Microprocessor, Microcomputer and Microcontroller

Microprocessor: The microprocessor is a clock driven general purpose programmable device that performs arithmetic or logical operations on numbers according to the program stored in memory and then gives us result.

It is a Silicon chip that has ALU, register array and control circuit.

Microcontroller: A microcontroller is a programmable device that includes microprocessor, memory and I/O signal lines **on a single chip**,

Microcomputer: A microcomputer consists of microprocessor as CPU and externally connected memory, input device and output device.

Evolution of Microprocessors

1. First Generation – 4bit Microprocessors

The Intel corporation came out with the first generation of microprocessors in **1971**. They were 4-bit processors namely **Intel 4004.** The speed of the processor was **740 kHz** taking **60k instructions per second**. It had **2300 transistors** and 16 pins inside.

Built on a single chip, it was useful for simple arithmetic and logical operations. A control unit was there to understand the instructions from memory and execute the tasks.

2. Second Generation – 8bit Microprocessor

The second generation began in **1973 by Intel** as the first 8 – bit microprocessor. It was useful for arithmetic and logic operations on 8-bit words.

The first processor was **8008** with a clock speed of **500kHz** and **50k** instructions per second.

Followed by an **8080** microprocessor in **1974** with a speed of **2 MHz** and **60k** instruction per second.

Lastly came the **8085** microprocessor in **1976** having an ability of **769230** instruction per second with 3 **MHz** speed.

3. Third Generation – 16bit Microprocessor

The third generation began with **8086-88** microprocessors in **1978** with **4.77, 8 & 10 MHz** speed and 2.5 million instructions per second.

Other important inventions were **Zilog Z800, and 80286**, which came out in **1982** and could read **4 million** instructions per second with 68 pins inside.

4. Fourth Generation – 32bit Microprocessors

Intel was still the leader as many companies came out with 32-bit microprocessors around **1986.** Their clock speed was between **16 MHz to 33 MHz** with **275k** transistors inside.

One of the first ones was the Intel **80486** microprocessor of **1986** with **16-100MHz** clock speed and **1.2 Million transistors** with **8 KB** of cache memory.

5. Fifth Generation – 64bit Microprocessors

Began in **1995**, the Pentium processor was one of the first 64-bit processors with **1.2 GHz to 3 GHz** clock speed. There were **291 Million** transistors and **64k billion instruction** per second.

Followed by **i3, i5, i7** microprocessors in **2007, 2009, 2010** respectively.

Case study of Evolution for your reference only:

Processor	Year	Feature Size (μm)	Transistors	Frequency (MHz)	Word Size	Power (W)	Cache (L1 / L2 / L3)	Package
4004	1971	10	2.3k	0.75	4	0.5	none	16-pin DIP
8008	1972	10	3.5k	0.5–0.8	8	0.5	none	18-pin DIP
8080	1974	6	6k	2	8	0.5	none	40-pin DIP
8086	1978	3	29k	5–10	16	2	none	40-pin DIP
80286	1982	1.5	134k	6–12	16	3	none	68-pin PGA
Intel386	1985	1.5–1.0	275k	16–25	32	1–1.5	none	100-pin PGA
Intel486	1989	1–0.6	1.2M	25–100	32	0.3–2.5	8K	168-pin PGA
Pentium	1993	0.8–0.35	3.2–4.5M	60–300	32	8–17	16K	296-pin PGA
Pentium Pro	1995	0.6–0.35	5.5M	166–200	32	29–47	16K / 256K+	387-pin MCM PGA
Pentium II	1997	0.35–0.25	7.5M	233–450	32	17–43	32K / 256K+	242-pin SECC
Pentium III	1999	0.25–0.18	9.5–28M	450–1000	32	14–44	32K / 512K	330-pin SECC2
Pentium 4	2000	180–65 nm	42–178M	1400–3800	32/64	21–115	20K+ / 256K+	478-pin PGA
Pentium M	2003	130–90 nm	77–140M	1300–2130	32	5–27	64K / 1M	479-pin FCBGA
Core	2006	65 nm	152M	1000–1860	32	6–31	64K / 2M	479-pin FCBGA
Core 2 Duo	2006	65–45 nm	167–410M	1060–3160	32/64	10–65	64K / 4M+	775-pin LGA
Core i7	2008	45 nm	731M	2660–3330	32/64	45–130	64K / 256K / 8M	1366-pin LGA
Atom	2008	45 nm	47M	800–1860	32/64	1.4–13	56K / 512K+	441-pin FCBGA

Types of microprocessor:

There are basically 5 kinds of microprocessors namely:

- **Complex Instruction Set Microprocessors:** They are also called as CISM in short and they categorize a micro processor in which orders can be executed together along with other low level activities. It mainly performs the task of uploading, downloading and recalling data into and from the memory card. Apart from that it also does complex mathematical calculations within a single command.

  Their instructions have more than one clock cycle. Some examples are – Intel 386 & 486, Pentium, etc.

- **Reduced Instruction Set Microprocessor:** This processor is also called as RISC. RISC is to carry out small specific commands at a faster rate and high optimization. The instruction set is shorter due to simple commands and the same length. They reduce memory references by adding registers.

