"Basic Computer Organization and Design"

Instruction Codes

Instructions:—

- Program → [A sequence of (machine) instructions]
- (Machine) Instruction → A group of bits that tell the computer to perform a specific operation (a sequence of micro-operations)
- The instruction of a program, along with any needed data are stored in memory
- The CPU reads the next instruction from memory
- It is placed in an Instruction Register (IR)
- Control circuitry in Control unit then translate the instruction into the sequence of micro-operations necessary to implement it.

Instruction Codes:— A set of instructions that specify the operations, operands, & the sequence by which processing has to occur. An instruction code is a group of bits that tells the computer to perform a specific operation part.

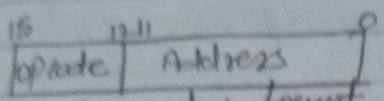
Operation Code:— The operation code of an instruction is a group of bits that define such operation as add, subtract, multiply, shift and complement. The number of bits required for the operation code of an instruction depends on the total number of operations available in the computer. The

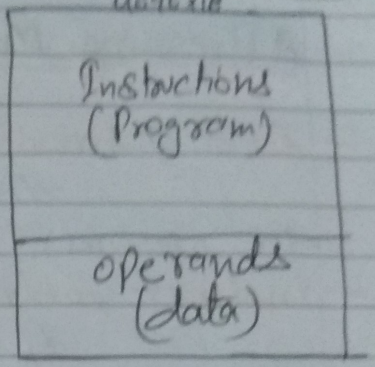
operation code must consist of at least n-bits for a given 2^n (or less) distinct operations.

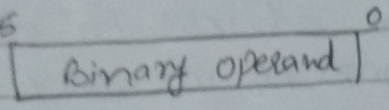
Accumulator (AC):— Computers that have a single processor registers usually assign to it the name Accumulator (AC) & label it AC. The operation is performed with the memory operands and the content of AC.

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 (opcode) (operation code) and the second part is an address that ~~specify~~ specifies the registers and/or locations in memory to use for that operation.
- 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. figure shows this type of organization
- Instruction are stored in one section of memory & data in another.
- In a Basic Computer, since the memory







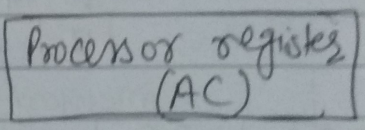
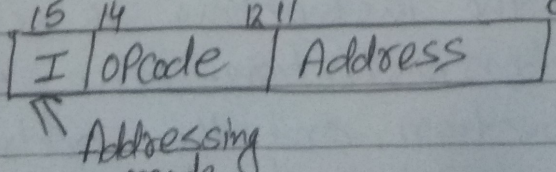


