# COMPUTER SCIENCE

## Computer Organization and Architecture

## ALU & Control unit

Lecture_02



Vijay Agarwal sir


TOPICS
TO BE
COVERED
01
Micro Operation

Micro operation.

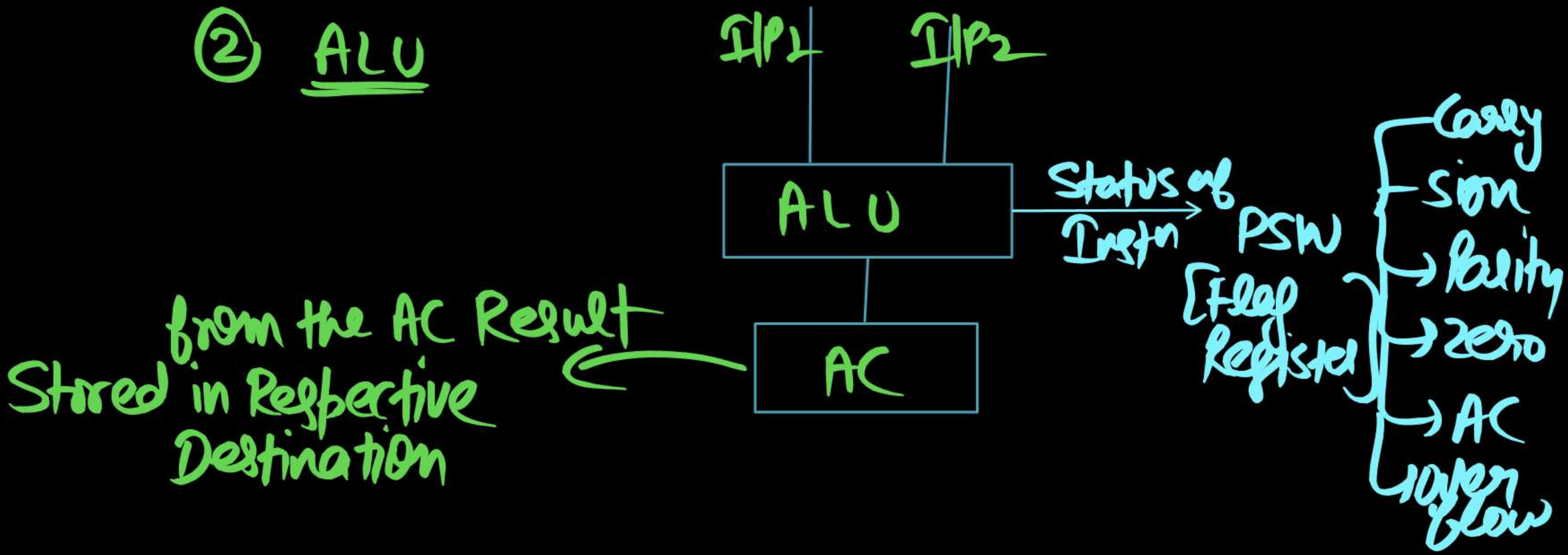
Component of Computer (CPU, I/O Memory)

CPU Org.

- Register
- ALU
- Control Unit
  (Control Signal & Timing Signal)

① Registers (Flip Flop) | LD | INC | CLR

② ALU



# ③ Timing Signal, Control Signals

## Q Why Common Bus is Needed ?

20 Component [16 Register, 1 ALU, 1 Memory, 1 PSW & 1 other]

200 + Connection Required

Instead of using 200+ Connection use Common (Internal) BUS.

But Only Parts Communicate at a time, Which Component Communicate Done by Control signal.

# Working of Register ?

$R_A \rightarrow R_B$ ; $R_{Aout}$ $R_{Bin}$

1. Multiplexer
2. Select line
3. Common Bus
4. Control Signal
5. Timing signal.

## Multiplexer (mux) : Many into 1.

> # Multiplexer = Size of Registers (# bits in Register)
> Size of mux = # Register.

(eg) 32 Registers & each Register is 8 bit

# mux = 8

size of mux = 32 [32 x 1]

4 Register ($R_A$, $R_B$, $R_C$, $R_D$), each Register

Size 4 bits

# Mux = 4

Size of MUX = 4X1 [4]

## ② Select line $[S_1\ S_0]$

| $S_1$ | $S_0$ | |
|---|---|---|
| 0 | 0 | $\rightarrow A$ |
| 0 | 1 | $\rightarrow B$ |
| 1 | 0 | $\rightarrow C$ |
| 1 | 1 | $\rightarrow D$ |

$(A_3\ A_2\ A_1\ A_0)$

$(B_3\ B_2\ B_1\ B_0)$

$(C_3\ C_2\ C_1\ C_0)$

$(D_3\ D_2\ D_1\ D_0)$

(input '0' will be Select across all Mux)

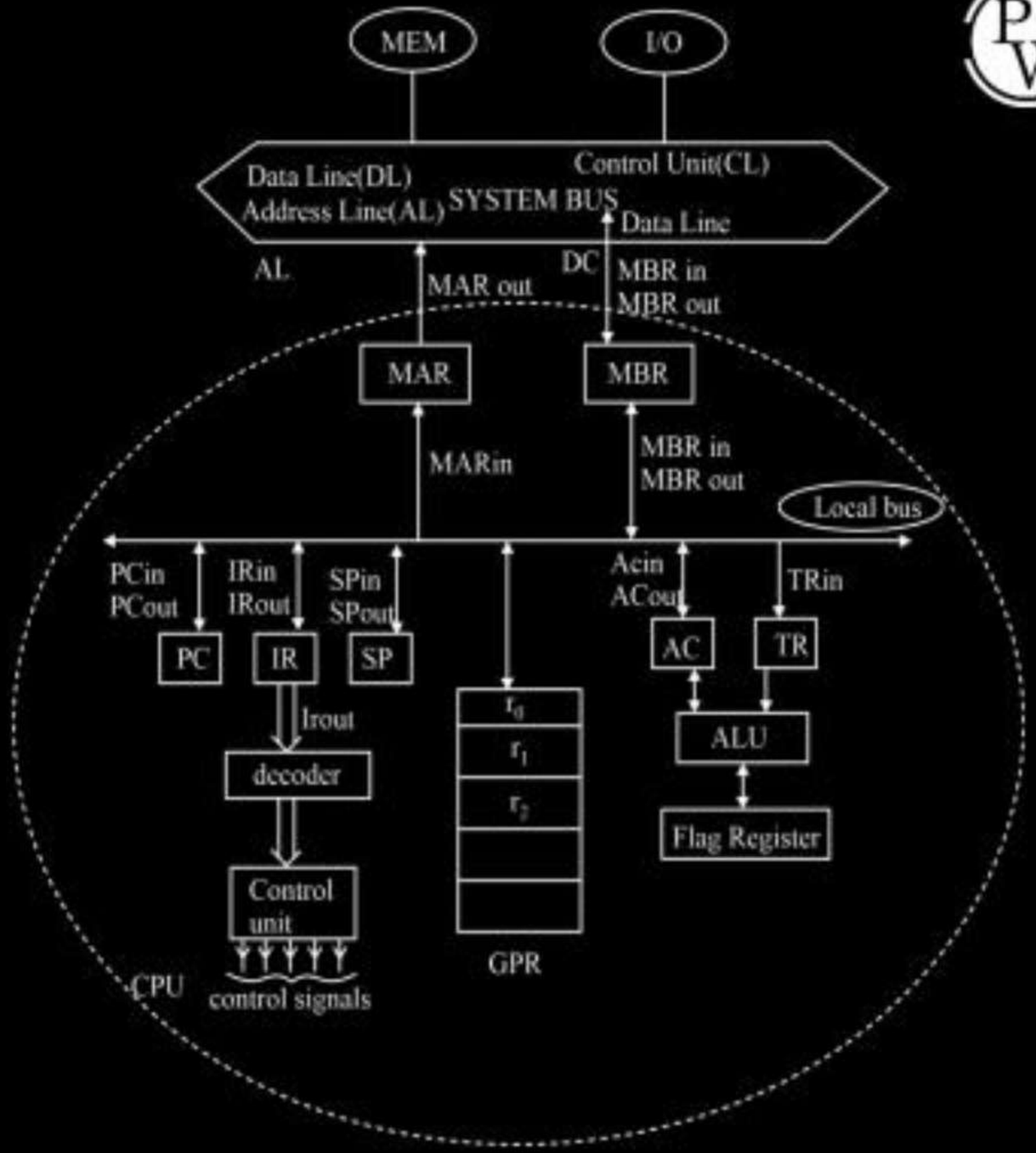
$R_A \rightarrow R_B$ $R_A$ to $R_B$ : $R_{Aout}$ $R_{Bin}$.

00 ⇒ A

---

Rout set to 1 then Register A to MUX, MUX to Common Bus then Common Bus to Load to $R_B$ ($R_{Bin} = 1$).

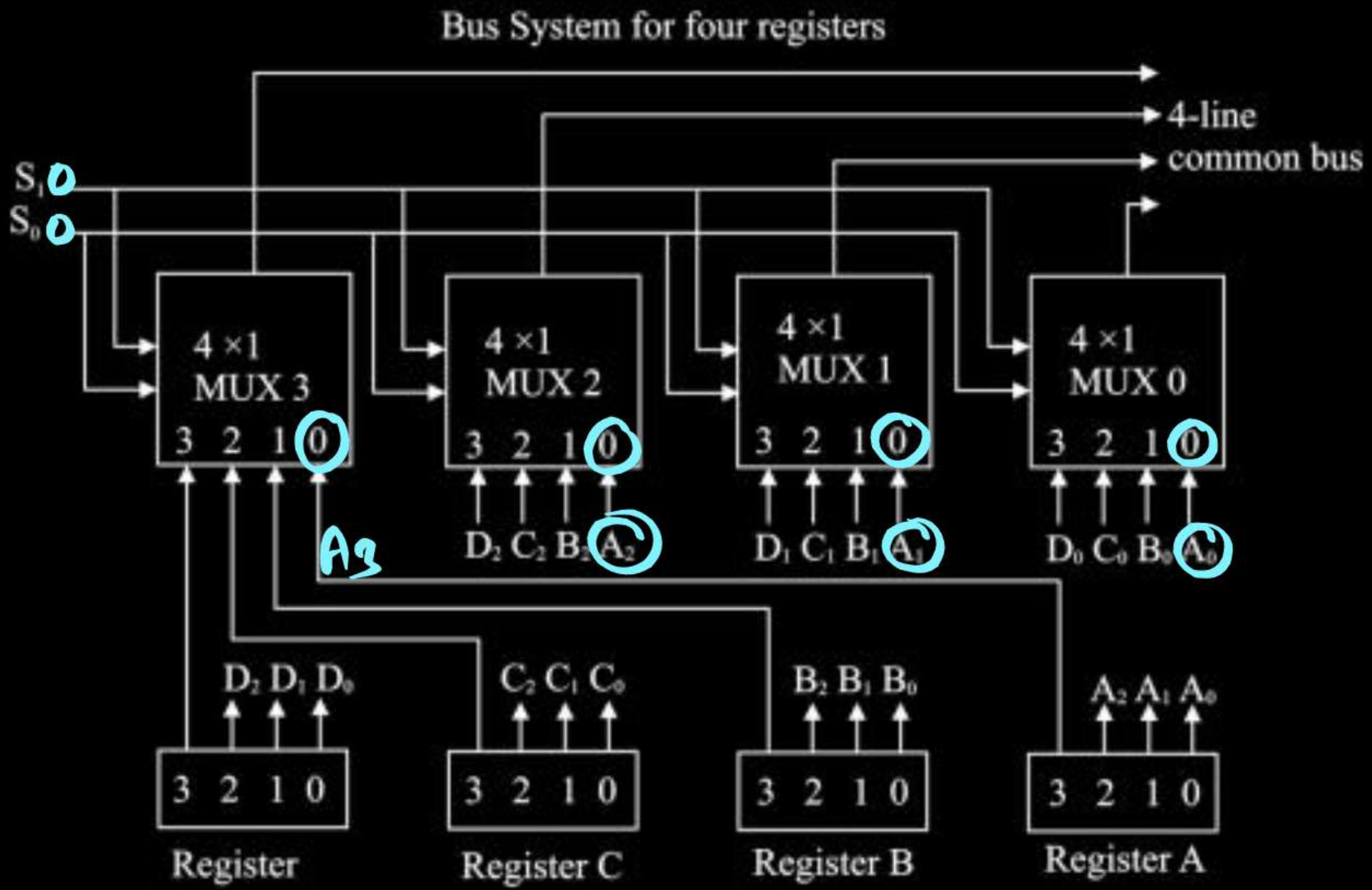
# Structure of Computer



# Component of Computer

1. CPU , 2. Memory & 3. IO

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


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
$D_2$ $C_2$ $B_2$ $A_2$
$D_1$ $C_1$ $B_1$ $A_1$
$D_0$ $C_0$ $B_0$ $A_0$
$A_3$
$D_2$ $D_1$ $D_0$
$C_2$ $C_1$ $C_0$
$B_2$ $B_1$ $B_0$
$A_2$ $A_1$ $A_0$
3 2 1 0
Register
Register C
Register B
Register A

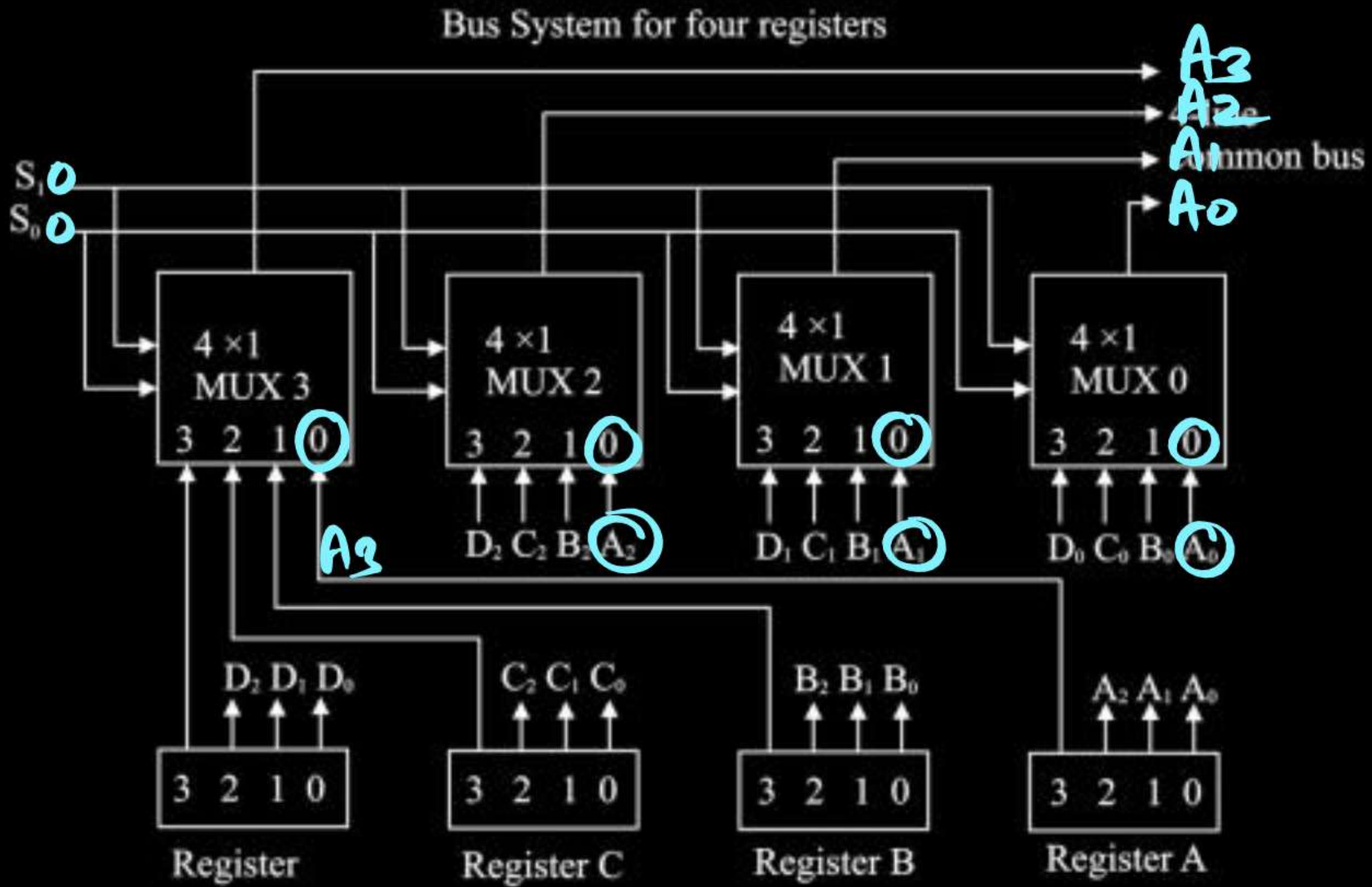
Function Table for Bus of Fig.

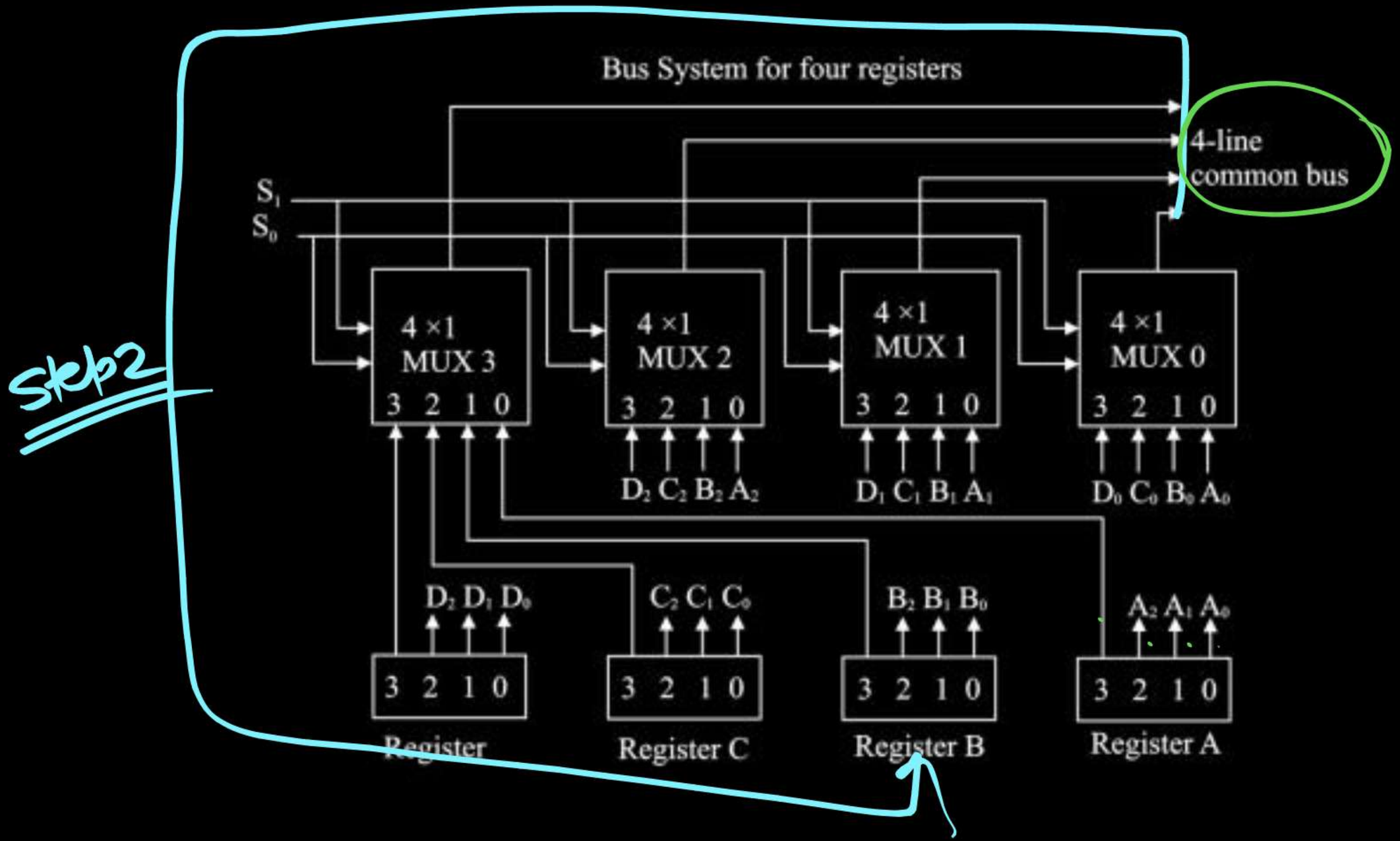
| $S_1$ | $S_0$ | Register selected |
|---|---|---|
| 0 | 0 | A |
| 0 | 1 | B |
| 1 | 0 | C |
| 1 | 1 | D |

# How Data is Transferred ?

Register A to Register B

$R_A \rightarrow R_B$


Bus System for four registers
A3
A2
A1
A0
common bus
S1
S0
4 ×1
MUX 3
4 ×1
MUX 2
4 ×1
MUX 1
4 ×1
MUX 0
3 2 1 0
D2 C2 B2 A2
D1 C1 B1 A1
D0 C0 B0 A0
A3
D2 D1 D0
C2 C1 C0
B2 B1 B0
A2 A1 A0
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
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
$D_2$ $C_2$ $B_2$ $A_2$
$D_1$ $C_1$ $B_1$ $A_1$
$D_0$ $C_0$ $B_0$ $A_0$
$D_2$ $D_1$ $D_0$
$C_2$ $C_1$ $C_0$
$B_2$ $B_1$ $B_0$
$A_2$ $A_1$ $A_0$
3 2 1 0
3 2 1 0
3 2 1 0
3 2 1 0
Register
Register C
Register B
Register A
Step2

# ② WHY Rout & Rin is Needed ?

Soln Register output Connected to the MUX
then MUX to Common Bus
then Common Bus to Respective Register

Process

When Rout is set to '1' then that Respective Register Content (Data) Load into the MUX (Multiplexer). then MUX to Common Bus, & Here Common Bus is Connected to All Registers, then the Register which having Rin is Set to 1 in that Respective Register Bus Content in Loaded in that Register.

