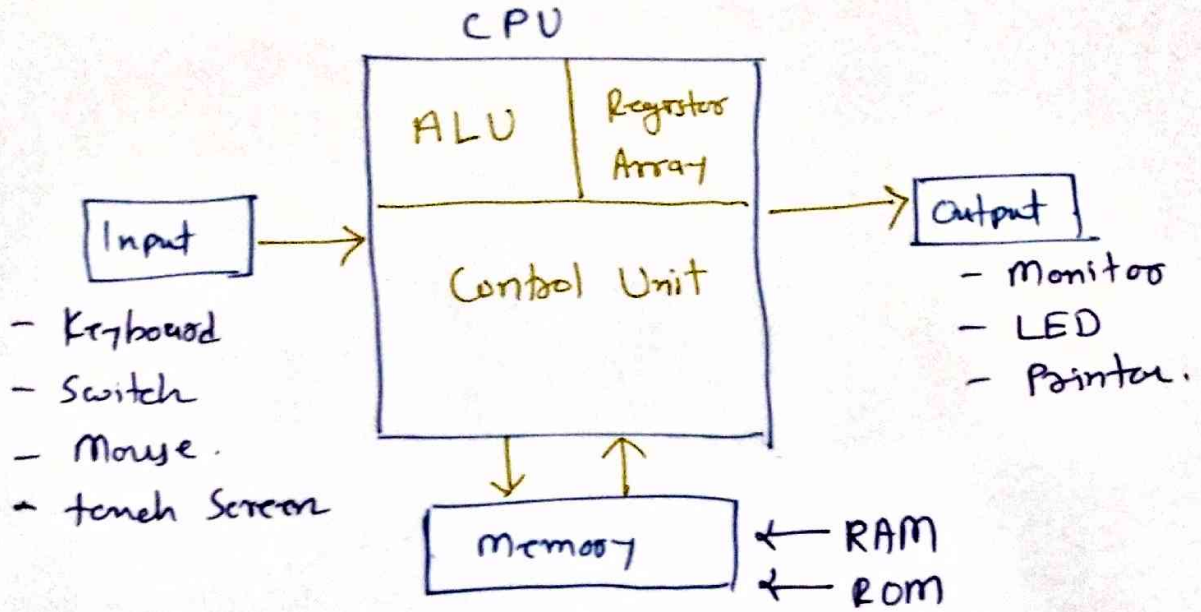
* Digital Computers

- Digital computers having a microprocessor as its CPU [Central Processing Unit].
- CPU combined with memory & I/O devices forms a Digital Computer.



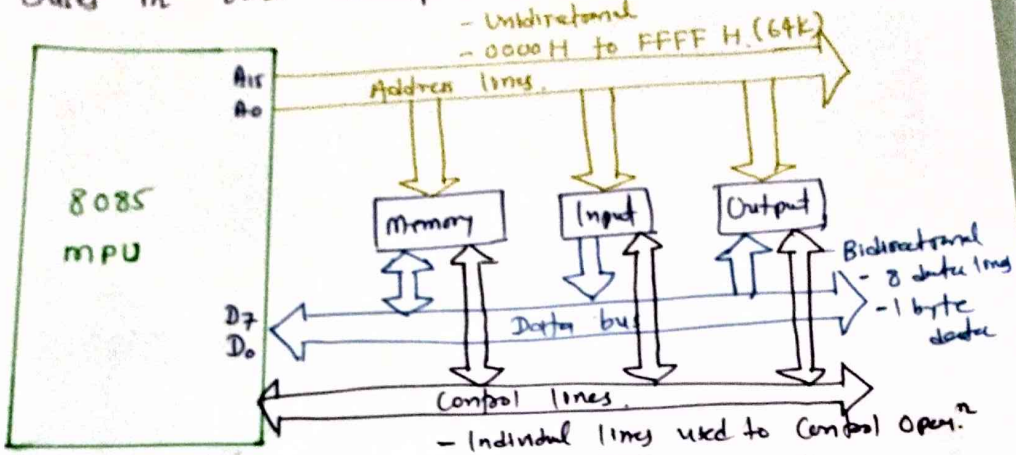
Youtube
↓
Engineering Junda

* Microprocessor basics & Intel series.

- Microprocessor is a Central Processing Unit CPU
- It is having following blocks.
  - ALU (Arithmetic & Logic Unit)
  - Register Array
  - Control Unit
- It is made up of transistors, resistors & diodes in a single Integrated Chip.

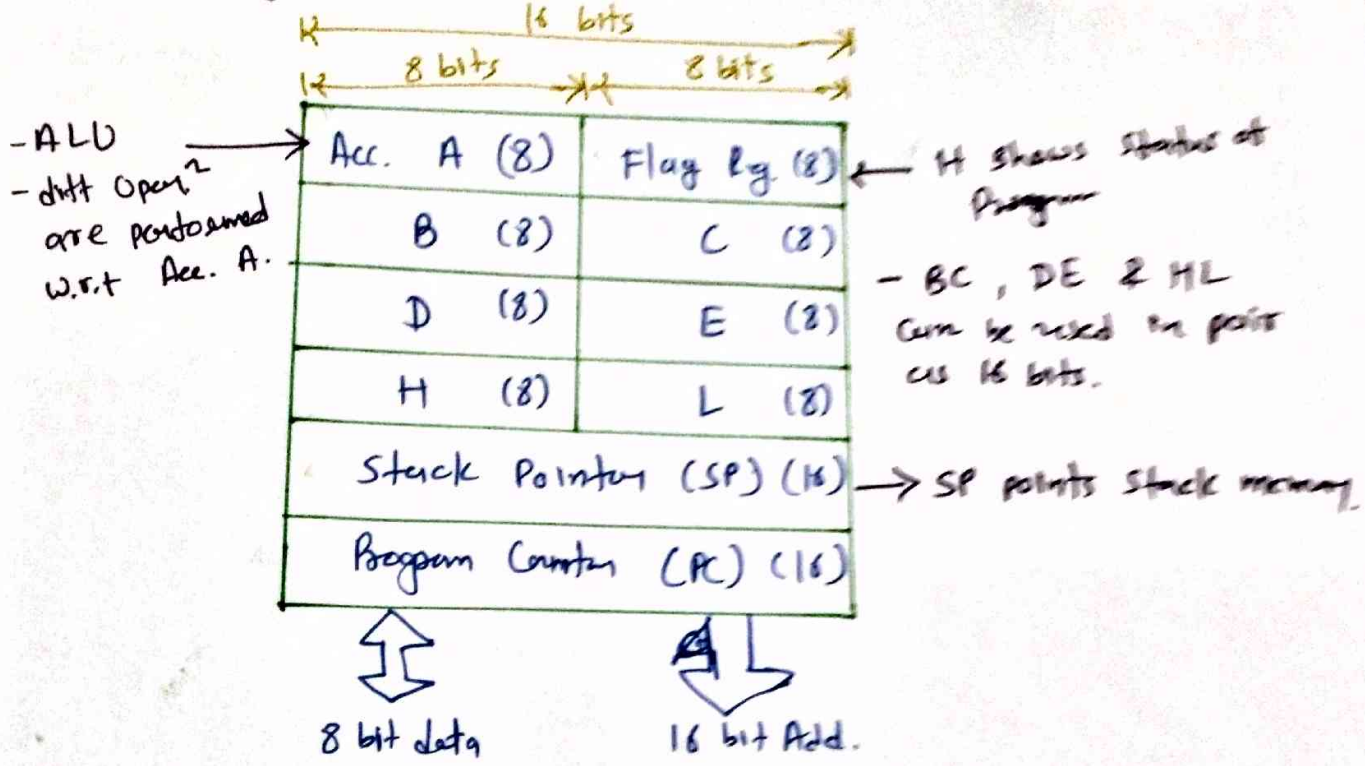
YouTube – Engineering funda

Buses in 8085 Microprocessor



YouTube - Engineering Sunda

Programming Model of 8085



Flag Register of 8085

S Z Ac P Cy

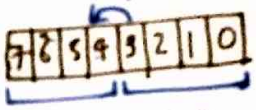
- It shows status of Program
- In 8085, Flag register is of 8 bits.
- There are 5 flag bits in flag register of 8085.

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

S - Sign flag - MSB of ALU (Accumulator A)

Z - Zero flag - Operation of Acc. A = 00H

AC - Auxiliary Carry - Nibble to Nibble carry in Operation [D_3 to D_4 carry]



P - Parity flag - Even Parity of ALU (Acc. A)

Cy - Carry flag - Carry in Operation of ALU (Acc. A).

ADD H

H = B C

A = D D

A = 9 9

Ac = 1

Cy = 1

12 + 13 = 25

1 1 1 1 1
1 0 1 1 1 1 0 0
1 1 0 1 1 1 0 1

1 0 0 1 1 0 0 1 ≠ 00H

Z = 0

MSB (S = 1)

four 1's → P = 1

Control signals and status signals in 8085

YouTube → Engineering funda

ALE – Address Latch Enable
- It is active high signal
- It used to demultiplex $AD_0 - AD_7$ lines.
- If ALE = 1 → $AD_0 - AD_7 = A_0 - A_7$
  ALE = 0 → $AD_0 - AD_7 = D_0 - D_7$

$\overline{RD}$ – Read data
- It is active low signal.
- If $\overline{RD} = 0$ → MPU performs Read Operⁿ

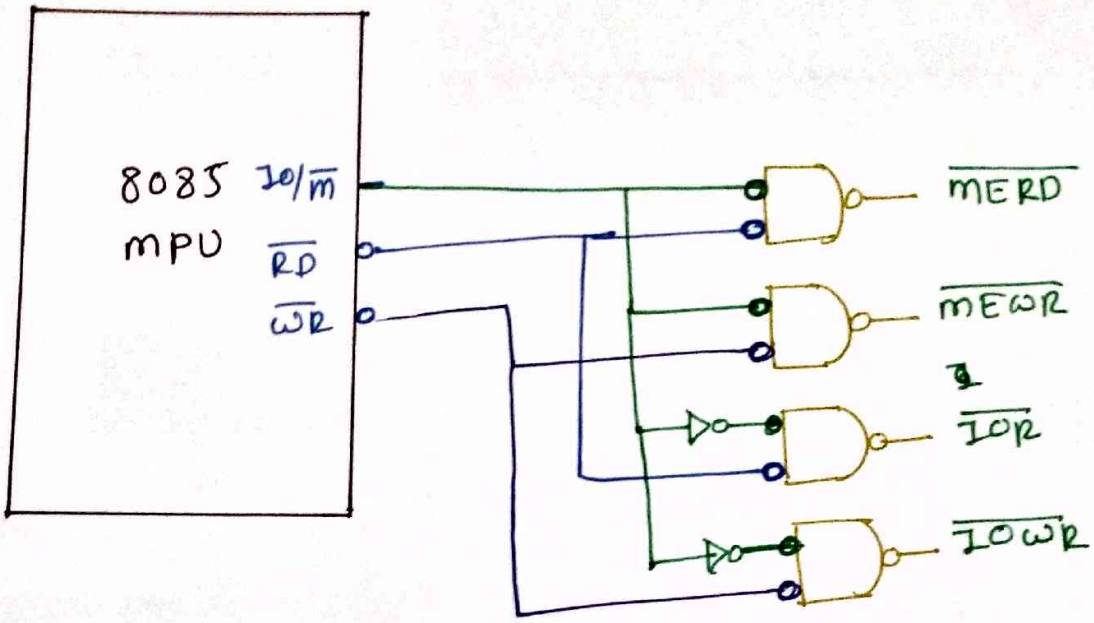
$\overline{WR}$ – Write data
- It is active low signal.
- If $\overline{WR} = 0$ → MPU Performs write Operⁿ

IO/$\overline{M}$ – Input output / Memory
- It is used to differentiate IO & memory devices.
- IO/$\overline{M}$ = 1 → IO Operⁿ
- IO/$\overline{M}$ = 0 → Memory Operⁿ

S_0 & S_1 – These are Status signal.

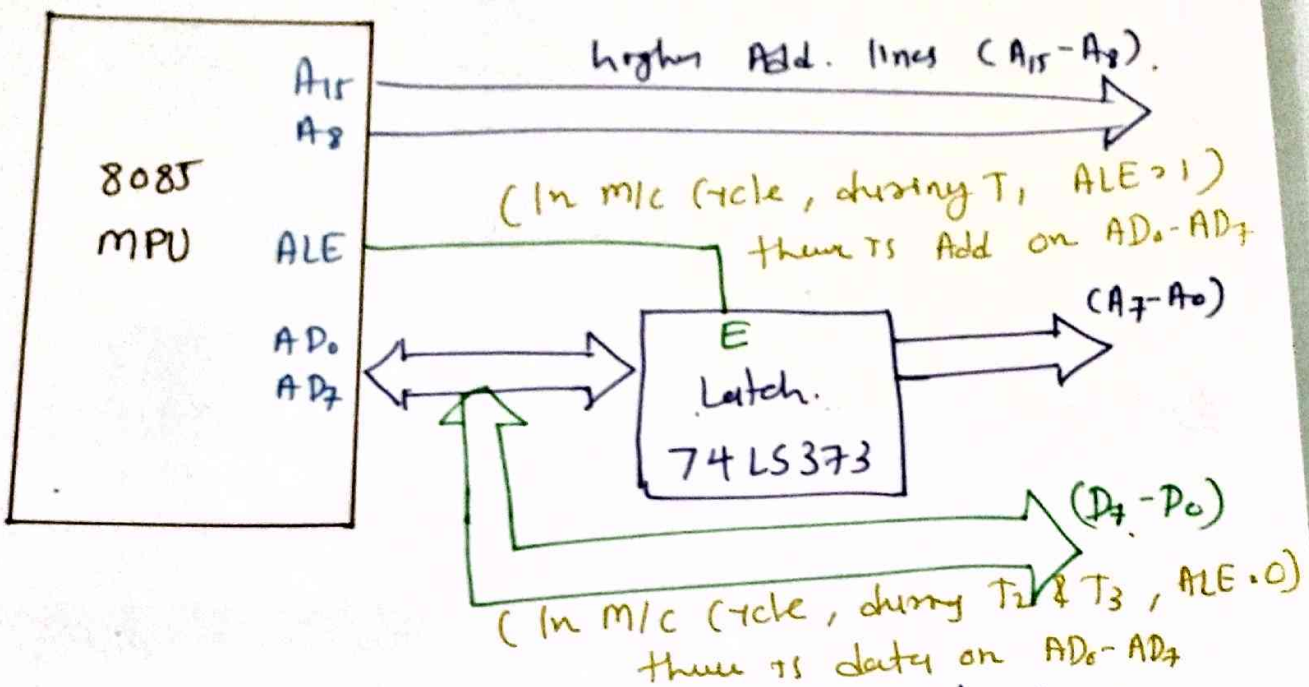
	IO/$\overline{M}$	S_1	S_0	
Opcode Fetch	0	1	1	$\overline{RD} = 0$
Mem. Read	0	1	0	$\overline{RD} = 0$
Mem. write	0	0	1	$\overline{WR} = 0$
IO Read	1	1	0	$\overline{RD} = 0$
IO write	1	0	1	$\overline{WR} = 0$
Interrupt Ack.	1	1	1	$\overline{INTA} = 0$
HALT	Z	0	0	$\overline{RD}, \overline{WR} = Z$
HOLD	Z	X	X	$\overline{INTA} = 1$
RESET	Z	X	X	

