DIGITAL ELECTRONICS

Understand The Basic Concepts of Analog and Digital Signals

Introduction :

The branch of electronics, which deals with digital circuits, is called digital electronics. Over the past several decades, digital electronics have been utilized in the design and manufacturing of various industrial, commercial and household electronic gadgets. Due to the proliferation of digital electronics, it is very important to inculcate the basic knowledge of digital electronics to develop conceptual knowledge and practical experience among the stakeholders.

Electronic systems can be classified into two types of systems in which the mode of electron transfer from one end to another end differs. They are,

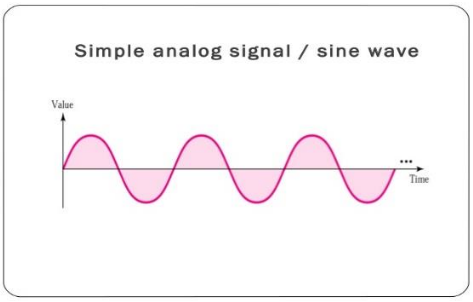
1. Analog system
2. Digital system

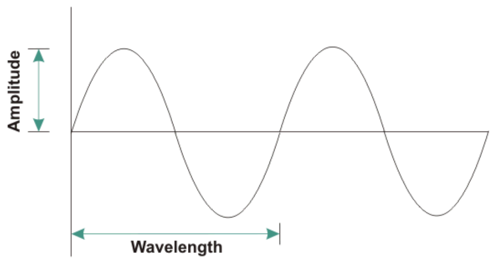
Analog and Digital Signals

i) Analog Signals :

A continuously varying signal(voltage or current) is called as an analog signal. Example: Sinusoidal waves.

A sample of analog signal that varies with time is shown in Figures.



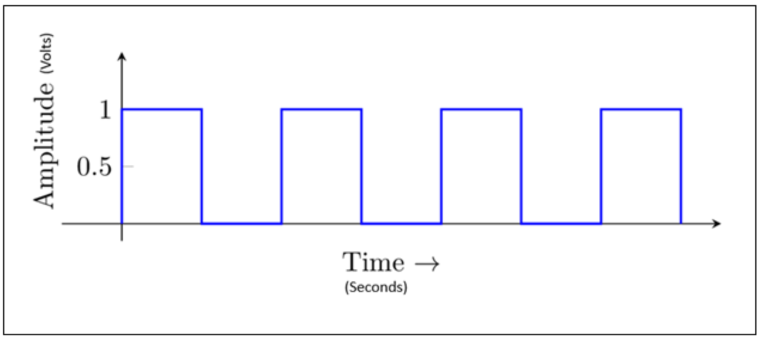


Representation of an Analog signal

ii) Digital Signal

A signal (voltage or current) that can have only two discrete values is called a digital signal.

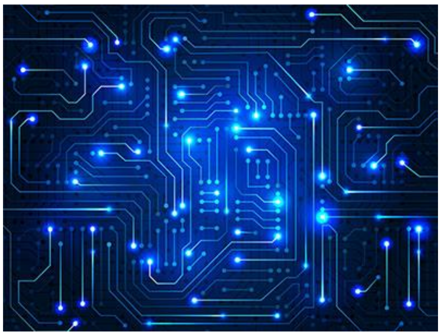
Example: Square wave. The digital waveform is shown in Figure



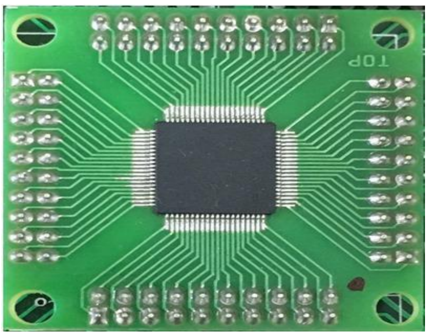
Digital operations have two states (i.e. ON or OFF) and hence it is more simple and reliable than many valued analog operations.

Digital Circuit

An electronic circuit that handles only a digital signal is called a digital circuit. Example: Digital calculator, Digital computer The digital operation is a two state operation (i.e. ON or OFF, 1 or 0) and therefore a digital circuit uses only two digits 1 and 0 in the binary number system. In order to understand the concepts in digital circuits, first we discuss about the number system in the following title.



Digital circuits



Digital Circuits Board

Number System

Introduction :

Number system is commonly used to count any activity or articles. In practical life, we are using decimal number system. In decimal number system, 10 digits(0,1,2,3,4,5,6,7,8,9) are used. But in digital electronics, we use '1' and '0'.

Computers, microprocessor and digital electronic devices do not process decimal numbers. Instead, they work with binary number, which use only the two digits'0' and'1'

People do not like working with binary numbers, owing to their very lengthy combinations of digits, while representing larger decimal values.

As a result, octal and hexadecimal numbers are widely used to compress long strings of binary numbers. Some number systems are given below.

Binary Number

Binary number contains only two numbers of '0' and '1'. It has radix or base of '2'.

Example: 10102

Almost all digital systems are based on binary number. A switch is one example of a natural binary device, because it exists only two states, namely ON or OFF, 1 or 0.

Octal Number

Octal number contains only eight numbers of 0,1,2,3,4,5,6 and 7. It has a radix or base of 8.

Example: 76128

Hexadecimal Number

Hexadecimal number contains only sixteen numbers of 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E, and F. It has a radix or base of 16.

Example: 508D16

Decimal Number System

Number	2 8 5 7 . 4 5
Weight of each digit	10^3 $10^2 10^1$ 10^0 .10^{-1} 10^{-2}

Binary Number System

Number	1 0 1 1 . 0 1
Weight of each digit	2^3 $2^2 2^1$ 2^0 .2^{-1} 2^{-2}

Octal Number System

Number	7 3 5 6. 3 2
Weight of each digit	8^3 $8^2 8^1$ 8^0 .8^{-1} 8^{-2}

Hexadecimal Number System

Number	8 A B 5 . C 9
Weight of each digit	16^3 $16^2 16^1$ 16^0 .16^{-1} 16^{-2}

CONVERSIONS

Introduction :

Conversion of binary number from one number format to another number format can be performed by adapting some rules and regulations. Some of the important conversion processes are explained below. For the conversion of integer and fractional number, separate conversion methods are used.

Decimal to Binary Conversion

In this case, the decimal number is divided by 2, and writing down the remainder after each division. The remainders are taken in reverse order to form the binary number.

Example: Conversion of 26_{10} to its equivalent binary number

2	26	
2	13	- 0
2	6	- 1
2	3	- 0
	1	- 1

Hence, $11010_2 = 26_{10}$

Binary to Decimal Conversion

To convert binary number to its equivalent decimal number, multiply each binary digit by its weight and then add the resulting products.

Example: Conversion of 1101_2 to its equivalent decimal number.

1 0 1 1

$2^3\ 2^2\ 2^1\ 2^0$

Equivalent decimal number

$= (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$

$= (1 \times 8) + (0 \times 4) + (1 \times 2) + (1 \times 1)$

$= 8 + 0 + 2 + 1 = 11$

Hence, $1011_2 = 11_{10}$

Decimal to Octal Conversion

In the case of decimal to octal conversion, the decimal number is divided by 8, and writes down the remainder after each division. The remainders are taken in reverse order to form the octal number.

Example: Conversion of the decimal number 408 to its equivalent octal number.

8 | 408
8 | 51 - 0
  6 - 3

Hence, $408_{10} = 630_8$

Octal to Decimal Conversion

To convert an octal number to its equivalent decimal number, multiply each octal digit by its weight and then add the resulting products.

Example: Conversion of an octal number 375 into its equivalent decimal number. The weight of 5 is 8^0, 7 is 8^1 and 3 is 8^2.

Hence, the equivalent decimal number is
$= (3x8^2)+(7x8^1)+(5x8^0)$
$= (3x64)+(7x8)+(5x1)$
$= 192+56+5= 253$
Hence, $375_8 = 253_{10}$

Decimal to Hexadecimal Conversion

In decimal to hexadecimal conversion, divide the decimal number by 16 and write down the remainder after each division. The remainders are taken in reverse order to form the hexadecimal number.

Example: Conversion of a decimal number 4538 to its equivalent hexadecimal number.

16 | 4538
16 | 283 - 10
16 | 17 - 11
  1 - 1

Hence, $4538_{10} = 11BA_{16}$

Hexadecimal to Decimal Conversion

To convert the hexadecimal to its equivalent decimal number, multiply each hexadecimal digit by its weight and then add the resulting products.

Example: Conversion of a hexadecimal number of B35 to its equivalent decimal number.

The weight of B is 16^2, 3 is 16^1 and 5 is 16^0

Hence its equivalent Decimal number is

$= (Bx16^2)+(3x16^1)+(5x16^0)$

$= (11x256)+(3x16)+(5x1)$

$= 2816+48+5 = 2869$

Hence, $B35_{16} = 2869_{10}$

Octal to Binary Conversion

In this, each octal digit is converted into its equivalent three digit binary form. The octal number and its equivalent three digit binary numbers are shown in the Table 1.

Table 1: Conversion of Octal into Equivalent Binary Number	
Octal number	**Equivalent Binary number**
0	**000**
1	**001**
2	**010**
3	**011**
4	**100**
5	**101**
6	**110**
7	**111**

Example: Conversion of an octal number 43 to its equivalent binary number.

4 3

100 011

$43_8 = 100011_2$

Binary to Octal Conversion

The binary numbers are grouped as 3-bit from left to right. If there is any binary digit left with one or two bits then sufficient numbers of zero are added to the left most side of the binary number. Then, grouped 3-bit number is converted into an equivalent octal number.