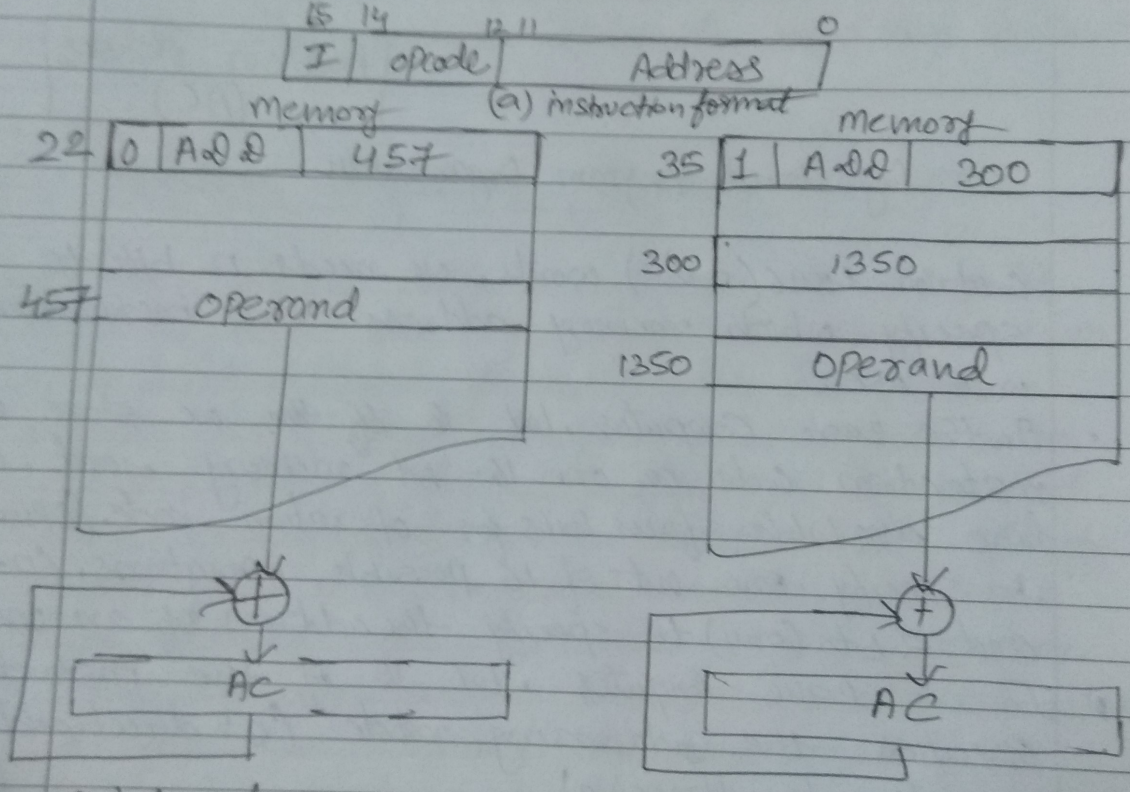
Fig: Stored Program Organization

contain 4096 ($=2^{12}$) words, we needs 12 bit to specify which memory address this instruction will use.

- In the Basic Computer, ~~bit 15 of these~~ we store each instruction code in one 16-bit memory word, we have available four bits for operation code (opcode) to specify one out of 16 possible operations. (12-15) and 12 bits (0-11) to specify the address of an operands.
- In a Basic Computer, bit 15 of the instruction specifies the addressing mode (0: direct addressing 1: indirect addressing)
- Since the memory words, and hence the instructions, are 16 bits long, that leaves 3 bits for the instruction's opcode



Addressing Modes: — The address field of an instruction can represent either
- **Direct Address:** — the address in memory of the data to use (the address of the operand), or
- **Indirect address:** — the address in memory of the address in memory of the data to use



- A direct address instruction is shown in fig (b) It 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 & add it to the content of AC.

- The instruction in address 35 shown in fig(c) has a mode bit I=1, 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.

Effective Address(EA) :- The address, that can be directly used without modification to access an operand for a computation-type instruction, or as the target address for a branch type instruction.

Basic Computer Registers:-

- It is necessary to provide a register in the control unit for storing the instruction code after it is read from memory.
- A processor has many registers to hold instructions, address, data etc.
- The Processor has a register, the program counter (PC) that holds the memory address of the next instruction to get.
  - Since the memory in the basic computer only has 4096 locations, the PC only needs 12 bits.
- In a direct or indirect addressing, the processor needs to keep track of what locations in memory it is addressing: The Address Register (AR) is used for this.
  - The AR is a 12 bit register in the Basic Comp
- When an operand is found, using either direct or indirect addressing, it is placed in the Data Register. (DR). The processor then uses this value as data for its operation.
- The Basic Computer has a single general purpose register the Accumulator (AC), it can be referred to in instructions
  - e.g. load AC with the contents of a specific memory location; store the contents of AC into a specified memory location.
- Often a processor will need a scratch register to store intermediate results or other temporary data; in the Basic Computer this is the Temporary Register (TR).

- The Basic computer uses a very simple model of input/output (I/O) operations.
  - Input devices are considered to send 8 bits of characters data to the processor
  - The processor can send 8-bits of characters data to output devices.
- The Input Register (INPR) holds an 8 bit characters gotten from an input device.
- The output Register (OUTR) holds an 8-bit character to be send to an output device.

