Instruction Cycle

- A program residing in the memory unit of the computer consists of a sequence of instructions.
- The program is executed in the computer by going through a cycle for each instruction.
- Each instruction cycle in turn is subdivided into a sequence of subcycles or phases.
- In the basic computer each instruction cycle consists of the following phases:

Instruction Cycle

1. Fetch an instruction from memory.
2. Decode the instruction.
3. Read the effective address from memory if the instruction has an indirect address.
4. Execute the instruction.

❖ Upon the completion of step 4, the control goes back to step 1 to fetch, decode, and execute the next instruction. This process continues indefinitely unless a HALT instruction is encountered. {The HALT instruction suspends operations}

Fetch and Decode

- Initially, the program counter PC is loaded with the address of the first instruction in the program.
- The sequence counter SC is cleared to 0, providing a decoded timing signal To.
- After each clock pulse, SC is incremented by one, so that the timing signals go through a sequence T0, T1, T2, and so on.
- Th rnicro operations for the fetch and decode phases can be specified by the following register transfer statements.

Fetch and Decode

T_0: $AR \leftarrow PC$
T_1: $IR \leftarrow M[AR]$, $PC \leftarrow PC + 1$
T_2: $D_0, \ldots, D_7 \leftarrow$ Decode $IR(12\text{–}14)$, $AR \leftarrow IR(0\text{–}11)$, $I \leftarrow IR(15)$

- Since only AR is connected to the address inputs of memory, it is necessary to transfer the address from PC to AR during the clock transition associated with timing signal T0.
- The instruction read from memory is then placed in the instruction register IR with the clock transition associated with timing signal T1.
- At the same time, PC is incremented by one to prepare it for the address of the next instruction in the program.

Fetch and Decode

- At time T2, the operation code in IR is decoded, the indirect bit is transferred to flip-flop I, and the address part of the instruction is transferred to AR.

- Note that SC is incremented after each clock pulse to produce the sequence To, T1, and T2.

Fetch and Decode

- Figure 5-8 shows how the first two register transfer statements are implemented in the bus system.

- To provide the data path for the transfer of PC to AR we must apply timing signal T0 to achieve the following connection:

- 1. Place the content of PC onto the bus by making the bus selection inputs S2S1S0 equal to 010.

- 2. Transfer the content of the bus to AR by enabling the LD input of AR .

Fetch and Decode

- The next clock transition initiates the transfer from PC to AR since T0 = 1. In order to implement the second statement.

$$T_1: \quad IR \leftarrow M[AR], \quad PC \leftarrow PC + 1$$

it is necessary to use timing signal T_1 to provide the following connections in the bus system.

1. Enable the read input of memory.
2. Place the content of memory onto the bus by making $S_2S_1S_0 = 111$.
3. Transfer the content of the bus to IR by enabling the LD input of IR.
4. Increment PC by enabling the INR input of PC.

Fetch and Decode

- The next clock transition initiates the read and increment operations since T, = 1 .

- Figure 5-8 duplicates a portion o f the bus system and shows how T0 and T1, are connected to the control inputs of the registers, the memory, and the bus selection inputs.

- **Multiple input OR gates are included in the diagram because there are other control functions that will initiate similar operations.**

