

8086 Microprocessor

Feature of Microprocessor 8086

❖ It is upgraded microprocessor from 8085. 8086 has following features:

- ☐ 8086 has 16 bits ALU. [Older version 8085 has 8 bits of ALU.] So, operating speed 8086 have increased significantly well.
- ☐ 8086 has 16 bits of Data bus. So in one machine cycle it can transfer 2 bytes. [Older version 8085 has 8 bits of data bus.]
- ☐ 8086 has 20 bits of Address bus. With 8086 we can interface total memory by $2^{20} = 1\text{MB}$ [Older version 8085 has 16bits Address bus.]
- ☐ 8086 has two internal hardware units. BIU – Bus Interface Unit & EU – Execution Unit.
- ☐ 8086 supports pipelining. So Multiple instructions can be executed in parallel to increase execution speed of microprocessor.
- ☐ Four 16 bits general purpose registers (AX, BX, CX & DX). We can use it with 8 bits also (AH, AL, BH, BL, CH, CL, DH, DL). Two 16 bits Index registers (SI and DI). Two 16 bits stack pointers (SP and BP).
- ☐ 8086 supports memory segmentation. (CS, DS, ES and SS)
- ☐ 16 bit flag register.
- ☐ Two hardware modes : Maximum Mode and Minimum Mode.
- ☐ 8086 provides memory banks. So 1MB is bisected into two bank of 512KB, Lower bank (even) and higher bank (Odd).
- ☐ 8086 had Upgraded Instructions set and Interrupt service.

Bus Interface Unit of Microprocessor 8086

❖ BIU is responsible for establishing communications with external peripheral devices and memory via system bus.

❖ Main Purpose of BIU serves is as followed

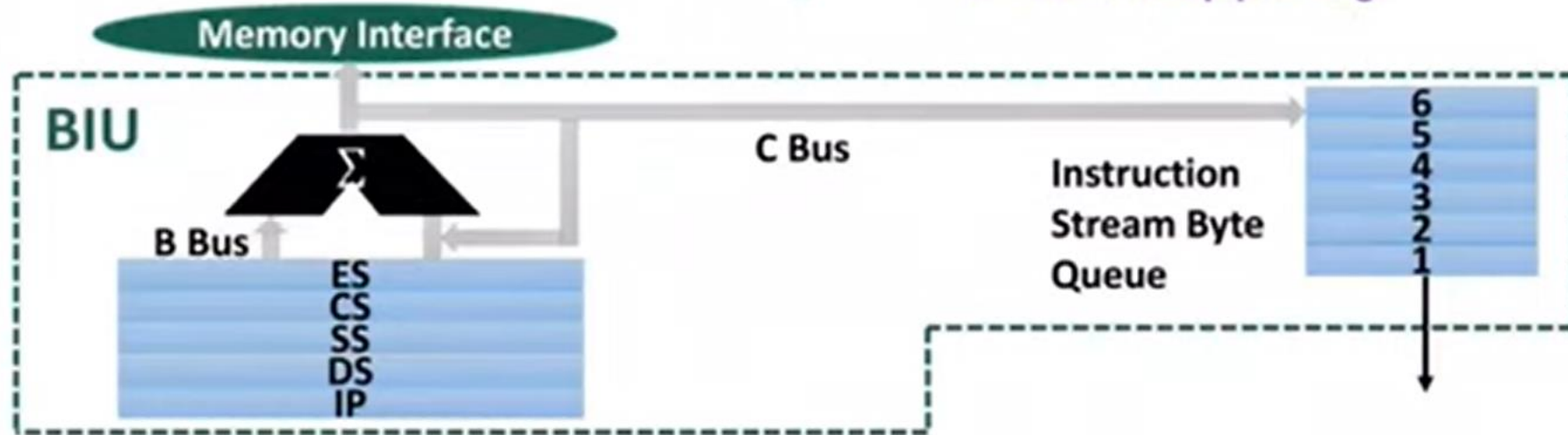
- ❑ It fetches the instructions from Memory
- ❑ It reads data from IO and Memory
- ❑ It writes data into IO and Memory
- ❑ It provides the address relocation facility

❖ BIU contains three main parts

- ❑ Segment Registers
- ❑ Instruction Queue
- ❑ Instruction Pointer

❖ Instruction Queue

- ❑ BIU prefetches six instruction bytes in advance from memory.
- ❑ The prefetched instructions are stored in a group of high speed registers known instruction queue.
- ❑ This Instruction queue works on FIFO order.
- ❑ BIU and EU works in parallel.
- ❑ The simultaneous operations of BIU and EU is possible only when the EU does not require the system bus.
- ❑ The process of fetching the next instruction in advance while the EU is executing the current instruction is pipelining.



