COMPUTER ORGANIZATION&ARCHITECHTURE

MODULE –I

BASIC COMPUTER ORGANIZATION AND DESIGN

FUNCTIONAL UNITS

1) Input Unit

2) Memory Unit

3) Arithmetic and logic unit

4) Output unit

5) Control Unit

A computer consist of 5 functional units

1) Input unit

Computers accept coded information through input unit, which read the data. The most well known input device is the keyboard. Whenever a key is pressed corresponding letters or digits automatically translated into its corresponding binary code and transmitted over a cable.

2) Memory unit

The function of memory unit is to store programs and data. There are two classes of storage primary and secondary storage.

3) Arithmetic and logic unit (ALU)

Most computer operations are executed in the arithmetic and logic unit of the processor.

4) Output unit

Its function is to send processed result to the outside world.

Example: Printers, speakers, mointor

5) Control unit

The memory, ALU and input and output units are used to store and process information and perform input and output operations. The operation of these units must be coordinated in some way. This is the task of control unit.

NOTE:

- The computer accept information in the form of programs and data through input unit
- Information stored in the memory by the memory unit
- The information stored in the memory is fetched by arithmetic and logic unit
  Where it processes the information and leaves the information through an output unit.
- All activities inside the machine are directed by the control unit.

BASIC OPERATIONAL CONCEPTS OF COMPUTER

To perform a given task an appropriate program consisting of a list of instructions is stored in the memory. Individual instructions are brought from the memory into the processor, which executes the specified operations. Data to be stored are also stored in the memory.

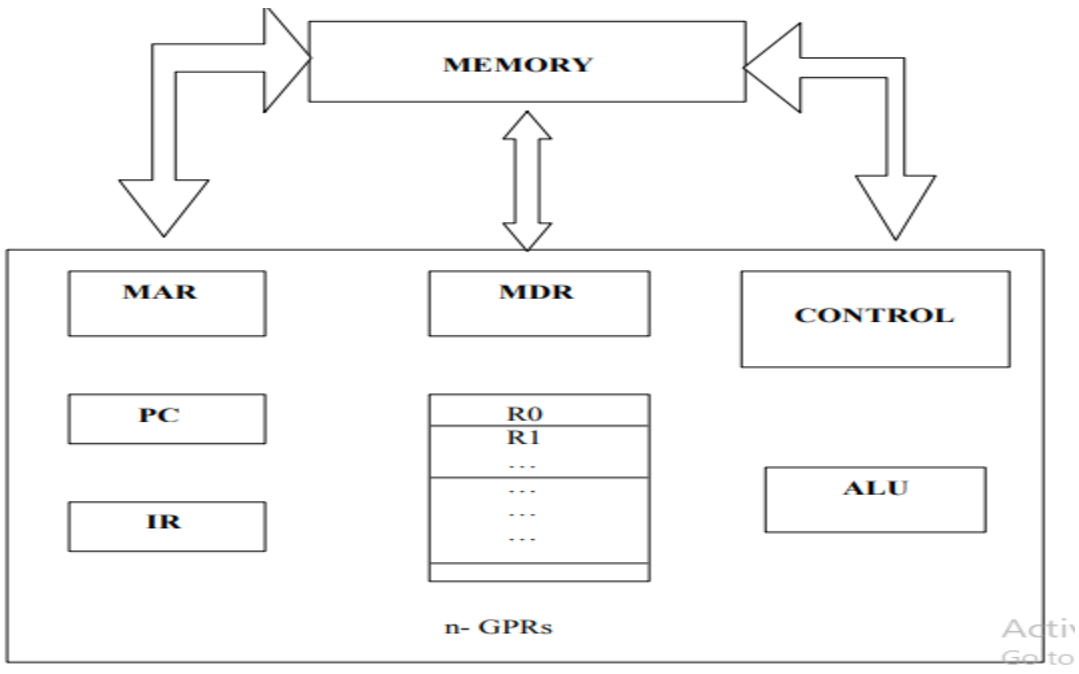
Examples: - Add LOCA, R0

This instruction adds the operand at memory location LOCA, to operand in register R0 & places the sum into register.

This instruction requires the performance of several steps,

1. First the instruction is fetched from the memory into the processor.
2. The operand at LOCA is fetched and added to the contents of R0
3. Finally the resulting sum is stored in the register R0

Transfers between the memory and the processor are started by sending the address of the memory location to be accessed to the memory unit and issuing the appropriate control signals. The data are then transferred to or from the memory.



Connections between the processor and the memory

The fig shows how memory & the processor can be connected. In addition to the ALU & the control circuitry, the processor contains a number of registers used for several different purposes.

The instruction register (IR):- Holds the instruction that is currently being executed.

The program counter PC:- This is another specialized register that keeps track of execution of a program. It contains the memory address of the next instruction to be fetched and executed. Besides IR and PC, there are n-general purpose registers R0 through Rn-1.

The other two registers which facilitate communication with memory are: -

1. MAR – (Memory Address Register):- It holds the address of the location to be accessed.
2. MDR – (Memory Data Register):- It contains the data to be written into or read out of the address location.

