UNIT - (3)

PAGE NO. .....

Advance Assembly Language Programming -

COUNTER: A counter is designed by loading an appropriate count in a Register. A loop is set up to decrement the count for the down counter or the increment the count for the up counter. A Timing delay is designed by loading a register with the delay count and setting up a loop to decrement the count until Zero.

Counter and time delay are important technique. They are commonly used in application such as traffic signal, digital clocks, process control and serial data transfer.

```
1        MVI B, 05
2   Loop DCR B
3        JNZ Loop
4
5
```

COUNTER – A counter is designed by loading an appropriate number into one of the Register and using the increment or decrement the instructions. A loop is designed to check whether the condition reached at the final position.

TIME DELAY – The procedure for design a specific delay and similar to set up the counter. A Register is loaded with a number depending on the time delay required and then the register is decremented for reaching the initial count.

Time delay using one Register –

Opcode	Operand	Comment	T-state
MVI	C, FFH	Load Register C	7
DCR	C	Decrement C	4
Loop: JNZ	Loop	Jump back to decrement C	10/7
		HLT	

16 16
FF
15 15

$15 \times 16 + 15 \times 16^0 + 1 \Rightarrow 255$

PAGE NO. .....

X_1, X_2 – Clock frequency of the system $f = 2MHz$

Clock period $T = \frac{1}{f} = \frac{1}{2 \times 10^{+6}} = 0.5 \mu sec.$

Time to execute MVI = 7 T-state $\times 0.5 \mu sec$

$= 3.5 \mu sec.$

Time delay in the Loop –

T_L = time delay in the loop

T = System clock period.

N_{10} = Equivalent decimal number of the hexadecimal count loaded in the delay Register.

$T_L = (0.5 \times 10^{-6} \times 14 \times 255)$

$= 1785 \mu sec.$

≈ 1.8 m sec.

The T-state for jump instruction is 10/7. The 8085 µP requires 10T-state to execute a conditional Jump. When it jumps to change and for sequence the program.

$T_{LA} = T_L - (3T\,state \times clock\ period)$

$1785.0\,\mu s - 1.5\,\mu s = 1783.5\,\mu s$

So the total delay for execution the instructions.

Total delay = Time to execute instruction outside loop.
+
Time to execute loop instruction.

$$T_D = T_O + T_{LA}$$
$$= (7 \times 0.5\,\mu s) + (1783.5\,\mu s)$$
$$= 1787\,\mu s.$$
$$\approx 1.8\,ms.$$

The time delay can be varied by changing the count FFH.

PAGE NO. .....

Time delay Using a loop within a Loop Technique -

A time delay can also be achieved by using two loops. Register C is used in the inner loop and Register B is used for the outer loop.

```
        MVI  B, 38H.      7T
Loop2:  MVI  C, FFH.      7T
Loop1:  DCR  C            4T
        JNZ  Loop1        10/7 T
        DCR  B            4T
        JNZ  Loop2        10/7 T
```

37

Delay Calculation -

The delay for the Loop1 is $T_{L_1} = 1783.5$ µs and for loop2 it is calculated by the following loop.

