Problem: Obtain the ~~logic~~ truth table directly from the logic (H.W) diagram without going through the derivation of Boolean functions.

## Sequential CIRCUITS (flip-flops)

→ Most interesting digital circuits also include storage elements (dipped in boxes full of pixie dust), which require that the system be described in terms of sequential circuits.

Digital circuits → Combinational ckts.
→ Sequential ckts.

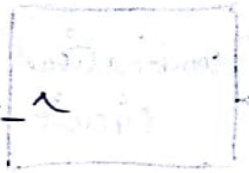
→ The most common type of sequential ckt. is the synchronous type.

⊛ Synchronous sequential circuits employ signals that affect the storage elements only at discrete instants of time.

- Synchronization is achieved by a ~~ter~~ timing device called a clock pulse generator that produces a periodic train of CLOCK PULSES.
- The clock pulses are distributed throughout the system in such a way that storage elements are affected only with the arrival of SYNCHRONIZATION PULSES.

~~Clocke~~ Clocked synchronous sequential circuits the type most frequently encountered in ~~paradi~~ practise

## Clocked (Syn) Sequential Circuits

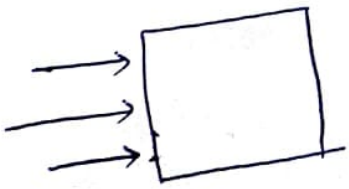


→ The storage elements employed in clocked sequential circuits are called flip-flops.

→ A flip flop is a binary cell capable of storing ONE BIT of information. It has two outputs, one for the normal value and the other for complement value of bit stored in it.

~~1~~ (P) A flip flop maintains a binary state until directed by a clock pulse to switch states.

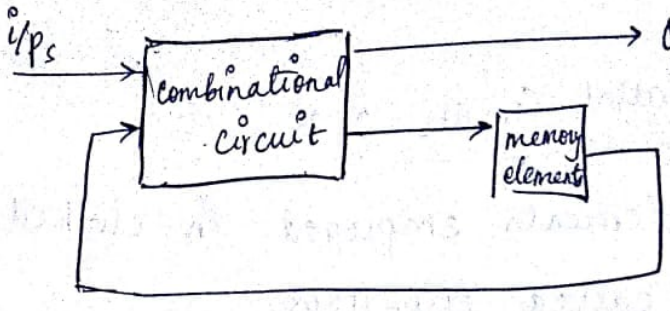
~~2~~ (R) The difference among various types of FLIP-FLOPS is in the number of inputs they possess. and in the manner in which the inputs affect the binary state.



→ Half adder is like 2 i/ps and Full adder be like more i/ps.

P 28/01/19

## Block diagram of a sequential circuit.




- Binary info. stored in Memory elements : STATE of the Seq. ckt.
- Inputs, present state also determine the condition for changing the state in memory element.
- The next state of memory element is also a function of external inputs & present state.
- Sequen. ckt is specified by a time sequence of inputs, outputs and internal states.

Sequential ckts → Synchronous
Sequential ckts → Asynchronous.

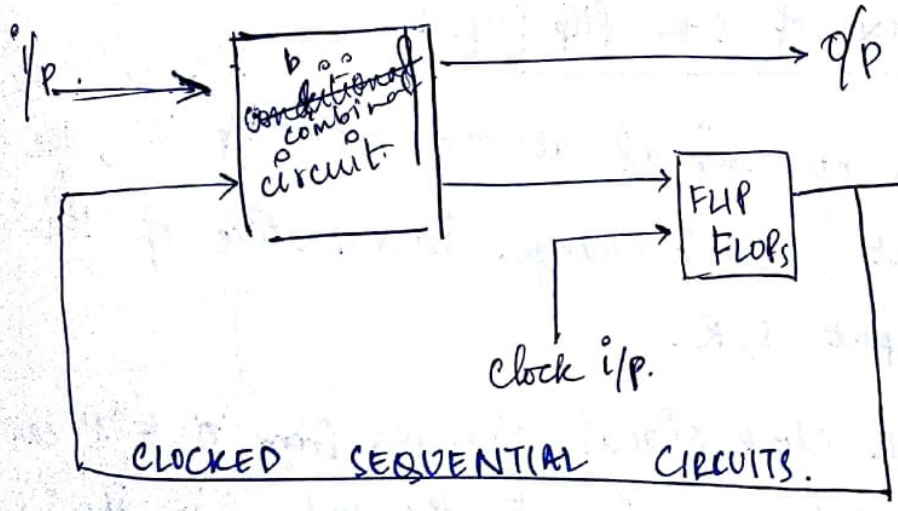
Asynchronous: Behaviour depends upon the order in which its input signals change and CAN BE AFFECTED at any time.

memory elements: Time-delay devices.

memory capability is due to the propogation time through the device.

- In gate type asynchronous systems.

→ Synchronous clocked sequential circuit.

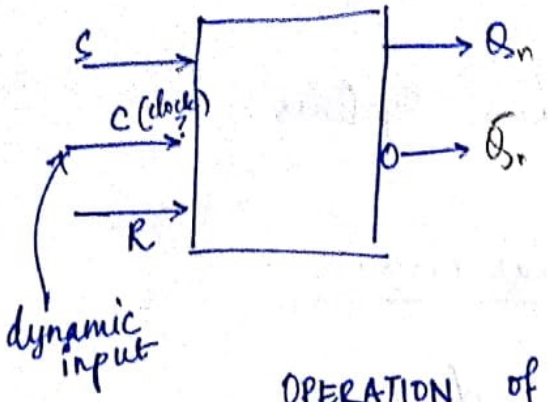



- Synchronization is achieved by using a ~~master~~ timing device called MASTER-CLOCK GENERATOR, which generates a periodic train of CLOCK PULSES.

→ ✱ clock pulses are distributed throughout the system in such a way that memory elements are affected only with the arrival of synchronization pulse.

single bit storage devices.

## S-R Flip Flop (SET-RESET Flip Flop):




The flip-flop responds to a positive transition (from 0 to 1) of the i/p clock signal.

## OPERATION of S-R flip-flops:

→ If there is no signal at the clock i/p 'C', the o/p of the circuit cannot change irrespective of the values at input S, R.

→ Only when the clock signal changes from '0' to '1' can the o/p be affected according to the values in i/p S & R

→ If S=1, R=0 ⟶ Q=1

Clock has to be in (1) to trigger the flipflop

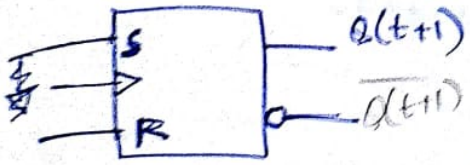
S=0, R=1 ⟶ Q=0 → Reset condition

S=0, R=0 ⟶ Q doesn't change. → same as in prev. state

S=1, R=1 ⟶ Q indeterministic / unpredictable.

[because $Q=\bar{Q}$. which is impossible →]

Q(t) may go either '0' or 1, depending on internal timing delays that occur within the




$Q_{n+1} = S + Q_n \bar{R}$

taking $Q_n$ as a variable (case 1) keeps '0' case

$Q_n(t)$: Binary state of the Q output at a given time (called PRESENT STATE)

$Q_n(t+1)$: Binary state of the Q o/p after the occurence of a clock transition.

1) **D- Flip Flop**: An S-R flipflop is converted to a D-flip flop by inserting an inverter b/w S and R. and assigning the symbol D to the single i/p.

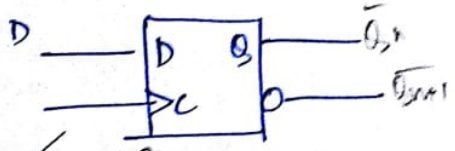
effectively there'll be a single i/p. but the state will never be retained.

→ D i/p is sampled during the occurence of clock transition from 0 to 1.

If D='1', the o/p, Q goes to '1'.

D='0', " '0'

$Q_{n+1} = D$




we're assuming the clock is always on.

| D | Q(t+1) |
|---|---|
| 0 | 0 ... clear to '0' → reset. |
| 1 | 1 ... set to '1' Set |

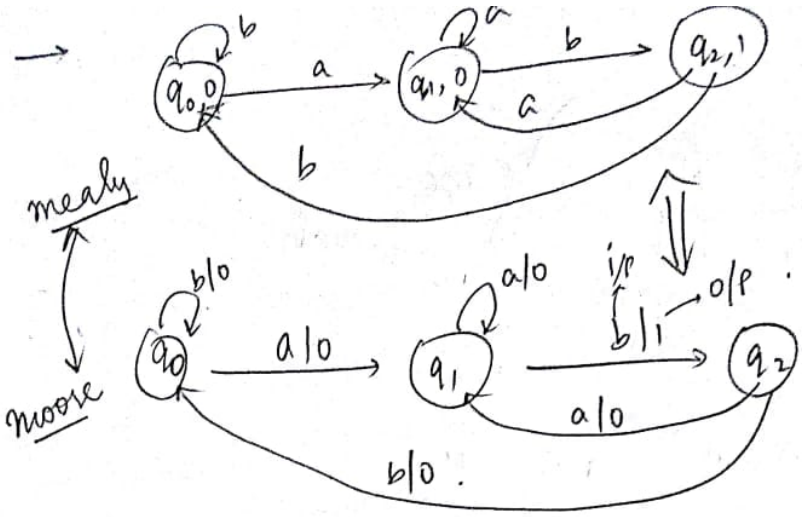
Characteristic eq$^n$: $Q_n(t+1) = D$.

NO CHANGE condition doesn't exist.

~~"No change" condition can be accomp~~ , Ex: D-1 Ex-2

→ The difference among various types of FLIP FLOPs is in the no. of inputs they possess and the manner in which the inputs affect the final o/p? I guess.

→ "No change" condition can be accomplished either by disabling the clock signal or feeding the o/p back to the i/p so that clock pulses keep the




prints 1 as o/p every occurence of as substring

30/01/19

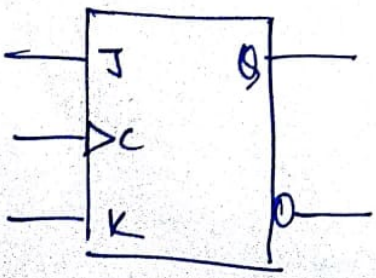
## JK FLIP FLOP / Refinement of SR Flip Flop

In SR F.F : There is a drawback, an indeterminate condition. It will be defined, in a JK F.F.

To fix the 1-1 case of SR →

→ Inputs J and K behave like i/p s 'S' and 'R' to set and clear the Flip Flop respectively.

→ When inputs J and K are both equal to '1', a clock transition (from '0' to '1') switches the outputs of the Flip Flop to their complement stat




| J | K | Q(t+1) |
|---|---|---|
| 0 | 0 | Q(t) - No change |
| 0 | 1 | 0 - clear to '0' |
| 1 | 0 | 1 - set to '1' |
| 1 | 1 | Q'(t) - complement |

pre state

of the pre. state

→ As the IC technology improved, no. of gates that could be put into integrated circuit increased.

SSI → small scale integration. → # of gates < 10.

MSI → medium " . → # → 10 < # < 200 gates.

↳ systems → generally used as decoders.

LSI → 200 < to few thousand.

↳ digital systems such as ~~programmers~~ processors, memory chips and programmable modules.

DIGITAL LOGIC FAMILIES

TTL → transistor-transistor logic

ECL → emitter-coupled logic

MOS → metal-oxide semiconductor.

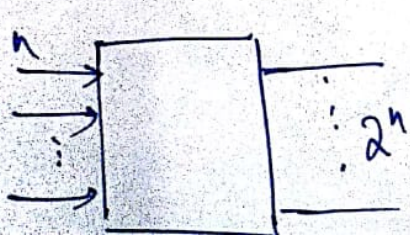
CMOS → complementary MOS.

DECODERS:

Discrete Quantities of Info: Binary Codes.

Binary code of n bits: Represents $2^n$ distinct element of coded informal.