Operating steps are

1. Programs reside in the memory & usually get these through the I/P unit.
2. Execution of the program starts when the PC is set to point at the first instruction of the program.
3. Contents of PC are transferred to MAR and a Read Control Signal is sent to the memory.
4. After the time required to access the memory elapses, the address word is read out of the memory and loaded into the MDR.
5. Now contents of MDR are transferred to the IR & now the instruction is ready to be decoded and executed
6. If the instruction involves an operation by the ALU, it is necessary to obtain the required operands.
7. An operand in the memory is fetched by sending its address to MAR & Initiating a read cycle.
8. When the operand has been read from the memory to the MDR, it is transferred from MDR to the ALU.
9. After one or two such repeated cycles, the ALU can perform the desired operation.
10. If the result of this operation is to be stored in the memory, the result is sent to MDR.
11. Address of location where the result is stored is sent to MAR & a write cycle is initiated.
12. The contents of PC are incremented so that PC points to the next instruction that is to be executed.

Instruction codes

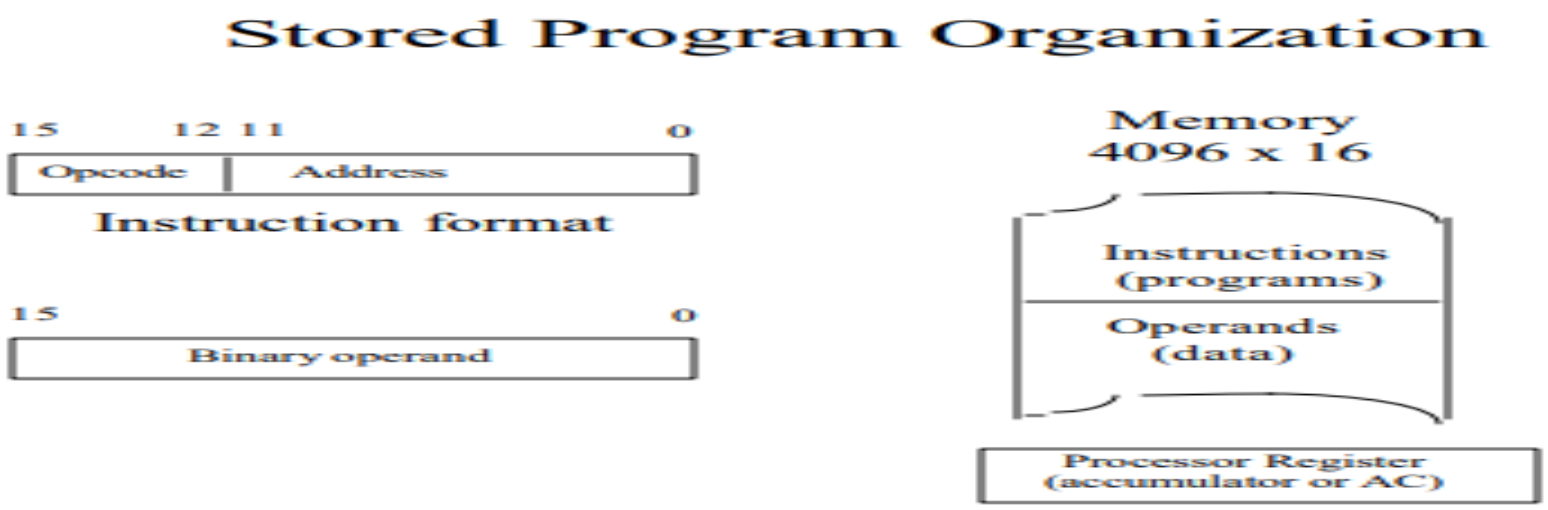
A set of instructions that specify the operations, operands, and the sequence by which processing has to occur. Computer can be instructed about the specific sequence of operations it must perform. An **instruction code** is a group of bits that instruct the computer to perform a specific operation. The **operation code** (opcode) of an instruction is a group of bits that define operations such as addition, subtraction, shift, complement, etc. An instruction must also include one or more operands, which indicate the registers and/or memory addresses from which data is taken or to which data is deposited.

Micro operations

The instructions are stored in computer memory in the same manner that data is stored. The control unit interprets these instructions and uses the operations code to determine the sequences of micro operations that must be performed to execute the instruction.