RISC follow pipelining which leads to overlapping of instruction fetching and execution. They take one CPU cycle to execute mostly. Some examples are – AMD K6, and K7, etc.

- **Superscalar Processors:** The superscalar processor supports performing multiple tasks simultaneously. They are commonly present in ALUs or multipliers as they are capable of carrying multiple commands. They use different operational units for transmitting instructions inside the processor.

- **Application Specific Integrated Circuit:** This processor is also known as ASIC. They are used for specific purposes that comprises of automotive emissions control or personal digital assistants computer. This kind of processor is made with proper specification but apart from that it can also be made using the off the shelf gears.

- **Digital Signal Multiprocessors:** Also called as DSP's, these are used for encoding and decoding videos or to convert the digital and video to analog and analog to digital. They need a microprocessor that is excellent in mathematical calculations. The chips of this processor are employed in SONAR, RADAR, home theaters audio gears, Mobile phones and TV set top boxes. Companies like Intel, Motorola, DEC, etc. have made many such microprocessors like this.

Microprocessor Architecture and its operation:

The microprocessor is a programmable logic device, designed with registers, flip-flops. The microprocessor has a set of instructions designed internally, to manipulate data and communicate with peripherals. This process of data manipulation and communication is determined by the logic design of the microprocessor, called the architecture.

The microprocessor can be programmed to perform functions on given data by selecting necessary instructions from its set.

These instructions are given to the microprocessor by writing them into its memory. Writing (Or entering) instruction and data is done through an input device such as a keyboard.

The microprocessor reads or transfers each instruction one at a time, matches it with its instruction set, and performs the data manipulation indicated by the instruction.

The result can be stored in memory or sent to such output devices as LEDs or a CRT terminal.

The microprocessor fetches the first instruction from its memory sheet, decodes it, and executes that instruction.

All the various functions performed by the microprocessor can be classified in three general categories

1. Microprocessor-initiated operations
2. Internal data operations
3. Peripheral (or externally) initiated operations.

To perform these functions, the microprocessor requires a group of logic circuits and a set of signals called control signals.

1. Microprocessor-Initiated Operation

The MPU performs primarily four operations.

1. Memory Read: Reads data from memory.

2. Memory Write: Writes data into memory.

3. I/O read: Accepts data from input devices.

4. I/O Write: Sends data to output devices.

All these operations are part of the communication process between the MPU and peripheral devices (including memory).

To communicate with a peripheral (or a memory location), the MPU needs to perform the following steps:

Step 1 : Identify the peripheral or the memory location (with its address).

Step 2: Transfer data

Step 3: Provide timing or synchronization signals. The 8085/8080A MPU performs these functions using three sets of communication lines called buses : The address bus, the data bus, and the control bus.

2. Internal operations

The internal architecture of the 8085/8080A microprocessor determines how and what operations can be performed with the data. These operations are

1. Store 8-bits data.

2. Perform arithmetic and logical operations.

3. Test for conditions.

4. Sequence the execution of instructions.

5. Store data temporarily during execution in the defined R/W memory locations called the stack.

To perform these operations, the microprocessor requires registers, an arithmetic logic unit (ALU) and control logic, and internal buses (path for information flow).

3. **External initiated operations:**

8085 microprocessor support some Externally initiated operations, which are also known as Peripheral operations. Different external input/output devices or signals can initiate these types of operations. In the 8085 microprocessor chip, their individual pins are assigned.

Following are some externally initiated operations:

- RESET –

This RESET key is used to clear the program counter and update it with the 0000H memory location. When this RESET pin is activated by any external key, then all the internal operations are suspended for that time. After that, the execution of the program can begin at the zero memory address.

Interrupt –

8085 microprocessor chips have some pins for interrupt like INTR, RST 5.5, RST 6.5, RST 7.5 and TRAP. The microprocessor can be interrupted from the normal instructions and asked to perform some other emergency operations, which are also known as Interrupt Service Routine (ISR). The microprocessor resumes its operation after the completion of ISR.

READY –

The 8085 microprocessor has a pin called READY. If the signal at this READY pin is in a low state then the microprocessor enters into the Wait state. The Input/Output devices that are connected to the microprocessor are of different speeds, which is need to be synchronized with the speed of the microprocessor. This signal is used mainly to synchronize slower external devices with the microprocessor.

HOLD –

When the HOLD pin is activated by an external signal, the microprocessor relinquishes control over the buses and allows the external peripheral to use them. For example, the HOLD signal is used as Direct memory access (DMA) data transfer. In this DMA, the external Input/Output devices are directly communicating with the memory without interfering with the processor every time.

Instruction and Data flow from Memory to MPU

The primary function of memory is to store instructions and data and to provide that information to the MPU whenever the MPU requests it.
The MPU requests the information by sending the address of a specific memory register on the address bus and enables the data flow by sending the control signal.

Example :
The instruction code 0100 1111 (4FH) is stored in memory location 2005H. Illustrate the data flow and list the sequence of events when the instruction code is fetched by the MPU.

