8086 µP

→ It is a 16-bit microprocessor. Thus, most of its instructions are designed to work with 16-bit binary words.
→ It has 20 address lines, so it can address 2^{20} = 1MB memory location.
→ It has multiplexed address and data bus which reduce the number of pins needed, but slightly slow down the transfer of data.
→ It requires only one +5 volt supply voltage.

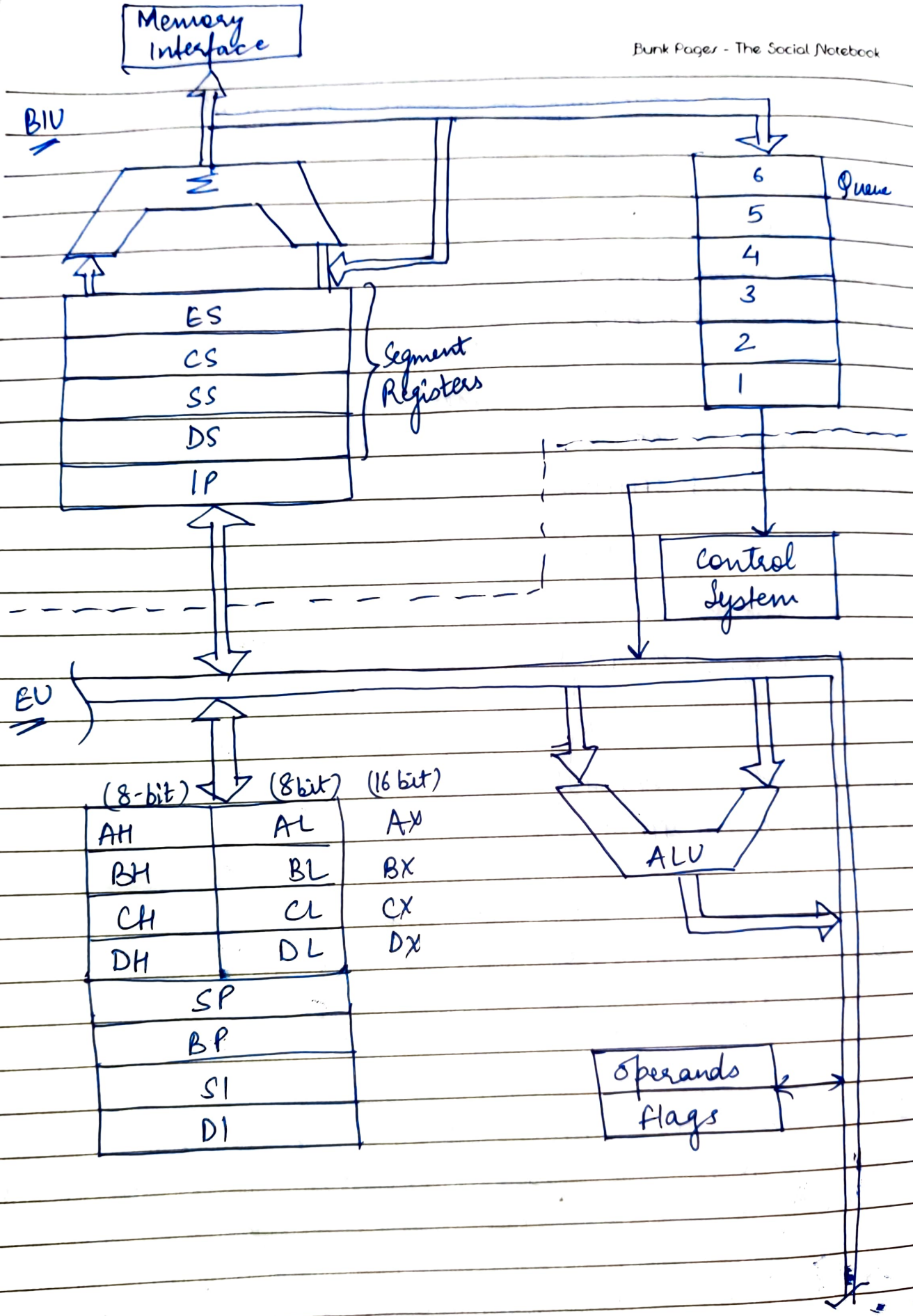
* 8086 Architecture

(1) BIU

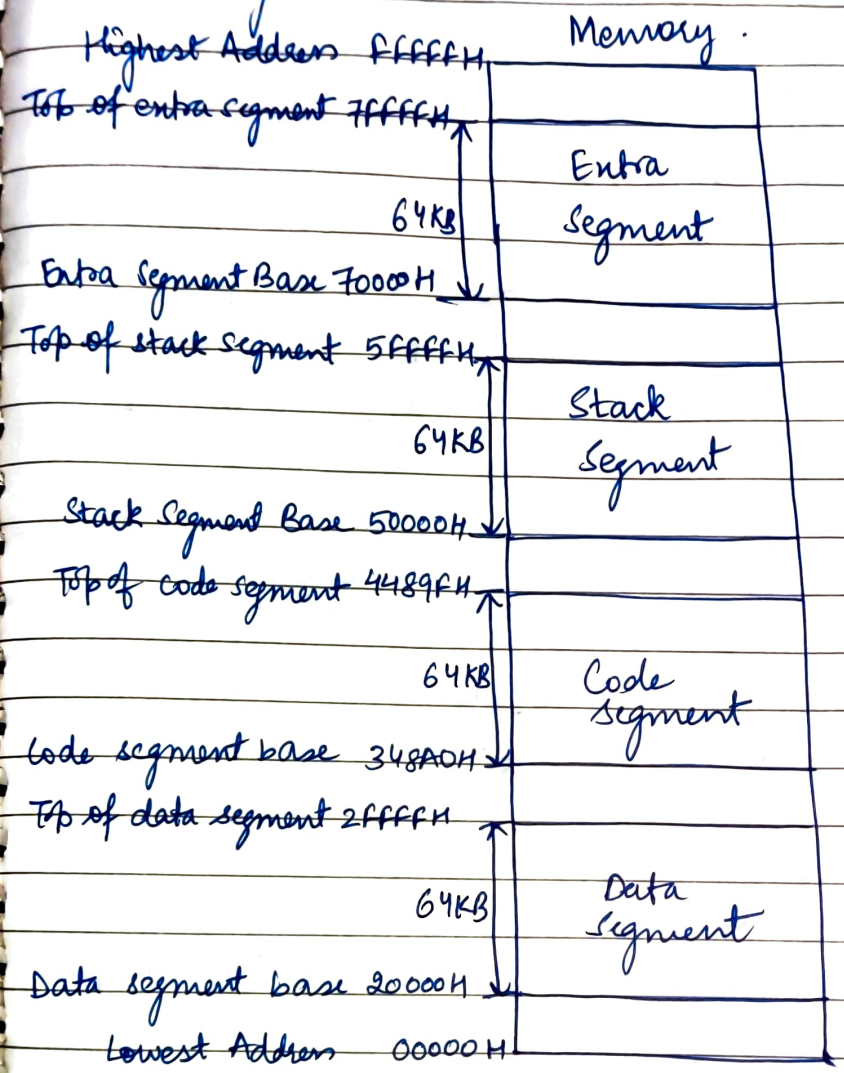
 # Instruction stream byte queue

→ The instruction stream byte queue is also called the queue. To speed up program execution, the BIU fetches six instruction byte ahead of time. These prefetched instruction bytes are held in a group of ~~instruction~~ registers called queue.
→ The BIU stores the prefetch bytes in a first in first out principle (FIFO)

When EU (Execution Unit) is ready for the execution of next instruction, the EU simply reads the prefetched instruction byte from this queue. This is much faster than getting the instruction bytes directly from memory. This process of prefetching the next instruction once the current instruction executes is called pipelining.


Memory Interface
BIU
6
5
4
3
2
1
Queue
ES
CS
SS
DS
IP
Segment Registers
Control System
EU
(8-bit)
(8 bit)
(16 bit)
AH
AL
AX
BH
BL
BX
CH
CL
CX
DH
DL
DX
SP
BP
SI
DI
ALU
Operands
flags

⇒ Segmented Addressing in which the available memory space is divided into chunks called segments such as memory is called segmented memory.



Merits of Memory Segmentation

1→ It allows the memory addressing capacity to be 1MB even though the address associated with individual instructions are only 16-bit long.

2→ It allows multiprogramming and multiprocessing

3→ It facilitate the use of separate memory areas for a program, its data and stack

The 8086 processor provides an instruction pointer (IP) which holds the 16-bit address of the next instruction to be executed in the current code segment (CS). The 16-bit content of instruction pointer is referred as offset or effective address

PA = Segment Address X 10H + offset Address
↑ Physical Address.

