COMPUTER ORGANIZATION

B.TECH III SEM

CSE-4

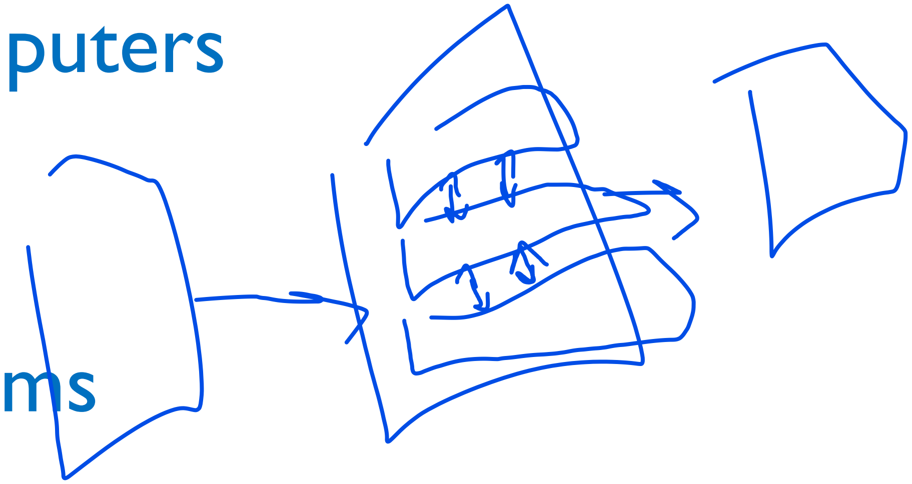
N SANTOSHI
ASSISTANT PROFESSOR
DEPARTMENT OF ECE

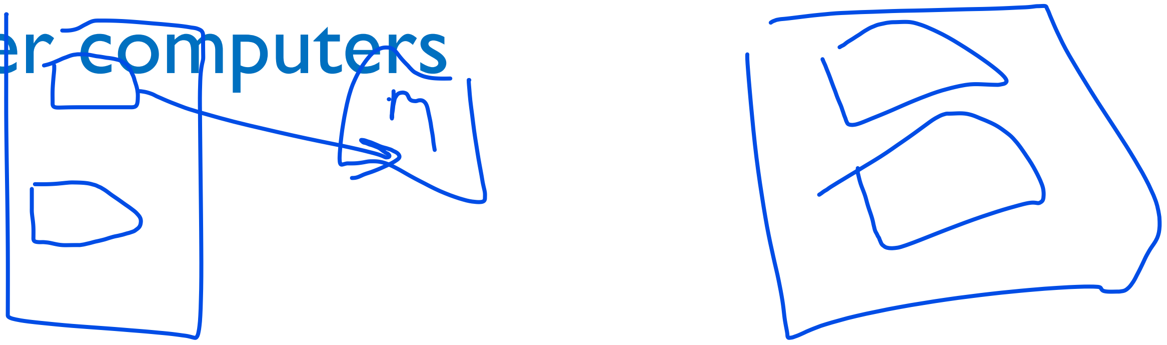
INTRODUCTION

- Definition of a computer: A computer can be defined as a fast electronic calculating machine that accepts the (data) digitized input information process it as per the list of internally stored instructions and produces the resulting information.
- List of instructions are called programs & internal storage is called computer memory.

The different types of computers are

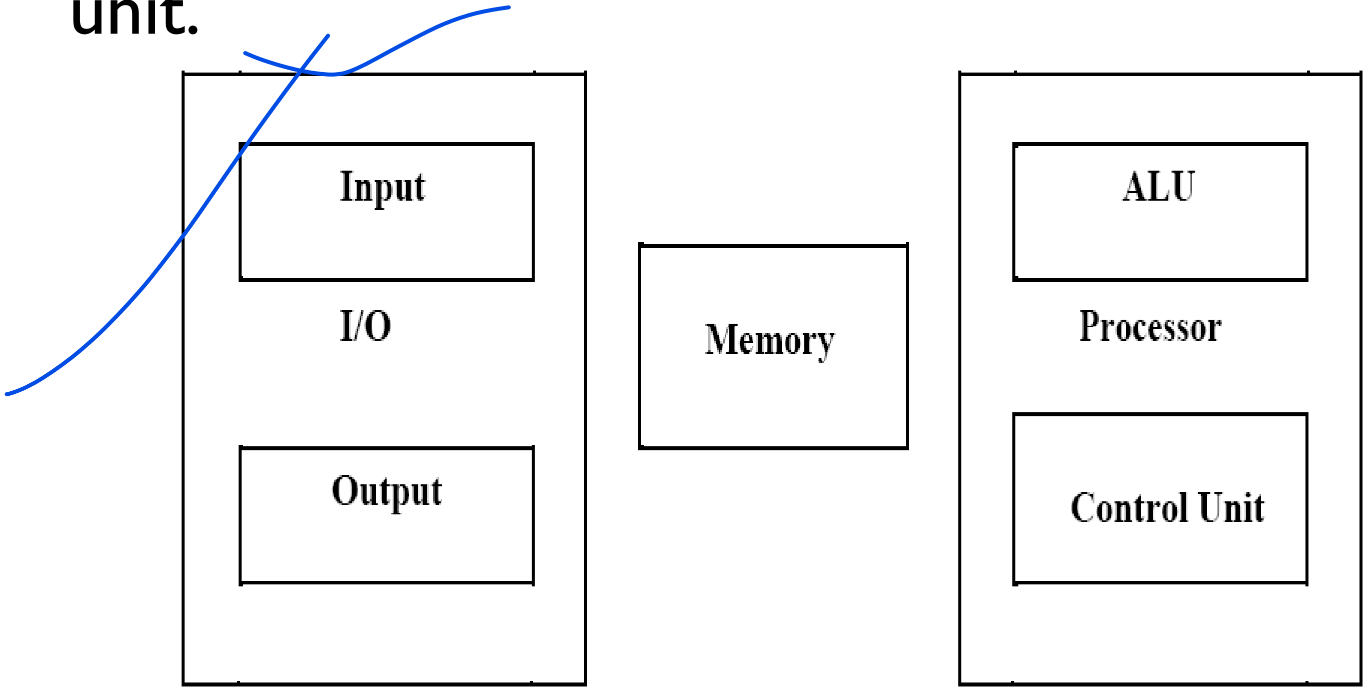
- Personal computers
- Note book computers
- Work stations
- Enterprise systems
- Super computers

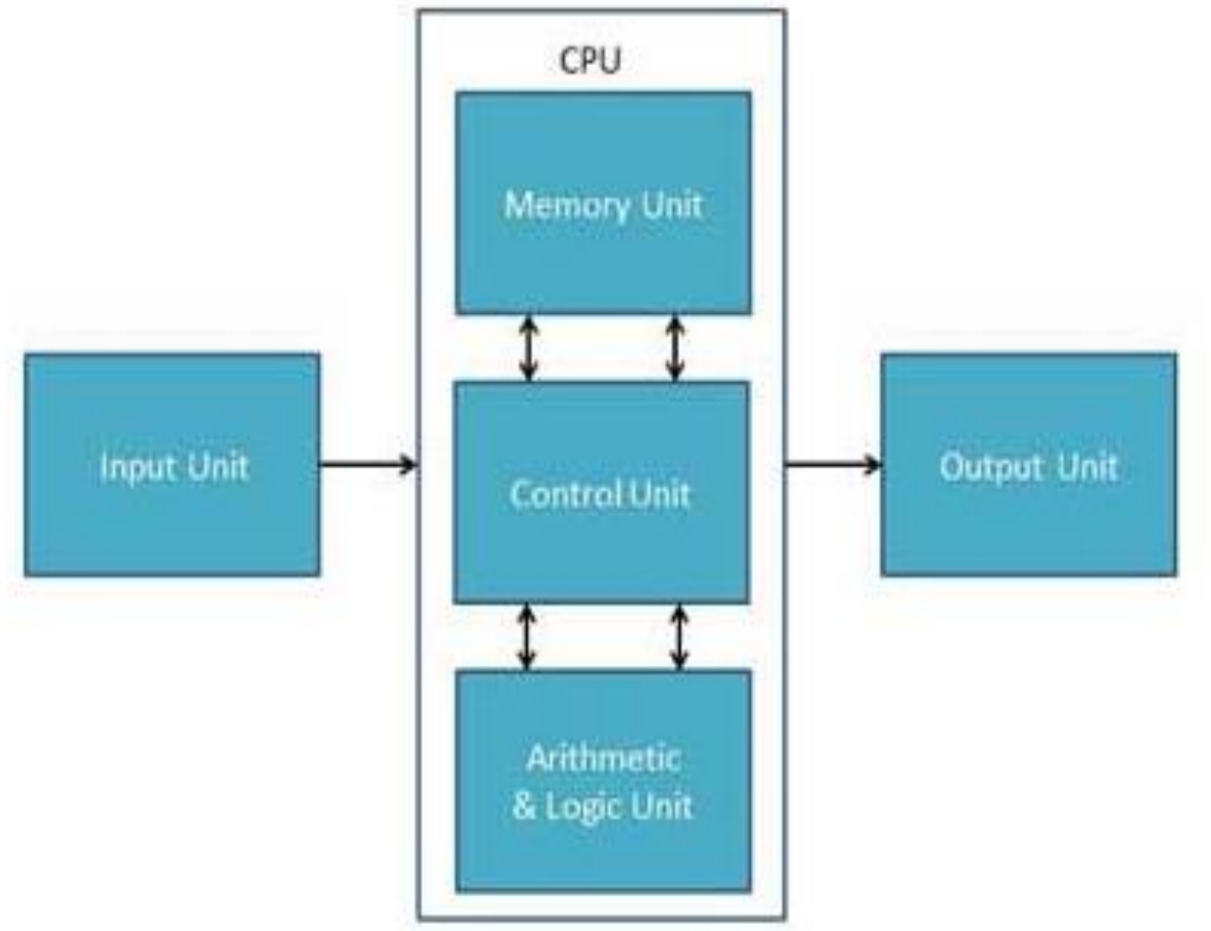




Functional units

A computer consists of five functionally independent main parts: input, memory, arithmetic logic unit (ALU), output and control unit.




CPU
Memory Unit
Input Unit
Control Unit
Output Unit
Arithmetic & Logic Unit

Central processing unit

- CPU is considered as the brain of the computer.
- CPU has following three components.
  - ALU(Arithmetic Logic Unit)
  - Control Unit
  - Memory or Storage Unit

Arithmetic logic unit (ALU)

- Most of the computer operators are executed in ALU of the processor like addition, subtraction, division, multiplication, comparing etc.
- The operands are brought into the ALU from memory and stored in high speed storage elements called register.

Control Unit

→ I ∂ [illegible]

- This unit controls the operations of all parts of the computer but does not carry out any actual data processing operations.

- This unit controls the operations of all parts of the computer but does not carry out any actual data processing operations.

Memory unit

- Its function is to store programs and data. It is basically of two types
  - Primary memory
  - Secondary memory

  - Primary memory is computer memory that is accessed directly by the CPU.
    - This includes several types of memory, such as the processor cache and ROM,RAM.