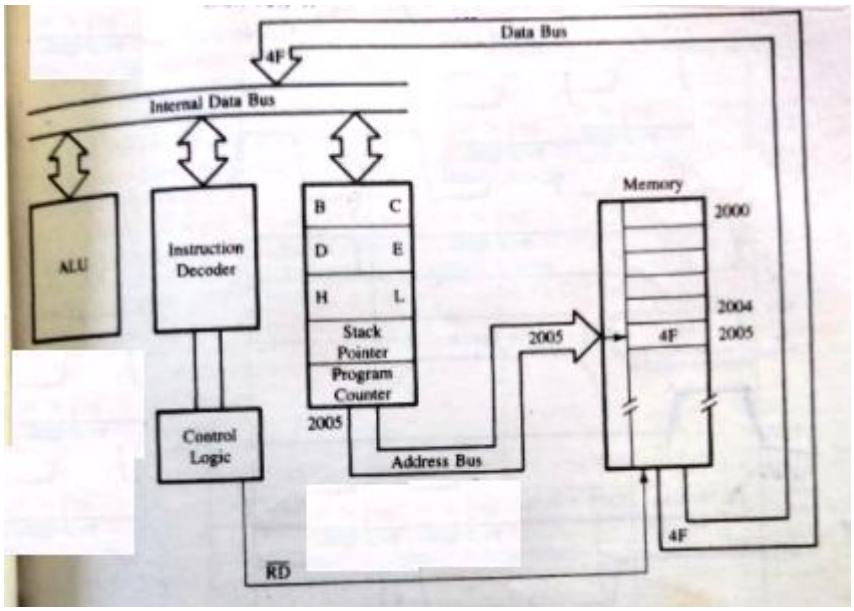
Opcode Fetch Cycle: To fetch the byte (opcode), the MPU needs to identify the memory location 2005 and enable the instrucion flow from memory.

Step 1: The microprocessor places the 16-bit memory address from the program counter (PC) on the address bus
Step 2: The control unit sends the control signal RD' to enable the memory chip
Step 3: The byte from the memory location is placed on the data bus.
Step 4: The byte is opcode so placed in the instruction decoder of the microprocessor, and the task is carried out according to the instruction.
Timing diagram shows this task is completed in 4 T-states.

Example :
Now, the data 0100 1111 (4FH) is stored in memory location 2005H. Illustrate the data flow and list the sequence of events when the byte is fetched by the MPU.

Data Fetch Cycle: To fetch the byte (data not opcode), the MPU needs to identify the memory location 2005 and enable the data flow from memory.

Step 1: The microprocessor places the 16-bit memory address from the program counter (PC) on the address bus
Step 2: The control unit sends the control signal RD' to enable the memory chip
Step 3: The byte from the memory location is placed on the data bus.
Step 4: The byte is data and comes to respective register.



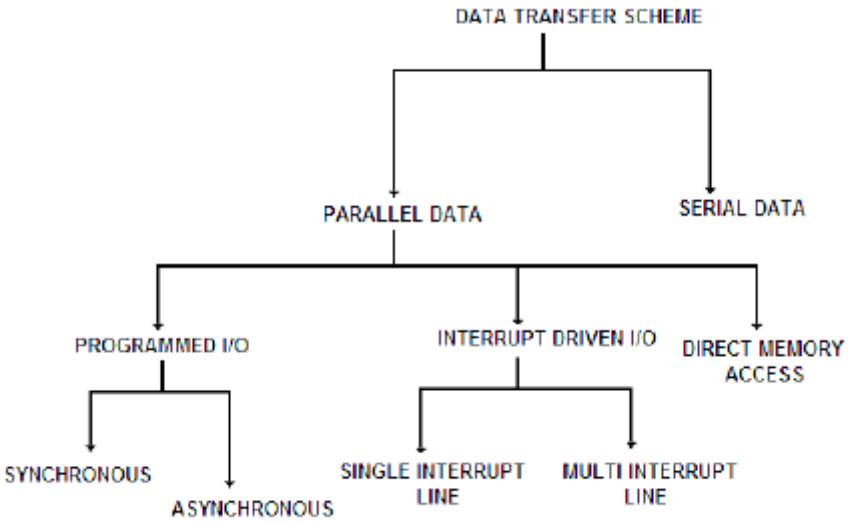
Timing diagram shows this task is completed in 3 T-states.

Data Transfer Schemes

Need for data transfer scheme

Data transfer may take place between microprocessor and memory, microprocessor and I/O devices and memory & I/O devices. We can get very smooth data communication between microprocessor and memory because the speed of the memory is almost compatible with the speed of 8085 microprocessor as same technology is used in the manufacturing of memory and microprocessor. So it has all most same bus speed.

Now the main concern is for data transfer between the microprocessor and I/O devices. In that case, the main problems arise due to mismatch of the speed of the I/O devices and the speed of microprocessor or memory. So we have to do something to overcome that problem. To overcome this problem of speed mismatching between the microprocessor and I/O devices, we introduce **data transfer schemes** of 8085 microprocessor.



Serial I/O mode transfer

Serial I/O mode transfer is useful in long distance communication where parallel data transfer becomes expensive.