Generation of Control signals in 8085



YouTube - Engineering funda

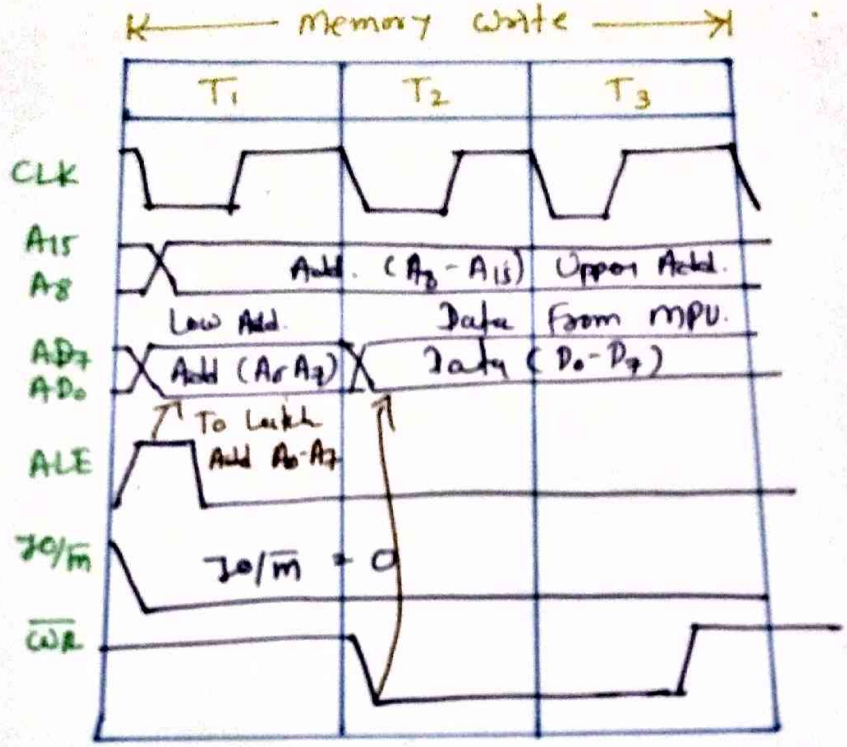
Address Data Demultiplexing in 8085



- AD_0 to AD_7 is time multiplex Add - data lines.
- to Interface 8085 with mem., Input & output devices, it is compulsory to demultiplex Add & data.

YouTube - Engineering funda

Memory write timing Diagram & working in 8085



- MVI A, 32H

Opcode Fetch Mem. Read.

T1	T2	T3	T4	T1	T2	T3

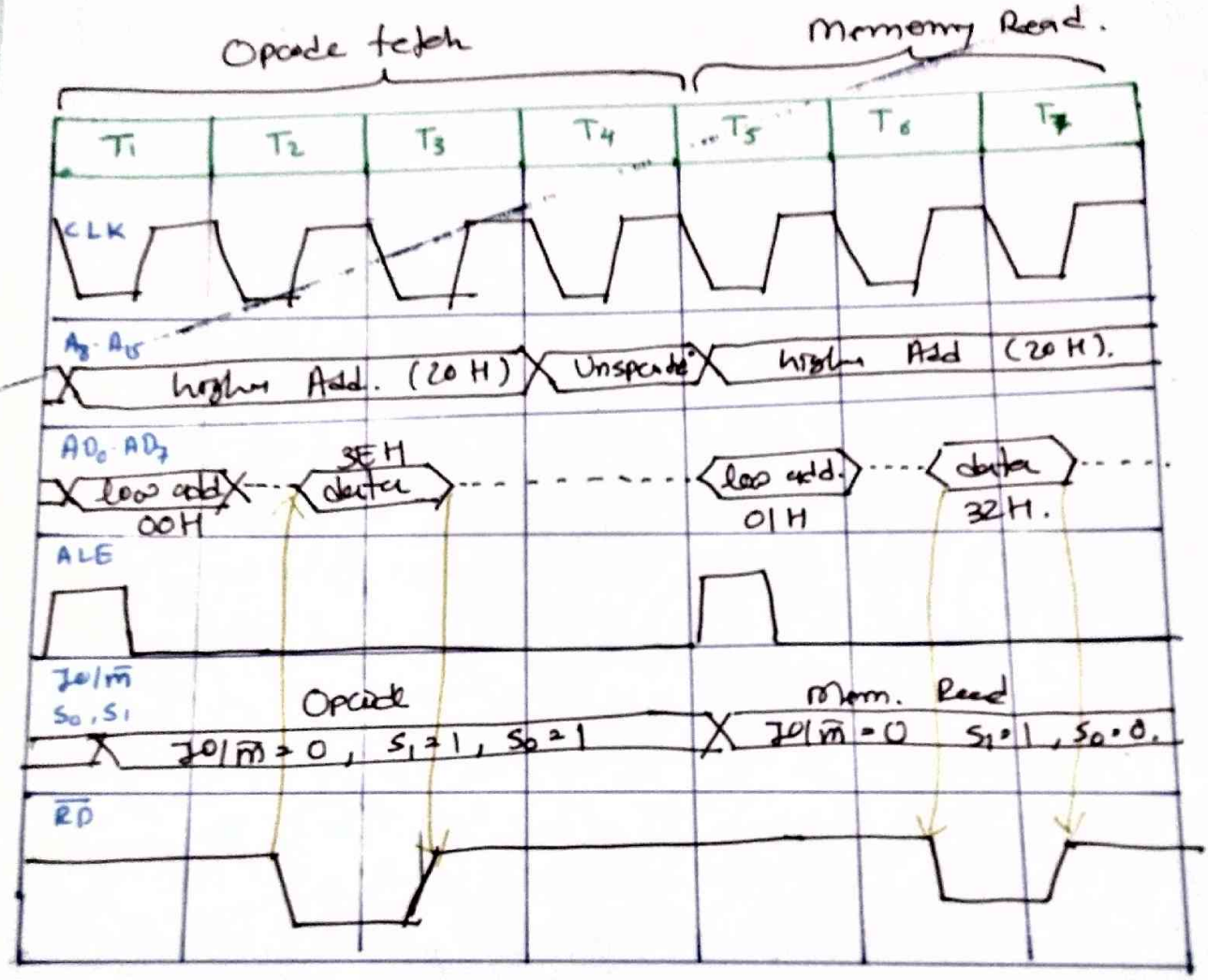
YouTube - Engineering funda

Timing Diagram of MVI Instruction

- MVI A, 32H

Memory Location	data
2000 H	3E H. (opcode)
2001 H	32 H. (mem. Rd).

– Opcode fetch, Mem. Read



YouTube - Engineering funda

Addressing modes in 8085

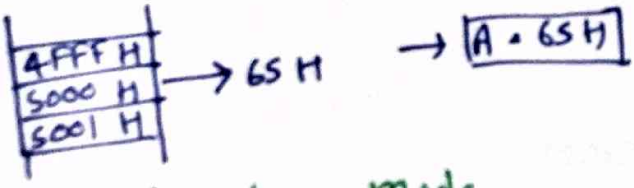
- There are five addressing modes in 8085.
  1. Immediate addressing mode
  2. Direct addressing mode
  3. Indirect addressing mode
  4. Register addressing mode
  5. Implied Addressing mode

- Immediate Addressing mode
- In this mode, the 8/16 bit data is specified in the instruction itself as one of its Operand.
- e.g. MVI A, 2F H

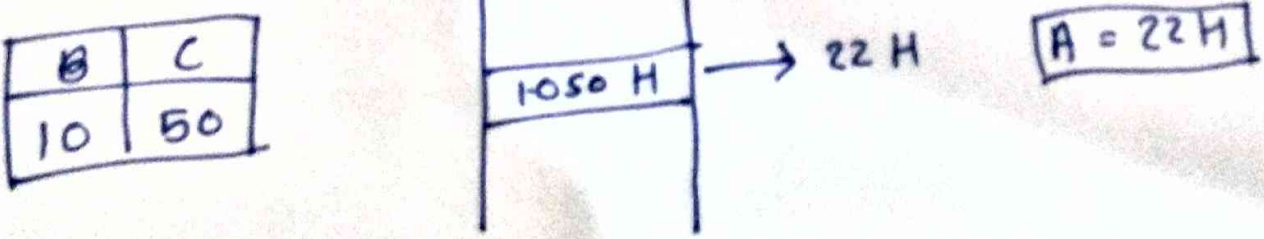
A = 2F H

- In this mode we directly transfer data into register.

- Direct Addressing mode
- In this mode, data is directly copied from given add. to the register.
- e.g. LDA 5000 H.



- Indirect Addressing mode
- In this mode, data is transfered from memory Location pointed by register.
- e.g. LDAX B



YouTube - Engineering funda

Register Addressing Mode

- In this mode, we copy data from one register to other register.

e.g. MOV A, B

before execution
A = 10 H
B = 20 H
→ After execution
A = 20 H
B = 20 H

Implied Addressing Mode

- In this mode, we don't need any operand.
- data execution will happen with that register only.

e.g. cmp A
A = CC H

A = CC H 1100 1100
↓ 1's comp.
0011 0011
↓
A = 33 H

YouTube – Engineering funda

Memory mapped IO and IO mapped IO in 8085

Memory mapped IO

- IO is treated as memory
- 16 bit Addressing ($A_0 - A_{15}$)
- It can Address = 2^{16}
  = 64K
- No of devices = 65536
- More decoder hardware
- Available memory is less
- $\overline{MERD}$ & $\overline{MEWR}$ control signal with IO.
- Instruction e.g.

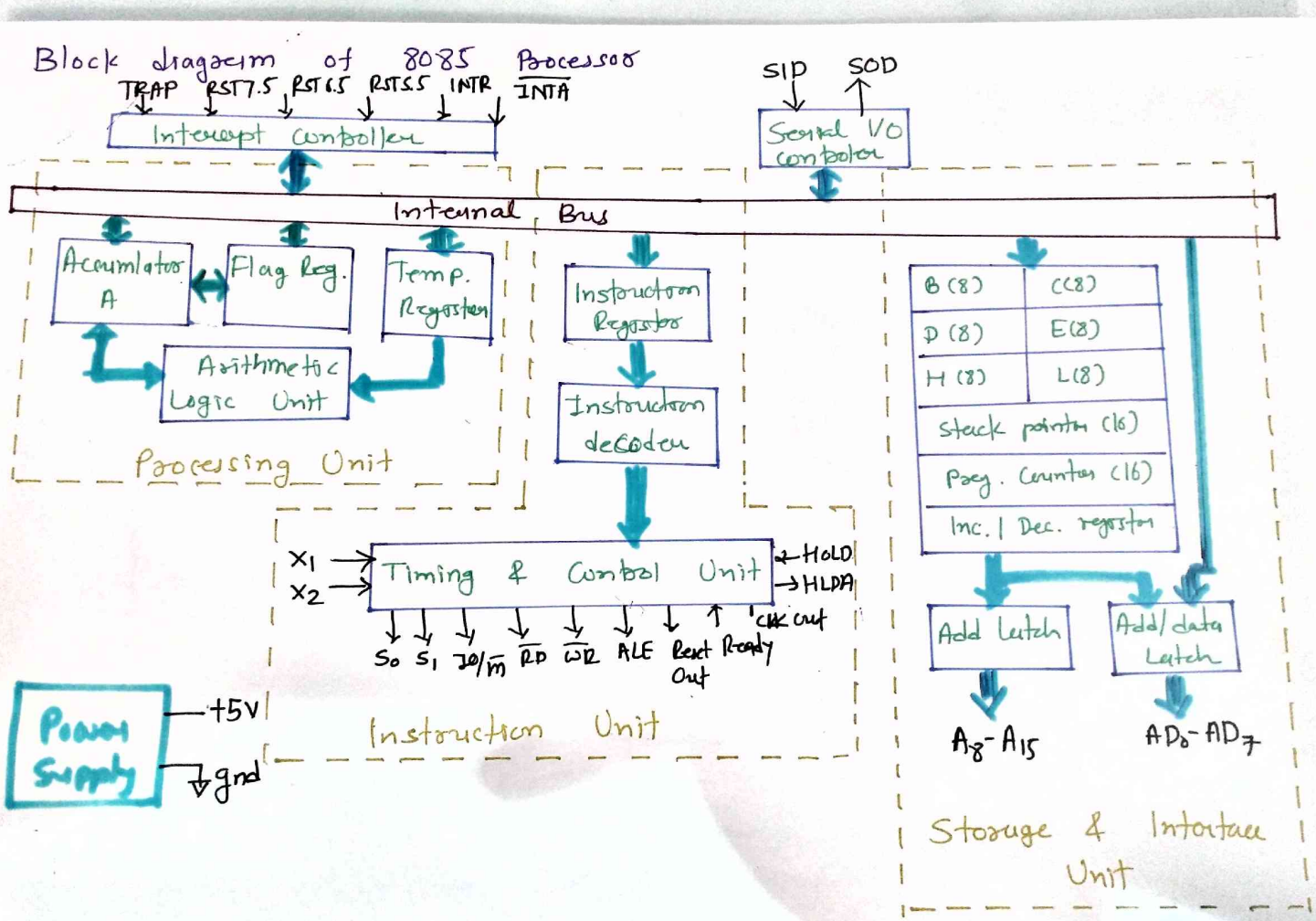
```
LDA  XXXX H.
STA  XXXX H.
MOV  A, M
```

IO mapped IO

- IO is treated as IO.
- 8 bit Addressing ($A_0 - A_7$)
- It can Address = 2^8
  = 256
- No of devices = 256
- Less decoder hardware.
- Available memory is more.
- $\overline{IOR}$ and $\overline{IOW}$ control signal with IO.
- Instruction e.g.

```
IN   XXXX H
OUT  XXXX H.
```

YouTube - Engineering funda



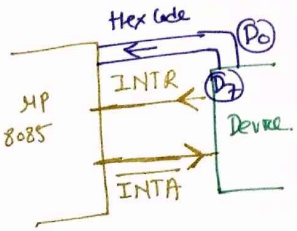
Engineering Junda

ISR (Interrupt Service Routine) in 8085

Interrupt process in 8085 microprocessor.

Step-1 - The Interrupt process should be enabled by writting the EI Instruction and disabled by DI Instruction

- EI [Enable Interrupt]
  - 1 byte Instruction
  - It is used to enable Interrupt
- DI [Disable Interrupt]
  - 1 byte Instruction
  - It is used to disable Interrupt



Step-2 - During execution of program in 8085, µp checks the INTR line during execution of all Instruction.

