COMPUTER ORGANIZATION MODULE-II (Instruction Code)

Instruction Code

- ✓ A computer instruction is a binary code that specifies a sequence of microoperations for the computer.
- ✓ Instruction codes together with data are stored in memory.
- ✓ An instruction code is a group of bits that instruct the computer to perform a specific operation.
- ✓ The most basic part of an instruction code is its operation part.
- ✓ The operation code must consist of at least n bits for a given 2^n (or less) distinct operations.
- ✓ An instruction code must therefore specify not only the operation but also the registers or the memory words where the operands are to be found, as well as the register or memory word where the result is to be stored.

Stored Program Organization:

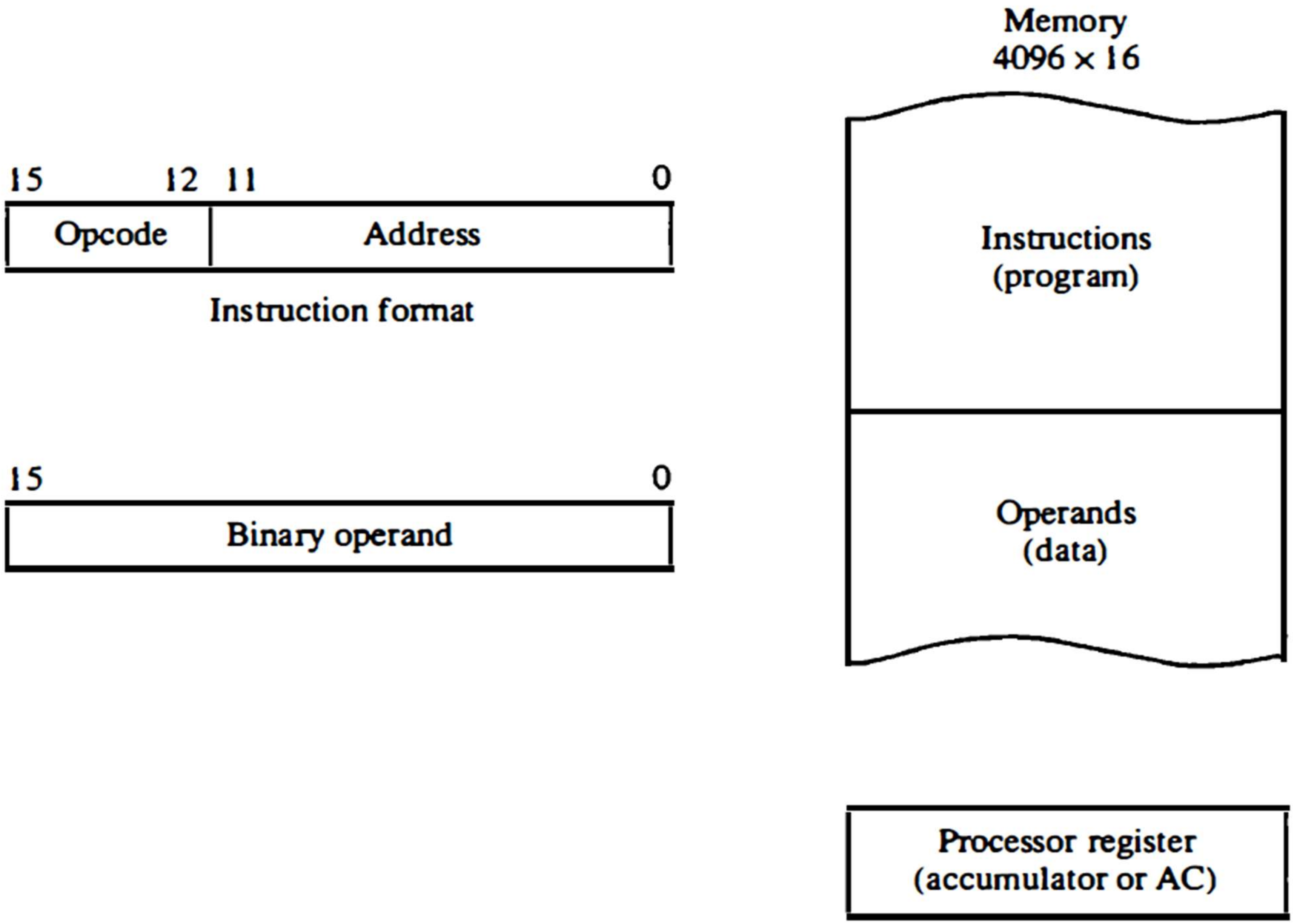
- ✓ The simplest way to organize a computer is to have one processor register and an instruction code format with two parts.

