



Dear students,

Welcome to the world of Microprocessors!

You are about to learn, how these little chips became the central force behind the computer revolution. Brain child of Alan Turing, nurtured by genius minds like Von Neumann, Maurice Wilkes, and materialized by corporations like Intel, Samsung, Apple... these chips transformed the way the world computed.

From traffic signals to air traffic control, drones to satellites, fitness bands to pace makers, Microprocessors have brought to you, ever improving standards of health, travel, entertainment, communication etc.

In this book, you begin with learning the 8086 Architecture.

You then learn 8086 in depth including its Memory module, Instruction set, Programming and interfacing with its set of coprocessors and peripherals.

You will then proceed to learning powerful microprocessors like Pentium.

You will see the speed advantage achieved in Pentium using superscalar architecture and on chip cache memories. You will then witness the brilliance of Branch Prediction Algorithm which minimized the biggest drawback of pipelining.

Before we start, allow me to take the opportunity to thank my mother,  
Prof. Veena D. Acharya.

A teacher all her life, she was the primary inspiration for me to pursue this noble profession.  
Her blessings are reflected in the enthusiasm shown in this book and in my lectures.

Bharat D. Acharya.  
B. E. Computer Science.  
Founder, Bharat Acharya Education.  
Teaching Microprocessors & Microcontrollers since 2000.



## INTRODUCTION TO MICROPROCESSORS

Drones, stump vision cameras, mobile phones, RFID sensors, autonomous cars, streaming servers, your favorite shopping website... all of these are fruits of a seed called "**Microprocessor**", planted years ago in mid 1970s and in fact conceived even earlier in 1940s.

So what does this Microprocessor (henceforth called  $\mu P$ ) actually do... Why do we need to learn it... And most importantly, after completing our education, how will this knowledge be useful to us...

**Where do we use a  $\mu P$ ?** Is a  $\mu P$  used in a fan, or in a tube light, or in a switch board? No, none of them. Is it used in a mobile phone, a computer, a microwave oven, a washing machine? Yes, all of them! This is because they run on programs, and all those programs are executed by the microprocessor within these devices. **This is the main function of a  $\mu P$ , to execute programs.**

In our day to day encounters we come across several devices and appliances. If you feel any of them works on a program, you should most certainly realize, it must contain a microprocessor. Take a microwave oven as an example. The  $\mu P$  inside the oven isn't directly cooking the food. It is running programs that are responsible for rotating the dish, maintaining the correct temperature, counting the desired number of seconds, displaying the time remaining on the screen, and finally ringing the sweet alarm (ding!) informing us that the cooking is complete. All of these require programs, that are executed by the oven's  $\mu P$ . And who writes the programs? The engineer, yes that's you! It is this combination of the engineers mind and the microprocessors execution abilities that has transformed the world in the past few decades and will continue to do so as both are getting ever so smart in the new age.

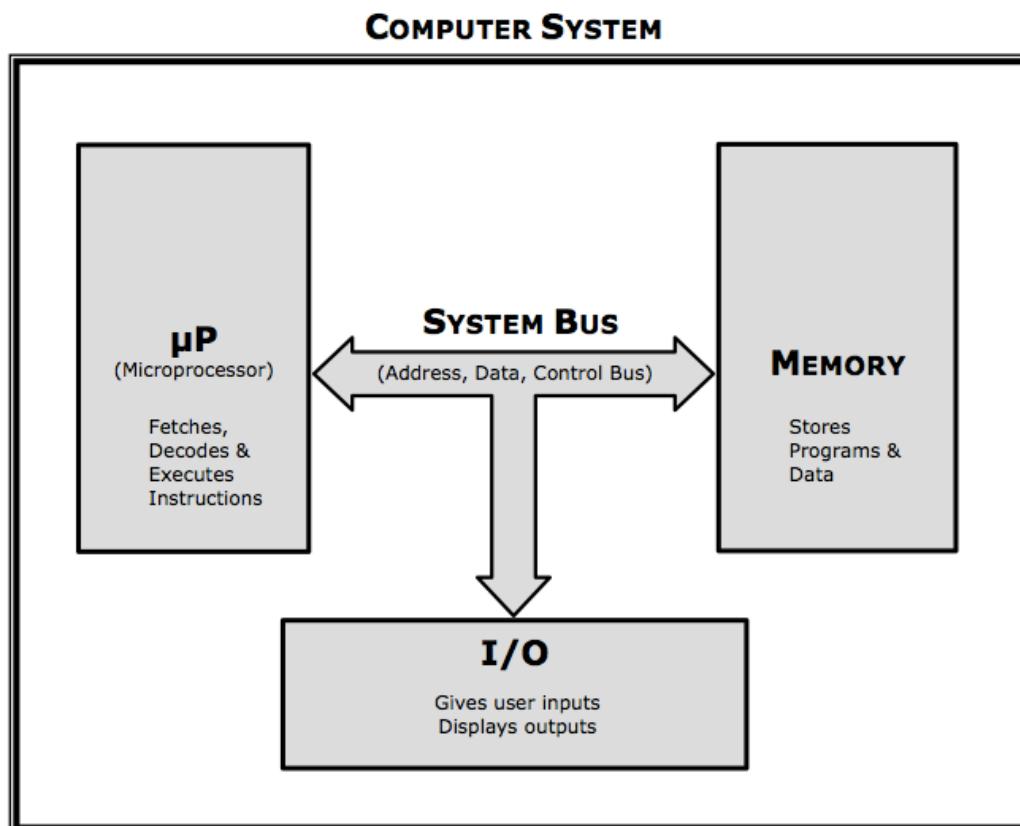
The simplest example of the use of a microprocessor, is in a **computer**. When you imagine a computer, lots of devices come to our mind, like the keyboard, mouse, printer, monitor and the big "box" also casually referred to by laymen as the CPU. Of course, you know their roles! The Keyboard and mouse are called input devices. Are they executing our programs? No! When we want to add two numbers, neither is the keyboard adding them nor is the mouse. So, are they required? Yes! To give inputs. That's their role. To provide inputs to the system. Similarly, the printer and the monitor are used to produce outputs. Hence these are called I/O devices (Input and Output devices). This leaves us with that big "box". Open it and you see an electric circuit board also called the motherboard. **On the motherboard, pretty much around the center lies a big chip, around the size of our palm, that's your  $\mu P$ .** You may notice in modern motherboards the  $\mu P$  is generally covered with a fan, to dissipate the heat generated by relentlessly performing millions and sometimes billions of operations per second. So how important is the  $\mu P$ ? Well, remove it and our computer becomes a piece of junk! Every activity of your computer needs a program, lets get you in agreement with that. You watch a movie, play a song, surf the net, be on social media etc., all of these are programs. Executing them keeps the  $\mu P$  busy round the clock, hence the heat and the fan.

Let's say we are headed to the market right now, to buy the latest computer. Which microprocessor will you find inside? Yes, the Intel Core i7, or maybe Core i5 or i3, Intel also has recently released the Core i9 but its way too expensive to be mass produced as yet. So are we learning Core i7, i5, i3... not so early! These are a result of over 40 years of cutting-edge development making them way too advanced, to be understood by a beginner. We begin with learning the  $\mu P$  that started this whole x86 family... **The 8086 Microprocessor.**



## BASIC ORGANIZATION OF A COMPUTER

A computer system, as we know it, consists of various components. They can be broadly classified into three sections:  
The Processor, Memory and I/O.





## THE PROCESSOR – “μP”

The heart of the computer is its μP (Microprocessor). Current generation computers use processors like Intel Core i3, i5 or i7 and so on. They have come a long way from the initial processors that you are about to learn E.g.: 8085, 8086 etc.

Back in the day (1940s), when micro-electronics was not invented, processors looked very different and were certainly not “micro” in appearance. They were created using huge arrays of physical switches which were operated manually and often occupied large rooms.

In the following decades, with the invention of micro-electronics, scientists managed to embed thousands of microscopic switches (transistors) inside a small chip, and called it a **“Micro-processor”**.

Over the years, microprocessors grew in strength. From housing a few thousand transistors (8085) to containing more than a billion transistors (Core i7), the computational power has been increasing exponentially. Having said that, some of the basics still remain the same.

To put it simply, **the main function of a μP is to Fetch, Decode and Execute instructions.**

**Instructions** are a part of programs. Programs are stored in the memory. First, **μP fetches an instruction** from the memory. It then **decodes the instruction**. This means, it “understands” the binary pattern of the instruction, also called its opcode. Every instruction when stored in the memory is in its unique binary form, which indicates the operation to be performed. **This is called its opcode**. Upon decoding the opcode, μP understands the operation to be performed and hence “executes” the instruction. This entire process is called an **“Instruction cycle”**. This process is repeated for the next instruction. Like this, one by one, all instructions of a program are executed. Of course, by new concepts like **pipelining, multitasking, multiprocessing** etc., this procedure has become very advanced and efficient today. You will get to learn all of them, in the due course of this ever-intriguing subject.

We begin learning with basic processors like **8085** or **8086**, but make no mistake, none of this is “outdated”. Yes, your mobile phone or your computer today uses the most advanced processors (Apple’s **A12 Bionic** etc.), but to run a **traffic light** or **TV remote** control you don’t need a core i7 now, do you? And these are used by the millions across the world. They simply use processors of the same grade as an 8085 or an 8086, with different product numbers as they are made by various manufacturers.

## MEMORY

Memory is used to **store information**.

It stores two kinds of information... **programs and data**. Yes, think about it! Everything that’s stored in your computer’s memory is either some program or some data. MS Word is a program, the Word Documents are data. Your Image Viewer is a program, the images are data. WhatsApp is a program, its messages are data. Instagram is a program, the feed on the wall is the data... and so on!

All programs and data are stored in the memory, in digitized form, where every information is represented in 1s and 0s called binary digits or simply bits.

There are various forms of memory devices.

The main memory also called primary memory consists of Ram and ROM.

Other memory devices like Hard disk, Floppy, CD/ DVD etc. are secondary storage devices.



Additionally there is also a high speed memory called Cache composed of SRAM. For the majority portion of this book, you are dealing with the initial processors like 8086. It will be in your best interest to think of Primary Memory only, whenever we speak of memory. That is because, secondary memory and high speed memories were implemented much later in the evolution of processors as the demand for mass storage and high speed performance started increasing. So, from now on in this book, unless specified otherwise, **the word memory refers to primary memory that is RAM and ROM.**

The memory is a series of locations.

Each location is identified by its own unique address.

**Every memory location contains 1 Byte (8 bits) of data.** There is a very good reason for this, and you will learn it when we discuss the topic of memory banking in 8086.

## I/O DEVICES

**I/O devices are used to enter programs and data as inputs and display or print the results as outputs.**

We are all familiar with devices such as the keyboard, mouse, printer, monitor etc. Every form of computer system has a set of I/O devices for human interaction. A device like a touch screen performs dual functions of both input and output.

The μP, Memory and I/O are all connected to each other using the System Bus.

## SYSTEM BUS

**A Bus is a set of lines.** They are used to transfer information, obviously in binary form.

A line connected to Vcc will carry a logic 1 and if connected to Gnd it will carry logic 0.

Hence one line can transfer 1 bit. If we want multiple bits to be transmitted together, then we use a set of lines grouped together and that's called a bus.

**The size of a bus refers to the number of lines** it contains and is always given in terms of bits.

E.g.: An 8-bit bus has 8 lines, a 16 bit bus has 16 lines and so on.

There are three types of busses... Address, Data and Control Bus. Collectively they are called the system bus. Lets take a closer look at them.

### i) ADDRESS BUS

**It carries the address where the operation has to be performed.** Say the processor wants to write some data at a memory location. Firstly the processor will give the desired address on the address bus. This address will select a unique memory location. That's when data will be transferred with that location.

It is therefore obvious that **bigger the address bus, more is the number of memory locations** it can address and hence bigger the total size of the memory. Here is the relation between size of the address bus and size of the memory.

An address bus of 1 bit can give a total of two addresses: 0 and 1.

Hence can access a total memory of two locations.

A 2-bit address bus can generate a total of 4 addresses 00, 01, 10, 11 and hence can access a total of 4 locations.  
3-bit address bus... 8 locations and so on.



So here is the rule,

**An N-bit address bus can totally access  $2^N$  locations.**

As mentioned earlier, one memory location contains one byte of data.  
This brings us to the following conclusion.

**A processor with an N-bit address bus can access a memory of  $2^N$  Bytes.**

Lets solve a typical VIVA (oral exam) question.

Given the size of address bus, you have to figure out the size of the memory.

ADDRESS BUS (N - BIT)	MEMORY ( $2^N$ BYTES)
4 – bit	$2^4 = 16$ Bytes
6 – bit	$2^6 = 64$ Bytes
10 – bit	$2^{10} = 1024$ Bytes = 1 KB
16 – bit	$2^{16} = 2^6 \times 2^{10} = 64 \times 1\text{ K} = 64\text{ KB}$ ... (8085)
20 – bit	$2^{20} = 2^{10} \times 2^{10} = 1\text{ K} \times 1\text{ K} = 1\text{ MB}$ ... (8086)
30 – bit	$2^{30} = 2^{10} \times 2^{20} = 1\text{ K} \times 1\text{ M} = 1\text{ GB}$
32 – bit	$2^{32} = 2^2 \times 2^{30} = 4 \times 1\text{ G} = 4\text{ GB}$ .... (80386 and Pentium)
40 – bit	$2^{40} = 2^{10} \times 2^{30} = 1\text{ K} \times 1\text{ G} = 1\text{ TB}$ ... (Typical Hard Disk)

In case you found it a little difficult after the 4<sup>th</sup> row, that is because you may have got a little confused with the powers of 2. This issue will persist all along the subject. The smarter thing to do is once for all, lets get this hurdle past us. Lets get all powers of 2 clearly sorted. You will realize in the due course of this subject, how helpful this small exercise will prove to be. Frankly speaking there's rarely a topic in this subject that is not connected with some power of 2. Lets sort this, totally!



## Powers of 2

<b>2<sup>N</sup></b>	<b>VALUE</b>
$2^0$	0
$2^1$	2
$2^2$	4
$2^3$	8
$2^4$	16
$2^5$	32
$2^6$	64
$2^7$	128
$2^8$	256
$2^9$	512
$2^{10}$	$1024 = 1K$
$2^{20}$	$2^{10} \times 2^{10} = 1K \times 1K = 1M$
$2^{24}$	$2^4 \times 2^{20} = 16 \times 1M = 16M$
$2^{28}$	$2^8 \times 2^{20} = 256 \times 1M = 256M$
$2^{30}$	$2^{10} \times 2^{20} = 1K \times 1M = 1G$
$2^{36}$	$2^6 \times 2^{30} = 64 \times 1G = 64G$
$2^{40}$	$2^{10} \times 2^{30} = 1K \times 1G = 1T$



Similarly, there is one more very important fundamental that needs to be sorted out before we start learning the bigger concepts. You need to be sure of number representation and number conversions. You may have been familiar with various number systems like Decimal, Hexadecimal, Binary etc... Out of all of them, Hexadecimal and Binary are the two most widely used number systems in our course of learning this subject.

Every number we write, either in programs or in theory examples, is a hexadecimal number. When this number gets stored inside the computer, it is in binary form. This simply means, you must be very well versed with hex-binary conversions as this will be needed while understanding various examples.

A single hexadecimal digit ranges from 0H... FH. This has 16 options. Hence, to represent the number in binary we need 4 bits as  $2^4 = 16$ . The following table shows this conversion.

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

NO! You do not need to mug this up. There is a simple trick of “8421” that can get you any value from the above table. Consider the number 5 which is basically 4+1. Now consider the four binary bits corresponding to values 8, 4, 2 and 1. Since we need the equivalent of 5 which is 4 plus 1, we need 4 and 1 but we don't need 8 and 2 so in positions of 8 and 2 we put a “0” and in positions of 4 and 1 we put a “1”. So we get 0101. Lets take the example of 0 H. We don't need any of them (8 or 4 or 2 or 1). So we put a “0” for all of them and hence get 0000. If we need F (which is 15), it is  $8 + 4 + 2 + 1$  so we need all of them and hence the binary equivalent is 1111. Take 9 as an example,  $9 = 8 + 1$ . So we need 8 and 1 but not 4 and 2 so we put a “1” for 8 and 1 positions and a “0” for 4 and 2 positions giving us 1001. Hope you can now convert any hex digit into binary.



Now consider a two digit number like 25H. To convert this to binary we need to substitute the 4 bit equivalents of 2 and 5 respectively. 2H is 0010 and 5H is 0101.

Hence the number 25H in binary will be 0010 0101.

Similarly a number like 74H will be 0111 0100 in binary. 89H will be 1000 1001 and so on.

As you noticed, the numbers 25H, 74H and 89H are all 2 digits in hexadecimal and hence need 8 bits in binary. Such numbers are called 8 bit numbers. The range of 8 bit and 16 bit numbers is mentioned below:

#### 8 BIT NUMBERS (ALSO CALLED A “BYTE”)

HEX	BINARY
00 H	0000 0000
01 H	0000 0001
...	...
68 H	0110 1000
...	...
FE H	1111 1110
FF H	1111 1111

#### 16 BIT NUMBERS (ALSO CALLED A “WORD”)

HEX	BINARY
0000 H	0000 0000 0000 0000
0001 H	0000 0000 0000 0001
...	...
4831 H	0100 1000 0011 0001
...	...
FFFFE H	1111 1111 1111 1110
FFFF H	1111 1111 1111 1111

## ii) DATA BUS

**It carries data to and from the processor.**

The size of data bus determines how much data can be transferred in one operation (cycle).

Bigger the data bus, faster the processor, as it can transfer more data in one cycle.

## iii) CONTROL BUS

**It Carries control signals like RD, WR etc.**

These signals determine the kind of operation that will be performed on the system bus.

## **Bharat Acharya Education**

Learn...

8085 | 8086 | 80386 | Pentium |

8051 | ARM7 | COA

Fees: 1199/-

Duration: 6 months

Activation: Immediate

Certification: Yes

Free: PDFs of theory explanation

Free: VIVA questions and answers

Free: PDF of Multiple Choice Questions

Start Learning... NOW!

## **Bharat Acharya Education**

Order our Books here...

8086 Microprocessor book

Link: <https://amzn.to/3qHDpJH>

8051 Microcontroller book

Link: <https://amzn.to/3aFQkXc>

Official WhatsApp number:

**+91 9136428051**



## 8085 BASICS

### ADDRESS BUS

- 8085 has a **16 bit address bus**
- Hence it can access  $2^{16} = 64 \text{ KB Memory}$
- The memory has address range **0000H ... FFFFH**
- **In Each location 1 Byte of data is stored**
- If 16-bit data (2 bytes) needs to be stored then its stored at 2 consecutive locations
- **The rule followed is Lower Byte – Lower address**
- This is called the little endian rule.

Memory accessed by 8085

	Address	Data	
First location	0000 H	34 H	(any 8-bit number from 00H ... FFH)
	0001 H	12 H	
	0002 H		
Last location	FFFF H		

← 16-bit →      ← 8 bit →

### DATA BUS

- 8085 has an **8 bit data bus**
- This means in One Cycle, 8085 can transfer 8-bits of data

### ALU

- 8085 has an **8 bit ALU**
- This means in One Cycle, 8085 can operate on 8-bits of data
- This classifies 8085 as an 8-bit microprocessor



## CONTROL BUS

- Control bus releases control signals, that decide which operation must be performed.
- The main control signals are **IO/ M** , **RD** and **WR**
- **IO/ M** is “1” for I/O operations and “0” for memory operations
- **RD** is “0” for any “Read” operation
- **WR** is “0” for any “Write” operation

Machine Cycle	<b>IO/ M</b>	<b>RD</b>	<b>WR</b>
Memory Read	0	0	1
Memory Write	0	1	0
I/O Read	1	0	1
I/O Write	1	1	0

## Bharat Acharya Education

Learn...

8085 | 8086 | 80386 | Pentium |

8051 | ARM7 | COA

Fees: 1199/-

Duration: 6 months

Activation: Immediate

Certification: Yes

Free: PDFs of theory explanation

Free: VIVA questions and answers

Free: PDF of Multiple Choice Questions

Start Learning... NOW!

## Bharat Acharya Education

Order our Books here...

8086 Microprocessor book

Link: <https://amzn.to/3qHDpJH>

8051 Microcontroller book

Link: <https://amzn.to/3aFQkXc>

Official WhatsApp number:

**+91 9136428051**

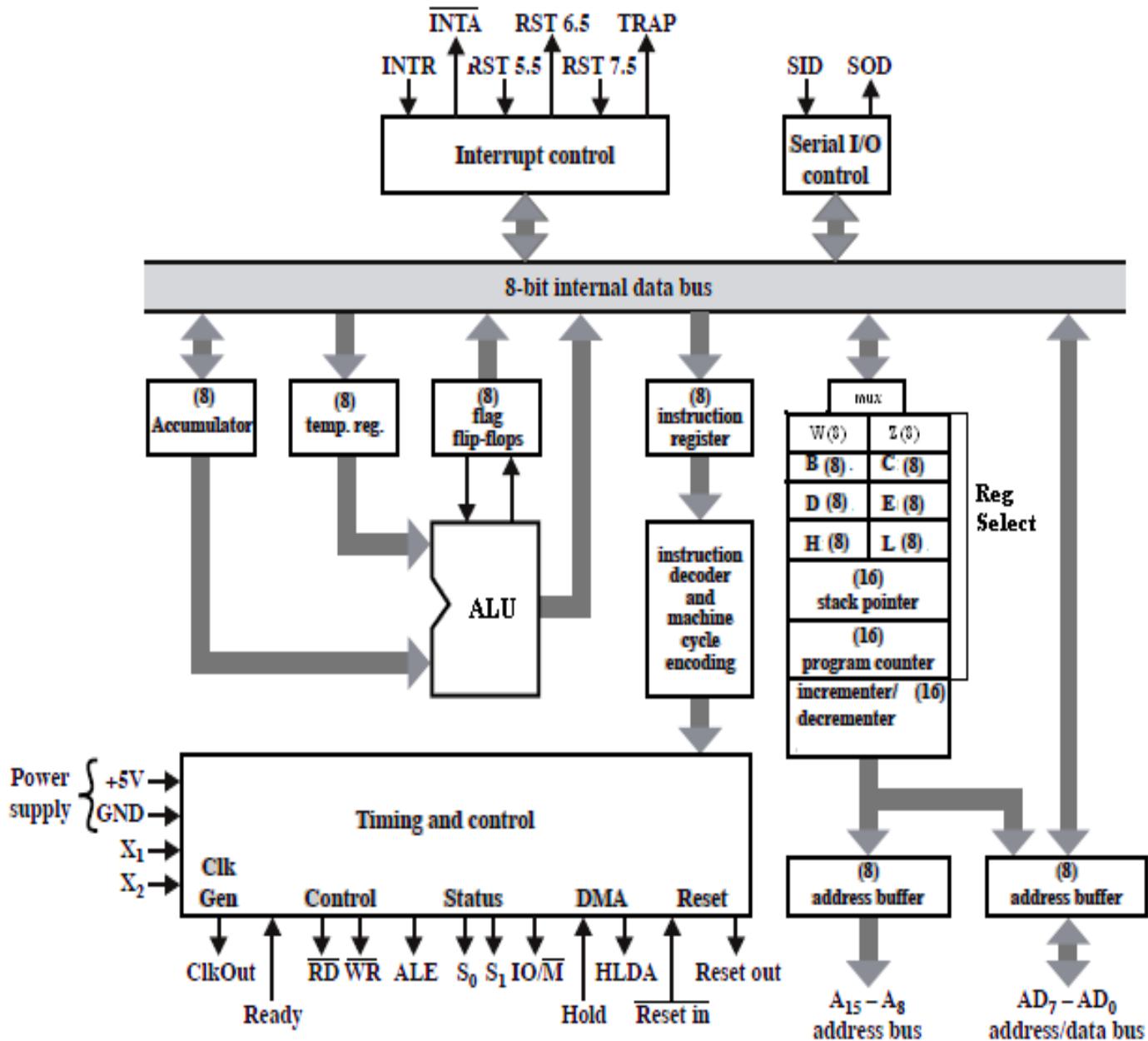
## | **8085 ARCHITECTURE, PINS AND FLAG REGISTER**

### **Note**

Dear Students, Architecture contains all the pins and the Flag Register. Hence, I have made a common PDF for all these three topics.

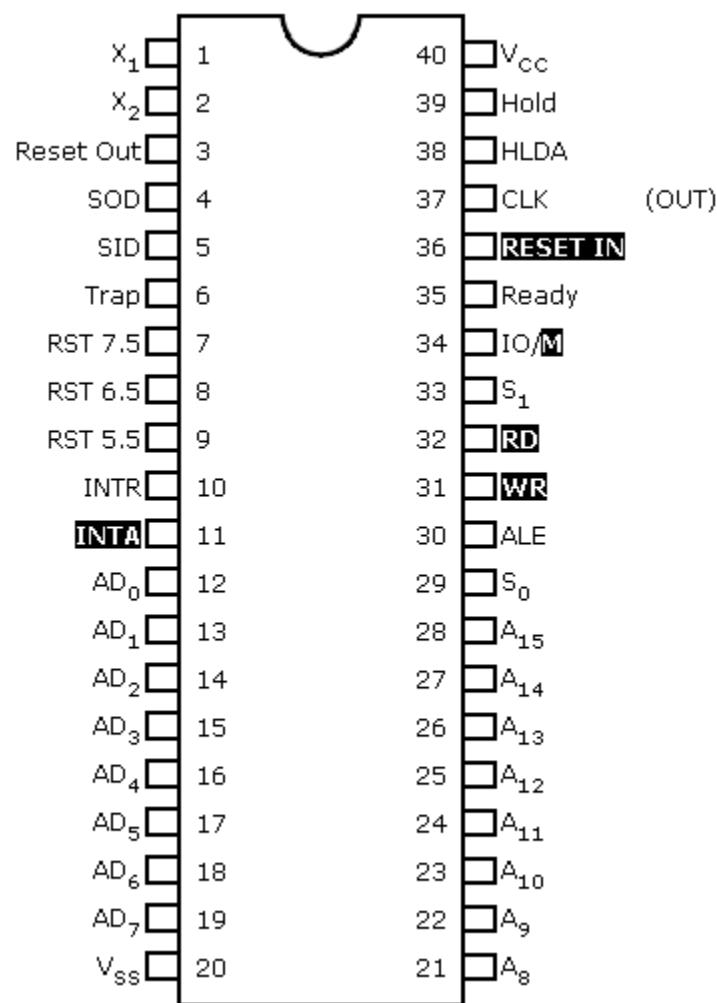


## 8085 ARCHITECTURE





## 8085 PIN DIAGRAM





## REGISTERS

### Program Counter (PC, 16-bits)

It is a 16-bit Special-Purpose register. It holds **address** of the **next instruction**.  
PC is incremented by the INR/DCR after every instruction byte is fetched.

### Stack Pointer (SP, 16-bits)

It is a 16-bit Special-Purpose register. It holds **address** of the **top of the Stack**.  
Stack is a set of memory locations operating in LIFO manner.  
SP is **decremented** on every **PUSH** operation and **incremented** on every **POP**.

### B, C, D, E, H, L (8-bits)

These are 8-bit General-Purpose registers.  
They can also be used to store 16-bit data in register pairs.  
The possible register **pairs** are **BC** pair, **DE** pair and **HL** pair.  
The **HL** pair also holds the **address** for the Memory Pointer "**M**".

### Temporary Register Pair (WZ, 16-bits)

This is a 16-bit register pair.  
It is **used by μP** to hold **temporary** values in some instructions like CALL/JMP/LDA etc.  
The **programmer** has **no access** to this register pair.

### INR/DCR Register (16-bits)

This is a 16-bit shift register.  
It is used to **increment PC after every instruction byte is fetched**  
It also **increments** or **decrements** **SP** after a Pop or a Push operation respectively.  
It is not available to the programmer.



## A - Accumulator (8-bits)

It is an 8-bit programmable register.

The user can read or write this register.

It has two **special properties** viz:

- It **holds the first operand** during most arithmetic operations.
- It **holds the result** of most of the arithmetic and logic operations

Eg: ADD B; This instruction will do A + B and store the result in A.

Eg: SUB B; This instruction will do A - B and store the result in A.

## Temp Register (8-bits)

This is an 8-bit register.

It is **used by μP** for storing one of the operands during an operation.

The **programmer** has **NO ACCESS** to this register.

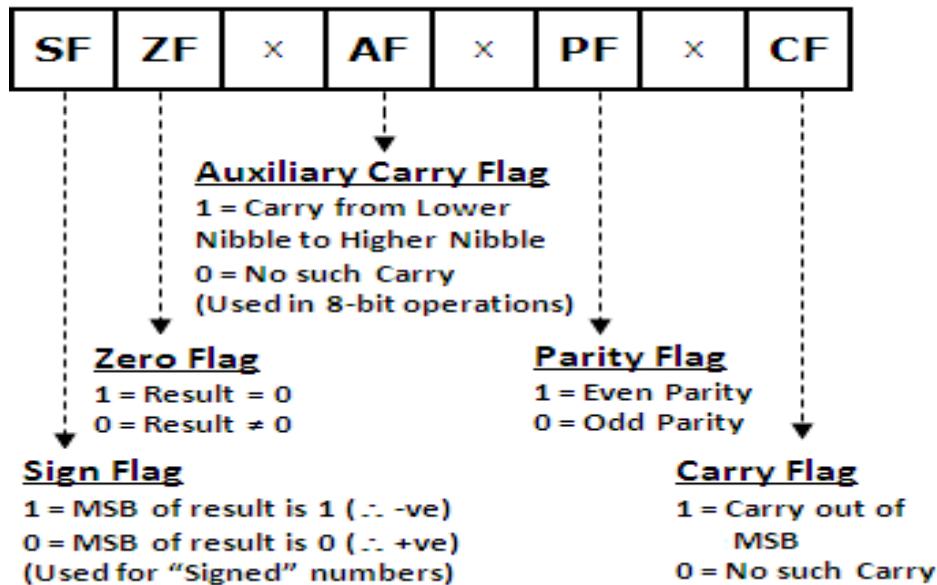
### Special Note:

If you are learning this by piracy, then you are not my student.

You are simply a thief! #PoorUpbringing



## 8085 FLAG REGISTER



### S - Sign Flag

It is **set** (1) when **MSB** of the result is **1** (i.e. result is a **-VE** number).

It is **reset** (0) when MSB of the result is 0 (i.e. result is a **+VE** number).

### Z - Zero Flag

It is **set** when the **result is = zero**.

It is **reset** when the result is not = zero.

### AC - Auxiliary Carry Flag

It is **set** when an **Auxiliary Carry / Borrow** is **generated**.

It is **reset** when an Auxiliary Carry / Borrow is not generated.

Auxiliary Carry is the Carry generated **between the lower nibble and the higher nibble** for an 8-bit operation. It is not affected after a 16-bit operation. It is used only in DAA operation.

### P - Parity Flag

It is **set (1)** when result has **even parity**. It is **reset** when result has odd parity.

## C - Carry Flag

It is **set** when a **Carry / Borrow is generated from the MSB.**

It is reset when a Carry / Borrow is not generated from the MSB.

# In the exam, Show at least 2 examples from Bharat Sir's video lecture

## INTERRUPT CONTROL

This Block is responsible for controlling the **hardware interrupts** of 8085.

8085 supports the following hardware interrupts:

### TRAP

This is an **edge as well as level triggered, vectored** interrupt.

It cannot be masked by SIM instruction and can neither be disabled by DI instruction.

It has the **highest priority**.

Its vector address is **0024H**.

### RST 7.5

This is an **edge triggered, vectored** interrupt.

It can be masked by SIM instruction and can also be disabled by DI instruction.

It has the **second highest priority**.

Its vector address is **003CH**.

### RST 6.5

This is a **level triggered, vectored** interrupt.

It can be masked by SIM instruction and can also be disabled by DI instruction.

It has the **third highest priority**.

Its vector address is **0034H**.

### RST 5.5

This is a **level triggered, vectored** interrupt.

It can be masked by SIM instruction and can also be disabled by DI instruction.

It has the **fourth highest priority**.

Its vector address is **002CH**.



## INTR

This is a **level triggered, non-vectored** interrupt.

It cannot be masked by SIM instruction but can be disabled by DI instruction.

It has the **lowest priority**.

It has an **acknowledgement signal INTA**.

The address for the ISR is **fetched from external hardware**.

## INTA

This is an **acknowledgement signal for INTR** (only).

This signal is used to **get** the Op-Code (and hence the ISR address) from External hardware in order to execute the ISR. ☺ In case of doubts, contact Bharat Sir: - 98204 08217.

**ALL** Interrupts **EXCEPT TRAP** can be **disabled** though the **DI** instruction.

These interrupts can be **enabled** again by the **EI** Instruction.

Interrupts can be individually **masked or unmasked by SIM instruction**.

TRAP and INTR are not affected by SIM instruction.

## SERIAL CONTROL

This Block is responsible for transferring data Serially to and from the  $\mu$ P.

### SID - Serial In Data

$\mu$ P receives data, bit-by-bit through this line.

### SOD - Serial Out Data

$\mu$ P sends out data, bit-by-bit through this line.

Serial transmission can be done by **RIM** and **SIM** Instructions.

## ALU

8085 has an **8-bit ALU**.

It performs 8-bit arithmetic operations like Addition and Subtraction.

It also performs logical operations like AND, OR, EX-OR NOT etc.

It takes **input** from the **Accumulator** and the **Temp** register.

The **output** of most of the ALU operations is stored back **into the Accumulator**.



## INSTRUCTION REGISTER AND DECODER

### Instruction Register

The 8085 places the contents of the PC onto the Address bus and fetches the instruction.

This fetched instruction is stored into the Instruction register.

### Instruction Decoder:

The fetched instruction from the Instruction register enters the Instruction Decoder. Here the instruction is decoded and the decode information is given to the Timing and Control Circuit where the instruction is executed.

## TIMING AND CONTROL CIRCUIT

The timing and control circuit issues the various internal and external control signals for executing an instruction.

The external pins connected to this circuit are as follows:

### X1 and X2

These pins provide the **Clock Input to the µP**.

Clock is provided from a crystal oscillator.

### ClkOut

8085 provides the **Clock input to all other peripherals** through the ClockOut pin. This takes care of **synchronizing** all peripherals with 8085.

### ResetIn

This is an active low signal activated when the manual reset signal is applied to the µP. This signal **resets the µP**. On Reset PC contains **0000H**. Hence, the **Reset Vector Address** of 8085 is 0000H.

### ResetOut

This signal is connected to the reset input of all the peripherals. It is used to **reset the peripherals once the µP is reset**.



## READY

This is an active high input.

It is used to **synchronize** the **μP** with "**Slower**" Peripherals.

The **μP samples** the **Ready** input in the beginning of every Machine Cycle.

If it is found to be **LOW**, the **μP executes** one **WAIT CYCLE** after which it re-samples the ready pin till it finds the Ready pin **HIGH**.

∴ The **μP remains** in the **WAIT STATE** until the **READY** pin becomes **high** again.

Hence, if the **Ready** pin is **not required** it should be **connected** to the **Vcc**, and not, left unconnected, **otherwise** would cause the **μP** to execute **infinite wait cycles**.

#Please refer Bharat Sir's video Lecture for this ...

## ALE - Address Latch Enable

This signal is **used to latch address** from the multiplexed Address-Data Bus (**AD0-AD7**). When the Bus contains **address**, **ALE** is **high**, **else** it is **low**.

## IO/ M

This signal is used to distinguish between an **IO** and a **Memory operation**. When this signal is high it is an IO operation else it is a Memory operation.

## RD

This is an active low signal used to indicate a **read operation**.

## WR

This is an active low signal used to indicate a **write operation**.

## S<sub>1</sub> and S<sub>0</sub>

These lines denote the status of the **μP**

S <sub>1</sub> S <sub>0</sub>	Status
0 0	Idle
0 1	Write
1 0	Read
1 1	Opcode fetch

## **HOLD and HLDA**

The Hold and Hold Acknowledge signals are used for **Direct Memory Access** (DMA).

The **DMA Controller issued** the **Hold** signal to the **μP**.

In response the **μP releases** the **System bus**.

After releasing the system bus the **μP acknowledges** the Hold signal with **HLDA** signal. The **DMA Transfer** thus **begins**.

**DMA Transfer** is **terminated** by **releasing** the **HOLD** signal.

## **Bharat Acharya Education**

Learn...

8085 | 8086 | 80386 | Pentium |

8051 | ARM7 | COA

Fees: 1199/-

Duration: 6 months

Activation: Immediate

Certification: Yes

Free: PDFs of theory explanation

Free: VIVA questions and answers

Free: PDF of Multiple Choice Questions

Start Learning... NOW!

## **Bharat Acharya Education**

Order our Books here...

8086 Microprocessor book

Link: <https://amzn.to/3qHDpJH>

8051 Microcontroller book

Link: <https://amzn.to/3aFQkXc>

Official WhatsApp number:

**+91 9136428051**

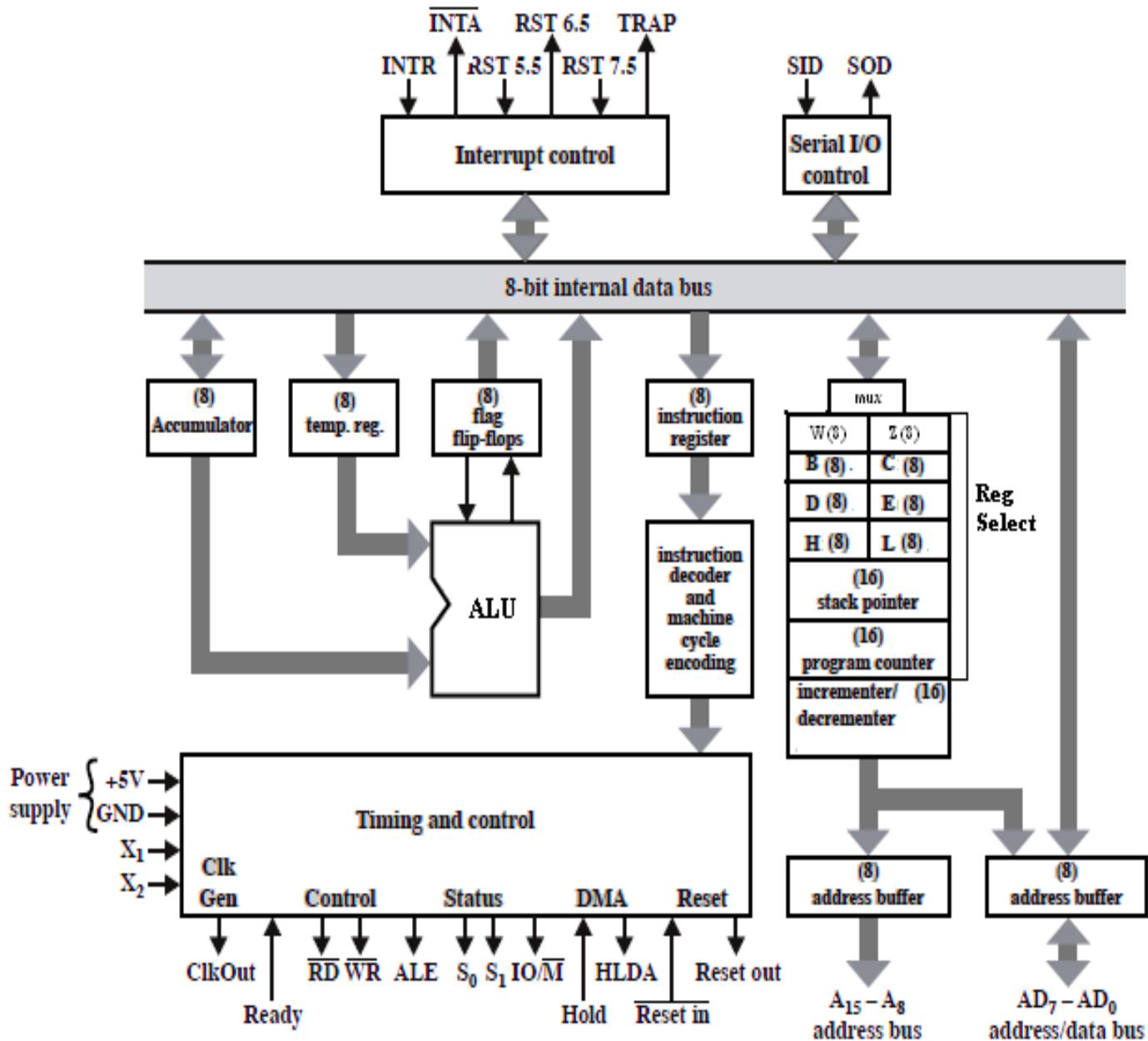
## | **8085 ARCHITECTURE, PINS AND FLAG REGISTER**

### **Note**

Dear Students, Architecture contains all the pins and the Flag Register. Hence, I have made a common PDF for all these three topics.

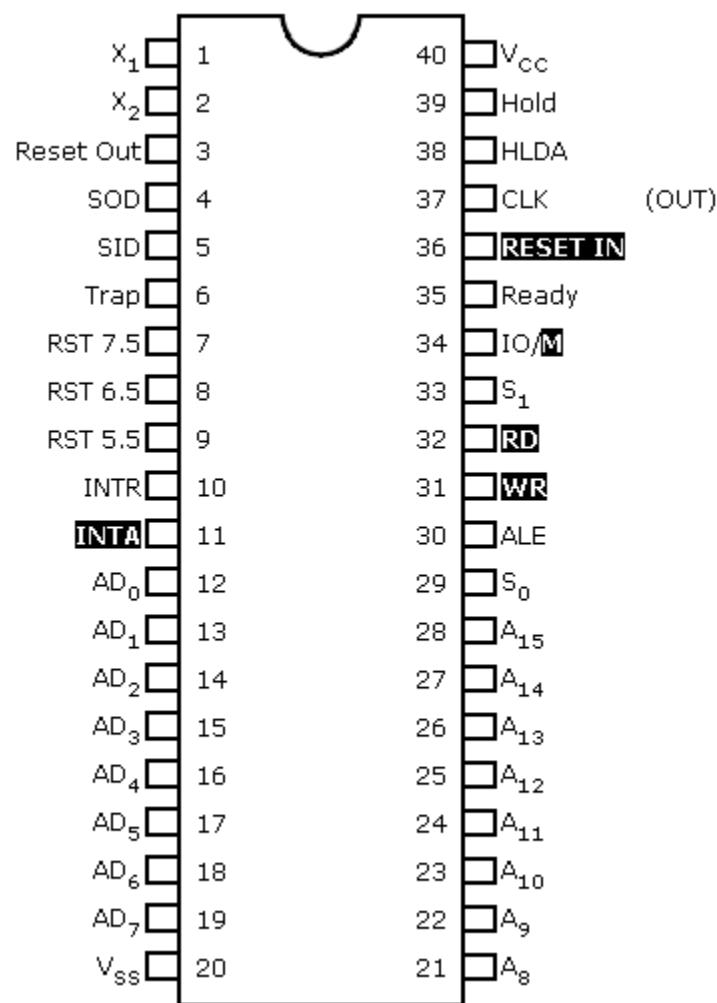


## 8085 ARCHITECTURE





## 8085 PIN DIAGRAM





## REGISTERS

### Program Counter (PC, 16-bits)

It is a 16-bit Special-Purpose register. It holds **address** of the **next instruction**.  
PC is incremented by the INR/DCR after every instruction byte is fetched.

### Stack Pointer (SP, 16-bits)

It is a 16-bit Special-Purpose register. It holds **address** of the **top of the Stack**.  
Stack is a set of memory locations operating in LIFO manner.  
SP is **decremented** on every **PUSH** operation and **incremented** on every **POP**.

### B, C, D, E, H, L (8-bits)

These are 8-bit General-Purpose registers.  
They can also be used to store 16-bit data in register pairs.  
The possible register **pairs** are **BC** pair, **DE** pair and **HL** pair.  
The **HL** pair also holds the **address** for the Memory Pointer "**M**".

### Temporary Register Pair (WZ, 16-bits)

This is a 16-bit register pair.  
It is **used by μP** to hold **temporary** values in some instructions like CALL/JMP/LDA etc.  
The **programmer** has **no access** to this register pair.

### INR/DCR Register (16-bits)

This is a 16-bit shift register.  
It is used to **increment PC after every instruction byte is fetched**  
It also **increments** or **decrements** **SP** after a Pop or a Push operation respectively.  
It is not available to the programmer.



## A - Accumulator (8-bits)

It is an 8-bit programmable register.

The user can read or write this register.

It has two **special properties** viz:

- It **holds the first operand** during most arithmetic operations.
- It **holds the result** of most of the arithmetic and logic operations

Eg: ADD B; This instruction will do A + B and store the result in A.

Eg: SUB B; This instruction will do A - B and store the result in A.

## Temp Register (8-bits)

This is an 8-bit register.

It is **used by μP** for storing one of the operands during an operation.

The **programmer** has **NO ACCESS** to this register.

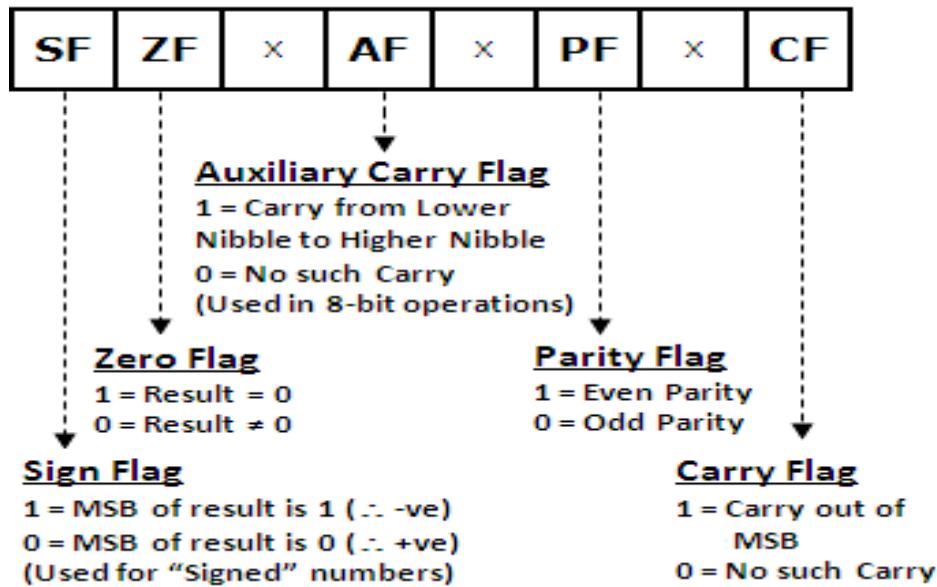
### Special Note:

If you are learning this by piracy, then you are not my student.

You are simply a thief! #PoorUpbringing



## 8085 FLAG REGISTER



### S - Sign Flag

It is **set** (1) when **MSB** of the result is **1** (i.e. result is a **-VE** number).

It is **reset** (0) when MSB of the result is 0 (i.e. result is a **+VE** number).

### Z - Zero Flag

It is **set** when the **result is = zero**.

It is **reset** when the result is not = zero.

### AC - Auxiliary Carry Flag

It is **set** when an **Auxiliary Carry / Borrow** is **generated**.

It is **reset** when an Auxiliary Carry / Borrow is not generated.

Auxiliary Carry is the Carry generated **between the lower nibble and the higher nibble** for an 8-bit operation. It is not affected after a 16-bit operation. It is used only in DAA operation.

### P - Parity Flag

It is **set (1)** when result has **even parity**. It is **reset** when result has odd parity.

## C - Carry Flag

It is **set** when a **Carry / Borrow is generated from the MSB.**

It is reset when a Carry / Borrow is not generated from the MSB.

# In the exam, Show at least 2 examples from Bharat Sir's video lecture

## INTERRUPT CONTROL

This Block is responsible for controlling the **hardware interrupts** of 8085.

8085 supports the following hardware interrupts:

### TRAP

This is an **edge as well as level triggered, vectored** interrupt.

It cannot be masked by SIM instruction and can neither be disabled by DI instruction.

It has the **highest priority**.

Its vector address is **0024H**.

### RST 7.5

This is an **edge triggered, vectored** interrupt.

It can be masked by SIM instruction and can also be disabled by DI instruction.

It has the **second highest priority**.

Its vector address is **003CH**.

### RST 6.5

This is a **level triggered, vectored** interrupt.

It can be masked by SIM instruction and can also be disabled by DI instruction.

It has the **third highest priority**.

Its vector address is **0034H**.

### RST 5.5

This is a **level triggered, vectored** interrupt.

It can be masked by SIM instruction and can also be disabled by DI instruction.

It has the **fourth highest priority**.

Its vector address is **002CH**.



## INTR

This is a **level triggered, non-vectored** interrupt.

It cannot be masked by SIM instruction but can be disabled by DI instruction.

It has the **lowest priority**.

It has an **acknowledgement signal INTA**.

The address for the ISR is **fetched from external hardware**.

## INTA

This is an **acknowledgement signal for INTR** (only).

This signal is used to **get** the Op-Code (and hence the ISR address) from External hardware in order to execute the ISR. ☺ In case of doubts, contact Bharat Sir: - 98204 08217.

**ALL** Interrupts **EXCEPT TRAP** can be **disabled** though the **DI** instruction.

These interrupts can be **enabled** again by the **EI** Instruction.

Interrupts can be individually **masked or unmasked by SIM instruction**.

TRAP and INTR are not affected by SIM instruction.

## SERIAL CONTROL

This Block is responsible for transferring data Serially to and from the  $\mu$ P.

### SID - Serial In Data

$\mu$ P receives data, bit-by-bit through this line.

### SOD - Serial Out Data

$\mu$ P sends out data, bit-by-bit through this line.

Serial transmission can be done by **RIM** and **SIM** Instructions.

## ALU

8085 has an **8-bit ALU**.

It performs 8-bit arithmetic operations like Addition and Subtraction.

It also performs logical operations like AND, OR, EX-OR NOT etc.

It takes **input** from the **Accumulator** and the **Temp** register.

The **output** of most of the ALU operations is stored back **into the Accumulator**.



## INSTRUCTION REGISTER AND DECODER

### Instruction Register

The 8085 places the contents of the PC onto the Address bus and fetches the instruction.

This fetched instruction is stored into the Instruction register.

### Instruction Decoder:

The fetched instruction from the Instruction register enters the Instruction Decoder. Here the instruction is decoded and the decode information is given to the Timing and Control Circuit where the instruction is executed.

## TIMING AND CONTROL CIRCUIT

The timing and control circuit issues the various internal and external control signals for executing an instruction.

The external pins connected to this circuit are as follows:

### X1 and X2

These pins provide the **Clock Input to the µP**.

Clock is provided from a crystal oscillator.

### ClkOut

8085 provides the **Clock input to all other peripherals** through the ClockOut pin. This takes care of **synchronizing** all peripherals with 8085.

