Module - II nd

ij Branch Instruction :- are used to jump the program counter from one memory location to the another. loading the program counter with the new address.

P.C
old memory

P.C
New Memory

Branching

conditional jump
it will check the condition if condition is fullfil then it will jump.

unconditional jump
P.C is loaded with the new address and it will execute the instruction of the memory location.

Branch

Jump

conditional unconditional

Call

unconditional conditional

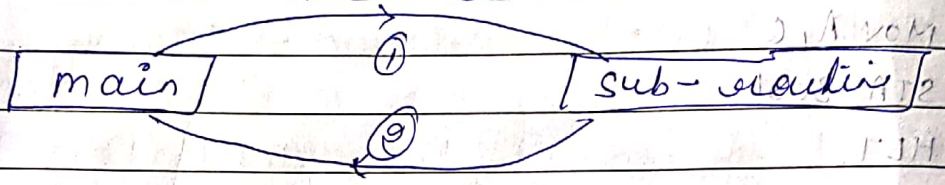
In jump instruction we will jump the new address and will not return the old address.

Main progr → jump → sub-routine

PAGE No
DATE : / 201

→ close loop

In call instruction we jump from the main to the new address after that we will return to the next address.



JUMP :

unconditional	conditional
JMP 16-bit address	

ex:-

JMP 2050 // unconditional jump.

↳ unconditional jump → break the normal sequence.

Conditional Jump :- In conditional jump, first the instruction will condition check karega and if the condition is fulfil then it will jump.

JNZ 16 bit address
↳ Jump on Non-zero
JZ 16-bit address
↳ Jump on Zero
JNC 16-bit address
↳ Jump on non-carry
JC 16-bit address
↳ Jump on carry
JPE 16-bit address
↳ Jump on even parity
JPO 16-bit address.
↳ Jump on odd parity

PAGE No
DATE : / 201

JP → 16-bit address
Jump on plus.

JM → 16-bit address
Jump on Minus.

eg:
```
A = A - B
MVI A ← 08
SUB B
JNZ 5060   Jump to 5060 when the zero flag is reset.
```

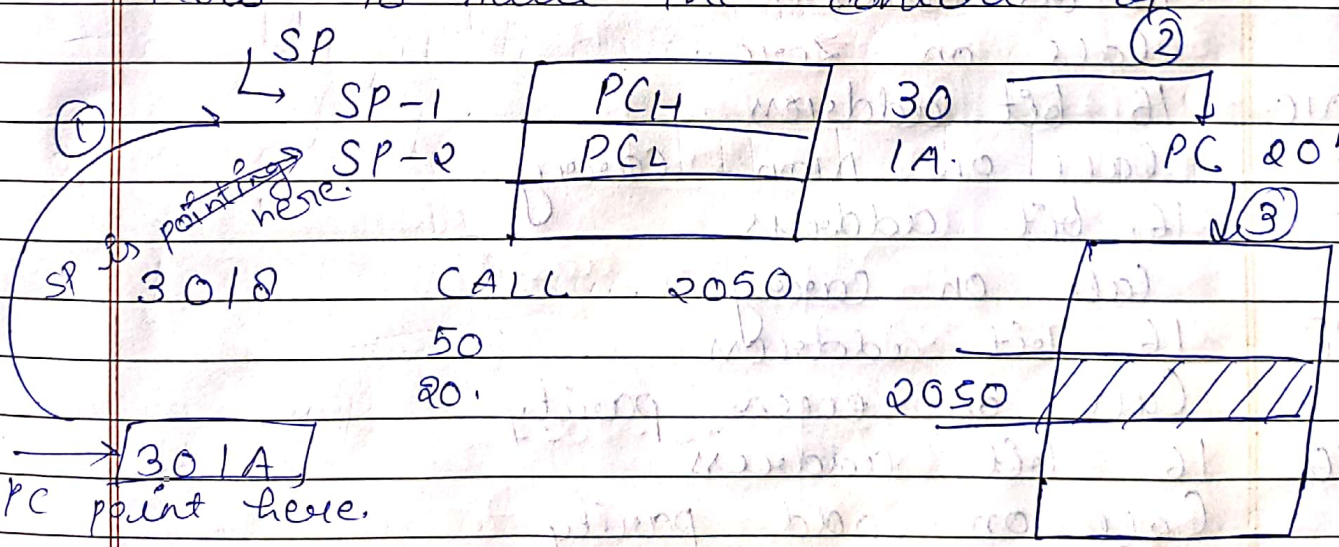
CALL

conditional unconditional.

