Advanced Microprocessor

Course Objectives

- To understand internal architectures of 8086
- To understand concepts of Assembly language programming using 8086 Microprocessor.
- To understand the organization of 8086, 80286, 386, 486 and Pentium based Processors.
- To understand the principles behind interfacing components to a Processor.
- To understand the Instruction sets of 8086, 286, 386, 486 and Pentium processors

Course contents

- **8086/8088 Microprocessors**
  - Architecture of 8086/8088 Microprocessor
  - Segmentation and Memory addressing in 8086/8088.
- **Assembly Language Programming**
  - Assembly Language programming: Data Transfer, Arithmetic and Logical Instructions
  - Addressing Modes, Assemblers.
  - String Instructions - Machine Control Instructions - Macros and Conditional Assembly.
- **Hardware Features of 8086/8088 and Memory Interfacing**
  - Features of 8086/8088 -Pin details, Clock generator, Min/Max modes.
  - Memory Interfacing, Address decoding
- **Interrupts and I/O Interfacing**

Course contents…

- Interrupts and its processing.
- I/O Interfacing, Address decoding, ADC and DAC.

- **The 80186/286 Microprocessors**
  - The 80186, 80286 Microprocessors, Architectures and Instruction Sets.
  - Addressing Modes, Memory Management, Protection.
  - Assembly language syntax, Data Types & variables, Writing COM programs
- **The 80386/486 Microprocessors**
  - The 80386/486 Microprocessor, Internal Architecture
  - Memory Organization & Segmentation, Data Types, Registers, Addressing Modes, 80386 Modes

References

- Doughlas V Hall, Microprocessors and its Interfacing.
- Barrey B. Brey, the Intel Microprocessor 8086/8088, 80186, 80286, 80386, 80486, Pentium and Pentium Pro-Processor -

Evaluation Scheme

- Mid ……..30%
- Projects/Assignments ……10%
- Lab practices and quizzes……..20%
- Final Examination……40%

Chapter 1:8086/8088 Microprocessors

- Introduction
- The 8086 Internal architecture
  - ✓ The Execution Unit
  - ✓ The Bus Interface Unit
- Segmentation and memory addressing

Introduction

- The word Microprocessor is a combination of two words:
  - micro and processor.
  - processor means a device which processes binary numbers (0's and 1's).
  - Micro means small.
- A microprocessor is a multi-purpose, programmable, integrated logic device that reads binary instructions from a memory, accepts binary data as input and processes the data according to those instructions and provides result as an output

Introduction: Evolution of Microprocessor (Intel)

- 4004
  - First Microprocessor
  - Introduced in 1971, 4-bit, 12-address lines(4kbytes), 740 KHz
- 8008
  - 1972, 8-bit, 14-address lines (16kb), 500-800 KHz
  - It was not complete CPU-needs additional devices to form functional CPU.
- 8085
  - First 8-bit complete functional CPU.
  - 8085 - 1976, 8-bits, 16-address lines (64Kb), 3-5 MHz, NMOS, 0.37 MIPS

Evolution of Microprocessor (Intel)

- **Limitations of the 8-bit MPs**
  - Low speed of execution
  - Low memory addressing capability
  - Limited number of general purpose registers
  - Less powerful instruction set
  - No support for pipelining or parallelism.
- These limitations led the designers to go for more powerful processors in terms of:
  - Advanced architecture
  - More processing capability
  - Larger memory addressing capability
  - More powerful instruction byte queue
- The 8086 MP was the result of such developmental design effort.

Introduction to 8086

- The 8086, announced in 1978, was the first 16-bit microprocessor introduced by Intel Corporation.
- 8086 and 8088 are internally 16-bit MPs.
- Externally the 8086 has a 16-bit data bus and the 8088 has an 8-bit data bus.
- 8086 has a 20 bit address bus and it can access up to 2^{20} memory locations (1 MB) and 64K of I/O port.
- The 8086 is housed in a 40-pin dual inline package and many pins have multiple functions.
- It requires +5V power supply.

Introduction to 8086 …

- It provides 14, 16-bit registers.
- It has multiplexed address and data bus AD0- AD15 and A16 – A19.
- The 8086 manufactured using **High-performance Metal-Oxide Semiconductor (HMOS) technology**
- 8086 is designed to operate in two modes, Minimum and Maximum.
- It can pre-fetches up to 6 instruction bytes from memory and queues them in order to speed up instruction execution.