### ResetIn

This is an active low signal activated when the manual reset signal is applied to the µP. This signal **resets the µP**. On Reset PC contains **0000H**. Hence, the **Reset Vector Address** of 8085 is 0000H.

### ResetOut

This signal is connected to the reset input of all the peripherals. It is used to **reset the peripherals once the µP is reset**.



## READY

This is an active high input.

It is used to **synchronize** the **μP** with "**Slower**" Peripherals.

The **μP samples** the **Ready** input in the beginning of every Machine Cycle.

If it is found to be **LOW**, the **μP executes** one **WAIT CYCLE** after which it re-samples the ready pin till it finds the Ready pin **HIGH**.

∴ The **μP remains** in the **WAIT STATE** until the **READY** pin becomes **high** again.

Hence, if the **Ready** pin is **not required** it should be **connected** to the **Vcc**, and not, left unconnected, **otherwise** would cause the **μP** to execute **infinite wait cycles**.

#Please refer Bharat Sir's video Lecture for this ...

## ALE - Address Latch Enable

This signal is **used to latch address** from the multiplexed Address-Data Bus (**AD0-AD7**). When the Bus contains **address**, **ALE** is **high**, **else** it is **low**.

## IO/ M

This signal is used to distinguish between an **IO** and a **Memory operation**. When this signal is high it is an IO operation else it is a Memory operation.

## RD

This is an active low signal used to indicate a **read operation**.

## WR

This is an active low signal used to indicate a **write operation**.

## S<sub>1</sub> and S<sub>0</sub>

These lines denote the status of the **μP**

S <sub>1</sub> S <sub>0</sub>	Status
0 0	Idle
0 1	Write
1 0	Read
1 1	Opcode fetch

## **HOLD and HLDA**

The Hold and Hold Acknowledge signals are used for **Direct Memory Access** (DMA).

The **DMA Controller issued** the **Hold** signal to the **μP**.

In response the **μP releases** the **System bus**.

After releasing the system bus the **μP acknowledges** the Hold signal with **HLDA** signal. The **DMA Transfer** thus **begins**.

**DMA Transfer** is **terminated** by **releasing** the **HOLD** signal.

## **Bharat Acharya Education**

Learn...

8085 | 8086 | 80386 | Pentium |

8051 | ARM7 | COA

Fees: 1199/-

Duration: 6 months

Activation: Immediate

Certification: Yes

Free: PDFs of theory explanation

Free: VIVA questions and answers

Free: PDF of Multiple Choice Questions

Start Learning... NOW!

## **Bharat Acharya Education**

Order our Books here...

8086 Microprocessor book

Link: <https://amzn.to/3qHDpJH>

8051 Microcontroller book

Link: <https://amzn.to/3aFQkXc>

Official WhatsApp number:

**+91 9136428051**

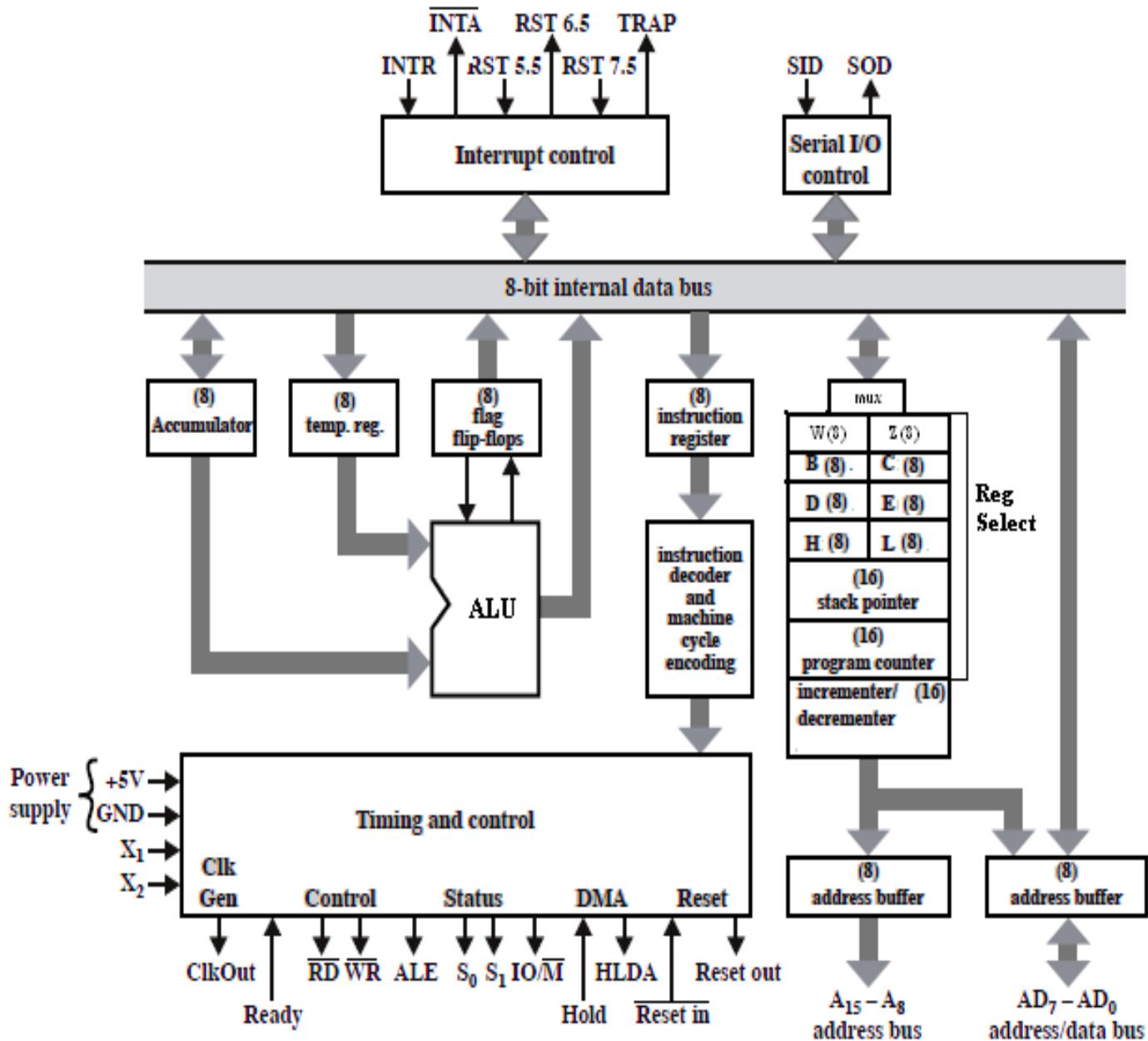
## | **8085 ARCHITECTURE, PINS AND FLAG REGISTER**

### **Note**

Dear Students, Architecture contains all the pins and the Flag Register. Hence, I have made a common PDF for all these three topics.

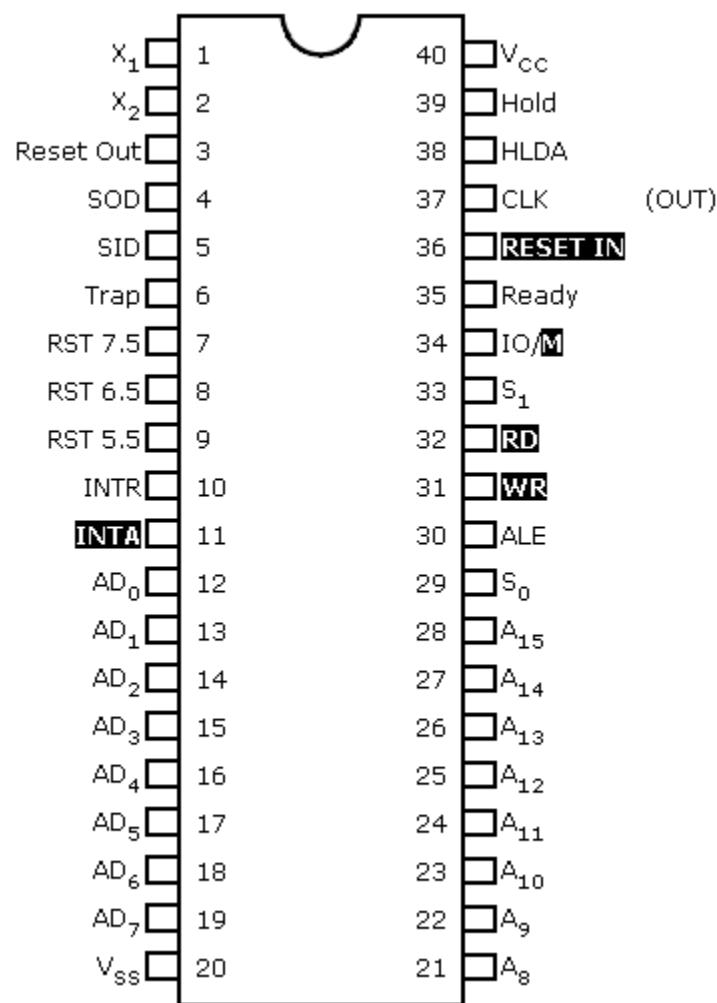


## 8085 ARCHITECTURE





## 8085 PIN DIAGRAM





## REGISTERS

### Program Counter (PC, 16-bits)

It is a 16-bit Special-Purpose register. It holds **address** of the **next instruction**.  
PC is incremented by the INR/DCR after every instruction byte is fetched.

### Stack Pointer (SP, 16-bits)

It is a 16-bit Special-Purpose register. It holds **address** of the **top of the Stack**.  
Stack is a set of memory locations operating in LIFO manner.  
SP is **decremented** on every **PUSH** operation and **incremented** on every **POP**.

### B, C, D, E, H, L (8-bits)

These are 8-bit General-Purpose registers.  
They can also be used to store 16-bit data in register pairs.  
The possible register **pairs** are **BC** pair, **DE** pair and **HL** pair.  
The **HL** pair also holds the **address** for the Memory Pointer "**M**".

### Temporary Register Pair (WZ, 16-bits)

This is a 16-bit register pair.  
It is **used by μP** to hold **temporary** values in some instructions like CALL/JMP/LDA etc.  
The **programmer** has **no access** to this register pair.

### INR/DCR Register (16-bits)

This is a 16-bit shift register.  
It is used to **increment PC after every instruction byte is fetched**  
It also **increments** or **decrements** **SP** after a Pop or a Push operation respectively.  
It is not available to the programmer.



## A - Accumulator (8-bits)

It is an 8-bit programmable register.

The user can read or write this register.

It has two **special properties** viz:

- It **holds the first operand** during most arithmetic operations.
- It **holds the result** of most of the arithmetic and logic operations

Eg: ADD B; This instruction will do A + B and store the result in A.

Eg: SUB B; This instruction will do A - B and store the result in A.

## Temp Register (8-bits)

This is an 8-bit register.

It is **used by μP** for storing one of the operands during an operation.

The **programmer** has **NO ACCESS** to this register.

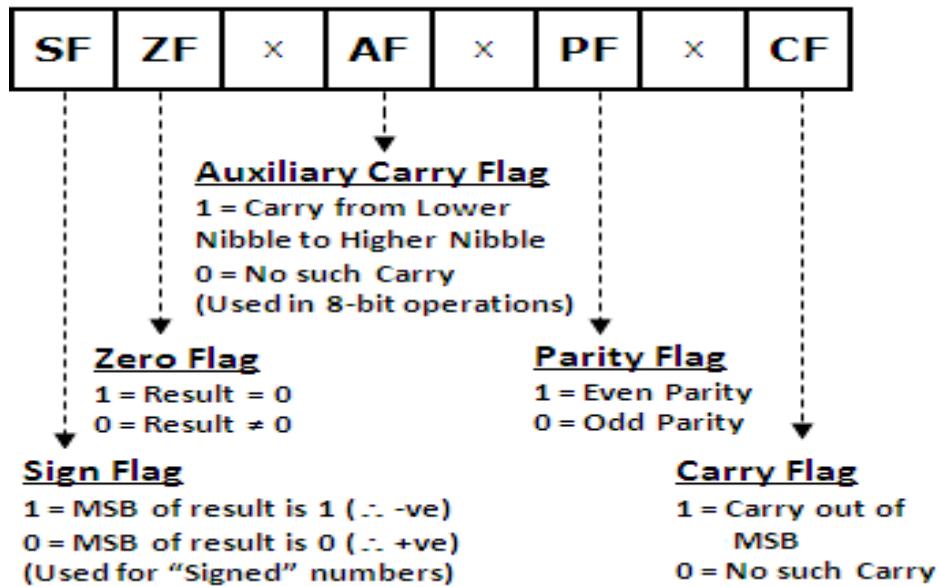
### Special Note:

If you are learning this by piracy, then you are not my student.

You are simply a thief! #PoorUpbringing



## 8085 FLAG REGISTER



### S - Sign Flag

It is **set** (1) when **MSB** of the result is **1** (i.e. result is a **-VE** number).

It is **reset** (0) when MSB of the result is 0 (i.e. result is a **+VE** number).

### Z - Zero Flag

It is **set** when the **result is = zero**.

It is **reset** when the result is not = zero.

### AC - Auxiliary Carry Flag

It is **set** when an **Auxiliary Carry / Borrow** is **generated**.

It is **reset** when an Auxiliary Carry / Borrow is not generated.

Auxiliary Carry is the Carry generated **between the lower nibble and the higher nibble** for an 8-bit operation. It is not affected after a 16-bit operation. It is used only in DAA operation.

### P - Parity Flag

It is **set (1)** when result has **even parity**. It is **reset** when result has odd parity.

## C - Carry Flag

It is **set** when a **Carry / Borrow is generated from the MSB.**

It is reset when a Carry / Borrow is not generated from the MSB.

# In the exam, Show at least 2 examples from Bharat Sir's video lecture

## INTERRUPT CONTROL

This Block is responsible for controlling the **hardware interrupts** of 8085.

8085 supports the following hardware interrupts:

### TRAP

This is an **edge as well as level triggered, vectored** interrupt.

It cannot be masked by SIM instruction and can neither be disabled by DI instruction.

It has the **highest priority**.

Its vector address is **0024H**.

### RST 7.5

This is an **edge triggered, vectored** interrupt.

It can be masked by SIM instruction and can also be disabled by DI instruction.

It has the **second highest priority**.

Its vector address is **003CH**.

### RST 6.5

This is a **level triggered, vectored** interrupt.

It can be masked by SIM instruction and can also be disabled by DI instruction.

It has the **third highest priority**.

Its vector address is **0034H**.

### RST 5.5

This is a **level triggered, vectored** interrupt.

It can be masked by SIM instruction and can also be disabled by DI instruction.

It has the **fourth highest priority**.

Its vector address is **002CH**.



## INTR

This is a **level triggered, non-vectored** interrupt.

It cannot be masked by SIM instruction but can be disabled by DI instruction.

It has the **lowest priority**.

It has an **acknowledgement signal INTA**.

The address for the ISR is **fetched from external hardware**.

## INTA

This is an **acknowledgement signal for INTR** (only).

This signal is used to **get** the Op-Code (and hence the ISR address) from External hardware in order to execute the ISR. ☺ In case of doubts, contact Bharat Sir: - 98204 08217.

**ALL** Interrupts **EXCEPT TRAP** can be **disabled** though the **DI** instruction.

These interrupts can be **enabled** again by the **EI** Instruction.

Interrupts can be individually **masked or unmasked by SIM instruction**.

TRAP and INTR are not affected by SIM instruction.

## SERIAL CONTROL

This Block is responsible for transferring data Serially to and from the  $\mu$ P.

### SID - Serial In Data

$\mu$ P receives data, bit-by-bit through this line.

### SOD - Serial Out Data

$\mu$ P sends out data, bit-by-bit through this line.

Serial transmission can be done by **RIM** and **SIM** Instructions.

## ALU

8085 has an **8-bit ALU**.

It performs 8-bit arithmetic operations like Addition and Subtraction.

It also performs logical operations like AND, OR, EX-OR NOT etc.

It takes **input** from the **Accumulator** and the **Temp** register.

The **output** of most of the ALU operations is stored back **into the Accumulator**.



## INSTRUCTION REGISTER AND DECODER

### Instruction Register

The 8085 places the contents of the PC onto the Address bus and fetches the instruction.

This fetched instruction is stored into the Instruction register.

### Instruction Decoder:

The fetched instruction from the Instruction register enters the Instruction Decoder. Here the instruction is decoded and the decode information is given to the Timing and Control Circuit where the instruction is executed.

## TIMING AND CONTROL CIRCUIT

The timing and control circuit issues the various internal and external control signals for executing and instruction.

The external pins connected to this circuit are as follows:

### X1 and X2

These pins provide the **Clock Input to the µP**.

Clock is provided from a crystal oscillator.

### ClkOut

8085 provides the **Clock input to all other peripherals** through the ClockOut pin. This takes care of **synchronizing** all peripherals with 8085.

### ResetIn

This is an active low signal activated when the manual reset signal is applied to the µP. This signal **resets the µP**. On Reset PC contains **0000H**. Hence, the **Reset Vector Address** of 8085 is 0000H.

### ResetOut

This signal is connected to the reset input of all the peripherals. It is used to **reset the peripherals once the µP is reset**.



## READY

This is an active high input.

It is used to **synchronize** the **μP** with "**Slower**" Peripherals.

The **μP samples** the **Ready** input in the beginning of every Machine Cycle.

If it is found to be **LOW**, the **μP executes** one **WAIT CYCLE** after which it re-samples the ready pin till it finds the Ready pin **HIGH**.

∴ The **μP remains** in the **WAIT STATE** until the **READY** pin becomes **high** again.

Hence, if the **Ready** pin is **not required** it should be **connected** to the **Vcc**, and not, left unconnected, **otherwise** would cause the **μP** to execute **infinite wait cycles**.

#Please refer Bharat Sir's video Lecture for this ...

## ALE - Address Latch Enable

This signal is **used to latch address** from the multiplexed Address-Data Bus (**AD0-AD7**). When the Bus contains **address**, **ALE** is **high**, **else** it is **low**.

## IO/ M

This signal is used to distinguish between an **IO** and a **Memory operation**. When this signal is high it is an IO operation else it is a Memory operation.

## RD

This is an active low signal used to indicate a **read operation**.

## WR

This is an active low signal used to indicate a **write operation**.

## S<sub>1</sub> and S<sub>0</sub>

These lines denote the status of the **μP**

S <sub>1</sub> S <sub>0</sub>	Status
0 0	Idle
0 1	Write
1 0	Read
1 1	Opcode fetch

## **HOLD and HLDA**

The Hold and Hold Acknowledge signals are used for **Direct Memory Access** (DMA).

The **DMA Controller issued** the **Hold** signal to the **μP**.

In response the **μP releases** the **System bus**.

After releasing the system bus the **μP acknowledges** the Hold signal with **HLDA** signal. The **DMA Transfer** thus **begins**.

**DMA Transfer** is **terminated** by **releasing** the **HOLD** signal.

## **Bharat Acharya Education**

Learn...

8085 | 8086 | 80386 | Pentium |

8051 | ARM7 | COA

Fees: 1199/-

Duration: 6 months

Activation: Immediate

Certification: Yes

Free: PDFs of theory explanation

Free: VIVA questions and answers

Free: PDF of Multiple Choice Questions

Start Learning... NOW!

## **Bharat Acharya Education**

Order our Books here...

8086 Microprocessor book

Link: <https://amzn.to/3qHDpJH>

8051 Microcontroller book

Link: <https://amzn.to/3aFQkXc>

Official WhatsApp number:

**+91 9136428051**



## 8085 ADDRESSING MODES

Addressing Modes are the different ways by which the µP address (specifies) the operands in an instruction. 8085 supports the following Addressing Modes:

### 1) Immediate Addressing Mode

In this mode, the **Data** is specified **in the Instruction** itself.

- Eg:      **MVI A, 35H**                          ; Move immediately the value 35 into the Accumulator.  
    ; i.e. A ← 35H  
    **LXI B, 4000H**                                  ; Move immediately the value 4000 into the register pair BC.  
    ; i.e. BC ← 4000H

**Advantage:**

Programmer can easily **identify** the **operands**.

**Disadvantage:**

Always more than one byte hence requires **more space**.

The µP requires **two or three machine cycles** to fetch the instruction hence **slow**.

**Special Notes:**

The “I” in the instruction indicates “Immediate Addressing Mode”

Hence the number in the instruction must be DATA.

Hereafter, when you see a number in any instruction look for an “I”.

If “I” is present then the number is DATA else its an address.

The “X” in the instruction (LXI) indicates “Register Pair”.

### 2) Register Addressing Mode

In this mode, the **Data** is specified **in Registers**.

- Eg:      **MOV B, C**                                  ; Move the Contents of C-Register into B-Register.  
    ; i.e. B ← C  
    **INR B**    ; Increments the contents of B-Register.  
    ; i.e. B ← B + 1

**Advantage:**

The µP requires **only one machine cycle** to Fetch the instruction.

**Disadvantage:**

Operands **cannot** be easily **identified**.

### 3) Direct Addressing Mode

In this mode, the **Address** of the operand is specified **in the Instruction** itself.



Eg: <b>LDA 2000H</b>	; Loads the Accumulator with the Contents of Location 2000. ; i.e. A $\leftarrow [2000]$
<b>STA 2000H</b>	; Stores the Contents of the Accumulator at the Location 2000. ; i.e. [2000] $\leftarrow A$

**Advantage:**

The programmer can identify the address of the operand.

**Disadvantage:**

These are three byte instructions hence require three fetch cycles.

#### 4) Indirect Addressing Mode

In this mode, the **Address** of the operand is specified in Registers.

Hence, the instruction indirectly points to the operands.

Even the Memory Pointer "M" can be used as it is pointed by the HL register pair.

Eg: <b>STAX B</b>	; Stores the contents of the Accumulator at the location ; pointed by the contents of BC pair. ; i.e. [[BC]] $\leftarrow A$ . ; So if contents of BC pair = 4000 i.e. [BC] = 4000 then ; [4000] $\leftarrow A$ . #Please refer Bharat Sir's Lecture Notes for this ...
<b>INR M</b>	; Increments the contents of the location pointed by HL pair ; (i.e. M) i.e. [[HL]] $\leftarrow [[HL]] + 1$

**Advantage:**

Address of the operand is not fixed and hence can be used in a loop.

Size of the instruction is small as compared to direct addressing mode.

**Disadvantage:**

Requires initialization of the register pair hence requires atleast one more instruction.

**Special Notes:**

Remember, during programming when you want to access only 1 or 2 locations, use Direct addressing mode as it is simpler.

But when you want to access a series of locations, use Indirect addressing mode.

Initialize the first address in a register pair.

Thereafter increment/decrement that pair in a loop to access a series of locations.

#### 5) Implied Addressing Mode

In this mode, the **Operand** is implied in the instruction.

This instruction will work only on that implied operand, and not on any other operand.

Eg: <b>STC</b>	; Sets the Carry Flag in the Flag register. ; Cy $\leftarrow 1$ .
<b>CMC</b>	; Complements the Carry Flag in the Flag register.



**Advantage:**

Instructions are generally **only one byte**.

**Disadvantage:**

Programmer **cannot** easily **identify** the value of the operand.

## **Bharat Acharya Education**

Learn...

8085 | 8086 | 80386 | Pentium |

8051 | ARM7 | COA

Fees: 1199/-

Duration: 6 months

Activation: Immediate

Certification: Yes

Free: PDFs of theory explanation

Free: VIVA questions and answers

Free: PDF of Multiple Choice Questions

Start Learning... NOW!

## **Bharat Acharya Education**

Order our Books here...

8086 Microprocessor book

Link: <https://amzn.to/3qHDpJH>

8051 Microcontroller book

Link: <https://amzn.to/3aFQkXc>

Official WhatsApp number:

**+91 9136428051**

## 8085 CLASSIFICATION OF INSTRUCTIONS

The Instruction Set of 8085 is classified into the following 5 groups:

### 1) Data Transfer Group

This group of instructions transfer data from one place to another place.

The data can be transferred from Register to Register, Memory to Register & Register to Memory

- Eg: **MOV A, B** ; Content of B register is transferred to accumulator.  
**MVI A, 03H** ; Move immediately 03 into the accumulator.

### 2) Arithmetic Group

This group of instructions performs arithmetic operation on the data.

The arithmetic operation may be data addition, subtraction, increment, decrement etc.

- Eg: **ADI 03H** ; Add immediately 03 with the content of accumulator, store  
**SUB B** ; result in accumulator.  
;b Subtract content of B Register from the accumulator, store  
;b result in accumulator.

### 3) Logic Group

This group of instructions performs logical operations on the data.

These instructions include AND, OR, EX-OR etc.

- Eg: **ANI 03H** ; AND logically 03 with the content of accumulator, store result  
**ORA B** ; in accumulator.  
;b OR logically content of B Register with the accumulator, store  
;b result in accumulator.

### 4) Branch Group

These instructions change the sequence of flow of the program.

There are two types of Branch Instructions

#### Conditional Branch instructions

In this type, control is transferred to another memory location only if a particular condition is satisfied.

- Eg: **JNZ 4000** ; Program Control is transferred to memory location 4000 provided result of  
;b the previous operation is non-zero else control continues sequentially.

#### Unconditional Branch instruction

In this type, control is transferred to another memory location without any condition.

- Eg: **JMP 4000** ; Simply control is transferred to the memory location 4000.

### 5) Stack, IO and Machine Control Instruction

These instructions control the STACK (Push and Pop), I/O (Input and Output) and Machine hardware.

- Eg: **PUSH B** ; The content of BC register pair is pushed to the stack, at the  
;b location pointed by SP.  
**IN 80H** ; The data from the input device at port address 80 is placed in  
;b accumulator.



## INSTRUCTION SET OF 8085

### Common Notations:

- 1) Addr → 16 bit address.
- 2) Data → 8 or 16 bit data.
- 3) R → one of the registers.
- 4) Rp → register pair. BC pair is called B, DE → D and HL → L

### Special Notes:

In the explanation of every instruction, I have mentioned its machine cycles and T-states. You will understand this part once you watch the two videos of timing diagrams

## DATA TRANSFER GROUP

### 1) MOV R<sub>Destination</sub>, R<sub>source</sub>

The contents of register R<sub>source</sub> is copied into register R<sub>Destination</sub>.

Eg: **MOV A,B** ; A ← B

Addr. Mode	Flags Affected	Cycles	T-States
Register	None	1	4

### 2) MOV R, M

The contents of the memory location pointed by HL (memory pointer) is copied into register R.

Eg: **MOV B,M** ; B ← [[HL]] i.e. B ← M

Addr. Mode	Flags Affected	Cycles	T-States
Indirect	None	2	7

### 3) MOV M, R

The contents of register R is copied into the memory location pointed by HL (memory pointer).

Eg: **MOV M,B** ; [[HL]] ← B i.e. M ← B

Addr. Mode	Flags Affected	Cycles	T-States
Register	None	2	7

### 4) MVI R, 8-bit data

The 8-bit data is immediately moved into the register specified in the instruction.

Eg: **MVI C, 23** ; C ← 23H

Addr. Mode	Flags Affected	Cycles	T-States
Immediate	None	2	7



## 5) LXI Rp, 16-bit data

The 16-bit data is immediately moved into the register pair specified in the instruction.

**Eg:** MVI B, 2300H ; BC  $\leftarrow$  2300H i.e. B  $\leftarrow$  23, C  $\leftarrow$  00

Addr. Mode	Flags Affected	Cycles	T-States
Immediate	None	3	10

## 6) MVI M, 8-bit data

The 8-bit data is immediately moved into the memory location pointed by HL (memory pointer).

**Eg:** MVI M, 23 ; [[HL]]  $\leftarrow$  23H

Addr. Mode	Flags Affected	Cycles	T-States
Immediate	None	3	10

## 7) LDA 16-bit address

The accumulator is loaded with the contents of the memory location having the given address.

**Eg:** LDA 2000H ; A  $\leftarrow$  [2000]

Addr. Mode	Flags Affected	Cycles	T-States
Direct	None	4	13

## 8) STA 16-bit address

The accumulator is stored into the memory location having the given address.

**Eg:** STA 2000H ; [2000]  $\leftarrow$  A

Addr. Mode	Flags Affected	Cycles	T-States
Direct	None	4	13

## 9) LHLD 16-bit address

The HL pair is loaded with the contents of the locations pointed by the given address and address + 1. ☺ In case of doubts, contact Bharat Sir: - 98204 08217.

**Eg:** LHLD 2000 ; HL  $\leftarrow$  [2000] & [2001] i.e. L  $\leftarrow$  [2000], H  $\leftarrow$  [2001]

Addr. Mode	Flags Affected	Cycles	T-States
Direct	None	5	16

## 10) SHLD 16-bit address

The HL pair is stored into the locations pointed by the given address and address + 1.

**Eg:** SHLD 2000 ; [2000] & [2001]  $\leftarrow$  HL i.e. [2000]  $\leftarrow$  L, [2001]  $\leftarrow$  H

Addr. Mode	Flags Affected	Cycles	T-States
Direct	None	5	16

### 11) LDAX rp

The accumulator is loaded with the contents of memory location pointed by value of the given register.

**Eg: LDAX B** ; A  $\leftarrow$  [[BC]] i.e. if [BC] = 2000, A gets the value from location ; 2000 i.e. A  $\leftarrow$  [2000]

Addr. Mode	Flags Affected	Cycles	T-States
Indirect	None	2	7

### 12) STAX rp

The accumulator is stored into the location pointed by value of the given register.

**Eg: STAX B** ; [[BC]]  $\leftarrow$  A i.e. if [BC] = 2000, location 2000 will get the ; value of A i.e. [2000]  $\leftarrow$  A.

Addr. Mode	Flags Affected	Cycles	T-States
Indirect	None	2	7

### 13) PCHL

The Program Counter gets the contents of the HL register pair.

This statement causes a branch in the sequence of the program.

**Eg: PCHL** ; PC  $\leftarrow$  HL

Addr. Mode	Flags Affected	Cycles	T-States
Register	None	1	6

### 14) SPHL

The Stack Pointer gets the contents of the HL register pair.

This statement relocates the stack in the 64 KB memory.

**Eg: SPHL** ; SP  $\leftarrow$  HL

Addr. Mode	Flags Affected	Cycles	T-States
Register	None	1	6

### 15) XCHG

This instruction exchanges the contents of HL pair and DE pair.

**Eg: XCHG** ; HL  $\leftrightarrow$  DE  
#Please refer Bharat Sir's Lecture Notes for this ...

Addr. Mode	Flags Affected	Cycles	T-States
Register	None	1	4



## 16) XTHL

This instruction exchanges the DE pair with the contents of location pointed by the SP and SP+1.

**Eg: XTHL** ; HL  $\leftrightarrow$  [[SP]] and [[SP]+1]  
; i.e. if [SP]=2000 then L  $\leftrightarrow$  [2000] and H  $\leftrightarrow$  [2001]  
#Please refer Bharat Sir's Video for more on this ...

Addr. Mode	Flags Affected	Cycles	T-States
Register	None	5	16

## Bharat Acharya Education

Learn...

8085 | 8086 | 80386 | Pentium |  
8051 | ARM7 | COA

Fees: 1199/-

Duration: 6 months

Activation: Immediate

Certification: Yes

Free: PDFs of theory explanation

Free: VIVA questions and answers

Free: PDF of Multiple Choice Questions

Start Learning... NOW!

## Bharat Acharya Education

Order our Books here...

8086 Microprocessor book

Link: <https://amzn.to/3qHDpJH>

8051 Microcontroller book

Link: <https://amzn.to/3aFQkXc>

Official WhatsApp number:

**+91 9136428051**



## ARITHMETIC GROUP

### Special Note:

In the explanation of every instruction, I have mentioned its machine cycles and T-states. You will understand this part once you watch the two videos of timing diagrams

### 1) ADD R

This instruction adds the contents of register R with the accumulator, stores result in the accumulator.

**Eg:** ADD B ; A  $\leftarrow$  A + B

Addr. Mode	Flags Affected	Cycles	T-States
Register	All	1	4

### 2) ADD M

This instruction adds the contents of the memory location pointed by HL, with the accumulator, and stores the result in the accumulator.

**Eg:** ADD M ; A  $\leftarrow$  A + [[HL]]

Addr. Mode	Flags Affected	Cycles	T-States
Indirect	All	2	7

### 3) ADI 8-bit data

This instruction adds the immediate data with the accumulator, and stores the result in A.

**Eg:** ADI 25 ; A  $\leftarrow$  A + 25

Addr. Mode	Flags Affected	Cycles	T-States
Immediate	All	2	7

### 4) ADC R

This instruction adds the contents of the register R with the accumulator, and also adds the carry flag, and stores the result in the accumulator. It is used **while adding large numbers**.

Refer example from our video at [www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

**Eg:** ADC B ; A  $\leftarrow$  A + B + Cy

Addr. Mode	Flags Affected	Cycles	T-States
Register	All	1	4



## 5) ADC M

This instruction adds the contents of the memory location pointed by HL, with the accumulator, and also adds the carry flag, and stores the result in the accumulator.

**Eg:** ADC M ; A  $\leftarrow$  A + [[HL]] + Cy

Addr. Mode	Flags Affected	Cycles	T-States
Indirect	All	2	7

## 6) ACI 8-bit data

This instruction adds the immediate data with the accumulator, and also adds the carry flag, and stores the result in the accumulator.

**Eg:** ACI 25 ; A  $\leftarrow$  A + 25 +Cy

Addr. Mode	Flags Affected	Cycles	T-States
Immediate	All	2	7

Simillarly subtraction is also done as above.

## 7) SUB R

## 8) SUB M

## 9) SUI 8-bit data

## 10)SBB R

## 11)SBB M

## 12)SBI 8-bit data

### Special Note:

During subtraction if a borrow is taken, then carry flag CY becomes 1.

### Special Note:

SBB is used to subtract two large numbers like 16-bit numbers.

First, we subtract the lower bytes using SUB

Then, we subtract the higher bytes using SBB to include the borrow taken by the lower byte.



### **INR R**

This instruction increments the contents of the specified register.

The incremented value is stored back in the same register.

**Eg:** INR B ; B  $\leftarrow$  B + 1

Addr. Mode	Flags Affected	Cycles	T-States
Register	All except carry	1	4

### **14) INR M**

This instruction increments the contents of memory location pointed by HL pair.

The incremented value is stored back at the same location.

**Eg:** INR M ; M  $\leftarrow$  M + 1 i.e. [[HL]]  $\leftarrow$  [[HL]] +1

Addr. Mode	Flags Affected	Cycles	T-States
Indirect	All except carry	3	10

### **15) INX Rp**

This instruction increments the contents of the specified register **pair**.

The incremented value is stored back in the same register **pair**.

**Eg:** INX BC ; BC  $\leftarrow$  BC + 1 i.e. if [BC]=3000 then [BC] becomes 3001.

Addr. Mode	Flags Affected	Cycles	T-States
Register	<b>NONE</b>	1	6

### **16) DCR R**

This instruction decrements the contents of the specified register.

The decremented value is stored back in the same register.

**Eg:** DCR B ; B  $\leftarrow$  B - 1

Addr. Mode	Flags Affected	Cycles	T-States
Register	All except carry	1	4

### **17) DCR M**

This instruction decrements the contents of memory location pointed by HL pair.

The decremented value is stored back at the same location.

**Eg:** DCR M ; M  $\leftarrow$  M - 1 i.e. [[HL]]  $\leftarrow$  [[HL]] - 1

Addr. Mode	Flags Affected	Cycles	T-States
Indirect	All except carry	3	10

### **18) DCX Rp**

This instruction decrements the contents of the specified register **pair**.

The decremented value is stored back in the same register **pair**.

**Eg:** DCX BC ; BC  $\leftarrow$  BC - 1 i.e. if [BC]=3001 then [BC] becomes 3000.

Addr. Mode	Flags Affected	Cycles	T-States
Register	<b>NONE</b>	1	6



### 19) DAD Rp

This instruction adds the contents of the given register pair with HL pair.

The result is stored in the HL pair.

**Eg:** DAD B ; HL  $\leftarrow$  HL + BC

Addr. Mode	Flags Affected	Cycles	T-States
Register	Only Carry	3	10

### 20) DAA

This instruction is used to get the answer in BCD form.

It adjusts the result of an addition operation to make the addition work like a decimal addition.

It is implied addressing and works strictly on A register.

It checks the nibbles of A as follows

If LN > 9 or AC = 1 then add 06H, If HN > 9 or CY = 1 then add 60H

#For examples, Please refer Bharat Sir's video fro more on this ...

Addr. Mode	Flags Affected	Cycles	T-States
Implied	ALL	1	4

### Bharat Acharya Education

Learn...

8085 | 8086 | 80386 | Pentium |

8051 | ARM7 | COA

Fees: 1199/-

Duration: 6 months

Activation: Immediate

Certification: Yes

Free: PDFs of theory explanation

Free: VIVA questions and answers

Free: PDF of Multiple Choice Questions

Start Learning... NOW!

### Bharat Acharya Education

Order our Books here...

8086 Microprocessor book

Link: <https://amzn.to/3qHDpJH>

8051 Microcontroller book

Link: <https://amzn.to/3aFQkXc>

Official WhatsApp number:

**+91 9136428051**



## LOGIC GROUP

### Special Note:

In the explanation of every instruction, I have mentioned its machine cycles and T-states. You will understand this part once you watch the two videos of timing diagrams

### 1) ANA R

Logically AND the contents of the specified register with accumulator, store result in accumulator.

Eg: **ANA B** ; A  $\leftarrow$  A AND B

Addr. Mode	Flags Affected	Cycles	T-States
Register	ALL	1	4

### 2) ANA M

Logically AND the contents of the memory location pointed by HL pair, with the accumulator.

Eg: **ANA M** ; A  $\leftarrow$  A AND M

Addr. Mode	Flags Affected	Cycles	T-States
Indirect	ALL	2	7

### 3) ANI 8-bit data

Logically AND the immediate 8-bit data, with the accumulator.

Eg: **ANI 25**; A  $\leftarrow$  A AND 25

Addr. Mode	Flags Affected	Cycles	T-States
Immediate	ALL	2	7

### Special Note: Use of AND operation

To “Clear any bit”, we must “AND” that bit with “0” and the remaining bits with “1”.

Eg: ANI F0H will Clear the Lower Nibble of A while the Higher Nibble will remain the same.

Similarly we have the other logic instructions as follows:

- 4) ORA R
- 5) ORA M
- 6) ORI 8-bit data

**Special Note: Use of OR operation**

To “Set any bit”, we must “OR” that bit with “1” and the remaining bits with “0”.  
Eg: ORI 0FH will Set the Lower Nibble of A while the Higher Nibble will remain the same.

- 7) XRA R
- 8) XRA M
- 9) XRI 8-bit data

**Special Note: Use of XOR operation**

To “Complement any bit”, we must “XOR” that bit with “1” and the remaining bits with “0”.  
Eg: XRI 0FH will Complement the Lower Nibble while the Higher Nibble will remain the same.

## 10) CMP R

Compares the contents of register R and accumulator.

Comparision essentially is subtraction. Hence, this instruction performs  $A - R$ .

It is very important to **remember** that the **result** of this comparision is **NOT stored in accumulator**, only the Flags are affected. ☺ In case of doubts, contact Bharat Sir: - 98204 08217.

**Eg: CMP B** ; Compares A and B i.e.  $A - B$  ( and not  $B - A$ )

We decide which one of the two is greater by checking the flags affected as follows:

Conclusion	Zero Flag 'Z'	Carry Flag 'Cy'
$A > B$	0	0
$A = B$	1	0
$A < B$	0	1

Addr. Mode	Flags Affected	Cycles	T-States
Register	ALL	1	4

Similarly we have the other comparision instructions as follows:

- 11) CMP M
- 12) CPI 8-bit data



### 13) STC

Sets the carry flag.

Cy  $\leftarrow$  1.

Addr. Mode	Flags Affected	Cycles	T-States
Implied	Only Carry	1	4

### 14) CMC

Complements the carry flag.

Cy  $\leftarrow$  Cy.

Addr. Mode	Flags Affected	Cycles	T-States
Implied	Only Carry	1	4

### 15) CMA

Complements the accumulator.

A  $\leftarrow$  1's complement of A.

Addr. Mode	Flags Affected	Cycles	T-States
Implied	None	1	4

#### Special Note: Use of CMA operation

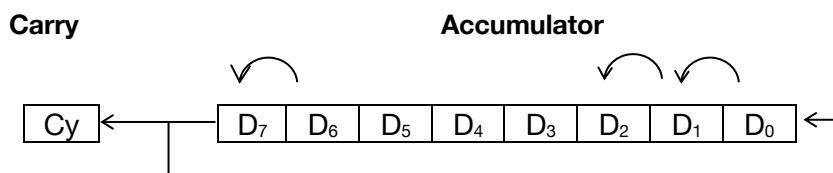
It acts as a NOT gate. AND followed by NOT gives a NAND.

Hence using CMA we can derive NAND, NOR and XNOR operations.

### 16) RLC

The Contents of accumulator are rotated left by 1.

The MSB goes to the Carry AND the LSB.



Addr. Mode	Flags Affected	Cycles	T-States
Implied	Carry	1	4



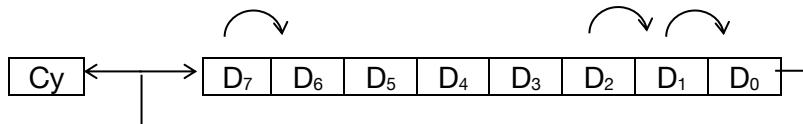
### 17) RRC

The Contents of accumulator are rotated right by 1.

The LSB goes to the Carry AND the MSB.

Carry

Accumulator



Addr. Mode	Flags Affected	Cycles	T-States
Implied	Carry	1	4

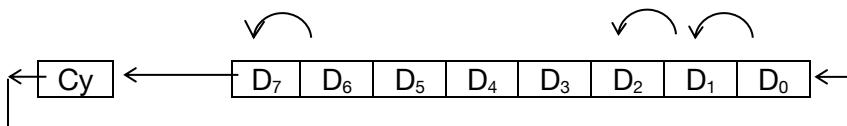
### 18) RAL

The Contents of accumulator are rotated left by 1.

The MSB goes to the Carry and THE CARRY goes to LSB.

Carry

Accumulator



Addr. Mode	Flags Affected	Cycles	T-States
Implied	Carry	1	4

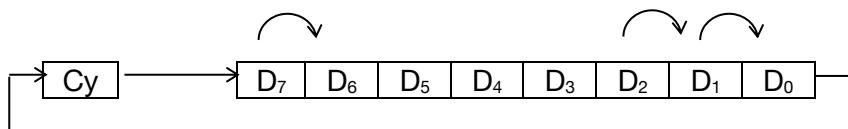
### 19) RAR

The Contents of accumulator are rotated right by 1.

The LSB goes to the Carry and the CARRY goes to the MSB.

Carry

Accumulator



Addr. Mode	Flags Affected	Cycles	T-States
Implied	Carry	1	4



## **Bharat Acharya Education**

Learn...

8085 | 8086 | 80386 | Pentium |

8051 | ARM7 | COA

Fees: 1199/-

Duration: 6 months

Activation: Immediate

Certification: Yes

Free: PDFs of theory explanation

Free: VIVA questions and answers

Free: PDF of Multiple Choice Questions

Start Learning... NOW!

## **Bharat Acharya Education**

Order our Books here...

8086 Microprocessor book

Link: <https://amzn.to/3qHDpJH>

8051 Microcontroller book

Link: <https://amzn.to/3aFQkXc>

Official WhatsApp number:

**+91 9136428051**



## BRANCH CONTROL GROUP

### Special Note:

In the explanation of every instruction, I have mentioned its machine cycles and T-states. You will understand this part once you watch the two videos of timing diagrams

### 1) JMP 16-bit address

#### (Unconditional Jump)

Loads PC with the 16-bit address specified in the instruction.

Eg: JMP 2500 ; PC ← 2500

Addr. Mode	Flags Affected	Cycles	T-States
Immidiate	None	3	10

### Special Note:

Remember the point discussed in the video...

PC will contain address of the next instruction before it gets the branch address from WZ pair.

### 2) JCondition 16-bit address

#### (Conditional Jump)

It is the same as UnConditional JUMP except that the action takes place ONLY if the condition is true.

Eg: JZ 2500 ; PC ← 2500 if Z =1

Addr. Mode	Flags Affected	Cycles	T-States
Immidiate	None	2/3	7/10

### Conditions

Condition	Description	True if:
NZ	No Zero	Z=0
Z	Zero	<b>Z=1</b>
NC	No Carry	C=0
C	Carry	<b>C=1</b>
PO	Parity Odd	P=0
PE	Parity Even	<b>P=1</b>
P	Plus	S=0
M	Minus	<b>S=1</b>



### 3) CALL 16-bit address

#### (Unconditional Call)

Loads PC with the 16-bit address specified in the instruction.  
Before doing so, it also Pushes the Current PC into the Stack.

**Eg: JMP 2500** ; SP  $\leftarrow$  SP - 1  
[SP]  $\leftarrow$  PC<sub>H</sub>  
SP  $\leftarrow$  SP - 1  
[SP]  $\leftarrow$  PC<sub>L</sub>  
PC  $\leftarrow$  2500

#Please refer Bharat Sir's video for clear understanding of this instruction ...

Addr. Mode	Flags Affected	Cycles	T-States
Immidiate	None	5	18

### 4) CCondition 16-bit address

#### (Conditional Call)

It is the same as UnConditional Call except that the action takes place ONLY if the condition is true.

**Eg: CNZ 2500** ; IF Z = 0 then  
SP  $\leftarrow$  SP - 1  
[SP]  $\leftarrow$  PC<sub>H</sub>  
SP  $\leftarrow$  SP - 1  
[SP]  $\leftarrow$  PC<sub>L</sub>  
PC  $\leftarrow$  2500

Addr. Mode	Flags Affected	Cycles	T-States
Immidiate	None	2/5	9/18

### 5) RET

#### (Unconditional Return)

This instruction is written at the end of the sub-routine and enables the control to go back to the main program.  
We enter a subroutine using CALL instruction in which we push the return address into the stack.

In RET instruction we do the opposite i.e. we POP the return address from the stack into PC.

**Eg: RET** ; PC<sub>L</sub>  $\leftarrow$  [SP]  
SP  $\leftarrow$  SP + 1  
PC<sub>H</sub>  $\leftarrow$  [SP]  
SP  $\leftarrow$  SP + 1

Addr. Mode	Flags Affected	Cycles	T-States
Indirect	None	3	10

## 6) RCondition

### (Conditional Return)

It is the same as UnConditional Return except that the action takes place ONLY if the condition is true.

**Eg: RC** ; IF C = 1 then

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

#Please refer Bharat Sir's Lecture Notes for this ...

Addr. Mode	Flags Affected	Cycles	T-States
Indirect	None	1/3	6/12

## 7) RSTn

### (Restart n)

This instruction is very similar to the CALL instruction.

Here the branch address, instead of being specified directly in the instruction, is calculated as  $(n \times 8)$ .

The current PC is Pushed into the stack.

The new value of PC is  $(n \times 8)$ .

The value of n = 0,1,2 ... 7.

These instructions are called as Software Interrupts.

Operationally it is thus simillar to CALL except that it is a **1 byte instruction**.

**Eg: RST1** ; SP ← SP - 1
   
[SP] ← PC<sub>H</sub>
  
SP ← SP - 1
   
[SP] ← PC<sub>L</sub>
  
PC ← 0008 ( $\because 1 \times 8 = 0008$ )

#Please refer Bharat Sir's Lecture Notes for this ...

Addr. Mode	Flags Affected	Cycles	T-States
Indirect	None	3	12

#### Special Note:

PCHL also causes a branch in the program flow (to the location pointed by the HL Pair), but is already included in the data transfer group. It is a very important instruction and during programming, you should remember that it can also cause a branch.



## **Bharat Acharya Education**

Learn...

8085 | 8086 | 80386 | Pentium |

8051 | ARM7 | COA

Fees: 1199/-

Duration: 6 months

Activation: Immediate

Certification: Yes

Free: PDFs of theory explanation

Free: VIVA questions and answers

Free: PDF of Multiple Choice Questions

Start Learning... NOW!

## **Bharat Acharya Education**

Order our Books here...

8086 Microprocessor book

Link: <https://amzn.to/3qHDpJH>

8051 Microcontroller book

Link: <https://amzn.to/3aFQkXc>

Official WhatsApp number:

**+91 9136428051**



## STACK, I/O AND MACHINE CONTROL GROUP

### Special Note:

In the explanation of every instruction, I have mentioned its machine cycles and T-states. You will understand this part once you watch the two videos of timing diagrams

## STACK INSTRUCTIONS

### 1) PUSH Rp

It pushes the given register pair into the stack.

Eg: **PUSH B** ; SP  $\leftarrow$  SP - 1  
           [SP]  $\leftarrow$  B  
           SP  $\leftarrow$  SP - 1  
           [SP]  $\leftarrow$  C

Addr. Mode	Flags Affected	Cycles	T-States
Register	None	3	12

### Special Notes:

All Stack operations (Push and Pop) are compulsorily **16-bit operations**.

We can push register pairs, and not individual 8-bit registers.

Push and Pop only support **Register Addressing Mode**. Hence if we want to push a number like 2000H, we must first move it into a register pair and then push it.

Eg:     **LXI B, 2000H**  
           **Push B**

### 2) POP Rp

It pops the top 2 elements ( $2 \times 8\text{-bit} \therefore 16\text{-bit}$ ) from the stack into the given register pair.

The lower byte comes out first as the higher byte was pushed in first and stack operates in LIFO manner.

Eg:     **POP B** ; C  $\leftarrow$  [SP]  
           SP  $\leftarrow$  SP + 1  
           B  $\leftarrow$  [SP]  
           SP  $\leftarrow$  SP + 1

Addr. Mode	Flags Affected	Cycles	T-States
Register	None	3	10



### 3) PUSH PSW

It pushes the PSW (Program Status Word) into the stack.

The PSW is the combination of the accumulator and the Flag register, accumulator being the higher byte.

∴ PSW → AF

PSW can **ONLY** be used in PUSH and POP instructions.

Eg: **PUSH PSW** ; SP ← SP – 1

[SP] ← A

SP ← SP - 1

[SP] ← F

Addr. Mode	Flags Affected	Cycles	T-States
Register	None	3	12

### 4) POP PSW

It pops the top 2 elements ( $2 \times 8\text{-bit} \therefore 16\text{-bit}$ ) from the stack into the PSW.

Eg: **POP PSW** ; F ← [SP]

SP ← SP + 1

A ← [SP]

SP ← SP + 1

Addr. Mode	Flags Affected	Cycles	T-States
Register	None	3	10

**Please Note:** There are other instructions like XTHL, SPHL, LXI SP, CALL RET etc which directly or indirectly affect the stack and are already included in various groups above.

#### Special Note:

PSW is only created to allow push and pop of Accumulator and Flags.  
Hence PSW can only be used in Push and Pop instructions.



## I/O INSTRUCTIONS

### 5) IN 8-bit I/O Port address

8085 has 256 I/O Ports having 8-bit addresses 00H ... FFH.

