Module III

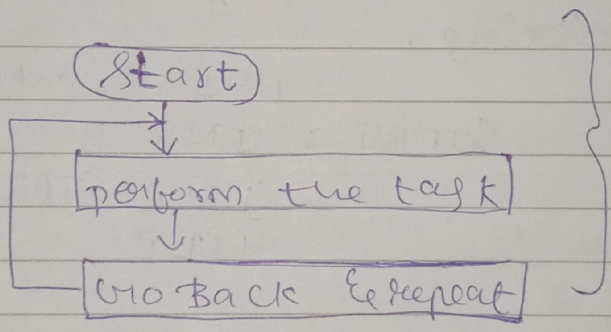
Assembly lang programming techniques.

1) **Looping** =

* Prgming technique used to instruct the µp to repeat tasks → looping
* A loop is set up by instructing the µp to changing the sequence of execution & perform the task again.
* This process is accomplished by using Jump (I).
* Loops are of 2 grps:

a) **continous loop** = (repeats a task continously)

* It is set up by using the unconditional jump (JMP) (I).



flowchart of a continous loop.

* A prgm with a continous loop does not stop repeating the tasks until the system is reset.
* eg →

```
        MVI B, 00H
START:  INR B
        JMP START
        HLT
```

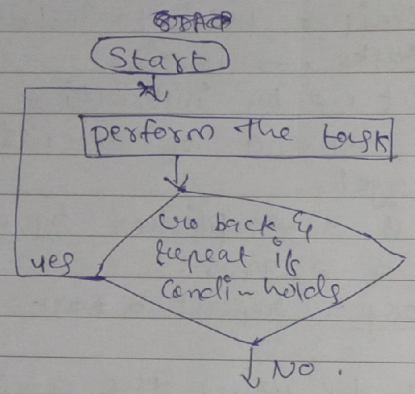
b) **Conditional loop** = (repeats a task until certain data condition is met)

* It is set up by the conditional jump (I):

  Jcondition Addr.

* Condi~ jump (I) checks the flags (S, Z, CY, P) & repeat the specified tasks if the conditions are satisfied.

*



flowchart of a condi~ loop.

* Condi~ loops usually include counting & indexing.
* eg =

```
        MVI B, 00H
START : INR B
        JNZ START
        HLT.
```

2) Counter =

* Typical appli~ of condi~ loop.
* Counter is set up by loading an appropriate count in a (R).
* Counting is performed by either increment or decremnt the counter.
* End of counting is indicated by a flag.

3) Indexing =

* Type of loop include indexing with counter
* Indexing means pointing objects with sequencial nos

4) Time Delay =

* Similar to technique used to set up a counter
* It is achieved by 2 methods :
  a) using a (R)
  b) using a (R) pair

=> Subroutine in 8085 =

* Sequence of prgm (I) that perform a specific task, packaged as a unit.
* This unit can then be used in prgms whenever that particular task has to be performed.
* A Subroutine is coded so that it can be called several times & from several places during 1 execution of prgm.
* It is implemented by using call & return (I).
* Types of Subroutine (I) →

a) Unconditional call (I) =

* CALL adress is the format for unconditional call (I).
* After execution of this (I), prgm control is transfered to a Subroutine.

b) Conditional call (I) =

* Here (I) prgm control is transfered to Subroutine & value of PC is pushed into stack only if condi~ is satisfied.

(I)	Cmnt
CC (16-bit)	call at adr if CY = 1.
CNC "	" " CY = 0.
CZ "	" " Z = 1
CNZ "	" " Z = 0.

(c) Unconditional return (I) =
* RET is the (I) used to mrk the end of subroutine.
* It has no paramtr.

(d) Conditional return (I) =
* prgm control is transferred back to main prgm & value of PC is popped from stack only if condn is satisfied

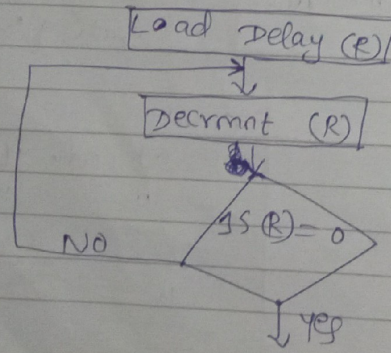
(I)	cmnt
RC	return from subroutine if cy=1
RNC	" " if cy=0
RZ	" " if z=1
RNZ	" " if z=0

(z → zero flag)

Adv * reducing duplicate code within a prgm.