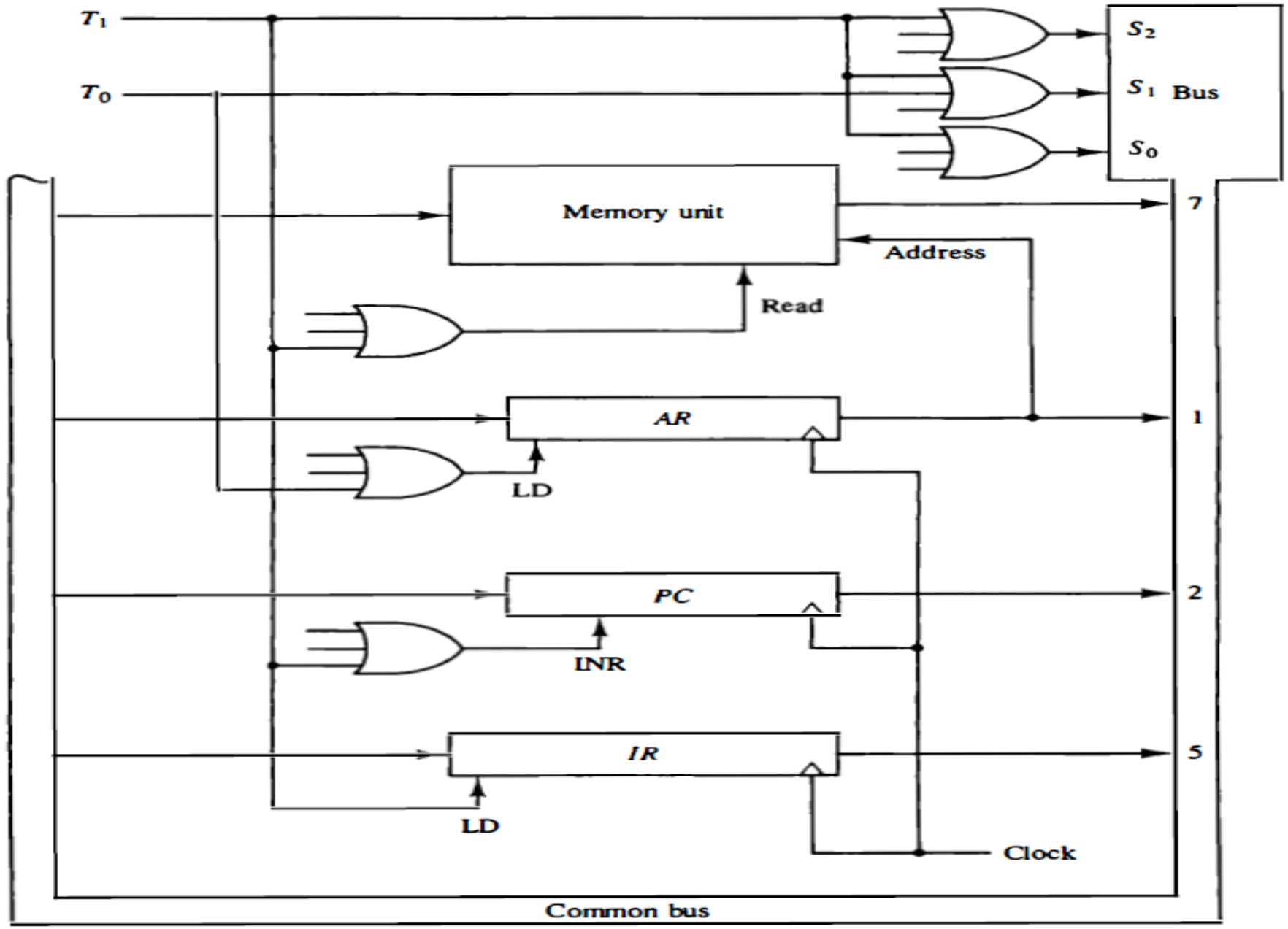


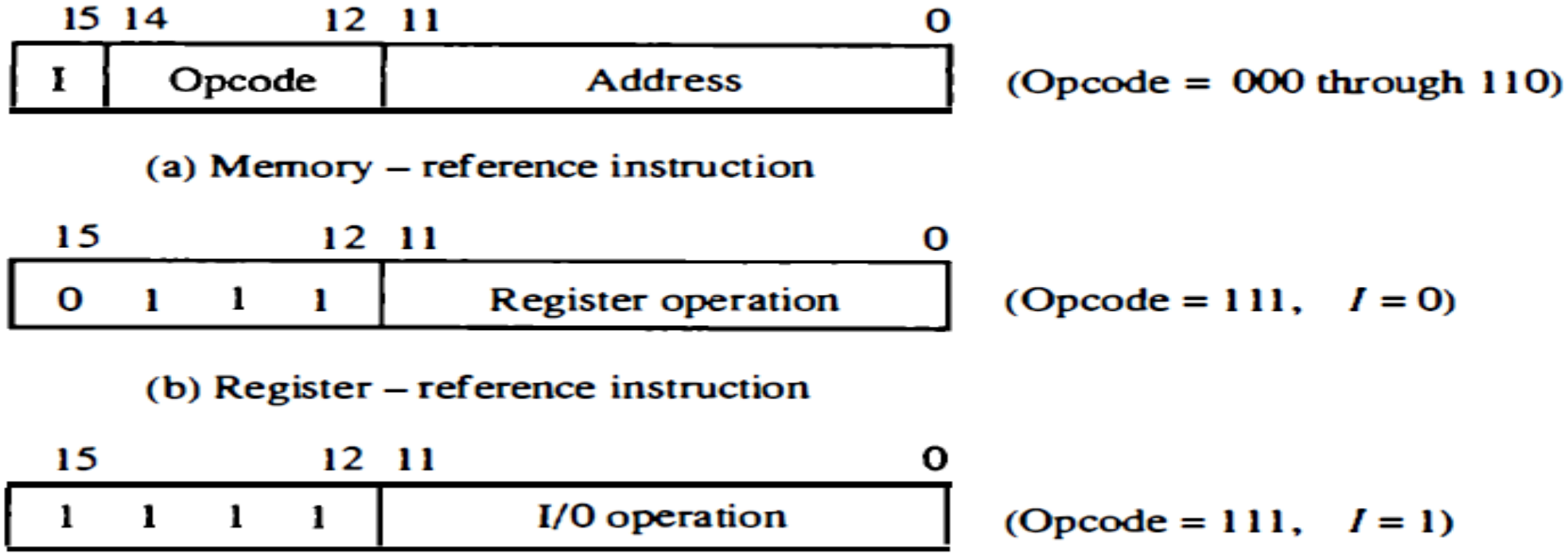
Figure 5-8 Register transfers for the fetch phase.

Determine the Type of Instruction

- The timing signal that is active after the decoding is T3.
- During time T3, the control unit determines the type of instruction that was just read from memory.
- The flowchart of Fig. 5-9 presents an initial configuration for the instruction cycle and shows how the control determines the instruction type after the decoding.
- The three possible instruction types available in the basic computer are specified in Fig. 5-5.

Computer Instructions

Figure 5-5 Basic computer instruction formats.



