CHAPTER 3

BASIC COMPUTER ORGANISATION AND DESIGN

Outlines

- Introduction: History of computer architecture
- Overview of computer organization
- Memory hierarchy and Cache
- Instruction Codes: Stored Program Organization, Indirect address, Computer Registers, Common Bus systems, Instruction set, Timing and Control, Instruction Cycle

INTRODUCTION: Description of Basic Computer

- The operation of Basic computer can be specified by the register transfer statements. Internal organization of the computer is defined by the sequence of microoperations it performs on data stored in its registers. Every different processor type has its own design (different registers, buses, microoperations, machine instructions, etc).. It contains:

– Many registers

– Multiple arithmetic units, for both integer and floating point calculations

– The ability to pipeline several consecutive instructions for execution speedup

However, to understand how processors work, we will start with a simplified processor model. M. Morris Mano introduces a simple processor model; he calls it a —Basic Computer. The Basic Computer has two components, a processor and memory.

The memory has 4096 words in it

–4096 = 212, so it takes 12 bits to select an address in memory

Each word is 16 bits long

Instruction Code and Stored Program Organisation

- An **instruction code** is a group of bits that instructs the computer to perform a specific operation.
- It is usually divided into parts each having one particular interpretation. Most basic part is operation (**operation code**). Operation code is group of bits that defines operations as add, subtract, multiply, shift, complement etc. The operation part of an instruction code specifies the operation to be performed.
- This operation must be performed on some data stored in processor register or in memory.
- An instruction code therefore specify not only the operation but also the register or the memory word where the operand (data on which operation is performed) are to be found.

Stored Program Organization:

- The program (instruction) as well as data (operand) is stored in the same memory. If the instruction needs data, the data is found in the same memory and accessed. This feature is called stored program organization.

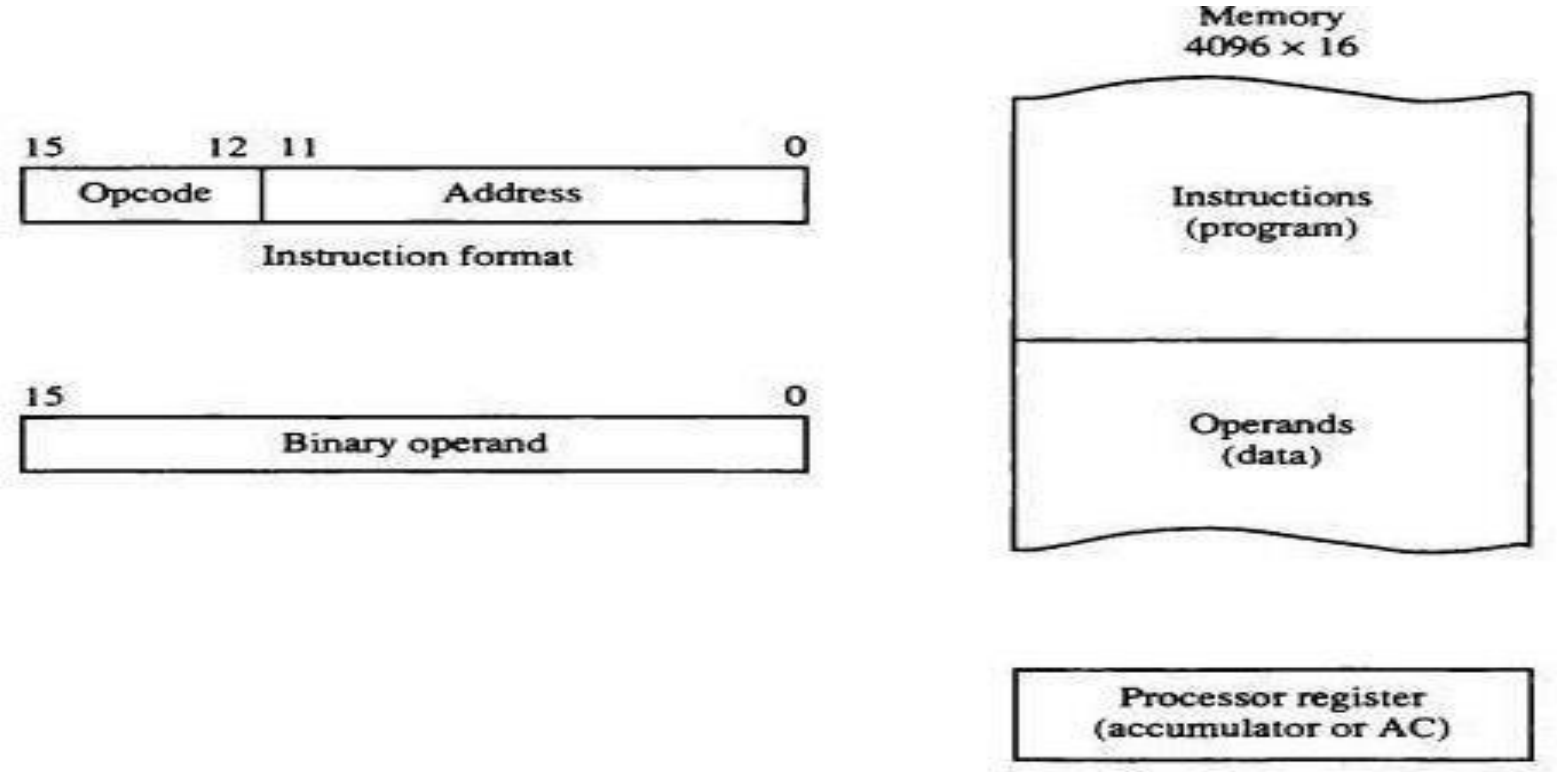


Figure: Stored Program Organisation

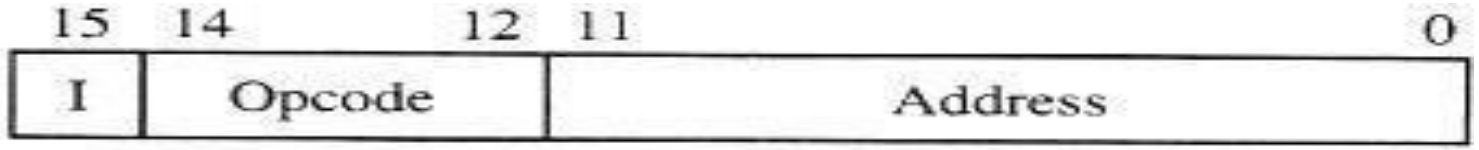
Instruction Formats

Instruction Format of Basic Computer:

A computer instruction is often divided into two parts

- An *opcode* (Operation Code) that specifies the operation for that instruction
- An *address* that specifies the registers and/or locations in memory to use for that operation.

In the Basic Computer, since the memory contains 4096 (= 2^{12}) words, we needs 12 bit to specify the memory address that is used by this instruction. In the Basic Computer, bit 15 of the instruction specifies the *addressing mode* (0: direct addressing, 1: indirect addressing). Since the memory words, and hence the instructions, are 16 bits long, that leaves 3 bits for the instruction's opcode.



