Microprocessor 8086

- Microprocessor 8086 is the first 16-bit microprocessor from INTEL, released in the year 1978.
- The term 16 bit means that its ALU, its internal registers and most of the instructions are designed to work with 16 bit binary words.
- Microprocessor 8086 has a 16-bit data bus and 20-bit address bus. So, it can address any one of 220 =1048576=1 megabyte memory locations.
- INTEL 8088 has the same ALU, same registers and same instruction set as the 8086.But the only difference is 8088 has only 8-bit data bus and 20-bit address bus.

Microprocessor 8086

- The 8086 microprocessor can work in two modes of operations. The 8088 can only read/write/ports of only 8-bit data at a time.
- They are Minimum mode and Maximum mode.
- In the minimum mode of operation the microprocessor do not associate with any co-processors and cannot be used for multiprocessor systems.
- In the maximum mode the 8086 can work in multi-processor or co-processor configuration.
- This minimum or maximum operations are decided by the pin MN/ MX. When this pin is high 8086 operates in minimum mode otherwise it operates in Maximum mode.

8086 Microprocessor Features:

1. It is 16-bit microprocessor.
2. It has a 16-bit data bus, so it can read data from or write data to memory and ports either 16-bit or 8-bit at a time.
3. It has 20 bit address bus and can access up to 220 memory locations (1MB).
4. It can support up to 64K I/O ports.
5. It provides 14, 16-bit registers.
6. It has multiplexed address and data bus AD0-AD15 & A16-A19.
7. Pre-fetches up to 6 instruction bytes from memory and queues them in order to speed up the processing.
8. 8086 supports 2 modes of operation: Minimum mode and Maximum mode

Architecture of 8086 Microprocessor

To improve the performance by implementing the parallel processing concept the CPU of the 8086 /8088 is divided into two independent sections :

- They are Bus Interface Unit (BIU).
- Execution Unit (EU).

