Difference b/w 8085 and 8086

8085	8086
→ 8-bit processor	16-bit processor
→ 8-bit data bus	16-bit data bus
→ 16-bit address bus	20-bit address bus
→ Clock speed 3MHz	Clock speed 5 to 10 MHz
→ Doesn't have memory segmentation	4 memory segments
→ no pipelining	pipelining
→ Accumulator based calculations only supported	Calculations based on 4 general purpose registers
→ Only program counter and address counter registers are 16 bits	All registers are 16 bits
→ Supports only single processor system	More than 1 processor system
→ 64 KB internal memory	1 MB internal memory
→ no control flag	16-bit control bit
→ 6500 transistors	29000 transistors
→ no min and max mode	supports min and max mode

(Architecture of 8086 (Software))

1) Code
2) Data
3) Stack
4) Extra

} Segment

We have 4 general purpose registers

AX Accumulator
BX Base
CX Counter
DX Data

We have 2 parts of architecture

EU Execution Unit
BIU Bus Interface Unit

Now,

SP Stack Pointer
BP Base Pointer
SI Source Index Register
DI Destination Index Register

→ Operands and flag (each of 16 bits) help the ALU store the operands

6 byte instruction queue is used for pipelining

Σ → Summation

PA → Physical Address 20 bits
BA → Base Address 16 bits
EA → Effective Address 16 bits

$$PA = (BA \times 10) + EA$$

0700 X 10 (Convert 16 bit → 20 bit)
= 07000 → Summation

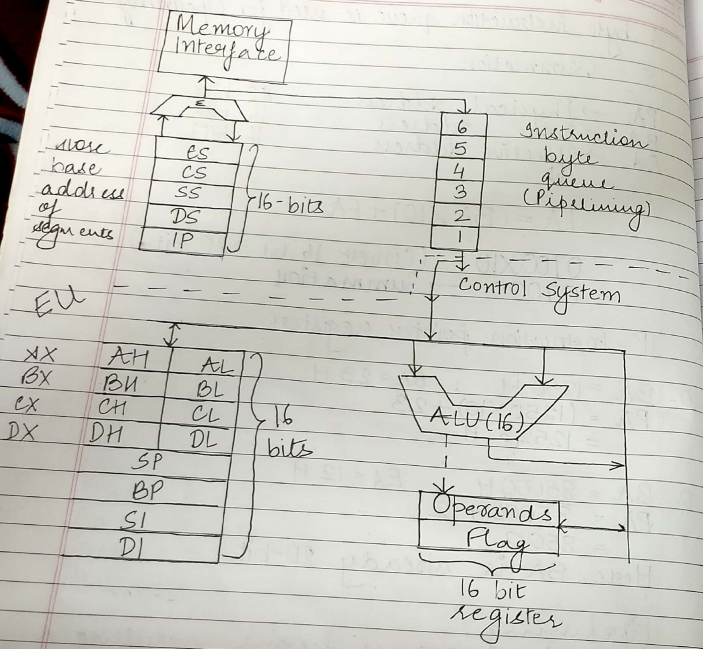
IP – Instruction pointer register

Q- BA = 1250 H ; EA = 23 H
PA = (1250 X 10) + 23
= 12523 H

Q- BA = 85000 H ; EA = 12 H
PA = 85000 + 12
= 85012
Here, BA is already 20-bit

Pipelining:
The microprocessor begins executing a second instruction before the first has been completed
As EU executes instructions, simultaneously BIU fetches next instruction to the queue

BIU



FLAG

8085
No flag register

8086
flag register

U	U	U	U	U	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

We have 6 status flags and 3 control flags
CONTROL FLAGS
IF TF DF
They control the execution of the processor.

0 CF Carry Flag
0 → Reset (By Default)
1 → Set (If carry is generated)

2 PF Parity Flag
PF = 1 for even number of 1's in the carry

4 AF Auxillary Flag
Nibble → 4 bits
AF = 1 in nibble operation
CF = 1 in whole addition

```
  A B C D  H
  0 0 0 5  H
```

↑ ← If carry is generated AF=1

If after the whole opⁿ carry is generated CF=0

6 ZF Zero flag
Result of arithmetic is 0

7 SF Sign flag
1 negative
0 positive
MSB is always stored in SF

8 TF Trap flag
TF = 1
Single step
Result after each instruction (allows debug
TF = 0
Run
Final Result

Its a control flag as the execution of the processor is affected.