→ A decoder is a COMBINATIONAL CIRCUIT that converts binary information from the 'n' coded input to a maximum of $2^n$ unique outputs.



Decoders are specified as

# i/p → n, # o/p → m → $2^n$, where $m \leq 2^n$.

n (decimal) (i/p) BCD. $2^n$ (o/p)

| E | $A_2$ | $A_1$ | $A_0$ | $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | x | x | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

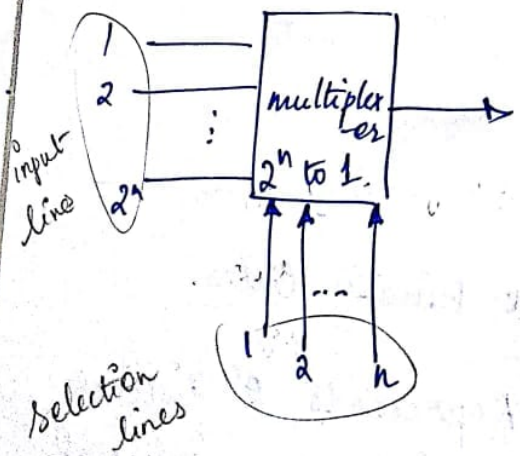
(E) ~~encoded~~ ~~it seems. wtf.~~

enable → 0 → decoder disabled, o/p = 0 irrespective of i/p.
→ 1 → enabled.

(D) ~~correspond~~ to octal digits. nice.

(A) → Binary.

MULTIPLEXERS : Combinational circuit.




It receives information from one of the $2^n$ input data lines and directs it to a single output line. The selection of particular input data line for the outputs is determined by a set of selection lines.

→ ~~4 to 1 Multiplexer~~

Multiplexer is like the opp. of decoder.

# 4- to -1 MULTIPLEXER

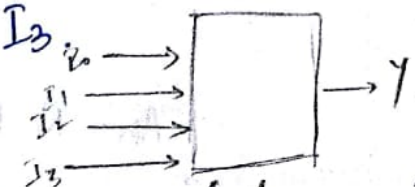
i/p → 4 (n=2, $2^n$) → o/p

The truth table would be having an entire 64 entries, but we ain't gonna draw that. These are just all combinations.

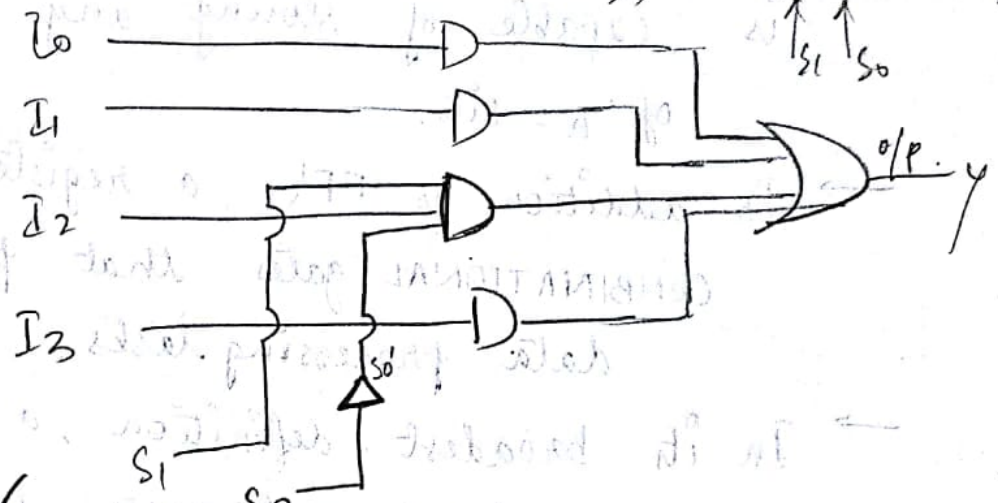
n selection lines are usually taken for $2^n$ inputs

Function Table

| $S_1$ | $S_0$ | Y |
|---|---|---|
| 0 | 0 | $I_0$ |
| 0 | 1 | $I_1$ |
| 1 | 0 | $I_2$ |
| 1 | 1 | $I_3$ |



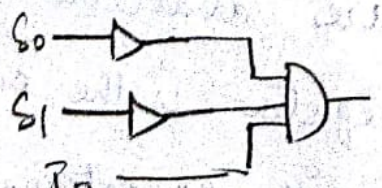

→ $S_1 S_0 = 10$.



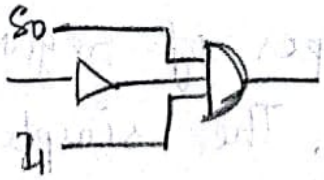

For ~~something~~ like 11,

Here, we're just sending '10' to $I_2$, ~~&~~ ~~we sent them all to all I's then it's perfect~~ like complete

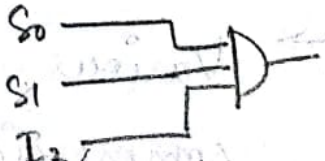
'00' → $I_0$




'01'




'11'




Sabka, OR gate lagaao

$$Y = I_0 S_0' S_1' + I_1 S_1' S_0 + I_2 S_0' S_1 + I_3 S_1 S_0$$

DECODER → takes

ENCODERS → opp. of that.

01/02/19

REGISTERS → Connection of For ex. 'N' flip-flops, Each FF stores 1 bit of informati

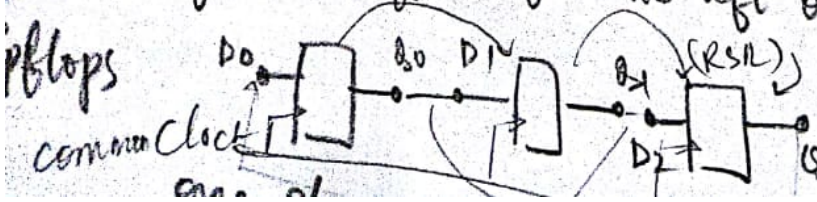
An N-bit register has a group of n-FF's. and is capable of storing any binary information of n-bits.

→ In addition to FF's, a register may have COMBINATIONAL gates that perform certain data processing tasks.

→ In its broadest definition, a register consists of a group of FF's and gates that effect their transition. The FF's hold the binary informat and gates control when and how new informati is transferred to the register.

→ Various types of register are available commercially. The simplest register is the one that consists only of FFs, with no external gate

Shif register → shifts info. to left or right. (LSR, RSR)




They all have common

DECODER → takes n inputs → converts it into 2^n o/p.

ENCODERS → opp. of that.

---

01/02/19

REGISTERS → Connection of For ex. 'N' flip-flops. Each FF stores 1 bit of informati

An N-bit register has a group of n-FF's. and is capable of storing any binary information of n-bits.

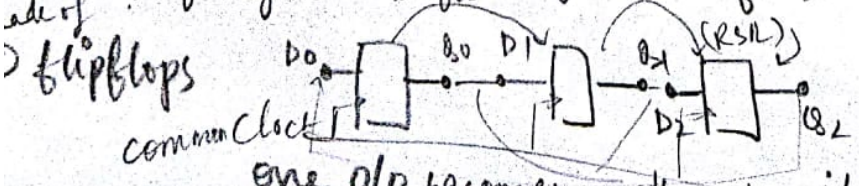
→ In addition to FF's, a register may have COMBINATIONAL gates that perform certain data processing tasks.

→ In its broadest definition, a register consists of a group of FF's and gates that effect their transition. The FF's hold the binary informati and gates control when and how new informati is transferred to the register.

→ Various types of register are available commercially. The simplest register is the one that consists only of FFs, with no external gate

---

Shif register → shifts info. to left or right. (LSR, RSR)




They all have common clock pulse.

## Shift registers:

f/f → 1 bit of info.
n bits → n f/f → registers.

→ A register capable of shifting its binary information in one or both directions is called a SHIFT REGISTER

→ The logical configuration of a shift register consists of a chain of FF's in ~~cascade~~ CASCADE with o/p of one FF connected to i/p of ~~other~~ next FF.

→ Common clock i/p is given to all of them.

LFSR → linear feedback shift register.

~~free conceiving~~

Frequency reuse

Service provider → should make max utilization with the given set of channels.

Time axis → divide into slots — each user given a slot

when on phone → next frame you get another slot — we keep getting disconnected every ms — we don't even notice

TDNA, FDNA → kya hai.

Q. Can a channel be used simultaneously by multiple users?

→ frequency division multiplexing

Signature sequences are generated by LFSR → specific to each

Synchronous → clock predefined, i/p not

asynchronous → clock not, i/p ✓

## Binary Counter:

A register that goes through a pre-determined sequence of states upon the application of ~~clock~~ input pulses (~~clock is asap~~ could be or) is called COUNTER

→ The input pulses may be clock pulses or may originate from an external source. They may occur at uniform intervals of time or at random.

→ Counters are found in almost all equipment containing digital logic. They are used for counting the # of occurences of an EVENT and are useful for generating timing signals to control the sequence of operations in digital computers.

→ Of the various sequences a counter may follow, the straight/uniform binary sequence is the simplest and most straightforward. A counter that follows the binary ~~nos~~ numbers sequence is called a binary counter.

→ An n-bit binary counter is a REGISTER of n-flip-flops and associated gates that follows a sequence of states according to the binary count of n-bits from '0' to '2^n-1'

MEMORY UNIT — most crucial element.

- A memory unit is a collection of storage cells,

cash memories, registers } very expensive ; RAM and all not so costly bcz

Wanna assess the information?

You need a reader & write to control the signals together with associated circuits needed to transfer information in and out of storage.

- Memory stores binary information in groups of bits called WORDS. A ~~word~~ WORD in memory is an ~~event~~ ENTITY of bits that move in and out of storage as a ~~unit~~ UNIT.

- A memory word is a group of 1's and 0's and may represent a NUMBER, an INSTRUCTION CODE, ONE or more ALPHANUMERIC CHARACTERS, or any other BINARY CODED INFORMATION.

- A group of 8 bits is called a BYTE, most computer memories use words whose number of bits is a multiple of 8. Thus, a 16-bit word contains 2 bytes, and a 32-bit word is made up of 4 bytes.

$F = \Sigma(1, 2, 3, 6, 7)$

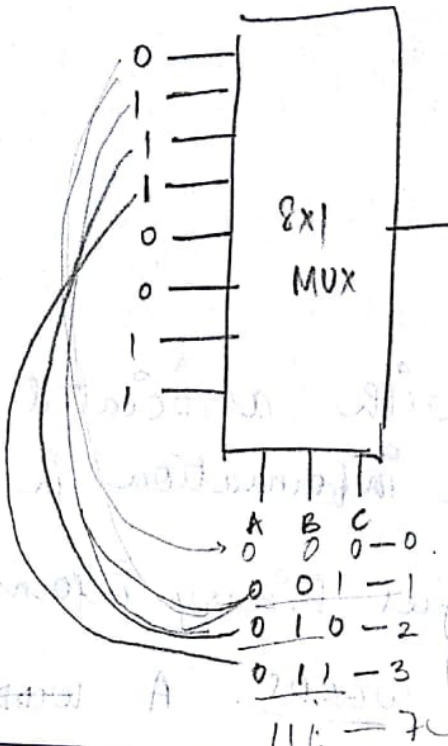
using multiplexer.

8-1 multiplexer.

| A | B | C | | |
|---|---|---|---|---|
| 0 | 0 | 0 | $m_0$ | 0 |
| 0 | 0 | 1 | $m_1$ | 1 |
| 0 | 1 | 0 | $m_2$ | 1 |
| 0 | 1 | 1 | $m_3$ | 1 |
| 1 | 0 | 0 | $m_4$ | 0 |
| 1 | 0 | 1 | $m_5$ | 0 |
| 1 | 1 | 0 | $m_6$ | 1 |
| 1 | 1 | 1 | $m_7$ | 1 |

$f \rightarrow m_1 + m_2 + m_3 + m_6 + m_7$

now for multiplexer,




4/2/19

## Memory unit:

→ WORDS ⟶ Basic storage / retrieval unit in memory.

