Chapter 3:
INTEL 8086

Syed Shakil Mahmud

Lecturer

Department of Computer Science and Engineering

BAIUST, Cumilla

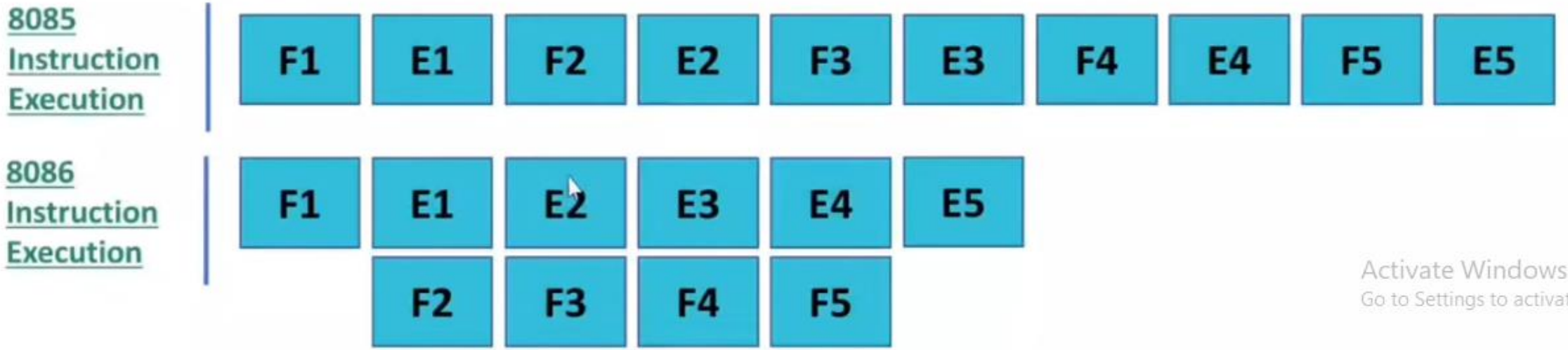
Main Features of 8086

1. 8086 is a 16-bit microprocessor.
2. 8086 has a 16-bit data bus.
3. 8086 has a 20-bit address bus which means it can address up to 2∧20= 1MB memory location
4. Frequency range of the 8086 is 6-10 MHz.
5. The Intel 8086 is designed to operate in two modes, namely the minimum mode and the maximum mode.
6. The 8086 works in a multiprocessor environment.
7. It uses two stages of pipelining, i.e. Fetch Stage and Execute Stage, which improves performance.
8. It fetches up to six instruction bytes (4 instruction bytes for 8088) from memory and queue stores them in order to speed up instruction execution.
9. It requires +5 V power supply.
10. It uses a 40-pin dual in line package.
11. It has 256 interrupts.

Pipelining in Microprocessor 8086

❖ How to increase Speed of execution of programs in microprocessor :

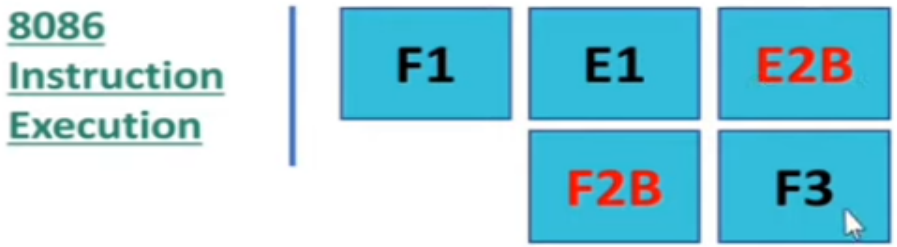
- ❑ Increasing data bus. [So, in single machine cycle we can transfer more data]
- ❑ Increasing System Clock. [So, more cycles can get executed in given time]
- ❑ Upgrading ALU. [Simplest explanation is 16 bits ALU performs more arithmetic and logical operations compared to 8 bits of ALU]
- ❑ Upgrading Memory technology. [Like now a days (2022), we have SSD & Cache in our computer]
- ❑ Upgrading Instruction Set. [Like in 8086 we have Multiplication and Division Instructions available which wasn't available with 8085]
- ❑ Pipelining [It helps to execute multiple instructions in parallel.]



Pipelining in Microprocessor 8086

❖ Issue of Pipelining in Microprocessor 8086 :

- ❑ In branch instruction execution, next fetch instruction will be discarded.



Reference for Pipelining: https://www.youtube.com/watch?v=02SNMOHRIvA&list=PLgwJf8NK-2e7oIAY26zml0DSE5f4r_Ei7&index=2

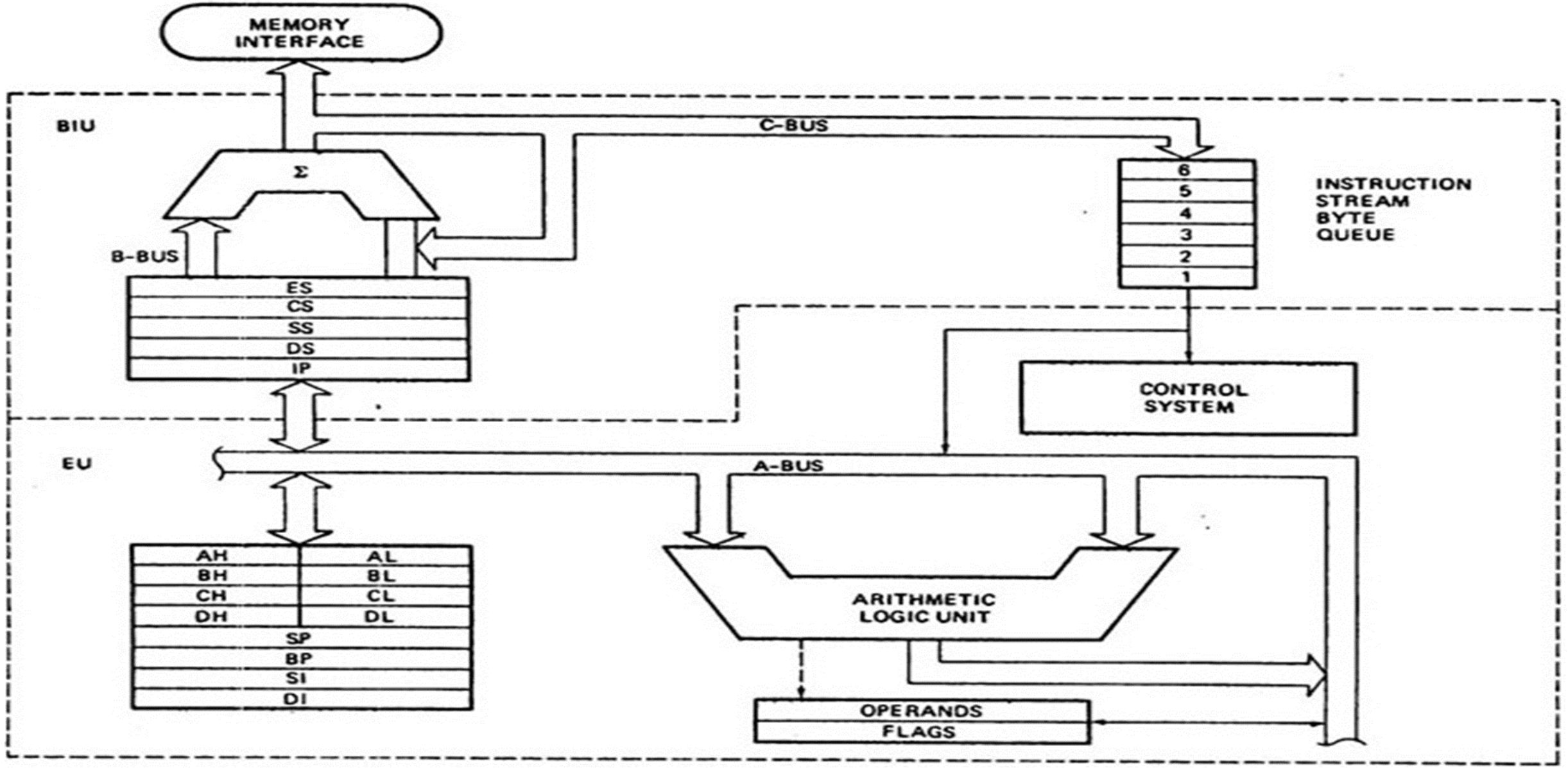
Pipelining in Microprocessor 8086

❖ Issue of Pipelining in Microprocessor 8086 :

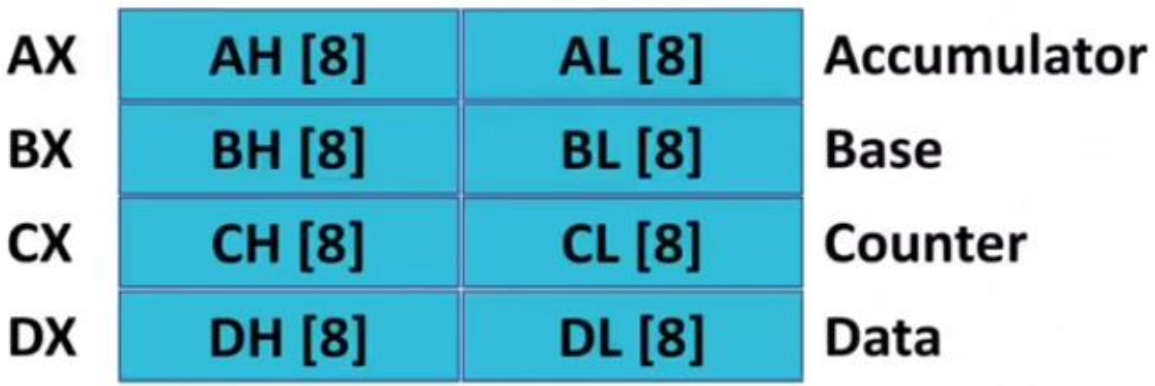
- ❑ In branch instruction execution, next fetch instruction will be discarded.

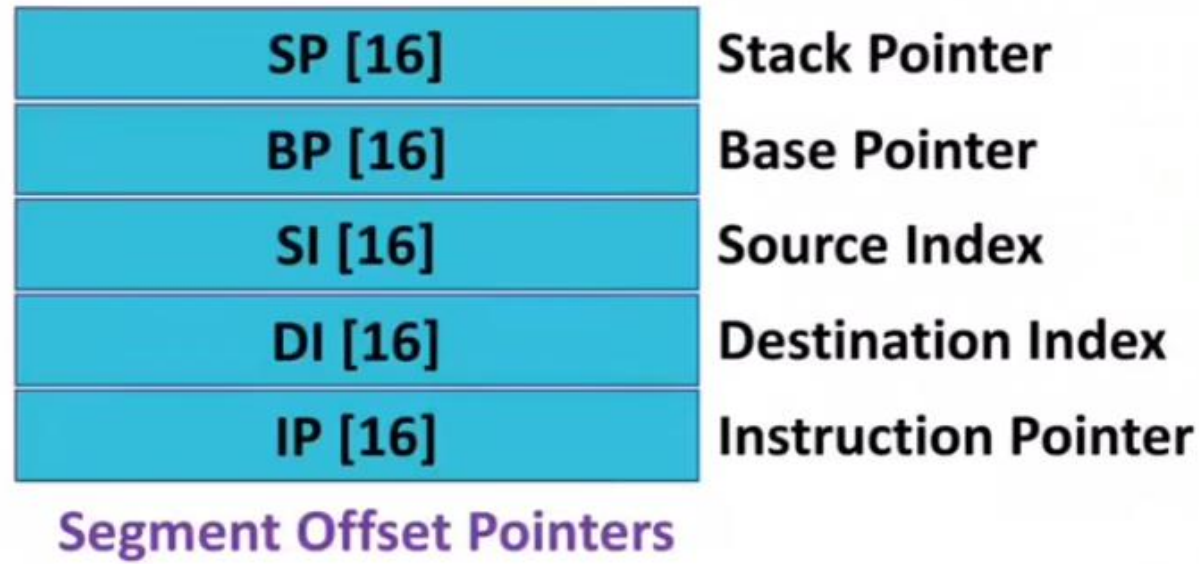
Reference for Pipelining: https://www.youtube.com/watch?v=02SNMOHRIvA&list=PLgwJf8NK-2e7oIAY26zml0DSE5f4r_Ei7&index=2

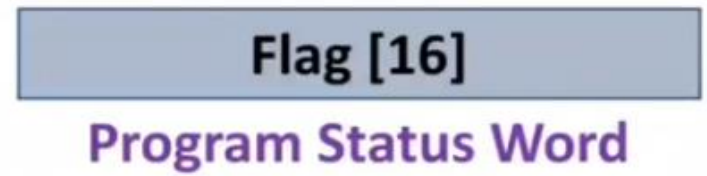
Internal Architecture of Intel 8086



Programming Model of 8086







Intel 8086 Microprocessor

8086 microprocessor has two units.

- Bus Interface Unit (BIU).
- Execution Unit (EU)

They are dependent and get worked by each other.

Bus Interface Unit(BIU)

BIU performs the following functions-

- It generates the 20 bit physical address for memory access.
- It fetches instruction from memory.
- It transfers data to and from the memory and I/O.
- It supports pipelining using the 6 byte instruction queue

Execution Unit (EU)

Execution Unit (EU) performs the following functions-

- It fetches instructions from the Queue in BIU, decodes and executes them.
- It performs arithmetic, logic and internal data transfer operations within the microprocessor.
- It sends request signals to the BIU to access the external module.

Main Components of Bus Interface Unit (BIU)

The main components of the BIU are as follows:

- 4 Segment Registers
- Instruction Pointer
- Address Generation Circuit and
- A prefetch queue

Main Components of BIU (Cont.)

Segment registers-

- **CS register:** CS holds the base address for the Code Segment. All programs are stored in the Code Segment. CS is multiplied by 10H to give the 20 bit physical address of the Code Segment. E.g. If CS = 4321H then CS x 10H = 43210H→ Starting address of Code Segment.
- **DS register:** DS holds the base address for the Data Segment. It is multiplied by 10H to give the 20 bit physical address of the Data Segment. E.g. If DS = 4321H then DS x 10H = 43210H→ Starting address of Data Segment.
- **SS register:** SS holds the base address for the Stack Segment. It is multiplied by 10H to give the 20 bit physical address of the Stack Segment. E.g. If SS = 4321H then SS x 10H = 43210H→ Starting address of Stack Segment.
- **ES register:** ES holds the base address for the Extra Segment. It is multiplied by 10H to give the 20 bit physical address of the Extra Segment. E.g. If ES = 4321H then ES x 10H = 43210H→ Starting address of Code Segment.

Main Components of BIU (Cont.)

Instruction Pointer (IP)-

- It is a 16 bit register. It holds offset of the next instructions in the Code Segment.
- Address of the next instruction is calculated as CS x 10H + IP.
- IP is incremented after every instruction byte is fetched.
- IP gets a new value whenever a branch occurs

Main Components of BIU (Cont.)

Address Generation Circuit-

The BIU has a Physical Address Generation Circuit. It generates the 20 bit physical address using Segment and Offset addresses using the formula: Physical Address = Segment Address x 10H + Offset Address

Main Components of BIU (Cont.)

6 Byte Pre-fetch Queue:

- It is a 6 byte queue (FIFO).
- Fetching the next instruction (by BIU from CS) while executing the current instruction is called pipelining.
- Gets flushed whenever a branch instruction occurs.