Step-3 - If the INTR is high and Interrupt is enabled, then µp completes current Instruction and then disable Interrupt Flip-Flop and then sends $\overline{INTA}$ [Interrupt Acknowledge] (Active low signal). The processor can not accept any Interrupt request untill the Interrupt F.F. is enabled again.

Page → 0000 H
FFFF H.

Step-4 - The $\overline{INTA}$ is used to insert RST Instruction through external hardware.

- The RST Instruction is 1 byte Instruction which transfers program control to specific location of 00H page.

Step-5 - When µp receives RST Instruction, it saves the memory address of next Instruction on the Stack.

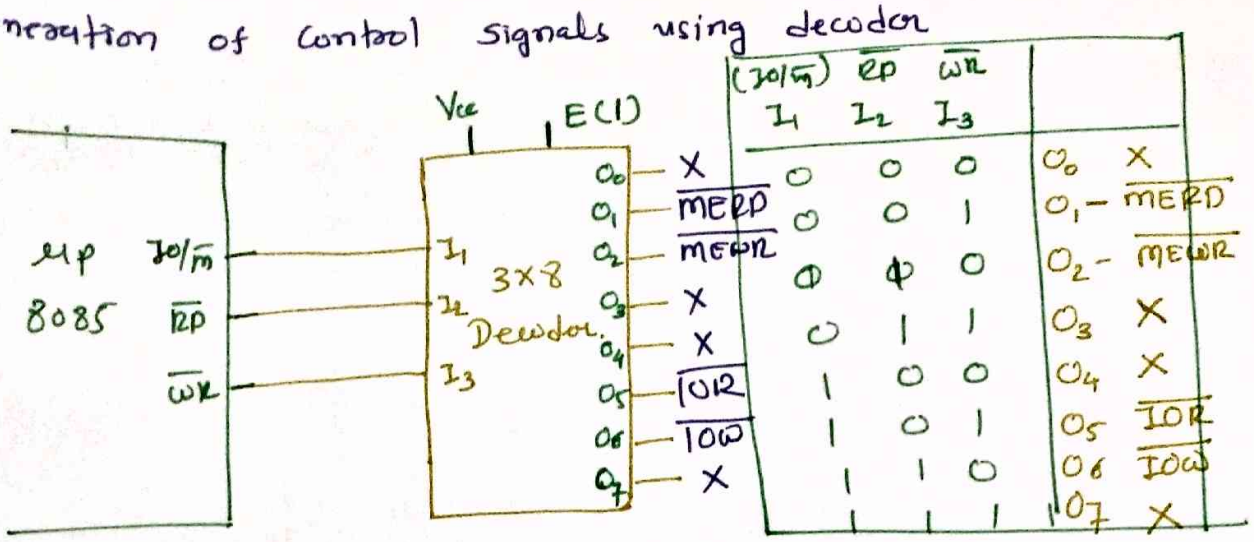
- Then program control get transfers to new location on page 00H.

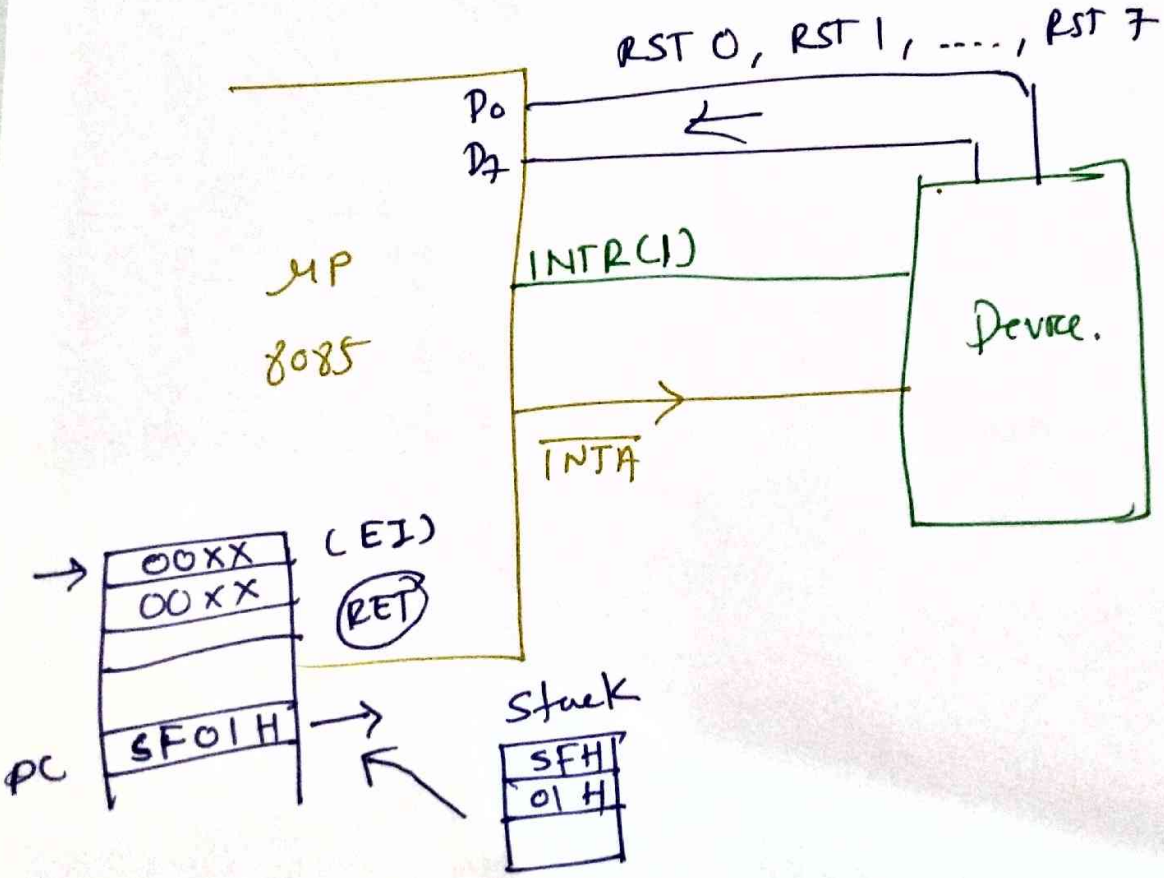
Step-6 - After performing Interrupt task, µp again Jumps to Original program. That Subroutine is known ISR (Interrupt Service Routine)

Step-7 - The ISR should Includes EI at the beggining.

Step-8 - at the end of ISR, RET Instruction retrives the memory add. of original add.

Generation of control signals using decoder





YouTube - Engineering Funda

* Crystal frequency and Microprocessor frequency in 8085

- Crystal Oscillator is connected in between X_1 and X_2 terminals of 8085 to provide crystal frequency.
- It is used for synchronization in 8085.

$$f_{cry} = 6.144 \text{ MHz} \approx 6 \text{ MHz}$$

$$f_{up} = \frac{f_{cry}}{2} = 3.072 \text{ MHz} \approx 3 \text{ MHz}.$$

YouTube – Engineering funda

RESET IN, RESET OUT Pins in 8085 Microprocessor

RESET IN

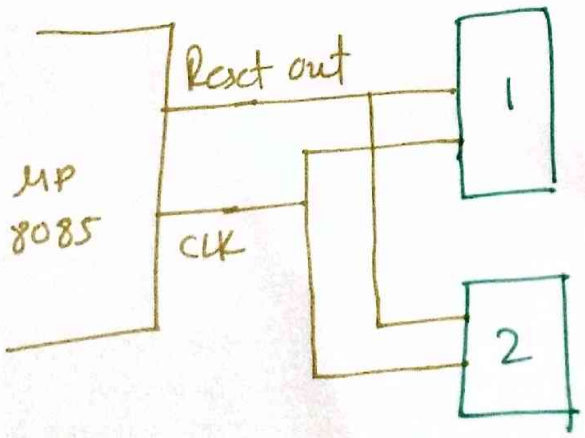
- It is active low signal.
- If this pin is 0, then up will get reset.
- When up is reset, then all registers including PC Program Counter will get cleared to Zero.

$$PC = 0000\ H.$$

- So up will fetch it's next Instruction from 0000 H location of address.
- During reset Operation, all the buses are in high Impedance state or tristated. So power Cusumption is also very less.

RESET OUT

- It is active high signal.
- When this pin is active high, it will reset Interfaced devices with it.
- All the peripherals are connected to Reset out along with Clock out.



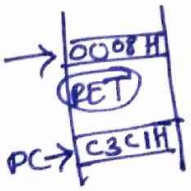
YouTube - Engineering Junda

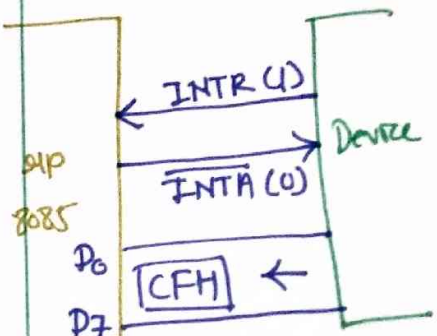
RST Instruction for Interrupt in 8085

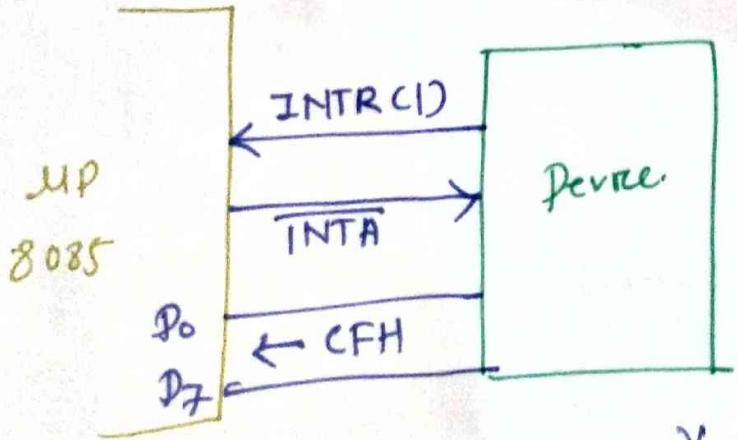
- RST Instruction will transfer program at 00H page.
- It is Similar to Call Instruction.
- At the begging PC (Prog Counter) address will be Stored at Stack.
- When processor encounters RET Instruction, Program Counter PC will retrive it's address from Stack.
- To Insert this One of RST Instruction using external hardware, $\overline{INTA}$ (Interrupt Acknowlege) Signal will be used.

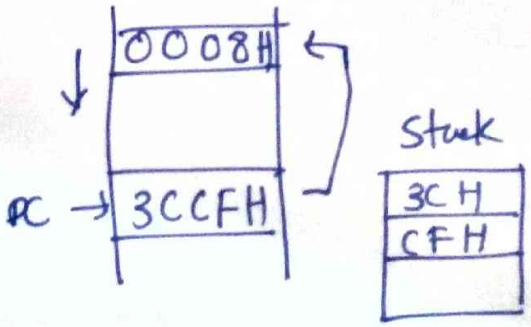
Mnemonics	Hex Code	Call location In Hex
RST 0	C7 H	0000 H
RST 1	CF H	0008 H
RST 2	D7 H	0010 H
RST 3	DF H	0018 H
RST 4	E7 H	0020 H
RST 5	EF H	0028 H
RST 6	F7 H	0030 H
RST 7	FF H	0038 H

00 page.









YouTube - Engineering funda

Hardware Interrupts in microprocessor 8085

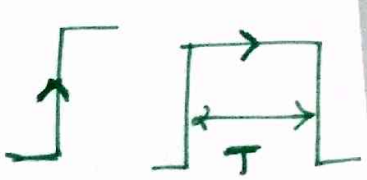
- Interrupt is a signal to the processor, generated by Hardware / Software indicating an immediate attention needed by an event.

In 8085, there are 5 hardware Interrupt.

No	Name	Priority	Vector Add	Masking	Types of trigger
1	TRAP	highest	0024H	NMI	edge & Level
2	RST 7.5	↓	003CH	Maskable	edge trigger
3	RST 6.5	↓	0034H		Level trigger
4	RST 5.5	↓	002CH		
5	INTR	Lowest	NVI [Non Vectored Interrupt]		

RST 0, RST 1, ... , RST 7

EI DI



YouTube - Engineering funda

GATE examples on Interrupt in Microprocessor 8085

* The number of hardware interrupts present in up 8085 are

① 1 ② 4 ③ 5 ④ 13

TRAP
RST 5.5
RST 6.5
RST 7.5
INTR.

* In 8085 up the RST 6 Instruction transfers the program execution to following Location 00?

= 6 × 8 = 48

= 30 H

① 30 H ② 24 H ③ 48 H ④ 60 H

Page
00 30H

* In a Microprocessor, The service routine for a certain interrupt starts from a fixed location of memory which can not be externally set, but the interrupt can be delayed or rejected such an Interrupt is

① Non Maskable & Non Vectored → Maskable. → Vectored

② Maskable & Non Vectored

③ Non Maskable & Vectored

④ Maskable & Vectored.

Stack in microprocessor 8085

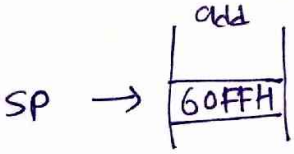
* Basics of stack

- Stack is a group of memory locations
- It is used for storage of information during execution of program
- Stack follows LIFO [Last In First Out]
- Stack is used for storing & retreving data but it does not used for data storage.

* Stack Instructions

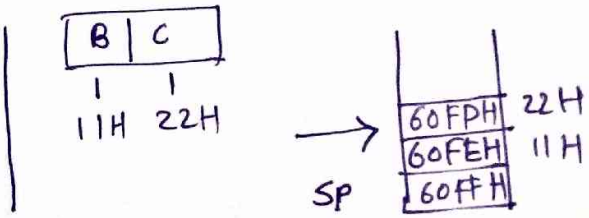
① LXI SP

- This Instruction will define the begining add. of stack.
- e.g. LXI SP, 60FFH.