- Memory in which any location can be reached in a short and fixed amount of time after specifying its address is called random-access memory (RAM)
- The time required to access one word is called memory access time.
- Memory which is only readable by the user and contents of which can't be altered is called read only memory (ROM). It contains operating system

Secondary memory

- It is used where large amount of data & programs have to be stored, particularly information that is accessed infrequently.
- Examples: - Magnetic disks & tapes, optical disks (ie CD-ROM's), floppies etc.,

Input units

- Joysticks
- Trackballs
- mouse
- scanners etc are other input devices

Output units

- sends the processed results to the outside world
  - Examples:- Printer, speakers, monitor etc.

MONITOR
PRINTER
SPEAKER
HEADPHONE
PROJECTOR

INTRODUCTION

- ORGANIZATION AND ARCHITECTURE
  - Computer Architecture refers to those attributes of a system that have a direct impact on the logical execution of a program.
    - Examples:
      - the instruction set
      - the number of bits used to represent various data types
      - I/O mechanisms
      - memory addressing techniques

Reference: W. Stallings

- Computer Organization refers to the operational units and their interconnections that realize the architectural specifications.
  - Examples are things that are transparent to the programmer:
    - control signals
    - interfaces between computer and peripherals
    - the memory technology being used

Reference: W. Stallings

Structure and Function

- Structure is the way in which components relate to each other

- Function is the operation of individual components as part of the structure

Reference: W. Stallings

Function

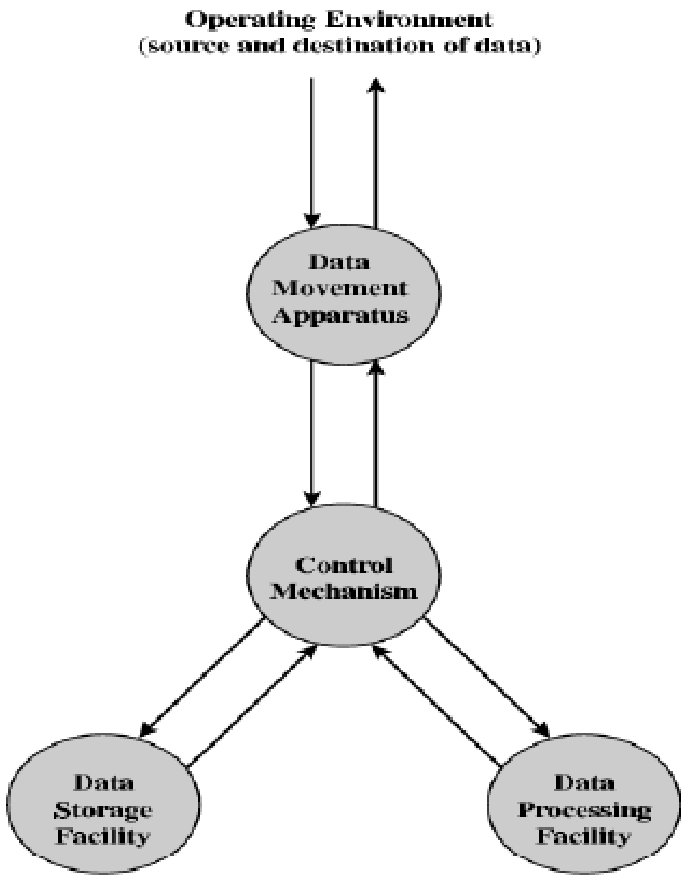


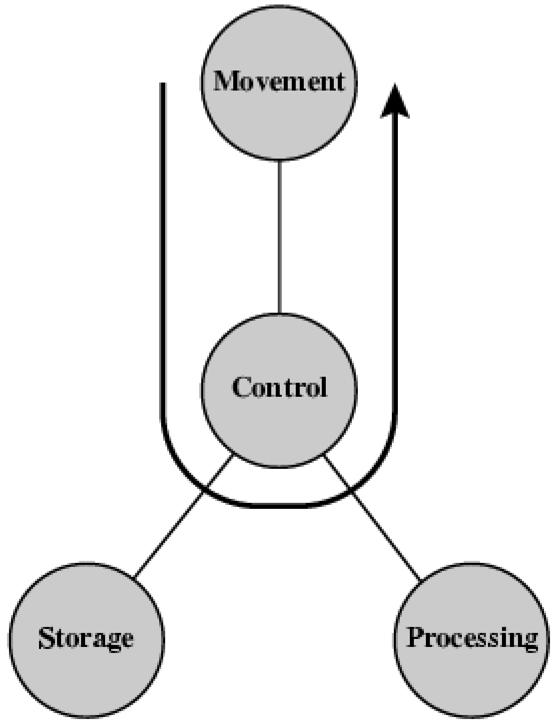
Fig: Functional view of a computer

All computer functions are

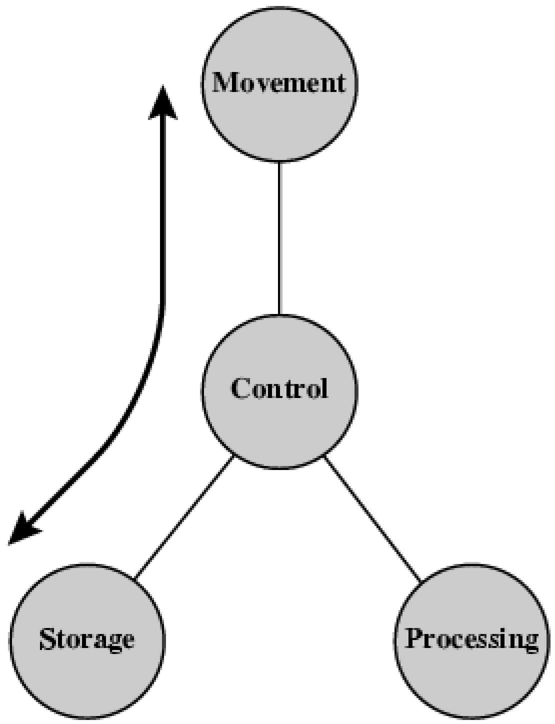
- Data processing: Computer must be able to process data which may take a wide variety of forms and the range of processing.
- Data storage: Computer stores data either temporarily or permanently.
- Data movement: Computer must be able to move data between itself and the outside world.
- Control: There must be a control of the above three functions.

Reference: W. Stallings

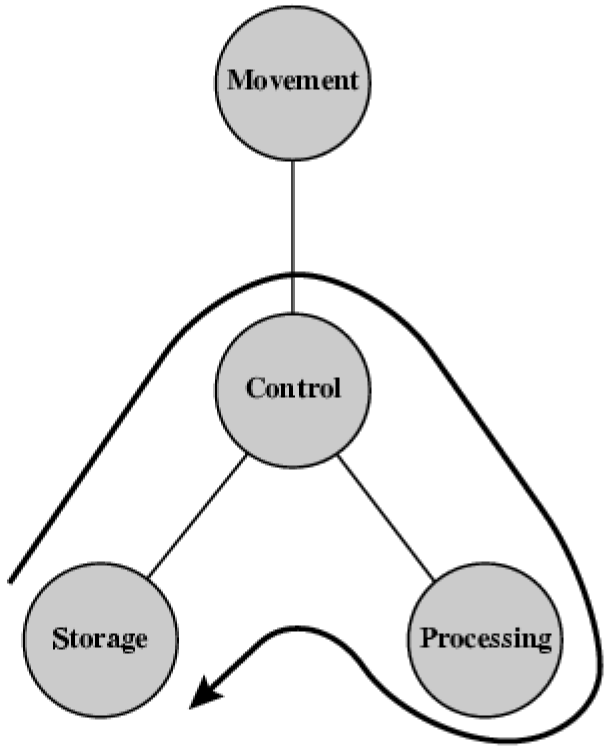
Data movement operation



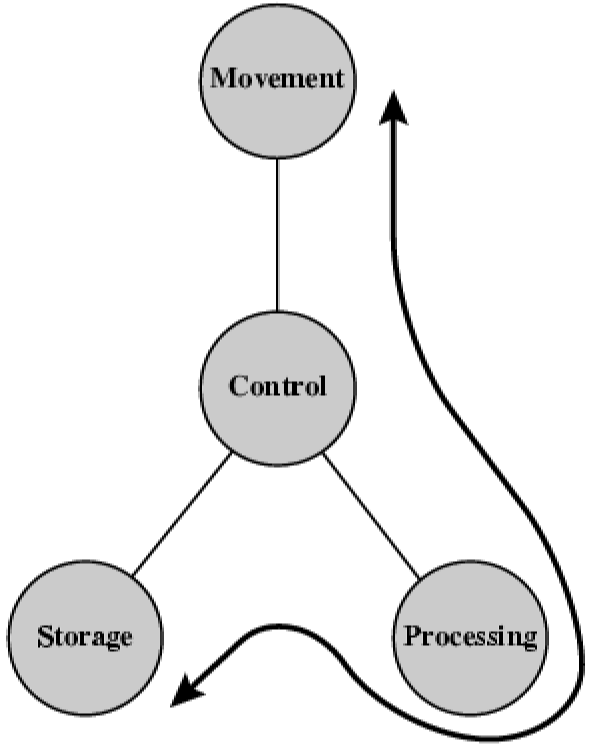
Storage operation



Processing from / to storage



Processing from storage to i/o

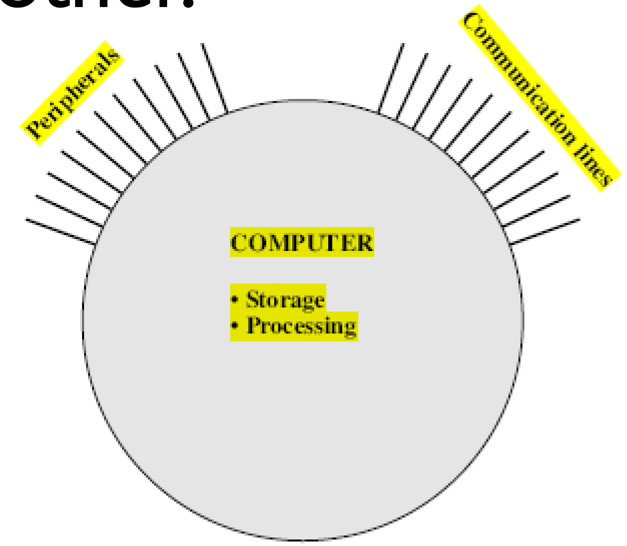


System bus

- Data bus- It defines the bit length of the processor.eg: 8bit processor have 8 bit data bus.
- Address bus- It defines the max memory capacity of the processor.eg:8bit address means memory capacity=2^8=256 bytes
- Control bus- The physical connections that carry **control** information between the CPU and other devices within the computer

Structure

•Structure is the way in which components relate to each other.



