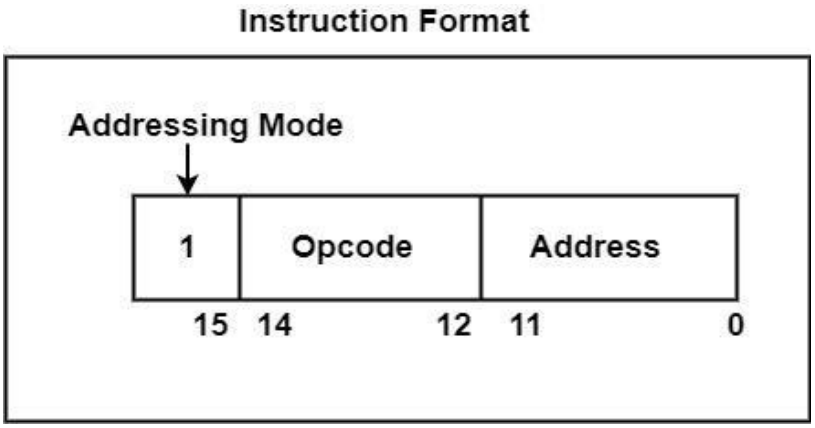
Unit 3. Basic Computer Organization and Design

3.1 Instruction Code

An instruction code is a group of bits instructing the computer to perform specific operations. They are saved in the memory along with the information. Each computer has its specific group of instructions. They can be categorized into two elements as Operation codes (Opcodes) and Address. Opcodes specify the operation for specific instructions. An address determines the registers or the areas that can be used for that operation. Operands are definite elements of computer instruction that show what information is to be operated on. It consists of 12 bits of memory that are required to define the address as the memory includes 4096 words. The 15th bit of the instruction determines the addressing mode (where direct addressing corresponds to 0, indirect addressing corresponds to 1). Therefore, the instruction format includes 12 bits of address and 1 bit for the addressing mode, 3 bits are left for Opcodes.

The following block diagram shows the instruction format for a basic computer.



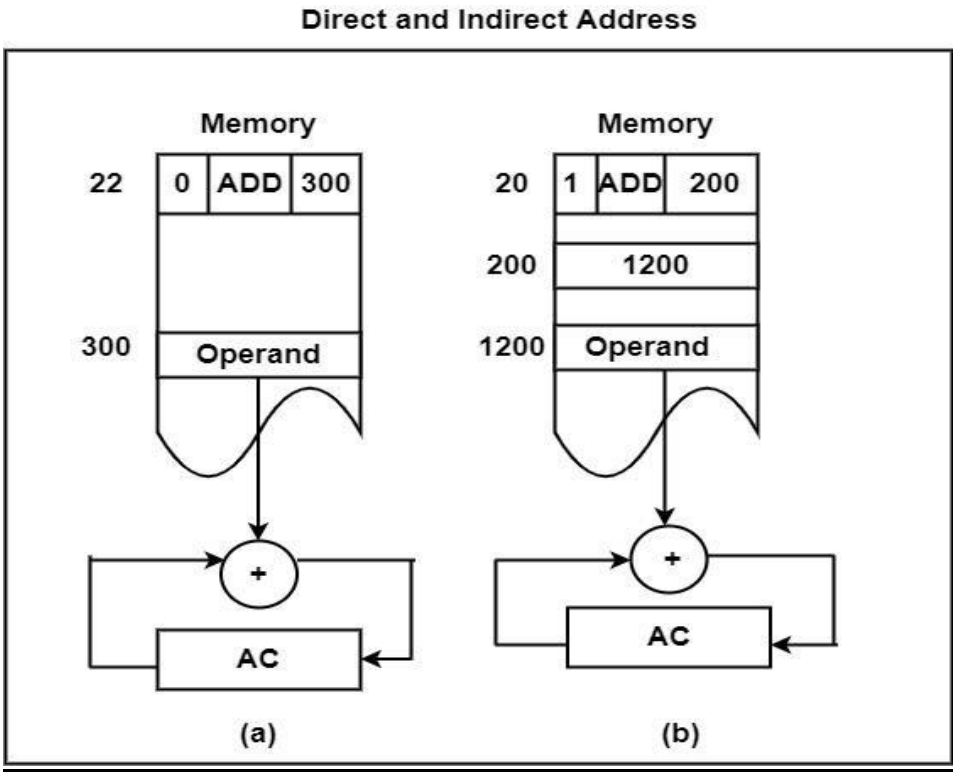
Operation code (Opcode)