Main Components of Execution Unit (EU)

The main components of the EU are as follows:

- **General purpose registers**-
- 8086 microprocessor has four 16 bit general purpose registers AX, BX, CX and DX. These are available to the programmer for storing values during programs. Each of these can be divided into two 8 bit registers such as AH, Al; BH, BL; etc. Beside their general use, these registers also have some specific functions.
  - AX register (16 bits): It holds operands and results during multiplication and division operations. All I/O data transfers using IN and OUT instructions use A register (AL/AH or AX). It functions as accumulator during string operations.
  - BX register (16 bits): It holds the memory address (offset address) in indirect addressing modes.
  - CX register (16 bits): It holds count for instructions like loop, rotate, shift and string operations.
  - DX register (16 bits): It is used with AX to hold 32 bit values during multiplication and division. It is used to hold the address of the I/O port in indirect I/O addressing mode.

Main Components of EU (Cont.)

Special purpose registers-

- **Stack Pointer (SP 16 bits):** It holds offset address of the top of the Stack. Stack is a set of memory locations operating in LIFO manner. Stack is present in the memory in Stack Segment. It is used during instructions like PUSH, POP, CALL, RET etc.
- **Base Pointer (BP 16 bits):** BP can hold offset address of any location in the stack segment. It is used to access random locations of the stack.
- **Source Index (SI 16 bits):** It is normally used to hold the offset address for Data Segment but can also be used for other segments using Segment Overriding. It holds offset address of source data in Data Segment during string operations.
- **Destination Index (DI 16 bits):** It is normally used to hold the offset address for Extra Segment but can also be used for other segments using Segment Overriding. It holds offset address of destination in Extra Segment during string operations.

Main Components of EU (Cont.)

- **ALU (Arithmetic Logic Unit)** - It has a 16 bit ALU. It performs 8 and 16 bit arithmetic and logic operations.
- **Operand register-** It is a 16 bit register used by the control register to hold the operands temporarily. It is not available to the programmer.

Main Components of EU (Cont.)

- **Instruction Register and Instruction Decoder-** The EU fetches an opcode from the queue into the instruction register. The instruction decoder decodes it and sends the information to the control circuit for execution.

Memory Segmentation of 8086

- 8086 has 20 bits of physical address.
- Here, 20 bits of physical address is not byte compatible physical address.
- To avoid the issue of byte compatibility, Microprocessor 8086 uses memory segmentation. In which we uses concept of virtual address.
- Here, memory is bisected into FOUR segments.
  - Code Segment
  - Stack Segment
  - Data Segment
  - Extra Segment
- To calculate physical address from virtual address we should know Segment address and OFFSET address.
- Segment Address is held by segment registers (16bits). [CS, SS, DS & ES]
- OFFSET Address is held by OFFSET pointers (16 bits). [IP, (SP & BP), SI & DI]
- So, Physical Address is calculated by

$$PA = SR \times 10H + OP$$

- As Offset Pointer is having size of 16bits, maximum size of any segment is $2^{16} = 2^6 \times 2^{10} = 64K$.
- Minimum size of any segment is 10H = 16Bytes

Physical Address Calculation in 8086

- SP (Stack Pointer) – SP points top of stack segment.
- BP (Base Pointer) – BP points any address of stack segment.
- Physical Address of 8086 is given by

$$PA = SR \times 10H + OP$$

- Example 1 : If Code segment register is 1245H and Instruction Pointer is 1561H. Then finds the physical address of given instruction.

$$PA = SR \times 10H + OP$$
$$PA = 1245 \times 10H + 1561H = 12450H + 1561H$$
$$PA = 139B1H$$

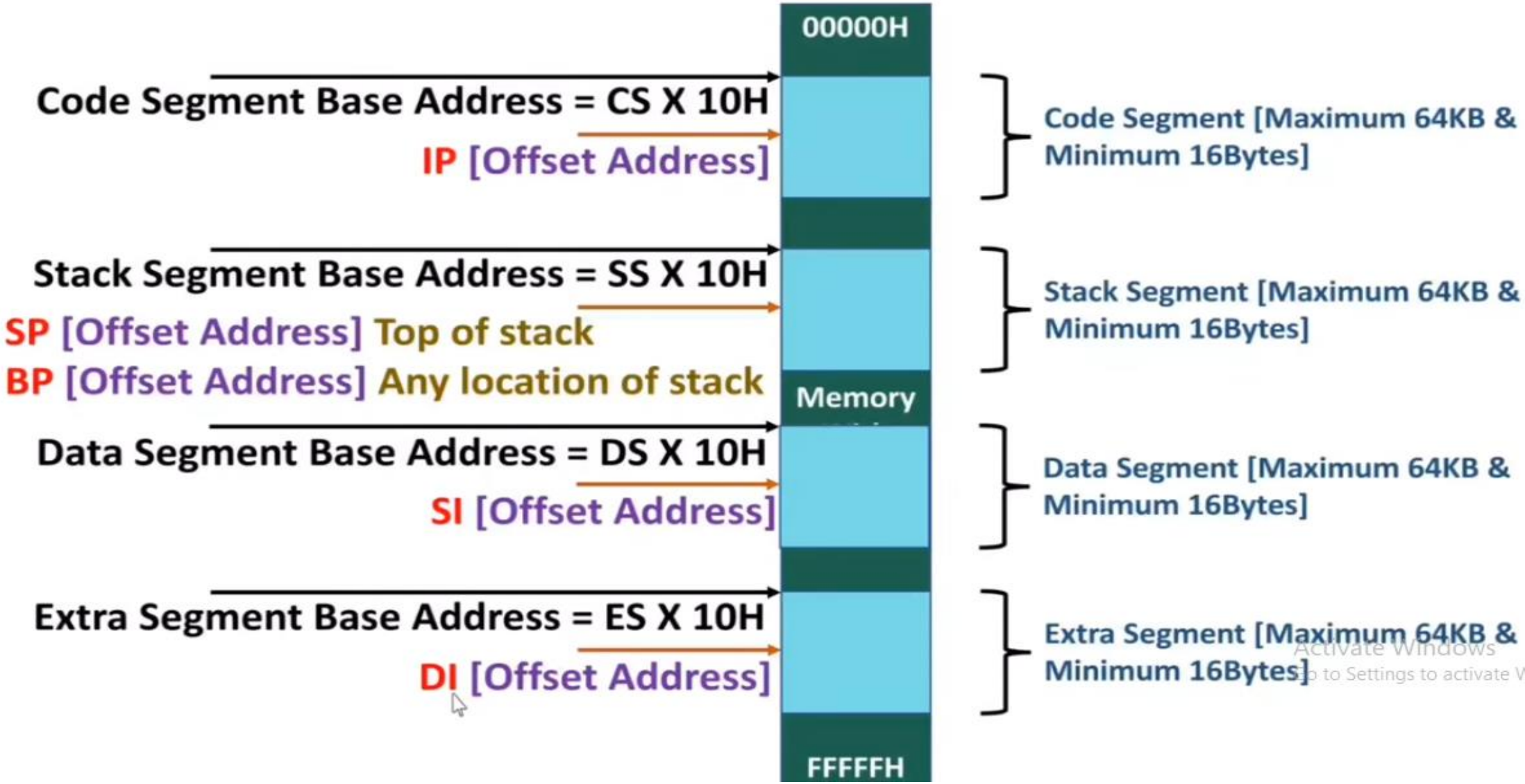
- Example 2 : If Data segment register is 78EFH. Then finds the maximum and minimum range of physical address.
  - For Maximum range

$$PA = SR \times 10H + OP$$
$$PA = 78EFH \times 10H + (0000H\ to\ FFFFH)$$ 64kb
$$PA = 78EF0H\ to\ 88EEFH$$

  - For Minimum range

$$PA = SR \times 10H + OP$$
$$PA = 78EFH \times 10H + 0000H\ to\ 000FH$$ 16 Bytes
$$PA = 78EF0H\ to\ 78EFFH$$

Memory Segmentation of 8086



Advantages of Memory Segmentation of 8086

- We can program 8086 with 16bits addressing.
- Due to Memory segmentation, 8086 becomes backward compatible with 8085.
- Change in One segment does not effect the other segment.
- Memory management gets easier.
- Debugging gets easier.
- It avoids memory overlap with different segments.

Offset for the Specific Segment

Segment	Offset Registers	Function
CS	IP	Address of the next instruction
DS	BX, DI, SI	Address of data
SS	SP, BP	Address in the stack
ES	BX, DI, SI	Address of destination data (for string operations)

Question

The contents of the following registers are

- CS=1111 H
- DS=3333 H
- SS=2526 H
- IP=1232 H
- SP=1100 H
- DI=0020 H

Calculate the corresponding physical addresses for the address bytes in CS, DS and SS

Solution

1. CS = 1111 H

   The base address of the code segment is 11110 H.

   Effective address of memory is given by 11110H + 1232H = 12342H.

2. DS = 3333 H

   The base address of the data segment is 33330 H.

   Effective address of memory is given by 33330H + 0020H = 33350H.

3. SS = 2526 H

   The base address of the stack segment is 25260 H.

   Effective address of memory is given by 25260H + 1100H = 26350H

Main Components of EU (Cont.)

Flag register (16 bits)-

- It has 9 flags.
- These flags are of two types: 6 Status flags namely carry flag, parity flag, auxiliary carry flag, zero flag, sign flag, overflow flag and 3 Control flags namely trap flag, interrupt flag and direction flag.
- Status flags are affected by the ALU after every arithmetic or logic operation. They give the status of the current result.
- The Control flags are used to control certain operations. They are changed by the programmer.

Flag Register of 8086

Control Flags

Overflow Flag
1 = Overflow Occurred
0 = No Overflow

Direction Flag
1 = Auto Decrement
0 = Auto Increment
#Used with in string Instructions

Interrupt Flag
1 = Enable Interrupt
0 = Disable Interrupt

Trap Flag
1 = Perform Single Step
0 = Do Not perform Single Step

Sign Flag
1 = MSB of result is 1 [-ve Number]
0 = MSB of result is 0
#Used with signed operation

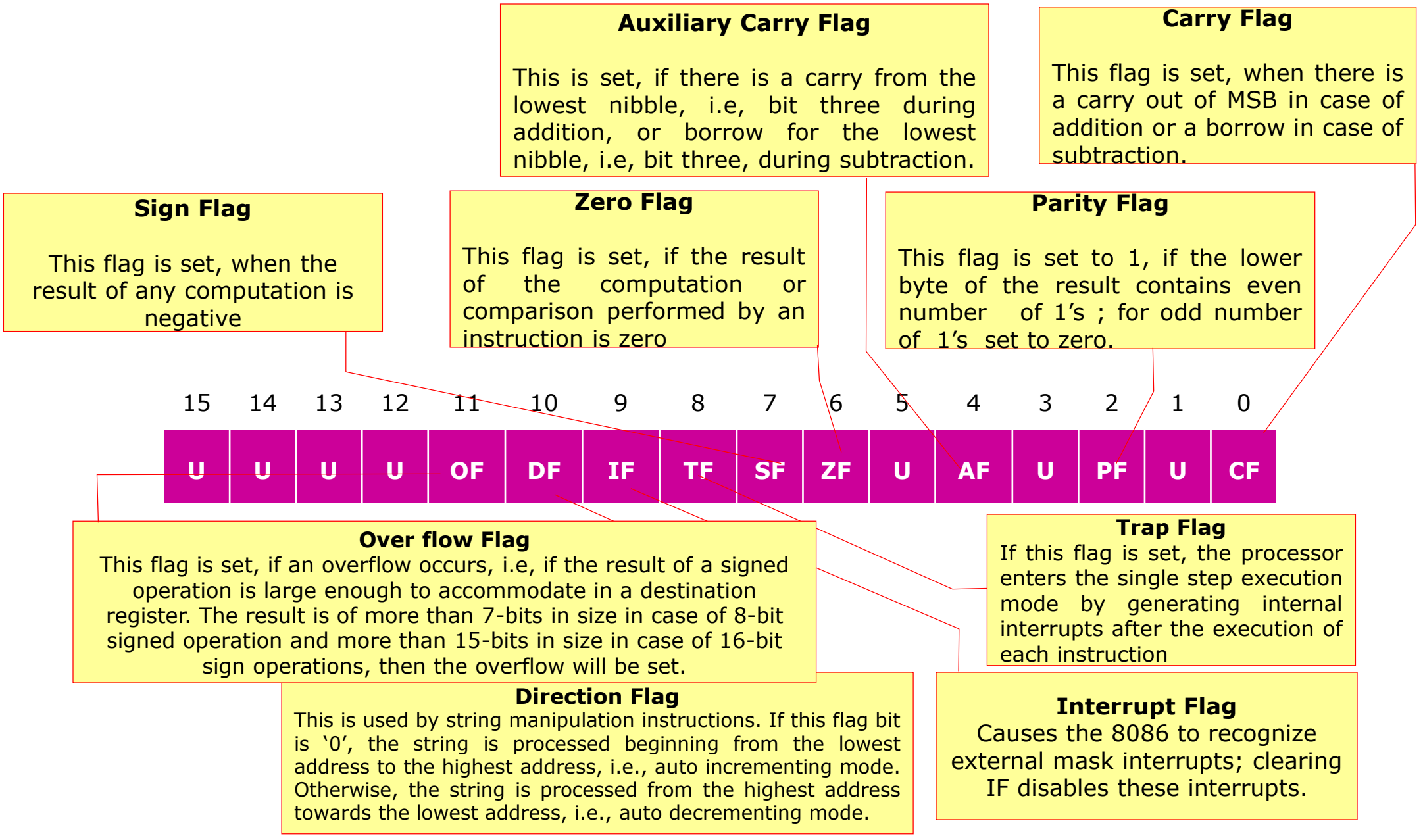
Zero Flag
1 = Result is Zero
0 = Result is not Zero

Auxiliary Carry Flag
1 = Carry from lower to higher Nibble
0 = No Auxiliary Carry
#Applicable in 8 bits Instructions

Parity Flag
1 = Even Parity
0 = Odd Parity

Carry Flag
0 = No Carry
1 = Carry from MSB

Flag Register



Flag Register Calculation

- Example:

1. Identify flag registers for the following operations of Hexadecimal numbers:
    - i. 41+4F
    - ii. 37+01
    - iii. 9E+D3

Flag Register Calculation (Solution)

- Example:

1. Flag registers for the following operations of Hexadecimal numbers:
   - i. 41+4F (CF=0, SF=1, PF=1, ZF=0, OF=1, AF=1)
   - ii. 37+01 (CF=1, SF=0, PF=0, ZF=0, OF=0, AF=0)
   - iii. 9E+D3 (CF=1, SF=0, PF=1, ZF=0, OF=1, AF=1)

OF=1, when $\begin{cases} Pos + Pos = Neg \\ Neg + Neg = Pos \\ MSB\ in\ \neq MSB\ out \end{cases}$

