Microprocessor and Microcontroller (EE 601)

Online Courseware (OCW)

B.TECH (3rd YEAR – 6th SEM)

Prepared by: Dr. Debasree saha

Department of Electrical Engineering

Guru Nanak Institute of Technology
(Affiliated to MAKUT, West Bengal , Approved by AICTE - Accredited by NAAC – 'A+' Grade)
157/F Nilgunj road, Panihati, Kolkata-700114, West Bengal

Microprocessor and Microcontroller
Code: EE601

Contacts: 3L **Total Contact Hours: 36**

Credits: 3 **(50 minutes/lecture)**

Pre requisites:

Knowledge in Digital Electronics.

Course Outcomes (COs):

On completion of the course students will be

CO1. Able to correlate the architecture, instructions, timing diagrams, addressing modes, memory interfacing, interrupts, data communication of 8085.

CO2. Able to interpret the 8086 microprocessor-Architecture, Pin details, memory segmentation, addressing modes, basic instructions, interrupts.

CO3. Recognize 8051 micro controller hardware, input/output pins, ports, external memory, counters and timers, instruction set, addressing modes, serial data i/o, interrupts.

CO4. Apply instructions for assembly language programs of 8085, 8086 and 8051.

CO5. Design peripheral interfacing model using IC 8255, 8253, 8251 with IC 8085, 8086 and 8051.

Course Content

Module 1: 8085 Microprocessor **[6]**

Introduction to Microcomputer based system, Evolution of Microprocessor and microcontrollers and their advantages and disadvantages, Architecture of 8085 Microprocessor, Address / Data Bus multiplexing and demultiplexing, Status and Control signal generation, Instruction set of 8085 Microprocessor, Classification of instructions, addressing modes, timing diagram of the instructions, Memory interfacing , IO interfacing, ADC / DAC interfacing, Stack and Subroutine, Delay Calculation, Interrupts of 8085 processor, classification of interrupts, Serial and parallel data transfer – Basic concept of serial I/O, DMA, Asynchronous and synchronous serial transmission using SID and SOD pins of 8085.

Module 2: Assembly language programming with 8085 **[2]**

Addition, Subtraction, Multiplication, Block Transfer, ascending order, descending order, Finding largest & smallest number, Look-up table etc. Programming using interrupts (programming using INTR is not required).

Module 3: 8086 Microprocessor **[8]**

8086 Architecture, Pin details, memory segmentation, addressing modes, Familiarization of basic Instructions, Interrupts & Direct Memory Access, Memory interfacing, ADC / DAC interfacing.

Module 4: Assembly language programming with 8086 **[3]**

Addition, Subtraction, Multiplication, Block, Transfer, ascending order, descending order, Finding largest & smallest number etc.

Module 5: 8051 Microcontroller **[7]**

8051 architecture, hardware, input/output pins, ports, internal and external memory, counters and timers, instruction set, addressing modes, serial data i/o, interrupts, Memory interfacing, ADC / DAC interfacing.

Module 6: Assembly language Programming using 8051 **[4]**

Moving data: External data moves, code memory read only data moves, PUSH and POP opcodes, data exchanges; Logical operations: Byte-level, bit-level, rotate and swap operations; Arithmetic operations: Flags, incrementing and decrementing, addition, subtraction, multiplication and division, decimal arithmetic; Jump and call instructions: Jump and call program range, jumps, calls and subroutines, interrupts and returns.

Module 7: Support IC chips **[6]**

8255, 8253 and 8251: Block Diagram, Pin Details, Modes of operation, control word(s) format. Interfacing of support IC chips with 8085, 8086 and 8051.

Text Books:

1. Microprocessor architecture, programming and application with 8085 – R. Gaonkar, Penram International

2. The 8051 microcontroller - K. Ayala, Thomson

3. Microprocessors & interfacing – D. V. Hall, Tata McGraw-hill

4. Ray & Bhurchandi, Advanced Microprocessors & Peripherals, TMH

5. The 8051 microcontroller and Embedded systems - Mazidi, Mazidi and McKinley, Pearson

6. An Introduction to Microprocessor and Applications –Krishna Kant,Macmillan

Reference Books:

1. Microprocessors and microcontrollers - N. Senthil Kumar, M. Saravanan and Jeevananthan, Oxford university press

2. 8086 Microprocessor –K Ayala, Cengage learning

3. The 8051 microcontrollers – Uma Rao and Andhe Pallavi, Pearson

JIS GROUP
Educational Initiatives

GNIT

CO-PO Mapping:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3	3	3	1	3	1	1	1	1	1	1	3
CO2	3	3	3	1	3	1	1	1	1	1	1	3
CO3	3	3	3	3	3	2	2	1	1	1	2	3
CO4	3	3	3	3	3	2	2	1	1	2	1	3
CO5	3	3	3	3	3	1	2	1	2	2	2	3

Lecture-I

Module 1: 8085 Microprocessor

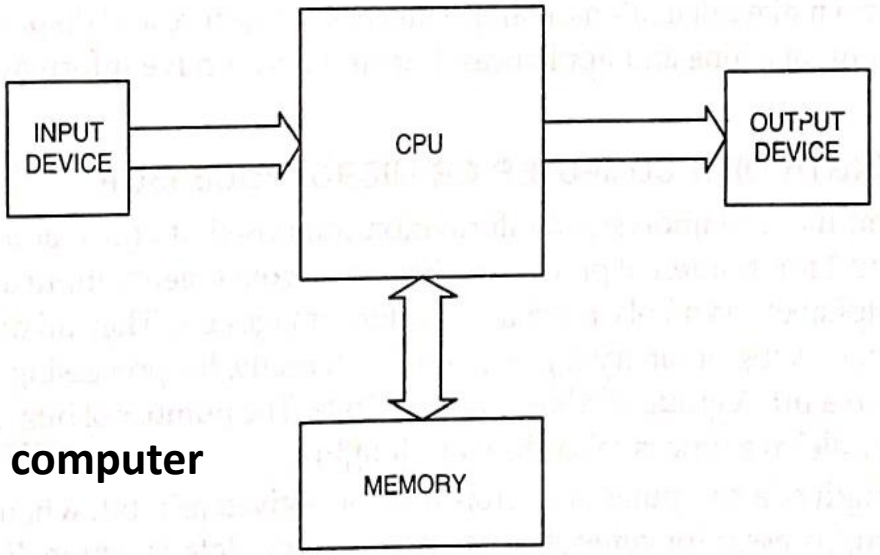
Content:

- Introduction to Microcomputer based system
- Evolution of Microprocessor and microcontrollers
- Advantages and disadvantages of Microprocessor and Microcontrollers

Introduction to Microcomputer based system

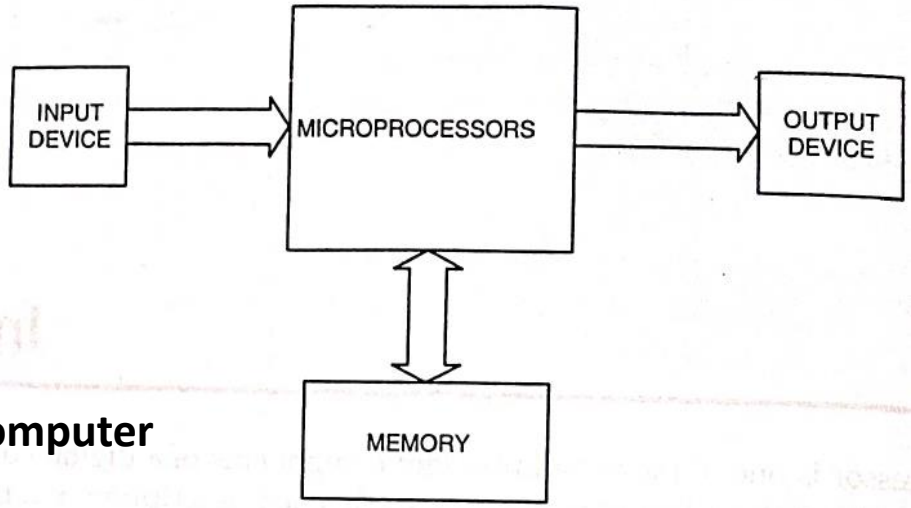
The microprocessor is one of the most important components of a digital computer. It acts as the brain of a computer system. Computer are of two types: digital computer and analog computers. A digital computer makes processing of numbers. An analog computer process analog signals. An analog signal is a continuous signal. Now-a-days computers which are commonly used are digital computers. Analog computers have specific applications. They are used for some specific scientific and engineering purposes. Earlier, they were used to study, analyze and simulate scientific and engineering systems. Today these works are done by digital computers.

computers are the most powerful tool man has ever created. A digital computer is a programmable machine. Its main components are: CPU, memory, input device and output device.



Schematic diagram of a digital computer

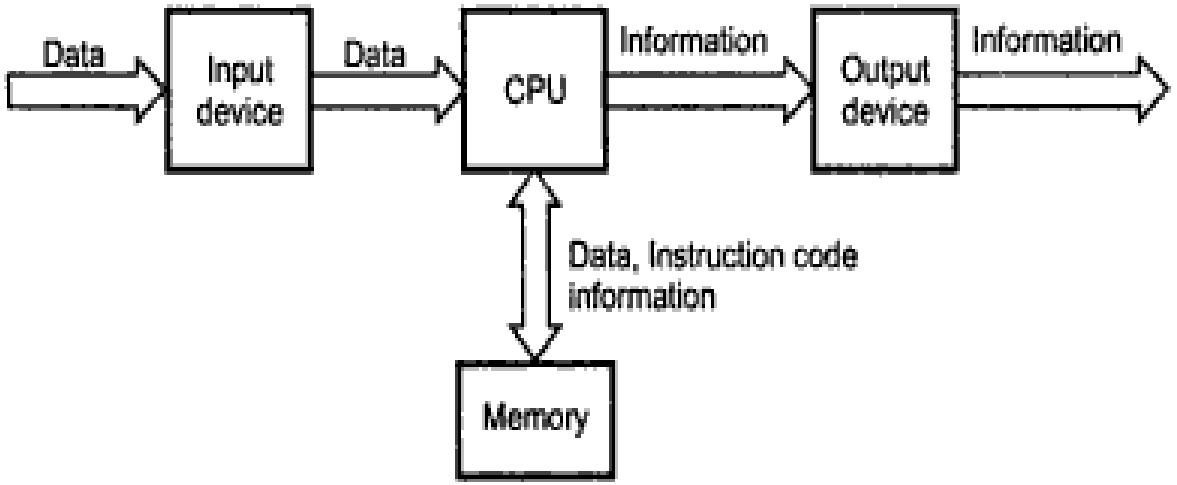
The CPU executes instructions. The input device is used to feed programs and data to the computer. The memory is a storage device. It stores programs, data and result. The output device displays or prints programs, data and/or results according to the instruction given to the computer. The central processing unit built on a single IC is called **microprocessor**. A digital computer in which one microprocessor has been provided to act as a CPU, is called **microcomputer**. A desktop computer and portable (or mobile) computers like laptop, notebook, palmtop, etc. contain one microprocessor to act as a CPU and hence they come under the category of microcomputer. A schematic diagram of a microcomputer is shown in Fig. The CPU of a large powerful digital computer contains more than one microprocessor. High-end powerful servers, supercomputers, etc. contain more than one microprocessor to act as CPU. These microprocessors placed in a CPU of a large powerful computer operate in parallel. A computer whose CPU contains more than one microprocessor is called a multiprocessor computer system.



Schematic diagram of a microcomputer

- A digital computer was developed for complex scientific and engineering calculations and it was a programmable machine. Hence, a computer was defined as a “programmable computing machine”. Today, besides computation work computers are used for a number of noncomputational work such as automatic control of industrial equipment, to control process, to measure physical and electrical quantities, to process text, graphics and image; to store information, to display information, to transmit information from one place to another, to receive information and so on. In the light of such developments, a computer now can be defined as programmable machine which can make calculations, manipulate, measure, store, display information; control process, equipment, machine and appliances, transmit and receive information and so on.

Introduction to Microcomputer based system



A microprocessor is a multipurpose, programmable, clock-driven, register-based electronic device that reads binary instructions from a storage device called *memory*, accepts binary data as input and processes data according to those instructions, and provides results as output. At a very elementary level, we can draw an analogy between microprocessor operations and the functions of a human brain that process information according to instructions (understanding) stored in its memory. The brain gets input from eyes and ears and sends processed information to output "devices" such as the face with its capacity to register expression, the hands or feet. However, there is no comparison between the complexity of a human brain and its memory and the relative simplicity of a microprocessor and its memory.

Evolution of Microprocessor

- First generation (1889-1954) -vacuum tube

The Electrical age:

-Hollerith machine(1889)
-ENIAC(Electronics Numerical Integrator & Calculator)
-first general-purpose, programmable electronic computer
-17,000 vacuum tube, 500 miles of wire, 6000 switches
-life of vacuum tube(3000 hours) : maintenance problem

IBM 650, 1954

❑ Second generation (1954-1959) -transistor

Bipolar Transistor : 1948, William Shockley, John Bardeen, Walter H. Brattain at Bell labs(1956, Novel physics award)

Mainframe : describe CPU portion of computer

Mainframe computer : designed to handle large volumes of data while serving hundreds of users simultaneously

Built on circuit boards mounted into rack panels(frame)

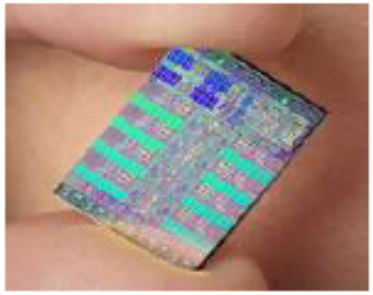
❑ Third generation (1959-1971) - IC

- **Integrated Circuit :** 1958,Jack Kilby (Texas Instruments) & Dr. Robert Noyce (Fairchild Semiconductor).
- **IBM** : 32-bit 360 series(1964)
- **INTEL**(Integrated Electronics) : 1968

PDP-8, Digital Equipment Corporation

— Thanks to the use of ICs, the DEC PDP-8 is the least expensive general purpose small computer in 1960s

❑ Fourth generation (1971-present) - microprocessor



4- Bit processor-

MCS-4 Family - 4004 (used in calculator),
4001, 4002,4003,4008,4009

MCS-40 Family- 4040, 4101,4207,4209 etc.

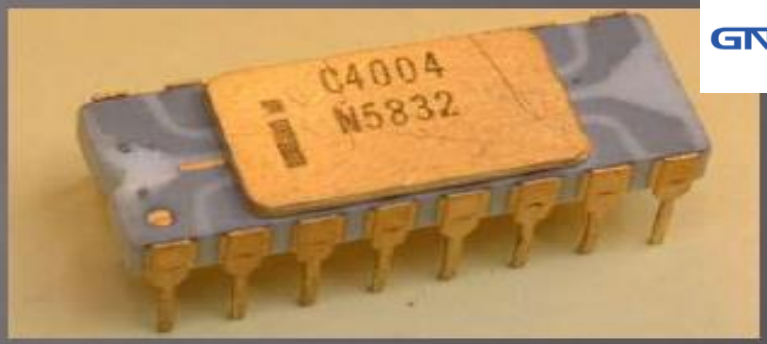
8- Bit processor- 8008,8080,8085 etc

16- Bit processor- 8086,8088,80186,80286 etc.

32- Bit processor- 80386DX, 80386SX, 80376,pentium
Pentium pro, II, etc.

64- Bit processor- Intel Pentium, core i3, core
i5,core i7,etc

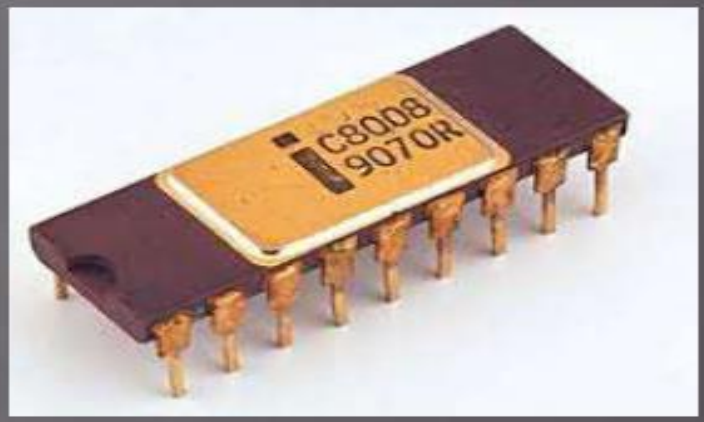
INTEL 4004



Year of Introduction – 1972

- 4-bit microprocessor.
- 4KB main memory
- 45 instructions
- P-MOS technology
- was first programmable device which was used in calculators.

INTEL 8008



Year of Introduction :- 1972

- 8 bit version of 4004
- 16 KB main memory
- 48 instructions
- P-MOS technology
- Slow

INTEL 8080

Year of Introduction:- 1973

- *8 bit microprocessor*
- *64 KB main memory*
- *2 microseconds clock cycle time*
- *500,000 instruction/sec*
- *10X faster than 8008*
- *N-MOS technology*
- *Drawbacks was that it needs three power supplies.*
- *Small companies (microcomputers) were designed in mid 1970s using 8080 as CPU.*

INTEL 8085

Year of Introduction:- 1975

- 8 bit microprocessor- upgrading version of 8080.
- 64 KB main memory.
- 1.3 microseconds clock cycle time.
- 246 instructions.
- Intel sold 100 millions copies of this microprocessors.
- Uses only one +5v supply.

INTEL 8086/8088

Year of introduction:- 1978 for 8086
1979 for 8088

- *16 bit microprocessors*
- *Data bus width of 8086 is 16 bit and 8 bit for 8088*
- *1MB main memory.*
- *400 nanoseconds clock cycle time*
- *6 byte instructions cache for 8086 and 4 byte for 8088*
- *Other improvements included more registers and additional instructions*
- *In 1981 IBM decided to use 8088 in its personal computers.*

INTEL 80186

Year of introduction:- 1982

- *16 bit microprocessor- upgrade version of 8086.*
- *1MB main memory.*
- *Contained special hardware like programmable counters, interrupt controller etc.*
- *Never used in the PC*
- *But was ideal for systems for systems that required a minimum of hardware.*

INTEL 80286

Year of Introduction:- 1983

- *16 bit high performance microprocessors with memory management & protection.*
- *16 MB main memory*
- *Few additional instructions to handle extra 15 MB*
- *Instruction execution time is as little as 250 ns*
- *Concentrates on the features needed to implement MULTITASKING*

INTEL 80386

Year of Introduction:- 1986

- Intel first practical 32 bit microprocessor
- 4GB main memory
- Improvements include page handling in virtual environment
- Includes hardware circuitry for memory management and memory assignment
- Memory paging and enhanced I/O permissions

INTEL 80486

Year of Introduction:- 1989

- 32 bit high performance microprocessor
- 4GB main memory
- Incorporates 80387 like floating point coprocessors
- 8K bytes cache on one package
- About half of the instructions executed in 1 clock instead of 2 on the 80386

INTEL PENTIUM

Year of Introduction:- 1993

- *32 bit microprocessor, 64 bit data bus and 32 bit address bus*
- *4GB main memory*
- *Double clocked 120 and 133MHz*
- *Dual integer processor*
- *Fastest version is the 233MHz*
- *16 KB L1 cache (split instruction and data: 8KB each)*

INTEL PENTIUM PRO

Year of Introduction:- 1995

- 32 bit microprocessor, formerly code-named P6
- 64GB main memory, 64 bit data bus and 36 bit address bus
- 16 KB L1 cache (split instruction and data: 8KB each), 256 KB L2 cache
- Uses three execution engines
- Intel launched this processor for server market

INTEL PENTIUM II

Year of Introduction:- 1997

- *32 bit microprocessor, 64 bit data bus and 36 bit address bus*
- *64 GB main memory*
- *32 KB split instruction /data L1 caches (16 KB each)*
- *Module integrated 512KB L2 cache (133MHz)*
- *A version of P2 called Xeon; specifically designed for high-end applications*

INTEL PENTIUM III

Year of Introduction:- 1999

- 32 bit microprocessor, 64 bit data bus and 36 bit address bus
- 64 GB main memory
- Dual Independent Bus (simultaneous L2 and System Memory Access)
- On chip 256 KB L2 cache
- P3 was available in clock Frequency of up to 1 GHz

INTEL PENTIUM IV

Year of Introduction:- 2002

- *32 bit microprocessor, 64 bit data bus and 36 bit address bus*
- *64 GB main memory*
- *1.4 to 1.9 GHz and the latest at 3.2GHz and 3.46Ghz (Hyper-Threading)*
- *1MB/512KB/256KB L2 cache*
- *Specialized for streaming video, game and DVD applications*

NTEL DUAL CORE /CORE 2 DUO

Year of Introduction:- Both in 2006

- *32 bit/64 bit microprocessor*
- *It has two cores.*
- *It support SMT technology (SMT: Simultaneously Multi Threading)*
- *E.g. : Adobe Photoshop supports SMT*

INTEL CORE I3

Year of Introduction:- 2009

- *32 bit/64 bit microprocessor*
- *3.2GHz dual core chip , and it is quite a big improvement over the Core 2 Duo.*
- *It's a good 700MHz faster, much faster bus with significantly faster RAM*
- *Uses less heat and energy than earlier processors*

JIS GROUP
Educational Initiatives

GNIT

INTEL CORE I5

Year of Introduction:- 2010

- *32 bit/64 bit microprocessor*
- *It provides the opportunity to the users to use the system with multitasking*
- *Turbo boost Technology if i5 processors is the key beneficial features of i5 processors that allows the user to do their regular and important working with the help of heavy applications.*
- *High speed performance rate so they are able to perform at the maximum CPU rate of 3.6GHz .*

JIS GROUP
Educational Initiatives

GNIT

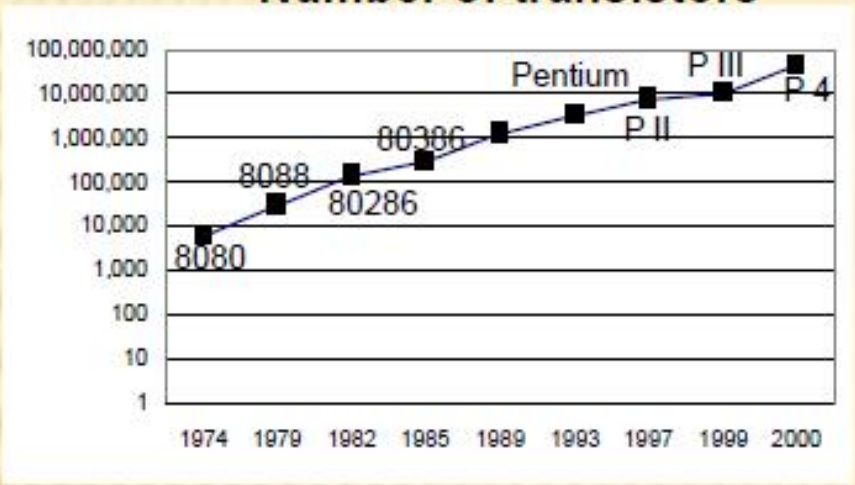
INTEL CORE i7

Year of Introduction:- 2008

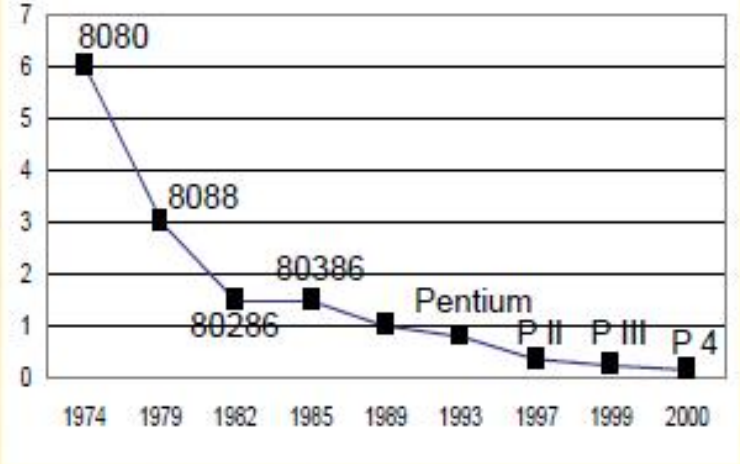
- 32 bit/64 bit microprocessor
- Intel i7 were designed to meet the challenges of the intelligent and the faster working performance of the computer system.
- i7 are so advanced to deals with the integrated memory of the system and have ability to increase the memory up to 1066 M bits and provide the working speed of 25.6 GB/sec

Evolution of Intel Microprocessors

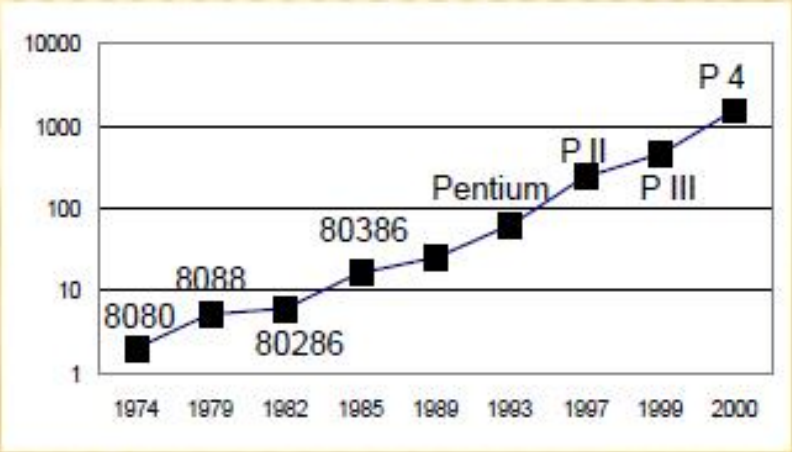
Number of transistors



Minimum transistor sizes (µm)



Clock frequencies (MHz)



Advantages and disadvantages of Microprocessor and Microcontrollers

- **There are some advantages of microprocessor are given below,**
- Microprocessor are general purpose electronic processing devices which can be programmed to execute a number of tasks.
- Speed of Microprocessor is measured in **hertz**. For instance, a microprocessor with 3 GHz, shortly GHz is capable of performing 3 billion tasks per second.
- Microprocessor is that which can quickly move data between the various memory locations.
- **There are some disadvantages of microprocessor are given below,**
- The microprocessor has a limitation on the size of data.
- Most of the microprocessor does not support floating point operations.
- The main disadvantage is it's over heating physically.
- It should not contact with the other external devices.
- The microprocessor does not have any internal peripheral like ROM, RAM and other I/O devices.

JIS GROUP
Educational Initiatives

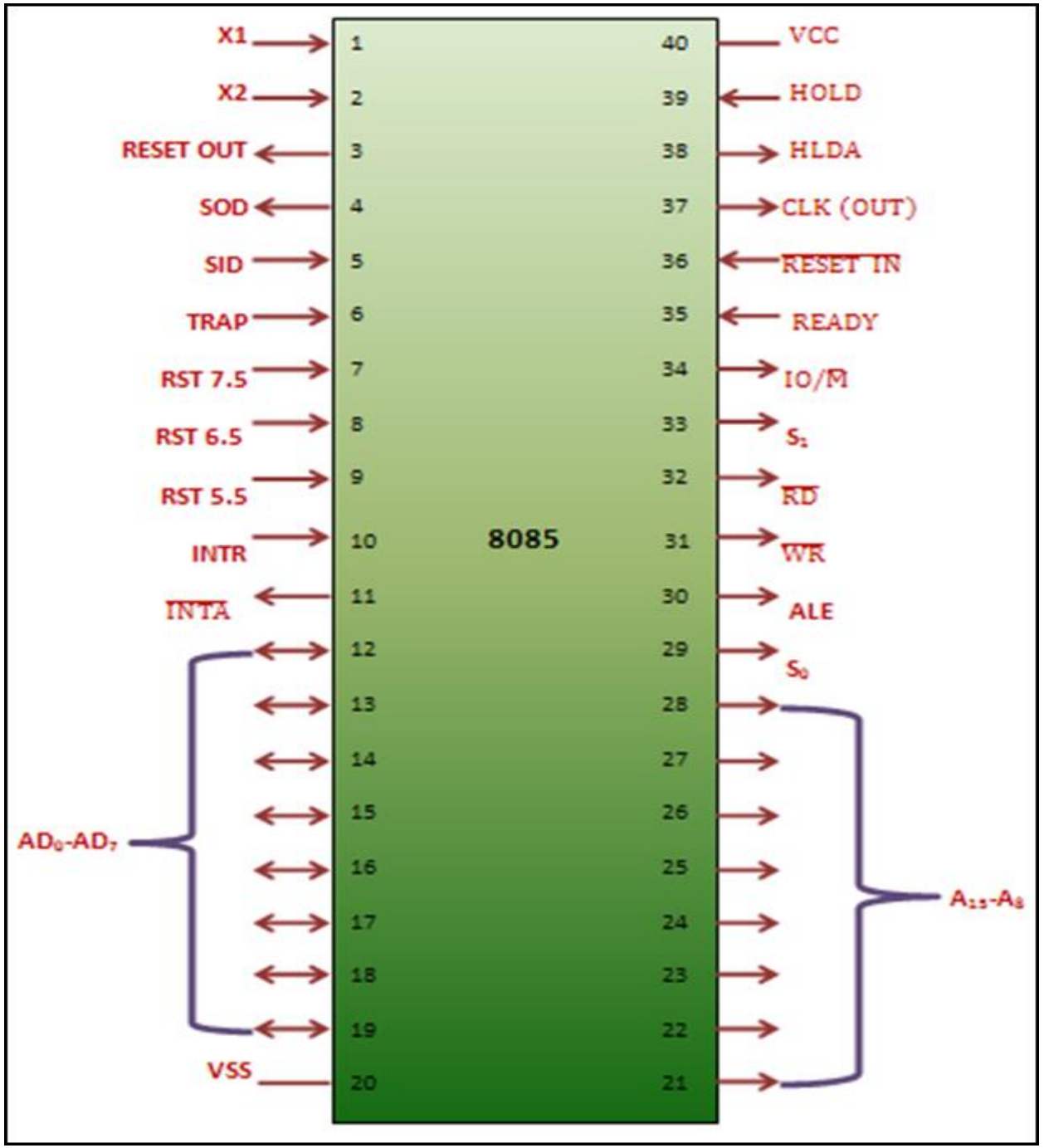
GNIT

Lecture-III

Module 1: 8085 Microprocessor

Content: Pin diagram of 8085 Microprocessor

Pin Diagram of 8085



Functions of various Pins of 8085

- **A8-A15 Higher Order Address bus:**
  These are o/p tri-state (a state of high impedance) signals used as higher order 8 bits of 16 bit address.
  These signals are unidirectional and are given from 8085 to select memory or I/O devices.
- **AD0-AD7 Multiplexed Address/Data bus:**
  These are I/O tri-state signals, having 2 sets of signals. They are address and data.
  The lower 8 bit of 16 bit address is multiplexed/time shared with data bus.
- **Address latch Enable(ALE):**
  It is an output signal used to give information of AD0-AD7 contents.
  It is a positive going pulse generated when a new operation is started by microprocessor.
  When pulse goes high it indicates that AD0-AD7 lines are address.
  When it is low it indicates that the contents are data.
- **IO/M(bar):**
  This is an output status signal used to give info of operation to be performed with memory or I/O devices.
  When = 0, the microprocessor is performing memory related operation.
  When = 1, the microprocessor is performing I/O device related operation.
  This signal separates memory and I/O devices.

