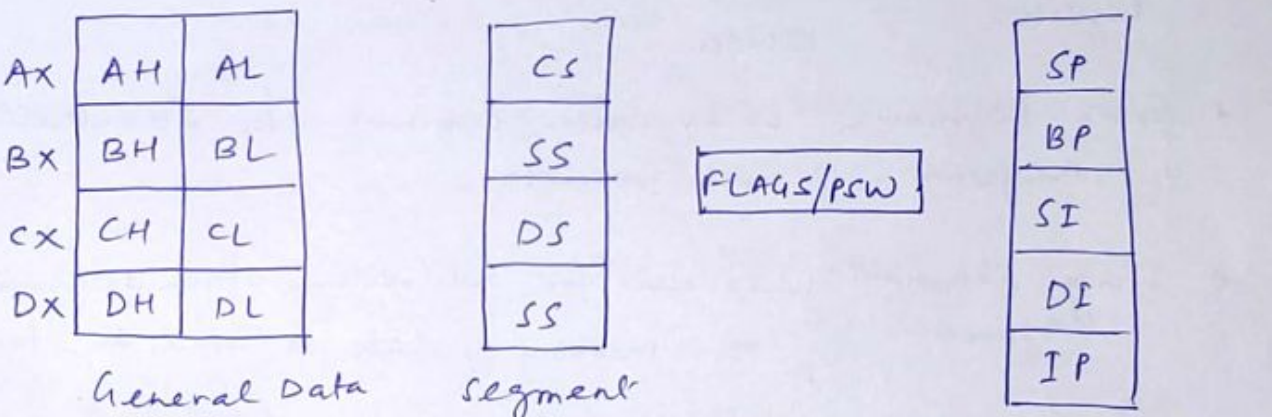
INTEL 8086 :-

① Register organisation :-

↳ General Purpose : can be used as both 8-bit and 16-bit registers.

Used for holding data, variables, intermediate results temporarily or for storing offset address, used like a counter etc.

↳ Special purpose : used as segment registers, pointers, index registers, offset storage registers.



General Data Registers. | Segment Registers. | Pointers and Index Registers.

(Register organisation of 8086)

General Data Registers :

- AX, BX, CX and DX — 4 - 16-bit registers
- AX - accumulator (AL - 8-bit accumulator)
- CX - default counter in case of string and loop instructions.
- BX - offset storage for forming physical address.
- DX - may be used as implicit operand or destination in case of a few instructions.

Segment Registers:-

* 8086 has a segmented memory.
* 1 MB memory divided into 16 logical segments. (may not be physically separated)
  ⇒ Each segment has 64 kb.
* 4 segment Registers – code segment (CS)
  Data segment (DS)
  Stack segment (SS)
  Extra segment (ES)

  single segment - more than 1 chip or more than 1 segment in a single chip.
* Code segment Register: addressing memory location in the code segment where executable program is stored.
* Data segment Register: point to the data segment, where data resides.
* Extra Segment Register: is an extra segment, also essentially used for data.
* Stack Segment Register: is used for addressing stack segment of memory, which is used to store stack data.
  → * Stack data – important data related to contents of CPU registers to be used at a later stage.
  → * SP grows down, data is pushed on to stack with decreasing addresses.
  → * while addressing any location in memory, the physical address is calculated from 2 parts:
    ① Segment address – in segment registers (16-bit)
    ② offset – pointers or index registers or ~~xxxx~~ BX. contain the offset.

* advantage of this scheme - instead of maintaining a 20-bit register for physical address, the processor just maintain 2 16-bit registers. (which are within word length capacity of the machine).

