# <u>Segmented Memory & Interleaved</u> <u>Memory Architecture of 8086</u>

# 1. Segmented Memory Architecture

# What is Segmented Memory?

Segmented memory divides the **1 MB physical address space** into smaller, manageable segments of **64 KB each**. This allows the processor to access a large memory space (20-bit addresses) using 16-bit segment and offset values.

```
1. Segmented Memory Diagram

You can visualize it as four distinct segments in the 1 MB memory:

diff

Copy code

Copy code

Data Segment (CS) |

Data Segment (DS) |

Stack Segment (SS) |

Extra Segment (ES) |

Each segment is 64 KB in size and starts at an address multiple of 16 (0x10).
```

# Why Segmented Memory?

- 1. Memory Organization:
  - o Easier organization and management of memory.
  - Logical separation of different types of data (e.g., code, data, stack, and extra).
- 2. Efficient Use of 16-bit Registers:

 8086 uses 16-bit registers for addresses, so segmentation extends the addressable memory space to 20 bits.

### 3. Program Modularity:

- Programs can be divided into smaller modules (code, data, etc.).
- Helps in multitasking and memory protection.

# Segment Registers and Address Calculation

- Segment Registers in 8086 hold the starting address (base) of a segment:
  - CS (Code Segment): Points to the instructions (code).
  - DS (Data Segment): Points to the data used by the program.
  - SS (Stack Segment): Points to the stack area for temporary storage.
  - ES (Extra Segment): Used for additional data or string operations.
- Physical Address Calculation:

Physical Address Calculation: The physical address is calculated by:  $\text{Physical Address} = (\text{Segment Address} \times 16) + \text{Offset}$ 

- Segment Address: Value in the segment register (16 bits).
- Offset: A 16-bit value that provides the relative address within the segment.

# Example:

#### Example:

If CS = 0x1234 and IP (Instruction Pointer) = 0x5678:

Physical Address =  $(0x1234 \times 16) + 0x5678 = 0x12340 + 0x5678 = 0x179B8$ 

#### In hexadecimal:

•  $0x1234 \times 16 = 0x12340$ 

#### Here's how:

- 1. Write **0x1234** in decimal for clarity: 0x1234 = 4660.
- 2. Multiply it by 16:  $4660 \times 16 = 74560$ .
- 3. Convert 74560 back to hexadecimal: 74560 = 0x12340.

So, the Base Address is 0x12340.

### In hexadecimal:

- 1. Write 0x12340 and 0x5678 in decimal:
  - 0x12340 = 74560
  - 0x5678 = 22136
- 2. Add them in decimal:

$$74560 + 22136 = 96696.$$

3. Convert the sum back to hexadecimal:

$$96696 = 0x179B8.$$

So, the **Physical Address** is **0x179B8**.

# Advantages of Segmented Memory

1. **Efficient Memory Usage**: Programs can reuse segments without altering the entire memory map.

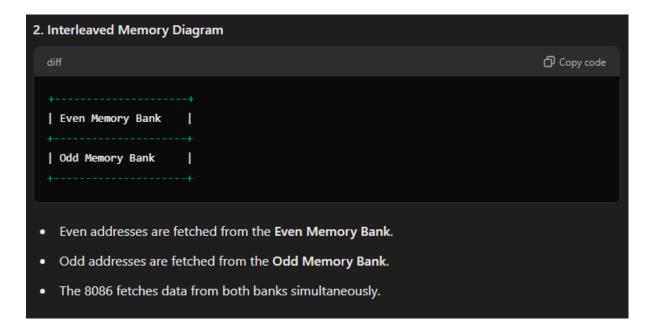
- 2. **Compatibility**: Allows running programs designed for smaller address spaces (e.g., 16-bit systems).
- 3. Scalability: Easily extendable for larger memory systems.

# 2. Interleaved Memory Architecture

# What is Interleaved Memory?

Interleaved memory splits memory into multiple banks that can be accessed simultaneously to increase speed. For example:

- Memory is divided into even and odd banks.
- The 8086 fetches **16-bit words** (2 bytes) using two memory banks simultaneously:
  - o The lower byte from the even bank.
  - o The higher byte from the odd bank.



### How Does It Work?

#### 1. Simultaneous Access:

- o The **8086 data bus** is 16 bits wide.
- Accesses even and odd memory addresses in parallel, reducing the time needed for memory operations.

### 2. Example:

- Physical address 0x0000 is fetched from the even bank.
- Physical address 0x0001 is fetched from the odd bank.
- o Together, they form a 16-bit word.

# **Advantages of Interleaved Memory**

#### 1. Improved Performance:

- o Reduces waiting time for memory fetches.
- Speeds up instruction execution by overlapping memory accesses.

# 2. Efficient Bandwidth Usage:

o Maximizes utilization of the memory bus.

#### 3. Better Parallelism:

Takes advantage of the 16-bit bus of the 8086.