Stored Program Organization

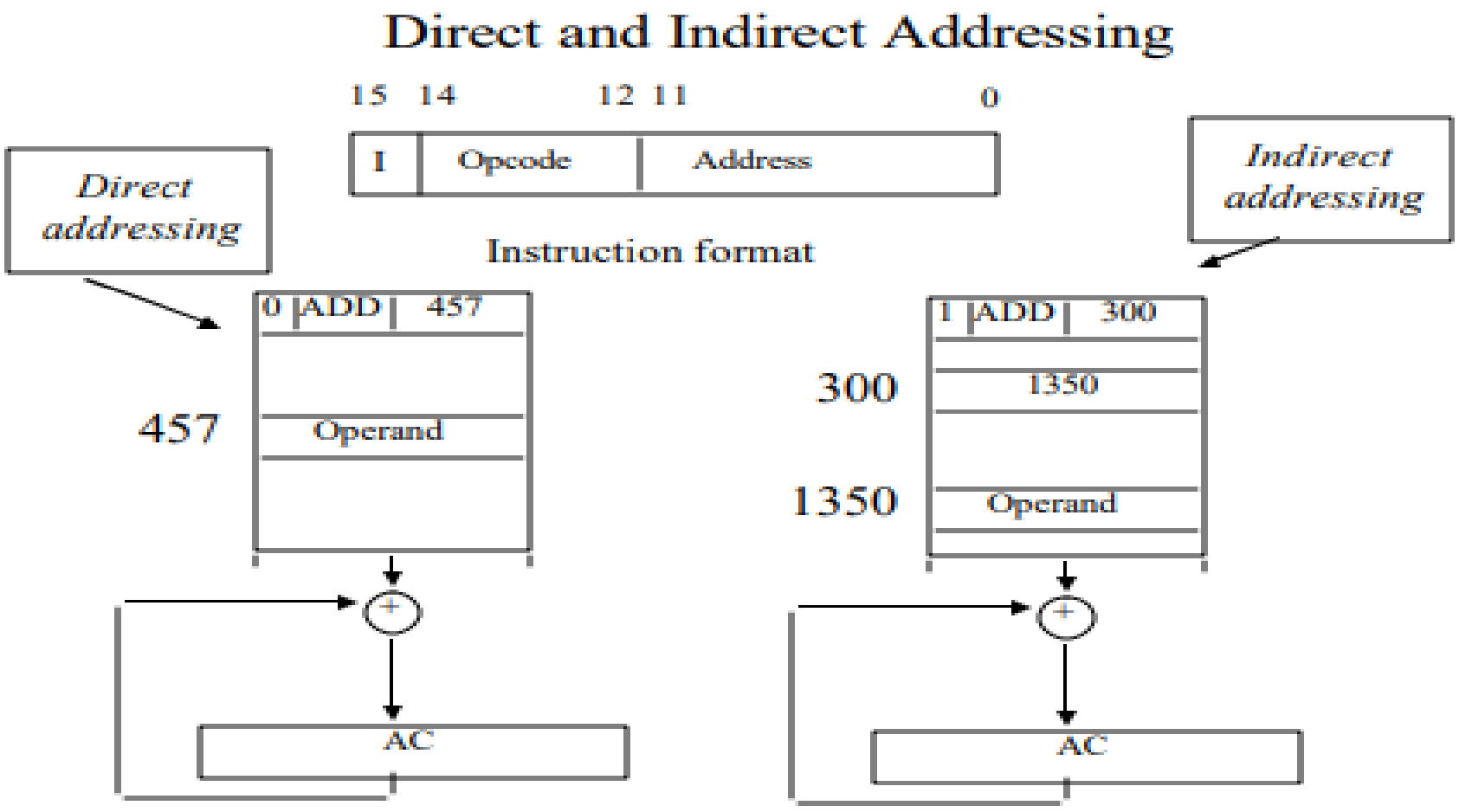
The ability to store and execute instructions is the most important property of a general-purpose computer. That type of stored program concept is called stored program organization. The simplest design is to have one processor register (called the accumulator) and two fields in the instruction, one for the opcode and one for the operand. Any operation that does not need a memory operand frees the other bits to be used for other purposes, such as specifying different operations.



There are 3 different types of operands that can appear in an instruction:

Direct operand - an operand stored in the register or in the memory location specified. Indirect operand - an operand whose address is stored in the register or in the memory location specified.

Immediate operand - an operand whose value is specified in the instruction.



