NUMBERSYSTEMS

2.1Introduction

- There are several kinds of data such as, numeric, text, date, graphics, image, audio and video that need to be processed by a computer. The text data usually consist of standard alphabetic, numeric, and special characters.
- The graphics data consist of still pictures such as drawings and photographs. Any type of sound, including music and voice, is considered as audio data. Video data consistofmotionpictures.
- Thedatahastobeconvertedintoa format that the computer understands. Data can be classified into two forms, analog data and digital data. Analog data can have any value within a defined range and it is continuous.
- Sound waves, telephone signals, temperatures and all other signals that are not broken into bits are examples of analog data. Digital data can be representedbyaseriesofbinarynumbersanditsdiscrete.

The Arithmetic and Logic Unit (ALU) of the computer performs arithmetic and logical operations on data.

Computer arithmetic is commonlyperformedontwodifferenttypesofnumbers,integerand floatingpoint.Asthehardwarerequiredforarithmeticismuchsimpler forintegersthanfloatingpointnumbers,thesetwotypeshaveentirely differentrepresentations.

Anintegerisawholenumberandthe floating-point number has a fractional part. To understand about howcomputersstoredatainthememoryandhowtheyhandlethem, onemustknowaboutbitsandbytesandthenumbersystems.

Bits and bytes are common computer jargons. Both the main memory (Random Access Memory or RAM) and the hard disk capacitiesaremeasuredintermsofbytes.

Theharddiskand memory capacity of a computer and other specifications are described in terms of bits and bytes. For instance, a computer may bedescribedashavinga32-bitPentiumprocessorwith128MegabytesofRAMandharddiskcapacityof40Gigabytes.

BitsandBytes

A numbering system is a way of representing numbers. The most commonly used numbering system is the decimal system. Computer systems can perform computations and transmit data thousands of times faster in binary form than they can use decimal representations.Itisimportantforeveryonestudyingcomputersto knowhowthebinarysystemandhexadecimalsystemwork.

A bit is small piece of data that is derived from the words "**bi**narydigi**t**".Bitshaveonlytwopossiblevalues,0and1.Abinary numbercontainsasequenceof0sand1slike10111.Acollectionof 8bitsiscalledasabyte.With8bitsinabyte,wecanrepresent256 valuesrangingfrom0to255asshownbelow:

0 = 0000 0000
1 = 0000 0001
2 = 0000 0010
3 = 0000 0011
……
……
……

254 = 11111110
255 = 11111111

Bytes are used to represent characters in a text. Different types of coding schemes are used to represent the character set and numbers. The most commonly used coding scheme is the AmericanStandardCodeforInformationInterchange(ASCII).Each binary value between 0 and 127 is used to represent a specific character.TheASCIIvalueforablankcharacter(blankspace)is32 and the ASCII value of numeric 0 is 48. The range of ASCII values for lower case alphabets is from 97 to 122 and the range of ASCII valuesfortheuppercasealphabetsis65to90.

Computermemoryisnormallyrepresentedintermsof Kilobytes orMegabytes. Inmetricsystem,oneKilorepresents1000,thatis, 10^3. Inbinarysystem,oneKilobyterepresents1024bytes,thatis, 2^{10}.Thefollowingtableshowstherepresentationofvariousmemory sizes.

Name	Abbreviation	Size(Bytes)
Kilo	K	2^10*
Mega	M	2^20
Giga	G	2^30
TeraP	T	2^40
etaEx	P	2^50
aZetta	E	2^60
Yotta	Z	2^70
	Y	2^80

*Read as2 power10.

Ina2GB(Gigabytes)storagedevice(harddisk),totally 21,47,483,648bytes can be stored. Nowadays, databases having sizeinTerabytesarereported;Zettaand Yottasizedatabasesare yettocome.

2.3DecimalNumberSystem

Inourdailylife,weuseasystembasedondigitstorepresent numbers.Thesystemthatusesthedecimalnumbersordigitsymbols0 to 9 is called as the decimal number system. This system is said tohaveabase,orradix,often.Sequenceofdigitsymbolsareused to represent numbers greater than 9. When a number is written as a sequence of decimal digits, its value can be interpreted using the positionalvalueofeachdigitinthenumber. Thepositionalnumber system is a system of writing numbers where the value of a digit depends not only on the digit, but also on its placement within a number. Inthepositionalnumbersystem,eachdecimaldigitis weighted relative to its position in the number. This means that eachdigitinthenumberismultipliedbytenraisedtoapower

correspondingto that digit's position.Thus the value of the decimal sequence948is:

$$948_{10_{10}} = 9X\ 10^2 + 4X\ 10^1 + 8\ X\ 10^0$$

Fractional values are represented in the same manner, but theexponentsarenegativefordigitsontherightsideofthedecimal point. Thusthevalueofthefractionaldecimalsequence948.23is:

$$948.23_{10} = 9X10^2 + 4X10^1 + 8X10^0 + 2X10^{-1} + 3X10^{-2}$$

Ingeneral,forthedecimalrepresentationof

X = {**….xxx.x** **2 1 0 -1 -2 -3** **xx..}**

thevalueof**X**is

$$X=S_i x_i 10^i \quad \text{where } i=\ldots.2,1,0,-1,-2,\ldots.$$

2.4 BinaryNumberSystem

Ten different digits 0 – 9 are used to represent numbers in the decimal system. There are only two digits in the binary system, namely,0and1.Thenumbersinthebinarysystemarerepresented to the base two and the positional multipliers are the powers of two. Theleftmostbitinthebinarynumberiscalledasthemostsignificant bit (MSB) and it has the largest positional weight. The rightmost bit istheleastsignificantbit(LSB)andhasthesmallestpositionalweight.

Thebinarysequence10111_2 $_2$ hasthedecimalequivalent:

10111

2

$$= 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$
$$= 16 + 0 + 4 + 2 + 1$$
$$= 23_{10}$$

10

Thedecimalequivalentofthefractionalbinarysequencecanbe estimatedinthesamemanner.Theexponentsarenegativepowersof twofordigitsontherightsideofthebinarypoint.Thebinaryequivalent ofthedecimalpointisthebinarypoint.Thusthedecimalvalueofthe fractionalbinarysequence0.1011_2 is:

$$0.1011_2 = 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4}$$
$$= 0.5 + 0 + 0.125 + 0.0625$$
$$= 0.6875_{10}$$

2.5HexadecimalNumberSystem

Hexadecimal representation of numbers is more efficient in digitalapplicationsbecauseitoccupieslessmemoryspaceforstoring largenumbers. Ahexadecimalnumberisrepresentedusingbase 16. Hexadecimal or Hex numbers are used asa shorthand form of binary sequence. This system is used to represent data in a more compact manner. In the hexadecimal number system, the binary digitsaregroupedintosetsof4andeachpossiblecombinationof4 binarydigitsisgivenasymbolasfollows:

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

Since 16 symbols are used, 0 to F, the notation is called hexadecimal.Thefirst ten symbols are the same as in the decimal system, 0 to 9 and the remaining six symbols are taken from the firstsixlettersofthealphabetsequence,AtoF. Thehexadecimal sequence2C $_{16}$ hasthedecimalequivalent:

$$= 2C_{16} = 2 \times 16^1 + C \times 16^0$$
$$= 32 + 12$$
$$= 44_{10}$$

The hexadecimal representation is more compact than binary representation. It is very easy to convert between binary and hexadecimal systems. Each hexadecimal digit will correspond to fourbinarydigitsbecause2^4 =16.Thehexadecimalequivalentof thebinarysequence110010011101_2 is:

$$1100\ 1001\ 1101 = C9D_{16}$$ (C 9 D)

2.6 DecimaltoBinaryConversion

To convert a binary number to a decimal number, it is required to multiply each binary digit by the appropriate power of 2 and add theresults. Therearetwoapproachesforconvertingadecimal numberintobinaryformat.

2.6.1RepeatedDivisionby2

Any decimal number divided by 2 will leave a remainder of 0 or 1. Repeated division by 2 will leave a string of 0s and 1s that become the binary equivalent of the decimal number. Suppose it is required to convert the decimal number M into binary form, dividing Mby2inthedecimalsystem,wewillobtainaquotientM_1 anda remainder r_1, where r_1 canhaveavalueofeither0or1.

$$\text{ie.,} M = 2 * M_1 + r_1 \qquad r_1 = 0 \text{ or } 1$$

NextdividethequotientM $_1$ by2. ThenewquotientwillbeM $_2$ and thenewremainder r_2.

$$\text{ie.,} M_1 = 2 * M_2 + r_2 \qquad r_2 = 0 \text{ or } 1$$

$$\text{sothat} \quad M = 2 (2 * M_2 + r_2) + r_1$$
$$= 2^2 M_2 + r_2 * 2^1 + r_1 * 2^0$$

NextdividethequotientM $_2$ by2. ThenewquotientwillbeM $_3$ and thenewremainder r_3.

i.e., $M_2 = 2*M + r_3$

so that $M = 2 (2 * (2 * M_3 + r_3) + r_2) + r_1$

$= 2^2(2*M_3 + r_3) + r_2 * 2^1 + r_1 * 2^0$

$= 2^3 M_3 + r_3 * 2^2 + r_2 * 2^1 + r_1 * 2^0$

The above process is repeated until the quotient becomes 0, then

$$M = 1*2^k + r_k * 2^{k-1} + \ldots + r_3 * 2^2 + r_2 * 2^1 + r_1 * 2^0$$

Example:

Convert 23_{10} into its equivalent binary number.

	Quotient	Remainder
23/2	11	1 (LSB)
11/2	5	1
5/2	2	1
2/2	1	0
1/2	0	1 (MSB)

Towritethebinaryequivalentofthedecimalnumber,readthe remaindersfromthebottomupwardas:

$$23_{10}\ 23_{10} = 10111_{2}$$

The number of bits in the binary number is the exponent of the smallestpowerof2thatislargerthanthedecimalnumber.Consider adecimalnumber23.Findtheexponentofthesmallestpowerof2 thatislargerthan23.

$$16 < 23 < 32$$
$$2^4 < 23 < 2^5$$

Hence, the number 23 has 5 bits as 10111. Consider another example.

Findthenumberofbitsinthebinaryrepresentationofthedecimal number36withoutactuallyconvertingintoitsbinaryequivalent.

Thenextimmediatelargenumberthan36thatcanbe representedinpowersof2is64.

$$32 < 36 < 64$$
$$2^5 < 36 < 2^6$$

Hence,thenumber36shouldhave6bitsinitsbinary representation.

2.6.2SumofPowersof2

Adecimalnumbercanbeconvertedintoabinarynumberby addingupthepowersof2andthenaddingbitsasneededtoobtain thetotalvalueofthenumber.Forexample,toconvert 36_{10} tobinary:

a.Findthelargestpowerof2thatissmallerthanorequalto36

$$36 - 32_{10} \quad {}_{10}$$

b. Setthe32'sbitto1andsubtract32fromtheoriginalnumber.

$$36 - 32 = 4$$

c.16isgreaterthantheremainingtotal.Therefore,setthe16'sbit to0
d. 8isgreaterthantheremainingtotal.Hence,setthe8'sbitto0
e.Astheremainingvalueisitselfinpowersof2,set4'sbitto1 andsubtract4

$$4 - 4 = 0$$

Conversioniscompletewhenthereisnothinglefttosubtract. Anyremainingbitsshouldbesetto0. Hence

$$36 = 100100_2$$

Theconversionstepscanbegivenasfollows:

32 16 8 4 2 1
1 36–32=4
32 16 8 4 2 1
1 0 0 1 4 – 4 =0
32 16 8 4 2 1
1 0 01 0 0 36 $_{10}$ =100100

Example:

Convert91 $_{10}$ tobinaryusingthesumofpowersof2method. $_{2}$

Thelargestpowerof2thatissmallerthanorequalto91is64.

64 3216 8 4 2 1
1 91-64 = 27 64 3216 8 4 2 1
1 0 191-(64+16)=11

(Since32>27,setthe32’sbit0and16<27.setthe16’sbit1)

64 3216 8 4 2 1

1 0 1 191-(64+16+8) = 3

64 3216 8 4 2 1

1 0 1 10 1 91-(64+16+8+2) = 1

64 3216 8 4 2 1

1 0 1 10 1 1 91-(64+16+8+2+1) = 0

Hence 91 $_{10}$ = 1011011 $_{2}$

2.7 Conversionoffractionaldecimaltobinary

Thedecimalfractionslike1/2,1/4,1/8etc.,canbeconvertedinto exactbinaryfractions.Sumofpowersmethodcanbeappliedtothese fractions.

$$0.5_{10} = 1*2^{-1} = 0.1_{2}$$

$$0.25_{10} = 0 * 2^{-1} + 1*2^{-2} = 0.01_{2}$$

$$0.125_{10} = 0 * 2^{-1} + 0 * 2^{-2} + 1*2^{-3} = 0.001_{2}$$

Thefraction5/8=4/8+1/8=1/2+1/8hasthebinaryequivalent:

$$5/8 = 1 * 2^{-1} + 0*2^{-2} + 1 * 2^{-3}$$
$$= 0.101_{2}$$

Exactconversionisnotpossibleforthedecimalfractionsthat cannotberepresentedinpowersof2.Forexample,0.2_{10} cannotbe exactlyrepresentedbyasumofnegativepowersof2.Amethodof repeated multiplication by 2 has to be used to convert such kind of decimalfractions.

Thestepsinvolvedinthemethodofrepeatedmultiplicationby2:

· Multiply the decimal fraction by 2 and note the integer part. Theintegerpartiseither0or1.

· Discard the integer part of the previous product. Multiply the fractional part of the previous product by 2. Repeat the first step untilthefractionrepeatsorterminates.

Theresultingintegerpartformsastringof0sand1sthat becomethebinaryequivalentofthedecimalfraction.

Example:

	Integerpart
0.2*2=0.4	0
0.4*2=0.8	0
0.8*2=1.6	1
0.6*2=1.2	1
0.2*2=0.4	0

(Fractionrepeats,theproductis thesameasinthefirststep)

Readtheintegerpartsfromtoptobottomtoobtaintheequivalent fractionalbinarynumber. Hence0.2 $_{10}$ =0.00110011… $_{2}$

2.8 ConversionofDecimaltoHexadecimal

Decimalnumbers'conversiontohexadecimalissimilartobinary conversion. Decimal numbers can be converted into hexadecimal format by the sum of weighted hex digits method and by repeated division by 16. The sum of weighted hex digits method is suitable for small decimal numbers of maximum 3 digits. The method of repeated division by 16 is preferable for the conversion of larger numbers.

The exponent of the smallest power of 16 that is greater than the given decimal number will indicate the number of hexadecimal digitsthatwillbepresentintheconvertedhexadecimalnumber.For example,thedecimalnumber948,whenconvertedintohexadecimal numberhas3hexadecimaldigits.

$$(16^3 = 4096) > 948 > (16^2 = 256)$$

Hence,thehexadecimalrepresentationof948has3hexdigits. Theconversionprocessisasfollows:

16^2	16^1	16^0
3		948–(3*256)=180

16^2 16^1 16^0
3 B 948–(3*256+11*16)=4

16^2 16^1 16^0
3 B 4 948–(3*256+11*16+4)=0

Hence, 948 $_{10}$ =3B4 $_{16}$

Thestepsinvolvedintherepeateddivisionby16toobtainthe hexadecimalequivalentareasfollows:

· Dividethedecimalnumberby16andnotetheremainder. Expresstheremainderasahexdigit.

· Repeattheprocessuntilthequotientiszero

Example:

Process quotient remainder

948 / 16 =59 4 (LSB)

59 / 16= 3 11(B)

3 / 16= 0 3 (MSB)

$948_{10} = 3B4_{16}$

2.9 OctalRepresentation

Anoctalnumberisrepresentedusingbase8. Octal representation is just a simple extension of binary and decimal representations but using only the digits 0 to7. To convert an octal number to a decimal number, it is required to multiply each octal digitbytheappropriatepowerof8andaddtheresults.

8

10

2.10.1Sign+magnituderepresentation

The simplest form of representing a negative integer is the sign+magnituderepresentation.Inasequenceofnbits,theleftmost bit is used for sign and the remaining n-1 bits are used to hold the magnitudeoftheinteger. Thusinasequenceof4bits,

0100 = +4
1100 = -4

As there are several drawbacks in this representation, this method has not been adopted to represent signed integers. There aretworepresentationsfor0inthisapproach.

0000 = +0 10
1000 = -0 10

Henceitisdifficulttotestfor0,whichisanoperation,performed frequently in computers. Another drawback is that, the addition and subtraction require a consideration of both the sign of the numbers and their relative magnitude, in order to carry out the required operation. This would actually complicate the hardware design of thearithmeticunitofthecomputer. Themostefficientwayof representingasignedintegerisa2's-complementrepresentation. In2'scomplementmethod,thereisonlyonerepresentationof0.

2.10.2. 2's-complementrepresentation

Thismethoddoesnotchangethesignofthenumberbysimply changingasinglebit(MSB)initsrepresentation.The2's-complement methodusedwith-venumbersonlyisasfollows:

a. Invertallthebitsinthebinarysequence(ie.,changeevery0 to1andevery1to0ie.,1'scomplement)
b. Add1totheresult

Thismethodworkswellonlywhenthenumberofbitsusedbythe systemisknownintherepresentationofthenumber.Careshouldbe

takentopad(fillwithzeros)theoriginalvalueouttothefullrepresentation widthbeforeapplyingthisalgorithm.

Example:

Inacomputerthatuses8-bitrepresentationtostoreanumber,the wrongandrightapproachestorepresent–23areasfollows:

Wrongapproach:

Thebinaryequivalentof23is10111. Invertallthebits=>01000
Add1totheresult=>01001
Padwithzerostomake8-bitpattern=>00001001=>+9

Rightapproach:

Thebinaryequivalentof23is10111
Padwithzerostomake8-bitpattern=>00010111
Invertallthebits=> 11101000
Add1totheresult=>11101001=>-23

2.10.3Manualmethodtorepresentsignedintegersin2's complementform

Thisisaneasierapproachtorepresentsignedintegers.Thisisfor-venumbersonly.

Step1:Copythebitsfromrighttoleft,throughandincludingthefirst1.

Step2:Copytheinverseoftheremainingbits.

Example1:

Torepresent–4ina4-bitrepresentation:

Thebinaryequivalentoftheinteger4is0100

Asperstep1,copythebitsfromrighttoleft,throughandincludingthe first1 =>100

Asperstep2,copytheinverseoftheremainingbits=>1100=>-4

Example2:

Torepresent–23ina8-bitrepresentation:

Thebinaryequivalentof23is00010111

As per step 1: 1

Asperstep2:11101001=>-23

2.10.4 Interpretationofunsignedandsignedintegers

Signed number versus unsigned number is a matter of interpretation.Asinglebinarysequencecanrepresenttwodifferent values. Forexample,considerabinarysequence11100110. $_{2}$

The decimal equivalent of the above sequence when consideredasanunsignedintegeris:

11100110 $_{2}$ =230 $_{10}$

Thedecimalequivalentofthesequencewhenconsideredas asignedintegerin2’scomplementformis:

11100110 $_{2}$ =-26 $_{10}$ (after2’scomplementandaddnegativesign).

Whencomparingtwobinarynumbersforfindingwhichnumber is greater, the comparison depends on whether the numbers are consideredassignedorunsignednumbers.

Example:

X=1001
Y=0011
Is(X>Y)/*Isthistrueorfalse?*/
ItdependsonwhetherXandYareconsideredassignedorunsigned. IfXandYareunsigned: XisgreaterthanY
IfXandYaresigned: Xislessthan Y.

2.10.5Rangeofunsignedandsignedintegers

Ina4-bitsystem,therangeofunsignedintegersisfrom0to 15,thatis,0000to1111inbinaryform.Eachbitcanhaveoneoftwo values 0 or 1. Therefore, the total number of patterns of 4 bits will be 2 X 2 X 2 X 2 = 16. In an n-bit system, the total number of patternswillbe2^n.Hence,ifnbitsareusedtorepresentanunsigned integervalue,therangeisfrom0to2^n-1,thatis,thereare2^ndifferent values.

Incaseofasignedinteger,themostsignificant(leftmost)bit is used to represent a sign. Hence, half of the 2^npatterns are used forpositivevaluesandtheotherhalffornegativevalues.Therange ofpositivevaluesisfrom0to2^{n-1}-1andtherangeofnegativevalues isfrom–1to–2^{n-1}.Ina4-bitsystem,therangeofsignedintegersis from–8to+7.

2.11BinaryArithmetic

Digitalarithmeticusuallymeansbinaryarithmetic. Binary arithmeticcanbeperformedusingbothsignedandunsignedbinary numbers.

2.11.1BinaryAddition–Unsignednumbers

Whentwodigitsareadded,iftheresultislargerthanwhatcanbe containedinonedigit,acarrydigitisgenerated.Forexample,ifwe add 5 and 9, the result will be 14. Since the result cannot fit into a singledigit,acarryisgeneratedintoaseconddigitplace.Whentwo bitsareaddeditwillproduceasumbitandacarrybit.Thecarrybit maybezero.

Example:

0+0=00
0+1=0 1

carrybit sum bit

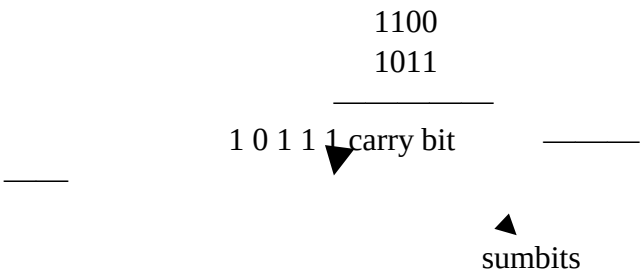
1 + 1 = 1 0

carrybit sum bit

Thesumbitistheleastsignificantbit(LSB)ofthesumoftwo 1-bitbinarynumbersandthecarrybitholdsthevalueofcarry(0or 1)resultingfromtheadditionoftwobinarynumbers.