Bus Interface Unit of Microprocessor 8086

❖ Segment Register of BIU

- ❑ The 8086 MP has the capability of addressing 1MB memory, which is divided into 16 local segments.
- ❑ Each segment contains 64KB memory
- ❑ But any instant 8086 works with only four 64KB segments.

❖ Each segment associated with segment register

- ❑ Code Segment Register CS – 16bits
- ❑ Data Segment Register DS – 16bits
- ❑ Stack Segment Register SS – 16bits
- ❑ Extra Segment Register ES – 16bits

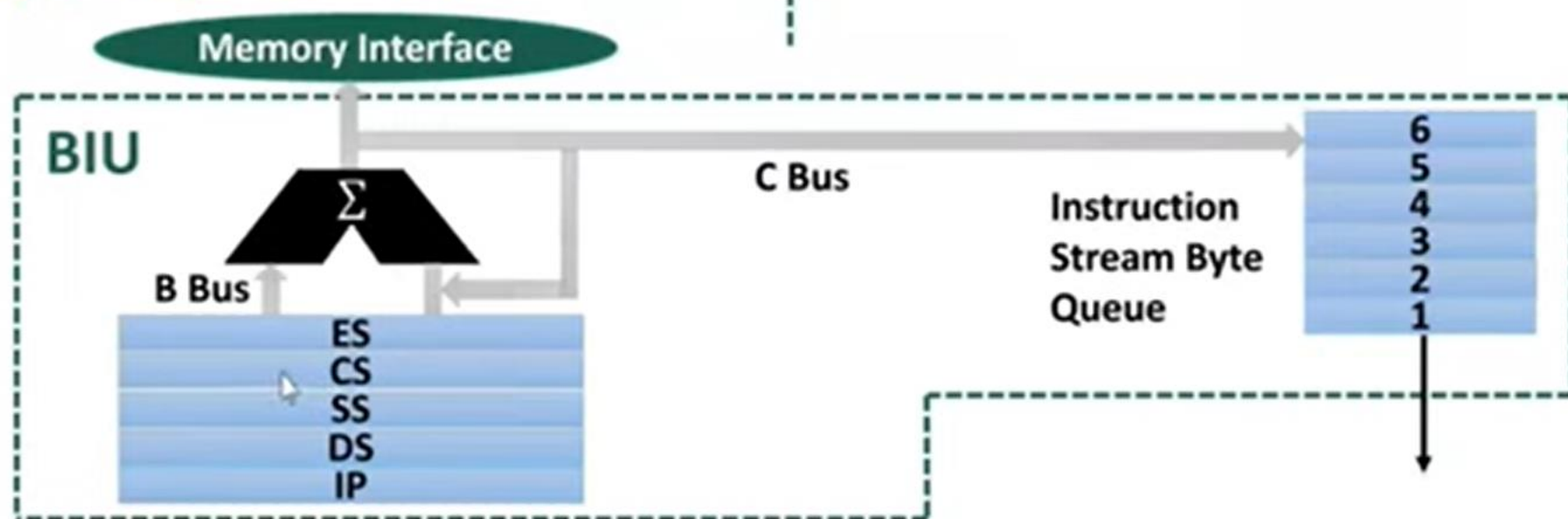
❖ Segment Register of BIU is used to store the starting address of Memory segment.

❖ BIU generates 20 bits address using segment register and Offset pointer registers.

