GraphicEra
(Deemed to be University)
Accredited by NAAC with Grade A

MCA I

Computer Organization and Architecture

By

Jaishankar Bhatt

Assistant Professor

Graphic Era Deemed to be University

Dehradun

Unit 2

Topic Name – Register Transfer & Micro-operations

Table of Contents

Computer Organization: - Computer organization is concerned with the way the hardware components operate and the way they are connected together to form the computer system. The various components are assumed to be in place and the task is to investigate the organizational structure to verify that the computer parts operate as intended.

Computer architecture: - It is concerned with the structure and behavior of the computer as seen by the user. It includes the information formats, the instruction set, and techniques for addressing memory. The architectural design of a computer system is concerned with the specifications of the various functional modules, such as processors and memories, and structuring them together into a computer system.

Functions of a basic Computer System: - Both the structure and functioning of a computer are, in essence, simple. In general terms, there are only four basic functions that a computer can perform:

- Data Processing
- Data Storage
- Data Movement
- Control

Structure of a Basic Computer System: - There are four main structural components:

- **Central processing unit (CPU):** often simply referred to as **processor**.
- **Main memory:** Stores data.
- **I/O:** Moves data between the computer and its external environment.
- **System interconnection:** Some mechanism that provides for communication among the control unit, ALU, and registers.

Register Transfer Language : - The symbolic notation used to describe the micro-operation transfers among registers is called a register transfer language.

Micro-operation: - A micro-operations is an elementary operation performed on the information or data stored in one or more registers. E.g. if R1 and R2 are two register of 4 bits each. If we add the contents of R1 and R2 and store the sum in R1, then it is known as add micro-operation. Similarly we can perform various micro operation with the contents of registers. Various types of micro operations are arithmetic micro operation, logic micro operation and shift micro operation etc. We can express an micro operation as follows

$$R1 \leftarrow R2+R3$$

Where R1, R2 and R3 are register of n bits and symbol ← is known as replacement operator which is used to denote information transfer from one register to another.

Micro instruction, micro program and micro operation: - To representation an micro operation , we need a micro instruction. Set of micro instruction is called micro program. Operation applied on the data stored in a register is called a micro operation.

Basic Symbols used in Register Transfer language

Symbol	**Description**	**Example**
Letters and (and numerals)	Denotes a register	MAR, R2
Parentheses ( )	Denotes a part of a register	R2(0-7), R1(8-15)
Arrow ←	Denotes transfer of information	R2 ← R l
Comma ,	Separates two micro-operations	R2 ← R l , R l ← R 2

MEMORY TRANSFER: -The transfer of information from a memory word to the outside environment is called a read operation. The transfer of new information to be stored into the memory is called a write operation. Two register are used during the memory transfer.

Data Register:- Data register holds data during memory transfer.

Address Register: - Address register holds address of memory location from where data transfer takes place.

A memory word will be symbolized by the **letter M**.

e.g. **M[AR]** species the memory word selected by the data register.

M[AR]

Memory Read: - Consider a memory unit that receives the address from a register, called the address register, symbolized by AR. The data are transferred to another register, called the data register, symbolized by DR. The read operation can be stated as follows:

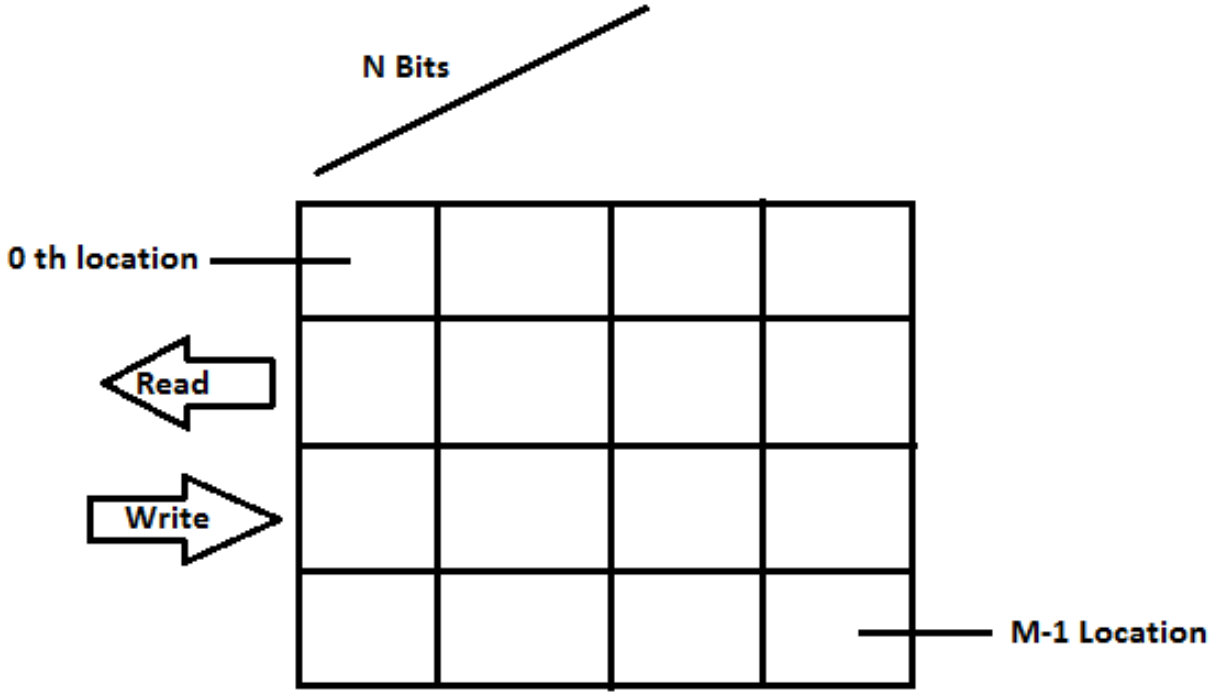
Read: DR ← M [AR]

This causes a transfer of information into DR from the memory word M selected by the address in AR.

Memory Write: - The write operation transfers the content of a data register to a memory word M selected by the address. Assume that the input data are in register R1 and the address is in AR. The operation can be stated symbolically as follows:

Write: M[AR] ←R1

This causes a transfer of information from R1 into the memory word M selected by the address in AR.



Memory of a Basic Computer
Memory Size M X N

Where M = Total number of memory locations or total number of memory words.

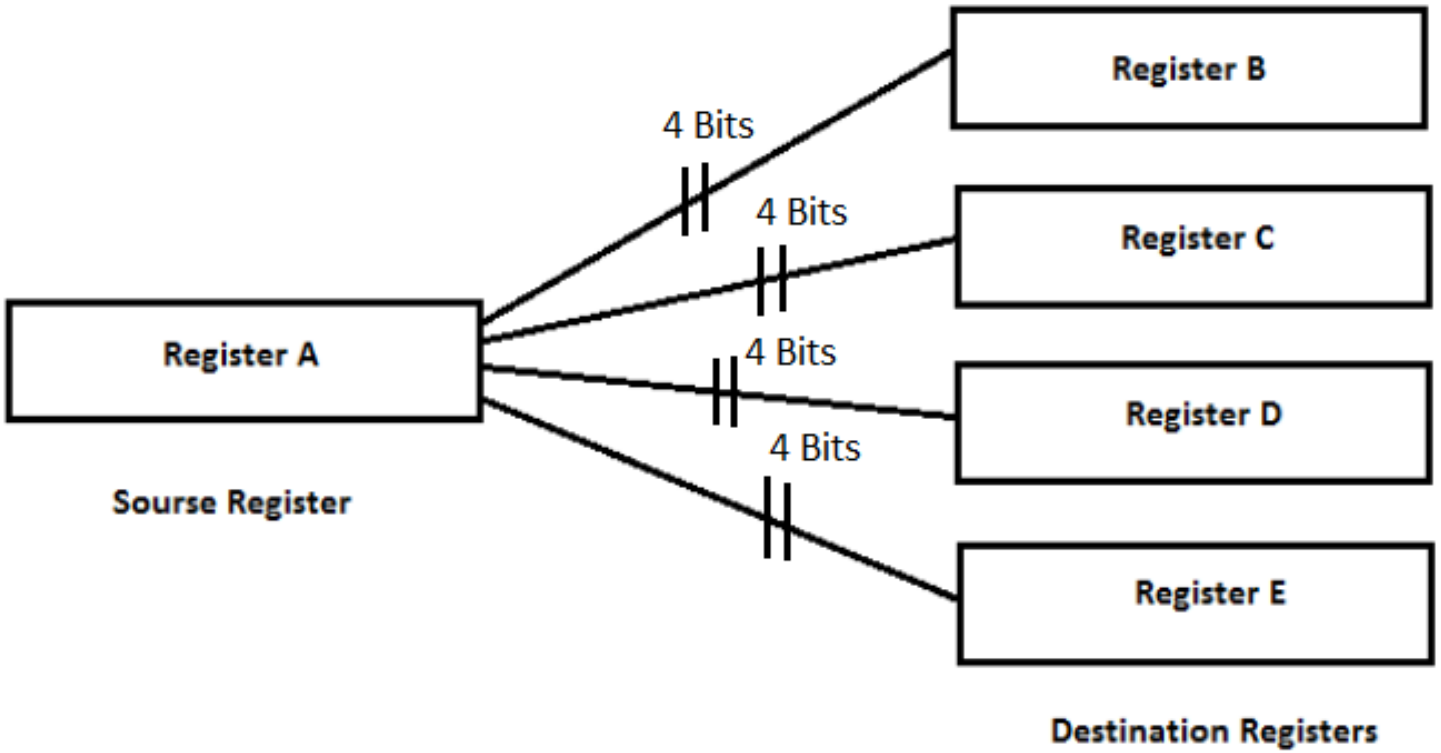
N = Number of bits in each locations (word length)