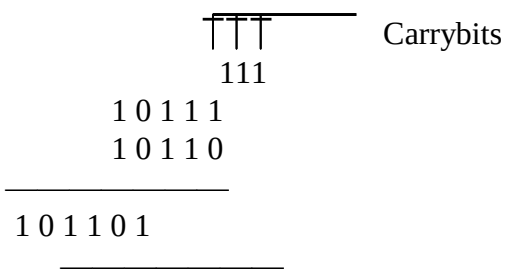
Example1:

Calculatethesumofthenumbers,1100and1011:



Example2:

Calculate10111+10110



In unsigned binary addition, the two operands are called augendandaddend. Anaugendisthenumberinanaddition operationtowhichanothernumberisadded. Anaddendisthe numberinanadditionoperationthatisaddedtoanother.

2.11.2Binaryaddition–signednumbers

Signedadditionisdoneinthesamewayasunsignedaddition. The only difference is that, both operands must have the same number of magnitude bits and each must have a sign bit. As we havealreadyseen,inasignednumber,themostsignificantbit(MSB) is a sign bit while the rest of the bits are magnitude bits. When the numberisnegative,thesignbitis1andwhenthenumberispositive, thesignbitis0.

Example1:

Add+2 $_{10}$ and+5 $_{10}$. Writetheoperandsandthesumas4-bitsigned binarynumbers.

+2 0010
+5 0101
——— ———————
+7 0111
——— ———————

magnitudebits signbit

Iftheresultoftheoperationispositive,wegetapositivenumberin ordinarybinarynotation.

Example2: (Useof2'scomplementinsignedbinaryaddition)

Add–7 $_{10}$ +5 $_{10}$ using4-bitsystem.

In2'complementform,-7isrepresentedasfollows:

Inbinaryform,7isrepresentedas: 0111

Invert the bits (1 to 0 and 0 to 1) 1 0 0 0

Add1 1

Hence,-7in2'scomplementformis 1001 (-7)

+0101(5)
———————
1110(-2)
———————

Iftheresultoftheoperationisnegative,wegetanegativenumber in2'scomplementform.Insomecases,thereisacarrybitbeyondthe endofthewordsizeandthisisignored.

Example3:

Add -4 $_{10}$ +4 $_{10}$. Use4-bitsystem.

1100 (-4in2'scomplementform)

0100 (+4)

1 0000 =0

In the above example, the carry bit goes beyond the end of the word and this can be ignored. In this case both operands are having different signs. There will be no error in the result. On any addition, the result may be larger than can be held in the word size beingusedandthiswouldresultinoverflow.

Theoverflowconditionisbasedontherule:

If two numbers are added and if they are either positive or negative,thenoverflowoccursifandonlyiftheresulthastheopposite sign.

Example4:

Add (-7 $_{10}$)+(-5 $_{10}$) usingthewordsize4.

1 00 1 (-7 in 2's complement form)

1 01 1 (-5 in 2's complement form)

1 0 10 0 (The result is wrong)

Intheaboveexamplebothoperandsarenegative.ButtheMSB oftheresultis0thatistheresultispositive(oppositesign)andhence overflowoccursandtheresultiswrong.

2.11.3BinarySubtraction

Subtrahendandminuendarethetwooperandsinanunsigned binarysubtraction. Theminuendisthenumberinasubtraction operationfromwhichanothernumberissubtracted.Thesubtrahend isthenumberthatissubtractedfromanothernumber.Simplebinary subtractionoperationsareasfollows:

0–0	=	0
1–0	=	1
1–1	=	0
10–1	=	1

Whensubtracting1from0,borrow1fromthenextmostsignificant bit(MSB).Whenborrowingfromthenextmostsignificantbit,ifitis1, replaceitwith0.Ifthenextmostsignificantbitis0,youmustborrow fromamoresignificantbitthatcontains1andreplaceitwith0andall 0suptothatpointbecome1s.

Example1:

Subtract 1101–1010

borrow

01 ◀

1101 (minuend)

-1010 (subtrahend)

0011

When subtracting the 2nd least significant bit (1 in the subtrahend) from 0 (in the minuend), a 1 is borrowed from the more significant bit (3rd bit from right in the minuend) and hence 10 – 1 = 1. The 3rd least significant bit is made as 0.

Example2:

Subtract 1000 – 101

```
 011        after borrowing, the minuend will become
 1 0 0 0            0 11 10
 - 1 0 1            1  0 1 (subtrahend)
 -------
 00 11      difference as per the basic operations for subtraction
```

To subtract one number (subtrahend) from another (minuend), take the 2's complement of the subtrahend and add it to the minuend.

Example3:

Subtract (+2) – (+7) using 4-bit system

```
  00 10  (+2)

  01 11  (+7)

1001 (-7 in 2's complement form)

    00 10   (2)
 +  1 001   (-7)
 ------------
    10 11   (-5)
 ------------
```

Example4:

Subtract(-6)–(+4)using4bitsystem

Minuend -6 1 01 0

2'scomplementoftheSubtrahend-4 1 10 0

10110

Both numbers are represented as negative numbers. While adding them, the result will be :10110. As the word size is 4, the carry bit goes beyond the end of the word and the result is positive astheMSBis0.Thiscaseleadstooverflowandhencetheresultis wrong. Theoverflowruleworksinsubtractionalso.

2.12 Booleanalgebra

Boolean algebra is a mathematical discipline that is used for designingdigitalcircuitsinadigitalcomputer.Itdescribestherelation between inputs and outputs of a digital circuit. The name Boolean algebrahasbeengiveninhonorofanEnglishmathematicianGeorge Boolewhoproposedthebasicprinciplesofthisalgebra.Aswithany algebra, Boolean algebra makes use of variables and operations (functions).ABooleanvariableisavariablehavingonlytwopossible valuessuchas,**trueorfalse,**oras,**1or0**. Thebasiclogical operations are**AND**,**OR**and**NOT**,whicharesymbolically representedbydot,plussign,andbyoverbar/singleapostrophe.

Example:

A AND B =A.B A OR B

=A+B

NOT A =A' (orA) $\overline{A}$

ABooleanexpressionisacombinationofBooleanvariables, Boolean Constants and the above logical operators. All possible operationsinBooleanalgebracanbecreatedfromthesebasiclogical operators. There are no negative or fractional numbers in Boolean algebra.

TheoperationANDyieldstrue(binaryvalue1)ifandonlyifbothof itsoperandsaretrue. TheoperationORyieldstrueifeitherorbothof itsoperandsaretrue.TheunaryoperationNOTinvertsthevalueofits operand.Thebasiclogicaloperationscanbedefinedinaformknown asTruthTable,whichisalistofallpossibleinputvaluesandtheoutput responseforeachinputcombination.

2.12.1Booleanoperators(functions) ANDoperator

TheANDoperatorisdefinedinBooleanalgebrabytheuseofthe dot(**.**)operator.Itissimilartomultiplicationinordinaryalgebra.The ANDoperatorcombinestwoormoreinputvariablessothattheoutput istrueonlyifalltheinputsaretrue.Thetruthtablefora2-inputAND operatorisshownasfollows:

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Theabove2-inputANDoperationisexpressedas:**Y=A.B ORoperator**

TheplussignisusedtoindicatetheORoperator.TheORoperator combines two or more input variables so that the output is true if at least one input is true. The truth table for a 2-input OR operator is shownasfollows:

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Theabove2-inputORoperationisexpressedas: **Y=A+B**

NOToperator

The NOT operator has one input and one output. The input is eithertrueorfalse,andtheoutputisalwaystheopposite,thatis,the NOT operator inverts the input. The truth table for a NOT operator whereAistheinputvariableandYistheoutputisshownbelow:

A	Y
0	1
1	0

TheNOToperatorisrepresentedalgebraicallybytheBoolean expression: $\mathbf{Y=\overline{A}}$

Example:ConsidertheBooleanequation: D = A + (B . C)

Disequalto1(true)ifAis1orif(B.C)is$\overline{1}$,thatis,B=0andC=
1. OtherwiseDisequalto0(false).

ThebasiclogicfunctionsAND,OR,andNOTcanalsobe combinedtomakeotherlogicoperators.

NANDoperator

The NAND is the combination of NOTandAND.The NAND is generatedbyinvertingtheoutputofanANDoperator. Thealgebraic expressionoftheNANDfunctionis:

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

TheNANDfunctiontruthtableisshownbelow:

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

A NAND B = NOT (A AND B)

NORoperator

The NOR is the combination of NOT and OR. The NOR is generated by inverting the output of an OR operator. The algebraic expressionoftheNORfunctionis:

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

TheNORfunctiontruthtableisshownbelow:

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

A NOR B = NOT (A OR B)

2.12.2LawsofBooleanalgebra

Boolean algebra helps to simplify Boolean expressions in order to minimize the number of logic gates in a digital circuit. You willstudyaboutlogicgatesintheforthcomingchapter.Thischapter focuses on the theorems of Boolean algebra for manipulating the Booleanexpressionsinordertosimplifythem.

BooleanIdentities

LawsofComplementation

Thetermcomplementsimplymeanstochange1sto0sand0sto1s. Theorem1 : IfA=0, thenA=1

Theorem2 : IfA=1, thenA=0 $\overline{}$

Theorem3 : Thecomplementtocomplement$\overline{\text{of}}$AisAitself.

A = A$\overline{\overline{}}$**Basic properties of**

AND operator Theorem4 : A **.** 1 = A

IfAequals0andtheotherinputis1,theoutputis0.
IfAequals1andtheotherinputis1,theoutputis1.

ThustheoutputisalwaysequaltotheAinput. Theorem5 : A **.**0 = 0

Asoneinputisalways0,irrespectiveofA,theoutputisalways0.

Theorem6 : A.A=A

TheoutputisalwaysequaltotheAinput. Theorem7 : A .A = 0

RegardlessofthevalueofA,theoutputis0.

BasicpropertiesofORoperator

Theorem8 : A+1 = 1

IfAequals0andtheotherinputis1,theoutputis1.
IfAequals1andtheotherinputis1,theoutputis1.
Thustheoutputisalwaysequalto1regardlessofwhatvalueA
takeson.

Theorem9 : A +0 = A

TheoutputassumesthevalueofA. Theorem10 :A+A=A

TheoutputisalwaysequaltotheAinput. Theorem11 :A +A = 1

RegardlessofthevalueofA,theoutputis1.

2.12.3SimplificationofBooleanexpressions

Before seeing the important theorems used in the simplification of Boolean expressions, some Boolean mathematical conceptsneedtobeunderstood.

Literal

Aliteralistheappearanceofavariableoritscomplementina Booleanexpression.

ProductTerm

AproductterminaBooleanexpressionisatermwhereoneor moreliteralsareconnectedbyANDoperators.Asingleliteralisalsoa productterm.

Example: AB,AC,AC,andEaretheproductterms.

Minterm

Amintermisaproductterm,whichincludesallpossiblevariables eithercomplementedoruncomplemented. InaBooleanexpression of3variables,x, y,andz,thetermsxyz,xyz,andxyzareminterms. Butxyisnotaminterm.Mintermisalsocalledasastandardproduct term.

Sumterm

AsumterminaBooleanexpressionisatermwhereoneormore literalsareconnectedbyORoperators.

Example: A+B+D

Maxterm

A maxtermis a sum term in a Boolean expression, which includesallpossiblevariablesintrueorcomplementform. Ina Boolean expression of 3 variables, x, y, and z, the terms x + y + z, andx+y+zarethemaxterms.Maxtermisalsocalledasstandard sumterm.

Sum-of-products(SOP)

Asumofproductsexpressionisatypeof Booleanexpression whereoneormoreproducttermsareconnectedbyORoperators.

Example: $A+AB+AB\overline{C}$ $\overline{\quad\quad\quad}$

Inanexpressionof3variables,A,B,andC,theexpression $\overline{ABC}+\overline{ABC}+\overline{ABC}$ isalsocalledasacanonicalsumorsumof standardproducttermsorsumofminterms.

Product-of-sums(POS)

ProductofsumsisatypeofBooleanexpressionwhereseveral sumtermsareconnectedbyANDoperators.

Example: $(A+B)(A+B)(A+B)$ $\overline{\quad\quad\quad}$

A canonical product or product of standard sum terms is a productofsumsexpressionwhereallthetermsaremaxterms.The above example is a canonical product in a Boolean expression of twovariablesAandB.

Theorem12:CommutativeLaw

Amathematicaloperationiscommutativeifitcanbeappliedto itsoperandsinanyorderwithoutaffectingtheresult.

Additionandmultiplicationoperationsarecommutative.

Example:

$$A + B = B + A \quad AB = BA$$

Subtractionisnotcommutative:

$$A - B \neq B - A$$

ThereisnosubtractionoperationinBooleanalgebra.

Theorem13: AssociativeLaw

Amathematicaloperationisassociativeifitsoperandscanbe groupedinanyorderwithoutaffectingtheresult.Inotherwords,the orderinwhichonedoestheORoperationdoesnotaffecttheresult.

$(A + B) + C = A + (B + C) = (A + C) + B$

Similarly,theorderinwhichonedoestheANDoperationdoes notaffecttheresult.

$(AB)C = A(BC) = (AC)B$

Theorem14: DistributiveLaw

ThedistributivepropertyallowsustodistributeanANDacross severalORfunctions.

Example:

$A(B + C) = AB + AC$

Thefollowingdistributivelawisworthnotingbecauseitdiffers fromwhatwewouldfindinordinaryalgebra.

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

Thesimplestwaytoprovetheabovetheoremistoproducea truth table for both the right hand side (RHS) and the left hand side (LHS)expressionsandshowthattheyareequal.

A	B	C	BC	LHS	A+B	A+C	RHS
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

MinimumSumofProducts

AminimumsumofproductsexpressionisoneofthoseSumof Products expressions for a Boolean expression that has the fewest numberofterms.

ConsiderthefollowingBooleanExpression:

$$A\ B\ C + \bar{A}\ B\ \bar{C} + A\ \bar{B}\ C + A\ B\ \bar{C} + A\ B\ \bar{C}$$

UsingAssociativityLaw

$$= (A\ B\ C + \bar{A}\ B\ \bar{C}) + (\bar{A}\ B\ C + A\ B\ \bar{C}) + A\ B\ \bar{C}$$

$$=AB(C+C)+\bar{A}B(\bar{C}+C)+AB\bar{C}$$ UsingTheorem11

$$=AB(1)+A\bar{B}(1)+ABC$$ UsingTheorem4

$$=AB+AB+ABC$$

$\bar{\ }$ $\bar{\ }$

Theaboveexpressionisintheminimumsumofproductsform. The given Boolean expression can be rewritten as follows using theorem10.

$ABC+AB\overline{C}+A\overline{B}C+\overline{A}BC+AB\overline{C}+\overline{A}BC$ $(ABC+ABC=AB\overline{C})$

$=(ABC+AB\overline{C})+(\overline{A}BC+ABC)+(AB\overline{C}+ABC)$

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

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

The same Boolean expression can be simplified into many minimumsumofproductsform.

Examples:

SimplifythefollowingBooleanExpression

$A\ B\ C + \overline{A}\ B\ \overline{C}$ Let $\overline{x} = A\ B$ and y

$= C$

TheaboveBooleanexpressionbecomes $x\ y + x\ y$

$= x(y + y)$

$= x = A\ B$

Prove thatA+AB=A+B

AccordingtoDistributiveLaw

$A+AB=(A+A)(\overline{A}+B) =1\cdot(\overline{A}+B)=A+B$

SimplifythefollowingBooleanExpression

$\bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C}$

$= AC(B + \bar{B}) + \bar{A}B\bar{C} + A\bar{B}C$

$= AC + AB\bar{C} + A\bar{B}C$

$=A(C+BC)+\bar{A}B\bar{C}$

$= A(C + B)(\bar{C} + C) + \bar{A}BC$

$= A(C + B) + \bar{A}BC$

$=AC+AB+\bar{A}B\bar{C}$ (oneminimalform)

InthegivenBooleanExpression,ifthesecondandthirdterms aregrouped,itwillgive

$ABC+(A\underline{B}\underline{C}+\underline{A}BC)+\underline{A}B\underline{C}$

$= AB\bar{C} + \bar{A}B(\bar{C} + C) + \bar{A}BC$

$= AB\bar{C} + \bar{A}B + \bar{A}B\bar{C}$

$= BC(A + \bar{A}) + \bar{A}B$

$=BC+\bar{A}B$ (mostminimalform)

2.12.4 DeMorgan'sTheorems Theorem 15: $A + B = \overline{A + B}$

Theorem 16: $AB = \bar{A} + \bar{B}$

The above identities are the most powerful identities used in Booleanalgebra.Byconstructingthetruthtables,theaboveidentities canbeprovedeasily.

Example:

GivenBooleanfunctionf(A,B,C,D)=DAB+AB+DAC̄,Findthe complementoftheBooleanfunction

f(A,B,C,D̄)=DAB+AB+DAC̄

ApplyDeMorgan'sLaw(theorem15)

=(DAB)(AB) (D̄AC)

ApplyDeMorgan'sLaw(theorem16)

= (D + A + B)(Ā + B)(D̄ + Ā + C)

Intheaboveproblem,thegivenBooleanfunctionisinthesum ofproductsformanditscomplementisintheproductofsumsform.

The DeMorgan'stheorem says that any logical binary expressionremainsunchangedifwe,

changeallvaribalestotheircomplements
changeallANDoperationstoORoperations
changeallORoperationstoANDoperations
takethecomplementoftheentireexpression

A practicaloperationalwaytolookatDeMorgan'stheorem isthattheinversionofanexpressionmaybebrokenatanypointand theoperationatthatpointreplacedbyitsoppostie(i.e.,ANDreplaced byORorviceversa)

UNIT - II

COMBINATIONAL VS SEQUENTIAL

A **combinational** circuit:

- At any time, outputs depends only on inputs
- Changing inputs changes outputs
- No regard for previous inputs
- No memory (history)
- Time is ignored!

A **sequential** circuit:

- A combinational circuit with **feedback** through **memory**
- The stored information at any time defines a **state**
- Outputs depends on inputs and previous inputs
- Previous inputs are stored as binary information into memory
- Next state depends on inputs and present state

Half Adder

The circuit that performs addition within the Arithmetic and Logic Unit of the CPU are called adders. A unit that adds two binary digits is called a half adder and the one that adds together three binary digits is called a full adder.

A half adder sums two binary digits to give a sum and a carry. This simple addition consists of four possible operations.

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 1

The first three operations produce a single digit sum, while the fourth one produces two digit sum. The higher significant bit in this operation is called a carry. So the carry of the first three operations are '0', where the fourth one produces a carry '1'.

The boolean realization of binary addition is shown in the truth table. Here A and B are inputs to give a sum S and a carry C.

Input		Sum	Minterms	Carry	Minterms
A	B	S	Of S	C	of C
0	0	0	-	0	
0	1	1	A.B	0	
1	0	1	0 A.B	0	
1	1	0		1	0 A.B

The boolean functions corresponding to the sum and carry are

$$S = A \cdot B + A \cdot \overline{B} \quad C = A \cdot B$$

Which can be realized using logic circuit as,

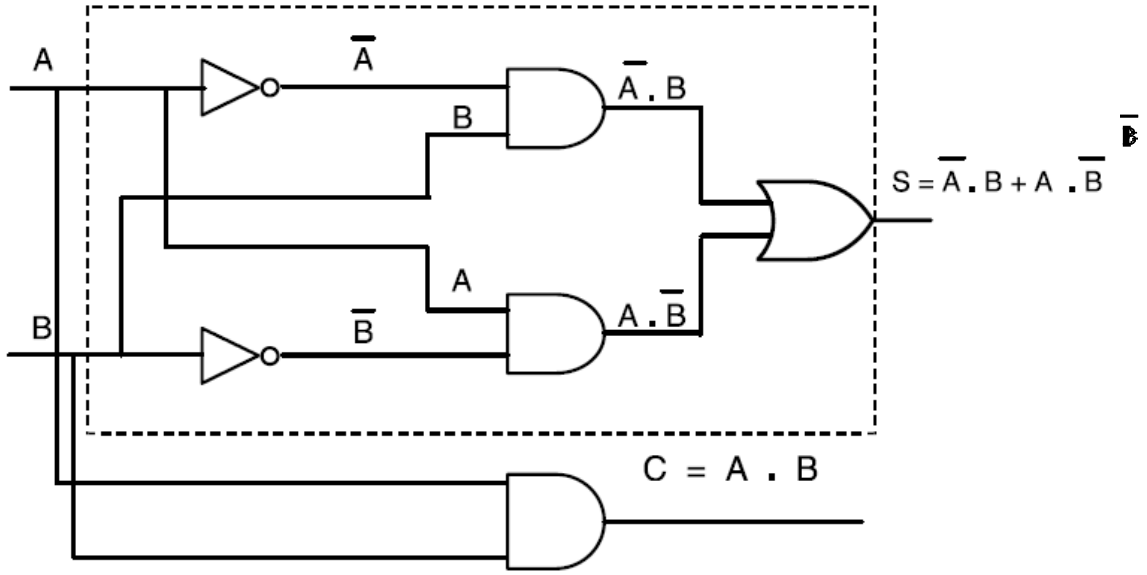
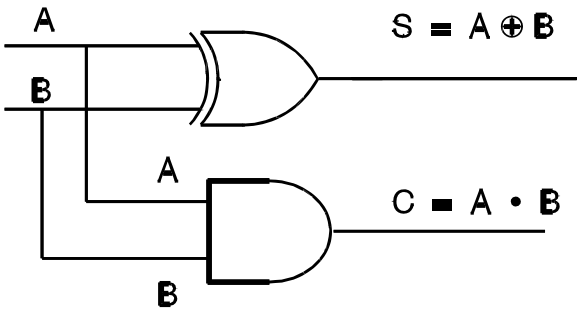


Fig. 4.19 Logic Circuit of Half Adder

which is further simplified as



In a half adder, an AND gate is added in parallel to the XOR gate to generate the carry and sum respectively. The 'sum' column of the truth table represents the output of the XOR gate and the
'carry' column represents the output of the AND gate.