- Four main structural components:
  - ✓ Central processing unit (CPU)
  - ✓ Main memory
  - ✓ I / O
  - ✓ System interconnections

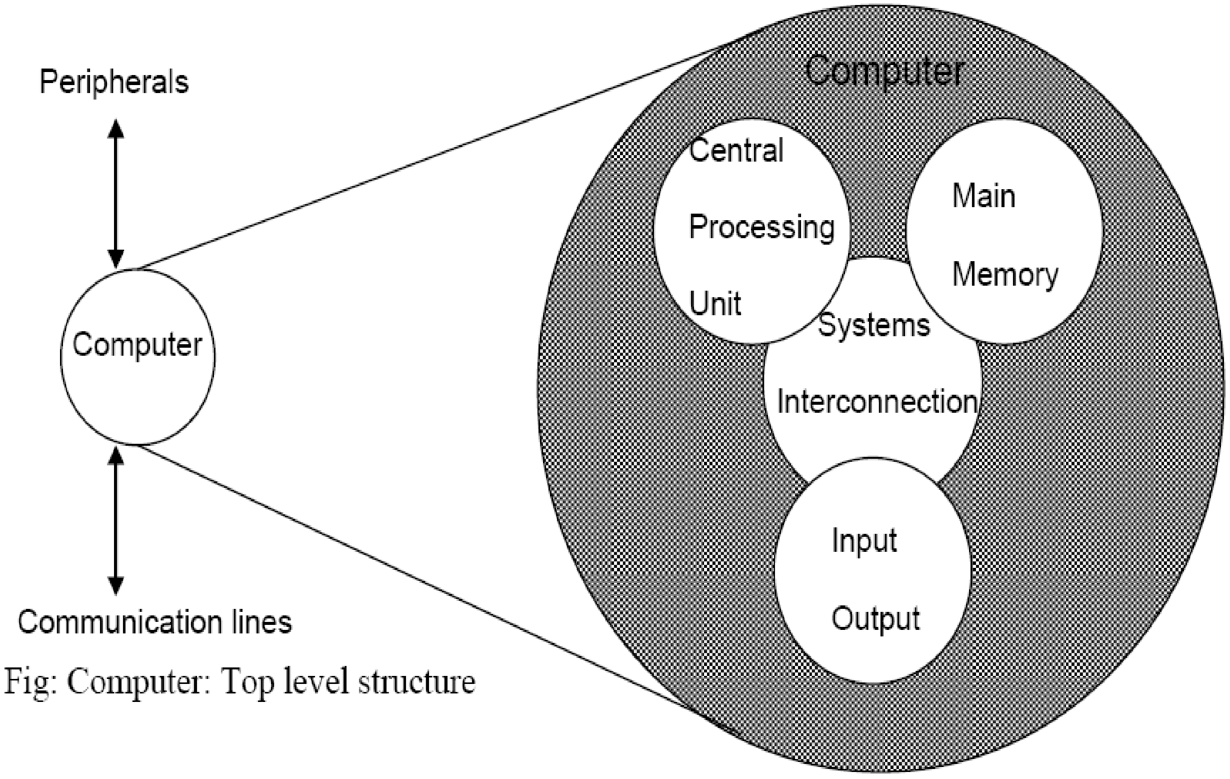


Fig: Computer: Top level structure

- CPU structural components:
  - Control unit
  - Arithmetic and logic unit (ALU)
  - Registers
  - CPU interconnections

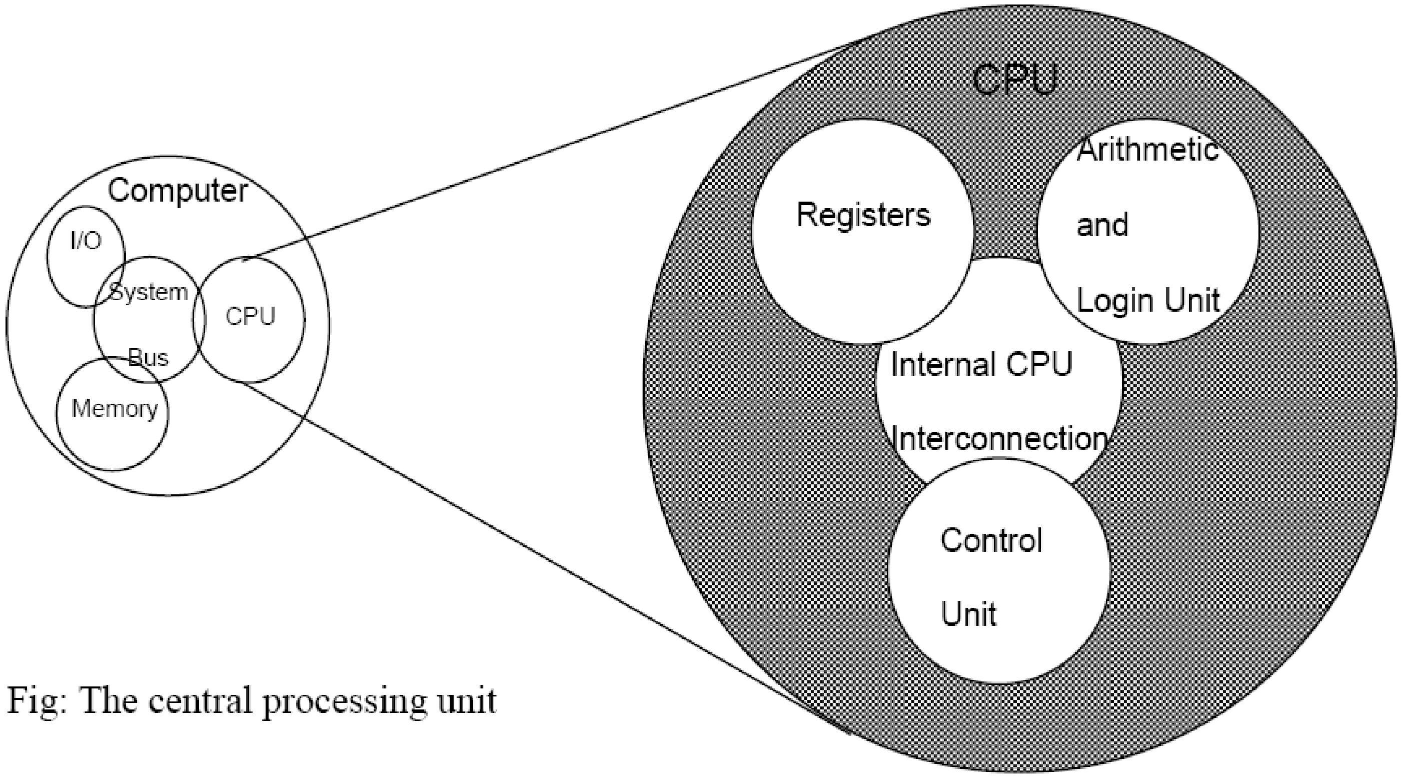


Fig: The central processing unit

- Control unit structural components:
  - Sequencing logic
  - Control unit registers and decoders
  - Control memory

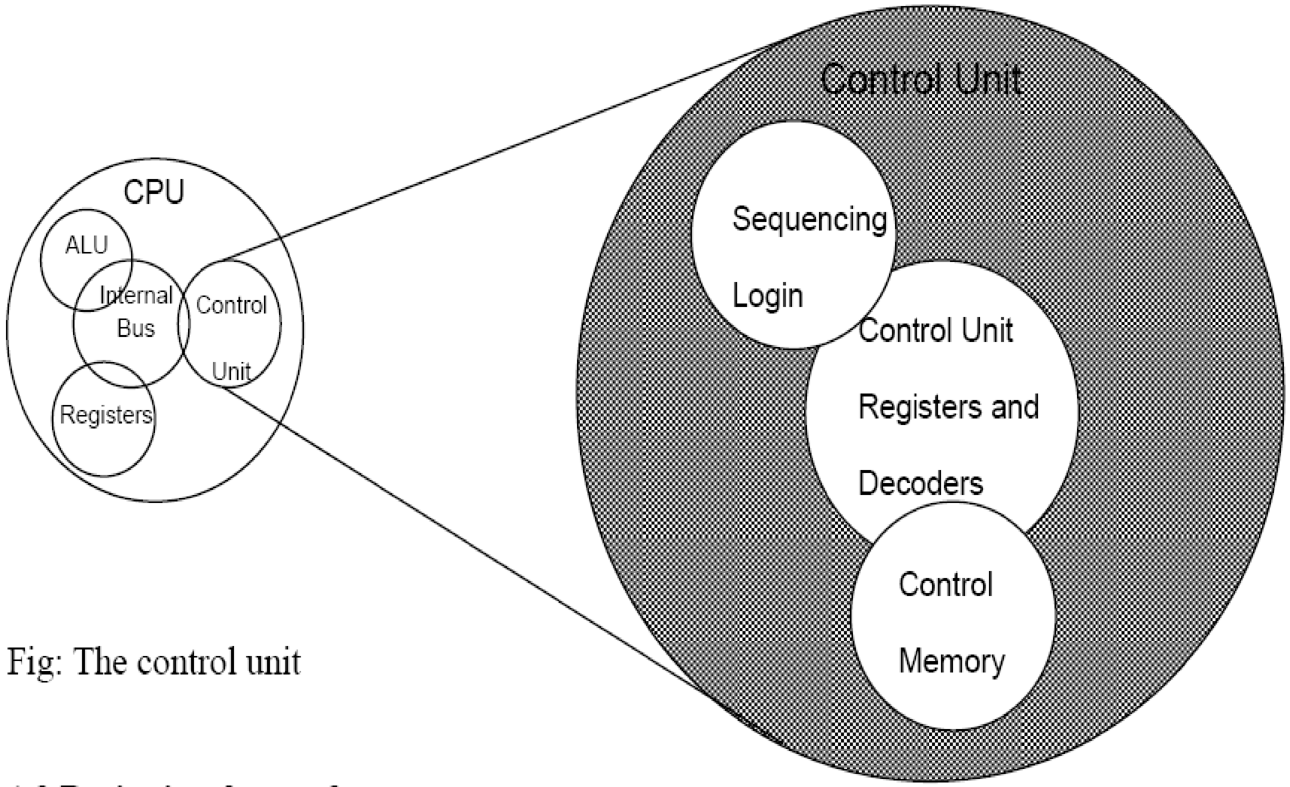


Fig: The control unit

Input output subsystems

- I/O subsystems controls all I/O devices.
- The I/O subsystems provide the mechanism for communication between the CPU and the outside world(I/O devices).
- Basic functions include:
  - Issuing commands to the devices.
  - Handling interrupts from devices.
  - Responding errors from devices.