MSB in ≠ MSB out

0 ≠ 1

```
  0100 0001
+ 0100 1111
-------------
  1001 0000
```

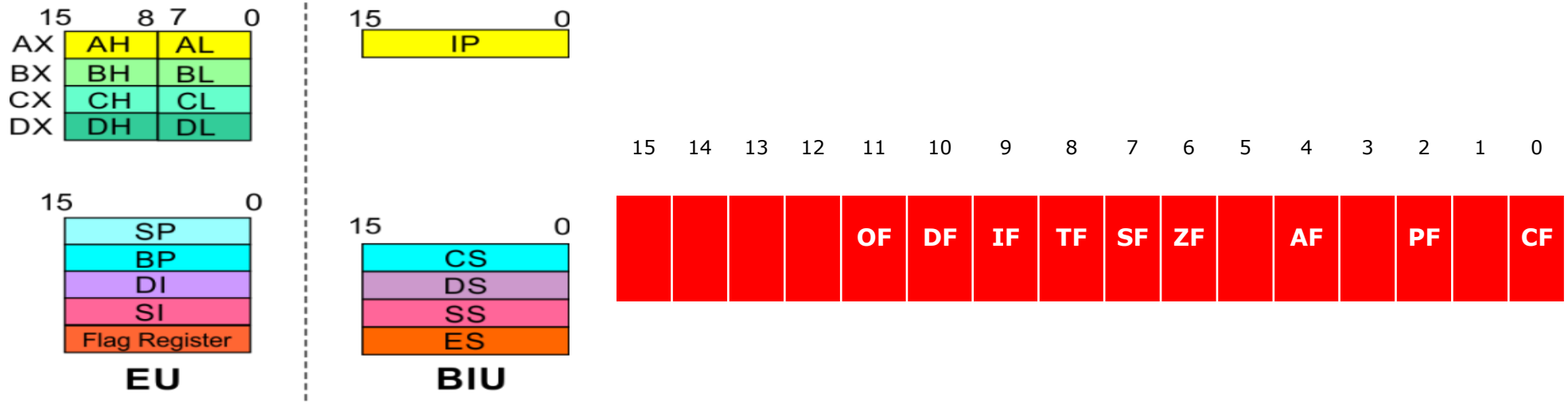
MSB in = MSB out

1 = 1

```
  0011 0111
+ 1101 0001
-------------
1 0000 1000
```

8086 Microprocessor Registers

8086 registers categorized into groups

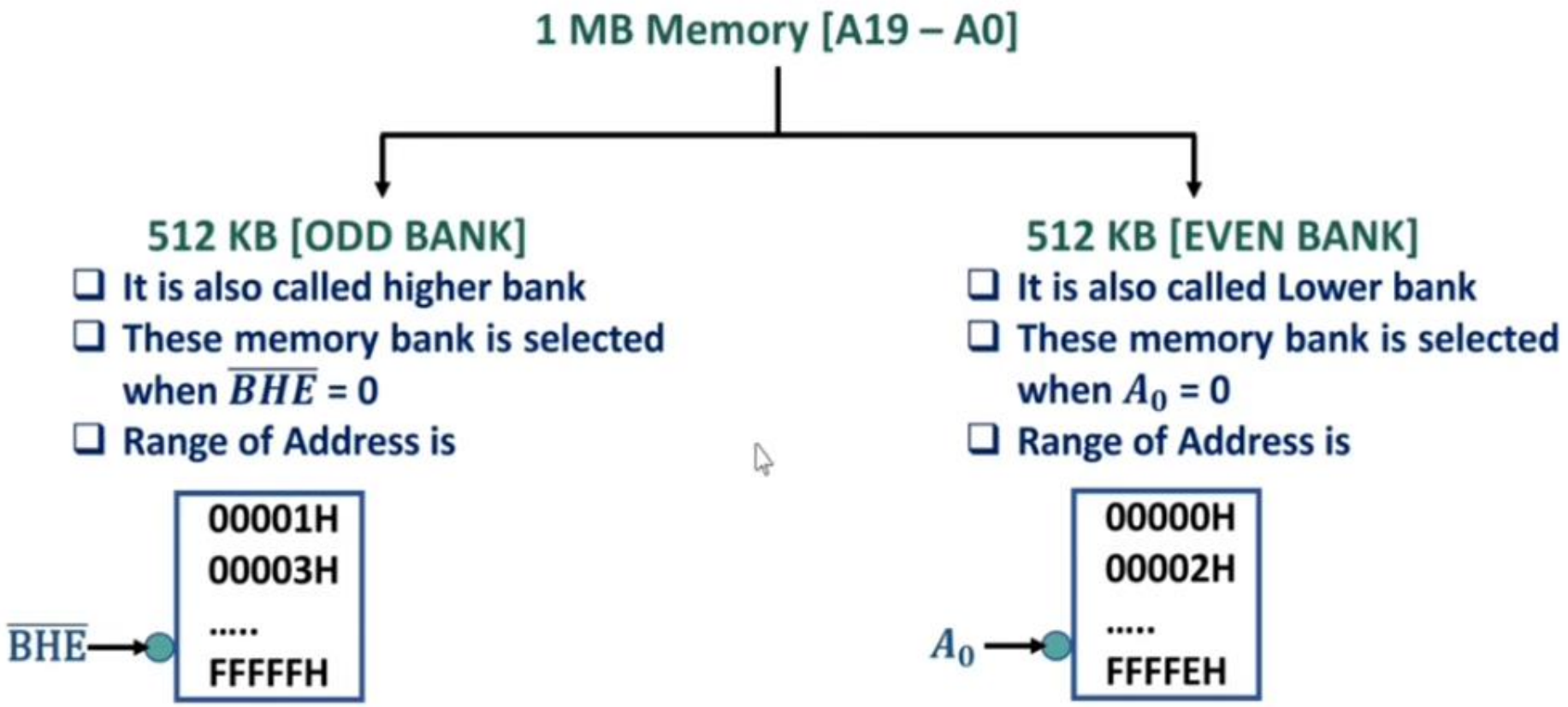


Sl.No.	Type	Register width	Name of register
1	General purpose register	16 bit	AX, BX, CX, DX
		8 bit	AL, AH, BL, BH, CL, CH, DL, DH
2	Pointer register	16 bit	SP, BP
3	Index register	16 bit	SI, DI
4	Instruction Pointer	16 bit	IP
5	Segment register	16 bit	CS, DS, SS, ES
6	Flag (PSW)	16 bit	Flag register

Memory Banking in 8086

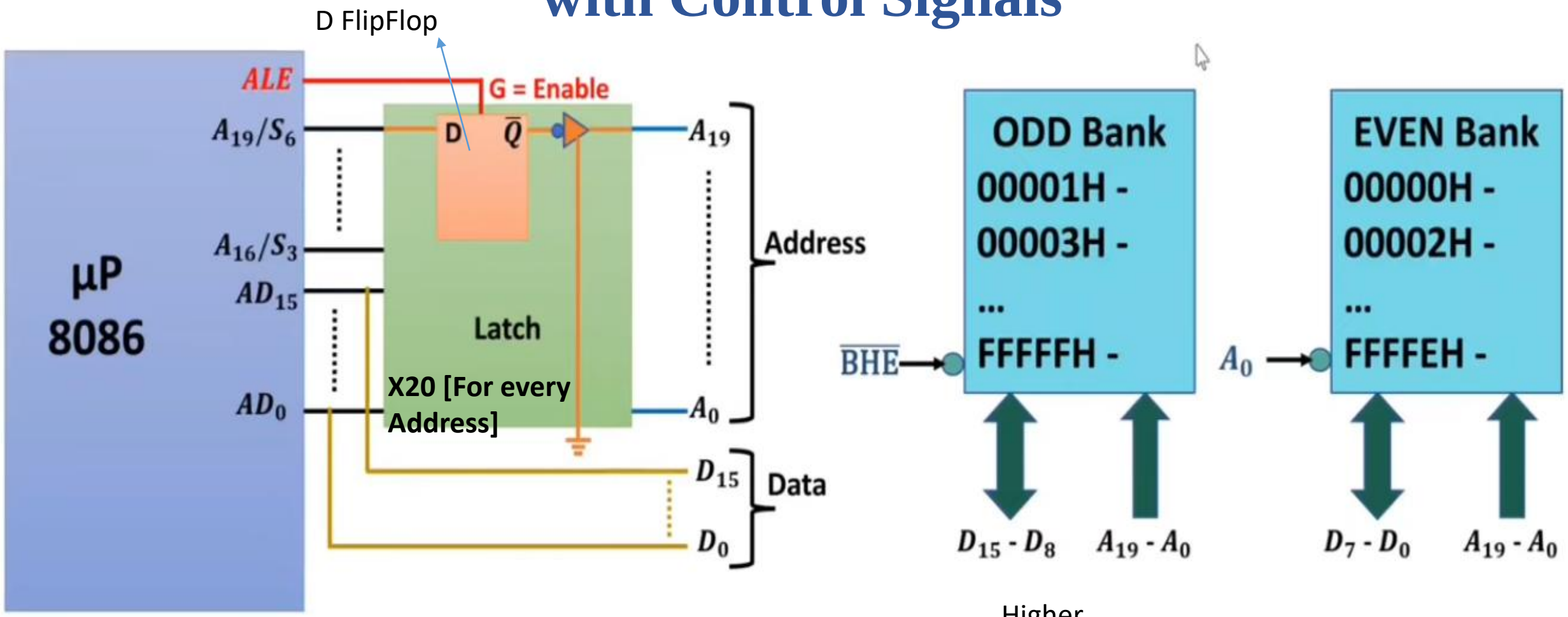
- 8086 has 16 bits of Data lines, so it should access 16 bits (2 Bytes) in one machine cycle.
- To perform 2 Bytes data transfer, So it needs to read 2 memory locations as one memory location has only one byte of data.
- If both of these memory locations are there in same memory then it can not be accessed by microprocessor.
- So, Microprocessor 8086 bisects the memory into two memory banks and each memory bank provides memory of one byte.
- The division is done in such a manner that any two consecutive memory locations lie in two memory chip. So, each memory chip holds alternate memory locations.
- Hence, One memory bank have all even address (Even Memory Bank) and other memory bank have all odd address (Odd Memory Bank).
- **Generally,** for any 16 bits operation, Even Memory Bank provides the lower Byte (Lower Bank) and Odd Memory Bank provides higher byte (Higher Bank). **So we should write program from Even Memory locations.**

Memory Banking with Control Signals



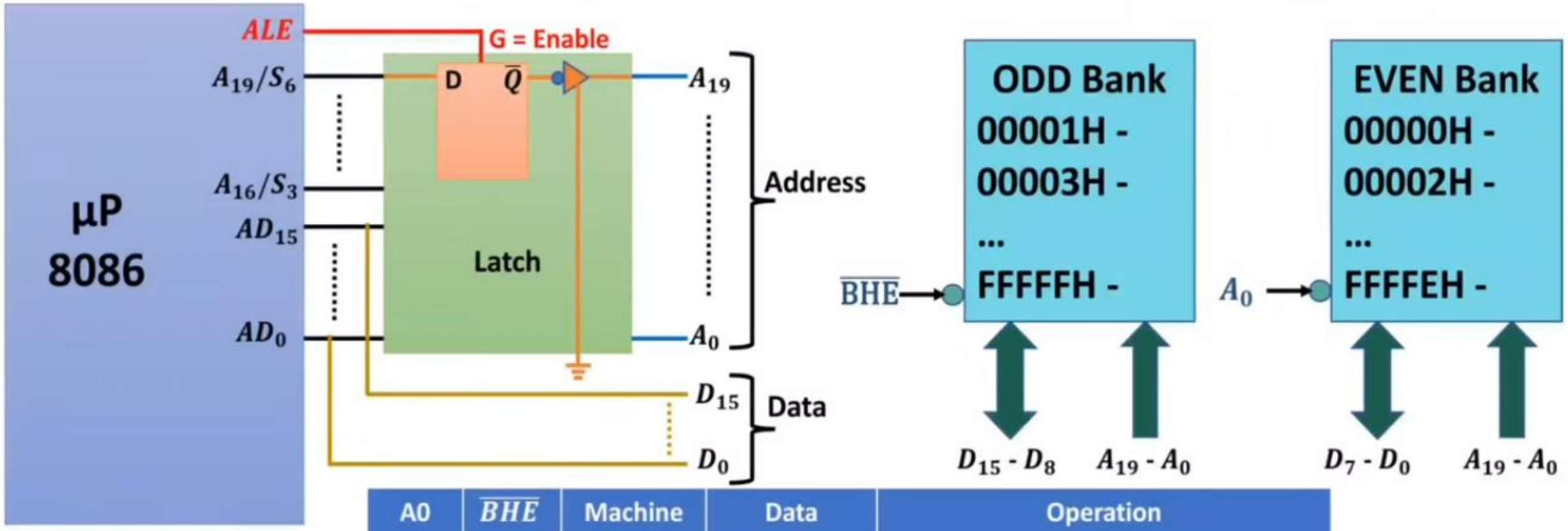
Little Endian: means microprocessor will store lower bytes first then store 2nd bytes in higher bytes Big endian: reverse of little endian

