BASIC COMPUTER DESIGN

PREPARED BY:

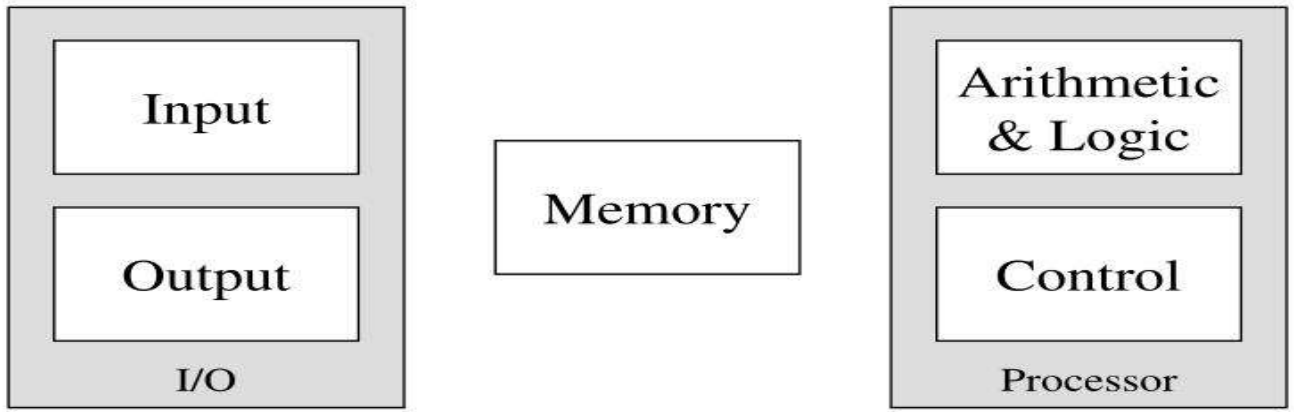
PROF. MIKSHA SOLANKI

Miksha.solanki21591@paruluniversity.ac.in

EC DEPT.

FUNCTIONAL UNIT

Computer Functional Units



THE OPERATIONS OF A COMPUTER

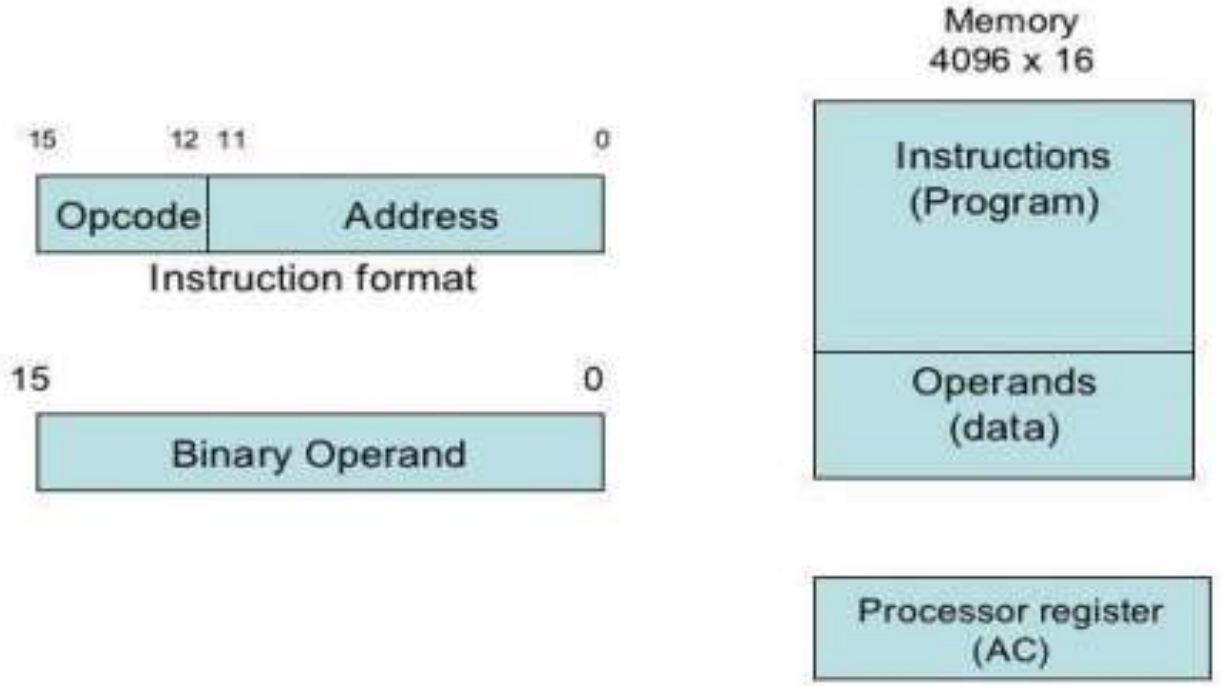
1. A set of instructions called a program reside in the main memory of computer.
2. The CPU fetches those instructions sequentially one-by-one from the main memory, decodes them and performs the specified operation on associated data operands in ALU.
3. Processed data and results will be displayed on an output unit.
4. All activities pertaining to processing and data movement inside the computer machine are governed by control unit.