38
16' 16°

3X16+8 =5

$$T_{L2} = 56(T_{L1} + 21\ T\text{-state} \times 0.5\ \mu sec)$$

$$= 56(1783.5\ \mu sec + 10.5\ \mu sec)$$

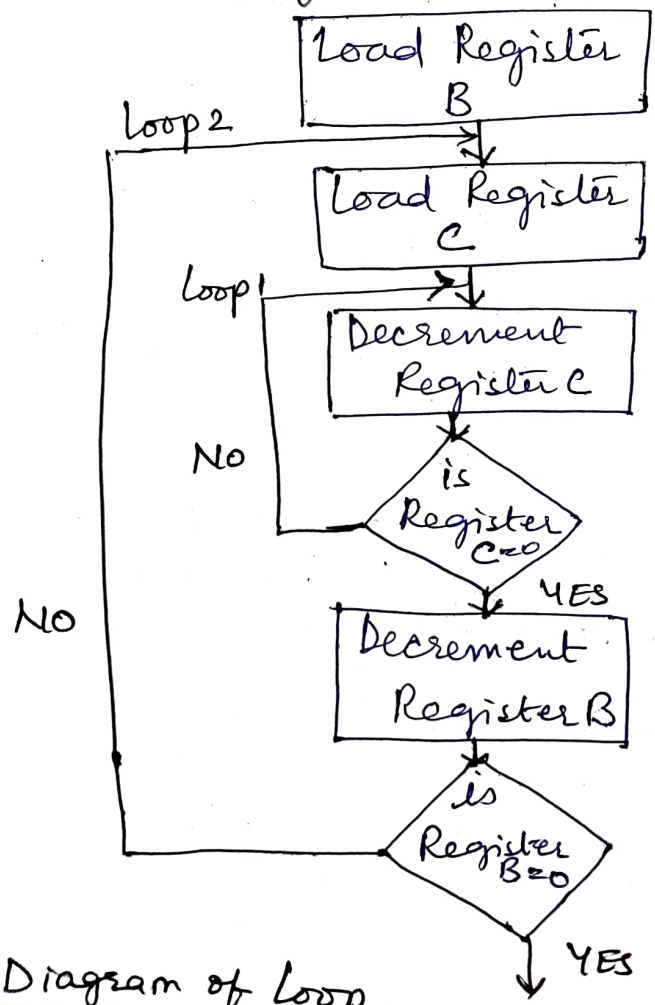
$$= 100.46\ ms.$$

The total delay include the execution time for the first instruction and the delay outside the loop provide the significant execution time.

The time delay within a loop can be increased by using instruction so that it not affect the program execution time and metoring process.

2 pt × 1/2

10.5



Flow Diagram of Loop.

PAGE NO. .....

Time Delay using a Register Pair :

It can be increased by setting a loop and using a Register pair with a 16 bit number (FFFFH). The 16 bit number is decremented by using the instruction DCX. The instruction DCX does not set the zero flag and without testing the test flag, jump instruction can not check desired data.

Label	Opcode	Operand	Comment	T-state
	LXI	B, 2384H.	Load BC with 16 Bit Count	10
Loop:	DCX	B	dec^n BC by one	6
	MOV	A, C	place content of C in A	4
	ORA	B	OR(B) with c. to set zero flag.	4
	JNZ	Loop	if result $\neq$ 0 jump back to loop	10/7

Time Delay:

$$(2384)_H = 2\times(16)^3 + 3\times(16)^2 + 8\times(16^1) + 4(16^0)$$
$$= 9092_{10}$$

Δt = the clock period of the system.

$$= 0.5\,\mu\text{sec}.$$

$$T_L = 0.5 \times 24 \times 9092_{10}$$

$\approx$ 109 ms (without adjusting last cycle)

Total delay $\Rightarrow$ $T_D = 109\text{ ms} + T_0$

$$= 109\text{ ms}.$$

MACROS :-

A macro is a named block of assembly language statement. Once defined it can be involved as many program as we wish. When the macro can be invoked the code is inserted directly into the program at the location where it was invoked.

This type of automatic code insertion, is known as inline expension. It is refer. to calling a MACRO. In that program the call instruction is not specified by the programmer.

* creating macro is similar to create a new opcode that can be used in the program.

Declaring macros are defined directly at the beginning of a source program or they are placed in a separate file & copied into a program by an include directive. Macros are explained during the assembler steps.

At every point when the macro is called. the assembler insert a copy to the macro source code into the program.

Defining Macros -

It is defined by MACRO and END In directions. The syntax is -

```
macro name   MACRO
             Parameter 1
             Parameter 2
             :
             :
             :
             END M.
```

There is no fixed rule regarding indentation but the indent statement between macro name & END m Specified. the value defination term.

