transferring the content of *DR* through the adder and logic circuit into *AC*, and enabling the LD (load) input of *AC*, all during the same clock cycle. The two transfers occur upon the arrival of the clock pulse transition at the end of the clock cycle.

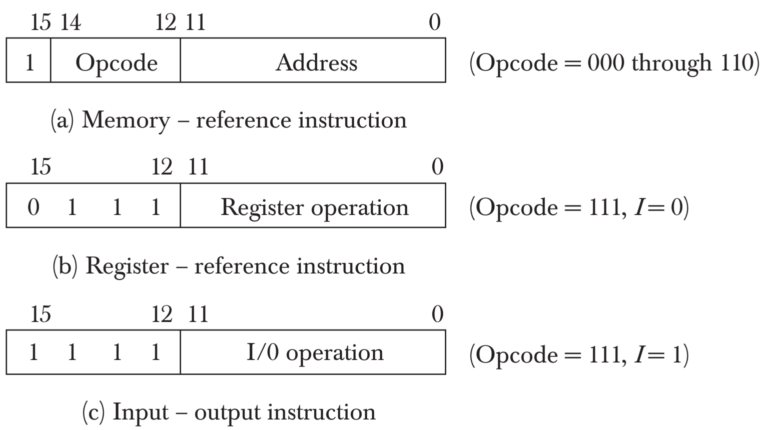
5-3 Computer Instructions

Instruction format

The basic computer has three instruction code formats, as shown in Fig. 5-5. Each format has 16 bits. The operation code (opcode) part of the instruction contains three bits and the meaning of the remaining 13 bits depends on the operation code encountered. A memory-reference instruction uses 12 bits to specify an address and one bit to specify the addressing mode *I*. *I* is equal to 0 for direct address and to 1 for indirect address (see Fig. 5-2). The register-reference instructions are recognized by the operation code 111 with a 0 in the leftmost bit (bit 15) of the instruction. A register-reference instruction specifies an operation on or a test of the *AC* register. An operand from memory is not needed; therefore, the other 12 bits are used to specify the operation or test to be executed. Similarly, an input–output instruction does not need a reference to memory and is recognized by the operation code 111 with a 1 in the leftmost bit of the instruction. The remaining 12 bits are used to specify the type of input–output operation or test performed.

The type of instruction is recognized by the computer control from the four bits in positions 12 through 15 of the instruction. If the three opcode bits in positions 12 though 14 are not equal to 111, the instruction is a memory-reference type and the bit in position 15 is taken as the addressing mode *I*. If the 3-bit opcode is equal to 111, control then inspects the bit in position 15. If

Figure 5-5 Basic computer instruction formats.



this bit is 0, the instruction is a register-reference type. If the bit is 1, the instruction is an input–output type. Note that the bit in position 15 of the instruction code is designated by the symbol *I* but is not used as a mode bit when the operation code is equal to 111.

Only three bits of the instruction are used for the operation code. It may seem that the computer is restricted to a maximum of eight distinct operations. However, since register-reference and input–output instructions use the remaining 12 bits as part of the operation code, the total number of instructions can exceed eight. In fact, the total number of instructions chosen for the basic computer is equal to 25.

The instructions for the computer are listed in Table 5-2. The symbol designation is a three-letter word and represents an abbreviation intended for

TABLE 5-2 Basic Computer Instructions

	Hexadecimal code		
Symbol	$I = 0$	$I = 1$	Description
AND	0xxx	8xxx	AND memory word to *AC*
ADD	1xxx	9xxx	Add memory word to *AC*
LDA	2xxx	Axxx	Load memory word to *AC*
STA	3xxx	Bxxx	Store content of *AC* in memory
BUN	4xxx	Cxxx	Branch unconditionally
BSA	5xxx	Dxxx	Branch and save return address
ISZ	6xxx	Exxx	Increment and skip if zero
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 instruction if *AC* positive
SNA	7008		Skip next instruction if *AC* negative
SZA	7004		Skip next instruction if *AC* zero
SZE	7002		Skip next instruction if *E* is 0
HLT	7001		Halt computer
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

hexadecimal code

programmers and users. The hexadecimal code is equal to the equivalent hexadecimal number of the binary code used for the instruction. By using the hexadecimal equivalent we reduced the 16 bits of an instruction code to four digits with each hexadecimal digit being equivalent to four bits. A memory-reference instruction has an address part of 12 bits. The address part is denoted by three x's and stand for the three hexadecimal digits corresponding to the 12-bit address. The last bit of the instruction is designated by the symbol I. When $I = 0$, the last four bits of an instruction have a hexadecimal digit equivalent from 0 to 6 since the last bit is 0. When $I = 1$, the hexadecimal digit equivalent of the last four bits of the instruction ranges from 8 to E since the last bit is 1.

Register-reference instructions use 16 bits to specify an operation. The leftmost four bits are always 0111, which is equivalent to hexadecimal 7. The other three hexadecimal digits give the binar) equivalent of the remaining 12 bits. The input–output instructions also use all 16 bits to specify an operation. The last four bits are always 1111, equivalent to hexadecimal F.

Instruction Set Completeness

Before investigating the operations performed by the instructions, let us discuss the type of instructions that must be included in a computer. A computer should have a set of instructions so that the user can construct machine language programs to evaluate any function that is known to be computable. The set of instructions are said to be complete if the computer includes a sufficient number of instructions in each of the following categories:

1. Arithmetic, logical, and shift instructions
2. Instructions for moving information to and from memory and processor registers
3. Program control instructions together with instructions that check status conditions
4. Input and output instructions

Arithmetic, logical, and shift instructions provide computational capabilities for processing the type of data that the user may wish to employ. The bulk of the binary information in a digital computer is stored in memory, but all computations are done in processor registers. Therefore, the user must have the capability of moving information between these two units. Decision-making capabilities are an important aspect of digital computers. For example, two numbers can be compared, and if the first is greater than the second, it may be necessary to proceed differently than if the second is greater than the first. Program control instructions such as branch instructions are used to change the sequence in which the program is executed. Input and output instructions are needed for communication between the computer and the