(a) Instruction format

Addressing Modes:

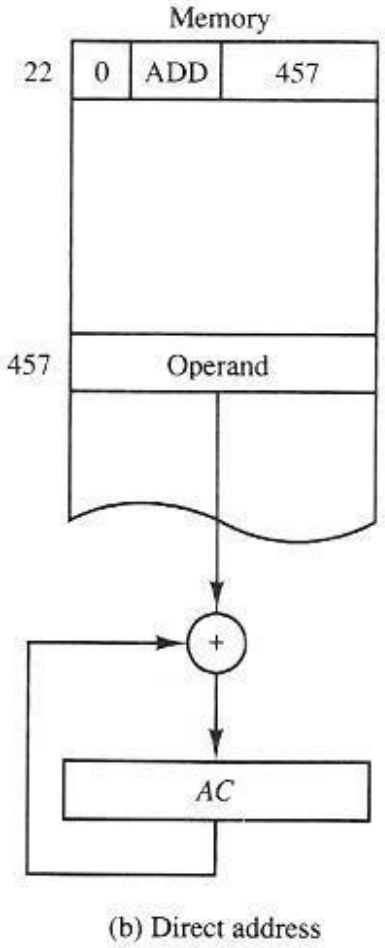
Addressing Modes:

The address field of an instruction can be represented in two ways

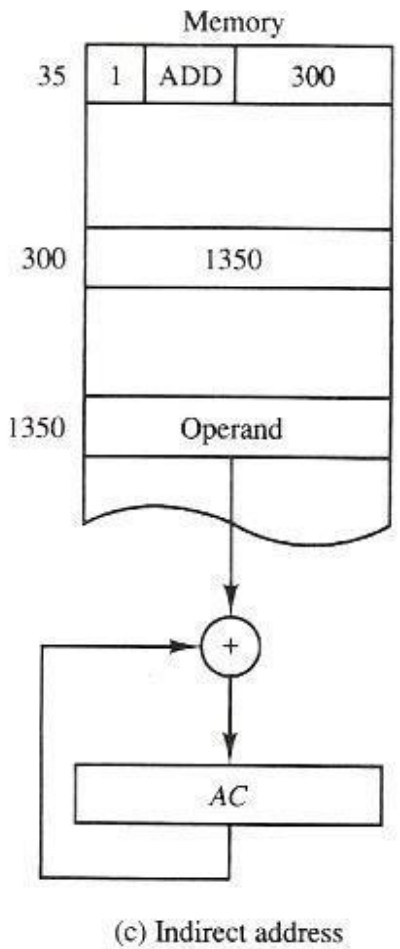
– **Direct address**: the address operand field is effective address (the address of the operand)

– **Indirect address:** the address in operand field contains the memory address where effective address resides.

Note: **Effective Address (EA):** The address, where actual data resides is called effective address.



(b) Direct address



(c) Indirect address

- A direct address instruction is shown in figure b. It is placed in address 22 in memory. The I bit is 0, so the instruction is recognized as a direct address instruction. The opcode specifies an ADD instruction, and the address part is the binary equivalent of 457. The control finds the operand in the memory at address 457 and adds it to the content of AC.
- The instruction in address 35 shown in figure c has a mode bit I=1. Therefore, it is recognized as indirect address instruction. The address part is the binary equivalent of 300. The control goes to address 300 to find the address of an operand. The address of an operand in this case is 1350. The operand found in address 1350 is then added to the content of AC. The indirect address instruction needs two references to memory to fetch an operand. The first reference is needed to read the address of the operand and second is for the operand itself. The effective address of direct addressing mode is 457 and that of indirect addressing mode is 1350.

Computer Registers

- Computer instructions are normally stored in the consecutive memory locations and are executed sequentially one at a time. T
- Thus computer needs processor registers for manipulating data and holding memory address which are shown in the following table:

Symbol	Size	Register Name	Description
DR	16	Data Register	Holds memory operand
AR	12	Address Register	Holds address for memory
AC	16	Accumulator	Processor register
IR	16	Instruction Register	Holds instruction code
PC	12	Program Counter	Holds address of instruction
TR	16	Temporary Register	Holds temporary data
INPR	8	Input Register	Holds input character
OUTR	8	Output Register	Holds output character

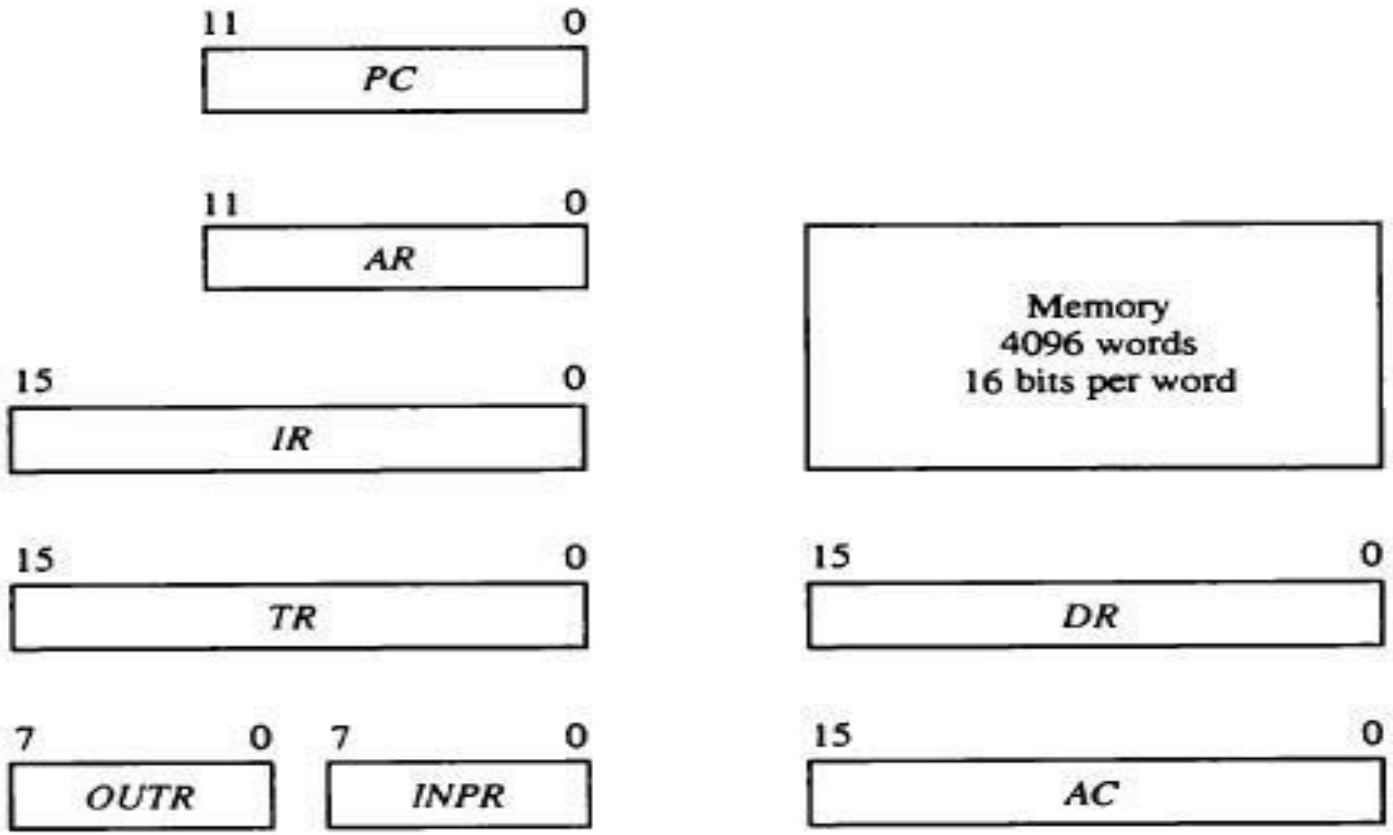


Figure 5-3 Basic computer registers and memory.

