
cisc architecture | COA
4:14 / 5:53

Complex Inst Set Comp.

⇓

⇒ To minimize the no: of inst per prog
ignoring the no:- of cycles per inst

⇒ Desktop & laptop comp

⇒ Very little RAM is req to store inst

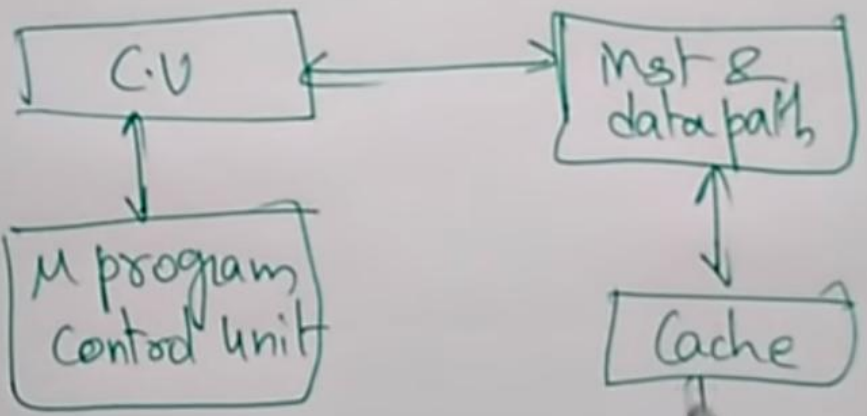
Reduced Inst Se

⇓

1 inst ⇒ 1 clock

⇒ tablets, smartph

Architecture :-



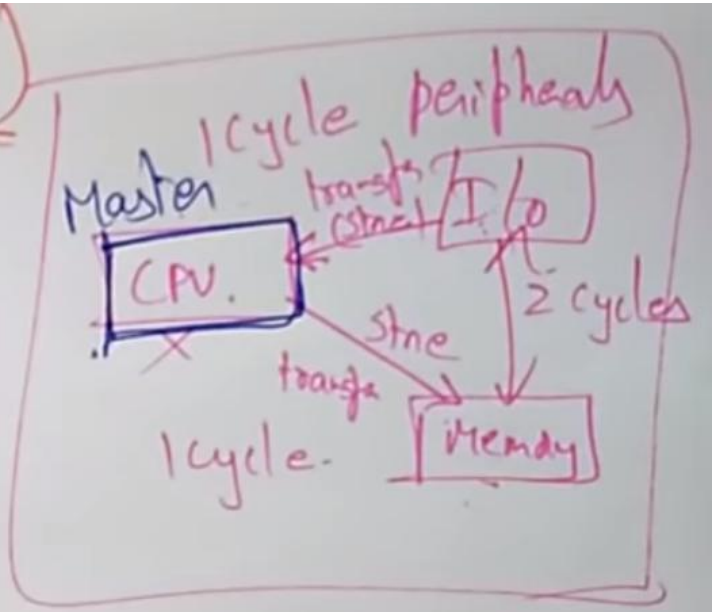
DMA Controller (8257)

//

Direct Memory Access

//

It is designed by Intel to transfer data at the fastest rate



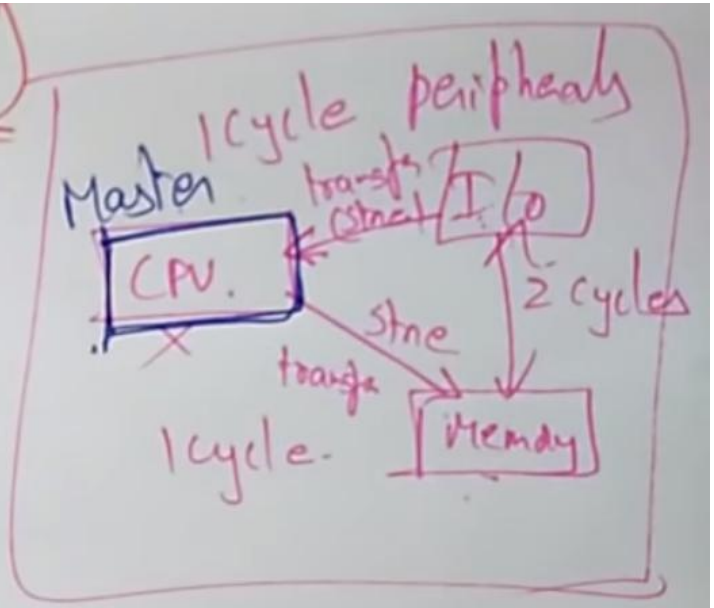
DMA Controller (8257)

||

Direct Memory Access

||

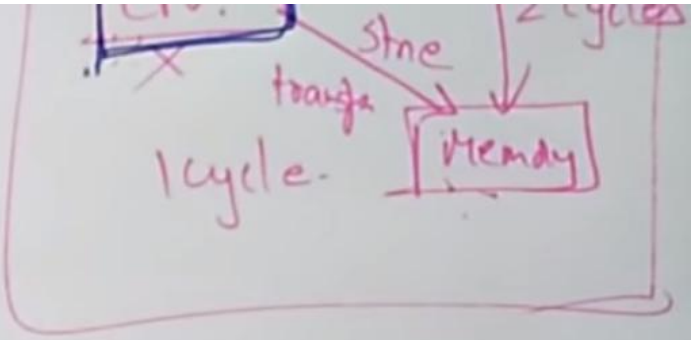
It is designed by Intel to transfer data at the fastest rate



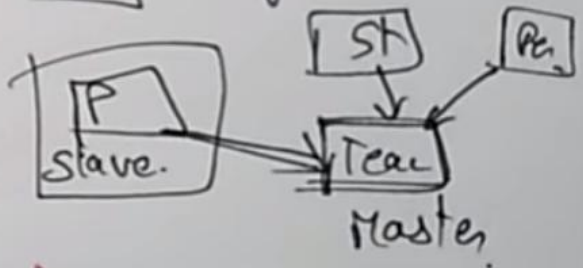
⇒ It allows the device to transfer data directly to/from memory without CPU involvement.

Direct Memory Access

It is designed by Intel to transfer data at the fastest rate



⇒ It allows the device to transfer data directly to/from memory without CPU involvement.



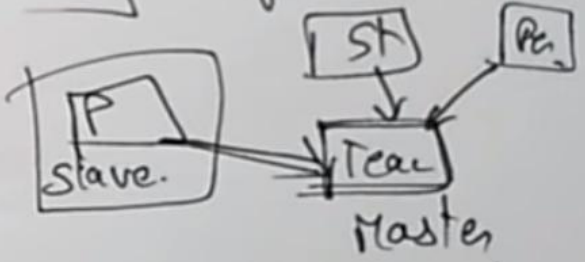
⇒ How DMA operations are performed?

① Device (peripheral) wants to send data to memory first device has to send DMA req (DRQ) ⇓ DMA Controller.

DMA controller basic operation

It is designed by Intel to transfer data at the fastest rate

⇒ It allows the device to transfer data directly to/from memory without CPU involvement.



How DMA operations are performed?

① Device (peripheral) wants to send data to memory
first device has to send DMA req (DRQ)
⇓
DMA Controller

Peripheral → DRQ → DMA → HLQ → CPU

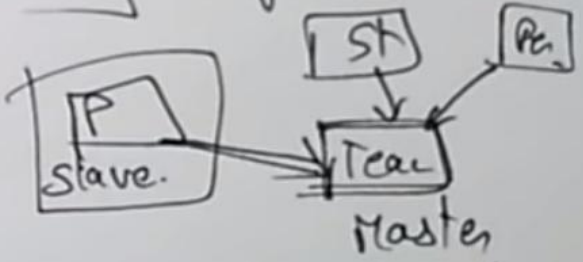
② DMA controller send ⟹ HLQ (hold req) to CPU

8:23 / 13:52

DMA controller basic operation

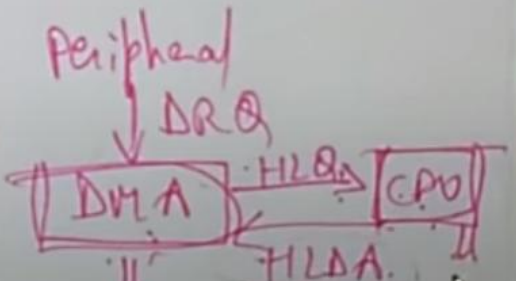
It is designed by Intel to transfer data at the fastest rate

⇒ It allows the device to transfer data directly to/from memory without CPU involvement.



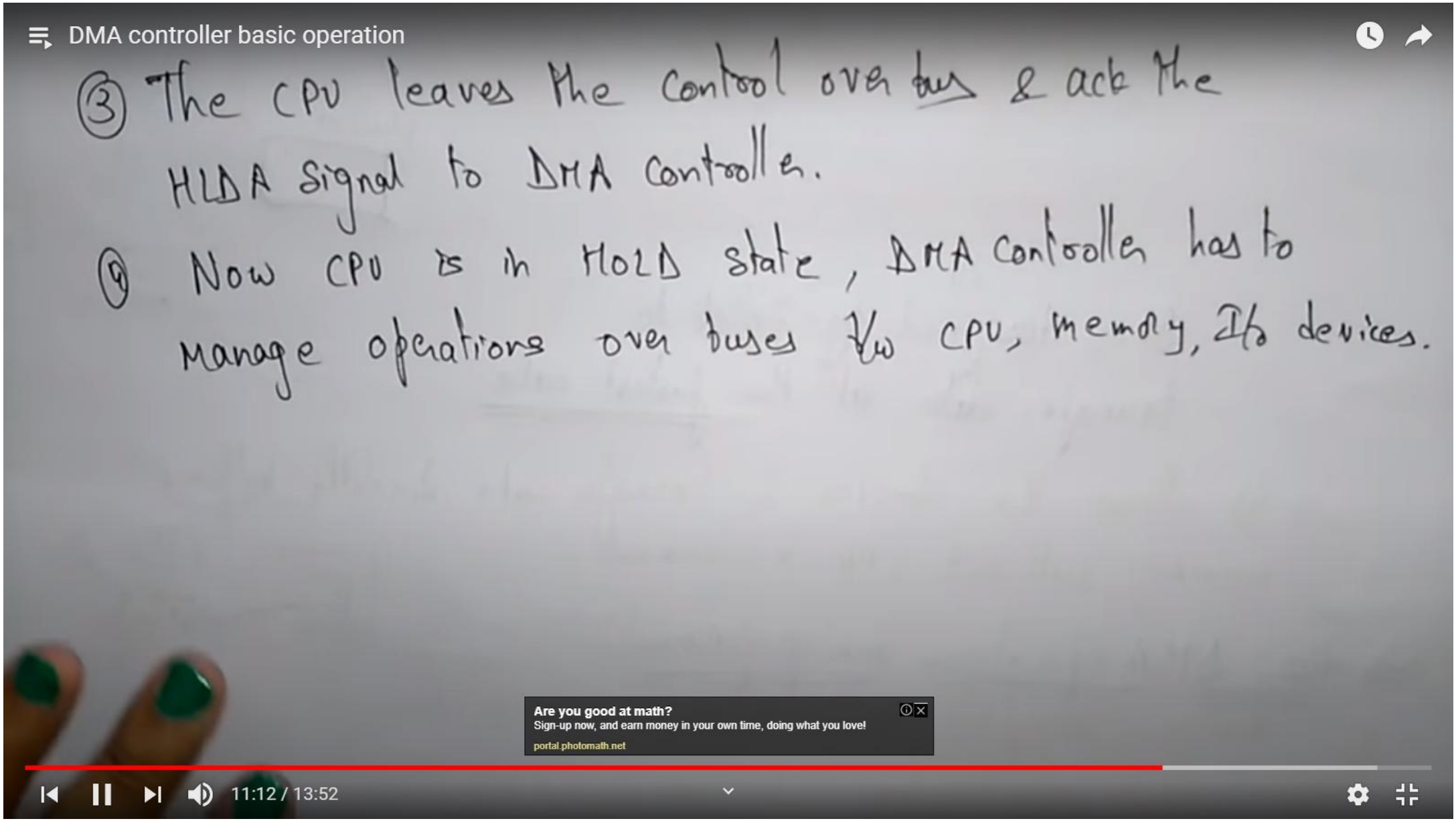
⇒ How DMA operations are performed?