Control unit

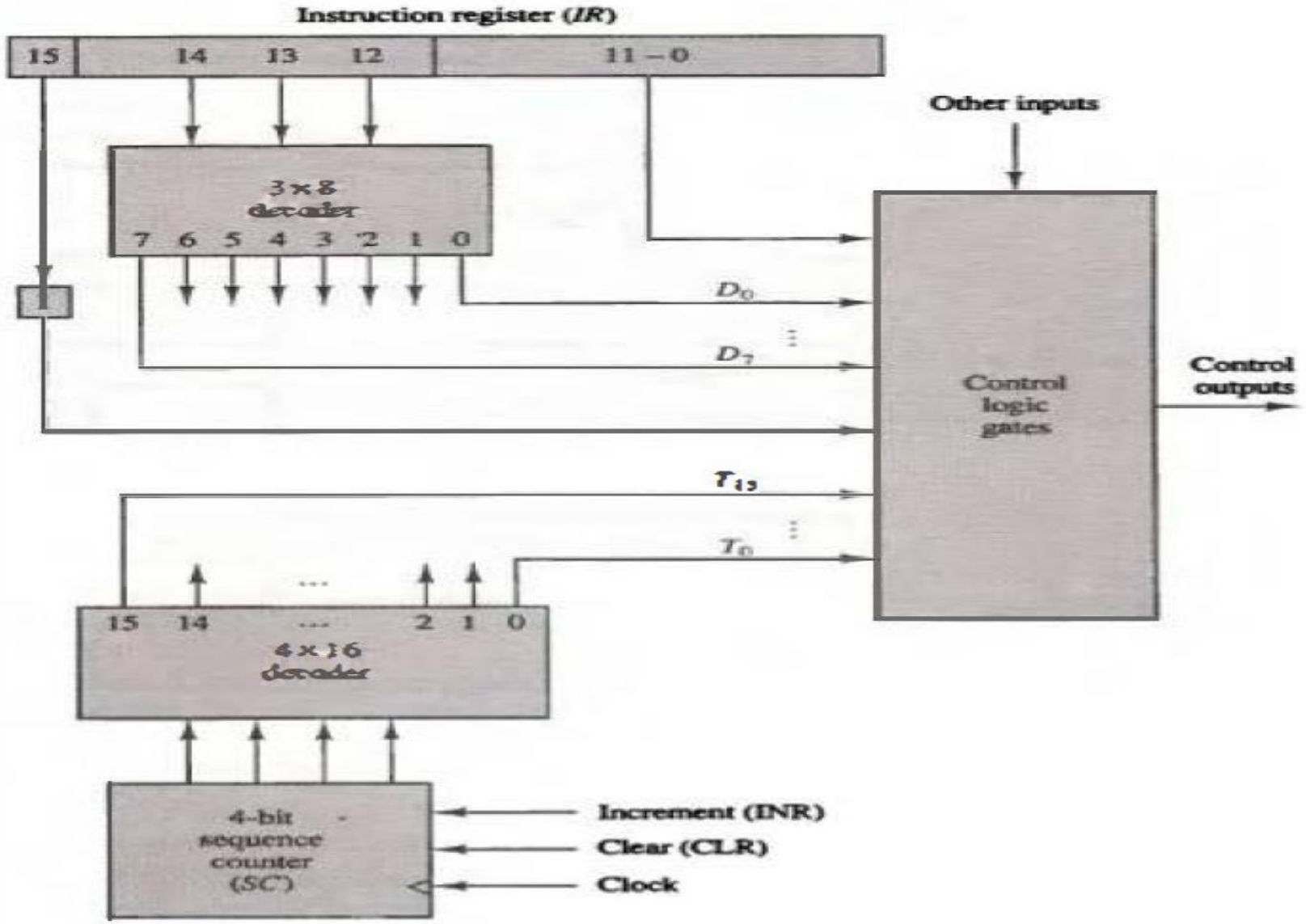


Figure 5-6 Control unit of basic computer.

Determine the Type of Instruction

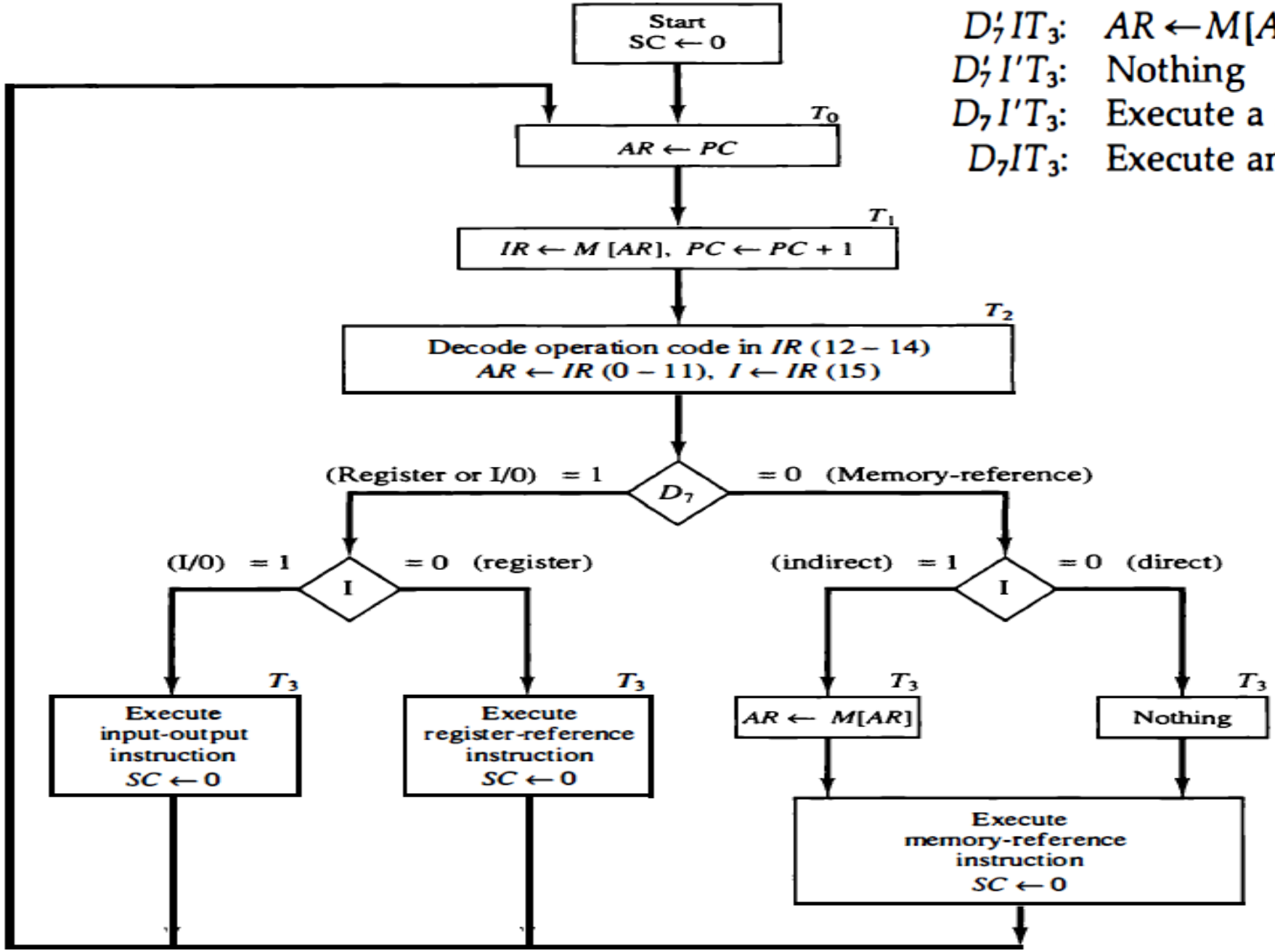
- Decoder output D, is equal to 1 if the operation code is equal to binary 111.
- From Fig. 5-5 we determine that if D7 = 1, the instruction must be a register-reference or input-output type.
- If D7, = 0, the operation code must be one of the other seven values 000 through 110, specifying a memory-reference instruction.
- Control then inspects the value of the first bit of the instruction, which is now available in flip-flop I. If D7 = 0 and I = 1, we have a memory reference instruction with an indirect address.

