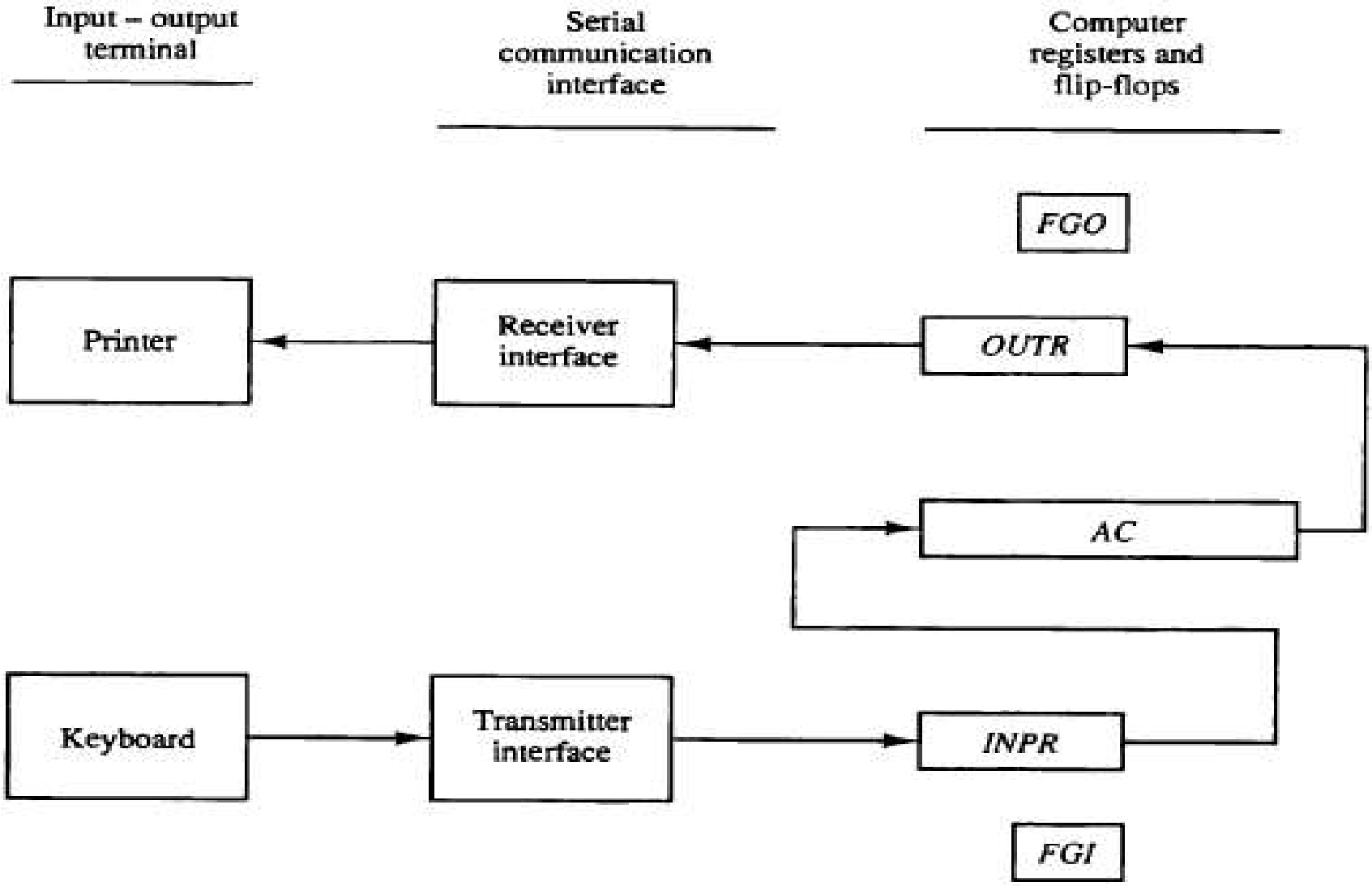
Input-Output and Interrupt

- Instruction and data stored in memory must come from some input device. Computational results must be transmitted to the user through some output device.
- **Input-Output Configuration:** The terminal sends and receives serial information. Each quantity of information has 8-bits of an alphanumeric code.
- The INPR and OUTR communicate with communication interface serially and with the AC in parallel.
- The transmitter interface receives serial information from the keyboard and transmits it to the INPR.
- The receiver interface receives information from OUTR and sends it to the printer serially.
- The 1-bit input flag (FGI) and output flag (FGO) are control flip-flop.
- The flag is needed to synchronize the timing rate difference between the input-output device and the computer.