BASIC OPERATIONAL CONCEPTS

- **Instruction Code:** An instruction code is a group of bits that instruct the computer to perform a specific operation.
- **Operation Code:** The operation code of an instruction is a group of bits that define such operations as add, subtract, multiply, shift, and complement. It is known as OPCODE.
- **Accumulator (AC) :** Computers that have a single-processor register usually assign to it the name accumulator (AC) accumulator and label it AC. The operation is performed with the memory operand and the contents of AC.

STORED PROGRAM ORGANIZATION

- The simplest way to organize a computer is to have one processor register and an instruction code format with two parts.
- The first part specifies the operation to be performed and the second specifies an address.
- The memory address tells the control where to find an operand in memory.
- This operand is read from memory and used as the data to be operated on together with the data stored in the processor register.
- 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.
- 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

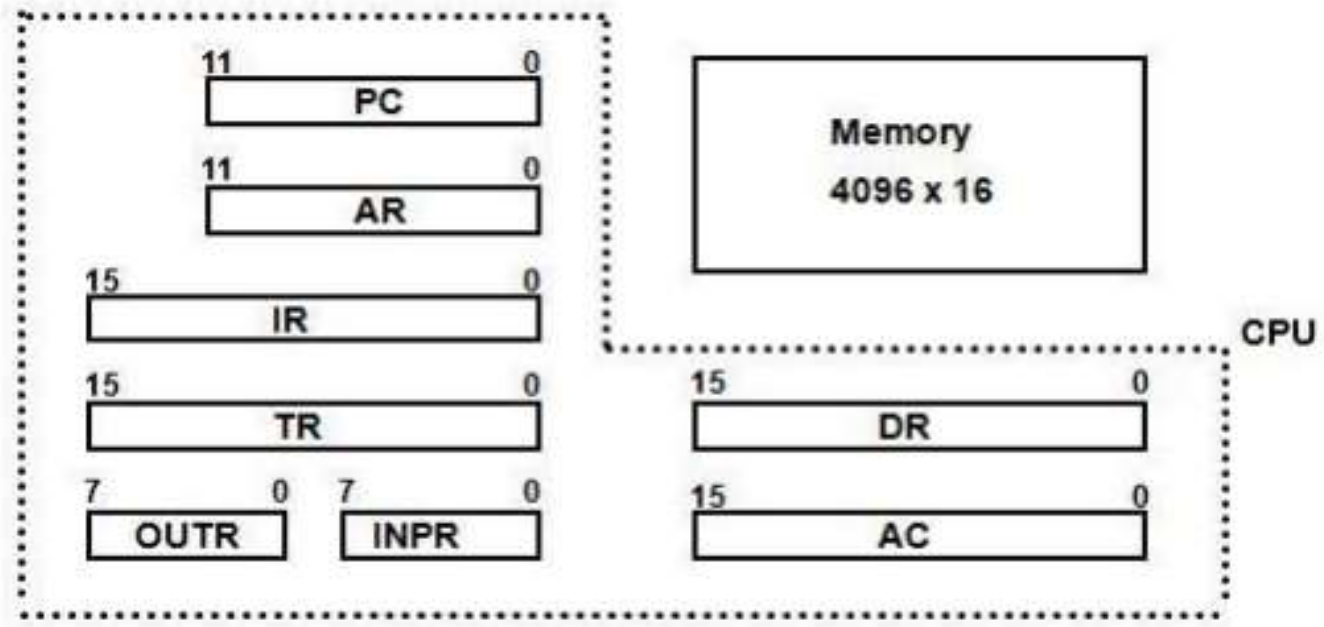
DIRECT AND INDIRECT ADDRESSING

- The second part of an instruction format specifies the address of an operand, the instruction is said to have a direct 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.

Direct Address	Indirect Address
When the second part of an instruction code specifies the address of an operand, the instruction is said to have a direct address.	When the second part of an instruction code specifies the address of a memory word in which the address of the operand, the instruction is said to have a direct address.
For instance the instruction MOV R0 00H. R0, when converted to machine language is the physical address of register R0. The instruction moves 0 to R0.	For instance the instruction MOV @R0 00H, when converted to machine language, @R0 becomes whatever is stored in R0, and that is the address used to move 0 to. It can be whatever is stored in R0.