- It specify what type of operation is to be performed such as add, sub, shift, complement.
- No of bits required for opcode depends on the number of operations available in computer.
- n bits opcode>= 2n operations
- These operations are implemented on information that is saved in processor registers or memory. Address (Operand)
- Specifies the location of operands (Memory or Memory words) ▪ Memory words are specified by their address ▪ Register are specified by k- bit binary code.
- k- bit address >=2k registers

Addressing of Operand

- When the second part of an instruction code specifies an operand, the instruction is said to have an immediate operand.
- When the second part specifies the address of an operand, the instruction is said to have a direct address.
- When the second part of the instruction designate an address of a memory word in which the address of the operand is found such instruction is said to have indirect address.
- One bit of an instruction code can be used to distinguish between a direct and an indirect address.
- The instruction code format shown in fig 3.1. It consists of a 3 bit operation code, a 12- bit address, and an indirect addressing designated by 1. The mode bit is 0 for direct address and 1 for indirect address.

The figure shows a diagram showing direct and indirect addresses.



3.2 Computer Register

Registers are a type of computer memory used to quickly accept, store, and transfer data and instructions that are being used immediately by the CPU. The registers used by the CPU are often termed as processor registers. A processor register may hold an instruction, a storage address, or any data (such as bit sequence or individual character). The computer needs processor registers for manipulating data and a register.

for holding, memory address. The register holding the memory location calculates the address of the next instruction after the execution of the current instruction is completed. The registers in computer for Instruction sequencing need a counter to calculate the address of the next instruction after execution of the current instruction is completed (PC).

Necessary to provide a register in the control unit for storing the instruction code after it read from memory (IR). Needs processor registers for manipulating data (AC and TR) and a register for holding a memory address (AR).. Following is the list of some of the most common registers used in a basic computer:

Register	Symbol	Number of bits	Function
Data Register	DR	16	Holds memory Operand
Address Register	AR	12	Holds address for the memory
Accumulator	AC	16	Processor Register

Instruction Register	IR	16	Holds instruction code
Program Counter	PC	12	Holds address of the next instructions
Temporary Register	TR	16	Holds temporary data
Input Register	INP R	8	Carries input character
Output Register	OU TR	8	Carries Output Character

The following are the various computer registers and their functions:

Accumulator Register (AC)

Accumulator Register is a general-purpose Register. The initial data to be processed, the intermediate result, and the final result of the processing operation are all stored in this register. If no specific address for the result operation is specified, the result of arithmetic operations is transferred to AC.

Address Register (AR)

The Address Register is the address of the memory location or Register where data is stored or retrieved.

Data Register (DR)

The operand is stored in the Data Register from memory. When a direct or indirect addressing operand is found, it is placed in the Data Register. This value was then used as data by the processor during its operation. It's about the same size as a word in memory.

Instruction Register (IR)

The instruction is stored in the Instruction Register. The instruction register contains the currently executed instruction.

Program Counter (PC)

The Program Counter serves as a pointer to the memory location where the next instruction is stored.

Temporary Register (TR)

The Temporary Register is used to hold data while it is being processed. As Temporary Register stores data, the number of bits it contains is the same as the number of bits in the data word.

Input Register (INPR)

Input Register is a register that stores the data from an input device. The computer's alphanumeric code determines the size of the input register.

