Since 1951

JAMAL MOHAMED COLLEGE (AUTONOMOUS)

Accredited (3rd Cycle) with 'A' Grade by NAAC

Affiliated to Bharathidasan University

TIRUCHIRAPPALLI – 620 020

DEPARTMENT OF COMPUTER SCIENCE

MICROPROCESSOR FUNDAMENTALS

20UCS6CC15

Prepared By: Dr. T. Abdul Razak

Semester	Code	Course	Title of the Course	Hours	Credits	Max. Marks	Internal Marks	External Marks
VI	20UCS6CC15	Core – XV	MICROPROCESSOR FUNDAMENTALS	5	5	100	25	75

Course Outcomes (COs):

On completion of the course, students will be able to

CO1. Understand the basics of microprocessors.
CO2. Understand the architecture of a microprocessor and its internal operation.
CO3. Classify the various instructions and study their usage.
CO4. Demonstrate programming proficiency by developing simple assembly language programs.
CO5. Identify the different ways of interfacing memory and I/O with a microprocessor.

UNIT I **15 hours**

Word Length of a Microprocessor – Evolution of Microprocessors – Single Chip Microcontrollers – Embedded Microprocessors – Hardware, Software and Firmware – Central Processing Unit – Memory – Buses – Processing Speed of a Computer – Classification of Computers – Von Neumann Architecture – Harvard Architecture – Data Flow Architecture – Types of Microprocessors – Microprocessor Applications.

UNIT II **15 hours**

Intel 8085 Microprocessor Architecture – Register – Status Flags – Pin Configuration – Opcode and Operands – Instruction Formats – Instruction Cycle – Fetch Operation – Execute Operation – Addressing Modes.

UNIT III **15 hours**

Instruction Set of 8085 – Data Transfer Instructions – Arithmetic Instructions – Logical Instructions – Shift and Rotate Instructions – Branch Instructions – Jump, Call and Return – Stack Instructions – I/O, Machine Control and other Instructions – Assembly Language – Assemblers – Stacks – Subroutines – Macros.

UNIT IV **15 hours**

Assembly Language Programs – Addition, Subtraction, Multiplication and Division of 8-bit numbers – Decimal Addition and Subtraction – Multibyte Addition and Subtraction – 1's and 2's Complements – Assembly and Disassembly of a Byte – Sum of a Series – Block Data Transfer – Finding the Smallest and the Biggest Number in an Array – Arranging a Series of Numbers in Descending and Ascending Order.

UNIT V **15 hours**

Peripheral Devices and Interfacing – Address Space Partitioning – Memory and I/O Interfacing – Data Transfer Schemes – Interrupts of Intel 8085 – Interfacing Devices and I/O Devices – I/O Ports – Programmable Peripheral Interface – Delay Subroutines – Seven-Segment Displays – Types of Seven-Segment Displays – Interfacing Seven-segment Displays.

Text Book:

Badri Ram, *Fundamentals of Microprocessors and Microcomputers*, DhanpatRai Publications, Sixth Revised and Enlarged Edition, 2010.

MICROPROCESSOR FUNDAMENTALS

<u>UNIT-1</u>

INTRODUCTION

A microprocessor is also known as a CPU (Central Processing Unit) in which numbers of peripherals' are fabricated on a single chip. It has ALU (Arithmetic and Logic Unit), a control unit, registers, bus systems, and a clock to perform computational tasks. A digital computer is one which has a microprocessor as its CPU.

Word Length of a Microprocessor

A digital computer can understand information composed of only 0 and 1. Hence, it uses only binary digits 0 and 1 for its internal processing. A binary digit 0 or 1 is called a bit. A group of 8 bits is called a **byte**. The number of bits that a digital computer can process in parallel at a time is called its **word length**.

Word Length of a Microprocessor is given as *n*-bit; where *n* may be 2, 16, 32 or 64. A binary digit 0 or 1 is called a bit. An 8 bit microprocessor can process 8-bit data at a time. If data consists of more than 8 bits, the processor takes up one by one for processing. Its ALU is designed to process 8-bit data. Its general purpose registers which hold data for processing are 8-bit registers. Similarly, a 16-bit processor handles 16-bit data at a time, its ALU processes 16-bit data and general purpose registers hold 16-bit data. Similarly, 32-bit and 64-bit processors process 32-bit and 64-bit data at a time respectively. A processor of longer word length is more powerful and can process data at faster speed as compared with to a processor of shorter word length.

Evolution of Microprocessors

In 1971, Intel invented the first microprocessor 4004. It was a 4-bit microprocessor using PMOS technology. Another 4-bit microprocessor, Intel 4040, was also developed during the same year. Other 4-bit microprocessors during this time were Rockwell International's PPS-4 and Toshiba's T3472. These microprocessor-based systems were used in industrial control applications, calculators, instrumentation and commercial appliances.

In 1972, Intel introduced the first 8-bit microprocessor, Intel 8008, which also uses PMOS technology. In 1973, Intel introduced a more powerful and faster 8-bit NMOS microprocessor, Intel 8080. The microprocessors using NMOS technology were faster and compatible with TTL logic. The drawback of the Intel 8080 was that it needed three power supplies. In 1975, Intel developed an improved and universally accepted NMOS 8-bit microprocessor, Intel 8085. 8-bit microprocessors of other manufactures were Zilog's Z80 and Z800, National Semiconductor's NSC800, and Motorola's MC6800.

In the year 1976, Intel introduced a 16-bit microprocessor, Intel 8086. Other 16-bit microprocessors were: Intel 8088, Intel 80186/80286, Zilog's Z8000, Motorola's 68000 and 68010. The 16-bit microprocessors use VLSI technology. In 1980s, personal computers used 16-bit microprocessors.

In 1985, Intel developed a powerful 32-bit microprocessor, Intel 80386, which became very popular and was widely used in desktop computers. Other 32-bit Intel's microprocessors are: the

486, Pentium, Pentium Pro, Pentium MMX, Pentium II, Pentium II Xeon, Celeron, Pentium III and Pentium 4. 32-bit microprocessors of other manufacturers were Motorola's 68020, 68030 and 68040, Zilog's Z80000. Motorola, IBM and Apple have jointly developed 32-bit RISC processors: PowerPC 601, 603 and 604. Intel also developed a 32-bit RISC processor, Intel 80960, for embedded control applications.

A number of 64-bit microprocessors have also been developed. Examples are: Intel Itanium, SUN's SPARC, AMD's Athlon, PowerPC620, etc.

Presently multicore processors have been developed. Two, four or more CPUs are placed on a single-chip IC. Examples of multicore processors having 2 CPUs on a single chip are: Intel Core2 Duo 8500, E6850, T2700. Multicore processors having 4 CPUs on a single chip are: Intel Core 2 Quad Q9550, Intel Core 2 Extreme QX9770, etc.

Single Chip Microcontrollers

With the development of VLSI and ULSI technologies it became possible to build a digital computer on a single IC. A digital computer built on a single IC is called single-chip microcomputer. Such computers are used in instrumentation, automatic industrial control, consumer and home appliances, etc. It is very small and compact. As it is used for control applications it is also called a microcontroller or embedded microcontroller. It contains a CPU, ROM or EPROM, RAM, I/O interfaces, A/D converters and DMA channels.

In the year 1976, Intel introduced the 8048 series of single-chip 8-bit microcomputers. It is also known as MCS-48. Its family members are 8048, 8748, 8041, 8042, etc. In the year 1980, Intel developed a more powerful series of 8-bit microcontrollers, Intel 8051. The Intel 8051 series of microcontrollers are faster and have an enhanced instruction set. Other microcontrollers include: Texas Instrument's TMS1000, Motorola's 6801 and Zilog's Z8. In the year 1983, Intel introduced 16-bit microcontrollers, Intel 8096 series. These microcontrollers are more powerful compared to 8-bit microcontrollers. They are used for sophisticated real-time applications such as industrial control, instrumentation and intelligent computer peripherals. Some examples are: control of large horse power motors, robots, guidance and control of missiles, etc.

Motorola has introduced 32-bit RISC microcontroller MPC503. It is used in advanced communication devices, speech processing system, automotive modules, etc. IBM has developed a 32-bit RISC embedded microcontroller, 403GA. It is used in office automation equipment, consumer electronics and video games, telecommunications and networking. 68300 is another microcontroller from Motorola.

Embedded Microprocessors

Embedded microprocessors have been developed for data control applications. A microprocessor which forms the part of the system to be controlled, is called an embedded microprocessor. It processes large amount of data. For example, a laser printer consists of an embedded microprocessor. The computer sends data to the laser printer. The embedded processor which resides in the laser printer manages operations required for printing and handles the data which are to be printed. The data to be handled becomes large as a number of dots are processed for each character of the data.

For each application, the memory requirement varies and hence memory is not incorporated on the embedded processor chip. Once programmed it manages the control functions for which it has been designed and programmed. Data control applications require multifunction control such as DMA control, data processing, data formatting, I/O control, etc. Embedded processors are quite powerful to meet such requirements.

Embedded control applications have two distinct areas of control namely, event control (real-time control) and data control. Microcontrollers are used for event control, and embedded microprocessors are used for data control. Event control requires small amount of data to be processed. Memory requirement for data control differs from application to application, and hence it is not possible to place the memory on the processor chip. Example of event control applications are: temperature controller, speed control of a motor, engine control, etc.

Some of the examples of such processors are: Embedded versions of Xeon and multicore Xeon processors, Embedded versions of Pentium processors family, Embedded versions of Celeron processors, Intel's i960 processor family and Motorola 68060.

Hardware, Software and Firmware

The physical devices of a computer are called **hardware**. A physical device may be electronic, electrical, magnetic, mechanical or an optical device. Examples are: microprocessors and other ICs, hard disk, floppy disk, printer, CD-ROM, keyboard, etc. A sequence of instructions given to a computer to perform certain task is called a program. A set of programs written for a computer is known as **software**. The term software includes both system software and user's programs. The system software includes operating system, assembler, compiler, interpreter, text editor, debugger, etc. The operating system is a collection of programs which controls the overall operation of a computer. The operating system allows users to communicate with a computer. It allows users to create files, use and control I/O devices, manage memory space, schedule jobs, execute programs, store an print results, etc. Examples of operating systems are: MS-DOS, UNIX, versions of WINDOWS, Linux, etc.

The programs stored in ROMs, PROMs, EPROMs, EEPROMs or flash memories are known as **firmware**. A large number of prewritten programs are available to solve specific type of problems. Such programs are called application programs or application packages. Some application programs are of general nature such as MS-Office, FOXPRO, ORACLE, etc. Some application programs have been developed for specific purposes such as for designing building, power systems, inventory control, payroll processing, accounting, etc.

Central Processing Unit

The Central Processing Unit (CPU) is the brain of a computer. It executes user's programs and controls memory and I/O devices. User's programs reside in the memory. The CPU fetches instructions of a program from the memory sequentially. It fetches one instruction at a time, decodes it and then executes it. After decoding an instruction the CPU comes to know what operations are to be performed. If the data are in the general purpose registers, the CPU executes the program. If the data are in the memory or at the I/O ports, it reads the data from the memory or I/O port. Then it executes the instruction and the stores the intermediate results. After completing the execution of one instruction, the CPU fetches the next instruction of the program. In this way the CPU fetches and executes all the instructions of a program and gives the final result.

The CPU controls memory and I/O devices to receive, store and send data of the program under execution. Under its control, data and results are displayed on the monitor, stored in the memory or printed by the printer.

The CPU of a microcomputer is a microprocessor. The CPU of a large computer such as server or supercomputer has more than one microprocessor which operates in parallel. The major components of a CPU are: ALU, timing and control unit, and registers.

ALU (Arithmetic and Logic Unit): An ALU performs arithmetic and logic operations. Arithmetic operations include – addition, subtraction, multiplication and division. Logic operations include – finding complement of a number, comparing two numbers, rotating the bits of a number left or right, incrementing or decrementing a number.

Timing and Control Unit: This unit controls all the operations of the CPU. It controls input, output and all other devices connected to the CPU. It generates the control signals which are required for all the operations to be performed by the CPU. Such signals are also called timing signals.

Registers: Registers include accumulator, general purpose registers and special purpose registers. The accumulator is a register which contains one of the operands of an instruction to be executed. It also stores the result after the execution of an arithmetic and logical instruction. General purpose registers are used to store data and hold intermediate results while executing a program. Special purpose registers are used by the microprocessor itself. They are not accessible to the users. Examples of special purpose registers are: program counter, stack pointer, instruction register, and flag register.

Memory

A memory is just like a human brain. It is used to store programs, data and results. A computer uses a number of memory devices of different technologies such as semiconductor memory, magnetic memory and optical memory. Computer memory is the storage space in the computer, where data is to be processed and instructions required for processing are stored.

Memory is primarily of three types:

- Cache Memory
- Semiconductor Memory (Primary Memory/Main Memory)
- Secondary / Auxiliary Memory

Cache Memory:

Cache memory is a very high speed semiconductor memory which can speed up the CPU. It acts as a buffer between the CPU and the main memory. It is used to hold those parts of data and program which are most frequently used by the CPU. The parts of data and programs are transferred from the disk to cache memory by the operating system, from where the CPU can access them.

There are two types of cache schemes: *Write-through* and *Write-back*. In a write-through cache the main memory is updated each time the CPU writes into cache. The advantage of this scheme is that the main memory always contains the same data as the cache contains. In the write-back scheme only the cache memory is updated during a write operation. The words are removed from the cache time to time to make room for the new block of words.

Semiconductor Memory:

There are two main types of semiconductor memory: RAM (Random Access Memory) and ROM (Read Only Memory).

RAM (Random Access Memory):

RAM is the internal memory of the CPU for storing data, program, and program result. It is a read/write memory which stores data until the machine is working. As soon as the machine is switched off, data is erased.

Data in the RAM can be accessed randomly. RAM is volatile, i.e. data stored in it is lost when we switch off the computer or if there is a power failure. Hence, a backup Uninterruptible Power System (UPS) is often used with computers. RAM is small, both in terms of its physical size and in the amount of data it can hold. RAM is of two types: Static RAM (SRAM) and Dynamic RAM (DRAM)

Static RAM (SRAM): The word **static** indicates that the memory retains its contents as long as power is being supplied. However, data is lost when the power gets down due to volatile nature. SRAM chips use a matrix of 6-transistors and no capacitors. Transistors do not require power to prevent leakage, so SRAM need not be refreshed on a regular basis. There is extra space in the matrix, hence SRAM uses more chips than DRAM for the same amount of storage space, making the manufacturing costs higher. SRAM is thus used as cache memory and has very fast access.

Dynamic RAM (DRAM): DRAM, unlike SRAM, must be continually **refreshed** in order to maintain the data. This is done by placing the memory on a refresh circuit that rewrites the data several hundred times per second. DRAM is used for most system memory as it is cheap and small. All DRAMs are made up of memory cells, which are composed of one capacitor and one transistor. These devices are less expensive and consume less power.

SDRAM (synchronous DRAM): It is a generic name for various kinds of dynamic random access memory (DRAM) that are synchronized with the clock speed that the microprocessor is optimized for. This tends to increase the number of instructions that the processor can perform in a given time.

