Lecture-27

M/C control instructions:

EI (Enable interrupts):

The interrupt system is enabled following the execution of the next instruction when this instruction is executed, then INTE F/F is set so that all the interrupts are enabled and 8083 A will recognize external interrupt request except those that are masked.

The operation code format is

1111 1 0 11	$=FB_H$

If is a single byte instruction no variation flags affected.

None the micro RTL flow is

MC-1

OFMC $T_1 : AD_7\text{-}\ AD_0 \leftarrow (PCL)$, $A_{15} - A_8 \leftarrow (PCH)$, ALE = ⎍

IO/$\overline{M}$ = 1 $T_2 : \overline{RD} = 0,$ $(PC) \leftarrow (PC)+1,\ AD_7\text{-}\ AD_0 \leftarrow$ M(AB)

$S_1 = 1$ $T_3 : \overline{RD} = 1, \uparrow$ $(IR) \leftarrow (AD_7\text{-}\ AD_0)$

$S_0 = 1$ $T_4 : EI = 1$

MC-2 $T_1 : \leftarrow$ {FEO into F/F = 1

Thus, it requires one m/c cycle & 4 states.

Note: Interrupts are not recognized during The EI instruction placing an EI instruction on the bus response to $\overline{INTA}$ during an INA cycle is prohibited.

DI (disable interrupts):

The interrupt system is disabled immediately following the execution of the DI instruction. The macro ETL implemented is,

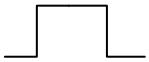
INTE F/F ⟵ 0

It is a single byte instruction. The operant code is

11110011	⇨	$F3_H$

It has no variation No flag affected. The micro RTL flow is

MC-1

OFMC T_1 : AD_7- AD_0 ← (PCL) , A_{15} – A_8← (PCH), ALE = 

IO/$\overline{M}$ = 1 T_2 : $\overline{RD}$ = 0, (PC) ← (PC)+1, AD_7- AD_0 ← M(AB)

S_1 = 1 T_3 : $\overline{RD}$ = 1, ↑. (IR) ← (AD_7- AD_0)

S_0 = 1 T_4 : DI = 1

MC-2 T_1 : ←{FEO into F/F = 0

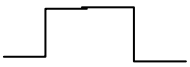
Thus it requires 4 states of OFMC.

Note: interrupts are not recognized during the DI instruction. Placing a DI instruction on the bus in response to $\overline{INTA}$ during an INA cycle is prohibited.

3) NOP: (no operation). No operation is performed. The registers and flags are unaffected the µp idles for 4 states. The operation code is

0000 0000	00_H

It is a single byte instruction. It has no variation. The micro RTL flow is



OFMC T_1 : AD_7- AD_0 ← (PCL) , A_{15} – A_8← (PCH), ALE =

IO/$\overline{M}$ = 1 T_2 : $\overline{RD}$ = 0, (PC) ← (PC)+1, AD_7- AD_0 ← M(AB)

S_1 = 1 T_3 : $\overline{RD}$ = 1, ↑. (IR) ← (AD_7- AD_0)

S_0 = 1 T_4 : NOP = 1

Thus one m/c of 4 states is required. This instruction has three main uses;

1) It is frequency used in delay loops to introduce a delay of 4 states.
2) It is also used to interface slower peripheral devices with 8085 A.
3) It is also used to remove or introduce any instruction in the program

HLT (Halt): when this instruction is executed the processor is stopped. The registered & flags are unaffected.

When this instruction fetched & decoded during T_4 state of instruction cycle, then an in the immediate next T_1 state this HALT F/F is checked to see whether it is set if it is formed SET, µp goes to HALT state. The micro RTL implemented is

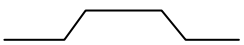
HALT F/F ← 1

The operation code is 0111 0110 $=516_H$

It is a single byte instruction & it has no variation.