9 IF Interrupt flag
IF = 1
Processor will stop its current execution and address the interrupt
IF = 0
Processor not ready to accept interrupts
You can manually set IF = 0 (disable the interrupt)

10 DF Direction flag
DF = 0
Strings are going to execute from higher to lower address

DF = 1
Execute from lower to higher address
eg: Reverse a String
CAT DF = 0
TAC DF = 1

11 OF Overflow flag
It is set if the result of the signed opn is too large to fit in the number of bits available to represent it.

Instruction Queue
Used to queue up the next set of instructions that are to be executed eliminating the time used to fetch these instructions.
6 byte queue → because the longest instruction in instruction set is 6 bytes long
→ While EU is busy in decoding the instruction corresponding to a memory location, the BIU fetches the next six instruction bytes from locations.
→ These instruction bytes are stored in 6 byte queue on FIFO

Segments helps the processor to become faster because then it fetches / looks for the value in a particular segment
Register stores base address of each segment

Address

→ Logical / offset / effective
Contained in 16-bit IP, BP, SP, BX, SI, DI

→ Base segment address
CS, DS, ES, SS

→ Physical address
Real address by combining offset and base segment addresses.
It is of 20 bits.

General Purpose Register
Used to store temporary data within the microprocessor.

1) AX - for arithmetical and logical instruction
2) BX - store the value of the offset.
MOV BL, [500] → BL = 500H
3) CX - Used in looping and rotation.
4) DX - Used in multiplication (I/O port addressing)
5) SP - Points to the topmost item of the stack. If the stack is empty, SP = (FFFE)H.
6) BP - Used in accessing parameters passed by the stack.
7) SI - Used in pointer addressing of data and as a source in some string related operations.
8) DI - Used in pointer addressing of data and as a destination in some string related operations.

Offset address relative
SP → Stack segment
BP → Stack segment
SI → Data segment
DI → Extra segment

Two stages of pipelining
fetch stage can prefetch up to 6 bytes of instructions and stores them in queue
execute stage executes these instructions.

ALU
handles all arithmetic and logical operations.

IP
holds the address of the next instruction to executed by the EU.

EU gives instructions to BIU stating from where to fetch the data and then decode and execute those instructions. It controls operations on data using instruction decoder and ALU.

BIU
Sending addresses, fetching instructions from the memory, reading data from the ports and the memory as well as writing data to the ports and the memory.
EU has no direct connection with system bus so this is possible with the BIU.

ADDRESSING MODES OF 8086

1. Immediate addressing mode

In this 8-bit and 16-bit data is directly given to memory / register

→ MOV AX, 1234H

→ MOV [1234H], 1234H

↑ effective address

↑ data

One memory cell can store only 8 bi of data

Effective address

0700 : 1234 H

↑ base address

Physical address = 08234

08234 will store 34 (Can store 8 bits)
08235 will store 12

2. Register direct addressing mode

In this we are passing register to register.

→ MOV AX, BX

→ MOV AX, 1234 H
MOV BX, AX

3. Direct addressing mode

We are fetching the value from memory.

→ MOV [1234 H], 1234 H
MOV AX, [1234 H]

effective address

Thus, AX will store 1234 H

Question:
Swap the values of two registers using memory location.

MOV AX, 1211 H
MOV BX, 1230 H
} Initialize AX, BX
MOV [1234 H], AX
MOV AX, BX
MOV BX, [1234 H]

e: MOV destination, source

4. Register Indirect addressing mode

4 registers

SI	Source Index
DI	Destination Index
BP	Base Pointer
BX	Base Register

Only these registers can store memory address.

→ MOV BX, 1234 H
MOV [BX], 1234 H

Ques: Store the 16-bit value at effective address 2500 using register indirect addressing mode.

MOV BX, 2500 H
MOV [BX], 1210 H

↑ effective address

↑ 16-bit data

note:

MOV BX, 2500 H

↑ Base Register

↑ address

Register Relative addressing mode

→ MOV BX, 1234 H
MOV AX, [BX + 8/16-bit]

We can use only SI, DI, BP, BX

Base Index Addressing Mode

Effective Address = [BX/BP] + [SI/DI]