* Macro sequences execute faster than subroutines because there are no CALL & RET instructions. to execute -

Subroutine :-

It is a group of instruction written separately from the main program to perform a function that occur repeatedly in the main program. If the time delay is required between three successive event, three delay can be written in the main program.

To avoid repetition of the same delay instruction the subroutine technique is used. Delay instruction are written once, separately from the main program.

It has two instruction to implement subroutine Call & Ret.

The call instruction is used in the main program to call a subroutine & RET instructions is used at the end of the subroutine to return the main program. When a subroutine is called the content of the program Counter, which is the address of the instruction following the call instruction.

Call : 16 bit memory addres

Call subroutine unconditionally -

- it is the 3 byte instruction that transfer the program sequence to the subroutine address.
- Save the content of the PC on the stack.
- Decrement the stack Pointer Register by two.
- Jump unconditionally to the memory location. Specified by second & third byte.
- The instruction is accompanied by RET instruction.

RET : → Return from subroutine unconditionally

→ 1 Byte instruction

→ Insert 2byte from top of the stack into the PC & increment the SP. Register by two.

→ Unconditionally RET from the Subroutine.

STACK IMPLEMENTATION :-

The stack in an 8085 described as a set of memory location in the R/W memory specified by the programmer in the main memory contend.

The beginning of the stack is defined by using the instruction LXI SP, which load a 16-bit memory address in the stack pointer Register.

Once, the Stack location is defined, storing of data byte begin at the memory location that is one less than the stack pointer Register.

Data byte in the Register Pair can be stored on the stack in the reverse order. By using the instruction push. Data byte can be transferred to the next memory location. By POP instruction.

The stack is shared by the programmer & the microprocessor. The Programme can store & retrieve the information automatically.

Instruction for Stack Implementation :

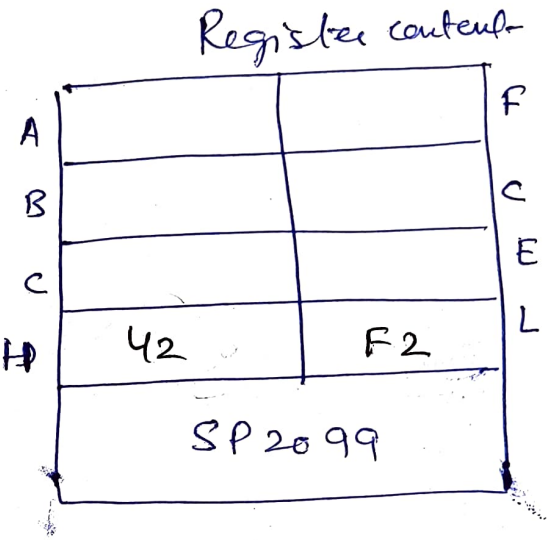
* LXI SP, 16 Bit ⇒ Load stack pointer with the 16 bit memory address.
* PUSH Rp ⇒ Store Register Pair on stack.
* PUSH B ⇒ Copies the content of Specified Register pair on the stack.
* PUSH PSW ⇒ Stack Pointer Register is decrement & the content of high order Register one copied in the memory.

  ⇒ Stack Pointer Register is again decremented & content of low order Register copied in that location.
* POP Rp ⇒ Retrieve Register Pair from stack.
* POP B ⇒ Copies the content of 2 top memory location of the stack into the specified Rp.
* POP PSW ⇒ First the content defined by SP the Register are copied into the low order Register the content of the next memory location are copied into the high order Register.