Continue…

Figure Input-output configuration



Continue…

- **Input register (INPR):** FGI = 1, when new information is available in the input device and cleared (FGI = 0), when the information is accepted by the computer.
- Initially, FGI = 0 (cleared), FGI is set to 1, when a key is struck in the keyboard and 8-bit alphanumeric code is shifted into INPR. As long as FGI is set, the information in INPR cannot be changed by striking another key.
- The computer checks the flag bit; if it is 1, the information from INPR is transferred in parallel into AC and FGI is cleared to 0. Once flag is cleared, new information can be shifted into INPR by striking another key.
- **Output register (OUTR):** Initially, FGO = 1, the computer checks the flag bit; if it is 1, the information from AC is transferred in parallel to OUTR and FGO = 0 (cleared).
- The output device accepts the coded information, prints the corresponding character, and when the operation is completed, it sets FGO to 1.
- The computer does not load a new character into OUTR when FGO is 0 because this condition indicates that the output device is in the process of printing the character.

Continue…

TABLE Input-Output Instructions

$D_7IT_3 = p$ (common to all input–output instructions)
$IR(i) = B_i$ [bit in $IR(6–11)$ that specifies the instruction]

	p:	$SC \leftarrow 0$	Clear SC
INP	pB_{11}:	$AC(0–7) \leftarrow INPR$, $FGI \leftarrow 0$	Input character
OUT	pB_{10}:	$OUTR \leftarrow AC(0–7)$, $FGO \leftarrow 0$	Output character
SKI	pB_9:	If $(FGI = 1)$ then $(PC \leftarrow PC + 1)$	Skip on input flag
SKO	pB_8:	If $(FGO = 1)$ then $(PC \leftarrow PC + 1)$	Skip on output flag
ION	pB_7:	$IEN \leftarrow 1$	Interrupt enable on
IOF	pB_6:	$IEN \leftarrow 0$	Interrupt enable off

Continue…

- **Program Interrupt:** When IEN (interrupt enable flip-flop) is cleared to 0 (with the IOF instruction), the flags cannot interrupt the computer.
- When IEN is set to 1 (with the ION instruction), the computer can be interrupted.
- When R (interrupt flip-flop) = 0, the computer goes through an instruction cycle. During the execution phase of the instruction cycle, IEN is checked by the control.
- If IEN = 0, it indicates that the programmer does not want to use the interrupt, so control continues with the next instruction cycle. If IEN = 1, control checks the flag bits (FGI and FGO).
- 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 neither flag set to 1 while, IEN = 1, flip-flop R is set to 1. At the end of the execution phase, control checks the value of R, and if R = 1, it goes to an interrupt cycle instead of an instruction cycle.