→ All interrupt is of CALL type.

① Save the content of program counter in stack memory then load the program counter with new address.

How to save the content of PC.



PC point here.

to return into the main-main program we write the return instruction by using RET instruction.

SP → PCL.
SP+1 → PCH
SP → SP+2.

SP — 3050

SP-2	304E	15
SP-1	304F	20
	SP	

SP → 304E

PC — 3050

RET:

SP → PCL (15)
SP+1 → PCH (20)

CNZ 16 bit address
↳ Call on Non-Zero.
CZ 16 bit address
Call on Zero.
CNC 16 bit address
Call on Non-Carry
CC 16 bit address
Call on carry.
CPE 16 bit address
Call on even parity
CPO 16 bit address
Call on odd parity
CP 16 bit address
Call on plus.
CM 16 bit address
Call on minus.

PAGE No
DATE : / 201

Q WAP for a addition of 8 bit number series starting 2101 onwards the location 2100 has the number of hexa-decimal bytes to be added. Store the final result at 2300 & MSB at 2301.

PAGE No
DATE : / 201

Q. WAP to multiply two 8 bit number store in memory location 2000 & 2001 and store the result at memory location 2002.

```
      MVI   A   00H
      LXI   H   2000H        B → 05
      MOV   B   M
      INX   H
add1. ADD   M
      DCR   B
      JNZ   add1
      STA   2002
      HLT
```

```
3000    3E ]  MVI A 00H
3001    00
3002    21 ]
3003    00 ]  LXI H 2000H
3004    20 ]
3005    46 → MOV B M
3006    23 → INX H
→3007   86 → ADD M
3008    05 → DCR B
3009    C2 ]
300A    07 ]  JNZ add1
300B    30 ]
300C    32 ]
300D    02 ]  STA 2002
300E    20 ]
300F    76 → HLT
```

(4) find the highest no. b/w the memory set 2001 - - - 200A and store the result at 200B.

```
      MVI  A, 00H
      LDA  200A
      MOV  B, A
      INX  H
      MVI  B, 09
      MOV  A, M
add2: INX  H
      MOV  C, M
      CMP  C
      JNC  add1
      MOV  A, C
add1: DCR  B
      JNZ  add2
      STA  200B
      HLT
```

(1) Programming for time delay and counter.

Sometime

Delay using Loop.
large delay using Nested Loop.

PAGE No
DATE : / 201

We use conditional jump here.

for jumping

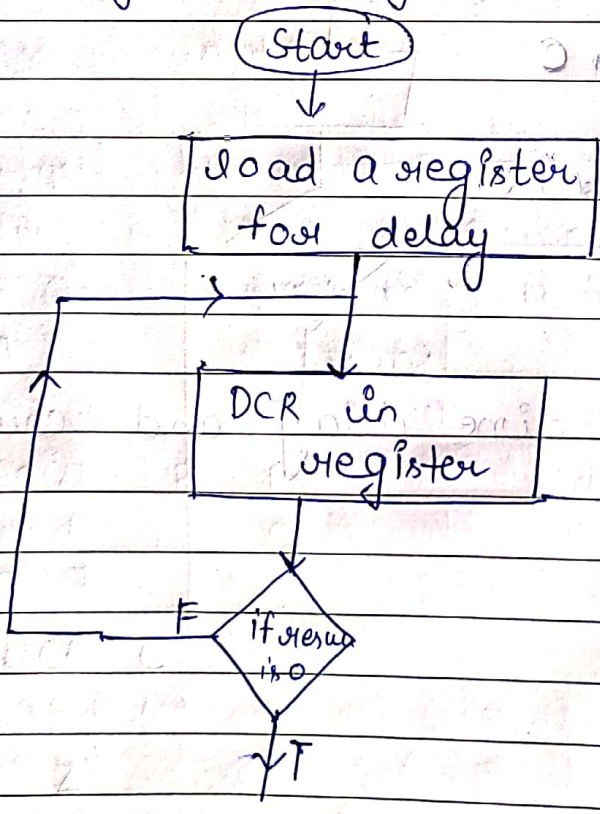
			↓	
2012	JNZ	16-bit	10T / 7T	
13			↳ during last.	
14			not jumping case.	
PC	16 bit	2015		
PC	2015			

