LECTURES NOTES ON

DIGITALELECTRIONICS AND MICROPROCESSOR

GOVT.POLYTECHNIC,BHADRAK

PREPARED BY

TAPAN KUMAR DAS

DEPARTMENT OF ELECTRICAL ENGINEERING

Government Polytechnic, Bhadrak

BASICSOFDIGITALELECTRONICS

The branch of electronics that deals with digital data in the form of codes. There areonly two codes in digital electronics, and they are 0 and 1. 0 is considered to be lowlogic while 1 is considered to be high logic.

Digital Electronicscan also be defined as the circuit which deals with Digital Signal is knows as Digital Electronics

AdvantagesOfDigitalElectronics

a. DigitalElectroniccircuitsarerelativelyeasytodesign.
b. Ithashigherprecisionrateintermsofaccuracy.
c. Transmittedsignalsarenotlostoverlongdistance.
d. DigitalSignalscanbestoredeasily.
e. Digital Electronics is more immune to 'error' and 'noise' than analog. But in case of high-speed designs, a small noise can induce error in the signal.
f. ThevoltageatanypointinaDigitalCircuitcanbeeitherhighorlow;hence there is less chance of confusion.
g. DigitalCircuitshavetheflexibilitythatcanchangethefunctionalityofdigital circuits by making changes in software instead of changing actual circuit.

DisadvantagesofDigitalElectronics

a. The real world is analog in nature, all quantities such as light, temperature, soundetc.DigitalSystemsisrequiredtotranslateacontinuoussignaltodiscrete which leads to small quantization errors. To reduce quantization errors a large amount of data needs to be stored in Digital Circuit.
b. DigitalCircuitsoperateonlywithdigitalsignalshence,encodersanddecoders are required for the process. This increases the cost of equipment.

NumberSystem

A digital system can understand positional number system only where there are a few symbols called digits and these symbols represent different values depending on the position they occupy in the number.

Avalueofeachdigitinanumbercanbedeterminedusing

a. Thedigit
b. Thepositionofthedigit inthenumber
c. The base of the number system (where base is defined as the total number of digits available in the number system).

TypenumberSystem

1. DecimalNumberSystem
2. BinaryNumberSystem
3. OctalNumberSystem
4. HexadecimalNumberSystem

DecimalNumberSystem

Thenumbersystemthatweuseinourday-to-daylifeisthedecimalnumbersystem The decimal number system contains ten digits from 0 to 9.(0,1,2,3,4,5,6,7,8,&9) Base=10
Thepositioninthedecimalnumbersystemspecifiesthepowerofthebase(10).

Example

Mathematically,wecanwriteitas

$2541=(2\times1000)+(5\times100)+(4\times10)+(1\times1)$

$=(2\times10^3)+(5\times10^2)+(4\times10^1)+(1\times10^0)$

$=2541$

BinaryNumberSystem

Generally,abinarynumbersystemisusedinthedigitalcomputers.Inthisnumber system, it carries only two digits, either 0 or 1
Thebinarynumbersystemcontains2digitsfrom0&1 Base=10
Thepositioninthebinarynumbersystemspecifiesthepowerofthebase(2) Mathematically,

we can write it as

$1101.011=(1\times2^3)+(1\times2^2)+(0\times2^1)+(1\times2^0)+(0\times2^{-1})+(1\times2^{-2})+(1\times2^{-3})$

OctalNumberSystem

Theoctalnumbersystemcontains8digitsfrom0to7(i.e.0,1,2,3,4,5,6&7) Base=8
Thepositionintheoctalnumbersystemspecifiesthepowerofthebase(8) Mathematically,

we can write it as

$12570=(1\times8^4)+(2\times8^3)+(5\times8^2)+(7\times8^1)+(0\times8^0)$

HexadecimalNumberSystem

Uses10digitsand6letters,0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.
Lettersrepresentsnumbersstartingfrom10.A=10,B=11,C=12,D=13,E=14,F =15. Base =16

ThepositionintheHexadecimalNumberSystemnumbersystemspecifiesthepowerof the base (8)

Mathematically,wecanwriteitas

$19FDE_{16}=(1\times16^4)+(9\times16^3)+(F\times16^2)+(D\times16^1)+(E\times16^0)$

NumberSystemandBaseConversions

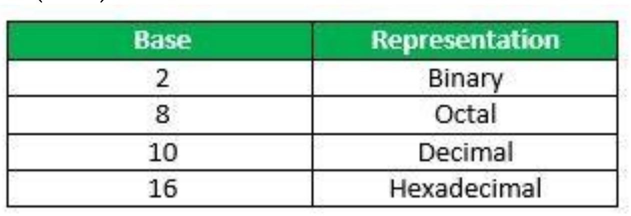
ElectronicandDigitalsystemsmayuseavarietyofdifferentnumbersystems,(e.g. Decimal, Hexadecimal, Octal, Binary).

AnumberNinbaseorradixbcanbewrittenas:

(N)b=dn-1dn-2--------d1d0.d-1d-2 ------------------- d-m

Intheabove,dn-1tod0istheintegerpart,thenfollowsaradixpoint, and then d-1 to d-m is the fractional part.

dn-1=Mostsignificantbit(MSB) d-m = Least significant bit (LSB)

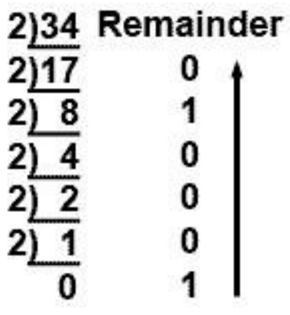


Base	Representation
2	Binary
8	Octal
10	Decimal
16	Hexadecimal

1. DecimaltoBinary

Convert$(34.25)_{10}$toBinaryequivalent

Step1: Dividethenumber34anditssuccessivequotientswithbase2.



Step 2:

Now,performthemultiplicationof0.25andsuccessivefractionwithbase2.

Operation	Result	carry
0.25×2	0.50	0
0.50×2	0	1

$(0.25)_{10}$=$(.01)_2$

FinalResultis

(□□.□□)□□=(□□□□□□.□□)□

2. BinarytoDecimal

Convert$(1010.01)_2$toequivalentDecimalNo.

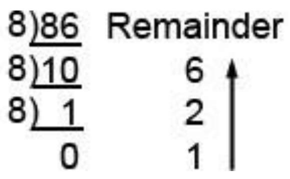
$(1010.01)_2=1x2^3+0x2^2+1x2^1+0x2^0+0x2^{-1}+1x2^{-2}=8+0+2+0+0+0.25=10.25$

$=(10.25)_{10}$

3. DecimaltoOctal

Convert$(86)_{10}$toOctalequivalent

Step1: Dividethenumber34anditssuccessivequotientswithbase8.



	Remainder
8)86	
8)10	6
8) 1	2
0	1

Step2:

Nowperformthemultiplicationof0.35andsuccessivefractionwithbase8.

Operation	Result	carry
0.35X8	2.8	2
0.8X8	6.4	6
0.4X8	3.2	3
0.3X8	2.4	2

$(0.35)_{10}$=$(2632)_8$

So,theoctalnumberofthedecimalnumber86.35is126.2632

4. OctaltoDecimal

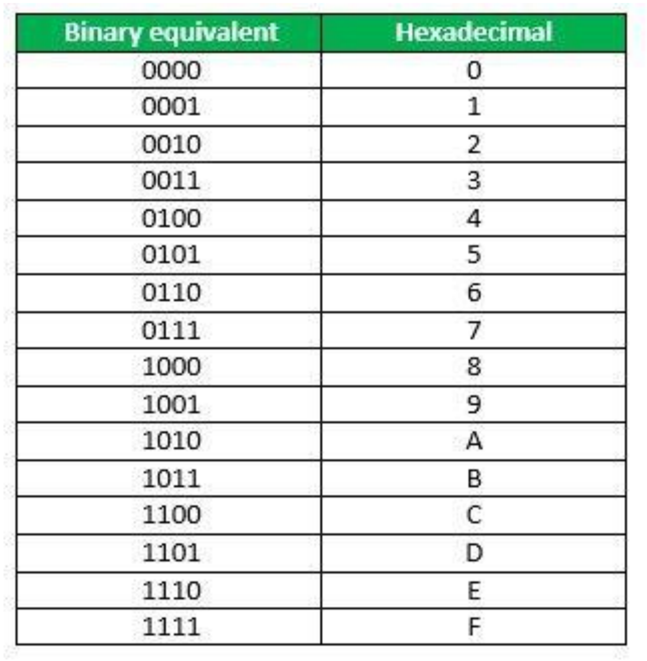
$(12.2)_8$

$1x8^1+2x8^0+2x8^{-1}=8+2+0.25=10.25$

$(12.2)_8=(10.25)_{10}$

5. HexadecimaltoBinary

To convert from Hexadecimal to Binary, write the 4-bit binary equivalent of hexadecimal.



Binary equivalent	Hexadecimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Example

$(3A)_{16}=(00111010)_2$

6. BinarytoHexadecimal

ToconvertfromBinarytoHexadecimal,startgroupingthebitsingroupsof4from the right-end and write the equivalent hexadecimal for the 4-bit binary. Add extra 0'son the left to adjust the groups.

1111011011

$\underline{001111011011}$

$(001111011011)_2=(3DB)_{16}$

7. Hexa-decimaltoDecimalConversion

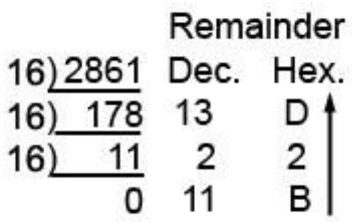
The process of converting hexadecimal to decimal is the same as binary to decimal. The process starts from multiplying the digits of hexadecimal numbers with its corresponding positional weights. And lastly, we add all those products.

Example1:(152A.25)$_{16}$

(152A.25)$_{16}=(1\times16^3)+(5\times16^2)+(2\times16^1)+(A\times16^0)+(2\times16^{-1})+(5\times16^{-2})$

$=5418.14453125$

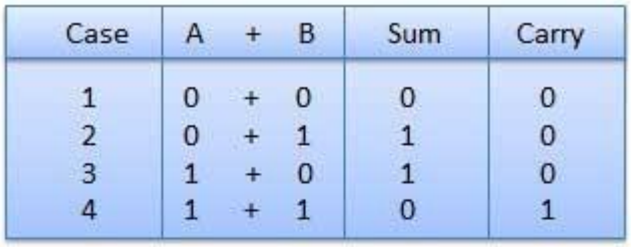
8. DecimaltoHexadecimal



□□□□$_{□□}$=□□□$_{□□}$

Binaryaddition,subtraction,MultiplicationandDivision

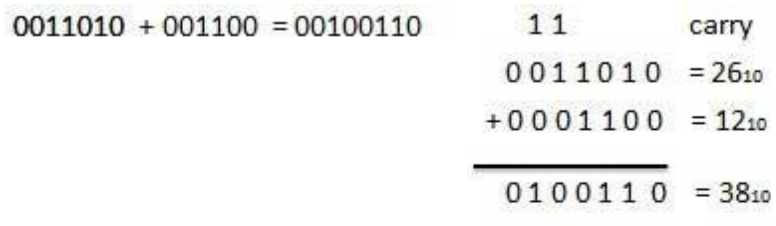
1.Binaryaddition



Case	A	+	B	Sum	Carry
1	0	+	0	0	0
2	0	+	1	1	0
3	1	+	0	1	0
4	1	+	1	0	1

Infourthcase,abinary additioniscreatingasumof(1+1=10)i.e.0is writteninthe given column and a carry of 1 over to the next column.

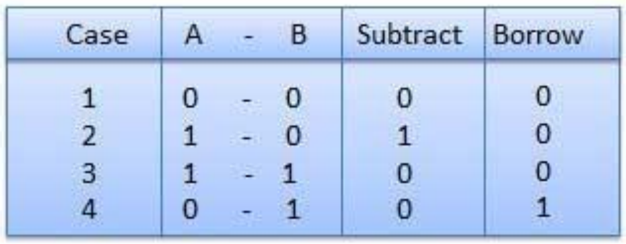
Example−Addition



```
0011010 + 001100 = 00100110        1 1          carry
                                   0 0 1 1 0 1 0   = 26₁₀
                                  +0 0 0 1 1 0 0   = 12₁₀
                                  ──────────────
                                   0 1 0 0 1 1 0   = 38₁₀
```

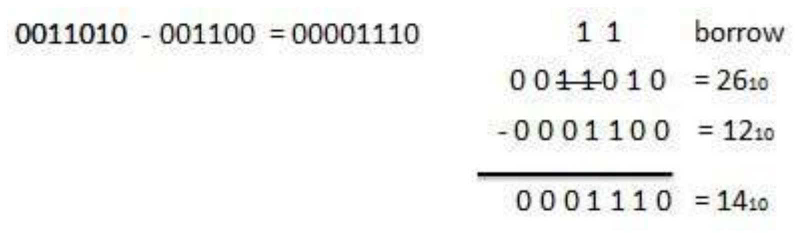
2. BinarySubtraction

Subtraction and Borrow, these two words will be used very frequently for the binary subtraction. There are four rules of binary subtraction.



Case	A	-	B	Subtract	Borrow
1	0	-	0	0	0
2	1	-	0	1	0
3	1	-	1	0	0
4	0	-	1	0	1

Example−Subtraction

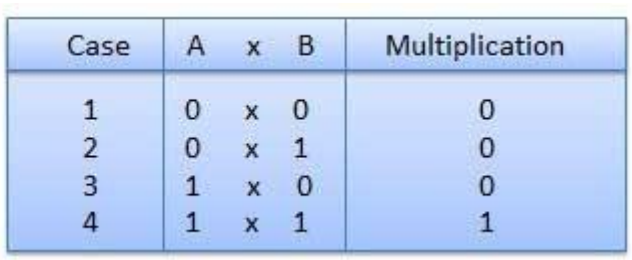


```
0011010 - 001100 = 00001110          1 1        borrow
                                   0 0 1 1 0 1 0   = 26₁₀
                                  -0 0 0 1 1 0 0   = 12₁₀
                                  ______________
                                   0 0 0 1 1 1 0   = 14₁₀
```

(In the minuend, the two 1s under the borrow marks are struck through: 0 0 ~~1 1~~ 0 1 0.)

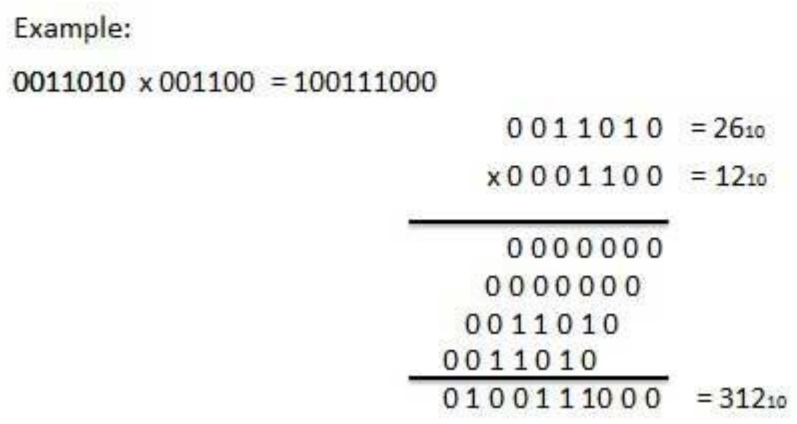
3. BinaryMultiplication

Binary multiplication is similar to decimal multiplication. It is simpler than decimal multiplication because only 0s and 1s are involved. There are four rules of binary multiplication.



Case	A x B	Multiplication
1	0 x 0	0
2	0 x 1	0
3	1 x 0	0
4	1 x 1	1

Example−Multiplication



Example:

```
0011010 x 001100 = 100111000
                                  0 0 1 1 0 1 0   = 26₁₀
                                 x0 0 0 1 1 0 0   = 12₁₀
                          _____________________
                                  0 0 0 0 0 0 0
                                0 0 0 0 0 0 0
                              0 0 1 1 0 1 0
                            0 0 1 1 0 1 0
                          _____________________
                            0 1 0 0 1 1 1 0 0 0   = 312₁₀
```

.1‘scomplementand2‘scomplementnumbersforabinary number

a.1‘s complement

1’scomplement

1’s complementof a binary number is another binary number obtained by toggling all bits in it, i.e., transforming the 0 bit to 1 and the 1 bit to 0.

Originalvalue		1’scomplement
0	$\longrightarrow$	1
1	$\longrightarrow$	0

Examples:

1'scomplementof7(0111)is8(1000)

1'scomplementof12(1100)is3(0011)

Useof1's complement

The main use of 1's complement is to represent a signed binary number. Apart from this, it is also used to perform various arithmetic operations such as addition and subtraction.

In signed binary number representation, we can represent both positive and negative numbers

.2’scomplement

2’scomplement ofabinarynumberis1addedtothe1′scomplementofthebinary number.

i.e.	Original value	1′s complement	
		2′scomplement1011	0100
		0100+1=0101	
	1101	0010	0010+1=0011

Useof2's complement

Negative binary numbers are represented in 2’s complement form so that the same logic circuit can be used to perform addition as well as subtraction

Subtractionofbinarynumbersin 2‘scomplementmethod.

Theoperationiscarriedoutbymeansofthefollowingsteps:

(i) Findthe 2’scomplementofthesubtrahend(negativeno.only,because2’s complement of positive no. is remain same) of given no..
(ii) Thenitisaddedtotheminuend.(add2’scomplementedwithpositivegivenno.)
(iii) Ifthefinalcarryoverofthesumis1,itisdroppedandtheresultispositive.
(iv) If there is no carry over, thetwo’s complement of the sum will be the result and it is negative.

Examples:

(i) 110110-10110

Solution:

Now,2’scomplementof010110is(101101+1)i.e.101010.Addingthiswiththe minuend.

110110 Minuend

101010 2’scomplementofsubtrahend

Carryover1 100000 Resultofaddition
Afterdroppingthecarryoverwegettheresultofsubtractiontobe100000.

(ii) 10110–11010

Solution:

2’scomplementof11010is(00101+1)i.e.00110.Hence
Minued - 10110

2’scomplementofsubtrahend- 0 0 1 1 0

Resultofaddition- 11100

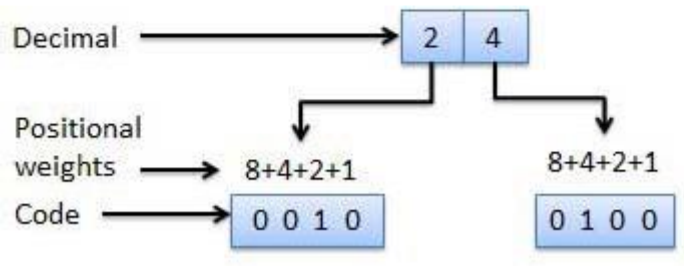
Asthereisnocarryover,theresultofsubtraction isnegativeandisobtainedbywriting the 2’s complement of 11100 i.e.(00011 + 1) or 00100.
Hencethedifferenceis–100.

Use of weighted and Un-weighted codes & write Binary equivalent number for a number in 8421, Excess-3 and Gray Code and vice-versa.

Weighted code

Weighted binary codes are those binary codes which obey the positional weight principle.Eachpositionofthenumber representsaspecificweight. Severalsystems of the codes are used to express the decimal digits 0 through 9. In these codes each decimal digit is represented by a group of four bits.

a. BCD(8421)
b. 6311
c. 2421
d. 642-3
e. 84-2-1



UseofWeightedcodes

a) Datamanipulationduringarithmeticoperation.
b) Weightedbinarycodeisessentialfordisplayingnumericvaluesindigital devices such as voltmeters and calculators
.c)Torepresentthedecimaldigitsincalculators,voltmetersetc.

Non-WeightedCodes

In this type of binary codes, the positional weights are not assigned. The examples of non-weighted codes are Excess-3 code and Gray code.

Nonweightedcodes areusedin:

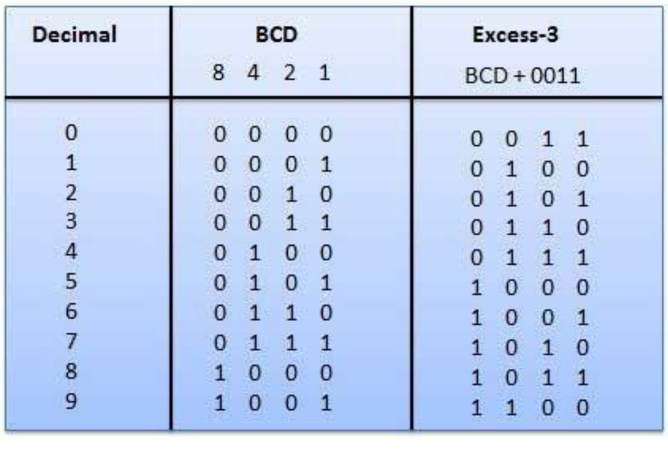
a) Toperformcertainarithmeticoperations.
b) Shiftpositionencodes.
c) Usedforerrordetectingpurpose.

Excess-3code

The Excess-3 code is also called as XS-3 code. It is non-weighted code used to express decimalnumbers.The Excess-3codewordsarederivedfromthe8421BCDcodewords adding $(0011)_2$or $(3)10$ to each code word in 8421. The excess-3 codes are obtained as follows −

$$\text{Decimal Number} \longrightarrow \text{8421 BCD} \xrightarrow[\text{0011}]{\text{Add}} \text{Excess-3}$$

Example



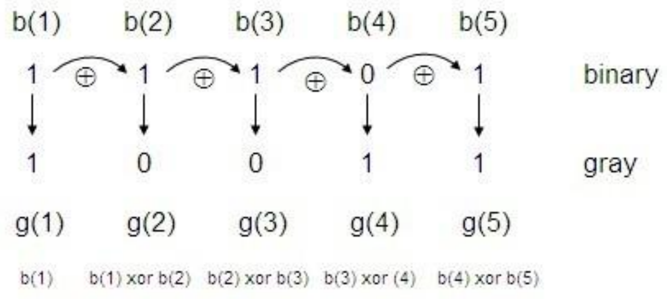
Decimal	BCD 8 4 2 1	Excess-3 BCD + 0011
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

GrayCode

It is the non-weighted code and it is not arithmetic codes. That means there are no specificweightsassignedtothebitposition.Ithasaveryspecialfeaturethat,onlyone bit will change each time the decimal number is incremented as shown in fig. As only one bit changes at a time, the gray code is called as a unit distance code. The gray code is a cyclic code. Gray code cannot be used for arithmetic operation.

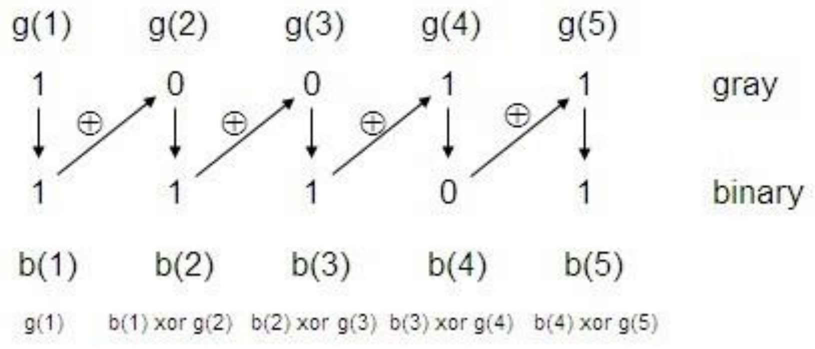
ConvertabinarynumbertoaGraynumber

Let'sunderstandthealgorithmtogofrombinarytoGray.Seetheconversionfrom'11101' binary to its equivalent in Gray code.



ConvertaGraynumber toabinarynumber

Let'sunderstandthealgorithmtogofrombinarytoGray.Seetheconversionfrom '11101' binary to its equivalent in Gray code.



ApplicationofGraycode

- Graycodeispopularlyusedintheshaftpositionencoders.
- Ashaftpositionencoderproducesacodewordwhichrepresentstheangular position of the shaft.

Importanceof parity Bit.

A parity bit is an extra bit included in binary message to make total number of 1's either odd or even. Parity word denotes number of 1's in a binary string. There are two parity system-even and odd.

Evenparitysystem

In even parity system 1 is appended to binary string it tḷ ereisanodd number of 1's in sꞁ ring otherwise 0 is appended to make tot levennumber of 1's.

Oddparitysystem

Inoddparitysystem,1isappendedtobinarystringifthereisevenanumber of 1's to make an odd number of 1's

ImportanceofparityBit.

Thepurposeofaparitybitistoprovideasimplewaytochec Errors ꝁ for

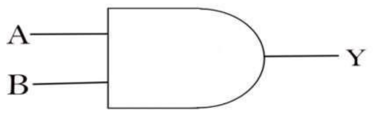
Logic Gates: AND, OR, NOT, NAND, NOR andwith truth table. **EX-ORgates**

WhatisLogicGates?

Logicgatesaretebaꞩicbuildingblocksofanydigitalsystem.Itisan electroniccircuithavingoneormorethanoneinputandolyoneoutput. Therelationship ꞓetweentheinputandtheoutputisbas dona certain logic. Based on this, logic gates are named as AND gate, OR gate, NOT gate etc.

ANDGate

An AND gate is a logic gate having two or more inputs and ꞓ singleoutput. An AND gate operates on logical multiplication rules

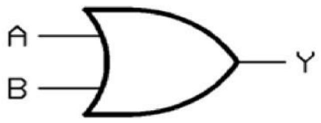


Expression for AND gate Y=A.B

Truth Table of AND gate

Input		Output
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

ORGate



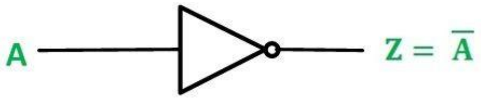
ExpressionforORgateY=A+B

TruthTable

Input		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

NOTGate

TheNOTgateisthemostbasiclogicgateofallotherlogicgates.NOTgateisalso known as an **inverter**
NOTgateonlyhasoneinputandoneoutput it
converts 0 into 1 or 1 into 0.



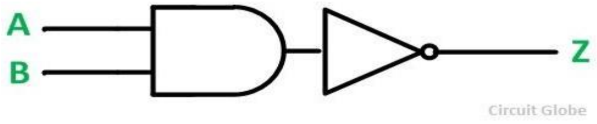
ExpressionforNOTgateZ=$\overline{A}$

TruthTable

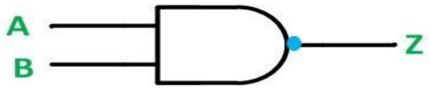
Inputs	Outputs
A	Y
0	1
1	0

NANDGate

TheNANDgateisaspecialtypeoflogicgateinthedigitallogiccircuit. The
NAND gate is the combination of AND -NOT gate
TheNANDgate istheuniversalgate.ItmeansallthebasicgatessuchasAND,OR,and NOT gate can be constructed using a NAND gate. The output state of the NAND gate will be low only when all the inputs are high. Simply, this gate returns the complement result of the AND gate.



=



ExpressionforNANDgate$Z=\overline{\square.\square}$

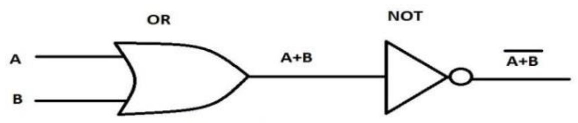
TruthTable

Input		Out Put
A	B	$Z=\overline{\square.\square}$
0	0	0
0	1	1
1	0	1
1	1	0

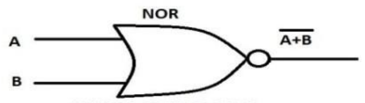
NORGate

TheNORgateisalsoauniversalgate
TheNORgateisthecombinationoftheOR-NOTgate
TheNORgateistheuniversalgate.ItmeansallthebasicgatessuchasAND,OR, and NOT gate can be constructed using a NOR gate.

TheoutputstateoftheNORgatewillbehighonlywhenalloftheinputsare low. Simply, this gate returns the complement result of the OR gate



LOGIC CIRCUIT OF NOR GATE



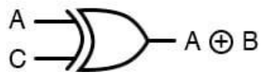
SYMBOL OF NOR GATE

ExpressionforNorgate$Z=\overline{A+B}$

TruthTable

Input		Out Put
A	B	**$Z=\overline{A+B}$**
0	0	1
0	1	0
1	0	0
1	1	0

EX-OR



ExpressionforEX-ORgate$Z=(\overline{A}B+A\overline{B})$

TruthTable

Input		Out Put
A	B	**Z=A⊕**B
0	0	1
0	1	0
1	0	0
1	1	0

RealizeAND,OR,NOToperationsusingNAND,NORgates.

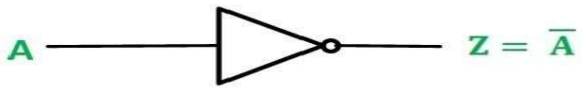
NANDandNOR gatesprovidethefollowingmeritsinthedigitallogic system design

1. Fabrication of NAND and NOR gates are easier than basic gates using in the integrated digital logic families
2. NumberoftransistorsusedtodesignNANDandNORgatesarealso less thanANDand ORgates.Sincethecoreareareducesintheintegrated digital circuits.
3. TheconversionofNANDandNORaremoreconvenietindigitaldesign.
4. AllotherlogicgatescanberealizedcompletelyusingNANDorNOR gates.
5. Anydigitalckt.canbeimplementedperfectlyusingeitherNANDor NOR gates thus these are called as universal gate

- **ImplementationofLogicgatesusingNANDGate**

i) NOTgate

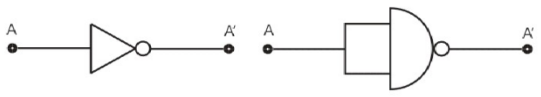
ThelogicsymbolandBooleanexpressionofNOTgateisrepresentedby



NANDequivalentrepresentationforNOTgateis

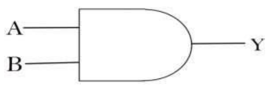
F=$\overline{\text{}=\text{}.\text{}}$

Theaboveexpression indicatesthatiftheinputterminalsofNANDgateare Same shown in fig



ii) ANDGate

ThelogicsymbolandBooleanexpressionofANDgateis representedby

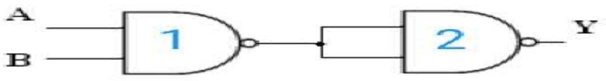


Y=A.B

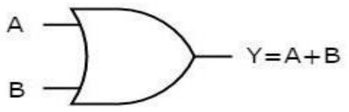
NANDequivalentrepresentationforANDgateis

$$Y=A.B=\overline{\overline{A.B}}$$

Nowaboveexpressioncandrawnas

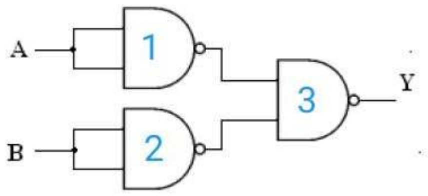


iii) OR gate:

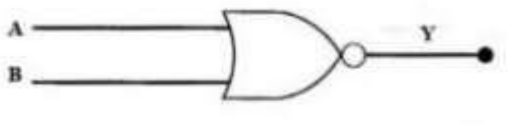


NANDequivalentrepresentationforORgateis

$$Y=A+B=\overline{\overline{A+B}}=\overline{\bar{A}.\bar{B}}$$



iv) NORgate:



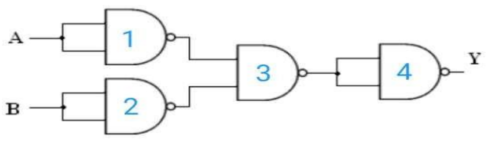
$$Y=\overline{A+B}$$

NANDequivalentrepresentationforNORgateis

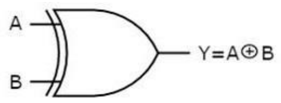
$$Y=\overline{A+B}=\overline{\bar{\bar{A}}+\bar{\bar{B}}}$$

$$=\overline{\overline{\bar{A}.\bar{B}}}$$

Nowaboveexpressioncandrawnas



v) Ex-OR gate:



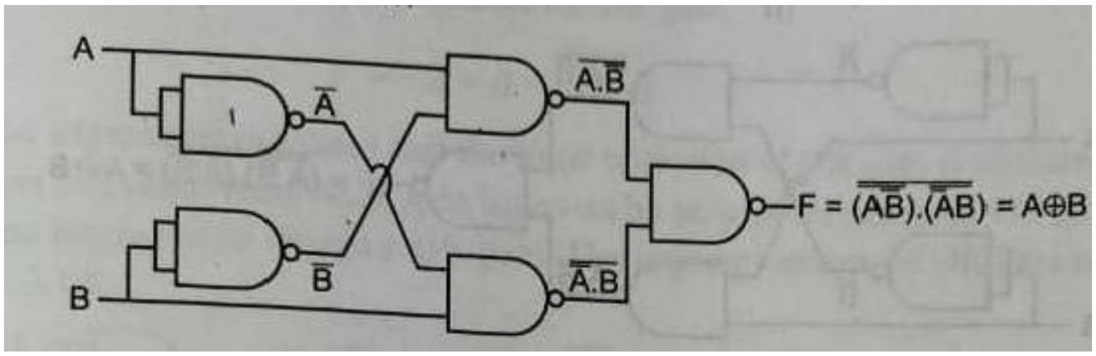
$$Y=A\bar{B}+\bar{A}B$$

NANDequivalentrepresentationforEx-ORgateis

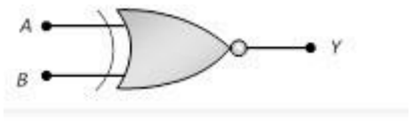
$$Y=A\bar{B}+\bar{A}B=\overline{\overline{A\bar{B}+\bar{A}B}}$$

$$=\overline{\overline{(A\bar{B})}.\overline{(\bar{A}B)}}$$

Nowaboveexpressioncandrawnas


A
B
$\bar{A}$
$\bar{B}$
$\overline{A.\bar{B}}$
$\overline{\bar{A}.B}$
$F=\overline{\overline{(A\bar{B})}.\overline{(\bar{A}B)}}=A\oplus B$

vi) Ex-NORgate:



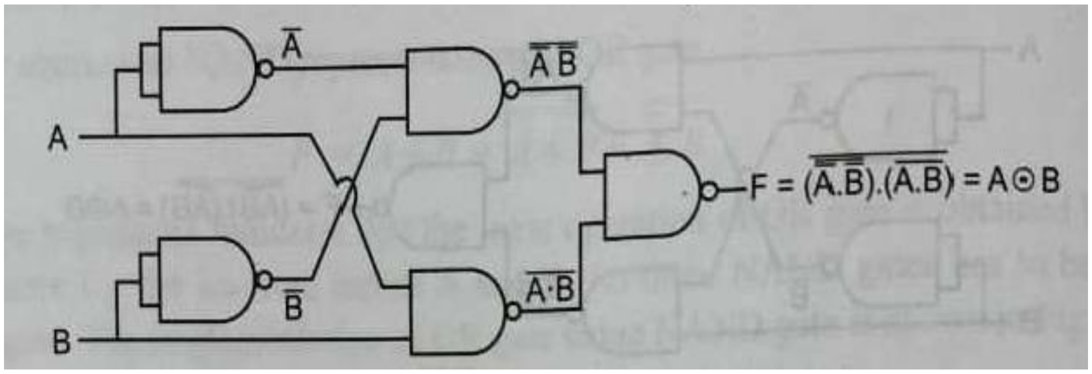
$$\mathbf{Y=A\odot B=\bar{A}\bar{B}+AB}$$

NANDequivalentrepresentationforEx-NORgateis

$$\mathbf{Y=\bar{A}\bar{B}+AB=\overline{\overline{\bar{A}\bar{B}+AB}}}$$

$$=\overline{\overline{(\bar{A}\bar{B})}.\overline{(AB)}}$$

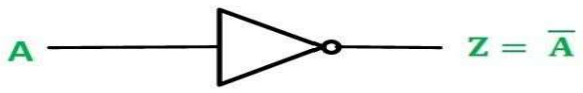
Nowaboveexpressioncandrawnas


A
B
$\overline{A}$
$\overline{B}$
$\overline{\overline{A}\,\overline{B}}$
$\overline{A \cdot B}$
$F = \overline{(\overline{\overline{A}.\overline{B}}).(\overline{A.B})} = A \odot B$