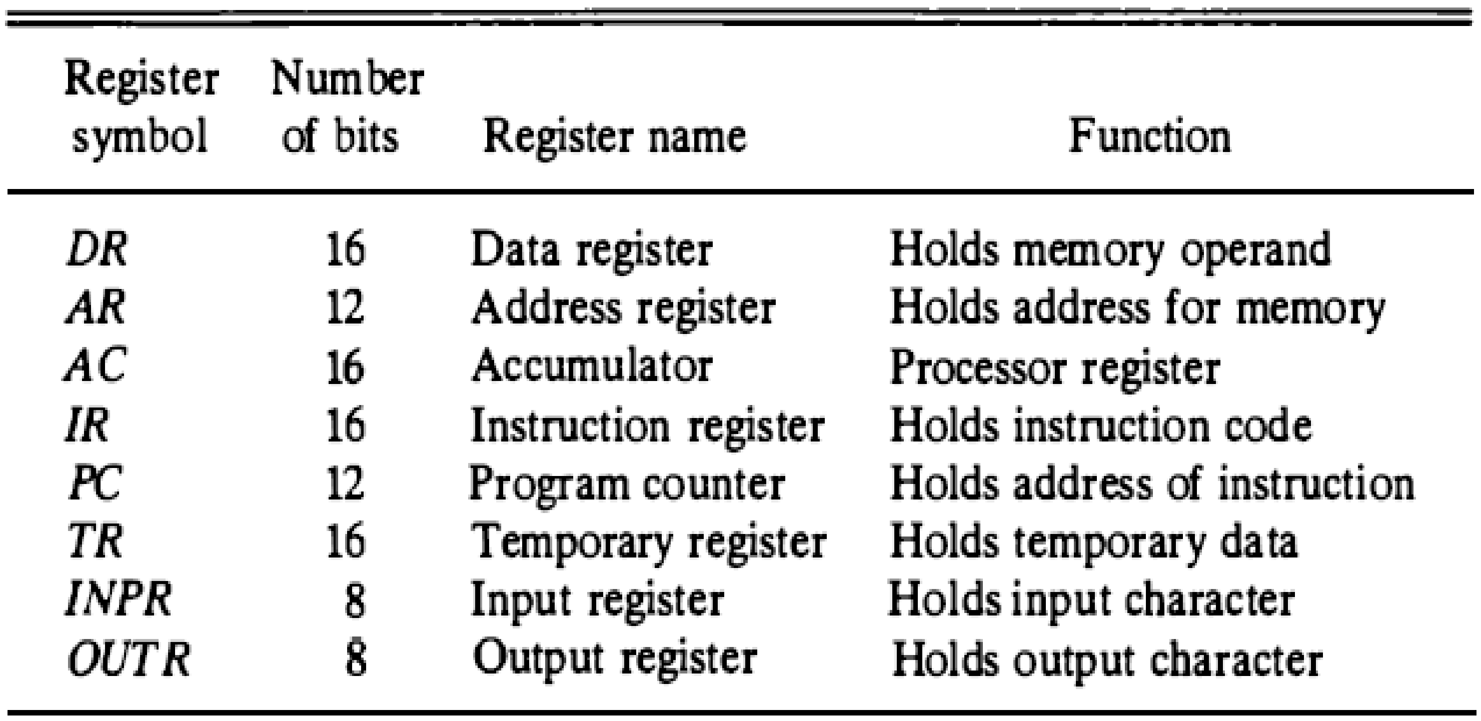
Computer Components

- The Control Unit (CU) and the Arithmetic and Logic Unit (ALU) constitute the Central Processing Unit (CPU)
- Data and instructions need to get into the system and results need to get out
  - Input/output (I/O module)
- Temporary storage of code and results is needed
  - Main memory (RAM)

What is a Register?

- A register is a very small amount of very fast memory that is built into the CPU (central processing unit).
- Contents can be accessed at extremely high speeds.
- Registers are used to store data temporarily during the execution of a program.
- Different processors have different register sizes.
- Registers are normally measured by the number of bits they can hold, for example, an 8-bit register means it can store 8 bits of data or a 32-bit register means it can store 32 bit of data

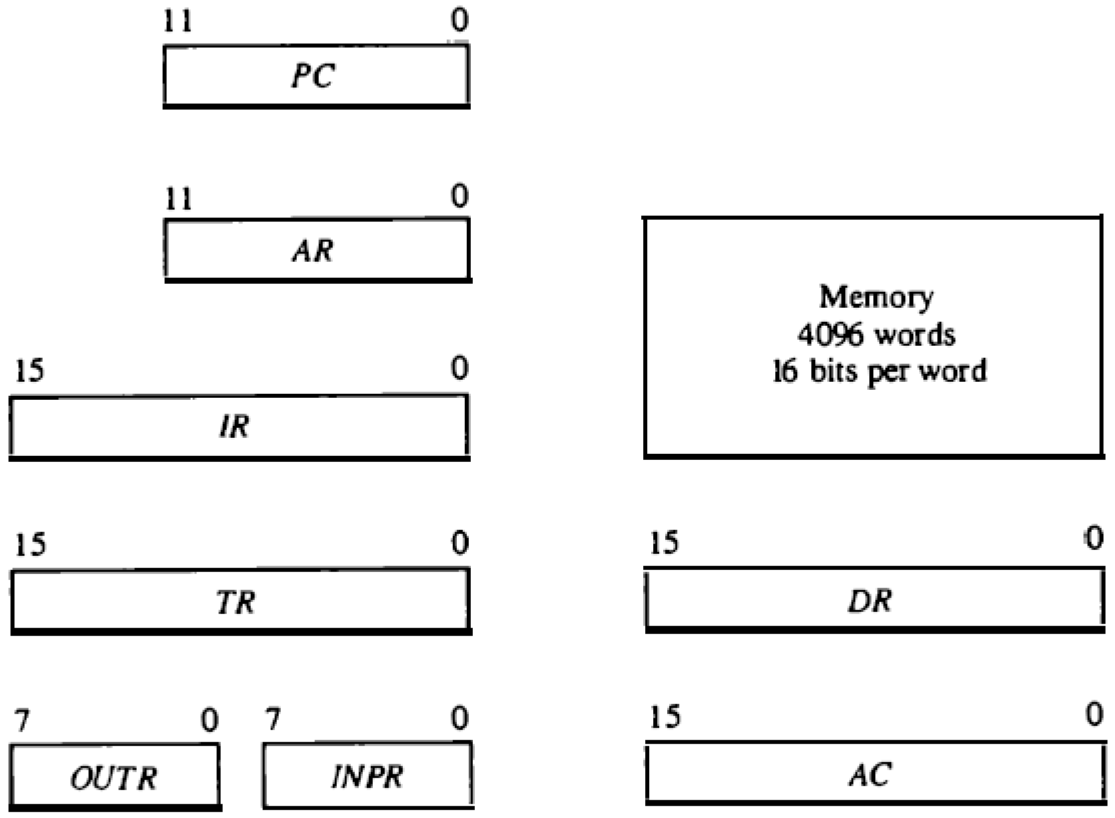
List of Registers for the Basic Computer



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

Reference: morris mano

Basic computer registers and memory



Computer registers

- ACCUMULATOR (AC): The processor register AC consists of 16-bits. It is used to hold the results or partial results of arithmetic and logical operations. An accumulator is a register in which intermediate arithmetic and logic results are stored.

- DATA REGISTER (DR): The register DR consists of 16-bits and it is used to hold memory operands (data). This register contains the data to be written into memory or receives the data read from memory.

Computer registers

- TEMPORARY REGISTER (TR): Temporary register have 16-bits and it provides temporary storage of variables or results.

- INSTRUCTION REGISTER (IR): The instruction register consists of 16-bits. The purpose of the instruction register is to hold a copy of the instruction which the processor is to execute. In our basic computer, instruction register (IR) holds instruction code which is read from memory.

Computer registers

- ADDRESS REGISTER (AR): This register specifies the address in memory for next read or writes operations. The address register consists of 12-bits.

- PROGRAM COUNTER (PC): Program counter has 12-bits and it holds the address of the next instruction to be read from memory after the current execution is executed. The instructions are read sequentially because the program counter automatically increments after fetching the current instruction.

Computer registers

- INPUT REGISTER (INPR): Input register has 8-bits. INPR register receives a character from an input device and delivers it to the AC.
- OUTPUT REGISTER (OUTR): Output register has 8-bits. The output register receives information from AC and transfer it to the output device.

example

- ADD AC,DR
  - AC=AC+DR
  - This instruction adds the contents of AC and DR stores the result in Accumulator.

AC=1234H

DR=2222H

After execution AC=3456H

Instruction

- A Computer Instruction is a binary code that That specify a sequence of micro-operations for the computer.

- Micro-operations are detailed low-level instructions used in some designs to implement complex machine instructions.

Instruction Set

- An Instruction set is a group of bits that instruct computer to perform a specific operation.

- The basic computer has three instructions code formats, each of 16 bit.

Types of Instructions

- These are of three types:
  - Memory-Reference instruction
  - Register-Reference instruction
  - Input-Output instruction

Memory-Reference instruction

- An instruction that has one or more of its operand addresses referring to a location in memory, as opposed to one of the CPU registers or some other way of specifying an operand.

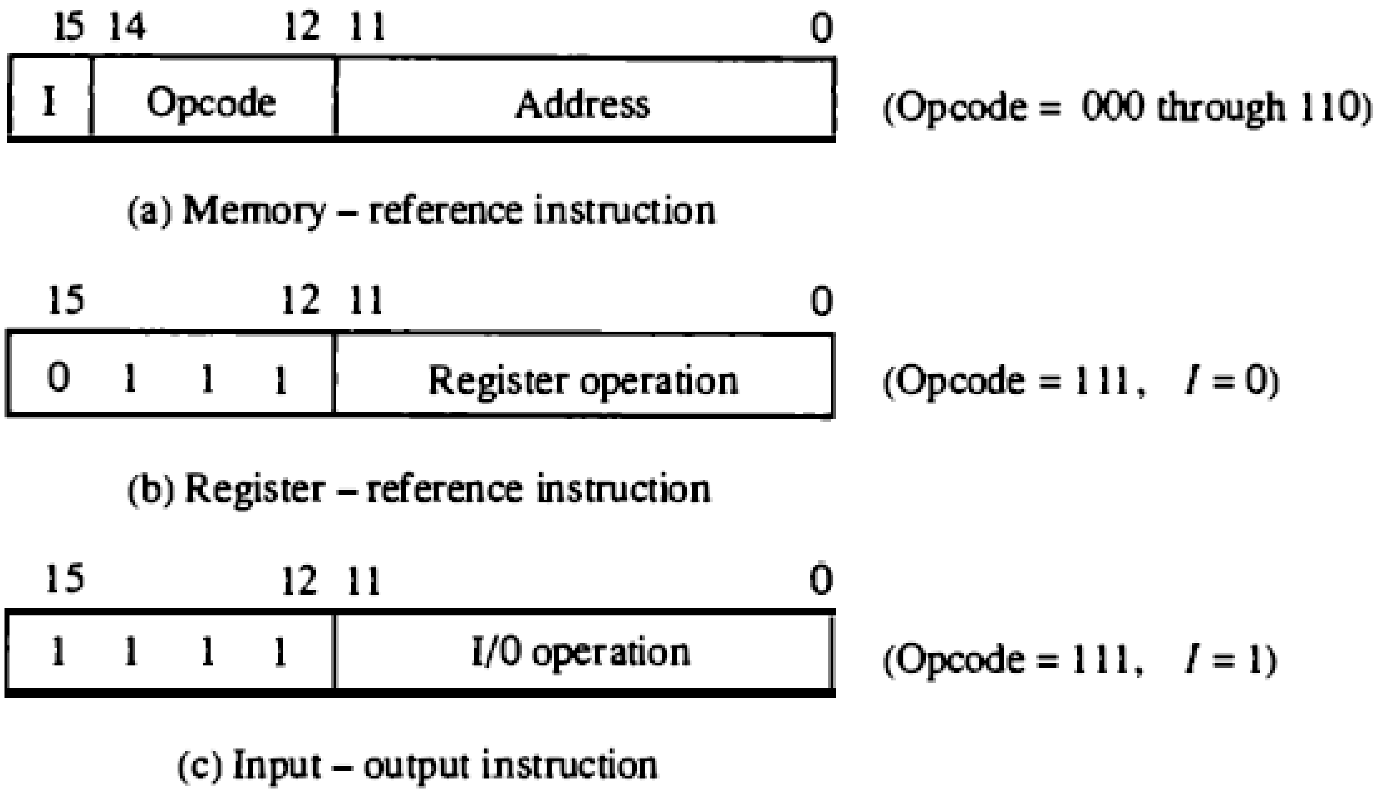
Register-Reference instruction

- 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.