In serial I/O mode transfer a single bit of data appears on a single line at a time. For serial I/O data transmission mode, 8-bit parallel word is converted to a stream of eight serial bit using parallel-to-serial converter. Similarly, in serial reception of data, the microprocessor receives a stream of 8-bit one by one. After receiving serial data, it converted to 8- bit parallel word using serial-to-parallel converter. For this purpose data transfer schemes of 8085 microprocessor are introduced.

Parallel data transfer scheme

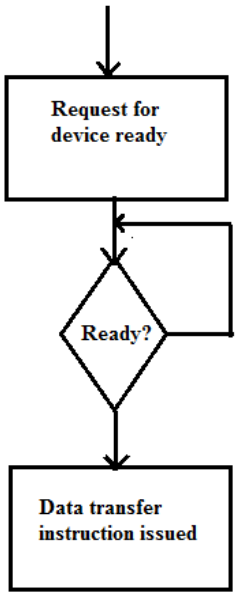
Parallel data transfer scheme is faster than serial I/O transfer. In parallel data transfer mode, 8-bit data send all together with 8 parallel wires. In 8085 microprocessor mainly three types of parallel data transfer scheme we observed. Those are

1. Programmed I/O Data Transfer
2. Interrupt Driven I/O Data Transfer
3. Direct Memory Access (DMA) Data Transfer.

The 8085 microprocessor is a parallel device. That means it transfers eight bits of data simultaneously over eight data lines (parallel I/O mode). However in many situations, the parallel I/O mode is either impractical or impossible. For example, parallel data communication over a long distance becomes very expensive. So using different transfer scheme we can connect keyboard, external memory, any input analogue sensor, motor etc with microprocessor.

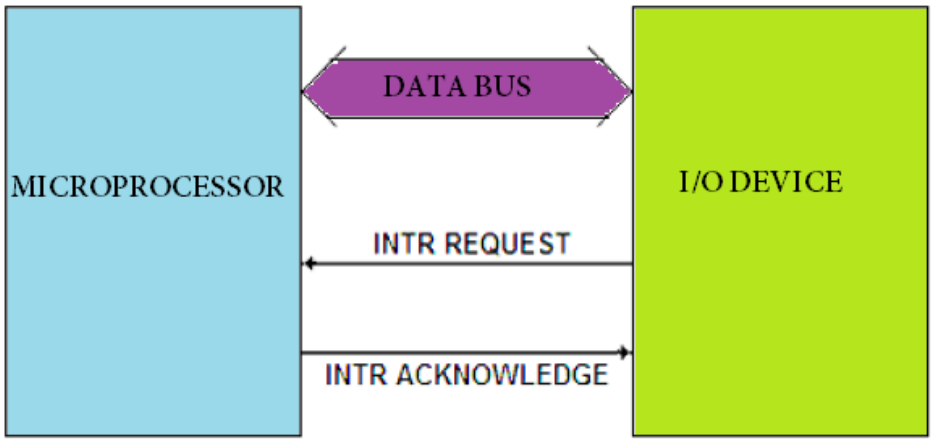
1. Programmed 1/0 Data Transfer

The user program controls the data transfer in the programmed 1/0 method. The data transfer can be synchronous or asynchronous depending upon the type and the speed of the 1/O devices. When the speed of the I/O devices matches the speed of the microprocessor the synchronous type of data transfer can be achieved. Since the speed of the I/O devices is the same as that of the microprocessor. the microprocessor issues commands for the data transter the moment it is required by the microprocessor. It does not have to wait for the availability of the data. On the contrary when the speed of the I/O devices is slower than the speed of the microprocessor, asynchronous type of data transfer method is used. In this method, the microprocessor has to check every now and then the status of the I/O devices for the availability of the data to be transferred to the microprocessor or whether the I/O device is free to accept the data from the microprocessor. A simple flowchart for asynchronous data transfer is depicted in Fig. shown below:



2. Interrupt I/O Data Transfer

In programmed data transfer scheme the microprocessor is busy all the time in checking for the availability of data from the slower 1/O devices and also whether the I/O device is ready to accept the data from the microprocessor. The simple hardware solution of this problem could be to inform the I/O device for the data transfer whenever the I/O devices become ready, This is achieved by interrupting the microprocessor. Interrupt is the hardware facilities provided on the uP chip to handle higher priority objects first. It is explained in Fig. below:



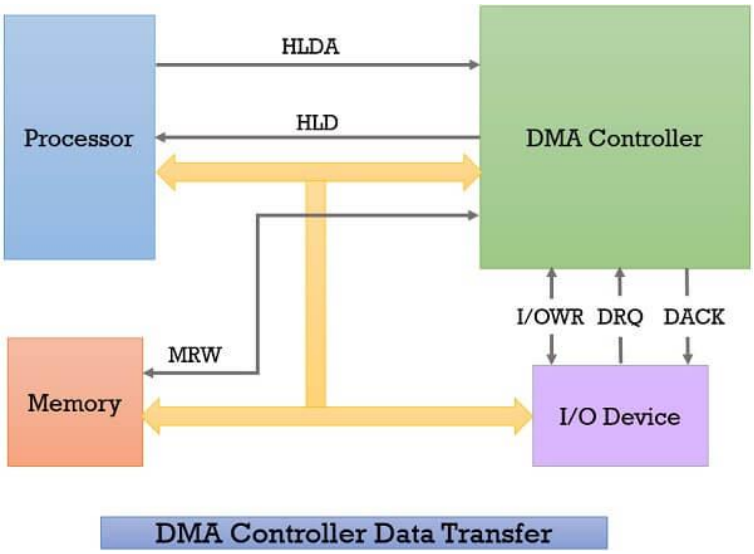
Microprocessor continues executing its original program rather wasting its time by checking the status of I/O device. In this method when the device is ready for data transfer it sends an interrupt request signal to the microprocessor. The microprocessor then sends back an interrupt acknowledge signal to the I/O device indicating that it has received the request and

suspends its job after completing its current instruction It saves the current address and status and executes the Interrupt Service Routine.