Continue…

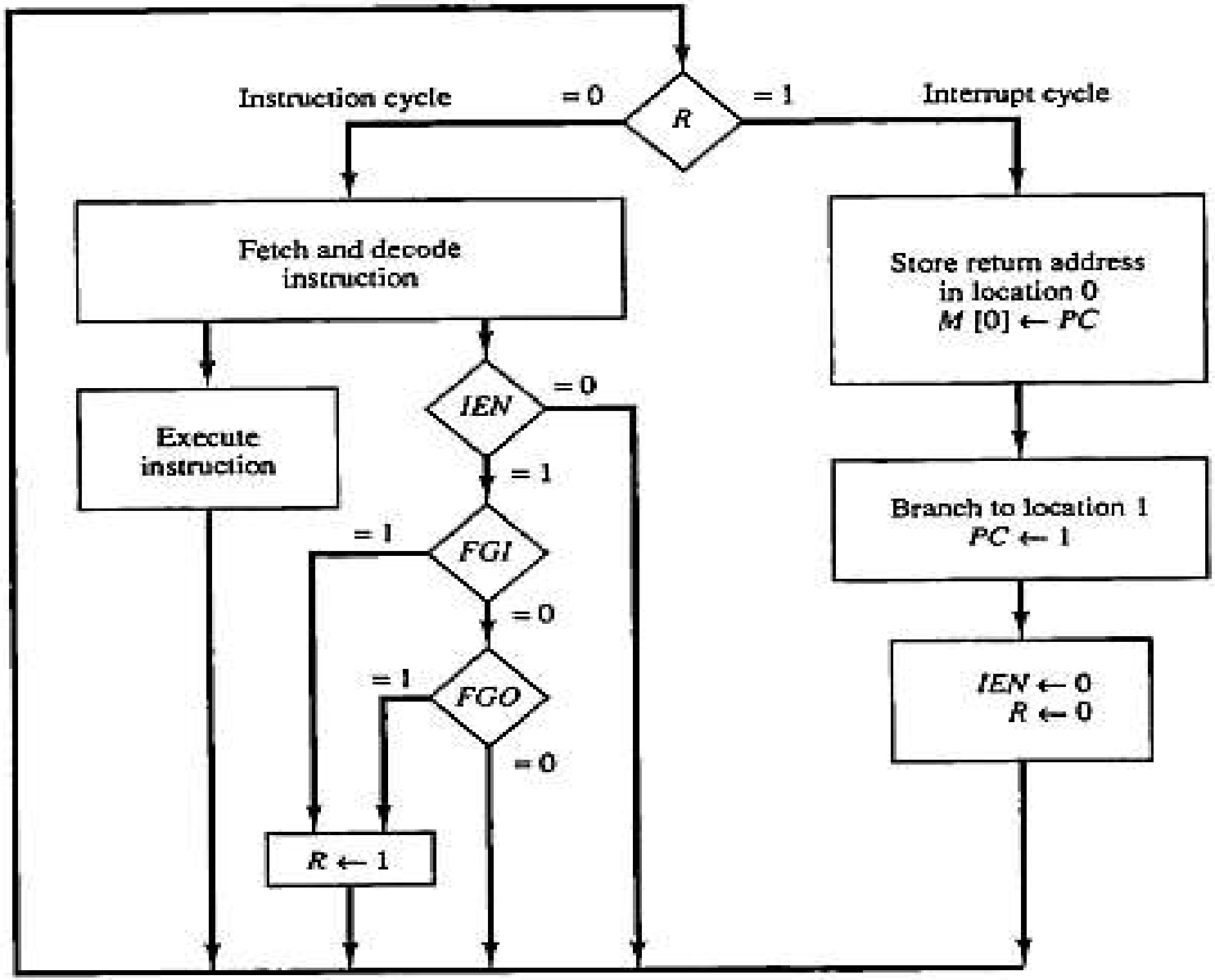


Figure Flowchart for interrupt cycle

Continue…

- The interrupt flip-flop (R) is set to 1 if IEN = 1 and either FGI or FGO = 1. This can happen with any clock transition except when timing signals T_0, T_1, or T_2 are active.

$$T'_0T'_1T'_2(IEN)(FGI + FGO): \quad R \leftarrow 1$$

