Code :

1000 : I_1

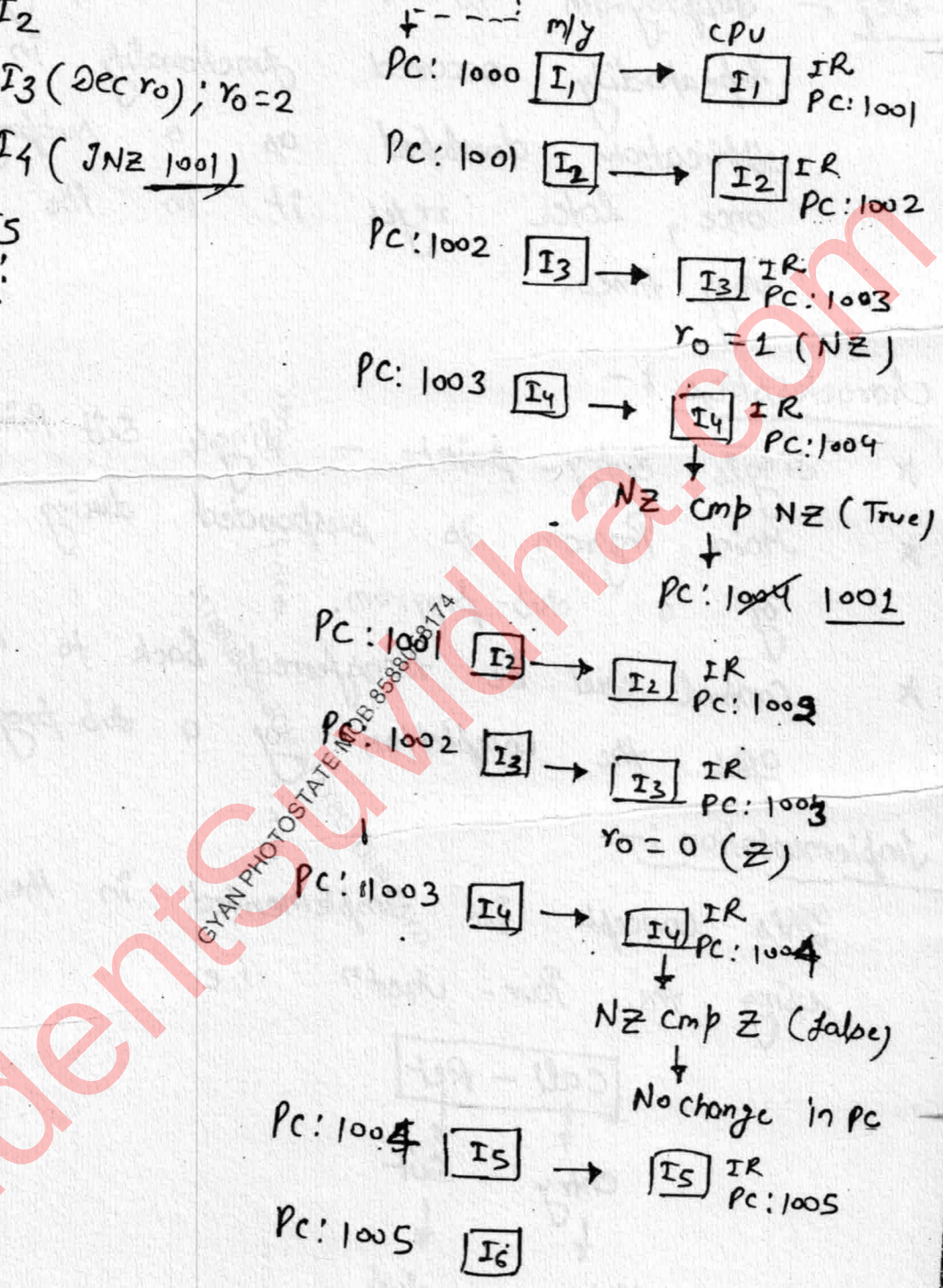
1001 : I_2

1002 : I_3 (Dec r_0) ; $r_0 = 2$

1003 : I_4 (JNZ 1001)

1004 : I_5

Execution :-



* This instn is used to implement the conditional selection statements (If, switch) and various iterative statement (loopp).