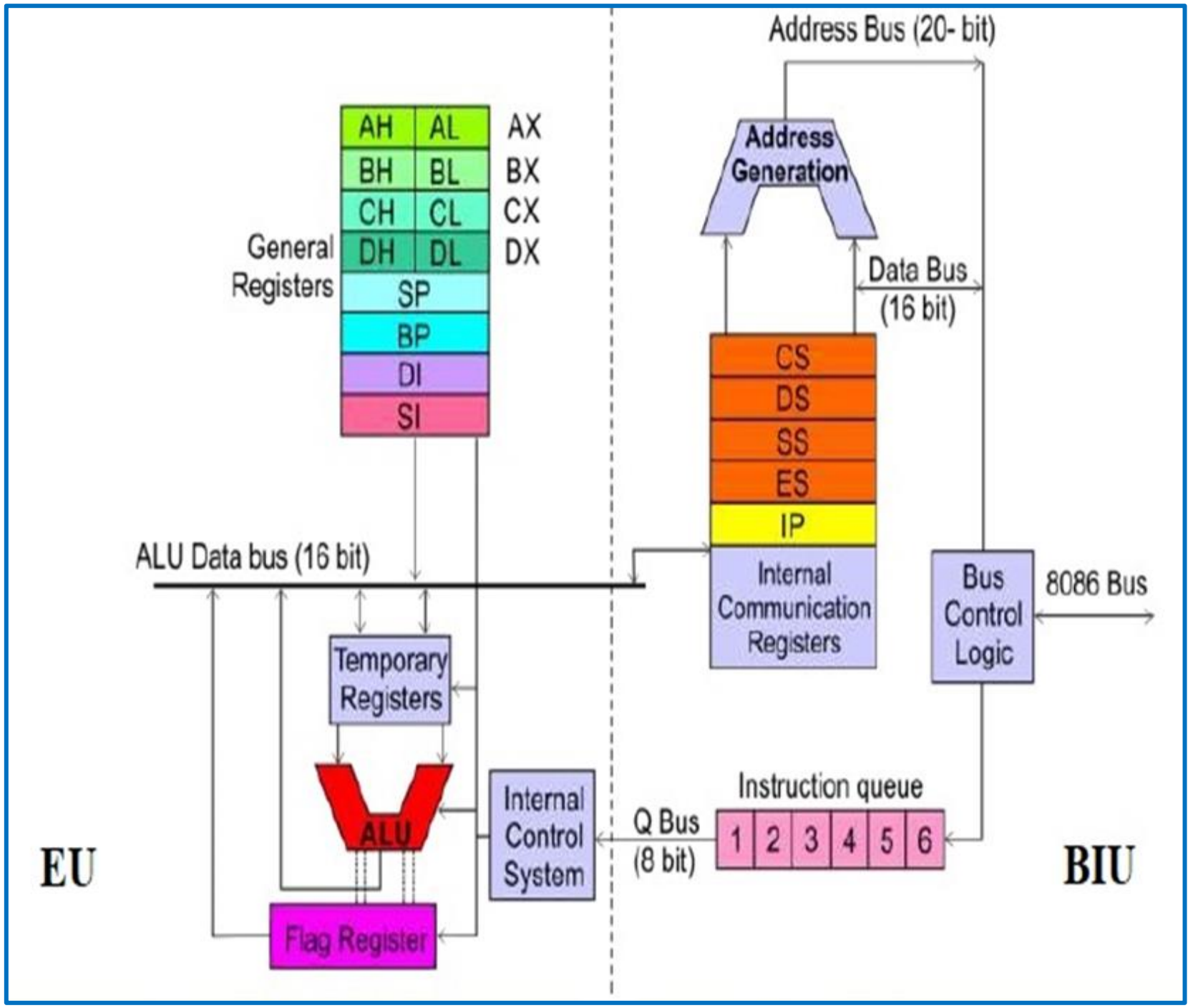


Fig. 3: Architecture of 8086 Microprocessor

Bus Interface Unit (BIU)

- Bus Interface Unit (BIU) provides a full 16 bit bidirectional data bus and 20 bit address bus.
- The Bus Interface Unit connects the microprocessor to external devices.

BIU performs following operations:

- Instruction fetching.
- Reading and writing data of data operands for memory.
- Inputting/outputting data for input/output peripherals. And other functions related to instruction and data acquisition.
- To implement above functions, the BIU contains the segment registers, the instruction pointer, address generation adder, bus control logic, and an instruction queue.
- The BIU uses a mechanism known as an instruction stream queue to implement pipeline architecture.

Execution Unit (EU)

- The Execution Unit is responsible for decoding and executing all instructions.
- The EU consists of arithmetic logic unit (ALU), status and control flags, general-purpose registers, and temporary-operand registers.
- 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 by 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.

Pipelining Architecture in 8086mp:

- While the EU is decoding an instruction or executing an instruction, which does not require use of the buses, the BIU fetches up to six instruction bytes for the following instructions.
- The BIU stores these pre-fetched bytes in a first-in-first-out register set called a **queue**.
- When the EU is ready for its next instruction from the queue in the BIU. This is much faster than sending out an address to the system memory and waiting for memory to send back the next instruction byte or bytes.

Pipelining Architecture in 8086mp:

- While the EU is decoding an instruction or executing an instruction, which does not require use of the buses, the BIU fetches up to six instruction bytes for the following instructions.
- The BIU stores these pre-fetched bytes in a first-in-first-out register set called a **queue**.
- When the EU is ready for its next instruction from the queue in the BIU. This is much faster than sending out an address to the system memory and waiting for memory to send back the next instruction byte or bytes.

Pipelining Architecture in 8086mp:

- Except in the case of JMP and CALL instructions, where the queue must be dumped and then reloaded starting from a new address, this pre-fetch and queue scheme greatly speeds up processing..
- Fetching the next instruction while the current instruction executes is called **pipelining**.

Register Organization

- The 14 registers of 8086 microprocessor are categorized into four groups. They are general purpose data registers, Pointer & Index registers, Segment registers, Instruction register, and Flag register as shown in the table below.