Microprocessor operate on 3MHz.

Time taken in 1T state = $\frac{1}{3}$ usec.

for 10T state = $\frac{10}{3}$ usec.

Time delay for single loop



(7T) MVI A ϕ ϕ = 00 .... FF

(4T) loop1: DCR A.

JNZ, loop

No. of T state in delay

$= 1X7 + 4\text{Ø} + (\text{Ø}-1)\times 10 + 1X7$

MVI A,Ø — DCR A — when loop execute — when don't

WAP 1

- WAP for 1 millisecond delay.
- Calculate the maximum delay produced by single loop delay.

Sol^n Total number of T state =

$7 + 4\text{Ø} + 10\text{Ø} - 10 + 7$

$= 14\text{Ø} + 4$

Total delay $= \frac{(14\text{Ø}+4)\cdot 10^{-6}}{3} = 10^{-3}$

$(14\text{Ø}+4) = 3000$

$14\text{Ø} = 2996$

$\text{Ø} = 214$

$\text{Ø} = D6$

16	214	13
	16	
	54	
	48	3
	6	40

a)

MVI A D6

loop1: DCR A

JNZ loop1

b) $\text{Ø} = FF$

$\text{Ø} = 255$

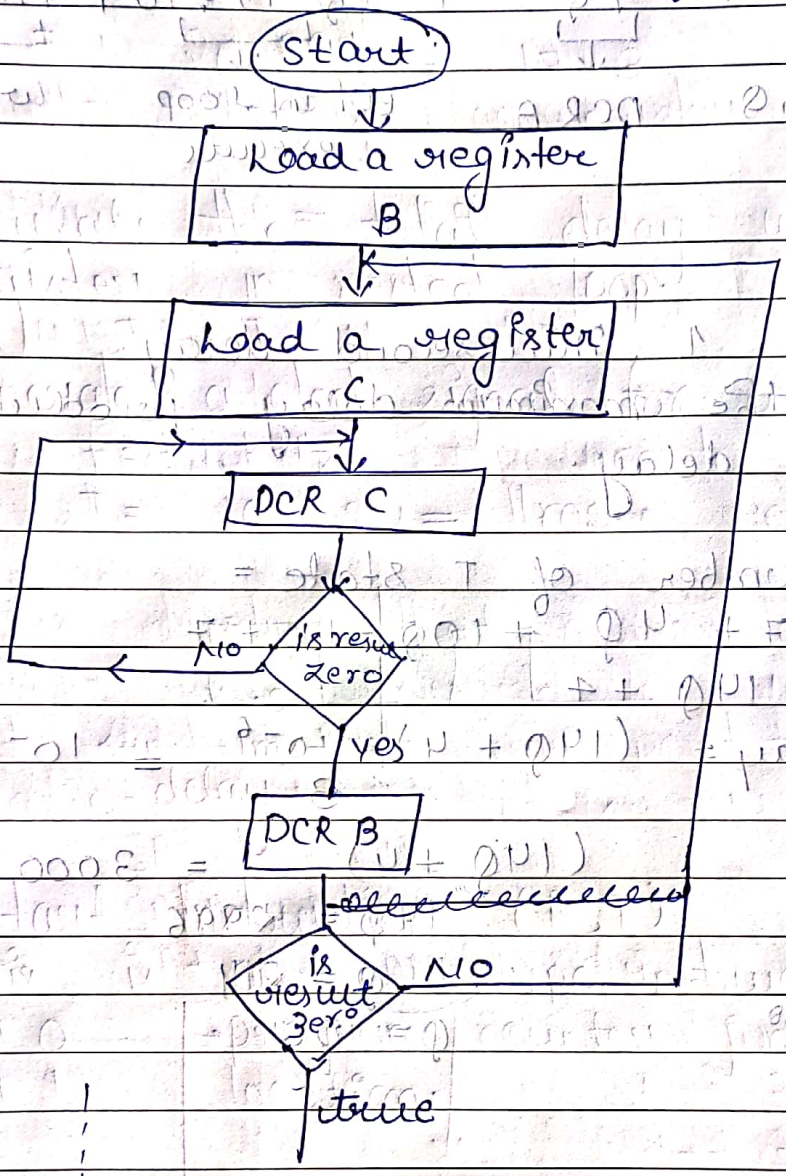
$(14X255+4)\times\frac{1}{3}\times 10^{-6}$