- **ImplementationofLogicgatesusingNOR Gate**

i) NOTgate

ThelogicsymbolandBooleanexpressionofNOTgateisrepresentedby

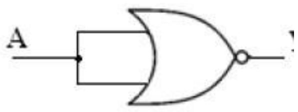


NORequivalentrepresentationforNOTgateis

$Z=\overline{\square}$

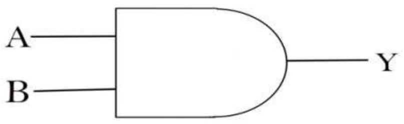
$=\overline{\square+\square}$

Nowaboveexpressioncandrawnas



ii) AND Gate

ThelogicsymbolandBooleanexpressionofANDgateisrepresented by



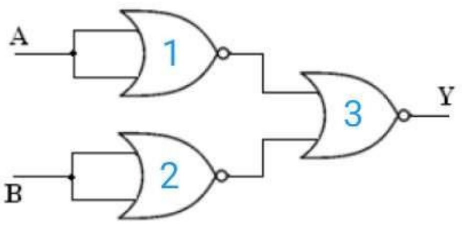
Y=A.B

NANDequivalentrepresentationforANDgateis

$Y=A.B=\overline{\overline{\square.\square}}$

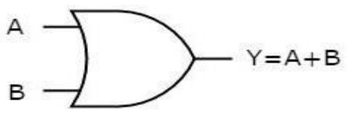
$=\overline{\overline{\square}.+\overline{\square}}$

Nowaboveexpressioncandrawnas



iii) ORgate:

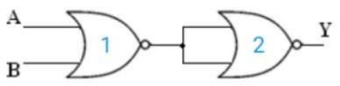
ThelogicsymbolandBooleanexpressionofORgateisrepresentedby



NORequivalentrepresentationforORgateis

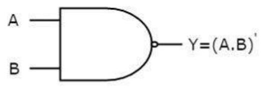
$$Y=A+B=\overline{\overline{A+B}}$$

Nowaboveexpressioncandrawnas



iv) NANDgate:

ThelogicsymbolandBooleanexpression ofNANDgateisrepresentedby



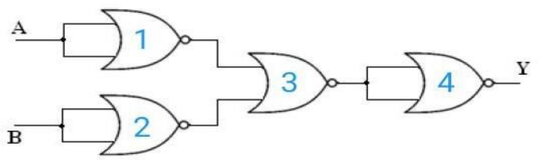
$$Y=\overline{A.B}$$

NORequivalentrepresentationforNANDgateis

$$Y=\overline{A.B}=\overline{A}+\overline{B}=$$

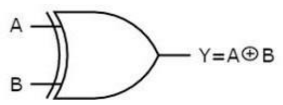
$$\overline{\overline{\overline{A}.+.\overline{B}}}$$

Nowaboveexpressioncandrawnas



v) Ex-ORgate:

ThelogicsymbolandBooleanexpressionof**Ex-OR**gateisrepresentedby



$Y=A\overline{\square}+\overline{\square}\square$NOR

equivalentrepresentationfor**Ex-OR**gateis

$$Y=\overline{\overline{A\overline{\square}+\overline{\square}\square}}$$

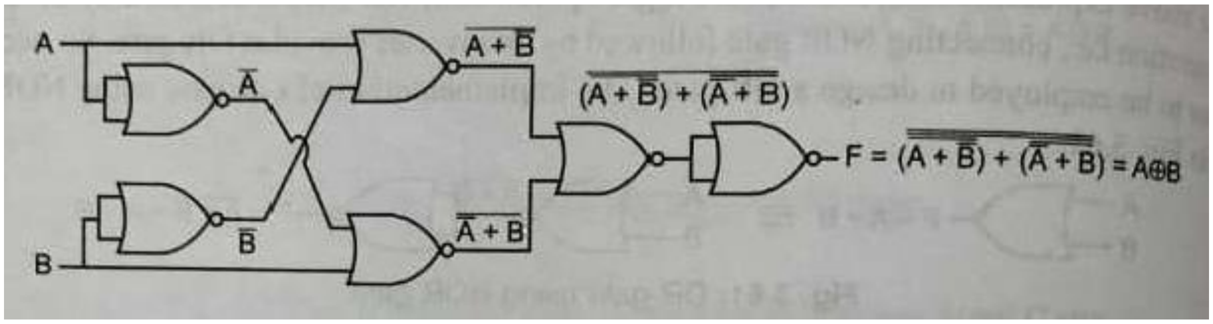
$$=\overline{\overline{(A\overline{\square})}}.\overline{\overline{(\overline{\square}\square)}}$$

$$=\overline{(\overline{\square}+\square).(A+\overline{\square})}$$

$$=\overline{(\overline{\square}+\square)}+\overline{(A+\overline{\square})}$$

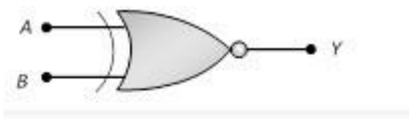
$$=\overline{\overline{\overline{(\overline{\square}+\square)}+\overline{(A+\overline{\square})}}}$$

Nowaboveexpressioncandrawnas


A
B
Ā
B̄
$\overline{A+\overline{B}}$
$\overline{\overline{A}+B}$
$\overline{(A+\overline{B})+(\overline{A}+B)}$
$F=\overline{\overline{(A+\overline{B})+(\overline{A}+B)}}=A\oplus B$

i) Ex-NORgate:

ThelogicsymbolandBooleanexpressionof **Ex-NOR**gateisrepresentedby



$$\mathbf{Y=A\odot B=\overline{\square}\,\overline{\square}+AB}$$

NORequivalentrepresentationfor**Ex-NOR**gateis

$$\mathbf{Y=A\odot B=}\overline{\overline{\overline{\square}\,\overline{\square}+\square\square}}$$

$$=\overline{\overline{\square}\,\overline{\square}}+\overline{\square\square}$$

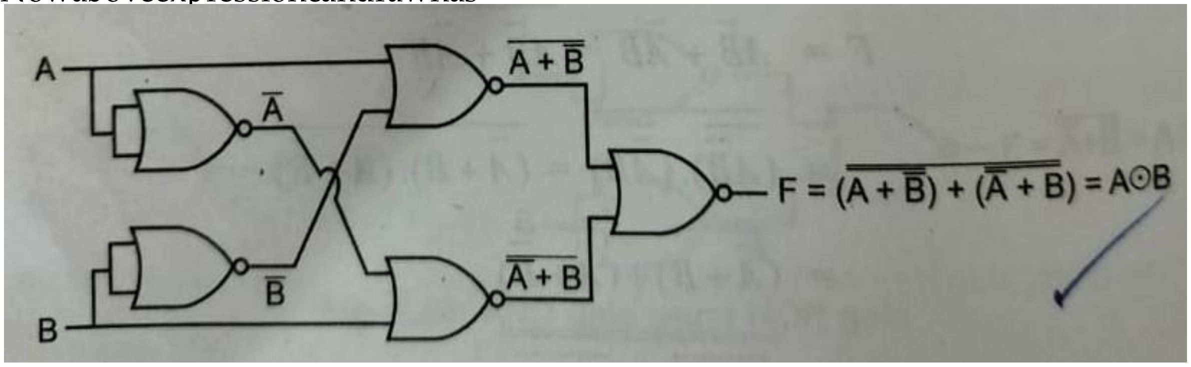
$$=(\overline{\square}\square).(\square\overline{\square})$$

$$=(\overline{\square}+\square).(\square+\overline{\square})$$

$$=\overline{\overline{(\square+\overline{\square}).(\overline{\square}+\square)}}$$

$$=\overline{\overline{(\square+\overline{\square})}+\overline{(\overline{\square}+\square)}}$$

Nowaboveexpressioncandrawnas


A
B
$\overline{A}$
$\overline{B}$
$\overline{A+\overline{B}}$
$\overline{\overline{A}+B}$
$F=\overline{(\overline{A+\overline{B}})+(\overline{\overline{A}+B})}=A\odot B$

<u>PROCEDURETOIMPLEMENTTHEBOOLEANFUNCTIONUSINGUNIVERSALGATE:</u>

1. DrawalogicdiagramfortheBooleanfunctionusingbasicgatesi.e.AND,OR,and NOT
2. ReplacethegatewithequivalentNANDorNORrealization.
3. Ifanypathhascontinuoustwoinversions,discardthosetermstoreducethe number of logic gates employed to implement the Boolean function.
4. RedrawthesimplifiedlogicdiagramastheUniversalgatesimplementationof Boolean function.

Example:ImplementthefollowingBooleanfunctionusingminimumnumberof(i)NAND gates, (ii)NOR gates

$$F=\overline{\square\square+\square\square}$$

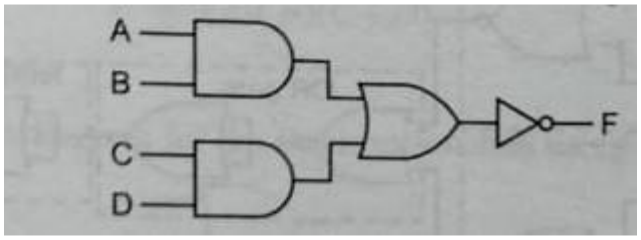
Solution:

GivenBooleanfunction,

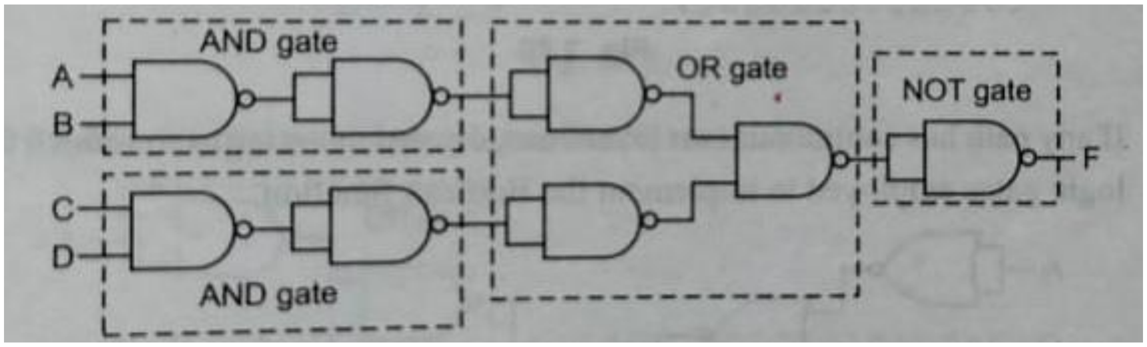
$$F=\overline{AB+CD}$$

(i) UsingNANDgates:

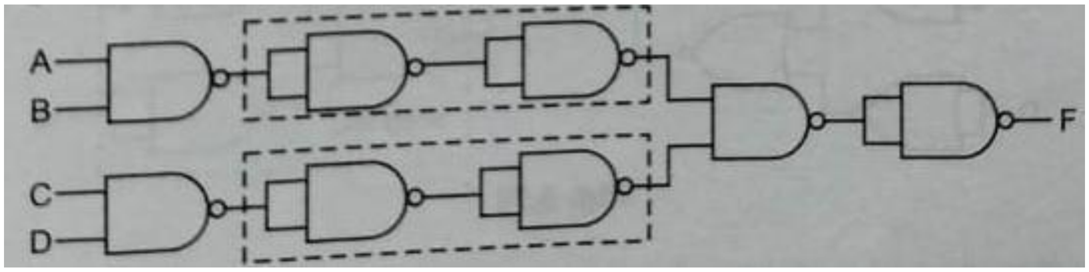
1. Step:1DrawalogicdiagramfortheBooleanfunctionusingbasicgatesi.e. AND,OR,and NOT



2. Step:2ReplacethegatewithequivalentNANDrealization.

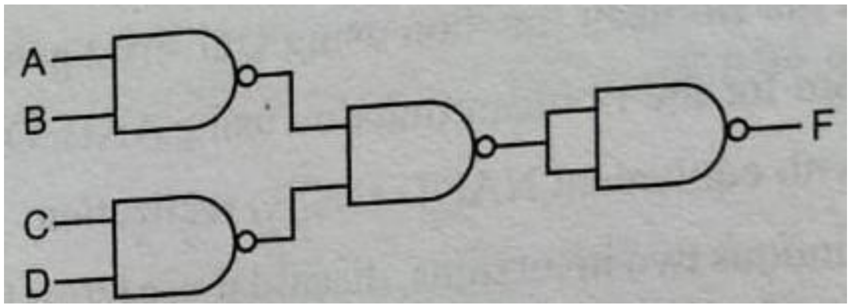


3. Step:3Ifanypathhascontinuoustwo inversions,discardthosetermstoreduce the number of logic gates employed to implement the Boolean function.



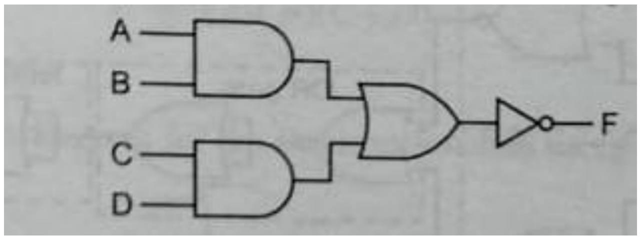
Inthiscase,bothpatharehavingtwoinversionsinseriessodiscardthoseinverter

4. Step:4RedrawthesimplifiedlogicdiagramastheUniversalgates implementation of Boolean function

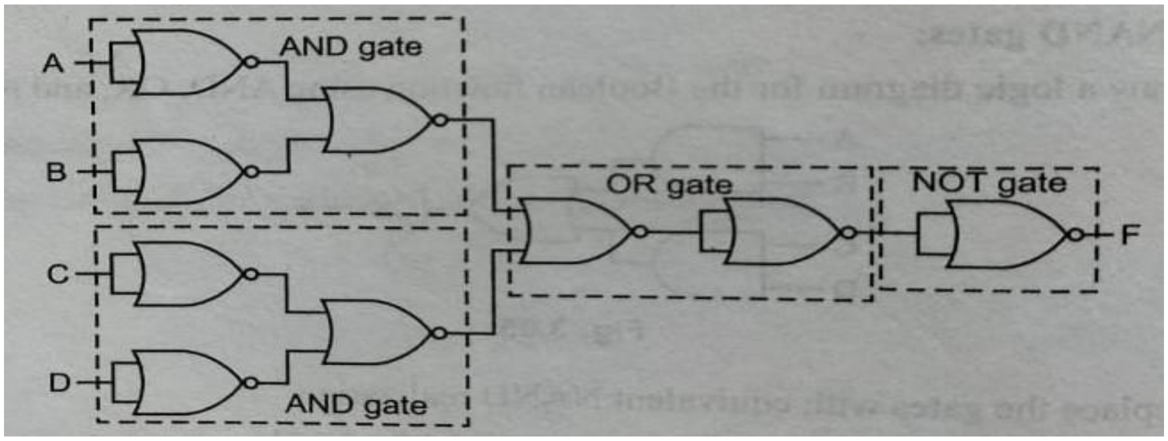


(ii) UsingNORgates:

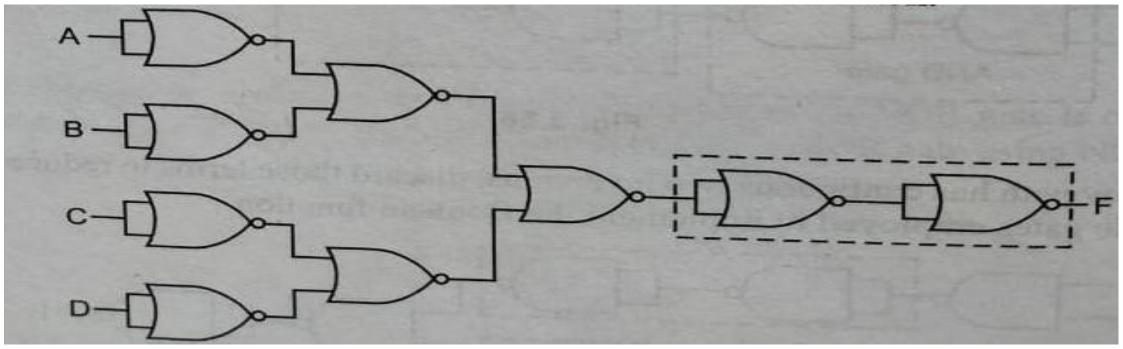
1. Step:1DrawalogicdiagramfortheBooleanfunctionusingbasicgatesi.e. AND,OR,and NOT



2. Step:2ReplacethegatewithequivalentNORrealization.

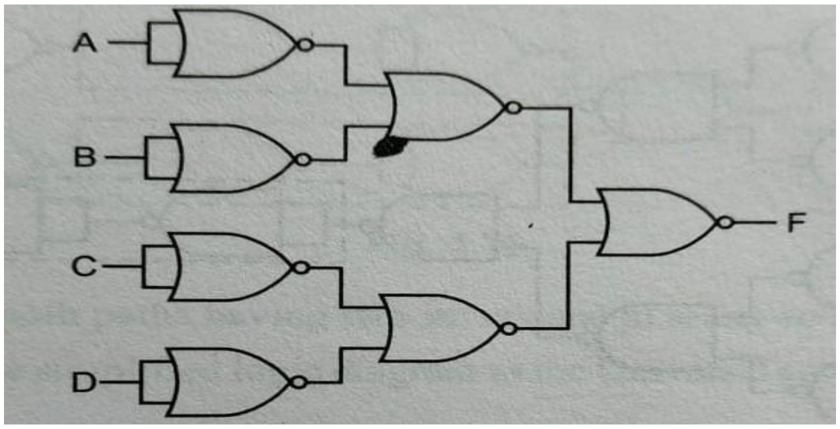


3. Step:3Ifanypathhascontinuoustwo inversions,discardthosetermstoreduce the number of logic gates employed to implement the Boolean function.



Inthiscase,outputsectionehavingtwoinversionsinseriessodiscardthose inverter

4. Step:4RedrawthesimplifiedlogicdiagramastheUniversalgates implementation of Boolean function



DifferentpostulatesandDe-Morgan‘stheoremsinBoolean algebra.

a. $\overline{AB}=\overline{A}+\overline{B}$

b. $\overline{(A+B)}=\overline{A}.\overline{B}$

UseOfBooleanAlgebraForSimplificationOfLogicExpression

WhatisBoolean Algebra?

Boolean Algebra is used to analyze and simplify the digital (logic) circuits. It uses only the binary numbers i.e. 0 and 1. It is also called as **Binary Algebra**or**logical Algebra**. Boolean algebra was invented by **George Boole** in 1854

RuleinBoolean Algebra

FollowingaretheimportantrulesusedinBooleanalgebra.

I. Variable used can have only two values. Binary 1 for HIGH and Binary 0 for LOW.

II. Complement of a variable is represented by an overbar (-). Thus, complement of variable B is represented as $\overline{B}$. Thus if B = 0 then $\overline{B}$ = 1 and B = 1 then $\overline{B}$ = 0.

III. ORingofthevariablesisrepresentedbyaplus(+)signbetweenthem.For example ORing of A, B, C is represented as A + B + C.

IV. LogicalANDingofthetwoormorevariableisrepresentedbywritingadot between them such as A.B.C. Sometime the dot may be omitted like ABC.

BASICLAWSOFBOOLEANALGEBRA:

1. **NOTLaw:**
   i. $\overline{0}=1$
   ii. $\overline{1}=0$
   iii. $A=\overline{\overline{A}}$
2. **ANDLaws**
   i. A.0=0
   ii. A.1=A
   iii. A.A= A
   iv. $A.\overline{A}=0$
3. **ORLaws:**
   i. A+0=A
   ii. A+1=1
   iii. A+A= A
   iv. $A+\overline{A}=1$
4. **CommutativeLaws:**
   i. A+A=B+A
   ii. A.B=B.A
   iii. A+B+C=C+B+C

iv. $A.B.C=B.C.A=C.A.B$

5. **Associativelaws:**
   i. $A+(B+C)=(A+B)+C$
   ii. $A.(B.C)=(A.B).C$
6. **Distributivelaw:**
   i. $A+BC=(A+B)(A+C)$
   ii. $A(B+C)=AB+AC$

BOOLEANTHEOREM

1. **$A+AB=A$**

Proof: $A+AB=A(B+\square)+A\overline{B}$

$=AB+A\overline{\square}+AB$

$=AB+\square\overline{\square}$

$=A(B+\overline{\square})$

$=A.1$

$=A$

2. **$A(A+B)=A$**

Proof: $A(A+B)=A.A+A.B$

$=A+AB$

$=A$

3. **$A+\overline{\square B}=A+B$**

Proof: $A+\overline{\square}B=A+AB+\square B\ \overline{\ }$

$=A+(A+\square)\overline{B}$

$=A+B$

4. **$A.(\overline{\square}+B)=A$**

Proof: $A.(\overline{\square}+B)=A.\square+A\overline{B}$

$=0+AB$

$=AB$

5. **$AB+A.\square\overline{=A}$**
6. **$(A+B).(A+\square)=A\overline{\ }$**
7. **$(A+B).(A+C)=A+BC$**
8. **$AC+\square\overline{B}C=AC+BC$**

DEMORGAN’S THEOREM:

I. $\overline{□.□}=\overline{□}+\overline{□}$

II. $\overline{□+□}=\overline{□}.\overline{□}$

DUALITYTHEOREM:

Dualitytheoremsaythatinthelogicfunctionapplyingthefollowingchangesin the AND, OR and NOT operation doesn’t affect the output.

1. Swap‘0’and‘1’presentintheexpression.
2. ReplacingANDoperationbyORoperation
3. ReplacingORoperationbyANDoperation

Examples:

a. A+0=A.1=A
b. A(B+C)=AB+AC

Afterapplyingdualitytheoremintheaboveexpression,itbecomes A + (

B C) =(A + B) . (A + C)

ABSORPTIVETHEOREM:

a. **A.(□+$\overline{\text{B}}$)=A.B**
b. **A+(□.$\overline{\text{B}}$)=A+B**

TRANSPOSITION THEOREM:

a. **A.B+□.$\overline{\text{C}}$ =(A+C)(□+B)**
b. **(A+B). (□+$\overline{\text{C}}$)=A.C +□.B**

USING THE THEOREM & LAWS, SIMPLIFY THE FOLLOWING EXPRESSION

2. **(**A+B)(A+C)

=A.A+A.C+A.B+B.C -Distributivelaw

=A+A.C +A.B+B.C -IdempotentANDlaw(A.A=A)

=A(1+C)+A.B+B.C -Distributivelaw

=A+AB+BC -IdentityORlaw(1+C=1)

=A(1+B)+BC

=A.1+BC

=A+BC

3. □□□□+□□□□+□□□□+□□□□+□□+□

=□□□□+□□□□+□□□□+□□□□+□□+□

=□(□□□+□□□+□□□+□□□+1)+□□

=□+□□

=(□ +□)(□+□)

=A+D

4. □□□+□□□ +□□□+□□□

=□□□+□□□+(□+□)□□

=□□□+□□□+□□

=□□□+□(□□+□)

=□□□+□{(□+□)(□+□)}

=□□□ +□(□+□)

=□□□+□□+□□

=□(□□+□)+ □□

=□{(□+□)(□+□)}+□□

=□(□+□)+□□

=□□+□□+□□

Karnaugh MapFor2,3,4 Variable,SimplificationOfSOP And POS Logic Expression Using K-Map.

A.BOOLEAN FUNCTION:

Boolean function consists of a set of Boolean variables to represent a number using Boolean connectivity's logical NOT, logical AND, logical OR operations, parenthesis and equality sign. Itis also known as Boolean expression.

Based on the arrangement of literals and terms Boolean expression is classified in two types such as,

1. SumofProduct(SOP)form
2. ProductofSum(POS)form

1. SumofProduct(SOP)form:

Sum of Product term is consisting of sum (OR operation) of many terms; the terms may consists of single literal or product of many literals (Variables).The sum of the terms is called SOP function.

Example:

i. F(A,B,C)=⍰⍰⍰+⍰⍰+⍰⍰+⍰⍰⍰

ii. F(x,y,z)=⍰⍰+⍰⍰+⍰⍰⍰

iii. F(A,B,C,D)=⍰⍰⍰⍰+⍰⍰⍰⍰+⍰⍰⍰⍰+⍰⍰⍰⍰+⍰⍰⍰⍰

a.StandardSumofProduct(SOP)form:

The SOP form of expression is said to be Standard Sum of Product formor Canonical form expression if the terms present in the expression contains all the literals present in the function.

Each individual term present in the expression must have all the literalsof a function.

ThestepstoconvertnoncanonicalSOPtoCanonicalorstandard SOP.

1. Findthemissingliteralineachproductterm.
2. Multiply(AND)eachproducttermtothetermhavingmissingliteralbyORingthe missing literal and its complement.
3. Expandthetermsandrearrangetheliteralsintheproduct terms.
4. Reducetheexpressionbyomittingtherepeatedtermsifany(i.e.A+A=A)

Example:

i) ConvertthegivenexpressionF(A,B,C)= A+⍰CintocanonicalSOP form.

Inthegivenexpression,literalBandC aremissinginthe1stproductterm. So (B+⍰) and (C+⍰) are multiplied (AND) with the term A. Similarly, literal A is missing in the 2ndproduct term. So (A+⍰) is multiplied (AND) with the product term ⍰C.

Given;

F(A,B,C)=⍰+⍰⍰

=⍰(⍰+ ⍰)+ ⍰⍰(⍰ +⍰)

=⍰⍰+⍰⍰+⍰⍰⍰+⍰⍰⍰

=⍰⍰(⍰+⍰)+⍰⍰(⍰+⍰)+⍰⍰⍰+⍰⍰⍰

=⍰⍰⍰+⍰⍰⍰+⍰⍰⍰+⍰⍰⍰+⍰⍰⍰+⍰⍰⍰

$=\square\square\square+\square\overline{\square}\square+\square\overline{\square}\square+\square\overline{\square}\,\overline{\square}\square+\square\overline{\square}\,\overline{\square}$

2. ProductofSum(POS)form:

Product of Sum (POS) term is consisting of sum (AND operation) of many terms;the terms may consists of single literal or product of many literals (Variables).The product of the set of sum terms is called POS function.

Example:

i. $F(A,B,C)=(\square+\square+\square)(\square+\square)(\square+\square)(\square+\square+\overline{\square})$

ii. $F(x,y,z)=(\square+\square)(\overline{\square}+\square)(\square+\overline{\square}+\square)$

iii. $F(A,B,C,D)=(\overline{\square}+\square+\overline{\square}+\square)(\square+\overline{\square}+\square+\square)(\square+\square+\square+\overline{\square})(\square+\square+\overline{\square}+\overline{\square})(\square+\square+\square+\square)$

a)StandardProductofsum(POS)form:

The POS form of expression is said to be **Product of sum** form or Canonical form expression if the terms present in the expression contains all the literalspresent in the function.

Each individual term present in the expression must have all the literals of a function.

ThestepstoconvertnoncanonicalPOStoCanonicalorstandard POS.

1. Findthemissingliteralineachsumterm.
2. OReachsumtermtothetermhavingmissingliteralbyANDing(product)the missing literal and its complement.
3. Expandthetermsandrearrangetheliteralsinthesumterms.
4. Reducetheexpressionbyomittingtherepeatedtermsifany(i.e.A.A=A) Let us

see an example here.

ConvertthegivenexpressionF(A,B,C)=(A+B)(B+C)intocanonicalPOSform.

In the given expression, literal C is missing in the 1st sum term. So (C.$\overline{\text{C}}$) is added with theterm(A+B).Similarly,literalAismissinginthe2nd sumterm.So(A.A)isaddedwith the term (B+C).

Given;

$F(A,B,C)=(\square+\square)(\square+\square)$

$=(\square+\square)+(\square.\overline{\square})(\square+\square)+(\square.\overline{\square})$

$=(\square+\square+\square)(\square+\square+\overline{\square})(\overline{\square}+\square+\square)(\square+\square+\overline{\square})$

$=(\square+\square+\square)(\square+\square+\overline{\square})(\square+\square+\square)$

3. SIMPLIFICATIONOFBOOLEANFUNCTION:

Thereare3-differentbasicsimplificationmethodsavailableforminimizingBoolean function

1. Booleanalgebra
2. Karnaughmap
3. QuineMcCluskeymethod

a. KARNAUGHMAP(K-MAP):

Simplifying theBoolean functions using Boolean postulates andtheorems.It isa time consuming process andto re-write the simplified expressions after each step.

To overcome this difficulty,**Karnaugh**introduced a method for simplification of Boolean functions in an easy way.

This method is a graphical method for simplification of Boolean function which consists of 2^n cells for 'n' variables. Each cell of K-map represents one of the **minterm**. The adjacent cells are differed only in single bit position.

ClassificationofK–Map:

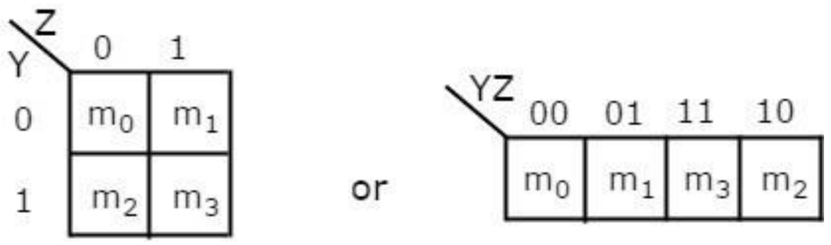
DependsonthenumberofvariablesusedintheK-mapitisclassifiedas

i. **2-Variablek-map**
ii. **3-Variablek-map**
iii. **4-Variablek-map**
iv. **5-Variablek-map**

1. 2-Variablek-map:

Thenumberofvariable(n)=2 The

number of cells $=2^n=2^2=4$



- Thepossiblecombinationsofgrouping2adjacentmintermsare{(m_0,m_1),(m_2, m_3), (m_0, m_2) and (m_1, m_3)}.

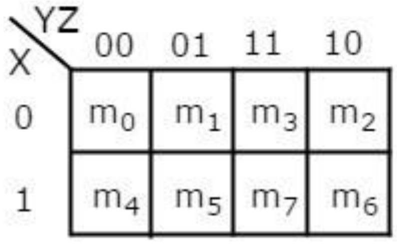
Variable		Minterms	
A	B	Representation	m_i
0	0	$\overline{A}\,\overline{B}$	m_1
0	1	$\overline{A}B$	m_2
1	0	$A\overline{B}$	m_3
1	1	AB	m_4

(Mintermsof2-variableexpression)

2. 3-Variablek-map:

Thenumberofvariable(n)=3 The

number of cells =2^n =2^3=8

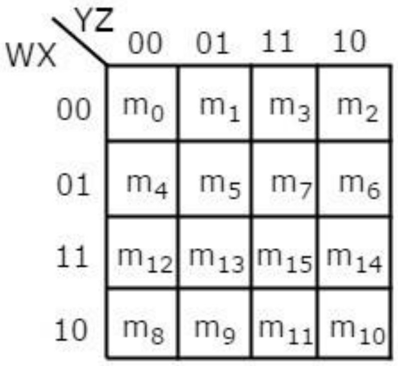


X \ YZ	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

3. 4-Variablek-map:

Thenumberofvariable(n)=4

Thenumberofcells=2^n=2^4=16



WX \ YZ	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

Don’tcarecondition:

In some digital systems, nonessential minterms or maxterms may be introduced intheinputsequences.Suchnonessentialmintermsormaxtermsarecalledasdon’tcare condition in the Boolean expression.

Thesenonessentialtermsneveroccurintheinputsequenceofthesystem.

Normally, in K-Map don’t care conditions are represented by symbol ‘X’. Don’t care values can be taken as either ‘0’ or ‘1’.

Don’tcareconditionsoccurinthedigitalsystemsunderthefollowingcondition:

i. If certain combinations of input variables are never occur, then the output functions of such combinations are considered as nonessential or don’t care condition.
ii. If certain combinations of variables are irrelevant even all the input combination of variables occurs, then the output functions of such combinations are considered as nonessential or don’t care condition.

GroupingcellforMinimization:

InK-map,mintermsaremarkedby’1’

maxterm are marked by’0’

don’tcarearemarkedby‘d’or‘x’i.eX=‘0’or ‘1’

In minterm function, don’t care condition is considered as ‘1’ if necessary for simplification or grouping cell. Else, it is marked by ‘0’

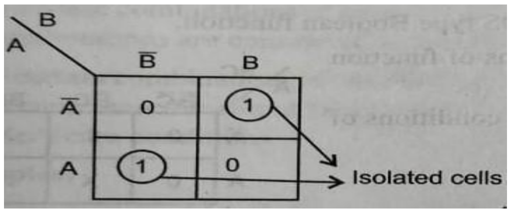
In maxterm function, don’t care condition is considered as ‘0’ if necessary for simplification or grouping cell. Else, it is marked by ‘1’

Grouping of cell or Loop of cell is process of combining adjacent cells for simplification.

Groupingisobtainedbycombining1’sor0’sof2^inumbercells,wherei=0,1,2…,n (n $\longrightarrow$number of variables used in the Boolean function.)