- ✓ Instructions are stored in one section of memory and data in another.

- ✓ For a memory unit with 4096 words, we need 12 bits to specify an address since $2^{12} = 4096$. If we store each instruction code in one 16-bit memory word, we have available four bits for the operation code (abbreviated opcode) to specify one out of 16 possible operations, and 12 bits to specify the address of an operand.

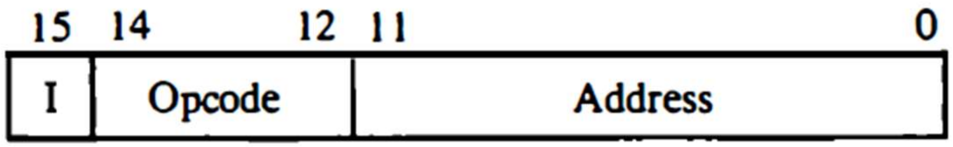
- ✓ The control reads a 16-bit instruction from the program portion of memory. It uses the 12-bit address part of the instruction to read a 16-bit operand from the data portion of memory. It then executes the operation specified by the operation code.

Stored Program Organization:



Indirect Address:

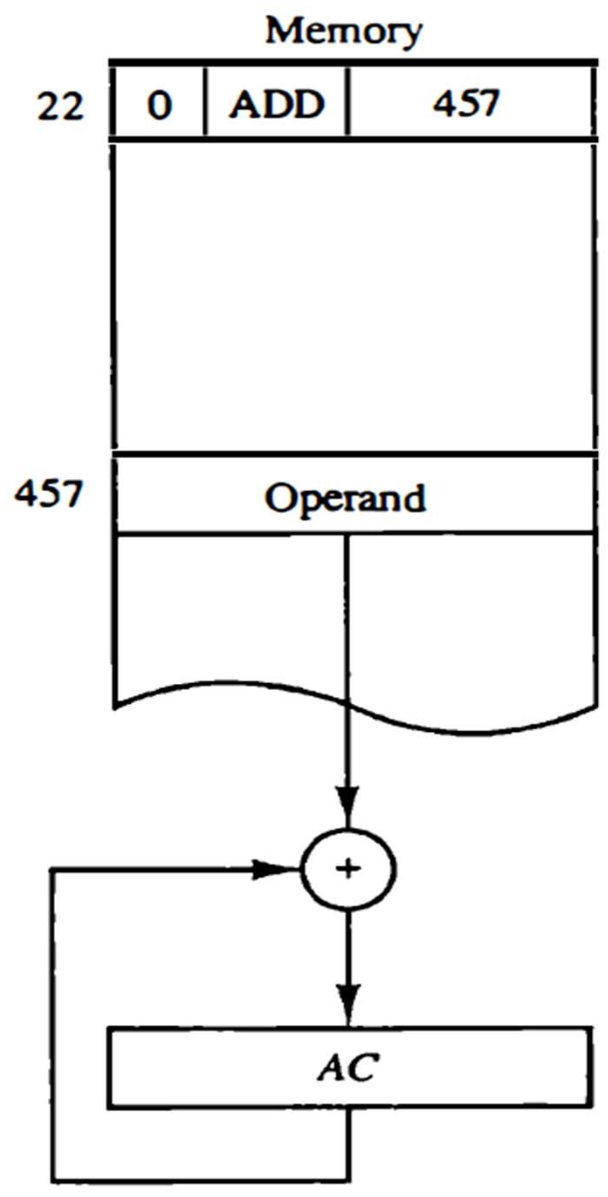
- ✓ In indirect address, the bits in the second part of the instruction designate an address of a memory word in which the address of the operand is found. One bit of the instruction code can be used to distinguish between a direct and an indirect address.
- ✓ Below instruction code format consists of a 3-bit operation code, a 12-bit address, and an indirect address mode bit designated by I. The mode bit is 0 for a direct address and 1 for an indirect address.