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

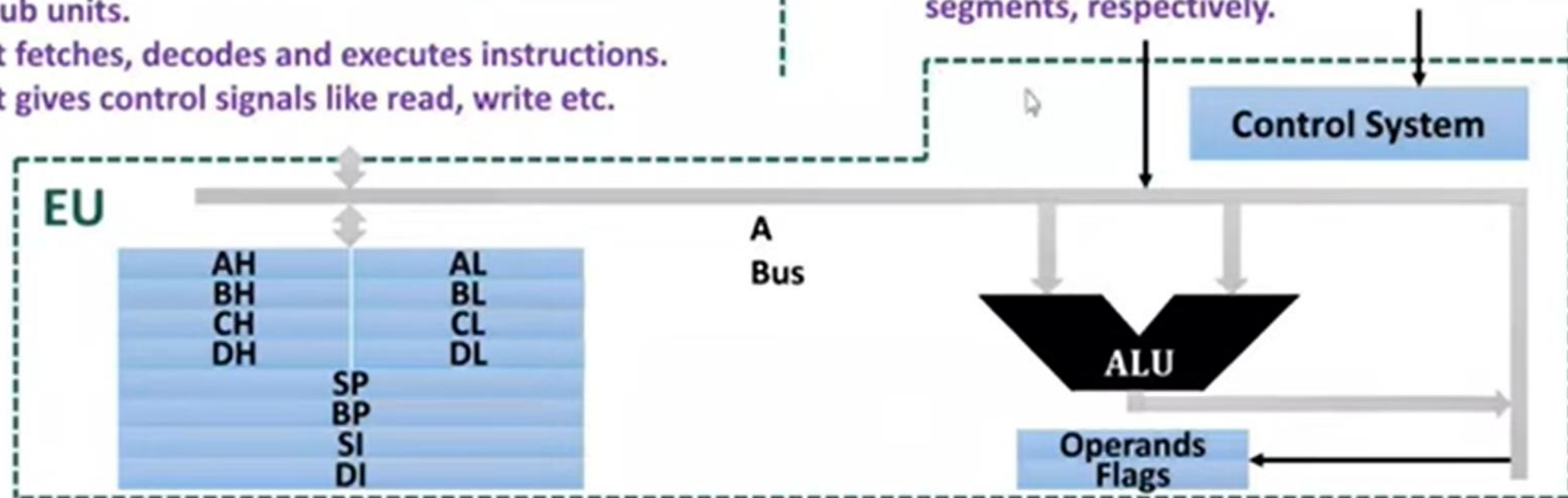
❖ Instruction Pointer of BIU

- ❑ IP holds the address of next instruction which is to be executed next.
- ❑ It contains the OFFSET value of next Address.



Execution Unit of Microprocessor 8086

- ❖ EU informs BIU from where the next instruction or data to be fetched.
- ❖ The EU performs following functions:
 - ❑ It picks up the instruction from the instruction queue of BIU.
 - ❑ It decodes the instructions and then executes the instruction.
 - ❑ It updates the status of flag register.
- ❖ Control Unit
 - ❑ It is controlling and coordinating all the activities of sub units.
 - ❑ It fetches, decodes and executes instructions.
 - ❑ It gives control signals like read, write etc.
- ❖ ALU of EU
 - ❑ It performs all the arithmetic and logical operations.
 - ❑ Results may be stored in general purpose registers or index registers.
 - ❑ It updates Flag register after every instructions.
- ❖ General Purpose Registers & Pointer of EU
 - ❑ EU have four general purpose registers.
 - ❑ These registers are used for accessing data very fast.
 - ❑ EU have SP and BP for stack.
 - ❑ Index registers are SI and DI for data and Extra segments, respectively.



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 use 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 $10H = 16\text{Bytes}$