Sub-programs :-

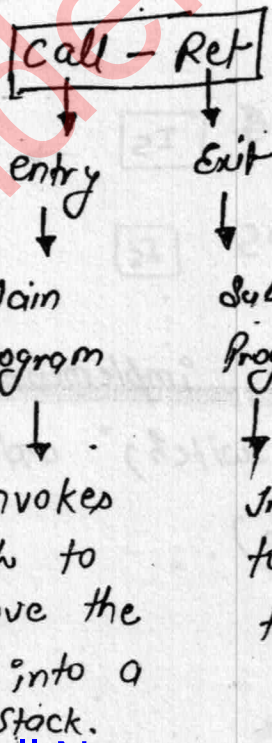
Def^n :- Subprogram is a Re-usable program, means repeatadily occured functionality in the Application developed as a subprogram only once, later refes it in the application many times.

Characterstics :-

* Single Entry-point – Single Exit Point.
* Main Program is suspended during the execution of a sub-program.
* Control will be transferred back to main program after the completion of a sub-program.

Implementation :-

This concept is implemented in the Hardware using the Pair-Instn i.e.

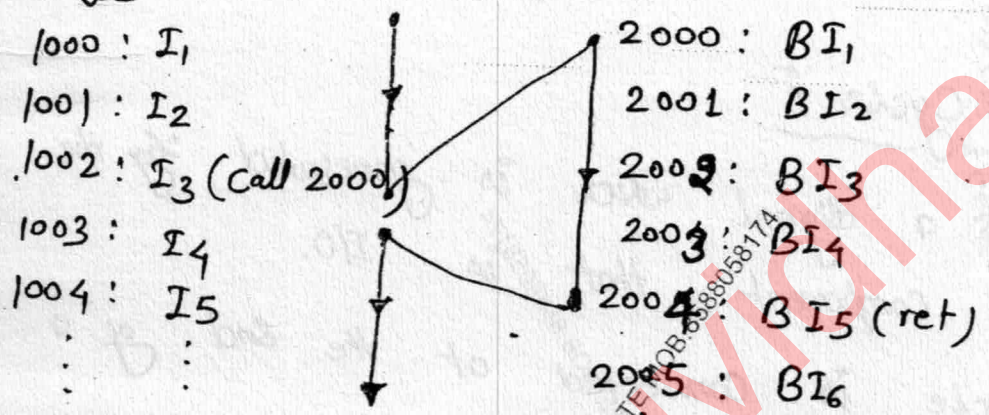


★ In this implementation a stack is used to store the return addresses.

★ Return address is a Next Instn address after the call Instn.

★ It will be varies depends on the calling position.