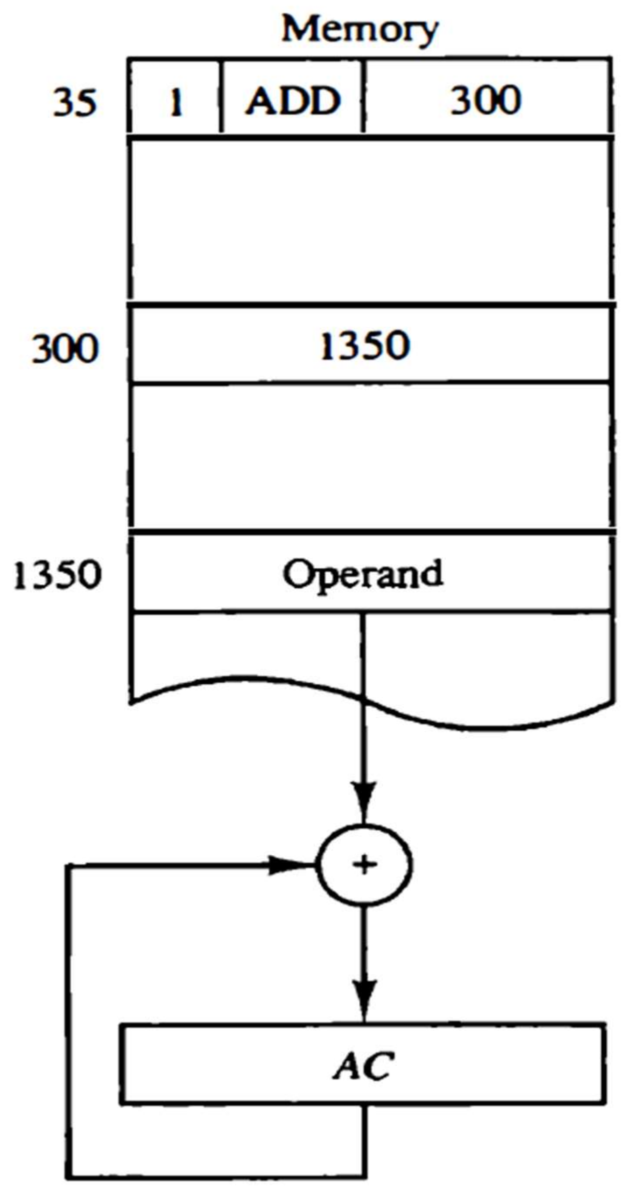
(a) Instruction format

- A direct address instruction as shown in next figure is placed in address 22 in memory. The I bit is 0, so the instruction is recognized as a direct address instruction. The opcode specifies an ADD instruction, and the address part is the binary equivalent of 457. The control finds the operand in memory at address 457 and adds it to the content of AC.