3. DIRECT MEMORY ACCESS (DMA)

Either in programmed I/O or Interrupt I/O transfer the data between the I/O device and external memory is via the accumulator For voluminous data transfer these methods are time consuming and quite uneconomical even though the I/O devices speed matches with the speed of the microprocessor. In such situation it is advisable to provide facilities for transfer of data directly between the I/O device and the external memory without going through the accumulator. In other words, the I/O devices are allowed to transfer data directly to the extemal memory bypassing microprocessor. This is achieved by sending a request to the microprocessor by the I/O device for the Direct Memory Access (DMA). On receipt of each request, microprocessor relinquishes the Address and Data buses and informs the requesting I/O device by sending DMA acknowledgement signal as indicated below. The I/O device withdraws DMA request as soon as the data transfer between the I/O and the external memory is over indicating that the task is complete.

In this system there is no one to indicate the address of the memory locations where the data can be loaded. Hence, a better DMA Transfer Scheme is shown in Fig. below wherein the help of a DMA controller is used to generate the addresses for the data to be transferred from the I/O devices.



DMA Controller Data Transfer

Logical Interfacing Devices:

1. Tristate Inverter
2. Tri state Buffers

3. Encoder
4. Decoder
5. Latches

1. TRI-STATE DEVICES

Tri-state logic devices have three states:

1) Logic 1 or Low

2) Logic 0 or High

3) High impedance

A tri-state logic device has an extra input line called Enable. When this line is active (Enabled), a tri-state device functions in the same way as ordinary logic devices. When this line is not active(disabled), the logic device goes into a high impedance state as if it is disconnected from the system and practically no current is drawn from the system.

Tristate Inverter

There are two different types of Tri-state inverter, one whose output is controlled by an "Active-HIGH" Enable signal and the other which is controlled by an "Active-LOW" Enable signal, as shown below

Active "HIGH" Tri-state Inverter Fig 1(a)

An Active-high Tri-state inverter is activated when a logic level "1" is applied to its "enable" control line and the data is inverted from its input to its output. When the enable control line is at logic level "0", the inverter output is disabled and a high impedance condition, Hi-Z is present on the output.

Active "LOW" Tri-state inverter Fig 1(b)

An Active-low Tri-state inverter is the opposite to the above, and is activated when a logic level "0" is applied to its "enable" control line. The data is inveted from its input to its output. When the enable control line is at logic level "1", the inverter output is disabled and a high impedance condition, Hi-Z is present on the output.

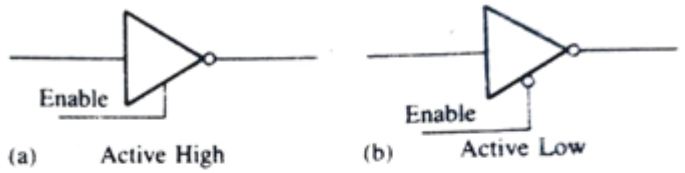


Fig 1

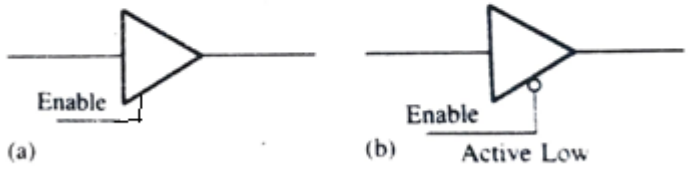


Fig 2

BUFFER

A Digital Buffer is a single input device that does not invert or perform any type of logical operation on its input signal. In other words, the logic level of the output is same as that of the input. The buffer is a logic circuit that amplifies the current or power. The buffer is used primarily to increase the driving capability of a logic circuit. It is also known as driver.

2. Tri-state Buffer

A Tri-state Buffer can be thought of as an input controlled switch which has an output that can be electronically turned "ON" or "OFF" by means of an external "Enable" signal input. This Enable signal can be either a logic "0" or a logic "1" type signal. When Enable line is low (logic „0"), the circuit functions as a buffer. When Enable line is high (logic „1"), its output produces an open circuit condition that is neither "High" nor "low", but instead gives an output state of very high impedance, high-Z, or more commonly Hi-Z. Then this type of device has two logic state inputs, "0" or a "1" but can produce three different output states, "0", "1" or "Hi-Z" which is why it is called a "3-state" device.