IsolationCellor Singlecellgroup(i=0):

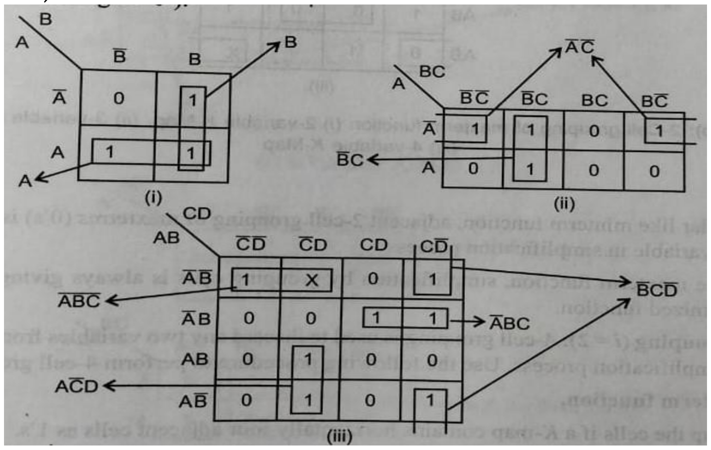
i. K-mapcelliscalledasIsolationgroupwhennoadjacenthorizontalorvertical cell is '1'for minterm and'0' for maxterm.
ii. Isolationcellcan'tbeusedforsimplification,itgivestheBooleanfunctionremain as same



2-Cellgroup(i=1):2cellgroupingisusedtodiscardanyvariablefromtwoadjacent cell in the simplification process

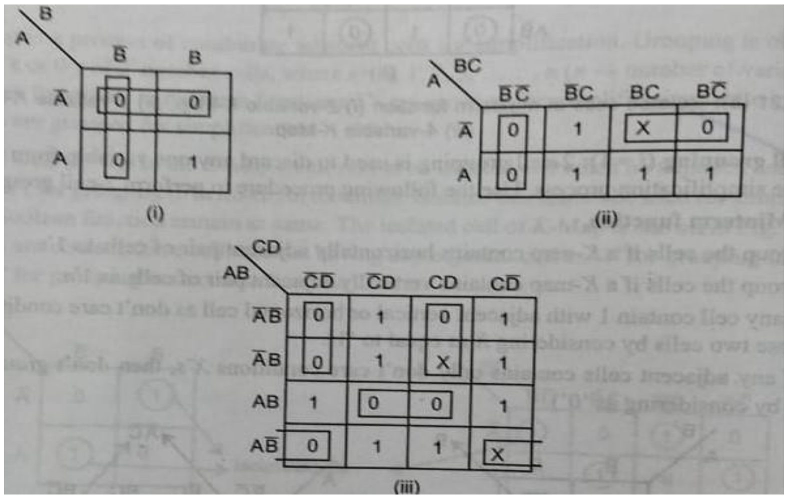
ProcedureforMintermfunction:

i. Groupthecellifak-mapcontainshorizontallyadjacentpair(2cell)ofcellsas1's
ii. Groupthecellifak-mapcontainsverticallyadjacentpair(2cell)ofcellsas1's
iii. Ifanycellcontain1withadjacentverticalorhorizontalcellasdon'tcare condition 'X' then group those two cells by considering X=1
iv. Ifanycellcontainonlydon'tcarecondition'X'thendon'tgroupthosecells(Discard by considering as X=0)



ProcedureforMaxtermfunction:

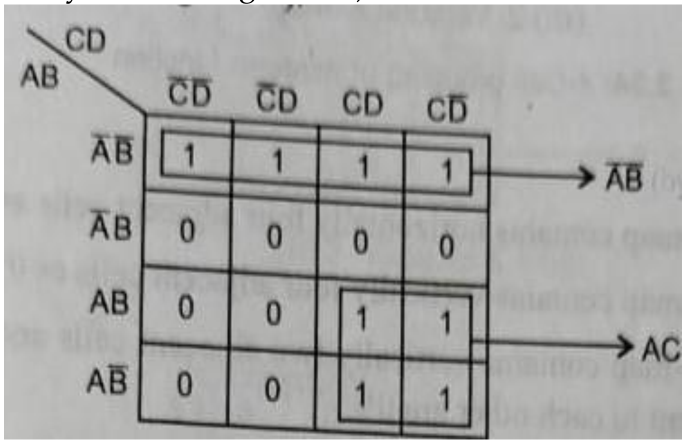
i. Groupthecellifak-mapcontainshorizontallyadjacentpairofcellsas0's
ii. Groupthecellifak-mapcontainsverticallyadjacentpair(2cell)ofcellsas0's
iii. Ifanycellcontain0withadjacentverticalorhorizontalcellasdon'tcare condition 'X' then group those two cells by considering X=0
iv. Ifanycellcontainonlydon'tcarecondition'X'thendon'tgroupthosecells(Discard by considering as X=1)

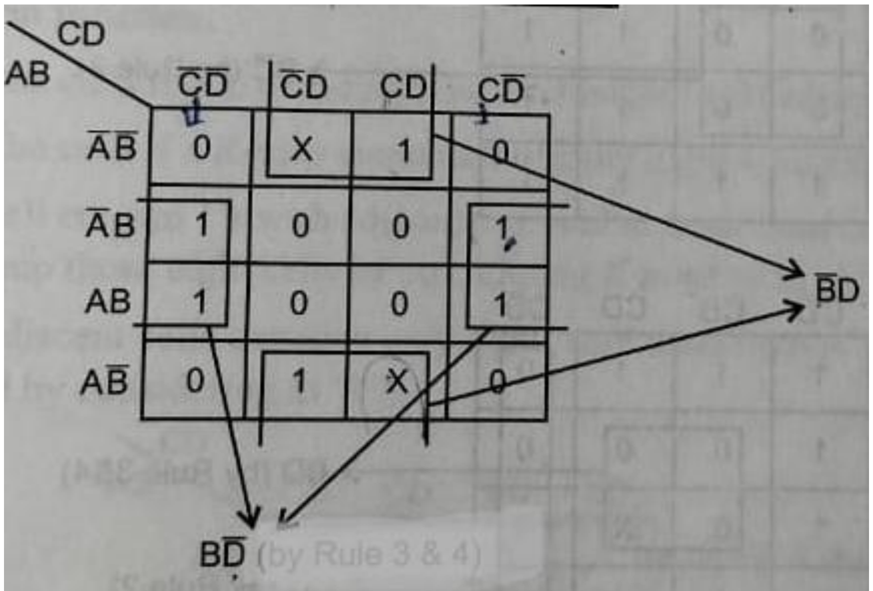


4-Cellgroup(i=2):4cellgroupingisusedtodiscardanytwovariablesfromfour(4) adjacent cells in the simplification process

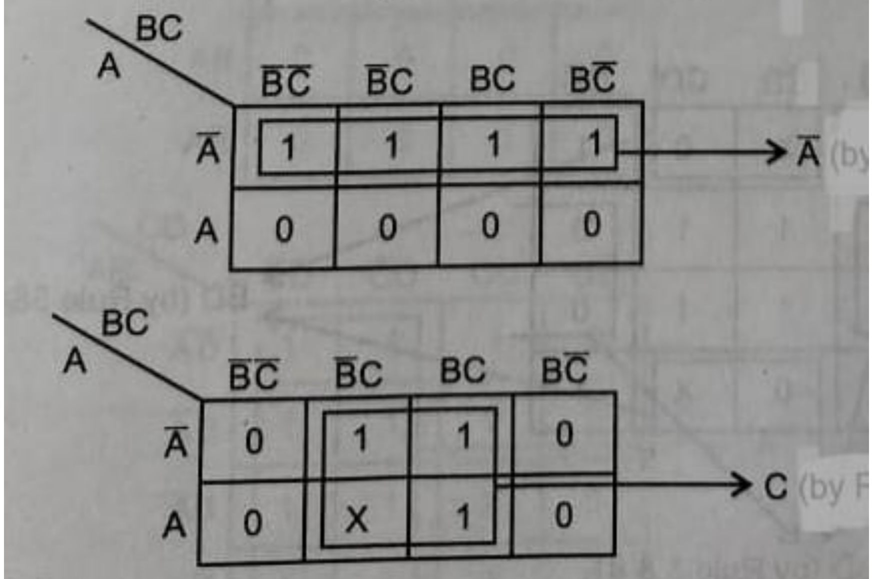
ProcedureforMintermfunction:

i. Groupthecellifak-mapcontainshorizontallyfour(4)adjacentofcellsas1's
ii. Groupthecellifak-mapcontainsverticallyfour)4)adjacentpailofcellsas1's
iii. Group the cell If a K-map contain vertically two adjacent cell and horizontal two adjacent cellwhich adjacent to each other are 1's.
iv. Ifany cellcontain 1's with adjacentvertically orhorizontal cell as don't care condition 'X' then group those four cell by considering X=1.
v. If any adjacent cell contain only don't care condition 'X' then don't group thosecells (Discard by considering as X=0)

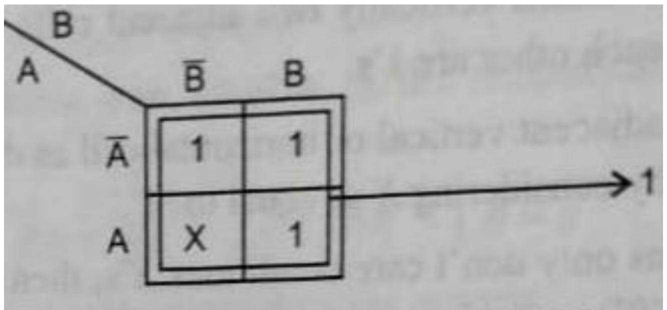




(i) 4-Variable *K*-Map



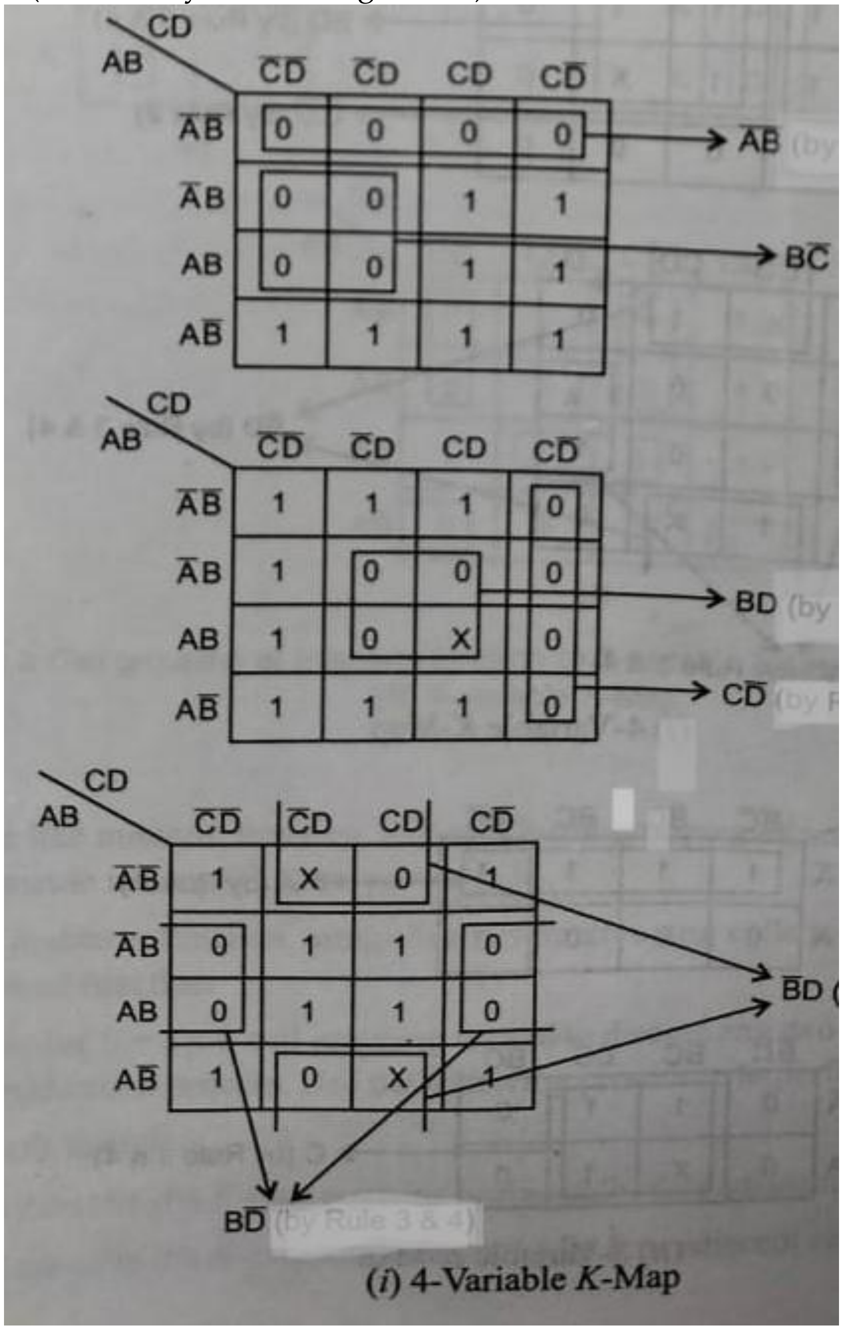
(ii) 3-Variable *K*-Map



(iii) 2-Variable *K*-Map

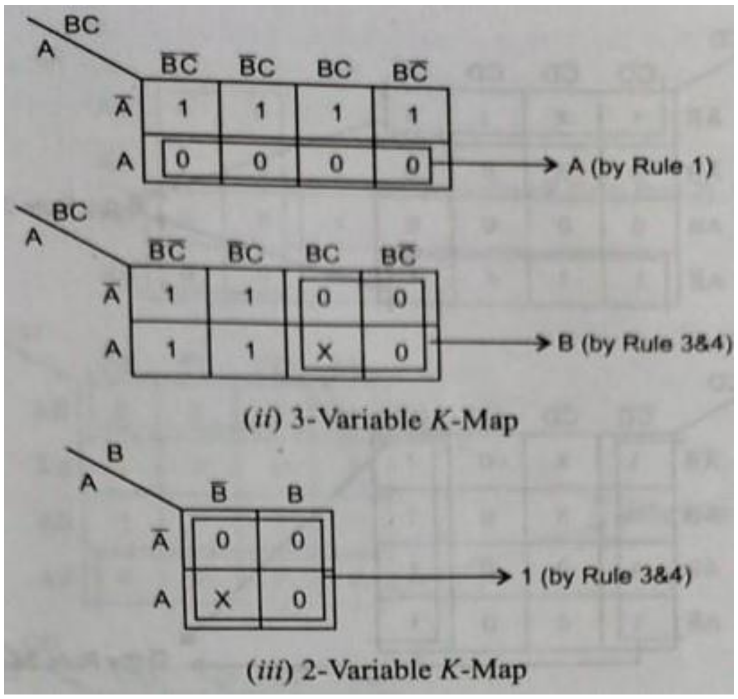
ProcedureforMaxtermfunction:

i. Groupthecellifak-mapcontainshorizontallyfour(4)adjacentofcellsas0's
ii. Groupthecellifak-mapcontainsverticallyfour)4)adjacentpailofcellsas0's
iii. Group the cell If a K-map contain vertically two adjacent cell and horizontal two adjacent cellwhich adjacent to each other are 0's.
iv. Ifany cellcontain 1's with adjacentvertically orhorizontal cell as don't care condition 'X' then group those four cell by considering X=0.
v. If any adjacent cell contain only don't care condition 'X' then don't group thosecells (Discard by considering as X=1)



CD
AB
$\overline{C}\overline{D}$ $\overline{C}D$ CD $C\overline{D}$
$\overline{A}\overline{B}$ 0 0 0 0
$\overline{A}B$ 0 0 1 1
AB 0 0 1 1
$A\overline{B}$ 1 1 1 1
$\overline{A}\overline{B}$
$B\overline{C}$
CD
AB
$\overline{C}\overline{D}$ $\overline{C}D$ CD $C\overline{D}$
$\overline{A}\overline{B}$ 1 1 1 0
$\overline{A}B$ 1 0 0 0
AB 1 0 X 0
$A\overline{B}$ 1 1 1 0
BD
$C\overline{D}$
CD
AB
$\overline{C}\overline{D}$ $\overline{C}D$ CD $C\overline{D}$
$\overline{A}\overline{B}$ 1 X 0 1
$\overline{A}B$ 0 1 1 0
AB 0 1 1 0
$A\overline{B}$ 1 0 X 1
$\overline{B}D$
$B\overline{D}$

(i) 4-Variable K-Map

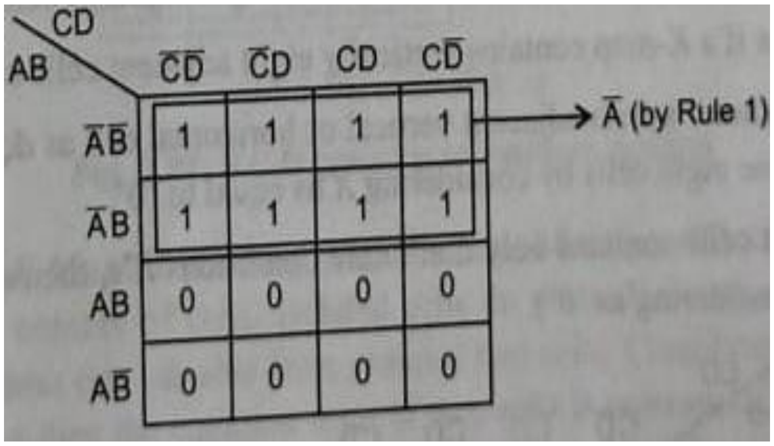


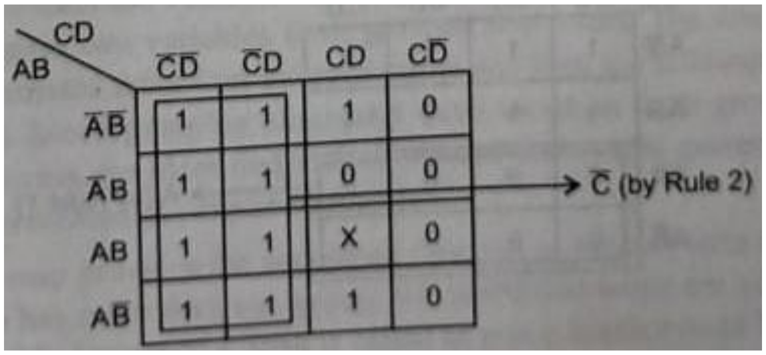
(*ii*) 3-Variable *K*-Map
(*iii*) 2-Variable *K*-Map

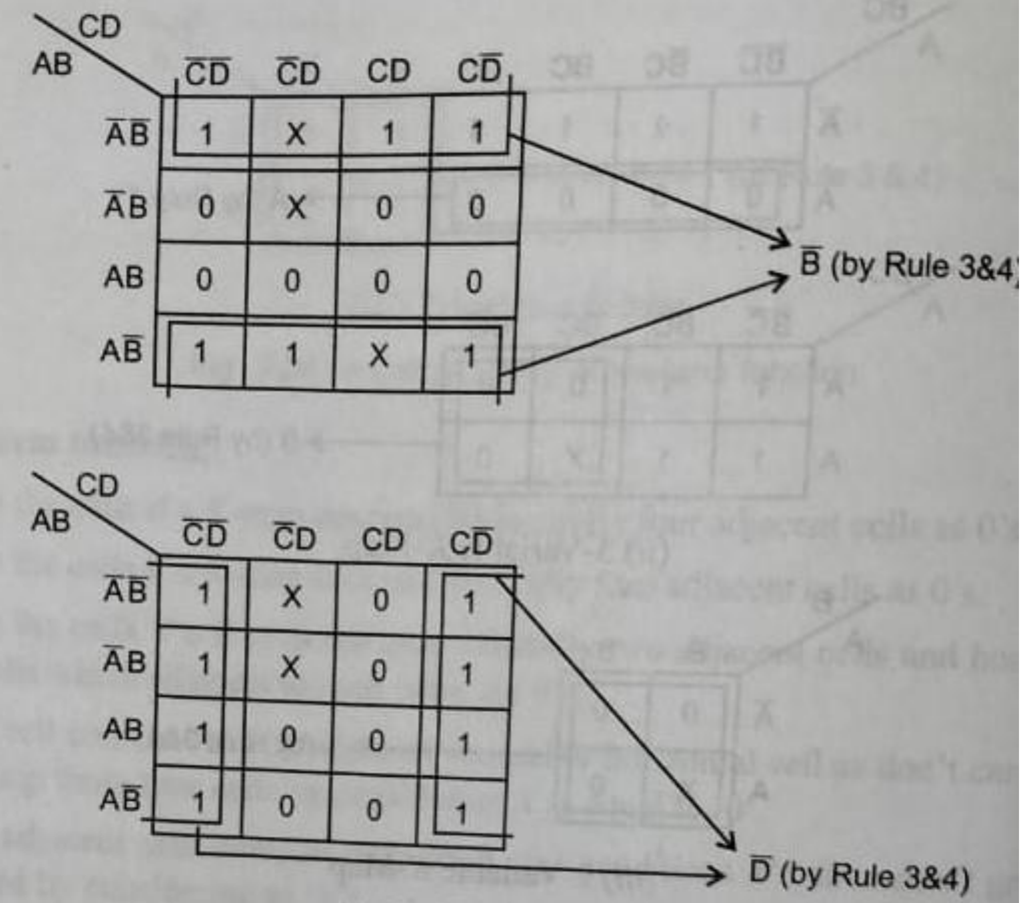
8-Cellgroup(i=3):8cellgroupingisusedtodiscardanythree(3)variablesfrom eight (8) adjacent cells in the simplification process

ProcedureforMintermfunction:

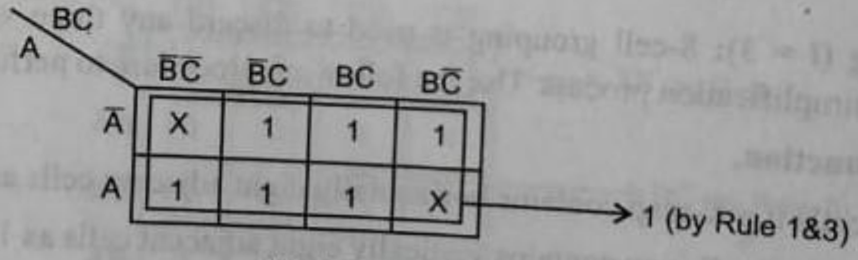
i. Groupthecellifak-mapcontainshorizontallyeight(8)adjacentofcellsas1's
ii. Groupthecellifak-mapcontainsverticallyeight(8))adjacentpailofcellsas1's
iii. Ifanycellcontain1'swithadjacentverticallyorhorizontalcellas don'tcare condition 'X' then group those eight (8) cell by considering X=1.
iv. If any adjacent cell contain only don't care condition 'X' then don't group thosecells (Discard by considering as X=0)







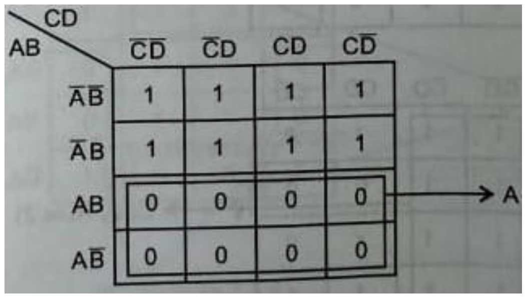
(*i*) 4-Variable *K*-Map



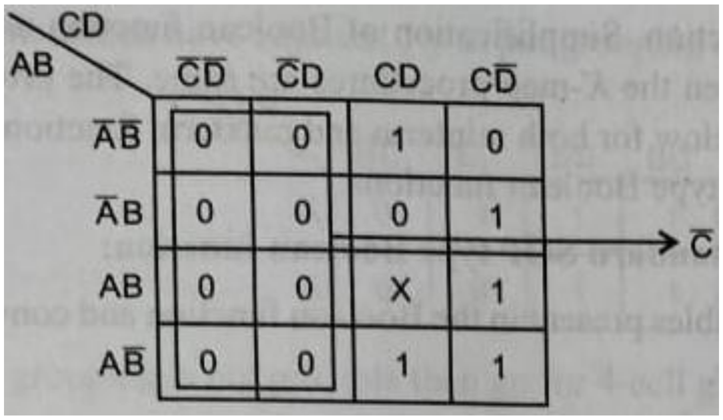
(*ii*) 3-Variable *K*-Map

ProcedureforMaxtermfunction:

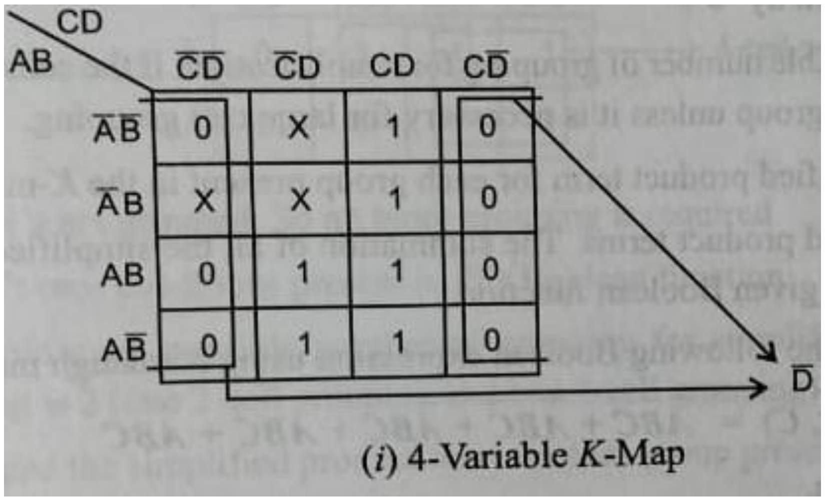
i. Groupthecellifak-mapcontainshorizontallyeight(8)adjacentofcellsas0's
ii. Groupthecellifak-mapcontainsverticallyeight(8))adjacentpailofcellsas0's
iii. Ifanycellcontain0'swithadjacentverticallyorhorizontalcellas don'tcare condition 'X' then group those eight (8) cell by considering X=0.
iv. If any adjacent cell contain only don't care condition 'X' then don't group thosecells (Discard by considering as X=1)

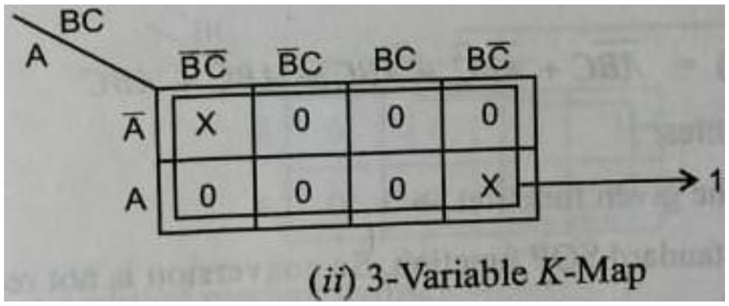


CD
AB
$\overline{C}\overline{D}$ $\overline{C}D$ CD $C\overline{D}$
$\overline{A}\overline{B}$ 1 1 1 1
$\overline{A}B$ 1 1 1 1
AB 0 0 0 0
$A\overline{B}$ 0 0 0 0
A



CD
AB
$\overline{C}\overline{D}$ $\overline{C}D$ CD $C\overline{D}$
$\overline{A}\overline{B}$ 0 0 1 0
$\overline{A}B$ 0 0 0 1
AB 0 0 X 1
$A\overline{B}$ 0 0 1 1
$\overline{C}$





(ii) 3-Variable K-Map

SHORTQUESTIONSANDANSWERS

1. **Definedigitalsystem?**

Ans.A digital system is a system which deals with discrete signal. The input and output of this system is two binary value which is 0 and 1. Examples of digital systems are mobile phones, radio, megaphones and many more

2. **Listtheapplicationsofdigitalsystem?**

Ans.MobilePhones,CalculatorsandDigitalComputers
Radios and communication Devices.

3. **Whatismeantbybit?**

Ans.Singledigitthatusedtorepresentthenumberiscalledbiti.e1or0

4. **Whatisradixnumbersystem?**

Ans. Radix (base)number systemisa generalrepresentationofallthenumber system. It represent the weight of each digits present in the number system.
Example:
Base of binary no. system =2
Baseofoctalno.system =8
Baseofhexadecimalno.system=16

5. **Definebinarycode?**

Ans. A group of binary bit that are used to represent the characters, numbers, lettersor words or symbol iscalled asbinary codes.

The digital data is represented, stored and transmitted as group of binary bits. This group is also called asbinary code. The binary code is represented by the number as well as alphanumeric letter.

6. **Whatareweightedbinarycodes?**

Ans. Acodewhichconsistsofbitweightforeachdigitpresentinthebinary code is called weighted binary codes

Example:

BCDcodes

7. **Whatarenon-weightedbinarycodes?**

Ans. Acodewhichisnothavinganybitweightforthedigitpresentinthe binary code is called non-weighted binary codes

Example:Excess-3code,graycode.

8. **Whatisgraycode?Whyisitcalledasreflectivecodeandcycliccode?**

Ans. Itisthe non-weightedbinary code, that means thereare nospecific weights assigned to the bit position. only one bit position will changeeach time the decimal number is incremented so called reflective code. Also the adjacent gray representation differs in only binary bit hence it is referred as cyclic code.

9. **StatetheassociativepropertyofBooleanalgebra**

Ans. Associative law defines that the grouping of variable in the multivariable AND and OR operation does not change the output.

i. A+(B+C)=(A+B)+ C

ii. A.(B.C)=(A.B).C

10. **StatethedistributivepropertyofBooleanalgebra**

Ans. Associative law defines that the distribution of variable with AND operation over OR operation is equal to distribution of variable with OR operation over AND operation

i. A+BC=(A+B)(A+C)

ii. A(B+C)=AB+AC

11. **StatetheDeMorgan’stheorem**

i. $\overline{\square.\square}=\overline{\square}+\overline{\square}$

ii. $\overline{\square+\square}=\overline{\square}.\overline{\square}$

2.COMBINATIONALLOGICCIRCUITS

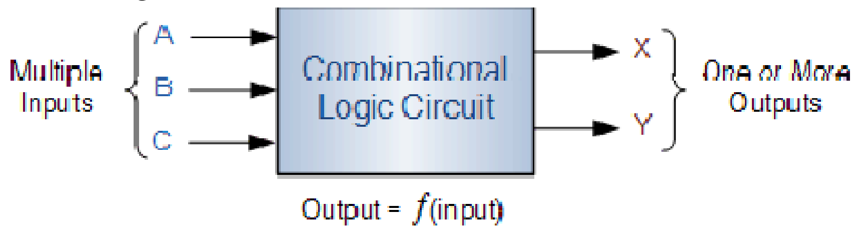
Givetheconceptofcombinationallogic circuits:

Acombinationalcircuitisthedigitallogiccircuitin whichtheoutputdependson the combination of present inputs applied to the circuit and It does not depend past input

Combinationalcircuitsaredevelopedusingcombinationof AND,OR,NOT, NAND, and NOR logic gates.

CombinationalLogic Circuitsare memory lessdigitallogiccircuitswhoseoutput at any instant in time depends only on the combination of its inputs

Thecombinationallogiccircuitshavenofeedbackcircuitisused.



Halfaddercircuitandverifyitsfunctionalityusingtruthtable:

Halfadderisacombinationalcircuitwhichconsistsoftwobinaryinputvariables called augend and addendand two binary output variables called sum and carry. In the addition result, the lower significant bit is called as sum and the higher significant bit is called as carry.

Truthtable

Input		Output	
A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

K –mapfor sum

	0	1
0	**0**	**1**
1	**1**	**0**

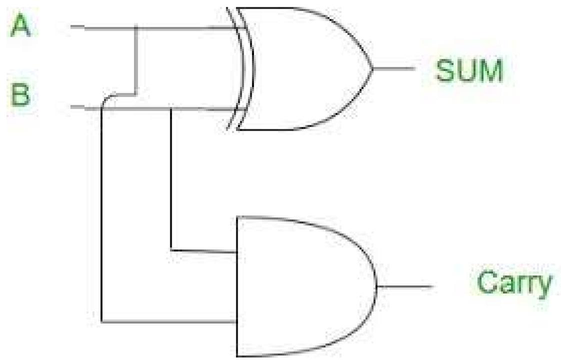
Sum=$\overline{\square}\square+\square\overline{\square}=\square\oplus\square$

K –mapfor carry

	0	1
0	**0**	**0**
1	**0**	**1**

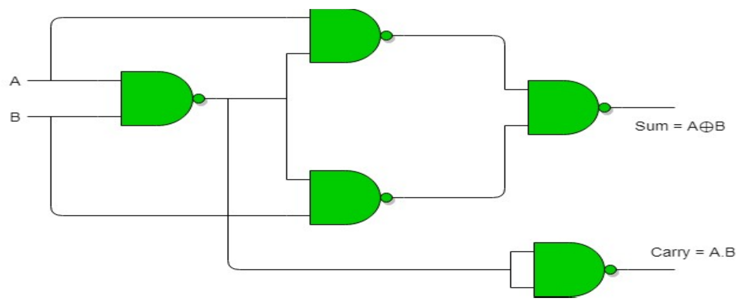
Carry=AB

Cktdiagram

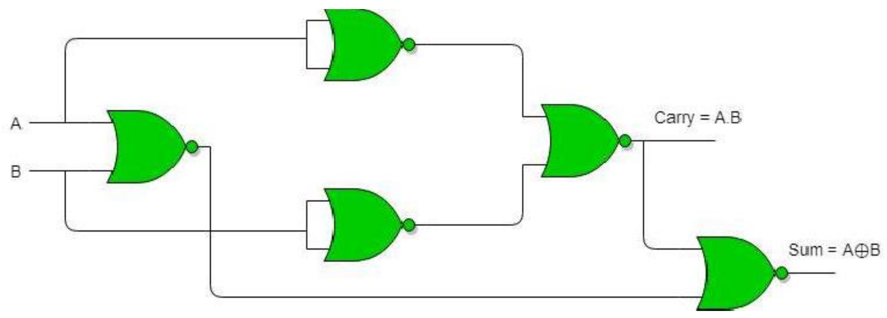


RealizeaHalf-adderusingNANDgatesonlyandNORgatesonly.

Half-adderusingNANDgates



Half-adderusingNANDgates



Fulladdercircuitandexplainitsoperationwithtruthtable:

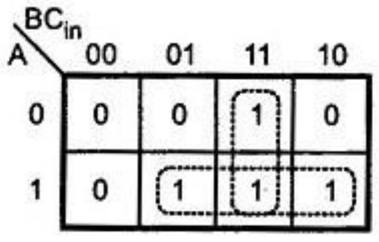
Full adder is a combinational circuit which consists of three binary input variables called augend and addendand two binary output variables called sum and carry. In the addition result, the lower significant bit is called as sum and the higher significant bit is called as carry

Truthtable

Inputs			Outputs	
A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

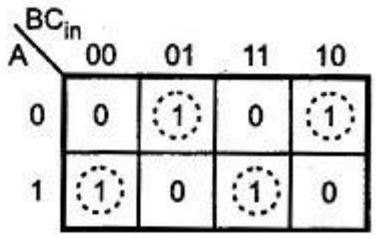
k-map

For Carry (C_{out})



$C_{out} = AB + A\,C_{in} + B\,C_{in}$

For Sum

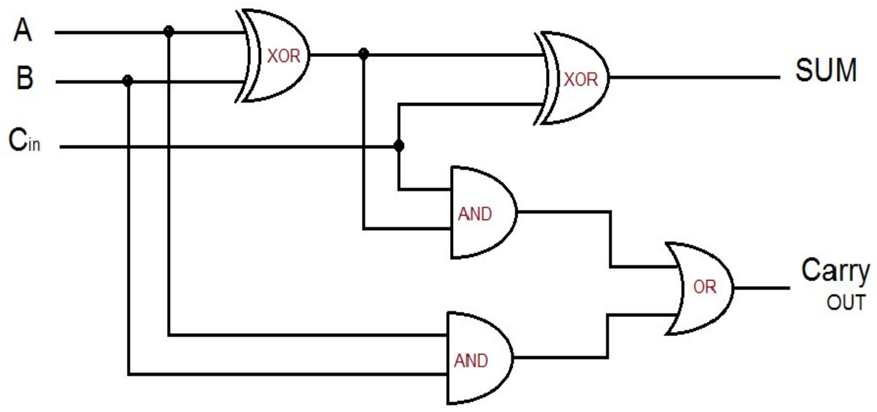


$\mathrm{Sum} = \bar{A}\,\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\,\bar{C}_{in} + ABC_{in}$

K-mapcanbesimplifiedas

$$
\begin{aligned}
\text{SUM} &= \bar{A}\,\bar{B}\,C_{in} + \bar{A}\,B\,\bar{C}_{in} + A\,\bar{B}\,\bar{C}_{in} + A\,B\,C_{in} \\
&= C_{in}\,(\bar{A}\,\bar{B} + A\,B) + \bar{C}_{in}\,(A\,\bar{B} + \bar{A}\,B) \\
&= C_{in}\,\left[(\bar{A} + B)\cdot(A + \bar{B})\right] + \bar{C}_{in}\,(A\,\bar{B} + \bar{A}\,B) \\
&= C_{in}\,(\overline{A\bar{B}}\cdot\overline{\bar{A}\,B}) + \bar{C}_{in}\,(A\,\bar{B} + \bar{A}\,B) \\
&= C_{in}\,(\overline{A\bar{B} + \bar{A}\,B}) + \bar{C}_{in}\,(A\,\bar{B} + \bar{A}\,B) \\
&= C_{in} \oplus (A\,\bar{B} + \bar{A}\,B) \\
&= C_{in} \oplus (A \oplus B)
\end{aligned}
$$

Fulladdercircuitdiagram



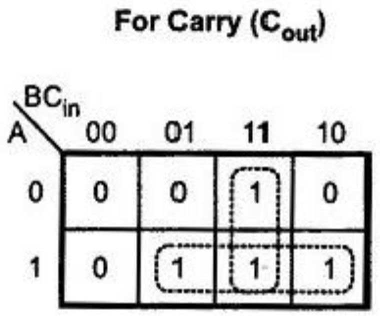
Realizefull-adderusingtwoHalf-addersandanOR– gateandwritetruth table.