Indirect Address :



(b) Direct address



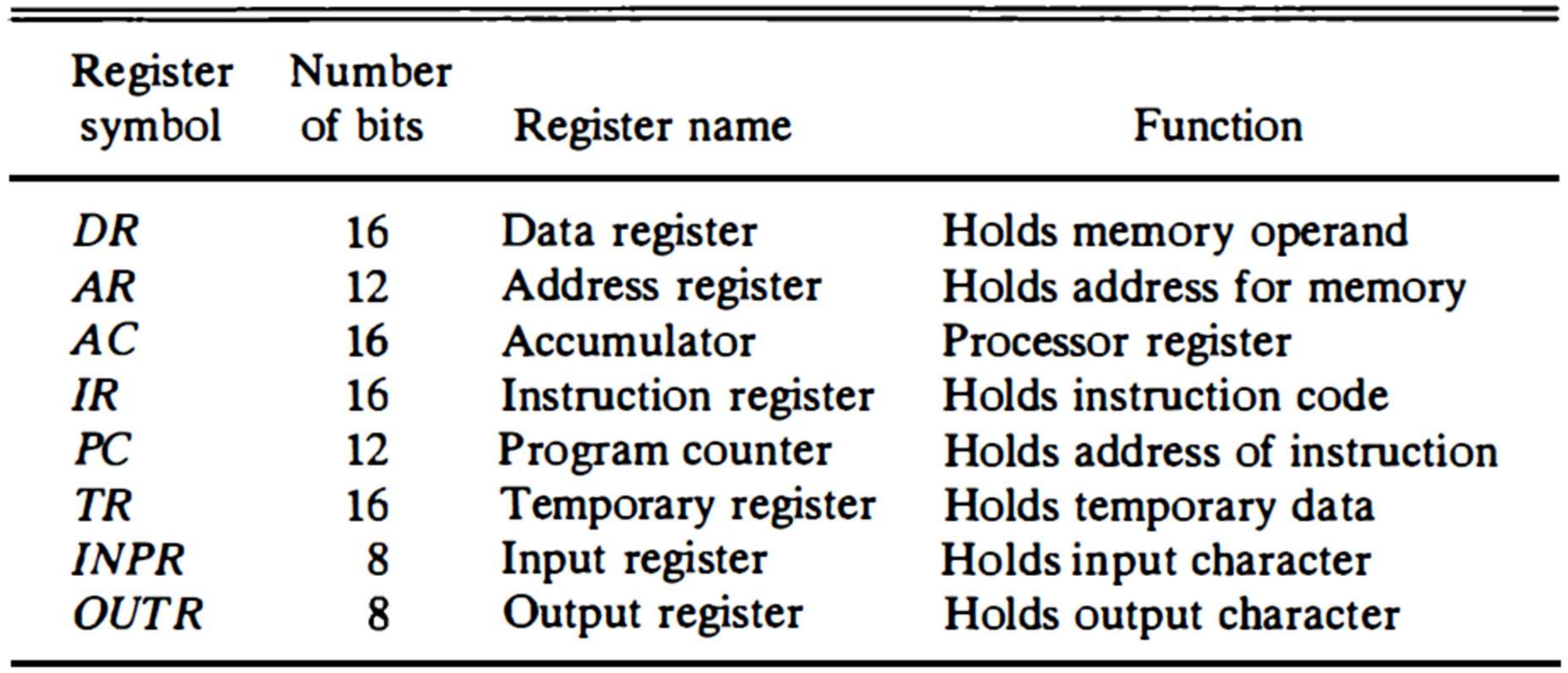
(c) Indirect address

Indirect Address:

- The instruction in address 35 has a mode bit I = 1. Therefore, it is recognized as an indirect address instruction. The address part is the binary equivalent of 300.
- The control goes to address 300 to find the address of the operand. The address of the operand in this case is 1350. The operand found in address 1350 is then added to the content of AC .
- The indirect address instruction needs two references to memory to fetch an operand. The first reference is needed to read the address of the operand; the second is for the operand itself. We define the effective address to be the address of the operand in a computation-type instruction or the target address in a branch-type instruction. Thus the effective address in direct and indirect address instruction is 457 and 1350 respectively.
- The memory word that holds the address of the operand in an indirect address instruction is used as a pointer to an array of data. The pointer could be placed in a processor register instead of memory as done in commercial computers.

Computer Registers:

- It is 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.