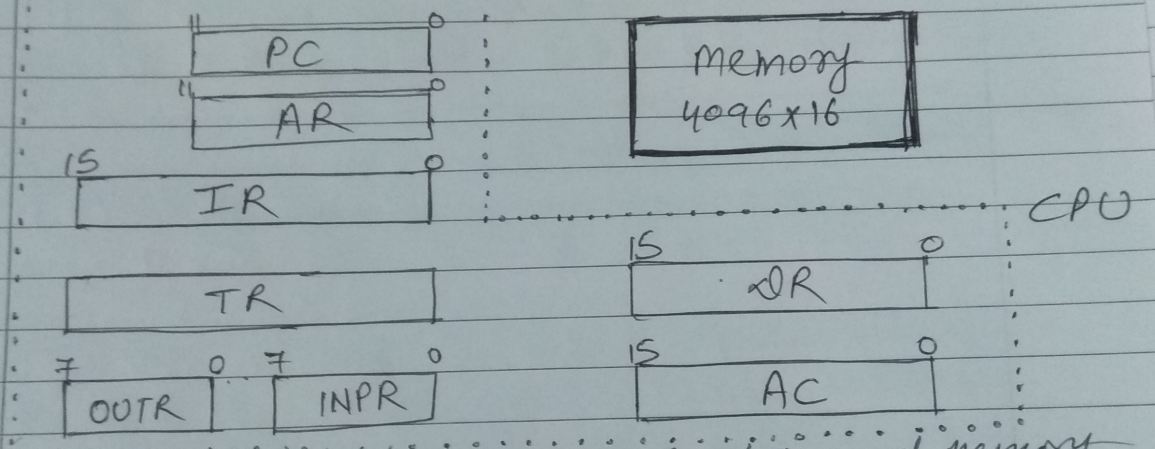


fig: Basic Computer Registers and memory

Register symbol	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

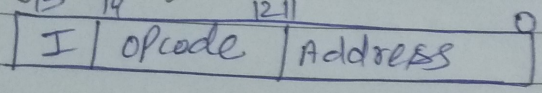
Table: List of Registers for Basic Computer

Computer Instructions:—

Instruction format with its types.

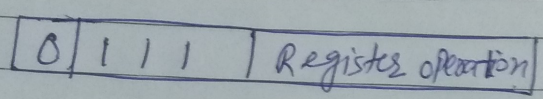
The basic computer has three instruction code formats.

(a) • Memory - Reference Instructions :—



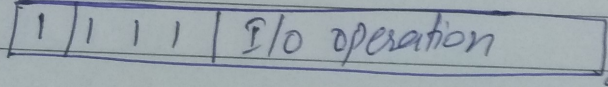
Opcode = 000 ~ 110

(b) Register - Reference Instructions:—



{ Op-code = 111, I = 0 }

(c) Input-output Instructions:—



{ Op-code = 111, I = 1 }

- each format has 16 bits. The operation code (Op-code) part of the instruction contains three bits & the meaning of the remaining 13 bits depends on the operation code.
- A memory reference instruction uses 12 bits to specify an address & one bit to specify the addressing mode I. I is equal to 0 for direct & 1 for indirect address.
- The register reference instruction are recognized by the operation code 111, with a 0 in the leftmost bit (bit 15) of the instruction. This 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 needed a reference to memory & is recognized by the operation code

111 with a 1 in the ~~left~~ leftmost bit of the instruction. The remaining 12 bits are used to specify the type of input-output operation or test performed.

Basic 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 & save return address
ISZ	6xxx	Exxx	Increment and skip if zero

Memory reference instruction

CLA	7800	Clear AC
CLE	7400	Clear E
CMA	7200	Complement AC
CME	7100	Complement E
CIR	7080	Circulate right AC & E
CIL	7040	Circulate left AC & E
INC	7020	Increment AC
SPA	7010	Skip next instruc. if AC positive
SNA	7008	Skip " " if AC negative
SZA	7004	" " " if AC zero
SZE	7002	" " " if E is 0
HLT	7001	Halt computer

Register Reference instruction

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

Input output instruction

Control Unit :— • Control unit (CU) of a processor translates from machine instructions to the control signals for the microoperations that implement them.

• all sequential circuits (flip-flops & registers) in the basic computer CPU are driven by a master clock, with the exception of the INPR register. At each clock pulse, the control unit sends control signals to control inputs of the bus, the registers, and the ALU.

Control unit design and implementation can be done by two general methods:

→ **A Hardwired Control :—** CU is made up of sequential & combinational circuits to generate the control signals.