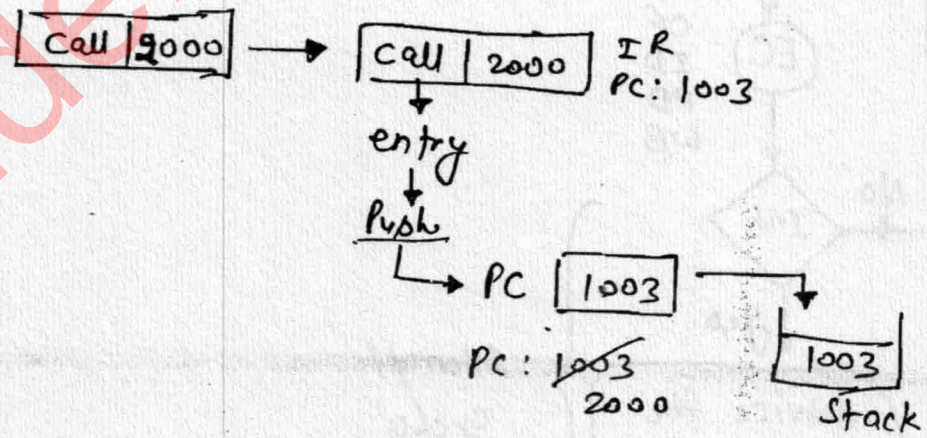
Prog:-



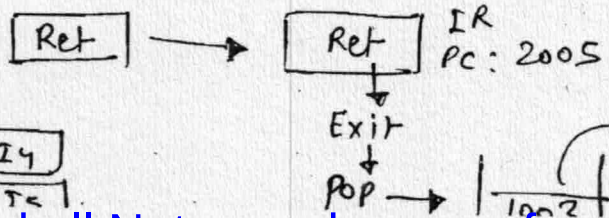
Execution :-

- t : 1000

PC : 1000
PC : 1001
PC : 1002



PC : 2000
PC : 2001
PC : 2002
PC : 2003
PC : 2004



PC: 1003 I4
PC: 1004 IR

PC: 2005
1003

ToC :-

- Jmp 2000
- JNZ 2000
- call 2000
- Ret
- Call Nz 2000
- Ret NZ

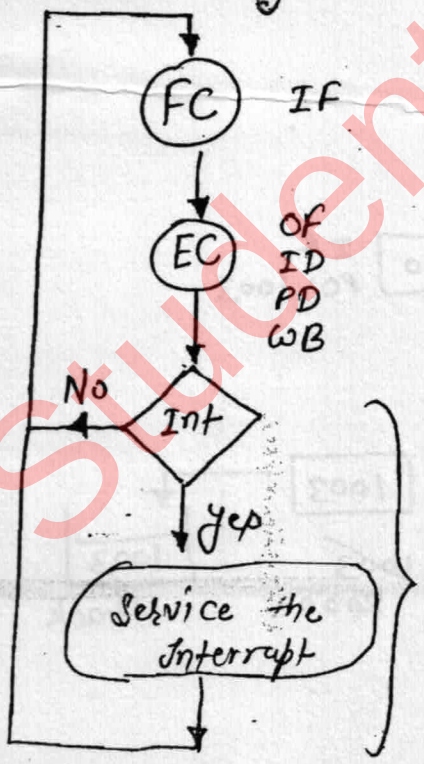
(Transfer of control)
only PC
is changed

Date
16/08/2015

Interuppt Cycle :-

★ Interrupt is a signal which is generated by the low speed component that is I/O.

★ Intersupt cycle is connected at the end of a execution cycle.

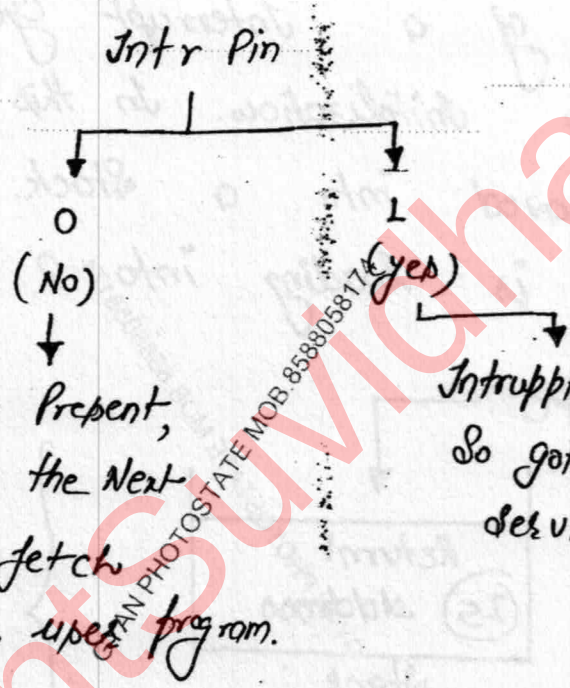


Interrupt cycle

★ Sequence diagram states that, CPU will be respond to a interrupt only after the completion of a current instruction execution.