Relative Base Index Addressing Mode

Base + Index + 8/16-bit

8086 Processor

→ 8086 does not have its internal clock

→ T1 T2 T3 T4 clock cycles

Pins carry addresses

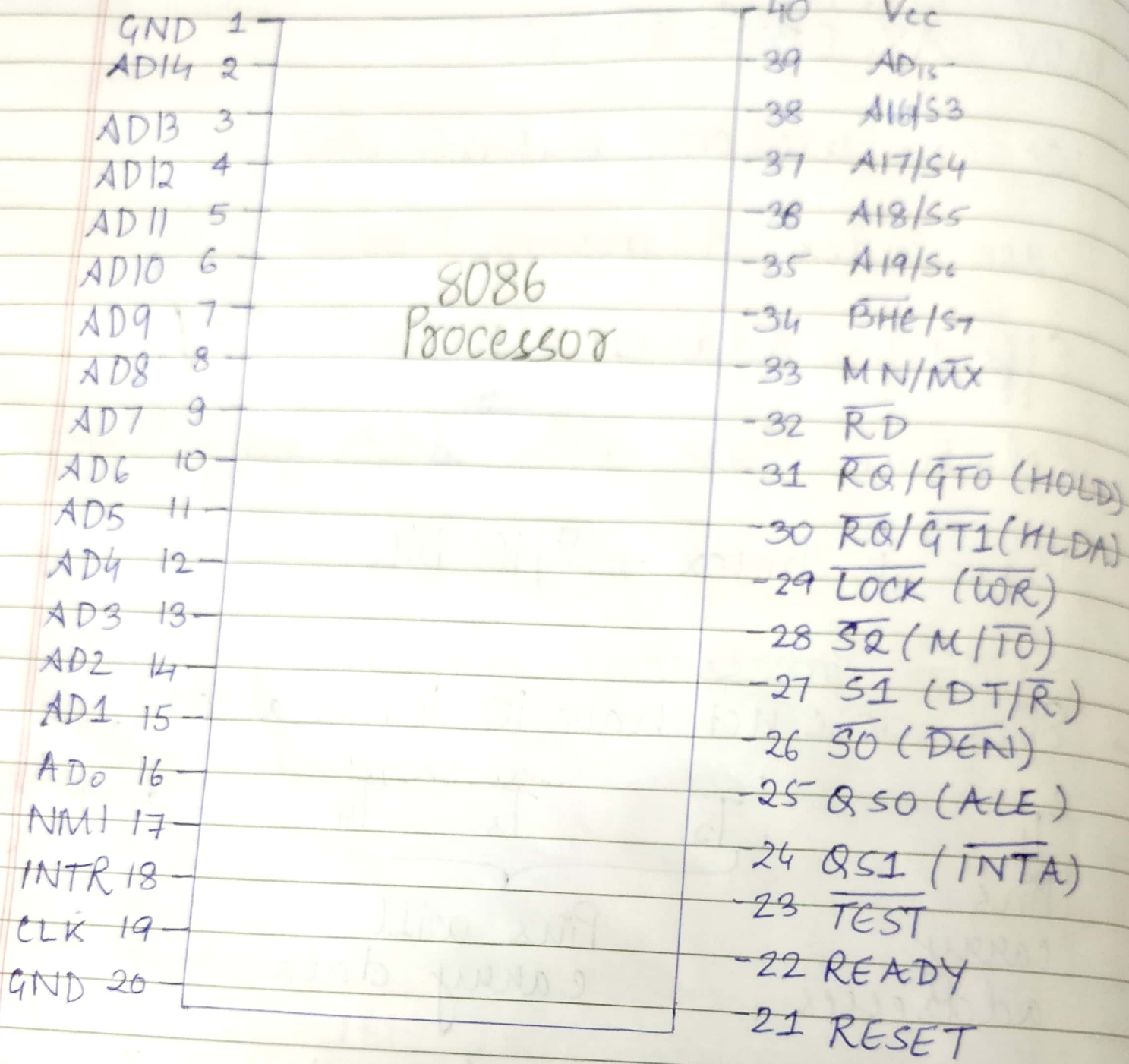
Pins will carry data

→ First the address is fetched and then the data at that address is fetched.

→ 8086 has an external clock connected.