→ **Microprogrammed Control :—** A control memory on the processor contains microprograms that activate the necessary control signals.

In the hardwired organization, the control logic is implemented with gates, flip-flops, decoders, and other digital circuits. It has the advantage that it can be optimized to produce a fast mode of operation. In the microprogrammed organization, the control information is stored in a control memory. The control memory is programmed to initiate the required sequence of microoperations. A hardwired control, as the name implies, requires changes in the wiring among the various components if the design has to be modified or changed. In the microprogrammed control, any required changes or modifications can be done by updating the microprogram in control memory.

Timing And Control :

A hardwired implementation of the control unit for the basic computer.

The block diagram of the control unit consists of two decoders, a sequence counter, and a number of control logic gates. An instruction read from memory is placed in the instruction register (IR), which is divided into three parts: I-bit, the operation code (12-14) bit, and bit 0 through 11. The operation code in bits 12 through 14 are decoded

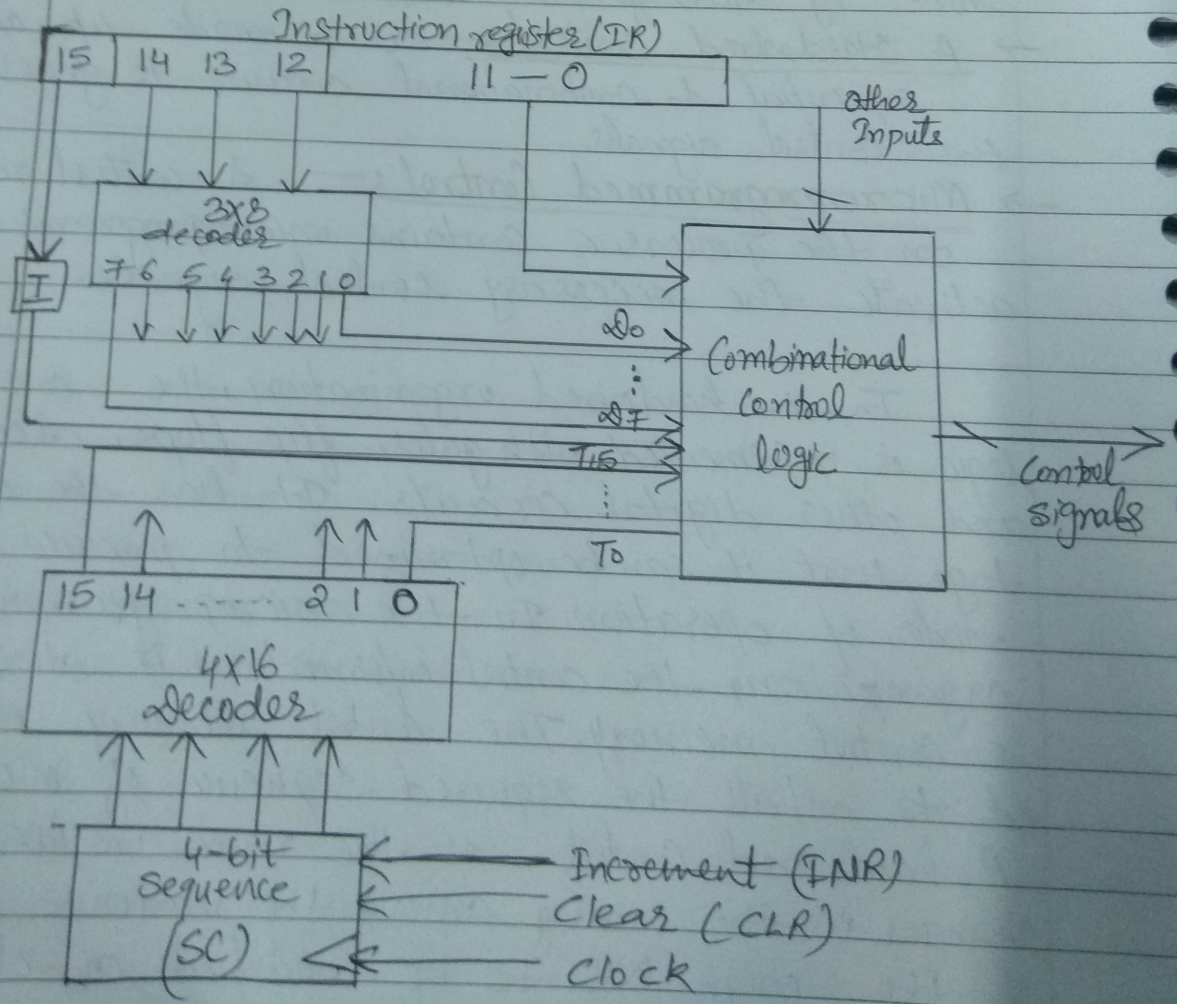


fig: Control unit of basic computer

with a 3x8 decoder. Bit 15 of the instruction is transferred to a flip-flop designated by the symbol I. The eight output of the decoder are designated by the symbols D_0 through D_7. 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 output of counter are decoded into 16 timing signals T_0 through T_{15}. 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 4x16 decoder. Once in a while, the counter is cleared to 0, causing the next timing signal to be T_0.

Ex: Consider the case where SC is incremented to provide timing signals, T_0, T_1, T_2, T_3 & T_4 in sequence. At time T_4, SC is cleared to 0 if decoder output D_3 is active. This is expressed symbolically by the statement

$$D_3 T_4 : SC \leftarrow 0$$

Timing diagram:— The timing diagram shows the time relationship of the control signals. The sequence counter SC responds to the positive transition of the clock. Initially, the CLR i/p of SC is active. The first positive transition of the clock clears SC to 0, which in turn activates the timing s/g T_0 out of the decoder. T_0 is active during one clock cycle. SC is incremented with every positive clock transition, unless its CLR i/p is active. This produces the sequence of timing s/g

