Computer Instructions

- Computer instructions are a set of machine language instructions that a particular processor understands and executes. A computer performs tasks on the basis of the instruction provided.

- An instruction comprises of groups called fields. These fields include:
- The Operation code (Opcode) field which specifies the operation to be performed.
- The Address field which contains the location of the operand, i.e., register or memory location.
- The Mode field which specifies how the operand will be located.

A basic computer has three instruction code formats which are:

- Memory - reference instruction
- Register - reference instruction
- Input-Output instruction

Figure 5-5 Basic computer instruction formats.

(Opcode = 000 through 110)

(a) Memory – reference instruction

(Opcode = 111, $I = 0$)

(b) Register – reference instruction

(Opcode = 111, $I = 1$)

(c) Input – output instruction

Memory - reference instruction

Register - reference instruction

Input-Output instruction

- Just like the Register-reference instruction, 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 the input-output operation or test performed.

- Note
- The three operation code bits in positions 12 through 14 should be equal to 111. Otherwise, the instruction is a memory-reference type, and the bit in position 15 is taken as the addressing mode I.
- When the three operation code bits are equal to 111, control unit inspects the bit in position 15. If the bit is 0, the instruction is a register-reference type. Otherwise, the instruction is an input-output type having bit 1 at position 15.

TABLE 5-2 Basic Computer Instructions

Symbol	Hexadecimal code $I = 0$	Hexadecimal code $I = 1$	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

Instruction Set Completeness

- A set of instructions is said to be complete if the computer includes a sufficient number of instructions in each of the following categories:
- Arithmetic, logical and shift instructions
- A set of instructions for moving information to and from memory and processor registers.
- Instructions which controls the program together with instructions that check status conditions.
- Input and Output instructions

- Arithmetic, logic and shift instructions provide computational capabilities for processing the type of data the user may wish to employ.

- A huge amount of binary information is stored in the memory unit, but all computations are done in processor registers. Therefore, one must possess the capability of moving information between these two units.

- Program control instructions such as branch instructions are used change the sequence in which the program is executed.

- Input and Output instructions act as an interface between the computer and the user. Programs and data must be transferred into memory, and the results of computations must be transferred back to the user.

- A Computer uses a memory unit with 256 K words of 32 bit each. A binary instruction code is stored in one of the memory. The instruction has four parts : an indirect bit, an operation code, a register code to specify one of the 64 registers and an address part.
- A) how many bits are there in the operation code, the register code part and the address code?
- B) Draw the instruction code format and indicate no of bits in each part.
- C) How many bits are there in the data and address inputs of the memory

$256\ K = 2^8 \times 2^{10} = 2^{18}$

$64 = 2^6$

(a) Address: 18 bits

Register code: 6 bits

Indirect bit: 1 bit

25 32 – 25 = 7 bits for opcode.

