COMPUTER ORGANIZATION
MODULE-II
(Control Unit & Instruction Cycle)

Instruction Set Completeness:

- The set of instructions are said to be complete if the computer includes a sufficient number of instructions in each of the following categories:

1. Arithmetic, logical, and shift instructions
2. Instructions for moving information to and from memory and processor registers
3. Program control instructions together with instructions that check status conditions
4. Input and output instructions

Timing and Control:

- The timing for all registers 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 change the state of a register unless the register is enabled by a control signal.

- The control signals are generated in the control unit and provide control inputs for the multiplexers in the common bus, control inputs in processor registers, and microoperations for the accumulator.

- Control unit generates control signals using two organizations :
  - ✓ Hardwired Control Unit.
  - ✓ Micro-programmed Control Unit.

Hardwired Control Unit:

- Hardwired control units are implemented through use of sequential logic units or circuits like gates, flipflops, decoders in hardware.
- Hardwired control units are generally faster than micro-programmed designs.
- This architecture is preferred in reduced instruction set computers (RISC) as they use a simpler instruction set.

Advantages:

- Hardwired Control Unit is fast because control signals are generated by combinational circuits.
- The delay in generation of control signals depends upon the number of gates.
- The performances is high as compared to micro-programmed control unit.

Disadvantages:

- The control signals required by the CPU will be more complex.
- Modifications in control signal are very difficult. That means it requires rearranging of wires in the hardware circuit.
- It is difficult to correct mistake in original design or adding new features in existing design of control unit.

Micro-programmed Control Unit:

Advantages:

- The design of micro-program control unit is less complex because micro-programs are implemented using software routines.
- The micro-programmed control unit is more flexible because design modifications, correction and enhancement is easily possible.
- The new or modified instruction set of CPU can be easily implemented by simply rewriting or modifying the contents of control memory.
- The fault can be easily diagnosed in the micro-program control unit using diagnostics tools by maintaining the contents of flags, registers and counters.

Disadvantages:

- The micro-program control unit is slower than hardwired control unit. That means to execute an instruction in micro-program control unit requires more time.
- The micro-program control unit is expensive than hardwired control unit in case of limited hardware resources.
- The design duration of micro-program control unit is more than hardwired control unit for smaller CPU.

