COMPUTER SCIENCE

Computer Organization and Architecture

ALU & Control unit

Lecture_03

Vijay Agarwal sir


TOPICS
TO BE
COVERED
01 Micro Operation
02 Control Unit

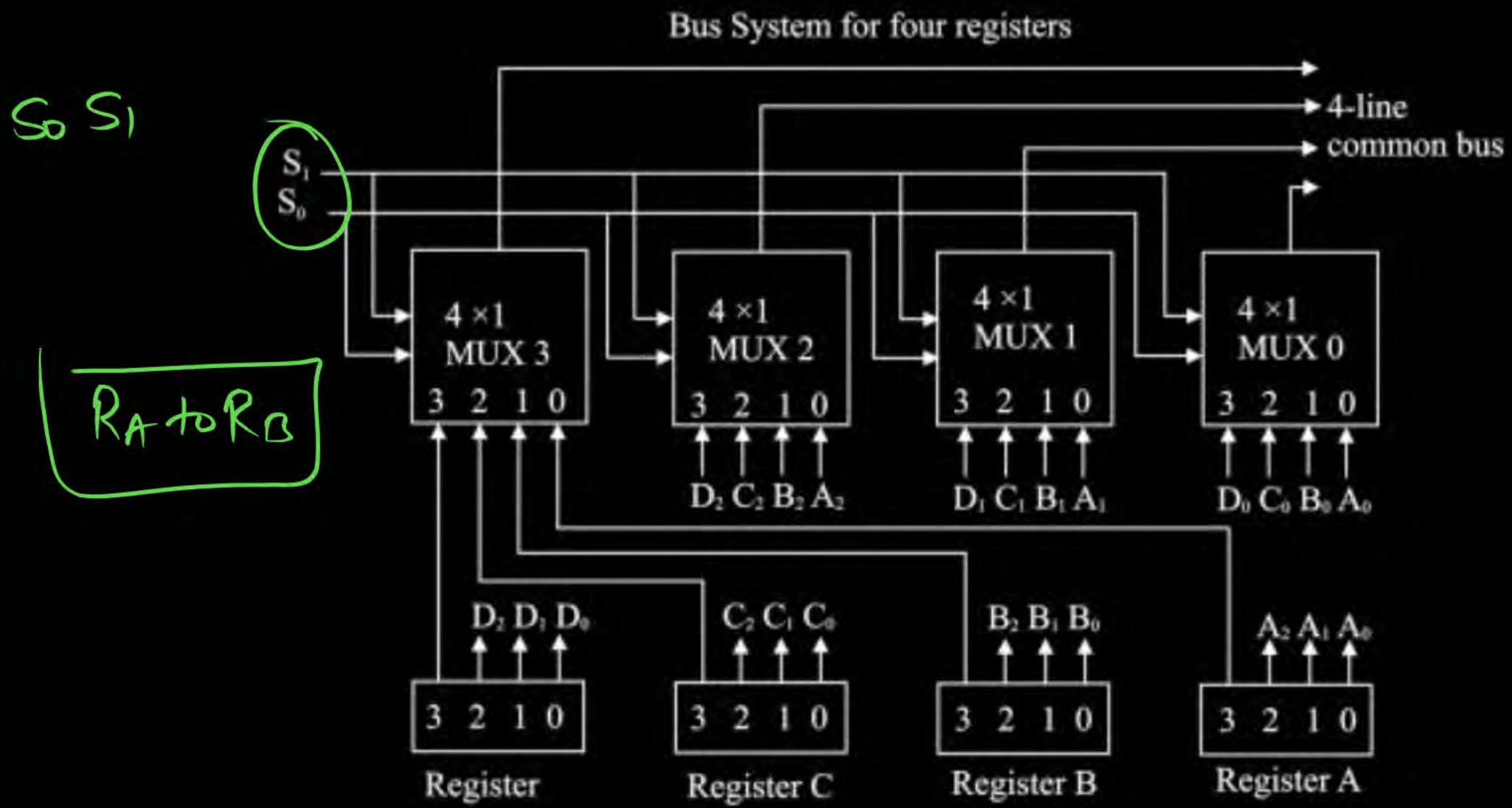
Micro operation

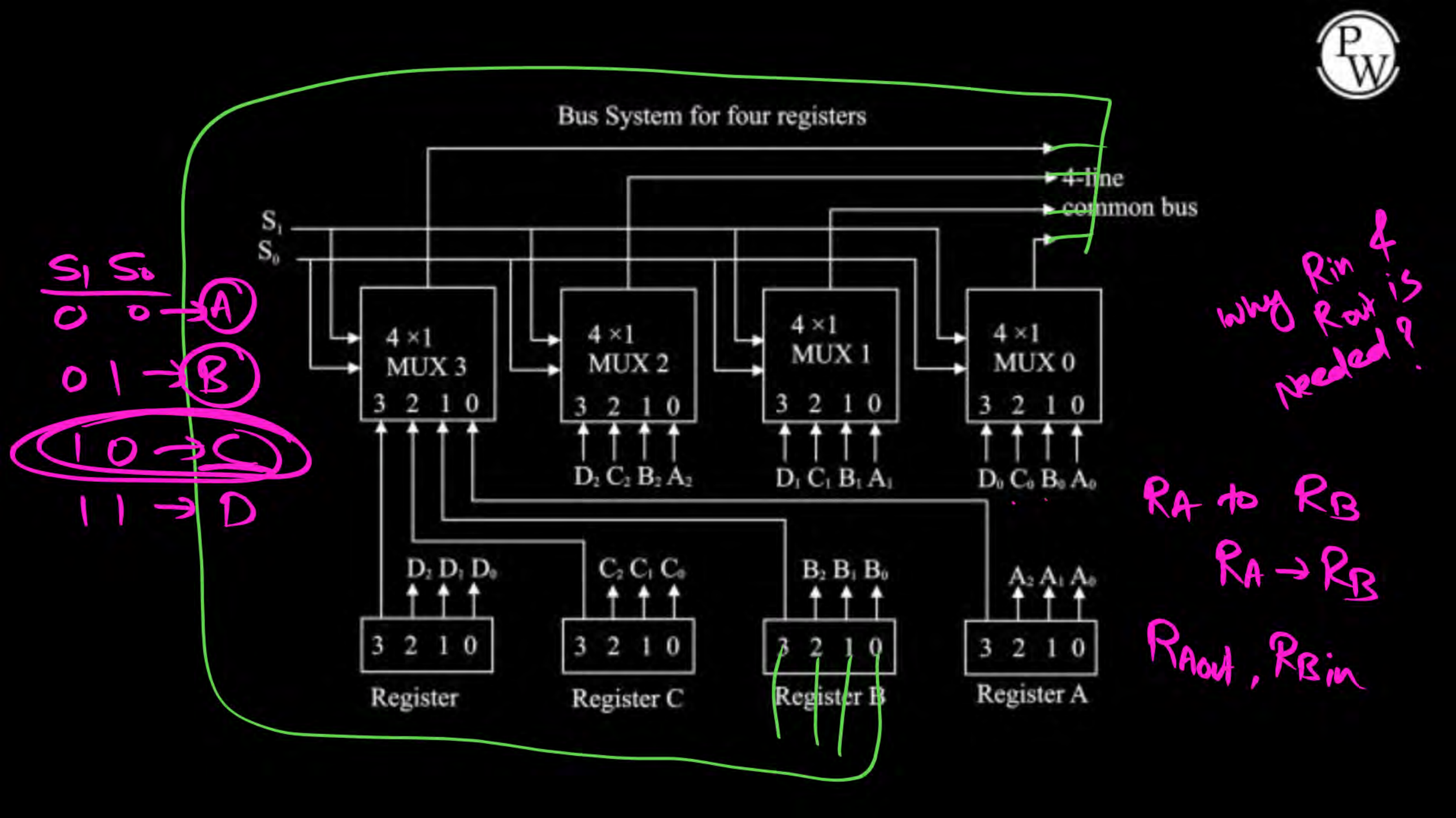
Working of MUX

Common Bus (WHY Need of Common Bus ?)

How the Data Transfer from One Register to Another Register

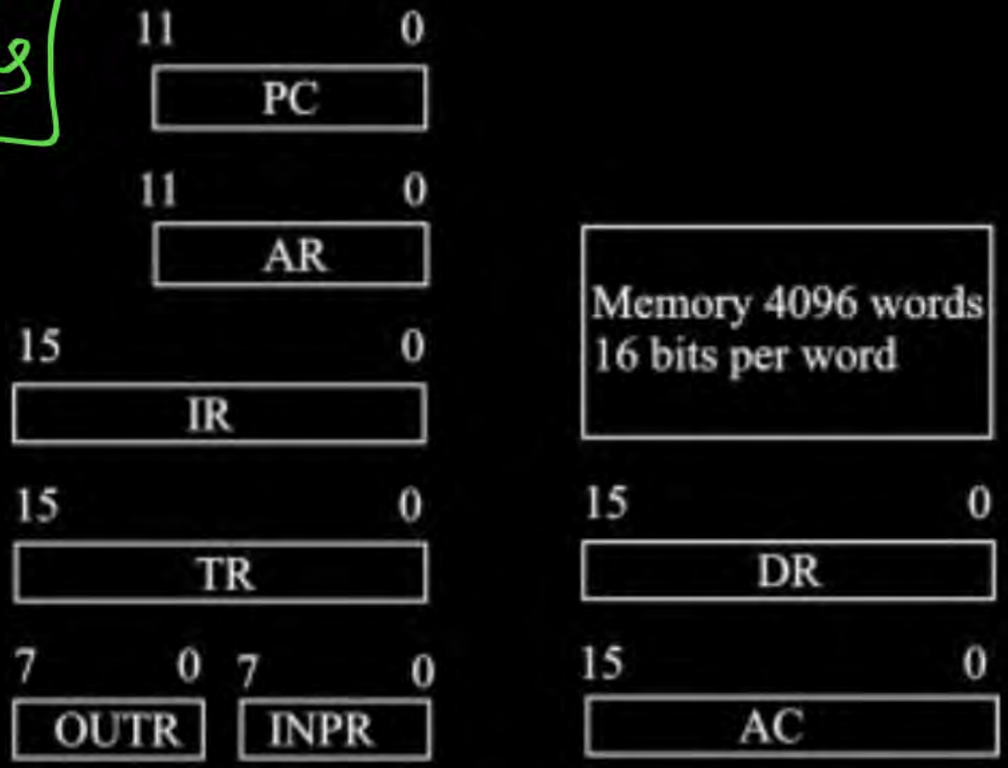
Working of Computer.


Bus System for four registers
S0 S1
S1
S0
4-line
common bus
4 ×1
MUX 3
3 2 1 0
4 ×1
MUX 2
3 2 1 0
4 ×1
MUX 1
3 2 1 0
4 ×1
MUX 0
3 2 1 0
RA to RB
D2 C2 B2 A2
D1 C1 B1 A1
D0 C0 B0 A0
D2 D1 D0
C2 C1 C0
B2 B1 B0
A2 A1 A0
3 2 1 0
3 2 1 0
3 2 1 0
3 2 1 0
Register
Register C
Register B
Register A

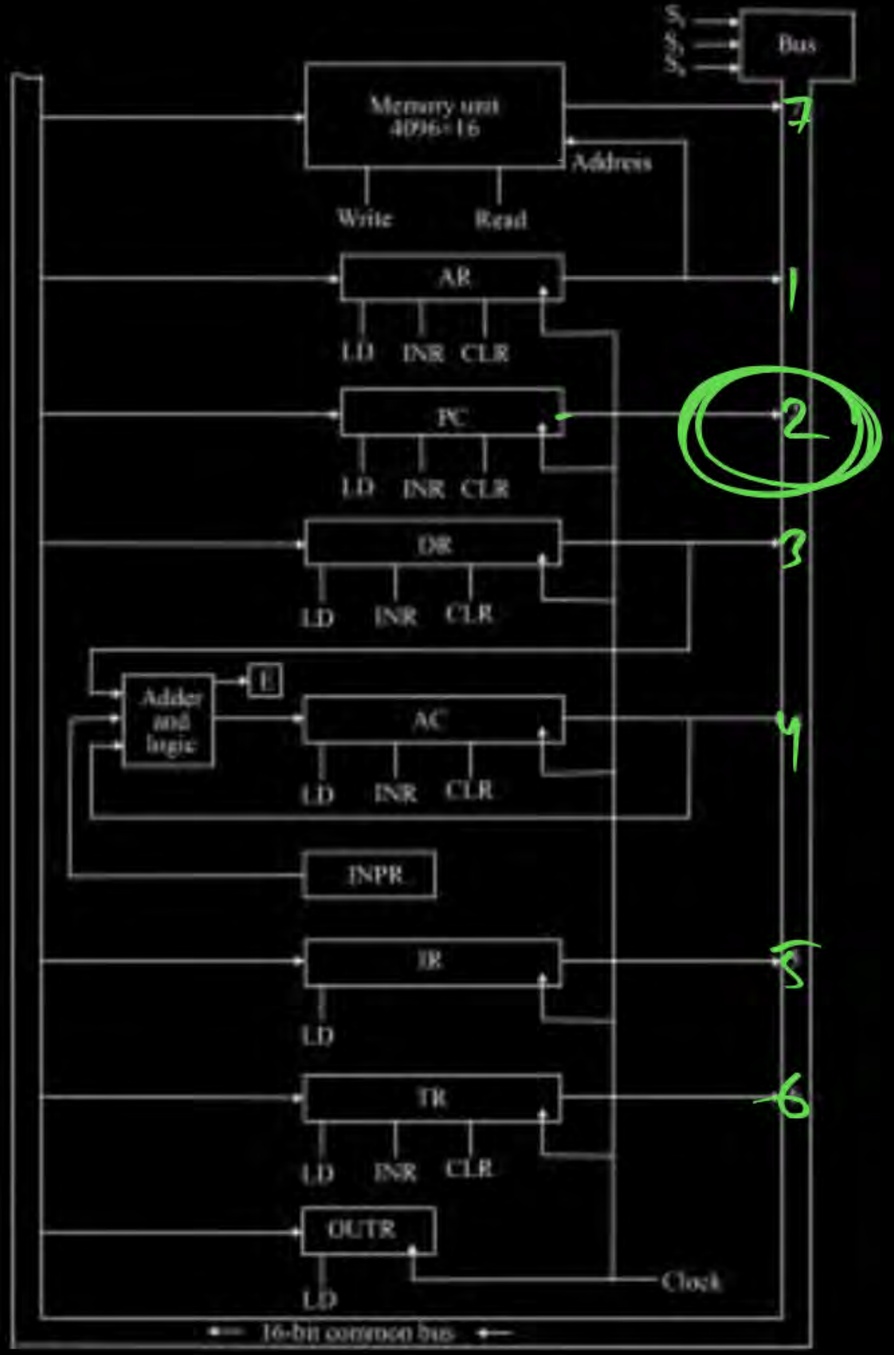

Bus System for four registers
4-line common bus
S_1
S_0
4 ×1 MUX 3
4 ×1 MUX 2
4 ×1 MUX 1
4 ×1 MUX 0
3 2 1 0
D_2 C_2 B_2 A_2
D_1 C_1 B_1 A_1
D_0 C_0 B_0 A_0
D_2 D_1 D_0
C_2 C_1 C_0
B_2 B_1 B_0
A_2 A_1 A_0
Register
Register C
Register B
Register A
S_1 S_0
0 0 → A
0 1 → B
1 0 → C
1 1 → D
Why Rin & Rout is needed?
R_A to R_B
$R_A \rightarrow R_B$
R_{Aout}, R_{Bin}