- Since the memory in the Basic Computer only has 4096 (=2^{12}) locations, PC and AR only needs 12 bits. Since the word size of Basic Computer only has 16 bit, the DR, AC, IR and TR needs 16 bits. The Basic Computer uses a very simple model of input/output (I/O) operations.
- Input devices are considered to send 8 bits of character data to the processor
- The processor can send 8 bits of character data to output devices

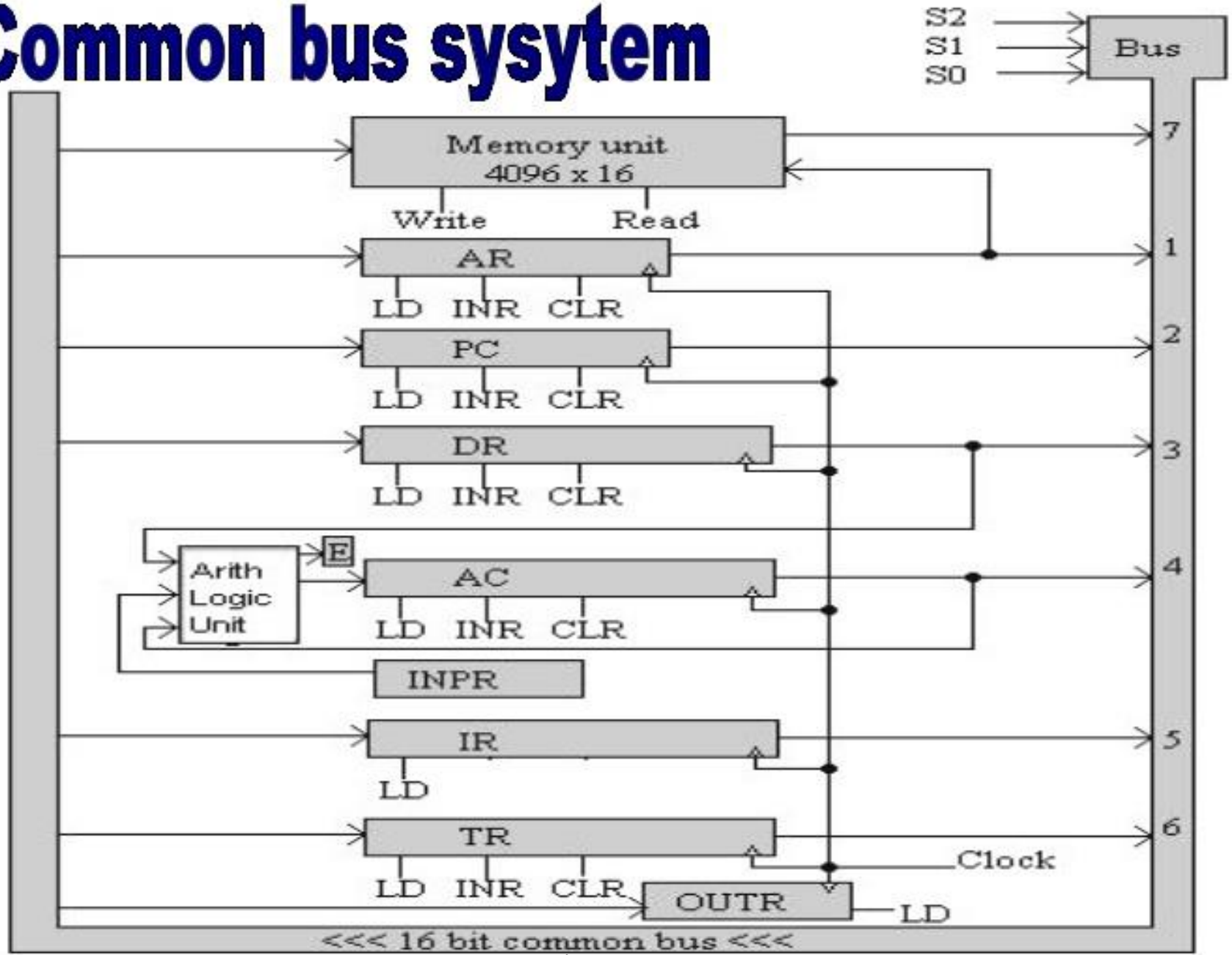
The Input Register (INPR) holds an 8-bit character gotten from an input device and the Output Register (OUTR) holds an 8-bit character to be sent to an output device.

Common Bus System

- The basic computer has eight registers, a memory unit, and a control unit. These registers, memory and control unit are connected using a path (bus) so that information can be transferred to each other. If separate buses are used for connecting each registers, it will cost high. The cost and use of extra buses can be reduced using a special scheme in which many registers use a common bus, called *common bus system*
- The registers in the Basic Computer are connected using a bus. This gives a savings in circuitry over complete connections between registers. Three control lines, S2, S1, and S0 control which register the bus selects as its input.

S_2	S_1	S_0	Register
0	0	0	X (nothing)
0	0	1	AR
0	1	0	PC
0	1	1	DR
1	0	0	AC
1	0	1	IR
1	1	0	TR
1	1	1	Memory

Common bus sysytem



- The particular register whose LD (Load) input is enabled receives the data from the bus during the next clock pulse transition. The memory receives the contents of the bus when its write input is activated.
- The memory places its 16 bit output onto the bus when the read input is activated and the value of S2, S1, and S0 is 111.
- Four registers DR, AC, IR and TR have 16 bits each and two registers AR and PC have 12 bits each since they hold a memory address.
- When the content of AR and PC are applied to the 16-bit common bus, the four MSBs are set to 0's.
- When AR and PC receive information from the bus, only the 12 least significant bits are transferred into the register.
- The input register INPR and the output register OUTR have 8 bits each and communicate with the eight least significant bits in the bus.

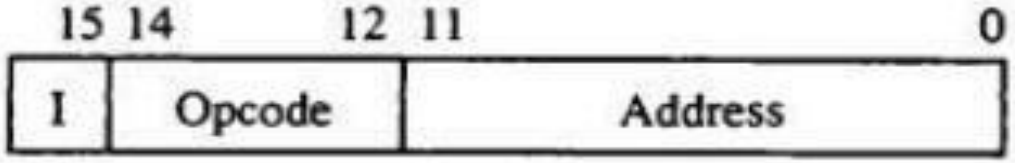
- INPR is connected to provide information to the bus but OUTR can only receive information from the bus.
- Five registers have three control inputs: LD (load), INR (increment) and CLR(Clear).
- Two registers have only a LD input.
- AR must always be used to specify memory address; therefore memory address is connected to AR. The 16 inputs of AC come from an Adder and Logic Circuit. This circuit has three inputs.
- Set of 16- bit inputs come from the outputs of AC.
- Set of 16-bit inputs come from the data register DR.
- Set of 8-bit inputs come from the input register INPR.
- The result of an addition is transferred to AC and the end carry out of the addition is transferred to flip flop E (extended AC-bit).
- The clock transition at the end of the cycle transfers the content of the bus into the designated destination register and the output of the adder and logic circuit into AC.

Computer Instructions

The basic computer has 3 instruction code formats. This type of the instruction is recognized by the computer control from 4-bit positions 12 through 15 of the instruction.