Full Adder

A half adder logic circuit is a very important component of computing systems. As this circuit cannot accept a carry bit from a previous addition, it is not enough to fully peform additions for binary number greater than 1. In order to achieve this a full adder is required.

A full adder sums three input bits. It consists of three inputs and two outputs. Two of the inputs represent the two significant bits to be added and the third one represents the carry from the previous significant position.

Here A, B referred as the inputs, C1 as carry input from the previous stage, C2 as carry output

and S as sum.

Input			Output	
A	B	C	C	S
0	1 0	0 2	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

$$S = \bar{A}\,\bar{B}\,C_1 + \bar{A}\,B\,\bar{C_1} + A\,\bar{B}\,\bar{C_1} + A\,B\,C_1$$

$$C_2 = \bar{A}\,B\,C_1 + A\,\bar{B}\,C_1 + A\,B\,\bar{C_1} + A\,B\,C_1$$

Consider

$$(A \oplus B) \oplus C_1 = (A\bar{B} + \bar{A}B) \oplus \bar{C_1}$$

$$= (A\bar{B} + \bar{A}B)\,\bar{C_1} + (A\bar{B} + \bar{A}B)\,\bar{C_1}$$

$$= ((A + B)(\bar{A} + \bar{B}))\,\bar{C_1} + (A\bar{B} + \bar{A}B)\,C_1$$

$$= ((A + B)(\bar{A}\bar{B}))\,\bar{C_1} + (A\bar{B} + \bar{A}B)\,C_1$$

$$= A\bar{A}\bar{C_1} + AB\bar{C_1} + \bar{A}\bar{B}\bar{C_1} + B\bar{B}\bar{C_1} + \bar{A}B\bar{C_1} + A\bar{B}\bar{C_1}$$

(Since, $A\bar{A} = B\bar{B} = 0$)

$$= S$$

Also

$$C_2 = \bar{A}BC_1 + A\bar{B}C_1 + AB\bar{C_1} + ABC_1$$

$$= (\bar{A}B + A\bar{B})C_1 + AB(C_1 + \bar{C_1})$$

$$= (A \oplus B)\,C_1 + AB$$

Hence

$$S = (A \oplus B) \oplus C_1 \text{ and}$$

$$C_2 = (A \oplus B)\, C_1 + AB$$

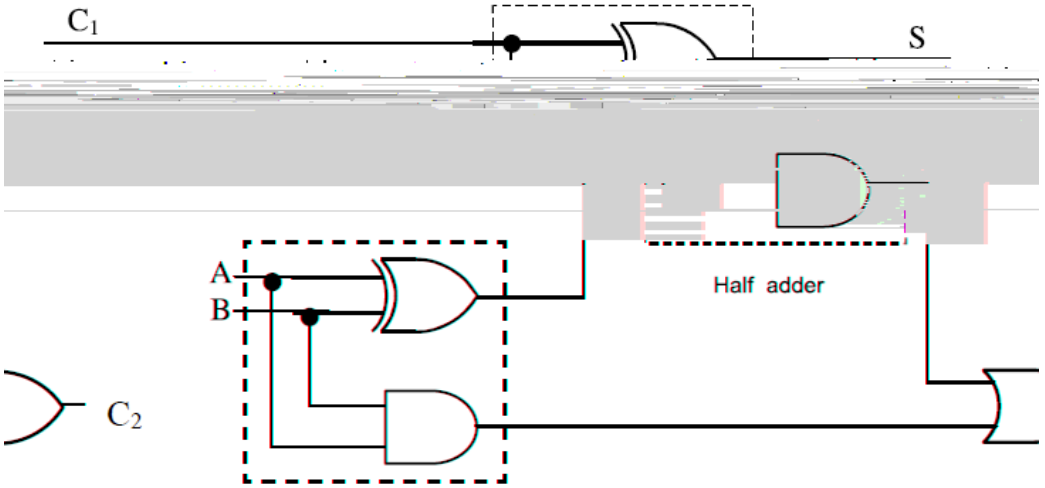
To realize the full adder we need two 2-input XOR, two 2- input AND gates and a 2-input OR gate.

Hence the full adder can be realized as.

Notice that the full adder can be constructed from two half adders and an OR gate.

If the logic circuit outputs are based on the inputs presented at that time, then they are called combinational circuit. The half adder and full adder circuits are the examples for the combinational circuits. On the other hand, if the logic circuit outputs are based on, not only the inputs presented at that time, but also the previous state output, then they are called sequential circuits.

There are two main types of sequential circuits. A synchronous sequential circuit is a system whose output can be defined from its inputs at discrete instant of time. The output of the asynchronous sequential circuit depends upon the order in which its input signals change at any instance of time. The flip-flop circuit is an example of sequential circuit.



The Flip-Flop

A flip flop is a circuit which is capable of remembering the value which is given as input. Hence it can be used as a basic memory element in a memory device. These circuits are capable of storing one bit of information.

Basic flip-flops

A flip-flop circuit can be constructed using either two NOR gates or two NAND gates.

A common example of a circuit employing sequential logic is the flip-flop, also called a bi-stable

gate. A simple flip-flop has two stable states. The flip-flop maintains its states indefinitely until an input pulse called a trigger is received. If a trigger is received, the flip-flop outputs change their states according to defined rules, and remain in those states until another trigger is received.

Flip – Flop Circuit using NOR Gates

By cross-coupling two NOR gates, the basic operation of a flip-flop could be demonstrated. In this circuit the outputs are fed back again to inputs.

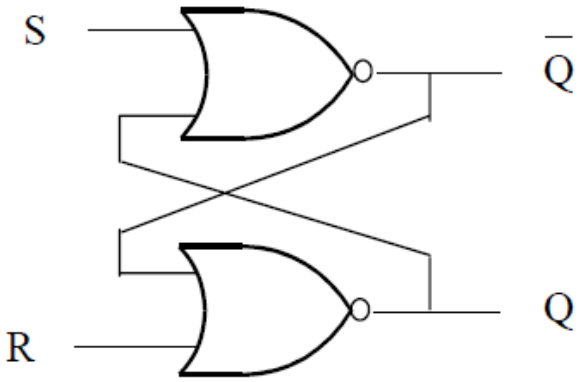


Fig. 4.21 Flip Flop Circuit using NOR Gates

The flip-flop circuit has two outputs, one for the normal value Q and another for the complement value $\overline{Q}$.

It also has two inputs S (set) and R (reset). Here, the previous output states are fed back to determine the current state of the output.

The NOR basic flip-flop circuit operates with inputs normally at '0' unless the state of the flip-flop has to be changed.

As a starting point, we assume S = 1 and R = 0. This makes Q = 0. This Q = 0 is again given along with R = 0 to make Q = 1.

ie. when S = 1 and R = 0 make Q = 1 and $\overline{Q}$ = 0.

When the input S returns to '0', the output remains the same, because the output Q remain as '1' and $\overline{Q}$ as '0'.

We assume S = 0 and R = 1. This make Q = 0. This Q = 0 is again given along with S = 0 to make Q = 1.

ie. when S = 0 and R = 1 make Q = 0 and $\overline{Q}$ = 1.

When the reset input returns to 0, the outputs do not change, because the output Q remains as '0' and $\overline{Q}$ as '1'.

ie. when S = 0 and R = 0 make Q = 0 and $\overline{Q}$ = 1 after S = 0 and R=1.

S	R	Q	$\overline{Q}$	
1	0	1	0	
0	0	1	0	(after S =1 and R = 0)
0	1	0	1	
0	0	0	1	(after S =0 and R = 1)

When '1' is applied to both S and R, the outputs Q and $\overline{Q}$ become 0.
These facts violate the output Q and $\overline{Q}$ are the complements of each other.
In normal operations this condition mustbe avoided.
Thus a basic flip-flop has two useful states.
When Q = 1 and $\overline{Q}$ = 0, it is called as set state. When Q = 0 and $\overline{Q}$ = 1, it is called as reset state.

Flip – Flop Circuit using NAND Gates

In a similar manner one can realize the basic flip-flop by cross coupling two NAND gates.

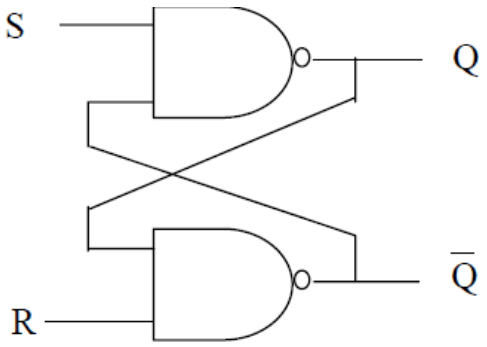


Fig. 4.22 Flip Flop Circuit using NAND Gates

The corresponding truth table is given as

S	R	Q	$\overline{Q}$	
1	0	0	1	
1	1	0	1	(after S =1 and R = 0)
0	1	1	0	
1	1	1	0	(after S =0 and R = 1)

The NAND basic flip-flop circuit operates with inputs normally at '1' unless the state of the flip-flop has to be changed. A momentary '0' to the input S gives Q = 1 and Q = 0. This makes the flip-flop to set state.

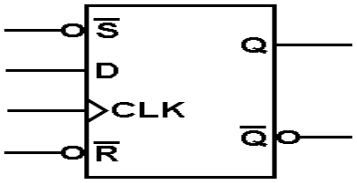
After the input S returns to 1, a momentary '0' to the input R gives Q = 0 and Q = 1.

This makes the flip-flop to reset state.

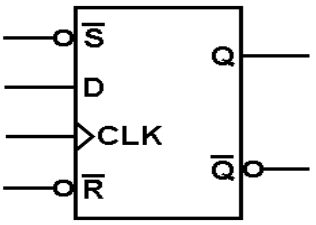
When both the inputs become 0, ie., S = 0 and R = 0, both the outputs become 1. This condition must be avoided in the normal operation

There are several kinds of flip-flop circuits, with designators such as D, T, J-K, and R-S. Flip-flop circuits are interconnected to form the logic gates that comprise digital integrated circuits (ICs) such as memory chips and microprocessors.

D-type Flip-Flops



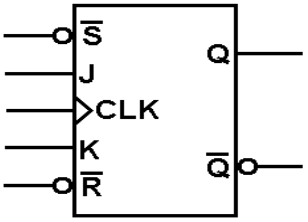
- Have following inputs:
  - D
  - Clock (CLK)
  - S
  - R
- Have following outputs
  - Q
  - Q
  - On clock edge, the value on D is transferred to Q and stays there



R and S are used to put device into known state

JK Flip-flops

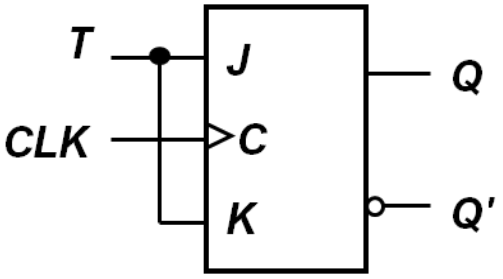
- Operation similar to D-type except has two inputs J and K
- When J is HIGH, flip-flop is SET
- When K is HIGH, flip-flop is RESET
- If both J and K are high, output simply TOGGLES

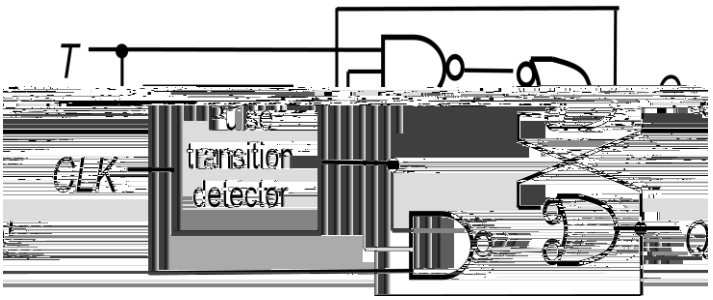


J	K	Next State of Q
0	0	No change
0	1	0
1	0	1
1	1	Toggle

T Flip-flop

- **T flip-flop: single-input version of the J-K flip flop, formed by tying both inputs together.**
- **Characteristic table.**





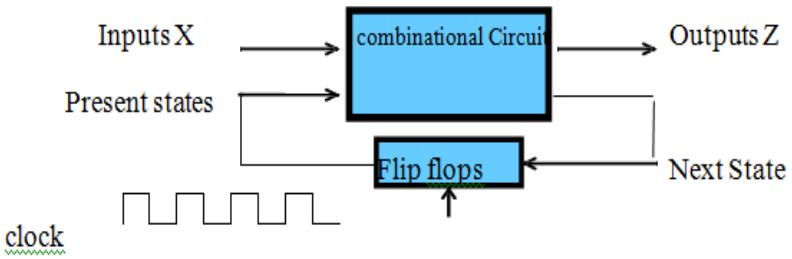
T	CLK	Q(t+1)	Comments
0	↑	Q(t)	No change
1	↑	Q(t)'	Toggle

Q	T	Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

$$Q(t+1) = T.Q' + T'.Q$$

SEQUENTIAL CIRCUITS

- Two types of sequential circuits:
  - **Synchronous**: The behavior of the circuit depends on the input signal at discrete instances of time (also called clocked)
  - **Asynchronous**: The behavior of the circuit depends on the input signals at any instance of time and the order of the inputs change
    - A combinational circuit with feedback



The storage elements (memory) used in clocked sequential circuits are called **flip-flops**

- Each flip-flop can store one bit of information 0,1
- A circuit may use many flip-flops; together they define the circuit state
  - Flip-Flops (memory/state) update **only** with the clock

Decoder

A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines. A decoder does not contain input data. A

encoder is a digital function that produces a reverse operation from that of a decoder.

Decoder types

1 Binary n to 2^n Decoder
1. 2 to 4 Binary Decoder
2 3 to 8 Binary Decoder

A decoder is a multiple-input, multiple-output logic circuit that converts coded inputs into coded outputs, where the input and output codes are different; e.g. n-to-2n, BCD decoders. Enable inputs must be on for the decoder to function, otherwise its outputs assume a single "disabled" output code word.

Decoding is necessary in applications such as data multiplexing, 7 segment display and memory address decoding.

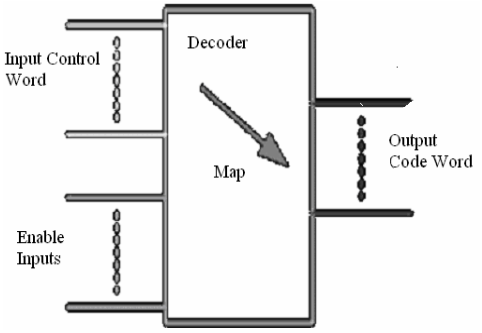


Figure 11.1 Pseudo Block of a Decoder

1. Binary n to 2^n Decoder

A binary decoder has n inputs and 2n outputs. Only one output is active at any one time, corresponding to the input value.

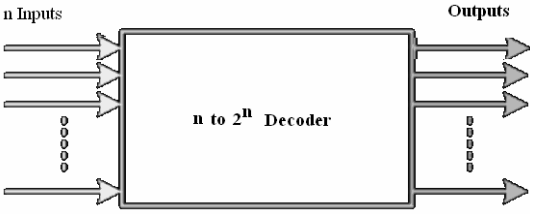


Figure 11.2 Symbol of n to 2^n Decoder

1.1 to 4 Binary Decoder

A 2 to 4 decoder consists of two inputs and four outputs, truth table and symbols of which is shown below.

X	Y	F0	F1	F2	F3
0	0	1	0	0	0
0	1	0	1	0	0

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

Table 11.1 Truth table of 2 to 4 Binary Decoder

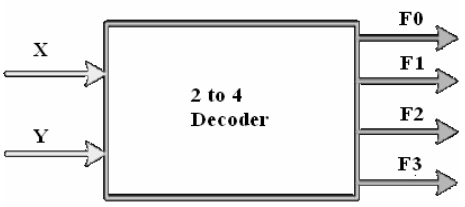


Figure 11.3 Symbol of 2 to 4 Decoder

Note: Each output is a 2-variable minterm (X'Y', X'Y, XY', XY)

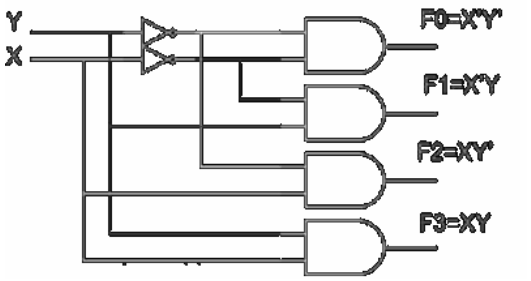


Figure 11.4 Circuit of 2 to 4 Decoder

1.2 3 to 8 Binary Decoder

A 3 to 8 decoder consists of three inputs and eight outputs, truth table and symbols of which is shown below.

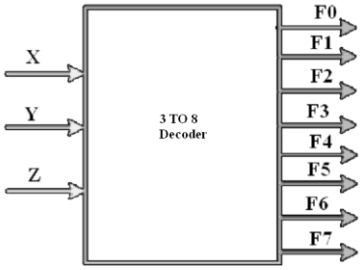


Figure 11.5 Symbol of 3 to 8 Binary Decoder

X	Y	Z	F0	F1	F2	F3	F4	F5	F6	F7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0

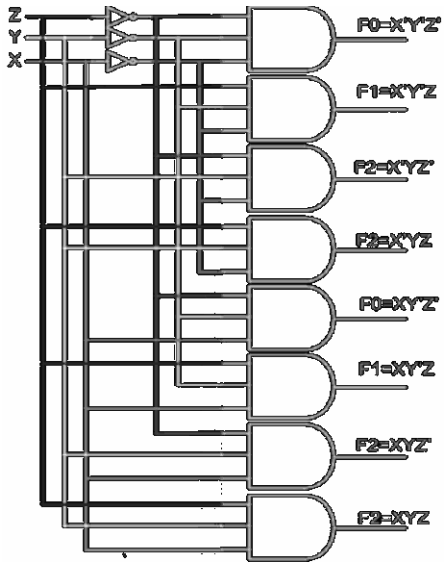


Figure 11.6 Circuit of 3 to 8 Decoder

Implementing Functions Using Decoders

Any n-variable logic function, in canonical sum-of-minterms form can be implemented using a single n-to-2n decoder to generate the minterms, and an OR gate to form the sum.
The output lines of the decoder corresponding to the minterms of the function are used as inputs to the or gate.
Any combinational circuit with n inputs and m outputs can be implemented with an n- to-2n decoder with m OR gates.
Suitable when a circuit has many outputs, and each output function is expressed with few minterms.

ENCODERS

2 Encoders

1 Binary 2^n to n Encoder
2 Octal to Binary Encoder
3 .3 Decimal to Binary Encoder

An encoder is a combinational circuit that performs the inverse operation of a decoder. If a device output code has fewer bits than the input code has, the device is usually called an encoder. e.g. 2^n-to-n, priority encoders.

11.2.2.1 Binary 2^n to n Encoder

The simplest encoder is a 2n-to-n binary encoder, where it has only one of 2n inputs = 1 and the output is the n-bit binary number corresponding to the active input.

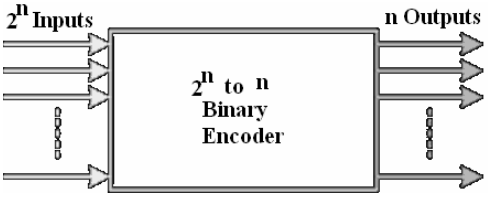


Figure 11.7 Symbol of 2^n to n Binary Encoder

11.2.2.2 Octal to Binary Encoder

Octal-to-Binary take 8 inputs and provides 3 outputs, thus doing the opposite of what the 3-to-8 decoder does. At any one time, only one input line has a value of 1.

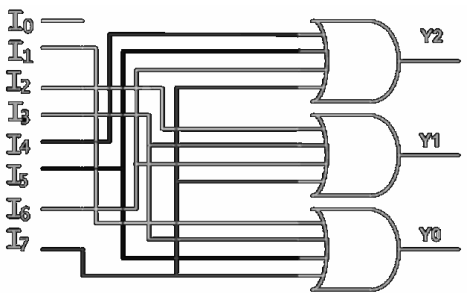
I0	I1	I2	I3	I4	I5	I6	I7	Y2	Y1	Y0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Table 11.3 Truth table of Octal-to-Binary Encoder

For an 8-to-3 binary encoder with inputs I0-I7 the logic expressions of the outputs Y0-Y2 are:

Y0 = I1 + I3 + I5 + I7
Y1= I2 + I3 + I6 + I7
Y2 = I4 + I5 + I6 +I7

Based on the above equations, we can draw the circuit as shown below



2.3 Decimal to Binary Encoder

Decimal-to-Binary take 10 inputs and provides 4 outputs, thus doing the opposite of what the 4-to-10 decoder does. At any one time, only one input line has a value of 1. The figure below shows the truth table of a Decimal-to-binary encoder

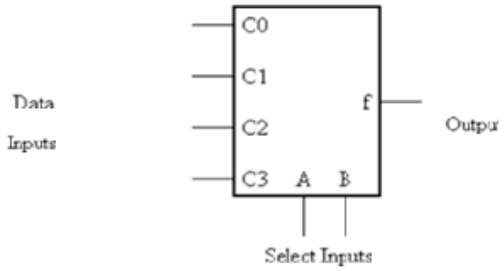
I	I	I	I	I	I	I	I	I	I	Y	Y	Y	Y
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

From the above truth table , we can derive the functions Y3, Y2, Y1 and Y0 as given below.