The micro RTL flows

OFMC

T_1: $AD_3 - AD_0$ ← (PCL), A_{15} - A_8 ← (ACH), ALF = 

T_2: $\overline{RD}$=0, (PC) ← (PC) +1, (AD_7 - AD_0) ← M (AB)

T_3: $\overline{RD}$=1, ↑, (IR) ← ($AD_3 - AD_0$)

T_4: if HALT F/F = 1, go to halt state.

Thus it requires 5 states flag affected none.

RIM & SIM:

The 8085A has three inputs, RST 5.5 RST 6.5, RST 7.5. Which are referred to as vectored interrupts. A proper input on these 8085A inputs will cause the program being executed by the CPU to branch to some known locations.

I.e. 00 2C H for RST5.5

00 34 H for RST6.5

00 3C H for RST7.5

These locations should have the first instruction of the corresponding service routine i.e. the jump instruction. The RST 5.5 & RST 6.5 input are level sensitive here as RST 7.5 is edge sensitive the RST5.5 & RST 6.5 input will have to be held high till the 8085A acknowledge the interrupt. A low to high transition of RST7.5 sets an internal flip flop so this input can then be lowered without losing the interrupt request.

The 8085 also has an INTR interrupt & one non mask able vectored interrupt called TRAR. A high at the TRAR input after a low to high edge causes the program to branch to 0024H.

The 8085A also provides on chip facility for serial I/O via the SID& SOD lines. The SID line can be used to input serial data to the 8085A. While the SOD line outputs serial data from the 8085A. Inputting & outputting these bits are also achieved using the RIM & SIM instruction

SIM (Set interrupt mask):

This is a single byte instruction. It operates on the contents of A before SIM is executed the operate is

0011 0000	= 30 H

The SIM instruction uses the contents of the accumulation to perform the following functions:

(i) Program the interrupts mask for the RST 5.5, RST 6.5, & 7.5 hardware interrupts. The interrupts can be masked or unmasked by controlling A_0, A_1, & A_2 bits of occur before using SIM. If these bits are the interrupt is unmasked. These bits will affect the interrupts only if MSE (mask set enable) bit (A_3) is also 1. If A_3 is '0'

When the SIM instruction is executed, the interrupt mask register not changed.

(ii) RST 7.5 is edge sinister (LO). A pulse at the RST 7.5 always sets an internal RST 7.45 F/F, even if the jump to the service routine is inhibited by masking If interrupts are disabled at the time the RST 7.5 pulse occurs, this input will still be recognized later since the RST 7.5 F/F has can be cleared (or Reset) by keeping bit A_4=1 when SIM is executed, this may be required if we do not want to service an earlier RST 7.5 interrupted.

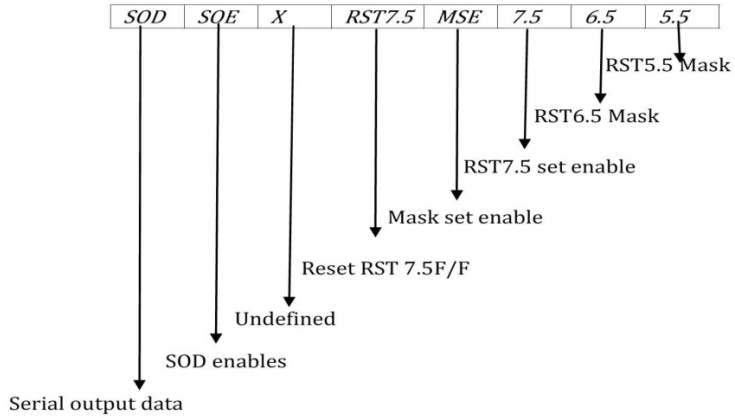
The RST 7.5 F/F is also cleared by a $\overline{RESET\ IN}$ signal input or when the CPU acknowledges a RST 7.5 interrupt. All three mask bit (A_2, A_1, A_0) are also set by its all vectored interrupt (except) are masked and therefore disabled are masked and therefore disabled.

(iii) Load the SOD (serial output data) latch.