Thefulladdercanbeimplementedwithtwohalfaddersby cscading them.ThesumoutputoffirsthalfadderisEx-ORofAandB. Thesum outputof full adder is Ex-OR of Cin and output of first half adder.

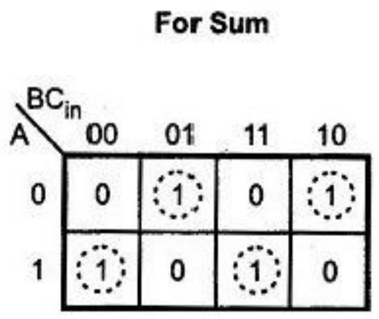
Truthtable

Inputs			Outputs	
A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

k-map



$C_{out} = AB + A\,C_{in} + B\,C_{in}$

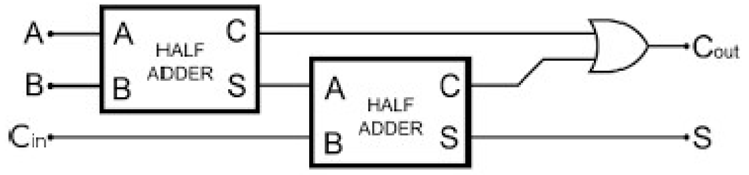


$Sum = \bar{A}\,\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\,\bar{C}_{in} + ABC_{in}$

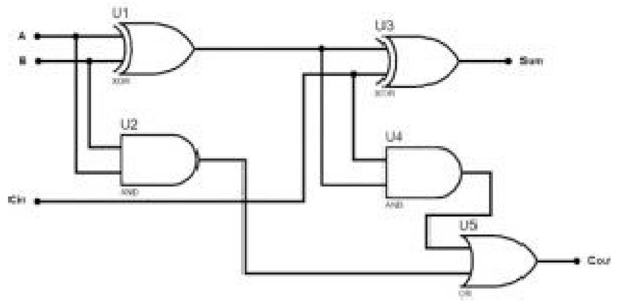
K-mapcanbesimplifiedas

$$\begin{aligned}
\textbf{SUM} &= \bar{A}\,\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in} \\
&= C_{in}\,(\bar{A}\,\bar{B} + AB) + \bar{C}_{in}\,(A\bar{B} + \bar{A}B) \\
&= C_{in}\,\left[(\bar{A} + B)\cdot(A + \bar{B})\right] + \bar{C}_{in}\,(A\bar{B} + \bar{A}B) \\
&= C_{in}\,(\overline{A\bar{B}}\cdot\overline{\bar{A}B}) + \bar{C}_{in}\,(A\bar{B} + \bar{A}B) \\
&= C_{in}\,(\overline{A\bar{B} + \bar{A}B}) + \bar{C}_{in}\,(A\bar{B} + \bar{A}B) \\
&= C_{in} \oplus (A\bar{B} + \bar{A}B) \\
&= C_{in} \oplus (A \oplus B)
\end{aligned}$$

$$\begin{aligned}
\textbf{Cout} &= AC_{in} + BC_{in} + AB \\
&= (B + \bar{B})AC_{in} + (A + \bar{A})BC_{in} + AB \\
&= ABC_{in} + A\bar{B}C_{in} + ABC_{in} + \bar{A}BC_{in} + AB \\
&= \bar{A}BC_{in} + A\bar{B}C_{in} + ABC_{in} + AB \\
&= C_{in}(\bar{A}B + A\bar{B}) \quad + ABC_{in} + AB \\
&= C_{in}(A \oplus B) \quad + AB(C_{in} + 1) \\
&= C_{in}(A \oplus B) \quad + AB
\end{aligned}$$



Blockdiagram



Circuit diagram

Fullsubtractorcircuitandexplainitsoperationwithtruthtable.:

a. Half adder circuit and verify its functionality using truth table: Half subtract is a combinational circuit which consists of two binary input variables calledminuendand subtrahendandtwobinaryoutputvariablescalleddifferenceand borrow. In the two bit result, the lower significant bit is called as difference and the higher significant bit is called as borrow.

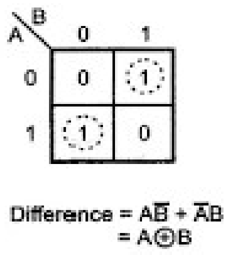
Truthtable

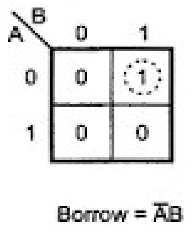
A	B	Borrow	Difference
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

K- map

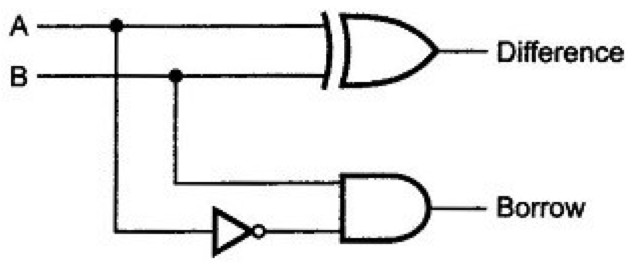
For Difference

For Borrow





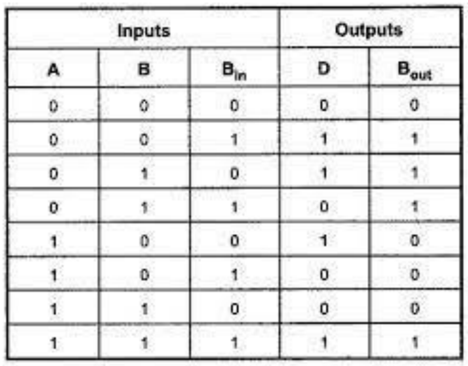
Logicdiagram


A
B
Difference
Borrow

b.Fullsubtractorcircuit ndexplainitsoperationwithtruthtable.:

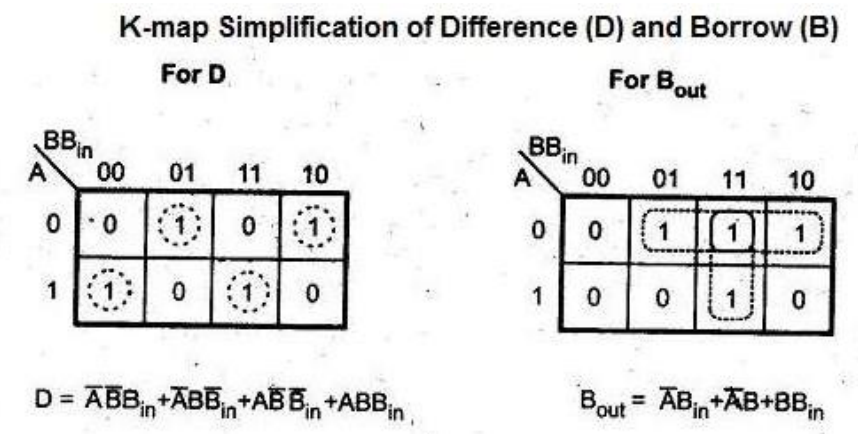
Fullsubtractionisacombinationalcircuitwhichconsistsofthreebinaryinput variablescalledminuendsandsubtrahends andtwobinaryoutputvariablescalled difference and borrow out. In the subraction result, the lower significant bit is called as differenceand the higher significant bit is called as borrowout

Truthtable

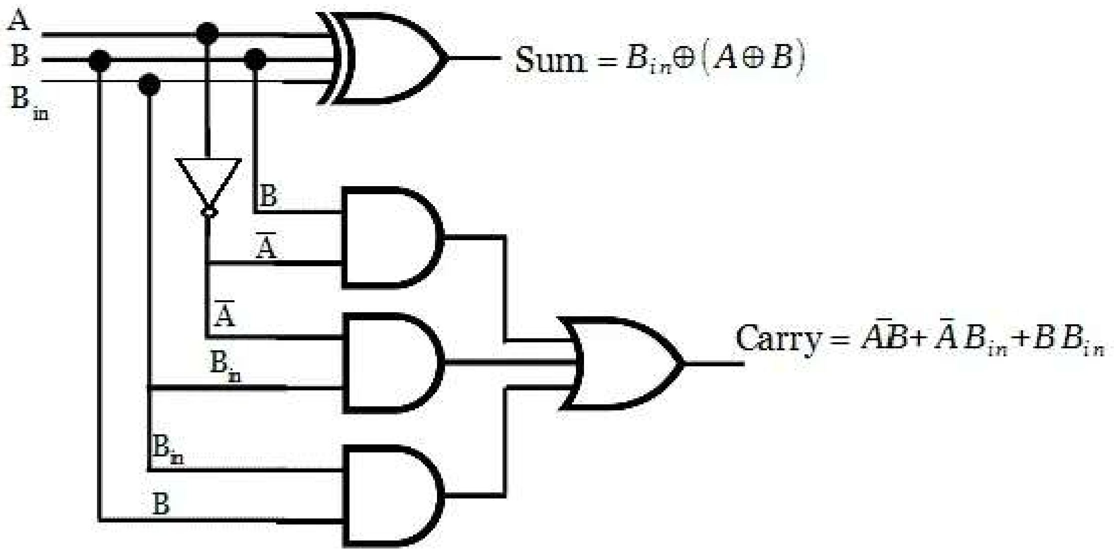


Inputs			Outputs	
A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Table 3.9 Truth table for full-subtractor


K-map Simplification of Difference (D) and Borrow (B)
For D
For B_{out}
BB_{in}
A
00
01
11
10
$D = \bar{A}\bar{B}B_{in}+\bar{A}B\bar{B}_{in}+A\bar{B}\bar{B}_{in}+ABB_{in}$
$B_{out} = \bar{A}B_{in}+\bar{A}B+BB_{in}$

$$\begin{aligned}
\text{Difference} &= \bar{A}\,\bar{B}B_{in}+\bar{A}\,B\,\bar{B}_{in}+A\,\bar{B}\,\bar{B}_{in}+A\,BB_{in}\\
&= B_{in}(\bar{A}\,\bar{B}+A\,B)+\bar{B}_{in}(\bar{A}\,B+A\,\bar{B})\\
&= B_{in}(A\odot B)+\bar{B}_{in}(A\oplus B)\\
&= B_{in}(\overline{A\oplus B})+\bar{B}_{in}(A\oplus B)\\
&= B_{in}\oplus(A\oplus B)
\end{aligned}$$



A
B
B in
Sum = $B_{in} \oplus (A \oplus B)$
B
$\bar{A}$
$\bar{A}$
B_{in}
B_{in}
B
Carry = $\bar{A}B + \bar{A}B_{in} + BB_{in}$

Logic circuit for Full subtractor

2.5Realizefull-subtractionusingtwoHalf-subtractor andanOR–gateand write truth table.

The full subtractor can be implemented with two half subtractors by cascading them. The difference output of first half subtractor is Ex-OR of A and B. The difference output of full subtractor is Ex-OR of Bin and output of first half subtractor.

Similarly,theborrowoutputoffirsthalfsubtractorisORedwiththeborrow outputofsecondhalfsubtractortogettheborrowoutputoffullsubtractor.

SimplificationofDifferenceandBorrow

$$\text{Difference} = \bar{A}\,\bar{B}B_{in} + \bar{A}\,B\,\bar{B}_{in} + A\,\bar{B}\,\bar{B}_{in} + A\,B B_{in}$$

$$= B_{in}(\bar{A}\,\bar{B} + A\,B) + \bar{B}_{in}(\bar{A}\,B + A\,\bar{B})$$

$$= B_{in}(A \odot B) + \bar{B}_{in}(A \oplus B)$$

$$= B_{in}(\overline{A \oplus B}) + \bar{B}_{in}(A \oplus B)$$

$$= B_{in} \oplus (A \oplus B)$$

$$\text{Borrow} = \bar{A}B + \bar{A}B_{in} + BB_{in}$$

$$= \bar{A}B + \bar{A}B_{in}(B+\bar{B}) + BB_{in}(A+\bar{A})$$

$$= \bar{A}B + \bar{A}BB_{in} + \bar{A}\bar{B}B_{in} + ABB_{in} + \bar{A}BB_{in}$$

$$= \bar{A}B(1 + B_{in} + B_{in}) + \bar{A}\bar{B}B_{in} + ABB_{in}$$

$$= \bar{A}B + \bar{A}\bar{B}B_{in} + ABB_{in}$$

$$= \bar{A}B + B_{in}(\bar{A}\bar{B} + AB)$$

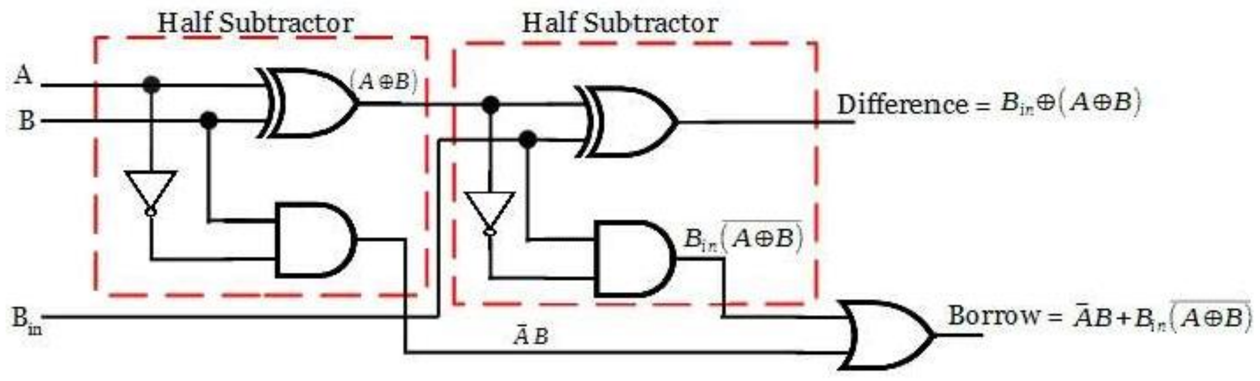
$$= \bar{A}B + B_{in}(A \odot B)$$

$$= \bar{A}B + B_{in}\overline{(A \oplus B)}$$

$A+A=A$

$A+1=1$

Usingthesimplified boolean expressions fordifferenceand borrowoutput, the full subtractor can be realized

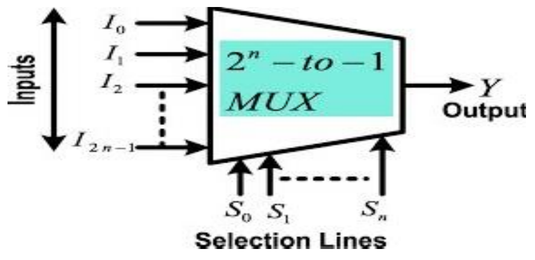


Half Subtractor
Half Subtractor
A
B
B_{in}
$(A\oplus B)$
Difference = $B_{in}\oplus(A\oplus B)$
$B_{in}\overline{(A\oplus B)}$
$\bar{A}B$
Borrow = $\bar{A}B+B_{in}\overline{(A\oplus B)}$

Realization of full subtractor with two half subtractors

2.7Operationof4X1Multiplexersand1 X4demultiplexer

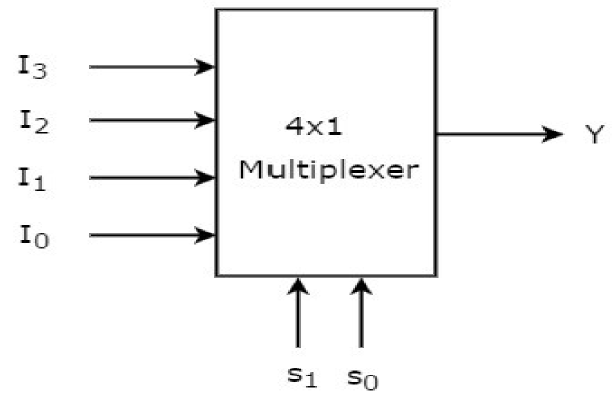
Multiplexer isacombinationalcircuitthathasmaximumof2^nnumberdata inputs,'n'numberofselectioncontrollinesandsingleoutputline.Oeofthesedata inputs will be connected to the output based on the values of selection lines. Shown in figure.



Where $I_0, I_1, I_3, I_4 \ldots\ldots I_n$ aretheinput line,Yis theoutputlineand$S_0, S_1, \ldots.. S_n$are theselection line.

a.4x1Multiplexer

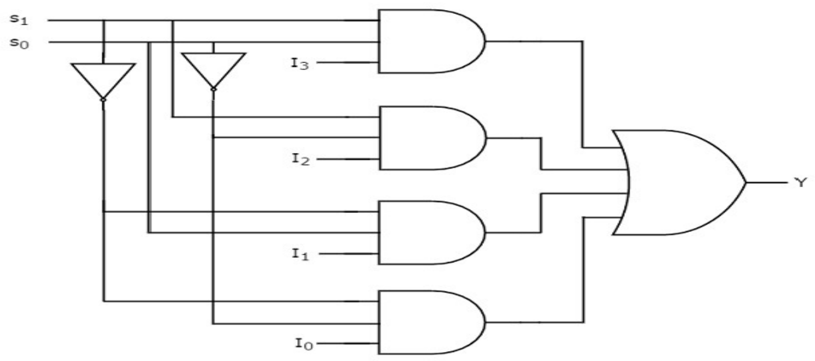
4x1MultiplexerhasfourdatainputsI_3, I_2, I_1&I_0,twoselectionliness_1&$s_0$andone output Y. The **block diagram** of 4x1 Multiplexer is shown in the following figure.



One of these 4 inputs will be connected to the output based on the combinationof inputs present at these two selection lines. Truth table of 4x1 Multiplexer is shown below.

Selection Lines		Output
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

$$Y = \overline{S_1}\,\overline{S_0}I_0 + \overline{S_1}S_0I_1 + S_1\overline{S_0}I_2 + S_1S_0I_3$$



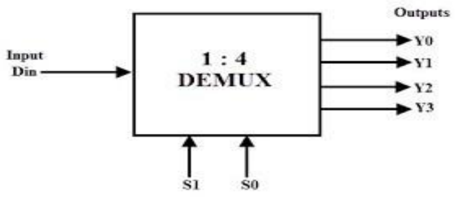
Logiccircuitdiagram

De-Multiplexer:

De-Multiplexeris a combinational circuit that performs the reverse operation of Multiplexer. It has single input, ‘n’ selection lines and maximum of 2^n outputs. De-Multiplexer is also called as **De-Mux**.

1x4De-Multiplexer

1x4 De-Multiplexer has one input I, two selection lines, s1& s0and four outputs Y3, Y2, Y1&Y0. Theblock diagramof 1x4 De-Multiplexer is shown in the following figure.



The single input ‘I’ will be connected to one of the four outputs, Y3to Y0based on the values of selection lines s1& s0. TheTruth tableof 1x4 De-Multiplexer is shown below.

Selection inputs		outputs			
S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Fromtheabove Truth table,wecan directly write the **Boolean functions**for each output as

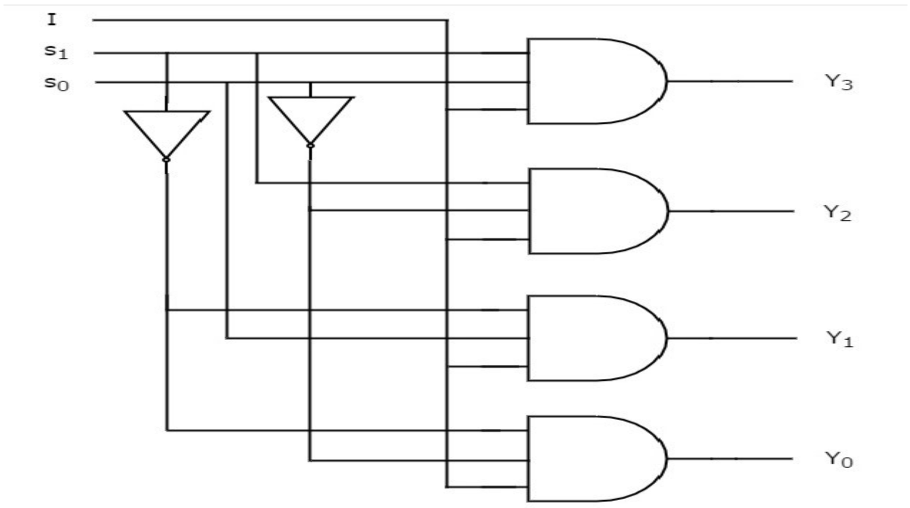
Y_0=I

$S_1S_0Y_1$=IS_1

S_0Y_2=IS_1

S_0Y_3=I

S_1S_0



Logiccircuitdiagram

WorkingofBinary-DecimalEncoder&3 X8Decoder.

a.**Decoder**

Decoder isacombinationalcircuitthathasmultipleinputmultipleoutputthatis'n' number ofinput lines and maximum of 2^nnumber of output lines.

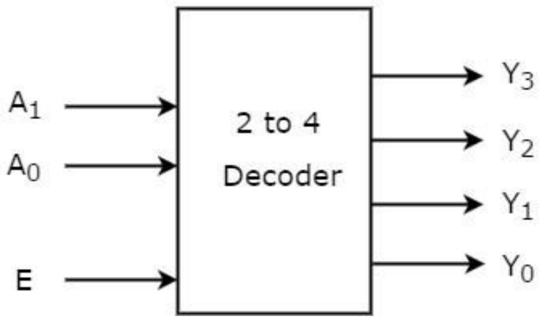
OneoftheseoutputswillbeactiveHighbasedonthecombinationofinputs present, when the decoder is enabled. That means decoder detects a particular code. i.e Inthedecoder,thecombinationofinput informationlinesdefinethelog coutputofany one. outputlineaslogichighat atimeandtherestoftheoutputlinesarebeigfixedto logic 0. The outputs of the decoder are nothing but the min termsof 'n' input variableslines, when it is enabled.

2 to4Decoder

Let2to4DecoderhastwoinputsA$_1$&A$_0$andfouroutputsY$_3$,Y$_2$,Y$_1$&Y$_0$.The **block diagram** of 2 to 4 decoder is shown in the following figure.

i.e inputlines'n'=2

output lines=2^n=2^2=4



One ofthese four outputs will be '1' foreach combinationof inputs when enable, Eis'1'.
The **Truth table** of 2 to 4 decoder is shown below.

Enable	Inputs		Outputs			
E	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

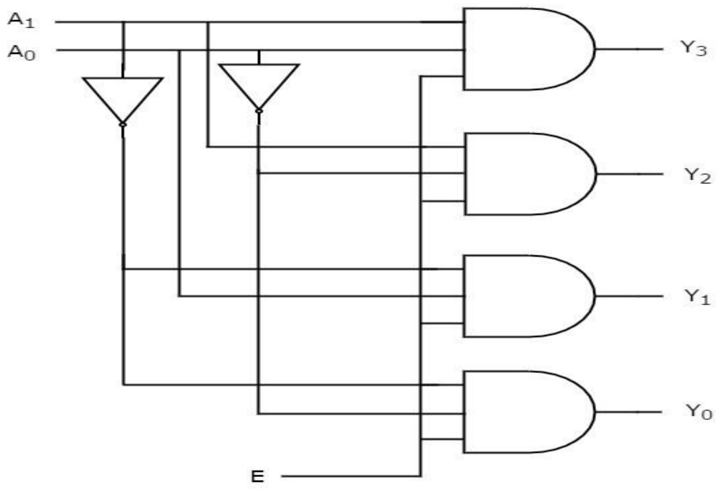
FromTruthtable,wecanwritethe**Booleanfunctions**foreachoutputas

Y_0=E

$A_1A_0Y_1$=E

$A_1A_0Y_2$= E

A_1 A_0Y_3= E



3 to8Decoder

Let3to8Decoderhas3inputsA_2A_1&A_0and8outputsY_7,Y_6,Y_5,Y_4,Y_3,Y_2,Y_1&Y_0. The **block diagram** of 3 to 8 decoder is shown in the following figure.

i.einputlines‘n’=3

outputlines=2^n=2^3=8

Enable	Inputs			Outputs							
E	A_3	A_1	A_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

FromTruthtable,wecanwritethe**Booleanfunctions**foreachoutputas

$$Y_0=E\overline{A_2}\ \overline{}\ \overline{A_1}$$

$$A_0Y_1=\overline{E}\ \overline{A_2}\ A_1$$

$$A_0Y_2=\overline{E}\ A_2\ \overline{A_1}$$

$$A_0Y_3=\overline{E}\ A_2\ A_1$$

$$A_0Y_4=E\ A_2\ A_1$$

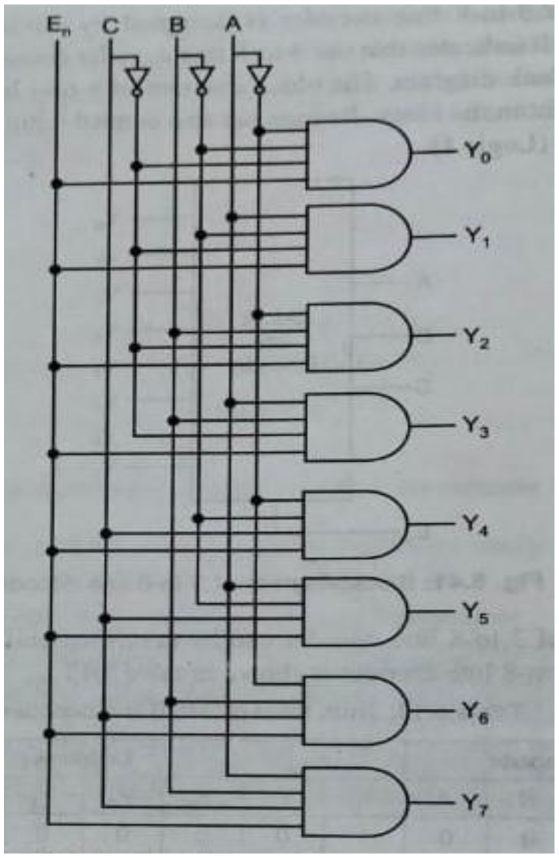
$$A_0Y_5=E\ \overline{A_2}\ \overline{A_1}$$

$$A_0Y_6=E\ \overline{A_2}\ A_1$$

$$A_0Y_7=EA_2A_1\overline{A}$$

0

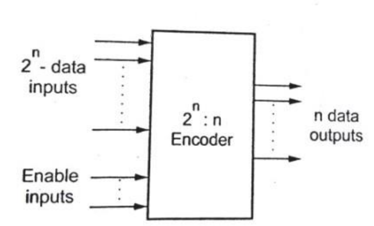
Logicalcircuitoftheaboveexpressionsisgivenbelow:



Logicdiagram3to8linedecoder

Encoder:

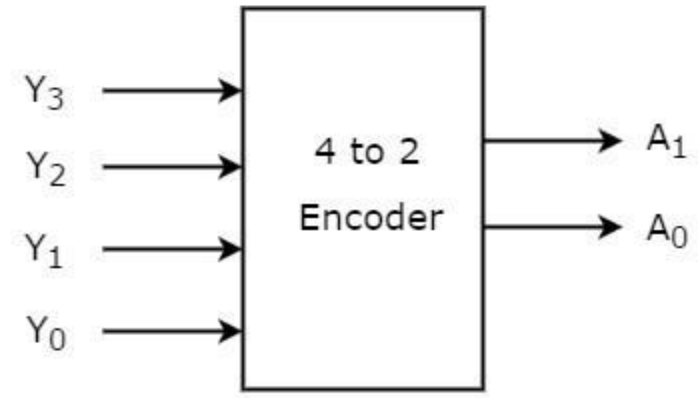
An encoder is a multiple input multi output combinational digital circuit that performs the inverse operation of a decoder. It means that an encoder converts the 2^nnumber of coded inputs into n number of coded outputs.



The output lines of a digital encoder generate the binary equivalent of the input line whose value is equal to 1 and are available to encode either a decimal or hexadecimal input pattern to typically a binary or B.C.D (binary coded decimal) output code

4 to2lineEncoder:

There are four inputs (Y0, Y1, Y2, and Y3) and two outputs (A0 and A1) in the 4 to2lineencoder.Inaddition,Togettherespectivebinarycodeontheoutputside,one inputlineatatimeissettotrueina4-inputline.The4to2lineencoder'sblock diagram and truth table are shown below.



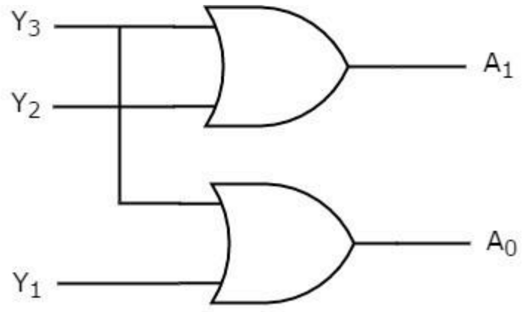
Inputs				Outputs	
Y_3	Y_2	Y_1	Y_0	A_1	A_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

ThetermsA0andA1arelogicallyexpressedasfollows:

$$A_1=Y_3+Y_2$$
$$A_0=Y_3+Y_1$$

Circuit Diagram

TwoinputORgatescanbeusedtoimplementtheaforementionedtwoBoolean functions. Further, The 4 to 2 encoder circuit diagram is given in the graphic below.



Usesof Encoder

Inalldigitalsystems,thesesystemsarerelativelysimpletooperate.

Toconvertadecimalnumber to a binarynumber,encodersareemployed.Thegoalisto complete a binary operation like addition, subtraction, multiplication, and so on.

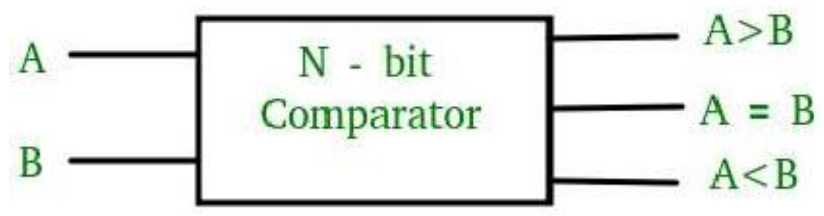
Disadvantages

Thedisadvantagesofastandardencoderarelistedbelow.

- Whenalloftheencoder'soutputsare0,there isambiguity.Becausewhenonlytheleast significantinputisoneorwhenallinputsarezero,itcouldbethecode inputs. n atchingthe
- Whenmorethanoneinputissettohigh,theencodergeneratesanoutputthatmayor maynotbethepropercode.IfbothY3andY6are'1',forexample,theecoder outnputs 111.ThisisneitherthecomparablecodeforY3,whenitis'1',norisittheequivalent code for Y6, when it is '1'.

WorkingofTwobitmagnitudecomparator.

AmagnitudedigitalComparatorisacombinationalcircuitthat**comparestwo digitalor binary numbers** in ordertofindoutwhetheronebinarynumber isequal,less than, or greater than the other binary number. We logically design a circuit for which we will have two inputs one for A and the other for B and have three output terminals, one for A > B condition, one for A = B condition, and one for A < B condition.

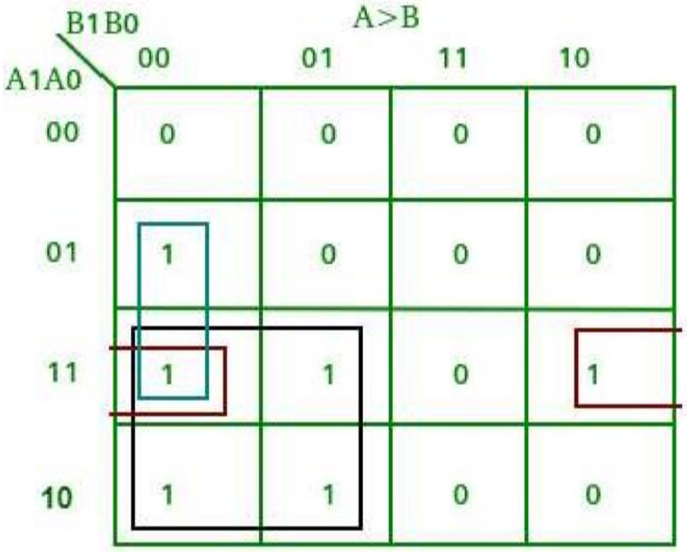


Truthtable

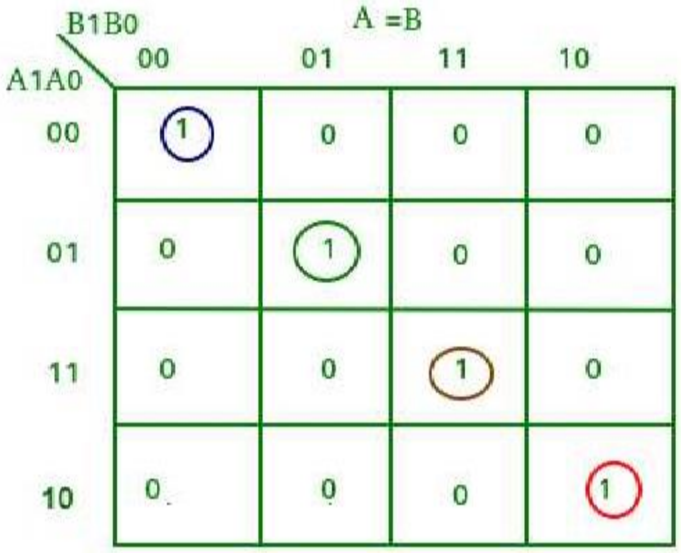
Input				Out put		
A		B		A>B	A=B	A<B
A_1	A_0	B_1	B_0			
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1

0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

FromtheabovetruthtableK-mapforeachoutputcanbedrawnasfollows:



$A>B=A_1A_0\bar{B}_0+\bar{B}_1\bar{B}_0A_0+A_1\bar{B}_1$

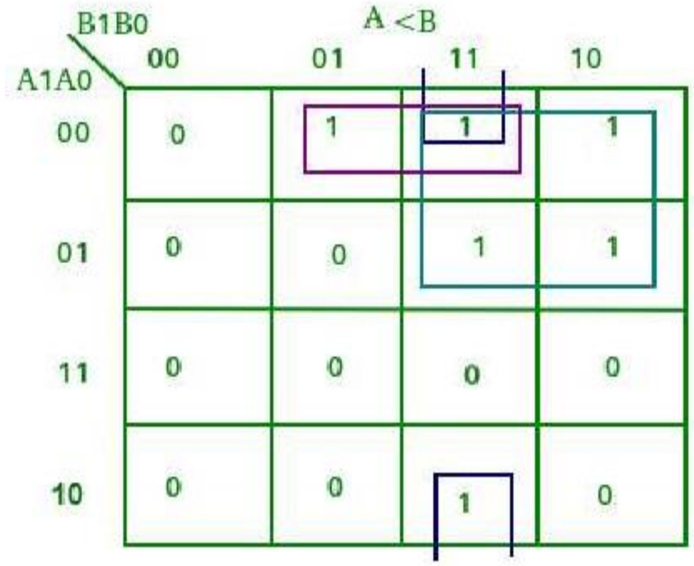


$A=B\quad=\bar{A}_1\bar{A}_0\bar{B}_1\bar{B}_0+\bar{A}_1A_0\bar{B}_1B_0+A_1A_0B_1B_0+A_1\bar{A}_0B_1\bar{B}_0$

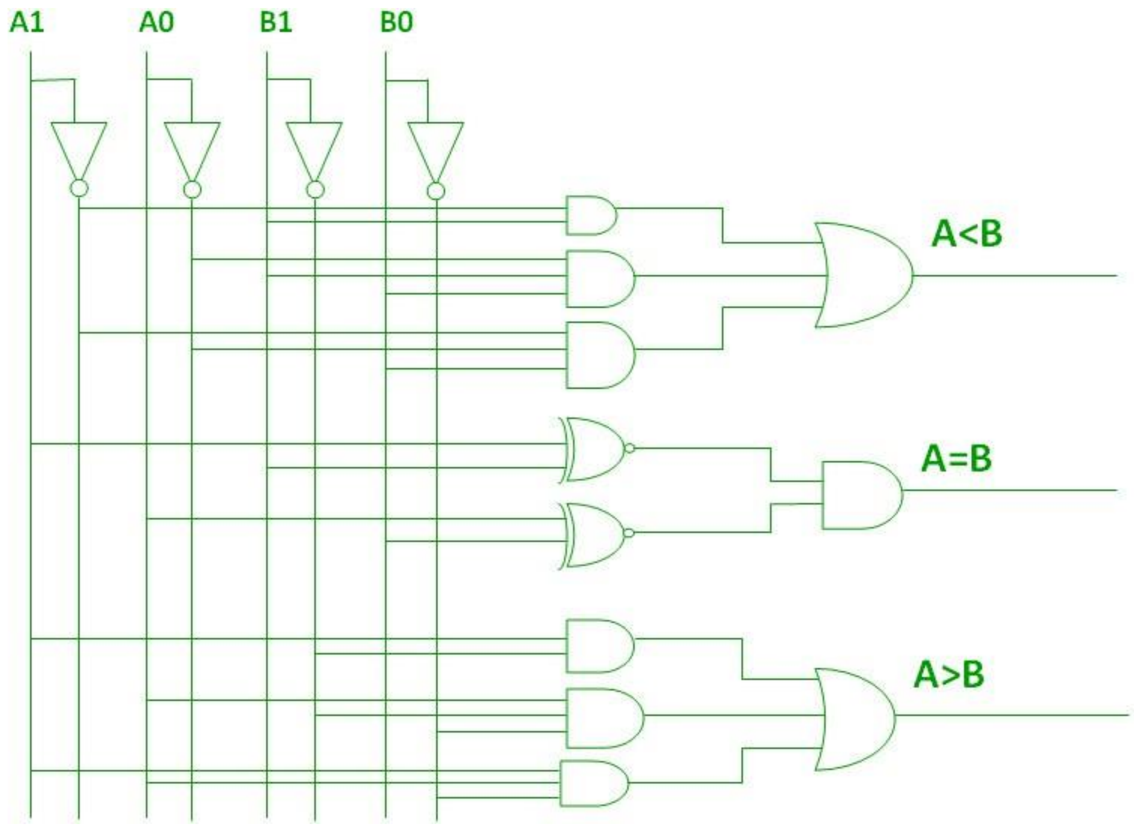
$=(\bar{A}_1\bar{B}_1(\bar{A}_0\bar{B}_0+A_0B_0)+A_1B_1(A_0B_0+\bar{A}_0\bar{B}_0))$

$=(\bar{A}_1\bar{B}_1+A_1B_1)(A_0B_0+\bar{A}_0\bar{B}_0)$

$=(A_1\odot B_1)(A_0\odot B_0)$

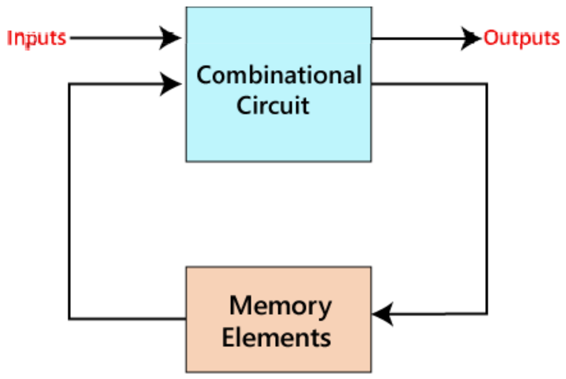


A<B $=\bar{A}_1\bar{A}_0B_1+B_1B_0\bar{A}_0+\bar{A}_1B_1$


A1
A0
B1
B0
A<B
A=B
A>B

3.SEQUENTIALLOGIC CIRCUITS

The outputs of the sequential circuits depend on both the combination of present inputs and previous outputs. The previous output is treated as the present state. So, the sequentialcircuitcontainsthecombinationalcircuitanditsmemory storageelements. A sequentialcircuitdoesn'tneedtoalwayscontainacombinational circuit.So,the sequential circuit can contain only the memory element.