Example: Conversion of a binary number of 010111011 to its equivalent octal number.

010 111 011

2 7 3

Hence, $010111011_2 = 273_8$

Hexadecimal to Binary Conversion

In this, each hexadecimal digit is converted into its equivalent four digit binary form.

The hexadecimal number and its equivalent 4 digit binary numbers are shown in the Table 2.

Table 2: Conversion of Hexadecimal into Equivalent Binary Number

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

Example: Conversion of a hexadecimal number 7B3 into its equivalent binary number.

7 B 3

0111 1011 0011

Hence, $7B3_{16} = 011110110011_2$ Note: Delete the left most zeros.

Binary to Hexadecimal Conversion

In this conversion, the binary number is arranged in group of 4 bits. Suppose the binary number grouping is not completed with the 4 digits, sufficient numbers of zero are added to the left most side of the binary number.

Example: Conversion of a binary number 110110101011100 into its equivalent hexadecimal number.

0110 1101 0101 1100

6 D 5 C

Hence, $110110\ 1\ 0\ 1\ 011100_2 = 6D5C_{16}$

Decimal to BCD(Binary Coded Decimal) **Conversion**

In this method, each decimal digit is converted into its equivalent 4 digits binary form (BCD).

Example: Conversion of a decimal number 892 to its equivalent BCD number.

8 9 2

1000 1001 0010

Hence, $892_{10} = 100010010010_{BCD}$

1U BCD

BCD(Binary Coded Decimal) **to Decimal Conversion**

In this method, each BCD number grouped in the form of 4 digit binary pattern is converted into its equivalent decimal number.

Example: Convert a BCD num- ber100100111000 to its equivalent decimal number.

1001 0011 1000

9 3 8

Hence, $100100111000_{BCD} = 938_{10}$

Binary Codes

All digital circuits operate with only two states namely, High and Low or ON and OFF or 1 and 0. In binary number system, the number of bits required goes on increasing as the numbers become larger and larger. So, some special binary codes are required to represent alphabets and special characters. Based on these points, different types of binary code have been developed.

They are,

1. BCD codes
2. Gray codes
3. Excess 3 code
4. ASCII code

BCD - 8421 Code Conversion

A group of bits (usually four) which are used to represent decimal numbers 0 to 9 are called BCD(Binary Coded Decimal) codes. The most popular BCD code is 8421 code. The 8421 indicates the binary weights of the four bits (2^3, 2^2, 2^1, 2^0). Using the four bits with weights 8,4,2,1, we can easily represent the decimal numbers 0 to 9 as given in the Table 1.

Table 1: Conversion of Decimal Number into BCD Code

Decimal Numbers	BCD Code
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	0001 0000
56	0101 0110
963	1001 0110 0011

Gray Code

The gray code is not a weighted code. Therefore it is not suitable for arithmetic operations, but finds applications in input/output devices and in some types of analog to digital converters.

Table 1: Gray code conversion

Decimal numbers	Binary code	Gray code
0	0000	0000
1	0001	0001

2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

The gray code is a minimum change code in which only one bit in the code group changes when moving from one step to the next. The gray code is also called as reflected binary code, which has a special property of containing two adjacent code numbers that differ by only one bit. The gray code representation for the decimal numbers 0 to 15, together with the binary code is given in the Table 1.

Excess-3 Code

The excess-3 code is another BCD code used in earlier computers. The excess-3 code is not a weighted code. It is a self-complementing code and helps in performing subtraction operations in digital computers. The excess-3 code is also a reflection code.

An excess-3 code is obtained by adding 3 to each digit of a decimal number. For example, to encode the decimal number 6 into an excess-3 code, we must first add 3, in order to obtain 9. The 9 is then encoded into its equivalent 4 bit binary code 1001.

Conversion of the decimal number 548 to its equivalent excess-3 code.

Decimal number	5	4	8
Add 3 to each bit	+3	+ 3	+ 3
Sum =	8	7	11

Hence, the equivalent excess-3code 1000 0111 1011

The representation of Excess-3 code for the decimal numbers is given in the Table 1.

Table 1: Excess-3 Code of Decimal Number

Decimal Number	Excess-3 Code
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

Logic Gates

Logic gates are digital circuits. Digital circuits operate in binary modes, each input and output signal is either '1' or '0'. The '1' and '0' designation represents predefined voltage ranges. These electronic switching circuits are called as logic gates. Each logic gate can have one or more inputs and only one output.

All logic gates can be analysed by constructing a truth table. A truth table represents all possible input and the corresponding output combinations.

The term "logic" is usually used to refer to a decision making process. A logic gate makes logical decisions regarding the existence of output depending upon the nature of the input. Hence, such circuits are called logic circuits.

Basic Logic Gates

The three basic logic gates that makeup all digital circuits are

i) OR gate
ii) AND gate
iii) NOT gate.

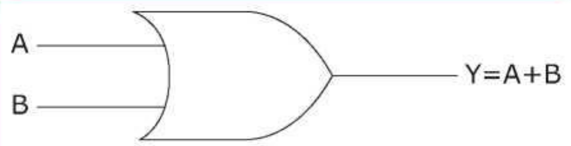
The following points may be noted about logic gates.

1. A binary '0' represents 0V and binary '1' represents +5V. It is common to refer to binary '0' as LOW input or output and binary '1' as HIGH input or output.
2. A logic gate has only one output and the output will depend upon the input signals and the type of gates.
3. The operation of a logic gate may be described either by truth table or Boolean algebra.

OR Gate

An OR gate has two or more input signals and only one output signal. An OR gate performs logical addition.

In OR gate, the inputs A,B, C, etc., produce the output as A+B+C+ etc. The symbol and the truth table of two input OR gate are shown in the Figure 1.



A	B	Y=A+B
0	0	0
0	1	1
1	0	1
1	1	1

Figure 1 Symbol and Truth Table of OR Gate

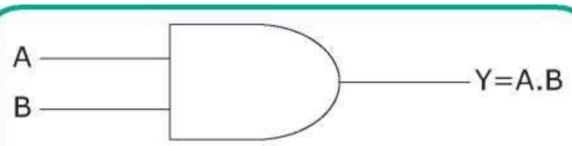
A two input OR gate contains two input signals and only one output signal. The two input signal makes $4(2^2)$ combination of outputs.

In OR gates, the output is high when any one of the input is in high level. Conversely, the output is low when all the inputs are in low level.

AND Gate

An AND gate has two or more inputs and one output. An AND gate performs logical multiplication. In an AND gates, the inputs A, B, C, etc., produce the output as A.B.C.etc. The symbol and the truth table of two input AND gate are shown in Figure 1.

It contains two input signals and only one output signal. In AND gates, the output is only high when all inputs are in high level. Conversely, the output is low only when any one of the input is in low level.



A	B	Y=A.B
0	0	0
0	1	0
1	0	0
1	1	1

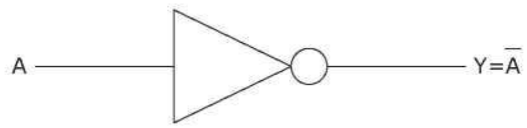
Figure 1 Symbol and Truth Table of AND Gate

NOT Gate

A NOT gate has only one input and one output. For the NOT gate, when the input is '0' (LOW), the output is '1' (HIGH)and when the input is '1' (HIGH), the output is '0' (LOW). That is, the output is complement or inverse of the input.

Figure 1 shows the symbol and truth table for the NOT gate. The input is marked as A and the output is marked as

$Y = A$. The output can be read as complement of A or inverse of A or simply A bar.



A	Y = $\overline{A}$
0	1
1	0

Figure 1 Symbol and Truth Table of NOT Gate

Digital Electronics

R.Senthamizh Selvan

Unit I

Numbers We Use in Digital Electronics

Number System in Digital Electronics
Preview

- **Counting in Decimal and Binary**
- **Place Value**
- **Binary to Decimal Conversion**
- **Decimal to Binary Conversion**
- **Electronic Translators**
- **Hexadecimal Numbers**
- **Octal Numbers**

COUNTING IN DECIMAL AND BINARY

- **Number System -**

  Code using symbols that refer to a number of items.

- **Decimal Number System -**

  Uses ten symbols (base 10 system)

- **Binary System -**

  Uses two symbols (base 2 system)

PLACE VALUE

- **Numeric value of symbols in different positions.**
- ***Example* - Place value in binary system:**

Place Value	**8s**	**4s**	**2s**	**1s**
Binary	**Yes**	**Yes**	**No**	**No**
Number	**1**	**1**	**0**	**0**

RESULT: Binary 1100 = decimal 8 + 4 + 0 + 0 = decimal 12

BINARY TO DECIMAL CONVERSION

Convert Binary Number 110011 to a Decimal Number:

Binary	**1**	**1**	**0**	**0**	**1**	**1**
	↓	↓	↓	↓	↓	↓
Decimal	**32**	+ **16**	+ **0**	+ **0**	+ **2**	+ **1** = **51**

DECIMAL TO BINARY CONVERSION

Divide by 2 Process

Decimal # 13 ÷ 2 = 6 remainder 1

6 ÷ 2 = 3 remainder 0

3 ÷ 2 = 1 remainder 1

1 ÷ 2 = 0 remainder 1

1 1 0 1

ELECTRONIC TRANSLATORS

Devices that convert from decimal to binary numbers and from binary to decimal numbers.