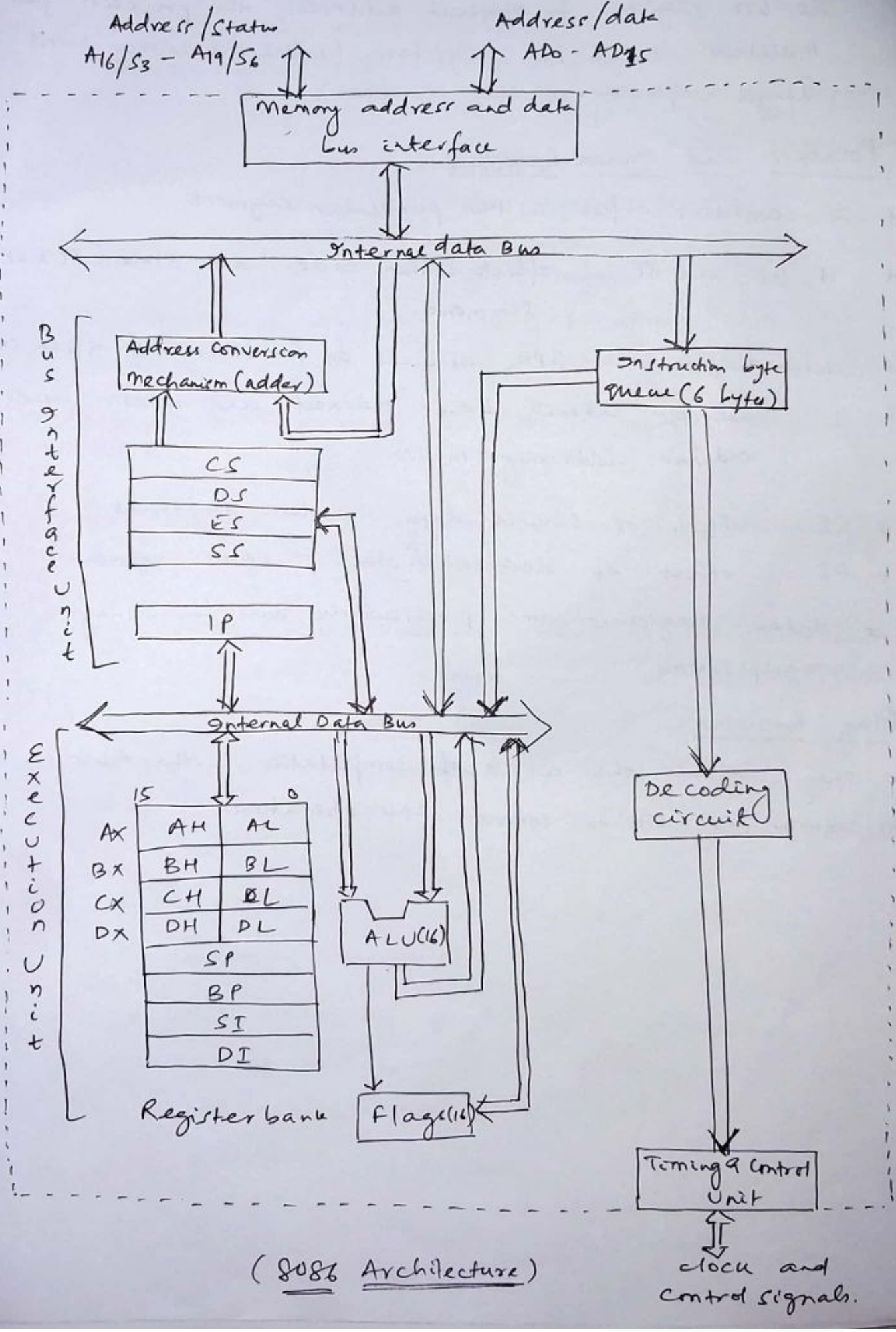
Pointers and Index Registers :-

* It contains offset within particular segments
* IP, BP and SP - offsets within code and stack (BP & SP) segments.
* Index Registers - GPR as well as for storage of offset in case of indexed, based indexed and relative based indexed addressing modes.
* SI - offset of source data in data segment
* DI - offset of destination data or extra segment
* Index Registers are particularly used for string manipulations.

Flag Register :

* They indicate the results of computation in the ALU.
* contain flag bits to control CPU operations.

Architecture :-



(8086 Architecture)

* Supports 16-bit ALU, a set of 16-bit registers, provides segmented memory addressing capability, rich instruction set, powerful interrupt structure, fetched instruction queue for overlapped fetching and execution etc.

* It is divided into 2 parts:
  (a) Bus Interface Unit (BIU)
  (b) Execution Unit (EU)

* BIU contains circuit for physical address calculations and a predecoding instruction byte queue (6-bytes long)

* BIU makes available system's bus signals for external interfacing.

* Responsible for making communication b/w external devices and peripherals including memory via bus.