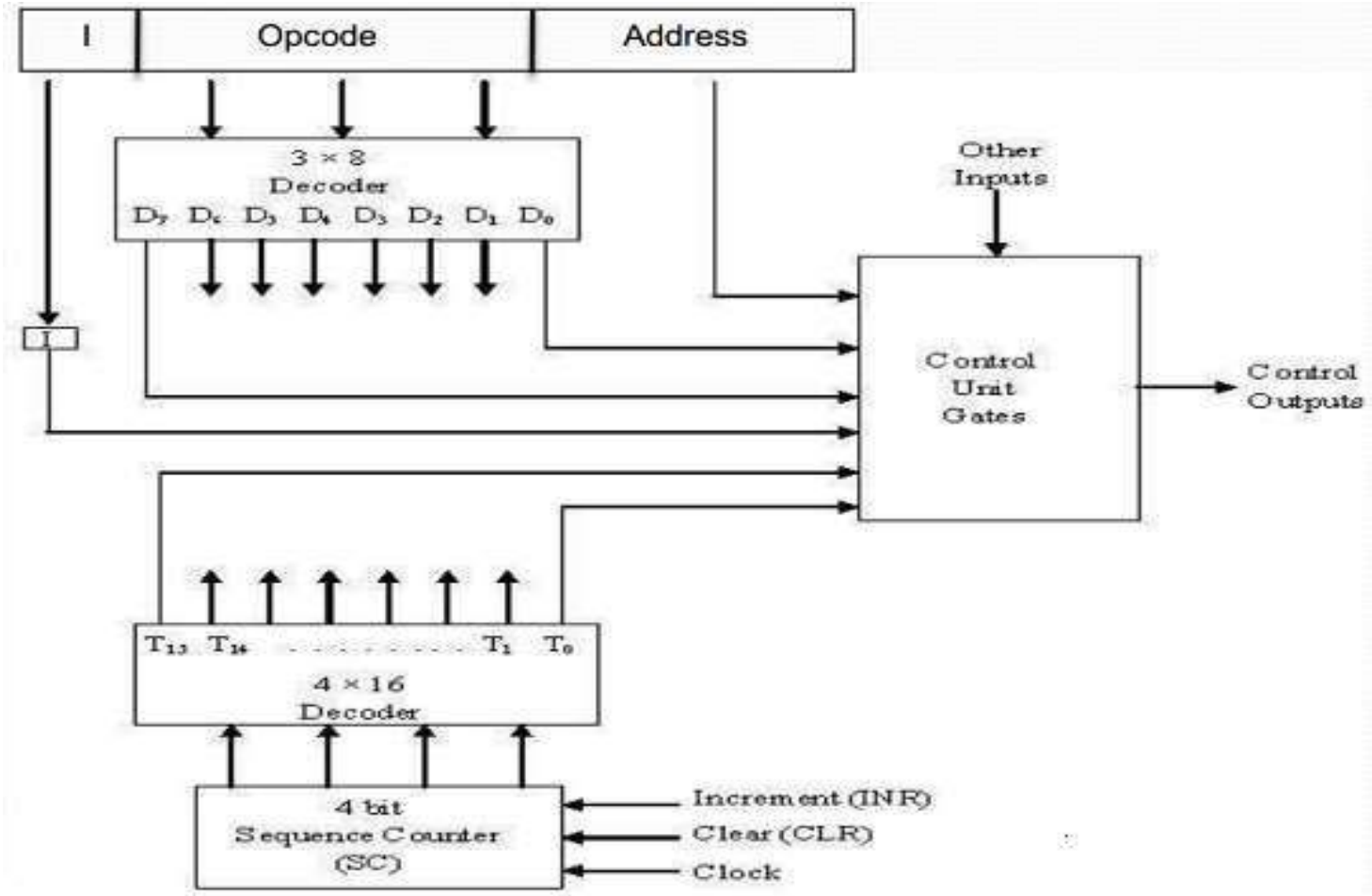
Comparison between Hardwired and Microprogrammed Control Unit:

ATTRIBUTE	HARDWIRED CONTROL UNIT	MICRO-PROGRAMMED CONTROL UNIT
SPEED	FAST	SLOW
COST OF IMPLEMENTATION	MORE	CHEAPER
FLEXIBILITY	NOT FLEXIBLE, DIFFICULT TO MODIFY FOR NEW INSTRUCTION	FLEXIBLE, NEW INSTRUCTIONS CAN BE ADDED
ABILITY TO HANDLE COMPLEX INSTRUCTION	DIFFICULT	EASIER
DECODING	COMPLEX	EASY
APPLICATION	RISC MICROPROCESSOR	CISC MICROPROCESSOR
INSTRUCTION SET SIZE	SMALL	LARGE
CONTROL MEMORY	ABSENT	PRESENT
CHIP AREA REQUIRED	LESS	MORE

Control Unit Architecture:

A control unit consist of:

- ❖ Instruction Register
- ❖ Number of Control Logic Gates
- ❖ Two Decoders
- ❖ 4-bit Sequence Counter



Control Unit:

- An instruction read from memory is placed in the instruction register (IR).
- The instruction register is divided into three parts: the I bit, operation code, and address part.
- First 12-bits (0-11) to specify an address, next 3-bits specify the operation code (opcode) field of the instruction and last left most bit specify the addressing mode I.
- I = 0 for direct address
- I = 1 for indirect address

- First 12-bits (0-11) are applied to the control logic gates.
- The operation code bits (12 – 14) are decoded with a 3 x 8 decoder.
- The eight outputs (D_0 through D_7) from a decoder goes to the control logic gates to perform specific operation.
- Last bit 15 is transferred to a I flip-flop designated by symbol I.

Control Unit:

- The 4-bit sequence counter SC can count in binary from 0 through 15.
- The counter output is decoded into 16 timing pulses T_0 through T_{15}.
- The sequence counter can be incremented by INR.
- Input or clear by CLR input synchronously.