Y3 = I8 + I9
Y2 = I4 + I5 + I6 + I7
Y1 = I2 + I3 + I6 + I7
Y0 = I1 + I3 + I5 + I7 + I9

MULTIPLEXER

- A multiplexer is a combinatorial circuit that is given a certain number (usually a power of two) *data inputs*, let us say 2n, and n *address inputs* used as a binary number to select one of the data inputs. The multiplexer has a single output, which has the same value as the selected data input.
- Multiplexing means transmitting a large number of information units over a smaller number of channels or lines. A digital multiplexer is a combinational circuit that selects from binary information one of many input lines and directs it to a single output line.
- The selection of a particular input is controlled by a set of selection lines. These circuits are used when a complex logic circuit is shared by number of input

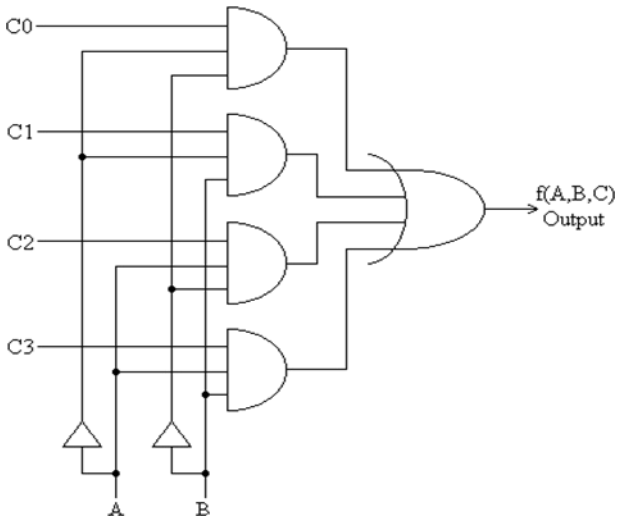


A	B	f
0	0	C_0
0	1	C_1
1	0	C_2
1	1	C_3

Table 12.1 Truth Table of 4 : 1 Multiplexer

Assume that we have four lines, ***C*0, *C*1, *C*2 and *C*3**, which are to be multiplexed on a single line, ***Output (f)***. The four input lines are also known as the ***Data Inputs***. Since there are four inputs, we will need two additional inputs to the multiplexer, known as the ***Select Inputs***, to select which of the ***C*** inputs is to appear at the output. Call these select lines ***A and B***.

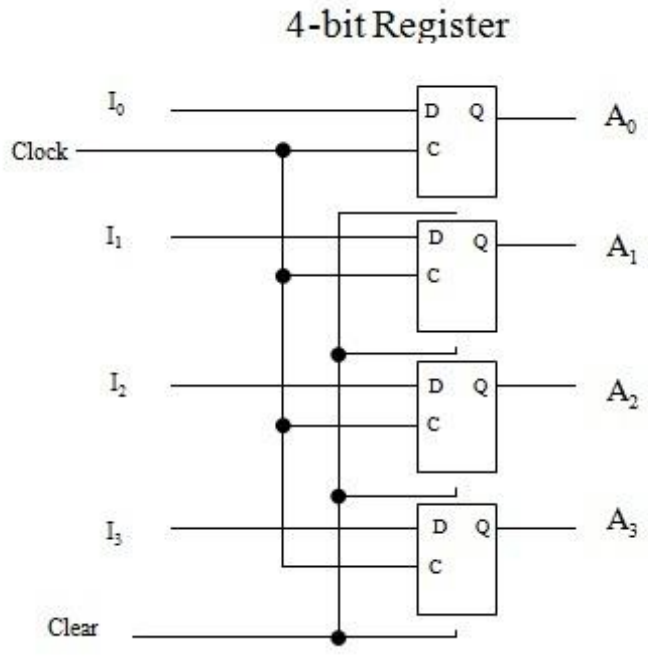
The gate implementation of a 4-line to 1-line multiplexer is shown below:



REGISTERS

- A register is a group of flip-flops.
  - Each flip-flop stores one bit of data; *n* flip-flops are required to store *n* bits of data.
  - There are several different types of registers available commercially.
  - The simplest design is a register consisting only of flip- flops, with no other gates in the circuit.
- ***Loading the register*** – transfer of new data into the register.
- The flip-flops share a common clock pulse (frequently using a buffer to reduce power requirements).
- *Output* could be sampled at any time.

- *Clearing* the flip-flop (placing zeroes in all its bit) can be done through a special terminal on the flip-flop

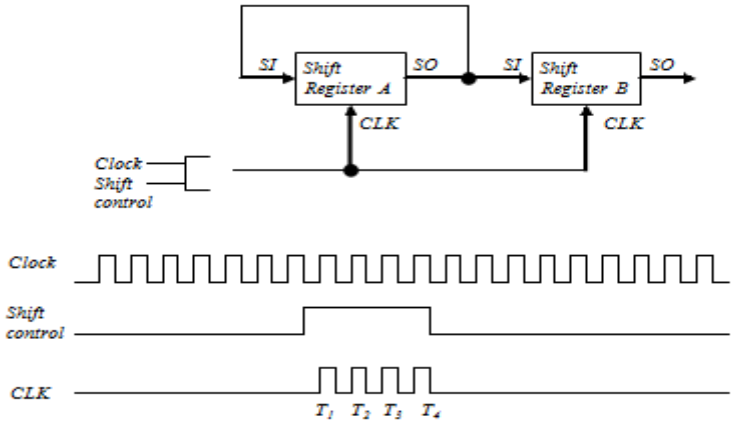


Shift Registers

- A shift register is a register which can shift its data in one or both directions.
- The simplest shift register simply connects the flip-flops to their respective neighbor with the clock controlling the operation.
- If we wish to shift on some clock pulses but not others, we can inhibit the clock pulses

on which we do not to shift.

Serial Transfer From Register A to Register B



Serial Transfer

- A digital system is operating in a serial mode when information is transferred and manipulated one bit at a time, with bits transferred out of the source register into the destination register.
- This is different from parallel transfer where all the bits of a register are transferred at once.
- Serial transfer of information from register A to register B is done with shift registers, where the serial output from register A serves as the serial input for register B.

Serial Transfer – State Table

Timing Pulse	Shift Register A	Shift Register B
Initial value	1 0 1 1	0 0 1 0
After T_1	1 1 0 1	1 0 0 1
After T_2	1 1 1 0	1 1 0 0
After T_3	0 1 1 1	0 1 1 0
After T_4	1 0 1 1	1 0 1 1

The Memory Hierarchy

Most modern programs can benefit greatly from a large amount of very fast memory. A physical reality, however, is that as a memory device gets larger, it tends to get slower. For example, cache memories are very fast but are also small and expensive. Main memory is inexpensive and large, but is slow.

At the top level of the memory hierarchy are the CPU's general purpose registers. The registers provide the fastest access to data possible on the 80x86 CPU. The register file is also the smallest memory object in the memory hierarchy (with just eight general purpose registers available

Working our way down, the Level One Cache system is the next highest performance subsystem in the memory hierarchy. On the 80x86 CPUs, the Level One Cache is provided on-chip by Intel and cannot be expanded. The size is usually quite small (typically between 4Kbytes and 32Kbytes), though much larger than the registers available on the CPU chip.

Although the Level One Cache size is fixed on the CPU and you cannot expand it, the cost per byte of cache memory is much lower than that of the registers because the cache contains far more storage than is available in all the combined registers.

The Level Two Cache is present on some CPUs, on other CPUs it is the system designer's task to incor- porate this cache (if it is present at all). For example, most Pentium II, III, and IV CPUs have a level two cache as part of the CPU package, but many of Intel's Celeron chips do not[1]. The Level Two Cache is gen- erally much larger than the level one cache (e.g., 256 or 512KBytes versus 16 Kilobytes).

On CPUs where Intel includes the Level Two Cache as part of the CPU package, the cache is not expandable. It is still lower cost than the Level One Cache because we amortize the cost of the CPU across all the bytes in the Level Two Cache.

Below the Level Two Cache system in the memory hierarchy falls the main memory subsystem. This is the general-purpose, relatively low-cost memory found in most computer systems. Typically, this is DRAM or some similar inexpensive memory technology.

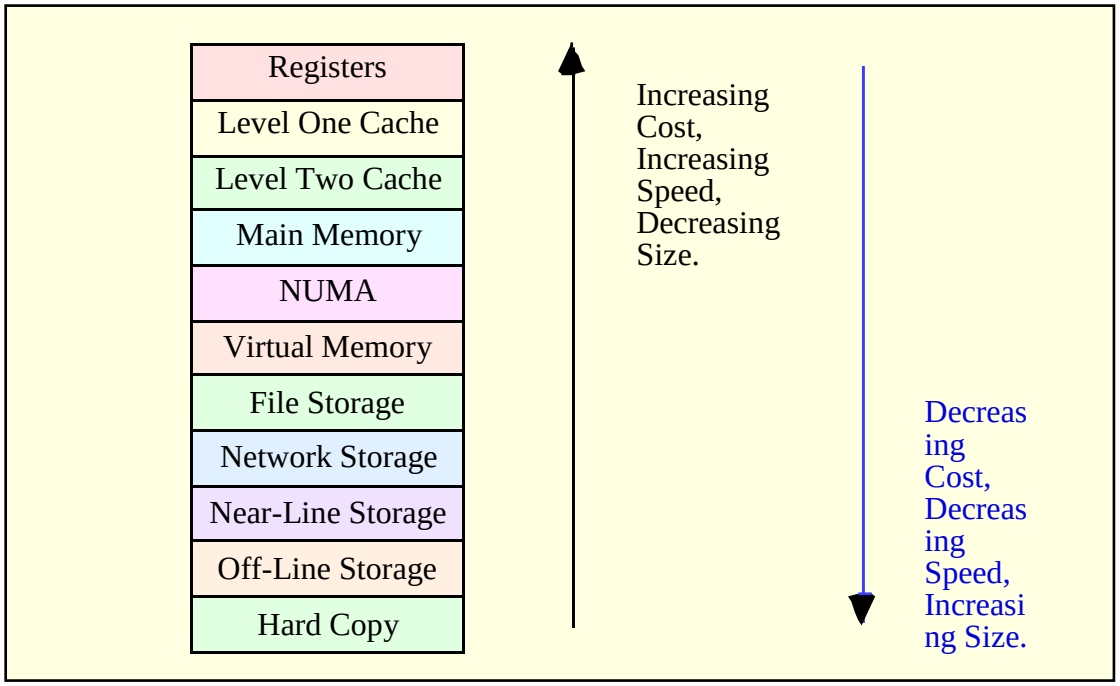


Figure 6.1 The Memory Hierarchy

Below main memory is the NUMA category. NUMA, which stands for NonUniform Memory Access is a bit of a misnomer here. NUMA means that different types of memory have different access times. There- fore, the term NUMA is fairly descriptive of the entire memory hierarchy. however, we'll use the term NUMA to describe blocks of memory that are electronically similar to main memory but for one reason or another operate significantly slower than main memory.

A good example is the memory on a video display card. Access to memory on video display cards is often much slower than access to main memory. Other peripheral devices that provide a block of shared memory between the CPU and the peripheral proba- bly have similar access times as this video card example.

Most modern computer systems implement a Virtual Memory scheme that lets them simulate main memory using storage on a disk drive. While disks are significantly slower than main memory, the cost per bit is also significantly lower. Therefore, it is far less expensive (by three orders of magnitude) to keep some data on magnetic storage rather than in main memory.

A Virtual Memory subsystem is responsible for trans- parently copying data between the disk and main memory as needed by a program.

.

Virtual Memory, File Storage, and Network Storage are examples of so-called *on-line memory sub-systems*. Memory access via these mechanism is slower than main memory access, but when a program requests data from one of these memory devices, the device is ready and able to respond to the request as quickly as is physically possible. This is not true for the remaining levels in the memory hierarchy.

The Near-Line and Off-Line Storage subsystems are not immediately ready to respond to a program's request for data. An Off-Line Storage system keeps its data in electronic form (usually magnetic or optical) but on media that is not (necessarily) connected to the computer system while the program that needs the data is running. Examples of Off-Line Storage include magnetic tapes, disk cartridges, optical disks, and floppy diskettes. When a program needs data from an off-line medium, the program must stop and wait for a someone or something to mount the appropriate media on the computer system

Hard Copy storage is simply a print-out (in one form or another) of some data. If a program requests some data and that data is present only in hard copy form, someone will have to manually enter the data into the computer. Paper (or other hard copy media) is probably the least expensive form of memory, at least for certain data types.

MEMORY UNITS

I. Introduction
- Basic units of Measurement

II. RAM,ROM,PROM,EPROM
- Storage versus Memory

III. Auxiliary Storage Devices-Magnetic Tape, Hard Disk, Floppy Disk
IV. Optical Disks: CD-R Drive, CD-RW disks, DVD, Blue ray Discs

I. Introduction

The computer system essentially comprises three important parts – input device, central processing unit (CPU) and the output device. The CPU itself is made of three components namely, the arithmetic logic unit (ALU), memory unit, and the control unit.

In addition to these, auxiliary storage/secondary storage devices are used to store data and instructions on a long-term basis.

Central processing unit

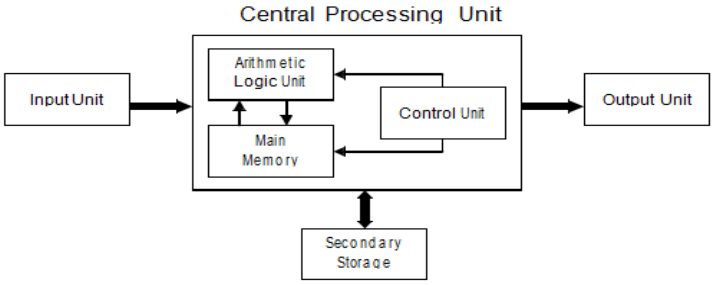


Figure 1.: Schematic Representation of a Computer

The objective of this chapter is to introduce the concept of Memory units of the computer which are shown in the above figure as main memory and secondary memory.

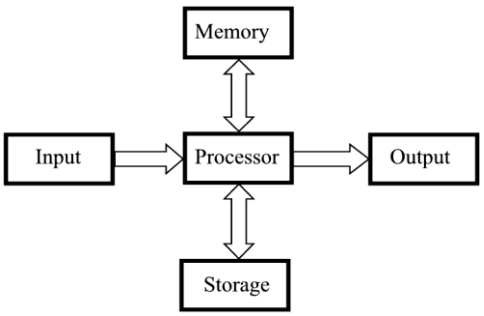


Figure 2: Linking Memory with other units

All storage devices are characterized with the following features:

- Speed
- Volatility
- Access method
- Portability
- Cost and capacity

1.1 Basic Units of Measurement

All information in the computer is handled using electrical components like the integrated circuits, semiconductors, all of which can recognize only two states – presence or absence of an electrical signal. Two symbols used to represent these two states are 0 and 1, and are known as BITS (an abbreviation for BInary DigiTS). 0 represents the absence of a signal, 1 represents the presence of a signal. A BIT is, therefore, the smallest unit of data in a computer and can either store a 0 or 1.

Since a single bit can store only one of the two values, there can possibly be only four unique combinations:

00 01 10 11

Bits are, therefore, combined together into larger units in order to hold greater range of values.

BYTES are typically a sequence of eight bits put together to create a single computer alphabetical or numerical character. More often referred to in larger multiples, bytes may appear as Kilobytes (1,024 bytes), Megabytes (1,048,576 bytes), GigaBytes

(1,073,741,824), TeraBytes (approx. 1,099,511,000,000 bytes), or PetaBytes (approx. 1,125,899,900,000,000 bytes).

Bytes are used to quantify the amount of data digitally stored (on disks, tapes) or transmitted (over the internet), and are also used to measure the memory and document size.

II. RAM,ROM,PROM,EPROM

The Term Computer Memory is defined as one or more sets of chips that store Data/program instructions, either temporarily or permanently. It is critical processing

component in any computer. The PCs use several different types. They are :

- Main Memory / Primary Memory units
  - –Two most important are
    - RAM(Random Access Memory)
    - ROM(Read-only Memory)
  - –They work in different ways and perform distinct functions
  - –CPU Registers
  - –Cache Memory
- Secondary Memory/Auxiliary Memory
  Also termed as 'auxiliary' or 'backup' storage, it is typically used as a supplement to main storage. It is much cheaper than the main storage and stores large amount of data and instructions permanently. Hardware devices like magnetic tapes and disks fall under this category.

Computer's memory can be classified into two types – RAM and ROM.

- RAM or Random Access Memory is the central storage unit in a computer system. It is the place in a computer where the operating system, application programs and the data in current use are kept temporarily so that they can be accessed by the computer's processor. The more RAM a computer has, the more data a computer can manipulate.
- Random access memory, also called the Read/Write memory, is the temporary memory of a computer. It is said to be 'volatile' since its contents are accessible only as long as the computer is on. The contents of RAM are cleared once the computer is turned off.
- ROM or Read Only Memory is a special type of memory which can only be read and contents of which are not lost even when the computer is switched off. It typically contains manufacturer's instructions. Among other things, ROM also stores an initial program called the 'bootstrap loader' whose function is to start the computer software operating, once the power is turned on.
- Read-only memories can be manufacturer-programmed or user-programmed. While manufacturer-programmed ROMs have data burnt into the circuitry, user- programmed ROMs can have the user load and then store read-only programs. PROM or Programmable ROM is the name given to such ROMs.
- Information once stored on the ROM or PROM chip cannot be altered. However, another type of memory called EPROM (Erasable PROM) allows a user to erase the information stored on the chip and reprogram it with new information. EEPROM (Electrically EPROM) and UVEPROM (Ultra Violet EPROM) are two types of EPROM's.
- Magnetic medium was found to be fairly inexpensive and long lasting medium and, therefore, became the preferred choice for auxiliary storage. Floppy disks and hard disks fall under this category. The newer forms of storage devices are optical storage

- devices like CDs, DVDs, Pen drive, Zip drive etc.

VOLATILE MEMORY

The memory is specifically meaning the RAM. This keeps the information for a shorter period of time (usually volatile), is faster and more expensive.

NON- VOLATILE MEMORY

By Storage we mean the Hard disk. Here the information is retained longer (non-volatile),It's Slower and Cheaper

III. Auxiliary Storage Devices-Magnetic Tape, Floppy Disk, Hard Disk.

The Magnetic Storage Exploits duality of magnetism and electricity. It converts electrical signals into magnetic charges, captures magnetic charge on a storage medium and then later regenerates electrical current from stored magnetic charge. Polarity of magnetic charge represents bit values zero and one.

Magnetic Disk

The Magnetic Disk is Flat, circular platter with metallic coating that is rotated beneath read/write heads. It is a Random access device; read/write head can be moved to any location on the platter.

Floppy Disk

These are small removable disks that are plastic coated with magnetic recording material. Floppy disks are typically 3.5″ in size (diameter) and can hold 1.44 MB of data. This portable storage device is a rewritable media and can be reused a number of times.

Floppy disks are commonly used to move files between different computers. The main disadvantage of floppy disks is that they can be damaged easily and, therefore, are not very reliable. The following figure shows an example of the floppy disk. Figure 3 shows a picture of the floppy di .

Figure 3: Floppy Disk

HARD DISK

Another form of auxiliary storage is a hard disk. A hard disk consists of one or more rigid metal plates coated with a metal oxide material that allows data to be magnetically recorded on the surface of the platters. The hard disk platters spin at a high rate of speed, typically 5400 to 7200 revolutions per minute (RPM).Storage capacities of hard disks for personal computers range from 10 GB to 120 GB (one billion bytes are called a gigabyte).

IV.**Optical Disks: CD-R Drive, CD-RW disks, DVD, Blue ray Discs**

Optical Mass Storage Devices Store bit values as variations in light reflection. They have higher area density & longer data life than magnetic storage. They are also Standardized and relatively inexpensive. Their Uses: read-only storage with low performance requirements, applications with high capacity requirements & where portability in a standardized format is needed.

Example of the Optical Drives

- CD's (Compact Disk)

Their storage:

~ 700 MB storage

Their Types:

- CD-ROM (read only)
- CD-R: (record) to a CD
- CD-RW: can write and erase CD to reuse it (re-writable)
- DVD(Digital Video Disk)

CD:

Compact Disk (CD) is portable disk having data storage capacity between 650-700MB. It can hold large amount of information such as music, full-motion videos, and text etc. It contains digital information that can be read, but cannot be rewritten. Separate drives exist for reading and writing CDs.

Since it is a very reliable storage media, it is very often used as a medium for distributing large amount of information to large number of users. In fact today most of the software is distributed through CDs.

DVD

Digital Versatile Disk (DVD) is similar to a CD but has larger storage capacity and enormous clarity. Depending upon the disk type it can store several Gigabytes of data (as opposed to around 650MB of a CD). DVDs are primarily used to store music or movies and can be played back on your television or the computer too. They are not rewritable media. Its also termed DVD (Digital Video Disk)

DVD-ROM

- Over 4 GB storage (varies with format)
- DVD- ROM (read only)
- Many recordable formats (e.g., DVD-R, DVD-RW; ..)
- Are more highly compact than a CD.
- Special laser is needed to read them

Blu-ray Technology

The name is derived from the blue-violet laser used to read and write data. It was developed by the Blu-ray Disc Association with more than 180 members. Some companies with the technology are Dell, Sony, LG.The Data capacity is very large because Blu-ray uses a blue laser(405 nanometers) instead of a red laser(650 nanometers) this allows the data tracks on the disc to be very compact. This allows for more than twice as small pits as on a DVD. Because of the greatly compact data Blu- ray can hold almost 5 times more data than a single layer DVD. Close to 25 GB!.Just like a DVD Blu-ray can also be recorded in Dual-Layer format. This allows the disk to hold up to 50 GB!!

The Variations in the formats are as follows:

- BD-ROM (read-only) - for pre-recorded content
- BD-R (recordable) - for PC data storage
- BD-RW (rewritable) - for PC data storage
- BD-RE (rewritable) - for HDTV recording

UNIT –III

Input and Output Devices

The main function of a computer system is to process data. The data to be processed by the computer must be input to the system and the result must be output back to the external world.