Design a Small Computer

4096 X 16 words



Basic computer registers and memory



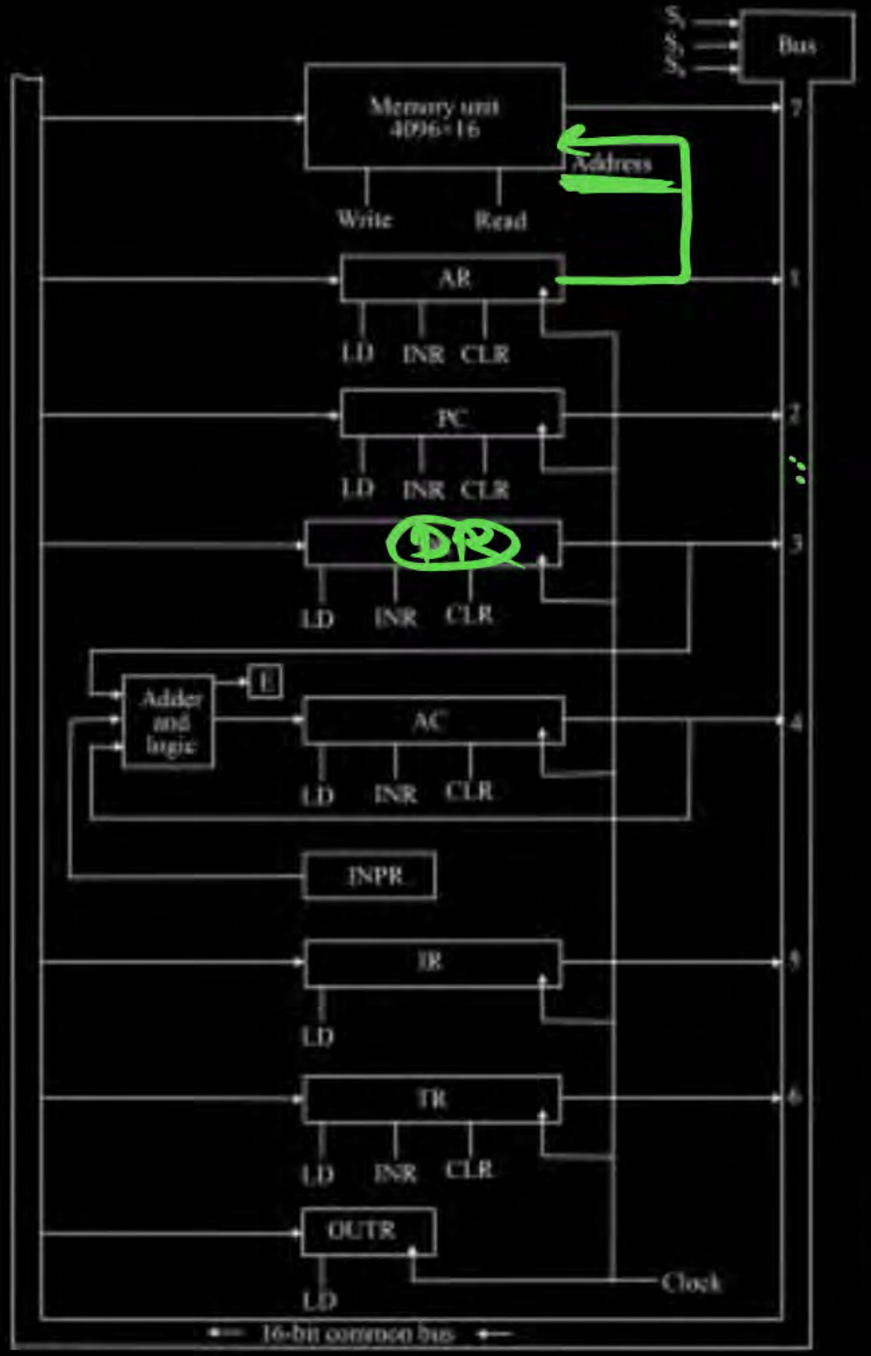
Basic computer registers connected to a common bus

How Computer is Working

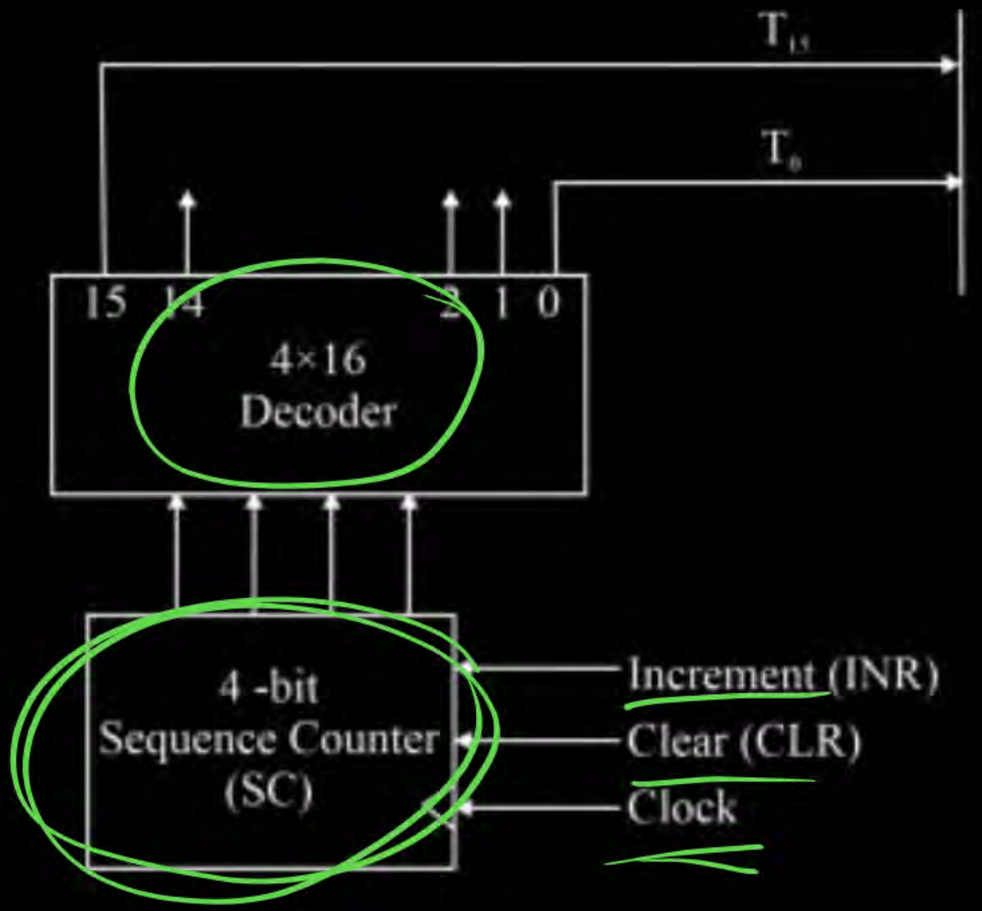
S_2	S_1	S_0	
0	1	0 → 2	[PC] PC → MAR

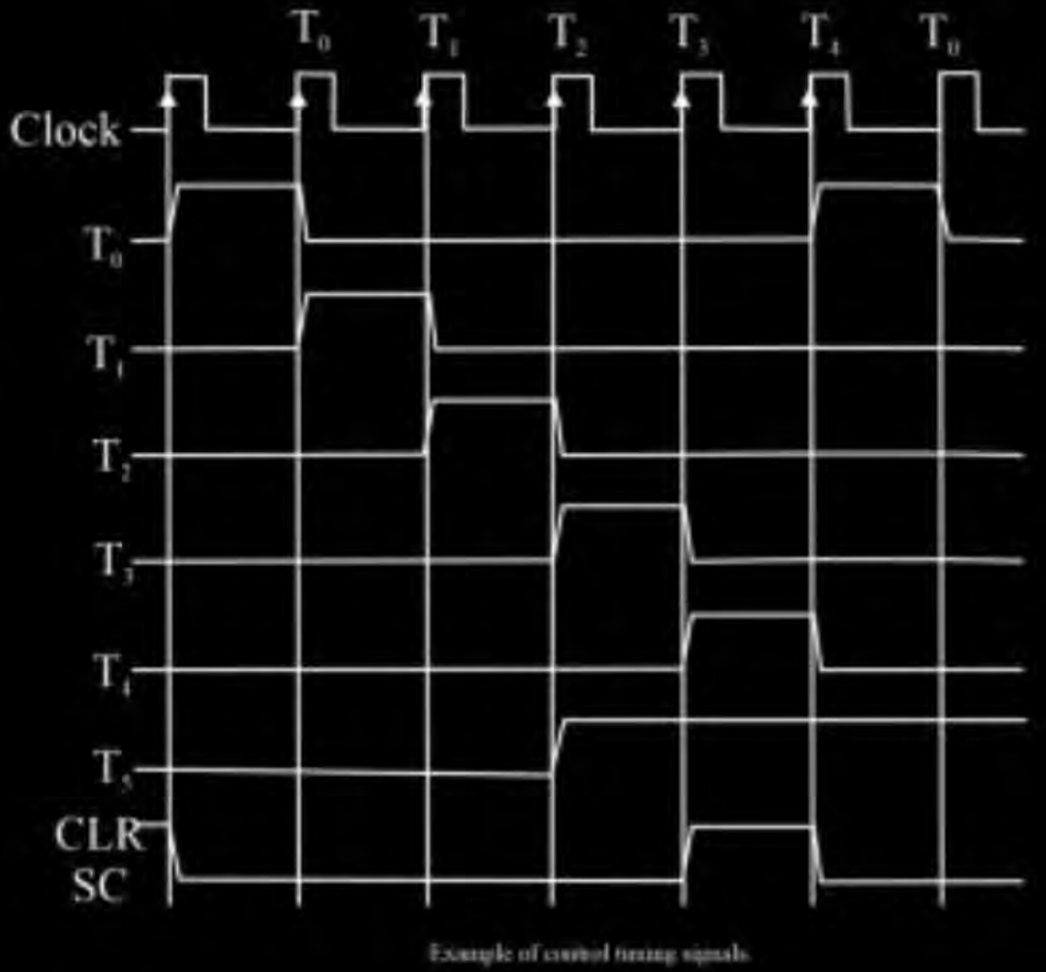
PC_{out} Set to 1

$MAR_{in} = 1$



Basic computer registers connected to a common bus





Example of control timing signals

How Data is Transferred ?

Register A to Register B

WHY Rout & Rin is Need ?

Register output is Connected to MUX
then from MUX to Common BUS,
then Common BUS to Respective Register.

When Rout is Set to 1 then that Respective Register Data Load into MUX then MUX to Common BUS, then Common is Connected to ALL Registers, the Register which have Rin is Set to 1 in that Register BUS Data is Loaded.

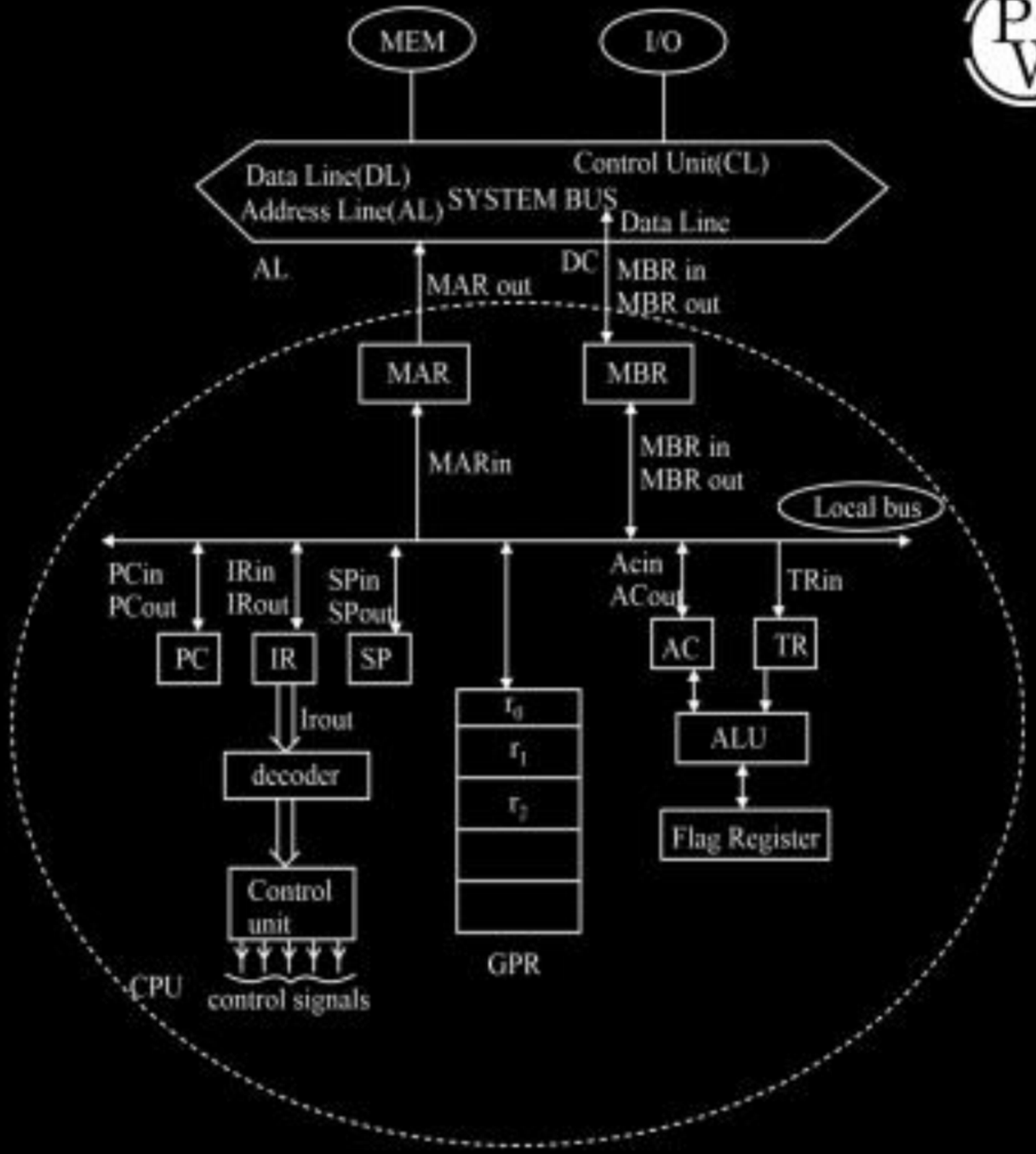
(eg) R_A to R_B ; R_{Aout} R_{Bin}

$R_{Aout} = 1$

then R_A DATA Load to MUX then Common Bus,

& Now we get Control Signal $R_{Bin} = 1$ then from the Common Bus that Data Loaded in Register R_B.

