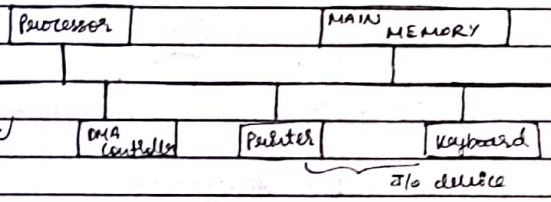

OSINIT → Initializes OS services
OSSERVICES → Calls appropriate I/O operations.
IOINIT → Initializes I/O devices.
KBDINIT → Initializes keyboard.
SCHEDULER → Selects processes.
KBDATA → Data transfer TO/FROM keyboard

* Direct Memory Access (DMA)




Status and control

* Use of DMA Controller in Computer (CMAC)




* DMA is an approach used in which large blocks of data are transferred directly between the external device and the main Memory without continous intervention by the processor.

* Sends starting address, number of words in block, direction of transfer (R/W).

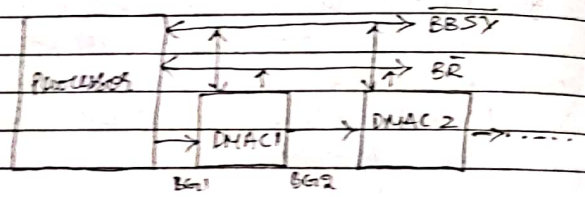
17/10/19

* When IE = 1 DMA controller raises an interrupt after completing transfer of block of data.
* Done refers that the DMA process is completed.
* IE tells the processor that the transfer is completed.
* IEq sends an interrupt request for the process asking for DMA to take place.

* "Cycle stealing" is a process where the DMA Controller steals the memory cycles from the processor as the requests by DMA devices is given high priority.

* Bus arbitration
  - Centralized
  - Distributed

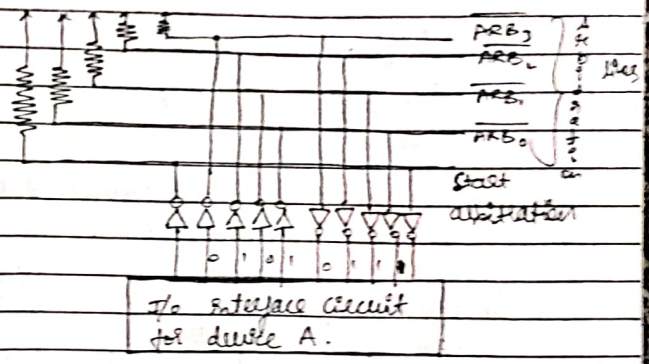
① Centralized arbitration




BG → Bus Grant.
BR → Bus Request.
BBSY → Bus Busy.

BUS ARBITRATION

② Distributed arbitration




Device A → 0101
Device B → 0110
Logical OR → 0111

In the case both the devices will want to have the bus for operation, logical OR of the arbitration bits will be done and whichever device is closer to the logical OR (result) will be given the bus for operation.

* Arbitration → It is the process carried out by the processor when it gives the bus control to any device which requests based on various priorities.

* Each device on the bus is assigned a 4 bit identification number, when one or more devices request the bus, the logical OR of their identification no is taken

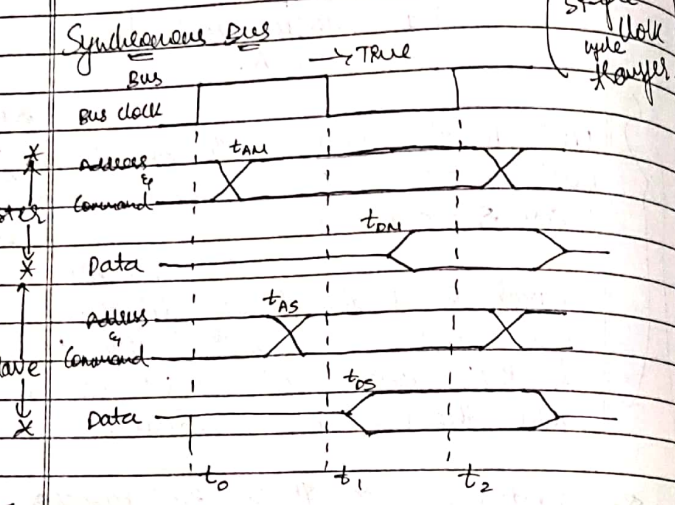
the device which is closer to the logical OR is assigned the bus control.