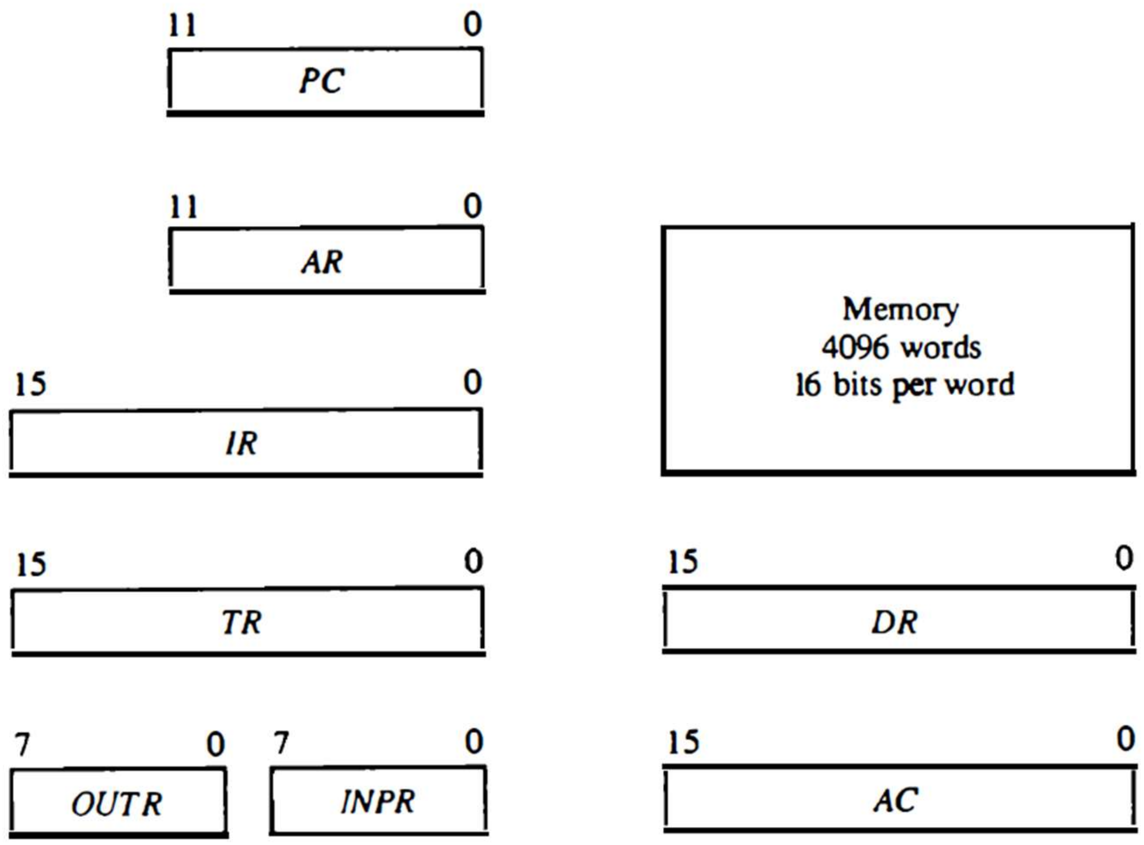


Register symbol	Number of bits	Register name	Function
DR	16	Data register	Holds memory operand
AR	12	Address register	Holds address for memory
AC	16	Accumulator	Processor register
IR	16	Instruction register	Holds instruction code
PC	12	Program counter	Holds address of instruction
TR	16	Temporary register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

- 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.

Computer Registers:

- 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.

