Segment 2: 8086 Microprocessor

CSE-3523: Microprocessors, Microcontrollers, and Embedded Systems

Department of CSE, IIUC

Intel 8086 Microprocessor

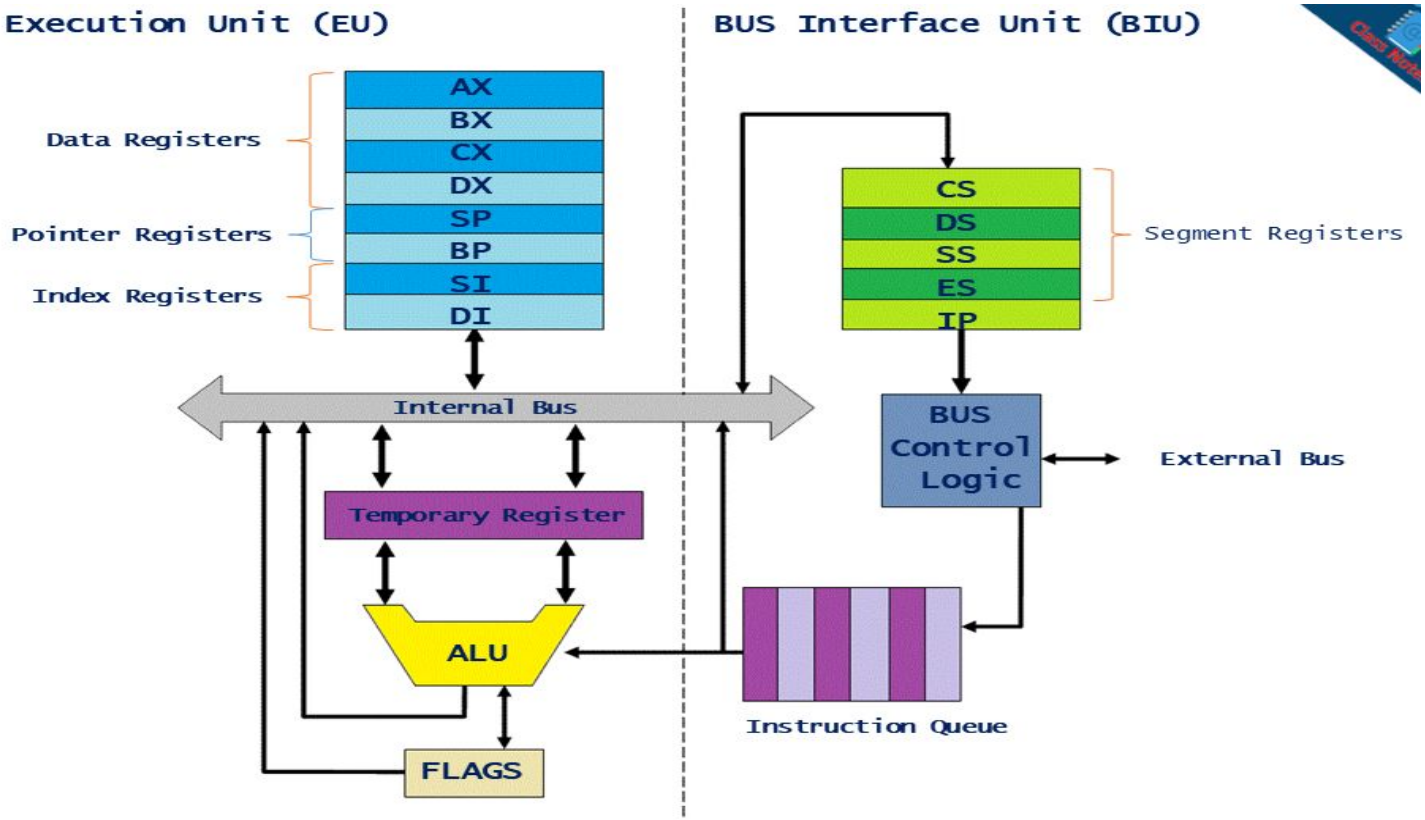
Clock Speed	Data Width	Data Bus	Address Bus	Addressable Memory
5 MHz	16 bit	16 bit	20 bit	2^20 =1 Megabyte

- Intel 8086 microprocessor is a 16 bit microprocessor
- Can process data and memory addresses that are represented by 16 bits at a time.

Intel 8086 Internal Architecture

- 2 independent functional unit
  - Bus Interface Unit (BIU):
    - Fetch Instruction from memory
    - Read data from port and memory
    - Write data to port and memory
  - Execution Unit (EU):
    - Tells the BIU where to fetch
    - Decodes instruction
    - Executes instruction

Intel 8086 Internal Architecture



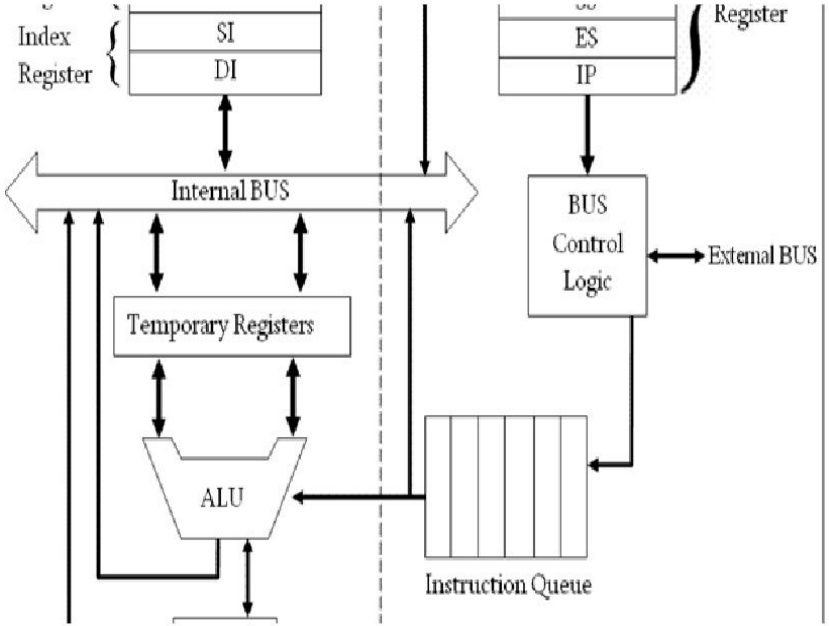
Intel 8086 Internal Architecture:

Bus Interface Unit (BIU):

- The BIU facilitates communication between the EU and the memory or I/O circuits.
- The BIU is responsible for transmitting addresses, data, and control signals on the buses.
- The **instruction pointer (IP)** contains the address of the next instruction to be executed by the EU.

Intel 8086 Internal Architecture:

Bus Interface Unit (BIU):Instruction Prefetch



- While the EU is executing an instruction, the BIU fetches up to six bytes of the next instruction and places them in the instruction queue.

Intel 8086 Internal Architecture:
Execution Unit

1. Control Circuitry
2. Instruction Decoder
3. ALU ():
    - Add, subtract, AND, OR XOR, increment, decrement, complement, shift binary numbers.

- The data for the operations are stored in circuits called registers.
- EU contains few registers for holding data and address

Essential Registers for Instruction Execution

Program Counter (PC) :

It contains the address of an instruction to be executed next. The PC is updated by the CPU after each instruction is executed so that it always points to the next instruction to be executed. A branch or skip instruction will also modify the content of the PC.

Instruction Register (IR) :

it contains the instruction most recently fetched or executed. The fetched instruction is loaded into an IR, where the opcode and operand specifier is analyzed.

Memory Buffer (or Data) Register (MBR or MDR) :

it contains a word of data to be written to memory are the words most recently read. Contents of MBR are directly connected to the data bus.

Memory Address Register (MAR) :

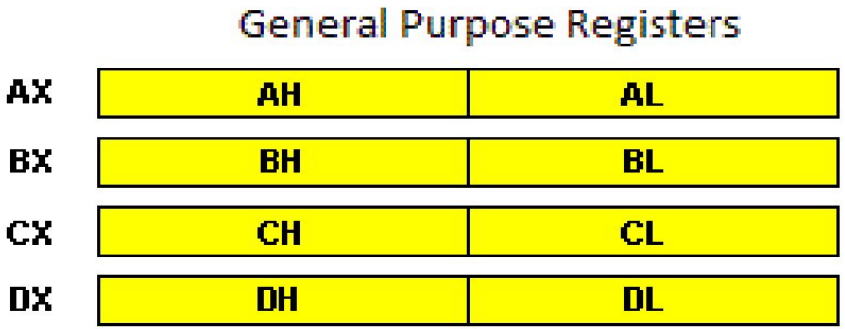
It contains the address of a location of main memory from where information has to be fetched for information has to be stored. Contents of MAR are directly connected to the address bus.

Intel 8086 Internal Architecture:
Registers

- Information inside microprocessor is stored in register
- Total Fourteen registers: All of these registers are 16-bit long
- Classified based on their functions they perform:
    1. Data Registers: hold data for operation
        - Four (4) general data registers
    2. Address Registers: hold address of data or instruction
        - Segment Register
        - Pointer Register
        - Index Register
    3. A Status Register: keep current status of the processor :FLAGS register
- Temporary register: for holding operands

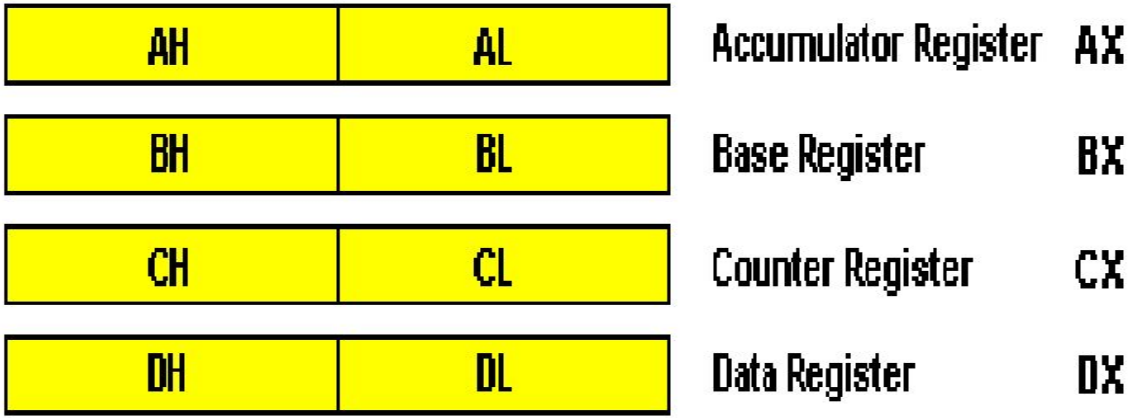
Intel 8086 Internal Architecture:
Data Registers

- Hold data for general data manipulation
- **Why Register? Why not Memory?**
  - Even though the processor can operate on data stored in memory, the same instruction is faster if stored in registers, requires fewer clock cycles
- High and low bytes can be accessed separately