- **Status signals(S0 and S1):**
  These are output status signals used to give information of operation performed by microprocessor.
  The S0 and S1 lines specify 4 different conditions of 8085 machine cycles.

Operation	S0	S1
Opcode fetch(instruction read from memory)	1	1
Read(data read from memory)	0	1
Write	1	0
Halt	0	0

- **Read:** This is an active low output control signal used to read data from memory or an I/O device.
- **Write:** This is an active low output signal used to write data to memory or an I/O device.
- **Ready:** This is an active high input control signal. It is used by microprocessor to detect whether a peripheral has completed (or is Ready for) the data transfer or not. The main function of this pin is to synchronize slower peripheral to faster microprocessor.
  If ready pin is high the microprocessor will complete the operation and proceeds for the next operation. If ready pin is low the microprocessor will wait until it goes high.

- **Trap:**
  This is an active high, level and edge triggered, non-maskable higher priority interrupt.
  When TRAP is active, the program counter of µp jumps automatically at address 0024.

  RST 7.5,RST 6.5 and RST 5.5:
  These are active high, edge (RST 7.5) or level (RST 6.5 and RST 5.5) triggered maskable interrupts.
  The priorities of these are TRAP, RST 7.5, RST 6.5, and RST 5.5.
  When RST 7.5, RST 6.5 and RST 5.5 are active, the program counter jumps automatically at address 003C, 0034, 002C respectively.

  INTR and INTA(Bar):
  INTR is an active high, level triggered general purpose interrupt.
  When INTR is active µp generates an interrupt acknowledge signal.
  If INTR is active, the Program Counter (PC) will be restricted from incrementing and an will be issued.
  During This cycle a RESTART or CALL instruction can be inserted to jump to the interrupt Service routine.
  The INTR is enabled and disabled by software. It is disabled by Reset and immediately after an interrupt is accepted.

- **HOLD:**
  HOLD indicates that another Master is requesting the use of the Address and Data Buses.
  The CPU, upon receiving the Hold request, will withdraw the use of buses as soon as the completion of the current machine cycle. Internal processing can continue.
  The processor can regain the buses only after the Hold is removed.
  When the Hold is acknowledged, the Address, Data, RD, WR, and IO/M lines are tri-stated.
  HLDA:
  HOLD ACKNOWLEDGE indicates that the CPU has received the Hold request and that it will withdraw the buses in the next clock cycle.
  HLDA goes low after the Hold Request is removed.
  The CPU takes the buses one half clock cycles after HLDA goes Low.

  Reset IN(Bar):
  Reset sets the Program Counter to zero and resets the Interrupt Enable and HLDA Flip-flops and makes address, data and control lines tri-stated.
  The CPU is held in the reset condition as long as Reset is applied.
  After reset status internal register and flag are unpredictable.
  After reset microprocessor starts executing from instruction from 0000H onwards.

RESET OUT:

This is an active high output signal used to indicate CPU is being reset and can be used as a system RESET. The signal is synchronized to the processor clock.

This signal is also used to reset the peripherals once the µP is reset.

It is an acknowledgement signal to RESET IN (bar).

Serial input data(SID):

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

Serial output data(SOD):

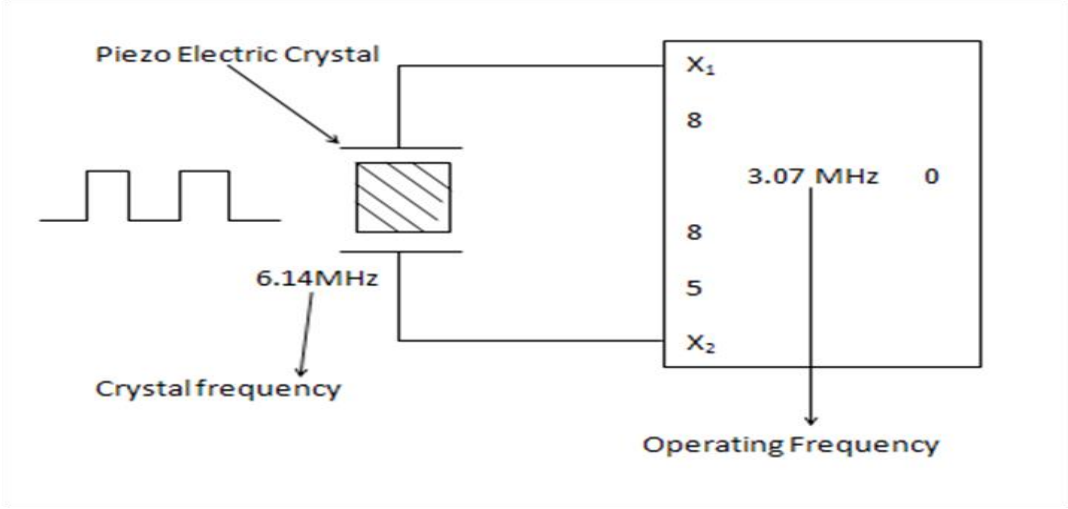
This is an active high Serial output data line.

The output SOD is set or reset as specified by the SIM instruction.

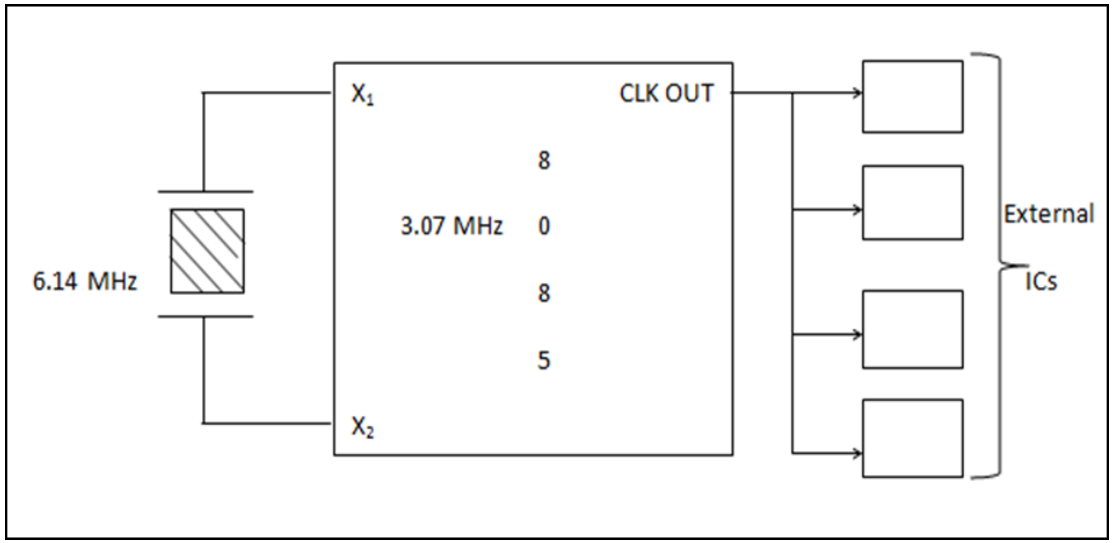
X1,X2:

Crystal or R/C network connections to set the internal clock generator X1 can also be an external clock input instead of a crystal.

The input frequency is divided by 2 to give the internal operating frequency as shown in fig.

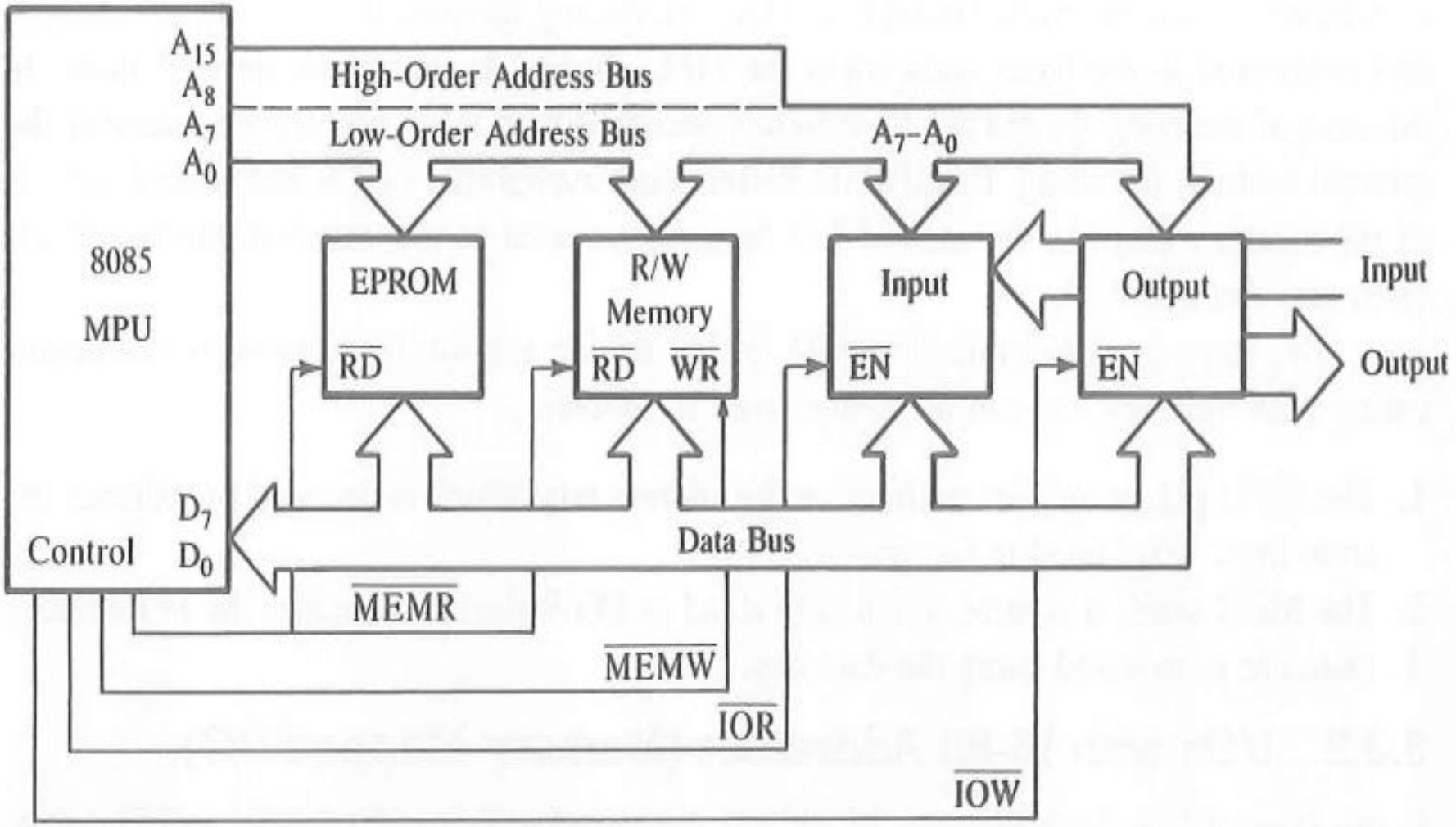


- **CLK OUT:**
  Clock Output for use as a system clock when a crystal or R/ C network is used as an Input to the CPU.
  Clock input to all other peripherals is provided through CLK OUT pin.
  The period of CLK is twice the X1, X2 input period.



- **VCC and VSS:**
  +5 volt supply and Ground Reference.

A Microcomputer system



Lecture-IV

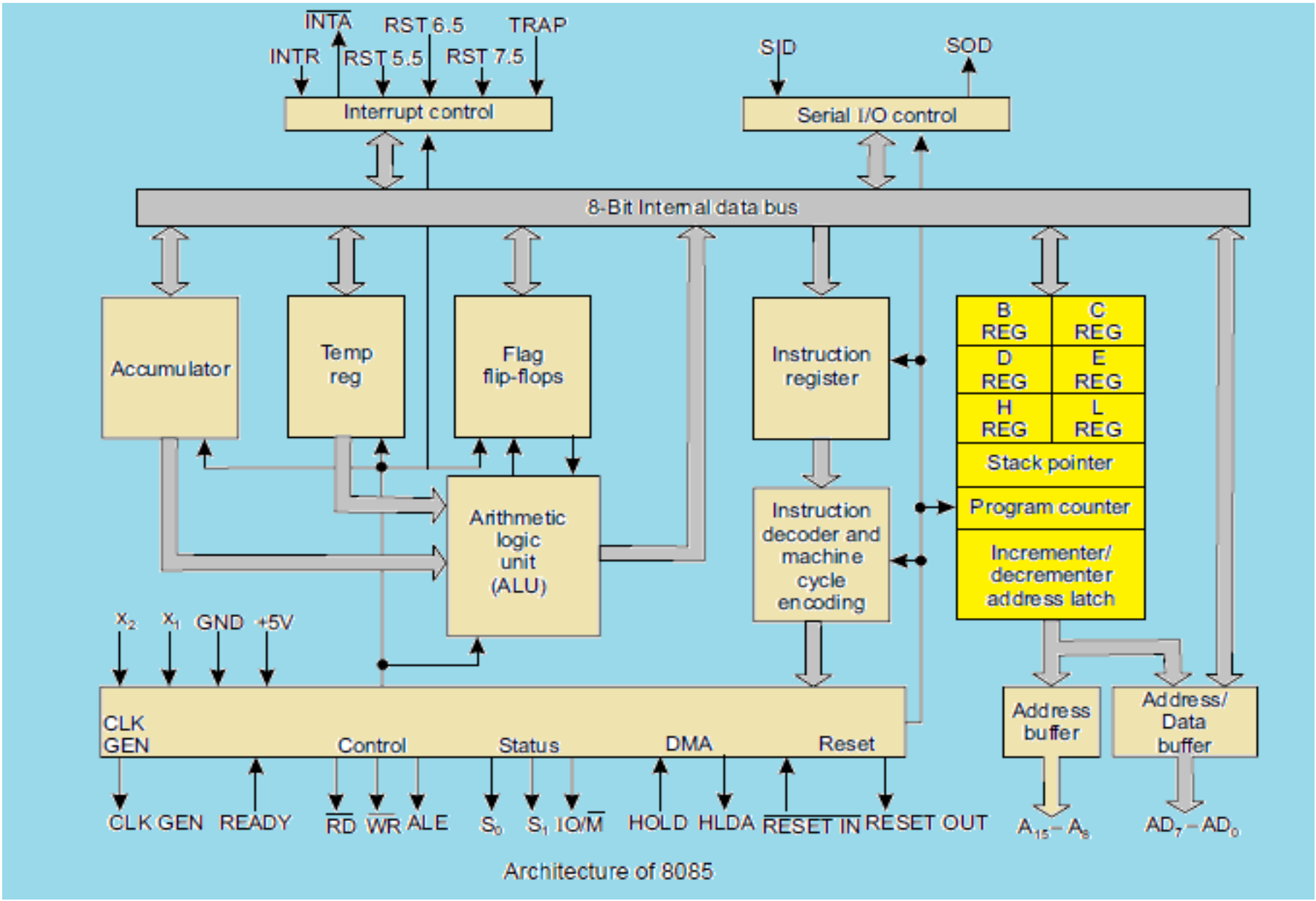
Module 1: 8085 Microprocessor

Content: Internal architecture of 8085 Microprocessor

Session objective:

Session outcome:

Internal architecture of 8085 microprocessor



Architecture of 8085

1. ALU

- The ALU performs the actual numerical and logic operation such as 'add', 'subtract', 'AND', 'OR' etc.
- Uses data from memory and from Accumulator to perform arithmetic operation and always stores result of operation in Accumulator.
- The ALU consists of accumulator, flag register and temporary register.

a. Accumulator

- The accumulator is an 8-bit register that is a part of arithmetic/logic unit (ALU). This register is used to store 8-bit data and to perform arithmetic and logical operations. The result of an operation is stored in the accumulator.
- The accumulator is also identified as register A.

b. Flag register

- 8085 has 8-bit flag register. There are only 5 active flags.

S	Z		AC		P		CY

Fig: 8085 flag register

- Flags are flip-flops which are used to indicate the status of the accumulator and other register after the completion of operation.
- These flip-flops are set or reset according to the data condition of the result in the accumulator and other registers.

i. Sign flag(S):

- Sign flag indicates whether the result of a mathematical or logical operation is negative or positive.
- If the result is negative, this flag will be set (i.e. S=1) and if the result is positive, the flag will be reset (i.e. S=0).

ii. Zero flag (Z):

- Zero flag indicates whether the result of a mathematical or logical operation is zero or not.
- If the result of current operation is zero, the flag will be set (i.e. Z=1) otherwise the flag will be reset (Z=0).
- This flag will be modified by the result in the accumulator as well as in the other register.

iii. Auxiliary carry flag (AC):

- In operation when a carry is generated by bit D_3 and passes on to bit D_4, the AC flag will be set otherwise AC flag will be reset.
- This flag is used only internally for BCD operation and is not available for the programmer to change the sequence of program with the jump instruction.

iv. Parity flag (P):

- This flag indicates whether the current result is of even parity (no. of 1's is even) or odd parity (no. of 1's is odd).
- If even parity, P flag will be set otherwise reset.

v. Carry flag (CY):

- This flag indicates whether during an addition or subtraction operation carry or borrow is generated or not.
- If carry or borrow is generated, the flag will be set otherwise reset.

2. Timing and control unit

- This unit produces all the timing and control signal for all the operation.
- This unit synchronizes all the MP operations with the clock and generates the control signals necessary for communication between the MP and peripherals.

3. Instruction register and decoder

- The instruction register and decoder are part of ALU. When an instruction is fetched from memory, it is loaded in the instruction register.
- The decoder decodes the instruction and establishes the sequence of events to follow.
- The IR is not programmable and cannot be accessed through any instruction.

4. Register array

- The register unit of 8085 consists of

  -Six general-purpose data registers B,C,D,E,H,L
  -Two internal registers W and Z
  -Two 16-bit address registers PC (program counter) and SP (stack pointer)
  -One increment/decrement counter register
  -And, one multiplexer (MUX)

- The six general-purpose registers are used to store 8-bit data. They can be combined as register pairs BC, DE, and HL to perform some 16-bit operations.
- The two internal registers W and Z are used to hold 8-bit data during the execution of some instructions, CALL and XCHG instructions.
- SP is 16-bit registers used to point the address of data stored in the stack memory. It always indicates the top of the stack.

5. System bus

a. Data bus

- It carries 'data', in binary form, between MP and other external units, such as memory.
- Typical size is 8 or 16 bits.

b. Address bus

- It carries 'address' of operand in binary form.
- Typical size is 16-bit.

c. Control Bus

- Control Bus are various lines which have specific functions for coordinating and controlling MP operations.
- E.g.: Read/Write control line.

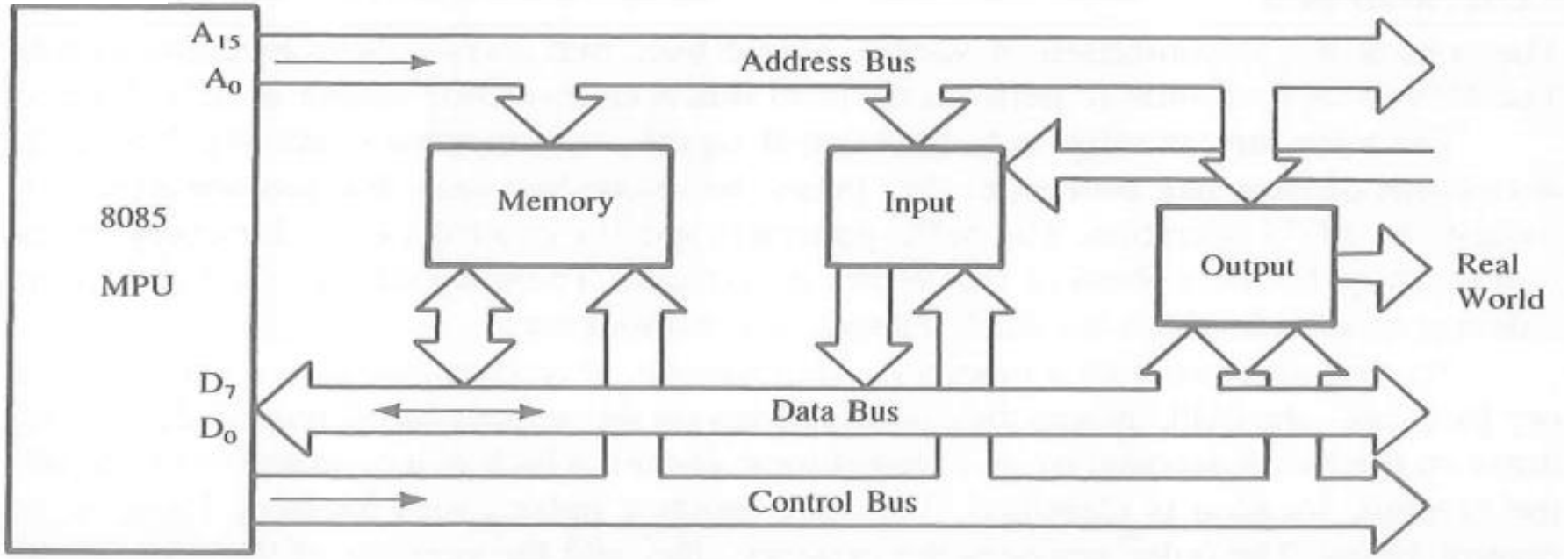
6. Interrupt Control

- Interrupt is a signal, which suspends the routine what the MP is doing, brings the control to perform the subroutine, completes it and returns to main routine.
- May be hardware or software interrupts. Some interrupts may be ignored (maskable), some cannot (non-maskable).
- E.g. INTR, TRAP, RST 7.5, RST 6.5, RST 5.5

7. Serial I/O Control

- The MP performs serial data input or output (one bit at a time). In serial transmission, data bits are sent over a single line, one bit at a time.
- The 8085 has two signals to implement the serial transmission: SID (serial input data) and SOD (serial output data).

8085 bus structure



8085 Bus Structure:

Address Bus:

- The address bus is a group of 16 lines generally identified as A0 to A15.
- The address bus is unidirectional: bits flow in one direction-from the MPU to peripheral devices.
- The MPU uses the address bus to perform the first function: identifying a peripheral or a memory location

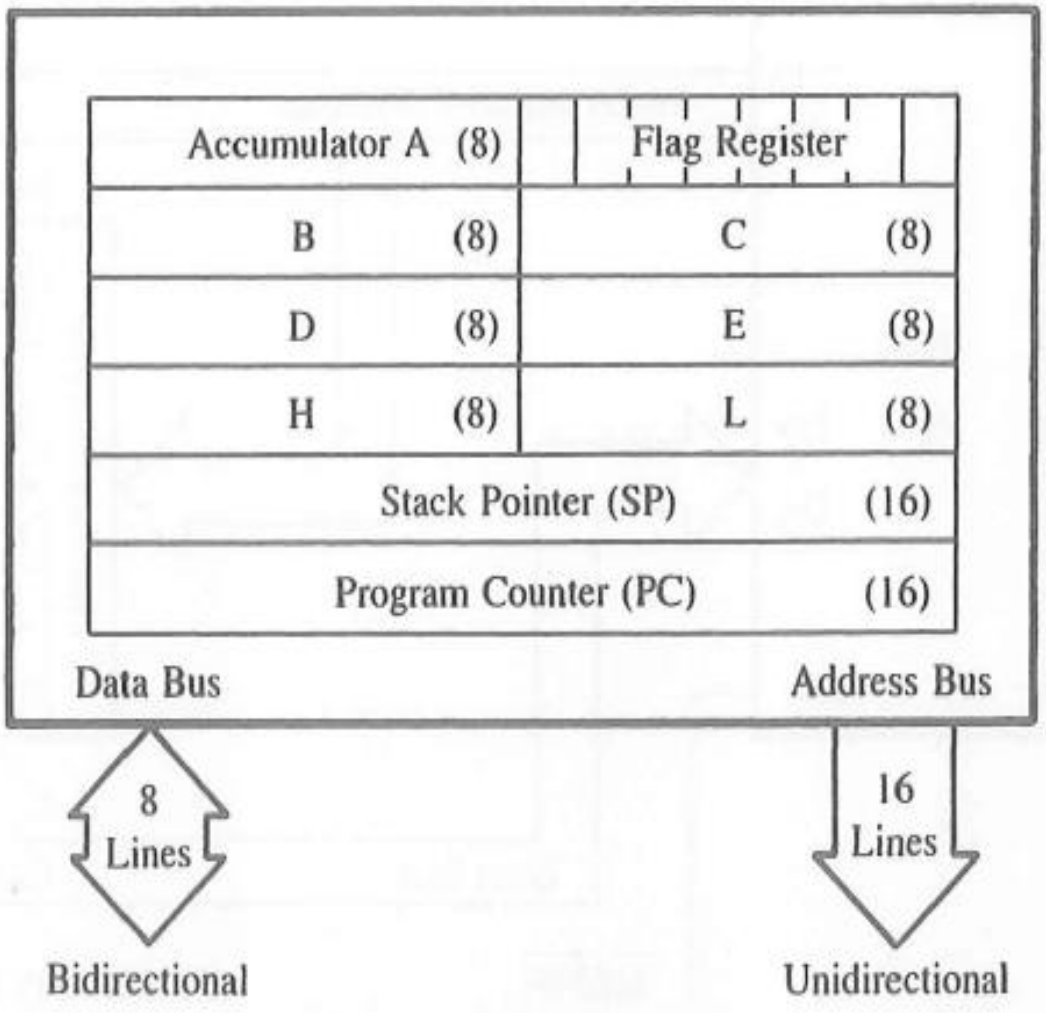
Data Bus:

- The data bus is a group of eight lines used for data flow.
- These lines are bi-directional - data flow in both directions between the MPU and memory and peripheral devices.
- The MPU uses the data bus to perform the second function: transferring binary information.
- The eight data lines enable the MPU to manipulate 8-bit data ranging from 00 to FF (28 = 256 numbers).
- The largest number that can appear on the data bus is 11111111.

Control Bus:

- The control bus carries synchronization signals and providing timing signals.
- The MPU generates specific control signals for every operation it performs. These signals are used to identify a device type with which the MPU wants to communicate.

Registers of 8085:



- The 8085 have six general-purpose registers to store 8-bit data during program execution.
- These registers are identified as B, C, D, E, H, and L.
- They can be combined as register pairs-BC, DE, and HL-to perform some 16-bit operations.

Accumulator (A):

- The accumulator is an 8-bit register that is part of the arithmetic/logic unit (ALU).
- This register is used to store 8-bit data and to perform arithmetic and logical operations.
- The result of an operation is stored in the accumulator.

Flags:

- The ALU includes five flip-flops that are set or reset according to the result of an operation.
- The microprocessor uses the flags for testing the data conditions.
- They are Zero (Z), Carry (CY), Sign (S), Parity (P), and Auxiliary Carry (AC) flags. The most commonly used flags are Sign, Zero, and Carry.

The bit position for the flags in flag register is,

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

Module 1: 8085 Microprocessor

Content: ADDRESSING MODES OF 8085

ADDRESSING MODES OF 8085

Every instruction of a program has to operate on a data.

The method of specifying the data to be operated by the instruction is called

Addressing.

The 8085 has the following 5 different types of addressing.

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

1. Immediate Addressing:

In immediate addressing mode, the data is specified in the instruction itself. The data will be a part of the program instruction.

EX. MVI B, 3EH - Move the data 3EH given in the instruction to B register; LXI SP, 2700H.

2. Direct Addressing:

In direct addressing mode, the address of the data is specified in the instruction. The data will be in memory. In this addressing mode, the program instructions and data can be stored in different memory.

EX. LDA 1050H - Load the data available in memory location 1050H in to accumulator; SHLD 3000H

3. Register Addressing:

In register addressing mode, the instruction specifies the name of the register in which the data is available.

EX. MOV A, B - Move the content of B register to A register; SPHL; ADD C.

4. Register Indirect Addressing:

In register indirect addressing mode, the instruction specifies the name of the register in which the address of the data is available. Here the data will be in memory and the address will be in the register pair.

EX. MOV A, M - The memory data addressed by H L pair is moved to A register.

LDAX B.

5. Implied Addressing:

In implied addressing mode, the instruction itself specifies the data to be operated.

Opcode and Operand

An **instruction** is a command to the microprocessor to perform a given task on specified data. Each instruction has two parts: one is the task to be performed, called the **operation code** (op-code), and the second is the data to be operated on, called the **operand.** The operand (or data) can be specified in various ways. It may include 8-bit (or 16-bit) data, an internal register, a memory location, or an 8-bit (or 16-bit) address. In some instructions, the operand is implicit.

- **Instruction word size**

The 8085 instruction set is classified into the following three groups according to word size or byte size.

In the 8085, "byte" and "word" are synonymous because it is an 8-bit microprocessor. However, instructions are commonly referred to in terms of bytes rather than words.

1. 1-byte instructions
2. 2-byte instructions
3. 3-byte instructions

One byte instruction

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

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

Two byte instruction

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

Task	Opcode	Operand	Binary Code	Hex Code	
Load an 8-bit data byte in the accumulator.	MVI	A,32H	0011 1110	3E	First Byte
			0011 0010	32	Second Byte
Load an 8-bit data byte in register B.	MVI	B,F2H	0000 0110	06	First Byte
			1111 0010	F2	Second Byte

Three byte instruction

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

Task	Opcode	Operand	Binary Code	Hex Code*	
Load contents of memory 2050H into A.	LDA	2050H	0011 1010	3A	First Byte
			0101 0000	50	Second Byte
			0010 0000	20	Third Byte
Transfer the program sequence to memory location 2085H.	JMP	2085H	1100 0011	C3	First Byte
			1000 0101	85	Second Byte
			0010 0000	20	Third Byte

Module 1: 8085 Microprocessor

Content: INSTRUCTION SET OF INTEL 8085

INSTRUCTION SET OF INTEL 8085

An Instruction is a command given to the computer to perform a specified operation on given data. The instruction set of a microprocessor is the collection of the instructions that the microprocessor is designed to execute. The programmer can write a program in assembly language using these instructions.