① Device (peripheral) wants to send data to memory first device has to send DMA req (DRQ) ⇓ DMA Controller



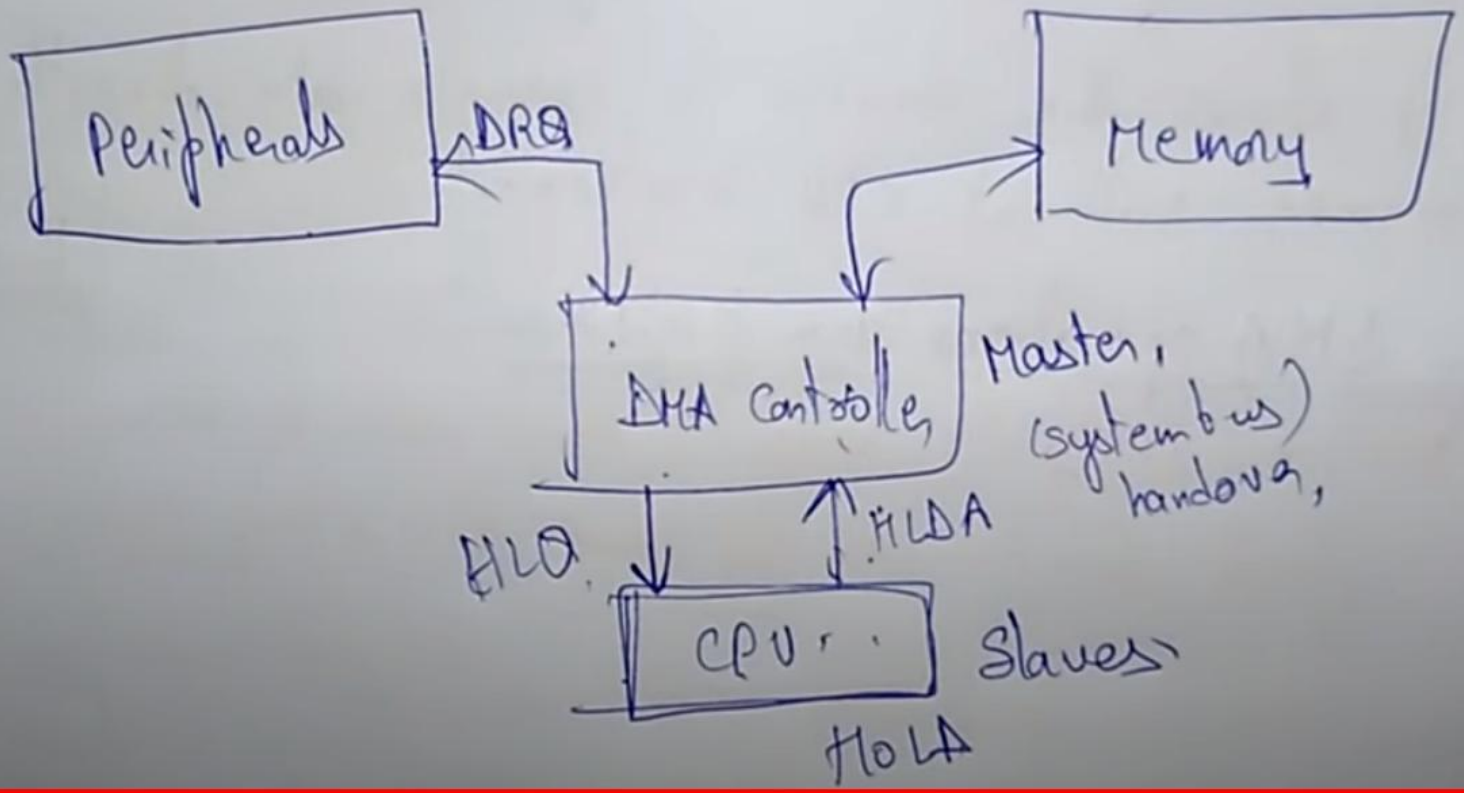
② DMA Controller Send ⟹ HLQ (hold req) to CPU & Waits for H CPU to se LDA

9:44 / 13:52


DMA controller basic operation
③ The CPU leaves the control over bus & ack the HLDA signal to DMA controller.
④ Now CPU is in HOLD state, DMA controller has to manage operations over buses I/O CPU, memory, I/O devices.
Are you good at math?
Sign-up now, and earn money in your own time, doing what you love!
portal.photomath.net
11:12 / 13:52

DMA controller basic operation

④ Now CPU is in HOLD state, DMA Controller has to manage operations over buses I/O CPU, memory, I/O devices.