8 bits ⟶ 1 byte.

Words ⟶ multiple of bytes.

→ CAPACITY OF MEMORY: Stated as the total # of bytes that can be stored.

→ Internal structure of a Memory unit:

Specified by the # of words it consists and the no. of bits in each word.

→ Special INPUT lines called ADDRESS lines to select a particular word.

→ Each word in memory is assigned an identification number, called address, starting from '0' and continuing with 1, 2, 3, ... $2^k - 1$, where k is the # of address lines.

→ The selection of SPECIFIC WORD inside the memory is done by applying the k-bit binary address to the address lines.

→ A decoder inside the memory accepts this address and OPENS THE PATHS needed to select the bits of a specific word.

→ Computer memories may range from 1024 words required requiring an address of 10 bits, to $2^{32}$ words requiring 32 address bits.

→ Types of Memories : 2 types
- Random Access Memory (RAM)
- Read Only Memory (ROM).

Semiconductor Memories
- RAMs.
- Sequence Access Memories (SAMs)

→ Memories constructed with SHIFT REGISTERS, CHARGE COUPLED DEVICES (CCDs) or BUBBLE Memories ... SAMs.

RAM = { ROMs, Read Mostly memories (RMMs)
Read write memories (RWMs).

ROMs → Masked Programme ROMs.
→ User Programmed ROMs.

RMMs → Erasable and Programmable (EPR
→ Electronically Erasable (EER
using UV rays

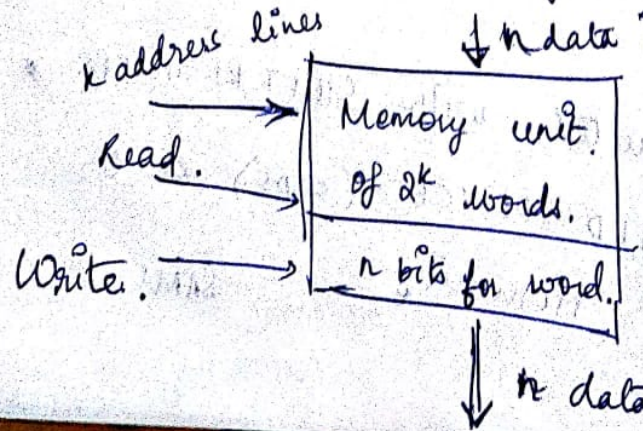
RWMs → Static RAMs — memory cells are flipflops.
→ Dynamic RAMs — memory cells that must be refreshed, re and return periodically to avoid loss of memory cells.

Ex: Flash memories - used in phones, cameras for pictures.

## RANDOM ACCESS MEMORY (RAM)

~~→~~ The amount of time taken to locate any word is same, irrespective of where it's physically present.

→ Communication between a memory and its environment is achieved through data input lines, data o/p lines, address selection lines and control lines. (specify the direction of transfer)



2 operations that a RAM can perform: Write and Read.

Write signal:

① You apply the binary address of the desired word into the address lines.

② Apply the data bits that must be stored in memory into the data input lines.

③ Activate the WRITE i/p.

→ To transfer the stored word out of the address (READ)

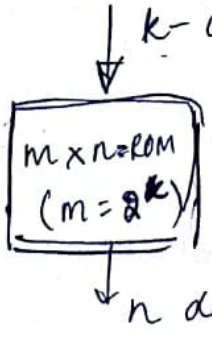
① Apply the binary address of the desired word into the address line.

② Activate the READ i/p.

③ RAM → reads words that're permanently stored in the mem

**m x n ROM**

ROM as a block is a combinational ckt.



22/01/19

Tutorial

# Binary codes

→ suitable for comp. applications & digital communication.

Attenuation
Distortion
Noise
} affect the analog signal, but they do not affect the digital signals that much.
So, we prefer digital. (DTH set top box something)

Binary
- weighted codes
- unweighted codes.

BCD → binary coded decimal
→ Every digit (0-9) substituted with a binary 4-bit string equivalent (0000 - 1001)

(1) 759 + 138 → BCD mode.

0111 0101 0100
+ 0001 0011 1000
1000 1000 1100
8 8 12 → not acceptable in BCD.

Whenever it exceeds 9, add binary 6 to it.

0110

1000 1001 0010

now if this was greater than 9, if too add 0110 to it.

For the last one, ignore carry.

⇒ 1100
0110
carry to next one ← 1 0010

BCD arithmetic is little complicated than binary arithmetic.

Excess-3 code (XS3).

- unweighted code (no positional weights)
- obtained by adding binary 3 to the BCD code. (0011)

| Decimal | BCD | Excess-3 |
|---|---|---|
| 0 | 0000 | 0011 |
| 1 | 0001 | 0100 |
| 2 | 0010 | 0101 |
| : | : | : |
| 8 | 1000 | 1011 |
| 9 | 1001 | 1100 |

self complementing code.

Gray code:

→ It is an unweighted case

→ Hamming distance always 1. (unit)

↓

also called unit distance codes.

| Decimal | BCD | Gray |
|---|---|---|
| 0 | 0000 | 0000 |
| 1 | 0001 | 0001 |
| 2 | 0010 | 0011 |
| : | : | : |
| 8 | 1000 | 1100 |
| 9 | 1001 | 1101 |

→ First bit written as it is and 2nd bit ka XOR gate is written at 2nd position. 2nd, 3rd → 2, 3 → 3, ... like that till the end.

not same for all.

Hamming distance

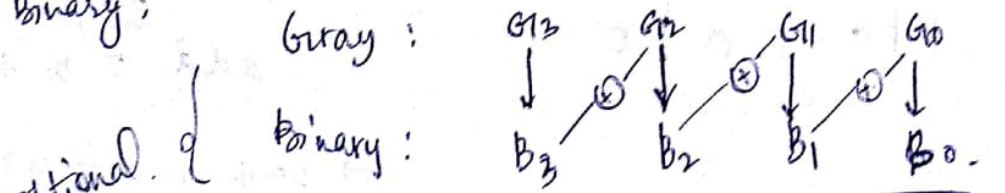
The corresponding no of bit difference.

⇒ Binary to Gray:

Binary: $B_3$ $B_2$ $B_1$ $B_0$

Gray: $G_3$ $G_2$ $G_1$ $G_0$.

Gray to Binary:




optional.

---

## Truth Table for Adder circuit.

| x | y | z |
|---|---|---|

---

23/01/19

## Subtractors:

$x \rightarrow$ minuend

$x - y = z$. $y \rightarrow$ subtrahend.

① 2's complement of 'y' and add binary to 'x'.

② If minuend is smaller than the subtrahend bit, a 1 is borrowed from the next significant position.

⇒ The fact that a 1 has been borrowed must be conveyed to the next higher pair of bits by means of a binary signal coming out of a given stage and going into (input) the next higher stage.

~~passion, zeal~~

Stack of the flip flop unchanged.

29/1/19 → TUTORIAL

RAM designed using registers

for doing multiple tasks simultaneously.

One processor can't do it all, it gets hung.

→ So we have multi core processor

to execute multiple processes at the same time.

DATA

Algorithms

Time complexity → Processors

Space complexity

$i_5$ → 4 processors

each (2.57, or 2.62)

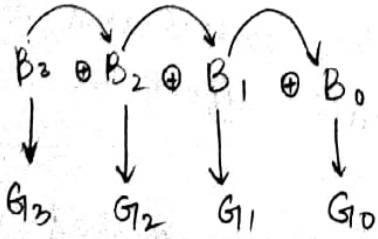
tells the speed

$i_7$ → 6 different processors

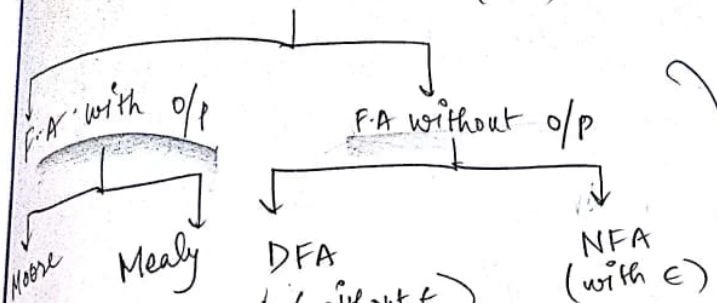
→ 64 bit registers → temp. storage of memory in the register in the processor

If it's in the register, then it won't be in the main memory.

→ All the i/ps are stored in the FLIP FLOPS.




Finite Automata (FA)

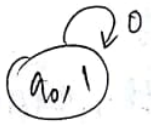



$M(\Sigma, Q, \delta, q_0, F)$

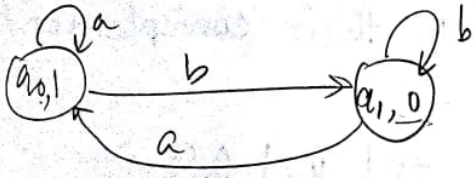
- $\Sigma$ → language
- $Q$ → set of states
- $\delta$ → transition function
- $q_0$ → initial state
- $F$ → final state
- $\lambda$ → o/p function ($\lambda$)
- $\Delta$ → o/p Alphabet. ($\Delta$)

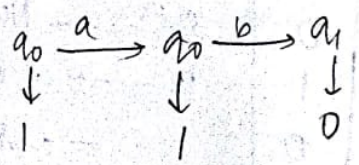
## Moore Machine

→ o/p is associated with the state




$\lambda : Q \rightarrow \Delta$

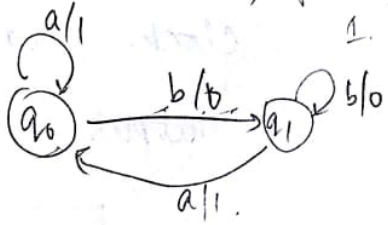


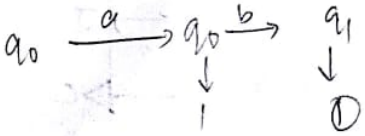




## Mealy Machine

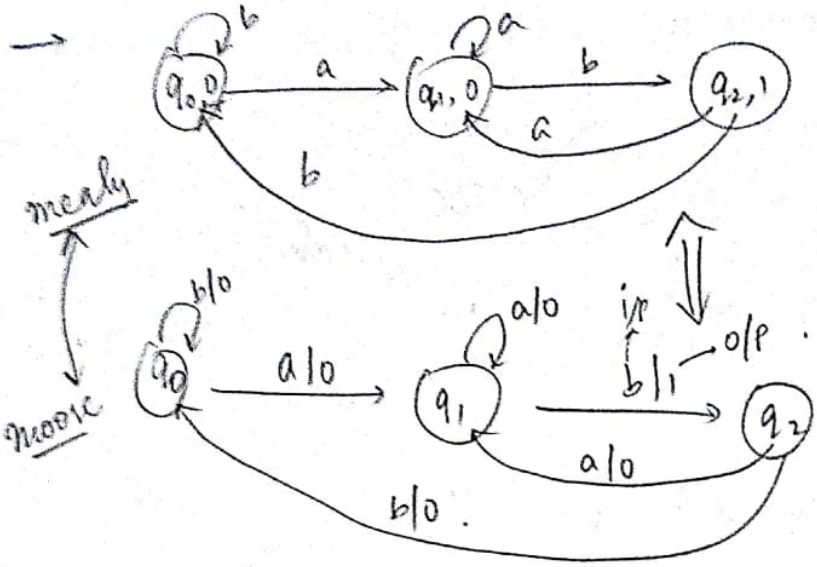
o/p is associated with i/p :


$q_0$ on 0, is producing 1.




process




Input is 'a', 'b' and prints 1 as o/p on every occurence of 'a' as substring

30/01/12

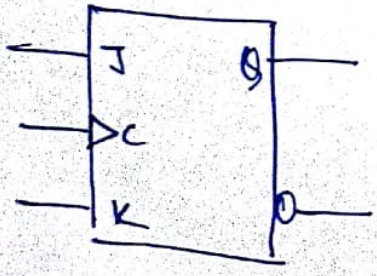
## JK FLIP FLOP / Refinement of SR Flip Flop

In SR F.F : There is a drawback, an indeterminate condition. It will be defined, in a JK F.F.