Microprocessors 8086 interfacing with memory Banking with Control Signals



BHE = Bus High Enable
ALE = Address Latch Enable
A0 = Chip Select for Even bank

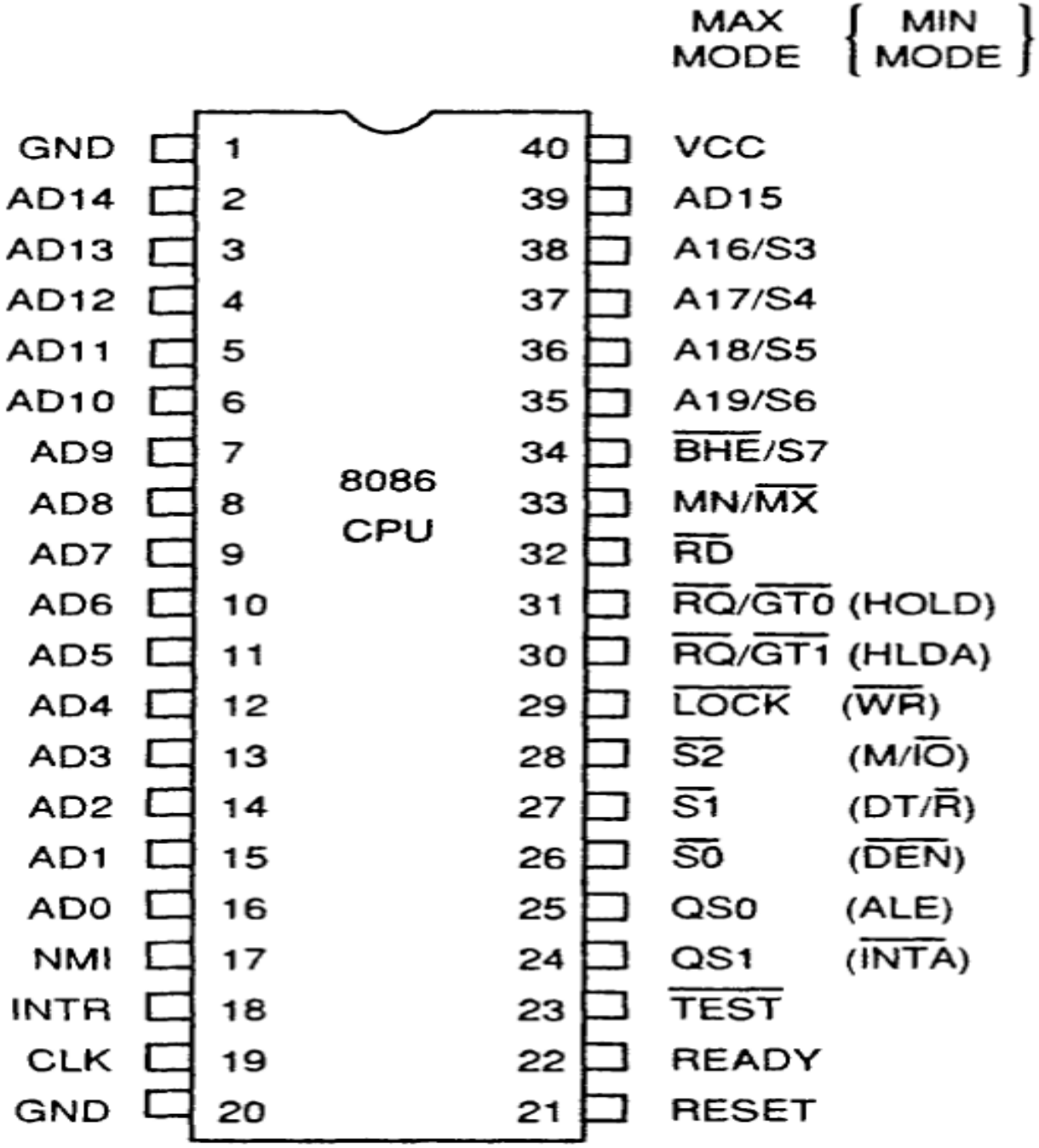
Memory Banking with Control Signals



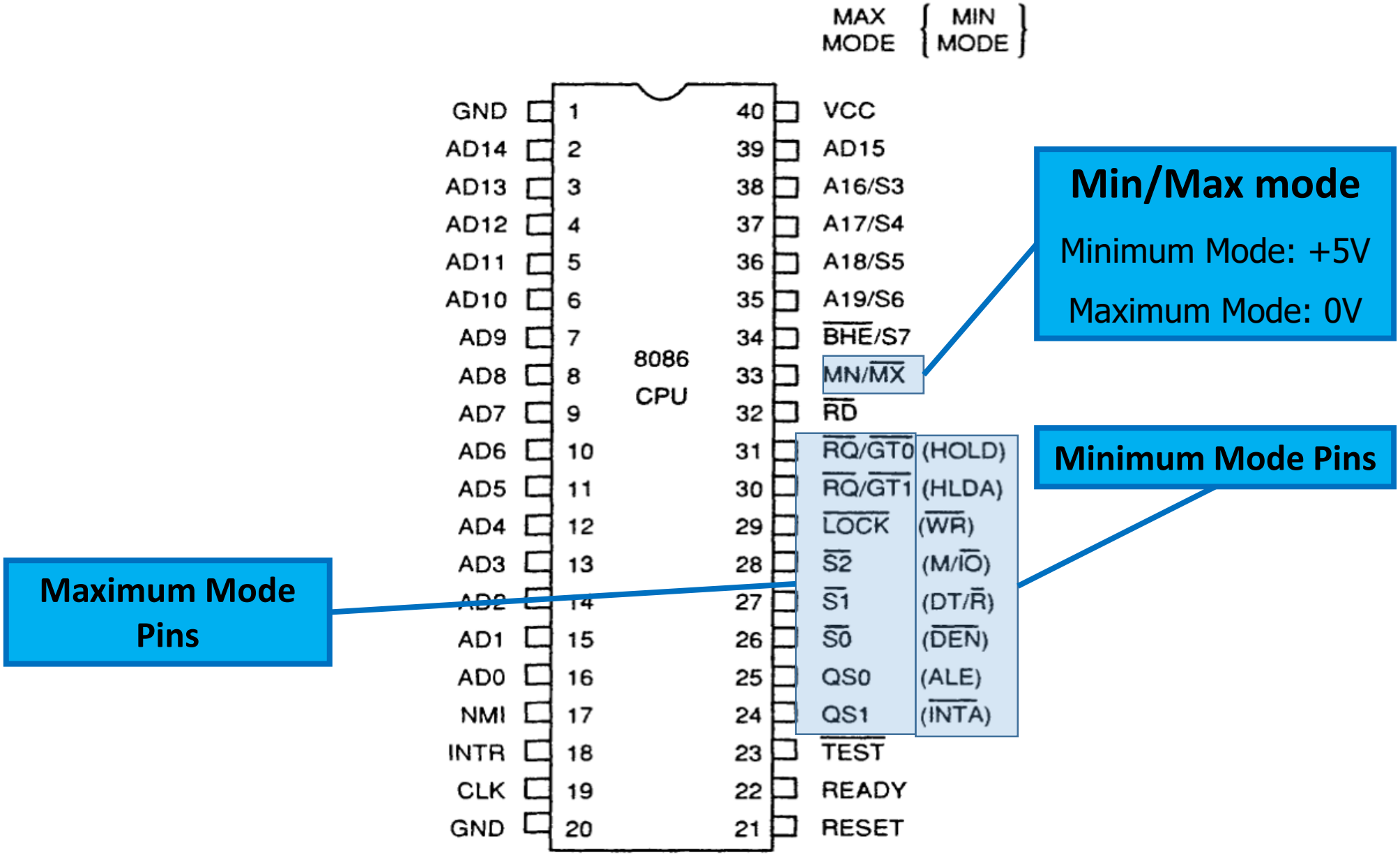
A0	$\overline{BHE}$	Machine Cycle	Data	Operation
0	0	1	D15-D0	Read a Word from Even Add
0	1	1	D7-D0	Read a Byte from Even Add
1	0	1	D15-D8	Read a Byte from Odd Add
1	0	2	D15-D8	Read a Word from Odd Add
0	1		D7-D0	

Pin Diagram of 8086

Pin Diagram of 8086



Intel 8086-Pin Details



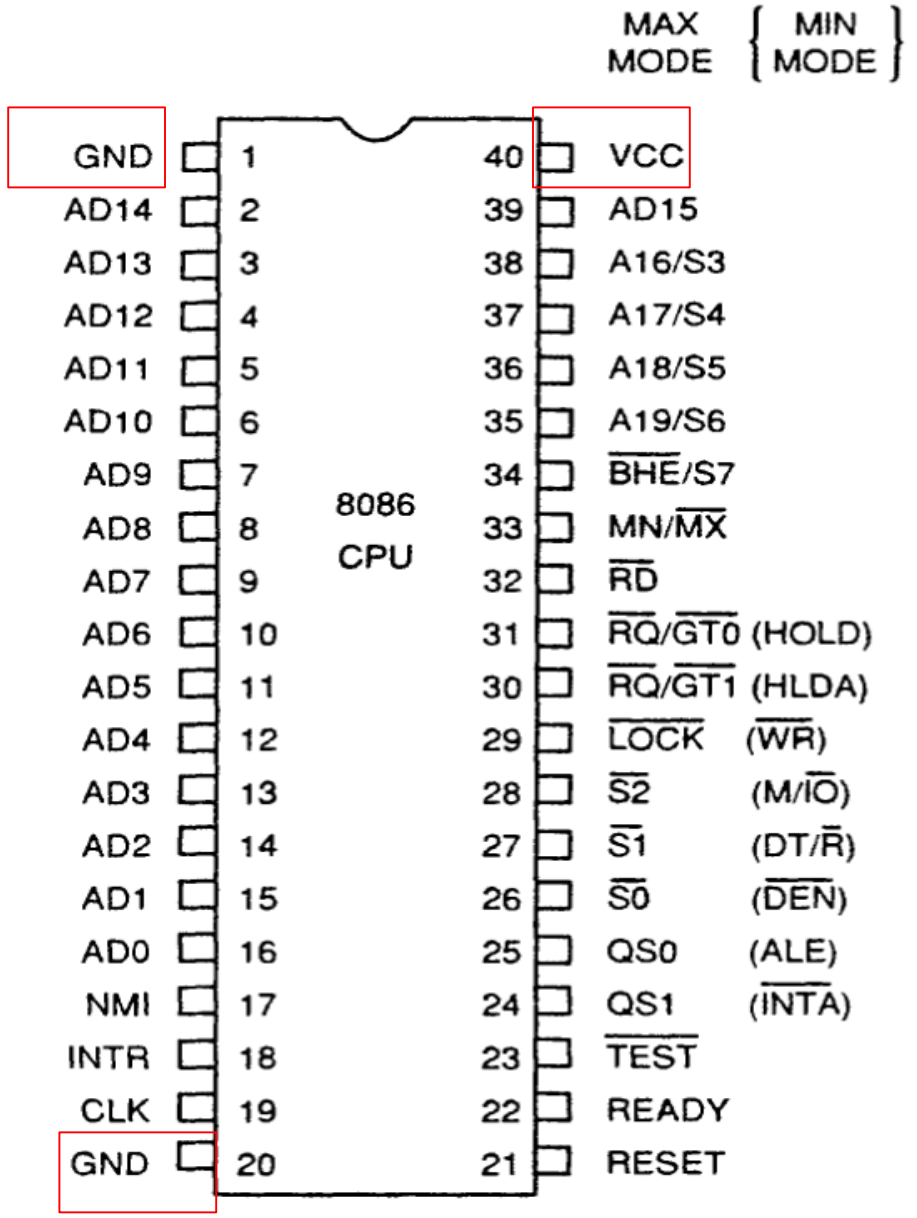
Pin Functions

- Out of 40 pins, 32 pins are having same function in minimum or maximum mode,
- And remaining 8 pins are having different functions in minimum and maximum mode.
- Following are the pins which are having same functions

Common Pin Description

V_{CC} – ***Pin number 40*** – At this pin, the external power supply of **+ 5V** is provided to the processor.

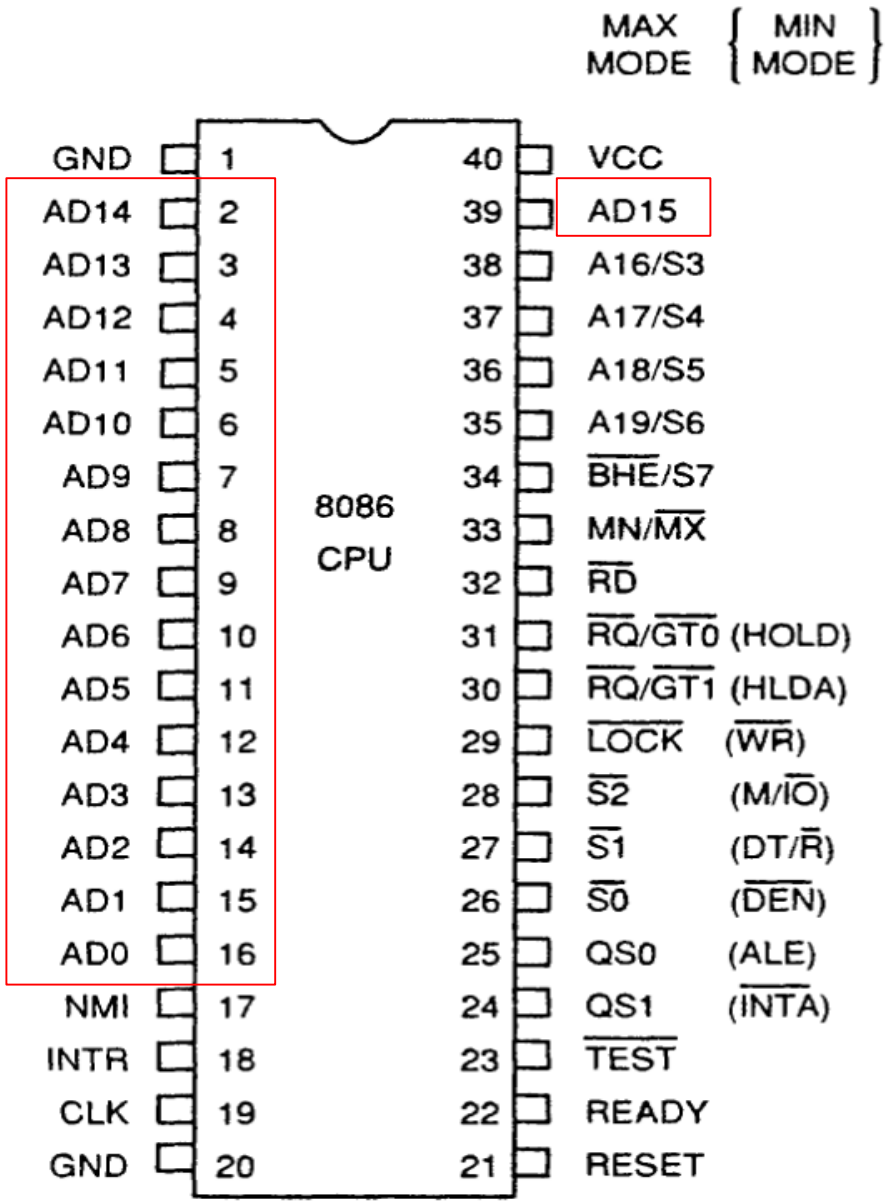
V_{SS} – ***Pin number 1 and 20*** – These two pins acts as the ground. This pin directs the extra current of the microprocessor to ground.



Common Pin Description

AD_0 – AD_{15} – ***Pin number 2 to 16 and 39*** – These are the multiplexed address and data bus.

We know that the 8086 microprocessor has 20-bit address bus and 16-bit data bus. So, the 16 lines of the address and data bus are multiplexed together so as to reduce the number of lines inside the IC.



Common Pin Description

A_{16}/S_3, A_{17}/S_4, A_{18}/S_5 and $A_{19}S_6$ – ***Pin number 35 to 38*** – Out of 20 address bits, 4 are present in the multiplexed form with the status signals. In the case of memory operations, these pins act as an address bus and contain the memory address of any particular instruction or data.

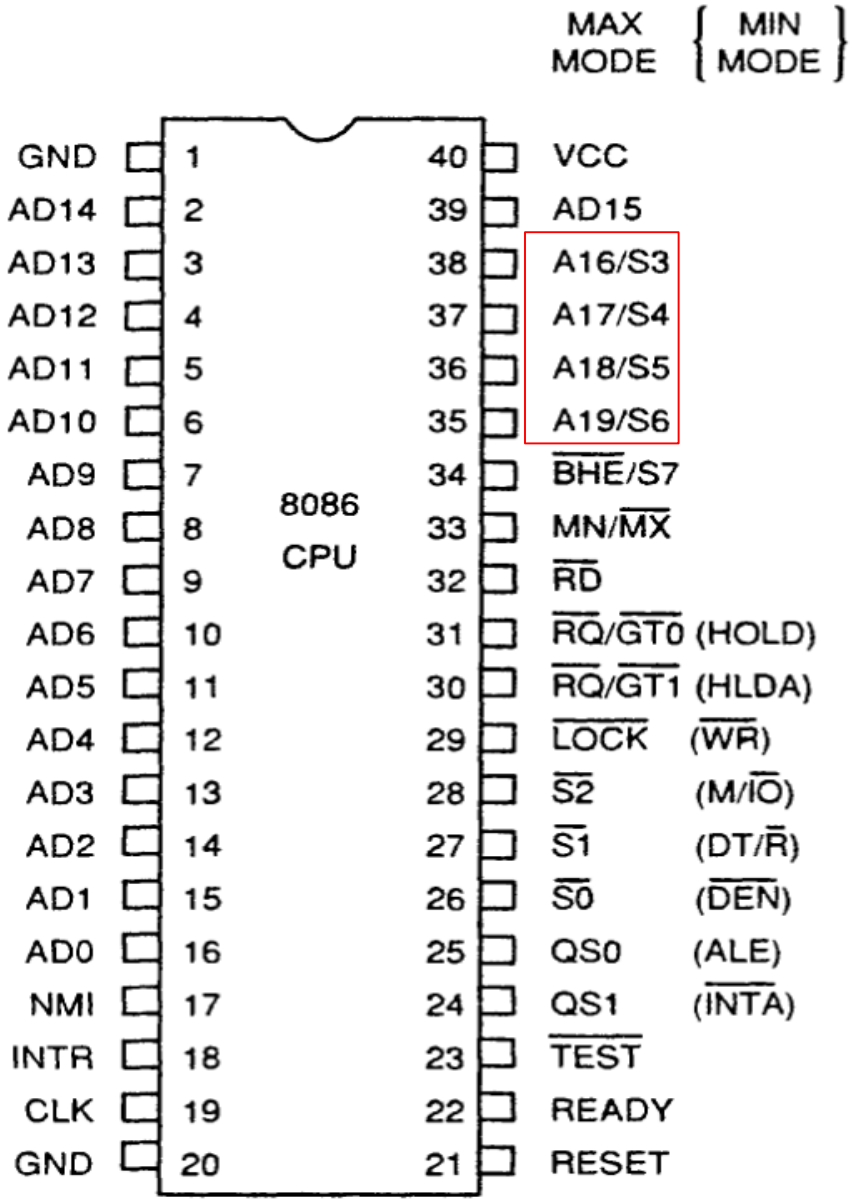
However, from I/O operations these pins are low that shows the status of the processor.