Input-Output instruction

- An input-output instruction does not need a reference to memory and is recognized by the operation code lll 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.

Various Instructions format



(a) Memory – reference instruction

(b) Register – reference instruction

(c) Input – output instruction

Reference: Morris mano

Instruction Cycle

- The time period during which one instruction is fetched from memory and execute when a computer gives an instruction in machine language.
- Each instruction cycle in turn is subdivided into a sequence of subcycles or phases.
- After the execution the program counter is incremented to point to the next instruction.

Phases of Instruction Cycle

1. Fetch the instruction
2. Decode the instruction
3. Read the effective address from memory if the instruction has an indirect address.
4. Execute the instruction

The program counter PC is loaded with the address of the first instruction in the program

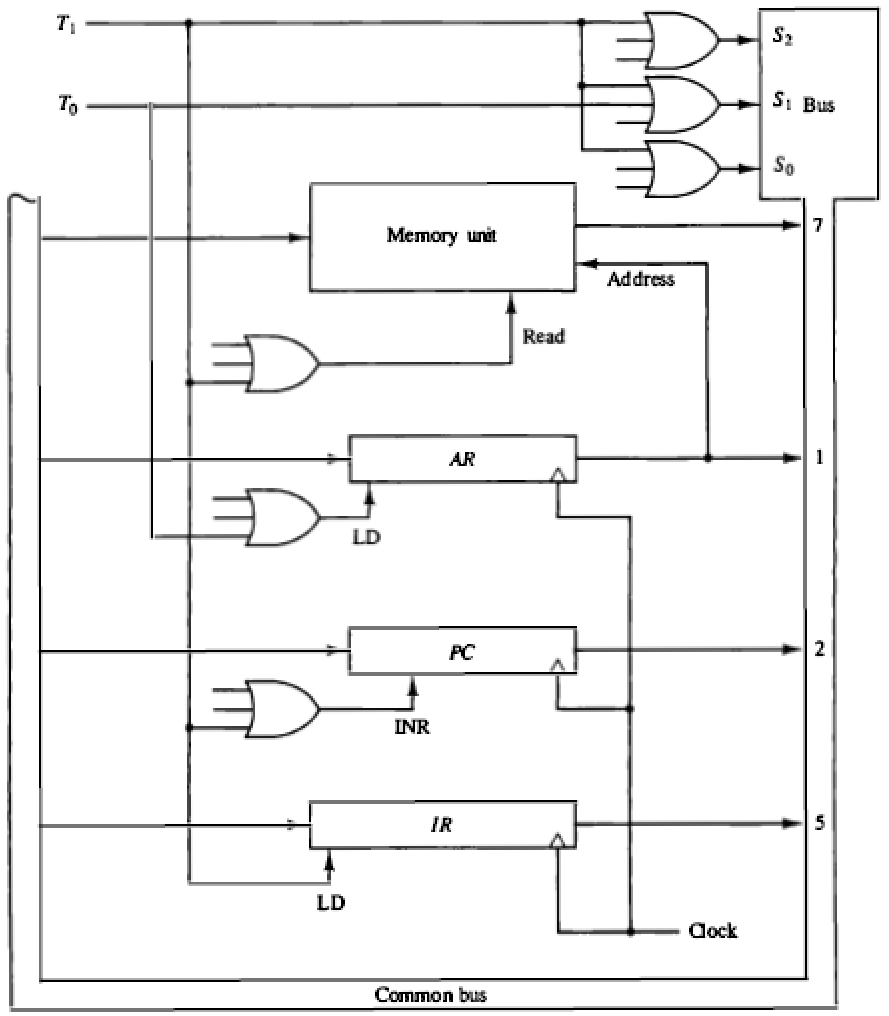
The sequence counter SC is cleared to 0, providing a decoded timing signal To.

After each clock pulse, SC is incremented by one, so that the timing signals go through a sequence T0, T1, T2, and so on.

The micro operations for the fetch and decode phases can be specified by the following register transfer statements.

$$\begin{aligned}
&T_0: \quad AR \leftarrow PC \\
&T_1: \quad IR \leftarrow M[AR], \quad PC \leftarrow PC + 1 \\
&T_2: \quad D_0, \ldots, D_7 \leftarrow \text{Decode } IR(12\text{–}14), \quad AR \leftarrow IR(0\text{–}11), \quad I \leftarrow IR(15)
\end{aligned}$$

Register Transfer for Fetch phase



- 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 T0
- The instruction read from memory is then placed in the instruction register IR with the clock transition associated with timing signal T1•

- At the same time, PC is incremented by one to prepare it for the address of the next instruction in the program.
- At time T2, 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:** SC is incremented after each clock pulse to produce the sequence To, T1, and T2•

To implement the first register transfer statement

$$T_0: \quad AR \leftarrow PC$$

To provide the data path for the transfer of PC to AR we must apply timing signal T0 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 .

To implement the second register transfer statement

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

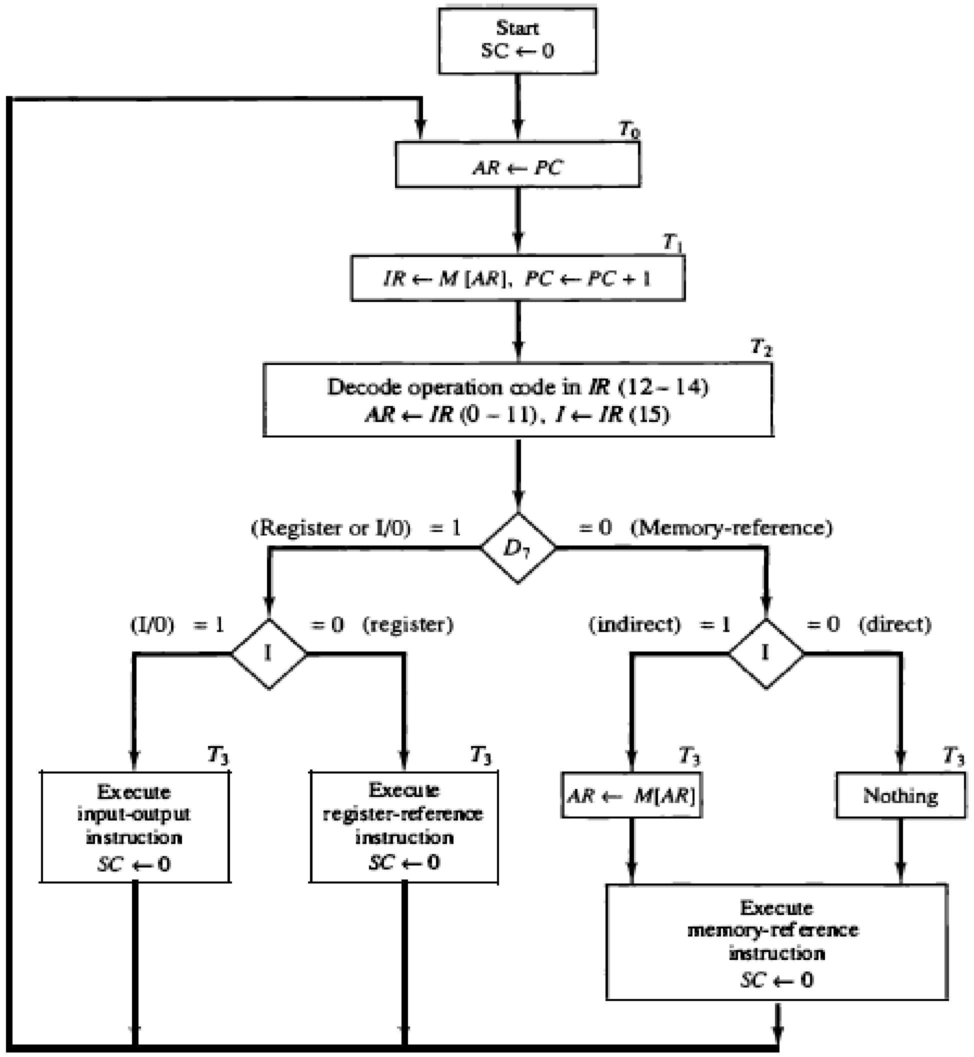
- It is necessary to use timing signal T_1 to provide the following connections in the bus system.
  1. Enable the read input of memory.
  2. Place the content of memory onto the bus by making $S_2S_1S_0$ = lll.
  3. Transfer the content o f the bus t o IR b y enabling the LD input of IR .
  4. Increment PC by enabling the INR input of PC .

Determining type of instruction

- The timing signal that is active after the decoding is T3.
- During time T3 the control unit determines the type of instruction that was just read from memory.
- Decoder output D, is equal to 1 if the operation code is equal to binary lll.
- we determine that if D7 = l, the instruction must be a register-reference or input-output type.

- If D, = 0, the operation code must be one of the other seven values 000 through 110, specifying a memory-reference instruction.
- If D7 = 0 and I = 1, we have a memoryreference instruction with an indirect address.