→ 1 machine cycle - Even memory address
2 machine cycle - Odd memory address

PIN DIAGRAM OF 8086



→ Address / Status Pins
38 - 35

→ When S_3 and S_4 are 0 we have ES

→

S_4	S_3	Characteristic
0	0	Extra Segment access
0	1	Stack segment access
1	0	Code segment access or none
1	1	Data segment access

S_6 always LOW
S_5 = IF (Interrupt flag)

→ BHE → Bus High Enable Pin (Pin 34)

→

$\overline{BHE}$	A_0	Characteristic
0	0	Whole word (16-bit transfer)
0	1	Upper byte transfer from/to odd address
1	0	Lower byte transfer from/to even address
1	1	None

→ When user gives the WAIT instruction, the processor checks the status of TEST Pin (23)

→ Single Processor - Min mode
Multiple Processor - Max mode - Master slave Architecture - Bus controller IC 8288

AD15-0
When the ALE pin is 1, these pins carry the address and when it is 0, they carry data.

multiplexed address data bus
Same bus is used to carry data as well as address.
Multiplexing is used to refer to a process where multiple analog message signals are combined into one signal over a shared medium.
Address bus gives the memory instructions on where to place the actual data that it will store.
Data bus carries the information that is going to be stored using the location that the address bus gave to memory.

NOTE:
→ 40 pins
→ 5V DC supply
→ 20-line address bus (operate in multiplexed mode)
→ 16-line data bus
→ The 16-low order address bus lines have been multiplexed with data
→ The 4-high order address bus lines have been multiplexed with status signals.

A19/S6 - A16/S3
Multiplexed to provide address signals A19-A16 and status bits S6-S3.
ALE = 1 → address
ALE = 0 → status

Clock signal
Provides timing to the processor for operations.

BHE Bus High Enable
Used to indicate the transfer of data using data bus D8-D15.

Read $\overline{RD}$
Whenever it is at logic 0, the 8086 reads the data from the memory or I/O device through the data bus.

Ready
When it is high, it indicates that the device is ready to transfer data.
When it is low, it indicates wait state.

RESET
Used to restart the execution. It is held at logic 1 for a minimum of 4 clocking periods
CS → FFFFH, IP → 0000H, other registers → 0000H
8086 begins executing instructions from memory address FFFF0H.

INTR
IF = 1, INTR is held high (logic 1)
8086 gets interrupted
IF = 0, INTR is disabled.

NMI Non-maskable interrupt
It is an edge triggered input which causes an interrupt request to the microprocessor.
Edge triggering
→ Positive
It will take input at exactly the time in which the clock signal goes from low to high.
→ Negative
clock signal goes from high to low.

An edge triggered changes states either at positive or negative edge of the clock pulse

$\overline{TEST}$
It is like wait state.
$\overline{TEST}$ = 0, the WAIT instruction functions as a NOP (no instruction). If it is 1, the WAIT instruction wait for the $\overline{TEST}$ = 0.

STACK INSTRUCTIONS

PUSH S
↑ Source
16 bit

top higher address
to lower address

Destination → stack segment
SP decrements by 2
One memory cell can store only 8 bits of data, thus, 2 memory locations are required as the source is of 16 bits.
Source can't be AL, AH, BL, etc as they are of 8 bits.

POP D
↑ Destination (can't be immediate data)
(16-bit)
Source → top of stack
SP incremented by 2

PUSHF
16 bit flag value → top of the stack
↑ Source ↑ Destination
SP decrements by 2

POPF
We can initialize any flags value. Suppose we have to set IF=1 and TF=1. We use this instruction

Q- Initialize different flags using 8086 instruction
Set TF, DF, PF
Reset CF, ZF
where Reset = 0 and Set = 1

U	U	U	U	O	D	I	T	S	Z	U	A	U	P	U	C
0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0

0 5 0 4 → hexadecimal

```
PUSH 0504H
POPF
```

or

```
MOV AX, 0504H
PUSH AX
POPF
```

move this value in the register

INSTRUCTION SET OF 8086

- → Data Transfer
- → Stack instruction
- → I/O
- → Arithmetic
- → Logical
- → flag control
- → String
- → Branch

DATA TRANSFER INSTRUCTIONS

1) MOV Destination, source

2) XCHG Destination, source → Swaps direct
   ↓
   Exchange

3) LAHF
   Load AH register from flag register.
   8 LSB bits will store to AH register.
   AH → 8 bits and flag → 16 bits

4) SAHF
   Store AH into flag register
   Transfer AH into LSB of CF.

5) MOV AH, 41H → Control word
   SAHF
   To make CF and ZF = 1

5) LDS R, M
R → Register M → Memory location
lower 16-bit data will go in R
higher 16-bit data will go in DS
DS Data segment → To store data segment of base address
M → 32 bits

6) LES R, M
extra segment
higher 16 bit → ES

7) XLAT
(Translate)

8) LEA R, EA
↳ Effective address
load
effective address

e.g. LEA DX, [SI + 86 H]
stores the EA in DX
MOV DX, [SI+86 H]
moves the data of the address in DX

INPUT / OUTPUT INSTRUCTIONS

8086 has no port to connect peripheral devices.