This instruction is used to read data from an I/O Port, whose address is given in the instruction.

This data can be read into the Accumulator ONLY.

Eg: IN 80 ; A ← [80]<sub>I/O</sub>

Addr. Mode	Flags Affected	Cycles	T-States
Direct	None	3	10

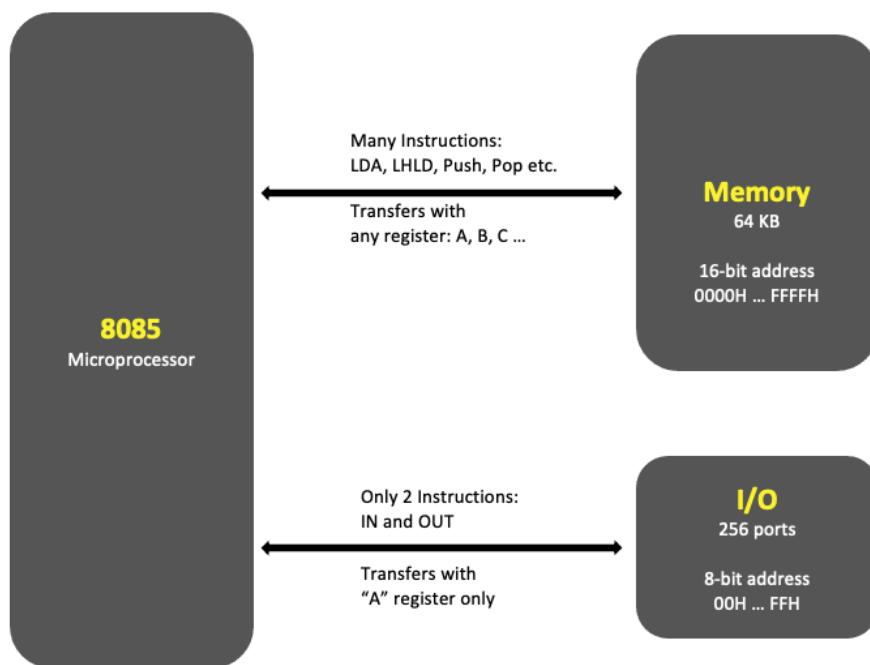
### 6) OUT 8-bit I/O Port address

This instruction is used to send data from the accumulator to an I/O Port, whose address is in the instruction.

This data can be sent from the Accumulator ONLY.

Eg: OUT 80; [80]<sub>I/O</sub> ← A

Addr. Mode	Flags Affected	Cycles	T-States
Direct	None	3	10





## MACHINE CONTROL INSTRUCTIONS

### 7) SIM (Set Interrupt Mask)

This instruction is used to set the interrupt masking pattern for the  $\mu$ P.

The appropriate bit pattern is loaded into the accumulator and then this instruction is executed.

It is basically used to mask/unmask the interrupts except TRAP and INTR.

It can also be used to send a 'bit' out through the serial out pin **SID**.

**Eg:** SIM ;  $\mu$ P accepts the masking pattern through the accumulator.

Addr. Mode	Flags Affected	Cycles	T-States
Implied	None	1	4

### 8) RIM (Read Interrupt Mask)

This instruction is used to read the interrupt masking pattern for the  $\mu$ P.

After executing this instruction, the  $\mu$ P loads the bit pattern into the accumulator.

It can also be used to receive a 'bit' out through the serial in pin **SID**.

**Eg:** RIM ;  $\mu$ P loads the masking pattern into the accumulator.

Addr. Mode	Flags Affected	Cycles	T-States
Implied	None	1	4

**Please Note:** For the bit patterns of SIM and RIM instruction, please refer the chapter on Interrupts.

### 9) EI (Enable Interrupts)

This instruction is used to enable the interrupts in the  $\mu$ P.

This instruction sets the INTE flip-flop (Interrupt Enable Flip-Flop).

This instruction effects all the interrupts except TRAP.

**Eg:** EI ; INTE<sub>F/F</sub> ← 1

Addr. Mode	Flags Affected	Cycles	T-States
Implied	None	1	4

### 10) DI (Disable Interrupts)

This instruction is used to disable the interrupts in the  $\mu$ P.

This instruction resets the INTE flip-flop.

This instruction effects all the interrupts except TRAP.

**Eg:** DI ; INTE<sub>F/F</sub> ← 0

Addr. Mode	Flags Affected	Cycles	T-States
Implied	None	1	4



### 11) NOP (No Operation)

This instruction performs no operation, but consumes time of the  $\mu$ P.  
It is the simplest method of causing a software delay.

Eg: NOP ; -----

Addr. Mode	Flags Affected	Cycles	T-States
Implied	None	1	4

### 12) HLT (Halt)

This instruction signifies the end of the program.  
It causes the  $\mu$ P to stop fetching any further instruction, hence program execution is stopped.  
It makes the Halt Flip Flop inside the  $\mu$ P = 1.  
 $\mu$ P checks the Halt Flip Flop in the beginning (1<sup>st</sup> T-State) of the next Machine Cycle and Stops all operations.

Eg: HLT ; Halt Flip-Flop ← 1

Addr. Mode	Flags Affected	Cycles	T-States
Implied	None	1 + 1T	5

## Bharat Acharya Education

Learn...

8085 | 8086 | 80386 | Pentium |  
8051 | ARM7 | COA

Fees: 1199/-

Duration: 6 months

Activation: Immediate

Certification: Yes

Free: PDFs of theory explanation

Free: VIVA questions and answers

Free: PDF of Multiple-Choice Questions

Start Learning... NOW!

## Bharat Acharya Education

Order our Books here...

8086 Microprocessor book

Link: <https://amzn.to/3qHDpJH>

8051 Microcontroller book

Link: <https://amzn.to/3aFQkXc>

Official WhatsApp number:

**+91 9136428051**



## 8085 PROGRAMMING

### ADD 2 8-BIT NUMBERS

**Q1) Add 2 8-bit numbers stored at 2000H and 2001H.  
Store Sum at 2002H and carry at 2003H.**

**Solution:** Without using M

	<b>MVI C, 00H</b>	; Assume Carry = 0
	<b>LDA 2000H</b>	; A = 1 <sup>st</sup> number
	<b>MOV B, A</b>	; B = 1 <sup>st</sup> number
	<b>LDA 2001H</b>	; A = 2 <sup>nd</sup> number
	<b>ADD B</b>	; A = Sum
	<b>JNC SKIP</b>	; Check for carry
	<b>INR C</b>	; Increment C register if Carry Flag = 1
<b>SKIP:</b>	<b>STA 2002H</b>	; Store Sum at 2002
	<b>MOV A, C</b>	; A = Carry from C Register
	<b>STA 2003H</b>	; Store Carry at 2003H
	<b>HLT</b>	; End of program

**Solution:** Using M

	<b>MVI C, 00H</b>	; Assume Carry = 0
	<b>LXI H, 2000H</b>	; HL = 2000H; M = 1 <sup>st</sup> number
	<b>MOV A, M</b>	; A = 1 <sup>st</sup> number
	<b>INX H</b>	; HL = 2001H; M = 2 <sup>nd</sup> number
	<b>ADD M</b>	; A = Sum
	<b>JNC SKIP</b>	; Check for carry
	<b>INR C</b>	; Increment C register if Carry Flag = 1
<b>SKIP:</b>	<b>INX H</b>	; HL= 2002H
	<b>MOV M, A</b>	; Store Sum at 200H
	<b>INX H</b>	; HL = 2003H
	<b>MOV M, C</b>	; Store Carry at 200H
	<b>HLT</b>	; End of program

#### Special Note:

Please refer our videos for full explanation of the programs and the diagram of the logic.



## BLOCK TRANSFER PROGRAM

**Q2) Transfer a Block of 10 bytes form 2000H to 3000H.**

**Solution:**

	<b>LXI B, 2000H</b>	; Source pointer at 2000H
	<b>LXI D, 3000H</b>	; Destination pointer at 3000H
	<b>MVI L, 0AH</b>	; Count of 10d = 0AH
BACK:	<b>LDAX B</b>	; A gets Source data
	<b>STAX D</b>	; A stored at Destination
	<b>INX B</b>	; Increment Source pointer
	<b>INX D</b>	; Increment Destination pointer
	<b>DCR L</b>	; Decrement Count register
	<b>JNZ BACK</b>	; Loop if count is not zero
	<b>HLT</b>	; End of program

## INVERTED BLOCK TRANSFER PROGRAM

**Q3) Perform inverted block transfer of 10 bytes form 2000H to 3000H.**

**Solution:**

	<b>LXI B, 2000H</b>	; Source pointer at 2000H
	<b>LXI D, 3009H</b>	; Destination pointer at 3009H
	<b>MVI L, 0AH</b>	; Count of 10d = 0AH
BACK:	<b>LDAX B</b>	; A gets Source data
	<b>STAX D</b>	; A stored at Destination
	<b>INX B</b>	; Increment Source pointer
	<b>DCX D</b>	; Decrement Destination pointer
	<b>DCR L</b>	; Decrement Count register
	<b>JNZ BACK</b>	; Loop if count is not zero
	<b>HLT</b>	; End of program



## BLOCK EXCHANGE PROGRAM

**Q4) Exchange two blocks of 10 bytes stored at 2000H and 3000H**

**Solution:**

	<b>LXI B, 2000H</b>	; Pointer to 1 <sup>st</sup> Block at 2000H
	<b>LXI D, 3000H</b>	; Pointer to 2 <sup>nd</sup> Block at 3000H
	<b>MVI L, 0AH</b>	; Count of 10d = 0AH
BACK:	<b>LDAX B</b>	; A = data from Block 1 (Eg: "p" from our video)
	<b>MOV H, A</b>	; H = p
	<b>LDAX D</b>	; A = data from Block 2 (Eg: "q" from our video)
	<b>STAX B</b>	; 1 <sup>st</sup> Block gets q
	<b>MOV A, H</b>	; A = p
	<b>STAX D</b>	; 2 <sup>nd</sup> Block gets p
	<b>INX B</b>	; Increment Source pointer
	<b>INX D</b>	; Increment Destination pointer
	<b>DCR L</b>	; Decrement Count register
	<b>JNZ BACK</b>	; Loop if count is not zero
	<b>HLT</b>	; End of program

### Bharat Acharya Education

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes

Free: PDFs of Theory explanation, VIVA questions and answers, Multiple-Choice Questions

Start Learning... NOW!

### Bharat Acharya Education

Order our Books here...

#### 8086 Microprocessor

Link: <https://amzn.to/3qHDpJH>

#### 8051 Microcontroller

Link: <https://amzn.to/3aFQkXc>

Official WhatsApp number:

**+91 9136428051**



## 8085 PROGRAMMING

### ADD A SERIES OF NUMBERS

**Q1) Add a series of ten 8-bit numbers stored from 2000H.  
Store Result at 200AH and 200BH.**

**Solution:**

	<b>LXI H, 2000H</b>	; HL = 2000H; M = 1 <sup>st</sup> number
	<b>MVI C, 0AH</b>	; Count
	<b>MVI A, 00H</b>	; Sum
	<b>MVI B, 00H</b>	; Carry
BACK:	<b>ADD M</b>	; Add current number
	<b>JNC SKIP</b>	; Check for carry
	<b>INR B</b>	; Increment B register if Carry Flag = 1
SKIP:	<b>INX H</b>	; Point to next number
	<b>DCR C</b>	; Decrement count
	<b>JNZ BACK</b>	; Loop
	<b>MOV M, A</b>	; Store Sum
	<b>INX H</b>	; Point to next location
	<b>MOV M, B</b>	; Store Carry
	<b>HLT</b>	; End of program

**Special Note:**

Please refer our videos for full explanation of the programs and the diagram of the logic.



## FIND HIGHEST NUMBER

Q2) Find the highest in a series of ten 8-bit numbers stored from 2000H.  
Store Result at 200AH.

Solution:

	<b>LXI H, 2000H</b>	; HL = 2000H; M = 1 <sup>st</sup> number
	<b>MVI C, 0AH</b>	; Count
	<b>MVI A, 00H</b>	; Highest number
BACK:	<b>CMP M</b>	; Compare current number
	<b>JNC SKIP</b>	; Check for carry
	<b>MOV A, M</b>	; If carry is 1 then A ← M
SKIP:	<b>INX H</b>	; Point to next number
	<b>DCR C</b>	; Decrement count
	<b>JNZ BACK</b>	; Loop
	<b>MOV M, A</b>	; Store Result
	<b>HLT</b>	; End of program

## FIND EVEN AND ODD NUMBERS

Q3) Find the number of even and odd numbers in a series of ten 8-bit numbers stored from 2000H.  
Store Result at 200AH and 200BH.

Solution:

	<b>LXI H, 2000H</b>	; HL = 2000H; M = 1 <sup>st</sup> number
	<b>MVI C, 0AH</b>	; Count
	<b>MVI D, 00H</b>	; Even count
	<b>MVI B, 00H</b>	; Odd count
BACK:	<b>MOV A, M</b>	; Get number into A
	<b>RRC</b>	; Check LSB
	<b>JC ODD</b>	; If carry, its Odd
	<b>INR D</b>	; Increment Even count
	<b>JMP SKIP</b>	; Move ahead
ODD:	<b>INR B</b>	; Increment Odd count
SKIP:	<b>INX H</b>	; Point to next number
	<b>DCR C</b>	; Decrement count
	<b>JNZ BACK</b>	; Loop
	<b>MOV M, D</b>	; Store Even count
	<b>INX H</b>	; Move to next location
	<b>MOV M, B</b>	; Store Odd count
	<b>HLT</b>	; End of program



## FIND NUMBER OF ONES

**Q4) Find the number of Ones in an 8-bit number stored at 2000H.  
Store Result at 2001H.**

**Solution:**

	<b>LDA 2000H</b>	; A = given number
	<b>MVI B, 00H</b>	; Number of 1s
	<b>MVI C, 08H</b>	; Loop count
BACK:	<b>RRC</b>	; Rotate right
	<b>JNC SKIP</b>	; If no carry, move ahead
	<b>INR B</b>	; Increment 1s count
SKIP:	<b>DCR C</b>	; Decrement loop count
	<b>JNZ BACK</b>	; Loop
	<b>MOV A, B</b>	; A ← 1s count from B
	<b>STA 2001H</b>	; Store result
	<b>HLT</b>	; End of program

### Bharat Acharya Education

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes  
Free: PDFs of Theory explanation, VIVA questions and answers, Multiple-Choice Questions

Start Learning... NOW!

### Bharat Acharya Education

Order our Books here...

#### 8086 Microprocessor

Link: <https://amzn.to/3qHDpJH>

#### 8051 Microcontroller

Link: <https://amzn.to/3aFQkXc>

Official WhatsApp number:

**+91 9136428051**

## 8085 TIMING DIAGRAMS

### **Instruction Cycle:**

This is the time required by the  $\mu$ P to fetch and execute one complete instruction.

The instruction cycle is in two parts:

1. Fetch Cycle
2. Execute Cycle

### **Fetch Cycle:**

This is the time required by the  $\mu$ P to fetch all bytes of an instruction

The length of the fetch cycle is thus determined by the no of bytes in an instruction.

### **Execution Cycle:**

This is the time required by the  $\mu$ P to execute a fetched instruction.

An Instruction cycle consists of various machine cycles.

The most important and also compulsory machine cycle of every instruction is OPCODE FETCH.

There are other machine cycles such as memory read, memory write, I/O read etc.

Machine cycles contain T-states

### **T-State:**

A T-State is one clock cycle of the  $\mu$ P.

$\therefore T = \text{Clock Period} = 1/\text{Clock Frequency}$

Since 8085 (standard version) works at 3 MHz, one T-state =  $1/3$  microseconds = 0.333 microseconds.



## MACHINE CYCLES

It is the time required by the  $\mu$ P doing one operation and accessing one byte from the external module (Memory or I/O). As the data bus of 8085 is 8-bit, one machine cycle will transfer one byte (8-bits). Instructions that require to read or write 16-bit data from memory, need multiple machine cycles.

The Machine cycles of 8085 are given below:

Name	IO/M	RD	WR	S1	S0	INTA	T-States
Opcode Fetch	0	0	1	1	1	1	<b>4/6</b>
Mem Read	0	0	1	1	0	1	3
Mem Write	0	1	0	0	1	1	3
IO Read	1	0	1	1	0	1	3
IO Write	1	1	0	0	1	1	3
Int. Acknowledge	1	1	1	1	1	0	<b>3 or 6</b>
Bus Idle	0	1	1	0	0	1	3

Opcode fetch is compulsory in every instruction and is the first machine cycle of every instruction.

Most instructions are memory based so they may need Memory Reads or Memory Writes after Opcode Fetch.

Only IN and OUT instructions involve IO Read and IO Write respectively.

Only DAD instruction needs Bus Idle

Interrupt Acknowledgment cycle is performed in response to INTR signal. This will be explained much later in the topic of interrupts

### Special Note:

Be smart and focus mainly on Opcode Fetch, Memory Read and Memory Write in the beginning as the other machine cycles are used by only a handful of instructions.



## OPCODE FETCH

- This cycle is used to **fetch** the **Opcode from the memory**.
- This is the **First** Machine Cycle of every instruction.
- It is a **compulsory** Machine Cycle
- It is **generally** of **4 T-States** but **some** instructions require a **6 T-State** Opcode Fetch.

### **During T1**

- A15-A8 contains the higher byte of the address (**PCH**)
- As **ALE** is **high** AD7-AD0 contains the lower byte of the address (**PCL**).
- Since it is an Opcode fetch cycle, **S1** and **S0** go **high**.
- Since it is a memory operation, **IO/M** goes **low**.

### **During T2**

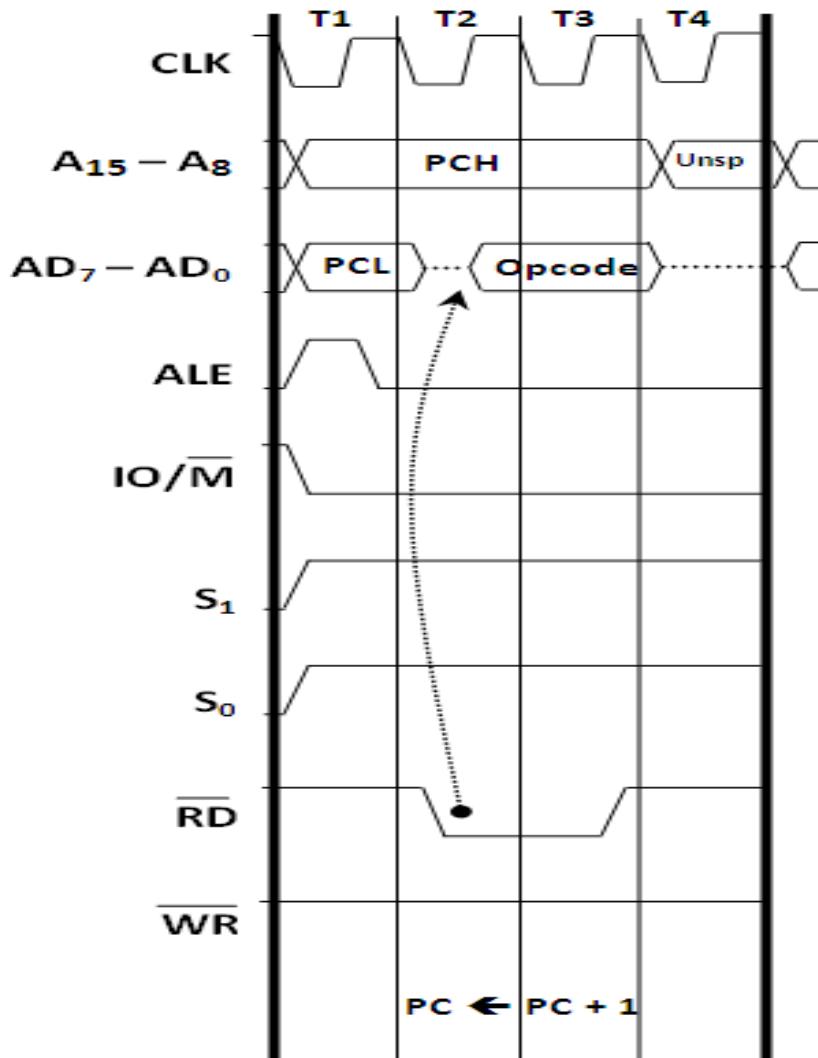
- As **ALE** goes **low** address is removed from AD7-AD0.
- As **RD** goes **low**, **data appears on AD7-AD0**. ☺ In case of doubts, contact Bharat Sir: - 98204 08217.
- As  $\mu$ P is reading the data (Opcode) from memory, there is a propagation delay between sending the address and arrival of data. This is the time required for data (Opcode) to travel from memory to  $\mu$ P. {It is shown by dots between address and data}
- The  $\mu$ P examines the state of the READY pin.  
If READY pin is “high”,  $\mu$ P will continue, but if it is low, then it means the device is not ready, as it is slow. Hence the  $\mu$ P enters wait-state by executing wait cycles and remains in the wait-state until READY goes high.

### **During T3**

- Data remains on AD7-AD0 till RD is low. This is the time given to  $\mu$ P to capture the data (Opcode) from the data bus.

### **During T4**

- T4 state is used by the  $\mu$ P to decode the Opcode.
- This is how an Opcode Fetch cycle is different from a Memory Read cycle.

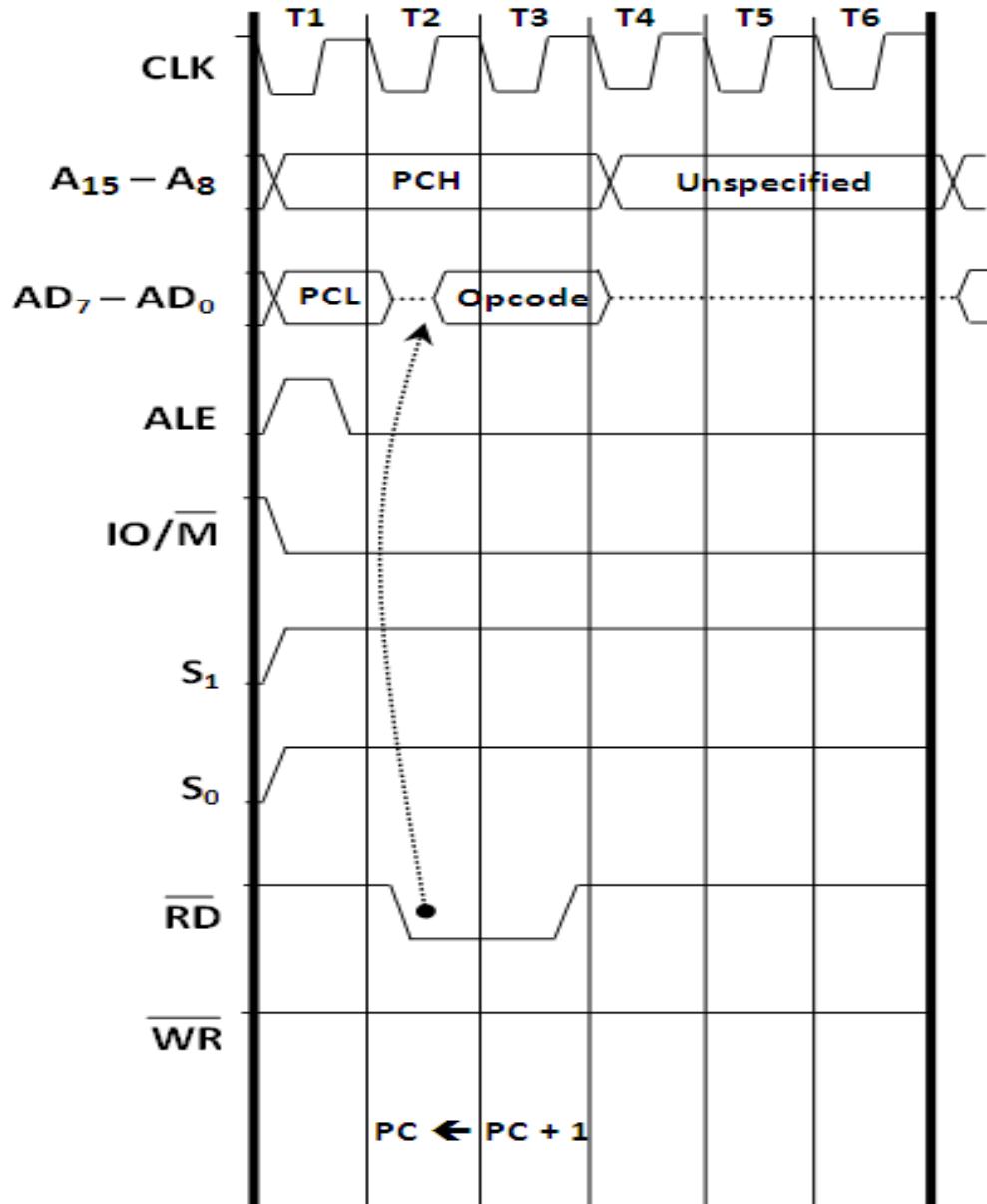


**Special Note:**

Timing diagrams show what is happening in the buses. Decoding in an internal operation and hence is not shown in timing diagrams.



## OPCODE FETCH OF 6 T-STATES



Instructions that use a 6T Opcode Fetch

PCHL  
SPHL  
INX  
DCX  
PUSH  
CALL (all types)  
RC (Conditional RET)  
RSTn

You will of course, understand this later as you learn timing diagrams of instructions.

### Special Note:

T5 and T6 are used for internal operation. Nothing happens on the buses during that time hence nothing is shown in the timing diagram.



## MEMORY READ

- This cycle is used to **fetch one byte from the memory.**
- This cycle can be used to fetch the operand bytes of an instruction or any data from the memory.
- It is a not compulsory Machine Cycle
- It requires **3 T-States.**

### During T1

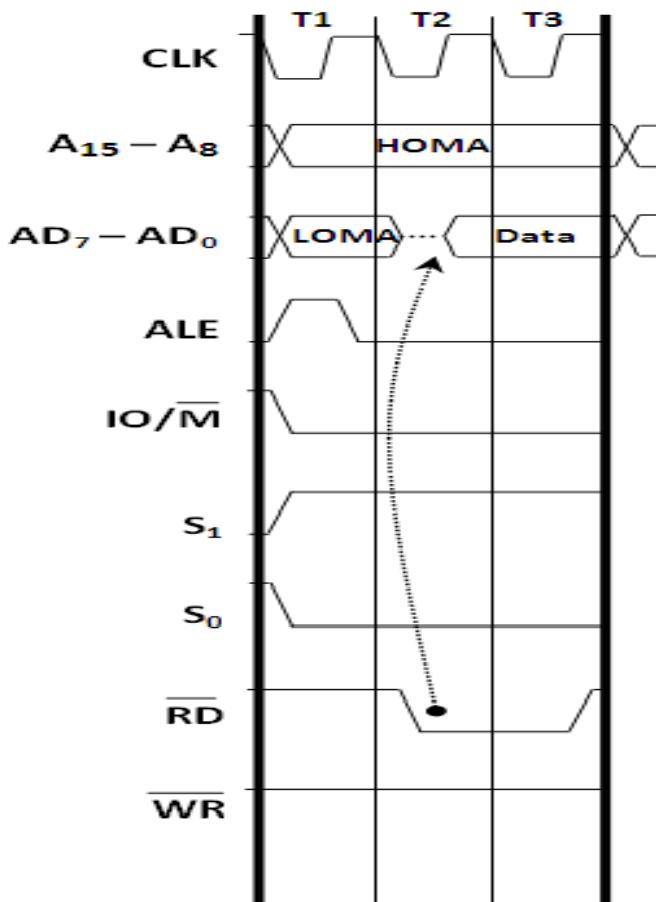
- A15-A8 contains the higher byte of the address (**PCH**)
- As **ALE** is **high**, AD7-AD0 contains the lower byte of the address (**PCL**).
- Since it is a Memory Read cycle, **S1** goes **high**.
- Since it is a memory operation, **IO/M** goes **low**.

### During T2

- **ALE** goes **low**.
- Address is removed from AD7-AD0.
- As **RD** goes **low**, **data** appears **on AD7-AD0**.

### During T3

- Data remains on AD7-AD0 till RD is low.



### Special Note:

Notice the propagation delay, as this is a read operation.



## MEMORY WRITE

- This cycle is used to **send** (write) **one byte** into the **memory**.
- It is a not compulsory Machine Cycle
- It requires **3 T-States**.

### During T1

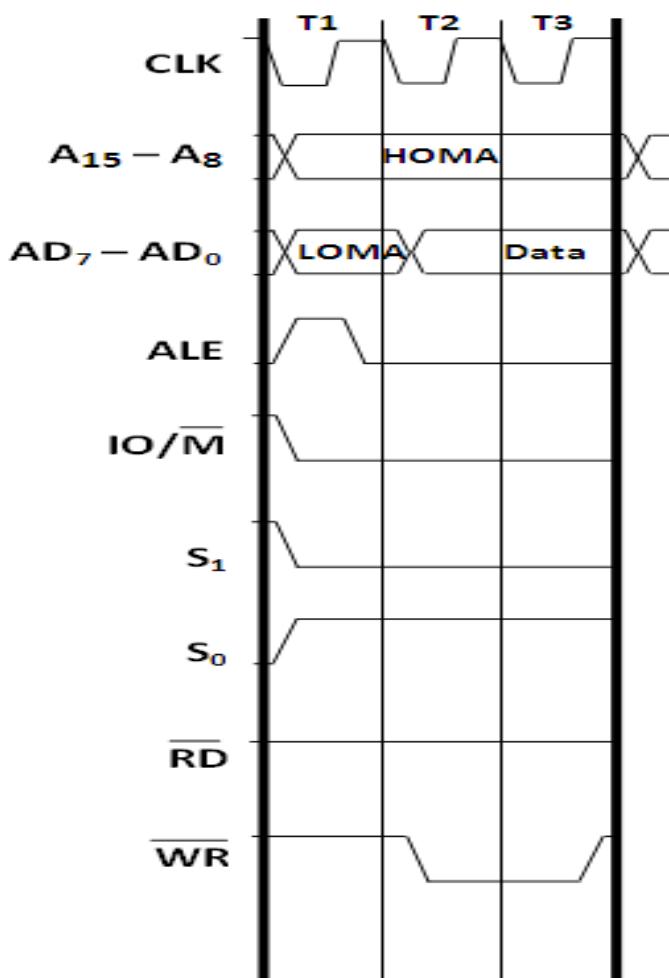
- A15-A8 contains the higher byte of the address (**PCH**)
- As **ALE** is **high**, AD7-AD0 contains the lower byte of the address (**PCL**).
- Since it is a Memory Write cycle, **S0** goes **high**.
- Since it is a memory operation, **IO/M** goes **low**.

### During T2

- **ALE** goes **low**.
- Address is removed from AD7-AD0.
- **Data** appears **on AD7-AD0** and **WR** goes **low**.

### During T3

- Data remains on AD7-AD0 till WR is low.



### Special Note:

Notice, NO propagation delay, as this is a write operation.



## IO READ

- This cycle is used to **fetch one byte from an IO Port.**
- It is a not compulsory Machine Cycle
- It requires **3 T-States.**

### During T1

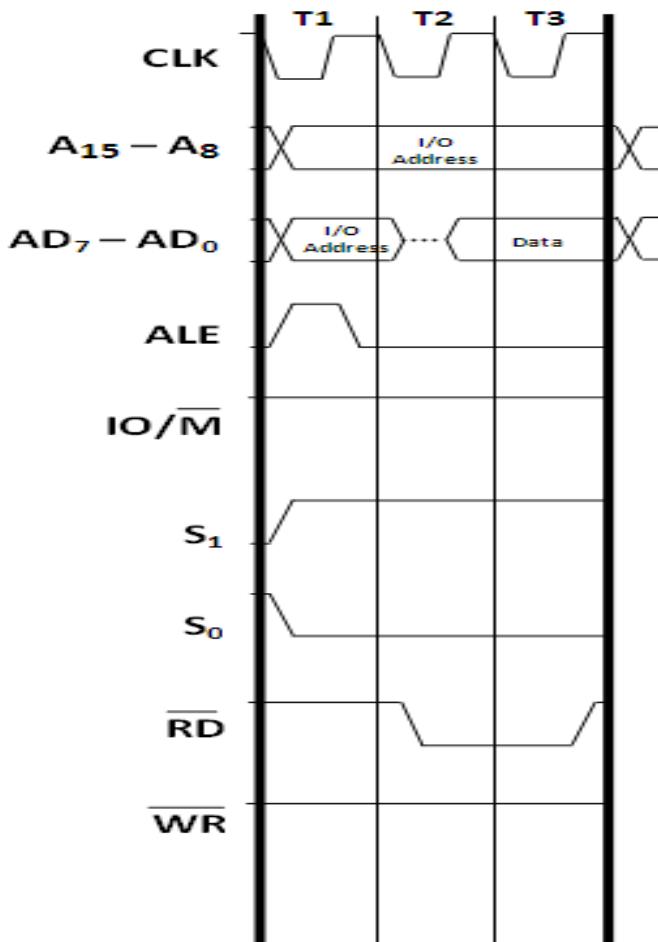
- The **lower 8 bits of the IO Port Address** are duplicated into the higher order address bus **A15-A8**.
- As **ALE is high AD7-AD0 contains the lower byte of the address**
- Since it is an **IO Read** cycle **S1** goes **high**.
- Since it is an **IO operation IO/M** goes **high**.

### During T2

- **ALE goes low.**
- Address is removed from AD7-AD0.
- As **RD goes low, data appears on AD7-AD0.**

### During T3

- Data remains on AD7-AD0 till RD is low.



### Special Note:

Notice the propagation delay, as this is a read operation.



## IO WRITE

- This cycle is used to **send** (write) **one byte** into an **IO Port**.
- It is a not compulsory Machine Cycle
- It requires **3 T-States**.

### During T1

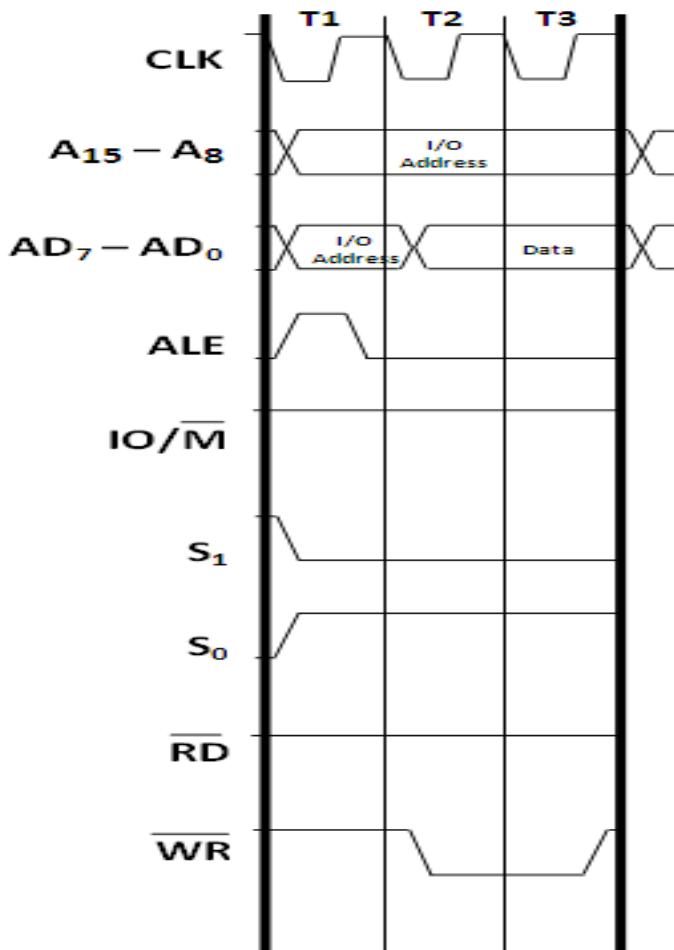
- The **lower 8 bits of the IO Port Address** are duplicated into the higher order address bus **A15-A8**.
- As **ALE** is **high** **AD7-AD0** contains the **lower byte of the address**
- Since it is an **IO Write** cycle, **S0** goes **high**.
- Since it is an **IO** operation, **IO/M** goes **high**.

### During T2

- **ALE** goes **low**.
- Address is removed from **AD7-AD0**.
- **Data** appears **on AD7-AD0** and **WR** goes **low**.

### During T3

- Data remains on **AD7-AD0** till **RD** is low.



### Special Note:

Notice, NO propagation delay, as this is a write operation.



## **Bharat Acharya Education**

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes

Free: PDFs of Theory explanation, VIVA questions and answers, Multiple-Choice Questions

Start Learning... NOW!

## **Bharat Acharya Education**

**Order our Books here...**

### **8086 Microprocessor**

Link: <https://amzn.to/3qHDpJH>

### **8051 Microcontroller**

Link: <https://amzn.to/3aFQkXc>

Official WhatsApp number:

**+91 9136428051**



## 8085 TIMING DIAGRAMS OF INSTRUCTIONS

### Special Note:

Any operation performed INSIDE the microprocessor does not require a machine cycle and hence will not be shown in timing diagrams.

### 1) MVI B, 25H

B  $\leftarrow$  25 H

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Memory Read	PC + 1	25	3
Total			7

### 2) LXI B, 2000H

BC  $\leftarrow$  2000 H

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Memory Read	PC + 1	00	3
Memory Read	PC + 2	20	3
Total			10

### 3) LDA 2000H

A  $\leftarrow$  [2000H]

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Memory Read	PC + 1	00 (Z)	3
Memory Read	PC + 2	20 (W)	3
Memory Read	2000	[2000]	3
Total			13



#### 4) STA 3000H

A → [3000H]

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Memory Read	PC + 1	00 (Z)	3
Memory Read	PC + 2	30 (W)	3
Memory Write	3000	A	3
		Total	13

#### 5) LDAX B

A ← [BC]

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Memory Read	BC	[BC]	3
		Total	7

#### 6) STAX D

A → [DE]

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Memory Write	DE	A	3
		Total	7

#### 7) LHLD 2000H

L ← [2000H], H ← [2001H]

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Memory Read	PC + 1	00 (Z)	3
Memory Read	PC + 2	20 (W)	3
Memory Read	2000	[2000]	3
Memory Read	2001	[2001]	3
		Total	16



### 8) SHLD 5140H

L → [5140H], H → [5141H]

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Memory Read	PC + 1	40 (Z)	3
Memory Read	PC + 2	51 (W)	3
Memory Write	5140	L	3
Memory Write	5141	H	3
Total			<b>16</b>

### 9) MOV B,C

B ← C

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Total			<b>4</b>

### 10) PCHL

**6T**

Opcode Fetch

PC ← HL

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	6
Total			<b>6</b>

### 11) SPHL

**6T**

Opcode Fetch

SP ← HL

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	6
Total			<b>6</b>

### 12) ADD B {all 8 bit arithmetic operations using register addressing mode}

A ← A + B

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Total			<b>4</b>



### 13) INR B

$B \leftarrow B + 1$

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
		Total	4

### 14) INX B

$BC \leftarrow BC + 1$

**6T**

Opcode Fetch

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	6
		Total	6

#### Special Note:

- If you are learning this by piracy, then you are not my student.
- You are simply a thief!  
#PoorUpbringing



## INSTRUCTIONS INVOLVING M – MEMORY POINTER

### Special Note:

- Instructions involving M must be examined more carefully.
- Remember M is not a register. M is a MEMORY LOCATION pointed by HL pair.
- Taking data from M will need a Memory Read cycle.
- Putting data in M will need a Memory Write cycle.

### 15) MVI B, 25H

$B \leftarrow B + 1$

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Memory Read	PC + 1	25	3
Memory Write	HL	25	3
Total			10

### 16) MOV B, M

$B \leftarrow M$

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Memory Read	HL	M	3
Total			7

### 17) MOV M, B

$M \leftarrow B$

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Memory Write	HL	B	3
Total			7



### 18) INR M {Very Important}

$M \leftarrow M + 1$

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Memory Read	HL	M	3
Memory Write	HL	M + 1	3
Total			10

### 19) ADD M

$A \leftarrow A + M$

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Memory Read	HL	M	3
Total			7



## STACK OPERATIONS

### 20) PUSH B

$SP \leftarrow SP - 1$  ... internal operation  
 $[SP] \leftarrow B$  ... memory write  
 $SP \leftarrow SP - 1$  ... internal operation  
 $[SP] \leftarrow C$  ... memory write

**6T**

Opcode Fetch

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	6
Memory Write	SP - 1	B	3
Memory Write	SP - 2	C	3
Total			<b>12</b>

### 21) POP B

$C \leftarrow [SP]$  ... memory read  
 $SP \leftarrow SP + 1$  ... internal operation  
 $B \leftarrow [SP]$  ... memory read  
 $SP \leftarrow SP + 1$  ... internal operation

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Memory Read	SP	[SP]	3
Memory Read	SP +1	[SP + 1]	3
Total			<b>10</b>



## BRANCH INSTRUCTIONS

### 22) JMP 2000H

PC  $\leftarrow$  2000 H

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Memory Read	PC + 1	00 (Z)	3
Memory Read	PC + 2	20 (W)	3
Total			<b>10</b>

### 23) JC 2000H

If CF = 1 then condition is true hence,

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Memory Read	PC + 1	00 (Z)	3
Memory Read	PC + 2	20 (W)	3
Total			<b>10</b>

If CF = 0 then condition is false hence,

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Memory Read (Idle)	---	---	3
Total			<b>7</b>

### 24) Call 2000H

SP $\leftarrow$ SP - 1	...	internal operation
[SP] $\leftarrow$ PCH	...	Memory Write
SP $\leftarrow$ SP - 1	...	internal operation
[SP] $\leftarrow$ PCL	...	Memory Write
PC $\leftarrow$ 2000 H	...	internal operation

**6T**

Opcode Fetch

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	6
Memory Read	PC + 1	00 (Z)	3
Memory Read	PC + 2	20 (W)	3
Memory Write	SP - 1	PCH	3
Memory Write	SP - 2	PCL	3
Total			<b>18</b>



## 25) CC 2000H

If CF=1 then condition is true hence,

SP $\leftarrow$ SP - 1	...	internal operation
[SP] $\leftarrow$ PCH	...	Memory Write
SP $\leftarrow$ SP - 1	...	internal operation
[SP] $\leftarrow$ PCL	...	Memory Write
PC $\leftarrow$ 2000 H	...	internal operation

**6T**

Opcode Fetch

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	<b>6</b>
Memory Read	PC + 1	00 (Z)	<b>3</b>
Memory Read	PC + 2	20 (W)	<b>3</b>
Memory Write	SP - 1	PCH	<b>3</b>
Memory Write	SP - 2	PCL	<b>3</b>
Total			<b>18</b>

If CF = 0 then condition is false hence,

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	<b>6</b>
Memory Read (Idle)	---	---	<b>3</b>
Total			<b>9</b>

## 26) RET

PCL $\leftarrow$ [SP]	...	Memory Read
SP $\leftarrow$ SP + 1	...	internal operation
PCH $\leftarrow$ [SP]	...	Memory Read
SP $\leftarrow$ SP + 1	...	internal operation

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	<b>4</b>
Memory Read	SP	[SP]	<b>3</b>
Memory Read	SP + 1	[SP + 1]	<b>3</b>
Total			<b>12</b>



## 27) RC

If CF = 1 then condition is true hence,

$PCL \leftarrow [SP]$  ... Memory Read  
 $SP \leftarrow SP + 1$  ... internal operation  
 $PCH \leftarrow [SP]$  ... Memory Read  
 $SP \leftarrow SP + 1$  ... internal operation

**6T**  
Opcode Fetch

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	6
Memory Read	SP	[SP]	3
Memory Read	SP + 1	[SP + 1]	3
Total			<b>12</b>

If CF = 0 then condition is false hence,

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	6
Total			<b>9</b>

## 28) RSTn

$SP \leftarrow SP - 1$  ... internal operation  
 $[SP] \leftarrow PCH$  ... Memory Write  
 $SP \leftarrow SP - 1$  ... internal operation  
 $[SP] \leftarrow PCL$  ... Memory Write  
 $PC \leftarrow (n \times 8)$  ... internal operation

**6T**  
Opcode Fetch

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	6
Memory Write	SP - 1	PCH	3
Memory Write	SP - 2	PCL	3
Total			<b>12</b>



## I/O OPERATIONS

29) IN 80H

A  $\leftarrow$  [80]<sub>I/O</sub>

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Memory Read	PC + 1	80	3
I/O Read	80	[80]	3
Total			10

30) OUT 80H

A  $\rightarrow$  [80]<sub>I/O</sub>

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Memory Read	PC + 1	80	3
I/O Write	80	A	3
Total			10



## ADDITIONAL INSTRUCTIONS

### 31) DAD D

$HL \leftarrow HL + DE$

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Bus Idle	---	---	3
Bus Idle	---	---	3
Total			<b>10</b>

### 32) HLT

Halt F/F  $\leftarrow 1$

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	<b>4T +1T</b>
Total			<b>5</b>

### 33) XTHL

$Z \leftarrow [SP]$	...	Memory Read
$W \leftarrow [SP + 1]$	...	Memory Read
$[SP + 1] \leftarrow H$	...	Memory Write
$[SP] \leftarrow L$	...	Memory Write
$HL \leftarrow WZ$	...	internal operation

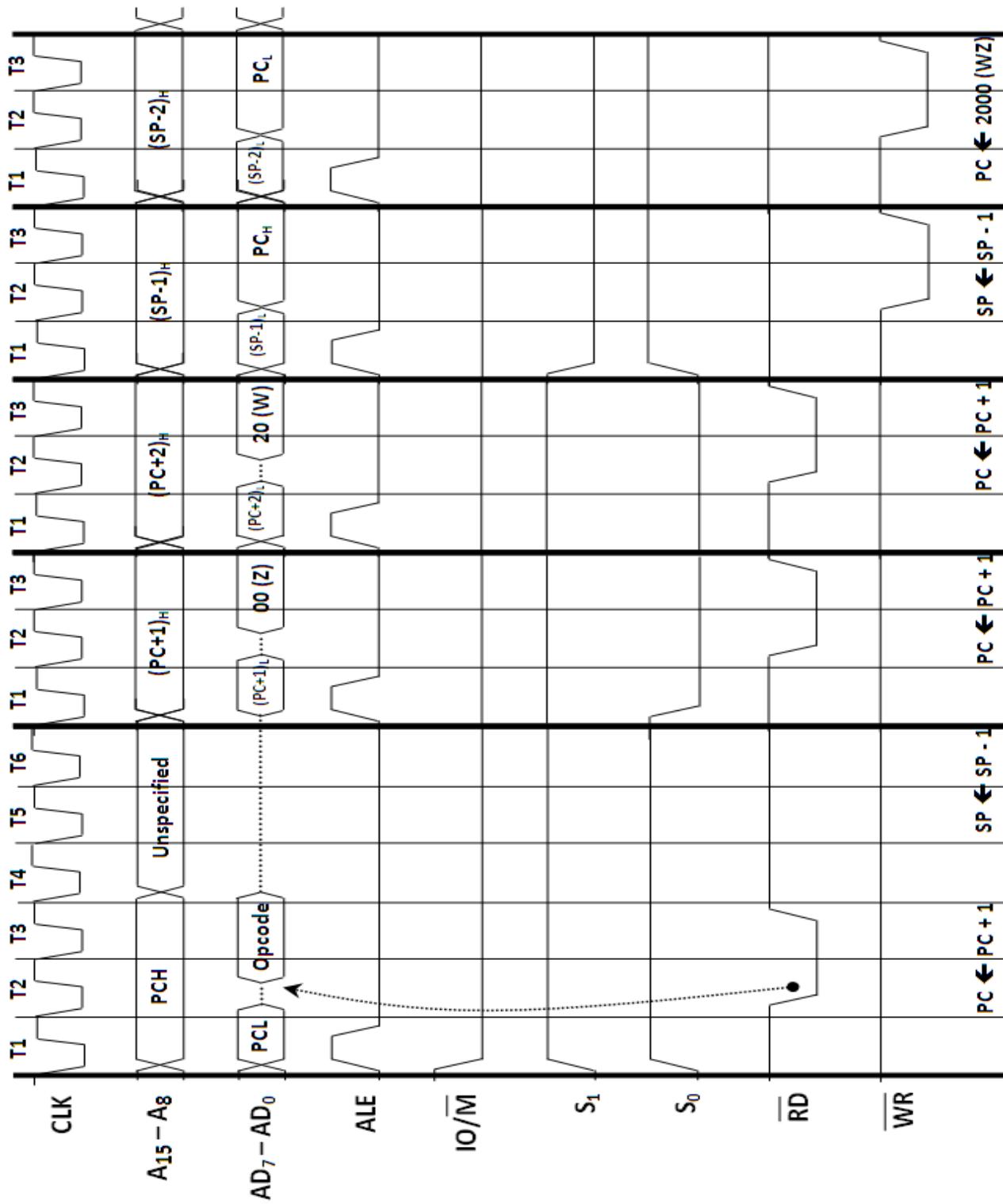
Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Memory Read	SP	[SP]	3
Memory Read	SP + 1	[SP + 1]	3
Memory Write	SP + 1	H	3
Memory Write	SP	L	3
Total			<b>16</b>

### 34) XCHG

$DE \leftrightarrow HL$  ... internal operation

Machine Cycle	Address Bus	Data Bus	T-States
Opcode Fetch	PC	Opcode (Eg: 3E)	4
Total			<b>4</b>

### Timing Diagram for Call 2000 H



## **Bharat Acharya Education**

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes

Free: PDFs of Theory explanation, VIVA questions and answers, Multiple-Choice Questions

Start Learning... NOW!

**[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)**

**Order our Books here...**

### **8086 Microprocessor**

Link: <https://amzn.to/3qHDpJH>

### **8051 Microcontroller**

Link: <https://amzn.to/3aFQkXc>

Official WhatsApp number:

**+91 9136428051**

# 8085 STACKS AND SUBROUTINES

## STACKS

A **stack** is a part of the Memory that **operates in LIFO** (Last In First Out) manner.

In 8085 the **Stack can located anywhere within the 64 KB memory**.

The **top of the Stack** is pointed by the **SP** (Stack Pointer) register.

**P.S.:** The SP contains the **Address** of the top of the Stack and **not the top of the Stack**.

By initializing SP with any suitable value, the programmer can decide where he/she would like to start the stack.

Remember SP gets decremented on every PUSH operation. This means the stack grows upwards. Hence it is advisable to start the stack at a lower location so that it does not overwrite on any other useful information.

### VIVA Question: Why is LXI SP, FFFFH preferred?

LXI SP, FFFFH is preferred due to two reasons,

- It gives the Stack max space to grow.
- It prevents the Stack from overwriting on programs in the memory.

