UNIT 2

BASIC COMPUTER ARCHITECTURE

CONTENTS

- MICROPROCESSOR ARCHITECTURE AND OPERATIONS, MEMORY, I/O DEVICES, MEMORY AND I/O OPERATIONS,
- 8085 MICROPROCESSOR ARCHITECTURE
- ADDRESS, DATA AND CONTROL BUSES
- 8085 PIN FUNCTIONS
- DE-MULTIPLEXING OF BUSES
- GENERATION OF CONTROL SIGNALS

OBJECTIVES

- To explain the various functions of the registers in the 8085 programming model.
- Define the term flag and explain how the flags are affected.
- Explain the term operation code (opcode) and the operand, and illustrate these terms by writing instructions.
- Classify the instructions in terms of their word size and specify the number of memory registers required to store the instructions in memory.
- Define and explain the term addressing mode.

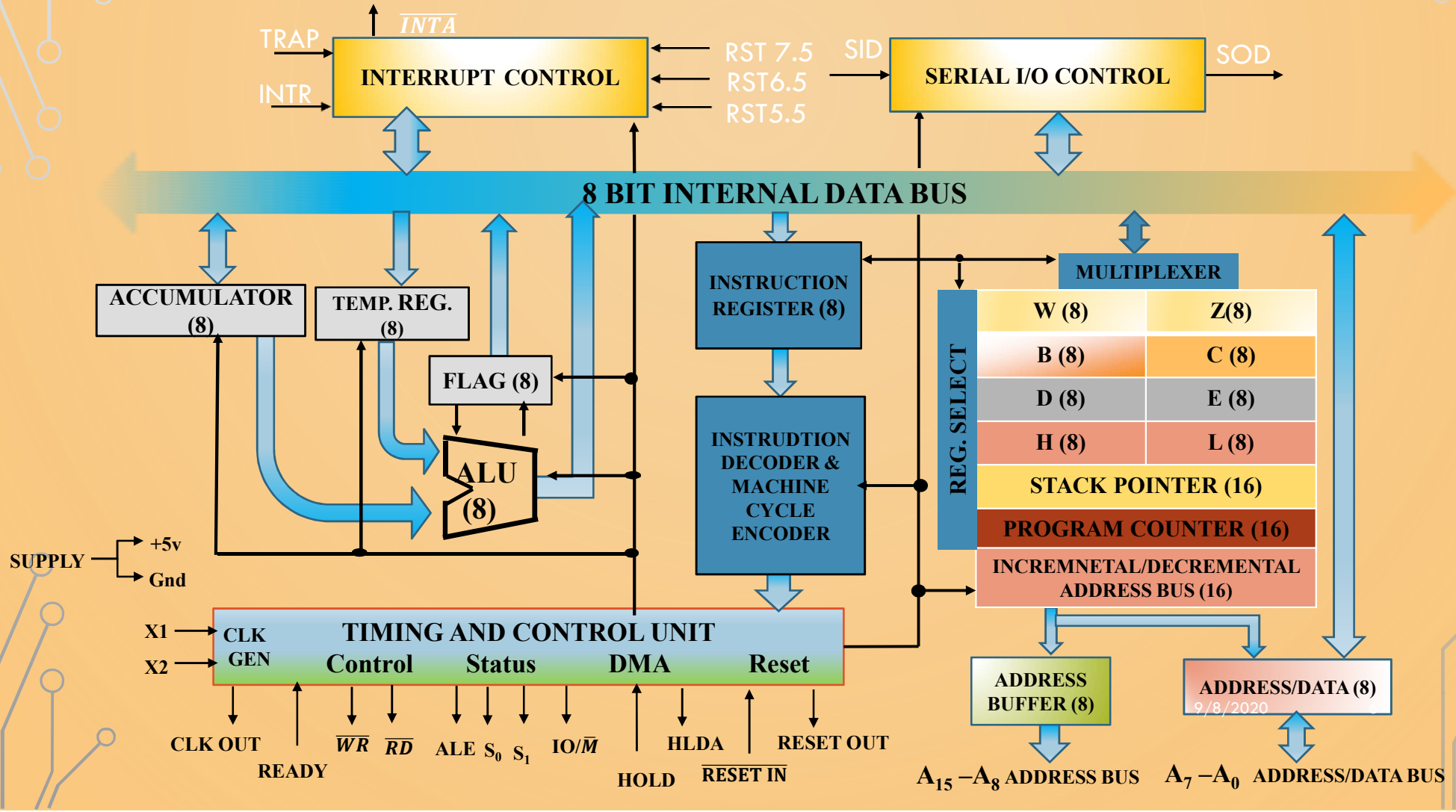
INTERNAL ARCHITECTURE OF AN 8-BIT 8085 MICROPROCESSOR AND ITS REGISTERS

- The Intel 8085A is a complete 8-bit parallel central processing unit.
- The main components are:-
  - Array of registers
  - The arithmetic and logic unit
  - The encoder/decoder
  - The timing and control unit
- All linked by internal data bus.

FEATURES OF 8085

- The main features of 8085 are:
  - It is an 8 bit processor.
  - It is a single chip N-MOS device with 40 pins.
  - It has multiplexed address and data bus.(AD0-AD7).
  - It works on 5 Volt dc power supply.
  - The maximum clock frequency is 3 MHz while minimum frequency is 500kHz.
  - It provides 74 instructions with 5 different addressing modes.
  - It provides 16 address lines so it can access 2^16 =64K bytes of memory.
  - It generates 8 bit I/O address so it can access 2^8=256 input ports.

INTERNAL ARCHITECTURE OF 8085 MICROPROCESSOR



- **The microprocessor is a clock driven semiconductor device consisting of electronic logic circuits manufactured either by using LSI or VLSI technique.**
- **Capable of performing various computing functions and making decisions to change the sequence of program execution.**
- **Can be divided into three segments for the sack of clarity; arithmetic unit(ALU), register array and control unit.**

1. ARITHMETIC & LOGIC UNIT (ALU)

- Performs the arithmetic/logical computing functions.
- Includes the accumulator, registers, the arithmetic and logic circuits and five flags and two temporary registers.
- Temporary registers are used to hold data during an arithmetic/logic operations.
- Result is stored in accumulator; the flags are set/reset according to the result of the operation.