$\frac{3574}{3}\times 10^{-6}$

$\Rightarrow 1.19$ ms

PAGE No
DATE : / 201

Delay using the Nesting of the loop




```
        MVI B= P
loop2   MVI C = Q
loop1   DCR C
        JNZ loop1
        DCR B
        JNZ loop2
```

(d1)P is representing data
P → (d2)

Calculate the T state required to execute this delay program.

~~1 × 7 + 7 + 4p + (p-1)10 + 1 × 7 + (p-1)10 + 4p + 1 × 7 + (p-1)~~

1 × 7 + p[1 × 7 + q × 4 + (q-1)10 + 1 × 7] + p × 4 + (p-1)10 + 7.

Calculate the maximum delay provided by the nested loop.

p = 255 q = 255 (FF)

7 + 255 [1 × 7 + 255 × 4 + 2540 + 7] + 255 × 4 + 2540 + 7.

$= \frac{914.944}{3}$ ms.

= 0.3049 m

Write a program to calculate the maximum delay in single loop program using 16 bit register.

LXI H FFFF // 3byte ≃ 10 T

loop1: DCX H // 1byte ≃ 4T

JNZ loop1.

Total T state = 1 × 10 + (4369 × 4 + 4368 × 10 + 1 × 7)

JNZ

= 10 + 17476 + 43680 + 7

= 61173

$= 20391 \times \frac{10^{-6}}{3}$ s

= 20.391 ms

43692
× 4 3
17476
43680
61156
17
73

16 bit delay to the 8 bit delay

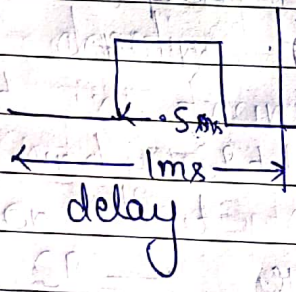
$= \frac{20.391}{1.19} = 17.3$

qmp Calculate the total delay using the 16 bit register in nested loop?

⇒ Generate a clock signal for a frequency 1kHz

$f = 1\,kHz$

$T = 10^{-3}\,s = 1\,ms$



** Restart Instruction (RST)

this is an (CALL) type instruction → program counter jump on a prefix location.

↓

bec3 we are loading the content PC of PC into Stack pointer.

RST 0		
RST 1		
⋮		
RST 7		

16 byte inst

RST 0	8X 0 = 0	0000
RST 1	0X 1 = 8	0008
RST 2	0X 2 = 16	0010
⋮		
RST 7	0X 7 = 56	0038

content of SP and SP-1 and SP-2.

2015 RST 5.

PC → 2016

content of PC store in SP-1 and SP-2 and RST 5 will load into the PC.