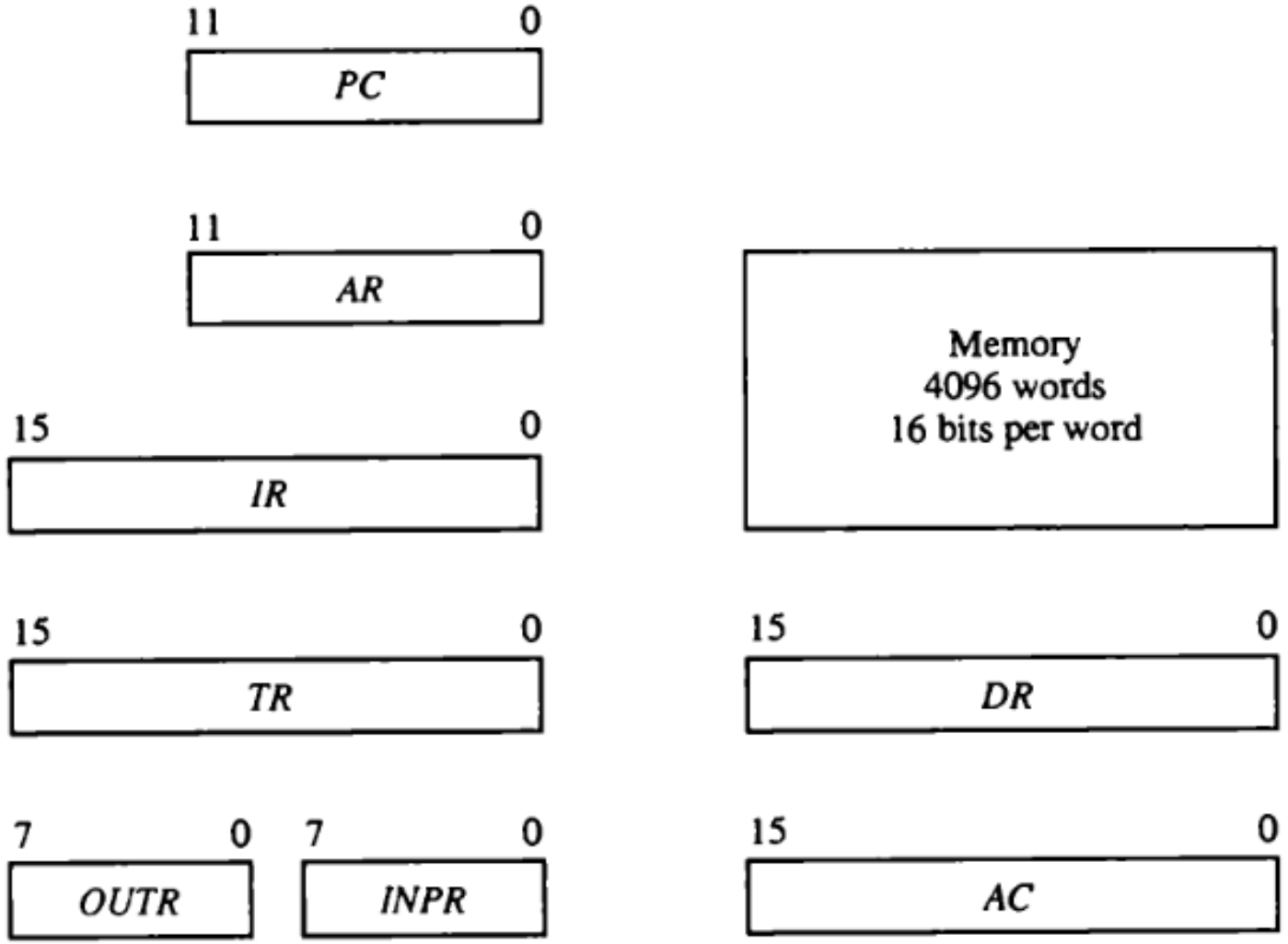
(b) 1 7 6 18 = 32 bits

(c) Data; 32 bits; address: 18 bits.