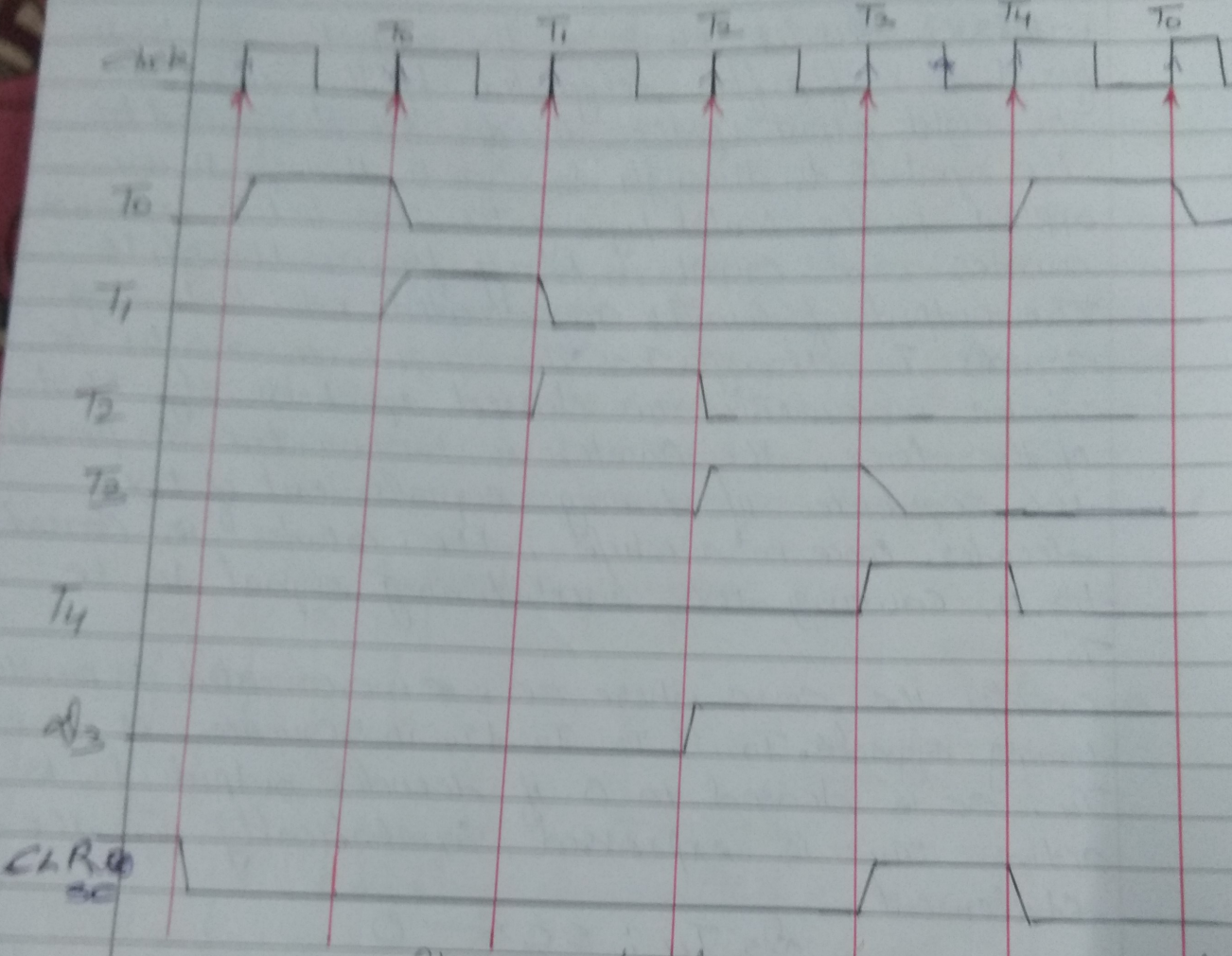


Fig:- example of control timing signal

T_0, T_1, T_2, T_3 & T_4 and so on. If SC is not cleared, the timing s/g will continue with T_0, T_1 up to T_4 & back to T_0. The last three waveform shows how SC is cleared when $D_3 T_4 = 1$. o/p D_3 from the operation decoder become active at the end of timing s/g T_2. When timing s/g T_4 becomes active, the output of the AND gate that implements the control function $D_3 T_4$ becomes active. This s/g is applied to the CLR input of SC. on the next positive clock transition the SC counter is cleared to 0. This cause the timing s/g T_0 to become active instead of T_5.

Instruction Cycle:—

- A program residing in the memory unit of the computer consists of a sequence of instructions. In the basic computer each instruction cycle consists of the following phases
1. Fetch an instruction from memory
2. Decode the instruction.
3. Read the effective address from memory if the instruction has an indirect address.
4. Execute the instruction.

- After step 4, the control goes back to step 1 to fetch, decode and execute the next instruction.
- This process continues unless a HALT instruction is encountered.

fetch and decode:- Initially the program counter is loaded with the address of the first instruction in the program. The sequence counter SC is cleared to 0, providing a decoded timing signal T_0. After each clock pulse, SC is incremented by one, so that the timing signal go through a sequence T_0, T_1, T_2 & so on. "The instruction fetch and decode phases are the same for all instructions. So the control functions & micro-operations will be independent of the instruction code."

all control inputs in the CPU during fetch and decode are functions of these three variables alone

T_0 : $AR \leftarrow PC$

T_1 : $IR \leftarrow M[AR]$, $PC \leftarrow PC+1$

T_2 : $D_0 \dots D_7 \leftarrow$ Decode $IR(12-14)$, $AR \leftarrow IR(0-11)$, $I \leftarrow IR(15)$

for every timing cycle, we assume $SC \leftarrow SC+1$ unless it is stated that $SC \leftarrow 0$.