SGRAM (Synchronous Graphics RAM) is clock-synchronized random access memory that is used for video memory. It is relatively low-cost video memory and very helpful for many graphics applications.

DDR SDRAM (Double Data Rate Synchronous Dynamic Random Access Memory) is a type of random-access memory module that allows for higher transfer rates and faster performance compared to earlier RAM modules. DDR SDRAM transfers memory on both the rising edge and falling edge of a clock cycle, doubling the transfer rate.

EDO DRAM (Extended Data Out Dynamic Random Access Memory) is a type of DRAM that is faster than conventional DRAM. Unlike conventional DRAM which can only access one block of data at a time, EDO RAM can start fetching the next block of memory at the same time that it sends the previous block to the CPU.

SIMM (Single Inline Memory Module) is a module containing one or several random access memory (RAM) chips on a small circuit board with pins that connect to the computer motherboard.

DIMM (Dual Inline Memory Module) is a small-scale circuit board that holds memory chips on the motherboard. DIMM incorporates a series of memory called dynamic random access memory (DRAM), which provides primary storage, the main memory that continually reads and executes stored instructions or data directly to the CPU.

ROM (Read Only Memory):

A ROM is a memory from which we can only read but cannot write on it. This type of memory is non-volatile. The information is stored permanently in such memories during manufacture. A ROM stores such instructions that are required to start a computer. This operation is referred to as **bootstrap**.

Types of ROM:

PROM (Programmable Read Only Memory): PROM is read-only memory that can be modified only once by a user. The user buys a blank PROM and enters the desired contents using a PROM program. Inside the PROM chip, there are small fuses which are burnt open during programming. It can be programmed only once and is not erasable.

EPROM (Erasable and Programmable Read Only Memory): EPROM can be erased by exposing it to ultra-violet light and it can be reprogrammed more than once. Usually, an EPROM eraser achieves this function. For erasing the contents, ultra-violet light is passed through a quartz crystal window.

EEPROM (Electrically Erasable and Programmable Read Only Memory): EEPROM is programmed and erased electrically. It can be erased and reprogrammed many number of times. Both erasing and programming take about 4 to 10 ms (millisecond). In EEPROM, any location can be selectively erased and programmed. EEPROMs can be erased one byte at a time, rather than erasing the entire chip. Hence, the process of reprogramming is flexible but slow.

Flash memory: Flash memory is a type of electronically erasable programmable read only memory (EEPROM), but may also be used as a standalone memory storage device such as a USB drive. EEPROM is a type of data memory device using an electronic device to erase or write digital data. Flash memory is a distinct type of EEPROM, which is programmed and erased in large blocks. It is often found in USB flash drives, MP3 players, digital cameras and solid-state drives.

Secondary / Auxiliary Memories:

The memory which stores information permanently is called a secondary or auxiliary memory. The secondary memory is a mass storage device. It stores operating systems, compilers, assemblers, interpreters and application programs. Following are the types of secondary memory:

- Magnetic Memory
- Optical Disks
- CCD Memory

Magnetic Memory:

Magnetic memories use a system of storing information through the alignment of small grains in a magnetic material. Once the grains have been aligned by an external magnetic field, the information remains stored for long periods of time. Magnetic memories are non-volatile permanent memories. The following types of magnetic memory are used in computer systems:

- Hard Disk
- Floppy Disk
- Magnetic Tape

Hard Disk:

A hard disk is part of a unit, often called a *disk drive*, *hard drive* or *hard disk drive* that stores and provides relatively quick access to large amounts of data on an electromagnetically charged surface or set of surfaces. Today's computers typically come with a hard disk that can contain anywhere from billions to trillions of bytes of storage.

A hard disk is actually a set of stacked disks, like phonograph records. Each disk has data recorded electromagnetically in concentric circles, or tracks, on the disk. A head, similar to a phonograph arm but in a relatively fixed position, writes or reads the information on the tracks. Two heads, one on each side of a disk, read or write the data as the disk spins. Each read or write operation requires that data be located, an operation called a *seek*. Data already in a disk cache, however, will be located more quickly.

A hard drive consists of several major components inside its casing. These include the platter for storing data, a spindle for spinning platters, a read/write arm for reading and writing data, an actuator to control the action and movement of the read/write arm and a logic board. Hard disks include one or more aluminum, glass or ceramic platters made of substrate material with a thin magnetic surface, or media layer, to store data. Platters store and organize data in specific structures (tracks, sectors and clusters) on this media layer, which is only a few millionths of an inch thick. A super thin protective and lubricating protective layer above the magnetic media guards against accidental damage and contamination by foreign material, like dust.

External Hard Disk: An external hard drive is a storage device located outside of a computer that is connected through a USB cable or wireless connection. An external hard drive is usually used to store media that a user needs to be portable, for backups, and when the internal drive of the computer is already at its full memory capacity. These devices have a high storage capacity compared to flash drives and are mostly used for backing up numerous computer files or serving as a network drive to store shared content. External hard drives are also known as removable hard drives.

Hybrid Hard Disk (HDD): It is combination of conventional magnetic hard disk with flash memory-based cache. The use of flash memory improves the performance of disk operations.

Hard Disk Controller: A Hard Disk Controller (HDC) is an electrical component within a computer hard disk that enables the processor or CPU to access, read, write, delete and modify data to and from the hard disk. Essentially, an HDC allows the computer or its processor to control the hard disk. There are two types of hard disk controller – IDE (Integrated Drive Electronics) and SCSI (Small Computer System Interface). IDE is a standard interface for connecting hard drives into your computer’s motherboard. You can attach up to 2 hard drives on a single IDE connectors giving you up to a maximum of 4 drives attached to the system. SCSI is not necessarily an interface for hard drives alone. It was intended as a universal interface for many devices; the devices that SCSI supported included hard drives, scanners, plotters, disc drives, and many more.

RAID (Redundant Arrays of Independent Disks): The basic idea behind RAID is to combine multiple small, inexpensive disk drives into an array to accomplish performance or redundancy goals not attainable with one large and expensive drive. This array of drives appears to the computer as a single logical storage unit or drive. This can speed up hard drive access time significantly. RAID allows information to access several disks.

SATA (Serial ATA) Interface: Serial ATA (SATA or Serial Advanced Technology Attachment) is a standard that has been defined to connect hard disk or optical drives to a computer. Its transfer rate is 150 MBPS. SATA is more flexible and intelligent interface.

Floppy Disks:

A floppy disk, often called a *diskette*, is a thin, round, flat piece of Mylar. It has an extremely thin coating of ferric oxide or magnetic oxide layer that is capable of storing magnetic fields, like thick recording tape. The *read-write head* in a *floppy disk drive* stores data on the disk by altering the magnetic particles. Floppy disks retrieve information far more slowly than HDD because they rotate at lower speed. *Data* is written on to the floppy disk by the disk drive's read/write heads as the disk rotates inside the jacket. Because floppy disks store information magnetically, any magnet can destroy the data (information) on the disk.

IBM Computer used 5.25" inch floppy disks that can contain 360 KB, while modern 3.5" inch disks hold 1.44 Mb. Floppy disks became the main medium for software distribution during the formative years of personal computing.

In Floppy Disks, information is organized in *tracks.* Each track is subdivided into *sectors* and each sector into bytes. The smaller 3.5" Floppy Disks use 80 tracks per side, 18 sectors per track, and 512 bytes per sector, yielding a capacity of 1.44 MB

The advantage of the floppy disk is that it is removable, and so can be used to distribute software*,* to transfer data from one computer to another, or to *back up files* from a *hard disk.* But compared to a hard disk, floppy disks are also slower, offer relatively small amounts of storage, and can be easily damaged.

Magnetic Tapes:

Magnetic tapes are serial access type mass storage devices suitable for backup storage. A magnetic tape is a low-cost device and it has large storage capacity. When a large volume of data is to be processed sequentially, it can be stored on a magnetic tape. A file or a particular information stored on a magnetic tape cannot be accessed directly on a random basis as it is done in case of a hard disk.

The magnetic tape is made of mylar plastic coated with magnetic material (iron oxide) only on only one side of the tape. The sizes of magnetic tapes are ½ inch, ¼ inch, 8 mm and 3 mm. Modern magnetic tapes for hard disk backup come in cassette. These units are called cartridge tapes. The tapes are 8 mm or 3 mm wide. Their storing capacities are a few GB to 400 GB. Reading and writing is performed by a helical scan system operating across the tape, similar to that used in video cassette tape drives. The Digital Audio Tapes (DAT) are the latest addition to the magnetic tape family. It uses a SCSI interface. The tape length is 60 or 90 meters, capacity up to 4 GB and the data transfer rate is 366 KB/sec. The data on the tape are organized in the form of records separated by gaps. A gap between the two consecutive records is called inter-block gap (IBG).

Optical Disks:

An optical disk is any computer disk that uses optical storage techniques and technology to read and write data. It is a computer storage disk that stores data digitally and uses laser beams to read and write data. Optical drive reads data by focusing a laser beam on the surface of the disk. A laser detects the presence of a pit. The presence of pit indicates 1 and absence of pit indicates 0. Laser beam converts these pits into digital data. The amount of space required to record an optical bit is much less than a magnetic bit. Optical disk storage capacity is from 600 MB to over 1 GB. The optical storage is much faster than magnetic media.

Types of optical disk:

CD-ROM: CD-ROM stands for Compact Disk Read-Only Memory. The data stored on CD-ROM can only be read. It cannot be deleted or changed. CD-ROM is a portable storage device. The data can be transferred easily by using CD-ROM. It can store about 650MB of data. It is the least expensive way to store large amounts of data and information. D-ROM disks are durable and easy to handle. Information can be stored on CD-ROM for many years. CD-ROM is mostly used to store a large amount of information like sound, color, graphics, and videos.

CD-R or WORM (Write Once Read Many): The user can write data on WORM only once. The written data can be read as many times as desired. It is suitable for data and files which do not change. They store permanent information for records. A laser beam is used to write information on the disk. The storage capacity of the disk is 700 MB. Its advantage is high capacity, better reliability and longer life.

CD-RW (CD-ReWritable): CD–Read Writable (CD-RW) refers to an optical CD that may be written and rewritten multiple times. CD-RW allows for data erasing during each rewritable session. However, data cannot be changed during CD-RW sessions. Some CD-RW discs have a multisession feature, in which additional data may be written at a later time if extra space is available. A CD-RW can hold data for several years if the disc is protected from direct sunlight. Most CD-RW discs hold approximately 74 minutes and 640 MB of data, but some hold 80 minutes and 700 MB of data.

Digital Versatile Disc (DVD): is an optical disc storage medium similar to a compact disc, but with enhanced data storage capacities as well as with higher quality of video and audio formats. The DVD is widely used for video formats, audio formats as well software and computer files. Digital versatile discs are also known as digital video discs. A DVD can be termed in different ways according to their applications:

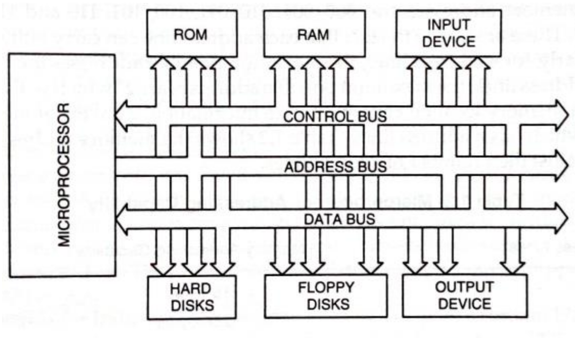
- If they are used for storing data that can only be used to read and cannot be written, it is called a DVD-ROM.
- If a DVD is used to record data and then work as a DVD-ROM, it is termed as DVD-R, where 'R' stands for 'recordable'. In a DVD-R, the data is can be inputted only once.
- If a DVD can be read and again be written over it, it will be called a DVD-RW, where 'RW' stands for 'Re-Writable'.

CCD Memory:

A charge-coupled device (CCD) is a light-sensitive integrated circuit that stores and displays the data for an image in such a way that each pixel (picture element) in the image is converted into an electical charge the intensity of which is related to a color in the color spectrum. CCDs are now commonly included in digital still and video cameras. They are also used in astronomical telescopes, scanners, and bar code readers. The devices have also found use in machine vision for robots, in optical character recognition (OCR), in the processing of satellite photographs, and in the enhancement of radar images, especially in meteorology.

Buses

Various I/O devices and memory devices are connected to a CPU by groups of lines called buses. There are three types of buses: Address Bus, Data Bus and Control Bus. A schematic connection of I/O and memory devices in a microprocessor through the address, data and control buses is shown in the figure below:



Schematic connection of memory and I/O devices to microprocessor

Address Bus: The address bus carries the address of a memory location or an I/O device that the CPU wants to access. When an address is sent by the CPU, all devices connected to the CPU through the address bus receive this address but only the selected device will respond. The address bus is a unidirectional bus.

Data Bus: The data bus is used to transfer data between the processor, memory and I/O devices. The data bus in bidirectional.

Control Bus: The control bus is used to carry necessary control signals between the CPU and memory or I/O devices.

Memory addressing capacity of a CPU: The memory addressing capacity of a CPU depends on the width of the address bus. If a CPU has n address lines, it can directly address 2^n memory locations. Intel 8085 microprocessor has 16 address lines. Hence, the memory addressing capacity of 8085 is 2^{16} = 64 K memory locations.

Bus Architecture:

The important types of bus architecture used in computer systems are: PCI bus, ISA bus, Universal Serial Bus (USB) and Accelerated Graphics Port (AGP).

PCI Bus: PCI stands for Peripheral Component Interconnect. PCI is a hardware bus used for adding internal components to a desktop computer. For example, a PCI card can be inserted into a PCI slot on a motherboard, providing additional I/O ports on the back of a computer. A PCI bus lets you change different peripherals that are attached to the computer system, so it allows the use of different sound cards and hard drives. Usually, there are three or four PCI slots on a motherboard. With PCI, you can unplug the component you want to swap and plug in the new one in the PCI slot. Or, if you have an open slot, you can add another peripheral like a second hard drive to dual boot your computer or a special sound card if you deal with music a lot.

ISA Bus: An Industry Standard Architecture bus is a computer bus that allows additional expansion cards to be connected to a computer's motherboard. It is a standard bus architecture for IBM compatibles. ISA was originally an 8-bit computer bus that was later expanded to a 16-bit bus.

Universal Serial Bus (USB): is a common interface that enables communication between devices and a host controller such as a personal computer. It connects peripheral devices such as digital cameras, mice, keyboards, printers, scanners, media devices, external hard drives and flash drives. Because of its wide variety of uses, including support for electrical power, the USB has replaced a wide range of interfaces like the parallel and serial port.

