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.

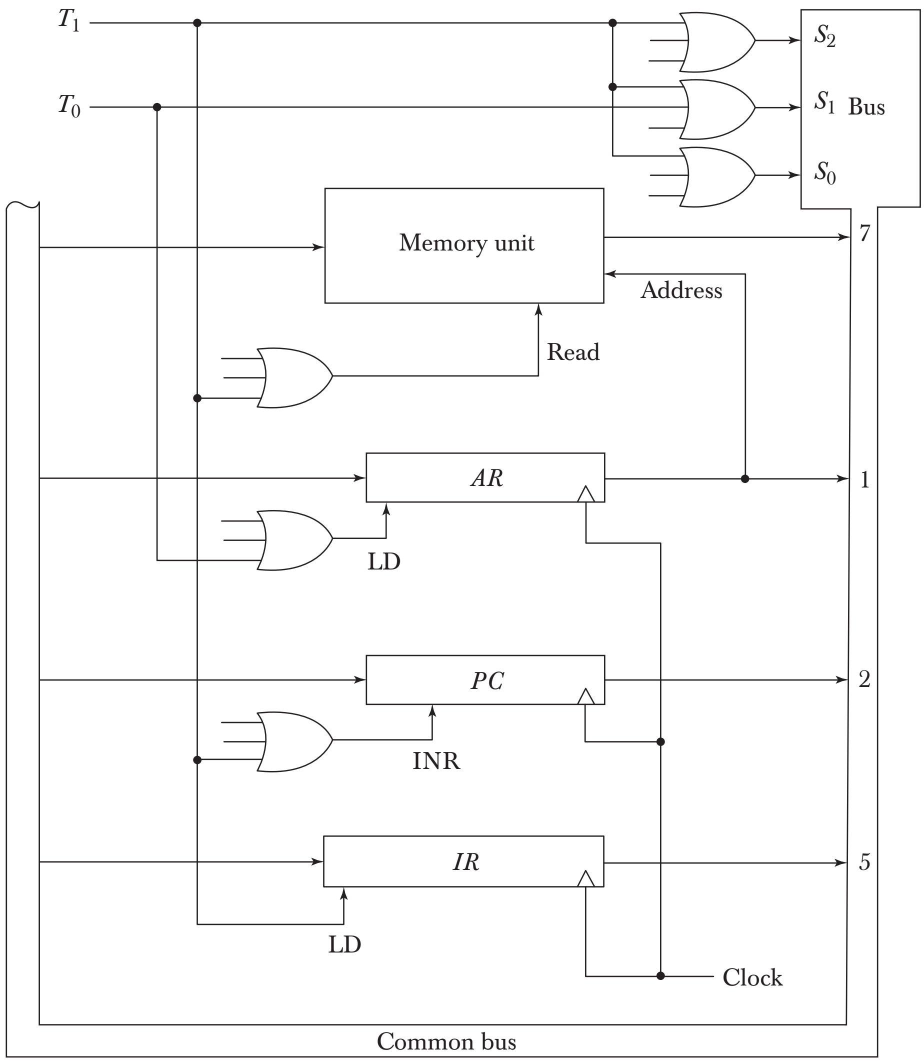
T_0: $AR \leftarrow PC$

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

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

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

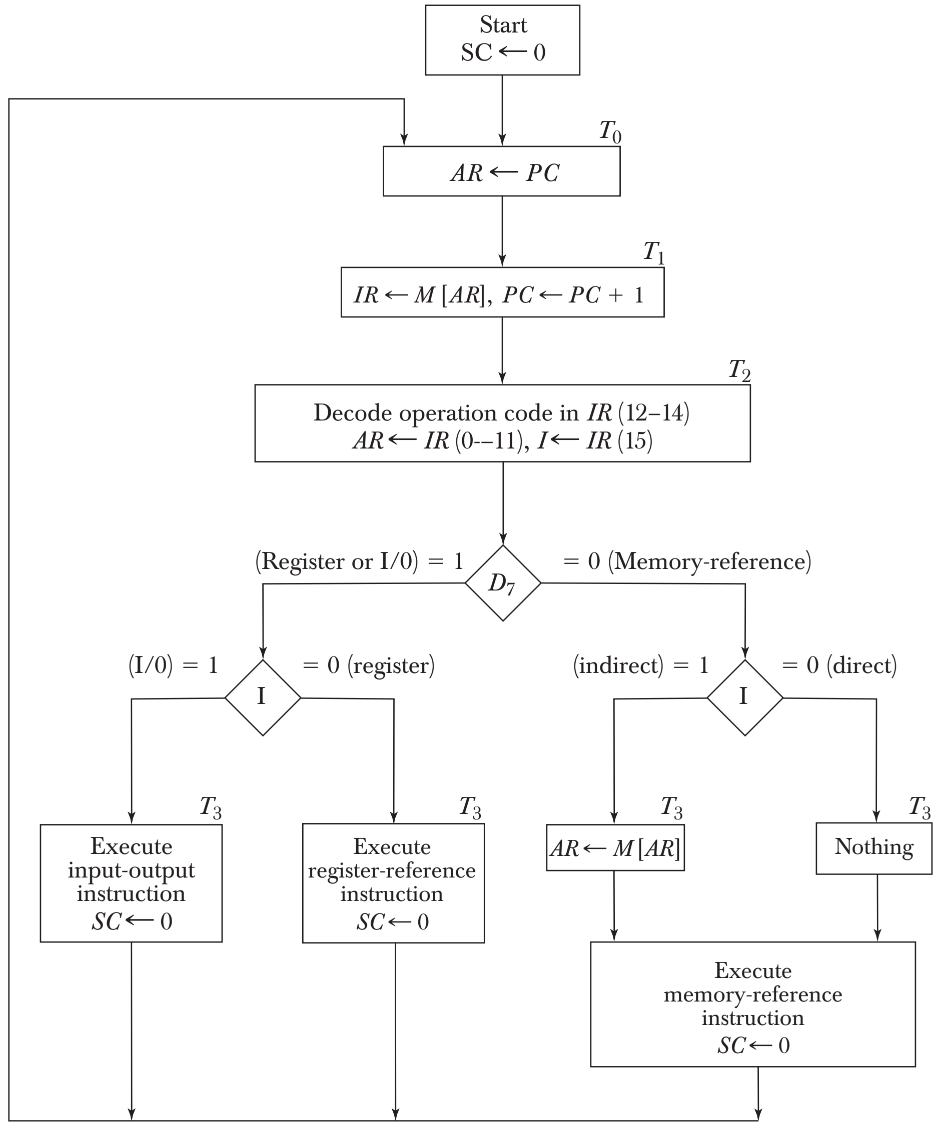


Figure 5-9 Flowchart for instruction cycle (initial configuration).

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 clock transition associated with timing variable T_3. Each control function needs the Boolean relation $D_7I'T_3$, which we designate for convenience by the symbol r. The control function is distinguished by one of the bits in $IR(0–11)$. By assigning the symbol B_i to bit i of IR, all control functions can be simply denoted by rB_i. For example, the instruction CLA has the hexadecimal code 7800 (see Table 5-2), which gives the binary equivalent 0111 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 D_7. Bit 11 in IR is 1 and is recognized from B_{11}. The control function that initiates the microoperation for this instruction is $D_7I'T_3B_{11} = rB_{11}$. The execution of a register-reference instruction is completed at time T_3. The sequence counter SC is cleared to 0 and the control goes back to fetch the next instruction with timing signal T_0.

The first seven register-reference instructions perform clear, complement, circular shift, and increment microoperations 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 T_1). The condition control statements must be recognized as part of the control conditions. The AC is positive when the sign bit in $AC(15) = 0$; it is negative when $AC(15) = 1$. 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.

TABLE 5-3 Execution of Register-Reference Instructions

$D_7I'T_3 = r$ (common to all registcr-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:	$AC^* \rightarrow AC + 1$	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

5-6 Memory-Reference Instructions

In order to specify the microoperations needed for the execution of each instruction, it is necessary that the function that they are intended to perform be defined precisely. Looking back to Table 5-2, where the instructions are listed, we find that some instructions have an ambiguous description. This is because the explanation of an instruction in words is usually lengthy, and not enough space is available in the table for such a lengthy explanation. We will now show that the function of the memory-reference instructions can be defined precisely by means of register transfer notation.