These instructions have been classified into the following groups:

1. Data Transfer Group
2. Arithmetic Group
3. Logical Group
4. Branch Control Group
5. I/O and Machine Control Group

Data Transfer Group

- Instructions, which are used to transfer data from one register to another register, from memory to register or register to memory, come under this group. Examples are: MOV, MVI, LXI, LDA, STA etc. When an instruction of data transfer group is executed, data is transferred from the source to the destination without altering the contents of the source. For example, when MOV A, B is executed the content of the register B is copied into the register A, and the content of register B remains unaltered. Similarly, when LDA 2500 is executed the content of the memory location 2500 is loaded into the accumulator. But the content of the memory location 2500 remains unaltered.

Arithmetic Group

- The instructions of this group perform arithmetic operations such as addition, subtraction; increment or decrement of the content of a register or memory. Examples are: ADD, SUB, INR, DAD etc.

Logical Group

- The Instructions under this group perform logical operation such as AND, OR, compare, rotate etc. Examples are: ANA, XRA, ORA, CMP, and RAL etc.

Branch Control Group

- This group includes the instructions for conditional and unconditional jump, subroutine call and return, and restart. Examples are: JMP, JC, JZ, CALL, CZ, RST etc.

I/O and Machine Control Group

- This group includes the instructions for input/output ports, stack and machine control. Examples are: IN, OUT, PUSH, POP, and HLT etc.

Data Transfer Group

1. MOV r1, r2 (Move Data; Move the content of the one register to another). [r1] <-- [r2]
2. MOV r, m (Move the content of memory register). r <-- [M]
3. MOV M, r. (Move the content of register to memory). M <-- [r]
4. MVI r, data. (Move immediate data to register). [r] <-- data.
5. MVI M, data. (Move immediate data to memory). M <-- data.
6. LXI rp, data 16. (Load register pair immediate). [rp] <-- data 16 bits, [rh] <-- 8 LSBs of data.
7. LDA addr. (Load Accumulator direct). [A] <-- [addr].
8. STA addr. (Store accumulator direct). [addr] <-- [A].
9. LHLD addr. (Load H-L pair direct). [L] <-- [addr], [H] <-- [addr+1].
10. SHLD addr. (Store H-L pair direct) [addr] <-- [L], [addr+1] <-- [H].
11. LDAX rp. (LOAD accumulator indirect) [A] <-- [[rp]]
12. STAX rp. (Store accumulator indirect) [[rp]] <-- [A]. 13. XCHG. (Exchange the contents of H-L with D-E pair) [H-L] <--> [D-E].

2. Arithmetic Group

1. ADD r. (Add register to accumulator) [A] <-- [A] + [r].
2. ADD M. (Add memory to accumulator) [A] <-- [A] + [[H-L]].
3. ADC r. (Add register with carry to accumulator). [A] <-- [A] + [r] + [CS].
4. ADC M. (Add memory with carry to accumulator) [A] <-- [A] + [[H-L]] [CS].
5. ADI data (Add immediate data to accumulator) [A] <-- [A] + data.
6. ACI data (Add with carry immediate data to accumulator). [A] <-- [A] + data + [CS].
7. DAD rp. (Add register paid to H-L pair). [H-L] <-- [H-L] + [rp].
8. SUB r. (Subtract register from accumulator). [A] <-- [A] – [r].
9. SUB M. (Subtract memory from accumulator). [A] <-- [A] – [[H-L]].
10. SBB r. (Subtract register from accumulator with borrow). [A] <-- [A] – [r] – [CS].

11. SBB M. (Subtract memory from accumulator with borrow). [A] <-- [A] – [[H-L]] – [CS].
12. SUI data. (Subtract immediate data from accumulator) [A] <-- [A] – data.
13. SBI data. (Subtract immediate data from accumulator with borrow). [A] <-- [A] – data – [CS].
14. INR r (Increment register content) [r] <-- [r] +1.
15. INR M. (Increment memory content) [[H-L]] <-- [[H-L]] + 1.
16. DCR r. (Decrement register content). [r] <-- [r] – 1.
17. DCR M. (Decrement memory content) [[H-L]] <-- [[H-L]] – 1.
18. INX rp. (Increment register pair) [rp] <-- [rp] – 1.
19. DCX rp (Decrement register pair) [rp] <-- [rp] -1.
20. DAA (Decimal adjust accumulator)

DAA (Decimal adjust accumulator)

- The instruction DAA is used in the program after ADD, ADI, ACI, ADC, etc instructions. After the execution of ADD, ADC, etc instructions the result is in hexadecimal and it is placed in the accumulator. The DAA instruction operates on this result and gives the final result in the decimal system. It uses carry and auxiliary carry for decimal adjustment. 6 is added to 4 LSBs of the content of the accumulator if their value lies in between A and F or the AC flag is set to 1. Similarly, 6 is also added to 4 MSBs of the content of the accumulator if their value lies in between A and F or the CS flag is set to 1. All status flags are affected. When DAA is used data should be in decimal numbers

Module 1: 8085 Microprocessor

Content: INSTRUCTION SET OF INTEL 8085

INSTRUCTION SET OF INTEL 8085

An Instruction is a command given to the computer to perform a specified operation on given data. The instruction set of a microprocessor is the collection of the instructions that the microprocessor is designed to execute. The programmer can write a program in assembly language using these instructions.

These instructions have been classified into the following groups:

1. Data Transfer Group
2. Arithmetic Group
3. Logical Group
4. Branch Control Group
5. I/O and Machine Control Group

Logical Group

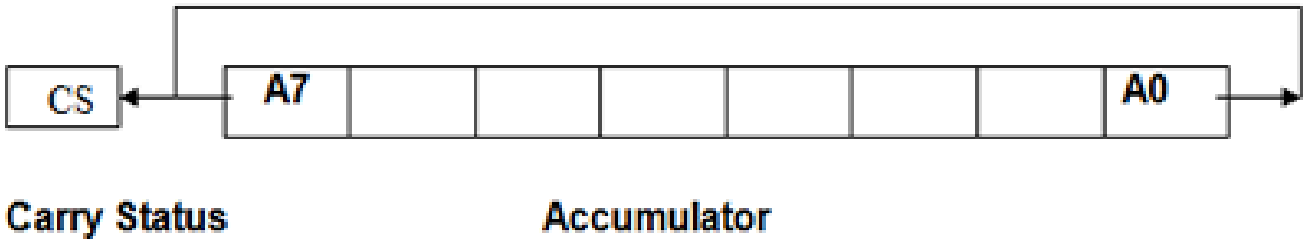
1. ANA r. (AND register with accumulator) [A] <-- [A] ^ [r].
2. ANA M. (AND memory with accumulator). [A] <-- [A] ^ [[H-L]].
3. ANI data. (AND immediate data with accumulator) [A] <-- [A] ^ data.
4. ORA r. (OR register with accumulator) [A] <-- [A] v [r].
5. ORA M. (OR memory with accumulator) [A] <-- [A] v [[H-L]]
6. ORI data. (OR immediate data with accumulator) [A] <-- [A] v data.
7. XRA r. (EXCLUSIVE – OR register with accumulator) [A] <-- [A] v [r]
8. XRA M. (EXCLUSIVE-OR memory with accumulator) [A] <-- [A] v [[H-L]]
9. XRI data. (EXCLUSIVE-OR immediate data with accumulator) [A] <-- [A]
10. CMA. (Complement the accumulator) [A] <-- [A]
11. CMC. (Complement the carry status) [CS] <-- [CS]
12. STC. (Set carry status) [CS] <-- 1.
13. CMP r. (Compare register with accumulator) [A] – [r]
14. CMP M. (Compare memory with accumulator) [A] – [[H-L]]

15. CPI data. (Compare immediate data with accumulator) [A] – data. The 2nd byte of the instruction is data, and it is subtracted from the content of the accumulator. The status flags are set according to the result of subtraction. But the result is discarded. The content of the accumulator remains unchanged.

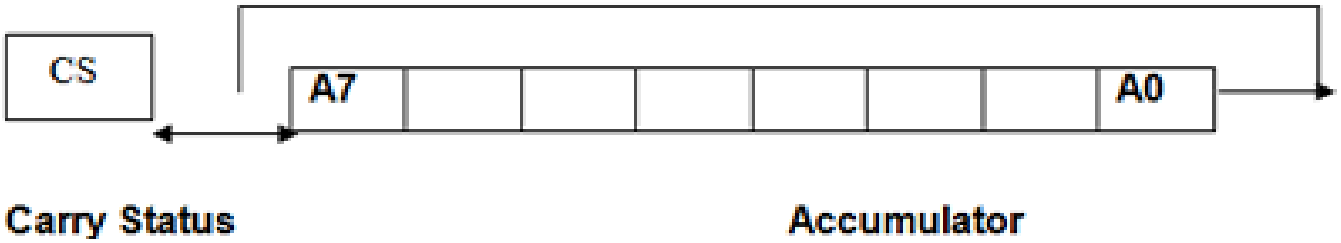
16. RLC (Rotate accumulator left) [An+1] <-- [An], [A0] <-- [A7],[CS] <-- [A7].

The content of the accumulator is rotated left by one bit.

The seventh bit of the accumulator is moved to carry bit as well as to the zero bit of the accumulator. Only CS flag is affected.



RRC. (Rotate accumulator right) [A7] <-- [A0], [CS] <-- [A0], [An] <--[An+1].



The content of the accumulator is rotated right by one bit. The zero bit of the accumulator is moved to the seventh bit as well as to carry bit. Only CS flag is affected.

18. RAL. (Rotate accumulator left through carry) [An+1] <-- [An], [CS] <-- [A7], [A0] <-- [CS].

19. RAR. (Rotate accumulator right through carry) [An] <-- [An+1], [CS] <-- [A0], [A7] <-- [CS]

4. Branch Group

1. JMP addr (label). (Unconditional jump: jump to the instruction specified by the address). [PC] <-- Label.
2. Conditional Jump addr (label): After the execution of the conditional jump instruction the program jumps to the instruction specified by the address (label) if the specified condition is fulfilled. The program proceeds further in the normal sequence if the specified condition is not fulfilled. If the condition is true and program jumps to the specified label, the execution of a conditional jump takes 3 machine cycles: 10 states. If condition is not true, only 2 machine cycles; 7 states are required for the execution of the instruction.

1. JZ addr (label). (Jump if the result is zero)
2. JNZ addr (label) (Jump if the result is not zero)
3. JC addr (label). (Jump if there is a carry)
4. JNC addr (label). (Jump if there is no carry)
5. JP addr (label). (Jump if the result is plus)
6. JM addr (label). (Jump if the result is minus)
7. JPE addr (label) (Jump if even parity)
8. JPO addr (label) (Jump if odd parity)

3. CALL addr (label) (Unconditional CALL: call the subroutine identified by the operand)

CALL instruction is used to call a subroutine. Before the control is transferred to the subroutine, the address of the next instruction 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 by the label.

4. RET (Return from subroutine)

5. RST n (Restart) Restart is a one-word CALL instruction. The content of the program counter is saved in the stack. The program jumps to the instruction starting at restart location.

5. Stack, I/O and Machine Control Group

1. IN port-address. (Input to accumulator from I/O port) [A] <-- [Port]
2. OUT port-address (Output from accumulator to I/O port) [Port] <-- [A]
3. PUSH rp (Push the content of register pair to stack)
4. PUSH PSW (PUSH Processor Status Word)
5. POP rp (Pop the content of register pair, which was saved, from the stack)
6. POP PSW (Pop Processor Status Word)
7. HLT (Halt)
8. XTHL (Exchange stack-top with H-L)
9. SPHL (Move the contents of H-L pair to stack pointer)
10. EI (Enable Interrupts)
11. DI (Disable Interrupts)
12. SIM (Set Interrupt Masks)
13. RIM (Read Interrupt Masks)
14. NOP (No Operation)

Module 1: 8085 Microprocessor

Content: Instruction cycle

Session objective:

Session outcome:

Instruction cycle

The timing and control unit generates timing signals for the execution of instruction and control of peripheral devices. The timing used for the execution of instructions and control of peripherals are different for different microprocessors. The design and cost of a processor also depends on the timing structure and register organization.

For the execution of an instruction a microprocessor fetches the instruction from the memory and executes it. The time taken for the execution of an instruction is called **instruction cycle (IC).**

An instruction cycle consists of a **fetch cycle (FC) and an execute cycle (EC).**

A fetch cycle is the time required for the fetch operation in which the machine code of the instruction (opcode) is fetched from the memory. This time is a fixed slot of time. An execute cycle is of variable width which depends on the instruction to be executed.

The total time for the execution is given by IC = FC + EC

Fetch Operation

In fetch operation the microprocessor gets the 1st byte of the instruction, which is operation code (opcode), from the memory. The program counter keeps the track of address of the next instruction to be executed. In the beginning of the fetch cycle the content of the program counter is sent to the memory. This **takes one clock cycle.**

The memory first reads the opcode. This operation also takes **one clock cycle**.

Then the memory sends the opcode to the microprocessor, which takes **one clock period.**

The total time for fetch operation is the time required for fetching an opcode from the memory. This time is **called fetch cycle**. Having received the address from the microprocessor the memory takes two clock cycles to respond as explained above. If the memory is slow, it may take more time. In that case the microprocessor has to wait for some time till it receives the opcode from the memory. The time for which the microprocessor waits is called wait cycle. Most of the microprocessor have provision for wait cycles to cope with slow memory.

Execute Operation

The opcode fetched from the memory goes to the data register, DR (data/address buffer in Intel 8085) and then to instruction register, IR. From the instruction register it goes to the decoder circuitry is within the microprocessor. After the instruction is decoded, execution begins. If the operand is in the general purpose registers, execution is immediately performed. The time taken in decoding and the address of the data, some read cycles are also necessary to receive the data from the memory. These read cycle are similar to opcode fetch cycle. The fetch quantities in these cycles are address or data. Figure (a) and Figure (b) shows an instruction and fetch cycle respectively

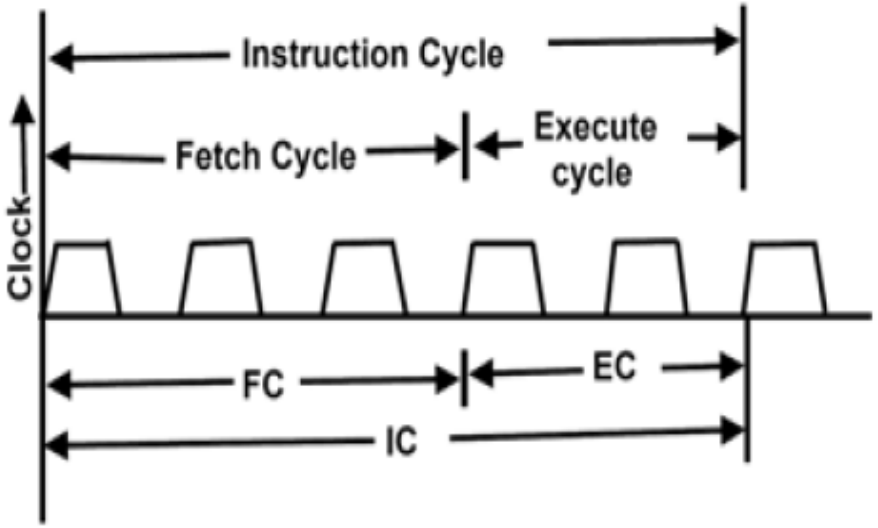


Figure (a) Instruction cycle showing FC, EC and IC

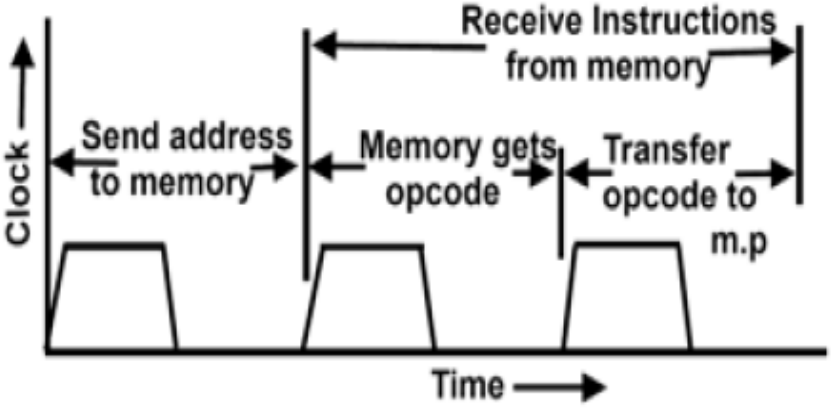


Figure (b) A Typical Fetch Cycle

Machine Cycle

Machine cycles of 8085

The 8085 microprocessor has 5 (seven) basic machine cycles. They are

- ✓ Opcode fetch cycle (4T)
- ✓ Memory read cycle (3 T)
- ✓ Memory write cycle (3 T)
- ✓ I/O read cycle (3 T)
- ✓ I/O write cycle (3 T)

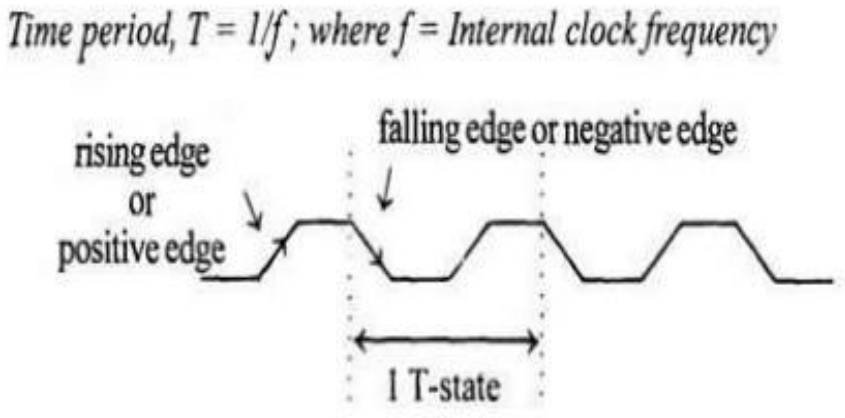
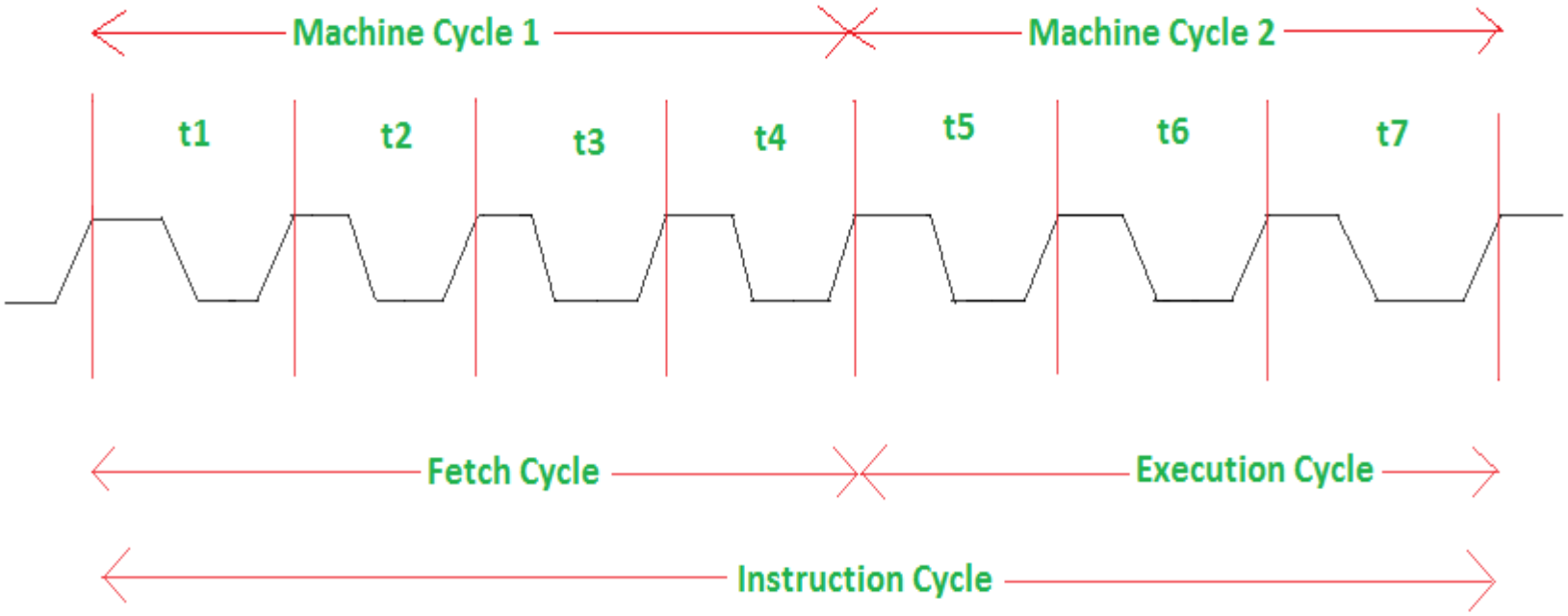


Fig 1.7 Clock Signal



Instruction cycle in 8085 microprocessor

The time required by the microprocessor to complete an operation of accessing memory or input/output devices is called machine cycle. One time period of frequency of microprocessor is called t-state. A t-state is measured from the falling edge of one clock pulse to the falling edge of the next clock pulse.

Fetch cycle takes four t-states and execution cycle takes three t-states.

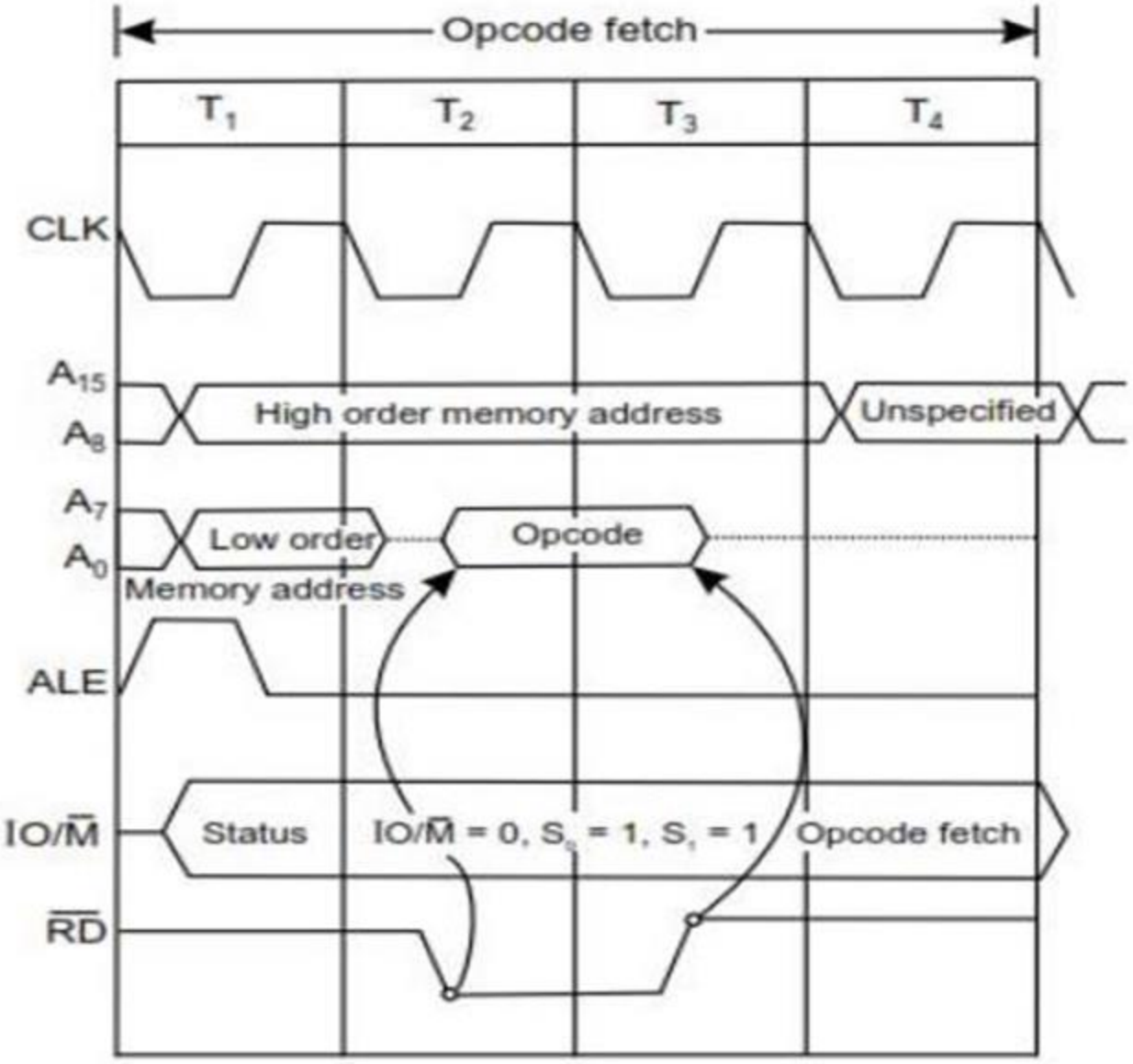


Fig Opcode fetch machine cycle

1.Opcode fetch machine cycle of 8085 :

- Each instruction of the processor has one byte opcode.

The opcodes are stored in memory. So, the processor executes the opcode fetch machine cycle to fetch the opcode from memory.

Hence, every instruction starts with opcode fetch machine cycle.

The time taken by the processor to execute the opcode fetch cycle is 4T.

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

2. Memory Read Machine Cycle of 8085:

- The memory read machine cycle is executed by the processor to read a data byte from memory.
- The processor takes 3T states to execute this cycle.
- The instructions which have more than one byte word size will use the machine cycle after the opcode fetch machine cycle.

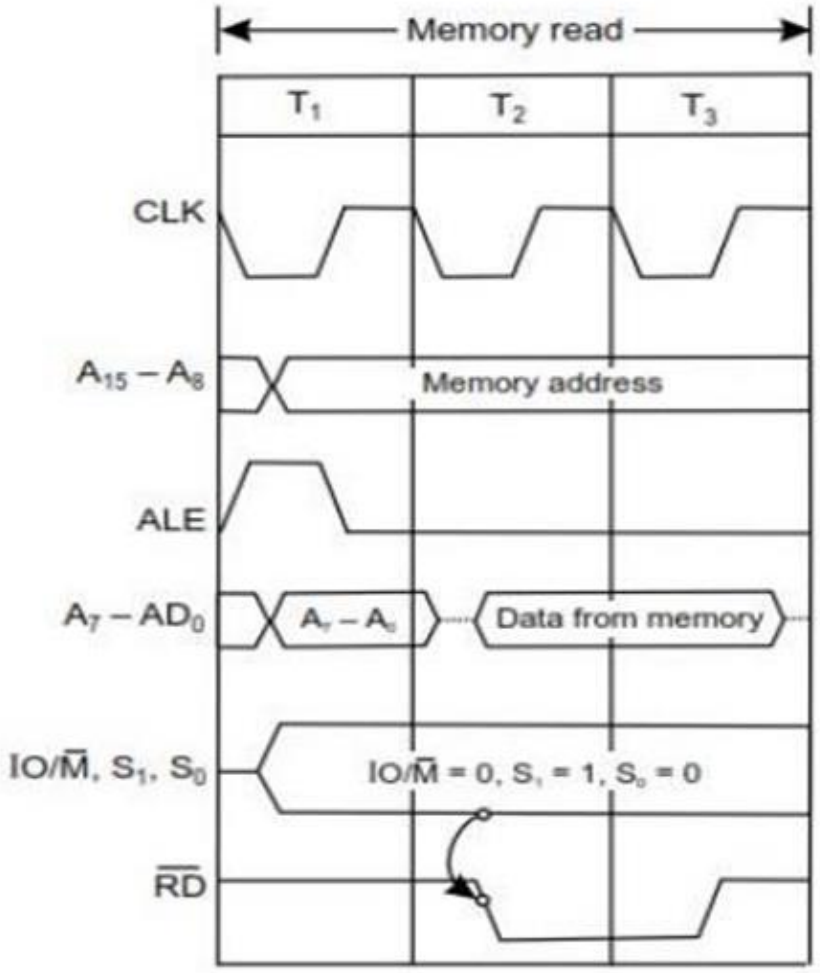


Fig Memory Read Machine Cycle

3. Memory Write Machine Cycle of 8085

- The memory write machine cycle is executed by the processor to write a data byte in a memory location.
- The processor takes, 3T states to execute this machine cycle.

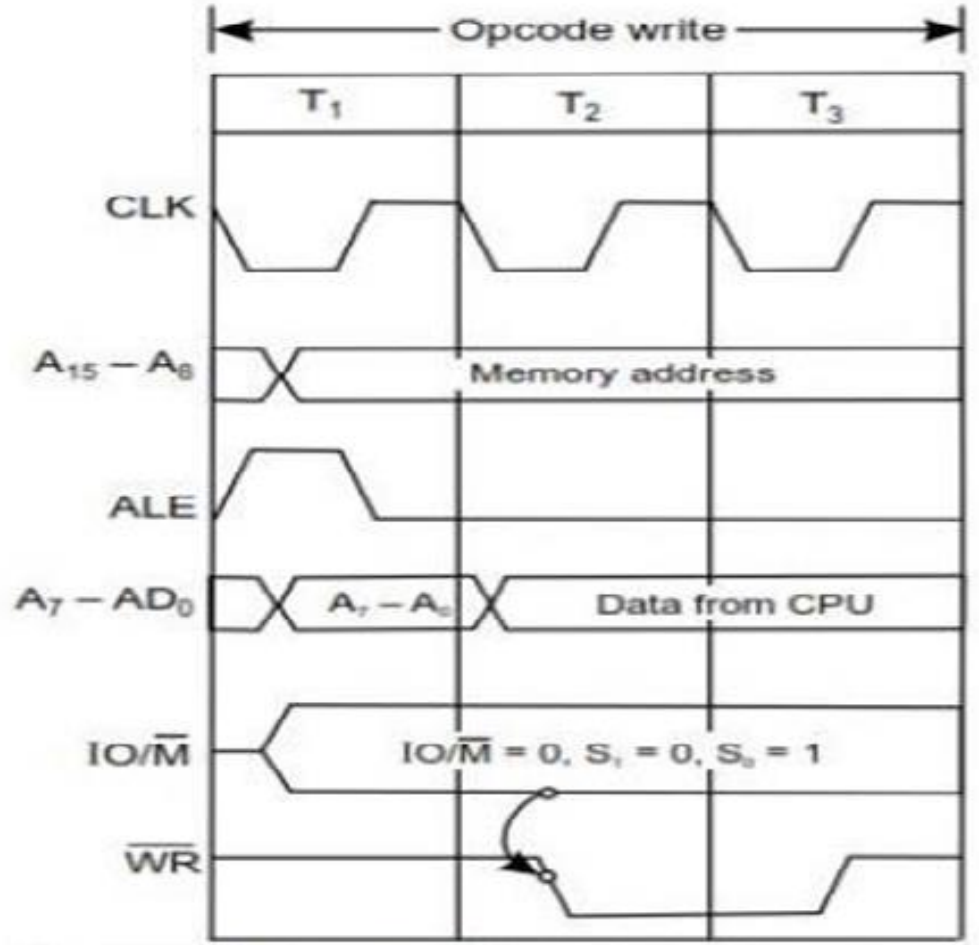


Fig Memory Write Machine Cycle

4. I/O Read Cycle of 8085

- The I/O Read cycle is executed by the processor to read a data byte from I/O port or from the peripheral, which is I/O, mapped in the system.
-
- The processor takes 3T states to execute this machine cycle.
- The IN instruction uses this machine cycle during the execution.

