

Max. Bus. size: 6 bytes

PAGE NO.	
DATE	

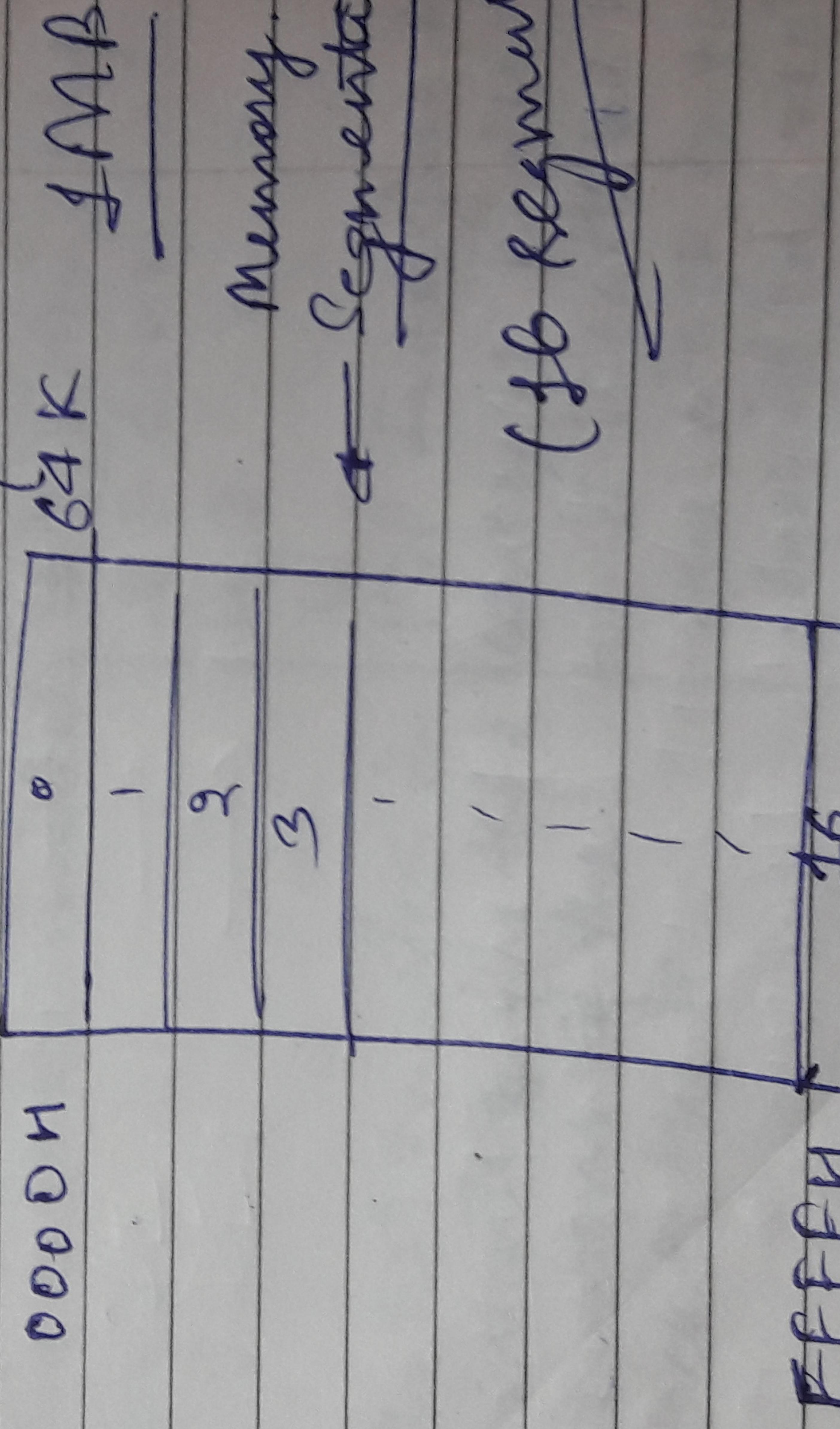
03/04/2018

8086 Architecture.

(Diagram from Assignment)
Disadvantage of Instruction Queue

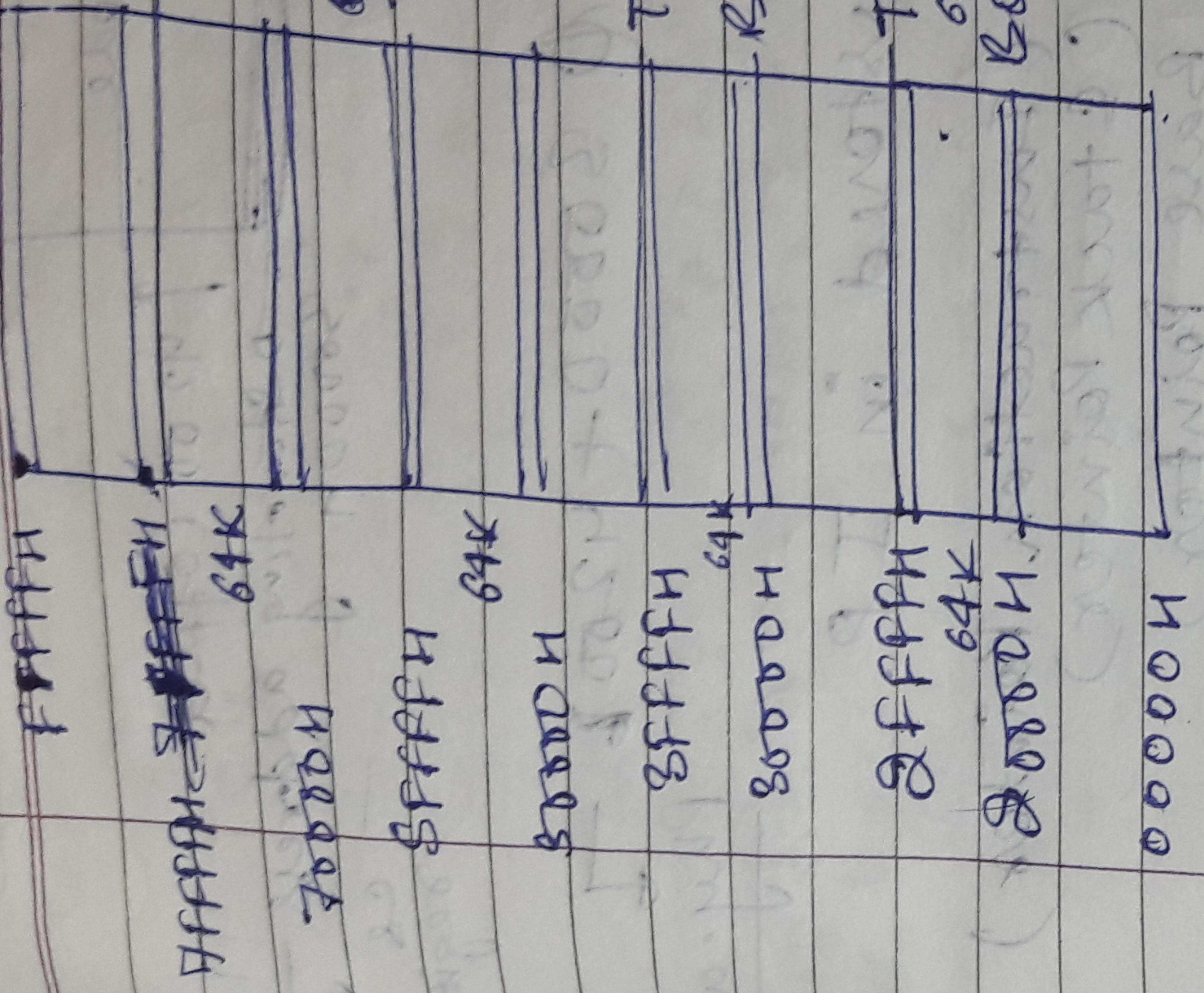
In case of a JUMP or CALL Instruction, all the existing ① bytes in the queue have to be removed & new ones to be entered.

Segment
Size



Code segment (cs): Stores the codes of all the instructions

Memory Segmentation

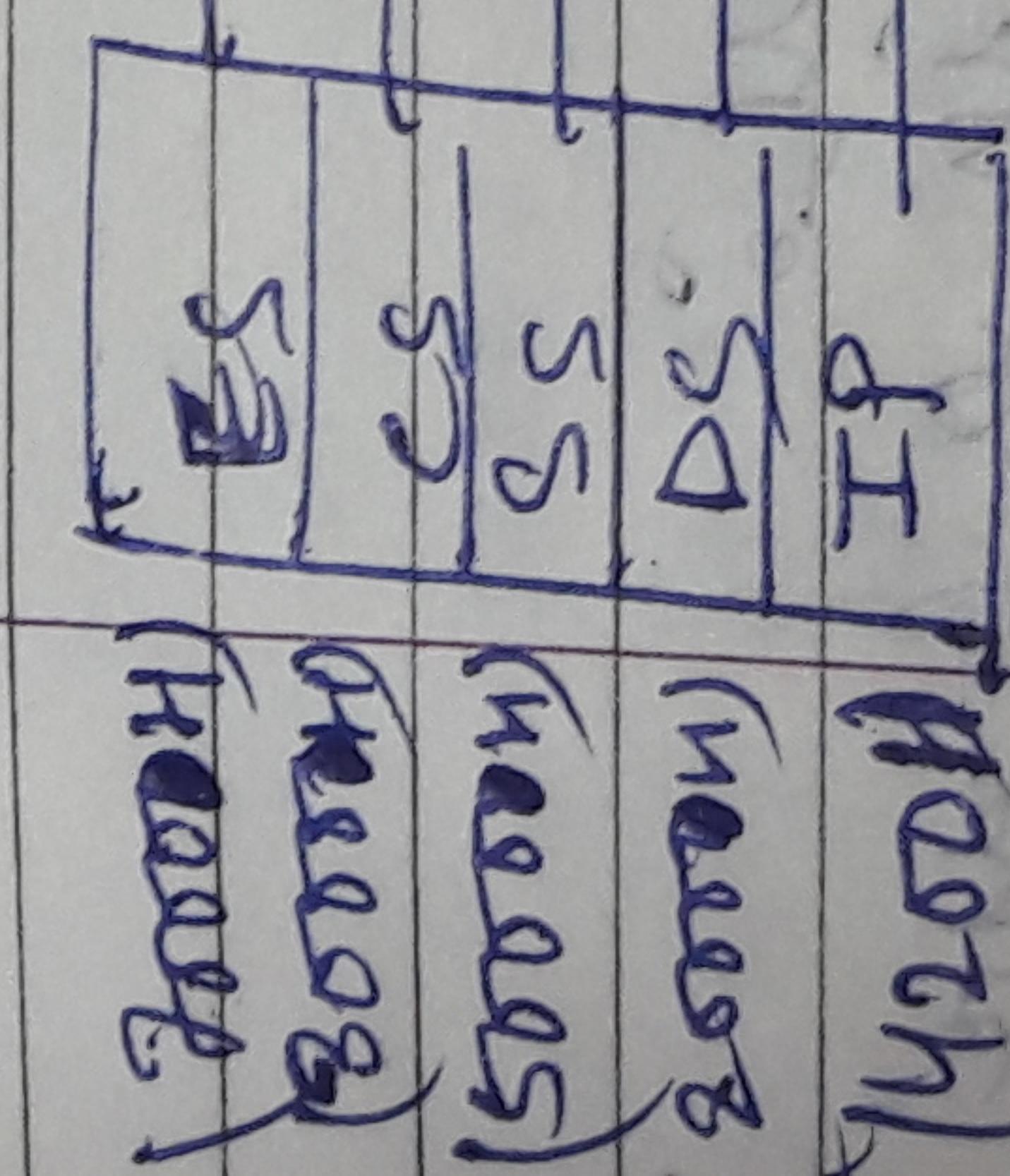


Top of CS → (70000H)

Top of SS → (70000H)

Top of DS → (70000H)

Top of ES → (70000H)



→ hold starting address of the segment.