- **Memory-Reference Instructions**
- **Register-Reference Instructions**
- **Input-Output Instructions**

1. Memory-Reference Instructions:

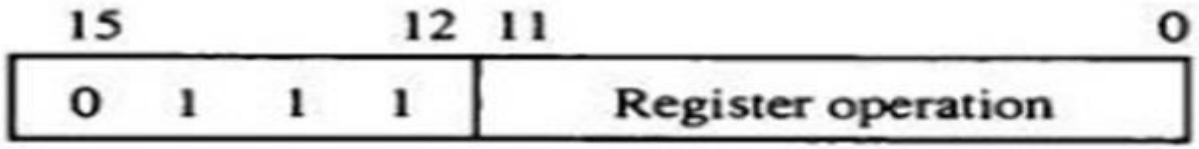


(Opcode = 000 through 110)

(a) Memory – reference instruction

Symbol	Hex Code I = 0	Hex Code I = 1	Description
AND	0xxx	8xxx	AND memory word to AC
ADD	1xxx	9xxx	Add memory word to AC
LDA	2xxx	Axxx	Load AC from memory
STA	3xxx	Bxxx	Store content of AC into memory
BUN	4xxx	Cxxx	Branch unconditionally
BSA	5xxx	Dxxx	Branch and save return address
ISZ	6xxx	Exxx	Increment and skip if zero

2. Register-Reference Instructions

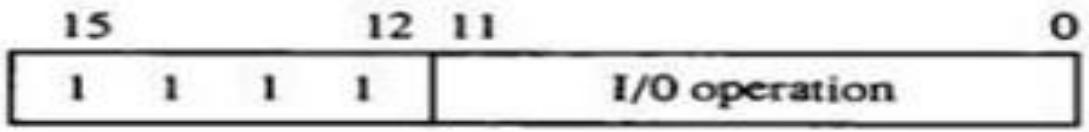


(Opcode = 111, $I = 0$)

(b) Register – reference instruction

CLA	7800	Clear AC
CLE	7400	Clear E
CMA	7200	Complement AC
CME	7100	Complement E
CIR	7080	Circulate right AC and E
CIL	7040	Circulate left AC and E
INC	7020	Increment AC
SPA	7010	Skip next instr. if AC is positive
SNA	7008	Skip next instr. if AC is negative
SZA	7004	Skip next instr. if AC is zero
SZE	7002	Skip next instr. if E is zero
HLT	7001	Halt computer

3. Input-Output Instructions



(Opcode = 111, $I = 1$)

(c) Input – output instruction

INP	F800	Input character to AC
OUT	F400	Output character from AC
SKI	F200	Skip on input flag
SKO	F100	Skip on output flag
ION	F080	Interrupt on
IOF	F040	Interrupt off

Instruction Set Completeness

An instruction set is said to be complete if it contains sufficient instructions to perform operations in following categories:

Functional Instructions

- Arithmetic, logic, and shift instructions
- Examples: ADD, CMA, INC, CIR, CIL, AND, CLA

Transfer Instructions

- Data transfers between the main memory and the processor registers
- Examples: LDA, STA

Control Instructions

- Program sequencing and control
- Examples: BUN, BSA, ISZ

Input/output Instructions

- Input and output
- Examples: INP, OUT

Instruction set of Basic computer is complete because:

- ADD, CMA (complement), INC can be used to perform addition and subtraction and CIR (circular right shift), CIL (circular left shift) instructions can be used to achieve any kind of shift operations. Addition subtraction and shifting can be used together to achieve multiplication and division. AND, CMA and CLA (clear accumulator) can be used to achieve any logical operations.
- LDA instruction moves data from memory to register and STA instruction moves data from register to memory.
- The branch instructions BUN, BSA and ISZ together with skip instruction provide the mechanism of program control and sequencing.
- INP instruction is used to read data from input device and OUT instruction is used to send data from processor to output device.

Timing and Control Unit

Control Unit

Control unit (CU) of a processor translates from machine instructions to the control signals for the microoperations that implement them. There are two types of control organization:

a) Hardwired Control

b) Microprogrammed Control

***a) Hardwired* Control**

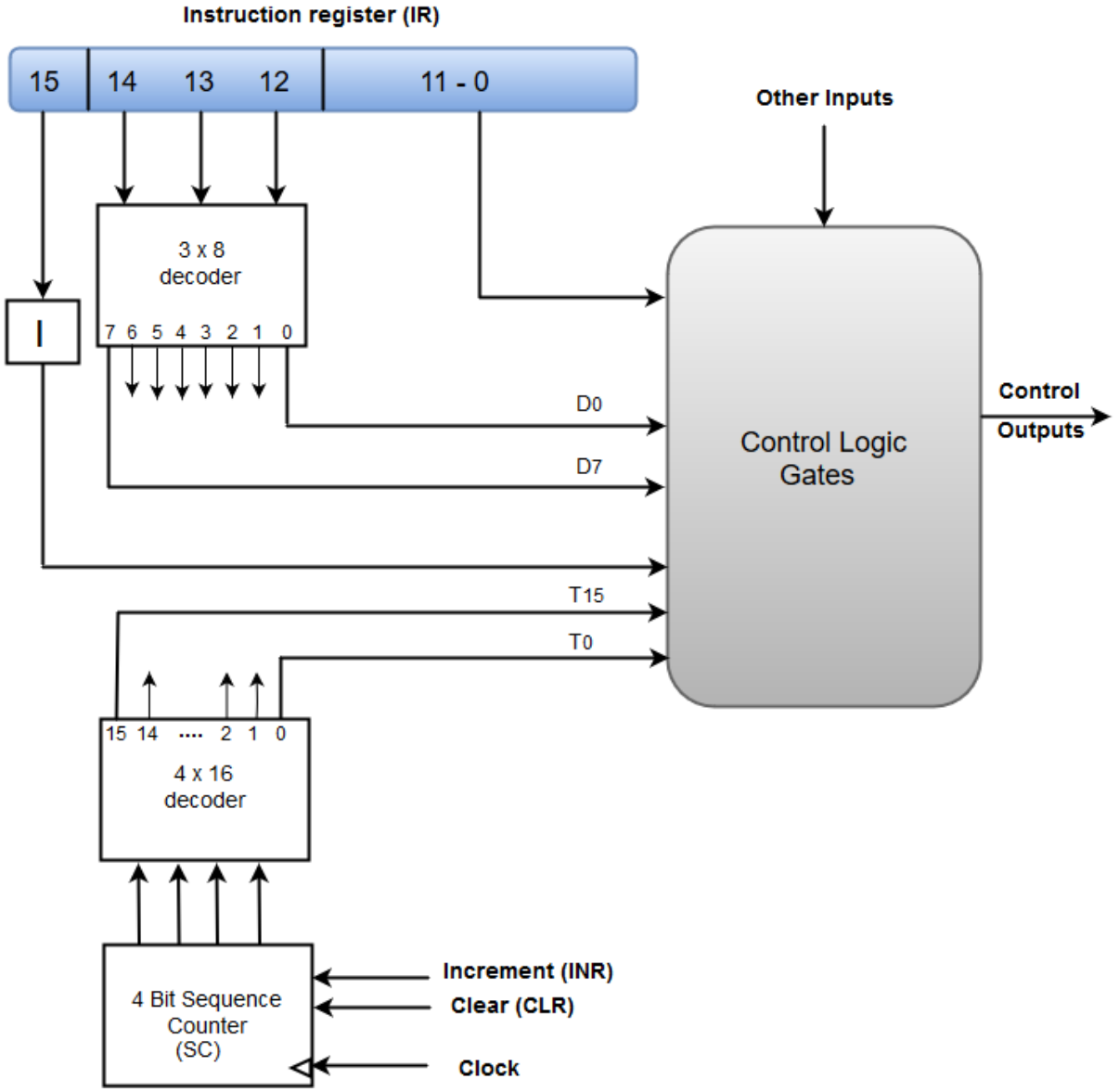
- CU is made up of sequential and combinational circuits to generate the control signals.
- If logic is changed, we need to change the whole circuit.
- Expensive
- Fast

b) Microprogrammed Control

- A control memory on the processor contains microprograms that activate the necessary control signals.
- If logic is changed, we only need to change the microprogram.
- Cheap
- Slow