## BUS.

Synchronous Bus. | Asynchronous Bus.

* In Syn Bus, all devices transfer data using a common clock.

Synchronous Bus




(single clock cycle transfer)

Q. Show the timing diagram for input transfer on a synchronous bus.

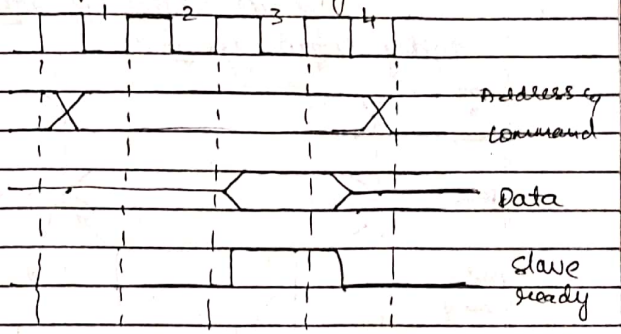
* $t_2 - t_0$ = time of one clock cycle
= It must be equal to time of slowest device.

(∵ time of one clock will be only one (fixed), so accomodate slowest device, $t_2 - t_0$ must be equal to time of slowest device).

NOTE:- In exam write what is happening at time $t_0, t_1, t_2, t_{DM}, t_{AM}, t_{AS}, t_{DS}$

* $t_0$ → Time at which I/O operation starts.
$t_1$ → Data operation starts.
$t_2$ → Data operation ends. (I/O transfer ends)
$t_{AM}$ → Address and command loaded by master.
$t_{AS}$ → " " " seen by slave.
$t_{DM}$ → Data seen by master.
$t_{DS}$ → " loaded by slave.

## Multiple Cycle Transfer

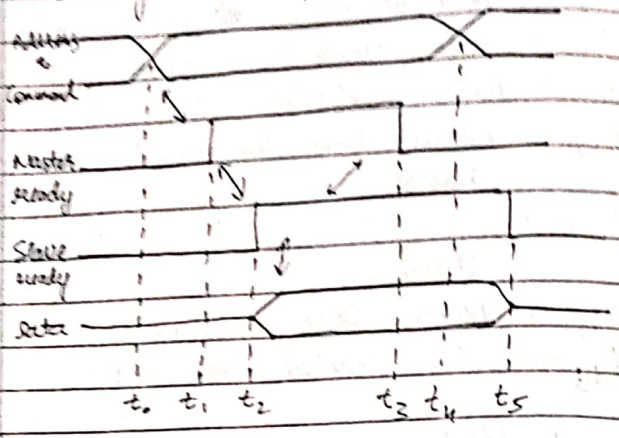



* In multiple cycle transfer, multiple pulses are produced by the clock for different devices to suit the timing needs of different devices

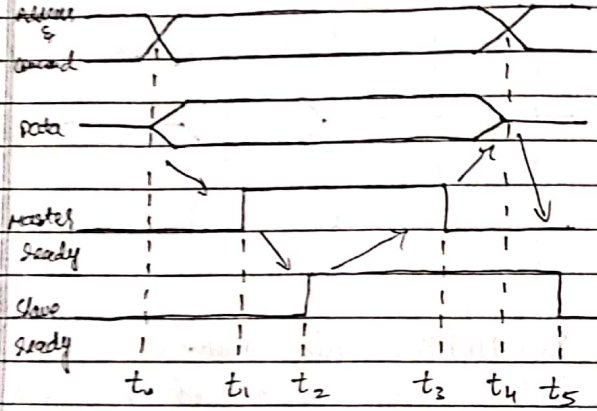
NOTE:- at any place, delay is due to the bus hardware circuitry.

Note

Advantages of
a. Handshake control of data transfer during an input operation. (eg: read)




b. Handshake control of data transfer during output operation. (eg: write)




a. $t_0$ → Master places address and command info on the bus.
$t_1$ → Master informs other devices that address and command is ready by enabling the master ready signal.
$t_2$ → The selected slave performs decoding the address and info by placing the data on the bus.
$t_3$ → Slave ready signal arrives at the master indicating that input data is available on the bus, so master ready signal is disabled.
$t_4$ → Master removes the address & command info from the bus.
$t_5$ → When the device interface receives 1 to 0 transition of master ready signal, it removes the master and slave ready signal from the bus.

b. Write similarly

2M Q. Compare sync and async bus

| | |
|---|---|
| ① Clock is present | ① No clock |
| ② Complexity due to clock | ② less complex |
| ③ Not slow when compared to async | ④ Slow |

OR

| | |
|---|---|
| ① Clock is used, so complex circuitry | ① No clock, so simple circuitry. |
| ② Faster | ② Slow due to handshake between master & slave. |

NOTE: In real world, synchronous bus with multiple cycle transfers is used.