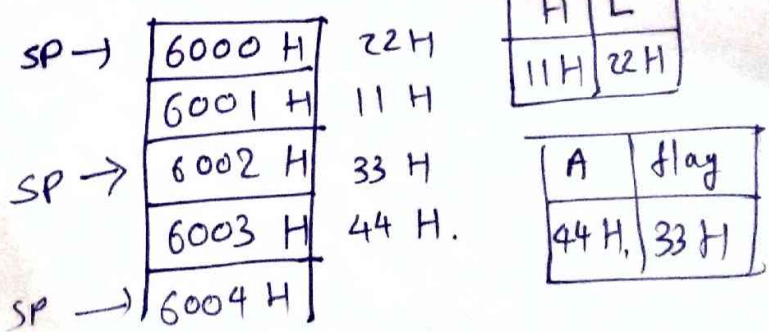
② PUSH RP

- This Instruction is used to load data of Reg. pair on the Stack.
- It is predecreament Instruction
- e.g. Push B



③ POP Rp

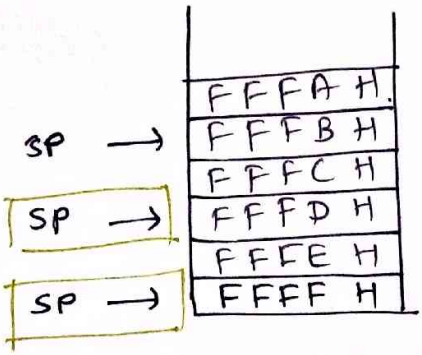
- This Instruction is used to retrive data from Stack.
- It is post Increment Instruction.
- e.g. POP H
  POP PSW



Example on Stack in Microprocessor 8085

Determine the values of DE pair & Program Status word after executing the program SP = FFFF H, BC = 3025 H , HL = B315 H

```
Push  B
Push  H
POP   D
POP   Psw
```



D	E
B3 H	15 H

A	flag
30 H	25 H

YouTube - Engineering Junda

Programmable Interrupt Controller (8259A)

Outlines

- Basics of 8259A
- Features of 8259A
- Block diagram of 8259A
- Working of 8259A.
- Operation of 8259A

Basics of 8259A

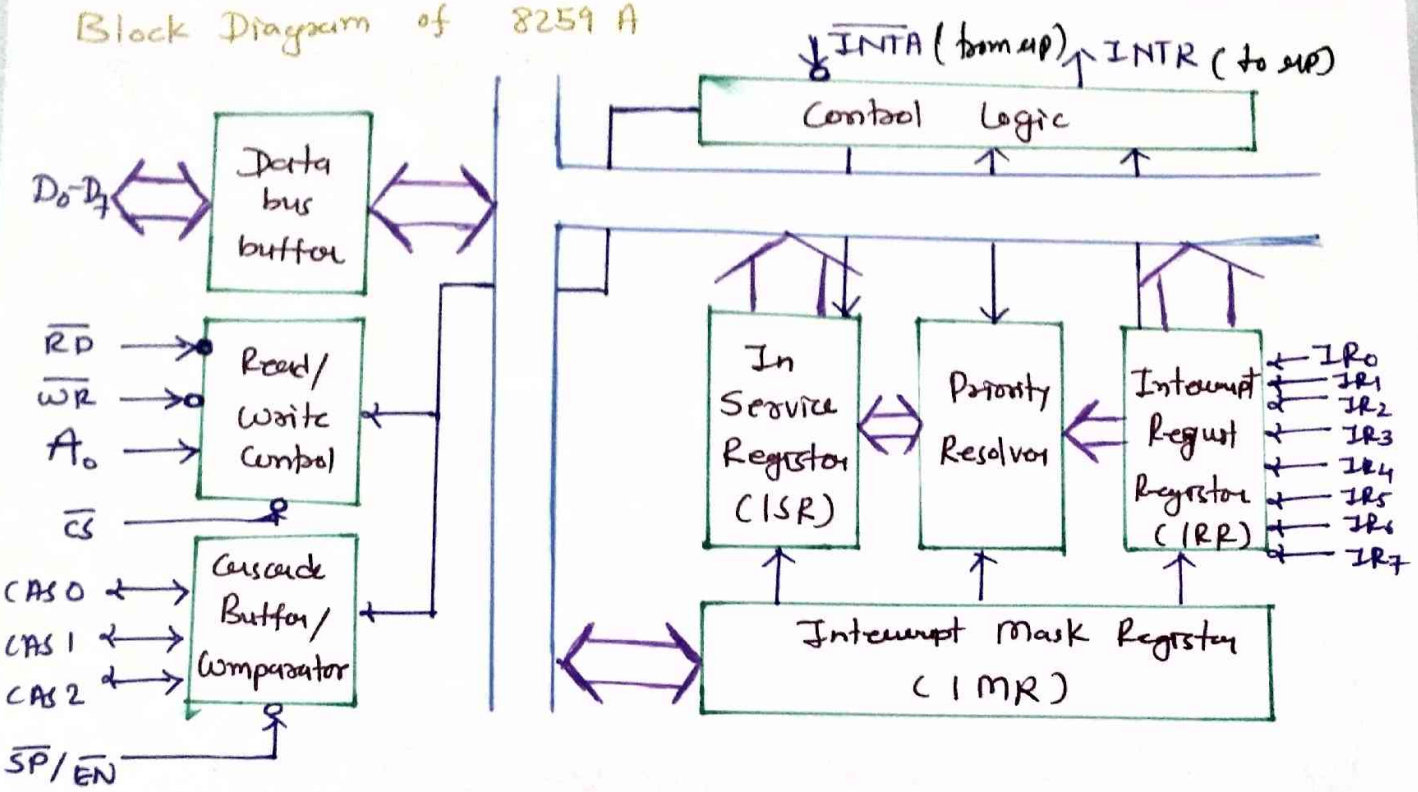
- 8259A is designed to operate with up 8085 with (INTR / $\overline{INTA}$) pins.
- 8085 up has limited hardware Interrupts with limited Interrupt priority.
- Extra hardware is req.ed to Insert RST Instructions.
- 8259A overcomes these limitations of up 8085.
- 8259A can be employed with 16-bit Intel up as 8086/8088

Features of 8259A

- It manages eight Interrupt requests.
- It can vector an Interrupt anywhere on memory map through program control without additional Hardware.
- It can solve eight levels of Interrupt priorities in a variety of modes.
- With additional 8259A devices, the priority scheme can be expanded to 64 Levels.

YouTube - Engineering funda

Block Diagram of 8259 A



Operation of 8259A

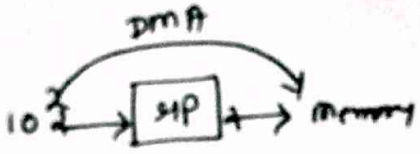
1. One or more Interrupt request lines go high on service.
2. 8259 A resolves priority and sends INTR to up.
3. The up acknowledges the Interrupt by $\overline{INTA}$
4. After $\overline{INTA}$, Opcode for CALL Instruction is placed on data bus.
5. Due to CALL Instruction up (8085) sends two more $\overline{INTA}$ signals.
6. At 1st $\overline{INTA}$, 8259 A place low add. on data bus and at 2nd $\overline{INTA}$, 8259 A place high add. on data bus.
7. Program Sequence now transferred to location specified by CALL Instruction.

YouTube - Engineering funda

Direct Memory Access (DMA) [8257]

Outlines

- Basics of DMA
- HOLD and HLDA signals in Microprocessor
- DMA data transfer blockdiagram
- Working of DMA



* Basics of Direct Memory Access

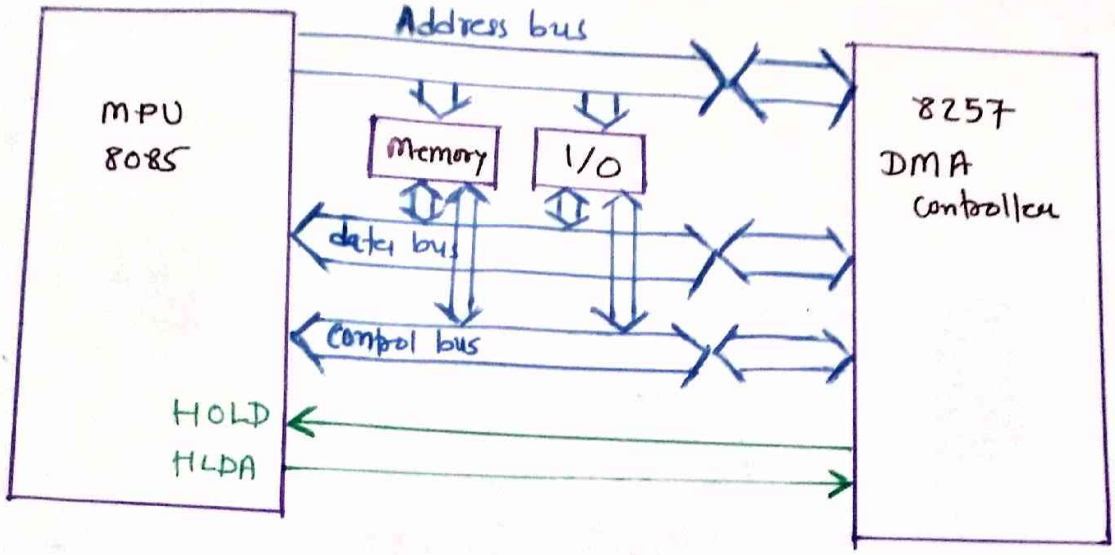
- The Direct memory Access DMA is a process of comm^{n} or data transfer controlled by an external peripheral.
- In situations, in which the μp controlled data transfer is too slow, the DMA is generally used.
- e.g. - Data transfer between a floppy disk and R/W memory.

* HOLD and HLDA signals in μp 8085

HOLD
- It is active high signal to the μp 8085.
- Another Master is requesting the use of address and data buses.
- After HOLD signal received by μp, μp relinquishes the buses in the following machine cycles.
- All buses are tri-stated and a Hold Acknowledge (HLDA) signal is sent out.
- MPU regains the control of buses after HOLD goes low.

HLDA
- It is hold Acknowledge signal.
- This is an active high signal by MPU indicating that the MPU is relinquishing the control of the buses.

DMA data transfer blockdiagram



Working of DMA

- DMA sends HOLD signal to up
- up Completes current machine cycle and floats Add/data on high impedance mode.
- Then up sends HLDA signal to DMA.
- Now DMA takes Control of Add/data buses and bypass up to transfer data Inbetween peripherals.
- At the end of data transfer HOLD goes low and up will take Control on Add/data buses.

YouTube- Engineering funda

Programmable Peripheral Interface [8255]

Outlines

- Basics of 8255
- Control Signals in 8255
- Block diagram of 8255
- Control word of 8255
- Modes of 8255

* Basics of 8255

- 8255 (Programmable Peripheral Interface) is designed to Increase I/O Interfacing Capability of µP.
- It has 24 I/O pins that can be grouped in two 8 bits Parallel ports : A and B, with the remaining 8 bits into Port C.
- Port C can be used as Individual bits or can be used as 4 bit ports : C_{upper} and C_{lower}.
- 8255 functions in following modes
  - I/O Modes [mode 0, mode 1 & mode 2]
  - BSR Mode.

* Control Signals of 8255

$\overline{RD}$ [Read] - Active low signal
- MPU reads data from Selected port of 8255

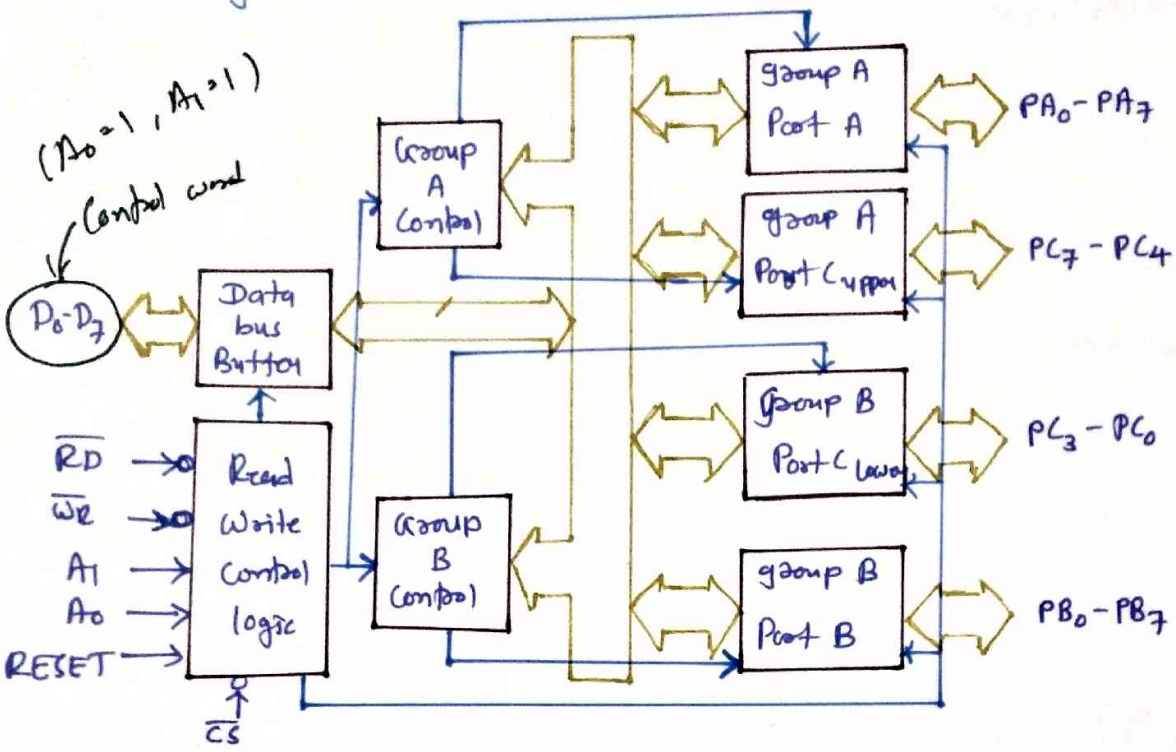
$\overline{WR}$ [write] - Active low signal
- MPU write data on to Selected port of 8255

RESET - Active high signal.
- It Clears Control registers and Sets all ports in Input mode.