There are two different types of Tri-state Buffer, one whose output is controlled by an "Active-HIGH" Enable signal and the other which is controlled by an "Active-LOW" Enable signal, as shown below

Active "HIGH" Tri-state Buffer Fig 2(a)

An Active-high Tri-state Buffer is activated when a logic level "1" is applied to its "enable" control line and the data passes through from its input to its output. When the enable control line is at logic level "0", the buffer output is disabled and a high impedance condition, Hi-Z is present on the output.

Active "LOW" Tri-state Buffer Fig 2(b)

An Active-low Tri-state Buffer is the opposite to the above, and is activated when a logic level "0" is applied to its "enable" control line. The data passes through from its input to its

output. When the enable control line is at logic level "1", the buffer output is disabled and a high impedance condition, Hi-Z is present on the output.

Examples of tri-state buffers

Commercial available 74LS244 is an examples of tri-state buffers

It is also known as a line driver or line receiver. This device is commonly used as driver for the address bus in a bus-oriented system

Figure 3 shows two groups of four buffers with noninverted tri-state output buffers are controlled by two active low Enable lines (1G and 2G). Until these lines enabled, the output of the drivers remains in the high impedance state.

BIDIRECTIONAL BUFFER

The data bus of a microcomputer system is bidirectional; therefore, it requires a buffer that allows data to flow in both directions. Figure 4 shows the logic diagram bidirectional buffer 74LS245, also called an octal bus transceiver. This is commonly used as a driver for the data bus

The 741S245 includes 16 bus drivers, eight for each direction, with tri-state output. The direction of data flow is controlled by the pin DIR. When DIR is high, data flows from A bus to the B bus; when it is low, data flows from B to A. The schematic also has an Enable signal (G), which is active low. The Enable signal and the DIR ANDed to activate the bus lines.

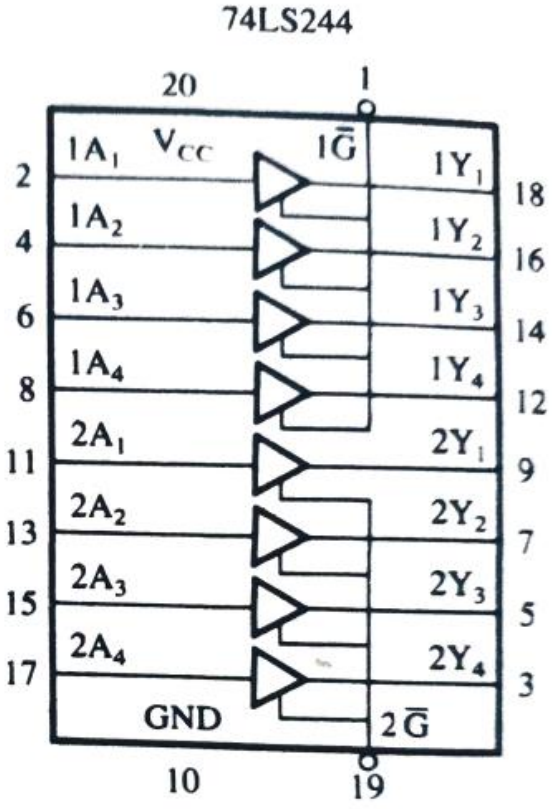
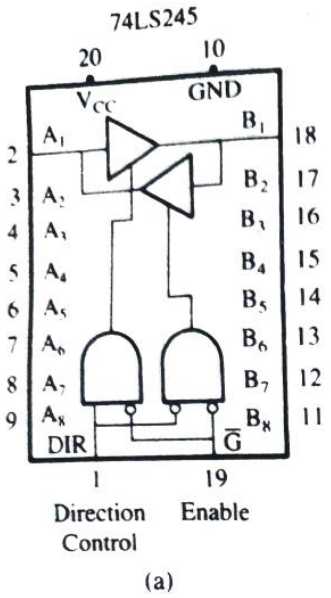


Fig 3



Function Table

Enable $\overline{G}$	Direction Control DIR	Operation
L	L	B Data to A Bus
L	H	A Data to B Bus
H	X	Isolation

H = high level, L = low level, X = irrelevant

(b)

Fig 4

3.ENCODER

The encoder is a logic circuit that provides the appropriate code (binary, BCD, etc.) as output for each input signal.