Structure of Computer



Component of Computer

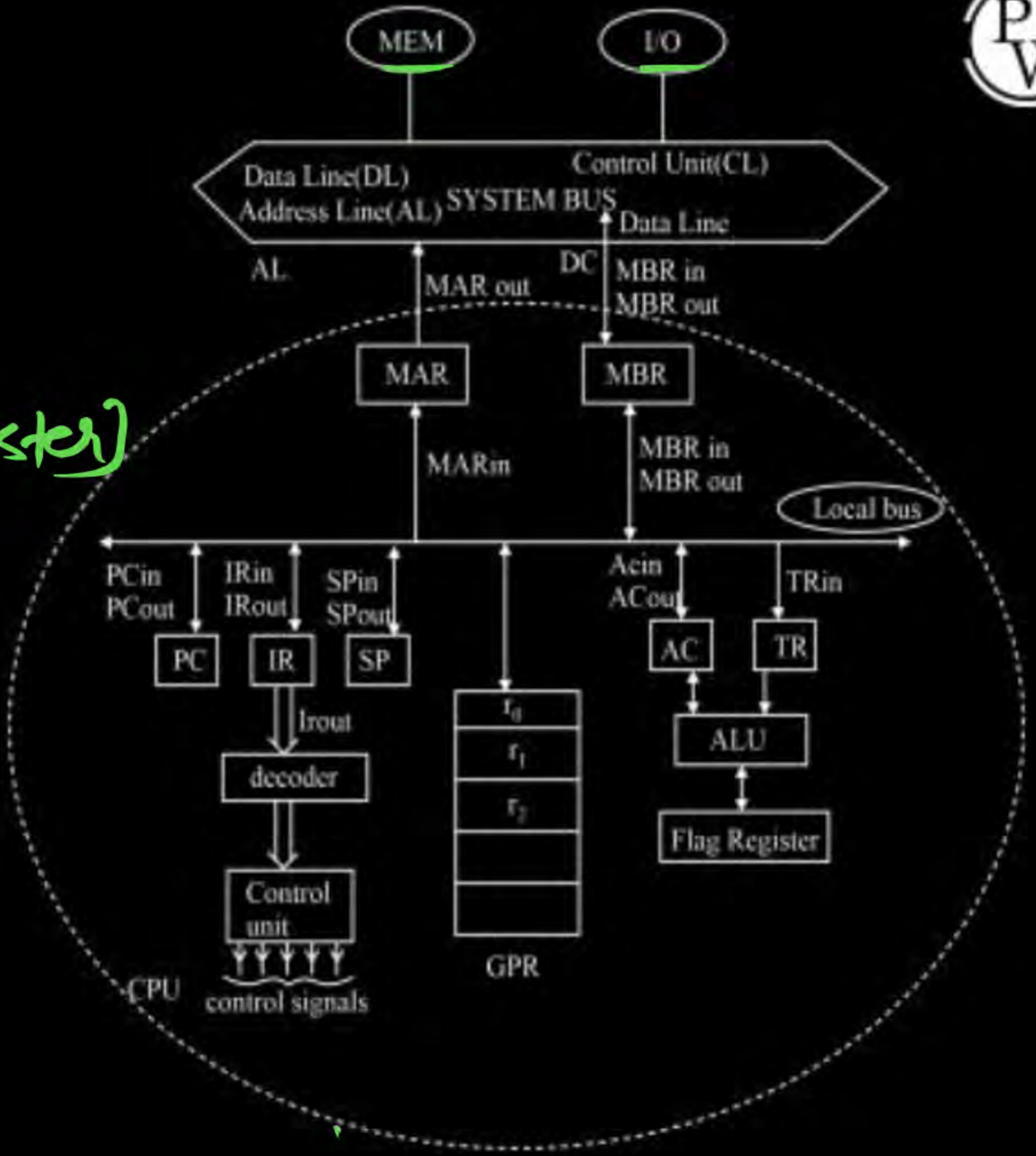
1. CPU , 2. Memory & 3. IO

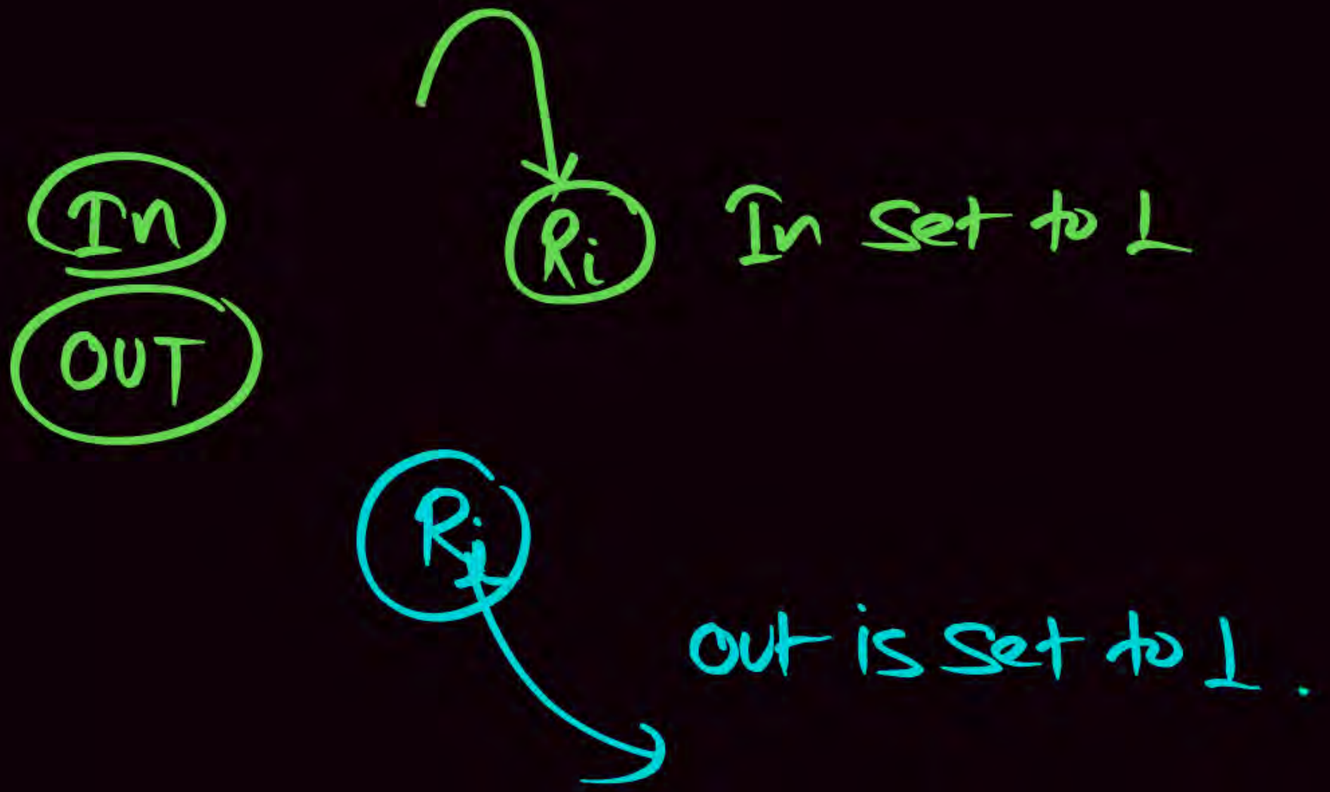
Memory
ALU
Register
Timing Signals, Control signals
Flags(PSW)

Structure of Computer

1. Memory
2. CPU → (ALU, CU, Register)
3. I/O

- PC
- MAR
- MBR
- IR
- AC
- SP
- PSW (Flag Register)




In
OUT
Ri
In set to 1
Ri
Out is set to 1.

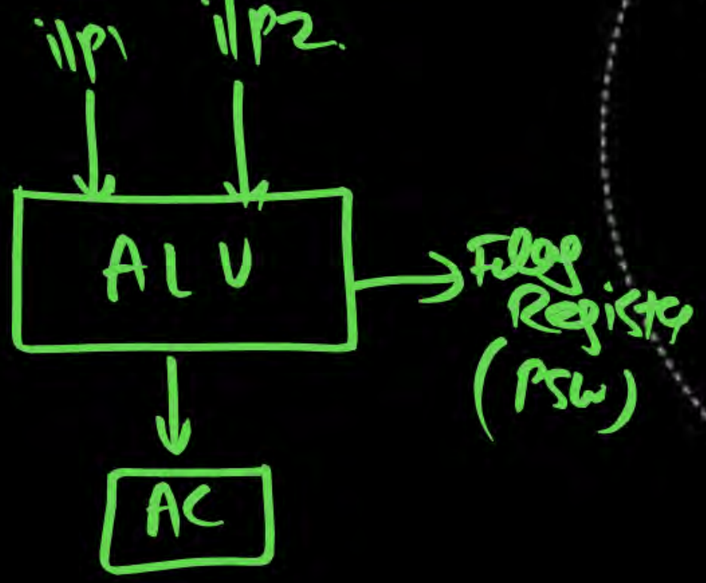
MAR | AR : (Memory Address Register) : it Contain the Address of memory locations. it Connected to Address line of the System BUS.

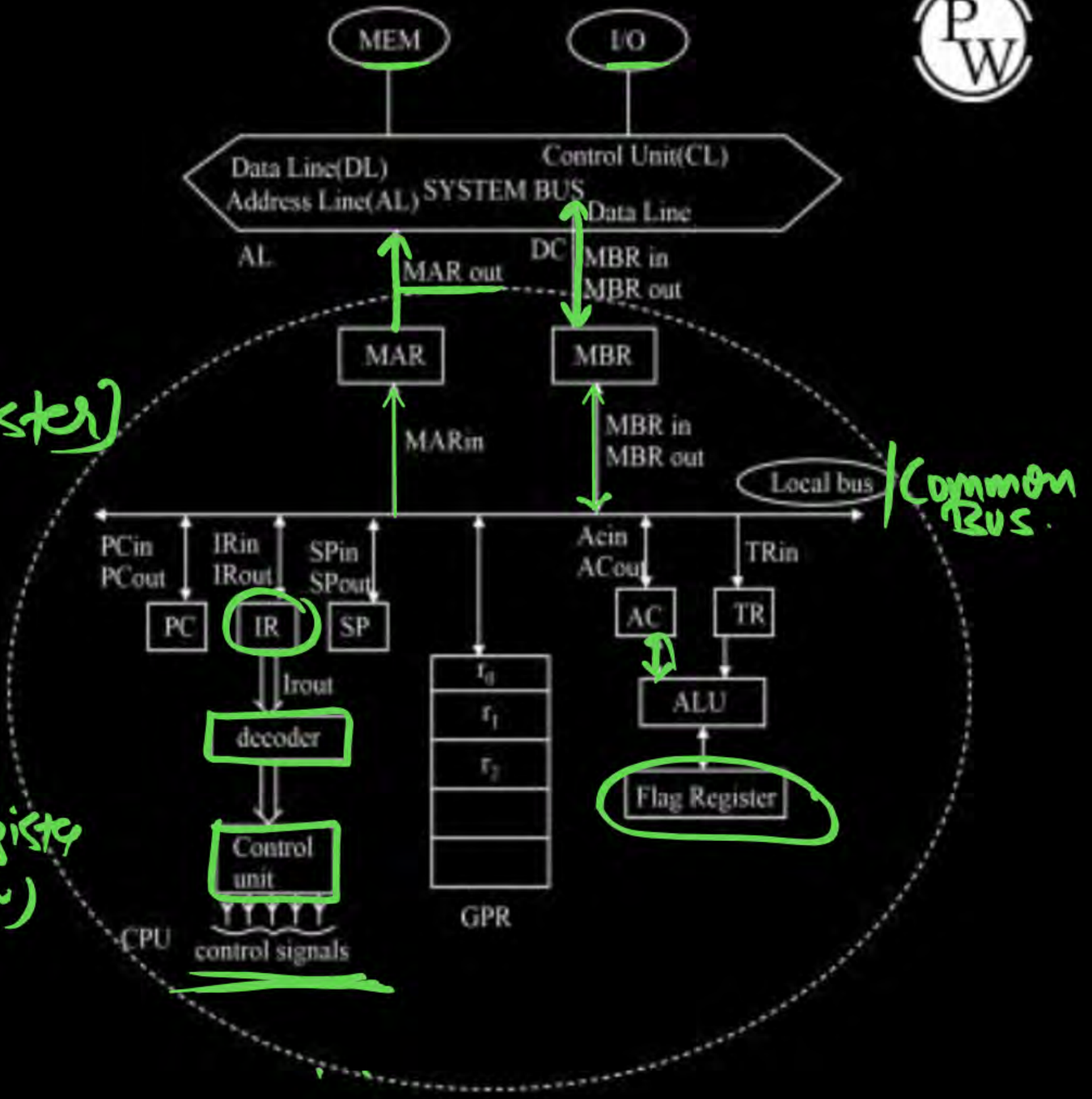
MBR | MDR | DR : it Contain the Memory Content (Instⁿ & Data) It is connect to Data Line of the system BUS.

IR (Instⁿ Register) : Contain the Instruction Currently being executed by the CPU.

Structure of Computer

1. Memory
2. CPU → (ALU, CU, Register)
3. I/O





- In the Processor some general purpose & some special purpose register are available. All the register are connected to a common path called Data Path (Bus). Data Path is collection of functional units(ALU & MUX..)

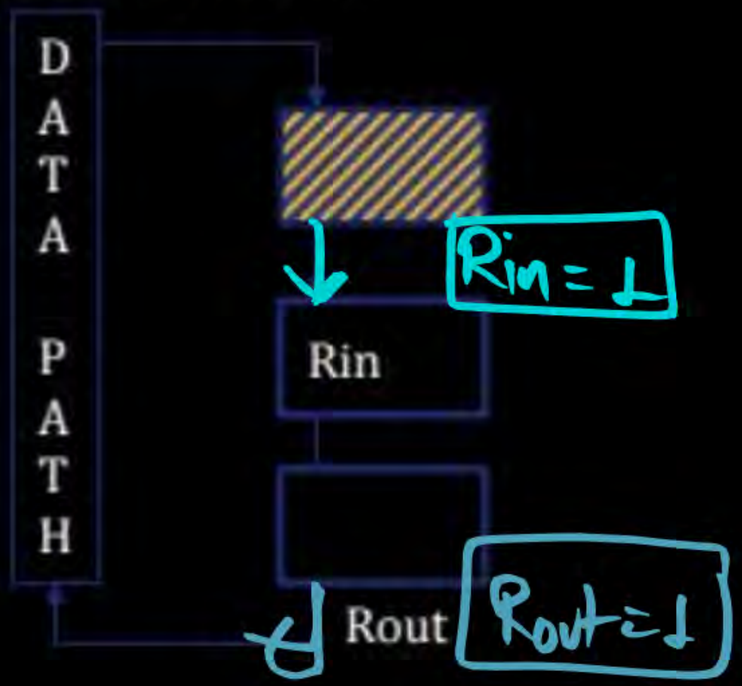
Internal / Local Bus.