- When control reaches T_0 and finds that R = 1 (interrupt occurs), it proceeds with the interrupt cycle, while the control is executing the instruction address 255. At this time, the return address 256 is in PC.

RT_0: $AR \leftarrow 0$, $TR \leftarrow PC$

$TR \leftarrow$ content of 256 (return address)

RT_1: $M[AR] \leftarrow TR$, $PC \leftarrow 0$

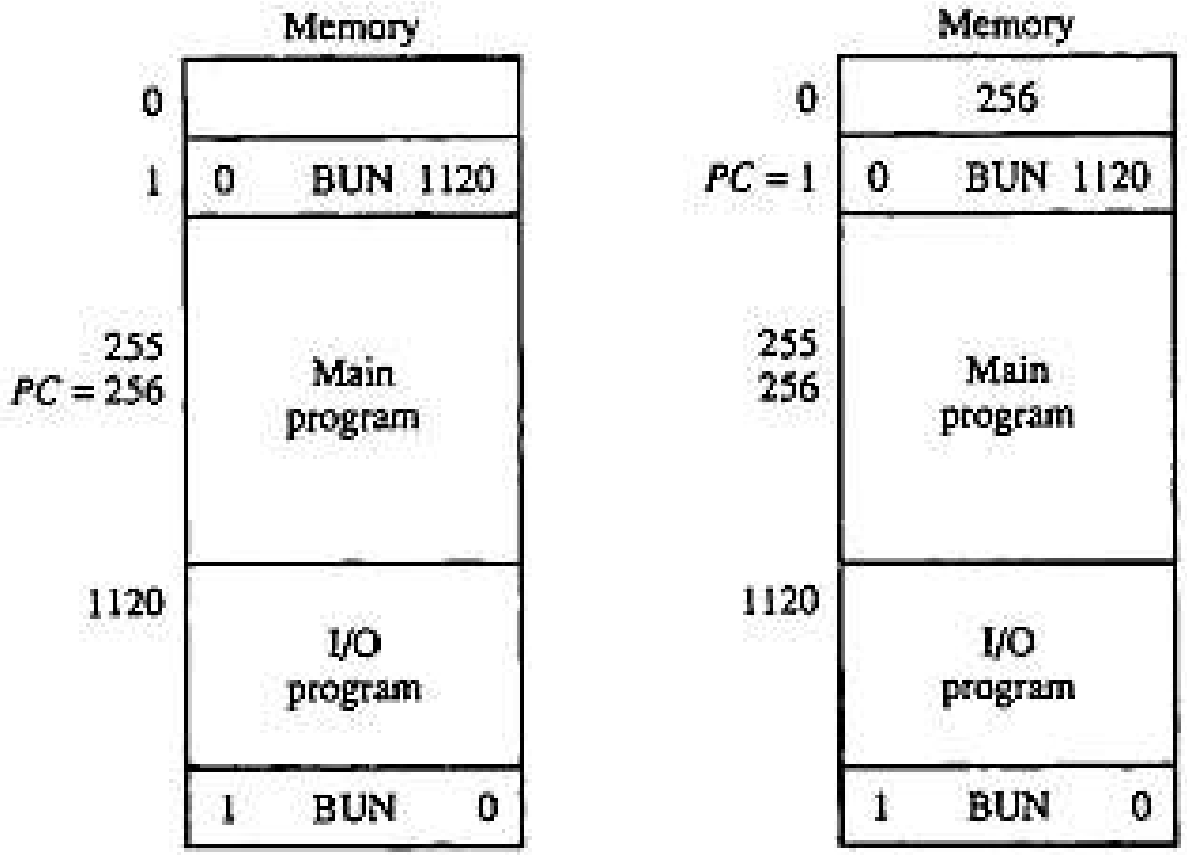
$M[0] \leftarrow$ content of 256 (return address)