2. ACCUMULATOR (Register A)

- 8-bit register that is part of ALU and accessible to users.
- Used to store 8-bit data and to perform arithmetic/logic operations.
- 8085 is called Accumulator based Microprocessor as of the two operands for all operations and result is also stored in it.
- When data is read from the input port, it is first placed in Accumulator and when data is sent to output port, it must be first placed in Accumulator.

3. Temporary Registers (W& Z)

- 8-bit registers and not accessible to programmers.
- Data is placed in it for short period of time during execution.

4. INSTRUCTION REGISTERS (IR)

- 8-bit register not accessible to programmers.
- Receives the operation code of instruction from the internal data bus and passes it to instruction decoder.
- Decoder decodes the instruction so that what operation is to be performed by the Microprocessor.

5. REGISTER ARRAYS (B,C,D,E, H and L)

- General purpose registers.
- Each 8-bit registers accessible to programmers.
- Data are stored on it during program execution.
- Can be used individually as 8-bit registers and as 16-bit registers in pair forming BC,DE, & HL.
- Data can be directly added or transformed to one another.
- Their content can be incremented or decrement and combined logically with the content of accumulator.

6. STACK POINTER (SP)

- 16-bit register used as memory pointer.
- Points to the memory location in R/W memory, called the stack.
- Also called LIFO queue.
- The beginning of the stack is defined by loading the 16-bit address in the stack pointer.

7. PROGRAM COUNTER (PC)

- 16-bit register that holds address of the next instruction to be executed.
- As microprocessor begins to execute a program , the memory location of first instruction is placed in PC.
- PC maintains the sequence of execution of instructions.
- Automatically incremented by one to point the next memory location when a byte is being fetched; i.e. it keeps the record of program by counting the memory address, hence name PC.

8. FLAGS

- 5 flip-flops in 8085, each holding the status of different states separately known as flag register.

D7	D6	D5	D4	D3	D2	D1	D0
S	Z		AC		P		CY

- Each flip-flop is called flags.
- 8085 can set or reset each flags depending on the type of operation.
- The flags are:
  - Sign (S)
  - Zero(Z)
  - Auxiliary Carry (AC)
  - Parity (P)
  - Carry (CY)

FLAGS Cond…

- The state of flags indicate the result of arithmetic/ logic operation, which in turns used for decision making processes.
- Carry (CY):
  - Stores the carry or borrow from one byte to another.
  - It is set (CY=1), when the last arithmetic operation generates carry or borrow, otherwise reset (CY=0).
- Zero (Z):
  - Z=1, if the result of last operation of ALU is zero, otherwise, Z=0.
  - Often used in loop control and in searching for particular data value.

FLAGS Cond…

- **Sign (S):**
  - **After the execution of an arithmetic/logic operation, if bit D7 (MSB) of the result (usually accumulator) is 1, the sign flag is set.**
  - **Used with signed numbers.**
  - **In a given byte, if D7 is 1, it is viewed as *negative*; if it is 0, it is viewed as *positive*.**
  - **This flag is irrelevant to the operation of unsigned numbers.**
- **Parity (P):**
  - **After arithmetic/logic operation, if the result has even number of 1's (even parity), the flag is set, otherwise reset.**

9. TIMING & CONTROL UNIT

- **Synchronizes all the operations with the clock.**
- **Generates the control signals necessary for communication between the microprocessor and peripherals.**
- **Control signals are similar to sync pulse in an *oscilloscope*.**
- **RD(bar) and WR(bar) signals are sync pulses indicating the presence of data on the data bus.**

