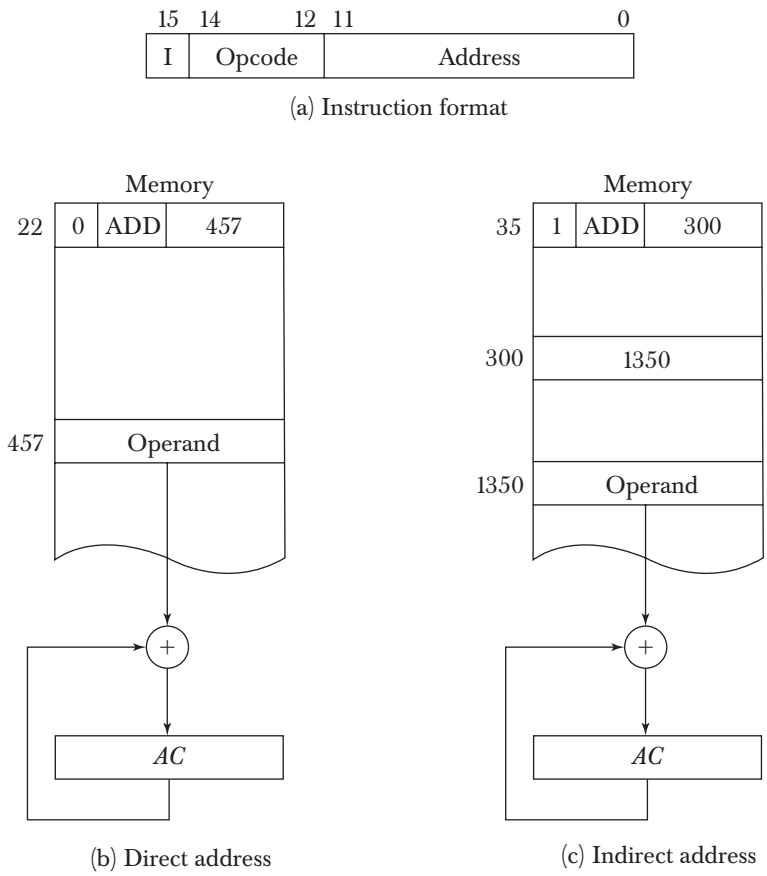


Figure 5-2 Demonstration of direct and indirect address.

data. The pointer could be placed in a processor register instead of memory as done in commercial computers.

5-2 Computer Registers

Computer instructions are normally stored in consecutive memory locations and are executed sequentially one at a time. The control reads an instruction from a specific address in memory and executes it. It then continues by reading the next instruction in sequence and executes it, and so on. This type of instruction sequencing needs a counter to calculate the address of the next instruction after execution of the current instruction is completed. It is also necessary to provide a register in the control unit for storing the instruction code after it is read

from memory. The computer needs processor registers for manipulating data and a register for holding a memory address. These requirements dictate the register configuration shown in Fig. 5-3. The registers are also listed in Table 5-1 together with a brief description of their function and the number of bits that they contain.

The memory unit has a capacity of 4096 words and each word contains 16 bits. Twelve bits of an instruction word are needed to specify the address of an operand. This leaves three bits for the operation, part of the instruction and a bit to specify a direct or indirect address. The data register (*DR*) holds the operand read from memory. The accumulator (*AC*) register is a general-purpose processing register. The instruction read from memory is placed in the instruction register (*IR*). The temporary register (*TR*) is used for holding temporary data during the processing.

TABLE 5-1 List of Registers for the Basic Computer

Register symbol	Number of bits	Register name	Function
DR	16	Data register	Holds memory operand
AR	12	Address register	Holds address for memory
AC	16	Accumulator	Processor register
IR	16	Instruction register	Holds instruction code
PC	12	Program counter	Holds address of instruction
TR	16	Temporary register	Holds temporary data
INPR	8	Input register	Holds input character
OUTR	8	Output register	Holds output character

program counter (PC)

The memory address register (*AR*) has 12 bits since this is the width of a memory address. The program counter (*PC*) also has 12 bits and it holds the address of the next instruction to be read from memory after the current instruction is executed. The *PC* goes through a counting sequence and causes the computer to read sequential instructions previously stored in memory. Instruction words are read and executed in sequence unless a branch instruction is encountered. A branch instruction calls for a transfer to a nonconsecutive instruction in the program. The address part of a branch instruction is transferred to *PC* to become the address of the next instruction. To read an instruction, the content of *PC* is taken as the address for memory and a memory read cycle is initiated. *PC* is then incremented by one, so it holds the address of the next instruction in sequence.

Two registers are used for input and output. The input register (*INPR*) receives an 8-bit character from an input device. The output register (*OUTR*) holds an 8-bit character for an output device.