RT_2: $PC \leftarrow PC + 1$, $IEN \leftarrow 0$, $R \leftarrow 0$, $SC \leftarrow 0$

$PC \leftarrow 0 + 1$

Continue…

Figure Demonstration of the interrupt cycle



Continue…

- The programmer has previously placed an I/O service program in memory starting from address 1120 and a BUN 1120 instruction at address 1.
- At the beginning of the next instruction cycle, the branch instruction that is read from memory is in address 1 (PC). This causes the program to transfer to the I/O service program at address 1120.
- This program checks the flags, determines which flag (FGI or FGO) is set, and then transfer the required input or output information.
- Once this is done, the ION instruction is executed to set IEN to 1 (to enable further interrupt), and the program returns to the location where it was interrupted.
- The instruction that returns the computer to the original place in the main program is a branch indirect instruction with an address part of 0. This instruction is placed at the end of the I/O service program.
- After this instruction is read from memory during fetch phase, control goes to read the effective address. The effective address is in location 0 and is the return address that was stored during the previous interrupt cycle.
- The execution of BUN instruction results in placing into PC the return address (256) from location 0.

Design of Basic Computer

- The basic computer consists of the following hardware components:

➢ A Memory Unit with 4096 words of 16 bits each

➢ Eight registers: AR, PC, DR, AC, IR, TR, OUTR, and INPR

➢ Seven flip-flops: I, S, E, R, IEN, FGI, and FGO

➢ Two decoders:3 x 8 operation decoder and a 4 x 16 timing decoder

➢ A 16-bit common bus

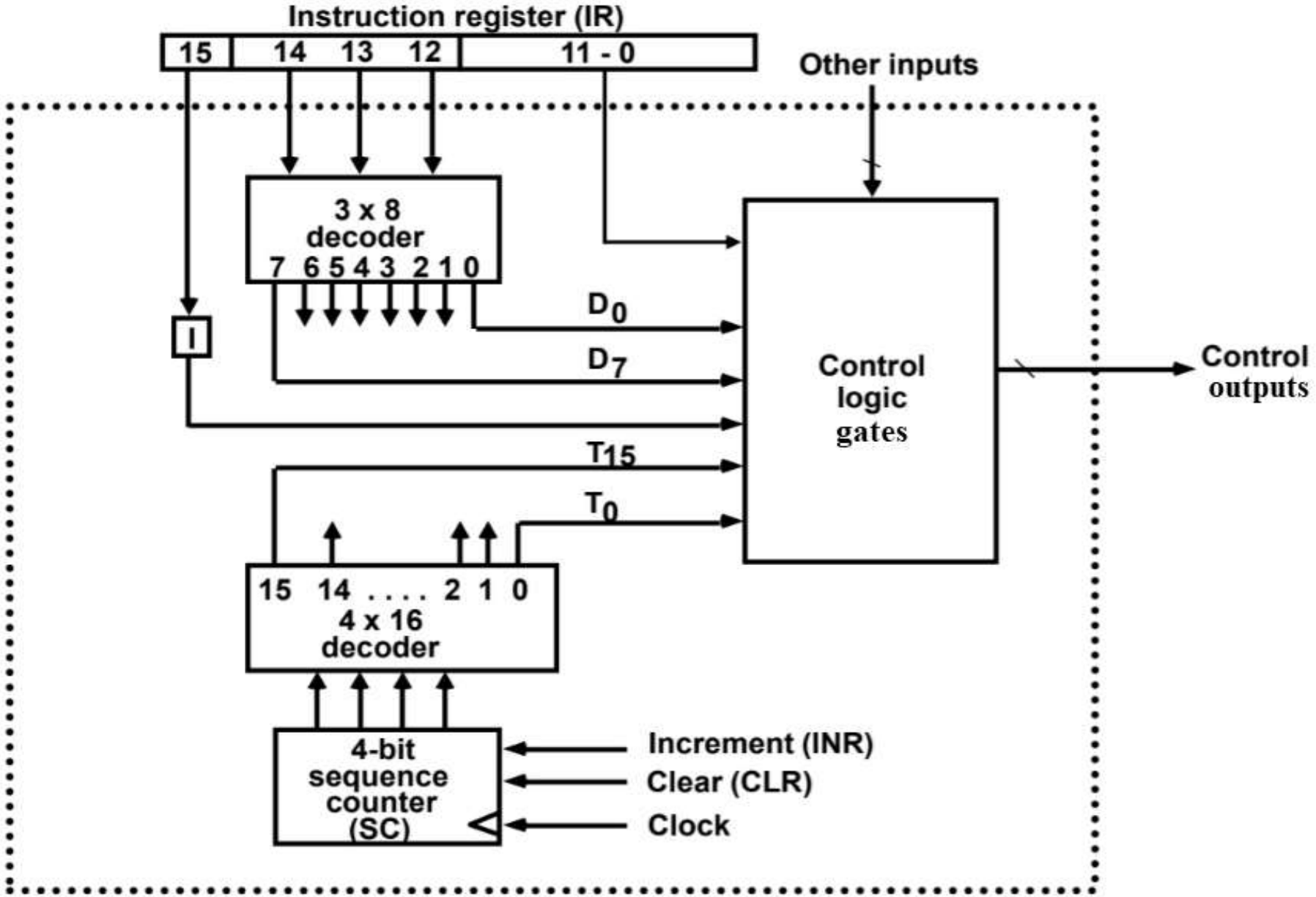
➢ Control logic gates

➢ Adder and logic circuit connected to the input of AC

Continue…

- **Control logic gates:**
- **The inputs for the control logic gate come from:** the two decoders, the I flip-flop, and bits 0-11 of IR (Instruction Register).
- **The other inputs to the control logic gate include:**
  - AC (Accumulator) bits 0-15, to check if AC=0 and to detect the sign bit in AC(15)
  - DR (Data Register) bits 0-15, to check if DR=0 and check the values of the seven flip-flops.
- **The outputs of the control logic circuit are as follows:**
- Signals to control the read and write inputs of memory
- Signals to control the inputs of the eight registers
- Signals to control the AC adder and logic circuit
- Signals to control the S2, S1, and S0 to select a register for the bus
- Signals to set, clear or complement the flip-flops

Continue…



Block diagram of Control Logic Gates

Continue…

- **Control of Registers and Memory:**
- **The control inputs or signals of the registers are:** LD (load), INR (increment), and CLR (clear).
- Let's consider the case of AR to understand the control of registers and memory. If we want to derive the gate structure associated with the control input of AR, we consider all the statements that change the content of AR:

R'T0: AR ← PC
R'T2: AR ← IR (0-11)
D7'IT3: AR ← M [AR]
RT0: AR ← 0 (clears AR)
D5T4: AR ← AR+1 (increment AR by 1)

- The first three statements specify transfer of information from a register or memory to AR. The content of the source register or memory is placed on the bus and the content of the bus is transferred into AR by enabling its LD control input.
- The control functions can be combined into three Boolean expressions as follows:

LD (AR) = R'T0 + R'T2 + D7'IT3 (load input of AR)
CLR (AR) = RT0 (clear input of AR)
INR (AR) = D5T4 (increment input of AR)

- In a similar fashion we can derive the control gates for the other registers as well as the logic needed to control the read and write inputs of memory.