Physical Address Calculation in 8086



Physical Address of 20 bits $A_{19} - A_0$

- 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

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

$$PA = 78EFH \times 10H + (0000H \text{ to } FFFFH)$$

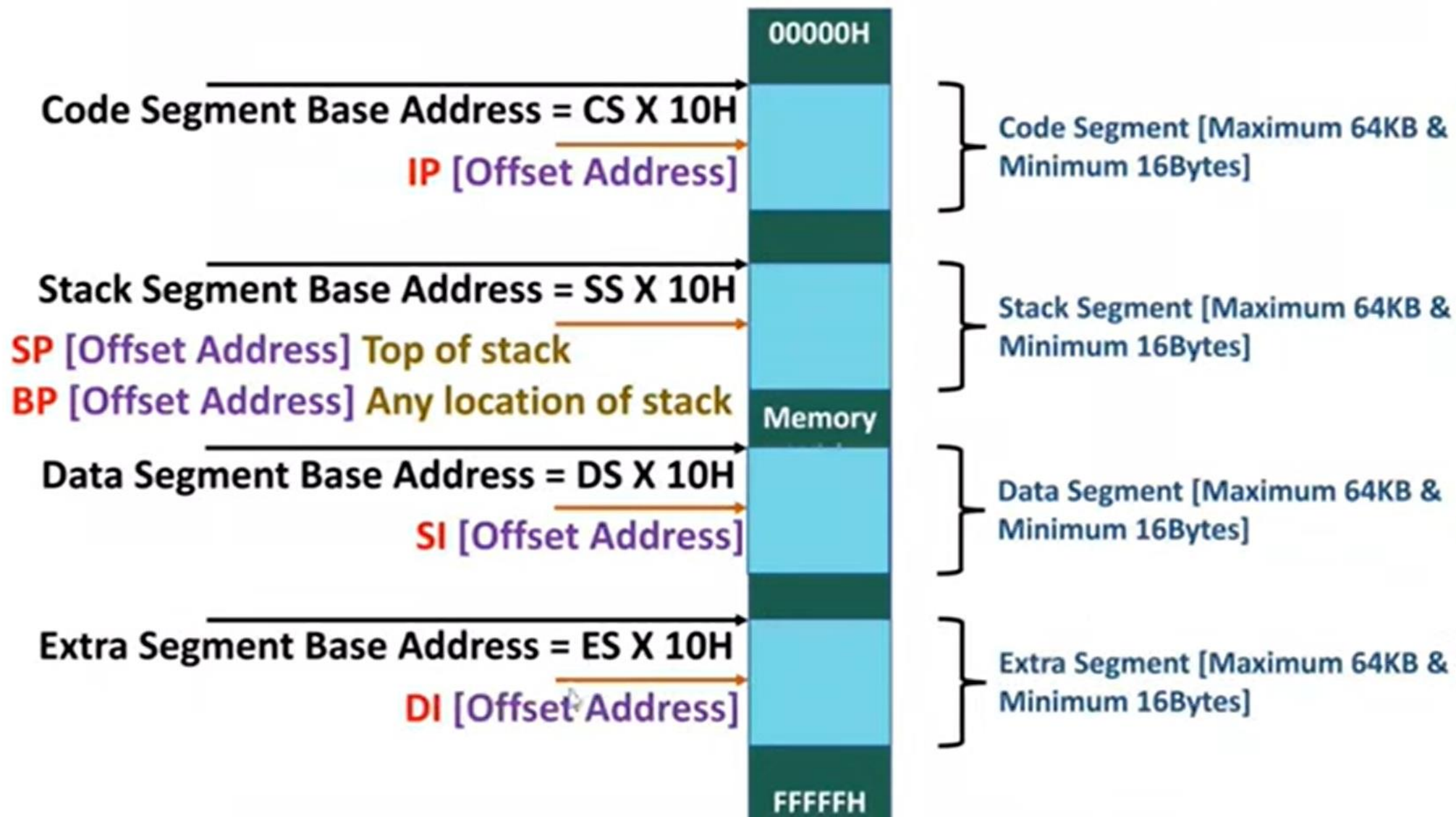
$$PA = 78EF0H \text{ to } 88EEFH$$

- For Minimum range

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

$$PA = 78EFH \times 10H + 0000H \text{ to } 000FH$$

Memory Segmentation of 8086



Advantages of Memory Segmentation of 8086

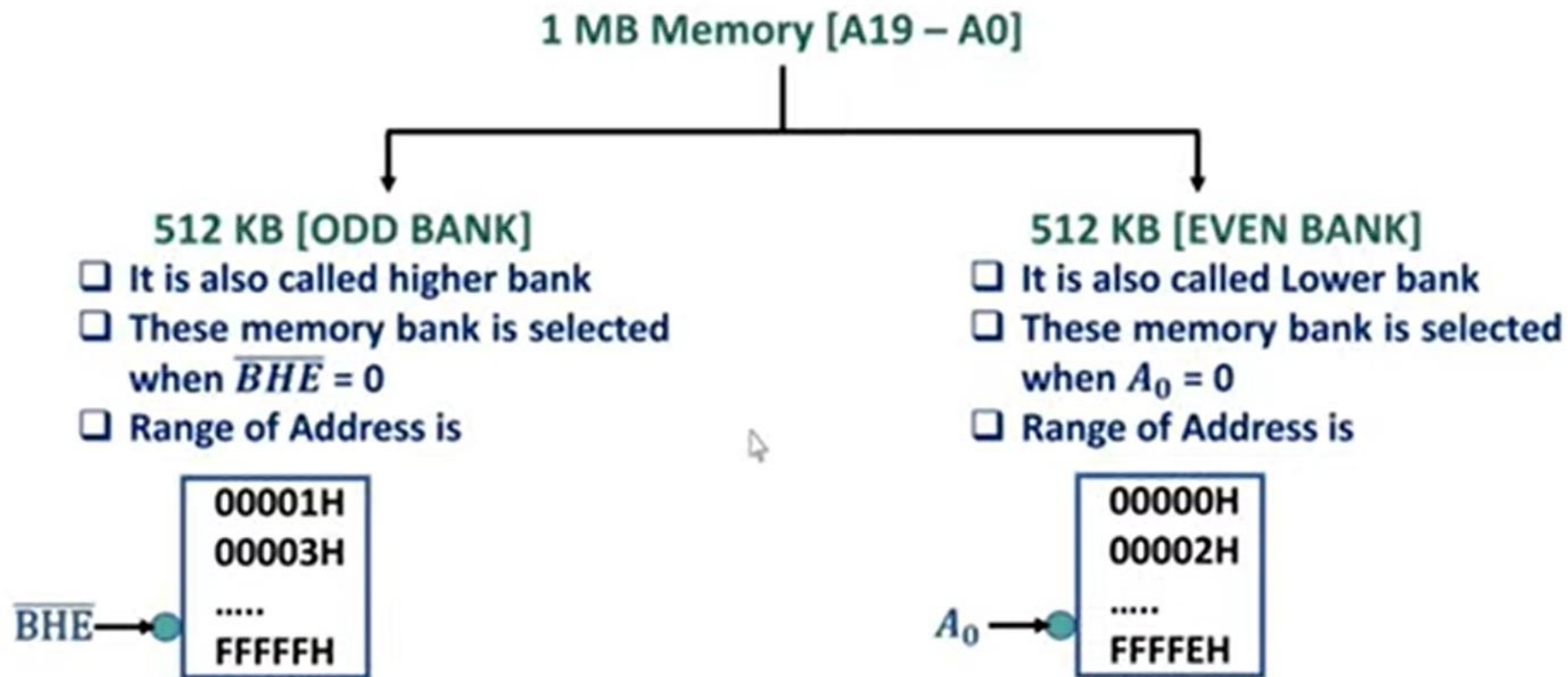
- ❖ We can program 8086 with 16bits addressing.