USB 1.x is an external bus standard that supports data transfer rates of 12 Mbps and is capable of supporting up to 127 peripheral devices. **USB 2.0**, also known as **hi-speed USB**, is capable of supporting a transfer rate of up to 480 megabits per second (Mbps), or 60 megabytes per second (MBps). **USB 3.0**, also known as **SuperSpeed USB**, supports transfer rates up to 5.0 gigabits per second (Gbps), or 640 megabytes per second (MBps). **USB 3.1**, also known as **SuperSpeed+**, is the latest version of the USB protocol. Today, many devices use the USB 3.0 and 3.1 revisions for improved performance and speed.

Processing Speed of a Microprocessor

The processing speed of a microprocessor is usually measured in number of instructions executed per second. The number of instructions executed per second is also known as throughput of the microprocessor. It is generally expressed in MIPS (Millions of instructions per second). The floating-point performance of a processor is given in MFLOPS (Millions of Floating-Point Instructions Per Second). MFLOPS rating gives the processor's performance for highly mathematical operations. MIPS rating gives the processor's performance for integer operations.

For online transaction processing applications, computer performance is measured in TPS (Transactions Per Second). In artificial intelligence applications, KLIPS (Kilo Logic Inferences Per Second) is often used to indicate the reasoning power of an AI machine. iCOMP is Intel's Comparative Microprocessor Performance index. It consists of a collection of benchmarks to evaluate an index or relative performance of Intel microprocessors. It is based on integer, floating-point, graphics and video performances.

Classification of Computers

Computers are classified as: Handheld or Palmtop computers, Notebook or Laptop computers, Desktop computers, Workstations, Mainframes, and Supercomputers.

Handheld or Palmtop Computers: They are also called as personal Digital Assistant (PDA), pocket PC or palm PC. These computers are small in size. They can be held in hands or kept in a pocket. They contain a tiny keyboard, micro disk memory and a small LCD screen. They are capable of doing word processing, spreadsheets and hand writing recognition, game playing, faxing and paging. They can be connected to wireless network GSM, GPRS and Bluetooth.

Notebook or Laptop Computers: Laptop computers are lightweight mobile PCs with a thin screen. They are often called notebook computers because of their small size. Laptops can operate on batteries, so you can take them anywhere. Laptops combine all the input/output components and capabilities of a desktop computer, including the display screen, small speakers, a keyboard, data storage device, sometimes a optical disc drive, pointing devices (such as a touchpad or track pad), a processor, and memory into a single unit. Most modern laptops feature integrated webcams and built-in microphones, while many also have touch screens.

Desktop Computers: Desktop computers are designed for use at a desk or table. They are typically larger and more powerful than other types of personal computers. Desktop computers are made up of separate components. The main component, called the system unit, is usually a rectangular case that sits on or underneath a desk. Other components, such as the monitor, mouse, and keyboard, connect to the system unit.

Workstations: The workstation is a computer used for engineering applications (CAD/CAM), desktop publishing, software development, and other such types of applications which require a moderate amount of computing power and relatively high-quality graphics capabilities. Workstations generally come with a large, high-resolution graphics screen, a large amount of RAM, inbuilt network support, and a graphical user interface. Most workstations also have mass storage device such as a disk drive, but a special type of workstation, called diskless workstations, comes without a disk drive. Common operating systems for workstations are UNIX and Windows NT.

Mainframe Computers: Computers with large storage capacities and very high speed of processing are known as mainframe computers. The mainframe is very large in size and is an expensive computer capable of supporting hundreds or even thousands of users simultaneously. Mainframe executes many programs concurrently and supports much simultaneous execution of programs like ATM transactions. They are also used as central host computers in distributed data processing system. Examples: IBM 370, AS/400.

Supercomputers: A supercomputer is a computer with a high level of performance as compared to a general-purpose computer. The performance of a supercomputer is commonly measured in floating-point operations per second (FLOPS). Supercomputers are one of the fastest computers currently available. Supercomputers are very expensive and are employed for specialized applications that require an immense amount of mathematical calculations, such as weather forecasting, scientific simulations, graphical animations, nuclear energy research and electronic design. Examples: IBM Deep Blue, PARAM.

Von Neumann Architecture

Von Neumann architecture was first developed by John von Neumann in 1945. His computer architecture design consists of a Control Unit, Arithmetic and Logic Unit (ALU), Memory Unit, Registers and Inputs/Outputs. Von Neumann architecture is based on the stored-program computer concept, where instruction data and program data are stored in the same memory. This design is still used in most computers produced today.

Harvard Architecture

This architecture is an enhancement of Von Neumann architecture. It contains separate instruction memory and data memory. The instruction and data memories have separate data paths. This increases the processing speed of the processor. The processors having separate instruction and data caches use Harvard architecture.

Data Flow Architecture

A data flow or data-driven processor does not employ a program counter. Instructions are executed depending upon the availability of data. Its fetch/decode unit fetches 20 to 30 instructions in advance. These instructions are decoded and their opcodes are placed in an instruction pool. The execute unit of the processor checks the opcode of an instruction, and examines whether data is available for its execution. If data are available, the instruction is executed. If data are not available for the execution of the instruction, it moves ahead and examines the next instruction in the instruction in the instruction pool. Thus it executes only those instructions for which data are available. Pentium II to Pentium 4 are data flow processors.

Types of Microprocessors

Vector Processors: Vector processor is basically a central processing unit that has the ability to execute the complete vector input in a single instruction. More specifically we can say, it is a complete unit of hardware resources that executes a sequential set of similar data items in the memory using a single instruction. A vector is an array of operands of the same type. A vector processor works in conjunction with a scalar processor. Scalar calculations are done by the scalar processor and the vector computations are performed by vector processor which acts as a co-processor for vector computations.

Array Processors or SIMD Processors: Array processors are also designed for vector computations. The difference between an array processor and a vector processor is that a vector processor uses multiple vector pipelines whereas an array processor employs a number of processing elements to operate in parallel. The array processor also works under the control of a control processor which is a scalar processor. An array processor contains multiple numbers of ALUs. Each ALU is provided with a local memory. The ALU together with the local memory is called a processing element (PE). An array processor is a Single Instruction Multiple Data (SIMD) processor.

Scalar and Superscalar Processors: A processor that executes scalar data is called a scalar processor. A scalar processor processes only one data item at a time, with typical data items being integers or floating-point numbers. A scalar processor is classified as a SISD processor (Single Instruction, Single Data). A superscalar processor, on the other hand, executes more than one instruction during a clock cycle by simultaneously dispatching multiple instructions to redundant functional units on the processor.

RISC: It stands for Reduced Instruction Set Computer. It is a type of microprocessor architecture that uses a small set of instructions of uniform length. These are simple instructions which are generally executed in one clock cycle. RISC chips are relatively simple to design and inexpensive. The setback of this design is that the computer has to repeatedly perform simple operations to execute a larger program having a large number of processing operations. Examples: SPARC, POWER PC etc.

CISC: It stands for Complex Instruction Set Computer. These processors offer the users, hundreds of instructions of variable sizes. CISC architecture includes a complete set of special purpose circuits that carry out these instructions at a very high speed. These instructions interact with memory by using complex addressing modes. CISC processors reduce the program size and hence lesser number of memory cycles is required to execute the programs. This increases the overall speed of execution. Example: Intel architecture.

EPIC: It stands for Explicitly Parallel Instruction Computing. The best features of RISC and CISC processors are combined in the architecture. It implements parallel processing of instructions rather than using fixed length instructions. The working of EPIC processors is supported by using a set of complex instructions that contain both basic instructions as well as the information of execution of parallel instructions. It substantially increases the efficiency of these processors.

Digital Signal Processors (DSP): These are microprocessors specifically designed to process signals. They receive digitized signal information, perform some mathematical operations on the information and give the result to an output device. Most DSPs accept analog signals using ADC, process them and send the result to DAC or to a host processor. They implement integration, differentiation, Fourier transforms, etc. using hardware.

Symbolic Processors: Such processors are designed for expert systems, machine intelligence, knowledge-based system, pattern recognition, etc. This type of processing does not require floating-point operations. Symbolic processors are also called Prolog processors or LISP processors.

Microprocessor Applications

Microprocessor-based systems are used in almost every sphere of life. Their applications are increasing at very fast rate day by day. A microprocessor is used as a CPU of a computer. The availability of low cost, low power and small weight, computing capability makes it useful in different applications.

Besides CPU, it is also used to control input, output and other devices of a computer. Single chip microcomputers also called microcontrollers are widely used for automatic control in industry for process and machine control, in instrumentation, for commercial and consumer appliances control, etc. For example, a microcontroller is used to measure and control the temperature of a furnace and oven, speed of an electric motor, deflection pressure of a boiler, etc. It is used in automatic control of cars to control air and fuel mixture, to test the condition of the engine, brakes, etc. Microcontrollers are used in military equipment, radars, tanks, etc.

Computers are used for word processing, database management, storing information, to connect any institution/organization to other institutions / organizations through internet, for data analysis, computation, scientific and engineering calculations, design of buildings, design of machines, for preparation of drawing, lift control, traffic lights control, automatic control of generator’s voltage, fuel control of furnaces in a power plant, etc.

Computers are used for railway ticket reservation, air ticket reservation, books and journals publishing, telephone exchange, in banks, in robots, video games, sports, hospitals, shops, hotels, schools, colleges, stores, smart cameras, energy meters, photograph processing laboratories, point-of-sale terminals, project management, general administration, etc.

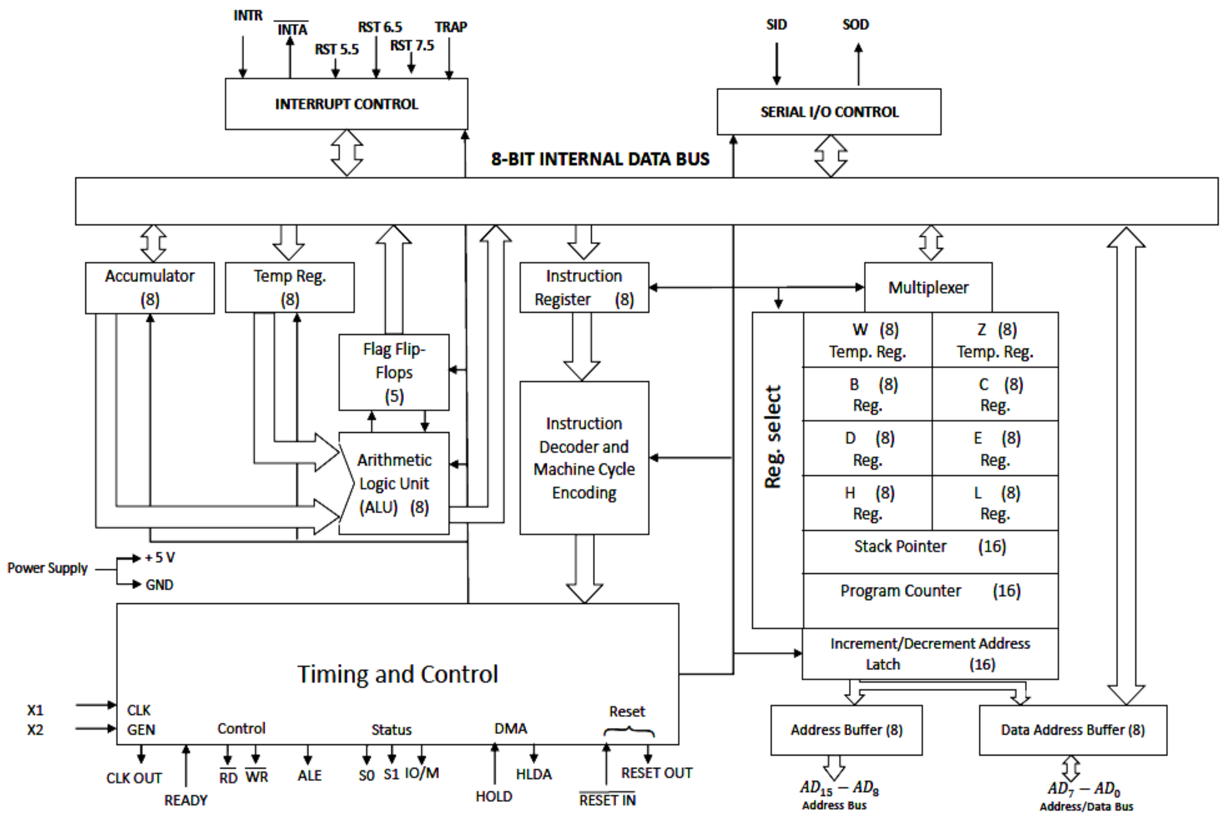
A point-of-sale terminal is connected to a computer for processing sale transactions and to issue cash memo. It uses optical barcode reader to identify items. When a barcoded item is passed through a scanning window at a automatic checking point, the barcode is read by an optical bar reader. The code is then decoded and data are sent to the computer that looks up the price of the items. The stock at hand and sale records are updated. The name of the items, its price and other descriptions are printed on the sale receipt which is issued to the customer. The point-of-sale terminals are used at shops and stores to process sales transactions.

Thus, we see that computers have numerous applications and they are going to have great impact on every aspect of our life in the near future.

UNIT-2

INTEL 8085 MICROPROCESSOR ARCHITECTURE

The microprocessor is the central processing unit (CPU) of a computer. It is the heart of the computer. INTEL 8085 is an 8-bit microprocessor. It is a 40 pin I.C. package fabricated on a single LSI chip. Figure below shows the block diagram of INTEL 8085 microprocessor.



The important units of the microprocessor are described in the subsequent sections.

ALU: The arithmetic and logic unit, ALU, performs the following arithmetic and logical operations.

- Addition
- Subtraction
- Logical AND
- Logical OR
- Logical EXCLUSIVE-OR
- Complement
- Increment
- Decrement
- Shift and Rotate, etc,

Timing and Control Unit: The timing and control unit generates the timing and control signals which are necessary for the execution of instructions. It controls data flow between CPU and peripherals (including memory). It provides status, control and timing signals which are required for the operation of memory and I/O devices. It controls the entire operations of the microprocessor and peripherals connected to it.

Registers: Registers are used by the microprocessor for temporary storage and manipulation of data and instructions. Intel 8085 microprocessor has the following registers:

- One 8-bit accumulator (ACC) i.e. register A
- Six 8-bit general purpose registers B, C, D, E, H and L
- One 16-bit stack pointer, SP
- One 16-bit program counter, PC
- Instruction register
- Temporary register

Accumulator(A): The Accumulator is an 8-bit register associated with the ALU. It is used to hold one of the operands of an arithmetic or logical operation. It serves as one input to the ALU. The other operands for an arithmetic or logical operation may be stored either in the memory or in one of the general-purpose registers. The final result of arithmetic or logical operation is placed in the accumulator.

General Purpose Registers: The 8085 microprocessor contains six 8-bit general-purpose registers. They are: B,C,D,E,H and L registers. To hold 16-bit data a combination of two 8-bit registers can be employed. The combination of two 8-bit registers is known as a **register-pair.** The valid register pairs in the 8085 are: B-C, D-E and H-L.

Program Counter (PC): It is a 16-bit special-purpose register. It is used to hold the memory address of the next instruction to be executed. It keeps the track of memory addresses of the instructions in a program while they are being executed. The microprocessor increments the content of the program counter during the execution of an instruction so that it points to the address of the next instruction in the program.