effective address

Table 5-4 lists the seven memory-reference instructions. The decoded output D_i 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 T_2 when $I = 0$, or during timing signal T_3 when $I = 1$. The execution of the memory-reference instructions starts with timing signal T_4. The symbolic description of each instruction is specified in the table in terms of register transfer notation. The actual execution of the instruction in the bus system will require a sequence of microoperations. 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 microoperations needed for their execution. A flowchart that summarizes all the microoperations is presented at the end of this section.

TABLE 5-4 Memory-Reference Instructions

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

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:

$$D_0T_4: \quad DR \leftarrow M[AR]$$

$$D_0T_5: \quad AC \leftarrow AC \wedge DR, \quad SC \leftarrow 0$$

The control function for this instruction uses the operation decoder D_0 since this output of the decoder is active when the instruction has an AND operation whose binary code value is 000. Two timing signals are needed to execute the instruction. The clock transition associated with timing signal T_4 transfers the operand from memory into DR. The clock transition associated with the next timing signal T_5 transfers to AC the result of the AND logic operation between the contents of DR and AC. The same clock transition clears SC to 0, transferring control to timing signal T_0 to start a new instruction cycle.

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 C_{out} is transferred to the E (extended accumulator) flip-flop. The microoperations needed to execute this instruction are

$$D_1T_4: \quad DR \leftarrow M[AR]$$

$$D_1T_5: \quad AC \leftarrow AC + DR, \quad E \leftarrow C_{out}, \quad SC \leftarrow 0$$

The same two timing signals, T_4 and T_5, are used again but with operation decoder D_1 instead of D_0, which was used for the AND instruction. After the instruction is fetched from memory and decoded, only one output of the operation decoder will be active, and that output determines the sequence of microoperations that the control follows during the execution of a memory-reference instruction.

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

$$D_2T_4: \quad DR \leftarrow M[AR]$$

$$D_2T_5: \quad AC \leftarrow DR, \quad SC \leftarrow 0$$

Looking back at the bus system shown in Fig. 5-4 we note that there is no direct path from the bus into AC. The adder and logic circuit receive information