Every Register has 2 switch Rin & Rout

R_{in}: R_{in} is set to 1, if the content of the bus loaded into Register Ri.

R_{out}: R_{out} is set to 1, the content from the register R_i will be placed on bus.

DATA PATH : MUX, ALU, BUS, Register How they Process, Path of Processing.



Computer ✓

↓

Program.

↓

Instruction

↓

Instruction cycle.

↓ subcycle.

Fetch & Execute cycle

Micro operation.

In Fetch, Execute cycle we have small-2 operation called Micro Operation (μoperation).

Micro Operations

The functional, or atomic, operations of a processor

- Series of steps, each of which involves the processor registers
- Micro refers to the fact that each step is very simple and accomplishes very little
- The execution of a program consists of the sequential execution of instructions

Each instruction is executed during an instruction cycle made up of shorter sub cycles (fetch, indirect, execute, interrupt)

The execution of each sub cycle involves one or more shorter operations (micro-operations)

Micro Operation

- Micro Operation is a elementary operation in the Hardware

  Example: Register to register transfer operation is a one kind of micro operation.

- Micro operation Consume 1 cycle to complete the execution.
- Control Signal are required to execute the micro operation.

Machine Instruction: MOV r_0, r_1

RTL (Register Transfer Language): $r_0 \leftarrow r_1$

Micro operation: T_1: $r_1$out $r_0$in

MOV γ_0 γ_1

$\gamma_0 \leftarrow \gamma_1$

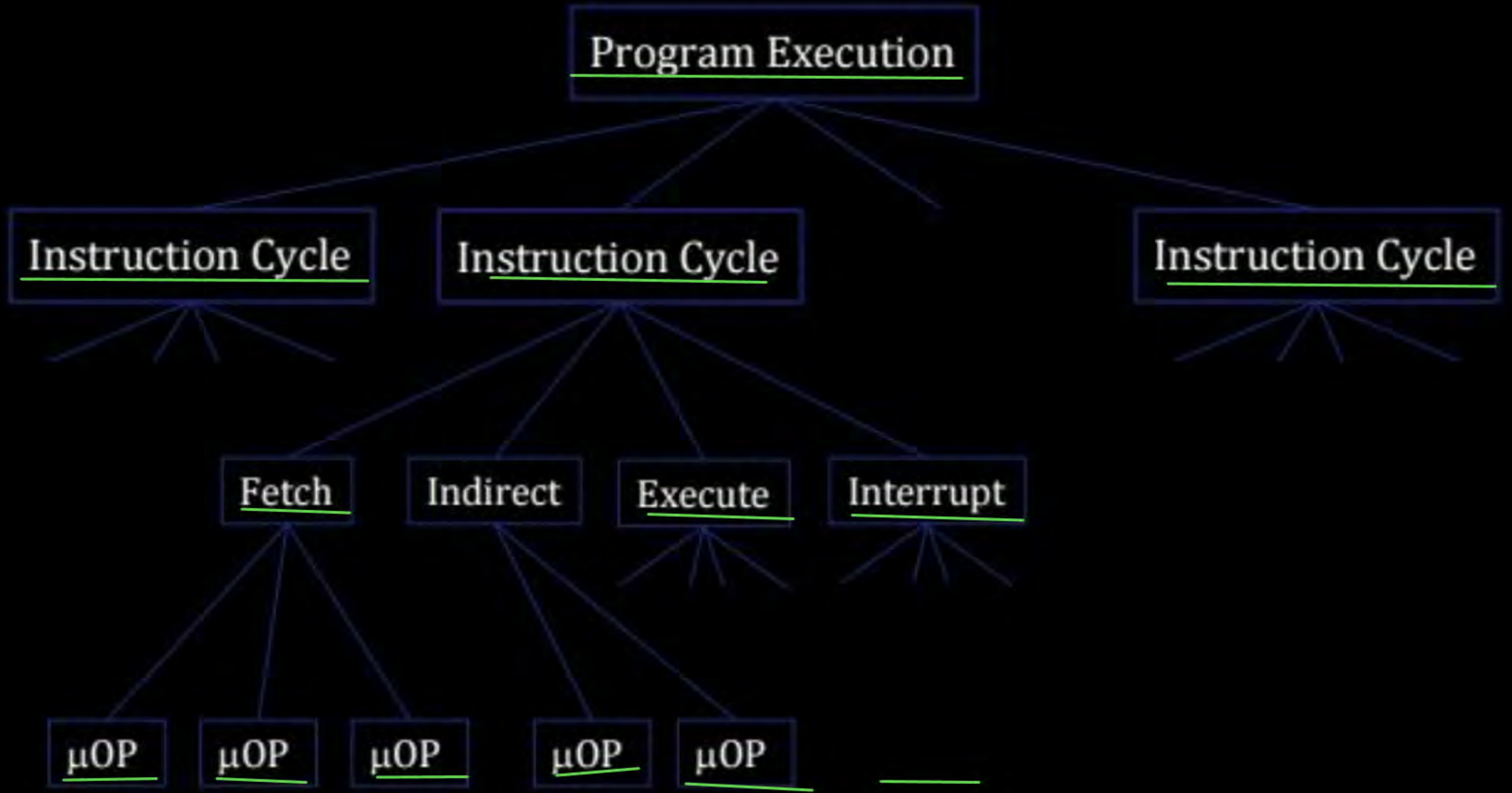
T_1 : $\gamma_{1_{out}}$ $\gamma_{0_{in}}$

Cycle time $\propto \frac{1}{\text{Clock frequency}}$

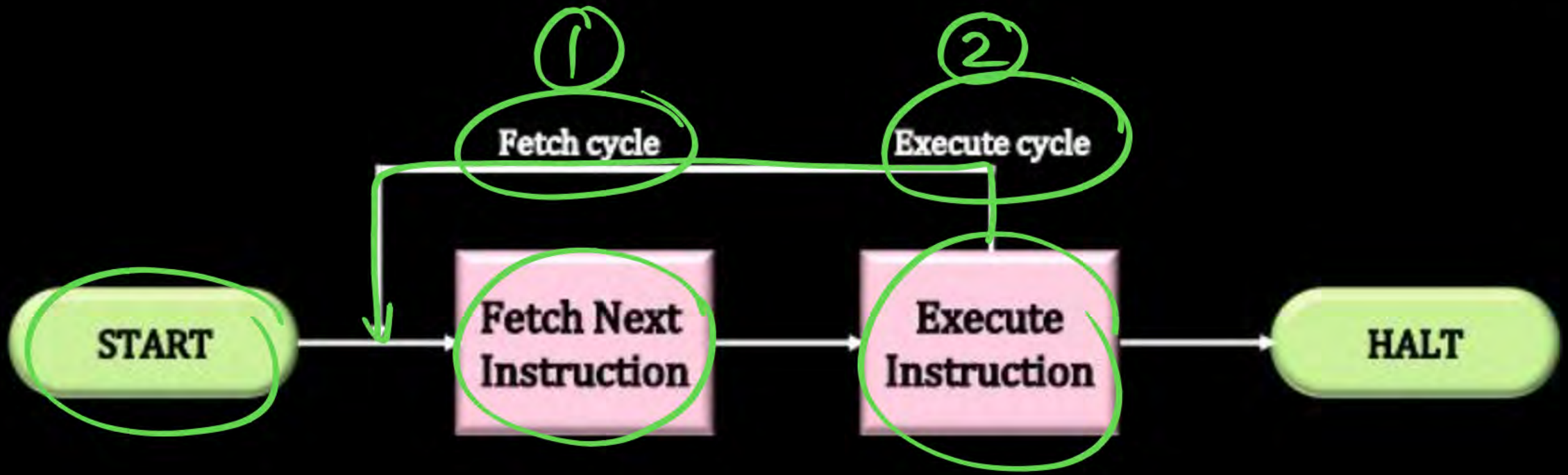
(eg) 1GHz Processor.

Cycle time $\propto \frac{1}{1G}$ sec $\Rightarrow 10^{-9}$ sec.

Cycle time = 1 nsec.



Constituent Elements of a Program Execution

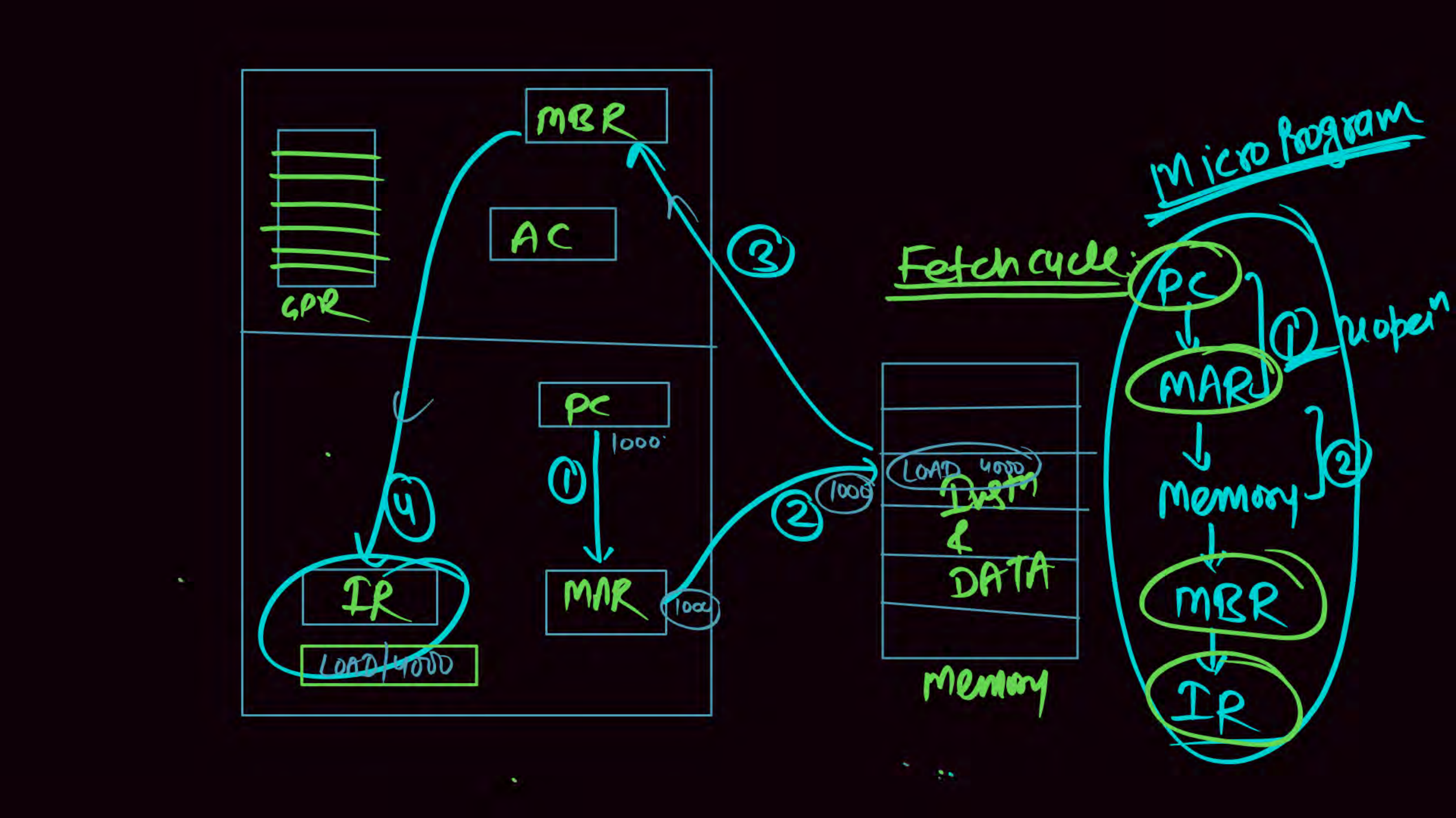


BASIC INSTRUCTION CYCLE

Micro Program (μ - Program)

- Sequence of Micro operation [μ operation] to perform

  Some operation in the Hardware level is called microprogram.


MBR
AC
GPR
PC
1000
IR
LOAD 4000
MAR
1000
LOAD 4000
DATA
&
DATA
Memory
Micro program
Fetch cycle:
PC
MAR
Memory
MBR
IR
1
2
3
4

The Fetch Cycle

- Occurs at the beginning of each instruction cycle and causes an instruction to be fetched from memory (MEM to CPU(IR))

Four registers are involved:

1. Memory Address Register (MAR)
   - Connected to address bus
   - Specifies address for read or write operation
2. Memory Buffer Register (MBR)
   - Connected to data bus
   - Holds data to write or last data read
3. Program Counter (PC)
   - Holds address of next instruction to be fetched
4. Instruction Register (IR)
   - Holds last instruction fetched

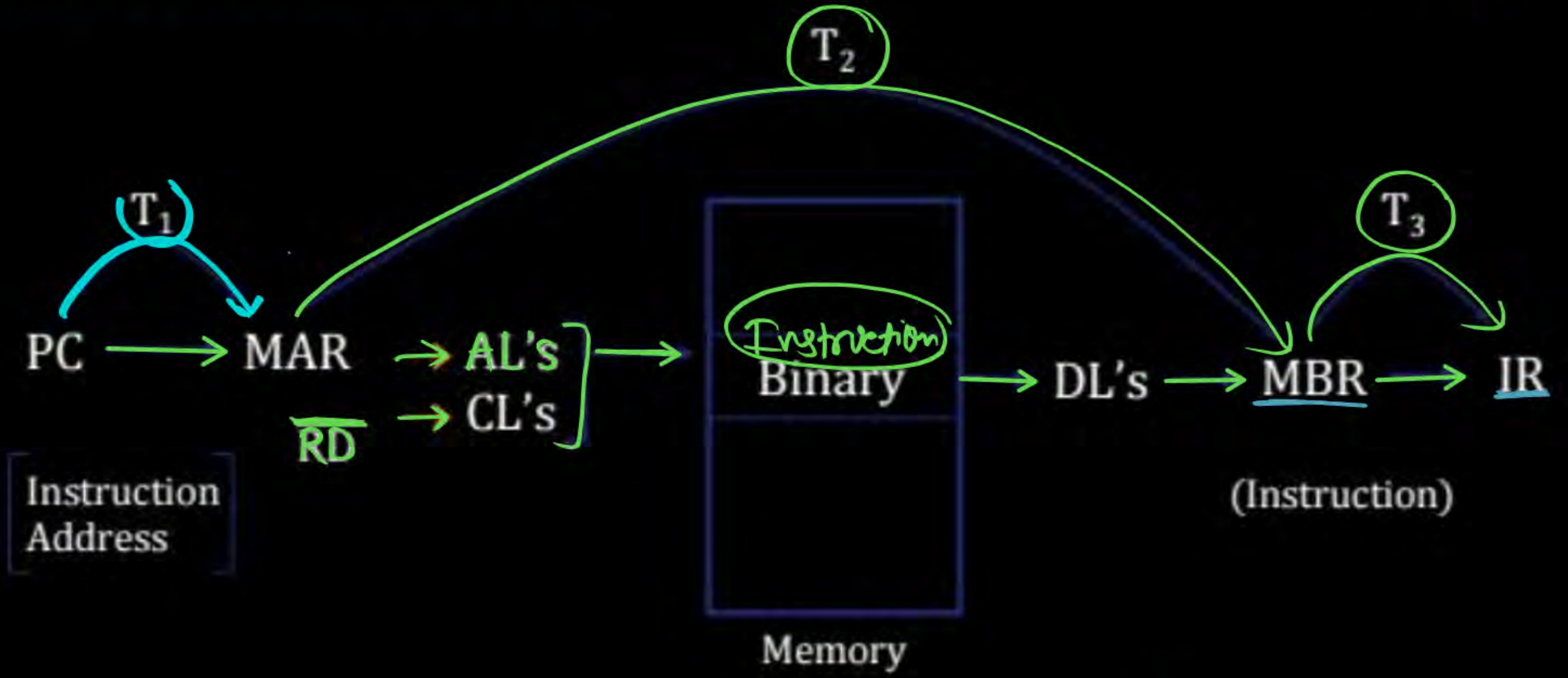
Instruction Cycle

1000 I_1: LOAD [4000]

(1) Fetch Cycle: Instruction Fetch.