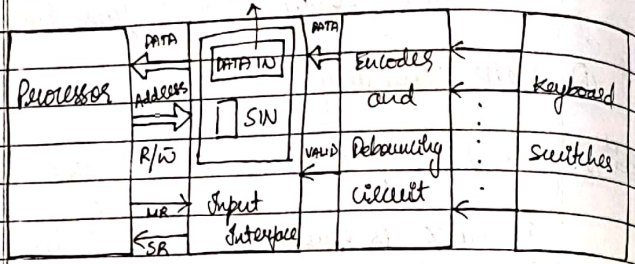
# PORTS (I/O INTERFACE)

① Parallel port ② Serial port

* I/O interface consist of circuitry required to connect the I/O device to the computer bus.

* Port → It represents the data path with its associated controls to transfer data b/w interface and I/O device.

ASCII code will be stored




MR → Master ready SR → Slave ready

* Parallel Port → Eg:- Keyboard, Printer (devices which are close).

* Serial Port → Eg:- long wired devices. (devices which are a bit far).

Status flags:

* SIN = 1 (Data placed in DATA IN register by the keyboard).

SIN = 0 (Processor read the data placed).

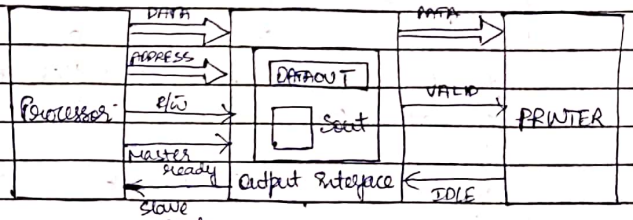
* VALID = 1 (Tells that a particular button is pressed).

* When a key is pressed on the keyboard, it switches closes and establishes a path for an electrical signal.

* Debouncing circuit denotes the effect of bouncing when a key is pressed.

21/10/19

# PARALLEL PORT




Printer to Processor Connection

Sout = 1, printer is ready to accept new character.

Sout = 0, the processor loads new character into DATAOUT.

* When a printer is ready to accept a character, it asserts (enables) the idle signal.

DATAOUT :→ Holds the data to be printed by processor.

VALID :→ Single enabled to indicate master (processor) is ready.

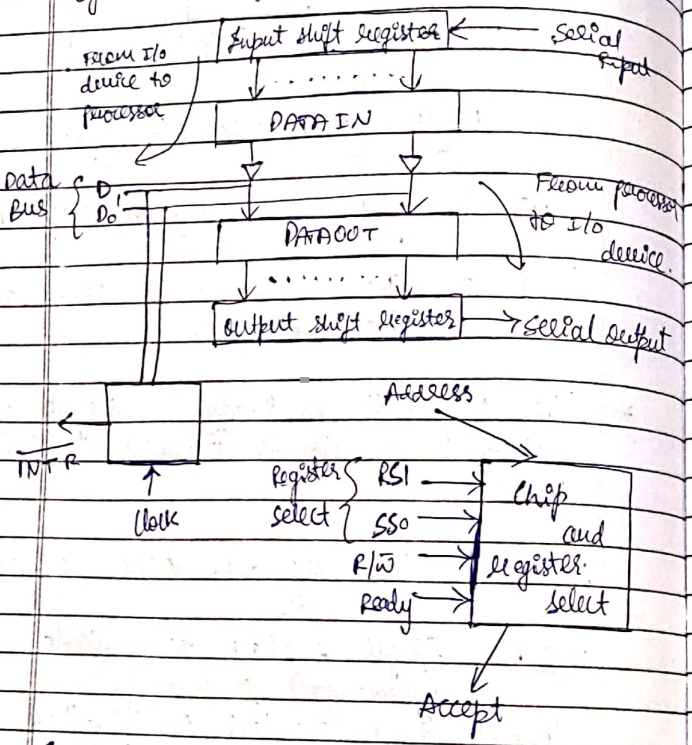
IDLE :→ Single enabled to indicate slave (printer) is ready.

NOTE:- Here, address means the address of I/O device.