→ add offset to the CS register.

CS Register stores the starting address of the CS. → where own space is created.

To reach to a particular location in the CS, an offset is added to the CS address to get which is known as the physical address.

→ where own space is created.

Code Register {
 Opcode } 14200 (offset)
 Starting address 7
 80000000000000000000000000000000
 ce register

0 30000 + 4200 → Phy. add.

Offset is stored in IP .

{ IP (Instruction Pointer)
 SP (Stack Pointer)
 BP (Base Pointer)
 SI
 DI }

- Significance of Queue :
 - Instructions are stored in a 6-byte queue in the FIFO form .
 - The Queue will always hold the byte of the next instruction to be executed by the EU .
 - The process of fetching the next instruction when the present instruction is being executed is called " pipelining ."
 - B10 fills the queue until it is full .
 - It starts refilling the queue when at least 2 locations off the queue are vacant .

→ The queue is 64 bytes long which is less than max. instruction size in the ~~8086~~²²⁶ bytes. So it is possible to prefetch the next instruction.

Advantages:

This is faster than sending an address to the memory & waiting for the next instruction byte to come. Thus speeding up the process.

Disadvantages:

If a jump call appears in the main program, then all the existing instruction bytes in the queue are flushed. Then the queue is loaded with new instruction bytes corresponding to the location of JMP/call point.

Advantages of Segmentation:

- It provides a powerful memory management mechanism.
- It allows 2 processes to easily share the data.
- It is possible to reuse the memory size of the code, data, stack or extra segment beyond 84 bytes by allocating more from 1 segment for each area.
- It helps to make it possible to write programs which are position independent.
- It provides a way to easily implement object oriented programming.

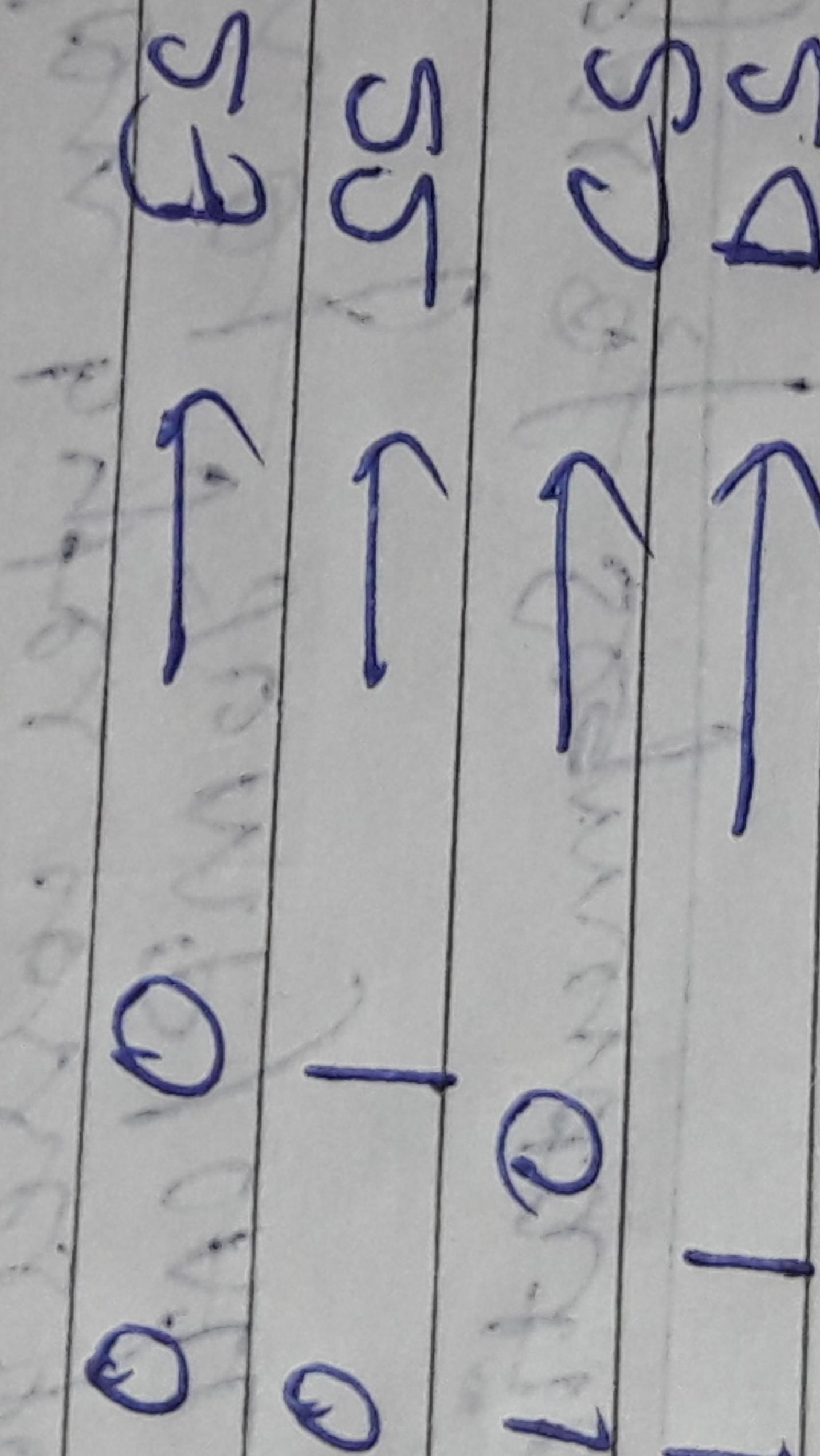
→ It allows the programmer to position their program in memory into modules that operate independent of each other.

* Odd memory Bank (Do)

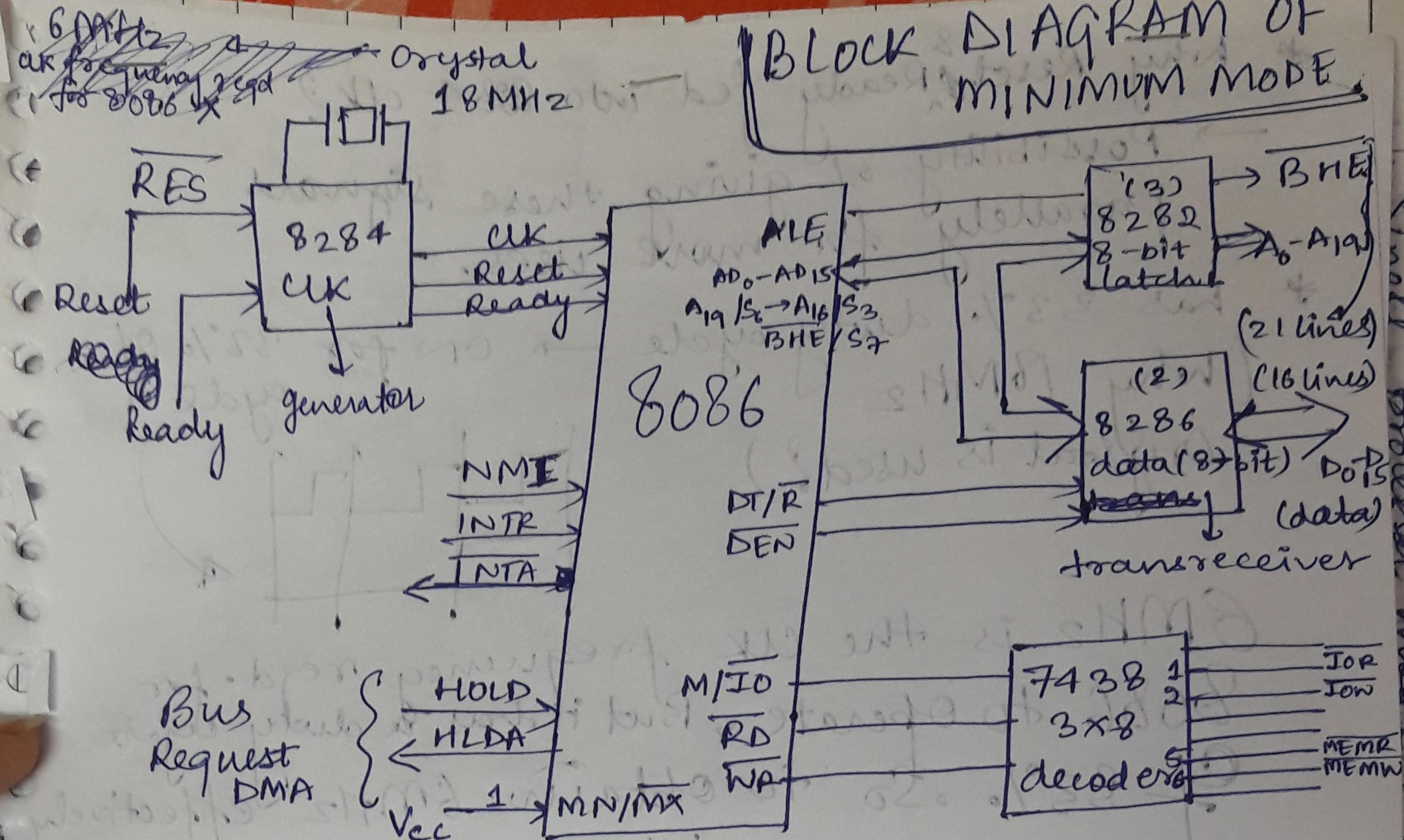
Single processor mode: Min Mode
Multiple $n \times n$ Max Mode

Data address & status lines are multiplexed (42 lines reduced to 21 lines).