SP	
SP-1	20 → higher bit of P.C
SP-2	16 → lower bit

PC will load with the instr.

* PUSH and POP Instruction → Work on (FILO)

PUSH (Rp) → register pair

BC ≅ B

DE ≅ D

HL ≅ H

⇒ This instruction is used to save the content of register pair into the stack memory.

PSW (processor status word)

A (8 bit) F (8 bit)

→ 16 bit register

		B	E
→3013	PUSH B	31	03

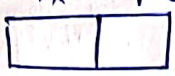
Stack Memory

SP → 3FFF	
3FFE	31
3FFD	03

Now SP ≅ 3FFD

SP-1	$(RP)_H$ (31)
SP-2	$(RP)_L$ (03)

register pair ki lower bit.



PUSH PSW

Before execution → SP ⟶ 3FFF
SP-1 ⟶ 3FEF [A]
SP-2 ⟶ 3FED flag register
Now SP = 3FED

(work on last in first out)

POP INSTRUCTION

POP rp

This instruction is used to copied the content of stack memory into the register pair.

SP is denoting that memory location that content is transferred to register pair lower bit and SP+1 will point to the memory location that content is transferred to higher register pair. SP ⟶ rpL

SP+1 ⟶ rpH

(SP+2) ≃ (SP) → now SP will point to SP+2.

3005 POP D

	SP → 3FFF	d1 (08)
	SP+1 → 4000	d2 (06)
D ≃	D E	SP → 4001
	06 08	

Q- Exchange the content of BC register with the DE register using push & pop instruction.

B	C	D	E
12	34	56	78

65535

PAGE No
DATE : / 201

Program

```
PUSH B
PUSH D
POP  B
POP  D
```

SP → 3013

	3013	
SP-1	3012	12
SP-2	3011	34
SP-3	3010	56
SP-4	300F	78

SP → SP-5

POP B	POP D
B | C	D | E
56 | 78	12 | 34

And finally SP → SP

** i) have a string of 10 data stored at memory location 2001 and 200A.
ii) arrange this data into ascending order.
iii) arrange this data into descending order.

```
      MVI  D 09
loop3 LXI  H 2001
      MVI  B 09
      MOV  A, B
loop2 INX  H
      CMP  M
      JNC  loop1
      MOV  C, A
      MOV  A, M
      MOV  M, C
      DCX  H
      MOV  M, A
      INX  H
loop1 DCR  B
      JNZ  loop2
      DCR  D
      JNZ  loop3
```

2001	32
2002	49
2003	39
2004	A1
2005	0F
2006	F2
2007	A3
2008	B3
2009	F5
200A	BA

0F
32
39
49
A1

PAGE No.
DATE: / 201

Some Instruction related to Interrupt

Overview of all interrupt and instruction related to it

Interrupt:- External agent request to perform some special task.

Interrupt

Software Interrupt

Which are writting into the program

RST 0 to RST 7 → Software interrupt.

Hardware Interrupt

pin (6-10) through external device

TRAP
RST 7.5
RST 6.5
RST 5.5

(Highest) → (lowest)

priority decrease

INTR

Hardware Interrupt

Vector Interrupt

In this program counter jump on prefix address

TRAP RST7.5 RST6.5 RST5.5

Ad (0024) (003C) (0034) (002C)

Non-vectored Interrupt

In this we give address of memory at which the program has to jump.

INTR

5.5 X 8 = 44 binary ≃ 2C. H.

0 1 00

6.5 X 8 =

7.5 X 8 = 60 ≃ 003C

PAGE No
DATE : / 201

Vectored Interrupt

Non-maskable Interrupt	Maskable Interrupt
→ means this type of interrupt cannot be mask (stop).	→ We can mask means it will not run during the program execution
→ TRAP (means it will always execute no control at it)	→ RST 6.5 RST 5.5 RST 7.5

Instruction related to Interrupt

i) EI → Enable Interrupt
   → 1 byte instruction
   → to recognise interrupt in a program
If EI enable then program counter will jump on the specified address.
→ to provide the service to the interrupt.

ii) DI → Disable Interrupt
   → 1 byte instruction
   → to deactive the interrupt in a program.
   → to change the priority

3) SIM → (set interrupt Mask) is used to mask / unmask a particular interrupt

		X		MSE	RST 7.5	RST 6.5	RST 5.5
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

1 → mask
0 → non-mask

We use accumulator as a register.