To fix the 1-1 case of SR →

→ Inputs J and K behave like i/p s 'S' and 'R' to set and clear the Flip Flop respectively.

→ When inputs J and K are both equal to '1', a clock transition (from '0' to '1') switches the outputs of the Flip Flop to their complement state




| J | K | Q(t+1) |
|---|---|---|
| 0 | 0 | Q(t) - No change |
| 0 | 1 | 0 - clear to '0' |
| 1 | 0 | 1 - set to '1' |
| 1 | 1 | Q'(t) - complement of the present state |

→ present state

→ As the IC technology improved, no. of gates that could be put into integrated circuit increased.

SSI → small scale integration. → # of gates < 10.

MSI → medium " . → # → $10 \leq$ # $< 200$ gates.

↳ systems → generally used as decoders.

LSI → 200 < to few thousand.

↳ digital systems such as ~~programmess~~ processors, memory chips and programmable modules.

DIGITAL LOGIC FAMILIES

TTL → transistor-transistor logic

ECL → emitter-coupled logic

MOS → metal-oxide semiconductor.

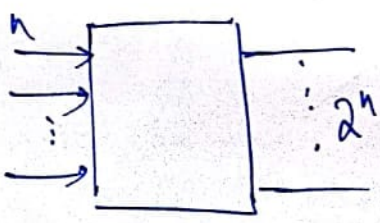
CMOS → complementary MOS.

DECODERS:

Discrete Quantities of Info: Binary Codes.

Binary code of n bits: Represents $2^n$ distinct element of coded informal.

→ A decoder is a COMBINATIONAL CIRCUIT. that converts binary information from the 'n' coded input to a maximum of $2^n$ unique outputs.



Decoders are specified as n to m line decoders, where $m \leq 2^n$.

n → m
# o/p → $2^n$
n (decimal) (i/p) BCD.
$2^n$ (o/p)

| E | $A_2$ | $A_1$ | $A_0$ | $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | x | x | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| : | : | : | : | : | : | : | : | : | : | : | : |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

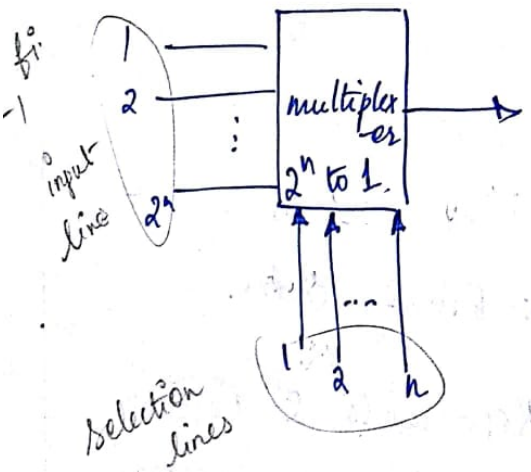
(E) ~~encoded~~ ~~it seems. wtf.~~

enable → 0 → decoder disabled, o/p = 0 irrespective of i/p.
→ 1 → enabled.

(D) correspond to octal digits nice.

(A) → Binary.

MULTIPLEXERS : combinational circuit.




It receives information from one the $2^n$ input data lines and directs it to a single output The selection of particular inp data line for the outputs is determined by a set of selecti line

→ ~~4 to 1 Multiplexer~~

Multiplexer is like the opp. of decoder

# 4- to -1 MULTIPLEXER

i/p → 2^n, n=2

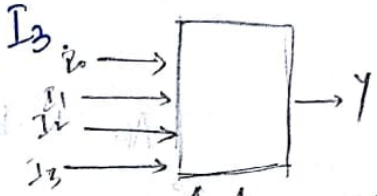
o/p

n selection lines are usually taken for $2^n$ inputs

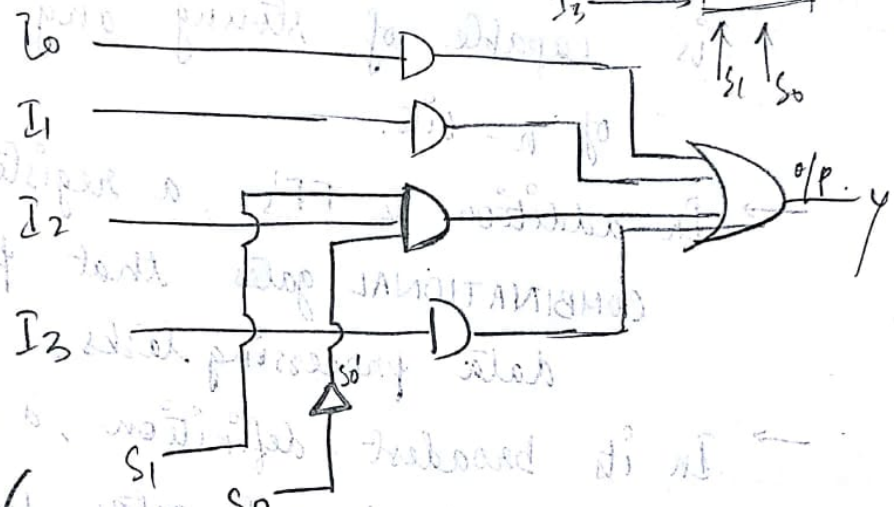
The truth table would be having an entire 64 entries, but we aint gonna draw that. These are just a few combinations.

## Function Table

| $S_1$ | $S_0$ | Y |
|---|---|---|
| 0 | 0 | $I_0$ |
| 0 | 1 | $I_1$ |
| 1 | 0 | $I_2$ |
| 1 | 1 | $I_3$ |



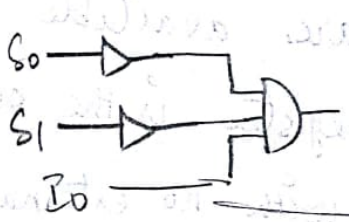

→ $S_1 S_0 = 10$.



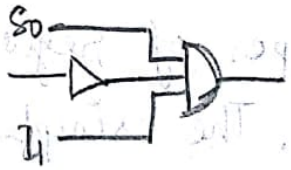

For ~~something~~ like 11,

Here, we're just sending '10' to $I_2$, ~~&~~ we ~~sent them all to~~ to all I's then it's perfect ~~like complete~~.

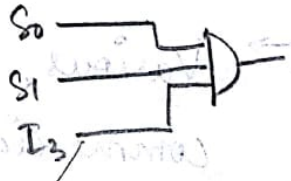
'00' → $I_0$




'01'




'11'




Sabka, OR gate lagaao

$$Y = I_0 S_0' S_1' + I_1 S_1' S_0 + I_2 S_0' S_1 + I_3 S_1 S_0$$

DECODER → takes n inputs → converts it into $2^n$ o/ps.

ENCODERS → opp. of that.

---

01/02/19

REGISTERS → Connection of For ex. 'N' flip-flops, Each FF stores 1 bit of information.

An N-bit register has a group of n-FF's. and is capable of storing any binary information of n-bits.

→ In addition to FF's, a register may have COMBINATIONAL gates that perform certain data processing tasks.

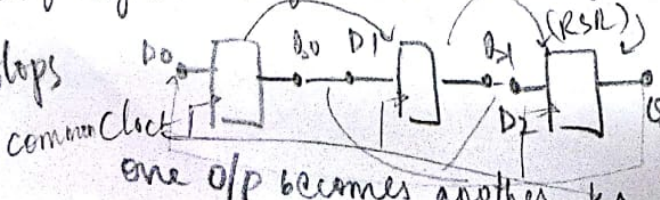
→ In its broadest definition, a register consists of a group of FF's and gates that effect their transition. The FF's hold the binary information and gates control when and how new information is transferred to the register.

→ Various types of register are available commercially. The simplest register is the one that consists only of FFs, with no external gates.

---

Shift register → shifts info. to left or right (LSR, RSR)

made of D flipflops




They all have common clock pulse.

## Shift registers:

f/f → 1 bit of info.
n bits → n f/f → registers.

→ A register capable of shifting its binary information in one or both directions is called a SHIFT REGISTER

→ The logical configuration of a shift register consists of a chain of FF's in ~~cascade~~ CASCADE with o/p of one FF connected to i/p of ~~other~~ next FF.

→ Common Clock i/p is given to all of them.

LFSR → linear feedback shift register.

~~Free Controlling~~

Frequency reuse

Service provider → should make max utilization with the given set of channels.

Time axis → divide into slots — each user given a slot

when on phone → next frame you get another slot — we keep getting disconnected every ms — we don't even notice

TDMA, FDMA → kya hai.

Q. Can a channel be used simultaneously by multiple users?

frequency division multiplexing.

Signature sequences are generated by LFSR → specific to each

Synchronous → clock predefined, i/p not

asynchronous → clock not, i/p ✓

## Binary Counter:

A register that goes through a pre-determined sequence of states upon the application of ~~clock~~ input pulses (~~clock ka case~~ could be) is called COUNTER.

→ The input pulses may be clock pulses or may originate from an external source. They may occur at uniform intervals of time or at random.

→ Counters are found in almost all equipment containing digital logic. They are used for counting the # of occurences of an EVENT and are useful for generating timing signals to control the sequence of operations in digital computers.

→ Of the various sequences a counter may follow, the straight/uniform binary sequence is the simplest and most straightforward. A counter that follows the binary number sequence is called a binary counter.

→ An n-bit binary counter is a REGISTER of n-flip-flops and associated gates that follows a sequence of states according to the binary count of n-bits from '0' to '$2^n-1$'

MEMORY UNIT — most crucial element.

- A memory unit is a collection of storage cells,

cash memories, registers } very expensive ; RAM and all not so costly bcf.

Wanna assess the information?

You need a reader & writer to control the signals.

together with associated circuits needed to transfer information in and out of storage.

- Memory stores binary information in groups of bits called WORDS. A ~~word~~ WORD in memory is an ~~event~~ ENTITY of bits that move in and out of storage as a ~~part~~ UNIT.

- A memory word is a group of 1's and 0's and may represent a NUMBER, an INSTRUCTION CODE, ONE or more ALPHANUMERIC CHARACTERS, or any other BINARY CODED INFORMATION.

- A group of 8 bits is called a BYTE, most computer memories use words whose number of bits is a multiple of 8. Thus, a 16-bit word contains 2 bytes, and a 32-bit word is made up of 4 bytes.

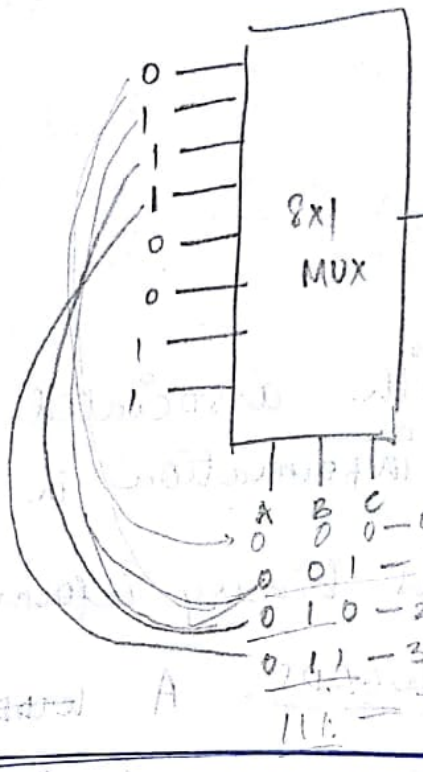
using multiplexer

8 - 1 multiplexer

| A | B | X | | |
|---|---|---|---|---|
| 0 | 0 | 0 | $m_0$ | 0 |
| 0 | 0 | 1 | $m_1$ | 1 |
| 0 | 1 | 0 | $m_2$ | 1 |
| 0 | 1 | 1 | $m_3$ | 1 |
| 1 | 0 | 0 | $m_4$ | 0 |
| 1 | 0 | 1 | $m_5$ | 0 |
| 1 | 1 | 0 | $m_6$ | 1 |
| 1 | 1 | 1 | $m_7$ | 1 |

$f \rightarrow m_1 + m_2 + m_3 + m_6 + m_7$

now for multiplexer,




4/2/19

## Memory unit:

→ WORDS → Basic storage / retrieval unit in memory.