Control Unit of a Basic Computer (Hardwired Control)

Control Unit of a Basic Computer:

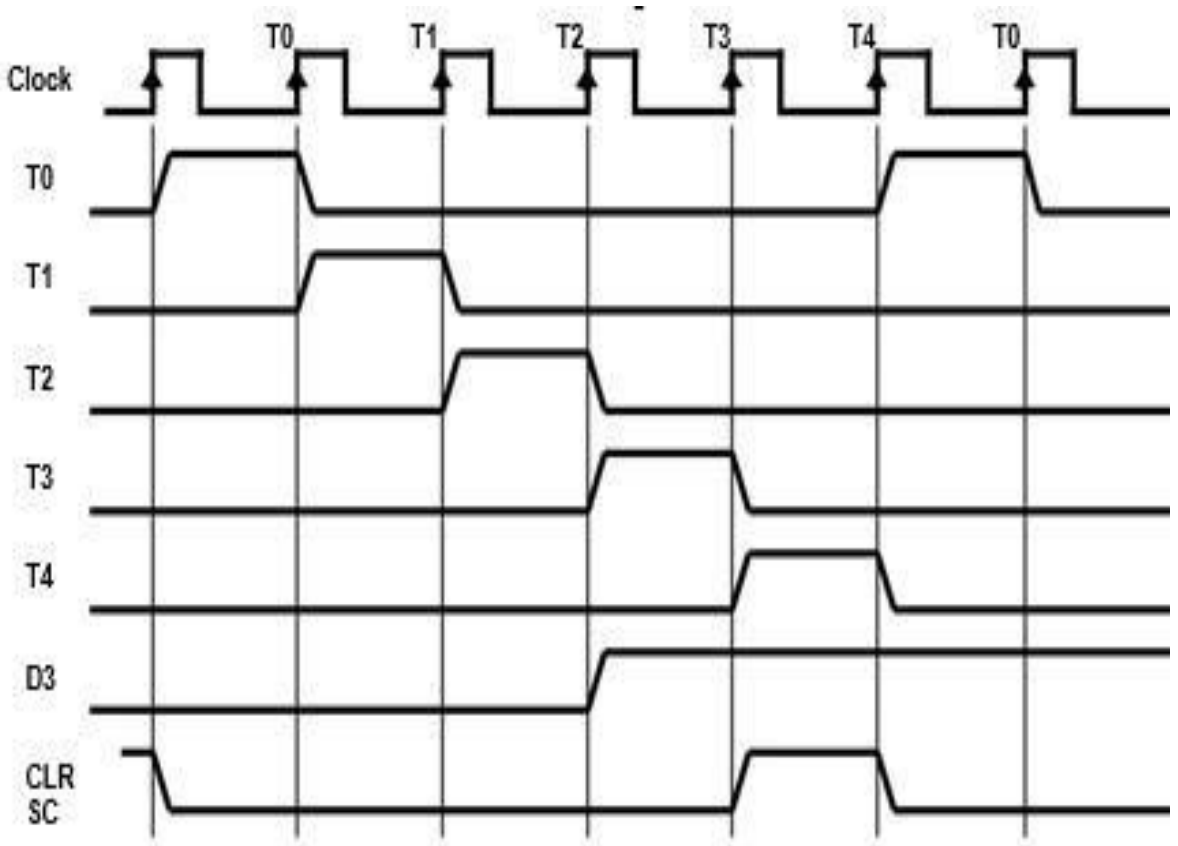


Description

- A Hard-wired Control consists of two decoders, a sequence counter, and a number of logic gates.
- An instruction fetched from the memory unit is placed in the instruction register (IR).
- The component of an instruction register includes; I bit, the operation code, and bits 0 through 11.
- The operation code in bits 12 through 14 are coded with a 3 x 8 decoder.
- The outputs of the decoder are designated by the symbols D0 through D7.
- The operation code at bit 15 is transferred to a flip-flop designated by the symbol I.
- The operation codes from Bits 0 through 11 are applied to the control logic gates.
- The Sequence counter (SC) can count in binary from 0 through 15.

Timing Signal

- Generated by 4-bit sequence counter and 4x16 decoder.
- The SC can be incremented or cleared.
- Example: T_0, T_1, T_2, T_3, T_4, T_0, T_1 . . .
- Assume: At time T4, SC is cleared to 0 if decoder output D3 is active:
- D_3T_4: SC← 0



Instruction Cycle

Processing required for complete execution of an instruction is called instruction cycle. In Basic Computer, a machine instruction is executed in the following cycle:

- Fetch an instruction from memory
- Decode the instruction
- Read the effective address from memory if the instruction has an indirect address
- Execute the instruction
- Upon the completion of step 4, control goes back to step 1 to fetch, decode and execute the next instruction. This process is continued indefinitely until HALT instruction is encountered.

Fetch and Decode

- Sequence of steps required for fetching instruction from memory to CPU internal register is known as fetch cycle.
- The microoperations for the fetch and decode phases can be specified by the following register transfer statements: (first two steps for fetch and third step for decode)