1.Input Devices

An input device is used to feed data into a computer. For example, a keyboard is an input device. It is also defined as a device that provides communication between the user and the computer. Input devices are capable of converting data into a form which can be recognized by computer. A computer can have several input devices.

Keyboard

`The most common input device is the keyboard. Keyboard consists of a set of typewriter like keys that enable you to enter data into a computer. They have alphabetic keys to enter letters, numeric keys to enter numbers, punctuation keys to enter comma, period, semicolon, etc., and special keys to perform some specific functions. The keyboard detects the key pressed and generates the corresponding ASCII codes which can be recognized by the computer.

Mouse

Mouse is an input device that controls the movement of the cursor on the display screen. Mouse is a small device, you can roll along a flat surface. In a mouse , a small ball is kept inside and touches the pad through a hole at the bottom of the mouse. When the mouse is moved, the ball rolls. This movement of the ball is converted into signals and sent to the computer. You will need to click the button at the top of the mouse to select an option. Mouse pad is a pad over which you can move a mouse. Mouse is very popular in modern computers.

Scanner

Scanner is an input device that allows information such as an image or text to be input into a computer. It can read image or text printed on a paper and translate the information into a form that the computer can use. That is, it is used to convert images (photos) and text into a stream of data. They are useful for publishing and multi-media applications.

Bar Code Reader

The barcode readers are used in places like supermarket, bookshops, etc. A bar code is a pattern printed in lines of different thickness. The bar-code reader scans the information on the bar- codes and transmits to the computer for further processing. The system gives fast and error-free entry of information into the computer.

Digital Camera

The digital camera is an input device mainly used to capture images. The digital

camera takes a still photograph, stores it and sends it as digital input to the computer. It is a modern and popular input device.

Touch Sensitive Screen

Sensitive Screen is a type of display screen that has a touch-sensitive panel. It is a pointing device that enables the user to interact with the computer by touching the screen. You can use your fingers to directly touch the objects on the screen. The touch screen senses the touch on the object (area pre-defined) and communicate the object selection to the computer.

Magnetic Ink Character Recognition (MICR)

MICR is widely used by banks to process cheques. Human readable numbers are printed on documents such as cheque using a special magnetic ink. The cheque can be read using a special input unit, which can recognize magnetic ink characters. This method eliminates the manual errors. It also saves time, ensures security and accuracy of data.

Optical Character Recognition (OCR)

The OCR technique permits the direct reading of any printed character like MICR but no special ink is required. With OCR, a user can scan a page from a book. The computer will recognize the characters in the page as letters and punctuation marks, and stores. This can be edited using a word processor.

Optical Mark Reading and Recognition (OMR)

In this method special pre-printed forms are designed with boxes which can be marked with a dark pencil or ink. Such documents are read by a reader, which transcribes the marks into electrical pulses which are transmitted to the computer. They are widely used in applications like objective type answer papers evaluation in which large number of candidates appear, time sheets of factory employees etc

.

Light Pen

A light pen is a pointing device shaped like a pen and is connected to a monitor. The tip of the light pen contains a light- sensitive element which, when placed against the screen, detects the light from the screen enabling the computer to identify the location of the pen on the screen. Light pens have the advantage of 'drawing' directly onto the screen, but this can become uncomfortable, and they are not accurate.

Magnetic Reader

- Magnetic reader is an input device which reads a magnetic strip on a card. It is handy and data can be stored and retrieved. It also provides quick identification of the card's owner.
- All the credit cards, ATM cards (banks), petro cards, etc. stores data in a magnetic strip which can be read easily by the magnetic reader.

Smart Cards

This input device stores data in a microprocessor embedded in the card. This allows information, which can be updated, to be stored on the card. These data can be read and given as input to the computer for further processing. Most of the identification cards use this method to store and retrieve the vital information.

Notes Taker

Notes taker is a device that captures natural handwriting on any surface onto a computer. Using an electronic pen, the **notes taker** displays the user's handwritten notes, memos or drawings on the computer, and stores the image for future use.

Microphone

Microphone serves as a voice input device. It captures the voice data and input to the computer. Using the microphone along with speech recognition software can offer a completely new approach to input information into your computer.

Speech recognition programs, although not yet completely exact, have made great strides in accuracy as well as ease of use. The voice-in or speech recognition approach can almost fully replace the keyboard and mouse. Speech recognition can now open the computer world to those who may have been restricted due to a physical handicap. It can also be a boon for those who have never learned to type

.

2. Output Devices

Output is anything that comes out of a computer. An output device is capable of presenting information from a computer. There are many output devices attached with the computers. But the monitors and printers are commonly used output devices.

Monitors

Monitor is a commonly used output device, sometimes called as display screen. It provides a visual display of data. Monitors are connected with the computer and are similar in appearance to a television set.

Initially there were only monochrome monitors. But gradually, we have monitors that display colour. Monitors display images and text. The smallest dot that can be displayed is called a pixel (picture element) The resolution of the screen improves as the number of pixels is increased. Most of the monitors have a 4 : 3 width to height ratio. This is called 'aspect ratio'.

The number of pixels that can be displayed vertically and horizontally gives the resolution of the monitor. The resolution of the monitor determines the quality of the display. Some popular resolutions are 640 x 480 pixels, 800 x 600 pixels and 1024 x 768 pixels. A resolution of 1024 x 768 pixels will produce sharper image than 640 x 480 pixels.

Printers

Printer is an output device that prints text or images on paper or other media (like transparencies). By printing you create what is known as a 'hard copy'. There are different kinds of printers, which vary in their speed and print quality. The two main types of printers are impact printers and non-impact printer

Impact printers include all printers that print by striking an ink ribbon. Impact printers use a print head containing a number of metal pins which strike an inked ribbon placed between the print head and the paper. Line printers, dotmatrix printers are some of the impact printers.

Characteristics of Impact Printers

- Ø In impact printers, there is physical contact with the paper to produce an image.
- Ø Due to being robust and low cost, they are useful for bulk printing.
- Ø Impact printers are ideal for printing multiple copies (that is, carbon copies) because they can easily print through many layers of paper.
- Ø Due to its striking activity, impact printers are very noisy.
- Ø Since they are mechanical in nature, they tend to be slow. Ø Impact printers do not support transparencies.

Non-impact printers are much quieter than impact printers as their printing heads do not strike the paper. Non-impact printers include laser printers, inkjet printers and thermal printers.

Characteristics of Non-Impact Printers

- Ø Non-impact printers are faster than impact printers because they have fewer moving parts.
- Ø They are quiet than impact printers because there is no striking mechanism involved.
- Ø They posses the ability to change typefaces automatically. Ø These printers produce high-quality graphics
- Ø These printers usually support the transparencies
- Ø These printers cannot print multipart forms because no impact is being made on the paper.

Line Printer

Line printers are high-speed printers capable of printing an entire line at a time. A line printer can print 150 lines to 3000 lines per minute. The limitations of line printer are they can print only one font, they cannot print graphics, the print quality is low and they are noisy to operate. But it can print large volume of text data very fast compared to the other printers. It is also used to print on multipart stationaries to prepare copies of a document.

Dot Matrix Printer

The most popular serial printer is the dot matrix printer. It prints one line of 8 or 14

points at a time, with print head moving across a line. They are similar to typewriters. They are normally slow. The printing speed is around 300 characters per second. It uses multipart stationaries to prepare copies of a document.

Thermal Printer

Thermal printers are printers that produce images by pushing electrically heated pins against special heat-sensitive paper. They are inexpensive and used widely in fax machines and calculators.

Thermal printer paper tends to darken over time due to exposure to sunlight and heat. So the printed matters on the paper fade after a week or two. It also produces a poor quality print.

Laser Printers

Laser printers use a laser beam and dry powdered ink to produce a fine dot matrix pattern. It can produce very good quality of graphic images. One of the chief characteristics of laser printers is their resolution – how many dots per inch (dpi) they lay down. The available resolutions range from 300 dpi at the low end to around1200 dpi at the high end.

Inkjet Printers

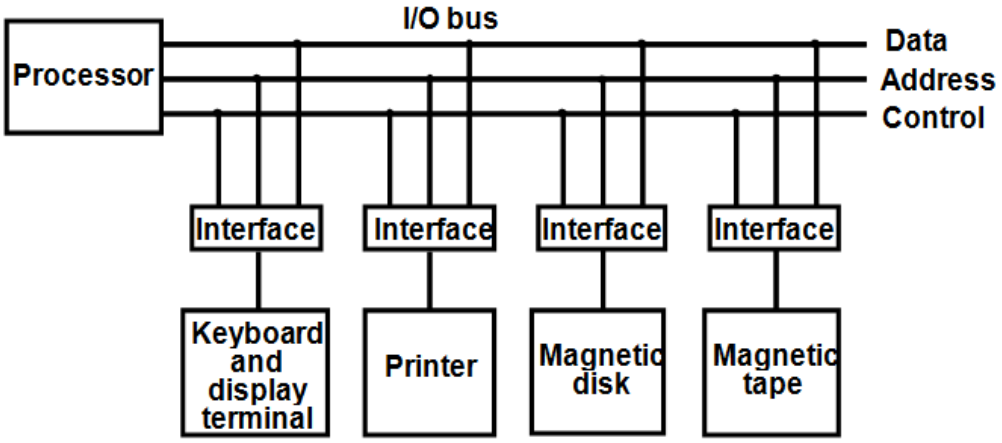
Inkjet printers use colour cartridges which combine magenta, yellow and cyan inks to create colour tones. A black cartridge is also used for crisp monochrome output. Inkjet printers work by spraying ionizing ink at a sheet of paper. Magnetized plates in the ink's path direct the ink onto the paper in the described shape.

INPUT/OUTPUT INTERFACE

Provides a method for transferring information between internal storage (such as memory and CPU registers) and external I/O devices

- Resolves the *differences* between the computer and peripheral devices
  - Peripherals – Electro-mechanical Devices
  - CPU or Memory - Electronic Device
  - Data Transfer Rate
    - » Peripherals - Usually slower
    - » CPU or Memory - Usually faster than peripherals
      - Some kinds of Synchronization mechanism may be needed
  - Unit of Information
    - » Peripherals – Byte, Block, …
    - » CPU or Memory – Word
  - Data representations may differ

I/O BUS AND INTERFACE MODULES

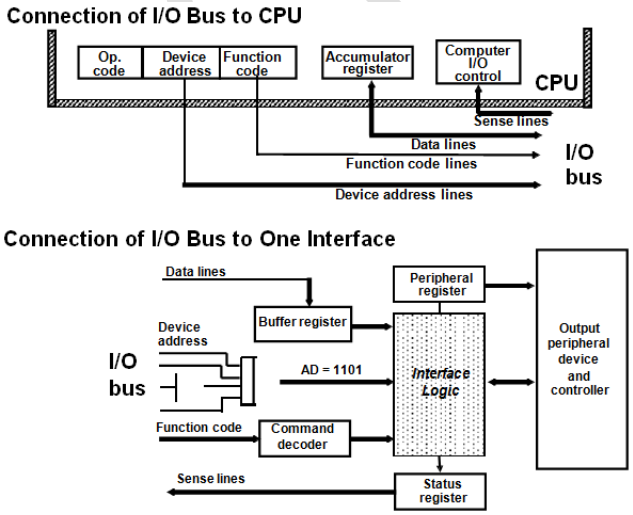


Each peripheral has an interface module associated with it

Interface

- Decodes the device address (device code)
- Decodes the commands (operation)
- Provides signals for the peripheral controller
- Synchronizes the data flow and supervises the transfer rate between peripheral and CPU or Memory

CONNECTION OF I/O BUS



I/O BUS AND MEMORY BUS

Functions of Buses

* *MEMORY BUS* is for information transfers between CPU and the MM

* *I/O BUS* is for information transfers between CPU and I/O devices through their I/O interface

Physical Organizations

* Many computers use a common single bus system

for both memory and I/O interface units

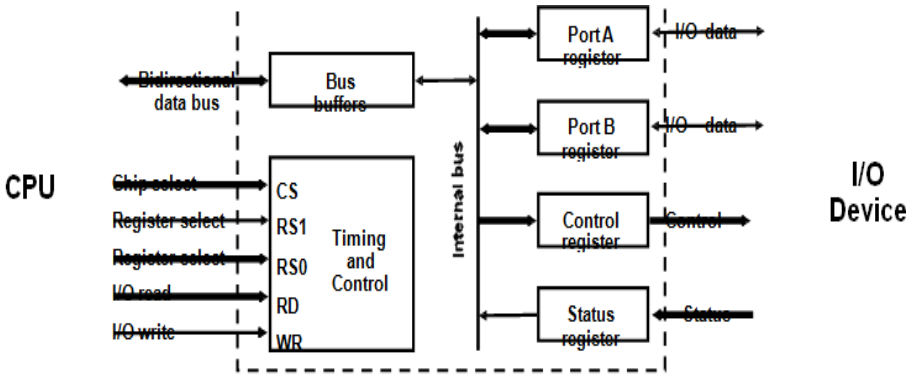
- Use one common bus but separate control lines for each function
- Use one common bus with common control lines for both functions

* Some computer systems use two separate buses,
one to communicate with memory and the other with I/O interfaces

I/O Bus

- Communication between CPU and all interface units is via a commonI/O Bus
- An interface connected to a peripheral device may have a number of *data registers* , a *control register*, and a *status register*
- A command is passed to the peripheral by sending to the appropriate interface register
- Function code and sense lines are not needed (Transfer of data, control, and status information is always via the common I/O Bus)

I/O INTERFACE



ASYNCHRONOUS DATA TRANSFER

- In a computer system, CPU and an I/O interface are designed independently of each other.
- When internal timing in each unit is independent from the other and when registers in interface and registers of CPU uses its own private clock.
- In that case the two units are said to be asynchronous to each other. CPU and I/O device must coordinate for data transfers.

METHODS USED IN A S Y N C H R O N O U S DATA TRANSFER

- Strobe Control: This is one way of transfer i.e. by means of strobe pulse supplied by one of the units to indicate to the other unit when the transfer has to occur.

- Handshaking: This method is used to accompany each data item being transferred with a control signal that indicates the presence of data in the bus. The unit receiving the data item responds with another control signal to acknowledge receipt of the data.

HANDSHAKING

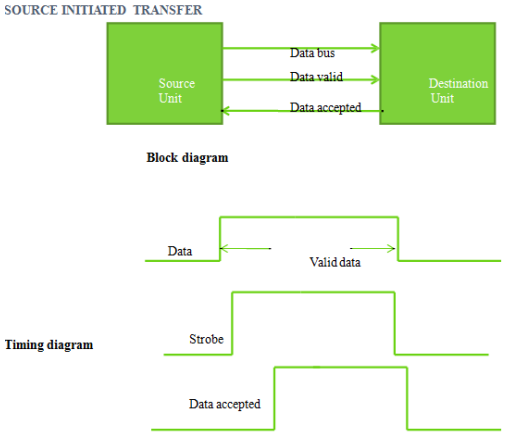
- In case of source initiated data transfer under strobe control method, the source unit has no way of knowing whether destination unit has received the data or not.
- Similarly, destination initiated transfer has no method of knowing whether the source unit has placed the data on the data bus.
- Handshaking mechanism solves this problem by introducing a second control signal that provides a reply to the unit that initiate the transfer.

There are two control lines in handshaking technique:

- Source to destination unit
- Destination to source unit

SOURCE INITIATED TRANSFER

- Handshaking signals are used to synchronize the bus activities.
- The two handshaking lines are *data valid*, which is generated by the source unit, and *data accepted*, generated by the destination unit.
- The timing diagram shows exchange of signals between two units.



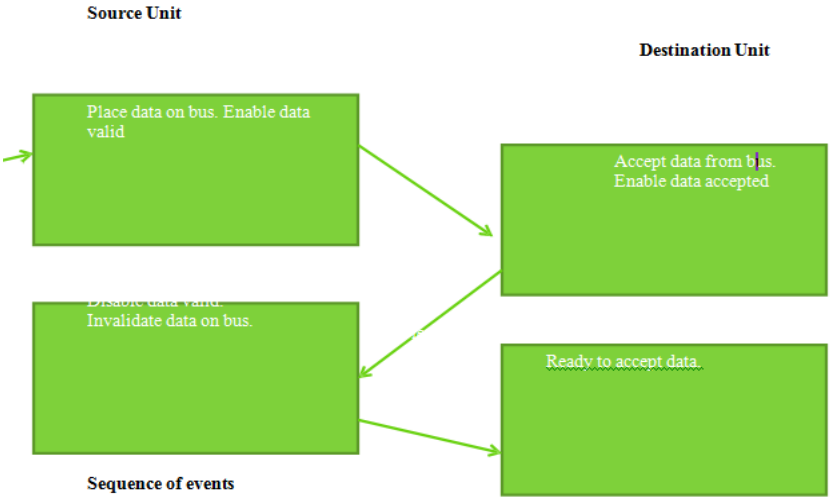
SOURCE INITIATED TRANSFER USING HANDSHAKING

The sequence of events:

- ✓ The source unit initiates the transfer by placing the data on the bus and enabling its data valid signal.
- ✓ The data accepted signals is activated by the
- ✓ destination unit after it accepts the data from the bus.
- ✓ The source unit then disables its data valid signal, which
- ✓ invalidates the data on the bus.
- ✓ The destination unit the disables its data accepted signal and the system goes into

its initial state.

SOURCE INITIATED TRANSFER USING HAND SHAKING

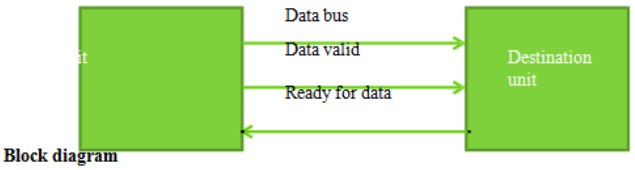


DESTINATION INITIATED TRANSFER USINGHAND SHAKING

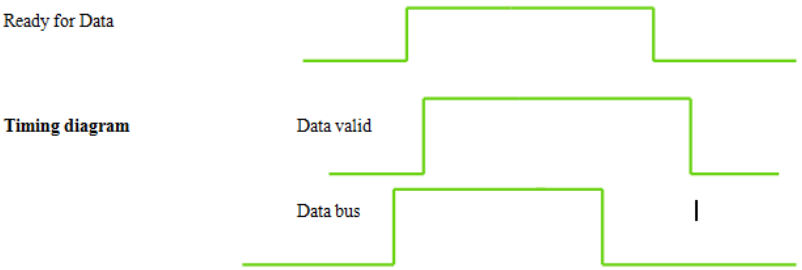
In this case the name of the signal generated by the destination unit is *ready for data*.

- The source unit does not place the data on the bus until it receives the *ready for data* signal from the destination unit.
- The handshaking procedure follows the same pattern as in source initiated case. The sequence of events in both the cases is almost same except the *ready for signal* has been converted from *data accepted* in case of source initiated.

DESTINATION INITIATED TRANSFER



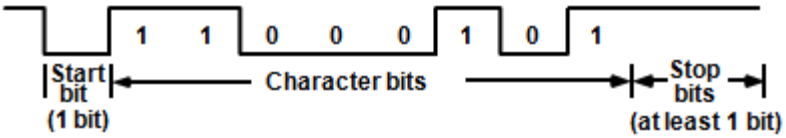
Block diagram



Timing diagram

ASYNCHRONOUS SERIAL TRANSFER

- Employs special bits which are inserted at both ends of the character code
- Each character consists of three parts; Start bit; Data bits; Stop bits.



A character can be detected by the receiver from the knowledge of 4 rules;

- When data are not being sent, the line is kept in the 1-state (idle state)
- The initiation of a character transmission is detected by a *Start Bit* , which is always a 0
- The character bits always follow the *Start Bit*
- After the last character , a *Stop Bit* is detected when the line returns to the 1-state for at least 1 bit time

The receiver knows in advance the transfer rate of the bits and the number of information bits to expect

COMMUNICATION INTERFACE

Transmitter Register

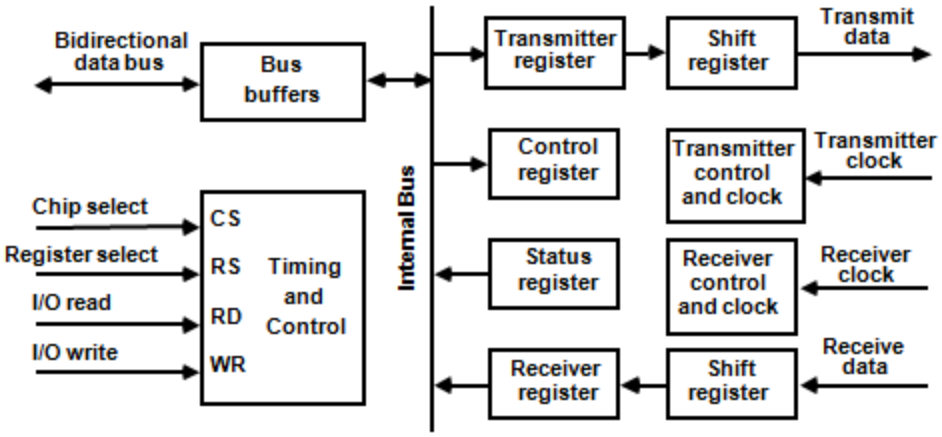
- Accepts a data byte(from CPU) through the data bus
- Transferred to a shift register for serial transmission Receiver
- Receives serial information into another shift register
- Complete data byte is sent to the receiver register

Status Register Bits

- Used for I/O flags and for recording errors Control Register Bits
- Define baud rate, no. of bits in each character, whether to generate and check parity,

and no. of stop bits

Block Diagram



Truth table

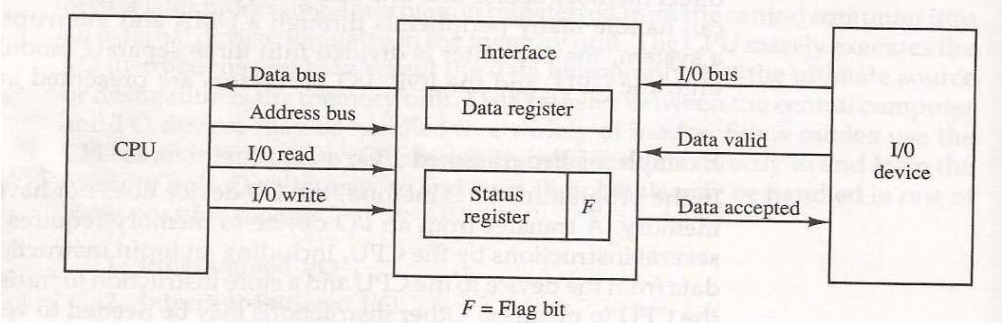
CS	RS	Oper.	Register selected
0	x	x	None
1	0	WR	Transmitter register
1	1	WR	Control register
1	0	RD	Receiver register
1	1	RD	Status register

MODES OF TRANSFER

Programmed I/O

– I/O device does not have direct access to memory

– Requires execution of several instructions by the CPU



INTERRUPT INITIATRED IN I/O

- Interrupt – refers to the transfer of control from a currently running program to another service
- program as a result of an external/internal generated request
- CPU detects interrupt from a set flag (when a interface is ready to transfer data)
- Upon detection CPU deviates its attention to another program
- Two types namely vectored interrupt and non vectored interrupt are available

PRIORITY INTRRUPT

- System that establishes a priority over the various sources to determine which condition is to be serviced first when two or more requests arrive simultaneously
- Polling – used to identify the highest priority source by software means
- Daisy chaining priority – establishing priority consists of serial connection of all devices that request an interrupt
- Devices are placed in the order of highest priority first
- VAD – Vector address in the data bus used by the CPU during the device interrupt cycle.