Topic Name – Common Bus System

Table of Contents

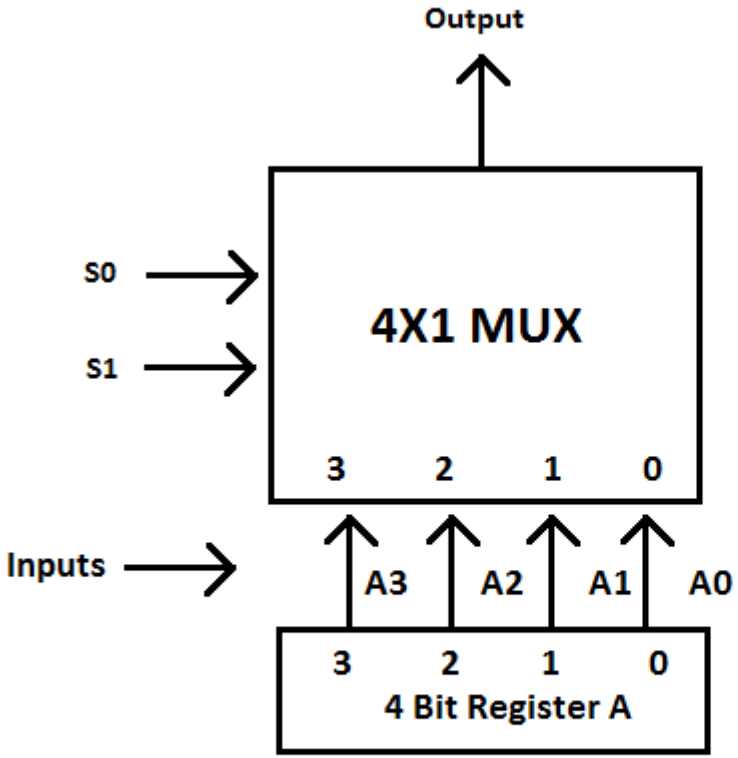
Common Bus System: - A typical digital computer has many registers, and paths must be provided to transfer information from one register to another. These paths are known as “BUS”. These communication pathways or buses are used to carry various types of signals. The number of wires or paths will be excessive if separate lines are used between each register and all other registers in the system. A more efficient scheme for transferring information between registers in a multiple-register configuration is a common bus system.

A bus structure consists of a set of common lines, one for each bit of a register, through which binary information is transferred one at a time. Control signals determine which register is selected by the bus during each particular register transfer.



A Dedicated Bus System

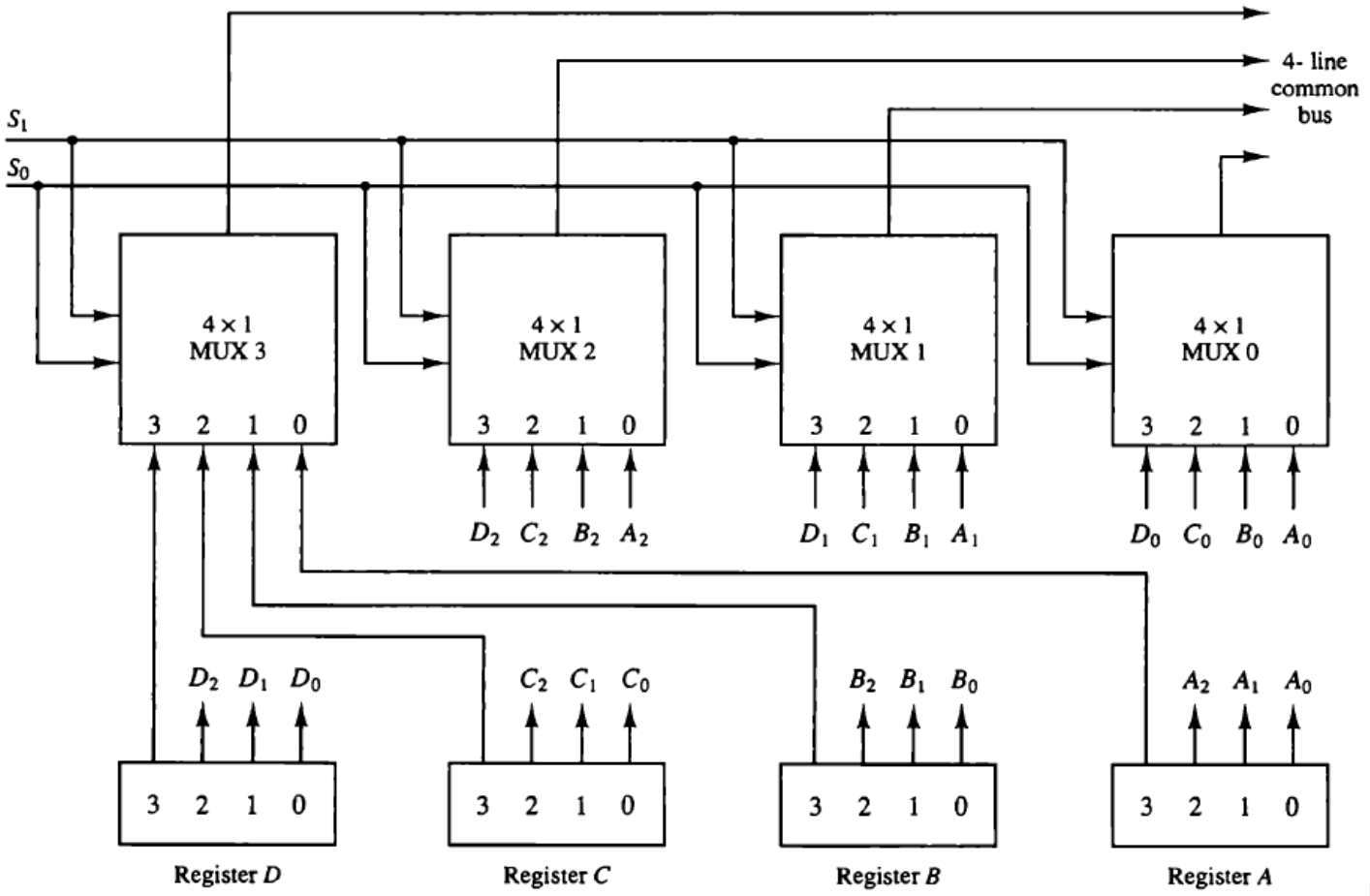
Use of a Multiplexer in a Common Bus System: - Multiplexer is a combinational circuit which produces only one output among 2^n inputs. One output is selected with the help of selection lines.



S1	S0	Output
0	0	A0
0	1	A1
1	0	A2
1	1	A3

Truth Table

4 X 1Multiplexer



Common bus system of 4 register of 4 bits each using multiplexers

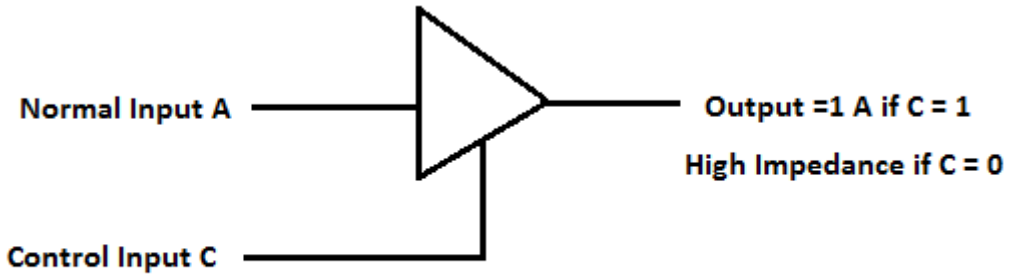
The two selection lines S_1 and S_0 are connected to the selection inputs of all four multiplexers. The selection lines choose the four bits of one register and transfer them into the four-line common bus. When $S_1S_0 = 00$, the 0 data inputs of all four multiplexers are selected and applied to the outputs that form the bus. This causes the bus lines to receive the content of register A since the outputs of this register are connected to the 0 data inputs of the multiplexers. Similarly, register B is selected if $S_1S_0 = 01$, and so on.

S1	S0	Output
0	0	Register A
0	1	Register B
1	0	Register C
1	1	Register D

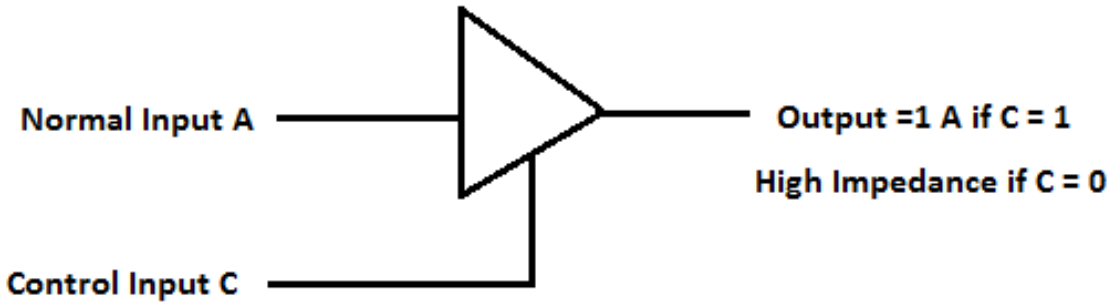
Truth Table

Three-State Bus Buffers: -

A bus system can be constructed with three-state gates instead of multiplexers. A three-state gate is a digital circuit that exhibits three states. Two of the states are signals equivalent to logic 1 and 0 as in a conventional gate. The third state is a high-impedance state. The high-impedance state behaves like an open circuit, which means that the output is disconnected and does not have a logic significance. Three-state gates may perform any conventional logic, such as AND or NAND. However, the one most commonly used in the design of a bus system is the buffer gate.