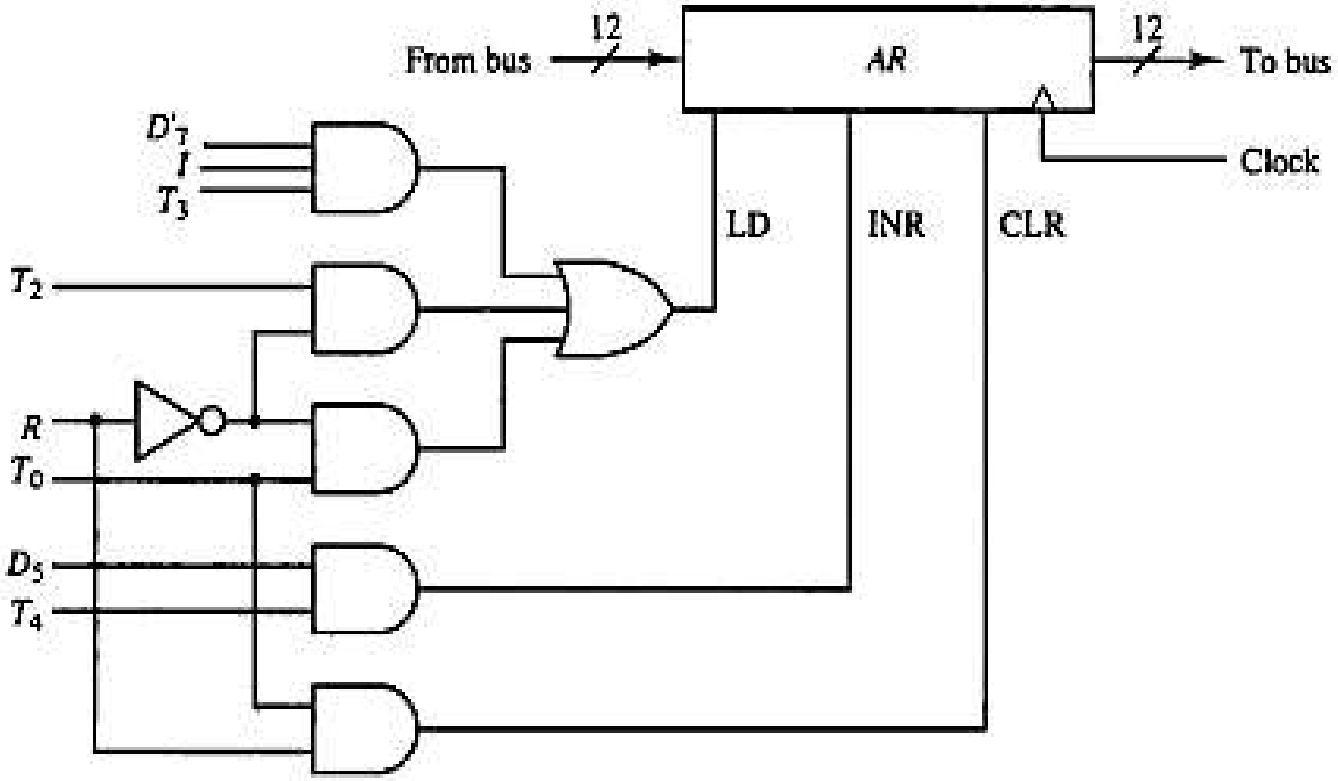
Continue…

- The read operation is recognized from the symbol ←M[AR]

$$Read = R'T1 + D7'IT3 + (D0 + D1 + D2 + D6)T4$$

- The output of the logic gates that implement the Boolean expression above must be connected to the read input of memory.

Figure Control gates associated with AR.



Continue…

- **Control of single flip-flops:**
- To understand the control of flip-flops, we will observe the control of IEN (Interrupt enable) flip-flop. Now, the IEN may change as a result of the two instructions, ION and IOF.