### Instructions related to stacks

- **LXI SP**, 16 bit value Eg LXI SP 4000.
- **SPHL**
- **INX SP**
- **DCX SP**
- **PUSH Rp** Eg PUSH B
- **POP Rp** Eg POP B
- **CALL** 16 bit address Eg CALL 2000
- **RSTn** Eg RST1
- **RET**
- **XTHL**

### Note

In the exam, explain any 2 of the above instructions as an introduction to stacks.  
Preferably PUSH and POP.



## USES of STACKS

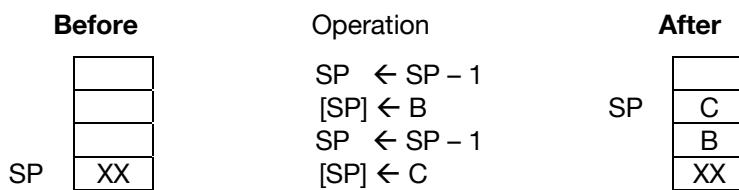
Stacks are used in the following ways:

### 1) To store data during programs

Stack can be used for **storing data by the programmer** as the number of General Purpose registers is very less. The programmer, at any point of time during the program, can store data in the stack and then retrieve it later.

Data can be Pushed into the Stack using Push instruction as below:

Eg : **PUSH B** ; Contents of BC Pair are pushed onto the top of the stack.



Similarly data can also be removed from the stack using the POP operation as:

Eg : **POP B** ; Contents of the top of the stack are popped into BC Pair.

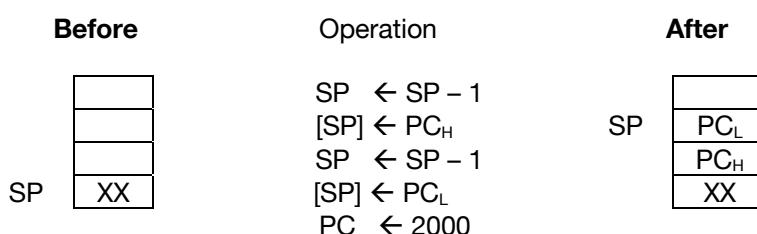
### 2) To Store return address during CALL instructions

The **μP uses** the Stack **for storing** the **return address** during **CALL** instruction.

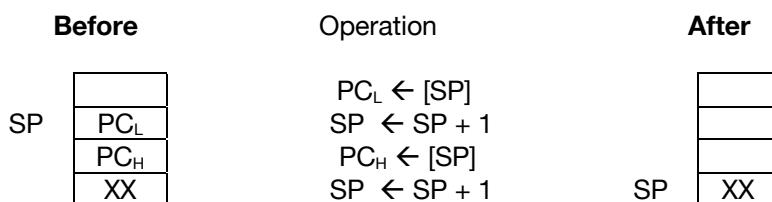
During a CALL instruction, **μP** first pushes the return address (i.e. the current contents of PC) into the stack and then loads the branch address into PC, to branch to the subroutine.

Inside the subroutine, when the **μP** gets a RET instruction, it pops back the return address from the stack, and thus successfully returns to the main program.

Eg: **CALL 2000**



Eg: **RET**





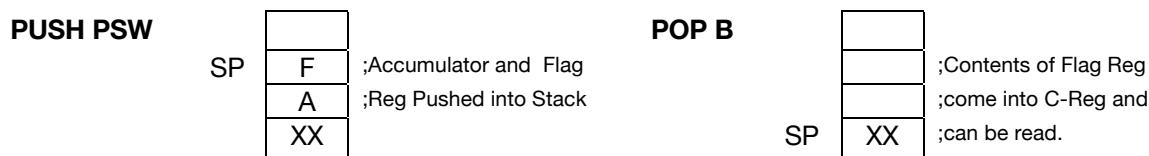
### 3) To Read/Write the contents of the Flag register

The programmer can Read / Write the contents of the Flag register using the Program Status Word (**PSW**). The PSW consists of the Accumulator as the higher byte and the Flag register as the lower byte. The PSW is available only for PUSH and POP instructions.

#### To Read the contents of Flag register

- The programmer pushes PSW into the stack and then pops it into any register pair (BC).
- The contents of the flag register are thus available in the C register (lower register).

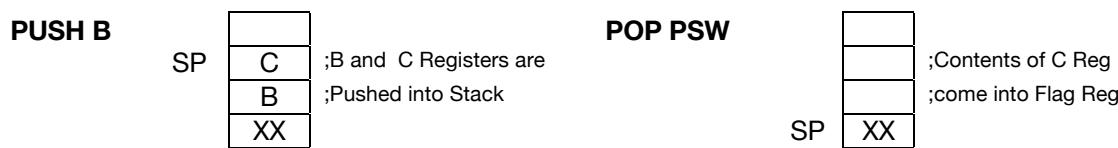
Eg:



#### To Write into the Flag register

- The programmer loads the appropriate byte into the lower register of a register pair (eg: C reg of BC pair).
- Then the register pair is pushed into the stack.
- These contents are then popped into the PSW, and thus the byte that was originally loaded into C register is now **written** into the Flag register.

Eg:



#### Note

This is the **ONLY method** by which the programmer can **write into the Flag register**.

#### 4) To Pass parameters to a Sub-Routine

The programmer can use the Stack to pass parameters to Sub-Routines.

**Before calling** the Sub-Routine the **parameter** is **Pushed** into the Stack and then **inside the Sub-Routine** the **parameter** is **Popped from the Stack**.

Eg:

```
LXI D 1200 ; Parameter 1020 Pushed
PUSH D ; into the Stack
CALL SUB
ADD B
SUB:    POP H ; Return address taken in HL
         POP B; Parameter is accepted into
         .      ; BC pair.
         .
         .
PCHL   ; PC gets the return address
         ; from HL
```

**HLT**

#Please refer Bharat Sir's video for a detailed explanation of this

**Special Note:**

If you are learning this by piracy, then you are not my student. You are simply a thief!  
#PoorUpbringing



## SUBROUTINES

- Subroutines are parts of a program, which can be re-executed by the programmer.
- Subroutines are Called (invoked) using the CALL instruction; control returns to the main program using the RET instruction.
- Subroutines generally perform tasks, which are used regularly by the program such as numerical calculations, Interrupt Service Routines (ISR) etc.

Subroutines are useful in the following ways:

1. Causes **Code Re-Usability** hence reduces the **size** of the Program and also **saves time**.
2. Makes the program easy to **Maintain** as it becomes **Modular**.

### Passing Parameters to Subroutines

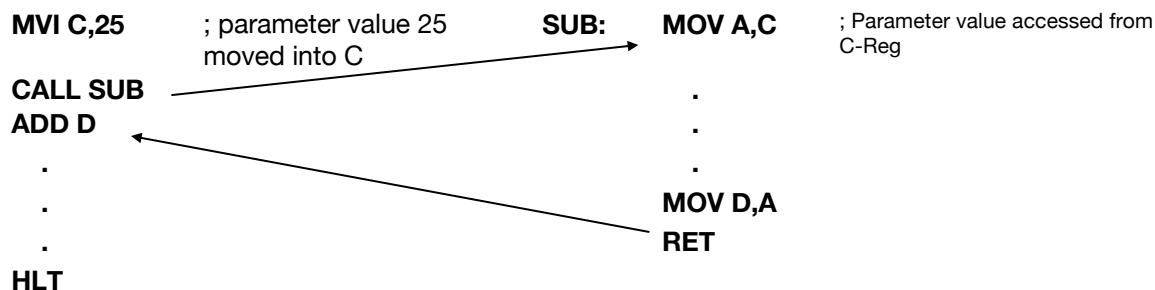
Passing parameters to Subroutines is the procedure of passing some values from the main program to the Subroutines. Passing parameters is very important as it **improves the flexibility of the Subroutine**.

In 8085 there are 4 methods of passing parameters to Subroutines.

#### 1) Using Registers

The parameter value to be passed is moved /loaded into the register before calling the SubRoutine. The SubRoutine accesses the value from the same register.

Eg:



This is the **simplest method** of passing parameters.

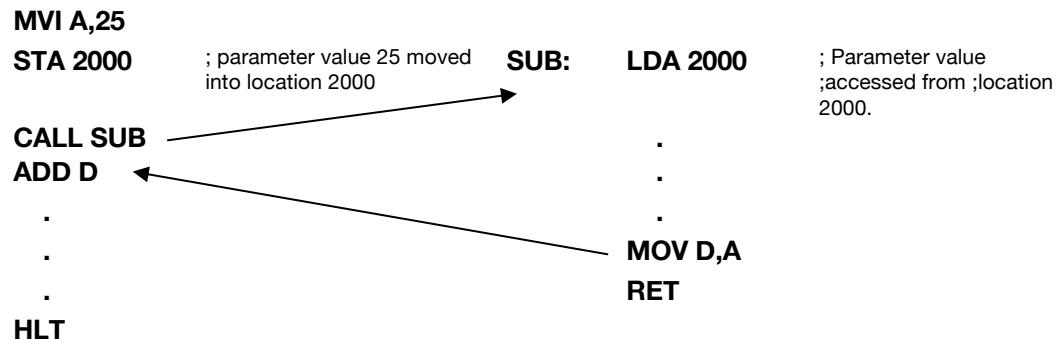
The only main **drawback** is that it **occupies the general-purpose registers** available to the programmer which are already very few. **Also the number** of parameters passed is **restricted by the number of registers available**.



## 2) Using Memory

The parameter value to be passed is stored into the Memory Location before calling the SubRoutine.  
The SubRoutine accesses the value from the same memory location.

Eg:



If **more parameters** are to be passed, then **consecutive memory locations** can be used (which are plenty), thus **eliminating the drawbacks** of the previous method.

The **only drawback** here is that the programmer needs to **remember the memory locations** associated with each subroutine. When the number of subroutines is large, this can be troublesome.

## 3) Using Memory Pointer "M"

The parameter value to be passed is moved into the Memory pointer "**M**" before calling the Subroutine.

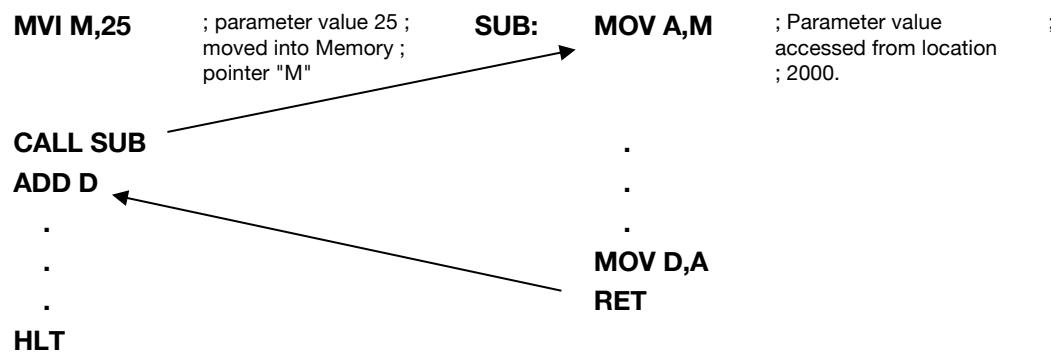
😊 In case of doubts, contact Bharat Sir: Connect with us on our Official WhatsApp number: +919136428051

The HL pair at this point of time may contain any random value.

The SubRoutine accesses the value from M.

Care has to be taken that after the value is put into M the value of **HL pair** should not change until the SubRoutine accesses the value of M.

Eg:





Thus the memory is used but without remembering the memory location.  
i.e. In the above example, we still **do not know** the value of **HL** pair, and thus the **address** of the memory location used (thus avoiding the previous drawback).  
But, when **more** than one values have to be passed, this method becomes **troublesome**.

#### 4) Using Stack \*

The **parameter** value **to be passed** is **Pushed** into the Stack before calling the SubRoutine.  
The SubRoutine **Pops** the **passed parameter** from the Stack.  
When the **μP** calls the SubRoutine it **pushes** the **return address** into the stack.  
This address appears above the passed parameter in the stack.  
Due to the **LIFO** property of the stack we cannot access the passed parameter unless we pop the return address.  
Hence inside the SubRoutine **firstly**, the **return address** is **popped into the HL pair**.  
Then the **passed parameter** is **popped** into the BC pair.  
Now in this case the **RET** instruction **cannot be used** to come back to the main program as the top of the stack no longer contains the return address (as we had taken it out in the HL pair).  
Thus the return address is obtained from the HL pair using the **PCHL** instruction which **causes** the control to **return to the main program**.

Eg:

<b>LXI D 1200</b> ; Parameter 1020 Pushed	→	<b>SUB:POP H;</b>	Return address taken in HL
<b>PUSH D</b> ; into the Stack		<b>POP B;</b>	Parameter is accepted into BC pair.
<b>CALL SUB</b>	→	<b>PCHL ;</b>	PC gets the return address from HL
<b>ADD B</b>			
.			

**HLT;**

#Please refer Bharat Sir's video for a detailed explanation of this

#### Special Note:

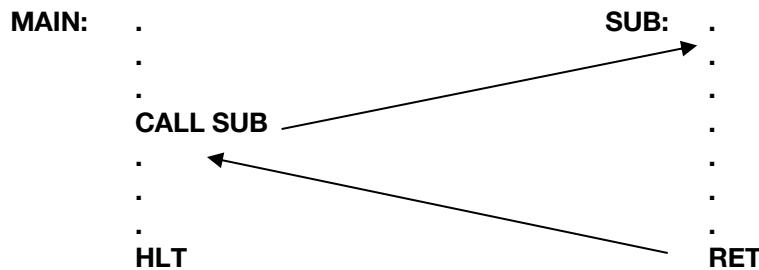
If you are learning this by piracy, then you are not my student. You are simply a thief!  
#PoorUpbringing



## TYPES OF SUB-ROUTINES

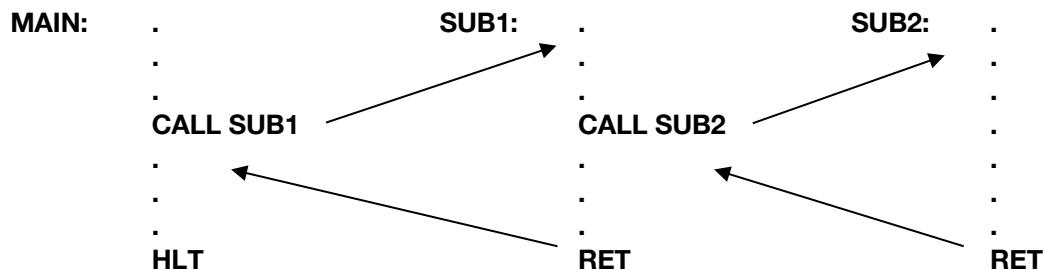
### 1. Simple Subroutines

When a SubRoutine does not call another SubRoutine it is called a Simple SubRoutine.



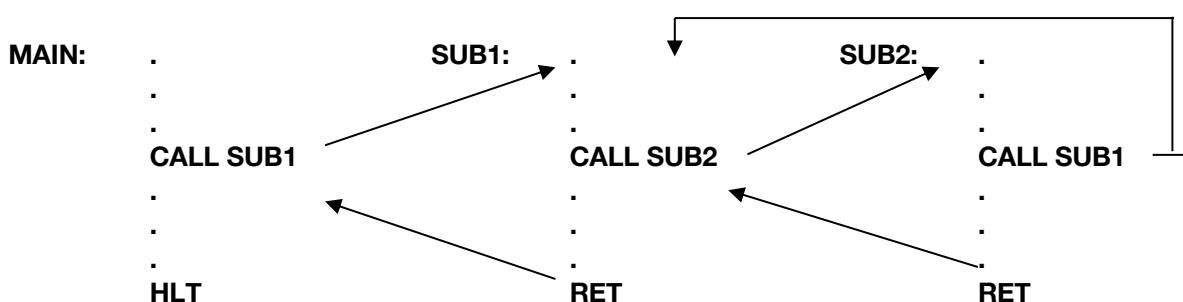
### 2. Nested Subroutines

When a SubRoutine calls another SubRoutine it is called a Nested SubRoutine.



### 3. Re-entrant Subroutines

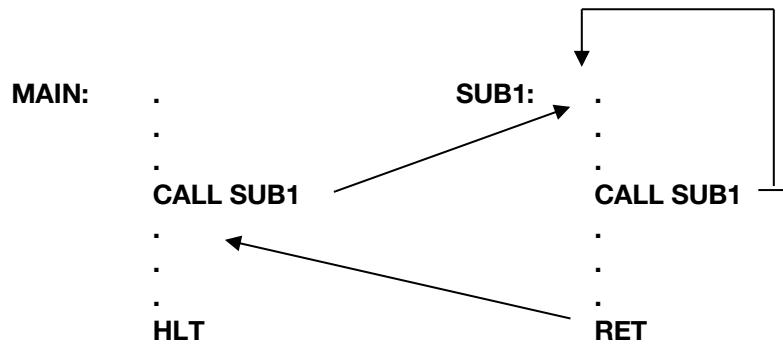
When a SubRoutine is re-entered by another SubRoutine it is called a Re-entrant SubRoutine.





#### 4. Recursive Subroutines

When a SubRoutine calls itself, it is called a Recursive SubRoutine.



#### Bharat Acharya Education

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes

Free: PDFs of Theory explanation, VIVA questions and answers, Multiple-Choice Questions

Start Learning... NOW!

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

Order our Books here...

#### 8086 Microprocessor

Link: <https://amzn.to/3qHDpJH>

#### 8051 Microcontroller

Link: <https://amzn.to/3aFQkXc>

Official WhatsApp number:

**+91 9136428051**

# 8085 DELAYS AND DELAY ROUTINES

## DELAYS

A delay is a time gap between two events. Delays are very useful in computer programs. Something as simple as a traffic light needs delays after every state.

Delays in a computer system can be generated in two ways:

- 1) Hardware delays
- 2) Software delays

	HARDWARE DELAYS	SOFTWARE DELAYS
1	Caused by external hardware (Timer IC like 8254)	Caused by a software delay routine (program)
2	$\mu$ P is not involved in causing the delay and hence is free for other applications.	$\mu$ P is busy as it is executing the delay routine so cannot be used otherwise.
3	Multiple delay routines are possible	Multiple delay routines are not possible
4	Flexibility is low as h/w is involved.	Flexibility is high as delay caused by s/w.
5	External h/w required so circuit becomes more complex	External h/w not required so circuit is simple and less expensive
6	Circuit is More Expensive	Circuit is Less Expensive.

### Note

In this section we will focus on Software delay routines. After a few lectures, we will learn Timer chips like 8253/54. There we will focus on Hardware delays.

## SOFTWARE DELAY ROUTINES

Software delays are produced in programs by one of the various software "**Delay Routines**".

As the amount of delay required varies from application-to-application different types of delay routines are present, as follows:

- 1) Using NOP Instruction**
- 2) Using One 8-bit register**
- 3) Using One 16-bit register**

### Note

The above three methods are the most standard ways of producing delays.  
We can also produce delays using Nested loops or any other user defined method.  
Though not advisable to be used in college exams, they too are interesting to learn.

### Nested delay routines: (Not standard)

- 4) Using Two 8-bit registers**
- 5) Using One 8-bit register and One 16-bit register**
- 6) Using Two 16-bit registers**



## 1) Using NOP Instruction

Eg: NOP;

1-byte instruction

Opcode fetch --- 4 T-states.

$$T_D = 4T.$$

$$T = 1/(\text{Clk freq})$$

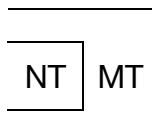
∴ assuming 8085 working at 3 MHz,  $T = 1/(3 \text{ MHz}) = 0.333 \mu \text{ sec.}$

$$\therefore T_D = 4 \times 0.333 = 1.332 \mu \text{ sec.}$$

$$\therefore \mathbf{T_D = 1.332 \mu sec.}$$

This is the maximum delay that can be achieved by writing a NOP instruction.

## 2) Using One 8-bit register

Delay:	<b>MVI B</b> , 8-bit count	7T	
Loop:	<b>DCR B</b>	4T	
	<b>JNZ Loop</b>	10/7T	
	<b>RET</b>	10T	

$$T_D = MT + [(Count)_d \times NT] - 3T$$

**NT** = No of T-states inside the loop {here  $NT = 10T + 4T = 14T$ }

**MT** = No of T-states outside the loop {here  $MT = 7T + 10T = 17T$ }

$\text{Count}_{\max} = 255$  {8-bit count in decimal}

$$\therefore T_{D \max} = 17T + [255 \times 14T] - 3T$$

$$\therefore T_{D \max} = 3584T.$$

Assuming 8085 working at 3 MHz i.e.  $T = 333 \text{ n sec.}$

$$\therefore \mathbf{T_{D \max} = 1.18 \text{ m sec.}}$$

### Very Important:

8-bit delays are extremely useful for the further topics where we need delays to generate square waves, perform serial communications etc.

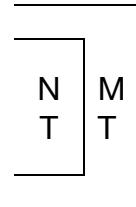
### Special Note:

If you are learning this by piracy, then you are not my student. You are simply a thief!  
#PoorUpbringing



### 3) Using One 16-bit register

Delay:	LXI B, 16-bit count	10T
Loop:	DCX B	6T
	MOV A, B	4T
	ORA C	4T
	JNZ Loop	10/7T
	RET	10T



$$T_D = MT + [(Count)_d \times NT] - 3T$$

$$\text{Here } NT = 6T + 4T + 4T + 10T = 24T$$

$$MT = 10T + 10T = 20T$$

Count<sub>max</sub> = 65535 {16-bit count in decimal}

$$\therefore T_{D\ max} = 20T + [65535 \times 24T] - 3T$$

$$\therefore T_{D\ max} = 1572057T.$$

Assuming 8085 working at 3 MHz i.e. T = 333 n sec.

$$\therefore T_{D\ max} = 0.525 \text{ sec.}$$

#### Note

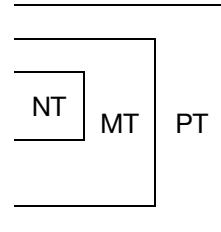
In the above method, you have learnt how to decrement a 16-bit register paid and check for zero. Remember this trick for other programs as well – Bharat Acharya.



Non-standard but interesting delay methods to learn just for knowledge...

#### 4) Using Two 8-bit registers

Delay:	<b>MVI C</b> , Count2	7T
Loop2:	<b>MVI B</b> , Count1	7T
Loop1:	<b>DCR B</b>	4T
	<b>JNZ Loop1</b>	10/7T
	<b>DCR C</b>	4T
	<b>JNZ Loop2</b>	10/7T
	<b>RET</b>	10T



$$T_D = PT + [(Count2)_d \times T_{loop1}] - 3T$$

$$T_{loop1} = MT + [(Count1)_d \times NT] - 3T$$

**NT** = No of T-states inside loop1 {here  $NT = 4T + 10T = 14T$ }.

**MT** = No of T-states outside loop1 but inside loop2 { $MT = 7T + 4T + 10T = 17T$ }

**PT** = No of T-states outside loop2 {here  $PT = 10T + 7T = 17T$ }.

$Count1_{max} = 255$  {8-bit count in decimal}

$Count2_{max} = 255$  {8-bit count in decimal}

$$\therefore T_D \text{ max} = 914954T.$$

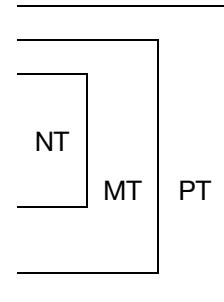
Assuming 8085 working at 3 MHz i.e.  $T = 333$  n sec.

$$\therefore T_D \text{ max} = 0.304 \text{ sec.}$$



## 5) Using One 8-bit register and One 16-bit register

Delay:	<b>MVI D, Count2</b>	7T
Loop2:	<b>LXI B, Count1</b>	10T
Loop1:	<b>DCX B</b>	6T
	<b>MOV A, C</b>	4T
	<b>ORA B</b>	4T
	<b>JNZ Loop1</b>	10/7T
	<b>DCR D</b>	4T
	<b>JNZ Loop2</b>	10/7T
	<b>RET</b>	10T



$$T_D = PT + [(Count2)_d \times T_{loop1}] - 3T$$

$$T_{loop1} = MT + [(Count1)_d \times NT] - 3T$$

Here **NT** = 6T + 4T + 4T + 10T = **24T**.

$$MT = MT = 10T + 4T + 10T = **24T**.$$

$$PT = 7T + 10T = **17T**.$$

$Count1_{max} = 65535$  {16-bit count in decimal}

$Count2_{max} = 255$  {8-bit count in decimal}

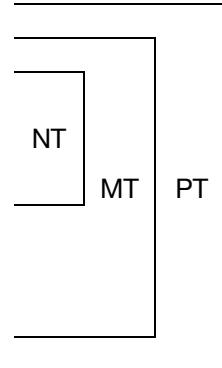
Assuming 8085 working at 3 MHz i.e.  $T = 333$  n sec.

$$\therefore T_D \text{ max} = **133.69** \text{ sec.}$$



## 6) Using Two 16-bit registers

Delay:	<b>LXI B, Count2</b>	10T
Loop2:	<b>LXI D, Count1</b>	10T
Loop1:	<b>DCX D</b>	6T
	<b>MOV A, E</b>	4T
	<b>ORA D</b>	4T
	<b>JNZ Loop1</b>	10/7T
	<b>DCX B</b>	6T
	<b>MOV A, C</b>	4T
	<b>ORA B</b>	4T
	<b>JNZ Loop2</b>	10/7T
	<b>RET</b>	10T



$$T_D = PT + [(Count2)_d \times T_{loop1}] - 3T$$

$$T_{loop1} = MT + [(Count1)_d \times NT] - 3T$$

Here **NT** = 6T + 4T + 4T + 10T = **24T**.

$$MT = MT = 10T + 6T + 4T + 4T + 10T = **34T**.$$

$$PT = 10T + 10T = **20T**.$$

$Count1_{max} = 65535$  {16-bit count in decimal}

$Count2_{max} = 65535$  {16-bit count in decimal}

Assuming 8085 working at 3 MHz i.e.  $T = 333$  n sec.

$$\therefore T_D_{max} = **9 hours, 32 min, 24 sec.**$$



Delay Method	Max Delay
NOP	1.332 $\mu$ sec
One 8-bit register	1.18 m sec
One 16-bit register	.525 sec
Two 8-bit registers	.304 sec
One 8-bit / one 16-bit reg	133.69 sec
Two 16-bit register	9 hrs, 32 min, 24 sec

### Note

All these calculations are w.r.t. 8085 operating at 3 MHz.

In case in the exam, the frequency is different then calculate  $1T = 1/(\text{Clk. freq.})$ , and then calculate the appropriate delay.

☺ In case of doubts, contact us on our Official WhatsApp number: +919136428051

## Bharat Acharya Education

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes

Free: PDFs of Theory explanation, VIVA questions and answers, Multiple-Choice Questions

Start Learning... NOW!

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

Order our Books here...

### 8086 Microprocessor

Link: <https://amzn.to/3qHDpJH>

### 8051 Microcontroller

Link: <https://amzn.to/3aFQkXc>

Official WhatsApp number:

**+91 9136428051**



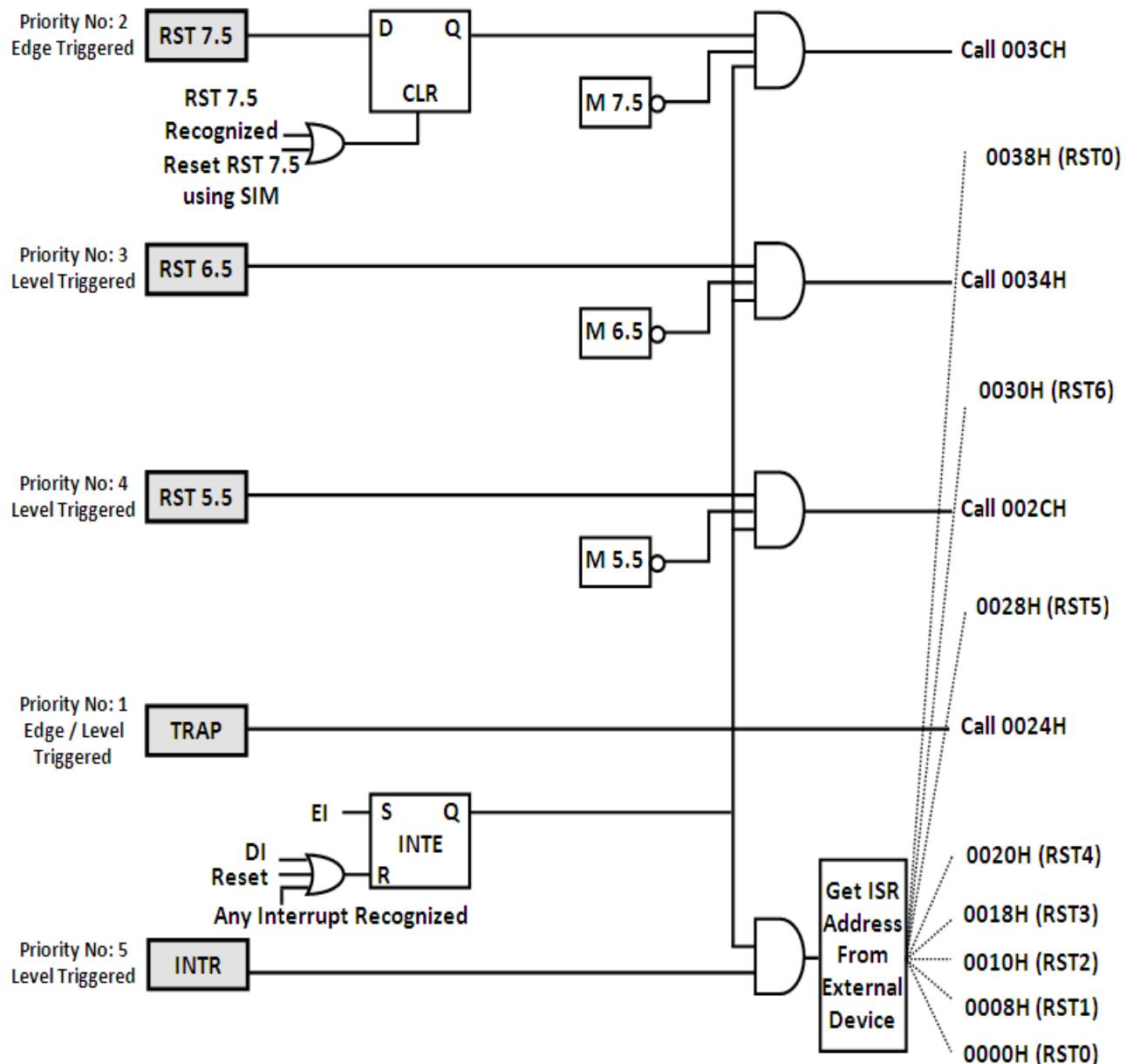
## 8085 INTERRUPTS

- An interrupt is a **special condition** that makes the  $\mu$ P execute an ISR.
- **$\mu$ P services** the interrupt **by executing** a subroutine called as the **Interrupt Service Routine**.
- The  **$\mu$ P checks** for interrupts **during every instruction**.
- When an **interrupt occurs**, the  **$\mu$ P first finishes the current instruction**.
- It then **Pushes** the address of the next instruction (**contents of PC**) on the **STACK**.
- It **resets** the **INTE flip-flop** so that **no more interrupts are recognized**.
- Thereafter the program control transfers to the **address of the Interrupt Service Routine (ISR)** and the  **$\mu$ P thus executes the ISR**.

	SOFTWARE INTERRUPTS	HARDWARE INTERRUPTS
1	They are caused by writing an instruction.	They occur as signals on external pins.
2	8085 has 8 software interrupt instructions called RST N, where N can be any value from 0...7. Hence we have RST 0 ... RST 7.	8085 has 5 hardware interrupt pins: TRAP RST 7.5 RST 6.5 RST 5.5 INTR
3	Software interrupts cannot be masked or disabled.	All Hardware interrupts can be disabled except TRAP.
4	All Software interrupts have the same priority.	Hardware interrupts have the following priority order: TRAP ... 1 <sup>st</sup> (Highest) RST 7.5 ... 2 <sup>nd</sup> RST 6.5 ... 3 <sup>rd</sup> RST 5.5 ... 4 <sup>th</sup> INTR ... 5 <sup>th</sup> (Lowest)
5	All Software interrupts are Vectored	All Hardware interrupts are Vectored except INTR.
6	The Vector addresses are as follows:  RST 0 ... 0000 H RST 1 ... 0008 H RST 2 ... 0010 H RST 3 ... 0018 H RST 4 ... 0020 H RST 5 ... 0028 H RST 6 ... 0030 H RST 7 ... 0038 H	The Vector addresses are as follows:  TRAP ... 0024 H RST 7.5 ... 003C H RST 6.5 ... 0034 H RST 5.5 ... 002C H INTR ... obtain ISR address from device



## INTERRUPT STRUCTURE OF 8085





### Software Interrupts:

Interrupts that are **initiated by writing an Instruction** (software) are called as software interrupts.

8085 has 8 software interrupts:

**RST<sub>n</sub>** where n = 0,1,2, ... 7 i.e. RST0, RST1 ... upto RST7.

- This instruction causes a **service routine** to be Called from the **address (n\*8)**.
- Hence, if RST1 occurs then the program control moves to location 0008 ( $1*8 = 0008$ ).

The respective addresses for software interrupts are given below.

S/W Interrupt	ISR Address
RST0	0000H
RST1	0008H
RST2	0010H
RST3	0018H
RST4	0020H
RST5	0028H
RST6	0030H
RST7	0038H

### Hardware Interrupts:

Interrupts that are initiated through a hardware pin are called as hardware interrupts.

8085 supports the following hardware interrupts:

- TRAP
- RST 7.5
- RST 6.5
- RST 5.5
- INTR

### Vectorized Interrupts:

- Interrupts that **have a FIXED Address** for their ISR are called as Vectored Interrupts.
- **Eg: TRAP** is a vectored interrupt. Its vector address is 0024H.

### Non-Vectored Interrupts:

- Interrupts that **have a Variable Address** for their ISR are called as Non-Vectored Interrupts.
- **Eg: INTR** is a Non-Vectored Interrupt.

### Methods of preventing an interrupt from occurring.

- **MASK Individual Bits** through **SIM Instruction**
- **Disable all** Interrupts through **DI Instruction**

### MASKING:

- We can prevent an interrupt from occurring by MASKING its individual bit through **SIM Instruction**.
- If an interrupt is masked it will not be serviced.
- One of the **main advantages** of masking as opposed to disabling interrupts is that by masking we can **selectively disable a particular interrupt** while keeping other interrupts active, whereas through DI instruction all interrupts are disabled.
- **ONLY RST 7.5, RST 6.5 and RST 5.5** can be masked by this method.



### DISABLING INTERRUPTS:

- Interrupts can be disabled through the **DI** Instruction.
- This instruction resets the INTE Flip Flop and hence none of the interrupts can occur (**Except TRAP**).
- I.e. INTE F/F  $\leftarrow 0$ .
- Once disabled, these interrupts can be **re-enabled** through **EI** instruction, which sets the INTE Flip Flop.
- I.e. INTE F/F  $\leftarrow 1$ .

## Hardware Interrupts (In detail)

### TRAP

- TRAP has the **highest priority**.
- It is **Edge as well as Level triggered** hence the signal must go High and also Remain high for some time for it to be recognized. This prevents any noise signal from being accepted.
- It is a Non-Maskable Interrupt i.e. it can **neither be masked nor be disabled**.
- It is a vectored interrupt and has a **vector address of 0024H**.

### RST 7.5

- RST 7.5 has the **priority lower than TRAP**.
- It is **Edge triggered**.
- It is a **Maskable Interrupt** i.e. it can be masked through the **SIM Instruction**.
- It can also be disabled though the **DI Instruction**.
- It is a vectored interrupt and has a **vector address of 003CH**.
- RST 7.5 can also be **reset** through the **R 7.5** bit in the **SIM Instruction** irrespective of whether it is Masked or not.

### RST 6.5

- RST 6.5 has the **priority lower than RST 7.5**.
- It is **Level triggered**.
- It is a **Maskable Interrupt** i.e. it can be masked through the **SIM Instruction**.
- It can also be disabled though the **DI Instruction**.
- It is a vectored interrupt and has a **vector address of 0034H**.

### RST 5.5

- RST 5.5 has the **priority lower than RST 6.5**.
- It is **Level triggered**.
- It is a **Maskable Interrupt** i.e. it can be masked through the **SIM Instruction**.
- It can also be disabled though the **DI Instruction**.
- It is a vectored interrupt and has a **vector address of 002CH**.

### INTR

- INTR has the **priority lower than RST 5.5**.
- It is **Level triggered**.
- It can only be disabled though the **DI Instruction**.
- **It cannot be masked through the SIM Instruction**.
- It is a **Non-Vectored** interrupt.



### Response to INTR:

- When INTR occurs the  $\mu$ P, in response, issues the **first INTA** cycle.
- The **External Hardware sends an opcode**, which can be of **RSTn** Instruction or of **CALL** instruction.
  - a) If opcode of **RSTn** is sent by the external hardware
    - The  $\mu$ P calculates the address of the ISR as  $n*8$ .
  - b) If opcode of **CALL** is sent:
    - As Call is a **3-Byte Instruction** the  $\mu$ P send **2 more INTA signals**
    - In response to the **2<sup>nd</sup> and the 3<sup>rd</sup> INTA** cycle the external hardware returns the **lower and the higher byte of the address** of the ISR respectively.

This is how the address is determined when INTR occurs.

### INTA : (Interrupt Acknowledge)

- This is an **active low acknowledge** signal going out of 8085.
- This signal is **given in response to** an interrupt on **INTR ONLY**.
- After the **first INTA** is given, the interrupting **peripheral sends an opcode**.
- **If the Opcode is of RSTn**, then the **ISR address is calculated as  $n \times 8$** .
- **If the Opcode is of CALL**, then **two more INTA signals** are given and, the lower byte, and then the higher byte of the ISR address are sent by the peripheral.
- #Please refer Bharat Sir's video at [www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com) for understanding this..

### EI and DI Instructions

### INTE F/F : (Interrupt Enable Flip Flop)

- This flip-flop decides if interrupts are enabled in the  $\mu$ P i.e. **if it is set, all interrupts are enabled**.
  - It is **set by the EI Instruction**.
  - It is **reset** in the following 3 ways:
    - i. **If  $\mu$ P is reset**.
    - **If DI instruction** is executed.
    - **If any other interrupt is recognized** by the  $\mu$ P. In this case the INTE F/F is later set in the ISR by EI.  
☺ In case of doubts, contact Bharat Sir: - 98204 08217.
- The INTE F/F affects all interrupts **EXCEPT TRAP**, as it cannot be disabled.

### Note

On reset by default interrupts are disabled.

If we don't write EI in our program hardware interrupts will not be serviced except TRAP.



Interrupt	Priority	Triggering	Maskable by SIM	Disabled by DI	Vectored	Vector Address
TRAP	1	Edge / Level	No	No	Yes	0024 H
RST 7.5	2	Edge	Yes	Yes	Yes	003C H
RST 6.5	3	Level	Yes	Yes	Yes	0034 H
RST 5.5	4	Level	Yes	Yes	Yes	002C H
INTR	5	Level	No	Yes	No	Get ISR Address from External Hardware

**Special Note:**

If you are learning this by piracy, then you are not my student. You are simply a thief!  
#PoorUpbringing

### **Bharat Acharya Education**

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes

Free: PDFs of Theory explanation, VIVA questions and answers, Multiple-Choice Questions

Start Learning... NOW!

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

**Order our Books here...**

### **8086 Microprocessor**

Link: <https://amzn.to/3qHDpJH>

### **8051 Microcontroller**

Link: <https://amzn.to/3aFQkXc>

Official WhatsApp number:

**+91 9136428051**

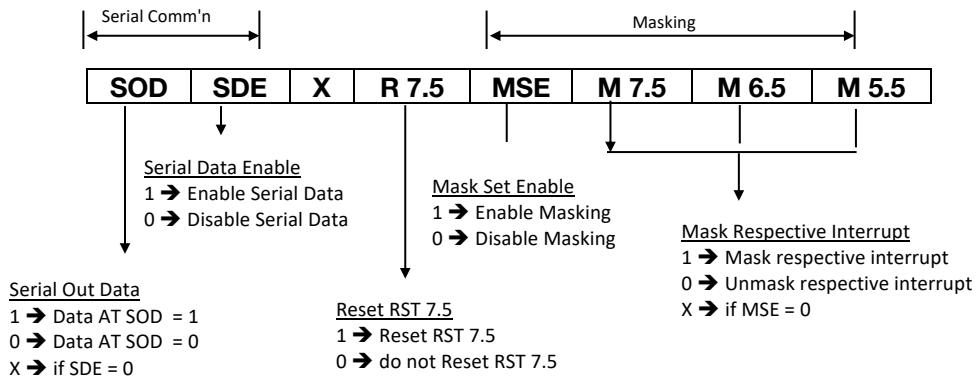


## SIM & RIM INSTRUCTIONS

### SIM: SET INTERRUPT MASK

SIM is a multipurpose instruction.  
It is used for the following...

- To **Mask** or Un-Mask the **RST7.5, RST6.5 and RST5.5** interrupts.
- To send the data out serially (bit - by - bit) through the **SOD** line of the **μP**.
- To **reset RST7.5** interrupt irrespective of whether it is masked or not.



#### Method of execution:

- The appropriate byte is formed and **loaded into the Accumulator**.
- Then the **SIM Instruction is executed**.
- The **μP reads the contents of the accumulator** in the above order.

#### Note

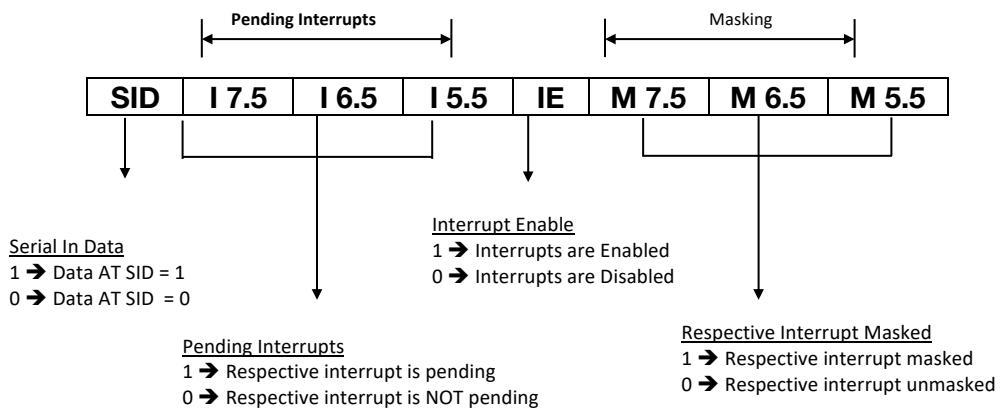
If We have disabled interrupts using DI, then whatever masking we do in SIM is of no use. First, we must enable interrupts using EI instruction. Only Then the masking pattern we give in SIM will come into effect.



## RIM: READ INTERRUPT MASK

This instruction is used for the following purposes:

- To read the **Interrupt Mask** of the  $\mu$ P.
- To accept data serially through the **SID** pin.
- To see the "**Pending Interrupts**" of the  $\mu$ P.



### Pending Interrupts:

Pending interrupts are those interrupts, which are waiting to be serviced. An interrupt becomes pending as a higher priority interrupt is currently being serviced. RIM Instruction indicates the Pending Status of RST7.5, RST6.5 and RST5.5.

### Method of execution:

- Then the **RIM** Instruction is **executed**.
- The  **$\mu$ P loads** the appropriate byte into the **Accumulator**.
- The programmer reads the contents of the Accumulator.

### Special Note:

If you are learning this by piracy, then you are not my student. You are simply a thief!  
#PoorUpbringing



## SQUARE WAVE PROGRAM USING SIM

Write a program to generate a SQUARE-WAVE of 1 KHz using SOD pin of 8085.

**Soln:**

```
BACK: MVI A, 40H ; SIM Command = 0100 0000
      SIM
      CALL DLAY
      MVI A, C0H ; SIM Command = 1100 0000
      SIM
      CALL DLAY
      JMP BACK
```

For a square wave of 1 KHz, the time period is 1 msec.  
Hence the required delay is of 0.5 msec.

**Assume 8085 is working at 3 MHZ**

```
DLAY: MVI B, XXH      ; 7 T-states ... ... ... Count is calculated later
BACK: DCR B          ; 4 T-states ... ... ... Decrement Count
      JNZ BACK        ; 10T (true) / 7T (false)
      RET             ; 10T-states
```

$$\begin{aligned} T_D &= MT + [(Count)_d \times NT] - 3T \\ \text{Here } MT &= \text{Time outside the loop} = 17T \\ NT &= \text{Time inside the loop} = 14T \\ T_D &= 17T + [(Count)_d \times 14T] - 3T \end{aligned}$$

$$\begin{aligned} \text{Required } T_D &= 0.5 \text{ msec} = 0.5 \times 10^{-3} \text{ sec} \\ 1T &= 0.333 \mu\text{sec} = 0.333 \times 10^{-6} \text{ sec} \end{aligned}$$

Substituting the above values we get:  
 $0.5 \times 10^{-3} = 17 \times (0.333 \times 10^{-6}) + [(Count)_d \times 14 \times (0.333 \times 10^{-6})] - 3 \times (0.333 \times 10^{-6})$

**Count = 6AH**

### Special Note:

If you are learning this by piracy, then you are not my student. You are simply a thief!  
#PoorUpbringing



## Bharat Acharya Education

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes

Free: PDFs of Theory explanation, VIVA questions and answers, Multiple-Choice Questions

Start Learning... NOW!

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

Order our Books here...

### 8086 Microprocessor

Link: <https://amzn.to/3qHDpJH>

### 8051 Microcontroller

Link: <https://amzn.to/3aFQkXc>

Official WhatsApp number:

**+91 9136428051**



# 8259 PROGRAMMABLE INTERRUPT CONTROLLER

## SALIENT FEATURES

- 8259 is a **Programmable Interrupt Controller** (PIC) designed to work with various microprocessors such as **8085, 8086** etc.
- **8259 is basically used to increase the number of interrupts.**
- A **single 8259** can handle **8 interrupts**
- A **cascaded** configuration of 8 slave 8259's and 1 master 8259 can handle **64 interrupts**.
- 8259 has a **flexible priority** structure.
- 8259 can **handle edge** as well as **level triggered** interrupts.
- In 8259 interrupts can be **masked** individually.
- The **Vector address** of the interrupts is **programmable**.
- Status of interrupts (pending, In-service, masked) can be easily read by the  $\mu$ P.

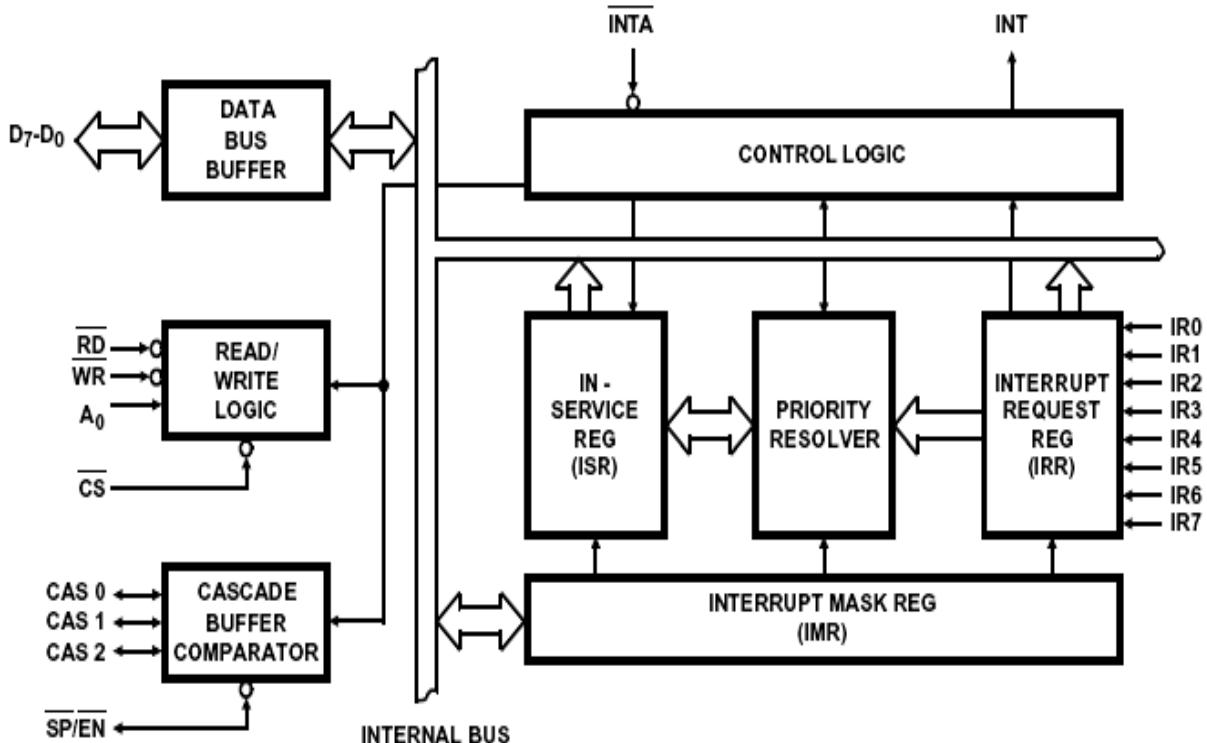
### Note: Initialization of 8259

- 8259 must be compulsorily initialized.
- During Initialization we give commands to 8259 by which we decide priority, trigger, masking etc.
- The MOST important thing we give are vector addresses for the interrupts.
- In a cascaded configuration every 8259 must be individually initialized.

### Special Note:

If you are learning this by piracy, then you are not my student.  
You are simply a thief! #PoorUpbringing

## ARCHITECTURE OF 8259



### 1) Interrupt Request Register (IRR)

- 8259 has **8 interrupt input lines IR<sub>7</sub> ... IR<sub>0</sub>**.
- The IRR is an **8-bit register** having **one bit for each** of the **interrupt lines**.
- When an **interrupt request** occurs on any of these lines, the **corresponding bit** is **set** in the **Interrupt Request Register (IRR)**.

### 2) In-Service Register (InSR)

- It is an **8-bit register**, which **stores** the **level** of the **Interrupt Request**, which is **currently** being **serviced**.

### 3) Interrupt Mask Register (IMR)

- It is an **8-bit register**, which stores the **masking pattern** for the **interrupts** of 8259.
- It stores **one bit per interrupt level**.



#### 4) Priority Resolver

- It **examines** the **IRR**, **InSR**, and **IMR** and determines which interrupt is of **highest priority** and should be sent to the **μP**.

#### 5) Control Logic

- It has **INT output** signal **connected to** the **INTR** of the **μP**, to **send** the **Interrupt** to the **μP**.
- It also has the **INTA input** signal **connected to** the **INTA** of the **μP**, to **receive** the interrupt **acknowledge**.
- It is also used to control the remaining blocks.

#### 6) Data Bus Buffer

- It is a bi-directional buffer used to **interface** the internal **data bus** of 8259 with the external (system) data bus.