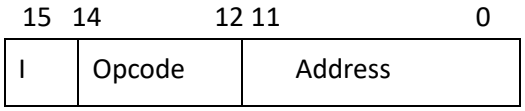
Output Register (OUTR)

The data that needs to be sent to an output device is stored in the Output Register.

3.3 Computer Instructions

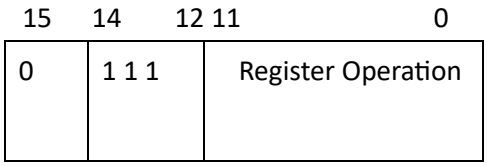
The basic computer has three instruction code formats, as shown in Fig.
. Each format has 16 bits.

- Memory- Reference instruction
- Register- Reference instruction
- Input- Output Instruction



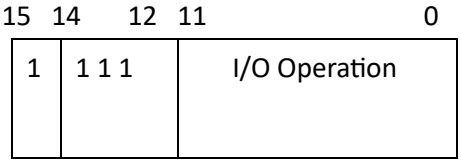
[Opcode= 000 through 110]

Fig: Memory- Reference Instruction



[Opcode= 111 & I =0]

Fig: Register- Reference Instruction



[Opcode=111 & I=1]

Fig: Input/ Output Instruction

i) Memory Reference Instruction

This type of instruction is divided into three parts-mode, opcode and address. The first 12 bits of memory (0-11) specify an operation address. The next three bits (12-14) specify an opcode, and the last bit (I) specifies the addressing mode. If I is 0, it specifies a direct addressing mode; if I is 1, it specifies an indirect one.

The operands specified by memory reference instructions are:

- AND: This instruction performs the 'AND' logical operation between the accumulator's contents and the content that resides in the memory address specified by the instruction. The final result of the operation is stored in the accumulator.

- ADD: This instruction adds the content stored in the accumulator with the content stored in the address mentioned in the instruction and stores the result in the accumulator.

- LDA: This instruction stores the operand from a memory location in the accumulator.

- STA: This instruction stores the accumulator's content in the address specified by the instruction.

- BUN (Branch Unconditionally): This instruction mentions the address of an instruction that is to be executed out of sequence.

- BSA: The 'Branch and Save Return Address' (BSA) instruction transfers the execution of a program to another portion (a subroutine) which is to be executed out of sequence.

- ISZ: This instruction increases the value of the effective address by 1. If the value after incremented is equal to zero, the value of the program counter increments by 1. Its full form is Increment and skip if zero.

ii) Register Reference Instruction

This type of instruction is divided into three parts – mode, opcode, and register operation. The first 12 bits of memory (0-11) specify a register operation. The next three bits (12-14) specify an opcode. The opcode for a register reference instruction is always 111. The last bit specifies the addressing mode. This bit is always zero.

The different types of register operations are as follows:

Symbol	Description	Hexadecimal code
HLT	Halt computer	7001
SZE	Skip if E is zero	7002
SZA	Skip if accumulator is zero	7004
SNA	Skip if accumulator is negative	7008
SPA	Skip if accumulator is positive	7010
INC	Increment accumulator	7020
CIL	Circulate left	7040
CIR	Circulate right	7080
CME	Complement E	7100
CMA	Complement accumulator	7200

CLE	Clear E	7400
CLA	Clear accumulator	7800

iii) Input/ Output Instruction

This type of instruction is divided into three parts – mode, opcode, and input/output operation. The first 12 bits of memory (011) specify an input/output operation. The next three bits (12-14) specify an opcode. The opcode for an I/O reference instruction is always 111. The last bit specifies the addressing mode. This bit is always 1.