* In the diagram the printer sends parallel the new character and negates the BUSY signal which causes the interface to deactivate the IDLE signal.

Def. SERIAL PORT (Eg:- Dial-up modems, routers)

It is used to connect processor to I/O devices that requires transmission of data one bit at a time.




Shift Register → Converts serial to parallel and vice versa.

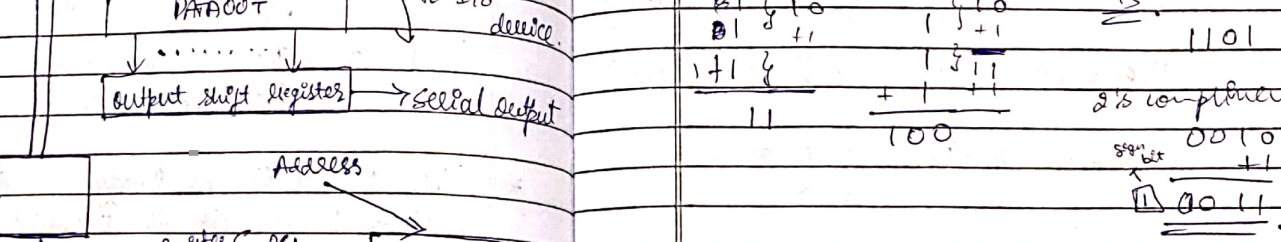
* The interface circuit for a serial port can communicate in a bit serial fashion on device side and bit parallel fashion on the bus side.

Q. What happens if shift registers are eliminated?

A. If shift registers are eliminated the processor gets slowed down because of serial transmission.

BOOTH ALGORITHM

carry
1+0 = 1 1+1 = 10
0+0 = 0 0+1 = 1



31/10/19

## BIT RECODING

Step①. Take 2's compliment of negative number.
Step②. Convert multiplier using Bit recoding table.

| | |
|---|---|
| 000 | 0 |
| 001 | +1 |
| 010 | +1 |
| 011 | +2 |
| 100 | -2 |
| 101 | -1 |
| 110 | -1 |
| 111 | 0 |

Step③. Multiply multiplicand and Bit Recoded multiplier.

Note:- ① Multiplicand x (+2) = multiplicand x 10
② Multiplicand x (-1) = 2's compliment of multiplicand.
③ Multiplicand x (-2) = 2's compliment of multiplicand x 10.

Q. Multiply +13 x -6 using bit recoding technique

→ Step① Convert -6 to 2's complement

110
1's comp → 001 → 001
+1
1]010 (2's comp)
sign bit ↓
④

13 → 0]1101 → ⑪
↓
sign bit

Make ④ and ⑪ equal
① Add one more sign bit
1]1]010 → ⑨
⑪ 0]1101

Step③. multiplier 1]1]010[0 (Add 0 to LSB)
Add 1 to MSB to solve
0 -1 -2

Step② multiply

| | | |
|---|---|---|
| 01101 x -2 | 0 -1 -2 | 1]1]0010 |
| 2's of multiplicand x10 | | 10011 ++ |
| 01101 | | 00000 ++++ |
| 10010 → 1's | | 1]1]0110010 → ③ |
| +1 | | answer ↓ |
| 10011 (2's) x10 | | (In excess, left most is enough) |
| 100110 | | To verify |

13 x -6 = -78
↓
convert to binary and take 2's comp
Compare with ③

Q. -11 x 27

→ Step① Convert -11 to 2's

| | |
|---|---|
| 01101 x -1 | |
| 2's of multiplicand | |
| 10010 (1's) | |
| +1 | |
| 10011 (2's) | |

-11 1011

0100 (1's)

+1

0101 (2's) → ①

sign bit

now, 24 , 0 11011 → ②

sign bit

now,

① = 11 0101

② = 0 11011

step② multiplier

0 1 1 0 1 1 0

+2 -1 -1

step③ multiply

1 1 0 1 0 1

+2 -1 -1

0 0 0 0 0 0 0 1 0 1 0 1

0 0 0 0 0 1 0 1 0 1 + +

1 1 0 1 0 1 0 + + + +

1 1 0 1 1 0 1 0 1 0 1

-11 × 29 ⇒ -297

100101001

011010110 (1's)

+1

011010111 (2's)

verified.