The 8088 microprocessor

- The microprocessor 8088 has all the programming facilities that 8086 has, along with some hardware features of 8086.
- It has 1Mbyte memory addressing capability, operating modes (MN/MX), interrupt structure.
- However 8088, unlike 8086, has 8-bit data bus.
- **Architecture of 8088:**
- The register set of 8088 is exactly the same as in to 8086.
- The architecture of 8088 is also similar to 8086 except for two changes;
  - 8088 has 4-byte instruction queue and,
  - 8088 has 8-bit data bus.

Internal architecture of 8086

- 8086 MP is divided into two independent functional parts:
  - Bus interface unit (BIU)
  - Execution unit (EU)
- Dividing the work b/n the two units speeds up the computation process.
- The BIU performs all bus operations such as instruction fetching, reading and writing operands from memory and calculating the addresses of the memory operands.
- The fetched instruction bytes are transferred to the instruction queue.
- EU decodes and executes instructions from the instruction system byte queue.

Internal architecture of 8086…

- Both units operate asynchronously
  - ✓ This gives the 8086 an overlapping instruction fetch and execution mechanism
  - ✓ This overlapping is called Pipelining. This results in efficient use of the system bus and system performance.
- BIU contains Instruction queue, segment registers, instruction pointer, address adder
- EU contains control circuitery, instruction decoder, ALU, pointer and Index register, flag register