Graphic Symbol for Three State Buffer



Graphic Symbol for Three State Buffer

Control Input C	Normal Input A	Output Y
o	0	High Impedance
	1	
1	0	0
	1	1

Truth Table

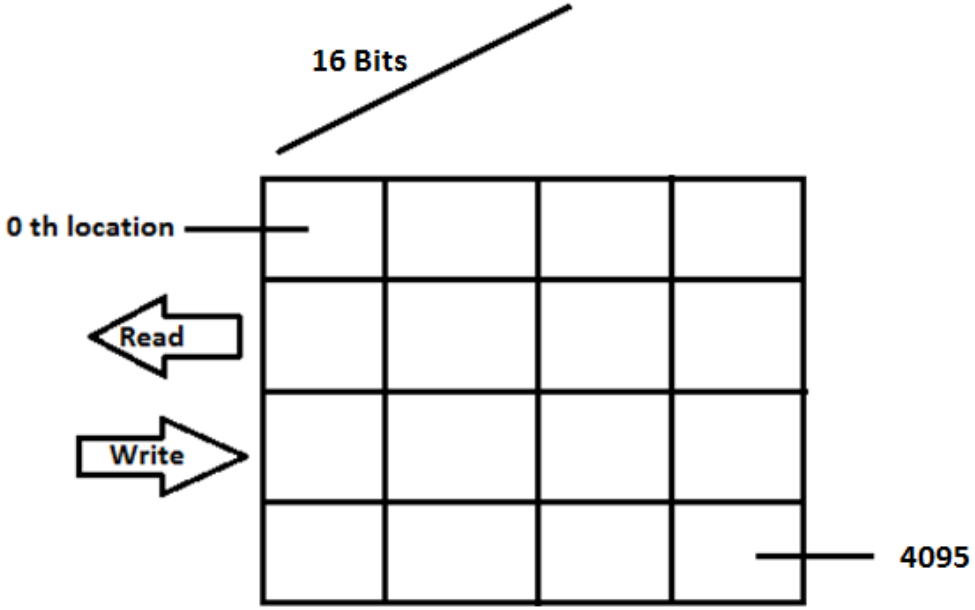
Summary

- Computer registers are designated by capital letters.
- Symbols are a handy tool to represent the micro-operation sequences in registers and the control functions that initiate them, in a lucid & concise form. Such a symbolic notation is referred to as register transfer language.
- A bus structure consists of a set of common lines, one for each bit of a register, through which binary information is transferred one at a time. Control signals determine which register is selected by the bus during each particular register transfer.
- A micro-operation is an elementary operation performed with the data stored in registers.
- These micro-operations perform some basic arithmetic operations on the numeric data stored in the registers.

Topic Name – CPU Registers and Instruction Formats

Table of Contents

Computer Registers: - The memory size of a basic computer is 4096 X 16 i.e. there are 4096 memory locations and each location can hold 16 bits of data. 4096 = 2^{12} i.e. in this memory model, each memory location will have 12 bits of memory address. To access a single memory location CPU will require 12 bits of memory address and in each location 16 bits of data can be stored.



Memory of a Basic Computer
Memory Size 4096 X 16

According to the memory used in a basic computer i.e. 4096X16, various special purpose registers available in a basic computer are as follows.

Register symbol	Number of bits	Register Name	Register Function
DR	16	Data register	Holds memory operands
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 next instruction
TR	16	Temporary register	Holds temporary data
INPR	8	Input register	Holds input character
OUTR	8	Output register	Holds output character

Basic Computer Registers

How to calculate the size of CPU registers if the memory size is given: -

Q. Calculate the size of various CPU registers (i.e. DR, AR, PC and AC etc.) if the memory size is 8KX16.

Sol. Memory Size is 8K X 16. It means there are total 8K locations in memory and each memory location can store 16 bits of data.

We can write 8K as 8 X 1024 (K=1024)

$$8 \text{ X } 1024 = 2^3 \text{ X } 2^{10} = 2^{13}$$

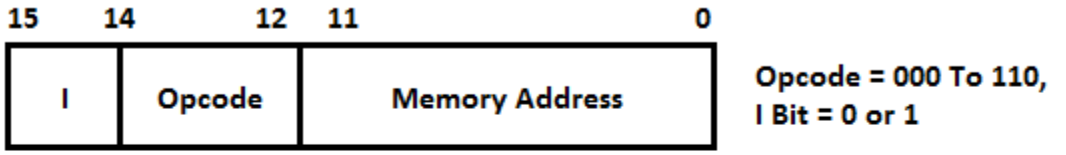
$8K = 2^{13}$ it means 13 bits of memory address is required to access a single memory location. So the memory address will be of 13 bits. Because the memory size is 8K X 16 i.e. one memory location can hold 16 bits of data, so the size of data will be of 16 bits. According to 8K X 16 size of various CPU registers will be as follows.

AR = 13 Bits, DR = 16 Bits, PC = 13 Bits, AC = 16 Bits.

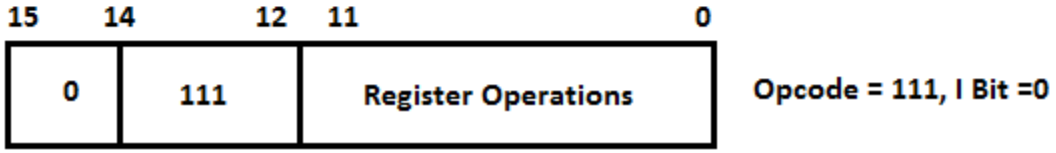
Computer Instructions: - An instruction is a command given to a computer to perform an operation on some data. There are three types of instructions and each of these instructions consists of three parts. There are three types of instruction used in a basic computer, which are as follows.

- Memory Reference Instruction: - Operand is available in a memory locations
- Register Reference Instruction : - Operand is available inside a register
- Input Out Instruction : - Specifies I/O operations

Basic Computer Instruction Formats: - A computer will usually have a variety of instruction code formats. The format of an instruction is usually depicted in a rectangular box symbolizing the bits of the instruction as they appear in memory words. The bits of the instruction are divided into groups called fields.



(a) Memory Reference Instruction



(b) Registration Reference Instruction

(c) Input Output Instruction

Bacic Computer Instruction Formats

1. **Memory Reference Instruction: -** Memory reference instruction are those instruction where the operand to be operated upon is fetched from memory. Following are the various types of memory reference instructions.

Instruction	Description
AND	AND memory word to AC
ADD	Add memory word to AC
LDA	Load memory word to AC
STA	Store content of AC in memory
BUN	Branch unconditionally
BSA	Branch and save return address
ISZ	Increment and skip if zero

2. Register Reference Instruction: - Register reference instructions are those instructions in which the operand to be operated upon is already present in a register. In such cases, the op code = 111, I=0 and the rest of 12 bits represent the operation to be performed. Following are the various types of register reference instructions

Instruction	Description
CLA	Clear Ac
CLE	Clear E
CMA	Complement AC
CME	Complement E
CIR	Circulate right AC and E
CIL	Circulate left AC and E
INC	Increment AC
SPA	Skip next instruction if AC is positive
SNA	Skip next instruction if AC is negative
AZA	Skip next instruction if AC is zero
SZE	Skip next instruction if AC is E is zero
HLT	Halt computer

3. Input Output Instruction: - Input out instruction are those instructions that are used for interacting with user, i.e. getting data from user and or showing data to the user on a monitor or printer. . Following are the various types of input output instructions

Instruction	Description
INP	Input character to AC
OUT	Output character to AC
SKI	Skip on input flag
SKO	Skip on output flag
ION	Interrupt on
IOF	Interrupt off

Topic Name – Fetch and Decode

Table of Contents

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. Each instruction cycle in turn is subdivided into a sequence of sub cycles or phases. 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 to 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_o. 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 micro-operations 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)

Description of fetch and decode phase: - Initially PC holds the address of the first instruction which is to be executed.

T_0: AR ← PC **----------1**

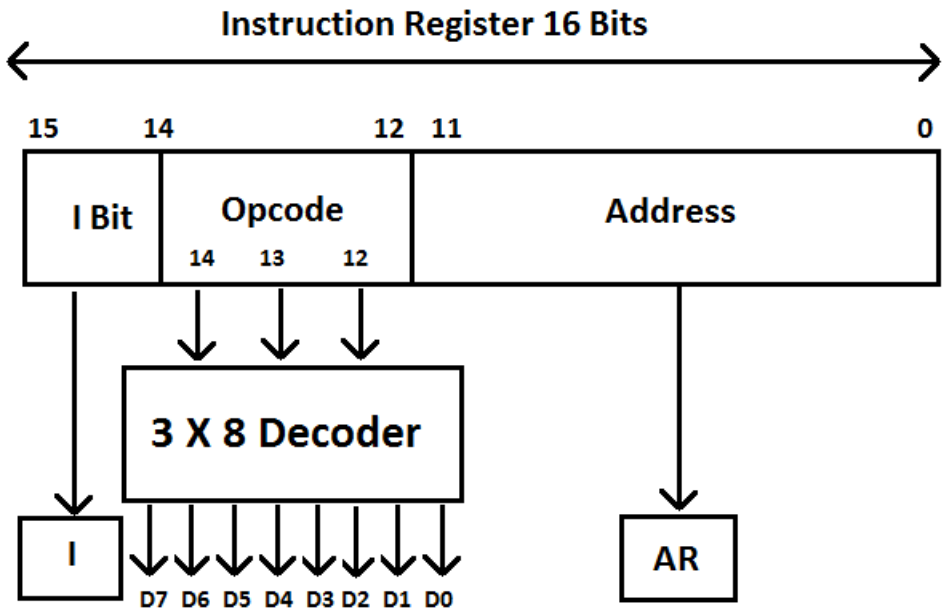
This instruction transfers the contents of PC (Program Counter) to the address register (AR) i.e. the address of the first instruction, which is to be executed first is transferred into the AR.

T_1: IR ←M[AR], PC ← PC + 1 **---------- 2**

The second instruction transfers the contents of the memory location into IR, whose address is available in AR. It means the instruction whose address is available in AR is transferred into IR.

T_2: D_0, • • • , D_7 ← Decode IR(12-14), AR ← IR(0-11), I ← IR(l5) **----------3**

In decode phase the 12, 13 and 14^{th} bits of instruction i.e. opcode is decoded through a 3X8 decoder. 0 to 11 bits are transferred into AR and 15^{th} bit is transferred into I flip flop (I is a 1 Bit flip flop).