Q. multiply 010111 × 110110

sol step① already given

step② multiplier

1 1 0 1 1 0 0

-1 +2 -2

step③ Multiply

0 1 0 1 1 1

-1 +2 -2

010111

101000 (1's)

+1

101001 (2's) × 10

1010010

1 1 1 1 0 1 0 0 1 0

0 0 1 0 1 1 1 0 + +

1 0 1 0 0 1 1 + + + +

1 1 0 0 0 1 1 0 1 0

010111 × 10

0101110

1100011010.

* Restoring Division Method for integer division of unsigned numbers.

step① let 'n' = no of bits in dividend

'M' = Divisor (n+1 bits)

A ← 0 (Accumulator with n+1 bits)

Q ← Dividend (n bits)

step② Shift left A&Q one bit.

step③ perform A = A-M

step④ If A[n] = 1 (MSB) set Q0 ← 0 (LSB) and restore A else set Q0 ← 1

step⑤ n = n-1, if n ≥ 0 goto step②

Finally A holds remainder

Q holds quotient.

4/11/19

① Perform 12 ÷ 4 using restoring division.

→ 12 (dividend) = 1100 ; 4 (divisor) = 100

| n | M (-M) | A | Q | Action |
|---|---|---|---|---|
| 4 | 00100 | 00000 | 1100 | Initialization |
| | | 00001 | 100? | SL → AQ |
| | | 11101 (-ve) | 100? | A = A-M |
| | | | | A = A+(-M) |
| | | | | $Q_0 \leftarrow 0$ |
| | | 00001 | 1000 (LSB) | restore A |
| 3 | | 00011 | 000? | SL → AQ |
| | | 11111 | 000? | A = A-M |
| | | | | A = A+(-M) |
| | | | | $Q_0 \leftarrow 0$ |
| | | 00011 | 0000 | restore A |
| 2 | | 00110 | 000? | SL → AQ |
| | | 00010 | 000? | A = A-M |
| | | | 0001 | $Q_0 \leftarrow 1$ |
| 1 | | 00100 | 001? | SL → AQ |
| | | 00000 | 001? | A = A-M |
| | | | 0011 | $Q_0 \leftarrow 1$ |
| 0 | | | | |

∴ Answer, A (remainder) = 00000 (0)
Q (quotient) = 0011 (3)

② Perform 11 ÷ 3 using restoring division.

→ 11 (dividend) = 1011 ; 3 (divisor) = 011

| n | M (-M) | A | Q | Action |
|---|---|---|---|---|
| 4 | 00011 | 00000 | 1011 | Initialization |
| | | 00001 | 011? | SL → AQ |
| | | 11110 | 011? | A = A+(-M) |
| | | | | $Q_0 \leftarrow 0$ |
| | | 00001 | 0110 | (restore A) |
| 3 | | 00010 | 110? | SL → AQ |
| | | 11111 | 110? | A = A+(-M) |
| | | | | $Q_0 \leftarrow 0$ |
| | | 00010 | 1100 | (restore A) |
| 2 | | 00101 | 100? | SL → AQ |
| | | 00010 | 100? | A = A+(-M) |
| | | 00010 | 1001 | $Q_0 \leftarrow 1$ |
| 1 | | 00101 | 001? | SL → AQ |
| | | 00010 | 001? | A = A+(-M) |
| | | 00010 | 0011 | $Q_0 \leftarrow 1$ |
| 0 | | | | |

∴ Answer, A (remainder) = 00010 (2)
Q (quotient) = 0011 (3)

③ Perform 1000 ÷ 11 using RD.

| n | M | A | Q | Action |
|---|---|---|---|---|
| 4 | 00011 | 00000 | 1000 | Initialise |
| | | 00001 | 000? | SL AQ |
| | | 11110 | 000? | A = A+(-M) |
| | | | 0000 | $Q_0 \leftarrow 0$ Restore A |

| | | | | |
|---|---|---|---|---|
| 3 | 00011 | 00001 | 0000 | |
| | | 00010 | 000? | SL AQ |
| | | 11111 | 000? | A = A+(-M) |
| | | | 0000 | $Q_0 \leftarrow 0$ (restore A) |
| 2 | | 00010 | 0000 | |
| | | 00100 | 000? | SL AQ |
| | | 00001 | 000? | A = A+(-M) |
| | | | 0001 | $Q_0 \leftarrow 1$ |
| 1 | | 00001 | 0001 | |
| | | 00010 | 001? | SL AQ |
| | | 11111 | 001? | A = A+(-M) |
| | | | 0010 | $Q_0 \leftarrow 0$ (Restore A) |
| 0 | | 00010 | 0010 | |