* 8086 has a segmented memory.
  ↳ complete physical address is 20-bits
    ↳ generated using segment and offset registers of 16-bits each.

segment address → 1005H
offset address → 5555H

segment 1005H → 0001 0000 0000 0101

left-shifted by 4-bit positions
→ 0001 0000 0000 0101 0000

+

offset address → 0101 0101 0101 0101

Physical address → 0001 0101 0101 1010 0101
1 5 5 A 5

Segment address - 1005H — 0000H to FFFFH } offset

i.e. 64K locations in the segment.

» segment register - Base address

offset - distance of the required memory location in the segment from the base address.

from segment registers → from IP, BX, SI, DI, SP, BP or an immediate 16-bit value.

* BIU and EU form a pipeline.

Memory Segmentation:-

* complete physical memory is divided into a number of logical segments.
* Each segment = 64K bytes and addressed by one of the segment registers.
* 16-bit content of the segment registers actually point to starting location of a particular segment.
* To address a specific location in a segment, offset is used.
* Physical address range : 00000H to FFFFFH
  offset address range : 0000H to FFFFH
  segment address range: 0000H to F000H (Non-overlapping segments)

Flag Register:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	O	D	I	T	S	Z	X	Ac	X	P	X	Cy

Two parts: ① Condition code / status flags
② Machine control flags.

condition code / status flags : lower 8-bits of flag register and overflow bit

machine control flags: 3 flags - direction, interrupt, trap.

O = overflow - if result overflows into sign bit, then set.
D = Direction - if = 0 ⇒ string processed from LSB to MSB side (lowest to highest address)
I = Interrupt - set ⇒ maskable interrupts recognized by CPU
T = Trap - set ⇒ processor in single step execution mode
S = Sign - set when result is negative. (MSB = Sign flag)
Z = Zero - set when result of prev. computation is zero.
Ac = Auxiliary carry
P = Parity - set when lower byte of result has even no. of 1's.
Cy = Carry - carry / borrow out of MSB.
X = Unused.

SIGNAL DESCRIPTION / PIN DIAGRAM:-

CLK - Clock input it provides basic timing for processor operation and bus control activity.

Vcc - +5V supply

GND - ground for internal ckt

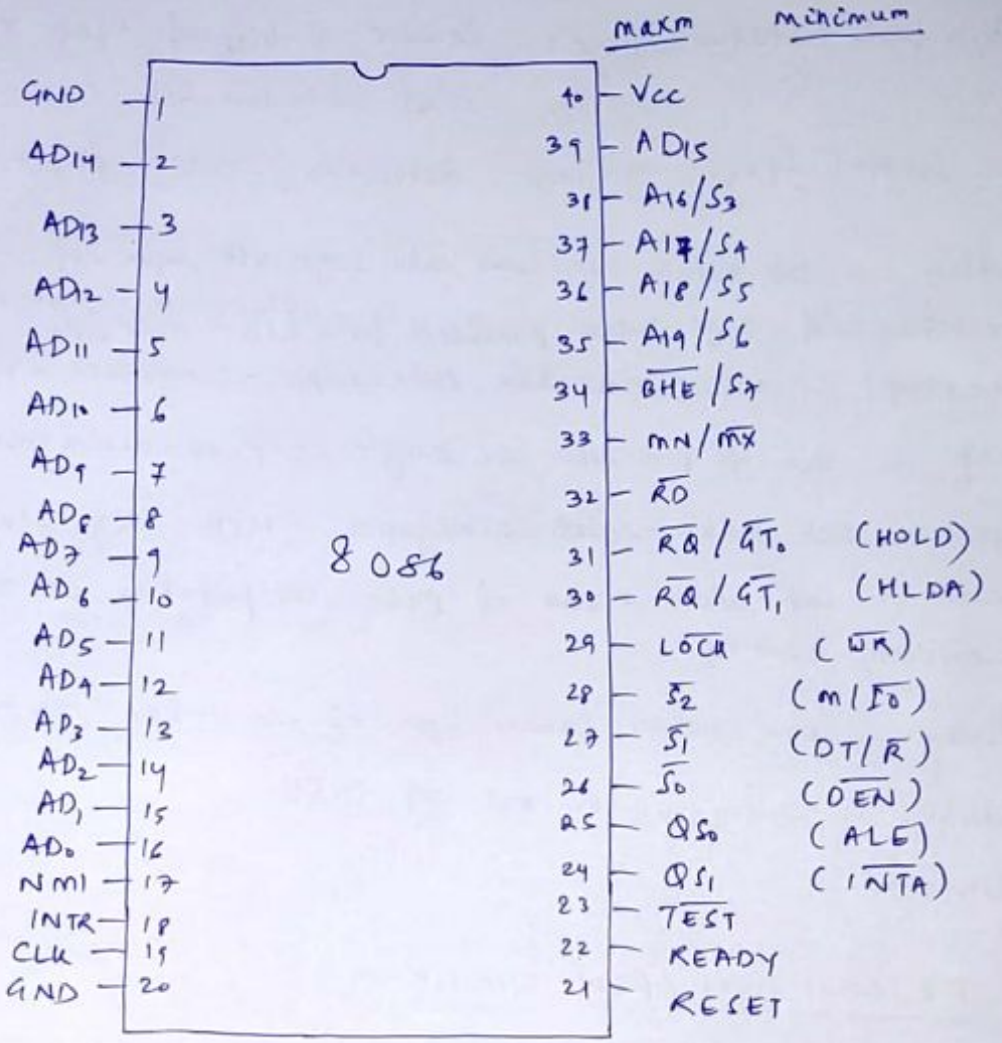
MN/$\overline{MX}$ - decides whether 8086 is going to operate in minimum (uni) or maximum mode. (multiprocessor)