8 bits → 1 byte.

Words → multiple of bytes.

→ CAPACITY OF MEMORY: Stated as the total # of bytes that can be stored.

→ Internal structure of a Memory unit:

Specified by the # of words it consists and the no. of bits in each word.

→ Special INPUT lines called ADDRESS lines to select a particular word.

→ Each word in memory is assigned an identification number, called address, starting from '0' and continuing with 1, 2, 3, ... $2^k - 1$, where k is the # of address lines.

→ The selection of SPECIFIC WORD inside the memory is done by applying the k-bit binary address to the address lines.

→ A decoder inside the memory accepts this address and OPENS THE PATHS needed to select the bits of a specific word.

→ Computer memories may range from 1024 words requiring an address of 10 bits, to $2^{32}$ words requiring 32 address bits.

→ Types of Memories : 2 types
- Random Access Memory (RAM)
- Read Only Memory (ROM).

Semiconductor Memories
- RAMs.
- Sequence Access Memories (SAMs)

→ Memories constructed with SHIFT REGISTERS,
CHARGE COUPLED DEVICES (CCDs) or
BUBBLE Memories ... SAMs.

RAM = { ROMs, Read Mostly memories (RMMs).
Read Write memories (RWMs).

ROMs → Masked Programme ROMs.
→ User Programmed ROMs.

RMMs → Erasable and Programmable (EPROM)
→ Electronically Erasable (EEROM)
using UV rays

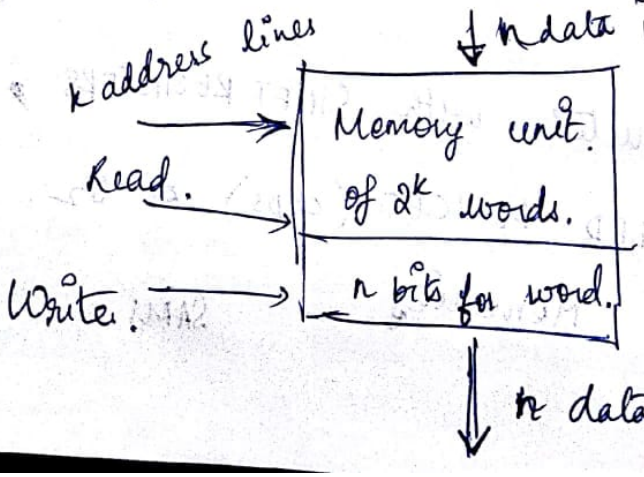
RWMs → Static RAMs — memory cells are flipflops.
→ Dynamic RAMs — memory cells that must be refreshed, read and return periodically to avoid loss of memory cells.

EX: Flash memories - used in phones, cameras for pictures.

## RANDOM ACCESS MEMORY (RAM)

~~The~~ The amount of time taken to locate any word is same, irrespective of where it's physically present.

→ Communication between a memory and its environment is achieved through data input lines, data o/p lines, address selection lines and control lines. (specify the direction of transfer)



2 operations that a RAM can perform: Write and Read.

Write signal:

① You apply the binary address of the desired word into the address lines.

② Apply the data bits that must be stored in memory into the data input lines.

③ Activate the WRITE i/p.

→ To transfer the stored word out of the address (READ)

① Apply the binary address of the desired word into the address line.

② Activate the READ i/p.

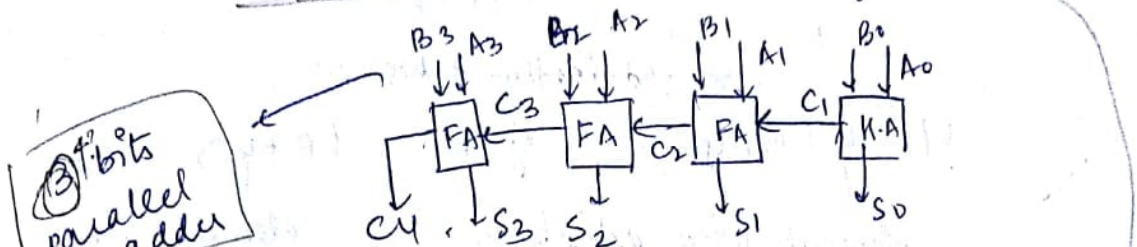
③ RAM → reads words that're permanently stored in the mem

**m x n ROM**

ROM as a block is a combinational ckt.

Tut 12/2/19.

→ How many half adders and full adders are needed for a 7-bit parallel binary adder?




(3) 4-bits parallel adder

So, there'll be 4 sums and 3 carrys.

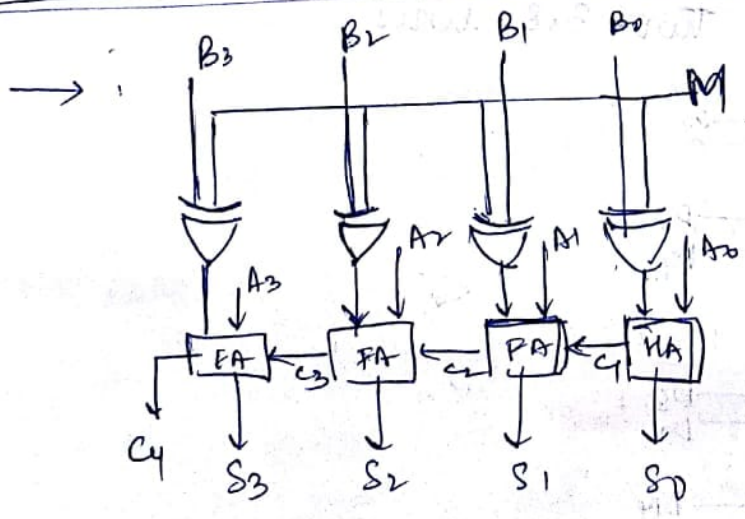
And has → 3 Full adders + 1 Half adder.

| A3 | A2 | A1 | A0 |
|---|---|---|---|
| B3 | B2 | B1 | B0 |

Ex: 1011
1101
11000

→ Here, 7-bit parallel binary adder

⇒ 6 F.A. and 1 H.A




When M=0

B ⊕ 0 = B

FA → A + B.

M=1 → B ⊕ 1 = B'

FA → A + B'

(b) So this is an adder-subtractor circuit

When M=0, it acts as an adder and when M=1, acts as a subtractor.

R's complement of A ⇒ R − A = B.

1's of 0 ⇒ $(1)^n$ (111)

= 000

$2^3$ ⇒ 222 − 111 = 111

~~So basically we~~

(R + k) complement of A ⇒ (B + k)

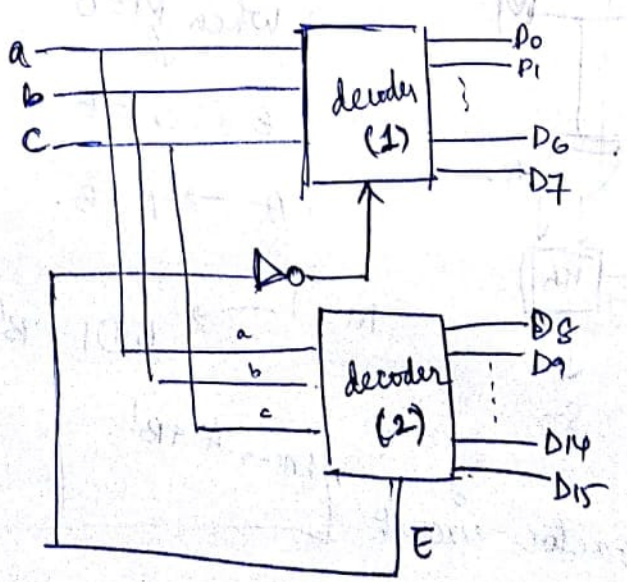
element wise addition — elementwise addition

→ Decoder:

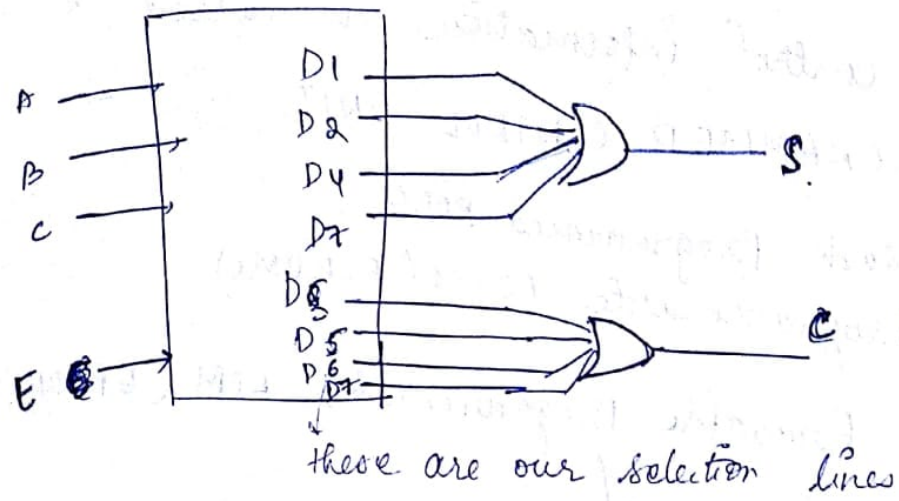
| E | $B_2$ | $B_1$ | $B_0$ | $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | x | x | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | | | | | | | | | | | |
| 1 | | | | | | | | | | | |

4×16 ⟶ not possible to design in real.

So, we use two 3×8 lines.



Q:→ Re Design the F.A from 3x8 decoder.




11/02/19

→ ROM is constructed externally with decoders and a set of OR gates.

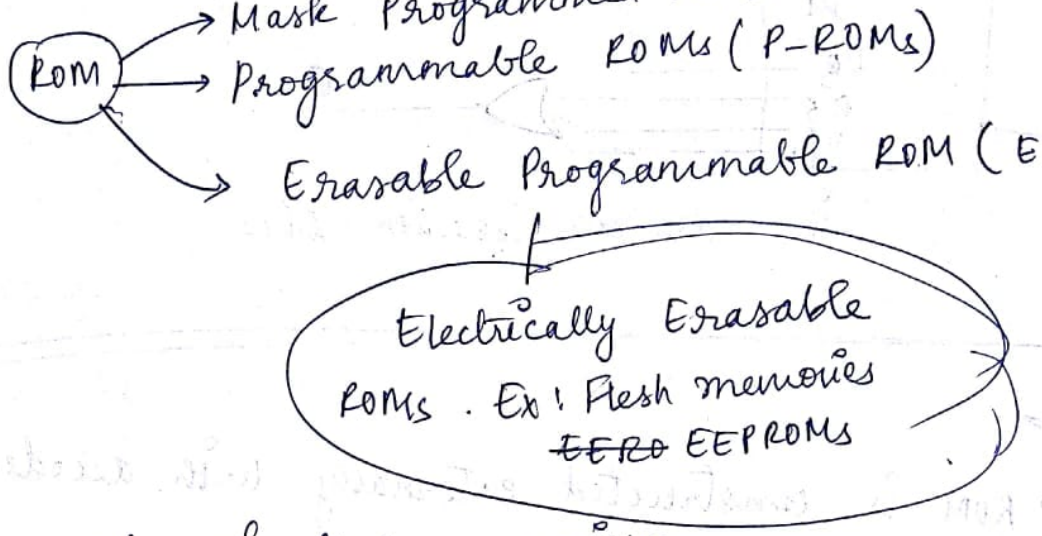
→ ROM generates an I/P – O/P relationship specified by a truth table. It can implement any combinational circuit with 'K' inputs and 'n' O/Ps.