- ❖ Due to Memory segmentation, 8086 becomes backward compatible with 8085.
- ❖ Change in One segment does not effect the other segment.
- ❖ Memory management gets easier.
- ❖ Debugging gets easier.
- ❖ It avoids memory overlap with different segments.



Memory Banking in 8086

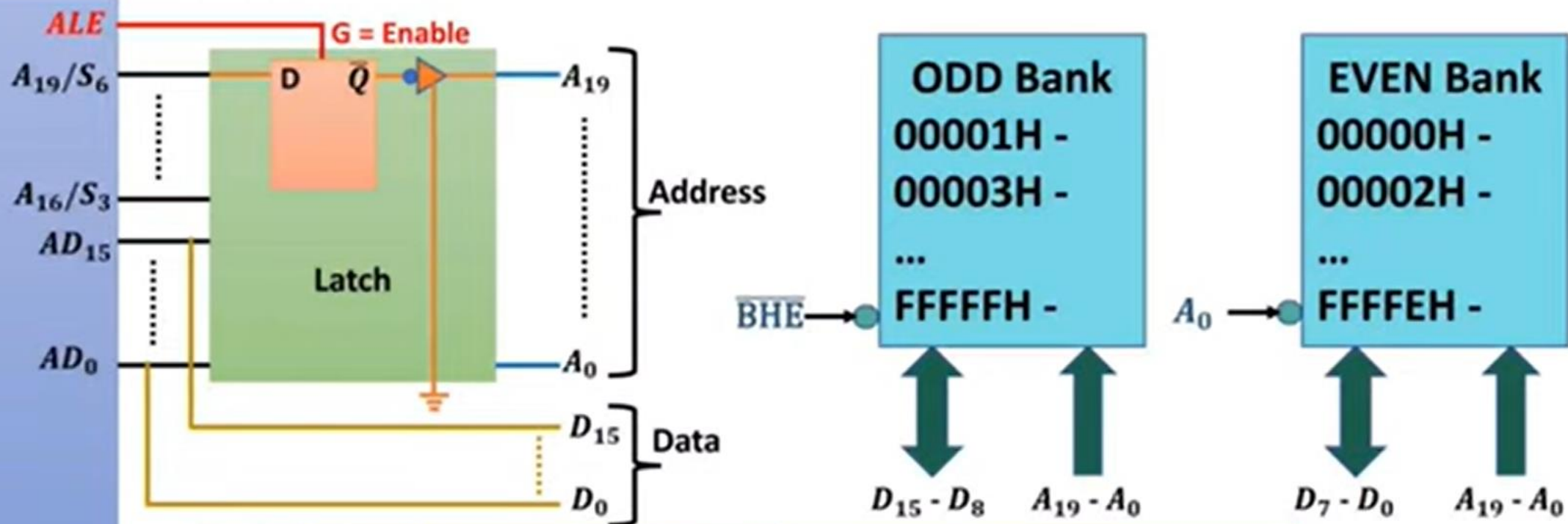
- ❖ 8086 has 16 bits of Data lines, so it should access 16 bits (2 Bytes) in one machine cycle.
- ❖ To perform 2 Bytes data transfer, So it needs to read 2 memory locations as one memory location has only one byte of data.
- ❖ If both of these memory locations are there in same memory then it can not 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 in 8086