PAGE No
DATE : / 201

↳ permission to mask.

ME ↦ Mask Enable
↳ 1 ⟶ Mask enable
↳ only then we can counter the value of D_0, D_1, D_2
↳ 0 ⟶ mask disable
↳ we are not bother about the value of D_0, D_1, D_2

D_4 → restart 7.5 or restart 7.5 flip-flop
↳ is used to retrigged the flip flop of RST 7.5

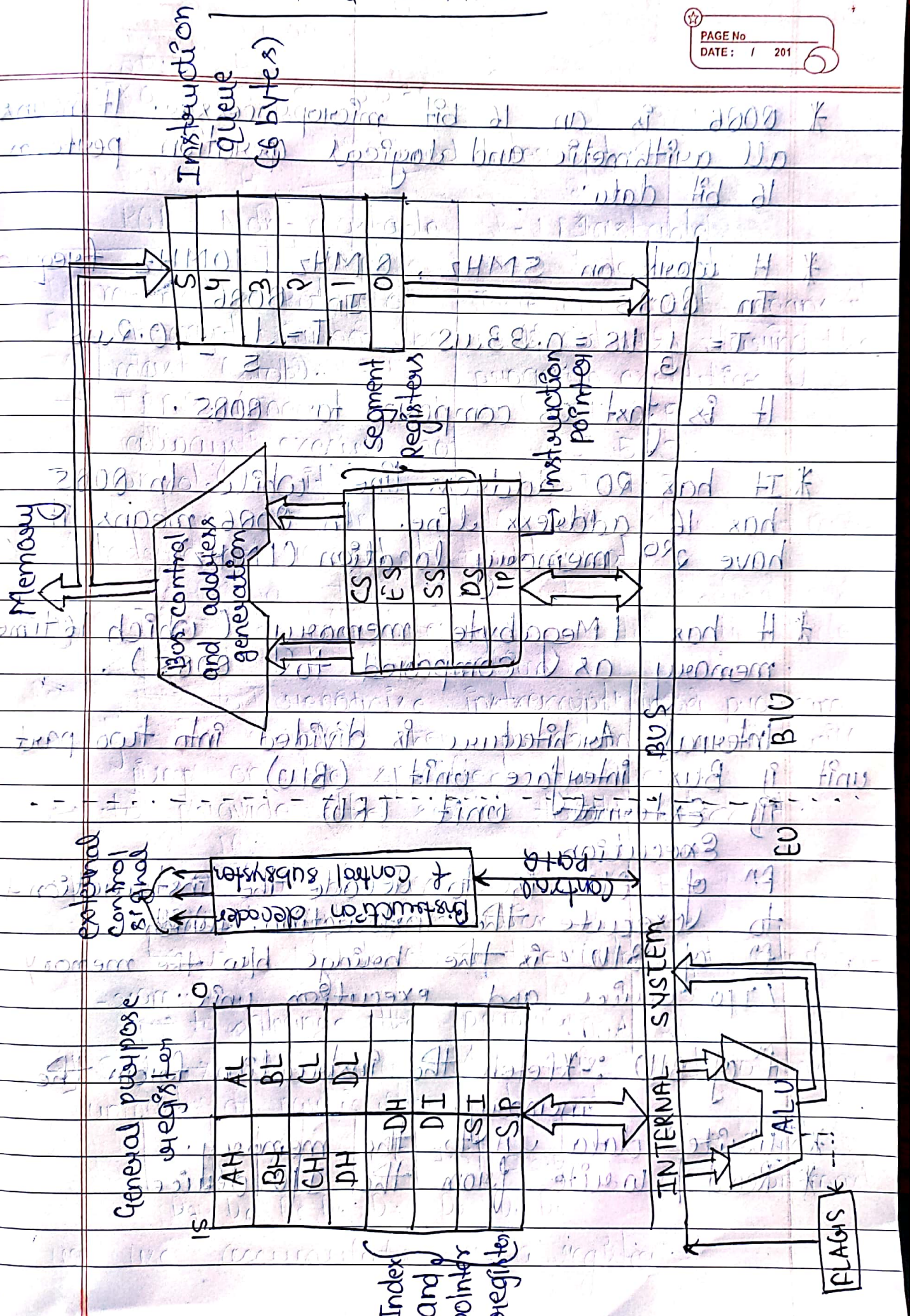
D_6 SDE :- Serial data enable } this bit are used to serial communication