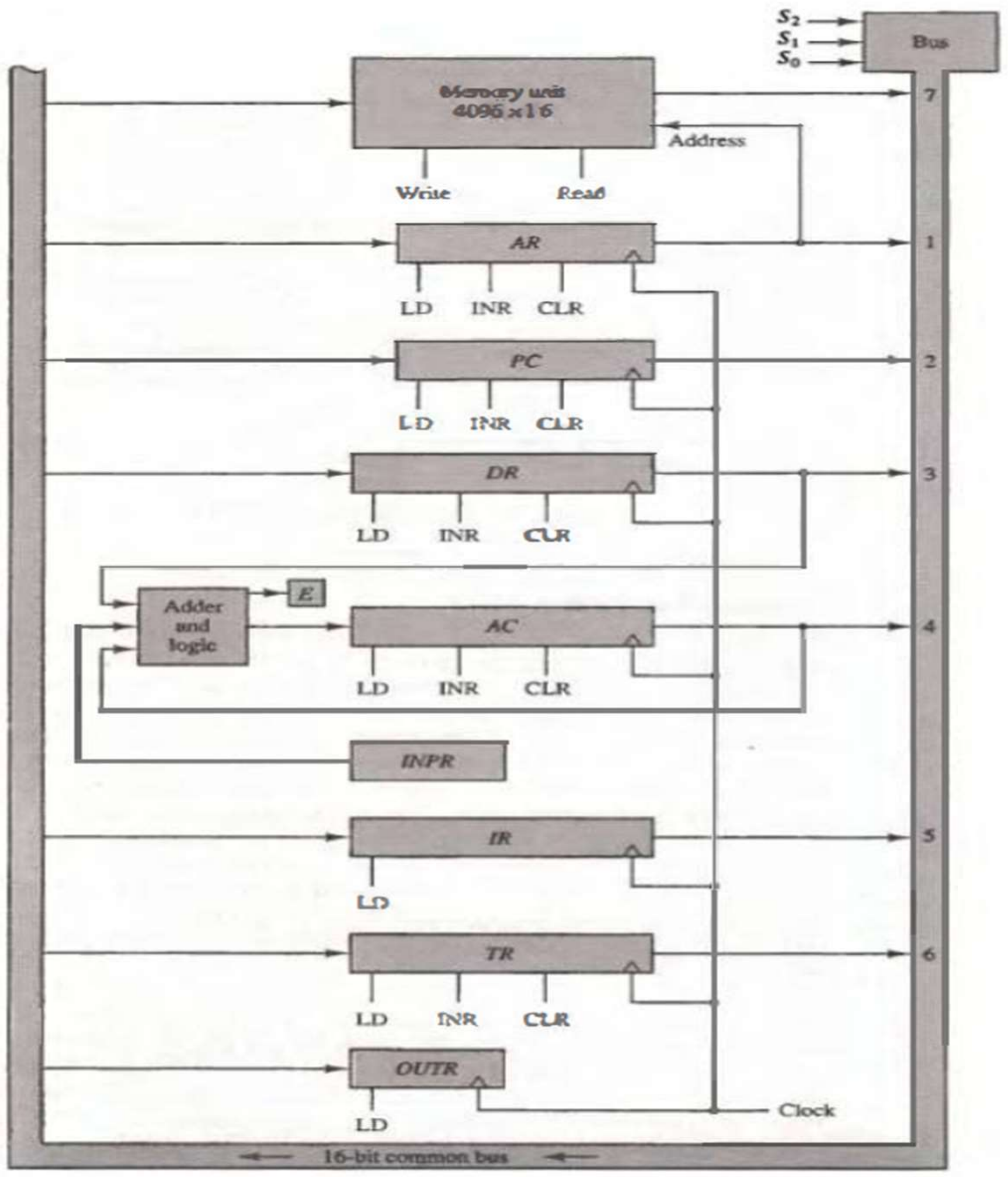


Computer Registers:

- 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.
- 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.

Common Bus System:

- A more efficient scheme for transferring information in a system with many registers is to use a common bus.



Common Bus System:

- 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 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 $S_2S_1S_0 = 111$.
- 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 MSB bits are set to 0's. When AR or PC receive information from the bus, only the 12 LSB bits are transferred into the register.

Common Bus System:

- 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).
- 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.

Common Bus System:

- 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 register 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.
- 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.