Determine the Type of Instruction

- effective address from memory. The microoperation for the indirect address condition can be symbolized by the register transfer statement

$$AR \leftarrow M[AR]$$

- Initially, AR holds the address part of the instruction.
- This address is used during the memory read operation.
- The word at the address given by AR is read from memory and placed on the common bus.
- The LD input of AR is then enabled to receive the indirect address that resided in the 12 least significant bits of the memory word.



$D'_7 IT_3$: $AR \leftarrow M[AR]$
$D'_7 I'T_3$: Nothing
$D_7 I'T_3$: Execute a register-reference instruction
$D_7 IT_3$: Execute an input–output instruction

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

Determine the Type of Instruction

The three instruction types are subdivided into four separate paths. The selected operation is activated with the clock transition associated with timing signal T_3. This can be symbolized as follows:

$D_7' I T_3$: $AR \leftarrow M[AR]$
$D_7' I' T_3$: Nothing
$D_7 I' T_3$: Execute a register-reference instruction
$D_7 I T_3$: Execute an input–output instruction

Register-Reference Instructions

- Register-reference instructions are recognized by the control when D7 = 1 and I = 0.
- These instructions use bits 0 through 11 of the instruction code to specify one of 12 instructions. These 12 bits are available in IR(0-11).
- They were also transferred to AR during time T2.

Register-Reference Instructions

- The control functions and microoperations for the register-reference are listed in Table 5-3.
- These instructions are executed with the clock transition associated with timing variable T3.
- Each control function needs the Boolean relation D7I' T3, which we designate for convenience by the symbol r .
- The control function is distinguished by one o f the bits in IR(0-11). By assigning the symbol Bi, to bit i of IR, all control functions can be simply denoted by rBi.

Register-Reference Instructions

- For example, the instruction CLA has the hexadecimal code 7800 (see Table 5-2), which gives the binary equivalent 01 1 1 1000 0000 0000. The first bit is a zero and is equivalent to I ‘ .

- The next three bits constitute the operation code and are recognized from decoder output D7. Bit 11 in IR is 1 and is recognized from B11.

- The control function that initiates the microoperation for this instruction is
- D7I ' T3 B11 = rB11.

- The execution of a register-reference instruction is completed at time T3. The sequence counter SC is cleared to 0 and the control goes back to fetch the next instruction with timing signal T0.

Register-Reference Instructions

- The first seven register-reference instructions perform clear, complement, circular shift, and increment micro-operations on the AC or E registers.
- The next four instructions cause a skip of the next instruction in sequence when a stated condition is satisfied.
- The skipping of the instruction is achieved by incrementing PC once again (in addition, it is being incremented during the fetch phase at time T1).
- The condition control statements must be recognized as part of the control conditions.