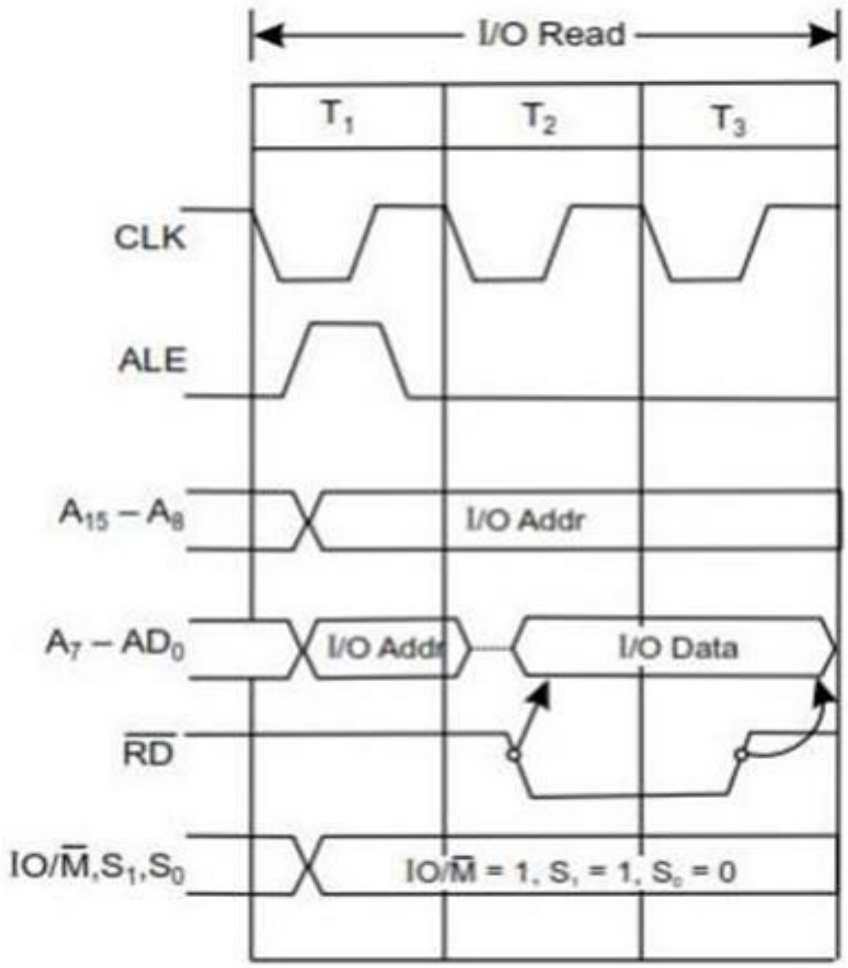


Fig I/O Read Cycle

Timing diagram for STA 526AH

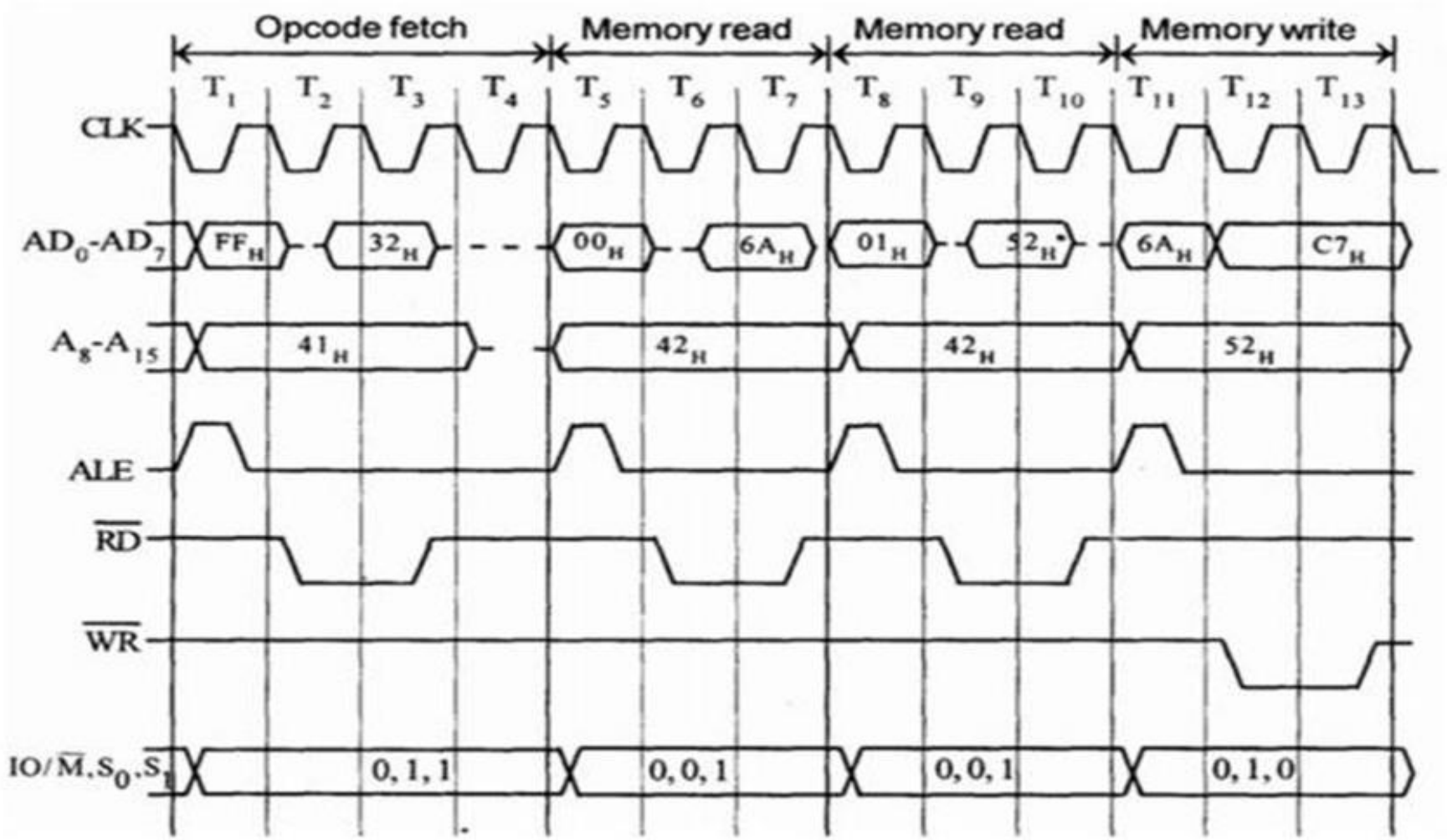


Fig Timing Diagram for STA 526A H

Address	Mnemonics	Opcode
41FF	STA 526A$_H$	32$_H$
4200		6A$_H$
4201		52$_H$

Timing diagram for TA 526AH

- STA means Store Accumulator -The contents of the accumulator is stored in the specified address (526A).
  The opcode of the STA instruction is said to be 32H. It is fetched from the memory 41FFH (see fig). - OF machine cycle
  Then the lower order memory address is read (6A). - Memory Read Machine Cycle
  Read the higher order memory address (52).- Memory Read Machine Cycle
  The combination of both the addresses are considered and the content from accumulator is written in 526A. - Memory Write Machine Cycle
  Assume the memory address for the instruction and let the content of accumulator is C7H. So, C7H from accumulator is now stored in 526A.

3. Timing diagram for INR M

- Fetching the Opcode 34H from the memory 4105H. (OF cycle)
- Let the memory address (M) be 4250H. (MR cycle - To read Memory address and data)
- Let the content of that memory is 12H.
- Increment the memory content from 12H to 13H. (MW machine cycle)

Address	Mnemonics	Opcode
4105	INR M	34_H

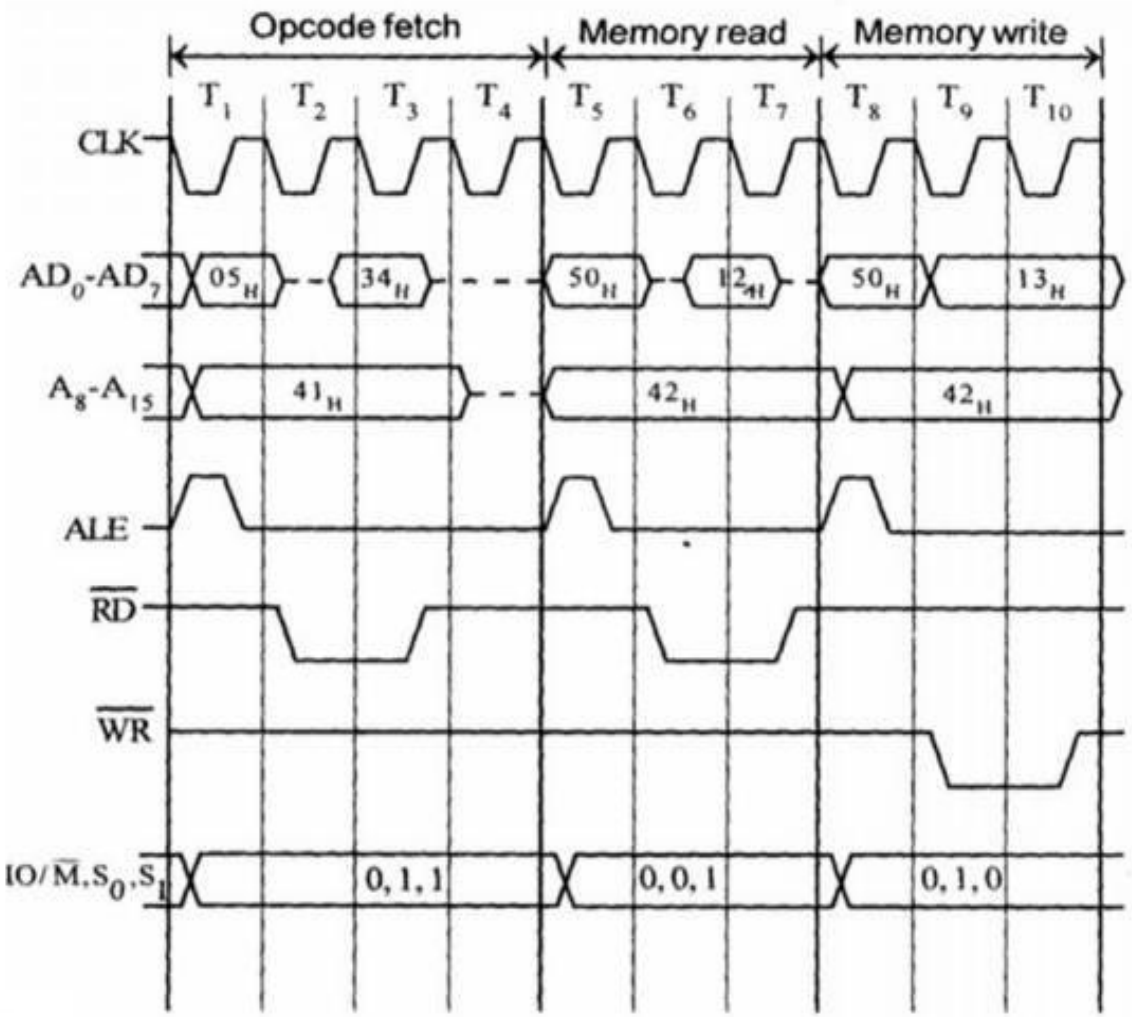


Fig 1.13 Timing Diagram for INR M

Instruction Fetch operation

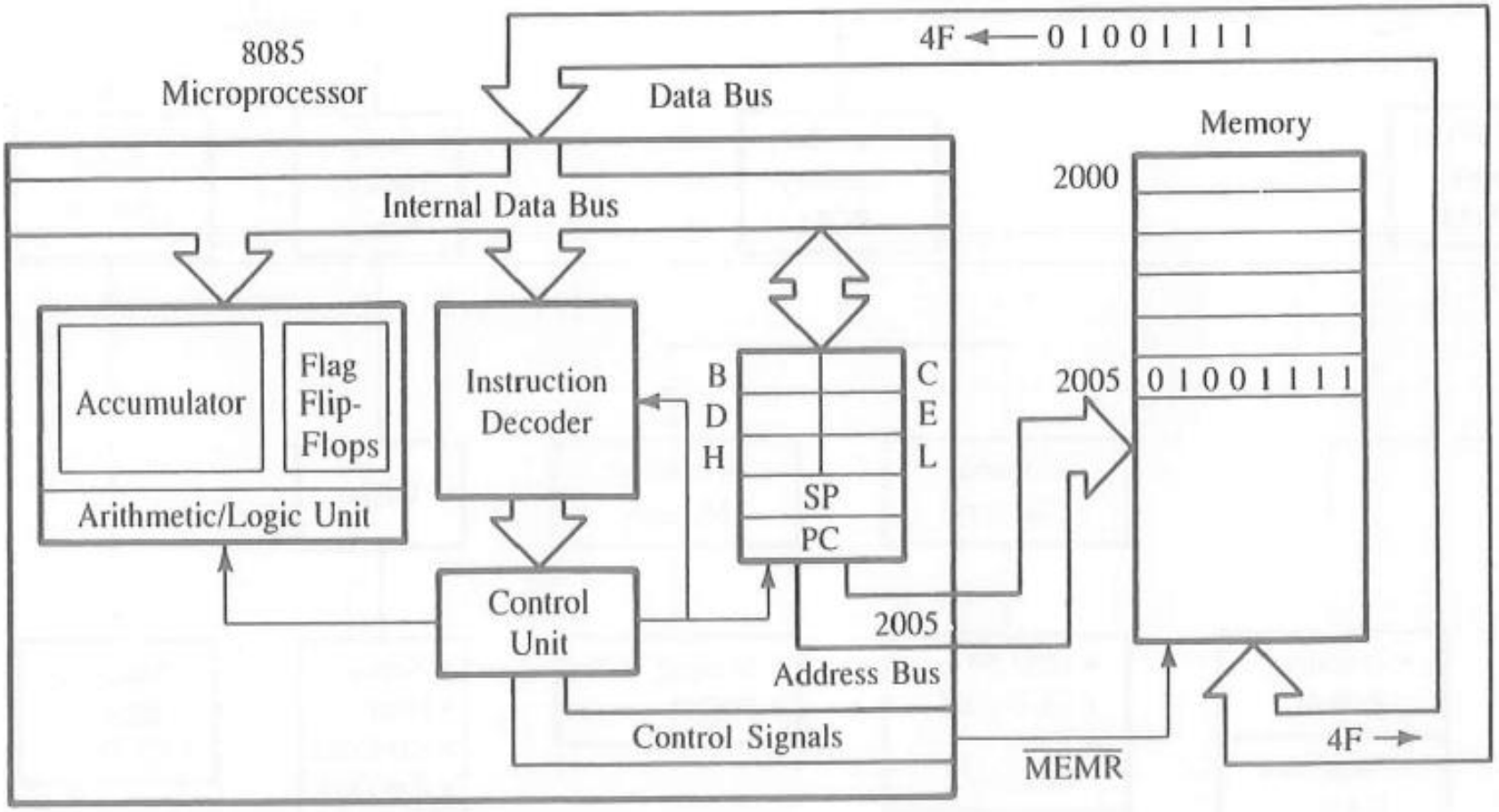


Figure 1

Lecture-XII

Module 1: 8085 Microprocessor

Content: Interrupts of INTEL 8085

Session objective:

Session outcome:

8085 interrupt structure and operation

Types of interrupt

The 8085 has multilevel interrupt system. It supports two types of interrupts:

a. Hardware b. Software

Hardware : Some pins on the 8085 allow peripheral device to interrupt the main program for I/O operations. When an interrupt occurs, the 8085 completes the instruction it is currently executing and transfers the program control to a subroutine that services the peripheral device. Upon completion of the service routine, the MPU returns to the main program. These types of interrupts, where MPU pins are used to receive interrupt requests, are called **hardware interrupts**.

Software : In software interrupts, the cause of the interrupt is an execution of the instruction. These are special instructions supported by the microprocessor. After execution of these instructions microprocessor completes the execution of the instruction it is currently executing and transfers the program control to the subroutine program. Upon completion of the execution of the subroutine program, program control returns to the main program.

Hardware interrupts in 8085

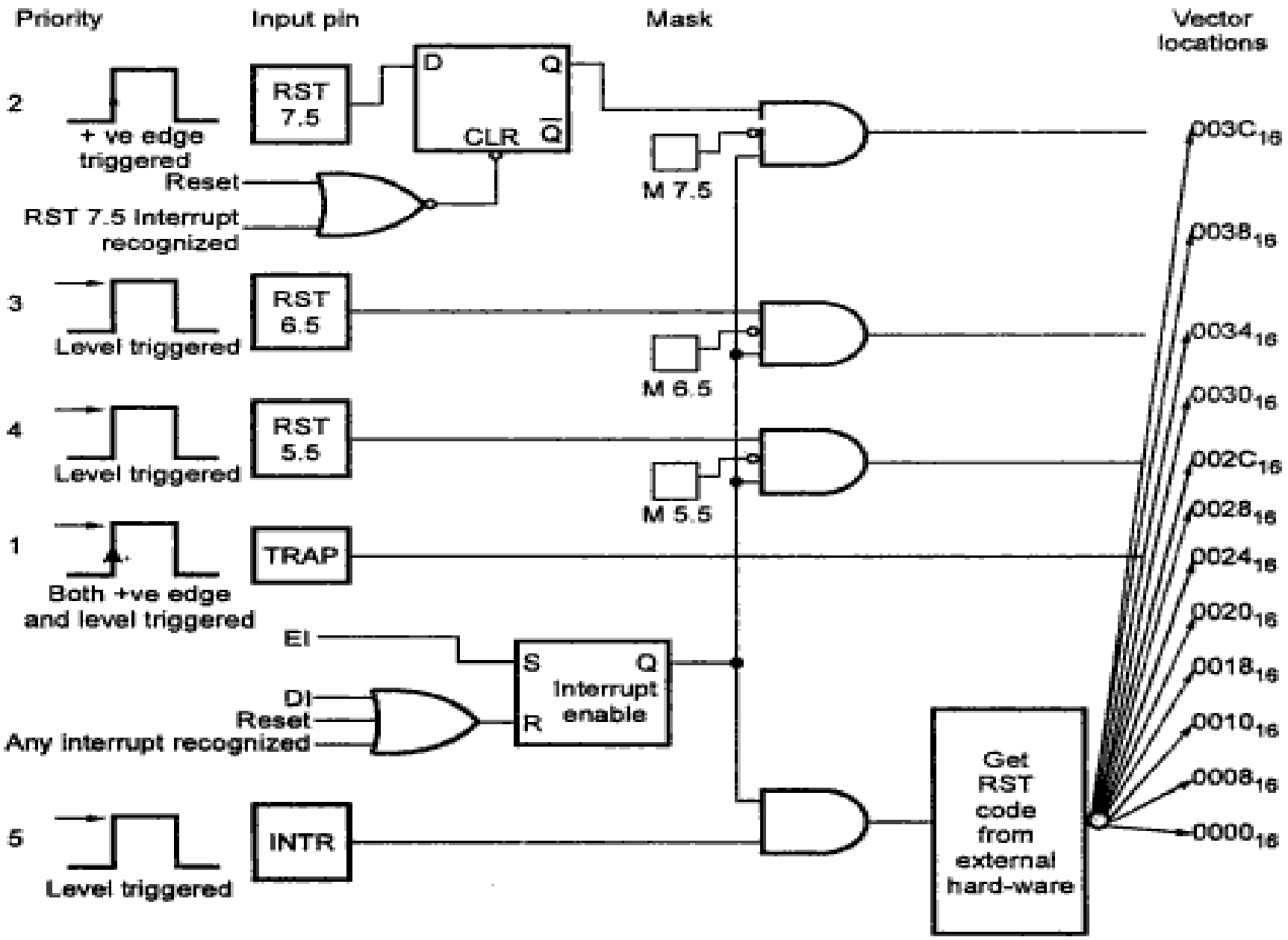
The 8085 has five hardware interrupts :

1. TRAP 2. RST 7.5 3. RST 6.5 4. RST 5.5 5. INTR

When any of these pins, except INTR, is active, the internal control circuit of the 8085 produces a CALL to a predetermined memory location. This memory location, where the subroutine starts is referred to as **vector location** and such interrupts are called **vectored interrupts**. The INTR is not a vectored interrupt. It receives the address of the subroutine from the external device.

In 8085, all interrupts except TRAP are maskable. When logic signal is applied to a maskable interrupt input, the 8085 is interrupted only if that particular input is enabled. These interrupts can be enabled or disabled under program control. If disabled, 8085 disables an interrupt request. The interrupt TRAP is nonmaskable whish means that it is not maskable by program control. The Fig. shows the interrupt structure of 8085. The figure indicates that, the 8085 is designed to respond to edge triggering, level triggering or both.

Interrupt structure of 8085



TRAP

This interrupt is a nonmaskable interrupt. It is unaffected by any mask or interrupt enable. TRAP has the highest priority. TRAP interrupt is edge and level triggered. This means that the TRAP must go high and remain high until it is acknowledged. This avoids false triggering caused by noise and transients.

There are two ways to clear TRAP interrupt :

1. By resetting microprocessor i.e. giving a low signal on $\overline{\text{RESETIN}}$ pin (External signal).
2. By giving a high TRAP ACKNOWLEDGE (Internal signal).

After recognition of TRAP interrupt, 8085 internally generates a high TRAP ACKNOWLEDGE which clears the flip flop. Once the TRAP is acknowledged, the 8085 completes its current instruction. It then pushes the address of the next instruction i.e. return address onto the stack and loads PC with the fixed vector address 0024H. Due to this, 8085 starts execution of instructions from address 0024H which is the starting address of an interrupt service routine for TRAP.

RST 7.5 : The RST 7.5 interrupt is a maskable interrupt. It has the second highest priority. As shown in Fig. it is positive edge triggered and the positive edge trigger is stored internally by the D-flip flop until it is cleared by software reset using SIM instruction or by internally generated ACKNOWLEDGE signal.

The positive edge signal on the RST 7.5 pin sets the D flip flop. If the mask bit M 7.5 is 0 i.e. RST 7.5 is unmasked then 8085 completes its current instruction. It then pushes the address of the next instruction onto the stack and loads PC with the fixed vector address 003CH. Due to this, 8085 starts execution of instructions from address 003CH which is the starting address of an interrupt service routine for RST 7.5.

RST 6.5 and RST 5.5 : The RST 6.5 and RST 5.5 both are level triggered. These interrupts can be masked using SIM instruction. The RST 6.5 has the third priority whereas RST 5.5 has the fourth priority. The vector addresses of RST 6.5 and RST 5.5 are 0034H and 002CH respectively. After recognition of RST 6.5 or RST 5.5 interrupt, 8085 completes its current instruction; pushes the address of next instruction onto the stack and loads PC with corresponding vector address.

INTR

: : INTR is a maskable interrupt, but not the vector interrupt. It has the lowest priority. The following sequence of events occur when INTR signal goes high.

1. The 8085 checks the status of INTR signal during execution of each instruction.
2. If INTR signal is high, then 8085 completes its current instruction and sends an active low interrupt acknowledge signal ($\overline{\text{INTA}}$) if the interrupt is enabled.
3. In response to the $\overline{\text{INTA}}$ signal, external logic places an instruction OPCODE on the data bus. In the case of multibyte instruction, additional interrupt acknowledge machine cycles are generated by the 8085 to transfer the additional bytes into the microprocessor.
4. On receiving the instruction, the 8085 saves the address of next instruction on stack and executes received instruction.

Note : Theoretically, the external logic can place any instruction code on the data bus in response to the $\overline{\text{INTA}}$. However, only CALL and RST codes save the contents of the PC on the stack and branch program control to the subroutine address.

Summary of hardware interrupts

Interrupt type	Trigger	Priority	Maskable	Vector address
TRAP	Edge and Level	1st (Highest)	No	0024 H
RST 7.5	Edge	2nd	Yes	003CH
RST 6.5	Level	3rd	Yes	0034H
RST 5.5	Level	4th	Yes	002CH
INTR	Level	5th (Lowest)	Yes	-

Software interrupts in 8085

The 8085 has eight software interrupts from RST 0 to RST 7. The vector address for these interrupts can be calculated as follows.

Interrupt number x 8 = vector address

For example :

$5 \times 8 = 40 = 28H$

∴ Vector address for interrupt RST 5 is 0028H.

Instruction	HEX code	Vector address
RST 0	C7	0000H
RST 1	CF	0008H
RST 2	D7	0010H
RST 3	DF	0018H
RST 4	E7	0020H
RST 5	EF	0028H
RST 6	F7	0030H
RST 7	FF	0038H

- Vector address of software interrupts

Masking and unmasking of Interrupts

- Maskable interrupts are enabled or disabled under program control. Three instructions for masking and unmasking of interrupts.

  1. EI 2.DI 3.SIM

EI : Enable Interrupt

The EI instruction sets the interrupt enable flip-flop, as shown in Fig. Thus RST 7.5, RST 6.5, RST 5.5 and INTR are enabled using EI instruction.

It is important to note that when any interrupt is acknowledged, interrupt enable flip flop resets and disables all interrupts. To enable interrupt in further process it is necessary to execute EI instruction within interrupt service routine.

DI : Disable Interrupt

The DI instruction resets the interrupt enable flip flop, as shown in Fig. . Thus it disables RST 7.5, RST 6.5, RST 5.5 and INTR interrupts.

SIM : Set Interrupt Mask

This instruction is used to set interrupt mask and to send serial output. It transfers the contents of accumulator to interrupt control logic and serial I/O port. Thus it is necessary to load appropriate contents in the accumulator before execution of SIM instruction.

Pending Interrupts

The Read Interrupt Mask, RIM, instruction is used to handle pending interrupts. It loads the status of the interrupt mask, the pending interrupts and the contents of the serial input data line, SID, into the accumulator. Thus, it is possible to monitor status of interrupt mask, pending interrupts and serial input. There are number of interrupts. When one interrupt is being serviced, other interrupt requests may occur. If the interrupt requests are of higher priority, 8085 branches program control to the requested interrupt service routines. But when the interrupt requests are of lower priority, 8085 stores the information about these interrupt requests. Such interrupts are called **pending interrupts**. The status of pending interrupts can be monitored using RIM instruction.

Accumulator content for SIM

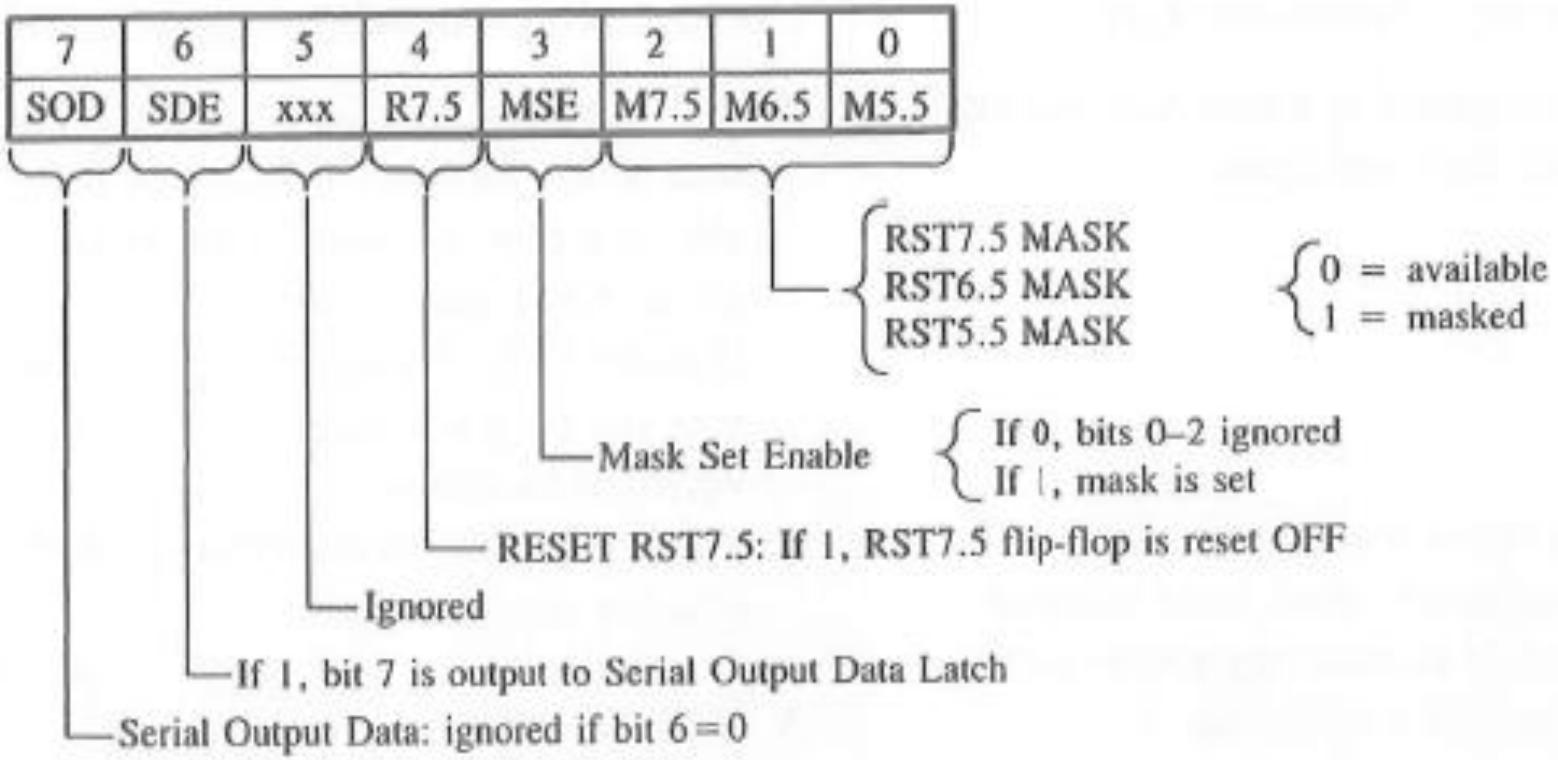


FIGURE
Interpretation of the Accumulator Bit Pattern for the SIM Instruction

SOURCE: Intel Corporation, *Assembly Language Programming Manual* (Santa Clara, Calif.: Author, 1979), pp. 3–59.

Example 1. Enable all the interrupts of Intel 8085.

The content of the accumulator for instructions SIM to enable RST 7.5, 6.5 and 5.5 are programmed as follows.