⇒ Delay Generation =

* It is a procedure used to design a specific delay.
* When the delay subroutine is executed, the µp does not execute other tasks.
* For the delay we are using the (T) execution times.



* Delay generation in 8085 µp can be achieved using several methods. —

a) Nop (I) =
* Simplest way is to use Nop (I).
* By executing a series of Nop (I)s, a delay can be generated.

(b) Looping =
* By adjusting the loop counter, the delay time can be varied.

(c) Times / counter =
* 8085 µp has 3 times / counters that can be used to generate delays.

(d) Software delay =
* This can be achieved by continously checking the condition in a loop & only proceeding when the condition is met.

⇒ Stack in 8085 =

* Stack is a spl area of memory used for temporary storage of data during subroutine calls & interrupts.
* Stack in 8085 is implemented as LIFO. Str.
* SP is a 16-bit (R) that points to the current top of the stack.
* PUSH (I) is used to store data onto stack.
PUSH B (BC) It decremnts the SP & then copies the contnts of specified (R) memory onto the stack
* POP (I) used to retrieve data from stack.
POP B (BC) & it incremnts the SP.
* CALL (I) used to call subroutines

* RET (I) used to return from a subroutine.
* stack can be limited in size depending on the available memory.

* Data transfer b/w Mp & stack =

- Data can be transferred b/w the stack & 8085 mp in either direction.
* for eg, data can be pushed onto stack from 8085 (R) / memory loc. & it can also be popped off the stack & into the 8085 (R) / m. loc.
* Stack can be used to temporarily store data during subroutine calls, interrupts & othr operations.
* It is imp to manage the stack carefully to avoid stack overflow errors.

* eg →

LXI SP, 2099H — ⓐ
LXI H, 42A2H — ⓑ
PUSH H — ⓒ
POP H. — ⓓ

2096.	
2097	A2.
2098	42.
2099	

— ⓐ
2099H → 2099H [illegible] m.loc-[illegible] SP point [illegible]

*always data store addr — 1.
(i.e) 2099 — 1 = 2098.

— ⓑ 42A2H [illegible] data H-[illegible] load [illegible].

ⓒ 16 bit { 42 A2. → H L → HL-pair; H → 2098, L → 2097.

*always H points to high m.loc ad (i.e) 2098. & L → 2097.

2096	
2097	
2098	
2099	

← SP. 1st then it decrement (PUSH) — 1 → 2098.

2096	
2097	
2098	42
2099	

← SP. push 42. & decrement — 1. → 2097.

2096	
2097	A2
2098	42
2099	

← SP push A2.

ⓓ POP H. ⇒ 1st pop the top element of stack (i.e) A2. then 42.
↓
retrieve data from stack to HL pair.

⇒ Interrupts in 8085 = (In)

* It is a signal to the processor initiated by hardware / Software indicating an event that need immediate attention.
(def) * Signals generated by the ex. devices to request the Mp to perform a task.
* whn a Mp receives (In) signals, it sends an acknowledgment (INTA) to the peripheral which is requesting for its service.

(TRAP → RST 4.5)

* classified into various categories based on different paramtrs. —
* ISR tells the processor wht to do when (In) occurs.
* 8085 has 4 maskable (In) & 1 non-maskable (I).
* (I) request are classified into 2 → Maskable & non maskable.
* classified into various categories — (7)

(a) Hardware (In) =

* when µp receives (In) through pins → Hardware (In).
* They are TRAP, RST 7.5, RST 6.5, RST 5.5, INTR.

(b) Software (In) =

* Inserted in b/w the prgm.
* They are RST 0, RST 1, RST 2, RST 3, RST 4, RST 5, RST 6, RST 7.

(c) Vectored (In) =

* Have fixed vector address (starting address of subroutine)

(d) Non-vectored (In) =

* vector address is not predefined
* INTR is the only non vectored (In) in 8085.

	In	vector adr
Hardware (I)	TRAP	24H
	RST 5.5	2CH
	RST 6.5	34H
	RST 7.5	3CH.

Software (I)	RST 0	00H
	RST 1	08H
	RST 2	10H
	RST 3	18H
	RST 4	20H
	RST 5	28H
	RST 6	30H
	RST 7	38H.