∴

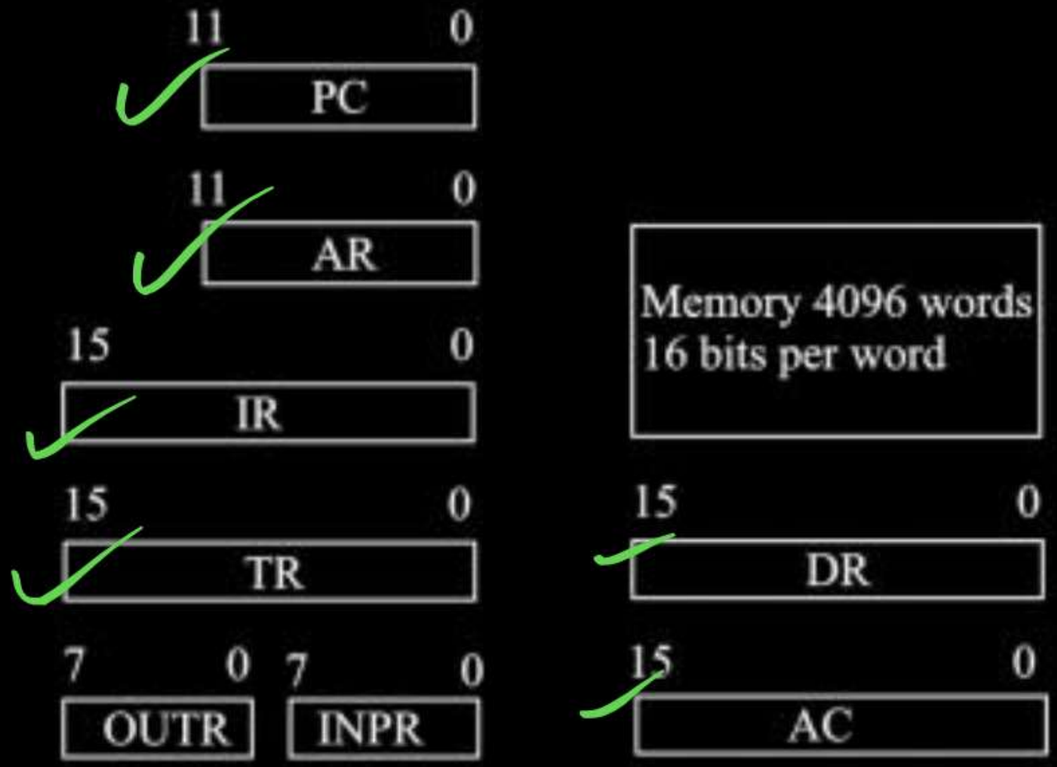
$R_A \rightarrow R_B$

$R_A$ to $R_B$ : $R_{Aout}$ , $R_{Bin}$ .

$R_{Aout} = 1$ Content of Register A Loaded into MUX then MUX to Common BUS

Now $R_{Bin} = 1$ then Content of BUS Load into Register $R_B$.