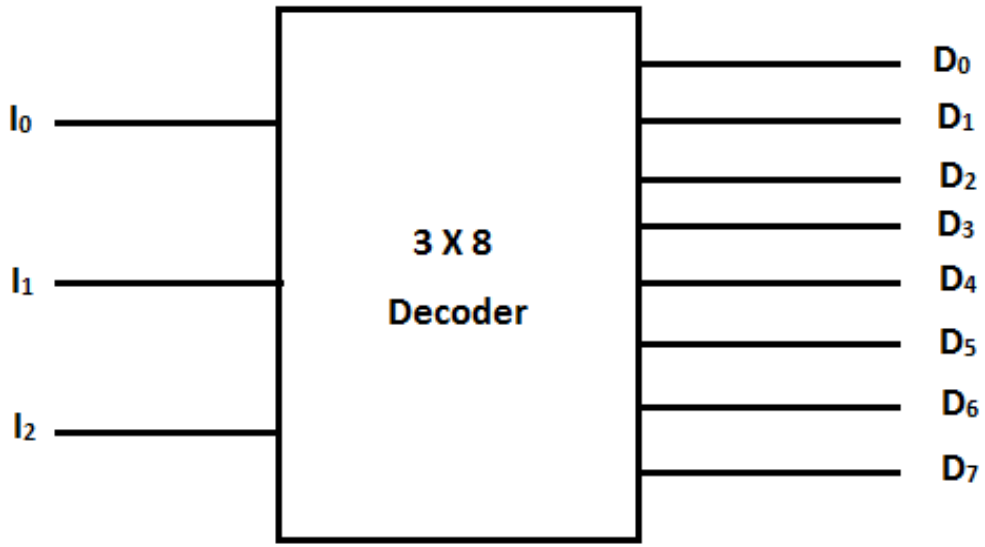


Instruction Decoding

1. 0 to 11th bits of Instruction register (IR) are transferred to address register (AR).

2. Opcode part i.e. 12, 13 and 14th bits are transferred as an input to 3X8 decoder.

3. Last bit i.e. 15th bit is passed to I flip flop.



Block Diagram of 3 X 8 Decoder

Inputs			Outputs							
I_2	I_1	I_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Truth Table of a 3X8 Decoder

Topic Name – Instruction Cycle

Table of Contents

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. Each instruction cycle in turn is subdivided into a sequence of sub cycles or phases. 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 to step 1 to fetch, decode, and execute the next instruction. This process continues indefinitely unless a HALT instruction is encountered.

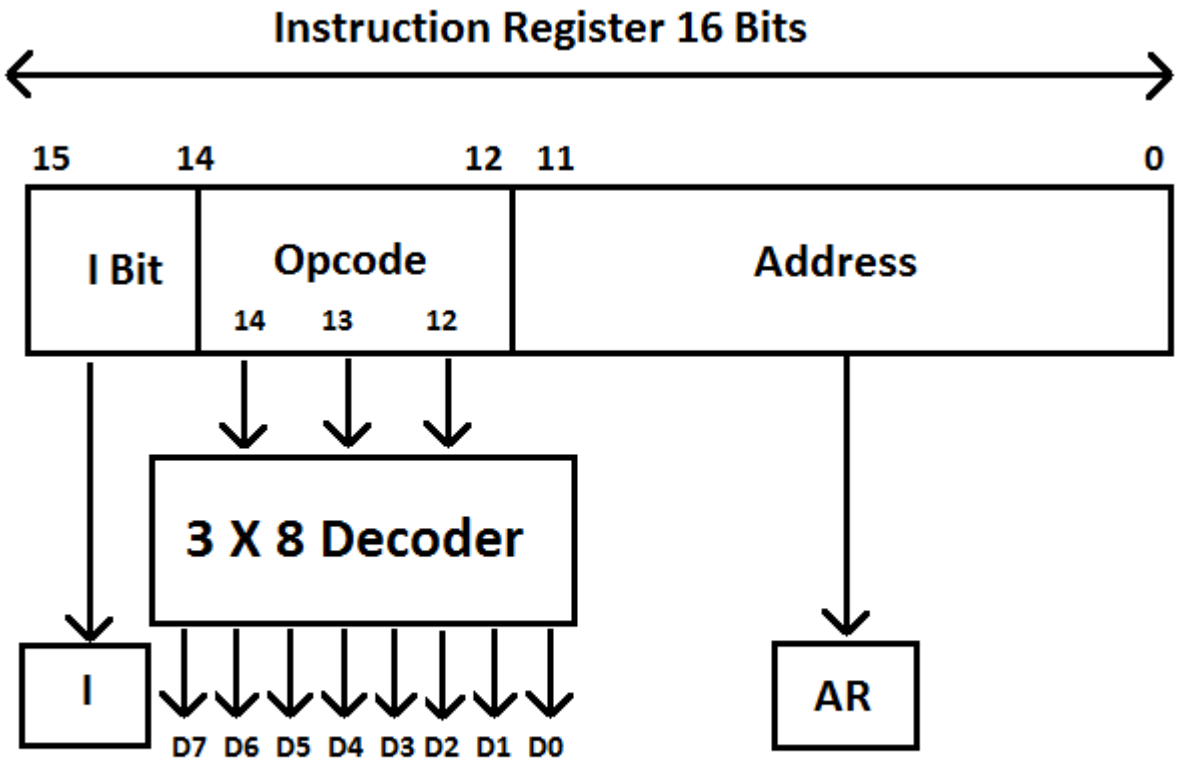
Fetch and Decode: -

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_o. 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 micro-operations 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

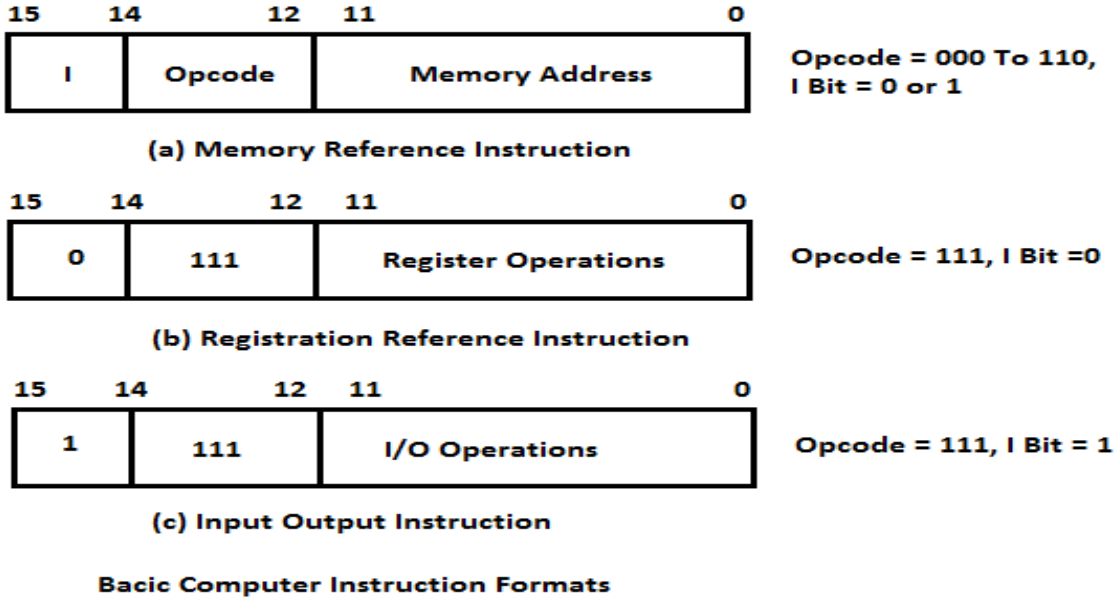
T_2: D_0, • • • , D_7 ← Decode IR(12-14), AR ← IR(0-11), I ← IR(l5)



Instruction Decoding

Inputs			Outputs							
I_2	I_1	I_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Truth Table of a 3X8 Decoder