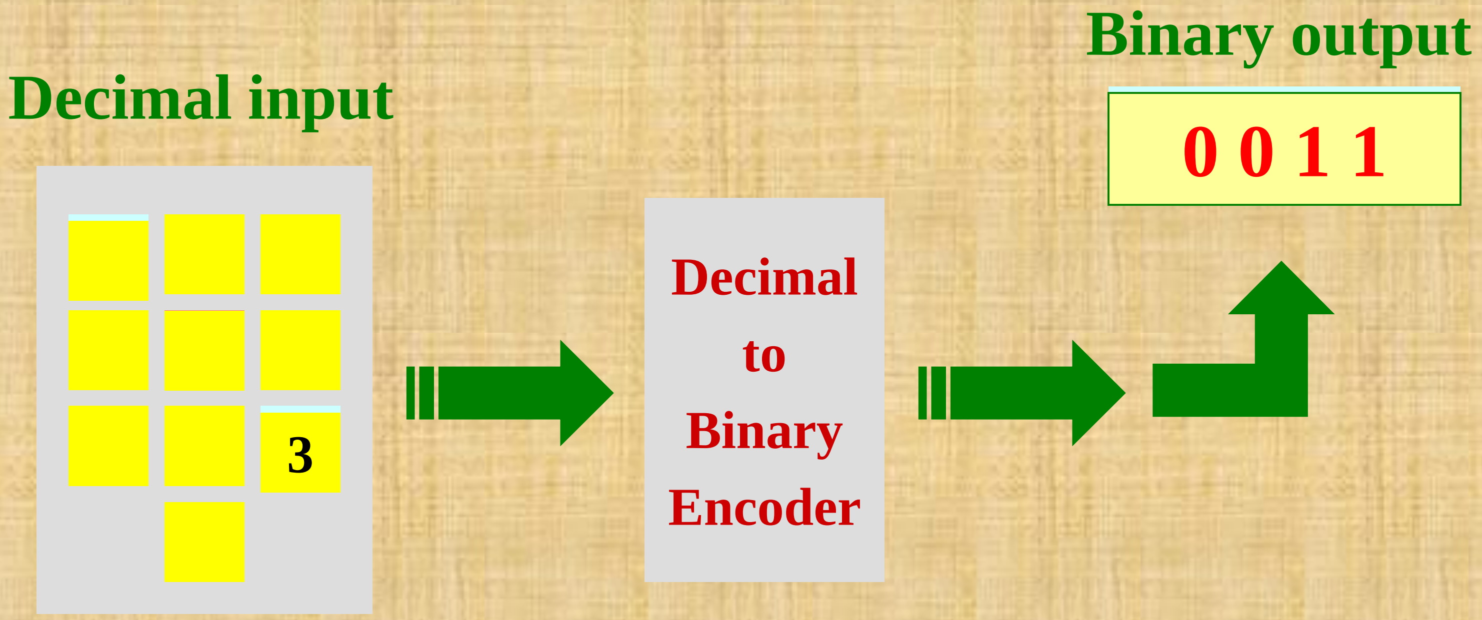
Encoders -

translates from decimal to binary

Decoders -

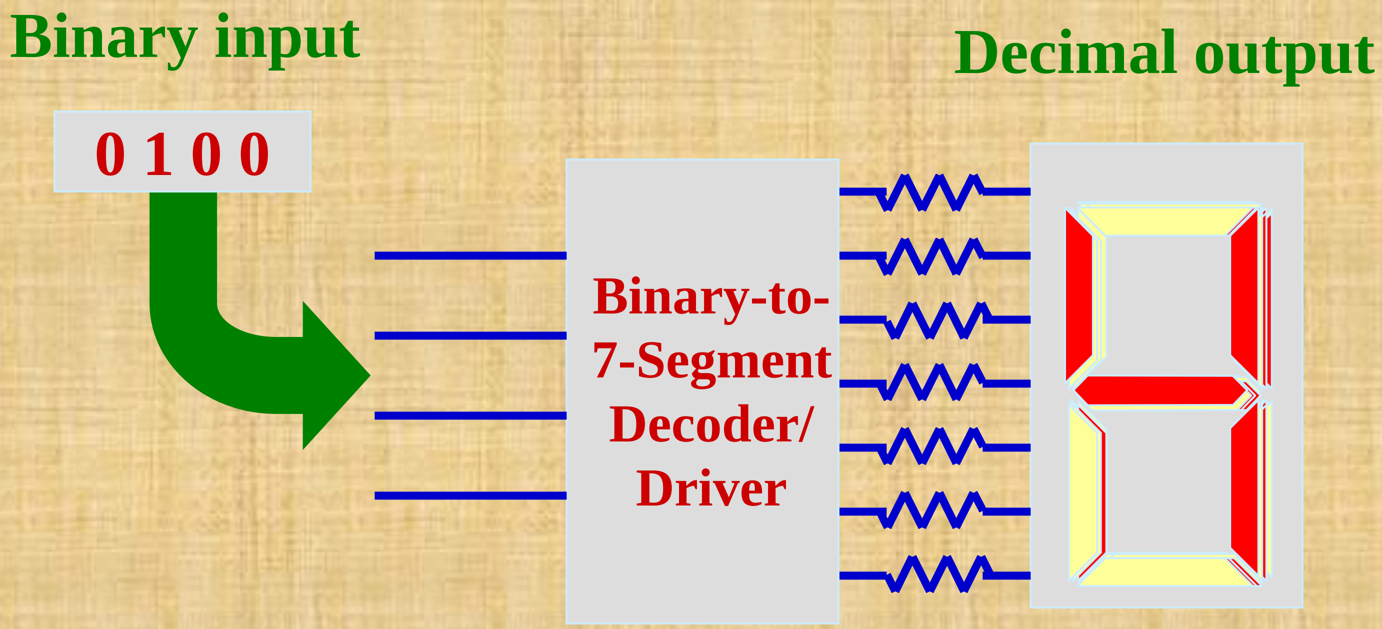
translates from binary to decimal

ELECTRONIC ENCODER - DECIMAL TO BINARY



- Encoders are available in IC form.
- This encoder translates from decimal input to binary (BCD) output.

ELECTRONIC DECODING: BINARY TO DECIMAL



- Electronic decoders are available in IC form.
- This decoder translates from binary to decimal.
- Decimals are shown on an 7-segment LED display.
- This decoder also drives the 7-segment display.

HEXADECIMAL NUMBER SYSTEM

Uses 16 symbols -Base 16 System

0-9, A, B, C, D, E, F

Decimal	Binary	Hexadecimal
1	0001	1
9	1001	9
10	1010	A
15	1111	F
16	10000	10

HEXADECIMAL AND BINARY CONVERSIONS

- **Hexadecimal to Binary Conversion**

Hexadecimal	C	3
	↓	↓
Binary	1100	0011

- **Binary to Hexadecimal Conversion**

Binary	1110	1010
	↓	↓
Hexadecimal	E	A

DECIMAL TO HEXADECIMAL CONVERSION

Divide by 16 Process

Decimal # 47 ÷ 16 = 2 remainder 15

2 ÷ 16 = 0 remainder 2

2 F

HEXADECIMAL TO DECIMAL CONVERSION

Convert hexadecimal number 2DB to a decimal number

Place Value	**256s**		**16s**		**1s**		
Hexadecimal	**2**		**D**		**B**		
	(256 x 2)		**(16 x 13)**		**(1 x 11)**		
Decimal	**512**	+	**208**	+	**11**	=	**731**

OCTAL NUMBERS

Uses 8 symbols -Base 8 System

0, 1, 2, 3, 4, 5, 6, 7

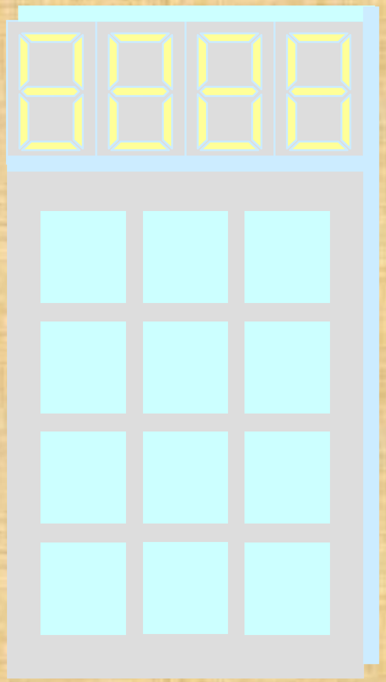
Decimal	Binary	Octal
1	001	1
6	110	6
7	111	7
8	001 000	10
9	001 001	11

PRACTICAL SUGGESTION ON NUMBER SYSTEM CONVERSIONS

- Use a scientific calculator

- Most scientific calculators have DEC, BIN, OCT, and HEX modes and can either convert between codes or perform arithmetic in different number systems.

- Most scientific calculators also have other functions that are valuable in digital electronics such as AND, OR, NOT, XOR, and XNOR logic functions.



The 8085 Microprocessor Architecture

Introduction

- ***INTRODUCTION***

  Evolution of Microprocessors

  Evolution of Digital computers

- ***Single-Chip Microcomputers***
- ***Microprocessor Applications***
- ***Programming***

Intro. Contd.

- ***Digital Computers***
- ***Memory***
- ***Buses***
- ***Memory Addressing Capacity of CPU***
- ***Processing Speed of Computer***
- ***Large and Small Computers***
- ***Batch Processing***

Intro. Contd.

- ***Multiprogramming***
- ***Multiuser System***
- ***Multitasking***
- ***Multiprocessor***
- ***Distributed Processing***
- ***Computer Network***
- ***LAN***

Intro. Contd.