# Design a Small Computer



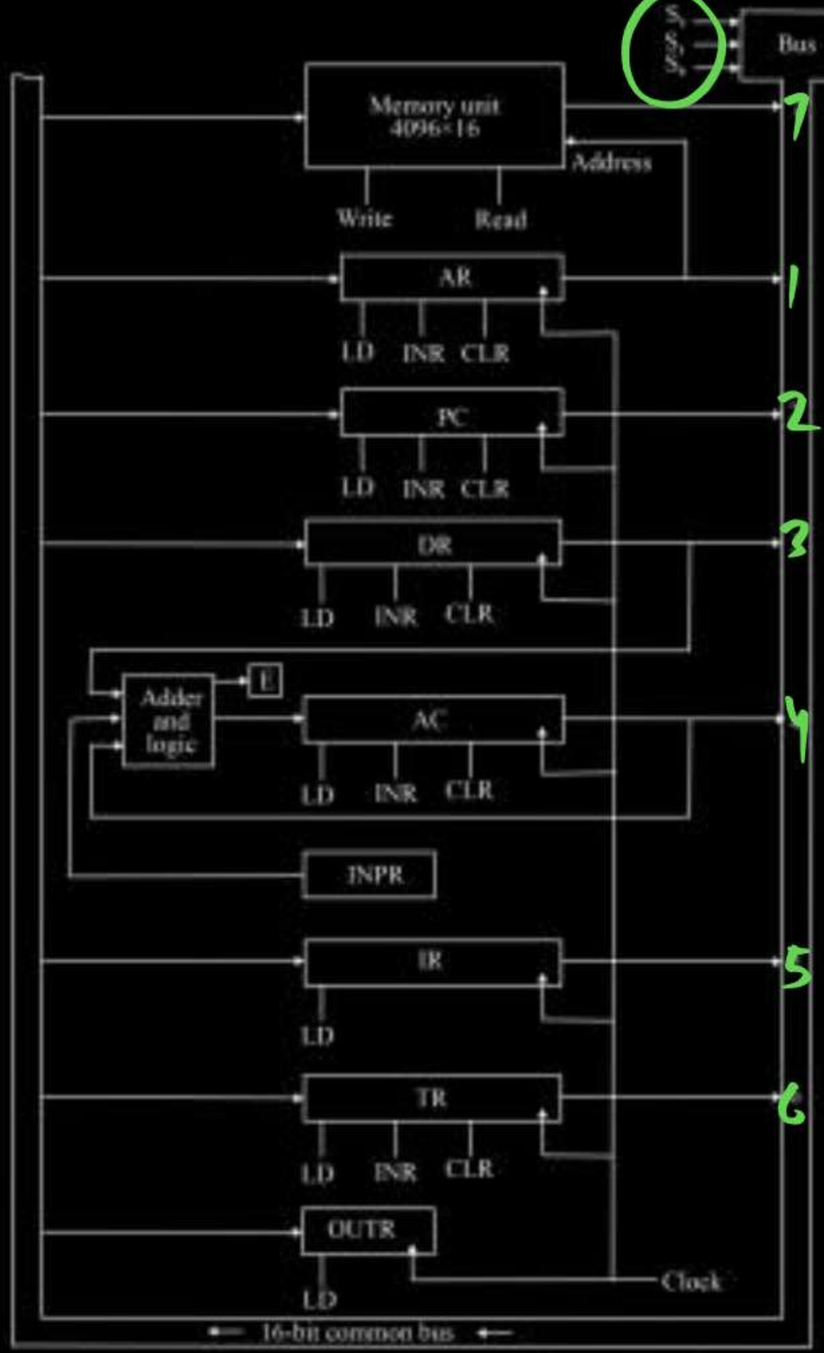

Basic computer registers and memory

4096 x 16

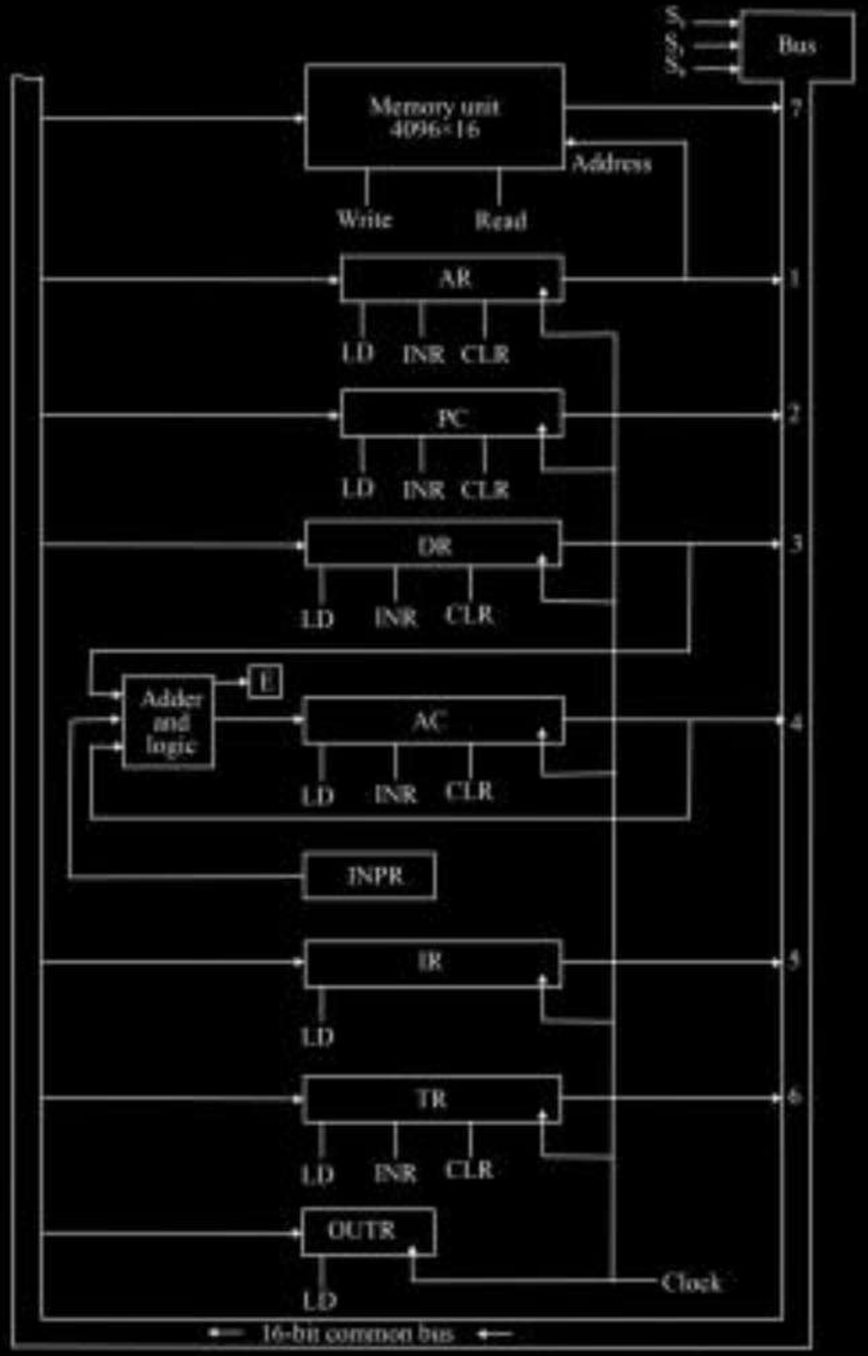
$2^{12}$ x 16 bit

Address = 12 bit

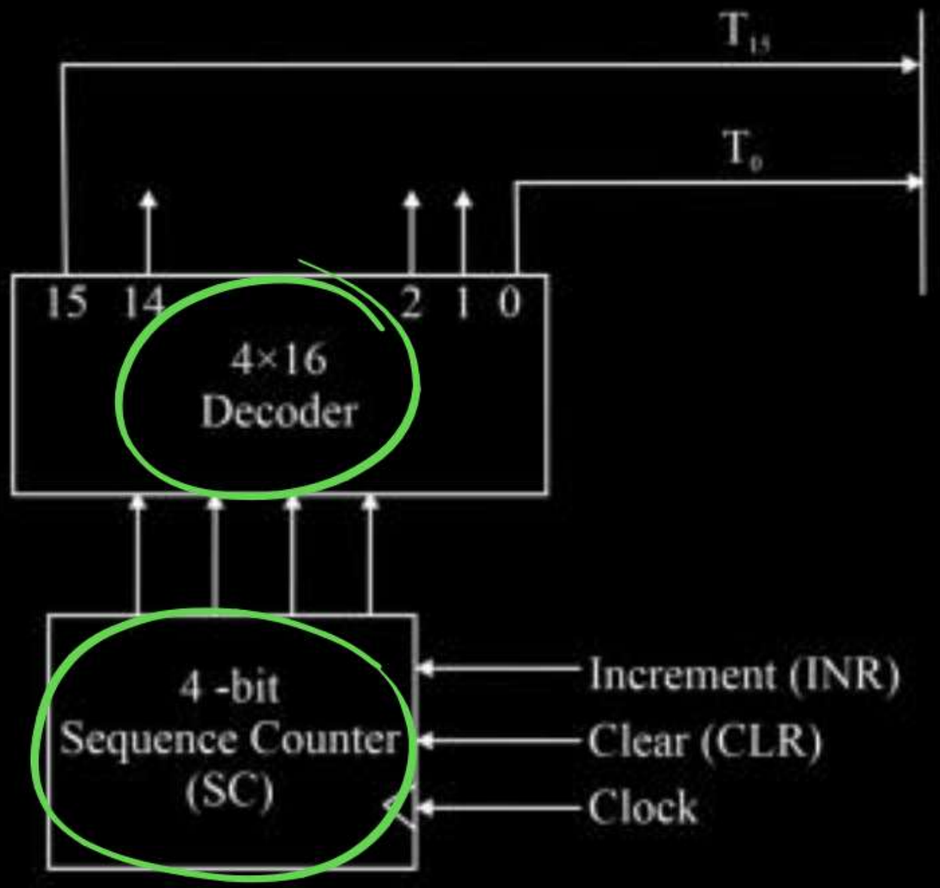
Data line = 16 bit

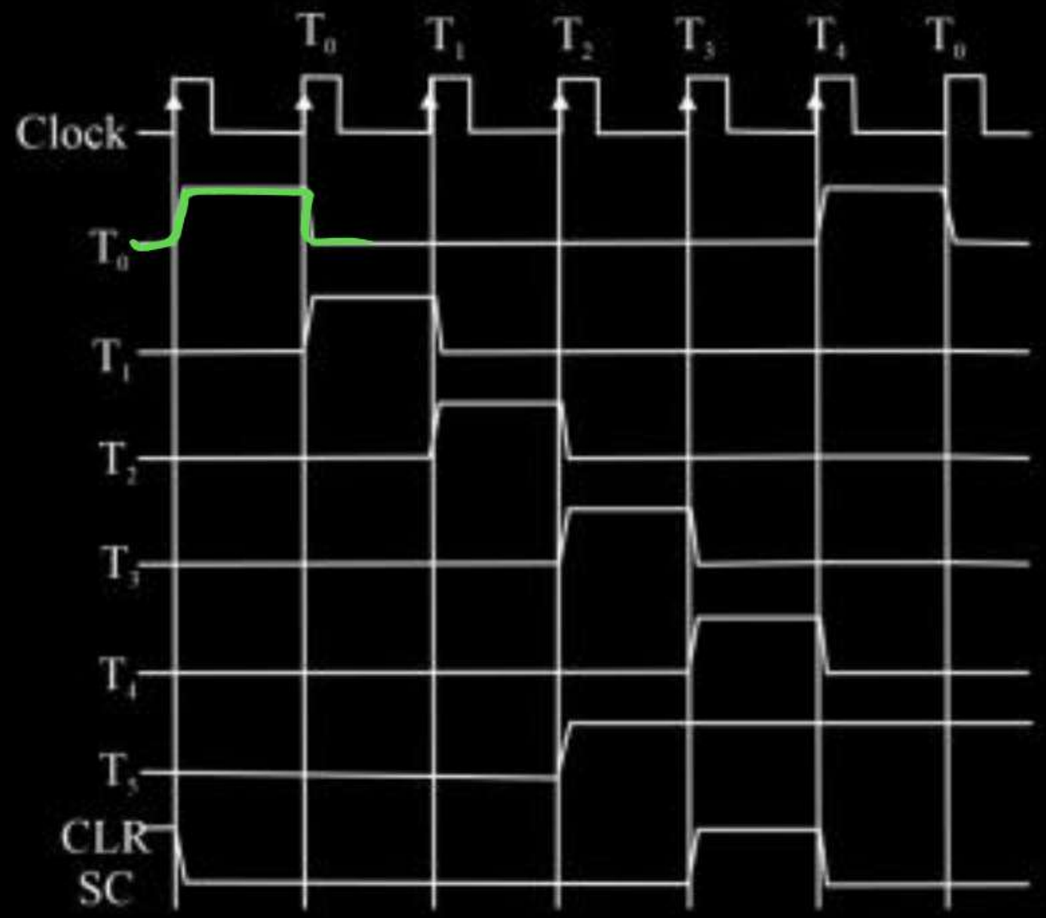



Basic computer registers connected to a common bus

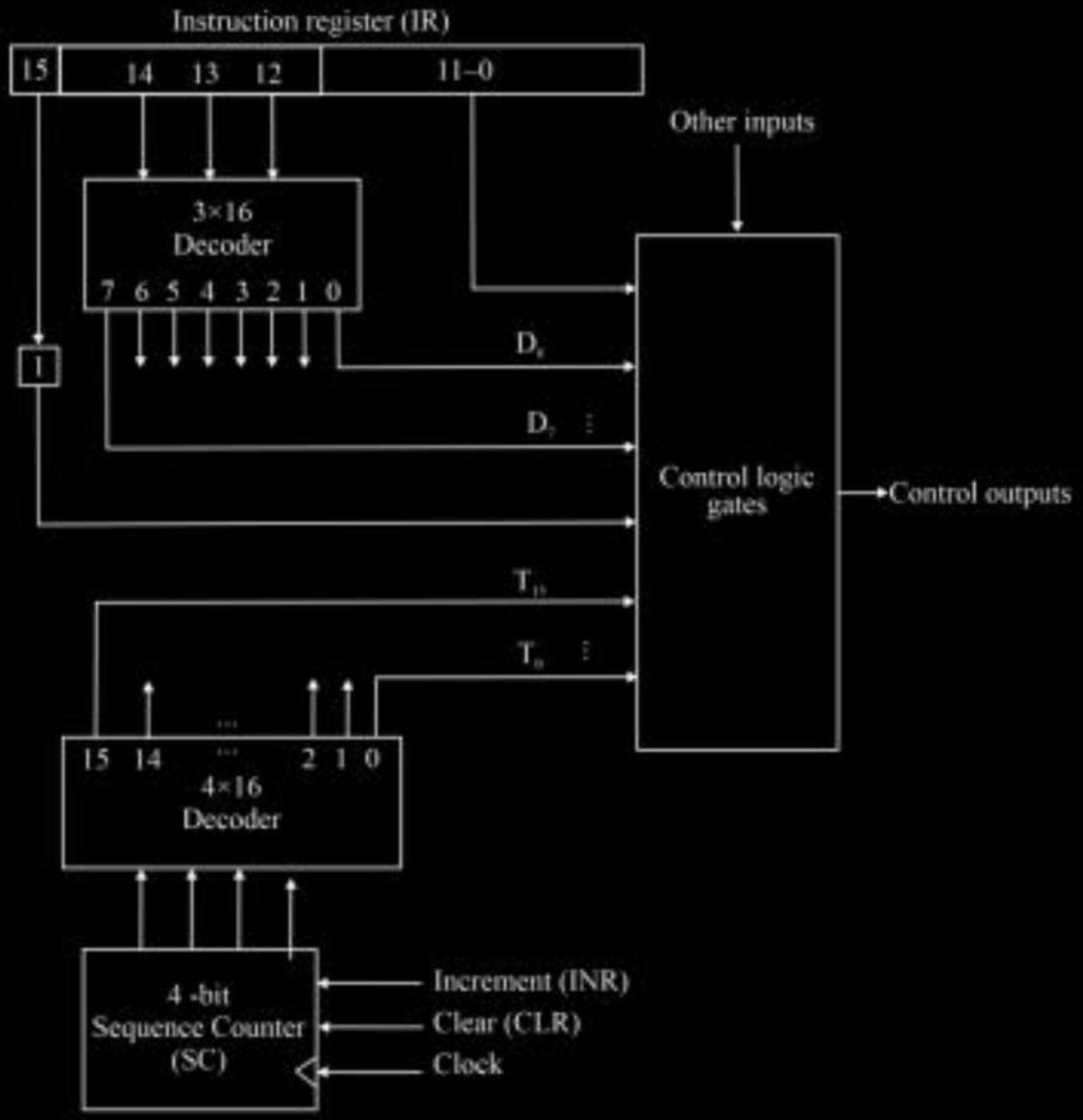



Basic computer registers connected to a common bus

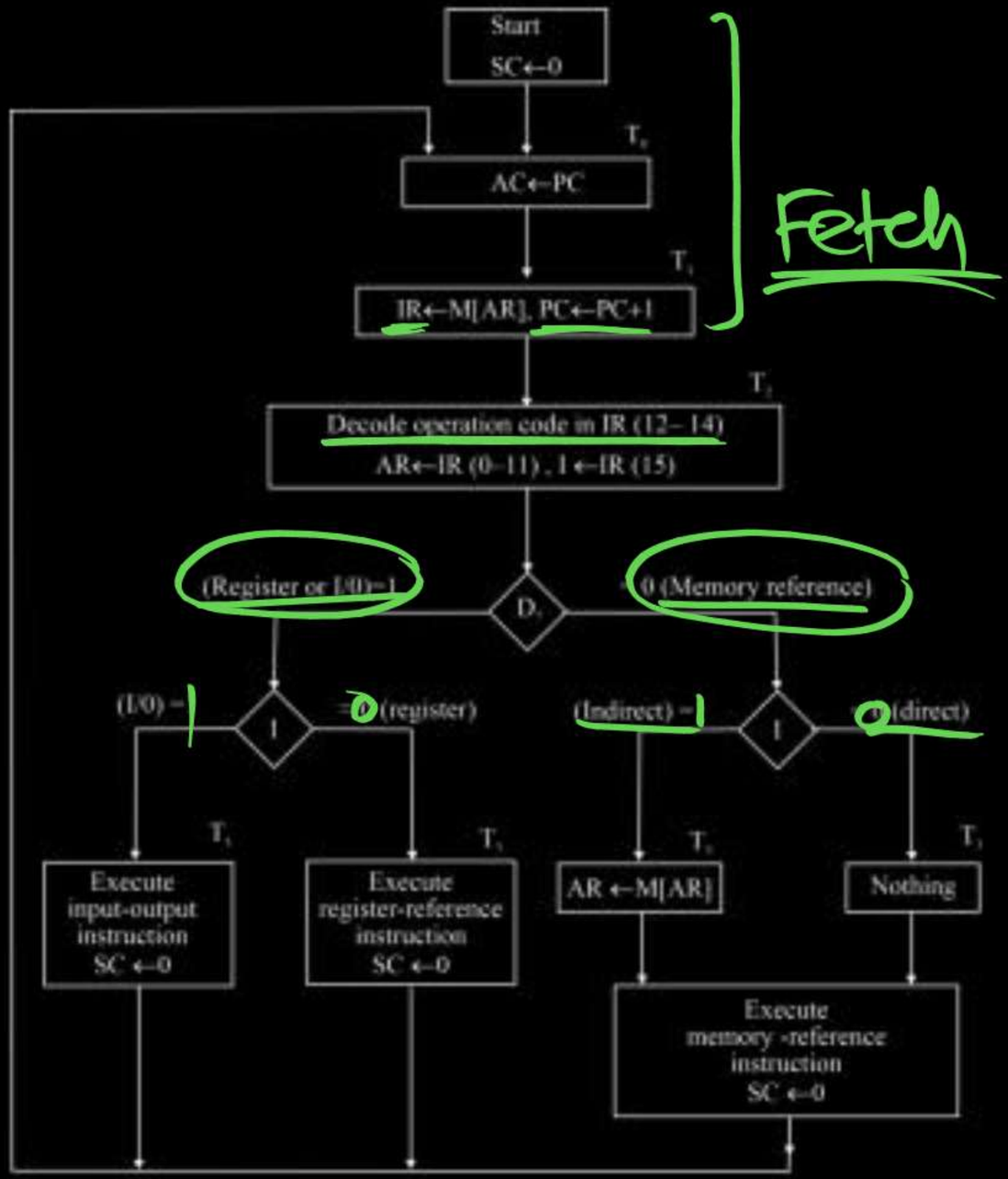



Example of control timing signals




Control unit of basic computer




Flow chart for instruction cycle (initial configuration)

Figure 4-3 Bus system for four registers.

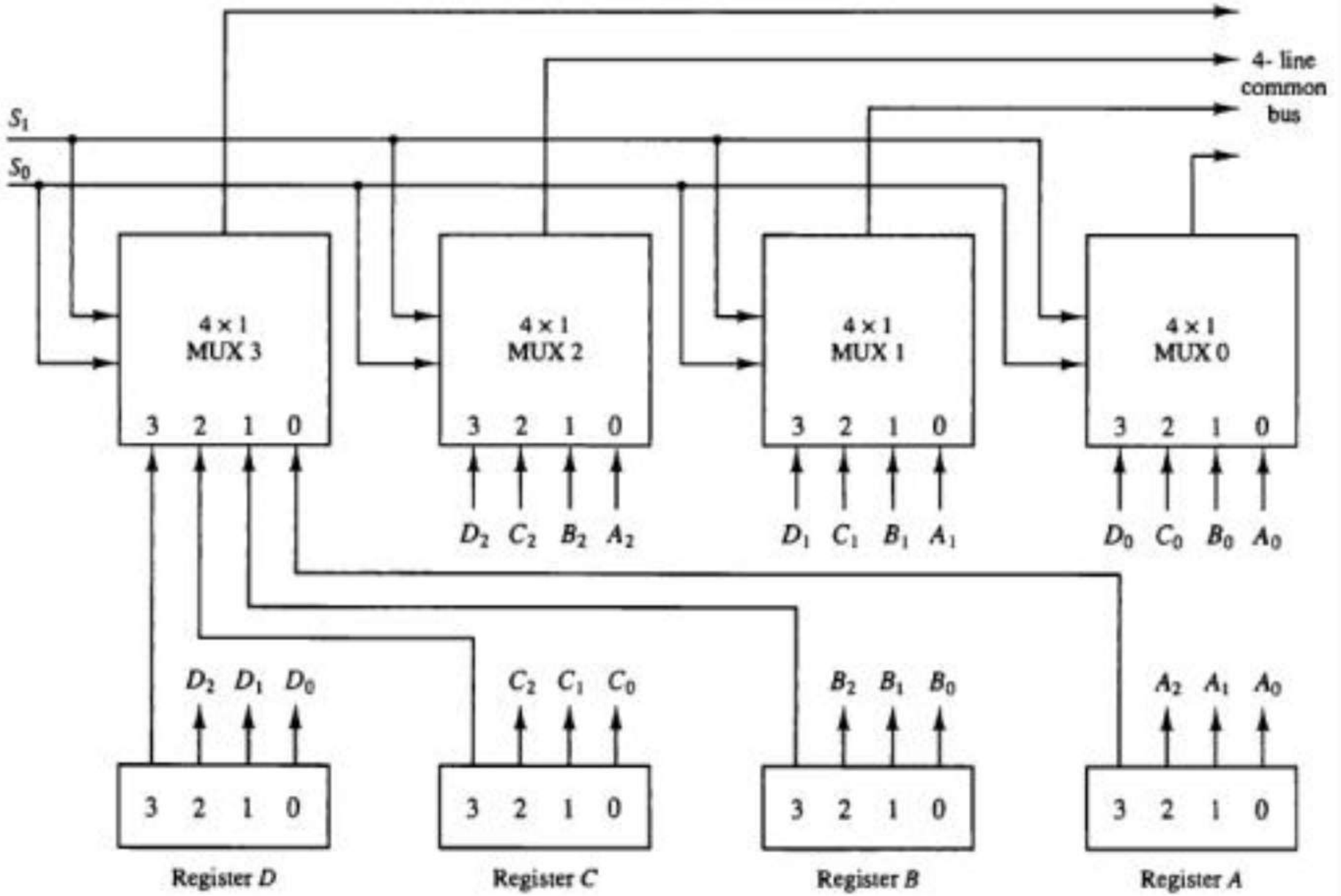


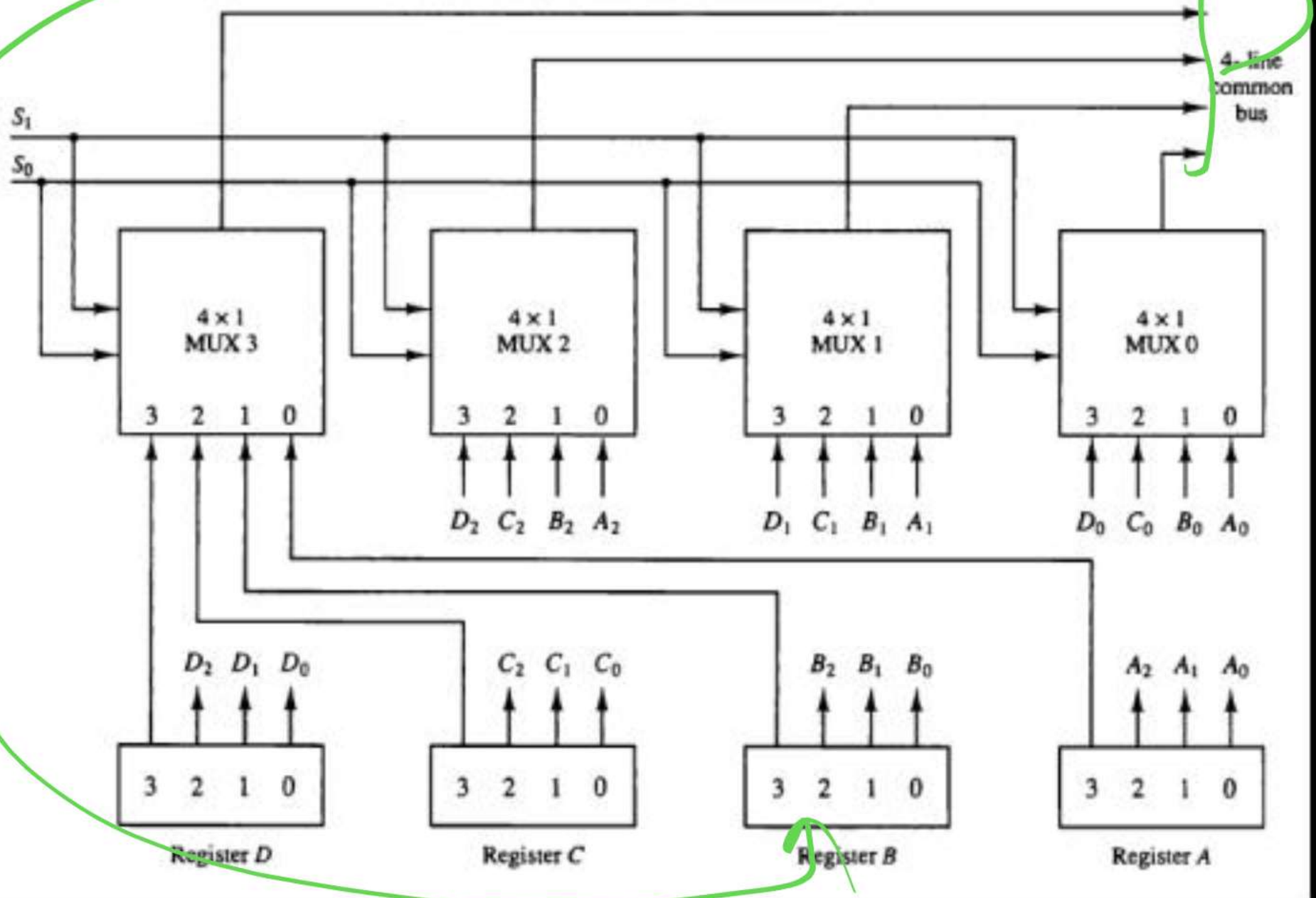
**TABLE 4-2** Function Table for Bus of Fig. 4-3

| $S_1$ | $S_0$ | Register selected |
|---|---|---|
| 0 | 0 | *A* |
| 0 | 1 | *B* |
| 1 | 0 | *C* |
| 1 | 1 | *D* |

# How Data is Transferred ?

Register A to Register B

Figure 4-3 Bus system for four registers.



# Computer Design

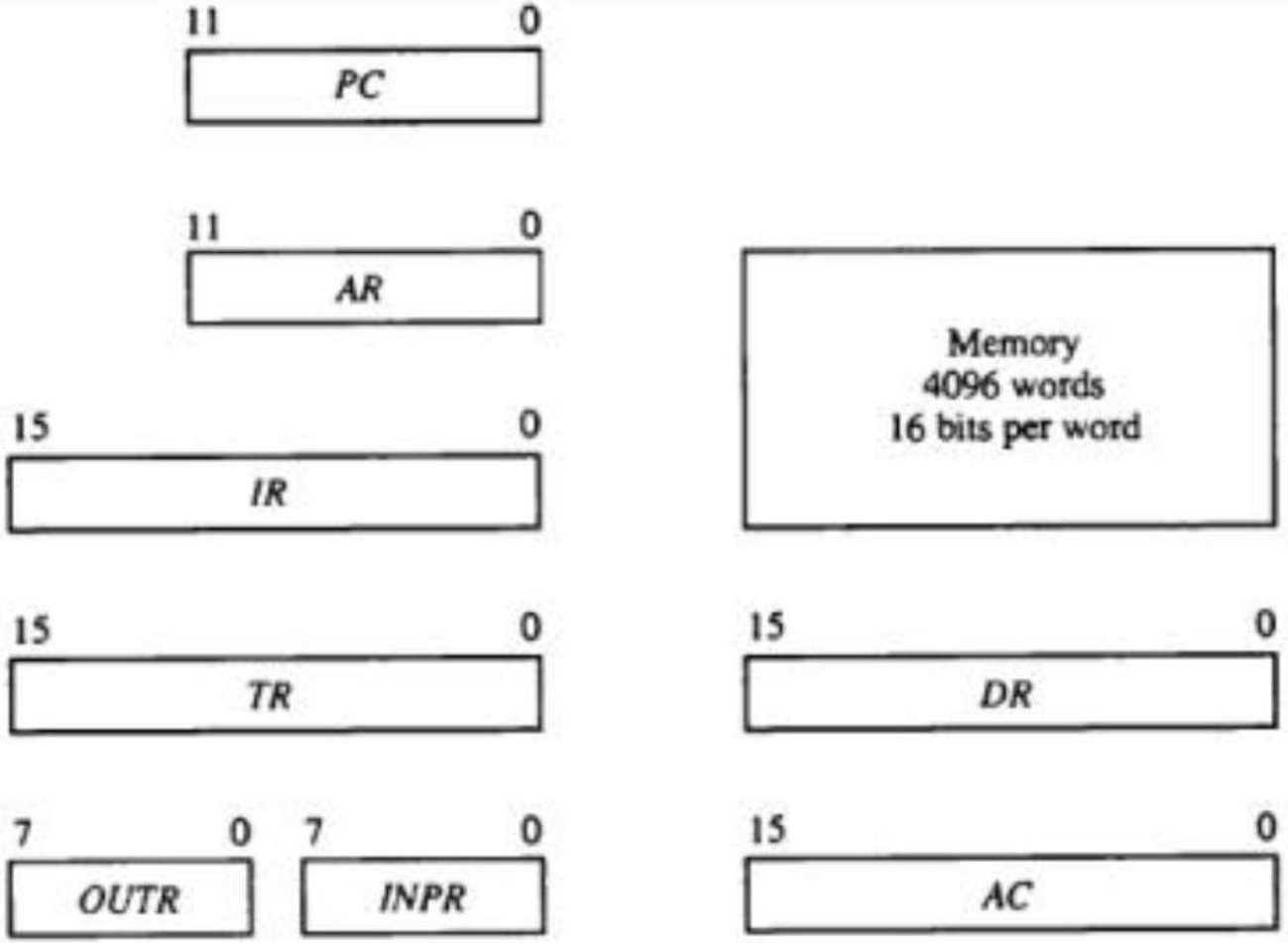



**Figure 5-3** Basic computer registers and memory.

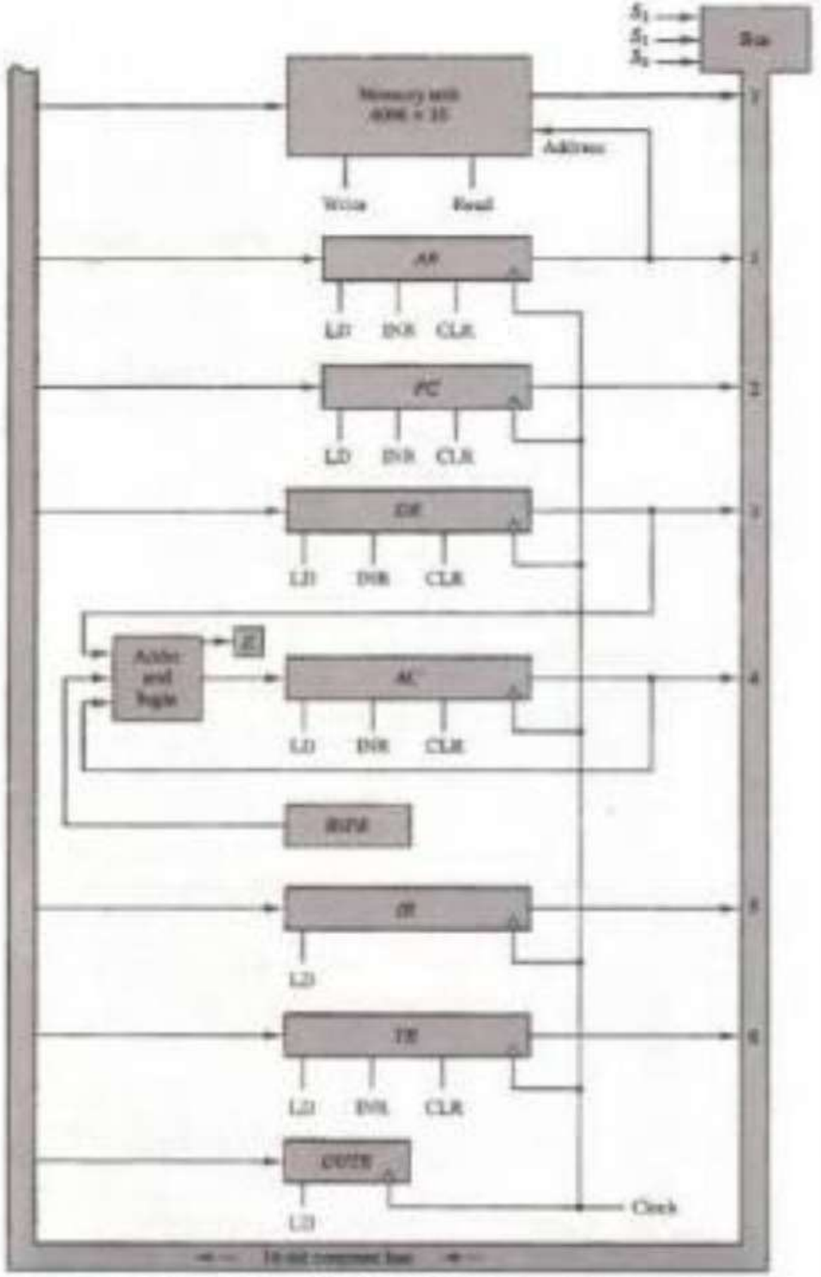



Figure 5-4 Basic computer registers connected to a common bus.