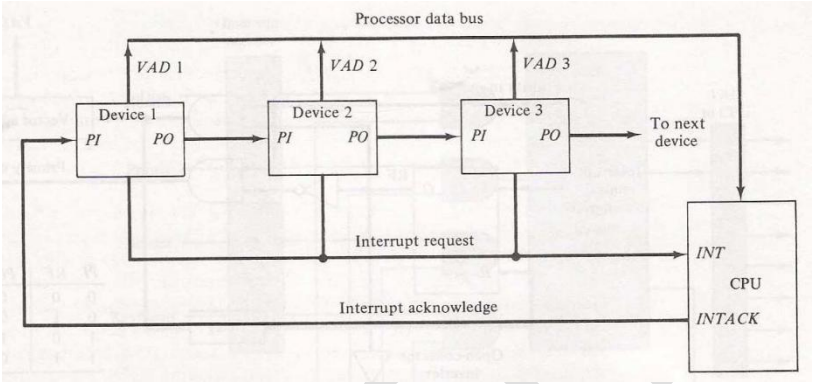


Fig: Daisy Chain Priority Interrupt

DIRECT MEMORY ACCESS (DMA)

Transfer of data between a fast storage device and memory is limited by the speed of CPU

- Remove CPU from the path of communication and the technique is DMA
- DMA controller takes over the buses to manage the transfer directly between the I/O device and memory
- Bus Request (BR) – used by the DMA controller to request the CPU to relinquish control of the buses
- CPU activates bus grant to inform the external DMA that the buses are in high impedance state
- Burst transfer – block sequence consisting of memory words is transferred in a continuous bus when DMA controller is the master
- Cycle Stealing – allows DMA controller to transfer one data word at a time after which it must return control of the buses to the CPU

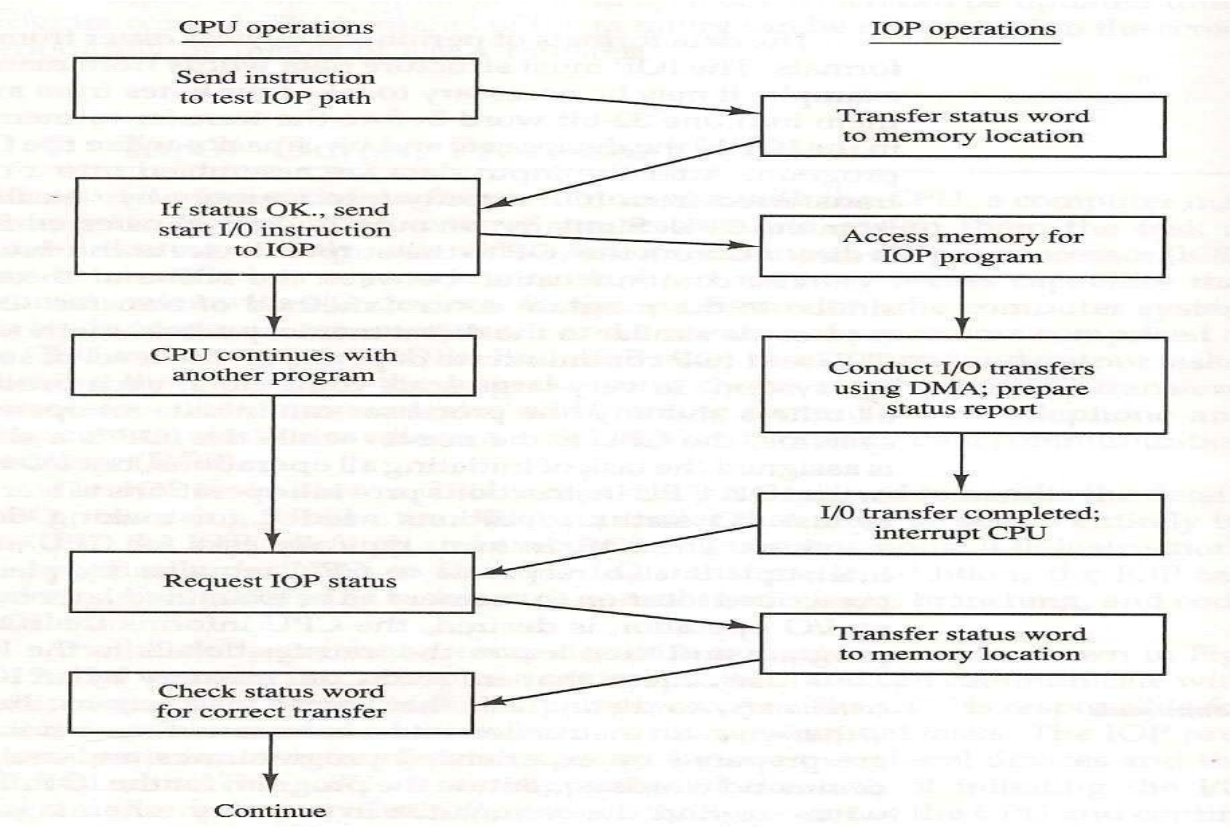
Input Output Processor

Processor with DMA capability that communicates with I/O devices

- IOP takes care of input and output tasks relieving the CPU from the housekeeping chores involved in I/O transfers

- IOP can fetch and execute its own instructions
- IOP instructions are specifically designed to facilitate I/O transfers

CPU – IO Processor Communication



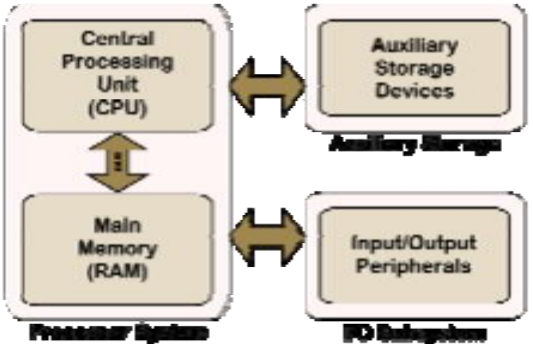
SERIAL COMMUNICATION

Data communication processor – communicates with each terminal through a single pair of wires

- Data and control information are transferred in a serial fashion
- Modems – converts digital signals into audio tones to be transmitted over telephone lines and also
- converts audio tones from the line to digital signals for machine use
- Transmission modes – Simplex, Half-duplex, Full- duplex
- Data link control protocol – set of rules that are followed by interconnecting computers and terminals

Auxiliary memory

- Main memory contains data and instructions that are in <u>active use</u>
- Auxiliary storage is for data and programs that aren't in active use
    - Usually disk drives or flash memory



Auxiliary Storage vs. Main Memory

Auxiliary Storage vs. Main Memory

Auxiliary Storage	Main Memory
Non-volatile ("permanent")	Volatile
Much greater capacity possible ("infinite" expansion)	Limited capacity and expansion possibilities
Much cheaper per MB than RAM	More expensive
Much slower than RAM	Very high speed access

- Auxiliary storage devices and media choices based on
  - Capacity
  - Speed
  - Cost

Types of Auxiliary Storage

- Auxiliary storage that is always available (like your laptop's hard disk) is called *online storage*
- Removable storage devices (like a CD-ROM or a USB jump drive) are called *offline storage*
- Three broad types of auxiliary storage
  - Sequential Access (Magnetic Tape)
  - Direct Access (Hard Drives / CDs / DVDs)
  - Random Access (Jump Drives / Memory Cards)

- Sequential Access Storage Devices (**SASD**)
  - Data items are organized in a linear sequence
  - Access time is highly variable
    - Items near the beginning of the sequence are accessed quickly, but accessing items near the end may take a long time
- Off-line storage
- Example
  - Magnetic tape

MicroComputers

Theterm*microcomputer*isgenerallysynonymouswith personalcomputer,ora computerthatdependsona microprocessor.

- Microcomputersaredesignedtobeusedbyindividuals,whetherintheformof PCs,workstationsornotebookcomputers.
- Amicrocomputercontainsa CPUona microchip(themicroprocessor),a memory system(typicallyROMandRAM),a bussystemandI/Oports,typicallyhoused ina motherboard

Theinventionofmicroprocessor(singlechipCPU)gavebirth tothemicrocomputers.Theyareseveraltimescheaperthanminicomputers.

The micro computer classified into 4 categories

1.Workstations

- Workstations are also desktop machines mainly used for intensive graphical applications. They have more processor speed thanthatofpersonalcomputers.
- Workstationsusesophisticateddisplayscreensfeaturinghigh- resolution colour graphics. Workstations are used for executing numericandgraphicintensiveapplicationssuchasComputerAided Design (CAD), simulation of complex systems and visualizing the resultsofsimulation.

2.PersonalComputers

- Todaythepersonalcomputersarethemostpopularcomputer systemssimplycalledPCs.Thesedesktopcomputersarealsoknown ashomecomputers.
- Theyareusuallyeasiertouseandmoreaffordable thanworkstations.Theyareself-containeddesktopcomputersintended foranindividualuser.Mostoftenusedforwordprocessingandsmall databaseapplications.

3.LaptopComputers

- Laptopcomputersareportablecomputersthatfitinabriefcase. **Laptop computers**,alsocalled**notebook computers**,are wonderfully portable and functional, and popular with travelers who needacomputerthatcangowiththem.

4.GettingSmallerStill

Fig. 1.23 Personal DigitalAssistants

Pen-based computersuseapenlikestylusandaccept handwritten input directly on a screen.

Pen-based computers are alsocalled**PersonalDigitalAssistants(PDA)**.Specialengineering and hardware design techniques are adopted to make the portable, smallerandlightweightcomputers.

BASICCONCEPTSOF MICROPROCESSORS

Microprocessor:Asiliconchipthatcontainsa CPU.Intheworldofpersonal computers,theterms*microprocessor*andCPUareusedinterchangeably.

- A**microprocessor**isa digitalelectroniccomponent withminiaturizedtransistorsona singlesemiconductorintegratedcircuit(IC).
- Oneor moremicroprocessorstypicallyserveasa centralprocessingunit(CPU)in a computersystemorhandhelddevice.
- Microprocessorsmadepossibletheadventofthemicrocomputer.
- Attheheartofall personalcomputersandmostworkingstationssitsa microprocessor.
- Microprocessorsalsocontrolthelogicofalmostalldigitaldevices,fromclock radiostofuel-injectionsystemsforautomobiles.
- Threebasiccharacteristicsdifferentiatemicroprocessors:
- **Instructionset**:Thesetofinstructionsthatthemicroprocessorcanexecute.
- **Bandwidth**:Thenumberofbitsprocessedina singleinstruction.
- **Clockspeed**:Giveninmegahertz(MHz),theclockspeeddetermineshowmany instructionspersecondtheprocessorcanexecute

- Inbothcases,thehigherthevalue,themorepowerfultheCPU.Forexample,a 32 bitmicroprocessorthatrunsat 50MHzismorepowerfulthana 16-bit microprocessorthatrunsat 25MHz.
- Inadditiontobandwidthandclockspeed,microprocessorsareclassifiedasbeing eitherRISC(reducedinstructionsetcomputer)orCISC(complexinstructionset computer).

➢ Differencesbetween:

–Microcomputer– acomputerwitha microprocessorasitsCPU. Includesmemory,I/O etc.

–Microprocessor– siliconchipwhichincludes ALU, registercircuits&controlcircuits

–Microcontroller– siliconchipwhichincludes microprocessor,memory&I/O inasingle package.

What is a microprocessor?

Theword comesfromthecombinationmicroand processor.

– Processormeansa devicethatprocesseswhatever.In thiscontextprocessormeansa devicethatprocesses numbers,specificallybinarynumbers,0's and 1's.

- Toprocessmeanstomanipulate.Itisa generaltermthat describesallmanipulation.Againinthiscontent,itmeansto

performcertainoperationsonthenumbersthatdependonthe microprocessor'sdesign.

Definitionofthe Microprocessor

The microprocessorisa programmabledevice that takes innumbers,performsonthem arithmeticorlogicaloperationsaccordingto the programstoredinmemoryand then producesother numbersasa result.

Assembly Language

Itisimportantto rememberthatamachine languageanditsassociatedassemblylanguageare completelymachinedependent.

– Inotherwords, theyarenottransferablefromone microprocessortoa differentone.

8085MICROPROCESSOR

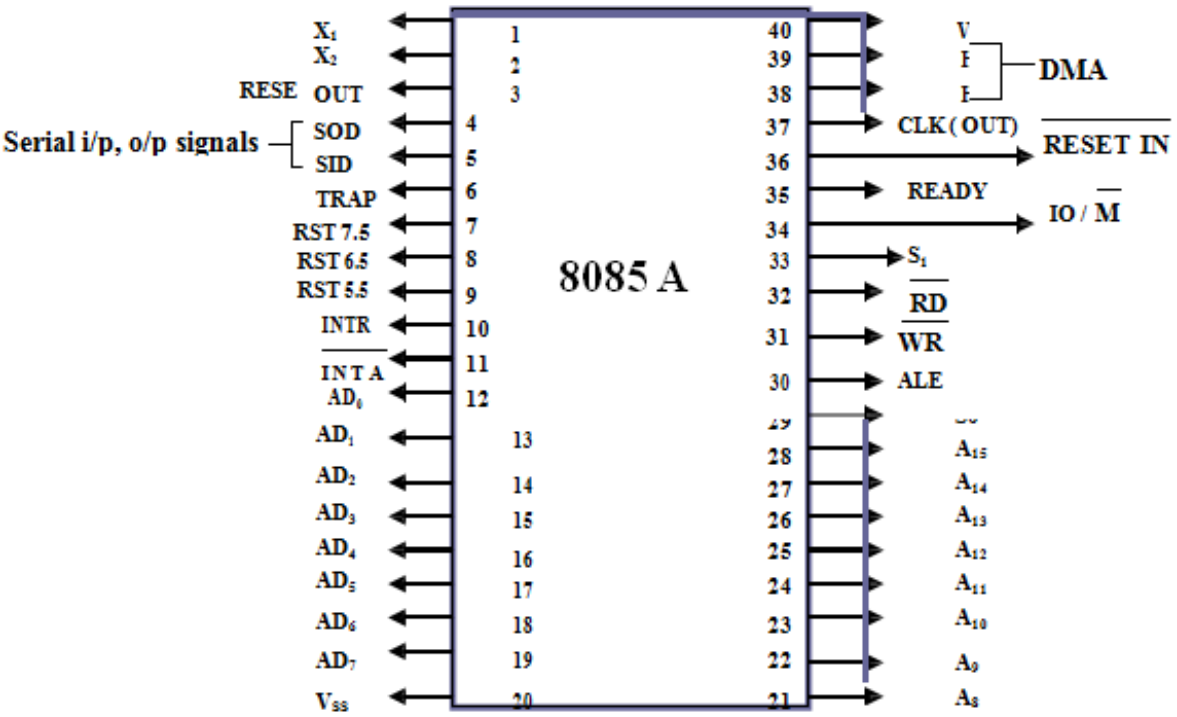
Thesalientfeaturesof8085µpare:

- Itisa 8bitmicroprocessor.
- ItismanufacturedwithN-MOStechnology.
- Ithas16-bitaddressbusandhencecanaddressupto216=65536bytes(64KB) memorylocationsthroughA_0–A_{15}.
- Thefirst8linesofaddressbusand8linesofdatabusaremultiplexedAD_0–AD_7.
- Databusisa groupof8linesD_0–D_7.
- Itsupportsexternalinterruptrequest.
- A16bitprogramcounter(PC)
- A16bitstackpointer(SP)
- Six8-bitgeneralpurposeregisterarrangedinpairs:BC,DE,HL.
- Itrequiresa signal+5Vpowersupplyandoperatesat 3.2MHZsinglephase clock.
- Itisenclosedwith40pinsDIP(Dualinlinepackage).

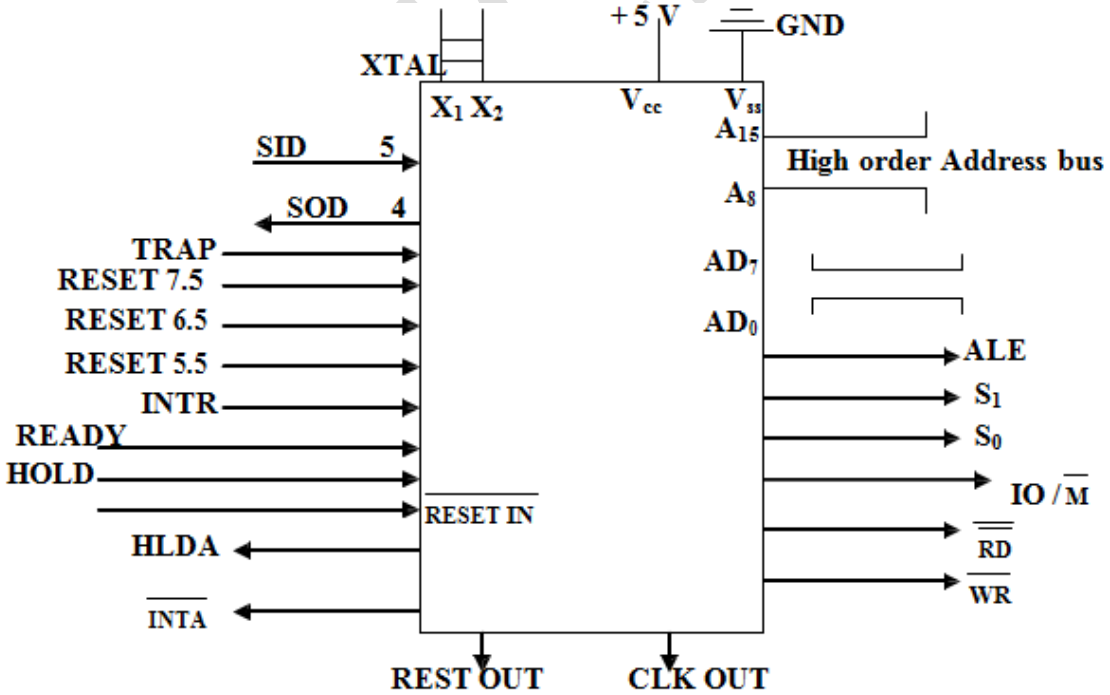
Overviewof8085microprocessor

¾ 8085Architecture

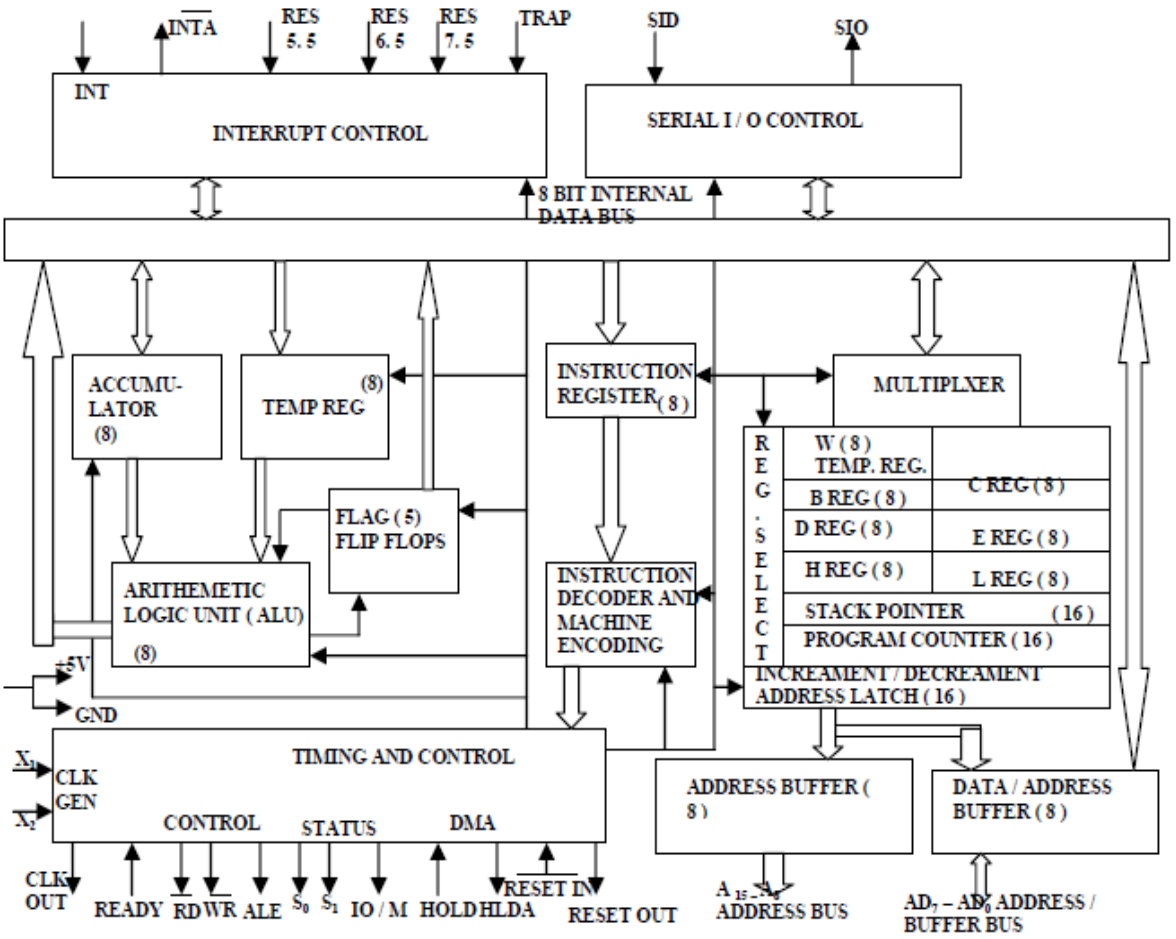
- PinDiagram
- FunctionalBlockDiagram



Pin Diagram of 8085



Signal Groups of 8085



Block Diagram

Flag Registers

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

General Purpose Registers

INDIVIDUAL	B, C, D, E, H, L
COMBININATON	B & C, D & E, H & L

Memory

- Program,dataandstackmemoriesoccupythesamememoryspace.Thetotal addressablememorysizeis64KB.
- **Programmemory**- programcanbelocatedanywhereinmemory.Jump,branch andcall instructionsuse16-bitaddresses,i.e.theycanbeusedtojump/branch anywherewithin64KB.Alljump/branchinstructionsuseabsoluteaddressing.
- **Datamemory**-theprocessoralwaysuses16-bitaddressessothatdatacanbe placedanywhere.
- **Stackmemory**islimitedonlybythesizeofmemory.Stackgrowsdownward.
- First64bytesina zeromemorypageshouldbereservedforvectorsusedbyRST instructions.