Bacic Computer Instruction Formats

Abbreviations used

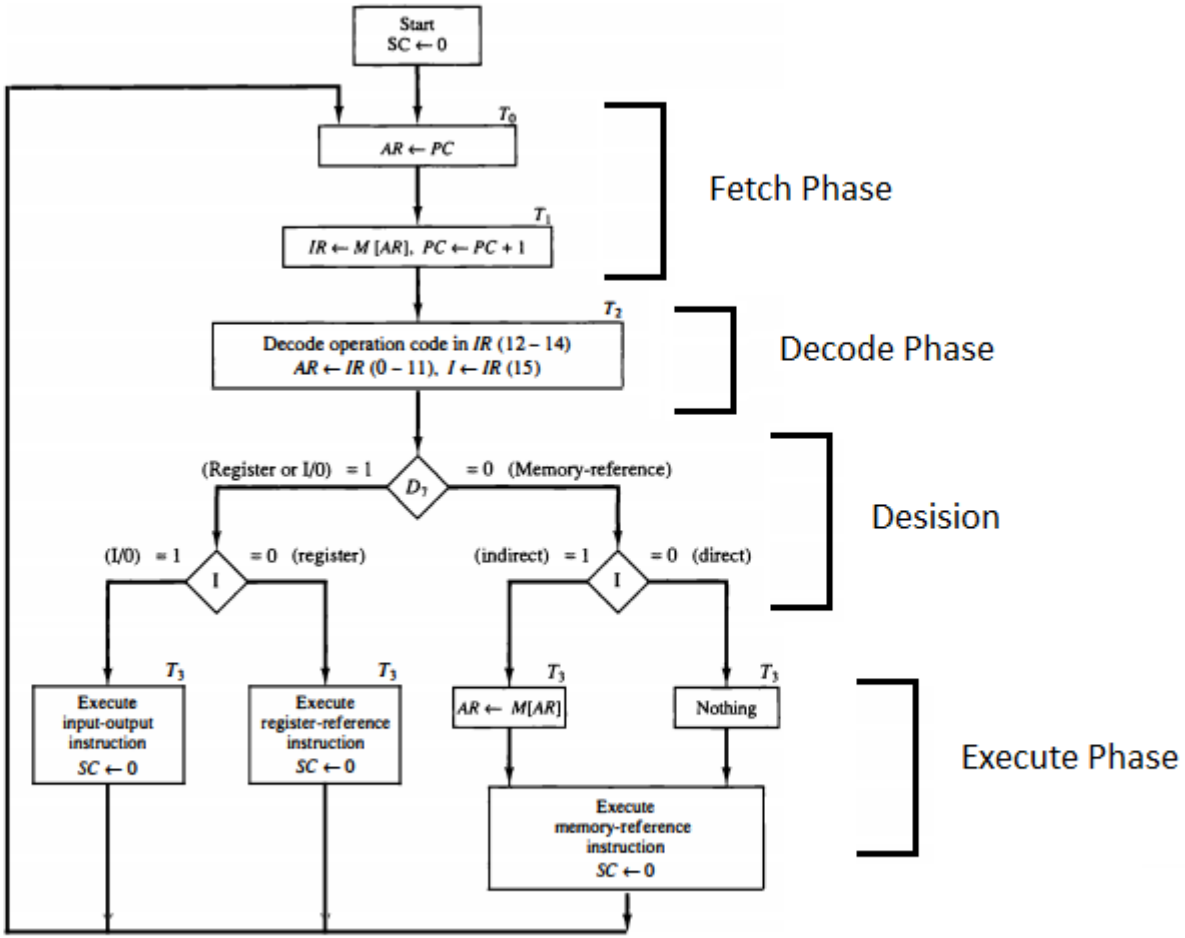
SC - 4 Bit Sequence Counter

AR - Address Register

PC - Program Counter

D_7 - Output of 3X8 decoder

I - 1 bit flip flop



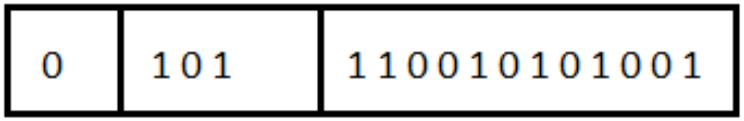
Flow Chart for instruction cycle

Decoder output D_7 is equal to 1 if the operation code is equal to binary III. From the fig. of flowchart of instruction cycle we determine that if D_7 = I, 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 micro-operation for the indirect address condition can be symbolized by the register transfer statement

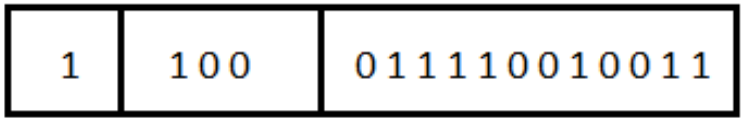
Examples: -

Instruction 1



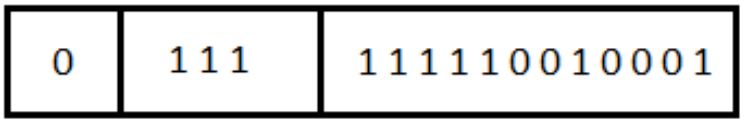
Opcode = 1 0 1 , I Bit or Mode bit = 0
Instruction type = Memory reference Instruction
Memory Address = Direct

Instruction 2



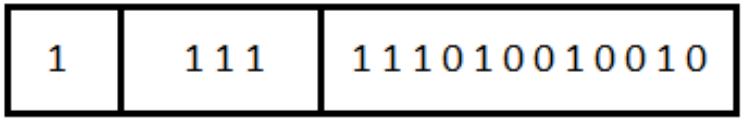
Opcode = 1 0 0 , I Bit or Mode bit = 1
Instruction type = Memory reference Instruction
Memory Address = Indirect

Instruction 3



Opcode = 1 1 1, I Bit = 0
Instruction type = Register reference Instruction

Instruction 4



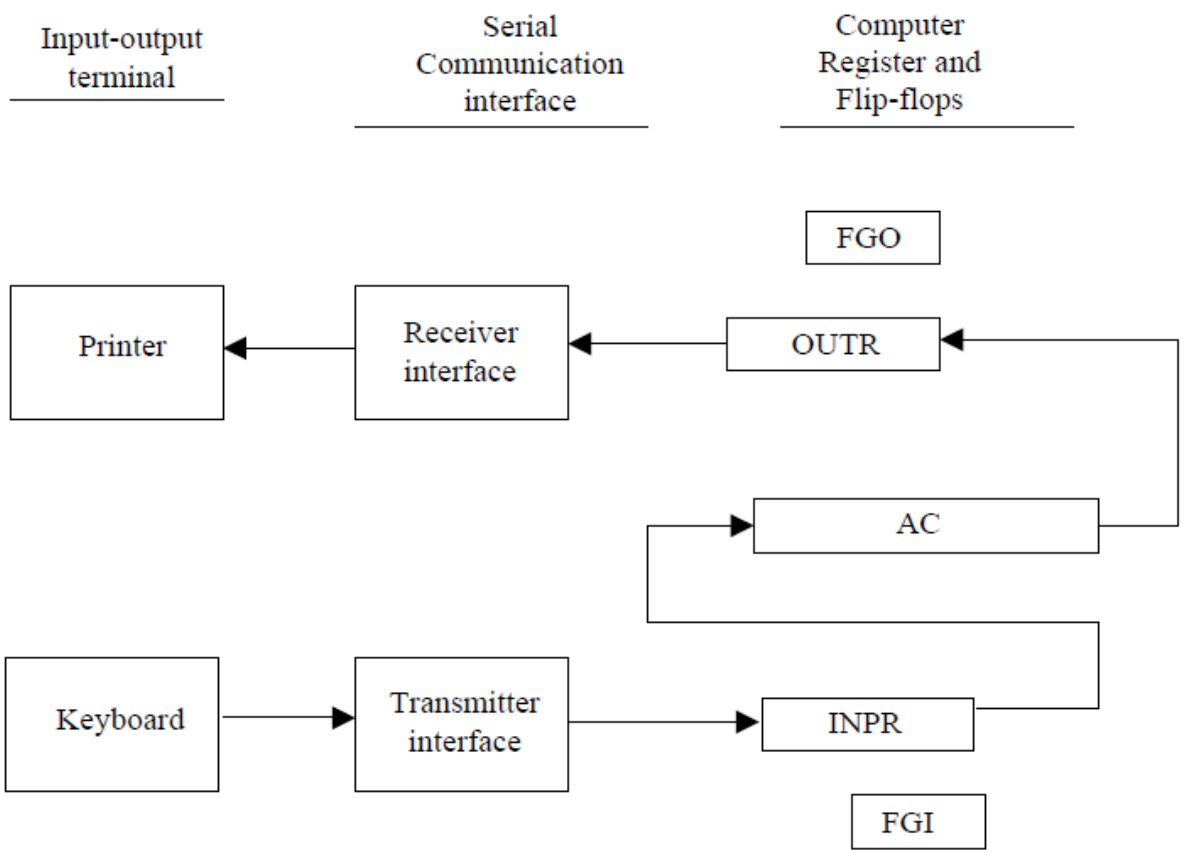
Opcode = 1 1 1, I Bit = 1
Instruction type = Input/Output Instruction

Topic Name : Input Output Organization/Configuration

Table of Contents

Input Output Configuration:- Input output configuration describe that how single character is transferred in from input device and transferred out and to an output device. The input/output terminals sends and receives serial information. Each quantity of information has eight bits of an alphanumeric code. The serial information from the keyboard is shifted into the input register INPR.

The serial information for the printer is stored in the output register OUTR. These two registers communicate with a communication interface serially and with the AC in parallel. The transmitter interface receives serial information from the keyboard and transmits it to INPR. The receiver interface receives information from OUTR and sends it to the printer serially.



Block diagram of Input Output Configuration

INPR – 8 Bit Input Register holds input characters

OUTR – 8 Bit Output Register holds output characters

FGI – Flag Input (1 Bit Flag), it indicates availability of input characters

FGO – Flag Output (1 Bit Flag), it indicates availability of output characters

AC – Accumulator register (processor register)

Various registers used in input output Configuration

Description of diagram: - Initially, the input flag FGI is cleared to 0. When a key is struck in the keyboard, an 8-bit alphanumeric code is shifted into INPR and the input flag FGI is set to 1. As long as the flag is set, the information in INPR cannot be changed by striking another key. The computer checks the flag bit; if it is 1, the information from INPR is transferred in parallel into AC and FGI is cleared to 0. Once the flag is cleared, new information can be shifted into INPR by striking another key.

.

The output register OUTR works similarly but the direction of information flow is reversed. Initially, the output flag FGO is set to 1. The computer checks the flag bit, if it is 1, the information from AC is transferred in parallel to OUTR and FGO is cleared to 0. The output device accepts the coded information, prints the corresponding character, and when the operation is completed, it sets FGO to the computer does not load a new character into OUTR when FGO is 0 because this condition indicates that the output device is in the process of printing the character.

Topic Name – Addressing Mode

Table of Contents

- Addressing Mode
- Types of addressing modes
- References

Addressing Modes: - Addressing modes specifies the way, the effective address of an operand is represented in the instruction or the different ways in which operands can be addressed is called the addressing mode.

Effective address is the address of the exact memory location where the value of the operand is present.

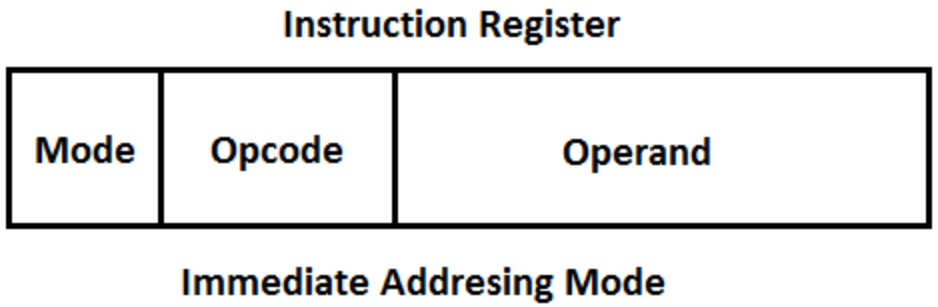
Various types of addressing modes are as follows.

1. Implied addressing mode
2. Immediate addressing mode
3. Register addressing mode
4. Register indirect addressing mode
5. Auto increment/
6. Auto decrement
7. Direct addressing mode
8. Indirect addressing mode
9. Relative addressing mode
10. Indexed addressing mode
11. Base Register addressing mode

1. Implied Mode: In this mode the operands are specified implicitly in the definition of the instruction. For example, the instruction **CMA** (complement accumulator) is an implied-mode instruction because the operand in the accumulator register is implied in the definition of the instruction.

2. 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. The operand field contains the actual operand to be used in conjunction with the operation specified in the instruction. Immediate mode instructions are useful for initializing registers to a constant value.