Register-Reference Instructions

- The AC is positive when the sign bit in AC(IS) = 0; it is negative when AC(IS) = I.
- The content of AC is zero (AC = 0) if all the flip-flops of the register are zero.
- The HLT instruction clears a start-stop flip-flop S and stops the sequence counter from counting.
- To restore the operation of the computer, the start-stop flip-flop must be set manually.

Register-Reference Instructions

TABLE 5-3 Execution of Register-Reference Instructions

$D_7I'T_3 = r$ (common to all register-reference instructions)
$IR(i) = B_i$ [bit in $IR(0–11)$ that specifies the operation]

	r:	$SC \leftarrow 0$	Clear SC
CLA	rB_{11}:	$AC \leftarrow 0$	Clear AC
CLE	rB_{10}:	$E \leftarrow 0$	Clear E
CMA	rB_9:	$AC \leftarrow \overline{AC}$	Complement AC
CME	rB_8:	$E \leftarrow \overline{E}$	Complement E
CIR	rB_7:	$AC \leftarrow \text{shr } AC,\ AC(15) \leftarrow E,\ E \leftarrow AC(0)$	Circulate right
CIL	rB_6:	$AC \leftarrow \text{shl } AC,\ AC(0) \leftarrow E,\ E \leftarrow AC(15)$	Circulate left
INC	rB_5:	$AC \leftarrow AC + 1$	Increment AC
SPA	rB_4:	If $(AC(15) = 0)$ then $(PC \leftarrow PC + 1)$	Skip if positive
SNA	rB_3:	If $(AC(15) = 1)$ then $(PC \leftarrow PC + 1)$	Skip if negative
SZA	rB_2:	If $(AC = 0)$ then $PC \leftarrow PC + 1$	Skip if AC zero
SZE	rB_1:	If $(E = 0)$ then $(PC \leftarrow PC + 1)$	Skip if E zero
HLT	rB_0:	$S \leftarrow 0$ (S is a start–stop flip-flop)	Halt computer

Memory-Reference Instructions

- Table 5-4 lists the seven memory-reference instructions.
- The decoded output D; for i = 0, 1, 2, 3, 4, 5, and 6 from the operation decoder that belongs to each instruction is included in the table.
- The effective address of the instruction is in the address register AR and was placed there during timing signal T2 when I = 0, or during timing signal T3 when I = 1.
- The execution of the memory-reference instructions starts with timing signal T4.
- The symbolic description of each instruction is specified in the table in terms of register transfer notation.

Control unit

- Note that the content of any

Memory-Reference Instructions

- The actual execution of the instruction in the bus system will require a sequence of micro-operations.
- This is because data stored in memory cannot be processed directly.
- The data must be read from memory to a register where they can be operated on with logic circuits.
- We now explain the operation of each instruction and list the control functions and micro-operations needed for their execution.
- A flowchart that summarizes all the microoperations is presented at the end of this section.

Memory-Reference Instructions

TABLE 5-4 Memory-Reference Instructions

Symbol	Operation decoder	Symbolic description
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR], \quad E \leftarrow C_{out}$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
BUN	D_4	$PC \leftarrow AR$
BSA	D_5	$M[AR] \leftarrow PC, \quad PC \leftarrow AR + 1$
ISZ	D_6	$M[AR] \leftarrow M[AR] + 1,$ If $M[AR] + 1 = 0$ then $PC \leftarrow PC + 1$

Memory-Reference Instructions

- **AND to AC**

- This is an instruction that performs the AND logic operation on pairs of bits in AC and the memory word specified by the effective address.

- The result of the operation is transferred to AC.

- The microoperations that execute this instruction are:

$$D_0T_4: \quad DR \leftarrow M[AR]$$
$$D_0T_5: \quad AC \leftarrow AC \wedge DR, \quad SC \leftarrow 0$$

Memory-Reference Instructions

- **ADD to AC**

- This instruction adds the content of the memory word specified by the effective address to the value of AC .

- The sum is transferred into AC and the output carry Cout is transferred to the E (extended accumulator) flip-flop.