★ During the Program execution, after the completion of a current instruction execution, CPU reads the status of interrupt pins. to detect the Interrupt

Eg:



★ when the CPU detect the Interrupt, then it saves the PC value in a stack and transfers the control to IVT (Interrupt vector table).

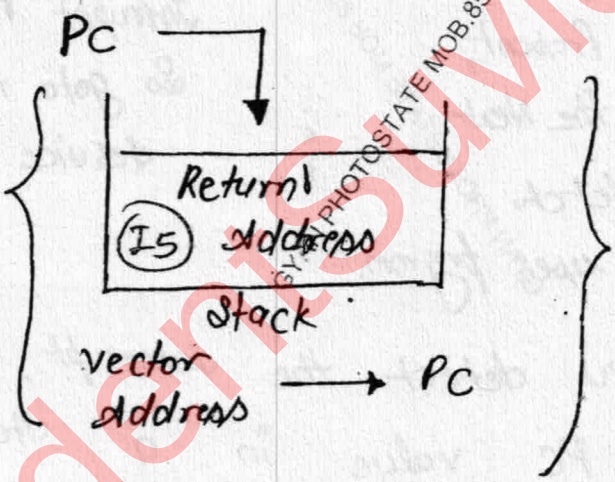
★ IVT is a part of the memory where the Interrupt subprograms address is present

★ Interrupt Subprogram is referred with a vector Address, and with a IRET (Interrupt ~~for~~return) or RETI (Return from Intr.) or RFE (Return from Exception)

* During the Execution of a IRET Instruction invokes the POP operation to restore the PC with a return address therefore after service the Interrupt, control will be transferred back to user program.

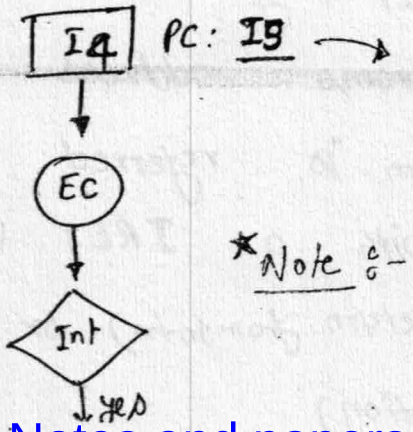
* **Note :-**

objective of a Interrupt cycle is Interrupt subprogram initialization. In this process PC value saved into a Stack. later vector address is loading into a PC.



Eg:- Prog { I_1 to I_{10} }

:- Intrupt occured during the execution of I_4.



→ return Address Pushed to stack → I5

* Note :- If the I_4 is a Halt Instn then I_4 will be pushed to stack

* There are two approaches present to process the interrupts :

i) Single interrupt processing approach

ii) Nested interrupt processing approach.

* In the first approach, CPU disables the further interrupts during the execution of a current interrupt (one after another). (Non-Pre-Emption)

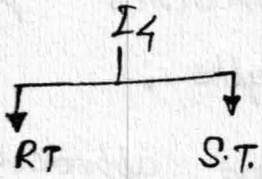
* In the second approach CPU services the interrupt based on the priority assignment (Pre-Emption).

Que:- Consides a Hypothetical processor which supports 4 interrupts {I_1, I_2, I_3, I_4} with the respective service times of 20ns, 10ns, 5ons, 30 ns. response time of a interrupt is 2ns. I_1 has the Highest priority and I_4 has the least priority. what is the Range of time required to service the I_4 interrupt when the interrupts may or may not occur simultaneously.

Soln :- Total time = Response time (R.T) + Service time (S.T)

min. time required for I_4 (without simultaneous occurrence)

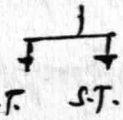
ie.



⇒ 32 ns.