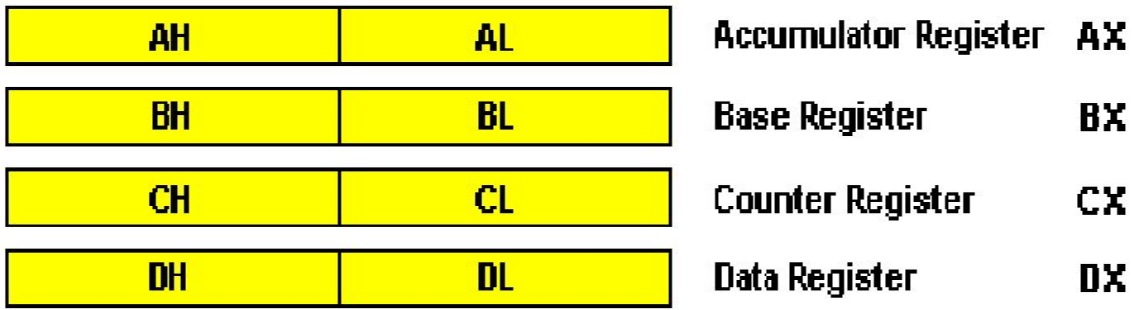
Intel 8086 Internal Architecture:
Data Registers

- In addition to being general-purpose data registers, these 4 also performs following special functions.



Intel 8086 Internal Architecture:
Data Registers

- AX (Accumulator):
  - preferred register to use arithmetic, logic and data transfer, since its use generate shortest machine code
  - In multiplication & division one of the number must be in AX or AL
  - Input & Output also requires to use of AX and AL



Intel 8086 Internal Architecture:
Data Registers

- BX (Base Register):
  - Also serves as an address register
- CX (Count Register):
  - Program loop constructions
- DX (Data Register):
  - In multiplication & division
  - Also used in Input & Output

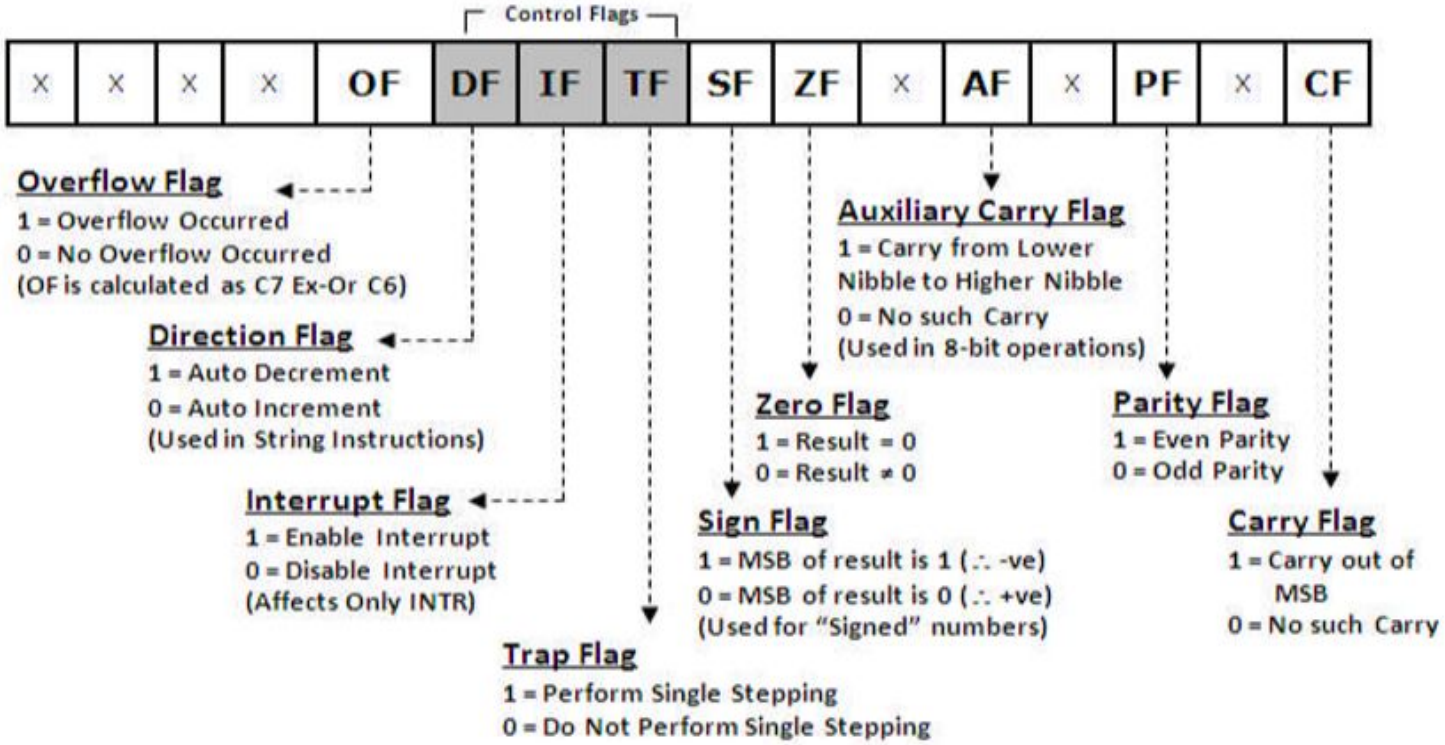
Intel 8086 Internal Architecture:
Flag Register

- A 16 bit register
- The purpose of FLAGS is to indicate the status of microprocessor by setting of individual bits called flags.
- 2 kinds of flags:
- Status flags:
  - When a subtraction operation results in a 0, ZF (zero flag) is set to 1 (true).
- Control flags: To enable or disable certain operation of microprocessor
  - Inputs from keyboard is ignored by processor if Interrupt flag is set zero.