Stack Pointer (SP): It is a 16-bit special function register. The stack is a sequence of memory locations that works on LIFO (Last In First Out) principle. The Stack Pointer (SP) holds the address of the top element of data stored in the stack.

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

Temporary Register: It is an 8-bit register that holds the second operand during an arithmetic/logical operation.

Flag Register (Status Flags): The 8085 has 8-bit flag register specifying the status of the result generated in accumulator. The flag register contains status bits that indicates various status about the result generated. There are five flags in 8085 microprocessor. They are:

- SF (Sign Flag)
- ZF (Zero Flag)
- AC (Auxiliary Carry Flag)
- PF (Parity Flag)
- CF (Carry Flag)

The format of the flag register is shown below:

SF	ZF	X	AF	X	PF	X	CF

Sign Flag (SF): The sign flag indicates whether result generated is positive or negative. The sign flag is set to 1 whenever the result generated is negative. If the sign flag is set to 0 than the result generated is positive. The negative result sets the SF to 1. Generally, the MSB is considered as the bit that indicates if the result is positive or negative. The MSB = 1 means the result is negative and if MSB=0 then the result is positive.

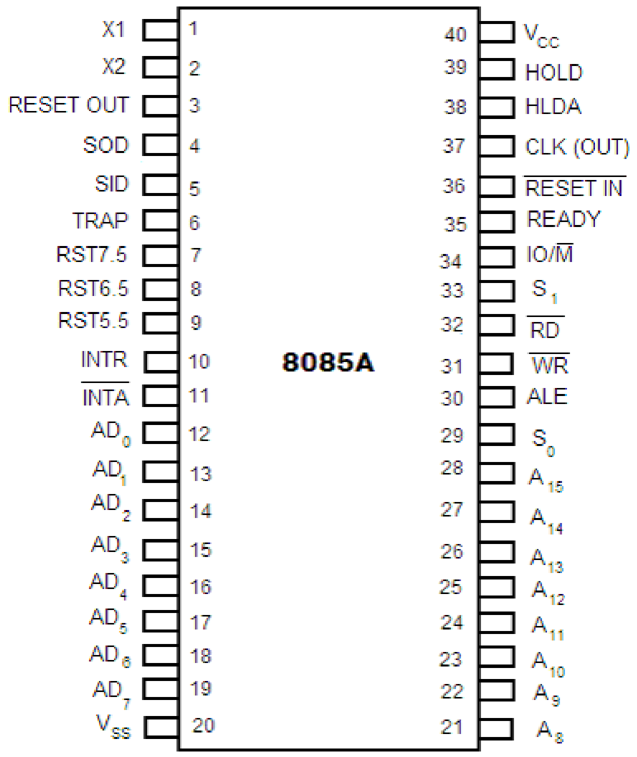
Zero Flag (ZF): The ZF indicates the result generated is zero or non-zero. If the result generated is zero than the Zero Flag is set to 1 else if the result is non-zero than the ZF=0.

Auxiliary Carry Flag (AF): The Auxiliary Carry flag is set to 1 whenever there is a carry or borrow at nibble in the data. If the carry generated from 4th bit to 5th bit or borrow is taken from 5th bit to 4th bit then the Auxiliary Carry Flag is set to 1.

Parity Flag (PF): The Parity Flag is set to 1 when the generated result is or even parity. The Parity Flag is reset to 0 when the parity of the result is odd.

Carry Flag (CF): The Carry flag is set to 1 whenever there is a carry from MSB or borrow to MSB. The flag is 0 otherwise.

Pin Configuration of 8085 microprocessor:



X1, X2: A quartz crystal required by the internal clock generator is kept outside the processor. This quartz crystal is connected to the internal clock generator through X1 and X2.

RESET IN: It is an active low input signal that resets the microprocessor.

RESET OUT: A high output signal on this pin indicates that the microprocessor is reset. This in turn resets external devices connected to the processor.

SID, SOD: These pins are used to carry out serial data transfer. The SID (Serial Input Data) pin is used to receive serial data, whereas the SOD (Serial Output Data) pin is used to send out serial data.

TRAP, RST 7.5, RST 6.5, RST 5.5, INTR, INTA: These pins are meant for interrupts. These are the signals initiated by an external device to request the microprocessor to do a particular task. TRAP has the highest priority and INTR the least priority. TRAP is a non-maskable interrupt whereas RST 7.5, RST 6.5 and RST 5.5 are maskable interrupts. On receipt of an interrupt, the microprocessor acknowledges the interrupt by an active low INTA (Interrupt Acknowledge) signal.

AD0 – AD7, A8 – A15: AD0 – AD7 are multiplexed address/data lines that are used to transfer the lower order 8 bits of the address and the 8-bit data. A8 – A15 are address lines that are used to transfer the higher order 8 bits of the address.

ALE: ALE stands for Address Latch Enable. A high output signal on this line indicates that the 16 lines AD0-AD7, A8-A15 contains valid address. This signal is low when data is being transferred through AD0-AD7. During the data transfer the lines A8-A15 are tristated (inactive).

IO/M: This pin is used to distinguish between an I/O operation and a memory operation. A high output signal on this pin indicates an I/O operation whereas a low output signal on this line indicates a memory operation.

S_0, S_1: These pins distinguish between various operations of the microprocessor. This is shown in the table below:

S1	**S0**	**Operation**
0	0	Halt
0	1	Write
1	0	Read
1	1	Fetch

RD: A low output signal on this pin indicates a read operation.

WR: A low output signal on this pin indicates a write operation.

READY: It is a signal sent by an input or output device to the microprocessor. This signal indicates that the device is ready to send or receive data.

HOLD, HLDA: These pins are used for carrying out DMA (Direct Memory Access) transfer of data. When a device requires address and data buses for data transfer, it sends HOLD signal to the microprocessor. The HOLD acknowledge signal is sent out by the microprocessor after receiving the HOLD signal.

CLOCK OUT: Through this pin the microprocessor generates clock signals that can used by other digital ICs.

V_{cc}, V_{ss}: These are power supply pins. +5V is applied to V_{cc}. and V_{ss} is ground reference.

Opcode and Operands

Each instruction contains two parts: operation code (opcode) and operand. The first part of the instruction which specifies the task to be performed by the computer is called opcode. The second part of the instruction is the data to be operated on, and it is called an operand. The operand (or data) given in the instruction may be in various forms such as 8-bit or 16-bit data, 8-bit or 16-bit address, or an internal register. In some instructions the operand is implicit.

Instruction Formats

According to the size of the instruction or the number of memory location(s) the instruction requires, the Intel 8085 instructions are classified into the following three types of instruction formats:

- 1-byte instruction
- 2-byte instruction
- 3-byte instruction

1-byte instruction: This type of instruction consists of a 1-byte opcode only. The opcode includes the details of the operands also. Examples of 1-byte instructions are:

MOV A, B	Move the content of register B to accumulator.
ADD B	Add the content of register B to the content of the accumulator.
RAL	Rotate the content of the accumulator left by one bit.

2-byte instruction: In a 2-byte instruction, the 1st byte of the instruction is its opcode and the 2nd byte is either an 8-bit data or 8-bit port address. Examples are:

MVI B, 05	Move 05 to register B.
IN 02	Receive the content of port 02 into accumulator

3-byte instruction: In a three-byte instruction the 1st byte of the instruction is its opcode and the 2nd and 3rd bytes are either 16-bit data or 16-bit address. Examples are:

LDA 4000	Load the content of memory location 4000 into accumulator.
LXI H, 2400	Load H-L pair with the 16-bit data 2400.

Instruction Cycle

An instruction is a command given to the computer to perform a specified operation on given data. To perform a particular task a programmer writes sequence of instructions, called a program. Program and data are stored in the memory. The CPU fetches one instruction from the memory at a time and executes it. It executes all the instructions of a program one by one to produce the final result.

The necessary steps that a CPU carries out to fetch an instruction and necessary data from the memory, and to execute it, constitute an instruction cycle (IC). An instruction cycle consists of fetch cycle and execute cycle, In fetch cycle a CPU fetches opcode from the memory. The necessary steps which are carried out to fetch an opcode from the memory, constitute a fetch cycle (FC). The necessary steps which are carried out to execute an instruction is called an execute cycle (EC). The time required to fetch an opcode is a fixed slot of time while the time required to execute an instruction is variable which depends on the type of instruction to be executed. The total time required to execute an instruction is given by IC=FC+EC

Fetch Operation: The 1st byte of an instruction is its opcode. An instruction may be more than one byte long. The other bytes are data or operand address. The Program Counter (PC) keeps the memory address of the next instruction to be executed. In the beginning of a fetch cycle the content of the program counter, which is the address of the memory location where opcode is available, is sent to the memory. The memory places the opcode on the data bus so as to transfer it to the CPU. The entire operation of fetching an opcode takes three clock cycles.

Execute Operation: The opcode fetched from the memory goes to the instruction register (IR). From the instruction register it goes to the instruction decoder which decodes the instruction. After the instruction is decoded, execution begins. If the operand is in a general-purpose register, execution is immediately performed. The time taken in decoding and execution is one clock cycle. If an instruction contains data or operand address which are still in the memory, the CPU has to perform some read operations to get the desired data. After receiving the data, it performs execute operation.

Addressing Modes

Each instruction requires certain data on which it has to operate. The addressing mode is a technique used by the microprocessor to fetch operands required by the instructions. There are five addressing modes in 8085 microprocessor. They are:

- Direct Addressing
- Register Addressing
- Register indirect addressing
- Immediate addressing
- Implicit / Implied addressing

Direct Addressing: In this mode of addressing, the address of the operand (data) is given in the instruction itself. Example:

    LDA 2400   : Load the content of memory location 2400 into accumulator.

In this instruction 2400 is the memory address from where data is to be fetched. It is given in the instruction itself. Other examples are STA 3000, LHLD 4300.

Register Addressing: In register addressing mode, the operand is in one of the general-purpose registers or accumulator. The opcode specifies the address of the register(s) in addition to the operation to be performed. Example:

    MOV A, B   : Move the content of register B to register A

Other examples are ADD B, CMP C

Register Indirect Addressing: In this mode of addressing the operand is specified by a register pair. In other words, the address of the operand is specified in a register pair.
Example:

```
        LXI H, 2500     :  Load H-L pair with 2500
        MOV A, M        : Move the content of the memory location, whose address is in H-L pair
(i.e.2500) to the accumulator.
```

In the above program the instruction MOV A, M is register indirect addressing instruction. For this instruction the operand is in the memory. The address of the memory is not directly given in the instruction. The address of the memory resides in the H-L pair.

Immediate Addressing: In this addressing mode, the operand is specified within the instruction itself. Example:

```
        MVI A, 05       : Move 05 in register A
```

In this instruction the 2nd byte specifies data.

Implied / Implicit Addressing: These instructions do not have any operands. The operations are implied. For instance, there are a few instructions such as CMA, RAL, RRC, etc. that operate on the content of the accumulator. Such instructions do not require the address of the operand.

<u>UNIT-3</u>

INSTRUCTION SET OF 8085

Data transfer Group

Move register to register:

MOV r1, r2 [r1] ← [r2]
The content of register r2 is moved to register r1.

Example: MOV A, B [A] ← [B]
This instruction moves the content of register B to register A.

Move memory to register:

MOV r, M [r] ← [[HL]]
The content of the memory location, whose address is in HL pair, is moved to register r.

Example: MOV B, M [r] ← [[HL]]
This instruction will move the content of the memory location, whose address is in HL pair, is moved to register B.

Move register to memory:

MOV M, r [[HL]] ← [r]
The content of register r is moved to the memory location addressed by HL pair.

Example: MOV M, C [[HL]] ← [C]
This instruction moves the content of register C to the memory location whose address is in HL pair.

Move immediate data to register:

MVI r, 8-bit data [r] ← 8-bit data
The 8-bit data specified in the instruction is moved to register r.

Example: MVI B, 05H [A] ← 8-bit data
The 8-bit data 05H specified in the instruction is moved to register B.

Move immediate data to memory:

MVI M, 8-bit data [[HL]] ← 8-bit data
The 8-bit data is moved to memory location whose address is in HL pair.

Example: MVI M, 35H [[HL]] ← 35H
The 8-bit data 35H is moved to memory location whose address is in HL pair.

Load register pair immediate:

LXI *rp*, 16-bit data [*rp*] ← 16-bit data
This instruction loads 16-bit data into a register pair *rp*. This instruction works with HL pair, BC pair, DE pair and SP.

Example: LXI H, 2000H [HL] ← 2000H
This instruction loads the 16-bit data 2000H into HL pair.

Load accumulator direct:

LDA address [A] ← [address]
The content of the memory location, whose address is specified in the instruction, is loaded into the accumulator.

Example: LDA 2400H [A] ← [2000H]
This instruction will load the content of the memory location 2400H into the accumulator.

Store accumulator direct:

STA address [address] ← [A]
The content of the accumulator is stored in the memory location whose address is specified in the instruction.

Example: STA 2000H [2000h] ← [A]
This instruction will store the content of the accumulator in the memory location 2000H.

Load HL pair direct:

LHLD address [L] ← [address], [H] ← [address+1]
The content of the memory location, whose address is specified in the instruction, is loaded into register L. The content of the next memory location is loaded into register H.

Example: LHLD 2500H [L] ← [2500H], [H] ← [2501H]
This instruction will load the content of the memory location 2500H into register L and the content of the memory location 2501H into register H.

Store HL pair direct:

SHLD address [address] ← [L], [address+1] ← [H].
The content of register L is stored in the memory location whose address is specified in the instruction. The content of register H is stored in the next memory location.

Example: SHLD 2500H [2500H] ← [L], [2501H] ← [H]
This instruction will store the content of register L in the memory location 2500H and the content of register H in the memory location 2501H.

Load accumulator indirect:

LDAX *rp* [A] ← [[*rp*]]
The content of the memory location, whose address is in the register pair *rp*, is loaded into the accumulator.

Example: LDAX B [A] ← [[BC]]
This instruction will load the content of the memory location, whose address is in the BC pair, into the accumulator. This instruction is used only for BC and DE register pairs.

Store accumulator indirect:

STAX *rp* [[*rp*]] ← [A]
The content of the accumulator is stored in the memory location whose address is in the register pair *rp*.

Example: STAX D [[DE]] ← [A]
This instruction will store the content of the accumulator in the memory location whose address is in DE pair. This instruction is used only for register pairs BC and DE.

Exchange the contents of HL with DE pair:

XCHG [HL] ↔ [DE]
The contents of HL pair are exchanged with contents of DE pair.

Arithmetic Instructions

Add register to accumulator:

ADD r [A] ← [A] + [r]
The content of register r is added to the content of the accumulator, and the sum is placed in the accumulator.

Example: ADD B [A] ← [A] + [B]
This instruction adds the content of register B to the content of the accumulator.

Add memory to accumulator:

ADD M [A] ← [A] + [[HL]]
The content of the memory location addressed by HL pair is added to the content of the accumulator. The sum is placed in the accumulator.