REGISTERS OF BASIC COMPUTER

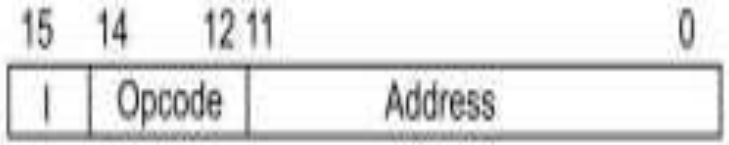
- It is necessary to provide a register in the control unit for storing the instruction code after it is read from memory.
- The computer needs processor registers for manipulating data and a register for holding a memory address.
- The Fig. shown basic computer register and memory:



- 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.
- The memory address register (AR) has 12 bits.
- 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.
- 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.
- 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.

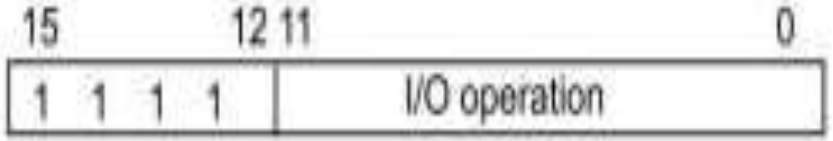
INSTRUCTION FORMAT WITH ITS TYPES

- The basic computer has three instruction code formats, as shown in figure:
- Memory-Reference Instructions (OP-code = 000 ~ 110)



- Register-Reference Instructions (OP-code = 111, I = 0)

- Input-Output Instructions (OP-code =111, I = 1)



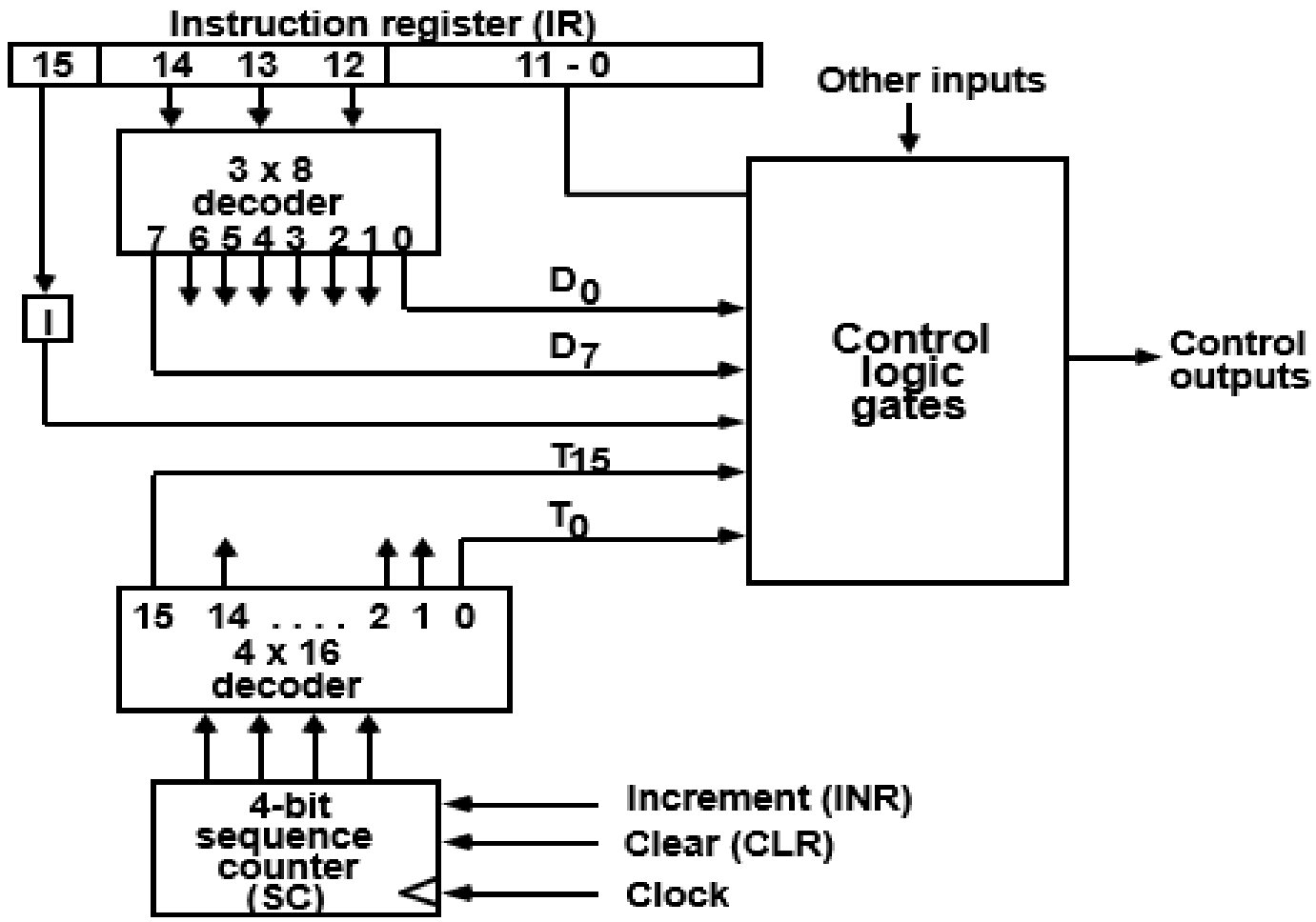
CONTINUE…

- 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.
- 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.
- 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.