- **Example:**

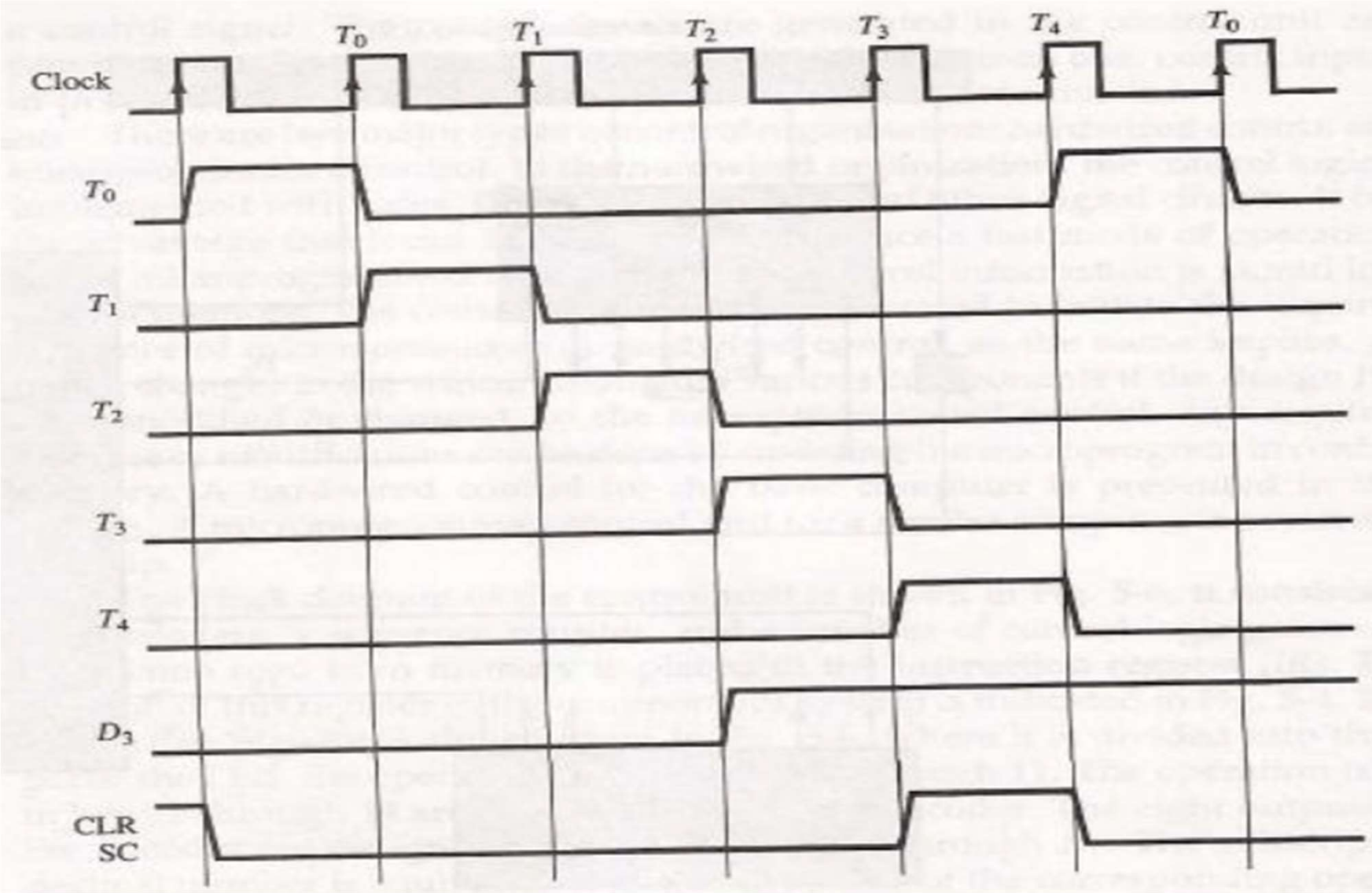
Consider the case where SC is incremented to provide timing signals T_0, T_1, T_2 , T_3, and T_4 in sequence. At time T_4 , SC is cleared to 0 if decoder output D_3 is active. This is expressed symbolically by the statement:

$$D_3\, T_4 : SC \leftarrow 0$$

- Output D_3 from the operation decoder becomes active at the end of timing signal T_2.
- SC is cleared when $D_3T_4 = 1$

Timing Diagram:

- The timing diagram shows the time relationship of the control signals.