* Pins for minimum mode of operation for 8086 are

M/$\overline{IO}$, $\overline{INTA}$, ALE, DT/$\overline{R}$, $\overline{DEN}$, HOLD, HLDA, $\overline{WR}$

* Pins for maximum mode of operation for 8086 are

$\overline{S_2}$, $\overline{S_1}$, $\overline{S_0}$, $\overline{LOCK}$, QS_1, QS_0, $\overline{RQ}/\overline{GT_0}$, $\overline{RQ}/\overline{GT_1}$



AD0 - AD15 - time multiplexed address and data
- address on the line - T_1 state
- data on the line - T_2, T_3, T_W (wait) & T_4.
- tristate during interrupt ack and local bus hold acknowledge

A19/S6 to A16/S3 - - time multiplexed address and status lines.
- address lines during T_1.
- during memory/IO operation, status info is available during T_2, T_3, T_W and T_4 states.
- status of interrupt enable flag (S_5) is updated at the beginning of each clock cycle.
- S_4 and S_3 are used together as follows:

S_4	S_3	Indication
0	0	Alternate data
0	1	stack
1	0	code or none
1	1	Data.

- tristate during local hold ack
- S_6 is always low.

$\overline{BHE}$ / S_7 :- $\overline{BHE}$ used to indicate transfer of data over the higher order ($D_{15} - D_8$) data bus. ($\overline{BHE} = 0$)

- tristated during hold.
- S_7 is not currently used.

$\overline{BHE}$	A_0	Indication
0	0	whole word (2 bytes)
0	1	Upper byte from or to odd address
1	0	Lower " " " even "
1	1	None.

$\overline{RD}$:- memory / IO read (Active low)
tristated during 'hold acknowledge'

READY :- acknowledgement from slow devices that they have completed data transfer.

INTR :- Interrupt signal (can be masked by resetting interrupt enable flag).

$\overline{TEST}$:- examined by a 'WAIT' instruction. If low, execution will continue, else processor remains in idle state.

NMI : Non-maskable Interrupt

RESET : causes the processor to terminate the current activity and start execution from FFFF0 H.

M/$\overline{IO}$: low - IO operation
high - memory operation.

$\overline{INTA}$: Interrupt acknowledgement

ALE : ALE = high, valid address on address/data lines.

DT/$\overline{R}$: direction of data flow.
low = processor receiving data
high : " transmitting data

$\overline{DEN}$: Indicates availability of valid data over the address/data lines.

HOLD/HLDA : HOLD = high ⇒ another device requesting bus access.
HLDA - hold ack.
low (means no one requesting)
↳ CPU lowers HLDA to 0.

HOLD is asynchronous and is externally synchronized.

$\overline{S_2}$, $\overline{S_1}$, $\overline{S_0}$ - Status lines.