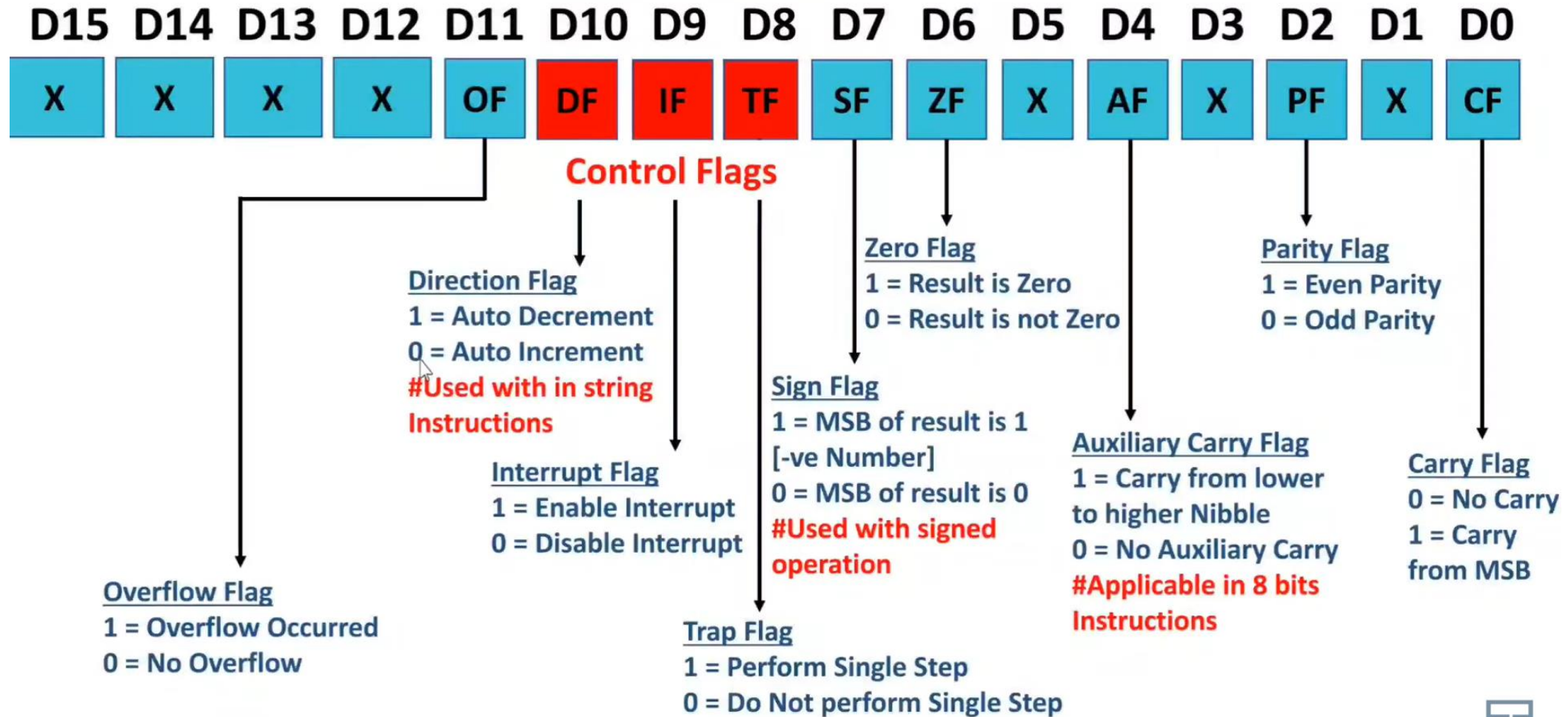
Memory Banking in 8086

μP
8086

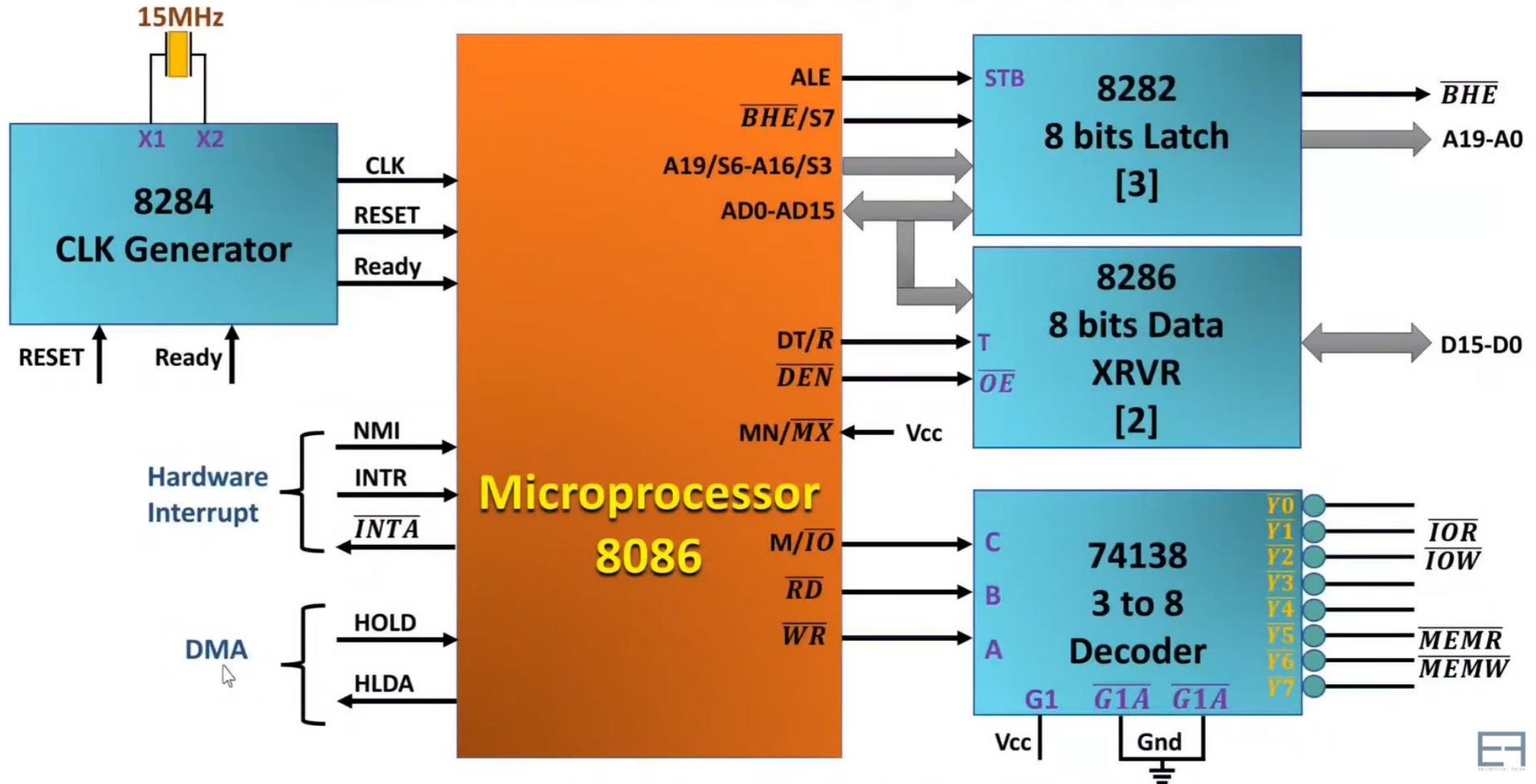


A_0	$\overline{BHE}$	Machine Cycle	Data	Operation
0	0	1	$D_{15}-D_0$	Read a Word from Even Add
0	1	1	D_7-D_0	Read a Byte from Even Add
1	0	1	$D_{15}-D_8$	Read a Byte from Odd Add
1	0	2	$D_{15}-D_8$	Read a Word from Odd Add
0	1		D_7-D_0	