Hardware Design(H/W Design)

AL: Address Line (BUS)
DL: Data Line (BUS)
CL: Control Line (BUS)



T_1: PC $\rightarrow$ MAR ; PC_{out} MAR_{in}

T_2: M[MAR] $\rightarrow$ MBR, PC $\leftarrow$ PC+1 ; MAR_{out} MBR_{in}

T_3: MBR $\rightarrow$ IR ; MBR_{out}, IR_{in}.

Microprogram

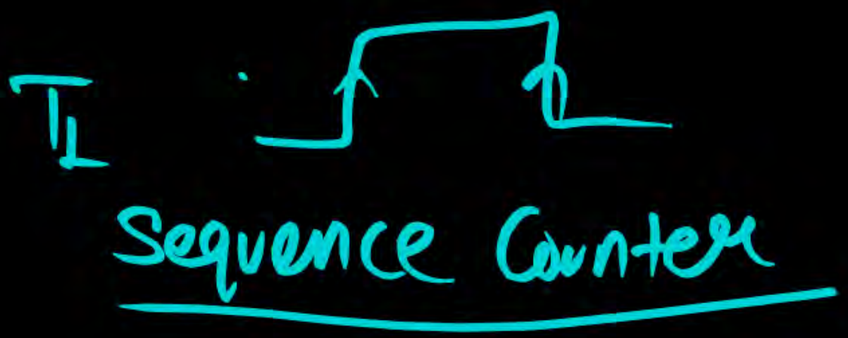
T1: $PC \rightarrow MAR;$ PC_{out} MAR_{in}

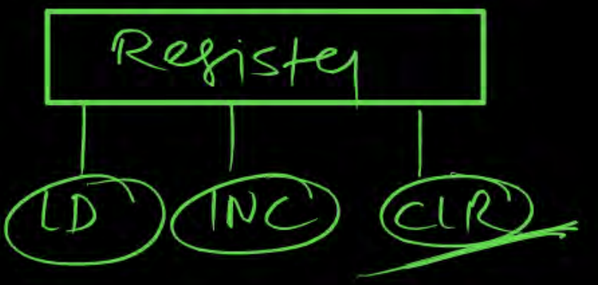
T2: $M[MAR] \rightarrow MBR;$ MAR_{out} MBR_{in} } System Bus

$PC + I \rightarrow PC$ Pc_{out} PC_{in} } Local Bus

Increment

T3: $MBR \rightarrow IR$ MBR_{out} IR_{in}

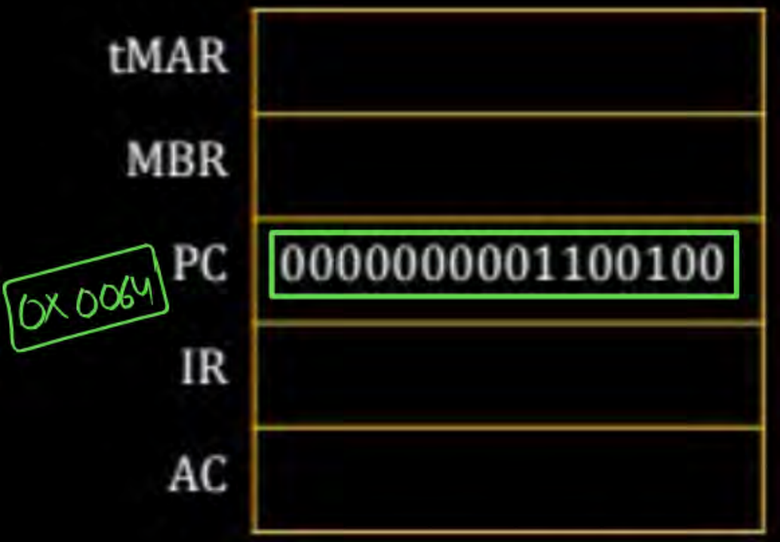




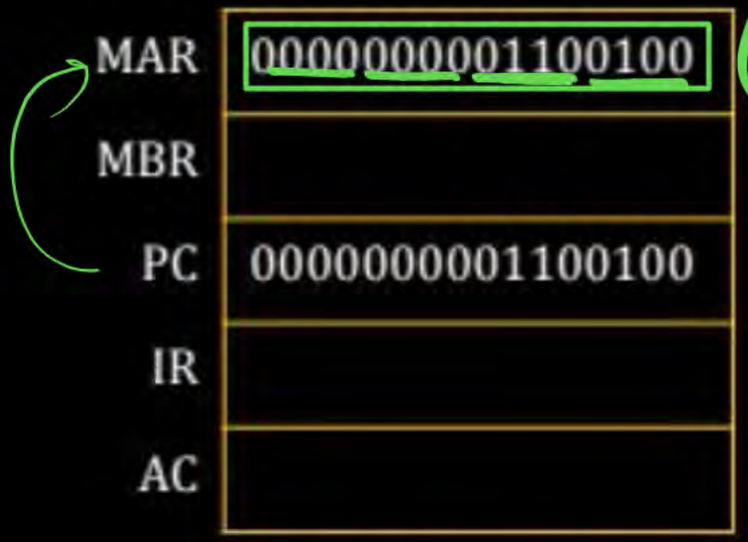
LD: Load

INC: Increment (Binary counter)

CLR: Clear.



(a) Beginning (before t_1)

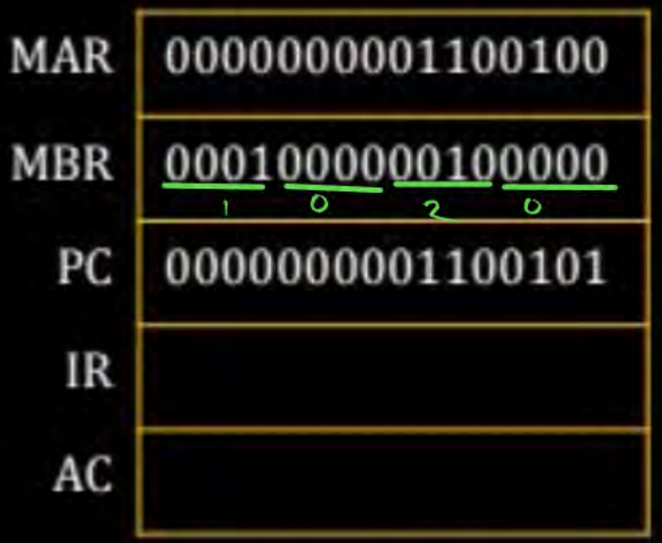


(b) After first step

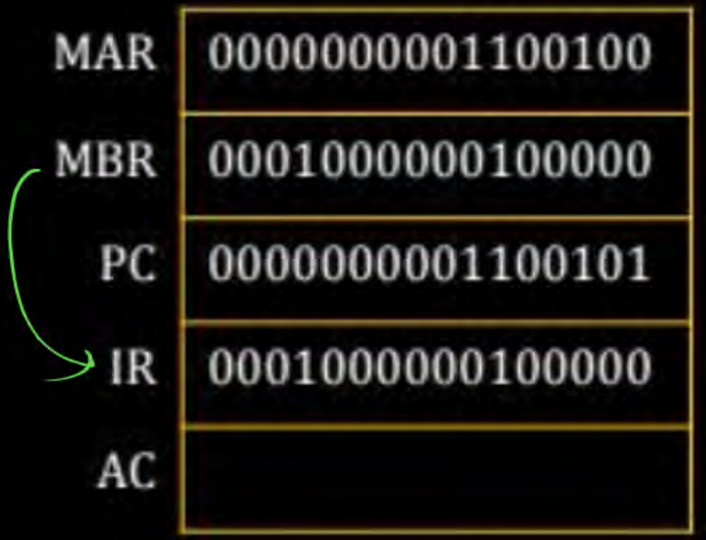
(0064)H ⇔ 0X 0064

PC → MAR → MEMORY → MBR → IR

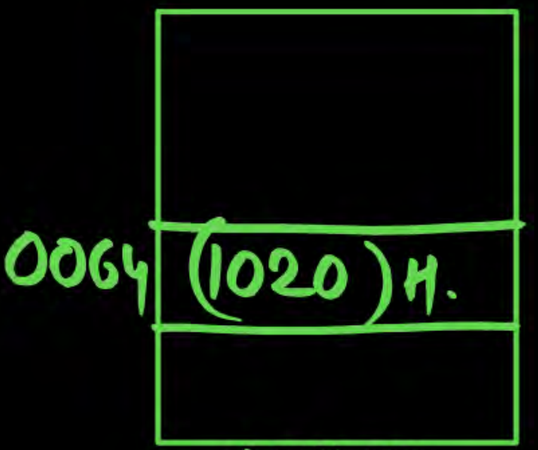
Sequence of events, fetch cycle



(c) After second step



(d) After third step



Rules for Micro Operations Grouping

Proper sequence must be followed

- MAR ← (PC) must precede MBR ← (memory)

T_1 PC → MAR

T_2 M(MAR) → MBR, PC ← PC+1.

T_3: MBR → IR

Conflicts must be avoided

- Must not read and write same register at same time
- MBR ← (memory) and IR ← (MBR) must not be in same cycle

If Different Compent are there then Perform in Single cycle (∵ No Conflict)

MBR ← M(MAR) & PC ← PC+Increment
Done in One cycle.

At the end of Fetch cycle

Instruction is load from Memory to CPU (IR Register)

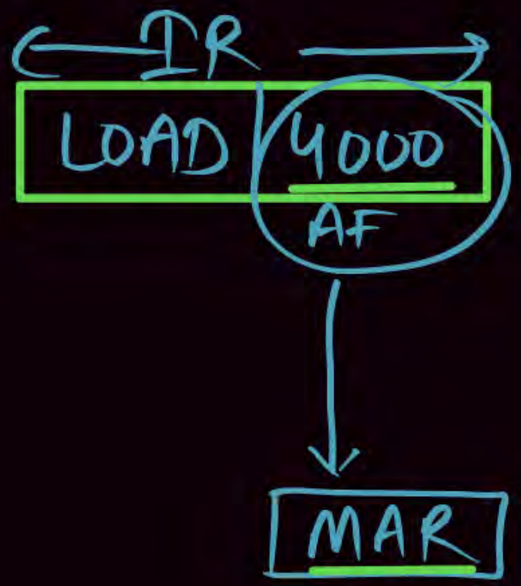
Execute cycle

1. Decode the Instⁿ (WHAT OPCODE, How Many operand, where are operand etc)
2. Operand Address Calculation
3. Operand Fetch (Addressing Mode eg Direct, Indirect, Reg Indirect or any other)

Data Processing

Result Store (WB)

...


IR
LOAD
4000
AF
MAR

Execute Cycle

I_1 : ADD R1, X

- Because of the variety of opcodes, there are a number of different sequences of micro-operations that can occur.
- Instruction decoding
  - The control unit examines the opcode and generates a sequence of micro-operations based on the value of the opcode
- A simplified add instruction:
  - ADD R1, X (which adds the contents of the location X to register R1)
    1. In the first step the address portion of the IR is loaded into the MAR
    2. Then the referenced memory location is read
    3. Finally the contents of R1 and MBR are added by the ALU
  - Additional micro-operations may be required to extract the register reference from the IR and perhaps to stage the ALU inputs or outputs in some intermediate registers

By Default AC

LOAD ⇒ Memory Read

'AC' ⟵ M[x]

STORE ⇒ Memory write

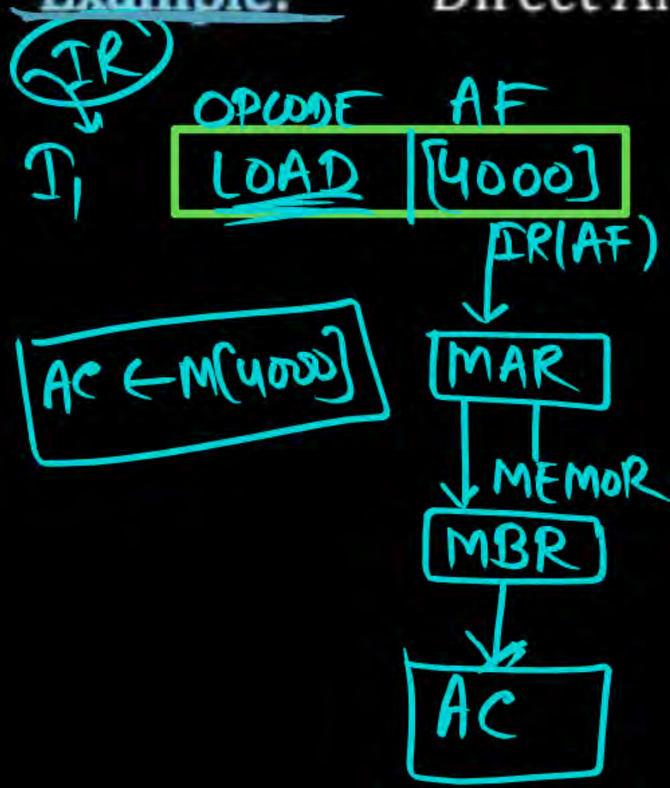
M[y] ⟵ 'AC'

Execute Cycle

(a) ID Stage: Enable the Hardwire to perform the operation (Instruction Decode)

(b) OF Stage: AM's (addressing mode) are required to access. (Operand Fetch)