D_7 SOD :- Serial Output data }

NOP :- No operation
↳ only for providing delay
↳ 1 byte
↳ 4T state provide

** Write a signal flow graph for the digital clock. (draw)
(second, minute, hour ⟶ to display)
Signal flow graph ≃ flow chart

Jai Maa Saraswati



PAGE No
DATE : / 201

* 8086 is an 16 bit microprocessor. It means all arithmetic and logical operation perform 16 bit data.

* It work on 5MHz, 8MHz, 10MHz frequency
In 8085 — In 8086
$T = \frac{1}{3}\ \mu s = 0.33\ \mu s$ — $T = \frac{1}{5} = 0.2\ \mu s$
It is fast as compared to 8085.

* It has 20 address line while In 8085 has 16 address line. In 8086 means it have 2^{20} memory location (1MHz)

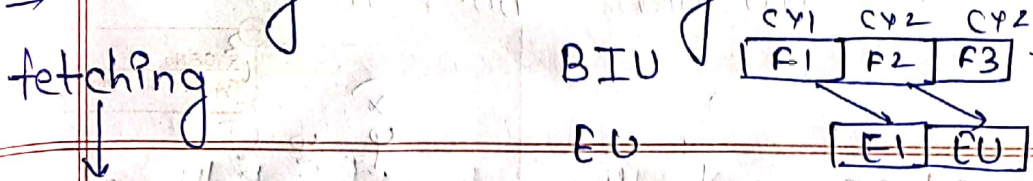
* It has 1 Megabyte memory (which 16 time memory as compared to 8085).

Internal Architecture is divided into two part unit
1) Bus interface unit (BIU)
2) Execution unit (EU)

f^n of EU is to decode the instruction & to execute the instruction.
f^n of BIU is the bridge b/w the memory / I/O device and execution unit.

f^n of BIU :- *fetch the instruction from the memory.
* Write data in to the memory.
* Read / write from the I/O device.

⇒ fetching and executing unit are different.



PAGE No
DATE : / 201

→ use the concept of pipelinning in 8086. but in 8085 there is no concept of pipelining.
→ execution and bus interface unit are working parallely.

In 8085

Execution unit :- It decode the instruction receive from the bus interface unit (BIU)
** generate the control signal required to execute the instruction.
** Execute the instruction.

Execution unit has following functional group:-
i) General purpose register
ii) Base & Index register
iii) ALU
flag register
Instruction decoder & Control subsystem.

i) General Purpose Register :- 8086 has 8 bit 8 general purpose register. AH, AL, BH, BL, CH, CL, DH, DL and can be used as 16 bit register. as

AX = AH + AL
BX = BH + BL
CX = CH + CL
DX = DH + DL

16 bit register

↳ Pointer.

ii) Base & Index Register :- In 8086 has two 16 bit base register one is SP (Stack Pointer) ij BP and ij SP & two Index DI (Data Index) register ij data Index & SI (Stack Index) ij stack Index register

Base & Index

The main purpose of these register is to store the offset address of the memory.