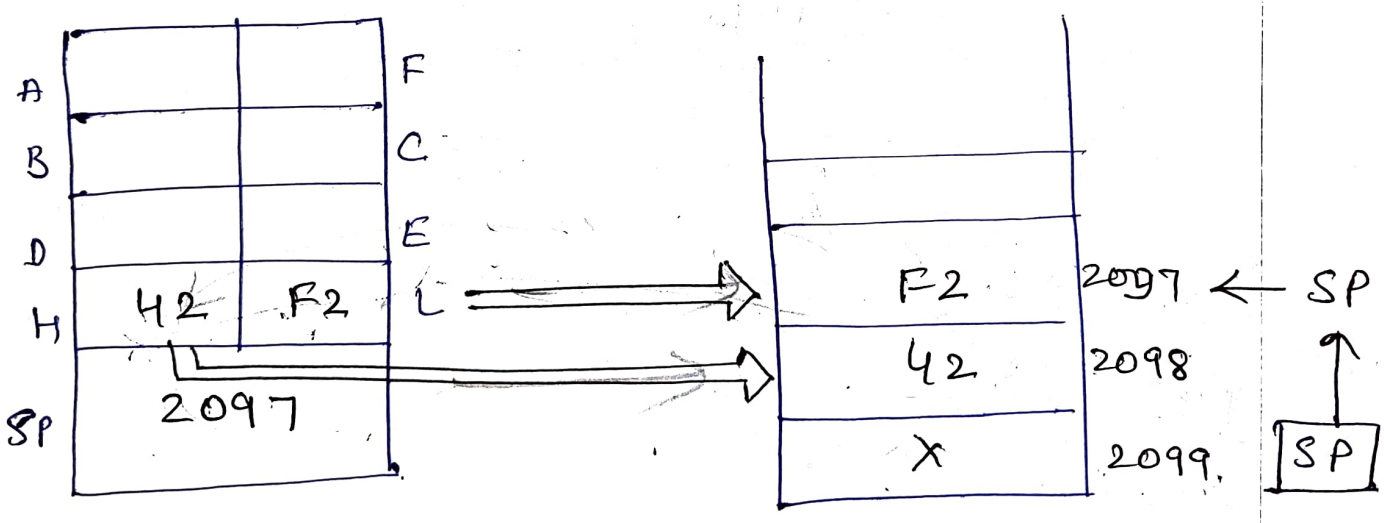
PAGE NO. .....

STACK OPERATION :-

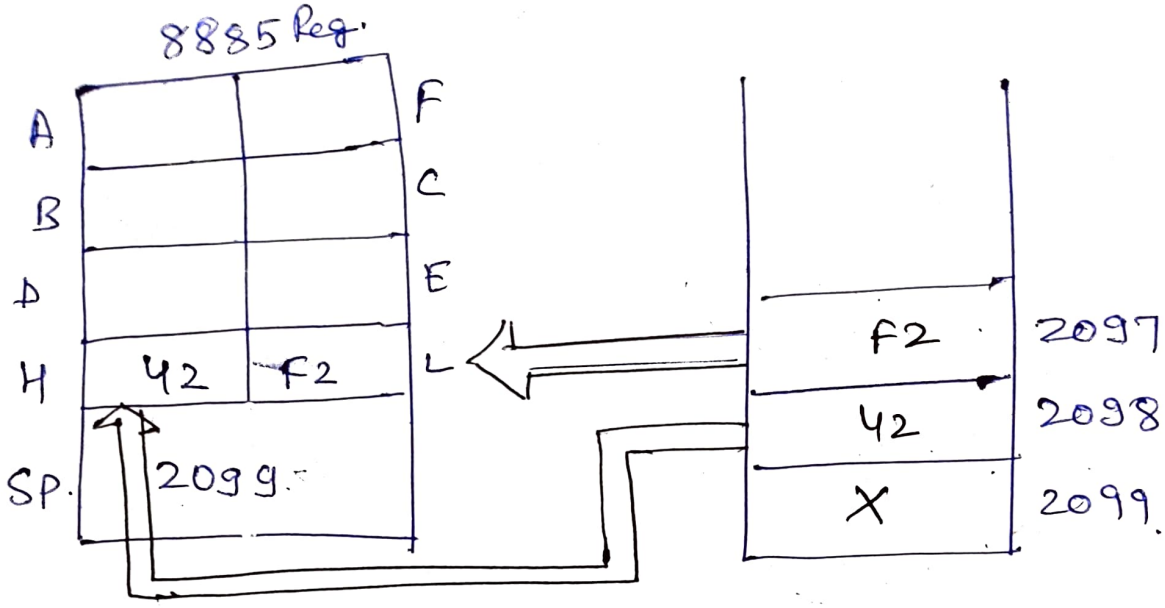
```
LXI SP, 2099 H.
LXI H, 42F2 H.
PUSH H.
Delay Counter
POP H.
```



Instructions & Register Content-



Register After the PUSH instruction.