The different types of I/O operations are as follows:

Symbol	Description	Hexadecimal code
IOF	Interrupt off	F040
ION	Interrupt on	F080
SKO	Skip on flag output	F100
SKI	Skip on flag input	F200
OUT	Output the contents from an accumulator	F400
INP	Input a character to accumulator	F800

3.4 Timing and Control

- The timing for all register in the basic computer is controlled by a master clock generator.
- The clock pulses are applied to all flip-flops and registers in the system, including the flip-flops and registers in the control unit.
- The clock pulses do not changes the state of a register unless the register is enabled by a control signal.
- The control signal are generated in the control unit and provide control inputs for the multiplexers in the common bus, control inputs in processor registers, and micro-operations for the accumulator.
- There are two major types of control organization:

  Hardwired control

  Microprogrammed control

The difference between hardwired and micro-programmed control unit is given below:

Hardwired Control **Micro-programmed**

1)	The control logic is implemented with gates, flip-flops, decoders and other digital circuit.	The control information is stored in a control memory. The control memory is programmed to initiate the required sequence of microoperations.
2)	The advantage that it can be optimized to produce a fast mode of operation.	Compared with the hardwired control operation is slow.
3)	Requires change in the wiring among the various components if the design has to be modified or changed.	Requires changes or modifications can be done by updating the microprogram in control memory.
4)	More costlier as everything has to be realized in terms of logic gates.	Less costlier than hardwired control as only micro instructions are used for generating control signals
5)	It cannot handle complex instructions as the circuit design for it becomes complex.	It can handle complex instructions
6)	Only limited number of instructions are used due to the hardware implementation.	Control signals for many instructions can be generated.
7)	Used in computer that makes use of Reduced Instruction Set Computers(RISC)	Used in computer that makes use of Complex Instruction Set Computers (CISC).

The block diagram of the hardwired control unit is shown in fig below.

- A Hard-wired Control consists of two decoders, a sequence counter, and a number of logic gates.
- An instruction fetched from the memory unit is placed in the instruction register (IR).
- The component of an instruction register includes; I bit, the operation code, and bits 0 through 11.
- The operation code in bits 12 through 14 are coded with a 3 x 8 decoder.
- The outputs of the decoder are designated by the symbols D0 through D7.
- The operation code at bit 15 is transferred to a flip-flop designated by the symbol I.
- The operation codes from Bits 0 through 11 are applied to the control logic gates.
- The Sequence counter (SC) can count in binary from 0 through 15.

Control Unit of a Basic Computer:

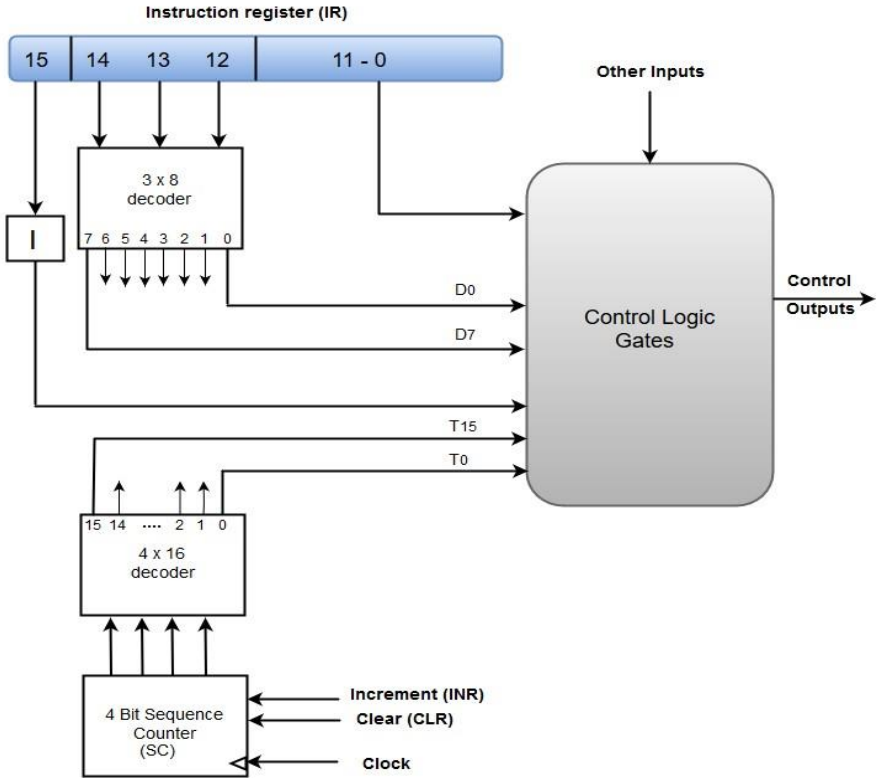


Fig: Control Unit of Basic Computer

The output of the counter are decoded into 16 timing signals T0 through T15.

- The sequence counter SC can be incremented or cleared synchronously.
- The counter is incremented to provide the sequence of timing signals out of the 4×16 decoder.
- As an example consider the case where SC is incremented to provide timing signals T0, T1, T2, T3 and T4 in sequence. At time T4, SC is cleared to 0 if decoder output D3 is active.
- This is expressed symbolically by the statement

  D3T4: SC— 0

The timing diagram below show the time relationship of the control signals.

- The sequence counter SC responds to the positive transition of the clock.
- Initially the CLR input of SC is active. The first positive transition of the clock clears SC to 0, which in turn activating the timing signal T0 out of the decoder. T0 is active during one clock cycle.
- SC is incremented with every positive clock transition, unless its CLR input is active.
- This produce the sequence of timing diagram T0, T1, T2, T3, T4 and so on, as shown in fig.
- The last three waveform in fig below shows how SC is cleared when D3T4=1.
- Output D3 from the operation decoder becomes active at the end of timing signal T2.
- When timing signal T4 becomes active, the output of the AND gate that implements the control function D3T4 is active.
- This signal is applied to the CLR input of SC. On the next positive clock transition the counter is cleared to 0.
- This cause timing signal T0 to become active instead of T5 that would have been active if SC were incremented instead of cleared.

3.5 Instruction Cycle

- A program residing in the memory unit of the computer consists of a sequence of instructions.
- In basic computer each instruction cycle consists of the following phases:
  - ➢ Fetch an instruction from memory
  - ➢ Decode the instruction
  - ➢ Read the effective address from memory if the operand has an indirect address.
  - ➢ Execute the instruction
- After step 4, the control goes back to step 1 to fetch, decode and execute the next instruction.
- This process continue unless a HALT instruction is encountered.

Fetch and Decode

- ➢ PC is loaded with the address of the first instruction in the program.
- ➢ The micro operations for fetch and decode phases are as follows:

T0: AR←PC
T1: IR← M[AR], PC←PC+1
T2: D0,……..,D7 ← Decode IR(12-14),
AR← IR(0-11), I ←IR(15)

Determine the type of Instruction

- During time T_3, the control unit determines the type of instruction i.e. Memory reference, Register reference or Input- Output instruction.
- If D_7=1 then instructions must be register reference or input output else memory reference instruction.