TIMING AND CONTROL UNIT

- Components of Control unit are

1. Two decoders
2. A sequence counter
3. Control logic gates

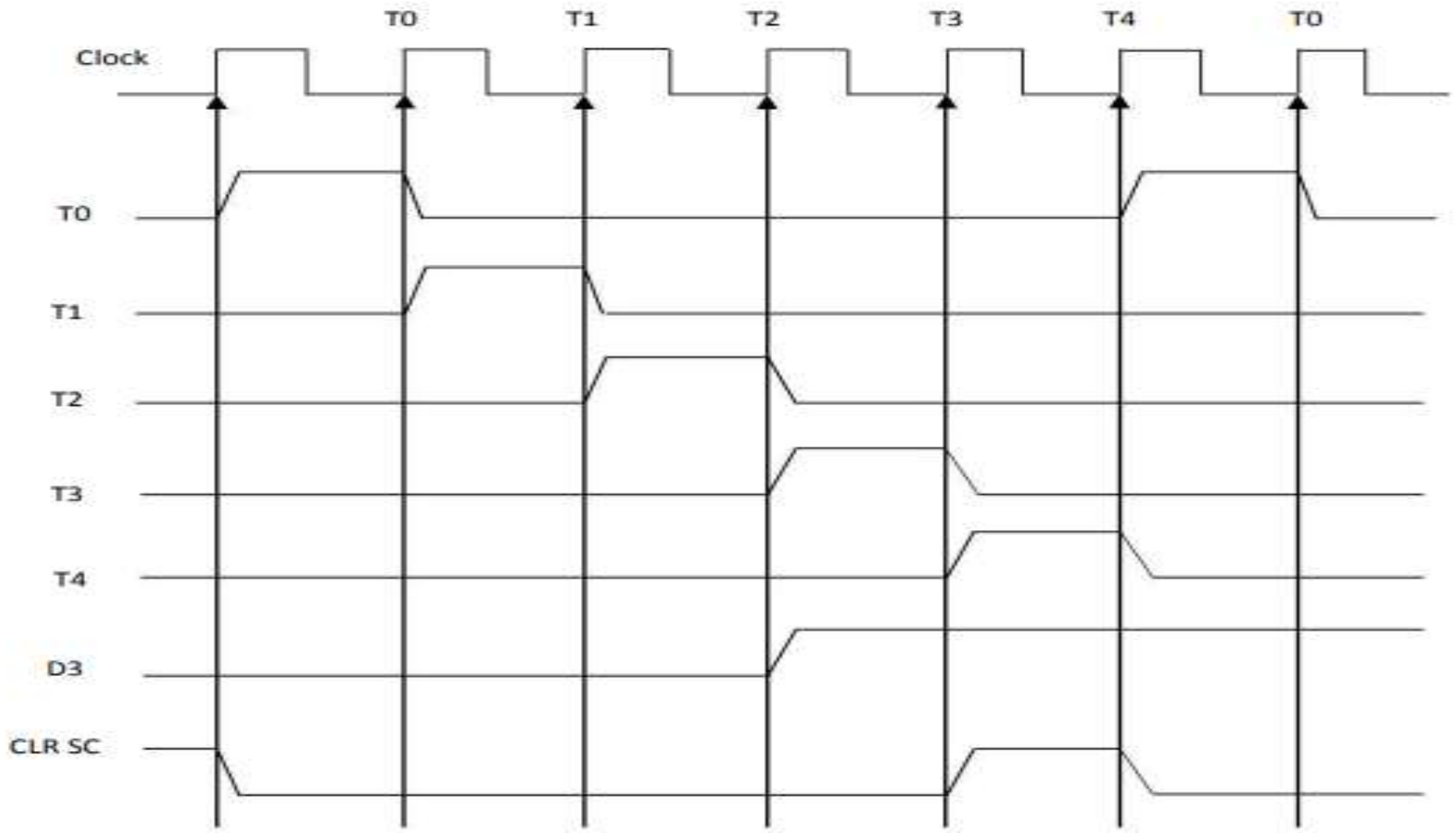


CONTINUE....

- An instruction read from memory is placed in the instruction register (IR). In control unit the IR is divided into three parts: I bit, the operation code (12-14)bit, and bits 0 through 11. The operation code in bits 12 through 14 are decoded with a 3 X 8 decoder.

- Bit-15 of the instruction is transferred to a flip-flop designated by the symbol I.
- The eight outputs of the decoder are designated by the symbols D0 through D7. Bits 0 through 11 are applied to the control logic gates. The 4-bit sequence counter can count in binary from 0 through 15.The outputs of counter are decoded into 16 timing signals T0 through T15.
- The sequence counter SC can be incremented or cleared synchronously. Most of the time, the counter is incremented to provide the sequence of timing signals out of 4 X 16 decoder. Once in awhile, the counter is cleared to 0, causing the next timing signal to be T0.