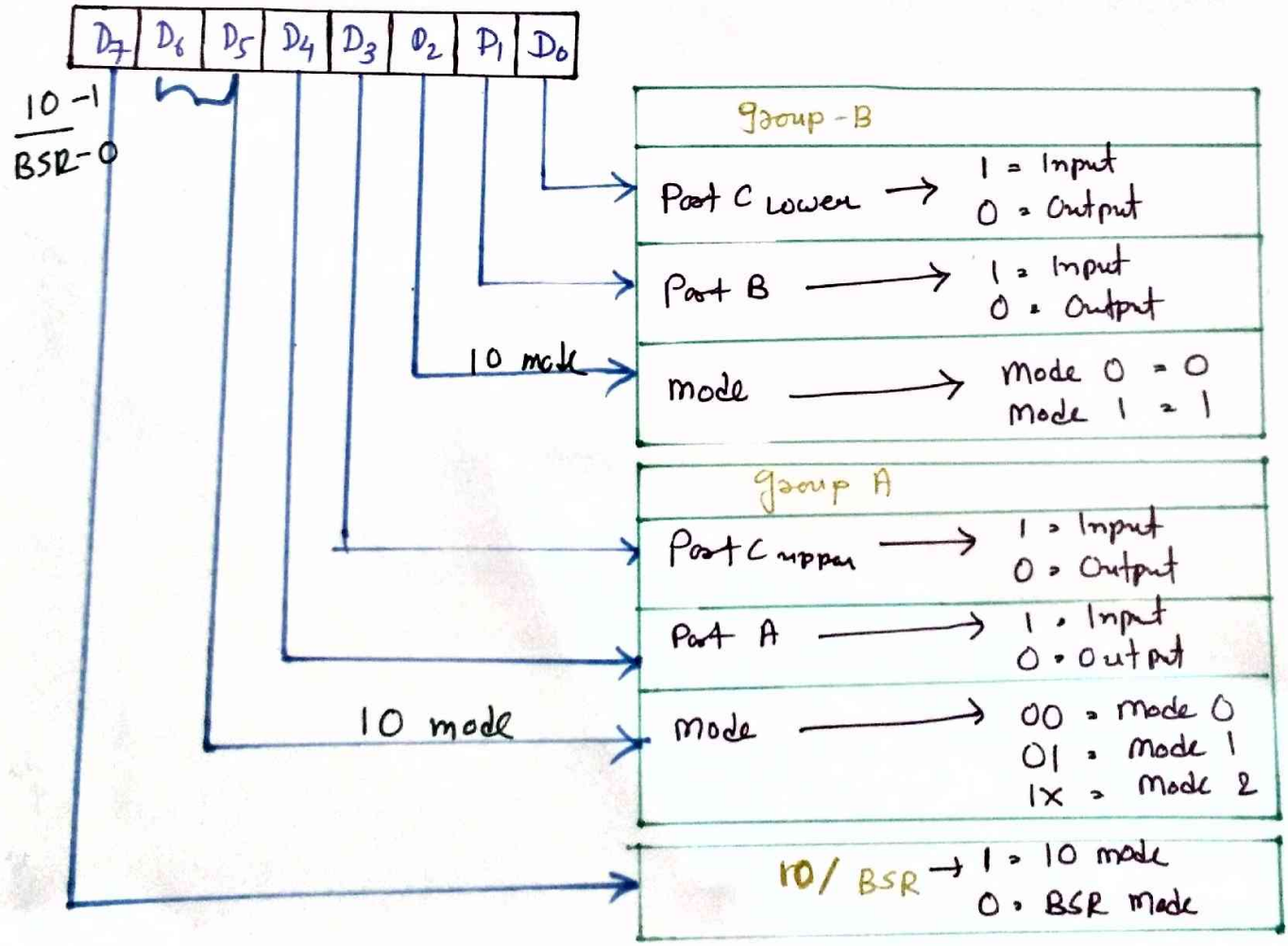
$\overline{CS}$, A_0 and A_1

$\overline{CS}$	A_1	A_0	
0	0	0	→ Port A
0	0	1	→ Port B
0	1	0	→ Port C
0	1	1	→ Control register
1	X	X	← 8255 not Selected

Block diagram of 8255



Control word of 8255



Modes in 8255

- BSR [Bit set Reset Mode] -

D_7	D_6 D_5 D_4	D_3 D_2 D_1	D_0
↓	↓	↓	↓
0	X	bit Select	S/R = 1 (Set)
[BSR]	(unused)		= 0 (Reset)

0	0	0	– 0^{th} bit
0	0	1	– 1^{st} bit
0	1	0	– 2^{nd} bit
0	1	1	– 3^{rd} bit
⋮			
1	1	1	– 7^{th} bit

- 10 modes

Mode 0 - Simple IO for Ports A, B & C.

Mode 1 - IO port for A and/or B.
- Port C used as handshake

Mode 2 - Bidirectional data buses for Port A
- Port B in (Mode 0 or Mode 1)
- Port C used as handshake.

Programmable Interval timer (8254/8253)

Outlines

- Basics of 8254/8253 - working of 8254/8253
- Control signals of 8254/8253
- Block diagram of 8254/8253
- Control register of 8254/8253
- Applications of 8254/8253

* Basics of 8254/8253

- 8254 functions like software designed counters and timers.
- It generates accurate time delays.
- 8254 includes three identical 16 bit counters that can operates in 6 different modes.
- 8254 can operate with (DC to 8 MHz) and 8253 can operate with DC to 2MHz
- 8254 Includes a status - back command that can latch the count and the status of the counter.

* Working of 8254/8253

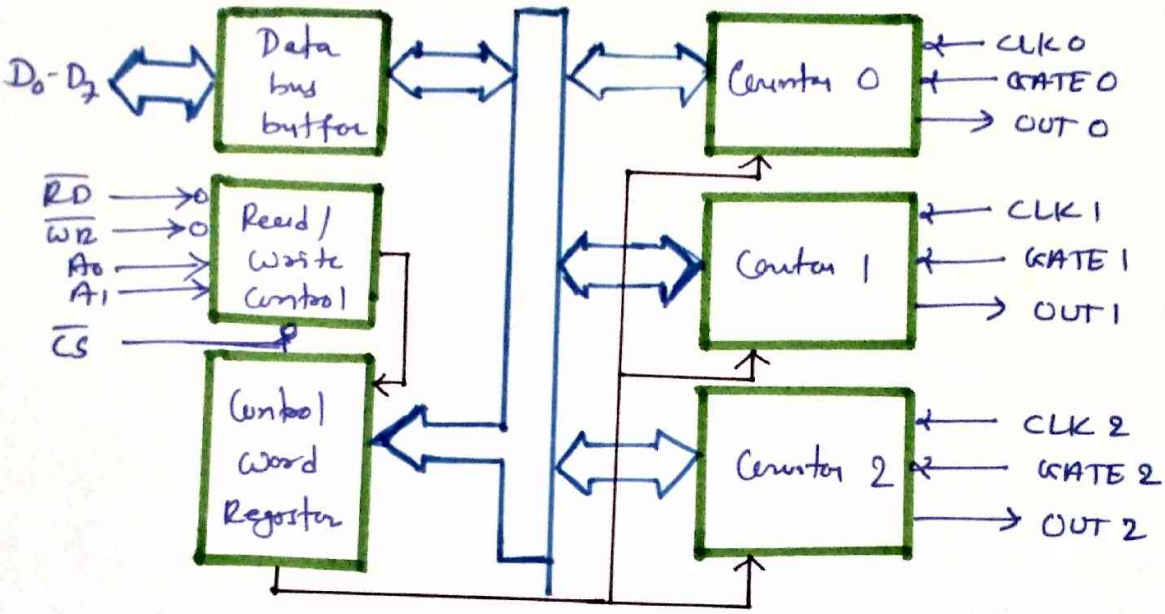
- 16 bit count is loaded in its register.
- On command, it begins to decrement until it reaches 0.
- At the end of the count, it generates a pulse that can be used to interrupt MPU.
- The counter either in binary or BCD

* Control signals of 8254/8253

- It has $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$, and A_0 & A_1 as control signals.
- In IO mode, $\overline{RD}$ & $\overline{WR}$ is connected to $\overline{IOR}$ & $\overline{IOW}$.

A_1	A_0	Selection
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control Register

Block Diagram of 8254 / 8253



Control Register of 8254 / 8253

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC	SC_0	RW_1	RW_0	M_2	M_1	M_0	BCD

SC - Select Counter

SC_1	SC_0	
0	0	Select Counter 0
0	1	select Counter 1
1	0	Select Counter 2
1	1	Read back command

M - Mode

M_2	M_1	M_0	
0	0	0	mode 0
0	0	1	mode 1
X	1	0	mode 2
X	1	1	mode 3
1	0	0	mode 4
1	0	1	mode 5

RW - Read / write.

RW_1	RW_0	
0	0	— Counter Latch
0	1	— R/W Least byte only
1	0	— R/W most byte Only
1	1	— R/W Least 1st & then most byte.

BCD

0	— Binary Counter 16 bits
1	— BCD Counter (4 decades)

Applications of 8254 / 8253

- Real time Clock
- An event Counter
- A digital One shot
- A Square wave generator

Programmable Communication Interface 8251 / 8251A

Outline

- Basics of 8251
- Working of 8251
- Control signals of 8251
- block diagram of 8251

Basics of 8251

- 8251 is a programmable chip designed for synchronous and asynchronous serial data communication
- 8251A is advanced version of 8251.
- Control logic of 8251 is connected with MPU to determine the function of 8251.
- It converts parallel to serial for transmission and it converts serial to Parallel for reception of data.

Working of 8251

- Transmitter section
  - Receives parallel data from MPU
  - Converts into Serial
  - Transmits serially on TxD line
- Receiver section
  - Receives serial data on RxD
  - Converts serial into Parallel
  - Transfer data to MPU.
- This data transmission can be done for synchronous and Asynchronous data transfer.

Control signals of 8251

$\overline{CS}$ - Chip Select - Active low signal
- It selects 8251 chip

C/$\overline{D}$ - Control / Data - If it is high (1) then control register is addressed
- If it is low (0) then data buffer is addressed

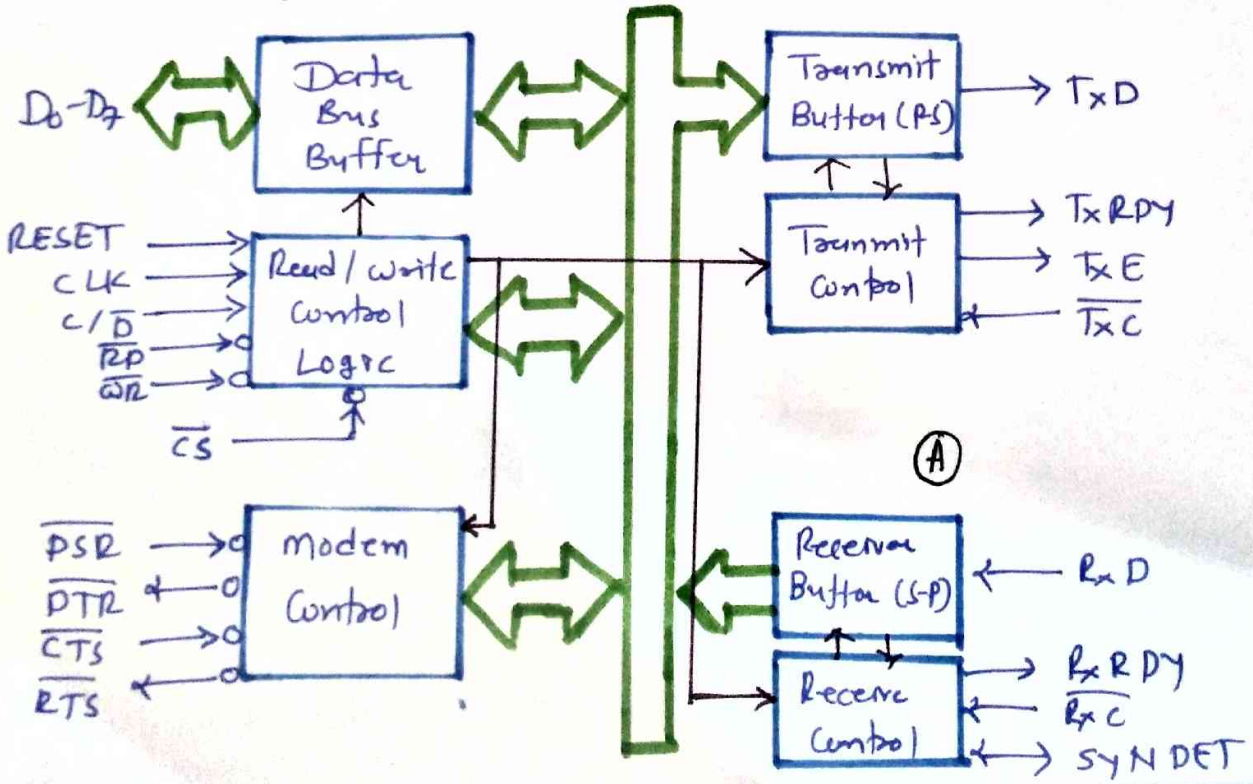
$\overline{WR}$ - Write - Active low signal
- This is connected to $\overline{MEMW}$ or $\overline{IOW}$ to write data

$\overline{RD}$ - Read - Active low signal
- This is connected to $\overline{MERD}$ or $\overline{IOR}$ to Read data

RESET - A high input resets 8251 and forced it to idle mode

$\overline{CS}$	C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	
0	1	1	0	- MPU writes instructions in control register
0	1	0	1	- MPU read status from the status register
0	0	1	0	- MPU output data to data Buffer
0	0	0	1	- MPU accepts data from data Buffer
1	x	x	x	- chip not selected.

Block Diagram of 8251



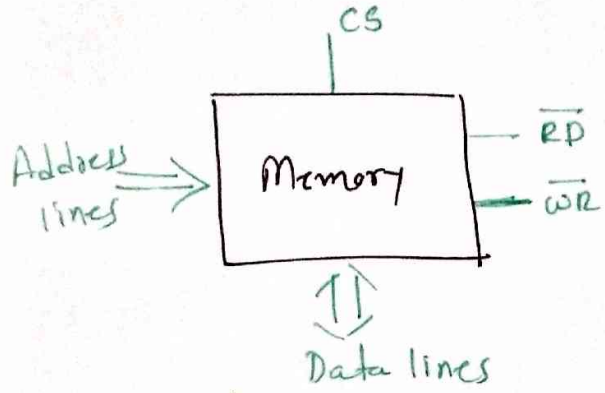
Series of Memory IC's

- EPROM -(27)
- RAM —(61)

Size	EPROM	RAM
1 KB	2708	6108
2 KB	2716	6116
4 KB	2732	6132
8 KB	2764	6164
16 KB	27128	61128

YouTube - Engineering funda

Starting and Ending Address Calculation of Memory IC's



→ 1kB Memory

→ Address lines = 2^{10}

↓

10 Address lines.

→ Data lines = 8

→ RAM – RD, WR | – ROM – RD

2 KB

→ 1 kB Memory

- total Add = 2×2^{10}
- Address lines = 10 = 11
- data lines = 8

→ for 8085 up, A_0–A_{15}

→ Starting 0000 H.

	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
Starting Address 0000H →	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ending Address 03FFH →	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

↑ Chip Select ↑ Address

2kB → 07FFH

→ 1 KB Memory, 0344 H. (Starting Address)

2KB

```
  0344 H  ← Starting Address
  03FF H
  ------
  0743 H  ← Ending Address.
```

→ 1 KB Memory, 15FF H (Ending Address).

2KB

```
  15FF H  ← Ending Address
- 03FF H
  ------
  1200 H  ← Starting Address
```

YouTube - Engineering funda

Calculation of starting address from ending address with offset.

- Determine the starting address of 4KB memory with ending address BABF H.

- For 4KB memory, If 0000H is Starting Address.
- For 4KB
- total Address = $4 \times 2^{10} = 2^{12}$
- total Address lines = 12
- total data lines = 8

	A_{15} A_{14} A_{13} A_{12}	A_{11} A_{10} A_9 A_8	A_7 A_6 A_5 A_4	A_3 A_2 A_1 A_0
Starting 0000 H	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
Ending OFFF H	0 0 0 0	1 1 1 1	1 1 1 1	1 1 1 1
	↑ Chip Select		↑ Addressing	

$$\begin{array}{r} {}^{16} \\ A\,9\,{}^{16} \\ BABF\ H. \leftarrow \text{Ending Address.} \\ OFFF\ H \\ \hline AACO\ H \leftarrow \text{Starting Address.} \end{array}$$

YouTube - Engineering funda

Calculation of size of memory from starting address and ending address.

Determine the size of memory whose starting address & ending address are 4A00 H & 69FF H, Respectively.