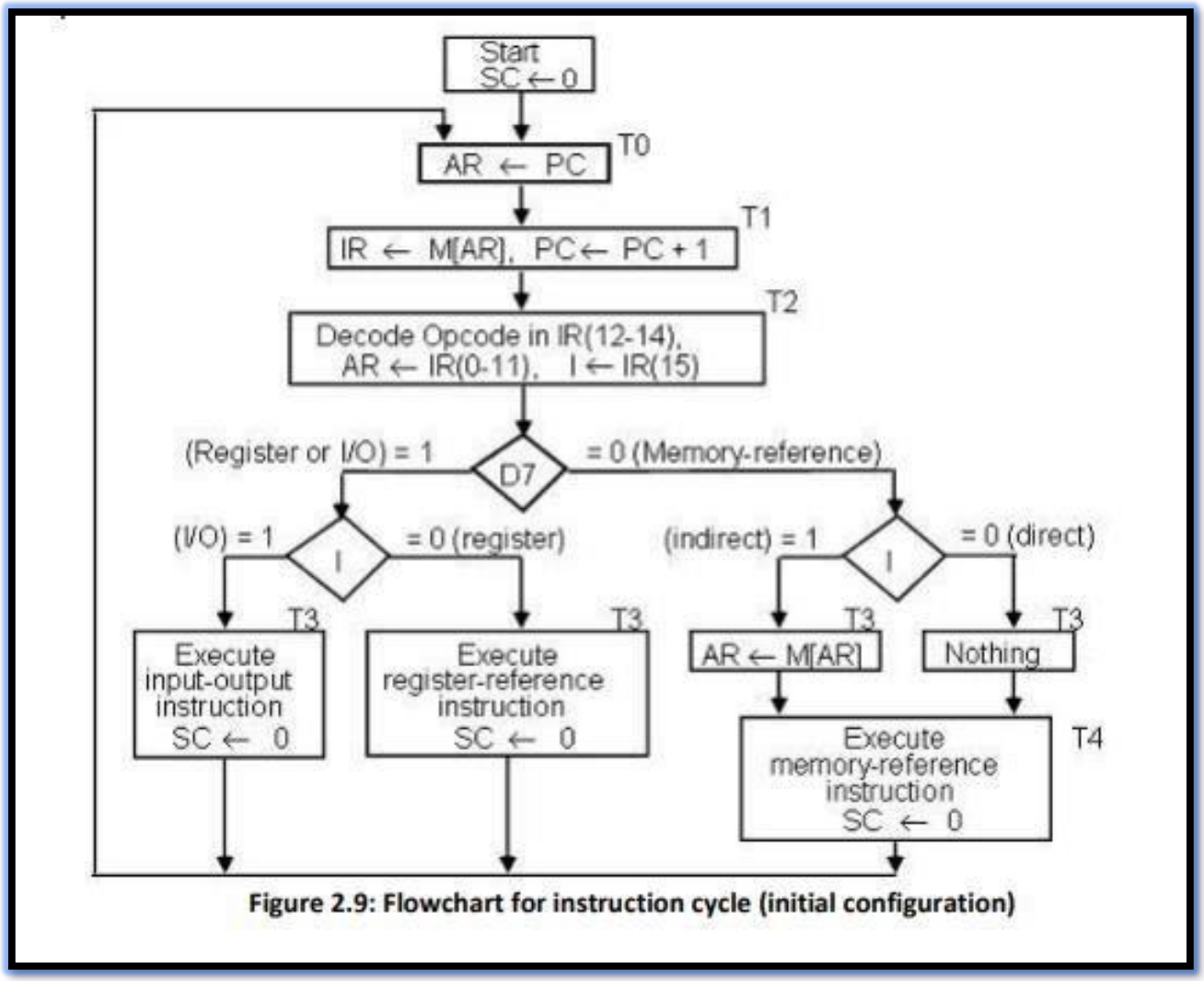


Figure 2.9: Flowchart for instruction cycle (initial configuration)

3.6 Input Output and interrupt

Input Output Configuration

- A computer can serve no useful purpose unless it communicates with the external environment.
- To exhibit the most basic requirement for input and output communication, we will use a terminal unit with a keyboard and printer.