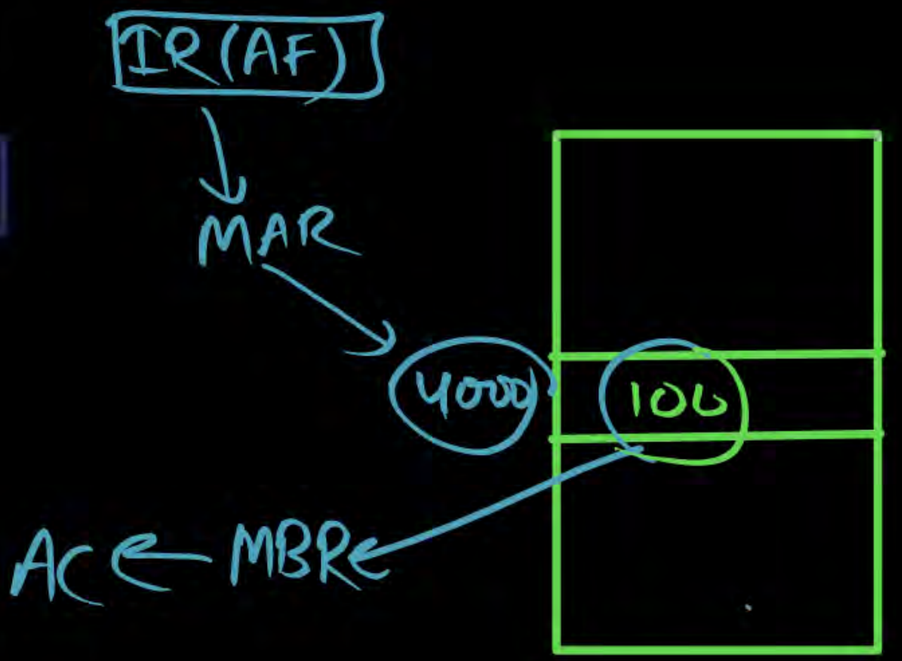
Example: Direct AM



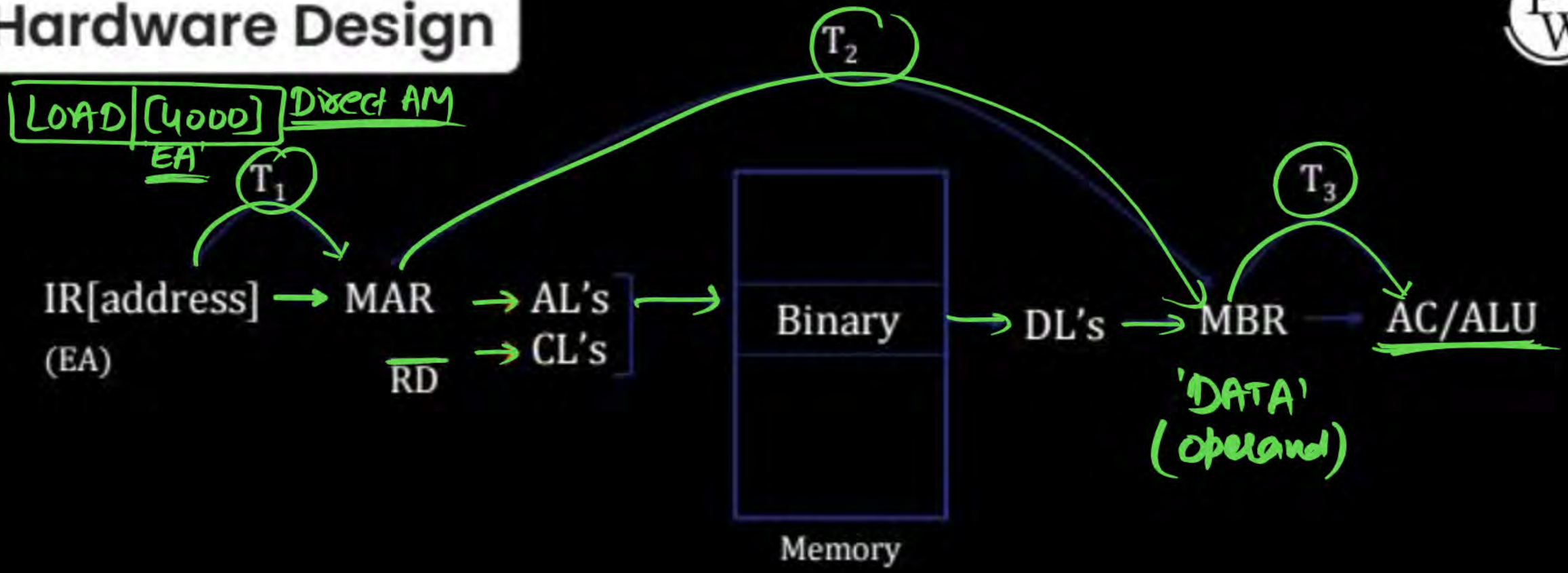
Memory Read

eg LOAD [4000]

AC ← m[4000]



Hardware Design



1. T1: IR(Address field) → MAR: IR_{out} MAR_{in}
2. T2: M[MAR] → MBR: MAR_{out} MBR_{in}
3. T3: MBR → AC/ALU MBR_{out} AC_{in}/ALU_{in}

Subbose

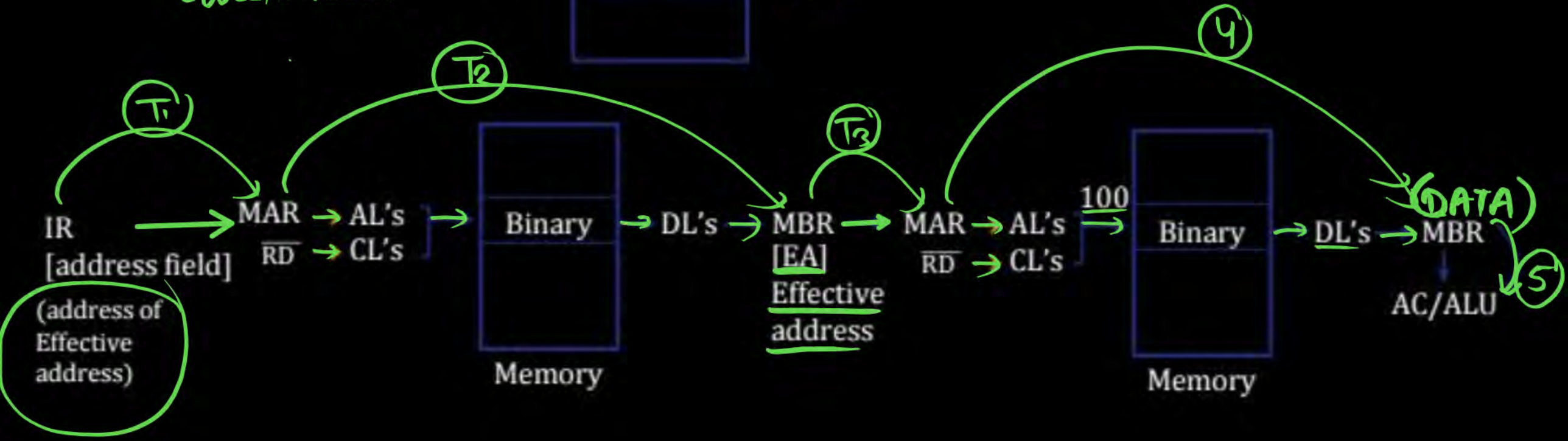
Example: Indirect AM

Load @4000

AC ← M[[4000]]

Address of Effective Address

(EA) 100	11 (DATA)	
4000	100	(EA)



Microprogram:

(1) T_1:	IR[Address] → MAR:	IR_{out}	MAR_{in}
(2) T_2:	M[MAR] → MBR(EA):	MAR_{out}	MBR_{in}
(3) T_3:	MBR → MAR:	MBR_{out}	MAR_{in}
(4) T_4:	M[MAR] → MBR:	MAR_{out}	MBR_{in}
(5) T_5:	MBR → AC/ALU:	MBR_{out}	AC_{in}/ALU_{in}

→ getting EA (Effective Address)

M[6000] ← AC

System Bus

Address Line — MAR | AR

Data Line — MBR | MDR | DR

Control Line — Control Signal] generated by Control Unit

Read ⇒ $\overline{RD}$ Control Signal to CL

Write | Store ⇒ $\overline{WR}$ Control Signal to CL.

Example:

Memory write

Store [6000]

↓

M[6000] ← AC

Micro Program

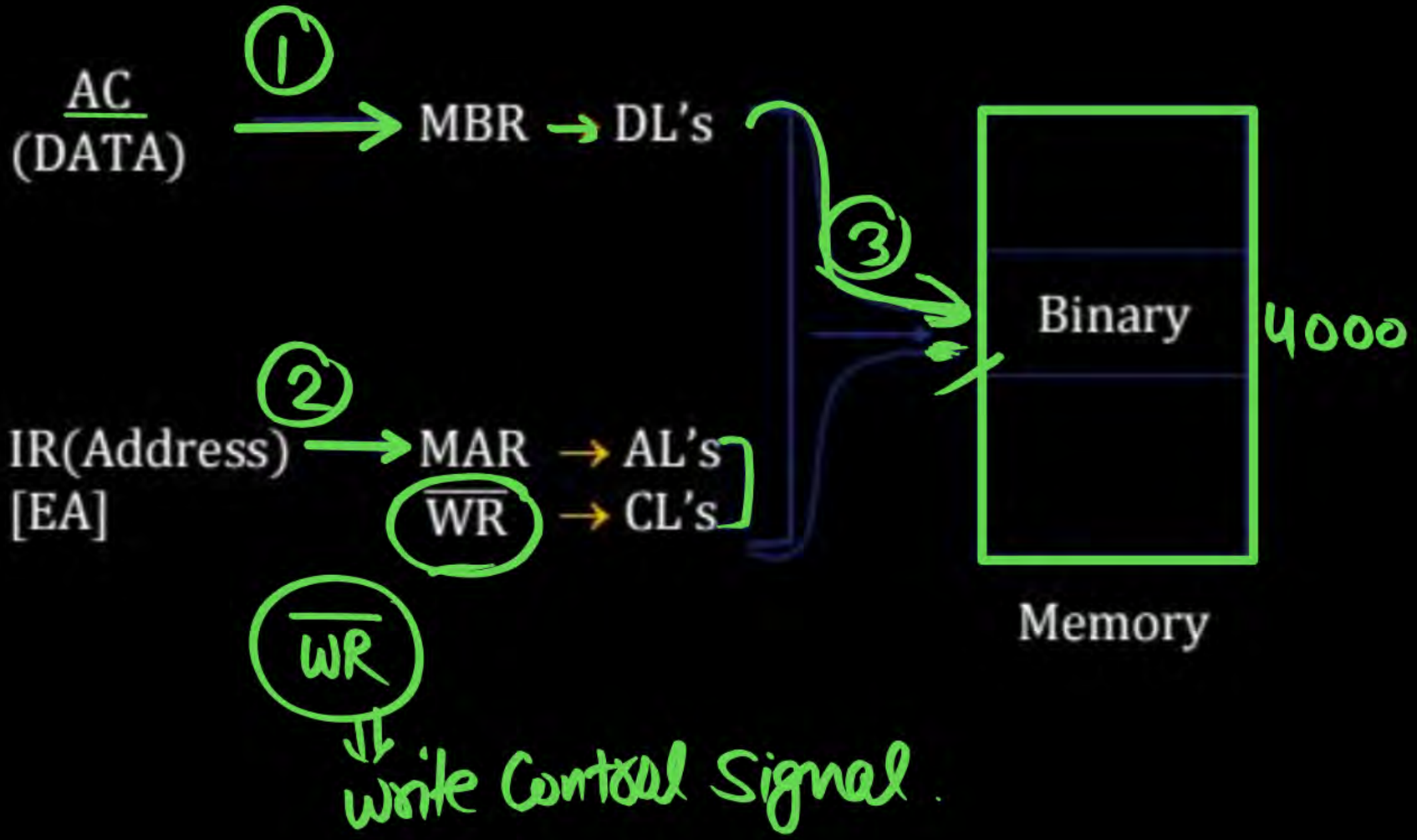
T_1: AC → MBR:	AC_{out}	MBR_{in}
T_2: IR(Address) → MAR	IR_{out}	MAR_{in}
T_3: MBR → M[MAR]	MBR_{out}	MAR_{in}

M[6000] ← AC → 'DATA'

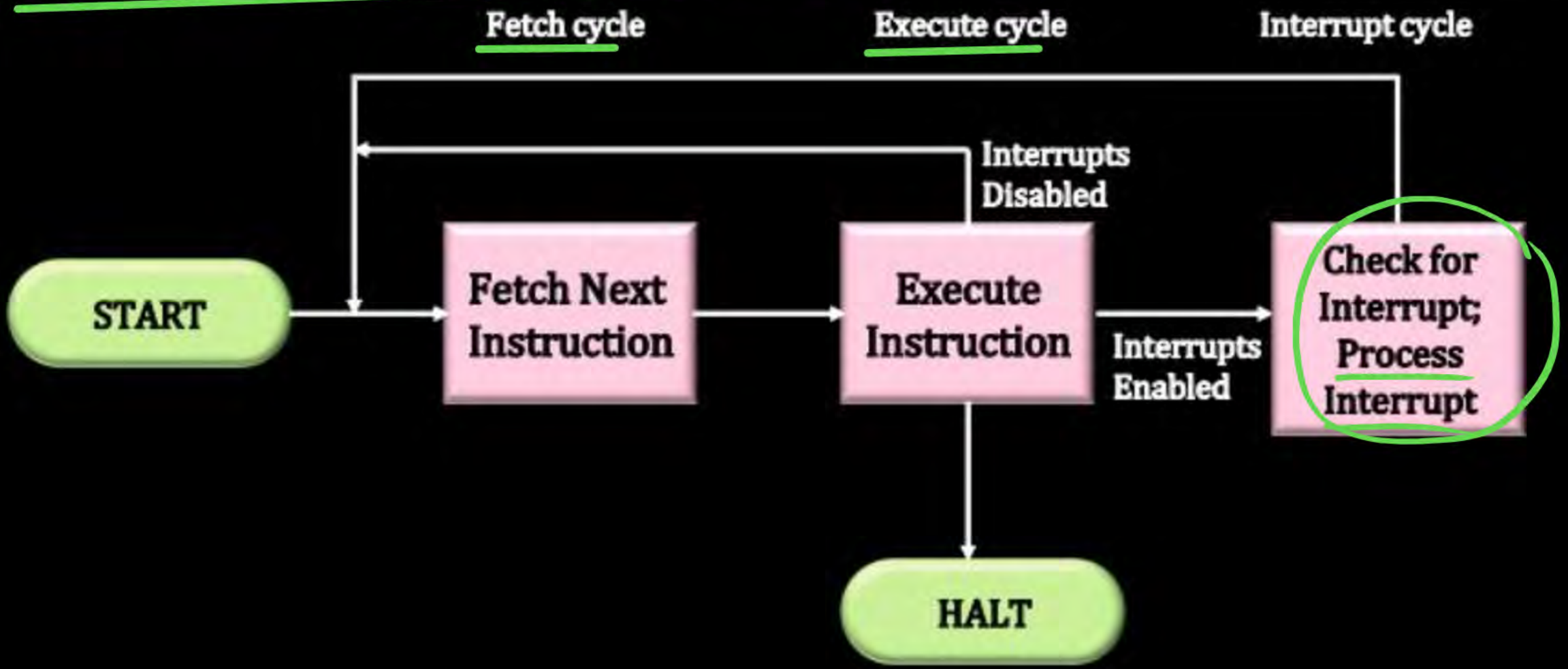
IR(AF) → MAR

AC → MBR

MBR → M[MAR]


AC
(DATA)
1
MBR → DL's
3
Binary
4000
2
IR(Address)
[EA]
MAR → AL's
WR → CL's
WR
Memory
WR
Write Control Signal.