TIMING DIAGRAM



CONTINUE…

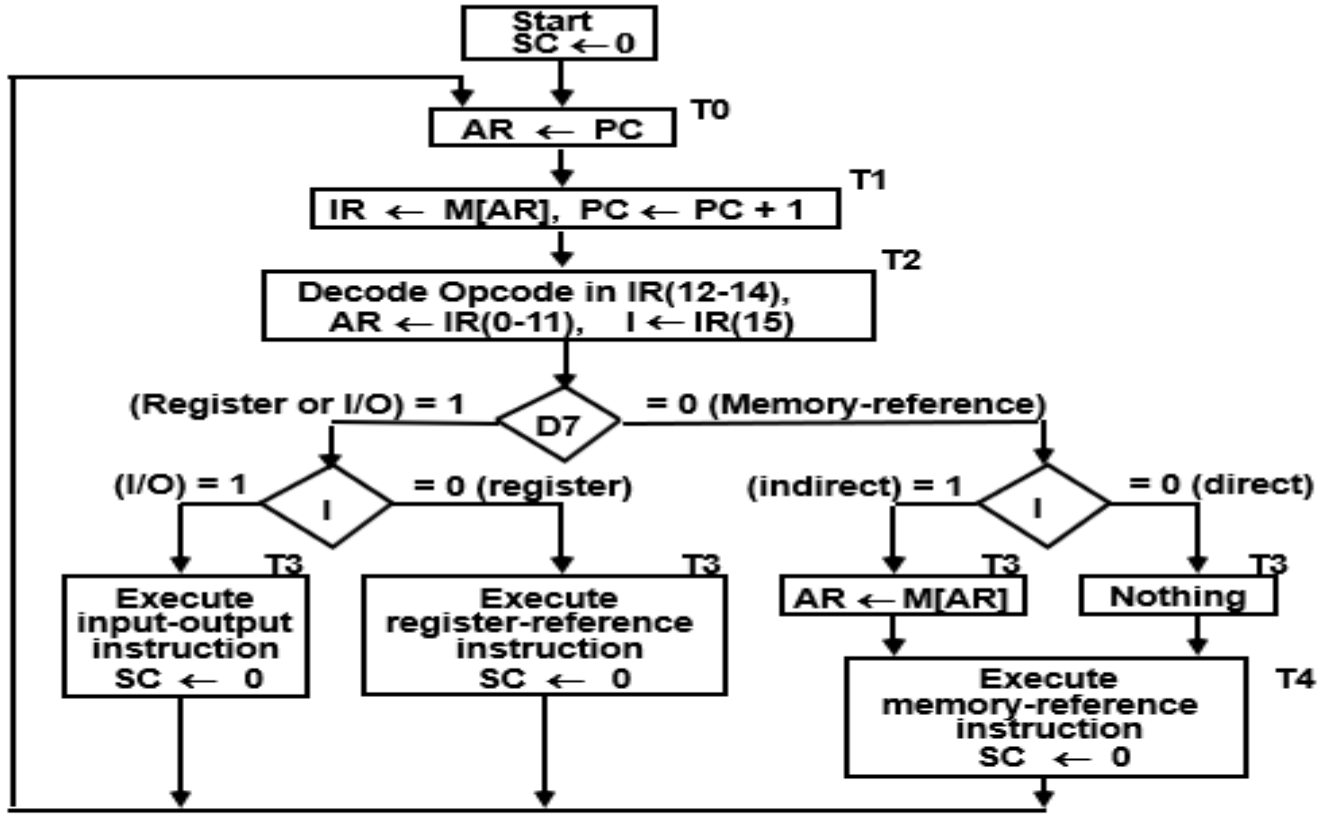
- The sequence counter SC responds to the positive transition of the clock.
- Initially, the CLR input of SC is active.
- The first positive transition of the clock clears SC to 0, which in turn activates the timing T0 out of the decoder. T0 is active during one clock cycle. The positive clock transition labeled T0 in the diagram will trigger only those registers whose control inputs are connected to timing signal T0.
- SC is incremented with every positive clock transition, unless its CLR input is active.
- This procedures the sequence of timing signals T0, T1, T2, T3 and T4, and so on. If SC is not cleared, the timing signals will continue with T5, T6, up to T15 and back to T0.
- The last three waveforms 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 becomes active.
- This signal is applied to the CLR input of SC. On the next positive clock transition the counter is cleared to 0. This causes the timing signal T0 to become active instead of T5 that would have been active if SC were incremented instead of cleared.

INSTRUCTION CYCLE

- A program residing in the memory unit of the computer consists of a sequence of instructions.