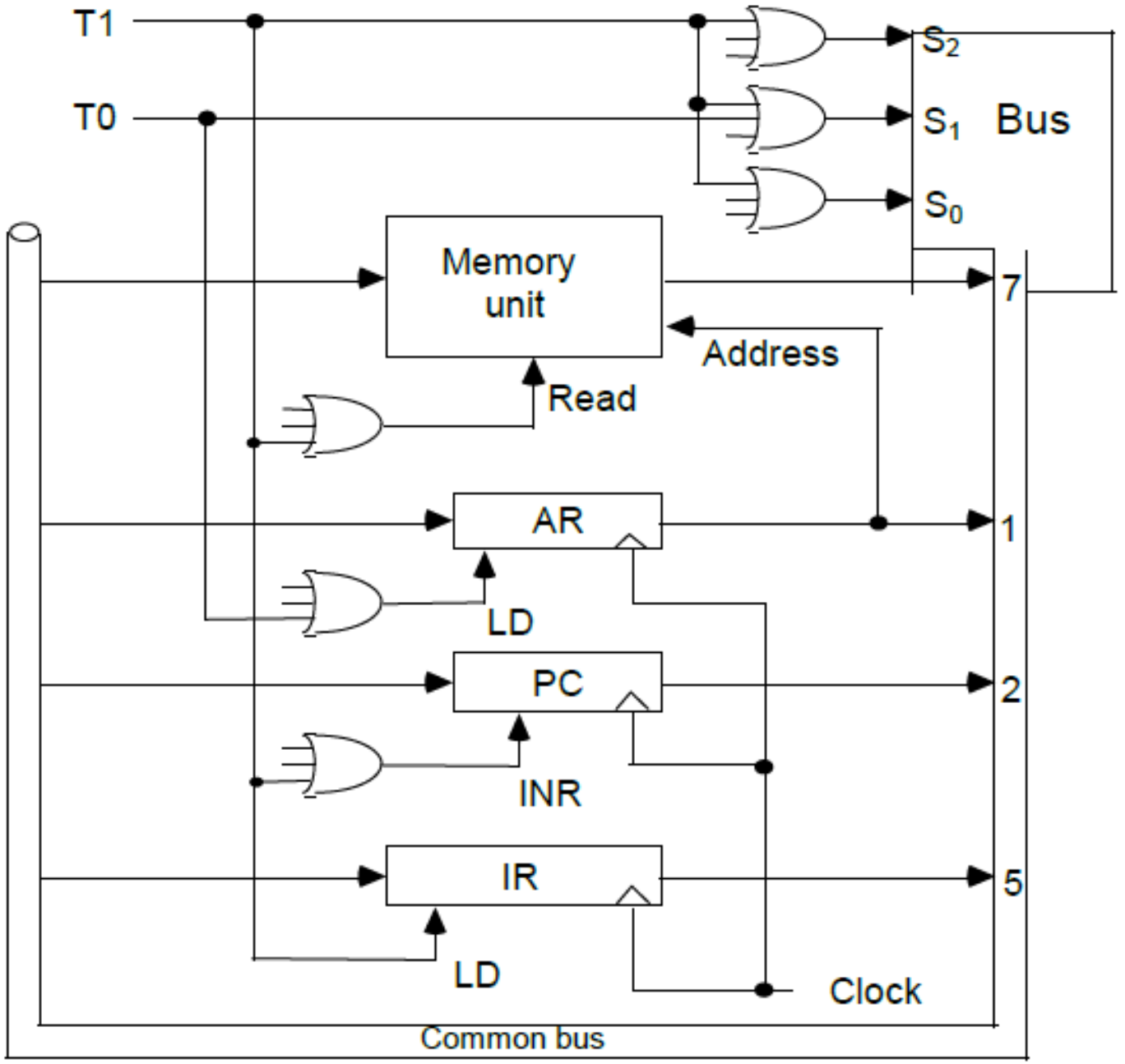
Fetch and Decode

T0: AR ←PC (S0S1S2=010, T0=1)

T1: IR ← M [AR], PC ← PC + 1 (S0S1S2=111, T1=1)

T2: D0, . . . , D7 ← Decode IR(12-14), AR ← IR(0-11), I ← IR(15)

Fetch phase of basic computer



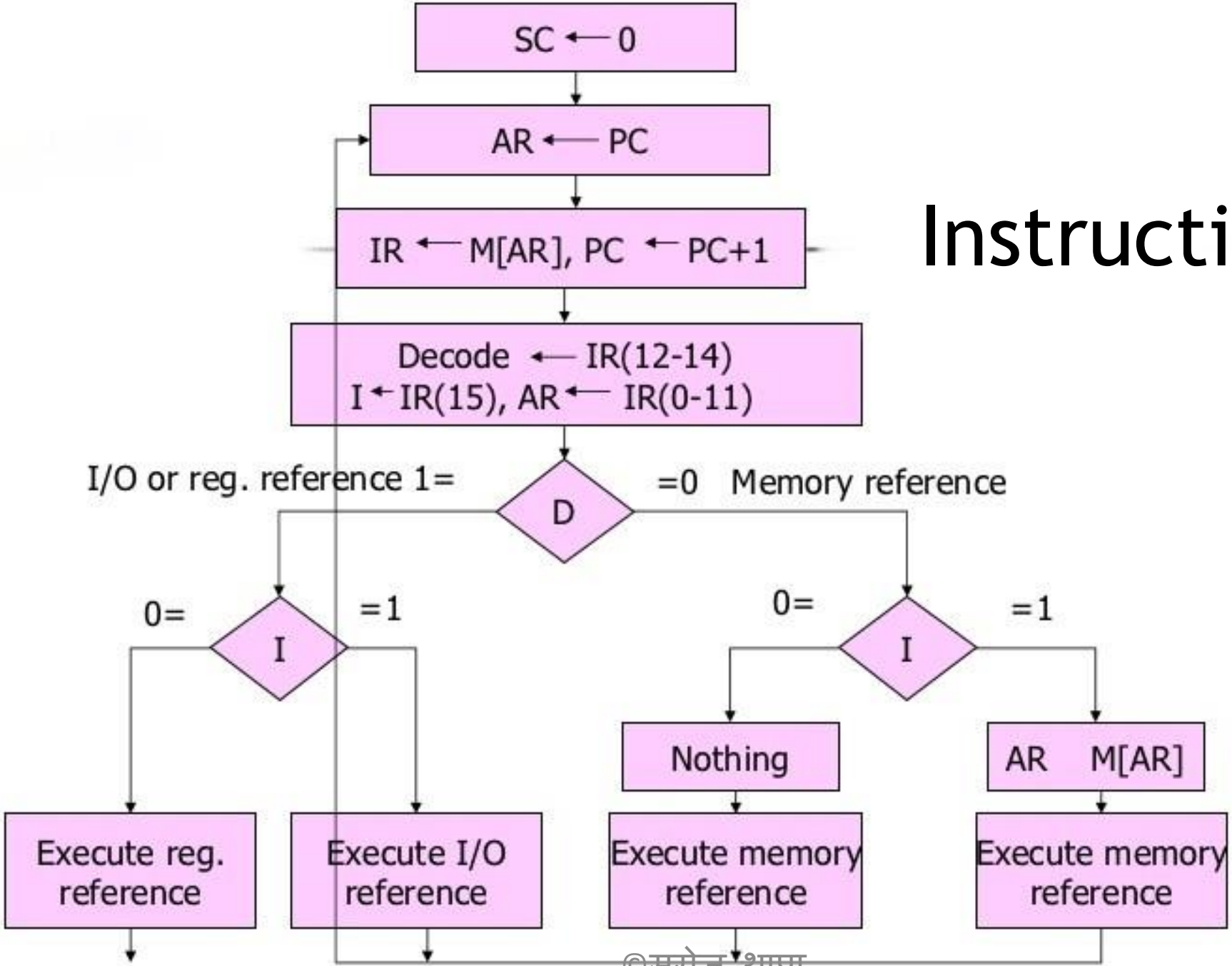
Description

- It is necessary to transfer the address from PC to AR during clock transition associated with the timing signal T0.
- The instruction read from memory is then placed in IR with clock transition associated with the timing signal T1.
- At the same time, PC is incremented by one to prepare for the next instruction in the program.
- At time T2, the opcode in IR is decoded, the indirect bit is transferred to flip-flop I, and the address part of the instruction is transferred to AR.

Determine the type of the instruction

- The timing signal that is active after decoding is T_3. During time T_3, the control unit determines the type of instruction that was just read from memory.
- Following flowchart presents an initial configuration for the **instruction cycle** and shows how the control determines the instruction type after decoding.

Instruction cycle



Then, among decoded, D7 determines which type of instruction.

1) If D7 = 1, it will be either register-reference or input-output instruction.

- If I = 1, input-output instruction is executed during T3.
- If I = 0, register-reference instruction is executed during T3.

2) If D7 = 0, it will be memory-reference instruction.