BLOCKDIAGRAMOFSEQUENTIALCKT

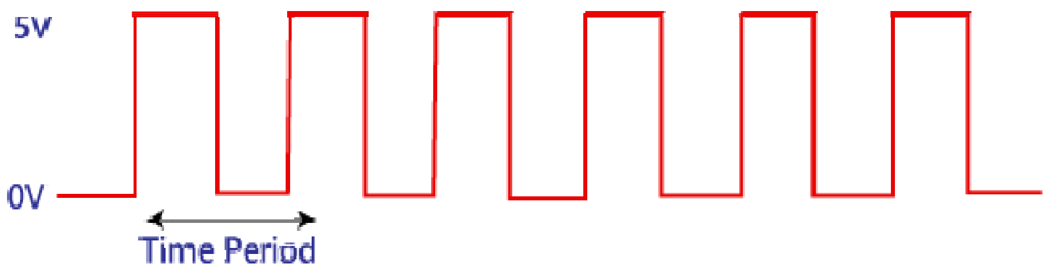
Differencebetweenthecombinationalcircuitsandsequentialcircuitsaregiven below:

	CombinationalCircuits	SequentialCircuits
1	The outputs of the combinational circuitdependonlyonthepresent inputs	Theoutputsofthesequentialcircuits dependonboth presentinputsand present state(previous output).
2	Thefeedbackpathisnotpresentinthe combinationalcircuit.	Thefeedbackpath ispresent in the sequentialcircuits.
3	In combinational circuits, memory elements are not required.	In the sequential circuit, memory elementsplayanimportantroleand require.
4	Theclocksignalisnotrequiredfor combinationalcircuits.	Theclocksignalisrequiredfor sequential circuits.
5	Thecombinationalcircuitissimpleto design.	Itisnotsimpletodesignasequential circuit.

State the necessity of clock and give the concept of level clocking andedge triggering,

1.Clock:

A clock signalis a periodic signalin whichON time and OFF time neednotbe the same.WhenONtimeandOFFtimeoftheclocksignal arethesame,a squarewaveisused torepresenttheclocksignal.Belowisa diagramwhich representstheclocksignal:



Aclocksignalisconsideredasthesquarewave.Sometimes,thesignal stays at logic, either high 5V or low 0V, to an equal amount of time. It repeats with a certain time period, which will be equal to twice the 'ON time' or 'OFF time'.

TypesofTriggering

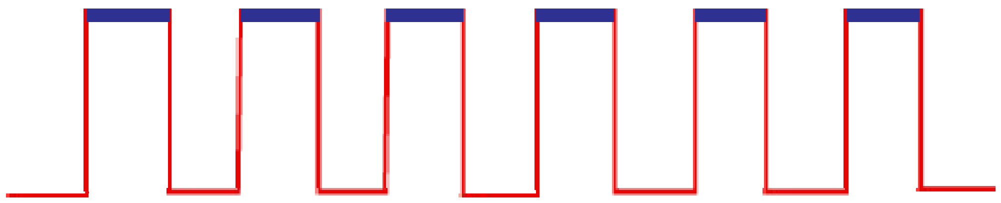
Thesearetwotypesoftriggeringinsequentialcircuits:

Level triggering

ThelogicHighandlogicLo w arethetwolevelsintheclocksignal.Inleveltriggering, whentheclockpulseisata particularlevel,onlythenthecircuitisacti ated.Thereare thefollowingtypesofleveltriggering:

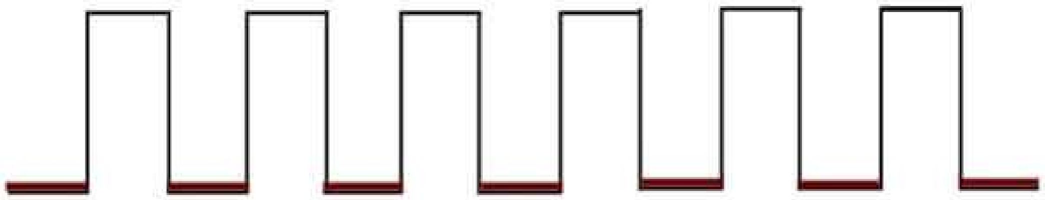
Positivelevel triggering

In a positive level triggering, the signal with Logic High occurs. So, in this triggering, the circuit is operated with such type of clock signal. Below is the diagram of positive level triggering:



Negativelevel triggering

Innegativeleveltriggering,thesignalwithLogicLowoccurs.So,inthitriggering,the circuit isoperatedwithsuchtypeofclock signal.BelowisthediagramofNegativelevel triggering:



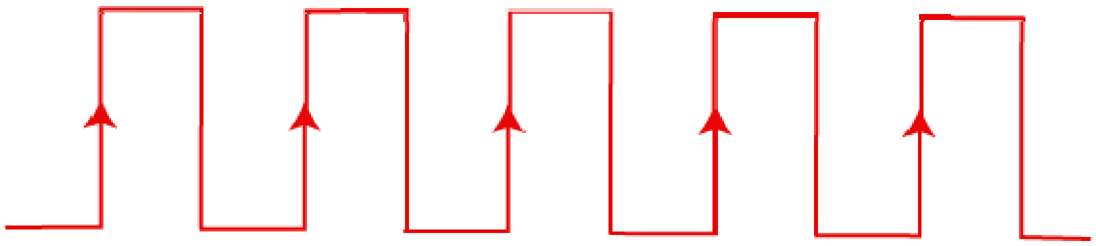
Edgetriggering

Inclocksignalofedgetriggering,twotypesoftransitions occur,i.e.,transitioneither from Logic Low to Logic High or Logic High to Logic Low.

Basedonthetransitionsof theclocksignal,therearethefollowing typesofedge triggering:

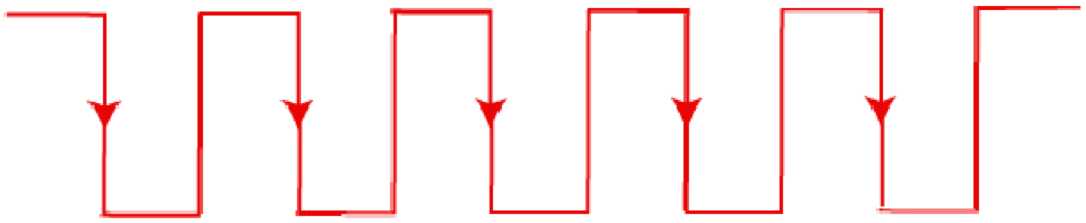
Positiveedge triggering

The transition from Logic Low to Logic High occurs in the clock signal of positive edge triggering.So,inpositiveedgetriggering,thecircuitisoperatedwithsuchtypeofclock signal.Thediagramofpositieedgetriggeringisgivenbelow.



Negativeedge triggering

ThetransitionfromLogicHightoLogiclowoccursintheclocksignalofnegativeedge triggering.So,innegativeedetriggering,thecircuitis operatedwithsuchtypeofclock signal.Thediagramofnegativeedgetriggeringisgivenbelow.

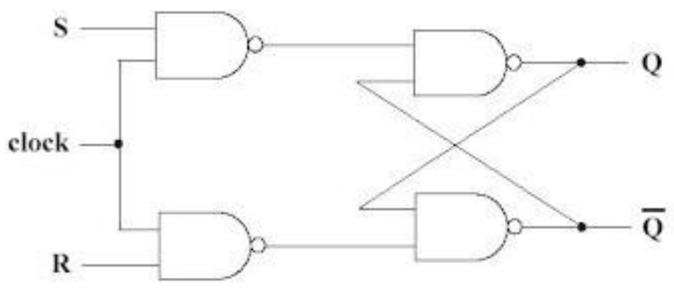


Whatisflipflop?

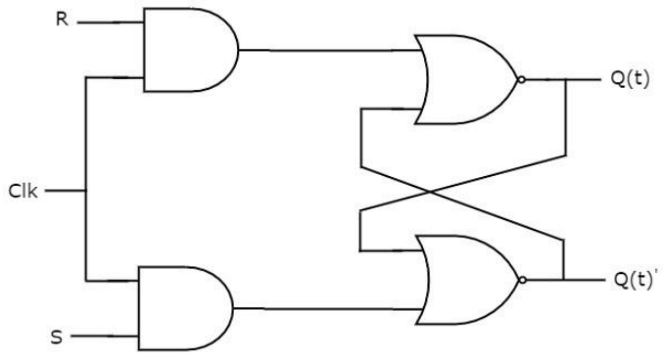
Flip-Flop is popularly known as the basic digital memory circuit. It is an edge triggered synchronoussequential logic circuit that is capable of storing single bit binary information.It has two states as logic 1(High) and logic 0(low) states. A flip flop is a sequential circuit which consists of a single binary state of information or data. The digital circuit is a flip flop which has two outputs and are of opposite states. It is also known as a Bistable Multivibrator.

ClockedSRflipflop

SR(Set-Reset)flip-flopisaclockedsequentialcircuitwhichiscontrolledbyedge triggered CLK control signal.



LogicdiagramusingNANDgate



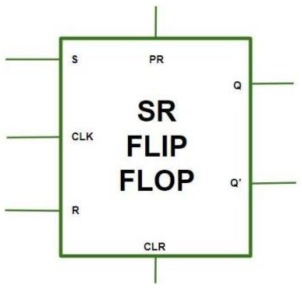
LogicdiagramusingANDandNORgate Truth

Table

Inputs			Outputs		States
CLK	S	R	Q	⍰	
0	0	0	NC	NC	No.change
0	0	1	NC	NC	No.change
0	1	0	NC	NC	No.change
0	1	1	NC	NC	No.change
1	0	0	NC	NC	No.change
1	0	1	0	1	Reset
1	1	0	1	0	Set
1	1	1	X	X	No.change

ClockedSRflipflopwithpresetandclearinputs.

In SR flip flop, with the help of Preset and Clear, when the power is switched ON,thestateofthecircuitkeepsonchanging,i.e.itisuncertain.Itmay cometo Set(Q= 1) or Reset (Q′ = 0) state. In many applications, it is desired to initially Set or Reset the flip flop. This thing is accomplished by the Preset (PR) and the Clear (CLR).



BLOCKDIAGRAMOFF/F

OperationsinSRFlip-Flop–

- **Case-1:**
  PR=CLR=1

  Theasynchronousinputsareinactiveandtheflipfloprespondsfreelytothe S,R and the CLK inputs in the normal way.

- **Case-2:**
  PR=0andCLR=1

  ThisisusedwhentheQissetto1.

- **Case-3:**
  PR=1andCLR=0

  ThisisusedwhentheQ′issetto1.

- **Case-4:**
  PR=CLR=0

  Thisisaninvalidstate.

INPUTS					OUTPUTS		Comments
PR	CLR	CLK	S	R	$Q_{(n+1)}$	$\overline{⍰}_{(n+1)}$	
0	1	NA	NA	NA	1	0	Set
1	0	NA	NA	NA	0	1	Re-set
1	1	0	NA	NA	Q_n	$\overline{⍰}_{n}$	No.change

1	1	1	0	0	Q_n	$\overline{Q}_n$	No.change
1	1	1	1	0	1	0	Set
1	1	1	0	1	0	1	Re-set
1	1	1	1	1	x	x	Notallowed

ApplicationsofFlip-Flop:

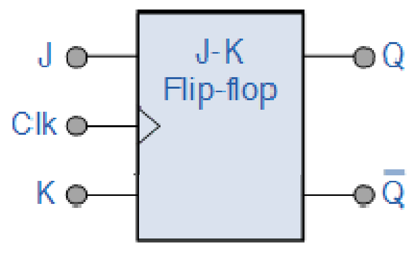
1. Flipflopsareusedasabounceeliminationswitch.
2. Theyareusedasaserialtoparallelandparalleltoserialconversion.
3. Itisusedforcounters.
4. Itisusedforfrequencydividerandalsoasalatch.

3.5ConstructlevelclockedJKflipflopusingS-Rflip-flopandexplain with truth table

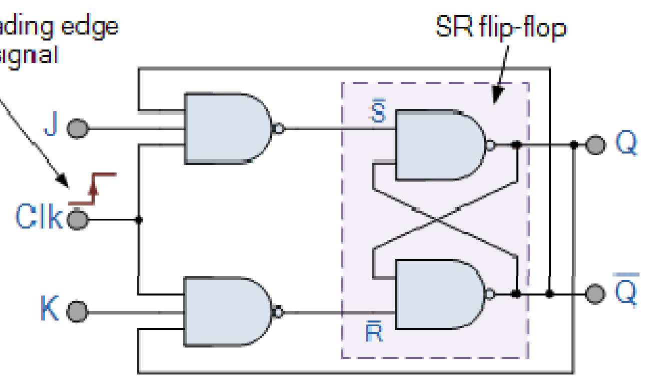
TheJK flip flopis one of the most used flip flops in digital circuits. The JK flip flop is a universal flip flop having two inputs 'J' and 'K'. In SR flip flop, the 'S' and 'R' are the shortened abbreviated letters for Set and Reset, but J and K are not. The J and K are themselves autonomous letters which are chosen to distinguish the flip flop designfrom other types. JK flip-flop can either be triggered upon the leading-edge of the clock or on its trailing edge and hence can either be positive- or negative- edge-triggered, respectively.

The JK flip flop work in the same way as the SR flip flop work. The JK flip flop has 'J' and 'K' flip flop instead of 'S' and 'R'. The only difference between JK flip flop and SR flip flop is that when both inputs of SR flip flop is set to 1, the circuit produces the invalid states as outputs, but in case of JK flip flop, there are no invalid states even if both 'J' and 'K' flip flops are set to 1.

The JK Flip Flop is a gated SR flip-flop having the addition of a clock input circuitry. The invalid or illegal output condition occurs when both of the inputs are set to 1 and are prevented by the addition of a clock input circuit. So, the JK flip-flop has four possible input combinations, i.e., 1, 0, "no change" and "toggle".



Symbol



Circuit

TRUTHTABLE

INPUTS		OUTPUT	STATES
J	K	Q^+	
0	0	Q	Previousstate
0	1	0	Re-set
1	0	1	Set
1	1	$\bar{Q}$	Toggles(Complementofpresent state)

[Typethedocumenttitle]

[Typethedocumentsubtitle]

leenamarndi

Microprocessor

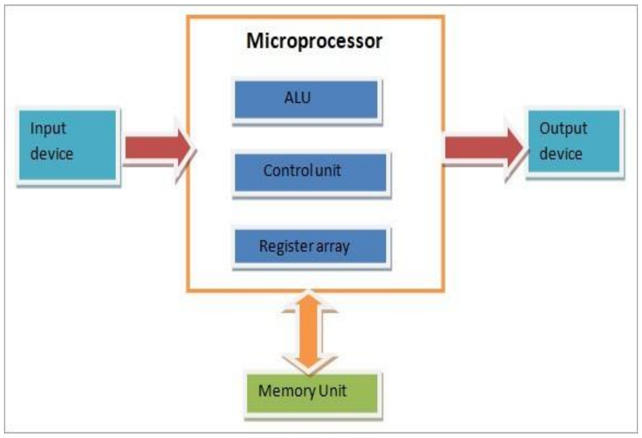
A microprocessor is a multipurpose, programmable, clock driven register based semiconductor device that read binary instructions from memory, accept binary data as input and process data according to instruction and provide result as output

Itisakindofintegratedcircuit(IC)unitwhichcombinesallthebasicfunctionsofa central processing unit (CPU) of the computer.

Itisaprogrammable unitthatisfabricatedonthe siliconchipand itconsistsofan ALU unit, clock, and control unit and register array which accepts the input in binary form (0’s and 1’s) and delivers the output after processing the input data as per the instructions fetched into the memory unit

Microcomputer

Adigitalcomputerinwhichonemicroprocessorhasbeenprovidedtoactasa CPUiscalled microcomputer



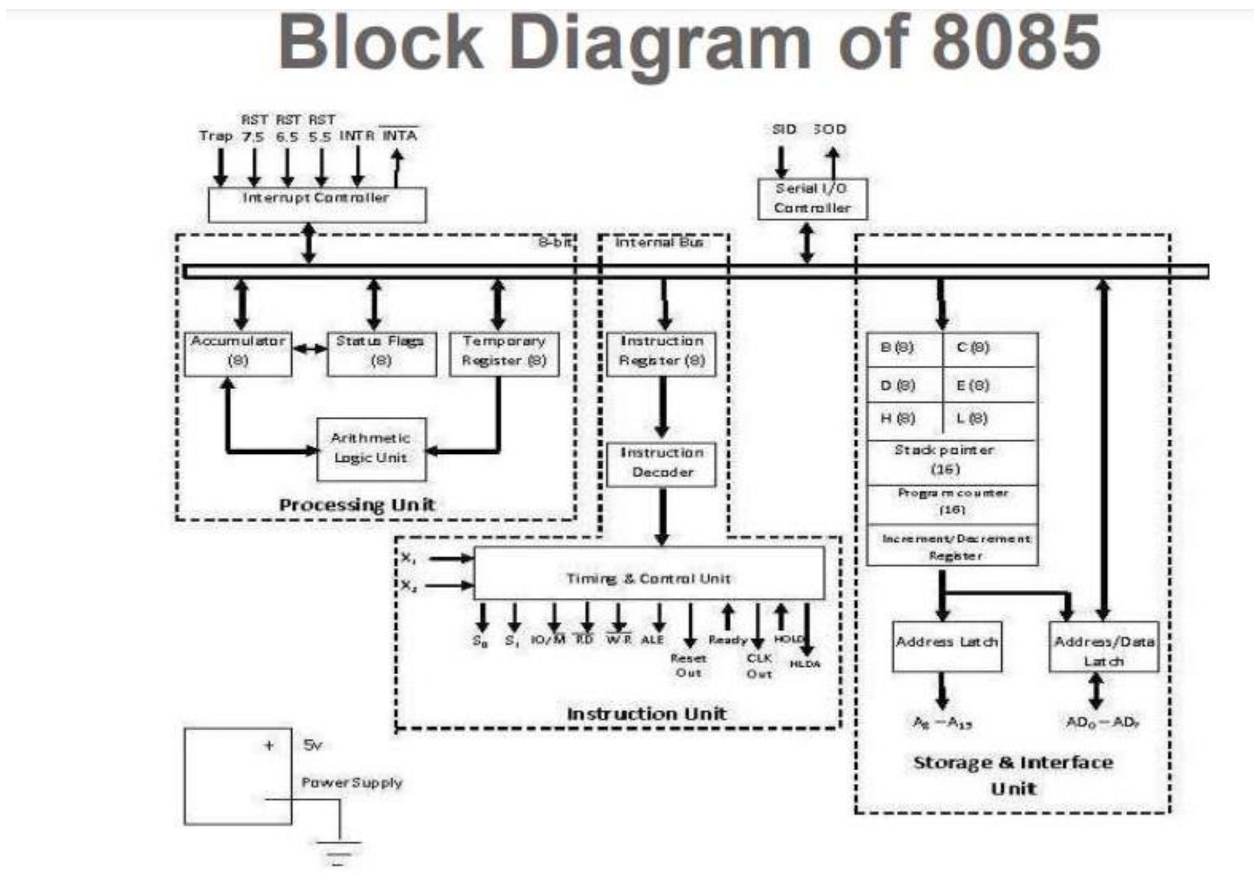
The basic building blocks of this processor are an ALU, register array, and the main control processing unit. The function of the arithmetic logical unit (ALU) is to perform the mathematical and logical operations based on the data fetched from the input units or the memory device.

Someimportant terms

- **Bit:**Adigitofthebinarynumberof code iscalledabit
- **Nibble**:The4bit(digit)binarynumberorcodeis calleda nibble
- **Byte:**8 bitbinaryno.iscalledByte
- **Word:**16bitbinaryno.iscalled byte

ArchitectureofIntel8085A Microprocesorsand descriptionofeachblock.

- Itisa40pinI.C.packagefabricatedonasingleLSI chip.
- TheIntel8085usesasingle +5Vd.c.supplyforits operation.
- Intel8085is clockspeedisabout 3 MHz; theclockcycleisof 320ns.
- 8bit databus.
- Address busisof16-bit,whichcanaddressupto64KB
- Ithas80basicinstructionsand246opcodes.



Itconsistsof3(Three)mainsectiontheseareasfollows

1. **Arithmetic&LogicUnit**
2. **TimingandControlunit**
3. **SetsofRegister**

1. Arithmetic&LogicUnit

The arithmetic and logic unit performs the following arithmetic and logic operation

i) Addition

ii) Subtraction

iii) LogicalAND

iv) LogicalOR

v) LogicalExclusiveor

vi) Increment

vii) Decrement

2. TimingandControlunit

Thetiming andcontrol unitcomesunderthesectionofCPU,anditgeneratesthe timing and control signals which are necessary for the execution of Instructions. It controls flow of data from CPU to other devices.It provides status, control and timing signals which are required for the operation ofmemory and I/O device .It isalso used to control the operations performed by the microprocessor and the devices connected to it. There are certain timing and control signals like: Control signals, DMA Signals, RESET signals, Status Signal.

3. SetsofRegister

Registersare used for temporary storage and manipulation of data and instructions by the microprocessor. Data remain in the registers till they are sent to the I/O devices or memory. Intel 8085 microprocessor has the following registers:

a) One8-bitaccumulator(ACC)i.e.registerA
b) Sixgeneralpurposeregistersof8-bit,theseareB,C,D,E,HandL
c) One16-bitstackpointer,SP
d) One16-bitProgramCounter,PC
e) Instructionregister
f) Temporaryregister

In addition to the above mentioned registers the 8085 microprocessor contains a set of five flip-flops which serve as flags (or status flags).

Aflagisaflip-flopwhichindicatessomeconditionswhicharisesafterthe execution of an arithmetic or logical instruction.

a) Accumulator(ACC):

The accumulator is an 8-bit register associated with the ALU. The register 'A' is an accumulator in the 8085. It is used to hold one of the operands of an arithmetic and logical operation. The final result of an arithmetic or logical operation is also placed in the accumulator.

b) General-PurposeRegisters:

The 8085 microprocessor contains six 8-bit general purpose registers.They are: B, D, C, E, H and L register.

Toholddataof16-bitacombinationoftwo8-bitregisterscanbeemployed. The

combination of two 8-bit registers is called register pair.

The valid register pairs in the 8085 are: D-E, B-C and H-L. The H-L pair is used to actas a memory pointer.

c) StackPointer(SP):

It is a 16-bit special function register used as memory pointer. A stack is nothing but a portion of RAM i.e. it is sequence of memory location set aside by a programmer to store/ retrieve the content of accumulator, flags, program counter and general-purpose register during the execution of a program.

StackworkonLIFO(lastinfirstout)Principle

Itsoperationisfastercomparednormalstore/retrieveofmemorylocation

Thestackpointer(SP)controlstheaddressingofthestack.TheStackPointercontainsthe address of the top element of data stored in the stack.

d) ProgramCounter(PC):

Itis a16-bit special purposeregister.It isusedtoholdtheaddressof memory of the next instruction to be executed. It keeps the track of the instruction in a program while they are being executed. The microprocessor increments the content of the next program counter during the execution of an instruction so that at the end of theexecution of an instruction it pointsto the next instructions address in the program.

e) Instructionregister

The instruction register holds the opcode (operation code or instruction code) of the instruction which is being decoded and executed.

f) Temporaryregister

It is an 8-bit register associated with the ALU. It holds data during an arithmetic/logical operation. It is used by the microprocessor. It is not accessible to programmer.

g) Flags:

The Intel 8085 microprocessor contains five flip-flops to serve as a status flags. The flip-flops are reset or set according to the conditions which arise during an arithmetic or logical operation.

a. CarryFlag(CS)
b. ParityFlag(P)
c. AuxiliaryCarryFlag(AC)
d. ZeroFlag(Z)
e. SignFlag(S)

$\square_7$	$\square_6$	$\square_5$	$\square_4$	$\square_3$	$\square_2$	$\square_1$	$\square_0$
S	Z	X	AC	X	P	X	CS

a) CarryFlag(CS)

Carry is generated when performing n bit operations and the result is more than n bits, thenthisflagbecomesseti.e.1,otherwiseitbecomesreseti.e.0. Duringsubtraction(A-B),ifA>Bitbecomesresetandif(A<B)itbecomesset. Carry flag is also called borrow flag.

1-carryoutfromMSBbitonadditionorborrowintoMSBbitonsubtraction0-no carry out or borrow into MSB bit

Example:

MVIA30(load30HinregisterA)
MVIB40 (load40HinregisterB)
SUB B (A = A – B)
Thesesetofinstructionswillsetthe carry flagto1as30 –40generatesacarry/borrow.

MVIA40(load40HinregisterA)
MVIB30 (load30HinregisterB)
SUB B (A = A – B)
Thesesetofinstructionswillresetthesignflagto0as40–30doesnotgenerateany carry/borrow.

b) ParityFlag(P)

Ifafteranyarithmeticorlogicaloperationtheresulthasevenparity,anevennumberof 1 bits, the parity register becomes set i.e. 1, otherwise it becomes reset i.e. 0.

1-accumulatorhasevennumberof1bits 0-
accumulator has odd parity

Example:

MVIA05(load05HinregisterA)
ThisinstructionwillsettheparityflagtoastheBCDcodeof05His00000101,which contains even number of ones i.e. 2.

c) AuxiliaryCarryFlag (AC)

This flag is used in BCD number system(0-9). If after any arithmetic or logical operationD(3)generatesanycarryandpassesontoB(4)thisflagbecomesset i.e.1,otherwiseitbecomesreseti.e.0.Thisistheonlyflagregisterwhichis notaccessiblebytheprogrammer1-carryout frombit3onadditionorborrow into bit 3 on subtraction 0-otherwise

Example:

MOVA2B(load2BHinregisterA)
MOV B 39 (load 39Hinregister B)
ADD B (A = A + B)

Thesesetofinstructionswillsettheauxiliarycarryflagto1,asonadding2Band39, addition of lower order nibbles B and 9 will generate a carry.

d) Zero Flag(Z)

Afteranyarithmeticalorlogicaloperationiftheresultis0(00)H,thezeroflagbecomes set i.e. 1, otherwise it becomes reset i.e. 0.
00Hzeroflag is1.
from 01HtoFFHzeroflagis 0

1-zeroresult
0-non-zero result

Example:

MVIA10(load10HinregisterA) SUB A
(A = A – A)
Thesesetofinstructionswillsetthezero flagto1as10H–10His00H

e) SignFlag(S)

AfteranyoperationiftheMSB(B(7))oftheresultis1,itindicatesthenumberisnegative and the sign flag becomes set, i.e. 1. If the MSB is 0, it indicates the number is positive and the sign flag becomes reset i.e. 0.
from00Hto7F,signflagis0
from80Hto FF,signflagis1

1- MSB is 1(negative)
0- MSB is 0(positive)

Example:

MVIA30(load30HinregisterA)
MVIB40 (load40HinregisterB)
SUB B (A = A – B)
Thesesetofinstructionswillsetthesignflagto1as30–40isanegativenumber.

MVIA40(load40HinregisterA)
MVIB30 (load30HinregisterB)
SUB B (A = A – B)
Thesesetof instructionswillresetthesignflagto0as40 –30 is apositivenumber.

PinDiagram8085microprocessor

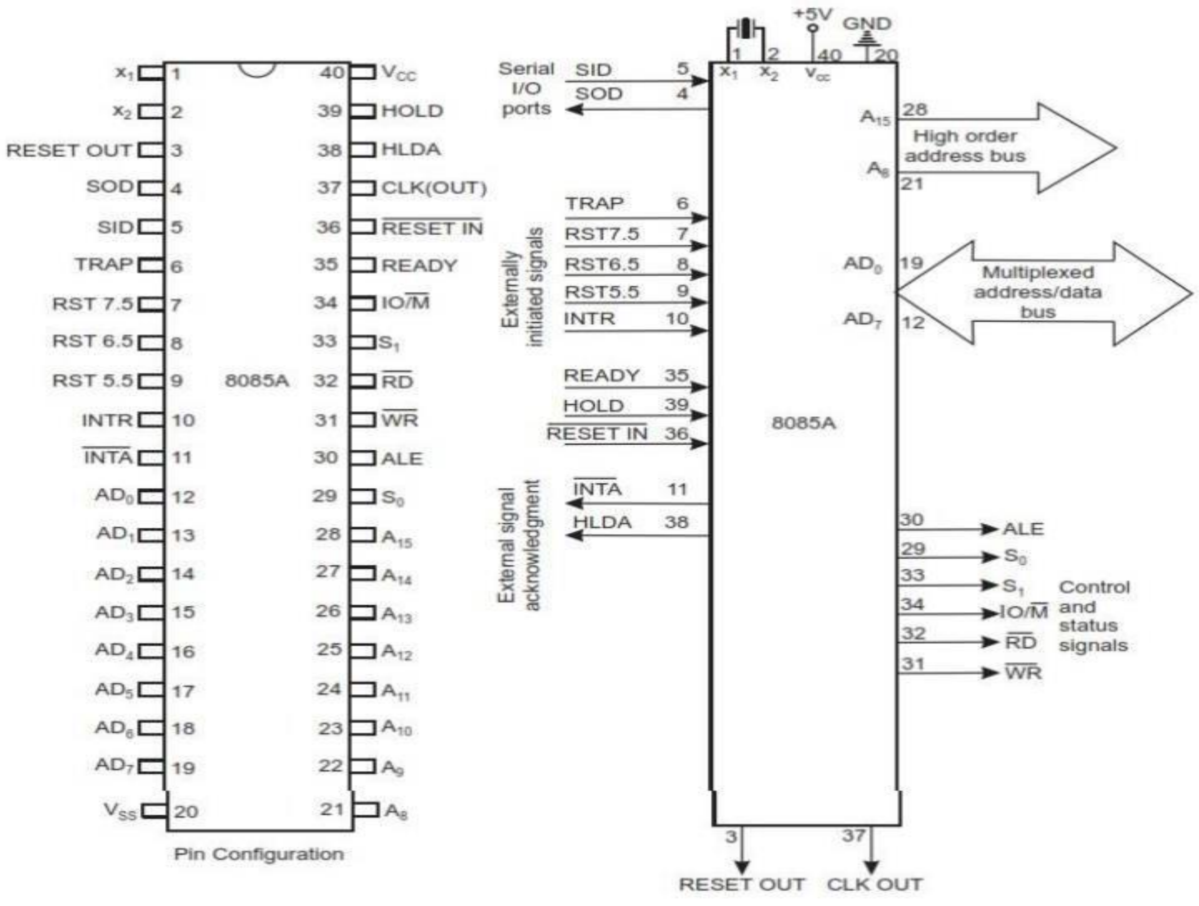


Fig 1.2 Pin Diagram of 8085

A8- A15(Output):

ThesearetheAddressBusandusedformost significant8bitsofthememoryaddress or the 8 bits of the I/0 address,

AD0- AD7(Input/Output)

Multiplexed Address/Data Bus it serve dual purpose. They are used forLeast significant 8 bits of the memory address (or I/0 address) during the first clock cycle of a machine cycle. Then itbecomes the data bus during the second and third clock cycles.

ALE (Output):

Address Latch Enable signal it goes high during the first clock cycle of a machine cycle and enables the lower 8 bit address to get latched either into the memory or external latch So when pulse goes high means ALE=1, it makes address bus enable and when ALE=0, means low pulse makes data bus enable.

IO/�(Output):

ItisastatussignalwhichdistinguisheswhetherI/Oormemoryoperationisbeing performed

Whenitgoeshigh,theaddressontheaddressbusisforanI/Odevice.