---

→ When employed (in CS) as a memory unit, the ROM is used for storing fixed programs that are not to be altered and for TABLE of constants that are not subjected to change.

→ ROM is employed in the design of control units for digital computers. As such, they are used to store coded info. that represents the sequence of internal control variables needed for enabling various operations in computers.

→ A COMPUTER UNIT thats utilizes a ROM to store binary control information is called a MICROPROGRAMMED CONTROL UNIT.

→ 


Examples of flesh memories:
① Storage Messages in a Mobile phone.

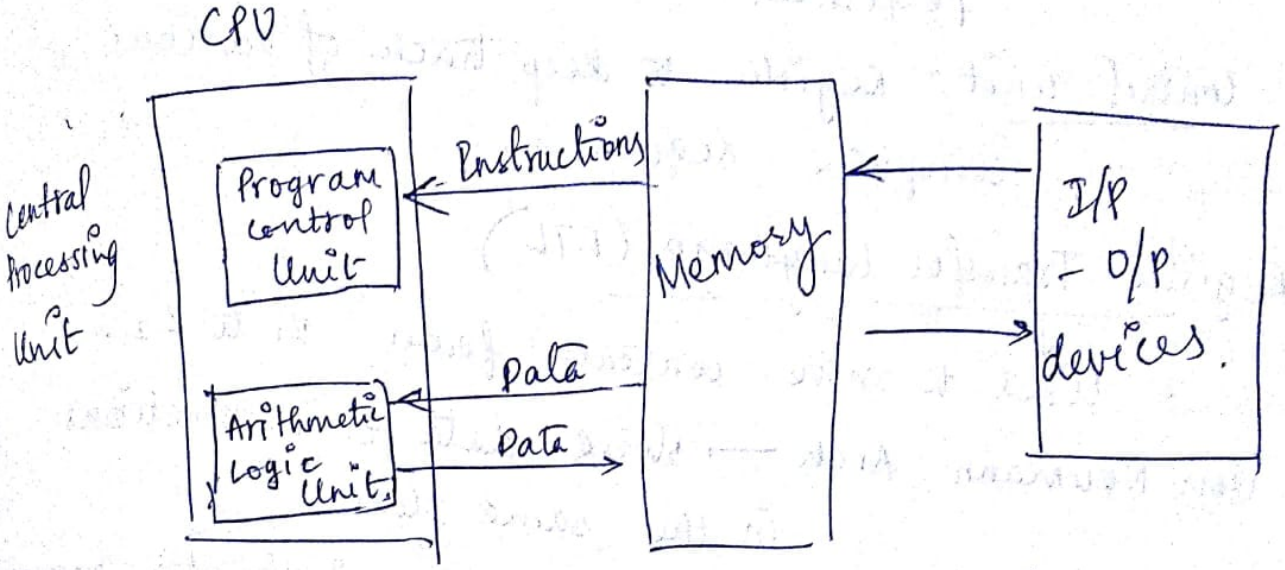
---

## COMPUTER ARCHITECTURE & (CA)
## COMPUTER ORGANIZATION (CO)

CA: This attributes of a system visible to a programmer (or those attributes that have a direct impact on the logical execution of a program).

Examples → Instruction Set:
- Number of bits used to represent data types (Ex: NUMBERS, CHARACTERS, ...)
- I/O mechanisms,
- Techniques for addressing memory

CO: ⇒ Refers to the OPERATIONAL UNITS and their interconnection that realize the architectural specifications. The organizations / attributes include - those hardware details transparent to the programmer, such as CONTROL signals
- Interfaces b/w computer & the peripherals
- Memory technology.




Model of Machine computation

- First Electronic Computer:

ENIAC (Electronic Number Integrator & Calculator)
- Clocked speed - 100 MHz
- Electronic ammulator → storage
- Additional units for → Additions / subtractions.
MULT, DIVISION, SQUARE ROOT and strong function tables.
- MANUAL PROGRAMMING by setting switches and plugging wires.

→ That day I didnt write notes & everything came in exam

FUCK YA!

IDK | 1 | 19

Memory unit : 1000s of registers.

Processor unit: composed of various registers which store operands upon which operations are performed.

Control unit: Register to keep track of various computer sequences.

Register Transfer Language (RTL)

- Used to move contents from R1 to R2.

- Von Neumann Arch → stores data & instructions in the same place.
- Harvard Arch → Program memory or Instruction memory, Data memory is seperate.
  DIGITAL SIGNAL PROCESSORS are based on this architecture.

I/P – O/P PROCESSOR: contains electronic circuits for communicating & controlling the transfer of information b/w computer & outside world.

$2^{2^n}$ → no. of functions!

# DIGITAL LOGIC CIRCUITS

Binary logic → Binary variables
→ Operations that assume a logical meaning

Electric signals ~ 2 states → 3 Volts – 1
0.5V – 0.

XOR

| x | y | o/p |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

$z = x'y + y'x$

X-NOR

$z = xy + x'y'$

| x | y | o/p |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

* Purpose of Boolean Algebra

→ Analysis & Design of digital circuits convenient

1. Express in algebraic form a truth table relationship b/w binary variables.
2. Express algebraically the 1-0 relationship of logic diagrams.
3. Find simpler circuit for the same function

$F(x, y, z) = x + x' + yz$ is always 1.

Boolean Expression: A boolean function specified by a truth table can be expressed algebraically in different ways.

## Basic identities

(1) $x + 0 = x$

(2) $x + 1 = 1$

(3) $x \cdot 0 = 0$

(4) $x \cdot 1 = x$.

(5) $x + x = x$.

(6) $x \cdot x = x$.

(7) $x + x' = 1$

(8) $x \cdot x' = 0$

(9) $(x')' = x$.

(10) $(x + (y + z)) = ((x + y) + z)$

(11) $x \cdot (y + z) = x \cdot y + x \cdot z$

(12) $x \cdot (yz) = (xy) \cdot z$

(13) $(x + y)' = x' * y'$

(14) $(xy)' = x' + y'$.

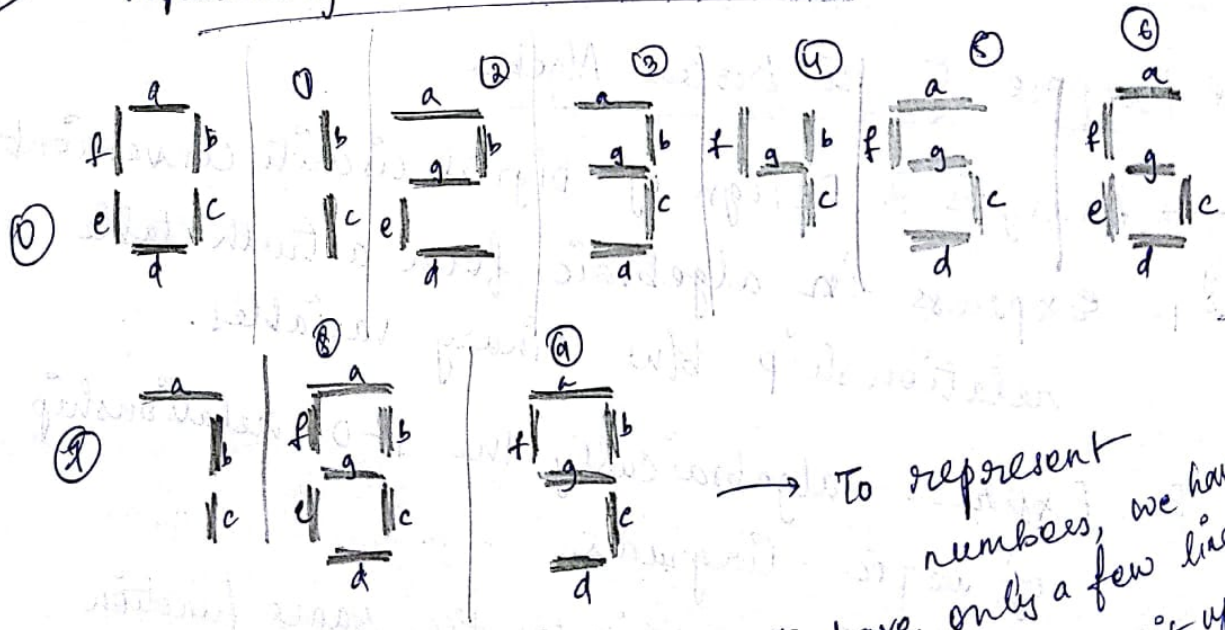
(15) $x + y = y + x$

(16) $x \cdot y = y \cdot x$.

---

13/2/19

## Representing number in digital clocks.




→ To represent numbers, we have to have only a few lines lit up.

lit signs

full too

| decimal | BCD | a | b | c | d | e | f | g |
|---|---|---|---|---|---|---|---|---|
| 0 | 0000 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0001 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0010 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 3 | 0011 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 4 | 0100 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 5 | 0101 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 6 | 0110 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 7 | 0111 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 8 | 1000 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 9 | 1001 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |

15/02/19

## Stored Program Machines:

* Introduction to binary system.
* Stored Program Concept where by program and data got stored in same memory.

Traditional machines: Harvard Mark I.

* Von Neumann architecture: "Sequential" execution of stored (parallel) program instructions.
* MIT's whirlwind: First computer designed with real time applications & computer networking.

Features of in first generation computers.

1. CPU → Control unit
   → Execution unit.

Program control unit → set of registers

## Execution / Data Processing unit.

- Set of registers for storing data.
- An ALU for arithmetic & logic operations.
  execution of
- Additional registers for temporary storage of data

(2) Main memory: Main storage control unit → MAR, MDR.

### Memory Address Register: (MAR)

- MAR is filled with some address (say 'x' denoted as M[x]) in a memory cycle.
- Data from addressed word is read into MDR for read operation and
- Data from MDR (data) is written into the addressed word to write operation.

The no. of bits in an MAR (say 'n') implies memory capacity, whereas $= 2^n$ words.

word size (#o) ⟶ no. of bits of MDR

(3) Secondary Memory: Program & Data.

Can your comp. execute 100! exactly → Nah! (Too Large bruh)

Input to the system from some input device (OR) SECONDARY STORAGE which is a low cost voluminous storage.

④ INPUT DEVICE: Card Reader: Read cards containing program and data.

→ Program or data is read into main storage (from I/P device or secondary storage) under the control of CPU input instructions.

⑤ OUTPUT DEVICES

O/P devices — Printer, Monitors
— Card Punch
→ etc.

O/P data from main storage goes to O/P device under the control of CPU output operation.

⑥ INSTRUCTION FORMATS:

- Consisted of subfield specifiers. Specifying OPCODE and 1, 2, 3, or 4 OPERAND addresses.
- Organisation is shown in Fig 1.6.

Instruction format: OP $A_1$ → ie. one address instruction i.e (Accumulator register is implied second operand)

Instructions are sequentially stored in MAIN STORAGE. CPU sequentially fetches each instruction & executes the operation.

(7) CPU operation:

CPU cycle → Fetch subcycle
CPU cycle → Execute "

(8) Information Representation:

Basic unit of information: transferred b/w main memory & CPU registers
(N-bit wide word) ie.
Word size of main memory

---

18/2/19

Program control unit decodes sub-fields of an instruction

(9) Instruction Set and Micro-operations.

Instruction set is divided broadly into:


Computers.

(1) Data Transfer
(2) Arithmetic.
(3) Logic
(4) Program control
(5) Input / Output.

Bisa's eye view of operation of computer:

Innovations in modern computers:

Main Frame computer } Mini computer → Micro computer
IBM 60 ↓
Personalised computer
laptops ←

Programming a computational task.