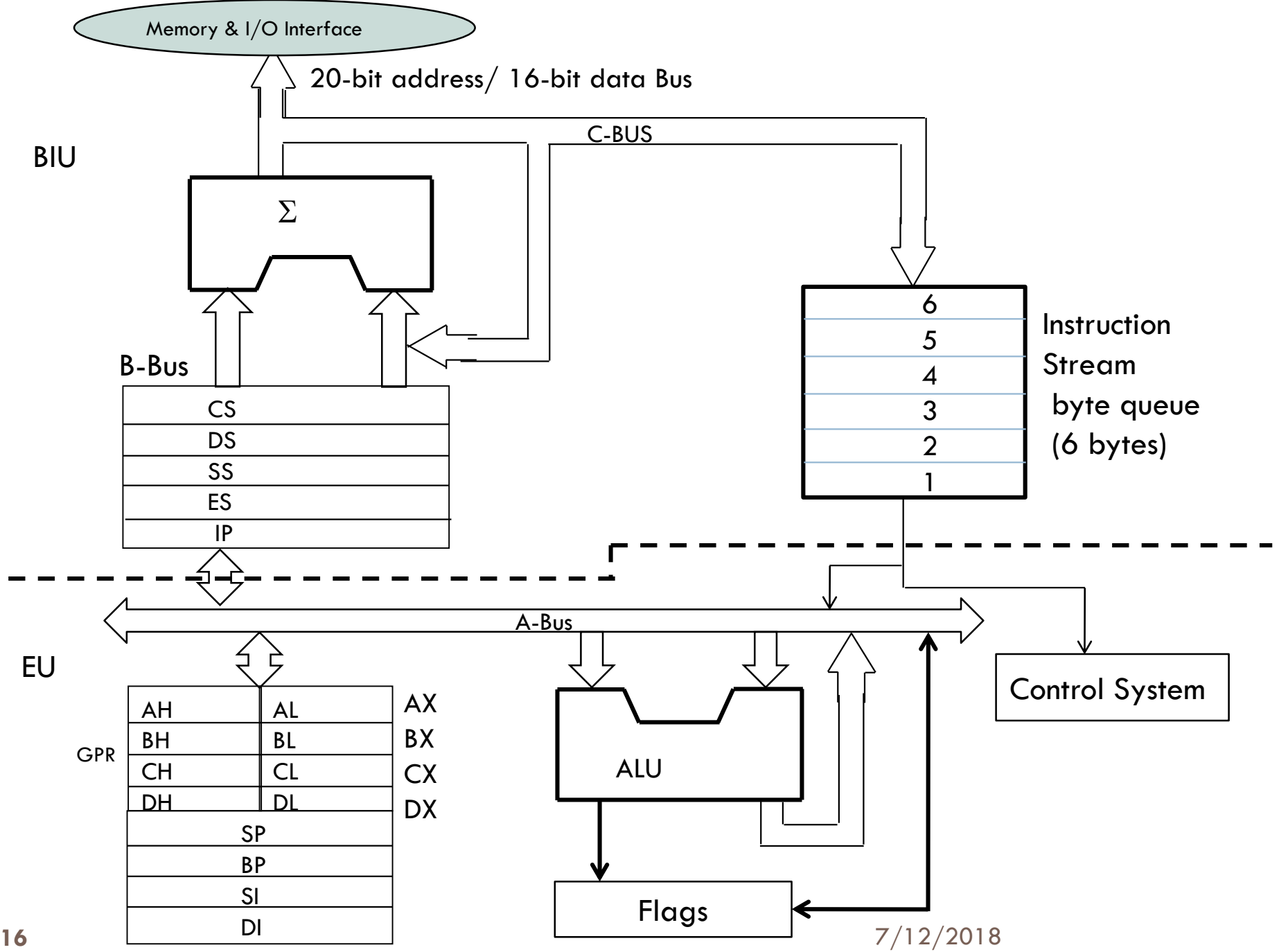


Fig. The 8086 Architecture (Internal Block Diagram)

Bus Interface Unit (BIU)

- This unit is responsible for establishing interface with external devices including memory via the bus.
- It provides a full 16 bit bidirectional data bus and 20 bit address bus.
- Functions of BIU:
  - Instruction fetch, Instruction queuing, Operand fetch and storage, Address calculation and Bus control.
  - The BIU uses a mechanism known as an instruction stream queue to implement a ***pipeline architecture***

BIU …

- This queue permits pre-fetch of up to six bytes of instruction code.
- The BIU fetches and stores these pre-fetched bytes in a **FIFO** register called a queue.
- With its 16 bit data bus, the BIU fetches two instruction bytes in a single memory cycle.
- When the **EU** is ready for its next instruction, it simply reads the instruction byte(s) from the queue in the BIU.
- This overlapping of instruction fetch and execution speeds up the instruction execution process.
- This overlapping of machine cycles is called pipelining.

Segment Registers

- The 8086 BIU sends out 20-bits address. So it can address any of 2^{20} or 1MB in memory.
- At any given time the 8086 works with only four 65,536 (64Kbytes) segments within this **1,048, 576** bytes range.
- The following four segment registers in the BIU are used to hold the upper 16-bits of the starting address of four memory segments that the 8086 is working at a particular time:

CS Code segment register

SS Stack segment register

DS Data segment register

ES Extra segment register

Physical address and segment registers

- The complete physical address which is 20-bits long is generated using segment and offset registers, each 16-bits long.
- The BIU always inserts **zeros** for the lowest 4 bits (nibble) of the 20-bit starting address for a segment.

N.B P.A=Physical Address

- If **CS** register contains **348AH**, then code segment will start at address **348A0 H**.