Add immediate data to accumulator:

ADI 8-bit data [A] ← [A] + 8-bit data.
The immediate data is added to the content of the accumulator. The sum is placed in the accumulator.

Example: ADI 08H [A] ← [A] + 08H
This instruction will add the data 08 to the content of the accumulator and place the result in the accumulator.

Add register with carry to accumulator:

ADC r [A] ← [A] + [r] + [CF]
The contents of register r and carry flag are added to the content of the accumulator. The sum is placed in the accumulator.

Example: ADC D [A] ← [A] + [D] + [CF]
This instruction adds the content of register D and the carry flag to the content of the accumulator.

Add memory with carry to accumulator:

ADC M [A] ← [A] + [[HL]] + [CF]
The contents of the memory location addressed by HL pair and carry flag are added to the content of the accumulator. The sum is placed in the accumulator.

Add with carry immediate data to accumulator:

ACI 8-bit data [A] ← [A] + 8-bit data + [CF]
The 8-bit data and the carry flag are added to the content of the accumulator. The sum is placed in the accumulator.

Example: ACI 23H [A] ← [A] + 23H + [CF]
The data 23H and the carry flag are added to the accumulator.

Add register pair to HL pair:

DAD *rp* [HL] ← [HL] + [*rp*]
The contents of register pair *rp* are added to the contents of HL pair and the result is placed in HL pair.

Example: DAD B [HL] ← [HL] + [BC]
This instruction adds the contents of BC pair to the content of HL pair.

Subtract register from accumulator:

SUB r [A] ← [A] - [r]
The content of register r is subtracted from the content of the accumulator, and the result is placed in the accumulator.

Example: SUB C [A] ← [A] - [C]
The content of register C is subtracted from the content of the accumulator, and the result is placed in the accumulator.

Subtract memory from accumulator:

SUB M [A] ← [A] - [[H-L]]
The content of the memory location addressed by HL pair is subtracted from the content of the accumulator. The sum is placed in the accumulator.

Subtract immediate data from accumulator:

SUI 8-bit data [A] ← [A] - 8-bit data.
The immediate data is subtracted from the content of the accumulator. The result is placed in the accumulator.

Example: SUI 08H [A] ← [A] - 08H
This instruction will subtract the data 08 from the content of the accumulator and place the result in the accumulator.

Subtract register with borrow from accumulator:

SBB r [A] ← [A] - [r] - [CF]
The contents of register r and carry flag are subtracted from the content of the accumulator. The result is placed in the accumulator.

Example: SBB D [A] ← [A] - [D] - [CF]
This instruction subtracts the content of register D and the carry flag from the content of the accumulator.

Subtract memory with borrow from accumulator:

SBB M [A] ← [A] - [r] - [[HL]]
The contents of the memory location addressed by HL pair and carry flag are subtracted from the content of the accumulator. The result is placed in the accumulator.

Subtract immediate data with borrow from accumulator:

SBI 8-bit data [A] ← [A] - 8-bit data - [CF]
The 8-bit data and the carry flag are subtracted from the content of the accumulator. The result is placed in the accumulator.

Example: SBI 23H [A] ← [A] - 23H - [CF]
The data 23H and the carry flag are subtracted from the accumulator

Increment register:

INR r [r] ← [r] + 1
The content of register r is incremented by 1.

Example: INR B [B] ← [B] + 1
The content of register B is incremented by 1.

Increment memory:

INR M [[HL]] ← [[HL]] + 1
The contents of the memory location addressed by HL pair, is incremented by 1.

Increment register pair:

INX *rp* [*rp*] ← [*rp*] + 1
The content of register pair *rp* is incremented by 1.

Example: INX H [HL] ← [HL] + 1
The content of HL pair is incremented by 1.

Decrement register:

DCR r [r] ← [r] - 1
The content of register r is incremented by 1.

Example: DCR B [B] ← [B] - 1
The content of register B is decremented by 1.

Decrement memory:

DCR M [[HL]] ← [[HL]] - 1
The contents of the memory location addressed by HL pair, is decremented by 1.

Decrement register pair:

DCX *rp* [*rp*] ← [*rp*] - 1
The content of register pair *rp* is decremented by 1.

Example: DCX B [BC] ← [BC] - 1
The content of BC pair is decremented by 1.

Decimal Adjust Accumulator:

DAA
This instruction converts the content of accumulator into decimal form.

Logical Instructions

AND register with accumulator:

ANA r [A] ← [A] ∧ [r]
The content of register r is logically ANDed with the content of the accumulator, and the result is placed in the accumulator.

Example: ANA B $[A] \leftarrow [A] \wedge [B]$
The content of register B is logically ANDed with the content of the accumulator, and the result is placed in the accumulator.

AND memory with accumulator:

ANA M $[A] \leftarrow [A] \wedge [[HL]]$
The content of memory location addressed by HL pair is logically ANDed with the content of the accumulator, and the result is placed in the accumulator.

AND immediate with accumulator:

ANI 8-bit data $[A] \leftarrow [A] \wedge$ 8-bit data
The 8-bit data is logically ANDed with the content of the accumulator, and the result is placed in the accumulator.

Example: ANI 34H $[A] \leftarrow [A] \wedge 34H$
The 8-bit data 34H is logically ANDed with the content of the accumulator, and the result is placed in the accumulator.

OR register with accumulator:

ORA r $[A] \leftarrow [A] \vee [r]$
The content of register r is logically ORed with the content of the accumulator, and the result is placed in the accumulator.

Example: ORA B $[A] \leftarrow [A] \vee [B]$
The content of register B is logically ORed with the content of the accumulator, and the result is placed in the accumulator.

OR memory with accumulator:

ORA M $[A] \leftarrow [A] \vee [[HL]]$
The content of memory location addressed by HL pair is logically ORed with the content of the accumulator, and the result is placed in the accumulator.

OR immediate with accumulator:

ORI 8-bit data $[A] \leftarrow [A] \vee$ 8-bit data
The 8-bit data is logically ORed with the content of the accumulator, and the result is placed in the accumulator.

Example: ORI 34H $[A] \leftarrow [A] \vee 34H$
The 8-bit data 34H is logically ORed with the content of the accumulator, and the result is placed in the accumulator.

XOR register with accumulator:

XRA r $[A] \leftarrow [A] \oplus [r]$
The content of register r is logically XORed with the content of the accumulator, and the result is placed in the accumulator.

Example: XRA B $[A] \leftarrow [A] \oplus [B]$
The content of register B is logically XORed with the content of the accumulator, and the result is placed in the accumulator.

XOR memory with accumulator:

XRA M $[A] \leftarrow [A] \oplus [[HL]]$
The content of memory location addressed by HL pair is logically XORed with the content of the accumulator, and the result is placed in the accumulator.

XOR immediate with accumulator:

XRI 8-bit data $[A] \leftarrow [A] \oplus$ 8-bit data
The 8-bit data is logically XORed with the content of the accumulator, and the result is placed in the accumulator.

Example: XRI 34H $[A] \leftarrow [A] \oplus 34H$
The 8-bit data 34H is logically XORed with the content of the accumulator, and the result is placed in the accumulator.

Complement the accumulator:

CMA $[A] \leftarrow [A']$

1's complement of the content of the accumulator is obtained, and the result is placed in the accumulator.

Complement the carry flag:

CMC $[CF] \leftarrow [CF']$ The carry flag, CF, is complemented.

Set the carry flag:

STC $CF \leftarrow 1$

Compare register with accumulator:

CMP r

The content of register r is compared with the content of the accumulator and the corresponding status flags are set. The content of the accumulator remains unchanged. If $[A] < [r]$, then $CF \leftarrow 1$. If $[A] = [r]$, then $ZF \leftarrow 1$. If the contents of CF and ZF are not 1's, then $[A] > [r]$.

Example: CMP B

The content of register B is compared with the content of the accumulator and the corresponding status flags are set. The content of the accumulator remains unchanged. If [A] < [B], then CF ← 1. If [A] = [B], then ZF ← 1. If the contents of CF and ZF are not 1's, then [A] > [B].

Compare memory with accumulator:

CMP M

The content of memory location addressed by HL pair is compared with the content of the accumulator and the corresponding status flags are set. The content of the accumulator remains unchanged. If [A] < [[HL]], then CF ← 1. If [A] = [[HL]], then ZF ← 1. If the contents of CF and ZF are not 1's, then [A] > [[HL]].

Compare immediate data with accumulator:

CPI 8-bit data

The 8-bit data is compared with the content of the accumulator and the corresponding status flags are set. The content of the accumulator remains unchanged. If [A] < 8-bit data, then CF ← 1. If [A] = 8-bit data, then ZF ← 1. If the contents of CF and ZF are not 1's, then [A] > 8-bit data.

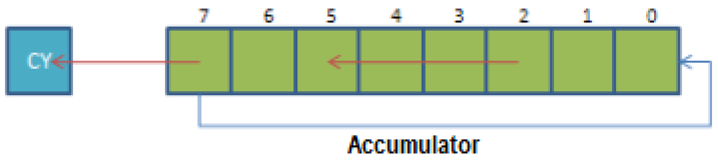
Example: CPI 32H

Rotate instructions

Rotate accumulator left:

RLC

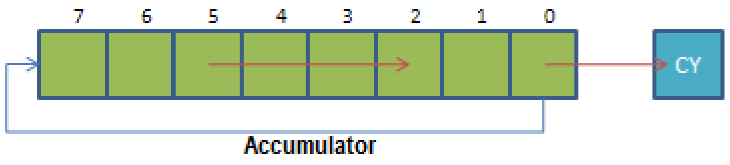
The content of the accumulator is rotated left by one bit. The MSB (A7) of the accumulator is moved to carry bit as well as to LSB (A0) of the accumulator.



Rotate accumulator right:

RRC

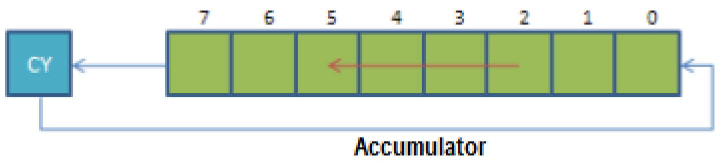
The content of the accumulator is rotated right by one bit. The LSB (A0) of the accumulator is moved to carry bit as well as to MSB (A7) of the accumulator.



Rotate accumulator left through carry:

RAL

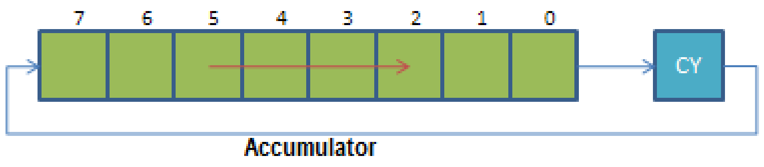
The content of the accumulator is rotated left one bit through carry. The MSB of the accumulator is moved to carry, and the carry bit is moved to the LSB of the accumulator.



Rotate accumulator right through carry:

RAL

The content of the accumulator is rotated right one bit through carry. The LSB of the accumulator is moved to carry, and the carry bit is moved to the MSB of the accumulator.



Branch instructions

The instructions of this group change the normal sequence of the program. There are two types of branch instructions: conditional and unconditional. The conditional branch instructions transfer the program to the specific label when certain condition is satisfied. The unconditional branch instructions transfer the program to the specified label unconditionally.

Unconditional Jump:

JMP address [PC] ← Address

The program jumps to the instruction specified by the 'address' unconditionally. In other words, the 'address' specified in the instruction is stored in the program counter so that the control is taken to the instruction specified by the 'address'.

Conditional Jump:

The program jumps to the instruction specified by the 'address', if the specified condition is fulfilled. In other words, the 'address' specified in the instruction is stored in the program counter only if the specified condition is fulfilled.

JC address [PC] ← Address, if CF = 1
The program jumps to the instruction specified by the 'address', if CF = 1

JNC address [PC] ← Address, if CF = 0
The program jumps to the instruction specified by the 'address', if CF = 0

JZ address [PC] ← Address, if ZF = 1
The program jumps to the instruction specified by the 'address', if ZF = 1

JNZ address [PC] ← Address, if ZF = 0
The program jumps to the instruction specified by the 'address', if ZF = 0

JM address [PC] ← Address, if SF = 1
The program jumps to the instruction specified by the 'address', if SF = 1

JP address [PC] ← Address, if SF = 0
The program jumps to the instruction specified by the 'address', if SF = 0

JPE address [PC] ← Address, if PF = 1
The program jumps to the instruction specified by the 'address', if PF = 1

JPO address [PC] ← Address, if PF = 0
The program jumps to the instruction specified by the 'address', if PF = 0

Unconditional Call:

CALL address
[[SP] – 1] ← [PCH]
[[SP] – 2] ← [PCL]
[SP] ← [SP]-2
[PC] ← address

CALL instruction is used to call a subroutine. Before the control is transferred to the subroutine, the address of the next instruction (in the PC) of the main program is saved in the stack. The content of the stack pointer is decremented by two to indicate the new stack top. Then the program jumps to subroutine starting at 'address', specified in the instruction.

Conditional Call:

The program calls a subroutine, if the specified condition is fulfilled.

CC address Call the subroutine, if CF = 1

If CF = 1, then
[[SP] – 1] ← [PCH]
[[SP] – 2] ← [PCL]
[SP] ← [SP]-2
[PC] ← address

CNC address	Call the subroutine, if CF = 0
CZ address	Call the subroutine, if ZF = 1
CNZ address	Call the subroutine, if ZF = 0
CM address	Call the subroutine, if SF = 1
CP address	Call the subroutine, if SF = 0
CPE address	Call the subroutine, if PF = 1
CPO address	Call the subroutine, if PF = 0

Unconditional Return:

RET (Return from subroutine to the main program)

[PCL] ← [[SP]],
[PCH] ← [[SP] + 1]
[SP] ← [SP] + 2

RET instruction is used at end of subroutine. Before execution of subroutine address of next instruction of main program is saved in stack. The execution of RET instruction brings back the saved address from stack to program counter. The content of stack pointer is incremented by two to indicate new stack top. Then the program jumps to instruction of main program next to CALL instruction which called the subroutine

Conditional Return:

If condition is true, the program returns from subroutine

[PCL] ← [[SP]]
[PCH] ← [[SP] +1]
[SP] ← [SP] +2

RC	address	Return from subroutine, if CF = 1
RNC	address	Return from subroutine, if CF = 0
RZ	address	Return from subroutine, if ZF = 1
RNZ	address	Return from subroutine, if ZF = 0
RM	address	Return from subroutine, if SF = 1
RP	address	Return from subroutine, if SF = 0
RPE	address	Return from subroutine, if PF = 1
RPO	address	Return from subroutine, if PF = 0

Restart Instructions:

RST *n* (where *n* = 0 to 7)

[[SP] – 1] ← [PCH]
[[SP] – 2] ← [PCL]
[SP] ← [SP] - 2
[PC] ← 8 times *n*

Restart is a one-word CALL instruction. The content of PC is saved in stack. The program jumps to instruction starting at restart location ‘*n*’. The address of restart location is 8 times *n*. The restart instructions and their locations are as follows:

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

Stack, I/O and Machine Control Group

Stack instructions:

PUSH *rp*
[[SP] – 1] ← [*rp*h]
[[SP] – 2] ← [*rp*l]
[SP] ← [SP] – 2
The content of a register pair ‘*rp*’ is pushed onto the stack.