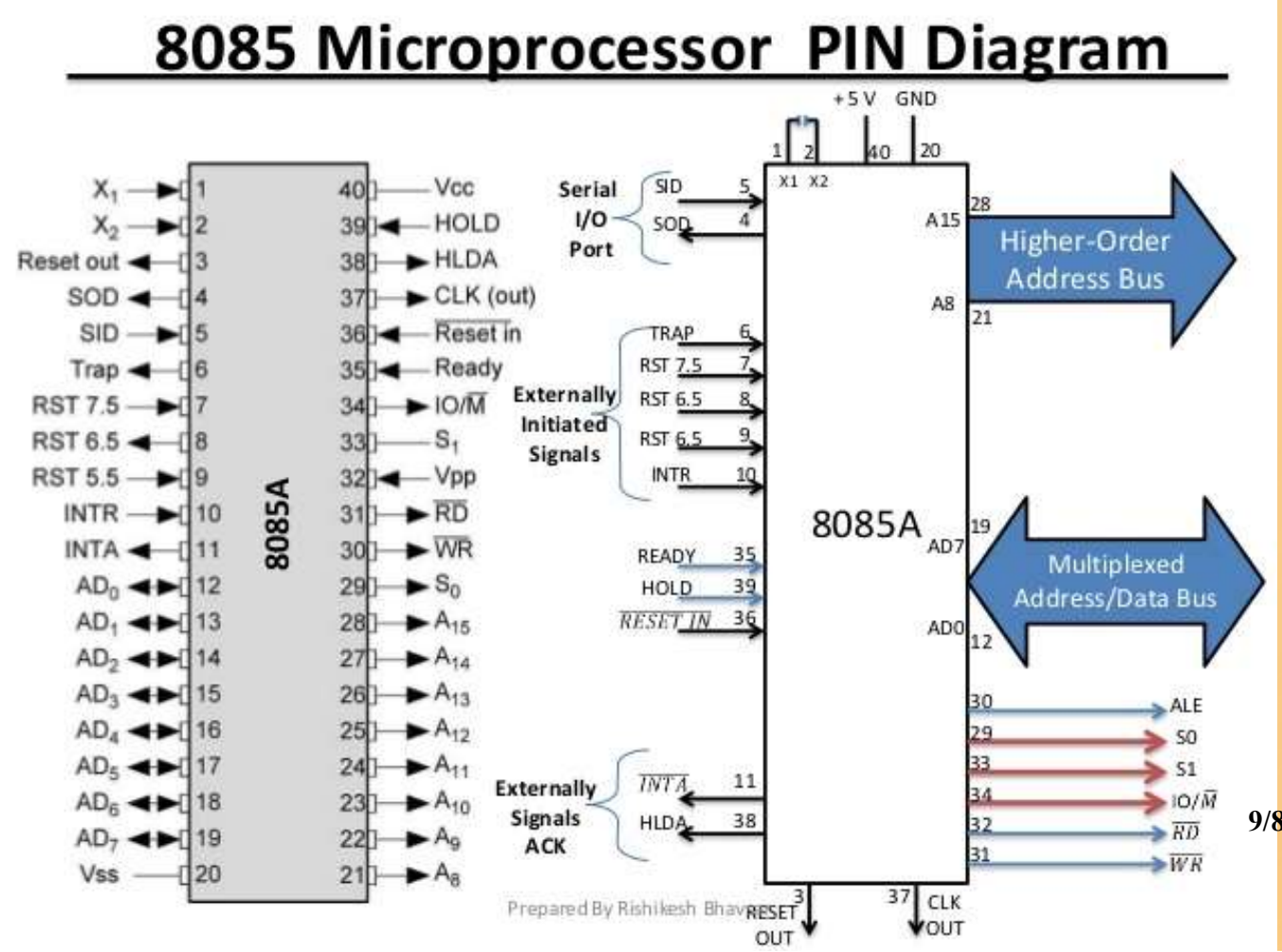
10. INTERRUPT CONTROL

- **Mainly 5 types of interrupt:**
  - **INTR**
  - **TRAP**
  - **RST5.5**
  - **RST6.5**
  - **RST7.5**

11. SERIAL I/O

- Two serial I/O control signals:
  - Serial In Data (SID) and
  - Serial Out Data (SOD)
- Used to implement the serial data communication.

8085 PIN DESCRIPTION



8085 is 40 pin IC, DIP package. The total pin can be categorized to six groups:

I. Address Bus

II. Multiplexed Address/Data Bus

III. Control and status signal

IV. Power supply and clock signal

V. Interrupt and externally initiated signals

VI. Serial I/O ports

1. ADDRESS BUS

- 16 signal lines that are used as the address bus; however, these lines are split into two segments A_{15}-A_8 and AD_7-AD_0.
- A_{15}-A_8 are unidirectional and carries higher order address and the lower order AD_7-AD_0 are multiplexed and bidirectional.

2. MULTIPLEXED ADDRESS/DATA BUS

- 8-bit data bus.
- Multiplexed bidirectional AD_7-AD_0 .
- These multiplexed lines are de-multiplexed to work as address bus and data bus separately using Address Latch Enable (ALE).
- If ALE=1, AD_7-AD_0 acts as address bus, otherwise it acts as data bus. By default, it acts as data bus.