7	6	5	4	3	2	1	0
SOD	SOE	X	R7.5	MSE	M7.5	M6.5	M5.5
0	0	0	0	1	0	0	0 = 08

Bits 0, 1 and 2 are set to 0 to enable RST 7.5, 6.5 and 5.5. Bit 3 is set to make bits 0, 1 and 2 effective. Bit 4 is set to 0 to enable RST 7.5 as it is an additional control for RST 7.5.

Instructions

Mnemonic	*Operands*	*Comments*
EI		Enable all interrupts
MVI	A, 08	Get accumulator bit pattern to enable RST 7.5, 6.5 and 5.5
SIM		Enable RST 7.5, 6.5 and 5.5

Instruction EI and SIM both are essential to enable RST 7.5, 6.5 and 5.5. In addition to RST 7.5, 6.5 and 5.5 the instruction EI also enables INTR.

Example 2. Enable RST 6.5 and disable RST 7.5 and 5.5.

The contents of the accumulator to enable RST 6.5 and disable RST 7.5 and 5.5 are :

7	6	5	4	3	2	1	0
SOD	SOD	X	R7.5	MSE	M7.5	M6.5	M5.5
0	0	0	1	1	1	0	1 = 1D

Bit 1 is set to 0 to enable RST 6.5.

Bits 0 and 2 are set to 1 to mask off (disable) RST 5.5 and 7.5 respectively. Bit 4 which is an additic control for RST 7.5 is set to 1 to disable RST 7.5.

Bit 3 is set to 1 to make bits 0, 1 and 2 effective.

Instructions

EI	Enable interrupts
MVI A, 1D	Accumulator bit pattern to enable RST 6.5 and mask off RST 7.5 and 5.5.
SIM	Enable RST 6.5 and disable RST 7.5 and 5.5.

MODULE 3

LECTURE 1: ARCHITECTURE OF 8086

1. ARCHITECTURE OF 8086

Unlike microcontrollers, microprocessors do not have inbuilt memory. Mostly Princeton architecture is used for microprocessors where data and program memory are combined in a single memory interface. Since a microprocessor does not have any inbuilt peripheral, the circuit is purely digital and the clock speed can be anywhere from a few MHZ to a few hundred MHZ or even GHZ. This increased clock speed facilitates intensive computation that a microprocessor is supposed to do.
We will discuss the basic architecture of Intel 8086 before discussing more advanced microprocessor architectures.

Internal architecture of Intel 8086:

Intel 8086 is a 16 bit integer processor. It has 16-bit data bus and 20-bit address bus. The lower 16- bit address lines and 16-bit data lines are multiplexed (AD0-AD15). Since 20-bit address lines are available, 8086 can access up to 2 20 or 1 Giga byte of physical memory.

The basic architecture of 8086 is shown below.

The internal architecture of Intel 8086 is divided into two units, viz., Bus Interface Unit (BIU) and Execution Unit (EU).

Bus Interface Unit (BIU)

The Bus Interface Unit (BIU) generates the 20-bit physical memory address and provides the interface with external memory (ROM/RAM). As mentioned earlier, 8086 has a single memory interface. To speed up the execution, 6-bytes of instruction are fetched in advance and kept in a 6- byte Instruction Queue while other instructions are being executed in the Execution Unit (EU). Hence after the execution of an instruction, the next instruction is directly fetched from the instruction queue without having to wait for the external memory to send the instruction. This is called pipe-lining and is helpful for speeding up the overall execution process.

8086's BIU produces the 20-bit physical memory address by combining a 16-bit segment address with a 16-bit offset address. There are four 16-bit segment registers, viz., the code segment (CS), the stack segment (SS), the extra segment (ES), and the data segment (DS). These segment registers hold the corresponding 16-bit segment addresses. A segment address is the upper 16-bits of the starting address of that segment. The lower 4-bits of the starting address of a segment is always zero. The offset address is held by another 16-bit register. The physical 20-bit address is calculated by shifting the segment address 4-bit left and then adding that to the offset address.

For Example:

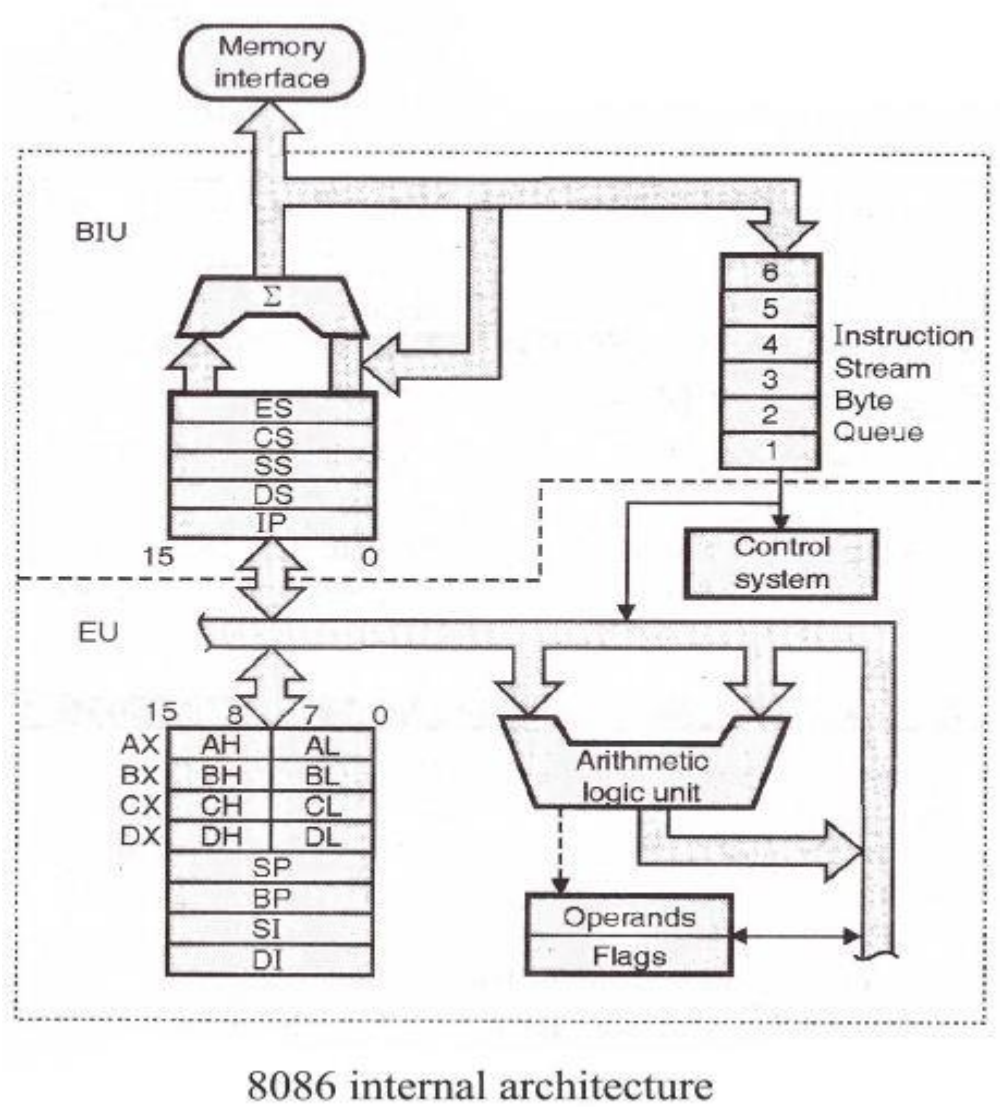


Figure 3.1: 8086 Architecture

Code segment Register CS holds the segment address which is 4569 H Instruction pointer IP holds the offset address which is 10A0 H The physical 20-bit address is calculated as follows

Segment address: 45690 H
Offset address:+ 10A0 H
Physical address: 46730 H

Most of the registers contain data/instruction offsets within 64 KB memory segment. There are four different 64 KB segments for instructions, stack, data and extra data. To specify where in 1 MB of processor memory these 4 segments are located the processor uses four segment

registers:

Code segment (CS) is a 16-bit register containing address of 64 KB segment with processor instructions. The processor uses CS segment for all accesses to instructions referenced by instruction pointer (IP) register. CS register cannot be changed directly. The CS register is automatically updated during far jump, far call and far return instructions.

Stack segment (SS) is a 16-bit register containing address of 64KB segment with program stack. By default, the processor assumes that all data referenced by the stack pointer (SP) and base pointer (BP) registers is located in the stack segment. SS register can be changed directly using POP instruction.

Data segment (DS) is a 16-bit register containing address of 64KB segment with program data. By default, the processor assumes that all data referenced by general registers (AX, BX, CX, DX) and index register (SI, DI) is located in the data segment. DS register can be changed directly using POP and LDS instructions.

Extra segment (ES) is a 16-bit register containing address of 64KB segment, usually with program data. By default, the processor assumes that the DI register references the ES segment in string manipulation instructions. ES register can be changed directly using POP and LES instructions.

It is possible to change default segments used by general and index registers by prefixing instructions with a CS, SS, DS or ES prefix.

Execution Unit:

All general registers of the 8086 microprocessor can be used for arithmetic and logic operations. The general registers are:

Accumulator register consists of 2 8-bit registers AL and AH, which can be combined together and used as a 16-bit register AX. AL in this case contains the low-order byte of the word, and AH contains the high-order byte. Accumulator can be used for I/O operations and string manipulation. **Base** register consists of 2 8-bit registers BL and BH, which can be combined together and used as a 16-bit register BX. BL in this case contains the low-order byte of the word, and BH contains the high-order byte. BX register usually contains a data pointer used for based, based indexed or register indirect addressing.

Count register consists of 2 8-bit registers CL and CH, which can be combined together and used as a 16-bit register CX. When combined, CL register contains the low-order byte of the word, and CH contains the high-order byte. Count register can be used as a counter in string manipulation and shift/rotate instructions.

Data register consists of 2 8-bit registers DL and DH, which can be combined together and used as a 16-bit register DX. When combined, DL register contains the low-order byte of the word, and DH contains the high-order byte. Data register can be used as a port number in I/O operations. In integer 32-bit multiply and divide instruction the DX register contains high-order word of the initial or resulting number.

The following registers are both general and index registers:

Stack Pointer (SP) is a 16-bit register pointing to program stack.

Base Pointer (BP) is a 16-bit register pointing to data in stack segment. BP register is usually used for based, based indexed or register indirect addressing.

Source Index (SI) is a 16-bit register. SI is used for indexed, based indexed and register indirect addressing, as well as a source data address in string manipulation instructions.

Destination Index (DI) is a 16-bit register. DI is used for indexed, based indexed and register indirect addressing, as well as a destination data address in string manipulation instructions.

Other registers:

Instruction Pointer (IP) is a 16-bit register.

Flags is a 16-bit register containing 9 1-bit flags:

- Overflow Flag (OF) - set if the result is too large positive number, or is too small negative number to fit into destination operand.
- Direction Flag (DF) - if set then string manipulation instructions will auto-decrement index registers. If cleared then the index registers will be auto-incremented.
- Interrupt-enable Flag (IF) - setting this bit enables maskable interrupts.
- Single-step Flag (TF) - if set then single-step interrupt will occur after the next instruction.
- Sign Flag (SF) - set if the most significant bit of the result is set.
    - Zero Flag (ZF) - set if the result is zero.
    - Auxiliary carry Flag (AF) - set if there was a carry from or borrow to bits 0-3 in the AL register.
    - Parity Flag (PF) - set if parity (the number of "1" bits) in the low-order byte of the result is even.
    - Carry Flag (CF) - set if there was a carry from or borrow to the most significant bit during last result calculation.

LECTURE 2: PIN DETAILS OF 8086

2. PIN DETAILS OF 8086

The following pin function descriptions are for 8086 systems in either minimum or maximum mode. The 'Local
Bus" in these descriptions is the direct multiplexed bus interface connection to the 8086 (without regard to additional bus buffers).

AD15±AD0 2±16, 39 -ADDRESS DATA BUS: These lines constitute the time multiplexed memory/IO address (T1), and data (T2, T3, TW, T4) bus. A0 is analogous to BHE for the lower byte of the data bus, pins D7±D0. It is LOW during T1 when a byte is to be transferred on the lower portion of the bus in memory or I/O operations. Eight-bit oriented devices tied to the lower half would normally use A0 to condition chip select functions. (See BHE.) These lines are active HIGH and float to 3-state OFF during interrupt acknowledge and local bus "hold acknowledge"

A19/S6,, A18/S5 , A17/S4, A16/S3 35±38 - ADDRESS/STATUS: During T1 these are the four most significant, address lines for memory operations. During I/O operations these, lines are LOW. During memory and I/O operations, status information is available on these lines during T2, T3, TW, T4. The status of the interrupt enable FLAG bit (S5) is updated at the beginning of each CLK cycle. A17/S4 and A16/S3 are encoded as shown. This information indicates which relocation register is presently being used for data accessing. These lines float to 3-state OFF during local bus "hold acknowledge."

BHE/S7 34 - BUS HIGH ENABLE/STATUS: During T1 the bus high enable signal (BHE) should be used to enable data onto the most significant half of the data bus, pins D15±D8. Eight-bit oriented devices tied to the upper half of the bus would normally use BHE to condition chip select functions. BHE is LOW during T1 for read, write, and interrupt acknowledge cycles when a byte is to be transferred on the high portion of the bus. The S7 status information is available during T2, T3, and T4. The signal is active LOW, and floats to 3-state OFF in hold". It is LOW during T1 for the first interrupt acknowledge cycle.

RD 32 READ: Read strobe indicates that the processor is performing a memory or I/O read cycle, depending on the state of the S2 pin. This signal is used to read devices which reside on the 8086 local bus. RD is active LOW during T2, T3 and TW of any read cycle, and is guaranteed to remain HIGH in T2 until the 8086 local bus has floated. This signal floats to 3-state OFF in 'hold acknowledge".

READY 22 - READY: is the acknowledgement from the addressed memory or I/O device that it will complete the data transfer. The READY signal from memory/IO is synchronized by the 8284A Clock Generator to form READY. This signal is active HIGH. The 8086 READY input is not synchronized. Correct operation is not guaranteed if the setup and hold times are not met.

INTR 18 - INTERRUPT REQUEST: is a level triggered input which is sampled during the last clock cycle of each instruction to determine if the processor should enter into an interrupt acknowledge operation. A subroutine is vectored to via an interrupt vector lookup table located in system memory. It can be internally masked by software resetting the interrupt enable bit. INTR is internally synchronized. This signal is active HIGH.

TEST 23 TEST: input is examined by the "Wait" instruction. If the TEST input is LOW execution continues, otherwise the processor waits in an "Idle" state. This input is synchronized internally during each clock cycle on the leading edge of CLK.

NMI 17 NON-MASKABLE INTERRUPT an edge triggered input which causes a type 2 interrupt. A subroutine is vectored to via an interrupt vector lookup table located in system memory. NMI is not maskable internally by software. A transition from LOW to HIGH initiates the interrupt at the end of the current instruction. This input is internally synchronized.

RESET 21 - RESET: causes the processor to immediately terminate its present activity. The signal must be active HIGH for at least four clock cycles. It restarts execution, as described in the Instruction Set description, when RESET returns LOW. RESET is internally synchronized.

CLK 19 CLOCK: provides the basic timing for the processor and bus controller. It is asymmetric with a 33% duty cycle to provide optimized internal timing.

VCC 40 VCC: a5V power supply pin**. ,**GND 1, 20 GROUND

MN/MX 33 I MINIMUM/MAXIMUM: indicates what mode the processor is to operate in. The two modes are discussed in the following sections.

Operating Modes of 8086

There are two modes of operation for Intel 8086, namely the minimum mode and the maximum mode. When only one 8086 CPU is to be used in a microcomputer system the 8086 is used in the minimum mode of operation. In this mode the CPU issues the control signals required by memory and I/O devices. In a multiprocessor system it operates in the maximum mode. In case of maximum mode of operation control signals are issued by Intel 8288 bus controller which is used with 8086 for this very purpose. The level of the pin MN/MX′ decides the operating mode of 8086. When MN/MX′ is high the CPU operates in the minimum mode. When it is low the CPU operates in the maximum mode. From pin 24 to 31 issue two different sets of signals. One set of signals is issued when the CPU operates in the maximum mode. Thus the pins from 24 to 31 have alternate functions.

Pin description for Minimum Mode:

For the minimum mode of operation the pin MN / MX′ is connected to 5 V D.C. supply, i.e., MN / MX′ = Vcc. The description of the pins from 24 to 31 for the minimum mode is as follows:

INTA′ (Output): Pin No. 24 Interrupt acknowledge. On receiving interrupt signal the processor issues an interrupt acknowledge signal. It is active LOW.

ALE (Output) Pin No. 25 Address latch enable. It goes High during T1. The microprocessor sends this signal to latch the address into the Intel 8282 / 8283 latch.

DEN′ (Output) Pin No. 26. Data enable. When Intel 8286 / 8287 octal bus transceiver is used this signal acts as an output enable signal. It is active low.

DT / R′ (Output) Pin No. 27. Data Transmit / Receive. When Intel 8286 / 8287 octal bus transceiver is used this signal controls the direction of data flow through the transceiver. When it is HIGH data are sent out. When it is LOW data are received.

M / IO′ (Output) Pin No. 28. Memory or I/O access. When it is HIGH the CPU wants to access memory. When it is LOW the CPU wants to access I/O device.

WR′ (Output): Pin No. 29 Write. When it is LOW the CPU performs memory or I/O write operation.
HLDA(Output): Pin No. 30. HOLD acknowledge. It is issued by the processor when it receives HOLD signal. It is active HIGH signal. When HOLD request is removed HLDA goes LOW.

HOLD (Input) Pin No. 31. Hold. When another device in microcomputer system system wants to use the address and data bus, it sends a HOLD request to CPU through this pin. It is an active HIGH signal.

Pin description for Maximum Mode:

For the maximum mode of operation the pin MN / MX′ is connected to Ground, i.e., MN / MX′ = Vcc. The description of the pins from 24 to 31 for the maximum mode is as follows:

QS1, QS0 (Output) : Pin No. 24, 25. Instruction Queue Status. Logics are given below:

QS1	QS2	Function
0	0	No Operation
0	1	1st byte of opcode from queue
1	0	Empty the queue
1	1	Subsequent byte from queue

Table 4.1: Functions of QS1 and QS2

S2, S1, S0 (Output) 26±28 STATUS: active during T4, T1, and T2 and is returned to the

passive state (1, 1, 1) during T3 or during TW when READY is HIGH. This status is used by the 8288 Bus Controller to generate all memory and I/O access control signals. Any change by S2, S1, or S0 during T4 is used to indicate the beginning of a bus cycle, and the return to the passive state in T3 or TW is used to indicate the end of a bus cycle

S2	S1	S0	Function
0	0	0	Interrupt Acknowledge
0	0	1	Read I/O Port
0	1	0	Write I/O Port
0	1	1	Halt
1	0	0	Code access
1	0	1	Write memory
1	1	0	Passive state
1	1	1	

Table 4.1: Functions of S2, S1 and S0

LOCK (O)

It indicates to another system bus master, not to gain control of the system bus while LOCK is active Low. The LOCK signal is activated by the "LOCK" prefix instruction and remains active until the completion of the instruction. This signal is active Low and floats to tri-state OFF during 'hold acknowledge'.

RQ/GT0 and RQ/GT1 (I/O): Request/Grant

These pins are used by other processors in a multi processor organization. Local bus masters of other processors force the processor to release the local bus at the end of the processors current bus cycle. Each pin is bi-directional and has an internal pull up resistors. Hence they may be left un-connected.

LECTURE 3: ADDRESSING MODES OF 8086

3. ADDRESSING MODES

Addressing mode indicates a way of locating data or operands. Depending upon the data types used in the instruction and the memory addressing modes, any instruction may belong to one or more addressing modes, or some instruction may not belong to any of the addressing modes. Thus the addressing modes describe the types of operands and the way they are accessed for executing an instruction. Here, we will present the addressing modes of the instructions depending upon their types. According to the flow of instruction execution, the instructions may be categorized as

i. Sequential control flow instructions and
ii. Control transfer instructions

Sequential control flow instructions are the instructions, which after execution, transfer control to the next instruction appearing immediately after it (in the sequence) in the program. For example, the arithmetic, logical, data transfer and processor control instructions are sequential control flow instructions. The control transfer instructions, on the other hand, transfer control to some predefined address somehow specified in the instruction after their execution. For example, INT, CALL, RET and JUMP instructions fall under this category.
The addressing modes for sequential control transfer instructions are explained as follows:

1. Immediate: In this type of addressing, immediate data is a part of instruction, and appears in the form of successive byte or bytes.

   Example: MOV AX, 0005H

   In the above example, 0005H is the immediate data. The immediate data may be 8-bit or 16-bit in size.

2. Direct: In the direct addressing mode, a 16-bit memory address (offset) is directly specified in the instruction as a part of it.

   Example: MOV AX, [5000H]

   Here, data resides in a memory location in the data segment, whose effective address may be computed using 5000H as the offset address and content of DS as segment address. The effective address, here, is 10H*DS+5000H.

3. Register: In the direct addressing mode, the data is stored in a register and it is referred using the particular register. All the registers, except IP, may be used in this mode.

   Example: MOV BX, AX
4. Register Indirect: Sometimes, the address of the memory location, which contains data or operand, is determined in an indirect way, using the offset registers. This mode of

addressing is known as register indirect mode. In this addressing mode, the offset address of data is in either BX or SI or DI registers. The default segment is either DS or ES. The data is supposed to be available at the address pointed to by the content of any of the above registers in the default data segment.

Example: MOV AX, [BX]

Here, data is present in a memory location in DS whose offset address is in BX. The effective address of the data is given as 10H*DS+ [BX].

5. Indexed: In this addressing mode, offset of the operand is stored in one of the index registers. DS and ES are the default segments for index registers SI and DI respectively. This mode is a special case of the above discussed register indirect addressing mode.

   Example: MOV AX,[SI]

   Here, data is available at an offset address stored in SI in DS. The effective address, in this case, is computed as 10H*DS+ [SI].

6. Register Relative: In this addressing mode, the data is available at an effective address formed by adding an 8-bit or 16-bit displacement with the content of any one of the registers BX, BP, SI and DI in the default (either DS or ES) segment. The example given before explains this mode.

   Example: MOV AX, 50H [BX]

   Here, effective address is given as 10H*DS+50H+ [BX].

7. Based Indexed: The effective address of data is formed, in this addressing mode, by adding content of a base register (any one of BX or BP) to the content of an index register (any one of SI or DI). The default segment register may be ES or DS.

   Example: MOV AX, [BX] [SI]

   Here, BX is the base register and SI is the index register. The effective address is computed as 10H*DS+ [BX] + [SI].

8. Relative Based Indexed: The effective address is formed by adding an 8-bit or 16-bit displacement with the sum of contents of any one of the bases registers (BX or BP) and any one of the index registers, in a default segment.
   Example: MOV AX, 50H [BX] [SI]

   Here, 50H is an immediate displacement, BX is a base register and SI is an index register. The effective address of data is computed as 160H*DS+ [BX]+ [SI] + 50H.

LECTURE 4: 8086 INSTRUCTION SET

4. 8086 INSTRUCTION SET

The 8086 microprocessor supports 8 types of instructions −

- Data Transfer Instructions
- Arithmetic Instructions
- Bit Manipulation Instructions
- String Instructions
- Program Execution Transfer Instructions (Branch & Loop Instructions)
- Processor Control Instructions
- Iteration Control Instructions
- Interrupt Instructions

Let us now discuss these instruction sets in detail.

Data Transfer Instructions

These instructions are used to transfer the data from the source operand to the destination operand. Following are the list of instructions under this group −

Instruction to transfer a word

- **MOV** − Used to copy the byte or word from the provided source to the provided destination.
- **PPUSH** − Used to put a word at the top of the stack.
- **POP** − Used to get a word from the top of the stack to the provided location.
- **PUSHA** − Used to put all the registers into the stack.
- **POPA** − Used to get words from the stack to all registers.
- **XCHG** − Used to exchange the data from two locations.
- **XLAT** − Used to translate a byte in AL using a table in the memory.

Instructions for input and output port transfer

- **IN** − Used to read a byte or word from the provided port to the accumulator.
- **OUT** − Used to send out a byte or word from the accumulator to the provided port.

Instructions to transfer the address

- **LEA** − Used to load the address of operand into the provided register.
- **LDS** − Used to load DS register and other provided register from the memory
- **LES** − Used to load ES register and other provided register from the memory.

Instructions to transfer flag registers

- **LAHF** − Used to load AH with the low byte of the flag register.
- **SAHF** − Used to store AH register to low byte of the flag register.
- **PUSHF** − Used to copy the flag register at the top of the stack.
- **POPF** − Used to copy a word at the top of the stack to the flag register.

Arithmetic Instructions

These instructions are used to perform arithmetic operations like addition, subtraction, multiplication, division, etc.

Following is the list of instructions under this group −

Instructions to perform addition

- **ADD** − Used to add the provided byte to byte/word to word.
- **ADC** − Used to add with carry.
- **INC** − Used to increment the provided byte/word by 1.
- **AAA** − Used to adjust ASCII after addition.
- **DAA** − Used to adjust the decimal after the addition/subtraction operation.

Instructions to perform subtraction

- **SUB** − Used to subtract the byte from byte/word from word.
- **SBB** − Used to perform subtraction with borrow.
- **DEC** − Used to decrement the provided byte/word by 1.
- **NPG** − Used to negate each bit of the provided byte/word and add 1/2's complement.
- **CMP** − Used to compare 2 provided byte/word.
- **AAS** − Used to adjust ASCII codes after subtraction.
- **DAS** − Used to adjust decimal after subtraction.

Instruction to perform multiplication

- **MUL** − Used to multiply unsigned byte by byte/word by word.
- **IMUL** − Used to multiply signed byte by byte/word by word.
- **AAM** − Used to adjust ASCII codes after multiplication.

Instructions to perform division

- **DIV** − Used to divide the unsigned word by byte or unsigned double word by word.
- **IDIV** − Used to divide the signed word by byte or signed double word by word.
- **AAD** − Used to adjust ASCII codes after division.
- **CBW** − Used to fill the upper byte of the word with the copies of sign bit of the lower byte.
- **CWD** − Used to fill the upper word of the double word with the sign bit of the lower word.

Bit Manipulation Instructions

These instructions are used to perform operations where data bits are involved, i.e. operations like logical, shift, etc.

Following is the list of instructions under this group −

Instructions to perform logical operation

- **NOT** − Used to invert each bit of a byte or word.
- **AND** − Used for adding each bit in a byte/word with the corresponding bit in another byte/word.
- **OR** − Used to multiply each bit in a byte/word with the corresponding bit in another byte/word.
- **XOR** − Used to perform Exclusive-OR operation over each bit in a byte/word with the corresponding bit in another byte/word.
- **TEST** − Used to add operands to update flags, without affecting operands.

Instructions to perform shift operations

- **SHL/SAL** − Used to shift bits of a byte/word towards left and put zero(S) in LSBs.
- **SHR** − Used to shift bits of a byte/word towards the right and put zero(S) in MSBs.
- **SAR** − Used to shift bits of a byte/word towards the right and copy the old MSB into the new MSB.

Instructions to perform rotate operations

- **ROL** − Used to rotate bits of byte/word towards the left, i.e. MSB to LSB and to Carry Flag [CF].
- **ROR** − Used to rotate bits of byte/word towards the right, i.e. LSB to MSB and to Carry Flag [CF].
- **RCR** − Used to rotate bits of byte/word towards the right, i.e. LSB to CF and CF to MSB.
- **RCL** − Used to rotate bits of byte/word towards the left, i.e. MSB to CF and CF to LSB.

String Instructions

String is a group of bytes/words and their memory is always allocated in a sequential order.
Following is the list of instructions under this group −

- **REP** − Used to repeat the given instruction till CX ≠ 0.
- **REPE/REPZ** − Used to repeat the given instruction until CX = 0 or zero flag ZF = 1.
- **REPNE/REPNZ** − Used to repeat the given instruction until CX = 0 or zero flag ZF = 1.
- **MOVS/MOVSB/MOVSW** − Used to move the byte/word from one string to another.
- **COMS/COMPSB/COMPSW** − Used to compare two string bytes/words.
- **INS/INSB/INSW** − Used as an input string/byte/word from the I/O port to the provided memory location.
- **OUTS/OUTSB/OUTSW** − Used as an output string/byte/word from the provided memory location to the I/O port.
- **SCAS/SCASB/SCASW** − Used to scan a string and compare its byte with a byte in AL or string word with a word in AX.
- **LODS/LODSB/LODSW** − Used to store the string byte into AL or string word into AX.

Program Execution Transfer Instructions (Branch and Loop Instructions)

These instructions are used to transfer/branch the instructions during an execution. It includes the following instructions −

Instructions to transfer the instruction during an execution without any condition −

- **CALL** − Used to call a procedure and save their return address to the stack.
- **RET** − Used to return from the procedure to the main program.
- **JMP** − Used to jump to the provided address to proceed to the next instruction.

Instructions to transfer the instruction during an execution with some conditions −

- **JA/JNBE** − Used to jump if above/not below/equal instruction satisfies.
- **JAE/JNB** − Used to jump if above/not below instruction satisfies.
- **JBE/JNA** − Used to jump if below/equal/ not above instruction satisfies.
- **JC** − Used to jump if carry flag CF = 1
- **JE/JZ** − Used to jump if equal/zero flag ZF = 1
- **JG/JNLE** − Used to jump if greater/not less than/equal instruction satisfies.
- **JGE/JNL** − Used to jump if greater than/equal/not less than instruction satisfies.

- **JL/JNGE** − Used to jump if less than/not greater than/equal instruction satisfies.
- **JLE/JNG** − Used to jump if less than/equal/if not greater than instruction satisfies.
- **JNC** − Used to jump if no carry flag (CF = 0)
- **JNE/JNZ** − Used to jump if not equal/zero flag ZF = 0
- **JNO** − Used to jump if no overflow flag OF = 0
- **JNP/JPO** − Used to jump if not parity/parity odd PF = 0
- **JNS** − Used to jump if not sign SF = 0
- **JO** − Used to jump if overflow flag OF = 1
- **JP/JPE** − Used to jump if parity/parity even PF = 1
- **JS** − Used to jump if sign flag SF = 1

Processor Control Instructions

These instructions are used to control the processor action by setting/resetting the flag values. Following are the instructions under this group −

- **STC** − Used to set carry flag CF to 1
- **CLC** − Used to clear/reset carry flag CF to 0
- **CMC** − Used to put complement at the state of carry flag CF.
- **STD** − Used to set the direction flag DF to 1
- **CLD** − Used to clear/reset the direction flag DF to 0
- **STI** − Used to set the interrupt enable flag to 1, i.e., enable INTR input.
- **CLI** − Used to clear the interrupt enable flag to 0, i.e., disable INTR input.