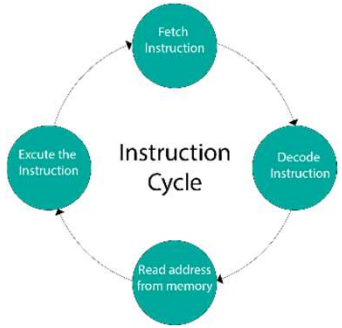


Instruction Cycle:

- A program residing in the memory unit of a computer consists of a sequence of instructions. These instructions are executed by the processor by going through a cycle for each instruction.

- In a basic computer, each instruction cycle consists of the following phases:

1. Fetch 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.



(This process continues indefinitely unless a HALT instruction is encountered)

- Instruction cycle is also defined as the combination of two cycles i.e. Fetch cycle (FC) and Execution cycle (EC).
- An ISA (Instruction set architecture) is a medium whereby a processor communicates with the human programmer.
- ISA acts as an interface between hardware & software and is divided into 2 parts:

RISC (Reduced Instruction Set Computer**)**

CISC (Complex Instruction Set Computer**)**

RISC	CISC
❖ It focuses on software	✓ It focuses on hardware
❖ Small number of instructions	✓ Large number of instructions
❖ Less addressing modes	✓ More addressing modes
❖ The program written for RISC architecture needs to take more space in memory(RAM)	✓ Needs less space in RAM
❖It has fixed instruction format	✓ Variable instruction format
❖ Low power consumption ❖ Uses more number registers ❖ Pipelining is easy ❖ It executes one instruction per clock cycle ❖ The execution time of RISC is very short. ❖ Ex: Alpha, ARC ARM, AVR, PIC	✓ High power consumption ✓ Less number of registers ✓ Pipelining is difficult ✓ It executes one instruction with more clock cycles ✓ The execution time of CISC is more. ✓ Ex: Motorola 6800 family, Intel x86 CPUs.

Fetch and Decode:

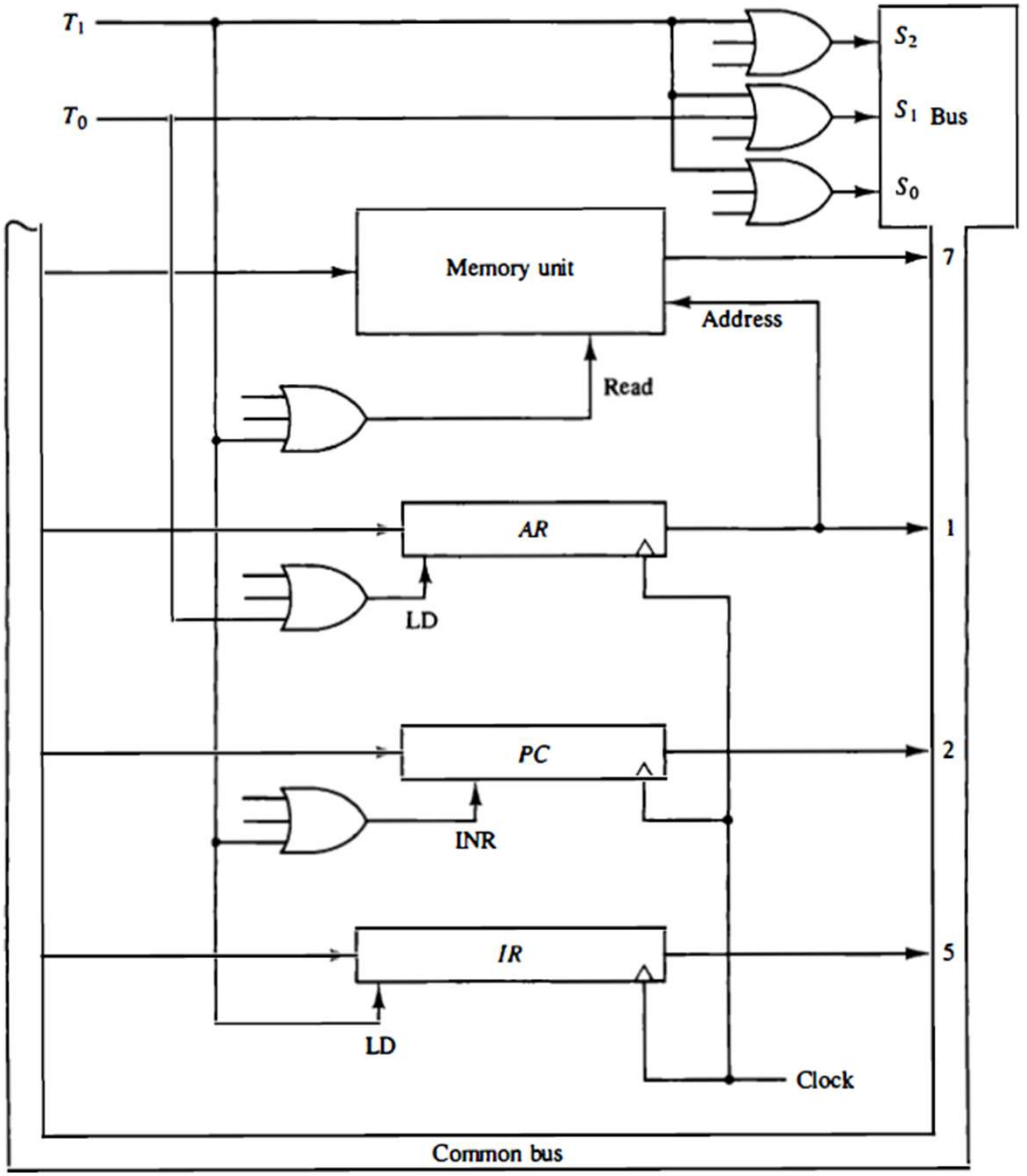
- Initially, the program counter PC is loaded with the address of the first instruction in the program. The sequence counter SC is cleared to 0, providing a decoded timing signal T_0. After each clock pulse, SC is incremented by one, so that the timing signals go through a sequence T_0, T_1, T_2, and so on.
- The microoperations for the fetch and decode phases can be specified by the following register transfer statements.

T_0: AR ← PC
T_1: IR ← M[AR], PC ← PC + 1
T_2: D_0, ..., D_7 ← Decode IR(12-14), AR ← IR(0-11), I ← IR(l5)

- Since only AR is connected to the address inputs of memory, it is necessary to transfer the address from PC to AR during the clock transition associated with timing signal T_0. The instruction read from memory is then placed in the instruction register IR with the clock transition associated with timing signal T_1. At the same time, PC is incremented by one to prepare it for the address of the next instruction in the program. At time T_2, the operation code in IR is decoded, the indirect bit is transferred to flip-flop I, and the address part of the instruction is transferred to AR . Note that SC is incremented after each clock pulse to produce the sequence T_0, T_1, and T_2.

Fetch and Decode:

- Below Figure shows how the first two register transfer statements are implemented in the bus system.