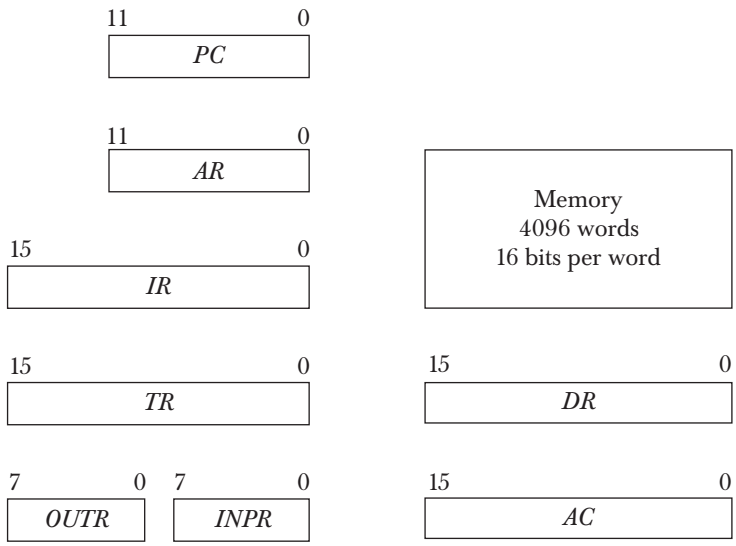


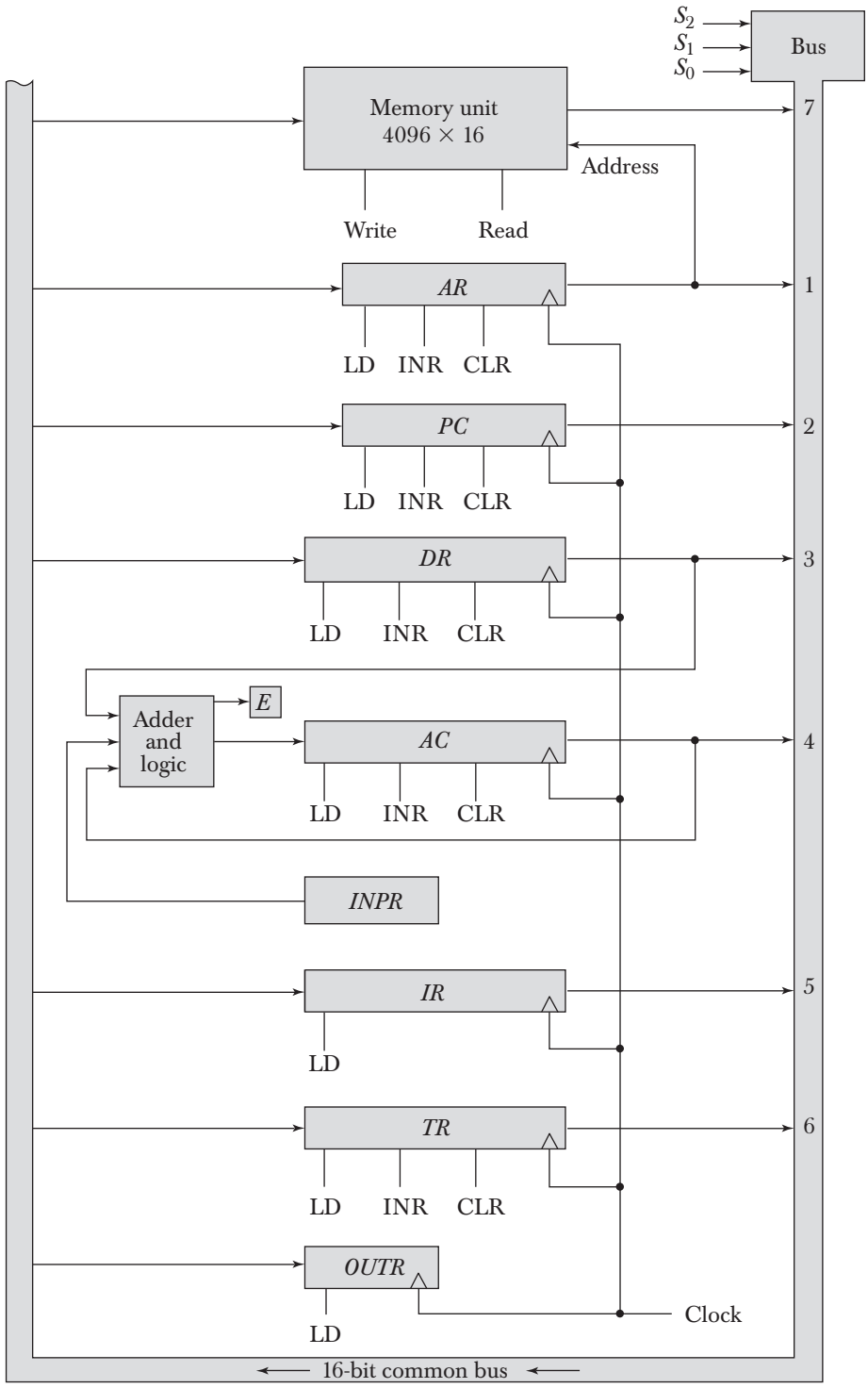
Figure 5-3 Basic computer registers and memory.