Interrupts

- Theprocessorhas5interrupts.Theyarepresentedbelowintheorderoftheir priority(fromlowesttohighest):
-
- **INTR**ismaskable8080Acompatibleinterrupt.Whentheinterruptoccursthe processorfetchesfromthebusoneinstruction,usuallyoneoftheseinstructions:
- Oneofthe8RSTinstructions(RST_0-RST_7).Theprocessorsavescurrent programcounterintostackandbranchestomemorylocationN*8(whereNisa 3-bitnumberfrom0to7 suppliedwiththeRSTinstruction).
- **CALL**instruction(3byteinstruction).Theprocessorcalls thesubroutine,address ofwhichisspecifiedinthesecondandthirdbytesoftheinstruction.
- **RST5.5**isa maskableinterrupt.Whenthisinterruptisreceivedtheprocessor savesthecontentsofthePCregisterintostackandbranchesto2CH (hexadecimal)address.
- **RST6.5**isa maskableinterrupt.Whenthisinterruptisreceivedtheprocessor savesthecontentsofthePCregisterintostackandbranchesto34H (hexadecimal)address.
- **RST7.5**isa maskableinterrupt.Whenthisinterruptisreceivedtheprocessor savesthecontentsofthePCregisterintostackandbranchesto3CH (hexadecimal)address.
- **TRAP**isa non-maskableinterrupt.Whenthisinterruptisreceivedtheprocessor savesthecontentsofthePCregisterintostackandbranchesto24H (hexadecimal)address.
- Allmaskableinterruptscanbeenabledordisabledusing EIandDIinstructions. RST5.5,RST6.5andRST7.5interruptscanbeenabledordisabledindividually usingSIMinstruction.

ResetSignals

- **RESETIN**:Whenthissignalgoeslow,theprogramcounter(PC)issettoZero, µpisresetandresetstheinterruptenableandHLDAflip-flops.
- Thedataandaddressbusesandthecontrollinesare3-statedduringRESETand

becauseofasynchronousnatureofRESET,theprocessorinternalregistersand flagsmaybealteredbyRESETwithunpredictableresults.

- RESETINisa Schmitt-triggeredinput,allowingconnectiontoanR-Cnetwork forpower-onRESETdelay.

Uponpower-up,RESETINmustremainlowforat least 10msafterminimum Vcchasbeenreached.

- Forproperresetoperationafterthepower–upduration,RESETINshouldbe keptlowa minimumofthreeclockperiods.
- TheCPUisheldintheresetconditionaslongasRESETINisapplied.Typical Power-onRESETRCvaluesR_1=75KΩ,C_1=1µF.
- **RESETOUT**:Thissignalindicatesthatµpisbeingreset.Thissignalcanbeused toresetotherdevices.Thesignalissynchronizedtotheprocessorclockandlasts anintegralnumberofclockperiods.

SerialcommunicationSignal

- **SID-SerialInputDataLine**:Thedataonthislineisloadedintoaccumulatorbit 7whenevera RIMinstructionisexecuted.
- **SOD–SerialOutputDataLine**:TheSIMinstructionloadsthevalueofbit7of theaccumulatorintoSODlatchifbit6(SOE)oftheaccumulatoris1.

DMASignals

- **HOLD**:Indicatesthatanothermasterisrequestingtheuseoftheaddressanddata buses.TheCPU,uponreceivingtheholdrequest,willrelinquishtheuseofthe busassoonasthecompletionofthecurrentbustransfer.
- Internalprocessingcancontinue.Theprocessorcanregainthebusonlyafterthe HOLDisremoved.
- WhentheHOLDisacknowledged,theAddress,DataRD,WRandIO/Mlinesare 3-stated.
- **HLDA:HoldAcknowledge**:IndicatesthattheCPUhasreceivedtheHOLD requestandthatit willrelinquishthebusinthenextclockcycle.
- HLDAgoeslowaftertheHoldrequestisremoved.TheCPUtakesthebusone half-clockcycleafterHLDAgoeslow.
- **READY:**ThissignalSynchronizesthefastCPUandtheslowmemory, peripherals.
- IfREADYishighduringa readorwritecycle,it indicatesthatthememoryor peripheralisreadytosendorreceivedata.
- IfREADYislow,theCPUwillwaitanintegralnumberofclockcyclefor READYtogohighbeforecompletingthereadorwritecycle.
- READYmustconformtospecifiedsetupandholdtimes.

Registers

- **Accumulator**orAregisterisan8-bitregisterusedforarithmetic,logic,I/Oand load/storeoperations.
- **FlagRegister**hasfive1-bitflags.
- **Sign**-setifthemostsignificantbit oftheresultisset.
- **Zero**-setiftheresultiszero.
- **Auxiliarycarry**-setiftherewasa carryoutfrombit3tobit4oftheresult.
- **Parity**-setiftheparity(thenumberofsetbitsintheresult)iseven.
- **Carry**-setiftherewasa carryduringaddition,orborrowduring subtraction/comparison/rotation.

GeneralRegisters

- 8-bitBand8-bitCregisterscanbeusedasone16-bitBCregisterpair.When usedasa pairtheCregistercontainslow-orderbyte.Someinstructionsmayuse BCregisterasa datapointer.
- 8-bitDand8-bitEregisterscanbeusedasone16-bitDEregisterpair.When usedasa pairtheEregistercontainslow-orderbyte.Someinstructionsmayuse DEregisterasa datapointer.
- 8-bitHand8-bitLregisterscanbeusedasone16-bitHLregisterpair.When used asa pairtheLregistercontainslow-orderbyte.HLregisterusuallycontains a datapointerusedtoreferencememoryaddresses.
- **Stackpointer**isa 16bitregister.Thisregisterisalways decremented/incrementedby2duringpushandpop.
- **Programcounter**isa 16-bitregister.

InstructionSet

- 8085instructionsetconsistsofthefollowinginstructions:
- Datamovinginstructions.
- Arithmetic-add,subtract,incrementanddecrement.
- Logic-AND,OR,XORandrotate.
- Controltransfer-conditional,unconditional,call subroutine,returnfrom subroutineandrestarts.
- Input/Outputinstructions.
- Other-setting/clearingflagbits,enabling/disablinginterrupts,stackoperations, etc.

Addressingmode

- **Register**- referencesthedataina registerorinaregisterpair.

  Registerindirect-instructionspecifiesregisterpaircontainingaddress,where thedataislocated.

Direct,Immediate-8or16-bitdata.

8085 INSTRUCTIONSET

1.DATATRANSFERINSTRUCTIONS

Opcode	Operand	Description
Copy from source to destination		
MOV	Rd, Rs M, Rs Rd, M	This instructioncopies the contents of the source registerinto the destination register;the contents of thesourceregisterarenotaltered. Ifoneoftheoperandsisa memorylocation,itslocationisspecifiedbythecontentsof the HL registers. Example: MOVB, C or MOVB, M
Move immediate 8-bit		
MVI	Rd, data M, data	The 8-bit data is stored in the destination registeror memory. Iftheoperandisamemorylocation,itslocationis specifiedby the contents of the HL registers. Example: MVIB, 57H or MVIM, 57H
Load accumulator		
LDA	16-bit address	The contents of a memory location, specifiedby a 16-bitaddressintheoperand,arecopiedtotheaccumulator. The contents of the source are not altered. Example: LDA 2034H
Load accumulator indirect		
LDAX	B/D Reg. pair	Thecontentsofthedesignatedregisterpairpointtoamemory location. Thisinstructioncopiesthecontentsofthatmemory locationintotheaccumulator. Thecontentsofeitherthe registerpair or the memory location are not altered. Example: LDAXB
Load registerpair immediate		
LXI	Reg. pair, 16-bit data	The instructionloads 16-bit data in the register pair designated in the operand. Example: LXI H, 2034H or LXI H, XYZ
Load H and L registersdirect		
LHLD	16-bit address	Theinstructioncopiesthecontentsofthememorylocation pointedoutbythe16-bitaddressintoregisterLandcopies thecontentsofthenextmemorylocationintoregisterH. The contents of source memory locations are not altered. Example: LHLD 2040H
Store accumulator direct		
STA	16-bit address	Thecontentsoftheaccumulatorarecopiedintothememory locationspecifiedbytheoperand. Thisisa3-byteinstruction, thesecondbytespecifiesthelow-orderaddressandthethird byte specifiesthe high-orderaddress. Example: STA 4350H
Store accumulator indirect		
STAX	Reg. pair	Thecontentsoftheaccumulatorarecopiedintothememory

locationspecifiedbythecontentsoftheoperand(register pair). The contents of the accumulator are not altered.
Example: STAXB

Store H and L registersdirect

SHLD 16-bit address — ThecontentsofregisterLarestoredintothememorylocation specifiedbythe16-bitaddressintheoperandandthecontents ofHregisterarestoredintothenextmemorylocationby incrementingtheoperand. ThecontentsofregistersHLare notaltered. Thisisa3-byteinstruction,thesecondbyte specifiesthelow-orderaddressandthethirdbytespecifiesthe high-orderaddress.
Example: SHLD2470H

Exchange H and L with D and E

XCHG none — ThecontentsofregisterHareexchangedwiththecontentsof registerD,andthecontentsofregisterLareexchangedwith the contents of registerE.
Example: XCHG

Copy H and L registersto the stack pointer

SPHL none — TheinstructionloadsthecontentsoftheHandLregistersinto the stack pointer register, the contents of the H register providethehigh-orderaddressandthecontentsoftheL registerprovidethelow-orderaddress. ThecontentsoftheH and L registersare not altered.
Example: SPHL

Exchange H and L with top of stack

XTHL none — ThecontentsoftheLregisterareexchangedwiththestack locationpointedoutbythecontentsofthestackpointer register. ThecontentsoftheHregisterareexchangedwith thenextstacklocation(SP+1);however,thecontentsofthe stack pointer registerare not altered.
Example: XTHL

Push registerpair onto stack

PUSH Reg. pair — Thecontentsoftheregisterpairdesignatedintheoperandare copiedontothestackinthefollowingsequence. Thestack pointerregisterisdecrementedandthecontentsofthehigh-orderregister(B,D,H,A)arecopiedintothatlocation. The stackpointerregisterisdecrementedagainandthecontentsof thelow-orderregister(C,E,L,flags)arecopiedtothat location.
Example: PUSHB or PUSHA

Pop off stack to registerpair

POP Reg. pair — Thecontentsofthememorylocationpointedoutbythestack pointerregisterarecopiedtothelow-orderregister(C,E,L, statusflags)oftheoperand. Thestackpointerisincremented by1andthecontentsofthatmemorylocationarecopiedto thehigh-orderregister(B,D,H,A)oftheoperand. Thestack pointer registeris again incremented by 1.

Example: POPH or POPA

Output data from accumulator to a port with 8-bit address

OUT	8-bit port address	ThecontentsoftheaccumulatorarecopiedintotheI/Oport specifiedby the operand. Example: OUT F8H

Input data to accumulator from a port with 8-bit address

IN	8-bit port address	Thecontentsoftheinputportdesignatedintheoperandare read and loaded into the accumulator. Example: IN 8CH

2.ARITHMETICINSTRUCTIONS

Opcode	Operand	Description
Add registeror memory to accumulator		
ADD	R M	The contents of the operand (registeror memory) are addedtothecontentsoftheaccumulatorandtheresultis stored in the accumulator. If the operand is a memory location,itslocationisspecifiedbythecontentsoftheHL registers. Allflagsaremodifiedtoreflecttheresultofthe addition. Example: ADD B or ADD M
Add registerto accumulator with carry		
ADC	R M	The contents of the operand (registeror memory) and theCarryflagareaddedtothecontentsoftheaccumulator andtheresultisstoredintheaccumulator. Iftheoperandisa memorylocation,itslocationisspecifiedbythecontentsof theHLregisters. Allflagsaremodifiedtoreflecttheresultof the addition. Example: ADC B or ADC M
Add immediate to accumulator		
ADI	8-bit data	The8-bitdata(operand)isaddedtothecontentsofthe accumulatorandtheresultisstoredintheaccumulator. All flags are modified to reflectthe result of the addition. Example: ADI 45H
Add immediate to accumulator with carry		
ACI	8-bit data	The8-bitdata(operand)andtheCarryflagareaddedtothe contentsoftheaccumulatorandtheresultisstoredinthe accumulator. Allflagsaremodifiedtoreflecttheresultofthe addition. Example: ACI 45H
Add registerpair to H and L registers		
DAD	Reg. pair	The16-bitcontentsofthespecifiedregisterpairareaddedto thecontentsoftheHLregisterandthesumis storedinthe HLregister. Thecontentsofthesourceregisterpairarenot altered. Iftheresultislargerthan16bits,theCYflagisset. No other flags are affected. Example: DAD H

Subtract registeror memory from accumulator

SUB	R M	The contents of the operand (registeror memory) are subtractedfromthecontentsoftheaccumulator,andtheresult isstoredintheaccumulator. Iftheoperandisamemory location,itslocationisspecifiedbythecontentsoftheHL registers. Allflagsaremodifiedtoreflecttheresultofthe subtraction. Example: SUB B or SUB M

Subtract source and borrowfrom accumulator

SBB	R M	The contents of the operand (registeror memory) and the Borrowflag are subtracted from the contents of the accumulatorandtheresultisplacedintheaccumulator. If the operand is a memory location, its location is specified by thecontentsoftheHLregisters. Allflagsaremodifiedto reflectthe result of the subtraction. Example: SBB B or SBB M

Subtract immediate from accumulator

SUI	8-bit data	The8-bitdata(operand)issubtractedfromthecontentsofthe accumulatorandtheresultisstoredintheaccumulator. All flags are modified to reflectthe result of the subtraction. Example: SUI 45H

Subtract immediate from accumulator with borrow

SBI	8-bit data	The8-bitdata(operand)andtheBorrowflagaresubtracted fromthecontentsoftheaccumulatorandtheresultisstored intheaccumulator. Allflagsaremodifiedtoreflecttheresult of the subtracion. Example: SBI 45H

Incrementregisteror memory by 1

INR	R M	The contents of the designated registeror memory) are incrementedby1andtheresultisstoredinthesameplace. If theoperandisamemorylocation,itslocationisspecifiedby the contents of the HL registers. Example: INR B or INR M

Incrementregisterpair by 1

INX	R	Thecontentsofthedesignatedregisterpairareincremented by 1 and the result is stored in the same place. Example: INXH

Decrementregisteror memory by 1

DCR R
M

The contents of the designated registeror memory are decrementedby1andtheresultisstoredinthesameplace. If theoperandisamemorylocation,itslocationisspecifiedby the contents of the HL registers.
Example: DCR B or DCR M

Decrementregisterpair by 1

DCX R

Thecontentsofthedesignatedregisterpairaredecremented by 1 and the result is stored in the same place.
Example: DCX H

Decimal adjust accumulator

DAA none

The contents of the accumulator are changed from a binary value to two 4-bit binary coded decimal (BCD) digits. This is the only instruction that uses the auxiliary flag to perform the binary to BCD conversion, and the conversion procedure is described below. S, Z, AC, P, CY flags are altered to reflect the resultsof the operation.

If the value of the low-order 4-bits in the accumulator is greater than 9 or if AC flag is set, the instruction adds 6 to the low-orderfour bits.

If the value of the high-order 4-bits in the accumulator is greater than 9 or if the Carry flag is set, the instruction adds 6 to the high-orderfour bits.

Example: DAA

3.BRANCHINGINSTRUCTIONS

Opcode Operand Description

Jump unconditionally

JMP 16-bit address Theprogramsequenceistransferredtothememorylocation specifiedby the 16-bit addressgiven in the operand.
Example: JMP2034H or

JMPXYZ Jump conditionally

Operand: 16-bit address

The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specifiedflag of the PSW as described below.
Example: JZ 2034H or JZ

XYZ Opcode	Description	Flag Status
JC	Jump on Carry	CY = 1
JNC	Jump on no Carry	CY = 0
JP	Jump on positive	S = 0
JM	Jump on minus	S = 1
JZ	Jump on zero	Z = 1
JNZ	Jump on no zero	Z = 0
JPE	Jump on parity even	P = 1
JPO	Jump on parity odd	P = 0

Unconditional subroutine call

CALL 16-bit address Theprogramsequenceistransferredtothememorylocation specifiedbythe16-bitaddressgivenintheoperand. Before thetransfer,theaddressofthenextinstructionafterCALL (the contents of the program counter) is pushed onto the stack.
Example: CALL 2034H or CALL XYZ

Call conditionally

Operand: 16-bit address

Theprogramsequenceistransferredtothememorylocation specifiedbythe16-bitaddressgivenintheoperandbasedon thespecifiedflagofthePSWasdescribedbelow. Beforethe transfer,theaddressofthenextinstructionafterthecall(the

contents of the program counter) is pushed onto the stack. Example: CZ 2034H or CZ XYZ

Opcode	Description	Flag Status
CC	Call on Carry	CY = 1
CNC	Call on no Carry	CY = 0
CP	Call on positive	S = 0
CM	Call on minus	S = 1
CZ	Call on zero	Z = 1
CNZ	Call on no zero	Z = 0
CPE	Call on parity even	P = 1
CPO	Call on parity odd	P = 0

Return from subroutine unconditionally

RET none Theprogramsequenceistransferredfromthesubroutineto thecallingprogram. Thetwobytesfromthetopofthestack arecopiedintotheprogramcounter,andprogramexecution begins at the new address.
Example: RET

Return from subroutine conditionally

Operand: none

The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW as described below. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.
Example
: RZ

Opcode	Description	Flag Status
RC	Return on Carry	CY

= 1		
RNC	Return on no Carry	
CY = 0		
RP	Return on positive	S = 0
RM	Return on minus	S = 1
RZ	Return on zero	Z
= 1		
RNZ	Return on no zero	
Z = 0		
RPE	Return on parity even	P = 1
RPO	Return on parity odd	P = 0

Load program counter with HL contents

PCHL none

ThecontentsofregistersHandLarecopiedintotheprogram counter. The contents of H are placed as the high-order byte and the contents of L as the low-orderbyte.