Basic Registers

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

Basic Computer Registers



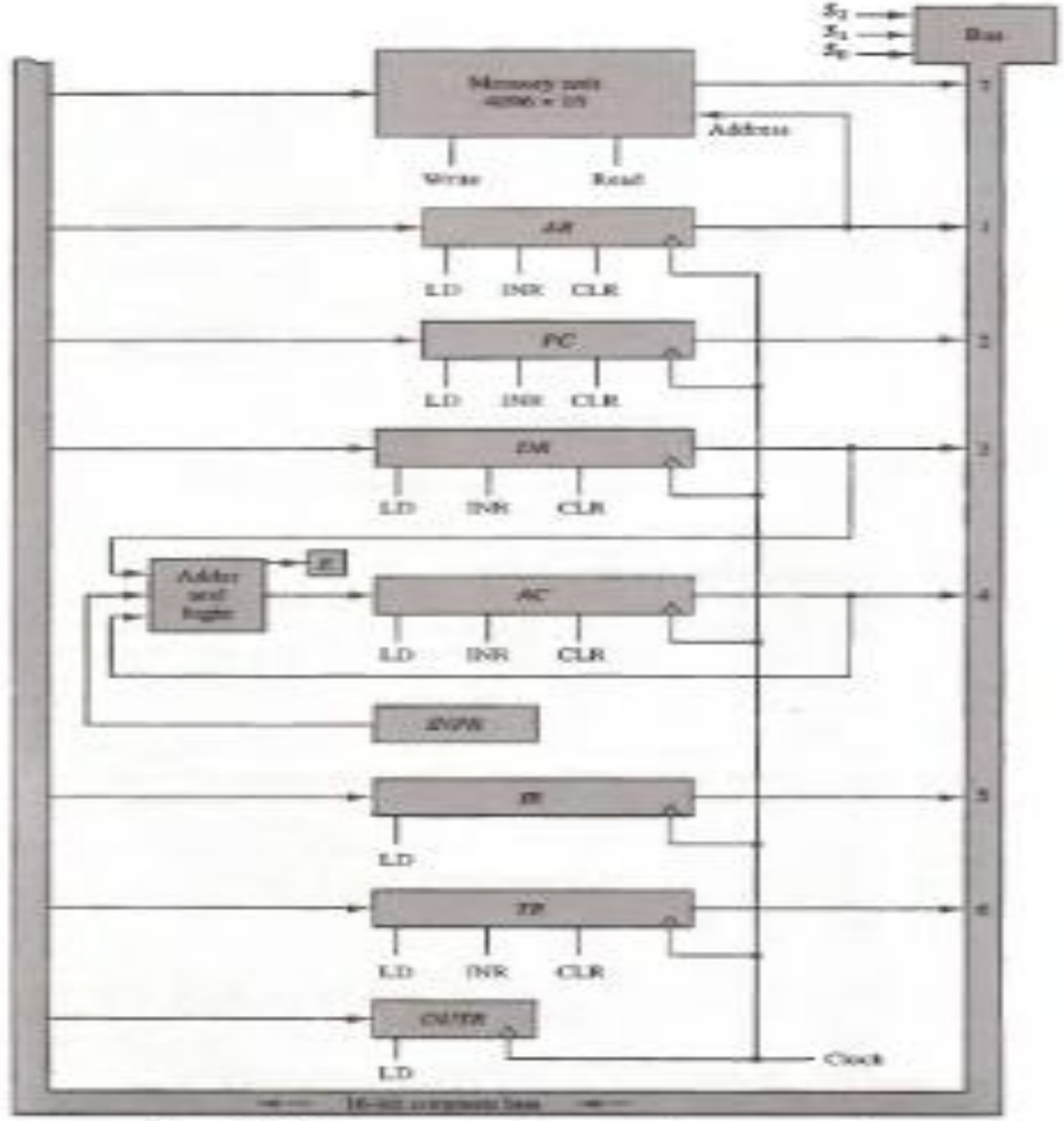


Figure 5-4 Basic computer registers connected to a common bus.

The following control inputs are active in the bus system shown in Fig. 5-4. For each case, specify the register transfer that will be executed during the next clock transition.

	S_2	S_1	S_0	LD of register	Memory	Adder
a.	1	1	1	*IR*	Read	—
b.	1	1	0	*PC*	—	—
c.	1	0	0	*DR*	Write	—
d.	0	0	0	*AC*	—	Add

5.3

(a) Memory read to bus and load to IR: IR ← M[AR]

(b) TR to bus and load to PC: PC ← TR

(c) AC to bus, write to memory, and load to DR:
DR ← AC, M[AR]← AC

(d) Add DR (or INPR) to AC: AC ← AC + DR

The following register transfers are to be executed in the system of Fig. 5-4. For each transfer, specify: (1) the binary value that must be applied to bus select inputs S_2, S_1, and S_0; (2) the register whose LD control input must be active (if any); (3) a memory read or write operation (if needed); and (4) the operation in the adder and logic circuit (if any).

a. $AR \leftarrow PC$
b. $IR \leftarrow M[AR]$
c. $M[AR] \leftarrow TR$
d. $AC \leftarrow DR$, $DR \leftarrow AC$ (done simultaneously)

		(1) $S_2S_1S_0$	(2) Load(LD)	(3) Memory	(4) Adder
(a)	AR ← PC	010 (PC)	AR	—	—
(b)	IR ← M[AR]	111 (M)	IR	Read	—
(c)	M[AR] ← TR	110 (TR)	—	Write	—
(d)	DR ← AC AC ← DR	100 (AC)	DR and AC	—	Transfer DR to AC