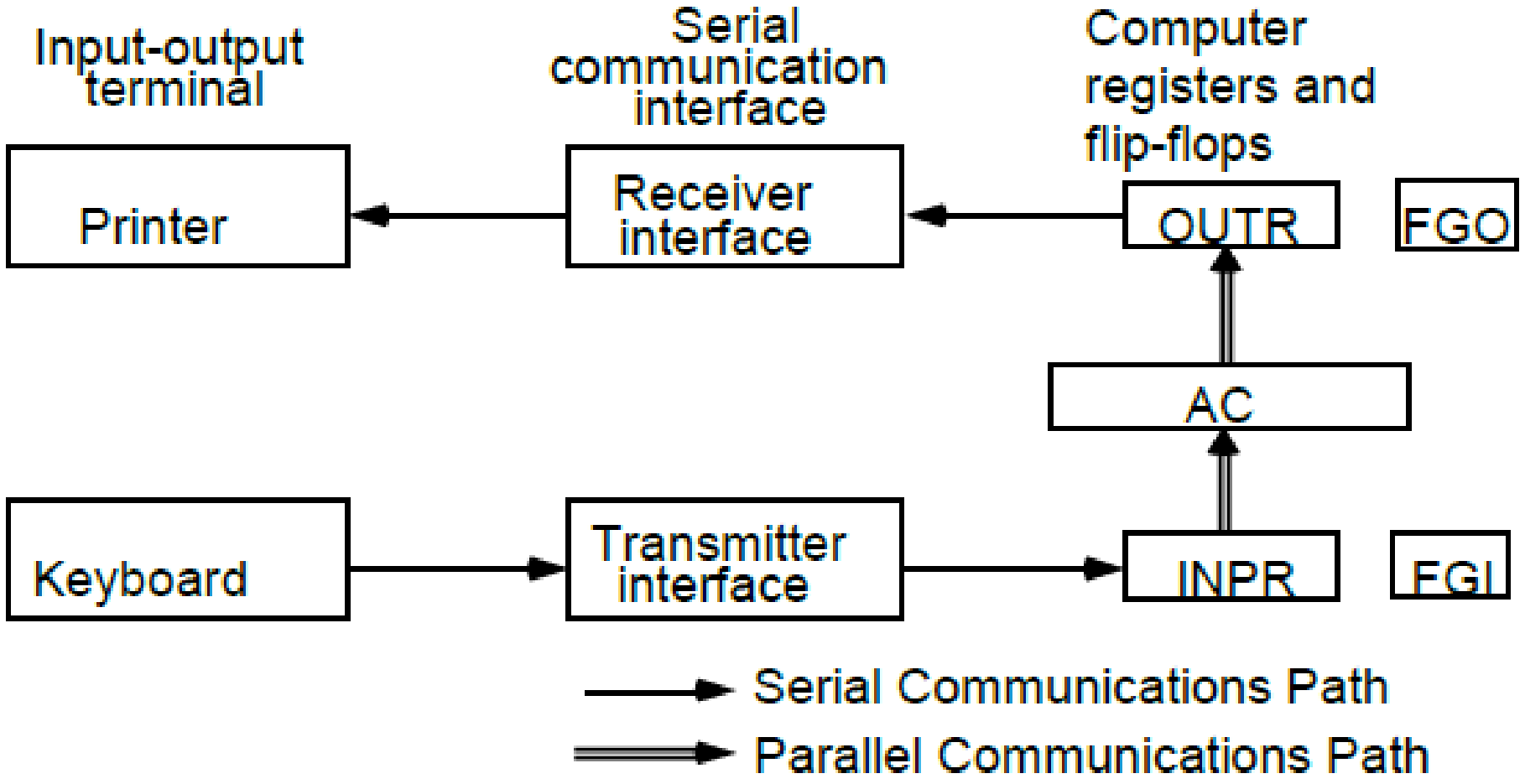
- If I = 1, indirect addressing mode instruction during T3.
- If I = 0, direct addressing mode instruction during T3.

The SC is reset after executing each instruction.

Input-output configuration

- The terminal sends and receives serial information.
- Each quantity of information has 8 bits of an alphanumeric code.
- Two basic computer registers INPR and OUTR communicate with a communication interfaces.
  - **INPR:** Input register - 8 bits
  - **OUTR:** Output register- 8 bits
  - **FGI:** Input flag - 1 bit (Is a control flip-flop, set to 1 when new information is available)
  - **FGO:** Output flag - 1 bit
  - **IEN**: Interrupt enable - 1 bit

IO communication



Illustration

- When a key is struck in the keyboard, an 8-bit alphanumeric code is shifted into INPR and the input flag FGI is set to 1.
- As long as the flag is set, the information in INPR can not be changed by striking another key.
- The control checks the flag bit, if 1, contents of INPR is transferred in parallel to AC and FGI is cleared to 0. Once the flag is cleared, new information can be shifted into INPR by striking another key.
- OUTR works similarly but the direction of information flow is reversed. Initially FGO is set to 1.
- The computer checks the flag bit; if it is 1, the information is transferred in parallel to OUTR and FGO is cleared to 0. The output device accepts the coded information, prints the corresponding character and when operation is completed, it sets FGO to 1.

Program Interrupt

- Input and Output interactions with electromechanical peripheral devices require huge processing times compared with CPU processing times I/O (milliseconds) versus CPU (nano/micro-seconds)
- Interrupts permit other CPU instructions to execute while waiting for I/O to complete
- The I/O interface, instead of the CPU, monitors the I/O device.
- When the interface find that the I/O device is ready for data transfer, it generates an interrupt request to the CPU
- Upon detecting an interrupt, the CPU stops momentarily the task it is doing, branches to the service routine to process the data transfer, and then returns to the task it was performing.