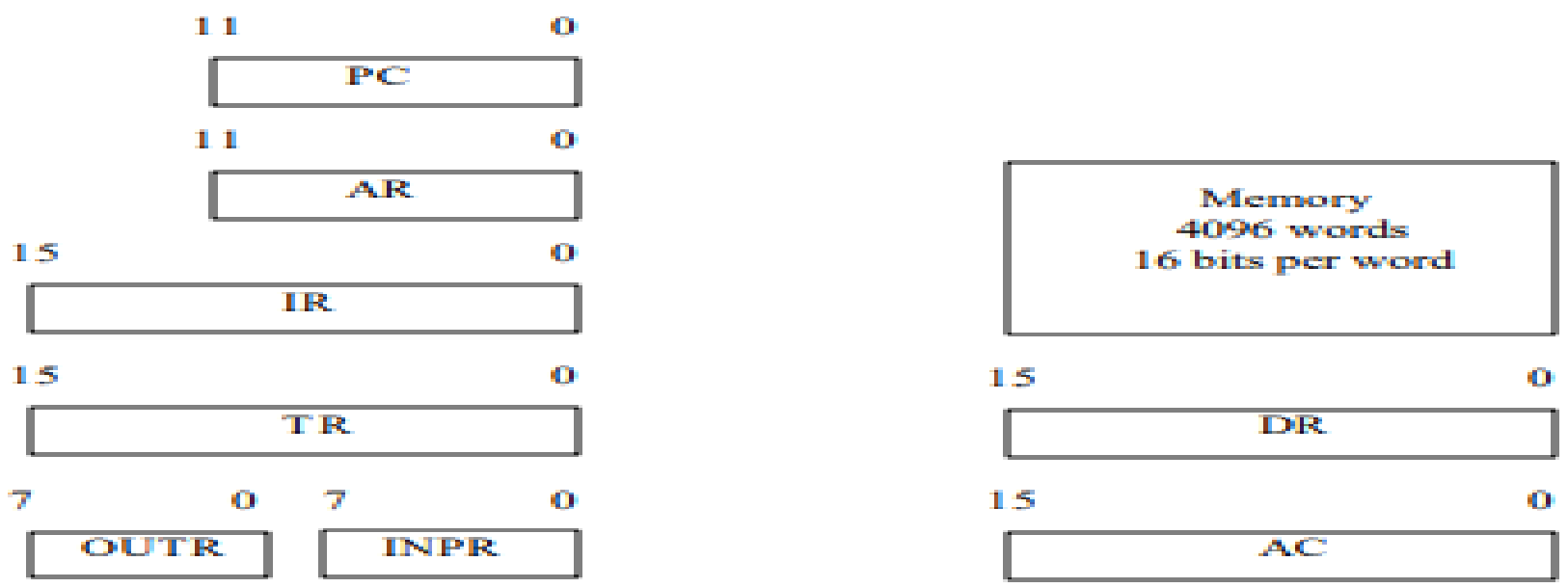
Computer Registers

Computer instructions are stored in consecutive locations and are executed sequentially; this requires a register which can stored the address of the next instruction. We call it the Program Counter. We need registers which can hold the address at which a memory operand is stored as well as the value itself. We need a place where we can store temporary data, the instruction being executed, a character being read in, a character being written out.

List of Registers for the Basic Computer

Register Symbol	# of Bits	Register Name	Function
DR	16	Data Register	Holds memory operand
AR	12	Address Register	Holds mem. address
AC	16	Accumulator	Processor Reg.
IR	16	Instruction Register	Holds instruction code
PC	12	Program Counter	Holds instruction address
TR	16	Temporary Register	Holds temporary data
INPR	8	Input Register	Holds input character
OUTR	8	Output Register	Holds output character

Basic Computer Registers and Memory



Computer Instructions

Computer instructions are a set of machine language instructions that a particular processor understands and executes. A computer performs tasks on the basis of the instruction provided.

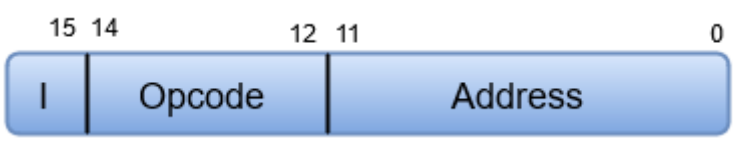
An instruction comprises of groups called fields. These fields include:

- The Operation code (Opcode) field which specifies the operation to be performed.
- The Address field which contains the location of the operand, i.e., register or memory location.
- The Mode field which specifies how the operand will be located.

A basic computer has three instruction code formats which are:

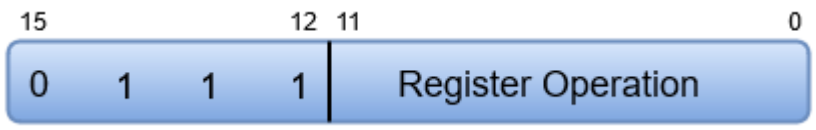
1. Memory - reference instruction
2. Register - reference instruction
3. Input-Output instruction

Memory - reference instruction



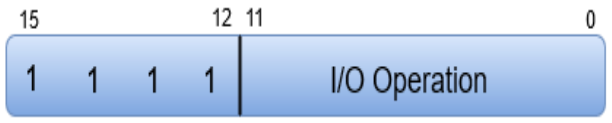
In Memory-reference instruction, 12 bits of memory is used to specify an address and one bit to specify the addressing mode 'I'.

Register - reference instruction



The Register-reference instructions are represented by the Opcode 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 (Accumulator) register.

Input-Output instruction



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 the input-output operation or test performed.

Instruction Set Completeness

A set of instructions is said to be complete if the computer includes a sufficient number of instructions in each of the following categories:

- Arithmetic, logical and shift instructions
- A set of instructions for moving information to and from memory and processor registers.
- Instructions which controls the program together with instructions that check status conditions.
- Input and Output instructions
- A huge amount of binary information is stored in the memory unit, but all computations are done in processor registers. Therefore, one must possess the capability of moving information between these two units.
- Program control instructions such as branch instructions are used change the sequence in which the program is executed.

- Input and Output instructions act as an interface between the computer and the user. Programs and data must be transferred into memory, and the results of computations must be transferred back to the user.

Symbol	Hexadecimal Code		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 AC content in memory
BUN	4xxx	Cxxx	Branch Unconditionally
BSA	5xxx	Dxxx	Branch and Save Return Address
ISZ	6xxx	Exxx	Increment and skip if 0
CLA	7800		Clear AC
CLE	7400		Clear E (overflow bit)
CMA	7200		Complement AC
CME	7100		Complement E
CIR	7080		Circulate right AC and E
CIL	7040		Circulate left AC and E
INC	7020		Increment AC
SPA	7010		Skip next instruction if AC > 0
SNA	7008		Skip next instruction if AC < 0
SZA	7004		Skip next instruction if AC = 0
SZE	7002		Skip next instruction if E = 0
HLT	7001		Halt computer
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

Timing and Control

The timing for all registers in the basic computer is controlled by a Master clock generator. The control signals are generated in the control unit and provide control inputs for the multiplexers in the common bus.