- ***CAD***
- ***CAM***
- ***Computer vision***
- ***Voice Recognition and Response***
- ***Artificial Intelligence***

Intro. Contd.

- ***A CPU built into a single LSI or VLSI chip – Microprocessor***
- ***A digital computer whose CPU is a microprocessor is called a microcomputer***

Evolution of Microprocesors - Intel

4004	*1971*	*4BIT*	*16*
8008	*1972*	*8*	*18*
8080	*1973*	*8*	*40*
8085	*1976*	*8*	*40*
8086	*1978*	*16*	*40*
8088	*1980*	*8/16*	*40*
80186/88	*1982*	*8/16*	*40* *68*
80286	*1982*	*16*	*68*
80386dx	*1985*	*32*	*132*
80386sx	*1988*	*16/32*	*100*
80486	*1989*	*32*	*168*
I860	*1989*	*64*	*168*

Evolution of Digital Computers

- *First Generation* – ***Vaccum Tubes***
- *Second Generation* - ***Transistors***
- *Third Generation* - ***IC***
- *Fourth Generation* - ***Microprocessors***
- *Fifth Generation* - ***Research and development stage.***

Single chip Microcomputers

- ***Microcomputers – CPU ,RAM,ROM &I/O ports on a single chip – also known as microcontrollers.***
- ***Eg. Intel 8048,8051 series, M6801 series***
- ***Uses VLSI technology***

Microprocessor Applications

- ***Word Processing***
- ***Teletex System***
- ***Reservation for Airlines and Railways***
- ***Industrial and Commercial Application***
- ***General Application***
- ***Use of Computers for Data Analysis***
- ***Use of Computers in Graphics***
- ***Use of computers in Database Management***
- ***Use of computers in Banks***
- ***Some other Applications***

Programming

- ***Machine Level Language***
- ***Assembly Level Language***
- ***Compiler***
- ***Assembler***

Processor System Architecture

The typical processor system consists of:

- *CPU (central processing unit)*
  - *ALU (arithmetic-logic unit)*
  - *Control Logic*
  - *Registers, etc…*
- *Memory*
- *Input / Output interfaces*

Interconnections between these units:

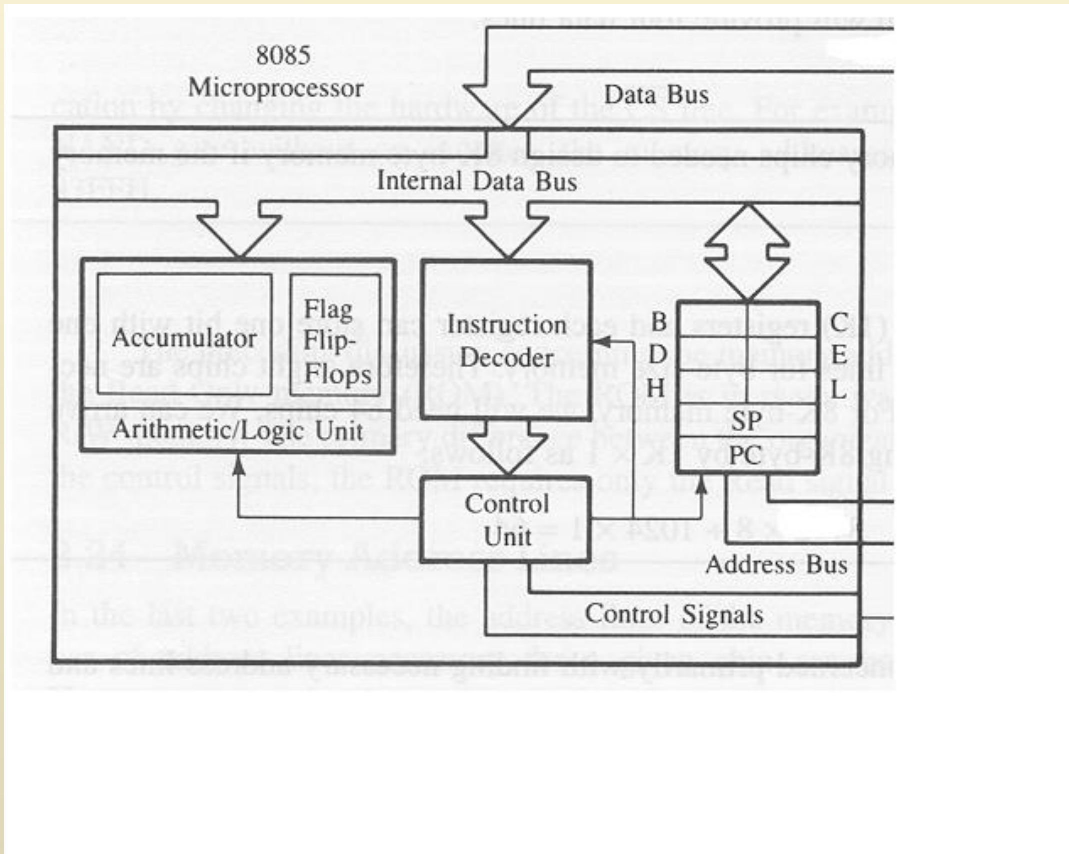
- Address Bus
- Data Bus
- Control Bus

The 8085: CPU Internal Structure

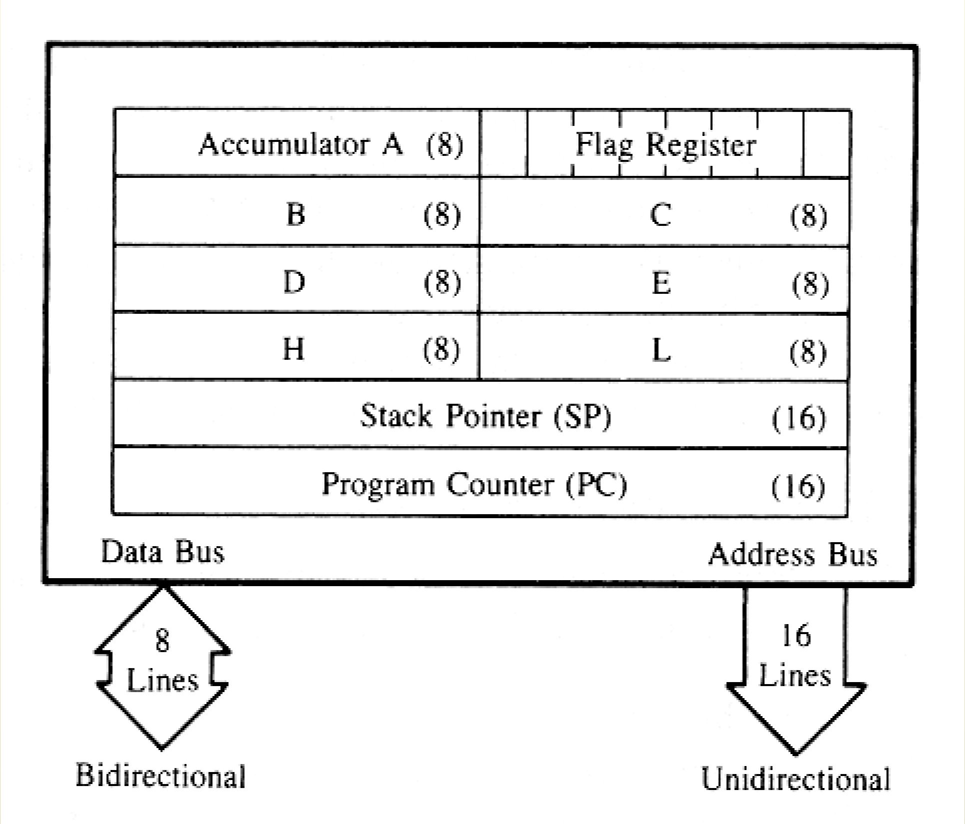
The internal architecture of the 8085 CPU is capable of performing the following operations:

- Store 8-bit data (Registers, Accumulator)
- Perform arithmetic and logic operations (ALU)
- Test for conditions (IF / THEN)
- Sequence the execution of instructions
- Store temporary data in RAM during execution

The 8085: CPU Internal Structure



The 8085: Registers



The 8085: CPU Internal Structure

Registers

- *Six general purpose 8-bit registers: B, C, D, E, H, L*
- *They can also be combined as register pairs to perform 16-bit operations: BC, DE, HL*
- *Registers are programmable (data load, move, etc.)*

Accumulator

- *Single 8-bit register that is part of the ALU !*
- *Used for arithmetic / logic operations – the result is always stored in the accumulator.*

The 8085: CPU Internal Structure

- *The Program Counter (PC)*
  - ***This is a register that is used to control the sequencing of the execution of instructions.***
  - ***This register always holds the address of the next instruction.***
  - ***Since it holds an address, it must be 16 bits wide.***
- *The Stack pointer*
  - ***The stack pointer is also a 16-bit register that is used to point into memory.***
  - ***The memory this register points to is a special area called the stack.***
  - ***The stack is an area of memory used to hold data that will be retreived soon.***
  - ***The stack is usually accessed in a Last In First Out (LIFO) fashion.***

The 8085: CPU Internal Structure

- ***IR***
  - ***This is a register that is used to holds the instruction until it is decoded..***
- ***Status Register***
  - ***There is a set of five flip-flops which act as status flags.***
  - ***Flag registers – holds 1-bit flag***

Temporary Register

PSW (Program Status Word) –

The ALU

- ***In addition to the arithmetic & logic circuits, the ALU includes the accumulator, which is part of every arithmetic & logic operation.***

- ***Also, the ALU includes a temporary register used for holding data temporarily during the execution of the operation. This temporary register is not accessible by the programmer.***

The Flags register

- ***Description***

There is also the flags register whose bits are affected by the arithmetic & logic operations.