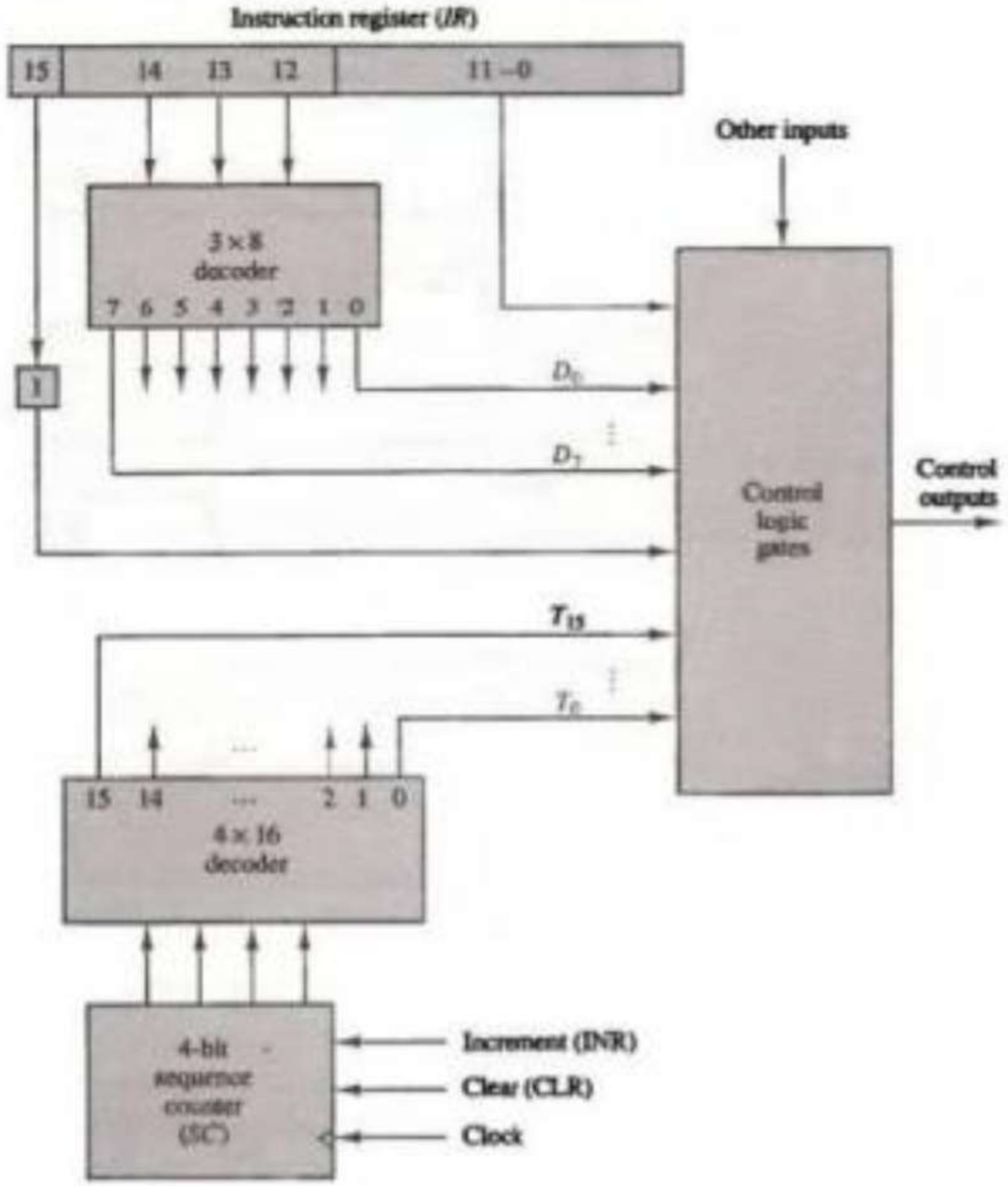



**Figure 5-6** Control unit of basic computer.

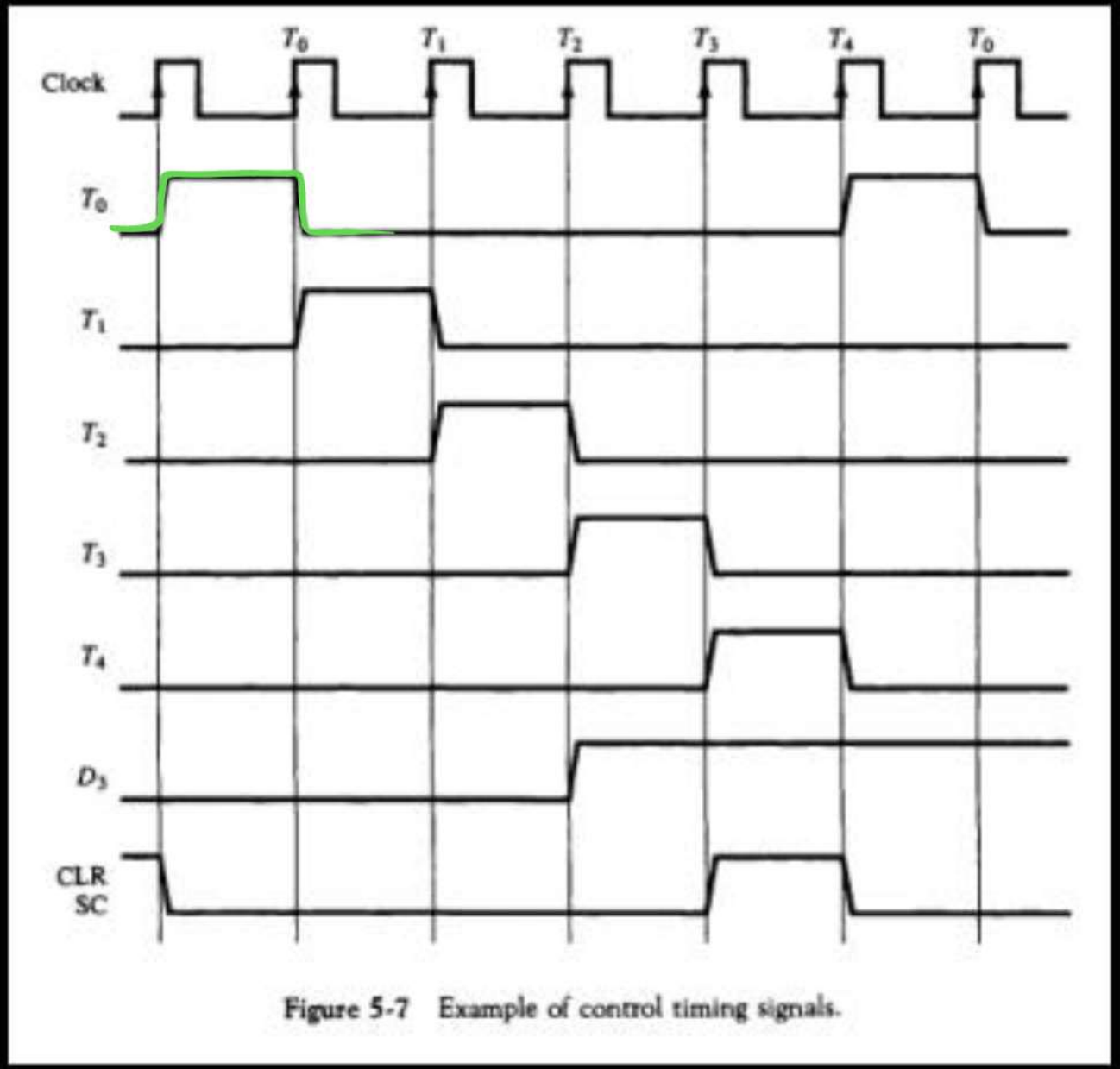



Figure 5-7 Example of control timing signals.

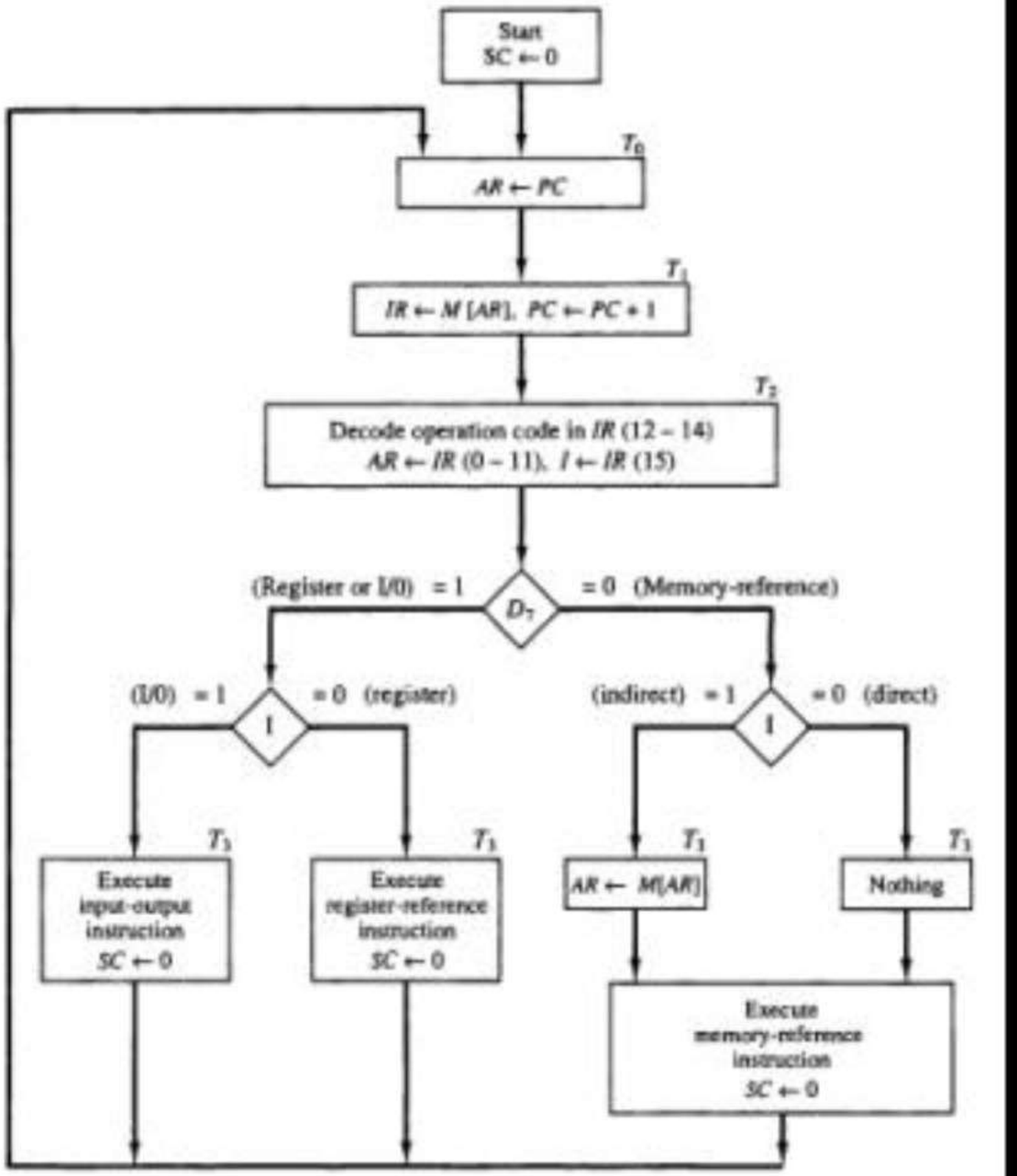



Figure 5-9 Flowchart for instruction cycle (initial configuration).

# Micro operation.

Working & DATA PATH. ($R_A$ to $R_B$)

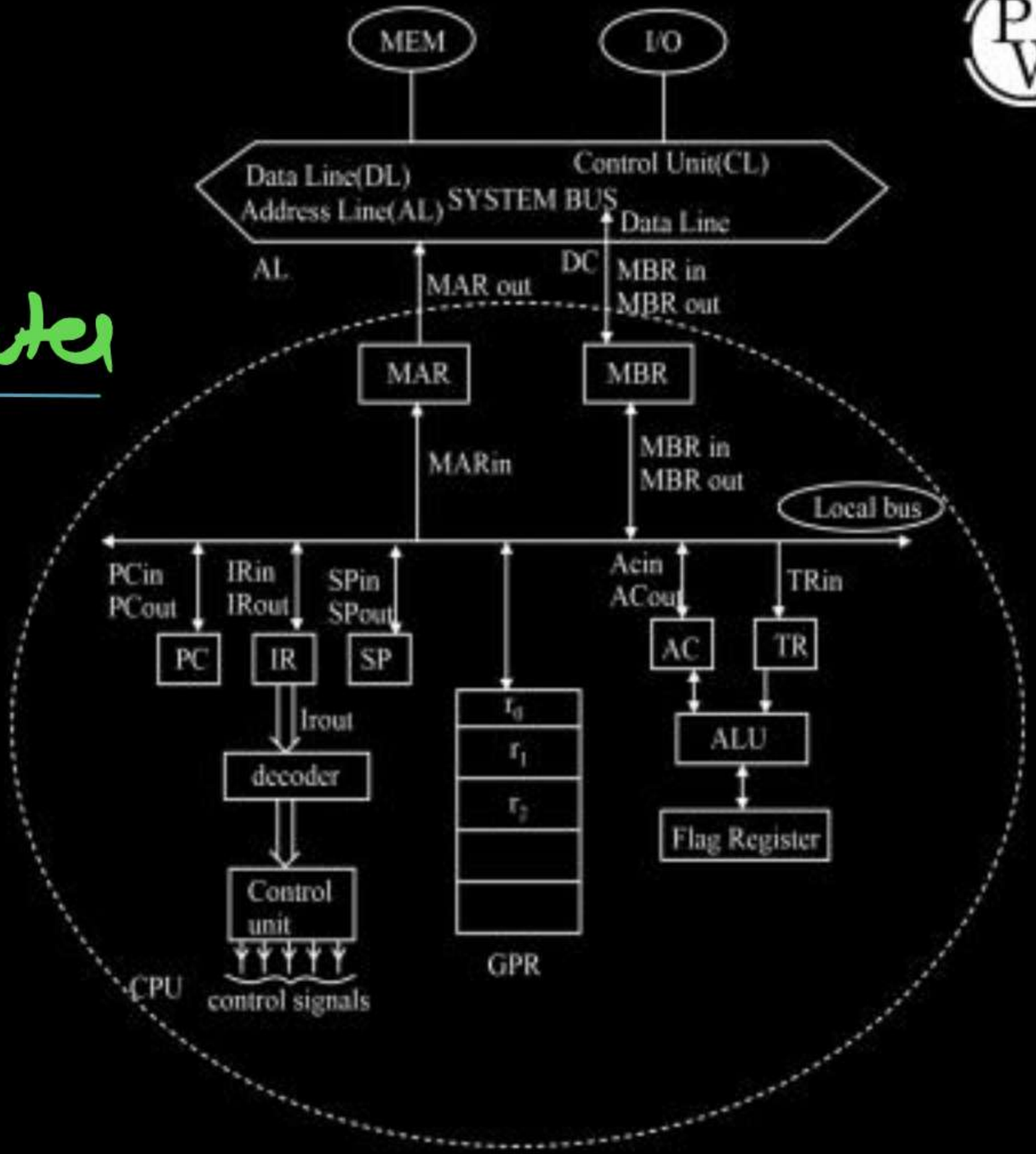
# ALU, Datapath & Control Unit

- Micro Instruction
- Micro Program
- Control Unit Design

# Structure of Computer

## Component of the Computer

1. Memory
2. CPU
3. I/O.



# Component of Computer

1. CPU , 2. Memory & 3. IO

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

. Common Bus

· Multiplexer .

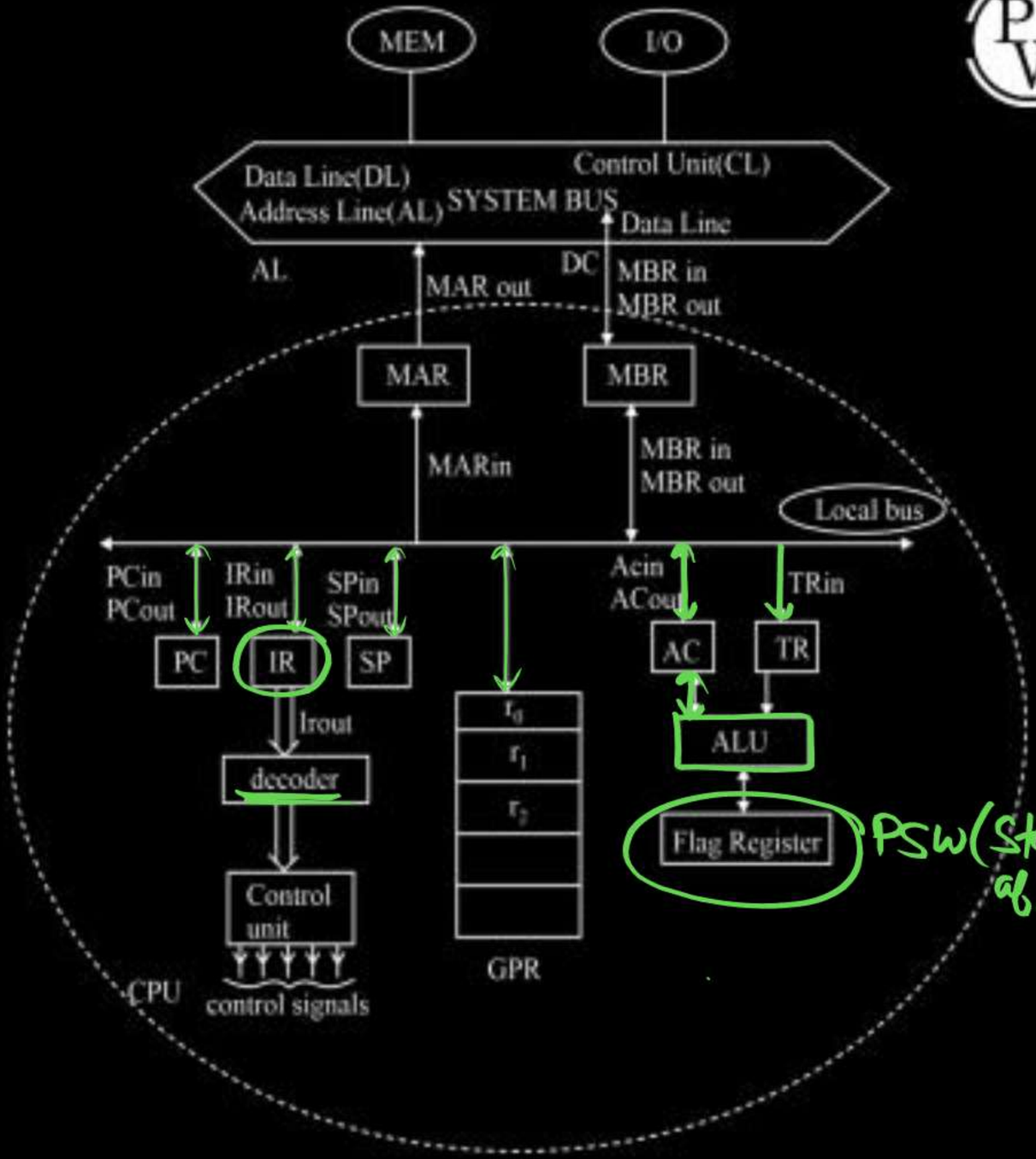
MAR/AR [Memory Address Register]: It Contain Address of Memory Location.
It is Connected to the Address line (AL) of the System BUS.