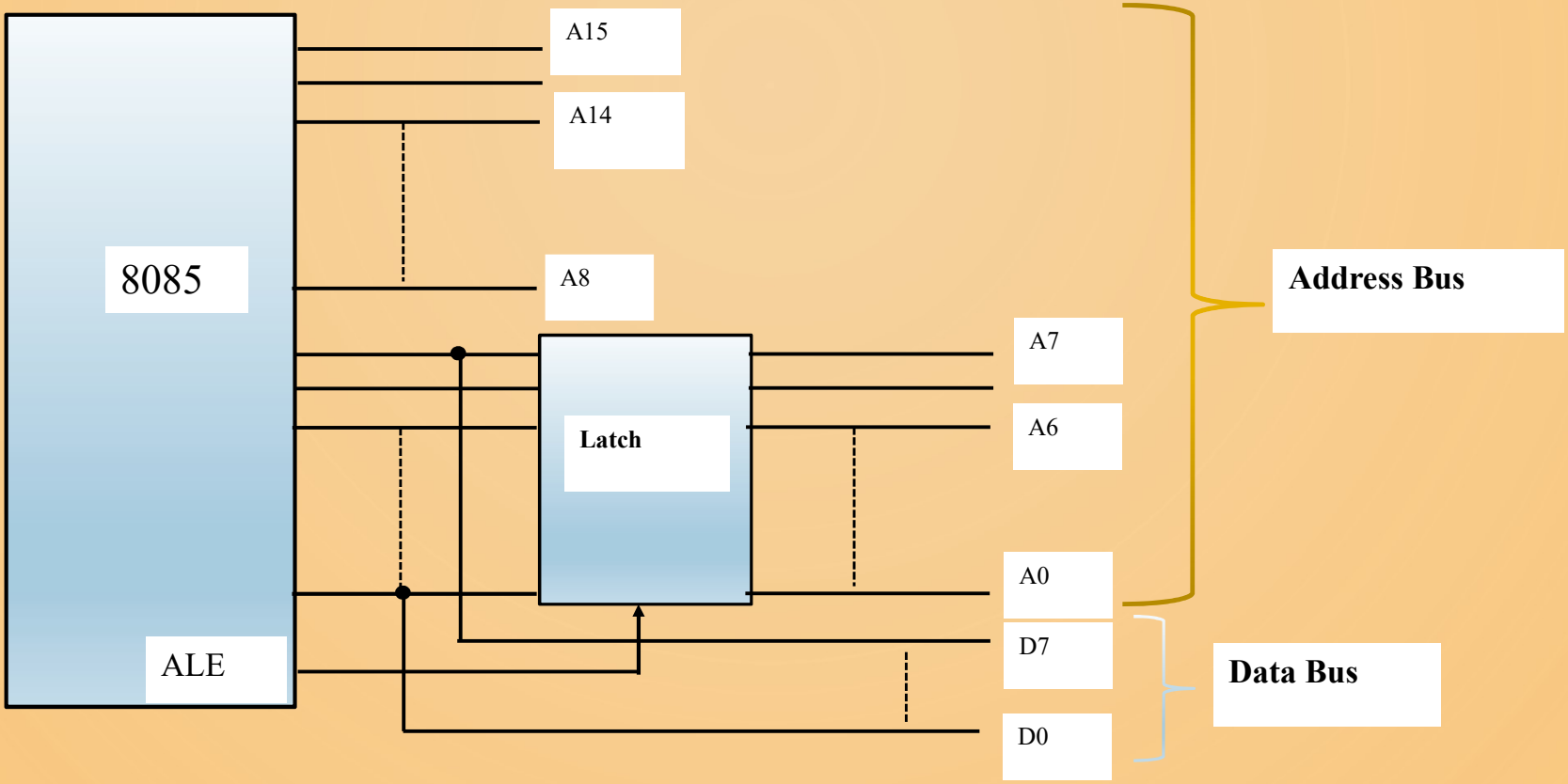
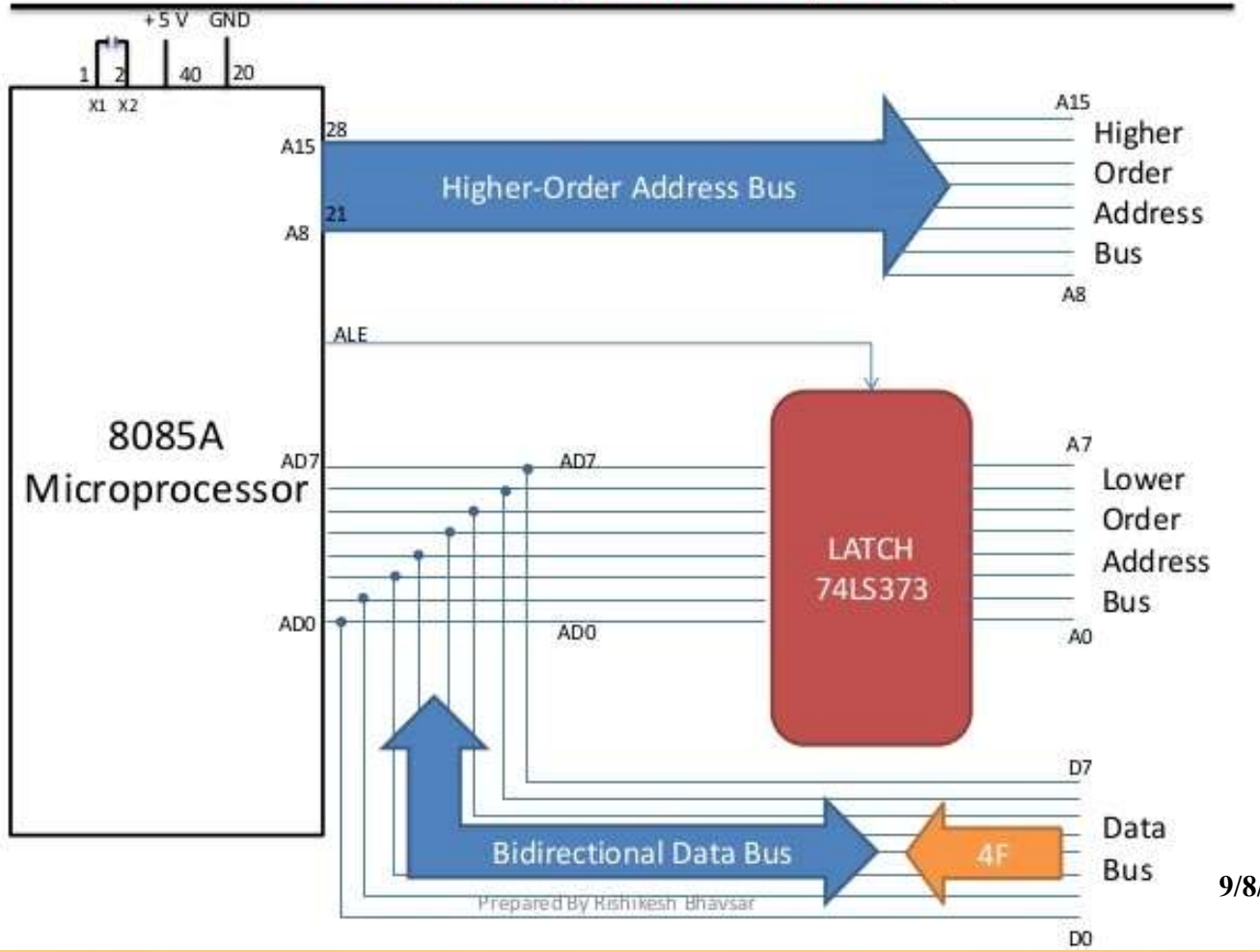


Fig: time multiplexed address and data bus

De-multiplexing the Bus AD_7 to AD_0



3. CONTROL AND STATUS SIGNAL

- This group of signals includes two control signals ($\overline{RD}$ and $\overline{WR}$).
- Three status signals (IO/$\overline{M}$, S_1, S_0) to identify the nature of the operation, one special signal (ALE) to indicate the beginning of operation.