Iteration Control Instructions

These instructions are used to execute the given instructions for number of times. Following is the list of instructions under this group −

- **LOOP** − Used to loop a group of instructions until the condition satisfies, i.e., CX = 0
- **LOOPE/LOOPZ** − Used to loop a group of instructions till it satisfies ZF = 1 & CX = 0
- **LOOPNE/LOOPNZ** − Used to loop a group of instructions till it satisfies ZF = 0 & CX = 0
- **JCXZ** − Used to jump to the provided address if CX = 0

Interrupt Instructions

These instructions are used to call the interrupt during program execution.

- **INT** − Used to interrupt the program during execution and calling service specified.
- **INTO** − Used to interrupt the program during execution if OF = 1
- **IRET** − Used to return from interrupt service to the main program

Sample Assembly Language Program:

1. **Write an assembly language program in 8086 to add two 16-bit numbers.**

PROGRAM:

LABEL	**OPCODE**	**OPERAND**	**COMMENTS**
START	MOV	CX, 9273	Get 16-bit data in AX
	MOV	DX, 2464	Get another 16-bit data in DX
	ADD	CX, DX	(CX) ← (CX) + (DX)
END	INT3		Halt the program

MODULE 5

LECTURE 1: ARCHITECTURE OF 8051

1. 8051 MICROCONTROLLER

The 8051 Microcontroller was designed in 1980's by Intel. Its foundation was on Harvard Architecture and was developed principally for bringing into play in Embedded Systems. At first it was created by means of NMOS technology but as NMOS technology needs more power to function therefore Intel re-intended Microcontroller 8051 employing CMOS technology and a new edition came into existence with a letter 'C' in the title name, for illustration: 80C51. These most modern Microcontrollers need fewer amount of power to function in comparison to their forerunners.

There are two buses in 8051 Microcontroller one for program and other for data. As a result, it has two storage rooms for both program and data of 64K by 8 size. The microcontroller comprise of 8 bit accumulator & 8 bit processing unit. It also consists of 8 bit B register as majorly functioning blocks and 8051 microcontroller programming is done with embedded C Language using Keil software. It also has a number of other 8 bit and 16 bit registers.

For internal functioning & processing Microcontroller 8051 comes with integrated built-in RAM. This is prime memory and is employed for storing temporary data. It is unpredictable memory i.e. its data can get be lost when the power supply to the Microcontroller switched OFF.

1.1. 8051 MICROCONTROLLER ARCHITECTURE

Microcontroller 8051 block diagram is shown below. Let's have a closer look on features of 8051 microcontroller design:

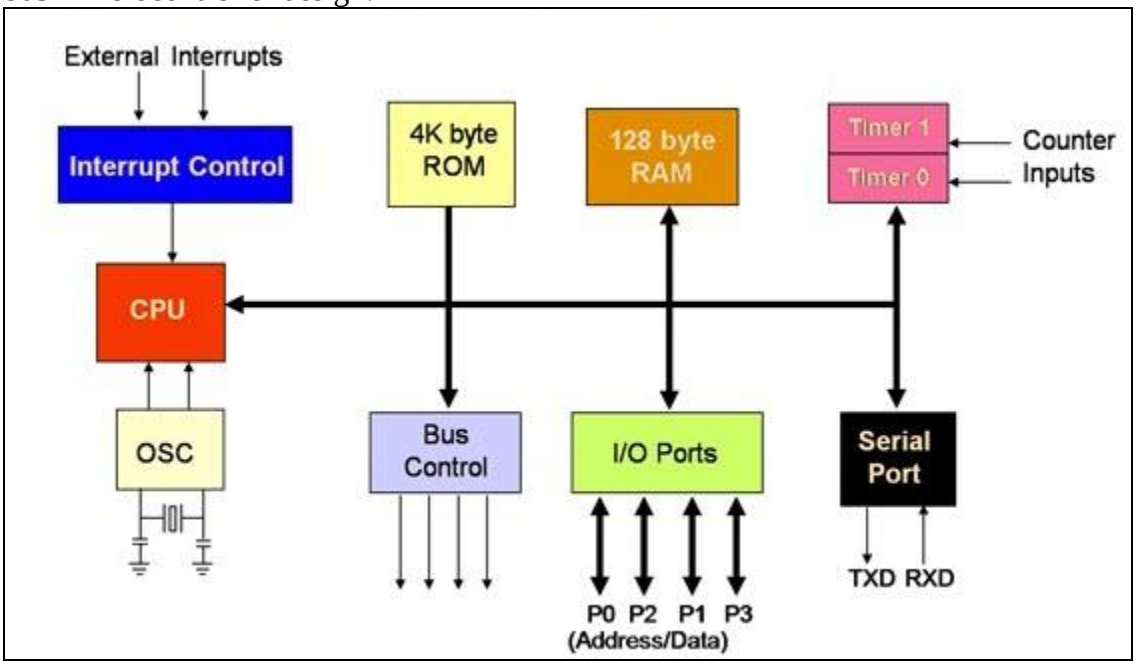


Figure 4.1: Block Diagram of 8051 Microcontroller

CPU (Central Processor Unit):
As we may be familiar that Central Processor Unit or CPU is the mind of any processing machine. It scrutinizes and manages all processes that are carried out in the Microcontroller. User has no power over the functioning of CPU. It interprets program printed in storage space (ROM) and carries out all of them and do the projected duty. CPU manages different types of registers in 8051 microcontroller.

Interrupts:
As the heading put forward, Interrupt is a sub-routine call that reads the Microcontroller's key function or job and helps it to perform some other program which is extra important at that point of time. The characteristic of 8051 Interrupt is extremely constructive as it aids in emergency cases. Interrupts provides us a method to postpone or delay the current process, carry out a sub-routine task and then all over again restart standard program implementation.
The Micro-controller 8051 can be assembled in such a manner that it momentarily stops or break the core program at the happening of interrupt. When sub-routine task is finished then the implementation of core program initiates automatically as usual. There are 5 interrupt supplies in 8051 Microcontroller, two out of five are peripheral interrupts, two are timer interrupts and one is serial port interrupt.

Memory:
Micro-controller needs a program which is a set of commands. This program enlightens Microcontroller to perform precise tasks. These programs need a storage space on which they can be accumulated and interpret by Microcontroller to act upon any specific process. The memory which is brought into play to accumulate the program of Microcontroller is recognized as Program memory or code memory. In common language it's also known as Read Only Memory or ROM.
Microcontroller also needs a memory to amass data or operands for the short term. The storage space which is employed to momentarily data storage for functioning is acknowledged as Data Memory and we employ Random Access Memory or RAM for this principle reason. Microcontroller 8051 contains code memory or program memory 4K so that is has 4KB Rom and it also comprise of data memory (RAM) of 128 bytes.

Bus:
Fundamentally Bus is a group of wires which functions as a communication canal or mean for the transfer Data. These buses comprise of 8, 16 or more cables. As a result, a bus can bear 8 bits, 16 bits all together. There are two types of buses:

1. **Address Bus:** Microcontroller 8051 consists of 16 bit address bus. It is brought into play to address memory positions. It is also utilized to transmit the address from Central Processing Unit to Memory.
2. **Data Bus:** Microcontroller 8051 comprise of 8 bits data bus. It is employed to cart data.

Oscillator:
As we all make out that Microcontroller is a digital circuit piece of equipment, thus it needs timer for its function. For this function, Microcontroller 8051 consists of an on-chip oscillator which toils as a time source for CPU (Central Processing Unit). As the productivity thumps of

oscillator are steady as a result, it facilitates harmonized employment of all pieces of 8051 Microcontroller. Input/output Port: As we are acquainted with that Microcontroller is employed in embedded systems to manage the functions of devices.
Thus to gather it to other machinery, gadgets or peripherals we need I/O (input/output) interfacing ports in Micro-controller. For this function Micro-controller 8051 consists of 4 input/output ports to unite it to other peripherals. Timers / Counters: Micro-controller 8051 is incorporated with two 16 bit counters & timers. The counters are separated into 8 bit registers. The timers are utilized for measuring the intervals, to find out pulse width etc.

1.2. 8051 MICROCONTROLLER MEMORY ORGANIZATION

The 8051 Microcontroller Memory is separated in Program Memory (ROM) and Data Memory (RAM). The Program Memory of the 8051 Microcontroller is used for storing the program to be executed i.e. instructions. The Data Memory on the other hand, is used for storing temporary variable data and intermediate results.

Program Memory (ROM) of 8051 Microcontroller

In 8051 Microcontroller, the code or instructions to be executed are stored in the Program Memory, which is also called as the ROM of the Microcontroller. The original 8051 Microcontroller by Intel has 4KB of internal ROM.
Some variants of 8051 like the 8031 and 8032 series doesn't have any internal ROM (Program Memory) and must be interfaced with external Program Memory with instructions loaded in it.
Almost all modern 8051 Microcontrollers, like 8052 Series, have 8KB of Internal Program Memory (ROM) in the form of Flash Memory (ROM) and provide the option of reprogramming the memory.

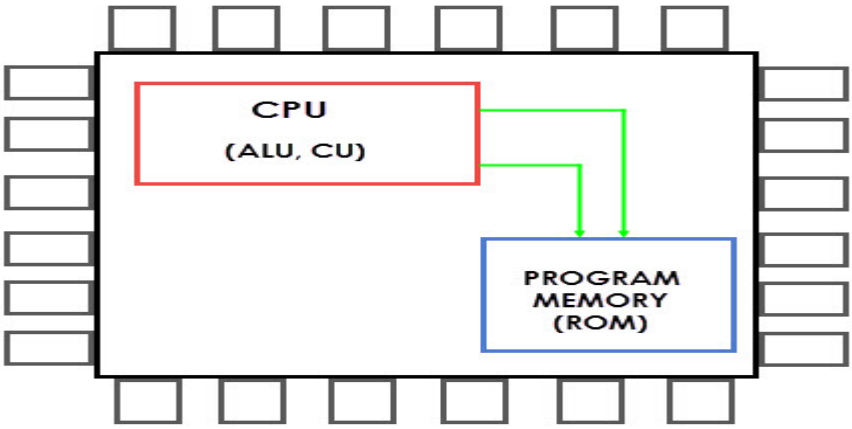


Figure 4.2a

In case of 4KB of Internal ROM, the address space is 0000H to 0FFFH. If the address space i.e. the program addresses exceed this value, then the CPU will automatically fetch the code from the external Program Memory.

For this, the External Access Pin (EA Pin) must be pulled HIGH i.e. when the EA Pin is high, the CPU first fetches instructions from the Internal Program Memory in the address range of

0000H to 0FFFFH and if the memory addresses exceed the limit, then the instructions are fetched from the external ROM in the address range of 1000H to FFFFH.

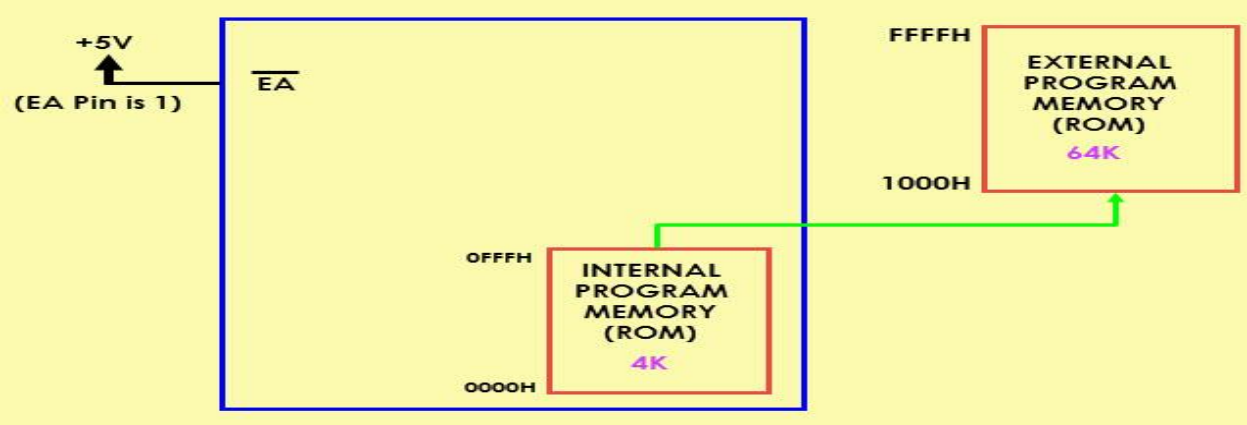


Figure 4.2b

There is another way to fetch the instructions: ignore the Internal ROM and fetch all the instructions only from the External Program Memory (External ROM). For this scenario, the EA Pin must be connected to GND. In this case, the memory addresses of the external ROM will be from 0000H to FFFFH.

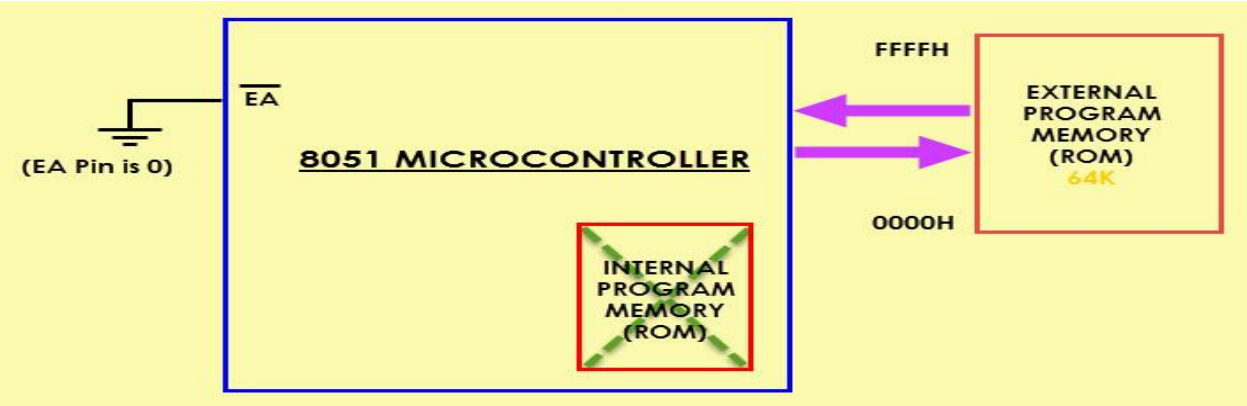


Figure 4.2c

Data Memory (RAM) of 8051 Microcontroller

The Data Memory or RAM of the 8051 Microcontroller stores temporary data and intermediate results that are generated and used during the normal operation of the microcontroller. Original Intel's 8051 Microcontroller had 128B of internal RAM.
But almost all modern variants of 8051 Microcontroller have 256B of RAM. In this 256B, the first 128B i.e. memory addresses from 00H to 7FH is divided in to Working Registers (organized as Register Banks), Bit – Addressable Area and General Purpose RAM (also known as Scratchpad area).
In the first 128B of RAM (from 00H to 7FH), the first 32B i.e. memory from addresses 00H to

1FH consists of 32 Working Registers that are organized as four banks with 8 Registers in each Bank.

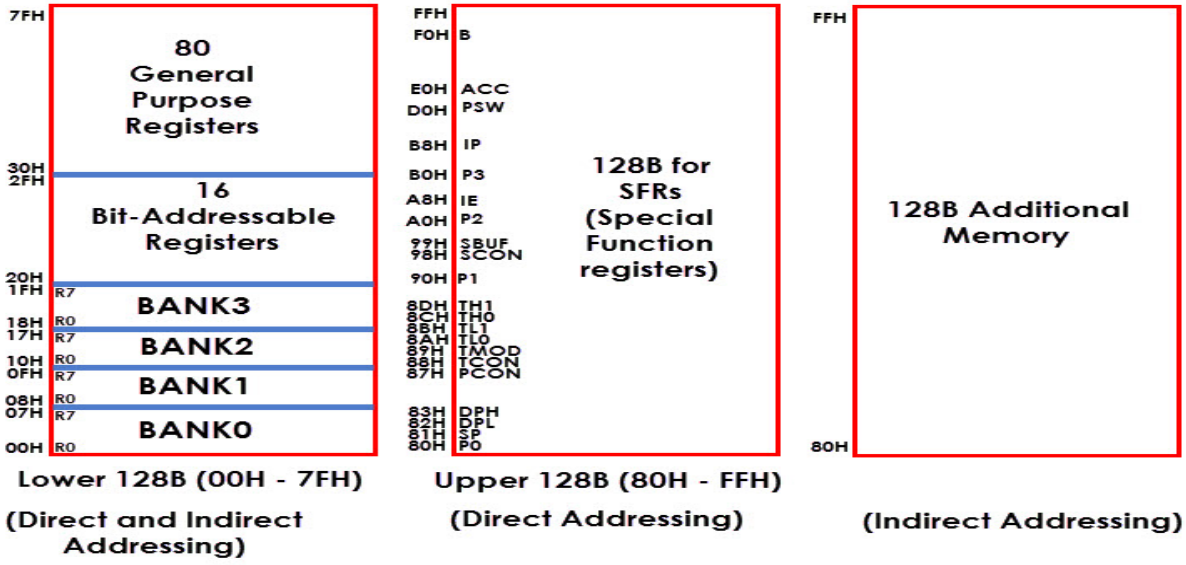


Figure 4.3: Data Memory of 8051

The 4 banks are named as Bank0, Bank1, Bank2 and Bank3. Each Bank consists of 8 registers named as R0 – R7. Each Register can be addressed in two ways: either by name or by address.
To address the register by name, first the corresponding Bank must be selected. In order to select the bank, we have to use the RS0 and RS1 bits of the Program Status Word (PSW) Register (RS0 and RS1 are 3rd and 4th bits in the PSW Register).
When addressing the Register using its address i.e. 12H for example, the corresponding Bank may or may not be selected. (12H corresponds to R2 in Bank2).
The next 16B of the RAM i.e. from 20H to 2FH are Bit – Addressable memory locations. There are totally 128 bits that can be addressed individually using 00H to 7FH or the entire byte can be addressed as 20H to 2FH.
For example 32H is the bit 2 of the internal RAM location 26H.
The final 80B of the internal RAM i.e. addresses from 30H to 7FH, is the general purpose RAM area which are byte addressable.
These lower 128B of RAM can be addressed directly or indirectly.
The upper 128B of the RAM i.e. memory addresses from 80H to FFH is allocated for Special Function Registers (SFRs). SFRs control specific functions of the 8051 Microcontroller. Some of the SFRs are I/O Port Registers (P0, P1, P2 and P3), PSW (Program Status Word), A (Accumulator), IE (Interrupt Enable), PCON (Power Control), etc.

Name of the Register	*Function*	*Internal RAM Address (HEX)*
ACC	Accumulator	E0H
B	B Register (for Arithmetic)	F0H
DPH	Addressing External Memory	83H
DPL	Addressing External Memory	82H
IE	Interrupt Enable Control	A8H
IP	Interrupt Priority	B8H
P0	PORT 0 Latch	80H
P1	PORT 1 Latch	90H
P2	PORT 2 Latch	A0H
P3	PORT 3 Latch	B0H
PCON	Power Control	87H
PSW	Program Status Word	D0H
SCON	Serial Port Control	98H
SBUF	Serial Port Data Buffer	99H
SP	Stack Pointer	81H
TMOD	Timer / Counter Mode Control	89H
TCON	Timer / Counter Control	88H
TL0	Timer 0 LOW Byte	8AH
TH0	Timer 0 HIGH Byte	8CH
TL1	Timer 1 LOW Byte	8BH
TH1	Timer 1 HIGH Byte	8DH

Table 4.1

SRFs Memory addresses are only direct addressable. Even though some of the addresses between 80H and FFH are not assigned to any SFR, they cannot be used as additional RAM area. In some microcontrollers, there is an additional 128B of RAM, which share the memory address with SFRs i.e. 80H to FFH. But, this additional RAM block is only accessed by indirect addressing.

LECTURE 2: PIN DESCRIPTION OF 8051

2. 8051 MICROCONTROLLER PIN DIAGRAM

As mentioned in the previous tutorial, 8051 Microcontroller is available in a variety of packages like 40 – pin DIP or 44 – lead PLCC and TQFP. The pin orientation of an 8051 Microcontroller may change with the package but the Pin Configuration is same.
The following image shows the 8051 Microcontroller Pin Diagram with respect to a 40 – pin Dual In-line Package (DIP).

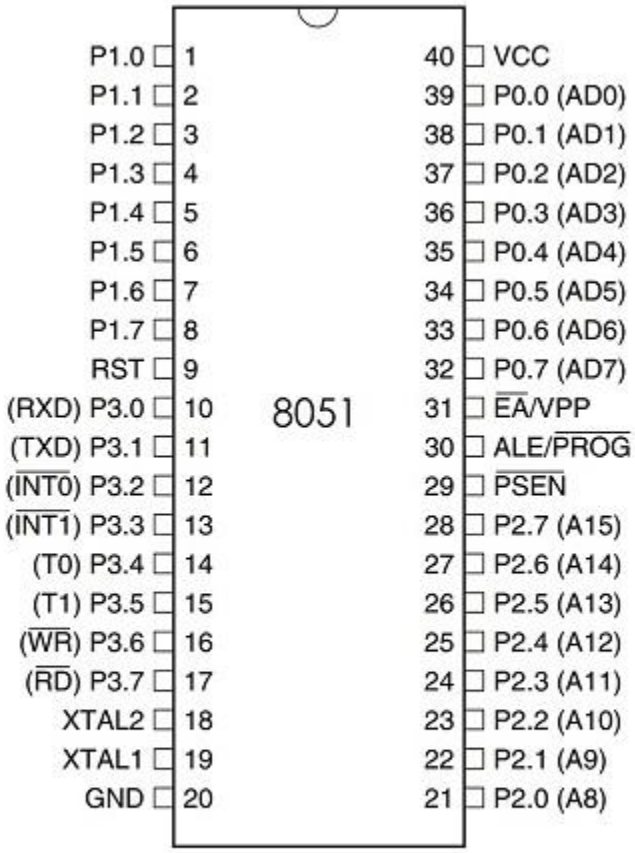


Figure 4.4: Pin Diagram of 8051 Microcontroller

Since it is a 40 – pin DIP IC, each side contains 20 Pins. We have also seen that there other packages of 8051 like the 44 – Lead PLCC and the 44 – Lead TQFP. The following image shows the 8051 Microcontroller Pin Diagram for these packages specifically.

The Pin Description or Pin Configuration of the 8051 Microcontroller will describe the functions of each pins of the 8051 Microcontroller. Let us now see the pin description.

Pins 1 – 8 (PORT 1): Pins 1 to 8 are the PORT 1 Pins of 8051. PORT 1 Pins consists of 8 – bit bidirectional Input / Output Port with internal pull – up resistors. In older 8051 Microcontrollers, PORT 1 doesn't serve any additional purpose but just 8 – bit I/O PORT.

In some of the newer 8051 Microcontrollers, few PORT 1 Pins have dual functions. P1.0 and P1.1 act as Timer 2 and Timer 2 Trigger Input respectively.

P1.5, P1.6 and P1.7 act as In-System Programming Pins i.e. MOSI, MISO and SCK respectively.

Pin 9 (RST): Pin 9 is the Reset Input Pin. It is an active HIGH Pin i.e. if the RST Pin is HIGH for a minimum of two machine cycles, the microcontroller will be reset. During this time, the oscillator must be running.

Pins 10 – 17 (PORT 3): Pins 10 to 17 form the PORT 3 pins of the 8051 Microcontroller. PORT 3 also acts as a bidirectional Input / Output PORT with internal pull-ups. Additionally, all the PORT 3 Pins have special functions. The following table gives the details of the additional functions of PORT 3 Pins.

PORT 3 Pin	Function	Description
P3.0	RXD	Serial Input
P3.1	TXD	Serial Output
P3.2	INT0	External Interrupt 0
P3.3	INT1	External Interrupt 1
P3.4	T0	Timer 0
P3.5	T1	Timer 1
P3.6	WR	External Memory Write
P3.7	RD	External Memory Read

Table 4.2

Pins 18 & 19: Pins 18 and 19 i.e. XTAL 2 and XTAL 1 are the pins for connecting external oscillator. Generally, a Quartz Crystal Oscillator is connected here.

Pin 20 (GND): Pin 20 is the Ground Pin of the 8051 Microcontroller. It represents 0V and is connected to the negative terminal (0V) of the Power Supply.

Pins 21 – 28 (PORT 2): These are the PORT 2 Pins of the 8051 Microcontroller. PORT 2 is also a Bidirectional Port i.e. all the PORT 2 pins act as Input or Output. Additionally, when external memory is interfaced, PORT 2 pins act as the higher order address byte. PORT 2 Pins have internal pull-ups.

Pin 29 (PSEN): Pin 29 is the Program Store Enable Pin (PSEN). Using this pins, external Program Memory can be read.

Pin 30 (ALE/PROG): Pin 30 is the Address Latch Enable Pin. Using this Pins, external address can be separated from data (as they are multiplexed by 8051).

During Flash Programming, this pin acts as program pulse input (PROG).

Pin 31 (EA/VPP): Pin 31 is the External Access Enable Pin i.e. allows external Program Memory. Code from external program memory can be fetched only if this pin is LOW. For normal operations, this pins is pulled HIGH.

During Flash Programming, this Pin receives 12V Programming Enable Voltage (VPP).
Pins 32 – 39 (PORT 0): Pins 32 to 39 are PORT 0 Pins. They are also bidirectional Input / Output Pins but without any internal pull-ups. Hence, we need external pull-ups in order to use PORT 0 pins as I/O PORT.
In addition to acting as I/O PORT, PORT 0 also acts as lower order address/data bus when external memory is accessed.
Pin 40 (VCC): Pin 40 is the power supply pin to which the supply voltage is given (+5V).

8051 Microcontroller Basic Circuit

Now that we have seen the 8051 Microcontroller Pin Diagram and corresponding Pin Description, we will proceed to the basic circuit or schematic of the 8051 Microcontroller. The following image shows the basic circuit of the 8051 Microcontroller.

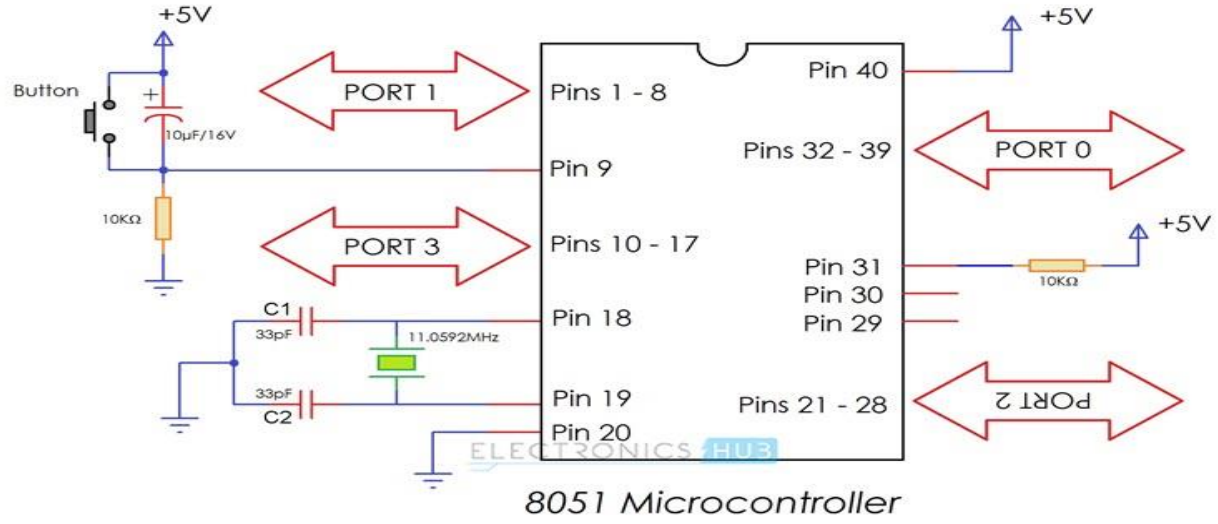


Figure 4.5: 8051 Microcontroller Basic Circuit

his basic circuit of 8051 microcontroller is the minimal interface required for it to work. The basic circuit includes a Reset Circuit, the oscillator circuit and power supply. Let us discuss a little bit deeper about this basic circuit of 8051 Microcontroller.
First is the power supply. Pins 40 and 20 (VCC and GND) of the 8051 Microcontroller are connected to +5V and GND respectively.
Next is the Reset Circuit. A logic HIGH (+5V) on Reset Pin for a minimum of two machine cycles (24 clock cycles) will reset the 8051 Microcontroller. The reset circuit of the 8051 Microcontroller consists of a capacitor, a resistor and a push button and this type of reset circuit provides a Manual Reset Option. If you remove the push button, then the reset circuit becomes a Power-On Reset Circuit.
The next part of the basic circuit of the 8051 Microcontroller is the Oscillator Circuit or the Clock Circuit. A Quartz Crystal Oscillator is connected across XTAL1 and XTAL2 pins i.e. Pins 19 and 18. The capacitors C1 and C2 can be selected in the range of 20pF to 40pF.

As mentioned in the 8051 Microcontroller Pin Description, PORTS 1, 2 and 3, all have internal pull – ups and hence can be directly used as Bidirectional I/O Ports. But, we need to add external Pull – ups for PORT 0 Pins in order to use it as an I/O Port.
Generally, a 1KΩ Resistor Pack of 8 Resistors is used as a Pull – up for the PORT 0 of the 8051 Microcontroller.

LECTURE 3: INSTRUCTION SET OF 8051

3. 8051 MICROCONTROLLER INSTRUCTION SET

Writing a Program for any Microcontroller consists of giving commands to the Microcontroller in a particular order in which they must be executed in order to perform a specific task. The commands to the Microcontroller are known as a Microcontroller's Instruction Set.
Just as our sentences are made of words, a Microcontroller's (for that matter, any computer) program is made of Instructions. Instructions written in a program tell the Microcontroller which operation to carry out.
An Instruction Set is unique to a family of computers. This tutorial introduces the 8051 Microcontroller Instruction Set also called as the MCS-51 Instruction Set.
As the 8051 family of Microcontrollers are 8-bit processors, the 8051 Microcontroller Instruction Set is optimized for 8-bit control applications. As a typical 8-bit processor, the 8051 Microcontroller instructions have 8-bit Opcodes. As a result, the 8051 Microcontroller instruction set can have up to $2^8 = 256$ Instructions.
.

A Brief Look at 8051 Microcontroller Instructions and Groups

Before going into the details of the 8051 Microcontroller Instruction Set, Types of Instructions and the Addressing Mode, let us take a brief look at the instructions and the instruction groups of the 8051 Microcontroller Instruction Set (the MCS-51 Instruction Set).
The following table shows the 8051 Instruction Groups and Instructions in each group. There are 49 Instruction Mnemonics in the 8051 Microcontroller Instruction Set and these 49 Mnemonics are divided into five groups.