1. Fetch an instruction from memory.
2. 2. Decode the instruction.
3. Read the effective address from memory if the instruction has an indirect address.
4. 4. Execute the instruction.
5. After step 4, the control goes back to step 1 to fetch, decode and execute the next instructions.
6. This process continues unless a HALT instruction is encountered.

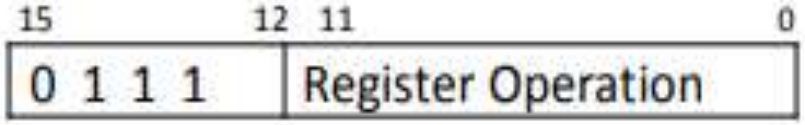
FLOW CHART FOR INSTRUCTION CYCLE



D'_7IT_3: $AR \leftarrow M[AR]$
$D'_7I'T_3$: Nothing
$D_7I'T_3$: Execute a register-reference instr.
D_7IT_3: Execute an input-output instr.

REGISTER REFERENCE INSTRUCTION

- When the register-reference instruction is decoded, D7 bit is set to 1.
- Each control function needs the Boolean relation D7 I' T3



There are 12 register-reference instructions listed below:

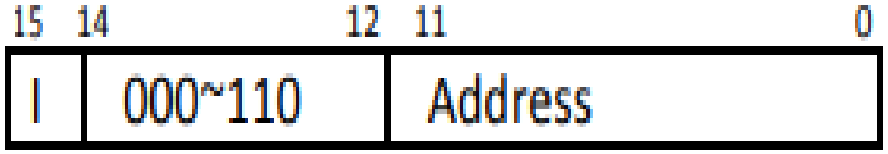
	r:	SC←0	Clear SC
CLA	rB_{11}:	AC ← 0	Clear AC
CLE	rB_{10}:	E ← 0	Clear E
CMA	rB_9:	AC ← AC'	Complement AC
CME	rB_8:	E ← E'	Complement E
CIR	rB_7:	AC ← shr AC, AC(15) ← E, E ← AC(0)	Circular Right
CIL	rB_6:	AC ← shl AC, AC(0) ← E, E ← AC(15)	Circular Left
INC	rB_5:	AC ← AC + 1	Increment AC
SPA	rB_4:	if (AC(15) = 0) then (PC ← PC+1)	Skip if positive
SNA	rB_3:	if (AC(15) = 1) then (PC ← PC+1	Skip if negative
SZA	rB_2:	if (AC = 0) then (PC ← PC+1)	Skip if AC is zero
SZE	rB_1:	if (E = 0) then (PC ← PC+1)	Skip if E is zero
HLT	rB_0:	S ← 0 (S is a start-stop flip-flop)	Halt computer

CONTINUE....

- These 12 bits are available in IR (0-11). They were also transferred to AR during time T2.
- These instructions are executed at timing cycle T3.
- The first seven register-reference instructions perform clear, complement, circular shift and increment microoperations on the AC or E registers.
- The next four instructions cause a skip of the next instruction in sequence when condition is satisfied. The skipping of the instruction is achieved by incrementing PC.
- The condition control statements must be recognized as part of the control conditions.
- The AC is positive when the sign bit in AC(15) = 0; it is negative when AC(15) = 1. The content of AC is zero (AC = 0) if all the flip-flops of the register are zero.
- The HLT instruction clears a start-stop flip-flop S and stops the sequence counter from counting. To restore the operation of the computer, the start-stop flip-flop must be set manually.

MEMORY REFERENCE INSTRUCTIONS

- When the memory-reference instruction is decoded, D7 bit is set to 0



The following table lists seven memory-reference instructions.

Symbol	Operation Decoder	Symbolic Description
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR], E \leftarrow C_{out}$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
BUN	D_4	$PC \leftarrow AR$
BSA	D_5	$M[AR] \leftarrow PC, PC \leftarrow AR + 1$
ISZ	D_6	$M[AR] \leftarrow M[AR] + 1$, if $M[AR] + 1 = 0$ then $PC \leftarrow PC+1$