IO/M'	S_1	S_0	Operation
Z	0	0	HALT
0	0	1	Memory Write
0	1	0	Memory Read
1	0	1	I/O Write
1	1	0	I/O Read
0	1	1	Opcode Fetch

4. POWER SUPPLY AND CLOCK SIGNAL

- **VCC: +5V power supply.**
- **VSS: Ground reference.**
- **X_1, X_2: A crystal (or RC, LC network) is connected at these two pins. The frequency is initially divided by 2; there for to operate a system at 3 MHz, the crystal should have frequency of 6 MHz**
- **CLK-clock output: This signal can be as a system clock for other devices.**

5. INTERRUPT AND EXTERNALLY INITIATED SIGNALS

- **The 8085 has 5 interrupt signals that can be used to interrupt a program execution.**
  - **I. INTR (input)**
  - ***II.*** **$\overline{\text{INTA}}$ (output)**
  - **III. RST 7.5, 6.5, 5.5 (inputs)**
  - **IV. TRAP (input)**
  - **V. HOLD (input)**
  - **VI. HLDA (output)**
  - **VII. READY (Input)**
  - **VIII. $\overline{\text{RESETIN}}$**
  - **IX. RESET OUT**

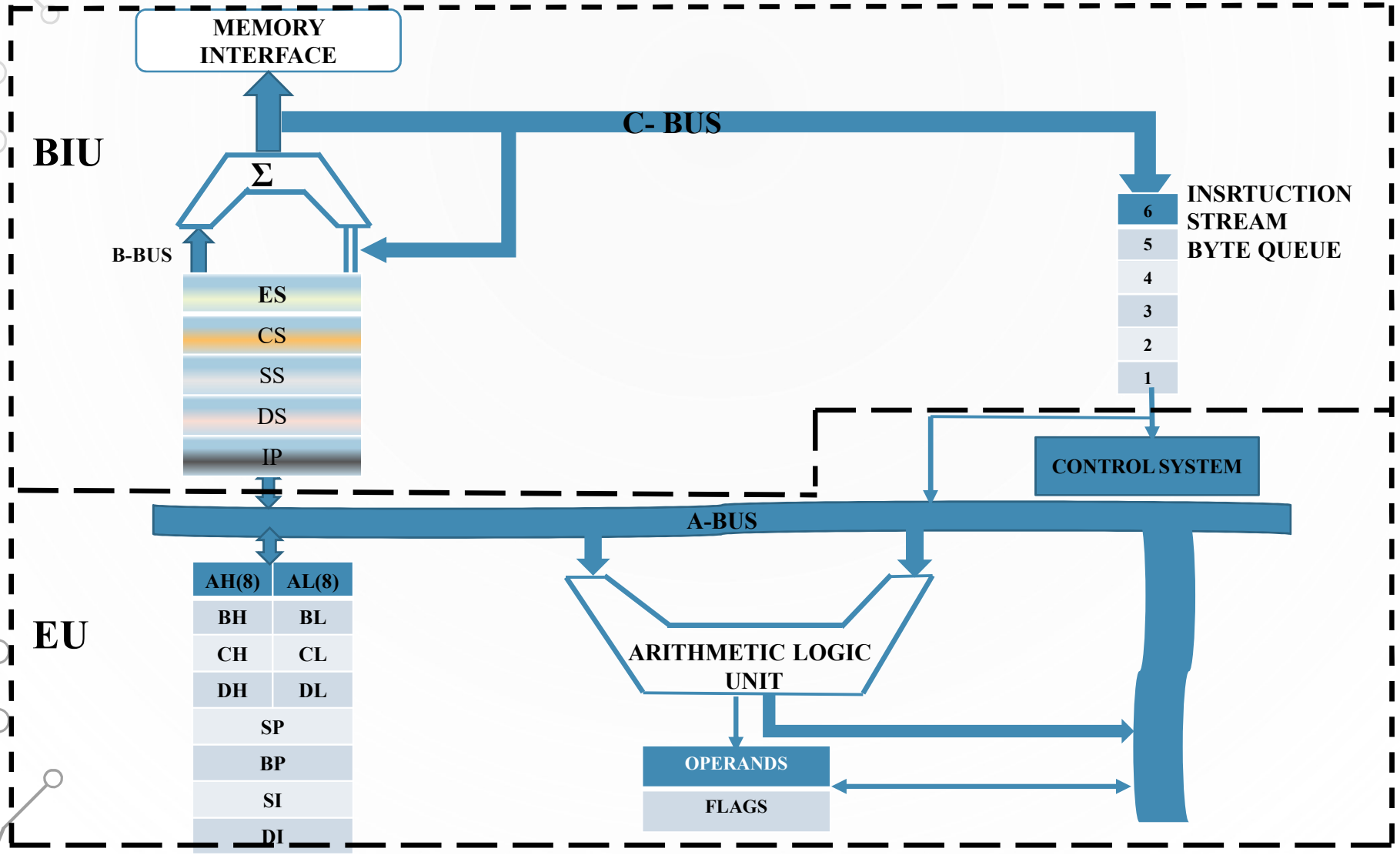
6. SERIAL I/O PORTS

- Two signals to implement the serial transmission: SID (Serial Input Data) and SOD (Serial Output Data).
- In serial transmission, data bits are sent over a single line, one bit at a time, such as the transmission over telephone lines.

INTEL 8086 MICROPROCESSOR

- **8086 Microprocessor Functional Block Diagram:**
  - **16-bit microprocessor (i.e. 16 bit data processing capability).**
  - **16-bit implies that its ALU, its internal registers, and most of its instructions are intended to work with 16 bit binary data.**
  - **16 bit data bus, so it can read data from or write data to memory and ports either 16 bits or 8 bits at a time.**
  - **20 bit wide address bus (can address any one of 2^20, or 1048576 memory locations).**
  - **CPU is divided into 2 independent functional parts:**
    1. **Bus Interface Unit (BIU)**
    2. **Execution Unit (EU)**

8086 MICROPROCESSOR INTERNAL ARCHITECTURE



BIU: It handles all transfers of data and addresses on the buses for the execution unit.

- Sends out addresses
- Fetches instructions from memory
- Read / write data from/to ports and memory i.e. handles all transfers of data and addresses on the busses.

EU

- Tells BIU where to fetch instructions or data from
- Decodes instructions
- Execute instructions

Execution Unit (EU)

1. **Instruction Decoder & ALU:**

- Decoder in the EU translates instructions fetched from the memory into a series of actions which the EU carries out.
- 16-bit ALU in the EU performs actions such as addition, subtraction, AND, OR, XOR, increment, decrement etc.