DATA TRANSFER	ARITHMETIC	LOGICAL	BOOLEAN	PROGRAM BRANCHING
MOV	ADD	ANL	CLR	LJMP
MOVC	ADDC	ORL	SETB	AJMP
MOVX	SUBB	XRL	MOV	SJMP
PUSH	INC	CLR	JC	JZ
POP	DEC	CPL	JNC	JNZ
XCH	MUL	RL	JB	CJNE
XCHD	DIV	RLC	JNB	DJNZ
	DA A	RR	JBC	NOP
		RRC	ANL	LCALL
		SWAP	ORL	ACALL
			CPL	RET
				RETI
				JMP

Table 4.3

A simple instruction consists of just the opcode. Other instructions may include one or more operands. Instruction can be one-byte instruction, which contains only opcode, or two-byte instructions, where the second byte is the operand or three byte instructions, where the operand makes up the second and third byte.
Based on the operation they perform, all the instructions in the 8051 Microcontroller Instruction Set are divided into five groups. They are:

- Data Transfer Instructions
- Arithmetic Instructions
- Logical Instructions
- Boolean or Bit Manipulation Instructions
- Program Branching Instructions

We will now see about these instructions briefly.

Data Transfer Instructions

The Data Transfer Instructions are associated with transfer with data between registers or external program memory or external data memory. The Mnemonics associated with Data Transfer are given below.

- MOV
- MOVC
- MOVX
- PUSH
- POP
- XCH
- XCHD

The following table lists out all the possible data transfer instruction along with other details like addressing mode, size occupied and number machine cycles it takes.

Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
MOV	A, #Data	A ← Data	Immediate	2	1
	A, Rn	A ← Rn	Register	1	1
	A, Direct	A ← (Direct)	Direct	2	1
	A, @Ri	A ← @Ri	Indirect	1	1
	Rn, #Data	Rn ← data	Immediate	2	1
	Rn, A	Rn ← A	Register	1	1
	Rn, Direct	Rn ← (Direct)	Direct	2	2
	Direct, A	(Direct) ← A	Direct	2	1
	Direct, Rn	(Direct) ← Rn	Direct	2	2
	Direct1, Direct2	(Direct1) ← (Direct2)	Direct	3	2
	Direct, @Ri	(Direct) ← @Ri	Indirect	2	2
	Direct, #Data	(Direct) ← #Data	Direct	3	2
	@Ri, A	@Ri ← A	Indirect	1	1
	@Ri, Direct	@Ri ← Direct	Indirect	2	2
	@Ri, #Data	@Ri ← #Data	Indirect	2	1
	DPTR, #Data16	DPTR ← #Data16	Immediate	3	2
MOVC	A, @A+DPTR	A ← Code Pointed by A+DPTR	Indexed	1	2
	A, @A+PC	A ← Code Pointed by A+PC	Indexed	1	2
	A, @Ri	A ← Code Pointed by Ri (8-bit Address)	Indirect	1	2
MOVX	A, @DPTR	A ← External Data Pointed by DPTR	Indirect	1	2
	@Ri, A	@Ri ← A (External Data 8-bit Addr)	Indirect	1	2
	@DPTR, A	@DPTR ← A (External Data 16-bit Addr)	Indirect	1	2
PUSH	Direct	Stack Pointer SP ← (Direct)	Direct	2	2
POP	Direct	(Direct) ← Stack Pointer SP	Direct	2	2
XCH	Rn	Exchange ACC with Rn	Register	1	1
	Direct	Exchange ACC with Direct Byte	Direct	2	1
	@Ri	Exchange ACC with Indirect RAM	Indirect	1	1
XCHD	A, @Ri	Exchange ACC with Lower Order Indirect RAM	Indirect	1	1

Table 4.4

Arithmetic Instructions

Using Arithmetic Instructions, you can perform addition, subtraction, multiplication and division. The arithmetic instructions also include increment by one, decrement by one and a special instruction called Decimal Adjust Accumulator.

The Mnemonics associated with the Arithmetic Instructions of the 8051 Microcontroller Instruction Set are:

- ADD
- ADDC
- SUBB
- INC
- DEC

- MUL
- DIV
- DAA

The arithmetic instructions has no knowledge about the data format i.e. signed, unsigned, ASCII, BCD, etc. Also, the operations performed by the arithmetic instructions affect flags like carry, overflow, zero, etc. in the PSW Register.
All the possible Mnemonics associated with Arithmetic Instructions are mentioned in the following table.

Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
ADD	A, #Data	A ← A + Data	Immediate	2	1
	A, Rn	A ← A + Rn	Register	1	1
	A, Direct	A ← A + (Direct)	Direct	2	1
	A, @Ri	A ← A + @Ri	Indirect	1	1
ADDC	A, #Data	A ← A + Data + C	Immediate	2	1
	A, Rn	A ← A + Rn + C	Register	1	1
	A, Direct	A ← A + (Direct) + C	Direct	2	1
	A, @Ri	A ← A + @Ri + C	Indirect	1	1
SUBB	A, #Data	A ← A – Data – C	Immediate	2	1
	A, Rn	A ← A – Rn – C	Register	1	1
	A, Direct	A ← A – (Direct) – C	Direct	2	1
	A, @Ri	A ← A – @Ri – C	Indirect	1	1
MUL	AB	Multiply A with B (A ← Lower Byte of A*B and B ← Higher Byte of A*B)	--	1	4
DIV	AB	Divide A by B (A ← Quotient and B ← Remainder)	--	1	4
DEC	A	A ← A – 1	Register	1	1
	Rn	Rn ← Rn – 1	Register	1	1
	Direct	(Direct) ← (Direct) – 1	Direct	2	1
	@Ri	@Ri ← @Ri – 1	Indirect	1	1
INC	A	A ← A + 1	Register	1	1
	Rn	Rn ← Rn + 1	Register	1	1
	Direct	(Direct) ← (Direct) + 1	Direct	2	1
	@Ri	@Ri ← @Ri + 1	Indirect	1	1
	DPTR	DPTR ← DPTR + 1	Register	1	2
DA	A	Decimal Adjust Accumulator	--	1	1

Table 4.5

Logical Instructions

The next group of instructions are the Logical Instructions, which perform logical operations like AND, OR, XOR, NOT, Rotate, Clear and Swap. Logical Instruction are performed on Bytes of data on a bit-by-bit basis.
Mnemonics associated with Logical Instructions are as follows:

- ANL
- ORL
- XRL
- CLR
- CPL
- RL
- RLC
- RR
- RRC
- SWAP

The following table shows all the possible Mnemonics of the Logical Instructions.

Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
ANL	A, #Data	A ← A AND Data	Immediate	2	1
	A, Rn	A ← A AND Rn	Register	1	1
	A, Direct	A ← A AND (Direct)	Direct	2	1
	A, @Ri	A ← A AND @Ri	Indirect	1	1
	Direct, A	(Direct) ← (Direct) AND A	Direct	2	1
	Direct, #Data	(Direct) ← (Direct) AND #Data	Direct	3	2
ORL	A, #Data	A ← A OR Data	Immediate	2	1
	A, Rn	A ← A OR Rn	Register	1	1
	A, Direct	A ← A OR (Direct)	Direct	2	1
	A, @Ri	A ← A OR @Ri	Indirect	1	1
	Direct, A	(Direct) ← (Direct) OR A	Direct	2	1
	Direct, #Data	(Direct) ← (Direct) OR #Data	Direct	3	2
XRL	A, #Data	A ← A XRL Data	Immediate	2	1
	A, Rn	A ← A XRL Rn	Register	1	1
	A, Direct	A ← A XRL (Direct)	Direct	2	1
	A, @Ri	A ← A XRL @Ri	Indirect	1	1
	Direct, A	(Direct) ← (Direct) XRL A	Direct	2	1
	Direct, #Data	(Direct) ← (Direct) XRL #Data	Direct	3	2
CLR	A	A← 00H	--	1	1
CPL	A	A ← A	--	1	1
RL	A	Rotate ACC Left	--	1	1
RLC	A	Rotate ACC Left through Carry	--	1	1
RR	A	Rotate ACC Right	--	1	1
RRC	A	Rotate ACC Right through Carry	--	1	1
SWAP	A	Swap Nibbles within ACC	--	1	1

Table 4.6

Boolean or Bit Manipulation Instructions

As the name suggests, Boolean or Bit Manipulation Instructions will deal with bit variables. We know that there is a special bit-addressable area in the RAM and some of the Special Function Registers (SFRs) are also bit addressable.
The Mnemonics corresponding to the Boolean or Bit Manipulation instructions are:

- CLR
- SETB
- MOV
- JC
- JNC
- JB
- JNB
- JBC
- ANL
- ORL
- CPL

These instructions can perform set, clear, and, or, complement etc. at bit level. All the possible mnemoics of the Boolean Instructions are specified in the following table.

Mnemonic	Instruction	Description	# of Bytes	# of Cycles
CLR	C	C ← 0 (C = Carry Bit)	1	1
	Bit	Bit ← 0 (Bit = Direct Bit)	2	1
SET	C	C ← 1	1	1
	Bit	Bit ← 1	2	1
CPL	C	C ← $\overline{C}$	1	1
	Bit	Bit ← $\overline{Bit}$	2	1
ANL	C, /Bit	C ← C. $\overline{Bit}$ (AND)	2	1
	C, Bit	C ← C . Bit (AND)	2	1
ORL	C, /Bit	C ← C + $\overline{Bit}$ (OR)	2	1
	C, Bit	C ← C + Bit (OR)	2	1
MOV	C, Bit	C ← Bit	2	1
	Bit, C	Bit ← C	2	2
JC	rel	Jump is Carry (C) is Set	2	2
JNC	rel	Jump is Carry (C) is Not Set	2	2
JB	Bit, rel	Jump is Direct Bit is Set	3	2
JNB	Bit, rel	Jump is Direct Bit is Not Set	3	2
JBC	Bit, rel	Jump is Direct Bit is Set and Clear Bit	3	2

Table 4.7

Program Branching Instructions

The last group of instructions in the 8051 Microcontroller Instruction Set are the Program Branching Instructions. These instructions control the flow of program logic. The mnemonics of the Program Branching Instructions are as follows.

- LJMP
- AJMP
- SJMP
- JZ
- JNZ
- CJNE
- DJNZ
- NOP
- LCALL
- ACALL
- RET
- RETI
- JMP

All these instructions, except the NOP (No Operation) affect the Program Counter (PC) in one way or other. Some of these instructions has decision making capability before transferring control to other part of the program.
The following table shows all the mnemonics with respect to the program branching instructions.

LECTURE 4: ADDRESSING MODES OF 8051

1.8051 ADDRESSING MODES

What is an Addressing Mode?

An Addressing Mode is a way to locate a target Data, which is also called as Operand. The 8051 Family of Microcontrollers allows five types of Addressing Modes for addressing the Operands. They are:

- Immediate Addressing
- Register Addressing
- Direct Addressing
- Register – Indirect Addressing
- Indexed Addressing

Immediate Addressing

In Immediate Addressing mode, the operand, which follows the Opcode, is a constant data of either 8 or 16 bits. The name Immediate Addressing came from the fact that the constant data to be stored in the memory immediately follows the Opcode.

The constant value to be stored is specified in the instruction itself rather than taking from a register. The destination register to which the constant data must be copied should be the same size as the operand mentioned in the instruction.

Example: MOV A, #030H

Here, the Accumulator is loaded with 30 (hexadecimal). The # in the operand indicates that it is a data and not the address of a Register.

Immediate Addressing is very fast as the data to be loaded is given in the instruction itself.

Register Addressing

In the 8051 Microcontroller Memory Organization Tutorial, we have seen the organization of RAM and four banks of Working Registers with eight Registers in each bank.

In Register Addressing mode, one of the eight registers (R0 – R7) is specified as Operand in the Instruction.

It is important to select the appropriate Bank with the help of PSW Register. Let us see a example of Register Addressing assuming that Bank0 is selected.

Example: MOV A, R5

Here, the 8-bit content of the Register R5 of Bank0 is moved to the Accumulator.

Direct Addressing

In Direct Addressing Mode, the address of the data is specified as the Operand in the instruction. Using Direct Addressing Mode, we can access any register or on-chip variable. This includes general purpose RAM, SFRs, I/O Ports, Control registers.

Example: MOV A, 47H

Here, the data in the RAM location 47H is moved to the Accumulator.

Register Indirect Addressing

In the Indirect Addressing Mode or Register Indirect Addressing Mode, the address of the Operand is specified as the content of a Register. This will be clearer with an example.

Example: MOV A, @R1

The @ symbol indicates that the addressing mode is indirect. If the contents of R1 is 56H, for example, then the operand is in the internal RAM location 56H. If the contents of the RAM location 56H is 24H, then 24H is moved into accumulator.
Only R0 and R1 are allowed in Indirect Addressing Mode. These register in the indirect addressing mode are called as Pointer registers.

Indexed Addressing Mode

With Indexed Addressing Mode, the effective address of the Operand is the sum of a base register and an offset register. The Base Register can be either Data Pointer (DPTR) or Program Counter (PC) while the Offset register is the Accumulator (A).
In Indexed Addressing Mode, only MOVC and JMP instructions can be used. Indexed Addressing Mode is useful when retrieving data from look-up tables.
Example: MOVC A, @A+DPTR
Here, the address for the operand is the sum of contents of DPTR and Accumulator.

Types of Instructions in 8051 Microcontroller Instruction Set

Before seeing the types of instructions, let us see the structure of the 8051 Microcontroller Instruction. An 8051 Instruction consists of an Opcode (short of Operation – Code) followed by Operand(s) of size Zero Byte, One Byte or Two Bytes.
The Op-Code part of the instruction contains the Mnemonic, which specifies the type of operation to be performed. All Mnemonics or the Opcode part of the instruction are of One Byte size.
Coming to the Operand part of the instruction, it defines the data being processed by the instructions. The operand can be any of the following:

- No Operand
- Data value
- I/O Port
- Memory Location
- CPU register

There can multiple operands and the format of instruction is as follows:

• Module 7: Support IC chips

- **Topic:**
- **Introduction to Programmable Peripheral Interface device -8255**
- **Programmable Peripheral Interface**
- **Architecture of Intel 8255A**

- **Session objective:** To discuss about peripheral device

Session outcome: Students will know about PPI 8255

Programmable Peripheral Interface (PPI)

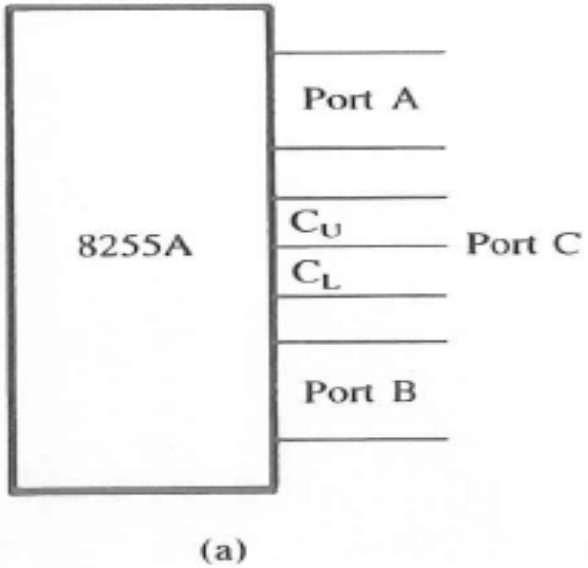
- A Programmable Peripheral Interface is a multiport device. The ports may be programmed in a verity of ways as required by the programmer. The device is very useful for interfacing peripheral devices the term PIA, peripheral Interface Adapter is also used by some manufacturer.
- **INTEL 8255**
- The Intel 8255 is a Programmable Peripheral Interface (PPI). It has two versions, namely Intel 8255A and Intel 8255A-5.

The 8255A is a widely used, programmable, parallel I/O device. It can be programmed to transfer data under various conditions, from simple I/O to interrupt I/O. It is flexible, versatile, and economical (when multiple I/O ports are required), but somewhat complex. It is an important general-purpose I/O device that can be used with almost any microprocessor.

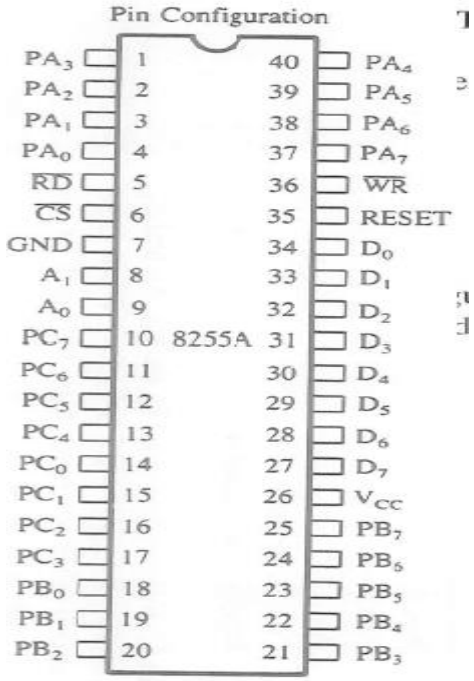
Intel 8255A

The 8255A has 24 I/O pins that can be grouped primarily in two 8-bit parallel ports: A and B, with the remaining eight bits as port C. The eight bits of port C can be used as individual bits or be grouped in two 4-bit ports: C_{UPPER} (C_U) and C_{LOWER} (C_L), as in Figure 1(a). The functions of these ports are defined by writing a control word in the control register.

The block diagram in Figure shows two 8-bit ports (A and B), two 4-bit ports (C_U and C_L), the data bus buffer, and control logic.



(a)



Architecture of Intel 8255A

Intel 8255A is a 40 pin I. C. Package. It operates on a single 5 Vdc supply. Its important characteristics are as follows:

Ambient temperature 0 to 70°C.

Voltage on any pin : 0.5 V to 7 V.

Power dissipation 1 Watt.

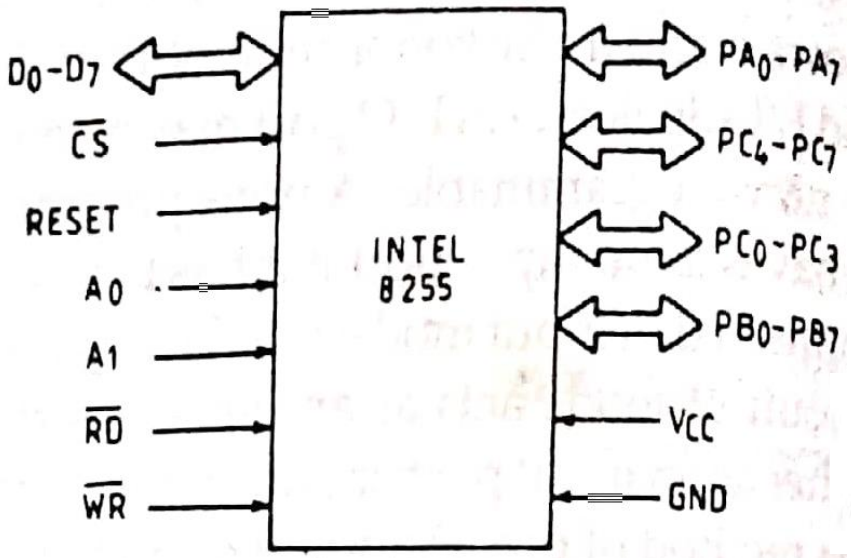
V_{IL} = Input low voltage = Minimum 0.5 V, Maximum 0.8 V.

V_{IH} = Input high voltage = Minimum 2 V, Maximum V_{CC}.

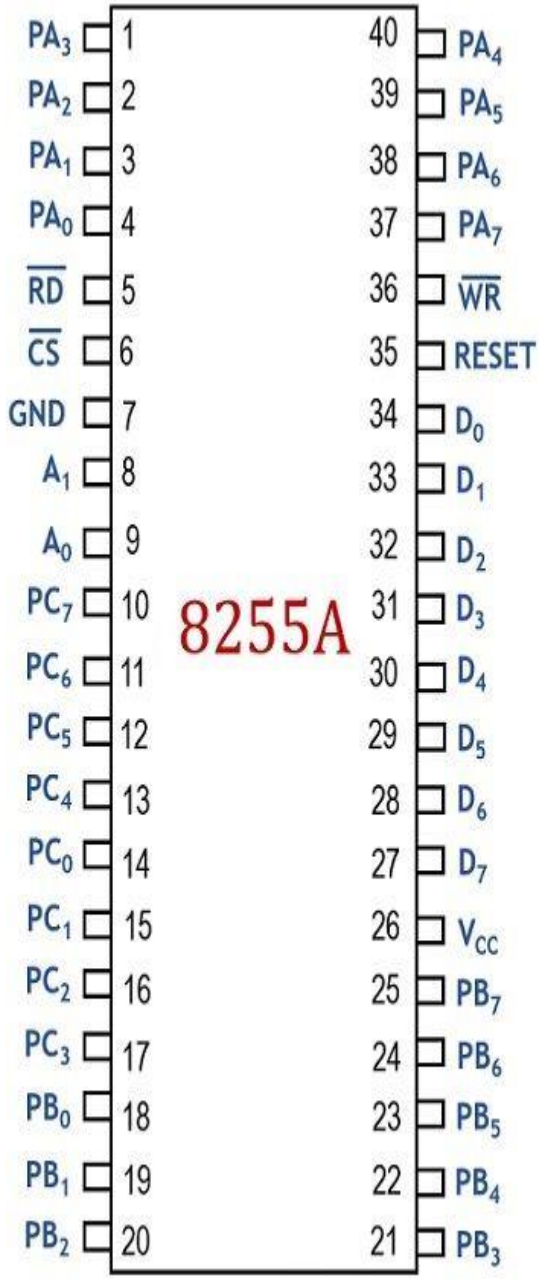
V_{OL} = Output low voltage = 0.45 V.

V_{OH} = Output high voltage = 2.4 V.

I_{DR} = Darlington drive current = Minimum 1 mA, Maximum 4 mA of any 8 pins of the port.



Schematic Diagram of Intel 8255 A.



Pin Diagram of 8255 PPI

Pin Names

D_7–D_0	Data Bus (Bidirectional)
RESET	Reset Input
$\overline{CS}$	Chip Select
$\overline{RD}$	Read Input
$\overline{WR}$	Write Input
A_0, A_1	Port Address
PA_7–PA_0	Port A (Bit)
PB_7–PB_0	Port B (Bit)
PC_7–PC_0	Port C (Bit)
V_{CC}	+5 Volts
GND	0 Volts

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

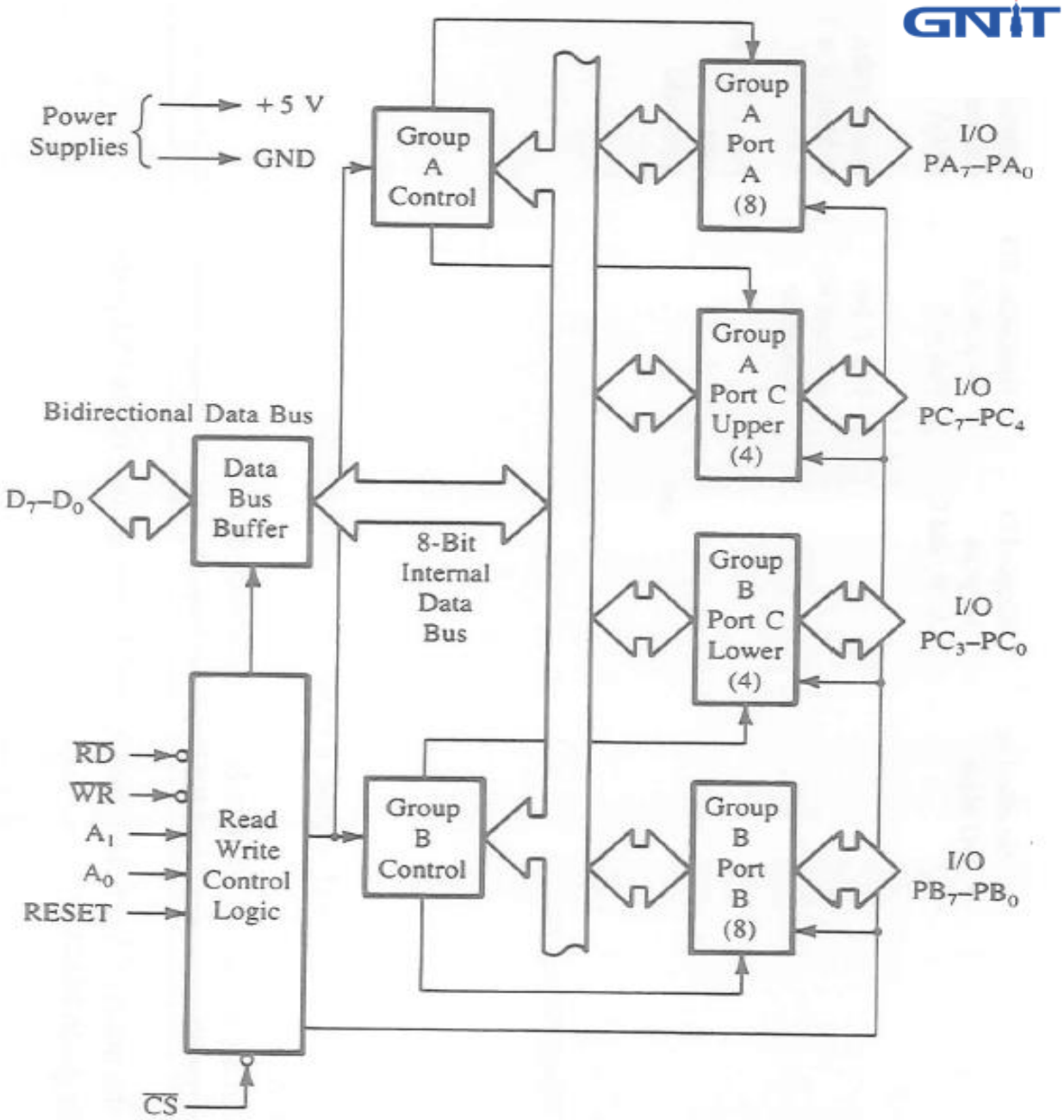
$PC_0 - PC_3$ 4 pins of port C_{lower}.

$PC_4 - PC_7$ 4 pins of port C_{upper}.

JIS GROUP
Educational Initiatives

GNIT

Architecture of 8255 PPI



The important control signals are as follows:

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

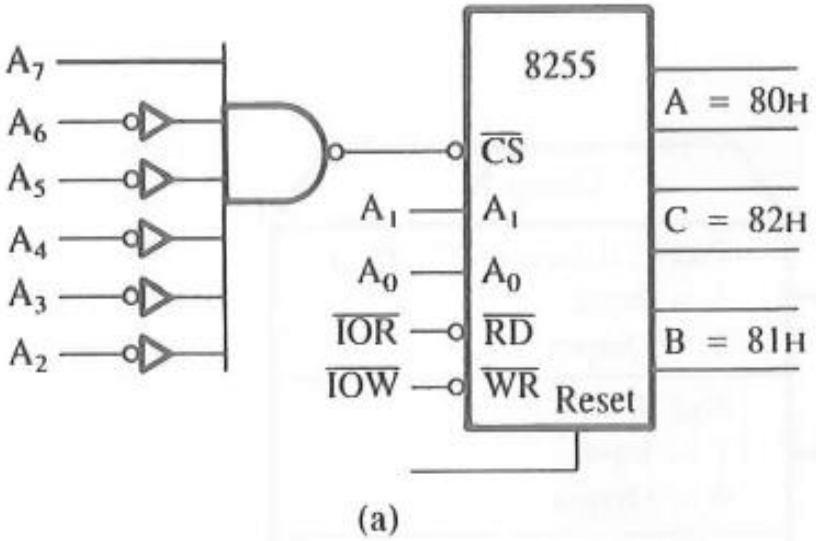
$\overline{RD}$ (Read). When $\overline{RD}$ goes LOW the 8255 sends out data or status information to the CPU on the data bus. In other words it allows the CPU to read data from the input port of 8255.

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

A_0 and A_1. The selection of input port and control word register is done using A_0 and A_1 in conjunction with $\overline{RD}$ and $\overline{WR}$. A_0 and A_1 are normally connected to the least significant bits of the address bus. If two 8255 units are used the address of ports are as follows:

For the 1st unit of 8255, i.e. 8255.1:

Port/Control word register	*Port/Control word register Address*
Port A	00
Port B	01
Port C	02
Control word register	03



(a)

$\overline{CS}$								Hex Address	Port
A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0		
1	0	0	0	0	0	0	0	= 80H	A
						0	1	= 81H	B
						1	0	= 82H	C
						1	1	= 83H	Control Register

(b)

FIGURE
8255A Chip Select Logic (a) and I/O Port Addresses (b)

Content:

- **Different Mode of operation -INTEL-8255**
- **Session objective:** To discuss about **Mode of operation -INTEL-8255**
- **Session outcome**: Students will know about **Mode of operation -INTEL-8255** PPI 8255

Modes of Operation of 8255 PPI

- 8255 has two modes of operation. These are as follows:
- **Bit Set-Reset mode:** When port C is utilized for control or status operation, then by sending an OUT instruction, each individual bit of port C can be set or reset.
- **I/O mode:** As we know that the I/ O mode is sub-classified into 3 modes. So, let us now discuss the 3 modes here.
- ***Mode 0***: Input/Output mode
- This mode is the simple input output mode of 8255 which allows the programming of each port as either input or output port.
- The input/output feature of mode 0 includes:
- It does not support handshaking or interrupt capability.
- The input ports are buffered while outputs are latched.

Mode 1: Input/output with handshaking

- Mode 1 of 8255 supports handshaking with the ports programmed as either input or output mode. We know that it is not necessary that all the time the data is transferred between two devices operating at same speed. So, handshaking signals are used to synchronize the data transfer between two devices that operates at different speeds.
- The figure below shows the data transferring between CPU and an output device having different operating speeds:

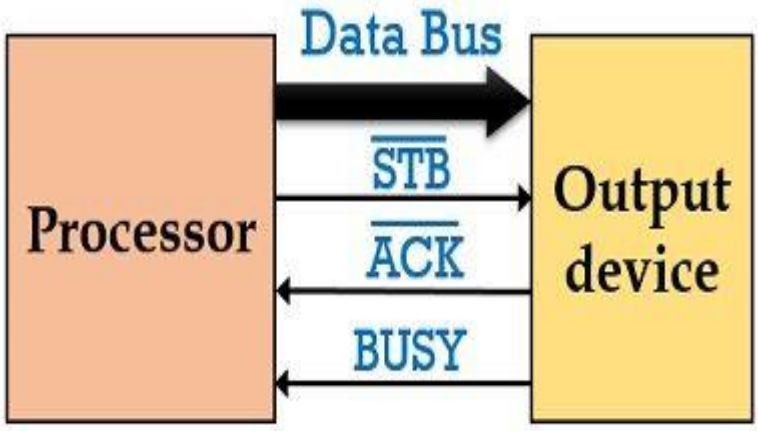
Here STB signal is used to inform the output device that data is available on the data bus by the processor.