ION PB7: IEN ← 1

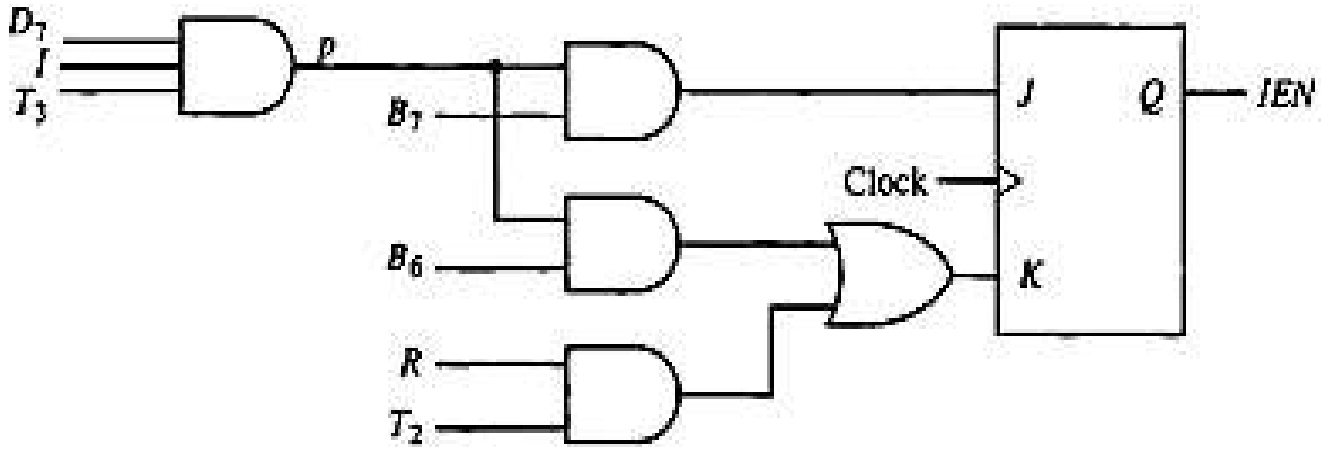
IOF PB6: IEN ← 0

- Where P = D7IT3 and B7 and B6 are bits 7 and 6 of IR, respectively. Moreover, at the end of the interrupt cycle, IEN is cleared to 0.

RT2: IEN ← 0

- The control logic gate for IEN is given below:

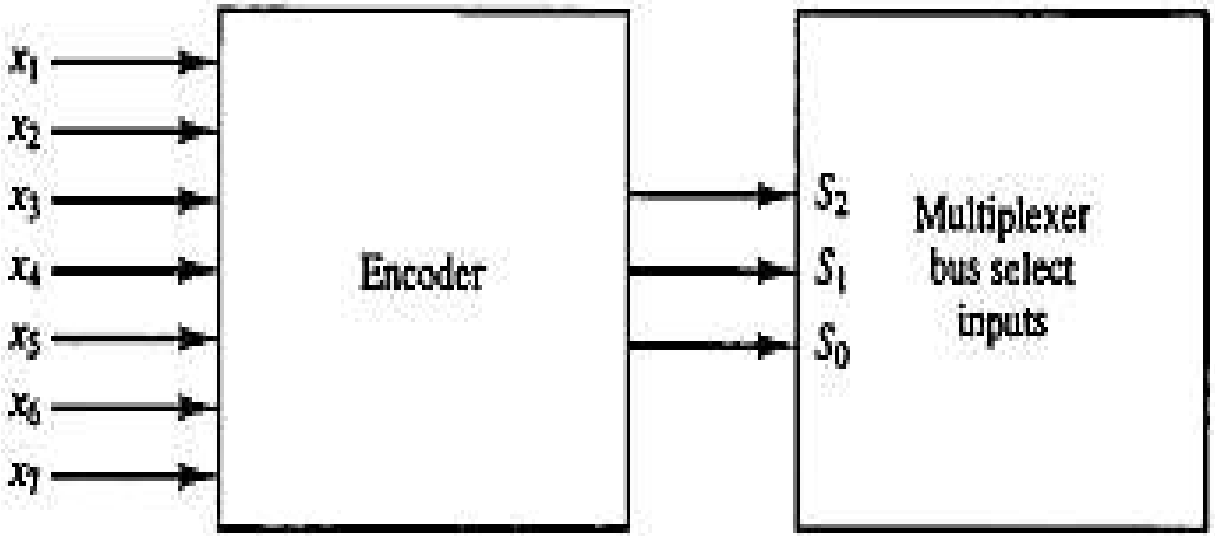
Figure Control inputs for IEN



Continue…

- **Control of Common Bus:**
- We use an encoder for the bus selection circuit where the output of the encoder determines the data of which register is to be transferred on the bus.
- The block diagram of the encoder used is given below:

Figure Encoder for bus selection inputs.



Continue…

TABLE Encoder for Bus Selection Circuit

Inputs							Outputs			Register selected for bus
x_1	x_2	x_3	x_4	x_5	x_6	x_7	S_2	S_1	S_0	
0	0	0	0	0	0	0	0	0	0	None
1	0	0	0	0	0	0	0	0	1	*AR*
0	1	0	0	0	0	0	0	1	0	*PC*
0	0	1	0	0	0	0	0	1	1	*DR*
0	0	0	1	0	0	0	1	0	0	*AC*
0	0	0	0	1	0	0	1	0	1	*IR*
0	0	0	0	0	1	0	1	1	0	*TR*
0	0	0	0	0	0	1	1	1	1	Memory