Common Bus System

The basic computer has eight registers, a memory unit, and a control unit (to be presented in Sec. 5-4). Paths must be provided to transfer information from one register to another and between memory and registers. The number of wires will be excessive if connections are made between the outputs of each register and the inputs of the other registers. A more efficient scheme for transferring information in a system with many registers is to use a common bus. We have shown in Sec. 4-3 how to construct a bus system using multiplexers or three-state buffer gates. The connection of the registers and memory of the basic computer to a common bus system is shown in Fig. 5-4.

The outputs of seven registers and memory are connected to the common bus. The specific output that is selected for the bus lines at any given time is determined from the binary value of the selection variables S_2, S_1, and S_0. The number along each output shows the decimal equivalent of the required binary selection. For example, the number along the output of *DR* is 3. The 16-bit outputs of *DR* are placed on the bus lines when $S_2S_1S_0 = 011$ since this is the binary value of decimal 3. The lines from the common bus are connected to the inputs of each register and the data inputs of the memory. The particular register whose LD (load) input is enabled receives the data from the bus during the next clock pulse transition. The memory receives the contents of the bus when its write input is activated. The memory places its 16-bit output onto the bus when the read input is activated and $S_2S_1S_0 = 111$.

load (LD)


S_2
S_1
S_0
Bus
Memory unit
4096 × 16
Address
Write
Read
7
AR
LD INR CLR
1
PC
LD INR CLR
2
DR
LD INR CLR
3
Adder and logic
E
AC
LD INR CLR
4
INPR
IR
LD
5
TR
LD INR CLR
6
OUTR
LD
Clock
16-bit common bus

Four registers, *DR*, *AC*, *IR*, and *TR*, have 16 bits each. Two registers, *AR* and *PC*, have 12 bits each since they hold a memory address. When the contents of *AR* or *PC* are applied to the 16-bit common bus, the four most significant bits are set to 0's. When *AR* or *PC* receive information from the bus, only the 12 least significant bits are transferred into the register.

The input register *INPR* and the output register *OUTR* have 8 bits each and communicate with the eight least significant bits in the bus. *INPR* is connected to provide information to the bus but *OUTR* can only receive information from the bus. This is because *INPR* receives a character from an input device which is then transferred to *AC*. *OUTR* receives a character from *AC* and delivers it to an output device. There is no transfer from *OUTR* to any of the other registers.

The 16 lines of the common bus receive information from six registers and the memory unit. The bus lines are connected to the inputs of six registers and the memory. Five registers have three control inputs: LD (load), INR (increment), and CLR (clear). This type of register is equivalent to a binary counter with parallel load and synchronous clear similar to the one shown in Fig. 2-11. The increment operation is achieved by enabling the count input of the counter. Two registers have only a LD input. This type of register is shown in Fig. 2-7.

memory address

The input data and output data of the memory are connected to the common bus, but the memory address is connected to *AR*. Therefore, *AR* must always be used to specify a memory address. By using a single register for the address, we eliminate the need for an address bus that would have been needed otherwise. The content of any register can be specified for the memory data input during a write operation. Similarly, any register can receive the data from memory after a read operation except *AC*.

The 16 inputs of *AC* come from an adder and logic circuit. This circuit has three sets of inputs. One set of 16-bit inputs come from the outputs of *AC*. They are used to implement register microoperations such as complement *AC* and shift *AC*. Another set of 16-bit inputs come from the data regisler *DR*. The inputs from *DR* and *AC* are used for arithmetic and logic microoperations, such as add *DR* to *AC* or AND *DR* to *AC*. The result of an addition is transferred to *AC* and the end carry-out of the addition is transferred to flip-flop *E* (extended *AC* bit). A third set of 8-bit inputs come from the input register *INPR*. The operation of *INPR* and *OUTR* is explained in Sec. 5-7.

Note that the content of any register can be applied onto the bus and an operation can be performed in the adder and logic circuit during the same clock cycle. The clock transition at the end of the cycle transfers the content of the bus into the designated destination register and the output of the adder and logic circuit into *AC*. For example, the two microoperations

$$DR \leftarrow AC \quad \text{and} \quad AC \leftarrow DR$$

can be executed at the same time. This can be done by placing the content of *AC* on the bus (with $S_2S_1S_0 = 100$), enabling the LD (load) input of *DR*,