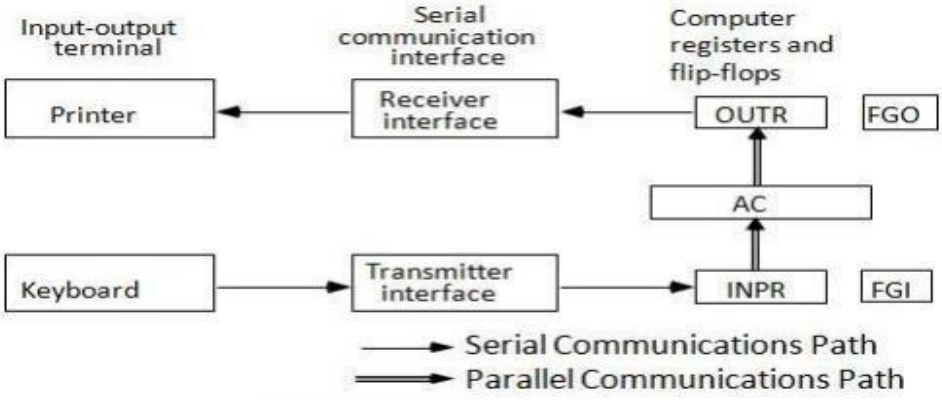


Figure 2.12: Input-output configuration

- The terminal sends and receives serial information and each quantity of information has eight bits of an alphanumeric code.
- The serial information from the keyboard is shifted into the input register INPR.
- The serial information for the printer is stored in the output register OUTR.
- These two register communicate with a communication interface serially and with the AC in parallel.
- The transmitter interface receives serial information from the keyboard and transmits it to INPR. The receiver interface receives information from OUTR and sends it to the printer serially.
- The 1-bit input flag FGI is a control flip-flop. It is set to 1 when new information is available in the input device and is cleared to 0 when the information is accepted by the computer.
- The flag is needed to synchronize the timing rate difference between the input device and the computer.

The process of input information transfer:

- Initially, the input flag FGI is cleared to 0. When a key is struck in the keyboard, an 8-bit alphanumeric code is shifted into INPR and the input flag FGI is set to 1.
- As long as the flag is set, the information in INPR cannot be changed by striking another key. The computer checks the flag bit; if it is 1, the information from INPR is transferred in parallel into AC and FGI is cleared to 0.
- Once the flag is cleared, new information can be shifted into INPR by striking another key.

The process of outputting information:

- The output register OUTR works similarly but the direction of information flow is reversed.
- Initially, the output flag FGO is set to 1. The computer checks the flag bit; if it is 1, the information from AC is transferred in parallel to OUTR and FGO is cleared to 0. The output device accepts the coded information, prints the corresponding character, and when the operation is completed, it sets FGO to 1.
- The computer does not load a new character into OUTR when FGO is 0 because this condition indicates that the output device is in the process of printing the character.

Input-Output Instruction

- Input and output instructions are needed for transferring information to and from AC register, for checking the flag bits, and for controlling the interrupt facility.
- Input-output instructions have an operation code 1111 and are recognized by the control when D7 = 1 and I = 1.
- The remaining bits of the instruction specify the particular operation.
- The control functions and Micro-operations for the input-output instructions are listed below.

INP	AC(0-7) ← INPR, FGI ← 0	Input char. to AC
OUT	OUTR ← AC(0-7), FGO ← 0	Output char. from AC
SKI	if(FGI = 1) then (PC ← PC + 1)	Skip on input flag
SKO	if(FGO = 1) then (PC ← PC + 1)	Skip on output flag
ION	IEN ← 1	Interrupt enable on
IOF	IEN ← 0	Interrupt enable off

Table 2.2: Input Output Instructions

- The INP instruction transfers the input information from INPR into the eight low-order bits of AC and also clears the input flag to 0.
- The OUT instruction transfers the eight least significant bits of AC into the output register OUTR and clears the output flag to 0.
- The next two instructions in Table check the status of the flags and cause a skip of the next instruction if the flag is 1.
- The instruction that is skipped will normally be a branch instruction to return and check the flag again.
- The branch instruction is not skipped if the flag is 0. If the flag is 1, the branch instruction is skipped and an input or output instruction is executed.
- The last two instructions set and clear an interrupt enable flip-flop IEN. The purpose of IEN is explained in conjunction with the interrupt operation.