Instruction Cycle With Interrupt



Instruction cycle with Interrupts

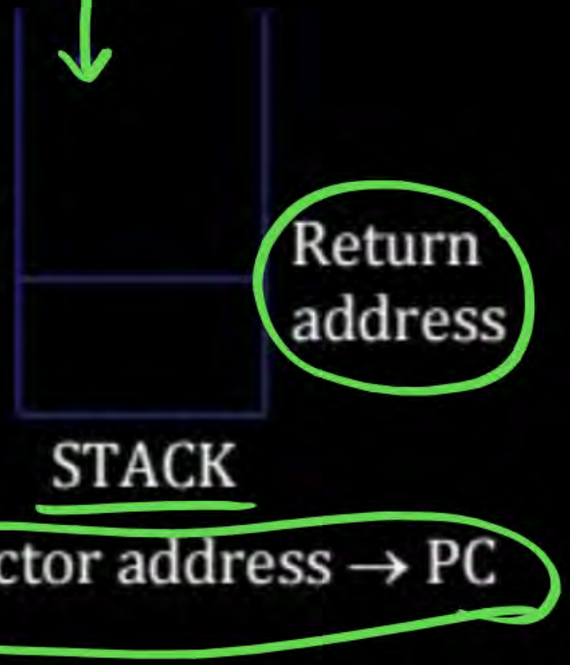
Interrupt cycle:

- At the completion of the execute cycle, a test is made to determine whether any enabled interrupts have occurred, and if so, the interrupt cycle occurs
- The nature of this cycle varies greatly from one machine to another
- In a simple sequence of events:
  - (1) In the first step the contents of the PC are transferred to the MBR so that they can be saved for return from the interrupt
  - (2) Then the MAR is loaded with the address at which the contents of the PC are to be saved, and the PC is loaded with the address of the start of the interrupt processing routine
  - These two actions may each be a single micro-operation
  - Because most processors provide multiple types and/or levels of interrupts, it may take one or more additional micro-operations to obtain the Save Address and the Routine Address before they can be transferred to the MAR and PC respectively
  - Once this is done, the final step is to store the MBR, which contains the old value of the PC, into memory
  - The processor is now ready to begin the next instruction cycle

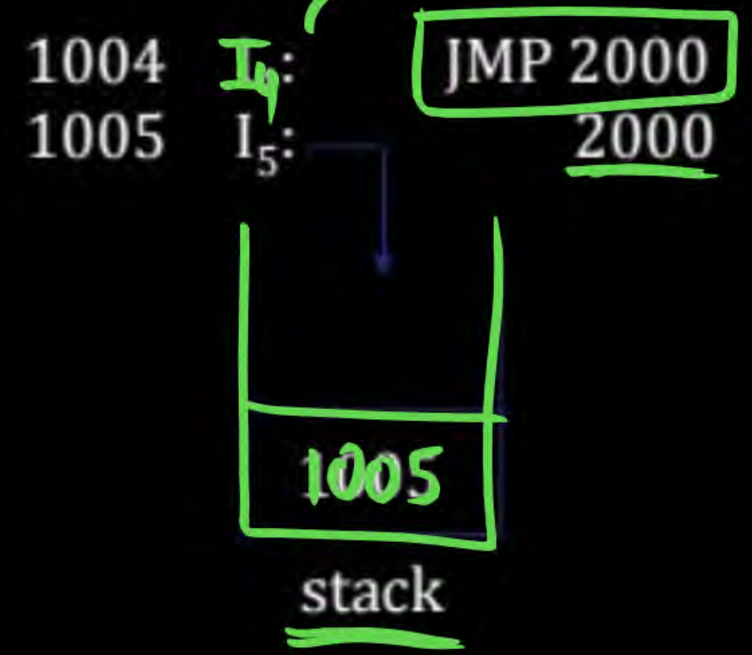
Interrupt cycle:

Whenever Interrupt occur after the completion of current Instruction execution interrupt will be serviced.

PC value stored in the stack as return address &
Control transfer to ISR (Interrupt subroutine)



If Fetch then PC: 1005



PC Value into Stack

as Return Address

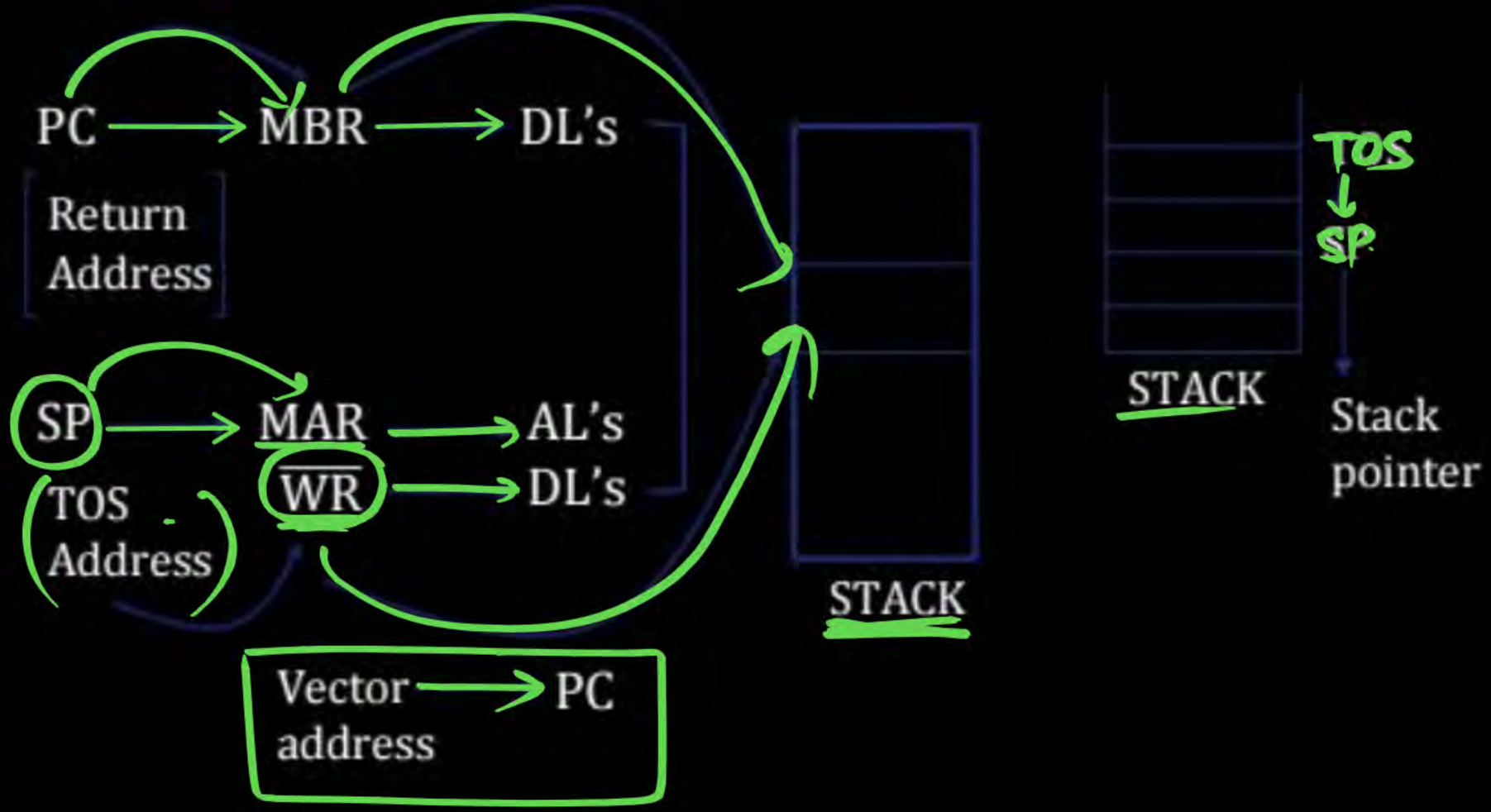
STACK

& Interrupt Address → PC

Interrupt service

& Pop the PC value for Next Inst^n execution.

Hardware Design



Microprogram
μ Program

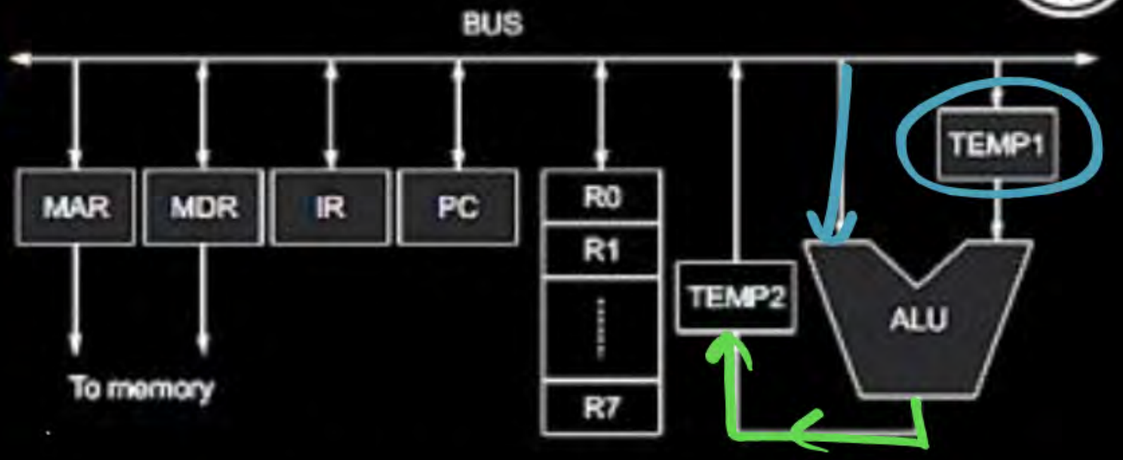
- T1: PC → MBR ; PCout MBRin
- T2: SP → MAR ; SPout MARin
- T3: MBR → M[MAR]: System Bus MBRout MARin(Stack (TOS))
  Vector Address → PC: Local Bus

IVT (Interrupt vector table)

ISR

MCQ

Consider the following data path diagram



Consider an instruction: R0 ← R1+R2.

The following steps are used to execute it over the given data path. Assume that PC is incremented appropriately. The subscripts r and w indicate read and write operations, respectively.

1. $R2_r$, $TEMP1_R$, ALU_{add}, $TEMP2_W$
2. $R1_r$, $TEMP1_W$,
3. PC_r, MAR_W, MEM_r
4. $TEMP2_R$, RO_W
5. MDR_r, IR_W

Which one of the following is the correct order of execution of the above steps?

[GATE-2020-CS: 1M]

ALU

$R_0 \leftarrow R_1 \oplus R_2$

Ans(C)

(A) 3, 5, 1, 2, 4

(B) 2, 1, 4, 5, 3

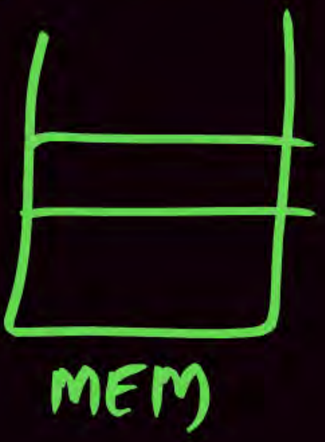
(C) 3, 5, 2, 1, 4

(D) 1, 2, 4, 3, 5

Fetch

$R_0 \in R_1 + R_2$

PC → MAR → MBR → IR.
Fetch cycle.



3. PC_{γ} MAR_W MEM_R ; PC → MAR

5 MBR_{γ}, IR_W ; MBR → IR. (RD)

2 $R_{1\gamma}$, $Temp_1 w$; $R_1 \rightarrow Temp1$.

1 $R_{2\gamma}$, $Temp_{1\gamma}$ ALU_{ADD} $Temp_{2w}$; $Temp_2 \in Temp_1 + R_2$

4. $Temp_{2\gamma}$, R_{0w} ; $R_0 \in R_1 + R_2$

3 5 2 – 4

MCQ

The following are some events that occur after a device controller issues an interrupt while process L is under execution.

(P) The Processor pushes the process status of L onto the control stack.

(Q) The processor finishes the execution of the current instruction.

(R) The processor executes the interrupt service routine.

(S) The processor pops the process status of L from the control stack.

(T) The processor loads the new PC value based on the interrupt.

Which one of the following is the correct order in which the events above occur?

[GATE-2018-CS: 1M]

(A) QPTRS

(B) PTRSQ

(C) TRPQS

(D) QTPRS

QPTR

MCQ

Consider the following sequence of micro -operations.

MBR ← PC

MAR ← X

PC ← Y

Memory ← MBR

Which one of the following is a possible operation performed by this sequence

[GATE-2013-CS: 2M]

A Instruction fetch

B operand fetch

Ans(D).

C Conditional branch

D Initiation of interrupt service

PC value stored in memory (X) then PC is updated by 'y' Address.

'Memory'

PC → MBR

X → MAR

y → PC

MBR → MAR

X ⓐ Instⁿ Fetch

PC → MAR → MEM → MBR → IR

Mem to CPU (IR)

X ⓑ Operand Fetch.

IR(AF) → MAR → Memory → MBR → AC/ALU.