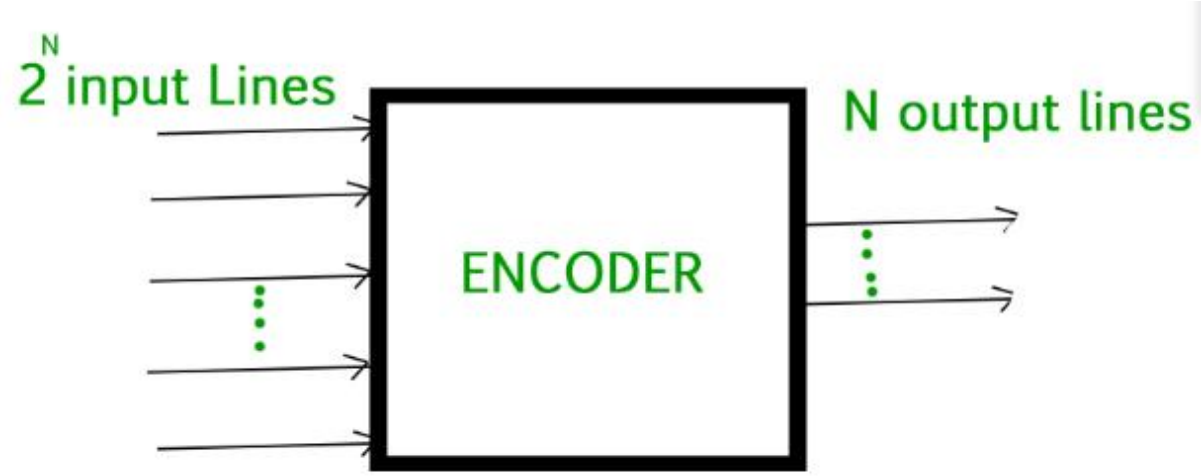


Fig 5

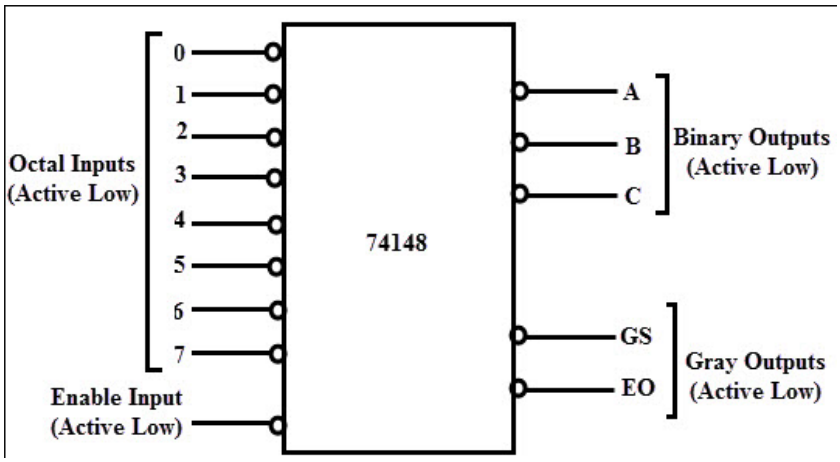
.1 Binary Encoder

A binary encoder, is a multi-input combinational logic circuit that converts the logic level "1" data at its inputs into an equivalent binary code at its output. Generally, digital encoders

produce outputs of 2-bit, 3-bit or 4-bit codes depending upon the number of data input lines. An "n-bit" binary encoder has 2n input lines and n-bit output lines with common types that include 4-to-2, 8-to-3 and 16-to-4 line configurations. The output lines of a digital encoder generate the binary equivalent of the input line whose value is equal to "1" and are available

to encode either a decimal or hexadecimal input pattern to typically a binary or B.C.D. output code.

The Priority Encoder solves the problems mentioned above by allocating a priority level to each input. The priority encoders output corresponds to the currently active input which has the highest priority. So when an input with a higher priority is present, all other inputs with a lower priority will be ignored. The priority encoder comes in many different forms with an example of an 8-input priority encoder along with its truth table shown below.



ENCODER TABLE

Inputs									Outputs				
EI	7	6	5	4	3	2	1	0	A3	A2	A1	GS	EO
1	x	x	x	x	x	x	x	x	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	x	x	x	x	x	x	x	0	0	0	0	1
0	1	0	x	x	x	x	x	x	0	0	1	0	1
0	1	1	0	x	x	x	x	x	0	1	0	0	1
0	1	1	1	0	x	x	x	x	0	1	1	0	1
0	1	1	1	1	0	x	x	x	1	0	0	0	1
0	1	1	1	1	1	0	x	x	1	0	1	0	1
0	1	1	1	1	1	1	0	x	1	1	0	0	1
0	1	1	1	1	1	1	1	0	1	1	1	0	1

Fig 6

8-to-3 Bit Priority Encoder

Priority encoders are available in standard IC form and the TTL 74LS148 is an 8-to-3 bit priority encoder which has eight active LOW (logic "0") inputs and provides a 3-bit code of the highest ranked input at its output. Priority encoders output the highest order input first for example, if input lines "D2", "D3" and "D5" are applied simultaneously the output code would be for input "D5" ("101") as this has the highest order out of the 3 inputs. Once input "D5" had been removed the next highest output code would be for input "D3" ("011"), and so on.