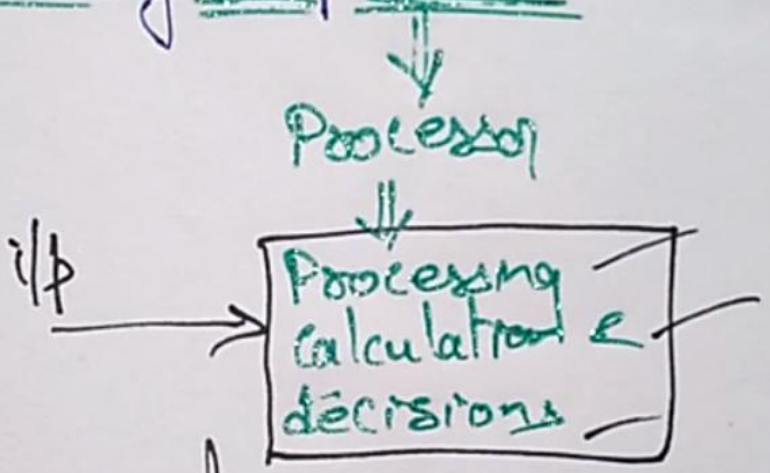
13:12 / 13:52

Design & Implementation of Simple CPU

- ALU
- Registers
- Memory & I/O interface
- Control unit
- clock
- Bus

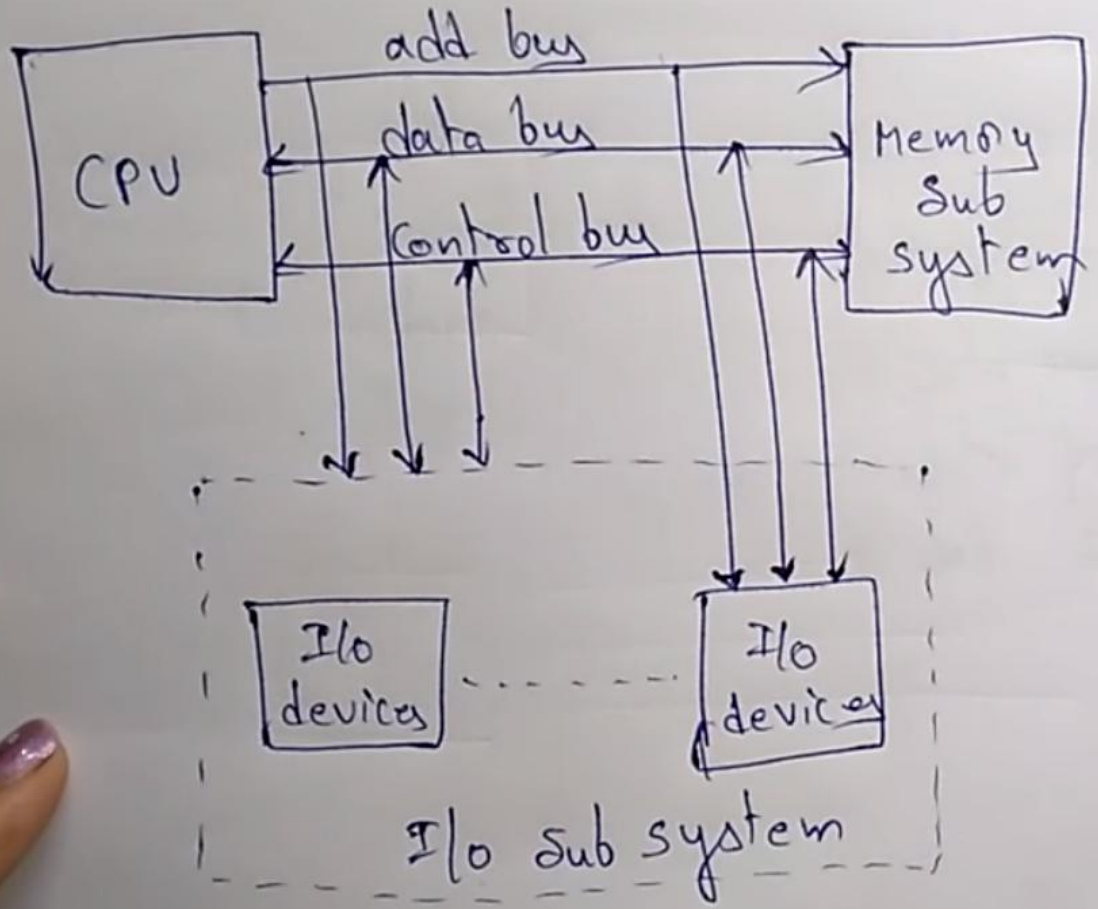
Design & Implementation of Simple CPU

- ALU
- Registers
- Memory & I/O interface
- Control unit
- clock
- Bus

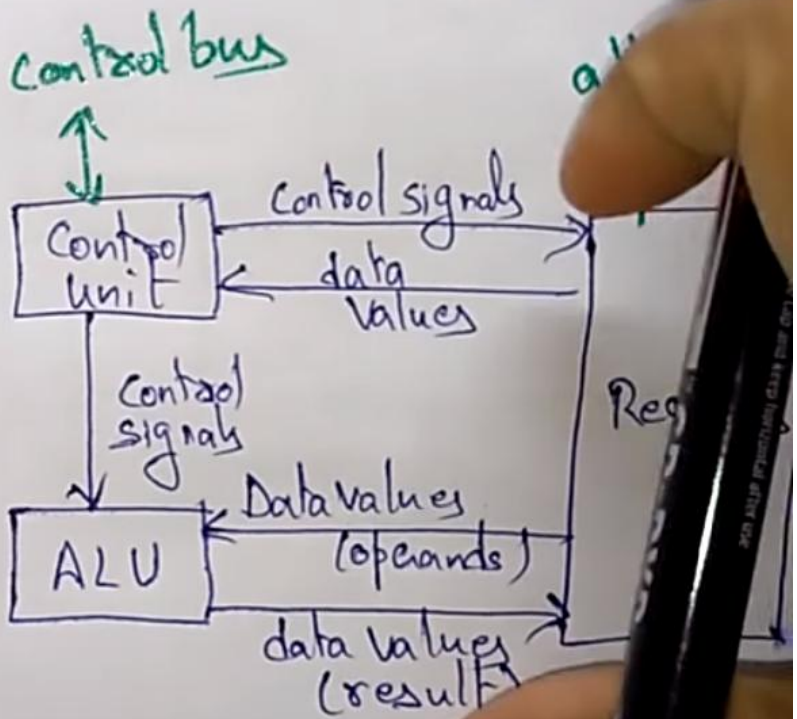


→ Control
→ data movement
→ data processing
→ data storage

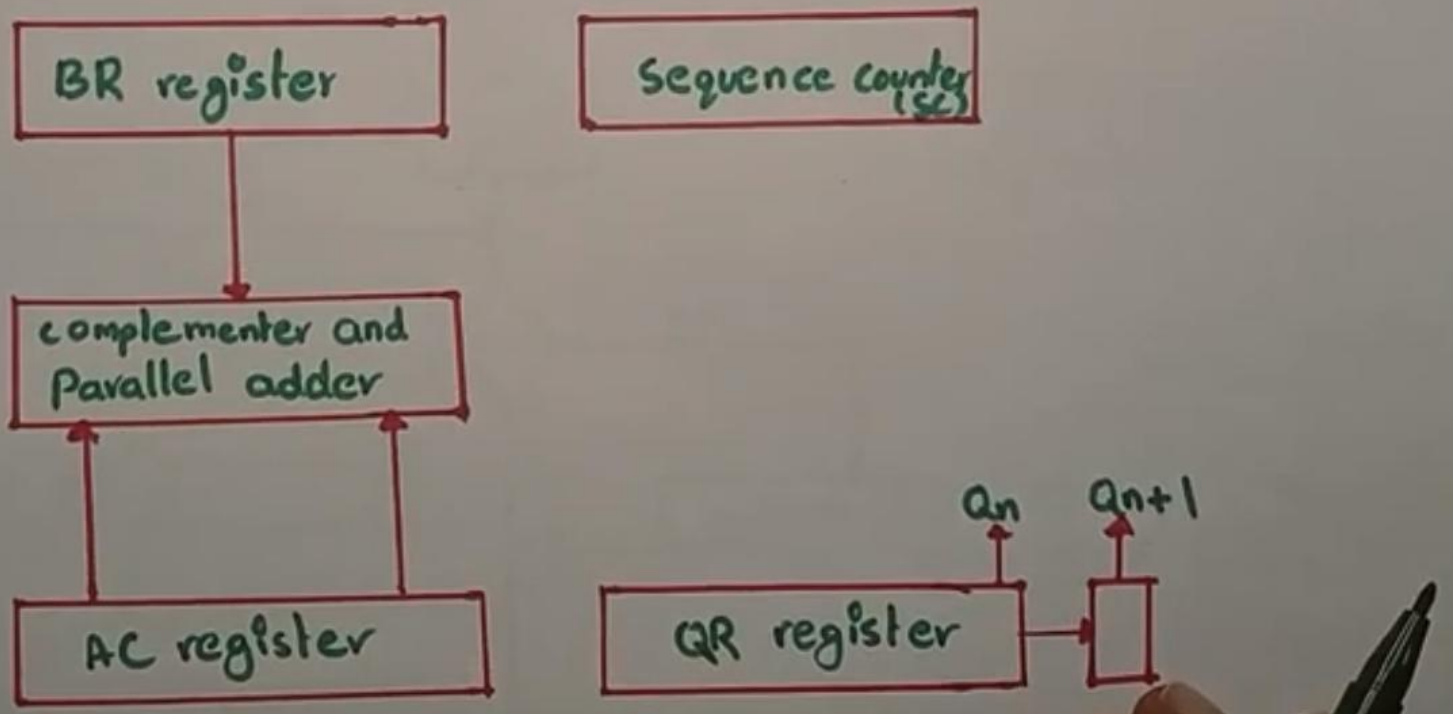
Generic Computer Organization

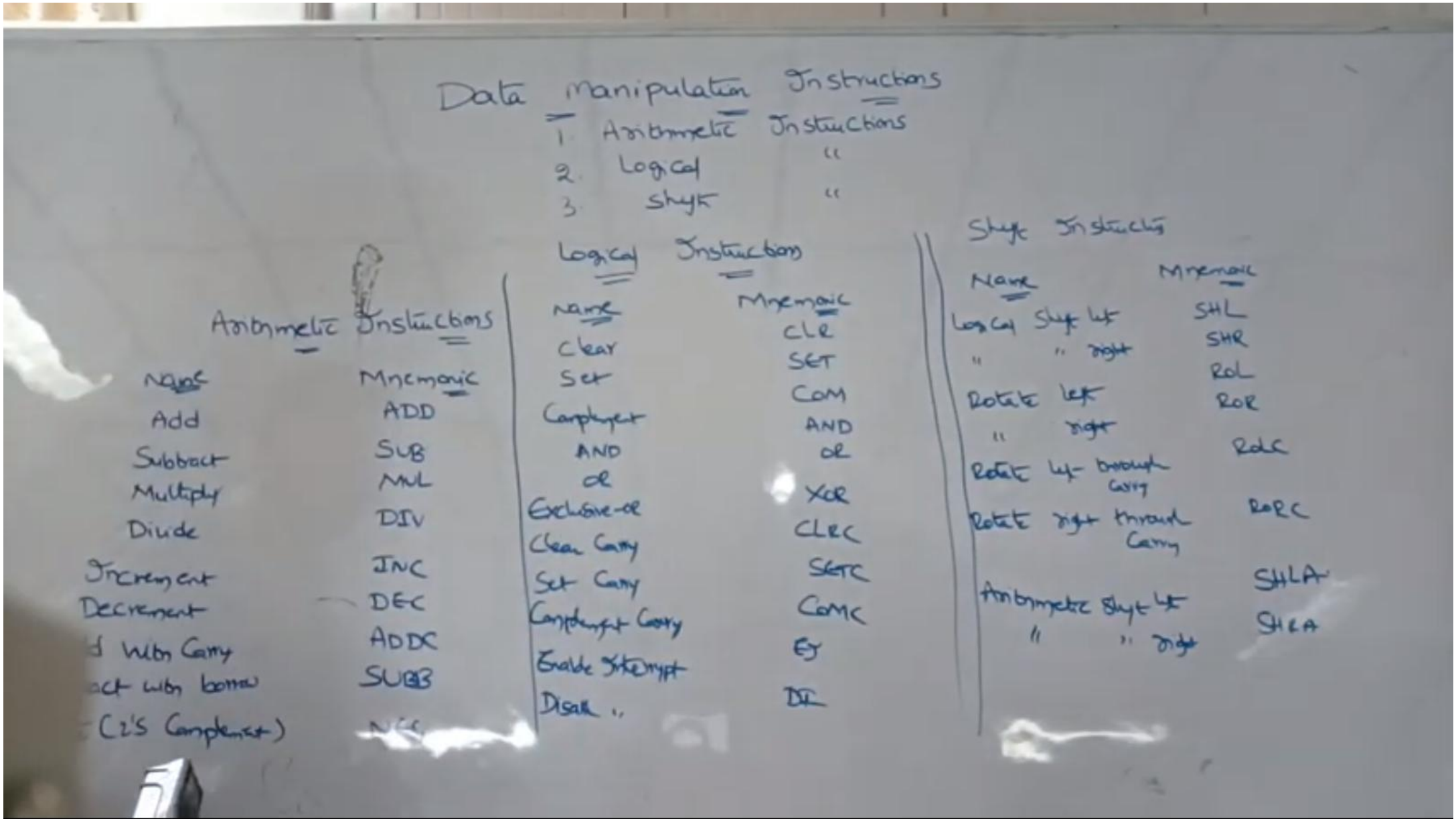


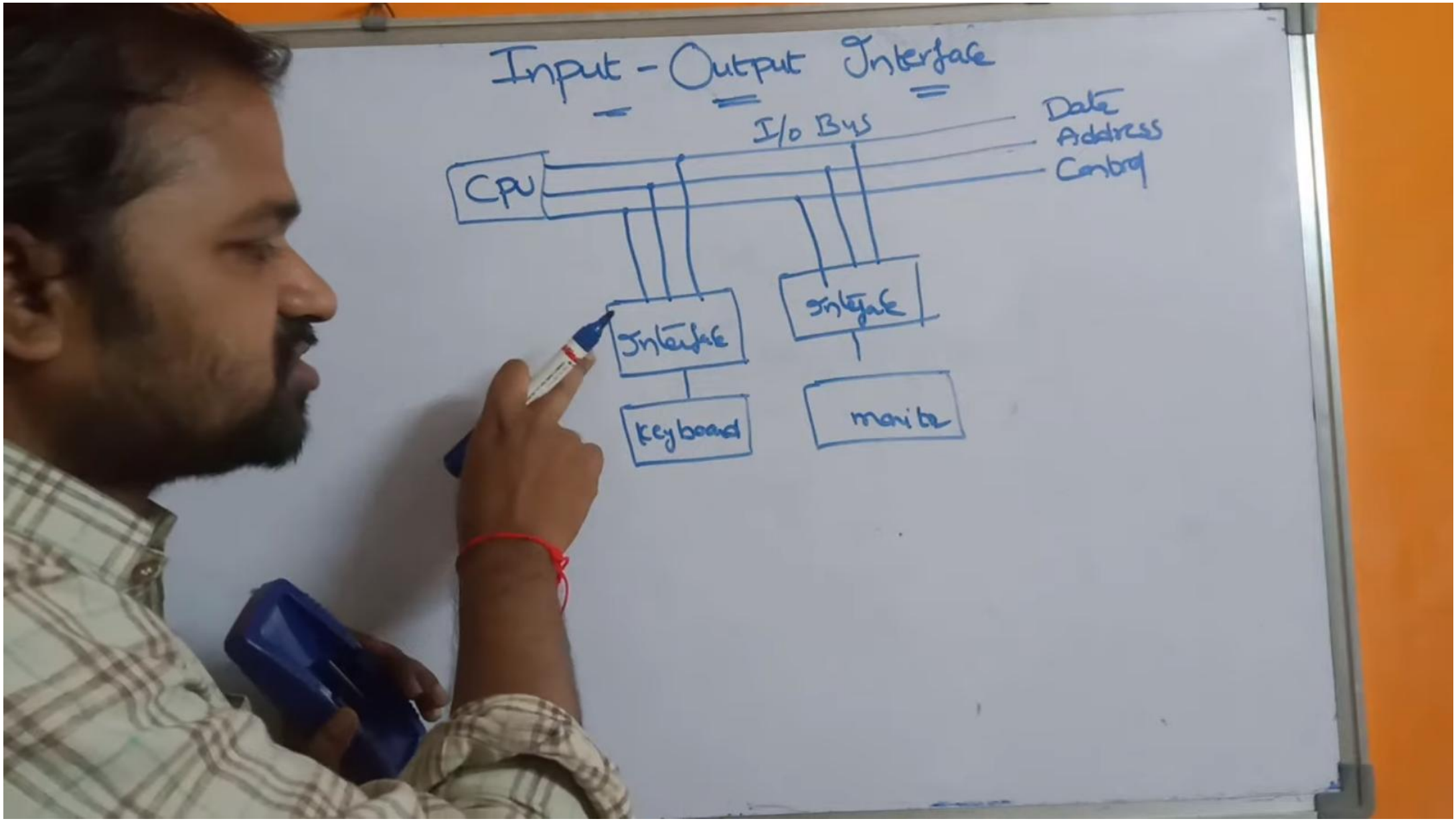
CPU Internal Organiza

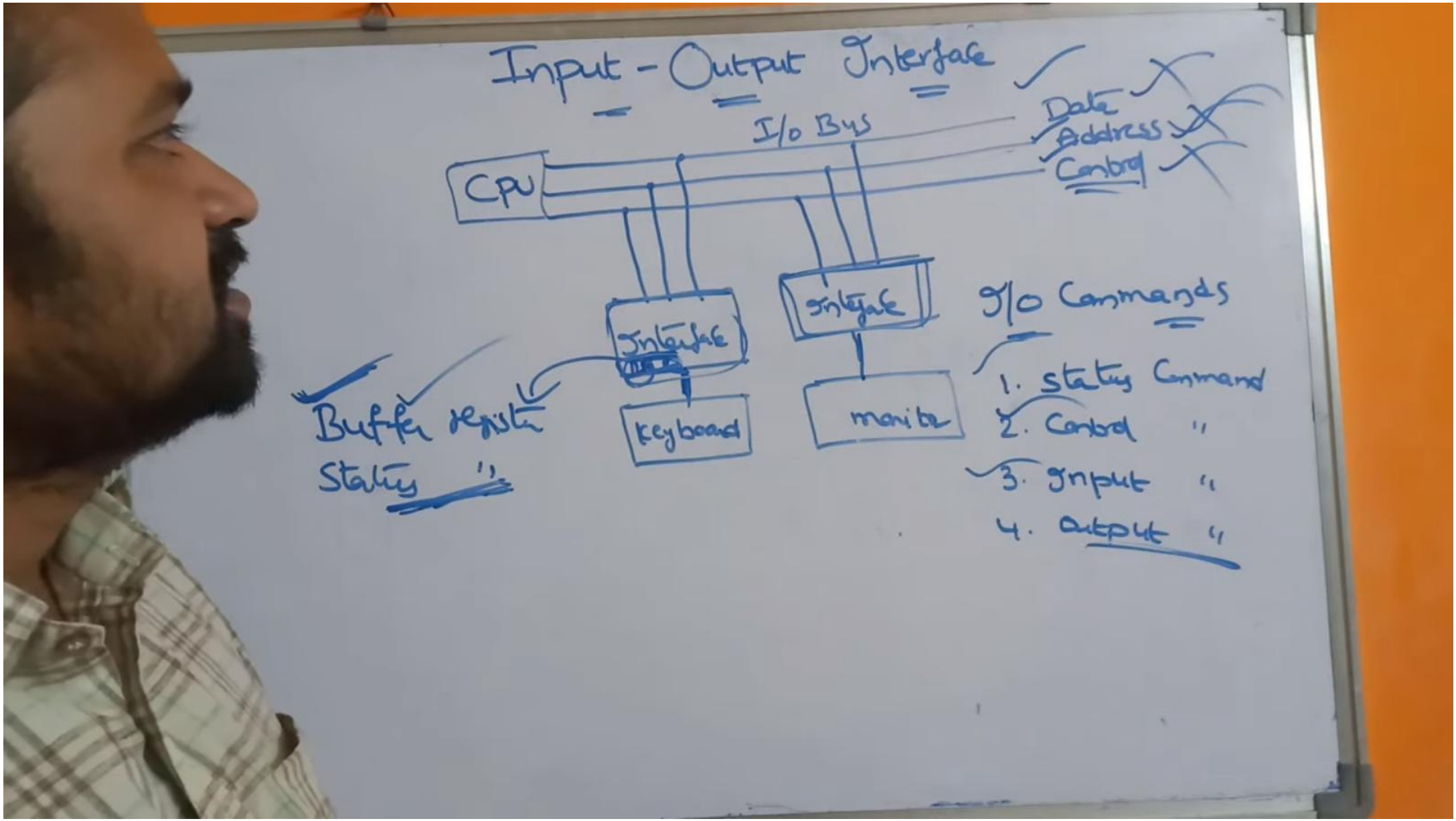


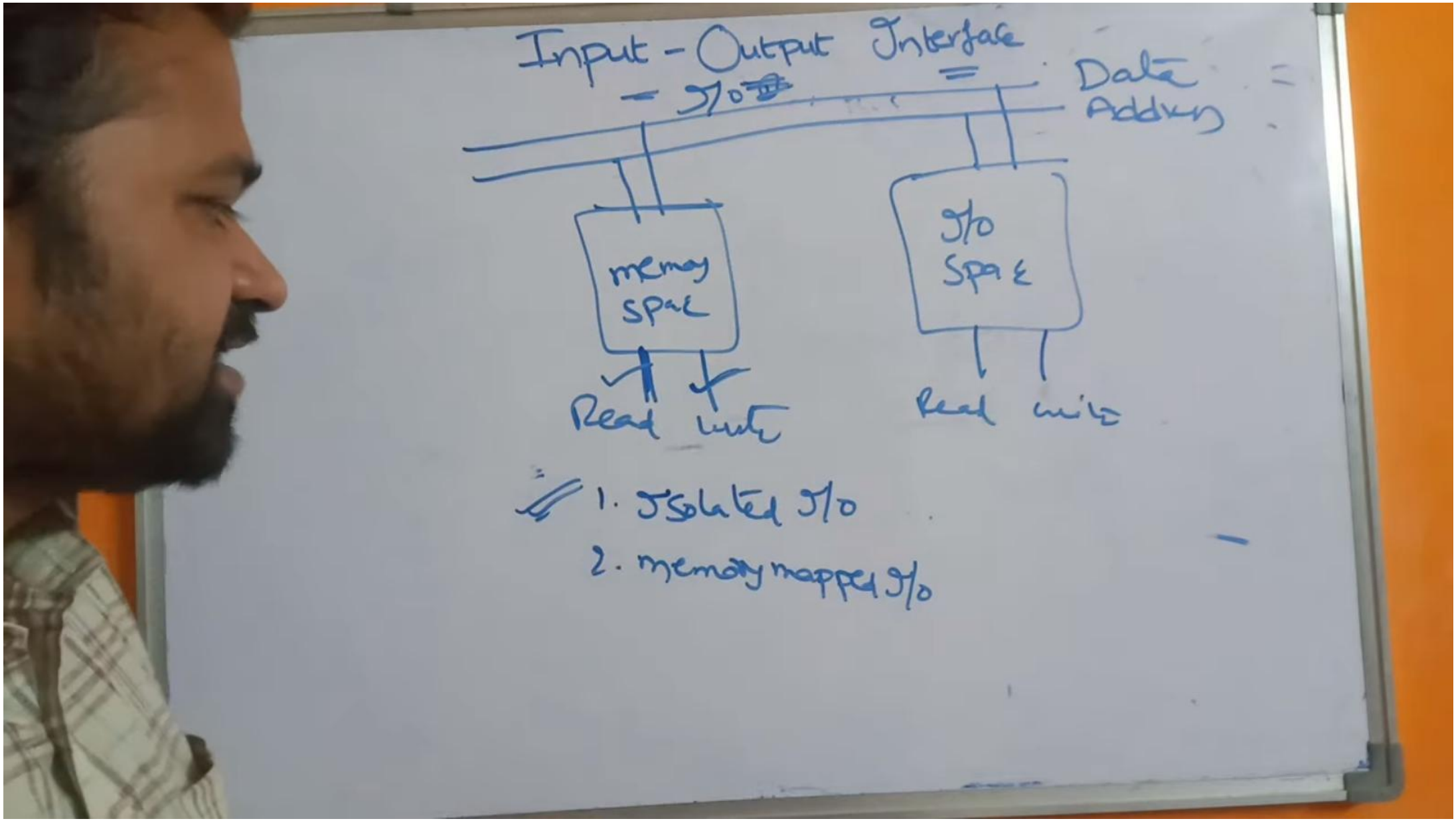
Hardware for Booth algorithm:

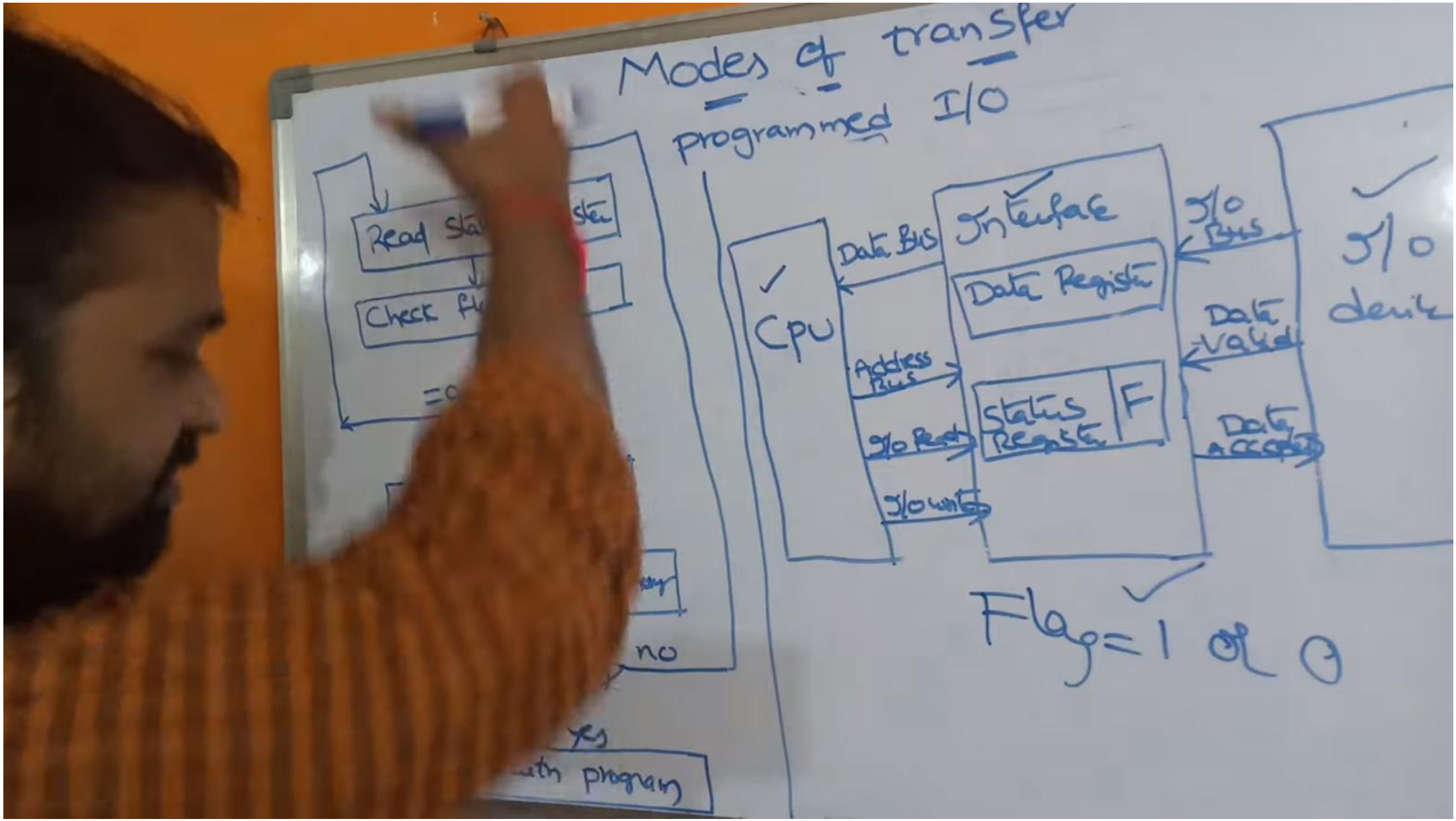


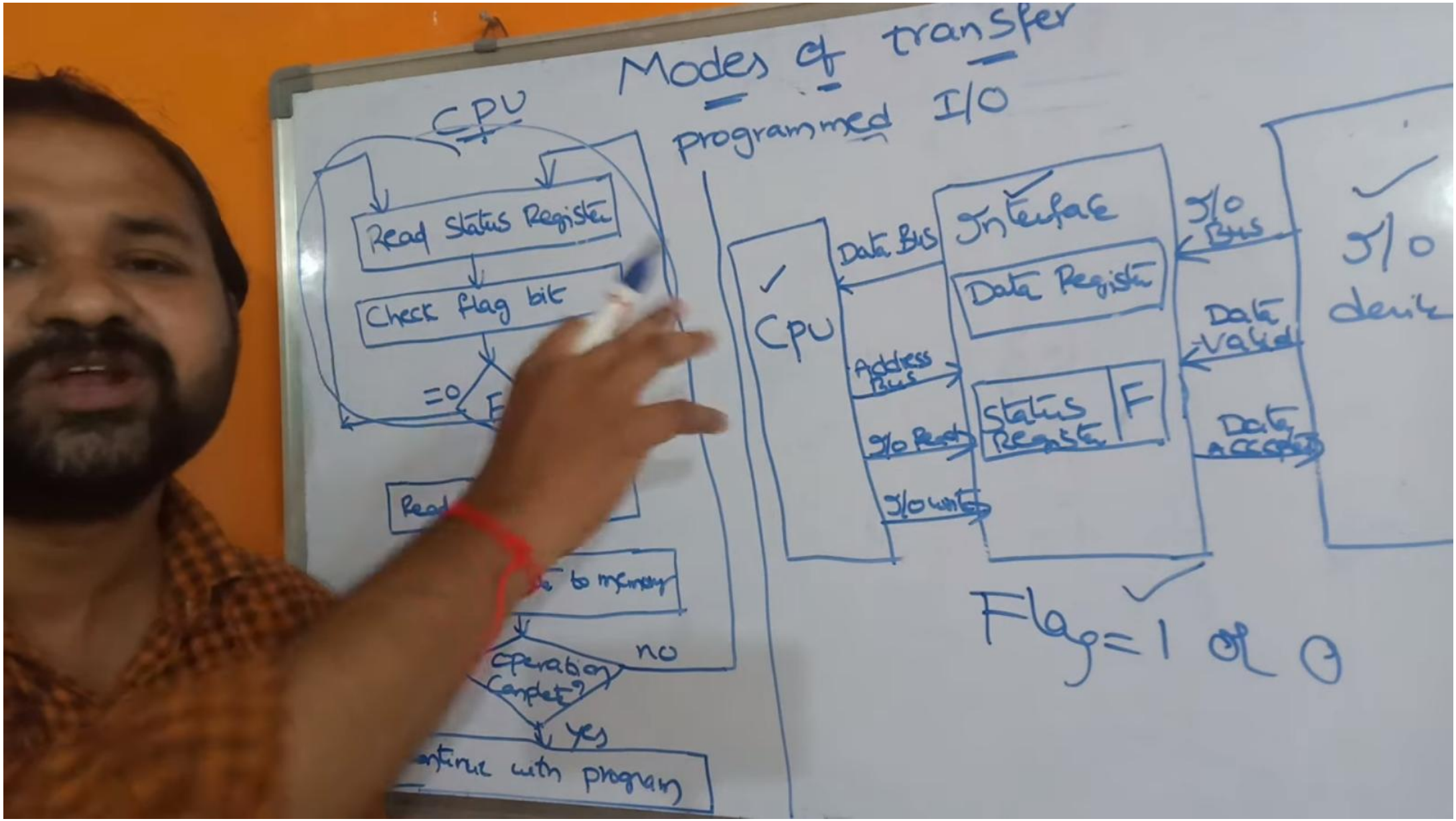

Data manipulation Instructions
1. Arithmetic Instructions
2. Logical "
3. Shift "
Arithmetic Instructions
Name Mnemonic
Add ADD
Subtract SUB
Multiply MUL
Divide DIV
Increment INC
Decrement DEC
d with Carry ADDC
act with borrow SUBB
(2's Complement)
Logical Instructions
Name Mnemonic
Clear CLR
Set SET
Complement COM
AND AND
OR OR
Exclusive-OR XOR
Clear Carry CLRC
Set Carry SETC
Complement Carry COMC
Enable Interrupt EI
Disable " DI
Shift Instructions
Name Mnemonic
Logical Shift left SHL
" " right SHR
Rotate left ROL
" right ROR
Rotate left through Carry RLC
Rotate right through Carry RRC
Arithmetic Shift left SHLA
" " right SHRA


Input - Output Interface
I/O Bus
Data
Address
Control
CPU
Interface
Interface
keyboard
monitor


Input - Output Interface
I/O Bus
Data
Address
Control
CPU
Interface
Interface
keyboard
monitor
Buffer register
Status "
I/O Commands
1. status Command
2. Control "
3. Input "
4. Output "

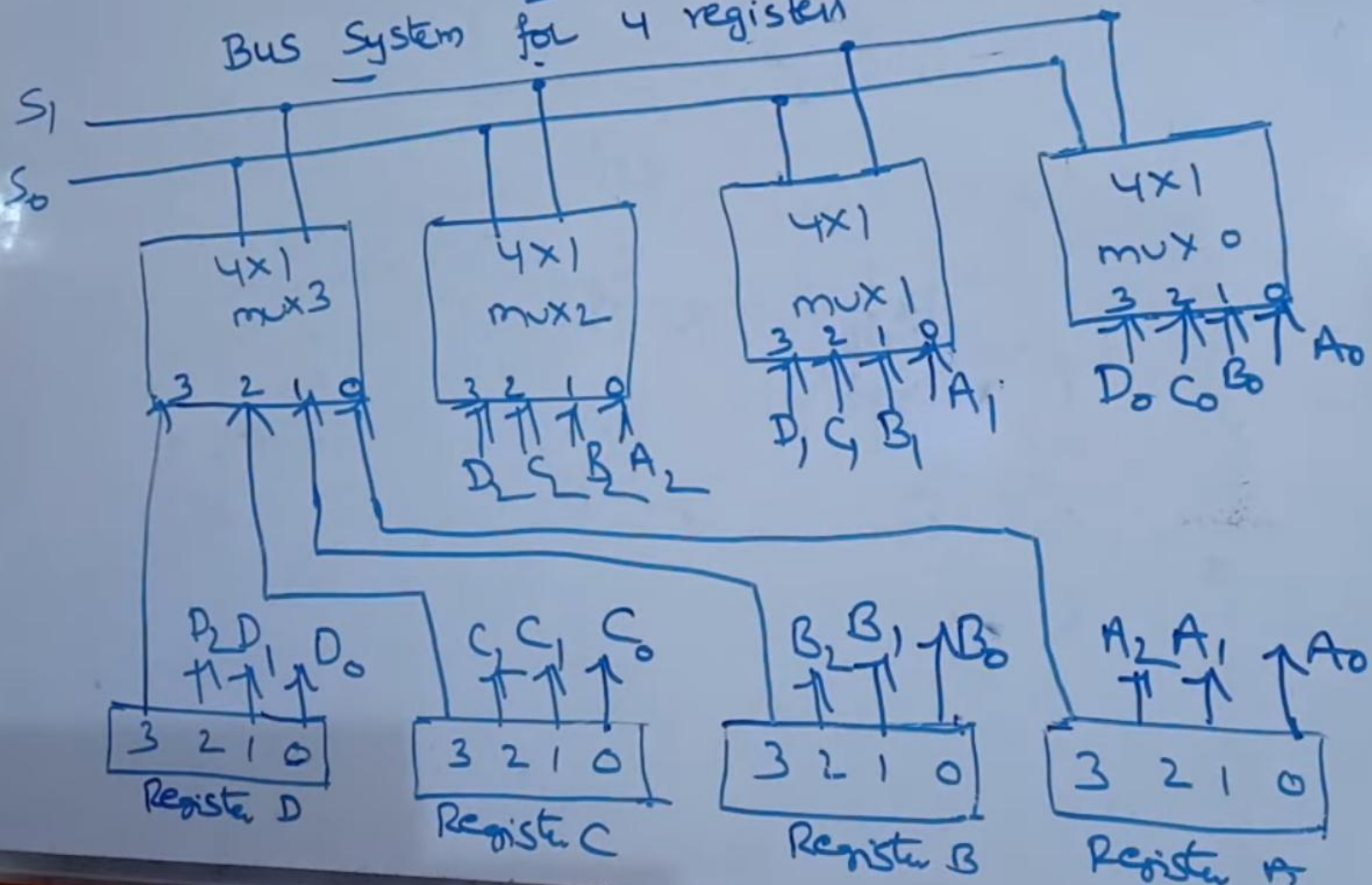

Input - Output Interface
I/O
Data
Addres
memory space
I/O Space
Read write
Read write
1. Isolated I/O
2. memory mapped I/O