MBR/MDR/DR [Memory Data Register]: It Contain Memory Content (Instruction & DATA).
It's Connected to the Data line (D.L) of the System BUS.

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

# Structure of Computer

PC
MAR
MBR
IR
AC
TR
SP
PSW
GPR



- 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..)

Common Internal BUS

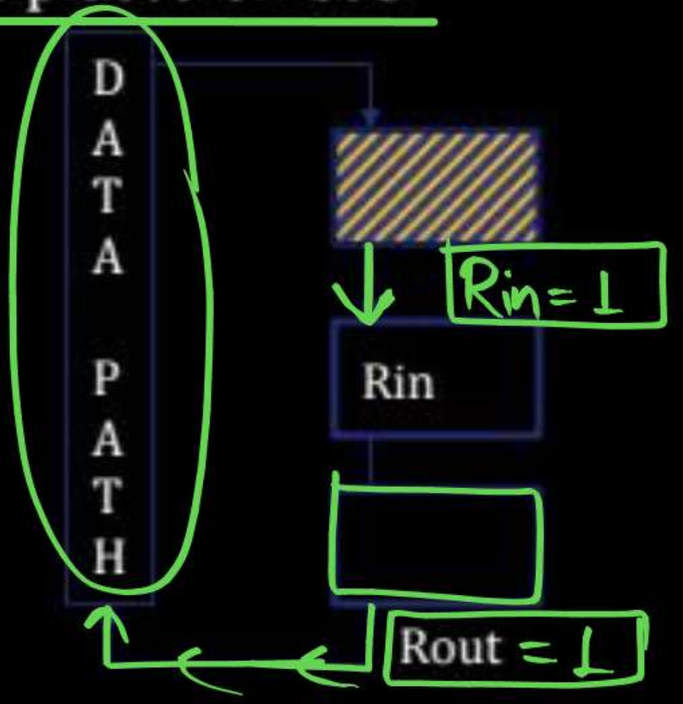
Every Register has 2 switch Rin & Rout

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: ALU, MUX, Register & other Component How they Process, Path for Processing.

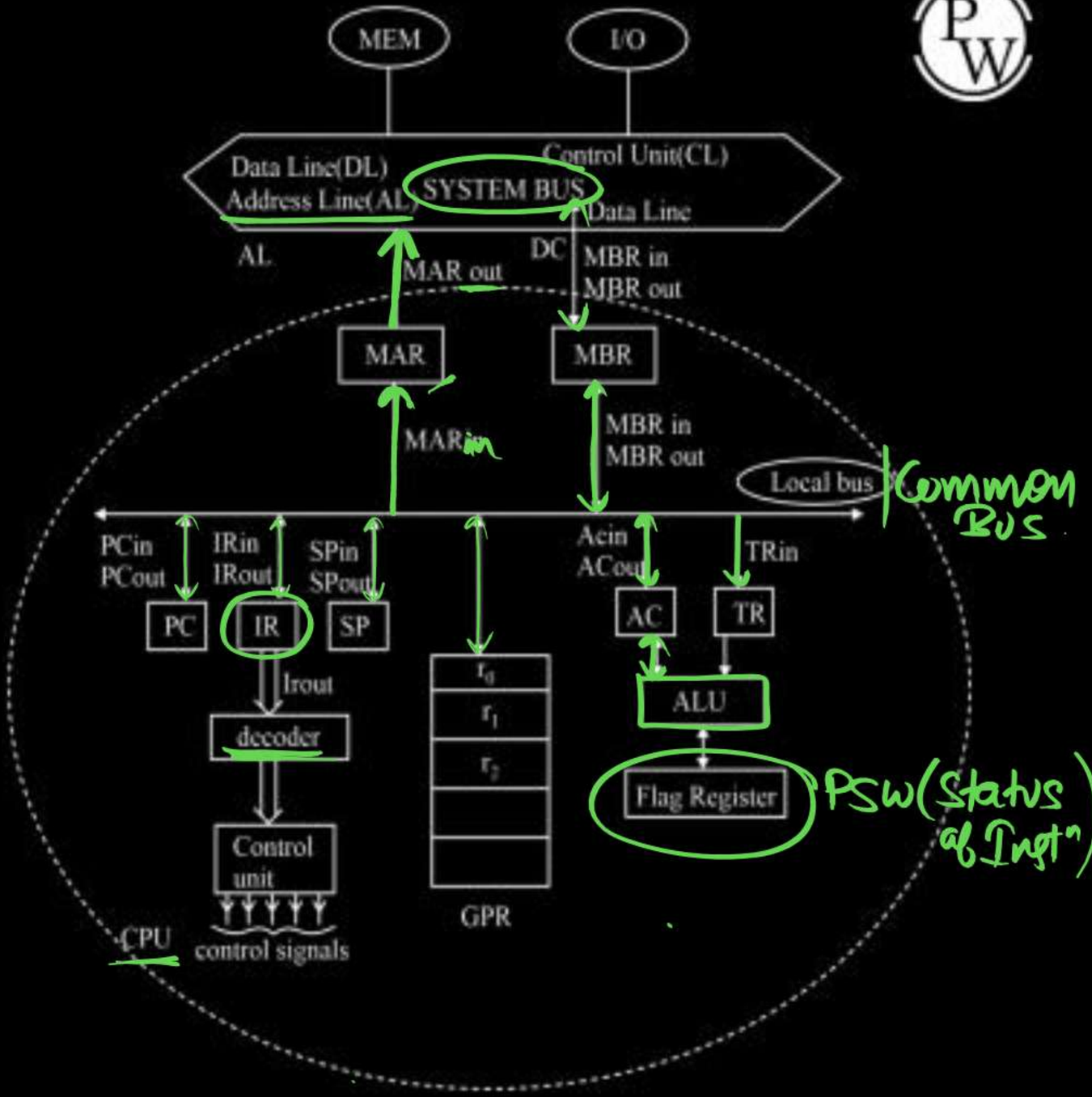


# Structure of Computer

MARin

MARout

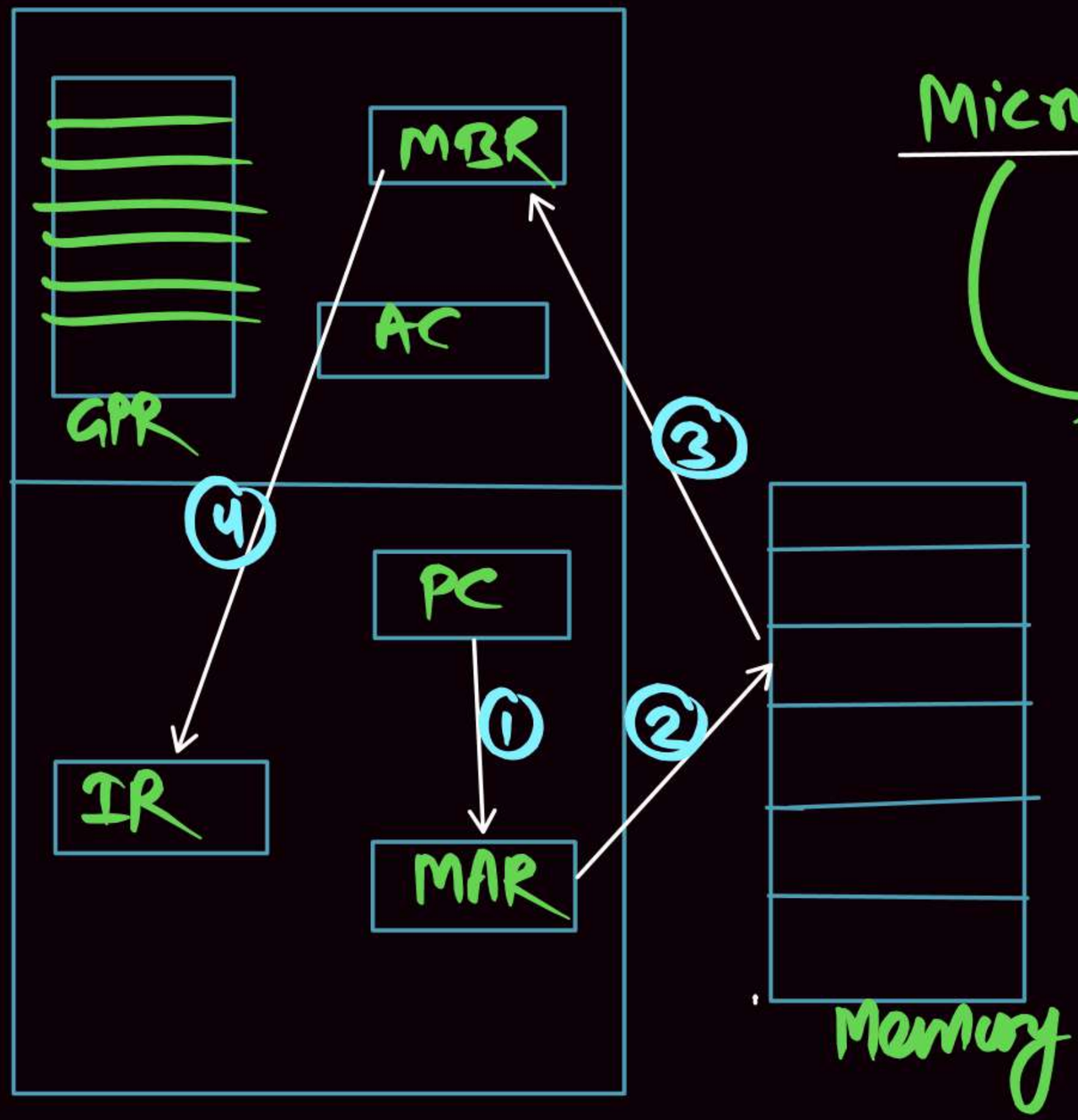
IRin . IRout



# Micro operation.

① **Fetch cycle:** To Fetch the Instruction from Memory to CPU.

MEM to CPU (IR).



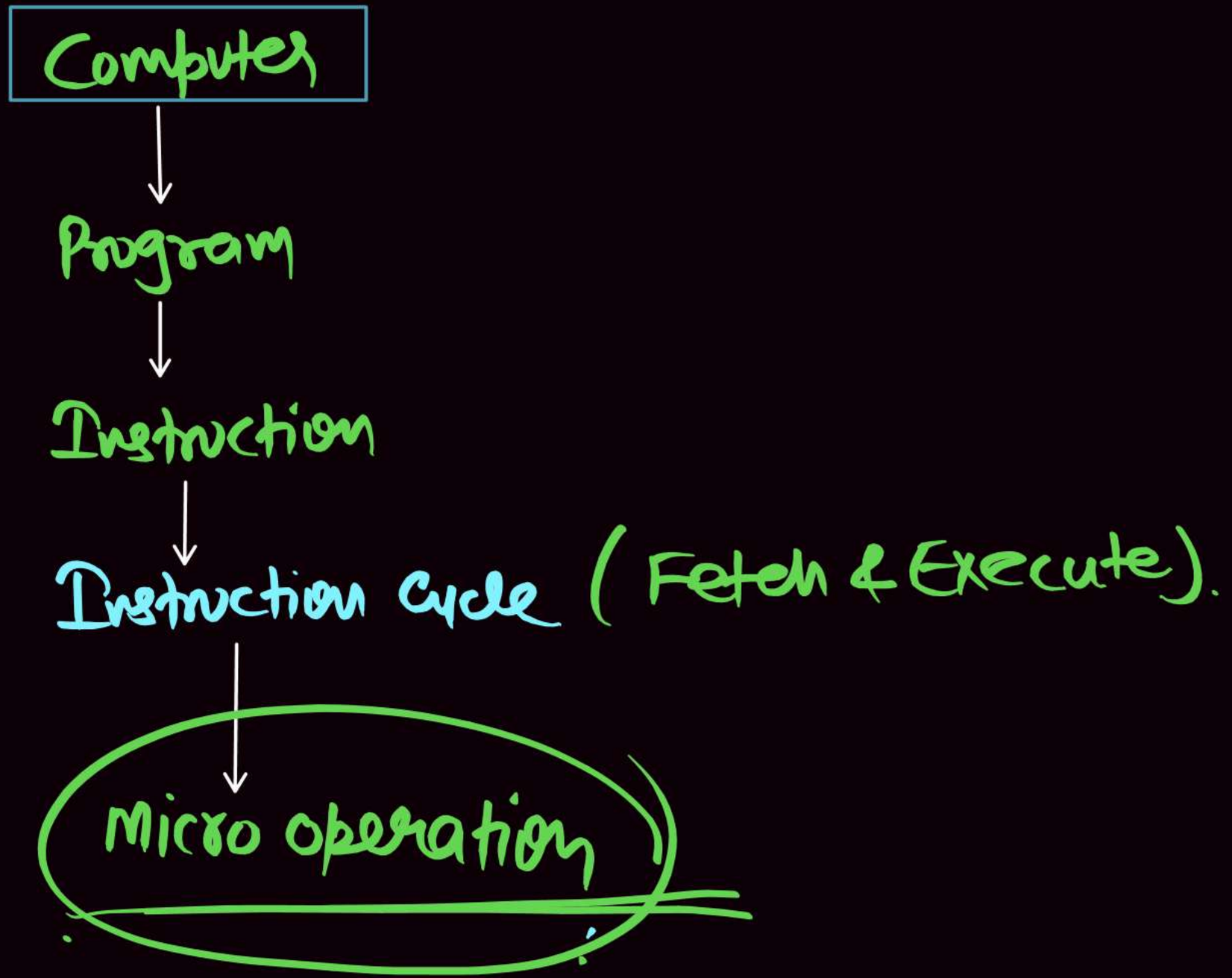

# Micro operation in Fetch cycle:

PC → MAR ; PCout MARin

MAR → Memory

Memory → MBR

MBR → IR


Computer
Program
Instruction
Instruction cycle ( Fetch & Execute).
Micro operation