Flow Chart For Instruction Cycle



Start
SC ← 0
T_0
AR ← PC
T_1
IR ← M [AR], PC ← PC + 1
T_2
Decode operation code in IR (12 – 14)
AR ← IR (0 – 11), I ← IR (15)
(Register or I/0) = 1
D_7
= 0 (Memory-reference)
(I/0) = 1
I
= 0 (register)
(indirect) = 1
I
= 0 (direct)
T_3
Execute input-output instruction SC ← 0
T_3
Execute register-reference instruction SC ← 0
T_3
AR ← M[AR]
T_3
Nothing
Execute memory-reference instruction SC ← 0

$D_7'IT_3$:	$AR \leftarrow M[AR]$
$D_7'I'T_3$:	Nothing
$D_7I'T_3$:	Execute a register-reference instruction
D_7IT_3:	Execute an input–output instruction

Definition of Register Reference Instructions

- Register Reference Instructions are those which refer registers instead of some memory address or memory location for data to operate on

Register reference instructions

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

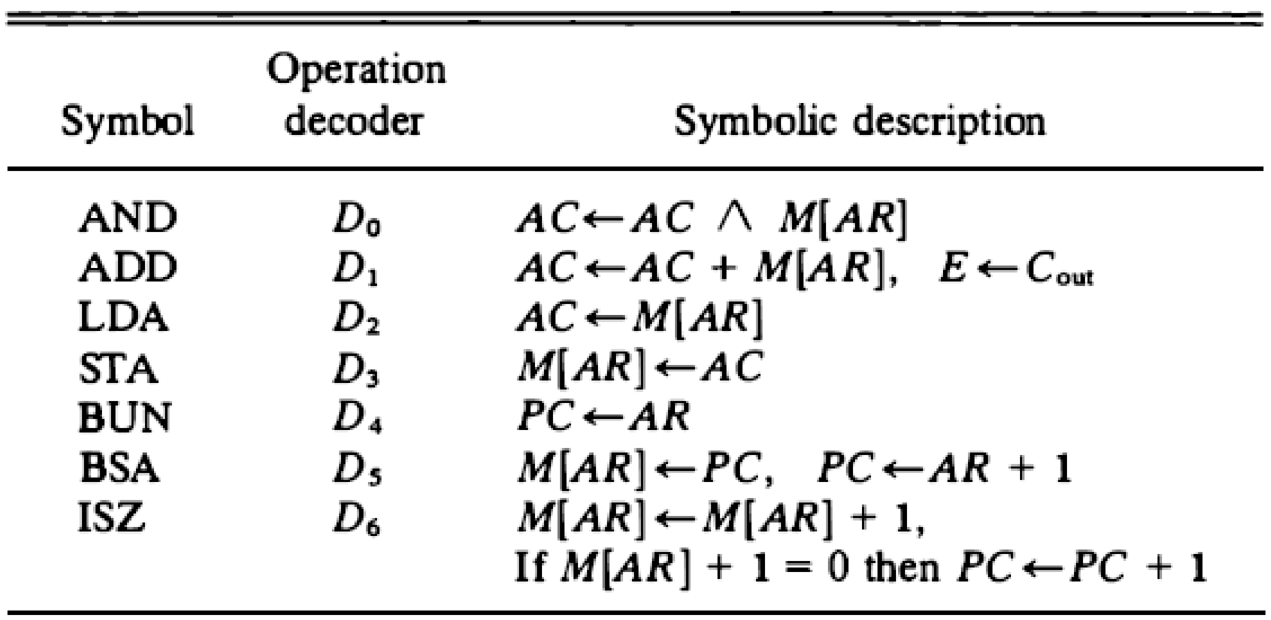
	r:	$SC \leftarrow 0$	Clear SC
CLA	rB_{11}:	$AC \leftarrow 0$	Clear AC
CLE	rB_{10}:	$E \leftarrow 0$	Clear E
CMA	rB_9:	$AC \leftarrow \overline{AC}$	Complement AC
CME	rB_8:	$E \leftarrow \overline{E}$	Complement E
CIR	rB_7:	$AC \leftarrow \text{shr } AC,\ AC(15) \leftarrow E,\ E \leftarrow AC(0)$	Circulate right
CIL	rB_6:	$AC \leftarrow \text{shl } AC,\ AC(0) \leftarrow E,\ E \leftarrow AC(15)$	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

Reference: Morris mano

- There are a total 12 register reference instructions in computer architecture that are in use and they are 1. CLA, 2. CMA, 3. CIL, 4. CIR, 5. INC, 6. SPA, 7. SNA, 8. SZA, 9. CLE, 10. CME, 11. SZE, 12. HLT
- Each instruction has its own functionality and all these register reference instructions operate on accumulator only

Memory reference instructions

There are seven different memory-reference instructions.



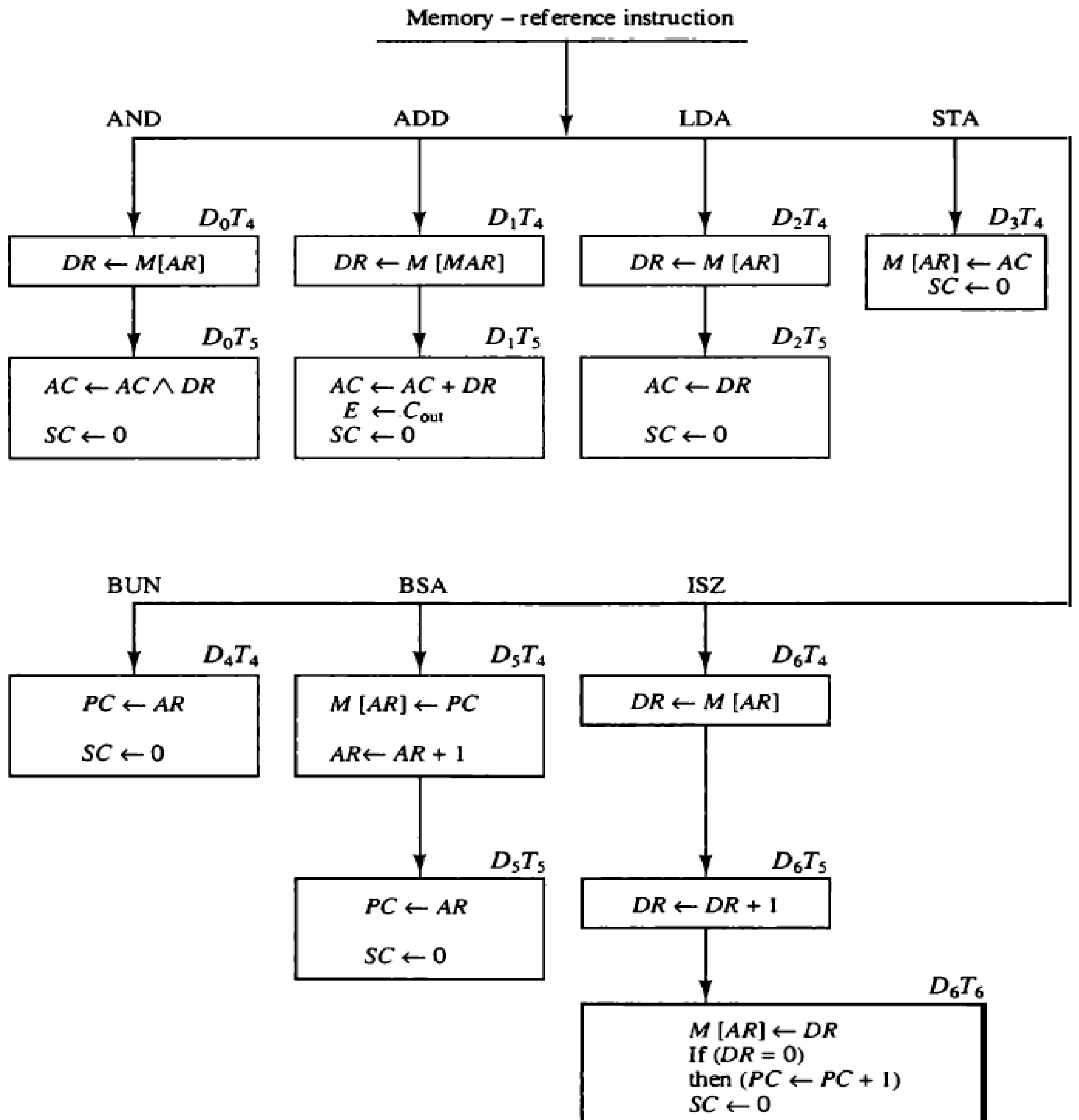
Symbol	Operation decoder	Symbolic description
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR], \quad E \leftarrow C_{out}$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
BUN	D_4	$PC \leftarrow AR$
BSA	D_5	$M[AR] \leftarrow PC, \quad PC \leftarrow AR + 1$
ISZ	D_6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

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 . The microoperations that execute this instruction are:

$$D_0T_4: \quad DR \leftarrow M[AR]$$
$$D_0T_5: \quad AC \leftarrow AC \wedge DR, \quad SC \leftarrow 0$$

Flow chart for memory reference instructions



Input and output instructions

$D_7IT_3 = p$ (common to all input–output instructions)
$IR(i) = B_i$ [bit in $IR(6–11)$ that specifies the instruction]

	p:	$SC \leftarrow 0$	Clear SC
INP	pB_{11}:	$AC(0–7) \leftarrow INPR$, $FGI \leftarrow 0$	Input character
OUT	pB_{10}:	$OUTR \leftarrow AC(0–7)$, $FGO \leftarrow 0$	Output character
SKI	pB_9:	If $(FGI = 1)$ then $(PC \leftarrow PC + 1)$	Skip on input flag
SKO	pB_8:	If $(FGO = 1)$ then $(PC \leftarrow PC + 1)$	Skip on output flag
ION	pB_7:	$IEN \leftarrow 1$	Interrupt enable on
IOF	pB_6:	$IEN \leftarrow 0$	Interrupt enable off

Addressing modes

- The operation field of an instruction specifies the operation to be performed.
- This operation must be executed on some data stored in computer registers or memory words.
- The way the operands are chosen during program execution is dependent on the addressing mode of the instruction.

Types of addressing modes

- Implied Mode
- Immediate Mode
- Register Mode
- Register Indirect Mode
- Autoincrement or Autodecrement Mode
- Direct Address Mode
- Indirect Address Mode
- Relative Address Mode
- Indexed Addressing Mode
- Base Register Addressing Mode

Implied addressing mode

- In this mode the operands are specified implicitly in the definition of the instruction.
- Example: CMA,CLA,CLE
- all register reference instructions that use an accumulator are implied-mode instructions.

Immediate Mode

- In this mode the operand is specified in the instruction itself. In other words, an immediate-mode instruction has an operand field rather than an address field.
- Immediate- mode instructions are useful for initializing registers to a constant value.
- Eg: ADD A, 1234H
- A=2222H
- AFTER EXECUTION A=3456H

Register Mode

- In this mode the operands are in registers that reside within the CPU. The particular register is selected from a register field in the instruction.
- Eg: Add A,DR

Register Indirect Mode

- In this mode the instruction specifies a register in the CPU whose contents give the address of the operand in memory. In other words, the selected register contains the address of the operand rather than the operand itself.
- The advantage of a register indirect mode instruction is that the address field of the instruction uses fewer bits to select a register than would have been required to specify a memory address directly.
- Eg: ADD A,[AR]
- AR=5000H
- 5000H=1234H
- ADD A,[5000H]

Autoincrement or Autodecrement Mode

- This is similar to the register indirect mode except that the register is incremented or decremented after (or before) its value is used to access memory.
- MOV A,[AR]
- INC AR
- LOOP

Direct Address Mode

- In this mode the effective address is equal to the address part of the instruction. The operand resides in memory and its address is given directly by the address field of the instruction.
- In a branch-type instruction the address field specifies the actual branch address.
- Eg: ADD A ,[5000H]
- Eg : BUN 2000H

Indirect Address Mode:

- In this mode the address field of the instruction gives the address where the effective address is stored in memory. Control fetches the instruction from memory and uses its address part to access memory again to read the effective address.

Relative Address Mode

- In this mode the content of the program counter is added to the address part of the instruction in order to obtain the effective address.
- The address part of the instruction is usually a signed number (in 2' s complement representation) which can be either positive or negative.
- When this number is added to the content of the program counter, the result produces an effective address whose position in memory is relative to the address of the next instruction.