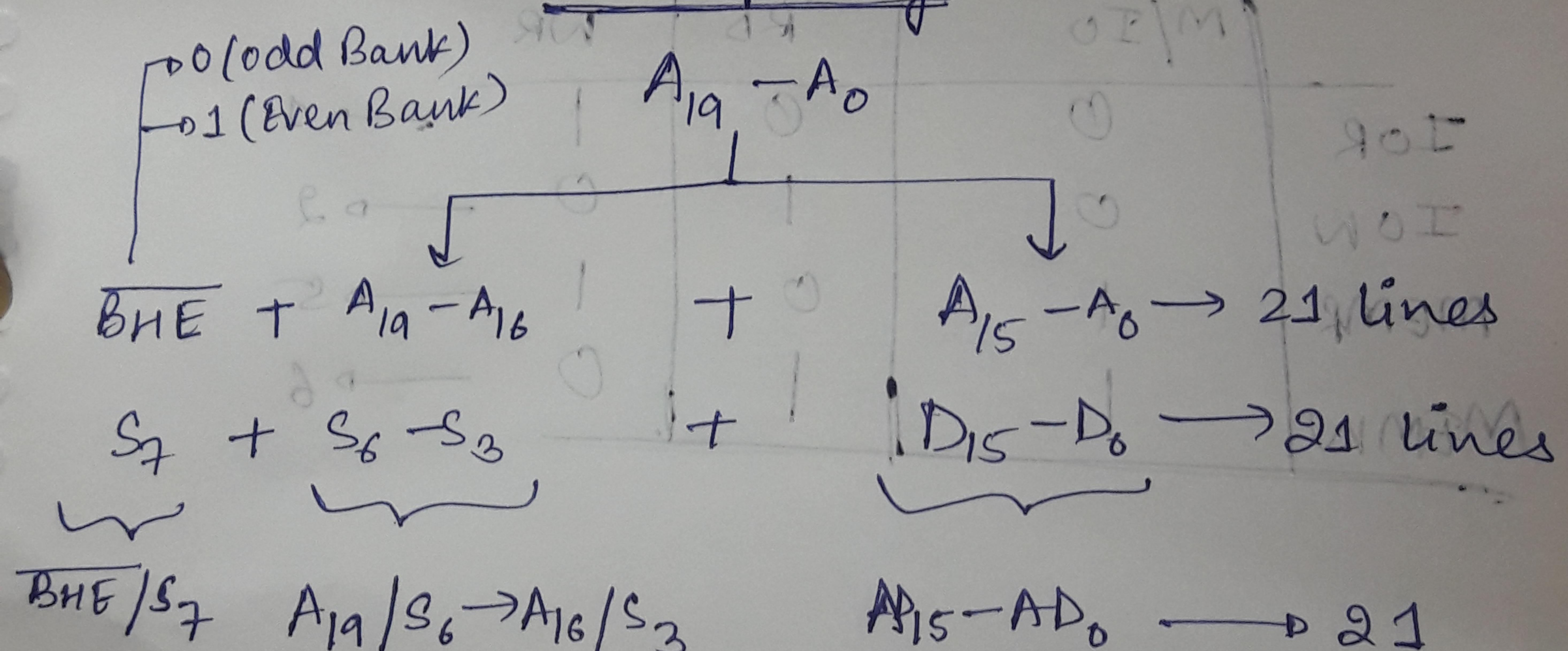
S₂ S₄



BLOCK DIAGRAM OF MINIMUM MODE



Multiplexing

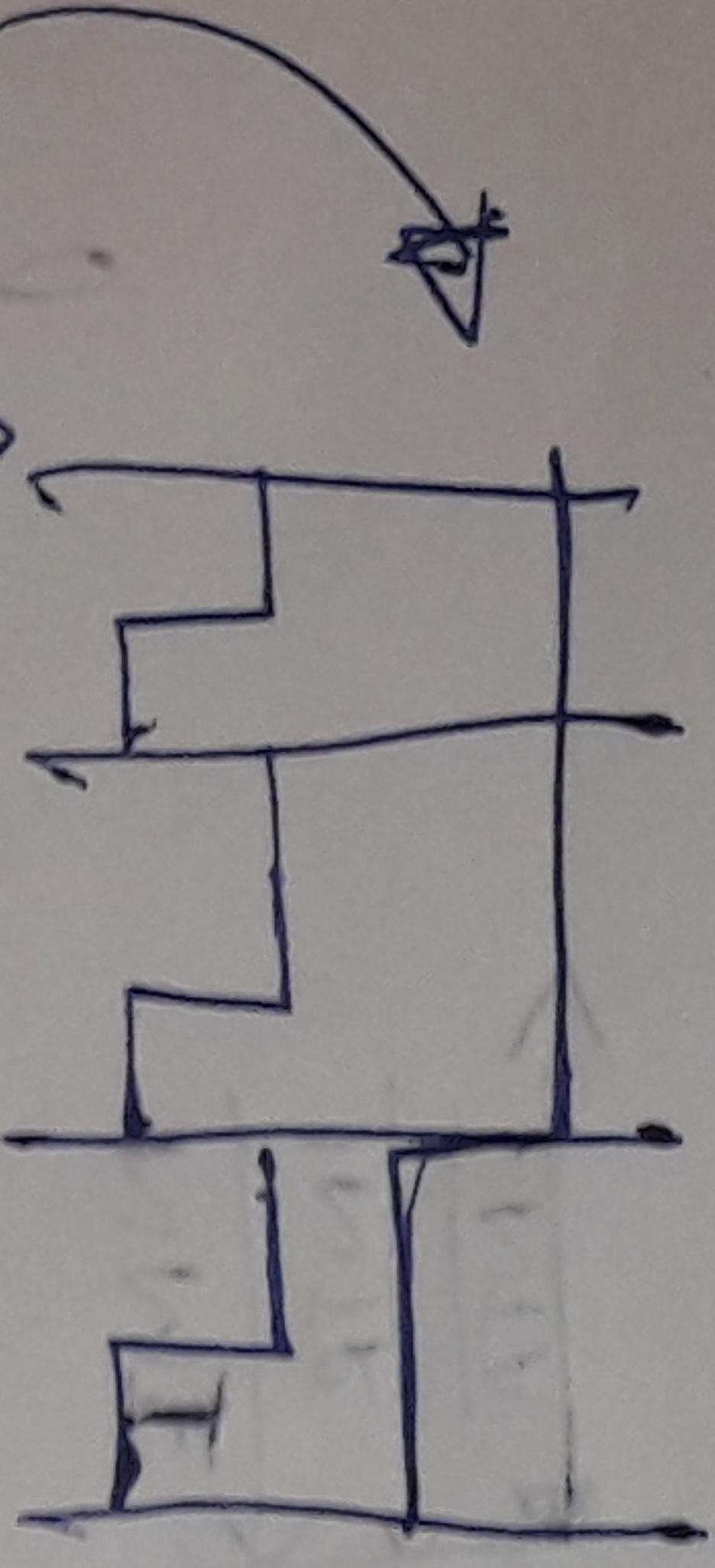


$S_5 \rightarrow 0 \rightarrow IF = 0/1 \rightarrow$ Interrupt flag

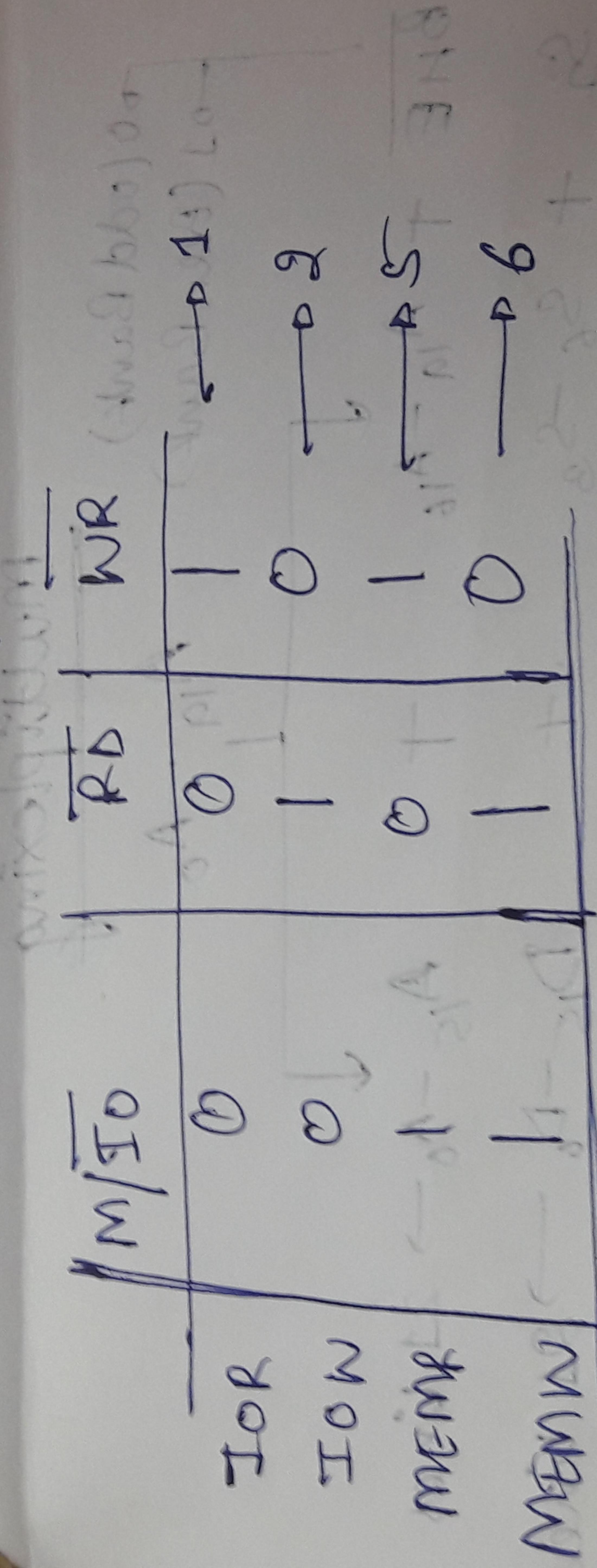
$S_6 \rightarrow 0 \rightarrow 8086 BM$ (Bus Master)

1 → Other BM

- * Why Reset & Ready fed into SSB + CLK?
 - Possibility of giving these signals parallelly for more users.
- * has $\geq 3\%$ duty cycle → on for 33% of cycle.
 - (why 18 MHz crystal is used?)



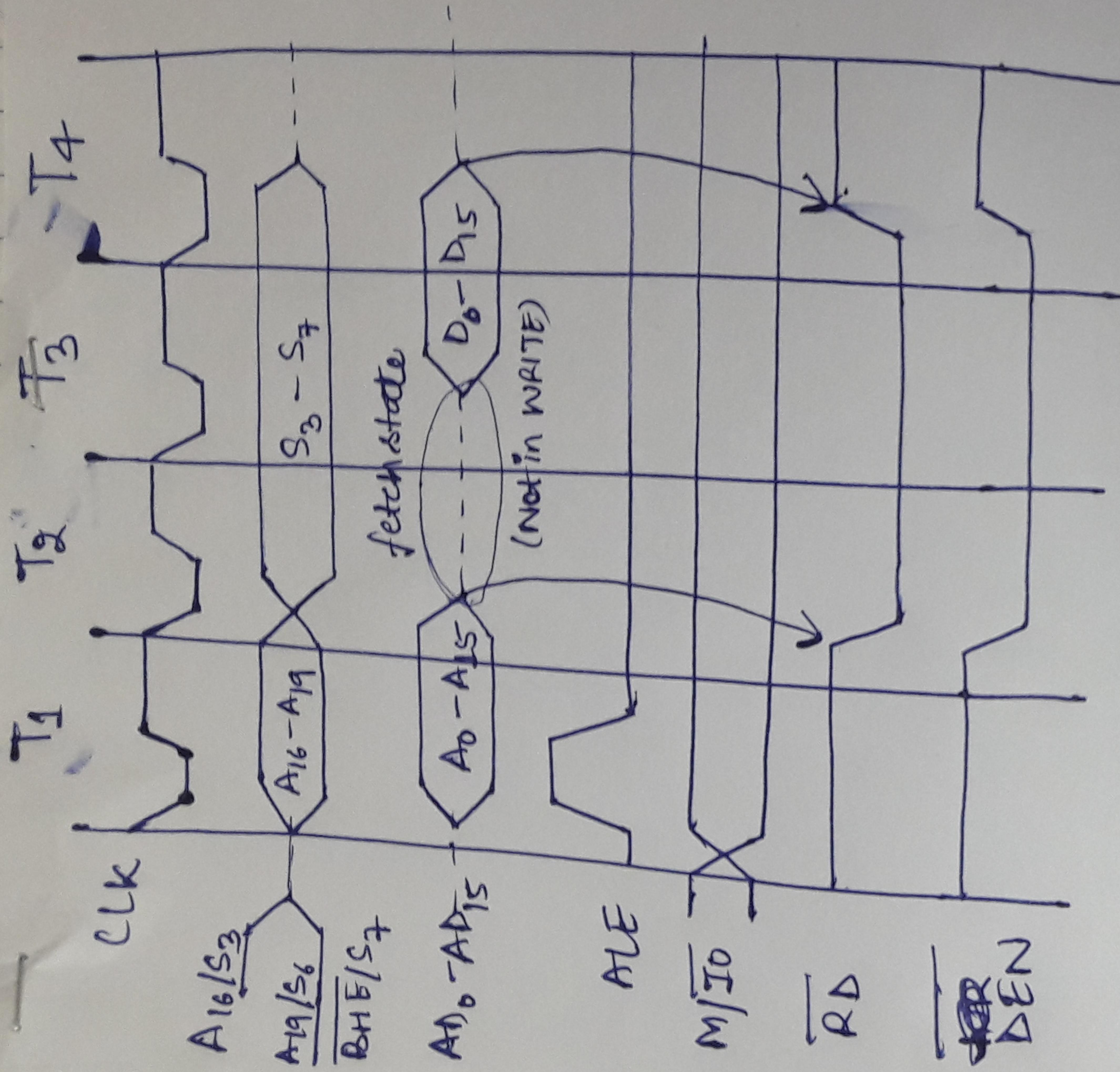
6MHz is the CLK frequency reqd. for 8086 to operate. But it has a duty cycle of 33%. To obtain 6MHz effective we use a crystal of 18MHz.