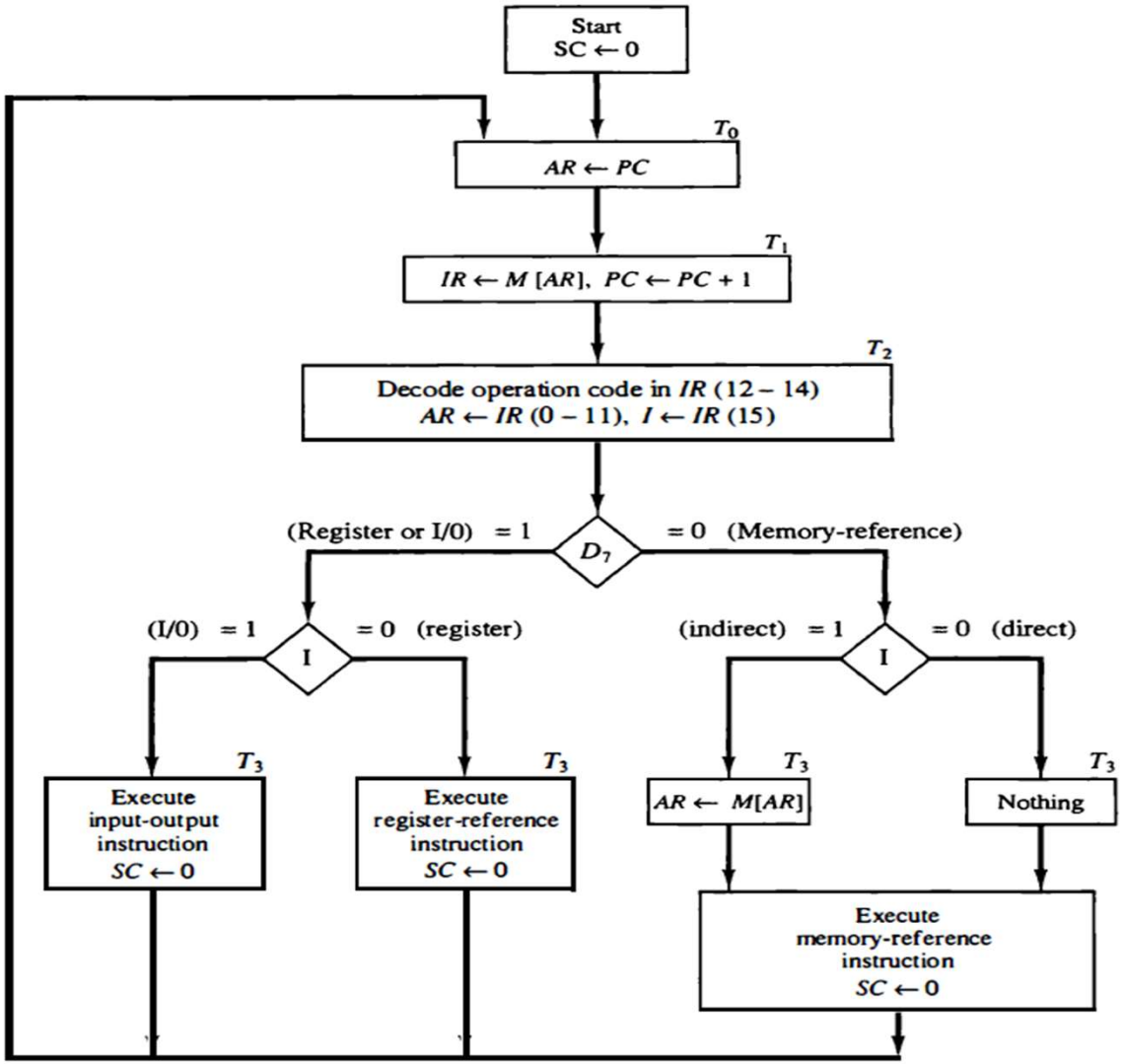
Fetch and Decode:

- To provide the data path for the transfer of PC to AR we must apply timing signal T_0 to achieve the following connection:
    1. Place the content of PC onto the bus by making the bus selection inputs $S_2S_1S_0$ equal to 010.
    2. Transfer the content of the bus to AR by enabling the LD input of AR.
- The next clock transition initiates the transfer from PC to AR since $T_0 = 1$. In order to implement the second statement

  T_1: IR ← M[AR], PC ← PC + 1
- It is necessary to use timing signal T_1 to provide the following connections in the bus system.
    1. Enable the read input of memory.
    2. Place the content of memory onto the bus by making $S_2S_1S_0 = 111$.
    3. Transfer the content of the bus to IR by enabling the LD input of IR.
    4. Increment PC by enabling the INR input of PC.
- The next clock transition initiates the read and increment operations since $T_1 = 1$.
- The figure duplicates a portion of the bus system and shows how T_0 and T_1 are connected to the control inputs of the registers, the memory, and the bus selection inputs. Multiple input OR gates are included in the diagram because there are other control functions that will initiate similar operations.

Determine the Type of Instruction:

- The timing signal that is active after the decoding is T_3. During time T_3, the control unit determines the type of instruction that was just read from memory. The below flowchart presents an initial configuration for the instruction cycle and shows how the control determines the instruction type after the decoding.