E.g. MVI A 2050



The **advantage** of immediate addressing is that no memory reference other than the instruction fetch is required to obtain the operand, thus saving one memory or cache cycle in the instruction cycle.

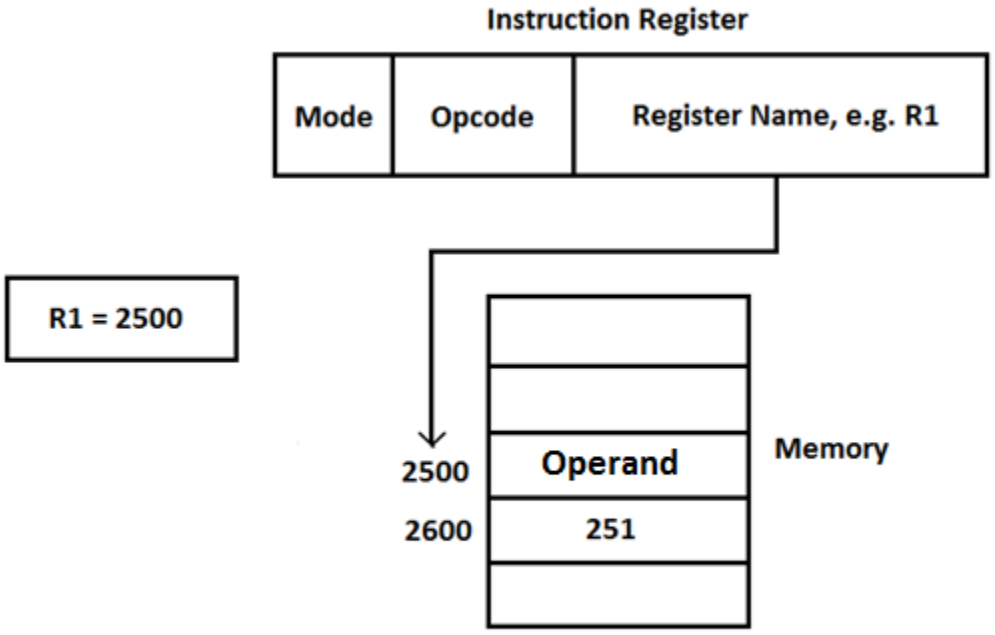
The **disadvantage** is that the size of the number is restricted to the size of the address field, which, in most instruction sets, is small compared with the word length.

3. 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. E.g. ADD R1

The **advantages** of register addressing are that (1) only a small address field is needed in the instruction, and (2) no time consuming memory references are required. The memory access time for a register internal to the processor is much less than that for a main memory address.

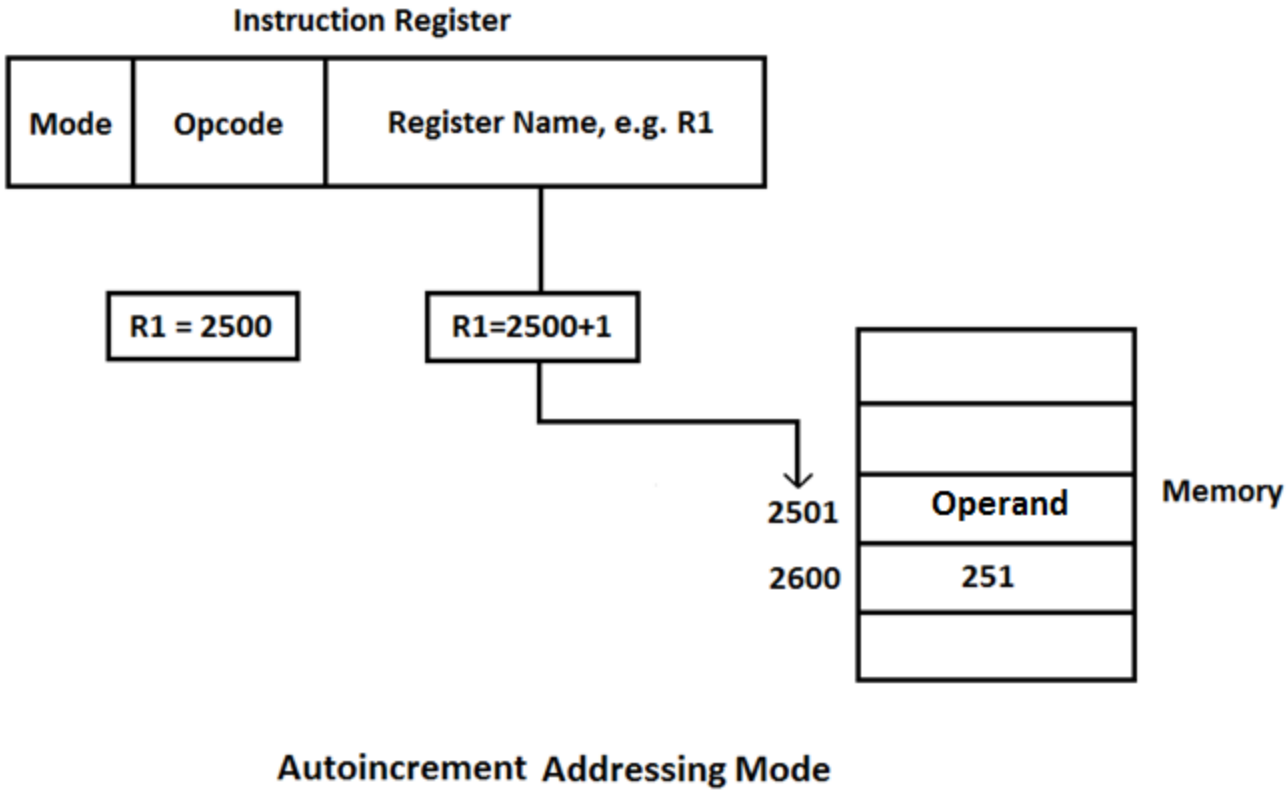
The **disadvantage** of register addressing is that the address space is very limited.

4. 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. A big advantage of register indirect addressing is that it can reference memory without paying the price of having a full memory address in the instruction.



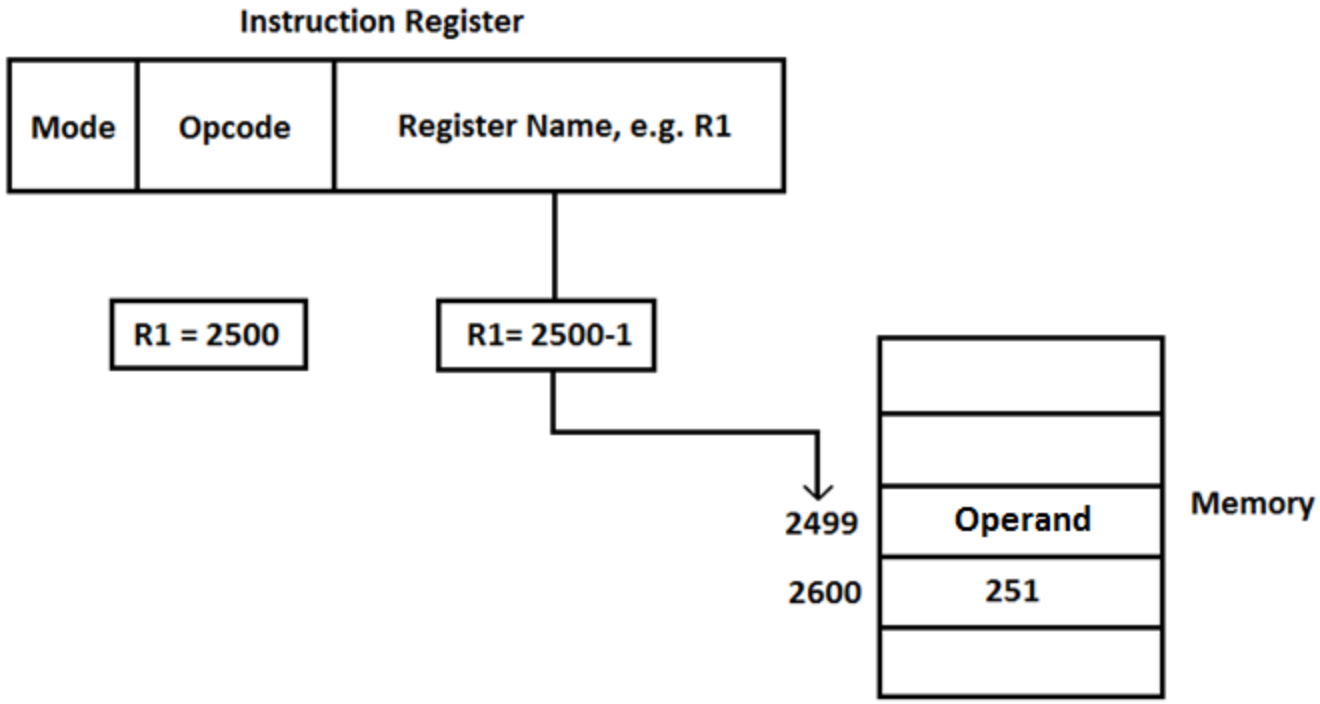
Register Indirect Addressing Mode

5. Auto-increment Mode: - This addressing mode is similar to the register indirect addressing mode in the sense that the effective address of the operand is the content of a register. However, with auto-increment, the content of the register is automatically incremented after accessing the operand.



Autoincrement Addressing Mode

6. Auto decrement Mode: - Similar to the auto increment mode, the auto decrement mode uses a register to hold the address of the operand. However, in this case the content of the auto decrement register is first decremented and the new content is used as the effective address of the operand.