Logic 4017 - 311A
(4017B) MG1808

Logic 4017 - 311A
S1A, S2A, P1A, P2A
T1, T2, T3, T4

Main Mode
Timing
Diagram
(READ)



22-7

07
20

76

76

10/01/2018

classmate

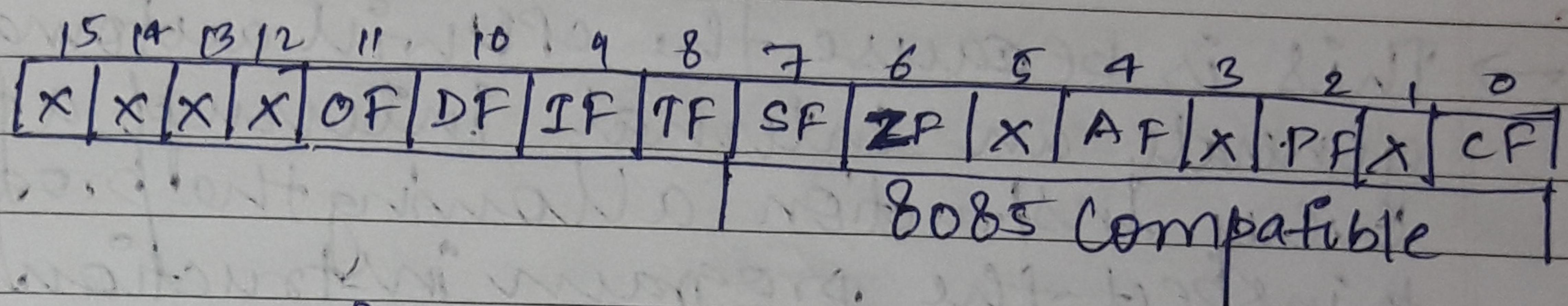
Date _____

Page _____

PF

Flag Register (16 bits) :

9 bits used



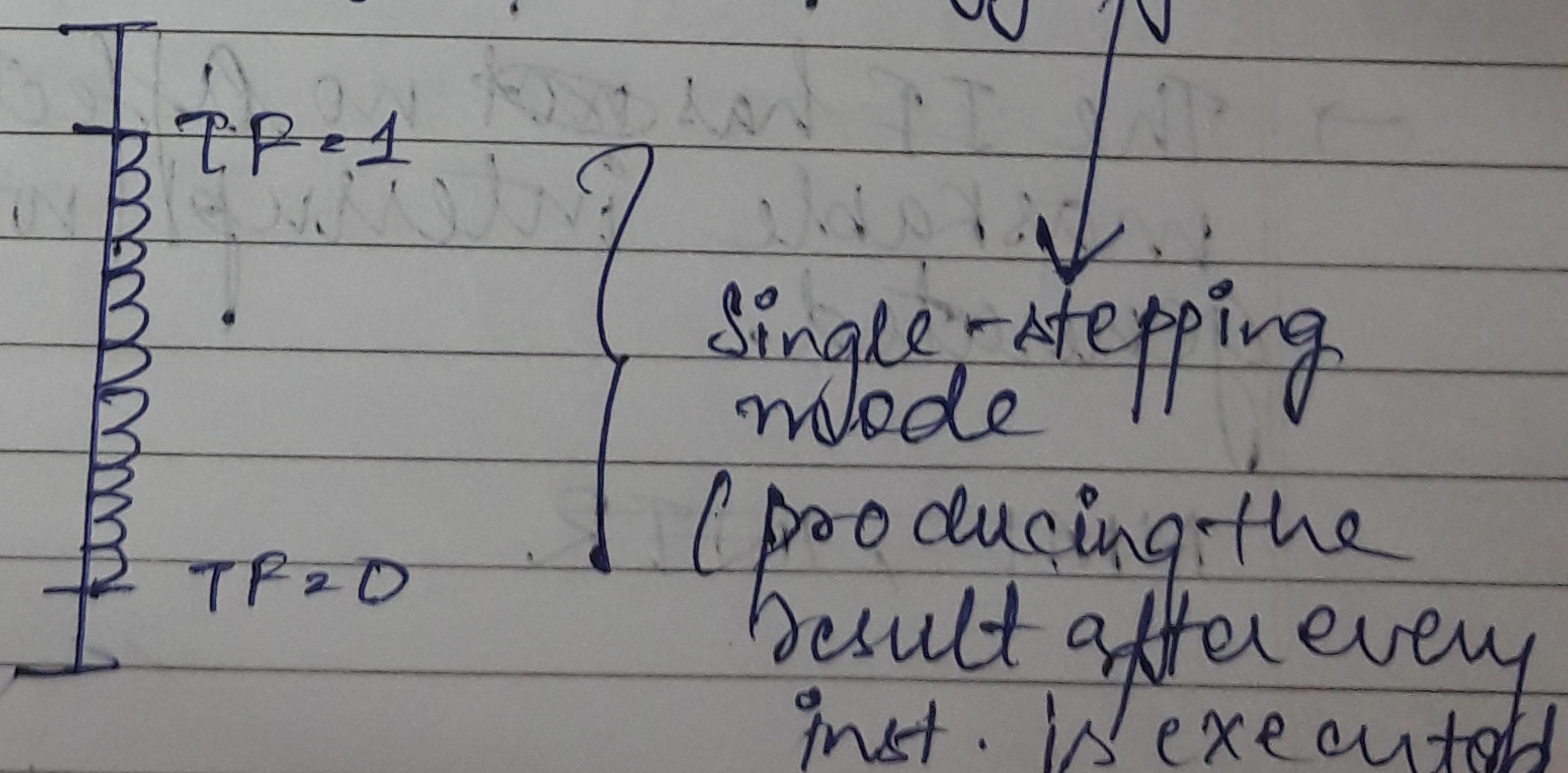
There are 2 types of flags in 8086 :

- Conditional flags : Set or Reset automatically
CF, PF, AF, ZF, SF, OF (overflow flag)
- Control flags : Trap Flag (TF)
Interrupt Flag (IF)
Direction Flag (DF)

The programmer can set or reset these flags.

Trap Flag :

When set, system goes into debugging mode.



- Setting this flag, the processor starts working in a single step mode for debugging. In single step mode, the processor executes an instruction & enters into a single step ISR.
- This is because the CPU will automatically generate an internal interrupt after every instruction allowing the programmer to inspect the program instruction by instruction.

Interrupt Flag :

No. of interrupts : 0 to 255 (256)

Some are user defined interrupts which can be generated by setting the IF equal to 1.

- If the user sets IF, the CPU will recognise an external maskable interrupt request.
- The IF has ~~not~~ no effect on the non-maskable interrupt which is externally generated.

~~NMI, INTR.~~

Direction Flag :

Used for string operation.

- It is used for string instructions
- In string instructions, we use SI & DI registers as offsets to point to the source area & the destination area.
 - SourceIndex \leftarrow
 - DestinationIndex \rightarrow
- The DF controls the direction of SI & DI pointers.
- If $DF = 1$, the string instruction will automatically decrement the pointer from the higher to the lower index.
- If $DF = 0$, vice versa.

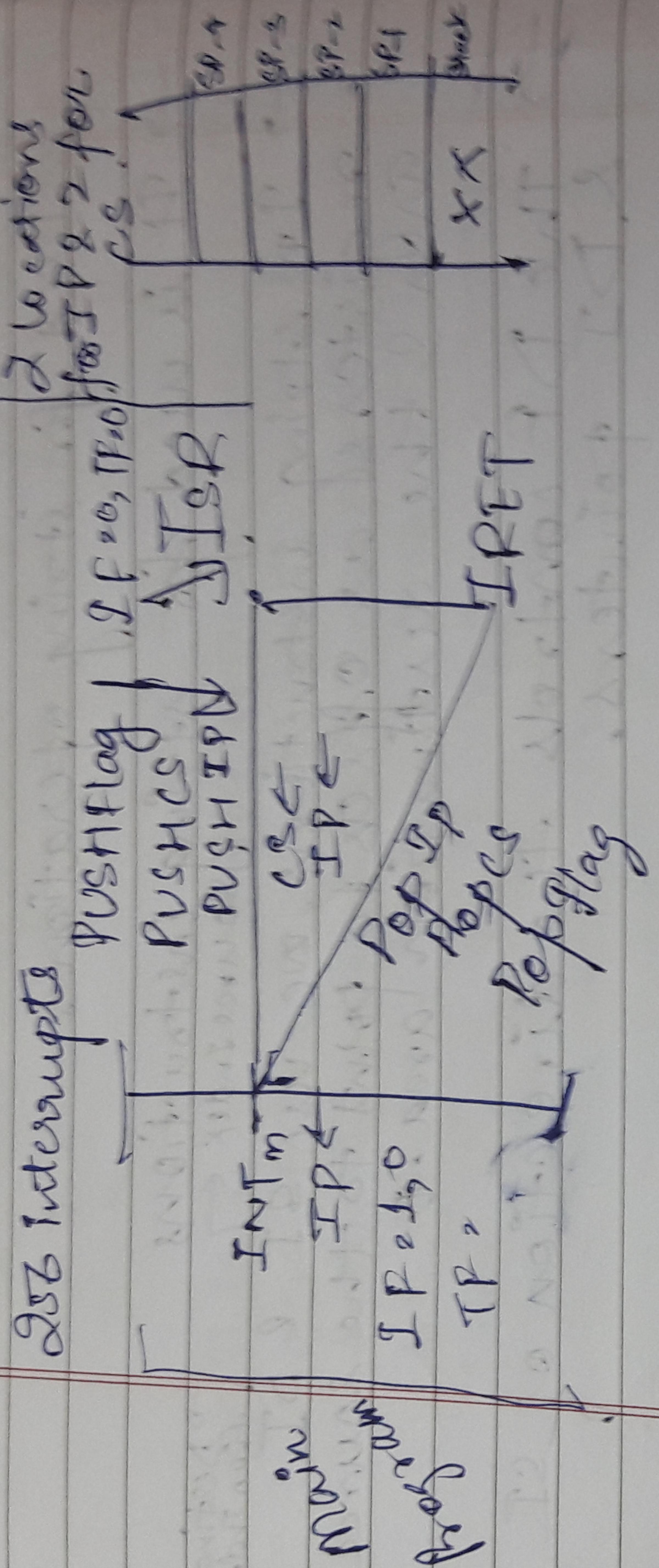
Overflow Flag:

If set, arithmetic overflow has occurred. That is, a significant bit has been lost becoz the size of the result exceeded the capacity of the destination location.