Control Unit Functional Requirements

- By reducing the operation of the processor to its most fundamental level we are able to define exactly what it is that the control unit must cause to happen
- Three step process to lead to a characterization of the control unit:
  - Define basic elements of processor
  - Describe micro-operations processor performs
  - Determine the functions that the control unit must perform to cause the micro-operations to be performed
- The control unit performs two basic tasks:
  - Sequencing
  - Execution

Control Unit

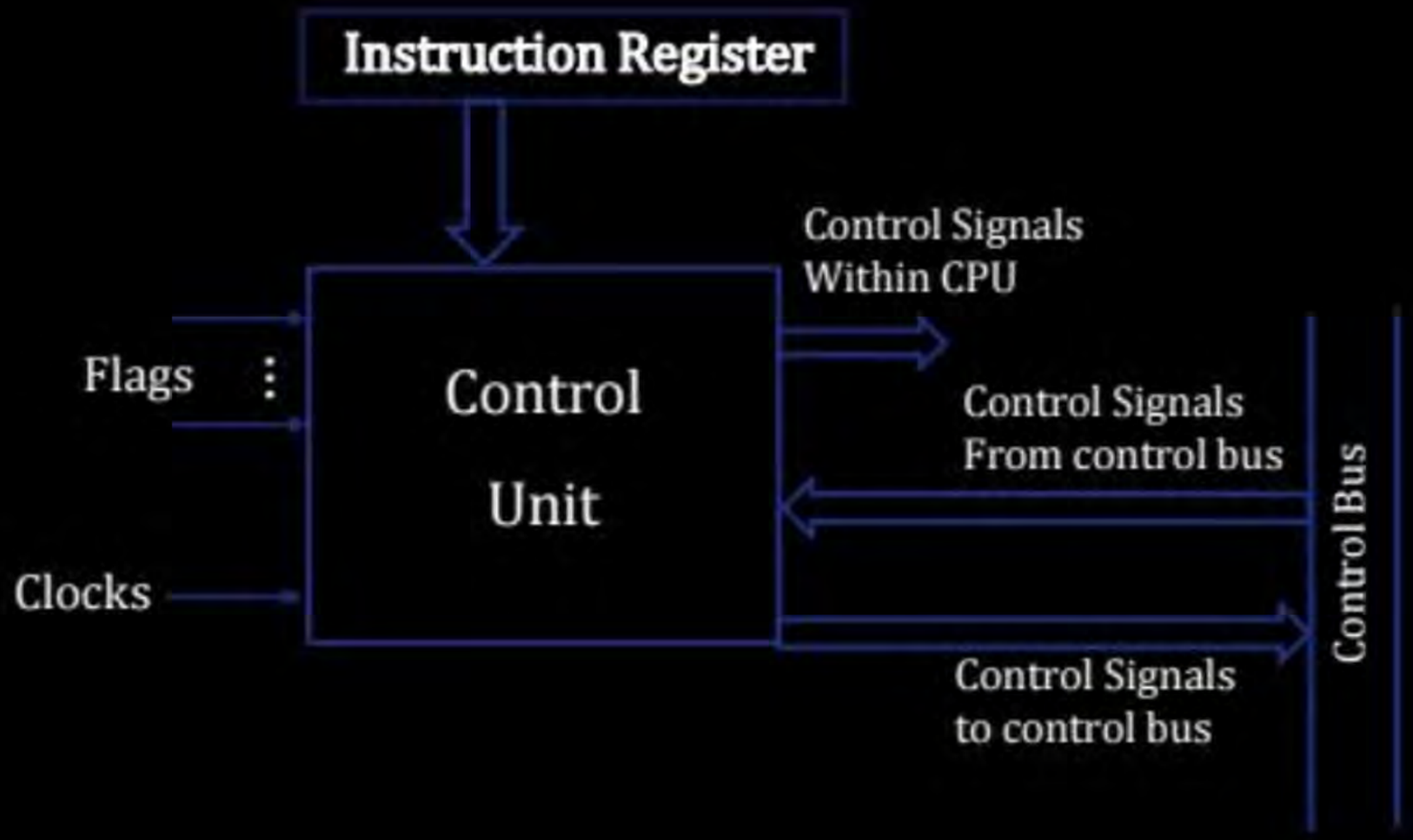
Control unit is the Supervisor in the System that control each & every activity.

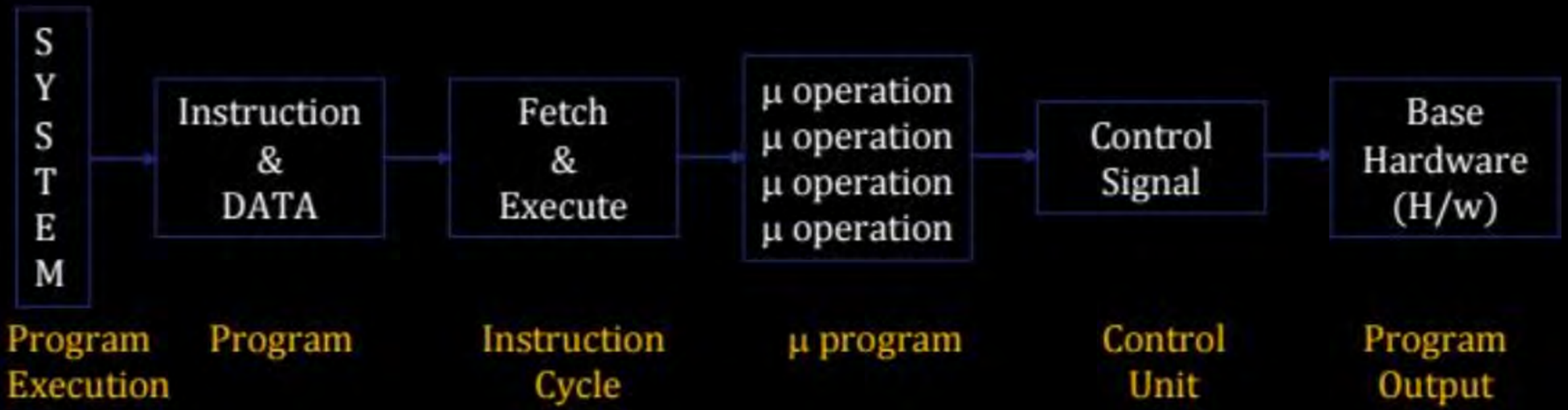
- Control Signals are implement in a Control Unit.
- Control Signal are Required to execute the micro operation.
- Micro operation is the elementary operation in the hardware.
- Control unit generates the sequence of control Signal.
- Control Signal are Directly executed on a Base Hardware (H/W)

  So H/W generate the desired Response.

Computer System Functionality is Program Execution.

Block Diagram of the Control Unit

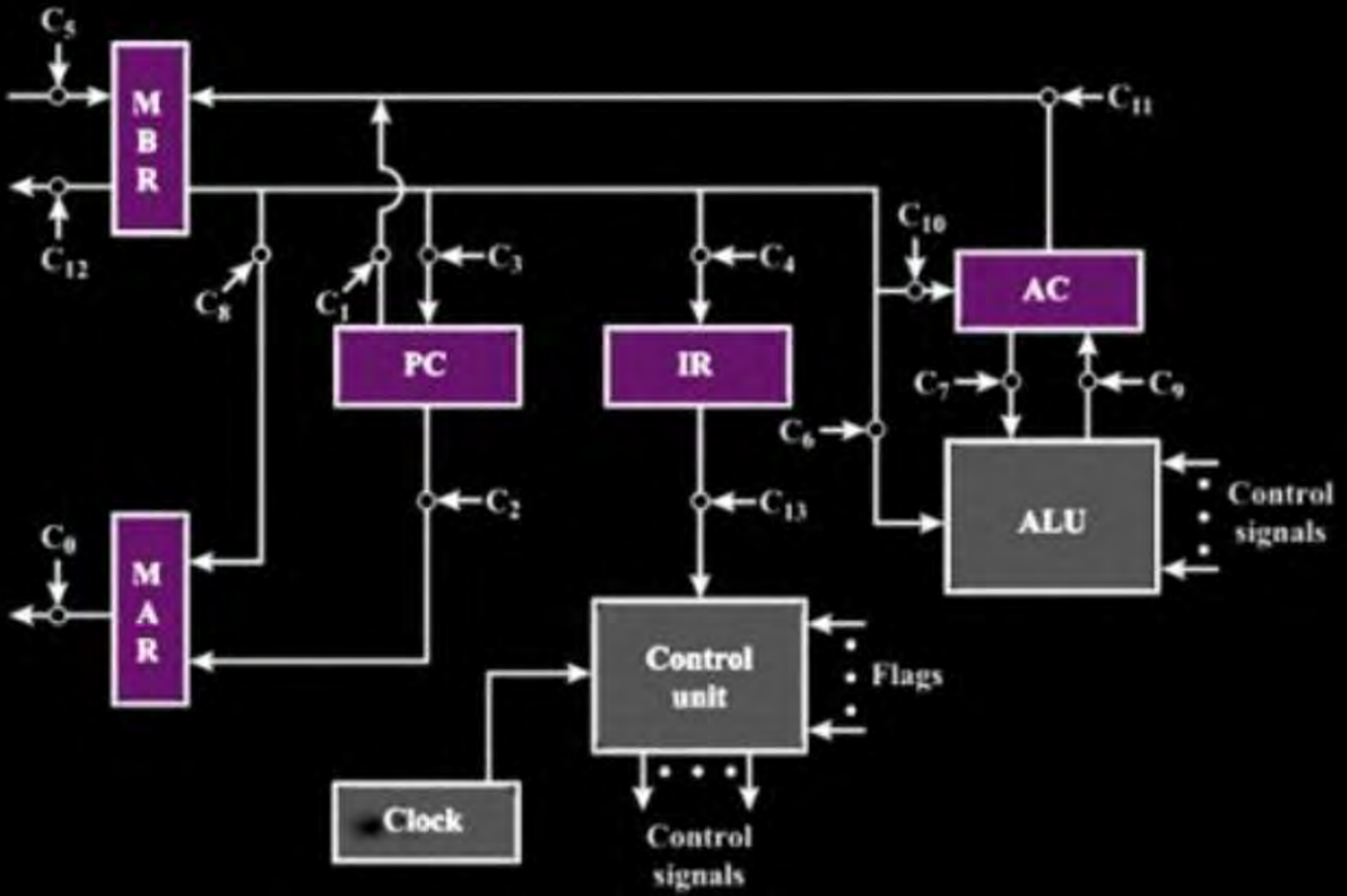



SYSTEM
Instruction & DATA
Fetch & Execute
μ operation
μ operation
μ operation
μ operation
Control Signal
Base Hardware (H/w)
Program Execution
Program
Instruction Cycle
μ program
Control Unit
Program Output

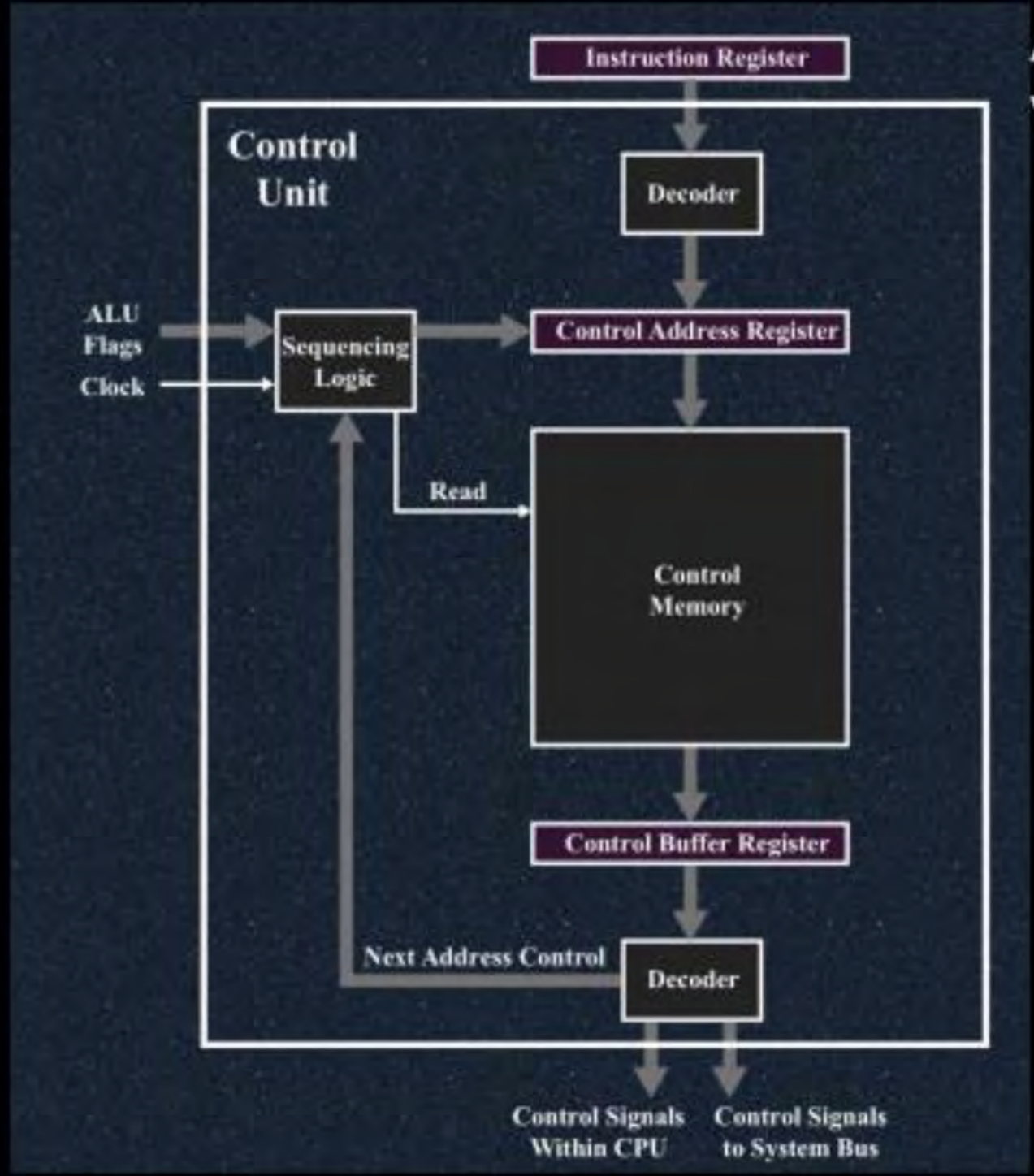
Micro-operations & Control Signals

	Micro-operations	Active-control Signals
Fetch:	T_1: MAR $\Re$ (PC) (or) PC $\rightarrow$ MAR	C_2
	T_2: MBR $\Re$ Memory PC $\Re$ (PC) + 1	C_5, C_R
	T_3: IR $\Re$ (MBR)	C_4
Indirect:	T_1: MAR $\Re$ (IR(Address))	C_8
	T_2: MBR $\Re$ Memory	C_5, C_R
	T_3: IR(Address) $\Re$ (MBR (Address))	C_4
Interrupt:	T_1: MBR $\Re$ (PC)	C_1
	T_2: MAR $\Re$ Save-address PC $\Re$ Routine-address	
	T_3: Memory $\Re$ (MBR)	C_{12}, C_W

Data Paths & Control Signals



Functioning of Microprogrammed Control Unit



Pre Requirement of the CU Design is as follow

1) How many Control Lines are present in the Hardware [S0, S1, S2, S3...]
2) How many Instruction are implemented in the Hardware [I1, I2, I3....]
3) How many Micro operation are required for each Instruction [T1, T2, T3...]
4) What the control Signal Required for each micro operation for each Instruction.

Control Signals will be Implemented into the Control Unit by using following Approach:

1) HARDWIRED CU Design

2) MICRO-PROGRAMMED CU Design

THANK YOU!