# 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.

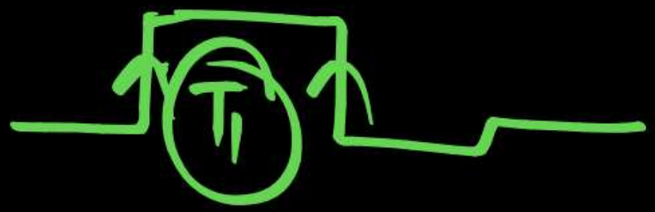
eg

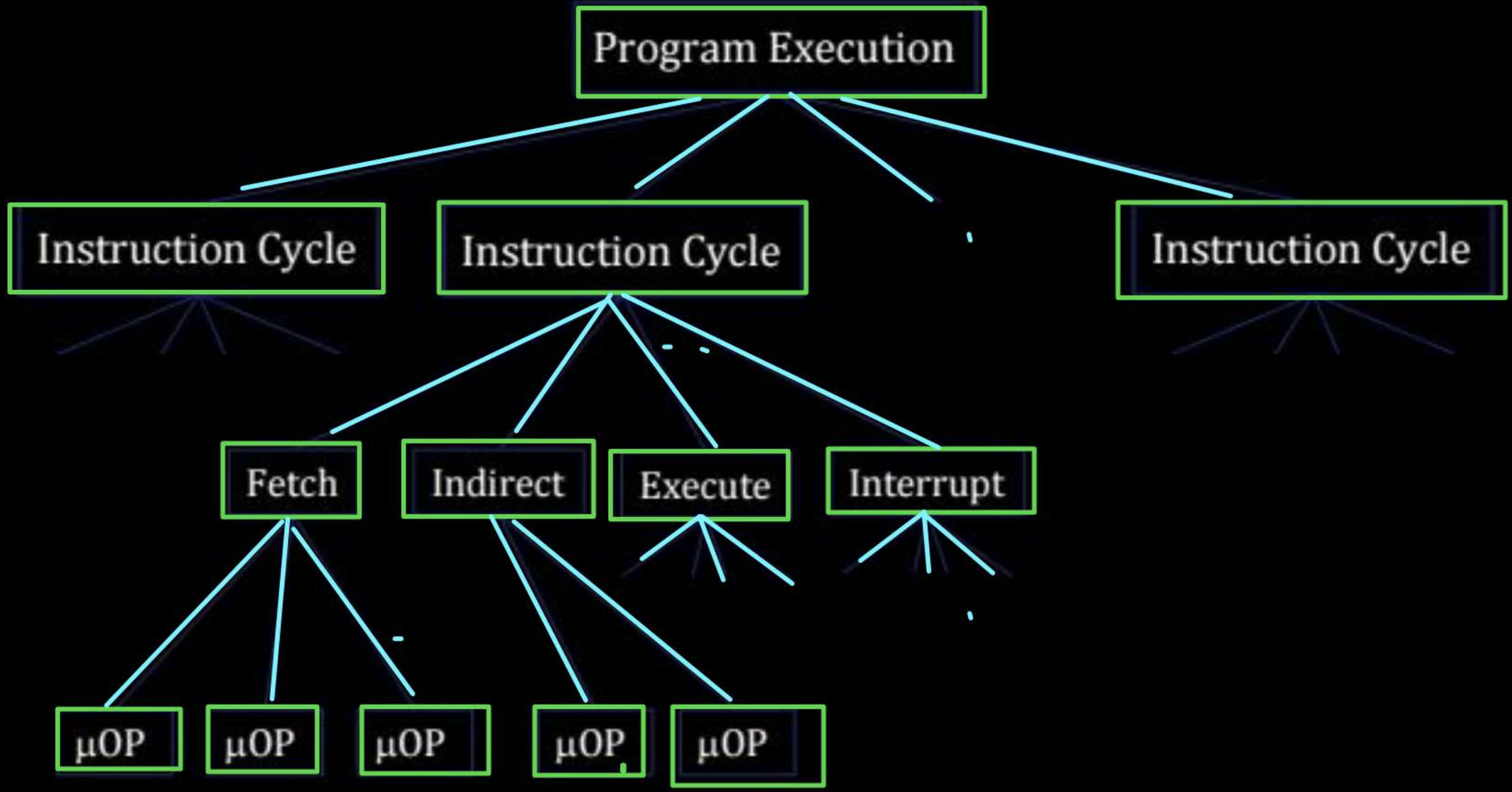
Machine Instruction: MOV $r_0$, $r_1$

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

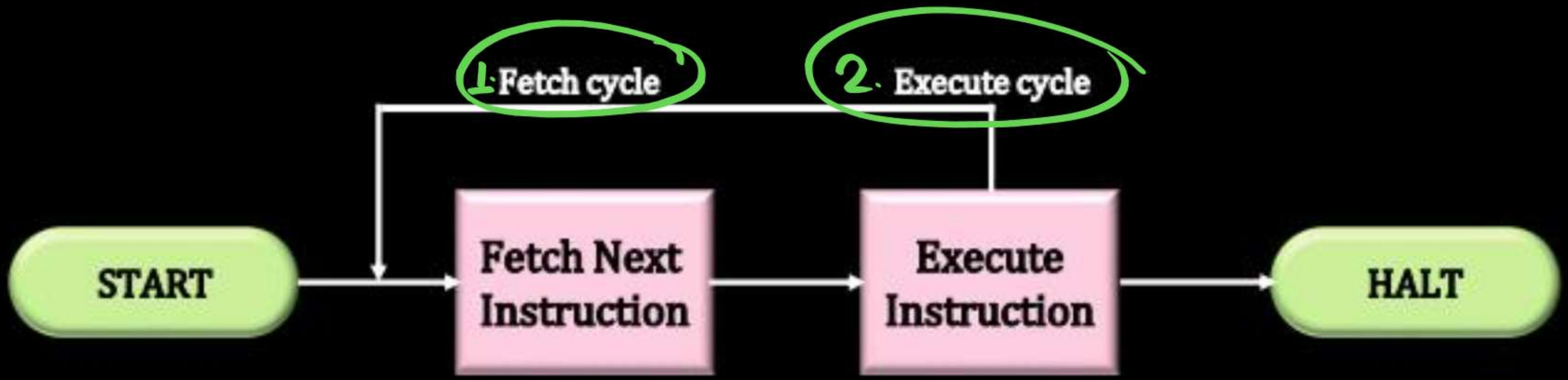
$\gamma_{1out}\ \gamma_{0in}$

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






Constituent Elements of a Program Execution



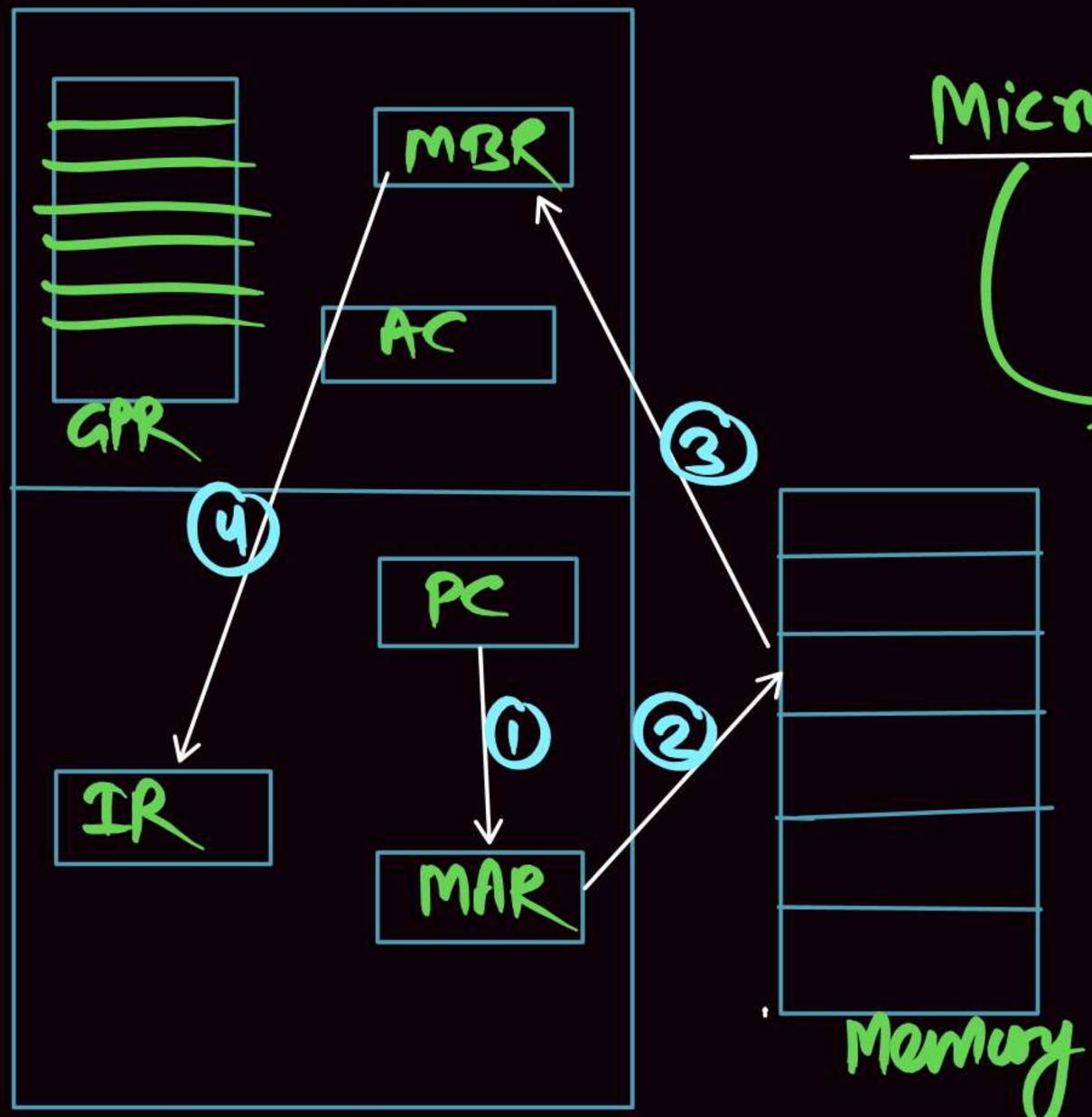

BASIC INSTRUCTION CYCLE

# Micro Program (μ - Program)

- Sequence of Micro operation [μ operation] to perform

  Some operation in the Hardware level is called microprogram.

Collection [sequence] of Micro Operation is Called Micro Program.




# Micro operation in Fetch cycle.

PC → MAR ; PCout MARin

MAR → Memory

Memory → MBR

MBR → IR

Memory to CPU (IR)

# 1. The Fetch Cycle

- Occurs at the beginning of each instruction cycle and causes an instruction to be fetched from memory

Four registers are involved:

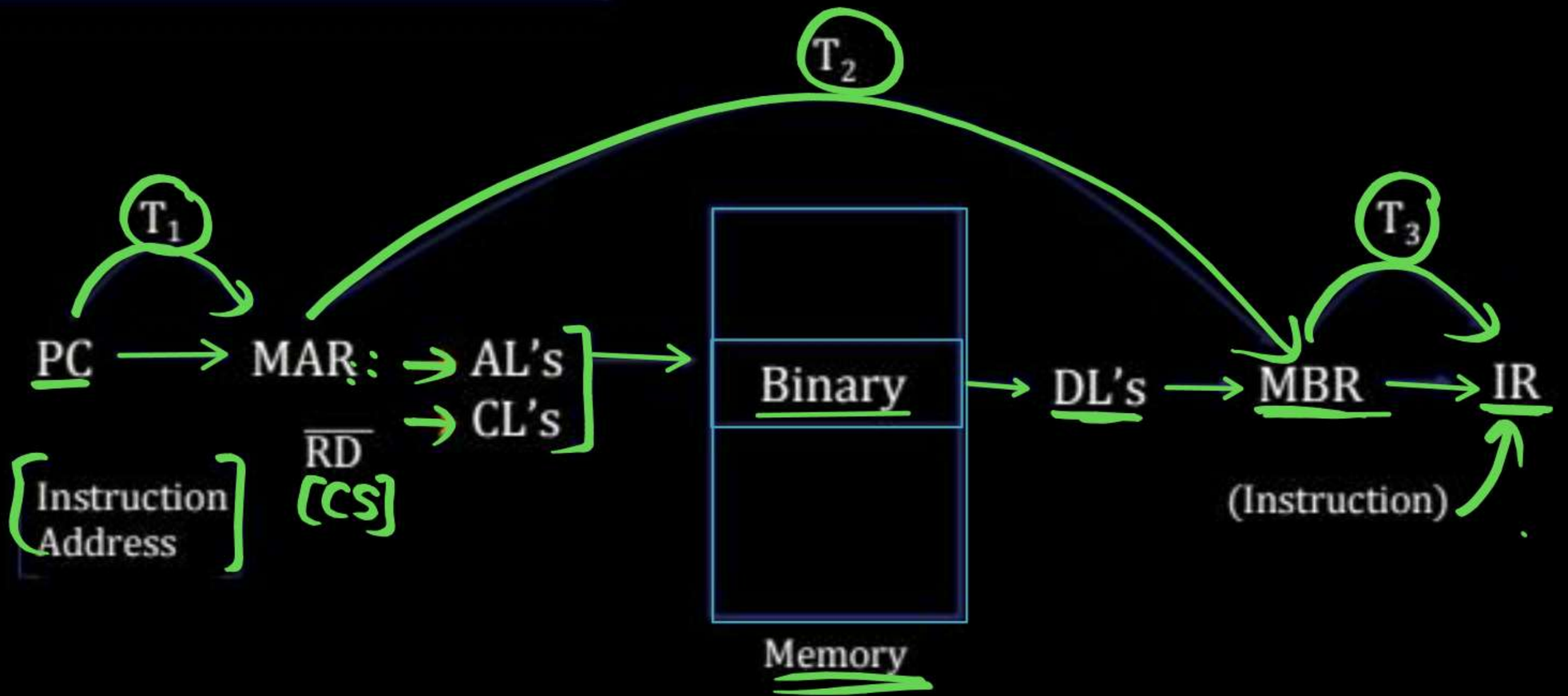
1. Memory Address Register (MAR)
   - Connected to address bus
   - Specifies address for read or write operation
2. Memory Buffer Register (MBR)
   - Connected to data bus
   - Holds data to write or last data read
3. Program Counter (PC)
   - Holds address of next instruction to be fetched
4. Instruction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

# Instruction Cycle

(1) Fetch Cycle: Instruction Fetch.

Hardware Design(H/W Design)

AL: Address Line
DL: Data Line
CL: Control Line

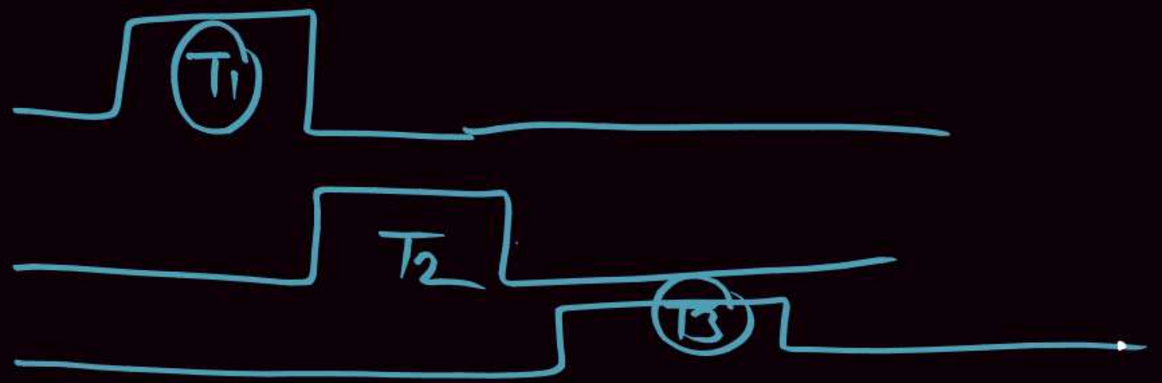


$T_1$: $PC \rightarrow MAR$ ; $PC_{out}$ $MAR_{in}$

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

$PC + I \rightarrow PC$ ; $PC_{out}$, $PC_{In}$.

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



# Micro program :

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

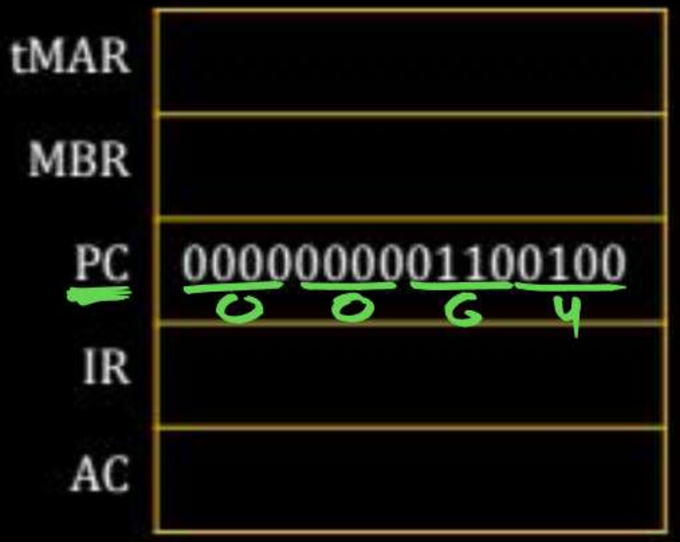
$T_2$ : $M[MAR] \rightarrow MBR$ ; $MAR_{out}$, $MBR_{in}$

$PC + I \rightarrow PC$ ; $PC_{out}$ $PC_{in}$

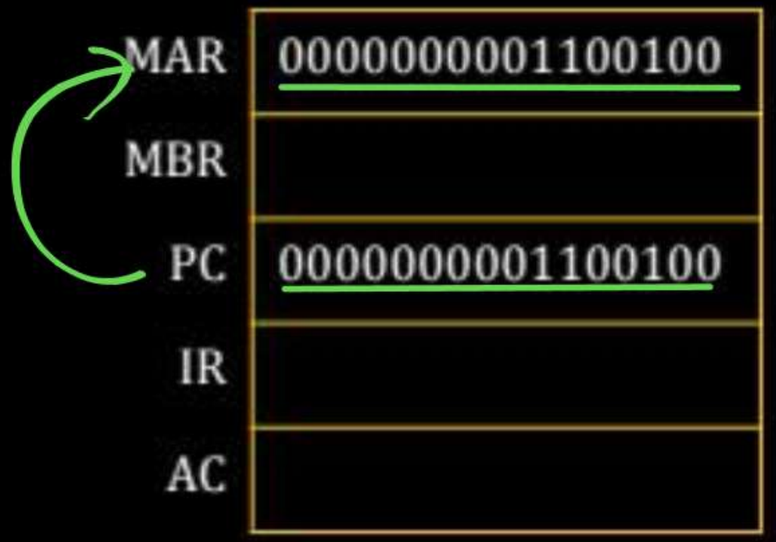
$T_3$ : $MBR \rightarrow IR$ ; $MBR_{out}$ $IR_{in}$

# Microprogram

| | | | |
|---|---|---|---|
| T1: PC → MAR; | $PC_{out}$ | $MAR_{in}$ | |
| T2: M[MAR] → MBR; | $MAR_{out}$ | $MBR_{in}$ | System Bus |
| PC + I → PC | $Pc_{out}$ | $PC_{in}$ | Local Bus |
| Increment | | | |
| T3: MBR → IR | $MBR_{out}$ | $IR_{in}$ | |



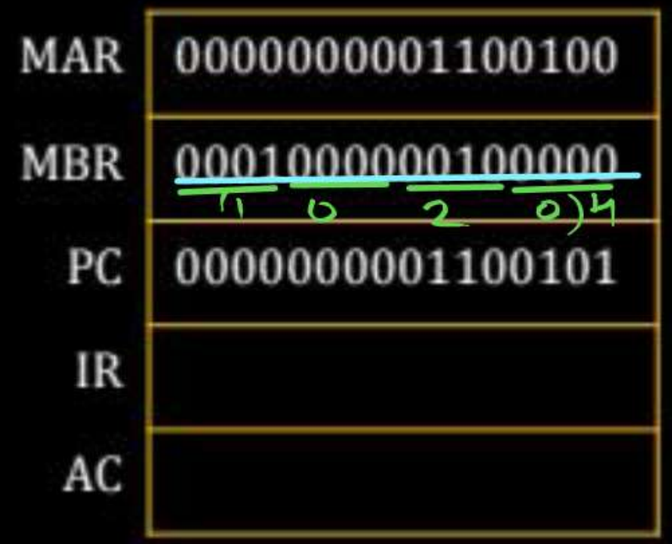

(a) Beginning (before $t_1$)



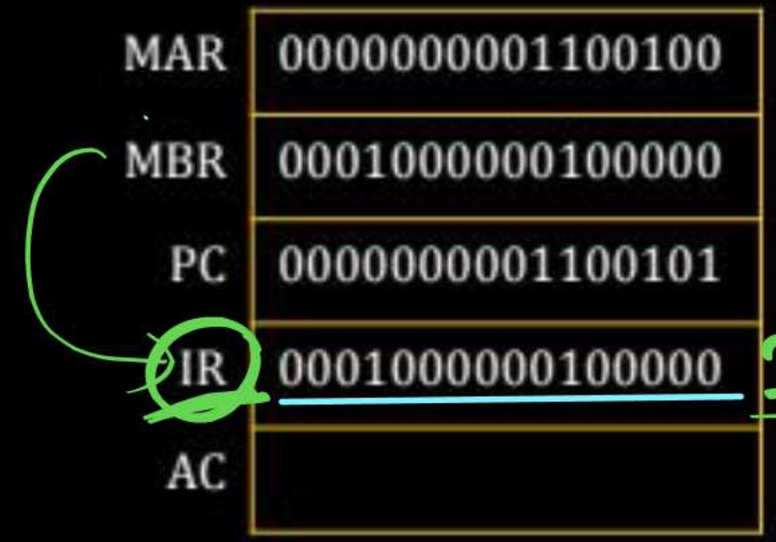

(b) After first step

Mem to CPU (IR).

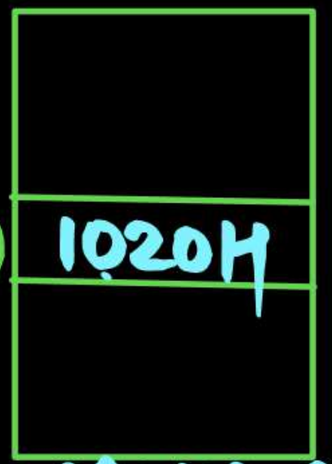
Sequence of events, fetch cycle




(c) After second step




(d) After third step



# Rules for Micro Operations Grouping

$T_1$: PC → MAR

$T_2$: M[MAR] → MBR
PC + I → PC

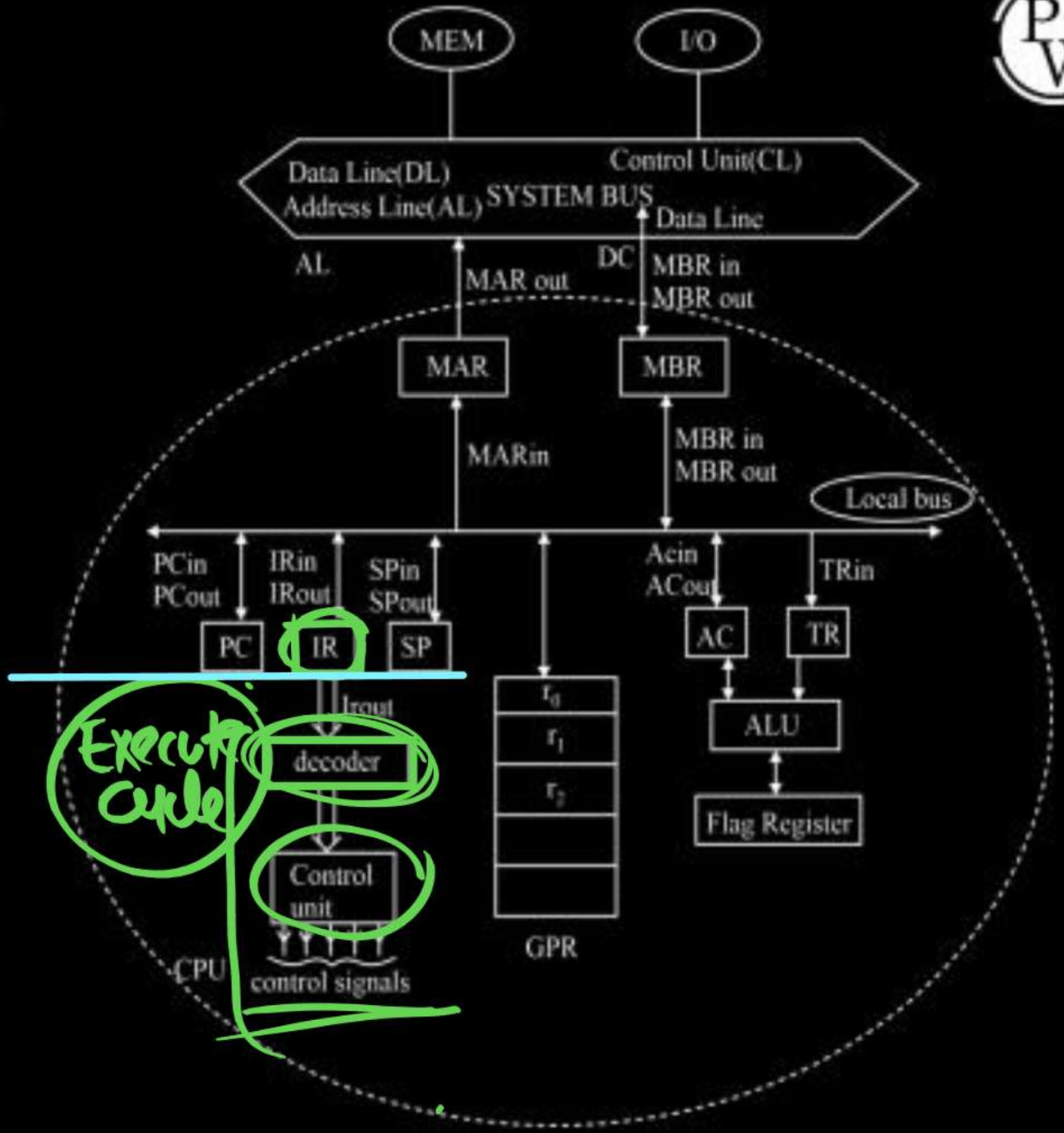
$T_3$: MBR → IR

1. Proper sequence must be followed
   - MAR ← (PC) must precede MBR ← (memory)
2. 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

Note: If Different Components, then Perform in Same Cycle.

eg: $T_2$: M[MAR] → MBR & PC + Increment → PC.