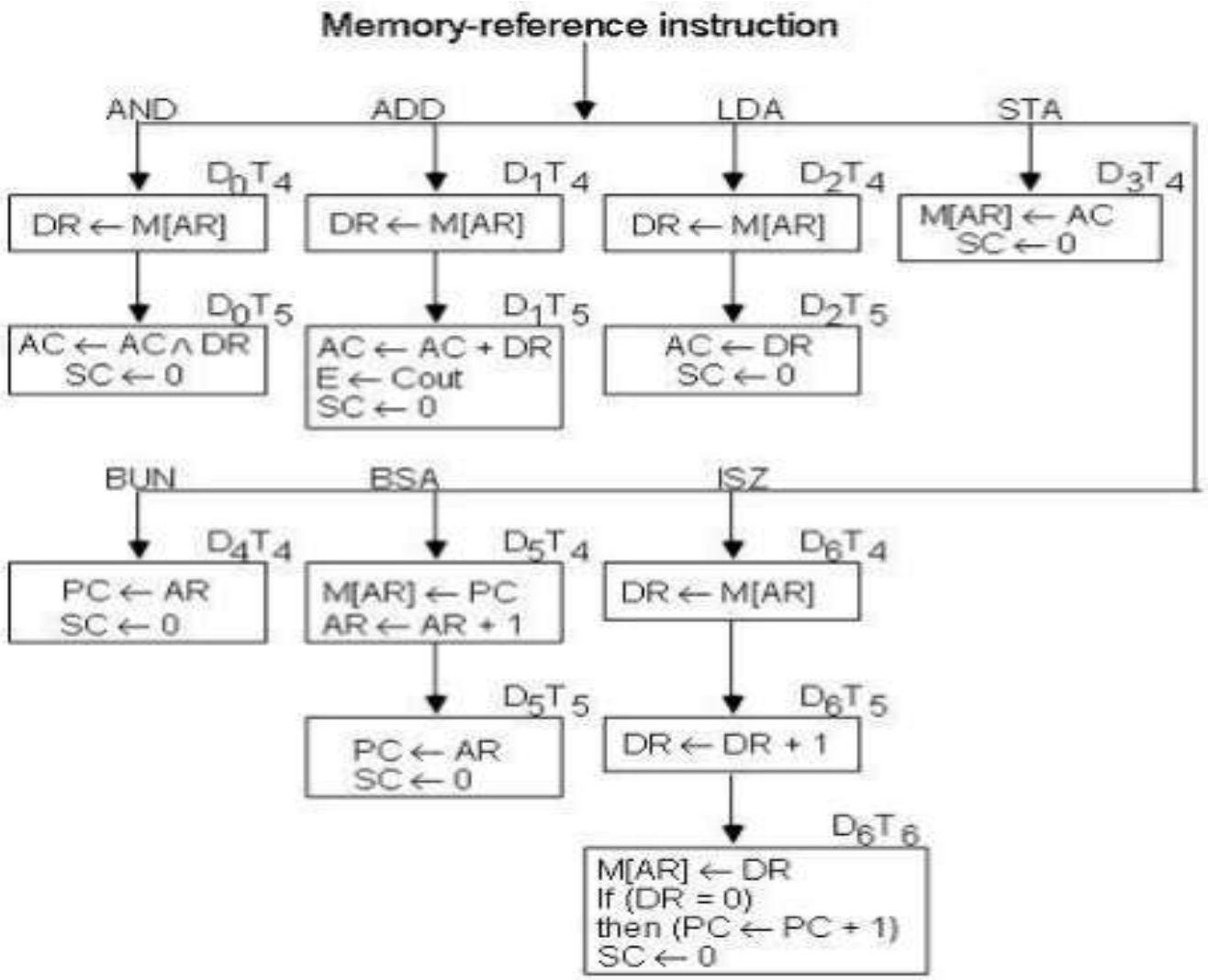
CONTINUE…

- The effective address of the instruction is in the address register AR and was placed there during timing signal T2 when I = 0, or during timing signal T3 when I = 1.
- The execution of the memory-reference instructions starts with timing signal T4.
- **AND to AC:** This is an instruction that performs the AND logic operation on pairs of bits in AC and the memory word specified by the effective address. The result of the operation is transferred to AC.

D0T4: DR ← M[AR]

D0T5: AC ← AC ^DR, SC ← 0

FLOW CHART


Memory-reference instruction
AND
ADD
LDA
STA
D_0T_4
DR ← M[AR]
D_1T_4
DR ← M[AR]
D_2T_4
DR ← M[AR]
D_3T_4
M[AR] ← AC
SC ← 0
D_0T_5
AC ← AC ∧ DR
SC ← 0
D_1T_5
AC ← AC + DR
E ← Cout
SC ← 0
D_2T_5
AC ← DR
SC ← 0
BUN
BSA
ISZ
D_4T_4
PC ← AR
SC ← 0
D_5T_4
M[AR] ← PC
AR ← AR + 1
D_6T_4
DR ← M[AR]
D_5T_5
PC ← AR
SC ← 0
D_6T_5
DR ← DR + 1
D_6T_6
M[AR] ← DR
If (DR = 0)
then (PC ← PC + 1)
SC ← 0

INPUT-OUTPUT INSTRUCTIONS

- 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 microoperations for the input-output instructions are listed below:

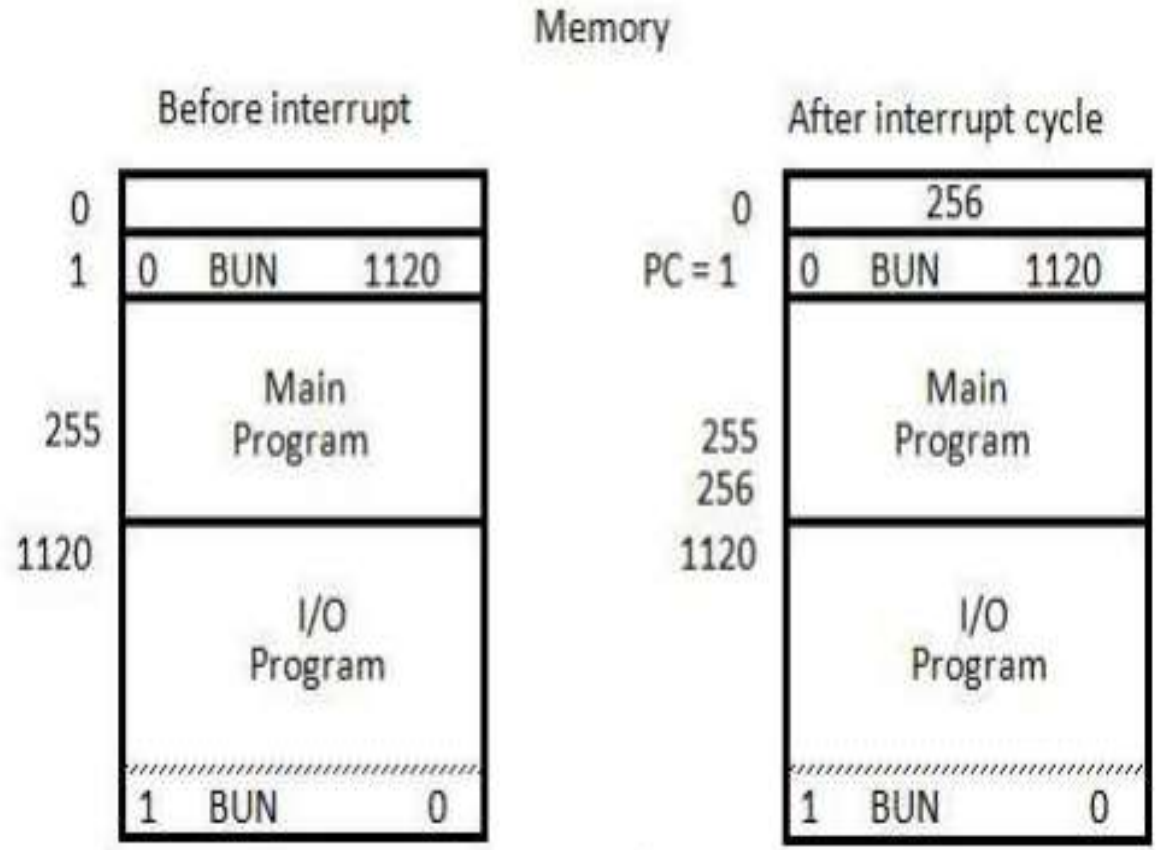
INP	AC(0-7) ← INPR, FGI ← 0	Input char. to AC
OUT	OUTR ← AC(0-7), FGO ← 0	Output char. from AC
SKI	if(FGI = 1) then (PC ← PC + 1)	Skip on input flag
SKO	if(FGO = 1) then (PC ← PC + 1)	Skip on output flag
ION	IEN ← 1	Interrupt enable on
IOF	IEN ← 0	Interrupt enable off

INTERRUPT CYCLE

- The interrupt cycle is a hardware implementation of a branch and save return address operation.
- The return address available in PC is stored in a specific location where it can be found later when the program returns to the instruction at which it was interrupted. This location may be a processor register, a memory stack, or a specific memory location.
- Here we choose the memory location at address 0 as the place for storing the return address.
- Control then inserts address 1 into PC and clears IEN and R so that no more interruptions can occur until the interrupt request from the flag has been serviced.

CONTINUE..

- An example that shows what happens during the interrupt cycle is shown in Figure:



CONTINUE..

- Suppose that an interrupt occurs and R = 1, while the control is executing the instruction at address 255. At this time, the return address 256 is in PC.
- The programmer has previously placed an input-output service program in memory starting from address 1120 and a BUN 1120 instruction at address 1.
- The content of PC (256) is stored in memory location 0, PC is set to 1, and R is cleared to 0.
- At the beginning of the next instruction cycle, the instruction that is read from memory is in address 1 since this is the content of PC. The branch instruction at address 1 causes the program to transfer to the input-output service program at address 1120.
- This program checks the flags, determines which flag is set, and then transfers the required input or output information. Once this is done, the instruction ION is executed to set IEN to 1 (to enable further interrupts), and the program returns to the location where it was interrupted.
- The instruction that returns the computer to the original place in the main program is a branch indirect instruction with an address part of 0. This instruction is placed at the end of the I/O service program.
- The execution of the indirect BUN instruction results in placing into PC the return address from location 0

REFERENCE

- COMPUTER SYSTEM ARCHITECTURE, MORRIS M. MANO, 3RD EDITION, PRENTICE HALL INDIA.