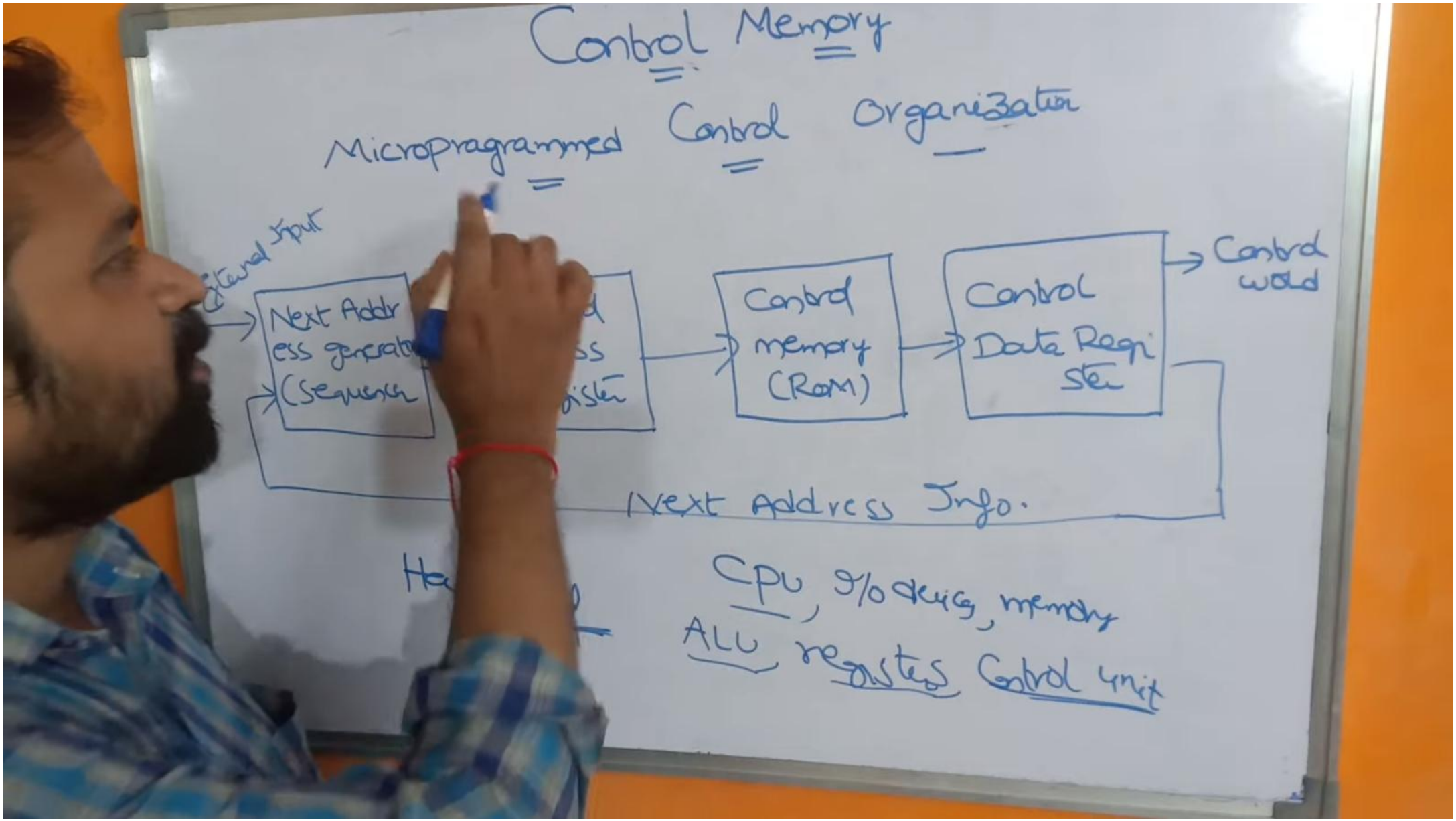

Modes of transfer
Programmed I/O
Read Sta
Check fl
=0
no
yes
th program
CPU
Data Bus
Interface
I/O Bus
Data Regist
I/O device
Data Valid
Address Bus
Status Regist
F
I/O Read
I/O write
Flag=1 or 0

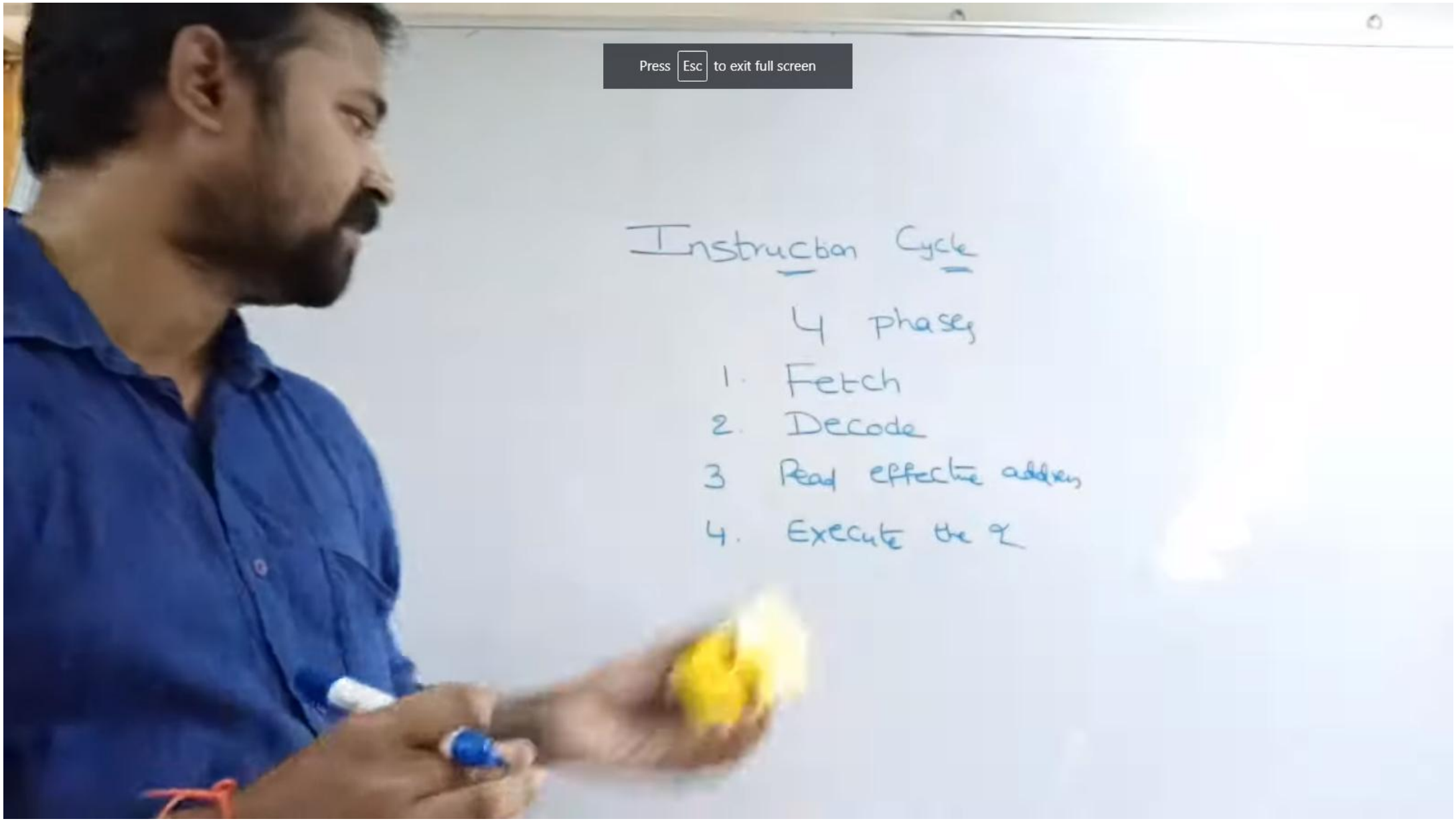

Modes of transfer
Programmed I/O
CPU
Read Status Register
Check flag bit
=0
F
Read
to memory
Operation Complete?
no
yes
ontinue with program
CPU
Data Bus
Interface
I/O Bus
Data Register
I/O device
Address Bus
Status Register
F
I/O Read
I/O write
Data Valid
Data Accepted
Flag=1 or 0

Bus and memory transfer

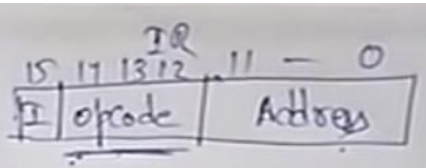
Bus System for 4 register




Control Memory
Microprogrammed Control Organization
External Input
Next Addr
ess generato
(Sequencer
Control
memory
(ROM)
Control
Data Regi
ster
Control
word
Next Address Info.
CPU, I/O devices, memory
ALU, registers, Control unit


Press Esc to exit full screen
Instruction Cycle
4 phases
1. Fetch
2. Decode
3 Read effective addres
4. Execute the