$CS \rightarrow 2^{16} \text{ bytes}$
 $IP \rightarrow 2^{16} \text{ bytes}$
 $\rightarrow [CS, IP]$

Interrupts

256 Interrupts



In segmentation, there are 2 types of branching:

- ① Inter segment ② Intra segment
(in different CS) (within the same CS)

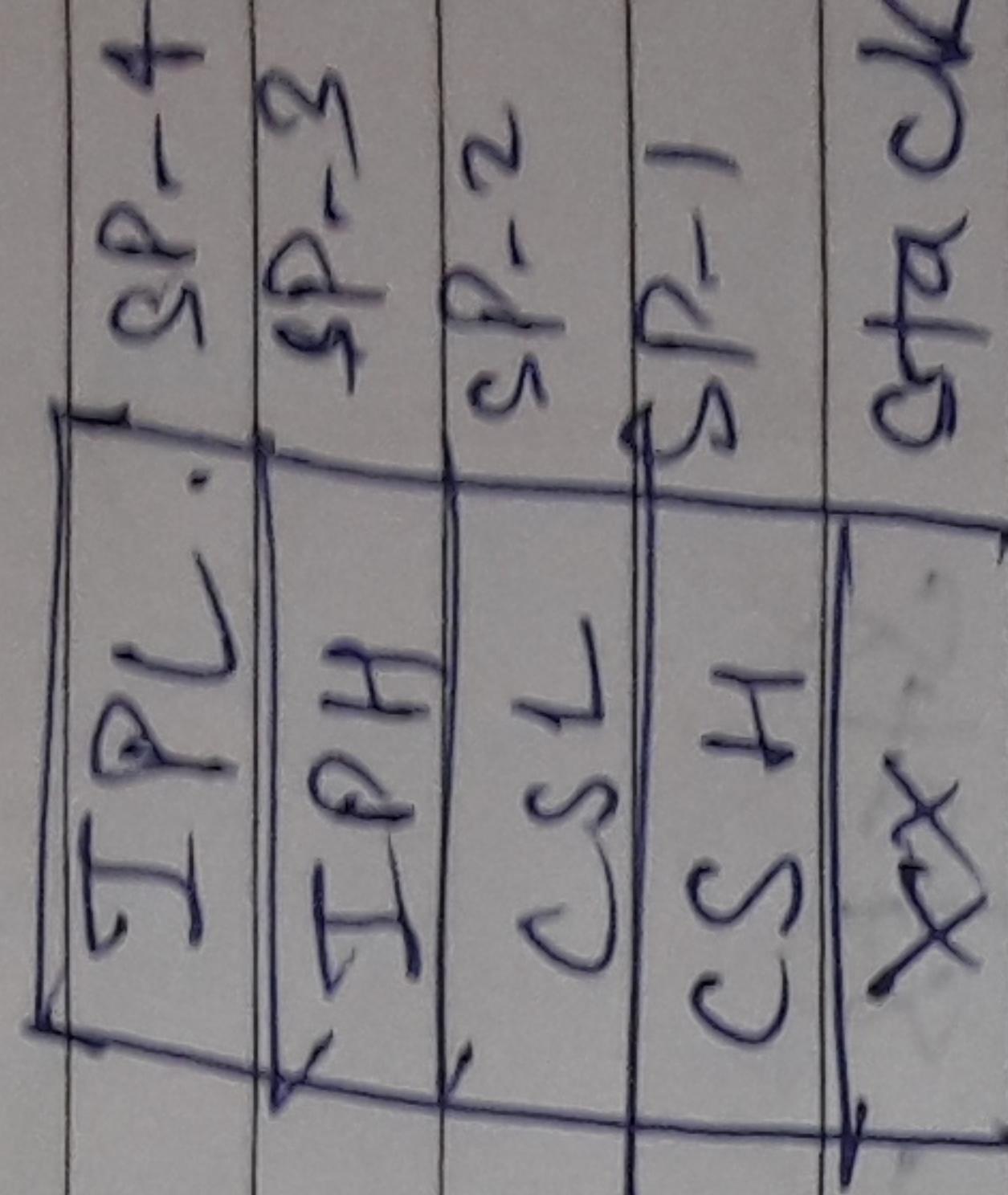
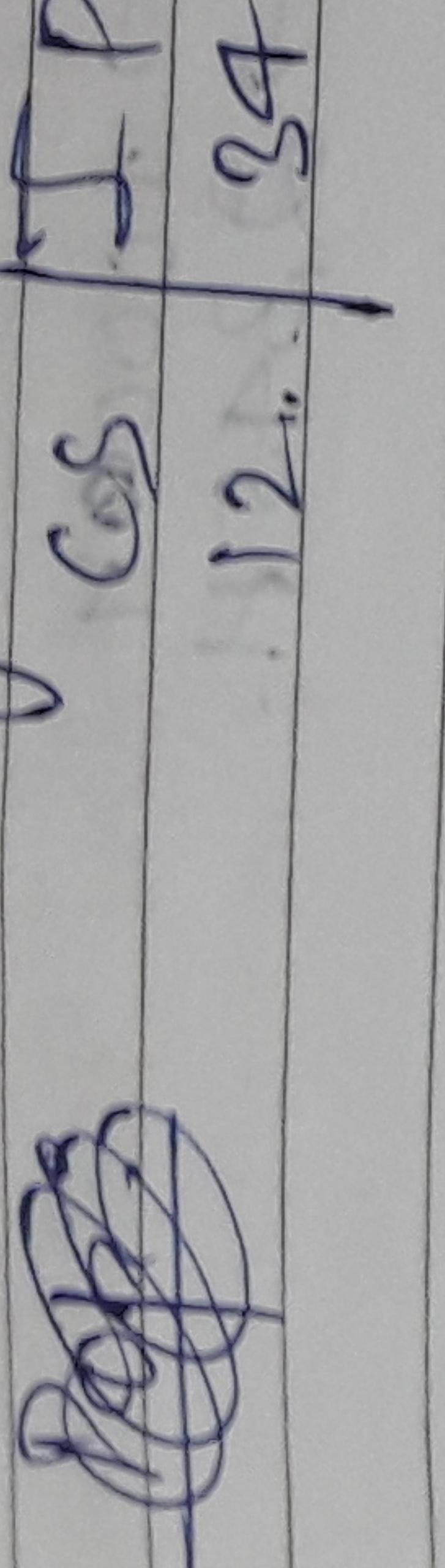
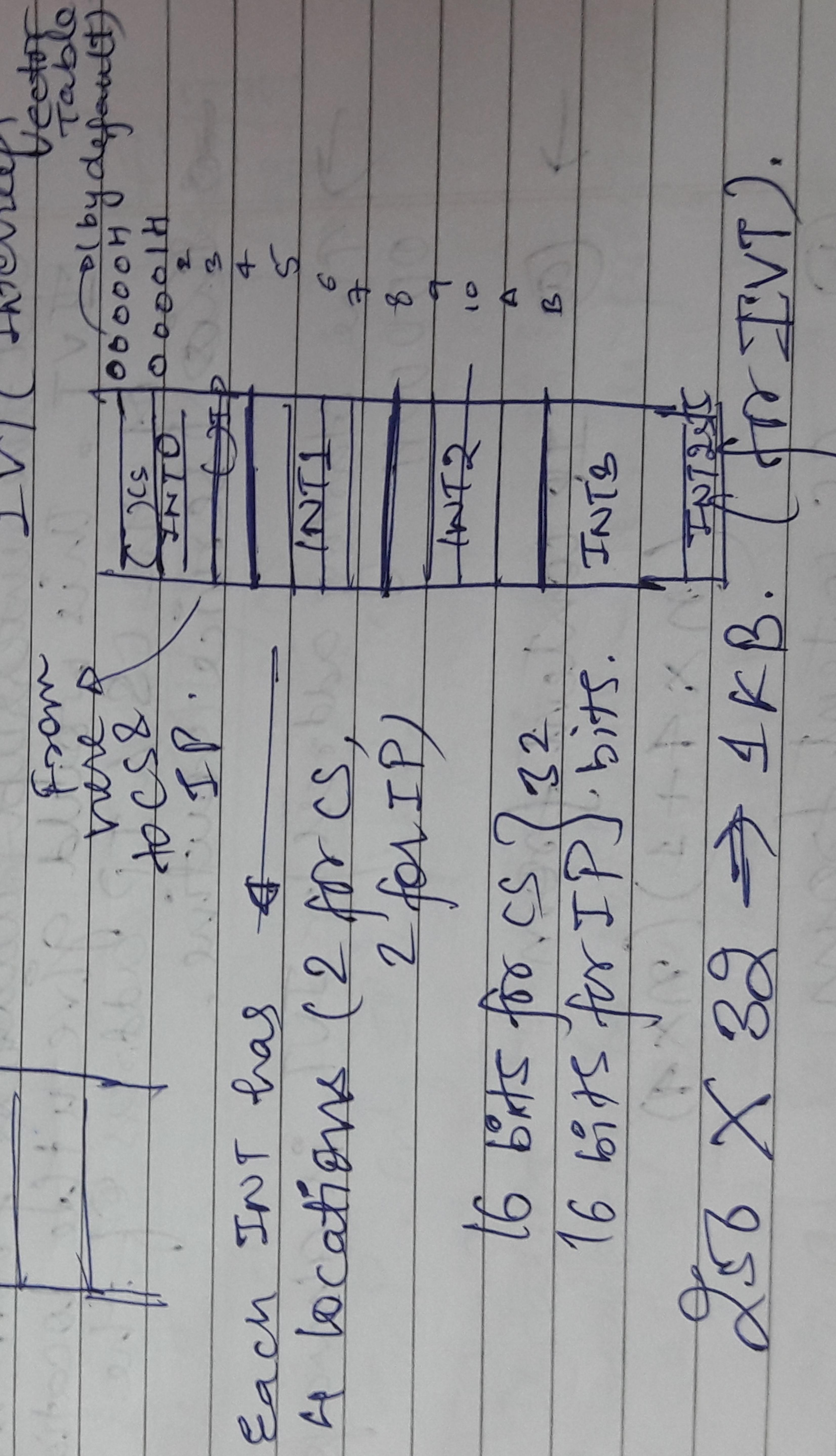
The main program & the ISR may lie in the same CS or different CS.

For Inter segment branching, the IP & CS address need to be stored on the stack.

For Intra segment, store IP on stack.

Push:

Lower byte \rightarrow lower address
 Higher byte \rightarrow higher address.

ISR

Eg: INT2 \rightarrow 12345
ISR address

CS \rightarrow 10000H
IP \rightarrow 2345H

Steps to follow if an interrupt is

→ Store the CS & IP of the next ~~loc~~ instruction of the main program in the stack

→ Get the interrupt address from the INT. This would give the location of the CS & IP address of the ~~the~~ service routine.

→ The starting address for INT is always 00000H

→ (a) IP content from

$$(n \times 4 + 1) \text{ } (n \times 4)$$

(b) CS content from

$$(n \times 4 + 3) \text{ } (n \times 4 + 2)$$

where $n \rightarrow$ interrupt no.