Common Bus System:

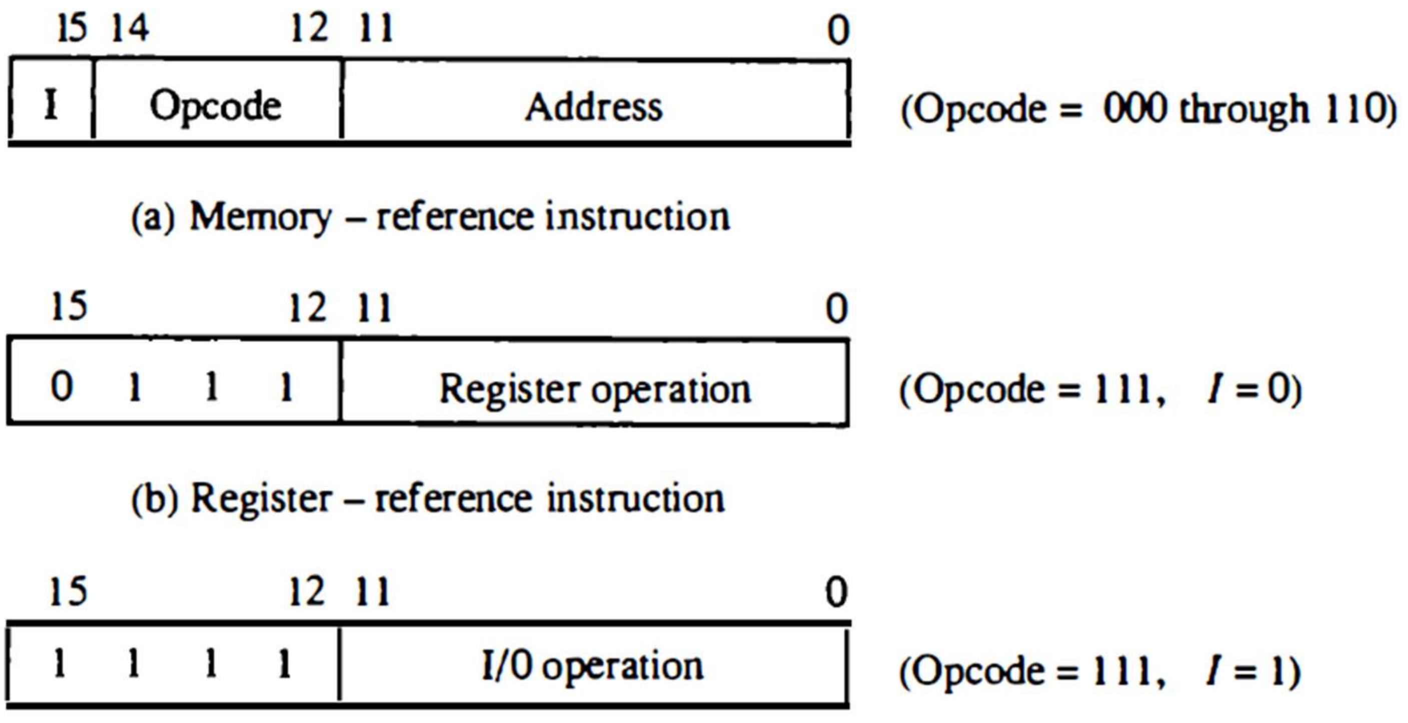
- For example, the two microoperations can be executed at the same time. This can be done by placing the content of AC on the bus (with $S_2S_1S_0 = 100$), enabling the LD (load) input of DR, transferring the content of DR through the adder and logic circuit into AC, and enabling the LD (load) input 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.

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

Computer Instructions:

- The basic computer has three instruction code formats, where 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.



(c) Input – output instruction

Computer Instructions:

- 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 this bit is 0, the instruction is a register-reference type. If the bit is 1, the instruction is an input-output type.