(e) Maskable (In) =

* can be disabled / ignored by the µp.
* These (In) are either edge-triggered / level triggered, so they are disabled.
* eg → INTR, RST 7.5, RST 6.5, RST 5.5

(f) Non-maskable (In) =

* cannot be disabled.
* These (In) of both edge & level triggered.
* eg → TRAP.

(g) priority (In) =

µp execute the interrupt service request (ISR) according to priority of the (In).

TRAP → RST 7.5 → RST 6.5 → RST 5.5 → INTR.

highest ↓ lowest.

* 8085 (In) process controlled by the (In) Enable flip-flop

* Instruction for (In) =

a) EI (Enable Interrupt) =

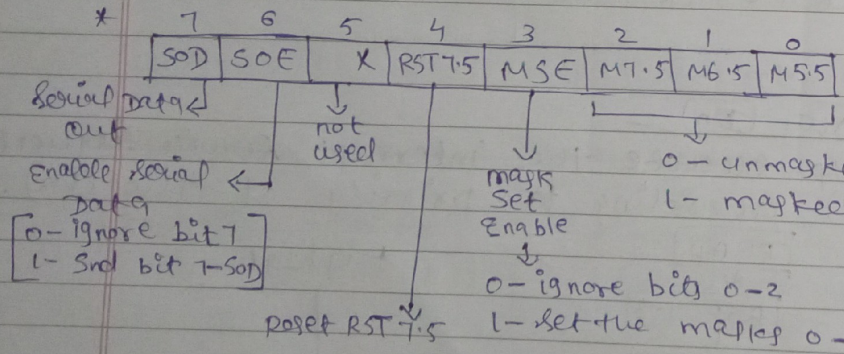
* 1-byte (I)
* Sets the (In) Enable flip-flop & enable (In) process.

b) DI (Disable Interrupts)

* 1-byte (I)
* Reset the (In) Enable flip-flop & disable (In) process.

c) SIM (Set (In) Mask) =

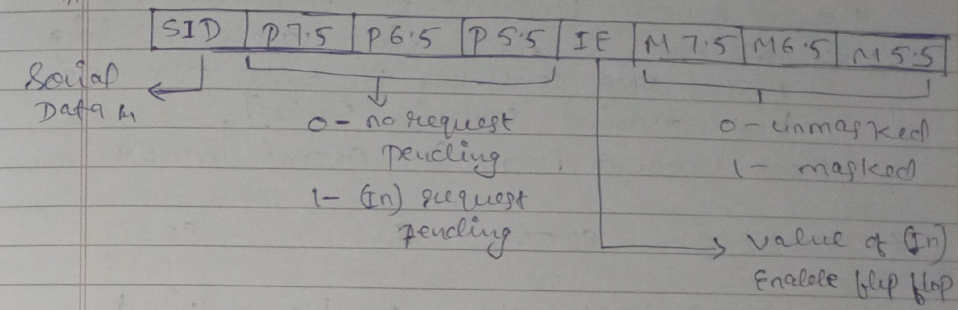
* Used to implemnt the hardware (In) by setting various bits to form masks.



d) RIM (Read (In) Mask) =

* used to read the Status of hardware (In) by loading to A (R).

*



⇒ Interfacing - progming peripheral devices - 8255A, 8254, 8237 =

a) 8255A :

* programable peripheral interface that provides 24 I/O ports that can be programmed as input/output.
* Has 2 control (R) that can be used to configure the mode of opration.

b) 8254 :

* It is a programmable intrval times that can be used to generate timing signals in a system.
* Has 3 independent 16-bit counters that can be progrmed to operate diffrnt modes.

c) 8237 :

* It is a direct memory access (DMA) Controller that manages data transfer b/e CPU & peripheral devices.

* 8055A, 8054 & 8037 are not MP.
They are Integrated circuits (ICs) that serve specific fns within a comp system.

* offers several features to enhance its functionality, including cycle stealing [allows DMA controller to take over the bus during certain cycles], block transfer mode & demand mode.

*	Features	8255A	8254	8237
*	function	prgmble peripheral Interface	prgmble Interval Timer	Direct memory Access Controller
*	no of I/O ports	24	No	No
*	Channels	24 I/O lines	3 counter/ timers	4 DMA channels
*	Main purpose	I/O interface	Timer/ counter	Data transfer b/w CPU & devices.
*	Compati-bility	compatible with Intel 8085 & 8086	compatible with Intel 8086/88	compatible with Intel 8086/88