#### 7) Read/Write Logic

- It is used to accept the RD, WR, A<sub>0</sub> and CS signal.
- It also holds the Initialization Command Words (ICW's) and the Operational Command Words (OCW's).

#### 8) Cascade Buffer / Comparator

- It is used in **cascaded mode** of operation.
- It has two components:

##### i. CAS<sub>2</sub>, CAS<sub>1</sub>, CAS<sub>0</sub> lines:

- These lines are **output for the master, input for the slave**.
- The **Master sends** the **address of the slave** on these lines (hence output).
- The **Slaves read** the **address** on these lines (hence input).
- As there are 8 interrupt levels for the Master, there are **3 CAS lines** ( $\because 2^3 = 8$ ).

##### ii. SP/EN (Slave Program/Master Enable):

- In **Buffered Mode**, it **functions** as the **EN line** and is used to **enable** the **buffer**.
- In **Non buffered mode**, it **functions** as the **SP output line**.
- **For Master 8259 SP** should be **high**, and **for the Slave SP** should be **low**.



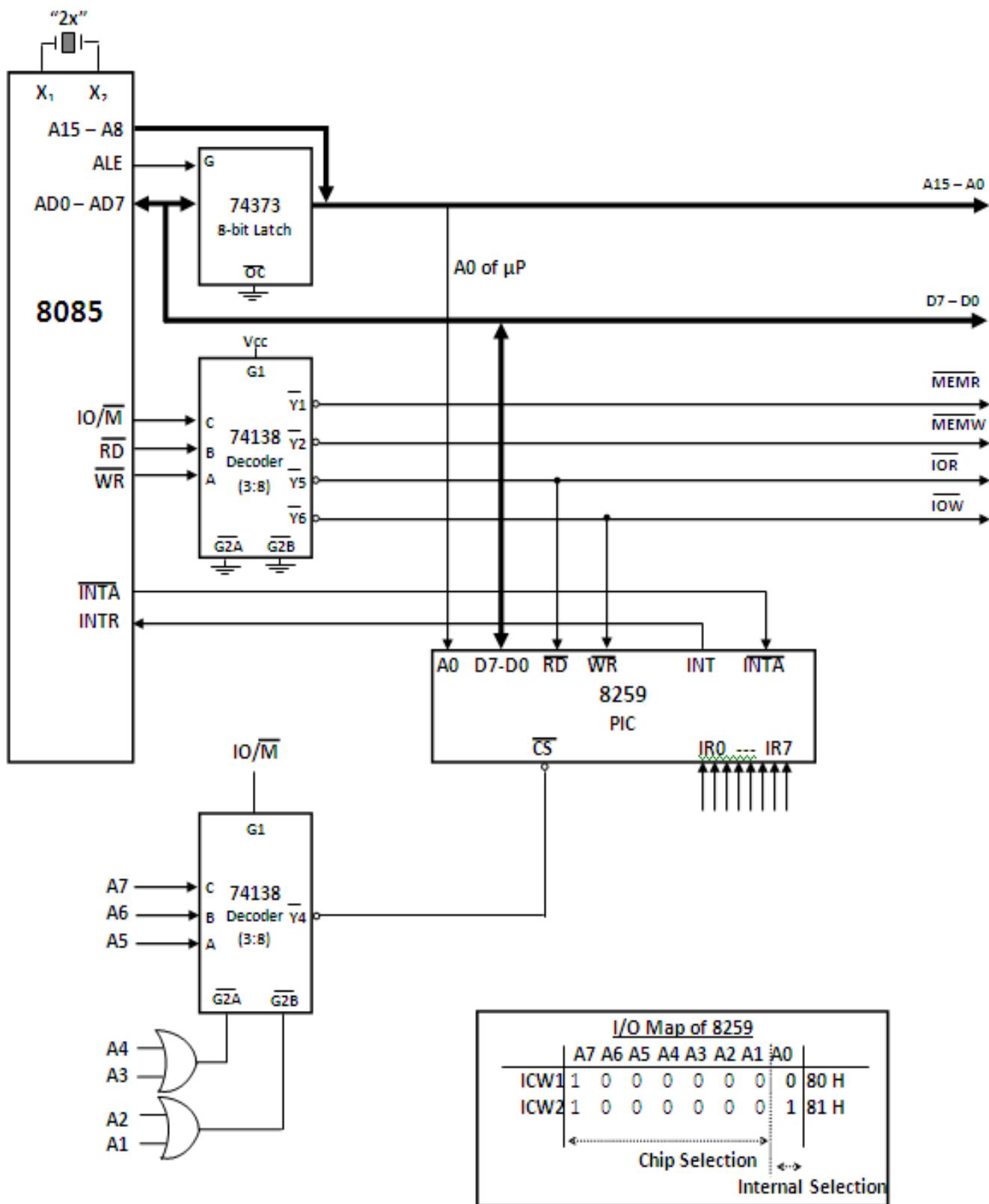
## INTERFACING AND WORKING OF A SINGLE 8259

The working of a single 8259 with 8085, 8259 is explained below.

1. **First of all, interrupt INTR of 8085 must be enabled** by the **EI** instruction.
2. **8259 is initialized** by giving the necessary commands. (**ICWs**)
3. **Once 8259 is initialized, the following sequence** of events takes place when one or more **interrupts occur** on the IR lines of the 8259.
4. The **corresponding bit** for an interrupt is **set in IRR**.
5. The **Priority Resolver checks** the 3 registers:  
**IRR** (for highest interrupt request)  
**IMR** (for the masking Status)  
**InSR** (for the current level serviced)  
and **determines the highest priority interrupt**.  
It **sends the INT signal to the μP**.
6. The **μP finishes the current instruction** and **acknowledges** the interrupt by **sending the first INTA pulse**.
7. On receiving the INTA signal, the **corresponding bit** in the **InSR** is **set** (indicating that the service of this interrupt is started) and the **bit** in the **IRR** is **reset** (to indicate that the request is accepted).  
**8259 now sends the Opcode of CALL instruction to the μP** on the data bus.
8. The **μP decodes the CALL instruction** and **sends 2 more INTA pulses** to the 8259.
9. In response to the two INTA pulses, the **8259 sends the address of the ISR to the μP**. First the lower byte and then the higher byte.
10. Thus, the complete 3-byte CALL Instruction code is released by the 8259.
11. The **μP pushes the contents of PC onto the Stack** and **transfers program to the address of the ISR sent by the 8259**. **The ISR thus begins**.
12. At the end of the ISR, 8085 will give an **EOI command** to make the corresponding bit 0 in In Service Register. *In case of doubts WhatsApp: +919136428051 (Only for registered students!)*

### Special Note:

You will learn the interfacing diagram in the next page in later lectures. But the explanation of the interface is already covered as you learn the architecture.





## **Bharat Acharya Education**

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes

Free: PDFs of Theory explanation, VIVA questions and answers, Multiple-Choice Questions

Start Learning... NOW!

**[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)**

**Order our Books here...**

### **8086 Microprocessor**

Link: <https://amzn.to/3qHDpJH>

### **8051 Microcontroller**

Link: <https://amzn.to/3aFQkXc>

Official WhatsApp number:

**+91 9136428051**

## 8259 PIC CASCADED CONFIGURATION

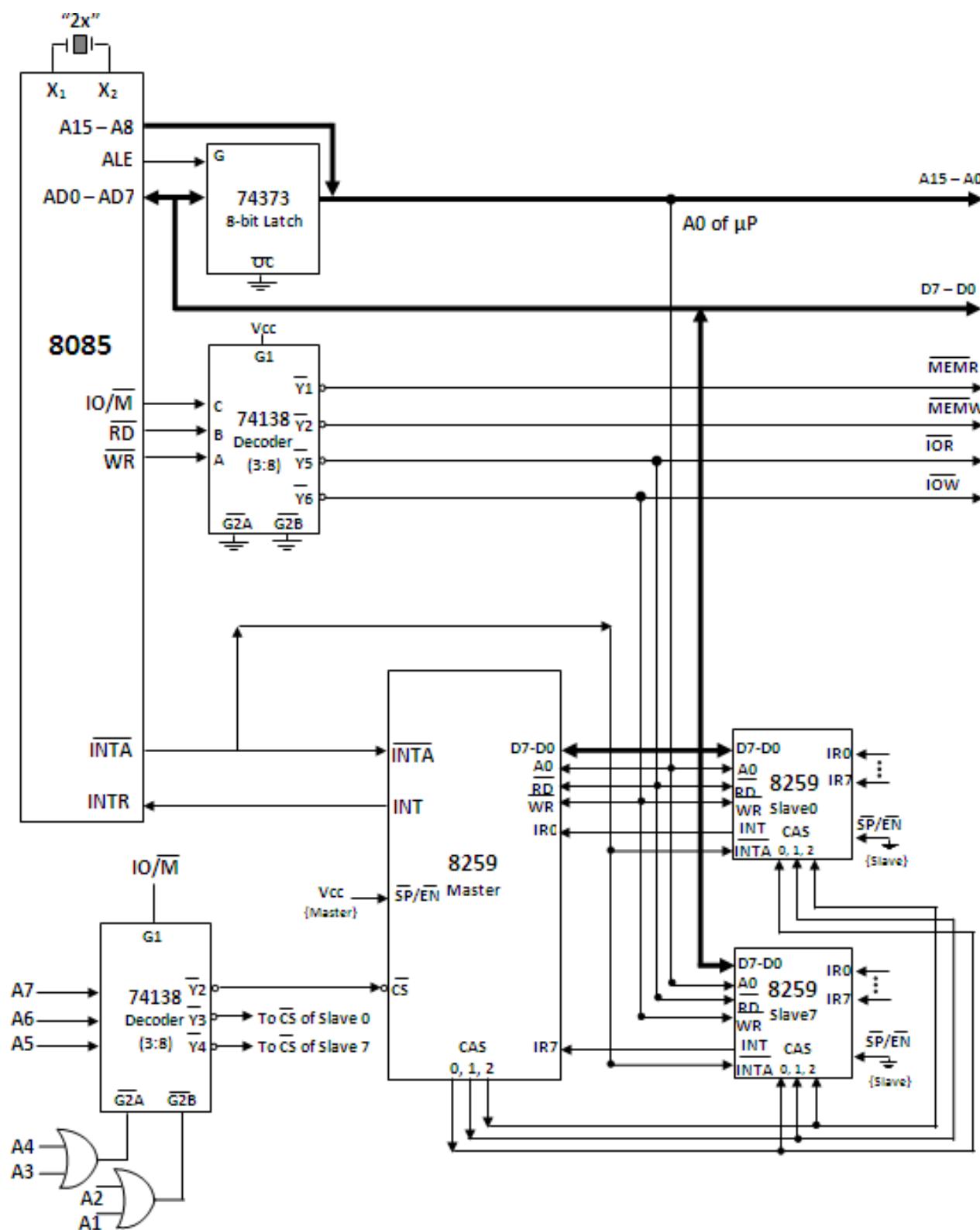
- When **more than one 8259s** are connected to the  $\mu P$ , it is called as a **Cascaded configuration**.
- A Cascaded configuration **increases the number of interrupts** handled by the system.
- As the **maximum** number of **8259s** interfaced can be **9** (1 Master and 8 Slaves) the **Maximum number of interrupts** handled can be **64**.
- The **master 8259** has **SP/EN = +5V** and the **slave** has **SP/EN = 0V**.
- **Each slave's INT output** is **connected to the IR input of the Master**.
- The **INT output of the Master** is **connected to the INTR input of the  $\mu P$** .
- The **master addresses** the individual **slaves through the CAS<sub>2</sub>, CAS<sub>1</sub>, CAS<sub>0</sub> lines** connected from the master to each of the slaves.
- **Each 8259** (Master or Slave) **has its own address** and **has to be initialized separately**.

When an **interrupt request** occurs on a **SLAVE**, the **following sequence** of events is executed:

- 1) The **slave 8259 resolves the priority** of the interrupt and **sends the interrupt to the master 8259**.
- 2) The **master resolves the priority** among its slaves and **sends the interrupt to the  $\mu P$** .
- 3) The  **$\mu P$  finishes the current instruction** and **responds to the interrupt by sending 3 INTA pulses**.
- 4) In response to the first INTA pulse the master does the following tasks:
  - It **sends the opcode of CALL instruction** to the  $\mu P$  on the data bus.
  - It **sends the 3-bit slave identification number** on the **CAS lines**.
  - The **Master sets the corresponding bit** in its **InSR**.
  - The **Slave identifies** its number on the **CAS lines** and **sets the corresponding bit in its InSR**. *In case of doubts WhatsApp: +919136428051 (Only for registered students!)*
  - In response to the second and third INTA pulse the **slave places the lower byte and then the higher byte of the ISR address** on the data bus **respectively**.
- 5) During the third INTA pulse the **InSR bit of the slave is cleared** in **AEOI mode**, otherwise it is cleared by the **EOI command** at the end of the ISR.
- 6) The  **$\mu P$  pushes** the contents of **PC onto the Stack** and **transfers program to the address of the ISR sent by the slave 8259**. The **ISR thus begins**.

### Note: Initialization of cascaded 8259

- Every 8259 must be initialized.
- Every 8259 is given vector address.
- Master is informed on which lines there are slaves.
- Slave is informed its ID number





## I/O map for the Cascaded Configuration

I/O Chip	Address Bus								I/O port Address
	A7	A6	A5	A4	A3	A2	A1	A0	
<b>8259 Master</b>									
<b>ICW1</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>40 H</b>
<b>ICW2</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>41 H</b>
<b>8259 Slave 0</b>									
<b>ICW1</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>60 H</b>
<b>ICW2</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>61 H</b>
<b>8259 Slave 7</b>									
<b>ICW1</b>	<b>1</b>	<b>0</b>	<b>80 H</b>						
<b>ICW2</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>81 H</b>

### Note: I/O map and designing

You will understand this once you learn 8259 designing.

### Special Note:

If you are learning this by piracy, then you are not my student.  
You are simply a thief!  
#PoorUpbringing

## Bharat Acharya Education

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes  
Free: PDFs of Theory explanation, VIVA questions and MCQs

Start Learning... NOW!

**[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)**

Official WhatsApp number:

**+91 9136428051**



## 8259 PIC OPERATING MODES

### PRIORITY MODES

#### Fully Nested Mode (FNM)

It is the **default priority mode** of 8259.

It is a **fixed priority** mode.

**IR<sub>0</sub>** has the **highest** priority and **IR<sub>7</sub>** has the **lowest** priority.

It is preferred for “**Single**” 8259.

#### Special Fully Nested Mode (SFNM)

This mode can be **used for the Master 8259 in a cascaded configuration**.

Its **priority structure** is fixed and is the **same as FNM** (IR<sub>0</sub> highest and IR<sub>7</sub> lowest).

**Additionally**, in SFCM, the **Master would recognize a higher priority interrupt from a slave, whose another interrupt is currently being serviced**. This is **possible only in SFCM**.

(Refer to the diagram in the video for this topic at [www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com))

### Rotating Priority Modes

There are **two** rotating priority modes: Automatic Rotation and Specific Rotation

#### Automatic Rotation Mode

This is a rotating priority mode.

It is **preferred** when **several interrupt sources are of equal priority**.

In this mode, **after a device receives service, it gets the lowest priority**.

**All other priorities rotate subsequently.** 😊 For doubts contact Bharat Sir on 98204 08217

**Eg:** If IR<sub>2</sub> is has just been serviced, it will get the lowest priority.

#### Specific Rotation Mode

It is **also** a rotating priority mode, **but here the user can select any IR level for lowest priority**, and thus fix all other priorities.

#### Note: Domination and Starvation

- FNM and SFCM are fixed priority modes. They cause domination by higher priority interrupts if they occur repeatedly. In that case the lower priority interrupts starve. To avoid this, we use rotating priority.



## Special Mask Mode (SMM)

Usually 8259 prevents interrupt requests lower or equal to the interrupt, which is currently in service. In SMM 8259 permits interrupts of all levels (lower or higher) except the one currently in service. As we are specially masking the current interrupt, it is called Special Mask Mode. This mode is preferred when we don't want priority

## Poll Mode (Polling means asking)

Here the INT line of 8259 is not used hence 8259 cannot interrupt the µP. Instead, the µP will give Poll command to 8259 using OCW3. In return, 8259 provides a Poll Word to the µP. The Poll Word indicates the highest priority interrupt, which requires service.

**Poll Word**

I	x	X	x	x	W <sub>2</sub>	W <sub>1</sub>	W <sub>0</sub>
1= Valid Interrupt 0 = No valid interrupt					0	0	0
					0	0	1
					0	1	0
					0	1	1
Level No of the highest priority interrupt to be serviced					1	0	0
					1	0	1
					1	1	0
					1	1	1

Thereafter the µP services the interrupt. 😊 For doubts contact Bharat Sir on 98204 08217

**Advantage:** The µP's program is not disturbed. It can be used when the ISR is common for several Interrupts. It can be used to increase the total number of interrupts beyond 64.

**Drawback:** If the polling interval is too large, the interrupts will be serviced after long intervals. If the polling interval is small, lot of time may be wasted in unnecessary polls.

## Buffered Mode

In this mode SP / EN becomes low during INTA cycle.

This signal is used to enable the buffer, and is mainly used in 8086 circuits.

### Special Note:

If you are learning this by piracy, then you are not my student.  
You are simply a thief! #PoorUpbringing

## **EOI – (End Of Interrupt)**

When the **μP** responds to an interrupt request by **sending the first INTA** signal, the **8259** sets the **corresponding bit** in the In Service Register (**InSR**).  
This **begins** the **service** of the interrupt.

When this **bit** in the In Service Register is **cleared**, it is called as **End of Interrupt (EOI)**.

### **EOI Modes**

#### **Normal EOI Mode:**

Here an EOI Command is necessary. The EOI Command is given by the programmer at the end of the ISR. It causes 8259 to clear the bit from In Service Register. There are two types of EOI Commands:

#### **Non Specific EOI Command:**

Here the programmer doesn't specify the Bit number to be cleared. 8259 automatically clears the highest priority bit from In Service Register.

#### **Specific EOI Command:**

Here the programmer specifies the Bit number to be cleared from In Service Register.

#### **Auto EOI Mode (AEOI):**

In AEOI mode the EI command is not needed. Instead, 8259 will itself clear the corresponding bit from In Service Register at the end of the 3<sup>rd</sup> **INTA** pulse.

### **Bharat Acharya Education**

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes  
Free: PDFs of Theory explanation, VIVA questions and MCQs

Start Learning... NOW!

**[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)**

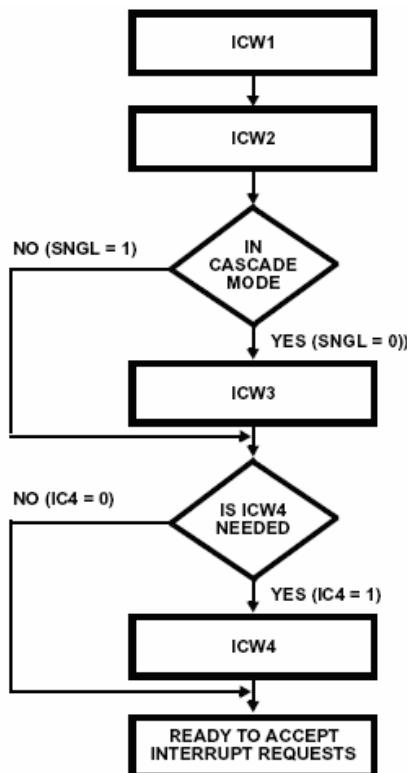
Official WhatsApp number:

**+91 9136428051**



## 8259 COMMANDS AND PROGRAMMING

### 8259 COMMANDS



As seen above there are **two types** of control words, **Initialization Control Words (ICWs)** and **Operational Control Words (OCWs)**.

#### ICWs

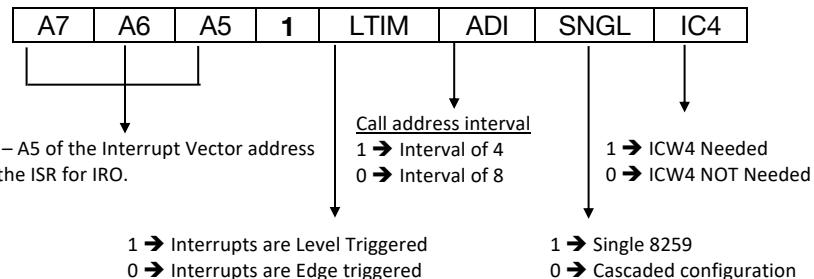
- ICWs have to be **given during the initialization** of 8259 (i.e. **before** the µP can start **using 8259**).
- ICW1 and ICW2 are **compulsory**.
- If Cascaded, ICW3 has to be given.
- Whether ICW4 is **required** or not, is **specified in the ICW1**.
- If ICW4 is **required**, it has to be **written**.
- It is **important** that the ICWs are **written in the above sequence only**.

#### OCWs

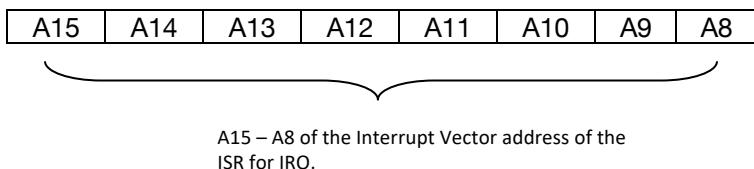
- OCWs are **given during the operation** of 8259 (i.e. **after** the µP has **started using 8259**).
- OCWs are **neither compulsory, nor** do they have a **specific sequence**.
- They are mainly used to alter the **masking** status and the **operation modes** of 8259.



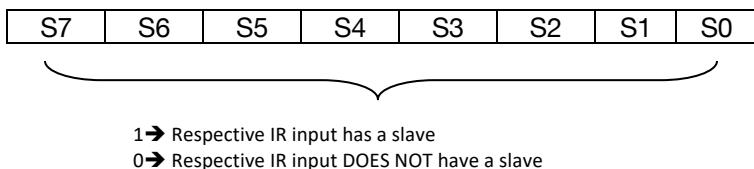
### ICW-1 ( $A_0 = 0$ )



### ICW-2 ( $A_0 = 1$ )



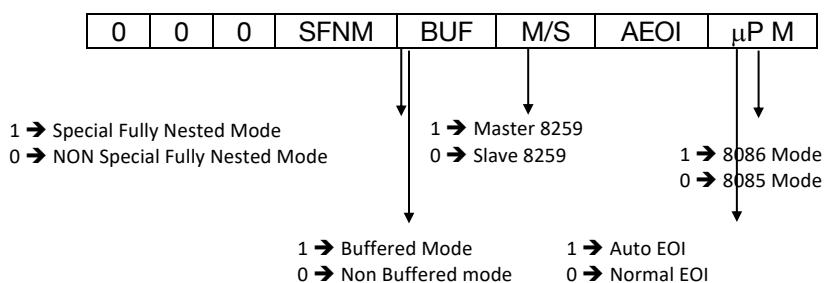
### ICW-3 MASTER ( $A_0 = 1$ )



### ICW-3 SLAVE ( $A_0 = 1$ )

0	0	0	0	0	<b>ID<sub>2</sub></b>	<b>ID<sub>1</sub></b>	<b>ID<sub>0</sub></b>
					<b>ID<sub>2</sub></b>	<b>ID<sub>1</sub></b>	<b>Slave ID</b>
					0	0	0
					0	0	1
					0	1	2
					0	1	3
					1	0	4
					1	0	5
					1	1	6
					1	1	7

### ICW-4 ( $A_0 = 1$ )





### OCW-1 ( $A_0 = 1$ )

M7	M6	M5	M4	M3	M2	M1	M0
----	----	----	----	----	----	----	----



1 → MASK Respective IR input  
0 → UNMASK Respective IR input

### OCW-2 ( $A_0 = 0$ )

R	SL	EOI	0	0	L2	L1	L0
---	----	-----	---	---	----	----	----

R	SL	EOI	Action	
0	0	1	NON Specific EOI Command	End Of Interrupt
0	1	1	Specific EOI Command	
1	0	1	Rotate on NON Specific EOI	Auto rotation
1	0	0	Rotate in AUTO EOI Mode (Set)	
0	0	0	Rotate in AUTO EOI Mode (Clear)	Specific rotation
1	1	1	Rotate on Specific EOI Command	
1	1	0	Set Priority command	Specific rotation
0	1	0	NOP	

L2	L1	L0	IR Level
0	0	0	IR0
0	0	1	IR1
0	1	0	IR2
0	1	1	IR3
1	0	0	IR4
1	0	1	IR5
1	1	0	IR6
1	1	1	IR7

### OCW-3 ( $A_0 = 0$ )

X	ESMM	SMM	0	1	P	RR	RIS
---	------	-----	---	---	---	----	-----



1 → POLL Command  
0 → No Poll

ESMM	SMM	Action
0	0	No Action
0	1	
1	0	Exit SMM
1	1	Enter SMM

RR	RIS	Action
0	0	No Action
0	1	
1	0	Read IRR on next RD pulse
1	1	Read InSR on next RD pulse

#### Special Note:

If you are learning this by piracy, then you are not my student.  
You are simply a thief! #PoorUpbringing

# 8259 PROGRAMMING

## Question

Initialise 8259 as follows...

- Edge Triggered
- Single
- Address interval of 8
- ISR Address of IR0 is 7580H
- AEOI
- Mask IR4 and IR5
- Assume I/O address of 8259 is 80H

### Special Note:

I have taken the exact same question as the one solved in my video lecture at [www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

## Solution

**ICW1 = 1001 0011 = 93H**

**ICW2 = 0111 0101 = 75H**

**ICW4 = 0000 0010 = 02H**

**OCW1 = 0011 0000 = 30H**

## Program

<b>MVI A, 93H</b>	; Load command value for <b>ICW1</b> into A
<b>OUT 80H</b>	; Send at appropriate address
<b>MVI A, 75H</b>	; Load command value for <b>ICW2</b> into A
<b>OUT 81H</b>	; Send at appropriate address
<b>MVI A, 02H</b>	; Load command value for <b>ICW4</b> into A
<b>OUT 81H</b>	; Send at appropriate address
<b>MVI A, 30H</b>	; Load command value for <b>OCW1</b> into A
<b>OUT 81H</b>	; Send at appropriate address

## Bharat Acharya Education

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes

Free: PDFs of Theory explanation, VIVA questions and MCQs

Start Learning... NOW!

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

Official WhatsApp number:

**+91 9136428051**

## 8259 INTERFACING & DESIGNING

### INTERFACING & WORKING OF A SINGLE 8259

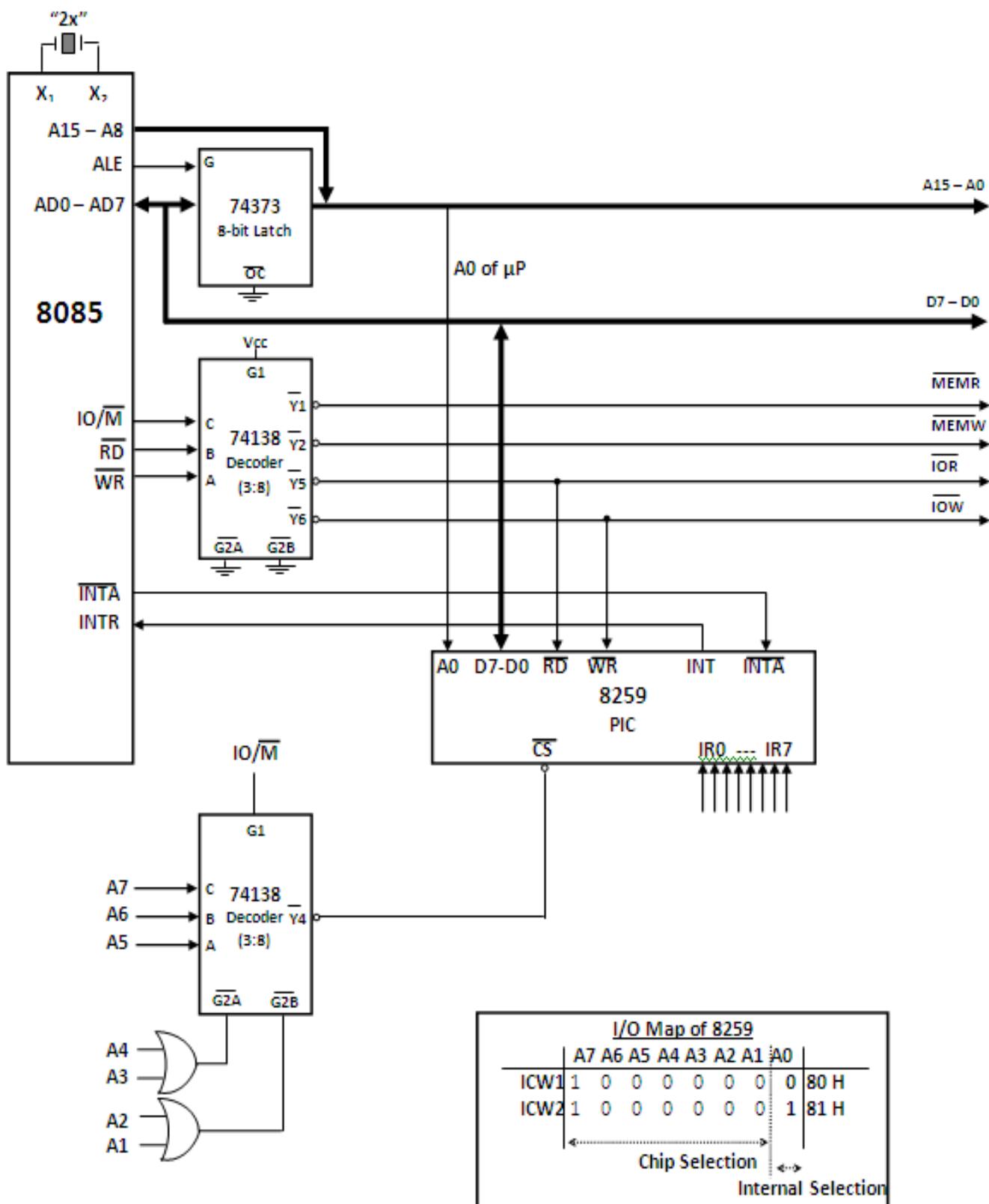
**INTE flip-flop** of the **μP** is **set** by the **EI** instruction and the **individual interrupts enabled** using **SIM** instruction. **8259 is initialized** by giving **ICW1** and **ICW2** (compulsory) and **ICW4** (optional). Note that **ICW3 is not given** as Single 8259 is used. OCWs are given if required.

Once 8259 is initialized, the **following sequence** of events takes place when one or more **interrupts occur** on the IR lines of the 8259.

- 1) The **corresponding bit** for an interrupt is **set in IRR**.
- 2) The **Priority Resolver** checks the 3 registers:  
**IRR** (for highest interrupt request)  
**IMR** (for the masking Status)  
**InSR** (for the current level serviced)  
and **determines** the **highest priority** interrupt.  
It **sends** the **INT** signal to the **μP**.
- 3) The **μP finishes** the **current instruction** and **acknowledges** the interrupt by **sending** the **first INTA pulse**.
- 4) On receiving the INTA signal, the **corresponding bit** in the **InSR** is **set** (indicating that the service of this interrupt is started) and the **bit** in the **IRR** is **reset** (to indicate that the request is accepted).  
**8259 now sends the Opcode of CALL instruction to the μP** on the data bus.
- 5) The **μP decodes** the **CALL instruction** and **sends 2 more INTA pulses** to the 8259.
- 6) In response to the two INTA pulses, the **8259 sends the address of the ISR to the μP**.  
First the lower byte and then the higher byte.
- 7) Thus, the complete 3-byte CALL Instruction code is released by the 8259.  
**In the AEOI Mode the InSR bit is reset** at this point, **otherwise it remains set until an appropriate EOI command is given at the End of the ISR**.
- 8) The **μP pushes** the contents of **PC** onto the **Stack** and **transfers program to the address** of the **ISR** sent by the 8259. **The ISR thus begins.** #Please refer Bharat Sir's Lecture Notes for this ...

#### Special Note:

If you are learning this by piracy, then you are not my student.  
You are simply a thief! #PoorUpbringing





## INTERFACING & WORKING OF CASCADED 8259

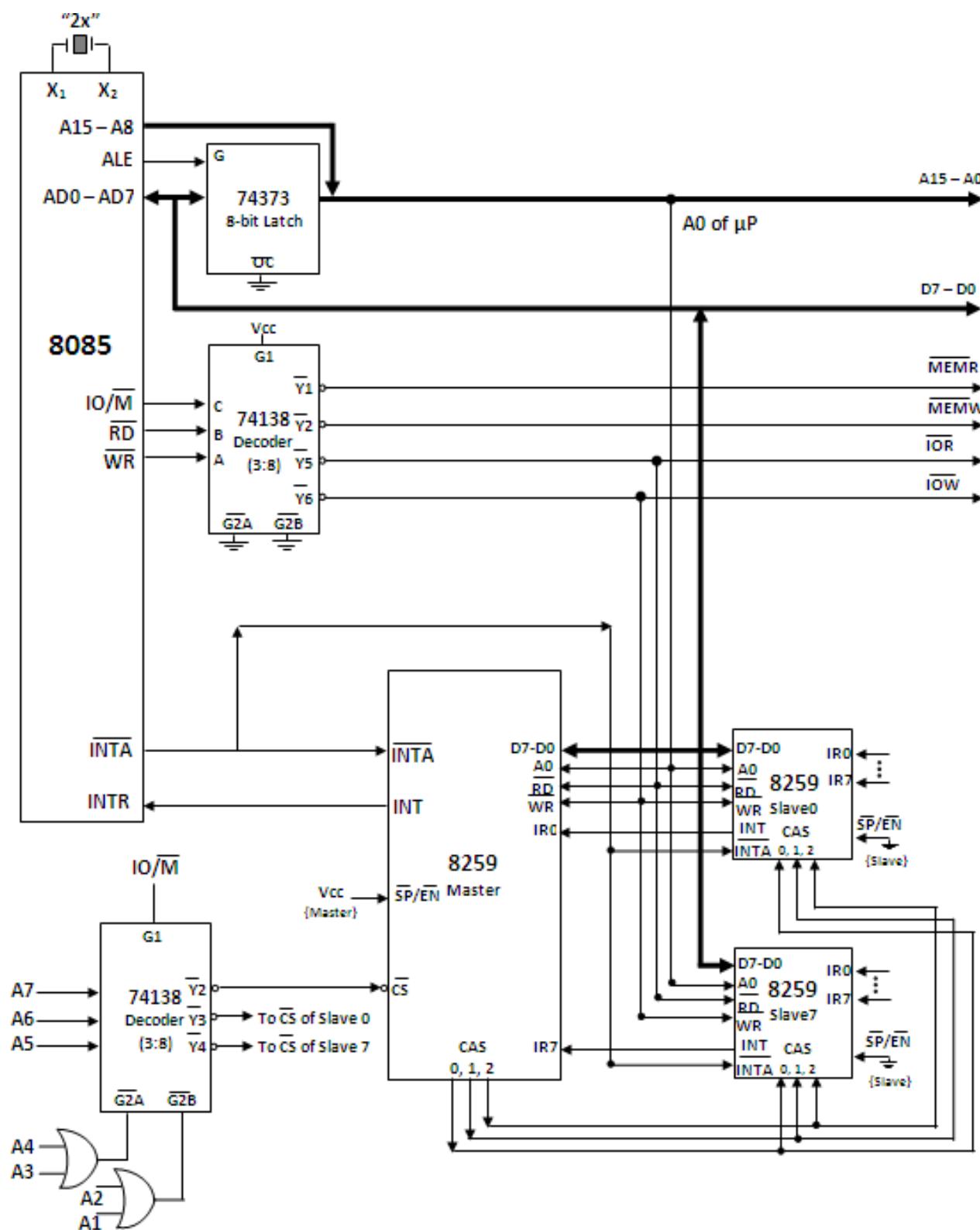
- When **more than one 8259s** are connected to the **μP**, it is called as a **Cascaded configuration**.
- A Cascaded configuration **increases** the **number of interrupts** handled by the system.
- As the **maximum** number of **8259s** interfaced can be **9** (1 Master and 8 Slaves) the **Maximum** number of **interrupts** handled can be **64**.
- The **master 8259** has **SP/EN = +5V** and the **slave** has **SP/EN = 0V**.
- **Each slave's INT output is connected to the IR input of the Master.**
- The **INT output of the Master is connected to the INTR input of the μP**.
- The **master addresses** the individual **slaves through the CAS<sub>2</sub>, CAS<sub>1</sub>, CAS<sub>0</sub> lines** connected from the master to each of the slaves.
- **Each 8259 (Master or Slave) has its own address and has to be initialized separately.**

When an **interrupt request** occurs on a **SLAVE**, the **following sequence** of events is executed:

- 1) The **slave 8259 resolves the priority** of the interrupt and **sends the interrupt to the master 8259**.
- 2) The **master resolves the priority** among its slaves and **sends the interrupt to the μP**.
- 3) The **μP finishes the current instruction** and **responds** to the interrupt by **sending 3 INTA pulses**.
- 4) **In response to the first INTA pulse the master does the following tasks:**
  - i. It **sends the opcode of CALL instruction to the μP** on the data bus.
  - ii. It **sends the 3-bit slave identification number** on the **CAS lines**.
  - iii. The **Master sets the corresponding bit** in its **InSR**.
  - iv. The **Slave identifies** its number on the **CAS lines** and **sets the corresponding bit** in its **InSR**.
- 5) **In response to the second and third INTA pulse** the **slave places the lower byte and then the higher byte of the ISR address** on the data bus **respectively**.
- 6) **During the third INTA pulse** the **InSR bit of the slave is cleared in AEOI mode, otherwise it is cleared by the EOI command** at the end of the ISR.
- 7) The **μP pushes** the contents of **PC onto the Stack** and **transfers program** to the **address of the ISR sent by the slave 8259**. **The ISR thus begins**.

### Special Note:

If you are learning this by piracy, then you are not my student.  
You are simply a thief! #PoorUpbringing





## I/O for the Cascaded Configuration

I/O Chip	Address Bus								I/O port Address
	A7	A6	A5	A4	A3	A2	A1	A0	
<b>8259 Master</b>									
<b>ICW1</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>40 H</b>
<b>ICW2</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>41 H</b>
<b>8259 Slave 0</b>									
<b>ICW1</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>60 H</b>
<b>ICW2</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>61 H</b>
<b>8259 Slave 7</b>									
<b>ICW1</b>	<b>1</b>	<b>0</b>	<b>80 H</b>						
<b>ICW2</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>81 H</b>

### **Bharat Acharya Education**

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes  
Free: PDFs of Theory explanation, VIVA questions and MCQs

Start Learning... NOW!

**[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)**

Official WhatsApp number:

**+91 9136428051**

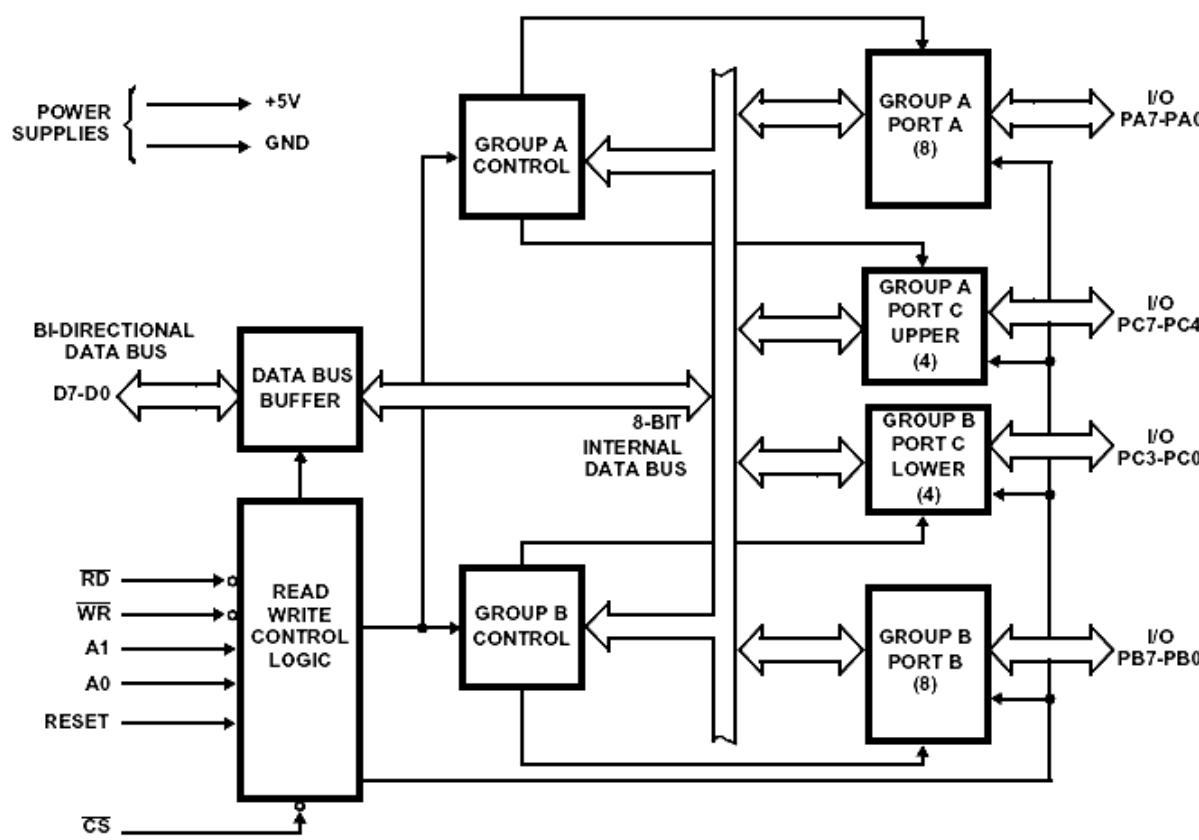


## 8255 PROGRAMMABLE PERIPHERAL INTERFACE

### SALIENT FEATURES OF 8255 PPI

- It is a programmable general-purpose I/O device.
- It has 3 8-bit bi-directional I/O ports - **Port A**, **Port B**, and **Port C**.
- It provides **3 modes of data transfer** – Simple I/O, Handshake I/O and Bi-directional Handshake I/O.
- Additionally it also provides a **Bit Set Reset Mode** to alter individual bits of **Port C**, for bit interface devices.

### ARCHITECTURE OF 8255 PPI





The architecture of 8259 can be divided into the following parts:

**1) Data Bus Buffer**

- This is a 8-bit bi-directional buffer used to interface the internal data bus of 8255 with the external (system) data bus.
- The CPU transfers data to and from the 8255 through this buffer.

**2) Read/Write Control Logic**

- It accepts address and control signals from the µP.
- The Control signals determine whether it is a read or a write operation and also select or reset the 8255 chip.
- The Address bits ( $A_1, A_0$ ) are used to select the Ports or the Control Word Register as shown:

<b><math>A_1\ A_0</math></b>	<b>Selection</b>	<b>Sample address</b>
0 0	Port A	80 H (i.e. 1000 0000)
0 1	Port B	81 H (i.e. 1000 0001)
1 0	Port C	82 H (i.e. 1000 0010)
1 1	Control Word	83 H (i.e. 1000 0011)

- The Ports are controlled by their respective Group Control Registers.

**3) Group A Control**

- This Control block controls Port A and Port C<sub>Upper</sub> i.e. PC<sub>7</sub>-PC<sub>4</sub>.
- It accepts Control signals from the Control Word and forwards them to the respective Ports.

**4) Group B Control**

- This Control block controls Port B and Port C<sub>Lower</sub> i.e. PC<sub>3</sub>-PC<sub>0</sub>.
- It accepts Control signals from the Control Word and forwards them to the respective Ports.

**5) Port A, Port B, Port C**

- These are 8-bit Bi-directional Ports.
- They can be programmed to work in the various modes as follows:

<b>Port</b>	<b>Mode 0</b>	<b>Mode 1</b>	<b>Mode 2</b>
Port A	✓	✓	✓
Port B	✓	✓	✗ (Mode 0 or Mode 1)
Port C	✓	✗ (Handshake signals)	✗ (Handshake signals)

- ONLY Port C can also be programmed to work in Bit Set reset Mode to manipulate its individual bits.  
😊 In case of doubts, contact Bharat Sir: - 98204 08217.

**6)**

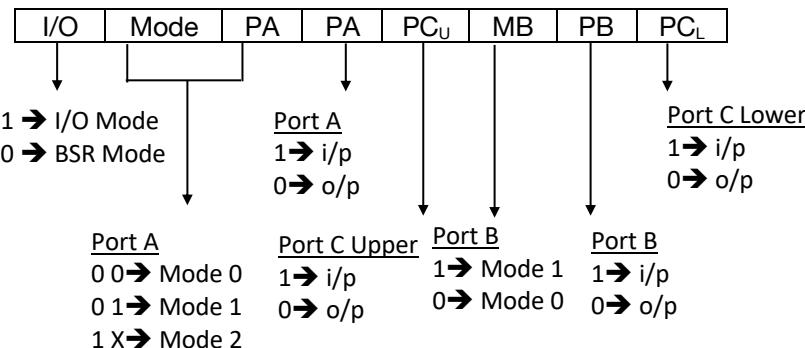
**Special Note:**

If you are learning this by piracy, then you are not my student.  
You are simply a thief! #PoorUpbringing



### Control Word of 8255 (I/O Mode)

To do 8-bit data transfer using the Ports A, B or C, 8255 needs to be in the IO mode.  
The bit pattern for the control word in the IO mode is as follows:



### 7) Control Word of 8255 (BSR Mode – Applicable ONLY for Port C)



Bit Select			Bit	Select respective bit of Port C
0	0	0	PC <sub>0</sub>	
0	0	1	PC <sub>1</sub>	
0	1	0	PC <sub>2</sub>	
0	1	1	PC <sub>3</sub>	
1	0	0	PC <sub>4</sub>	
1	0	1	PC <sub>5</sub>	
1	1	0	PC <sub>6</sub>	
1	1	1	PC <sub>7</sub>	

- The BSR Mode is used ONLY for Port C.
- In this Mode the individual bits of Port C can be set or reset.
- This is very useful for interfacing those devices, which accept bit-wise data.  
Eg: ADC Converters.
- The individual bit is selected and Set/reset through the control word.
- Since the D7 bit of the Control Word is 0, the BSR operation will not affect the I/O operations of 8255.



## **Bharat Acharya Education**

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes  
Free: PDFs of Theory explanation, VIVA questions and MCQs

Start Learning... NOW!

**[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)**

Official WhatsApp number:

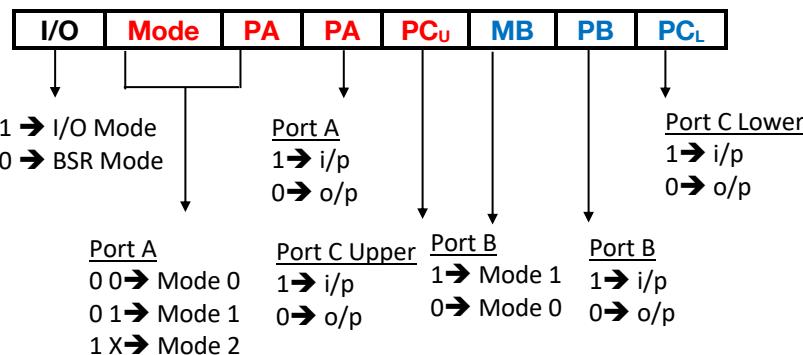
**+91 9136428051**



## 8255 PROGRAMMABLE PERIPHERAL INTERFACE

### CONTROL WORD – I/O COMMAND

- Control Word is used to give commands to 8255
- There are two commands, I/O Command and BSR Command.
- Do keep in mind, that control word is selected when A1 and A0 are 11.
- This means the address will be the last address of 8255.
- Assume 8255 is at the address 80H, then control word will be at address 83H.



- In the I/O command we decide the Mode and the direction of the ports
- I/O bit:** 1 for I/O command | 0: BSR command
- The next two bits decide the **MODE of PA**  
00: Port A is in Mode 0 | 01: Port A is in Mode 1 | 10 or 11: Port A is in Mode 2
- The next bit decides the **direction of Port A**  
1: Port A is an Input Port | 0: Port A is an Output Port
- The next bit decides the **direction of Port C Upper**  
1: Port C Upper is an Input Port | 0: Port C Upper is an Output Port
- The next bit decides the **MODE of PB**  
0: Port B is in Mode 0 | 1: Port B is in Mode 1
- The next bit decides the **direction of Port B**  
1: Port B is an Input Port | 0: Port B is an Output Port
- The next bit decides the **direction of Port C Lower**  
1: Port C Lower is an Input Port | 0: Port C Lower is an Output Port

#### Special Note:

If you are learning this by piracy, then  
you are not my student.  
You are simply a thief! #PoorUpbringing

#### Special Note:

Generally, Direction pf PC<sub>upper</sub> and PC<sub>lower</sub> is  
the same but if your question asks you to  
keep them different it is allowed to do so.



## PROGRAM QUESTION BASED ON I/O COMMAND

Q: Initialise 8255 as follows:

PA – Mode 0 – Input  
PB – Mode 0 – Output  
PC – Mode 0 – Input

Solution:

Assume 8255 is at 80H

Its various addresses will be as follows:

PA at address 80H

PB at address 81H

PC at address 82H

CW at address 83H

We will give our command at 83H to the control word.

I/O	Mode	PA	PA	PC <sub>U</sub>	MB	PB	PC <sub>L</sub>
1	0	0	1	1	0	0	1

Command Value = 99H

Instructions:

**MVI A, 99H**  
**Out 83H, A**



## CONTROL WORD – BSR COMMAND

- BSR stands for Bit – Set – Reset
- It is a facility by which the programmer can set or reset a single bit (line) of port C without affecting any other line.
- We select the line number (PC0 – PC7) out of 8 options (000 - 111)
- We then select if we want to Set it (1) or Reset it (0)
- This is very useful for interfacing those devices, which accept bit-wise data.  
Eg: ADC Converters, Waveform generation etc.



1 → Set the selected Port C bit  
0 → Reset the selected Port C bit

Bit Select			Bit
0	0	0	PC <sub>0</sub>
0	0	1	PC <sub>1</sub>
0	1	0	PC <sub>2</sub>
0	1	1	PC <sub>3</sub>
1	0	0	PC <sub>4</sub>
1	0	1	PC <sub>5</sub>
1	1	0	PC <sub>6</sub>
1	1	1	PC <sub>7</sub>

Select respective  
bit of Port C

- **MSB bit:** 0: BSR Command | 1 for I/O command
- Next 3 bits: Don't Care (keep 000)
- Next 3 bits: Select a line of Port C (PC0 ... PC7)
- LSB: 1: Set the line | 0: Reset the line

**Special Note:**

Do keep in mind, BSR Command is  
**APPLICABLE ONLY FOR PORT C**

**Special Note:**

If you are learning this by piracy, then  
you are not my student.  
You are simply a thief! #PoorUpbringing



## PROGRAM QUESTION BASED ON BSR COMMAND

Q: Using BSR, generate a square wave of 1 KHz on a device connected to PC3 of 8255

Solution: Assume 8255 is at 80H

Its various addresses will be as follows:

PA at address 80H | PB at address 81H | PC at address 82H | CW at address 83H

We will give our command at 83H to the control word.