Example:

PCHL

Restart

RST 0-7

TheRSTinstructionisequivalenttoa1-bytecallinstruction to one of eight memory locations depending upon the number. The instructions are generally used in conjunction with interruptsandinsertedusingexternalhardware. However these can be used as software instructions in a program to transfer program execution to one of the eight locations. The addressesare:

Instruction	Restart Address
RST	0
0000H RST	1
0008H RST	2
0010H RST	3
0018H RST	4
0020H RST	5
0028H RST	6
0030H RST	7
0038H	

The 8085 has four additional interrupts and these interrupts generate RST instructions internally and thus do not require any external hardware. These instructions and their Restart addressesare:

Interrupt	Restart

Address		
TRAP		
0024H	RST	5.5
002CH	RST	6.5
0034H	RST	7.5
003CH		

4.LOGICALINSTRUCTIONS

Opcode	Operand	Description
Compare registeror memory with accumulator		

CMP	R
	M
		The contents of the operand (registeror memory) are compared withthe contents of the accumulator. Both contentsarepreserved.Theresultofthecomparisonis shown by setting the flags of the PSW as follows:
		if (A)< (reg/mem): carry flag
		is set if (A)= (reg/mem): zero
		flag is set
		if (A)> (reg/mem): carry and zero flags are reset
		Example: CMP B or CMP M

Compare immediate with accumulator		

CPI	8-bit data	Thesecondbyte(8-bitdata)iscomparedwiththecontentsof the accumulator. The values being compared remain unchanged. The result of the comparison is shown by setting the flags of the PSW as follows:
		if (A)< data: carry flag
		is set if (A)= data: zero
		flag is set
		if (A)> data: carry and zero flags are reset
		Example:
		CPI 89H

Logical AND registeror memory with accumulator		
ANA	R M	The contents of the accumulator are logically ANDedwith thecontentsoftheoperand(registerormemory),andthe resultisplacedintheaccumulator. Iftheoperandisa memorylocation,itsaddressisspecifiedbythecontentsof HLregisters. S,Z,Paremodifiedtoreflecttheresultofthe operation. CY is reset. AC is set. Example: ANA B or ANA M
Logical AND immediate with accumulator		
ANI	8-bit data	ThecontentsoftheaccumulatorarelogicallyANDedwiththe 8-bit data (operand) and the result is placed in the accumulator. S,Z,Paremodifiedtoreflecttheresultofthe operation. CY is reset. AC is set. Example: ANI86H

XRA	R M	The contents of the accumulator are Exclusive ORed with thecontentsoftheoperand(registerormemory),andthe resultisplacedintheaccumulator. Iftheoperandisa memorylocation,itsaddressisspecifiedbythecontentsof HLregisters. S,Z,Paremodifiedtoreflecttheresultofthe operation. CY and AC are reset. Example: XRA B or XRA M

Exclusive OR registeror memory with accumulator

Exclusive OR immediate with accumulator

XRI	8-bit data	ThecontentsoftheaccumulatorareExclusiveORedwiththe 8-bit data (operand) and the result is placed in the accumulator. S,Z,Paremodifiedtoreflecttheresultofthe operation. CY and AC are reset. Example: XRI 86H

Logical OR registeror memory with accumulaotr

ORA	R M	The contents of the accumulator are logically ORed with thecontentsoftheoperand(registerormemory),andthe resultisplacedintheaccumulator. Iftheoperandisa memorylocation,itsaddressisspecifiedbythecontentsof HLregisters. S,Z,Paremodifiedtoreflecttheresultofthe operation. CY and AC are reset. Example: ORA B or ORA M

Logical OR immediate with accumulator

ORI	8-bit data	ThecontentsoftheaccumulatorarelogicallyORedwiththe 8-bit data (operand) and the result is placed in the accumulator. S,Z,Paremodifiedtoreflecttheresultofthe operation. CY and AC are reset. Example: ORI 86H

Rotate accumulator left

RLC	none	Eachbinarybitoftheaccumulatorisrotatedleftbyone position. BitD7isplacedinthepositionofD0aswellasin theCarryflag. CYismodifiedaccordingtobitD7. S,Z,P, AC are not affected.

Example: RLC

Rotate accumulator right

RRC none — Eachbinarybitoftheaccumulatorisrotatedrightbyone position. Bit D_0isplacedinthepositionofD7aswellasin theCarryflag. CYismodifiedaccordingtobitD_0. S,Z,P, AC are not affected.
Example: RRC

RAL none — Eachbinarybitoftheaccumulatorisrotatedleftbyone positionthroughtheCarryflag. Bit D7isplacedintheCarry flag,andtheCarryflagisplacedintheleastsignificant positionD_0. CYismodifiedaccordingtobitD7. S,Z,P,AC are not affected.
Example: RAL

Rotate accumulator left through carry

Rotate accumulator right through carry

RAR none — Eachbinarybitoftheaccumulatorisrotatedrightbyone positionthroughtheCarryflag. Bit D_0isplacedintheCarry flag,andtheCarryflagisplacedinthemostsignificant positionD_7. CYismodifiedaccordingtobitD_0. S,Z,P,AC are not affected.
Example: RAR

Complement accumulator

CMA none — Thecontentsoftheaccumulatorarecomplemented. Noflags are affected.
Example:
CMA

Complement carry

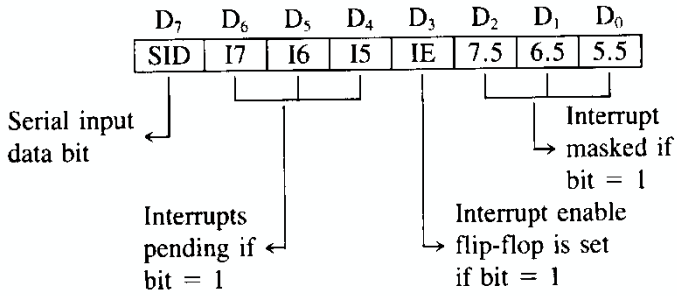
CMC none — The Carry flag is complemented. No other flags are affected.
Example:
CMC

Set Carry

STC none — The Carry flag is set to 1. No other flags are affected.
Example:
STC

CONTROLINSTRUCTIONS

Opcode	Operand	Description
No operation NOP	none	Nooperationisperformed. Theinstructionisfetchedand decoded. Howeverno operation is executed. Example: NOP
Halt and enter wait state HLT	none	TheCPUfinishesexecutingthecurrentinstructionandhalts anyfurtherexecution. Aninterruptorresetisnecessaryto exit from the halt state. Example: HLT
Disable interrupts DI	none	Theinterruptenableflip-flopisresetandalltheinterrupts except the TRAP are disabled. No flags are affected. Exampl
e: DI Enable interrupts EI	none	Theinterruptenableflip-flopissetandallinterruptsare enabled. Noflagsareaffected. Afterasystemresetorthe acknowledgementofaninterrupt,theinterruptenableflip-flopisreset,thusdisablingtheinterrupts. Thisinstructionis necessaryto reenable the interrupts(except TRAP). Example: EI
Read interruptmask RIM	none	Thisisamultipurposeinstructionusedtoreadthestatusof interrupts7.5,6.5,5.5andreadserialdatainputbit. The instruction loads eight bits in the accumulator with the following interpretations. Example: RIM



Set interruptmask

SIM	none	Thisisamultipurposeinstructionandusedtoimplementthe 8085interrupts7.5,6.5,5.5,andserialdataoutput. The instructioninterpretsthe accumulator contents as follows. Example: SIM

ASSEMBLY LANGUAGE

An assembly language is a **low-level programming language** for a computer, or other programmable device, in which there is a very strong (generally one-to-one) correspondence between the language and the architecture's machine code instructions.

WRITING ASSEMBLY LANGUAGE PROGRAMMS

1. Store the data byte 32H into memory location 4000H. MVI A, 52H : Store 32H in the accumulator

STA 4000H : Copy accumulator contents at address 4000H
HLT : Terminate program execution

Program 2:
LXI H : Load HL with 4000H
MVI M : Store 32H in memory location pointed by HL register pair
HLT : Terminate program execution

2. Exchange the contents of memory locations 2000H and 4000H.

Program 1:

LDA 2000H : Get the contents of memory location 2000H into accumulator
MOV B, A : Save the contents into B register
LDA 4000H : Get the contents of memory location 4000Hinto accumulator
STA 2000H : Store the contents of accumulator at address 2000H
MOV A, B : Get the saved contents back into A register
STA 4000H : Store the contents of accumulator at address 4000H

Program 2:

LXI H 2000H : Initialize HL register pair as a pointer to memory location 2000H.
LXI D 4000H : Initialize DE register pair as a pointer to memory location 4000H.
MOV B, M : Get the contents of memory location 2000H into B register.

LDAX D : Get the contents of memory location 4000H into A register.
MOV M, A : Store the contents of A register into memory location 2000H.
MOV A, B : Copy the contents of B register into accumulator.
STAX D : Store the contents of A register into memory location 4000H.
HLT : Terminate program execution.

3.Find the 2's complement of the number stored at memory location 4200H and store the complemented number at memory location 4300H.
Source program:

LDA 4200H : Get the number
CMA : Complement the number
ADI, 01 H : Add one in the number
STA 4300H : Store the result
HLT : Terminate program execution

4. Add the contents of memory locations 4000H and 4001H and place the result in memory location 4002H.

Sample problem
(4000H) = 14H
(4001H) = 89H
Result = 14H + 89H = 9DH

Source program

LXI H 4000H : HL points 4000H
MOV A, M : Get first operand
INX H : HL points 4001H
ADD M : Add second operand
INX H : HL points 4002H
MOV M, A : Store result at 4002H
HLT : Terminate program execution

5. Subtract the contents of memory location 4001H from the memory location 2000H and place the

result in memory location 4002H.

Sample problem:
(4000H) = 51H
(4001H) = 19H
Result = 51H – 19H = 38H

LXI H, 4000H : HL points 4000H
MOV A, M : Get first operand
INX H : HL points 4001H
SUB M : Subtract second operand
INX H : HL points 4002H
MOV M, A : Store result at 4002H.
HLT : Terminate program execution

6. Pack the two unpacked BCD numbers stored in memory locations 4200H and 4201H and store result in memory location 4300H. Assume the least significant digit is stored at 4200H.

Sample problem:
(4200H) = 04
(4201H) = 09
Result = (4300H) = 94

LDA 4201H : Get the Most significant BCD digit
RLC
RLC
RLC
RLC : Adjust the position of the second digit (09 is changed to 90)
ANI FOH : Make least significant BCD digit zero
MOV C, A : store the partial result
LDA 4200H : Get the lower BCD digit
ADD C : Add lower BCD digit
STA 4300H : Store the result

HLT : Terminate program execution

PROGRAMMING TECHNIQUES: LOOPING. COUNTING, AND INDEXJHG

The examples illustrated in the previous sections are simple and can be solved manually. However, a computer is at its best, surpassing human capability, when it has to repeat such tasks as adding a large set of numbers or copying bytes from one block of memory locations to another. It is fast and accurate.

To perform a given repetitive task, commonly used techniques are looping, counting, and indexing. To add data bytes stored in memory, for example, the following steps are necessary.

I.**Looping**

In this technique, the program is instructed to execute certain set of instructions repeatedly to execute a particular task number of times.

A loop is set up by using either a conditional Jump or an unconditional Jump as illustrated in Examples.

2. **Counting**.

This technique allows programmer to count how many times the instruction/set of instructions are executed

The counter is set by loading a count (number of times the task is to be repeated) .into a register or a register pair, and the counting is done by decrementing the count every time the loop is repeated. The counter can also be set up to count from 0 to the final count using increment instructions.

3. **Indexing**.

This technique allows programmer to point or refer the data stored in sequential memory location one by one.

The starting location of the data can be specified by loading the memory address into a register pair and using the register pair as a memory pointer or index.

Eg:

Eg:

Example 3-5

Find the sum of the values 79H, F5H, and E2H. Put the sum in registers R0 (low byte) and R5 (high byte).

Solution:

```
            MOV   A,#0        ;clear A(A=0)
            MOV   R5,A        ;clear R5
            ADD   A,#79H      ;A=0+79H=79H
            JNC   N_1         ;if no carry, add next number
            INC   R5          ;if CY=1, increment R5
N_1:        ADD   A,#0F5H     ;A=79+F5=6E and CY=1
            JNC   N_2         ;jump if CY=0
            INC   R5          ;If CY=1 then increment R5(R5=1)
N_2:        ADD   A,#0E2H     ;A=6E+E2=50 and CY=1
            JNC   OVER        ;jump if CY=0
            INC   R5          ;if CY=1, increment 5
OVER:       MOV   R0,A        ;Now R0=50H, and R5=02
```

ADDRESSING MODES

Every instruction of a program has to operate on a data. Themethod of specifying the data to be operatedby the instruction is called *Addressing.*

The 8085 has the following 5 different types of addressing.

1. Immediate Addressing
2. Direct Addressing
3. Register Addressing
4. Register Indirect Addressing
5. Implied Addressing

Immediate Addressing

In immediate addressing mode, the data is specified in the instruction itself. The data will be apartof the program instruction. All instructions that have ‘I’ in their mnemonics are of Immediateaddressing type.

*Eg.*MVI B, 3EH- Move the data 3EH given in the instruction to B register.

Direct Addressing

In direct addressing mode, the address of the data is specified in the instruction. The data will be in memory. In this addressing mode, the program instructions and data can be stored in differentmemory blocks. This type of addressing can be identified by 16bit address present in theinstruction.

*Eg.*LDA
1050H-Load the data available in memory location 1050H in accumulator.

Register Addressing

In register addressing mode, the instruction specifies the name of the register in which the data is available. This type of addressing can be identified by register names (such as 'A', 'B'....) in the instruction.

Eg. MOV A, B -Move the content of B register to A register.

Register Indirect Addressing

In register indirect addressing mode, the instruction specifies the name of the register in which the address of the data is available. Here the data will be in memory and the address will be in the register pair. This type of addressing can be identified by letter 'M' present in the instruction.

Eg. MOV A, M - The memory data addressed by HL pair is moved to A register.

Implied Addressing

In implied addressing mode, the instruction itself specifies the type of operation and location of data to be operated. This type of instruction does not have any address, register name, immediate data specified along with it.

Eg. CMA - Complement the content of accumulator

DATATRANSFERINSTRUCTIONS

Opcode	Operand	Description
Copy from source to destination		
MOV	Rd, Rs M, Rs Rd, M	This instructioncopies the contents of the source registerinto the destination register;the contents of thesourceregisterarenotaltered. Ifoneoftheoperandsisa

memorylocation,itslocationisspecifiedbythecontentsof the HL registers.
Example: MOVB, C or MOVB, M

Move immediate 8-bit

MVI	Rd, data M, data	The 8-bit data is stored in the destination registeror memory. Iftheoperandisamemorylocation,itslocationis specifiedby the contents of the HL registers. Example: MVIB, 57H or MVIM, 57H

Load accumulator

LDA	16-bit address	The contents of a memory location, specifiedby a 16-bitaddressintheoperand,arecopiedtotheaccumulator. The contents of the source are not altered. Example: LDA 2034H

Load accumulator indirect

LDAX	B/D Reg. pair	Thecontentsofthedesignatedregisterpairpointtoamemory location. Thisinstructioncopiesthecontentsofthatmemory locationintotheaccumulator. Thecontentsofeitherthe registerpair or the memory location are not altered. Example: LDAXB

Load registerpair immediate

LXI	Reg. pair, 16-bit data	The instructionloads 16-bit data in the register pair designated in the operand. Example: LXI H, 2034H or LXI H, XYZ

Load H and L registersdirect

LHLD	16-bit address	Theinstructioncopiesthecontentsofthememorylocation pointedoutbythe16-bitaddressintoregisterLandcopies thecontentsofthenextmemorylocationintoregisterH. The contents of source memory locations are not altered. Example: LHLD 2040H
STA	16-bit address	Thecontentsoftheaccumulatorarecopiedintothememory locationspecifiedbytheoperand. Thisisa3-byteinstruction, thesecondbytespecifiesthelow-orderaddressandthethird byte specifiesthe high-orderaddress. Example: STA 4350H
STAX	Reg. pair	Thecontentsoftheaccumulatorarecopiedintothememory locationspecifiedbythecontentsoftheoperand(register pair). The contents of the accumulator are not altered. Example: STAXB

Store H and L registersdirect

SHLD	16-bit address	ThecontentsofregisterLarestoredintothememorylocation specifiedbythe16-bitaddressintheoperandandthecontents ofHregisterarestoredintothenextmemorylocationby incrementingtheoperand. ThecontentsofregistersHLare notaltered. Thisisa3-byteinstruction,thesecondbyte specifiesthelow-orderaddressandthethirdbytespecifiesthe high-orderaddress. Example: SHLD2470H

Exchange H and L with D and E

XCHG	none	ThecontentsofregisterHareexchangedwiththecontentsof registerD,andthecontentsofregisterLareexchangedwith the contents of registerE. Example: XCHG

Copy H and L registersto the stack pointer

SPHL	none	TheinstructionloadsthecontentsoftheHandLregistersinto the stack pointer register, the contents of the H register providethehigh-orderaddressandthecontentsoftheL registerprovidethelow-orderaddress. ThecontentsoftheH and L registersare not altered. Example: SPHL

Exchange H and L with top of stack

XTHL	none	ThecontentsoftheLregisterareexchangedwiththestack locationpointedoutbythecontentsofthestackpointer register. ThecontentsoftheHregisterareexchangedwith thenextstacklocation(SP+1);however,thecontentsofthe stack pointer registerare not altered. Example: XTHL

Push registerpair onto stack

PUSH	Reg. pair	Thecontentsoftheregisterpairdesignatedintheoperandare copiedontothestackinthefollowingsequence. Thestack pointerregisterisdecrementedandthecontentsofthehigh-orderregister(B,D,H,A)arecopiedintothatlocation. The stackpointerregisterisdecrementedagainandthecontentsof thelow-orderregister(C,E,L,flags)arecopiedtothat location. Example: PUSHB or PUSHA

Pop off stack to registerpair

POP	Reg. pair	Thecontentsofthememorylocationpointedoutbythestack pointerregisterarecopiedtothelow-orderregister(C,E,L, statusflags)oftheoperand. Thestackpointerisincremented by1andthecontentsofthatmemorylocationarecopiedto thehigh-orderregister(B,D,H,A)oftheoperand. Thestack pointer registeris again incremented by 1. Example: POPH or POPA

Output data from accumulator to a port with 8-bit address

OUT	8-bit port address	Thecontentsoftheaccumulatorarecopiedintotheı/Oport specifiedby the operand. Example: OUT F8H

Input data to accumulator from a port with 8-bit address

IN	8-bit port address	Thecontentsoftheinputportdesignatedintheoperandare read and loaded into the accumulator. Example: IN 8CH

ARITHMETIC OPERATIONS

1. ADD
2. SUB
3. INR
4. DCR

1.ADD

• ADD - adds a *source* operand to the *destination* operand of the **same size**.

Format:

ADD *destination,source*

• Source is unchanged; *destination* stores the sum. All the status flags are affected.

• The sizes must match and only one can be a memory location.

Sample problem

(4000H) = 14H

(4001H) = 89H

Result = 14H + 89H = 9DH

Source program

LXI H 4000H	: HL points 4000H
MOV A, M	: Get first operand
INX H	: HL points 4001H
ADD M	: Add second operand
INX H	: HL points 4002H
MOV M, A	: Store result at 4002H
HLT	: Terminate program execution

SUB

SUB - subtracts as *source* operand from the *destination* operand of the **same size**.

• Format:

SUB *destination , source*

• Source is unchanged; *destination* stores the difference. All the status flags are affected.

• The sizes must match and only one can be a memory location. Subtract the contents of memory location 4001H from the memory location 2000H and place the result in memory location 4002H.

Sample problem:

(4000H) = 51H

(4001H) = 19H

Result = 51H – 19H = 38H

LXI H, 4000H	: HL points 4000H
MOV A, M	: Get first operand
INX H	: HL points 4001H
SUB M	: Subtract second operand
INX H	: HL points 4002H
MOV M, A	: Store result at 4002H.
HLT	: Terminate program execution

INC and **DEC**

INC – To increase the register address

DEC – To decrease the Register address

```
INC     Reg/Mem    ; add1todestination's
                     contents
DEC     Reg/Mem    ; subtract1to
                   ; destination'scontents
```

• The operand can be either a register or memory operand.

- A11statusflags(exceptCarry)areaffected.

INCandDEC- Examples

- Simple examples

```
INCal   ;increment8-bitregister
DECbx;decrement16-bitregister
INC eax;increment32-bitregister
INCval1;incrementmemoryoperand
```

- Anotherexample

```
.DATA
        myWordWORD 1000h
.CODE
        INCmyWord  ;1001h
        MOVbx,myWord
        DECbx           ;1000h
```

STACK

Copy H and L registers to the stack pointer

SPHL none The instruction loads the contents of the H and L registers into the stack pointer register, the contents of the H registerprovide the high-order address and the contents of the L register provide the low-order address. The contents of the H and L registers are not altered.

Example:
SPHL Exchange H and L with top of stack
XTHL none Push register pair onto stack
PUSH Reg. pair

The contents of the L register are exchanged with the stacklocation pointed out by the contents of the stack pointerregister. The contents of the H register are exchanged withthe next stack location (SP+1); however, the contents of thestack pointer register are not altered.

Example: XTHL

The contents of the register pair designated in the operand are copied onto the stack in the following sequence. The stackpointer register is decremented and the contents of the highorderregister(B,D,H,A) are copied into that location.
The stack pointer register is decremented again and the contents of the low-order register(C,E,L,flags) are copied to that location.

Example: PUSH B or PUSH A
POP Reg. pair

The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C,E, L, status flags) of the operand. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H, A) of the operand. The stack pointer register is again incremented by

1. Example: POP H or POP A

SUBROUTINE

A subroutine is a group of instructions that will be used repeatedly in different locations of the program. Rather the several times, they can be grouped into a subroutine that is called from different locations.
The 8085 has two instructions.
1.The CALL instruction is used to redirect program execution to the subroutine
2. The RET instruction is used to return the execution of the calling routine.

CONDITIONAL CALL AND RETURN INSTRUCTIONS

CALL CONDITIONALLY

Operand: 16-bit address

The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based onthe specified flag of the PSW as described below. Before thetransfer, the address of the next instruction after the call (thecontents of the program counter) is pushed onto the stack.

Example: CZ 2034 or CZ XYZ

Opcode Description
Flag Status
CC Call on Carry CY = 1
CNC Call on no Carry CY = 0
CP Call on positive S = 0
CM Call on minus S = 1
CZ Call on zero Z = 1
CNZ Call on no zero Z = 0
CPE Call on parity even P = 1

RETURN FROM SUBROUTINE UNCONDITIONALLY

RET none The program sequence is CPO Call on parity odd P = 0 transferred from the subroutine to thecalling program. The two bytes from the top of the stack are copied into the program counter, andprogram execution begins at the new address.
Example: RET
Return from subroutine conditionally

Operand: none

The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW asdescribed below. The two bytes from the top of the stack arecopied into the program counter, and program executionbegins at the new address.

Example: RZ

Opcode Description Flag Status

RC Return on Carry CY = 1

RNC Return on no Carry CY = 0

RP Return on positive S = 0

RM Return on minus S = 1

RZ Return on zero Z = 1

RNZ Return on no zero Z = 0

RPE Return on parity even P = 1

RPO Return on parity odd P = 0