S.NO	Type	Register width	Name of the Registers
1	Data Register	16-bit	AX,BX,CX,DX
		8-bit	AL,AH,BL,BH,CL,CH, DL,DH
2	Pointer Registers	16-bit	Stack Pointer (SP) Base Pointer (BP)
3	Index Registers	16-bit	Source Index (SI) Destination Index (DI)
4	Segment Registers	16-bit	Code Segment (CS) Data Segment (DS) Stack Segment (SS) Extra Segment (ES)
5	Instruction	16-bit	Instruction Pointer (IP)
6	Flag	16-bit	Flag Register

1. General Purpose Registers

- 8086 CPU has 8 general purpose registers; these registers can be divided into:

a) **Data registers:** four 16 bits data registers

b) **Pointer and index registers:** two 16 bits pointer registers and two 16 bits index registers.

1. General Purpose Registers

a) Data registers

- They are four registers (AX, BX, CX, and DX) which used for arithmetic and data movement.
- Each register can be addressed as either 16-bit or 8 bit value. Example, AX register is a 16-bit register, its upper 8-bit is called AH, and its lower 8-bit is called AL.
- Bit 0 in AL corresponds to bit 0 in AX and bit 0 in AH corresponds to bit 8 in AX as shown in the figure below.

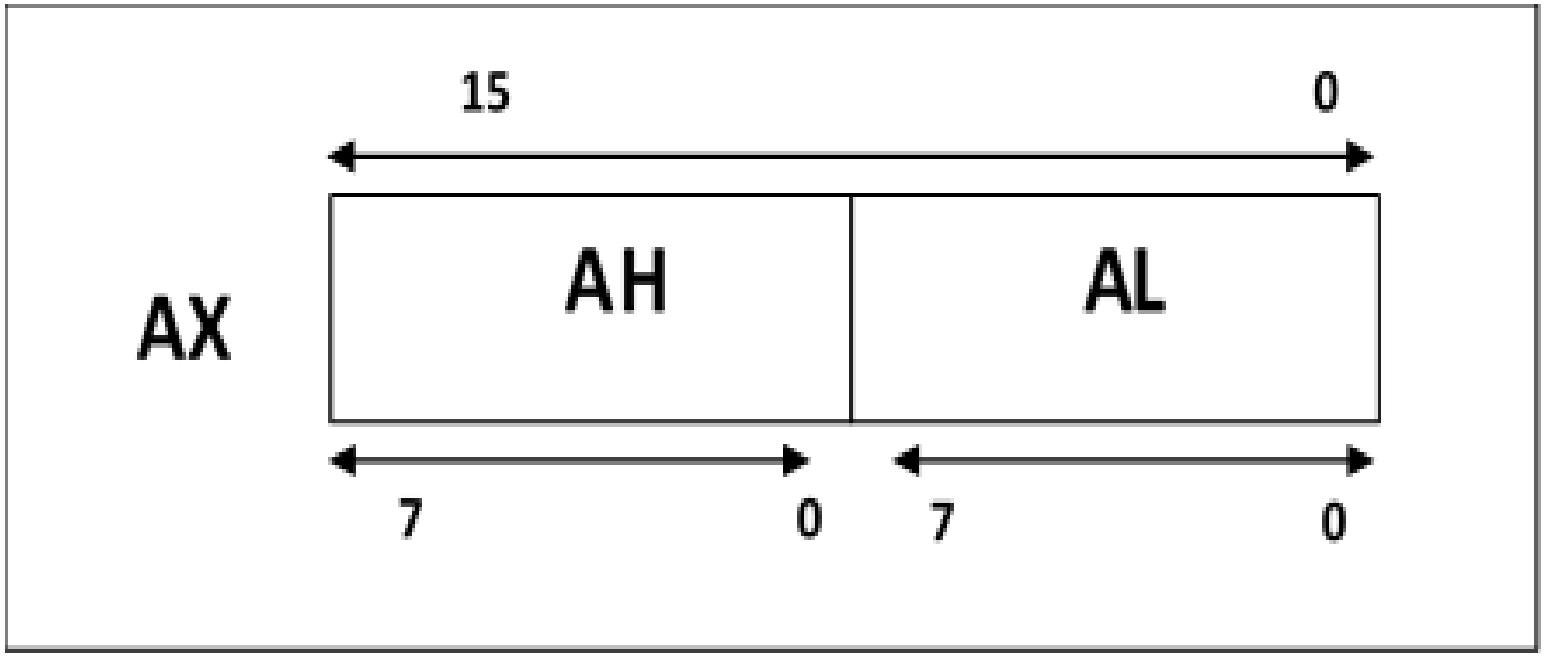


Fig. 4: AX register

a) Data registers

- **Accumulator Register (AX):** It is the accumulator register because it is favored by the CPU for arithmetic operations. Other operations are also slightly more efficient when performed using AX.
- **Base Register (BX):** The BX register can hold the address of a procedure or variable. Three other registers with this ability are SI, DI and BP. The BX register usually contains a data pointer used for based, based indexed or register indirect addressing. BX register can also perform arithmetic and data movement.