Basically, the signal at S_3 and S_4 show that which segment is currently accessed by the microprocessor among the four segments present in it.

The table below will show the encoding of S_3 and S_4:

S_3	S_4	STATUS
0	0	ES
0	1	SS
1	0	CS or idle
1	1	DS

Table 1

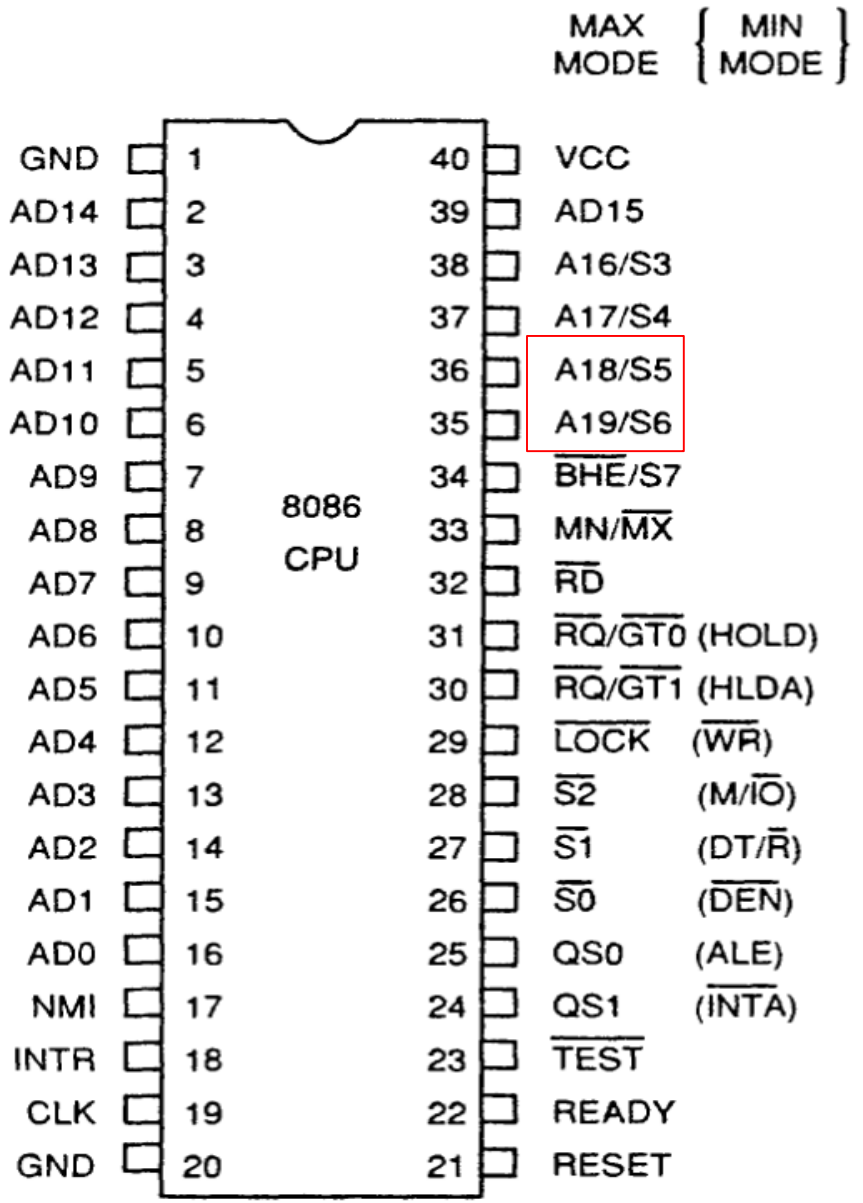


Common Pin Description

Also, S_5, when enabled, shows the presence of an interrupts in the microprocessor. So, basically, it serves as an **interrupt flag**.

The signal at S_6 shows the status of the bus master for the current operation. More simply we can say, whether the 8086 is the bus master or any other proficient device is acting as the bus master.

When 0 is present as the signal at this pin then it indicates the 8086 is holding the access of the bus otherwise it is high i.e., 1.



Common Pin Description

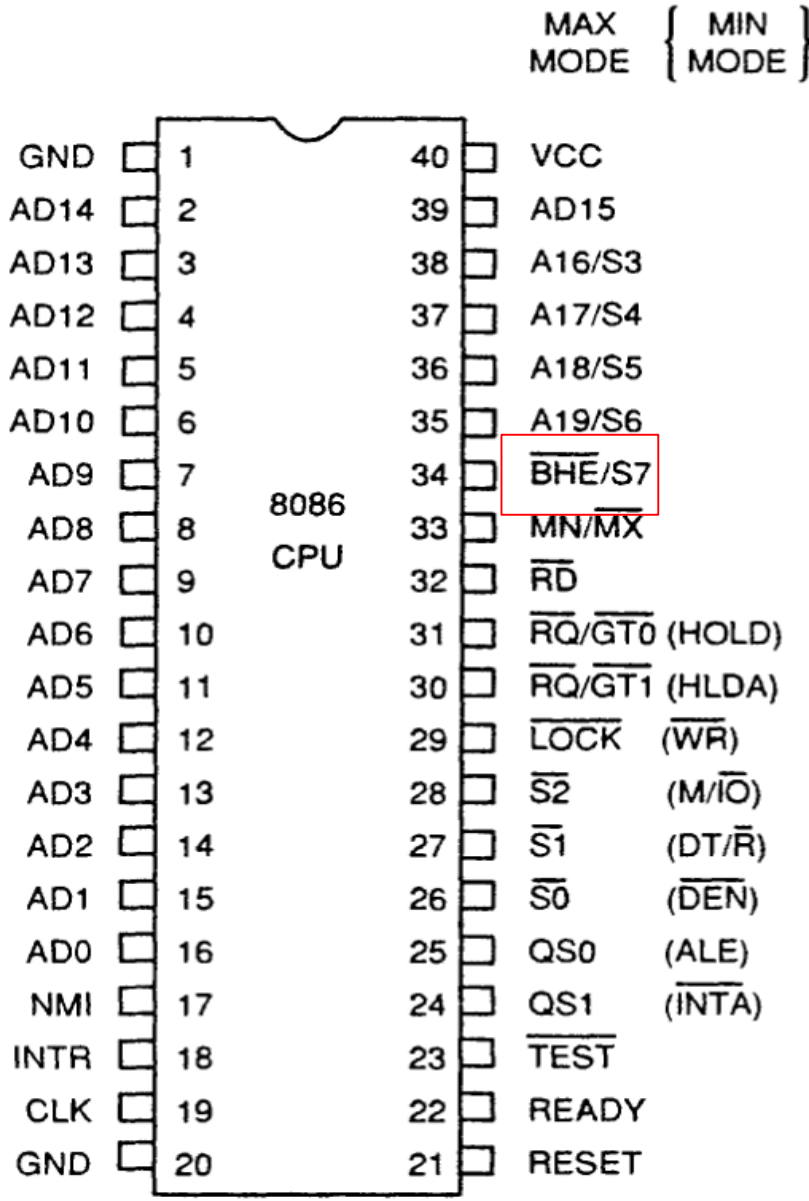
BHE' / S_7 – ***Pin number 34*** – BHE is an acronym for Bus High Enable. The combination of the BHE signal and S_7 status informs about the existence of the data on the bus. Also, different combinations show whether the bus is containing overall 16 bit, upper byte or lower byte of the data.

The table below represents the status for the signal at this pin:

BHE	S_7	STATUS
0	0	All 16 bits will be accessed
0	1	Upper byte will be accessed
1	0	Upper byte will be accessed
1	1	None or idle state

Table 2

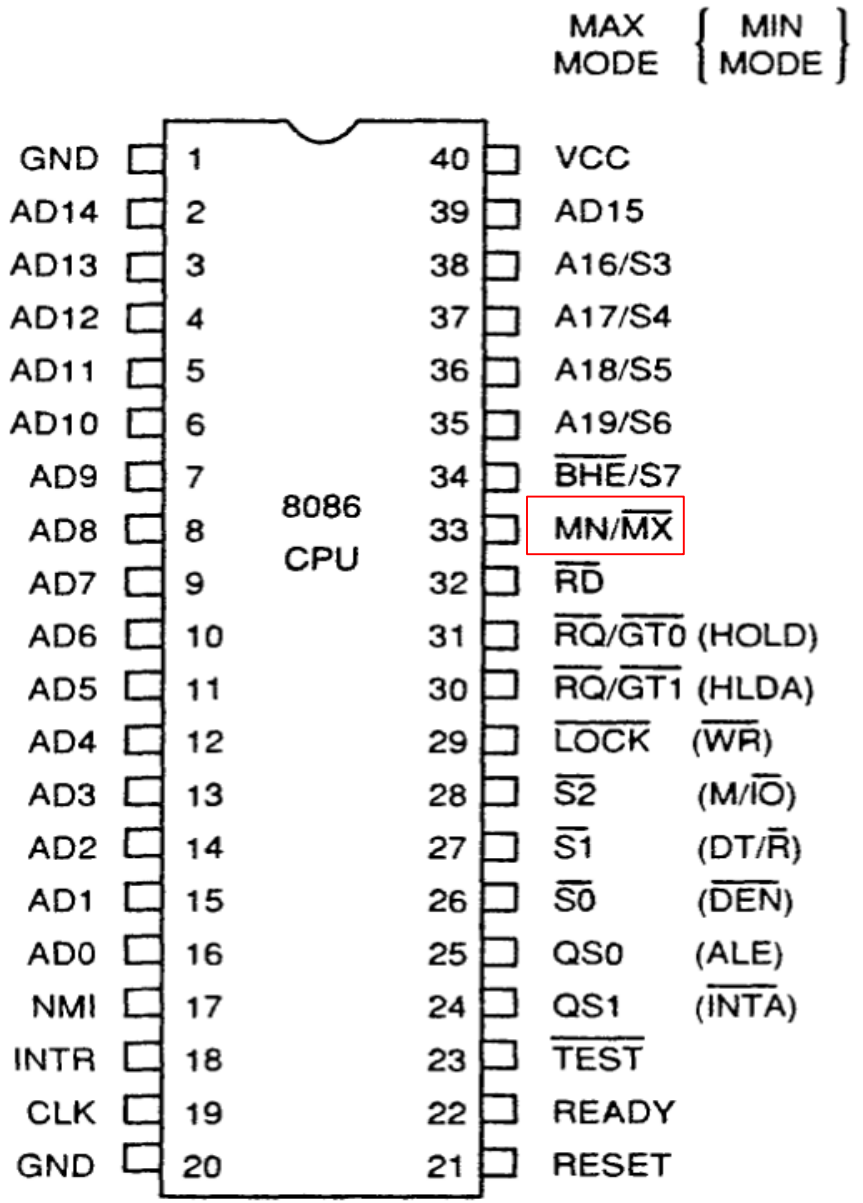
Electronics Desk



Common Pin Description

MN/MX’ – ***Pin number 33*** –The status at this particular pin shows whether the processor is operating in the minimum mode or maximum mode.

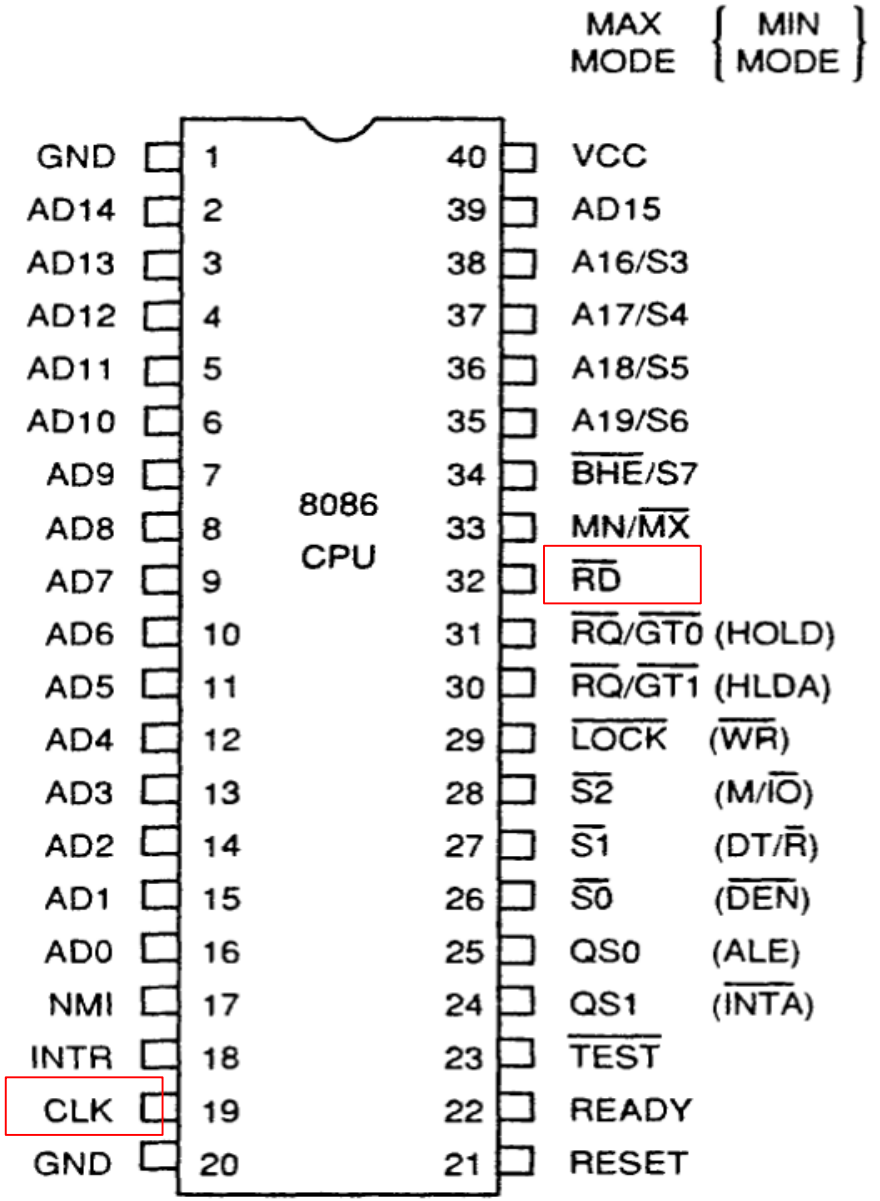
A signal 0 at this pin informs that the 8086 is operating in maximum mode i.e., multiple processors. While signal 1 shows the operation under minimum mode i.e., single processor.



Commmon Pin Description

RD’ – ***Pin number 32*** – An active low signal at this pin shows that the microprocessor is performing read operation with either memory or I/O devices.