- Size of memory = Ending Add - Starting Address

$$\begin{array}{r} {}^{5}{}^{16} \\ \$9FF\ H \\ -\ 4A00\ H \\ \hline \text{total size} = 1FFF\ H \end{array}$$

↑ total 13 lines are there in Address.

- total Add. lines = 13
- Size of memory = $2^{13} = 2^3 \times 2^{10}$ = **8 KB**

YouTube - Engineering funda

Memory expansion by multiple memory Interface with processor

Construct 4K X8 EPROM by using 1K X8 EPROM IC's.

- To make 4K X8 by using 1K X 8 we need 4 IC's of 1K X8 Chip.

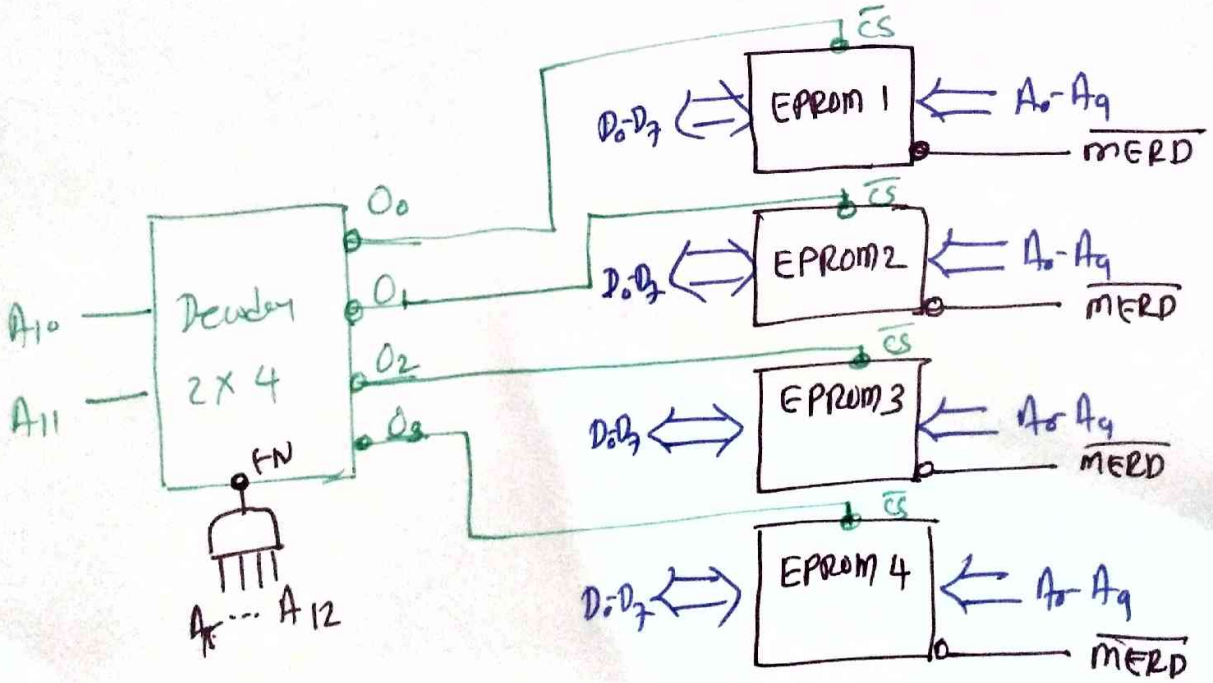
- For (1K)X(8) chip. (1K: Add, 8: data)

  Add lines = 2^{10} = 10 (Add. lines)

  Data lines = 8

	A_{15} A_{14} A_{13} A_{12}	A_{11} A_{10}	A_9 A_8	A_7 A_6 A_5 A_4	A_3 A_2 A_1 A_0	
EPROM - 1	0 0 0 0	0 0	0 0	0 0 0 0	0 0 0 0	0000H
	0 0 0 0	0 0	1 1	1 1 1 1	1 1 1 1	03FFH
EPROM - 2	0 0 0 0	0 1	0 0	0 0 0 0	0 0 0 0	0400H
	0 0 0 0	0 1	1 1	1 1 1 1	1 1 1 1	07FFH
EPROM - 3	0 0 0 0	1 0	0 0	0 0 0 0	0 0 0 0	0800H
	0 0 0 0	1 0	1 1	1 1 1 1	1 1 1 1	0BFFH
EPROM - 4	0 0 0 0	1 1	0 0	0 0 0 0	0 0 0 0	0C00H
	0 0 0 0	1 1	1 1	1 1 1 1	1 1 1 1	0FFFH.

↑ unused ↑ decode IC ↑ Addressing.



Number of IC's required to construct memory

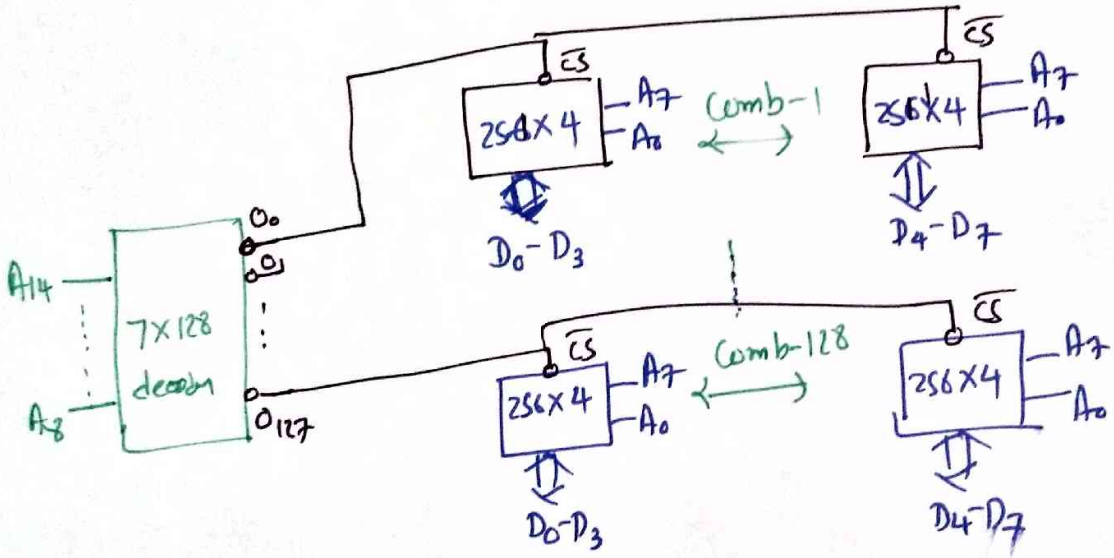
e.g. Construct 32 kB memory useing 256 x 4 IC's.

$$\text{No of IC's} = \frac{(\text{req}^{\text{ed}}\ \text{data lines})}{(\text{given data lines})} \times \frac{(\text{req.}^{\text{ed}}\ \text{Memory})}{(\text{given Memory})}$$

$$= \frac{8}{4} \times \frac{32 \times 2^{10}}{2^{8}} 4$$

$$= 2 \times 32 \times 4$$

$$= 256$$



Number of memory IC's can be interfaced with 8085

e.g. How many memory IC's can be interfaced with 8085 up for given specification

(a) 2K × 8
(b) 4K × 8
(c) 8K × 4

- (a) 2K × 8 (for up total memory 64 KB)

- total IC's $= \left(\frac{\text{req}^{ed}\ \text{data}}{\text{given data}}\right) \times \left(\frac{\text{req}^{ed}\ \text{Memory}}{\text{given Memory}}\right)$

$= \frac{8}{8} \times \frac{64}{2} = \boxed{32}$

- (b) 4K × 8 (for up total memory 64 KB)

total IC's $= \left(\frac{\text{req}^{ed}\ \text{data}}{\text{given data}}\right) \times \left(\frac{\text{req}^{ed}\ \text{Add}}{\text{given Add}}\right)$

$= \frac{8}{8} \times \frac{64}{4} = \boxed{16}$

- (c) 8K × 4 (for up total memory 64 KB)

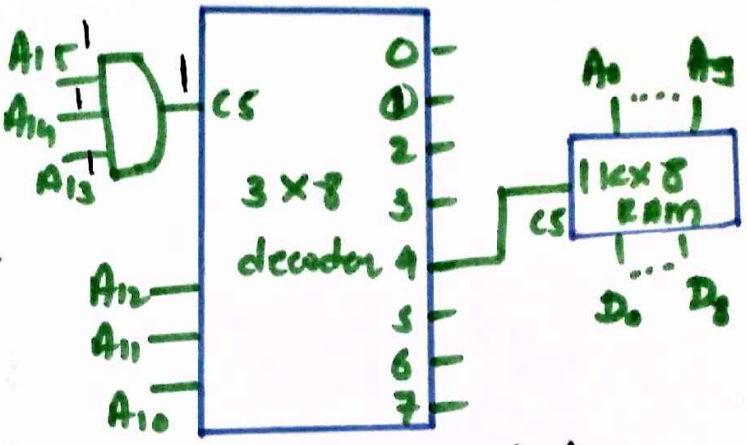
total IC's $= \left(\frac{\text{req}^{ed}\ \text{data}}{\text{given data}}\right) \times \left(\frac{\text{req}^{ed}\ \text{Add}}{\text{given Add}}\right)$

$= \frac{8}{4} \times \frac{64}{8} = 2 \times 8 = \boxed{16}$

Examples on Memory Mapping (GATE Examples)

e.g. A 8 bit microprocessor has 16 bit Address lines (A_0-A_{15}) with 1 Kb memory chip as shown in figure. The address for the chip is

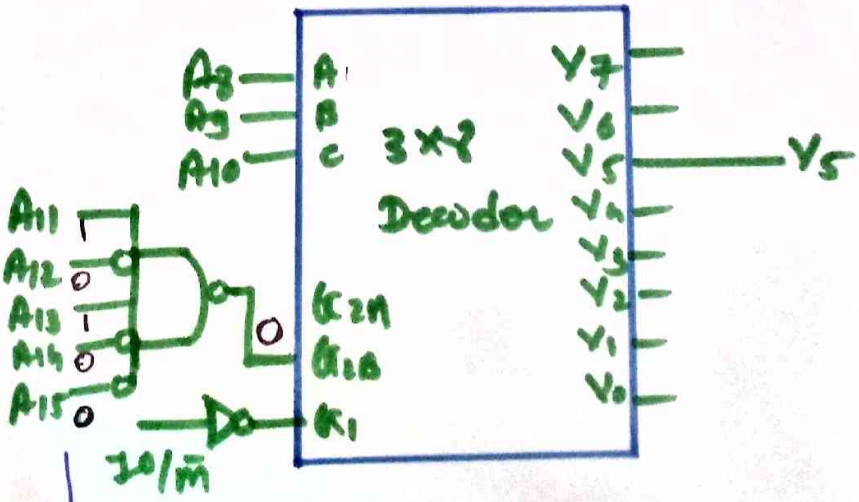
(a) F00F H - F40E H
(b) F100 H - F4FF H
(c) F000 H - F3FF H
(d) F700 H - FAFF H.



A_{15} A_{14} A_{13}	A_{12}	A_{11} A_{10}	A_9 A_8	A_7 A_6 A_5 A_4	A_3 A_2 A_1 A_0	
1 1 1	1	0 0	0 0	0 0 0 0	0 0 0 0	- F000 H
1 1 1	1	0 0	1 1	1 1 1 1	1 1 1 1	- F3FF H.
Chip	Decoder		Addressing			

e.g In the circuit shown, device connected V_5 can have address in the range.

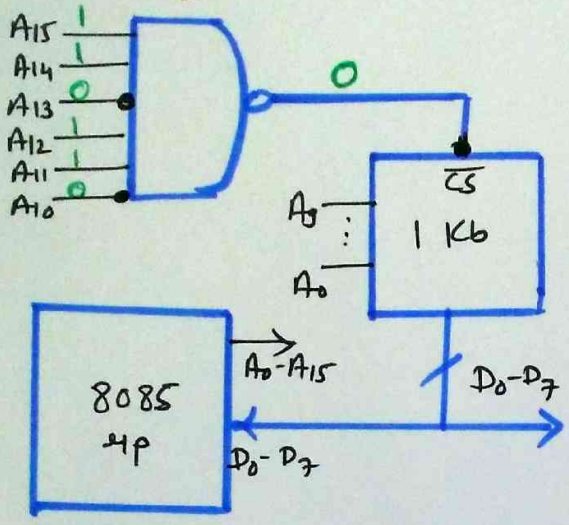
(a) 2000 - 20FF
(b) 2D00 - 2DFF
(c) 2E00 - 2EFF
(d) FD00 - FDFF



A_{15}	A_{14} A_{13} A_{12}	A_{11}	A_{10} A_9 A_8	A_7 A_6 A_5 A_4	A_3 A_2 A_1 A_0	
0	0 1 0	1	1 0 1	0 0 0 0	0 0 0 0	2D00 H
0	0 1 0	1	1 0 1	1 1 1 1	1 1 1 1	2DFF H.
chip Select			decoder	Addrsing.		

GATE examples on memory Mapping

e.g. A 1Kb IC is interfaced to the 8085 µp as shown in figure. Determine the address range of memory IC.



$A_{15}A_{14}A_{13}A_{12}$	$A_{11}A_{10}$	A_9A_8	$A_7A_6A_5A_4$	$A_3A_2A_1A_0$
1 1 0 1	1 0	0 0	0 0 0 0	0 0 0 0
1 1 0 1	1 0	1 1	1 1 1 1	1 1 1 1
Chip selat.			Addressing.	

D800 H ← Starting Add

DBFF H ← Ending Add.

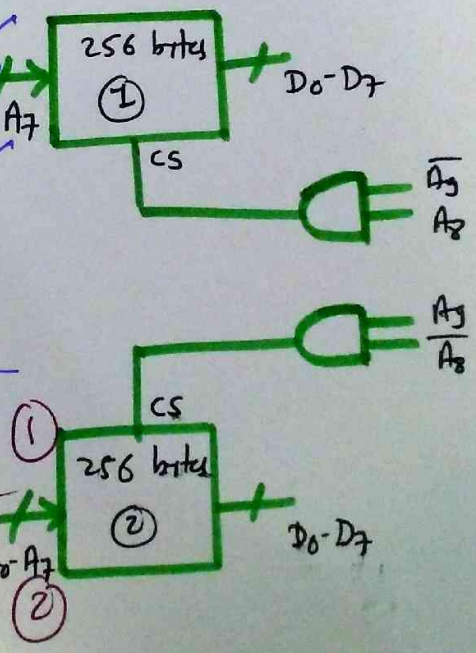
e.g. which memory address range is not represented by chip 1 and chip 2 in given figure.