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	Indication
0	0	0	Interrupt acknowledge
0	0	1	Read I/O port
0	1	0	write I/O port
0	1	1	Halt
1	0	0	code access
1	0	1	Read memory
1	1	0	write memory
1	1	1	passive.

$\overline{LOCK}$: $\overline{LOCK}$ = 0 (No other device can control the system bus at that time as CPU is executing a critical instruction)

QS_1, QS_0 : status of code-prefetch queue.

QS_1	QS_0	Indication
0	0	No operation
0	1	First byte of opcode from Queue
1	0	Empty Queue
1	1	subsequent Byte from Queue

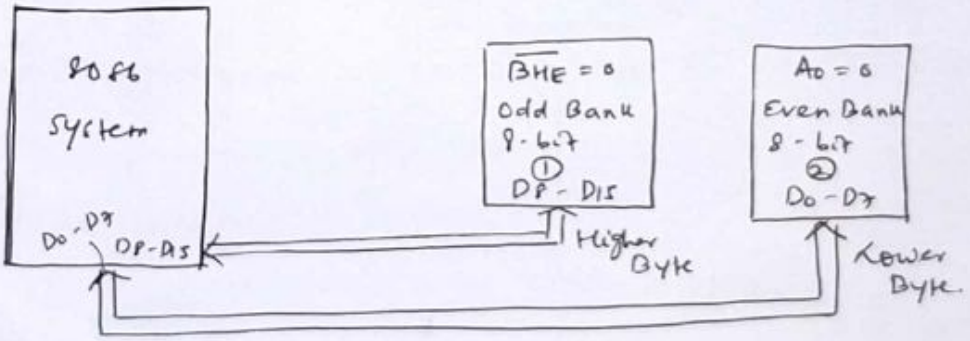
$\overline{RQ}/\overline{GT_0}$, $\overline{RQ}/\overline{GT_1}$:- used by local bus masters (devices) to force the processor to release the local bus at the end of processor's current bus cycle.

$\overline{RQ}/\overline{GT_0}$ - higher priority than $\overline{RQ}/\overline{GT_1}$

Working :- * request to CPU
* Hold ack and CPU BIU disconnected.
* 1 clock cycle wide pulse from another master indicates that 'hold' request is about to end. and 8088 may regain control of the bus on the next clock cycle.

PHYSICAL MEMORY ORGANISATION:-

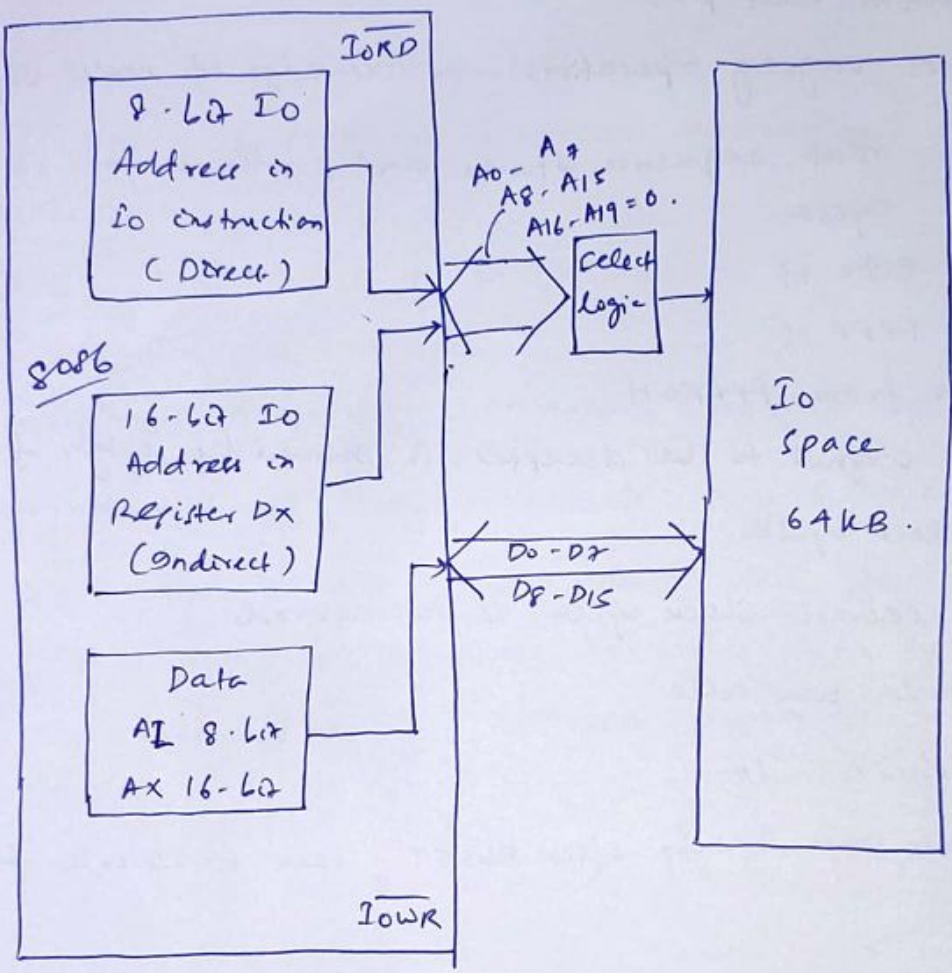
* 1 MB < odd bank / even bank } 512 Kbytes each.
* Byte data - even address - D0 - D7
  " - odd address - D8 - D15
* $\overline{BHE}$ and A0 handle selection of banks.
* Processor fetches a word from memory →
  1. Both the bytes may be data.
  2. Both the bytes may contain opcode bits.
  3. One of the bytes may be opcode and other may be data.
* Word data is located at even address ⇒ Better, because only one read/write cycle is required.
* 8086 is a 16-bit µp and can access 2 bytes of data in one operation (i.e. in a single machine cycle) but commercially available memory chips are 1-byte in size.
  - In order to store 16-bit data, 2 successive memory locations are used.