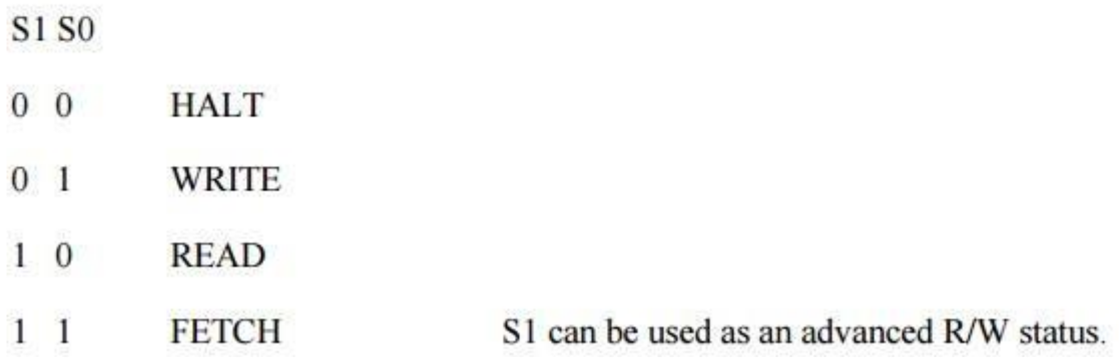
i.eIf$IO/\overline{M}$=1thenI/Ooperationisbeingperformed.

Whenitgoeslow,theaddressontheaddressbusisforanmemorylocation

i.eIf$IO/\overline{M}$=0then◦Memoryoperationisbeingperformed.

SO,S1(Output):

These are the status signals sent by the microprocessor to distinguish the various typeof operation



S1 S0		
0 0	HALT	
0 1	WRITE	
1 0	READ	
1 1	FETCH	S1 can be used as an advanced R/W status.

$\overline{RD}$(Output):

RDstandsforRead.

Itisanactivelowsignal.i.e$\overline{RD}$=0thenreadoperationisperform

It is a control signalsent by the microprocessor to the memory/input device to control READ operation.A low signal indicates that data on the data bus must be placed either from selected memory location or from input device.

$\overline{RD}$indicatesthe selected memory or input device is to be read and that the Data Bus is available for the data transfer.

$\overline{WR}$(Output):

WRstandsforwrite.

Itisanactivelowsignal.i.e$\overline{WR}$=0thenwriteoperationisperform

It is a control signal sent by microprocessor to the memory/ output device to control Writeoperation A low signal indicates that data on the data bus must be written into selected memory location or into output device.

$\overline{WR}$ indicates the data on the Data Bus is to be written into the selected memory or output device.

READY(Input):

Itisasignalsentbyaninputoroutputdevicetothemicroprocessor.

Itindicatesthattheinputoroutputdeviceisreadytosendorreceivedata.

ThemicroprocessorexaminesREADYsignalbeforeitperformsdatatransferoperation

If Ready is high, it indicates that the input or output device is ready tosend orreceivedata.

IfReadyislow,the microprocessorwillwaitfor Readyto gohighbeforecompletingthe read or write cycle.

HOLD(Input):

It indicates that another device is requesting the use of the address and data bus. Having received HOLD request the microprocessor relinquishes(give up) the use of the buses as soon as the current machine cycle is completed. Internal processing may continue. After the removal of the HOLD signal the processor regains the bus.

ExplainwithExample

The HOLD pin specifies when any device is demanding the employ of address as well as a data bus. The two devices are LCD as well as A/D converter. Assume that if A/D converter is employing the address bus as well as a data bus. When LCD desires the utilize of both the buses by providing HOLD signal, subsequently the microprocessor transmits the control signal toward the LCD after that the existing cycle will be ended. When the LCD procedure is over, then the control signal is transmitted reverse to A/D converter.

HLDA(Output):

This is the response signal of HOLD, and it specifies whether this signal is obtained or not obtained. After the implementation of HOLD demand, this signal will go low.

INTR(Input):

It is an Interrupt signal sent by an external device to the microprocessor, when it goeshighthemicroprocessorsuspendstheexecutionofitsnormalsequenceofinstructions i.eIfitisactive,theProgramCounter(PC)willbeinhibitedfromincrementingandan $\overline{\text{INTA}}$willbeissued.

$\overline{\text{□□□□}}$(Output):

Itisaninterruptacknowledgesignalissuedbythemicroprocessorafterreceivingan interrupt request from an external device. it is low active signal.

RST5.5, 6.5,7.5:

Thesepinsaretherestartmaskableinterruptsor VectoredInterrupts,usedtoinsert an inner restart function repeatedly. All these interrupts are maskable, they can be allowed or not allowed by using programs.

TRAP (Input):

Trap interrupt is a non maskable restart interrupt. It is recognized at the same time as INTR.Itisunaffectedbyanymaskorinterrupt Enable.Ithasthehighestpriorityof any interrupt.

RESET IN (Input):

Reset sets the Program Counter to zero and resets the Interrupt Enable and HLDA flipflops. None ofthe other flags or registers (except the instruction register) areaffected The CPU is held in the reset condition as long as Reset is applied.

RESETOUT(Output):

IndicatesCPUisbeingreset.CanbeusedasasystemRESET.

X1, X2 (Input):

Crystal or R/C network connections to set the internal clock generator X1 can also bean external clock input instead of a crystal. The input frequency is divided by 2 to give the internal operating frequency.

CLK(Output):

Clock Output for use as a system clock when a crystal or R/ C network is used as an input to the CPU. The period of CLK is twice the X1, X2 input period.

SID (Input):

Serial input data line the data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.

SOD (output):

Serialoutputdataline.TheoutputSODissetorresetasspecifiedbytheSIM instruction.

Vcc:

+5 voltsupply.

Vss:

GroundReference.

4.4.Stack,Stackpointer&stacktop

- StackisaportionofRAMmemorydefinedbytheuserfortemporarystorage and retrieve of data while executing a program.
- Themicroprocessorwillhavededicatedinternalregistercalledastackpointer toholdtheaddressofthestack
- Alsotheprocessorwillhavefacilitytoautomaticallydecrement/incrementthe content of SP after every Write/read into stack

- ForeverywriteoperationintothestacktheSPautomaticallydecrementedby two
- ForeveryreadoperationintothestacktheSPautomaticallyincrementedby two
- ThecontentsregisterareremovedtocertainmemorylocationbyPUSH operation, then the register are used for other operations
- Afterpushoperationthosecontentswhichweresavedinthememoryare transferredbacktotheregisterbyPOPoperation
- ThesetofmemorylocationkeptforthisoperationiscalledStack
- ThelastmemorylocationoftheoccupiedportionoftheStackiscalledStack top
- Aspecial16bitregisterisknownasstackpointerholdtheaddressofstacktop
- ThestackpointerisinitializedinbeginningoftheprogrambyLXISPorSPHL instruction
- DataarestoredinthestackonLast-in-first-out(LIFO)principle
- SPregisterholdtheaddressofstacktoplocation

PUSHOPERATION

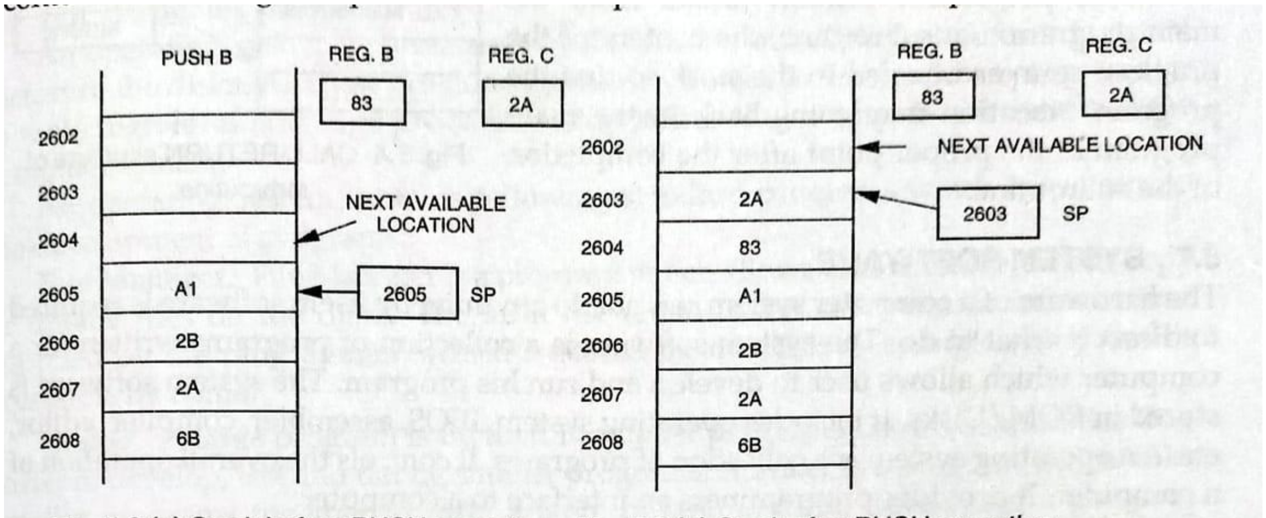


Fig. 5.2 (a) Stack before PUSH operation **Fig. 5.2 (b)** Stack after PUSH operation

POPOPERATION

POPoperationisusedtotransferthecontentsfromthestacktotheregister

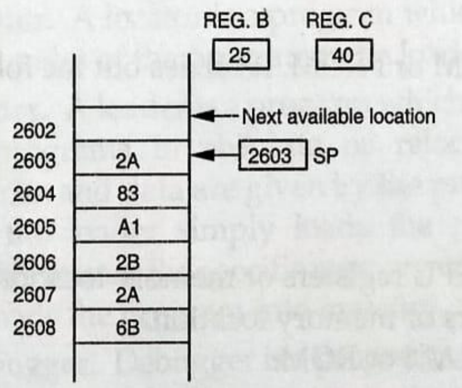


Fig. 5.3 (a) Stack before POP operation

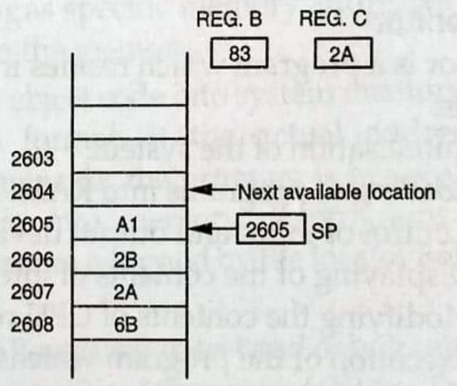


Fig. 5.3 (b) Stack after POP operation

Interruptsin8085 microprocessor:

When microprocessor receives any interrupt signal from peripheral(s) which are requesting its services, it stops its current execution and program control is transferred to a sub-routine by generating CALL signal and after executing sub-routine by generating RET signalagainprogramcontrolistransferredtomainprogramfrom where it had stopped.

When microprocessor receives interrupt signals, it sends an acknowledgement (INTA) to the peripheral which is requesting for its service.

Interrupts can be classified into various categories based on different parameters:

1. **HardwareandSoftwareInterrupts–**

When microprocessors receive interrupt signals through pins (hardware) of microprocessor, they are known as Hardware Interrupts. There are 5 Hardware Interrupts in 8085 microprocessor.

Theyare–INTR,RST7.5,RST6.5,RST5.5,TRAP

Software Interrupts are program instruction those which are inserted in between the program which means these are mnemonics of microprocessor. There are 8 software interrupts in 8085 microprocessor.

They are–RST0,RST1,RST2,RST3,RST4,RST5,RST6,RST7.

2. **VectoredandNon-VectoredInterrupts–**

VectoredInterrupts arethosewhichhavefixedvectoraddress(startingaddressof sub-routine) and after executing these, program control is transferred to that address.

Vector Addresses are calculated by the formula

Vector Addresses=Interrupt No.*8

INTERRUPT	VECTORADDRESS
TRAP (RST 4.5)	24H
RST5.5	2CH
RST6.5	34H
RST7.5	3CH

ForSoftwareinterruptsvectoraddressesaregivenby:

INTERRUPT	VECTORADDRESS
RST0	00H
RST1	08H
RST2	10H
RST3	18H
RST4	20H
RST5	28H
RST6	30H
RST7	38H

Non-Vectored Interrupts are those in which vector address is not predefined. The interrupting device gives the address of sub-routine for these interrupts. INTR is the only non-vectored interrupt in 8085 microprocessor.

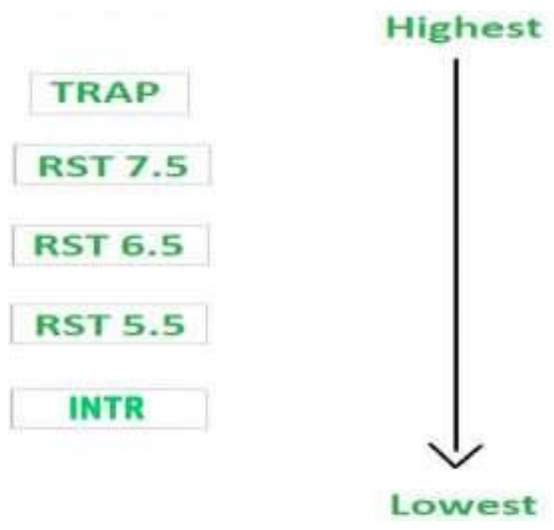
3. <u>MaskableandNon-MaskableInterrupts</u>–

Maskable Interrupts are those which can be disabled or ignored by the microprocessor. These interrupts are either edge-triggered or level-triggered, so they can be disabled.
INTR, RST 7.5, RST 6.5, RST 5.5 are maskable interrupts in 8085 microprocessor

Non-Maskable Interrupts are those which cannot be disabled or ignored by microprocessor.
TRAP ***is a non-maskable interrupt. It consists of both level as well as edge triggering and is used in critical power failure conditions***.

<u>PriorityofInterrupts</u>–

Whenmicroprocessorreceivesmultipleinterruptrequestssimultaneously,itwill executetheinterrupt servicerequest(ISR)accordingtothe priorityofthe interrupts.



<u>InstructionforInterrupts–</u>

5. **EnableInterrupt(EI)**
6. **DisableInterrupt(DI)**
7. **SetInterruptMask(SIM)–**Itisusedtoimplementthehar dwareinterrupts (RST7.5,RST 6.5,RST5.5)bysettingvariousbitsto formmasksor generateoutputdataviatheSerialOutputData(SOD)line
8. **ReadInterrupt Mask(RIM)–**Thisinstructionisusedtoreadthestatusof thehardwareinterrupts(RST7.5,RST6.5,RST5.5)bylodingintotheA registerabyte whichdefinestheconditionofthema skbitsforthe interrupts.Italsoreadstheconditionof SID(SerialInput Data)bitonthe microprocessor.

<u>4.6Opcode&Operand,</u>

Whatis Opcode?

Opcodesmean“operationcodes”.Anopcodeisthefirstpartofaninstruction which specifies the task to be performed by the computer is called opcode.

Itisaninstructionthattellstheprocessorwhattodowiththevariableordatawritten beside it.

What isOperand?

An operand is the second part of the instruction,is the data to be operated on and it is called **operand** .

InstructionWordSize

The 8085 instruction set is classified into the following three groups according to word size:

1. **One-wordor1-byte instructions**
2. **Two-wordor2-byte instructions**
3. **Three-wordor3-byteinstructions**

One-ByteInstructions

In 1-byte instruction, the opcode and the operand of an instruction are represented in one byte.

Operand(s) are internal registers and are in the instruction in form of codes. If there is no numeral present in the instruction then that instruction will be ofone-byte.

InstructionarerequiredoneMemorylocationtostoreonebyteinthe memory

Example,MOVC,A,RAL,andADDB, etc.

Two-wordor2-byteinstructions

Two-byte instruction is the type ofinstruction in which the first 8 bits indicates the opcode and the next 8 bits indicates the operand.

In a two-byte instruction, the first byte specifies the operation code and secondbyte specifies the operand.

Source operand is a data byte and immediately following the opcode. If an 8-bit numeral is present in theinstruction then that instruction will be of two-byte. Here, the numeral may be a data or an address.

Instructionarerequiredtwo Memorylocationtostoreinthe memory

Forexample,MVIA,35H andIN 29H,etc.

In a two-byte instruction, thefirst bytewill be the opcodeand thesecond byte will be for the numeral present in the instruction.

Three-wordor3-byteinstructions

Three-byte instruction is the type of instruction in which the first 8 bits indicates the opcode and the next two bytes specify the 16-bit address. The low-order address is represented in second byte and the high-order address is represented in the third byte.

In a three-byte instruction, the first byte specifies the opcode, and the following two bytes specify the 16-bit operand.

InstructionarerequiredthreeMemorylocationto storeinthememory

Example,LXIH,3500HandSTA2500H,etc

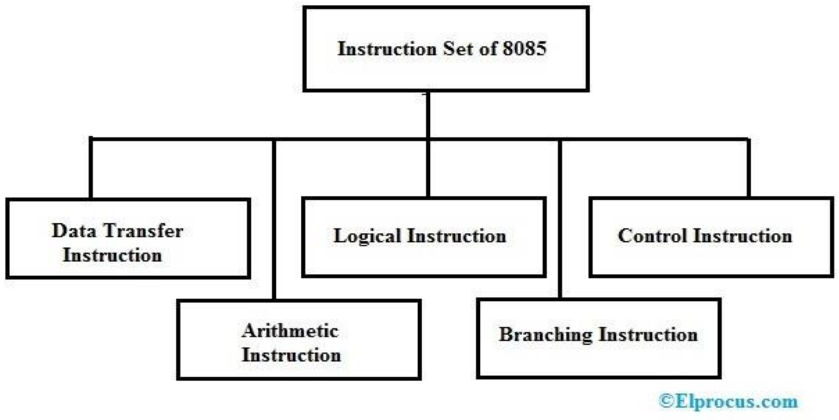
Instructionset of8085

Aninstructionisabinarypatterndesignedinsideamicroprocessortoperforma specific function.

In microprocessor, the**instruction**set is the collection of the instructions that themicroprocessor is designed to execute.

ClassificationofInstructionSetof8085

Theinstructionsetof8085microprocessorisclassifiedintofivetypeswhichinclude the following.



DataTransfer Instruction

An instruction that is used to transfer the data from one register to another isknown as data transfer instruction. So, the data transfer can be done fromsource to destination without changing the source contents.

Data transfer mainly occurs from one register to another register, from memory locationto register, register to memory,and between an I/Odevice &accumulator.

MOVM, Data
This type of instruction specifies the data transfer immediately to a location of memory. This memory location address can be specified at the H-L registers.

Example:MOVM,28H

MVIr,Data (MoveImmediate)
In this type of instruction, the transmission of data can be done immediatelytoward the particular register.

Example:MVIr,32H

LDAaddress (Load Accumulator)
LDA is a load accumulator instruction that is mainly used for copying the data available in the address of memory indicated as the instruction's operand to the accumulator. Particularly, in this case, the available data in the 16-bit address memory is transferred toward the accumulator.

Example:LDA500H

LDAX(LoadAccumulatorbyextendedRegister Pair)

It is a load accumulator from an address in the register pair. In this type of data transfer instruction, the register holds the address of the data that needs to be loaded to the accumulator.

Example:LDAXC/D

LHLD(Load H&LRegistersDirect)
LHLD instruction is a direct load instruction, where it loads the H-L register with the data fromthe memory. In this type of instruction, the data which is availablein the address specified is copied to the L register first and then the available data within the next memory location will be loaded in the H register.

Example:LHLD2500H

STAAddress(StoreAccumulatorContentsinMemory)
STA stands for stored accumulator direct instruction. Once this instruction is accepted, then the available data within the accumulator can be transferred to the address of memory indicated within the operand.

Example:STA2030H

In the above example data stored in the accumulator will be stored tomemory location 2030. LSB followed by MSB will be stored in the memory location.

STAXRegister(StoreAccumulatorbyExtendedRegister)
It is a stored accumulator indirect instruction. In this instruction, the register is available as the operand that holds a memory address. Thus, the accumulator data can be copied to that specific memory location.

Example:STAX D

XCHG(Exchange)
This type of data transfer instruction can be used to exchange the data available within two registers.

Example: XCHGH-L &D-E.Inthis,thecontentsofH&DandL&Eare exchanged.

SPHL (StackPointerHL Register)
In this data transfer instruction, the data of H &L can be moved to the stackpointer.

PCHL(ProgramCounterwithHL Data)

Similar to SPHL instruction, this PCHL instruction simply copies the H-L register’sdataintotheSPbyloadingthehigh orderbytesatH&loworderbytesat L.

PUSH
In this type of instruction, the stack can be loaded with the available data withinthe register provided in the operand. Initially, the stack pointer gets decreased & high order bytes are copied to the stack. Further stack pointer gets decreased to load the low order register bytes.

Example:PUSHD

POP
This instruction indicates the data transfer from the top of the stack to the register provided as the operand.

Example:POPC

OUT
In this typeofdatatransfer instruction, thedataavailableattheaccumulatorcanbe copied toward the I/O port. An 8-bit port address at the operand is present.

Example:OUT36H

IN
This type of instruction is used to load the data available at the I/O port to the accumulator. The operand simply holds the port address from where the data canbe copied.

Example:IN,6BH

ArithmeticInstructionof8085

The arithmetic instructions perform different operations like addition, subtraction, increment & decrement on the data within memory & register in the 8085 microprocessor.

ADDr
This arithmetic instruction adds the data which is available in the register to the data available within the accumulator & the final result will be stored in the accumulator.

Example:ADDC

ADDM

This typeofinstructionis mainlyusedtoaddthedateinthememoryaddress data denoted at the operand to the data available at the accumulator. So the addition result will be stored within the accumulator.

Example:ADD28H

ADIData (Add Immediate)
In this instruction, the 8-bit data is specified as an operand is added immediatelyto the data available at the accumulator & the result is stored at the accumulator.

Example:ADI24H

ACIData(AddwithCarry Immediate)
This type of instruction simply adds the 8-bit data available at the operand & carries the flag by the data available at the accumulator. After every addition, the flag reproduces the output of the addition.

Example:ACI35H

ADCr (Add with Carry)
In this type of instruction, the data present at the register can be added to the data available at the accumulator with the carry bit & output is simply reflected at the accumulator.

Example:ADCD

AMCM
This type of instruction is mainly used to add the available data at the location of memory whose address is denoted within the operand specified & the carry bitwith the data available within the accumulator. So the output of addition can be stored within the accumulator.

Example:AMC25H

SUBr
This type of instruction is used to subtract theavailable data at theregister given at theoperandfromthedatapresentin theaccumulator.Thefinalresultwillbestored at the accumulator.

**Example:SUBC SUB
M**

This instruction is used to subtract the available data at the location of memory whose address is provided by the H-L register from the data present at the accumulator.

Example:SUB128H

SUIData(SubtractImmediatefromAccumulator)
This type of instruction is mainly used to instantly subtract the data available as operand within the instruction from the available data at the accumulator. After every subtraction, the flag can be changed to show the result of subtraction.

Example:SUI35H

SBIData(SubtractwithBorrowImmediatefromAccumulator)
This type of instruction helps subtract the 8-bit data provided as the operand & the borrow bit from the available data at the accumulator, and the result will be stored within the accumulator.

Example:SBI24H

SBB r
This instruction is used to subtract the data present at the register & the borrow bit from the data present at the accumulator.

Example:SBBC

SBB M(SubtractionwithBorrow)
This instruction is used to specify the subtraction of data available at the memory location, whose address is available at the H-L register & the borrow bit from the data present at the accumulator.

Example:SBB1000H

INXr (IncrementExtendedRegister)
This type of instruction is used to increase the data by 1 which is availableat theregister provided at the operand. The result will be stored at the same register.

Example:INXC

DCXr(DecrementExtendedRegister)
This typeofinstructiondecreasesthedataavailable at theregisterby1 &theresult will be stored in the same register.

Example:DCXC

<u>DCRM(DecrementRegister)</u>
In aninstruction,sometimes the operand holds a location of memory. The memory location address is available at the H-Lpair. Thus the data available at that specific location will be decreased by 1.

Example:DCR28H

DAA(DecimalAdjust Accumulator)
DAA is a decimal adjust accumulator, used to break the binary number from 8-bit to two 4-bit binary-coded decimal numbers.

LogicalInstruction

Logicalinstructionsaremainlyusedtoperformdifferentoperationslikelogicalor Boolean over the data available in either memory or register. These instructions will modify the flag bits based on the operation executed.

<u>CMPR/M(ComparetheRegister/MemorywiththeAccumulator</u>)
This instruction is used to compare the data at the accumulator with the data present at the register or memory which is given as operand. According to the result obtained by the comparison, the flags are set. While the data that iscompared remains unchanged.

Example:CMPB

CPI Data(CompareimmediatethroughtheAccumulator)
This type of instruction compares the 8-bit data provided as operand within the instruction by the data available within the accumulator. This result is shown through the flags.

Example:CPI50

<u>ANAR/M(LogicalANDregisteror memorywiththeaccumulator)</u>
This instruction executes the AND operation of the data available within the accumulator to the data available in the memory or register. After the operation of AND, S, P, Z will be changed to show the outcome of the comparison.

Example:ANAC

ANIdata (AndImmediatewithAccumulator)

This instruction executes AND operation for the immediate 8-bit data provided as operand by the data available in the accumulator.

Example:ANI35H

ORAR/M(ORAccumulatorRegisterorMemory)
Thisinstructionisusedtoperform OR operationofthedataavailablewithinthe accumulator by the data available in the memory location or register.

Example:ORAC

ORIdata(ORImmediateData)
The 8-bit data provided as an operand is ORed logically with the data within the accumulator.So,theoutputofthis instruction canbesavedwithinthe accumulator.

Example:ORI36H

XRAR/M(ExclusiveOR Immediatewith Accumulator)
ThisinstructionisusedtoexecuteXORoperationthroughdataavailableatthe accumulator & the data present at the memory or register.

Example:XRA2030

XRIdata (ExclusiveOR Accumulator)
This type of instruction is used to execute the XOR operation of the 8-bit data specified as operand & the data present at the accumulator. The output will be stored at the accumulator.

Example:XRI30

RLC(RotateLeftAccumulator)
This instruction holds significance when there exists a need to rotate the bits present in the accumulator. Basically, for an 8-bit value, each bit is rotated or shifted left by one position. Also, the rotation of the last bit of the sequence i.e., D7, sets the CY flag.

RRC(RightRotateAccumulator)
This instruction is used to rotate the bit toward the right with one position. So, in this case, D0 sets the CY flag.

Example:RRC

RAL(RotateAccumulatorLeft)
This typeofinstructionisusedto rotatethebitstowardtheleftwithoneofthedata available within the accumulator through the carry flag. Here, D7 can be shifted to hold the flag & the bit within the carry flag can be shifted to D0.

Example:RAL

RAR(RotateAccumulatorRight)
This type of instruction is mainly used to rotate the data bits to the right which are available within the accumulator by the carry flag. Here, D0 can be shifted to hold the flag & the carry bit can be moved to the D7 position.

Example:RAR

STC(SettheCarryFlag)
This type of instruction is used to set the carry flag (CF) to 1 by not affecting anyother flags.

Example:STC

CMA(Complementthe Accumulator)
This type of instruction generates the complement of data at the accumulator. So,this function does not change any of the flags.

Example:CMA

CMC(ComplementtheCarryFlag)
This type of instruction is used to complement the data available at the carry flag(CF). So this instruction does not affect any other flag.

Example:CMC

Branching Instruction

These types of instructions are mainly used to transfer or switch themicroprocessor from one location to another. So, it simply changes the general sequential flow.

JMPaddress(Jump unconditionally)
This type of instruction is mainly used to transfer the series of the current program to thatlocationofmemorywhose16-bitaddress can besimplyspecified withinthe operand of the instruction.

Example:JMP2014H

JxAddress
This is a conditional branching type instruction, where the series of current programs can be transferred to that specific location whose address can beprovidedattheoperand.Howeverthistransferringmainlydependsonthespecified PSX flag.

Example:JZ 1200H

CALL address
This instruction shifts the control of a series of current programs toward the memory address available at the operand. However the PC gets decreased before transferring,

Example:CALL2400H

RET(Returnfromthe Subroutine)
This type of instruction cancause the unconditional return ofthe sub-routine to the actual program.

RST(RestartInstruction)
This typeof instruction is mainlyused to transferthe series fromthe main program to the interrupt service routine. Mostly, the transfer can be performed above one of the 8-bits which are indicated within the operand.

ControlInstruction

These instructions are mainlyused to control the microprocessor operations. These instructions are discussed below.

NOP(Nooperation)
NOP stands for no operation. Once the 8085 microprocessor gets this instruction, then it does not perform any operation based on execution.

DI(DisableInterrupts)
DI is the disabling of the interrupt that is generated within the microprocessor. Interrupt resetting will allows to disable all the interrupts apart from TRAP.

EI(EnableInterrupts)
This type of instruction is mainly used to allow the interrupt. Once the interrupt enable pin is set then leads to enabling the interrupts within the system.

HLT (Halt &EnterWait State)
Once the HLT interrupt is decoded through the microprocessor, it stops the current operation and waits for furtherinstruction.To escapefromthe halt condition either a reset or an interrupt is necessary.

SIM(SetInterruptMask)
SIM is the set interrupt mask, which is used to execute the hardwareinterruptsprogramming & serial output.
RIM(ReadInterrupt Mask)
RIMisthereadinterruptmaskthatisusedtosituatethepreferreddataatthe accumulator based on the serial input & interrupt.

Addressingmode

These are the instructions used to transfer the data from one register to another register, from the memory to the register, and from the register to the memory without any alteration in the content

Thetermaddressingmodereferstothewayinwhichtheoperandofthe instruction is specified

TypesofAddressingModes

Intel8085usesthefollowingaddressingmodes:

1. **DirectAddressingMode**
2. **RegisterAddressingMode**
3. **RegisterIndirectAddressingMode**
4. **ImmediateAddressingMode**
5. **ImplicitAddressingMode**

1. **DirectAddressingMode:-**

Theaddressoftheoperand(data)isdirectlyavailablein theinstructionitself.

Indirectaddressingmode,thedatatobeoperatedisavailableinsideamemory location and that memory location is directly specified as an operand

Examples:
LDA2050(loadthecontentsofmemorylocationintoaccumulatorA) LHLD address (load contents of 16-bit memory location into H-L register pair)IN 35 (read the data from port whose address is 35)

2. **RegisterAddressingMode:-**

   In register addressing the operand is one of the general purpose registers. the opcode specifies the address of the register in addition to the operation to be performed.

   In register addressing mode, the data to be operated is available inside the register(s) and register(s) is operands. Therefore the operation is performed within various registers of the microprocessor.

Examples:
MOVA,B(movethecontentsofregisterBtoregister A)
ADDB(add contentsofregistersAand Bandstoretheresultinregister A) INR A (increment the contents of register A by one)

3. **RegisterIndirectAddressingMode**

   In this mode of addressing the address of the operand is specified by a register pair.

   Inregisterindirectaddressingmode,thedata tobeoperatedis availableinsidea memory location and that memory location is indirectly specified by a register pair.

Examples:
MOVA,M(movethecontentsofthememorylocation pointedbytheH-Lpairto the accumulator)
LDAXB(movecontentsofB-Cregistertotheaccumulator)
LXIH9570(loadimmediatetheH-Lpairwith theaddressofthelocation9570)

4. **ImmediateAddressingMode**

In immediate addressing mode the source operand is always data. If the data is 8-bit, then the instruction will be of 2 bytes, if the data is of 16-bit then the instruction will be of 3 bytes.

Examples:

MVIB45(movethedata45Himmediatelytoregister B)
LXIH3050(loadtheH-Lpairwiththeoperand3050Himmediately) JMP address (jump to the operand address immediately)

5. ImplicitAddressingMode

Inimplied/implicitaddressingmodetheoperandishiddenandthedatatobe operated is available in the instruction itself.

Examples:

CMA(findsandstoresthe1'scomplementofthecontentsofaccumulatorAinA) RRC (rotate accumulator A right by one bit)
RLC(rotateaccumulatorAleft byonebit)

TimingDiagram:

Timing Diagram is a graphical representation. It represents the execution time taken by each instruction in a graphical format. The execution time is represented in T-states.

InstructionCycle:

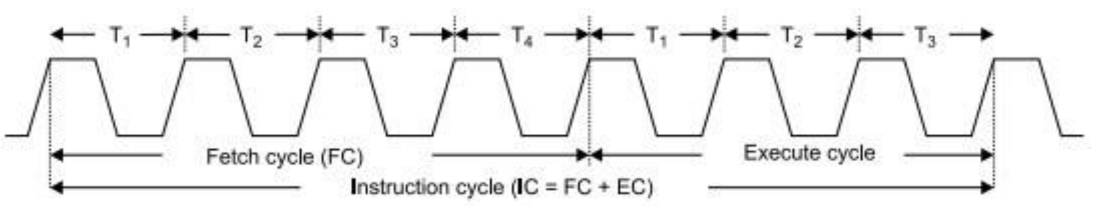
Thetimerequiredtoexecuteaninstructioniscalledinstructioncycle. or

Thetimetakenbytheprocessortocompletetheexecutionofaninstruction.An

instruction cycle consists of one to six machine cycles.

Fetchcycle:

The fetch cycle in a microprocessor comprises(consist) of several time states during which the next instruction to be executed is copied (fetched) from the memory location (whose address is in the Program Counter) to the Instruction Register.



IC=FC+EC

MachineCycle:

The time required to access the memory or input/output devices is called machine cycle.

or

The time required to complete one operation; accessing either the memory or I/Odevice. A machine cycle consists of three to six T-states.

T-State:

The machine cycle and instruction cycle takes multiple clock periods. A portion of an operation carried out in one system clock period is called as T-state.

Or

Time corresponding to one clock period. It is the basic unit to calculate execution of instructions or programs in a processor.

Rulestoidentifynumberofmachinecyclesinaninstruction:

1. Ifanaddressingmodeisdirect,immediateorimplicitthenNo.ofmachine cycles = No. of bytes.

2. IftheaddressingmodeisindirectthenNo.ofmachinecycles=No.ofbytes+1. Add +1 to the No. of machine cycles if it is memory read/write operation.

3. Iftheoperandis 8-bitor16-bitaddressthen,No.ofmachinecycles=No.of bytes +1.

4. Theserulesareapplicableto80%oftheinstructionsof8085.

CONCEPTOFTIMINGDIAGRAM:

The8085microprocessorhas5(seven)basic machinecycles.Theyare

1. **Opcodefetchcycle(4T)**
2. **Memoryreadcycle(3T)**
3. **Memorywritecycle(3T)**
4. **I/Oreadcycle(3T)**
5. **I/O writecycle(3 T**

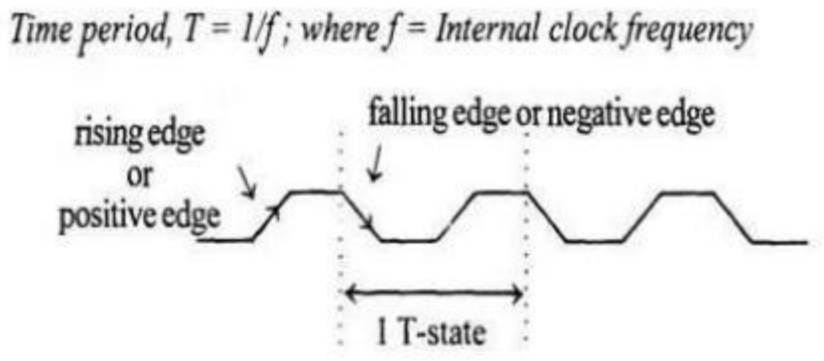

Time period, T = 1/f ; where f = Internal clock frequency
rising edge or positive edge
falling edge or negative edge
1 T-state

Fig 1.7 Clock Signal

TimingDiagramofOpcodefetchof8085:

The microprocessor requires instructions to perform any particular action. In order to perform these actions microprocessor utilizes Opcode which is a part of an instruction which provides detail(ie. Which operation µp needs to perform) to microprocessor.