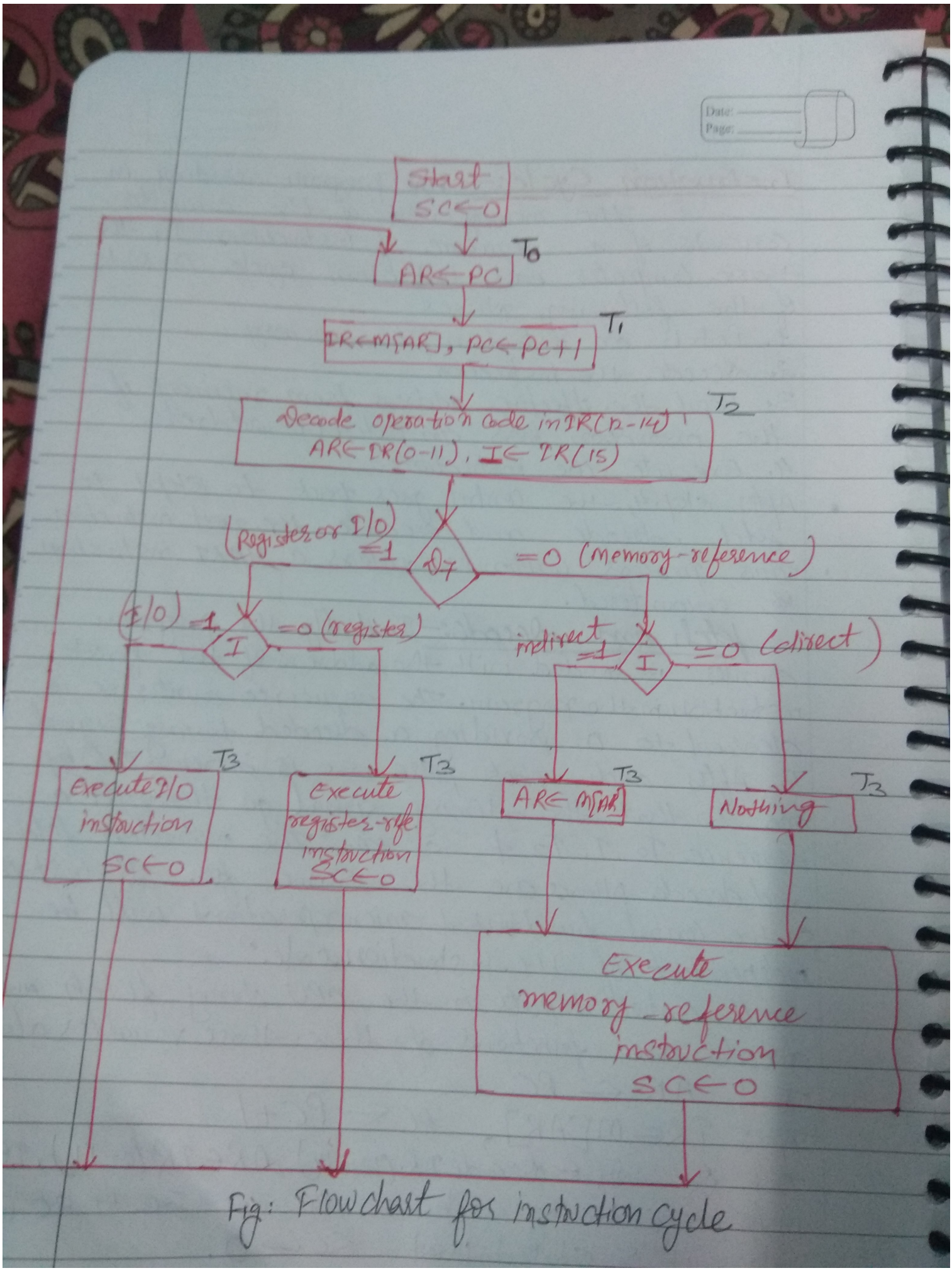


Fig: Flowchart for instruction cycle

Decoder output D_7 is equal to 1 if the operation code is equal to binary 111. if $D_7 = 1$, the instruction must be a register reference or input-output type. If $D_7 = 0$, the operation code must be one of the other seven values 000 through 110, specifying a memory-reference instruction. Control then inspect the value of the first bit of the instruction, which now available in flip-flop I.

The three instruction types are subdivided into four separate paths. The selected operation is activated with the clock transition associated with timing sig T_3. This can be symbolized as follows:-

1) $D_7' I T_3$: $AR \leftarrow M[AR]$
2) $D_7' I' T_3$: Nothing
3) $D_7 I' T_3$: execute a register reference instruction
4) $D_7 I T_3$: execute an input output instruction

1) if $D_7 = 0$ & $I = 1$, we have a memory reference instruction with an indirect address. It is necessary to read the effective address from memory. (sequence counter SC must be incremented) so that execution of the MRI can be continued with T_4.
2) with $I = 0$, it is not necessary to do anything since the effective address is already in AR.
3) A register reference or input output instruction can be executed with the clock associated with timing signal T_3. After the instruction is executed, SC is cleared to 0, & control returns to the fetch phase with $T_0 = 1$.