S-sign flag

The sign flag is set if bit D7 of the accumulator is set after an arithmetic or logic operation.

Z-zero flag

Set if the result of the ALU operation is 0. Otherwise is reset. This flag is affected by operations on the accumulator as well as other registers. (DCR B).

AC-Auxiliary Carry

This flag is set when a carry is generated from bit D3 and passed to D4 . This flag is used only internally for BCD operations.

P-Parity flag

After an ALU operation if the result has an even # of 1's the p-flag is set. Otherwise it is cleared. So, the flag can be used to indicate even parity.

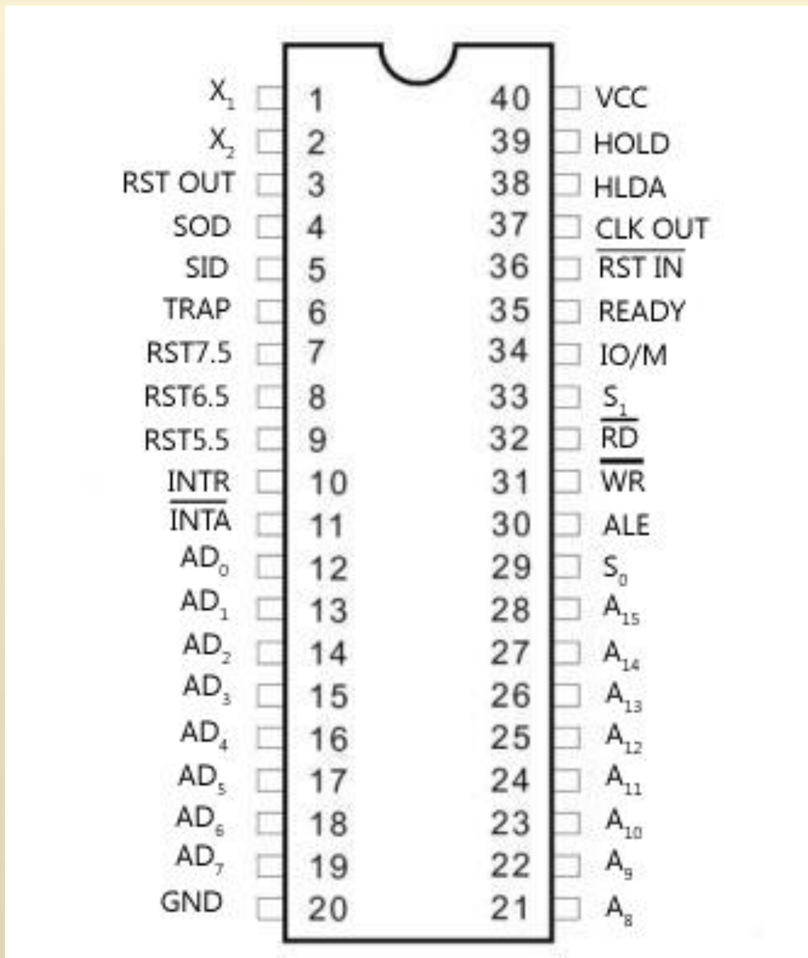
CY-carry flag

The Carry flag is set if there is a carry from addition or a borrow from subtraction or comparison otherwise zero

The 8085 and Its Busses

- ***The 8085 is an 8-bit general purpose* microprocessor that can address *64K Byte of memory*.**
- ***It has 40 pins and uses +5V for power. It can run at a maximum frequency of 3 MHz.***
  - ***The pins on the chip can be grouped into 6 groups:***
    - ***Address* Bus.**
    - ***Data* Bus.**
    - ***Control and Status* Signals.**
    - ***Power supply* and *frequency*.**
    - ***Externally Initiated Signals*.**
    - ***Serial I/O ports*.**

Pin Configuratrion



Power supply

- ***VCC:-Vcc is to be connected to +5V power supply.***
- ***Vss:-Ground reference***

Serial I/O ports.

- ***SID (I/P)and SOD(O/P):-These pins are used for serial data communication.***

Externally Initiated Signals.

- ***Pin 6 to 11:- (I/P)***
- ***These pins are used for interrupt signals. Generally and external devices are connected here which requests the microprocessor to perform a particular task.***

  There are 5 pins for hardware interrupts-
  TRAP, RST7.5, RST 6.5, RST5.5 and INTR
  INTA(O/P) is used for acknowledgement.

- ***Microprocessor sends the acknowledgement to external devices through the INTA pin.***

Externally Initiated Signals

READY (I/P)

READY is used by the microprocessor to check whether a peripheral is ready to accept or transfer data If READY is high then the periphery is ready for data transfer. If not the microprocessor waits until READY goes high.

HOLD☹I/P)

This indicates if any other device is requesting the use of address and data bus

HLDA: (O/P)

HLDA is the acknowledgment signal for HOLD. It indicates whether the HOLD signal is received or not. After the execution of HOLD request, HLDA goes low.

Externally Initiated Signals

- ***RESET IN': (I/P)***

  This pin resets the program counter to 0 and resets interrupt enable and HLDA flip-flops. The CPU is held in reset condition until this pin is high. However the flags and registers won't get affected except for instruction register.

- ***RESET OUT: (O/P)***

  This pin indicates that the CPU has been reset by RESET IN'.

The Address and Data Busses

- ***The address bus has 8 signal lines A8 – A15(O/P) which are unidirectional.***
- ***The other 8 address bits are multiplexed (time shared) with the 8 data bits.***
  - ***So, the bits AD0 – AD7(I/P) are bi-directional and serve as A0 – A7 and D0 – D7 at the same time.***
    - ***During the execution of the instruction, these lines carry the address bits during the early part, then during the late parts of the execution, they carry the 8 data bits.***
  - ***In order to separate the address from the data, we can use a latch to save the value before the function of the bits changes.***

The Control and Status Signals

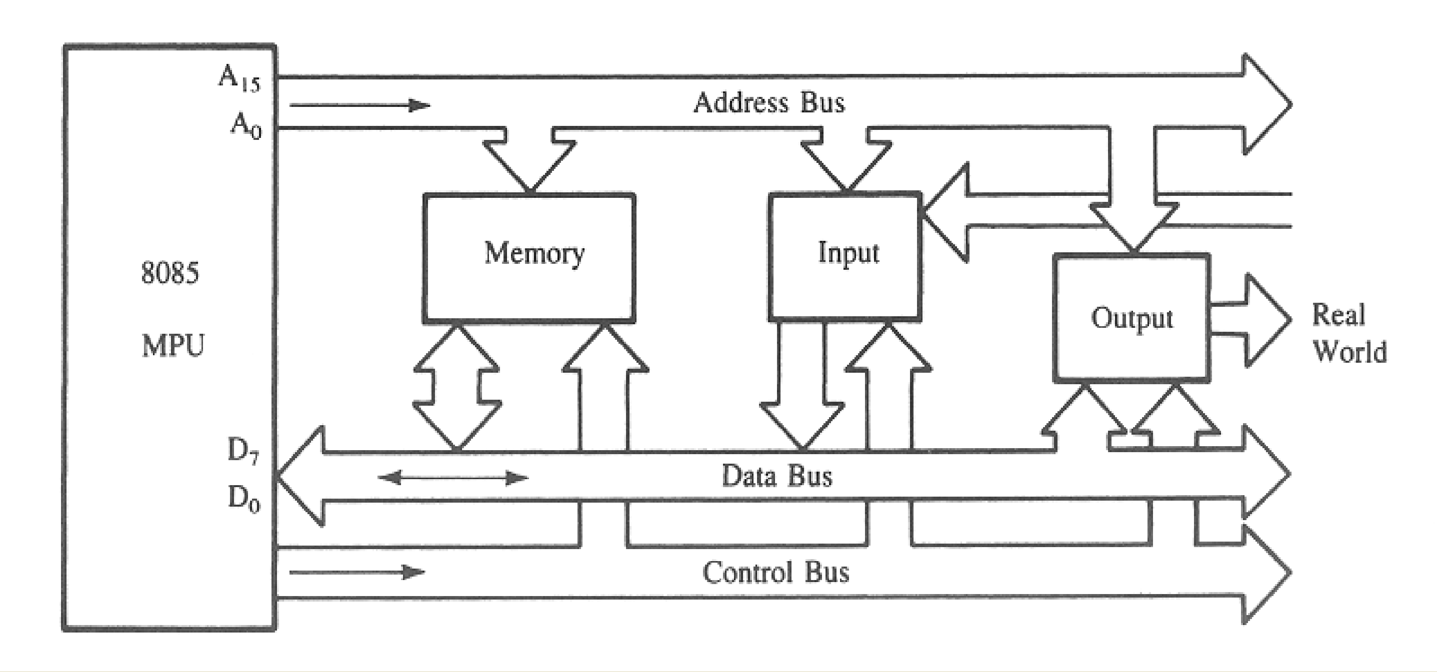
- There are 4 main control and status signals. These are:
  - ALE: (O/P)Address Latch Enable. This signal is a pulse that become 1 when the AD0 – AD7 lines have an address on them. It becomes 0 after that. This signal can be used to enable a latch to save the address bits from the AD lines.
  - RD: (O/P)Read. Active low.
  - WR: (O/P)Write. Active low.
  - IO/M: O/P)This signal specifies whether the operation is a memory operation (IO/M=0) or an I/O operation (IO/M=1).
  - S1 and S0 : (O/P)Status signals to specify the kind of operation being performed .Usually un-used in small systems.