2. General purpose Registers (GPRs):

- **8 GPRs AH, AL (Accumulator), BH, BL, CH, CL, DH, DL are used to store 8 bit data.**
- **Used individually for the temporary storage of data.**
- **Used together (as register pair) to store 16-bit data words.**
- **Acceptable register pairs are:**
  - AH-AL pair AX register
  - BH-BL pair BX register (to store the 16-bit data as well as the base address of the memory location)
  - CH-CL pair CX register (to store 16-bit data and can be used as counter register for some instructions like loop)
  - DH-DL pair DX register (to store 16-bit data and also used to hold the result of 16-bit data multiplication and division operation)

3. FLAG REGISTER:

- It is a 16-bit register.
- 9-bit are used as different flags, remaining bits unused

				OF	DF	IF	TF	SF	ZF		AF		PF		CF

Fig: 16-bit flag register

Out of 9-flags, 6 are conditional (status) flags and three are control flags.

Conditional flags:

- set or reset by the EU on the basis of the results of some arithmetic or logic operation.
- 8086 instructions check these flags to determine which of two alternative actions should be done in executing the instructions.
  - **OF (Overflow flag):** is set if there is an arithmetic overflow, i.e. the size of the result exceeds the capacity of the destination location.
  - **SF (Sign flag):** is set if the MSB of the result is 1.
  - **ZF (Zero flag):** is set if the result is zero.
  - **AF (Auxiliary carry flag):** is set if there is carry from lower nibble to upper nibble or from lower byte to upper byte.
  - **PF (Parity flag):** is set if the result has even parity.
  - **CF (Carry flag):** is set if there is carry from addition or borrow from subtraction.

Control flags:

- They are set using certain instructions.
- They are used to control certain operations of the processor.
  - **TF (Trap flag):** for single stepping through the program.
  - **IF (Interrupt flag):** to allow or prohibit the interruption of a program.
  - **DF (Direction flag):** Used with string instructions.

4. POINTER REGISTERS:

- **SP (Stack Pointer)**
- **BP (Base pointer)**
  - are used to access data in the stack segment.
  - SP is used as offset from current Stack Segment during execution of instruction that involve stack.
  - SP is automatically updated.
  - BP contains offset address and is utilized in based addressing mode.
- **Overall, these are used to hold the offset address of the stack address.**

5. INDEX REGISTERS:

- **SI (Source Index)**
- **DI (Destination index)**
  - **Both 16-bit long.**
  - **used for temporary storage of data similarly as the general purpose registers.**
  - **Specially used to hold the 16-bit offset of the data *word*.**

SI and DI are used to hold the offset address of the data segment and extra segment memory respectively.

BUS INTERFACE UNIT

- MAINLY TWO PARTS:
  1. Instruction queue
  2. Segment registers

1. INSTRUCTION QUEUE

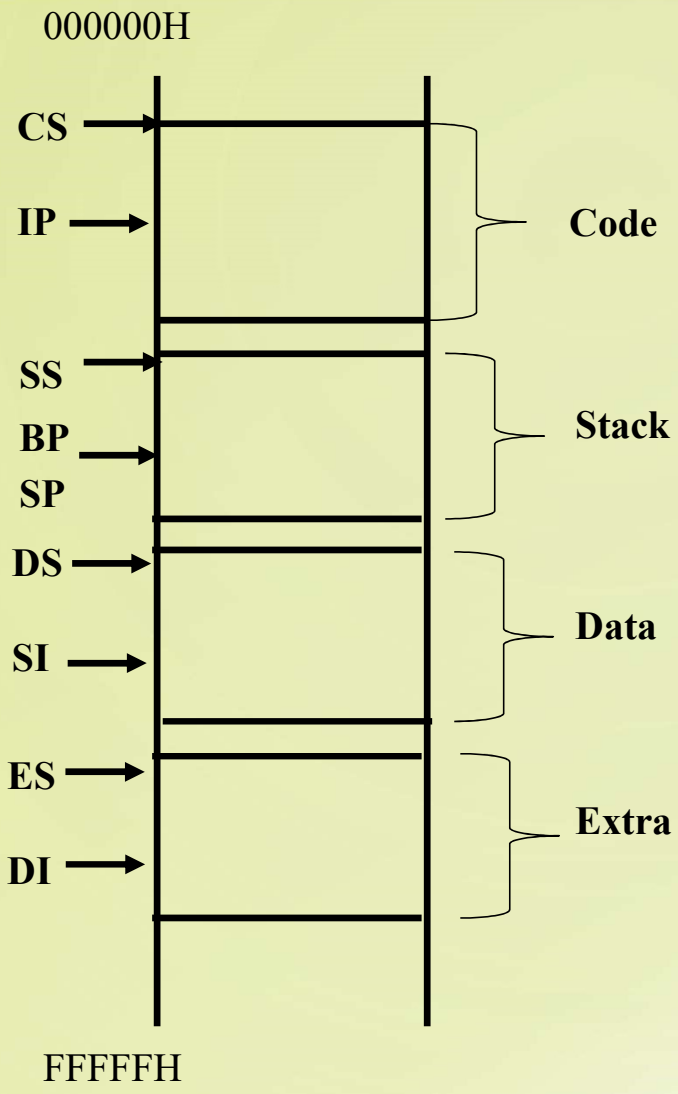
- **BIU pre-fetches up to 6- instructions bytes to be executed and places them in QUEUE.**
- **EU just picks up the fetched instruction byte from the QUEUE.**
- **This improves the overall speed of the system.**
- **The BIU stores these pre-fetched bytes in a first-in-first-out (FIFO) register set called a queue.**
- **Fetching the next instruction while the current instruction executes is called *pipelining.***

2. SEGMENT REGISTERS

- **The BIU contains a dedicated address, which is used to produce the 20 bit address.**
- **The bus control logic of the BIU generates all the bus control signals, such as the READ and WRITE signals, for memory and I/O.**
- **The BIU also has *four 16-bit* segments registers:**
  - **Code segment(CS):** holds the upper 16-bits of the starting addresses of the segment from which BIU is currently fetching instruction code bytes.
  - **Stack segment(SS):** store addresses and data while subprogram executes
  - **Extra segment(ES):** store upper 16-bits of starting addresses of two memory segments that are used for data.
  - **Data segment(DS):** store upper 16-bits of starting addresses of two memory segments that are used for data.