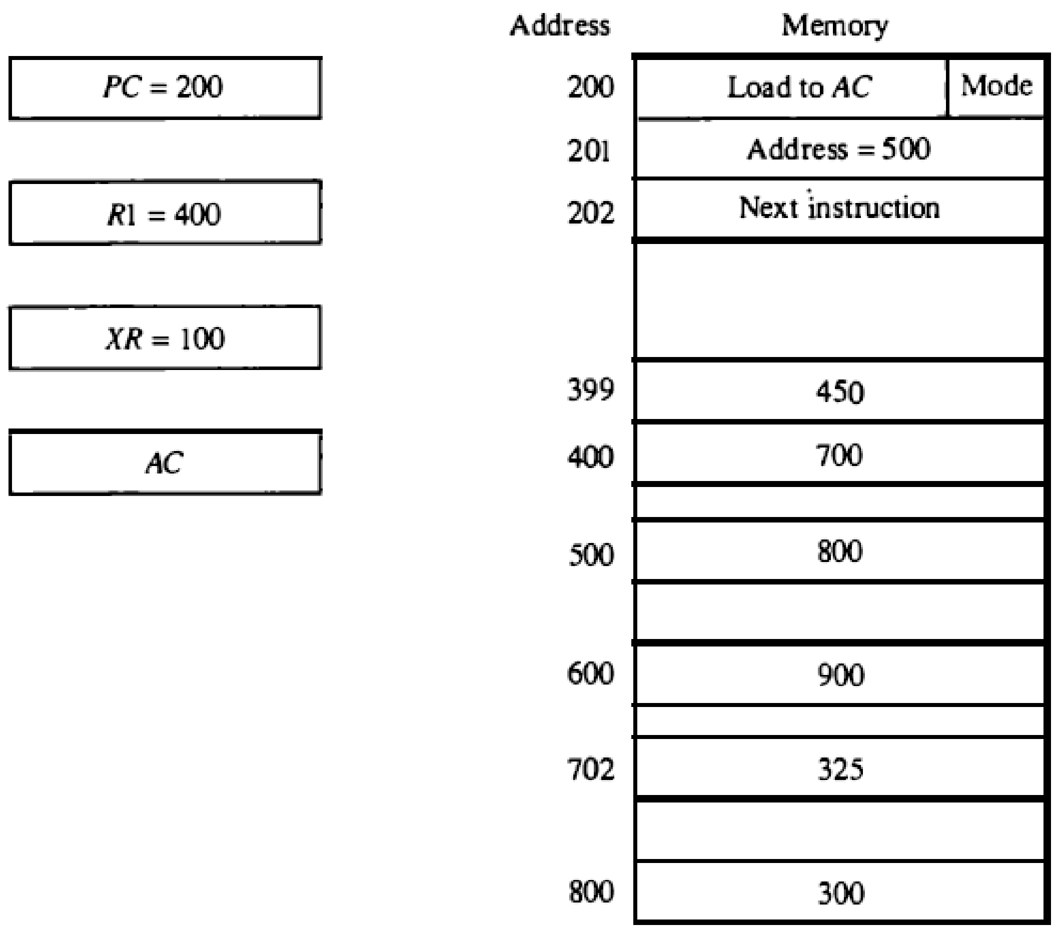
Indexed Addressing Mode

- In this mode the content of an index register is added to the address part of the instruction to obtain the effective address.
- The index register is a special CPU register that contains an index value.
- The address field of the instruction defines the beginning address of a data array in memory.

Base Register Addressing Mode

- In this mode the content of a base register is added to the address part of the instruction to obtain the effective address.
- This is similar to the indexed addressing mode except that the register is now called a base register instead of an index register.
- An index register is assumed to hold an index number that is relative to the address part of the instruction.
- A base register is assumed to hold a base address and the address field of the instruction gives a displacement relative to this base address.

Numeric example for Addressing modes



Addressing Mode	Effective Address	Content of *AC*
Direct address	500	800
Immediate operand	201	500
Indirect address	800	300
Relative address	702	325
Indexed address	600	900
Register	—	400
Register indirect	400	700
Autoincrement	400	700
Autodecrement	399	450

INSTRUCTION SET

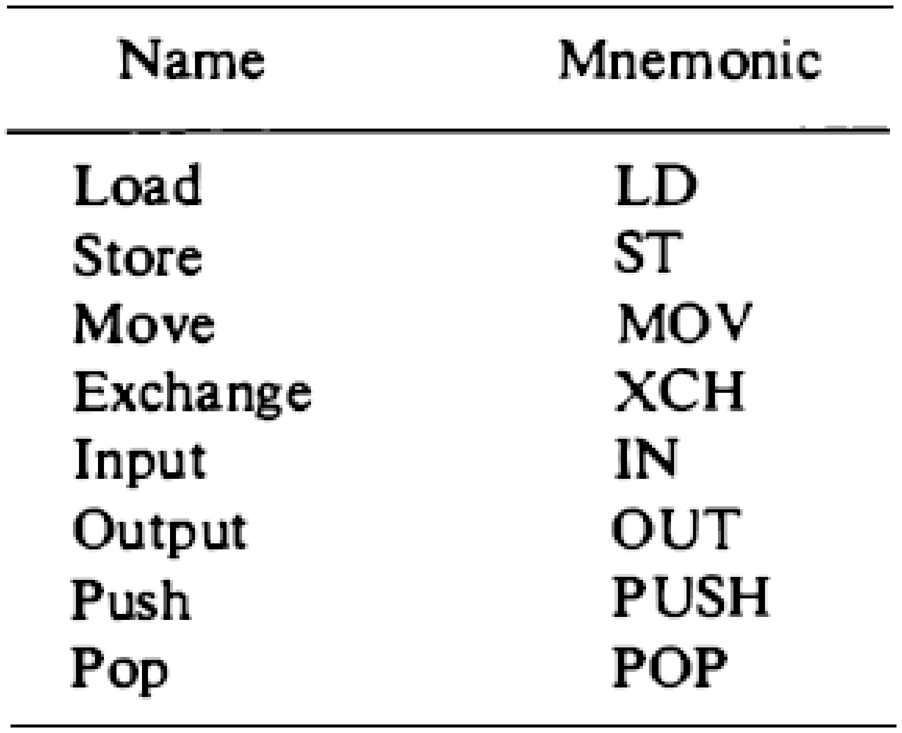
- computer instructions can be classified into three categories:

1. Data transfer instructions
2. Data manipulation instructions
3. Program control instructions

Data transfer instructions

- Data transfer instructions cause transfer of data from one location to another without changing the binary information content.
- Data manipulation instructions are those that perform arithmetic, logic, and shift operations.

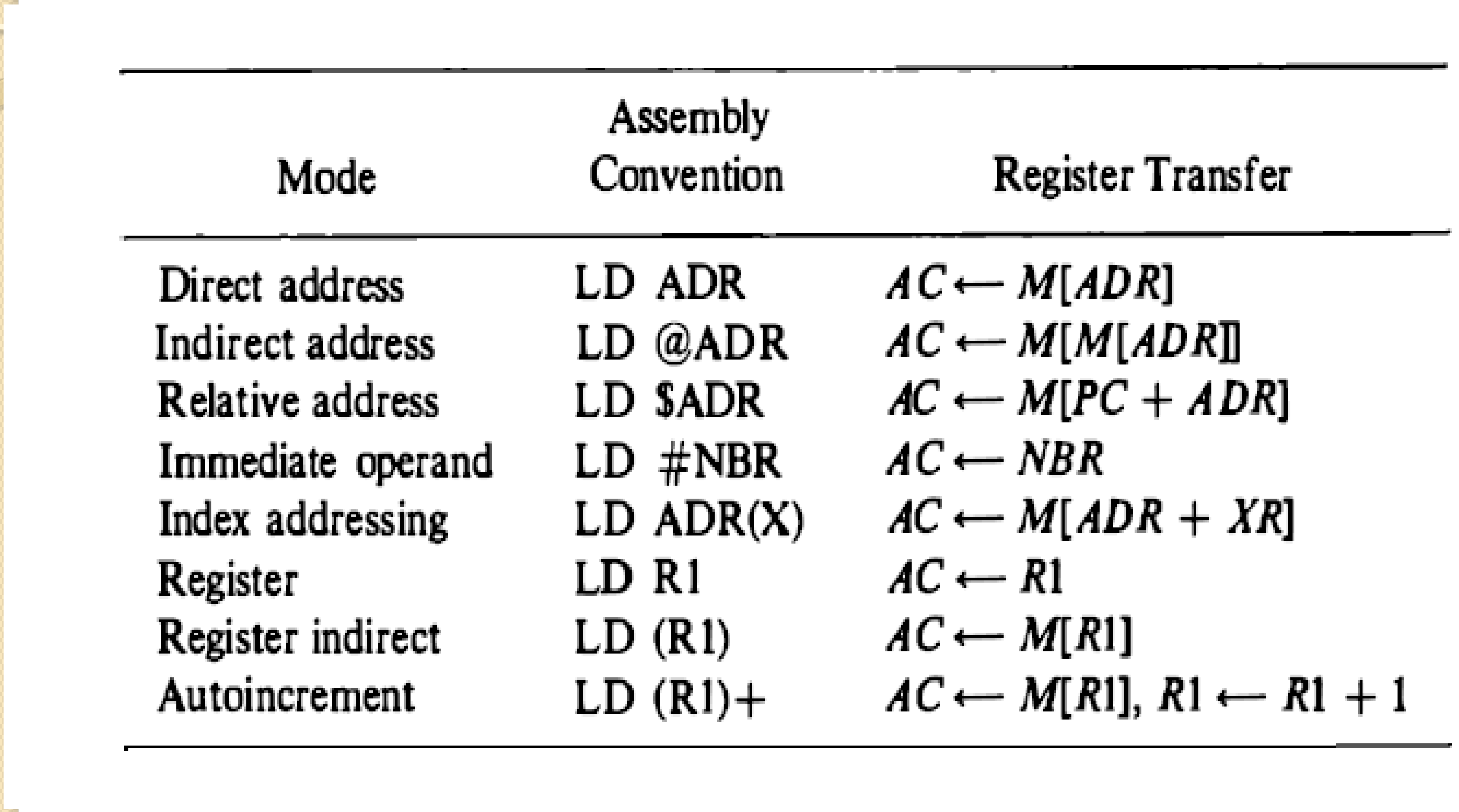
Data transfer instructions



Name	Mnemonic
Load	LD
Store	ST
Move	MOV
Exchange	XCH
Input	IN
Output	OUT
Push	PUSH
Pop	POP

Mov AC,R1

ADDRESSING MODES FOR LOAD INSTRUCTION



Mode	Assembly Convention	Register Transfer
Direct address	LD ADR	$AC \leftarrow M[ADR]$
Indirect address	LD @ADR	$AC \leftarrow M[M[ADR]]$
Relative address	LD $ADR	$AC \leftarrow M[PC + ADR]$
Immediate operand	LD #NBR	$AC \leftarrow NBR$
Index addressing	LD ADR(X)	$AC \leftarrow M[ADR + XR]$
Register	LD R1	$AC \leftarrow R1$
Register indirect	LD (R1)	$AC \leftarrow M[R1]$
Autoincrement	LD (R1)+	$AC \leftarrow M[R1],\ R1 \leftarrow R1 + 1$