0	X	X	X	BIT SELECT	S/R	
0	0	0	0	011	0	06H → Will send a 0 on PC3
0	0	0	0	011	1	07H → Will send a 1 on PC3

We will send 0 and 1 in a loop and include a delay after every output.

Since the square wave frequency is 1 KHz, Time period is 1 ms and hence delay will be 0.5 ms

We will use 8-bit delay method with a count of 106d (6AH) (This delay has been made before)

Program:

**MVI A, 80H**

**Out 83H, A** ; This makes all ports Output. It is optional to do this in this question

**Back: MVI A, 06H**

**Out 83H, A** ; Send 0 on PC3

**Call Delay** ; 0.5 ms, 8-bit method, count of 6AH

**MVI A, 07H**

**Out 83H, A** ; Send 1 on PC3

**Call Delay** ; 0.5 ms, 8-bit method, count of 6AH

**JMP BACK** ; loop

### Bharat Acharya Education

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes

Free: PDFs of Theory explanation, VIVA questions and MCQs

Start Learning... NOW

**[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)**

Official WhatsApp number:

**+91 9136428051**



## 8255 DATA TRANSFER MODES

### MODE 0: SIMPLE I/O

- Port A and Port B used as 2 Simple 8-bit I/O Ports.
- Port C is used as 2 simple 4-bit I/O Ports.
- Each port can be programmed as input or output individually.
- Ports do not have handshake or interrupting capability.
- Hence, **slower** devices cannot be interfaced.
- The only real advantage of mode 0 is that all three ports are available for date transfer

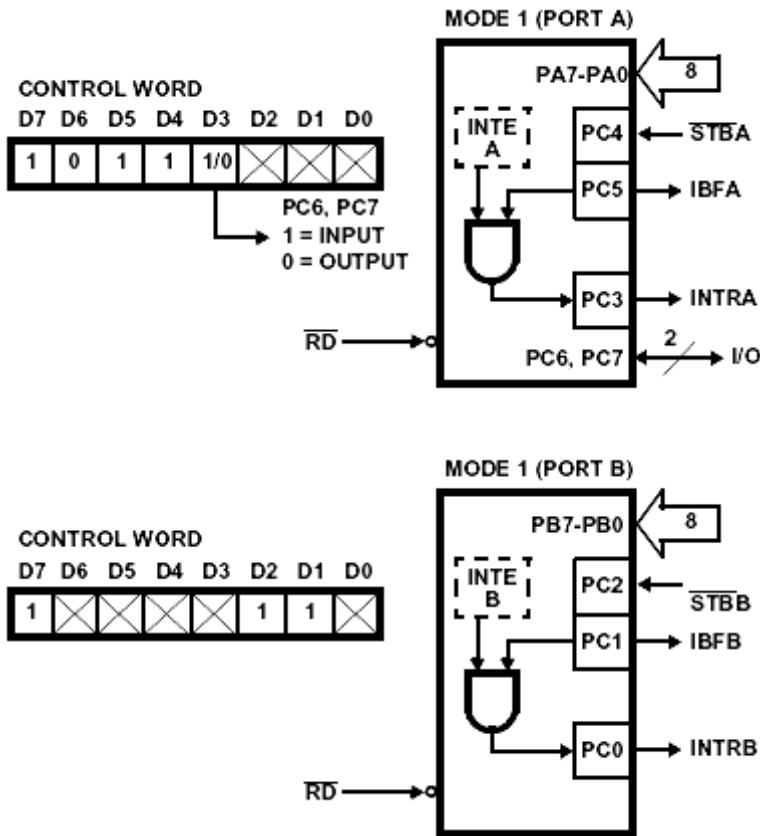
### MODE 1: HANDSHAKE I/O

- In Mode 1, handshake signals are exchanged between the devices before the data transfer takes place.
- Port A and Port B used as 2 8-bit I/O Ports that can programmed in Input OR in output mode.
- Each Port uses 3 lines from Port C for handshake. The remaining lines of Port C can be used for simple IO.
- **Interrupt driven** data transfer and **Status driven** data transfer possible.
- Hence, **slower** devices can be interfaced.
- The handshake signals are different for input and output modes.

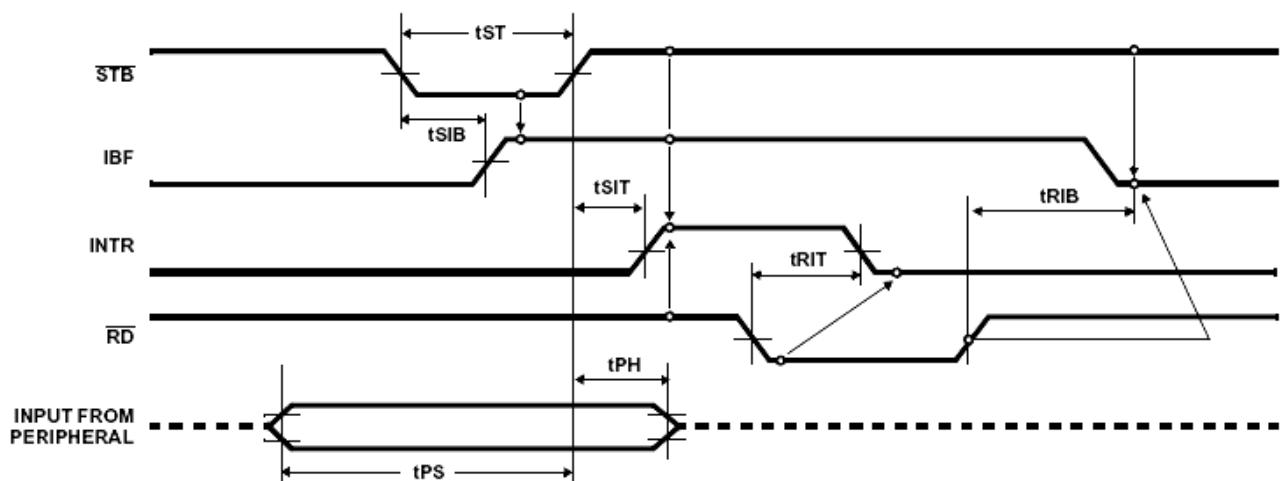
#Please refer Bharat Sir's video for this on [www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



## MODE 1 – INPUT HANDSHAKING



### Timing Diagram for Mode 1 Input Transfer





## Working:

- **Each port uses 3 lines of Port C** for the following signals:
- **STB** (Strobe), **IBF** (Input Buffer Full) → Handshake signals
- **INTR** (interrupt) → Interrupt signal
- Additionally, the **RD** signal of 8255 is also used.

**Handshaking** takes place in the following manner:

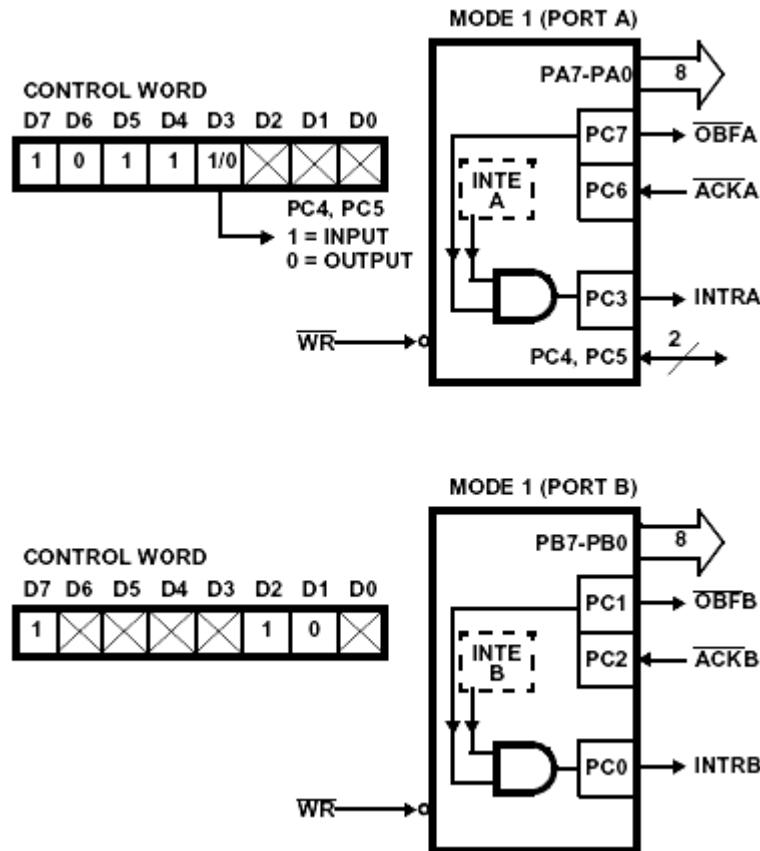
- The **peripheral** device **places data** on the Port **bus** and informs the Port by **making STB low**.
- The **input Port accepts** the **data** and informs the peripheral to wait by making **IBF high**.  
This **prevents** the peripheral from **sending more data** to the 8255 and **hence data loss** is prevented.  
😊 In case of doubts, connect with us on WhatsApp: +919136428051
- **8255 interrupts the μP** through the **INTR** line provided the **INTE flip-flop is set**.
- **In response** to the Interrupt, the **μP issues** the **RD** signal and **reads the data**.  
The **data byte is thus transferred** to the **μP**.
- Now, the **IBF signal goes low** and the peripheral can **send more data** in the above sequence.

**Special Note:**

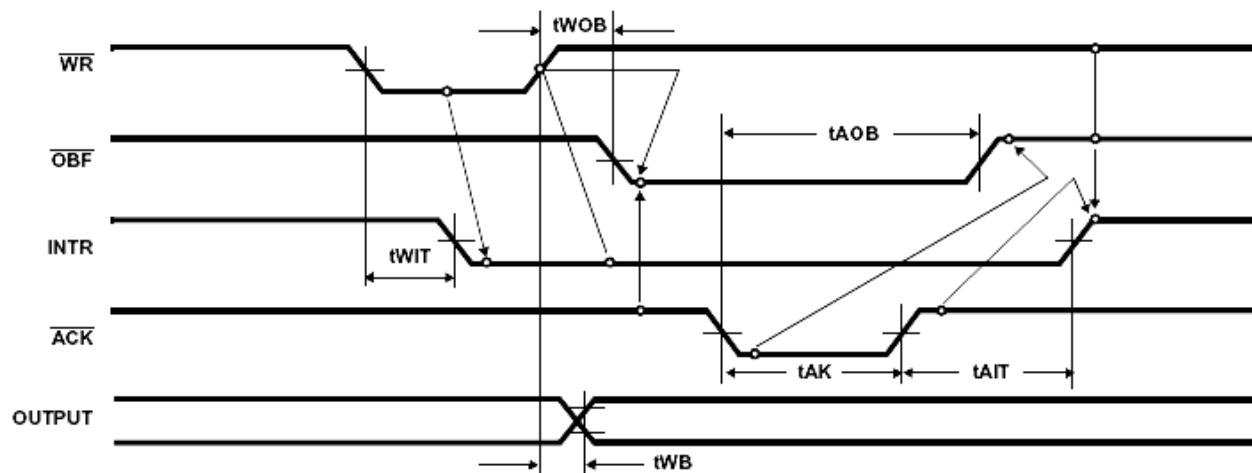
If you are learning this by piracy, then  
you are not my student.  
You are simply a thief! #PoorUpbringing



## MODE 1 – OUTPUT HANDSHAKING



### Timing Diagram for Mode 1 Output Transfer



## Working

- **Each port** uses **3 lines of Port C** for the following signals:
- **OBF** (Output Buffer Full), **ACK** (Acknowledgement) → Handshake signals
- **INTR** (interrupt) → Interrupt signal
- Additionally, the **WR** signal of 8255 is also used.

**Handshaking** takes place in the following manner:

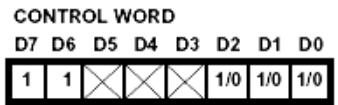
- When the output port is empty (indicated by a high on the INTR line), the **μP writes data** on the output port by giving the **WR** signal.
- As soon as the WR operation is complete, the **8255 makes the INTR low**, indicating that the **μP** should **wait**.  
This **prevents** the **μP** from **sending more data** to the 8255 and **hence data loss** is prevented.
- **8255 also makes the OBF low** to indicate to the output peripheral that **data is available** on the data bus.
- The **peripheral accepts the data** and sends an acknowledgement by making the **ACK low**.  
The **data byte is thus transferred** to the peripheral.
- Now, the **OBF and ACK lines go high**.
- The **INTR line becomes high** to **inform** the **μP** that **another byte** can be **sent**. i.e. the output port is empty.  
This process is repeated for further bytes.

**Special Note:**

If you are learning this by piracy, then  
you are not my student.  
You are simply a thief! #PoorUpbringing



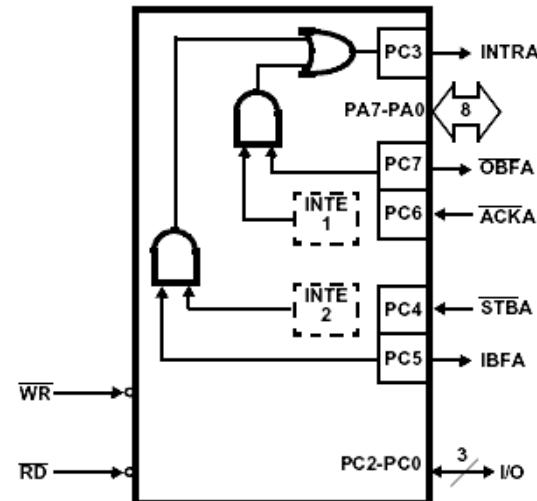
## MODE 2: BI-DIRECTIONAL HANDSHAKING



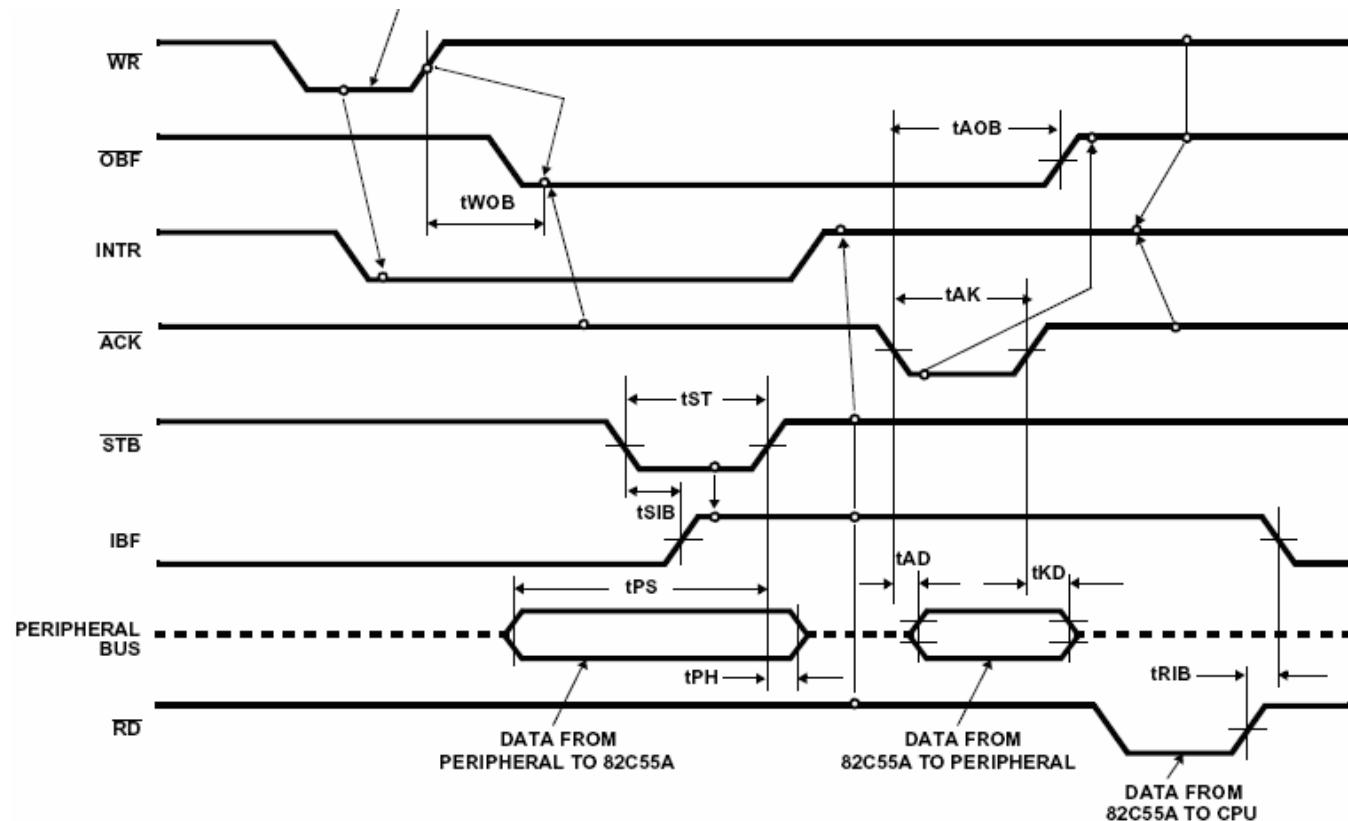
PC2-PC0  
1 = INPUT  
0 = OUTPUT

PORT B  
1 = INPUT  
0 = OUTPUT

GROUP B MODE  
0 = MODE 0  
1 = MODE 1



Timing Diagram for Mode 2 Bi-Directional Transfer





### **Bharat Acharya Education**

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes  
Free: PDFs of Theory explanation, VIVA questions and MCQs

Start Learning... NOW

**[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)**

Official WhatsApp number:

**+91 9136428051**



## 8254 PIT INTRODUCTION AND ARCHITECTURE

### INTRODUCTION – SALIENT FEATURES

- IC **8254** is used as a device to produce **Hardware delays**.
- It can also be used to generate a **real-time clock**, or as a **square wave generator etc.**
- Hardware delays are **more useful** than software delays because the **μP** is not actively involved in producing the delay. Thus when the delay is being produced the **μP** is free to execute its own program.
- The counting is done using **3 independent 16-bit down counters**.
- These counters can take the count in **BCD** or in **Binary**.
- After the Counter has finished counting and the required delay is produced, the 8254 interrupts the **μP**.

### BASIC STEPS FOR OPERATION

- Load the count. This is done in two steps. First the lower byte and then the higher byte.
- Apply clock pulses on CLK pin
- Apply Gate signal to enable counting
- Counting now starts...
- On every clock puls, the count will decrement
- When count reaches 0, that's called Terminal Count
- In single modes, an OUT signal is generated and counting stops
- In continuous modes, the out signal is generated. Original count is now automatically reloaded, and the counter starts all over again.

**Special Note:**

If you are learning this by piracy, then  
you are not my student.  
You are simply a thief! #PoorUpbringing

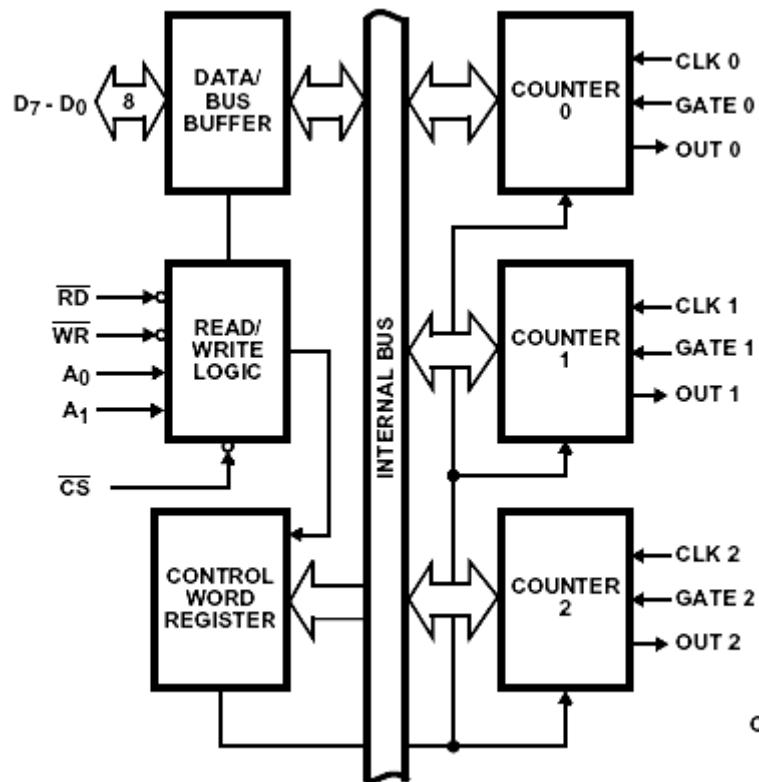
**Special Note:**

In continuous modes, the frequency generated at  
the output follows the rule:

$$\text{Output Frequency} = \text{Input frequency} \div \text{Count}$$



## ARCHITECTURE OF 8253/54



The architecture of 8254 can be divided into the following parts:

### 1) Data Bus Buffer

- This buffer is used to **interface** the internal **data bus** of the timer with the external (system) data bus.
- It is thus connected to D<sub>7</sub> - D<sub>0</sub> form the μP.



## 2) Read Write Logic

- It accepts the **RD** and **WR** signals, which are used to **control** the flow of **data** through the data bus.
- It also accepts the **A<sub>1</sub> – A<sub>0</sub>** address lines which are used to **select** one of the **Counters** or the **Control Word** as shown below:

<b>A<sub>1</sub> A<sub>0</sub></b>	<b>Selection</b>
0 0	Counter 0
0 1	Counter 1
1 0	Counter 2
1 1	Control Word

- It also accepts the **CS** signal to **select** the **8254** chip.

## 3) Control Word Register

<b>SC<sub>1</sub></b>	<b>SC<sub>0</sub></b>	<b>RW<sub>1</sub></b>	<b>RW<sub>0</sub></b>	<b>M<sub>2</sub></b>	<b>M<sub>1</sub></b>	<b>M<sub>0</sub></b>	<b>BCD</b>
-----------------------	-----------------------	-----------------------	-----------------------	----------------------	----------------------	----------------------	------------

<b>SC<sub>1</sub> SC<sub>0</sub></b>	<b>Selection</b>
0 0	Select Counter 0
0 1	Select Counter 1
1 0	Select Counter 2
1 1	READ BACK COMMAND (Only for 8254; illegal for

<b>RW<sub>1</sub> RW<sub>0</sub></b>	<b>Selection</b>
0 0	Counter Latch Command
0 1	Read/Write LSB Only
1 0	Read/Write MSB Only
1 1	Read/Write LSB First and then MSB

<b>M<sub>2</sub> M<sub>1</sub> M<sub>0</sub></b>	<b>Mode Selection</b>
0 0 0	Mode 0 --- Interrupt On Terminal Count
0 0 1	Mode 1 --- Monostable Multivibrator
X 1 0	Mode 2 --- Rate Generator
X 1 1	Mode 3 --- Square Wave Generator
1 0 0	Mode 4 --- Software Triggered Strobe
1 0 1	Mode 5 --- Hardware Triggered Strobe

<b>BCD</b>	<b>Type of Count</b>
0	Binary Counter
1	BCD Counter

- The Control Word Register is an **8-bit** register that holds the Control Word as shown above.
- It is selected when **A<sub>1</sub> – A<sub>0</sub>** contain 11.
- It has a different format when a Read Back command is given for 8254, as shown below



## Read Operations

There are 3 ways in which the  $\mu$ P can read the current count:

A) Ordinary Read

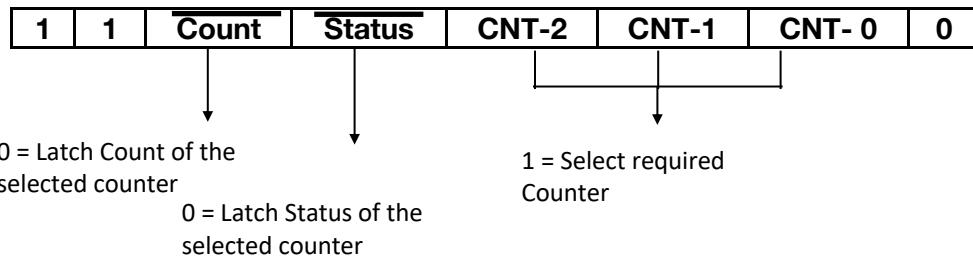
- In this method, the **counting is stopped** by controlling the gate input of a selected counter.
- The Counter is then selected by  $A_1 - A_0$  and IO Read operation is performed.
- First **IO Read** will give the Lower byte of the Count value, and the second IO Read will give the higher byte.
- The **disadvantage** here is that **counting is disturbed/stopped**.

B) Read on Fly

- In this method, the  $\mu$ P reads the **count value while the counting is still in progress**.
- Thus, it is called as **Read On Fly**.
- The appropriate value is written in the control word, and IO Read operation is performed.
- The current value of the **Count** is "**latched**" internally and returned to the  $\mu$ P.
- The **advantage** here is that counting is **not disturbed**.

C) Read Back Command

### Control Word Register for Read Back Command



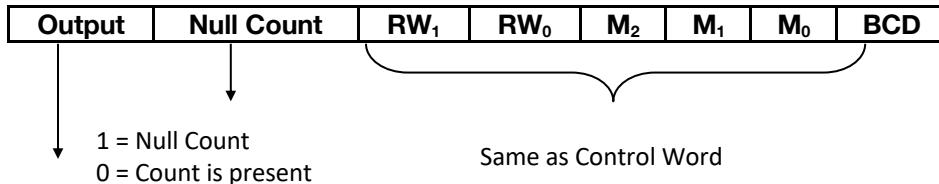
- The Read Back Command is available **only** for **8254** and not for 8253.
- The Read Back Command **reads the Count** value in the **same manner** as **Read On Fly**.
- In **addition** to the Count, the current **Status** can **also** be **latched** using the Read Back command.
- Thus, the appropriate value (for latching the Count and/or Status of the selected counter) is placed in the Control word as shown above.
- The **advantage** here is that the **Count** and the **Status both** can be read **without disturbing the counting**.

**Special Note:**

If you are learning this by piracy, then  
you are not my student.  
You are simply a thief! #PoorUpbringing



**Status Word** (Status returned after the Read Back Command)



1 = OUT pin is high

0 = OUT pin is low

#### 4) 3 Independent Counters

- 8254 has **3 Independent, 16-bit down counters.**
- Each counter can operate have a **Binary** or **BCD** count.
- Each counter can be in **one of the six possible modes.**
- Each counter can have a **max count of  $2^{16} = 65535$  i.e. FFFFH.**
- Each counter has the following signals:
  - i. **Clk (Clock Input)**
  - ii. **Gate(Gate Input)**
  - iii. **Out (Clock Output)**
- The **input** clock signal is applied on the **CLK** line.
- The **counter decrements** the "count value" on **every pulse of the input clock at CLK**.
- **When the count becomes zero** (Terminal Count i.e. TC), the **status** of the **OUT** pin **changes**. This can be used to interrupt the  $\mu$ P.
- The **GATE** pin is used to **control** the **Counting**. In most modes, the **count value gets decremented only if the GATE pin is high**.

### Bharat Acharya Education

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes

Free: PDFs of Theory explanation, VIVA questions and MCQs

Start Learning... NOW

**[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)**

Official WhatsApp number:

**+91 9136428051**



# 8254 PIT COMMAND AND PROGRAMMING

## COMMANDS

### Control Word Register

SC <sub>1</sub>	SC <sub>0</sub>	RW <sub>1</sub>	RW <sub>0</sub>	M <sub>2</sub>	M <sub>1</sub>	M <sub>0</sub>	BCD
-----------------	-----------------	-----------------	-----------------	----------------	----------------	----------------	-----

SC <sub>1</sub> SC <sub>0</sub>	Selection
0 0	Select Counter 0
0 1	Select Counter 1
1 0	Select Counter 2
1 1	READ BACK COMMAND (Only for 8254; illegal for

RW <sub>1</sub> RW <sub>0</sub>	Selection
0 0	Counter Latch Command
0 1	Read/Write LSB Only
1 0	Read/Write MSB Only
1 1	Read/Write LSB First and then MSB

M <sub>2</sub> M <sub>1</sub> M <sub>0</sub>	Mode Selection
0 0 0	Mode 0 --- Interrupt On Terminal Count
0 0 1	Mode 1 --- Monostable Multivibrator
X 1 0	Mode 2 --- Rate Generator
X 1 1	Mode 3 --- Square Wave Generator
1 0 0	Mode 4 --- Software Triggered Strobe
1 0 1	Mode 5 --- Hardware Triggered Strobe

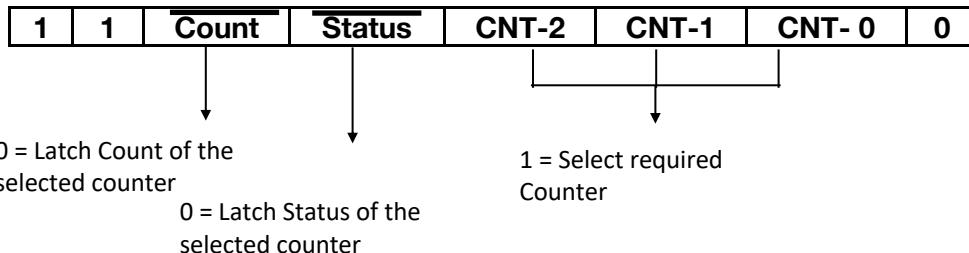
BCD	Type of Count
0	Binary Counter
1	BCD Counter

- The Control Word Register is an **8-bit** register that holds the Control Word as shown above.
- It is selected when A<sub>1</sub> – A<sub>0</sub> contain 11.
- It has a different format when a Read Back command is given for 8254, as shown below



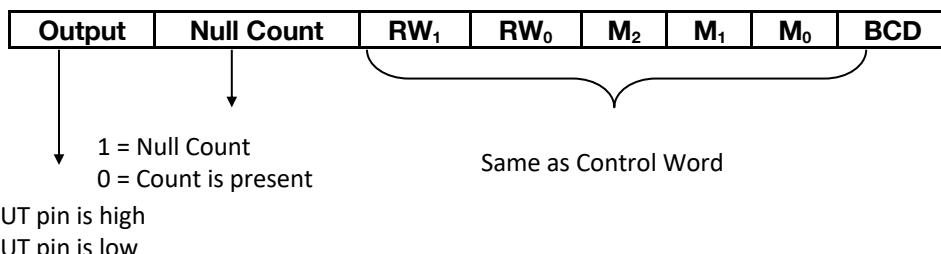
### Read Back Command

#### Control Word Register for Read Back Command



- The Read Back Command is available **only** for **8254** and not for 8253.
- The Read Back Command **reads** the **Count** value in the **same manner** as **Read On Fly**.
- In **addition** to the Count, the current **Status** can **also** be **latched** using the Read Back command.
- Thus, the appropriate value (for latching the Count and/or Status of the selected counter) is placed in the Control word as shown above.
- The **advantage** here is that the **Count** and the **Status both** can be read **without disturbing the counting**.

#### Status Word (Status returned after the Read Back Command)



1 = OUT pin is high  
0 = OUT pin is low

#### Special Note:

If you are learning this by piracy, then  
you are not my student.  
You are simply a thief! #PoorUpbringing



## PROGRAMMING

**Q:** Write a program to generate a continuous Square Wave of 1 KHz on Port 80H using an input frequency of 16.525 MHz to 8254.

**Solution:**

Assume 8254 is at address 80h

Then its various addresses will be as follows...

Counter 0 at 80h

Counter 1 at 81h

Counter 2 at 82h

Control Word at 83h

Hence, we will give our count at 80h for counter 0 and command at 83h

Input frequency = 16.525 MHz

Output frequency = 1 KHz

Counter = Counter 0

Mode = Mode 3 for Square wave (refer previous page for list of modes)

Output Frequency = input frequency ÷ count

Hence count = 16.525 MHz ÷ 1 KHz = 16525<sub>dec</sub> = 408D<sub>hex</sub>

Program:

**MVI A, 36h** ; Counter = Counter 0, RW = 11, Mode = 010, BCD = 0  
**OUT 83H** ; Give Command to Control Word at 83h

**MVI A, 8Dh** ; Lower byte of count 408Dh  
**OUT 80H** ; Send to Counter 0 at 80h

**MVI A, 40h** ; Higher byte of count 408Dh  
**OUT 80H** ; Send to Counter 0 at 80h

**Special Note:**

If you are learning this by piracy, then  
you are not my student.

You are simply a thief! #PoorUpbringing



**Q: Write a program to operate Counter 1 as a Monostable Multivibrator with a count of 1845H**

**Solution:**

Assume 8254 is at address 80h

Then its various addresses will be as follows...

Counter 0 at 80h

Counter 1 at 81h

Counter 2 at 82h

Control Word at 83h

Hence, we will give our count at 81h for counter 1 and command at 83h

Count = 1845H (already given in the question)

Counter = Counter 1

Mode = Mode 1 for monostable Multivibrator (refer previous page for list of modes)

Program:

**MVI A, 72h ; Counter = Counter 1, RW = 11, Mode = 001, BCD = 0**

**OUT 83H ; Give Command to Control Word at 83h**

**MVI A, 45h ; Lower byte of count 1845h**

**OUT 81H ; Send to Counter 1 at 81h**

**MVI A, 18h ; Higher byte of count 1845h**

**OUT 81H ; Send to Counter 1 at 81h**

**Special Note:**

If you are learning this by piracy, then  
you are not my student.

You are simply a thief! #PoorUpbringing

**Bharat Acharya Education**

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes

Free: PDFs of Theory explanation, VIVA questions and MCQs

Start Learning... NOW

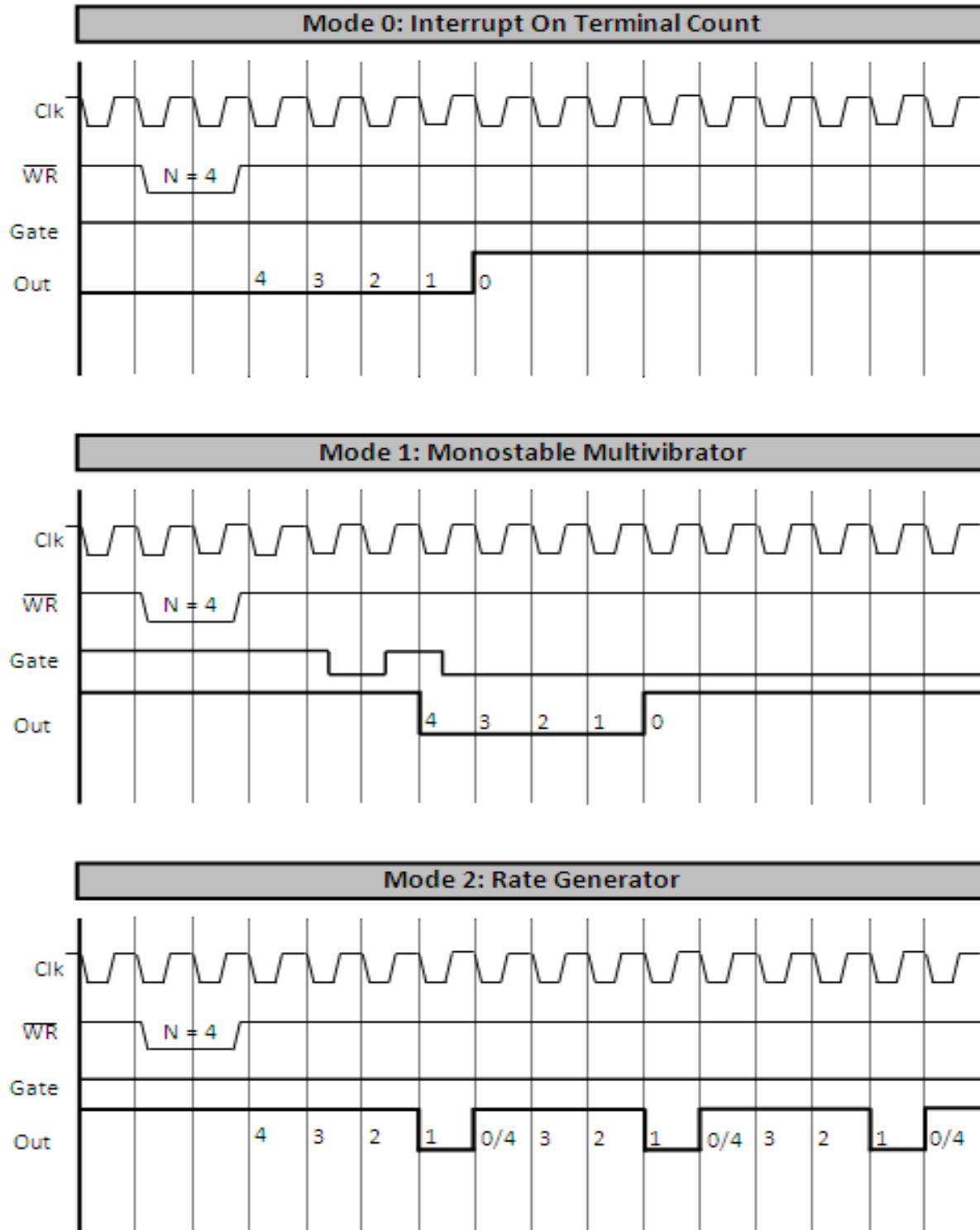
**[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)**

Official WhatsApp number:

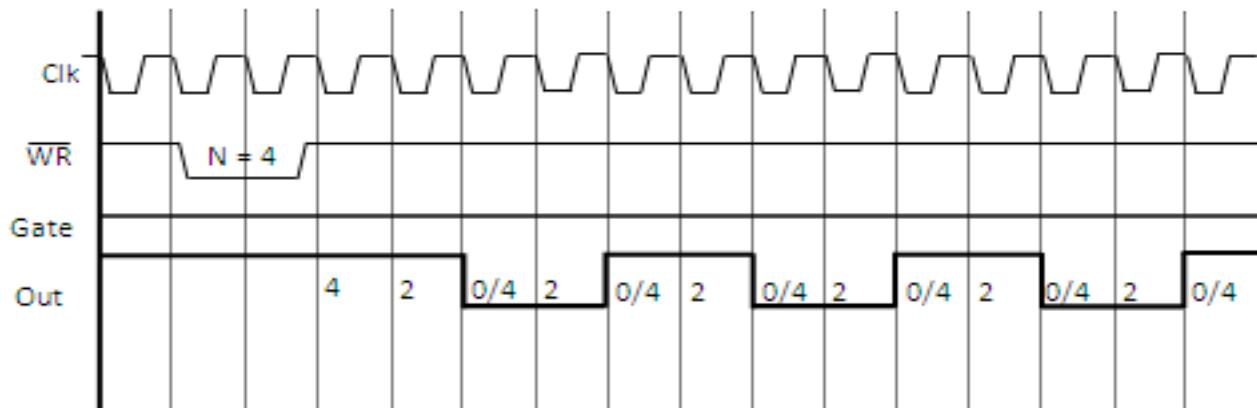
**+91 9136428051**



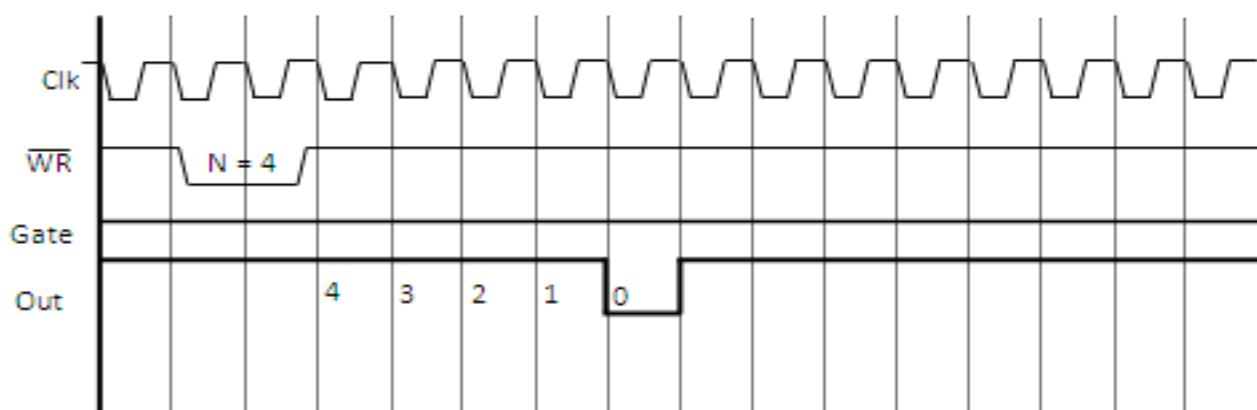
## 8254 PIT TIMER MODES



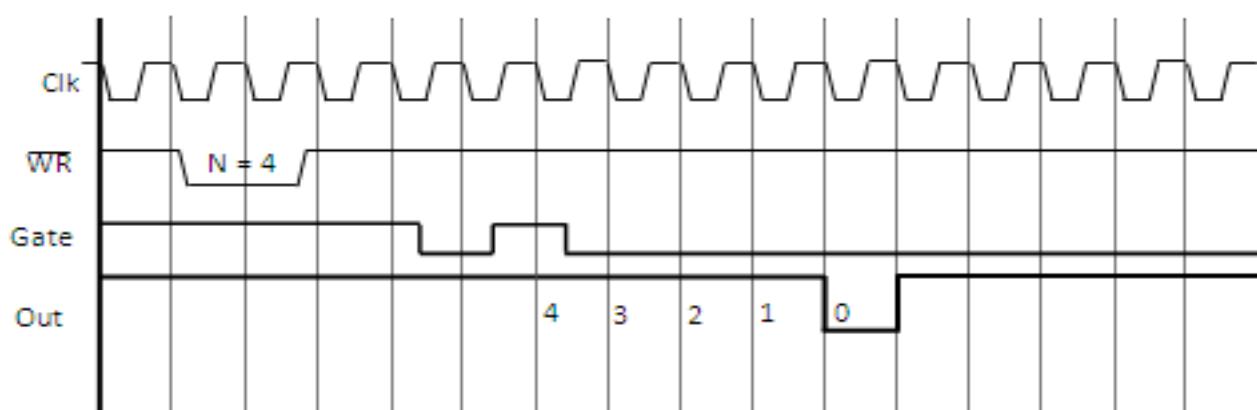
### Mode 3: Square Wave Generator



#### Mode 4: Software Triggered Strobe



## Mode 5: Hardware Triggered Strobe





## MODE 0: INTERRUPT ON TERMINAL COUNT

- When this mode is selected **OUT** pin is **initially low**.
- The **count** value is **loaded**.
- **GATE** pin is made **high**, so **counting** is **enabled**.
- During **counting**, **OUT** pin remains **low**.
- On Terminal Count (TC) the **OUT** pin goes **high** and remains high.
- During **counting** if **GATE** is made **low**, it **disables counting**.  
When **GATE** is made **high**, counting **Resumes**.
- **Effect of Gate:**  
Low → Disables Counting  
High → Enables (Resumes) Counting

**BASIC APPLICATION:** To produce a delay.

## MODE 1: MONOSTABLE MULTIVIBRATOR

- When this mode is selected **OUT** pin is **initially high**.
- The **count** value is **loaded**.
- **Counting begins ONLY when a rising edge** is applied to the **GATE**.
- **OUT** pin goes **low** and remains low **during counting**.
- On Terminal Count (TC) the **OUT** pin goes **high** and remains high.
- During **counting** if **GATE** is made **low**, it **has no effect** on the Counting.
- The **GATE** pin can be used as a **Trigger**. ☺ In case of doubts, WhatsApp Bharat Sir on 91364 28051.  
The Counter can be **re-triggered** by applying a **rising edge** on the **GATE**.  
This would **Restart** the **counting**, and hence re-trigger it.
- **Effect of Gate:**  
Low → No Effect  
High (Rising Edge Trigger) → Starts Counting, can also re-trigger it.

**BASIC APPLICATION:** To re-trigger counting during the delay like timer IC 555.

### Special Note:

If you are learning this by piracy, then you are not my student.  
You are simply a thief! #PoorUpbringing



## MODE 2: RATE GENERATOR

- When this mode is selected **OUT** pin is **initially high**.
- The **count** value is **loaded**.
- **GATE** pin is made **high**, so **counting** is **enabled**.
- **During counting**, **OUT** pin remains **high**.
- The **OUT** pin goes low for one clock cycle just before the TC.
- The initial count is reloaded and the above process repeats.  
Thus, this mode produces a Continuous Pulse.
- **During counting if GATE is made low, it disables counting.**  
When **GATE** is made **high**, counting **Restarts**.
- **Effect of Gate:**  
Low → Disables Counting  
High → Enables (Restarts) Counting
- It is also called a divide by n counter

**BASIC APPLICATION:** To generate a desired frequency (output type – rectangular wave).

## MODE 3: SQUARE WAVE GENERATOR

- When this mode is selected **OUT** pin is **initially high**.
- The **count** value is **loaded**.
- **GATE** pin is made **high**, so **counting** is **enabled**.
- **OUT** pin remains **high** for **half of the count** ( $n/2$ ) and remains **low** for the **remaining half**.
- **On TC, the Count is reloaded** and the **process repeats** itself producing a **continuous square wave**.
- **During counting if GATE is made low, it disables counting.**  
When **GATE** is made **high**, counting **Restarts**.
- **Effect of Gate:**  
Low → Disables Counting  
High → Enables (Restarts) Counting
- If the **count** is **ODD**, the **OUT** pin remains **high** for  $(n+1)/2$  and **low** for  $(n-1)/2$ .

**Special Note:**

If you are learning this by piracy, then you are not my student.  
You are simply a thief! #PoorUpbringing

**BASIC APPLICATION:** To generate a desired frequency (output type – square wave).



## MODE 4: SOFTWARE TRIGGERED STROBE

- When this mode is selected **OUT** pin is **initially high**.
- The **count** value is **loaded**.
- **GATE** pin is made **high**, so **counting** is **enabled**.
- **During counting**, **OUT** pin remains **high**.
- The **OUT** pin goes **low** for **one clock cycle**, just after **TC**.
- **After that** **OUT** pin goes **high** and remains **high**.
- **During counting** if **GATE** is made **low**, it **disables counting**.  
When **GATE** is made **high**, counting **Restarts**.
- **Effect of Gate:**  
Low → Disables Counting  
High → Enables (Restarts) Counting

**BASIC APPLICATION:** To generate a delay (output type – strobed).

## MODE 5: HARDWARE TRIGGERED STROBE

- When this mode is selected **OUT** pin is **initially high**.
- The **count** value is **loaded**.
- Counting starts **ONLY** after a trigger is applied to the **GATE** pin.
- Also, the **GATE** pin need not remain **high** for the counting to continue.
- **During counting**, **OUT** pin remains **high**.
- The **OUT** pin goes **low** for **one clock cycle**, just after **TC**.
- **After that** **OUT** pin goes **high** and remains **high**.
- **Thus GATE is used as a Trigger.**  
I.e. It has to be triggered to start counting.
- **Effect of Gate:**  
Low → No Effect on counting  
High (Trigger) → Starts Counting, can also re-trigger it.

**BASIC APPLICATION:** To re-trigger counting during the delay (output type – strobed).

### Bharat Acharya Education

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes  
Free: PDFs of Theory explanation, VIVA questions and MCQs

Start Learning... NOW [www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

Official WhatsApp number: **+91 9136428051**

## 8155 PROGRAMMABLE I/O AND TIMER

### SALIENT FEATURES OF 8155

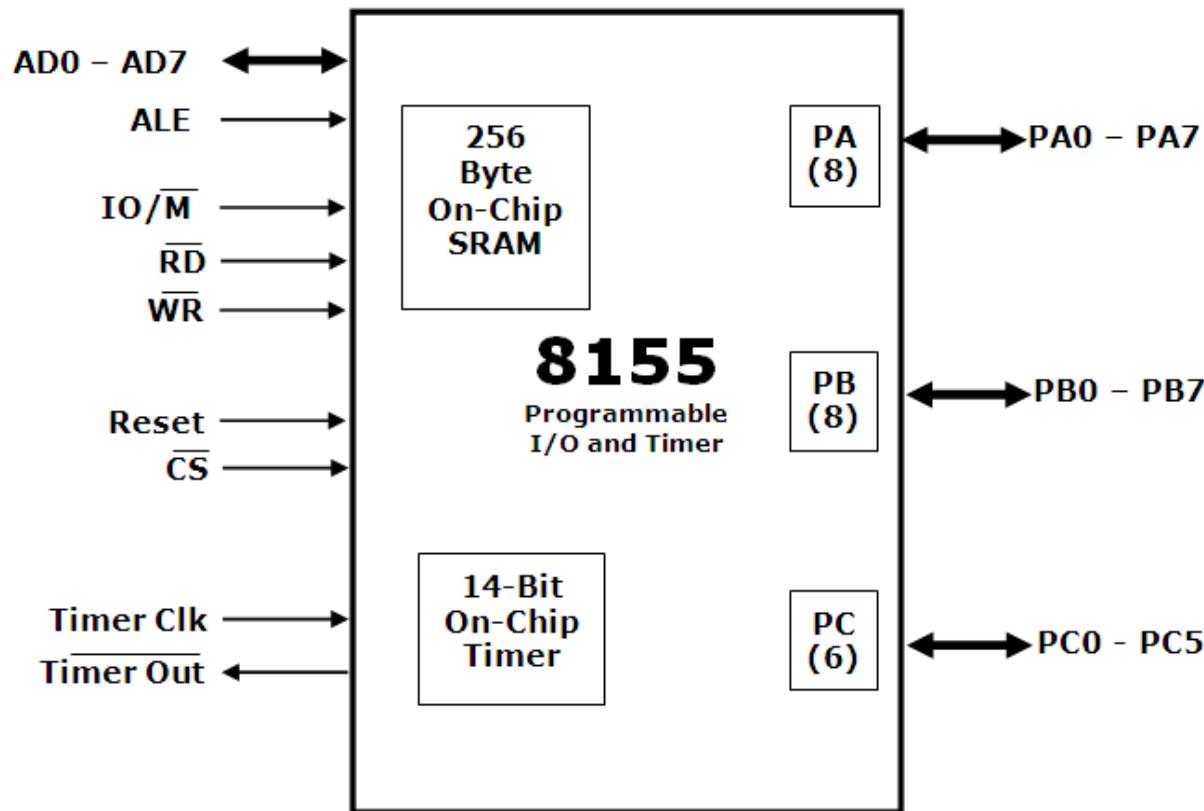
- IC 8155 is a Programmable I/O device and a Timer.
- It has **3 bi-directional I/O ports** - Port A, Port B, and Port C.  
**Port A and Port B** are **8-bit**, **Port C** is **6-bit**.
- It provides **2 modes** of **data transfer** – **Simple I/O, Handshake I/O**.  
Bi-directional Handshake I/O is **NOT provided**.
- It has a **14-bit Down Counter**, which can work as a Timer/Counter.  
The Timer/Counter has **4 modes** of Operation.
- It has **256 Bytes** ( $256 \times 8$ ) **on-chip RAM**.
- It has **on-chip de-multiplexing logic** for separating the address and the data from the multiplexed address-data bus.  
Thus **AD<sub>7</sub>-AD<sub>0</sub>** from 8085 are **directly interfaced** with 8155 and not D<sub>7</sub>-D<sub>0</sub> separately.  
**ALE** signal is **required** for this de-multiplexing.
- Also, **IO/M, RD, WR** are **directly interfaced** with 8155 and instead of IOR, IOW, MEMR, MEMW.
- **8155** alone satisfies the requirement of **I/O Ports, Timer, RAM**.
- Moreover, it doesn't need external **latch** for address data demultiplexing.
- It doesn't even need a **decoder** for IO/M, RD, WR decoding.
- This makes the circuit **compact** with minimum components.
- Hence 8155 is used in a circuit called **MINIMUM SYSTEM of 8085**.