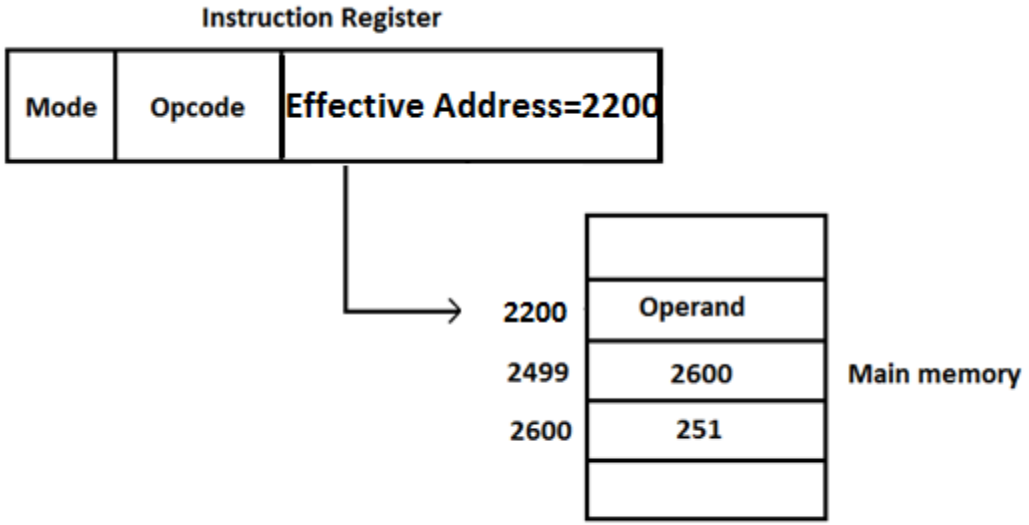


Autodecrement Addressing Mode

7. Direct addressing mode: - A very simple form of addressing is direct addressing, in which the address field contains the effective address of the operand. According to this addressing mode, the address of the memory location that holds the operand is included in the instruction.

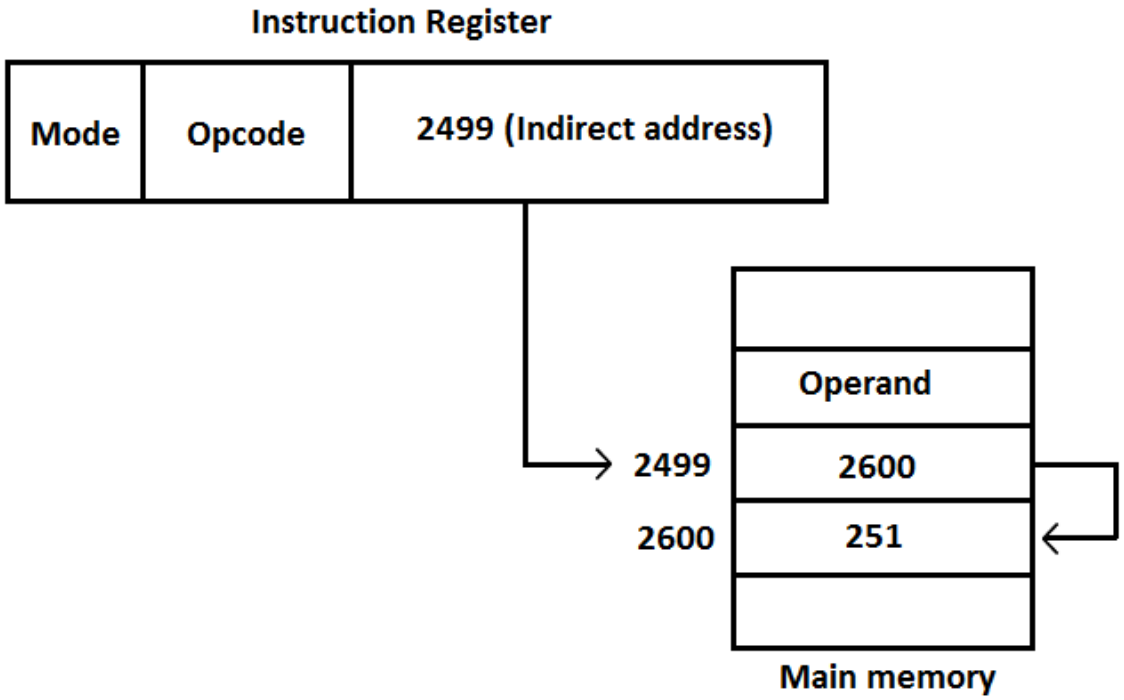
The **advantage** of this mode is that It requires only one memory reference and no special calculation.

The obvious **limitation or disadvantage** is that it provides only a limited address space.



Direct Addressing Mode

8. Indirect addressing mode: - In this mode the address field of the instruction gives the address where the effective address is stored in memory. In the indirect mode, what is included in the instruction is not the address of the operand, but rather a name of a register or a memory location that holds the (effective) address of the operand.



Indirect Addressing Mode

The obvious a**dvantage** of this approach is that for a word length of *N*, an address space of 2^N is now available.

The **disadvantage** is that instruction execution requires two memory references to fetch the operand: one to get its address and a second to get its value.

9. 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.

10. 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.

11. 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.

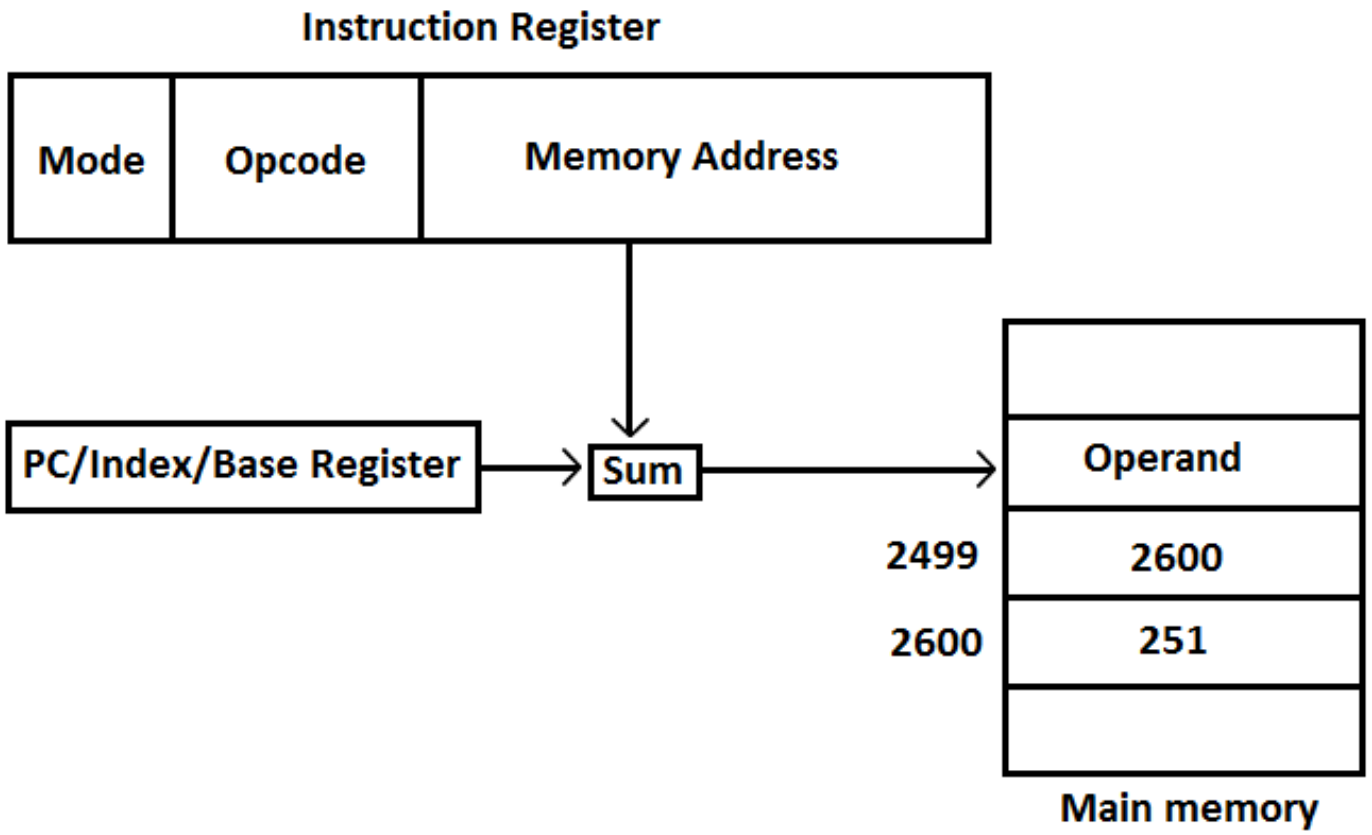


Illustration of Relative/Index/Base Register addressing Mode

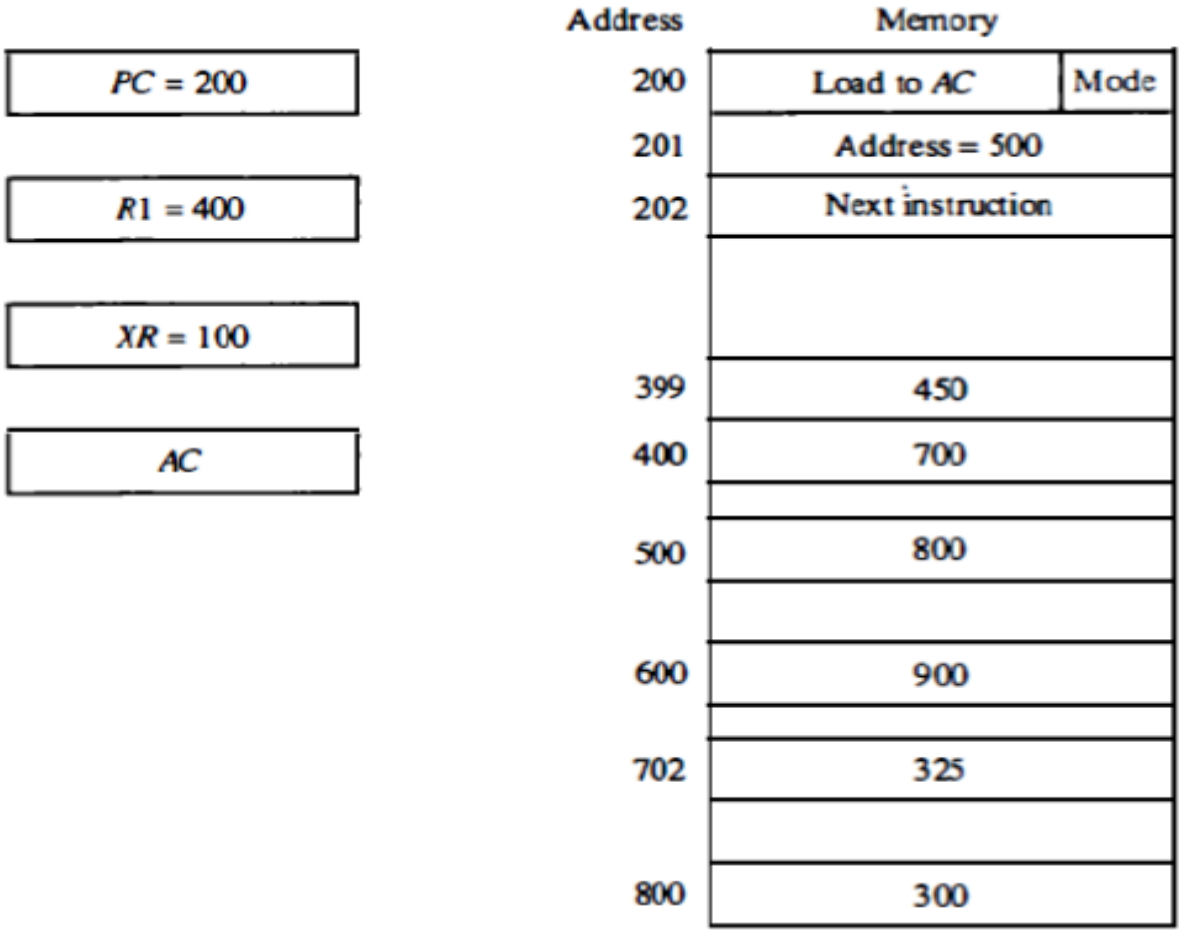
Topic Name : Numerical on Addressing Modes