Max time required for I_4 { with simultaneous occurance } ie All are present at a time.

(I_1, I_2, I_3, I_4) ⇒ 118 ns



Range of time req to service

I_4 = { 32 ns - to 118 ns }.

Que:- Consider A Hypothetical CPU with a Average CPI (cycle per instn) of 6 cycles. used to execute the program code which contain 20 instructions. If the Interrupt is occured during the execution of a I_8 instn, After How many cycles CPU will be respond to a interrupt.

Soln:- After execution of I_8 instn, CPU will be respond to interrupt

therefore # of cycles after require to service 8 instn. = 8 × 6 cycles = 48 cycles.

After 48 cycles, CPU will respond to interrupt.

Que:- Consider a 1ns clock cycle processor used to execute the following code: Program is stored in the m/y with a starting address of $(AAA)_H$. word length of CPU is 16 bit. CPU interfaced with a byte addressible m/y m/y reference consumes 4 cycles, ALU operation takes 2 cycles and Reg. Ref. consumes 0 cycles.

Instn	meaning	Size (in words)
1) Mov r0@3000	$r_0 \leftarrow m[[3000]]$	4
2) Mov r1,[4000]	$r_1 \leftarrow m[4000]$	3
3) Add r0,@5000	$r_0 \leftarrow r_0 + m[[5000]]$	4
4) Mul r0, r1	$r_0 \leftarrow r_0 * r_1$	2
5) Mov 3(r1), r0		3
6) Add @r2, r3		4
7) Mov [2000], r2		2
8) Halt		1

a) If an interrupt is occured during the execution of Halt instn what will be the return address pushed into stack.

b) How many cycles are required to access the operands in the Program Execution (Read Book)

c) what is the Program execution time.

Ic

	IF	ID	OF		PD	WB
			S_1	S_2		D
-	1MR	3MR	2MR	—	—	1RR
2	1MR	2MR	1MR	—	—	1RR
3	1MR	2MR	2MR	1RR	1ALU	1RR
4	1MR	1MR	1RR	1RR	1ALU	1RR
5	1MR	2MR	1RR	1RR		1RR 1ALU 1MR
6	1MR	3MR	1RR 1MR			1RR 1MR
7.	1MR	1MR	1RR			1MR
8.	1MR					

MW	CC

25MR
100 cycles
102 cycles

Oprand Access :- 32MR 4ALU

128 8 = 138 cycles

9) word size = 16 bit
= 2B

Byte addressible m/y :- cell size = 1B = 8 bit

m/y storage :- I_1: AAA :- 4w = 8 cells

AAA :- AB1

I_2: 3w = 6 cells:

I_3 :- $AB_2 \rightarrow AB_7$ (4w → 8 cells)

I_4 :- $ABF \rightarrow ABF$ (2w → 4 cells)

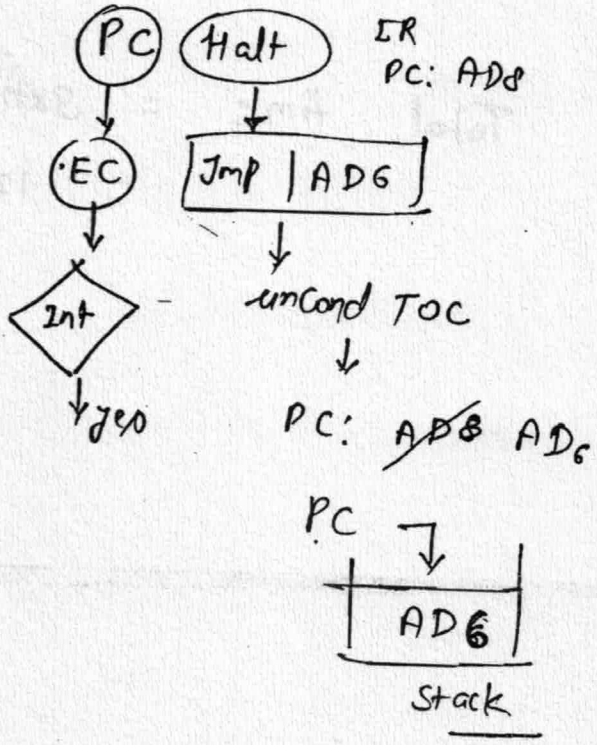
I_5 :- $AC_0 \rightarrow AC_3$ (3w → 6 cells)