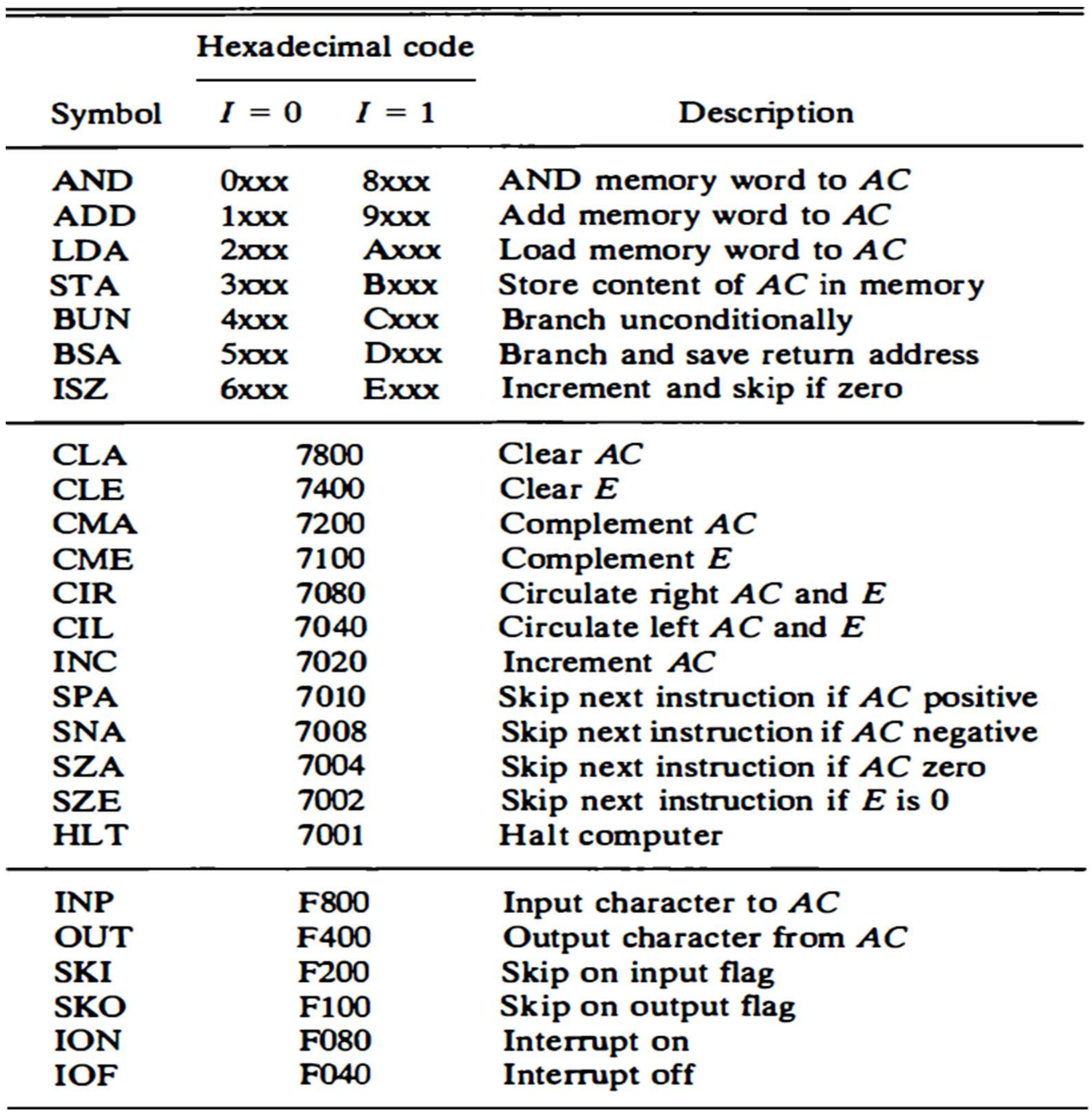
Computer Instructions:

- 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.
- In the instructions for the computer, the symbol designation is a three-letter word and represents an abbreviation intended for 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.

Computer Instructions:

- 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 binary 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.

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

THANK YOU