4. DECODER

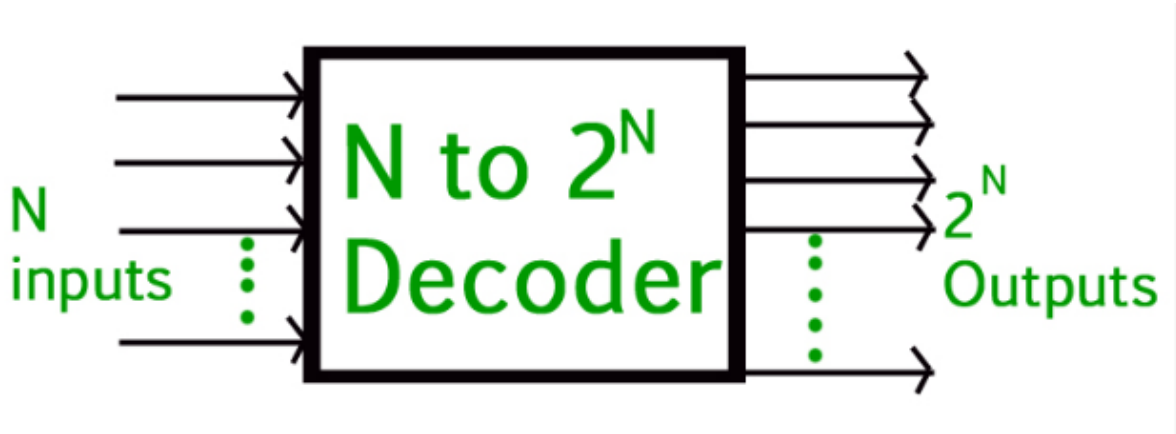
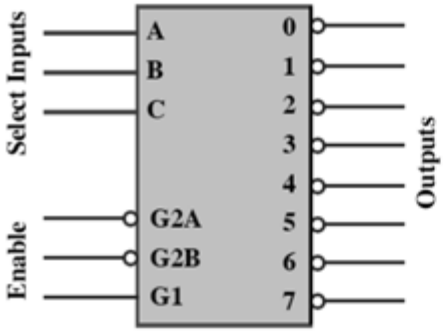


Fig 7

A Decoder is the exact opposite to that of an "Encoder". It is basically, a combinational type logic circuit that converts the binary code data at its input into an equivalent decimal code at its output.

Binary Decoders have inputs of 2-bit, 3-bit or 4-bit codes depending upon the number of data input lines, and a n-bit decoder has 2^n output lines. Therefore, if it receives n inputs (usually grouped as a binary or Boolean number) it activates one and only one of its 2^n outputs based on that input with all other outputs deactivated. A decoders output code normally has more bits than its input code and practical binary decoder circuits include, 2-to-4, 3-to-8 and 4-to-16 line configurations.

The 74LS138 is a 3-line-to-8-line decoder with the enable function. Its logic symbol and truth table are shown in Figure



Inputs						Output							
Enable			Select										
G2A	G2B	G1	C	B	A	0	1	2	3	4	5	6	7
1	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	0	X	X	X	1	1	1	1	1	1	1	1
0	0	1	0	0	0	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	0

Fig 8

A binary decoder converts coded inputs into coded outputs, where the input and output codes are different and decoders are available to "decode" either a Binary or BCD (8421 code) input pattern to typically a Decimal output code.

5. LATCHES

The D Latch

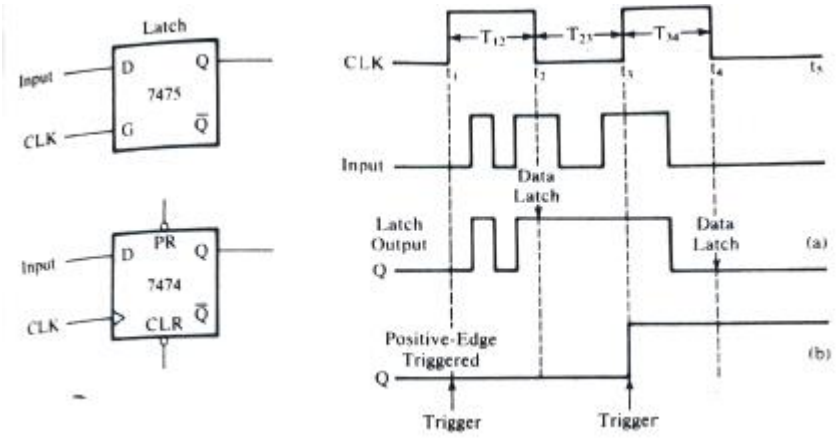
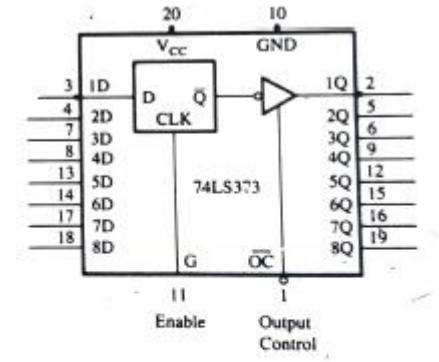


Fig 9

D Flip-Flops: Latch and Clocked

In its simplest form, a latch is a D flip-flop, as shown in Figure 9 it is also called a transparent latch A typical example of a laches is the 7475 D flip flop. In this latch, when the enable signal (G) is high, the output changes according to the input D. Figure On the other hand, in a positive-edge-triggered flip-flop, the output changes with the positive edge of the clock Figure shows the output of the 7474 positive edge-triggered flip-flop At the first positive going clock (t), the input is low therefore, the output remains low until the next positive edge (1). There is no effect on the output at any other time A latch is used commonly to interface output devices.



Function Table

Output Control	Enable G	D	Output
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

Fig 10

The D flip-flop will store and output whatever logic level is applied to its data terminal so long as the clock input is HIGH. Once the clock input goes LOW the "set" and "reset" inputs of the flip-flop are both held at logic level "1" so it will not change state and store whatever data was present on its output before the clock.