Intel 8086 Internal Architecture:

Flag Register

In 16 bit flag: 9 active flag



Intel 8086 Internal Architecture:
Memory Segment

- The idea of memory segments is a direct consequence of using a 20-bit (physical) address in a 16-bit processor.
- Segmentation:

  Segmentation is the process in which the main **memory** of the computer is **divided into** different **segments**
- A **segment** is a logical unit of memory that may be up to 2^16=64 kilobytes long.

Intel 8086 Internal Architecture:

Segment: Why memory is divided into segments?

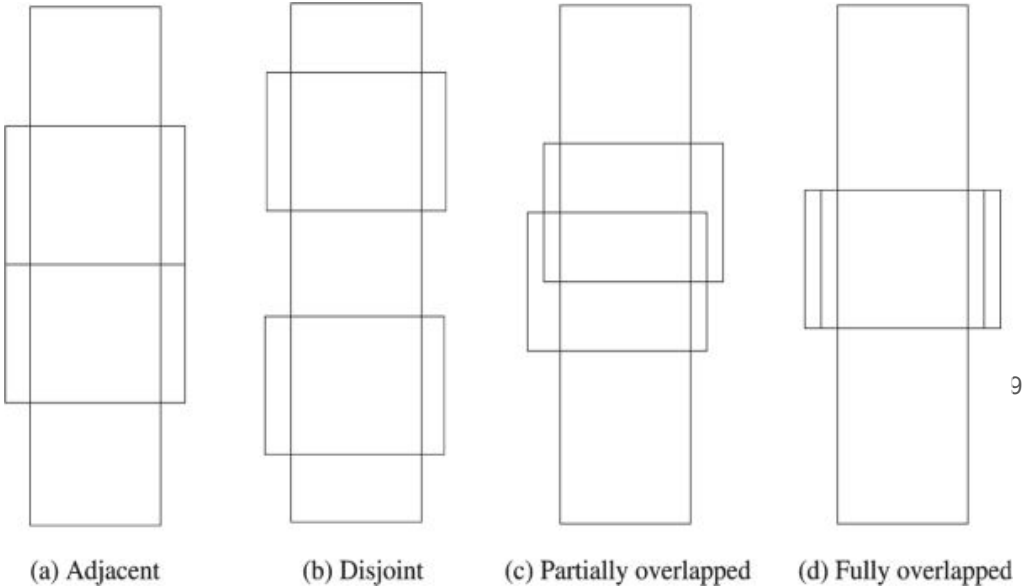
- 8086 BIU sends out 20 bit address, so it can address any of 2^20 = 1 Mega bytes in memory
- But, at any time 8086 can address any of 2^16= 64 Kbyte memory.
- Hence, memory is divided into (4)four 2^16= 64 Kbyte segments.
- At any given time 8086 can work with only this 4 (four) 64 Kbyte segments within 1Mbyte memory.
- Four segment registers hold the upper 16 bit of starting address of this four memory segments that 8086 is working at a particular time.

Intel 8086 Internal Architecture:
Segment

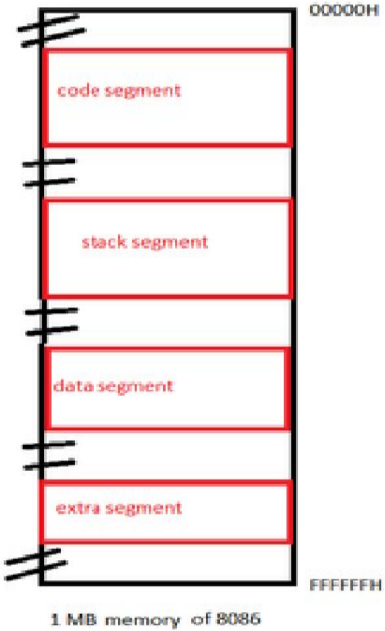
- 4 Segments:
  - Data Segment
  - Code Segment
  - Stack Segment
  - Extra Segment
- Each **segment** is made up of contiguous memory locations.
- Each **segment** has its own segment base address with zeros in the lowest 4 bits.
- A segment number is 16 bits, so the highest segment number is FFFFh.

Intel 8086 Internal Architecture:
Segment

- Memory segments can be separated as shown



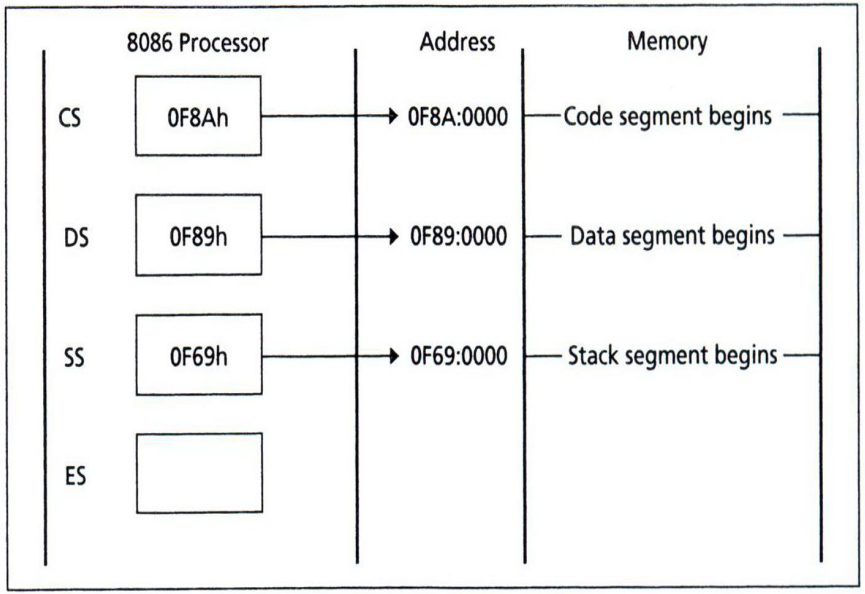
(a) Adjacent (b) Disjoint (c) Partially overlapped (d) Fully overlapped