#### Important Tip:

If you have already studied 8255 and 8254 you should find 8155 fairly simple to understand.

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

## BLOCK DIAGRAM OF 8155



**Special Note:**

If you are learning this by piracy, then you are not my student.  
You are simply a thief! #PoorUpbringing

### 1) Memory Section

- 8155 has a **256 byte on-chip RAM**.
- It can be used to store data or address.  
i.e. as Data Memory or Stack Memory in the minimum system.
- The method of using this memory is described later.

#Please refer Bharat Sir's video at [www.bharatacharyaeducation.com](http://www.bharatacharyaeducation.com) for full explanation

### 2) Port A, Port B, Port C

- **Port A and Port B are 8-bit, Port C is 6-bit.**
- Port A and Port B can work in Mode 0 and Mode 1, Port C only in Mode 0.
- In Mode 1, Port C provides handshake signals for Port A and Port B.

Port	Mode 0	Mode 1
Port A	✓	✓
Port B	✓	✓
Port C	✓	✗ (Handshake signals)

### 3) Interfacing Pins

#### ◆ AD<sub>7</sub>-AD<sub>0</sub>

This is the multiplexed address-data bus.

8155 has internal logic for de-multiplexing this bus to get D<sub>7</sub>-D<sub>0</sub> and A<sub>7</sub>-A<sub>0</sub>.  
A<sub>2</sub>-A<sub>0</sub> are used to make the following selection.

A <sub>2</sub> A <sub>1</sub> A <sub>0</sub>	Selection	Sample address
0 0 0	Control Register	80 H (i.e. 1000 0000)
0 0 1	Port A	81 H (i.e. 1000 0001)
0 1 0	Port B	82 H (i.e. 1000 0010)
0 1 1	Port C	83 H (i.e. 1000 0011)
1 0 0	Timer LSB	84 H (i.e. 1000 0100)
1 0 1	Timer MSB	85 H (i.e. 1000 0101)

#### ◆ ALE

ALE is used internally for de-multiplexing the multiplexed address-data bus.

#### ◆ IO/M, RD, WR

These signals are decoded internally to perform IO/Memory Read/Write operations.

#### ◆ Reset

This signal is used to reset the 8155.

## **Bharat Acharya Education**

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes  
Free: PDFs of Theory explanation, VIVA questions and MCQs

Start Learning... NOW!

**[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)**

Official WhatsApp number:

**+91 9136428051**

## 8155 PROGRAMMABLE I/O AND TIMER

### 8155 TIMER SECTION

- The Timer section of 8155 has two 8-bit registers: Timer LSB and Timer MSB.
- Collectively these registers hold the 14 Bits of the Count value + 2 bits for mode selection.
- 8155 has a 14 bit down counter.
- It functions as follows:
  - Clock input to the timer is provided by the "Timer Clk".
  - Mode is selected by the Timer MSB (M1 and M0).
  - Timer MSB and LSB hold the 14-bit count.
  - On every input pulse, the 14-bit Count value is decremented (down counter).
  - When this Count value reaches 0, it is called as "Terminal Count" (TC).
  - The output of the Timer is given by the "Timer Out" pin.
  - This output varies with different modes as shown.

M <sub>1</sub>	M <sub>0</sub>	T <sub>13</sub>	T <sub>12</sub>	T <sub>11</sub>	T <sub>10</sub>	T <sub>9</sub>	T <sub>8</sub>	<b>Timer MSB</b>
----------------	----------------	-----------------	-----------------	-----------------	-----------------	----------------	----------------	------------------

T <sub>7</sub>	T <sub>6</sub>	T <sub>5</sub>	T <sub>4</sub>	T <sub>3</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>0</sub>	<b>Timer LSB</b>
----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	------------------

M <sub>1</sub> M <sub>0</sub>	Timer Mode
0 0	Mode 0
0 1	Mode 1
1 0	Mode 2
1 1	Mode 3

#### Mode 0

- In this mode, 8155 generates a **Single Square Wave**.
- The **Timer Out** pin remains **high** for **half** the **count**, and **low** for the remaining **half**.
- On TC**, the **Timer stops**.

#### Mode 1

- In this mode, 8155 generates a **Continuous Square Wave**.
- The **Timer Out** pin remains **high** for **half** the count, and **low** for the remaining **half**.
- On TC**, the **Timer gets reloaded** with the original values, and the **process repeats** thus giving a **continuous** square wave.

## Mode 2

- In this mode, 8155 generates a **Single Active Low Pulse**.
- The **Timer Out** pin remains **high throughout the counting process**.
- **On Terminal Count**, the **Timer Out** pin goes **low for one pulse**.  
Thereafter it **stops**.

**Special Note:**

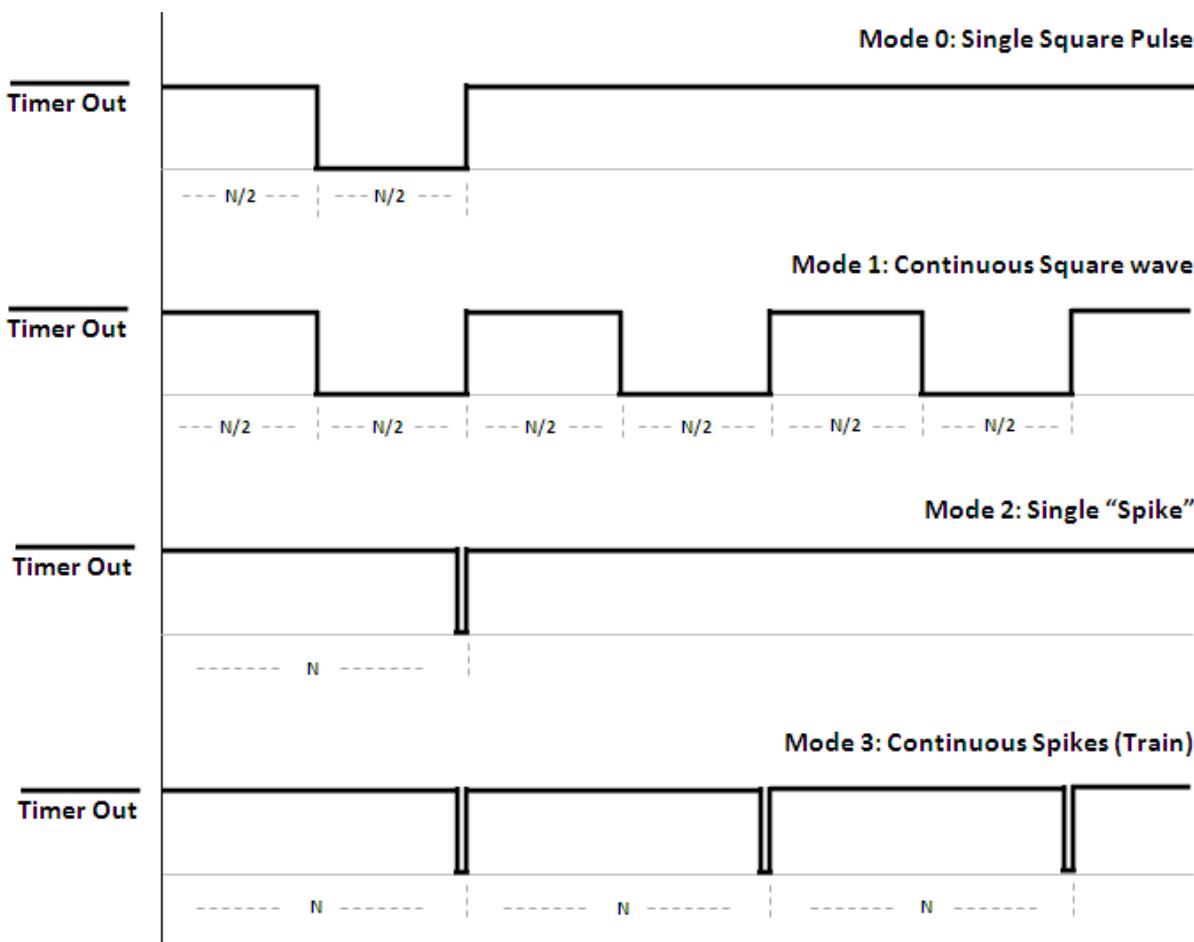
If you are learning this by piracy, then you are not my student.

You are simply a thief!  
#PoorUpbringing

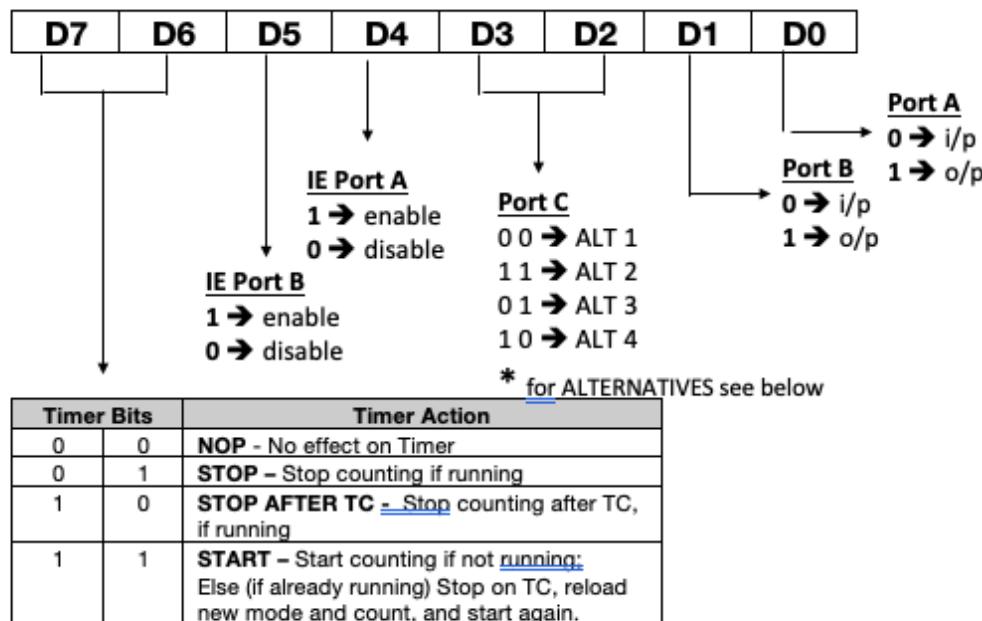
## Mode 3

- In this mode, 8155 generates **Continuous Active Low Pulses ("Train")**.
- The **Timer Out** pin remains **high throughout the counting process**.
- **On Terminal Count**, the **Timer Out** pin goes **low for one pulse**.  
Thereafter the **Timer is reloaded** and the process **repeats** giving **continuous active low pulses**.

#Please refer Bharat Sir's videos for this ...



## 8155 CONTROL WORD



**Port C Alternative functions**

ALT	PC <sub>5</sub>	PC <sub>4</sub>	PC <sub>3</sub>	PC <sub>2</sub>	PC <sub>1</sub>	PC <sub>0</sub>
<b>ALT 1</b>	i/p	i/p	i/p	i/p	i/p	i/p
<b>ALT 2</b>	o/p	o/p	o/p	o/p	o/p	o/p
<b>ALT 3</b>	o/p	o/p	o/p	STB <sub>A</sub>	BF <sub>A</sub>	INTR <sub>A</sub>
<b>ALT 4</b>	STB <sub>B</sub>	BF <sub>B</sub>	INTR <sub>B</sub>	STB <sub>A</sub>	BF <sub>A</sub>	INTR <sub>A</sub>

**Special Note:**

- ALT1 means: Port C is an input port in Mode 0. Port A and port B are also in Mode 0.
- ALT2 means: Port C is an output port in Mode 0. Port A and port B are also in Mode 0.
- ALT3 means: Port C upper output. Port A is in Mode 1. Port B is in Mode 0.
- ALT4 means: Port C fully used for handshaking. Port A and port B are in Mode 1.

## PROGRAM FOR TIMER SECTION

Q: Generate a continuous square wave of 1 KHz using an input frequency of 100 KHz

Solution:

Assume 8155 is at the address 20H.

Then its various addresses will be as follows...

Control Word:	20H	<i>{Will be used to START the Timer}</i>
Port A:	21H	
Port B:	22H	
Port C:	23H	
Timer LSB:	24H	<i>{Will be used to give lower byte of count}</i>
Timer MSB:	25H	<i>{Will be used to give higher byte of count and mode}</i>

Input frequency = 100 KHz

Output frequency = 1 KHz

Count = input frequency ÷ output frequency

Count =  $100_d = 0064_{16}$

Timer LSB = 0110 0100 = 64H

Timer MSB = 0100 0000 = 40H (We select mode 1 for square wave)

Control Word = 1100 0000 = C0H (Start the timer)

Program:

**MVI A, 64H**

**OUT 24H; ... Timer LSB**

**MVI A, 40H**

**OUT 25H; ... Timer MSB**

**MVI A, C0H**

**OUT 20H; ... Control Word**

### Special Note:

If you are learning this by piracy, then you are not my student.  
You are simply a thief!  
#PoorUpbringing

## **Bharat Acharya Education**

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes  
Free: PDFs of Theory explanation, VIVA questions and MCQs

Start Learning... NOW!

**[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)**

Official WhatsApp number:

**+91 9136428051**

## 8155 PROGRAMMABLE I/O AND TIMER

### 8155 DATA TRANSFER MODES

- 8155 has 3 I/O Ports
- Port A and Port B are 8-bit ports
- Port C is a 6-bit port
- There are 2 modes of data transfer, Mode 0 and Mode 1
- All 3 ports can operate in mode 0
- Only Port A and Port B can do data transfers in Mode 1, Port C lines are used for Handshaking.

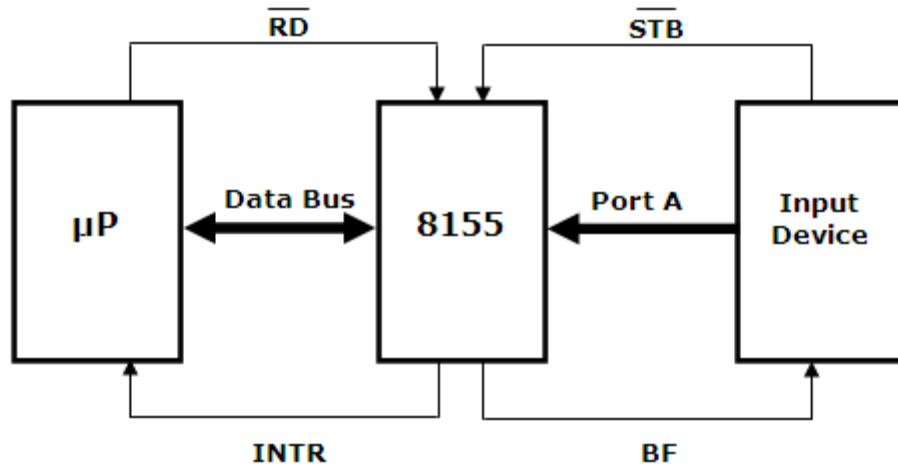
### MODE 0: SIMPLE I/O

- **Port A and Port B** used as **2 Simple 8-bit I/O Ports**.
- Port C is used as a Simple 6-bit I/O Port.
- Each port can be programmed as input or output individually.
- Ports do not have handshake or interrupting capability.
- Hence, **slower** devices **cannot** be interfaced.

### MODE 1: HANDSHAKE I/O

- In Mode 1, handshake signals are exchanged between the devices before the data transfer takes place.
- Port A and Port B used as 2 8-bit I/O Ports that can be programmed in Input OR in output mode.
- Each Port uses 3 lines from Port C for handshake.
- **Interrupt driven** data transfer and **Status driven** data transfer possible.
- Hence, **slower** devices **can be interfaced**.
- The handshake signals for input and output modes are as follows.

## MODE 1: INPUT HANDSHAKING



Timing diagram same as 8255

- **Each port uses 3 lines of Port C** for the following signals:
- **STB** (Strobe), **BF** (Buffer Full) → Handshake signals
- **INTR** (interrupt) → Interrupt signal
- Additionally the **RD** signal of 8155 is also used.
- **Handshaking** takes place in the following manner:

Step 1) The **input device places data** on the Port **bus** and informs the Port by **making STB low**.

Step 2) 8155 **accepts the data** and informs the peripheral to wait by **making BF high**.

This **prevents** the peripheral from **sending more data** to the 8155 and **hence data loss** is prevented. ☺ In case of doubts, contact Bharat Sir: - 98204 08217.

Step 3) **8155 interrupts the μP** through the **INTR** line provided the INTE flip-flop is set.

Step 4) **In response** to the Interrupt, the **μP issues the RD signal and reads the data**.

The **data byte is thus transferred** to the **μP**.

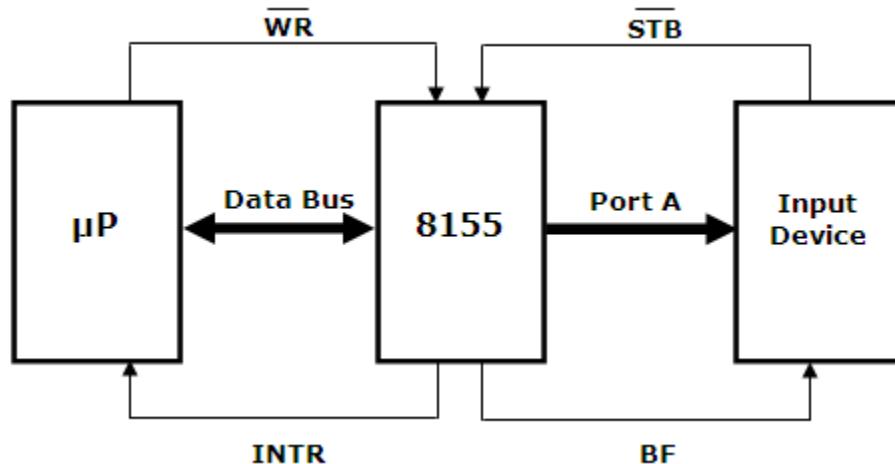
Step 5) Now, the **BF signal goes low** and the peripheral can **send more data** in the above sequence.

### Special Note:

If you are learning this by piracy, then you are not my student.  
You are simply a thief! #PoorUpbringing

## MODE 1: OUTPUT HANDSHAKING

Step 6)



Timing Diagram same as 8255

- Each port uses 3 lines of Port C for the following signals:  
**BF** (Buffer Full), **STB**(Strobe) → Handshake signals  
**INTR** (interrupt) → Interrupt signal
- Additionally the **WR** signal of 8155 is also used.

**Handshaking** takes place in the following manner:

- 1) When the output port is empty (indicated by a high on the INTR line), the **μP writes data** on the output port by giving the **WR** signal.
- 2) As soon as the WR operation is complete, the **8155 makes the INTR low**, indicating that the **μP** should **wait**. This **prevents** the **μP** from **sending more data** to the 8155 and **hence data loss** is prevented.
- 3) **8155 also makes the BF high** to indicate to the output peripheral that **data is available** on the data bus.
- 4) The **peripheral accepts the data** and sends an acknowledgement by making the **STB low**. The **data byte is thus transferred** to the peripheral.
- 5) Now, the **BF goes low and ACK goes high**.
- 6) The **INTR line becomes high** to inform the **μP** that **another byte** can be **sent**. i.e. the output port is empty. This process is repeated for further bytes.

## **Bharat Acharya Education**

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes  
Free: PDFs of Theory explanation, VIVA questions and MCQs

Start Learning... NOW!

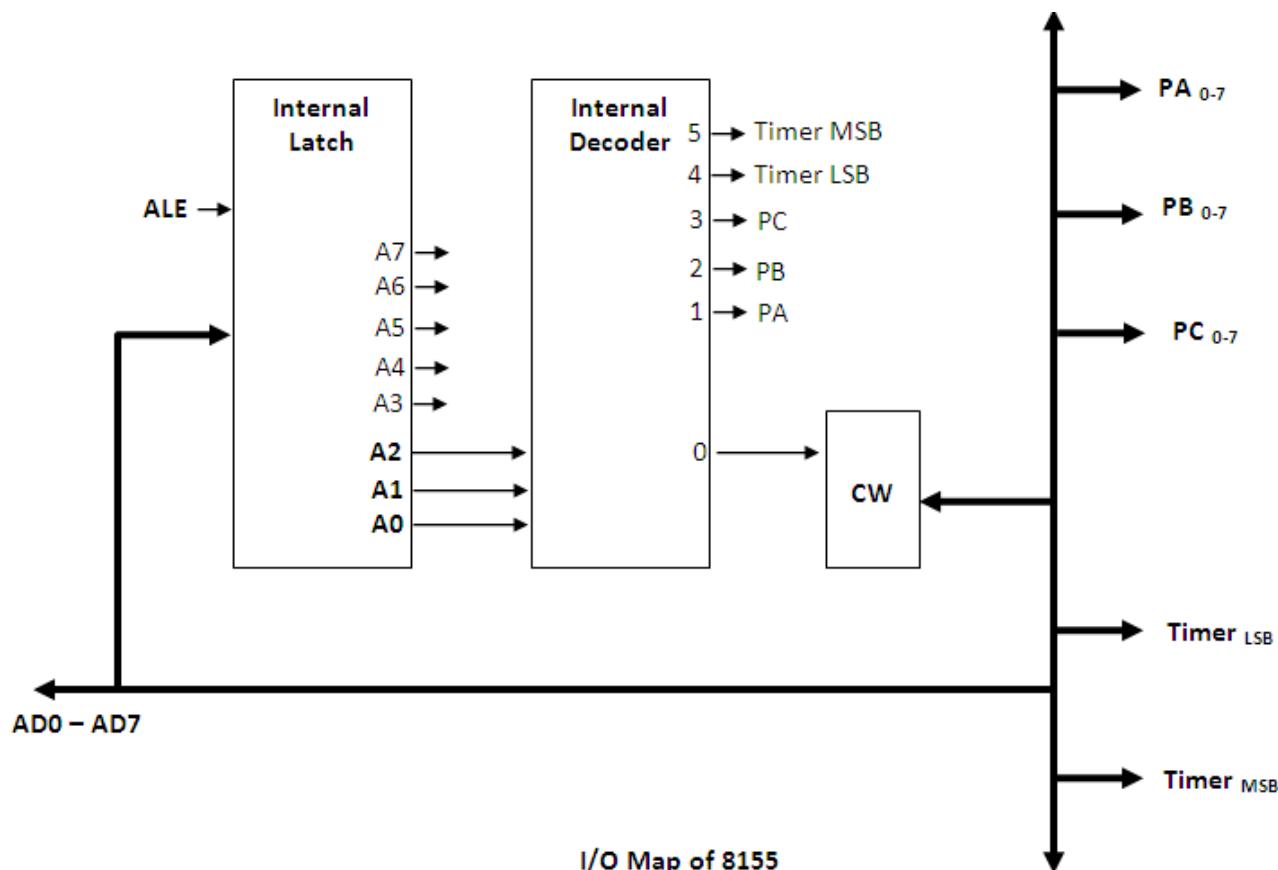
**[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)**

Official WhatsApp number:

**+91 9136428051**

## 8155 PROGRAMMABLE I/O AND TIMER

### 8155 INTERNAL I/O SECTION AND INTERFACING



	A15 (A7)	A14 (A6)	A13 (A5)	A12 (A4)	A11 (A3)	A10 (A2)	A9 (A1)	A8 (A0)	
Control Word	0	0	1	0	0	0	0	0	20 H
Port A	0	0	1	0	0	0	0	1	21 H
Port B	0	0	1	0	0	0	1	0	22 H
Port C	0	0	1	0	0	0	1	1	23 H
Timer LSB	0	0	1	0	0	1	0	0	24 H
Timer MSB	0	0	1	0	0	1	0	1	25 H
----- Chip Selection -----					--- internal --- selection				

- The internal I/O Section of 8155 has these 6 components
  1. Control Word
  2. Port A
  3. Port B
  4. Port C
  5. Timer LSB
  6. Timer MSB
- Each of these components has a unique address.
- If We map 8155 at the address 20H then its internal I/O Address are as follows...
  1. Control Word at 20H
  2. Port A at 21H
  3. Port B at 22H
  4. Port C at 23H
  5. Timer LSB at 24H
  6. Timer MSB at 25H
- The “Internal latch” will separate A0-A7 and D0-D7 from the multiplexed AD0-AD7 with the help of ALE.
- The “Internal Decoder” will decode A0 A1 A2 to identify the 6 internal options mentioned above.
- Based on the address given by A0-A7, the data carried in D0-D7 will go to that appropriate address.
- In the external interface a latch is not needed as internal latch is present in 8155.
- Similarly, IO/M, RD, WR are internally decoded by 8155 eliminating the need for a decoder in the circuit.
- This makes a very compact circuit and hence is useful in building the minimum system of 8185 along with 8155 (provides RAM, ports and Timer) and 8355/8755 (provides ROM).

**Special Note:**

If you are learning this by piracy, then you are not my student.  
You are simply a thief! #PoorUpbringing

**Special Note: A15-A8**

We use A5-A8 for external decoding because A0-A7 is not explicitly available externally as there is no external latch.

I/O Address are duplicated. Whatever is produced on A0-A7 is also available on A8-A15.

## **Bharat Acharya Education**

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes  
Free: PDFs of Theory explanation, VIVA questions and MCQs

Start Learning... NOW!

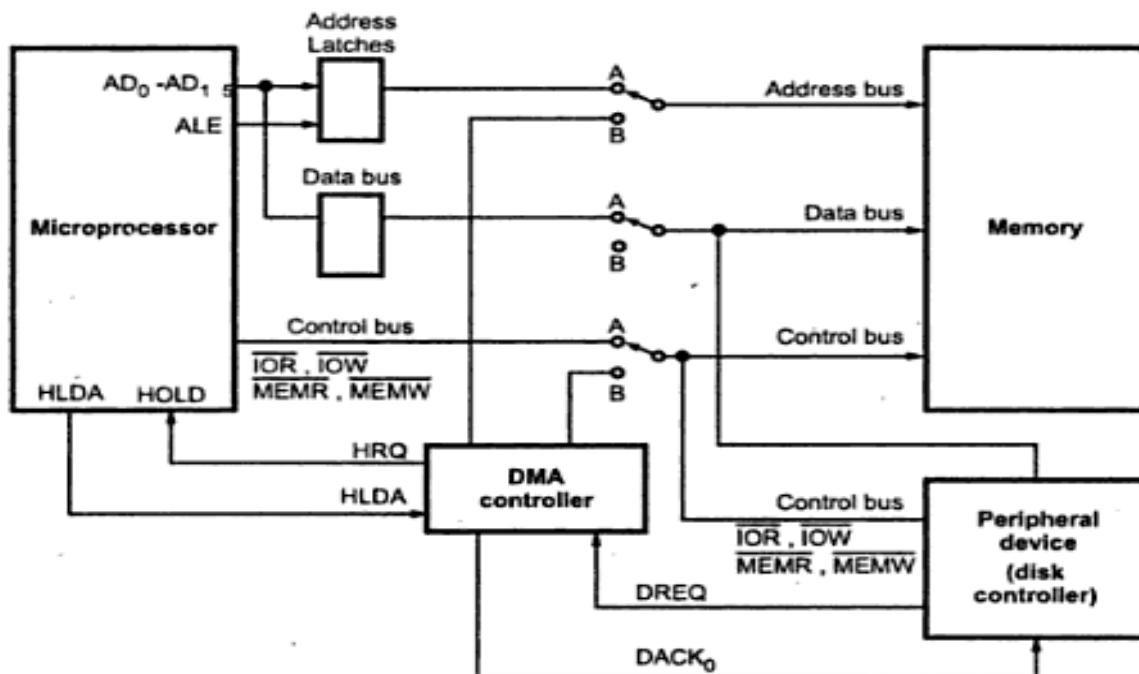
**[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)**

Official WhatsApp number:

**+91 9136428051**

## 8237 / 8257 DMA CONTROLLER

### CONCEPT OF DMA: DIRECT MEMORY ACCESS



DMA Transfer is a hardware controlled I/O Transfer technique.

It is mainly used for high-speed data transfer between I/O and Memory where the speed of the peripheral is generally faster than the  $\mu$ P. 😊 For doubts contact Bharat Sir on 98204 08217

In Program Controlled I/O, Status or interrupt driven I/O the speed of transfer is slow mainly because instructions need to be decoded and then executed for the transfer.

DMA transfer is software independent and hence much faster.

A device known as the DMA Controller (DMAC) is responsible for the DMA transfer.

The sequence of DMA transfer is as follows:

- 1) μP initializes the DMAC by giving the starting address and the number of bytes to be transferred.
- 2) An I/O device requests the DMAC, to perform DMA transfer, through the DREQ line.
- 3) The DMAC in turn sends a request signal to the μP, through the HOLD line.
- 4) The μP finishes the current machine cycle and releases the system bus (gets disconnected from it).  
It also acknowledges receiving the HOLD signal through the HLDA line.
- 5) The DMAC acquires control of the system bus.  
The DMAC sends the DACK signal to the I/O peripheral and the DMA transfer begins.
- 6) After every byte is transferred, the Address Reg is incremented (or decremented) and the Count Reg is decremented.
- 7) This continues till the Count reaches zero (Terminal Count). Now the DMA transfer is completed.
- 8) At the end of the transfer, the system bus is released by the DMAC by making HOLD = 0.

Thus μP takes control of the system bus and continues its operation.

The DMA Controller (DMAC) does DMA transfer through its channels.

The minimum requirements of each channel are:

- i. Address Register (to store the starting address for the transfer).
- ii. Count Register (to store the number of bytes to be transferred).
- iii. A DREQ signal from the IO device.
- iv. A DACK signal to the IO device.

#### **Special Note: DMA SPEED-UP**

DMA is fast due to two reasons

- 1) Transfer is direct hence it takes one cycle instead of two cycles
- 2) It is a hardware-based transfer hence no time is wasted in fetching and decoding instructions by the DMAC

#### **Special Note:**

If you are learning this by piracy, then you are not my student.  
You are simply a thief! #PoorUpbringing

## TRANSFER MODES OF 8237 DMAC

8237 has four modes of data transfer:

### 1) Block Transfer Mode

In this mode, the DMAC is programmed to transfer ALL THE BYTES in one complete DMA operation.

After a byte is transferred, the CAR and CWCR are adjusted accordingly.

The system bus is returned to the  $\mu$ P, ONLY after all the bytes are transferred. I.e. TC is reached or EOP signal is issued.

The DREQ signal needs to be active only in the beginning for requesting the DMA service initially. Thereafter DREQ can become low during the transfer.

### 2) Single Transfer Mode (Cycle Stealing)

In this mode, the DMAC is programmed to transfer ONLY ONE BYTE in one complete DMA operation.

After a byte is transferred, the CAR and CWCR are adjusted accordingly.

The system bus is returned to the  $\mu$ P.

For further bytes to be transferred, the DREQ line must go active again, and then the entire operation is repeated.

### 3) Demand Transfer Mode

It is very similar to Block Transfer, except that the DREQ must active throughout the DMA operation.

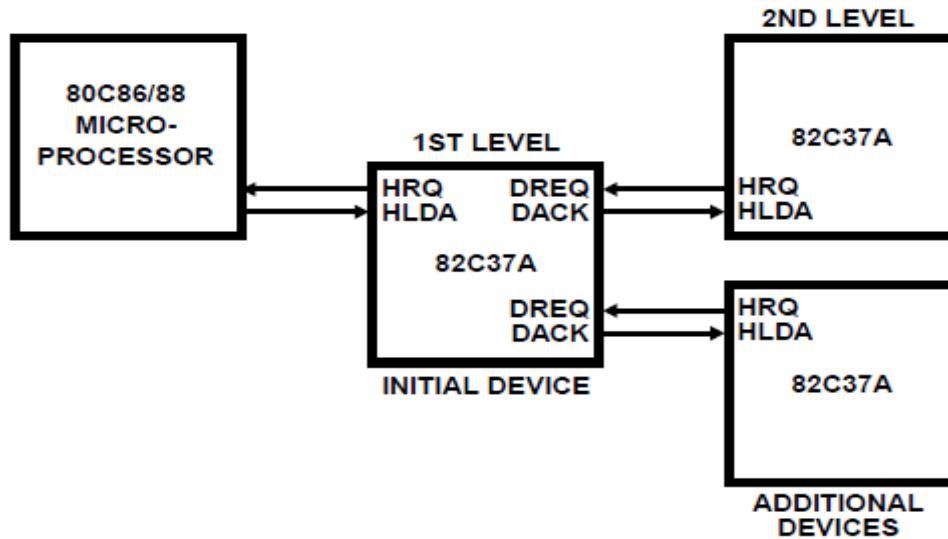
If during the operation DREQ goes low, the DMA operation is stopped and the busses are returned to the  $\mu$ P. #Please refer Bharat Sir's Lecture Notes for this ... In the meantime, the  $\mu$ P can continue with its own operations. Once DREQ goes high again, the DMA operation continues from where it had stopped.

### 4) Cascade Transfer Mode (Only for 8237, not in general)

In this mode, more than one DMACs are cascaded together.

It is used to increase the number of devices interfaced to the  $\mu$ P.

Here we have one Master DMAC, to which one or more Slave DMACs are connected. The Slave gives HRQ to the Master on the DREQ of the Master, and the Master gives HRQ to the  $\mu$ P on the HOLD of the  $\mu$ P.



#### Special Note: Transparent Mode

Some DMA Controllers also have a Hidden Mode (Transparent Mode)

Here transfer happens only when the mpu is idle

It is very similar to “Download when CPU free” mode in many Internet Download Managers



#### Special Note:

If you are learning this by piracy, then you are not my student.  
You are simply a thief! #PoorUpbringing

## PRIORITY MODES OF 8237 DMAC

8237 has two priority methods

### 1) Fixed Priority

This is the default mode of 8237.

Here the priorities of the channels are fixed.

Channel 0 has the highest priority, followed by Channel 1, Channel 2 and finally Channel 3 having the lowest priority.

### 2) Rotating Priority

It is a flexible priority mode.

In this mode, the Channel, which was most recently serviced, receives the lowest priority.

This prevents any one Channel from dominating the system.

Eg: If Channel 0 was just serviced then Channel 0 gets lowest priority.

Channel 1 gets highest priority, followed by Channel 2, Channel 3 and finally Channel 0.

Before CH.0 is serviced

Highest	CH.0
	CH.1
	CH.2
Lowest	CH.3

After CH.0 is serviced

CH.1	Highest
CH.2	
CH.3	
CH.0	Lowest

← Active DMA request

## OPERATING CYCLES OF 8237 DMAC

There are mainly two operation cycles of a DMAC.

### i. Idle Cycle

After Reset, the DMAC is in idle state (idle cycle).

During idle state, no DMA operation is taking place.

No DMA requests are active.

The initialization of the DMAC takes place in the idle mode.

### ii. Active Cycle

Once DMA operation begins, the DMAC is said to be in active mode.

Now the DMAC controls the system bus.

There are three types of ACTIVE DMA Cycles while performing DMA transfer:

#### 1) DMA Read

The DMAC reads data from the memory and writes into to the I/O device.

Thus, MEMR and IOW signals are used.

#### 2) DMA Write

The DMAC reads data from the I/O device and writes into to the memory.

Thus, IOR and MEMW signals are used.

#### 3) DMA Verify

In this cycle, 8237 does not generate any control signals.

Hence, no data transfer takes place.

During this time, the peripheral and the DMAC verify the correctness of the data transferred, using some error detection method.

## **Bharat Acharya Education**

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes  
Free: PDFs of Theory explanation, VIVA questions and MCQs

Start Learning... NOW!

**[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)**

Official WhatsApp number:

**+91 9136428051**



## MEMORY MAPPED I/O vs I/O MAPPED I/O

	Memory Mapped I/O	I/O Mapped I/O
1)	The I/O devices are <b>mapped into the memory space</b>	The I/O devices are <b>mapped independent into the I/O Space.</b>
2)	The <b>size of the I/O address</b> is the <b>same as</b> the size of the <b>memory address</b> i.e. 16 bits	The <b>size of the I/O address</b> is <b>independent of the memory address</b> . Here I/O address is of 8-bits
3)	<b>Increases</b> the <b>number of I/O ports</b> interfaced. Eg: 65536 i.e. <b>64K in 8085</b>	<b>Restricts</b> the number of I/O Ports interfaced. Eg: <b>256 in 8085</b> .
4)	Many instructions such as: <b>LDA, STA, LHLD</b> etc can be used to access the I/O devices	<b>Only IN</b> and <b>OUT</b> instructions can be used to access the I/O devices.
5)	<b>Data transfer</b> can take place <b>between any register</b> and an I/O device	Data transfer can take place <b>only between the Accumulator</b> and an I/O device.
6)	<b>Only MEMR, MEMW</b> control signals are required	All Four Control Signals, <b>IOR, IOW, MEMR, MEMW</b> signals are required.
7)	<b>Execution is slower as access time is more</b>	Execution is <b>faster as access time is less</b> .
8)	<b>Decoding 16-bit</b> address requires <b>more hardware</b>	Decoding <b>8-bit</b> address requires <b>less hardware</b>
9)	<b>μP does not differentiate</b> between memory and I/O	<b>μP differentiates</b> between memory and I/O
10)	Used in <b>microcontrollers for industrial applications</b> where we need to interface a large number of I/O devices.	Used in <b>microprocessors for PCs</b> where the number of I/O devices interfaced is less.

### Important Tip:

If nothing is mentioned please use I/O mapped I/O by default

### Special Note:

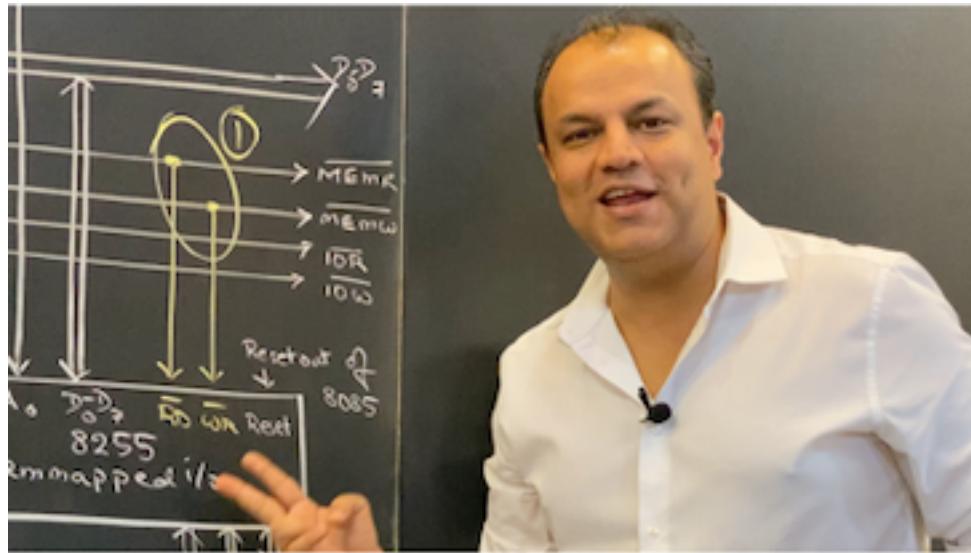
If you are learning this by piracy, then you are not my student.  
You are simply a thief! #PoorUpbringing



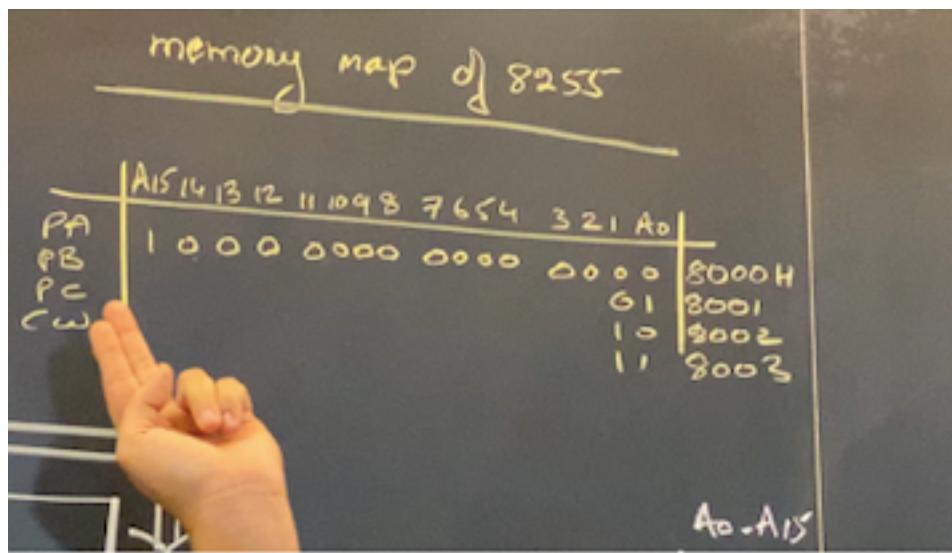
## 8255 INTERFACE WITH 8085 USING MEMORY MAPPED I/O

Points that will change when we use memory mapped I/O are...

- MEMR and MEMW signals will be given to the RD and WR signals of 8255 instead of IOR and IOW



- Memory map of 8255 will be made instead of IO map. Use 16-bit address like 8000H

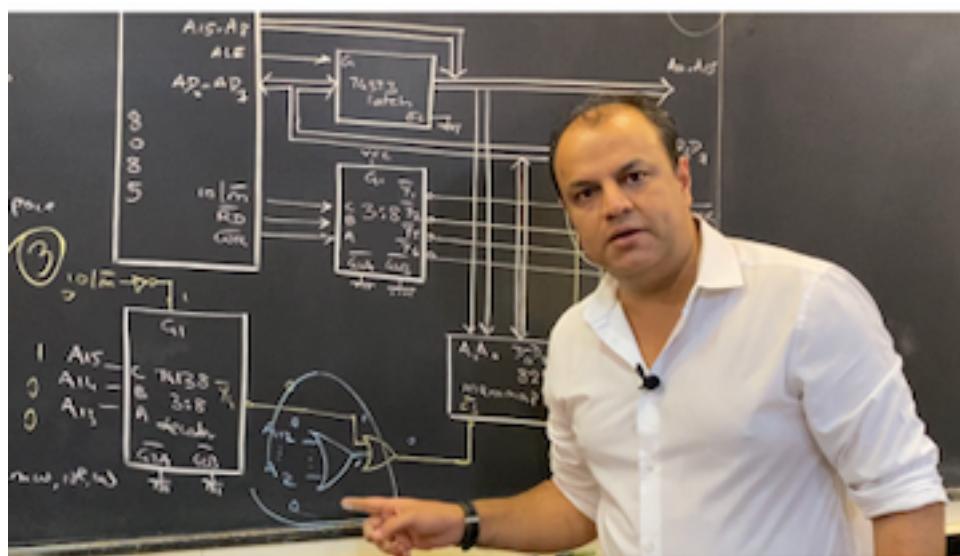




- The 3:8 decoder used for CS to 8255 should be enabled only for memory operations. Hence connect IO/M with an inverter to the G1 line of the decoder.



- Manage the spare lines A12-A2 using a big OR gate to complete FULL address decoding.



**Special Note:**

If you are learning this by piracy, then you are not my student.  
You are simply a thief! #PoorUpbringing



## Bharat Acharya Education

Learn...

8085 | 8086 | 80386 | Pentium | 8051 | ARM7 | COA

Fees: 1199/- | Duration: 6 months | Activation: Immediate | Certification: Yes  
Free: PDFs of Theory explanation, VIVA questions and MCQs

Start Learning... NOW [www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

Official WhatsApp number: **+91 9136428051**

My 8085 MCQ Course on UDEMY

<https://www.udemy.com/course/8085-microprocessor-multiple-choice-questions-with-solutions/?referralCode=22D1E2B940FBD516CF9E>

**Q1: Write a program to multiply 2 8-bit numbers stored at 2000h and 2001h.  
Store the result at 2002H and 2003H**

```

LXI H, 0000H      ; Result will be in HL, initialized to 0
LXI D, 0000H      ; DE initialized to 0

LDA 2000H          ; Take 1st operand
ADI 00H            ; For Zero check
JZ Store            ; For Zero check
MOV E,A            ; DE = 1st operand

LDA 2001H          ; Take 2nd operand
ADI 00H            ; For Zero check
JZ Store            ; For Zero check
MOV C,A            ; C = 2nd operand

Back: DAD D         ; Add the 1st operand to HL, HL = HL + DE
DCR C              ; Decrement the 2nd operand
JNZ Back            ; Loop till C becomes 0

Store:SHLD 2002H    ; Store the result

HLT                ; End your program

```



**Q2: Write a program to divide an 8-bit number stored at 2000h by another stored at 2001h. Store the result at 2002H (Q) and 2003H (R)**

```
LDA 2001H      ; Take divisor in A
ADI 00H      ; For Zero check
JZ Exit      ; If Zero, simply exit the program
MOV C,A      ; C = divisor
MVI E,00H      ; E = 0 (this will be the quotient)
LDA 2000H      ; A = Dividend
```

```
Back: CMP C      ; Compare A and C by doing A-C
JC Next      ; If A < C then move out of the loop
SUB C      ; A ← A - C, actually do the subtraction
INR E      ; Increment Quotient
JMP Back      ; Loop back
```

```
Next: STA 2003H      ; Store the remainder from A to 2003H
MOV A,E      ; Move Quotient from E to A
STA 2002H      ; Store the quotient from A to 2002H
```

```
Exit: HLT      ; End your program
```

<https://www.bharatacharyaeducation.com>

Learn...

8085 | 8086 | 80386 | Pentium |

8051 | ARM7 | COA

Fees: 1199/-

Duration: 6 months

Activation: Immediate

Certification: Yes

Free: PDFs of theory explanation

Free: VIVA questions and answers

Free: PDF of Multiple Choice Questions

Start Learning... NOW!

<https://www.bharatacharyaeducation.com>

Order my Books here...

8086 Microprocessor book

Link: <https://amzn.to/3qHDpJH>

8051 Microcontroller book

Link: <https://amzn.to/3aFQkXc>

#bharatacharya

#bharatacharyaeducation

#8086 #8051 #8085 #80386 #pentium

#microprocessor #microcontrollers



## Programming on the SIMULATOR

# 8085

## Microprocessor

Mobile	+91 9820408217
Email	bharatsir@hotmail.com
Website	<a href="http://www.BharatAcharyaEducation.com">www.BharatAcharyaEducation.com</a>
Youtube	<a href="https://youtube.com/c/BharatAcharyaEducation">youtube.com/c/BharatAcharyaEducation</a>
Instagram	<a href="https://www.instagram.com/@BharatAcharya.in">@BharatAcharya.in</a>
Facebook	<a href="https://www.facebook.com/BharatAcharyaEducation">facebook.com/BharatAcharyaEducation</a>



## 8085 Programming – On the Simulator

**Q1) Write a Program to Add Two Numbers Stored at 2000h and 2001h. Store the Sum at 2002h and the carry at 2003h.**

Solution:

```
mvi c, 00h      ; assume there is no carry
lda 2000h      ; a = 1st number
mov b, a        ; b = 1st number
lda 2001h      ; a = 2nd number
add b          ; a = sum
jnc skip
inr c          ; make C = 1 because there is a carry
skip:sta 2002h ; [2002] = sum
mov a, c        ; a = carry from C register
sta 2003h      ; [2003] = carry
hlt
```

**Q2) Write a Program to Add Two Numbers Stored at 2000h and 2001h. Store the Sum at 2002h and the carry at 2003h, using Memory pointer - M**

Solution:

```
mvi c, 00h      ; assume there is no carry
lxi h, 2000h    ; m points to 2000h
mov b, m        ; b = 1st number
inx h          ; m points to 20001h
mov a, m        ; a = 2nd number
add b          ; a = sum
jnc skip
inr c          ; make c=1 if there is a carry
skip: inx h    ; m points to 2002h
mov m, a        ; [2002h] = sum
inx h          ; make m ponit to 2003h
mov m, c        ; [2003h] = carry
hlt
```





**Q3) Write a Program to transfer a block of 10 bytes from 2000h to 2020h**

Solution:

```
Ixi b, 2000h      ; bc points to source
Ixi d, 2020h      ; de points to destination
mvi l, 0ah        ; l = count of 10

back:ldax b       ; a = data from source
stax d            ; a gets stored at the destination
inx b             ; increment source pointer
inx d             ; increment destination pointer
dcr l             ; decrement the count
jnz back
hlt
```

**Q4) Write a Program to invert and transfer a block of 10 bytes from 2000h to 2020h**

Solution:

```
Ixi b, 2000h      ; bc points to source
Ixi d, 2029h      ; de points to end of destination
mvi l, 0ah        ; l = count of 10

back:ldax b       ; a = data from source
stax d            ; a gets stored at the destination
inx b             ; increment source pointer
dcx d             ; decrement destination pointer
dcr l             ; decrement the count
jnz back
hlt
```



**Q5) Write a Program to transfer an overlapping block of 10 bytes from 2000h to 2004h**

Solution:

```
Ixi b, 2009h      ; bc points to end of source
Ixi d, 200dh      ; de points to end of destination
Mvi l, 0ah        ; l = count of 10

back:ldax b       ; a = data from source
Stax d            ; a gets stored at the destination
Dcx b             ; decrement source pointer
Dcx d             ; decrement destination pointer
Dcr l             ; decrement the count
Jnz back
Hlt
```

### Contact us

---

<b>Instagram</b>	<a href="https://www.instagram.com/@bharatacharya.in">@bharatacharya.in</a>
<b>Facebook</b>	<a href="https://www.facebook.com/BharatAcharyaEducation">@BharatAcharyaEducation</a>
<b>Youtube</b>	Bharat Acharya Education
<b>WhatsApp</b>	+91 91364 28051
<b>Email</b>	<a href="mailto:bharatsir@hotmail.com">bharatsir@hotmail.com</a>
<b>Website</b>	<a href="http://www.BharatAcharyaEducation.com">www.BharatAcharyaEducation.com</a>

**Follow our tags** | #bharatacharyaeducation #bharatacharya



[BharatAcharya.in](https://www.instagram.com/@bharatacharya.in)



[Bharat Acharya Education](https://www.youtube.com/BharatAcharyaEducation)



[Bharat Acharya Education](https://www.facebook.com/BharatAcharyaEducation)



[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



+91 913 642 8051



[bharatsir@hotmail.com](mailto:bharatsir@hotmail.com)



## Sorting Program on the Simulator

# 8085

## Microprocessor

### Contact us

---

**Instagram** | @bharatacharya.in

**Facebook** | @BharatAcharyaEducation

**Youtube** | Bharat Acharya Education

**WhatsApp** | +91 91364 28051

**Email** | bharatsir@hotmail.com

**Website** | www.BharatAcharyaEducation.com

**Follow our tags** | #bharatacharyaeducation #bharatacharya

UNACADEMY Coupon

**ACHARYA10**

Min 10% OFF



## 8085 Programming

Sort a series of 5 numbers stored at 2000H in ACSENDING order.