Table of Contents

Numerical 1:- The two-word instruction at address 200 and 201 is a "**load to AC**" (**LDA**) instruction with an address field equal to 500. The first word of the instruction specifies the operation code and mode, and the second word specifies the address part. PC has the value 200 for fetching this instruction. The content of processor register R 1 is 400, and the content of an index register XR is 100. AC receives the operand after the instruction is executed.

A two word instruction means an instruction which occupies the two memory locations. Half part of an instruction stored in one memory location and another part of the instruction stored in another location.

An instruction has 3 parts – address part, opcode part and I bit or mode bit.



In the above diagram instruction is LDA (Load to AC) and it is stored in two memory locations. Opcode part and mode part stored in memory location 200 and address part stored in memory location 201. Next instruction is stored at location 202.

The table shown below lists the values of the effective address and the operand loaded into AC for the nine 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

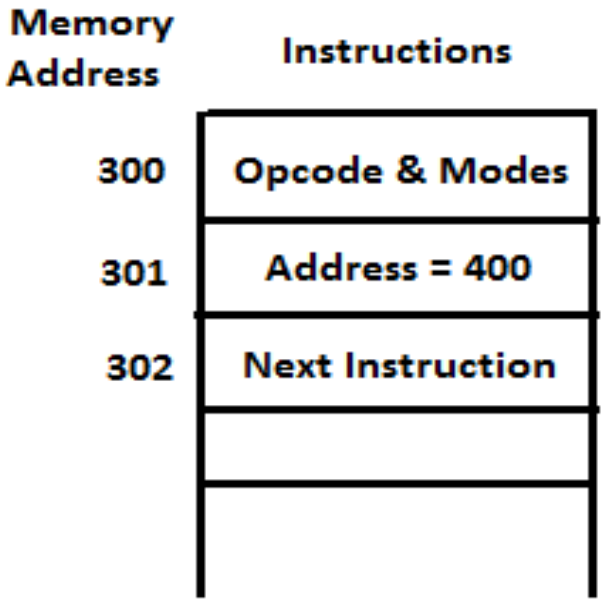
Effective address is the actual address of the operand in memory where it is stored.

Question 2. : - An instruction is stored at location 300 with its address field at location 301. The address field has the value 400. A processor register R1 contains the number 200. Evaluate the effective address if the addressing mode of the instruction is **(a)** direct **(b)** immediate **(c)** relative **(d)** register indirect **(e)** index with R1 as the index register.

Solution: - An instruction has three fields- **address field**, **opcode field** and **mode bit or I bit.** It is given in the question that address field of the instruction is stored in the memory location 301 and rest part (i.e. opcode part and I bit) is stored at 300. the address field contains a 400.

A register R1 is specified which contains 200. Now we have to calculate the effective address (**effective address is the address of the operand in memory**) for the various addressing modes.

The following diagram shows memory addresses and the instructions stored in various memory locations.



We can observe in the diagram that an instruction stored in memory locations 300 and 301. Opcode part and I bit is stored in memory location 300 and address part of the instruction is stored in 301. obviously the next instruction will stored in memory location 302.

Now we will calculate the effective address.

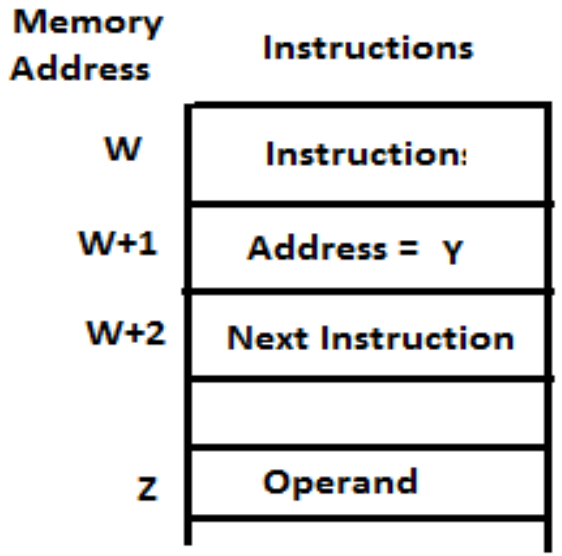
1. In case of direct addressing mode the effective address will be 400. because the address part has the value 400. (See definition of direct addressing mode).
2. In case of immediate address mode the effective address will be 301. Because in this mode address part of the instruction contains value not address. So 400 will be treated as operand or value.
3. In case or relative addressing mode the effective address will be, 400 (value at address part) + 302 (value of program counter PC) = 702.
4. In register indirect effective address will be 200. Because in this addressing mode register will contain address of the operand.
5. In Index addressing mode the effective address will be, 400 (value at address part) + 200 (value of index register) = 600.

Note : Read the definition of various addressing mode before solving this numerical.

Question 3. : - A two-word instruction is stored in memory at an address designated by the symbol W. The address field of the instruction (stored at W + 1) is designated by the symbol Y. The operand used during the execution of the instruction is stored at an address symbolized by Z. An index register contains the value X. State how Z is calculated from the other addresses if the addressing mode of the instruction is

a. Direct b. Indirect c. Relative d. Indexed

Solution: -



Solution: - (a) In direct addressing mode the address part of the instruction is the address of the operand. Thus Z = Y

(b) In indirect addressing mode the address part of the instruction gives the memory address which will be the effective address of the operand.

Thus Z = M[Y]

(c) In case of relative addressing mode, the effective address is sum of address part (Y) plus the contents of program counter (W+2).

Thus Z= Y+ W+2

(d) In index addressing mode, the effective address is the sum of address part plus contents of index register.

Thus effective address Z= Y+X

Questions 4: - The memory unit of a computer has 256K words of 32 bits each. The computer has an instruction format with four fields: an operation code field, a mode field to specify one of seven addressing modes, a register address field to specify one of 60 processor registers, and a memory address. Specify the instruction format and the number of bits in each field if the in instruction is in one memory word.

Solution: - Memory unit = 256K words = 2^{18} words (**Memory size is 256K X 32**)

Hence 18 bits are required for memory address.

There are 60 processor register and we know that $2^5 < 60 < 2^6$

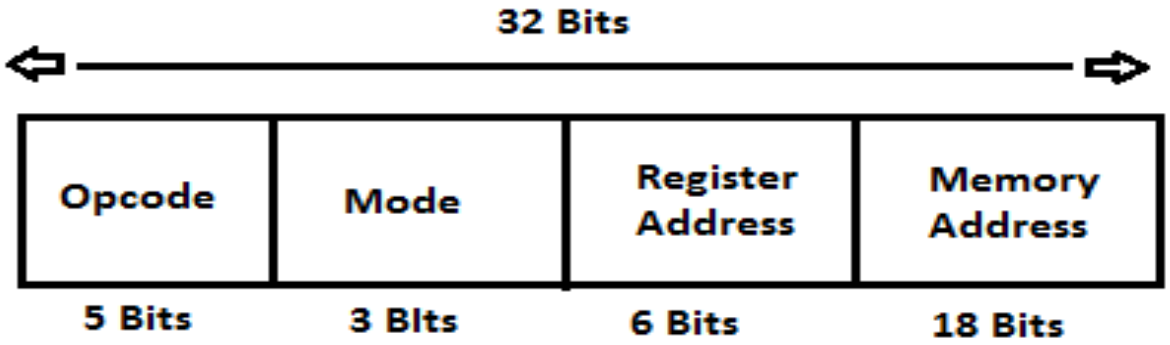
Hence 6 bits are required for register. Also there are 7 addressing mode and $2^2 < 7 < 2^3$. Thus we require 3 bits to specify 6 addressing modes.

Total number of bits in instruction format are 32.

Hence number of bits in opcode field = 32 – (18+6+3)

= 32-27 = 5 bits

Thus the instruction format is as follows.



Test yourself: - An instruction is stored at location 500 with its address field at location 501. The address field has the value 300. A processor register R1 contains the number 100. Evaluate the effective address if the addressing mode of the instruction is

(i) Direct

(ii) Relative

(iii) Register indirect

(iv) Index with R1 as lndex register

Make suitable assumptions if any

References

- Mano Morris, “Computer System Architecture”, PHI
- William Stalling, “Computer Organization & Architecture”, Pearson education Asia
- Hamacher vranesic zaky, “Computer Organization”, McGraw Hill
- B. Ram, “Computer Fundamental Architecture & Organization”, New Age.
- Tannenbaum, “Structured Computer Organization”, PHI.