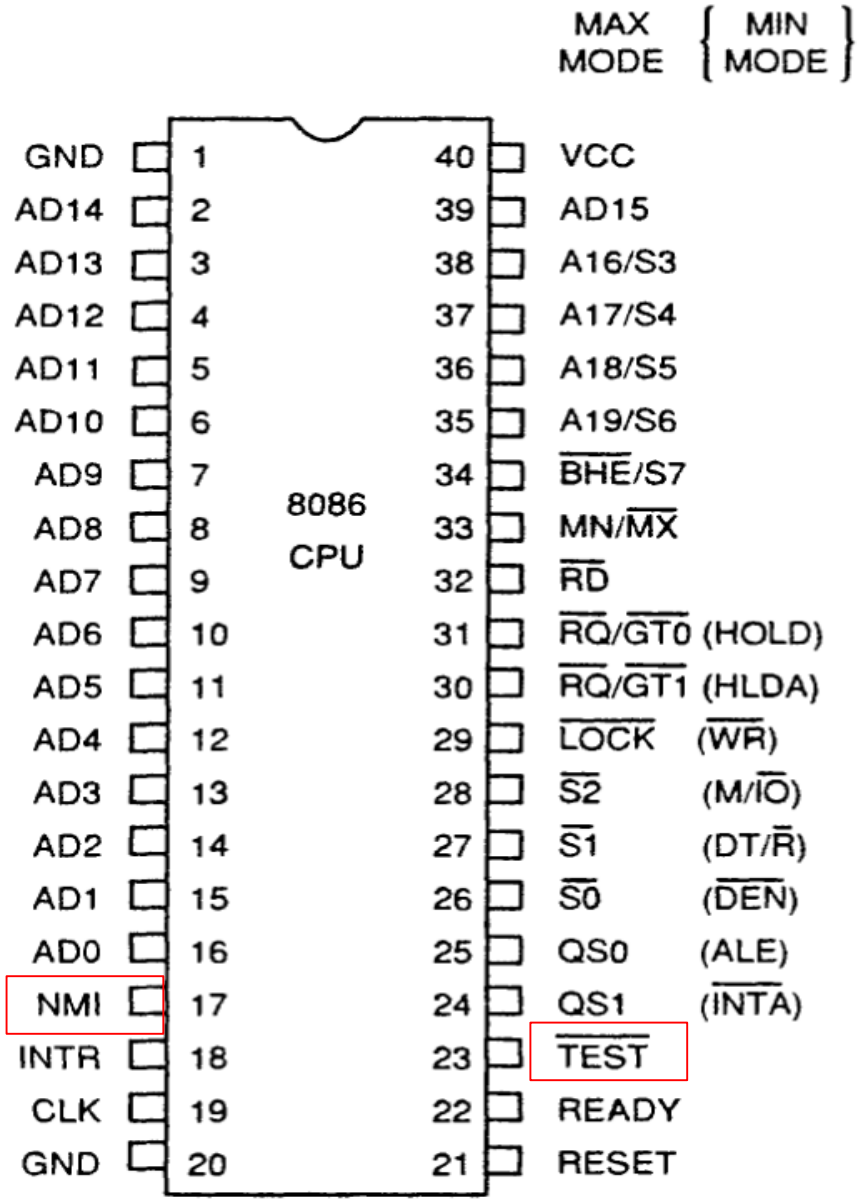
CLK – ***Pin number 19*** – A signal at this pin provides the timing to the internal operations that are being executed inside the microprocessor.



Common Pin Description

NMI – ***Pin number 17*** – NMI is Non-maskable interrupt. These are basically uncontrollable interrupts generated inside the processor. When an NMI occurs, then an interrupt service routine is generated by the interrupt vector table.

TEST – ***Pin number 23*** – This pin basically shows the wait instruction. Whenever a low signal at this pin occurs then the processing inside the processor continues. As against, in case of the high signal, the processor has to wait for the disabling of this pin.

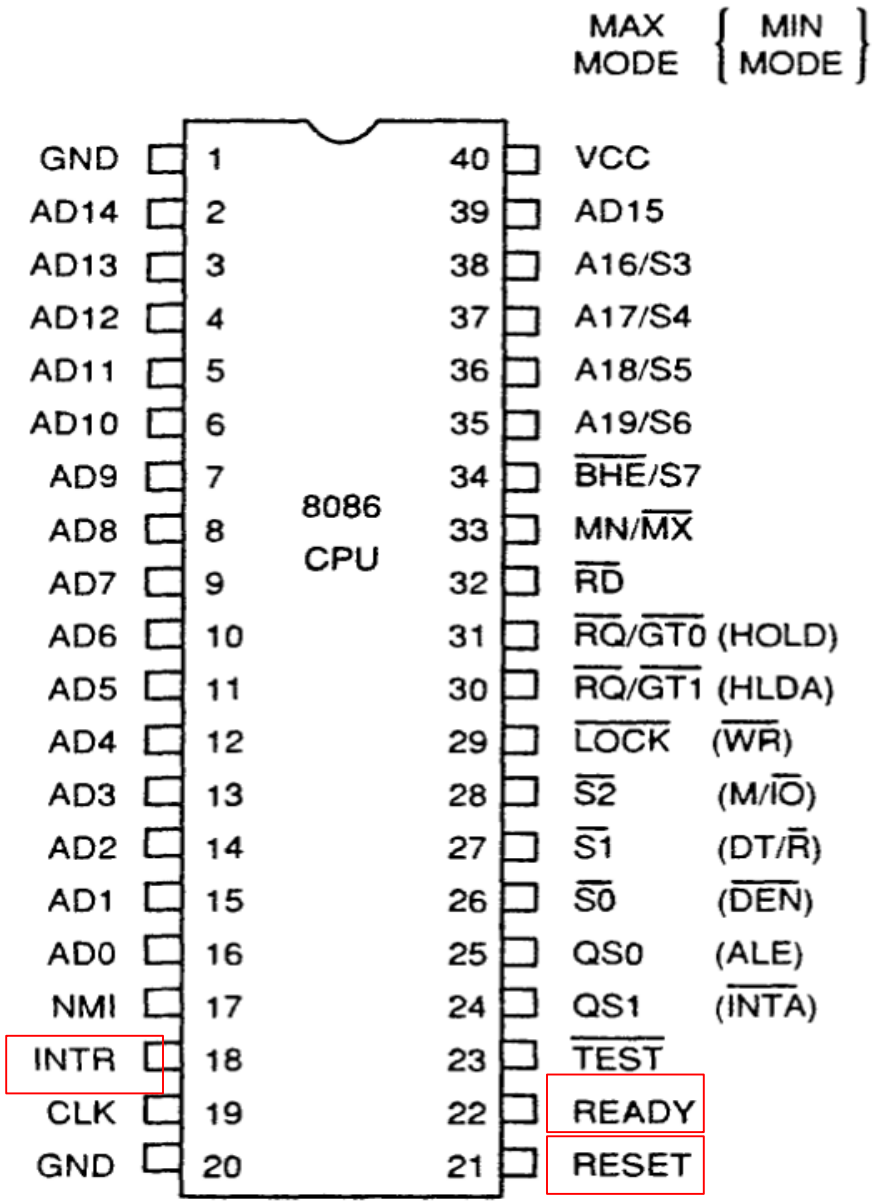


Common Pin Description

INTR – ***Pin number 18*** – INTR stands for an interrupt request. The processor after each clock cycle samples the INTR and if the signal at this pin is found to be high then the processor controls that interrupt internally.

READY – ***Pin number 22*** – This signal is used by the peripherals and memory devices in order to show the readiness for the next operation.

RESET – ***Pin number 21*** – Whenever this pin is enabled then it resets the processor and other devices connected to the system by immediately terminating the recent task.



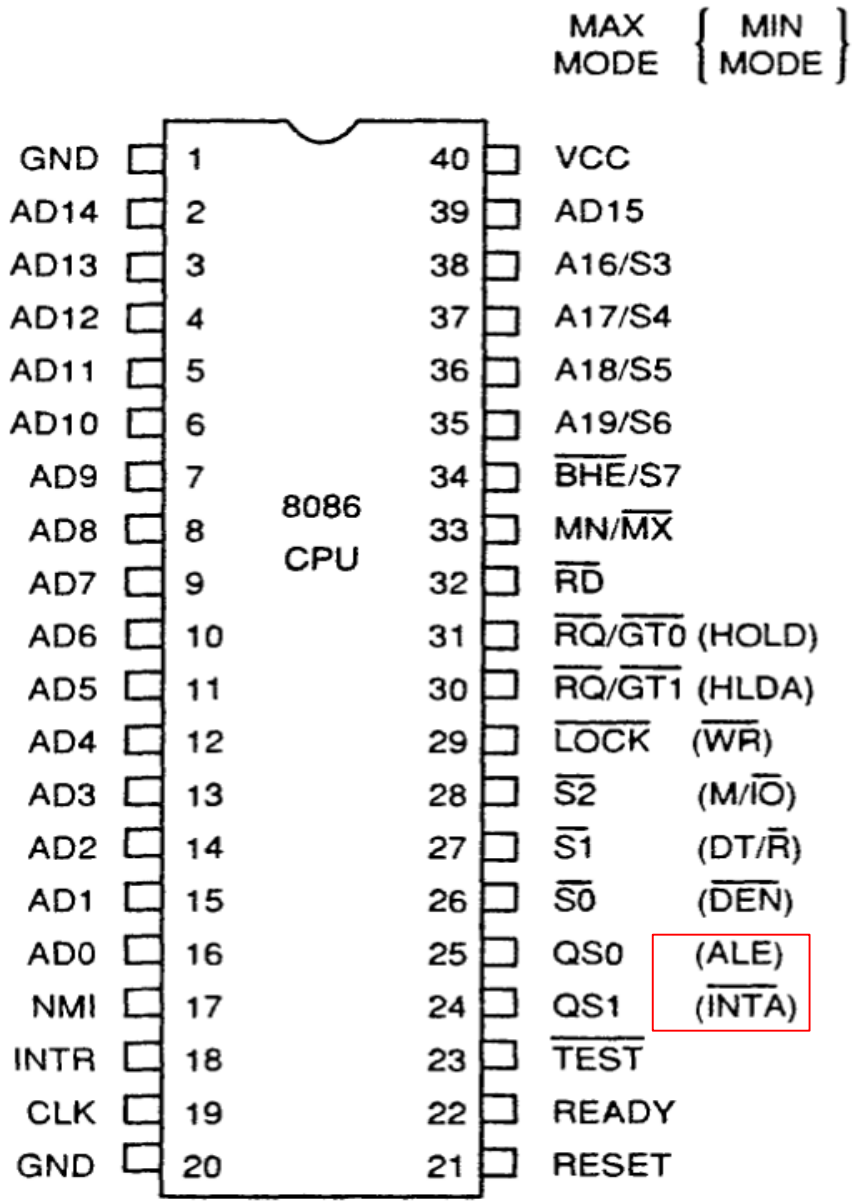
Pins in Minimum Mode

INTA' – ***Pin number 24*** – It is an interrupt acknowledge pin. Whenever an INTR signal is generated, then the microprocessor generates INTA signal, as a response to that interrupt.

ALE –***Pin number 25*** -**Address Latch Enable:**

When high, multiplexed address/data bus contains address information

.



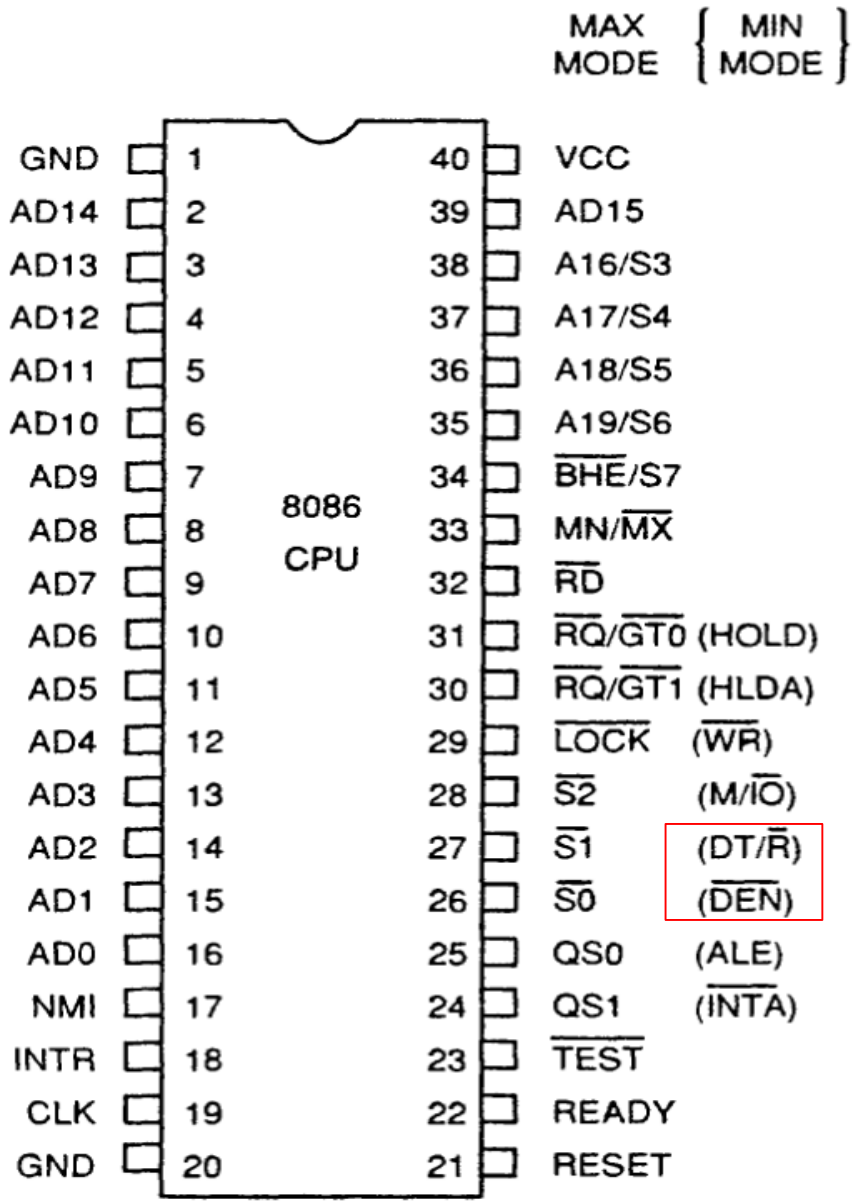
Pins in Minimum Mode

DEN’ – ***Pin number 26*** – DEN is used for data enable. This is an active low pin that means whenever a 0 is present at this pin then the transceiver gets enabled and it separates the data from the multiplexed address and data bus.

DT/R’ – ***Pin number 27*** – This pin is used to show whether the data is getting transmitted or is received. A high signal at this pin indicates that data is being transmitted. While a low indicates reception of data.

.

.



Pins in Minimum Mode

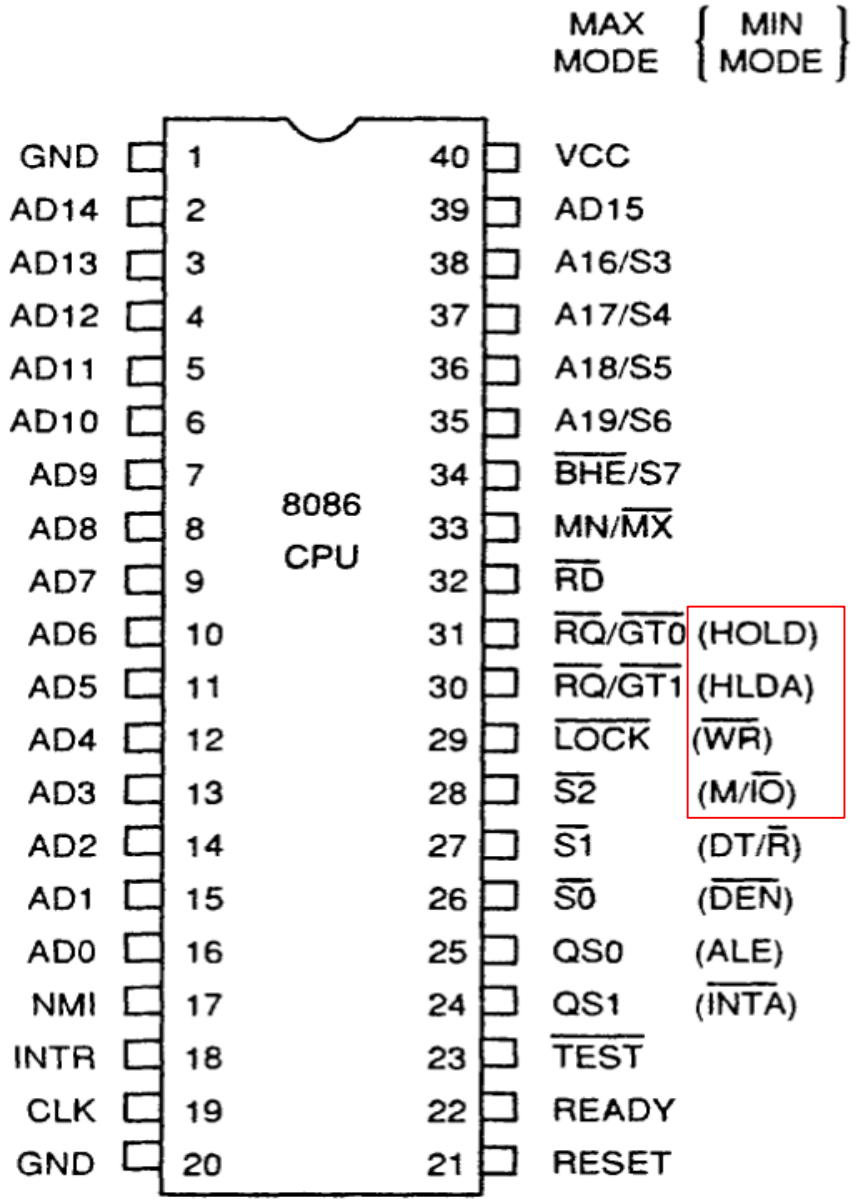
M/IO’ – ***Pin number 28*** – This pin indicates whether the processor is performing an operation with memory or I/O devices. Whenever a high is present at this pin then it shows the operation is carried out through the memory. While a low signal shows operation through I/O devices.

