processor until the memory word is available. To facilitate the presentation, we will assume that a wait period is not necessary in the basic computer.

To fully comprehend the operation of the computer, it is crucial that one understands the timing relationship between the clock transition and the timing signals. For example, the register transfer statement

$$T_0: \quad AR \leftarrow PC$$

specifies a transfer of the content of *PC* into *AR* if timing signal T_0 is active. T_0 is active during an entire clock cycle interval. During this time the content of *PC* is placed onto the bus (with $S_2S_1S_0 = 010$) and the LD (load) input of *AR* is enabled. The actual transfer does not occur until the end of the clock cycle when the clock goes through a positive transition. This same positive clock transition increments the sequence counter *SC* from 0000 to 0001. The next clock cycle has T_1 active and T_0 inactive.

5-5 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:

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.

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 T_0. After each clock pulse, *SC* is incremented by one, so that the timing signals go through a sequence T_0, T_1, T_2, and so on. The microoperations for the fetch and decode phases can be specified by the following register transfer statements.

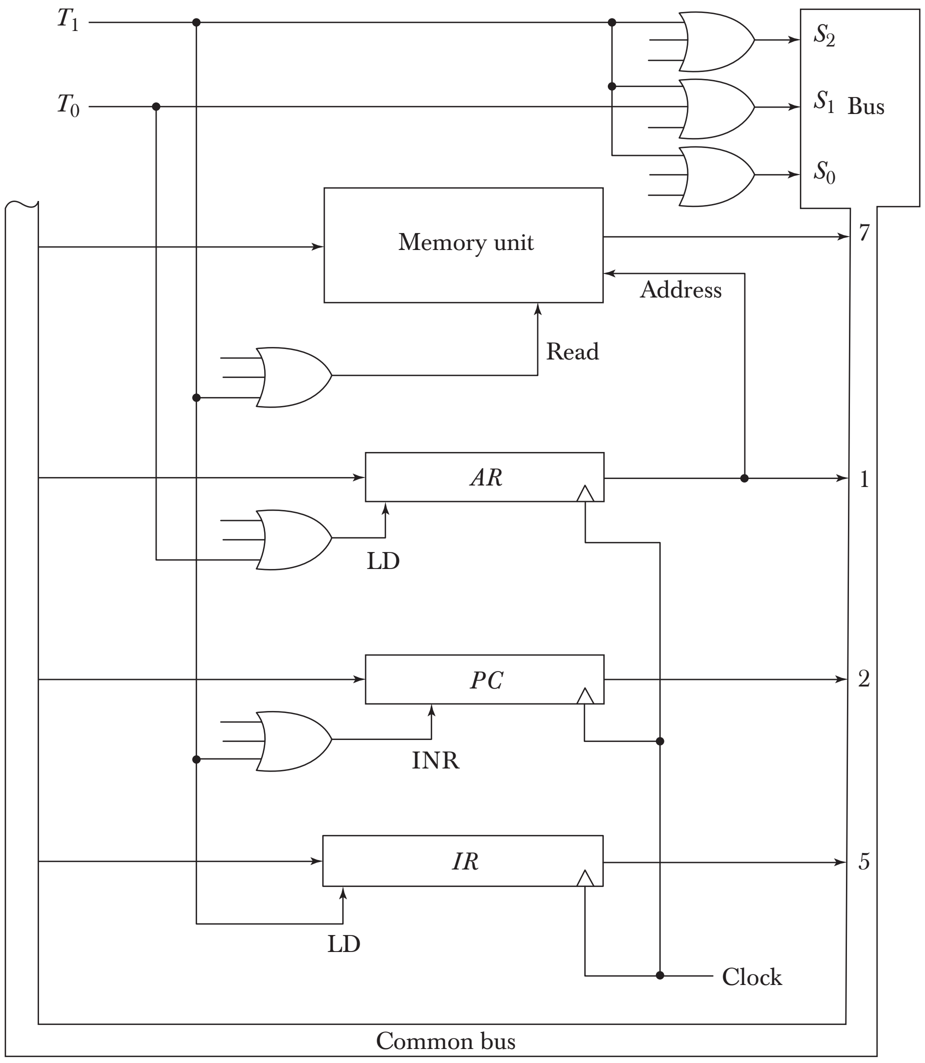
T_0: $AR \leftarrow PC$

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

T_2: $D_0, \ldots, D_7 \leftarrow$ Decode $IR(12\text{–}14),\ AR \leftarrow IR(0-11),\ I \leftarrow IR(15)$

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 T_0. The instruction read from memory is then placed in the instruction register *IR* with the clock transition associated with timing

Figure 5-8 Register transfers for the fetch phase.



signal T_1. At the same time, *PC* is incremented by one to prepare it for the address of the next instruction in the program. At rime T_2, 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 T_0, T_1, and T_2.

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 T_0 to achieve the following connection:

1. Place the content of *PC* onto the bus by making the bus selection inputs $S_2S_1S_0$ equal to 010.
2. Transfer the content of the bus to *AR* by enabling the LD input of *AR*.

The next clock transition initiates the transfer from *PC* to *AR* since $T_0 = 1$. 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*.

The next clock transition initiates the read and increment operations since $T_1 = 1$.

Figure 5-8 duplicates a portion of the bus system and shows how T_0 and T_1 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.

Determine the Type of Instruction

The timing signal that is active after the decoding is T_3. During time T_3, 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.