② EU (Execution Unit)

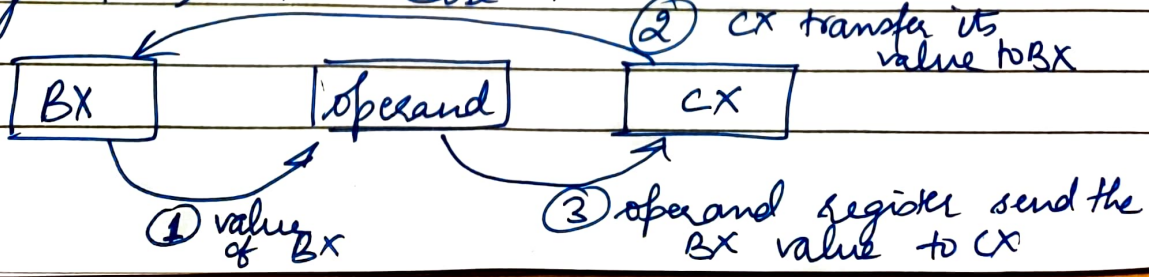
Control System :-
→ It is used to decode the instruction and provide the result to ALU so that the ALU can performed the desired operation.

operands -
→ Operand register is a temporary register.
→ This is used for storing the temporary value

For ex :- We need to perform.

XCHG BX, CX

then in this case BX and CX will exchange their values, but if CX first gives the value to BX then its value get updates and the previous value of the BX gets lost. So we use operand register in this case.



8086 uP

Registers in Execution Unit

(i) **AX [Accumulator Register]**

The register AX is used as 16-bit accumulator whereas the lower byte register (AL) can be used as 8-bit accumulator for 8-bit operation.

(ii) **BX [Base Register]**

This register is used as offset storage for generally physical address in case of a certain addressing modes.

(iii) **CX [Count Register]**

The register is used as a default counter in case of string and loop instructions.

(iv) **DX [Data Register]**

The register is used as a destination in case of few instructions. It can also hold the overflow from certain arithmetic instructions.

Stack pointer

→ The stack pointer register contains the 16-bit offset address of the stack top.

⇒ For stack operations, physical address is produced by adding the content of stack pointer register to the stack segment base address.

Base Pointer

→ The base pointer can be used instead of stack pointer for accessing the stack in the base addressing mode ~~this~~.

→ This physical address can be calculated by adding the content of base pointe to the stack segment base address.

Source Index

→ The source index register is used to store the offset address of data in data segment.

→ In this, physical address can be calculated by adding the offset of source index to the data segment base address.

Destination Index

→ The DI is used to store offset address of data in data or extra segment.

→ In this, physical address can be calculated by adding the content of destination index to the data segment (or extra segment) base address.

Flag Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U	U	U	U	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF

Flags bit from 0-7 are ~~as~~ as same as 8085 flag register

⇒ **Overflow flag**

If this flag is set, it ~~decide~~ indicates that the result is out of range. If the result is not out of range, the overflow flag is reset.

⇒ **TRAP flag**

When the 8086 goes into the single step mode (i.e 8086 executes only one instruction at a time) this flag is set otherwise it is reset.

⇒ **Direction flag**

→ This flag determines the direction in which string operation will occur.

→ If DF = 1, the string instruction automatically decrements the address, so that data transfers from high address to the lower address.

→ If DF=0, the string instruction increments the address and data transfers from lower address to higher address.

⇒ **Interrupt flag**

→ This flag must be set to recognize interrupt request at the INTR input of 8086.

→ When this flag is reset, the request at INTR input is ignored and interrupt interface is disabled.

Note :-

① Queue store 6 bytes of instructions, not 6 instructions.

② A no. like (25H, 20H, ...) is never included in opcode.

③ Opcode only specifies operations & registers.

④ BIU waits for 2 bytes of queue to get empty to refill it. Why?
2 bytes means 16 bits, 8086 transfers 16 bits in one cycle.