1 MB memory of 8086

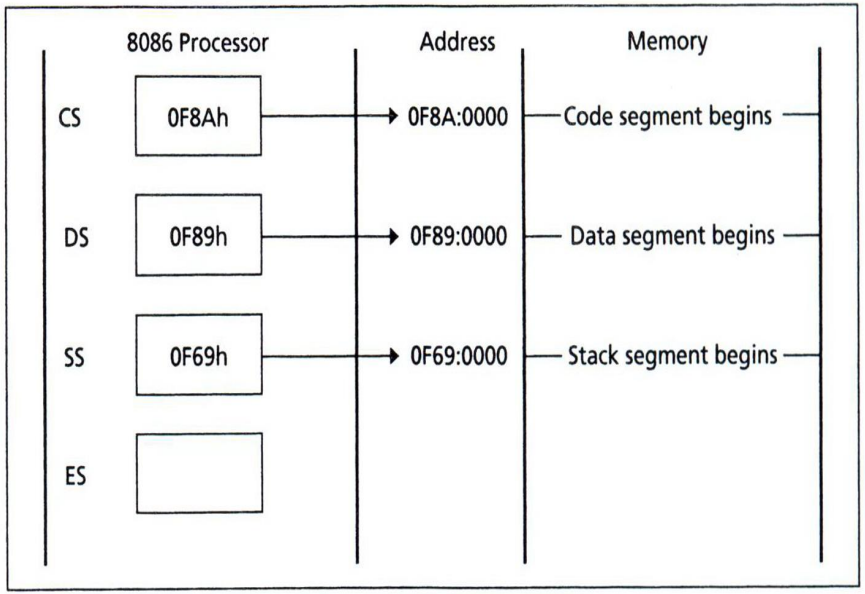
Intel 8086 Internal Architecture:
Segments

- 4 segments registers in BIU holds the Upper 16 bits of starting address of four memory segments that the 8086 is currently working with.
- BIU always insert zeros for lowest 4 bit(nibble) of the 20 bit starting address for a segment
- 64 Kbyte segment can be located anywhere within the 1Mbyte address space but always start at an address with zeros in the lowest 4 bits



Intel 8086 Internal Architecture:
Segments

- Segments start every 10h= 16 bytes
- And Stating address of any segment always ends with a hex digit 0.
- We call 16 bytes a paragraph
- An address that is divisible by 16 a paragraph boundary.

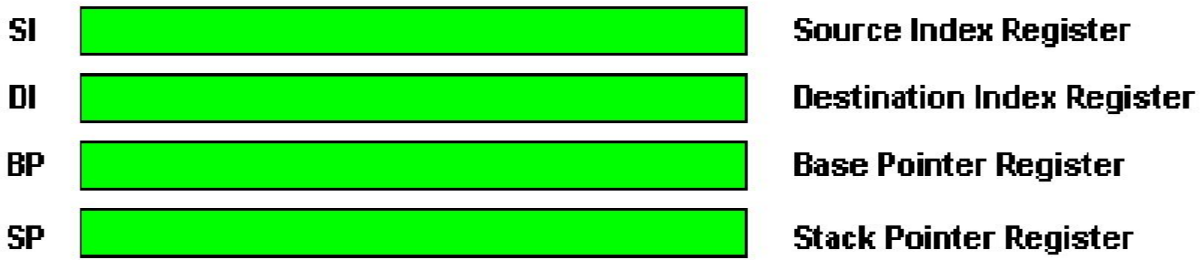


Intel 8086 Internal Architecture:
Segment Registers

Memory segment	Segment register	Offset register
Code segment	Code segment Register (CSR)	Instruction Pointer (IP)
Data segment	Data segment Register (DSR)	Source index (SI)/ Destination index (DI)
Stack segment	Stack segment Register (SSR)	Stack Pointer (SP)/ Base Pointer (BP)
Extra segment	Extra segment Register (ESR)	Destination Index(DI)

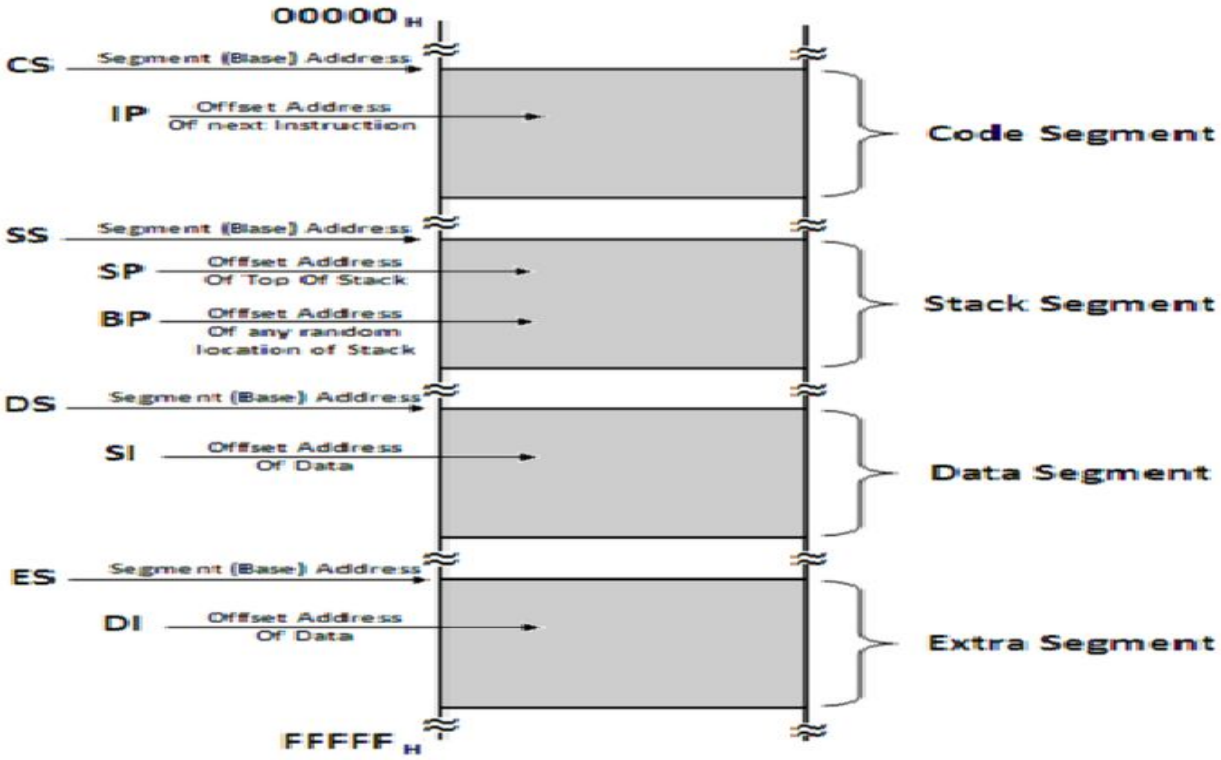
Intel 8086 Internal Architecture:
Pointer and Index Registers