(a) 0100 H - 02FFH (0001 - 0010) ✓

(b) 1500 H - 16FFH (0101 - 0110) ✓

(c) F900 H - FAFFH (1001 - 1010) ✓

(d) F800 H - F9FFH (1000 - 1001) ✓ (X)



$A_{15}A_{14}A_{13}A_{12}$	$A_{11}A_{10}$	A_9A_8	$A_7A_6A_5A_4$	$A_3A_2A_1A_0$
X X XX	X X	0 1	0 0 0 0	0 0 0 0
X X XX	X X	0 1	1 1 1 1	1 1 1 1
X X XX	X X	1 0	0 0 0 0	0 0 0 0
X X XX	X X	1 0	1 1 1 1	1 1 1 1
Unused		Chip	Addressing.	

Example of Memmory Mapping [GATE]

Figure shows the memory circuit of 8085 µp.

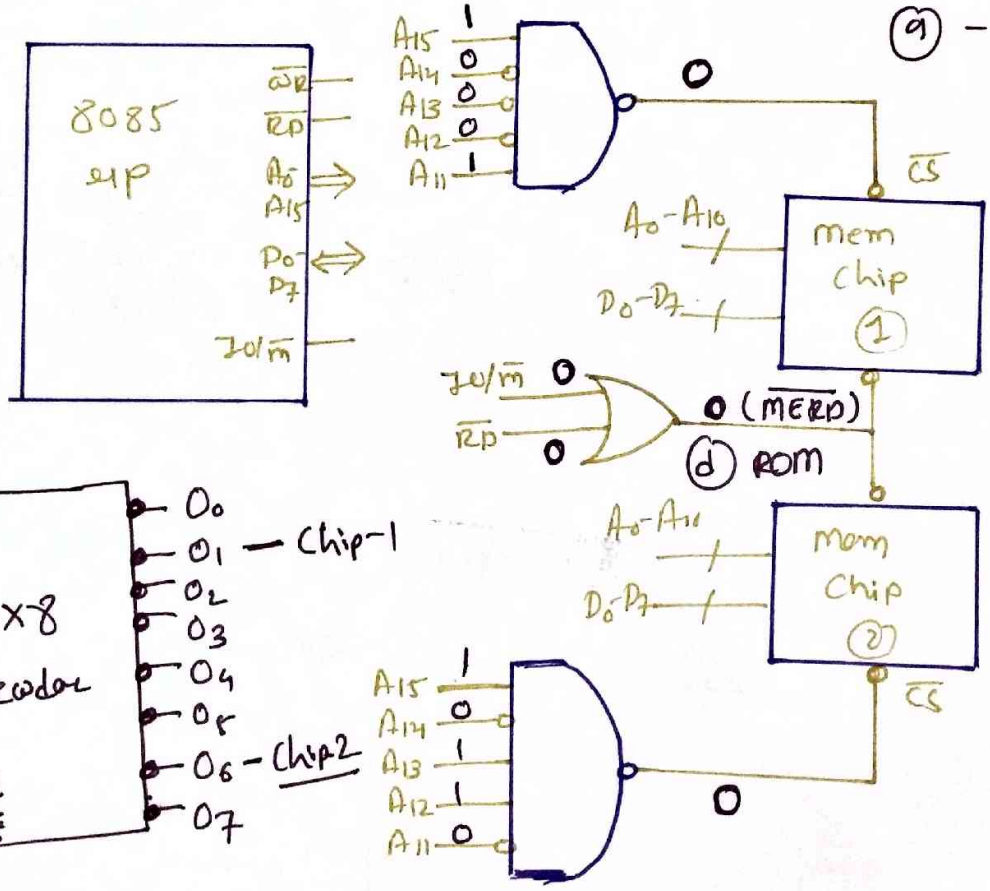
(a) what is the total size of memory in circuit.

(b) what are the begining & ending address of chip 1.

(c) what are the begining & ending address of chip 2.

(d) The memory chips in circuits are RAM or ROM ?

(e) How will you replace the NAND gates in circuit using one 3 to 8 decoder without changing the memory size or address. Assume the decoder is having high & low enable.



(a) – Size of memory

$= 2^{11} = 2 \times 2^{10}$

$= 2$ KB

for Chip-1

$= 2^{11} = 2 \times 2^{10}$

$= 2$ KB

– total size

$= 2$ KB + 2KB

$= 4$ KB.

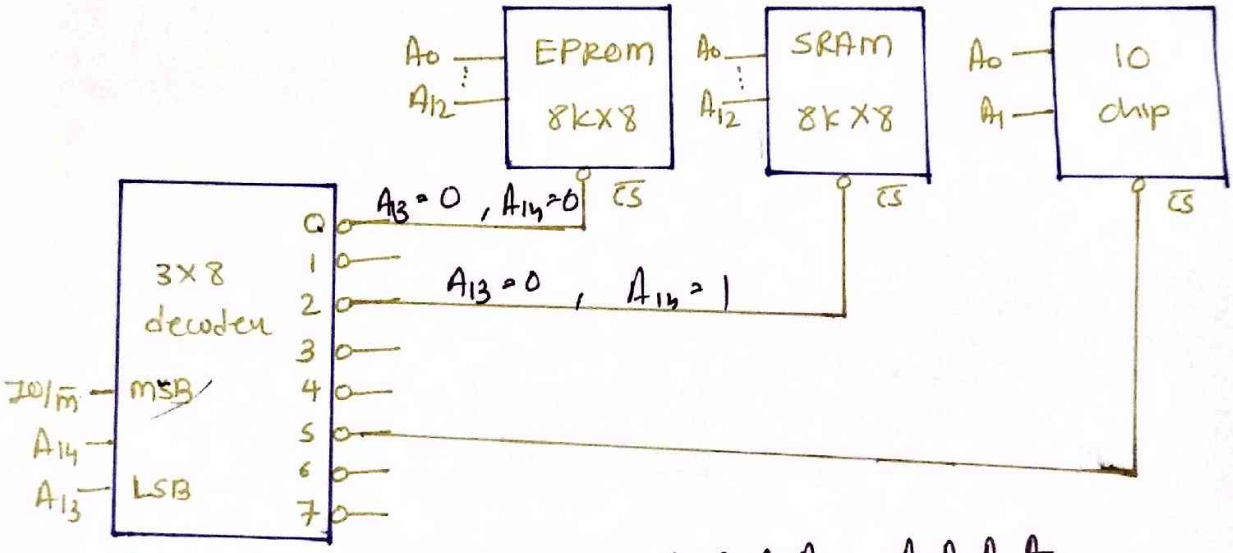
A_{15} A_{14} A_{13} A_{12}	A_{11}	A_{10} A_9 A_8	A_7 A_6 A_5 A_4	A_3 A_2 A_1 A_0	
1 0 0 0	1	0 0 0	0 0 0 0	0 0 0 0	Chip 1 – 8800 H
1 0 0 0	1	1 1 1	1 1 1 1	1 1 1 1	Chip 1 – 8FFF H
1 0 1 1	0	0 0 0	0 0 0 0	0 0 0 0	Chip 2 – B000 H
1 0 1 1	0	1 1 1	1 1 1 1	1 1 1 1	Chip 2 – B7FF H
Chip Select.		Addressing			

Example of Memmory Mapping [GATE]

Consider the decoder Circuit shown in figure for providing chip select signal to an EPROM, a RAM and an IO chip with four addressable registers from a demultiplexed 8085 address bus then,

① Specify all the memory address ranges to which the EPROM will respond.

② Specify all the memory address ranges to which the RAM will respond.



A_{15}	A_{14}	A_{13}	A_{12}	$A_{11}A_{10}A_9A_8$	$A_7A_6A_5A_4$	$A_3A_2A_1A_0$	
X	0	0	0	0000	0000	0000	EPROM
X	0	0	1	1111	1111	1111	EPROM
X	1	0	0	0000	0000	0000	SRAM.
X	1	0	1	1111	1111	1111	SRAM.
	Chip select			Addressing.			

→ If $A_{15}=0$ - EPROM - 0000H
- 1FFFH.
- SRAM - 4000H
- 5FFFH.

If $A_{15}=1$ - EPROM - 8000H
- 9FFFH
- SRAM - C000H.
- DFFFH.

GATE examples on Microprocessor Interfacing with Peripherals and Addressing Registers.

* For a microprocessor system using IO mapped IO, the following statement is not true.

✓ A) Memory space available is greater

✓ B) Not all data transfer instructions are available

✓ C) IO & memory address space are distinct.

✗ D) IO address space is greater.

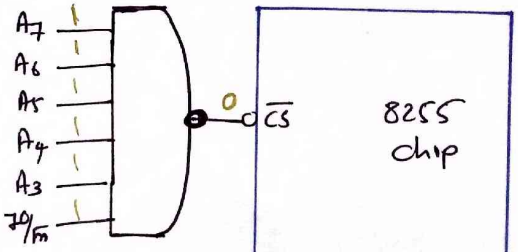
* An 8255 chip is interfaced to an 8085 up system as an IO mapped IO as shown in figure, The address lines A_0 & A_1 of the 8085 are used by 8255 chip to decode internally its three ports & the control register. The address lines A_3 to A_7 as well as $IO/\overline{M}$ signals are used for address decoding. the range of address for which the 8255 chip would get selected is

(a) F8 - FB H

(b) F8 - FC H

(c) F8 - FF H

(d) F0 - F7 H.



A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
1	1	1	1	1	X	0	0	
1	1	1	1	1	X	1	1	
Chip Select								Addressing.

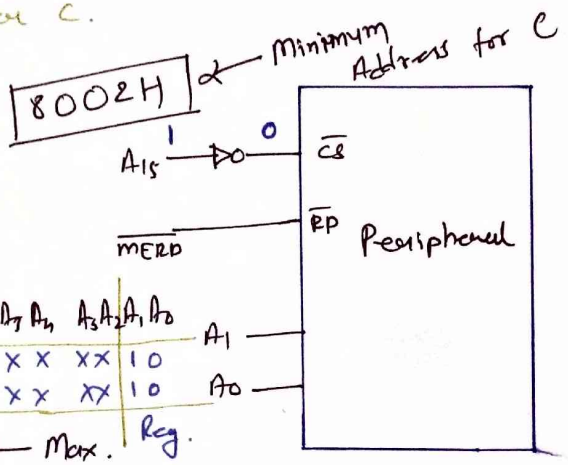
- IF $A_2 = 0$
  F8 - FB H

- IF $A_2 = 1$
  FC - FF H

GATE examples on Microprocessor Interfacing with Peripherals and Addressing registers.

* In the following peripherals, determine the maximum address of register C.

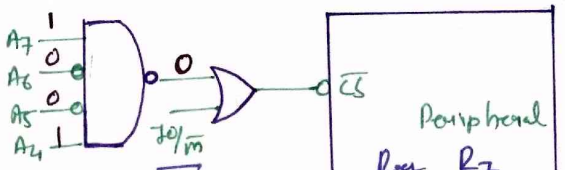
A_1 A_0
0 0 — reg A
0 1 — reg B
1 0 — reg C
1 1 — reg D



A_{15}	$A_{14}A_{13}A_{12}$	$A_{11}A_{10}A_9A_8$	$A_7A_6A_5A_4$	$A_3A_2A_1A_0$
1	X X X	X X X X	X X X X	X X 1 0
1	X X X	X X X X	X X X X	X X 1 0

chip select — FFFEH ← Max. — Reg.

* A Peripheral is interfaced to 8p as shown in fig. Determine.
  (a) Mode of Interfacing [IO mapped IO]
  (b) the number of internal registers & their address



— total 8 Registers.

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0			
1	0	0	1	0	X	0	0	R_0	90	94 H
1	0	0	1	0	X	0	1	R_1	91	95 H
1	0	0	1	0	X	1	0	R_2	92	96 H
1	0	0	1	0	X	1	1	R_3	93	97 H
1	0	0	1	1	X	0	0	R_4	98	9C H
1	0	0	1	1	X	0	1	R_5	99	9D H
1	0	0	1	1	X	1	0	R_6	9A	9E H
1	0	0	1	1	X	1	1	R_7	9B	9F H.

Basics and details of Instruction Set for 8085 μp

- An Instruction is a command given to the μp to Perform a Specific Operation on data. (S, Z, Ac, P, Cy)
- Basically we are having five groups of Instructions

① data transfer Instructions (Reg - Reg.) (Reg - Mem) (No flag)
② Arithmatic Instructions (Add, Sub) All flag.
③ logical Instruction All flag.
④ branching Instructions (Jump, Call, Ret) AC (call flag).
⑤ IO & Machine Control Instructions.

Data transfer Instructions

- These instructions are used to transfer data inbetween Register to Register and Register to memory.

① mov R_1, R_2. (transfer data R_2 to R_1).

e.g. - mov A, B

B = 20H, After execution A = 20H.

② mov M, R. (transfer data of Reg. R into m).

H = 30H
L = 40H
m = 3040 H = ____

befor Instruction B = 30 H.

mov m, B

After Instruction m → 3040 H = 30 H.

③ mov R, M (transfer data mem M into Reg. R).

H = 40 H
L = 60 H.
m = 4060 H = 11 H

mov C, m.

After execution C = 11 H

④ mvi R, data (8 bit) (8 bit data → Reg R).

e.g. mvi B, 77H.

B = 77H

⑤ mvi m, (data 8 bit). (8 bit data → Mem. m).

e.g. mvi m, 77 H.

HL = 3040 H = 77 H

⑥ LXI SP, (8 bit data)

e.g. LXI SP, 77H.

SP → 2010 H = 77 H

→ data transfer Instruction does not effects flag Reg.

Arithmetic Instructions

Addition

ADD R → A = A + R
ADD m → A = A + m
ADI data (8 bit) → A = A + data (8 bit)
ADC R → A = A + R + CY.
ADC m → A = A + m + CY
ADC data (8 bit) → A = A + data (8 bit) + CY.

Subtraction.

SUB R → A = A - R.
SUB m → A = A - m
SUI (8 bit) → A = A - (8 bit)
SBB R → A = A - R - CY
SBB m → A = A - m - CY
SBI (8 bit) → A = A - (8 bit) - CY

Increment

INR R → R = R + 1
INR m → m = m + 1
INX Rp → Rp = Rp + 1

Decrement

DCR R → R = R - 1
DCR m → m = m - 1
DCX Rp → Rp = Rp - 1

Special Arithmetic Instructions.

DAD Rp → HL = HL + Rp

Logical Instructions

AND Operation

ANA	R	⟶	A = A AND R
ANA	M	⟶	A = A AND M
ANI	(8 bit)	⟶	A = A AND (8 bit)

→ ANA B
→ e.g. A = 55H
B = 33H.
A = 0101 0101
B = 0011 0011
A = 0001 0001
A = 11H

OR Operation

ORA	R	⟶	A = A OR R
ORA	M	⟶	A = A OR M
ORI	(8 bit)	⟶	A = A OR (8 bit)

XOR Operation

XRA	R	⟶	A = A ⊕ R
XRA	M	⟶	A = A ⊕ M
XRA	(8 bit)	⟶	A = A ⊕ (8 bit).