Register after the POP instruction.

- The stack pointer is decremented by one to 2098H and the content of the H Register copied to memory location 2098H.
- The stack pointer is again decremented by one to 2097 & the content of Register L are copied to memory location 2097H.
- The content of Register Pair HL are not destroyed, however HL → is made available for the delay counter.

PAGE NO. .....

Interrupts :-

An interrupt is the process of data transfer whereby an external device or a peripheral can inform the processor that it is ready for communication and it required attention.

The interrupt request can be classified in 2 categories -

→ Maskable Interrupt
→ Non maskable Interrupt

8085 includes 4 maskable interrupts & 01 Non-maskable interrupt.

The microprocessor can ignore or delay a maskable interrupt request if it is performing some critical task. It has to respond to the non-maskable request immediately.

The interrupts proces allows the microphones to respond the external request for attention. or service on a demand basis and leave the microprocesor free to perfoem other task.

8085 Interrupts ?

It is controlled by the interrupt enable flip flop. If the flip flop is enabled & the input of the interrupt signal. This is maskable interrupt & can be disabled.

EI (Enable Interrupt) -

- It is 1 byte instruction
- Set the interrupt enable flip flop.
- System reset or an interrupt disable the interrupt process.

DI (Disable Interrupt) -

- It is 1 byte instruction
- It reset the interrupt enable flip-flop & disable the interrupt.
- It include in a program segment where an interrupt form an outside some cannot be tolerated.

PAGE NO. .....

8085 Vectored Interrupt :-

8085 has 5 interrupt input. One is called. INTR, three one called RST 5.5, 6.5, 7.5 the fifth is called TRAP a non-maskable interrupt.

They do not require the INTA signal or an input part.

Interrupt		Call duration (address)
TRAP	→	0024 H. (Mor request)
RST 7.5 X 8	→	003C H.
RST 6.5	→	0034 H.
RST 5.5	→	002C H

TRAP has the highest priority followed by RST 7.5, 6.5, 5.5 and INTR.

TRAP - is a non-maskable interrupt known as NMI. It has the highest priority interrupt Signals.

it need not be enabled & it cannot be disabled. It is level & edge sensitive, input should be high & stay high to be acknowledge. It cannot be acknowledged again until it makes a transition from high to low to high.

RST 7.5, 6.5 and 5.5 :-

These maskable interrupt are enabled for the program & controlled by 2 instruction EI, SIM.