→ ALU :- fn same as 8085
↳ to perform an arithmetic & logical operation.
↳ how u can control the status.

* Flag Register :- 16 bit, 9 flag
controlling conditional.

It is a 16 bit register and it has total 9 flags divided into two categories i) Conditional flag ii) Controlled flag

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	OF	DF	IF	TF	S	Z	X	AC	X	P	X	CY

flag

Conditional
- Carry flag (0)
- Parity flag (2)
- Auxillary carry (4)
- Zero (6)
- Sign (7)
- Overflow (11)

Controlled
- TRAP (8)
- Interrupt (9)
- Directional flag (10)

(software)

Conditional flag :- set or reset on the basis of operation.

Controlled flag :- It control the operation (Hardware)

IP ≃ PC
↳ store the address of next

PAGE No
DATE : / 201

of the program.

Carry flag :- When there generate a carry from the MSB of the data. the carry flag will be set.
D_7 → 8 bit
D_{15} → 15 bit.

Parity flag :- If result has even no. of 1's then parity flag will be set.

Auxillary flag :- In case of 16 bit data, when there generate a carry from lower byte to upper byte. and
In case of 8 bit data, when there generate a carry from lower nibble to upper nibble the A.F will be set.

Zero flag :- If the result is zero then ZF will be set.

Sign flag :- When MSB is 1 then sign flag is 1 otherwise sign flag is 0.
↳ (0-255).

Overflow - When result is greater than 16 bit, then overflow flag is set. ↓ in case of data max^m value means all (11111) of unsigned.

Controlled flag ↳ if there generate a carry then carry flag will be set.

Directional flag :- represent the direction of the program.
If it is 0, (i++) the bottom to top.

If it is 1, (i--) then top to bottom.
↳ decreasing

* Interrupt flag :- (non-maskable interrupt)
If it is 1, current instruction execute then it will recognize the flag and execute the interrupt.
If it is 0, interrupt will not recognize.
(↳ no service provided to the interrupt.)
This flag is used to recognize the non-maskable interrupt. If this flag value is 1 then processor execute the current instruction & check the status of non-maskable interrupt.

* TRAP :- (Non-maskable interrupt)
If this flag is then after the execution of current instruction it will check the status of NMI (Non-maskable interrupt). If this flag is 0 then after the execution of instruction it will not check the status of NMI

Instruction decoder and control subsystem :-
* Receive the instruction from bus interface unit.
* Decode the instruction.
* Generate the control signal.

Bus Interface unit :- (fetching the data) and
↳ working parallely with EU (CPU)
→ It work as the interface b/w execution unit and the memory and I/O device. Main purpose of BIU is to fetch the data.

and provide it to the execution unit.
Main functional block of BIU are
i) Instruction Pointer
ii) Segment Register
iii) Bus control and address generation.
iv) Instruction queue.

Instruction Pointer :- It is an 16-bit register. Its work is same as PC in 8085.
→ It represent the address of next instruction to be fetched.

Segment Register :- 8086 has 4, (16 bit) segment register.
CS (Code Section)
DS (Data Section) → segment.
ES (Extra Segment)
SS (Stack segment)

and the main of these register is represent the base address (physical) of the memory.

Bus control and Address generation unit :-
MA = Base Address + offset Address
(Memory Address)
BA is the combination of CS, SS, ES, DS
OA is Base + Index register
BA address remain in segment register.

Base	Offset
Code	I.P
Data	DI or SI
Extra	DI or SI
Stack	SP or BP

→ Segment register mei MSB ki 4 bit left hand side mei shift ho jayengi and add 4'0 at the end of the address.

DS → 2000 SI :- 3240
base address Offset Address

New DS :- 20000

Memory generation :- 20000
3240
23240

Range of DS :- 20000 } DS
2FFFF }

Range of SI :- 3240 → 0000 } SI.
FFFF }

* Instruction queue :- It is a 6 byte register When BIU is free then it will store the instruction in the queue and work on the principle of FIFO.

Advan :- ↑ the speed of up.