I_6 :- $AC_4 \rightarrow AC_9$ (4w → 8 cells)

I_7 :- $ACA \rightarrow AD_1$ (2w → 4 cells)

I_8 :- $AD_2 \rightarrow AD_5$; $AD_6 \rightarrow AD_7$; AD_8

valid PC



FC

EC

IF	ID	OF		PD	WB	
		S_1	S_2		D	
1MR	3MR	2MR			1RR	I_1 – 4
1MR	2MR	1MR			1RR	I_2 – 3
1MR	3MR	1RR	2MR	1ALU	1RR	I_3 – 4
1MR	1MR	1RR	1RR	1ALU	1RR	I_4 – 2
1MR	2MR	1RR	—	—	1RR 1ALU 1MR	I_5 – 3
1MR	3MR	1RR 1MR	1RR	1ALU	1RR 1MR	I_6 – 4
1MR	1MR	1RR	—	—	1MR	I_7 – 2
1MR						I_8 – 1

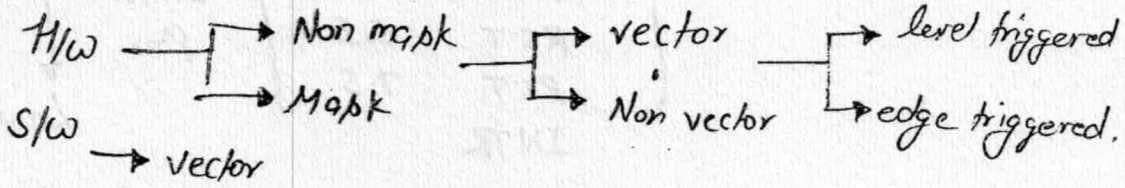
Operand Fetch * + WB (Access) :- 9MR + 1ALU

= 36+2 = 38 cycles

Total time = 32MR + 4 ALU

= 128 + 8 = 136 cycles.

Types of Interrupts :-



* "Hardware Interrupt" :-

* It is a signal, which is generated by the Hardware Component. In the Computer design, Hardware Components are interfaced to a CPU using internal interface and external interface so Hardware Interrupts may be a external interrupt or internal interrupt.

* External Intesuppt is a signal which is generated by the External Hardware.
  Eg: Power supply, Basic I/O C keyboard, Printer, mouse etc).

* Internal Interrupt is a signal which is generated by a internal component of a CPU.
  eg:- Temperature sensor, Timer, clock, critical sensor, invalid opcode, Divide by zero, Stack overflow.

* In the CPU design, Hardware Pins are reserved to hold the Hardware Interrupts.
  Ex:- In 8085, {RST 4.5, RST 5.5, RST 6.5, RST 7.5, INTR} Interrupt Pins.

In 8085 ;

RST	4.5	INTR Pins
RST	5.5	
RST	6.5	
RST	7.5	
INTR		

High → Priority → Low

Software Interrupt :-

* It is an Instruction used to execute the Pre-defined I/O functions in the Processor area.

Ex :- System calls.

Maskable Interrupt :-

This Interrupt is Low priority Interrupt so Low priority Hardware is connected to it. Therefore CPU may or may not read the status of a maskable interrupts to service.