Example: PUSH H
[[SP] – 1] ← [H]
[[SP] – 2] ← [L]
[SP] ← [SP] – 2
The content of a HL pair is pushed onto the stack.

PUSH PSW (PSW = Accumulator [A] + Flags [F])
[[SP] – 1] ← [A]
[[SP] – 2] ← [F]
[SP] ← [SP] – 2
The contents of the accumulator and flags are pushed onto the stack. The content of SP is decremented by 2 to indicate new stack top.

POP *rp*
[*rp*l] ← [[SP]]
[*rp*h] ← [[SP] + 1]
[SP] ← [SP] + 2
The contents of the top two locations of the stack are moved to a register pair.

Example: POP B
[C] ← [[SP]]
[B] ← [[SP] + 1]
[SP] ← [SP] + 2
The contents of the top two locations of the stack are moved to a BC pair.

POP PSW (PSW = Accumulator [A] + Flags [F])
[F] ← [[SP]]
[A] ← [[SP] + 1]
[SP] ← [SP] + 2
The contents of the top two locations of the stack are moved to the flag register and accumulator.

XTHL
[L] ↔ [[SP]]
[H] ↔ [[SP] +1]
The contents of HL pair and the top two elements of stack are mutually exchanged.

SPHL
[SP] ← [HL]
The contents of HL pair are transferred to stack pointer SP.

I/O Instructions:

IN port# (where port# is an 8-bit port address)
[A] ← [port#]
The data available on the specified port is moved to accumulator

OUT port# (where port# is an 8-bit port address)
[port#] ← [A]
The data available in the accumulator is moved to the specified port.

Other Instructions:

PCHL
[PC] ← [HL]

The contents of HL pair are transferred to the program counter PC.

EI (Enable Interrupt)
When this instruction is executed, the interrupts are enabled.

DI (Enable Interrupt)
When this instruction is executed, the interrupts are disabled.

SIM (Set Interrupt Masks)
This instruction is used to enable/disable the maskable interrupts RST7.5, RST6.5 and RST5.5. It is also used to send serial data output to SOD pin of the microprocessor.

RIM (Read Interrupt Masks)
This instruction is used to read the status of the maskable interrupts including the pending interrupts. It also reads serial input data from the SID pin of the microprocessor.

HLT (Halt)
The execution of instruction HLT stops the microprocessor. The registers and status flags remain unaffected.

NOP (No Operation)
No operation is performed when this instruction is executed. The registers and flags remain unaffected.

Assembly Language:

Each personal computer has a microprocessor that manages the computer's arithmetical, logical, and control activities. Each family of processors has its own set of instructions for handling various operations such as getting input from keyboard, displaying information on screen and performing various other jobs. These set of instructions are called 'machine language instructions'. A processor understands only machine language instructions, which are strings of 1's and 0's. However, writing of programs in machine language is very difficult, tiresome and error prone. Hence, to facilitate programmers, easily understandable languages have been developed. Meaningful and easily understandable symbolic codes are used for this purpose. Examples are ADD for addition, SUB for subtraction, CMP for compare, etc. Such symbols are called *mnemonics*. A program written in mnemonics is known as *assembly language program*.

Assemblers:
A program which translates an assembly language program into a machine language program is called an *assembler*. A *self-assembler* or *resident assembler* is an assembler which runs on the microcomputer for it produces object codes (machine codes). A *cross-assembler* is an assembler that runs on computer other than that for which it produces object codes.

Disassembler:
A *disassembler* is a software that converts a machine language program to an assembly language program. Its role is opposite to that of an assembler. It is useful in situations, when a program is available in machine language and it is to converted to assembly language for better understanding.

One-Pass Assembler:
An assembler which goes through an assembly language program only once is known as *one-pass assembler*. Such an assembler must have some technique to take the forward references into account. Assembly language programs use labels that may appear later on in the program. Such labels point to forward references.

Two-Pass Assembler:
An assembler which goes through an assembly language program twice is known as *two-pass assembler*. Such an assembler does not face difficulty with forward references. During the first pass it collects all labels. During the second pass it produces the machine code of each instruction and assigns addresses to each of them. It assigns addresses to labels by counting their position from the starting address. Two-pass assemblers are commonly used.

Stacks

During the execution of a program sometimes it becomes necessary to save the contents of certain registers, in the memory, because the registers are required for some other operation in subsequent steps. The memory locations used for storing the contents of such registers is called a *stack*. Stack is a Last in First out memory. The stack is used by the program to save the *return address* (the address to which the control has to return after executing a subroutine) before transferring control to a subroutine. The 16-bit register that holds the address of the top element of stack is called the *stack pointer*. The stack pointer is initialized using the instruction, LXI SP, 5000, where 5000 will considered as the stack top location.

Operations on Stack:

Intel 8085 microprocessor provides two operations on stack – namely, *push* and *pop* operations. The push operation is carried out using PUSH instruction and the pop operation is carried out using POP instruction.

The PUSH instruction is used to push the contents of a register pair or PSW (Processor Status Word) onto stack. For example, PUSH B will push the contents of BC pair onto stack. The POP instruction is used to pop out the top two elements of stack into the specified register pair or PSW. For example, POP D will pop out the top two elements of stack and store them into the DE pair.

Subroutines

A *subroutine* is a group of instructions that will be used repeatedly in different locations of the program. Rather than repeating the same instructions several times, they can be grouped into a subroutine that is called from the different locations. In Assembly language, a subroutine can exist anywhere in the code. However, it is customary to place subroutines separately from the main program. They are called at various points of the main program by a CALL instruction where they are required. When a CALL instruction is executed, the current contents of the program counter (the address of the next instruction – return address), is saved on top of stack. The program then transfers control to the subroutine. A subroutine ends with a RET instruction. When the RET instruction is executed, the program returns to the main program and begins executing the instruction next to CALL instruction until the end of program is reached.

Macros

A sequence of instructions to which a name is assigned is called a *macro.* The name of the macro is used in assembly language programs. The macro facility is available with many assemblers. Each macro is clearly defined and unique name is assigned to it. The structure of a macro is shown below:

```
<Name of Macro>    MACRO      <Arguments>
                   ---
                   ---
                   ---
                   ENDM
```

where ENDM denotes the end of macro.

```
Example:    COMP        MACRO       ADDR
                        LXI H, ADDR
                        MOV A, M
                        CMA
                        ENDM
```

In the above example, COMP is the name of the macro, ADDR is an argument and ENDM indicates the end of macro. If we include the macro call as COMP 2500 in the main program, the assembler will replace this macro with the following sequence of instructions in the main program.

```
                        LXI H, 2500
                        MOV A, M
                        CMA
```

This sequence of instructions will take the complement of the content of memory location 2500.

UNIT-4

ASSEMBLY LANGUAGE PROGRAMS

8-Bit Addition

The first number 04 is in the memory location 4200.
The second number 07 is in the memory location 4201
The result is to be stored in the memory location 4202

Address	Data
4200	04
4201	07
4202	**0B**

```
LXI H, 4200    Get the address of the first number in HL pair
MOV A, M       Move the first number into the accumulator
INX H          Increment the content of HL pair
MOV B, M       Move the second number into register B
ADD B          Add the two numbers
INX H          Increment the content of HL pair
MOV M, A       Store the result in [[HL]] pair i.e. in the location 4202
HLT            Stop
```

8-Bit Subtraction

The first number 04 is in the memory location 4200.
The second number 07 is in the memory location 4201
The result is to be stored in the memory location 4202

Address	Data
4200	08
4201	02
4202	**06**

```
LXI H, 4200    Get the address of the first number in HL pair
MOV A, M       Move the first number into the accumulator
INX H          Increment the content of HL pair
MOV B, M       Move the second number into register B
SUB B          Subtract content of B from A
INX H          Increment the content of HL pair
MOV M, A       Store the result in [[HL]] pair i.e. in the location 4202
HLT            Stop
```

8-Bit Multiplication

The first number 04 is in the memory location 4200.
The second number 03 is in the memory location 4201
The result is to be stored in the memory location 4202

Address	Data	
4200	04	Data-1
4201	03	Data-2
4202	**0C**	Result

```
        LXI H,4200
        MOV B, M
        INX H
        MOV C, M
        SUB A
NEXT:   ADD B
        DCR C
        JNZ NEXT
        INX H
        MOV M, A
        HLT
```

8-Bit Division

The Numerator 0A is in the memory location 4200.
The Denominator 03 is in the memory location 4201
The Quotient is to be stored in the memory location 4202
The Remainder is to be stored in the memory location 4203

```
                LXI H,4200
                MOV A, M
                INX H
                MOV B, M
                MVI C, 00
NEXT:           CMP B
                JC DONE
                SUB B
                INRC
                JMP NEXT
DONE:           INX H
                MOV M, C
                INX H
                MOV M, A
                HLT
```

Address	Data	
4200	0A	NR
4201	03	DR
4202	**03**	Quotient
4203	**01**	Remainder

Decimal Addition

```
                LXI H, 4200
                MOV A, M
                INX H
                MOV B, M
                ADD B
                DAA
                INX H
                MOV M, A
                HLT
```

Address	Data	
4200	09	
4201	07	
4202	**16**	Sum

Decimal Subtraction

```
                LXI H, 4200
                MOV D, M
                INX H
                MOV E, M
                MVI A, 99
                SUB E
                INR A
                ADD D
                DAA
                INX H
                MOV M, A
                HLT
```

Address	Data	
4200	10	
4201	07	
4202	**09**	Result

Multibyte Addition

```
        STC
        CMC
        LXI D, 4200
        LXI H, 4300
        MVI C, 05
NEXT :  LDAX D
        ADC M
        MOV M,A
        INX H
        INX D
        DCR C
        JNZ NEXT
        JNC LAST
        MVI M, 01
LAST :  HLT
```

Address	Data
Number-1	
4200	44
4201	44
4202	44
4203	44
4204	44
Number-2	
4300	33
4301	33
4302	33
4303	33
4304	33

Address	Data
Result	
4300	**77**
4301	**77**
4302	**77**
4303	**77**
4304	**77**
4305	**00**

Multibyte Subtraction

```
        STC
        CMC
        LXI D, 4200
        LXI H, 4300
        MVI C, 05
NEXT :  LDAX D
        SBB M
        MOV M, A
        INX H
        INX D
        DCR C
        JNZ NEXT
        HLT
```

Address	Data
Number-1	
4200	77
4201	77
4202	77
4203	77
4204	77
Number-2	
4300	33
4301	33
4302	33
4303	33
4304	33

Address	Data
Result	
4300	**44**
4301	**44**
4302	**44**
4303	**44**
4304	**44**

1's Complement of an 8-Bit Number

```
        LXI H, 4200
        MOV A, M
        CMA
        INX H
        MOV M, A
        HLT
```

Address	Data	
4200	00	
4201	**FF**	1's Complement

2's Complement of an 8-Bit Number

```
        LXI H, 4200
        MOV A, M
        CMA
        INR A
        INX H
        MOV M, A
        HLT
```

Address	Data	
4200	05	
4201	**FB**	2's Complement

Assembly of a Byte

```
LXI H, 4200
MOV B, M
INX H
MOV A, M
RRC
RRC
RRC
RRC
ADD B
INX H
MOV M, A
HLT
```

Address	Data	
4200	05	
4201	04	
4202	**45**	Result

Disassembly of a Byte

```
LXI H, 4200
MOV A, M
MOV B, A
ANI 0F
INX H
MOV M, A
MOV A, B
RRC
RRC
RRC
RRC
ANI 0F
INX H
MOV M, A
HLT
```

Address	Data	
4200	45	
4201	**05**	Result
4202	**04**	

Sum of a Series

```
        LXI H, 4200
        MVI C, 05
        SUB A
NEXT :  ADD M
        INX H
        DCR C
        JNZ NEXT
        MOV M, A
        HLT
```

Address	Data	
4200	15	
4201	24	
4202	21	
4203	10	
4304	11	
4305	**51**	Sum

Block Data Transfer

```
            LXI H, 4200
            LXI D, 4300
            MVI C, 05
NEXT :      MOV A, M
            STAX D
            INX H
            INX D
            DCR C
            JNZ NEXT
            HLT
```

Address	Data
Source	
4200	23
4201	35
4202	46
4203	24
4204	65

Address	Data
Destination	
4300	**23**
4301	**35**
4302	**46**
4303	**24**
4304	**65**

Finding the Smallest Number in an Array

```
            LXI H, 4200
            MVI C, 05
            DCR C
            MOV A, M
LOOP :      INX H
            CMP M
            JC NEXT
            MOV A, M
NEXT :      DCR C
            JNZ LOOP
            INX H
            MOV M, A
            HLT
```

Address	Data	
4200	15	
4201	24	
4202	21	
4203	10	
4304	11	
4305	**10**	Smallest

Finding the Biggest Number in an Array

```
            LXI H, 4200
            MVI C, 05
            DCR C
            MOV A, M
LOOP :      INX H
            CMP M
            JNC NEXT
            MOV A, M
NEXT :      DCR C
            JNZ LOOP
            INX H
            MOV M, A
            HLT
```

Address	Data	
4200	15	
4201	24	
4202	21	
4203	10	
4304	11	
4305	**24**	Biggest

Arranging a Series of 8-bit Numbers in Ascending Order

```
            MVI D, 04
START :     LXI H, 4200
            MVI C, 04
LOOP :      MOV A, M
            INX H
            MOV B, M
            CMP B
            JC NEXT
            MOV M, A
            DCX H
            MOV M, B
            INX H
NEXT:       DCR C
            JNZ LOOP
            DCR D
            JNZ START
            HLT
```

Address	Data
Given Data	
4200	20
4201	65
4202	46
4203	24
4204	15

Address	Data
Result	
4200	**15**
4201	**20**
4202	**24**
4203	**46**
4204	**65**

Arranging a Series of 8-bit Numbers in Descending Order

```
            MVI D, 04
START :     LXI H, 4200
            MVI C, 04
LOOP :      MOV A, M
            INX H
            MOV B, M
            CMP B
            JNC NEXT
            MOV M, A
            DCX H
            MOV M, B
            INX H
NEXT:       DCR C
            JNZ LOOP
            DCR D
            JNZ START
            HLT
```

Address	Data
Given Data	
4200	20
4201	65
4202	46
4203	24
4204	15

Address	Data
Result	
4200	**65**
4201	**46**
4202	**24**
4203	**20**
4204	**15**

UNIT-5

PERIPHERAL DEVICES AND INTERFACING

Memories and I/O devices are interfaced to the microprocessor to form a microcomputer. In the case of large and minicomputers the memories and input/output devices are interfaced to CPU by the manufacturer. In a microprocessor-based system the designer has to select suitable memories and input/output devices for his task and interface them to the microprocessor. The selected memories and I/O devices should be compatible with microprocessor.