MEMORY SEGMENTATION

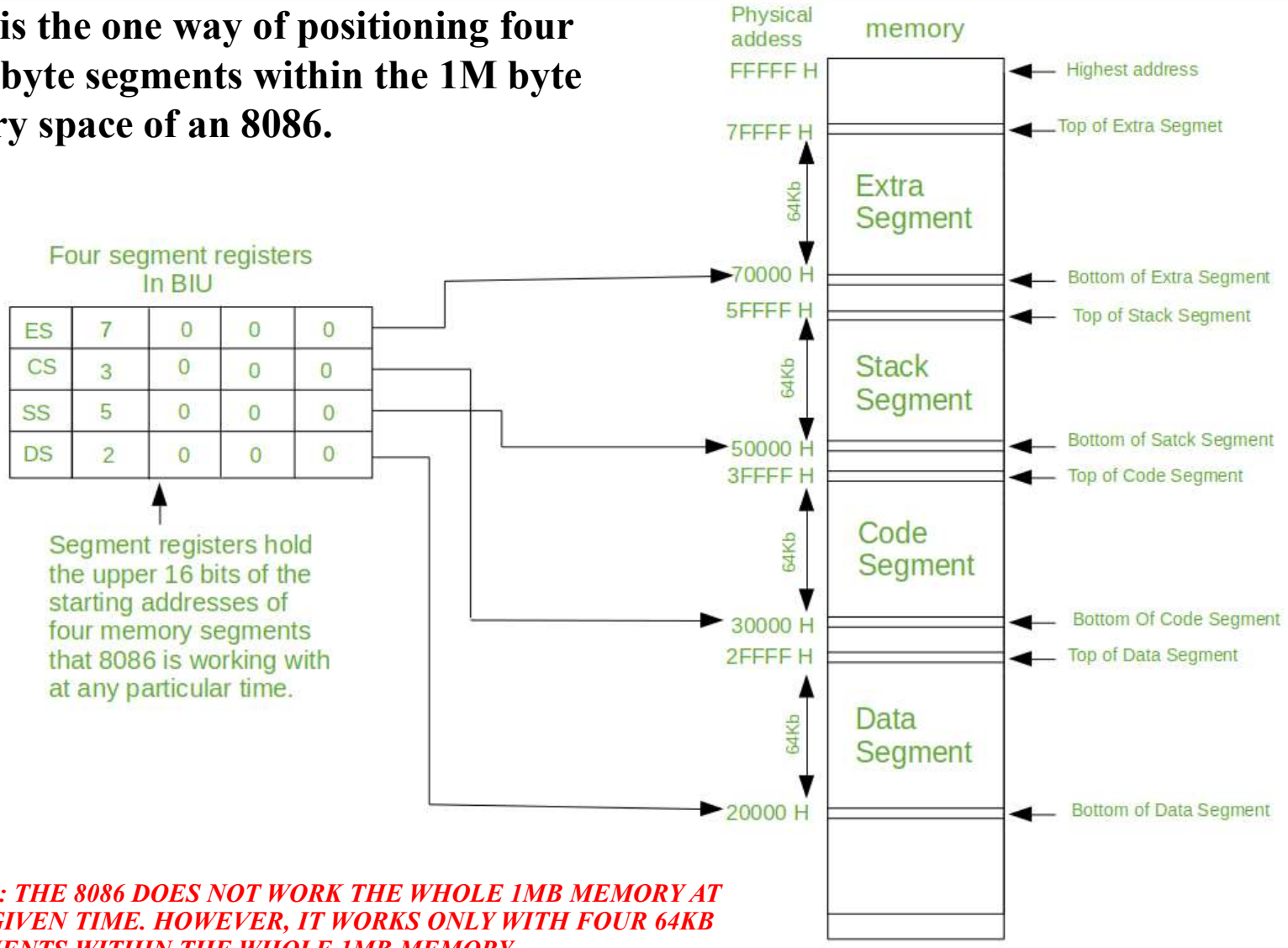
- address lines in 8086 is 20, BIU will send 20bit address, so as to access one of the 1MB memory locations.
- A segment is a logical unit of memory that may be up to 64 kilobytes long.
- Each segment is made up of contiguous memory locations.
- It is an independent, separately addressable unit.
- Starting address will always be changing and will not be fixed.



2^20=1MB
PA=Seg. Add.*10H+Offset Add.

Assume:
CS= 154EH
IP= 4308H
Then, PA=154EH*10H+4308H
=197E8H

- **Below is the one way of positioning four 64 kilobyte segments within the 1M byte memory space of an 8086.**



NOTE: THE 8086 DOES NOT WORK THE WHOLE 1MB MEMORY AT ANY GIVEN TIME. HOWEVER, IT WORKS ONLY WITH FOUR 64KB SEGMENTS WITHIN THE WHOLE 1MB MEMORY.

Types Of Segmentation:

1. **Overlapping Segment:**

A segment starts at a particular address and its maximum size can go up to 64kilobytes. But if another segment starts before this 64kilobytes location of the first segment, then the two are said to be *Overlapping Segment.*

2. **Non-Overlapped Segment:**

A segment starts at a particular address and its maximum size can go up to 64kilobytes. But if another segment starts after this 64kilobytes location of the first segment, then the two segments are said to be *Non-Overlapped Segment.*

Rules of Segmentation:

Segmentation process follows some rules as follows:

1. The starting address of a segment should be such that it can be evenly divided by 16.
2. Minimum size of a segment can be 16 bytes and the maximum can be 64 kB.