Instruction Cycle



Program :- → Memory unit

↓

Sequence of inst → IR

↓ Instruction cycle

Fetch → decode → Execute → decision

IR → 15
I=0 direct
I=1 indirect
↓
EA from mem

→ Step1: start SC ← 0

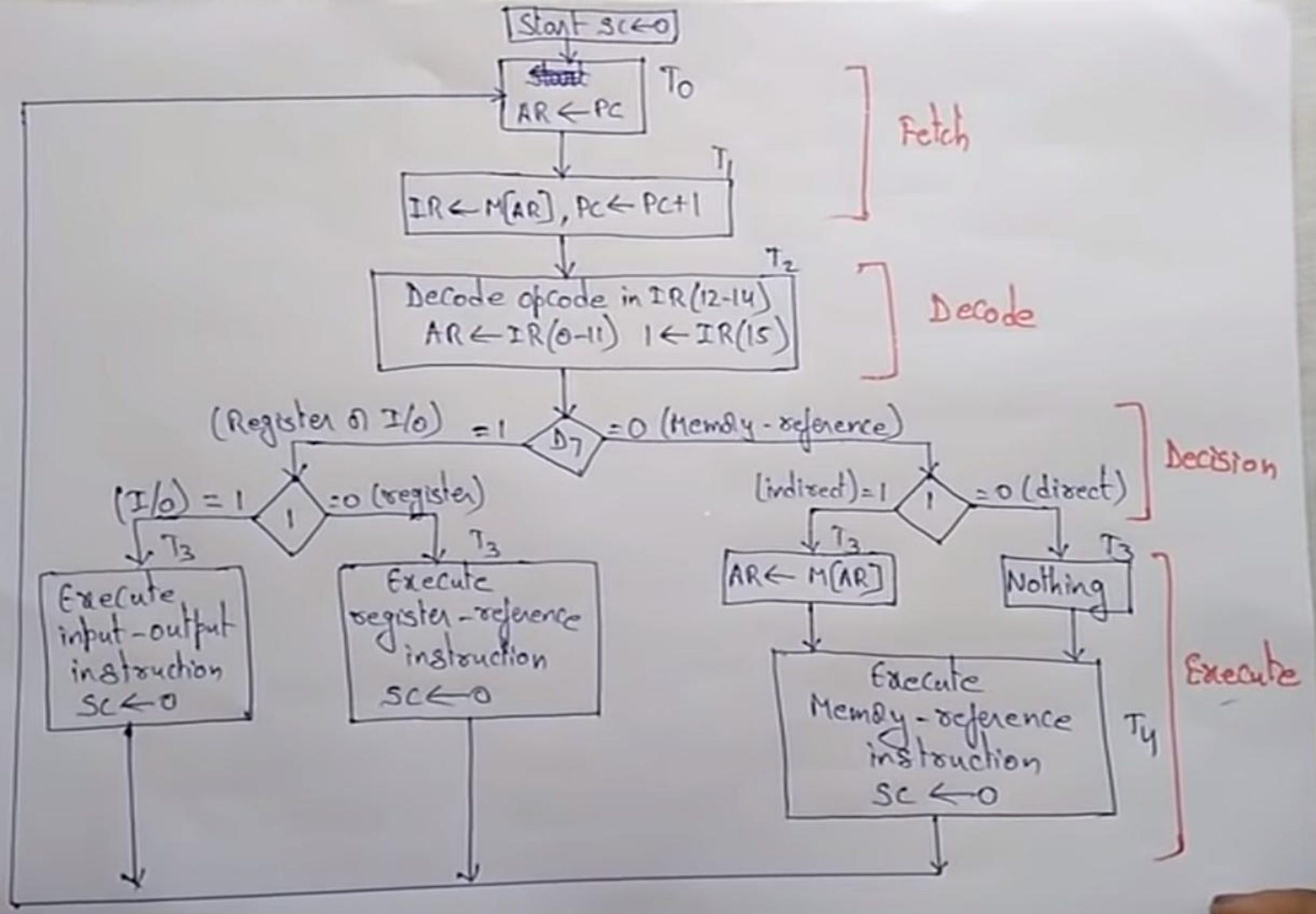
Step2: T_0 :- AR ← PC

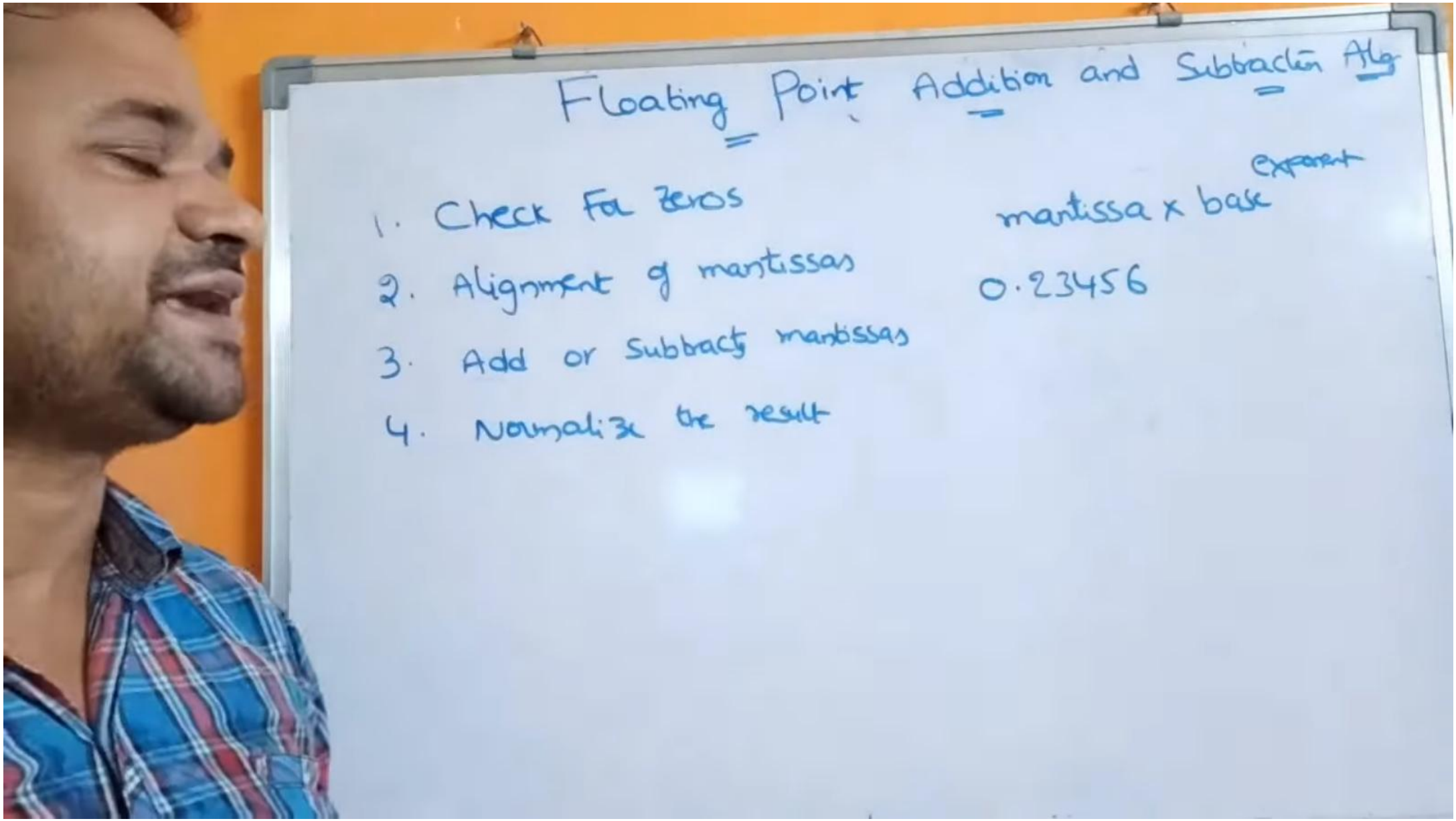
Step3: T_1 :- IR ← M[AR], PC ← PC+1 } fetch from memory

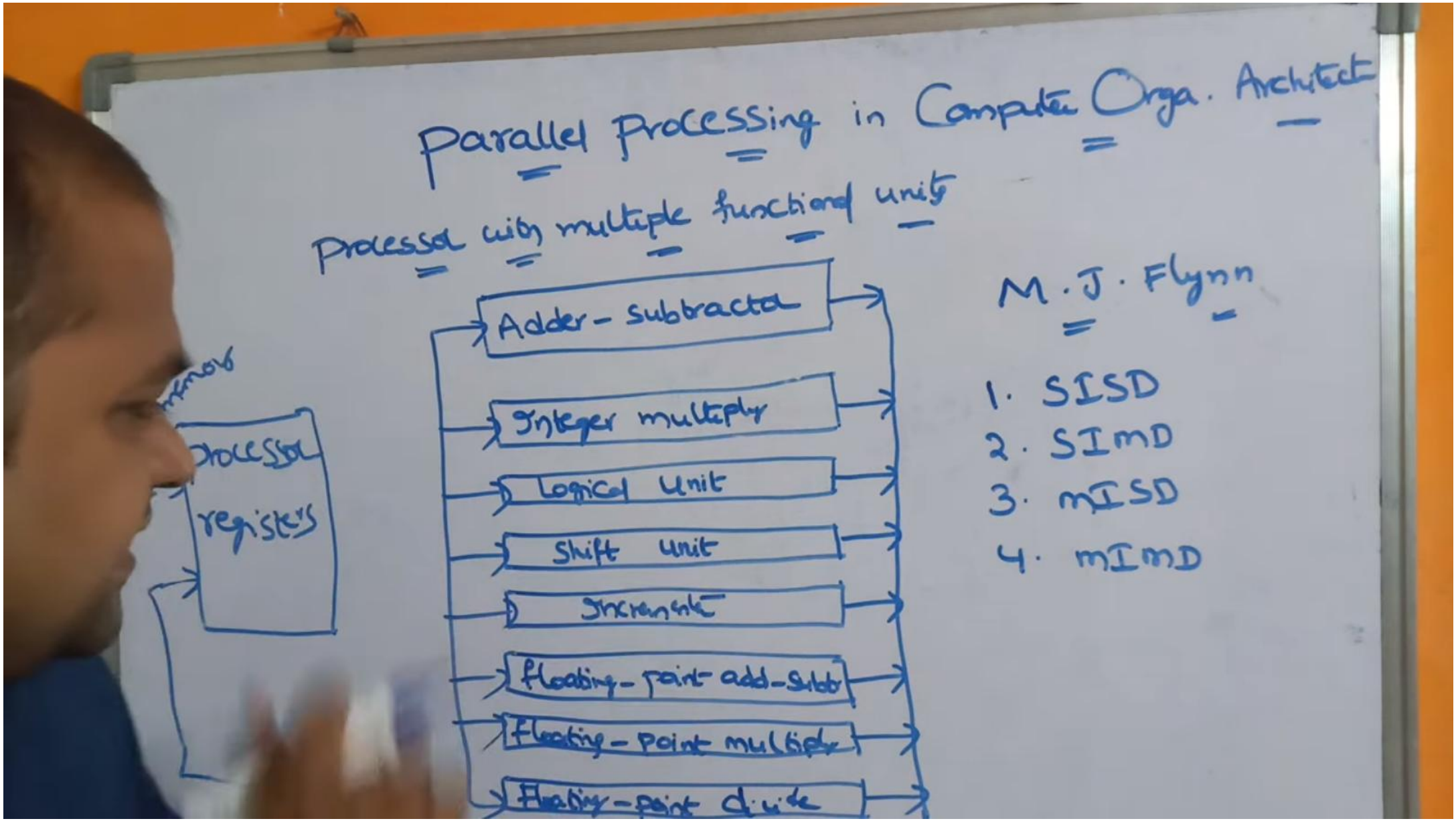
Step4: T_2 :- decode opcode IR (12-14)
AR ← IR(0-11), I ← IR(15) } decode

Step5: T_3 :- decision

Memory reference
Register reference Instruction
I/O reference


Start SC←0
AR←PC
T0
IR←M[AR], PC←PC+1
T1
Fetch
Decode opcode in IR(12-14)
AR←IR(0-11) I←IR(15)
T2
Decode
(Register or I/O) =1
D7
=0 (Memory-reference)
(I/O) = 1
I
=0 (register)
(indirect)=1
I
=0 (direct)
Decision
T3
Execute input-output instruction SC←0
Execute register-reference instruction SC←0
AR←M[AR]
Nothing
Execute Memory-reference instruction SC←0
T4
Execute


Floating Point Addition and Subtraction Alg
1. Check Fa Zeros
2. Alignment of mantissas
3. Add or Subtract mantissas
4. Normalize the result
mantissa x base exponent
0.23456


Parallel Processing in Compute Orga. Architect
Processor with multiple functional unit
Adder - subtractor
Integer multiply
Logical Unit
Shift Unit
Increment
Floating - point add-subb
Floating - point multiply
Floating - point divide
processor
registers
M. J. Flynn
1. SISD
2. SIMD
3. MISD
4. MIMD

Types of instructions

⇒ Most comp inst. can be classified into 3 categories

1. Data transfer inst
2. Data Manipulation inst
3. Program control inst.

into 3 Categories

① Data transfer Inst

② Data Manipulation inst

③ Program Control Inst

of data from
c to another
changing the
inf content

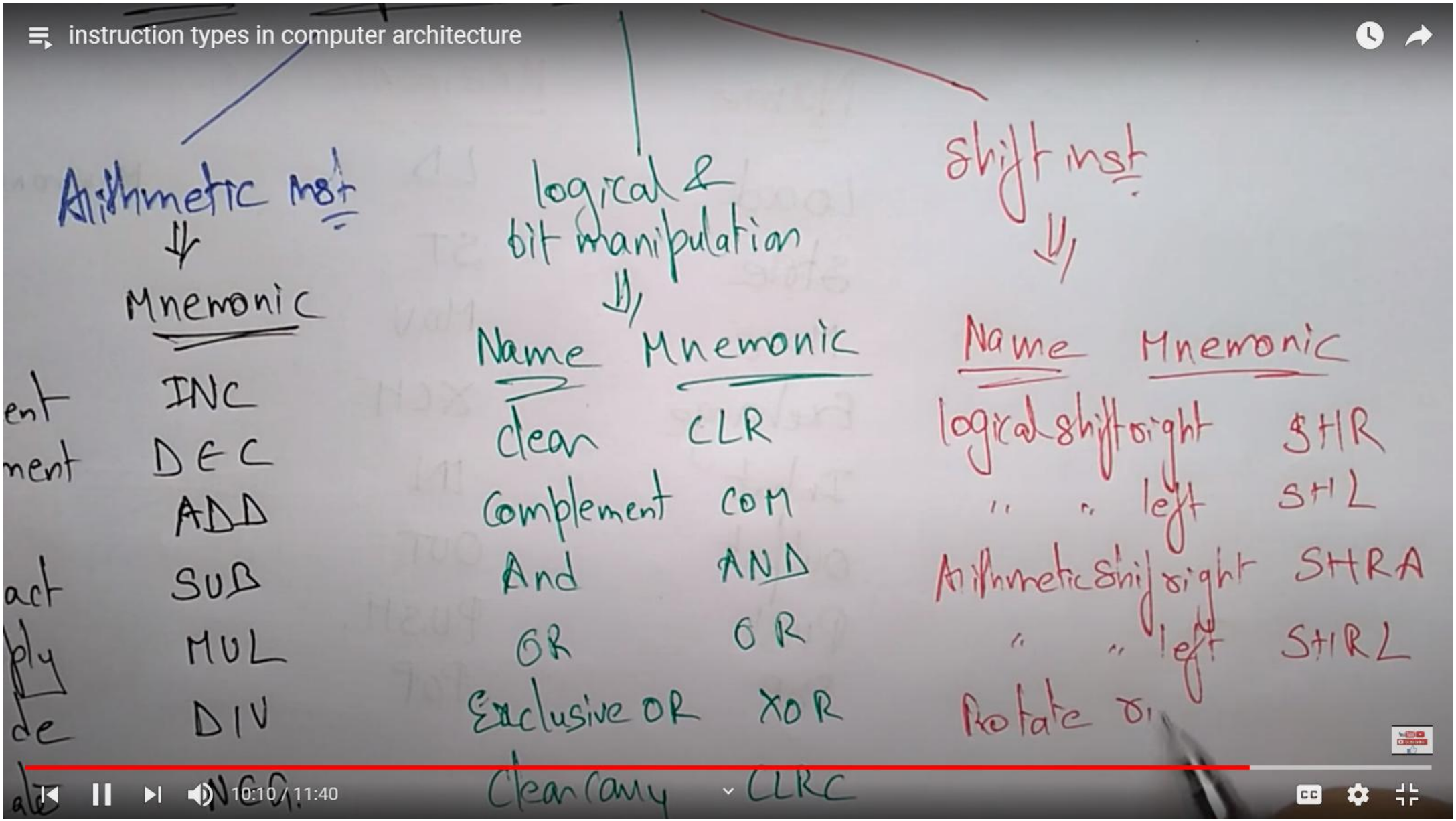
Are those that
perform arithmetic,
logic, shift-oper

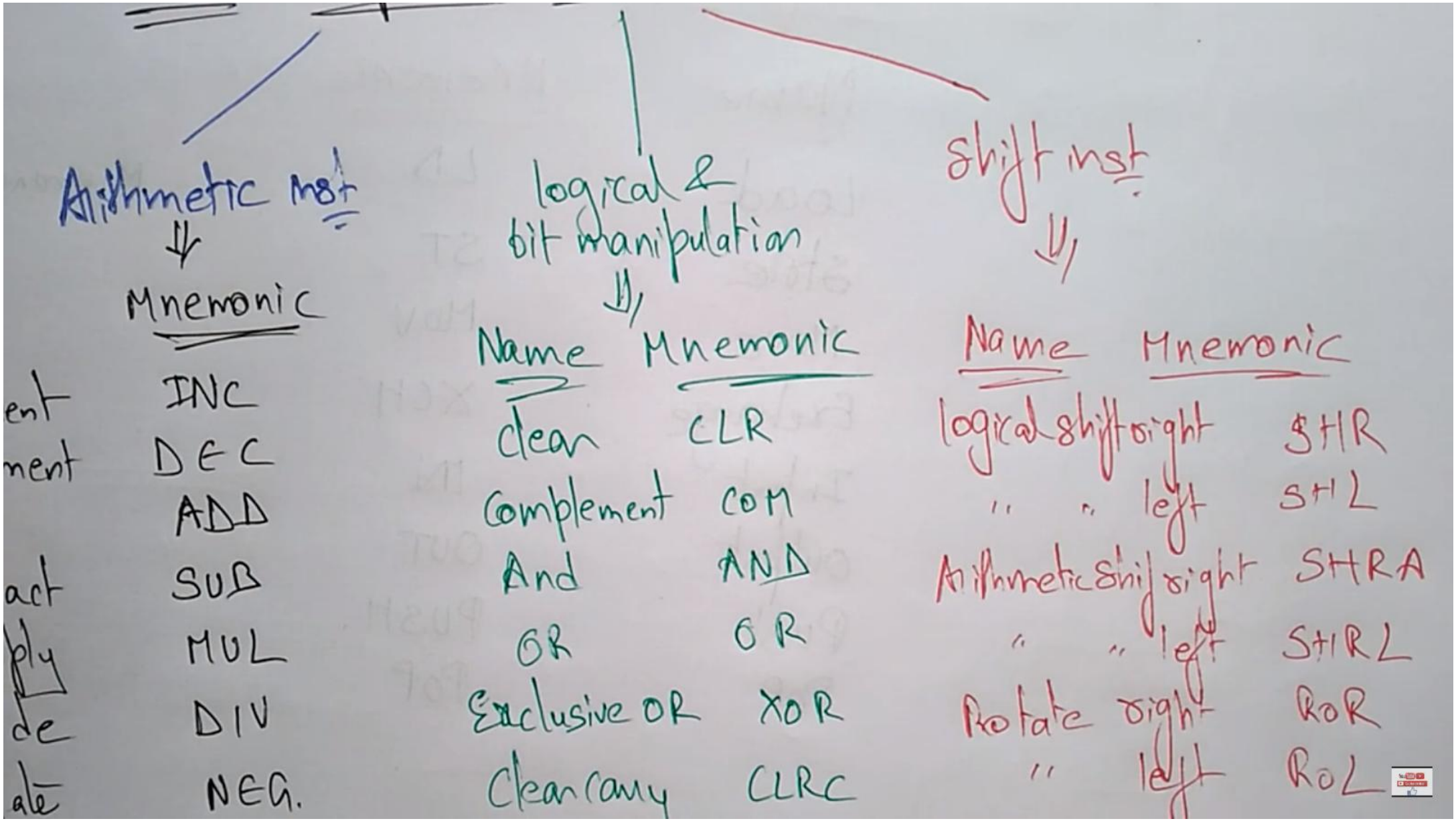
This inst provide decision
making capabilities &
change the path
taken by the prog
when executed in the