Address Space Partitioning

The Intel 8085 uses 16-bit wide address bus for addressing memories and I/O devices. Using 16-bit wide address bus it can access 2^{16} = 64K bytes of memory and I/O devices. The 64K addresses are to be assigned to memories and I/O devices for their addressing. There are two schemes for allocation of addresses to memories and I/O devices:

1. Memory mapped I/O scheme
2. I/O mapped I/O scheme

Memory mapped I/O scheme

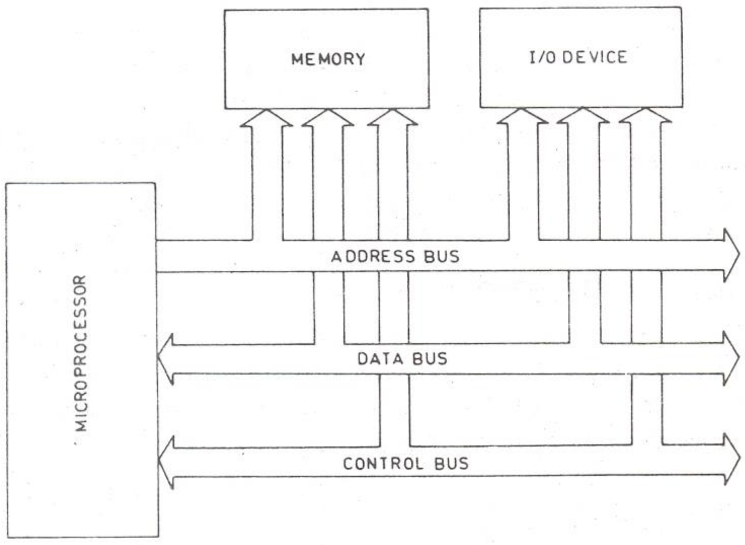
In this scheme, there is only one address space. Address space is defined as set of all possible addresses that a microprocessor can generate. Some addresses are assigned to memories and some addresses to I/O devices. An I/O device is also treated as a memory location and one address is assigned to it. One address is assigned to each memory location. The addresses for I/O devices are different from the addresses which have been assigned to memories and vice-versa. In this scheme all the data transfer instructions of microprocessor can be used for both memory as well as I/O devices. This scheme is suitable for small systems.

I/O mapped I/O scheme

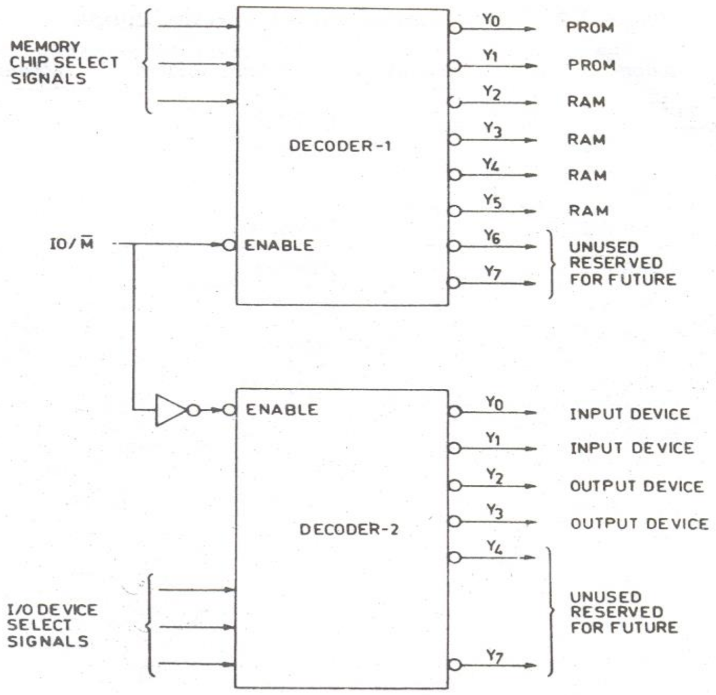
In this scheme the addresses assigned to memory locations can also be assigned to I/O devices. Since the same address may be assigned to memory location or I/O devices, the microprocessor must issue a special signal to distinguish whether the address on the address bus is for a memory location or an I/O device. The Intel 8085 issues an IO/M signal for this purpose. When it is high, address on address bus is for an I/O device. If it is low, the address on the address bus is for a memory location. Two extra instructions IN and OUT are used to address I/O devices. The IN instruction is used to read data of an input device. The OUT instruction is used to send data to an output device. This scheme is suitable for large systems.

Memory and I/O Interfacing

Several memory chips and I/O devices are connected to a microprocessor. The following diagram shows a schematic diagram to interface memory chips or I/O devices to the microprocessor.



An address decoding circuit is employed to select the required I/O device or memory chip. The following diagram shows a schematic diagram of the decoding circuit.



: Interfacing of Memory and I/O Devices.

If IO/M’ is high the DECODER-2 is activated and required I/O device is selected. If it is low, DECODER-1 is activated and required memory chip is selected. A few MSBs of address lines are applied to decoder to select a memory chip or an I/O device.

Data Transfer Schemes

In a microprocessor-based system, data transfer takes place between two devices such as microprocessor and memory, microprocessor and I/O devices, and memory and I/O devices. Such a system has several I/O devices of different speed. A slow I/O device cannot transfer data when the microprocessor issues instruction for the same because it takes some time to get ready. To solve the problem of speed mismatch between a microprocessor and I/O devices a number of data transfer schemes have been developed. The data transfer schemes are classified into two broad categories:

- Programmed data transfer schemes
- DMA (Direct Memory Access) data transfer scheme.

Programmed data transfer schemes

These schemes are controlled by the central processing unit. Data are transferred from an I/O device to CPU which resides in memory. These programs are executed by CPU when an I/O device is ready to transfer data. The programmed data transfer schemes are employed when small amount of data is to be transferred. They are classified into following categories.

- Synchronous data transfer scheme
- Asynchronous data transfer scheme
- Interrupt driven data transfer scheme

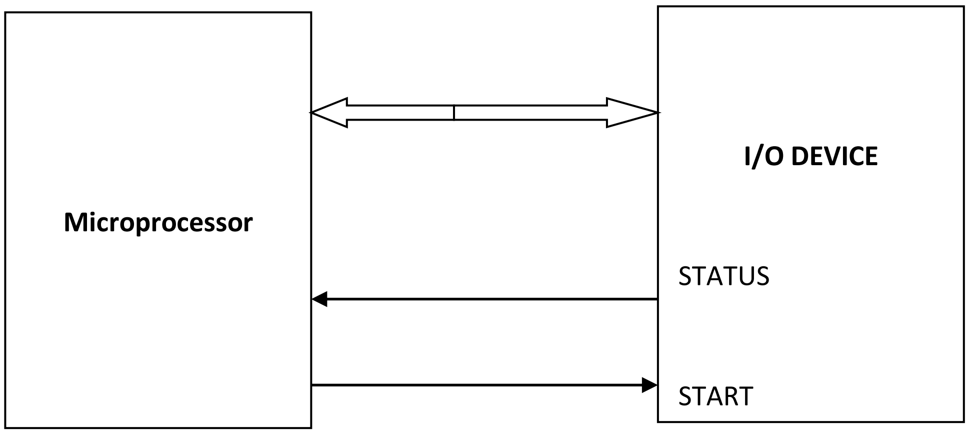
Synchronous Data Transfer

Synchronous means at the same time. The device which sends data and the device which receives data are synchronized with the same clock. When the CPU and I/O devices match in speed, this technique of the data transfer is employed. The data transfer with I/O device is performed executing IN or OUT instructions for I/O mapped I/O devices or using memory read/write instructions for memory mapped I/O devices. The IN instruction is used to read data from an input device or input port. The OUT instruction is used to send data from the CPU to an output device or output port. As the CPU and the I/O device match in speed, the I/O device is ready to transfer data when IN or OUT instruction is executed by the CPU. But I/O devices compatible with microprocessors in speed are usually not available. Hence this technique of data transfer is rarely used for I/O devices.

Asynchronous Data Transfer

This technique of data transfer is used when the speed of an I/O device does not match the speed of the microprocessor, and the timing characteristic of I/O device is not predictable. In this technique the status of the I/O device is checked by the microprocessor before the data are transferred. The microprocessor initiates the I/O device to get ready and then continuously checks the status of the I/O device till the I/O device becomes ready to transfer data. This mode of data transfer is called handshaking mode of transfer. In this handshaking mode of data transfer some signals are exchanged between the I/O device and the microprocessor before the actual data transfer takes place. The microprocessor issues an initiating signal to the I/O device to get ready. When an I/O device becomes ready it sends signals to the processor to indicate that it is ready. Such signals are called handshake signals.

The following diagram shows the schematic diagram for asynchronous data transfer. Asynchronous data transfer is used for slow I/O devices. This technique is inefficient technique because time is wasted.



Asynchronous data transfer

Interrupt Driven Data Transfer

In this scheme, the microprocessor initiates an I/O device to get ready, and then it executes its main program instead of remaining in a program loop to check the status of the I/O device. When the I/O device becomes ready to transfer data, it sends a high signal to the microprocessor through a special input line called an interrupt line. In other words, it interrupts the normal processing sequence of the microprocessor. On receiving an interrupt, the microprocessor completes the current instruction at hand, and then attends the I/O device. It saves the contents of the program counter on the stack first, and then takes up a subroutine called ISS (Interrupt Service Subroutine). It executes ISS to transfer data from or to the I/O device. After completing the data transfer, the microprocessor returns back to the main program which it was executing before the interrupt occurred. Interrupt driven data transfer is used for slow I/O devices. It is an efficient technique as compare asynchronous data transfer scheme because precious time of the microprocessor is not wasted in waiting while an I/O device is getting ready.

DMA data transfer scheme

In DMA (Direct Memory Access) method of data transfer, the CPU does not participate. Data are directly transferred from an I/O device to memory or vice-versa. The data transfer is controlled by the I/O device or a DMA controller. It is employed when large amount of data is to be transferred. If bulk data are transferred through the CPU, it takes more time and the process becomes slow. An I/O device, which wants to send data using DMA technique, sends the HOLD signal to the CPU. On receiving the HOLD signal from an I/O device, the CPU gives the control of buses as soon as the current bus cycle is completed. The CPU sends a HLDA (Hold Acknowledge) signal to the I/O device to indicate that it has received the HOLD request and it has released the buses. The I/O device takes over the control of buses and directly transfers data to the memory or reads data from memory. When data transfer is over, CPU regains the control over the buses. There are two methods of DMA data transfer schemes:

Burst mode of DMA data transfer: A scheme of DMA data transfer, in which I/O device withdraws the DMA request only after all data bytes have been transferred, is called burst mode of data transfer. A block of data is transferred. It is employed by magnetic disk drives. In case of magnetic disks, data transfer cannot be stopped or slowed without loss of data.

Cycle Stealing Technique: In this method, after transferring one byte or several bytes, the I/O device withdraws DMA request. It reduces interference in CPU's activities. The interference can be eliminated completely by designing an interfacing circuitry which can steal bus cycle for DMA data transfer only when CPU is not using the system bus.

Interrupts of Intel 8085

It has five interrupts namely TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR. TRAP has highest priority followed by RST 7.5, RST 6.5 and RST 5.5. The INTR has the lowest priority. Intel 8085 has two categories of interrupts: *maskable* and *non-maskable.* The interrupts which can be masked off are called *maskable interrupts.* Masking is done by software. RST 7.5, RST 6.5 and RST 5.5 are maskable. TRAP is non-maskable. It cannot be disabled.

When interrupts are to be used, they are enabled by software using the instruction EI (Enable Interrupt) in main program. EI sets the interrupt enable flip-flop to enable the interrupts. The instruction DI (Disable Interrupt) is used to disable interrupts. DI resets the interrupt enable flip-flop and disables all the interrupts except non-maskable interrupt TRAP. The system RESET also resets the Interrupt Enable flip-flop. When the I/O device becomes ready to transfer data, it sends a high signal to microprocessor through a special input line called an *interrupt line.* It interrupts the normal processing sequence of microprocessor. On receiving an interrupt microprocessor completes current instruction at hand, and then attends I/O device. It saves contents of PC on stack first, and then takes up a subroutine called ISS (Interrupt Service Routine). It executes ISS to transfer data from or to the I/O device.

Hardware and Software Interrupts: Interrupts caused by I/O devices are called *hardware interrupt.* The normal operation of microprocessor can also be interrupted by abnormal internal conditions or special instructions. Such an interrupt is called *software interrupt.* RST*n* instructions of 8085 are used for software interrupt. When RST*n* instruction is inserted in a program, the program is executed up to the point where RST*n* is inserted. The internal abnormal or unusual conditions which prevent the normal processing sequence of a microprocessor are also called as exceptions. For example, divide by zero will cause an exception. When several I/O devices are connected to INTR interrupt line, an external hardware is used to interface I/O devices. The external hardware circuit generates RST*n* codes to implement the multiple interrupt scheme.

Interfacing Devices and I/O Devices

To communicate with outside world, microcomputers use peripherals (I/O devices). Commonly used peripherals are: A/D converter, D/A converter, CRT, printers, hard disks, floppy disks, magnetic tapes, etc. Peripherals are connected to microcomputer through electronic circuits known as *interfacing circuits.* The interface associated with output device converts output of microcomputers into the desired peripheral format. Two types of interfacing devices are available. They are general-purpose interfacing devices and special-purpose interfacing devices.

Some of general-purpose interfacing devices:

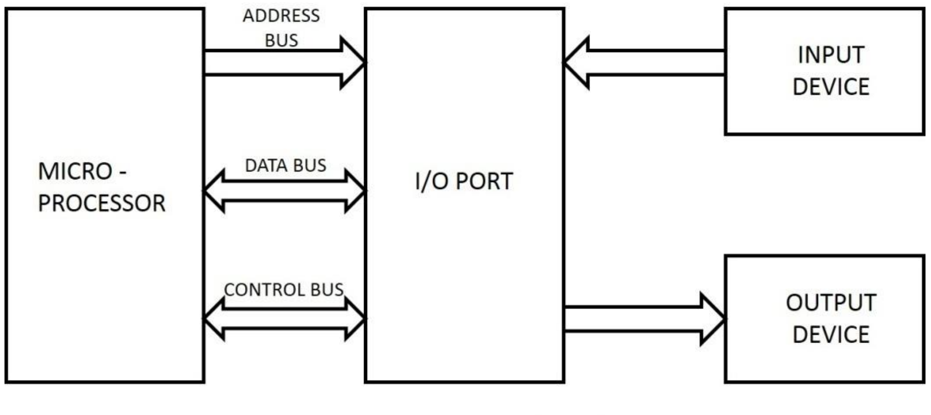
i. I/O port
ii. Programmable Peripheral Interface (PPI)
iii. DMA controller
iv. Interrupt controller
v. Communication Interface

Special purpose interfacing devices are designed to interface a particular type of I/O device to microprocessor. Examples of such devices are:

i. CRT controller
ii. Floppy Disk Controller
iii. Key Board and Display Interface.