- Pointer Register->points to memory in Stack Segment and Code
- Index Register -> points to memory in Data Segment
- Unlike segment registers, pointer and index registers can be used in arithmetic and other operations.



Intel 8086 Internal Architecture:

Segment



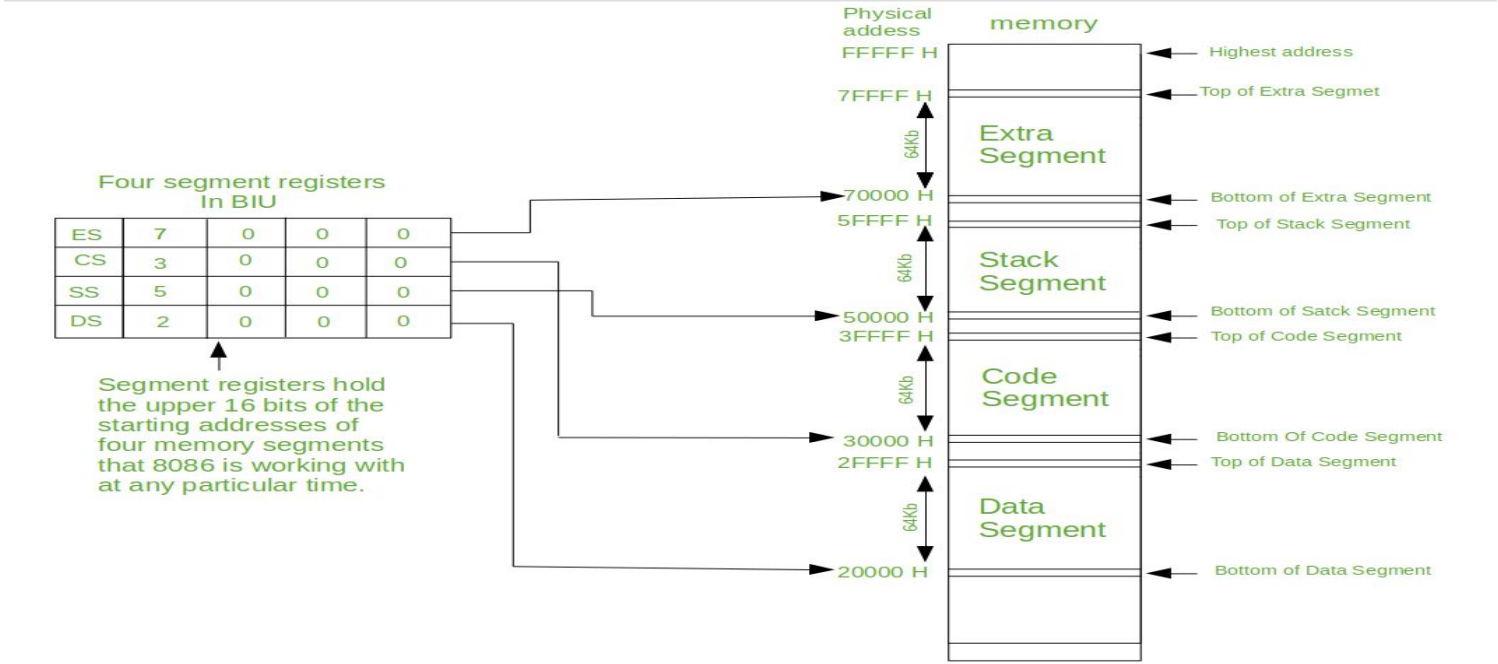
Intel 8086 Internal Architecture:
Physical Address

- 16 bit 8086 assigns a 20-bit physical address to its memory.
- It is possible to address 2^{20} = 1,048,576 bytes (one megabyte) of memory.

0000 0000 0000 0000 0000	00000h
0000 0000 0000 0000 0001	00001h
0000 0000 0000 0000 0010	00002h
0000 0000 0000 0000 0011	00003h
.....	
1111 1111 1111 1111 1110	FFFFEh
1111 1111 1111 1111 1111	FFFFFh

Intel 8086 Internal Architecture:
Segment Registers

Segment registers store addresses of instructions and data in memory.