Determine the Type of Instruction:

- Decoder output $D_7 = 1$ if the operation code is equal to binary 111. From instruction code format we determine that 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 inspects the value of the first bit of the instruction, which is now available in flip-flop I. If $D_7 = 0$ and $I = 1$, we have a memory reference instruction with an indirect address. It is then necessary to read the effective address from memory. The microoperation for the indirect address condition can be symbolized by the register transfer statement.

$$AR \leftarrow M[AR]$$

- Initially, AR holds the address part of the instruction. This address is used during the memory read operation. The word at the address given by AR is read from memory and placed on the common bus. The LD input of AR is then enabled to receive the indirect address that resided in the 12 least significant bits of the memory word.

Determine the Type of Instruction:

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

$D'_7 IT_3$: $AR \leftarrow M[AR]$
$D'_7 I'T_3$: Nothing
$D_7 I'T_3$: Execute a register-reference instruction
$D_7 IT_3$: Execute an input–output instruction

- When a memory-reference instruction with I = 0 is encountered, it is not necessary to do anything since the effective address is already in AR. However, the sequence counter SC must be incremented when $D'_7T_3 = 1$, so that the execution of the memory-reference instruction can be continued with timing variable T_4. 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 and control returns to the fetch phase with $T_0 = 1$.

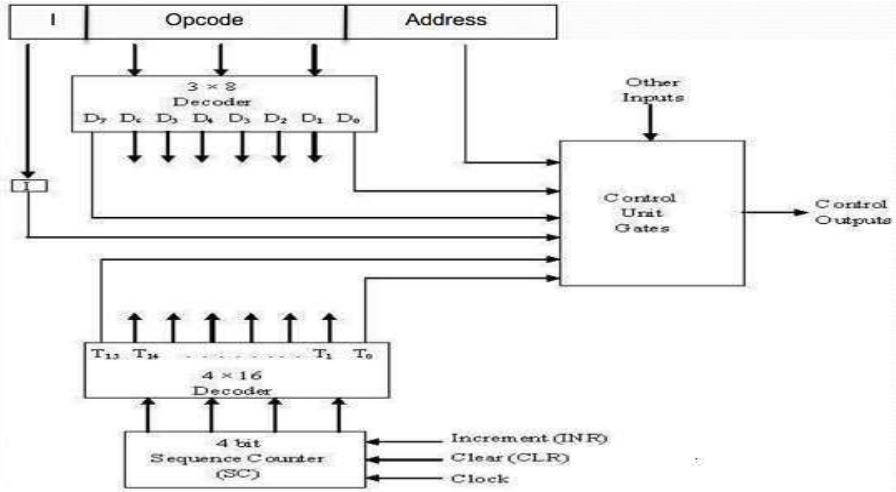
- Note that the sequence counter SC is either incremented or cleared to 0 with every positive clock transition. We will adopt the convention that if SC is incremented, we will not write the statement SC ← SC+1, but it will be implied that the control goes to the next timing signal in sequence. When SC is to be cleared, we will include the statement SC ← 0.

Register-Reference Instructions:

- The control functions and microoperations for the register-reference instructions and memory-reference are listed in below Table.

$D_7I'T_3 = r$ (common to all register-reference instructions)
$IR(i) = B_i$ [bit in $IR(0–11)$ that specifies the operation]

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



- For example, the instruction CLA has the hexadecimal code 7800, which gives the binary equivalent 0111 1000 0000 0000.
- The first bit is a zero and is equivalent to I'.
- The next three bits constitute the operation code and are recognized from decoder output D_7.
- Bit 11 in IR is 1 and is recognized from B_{11}. The control function that initiates the microoperation for this instruction is $D_7I'T_3\ B_{11} = rB_{11}$.
- The execution of a register-reference instruction is completed at time T_3.
- The sequence counter SC is cleared to 0 and the control goes back to fetch the next instruction with timing signal T_0.