The SOD output of the 8085A shown the status of a 1-bit output port, execution of the SIM instruction sets the output port content to that of A_7 provided the serial output enable A_6 is also

1. If SOE = A_6=0, the contents of the output port is unaffected SOD is reset by the $\overline{RESET\ IN}$ input.

Using SOE & MSE properly one can use the SIM instruction is different ways the format of A_{ce} for SIM instruction is



A_{ce} content before SIM

This instruction requires are m/c cycle of four states no flags affected. E.g. (1) send a1 to the SOD output line. The following sequence of instruction shall do the word.

MVI A, CO (1100 0000)

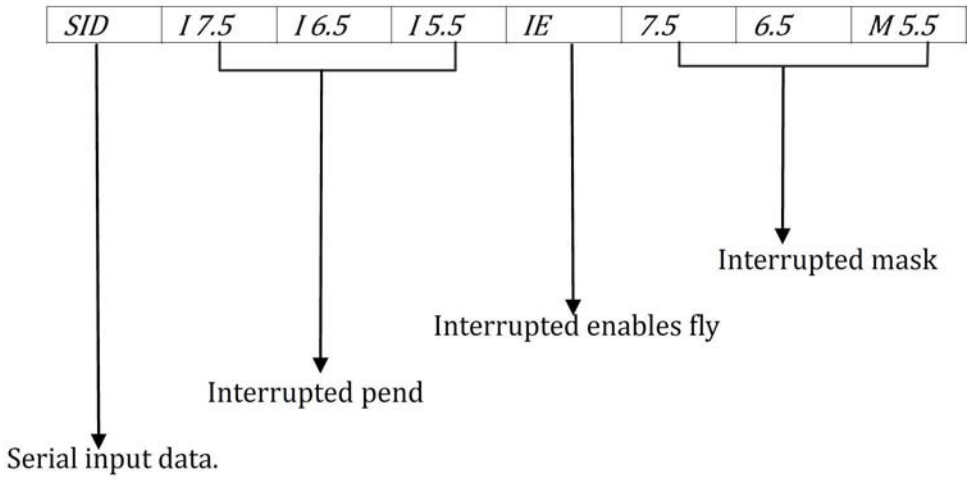
SIM:

(2) The following sequence of instruction shell unmask RST 6.5 interrupt control signal input & mask all other after power on.

MVI A, OD (0000 1101)

SIM, EI:

RIM: (Read interrupt mask): Whenever their instruction is executed the status of the mask F/F INTE F/F the SID input & if any interrupts are pending is read into accumulation as follows.



The RIM instruction loads data into accumulation relating to interrupts & the serial input. After RIM is executed the content of A are as follows:

1) Current interrupt enables status for the RST 5.5, 6.5, 7.5 hardware interrupts in A_0 A_1 & A_2 bits (1= if mask disabled, 0 if they are enabled).

2) Current interrupt enable flag status in bit A_3. (=1 interrupts enabled) except immediately following a TRAP interrupt.

Following a TRAP interrupt (enables/disabled) prior to the TRAP interrupt.

This is useful to retrieve current interrupt since TRAP. This is important since TRAP is a non mask able interrupt which can happen at any time.

3) Hardware interrupts pending. (Received but not serviced). On the RST 7.5, 6.5,& 5.5 line. A '1' at A_6, A_5, A_4, respectively indicate that RST 7.5, 6.5 & 5.5 interrupts are pending.

4) Transfer the bit present at 8085's SID input to A_7.

Apart from the CPU get the SID input, the RIM instruction is primarily used to monitor interrupt status e.g.

a) Monitor whether or not an interrupt is pending without actually servicing it.

b) Check using IE properly if CPU is currently, servicing an interrupt.

c) By properly using RIM & SIM one can design any other prior structure. The operation code is (20)4. 2f is a single byte instruction If requires 4 states.

MC – 1 OFMC – 4

MC – 2 T1 = FEO

T2 = INTERRUPT MASK (~~A)~~ →