Frequency Control Signals

- *There are 3 important pins in the frequency control group.*
  - *X0 and X1 (I/P)are the inputs from the crystal or clock generating circuit.*
    - *The frequency is internally divided by 2.*
      - *So, to run the microprocessor at 3 MHz, a clock running at 6 MHz should be connected to the X0 and X1 pins.*
  - *CLK (OUT): An output clock pin to drive the clock of the rest of the system.*

The 8085 Bus Structure

The 8-bit 8085 CPU (or MPU – Micro Processing Unit) communicates with the other units using a 16-bit address bus, an 8-bit data bus and a control bus.



Instruction Cycle

- ***Instruction cycle* is defined,**
  as the time required completing the execution of an instruction.

- ***Fetch Operation***

- ***Execute Operation***

Steps For Fetching an Instruction

- ***Lets assume that we are trying to fetch the instruction at memory location 2005. That means that the program counter is now set to that value.***
  - ***The following is the sequence of operations:***
    - ***The program counter places the address value on the address bus and the controller issues a RD signal.***
    - ***The memory's address decoder gets the value and determines which memory location is being accessed.***
    - ***The value in the memory location is placed on the data bus.***
    - ***The value on the data bus is read into the instruction decoder inside the microprocessor.***
    - ***After decoding the instruction, the control unit issues the proper control signals to perform the operation.***

Steps For Execute an Instruction

- ***The following is the sequence of operations:***
- ***The program counter places the address value on the address bus and the controller issues a RD signal.***

Cycles and States

- *From the above discussion, we can define terms that will become handy later on:*
  - *T- State: One subdivision of an operation. A T-state lasts for one clock period.*
    - *An instruction's execution length is usually measured in a number of T-states. (clock cycles).*
  - *Machine Cycle: The time required to complete one operation of accessing memory, I/O, or acknowledging an external request.*
    - *This cycle may consist of 3 to 6 T-states.*
  - *Instruction Cycle: The time required to complete the execution of an instruction.*
    - *In the 8085, an instruction cycle may consist of 1 to 6 machine cycles.*

More on the 8085 machine cycles

- ***The 8085 executes several types of instructions with each requiring a different number of operations of different types. However, the operations can be grouped into a small set.***
- ***The three main types are:***
  - ***Memory Read and Write.***
  - ***I/O Read and Write.***
  - ***Request Acknowledge.***

- ***These can be further divided into various operations (machine cycles).***

Opcode Fetch Machine Cycle

- ***The first step of executing any instruction is the Opcode fetch cycle.***
  - ***In this cycle, the microprocessor brings in the instruction's Opcode from memory.***
    - ***To differentiate this machine cycle from the very similar "memory read" cycle, the control & status signals are set as follows:***
      - ***IO/M=0, s0 and s1 are both 1.***
  - ***This machine cycle has four T-states.***
    - ***The 8085 uses the first 3 T-states to fetch the opcode.***
    - ***T4 is used to decode and execute it.***
  - ***It is also possible for an instruction to have 6 T-states in an opcode fetch machine cycle.***

Memory Read Machine Cycle

- ***The memory read machine cycle is exactly the same as the opcode fetch except:***
  - ***It only has 3 T-states***
  - ***The s0 signal is set to 0 instead.***

The Memory Read Machine Cycle

- *To understand the memory read machine cycle, let's study the execution of the following instruction:*
  - *MVI A, 32*
- *In memory, this instruction looks like:*
  - *The first byte 3EH represents the opcode for loading a byte into the accumulator (MVI A), the second byte is the data to be loaded.*

2000H	3E
2001H	32

- *The 8085 needs to read these two bytes from memory before it can execute the instruction. Therefore, it will need at least two machine cycles.*
  - *The first machine cycle is the opcode fetch discussed earlier.*
  - *The second machine cycle is the Memory Read Cycle.*
  - *Figure 3.10 page 83.*

Machine Cycles vs. Number of bytes in the instruction

- *Machine cycles and instruction length, do not have a direct relationship.*
  - *To illustrate lets look at the machine cycles needed to execute the following instruction.*
    - *STA 2065H*
    - *This is a 3-byte instruction requiring 4 machine cycles and 13 T-states.*
    - *The machine code will be stored in memory as shown to the right*
    - *This instruction requires the following 4 machine cycles:*
      - *Opcode fetch to fetch the opcode (32H) from location 2010H, decode it and determine that 2 more bytes are needed (4 T-states).*
      - *Memory read to read the low order byte of the address T-states).*
      - *Memory read to read the high order byte of the addres T-states).*
      - *A memory write to write the contents of the accumulator into the memory location.*

32H	2010H
65H	2011H
20H	2012H

The Memory Write Operation

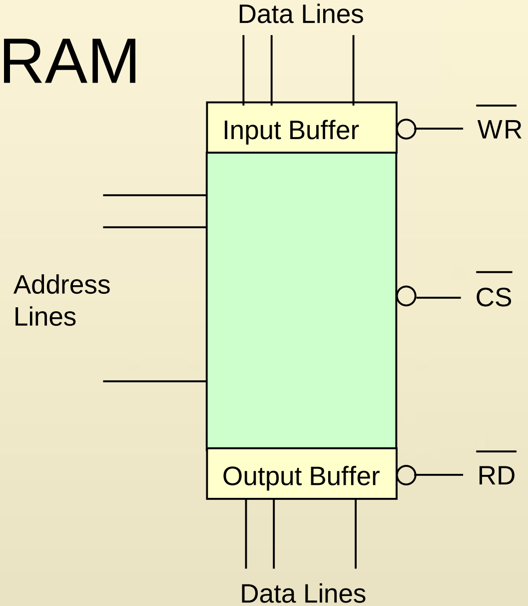
- ***In a memory write operation:***
  - ***The 8085 places the address (2065H) on the address bus***
  - ***Identifies the operation as a memory write (IO/M=0, s1=0, s0=1).***
  - ***Places the contents of the accumulator on the data bus and asserts the signal WR.***
  - ***During the last T-state, the contents of the data bus are saved into the memory location.***

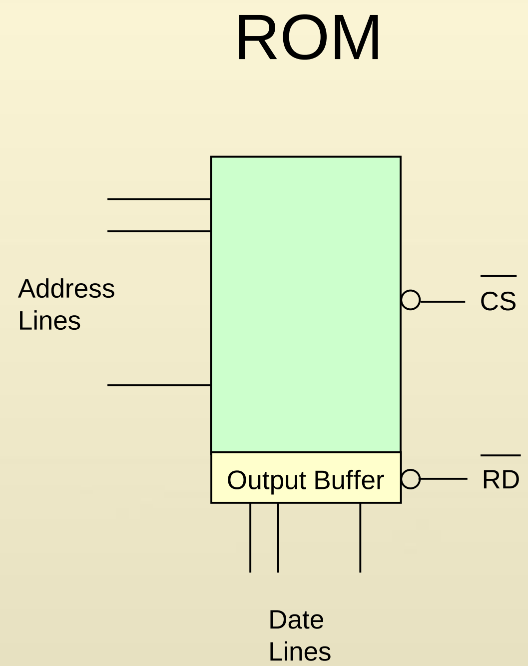
Memory interfacing

- ***There needs to be a lot of interaction between the microprocessor and the memory for the exchange of information during program execution.***
  - ***Memory has its requirements on control signals and their timing.***
  - ***The microprocessor has its requirements as well.***

- ***The interfacing operation is simply the matching of these requirements.***

Memory structure & its requirements





- ***The process of interfacing the above two chips is the same.***
  - ***However, the ROM does not have a WR signal.***

Interfacing Memory

- *Accessing memory can be summarized into the following three steps:*
    - *Select the chip.*
    - *Identify the memory register.*
    - *Enable the appropriate buffer.*