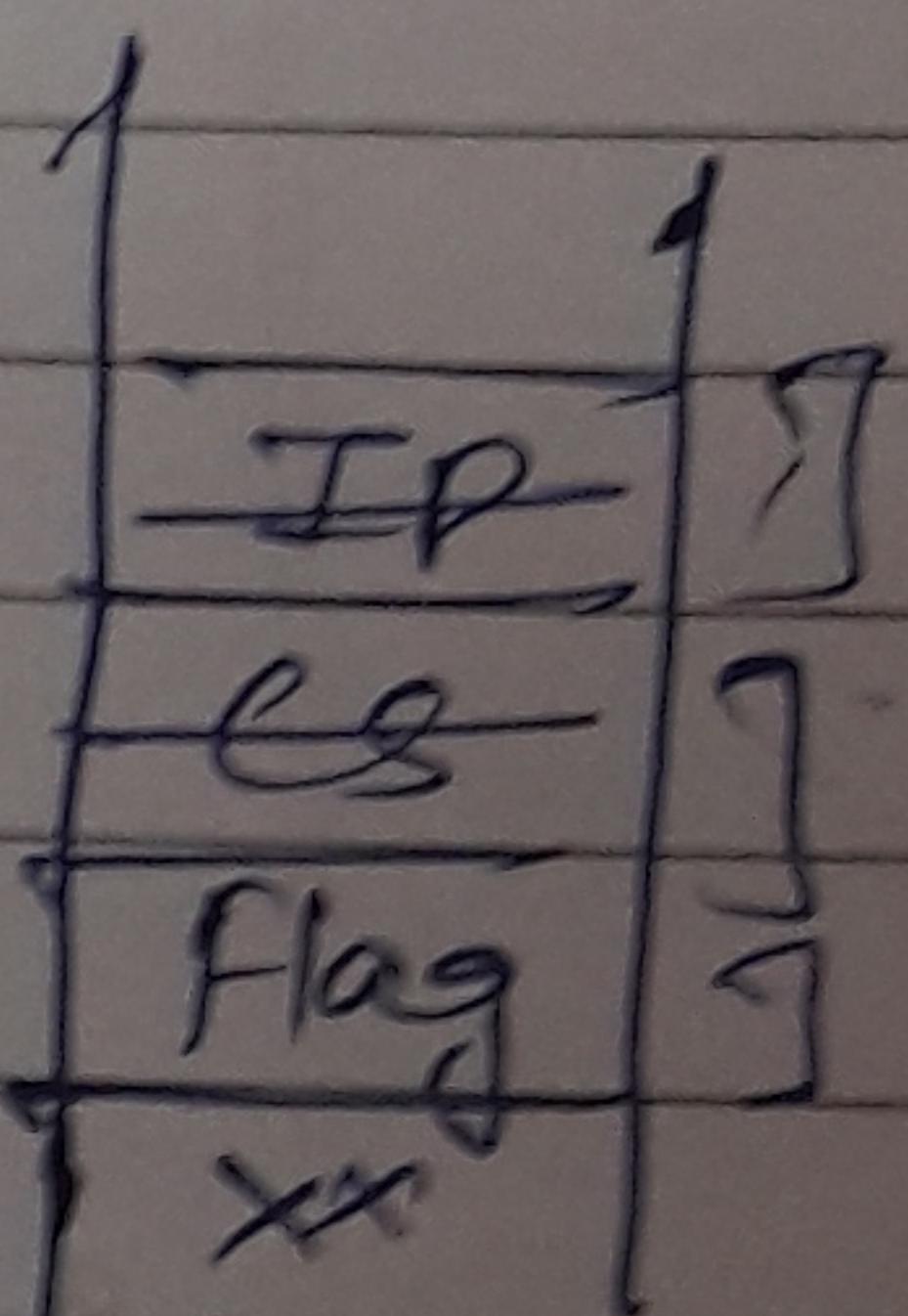
→ While coming back from the service routine, the microprocessor gets the return address from the stack using the POP instruction.

At a time, microprocessor can service only 1 interrupt.

When INTn is executed, IF was 1.
We need to clear it.

~~When~~ IF = 0, it ensures that no other interrupt is serviced.

How Interrupt Instruction is different from CALL/JMP?



- No need to store flags in CALL/JMP. Only IP is pushed.
- Location to which the control is transferred is within the main program in case of CALL/JMP.

Normal Service routine → RET
ISR → IRET

How to define for
are defined for
size of INT.

2 diagrams

Previous diagrams

classmate

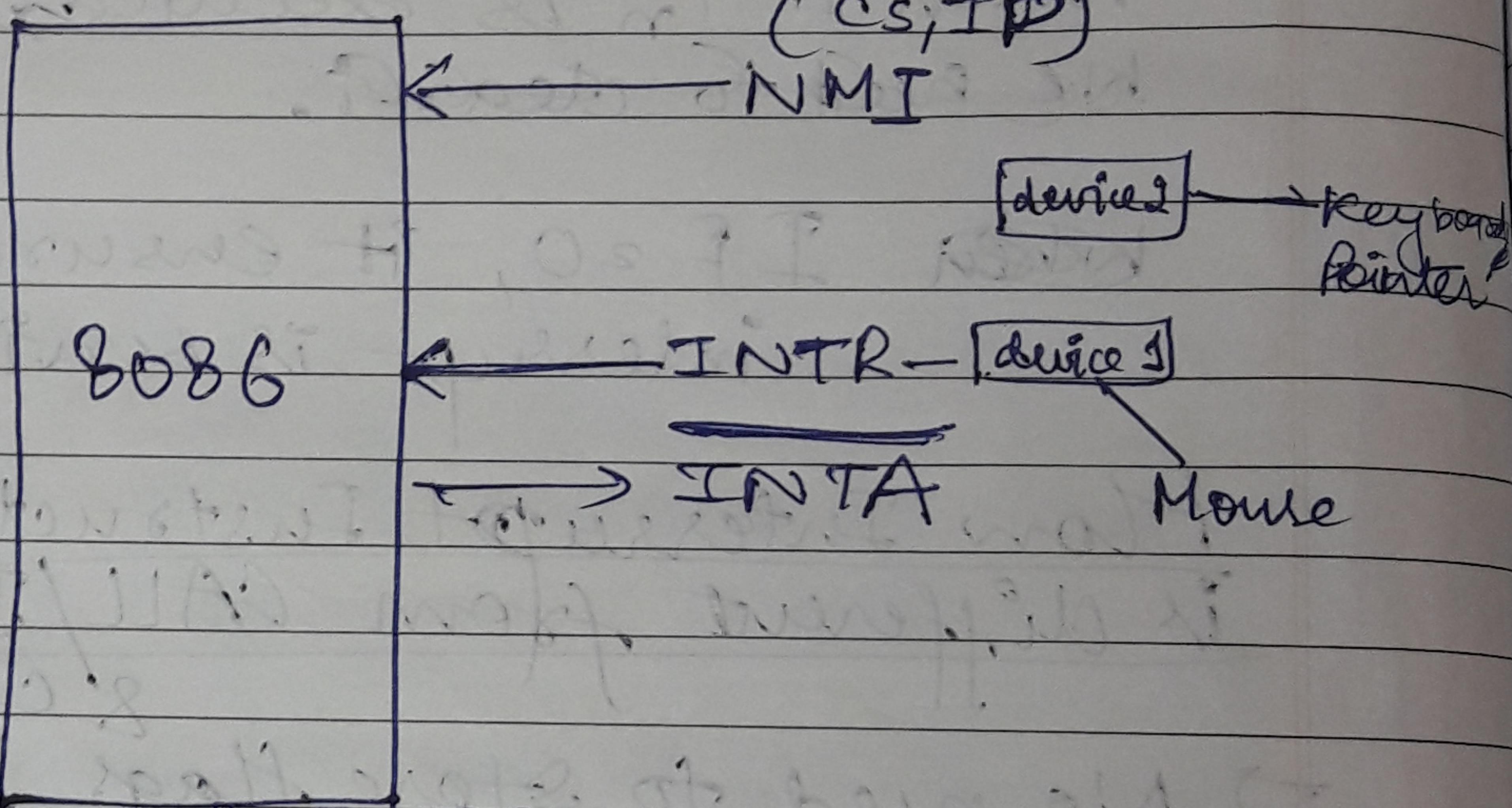
Date _____
Page _____

→ IVT

Divide error
NMI

- ① INT 0 → Divide error (very large)
- ② INT 1 → Single stepping (small)
- ③ INT 2 → NMI (Non interrupt, ext. generated)
- ④ INT 3 → Break Point
- ⑤ INT 4 → Interrupt overflow
- ⑥ INT 5 to 31 → Reserved (not defined)
- ⑦ INT 8 to 255 → User defined

IVT



Difference b/w NMI & INTR:

- ① NMI is faster as compared to INTR because the location of CS & IP for NMI is present in the IVT (that is known to microprocessor) & can be retrieved from the IVT.

But for INTR, location of CS & IP is not known. INTA is used to request for the CS & IP of INTR.

- ② NMI has higher priority than INTR.
- ③ NMI is non-maskable (but INTR is maskable).

(808) + NMI (CCS, IP)

8086

NMI

(CCS, IP)

INTR

INTA

IR 0

8259

PIC

IR 7

IR : Interrupt Request

PIC → Programmable Interrupt Controller

① NMI is not flexible.

② NMI is a vectored interrupt (has a fixed location in the INT). INTR is non-vectored.

$\leftarrow \rightarrow D \times A \times \frac{1}{B} X$

6 bits

X

for reading part.

not suppose

(8086 does

$A \times \frac{1}{B} \leftarrow \rightarrow D \times A \times \frac{1}{B}$

To multiply floating point numbers.

- Integer multiplication and division

$(C 8 \text{ bits} \times 16 \text{ bits}) \rightarrow A \times B \leftarrow C 16 \text{ bits} \times 16 \text{ bits}$

UB, SBE, INC, DEC

Instruction soft (8086)

along with the situation.
and in fax.

drawbar road → CWD
11110010100000
to word
a division

$$\begin{array}{r} 868 \\ \times 5 \\ \hline 430 \\ + 868 \\ \hline 4340 \end{array}$$

①

$$\begin{array}{r} 66 \\ \times 6 \\ \hline 396 \end{array}$$

$$\begin{array}{r} 88 \\ \times 4 \\ \hline 352 \end{array}$$

$$\begin{array}{r} 88 \\ \times 4 \\ \hline 352 \end{array}$$

② ACE

$$\begin{array}{r} 88 \\ \times 4 \\ \hline 352 \end{array}$$

(ADD. 6)
AC = 4
32

lower ~~upper~~ number \times 100 \geq 9

1

~~the earliest evidence of
the earliest known
of early human
and human~~

date of earliest
the earliest evidence
the earliest evidence

750

750

1

curve (parallel to left)

* Solution:

ProcessorProcessor Control Group Instructions:

① ~~PUSH~~ F ② ~~POP~~ F ③ ~~SAH~~ F

→ Store AH
into flag register

LAHF → Load flag register into AH.

Instructions for individual flags!

8 bits
(lower)

① STC → Set carry flag to 1.

② CLC → clear

③ STD → Set direction flag

④ CLD → Clear DF

⑤ STI → Set IF

⑥ CLI → Clear IF

Steps to set the trap flag:

PUSHF

POP BX

OR BH, AH.