- The microoperations needed to execute this instruction are

Memory-Reference Instructions

$$D_1T_4: \quad DR \leftarrow M[AR]$$
$$D_1T_5: \quad AC \leftarrow AC + DR, \quad E \leftarrow C_{out}, \quad SC \leftarrow 0$$

- The same two timing signals, T, and T5, are used again but with operation decoder D1 instead of D0, which was used for the AND instruction.

Memory-Reference Instructions

- LDA: Load to AC

- This instruction transfers the memory word specified by the effective address to AC . The microoperations needed to execute this instruction are

$$D_2T_4: \quad DR \leftarrow M[AR]$$
$$D_2T_5: \quad AC \leftarrow DR, \quad SC \leftarrow 0$$

Memory-Reference Instructions

STA: Store *AC*

This instruction stores the content of *AC* into the memory word specified by the effective address. Since the output of *AC* is applied to the bus and the data input of memory is connected to the bus, we can execute this instruction with one microoperation:

$$D_3T_4: \quad M[AR] \leftarrow AC, \quad SC \leftarrow 0$$

Memory-Reference Instructions

- BUN: Branch Unconditionally

- This instruction transfers the program to the instruction specified by the effective address. Remember that PC holds the address of the instruction to be read from memory in the next instruction cycle.

- PC is incremented at time T1 to prepare it for the address of the next instruction in the program sequence.

- The BUN instruction allows the programmer to specify an instruction out of sequence and we say that the program branches (or jumps) unconditionally. The instruction is executed with one microoperation:

Memory-Reference Instructions

$$D_4T_4: \quad PC \leftarrow AR, \quad SC \leftarrow 0$$

The effective address from AR is transferred through the common bus to PC. Resetting SC to 0 transfers control to T_0. The next instruction is then fetched and executed from the memory address given by the new value in PC.

Memory-Reference Instructions

- BSA: Branch and Save Return Address
- This instruction is useful for branching to a portion of the program called a subroutine or procedure.

- When executed, the BSA instruction stores the address of the next instruction in sequence (which is available in PC) into a memory location specified by the effective address.

- The effective address plus one is then transferred to PC to serve as the address of the first instruction in the subroutine.

$$M[AR] \leftarrow PC, \quad PC \leftarrow AR + 1$$

Memory-Reference Instructions

- A numerical example that demonstrates how this instruction is used with a subroutine is shown in Fig. 5-10. The BSA instruction is assumed to be in memory at address 20.

- The I bit is 0 and the address part of the instruction has the binary equivalent of 135.

- After the fetch and decode phases, PC contains 21, which is the address of the next instruction in the program (referred to as the return address).

- AR holds the effective address 135. This is shown in part (a) of the figure. The BSA instruction performs the following numerical operation:

Memory-Reference Instructions

$$M[135] \leftarrow 21, \quad PC \leftarrow 135 + 1 = 136$$

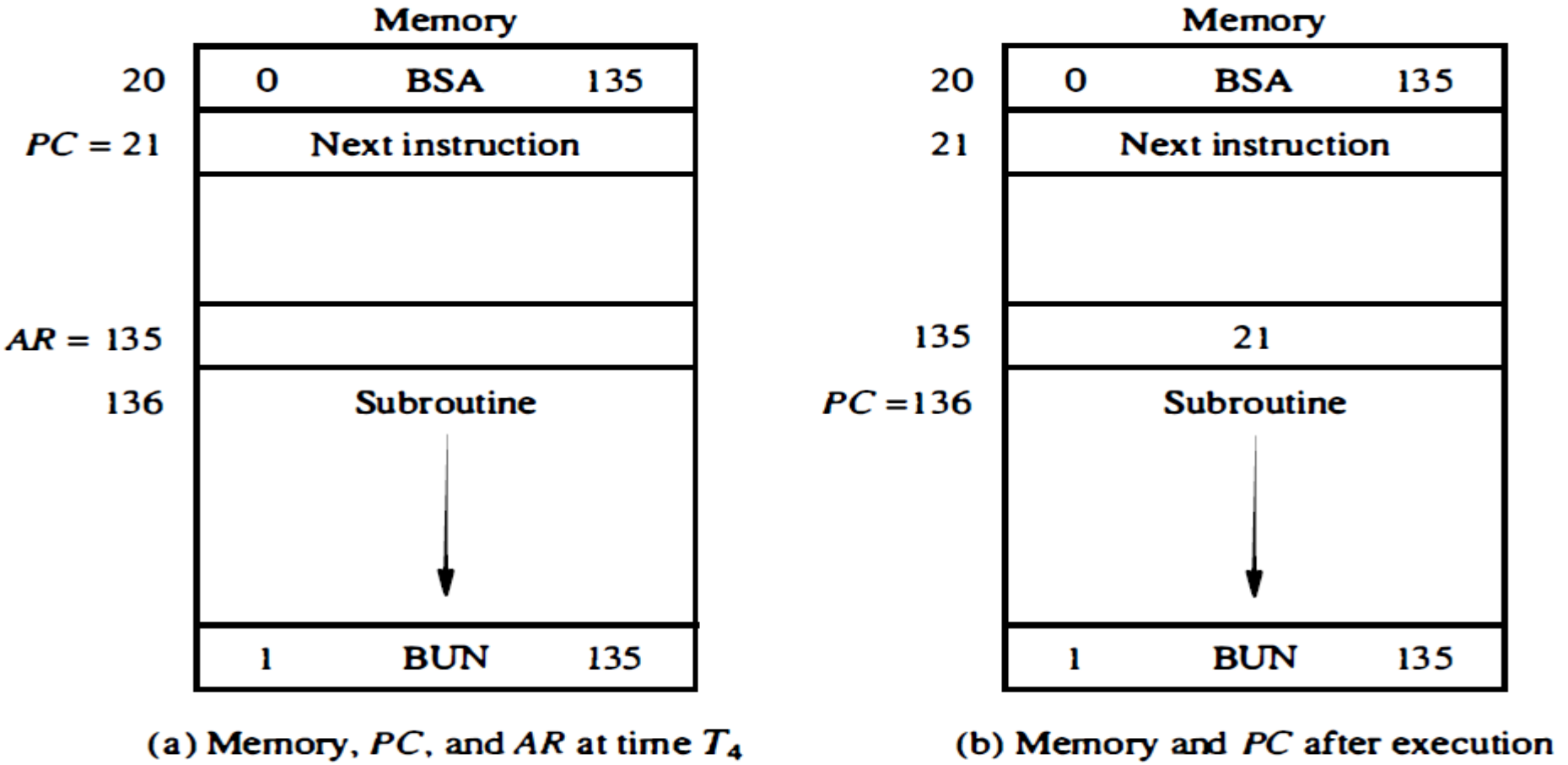
- The result of this operation is shown in part (b) of the figure.
- The return address 21 is stored in memory location 135 and control continues with the subroutine program starting from address 136.
- The return to the original program (at address 21) is accomplished by means of an indirect BUN instruction placed at the end of the subroutine.
- When this instruction is executed, control goes to the indirect phase to read the effective address at location 135, where it finds the previously saved address 21. When the BUN instruction is executed, the effective address 21 is transferred to PC . The next instruction cycle finds PC with the value 21, so control continues to execute the instruction at the return address.

Memory-Reference Instructions

- The BSA instruction performs the function usually referred to as a subroutine call.
- The indirect BUN instruction at the end of the subroutine performs the function referred to as a subroutine return.

Memory-Reference Instructions

Figure 5-10 Example of BSA instruction execution.



(a) Memory, *PC*, and *AR* at time T_4

(b) Memory and *PC* after execution

Memory-Reference Instructions

- To use the memory and the bus properly, the BSA instruction must be executed With a sequence of two microoperations

$$D_5T_4: \quad M[AR] \leftarrow PC, \quad AR \leftarrow AR + 1$$
$$D_5T_5: \quad PC \leftarrow AR, \quad SC \leftarrow 0$$

- Timing signal T4 initiates a memory write operation, places the content of PC onto the bus, and enables the INR input of AR .

- The memory write operation is completed and AR is incremented by the time the next clock transition occurs. The bus is used at T5 to transfer the content of AR to PC .

Memory-Reference Instructions

- This instruction increments the word specified by the effective address, and if the incremented value is equal to 0, PC is incremented by 1.

D_6T_4: $DR \leftarrow M[AR]$
D_6T_5: $DR \leftarrow DR + 1$
D_6T_6: $M[AR] \leftarrow DR$, if $(DR = 0)$ then $(PC \leftarrow PC + 1)$, $SC \leftarrow 0$