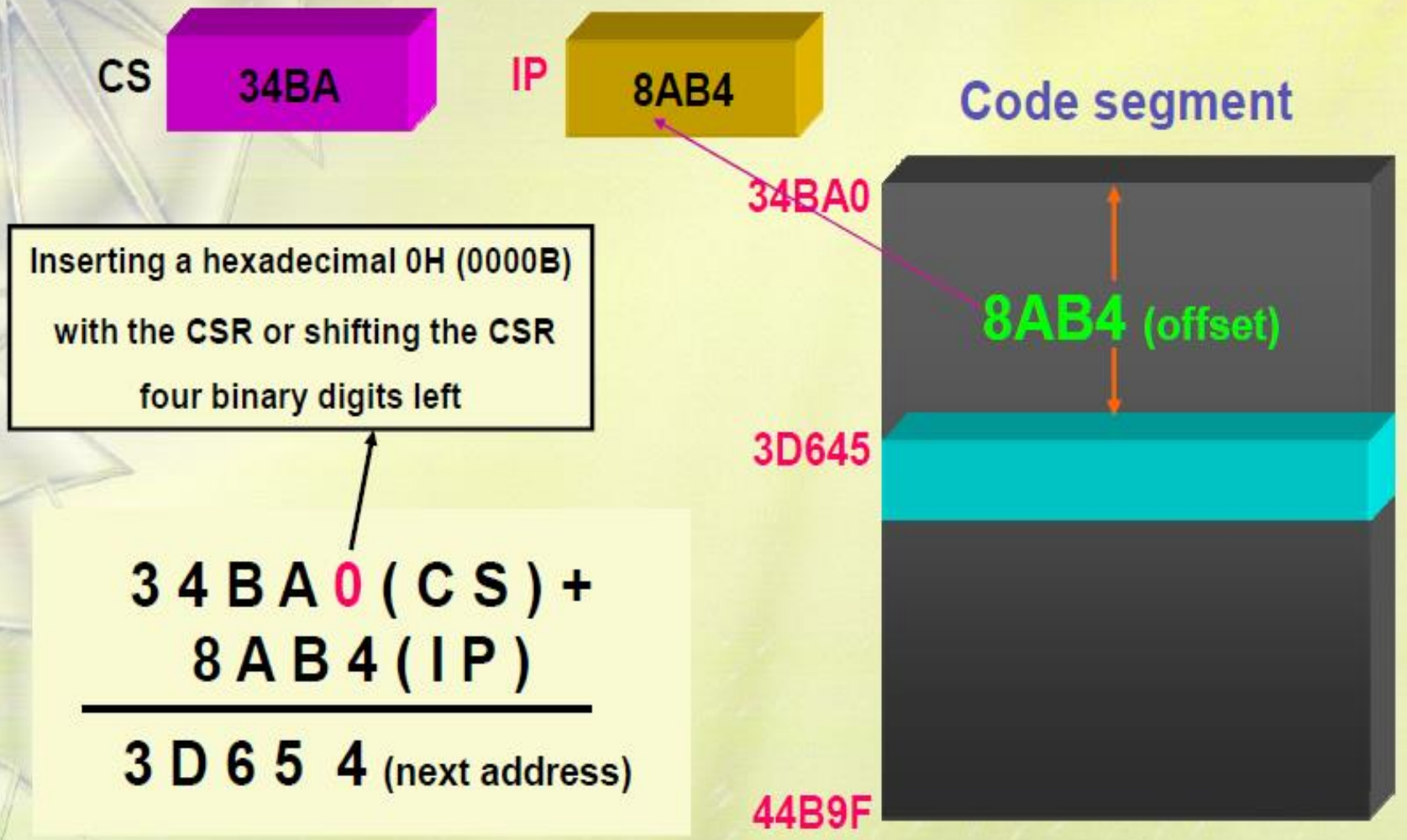
  0011 0100 1000 1010 ← 0000 = 0011 0100 1000 1010 0000 = **348A0H**

- In other words, a 64k bytes segment can be located anywhere within the 1Mb address space, but the **segment** will always start at an address with **zeros** in the lowest 4-bits.
- The IP contains the distance or **offset** from the base address to the next instruction byte to be fetched.
- An alternate way of representing a **20-bit physical address** is

  Segment base : Offset

- **Example: CS:IP** i.e. **348A : 4214**

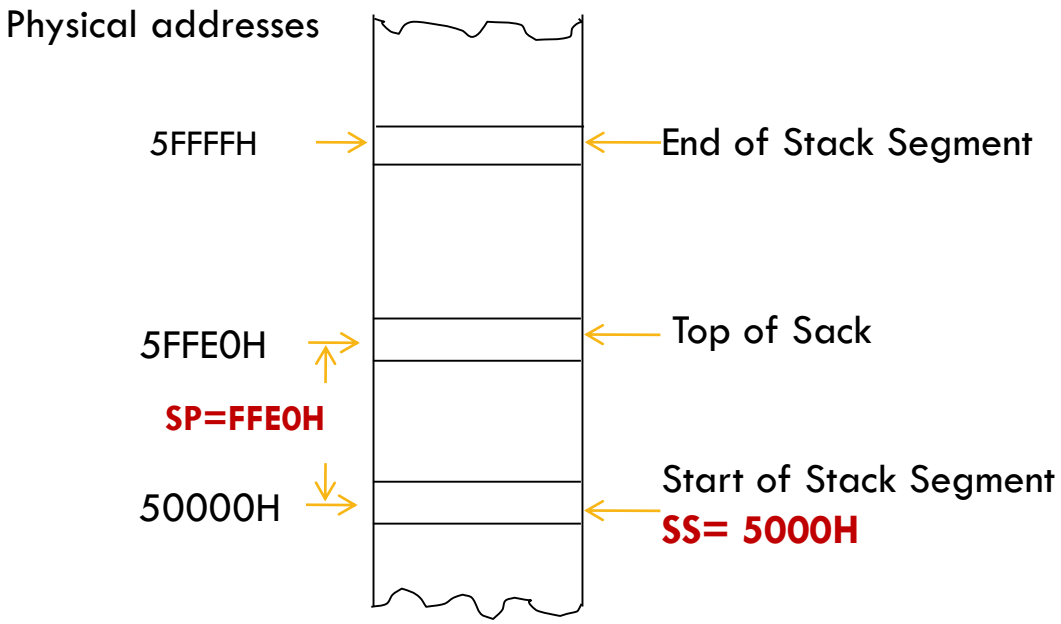
- The following examples shows the CS:IP scheme of address formation:

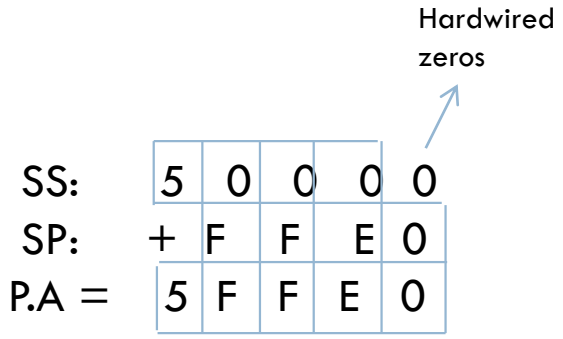


The **SS** and **SP** registers

- A stack is a part of memory to store addresses and data during subprogram processing. The 8086 allows an entire of **64Kb** segment as a stack.
- **SS** - Stack segment register – holds the upper 16-bits of the starting address of the

  stack segment.
- **SP**- Stack pointer register– holds the 16-bit offset from the start of the segment to the memory location where a word was most recently stored on the stack (**top stack**).





Default segment and offset registers

Segment	offset	special purpose
CS	IP	Instruction address
SS	SP or BP	Stack address
DS	BX, DI, SI, val8, val16	Data address
ES	DI for string instruction	String destination address

Execution Unit (EU)

- The Execution unit is responsible for decoding and executing all instructions.
- The EU
  - Extracts instructions from the top of the queue in the BIU,
  - Decodes them, generates operands if necessary,
  - Passes them to the BIU and requests it to perform the read or write cycles to memory or I/O and perform the operation specified by the instruction on the operands.
- During the execution of the instruction, the EU tests the status and control flags and updates them based on the results of executing the instruction.

EU contd…

- The **EU** contains:

-Control circuitry - which directs internal operations.

- **Decoder-** translates instructions fetched from memory into a series of actions which the EU carries out.

- A **16-bit ALU** – which can add, subtract, AND, OR, XOR, increment, decrement, complement, or shift binary numbers.

- **Registers:**–

General (multi-purpose) registers:

AH, AL, BH, BL, CH, CL, DH, DL or AX, BX, CX, DX

BP, DI, SI

Special purpose register : SP

- **Flags**:- Nine individual bits of the status register are used as **control** flags (3 of them) and **status** flags (6 of them).

Pointer and Index registers in the EU

BP - Base pointer register
SI - Source index register
DI - Destination index register

These three register can be used for temporary storage of data just as the general-purpose registers. However, their main use is to hold the 16-bit **offset** of data word in one of the **data segments.**

E.g. DS:SI ES:DI

The index registers are particularly useful for string manipulations.

Some registers have additional specific uses. **E.g.**

AX – as 16-bit accumulator.

CX – as default counter incase of string and loop instructions.

BX – as offset storage in case of certain addressing modes.