Flag Register of 8086



Minimum Mode of 8086

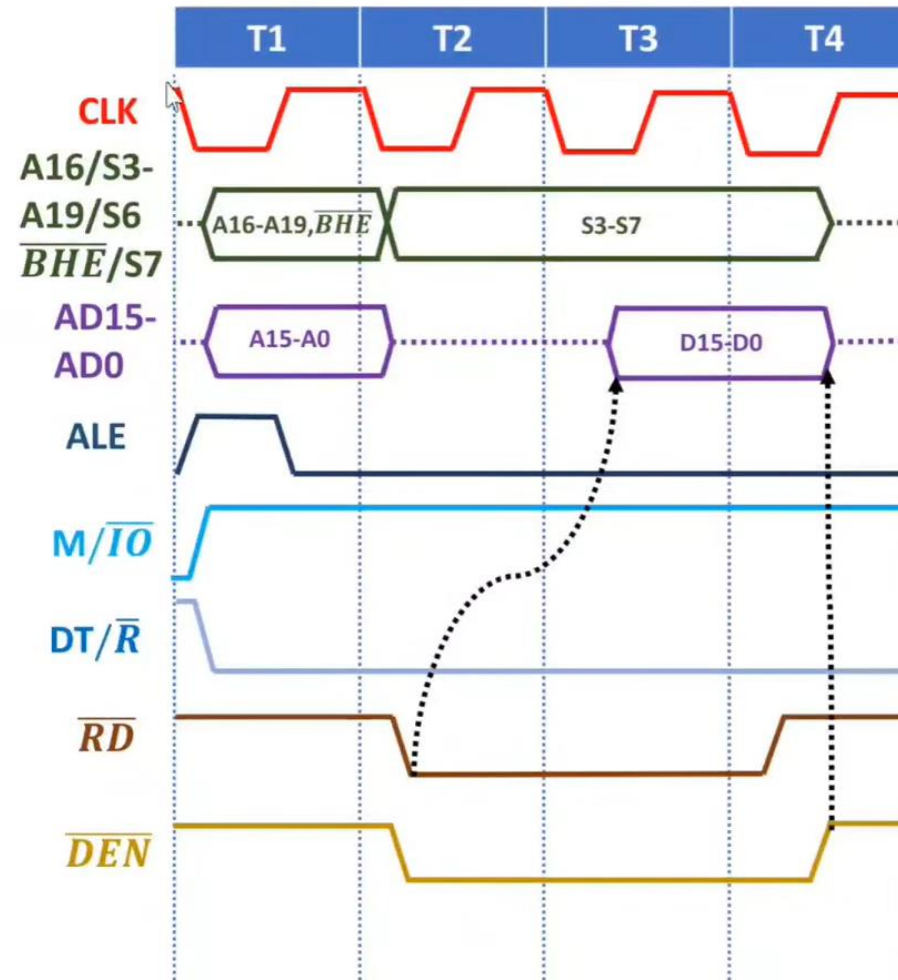


Minimum Mode Timing Diagram of 8086

□ By using three terminals $\overline{RD}$, $\overline{WR}$ & $M/\overline{IO}$, we can execute four Machine Cycles.

- Memory Read [$\overline{RD} = 0$, $\overline{WR} = 1$ & $M/\overline{IO} = 1$]
- Memory Write [$\overline{RD} = 1$, $\overline{WR} = 0$ & $M/\overline{IO} = 1$]
- IO Read [$\overline{RD} = 0$, $\overline{WR} = 1$ & $M/\overline{IO} = 0$]
- IO Write [$\overline{RD} = 1$, $\overline{WR} = 0$ & $M/\overline{IO} = 0$]

❖ Memory Read Timing Diagram



Maximum Mode in 8086