PUSH BX

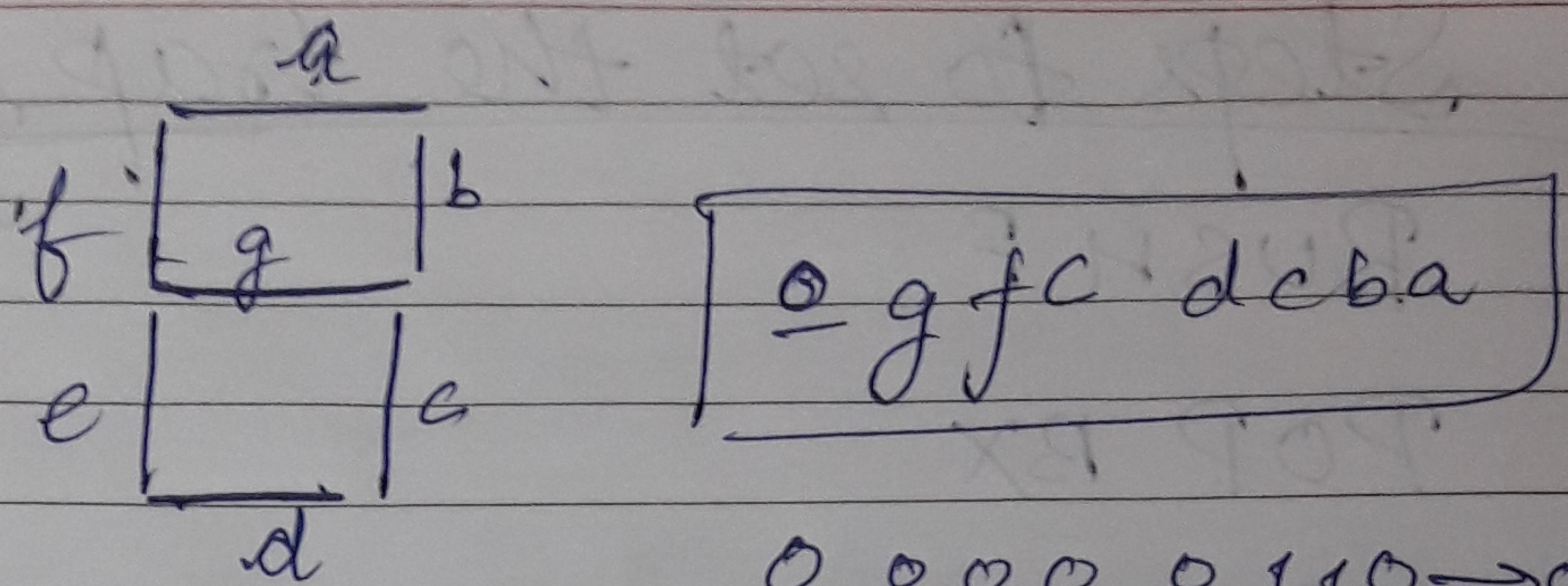
POPF

BH 0.000 0111
 0000 0001 (OR)
 0000 0111

BH 00000111
 11111110 B(AND)
 11111110 To ~~Set~~ Reset)

XLAT → Translate

AL ← BX + AL
 B ↗ offset
 address



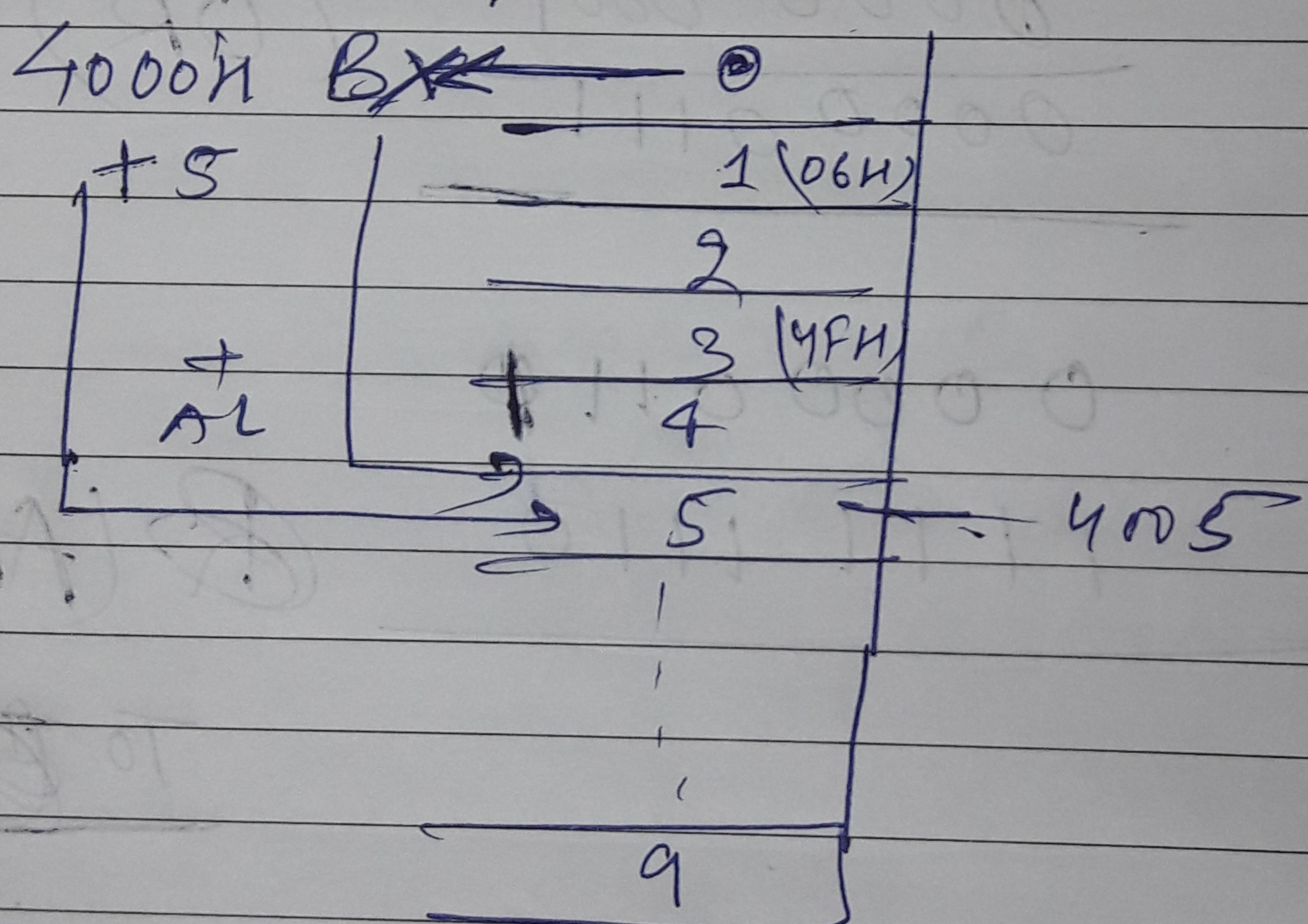
$0\ 0\ 0\ 0\ 0\ 1\ 1\ 0 \rightarrow 06H$

\rightarrow
decimal 1

$0\ 1\ 0\ 0\ 1\ 1\ 1 \rightarrow 4PH$

\rightarrow
Decimal
3

Look up table.



D Stores Lookup

MOV BX, 4000H.

MOV AL, 05H.

table &
Starting
address

\times LAT..