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

Advantages of the Segmentation:

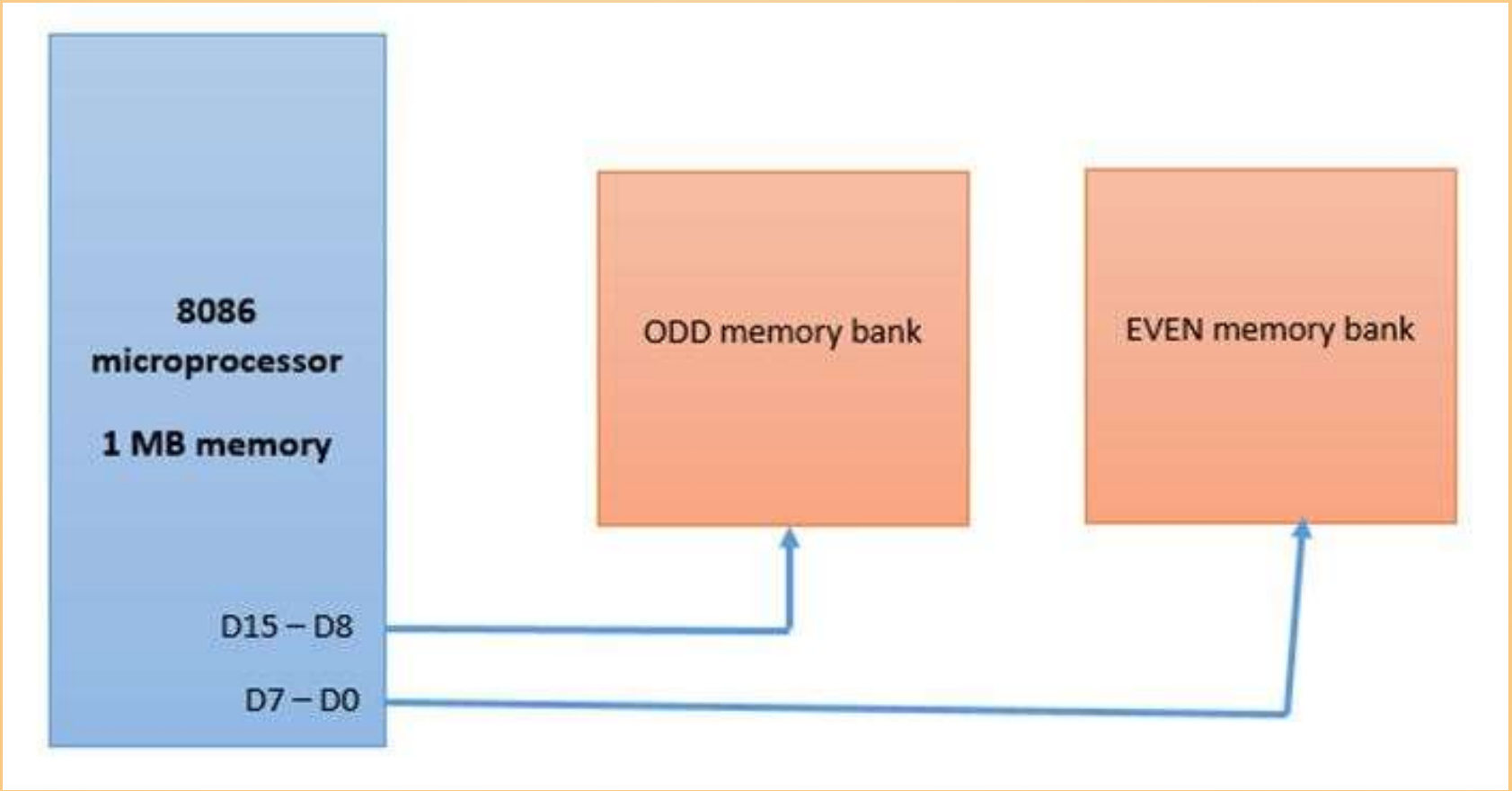
The main advantages of segmentation are as follows:

- It provides a powerful memory management mechanism.
- Data related or stack related operations can be performed in different segments.
- Code related operation can be done in separate code segments.
- It allows to processes to easily share data.
- It allows to extend the address ability of the processor, i.e. segmentation allows the use of 16 bit registers to give an addressing capability of 1 Megabytes. Without segmentation, it would require 20 bit registers.
- It is possible to enhance the memory size of code data or stack segments beyond 64 KB by allotting more than one segment for each area.

MEMORY ORGANIZATION IN THE 8086

- Address bus 20 bit so, 8086 can address 2^20 different unique locations.
- To organize the memory efficiently, the entire memory in 8086 is divided into two memory banks: ***odd bank*** and the ***even bank***.
- The way in which data is read or written is decided by the value of BHE, and the last address bit, that is the A0 line. It is done in the following way:

BHE'	A0	Operation performed on memory
0	0	16 bits of data will be read or written into the memory
0	1	8 bits of data will be read/written into the odd memory bank
1	0	8 bits of data will be read/written into the even memory bank
1	1	No operation is performed


8086
microprocessor
1 MB memory
D15 – D8
D7 – D0
ODD memory bank
EVEN memory bank

- **To read or write 8 bits of data, it would require only 1 CPU cycle, no matter the data is stored in any of the memory banks, but to read or write 16 bits of data, the BIU of the 8086 may require either 1 or 2 memory cycles depending upon whether the lower byte of word is located at even or odd memory address.**
- **If the lower byte of the word is stored at even memory bank and the upper byte is stored at odd memory bank then the CPU will require only 1 memory cycle. So, it is better to store data in this way.**
- **If the lower byte of the word is located at an odd memory address, then the CPU will require 2 memory cycles. The first memory cycle is required for accessing the lower byte of the word through the higher data bus, i.e. D15 to D8, and the second memory cycle is required for accessing the upper byte of the word through the lower data bus, i.e. D7 to D0.**