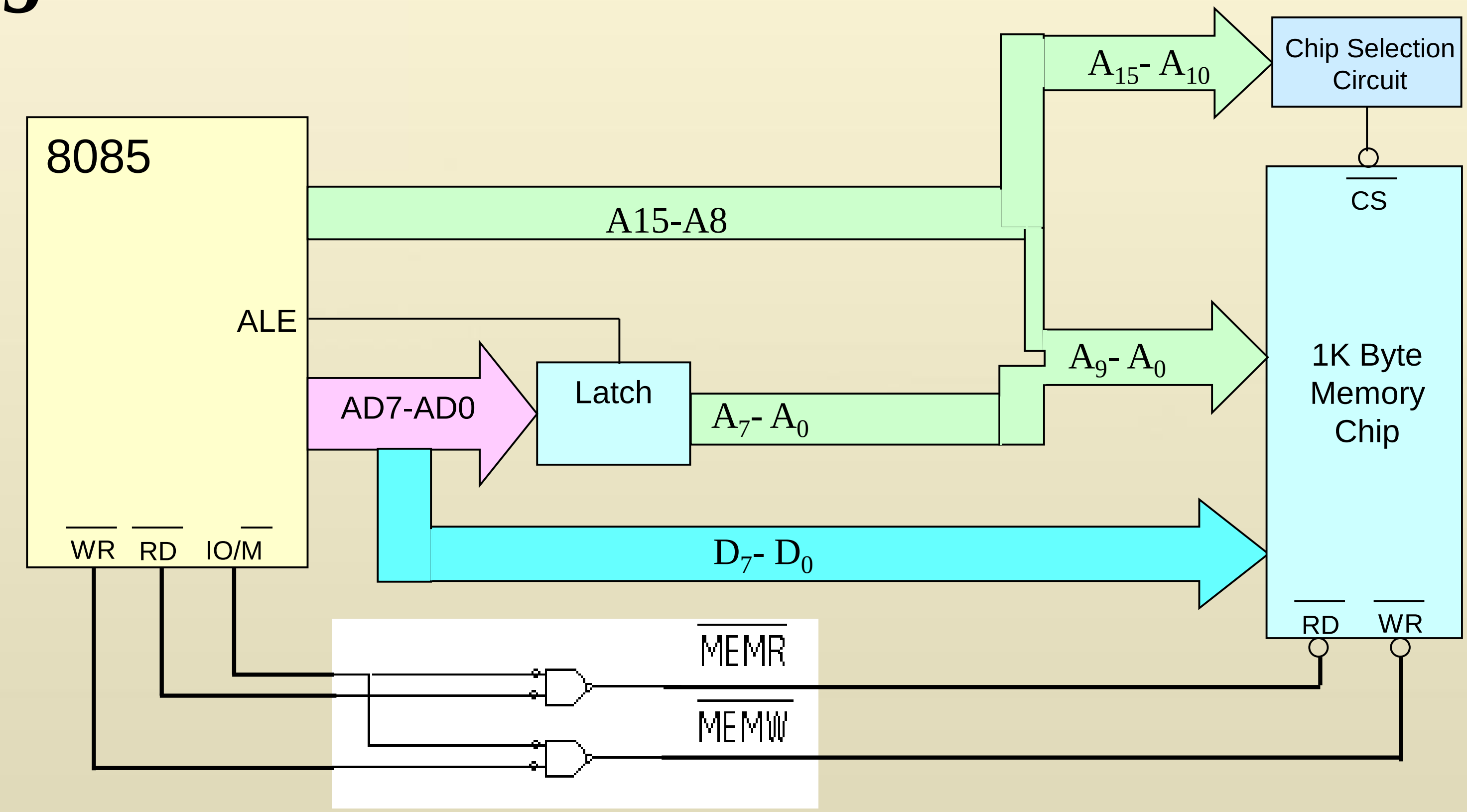
- *Translating this to microprocessor domain:*
    - *The microprocessor places a 16-bit address on the address bus.*
    - *Part of the address bus will select the chip and the other part will go through the address decoder to select the register.*
    - *The signals IO/M and RD combined indicate that a memory read operation is in progress. The MEMR signal can be used to enable the RD line on the memory chip.*

Address decoding

- ***The result of address decoding is the identification of a register for a given address.***
  - ***A large part of the address bus is usually connected directly to the address inputs of the memory chip.***
  - ***This portion is decoded internally within the chip.***
  - ***What concerns us is the other part that must be decoded externally to select the chip.***
  - ***This can be done either using logic gates or a decoder.***

The Overall Picture

- *Putting all of the concepts together, we get:*



8085 Instruction Set

- **Data transfer operations**

- Between registers
- Between memory location and a register
- Direct write to a register / memory
- Between I/O device and accumulator

- **Arithmetic operations (ADD, SUB, INR, DCR)**
- **Logic operations**
- **Branching operations (JMP, CALL, RET)**

8085 Instruction Types

ONE-BYTE INSTRUCTIONS

A 1-byte instruction includes the opcode and the operand in the same byte. For example:

Task	Opcode	Operand*	Binary Code	Hex Code
Copy the contents of the accumulator in register C.	MOV	C,A	0100 1111	4FH
Add the contents of register B to the contents of the accumulator.	ADD	B	1000 0000	80H
Invert (complement) each bit in the accumulator.	CMA		0010 1111	2FH

8085 Instruction Types

TWO-BYTE INSTRUCTIONS

In a 2-byte instruction, the first byte specifies the operation code and the second byte specifies the operand. For example:

Task	Opcode	Operand	Binary Code	Hex Code	
Load an 8-bit data byte in the accumulator.	MVI	A,Data	0011 1110	3E	First Byte
			DATA	Data	Second Byte

8085 Instruction Types

THREE-BYTE INSTRUCTIONS

In a 3-byte instruction, the first byte specifies the opcode, and the following two bytes specify the 16-bit address. Note that the second byte is the low-order address and the third byte is the high-order address. For example:

Task	Opcode	Operand	Binary Code	Hex Code	
Transfer the program sequence to the memory location 2085H.	JMP	2085H	1100 0011	C3*	First Byte
			1100 0011	85	Second Byte
			0010 0000	20	Third Byte

Simple Data Transfer Operations

MOV	Rd,Rs*	Move ☐ This is a 1-byte instruction ☐ Copies data from source register Rs to destination register Rd
MVI	R,8-bit	Move Immediate ☐ This is a 2-byte instruction ☐ Loads the 8 bits of the second byte into the register specified

Examples:

MOV	B,A	47	From ACC to REG
MOV	C,D	4A	Between two REGs
MVI	D,47	16	Direct-write into REGD

Simple Data Transfer Operations

OUT	8-bit port address	Output to Port ☐ This is a 2-byte instruction ☐ Sends (copies) the contents of the accumulator (A) to the output port specified in the second byte
IN	8-bit port address	Input from Port ☐ This is a 2-byte instruction ☐ Accepts (reads) data from the input port specified in the second byte, and loads into the accumulator

Example:

- OUT 05 D3
  05

  Contents of ACC sent to output port number 05.

Simple Memory Access Operations

LDA: Load Accumulator Direct

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
LDA	16-bit address	3	4	13	3A

Description The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. The contents of the source are not altered. This is a 3-byte instruction; the second byte specifies the low-order address and the third byte specifies the high-order address.

Flags No flags are affected.

Example Assume memory location 2050H contains byte F8H. Load the accumulator with the contents of location 2050H.

Instruction: LDA 2050H Hex Code: 3A 50 20 (note the reverse order)

A | F8 | X | F 2050 | F8 |

Simple Memory Access Operations

STA: Store Accumulator Direct

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
STA	16-bit	3	4	13	32

Description The contents of the accumulator are copied to a memory location specified by the operand. This is a 3-byte instruction; the second byte specifies the low-order address and the third byte specifies the high-order address.

Flags No flags are affected.

Example Assume the accumulator contains 9FH. Load the accumulator contents into memory location 2050H.

Instruction: STA 2050H Hex Code: 32 50 20

Register contents before instruction				Memory contents after instruction	
A	9F	XX	F	2050	9F

Arithmetic Operations

ADD	R†	Add ☐ This is a 1-byte instruction ☐ Adds the contents of register R to the contents of the ac- cumulator
ADI	8-bit	Add Immediate ☐ This is a 2-byte instruction ☐ Adds the second byte to the contents of the accumulator
SUB	R†	Subtract ☐ This is a 1-byte instruction ☐ Subtracts the contents of register R from the contents of the accumulator
SUI	8-bit	Subtract Immediate

Arithmetic Operations

INR	R*	Increment ☐ This is a 1-byte instruction ☐ Increases the contents of register R by 1 *Caution:* All flags except the CY are affected
DCR	R*	Decrement ☐ This is a 1-byte instruction ☐ Decreases the contents of register R by 1 *Caution:* All flags except the CY are affected

Arithmetic Operations

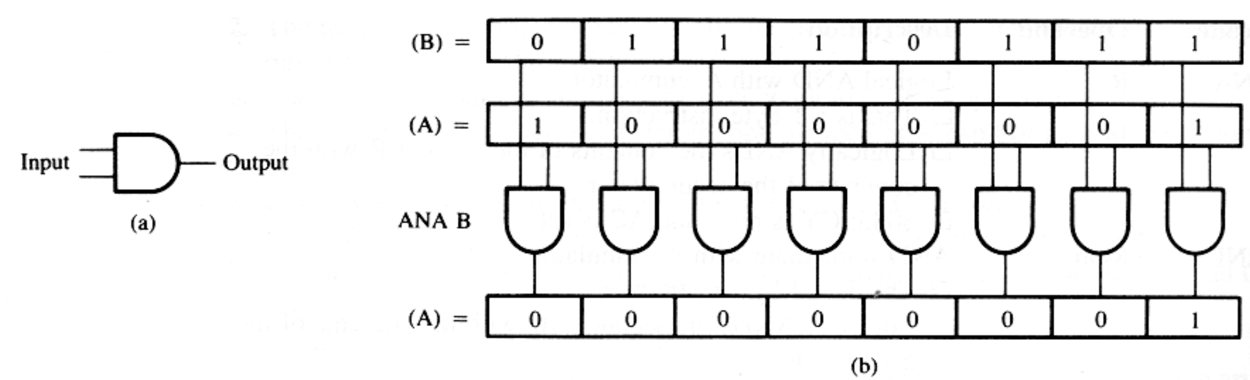
Instruction ADD C

			CY	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
(A)	:	93H =		1	0	0	1	0	0	1	1	
	+											
(C)	:	B7H =		1	0	1	1	0	1	1	1	
			1		1	1		1	1	1		Carry
SUM (A)	:	[1] 4AH =	[1]	0	1	0	0	1	0	1	0	
		CY										

Flag Status:† S = 0, Z = 0, CY = 1

Overview of Logic Operations

ANA:	AND	Logically AND the contents of a register.
ANI :	AND Immediate	Logically AND 8-bit data.
ORA:	OR	Logically OR the contents of a register.
ORI :	OR Immediate	Logically OR 8-bit data.
XRA:	X-OR	Exclusive-OR the contents of a register.
XRI :	X-OR Immediate	Exclusive-OR 8-bit data.