		B = 4	C = 4	C = 4	C = 4
HL	Address	Data			
	2000H	05	04	03	02
	2001H	04	03	02	01
	2002H	03	02	01	03
	2003H	02	01	04	04
	2004H	01	05	05	05
		C = 0	C = 0	C = 0	C = 0
		B = 3	B = 2	B = 1	B = 0

### Contact us

- Instagram** | [@bharatacharya.in](https://www.instagram.com/bharatacharya.in)  
**Facebook** | [@BharatAcharyaEducation](https://www.facebook.com/BharatAcharyaEducation)  
**Youtube** | Bharat Acharya Education  
**WhatsApp** | +91 91364 28051  
**Email** | [bharatsir@hotmail.com](mailto:bharatsir@hotmail.com)  
**Website** | [www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

**Follow our tags** | #bharatacharyaeducation #bharatacharya

UNACADEMY Coupon

**ACHARYA10**

Min 10% OFF



BharatAcharya.in



Bharat Acharya Education



Bharat Acharya Education



[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



+91 913 642 8051



[bharatsir@hotmail.com](mailto:bharatsir@hotmail.com)



## Registers

A/PSW	0x 03 57
BC	0x 00 00
DE	0x 01 00
HL	0x 20 04
SP	0x FF FF
PC	0x 08 1E

## Flags

Z	<input checked="" type="checkbox"/>
S	<input type="checkbox"/>
P	<input checked="" type="checkbox"/>
C	<input checked="" type="checkbox"/>
AC	<input checked="" type="checkbox"/>



Load at 0x0800

main.asm

```
1 ; Program to sort a series
2 ; Bharat Acharya Education
3
4 MVI B, 04H; outer loop count
5 BACK2: LXI H, 2000H; pointer
6 MVI C, 04H; inner loop count
7 BACK1: MOV A, M; A = 1st Number
8 INX H; M = 2nd location
9 CMP M; 1st - 2nd number
10 JC SKIP; if 1st is bigger
11 JZ SKIP; if both equal
12 MOV D, M; D = 2nd No
13 MOV M, A; 1st No is now at 2nd loc
14 DCX H; M points to 1st loc
15 MOV M, D; 2nd No is now at 1st loc
16 INX H; M points to 2nd No
17 SKIP: DCR C; dec count
18 JNZ BACK1; inner loop
19 DCR B; dec count
20 JNZ BACK2; outer loop
21 HLT
```

## Contact us

- Instagram** | @bharatacharya.in  
**Facebook** | @BharatAcharyaEducation  
**Youtube** | Bharat Acharya Education  
**WhatsApp** | +91 91364 28051  
**Email** | bharatsir@hotmail.com  
**Website** | www.BharatAcharyaEducation.com  
**Follow our tags** | #bharatacharyaeducation #bharatacharya

UNACADEMY Coupon

**ACHARYA10**

Min 10% OFF



BharatAcharya.in



Bharat Acharya Education



Bharat Acharya Education



www.BharatAcharyaEducation.com



+91 913 642 8051



bharatsir@hotmail.com



## Memory View



0x

2000

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
200	01	02	02	03	05	00	00	00	00	00	00	00	00	00	00	00
201	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
202	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
203	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
204	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
205	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
206	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
207	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
208	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
209	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

## Contact us

**Instagram** | [@bharatacharya.in](#)

**Facebook** | [@BharatAcharyaEducation](#)

**Youtube** | Bharat Acharya Education

**WhatsApp** | +91 91364 28051

**Email** | [bharatsir@hotmail.com](mailto:bharatsir@hotmail.com)

**Website** | [www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

**Follow our tags** | #bharatacharyaeducation #bharatacharya

UNACADEMY Coupon

**ACHARYA10**

Min 10% OFF



[BharatAcharya.in](#)



[Bharat Acharya Education](#)



[Bharat Acharya Education](#)



[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



+91 913 642 8051



[bharatsir@hotmail.com](mailto:bharatsir@hotmail.com)



# 8085

MCQ – Aug 2021



Bharat  
Acharya  
Education

Learn the full subject at:  
[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

6 Months | Unlimited Views | Rs 1199/-

Website	<a href="http://www.BharatAcharyaEducation.com">www.BharatAcharyaEducation.com</a>
Mobile	+91 9820408217
Email	<a href="mailto:bharatsir@hotmail.com">bharatsir@hotmail.com</a>
Youtube	<a href="https://youtube.com/c/BharatAcharyaEducation">youtube.com/c/BharatAcharyaEducation</a>
Instagram	<a href="https://www.instagram.com/@BharatAcharyaEducation">@BharatAcharyaEducation</a>
Facebook	<a href="https://www.facebook.com/BharatAcharyaEducation">facebook.com/BharatAcharyaEducation</a>



## Section 1

### **Introduction and Architecture of 8085**



**Q1) 8085 is an 8-bit processor because...**

- (a) It has an 8-bit data bus
- (b) It has an 8-bit ALU
- (c) It has an 8-bit address bus
- (d) It has an 8-bit control bus

**Answer...**

- (a) It has an 8-bit data bus
- (b) It has an 8-bit ALU**
- (c) It has an 8-bit address bus
- (d) It has an 8-bit control bus

Video Reference: 8085 | **Architecture**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q2) Memory pointer M is addressed by...**

- (a) BC Pair
- (b) DE Pair
- (c) HL Pair
- (d) WZ Pair

**Answer...**

- (a) BC Pair
- (b) DE Pair
- (c) HL Pair**
- (d) WZ Pair

Video Reference: 8085 | Architecture

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q3) The total memory accessed by 8085 is**

- (a) 256 bytes
- (b) 64 KB
- (c) 64 bytes
- (d) undefined

**Answer...**

- (a) 256 bytes
- (b) 64 KB**
- (c) 64 bytes
- (d) undefined

Video Reference: 8085 | **Pin Diagram**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q4) The total Number of I/O Address in 8085 ...**

- (a) 8
- (b) 16
- (c) 32
- (d) 256

**Answer...**

- (a) 8
- (b) 16
- (c) 32
- (d) 256**

Video Reference: 8085 | **Instruction Set Part 5**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



### **Q5) XCHG instruction works between?**

- (a) HL pair and BC pair
- (b) HL pair and DE pair
- (c) Any two pairs
- (d) HL pair and SP

### **Answer...**

- (a) HL pair and BC pair
- (b) HL pair and DE pair**
- (c) Any two pairs
- (d) HL pair and SP

Video Reference: 8085 | **Architecture**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q6) MSB of the result is indicated by the ...**

- (a) Carry Flag
- (b) Zero Flag
- (c) Parity Flag
- (d) Sign Flag

**Answer...**

- (a) Carry Flag
- (b) Zero Flag
- (c) Parity Flag
- (d) Sign Flag**

Video Reference: 8085 | **Flag Register**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q7) What is the range of an 8-bit signed number?**

- (a) 0...255
- (b) -127...+128
- (c) -128....+128
- (d) -128...+127

**Answer...**

- (a) 0...255
- (b) -127...+128
- (c) -128....+128
- (d) -128...+127**

Video Reference: 8085 | **Flag Register**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q8) How is -01H stored in “A” register?**

- (a) – 0000 0001
- (b) 1000 0001
- (c) 1111 1111
- (d) 1111 1110



**Answer...**

- (a) – 0000 0001
- (b) 1000 0001
- (c) 1111 1111**
- (d) 1111 1110

**Important Type  
of Question**

Video Reference: 8085 | **Flag Register**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q9) SID and SOD are accessed by...**

- (a) RIM and SIM instructions
- (b) IN and OUT instructions
- (c) LDA and STA instructions
- (d) Cannot be accessed

**Answer...**

- (a) RIM and SIM instructions**
- (b) IN and OUT instructions
- (c) LDA and STA instructions
- (d) Cannot be accessed

Video Reference: 8085 | **Instruction Set Part 5**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q10) 8085 memory address range is,,,**

- (a) 0000... 0FFFH
- (b) 0000... 3FFFH
- (c) 0000... 7FFFH
- (d) 0000... FFFFH



**Important Type  
of Question**

**Answer...**

- (a) 0000... 0FFFH
- (b) 0000... 3FFFH
- (c) 0000... 7FFFH
- (d) 0000... FFFFH**

Video Reference: 8085 | **Pin Diagram**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q) Should you learn by cheating / piracy?**

- (a) Yes, I am raised like this!
- (b) Who cares about the teacher's effort? I am cheap!
- (c) Education by cheating will still help me!
- (d) NO. We should support good teachers who work so hard yet charge very less!

**Answer...**

I leave this answer to YOU!



**Video Courses for You**

- Microprocessor | 8085
- Microprocessor | 8086
- Microprocessor | 80386
- Microprocessor | Pentium
- Microcontroller | 8051
- Microcontroller | ARM
- Computer Organisation & Architecture

**6 Months | Unlimited Views | 1199/-**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



## Section 2

# **Addressing Modes and Instruction Set**



**Q11) Immediate addressing mode means**

- (a) Data is given in the instruction
- (b) Address is given in the instruction
- (c) Data is given in registers
- (d) Address is given using a register

**Answer...**

- (a) Data is given in the instruction**
- (b) Address is given in the instruction
- (c) Data is given in registers
- (d) Address is given using a register

Video Reference: 8085 | **Addressing Modes**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q12) ADD B is an example of**

- (a) Immediate Addressing mode
- (b) Register Addressing mode
- (c) Direct Addressing mode
- (d) Indirect Addressing mode

**Answer...**

- (a) Immediate Addressing mode
- (b) Register Addressing mode**
- (c) Direct Addressing mode
- (d) Indirect Addressing mode

Video Reference: 8085 | **Addressing Modes**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q13) In which addressing mode we give the address of the operand?**

- (a) Immediate Addressing mode
- (b) Register Addressing mode
- (c) Direct Addressing mode
- (d) Indirect Addressing mode

**Answer...**

- (a) Immediate Addressing mode
- (b) Register Addressing mode
- (c) Direct Addressing mode**
- (d) Indirect Addressing mode

Video Reference: 8085 | **Addressing Modes**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q14) Indirect addressing mode is used when we want to...**

- (a) Not disclose the address
- (b) Access a series of locations
- (c) Access the odd bank
- (d) Indirectly end the program

**Answer...**

- (a) Not disclose the address
- (b) Access a series of locations**
- (c) Access the odd bank
- (d) Indirectly end the program

Video Reference: 8085 | **Addressing Modes**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



### Q15) Assume

[2000] = 10H, [2001] = 20H, [2002] = 30H

[2003] = 40H, [2004] = 50H, [2005] = 60H

What will happen after..

LDA 2000H

STA 2001H



- (a) [2000] = 10H, A = 20H
- (b) [2000] = 20H, A = 20H
- (c) [2000] = 10H, A = 20H
- (d) [2000] = 10H, [2001] = 10H

**Important Type**

### Answer...

- (a) [2000] = 10H, A = 20H
- (b) [2000] = 20H, A = 20H
- (c) [2000] = 10H, A = 20H
- (d) [2000] = 10H, [2001] = 10H**

Video Reference: 8085 | **Addressing Modes**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



### Q16) Assume

[2000] = 10H, [2001] = 20H, [2002] = 30H

[2003] = 40H, [2004] = 50H, [2005] = 60H

What will happen after..

MVI A, 25H

STA 2001H



- (a) [2001] = 25H, A = 20H
- (b) [2001] = 25H, A = 25H
- (c) [2001] = 20H, A = 25H
- (d) [2001] = 20H, A = 20H

**Important Type**

### Answer...

- (a) [2001] = 25H, A = 20H
- (b) [2001] = 25H, A = 25H**
- (c) [2001] = 20H, A = 25H
- (d) [2001] = 20H, A = 20H

Video Reference: 8085 | **Addressing Modes**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

### Q17) Assume

[2000] = 10H, [2001] = 20H, [2002] = 30H  
[2003] = 40H, [2004] = 50H, [2005] = 60H

What will happen after..

LDA 2000H

MOV B, A

LDA 2001H

ADD B

STA 2005H



**Important Type**

- (a) [2005] = 60H, A = 60H
- (b) [2005] = 30H, A = 60H
- (c) [2005] = 60H, A = 30H
- (d) [2005] = 30H, A = 30H

### Answer...

- (a) [2005] = 60H, A = 60H
- (b) [2005] = 30H, A = 60H
- (c) [2005] = 60H, A = 30H
- (d) [2005] = 30H, A = 30H**

Video Reference: 8085 | **Addressing Modes**



### Q18) Assume

[2000] = 10H, [2001] = 20H, [2002] = 30H

[2003] = 40H, [2004] = 50H, [2005] = 60H

What will happen after..

LDA 2000H

MOV B, A

LDA 2001H

STA 2000H

MOV A, B

STA 2001H



Important Type

- (a) [2000] = 10H, A = 60H
- (b) [2001] = 20H, A = 30H
- (c) [2000] = 20H, [2001] = 10H
- (d) All of the above

### Answer...

- (a) [2000] = 10H, A = 60H
- (b) [2001] = 20H, A = 30H
- (c) [2000] = 20H, [2001] = 10H**
- (d) All of the above

Video Reference: 8085 | Addressing Modes



### Q19) Assume

[2000] = 10H, [2001] = 20H, [2002] = 30H

[2003] = 40H, [2004] = 50H, [2005] = 60H

What will happen after..

LXI H, 2000H

MOV A, M

INX H

ADD M

DCX H

MOV M, A



**Important Type**

- (a) HL = 2000H, A = 30H
- (b) HL = 2000H, [2000] = 30H
- (c) A = 30H, [2001H] = 20H
- (d) All of the above

### Answer...

- (a) HL = 2000H, A = 30H
- (b) HL = 2000H, [2000] = 30H
- (c) A = 30H, [2001H] = 20H
- (d) All of the above**

Video Reference: 8085 | **Addressing Modes**

## Q20) Assume

[2000] = 10H, [2001] = 20H, [2002] = 30H  
[2003] = 40H, [2004] = 50H, [2005] = 60H

What will happen after..

LHLD 2000H

DAD H

SHLD 200H



**Important Type**

- (a) HL = 2010H, [2000] = 30H
- (b) HL = 2010H, [2000] = 20H
- (c) HL = 4020H, [2000] = 30H
- (d) HL = 4020H, [2000] = 20H

## Answer...

- (a) HL = 2010H, [2000] = 30H
- (b) HL = 2010H, [2000] = 20H
- (c) HL = 4020H, [2000] = 30H
- (d) HL = 4020H, [2000] = 20H**

Video Reference: 8085 | Instructions Part 1 and 2

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



## Q21) XTHL exchanges

- (a) HL with top two elements of stack
- (b) HL and SP
- (c) DE and SP
- (d) Invalid Instruction

**Answer...**

- (a) HL with top two elements of stack**
- (b) HL and SP
- (c) DE and SP
- (d) Invalid Instruction

Video Reference: 8085 | **Instructions Part 1**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q22) DAA is used to...**

- (a) Convert Hex to Decimal
- (b) Convert BCD to ASCII
- (c) Adjust after Decimal Addition
- (d) Perform 16-bit addition

**Answer...**

- (a) Convert Hex to Decimal
- (b) Convert BCD to ASCII
- (c) Adjust after Decimal Addition**
- (d) Perform 16-bit addition

Video Reference: 8085 | **Instructions Part 2**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



### **Q23) Are CM and CP valid instruction names?**

- (a) CM is valid, CP invalid
- (b) CM is invalid, CP is valid
- (c) Both are invalid
- (d) Both are valid

**Answer...**

- (a) CM is valid, CP invalid
- (b) CM is invalid, CP is valid
- (c) Both are invalid
- (d) Both are valid**

Video Reference: 8085 | **Instruction Set Part 4**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q24) What is the outcome of this program...**

MVI A, 20H

MVI B, 30H

ADD B

PCHL

HLT

- (a) A = 50H
- (b) B = 50H
- (c) A = 20H
- (d) Can't say

**Answer...**

- (a) A = 50H
- (b) B = 50H
- (c) A = 20H
- (d) Can't say**



**Important Type**

Video Reference: 8085 | **Stacks and Subroutines**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q25) How many iterations will this loop have...**

MVI C, FFH

Back: ...

DCR C

JNZ Back

- (a) 255 Iterations
- (b) 256 Iterations
- (c) 257 Iterations
- (d) 1 Iteration



**Important Type  
of Question**

**Answer...**

- (a) 255 Iterations**
- (b) 256 Iterations
- (c) 257 Iterations
- (d) 1 Iteration

Video Reference: 8085 | **Instruction Set Part 4**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q26) How many iterations will this loop have...**

MVI C, 00H

Back: ...

DCR C

JNZ Back

- (a) 255 Iterations
- (b) 256 Iterations
- (c) 0 Iterations
- (d) 1 Iteration



**Important Type  
of Question**

**Answer...**

- (a) 255 Iterations
- (b) 256 Iterations**
- (c) 0 Iterations
- (d) 1 Iteration

Video Reference: 8085 | **Instruction Set Part 4**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q27) MVI A, 25H**

**CPI 24H**

**Show effect on CF and ZF...**

- (a) CF = 1, ZF= 0
- (b) CF = 0, ZF= 1
- (c) CF = 0, ZF= 0
- (d) CF = X, ZF= 0



**Answer...**

- (a) CF = 1, ZF= 0
- (b) CF = 0, ZF= 1
- (c) CF = 0, ZF= 0**
- (d) CF = X, ZF= 0

**Important Type  
of Question**

Video Reference: 8085 | **Programs**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q28) MVI A, 24H**

**CPI 25H**

**Show effect on CF and ZF...**

- (a) CF = 1, ZF= 0
- (b) CF = 0, ZF= 1
- (c) CF = 0, ZF= 0
- (d) CF = X, ZF= 0



**Answer...**

- (a) CF = 1, ZF= 0**
- (b) CF = 0, ZF= 1
- (c) CF = 0, ZF= 0
- (d) CF = X, ZF= 0

**Important Type  
of Question**

Video Reference: 8085 | **Programs**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q29) MVI A, 25H**

**CPI 25H**

**Show effect on CF and ZF...**

- (a) CF = 1, ZF= 0
- (b) CF = 0, ZF= 1
- (c) CF = 0, ZF= 0
- (d) CF = X, ZF= 0



**Answer...**

- (a) CF = 1, ZF= 0
- (b) CF = 0, ZF= 1**
- (c) CF = 0, ZF= 0
- (d) CF = X, ZF= 0

**Important Type  
of Question**

Video Reference: 8085 | **Programs**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q30) MVI A, 7FH**

**ADI 01H**

**Show effect on SF and CF...**

- (a) SF = 1, CF = 1
- (b) SF = 0, CF = 0
- (c) SF = 0, CF = 1
- (d) SF = 1, CF = 0



**Answer...**

- (a) SF = 1, CF = 1
- (b) SF = 0, CF = 0
- (c) SF = 0, CF = 1
- (d) SF = 1, CF = 0**

**Important Type  
of Question**

Video Reference: 8085 | **Programs**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Video Courses for You**

- Microprocessor | 8085
- Microprocessor | 8086
- Microprocessor | 80386
- Microprocessor | Pentium
- Microcontroller | 8051
- Microcontroller | ARM
- Computer Organisation & Architecture

**6 Months | Unlimited Views | 1199/-**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



## Section 3

### **8085 Timing Diagrams**



**Q31) Which is the biggest of the three...**

Instruction cycle, Machine Cycle or T-State

- (a) Instruction Cycle
- (b) Machine Cycle
- (c) T-state
- (d) All the same

**Answer...**

- (a) Instruction Cycle**
- (b) Machine Cycle
- (c) T-state
- (d) All the same

Video Reference: 8085 | **Timing Diagrams**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q32) Opcode Fetch has \_\_\_ T states?**

- (a) 4
- (b) 6
- (c) 4 or 6
- (d) undefined

**Answer...**

- (a) 4
- (b) 6
- (c) 4 or 6**
- (d) undefined

Video Reference: 8085 | **Timing Diagrams**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q33) Propagation delay between address and data  
is in \_\_\_\_\_ cycles?**

- (a) Read cycles
- (b) Write cycles
- (c) All cycles
- (d) Can't say



**Answer...**

- (a) **Read cycles**
- (b) Write cycles
- (c) All cycles
- (d) Can't say

**Important Type  
of Question**

Video Reference: 8085 | **Timing Diagrams**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q) We cheat and learn by piracy because...**

- (a) We are cheap
- (b) We are educated thieves
- (c) We were raised with low morals
- (d) We love to backstab our teacher!

**Answer...**

**All of the above**

Learn the full subject here...

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q34) Conditional Jump has \_\_\_ T-States**

- (a) 7 (True), 10 (False)
- (b) 7 (True), 13 (False)
- (c) 13 (True), 10 (False)
- (d) 10 (True), 7 (False)



**Answer...**

- (a) 7 (True), 10 (False)
- (b) 7 (True), 13 (False)
- (c) 13 (True), 10 (False)
- (d) 10 (True), 7 (False)**

**Important Type  
of Question**

Video Reference: 8085 | **Timing Diagrams**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q35) INR B has \_\_\_ T-States**

- (a) 4
- (b) 6
- (c) 7
- (d) 10



**Answer...**

- (a) 4**
- (b) 6
- (c) 7
- (d) 10

**Important Type  
of Question**

Video Reference: 8085 | **Timing Diagrams**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q36) INX B has \_\_\_ T-States**

- (a) 4
- (b) 6
- (c) 7
- (d) 10



**Answer...**

- (a) 4
- (b) 6**
- (c) 7
- (d) 10

**Important Type  
of Question**

Video Reference: 8085 | **Timing Diagrams**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q37) INR M has \_\_\_ T-States**

- (a) 4
- (b) 6
- (c) 7
- (d) 10



**Answer...**

- (a) 4
- (b) 6
- (c) 7
- (d) 10**

**Important Type  
of Question**

Video Reference: 8085 | **Timing Diagrams**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q38) Call 2000H has \_\_\_ T-States**

- (a) 10
- (b) 13
- (c) 16
- (d) 18



**Answer...**

- (a) 10
- (b) 13
- (c) 16
- (d) 18**

**Important Type  
of Question**

Video Reference: 8085 | **Timing Diagrams**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Video Courses for You**

- Microprocessor | 8085
- Microprocessor | 8086
- Microprocessor | 80386
- Microprocessor | Pentium
- Microcontroller | 8051
- Microcontroller | ARM
- Computer Organisation & Architecture

**6 Months | Unlimited Views | 1199/-**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



## Section 4

### **8085 Interrupts**



**Q39) 8085 has \_\_\_ hardware interrupts**

- (a) 2
- (b) 5
- (c) 13
- (d) 8

**Answer...**

- (a) 2
- (b) 5**
- (c) 13
- (d) 8

Video Reference: 8085 | **Interrupts**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

**Q40) The only pure edge triggered interrupt is...**

- (a) TRAP
- (b) RST 7.5
- (c) RST 6.5
- (d) RST 5.5
- (e) INTR



**Answer...**

- (a) TRAP
- (b) RST 7.5**
- (c) RST 6.5
- (d) RST 5.5
- (e) INTR

**Important Type**

Video Reference: 8085 | **Interrupts**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q41) The only edge/level triggered interrupt is...**

- (a) TRAP
- (b) RST 7.5
- (c) RST 6.5
- (d) RST 5.5
- (e) INTR

**Answer...**

- (a) TRAP**
- (b) RST 7.5
- (c) RST 6.5
- (d) RST 5.5
- (e) INTR

Video Reference: 8085 | **Interrupts**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q42) The highest priority interrupt is...**

- (a) TRAP
- (b) RST 7.5
- (c) RST 6.5
- (d) RST 5.5
- (e) INTR

**Answer...**

- (a) **TRAP**
- (b) RST 7.5
- (c) RST 6.5
- (d) RST 5.5
- (e) INTR

Video Reference: 8085 | **Interrupts**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q43) The only non-vectored interrupt is...**

- (a) TRAP
- (b) RST 7.5
- (c) RST 6.5
- (d) RST 5.5
- (e) INTR



**Answer...**

**Important Type**

- (a) TRAP
- (b) RST 7.5
- (c) RST 6.5
- (d) RST 5.5
- (e) INTR**

Video Reference: 8085 | **Interrupts**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q44) SIM can mask ...**

- (a) All interrupts
- (b) TRAP, RST 7.5, RST 6.5
- (c) RST 7.5, RST 6.5, RST 5.5
- (d) RST 7.5, RST 6.5, INTR
- (e) All except TRAP



**Answer...**

**Important Type**

- (a) All interrupts
- (b) TRAP, RST 7.5, RST 6.5
- (c) RST 7.5, RST 6.5, RST 5.5**
- (d) RST 7.5, RST 6.5, INTR
- (e) All except TRAP

Video Reference: 8085 | **Interrupts**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q45) SIM instruction is used to...**

- (a) Send Serial Data
- (b) Mask Interrupts
- (c) Reset Rst 7.5
- (d) All of the above



**Important Type**

**Answer...**

- (a) Send Serial Data
- (b) Mask Interrupts
- (c) Reset Rst 7.5
- (d) **All of the above**

Video Reference: 8085 | **Interrupts**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

**Q46) RIM is used to...**

- (a) Read Serial data
- (b) Find Masking Pattern
- (c) Find Pending Interrupts
- (d) All of the above



**Important Type**

**Answer...**

- (a) Read Serial data
- (b) Find Masking Pattern
- (c) Find Pending Interrupts
- (d) All of the above**

Video Reference: 8085 | **Interrupts**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

**Q47) INTA (bar) is given in response to INTR...**

- (a) For Handshaking
- (b) It's a Protocol
- (c) To obtain ISR address
- (d) To Start DMA



**Answer...**

- (a) For Handshaking
- (b) It's a Protocol
- (c) To obtain ISR address**
- (d) To Start DMA

**Important Type  
of Question**

Video Reference: 8085 | **Interrupts**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q48) MVI A, 40H,**

SIM

**The above program will...**

- (a) Send 0 on SOD and unmask interrupts
- (b) Send 0 on SOD
- (c) Unmask Interrupts
- (d) None of the above



**Important Type**

**Answer...**

- (a) Send 0 on SOD and unmask interrupts
- (b) Send 0 on SOD**
- (c) Unmask Interrupts
- (d) None of the above

Video Reference: 8085 | **Interrupts**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q49) The Vector address of RST 6.5 is...**

- (a) 0024H
- (b) 0026H
- (c) 0034H
- (d) 003CH

**Answer...**

- (a) 0024H
- (b) 0026H
- (c) 0034H**
- (d) 003CH



**Important Type  
of Question**

Video Reference: 8085 | **Interrupts**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



### **Q50) DI instruction will disable ....**

- (a) All h/w interrupts
- (b) All h/w interrupts except TRAP
- (c) All h/w interrupts except TRAP and INTR
- (d) It doesn't work

### **Answer...**

- (a) All h/w interrupts
- (b) All h/w interrupts except TRAP**
- (c) All h/w interrupts except TRAP and INTR
- (d) It doesn't work

Video Reference: 8085 | **Simulation Programs**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



### **Q51) On Reset which interrupts are disabled...**

- (a) All h/w interrupts
- (b) All h/w interrupts except TRAP
- (c) All h/w interrupts except TRAP and INTR
- (d) All h/w interrupts except RST 7.5

### **Answer...**

- (a) All h/w interrupts
- (b) All h/w interrupts except TRAP**
- (c) All h/w interrupts except TRAP and INTR
- (d) All h/w interrupts except RST 7.5

Video Reference: 8085 | **Interrupts**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Video Courses for You**

- Microprocessor | 8085
- Microprocessor | 8086
- Microprocessor | 80386
- Microprocessor | Pentium
- Microcontroller | 8051
- Microcontroller | ARM
- Computer Organisation & Architecture

**6 Months | Unlimited Views | 1199/-**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



## Section 5

### **Peripherals: 8259, 8255, etc**



**Q52) 8259 is used to...**

- (a) Increase the number of interrupts
- (b) Increase the power supply to interrupts
- (c) Perform DMA
- (d) Generate hardware delays

**Answer...**

- (a) Increase the number of interrupts**
- (b) Increase the power supply to interrupts
- (c) Perform DMA
- (d) Generate hardware delays

Video Reference: 8085 | **8259 PIC**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q53) Cascaded 8259 gives max \_\_\_ interrupts**

- (a) 9 interrupts
- (b) 18 interrupts
- (c) 64 interrupts
- (d) infinite interrupts

**Answer...**

- (a) 9 interrupts
- (b) 18 interrupts
- (c) 64 interrupts**
- (d) infinite interrupts

Video Reference: 8085 | **8259 PIC**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q54) EOI command in 8259 is given to...**

- (a) Give control of the bus back to 8085
- (b) Change trigger of interrupts
- (c) Mask Interrupts
- (d) Clear corresponding bit in In Service Reg

**Answer...**

- (a) Give control of the bus back to 8085
- (b) Change trigger of interrupts
- (c) Mask Interrupts
- (d) Clear corresponding bit in In Service Reg**

Video Reference: 8085 | **8259 PIC**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q55) To give Poll Command we must give...**

- (a) OCW1
- (b) OCW2
- (c) OCW3
- (d) OCW4

**Answer...**

- (a) OCW1
- (b) OCW2
- (c) OCW3**
- (d) OCW4



**Important Type  
of Question**

Video Reference: 8085 | **8259 PIC | ICW and OCW**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q56) To enter/ exit SMM we must give...**

- (a) OCW1
- (b) OCW2
- (c) OCW3
- (d) OCW4



**Answer...**

- (a) OCW1
- (b) OCW2
- (c) OCW3**
- (d) OCW4

**Important Type  
of Question**

Video Reference: 8085 | **8259 PIC | ICW and OCW**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q57) To mask/ unmask interrupts we give...**

- (a) OCW1
- (b) OCW2
- (c) OCW3
- (d) OCW4

**Answer...**

- (a) OCW1
- (b) OCW2
- (c) OCW3
- (d) OCW4



**Important Type  
of Question**

Video Reference: 8085 | **8259 PIC | ICW and OCW**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q58) To decide vector numbers we give...**

- (a) ICW1
- (b) ICW2
- (c) ICW3
- (d) ICW4

**Answer...**

- (a) ICW1
- (b) ICW2**
- (c) ICW3
- (d) ICW4



**Important Type  
of Question**

Video Reference: 8085 | **8259 PIC | ICW and OCW**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q59) To configure cascading we give...**

- (a) ICW1
- (b) ICW2
- (c) ICW3
- (d) ICW4

**Answer...**

- (a) ICW1
- (b) ICW2
- (c) ICW3**
- (d) ICW4



**Important Type  
of Question**

Video Reference: 8085 | **8259 PIC | ICW and OCW**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q60) 8255 is used for...**

- (a) Increase the number of interrupts
- (b) Reliable data transfers by handshaking
- (c) Perform DMA
- (d) Generate hardware delays

**Answer...**

- (a) Increase the number of interrupts
- (b) Reliable data transfers by handshaking**
- (c) Perform DMA
- (d) Generate hardware delays

Video Reference: 8085 | **8255 PPI**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q61) 8255 BSR command is for...**

- (a) Set and reset bits of Port A
- (b) Set and reset bits of Port A
- (c) Set and reset bits of Port C
- (d) No such command

**Answer...**

- (a) Set and reset bits of Port A
- (b) Set and reset bits of Port A
- (c) Set and reset bits of Port C**
- (d) No such command

Video Reference: 8085 | **8255 PPI**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q62) 8255 I/O command is for...**

- (a) Decide mode and direction of ports
- (b) Give vector numbers
- (c) Decide priority of ports
- (d) Use alternate functions of ports

**Answer...**

- (a) Decide mode and direction of ports**
- (b) Give vector numbers
- (c) Decide priority of ports
- (d) Use alternate functions of ports

Video Reference: 8085 | **8255 PPI**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q63) 8255 Mode 0 is for...**

- (a) Port A
- (b) Port B
- (c) Port C
- (d) All of the above



**Answer...**

- (a) Port A
- (b) Port B
- (c) Port C
- (d) All of the above**

**Important Type  
of Question**

Video Reference: 8085 | **8255 PPI**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q64) 8255 Mode 1 is for...**

- (a) Port A, B
- (b) Port B, C
- (c) Port A only
- (d) None of the above



**Answer...**

- (a) Port A, B**
- (b) Port B, C
- (c) Port A only
- (d) None of the above

**Important Type  
of Question**

Video Reference: 8085 | **8255 PPI**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q65) 8255 Mode 2 is for...**

- (a) Port A, B
- (b) Port B, C
- (c) Port A only
- (d) None of the above

**Answer...**

- (a) Port A, B
- (b) Port B, C
- (c) Port A only**
- (d) None of the above



**Important Type  
of Question**

Video Reference: 8085 | **8255 PPI**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q66) Give BSR Command to Send 1 on PC3**

- (a) 03H
- (b) 07H
- (c) 83H
- (d) 87H

**Answer...**

- (a) 03H
- (b) 07H**
- (c) 83H
- (d) 87H



**Important Type  
of Question**

Video Reference: 8085 | **8255 PPI**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q67) Input handshake signals are...**

- (a) STB, IBF, READY
- (b) OBF, ACK INTR
- (c) STB, IBF, HOLD
- (d) STB, IBF, INTR

**Answer...**

- (a) STB, IBF, READY
- (b) OBF, ACK INTR
- (c) STB, IBF, HOLD
- (d) **STB, IBF, INTR**



**Important Type  
of Question**

Video Reference: 8085 | **8255 PPI**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q68) Output handshake signals are...**

- (a) STB, IBF, READY
- (b) OBF, ACK INTR
- (c) STB, IBF, HOLD
- (d) STB, IBF, INTR



**Answer...**

- (a) STB, IBF, READY
- (b) OBF, ACK INTR**
- (c) STB, IBF, HOLD
- (d) STB, IBF, INTR

**Important Type  
of Question**

Video Reference: 8085 | **8255 PPI**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q69) 8237/8257 is used for...**

- (a) Increase the number of interrupts
- (b) Reliable data transfers by handshaking
- (c) Perform DMA
- (d) Generate hardware delays

**Answer...**

- (a) Increase the number of interrupts
- (b) Reliable data transfers by handshaking
- (c) Perform DMA**
- (d) Generate hardware delays

Video Reference: 8085 | **8237/57 DMAC**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q70) 8237/8257 is how many channels...**

- (a) 4 channels
- (b) 16 channels
- (c) 2 channels
- (d) 8 interrupts

**Answer...**

- (a) 4 channels**
- (b) 16 channels
- (c) 2 channels
- (d) 8 interrupts

Video Reference: 8085 | **8237/57 DMAC**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q71) 8253/ 8254 is used for...**

- (a) Increase the number of interrupts
- (b) Reliable data transfers by handshaking
- (c) Perform DMA
- (d) Generate hardware delays

**Answer...**

- (a) Increase the number of interrupts
- (b) Reliable data transfers by handshaking
- (c) Perform DMA
- (d) Generate hardware delays**

Video Reference: 8085 | **8253/ 54 PIT**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q72) 8253/ 8254 has how many counters...**

- (a) 4, 16-bit down counters
- (b) 2, 16-bit down counters
- (c) 3, 16-bit up counters
- (d) 3, 16-bit down counters
- (e) 4, 16-bit down counters

**Answer...**

- (a) 4, 16-bit down counters
- (b) 2, 16-bit down counters
- (c) 3, 16-bit up counters
- (d) 3, 16-bit down counters**
- (e) 4, 16-bit down counters

Video Reference: 8085 | **8253/ 54 PIT**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q73) 8253/ 8254 Mode 0 is called...**

- (a) Interrupt on TC
- (b) Monostable Multivibrator
- (c) Rate Generator
- (d) Square Wave Generator
- (e) Software Triggered Strobe

**Answer...**

- (a) Interrupt on TC**
- (b) Monostable Multivibrator
- (c) Rate Generator
- (d) Square Wave Generator
- (e) Software Triggered Strobe

Video Reference: 8085 | **8253/ 54 PIT**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q74) 8253/ 8254 Mode 3 is called...**

- (a) Interrupt on TC
- (b) Monostable Multivibrator
- (c) Rate Generator
- (d) Square Wave Generator
- (e) Software Triggered Strobe

**Answer...**

- (a) Interrupt on TC
- (b) Monostable Multivibrator
- (c) Rate Generator
- (d) Square Wave Generator**
- (e) Software Triggered Strobe

Video Reference: 8085 | **8253/ 54 PIT**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Video Courses for You**

- Microprocessor | 8085
- Microprocessor | 8086
- Microprocessor | 80386
- Microprocessor | Pentium
- Microcontroller | 8051
- Microcontroller | ARM
- Computer Organisation & Architecture

**6 Months | Unlimited Views | 1199/-**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



## Section 5

### **8085 Pins and Interfacing**



**Q75) Ready signal is used to...**

- (a) Increase the number of interrupts
- (b) Synchronize with slow devices
- (c) Perform DMA
- (d) Generate hardware delays

**Answer...**

- (a) Increase the number of interrupts
- (b) Synchronize with slow devices**
- (c) Perform DMA
- (d) Generate hardware delays

Video Reference: 8085 | **Pin Diagram**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q76) Hold and HLDA are used to...**

- (a) Increase the number of interrupts
- (b) Synchronize with slow devices
- (c) Perform DMA
- (d) Generate hardware delays

**Answer...**

- (a) Increase the number of interrupts
- (b) Synchronize with slow devices
- (c) Perform DMA**
- (d) Generate hardware delays

Video Reference: 8085 | **Pin Diagram**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q77) ALE is used to...**

- (a) Increase the number of interrupts
- (b) Synchronize with slow devices
- (c) Perform DMA
- (d) Latch address from the multiplexed bus



**Important Type**

**Answer...**

- (a) Increase number of interrupts
- (b) Synchronize with slow devices
- (c) Perform DMA
- (d) Latch address from the multiplexed bus**

Video Reference: 8085 | **Designing**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



### **Q78) Crystal frequency is divided by 2 to...**

- (a) Make it even
- (b) To Rectify the wave
- (c) For noise reduction
- (d) To generate 50% duty cycle

### **Answer...**

- (a) Make it even
- (b) To Rectify the wave
- (c) For noise reduction
- (d) To generate 50% duty cycle**

Video Reference: 8085 | **Pin Diagram**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q79) In I/O mapped I/O, I/O address is..**

- (a) 8-bit
- (b) 16-bit
- (c) 256-bit
- (d) undefined



**Answer...**

- (a) 8-bit**
- (b) 16-bit
- (c) 256-bit
- (d) undefined

**Important Type**

Video Reference: 8085 | **Designing**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q79) In Mem mapped I/O, I/O address is..**

- (a) 8-bit
- (b) 16-bit
- (c) 256-bit
- (d) undefined

**Answer...**

- (a) 8-bit
- (b) 16-bit**
- (c) 256-bit
- (d) undefined



**Important Type**

Video Reference: 8085 | **Designing**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q80) Chip number 6264 is of size...**

- (a) 64 bit
- (b) 64 KB
- (c) 8 KB
- (d) insufficient information



**Answer...**

**Important Type**

- (a) 64 bit
- (b) 64 KB
- (c) 8 KB**
- (d) insufficient information

Video Reference: 8085 | **Designing**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q81) Chip number 17128 is of size...**

- (a) 64 bit
- (b) 64 KB
- (c) 8 KB
- (d) 16 KB



**Answer...**

- (a) 64 bit
- (b) 64 KB
- (c) 8 KB
- (d) 16 KB**

**Important Type**

Video Reference: 8085 | **Designing**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q82) (4K x 8) means...**

- (a) 8 chips of 4B
- (b) 4KB
- (c) 32KB
- (d) Wrong notation



**Answer...**

- (a) 8 chips of 4B
- (b) 4KB**
- (c) 32KB
- (d) Wrong notation

**Important Type**

Video Reference: 8085 | **Designing**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q83) 8KB chip needs ...**

- (a) 10 address lines
- (b) 11 address lines
- (c) 12 address lines
- (d) 13 address lines



**Answer...**

- (a) 10 address lines
- (b) 11 address lines
- (c) 12 address lines
- (d) **13 address lines**

**Important Type**

Video Reference: 8085 | **Designing**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q84) IN and OUT instructions are useful in...**

- (a) I/O Mapped I/O
- (b) Memory Mapped I/O
- (c) Both of them
- (d) Invalid instructions



**Answer...**

**Important Type**

- (a) I/O Mapped I/O**
- (b) Memory Mapped I/O
- (c) Both of them
- (d) Invalid instructions

Video Reference: 8085 | **Designing**  
[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



### Q85) What is mapped at RESET Vector address

- (a) RAM
- (b) ROM
- (c) Doesn't matter
- (d) There is no such address



Important Type

Answer...

- (a) RAM
- (b) ROM**
- (c) Doesn't matter
- (d) There is no such address

Video Reference: 8085 | **Designing**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Video Courses for You**

- Microprocessor | 8085
- Microprocessor | 8086
- Microprocessor | 80386
- Microprocessor | Pentium
- Microcontroller | 8051
- Microcontroller | ARM
- Computer Organisation & Architecture

**6 Months | Unlimited Views | 1199/-**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

### **Q86) Identify the invalid instruction**

- (a) LDAX B
- (b) LDAX D
- (c) LDAX H
- (d) All are valid

Important Type

### **Answer...**

- (a) LDAX B
- (b) LDAX D
- (c) LDAX H**
- (d) All are valid

Video Reference: 8085 | **Programming**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q87) Which instruction does not affect the stack?**

- (a) CALL
- (b) RET
- (c) XTHL
- (d) XCHG



**Answer...**

- (a) CALL
- (b) RET
- (c) XTHL
- (d) XCHG**

Video Reference: 8085 | **Stacks and Subroutines**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



### Q88) When ALE = 1

- (a) AD0-AD7 carries data
- (b) AD0-AD7 carries address
- (c) AD0-AD7 is in tri-state
- (d) AD0-AD7 is in Hold State

Important Type



### Answer...

- (a) AD0-AD7 carries data
- (b) AD0-AD7 carries address**
- (c) AD0-AD7 is in tri-state
- (d) AD0-AD7 is in Hold State

Video Reference: 8085 | Pin Diagram

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q89) 8085 enters “wait” state when**

- (a) READY = 1
- (b) READY = 0
- (c) HOLD = 1
- (d) HOLD = 0



**Answer...**

- (a) READY = 1
- (b) READY = 0**
- (c) HOLD = 1
- (d) HOLD = 0

Video Reference: 8085 | Pin Diagram

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q90) HOLD signal is used...**

- (a) To request an Interrupt to the processor
- (b) To request for Wait states for slow devices
- (c) To request for control of the system bus
- (d) To request for longer data cycles



**Answer...**

- (a) To request an Interrupt to the processor
- (b) To request for Wait states for slow devices
- (c) To request for control of the system bus**
- (d) To request for longer data cycles

Video Reference: 8085 | **Pin Diagram**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



### **Q91) Address Data Multiplexing is done to...**

- (a) To reduce the number of pins
- (b) To reduce interrupt latency
- (c) To reduce wait states due to slow devices
- (d) To reduce the size of memory

Important Type



**Answer...**

- (a) To reduce the number of pins**
- (b) To reduce interrupt latency
- (c) To reduce wait states due to slow devices
- (d) To reduce the size of memory

Video Reference: 8085 | Pin Diagram

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q) Should you learn by cheating / piracy?**

- (a) Yes, I am raised like this!
- (b) Who cares about the teacher's effort? I am cheap!
- (c) Education by cheating will still help me!
- (d) NO. We should support good teachers who work so hard yet charge very less!

**Answer...**

I leave this answer to YOU!



**Video Courses for You**

- Microprocessor | 8085
- Microprocessor | 8086
- Microprocessor | 80386
- Microprocessor | Pentium
- Microcontroller | 8051
- Microcontroller | ARM
- Computer Organisation & Architecture

**6 Months | Unlimited Views | 1199/-**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q92) Which is a VALID instruction...**

- (a) PUSH BC
- (b) PUSH PC
- (c) PUSH M
- (d) PUSH PSW



**Answer...**

- (a) PUSH BC
- (b) PUSH PC
- (c) PUSH M
- (d) PUSH PSW**

Video Reference: 8085 | Instruction Set | Stack operations

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q93) Which is an INVALID instruction...**

- (a) JC 2000H
- (b) CC 2000H
- (c) RC 2000H
- (d) All are valid



**Answer...**

- (a) JC 2000H
- (b) CC 2000H
- (c) RC 2000H**
- (d) All are valid

Video Reference: 8085 | **Instruction Set | Branch operations**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

**Q94) Which is NOT a BRANCH operation...**

- (a) JMP 2000H
- (b) RSTn
- (c) SPHL
- (d) PCHL

Important Type



**Answer...**

- (a) JMP 2000H
- (b) RSTn
- (c) SPHL**
- (d) PCHL

Video Reference: 8085 | **Instruction Set | Branch operations**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

### **Q95) Which is an INVALID instruction**

- (a) DAD B
- (b) DAD D
- (c) DAD H
- (d) DAD M



**Answer...**

- (a) DAD B
- (b) DAD D
- (c) DAD H
- (d) DAD M**

Video Reference: 8085 | **Instruction Set | Arithmetic group**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q96) Which executes the slowest**

- (a) INR M
- (b) INR B
- (c) INX B
- (d) All take the same time



**Answer...**

- (a) INR M**
- (b) INR B
- (c) INX B
- (d) All take the same time

Video Reference: 8085 | **Timing diagrams**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

### **Q97) Which instruction has a 4T Opcode fetch**

- (a) Conditional Jump
- (b) Conditional Call
- (c) Conditional Return
- (d) Unconditional Call

Important Type



**Answer...**

- (a) Conditional Jump**
- (b) Conditional Call
- (c) Conditional Return
- (d) Unconditional Call

Video Reference: 8085 | **Timing diagrams**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

**Q98) Which instruction has a 6T Opcode fetch**

- (a) RET
- (b) XTHL
- (c) RC
- (d) XCHG



**Answer...**

- (a) RET
- (b) XTHL
- (c) RC**
- (d) XCHG

Video Reference: 8085 | **Timing diagrams**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)

**Q99) Which instruction takes the most T-States**

- (a) ADD B
- (b) ADC B
- (c) ADD M
- (d) INR M



**Answer...**

- (a) ADD B
- (b) ADC B
- (c) ADD M
- (d) INR M**

Video Reference: 8085 | **Timing diagrams**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



**Q100) The machine cycles for OUT 80H are..**

- (a) Opcode fetch, IO Read
- (b) Opcode fetch, IO Write, IO Write
- (c) Opcode fetch, Mem Read, IO Read
- (d) Opcode fetch, Mem Read, IO Write

Important Type



**Answer...**

- (a) Opcode fetch, IO Read
- (b) Opcode fetch, IO Write, IO Write
- (c) Opcode fetch, Mem Read, IO Read
- (d) **Opcode fetch, Mem Read, IO Write**

Video Reference: 8085 | **Timing diagrams**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



### Q101) In I/O Machine cycles, the 8bit IO Address...

- (a) Appears on A0-A7 only
- (b) Appears on A8-A15 only
- (c) Appears on A0-A7 and A8-A15 both
- (d) Doesn't appear because its implied



### Answer...

- (a) Appears on A0-A7 only
- (b) Appears on A8-A15 only
- (c) Appears on A0-A7 and A8-A15 both**
- (d) Doesn't appear because its implied

Video Reference: 8085 | **Timing diagrams**

[www.BharatAcharyaEducation.com](http://www.BharatAcharyaEducation.com)



## Useful Links...

My Recommended  
Reference books

---

**8085 Microprocessor** | Ramesh Gaonkar

Link: <https://amzn.to/2VtJTOz>

**8086 Microprocessor** | Douglas Hall

Link: <https://amzn.to/3lyFOTC>

**8086 Microprocessor** | Liu Gibson

Link: <https://amzn.to/36yFCzG>

**8086 Microprocessor** | Barry Brey

Link: <https://amzn.to/37czFc3>

**8086 Microprocessor** | Kenneth Ayala

Link: <https://amzn.to/33DJAFn>

**8051 Microcontroller** | Bharat Acharya

Link: <https://amzn.to/3obYQRs>

**8051 Microcontroller** | Mazidi

Link: <https://amzn.to/3lyKxVA>

**COA** | William Stallings

Link: <https://amzn.to/33CuQXb>

**COA** | John Hayes

Link: <https://amzn.to/2VujTlV>

**COA** | Tanenbaum

Link: <https://amzn.to/3mM8MRe>



**Raspberry Pi Book | Raspberry Pi Made Easy**

Link: <https://amzn.to/3acdDII>

**Raspberry Pi Board | Pi3-MODB-1GB Motherboard**

Link: <https://amzn.to/3m9xzO5>

**Arduino Book | 26 Basic Arduino Projects**

Link: <https://amzn.to/3mgep9r>

**Arduino Board | T9-NHXR-186H Uno R3**

Link: <https://amzn.to/344cnCU>

**English Speaking Book | Effortless English**

Link: <https://amzn.to/3gFCVzH>

**English Speaking Book | How to Talk to Anyone**

Link: <https://amzn.to/349cugB>

**Personality Development | The Art of Public Speaking**

Link: <https://amzn.to/3aj0xsT>

**Aptitude Test | RS Aggarwal**

Link: <https://amzn.to/2Waq4vO>

**Motivational book | The 7 Habits of Highly Effective People**

Link: <https://amzn.to/3ngomou>

**Motivational book | Think and Grow Rich**

Link: <https://amzn.to/3a9jNbY>



### Our Social Media

---

**Instagram:** <https://www.instagram.com/bharatacharya.in/>

**Facebook:** <https://www.facebook.com/BharatAcharyaEducation>

**Youtube:** <http://www.youtube.com/c/bharatacharyaeducation>

### Contact Us

---

**Call or WhatsApp:** +919820408217

**Email:** [bharatsir@hotmail.com](mailto:bharatsir@hotmail.com)

**Website:** <https://www.bharatacharyaeducation.com>

#bharatacharyaeducation

#bharatacharya