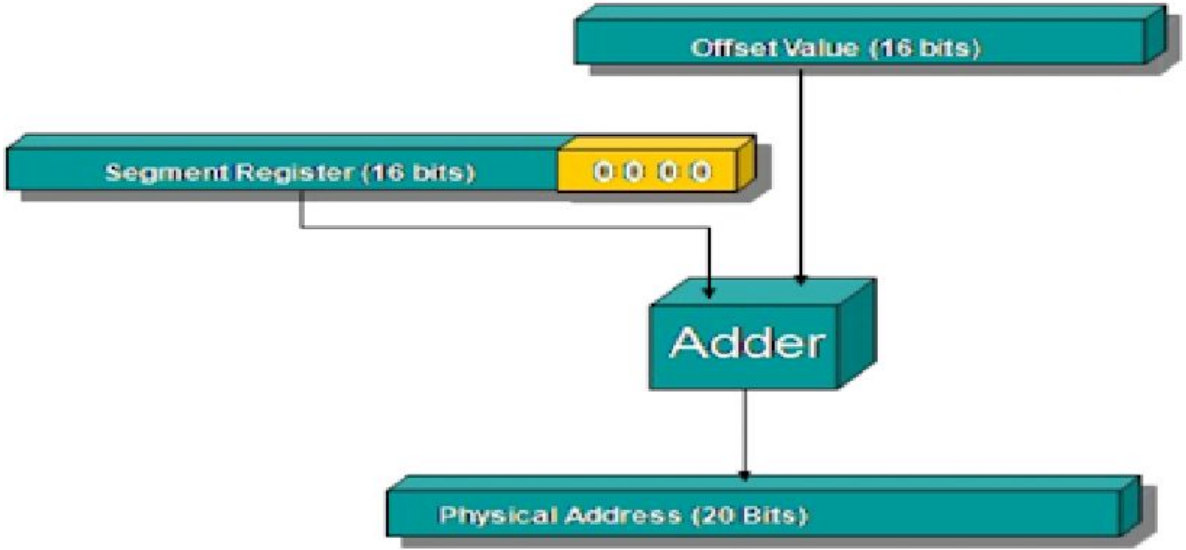


Intel 8086 Internal Architecture:
How 16 bit microprocessor works with 20 bit memory address?

- Segment : Offset also known as Logical Address
- To obtain a 20-bit physical address:
    - First shifts the segment address 4 bits to the left (this is equivalent to multiplying by 10h),
    - and then adds the offset.
- **Physical Address = Segment × 10h + Offset**
- A4FB:4872 = A4FB0h + 4872h = A9822h

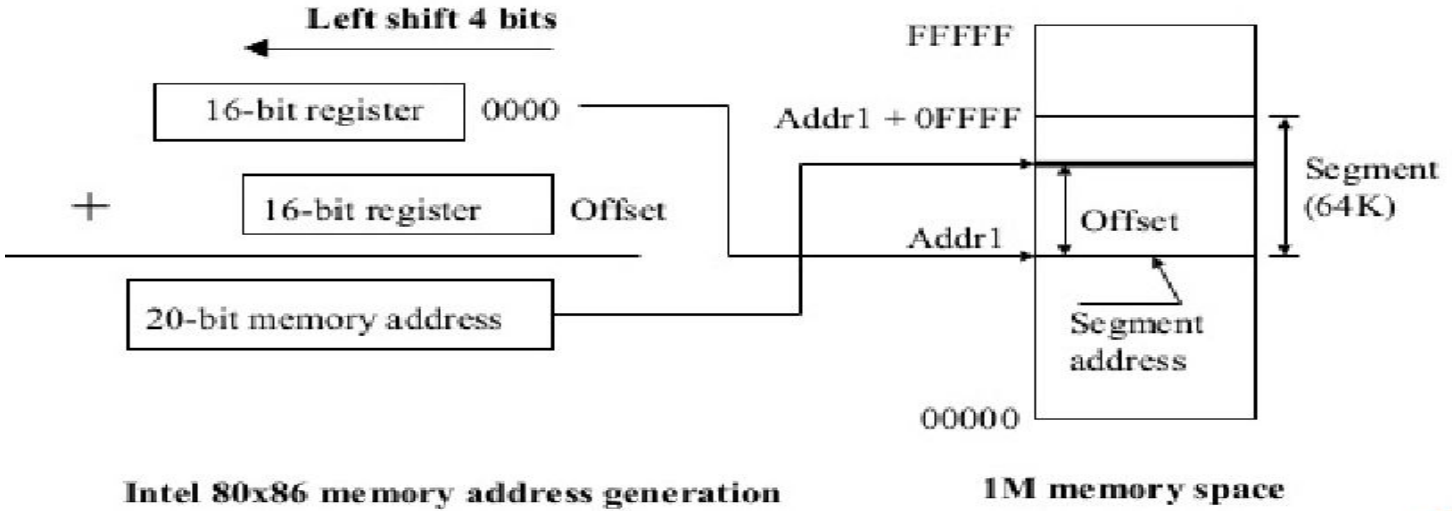
Intel 8086 Internal Architecture:

How 16 bit microprocessor works with 20 bit memory address?



Intel 8086 Internal Architecture:

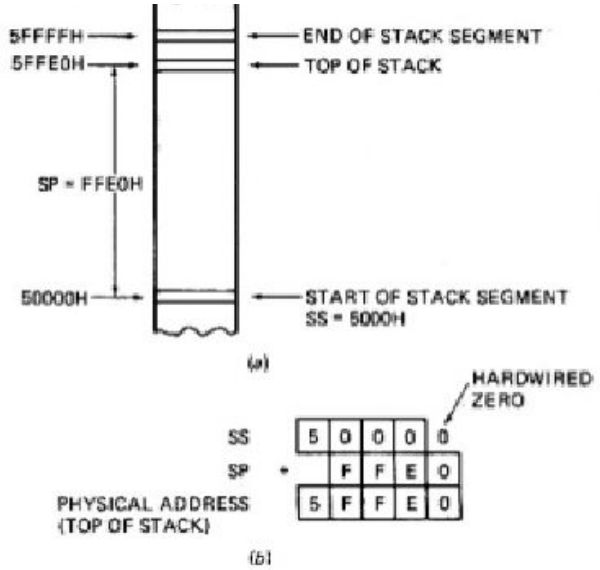
How 16 bit microprocessor works with 20 bit memory address?