- The Boolean function for the encoder outputs can be written as:

$$S0 = X1 + X3 + X5 + X7$$

$$S1 = X2 + X3 + X6 + X7$$

$$S2 = X4 + X5 + X6 + X7$$

Continue…

- It is necessary to find the control functions that place the corresponding register onto the bus to determine the logic for each encoder input. Let's find this for X1 input of encoder.
- First, we will scan all the register transfer statements and extract the statements having AR as a source. We get the following instructions after this:

$$D4T4: \quad PC \leftarrow AR$$

$$D5T5: \quad PC \leftarrow AR$$

- So the Boolean expression for X1 becomes

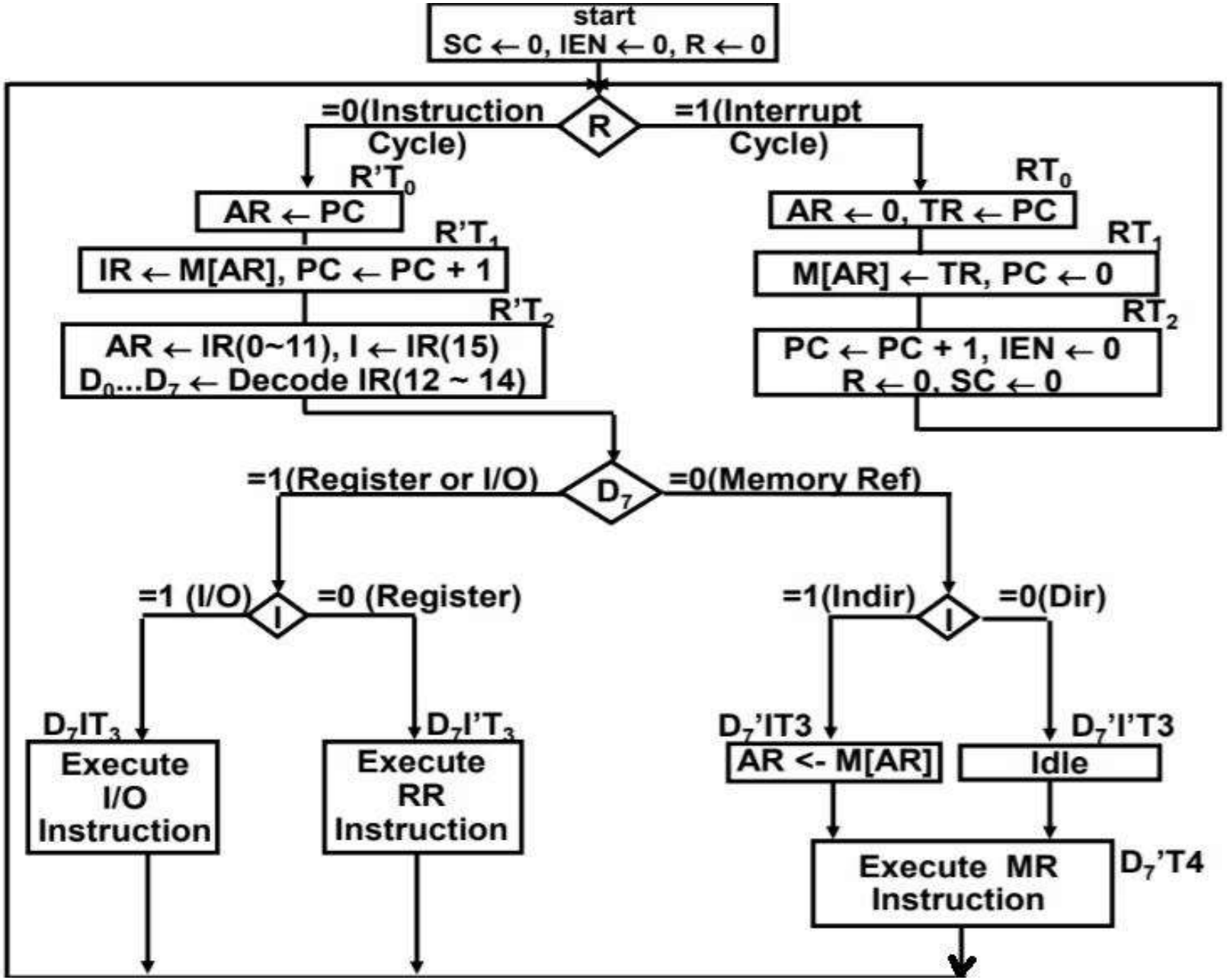
$$X1 = D4T4 + D5T5$$

- Similarly, we can find the gate logic for X2 to X7 for different registers.
- The data output from memory are selected for the bus when X7 = 1 and S2S1S0 = 111. The gate logic that generates X7 must also be applied to the read input of memory. Therefore, the Boolean function for X7 is the same as the one derived previously for the read operation.

$$X7 = R'T1 + D7'IT3 + (D0 + D1 + D2 + D6)T4$$

- In the similar manner we can determine the gate logic for the other registers.

Continue



Flowchart for computer operation