# Structure of Computer



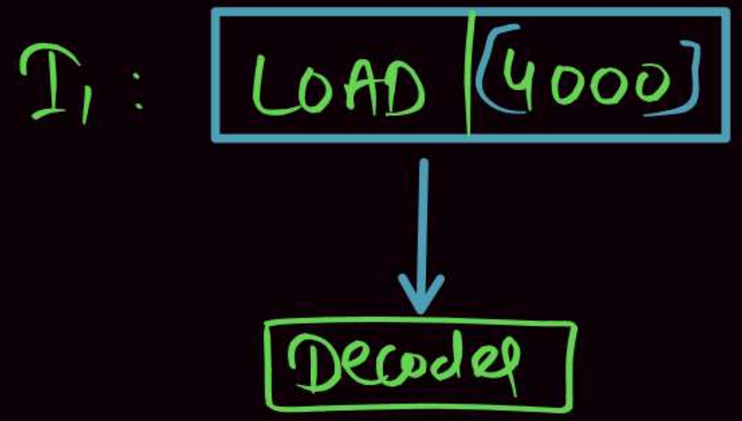
At the end of Fetch cycle Instruction is Loaded in to IR Register.

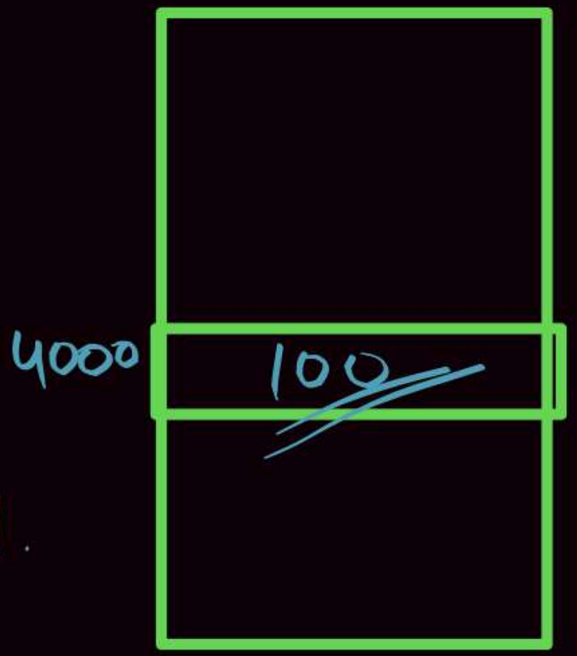

Assume.


---

**Execute cycle:**

1. Decode the Instruction (WHAT opcode, How Many operand, WHERE operand are available etc)
2. Operand Address Calculation
3. Operand Fetch with the Help of Addressing Mode.
4. Data Processing
5. Result Storage.





Memory Read : Direct AM.

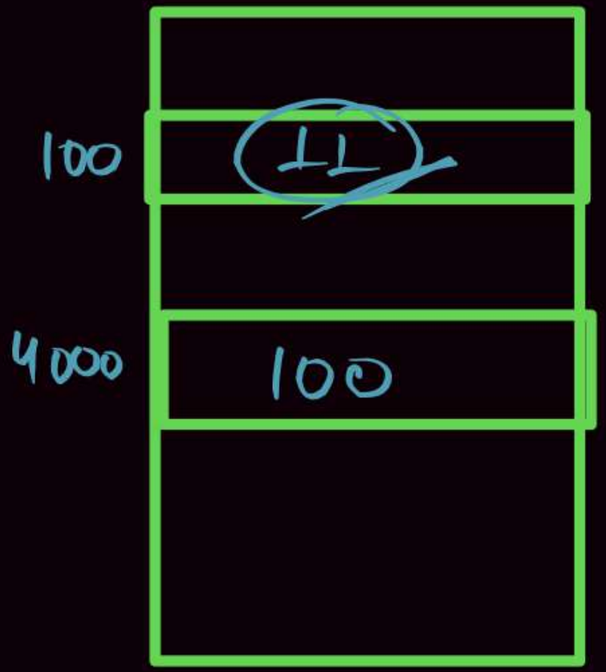
AC ← M[4000]

@$I_1$:

| LOAD | @4000 |
|---|---|

IR

Memory Indirect AM




| ADD/MUL/JMP | Address |
|---|---|

OPCODE Address

# Execute Cycle

- 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 )
  - (i) In the first step the address portion of the IR is loaded into the MAR
  - (ii) Then the referenced memory location is read
  - (iii) Finally the contents of R1 and MBR are added by the ALU
  - (N) 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

LOAD

Memory Read

AC ← M[X]

STORE

Memory write

M[X] ← AC

# Execute Cycle

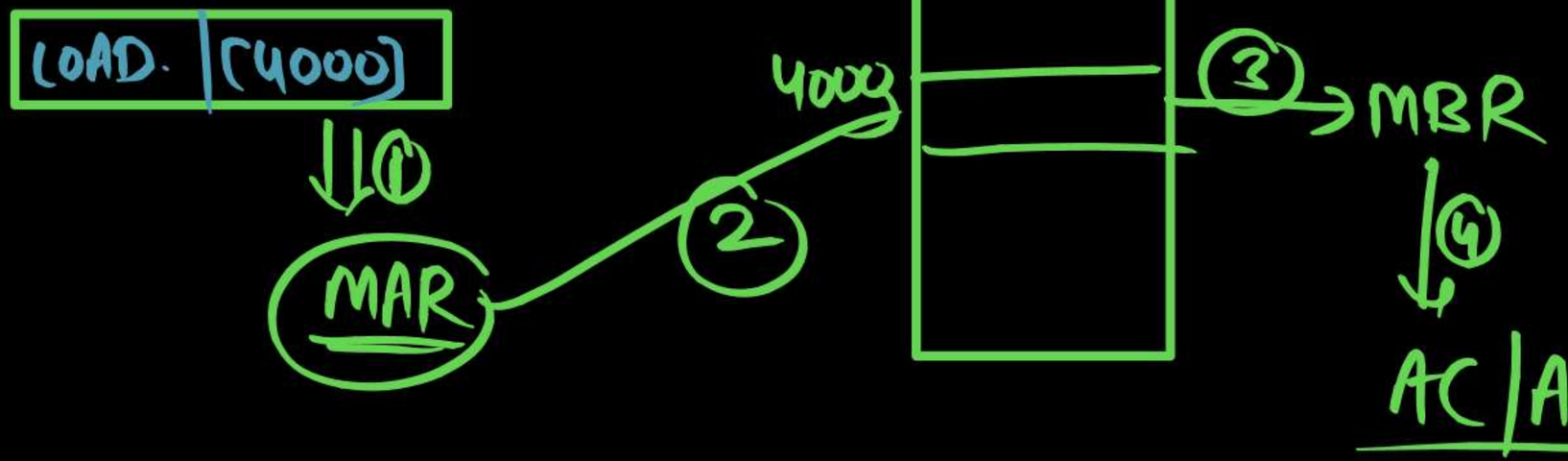
(a) ID Stage: Enable the Hardwire to perform the operation (Instruction Decode)

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

**Example:** Direct AM Memory Read

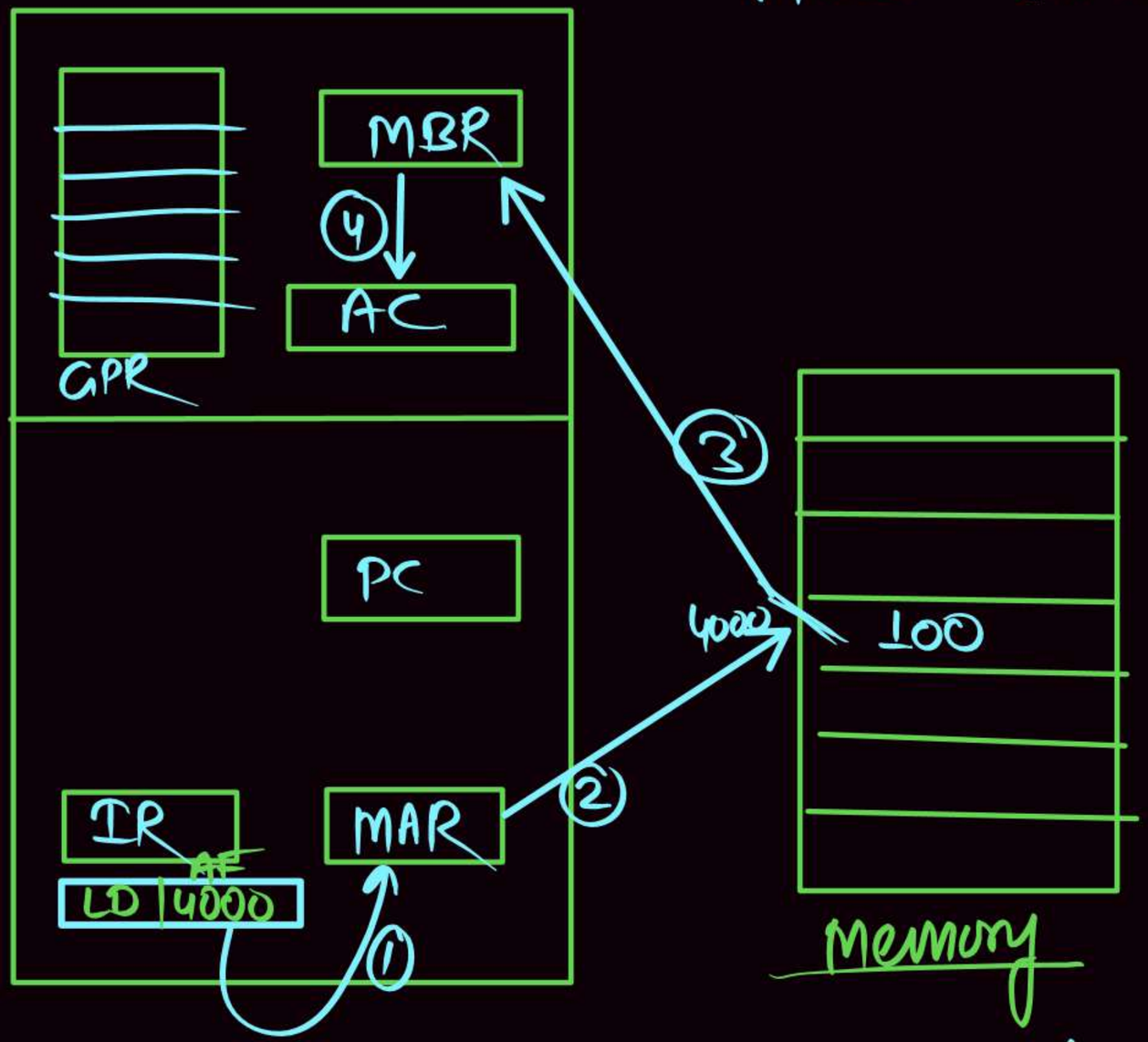
LOAD [4000]

AC ← m[4000]



Execute cycle

LOAD [4000]
AC ← M[4000].



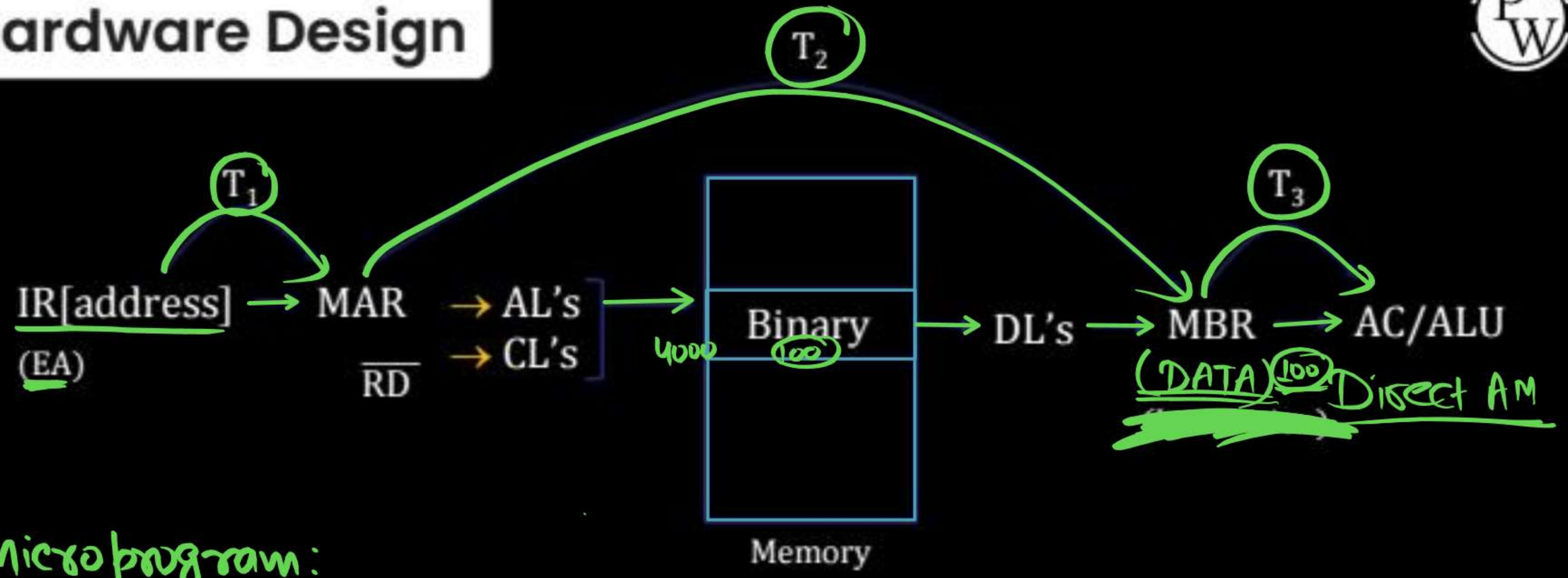

IR(AF) → MAR

M(MAR) → memory

memory → MBR

MBR → AC.

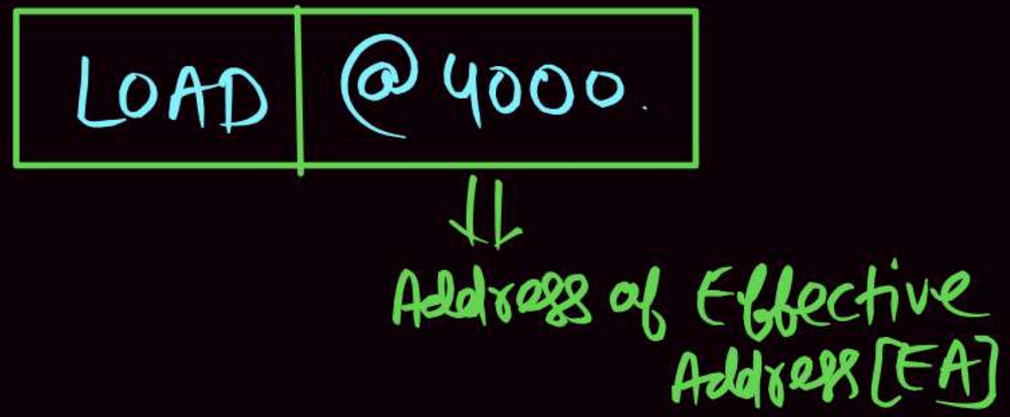
# Hardware Design




Microprogram:

| | | | |
|---|---|---|---|
| T1: | IR(Address field) → MAR: | $IR_{out}$ | $MAR_{in}$ |
| T2: | M[MAR] → MBR: | $MAR_{out}$ | $MBR_{in}$ |
| T3: | MBR → AC/ALU | $MBR_{out}$ | $AC_{in}/ALU_{in}$ |