I/O Ports

An input device is connected to microprocessor through an input port. An input port is a place for unloading data. An input device unloads data into port. The microprocessor reads data from input port. Similarly, an output device is connected to microprocessor through an output port. As the output port is connected to output device, data are transferred to output device. An I/O port may be programmable or non-programmable. A non-programmable port behaves as an input port if it has been designed and connected in input mode. A port connected in output mode acts as an output port. But, a programmable I/O port can be programmed to act either as an input port or output port.



Interfacing of I/O Devices through I/O Port

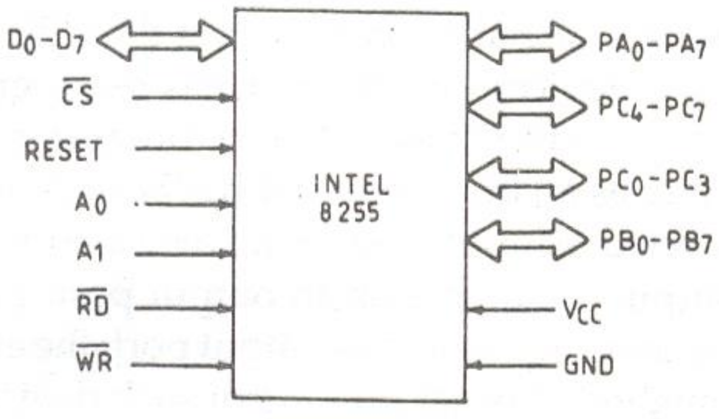
Programmable Peripheral Interface (PPI)

A programmable peripheral interface is a multiport device. The ports may be programmed in a variety of ways as required by the programmer. The device is very useful for interfacing peripheral devices. The term PIA called as peripheral Interface Adapter is also used by some manufacturer.

Intel 8255:

The Intel 8255 is a PPI. Its main function is to interface peripheral devices to microcomputer. It has three 8-bit ports, Port A, B and C. The Port C has been further divided into two of 4-bit ports, Port C upper and Port C lower. Thus, a total of 4 ports are available, two 8-bit ports and two 4-bit ports. Each port can be programmed either as an input port or an output port. These four ports are further categorized into two Groups: Group-A and Group-B. Group-A consists of Port A and Port C_{upper}, and Group-B consists of Port-B and Port C_{lower}.

Pin Configuration of Intel 8255



Schematic Diagram of Intel 8255 A.

The pins for various ports are as follows:

$PA_0 – PA_7$	8 pins of Port A
$PB_0 – PB_7$	8 pins of Port B
$PC_0 – PC_3$	4 pins of Port C_{lower}
$PC_4 – PC_7$	4 pins of Port C_{upper}

The important control signals are as follows:

$\overline{CS}$: It is a chip select signal. The LOW status of this signal enables communication between the CPU and 8255.

$\overline{RD}$ (Read) : When it goes low the 8255 sends out data or status information to CPU on data bus. In other words, it allows CPU to read data from input port of 8255.

$\overline{WR}$ (Write) : When it goes low the CPU writes data or control word into 8255. The CPU writes data into output port of 8255 and control word into control word register.

A_0 and A_1 : The selection of input port and control word register is done using this in conjunction with $\overline{\mathrm{RD}}$ and $\overline{\mathrm{WR}}$. They are normally connected to LSBs of address bus.

A_1	A_0	Selected Port
0	0	Port A
0	1	Port B
1	0	Port C
1	1	Control Word Register

Operating modes of 8255:

The 8255 has the following three modes of operation:

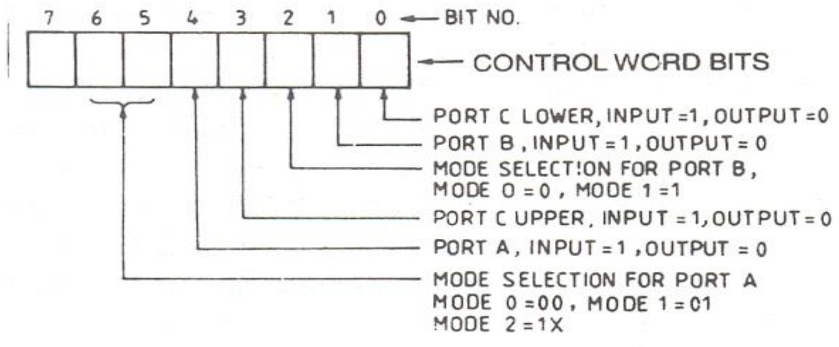
Mode-0 (Simple I/O): In this mode, a port can be operated as simple I/O port. Each of four ports of 8255 can be programmed to be either an input or output port. In this mode, the data transfer takes without any control signals.

Mode-1 (Strobed I/O): Mode-1 is a strobed input/output mode of operation. Port A and Port B both are designed to operate in this mode of operation. When Port A and Port B are programmed in Mode1, Port C is used for generating control signals. PC0, PC1 and PC2 are used for the control of the Port B, which can be used either as input or output port. If the Port A is operated as in input port, PC3, PC4 and PC5 are output. When Port A is operated as an output port, pins PC3, PC6 and PC7 are used for its control. The pins PC4 and PC5 can be used as either input or output.

Mode-2 (Bidirectional Port): Mode-2 is strobed bidirectional mode of operation. In this mode, Port A can be programmed to operate as a bidirectional port. The Mode 2 operation is only for Port A. When Port A is pro0grammed in Mode 2, the Port B can be used in either Mode-1 or Mode-0.

Control Word Format of 8255

According to the requirement a port an programmed to act either as n input port or an output port. For programming the ports of 8255, a control word is formed. The bits of the control word are shown in the following diagram:


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

Control Word Bits for Intel 8255

The control word is written onto the control word register which is within the 8255. The control word bit corresponding to a particular port is set to either 1 or 0 depending upon the definition of the port. If a particular port to be made an input port, the bit corresponding to that port is set to 1. For making a port an output port, the corresponding bit for the port is set to 0. Bit 7 is set to 1 if Port A, B and C are defined as I/O port. It is set to 0 if the individual pins of the Port C are to be set or reset.

Delay Subroutines

A delay subroutine is used to provide the desired delay in industrial control before issuing control signal by microcomputer. To generate delay a few registers of microprocessor are loaded with desired numbers and then decremented to zero. The delay time depends on numbers loaded in registers.

Delay Subroutine using One Register

```
                MVI B, 10
LOOP:           DCR B
                JNZ LOOP
                RET
```

The above program explains the delay subroutine using one register. To generate very small delay only one register can be used. In the above program register B is loaded by 10H (16 decimal). Then the register B is decremented and program moves in a loop till the content of register B becomes zero. After this the program returns to the main program.

Calculation of Delay Time: To calculate the delay time, three parameters are required:

- Number of times each instruction is executed in the program.
- The number of T-states (Clock cycles per second) for each instruction.
- The time for one T-state in 8085 microprocessor.

The delay time of the delay subroutine using one register is calculated as follows.

Delay subroutine	No. of T-states	No. of times each instruction is executed	Total T-states	
MVI B, 10	7	1	7	(7 x 1)
LOOP: DCR B	4	16	64	(4 x 16)
JNZ LOOP	7/10	16	157	(10 x 15 + 7 x 1)
RET	10	1	10	(10 x 1)
		TOTAL	238	

Total T-states of the delay subroutine	= 238
Time for one T-state for 8085	= 320 ns (320×10^{-9} Secs)
Delay Time for 238 T-states	= $238 \times 320 \times 10^{-9}$ Secs
	= 0.07616 millisecond

Delay subroutine using Register pair

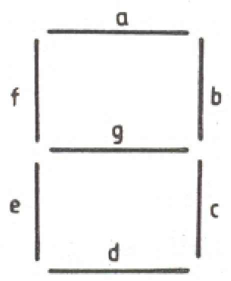
```
            LXI D, FFFF
LOOP:       DCX D
            MOV A, D
            ORA E
            JNZ LOOP
            RET
```

Delay Subroutine using Three Registers

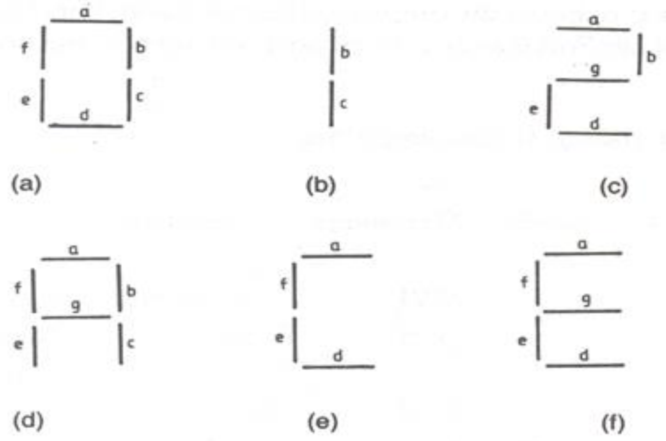
```
            MVI B, FF
LOOP1:      MVI C, FF
LOOP2:      MVI D, FF
LOOP3:      DCR D
            JNZ LOOP3
            DCR C
            JNZ LOOP2
            DCR B
            JNZ LOOP1
            RET
```

Seven Segment Displays

The seven-segment LED display is a multiple display. It can display all decimal digits and some letters. It is very popular among multiple displays as it has smallest number of separately controlled light emitting diodes (LED) as shown in the figure below:

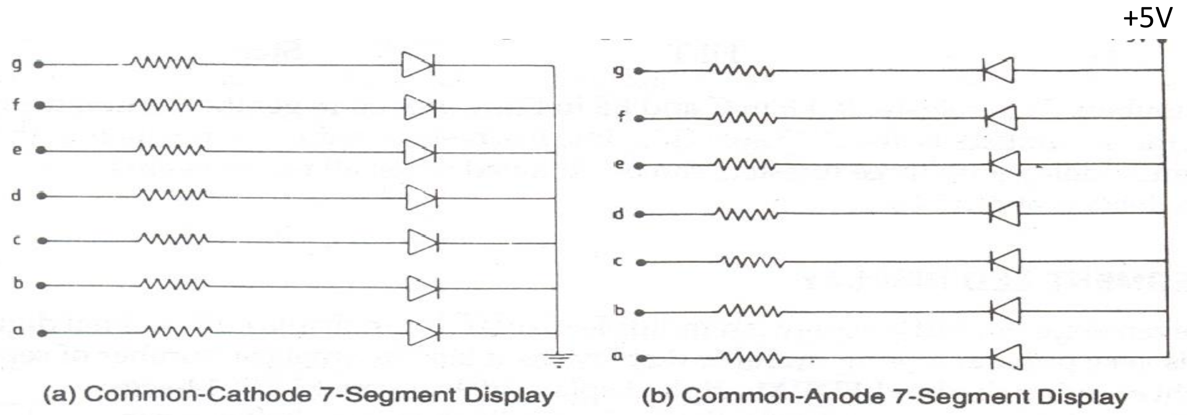


In a seven-segment display, there are 7 LEDs. Each LED can be controlled separately. To display a digit or letter the desired segments are made ON as shown in the figure below:



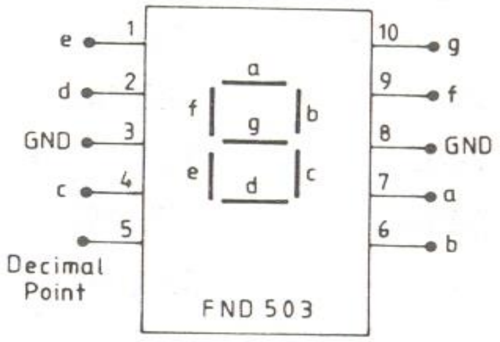
There are two types of seven-segment displays. They are common-cathode type seven-segment displays and common-anode type seven-segment displays.

In a common cathode type display all the 7 cathodes of LEDS are tied together to the ground as in the diagram.   When a +5V is applied to any segment, the corresponding diode emits light. Thus, applying logic 1 to the desired segments, the desired letter or decimal number can be displayed. In a common anode type display all the 7 anodes are tied together and connected to + 5 V supply as in the diagram.



(a) Common-Cathode 7-Segment Display (b) Common-Anode 7-Segment Display

FND 500 and FND 503 are common-cathode 7-segment displays. The following diagram shows the pin configuration of FND 503. FND 507 and FND 510 are common-anode 7-segment displays.



Display of Decimal Numbers 0 to 9

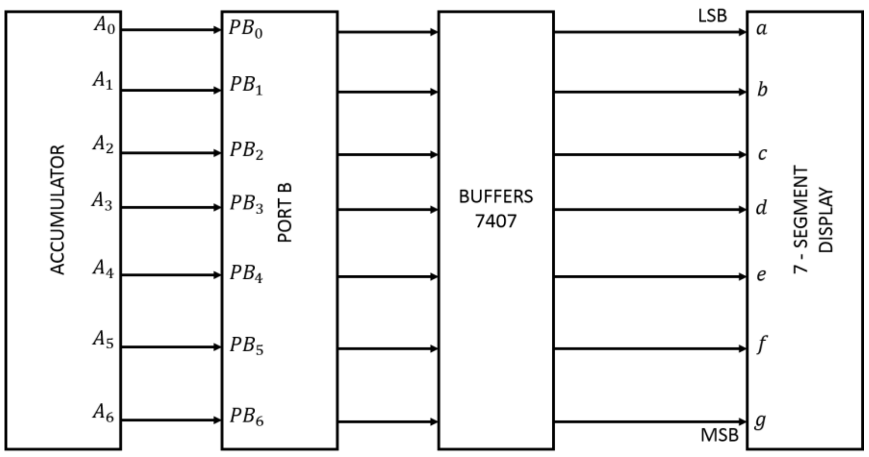
A program has been developed to display the decimal numbers 0 to 9, one by one. First of all, decimal number 0 is displayed for some time, then 1 and so on. After displaying all numbers, program will repeat the process of displaying the numbers again and again. A delay subroutine has been included in program. The time for displaying the numbers can be adjusted by adjusting data for delay subroutine. The program is as follows:

```
                MVI A, 98
                OUT CW#
START :         LXI H, 2500
                MVI C, 0A
NEXT :          MOV A, M
                OUT PB#
                INX H
                CALL DELAY
                DCR C
                JNZ NEXT
                JMP START

                MVI B, FF
LOOP1:          MVI C, FF
LOOP2:          MVI D, FF
LOOP3:          DCR D
                JNZ LOOP3
                DCR C
                JNZ LOOP2
                DCR B
                JNZ LOOP1
                RET
```

Interfacing of Seven-Segment Displays

A seven-segment display can be interfaced to the microprocessor as shown in the figure below.



Interfacing of 7-segment Display

The microprocessor is connected to the Port B of the 8255 Programmable Peripheral Interface. The Port B of 8255 is connected to the seven-segment display through the Hex Buffers 7407.

Suppose we want to display the decimal digit 1. To display a 1, segments b and c are switched on. Segments *a, b, c,* and *d* are connected through pins PB_0, PB_1, PB_2, and PB_3 respectively. To display a 1, segment *b* and *c* receive logic '1' and segments *a* and *d* receive logic '0' from the microprocessor through Port B.