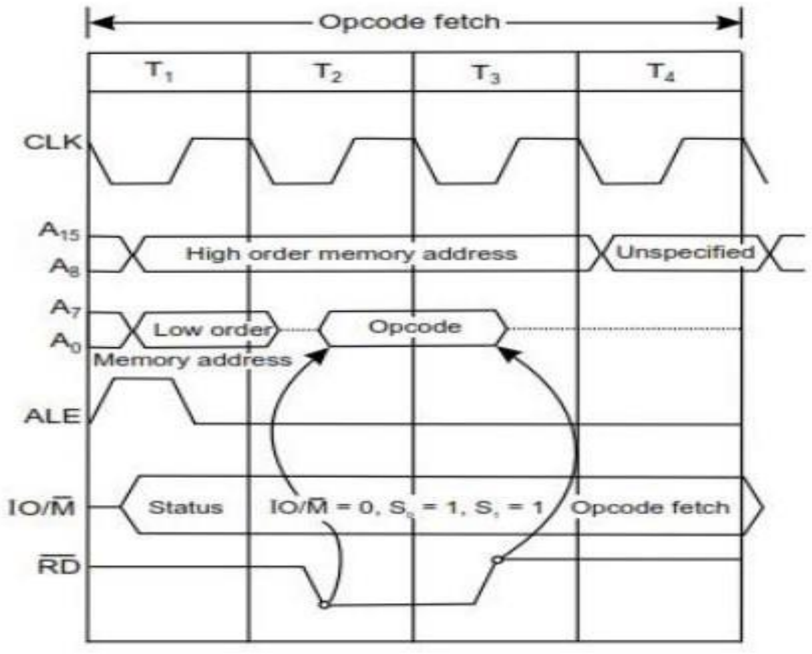


Fig 1.8 Opcode fetch machine cycle

Eachinstructionoftheprocessorhasonebyteopcode.

Theopcodesarestoredinmemory.So,theprocessorexecutestheopcodefetch machine cycle to fetch the opcode from memory.

Hence,everyinstructionstartswithopcodefetchmachinecycle.

Thetimetakenbytheprocessorto executetheopcodefetchcycleis4T.

In this time, the first, 3 T-states are used for fetching the opcode from memory and the remaining T-states are used for internal operations by the processor.

TimingDiagramofMemoryRead

The memory readmachine cycle is executed by the processor to read a data bytefrom memory.

Theprocessortakes3Tstatestoexecute thiscycle.

Theinstructionswhichhavemorethanonebytewordsizewillusethemachine cycle after the opcode fetch machine cycle.

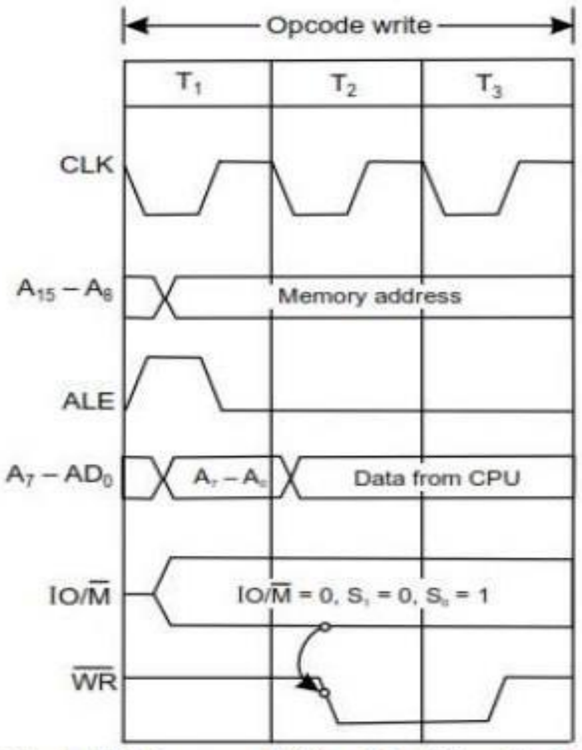


Fig 1.10 Memory Write Machine Cycle

TimingDiagramofMemoryWrite

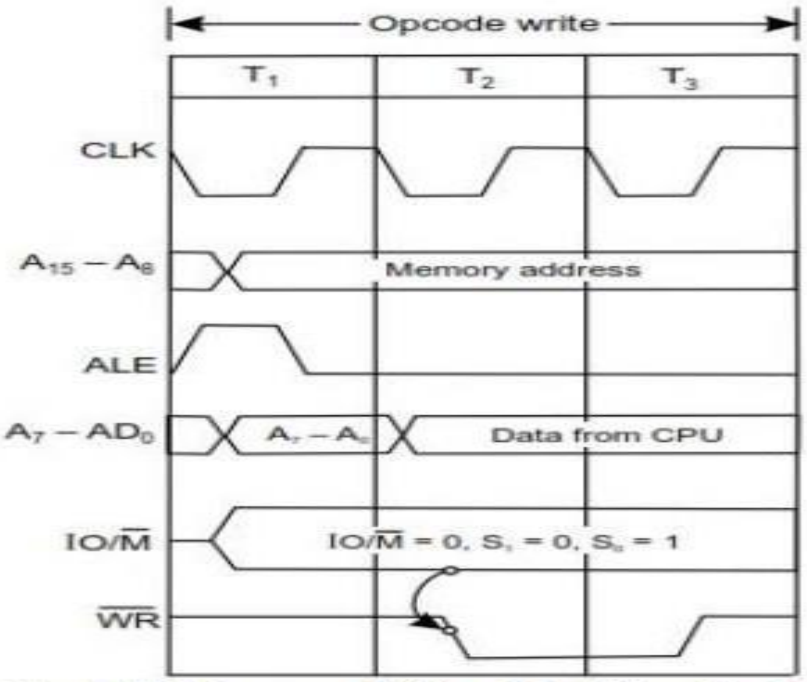


Fig 1.10 Memory Write Machine Cycle

The memory write machine cycle is executed by the processor to write a data byte in a memory location.

Theprocessortakes,3Tstatesto executethismachinecycle.

TimingDiagramofI/ORead

The I/O Read cycle is executed by the processor to read a data byte from I/O port or from the peripheral, which is I/O, mapped in the system.

The processor takes 3T states to execute this machine

cycle.TheINinstructionusesthismachinecycleduringtheexecution

.

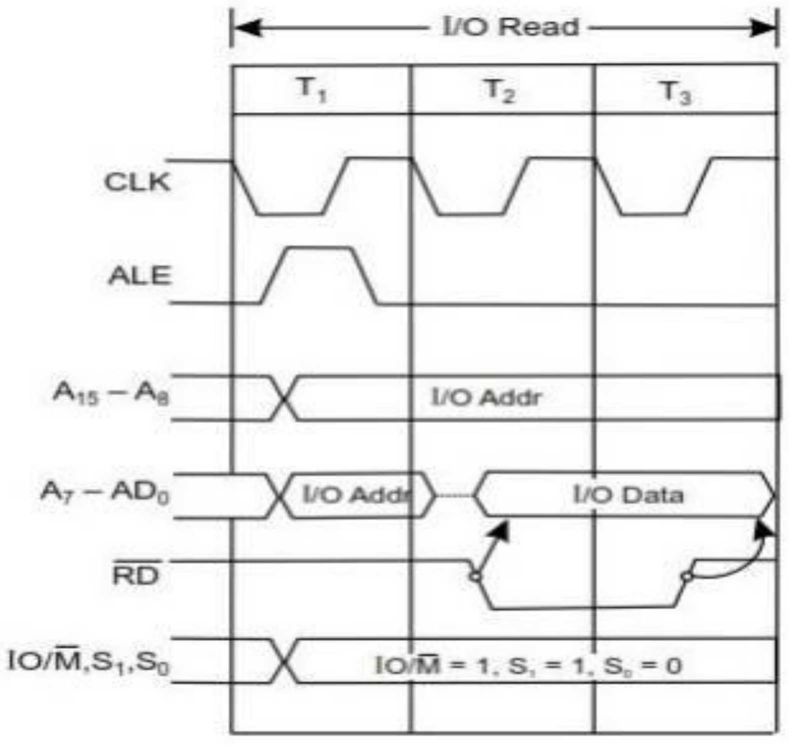


Fig 1.11 I/O Read Cycle

TimingdiagramforSTA526AH

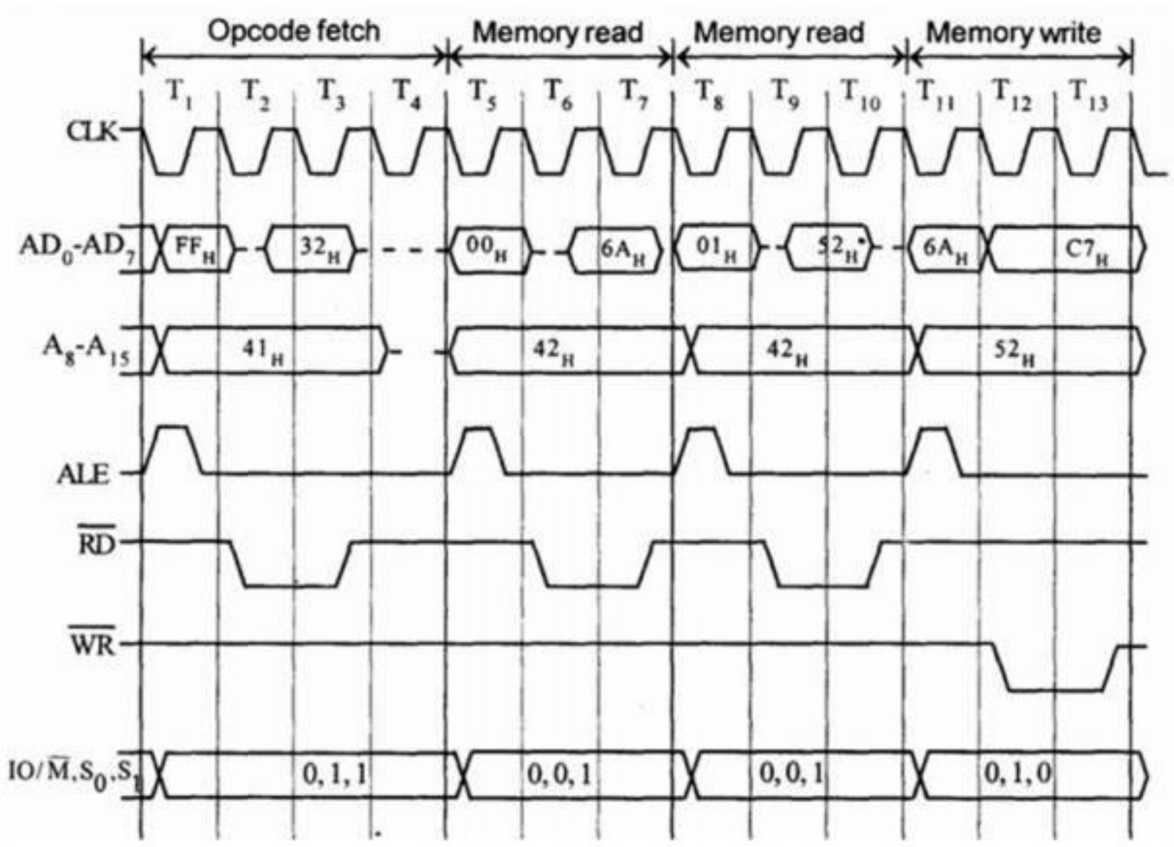


Fig 1.12 Timing Diagram for STA 526A H

Address	Mnemonics	Opcode
41FF	STA 526AH	32H
4200		6AH
4201		52H

STA means Store Accumulator -Thecontents of the accumulator isstored inthespecified address (526A).

TheopcodeoftheSTAinstructionissaidtobe32H.Itisfetchedfromthe memory 41FFH (see fig). - OF machine cycle

Thenthelowerordermemoryaddressisread(6A).-MemoryReadMachine Cycle

Readthehigherordermemoryaddress(52).-MemoryReadMachine Cycle

Thecombinationofboththeaddressesareconsideredandthecontentfrom accumulator is written in 526A. - Memory Write Machine Cycle

Assume the memory address for the instruction and let the content of accumulator is C7H. So, C7H from accumulator is now stored in 526A.

TimingdiagramforINRM

Fetchingthe Opcode34Hfromthememory4105H.(OF cycle)

Let the memory address (M) be 4250H. (MR cycle -To read Memory address and data)

Letthecontentofthatmemoryis12H.

Incrementthe memorycontentfrom12Hto13H.(MWmachine cycle)

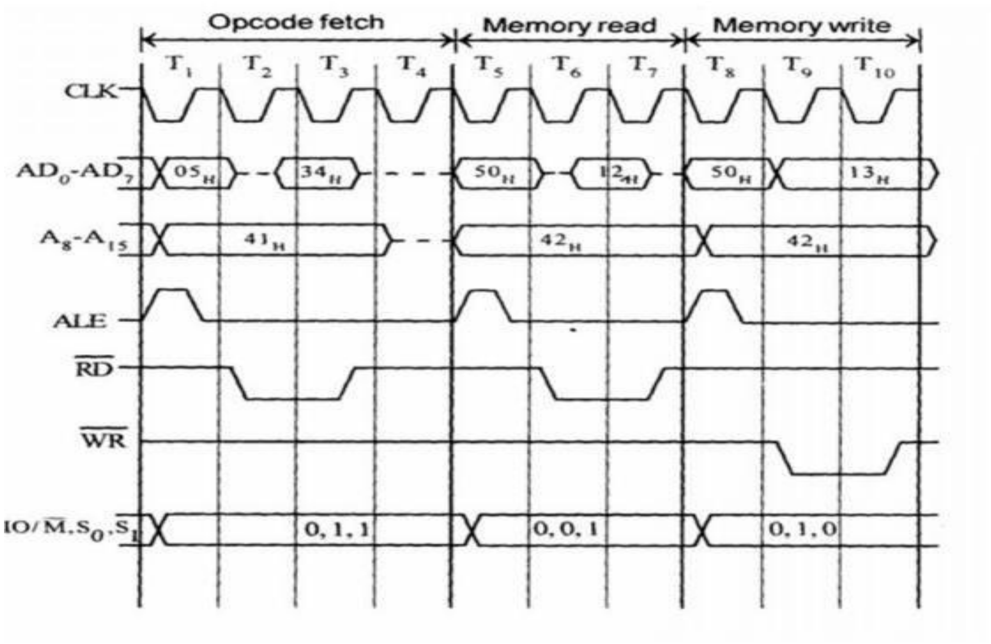
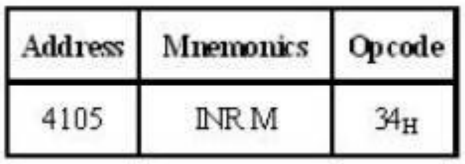


Fig 1.13 Timing Diagram for INR M



Address	Mnemonics	Opcode
4105	INR M	34_H

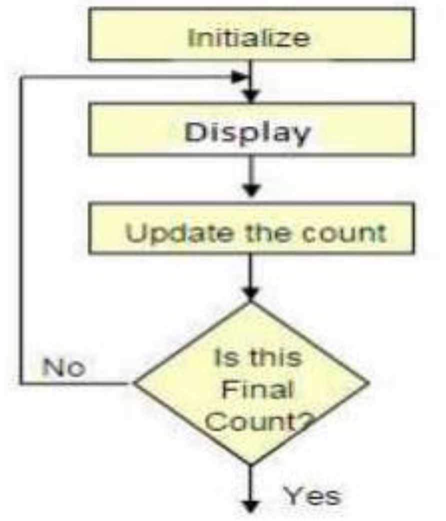
Counterandtime delay.

Counter:

Acounterisdesignedsimplybyloadingappropriatenumberintooneofthe registers and using INR or DNR instructions.

Loopisestablishedtoupdate thecount.

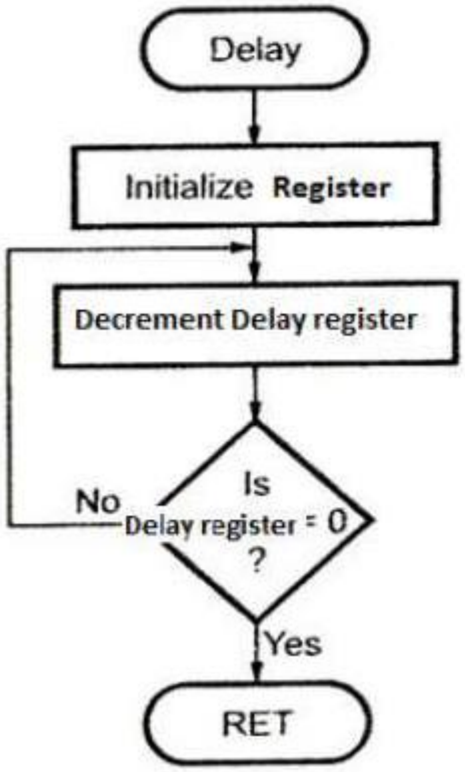
Each count is checked to determine whether it has reached final number ;if not, the loop is repeated.



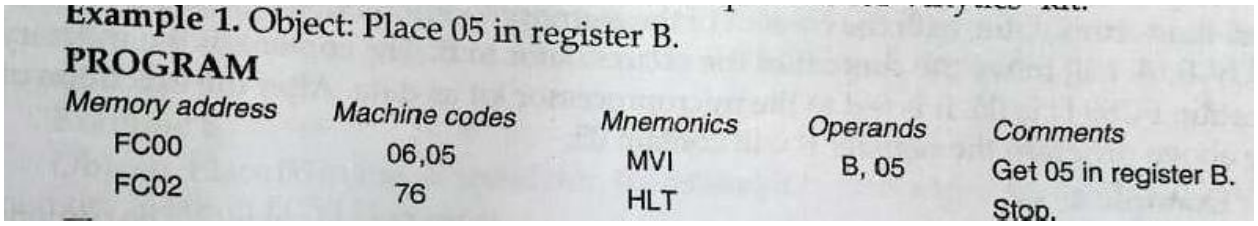
Timedelay:

Procedureusedtodesign aspecific delay.′

Aregisterisloadedwithanumber,depending onthe′timedelayrequiredandthen theregisterisdecrementeduntilitreacheszerobysettingup conditional aloopwith jump instruction.



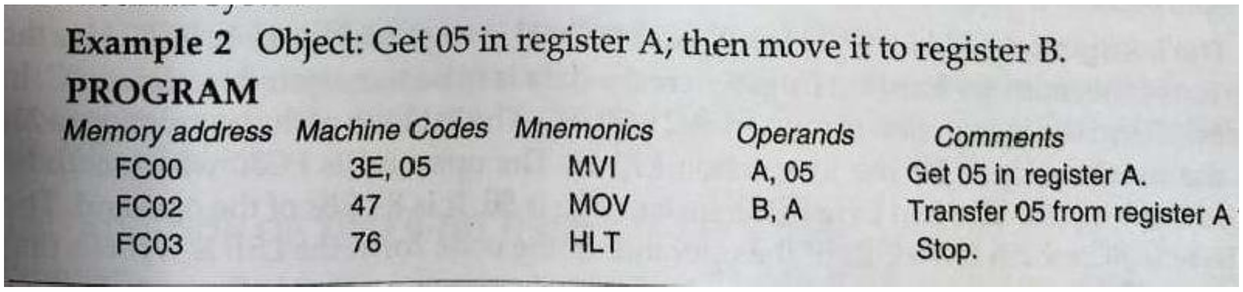
Simpleassemblylanguageprogrammingof8085.



Example 1. Object: Place 05 in register B.

PROGRAM

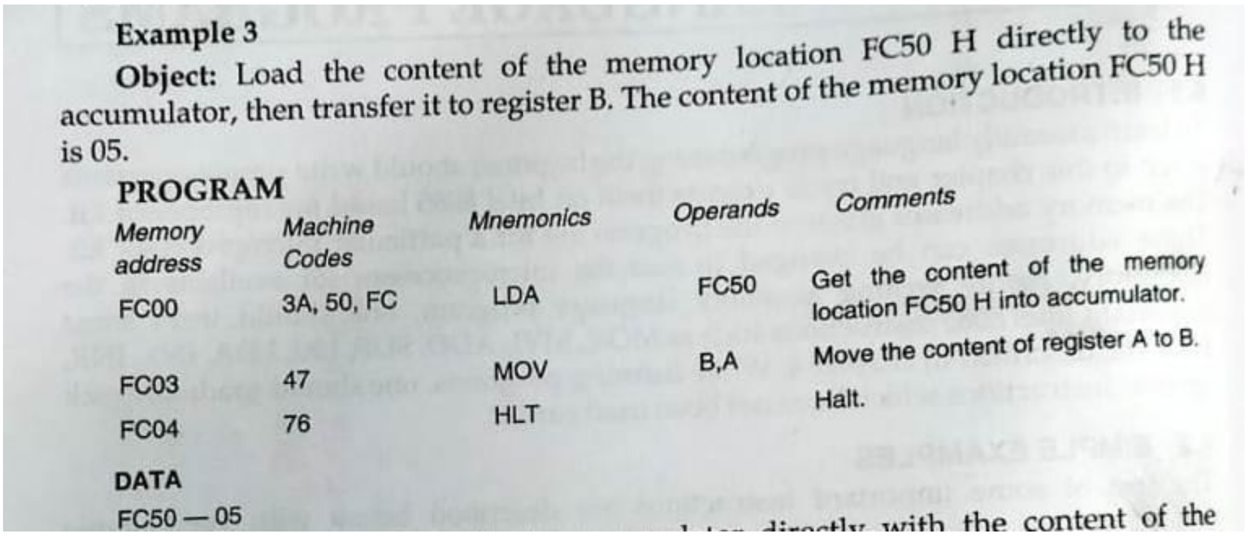
Memory address	*Machine codes*	*Mnemonics*	*Operands*	*Comments*
FC00	06,05	MVI	B, 05	Get 05 in register B.
FC02	76	HLT		Stop.



Example 2 Object: Get 05 in register A; then move it to register B.

PROGRAM

Memory address	*Machine Codes*	*Mnemonics*	*Operands*	*Comments*
FC00	3E, 05	MVI	A, 05	Get 05 in register A.
FC02	47	MOV	B, A	Transfer 05 from register A
FC03	76	HLT		Stop.



Example 3

Object: Load the content of the memory location FC50 H directly to the accumulator, then transfer it to register B. The content of the memory location FC50 H is 05.

PROGRAM

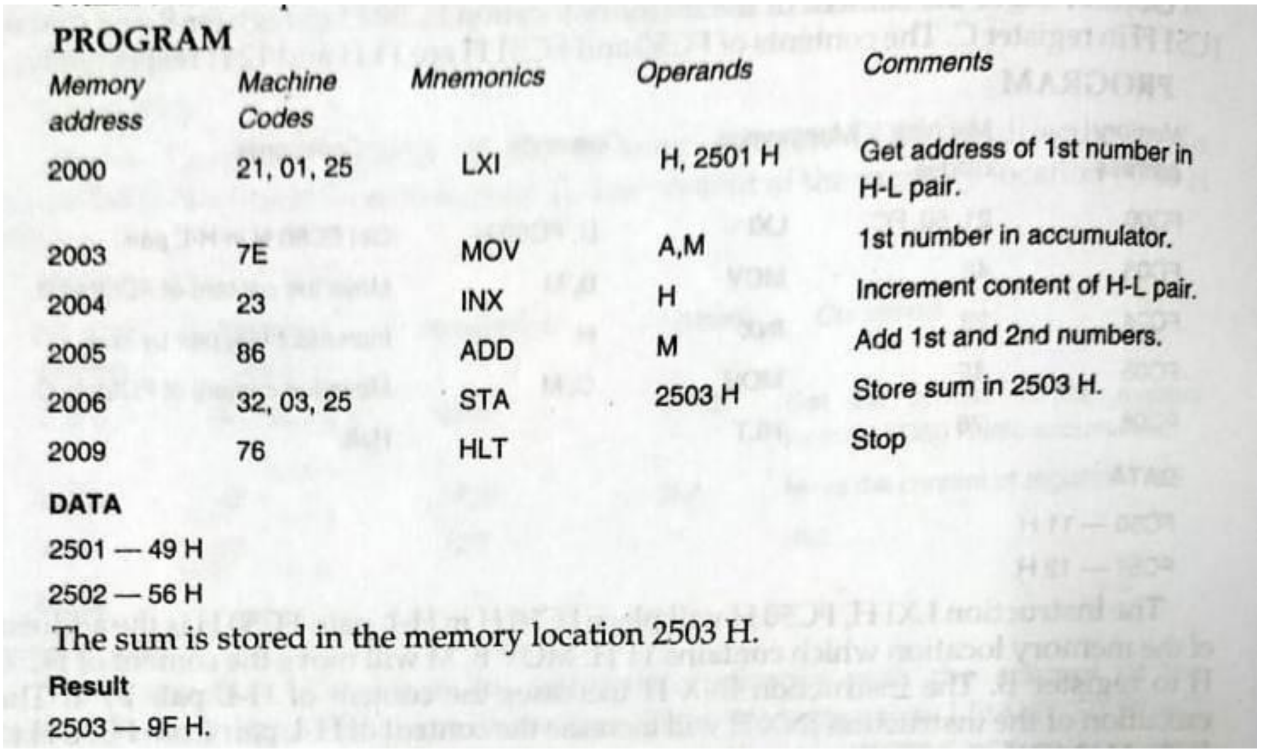
Memory address	*Machine Codes*	*Mnemonics*	*Operands*	*Comments*
FC00	3A, 50, FC	LDA	FC50	Get the content of the memory location FC50 H into accumulator.
FC03	47	MOV	B,A	Move the content of register A to B.
FC04	76	HLT		Halt.

DATA

FC50 — 05

directly with the content of the

AdditionofTwo8-bitNo.;sum8-bit



PROGRAM

Memory address	*Machine Codes*	*Mnemonics*	*Operands*	*Comments*
2000	21, 01, 25	LXI	H, 2501 H	Get address of 1st number in H-L pair.
2003	7E	MOV	A,M	1st number in accumulator.
2004	23	INX	H	Increment content of H-L pair.
2005	86	ADD	M	Add 1st and 2nd numbers.
2006	32, 03, 25	STA	2503 H	Store sum in 2503 H.
2009	76	HLT		Stop

DATA

2501 — 49 H

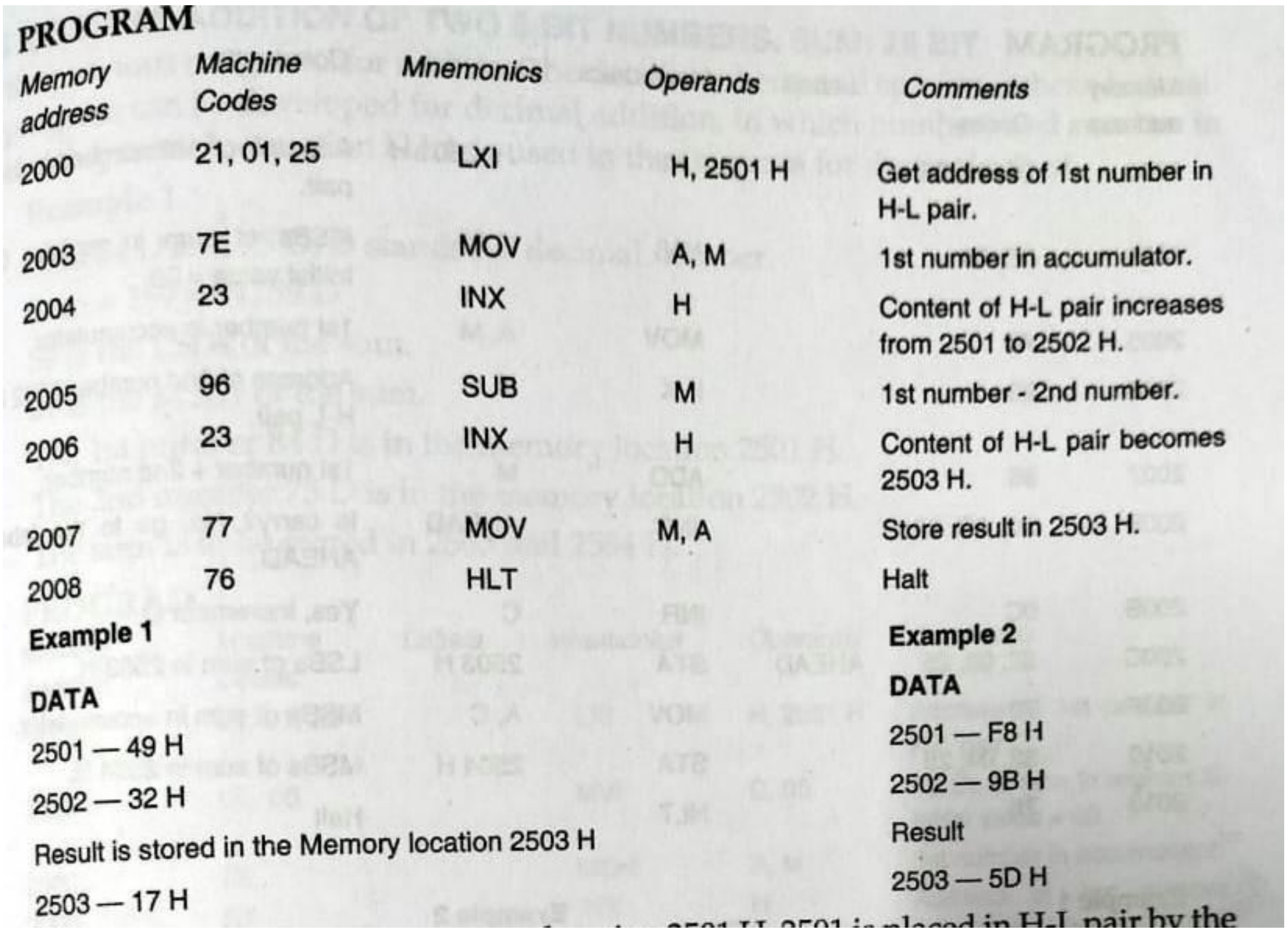
2502 — 56 H

The sum is stored in the memory location 2503 H.

Result

2503 — 9F H.

8-bitSubtraction



PROGRAM

Memory address	*Machine Codes*	*Mnemonics*	*Operands*	*Comments*
2000	21, 01, 25	LXI	H, 2501 H	Get address of 1st number in H-L pair.
2003	7E	MOV	A, M	1st number in accumulator.
2004	23	INX	H	Content of H-L pair increases from 2501 to 2502 H.
2005	96	SUB	M	1st number - 2nd number.
2006	23	INX	H	Content of H-L pair becomes 2503 H.
2007	77	MOV	M, A	Store result in 2503 H.
2008	76	HLT		Halt

Example 1

DATA

2501 — 49 H

2502 — 32 H

Result is stored in the Memory location 2503 H

2503 — 17 H

Example 2

DATA

2501 — F8 H

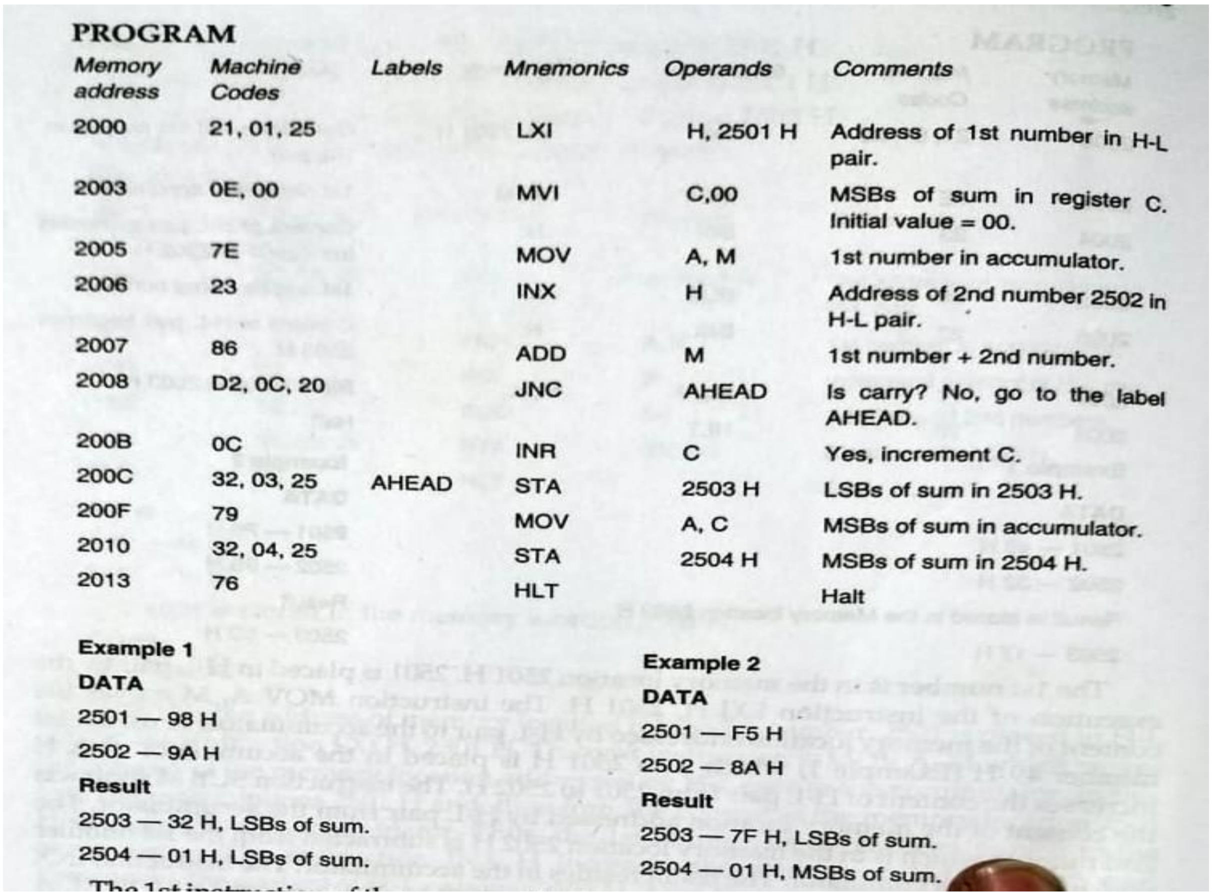
2502 — 9B H

Result

2503 — 5D H

2501 H. 2501 is placed in H-L pair by the

ADDITIONOFTWO8-BITNO.;SUM:16-BIT



PROGRAM

Memory address	*Machine Codes*	*Labels*	*Mnemonics*	*Operands*	*Comments*
2000	21, 01, 25		LXI	H, 2501 H	Address of 1st number in H-L pair.
2003	0E, 00		MVI	C,00	MSBs of sum in register C. Initial value = 00.
2005	7E		MOV	A, M	1st number in accumulator.
2006	23		INX	H	Address of 2nd number 2502 in H-L pair.
2007	86		ADD	M	1st number + 2nd number.
2008	D2, 0C, 20		JNC	AHEAD	Is carry? No, go to the label AHEAD.
200B	0C		INR	C	Yes, increment C.
200C	32, 03, 25	AHEAD	STA	2503 H	LSBs of sum in 2503 H.
200F	79		MOV	A, C	MSBs of sum in accumulator.
2010	32, 04, 25		STA	2504 H	MSBs of sum in 2504 H.
2013	76		HLT		Halt

Example 1

DATA

2501 — 98 H

2502 — 9A H

Result

2503 — 32 H, LSBs of sum.

2504 — 01 H, LSBs of sum.

Example 2

DATA

2501 — F5 H

2502 — 8A H

Result

2503 — 7F H, LSBs of sum.

2504 — 01 H, MSBs of sum.

The 1st instruction of th

CHAPTER-5

INTERFACINGANDSUPPORTCHIPS

BasicInterfacingConcepts

Interface isthepathforcommunicationbetweentwo components.Interfacingisoftwo types, memory interfacing and I/O interfacing.

Memorymapping&I/Omapping

Functional block diagram and description of each blockofProgrammableperipheralinterfaceIntel8255

The parallel input-output port chip 8255 is also called as programmable peripheral input- output port. The Intel's 8255 is designed for use with Intel's 8-bit, 16-bitand higher capability microprocessors.