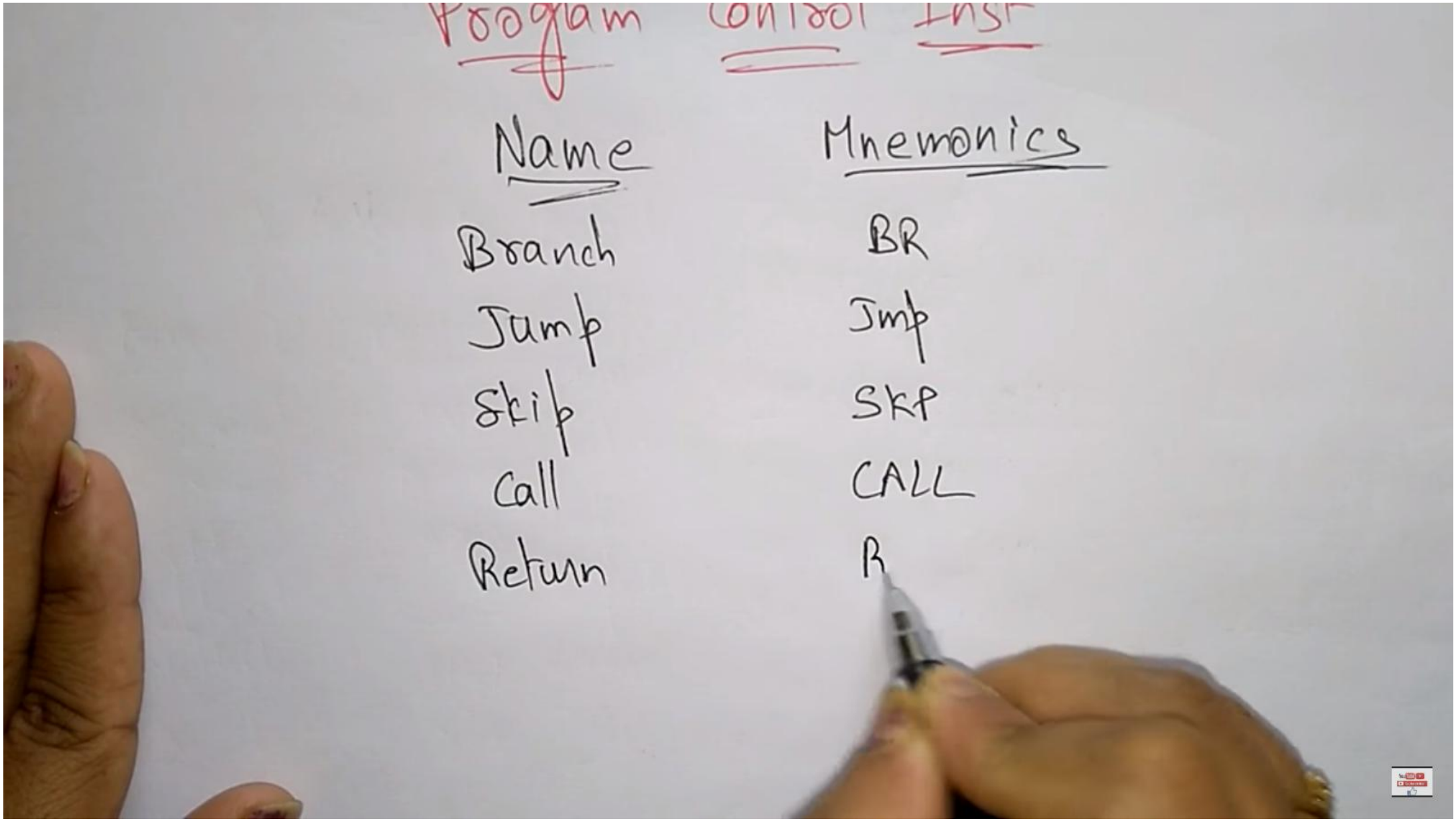
① Data transfer Inst

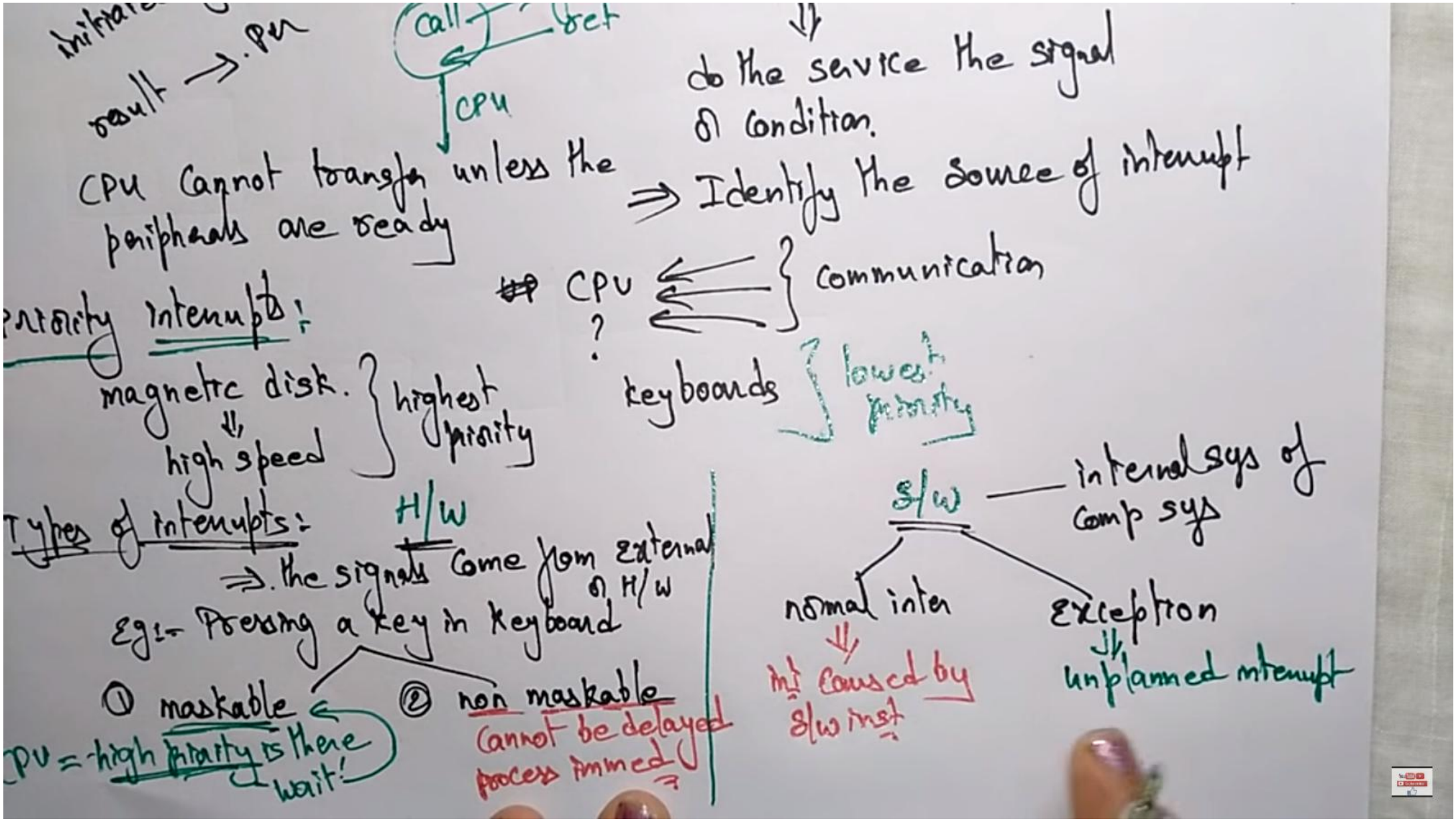
LD Ax, Bx.

Name	Mnemonic
Load	LD
store	ST
Move	MOV
Exchange	XCH
Input	IN
output	OUT
Push	PUSH


instruction types in computer architecture
Arithmetic inst
Mnemonic
ent INC
ment DEC
ADD
act SUB
ply MUL
de DIV
ate NEG
logical &
bit manipulation
Name Mnemonic
clear CLR
Complement COM
And AND
OR OR
Exclusive OR XOR
Clear Carry CLRC
shift inst
Name Mnemonic
logical shift right SHR
" " left SHL
Arithmetic shift right SHRA
" " left SHRL
Rotate ri
10:10 / 11:40


Arithmetic Inst
Mnemonic
ent INC
ment DEC
ADD
act SUB
ply MUL
de DIV
ate NEG.
logical & bit manipulation
Name Mnemonic
clear CLR
Complement COM
And AND
OR OR
Exclusive OR XOR
Clear carry CLRC
shift inst
Name Mnemonic
logical shift right SHR
" " left SHL
Arithmetic shi right SHRA
" " left SHRL
Rotate right ROR
" left ROL


Program Control Inst
Name
Mnemonics
Branch
BR
Jump
Jmp
Skip
SKP
Call
CALL
Return
R


default → per
call
CPU
CPU cannot transfer unless the peripherals are ready
do the service the signal of condition.
⇒ Identify the source of interrupt
CPU ? communication
priority interrupts :-
magnetic disk. highest priority
high speed
keyboards lowest priority
Types of interrupts :-
H/W
⇒ the signals come from external of H/W
Eg :- Pressing a key in keyboard
① maskable
② non maskable
cannot be delayed
process immed
CPU = high priority is there wait!
S/W — internal sys of comp sys
normal inter
exception
int caused by S/W inst
unplanned interrupt

M.M.

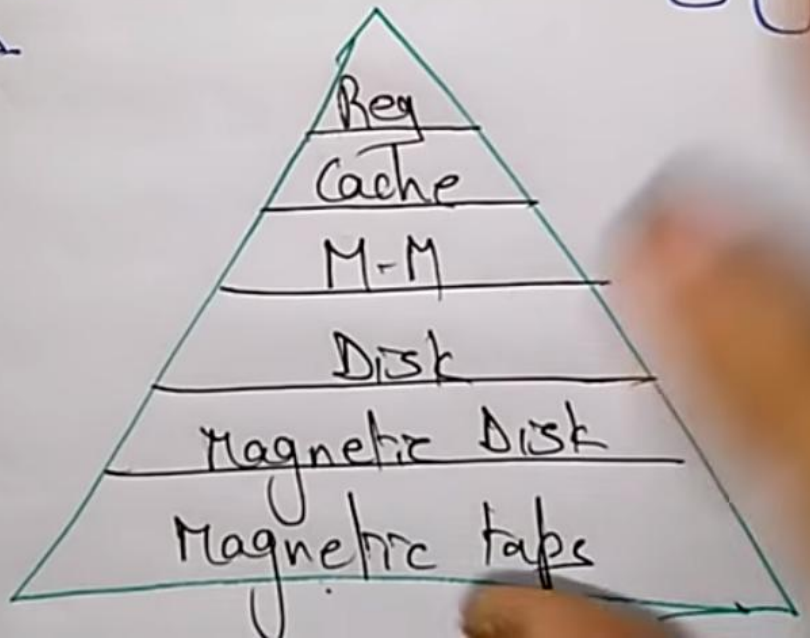
It is the memory which is very nearest to the CPU, all recent inst. are stored into cache memory.

⇒ Cache Memory is attached for storing the c/p which is given by the user & which is necessary for the CPU to perform a task.