Logic Operations

ANA	R	Logical AND with Accumulator ☐ This is a 1-byte instruction ☐ Logically ANDs the contents of the register R with the contents of the accumulator ☐ 8085: CY is reset and AC is set
ANI	8-bit	AND Immediate with Accumulator ☐ This is a 2-byte instruction ☐ Logically ANDs the second byte with the contents of the accumulator

Logic Operations

ORA	R	Logically OR with Accumulator ☐ This is a 1-byte instruction ☐ Logically ORs the contents of the register R with the contents of the accumulator
ORI	8-bit	OR Immediate with Accumulator ☐ This is a 2-byte instruction ☐ Logically ORs the second byte with the contents of the accumulator

Logic Operations

XRA	R	Logically Exclusive-OR with Accumulator ☐ This is a 1-byte instruction ☐ Exclusive-ORs the contents of register R with the contents of the accumulator
XRI	8-bit	Exclusive-OR Immediate with Accumulator ☐ This is a 2-byte instruction ☐ Exclusive-ORs the second byte with the contents of the accumulator
CMA		Complement Accumulator ☐ This is a 1-byte instruction that complements the contents of the accumulator ☐ No flags are affected

Branching Operations

INSTRUCTION

Opcode	Operand	Description
JMP	16-bit	Jump ☐ This is a 3-byte instruction ☐ The second and third bytes specify the 16-bit memory address. However, the second byte specifies the low-order and the third byte specifies the high-order memory address

Note: This is an unconditional jump operation.
It will always force the program counter to a fixed memory address continuous loop !

Branching Operations

Opcode	Operand	Description
JC	16-bit	Jump On Carry (if result generates carry and CY = 1)
JNC	16-bit	Jump On No Carry (CY = 0)
JZ	16-bit	Jump On Zero (if result is zero and Z = 1)
JNZ	16-bit	Jump On No Zero (Z = 0)
JP	16-bit	Jump On Plus (if D_7 = 0, and S = 0)
JM	16-bit	Jump On Minus (if D_7 = 1, and S = 1)
JPE	16-bit	Jump On Even Parity (P = 1)
JPO	16-bit	Jump On Odd Parity (P = 0)

Conditional jump operations are very useful for decision making during the execution of the program.

Direct Memory Access Operations

LDA: Load Accumulator Direct

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
LDA	16-bit address	3	4	13	3A

Description The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. The contents of the source are not altered. This is a 3-byte instruction; the second byte specifies the low-order address and the third byte specifies the high-order address.

Flags No flags are affected.

Example Assume memory location 2050H contains byte F8H. Load the accumulator with the contents of location 2050H.

Instruction: LDA 2050H Hex Code: 3A 50 20 (note the reverse order)

A [F8 | X] F 2050 [F8]

Direct Memory Access Operations

STA: Store Accumulator Direct

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
STA	16-bit	3	4	13	32

Description The contents of the accumulator are copied to a memory location specified by the operand. This is a 3-byte instruction; the second byte specifies the low-order address and the third byte specifies the high-order address.

Flags No flags are affected.

Example Assume the accumulator contains 9FH. Load the accumulator contents into memory location 2050H.

Instruction: STA 2050H Hex Code: 32 50 20

Register contents before instruction	Memory contents after instruction
A [9F | XX] F	2050 [9F]

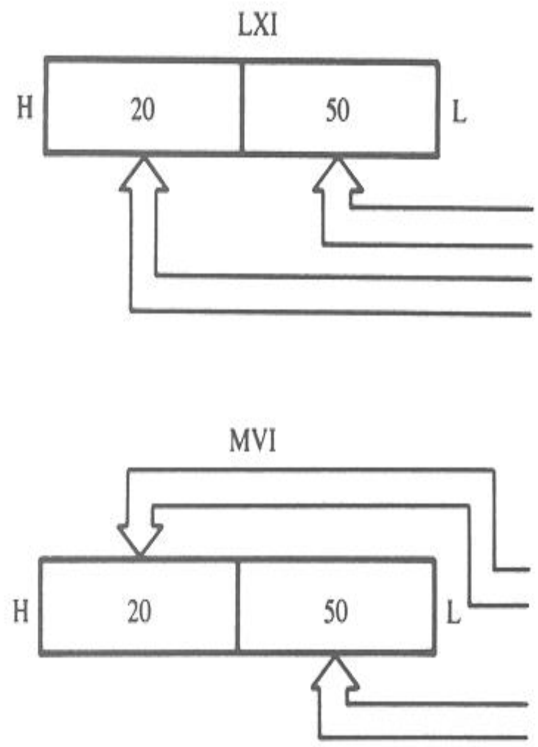
Indirect Memory Access Operations

- *Use a register PAIR as an address pointer !*
- *We can define memory access operations using the memory location (16 bit address) stored in a register pair: BC, DE or HL.*
- *First, we have be able to **load** the register pairs.*

  LXI B, (16-bit address)
  LXI D, (16-bit address)
  LXI H, (16-bit address)

- *We can also increment / decrement register pairs.*

Loading Register Pairs



Machine Code	Mnemonics	Comments
21	LXI H,2050H	;Load HL registers
50*		;50H in L register and
20		;20H in H register
26	MVI H,20H	;Load 20H in register H
20		
2E	MVI L,50H	;Load 50H in register L
50		

Interrupts in 8085

In many real-time operations, the microprocessor should be able to receive an external asynchronous signal (interrupt) while it is running a routine.

When the interrupt signal arrives:

- The processor will break its routine
- Go to a different routine (service routine)
- Complete the service routine
- Go back to the "regular" routine

Interrupts in 8085

In order to execute an interrupt routine, the processor:

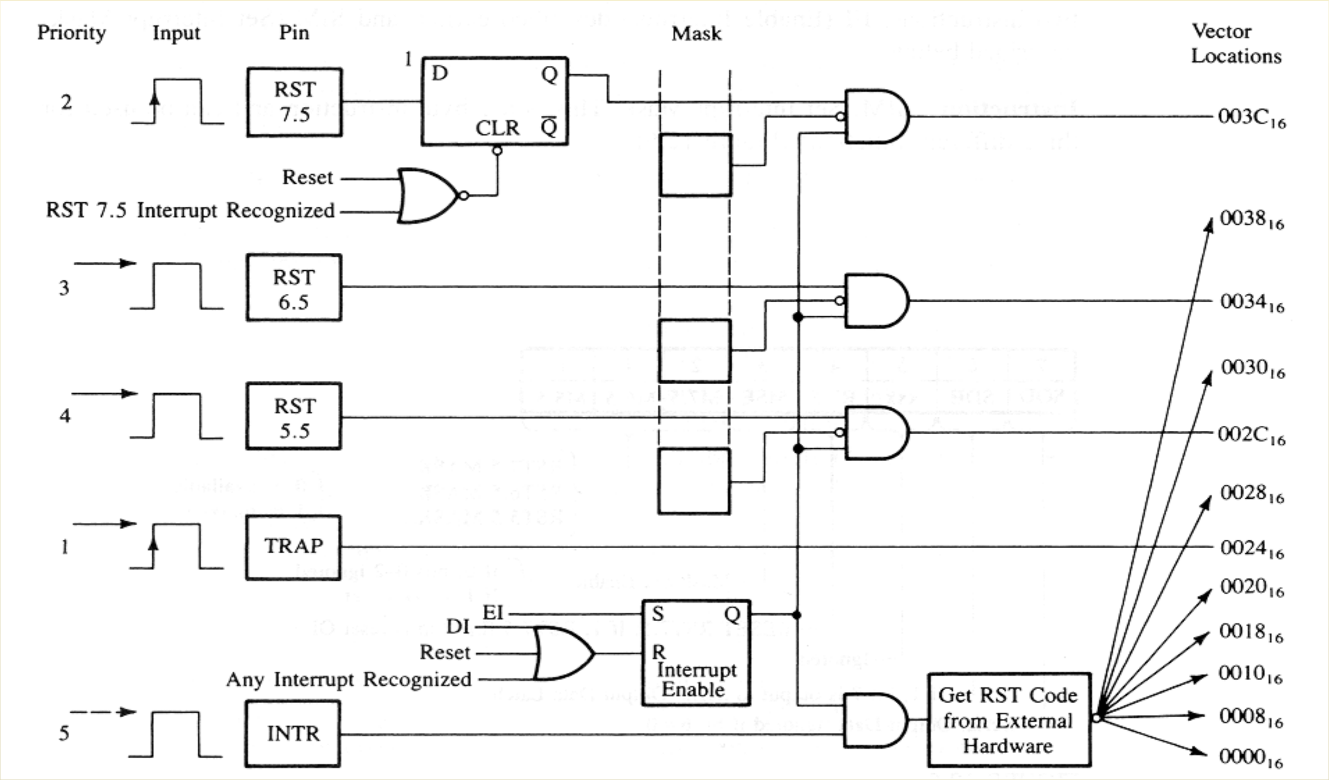
- Should be able to accept interrupts (interrupt enable)
- Save the last content of the program counter (PC)
- Know where to go in program memory to execute the service routine
- Tell the outside world that it is executing an interrupt
- Go back to the saved PC location when finished.

Vectored Interrupts

There are four other interrupt inputs in 8085 that transfer the operation immediately to a specific address:

- TRAP: go to 0024
- RST 7.5: go to 003C
- RST 6.5 0034
- RST 5.5 002C
- RST 7.5, RST 6.5 and RST 5.5 are maskable interrupts, they are acknowledged only if they are not masked !

Vectored Interrupts



SIM: Set Interrupt Mask