WR’ – ***Pin number 29*** – An active low signal at this pin indicates that the processor is performing write operation from either memory or I/O devices.

HOLD – ***Pin number 31*** – When an external device enables this pin then the processor stops accessing the buses immediately after the recent task gets over.

HLDA – ***Pin number 30*** – This pin is used as a response pin for the hold request. Once request for accessing the buses is produced by an external entity. Then the microprocessor acknowledges the device that its request will be considered once it gets over by the current operation..

.

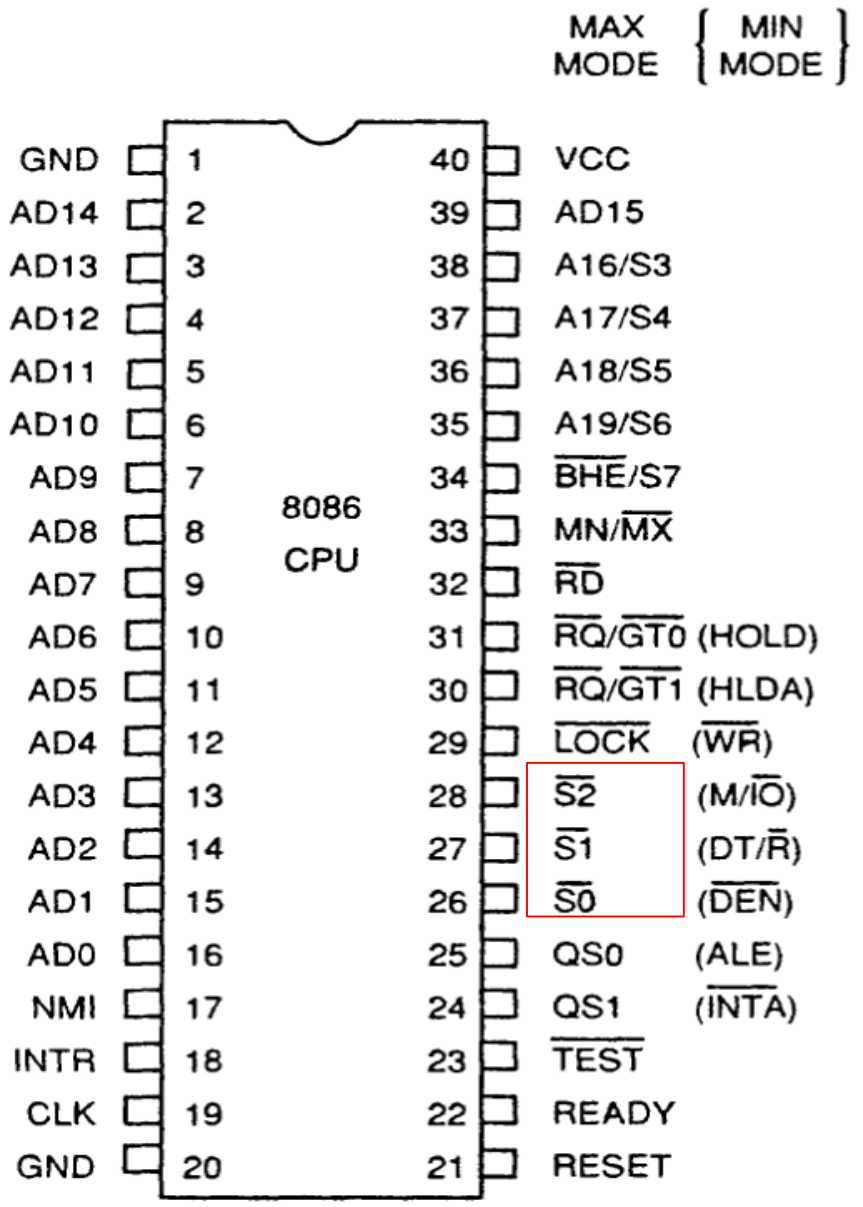


Pins in Maximum Mode

S_0', S_1' and S_2' – ***Pin number 26 to 28*** – These are basically 3 status pins and are active low. This means that if the status at all the 3 pins is 0 then it shows that multiple interrupts are to be handled in maximum mode.

The table below is representing the status of the processor in different combinations:

S2	S1	S0	Characteristics
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 state



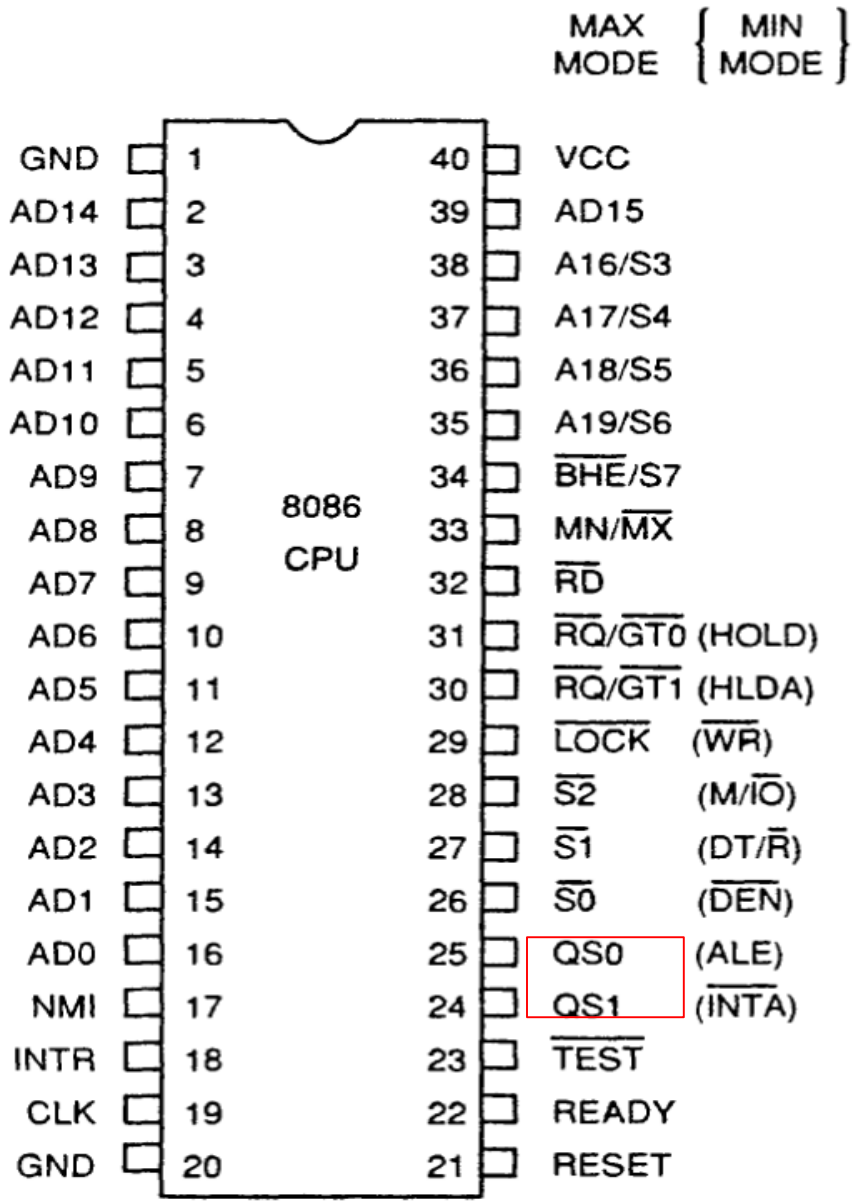
Pins in Maximum Mode

QS_0 and QS_1 – ***Pin number 24 and 25*** – These two pins indicate the status of the 6-byte pre-fetch queue present in the architecture of 8086.

QS_0	QS_1	STATUS
0	0	No operation
0	1	First byte from queue
1	0	Empty queue
1	1	Subsequent byte from queue

Table 4

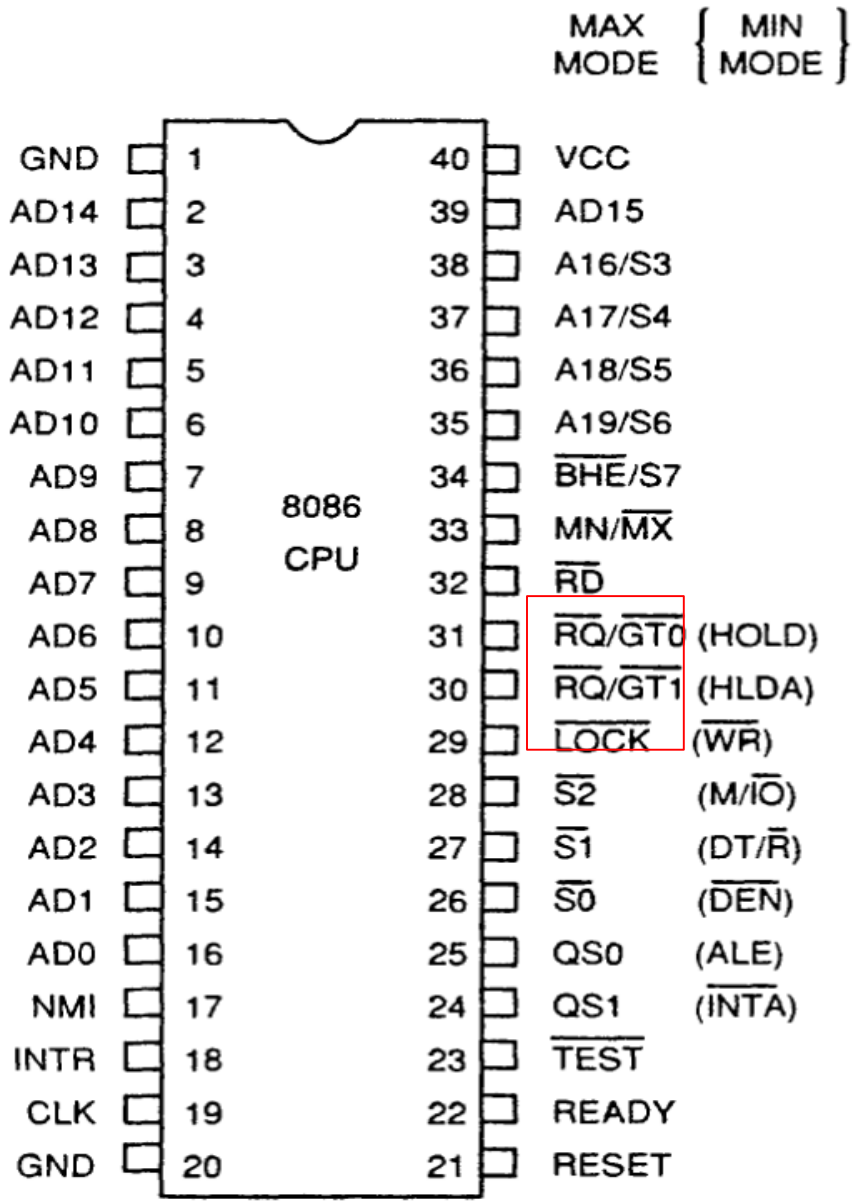
Electronics Desk



Pins in Maximum Mode

LOCK’ – ***Pin number 29*** –This pin is involved in maximum mode operation. So, basically, when a single processor is accessing the buses and peripherals then it locks the resources being used by it. So, that no other entity can access it until the recent processor frees it.

RQ’/ GT_0‘ and RQ’/ GT_1‘ – ***Pin number 30 and 31*** – Due to the involvement of multiple processors, these pins indicate the request and grant permission for accessing the buses, memory and peripherals.



Addressing Modes of 8086

Addressing Mode

- The **addressing mode** is the method to specify the operand of an instruction. The job of a microprocessor is to execute a set of instructions stored in memory to perform a specific task. Operations require the following:
- The operator or opcode - determines what will be done
- The operands - define the data to be used in the operation
- An assembly language program instruction consists of two parts

Example: ADD 7,8

Opcode	Operand

Addressing Mode

- **IMPORTANT TERMS**
- **Starting address** of memory segment.
- **Effective address or Offset**: An offset is determined by adding any combination of three address elements: **displacement, base and index.**
  - **Displacement:** It is an 8 bit or 16 bit immediate value given in the instruction.
  - **Base**: Contents of base register, BX or BP.
  - **Index**: Content of index register SI or DI.

Types of 8086 Addressing Mode

- The 8086 has 8 basic addressing modes (register, immediate and memory).
- The basic and other 8086 addressing modes can be classified into five groups:
  1. Addressing modes for accessing immediate and register data (register and immediate mode)
  2. Addressing modes for accessing data in memory (memory modes)
  3. Addressing modes for accessing I/O ports (I/O modes)
  4. Relative addressing mode
  5. Implied addressing mode

Types of Addressing Mode

Addressing modes for accessing immediate and register data (register and immediate mode)

1) Register addressing mode-

- In this mode, operands are specified using registers. This addressing mode is normally preferred because the instructions are compact and fastest executing of all instruction forms.
- Registers may be used as source operands, destination operands or both.
- Example:
- MOV AX, BX
- This instruction copies the contents of BX register into AX register.

$$AX \leftarrow BX$$

Types of Addressing Mode

2) Immediate addressing mode-

- In this mode, the operand is specified in the instruction itself. Instructions are longer but the operands are easily identified.
- Example:
- MOV CL, 12H
- This instruction moves 12 immediately into CL register. CL ← 12H

Types of 8086 Addressing Mode

Addressing modes for accessing data in memory (memory modes)

1) Direct addressing mode
2) Register indirect addressing mode
3) Based addressing mode
4) Indexed addressing mode
5) Based indexed addressing mode
6) String addressing mode

Types of Addressing Mode

Addressing modes for accessing data in memory (memory modes)

1) Direct addressing mode

- In this mode, the 16-bit effective address(EA) is directly included with the instruction.
- Example:
- MOV CX, DS:START
- This instruction moves the 16-bit contents of a 20-bit physical memory location computed from START and DS into CX.

Types of Addressing Mode

2) Register indirect addressing mode

- In this mode, the effective address (EA) is specified in either a pointer or an index register.
- The pointer register can be either base register BX or base pointer register BP and index register can be either the Source Index (SI) register or the Destination Index (DI) register.
- The 20-bit physical address is computed using DS and EA.
- Example:
- MOV [DI], BX
- The destination operand of the above instruction is in register indirect mode, while the source operand is in register mode.
- This instruction moves the 16-bit content of BX into a memory location offset by the value of EA specified in DI from the current contents in DS*16.