① Memory Hierarchy :

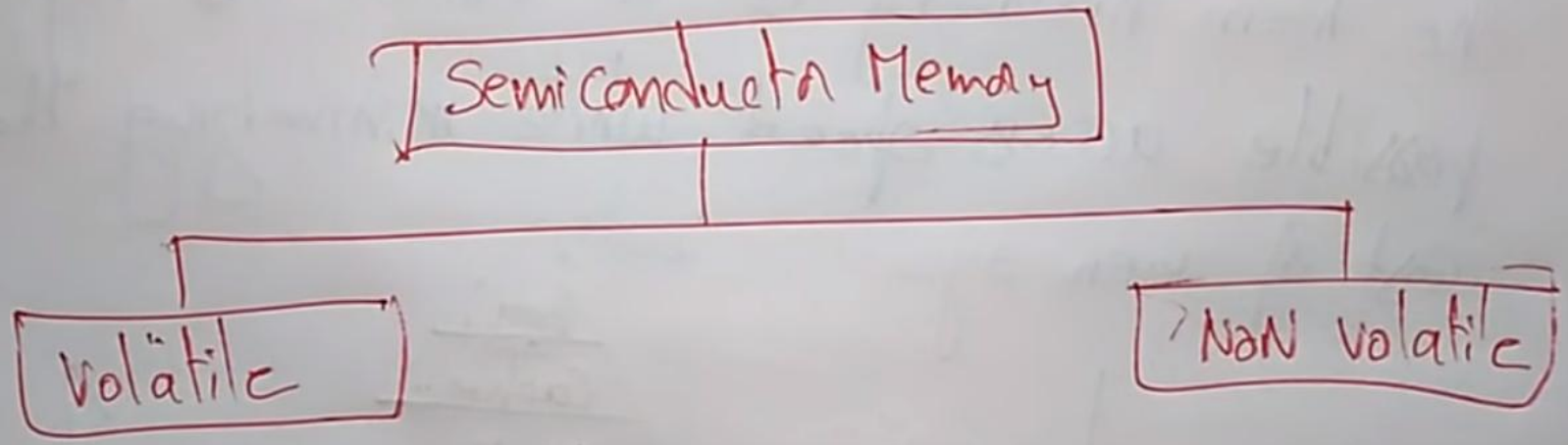
Comp mem should be fast, large & inexpensive

The mem hierarchy is to obtain the highest possible access speed while minimizing the total cost of mem sys



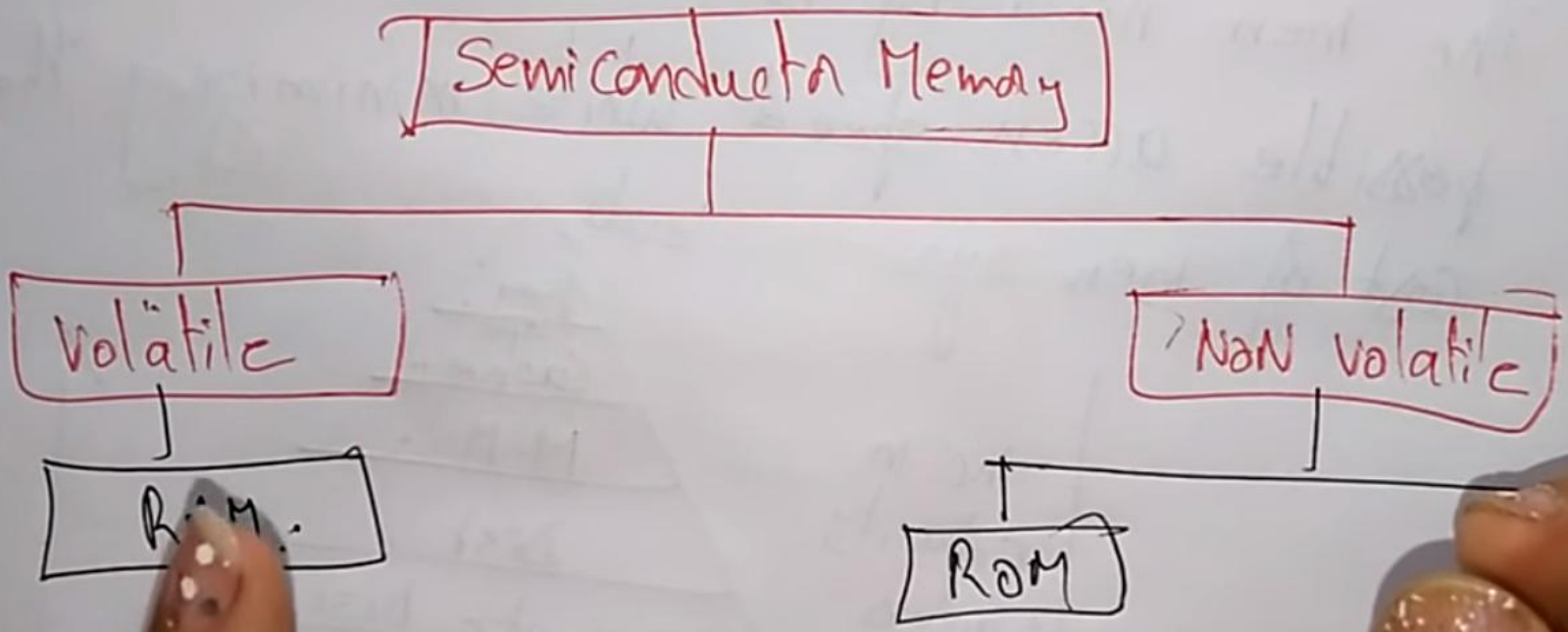
Main Memory :- It stores data & program during Computer oper.

⇒ It uses Semiconductor technology hence known as semiconductor memory.



computer system.

=> It uses Semiconductor technology hence known as semiconductor memory.



CISC

- ⇒ Emphasis on Hardware
- ⇒ Multiple clock cycle
- ⇒ Not a less pipelined
- ⇒ memory to memory
- ⇒ Variable format instruction
- ⇒ Any inst may refer memory

RISC

- ⇒ Emphasis on software
- ⇒ Single clock cycle
- ⇒ Highly pipelined
- ⇒ Register to Register
- ⇒ Fixed format instruction

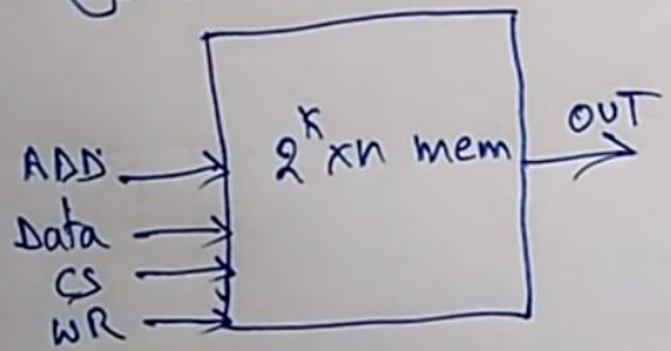
Memory Cells - SRAM and DRAM Cells

RAM is a volatile mem

⇓ turned on again

BIOS ⇒ reads ur OS & load the related files from hard disk back into RAM.

block dig of RAM :



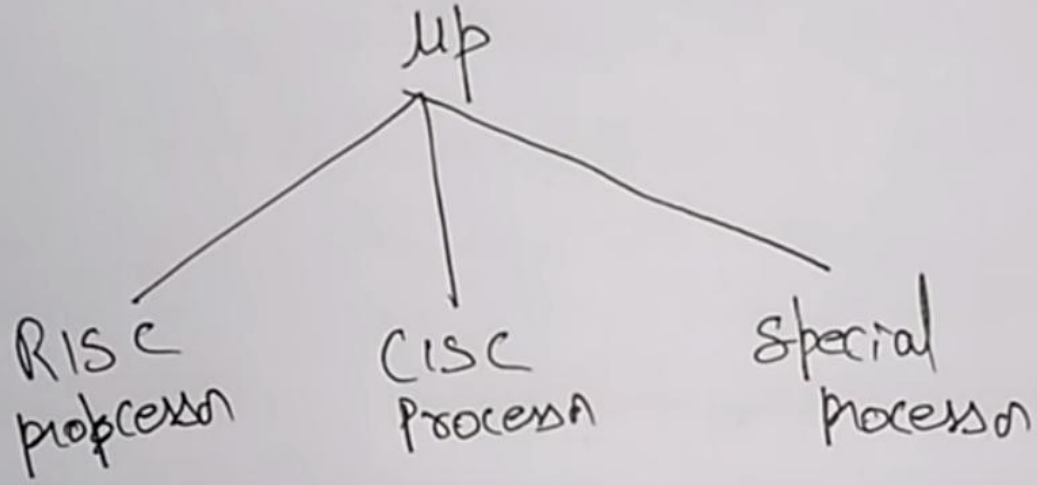
CS	WR	mem operation
0	X	None
1	0	Read selected word
1	1	Write selected word

SRAM & DRAM

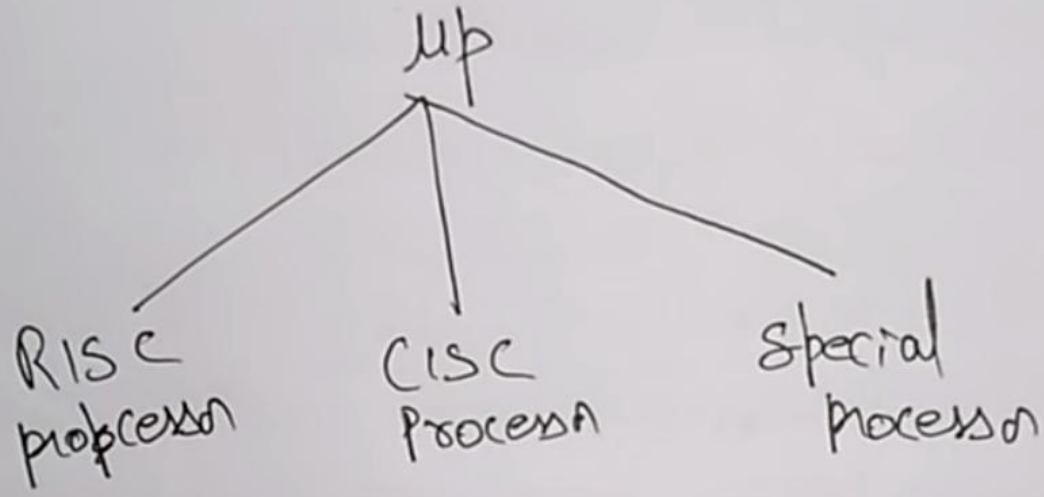
- ⇒ Semiconductor mem
  Each bit will stored
- ⇒ doesn't need refreshing
- ⇒ retains content as long as power applied
- ⇒ ion sec
- ⇒ Cache mem &
  fn date & time settings

- ⇒ Mos & a capictor for basic storage
- ⇒ refreshing required

RISC (Reduced Instruction Set Computer)



RISC:- It is designed to reduce the execution time by simplifying the inst. set comp.



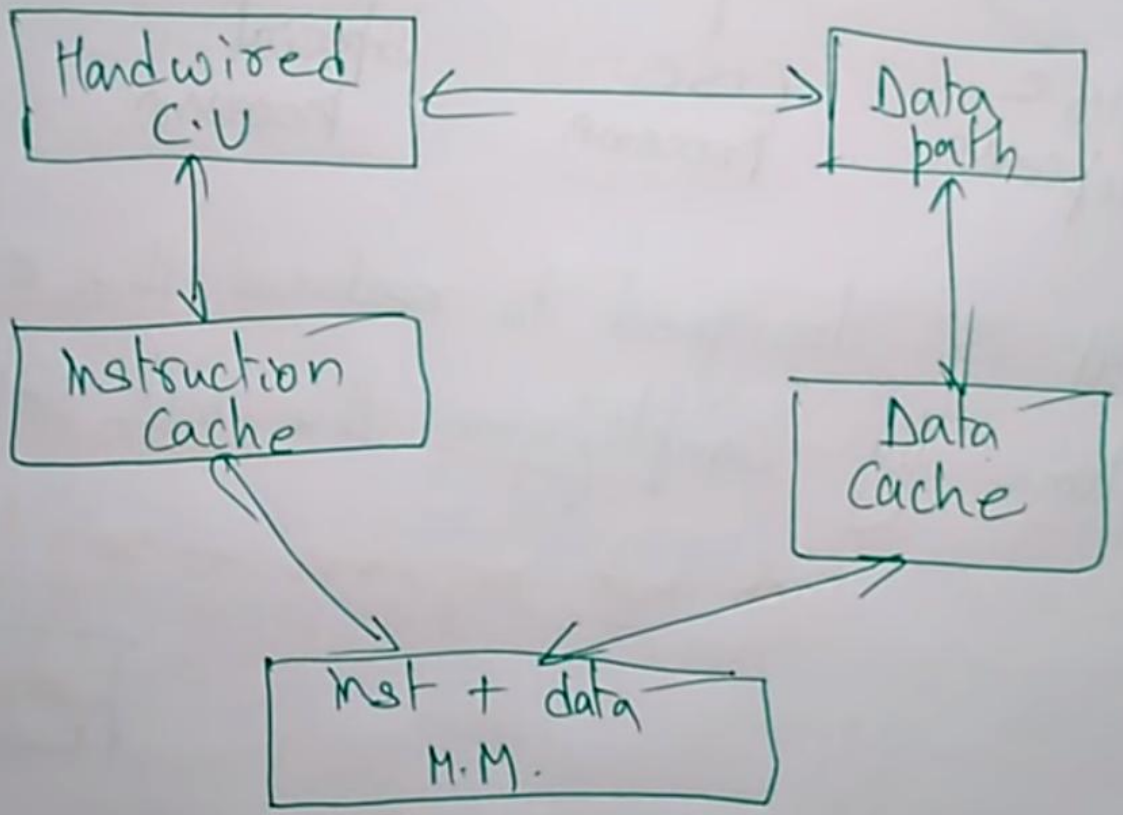
RISC:- It is designed to reduce the execution time by simplifying the inst set comp.

Each inst requires — only 1 clock cycle.

↓

fetch, decode, execute

It used in portable devices like Apple ipod, smartphones & tablets.



Characteristics of RISC

⇒ It consists of simple inst

⇒ It supports various data type formats

⇒ It utilizes simple addressing modes

⇒ one cycle execution time

⇒ LOAD & STORE inst are used to access the mem loc.

Complex Inst set Comp.

⇓

⇒ To minimize the no: of inst per prog
ignoring the no: of cycles per inst

⇒ Desktop & laptop comp

⇒ Very little RAM is req to store inst.

Architecture :-

Reduced Inst Set

⇓

1 inst ⇒ 1 clock cy

⇒ tablets, smartpho