→ IN
→ OUT

We need to connect 8086 to 8255.

IN AL/AX, 8 bit port address / DX
Transfers content of port (register) address into AL/AX

IN AL, 45H
Source → port address
No immediate data / memory location

For 16-bit, move it into DX register.
IN AL, 1234H X
We can't give 16-bit directly.

OUT 8-bit port address / DX, AL/AX

Q- Transfer content stored in 2500H memory location to port address 5500H

MOV DX, 2500H
IN AX, DX
MOV DX, 5500H
OUT DX, AX
	X

MOV AX, [2500H]
MOV DX, 5500H
OUT DX, AX
	✓

ARITHMETIC INSTRUCTIONS

ADD D, S
D+S → D
Both should be of same no. of bits
D can't be immediate data
Operand will store the result of D+S
The carry generated will not be shown in the register but CF=1.

ADC D, S
D+S → Result +CF → D
It will take carry in account

SUB
SBB

Q- Add two 32-bit numbers stored in register pair AX, DX and SI, DI. Store the result into memory location 7100 H.

DAA Decimal Adjustment after Addition

Add 2 BCD valid numbers 5 and 6 → 11 which is not a BCD valid number

Processor will consider 11 as B (hexadecimal)

DAA will understand AL (8-bits)

```
ADD BX, CX
MOV AL, BL      ← Store the 8 bits LS
                  of BX to AL.
```

4 LSB of result > 9 or AF=1 then DAA will add 6 (decimal number)

e.g. 10 > 9

10 + 6 = 16

1 carry

0 result → valid BCD

DAS

Decimal Adjustment after subtraction

CMP D, S

Compare

D - S → here the result is not be stored flags value is updated

If D = S , ZF = 1

D > S , ZF = 0, CF = 0 } Result is positive

D < S , ZF = 0, CF = 1 } Result is negative

↳ As borrow is generated

INC D
↳ Register / memory location
Increment by 1

DEC D
Decrement by 1

AX → 1234 H
DEC AX → AX 1233H

NEG D
Negative
2's complement

C W
CWD

MUL S
↑
any register except AL (8-bit)
S → any register / memory except AL

WAP to multiply 2 8-bit data and store in 7500H memory location.

AL X S → AX
↑ ↑ ↑
8-bits 8-bits 16-bits
↑ ↑ any ↑ By default
By default register/ memory location

```
MOV  AL, 12H
MOV  BL, 34H
MUL  BL
MOV  [7500H], AX
```

When S = 16 bits

AX → One (default) number

S → Any register / memory location / data

```
  AX
X  S
-----
DX-AX
```

→ register pair will store the 32-bits

AX → LSB of the result (16-bits)

DX → MSB of the result (16-bits)

Solution :

```
MOV  AX, 1234H
MOV  BX, 2500H
MUL  BX
MOV  [7500H], AX
MOV  [7502H], DX
```

We can use MUL AX.

It will multiply AX x AX → AX

DIV S

→ If S → 8 bit then dividend → 16 bit
AX / S → AX

AH AL
Remainder Quotient

→ If S → 16 bit then dividend → 32 bits
(AX-DX pair) → dividend
(DX-AX) / S → DX - Remainder
AX - Quotient

LOGICAL INSTRUCTIONS

AND
OR
X-OR
Invert
Rotation
Shift

Bit by bit execution → Logical

AND D, S

both should be same

0 & 1 → 0
1 & 0 → 0
0 & 0 → 0
1 & 1 → 1

```
11011000
11010010
11010000
```

D 0 → Store in the destination

If the result is 0, ZF=1

Bit by bit AND

X-OR

```
1101
1111
0010  → complement (invert)
```

```
1101
0000
1101  → same number
```

Q. Test the value of D3 bit of CH register without changing other bits.

TEST D, S

Bit by Bit AND but the result is not stored in destination.

TEST CH, control word

↑ 8-bit

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	1	0	0	0
		0				8	

If D3 = 0
and D3 AND 1 = 0 ZF = 1

D3 = 1
1 AND 1 = 1 ZF = 0, PF = 0

note: When ZF=1 the whole result = 0 signifying D3 being 0.

NOT D
1 (NOT) → 0 → store in D itself
NOT 0 → 1

CLC
Clear Carry flag
It clears the CF (CF = 0) and does not affect any other flags.

CLD
Clear Direction flag
It clears the DF (DF = 0)
It does not affect any other flags