Machine language : 0's and 1's.

↑ ASSEMBLER 01001110

Assembly language ADD.

↑

High level language

Kya.

Add $R_1$, $R_2$.
move $R_1 \rightarrow R_2$.

FORTRAN : 40, 70, COBOL, BASIC, ALGOL.

↓

PASCAL

↓

C

↓

C++.

↓

JAVA

Structured PL

↓

IF, TAEN, ELSE

Kya 2.0

CROSS ASSEMBLER

CDMA, FDMA, TDMA

OPERATING SYSTEMS:

Systems software → kernel programming

UNIX application, software WINDOWS

Open source: LINUX.

VIRTUALIZATION:

(Software engineering)

Share the Process / CPU time among multiple users.

TIME SHARING ⟷ Multi tasking
[SINGLE COMPUTER]

PARALLEL COMPUTATION: Multiple computers are available to execute jobs / programs / applications. i.e., multiple CPUs can be effectively used to perform computational tasks.

---

20/2/19 PARALLEL COMPUTATION: (In multicore machines)
Dual core, quad core.

Intel:
Xion – Phi
Processor – 72 cores.

Trivially parallelizable problems:
Serial algorithms.
Models of parallel computation:
SIMD computer.
Single instruction, Multiple data.
↓
Multiple Instruction, MIMD.

Challenging problem } → design and analysis of parallel algorithm

## DISTRIBUTER COMPUTER :

Wireless Sensor Networks (WSNs)

→ distributed and parallel computation.

→ scheduling algorithms for multiple core and memory management.

→ Application specific micro-processors.

(I) Digital signal processors (DSP):

for signal processing applications. Models of computation in most DSPs : Harvard Architecture (Instructions and data, stored in different memory unit).

$$x(n+M) = a_1 x(n) + a_2 x(n+1) + \cdots + a_m x(n+m-1).$$

(II) Graph Processing Unit

~~for digital~~

design for graphic (eg: video games) applications.

GP GPUs : General purpose GPUs.

The company that came up with this — NVIDIA.

(III) In memory computation

(IV) Field programmable gate arrays. (FPGAs)

EPGAs ⟶ systolic arrays (it pumps elements)

(1) Deep learning architectures:

Ex: Hardware accelerators.

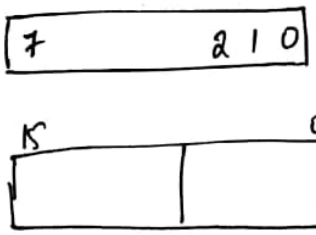
CLUSTER → GRID → CLOUD

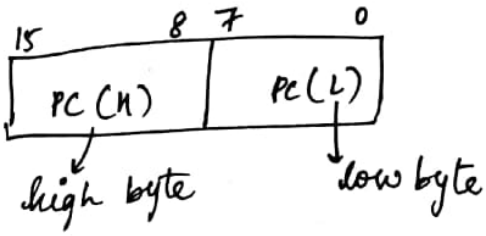
CLOUD → Edge computing

Edge computing → FOG, MIST, DEW.

IPV4 (32 bits) → IPV6 (16 bytes, 128 bits) → IOT

25/02/19




REGISTER Transform: Information transfer : Symbolic form.

$R_2 \leftarrow R_1$ } Destination register has a parallel load capability.

($R_2$ : destination, $R_1$ : source)

→ We want transfer to occur only under a predetermined control condition. IF-THEN statement.

IF $(P=1)$, THEN $(R_2 \leftarrow R_1)$

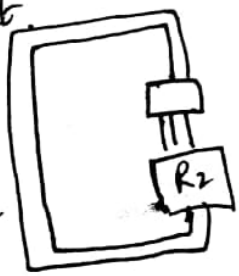
P is a control signal generated in the control section.

* CONTROL FUNCTION : Convenient to seperate the control variables from the register transfer operation by specifying a CONTROL function./

A control function is a Boolean variable that is equal to '1' or '0'.

Control function is included in the statement.

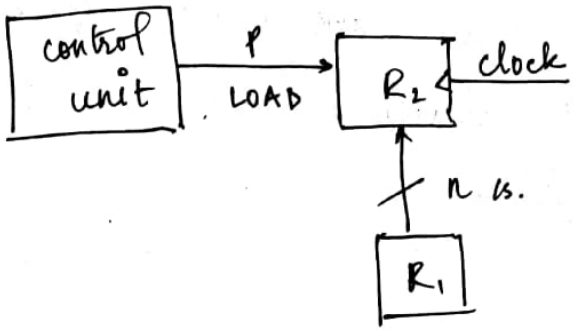
$P : R_2 \rightarrow R_1$.

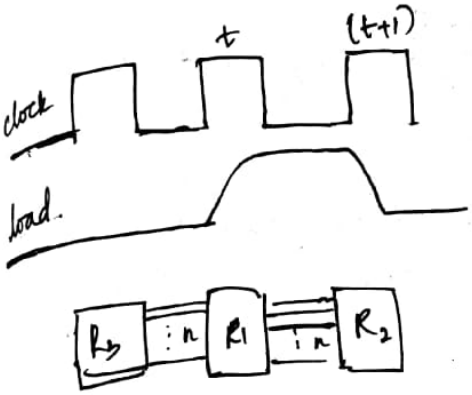



→ Every statement written in register transfer notation implies hardware for implementing it.

T: $R_2 \leftarrow R_1$, $R_1 \leftarrow R_2$

Block diagram that depicts transfer from $R_1$ to $R_2$:




## Basic symbols for Register Transfer

Letter: MAR, R2.

~~Parameter~~ Paranthesis: $R_2(0-7)$, $R_2(L)$ -

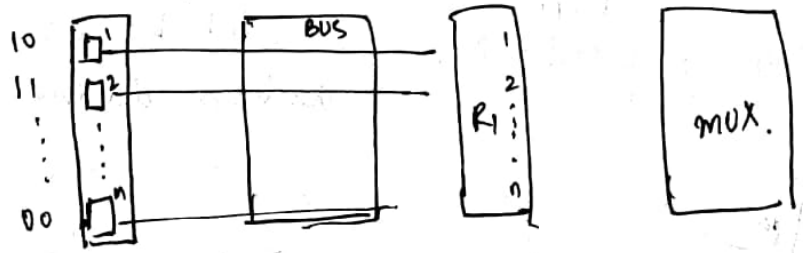
Arrow: $R_2 \leftarrow R_1$.

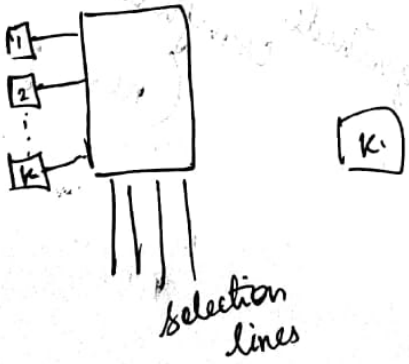
Comma: Seperates two micro-operations.

$R_2 \leftarrow R_1$, $R_1 \leftarrow R_2$.

## 4.3 BUS AND MEMORY TRANSFERS

Multiple Register Configuration: COMMON BUS.



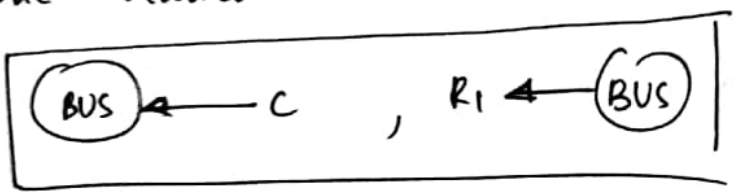

For each bus line, we need a multiplexer and each mux has k inputs.

EXAMPLE: Common bus for 8 registers of 16 bits require 16 multiplexers, one for each line of bus.

- Each multiplexer must have 8 data i/p lines and 3 selection lines to multiplex one significant bit in the 8 registers.

---

Symbolic Notation




(or) $R_1 \leftarrow C$.

Alternative method : 3 state gates ← 0, 1, high impedance

MEMORY TRANSFER:

READ: Transfer from Memory Word, M to the outside environment

WRITE: Transfer of new information to be stored into the memory is called WRITE operation.

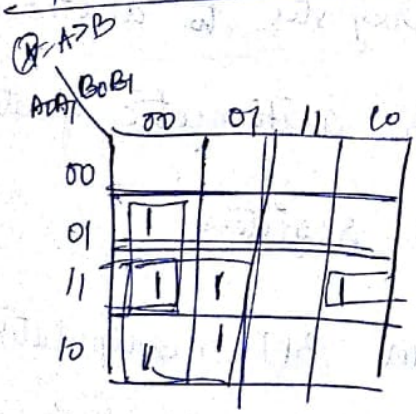
Memory Word ≡ M. Selected by the memory address during the transfer.

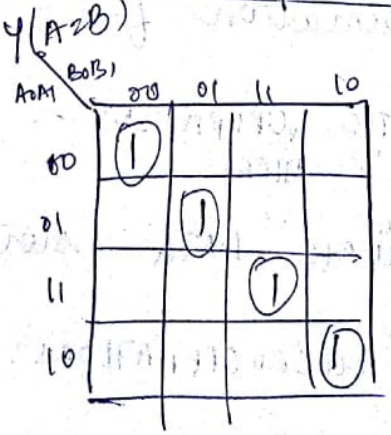
---

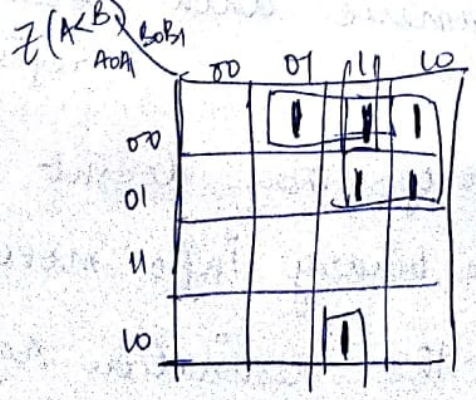
Tutorial 26/02/19.

2-bit magnitude comparator (next page)

| $A_0$ | $A_1$ | $B_0$ | $B_1$ | o/p $x\,(A>B)$ | $y\,(A=B)$ | $z\,(A<B)$ | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 2 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 3 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 4 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 5 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 6 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 7 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 8 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 9 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 10 |
| 1 | 0 | 1 | 1 | | 0 | 1 | 11 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 12 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 13 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 14 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 15 |



26/02/19

Currently computing paradigms for machine learning

Assembly language - fast but very tough to program.

## 4.4 Types of Microprocessors

Definition: Elementary operation performed with data stored in registers.

The micro operations most often encountered in DIGITAL COMPUTERS are classified into 4 categories.

① Register Transfer Microoperations: Transfer binary information from one register to another.

② ARITHMETIC MICRO OPERATIONS: perform arithmetic operation on NUMERIC DATA stored in registers

③ LOGIC MICRO OPERATIONS: Perform BIT manipulations operations on non numeric data stored in registers.

Note: Register transfer micro operation doesnt change the info. content when the binary info moves from the source register to the destination register.

- The other 3 types of microoperations change the info. content during transfers.

→ BASIC ARITHMETIC MICROOPERATIONS

Addition, subtraction, increment, decrement, shift.

shifts → arithmetic shifts.

Example: $R_3 \leftarrow R_1 + R_2$ : ADD Microoperation.

$R_5 \leftarrow R_1 - R_2$

→ The arithmetic operations of multiply and divide are not listed in the table 4.3. These two operations are VALID ARITHMETIC OPERATIONS, but are not included in the basic set of microoperations. The only place where these operations can be considered as microoperations is in a digital system, where they're implemented by means of a combinational ckt.

Example hardware

- 4 bit binary adder based on.
- Increment/Decrement microoperations are implemented with a combinational circuit or with a binary up down counter.

## 4.5 LOGIC MICRO OPERATIONS :

- They specify binary operations for strings of bits stored in registers. These operations consider each bit of the register seperately and treat them as binary variables.