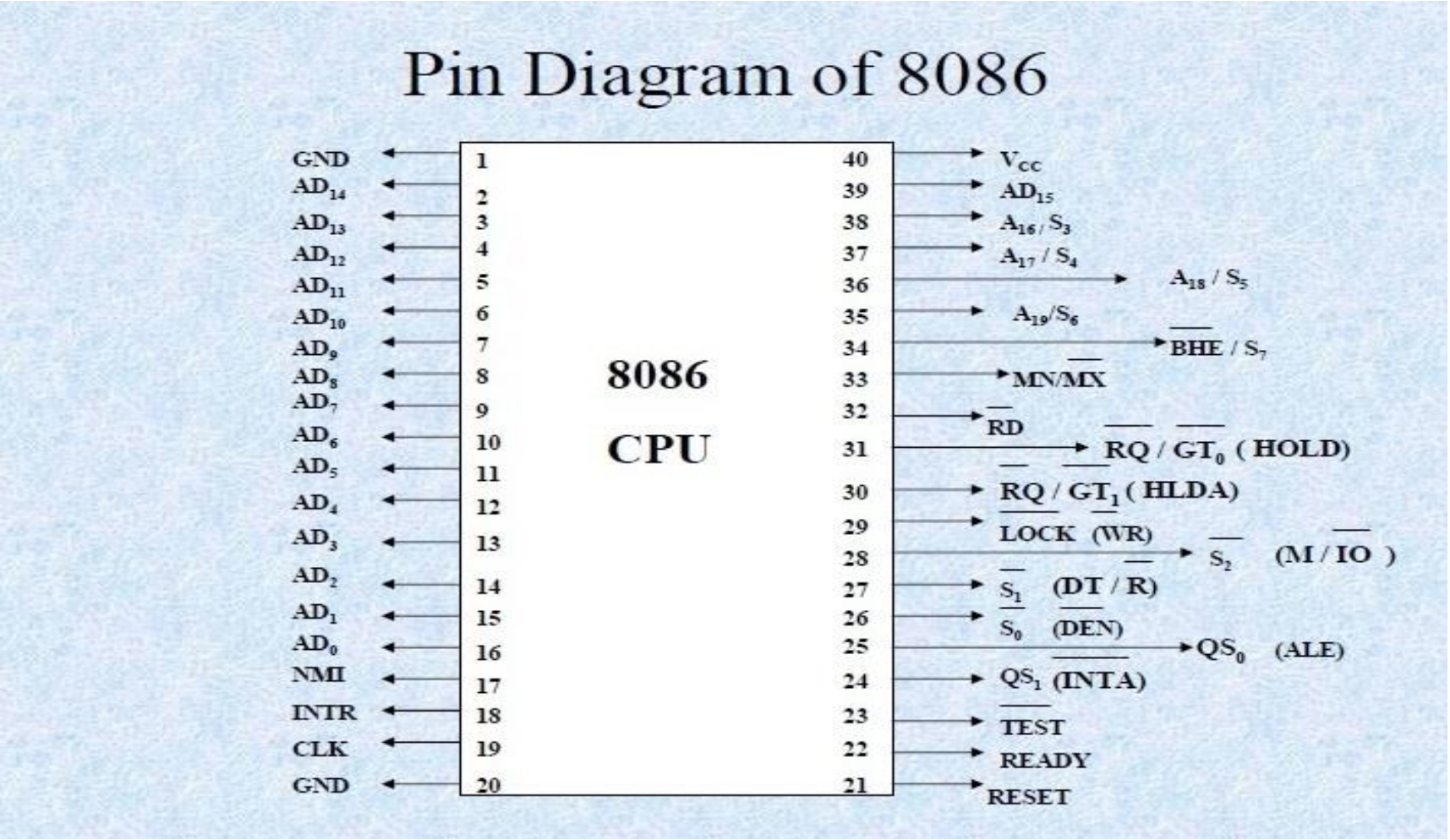
DX – as an implicit operand or destination in case of few instructions.

E.g.

- For memory addressing when data are transferred between I/O port and memory using certain I/O instructions.
- Holds a part of the result from a multiplication or part of the dividend before a division.

Note: The offset may be the content of IP, BX, SI, DI, SP or an immediate 16-bit value, depending upon the addressing mode.