Data Manipulation Instructions

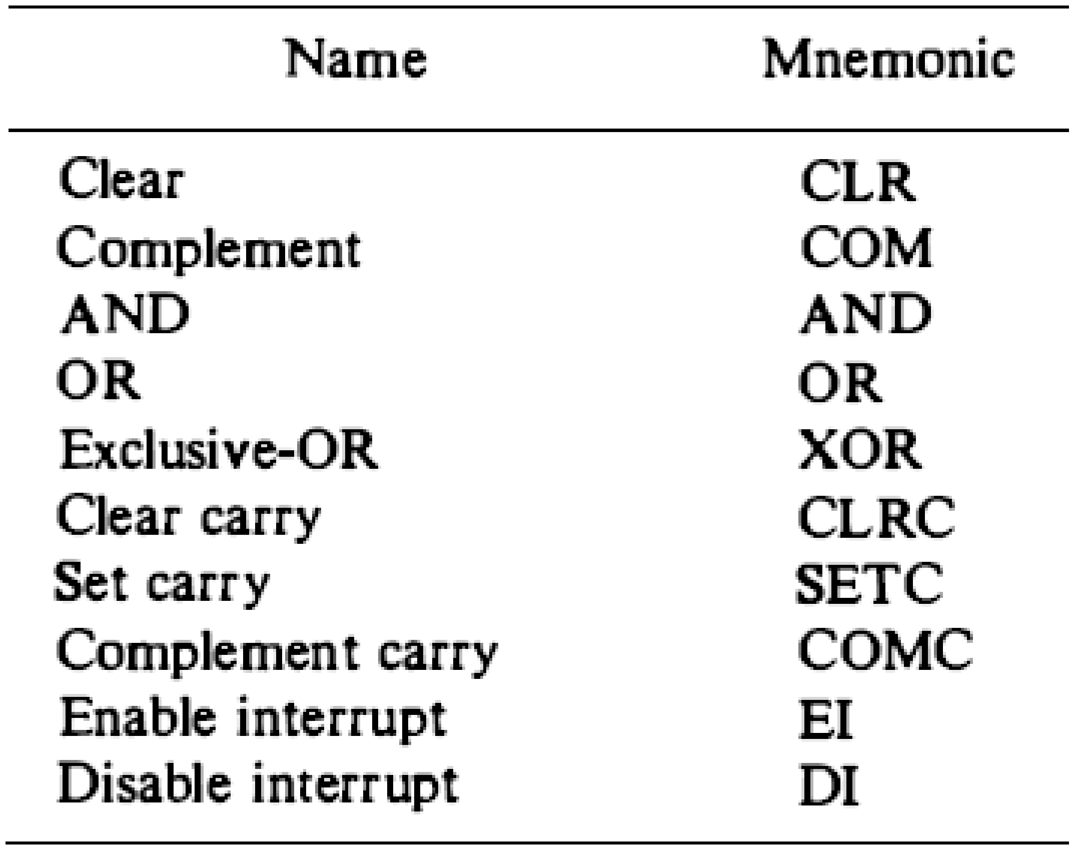
- Data manipulation instructions perform operations on data and provide the computational capabilities for the computer.
- The data manipulation instructions in a typical computer are usually divided into three basic types:

1. Arithmetic instructions
2. Logical and bit manipulation instructions
3. Shift instructions

Arithmetic Instructions

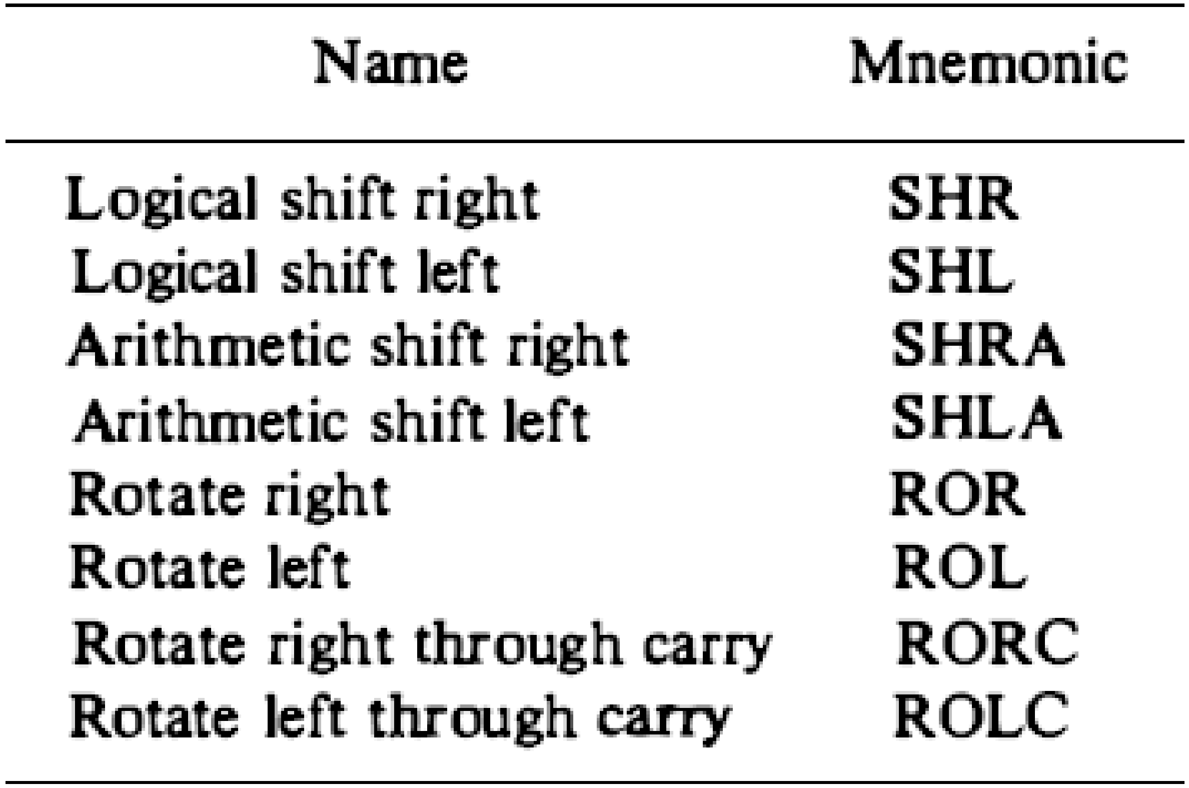
Name	Mnemonic
Increment	INC
Decrement	DEC
Add	ADD
Subtract	SUB
Multiply	MUL
Divide	DIV
Add with carry	ADDC
Subtract with borrow	SUBB
Negate (2's complement)	NEG

Logical and Bit Manipulation Instructions



Name	Mnemonic
Clear	CLR
Complement	COM
AND	AND
OR	OR
Exclusive-OR	XOR
Clear carry	CLRC
Set carry	SETC
Complement carry	COMC
Enable interrupt	EI
Disable interrupt	DI

SHIFT INSTRUCTIONS

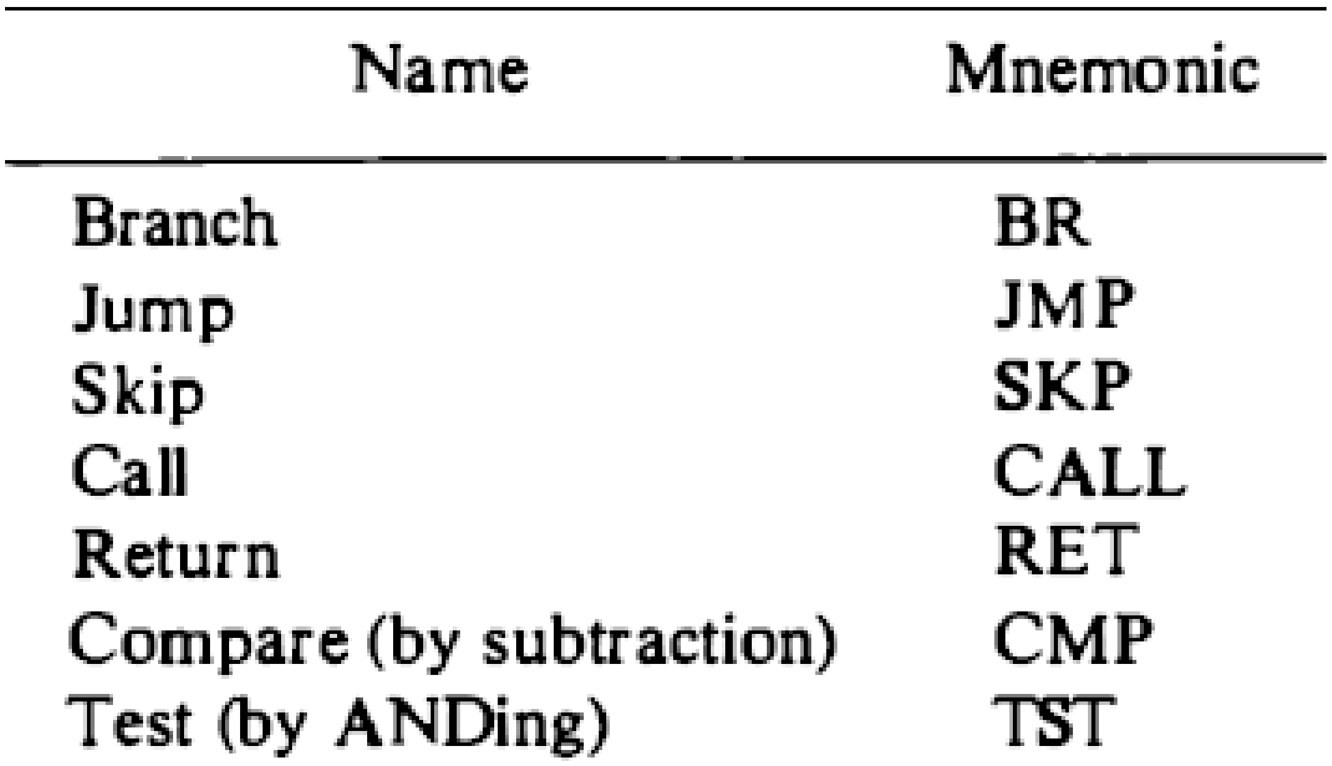


Name	Mnemonic
Logical shift right	SHR
Logical shift left	SHL
Arithmetic shift right	SHRA
Arithmetic shift left	SHLA
Rotate right	ROR
Rotate left	ROL
Rotate right through carry	RORC
Rotate left through carry	ROLC

Program Control

- Program control instructions provide decision-making capabilities and change the path taken by the program when executed in the computer.
- The instruction set of a particular computer determines the register transfer operations and control decisions that are available to the user.

Program Control



Name	Mnemonic
Branch	BR
Jump	JMP
Skip	SKP
Call	CALL
Return	RET
Compare (by subtraction)	CMP
Test (by ANDing)	TST