Intel 80x86 memory address generation **1M memory space**

Intel 8086 Internal Architecture:

How 16 bit microprocessor works with 20 bit memory address?



FFE0h

- Now the actual address in the physical memory space is given by SS:SP and calculated as:
- SS is first shifted left four times

SS<<4 = 50000h

Then the offset in SP is added

50000h

+ FFE0h

Top of stack **5FFE0h**

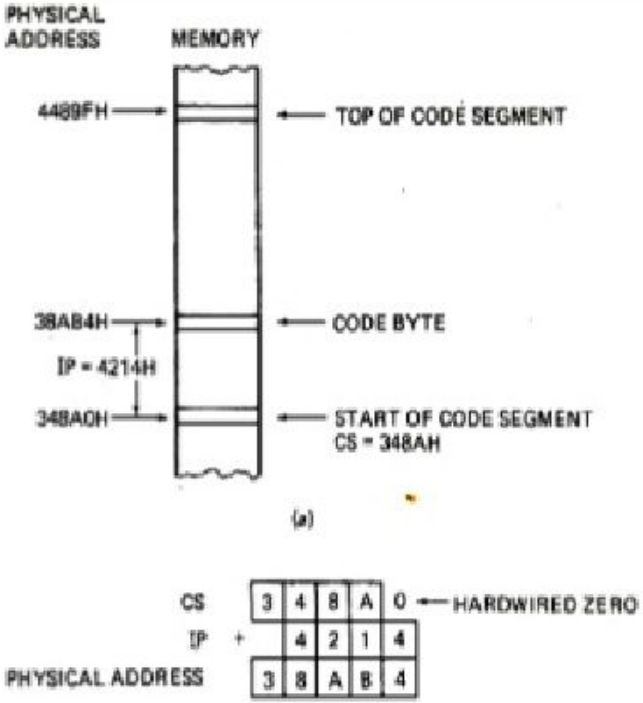
Intel 8086 Internal Architecture:
IP (Instruction Pointer)

- To access instructions, 8086 uses CS(Code Segment) and IP (instruction pointer).
- CS contains the segment number or segment base address of the next instruction,
- IP contains the distance or offset from base address to the next instruction byte to be fetched.
- IP is updated each time an instruction is executed so that it will point to the next instruction.
- Unlike other registers, IP cannot be directly manipulated by an instruction; that is, an instruction may not contain IP as its operand.

Intel 8086 Internal Architecture:
IP (Instruction Pointer)

- To access instructions, 8086 uses CS(Code Segment) and IP.
- CS contains the segment number of the next instruction, and IP contains the offset.
- IP contains the distance or offset from base address to the next instruction byte to be fetched.

Intel 8086 Internal Architecture:
IP (Instruction Pointer)



- E.g. Let CS holds **348Ah**, and IP holds **4214h.** Now the actual address in the physical memory space is given by CS:IP and calculated as:
- CS is first shifted left four times

CS <<4= 348A0h

Then the offset in IP is added

348A0h

+ 4214h

Actual address **38AB4h**

Addressing Modes of 8086

Addressing modes are a technique of describing how an instruction will operate data. For instructions that induce a program branch, this addressing method is necessary. A branch is an intra-segment branch or a nearby branch if it is located within the same segment. A branch is called an Inter-Segment Branch or a Far Branch if it is in a different segment. Addressing modes store the operands (source or destination) in any internal register or memory location. Addressing modes refer to the various approaches to addressing the operands.

Types Of Addressing Modes Of 8086

Immediate addressing mode:

The addressing mode in which the data operand is a part of the instruction itself is known as immediate addressing mode.

Example

MOV CX, 4929 H, ADD AX, 2387 H, MOV AL, FFH

Register addressing mode

It means that the register is the source of an operand for an instruction.

Example

MOV CX, AX ; copies the contents of the 16-bit AX register into the 16-bit CX register),

ADD BX, AX

Direct addressing mode

The addressing mode in which the effective address of the memory location is written directly in the instruction.

Example

MOV AX, [1592H], MOV AL, [0300H]

Register indirect addressing mode

This addressing mode allows data to be addressed at any memory location through an offset address held in any of the following registers: BP, BX, DI & SI.

Example

MOV AX, [BX] ; Suppose the register BX contains 4895H, then the contents 4895H are moved to AX

ADD CX, {BX}

Based addressing mode

In this addressing mode, the offset address of the operand is given by the sum of contents of the BX/BP registers and 8-bit/16-bit displacement.

Example

MOV DX, [BX+04], ADD CL, [BX+08]

Indexed addressing mode

In this addressing mode, the operands offset address is found by adding the contents of SI or DI register and 8-bit/16-bit displacements.

Example

MOV BX, [SI+16], ADD AL, [DI+16]

Based-index addressing mode

In this addressing mode, the offset address of the operand is computed by summing the base register to the contents of an Index register.

Example

ADD CX, [AX+SI], MOV AX, [AX+DI]

Based indexed with displacement mode

In this addressing mode, the operands offset is computed by adding the base register contents. An Index registers contents and 8 or 16-bit displacement.

Example

MOV AX, [BX+DI+08], ADD CX, [BX+SI+16]