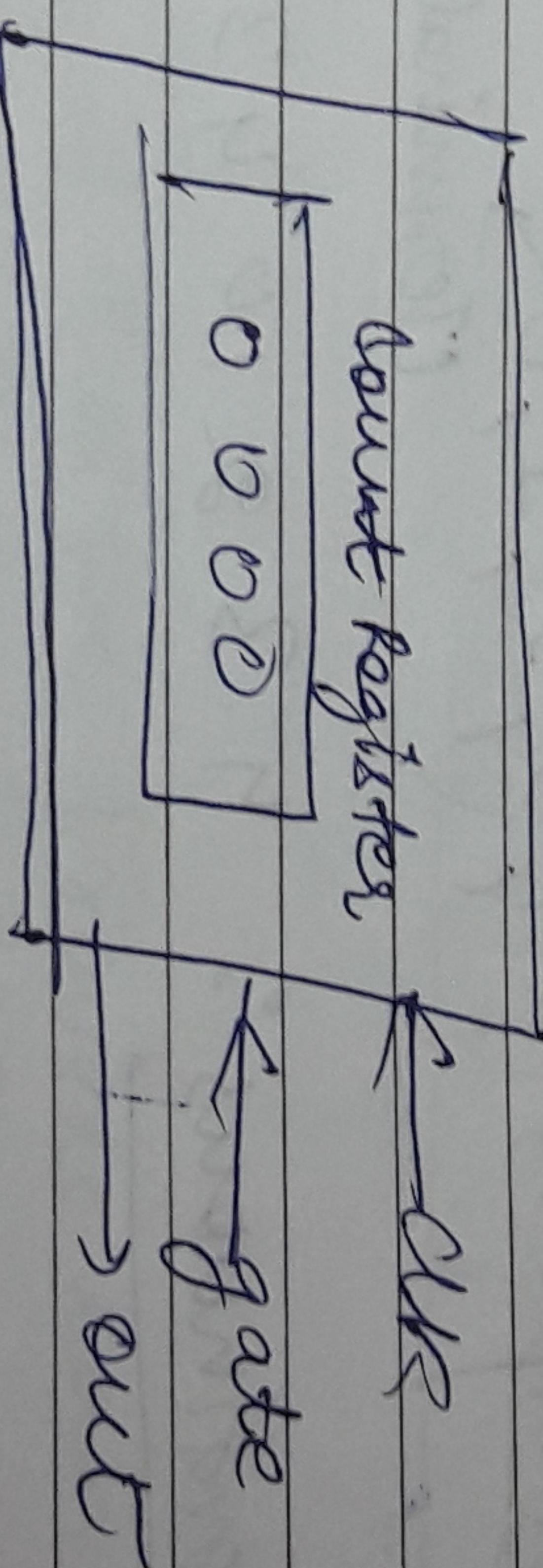
\rightarrow AL gets the code of 5

~~base~~ XLAT B
~~XLAT address~~
~~XLAT~~

Logical Instructions

NOT BL → BL ← BL

S254 Programmable Interval Timer



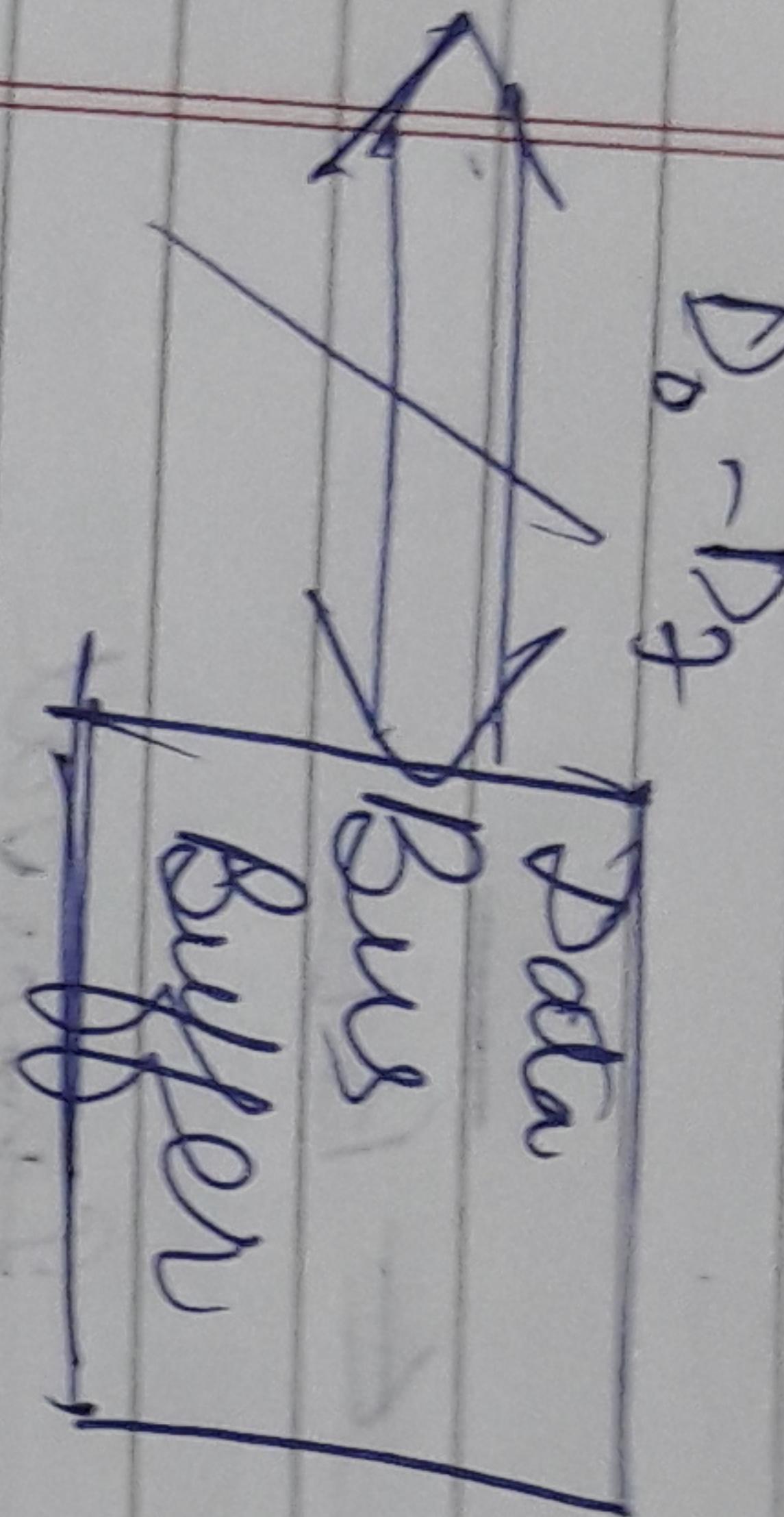
two modes

- ① Simple mode ② Continuous mode

Ques

It has 3 16-bit down counters which helps in producing 3 delays simultaneously.
 → The 16-bit count can go upto FFFFH, hence providing a very large amt. of delay.
 As 3 counters are down counters.

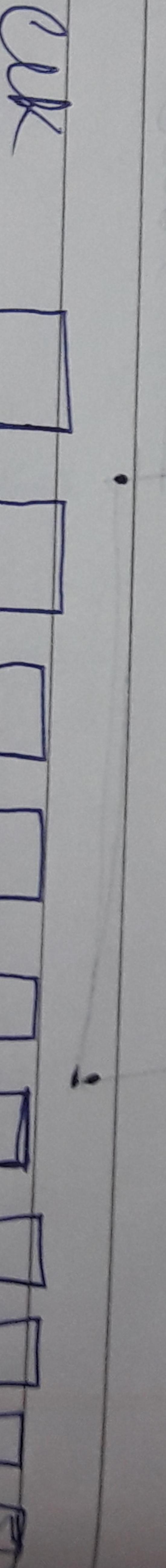
The data bus of 8259 is of 8-bit.
thus the count is given in 2 parts
the lower bit followed by the
higher bit.



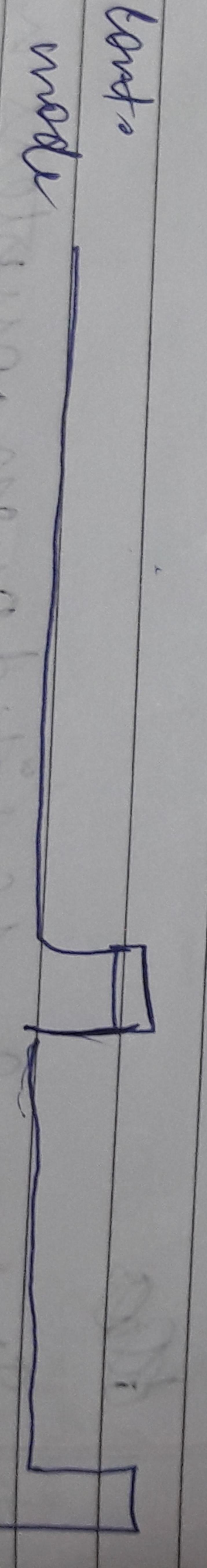
Simple: 9, 8, 9, 1, 0

(Terminal count)

Continuous: 9, 3, 2, 1, 0, 4, 3, 2, 0



~~Simple~~
~~Continuous~~
mode



cont.
mode

H/W Timer is better than S/w Timer.
 Up in case of S/w delay is engaged in
 doing this & cannot perform any other
 task.

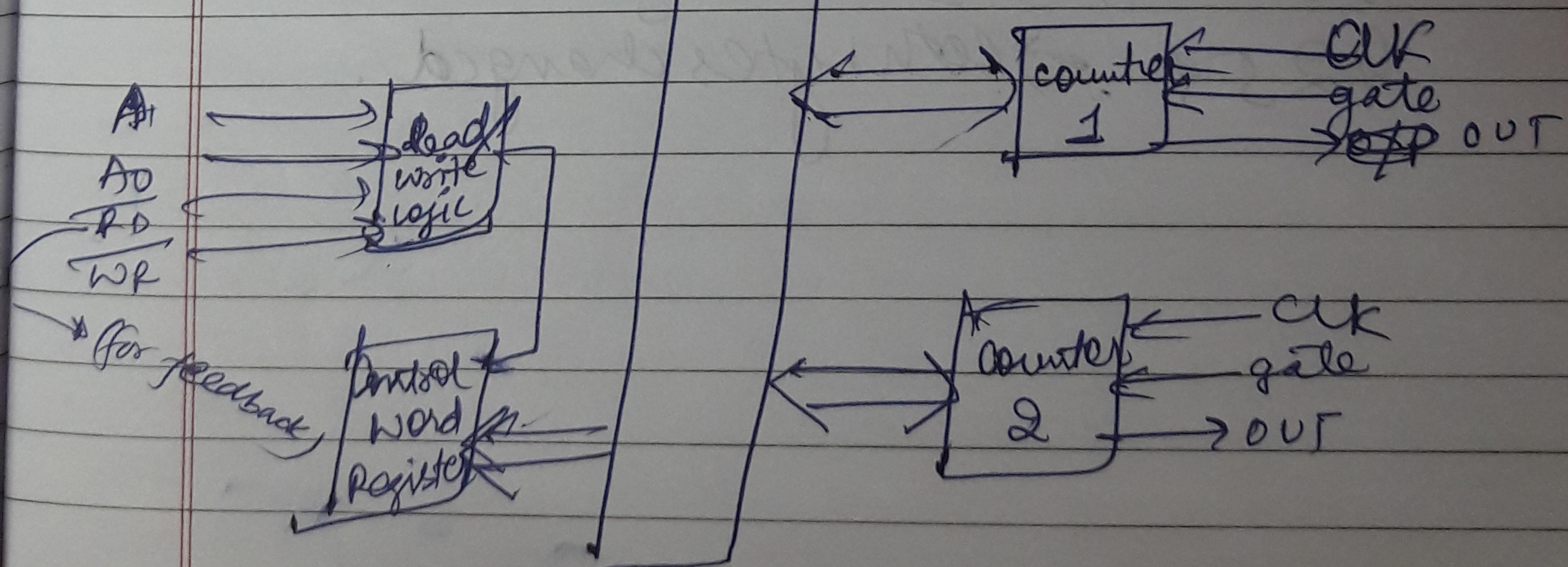
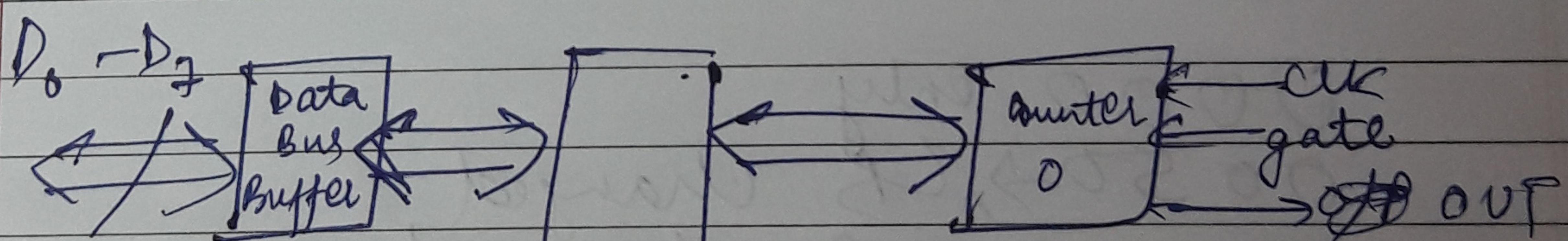
Gate is like the start signal to the timer to
 start functionality

gate $\leftarrow 0 \Rightarrow$ stops.

$$\text{O/P Freq} = \frac{\text{I/P Freq}}{\text{count}}$$

\hookrightarrow freq - generator

The count that we provide can be
 either in BCD or HEX.

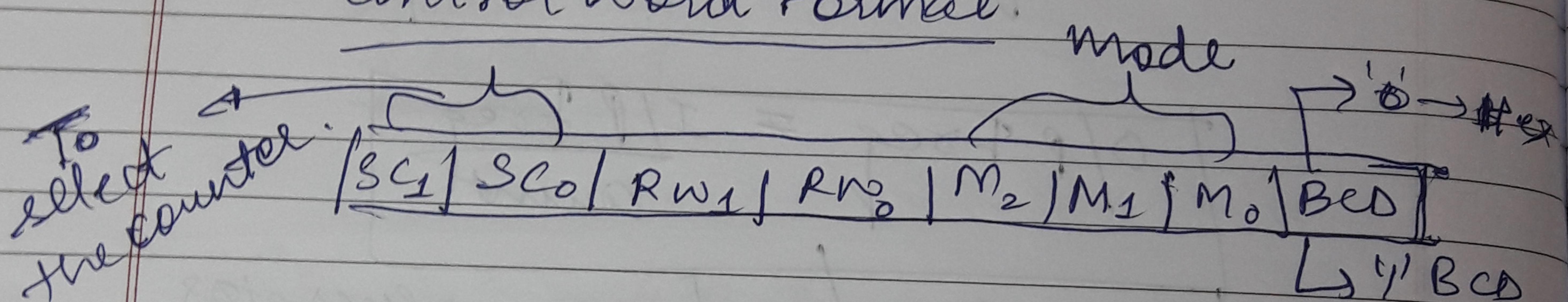


$A_1 \ A_0$

Address

- 0 0 → Counter 0
- 0 1 → Counter 1
- 1 0 → Counter 2
- 1 1 → CW (Control word)

Control word Format:

 $SC_1 \rightarrow SC_0 \rightarrow Status \rightarrow 00 \ 0$ $01 \ 1$ $10 \ 2$ $11 \ \text{Read}$

back

command

0000, Only

0050 → LB changed

5000 → HB changed

9259 → Both bytes changed.