Control Unit

It 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). An Instruction register where it is divided into three parts: The I bit. I is equal to 0 for direct address and to 1 for indirect address. The operation code and address through 11 .The operation code (12-14) bit decoded with the 3 x 8 decoder generate 8 outputs of decoder denoted as D_0–D_7were connected to the control logic gates.

The sequence counter SC can be incremented or cleared synchronously. The 4 bit sequence counter, it has a binary of 4 x 16 decoder of 0 – 15 timing signals T_0 – T_{15} connected to the control logic gates. Then a counter is cleared to 0 the next timing signal will be T_0, inputs for the sequence counter are INR, Clear, Clock.

Timing Signals

Once in a while, the counter is cleared to 0, causing the next active timing signal to be T_0.

Example

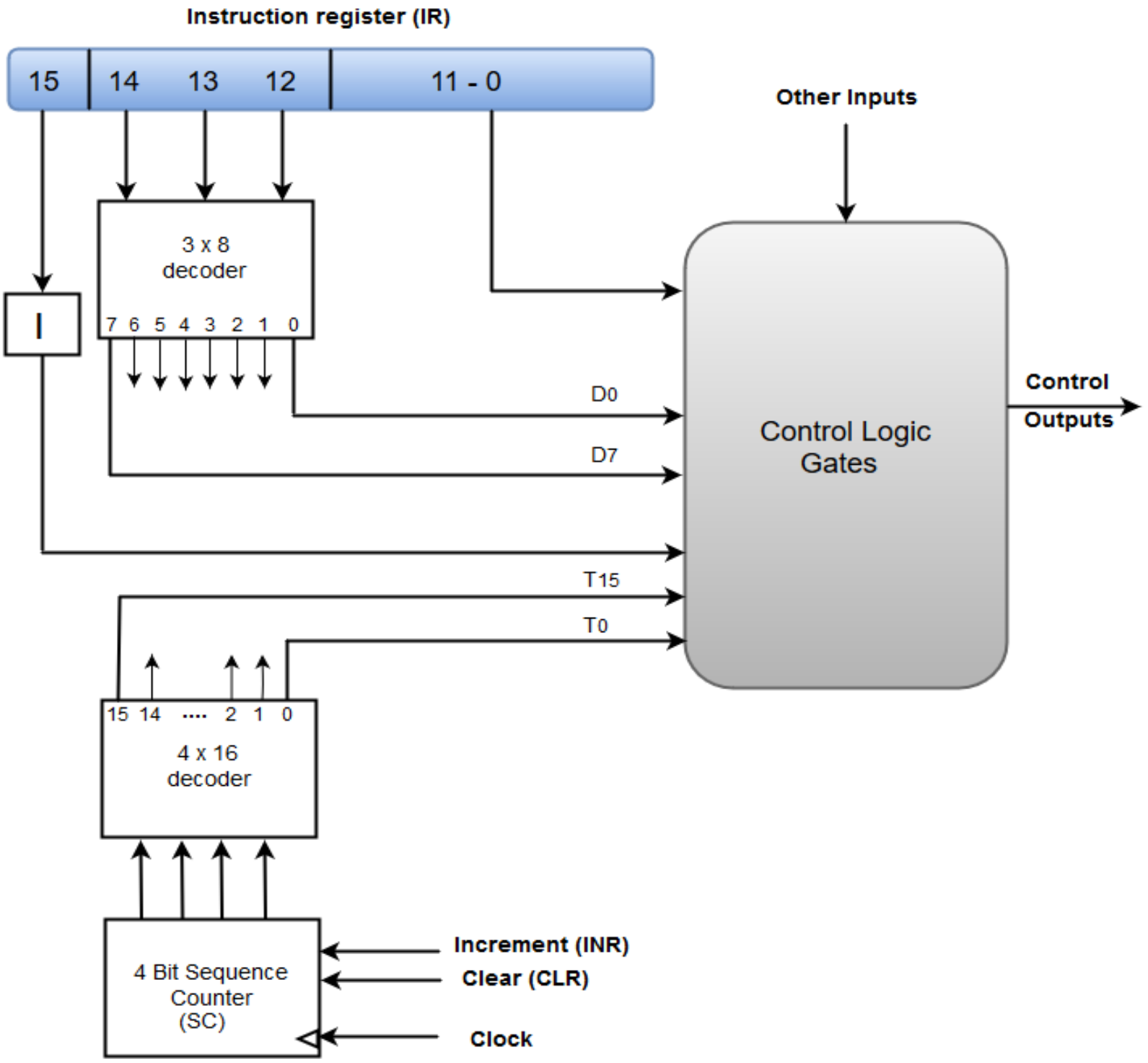
Consider the case where SC is incremented to provide timing signals T_0, T_1, T_2, T_3 and T_4 in sequence. At time T_4, SC is cleared to 0 if decoder output D_3 is active. This is expressed symbolically as