Non - maskable Interrupts :-

It is a High Priority Interrupt, so High Priority Hardware is connected to it therefore CPU compulsory read the status of this interrupt to service.

vectored Interrupt :-

This Interrupt Contain vector, so vector Address is calculated based on Interrupt vector.

eg: In 8085; ~~RST 8085~~

RST 4.5

↓

Vector

Vector Addr. = 4.5 * word size = 4.5 * 8

= $(36)_{10}$

= $(24)_H$

= $(0024)_H$

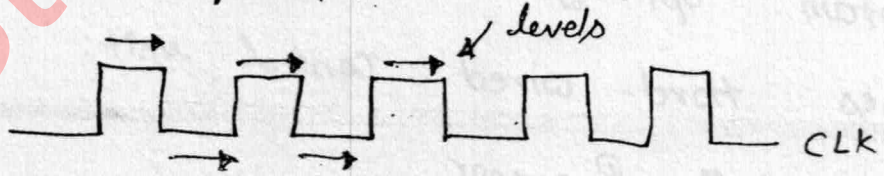
Non-vectored Interrupt :-

This Interrupt doesn't contain vector, so CPU enables the Acknowledgement Signal and waiting until the Interrupt source supplies the vector Address.

eg:- INTR pair with $\overline{INTA}$.

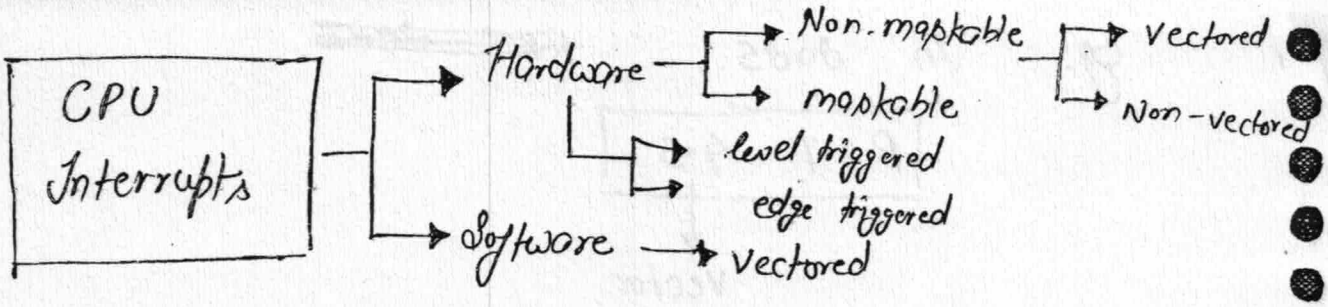
Level triggered Interrupt :-

It is enabled based on the levels of a clock pulse.



Edge Triggered Interrupt :-

It is enabled based on the raising edge or falling edge transition of a clock pulse.



RISC (vs) CISC :-

Characteristic of RISC :-

(Reduced instruction set computer)

* It supports more Registers.
* It supports less Addressing modes.
* It supports fixed length instructions.
* It supports successful pipeline.
* CPI (cycle per instn) = 1
* It is a Supes-computer.
* It is used in the Real-time Applications.
* It is a High expansive processor.
* It contain optimized smaller instruction set.
* It uses Hard-wired control unit.
* eg: Motorolla Processor
  Power PC Processor
  ARM Processor.

Characteristics of CISC :-

(complex Instruction set computer)

★ It supports less Registers.
★ It supports more addressing modes.
★ It supports variable length instructions.
★ It supports unsuccessful pipeline.
★ CPI ≠ 1.
★ It is a general purpose computer
★ It is used in the personal applications.
★ It is a less expansive processor.
★ It contain complex instruction set.
★ It uses micro-programmed control unit
★ eg: Pantium processor.

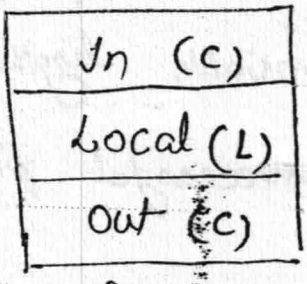
Register Organization in RISC CPU :-

RISC Processor supports more numbers of Registers, categorized into four groups :

1.) Global Register set (G)
2.) Local Register set (L)
3.) In Register set (C)
4.) Out Register set (C)
↳ Common

★ In the RISC CPU, Register window formed by grouping Local, In and out register set. global Registers are also accessible by all the windows.