∴ Answer, A (Remainder) = 00010 (2)
Q (quotient) = 0010 (2)

H.W * 16 ÷ 4 , 13 ÷ 5 , 17 ÷ 4

** Non-Restoring Division For unsigned numbers

Step ① let n → Number of bits in Dividend
M → n+1 bits of Divisor
A → 0 (n+1) bits (Accumulator)
Q → Dividend (n bits)

Step ② Do the following 'n' times
① If A[n] = 0 (MSB) then shift left AQ
A = A − M

else
{ Shift left AQ
A = A+M
}

② If A[n] = 0 → $Q_0 \leftarrow 1$
else $Q_0 \leftarrow 0$

Step ③ If A[n] = 1 → A = A+M

At last,
Q is quotient
A is Remainder

* 11 ÷ 3

11 → 1011 3 → 0011

| n | M | A | Q | Action |
|---|---|---|---|---|
| 4 | 00011 | 00000 | 1011 | Initialisation |
| | | 00001 | 011? | Shift left AQ |
| | | 11110 | 011? | A = A+(-M) |
| | | 11110 | 0110 | $Q_0 \leftarrow 0$ |
| 3 | | 11100 | 110? | Shift left AQ |
| | | 11111 | 110? | A = A+M |
| | | 11111 | 1100 | $Q_0 \leftarrow 0$ |
| 2 | | 11111 | 100? | Shift left AQ |
| | | 00010 | 100? | A = A+M |
| | | 00010 | 1001 | $Q_0 \leftarrow 1$ |
| 1 | | 00101 | 001? | Shift left AQ |
| | | 00010 | 001? | A = A+(-M) |
| | | 00010 | 0011 | $Q_0 \leftarrow 1$ |
| 0 | | | | |

∵ A[n] = 0

∴ Q (quotient) = 0011 (3)
A (Remainder) = (00010) (2)

$11 \div 4$

$11 \to 1011 \qquad 4 \to 100$

| n | M | A | Q | Action |
|---|---|---|---|---|
| 4 | 00100 | 00000 | 1011 | Initialization |
| | | 00001 | 011? | SL AQ |
| | | 11101 | 011? | A = A+(-M) |
| | | 11101 | 0110 | $Q_0 \leftarrow 0$ |
| 3 | | 11010 | 110? | SL AQ |
| | | 11110 | 110? | A = A+M |
| | | 11110 | 1100 | $Q_0 \leftarrow 0$ |
| 2 | | 11101 | 100? | SL AQ |
| | | 00001 | 100? | A = A+M |
| | | 00001 | 1001 | $Q_0 \leftarrow 1$ |
| 1 | | 00011 | 001? | SL AQ |
| | | 11111 | 001? | A = A+(-M) |
| | | 11111 | 0010 | $Q_0 \leftarrow 0$ |
| 0 | | | | |

$\because A[n] = 1 \quad \therefore A = A+M$

00011 0010

$\therefore$ A = 00011 (3) (Remainder)

Q = 0010 (2) (Quotient)

$1000 \div 11$

| n | M | A | Q | Action |
|---|---|---|---|---|
| 4 | 00011 | 00000 | 1000 | Initialization |
| | | 00001 | 000? | SL AQ |
| | | 11110 | 000? | A = A+(-M) |
| | | 11110 | 0000 | $Q_0 \leftarrow 0$ |
| 3 | | 11100 | 000? | SL AQ |
| | | 11111 | 000? | A = A+M |
| | | 11111 | 0000 | $Q_0 \leftarrow 0$ |
| 2 | | 11110 | 000? | SL AQ |
| | | 00001 | 000? | A = A+M |
| | | 00001 | 0001 | $Q_0 \leftarrow 1$ |
| 1 | | 00010 | 001? | SL AQ |
| | | 11111 | 001? | A = A+(-M) |
| | | 11111 | 0010 | $Q_0 \leftarrow 0$ |
| 0 | | | | |

$\because A[n] = 1 \quad , A = A+M$

00010 0010

$\therefore$ Q = 0010 (2) (Quotient)

A = 00010 (2) (Remainder)

Q. Suggest a method to solve integer division $12 \div 4$. (any method can be used)

Q. Suggest a suitable method to encode Multiplier ① 11011 (use booth algorithm / bit scanning)

↙ ② -24

Don't use sign bit binary number is given ③ Have two's complement and add a sign bit ④ then encode