Signal description of 8086 MP



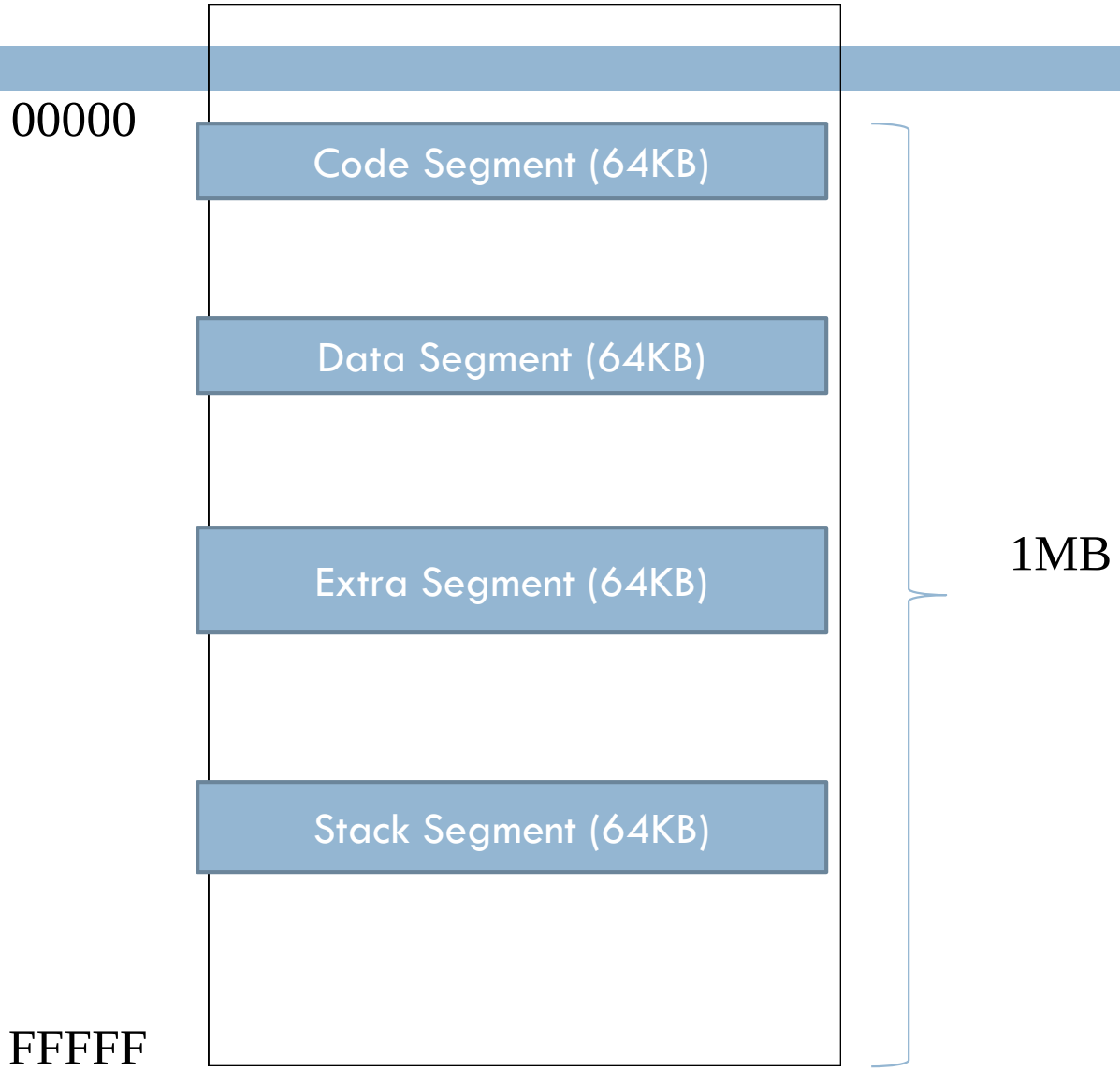
Signal description

- 8086 MP operates in **single processor** or **multiprocessor configurations** to achieve high performance (**minimum mode** & **maximum mode**).
  - **Minimum mode**: The 8086 processor works in a single processor environment. All control signals for memory and I/O are generated by the microprocessor.
  - **Maximum mode:** is designed to be used when a coprocessor exists in the system. 8086 works in a multiprocessor environment. Control signals for memory and I/O are generated by an external Bus Controller.
- The 8086 signals can be categorized in **three** groups:
  - Signals having common functions in **minimum** as well as **maximum** modes.
  - Signals which have special functions for **minimum** mode.
  - Signals having special functions for **maximum** mode.

Segmented Memory

- The memory in an 8086/88 based system is organized as segmented memory.
- The CPU 8086 is able to address 1Mbyte of memory.
- The Complete physically available memory may be divided into a number of logical segments.
- The size of each segment is 64 KB.
- A segment is an area that begins at any location which is divisible by 16.
- A segment may be located any where in the memory.
- Each of these segments can be used for a specific function.
- Code segment is used for storing the instructions.

Physical Memory



Contd…

- The stack segment is used as a stack and it is used to store the return addresses.
- The data and extra segments are used for storing data byte.
- **Advantages of Segmented memory Scheme**
  - Allows the memory capacity to be 1MB although the actual addresses to be handled are of 16 bit size.
  - Allows the placing of code, data and stack portions of the same program in different parts (segments) for data and code protection.