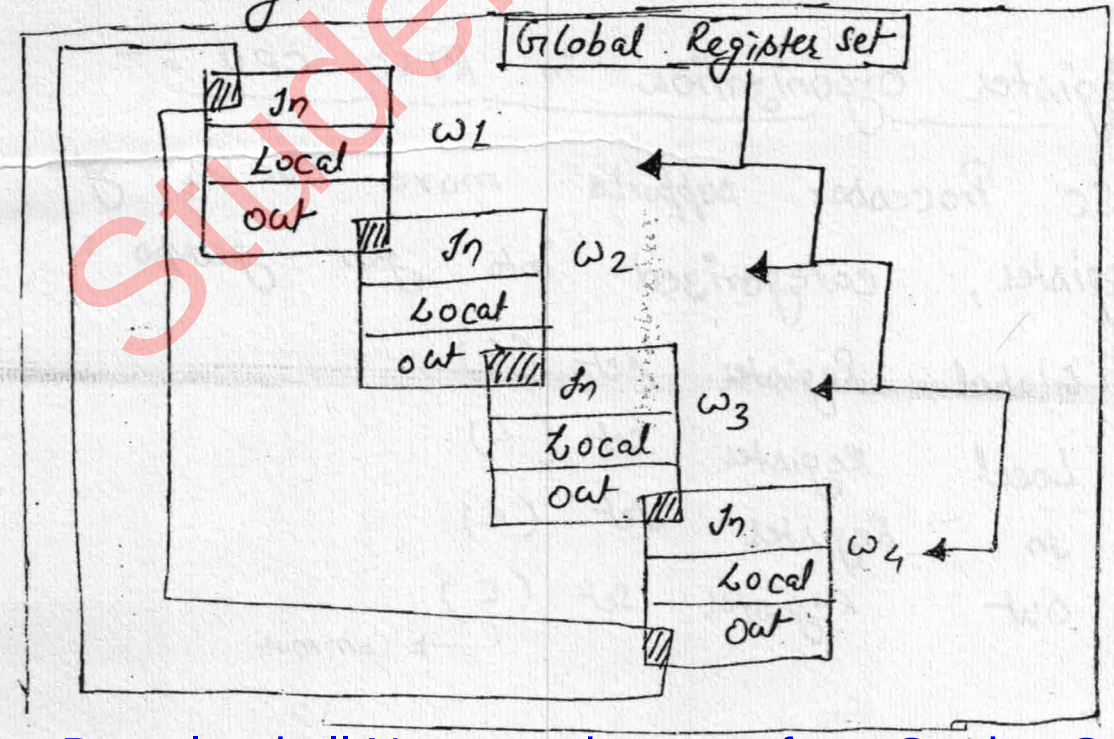
Register window :-



window

window size = L + 2C + G → global Register

★ In the RISC CPU Register windows are organized in a overlapping order i.e. 1 window 'out' Register is used as a 'In' Register in the another window.



(Register file size) or

$$\#\text{ of Register present in the CPU} = \omega(L+C)+G$$

ω = # of windows in the CPU

L = # of Local Registers

G = # of global Registers

C = # IN / # out Register

Que:- In a certain Processor 40 windows are overlapped to manage the Registers. CPU Contain 32 global Registers, 16 local Registers, 8 IN Registers and 8 out Registers. what is the size of the window and Register file in CPU.

$$\text{window size} = \text{In} + \text{Local} + \text{out} + G$$

$$= 8 + 16 + 8 = 32 + 32 = \underline{64}$$

$$\text{file size} = \omega(L+C)+G$$

$$= 40(16+8)+32$$

$$= 24 \times 40 + 32$$

$$= 960 + 32$$

$$= \underline{992}$$

module 1 End