* 00000H - lower 8-bits.
  00001H → Higher 8-bits
* certain locations are reserved for specific CPU operations.
  FFFF0H to FFFFFH - jump to initialisation
  00000H to 003FFH - interrupt vector table.
  ↓
  256 types of interrupt / CS & IP for each interrupt require 4 bytes for storing interrupt vector table.

I/O Addressing capability :-

* upto 64K I/O byte registers or 32K word registers.



(IO Addressing in 8086)

* IO Address appears on address lines A_0-A_{15} for T_1. (using ALE signal)
* A_{16} to $A_{19} = 0$. during I/O operations.
* DX is 16-bit IO address pointer.
* Addressing mode is ~~decided~~ based addressed mode using contents of BX.
* Even addressed bytes - D_0 to D_7 (8-bit of Even addressed).
* odd addressed bytes - D_8 to D_{15}.

Special Processor activities :-

① Processor Reset and Initialisation :

* Logic 1 applied to reset pin,
* CPU terminates ongoing operation. - on +ve edge of reset signal
* negative edge - reset sequence starts and continues for 10 clock cycles.
* All register - 0000 H
  CS = FFFF H
* Execution starts from FFFF0 H
* For a reset signal to be accepted, it should be high for atleast 4 clock cycles.
* NMI before second clock cycle is not served.
* Status signals are idle
* ALE and HLDA - low.
* HOLD appears immediately after RESET, then HOLD will be served.

② Halt :

* Halt ~~duer~~ when HLT instruction is encountered
* Halt — Minimum mode (issues ALE pulse but does not issue any control signal)
  \ Maximum mode (status 011 in $\overline{S_2}$, $\overline{S_1}$, $\overline{S_0}$ pins, issues an ALE pulse but no qualifying signal (no appropriate address / control signals are issued to the bus).
* only an interrupt request / reset will force 8086 to come out of halt state.

$\overline{TEST}$ and Synchronisation with External Signals:

* When CPU executes a wait instruction, CPU preserves the contents of the registers before execution of WAIT instruction.
* Waits for $\overline{TEST}$ pin to go low
* $\overline{TEST}$ = 0 (continues further execution) otherwise keeps on waiting.
* For a $\overline{test}$ signal to be accepted, it should be low for atleast 5 clock cycles.
* waiting does not consume any extra bus cycle.
* while waiting, HOLD request may be served.
* Interrupt occurs ⇒ one more wait then interrupt served.
* After interrupt handling complete, then fetches wait instruction and continues to be in 'wait' state.