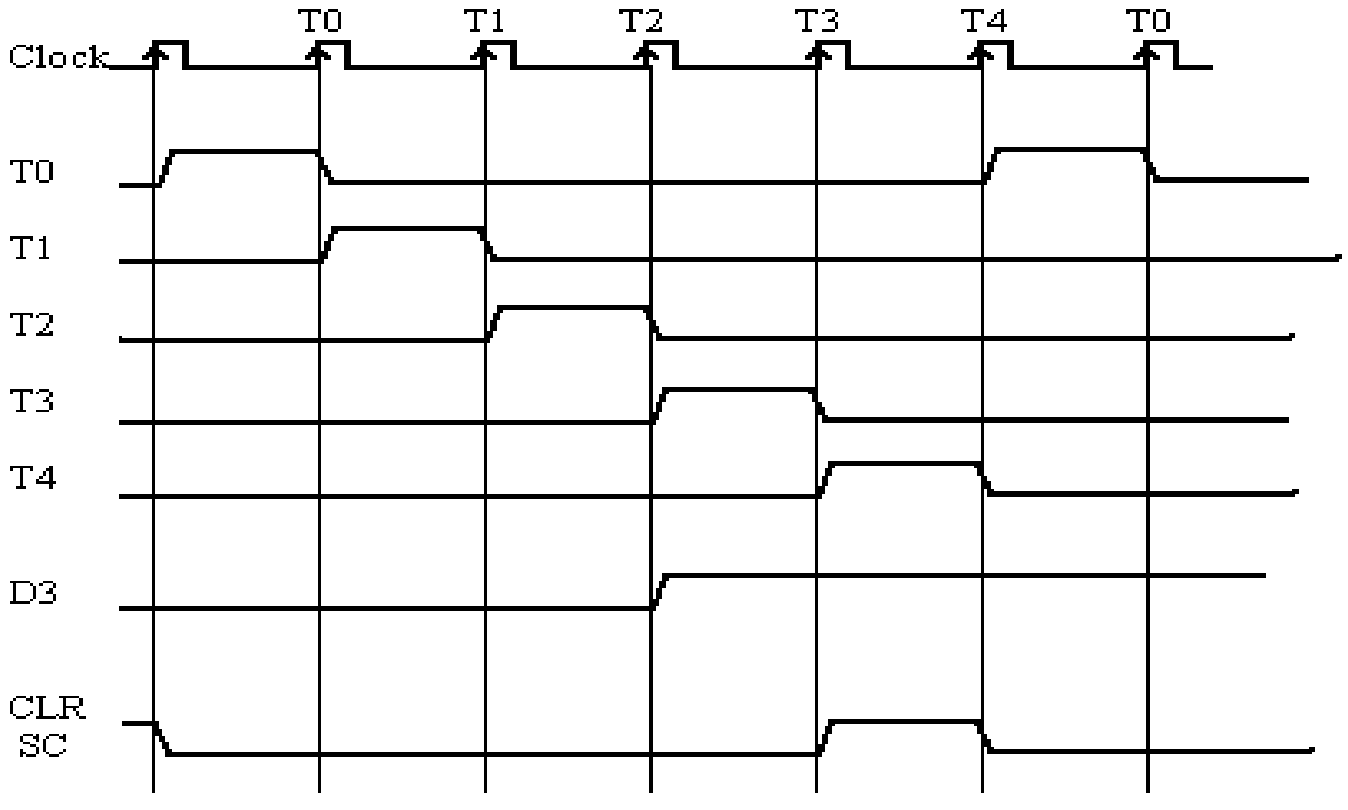
D_3 T_4: SC ← 0

Ie,$D_3$is active at timing pluse T_4 when sequence counter is cleared to 0.

The timing diagram shows the time relationship of the control signals. After the expression again the Time impulse start with T_0

Control Unit of a Basic Computer:





Examole of control timing signals

Instruction Cycle

A program residing in the memory unit of the computer consists of a sequence of instructions. The program is executed in the computer by going through a cycle for each instruction. 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.

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

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 of T_0, T_1, $T_2$and so on.

Fetch the data from the memory after fetching data that will be decoded in the decode phase. After decoding it reads the effective address from memory of instructions if it is indirect address.