⑤ Limitation of pipelining
→ It fails when there is a branching operation.
The queue gets cleared or discarded when there is branching

8086 pin configuration

			MAX MODE	MIN MODE
Ground → V_{SS} (GND)	1	40	V_{CC} (5P) ← supply	
AD_{14}	2	39	AD 15	
AD_{13}	3	38	A16/S3	
AD_{12}	4	37	A17/S4	
AD_{11}	5	36	A18/S5	
AD_{10}	6	35	A19/S6	
AD_9	7	34	$\overline{BHE}$/S7	
AD_8	8	33	MN/$\overline{MX}$ ← Shows min/Max mode	
AD_7	9	32	$\overline{RD}$	
AD_6	10	31	$\overline{RQ}$/$\overline{GT0}$	HOLD
AD_5	11	30	$\overline{RQ}$/$\overline{GT1}$	HLDA
AD_4	12	29	$\overline{LOCK}$	$\overline{WR}$
AD_3	13	28	$\overline{S2}$	M/$\overline{IO}$
AD_2	14	27	$\overline{S1}$	DT/$\overline{R}$
AD_1	15	26	$\overline{S0}$	$\overline{DEN}$
AD_0	16	25	QS0	ALE
NMI	17	24	QS1	$\overline{INTA}$
INTR	18	23	$\overline{TEST}$	
CLK	19	22	READY	
V_{SS} (GND)	20	21	RESET	

This is divided in three modes.

1. Common Mode
2. Maximum Mode
3. Minimum Mode.

COMMON MODE

(1) AD_{15} AD_0 (Address/Data) –

→ These are time multiplexed address and data bus.

→ These lines acts as address bus during the first part of machine cycle. and data bus for the remaining part of the machine cycle.

(2) $A16/S3$ – $A19/S6$ (Address/ status)

→ Status bit S_6 is always zero

→ S_5 gives the current value of the interrupt flag (IF).

S_4	S_3	Register
0	0	ES
0	1	SS
1	0	CS or more
1	1	DS

(3) $\overline{BHE}/S7$ (Bus high Enable/status) –

→ $\overline{BHE}=0$ to indicate the transfer of data over the higher order (D_{15} – D_8) data bus for read, write.

(4) $\overline{RD}$ (Read control) –

This is an active low signal used to indicate that the processor is performing a memory or I/o read operation.

(5) ~~RESET~~ READY –

This is an active high signal used to indicate that the slow devices or memory have completed the data transfer. If this pin is at logic 0 level, the

microprocessor (8086) enters into the wait state and remains idle.

(6) RESET -

This is an active high signal which must be high for at least 4 clock cycles. It clears IP, DS, SS, ES and also resets all the flags and instruction queue.

(7) $\overline{\text{TEST}}$ -

→ This is an active low input signal which is only used by the WAIT instruction.

→ If this pin is low, 8086 enters into the wait state and remain idle while the execution is continue.

(8) NMI - (Non Maskable Interrupt)

→ This is a positive edge triggered non-maskable interrupt request.

→ A transition from low to high on this pin initiates the interrupt response at the end of the current instruction

Min/Max Mode

DT/$\overline{R}$ (Data Transmit/Receive)

→ This is an output signal used to control the direction of data flow.

→ When DT/$\overline{R}$ = 1 it indicates that 8081 is transmitting the data and when DT/$\overline{R}$ = 0 it is receiving the data.

$\overline{DEN}$ (Data enable)

This is an active low signal which informs the transceivers that 8086 is ready to send or receive data.

~~Data~~ $\overline{INTA}$ (Interrupt Acknowledge)

This is an active low output signal which indicates that the 8086 processor has accepted the interrupt when $\overline{INTA}$ = 0.

$\overline{LOCK}$ (Bus priority lock control)

→ This is an active low output signal.

→ When $\overline{LOCK}$ = 0, it indicates that the system bus is not used by any other processor.

$\overline{RQ}$ / $\overline{GT0}$; $\overline{RQ}$ / $\overline{GT1}$

→ This is request and grant pin.

→ 8086 can get two ~~interrupts request~~ request at the same time from external devices but the request can be granted only on priority basis.

→ $\overline{RQ}$/$\overline{GT0}$ has higher priority than $\overline{RQ}$ / $\overline{GT1}$

Status Signals

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	Machine Cycle
0	0	0	Interrupt Acknowledge
0	0	1	I/O Read
0	1	0	I/O Write
0	1	1	HALT
1	0	0	Instruction Fetch
1	0	1	Memory Read
1	1	0	Memory Write
1	1	1	Inactive - Passive

QS0 / QS1

→ Shows the status of Queue.

QS_1	QS_0	Status
0	0	No operation (Queue is idle)
0	1	First byte of opcode is taken from queue
1	0	Queue has been reinitiated (empty queue)
1	1	Subsequent byte of opcode is taken from queue.