Here port A and port B can be separately configured as either input or output port.

Both the port utilizes 3-3 lines of port C for handshaking signals. The rest two lines operates as input/output port.

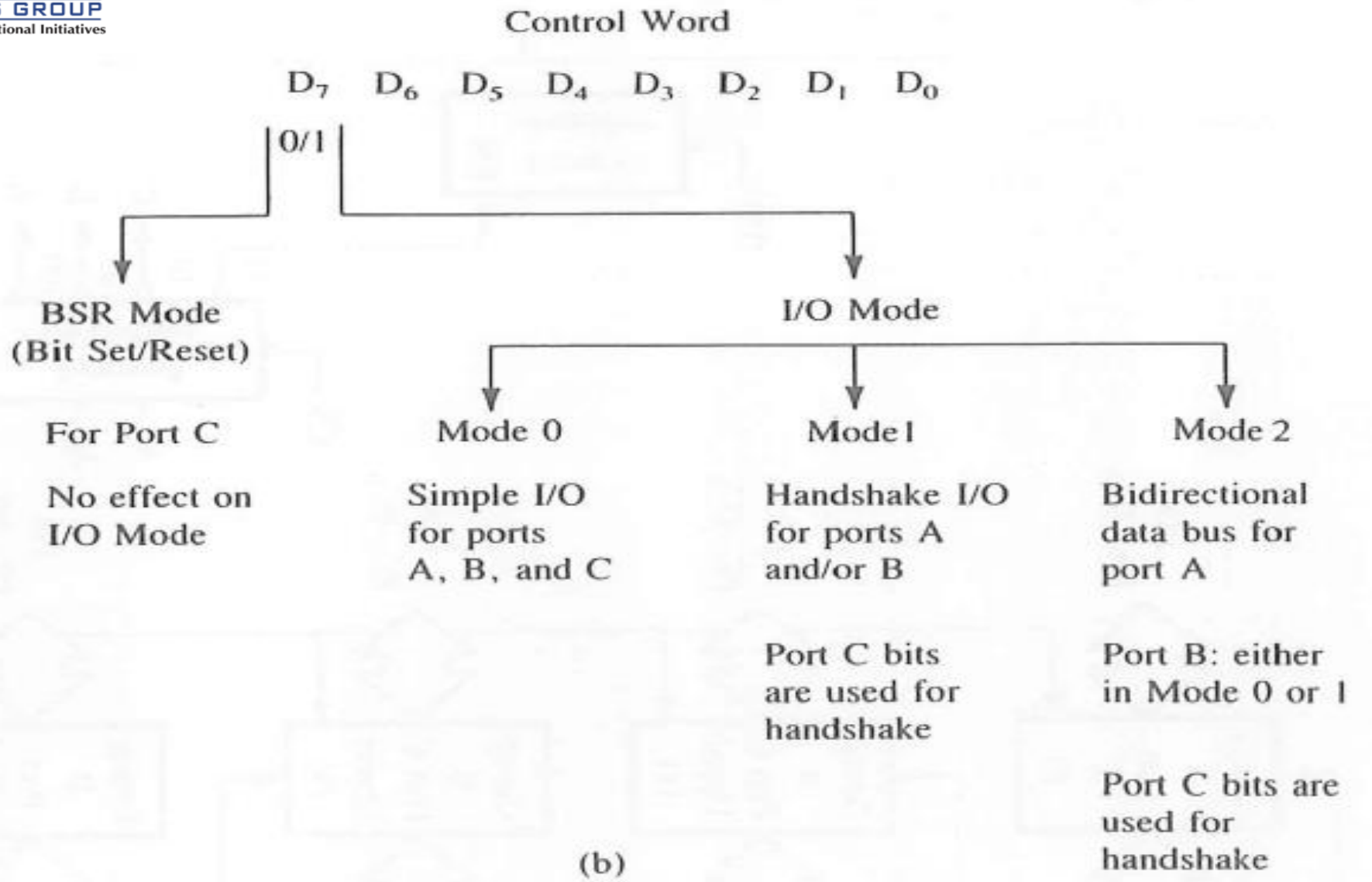
It supports interrupt logic.

The data at the input or output ports are latched.

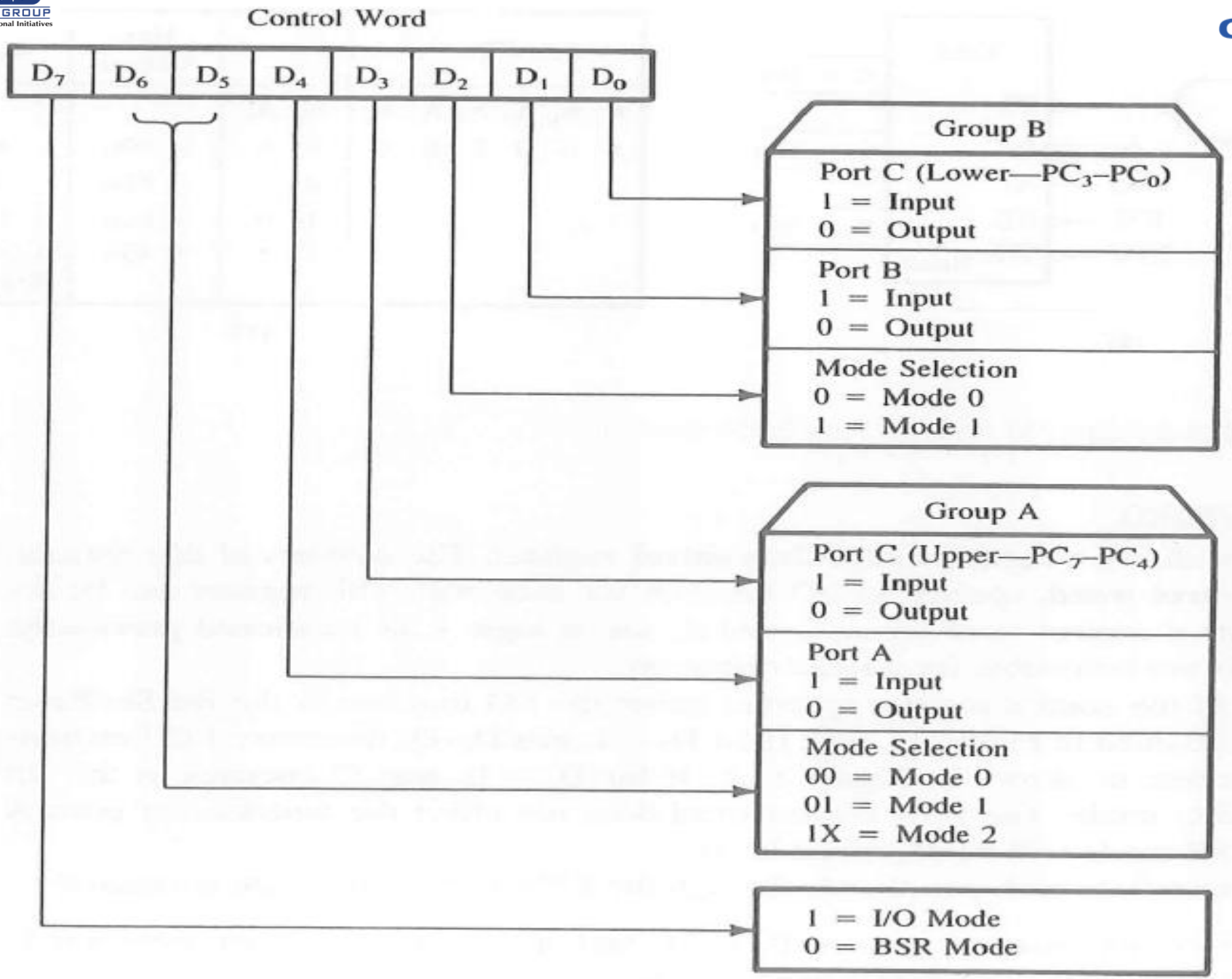


Data Transfer using handshaking signals

- ***Mode 2***: Bidirectional I/O port with handshaking
- In this mode, the ports can be utilized for the bidirectional flow of information by handshaking signals. The pins of group A can be programmed to acts as bidirectional data bus and the port C upper (PC_7 – PC_4) are used by the handshaking signal. The rest 4 lower port C bits are utilized for I/O operations.
- As the data bus exhibits bidirectional nature thus when the peripheral device request for a data input only then the processor load the data in the data bus. Port B can be programmed in mode 0 and 1. And in mode 1 the lower bits of port C of group B are used for handshaking signals.



(b)


Control Word
D7 D6 D5 D4 D3 D2 D1 D0
Group B
Port C (Lower—PC3–PC0)
1 = Input
0 = Output
Port B
1 = Input
0 = Output
Mode Selection
0 = Mode 0
1 = Mode 1
Group A
Port C (Upper—PC7–PC4)
1 = Input
0 = Output
Port A
1 = Input
0 = Output
Mode Selection
00 = Mode 0
01 = Mode 1
1X = Mode 2
1 = I/O Mode
0 = BSR Mode

Example 1. Make control word when the ports of Intel 8255 are defined as follows :

Port A as an input port

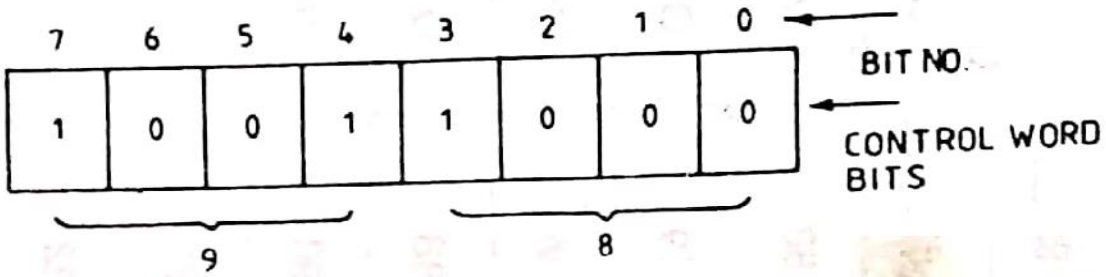
Mode of the Port A—Mode 0

Port B as an output port.

Mode of the Port B—Mode 0.

Port C_{upper} as an input port

Port C_{lower} as an output port.



The control word = 98 H

Bit No. 0 is set to 0, as the Port C_{lower} is an output port.

Bit No. 1 is set to 0, as the port B is an output port.

Bit No. 2 is set to 0, as the Port B has to operate in Mode 0.

Bit No. 3 is set to 1, as the Port C_{upper} is an input port.

Bit No. 4 is set to 1, as the Port A is an input port.

Bit No. 5 and 6 are set to 00 as the Port A has to operate in Mode 0.

Bit No. 7 is set to 1, as the Ports A, B and C are used as simple input/output port.

Thus the control word = 98 H.

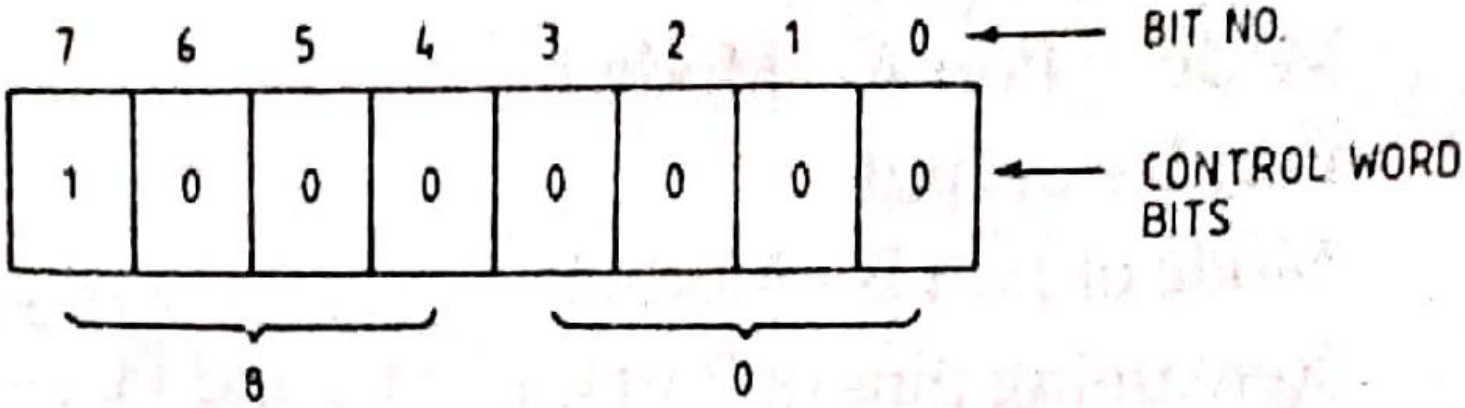
Example . Make control word for the following arrangement of the ports of Intel 8255 for mode 0 operation :

Port A—output

Port B—output

Port C_{upper}—output

Port C_{lower}—output

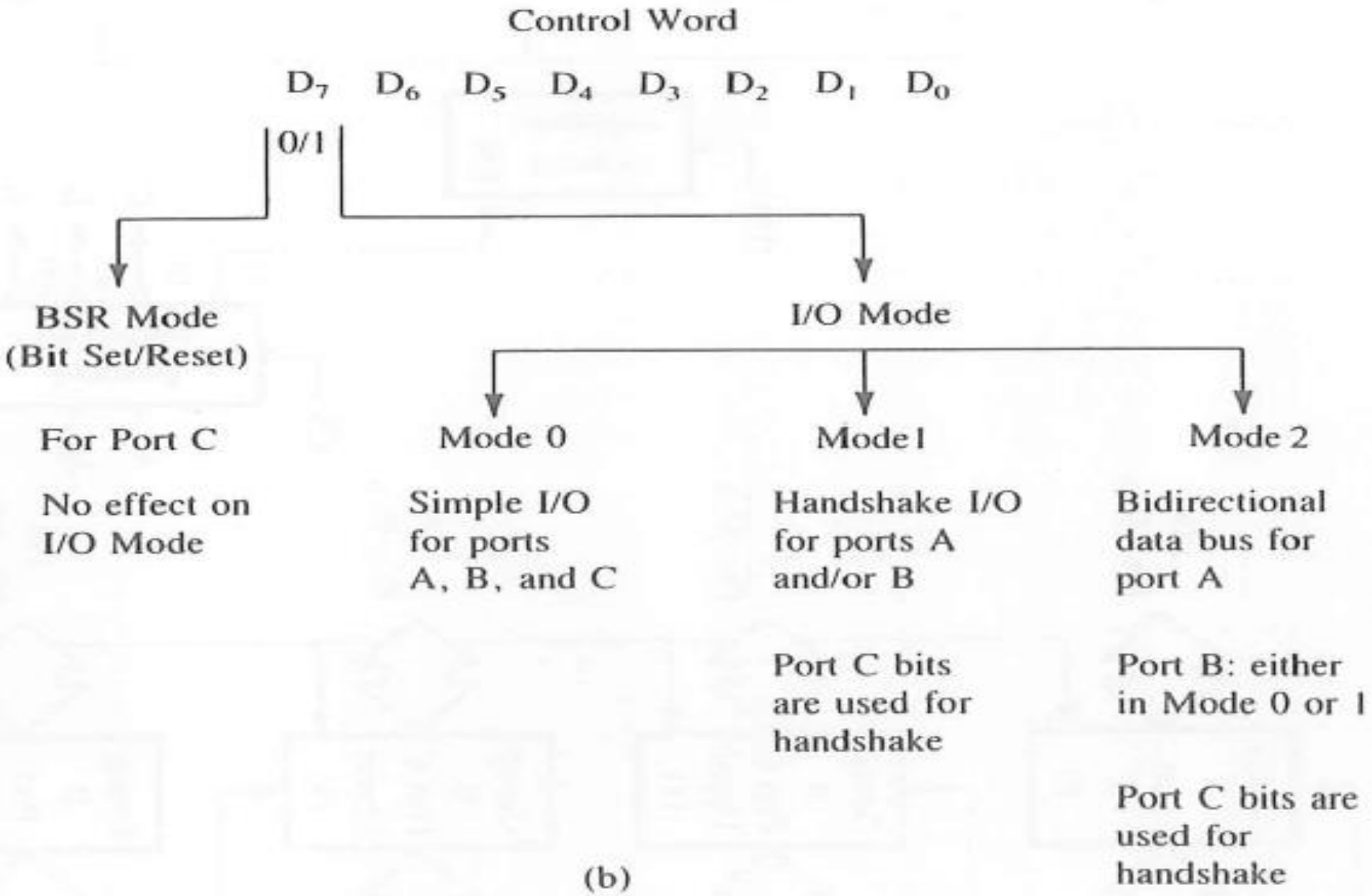


Content:

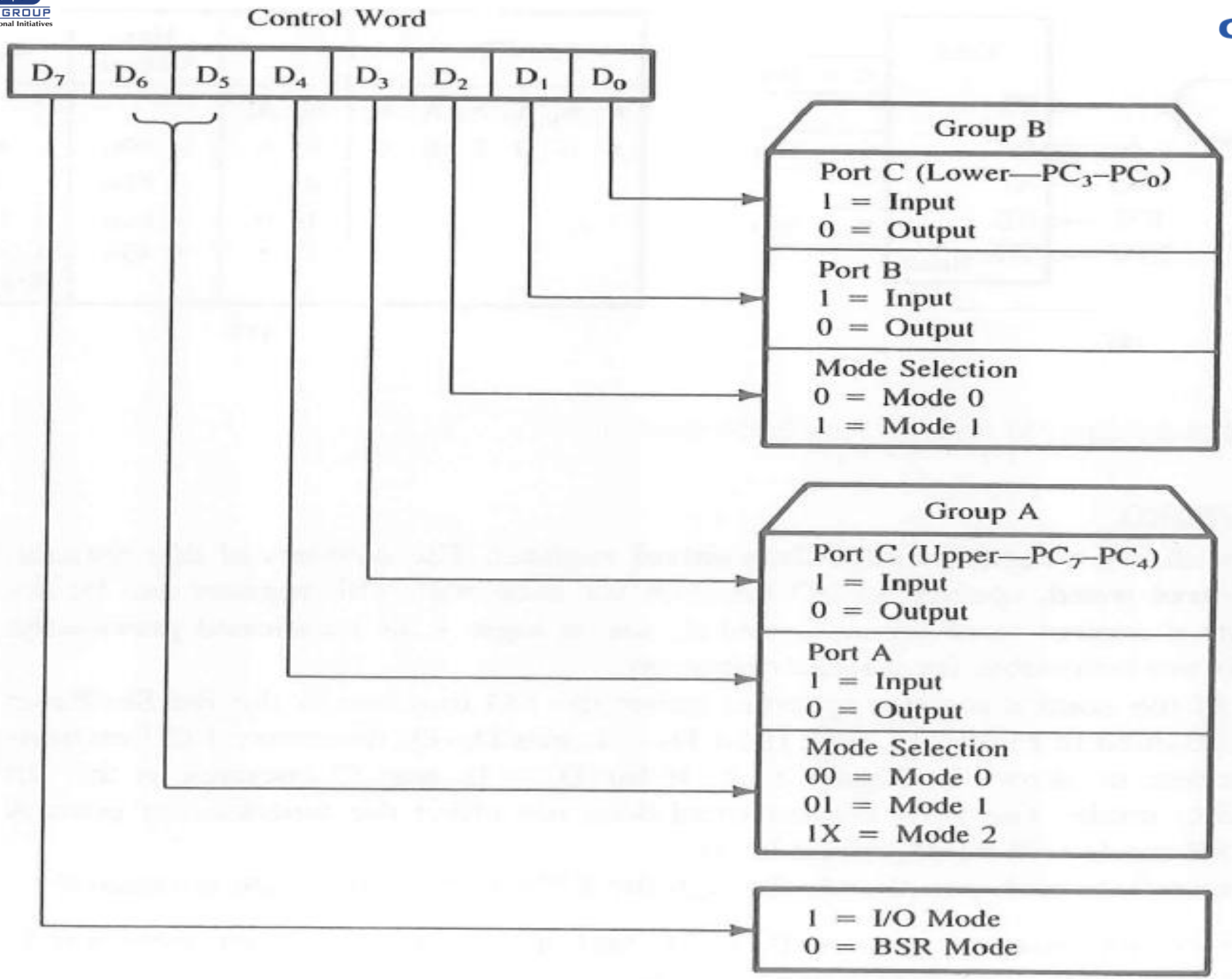
- Mode 0 operation of INTEL 8255

- **Session objective:** To discuss about Mode 0 operation of INTEL 8255
- **Session outcome**: Students will know about Mode 0 operation of INTEL 8255

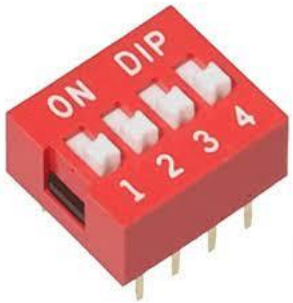
INTEL 8255 operation



(b)


Control Word
D7 D6 D5 D4 D3 D2 D1 D0
Group B
Port C (Lower—PC3–PC0)
1 = Input
0 = Output
Port B
1 = Input
0 = Output
Mode Selection
0 = Mode 0
1 = Mode 1
Group A
Port C (Upper—PC7–PC4)
1 = Input
0 = Output
Port A
1 = Input
0 = Output
Mode Selection
00 = Mode 0
01 = Mode 1
1X = Mode 2
1 = I/O Mode
0 = BSR Mode

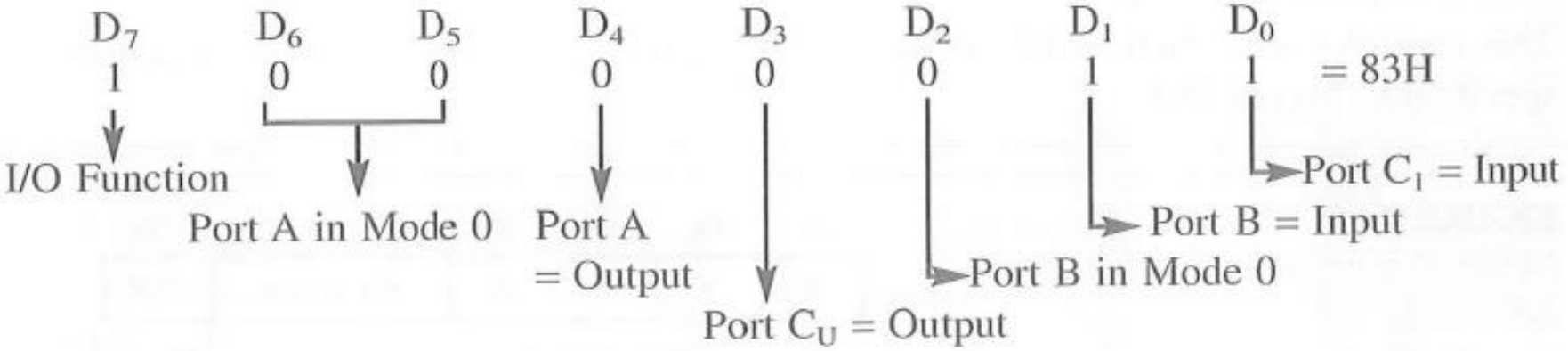
Write a program to read the DIP switches and display the reading from port B at port A and from port C_L at port C_U.



1. Port Addresses This is a memory-mapped I/O; when the address line A_{15} is high, the Chip Select line is enabled. Assuming all don't care lines are at logic 0, the port addresses are as follows:

Port A	=	8000H ($A_1 = 0, A_0 = 0$)
Port B	=	8001H ($A_1 = 0, A_0 = 1$)
Port C	=	8002H ($A_1 = 1, A_0 = 0$)
Control Register	=	8003H ($A_1 = 1, A_0 = 1$)

2. Control Word



3. Program

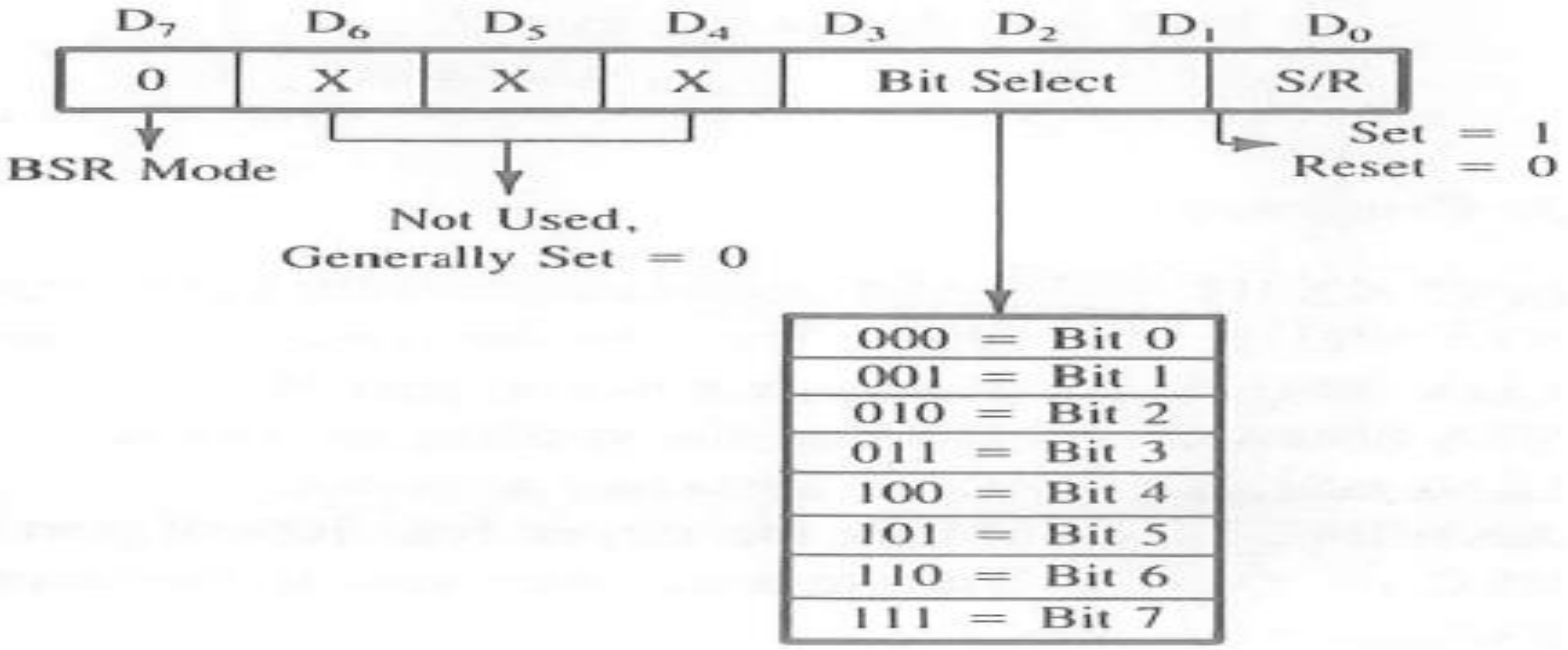
```
MVI A,83H       ;Load accumulator with the control word
STA 8003H       ;Write word in the control register to initialize the ports
LDA 8001H       ;Read switches at port B
STA 8000H       ;Display the reading at port A
LDA 8002H       ;Read switches at port C
ANI 0FH         ;Mask the upper four bits of port C; these bits are not input d
RLC             ;Rotate and place data in the upper half of the accumulator
RLC
RLC
RLC
RLC
STA 8002H       ;Display data at port C_U
HLT
```

BSR (Bit Set/ Reset) mode

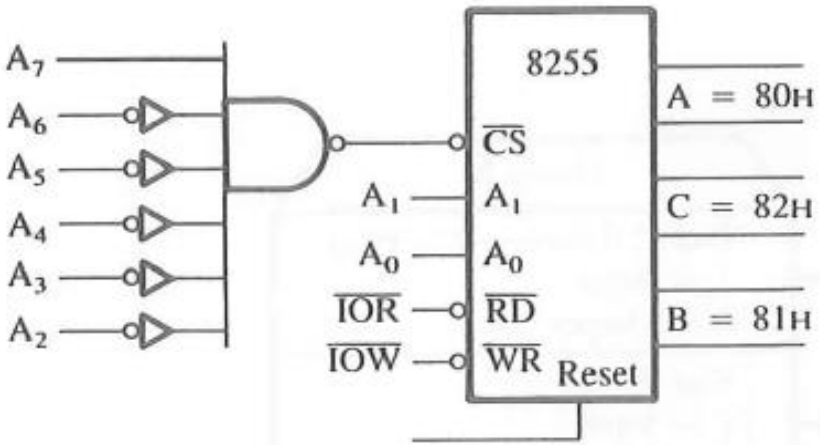
The BSR mode is concerned only with the eight bits of port C, which can be set or reset by writing an appropriate control word in the control register. A control word with bit $D_7 = 0$ is recognized as a BSR control word, and it does not alter any previously transmitted control word with bit $D_7 = 1$; thus the I/O operations of ports A and B are not affected by a BSR control word. In the BSR mode, individual bits of port C can be used for applications such as an on/off switch.

BSR CONTROL WORD

This control word, when written in the control register, sets or resets one bit at a time,



Write a BSR control word subroutine to set bits PC_7 and PC_3 and reset them after 10 ms. Use the schematic in Figure and assume that a delay subroutine is available.



(a)

$\overline{CS}$								Hex Address	Port
A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0		
1	0	0	0	0	0	0	0	= 80H	A
						0	1	= 81H	B
						1	0	= 82H	C
						1	1	= 83H	Control Register

(b)

BSR CONTROL WORDS

		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
To set bit PC_7	=	0	0	0	0	1	1	1	1	=	0FH
To reset bit PC_7	=	0	0	0	0	1	1	1	0	=	0EH
To set bit PC_3	=	0	0	0	0	0	1	1	1	=	07H
To reset bit PC_3	=	0	0	0	0	0	1	1	0	=	06H

PORT ADDRESS

Control register address = 83H; refer to Figure

SUBROUTINE

```
BSR:    MVI A,0FH       ;Load byte in accumulator to set PC7
        OUT 83H         ;Set PC7 = 1
        MVI A,07H       ;Load byte in accumulator to set PC3
        OUT 83H         ;Set PC3 = 1
        CALL DELAY      ;This is a 10-ms delay
        MVI A,06H       ;Load accumulator with the byte to reset PC3
        OUT 83H         ;Reset PC7
        MVI A,0EH       ;Load accumulator with the byte to reset PC7
        OUT 83H         ;Rest PC7
        RET
```

From an analysis of the above routine, the following points can be noted:

1. To set/reset bits in port C, a control word is written in the control register and not in port C.
2. A BSR control word affects only one bit in port C.
3. The BSR control word does not affect the I/O mode.

THANK YOU!

Dr. Debasree Saha

Email : debasree.saha@gnit.ac.in

Ph: 7005338712