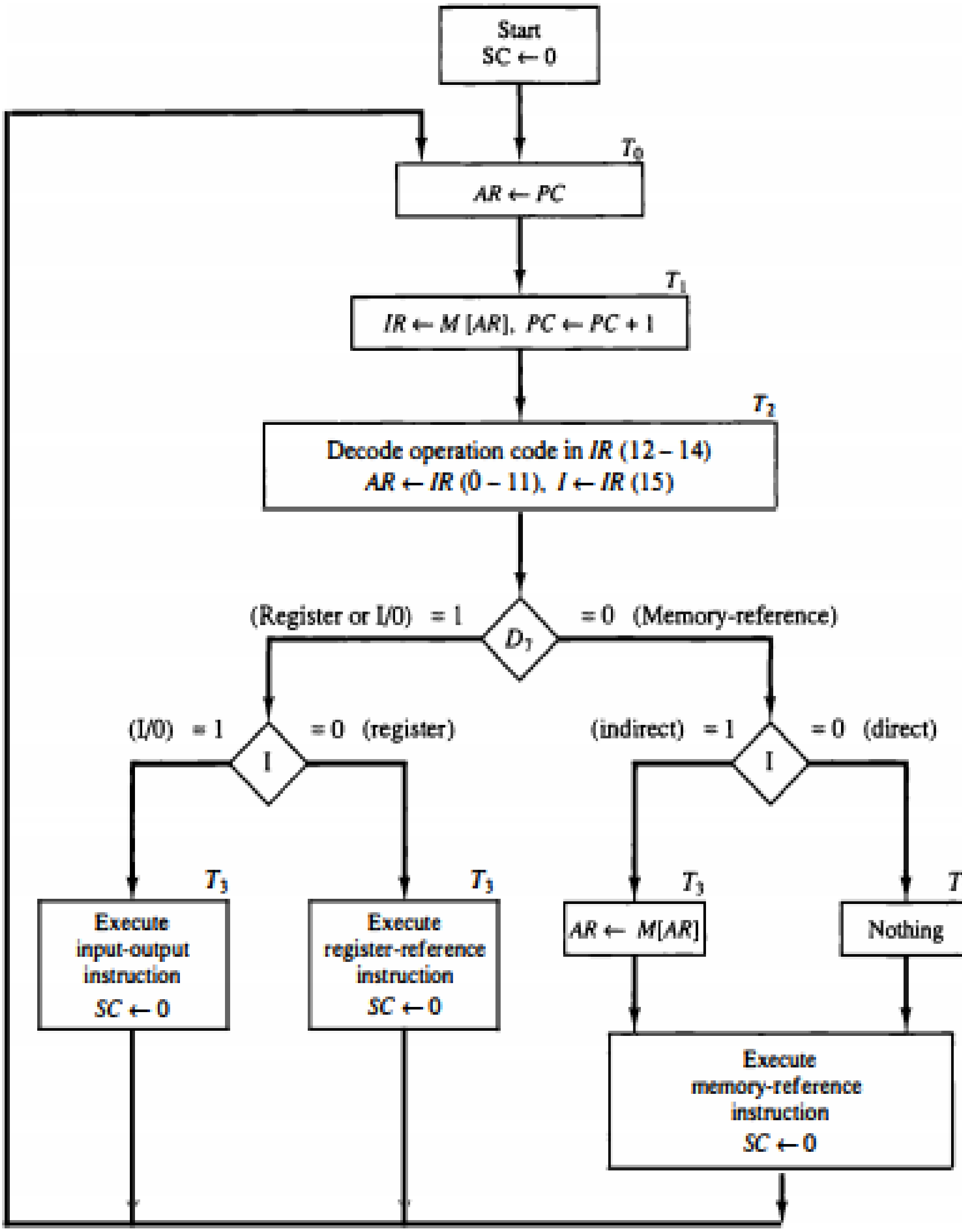


Figure Flowchart for instruction cycle (initial configuration).

Micro Operations for Fetch and Decode

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 (15)

1. Start the program, SC ← 0 were timing pluse start with T_0.

2. Whatever the address in the program counter (PC) is stored in address register AR ie, T_0 : AR ← PC

3. Content of memory address of address register will be placed in instruction register (IR). The instruction register holds the op code. After program counter is incremented

Ie, T_1 : IR ← M [AR], PC ←PC + 1

- During T_0 and T_1 the fetching of data from memory occurs.

4. Decode the operation code were op code is present in instruction register (IR)

Ie, T_2 : Decode the op code in IR

AR ← IR, 1 ← IR (15)

5. We need to execute the decoded instruction, here we to determine whether I bit is direct or indirect address.

6. After decision taking the execution will happen.

Execution can be of three types,

- Memory reference instruction
- Register reference instruction
- Input / Output reference instruction

The D_7 bit will decide the type instructions

7. Instruction execution completed.

BUS ORGANIZATION

A bus is a collection of wires that carries electrical current, the current is the information being passed between components .Typically the information is a data or program instruction, but can also include control signals and memory or I/O addresses.A bus organization is a group of conducting wires which carries information; all the peripherals are connected to microprocessor through the bus.

- The bus carries three types of information othe address from the CPU of the intended item to be accessed othe control information othe data, either being sent to the device, or from the device to CPU