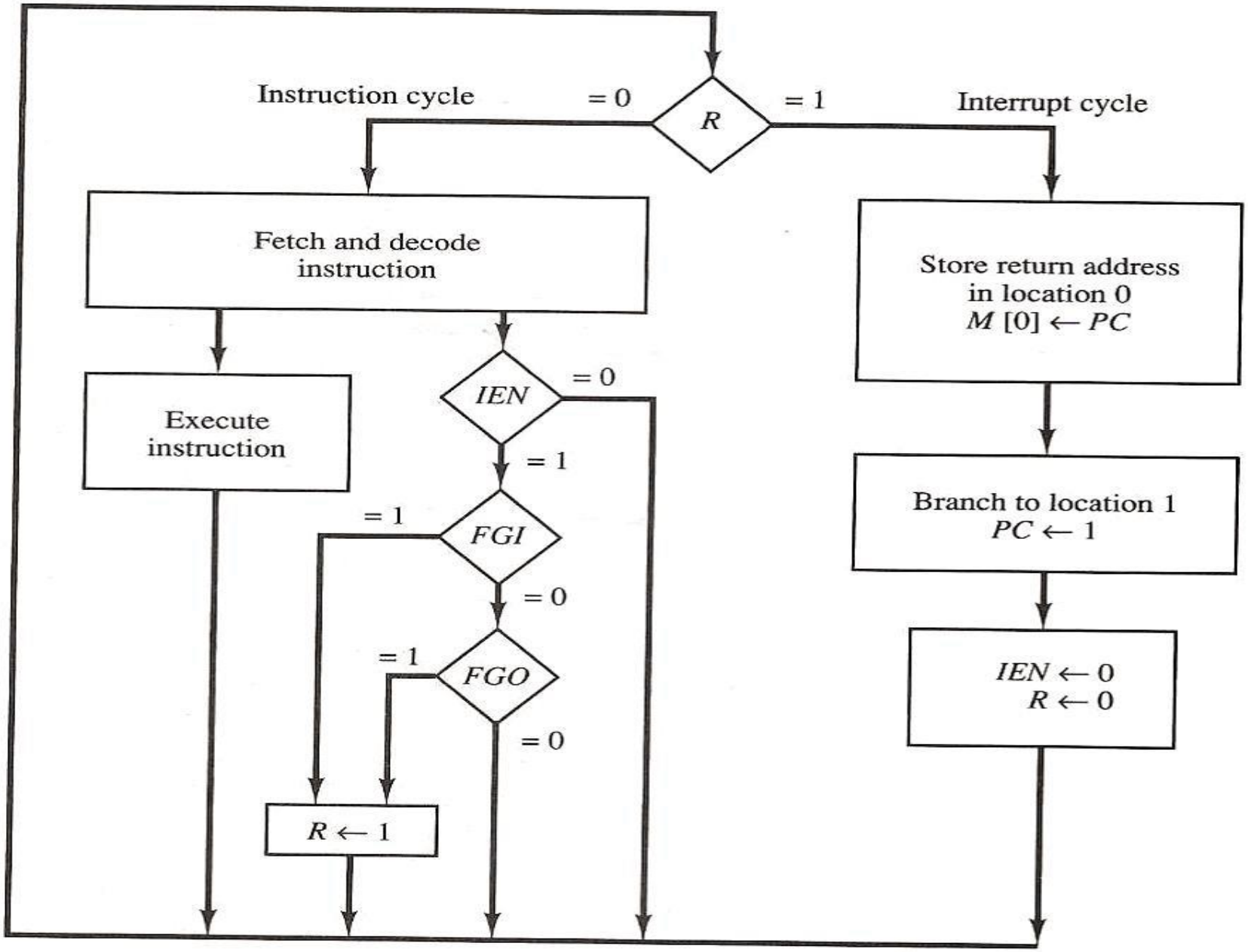


fig: flowchart of interrupt cycle

Description

The way that the interrupt is handled by the computer can be explained by means of the flowchart shown in figure :

- An interrupt flip-flop R is included in the computer.
- When R = 0, the computer goes through an instruction cycle.
- During the execute phase of the instruction cycle IEN is checked by the control.

If it is 0, it indicates that the programmer does not want to use the interrupt, so control continues with the next instruction cycle.   If IEN is 1, control checks the flag bits.

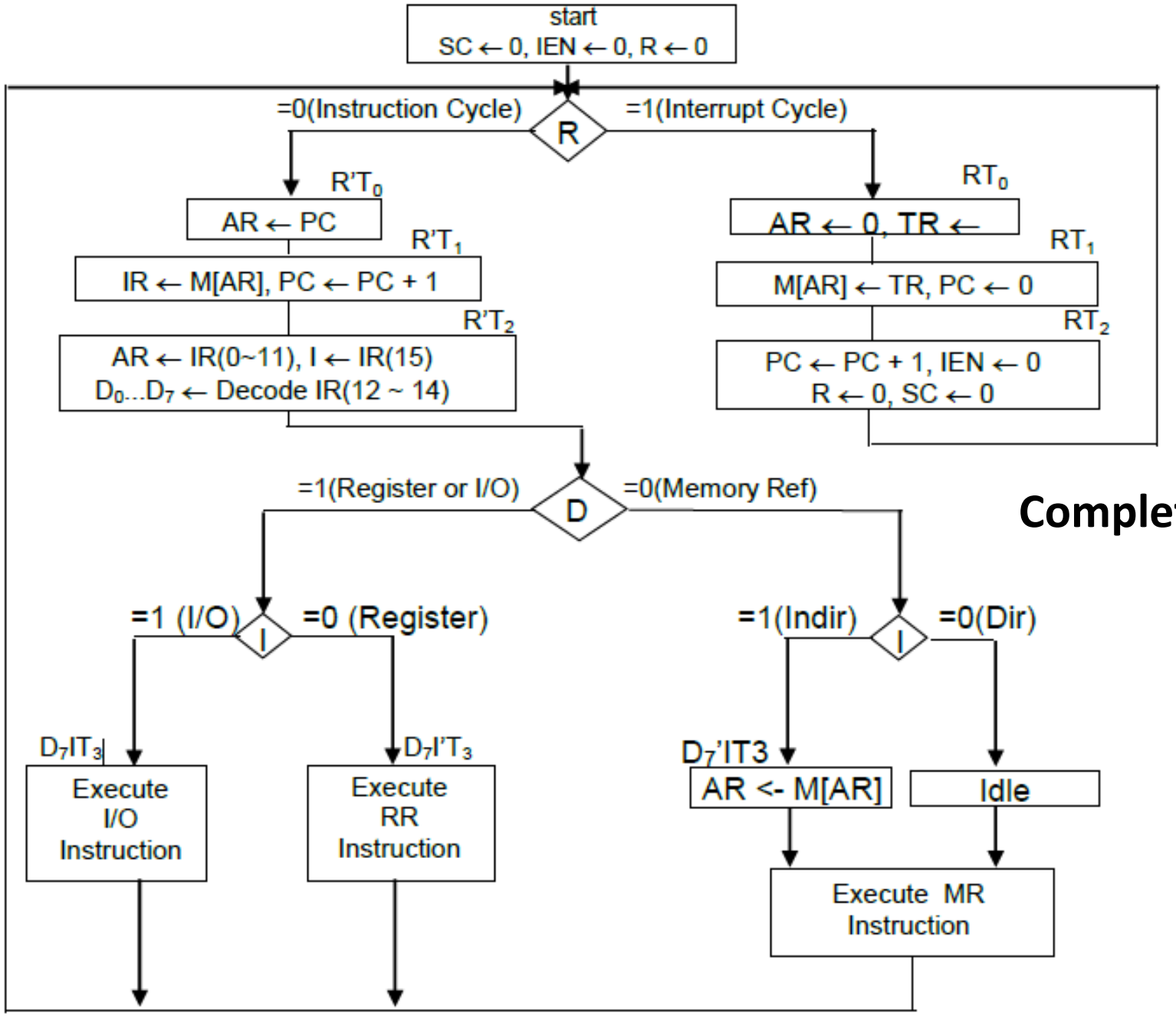
- If both flags are 0, it indicates that neither the input nor the output registers are ready for transfer of information.
- In this case, control continues with the next instruction cycle. If either flag is set to 1 while IEN = 1, flip-flop R is set to 1.
- At the end of the execute phase, control checks the value of R, and if it is equal to 1, it goes to an interrupt cycle instead of an instruction cycle.

Register transfer statements for the interrupt cycle

- The flip-flop is set to 1 if IEN = 1 and either FGI or FGO are equal to 1.
- This can happen with any clock transition except when timing signals T0, T1 or T2 are active.
- The condition for setting flip-flop R= 1 can be expressed with the following register transfer statement:

$$T0'T1'T2'\,(IEN)\,(FGI + FGO): R \leftarrow 1$$

The symbol + between FGI and FGO in the control function designates a logic OR operation. This is AND with IEN and T0'T1'T2' .



Complete description of Basic Computer

Description

The final flowchart of the instruction cycle, including the interrupt cycle for the basic computer, is shown in Figure

- The interrupt flip-flop R may be set at any time during the indirect or execute phases. The control returns to timing signal T0 after SC is cleared to 0.
- If R = 1, the computer goes through an interrupt cycle. If R = 0, the computer goes through an instruction cycle.
- If the instruction is one of the memory-reference instructions, the computer first checks if there is an indirect address and then continues to execute the decoded instruction according to the flowchart.
- If the instruction is one of the register-reference instructions, it is executed with one of the microoperations register reference.
- If it is an input-output instruction, it is executed with one of the microoperation's inputoutput reference.

END OF CHAPTER 2