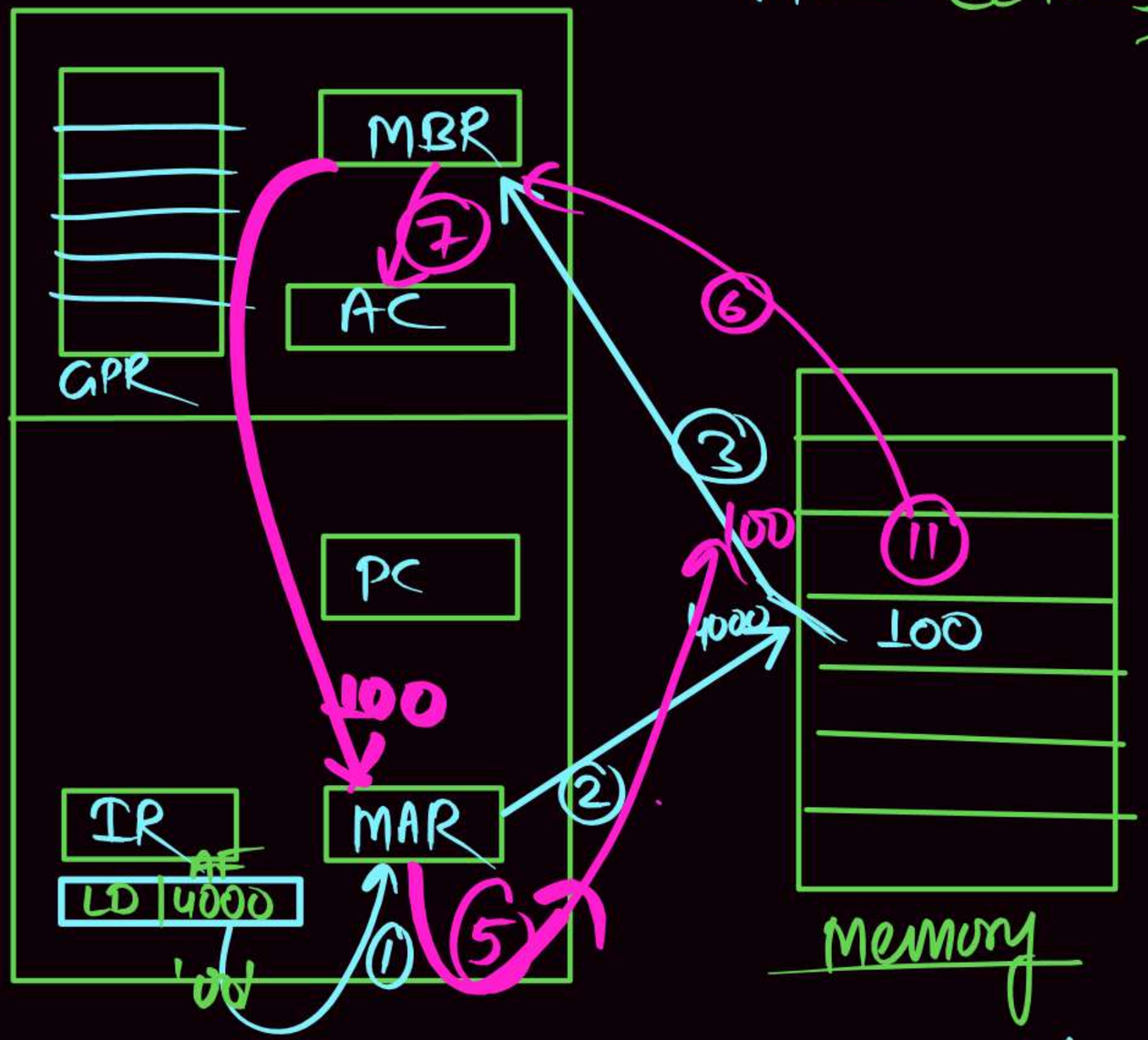
# e.g. Memory Indirect



# Execute cycle

LOAD @4000

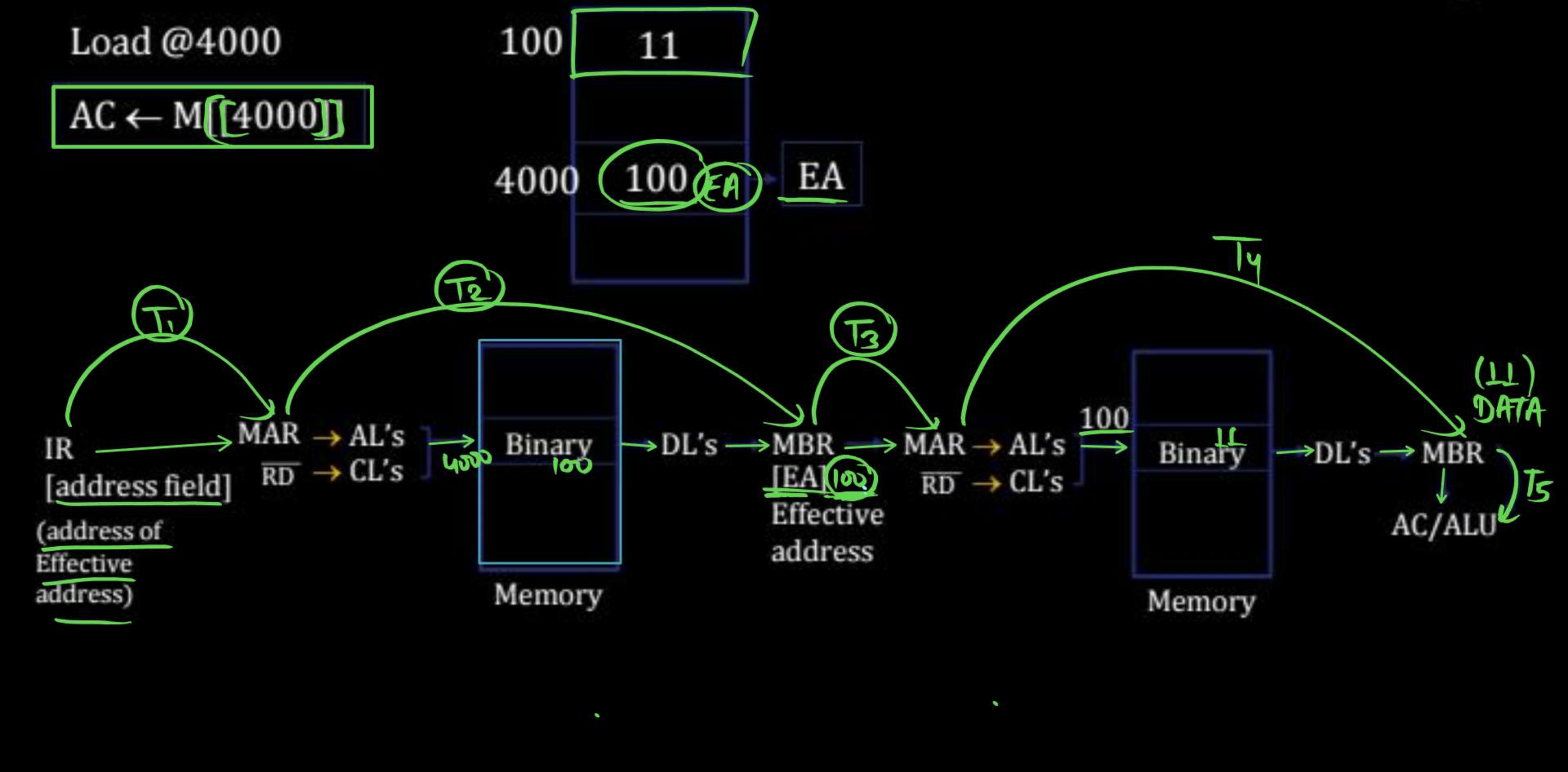
AC ← [[4000]]



# Example: Indirect AM

Load @4000

AC ← M[[4000]]



# Microprogram.

- $T_1$: IR(AF) → MAR ; $IR_{out}$ $MAR_{in}$
- $T_2$: M(MAR) → MBR(EA) ; $MAR_{out}$ $MBR_{in}$
- $T_3$ MBR → MAR ; $MBR_{out}$ $MAR_{in}$
- $T_4$: M[MAR] → MBR ; $MAR_{out}$ $MBR_{in}$
- $T_5$: MBR → AC ; $MBR_{in}$ $AC_{in}$

LOAD @40000.

# Microprogram:

| | | | |
|---|---|---|---|
| $T_1$: | IR[Address] → MAR: | $IR_{out}$ | $MAR_{in}$ |
| $T_2$: | M[MAR] → MBR(EA): | $MAR_{out}$ | $MBR_{in}$ |
| $T_3$: | MBR → MAR: | $MBR_{out}$ | $MAR_{in}$ |
| $T_4$: | M[MAR] → MBR: | $MAR_{out}$ | $MBR_{in}$ |
| $T_5$: | MBR → AC/ALU: | $MBR_{out}$ | $AC_{in}/ALU_{in}$ |

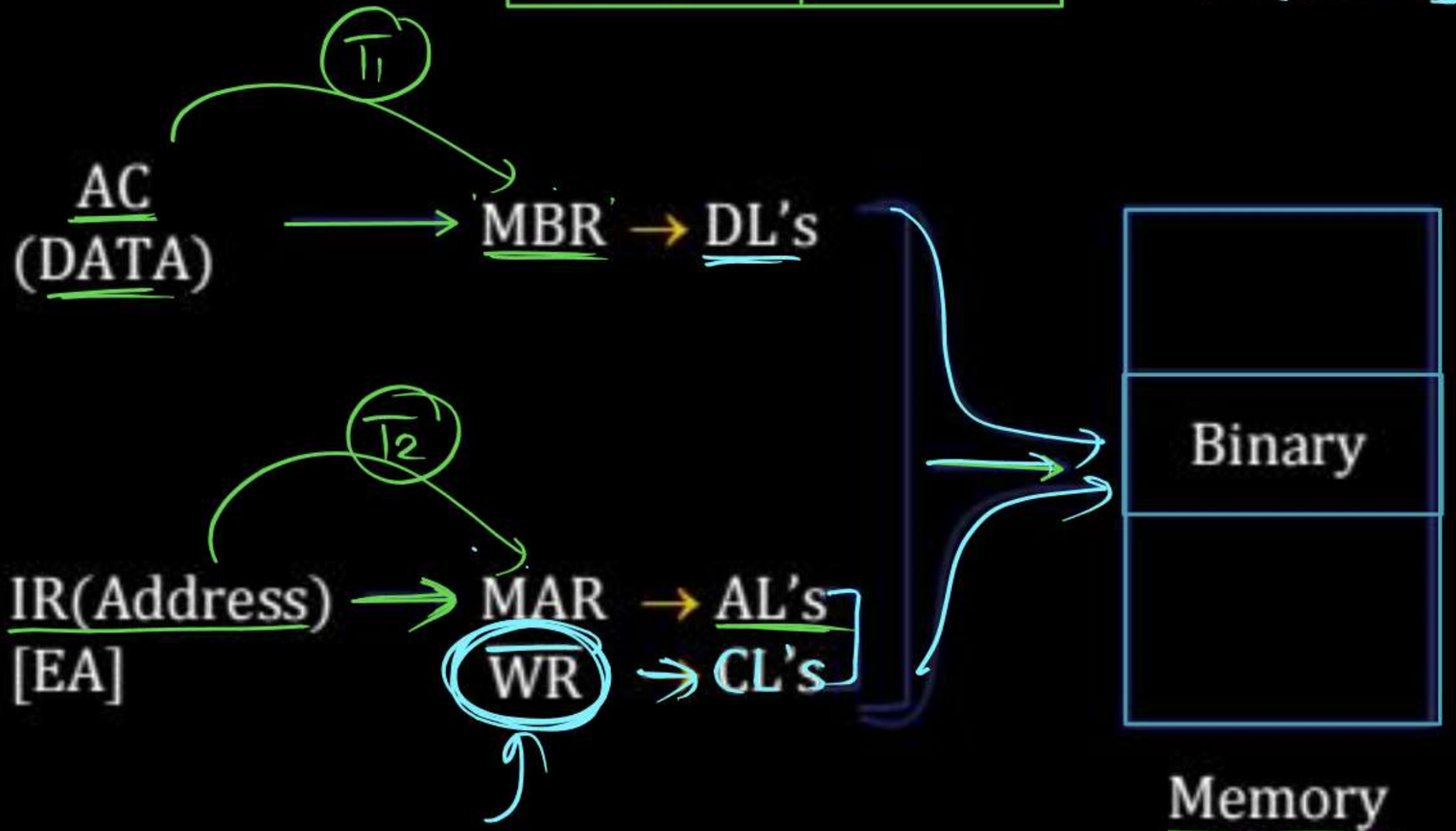
Memory Write

STORE [6000]

M[6000] ← AC

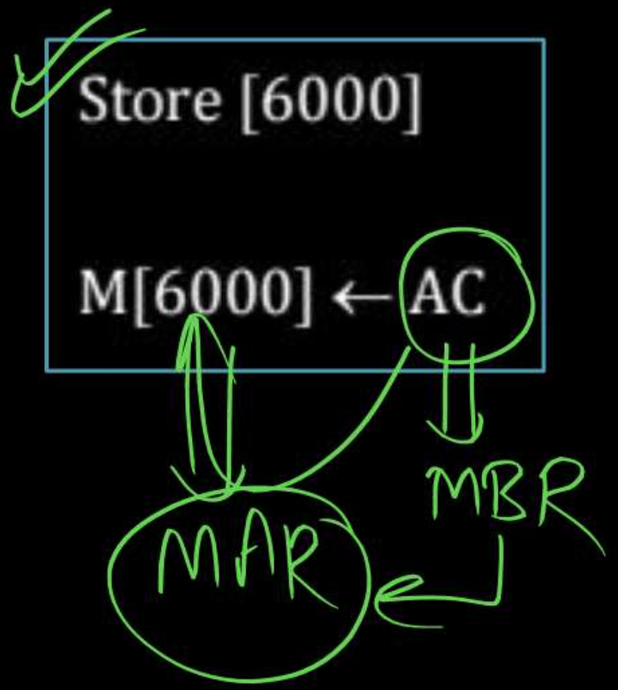
| STORE | 6000. |
|---|---|

M[6000] ← AC.



## Example:

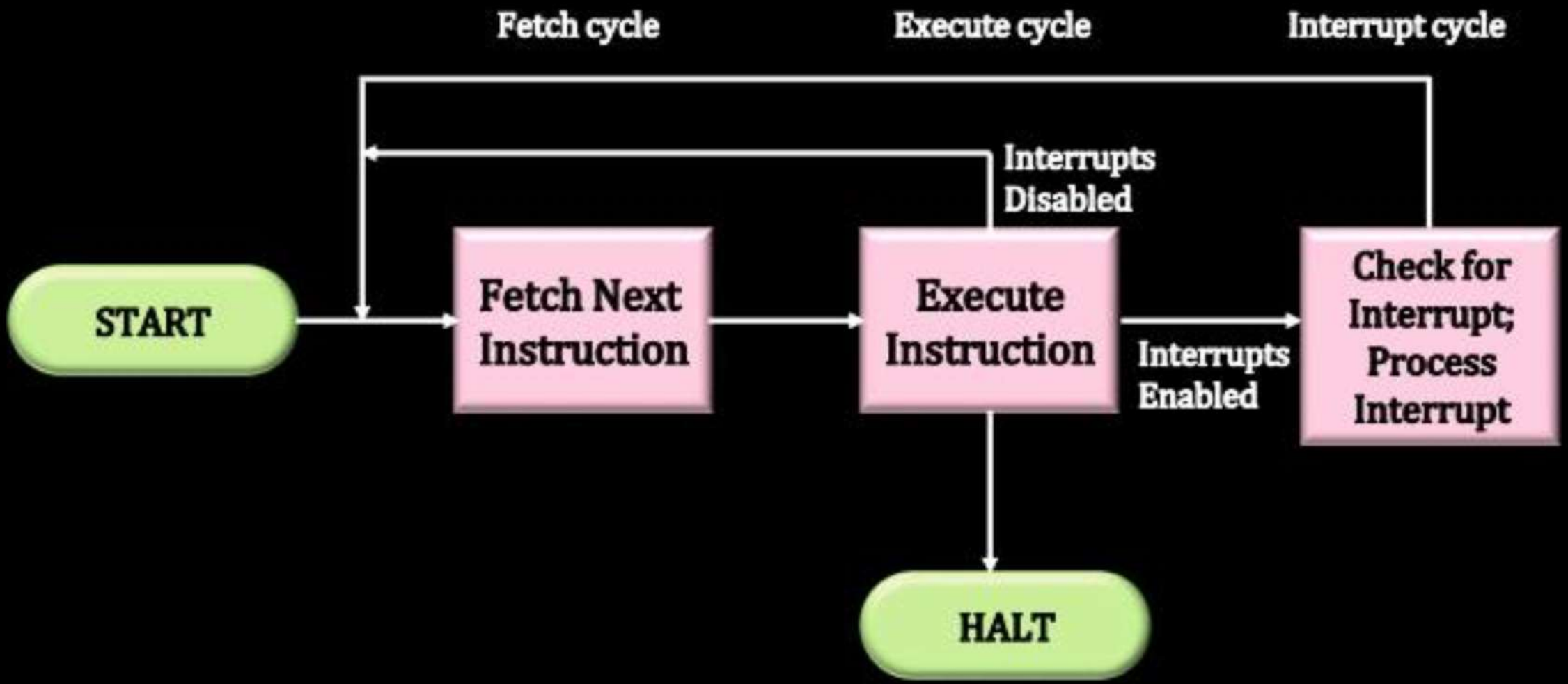
Memory write




Micro Program

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

# Interrupt cycle




Instruction cycle with Interrupts

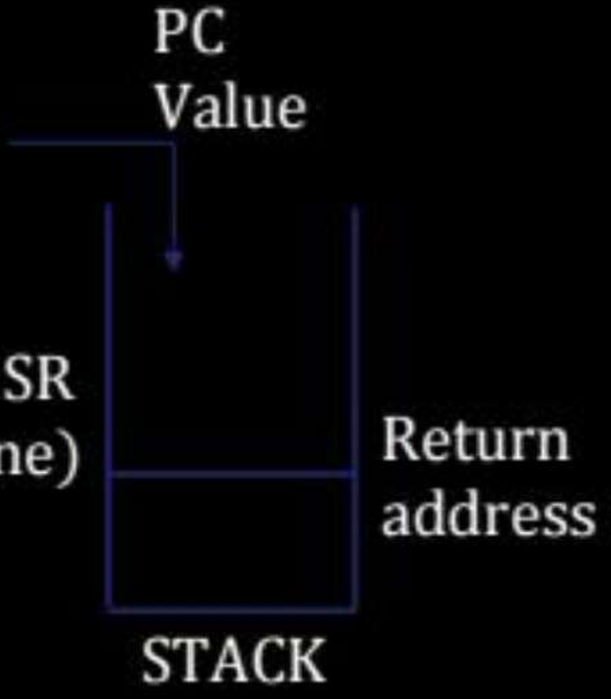
# Interrupt cycle:

- 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:
  - 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
  - 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.

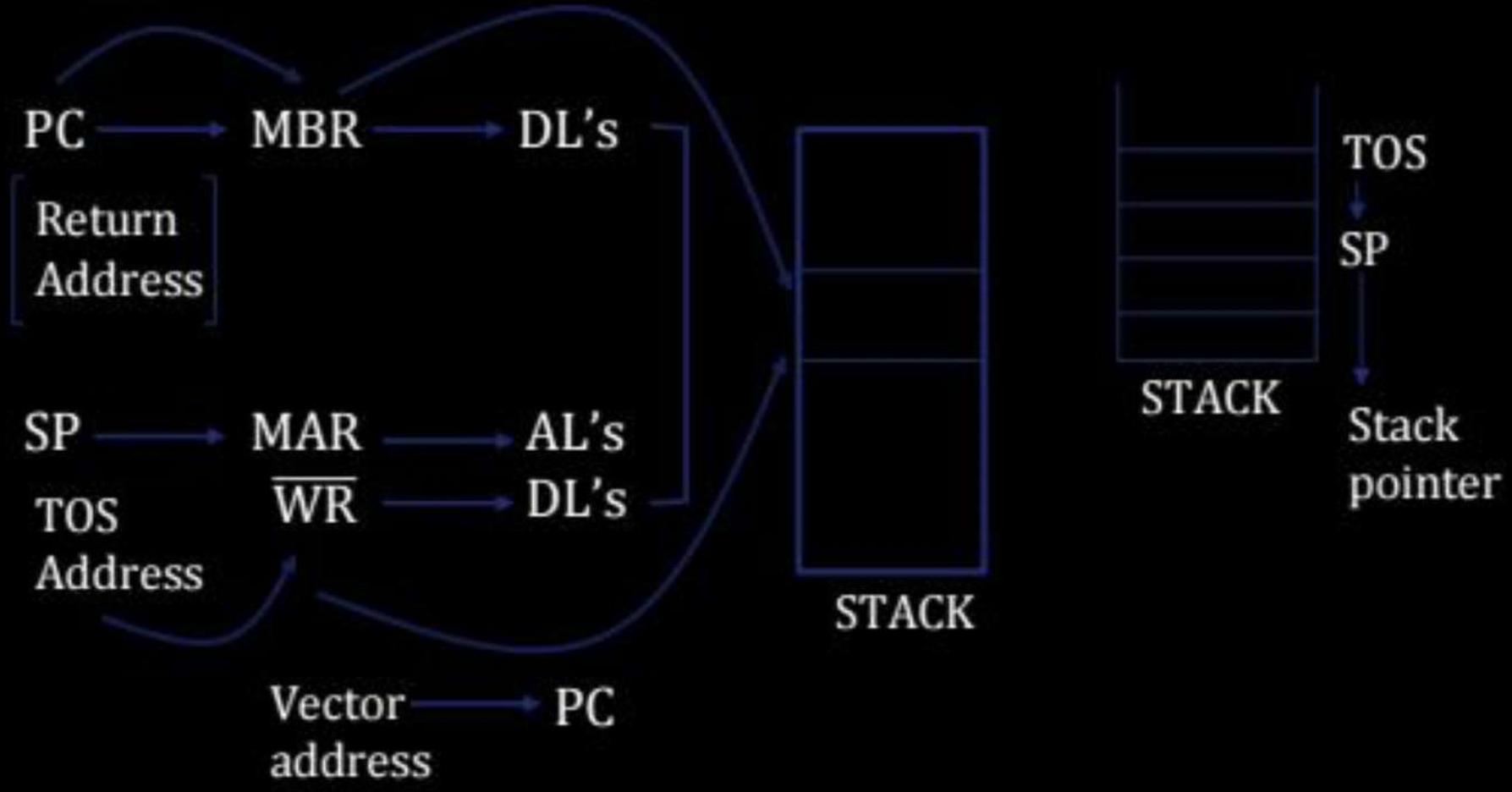
PC value stored in the stack as return address & Control transfer to ISR (Interrupt subroutine)




Vector address → PC

1004 $I_4$: JMP 2000
1005 $I_5$: 2000

# Hardware Design



Microprogram
μ Program

- T1: PC → MBR
- T2: SP → MAR
- T3: MBR → M[MAR]: System Bus

  Vector Address → PC: Local Bus

# THANK YOU!