Memory-Reference Instructions:

- The decoded output D_i for i = 0, 1, 2, 3, 4, 5, and 6 from the operation decoder that belongs to each instruction is included in the below table.
- The effective address of the instruction is in the address register AR and was placed there during timing signal T_2 when I = 0, or during timing signal T_3 when I = 1.
- The execution of the memory-reference instructions starts with timing signal T4.
- The symbolic description of each instruction is specified in the table in terms of register transfer notation.

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

Microoperations that execute this instruction are:

- **AND to AC:**

  T_0: AR ← PC
  T_1: IR ← M[AR], PC ← PC + 1
  T_2: D_0, ..., D_7 ← Decode IR(12-14), AR ← IR(0-11), I ← IR(l5)
  D_0T_4: DR ← M [AR]
  D_0T_5: AC ← AC /\ DR, SC ← 0

- **ADD to AC:**

  T_0: AR ← PC
  T_1: IR ← M[AR], PC ← PC + 1
  T_2: D_0, ..., D_7 ← Decode IR(12-14), AR ← IR(0-11), I ← IR(l5)
  D_1T_4: DR ← M [AR]
  D_1T_5: AC ← AC + DR, E ← C_{out}, SC ← 0

- **LDA: Load to AC:**

  T_0: AR ← PC
  T_1: IR ← M[AR], PC ← PC + 1
  T_2: D_0, ..., D_7 ← Decode IR(12-14), AR ← IR(0-11), I ← IR(l5)
  D_2T_4: DR ← M [AR]
  D_2T_5: AC ← DR, SC ← 0

Microoperations that execute this instruction are:

- **STA: Store AC:**

  T_0: AR ← PC
  T_1: IR ← M[AR], PC ← PC + 1
  T_2: D_0, ..., D_7 ← Decode IR(12-14), AR ← IR(0-11), I ← IR(l5)
  D_3T_4: M [AR] ← AC, SC ← 0

- **BUN: Branch Unconditionally:**

  T_0: AR ← PC
  T_1: IR ← M[AR], PC ← PC + 1
  T_2: D_0, ..., D_7 ← Decode IR(12-14), AR ← IR(0-11), I ← IR(l5)
  D_4T_4: PC ← AR, SC ← 0

- **BSA: Branch and Save Return Address:**

  T_0: AR ← PC
  T_1: IR ← M[AR], PC ← PC + 1
  T_2: D_0, ..., D_7 ← Decode IR(12-14), AR ← IR(0-11), I ← IR(l5)
  D_5T_4: M[AR] ← PC, AR ←AR + 1
  D_5T_5: PC ←AR, SC ← 0

ISZ: Increment and Skip if Zero:

  T_0: AR ← PC
  T_1: IR ← M[AR], PC ← PC + 1
  T_2: D_0, ..., D_7 ← Decode IR(12-14), AR ← IR(0-11), I ← IR(l5)
  D_6T_4: DR ←M [AR]
  D_6T_5: DR ←DR + 1
  D_6T_6: M [AR] ←DR, if (DR = 0) then (PC ←PC + 1), SC ←0

Difference between Horizontal and Vertical micro-programmed Control Unit:

S. No	Horizontal micro-programmed CU	Vertical micro-programmed CU
1.	It supports longer control word.	It supports shorter control word.
2.	It allows a higher degree of parallelism. If degree is n, then n Control Signals are enabled at a time.	It allows a low degree of parallelism i.e., the degree of parallelism is either 0 or 1.
3.	No additional hardware is required.	Additional hardware in the form of decoders is required to generate control signals.
4.	It is faster than a Vertical micro-programmed control unit.	It is slower than a Horizontal micro-programmed control unit.
5.	It is more flexible than a vertical micro-programmed control unit.	It is less flexible than horizontal but more flexible than that of a hardwired control unit.
6.	A horizontal micro-programmed control unit uses horizontal micro-instruction, where every bit in the control field attaches to a control line.	A vertical micro-programmed control unit uses vertical micro-instruction, where a code is used for each action to be performed and the decoder translates this code into individual control signals.
7.	The horizontal micro-programmed control unit makes less use of ROM encoding than the vertical micro-programmed control unit.	The vertical micro-programmed control unit makes more use of ROM encoding to reduce the length of the control word.

THANK YOU