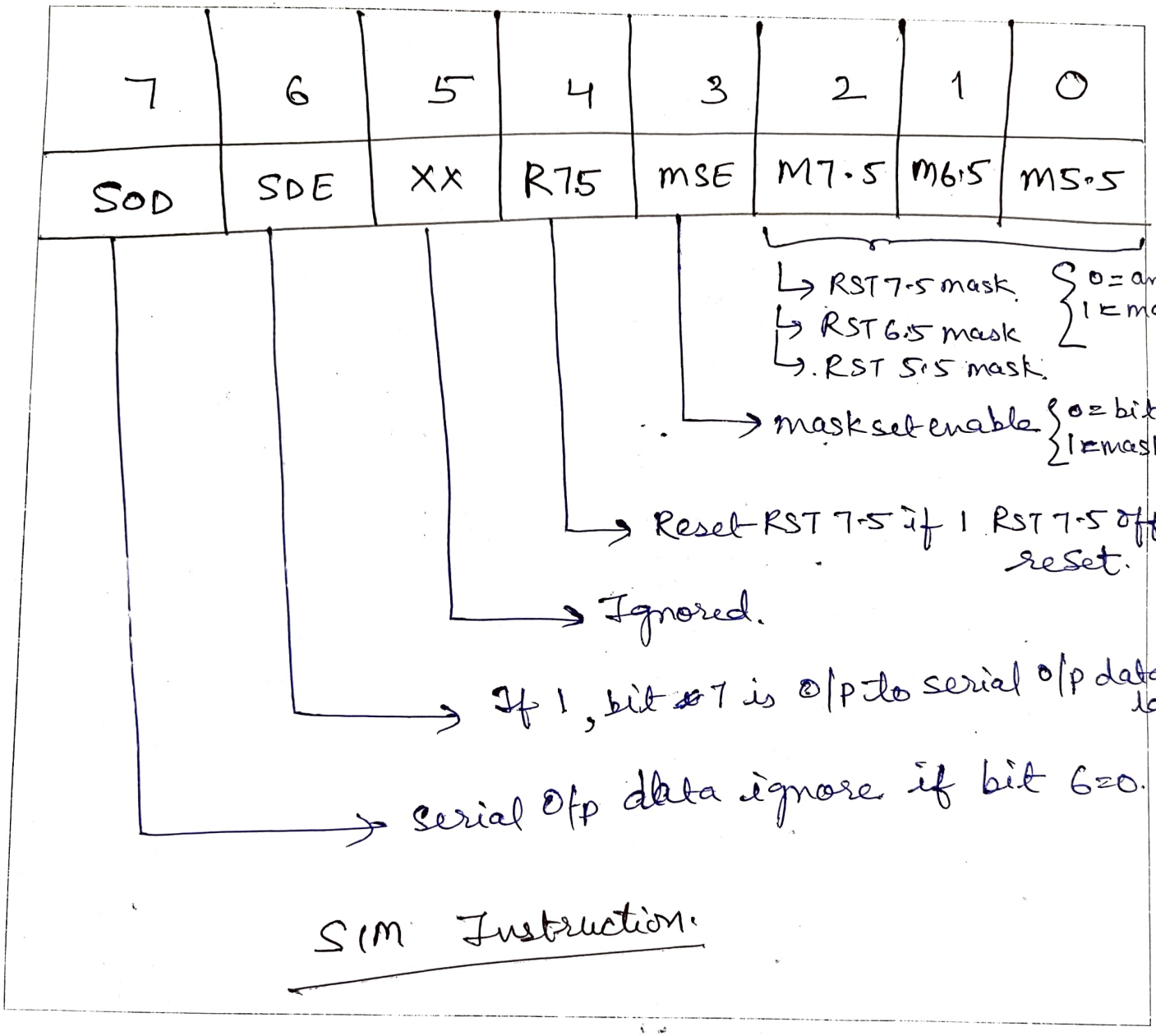
SIM: Set Interrupt Mask :-

It is a 1 byte instruction can be used for three different function -

(1) To set mask for RST 7.5, 6.5 and 5.5 interrupt. The instruction Read the content of the accumulator Bit D_3 is a control bit

(2) To reset RST 7.5 flip flop bit D_4 is additional control for RST 7.5, If $D_4 = 1$, RST 7.5 is reset.

(3) To implement serial I/O Bit D_7 and D_6 of the accumulator used for serial I/O & it do not affect the interrupt.



SIM Instruction.

TRIGGERING LEVELS :-

(RIM (Read Interrupt mask) –

It is one byte instruction that can be used for the following functions –

1. To read Interrupt mask.
2. This instruction load the accumulator with 8 bit the current status of the interrupt mask.
3. To identify pending interrupt.
4. To receive serial Data . Bit D7 is used to receive serial data.

7	6	5	4	3	2	1	0
SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5

↓ interrupt mask.
1 = masked.

→ Interrupt Enable flag
1 = enabled.

→ Pending Interrupt
1 = Pending

→ Serial Input data if any.

RIM Instruction