Example: Exclusive → 'OR' microoperation with the contents of two registers $R_1$, $R_2$.

$$P: R_1 \leftarrow R_1 \oplus R_2.$$

```
1010  — content of R1
1100  → R2
----
0110  → content of R1 after P=1.
```

### SPECIFIC SYMBOLS

Specific symbols will be adopted for the ~~LO~~ LOGIC MICRO OPERATIONS OR, AND and COMPLEMENT. to DISTINGUISH them from the corresponding symbols used to express Boolean functions.

$\vee$ ... OR , $\wedge$ ... AND , $\bar{B}$ ... COMPLEMENT.
ie. Bar on top of symbol.

- By using different symbols, it will be POSSIBLE to DIFFERENTIATE b/w a LOGIC MICRO OPERATION and ~~control~~ CONTROL (or Boolean) functions.

+ used for arithmetic plus or logic OR.

- Although the plus symbol has two meanings, it will be possible to distinguish b/w them by noting WHERE THE SYMBOL OCCURS.
- When the symbol '+' occurs in micro-operations, it will denote an arithmetic plus.
- When it occurs in a control (or boolean) function it will denote an OR operation.
- We will never use it to symbolize an OR microoperation.

$P + Q : R_1 \leftarrow R_2 + R_3 ; R_4 \leftarrow R_5 \vee R_6$.

'+' b/w 'P' and 'Q' is an OR operation b/w two binary variables of a CONTROL FUNCTION.

+ b/w $R_2, R_3$ .... Add microoperation.

- OR logic operation is represented by the symbol 'V' between $R_5$ and $R_6$.

Left shift / double notaa

SOME APPLICATION:

- Logic microoperations are very useful for manipulating individual bits or a portion of word stored in register.
- They can be used to change bit values, delete a group of bits or insert new bit values into a register.

V

05/03/19

Tutorial

## Register transfer & μ-ops

Simple digital system :- Combinational & Sequential ckts. are used to create simple digital sys

→ low level building blocks

→ No. of registers
No. of ops

## REGISTERS

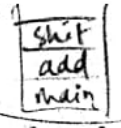
→ More the general purpose registers, faster the execution
↓
temporary storage

→ AC – Accumulator – processor register.
↳ used to hold the first ~~oeld~~ value & o/p.

LDA, STA, ADD.

→ PC – program counter – the next instruction's address is stored here
the content of PC is stored in stack.

```
main( ){
add( );
}
add( ){
shift( );
}
shift( ){
}
```




(or)

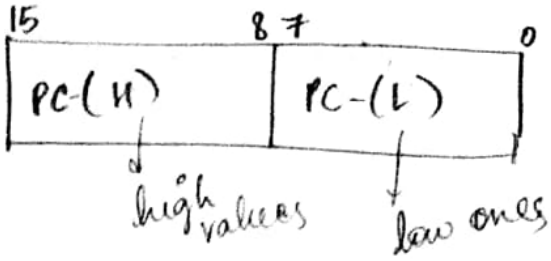
→ Instruction Register → holds the instruction to be executed.

→ MAR (memory address register) – holds address (main memory address content)
then → DR( data R) & holds the content.

→ PC (address register)

# COMPUTER ORGANIZATION :

→ Set of Registers
→ Microoperations
→ Control signals




RTL

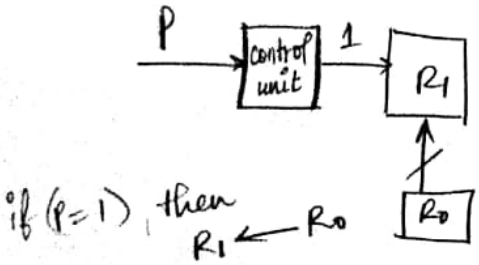
$R_1 \leftarrow R_0$

P: $R_1 \leftarrow R_0$

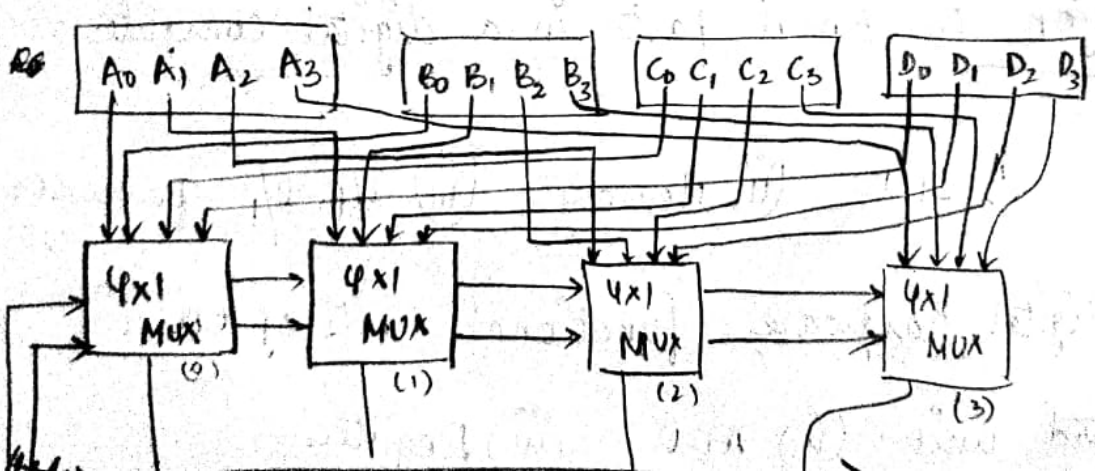
on reading the instruction P, if its 1,

$R_0$ will move to $R_1$

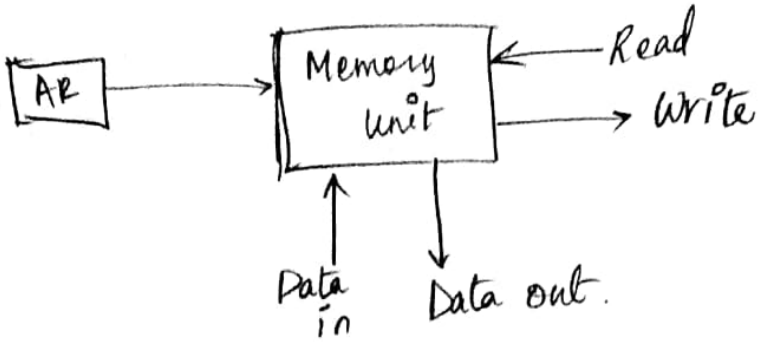
Represented as




(b) For a 4-bit Bus Transfer, we use a 4×1 multiplexer.



# Memory Transfer




$R_3 \leftarrow R_1 + R_2$ ——— addition

$R_3 \leftarrow R_1 - R_2$ ——— subtraction

$R_2 \leftarrow R_1'$ ——— 1's complement

$R_2 \leftarrow R_1' + 1$ ——— 2's complement

$R_3 \leftarrow R_1 + R_2' + 1$ ——— 2's complement subtraction

$R_1 \leftarrow R_1 + 1$ ——— Increment

$R_1 \leftarrow R_1 - 1$ ——— Decrement.

ALU + control unit ⟶ CPU.

---

06/03/19

- Major Functional parts in a digital computer:

(i) CPU (ii) Memory (iii) I/P-O/P processor.

- Main digital hardware functional units of CPU:

(i) Control unit (ii) ALU (iii) Registers

- Functions of Control unit: Initiates a sequence of micro-operations.

- The # of different types of micro operations that are available in a given system is finite.
- Complexity of the digital system is derived from the # of sequences of micro operations that are performed.

→ Two methods of Implementing CONTROL UNIT:

(i) Hardwired control unit.
(ii) Microprogrammed control units.

---

(2) Design of hardwired control unit involves the use of FIXED INSTRUCTIONS, FIXED LOGIC BLOCKS of arrays of ENCODERS, DECODERS.

→ Key characteristics of hardwired control logic are :-

(i) High speed of operation.
(ii) Expensive
(iii) Relatively complex.
(iv) No flexibility of adding.

Example CPU's: with hardwired logic control:

Intel 8085, Motorola 6802, Zilog 80 and any.
RISC (Reduced Instruction Set Computers) CPU's.

---

Note: When the control signals are generated by hardware using conventional logic design techniques, the CONTROL UNIT is said to be hardwired.

MICROPROGRAMMING: is a SECOND alternative for designing the CONTROL UNIT of a digital computer.

- The principle of MCU is an ELEGANT any SYSTEMATIC method of controlling the MICROOPERATION sequences in a digital computer.

CPU's with MCU are Intel 8080, Motorola 68000 and any CISC (Complex Instruction Set Computer (CPUs).

Details of MCU → P: $R_2 \leftarrow R_1$

- The control function that specifies a MICRO OPERATIONS. is a binary variable. When it is in one binary state, the corresponding micro-operation is executed.

- A control variable in the opposite binary state doesn't change the state of the registers in the system. The active state of a control variable may be either '1' state or the '0' state depending on the application.

- In a BUC organised system, the control signal that specify micro-operations are GROUPS of BITS that select the paths in multiplexers, decoders and Arithmetic Logic units.

~~The control~~

- The control unit initiates a series of sequential steps of MICRO OPERATIONS. During any given time, certain operations are to be micro

→ The control variables at any point of time can be represented by a string of 1s and 0s called a CONTROL WORD. As such control words can be programmed to perform various operations on the components of the system.

→ A control unit whose BINARY CONTROL VARIABLES are stored in a memory is called a MPCU → μprogrammed Control unit.

→ Each word in CONTROL MEMORY contains within it a MICRO-OPERATION. The μ-instruction specifies one or more LNSTRUCTION microoperations for the system.

→ A sequence of microinstructions constitutes a MICROPROGRAM

→ Since attenuation of the MICROPROGRAM are not needed once

→

Fuck you

07/03/19

## Hardware Description Languages:

→ The design of complex digital systems at the gate and flip-flop level is tedious and time consuming process.

→ Digital systems can be described at the register transfer level by means of Hardwire Description Language. (HDL)

→ The HDL is a viable solution for designing & debugging a digital system at a higher level before conversation conversion to the gate & flip-flop level. To do the conversion, computer aided design (CAD) tools can be used (FPGA, PLA → programmable logic)

Note: This is similar to the converting a high level language program into machine language using compiler.

→ Popular HDL's are VHSIC HDL or VHDL and verilog HDL.

VHSIC ⟶ very high speed integrated circuit.

$R_1 \rightarrow R_2 \rightarrow R_3$

## BASIC COMPUTER ORGANIZATION & DESIGN

Organization of computer: Defined by its internal registers, the TIMING and CONTROL STRUCTURE and the SET OF INSTRUCTIONS that it uses.

## DESIGN OF A DIGITAL COMPUTER (DC)

- The internal organization of a digital system is defined by the sequence of MICRO-OPERATIONS it performs on the data stored in registers.
- The general purpose DC is capable of executing various μ-OPERATIONS and in addition can be instructed as to what specific sequence of operations it has to perform.
- The user of a computer can CONTROL the process by means of a PROGRAM.
- Program is a set of INSTRUCTIONS that specify operations, operands and the sequence by which processing has to occur.
- The data processing task may be altered by specifying a new program with a different instructions or specifing with the same instructions with different data

## A COMPUTER INSTRUCTION:

- Binary code that specifies a sequence of μ-OPERATIONS.
- Instruction code, together with DATA are stored in memory.
- The computer reads each instruction from memory and places it on a CONTROL REGISTER.
- The control then interprets the binary code of instruction and proceeds to executing it by issuing sequence of μ-operations.

---

- Every computer has its own unique instruction set.

- It is usually divided into parts, each having its own particular interpretation.
- MOST BASIC PART of an instruction : operation part. The operation code is a group of bits that define such operations as ADD, SUBTRACT, MULTIPLY, SHIFT and COMPLEMENT.
- The number of bits required for the OPERATION CODE (op code) of an instruction depends on TOTAL NO OF OPERATIONS.