There are 3 parts to a bus

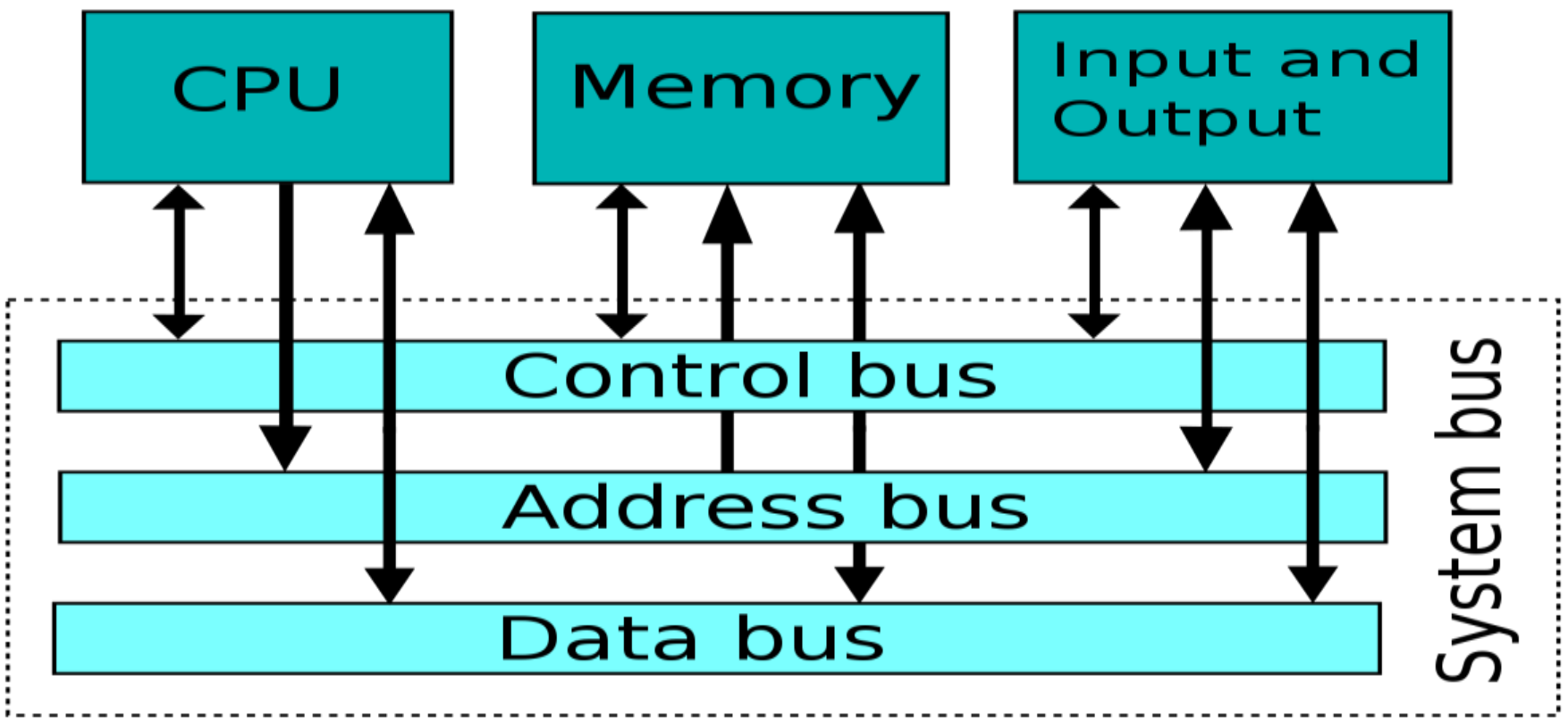
- the data bus (for both data and program instructions)
- the control bus (control signals from the Control Unit to devices)
- address bus (the address of the memory location or I/O device that is to perform the given data movement operation)

Additionally, computers may have multiple buses

- local bus – connects registers, ALU and control unit together
- system bus – connects CPU to main memory

•System Bus connects the CPU to memory and I/O devices

- expansion or I/O bus – connects system bus to I/O devices
- The CPU connects to the bus through pins
- The bus is on the motherboard inside the system unit
- Main memory (a collection of chips) connects to this bus through pins



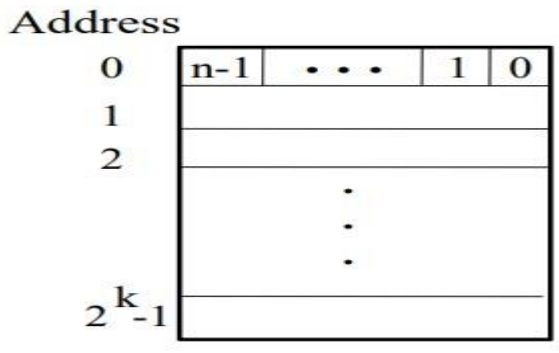
Memory location &Addresses

Memory

• Holds both instructions and data

• With k address bits and n bits per location, n is typically 8 (byte), 16 (word), 32 (long word)…

Memory locations and addresses determine how the computer's memory is organized so that the user can efficiently store or retrieve information from the computer.



Computer memory holds instructions and data. And a single bit is very small to hold this information so bits are rarely used individually. As a solution to this, the bits are grouped in fixed sizes of n bits. The memory of the computer is organized in such a way that the group of these n bits can be stored and retrieved easily by the computer in a single operation.The group of n bit is termed as *word* where n is termed as the word length. The word length of the computer has evolved from 8, 16, 24, 32 to 64 bits. General-purpose computers nowadays have 32 to 64 bits. The group of 8 bit is called a *byte*. Whenever you want to store any instruction or data may it be of a byte or a word you have to access a memory location. To access the memory location either you must know the memory location by its unique name or it is required to provide a unique address to each memory location. The memory locations are addressed from 0 to 2^K-1 i.e. a memory has 2^K addressable locations.

Byte Addressability

Most modern computers assign successive addresses to successive byte locations in memory. This assignment of addresses to individual byte locations is termed byte addressability and memory is referred to as byte-addressable memory.

Big-Endian and Little-Endian Assignments in Byte Addresses

The big-endian and little-endian are two methods of assigning byte addresses across the words in the memory.

Word Alignment

It is said that the word has *aligned addresses* if they begin with the byte address that is multiple of the number of bytes present in that word. For example, the word address 4 has four bytes in it with byte address 4, 5 and 6.