Types of Addressing Mode

3) Based addressing mode

- In this mode, EA is obtained by adding an 8 bit or a 16 bit displacement value to the contents of BX or BP.

- Example:
- MOV AL, START [BX]
- The source operand in the above instruction is in based mode.
- EA is obtained by adding the value of START and BX.
- The 20-bit physical address is produced from DS and EA.
- The 8-bit content of this memory location is moved to AL. The displacement START can be either 16-bit or 8-bit.

Types of Addressing Mode

4) Indexed addressing mode-

In this addressing mode, the EA is calculated by adding the contents of SI or DI register and 8-bit/16-bit displacements.

Example

MOV BH, ARRAY[SI]

This instruction moves the contents of the 20-bit address computed from the displacement ARRAY, SI and DS into BH.

Types of Addressing Mode

5) Based indexed addressing mode-

- Here, EA is calculated as base register (BX or BP), an index register (SI or DI), and a displacement..
- Example:
- MOV ALPHA [SI] [BX], CL
- This instruction moves the 8-bit content of CL to 20-bit physical address ALPHA+SI+BX+DS.

Types of Addressing Mode

6) String addressing mode

- Employed in string operations to operate on string data.
- The effective address (EA) of source data is stored in SI register and the EA of destination is stored in DI register.
- Segment register for calculating base address of source data is DS and that of the destination data is ES
- Example:

MOVS BYTE

Types of Addressing Mode

Addressing modes for accessing I/O ports (I/O modes)

Standard I/O uses port addressing modes. Two types:

i) **I/O mode (direct):**

Port number is an 8-bit immediate operand.
Example: OUT 05 H, AL
Outputs [AL] to 8 bit port 05 H

ii) I/O mode (indirect):

The port number is taken from DX.

Example 1: IN AL, DX

If [DX] = 5040
8 bit content by port 5040 is moved into AL.

Example 2: IN AX, DX

Inputs 8 bit content of ports 5040 and 5041 into AL and AH respectively.

Types of Addressing Mode

Relative addressing mode

- Instructions using this mode specify the operand as a signal 8-bit displacement relative to PC.
- Example:
- JNC START
- This instruction means that if carry=0, then PC is loaded with current PC contents plus the 8-bit value of START; otherwise the next instruction is executed.

Types of Addressing Mode

Implied addressing mode

- Instruction using this mode have no operands.
- Example:
- CLC
- This clears the carry flag to zero.

Instruction Set of 8086 Microprocessor

Instruction Set of 8086

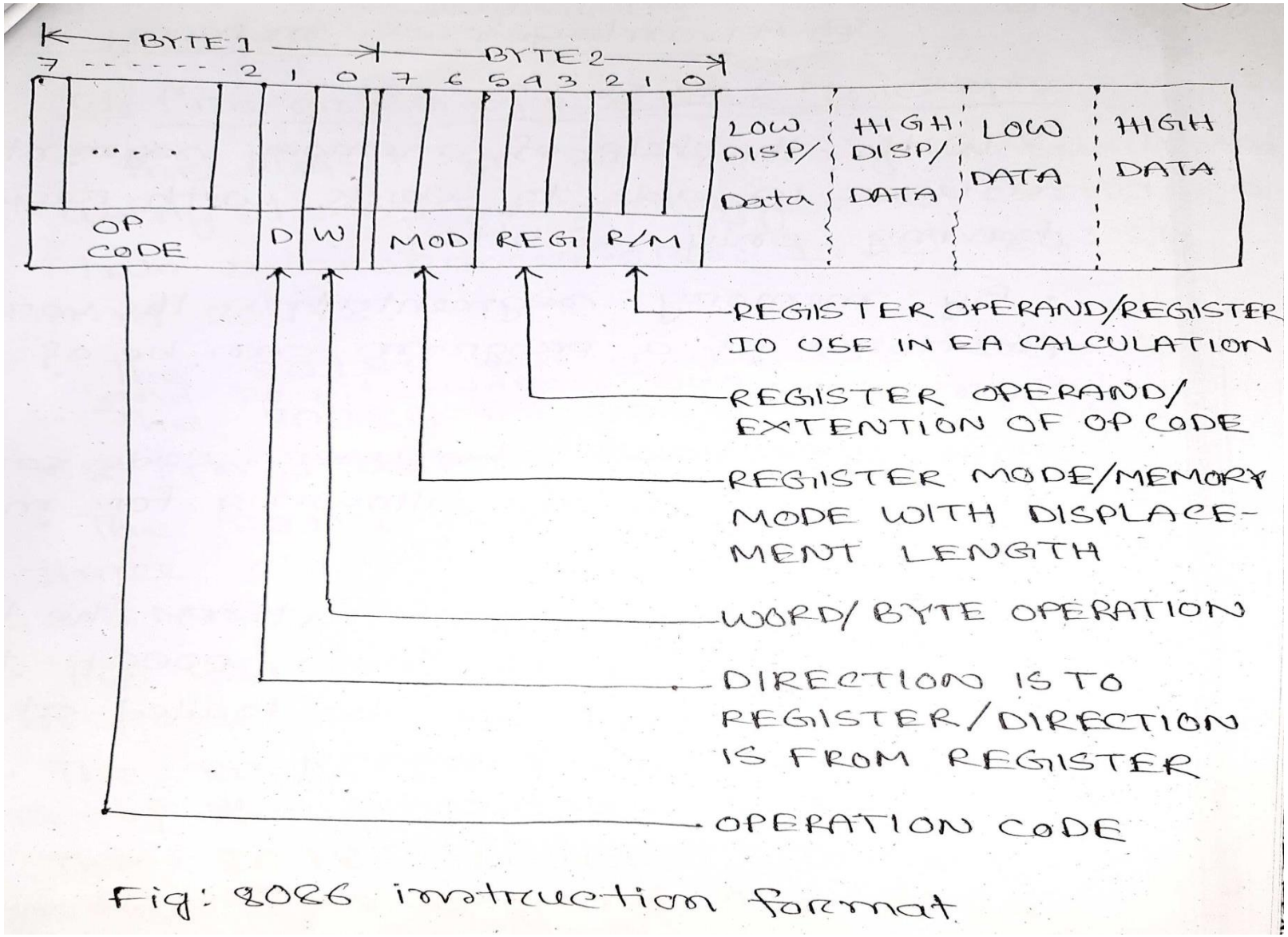


Fig: 8086 instruction format

Accessing words from odd address takes longer time than even address

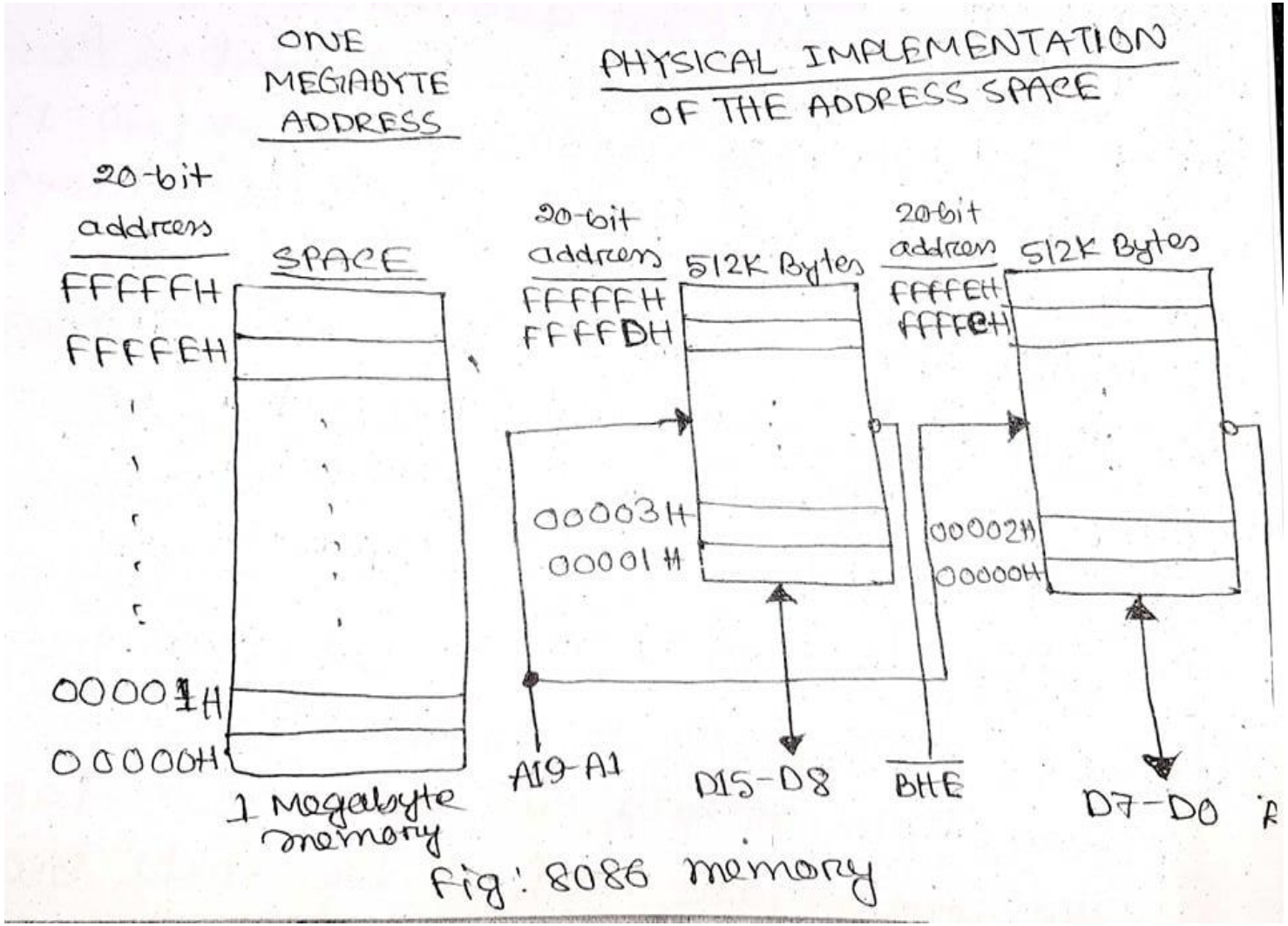


Fig: 8086 memory

Offset for the Specific Segment

Segment	Offset Registers	Function
CS	IP	Address of the next instruction
DS	BX, DI, SI	Address of data
SS	SP, BP	Address in the stack
ES	BX, DI, SI	Address of destination data (for string operations)

Example:

- Determine the effect of each one of the following 8086 instructions:
- i) PUSH [BX]
- ii) DIV DH
- iii) CWD
- iv) MOVSB
- v) MOV START [BX], AL

Assume the following data prior to execution of each one of the above instructions independently.

[DS] = 3000H	**[SI] = 0400H**	**[36000H] = 02H**
[ES] = 5000H	[DI] = 0500H	[36001H] = 03H
[DX] = 0400H	DF = 0	[50500H] = 05H
[SP] = 5000H	[BX] = 6000H	[30400H] = 02H
[SS] = 6000H	START = 05H	[30401H] = 03H
[AX] = 00A9H		

Chapter 4
INTEL 80186/ 80286 / 80386

Questions

- Write down the properties/characteristics of 80186/80286/80386 microprocessor.
- Draw the functional block diagram of the 80186/80286/80386 microprocessor and explain briefly the functions of its different blocks.
- Mention the addressing modes of 80186 microprocessor.
- Explain with necessary figures how the 80286 microprocessor translates logical addresses to physical addresses.
- What are the differences between 80286 and 8086 microprocessor?
- Draw a comparison chart for 80186, 80286 and 80286 microprocessor.
- What are the differences between a register and a memory location?

80286 Address Translation

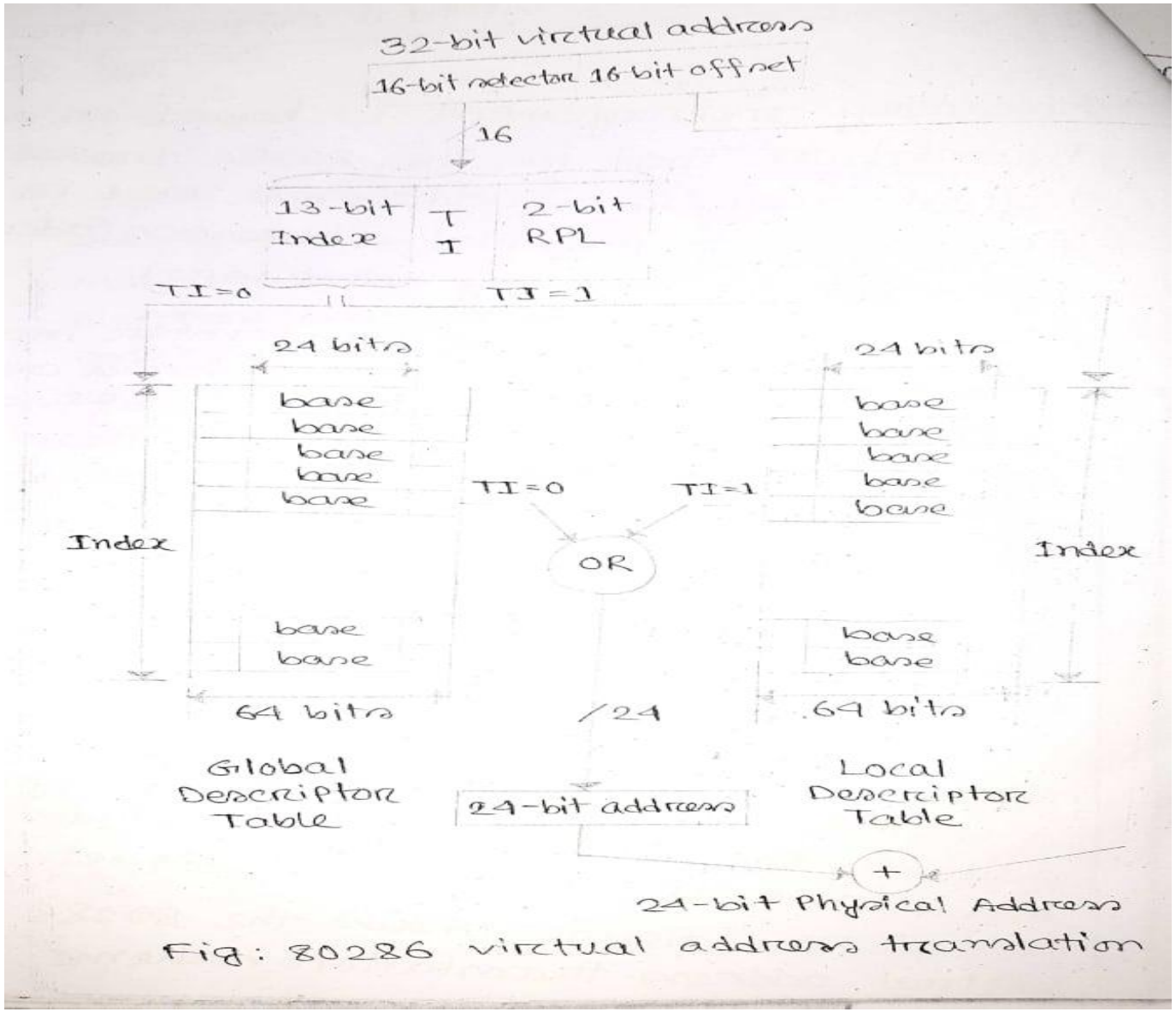


Fig: 80286 virtual address translation