Comparision

CMP	R	⟶	Comparision of A and R. (A-R)
CMP	M		
CPI	(8 bit)		

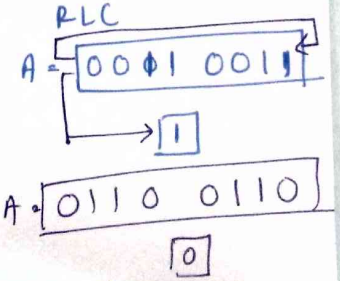
A > R , Cy = 0, Z = 0
A < R , Cy = 1 , Z = 0
A = R , Cy = 0 , Z = 1

Rotate

RLC → Rotate data Left without Carry
RRC → Rotate data Right without Carry.
RAL → Rotate data Left with Carry
RAR → Rotate data Right with Carry.

A = 33H. , Cy = 1

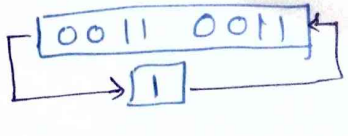
RLC



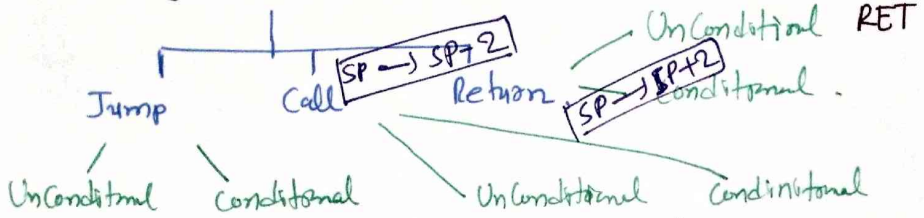
Special Operation

CMA → $A = \overline{A}$
CMC → $Cy = \overline{Cy}$
STC → Cy = 1

RLA



Branching Instructions



JMP (Add)

CALL (Add)

Conditional Jump.		Conditional Call		Conditional Return
Z = 0 (Zero)	JNZ Add	CNZ Add		RNZ
Z = 1 (Not Zero)	JZ Add	CZ Add		RZ
C = 0 (No Carry)	JC Add	CC Add		RC
C = 1 (Carry)	JNC Add	CNC Add		RNC
P = 1 (even)	JPE Add	CPE Add		RPE
P = 0 (odd)	JPO Add	CPO Add		RPO
S = 0 (+ve)	JP Add	CP Add		RP
S = 1 (-ve).	JM Add.	CM Add		RM

GATE Examples on Microprocessor Programming [8085]

① The following five Instructions were executed on 8085 microprocessor

MVI A, 33H → A = 33H

MVI B, 78H → B = 78H

ADD B → A+B = A = ABH

CMA → $\bar{A}$ = A = 54H

ANI 32H → 32H AND A = A = 10H

$$\begin{array}{r} 33 \\ 78 \\ \hline AB \end{array}$$

A = 1010 1011

$\bar{A}$ = 0101 0100

= 54H

54H → 0101 0100

32H → 0011 0010

0001 0000

= 10H

The accumulator value immediately after execution of 5th Instruction is

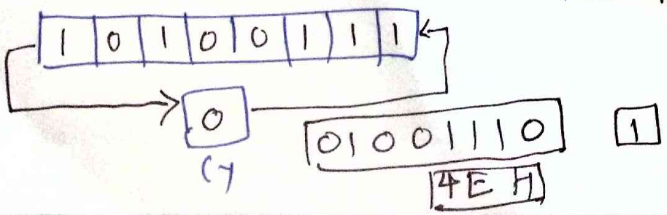
(A) 00 H ✓(B) 10 H (C) 11 H (D) 32 H

② In an 8085 microprocessor, the content of Accumlator and the carry flag are A7H and 0, respectively. If the Instruction RLC is executed, then the content of accumlator (In Hex) and carry flag, respectively will be

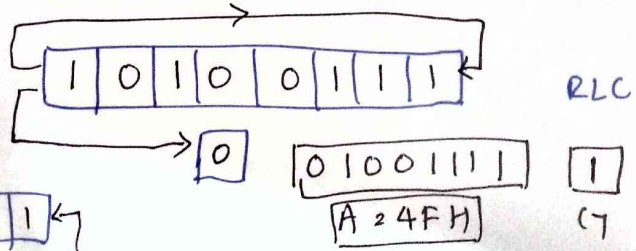
(A) 4E & 0 (B) 4E & 1 (C) 4F & 0 ✓(D) 4F & 1

A = A7H, Cy = 0

RAL



RLC



GATE Examples on Microprocessor programming [8085]

① For 8085 microprocessor, the following program is executed

```
        MVI  A, 05 H  → A = 05 H
        MVI  B, 05 H  → B = 05 H
PTR:    ADD  B       → A + B = A
        DCR  B       → B - 1 = B
        JNZ  PTR
        ADI  03 H    → A + 03 = A
        HLT
```

5 + 5 + 4 + 3 + 2 + 1 = 20 = 14 H ≡ A

14 + 3 = 17 H

= 17H

At the end of program, accumulator contains

(A) 17 H ✓ (B) 20 H (C) 23 H (D) 05 H

② An 8085 assembly language program is given below. Assume carry flag is initially reset. The content of the accumulator after the execution of the program is

```
MVI A, 07 H  → A = 07H
RLC          → A = 0E H
MOV B, A     → B = A = 0E H
RLC
RLC          → A = 38H
ADD B        → B + A = A = 46H
RRC
```

00000111 → 0 → 00001110 (0E), CY = 0

00001110 → 0 → 00111000 (38 H), CY = 0

38 + 0E = 46 H

01000110 → 0 → 00100011 (23 H), CY = 0

(A) 8C H

(B) 64 H

(C) 23 H ✓

(D) 15 H.

(3) which one the following 8085 microprocessor program correctly calculates the product of two 8-bit numbers stored in registers B and C ?

(A)
```
    MVI  A, 00H
    JNZ  loop
    CMP  C
loop: DCR  B
    HLT
```
B+B+ .... C times
or
C+C+ .... B times.

(B)
```
    MVI  A, 00H
    CMP  C
loop: DCR  B
    HLT
```

✓(C)
```
    MVI  A, 00H
loop: ADD  C
    DCR  B
    JNZ  loop
    HLT
```

(D)
```
    MVI  A, 00H
    ADD  C
    JNZ  loop
loop: INR  B
    HLT
```

```
    MVI A, 00H.
loop: ADD C
    DCR B
    JNZ loop
    HLT
```

```
    MVI A, 00H
loop: ADD B
    DCR C
    JNZ loop
    HLT
```

GATE Examples on Microprocessor programming [8085]

Statement for linked answer questions 1 & 2 an 8085 assembly language program is given below

1. MVI A, B5 H → A = B5H
2. MVI B, 0E H → B = 0E H
3. XRI 69 H → A ⊕ 69H → A = DC H
4. ADD B → A+B = A = EAH
5. ANI 9B H → A AND 9BH. A = 8AH
6. CPI 9F H → Compare A with 9FH. A - 9F (-Ve) Cy = 1, Z = 0 (A < data)
7. STA 3010 H → 3010 H = A
8. HLT

BF → 1011 0101
69 → 0110 1001
1 1101 1100 = DC H

0E H
EA H = 1110 1010

1001 1011
1000 1010 = 8AH

(1) The contents of the accumlator just after execution of ADD instruction in Line 4 will be

(A) C3 H (B) EA H (C) DC H (D) 69 H

(2) After execution of line 7 of the program the status of Cy and Z flags will be

(A) Cy = 0, Z = 0
(B) Cy = 0, Z = 1
(C) Cy = 1, Z = 0
(D) Cy = P, Z = 1

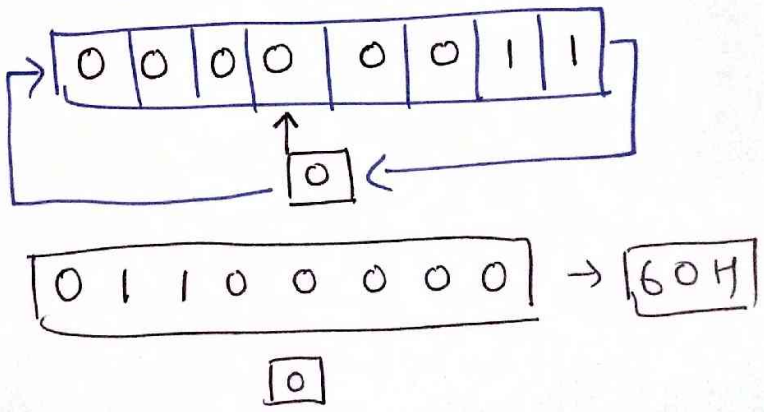
(B) In an 8085 microprocessor, the following program is executed

Address location	Instructions	
2000 H	XRA A	A ⊕ A → A = 00
2001 H	MVI B, 04 H	B = 04
2003 H	MVI A, 03 H	A = 03
2005 H	RAR	↺ 4 times
2006 H	DCR B	
2007 H	JNZ 2005	
200A H	HLT	

At the end of program, register A contains

(A) 60 H (B) 30 H (C) 06 H (D) 03 H

A = 03 H →



GATE Examples on Microprocessor programming [8085]

[Stack, Push & Pop] Push & Call → SP = SP-2, Pop & Ret → SP = SP+2

① following is the segment of a 8085 assembly language program

LXI SP, EFFF H → SP = EFFFH

CALL 3000 H → SP = SP-2 = EFFD H

3000H: LXI H, 3CF4 H → HL = 3CF4 H

Push PSW → SP = SP-2 = EFFB H

SPHL → SP = HL = 3CF4 H

POP PSW → SP = SP+2 = 3CF6 H

Ret → SP = SP+2 = 3CF8 H

On completion of RET execution, the contents of SP:

(A) 3CF0 H (B) ✓ 3CF8 H (C) EFFD H (D) EFFF H

② A portion of the main program to CALL a Subroutine in an 8085 environment is given below as

LXI D, (DISP) on Stack

LP: CALL SUB (LP+3) Normally control without effect

It is desired that control be returned to LP + DISP +3 when the RET instruction is executed in subroutine. The set of instructions that precede the RET instruction in the subroutine are

(A) POP D
X DAD H
PUSH D

(B) POP H
X DAD D
INX H
INX H
INX H
PUSH H
LP+3+DISP+3

(C) ✓ POP H
DAD D
Push H
HL + PE = HL

(D) XTHL X
INX D
INX D
INX D
XTHL

GATE Examples on Microprocessor Programming [8085]

The program & machine code for an 8085 µp are given by the starting address of the program is 7FFFH. what would happen if it is executed from 8000 H.

data	Instruction	
3E	MVI A, C3H	7FFFH
C3		8000 H ← PC
00	NOP	8001 H
80	ADD B	8002 H
3D	DEC A	8003 H
C2	JNZ 800A	8004 H
0A		8005 H
80		8006 H
C3	Jmp 800C	8007 H
0C		8008 H
80		8009 H
D3	Out 10	800A H
10		800B H
76	HLT	800C H.

From Instructions given below. how many memory Operations (Read / Write) are performed during the execution of 8085 µp

① CALL 2000 H

M_1 – Opcode fetch
M_2 – Read High Order Add.
M_3 – Read lower Order Add.
M_4 – write high Order Add of PC on Stack
M_5 – write low Order Add of PC on Stack

② LDA 2000 H

M_1 – Opcode fetch
M_2 – Read high Order Add
M_3 – Read low Order Add
M_4 – Read data from 2000 H

Write an Instruction which takes the minimum possible time to clear the accumulator of 8085.

		Byte	Oper^n n	T-State.
✓ XRA A	→	1	1	4T
ANI 00H	→	2	2	7T
MVI A, 00H	→	2	2	7T
STA (16 bit Add)	→	3	6	13T
✓ SUB A	→	1	1	4T

(3n+1)

GATE Examples on Microprocessor Programming [8085]

The following Instructions have been executed by 8085 µp

Address (Hex)	Instructions	
6010	LXI H, 8A79H	→ HL = 8A79H (H = 8A, L = 79)
6013	MOV A, L	A = L = 79H
6015	ADD H	A = A + H = 03H
6016	DAA (Decimal Add)	A = 68H
6017	MOV H, A	H = A = 68H
6018	PCHL	PC = HL = 6879H

```
    1111←
79 → 0111 1001
8A → 1000 1010
     ++
CY=1 ← 0000 0011 = 03H
       0110 0110
     ---------
       0110 1000 = 68H
```

From which address the next Instruction be fetched

(a) 8019H (b) 0379 (c) ✓ 6979 (d) None of these.

An 8085 µp uses 2 MHz Crystal. Find the time taken by it to execute the following delay subroutine, Inclusive of the call Instruction in the calling program.

```
Delay : Call Delay   18T
        Push PSW     12T
        MVI A, 64H    7T
Loop :  NOP           4T
        DCR A         4T
        JNZ loop     10T/7T
        POP PSW      10T
        RET          10T
```

(Loop: 100 times)

→ $F_{\mu p} = \frac{f_{Crystal}}{2} = \frac{2\,MHz}{2} = 1\,MHz$

→ $T = \frac{1}{F_{\mu p}} = 1\,\mu sec.$

→ 64H = 100

→ $t = 18T + 12T + 7T + 100(18T) - 3T + 10T + 10T$

$= 1854T = 1854\,\mu sec$

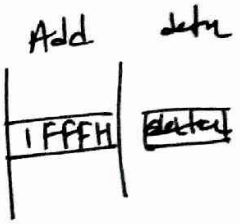
$= 1.854\,msec$

You are given that a Call Instruction takes 18 Cycles of the system Clock, Push req.es 12 Cycles & Conditional Jump takes 10 Cycles if Jump is taken & 7 Cycles if it is not. all Other Instructions used above are takes 3n+1 Clock cycles. where n is the acces to memory Inclusive of Opcode fetch.

GATE Examples on Microprocessor Programing [8085]

Write down the Sequence of Instructions (till HLT Instruction) If the program begins with the location 1FF5 H.

Address		Instructions	
1FF5H	①	XRA A	→ A = A ⊕ A = 00H
1FF6H	②	LXI H, 2000H	→ HL = 2000H
1FF9H	③	PCHL	→ PC = HL = 2000H.
1FFAH		HLT	
1FFBH		LXI H, 2100H	
1FFEH		ANI 00	
2000H	④	LXI H, FFFFH	→ HL = FFFFH
2003H	⑤	INX H	HL = 0000H, C=1, Z=1
2004H	⑥	JZ 2100H	
2007H		HLT	
2100H	⑦	LXI H, 1FFFH	→ HL = 1FFFH
2103H	⑧	MOV A, M	A = data
2104H	⑨	INR A	A = data + 1
2105H	⑩	HLT	



The following Sequence of Instructions are executed by an 8085 µp

1000	LXI	SP, 27FF	→ SP = 27FF H
1003	CALL	1006	→ SP = 27FD H
1006	POP	H	→ HL = 1006 H, SP = 27FF H

The Contents of SP, HL on execution of these Instructions

(a) SP = 27FF, HL = 1003
(b) SP = 27FD, HL = 1003
(c) SP = 27FF, HL = 1006
(d) SP = 27FD, HL = 1006