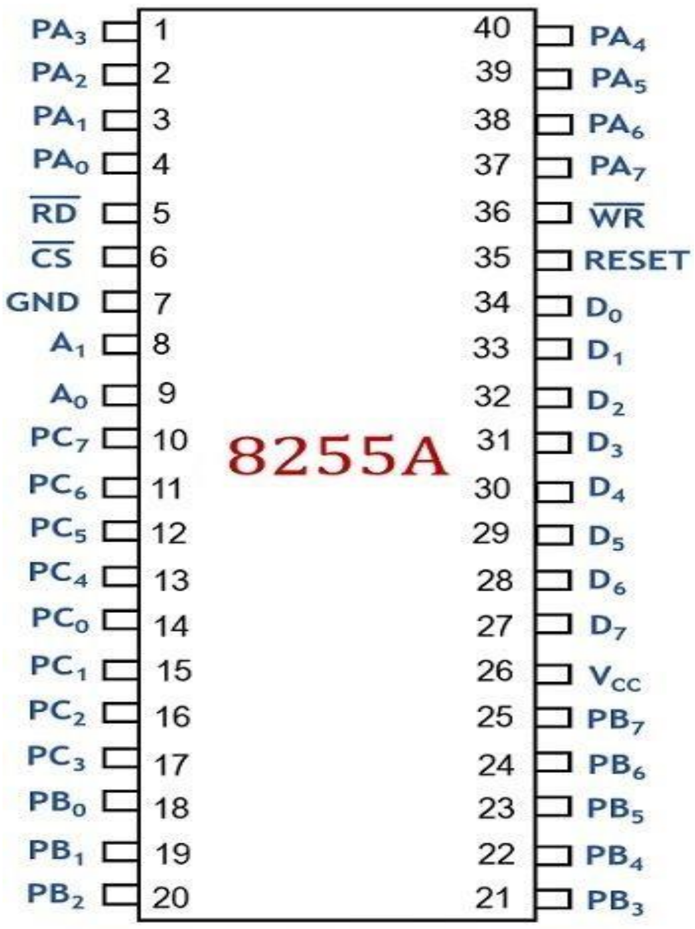
The 8255A is a general purpose programmable I/O device designed to transfer the data from I/O to interrupt I/O under certain conditions as required. It can be used with almost any microprocessor.

Portsof8255A
8255Ahasthreeports,i.e.,PORTA,PORTB,andPORTC.

PortAcontainsone8-bitparallelporti.ePA_0 $-PA_7$

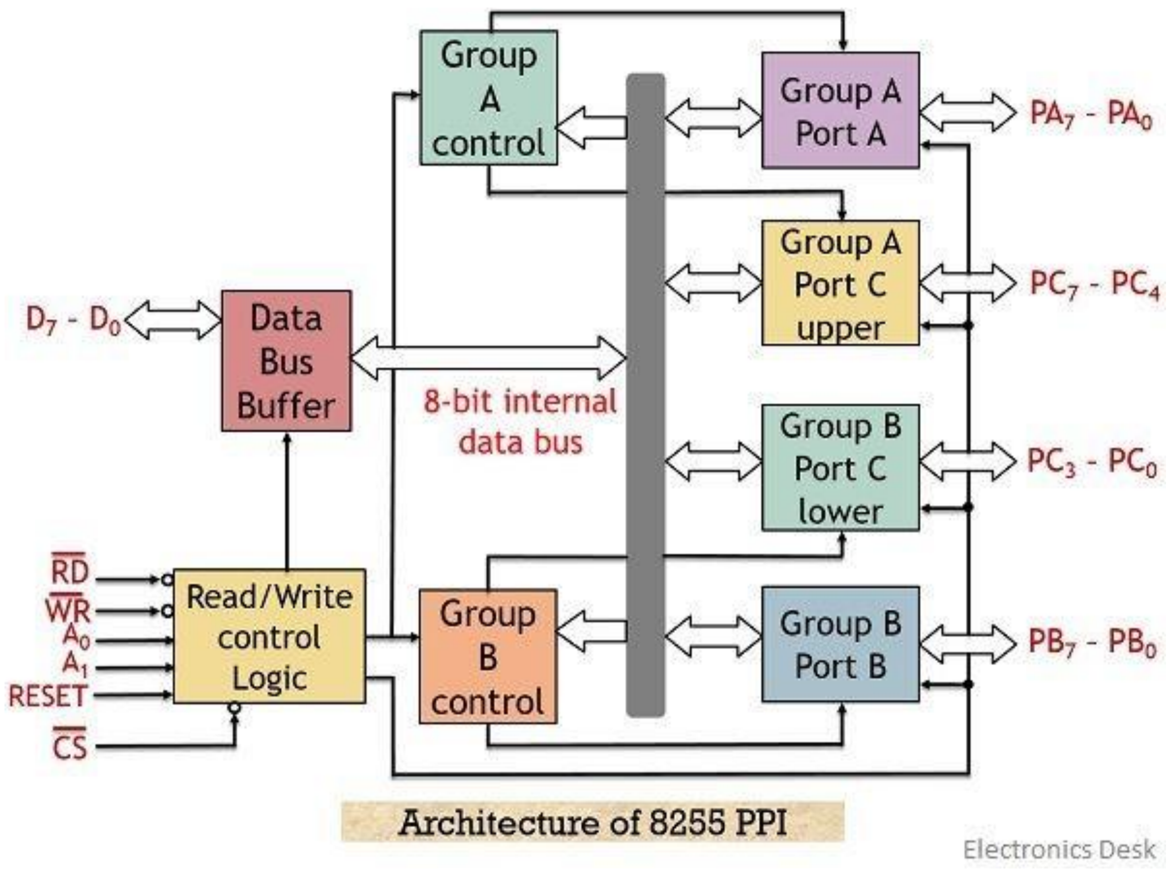
PortBcontainsone8-bitparallelporti.e$PB_0$$-PB_7$

Port Ccan be split into two parts, i.e. PORT C lower ($PC_0$$-PB_3$) and PORT C upper(e$PC_4$$-$ PC_7)bythecontrol word.



Pin Diagram of 8255 PPI

Electronics Desk



DataBusBuffer

It is a tri-state 8-bit buffer, which is used to interface the microprocessor to the system data bus. Data is transmitted or received by the buffer as per the instructions by the CPU. Control words and status information is also transferred using this bus.

Read/Writecontrollogic:

This unit manages the internal operations of the system. This unit holds theability to control the transfer of data and control or status words both internally and externally.

Whenevertherexistsaneedfordatafetchthenitacceptstheaddressprovidedby the processor through the bus and immediately generates command to the 2 control groups for the particular operation.

Group AandGroupBcontrol:

These two groups are handled by the CPU and functions according to the command generated by the CPU. The CPU sends control words to the group A and group B control and they in turn sends the appropriate command to their respective port.

the group A has the access of the port A and higher order bits of port C. While group B controls port B with the lower order bits of port C.

$\overline{\square\square}$: It stands for chip select. A low signal at this pin shows the enabling of communication between the 8255 and the processor. More specifically we can say that the data transfer operation gets enabled by an active low signal at this pin.

$\overline{\square\square}$ – It is the signal used for read operation. A low signal at this pin shows that CPU is performing read operation at the ports or status word. Or we can say that 8255 is providing data or information to the CPU through data buffer.

$\overline{\square\square}$ - It shows write operation. A low signal at this pin allows the CPU to perform write operation over the ports or control register of 8255 using the data bus buffer.

$A_0$and A_1: Thesearebasicallyusedtoselectthedesiredportamongalltheportsof the 8255 and it do so by forming conjunction with RD and WR. It forms connection with the LSB of the address bus.
Thetablebelowshowstheoperationofthecontrolsignals:

Forthe1stunitof8255,i.e8255.1

A_1	A_0	Port/Control word Register address	Deviceselected
0	0	00	Port-A
0	1	01	Port-B
1	0	02	Port-C
1	1	03	ControlRegister

Forthe 2ndunitof8255,i.e8255.2

A_1	A_0	Port/Control word Register address	Deviceselected
0	0	08	Port-A
0	1	09	Port-B
1	0	0A	Port-C
1	1	0B	ControlRegister

Reset: It is an active high signal that shows the resetting of the PPI. A high signal at this pin clears the control registers and the ports are set in the input mode. Initializingtheportstoinputmodeisdonetoprevent circuitbreakdown.Asincase of reset condition, if the ports are initialized to output mode then there exist chances of destruction of 8255 along with the processor.

Operatingmodeof8255

Operatingmodecanbeclassifiedasfollows
Mode0:Simpleinput/output
Mode1:Inputoutput withhandshaking
Mode2:BidirectionalI/Ohandshaking

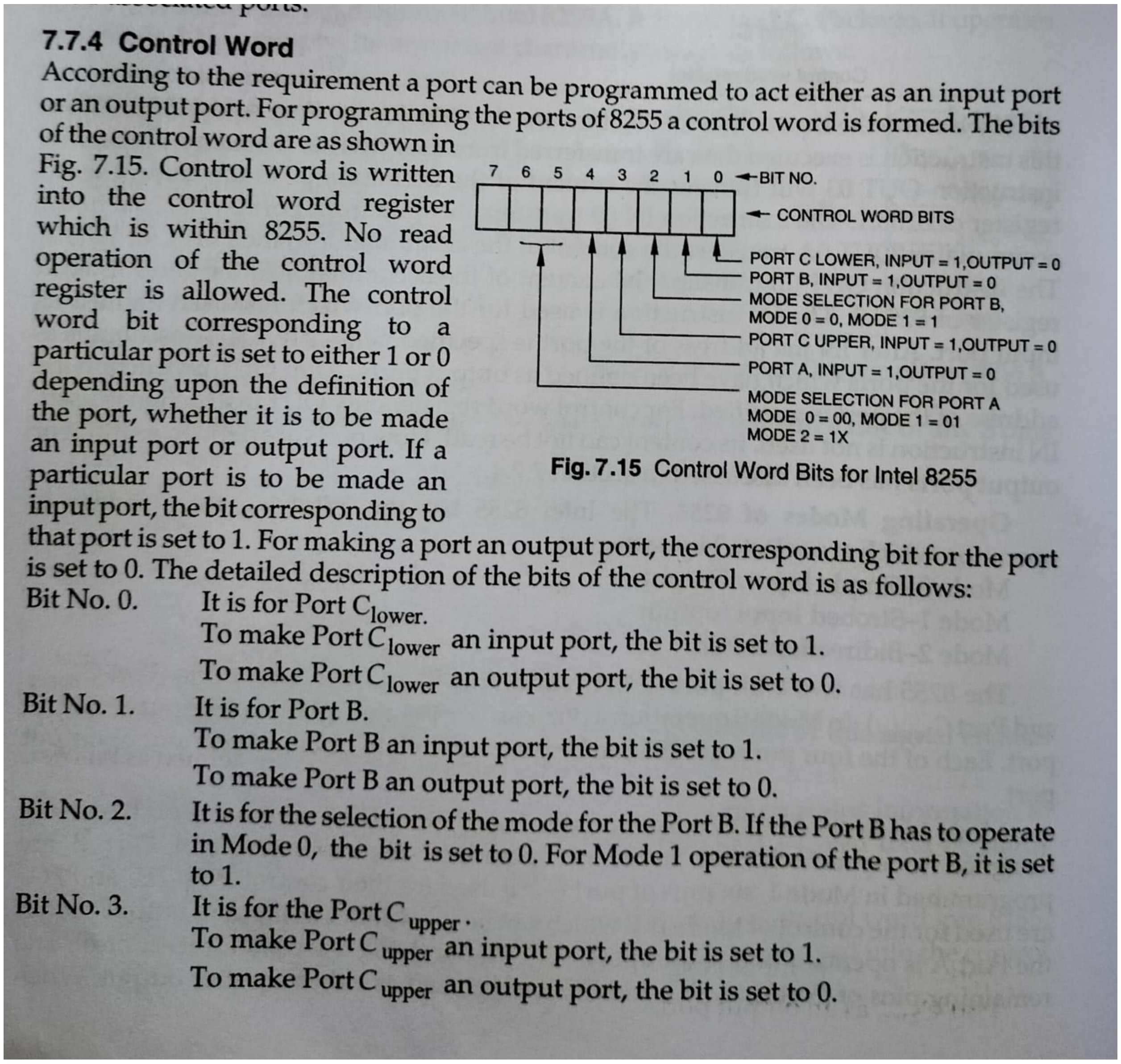
Mode0:**Simpleinput/output:-**
In this mode, all the three ports can be programmed either as the input or the output port. Each port can be programmed in either input mode or output mode where outputs are latched and inputs are not latched. Ports do not have interrupt capability. The ports in mode-0 can be used to interface DIP switches, hexa-keypad, LEDs and 7-segment LEDs to the processor

Mode1:Inputoutputwithhandshaking
In mode 1 , only port A and B can be programmed either as theinput or output port . the port-Care used for handshaking and interrupt control signals. Input and output data are latched. Interrupt driven data transfer scheme is possible

Mode2:BidirectionalI/Ohandshaking

Inthismode,alltheportwillbeabidirectionalport(i.e.theprocessorcanperform both read and write operations with an IO device connected to a port in mode-2) only port-A can be programmed to work in mode-2. Five pins of port-C are used for handshake signals. This mode is used primarily in applications such as data transfer between two computers or floppy disk controller interface

ControlWord:-



associated ports.

7.7.4 Control Word

According to the requirement a port can be programmed to act either as an input port or an output port. For programming the ports of 8255 a control word is formed. The bits of the control word are as shown in Fig. 7.15. Control word is written into the control word register which is within 8255. No read operation of the control word register is allowed. The control word bit corresponding to a particular port is set to either 1 or 0 depending upon the definition of the port, whether it is to be made an input port or output port. If a particular port is to be made an input port, the bit corresponding to that port is set to 1. For making a port an output port, the corresponding bit for the port is set to 0. The detailed description of the bits of the control word is as follows:

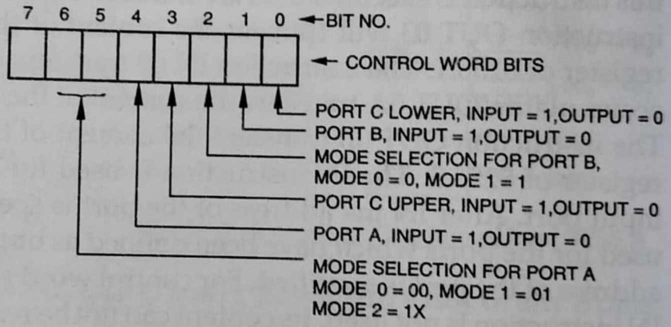

7 6 5 4 3 2 1 0 ← BIT NO.
← CONTROL WORD BITS
PORT C LOWER, INPUT = 1,OUTPUT = 0
PORT B, INPUT = 1,OUTPUT = 0
MODE SELECTION FOR PORT B, MODE 0 = 0, MODE 1 = 1
PORT C UPPER, INPUT = 1,OUTPUT = 0
PORT A, INPUT = 1,OUTPUT = 0
MODE SELECTION FOR PORT A MODE 0 = 00, MODE 1 = 01 MODE 2 = 1X

Fig. 7.15 Control Word Bits for Intel 8255

Bit No. 0.	It is for Port C_{lower}.
	To make Port C_{lower} an input port, the bit is set to 1.
	To make Port C_{lower} an output port, the bit is set to 0.
Bit No. 1.	It is for Port B.
	To make Port B an input port, the bit is set to 1.
	To make Port B an output port, the bit is set to 0.
Bit No. 2.	It is for the selection of the mode for the Port B. If the Port B has to operate in Mode 0, the bit is set to 0. For Mode 1 operation of the port B, it is set to 1.
Bit No. 3.	It is for the Port C_{upper}.
	To make Port C_{upper} an input port, the bit is set to 1.
	To make Port C_{upper} an output port, the bit is set to 0.

Bit No. 4. It is for Port A.

To make Port A an input port, the bit is set to 1.

To make Port A an output Port, the bit is set to 0.

Bit No. 5 and 6. These bits are to define the operating mode of the Port A. For the various modes of Port A these bits are defined as follows:

Mode of Port A	*Bit No. 6*	*Bit No. 5*
Mode 0	0	0
Mode 1	0	1
Mode 2	1	0 or 1

For mode 2 bit No. 5 is set to either 0 or 1; it is immaterial.

Bit No. 7. It is set to 1 if Port A, B and C are defined as input/output port. It is set to 0 if the individual pins of the Port C are to be set or reset.

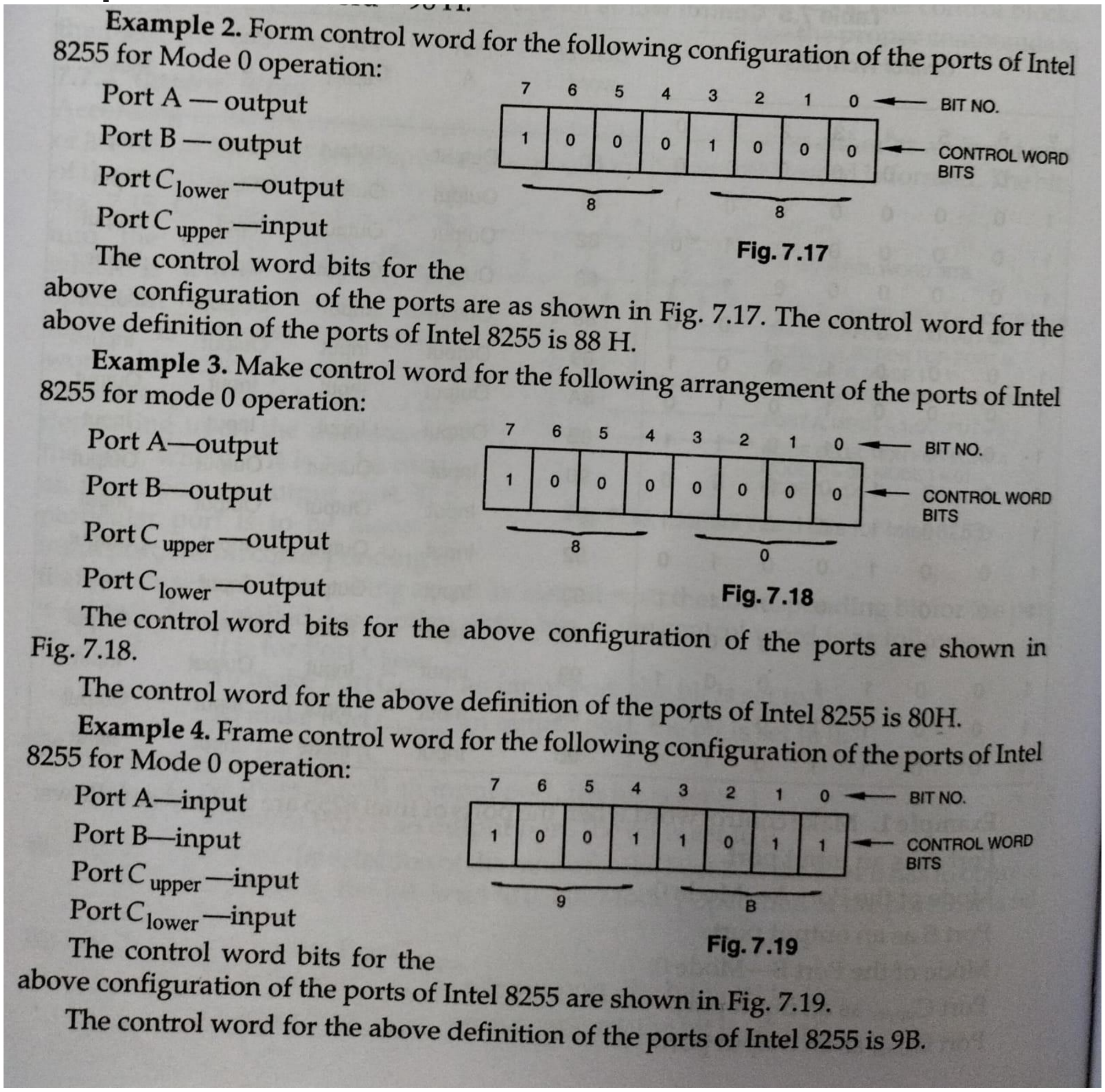
Table 7.5 shows control words for various configurations of the ports of 8255 for Mode 0 operation. The following examples will illustrate how to make control words:

Table 7.5 Control Words for 8255A for Mode 0 Operation

Control Word Bits								*Control word*	*Port A*	*Port C_{upper}*	*Port B*	*Port C_{lower}*
7	6	5	4	3	2	1	0					
1	0	0	0	0	0	0	0	80	Output	Output	Output	Output
1	0	0	0	0	0	0	1	81	Output	Output	Output	Input
1	0	0	0	0	0	1	0	82	Output	Output	Input	Output
1	0	0	0	0	0	1	1	83	Output	Output	Input	Input
1	0	0	0	1	0	0	0	88	Output	Input	Output	Output
1	0	0	0	1	0	0	1	89	Output	Input	Output	Input
1	0	0	0	1	0	1	0	8A	Output	Input	Input	Output
1	0	0	0	1	0	1	1	8B	Output	Input	Input	Input
1	0	0	1	0	0	0	0	90	Input	Output	Output	Output
1	0	0	1	0	0	0	1	91	Input	Output	Output	Input
1	0	0	1	0	0	1	0	92	Input	Output	Input	Output
1	0	0	1	0	0	1	1	93	Input	Output	Input	Input
1	0	0	1	1	0	0	0	98	Input	Input	Output	Output
1	0	0	1	1	0	0	1	99	Input	Input	Output	Input
1	0	0	1	1	0	1	0	9A	Input	Input	Input	Output
1	0	0	1	1	0	1	1	9B	Input	Input	Input	Input

of Intel 8255 are defined as follows:

Exampleareasfollows:



Example 2. Form control word for the following configuration of the ports of Intel 8255 for Mode 0 operation:

Port A — output

Port B — output

Port C_{lower} —output

Port C_{upper} —input

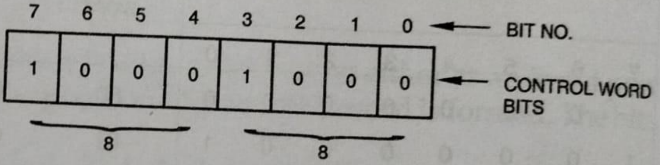


Fig. 7.17

The control word bits for the above configuration of the ports are as shown in Fig. 7.17. The control word for the above definition of the ports of Intel 8255 is 88 H.

Example 3. Make control word for the following arrangement of the ports of Intel 8255 for mode 0 operation:

Port A—output

Port B—output

Port C_{upper} —output

Port C_{lower} —output

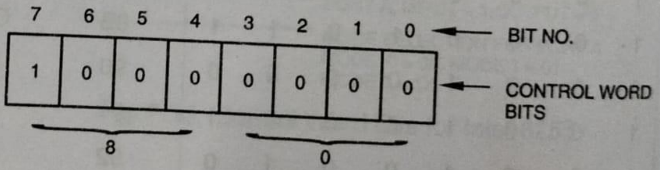


Fig. 7.18

The control word bits for the above configuration of the ports are shown in Fig. 7.18.

The control word for the above definition of the ports of Intel 8255 is 80H.

Example 4. Frame control word for the following configuration of the ports of Intel 8255 for Mode 0 operation:

Port A—input

Port B—input

Port C_{upper} —input

Port C_{lower} —input

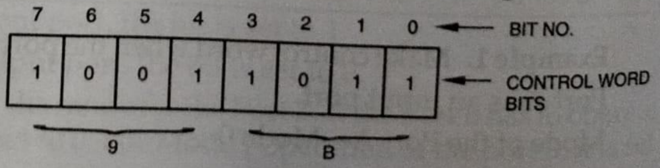
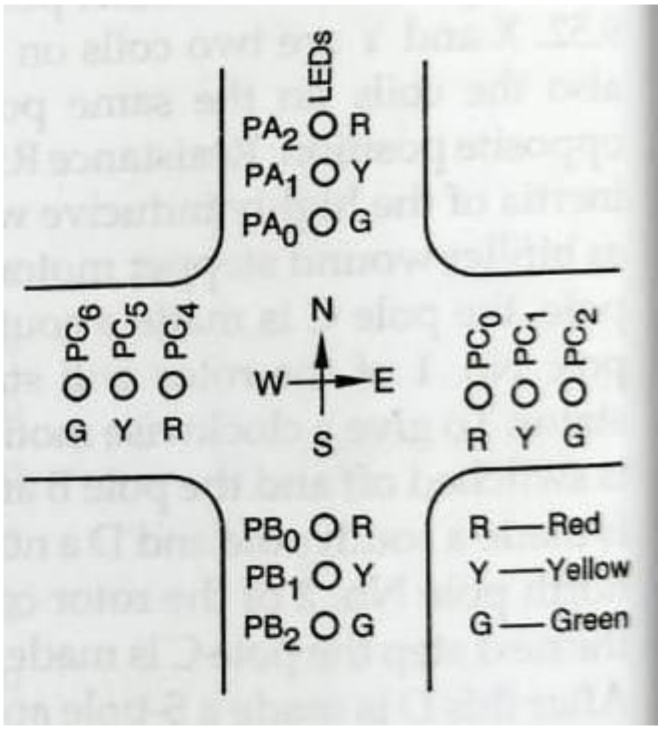


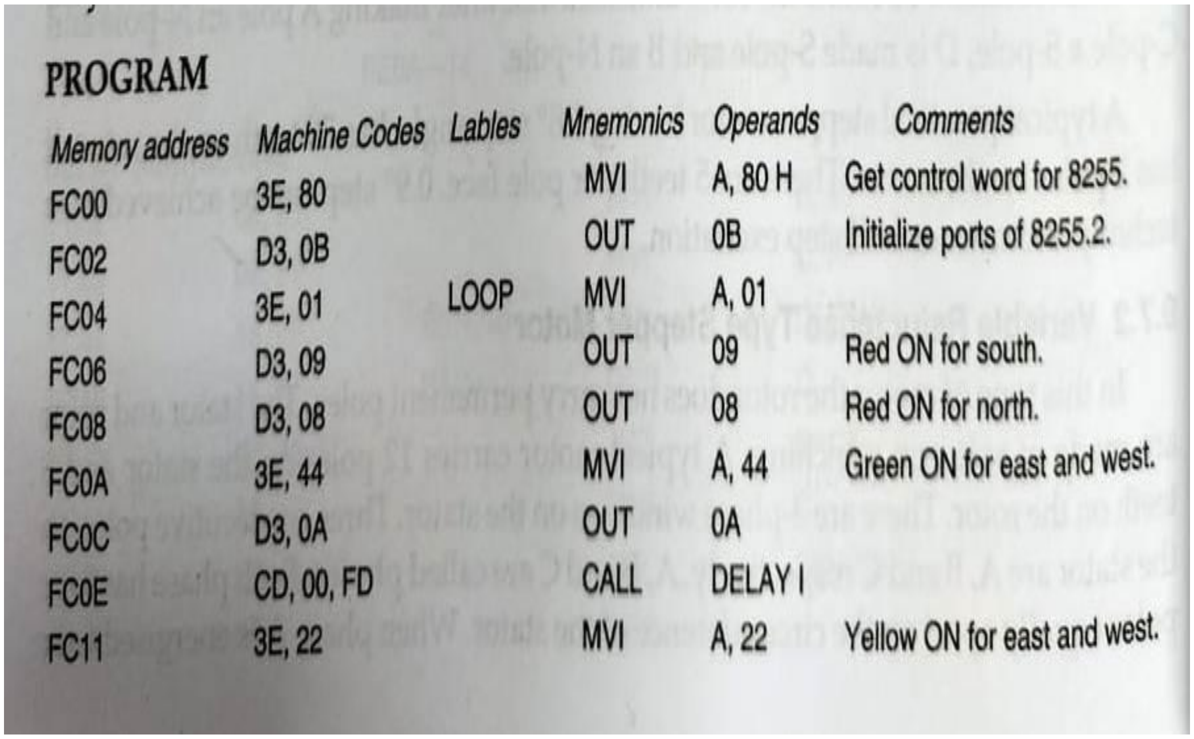
Fig. 7.19

The control word bits for the above configuration of the ports of Intel 8255 are shown in Fig. 7.19.

The control word for the above definition of the ports of Intel 8255 is 9B.

ProgramforTrafficlightControlusing8085microprocessor





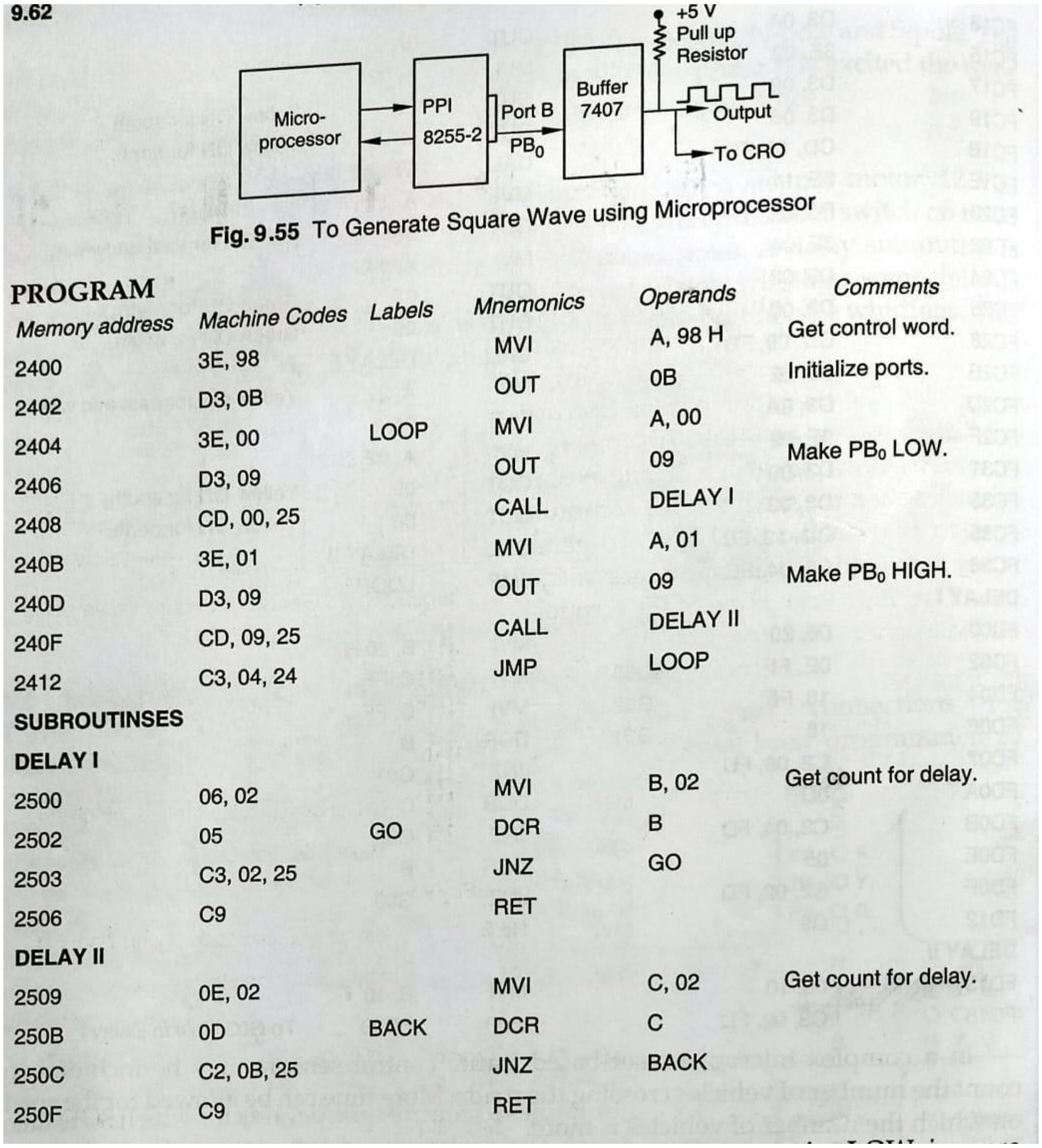
PROGRAM

Memory address	*Machine Codes*	*Lables*	*Mnemonics*	*Operands*	*Comments*
FC00	3E, 80		MVI	A, 80 H	Get control word for 8255.
FC02	D3, 0B		OUT	0B	Initialize ports of 8255.2.
FC04	3E, 01	LOOP	MVI	A, 01	
FC06	D3, 09		OUT	09	Red ON for south.
FC08	D3, 08		OUT	08	Red ON for north.
FC0A	3E, 44		MVI	A, 44	Green ON for east and west.
FC0C	D3, 0A		OUT	0A	
FC0E	CD, 00, FD		CALL	DELAY I	
FC11	3E, 22		MVI	A, 22	Yellow ON for east and west.

```
        D3, 0A              OUT    0A
FC13    3E, 02              MVI    A, 02
FC15    D3, 09              OUT    09         Yellow ON for south
FC17    D3, 08              OUT    08         Yellow ON for north.
FC19    CD, 13, FD          CALL   DELAY II
FC1B    3E, 11              MVI    A, 11
FC1E    D3, 0A              OUT    0A         Red ON for east and west.
FC20    3E, 04              MVI    A, 04
FC22    D3, 08              OUT    08         Green ON for north.
FC24    D3, 09              OUT    09         Green ON for south.
FC26    CD, 00, FD          CALL   DELAY I
FC28    3E, 22              MVI    A, 22      Yellow ON for east and west.
FC2B    D3, 0A              OUT    0A
FC2D    3E, 02              MVI    A, 02
FC2F    D3, 09              OUT    09         Yellow ON for south.
FC31    D3, 08              OUT    08         Yellow ON for north.
FC33    CD, 13, FD          CALL   DELAY II
FC35    C3, 04, FC          JMP    LOOP
FC38
DELAY I
FD00    06, 20              MVI    B, 20 H
FD02    0E, FF        G03   MVI    C, FF
FD04    16, FF        G02   MVI    D, FF
FD06    15            G01   DCR    D
FD07    C2, 06, FD          JNZ    G01
FD0A    0D                  DCR    C
FD0B    C2, 04, FD          JNZ    G02
FD0E    05                  DCR    B
FD0F    C2, 02, FD          JNZ    G03
FD12    C9                  RET
DELAY II
FD13    06, 10              MVI    B, 10
FD15    C3, 02, FD          JMP    FD02       To G03 lable in Delay I
```

sensors may be included to

ProgramforSquarewavegeneratorusing8085 microprocessor



9.62

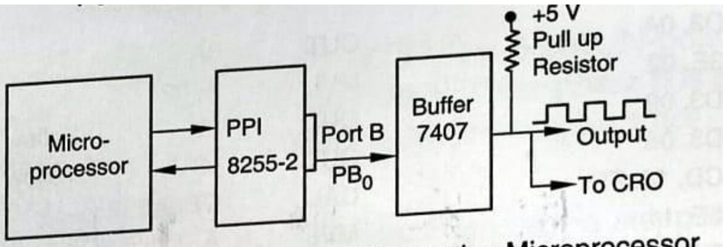


Fig. 9.55 To Generate Square Wave using Microprocessor

PROGRAM

Memory address	*Machine Codes*	*Labels*	*Mnemonics*	*Operands*	*Comments*
2400	3E, 98		MVI	A, 98 H	Get control word.
2402	D3, 0B		OUT	0B	Initialize ports.
2404	3E, 00	LOOP	MVI	A, 00	
2406	D3, 09		OUT	09	Make PB_0 LOW.
2408	CD, 00, 25		CALL	DELAY I	
240B	3E, 01		MVI	A, 01	
240D	D3, 09		OUT	09	Make PB_0 HIGH.
240F	CD, 09, 25		CALL	DELAY II	
2412	C3, 04, 24		JMP	LOOP	
SUBROUTINSES					
DELAY I					
2500	06, 02		MVI	B, 02	Get count for delay.
2502	05	GO	DCR	B	
2503	C3, 02, 25		JNZ	GO	
2506	C9		RET		
DELAY II					
2509	0E, 02		MVI	C, 02	Get count for delay.
250B	0D	BACK	DCR	C	
250C	C2, 0B, 25		JNZ	BACK	
250F	C9		RET		