Decoder output D_7 is equal to 1 if the operation code is equal to binary 111. From Fig. 5-5 we determine that if $D_7 = 1$, the instruction must be a

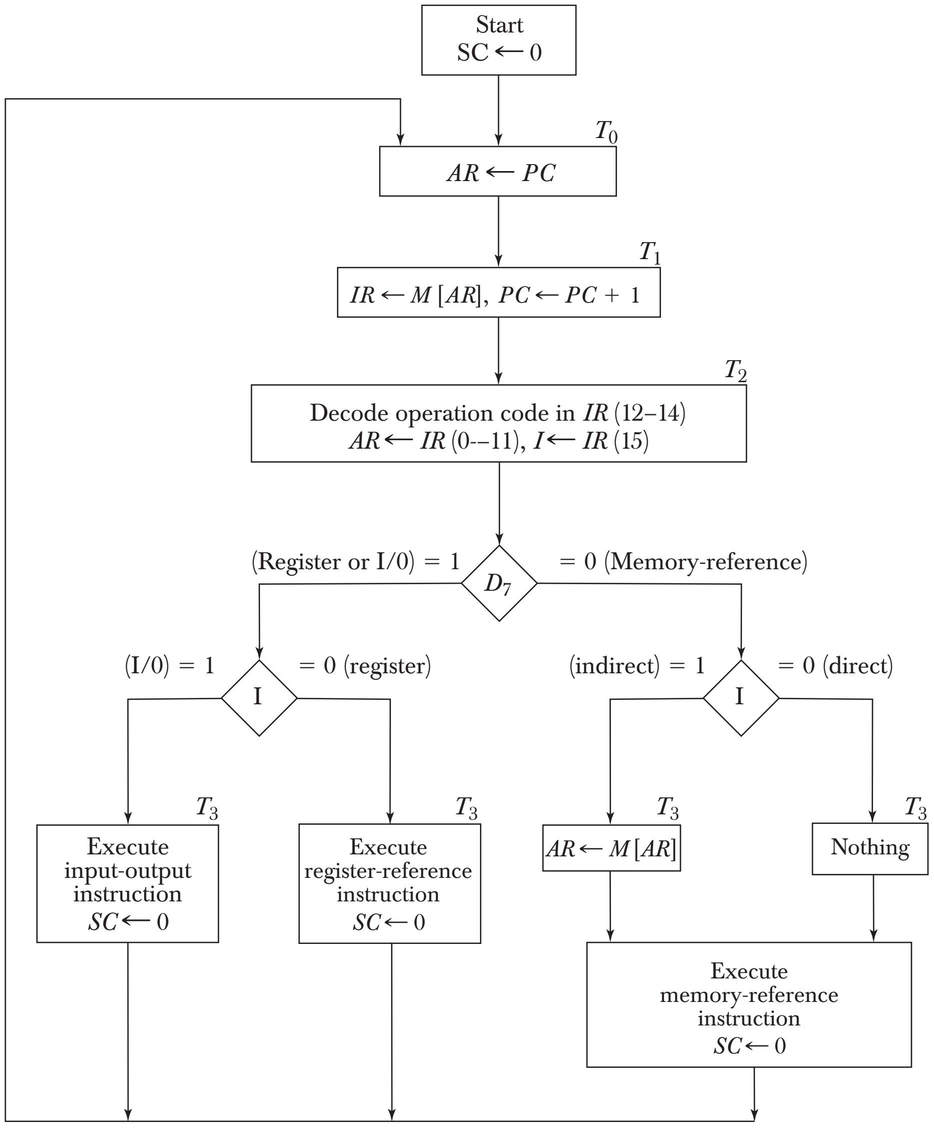

Start
SC ← 0
T_0
$AR \leftarrow PC$
T_1
$IR \leftarrow M[AR]$, $PC \leftarrow PC + 1$
T_2
Decode operation code in IR (12–14)
$AR \leftarrow IR$ (0--11), $I \leftarrow IR$ (15)
(Register or I/0) = 1
D_7
= 0 (Memory-reference)
(I/0) = 1
I
= 0 (register)
(indirect) = 1
I
= 0 (direct)
T_3
Execute input-output instruction $SC \leftarrow 0$
T_3
Execute register-reference instruction $SC \leftarrow 0$
T_3
$AR \leftarrow M[AR]$
T_3
Nothing
Execute memory-reference instruction $SC \leftarrow 0$

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

register-reference or input–output type. If $D_7 = 0$, 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 $D_7 = 0$ and $I = 1$, we have a memory-reference instruction with an indirect address. It is then necessary to read the *indirect address*

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.

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

When a memory-reference instruction with $I = 0$ is encountered, it is not necessary to do anything since the effective address is already in AR. However, the sequence counter SC must be incremented when $D_7' T_3 = 1$, so that the execution of the memory-reference instruction can be continued with timing variable T_4. A register-reference or input-output instruction can be executed with the clock associated with timing signal T_3. After the instruction is executed, SC is cleared to 0 and control returns to the fetch phase with $T_0 = 1$.

Note that the sequence counter SC is either incremented or cleared to 0 with every positive clock transition. We will adopt the convention that if SC is incremented, we will not write the statement $SC \leftarrow SC + 1$, but it will be implied that the control goes to the next timing signal in sequence. When SC is to be cleared, we will include the statement $SC \leftarrow 0$.

The register transfers needed for the execution of the register-reference instructions are presented in this section. The memory-reference instructions are explained in the next section. The input-output instructions are included in Sec. 5-7.

Register-Reference Instructions

Register-reference instructions are recognized by the control when $D_7 = 1$ 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 T_2.

The control functions and microoperations for the register-reference instructions are listed in Table 5-3. These instructions are executed with the