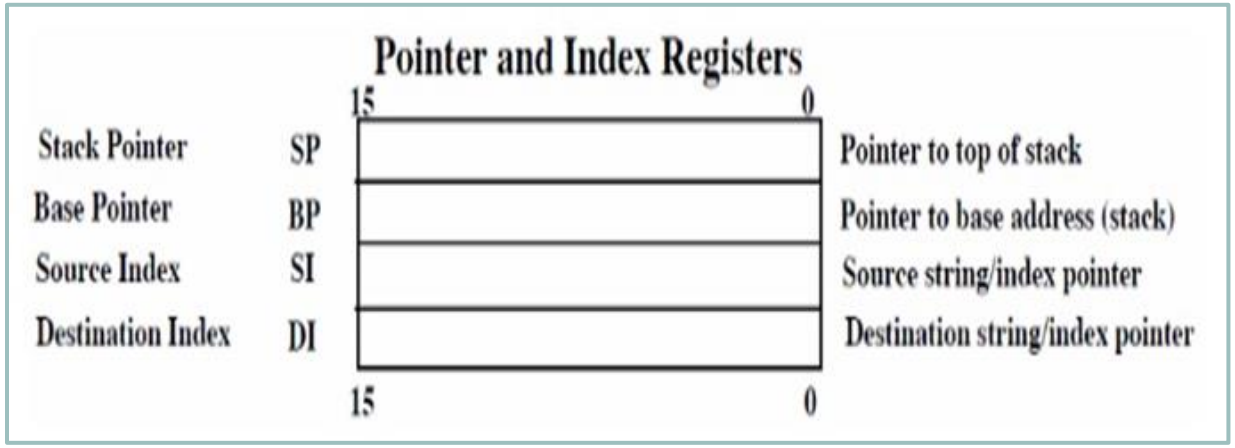
a) Data registers

- **Count Register (CX):** The CX register acts as a counter for repeating or looping instructions. These instructions automatically repeat and decrement CX.
- **Data Register (DX):** In integer 32-bit multiply and divide instruction the DX register contains high order word of the resulting number. DX register can be used as a port number in I/O operations.

1. General Purpose Registers

b) Pointer and index registers

- There are four 16-bits registers two serve as pointers and two serve as indexes. These registers usually store offset address used for addressing within the segment.



b) Index and Pointer Register

- **Source Index (SI):** is a 16-bit register. SI is used for indexed, based indexed and register indirect addressing. As well as source data address in string manipulation instructions. Used in conjunction with DS register to point to data locations in the data segment.
- **Destination Index (DI):** is a 16-bit register. Used with the ES register in string operations. DI is used for indexed, based indexed and register indirect addressing, as well as a destination data address in string manipulation instructions..

b) Index and Pointer Register

- **Source Index (SI):** is a 16-bit register. SI is used for indexed, based indexed and register indirect addressing. As well as source data address in string manipulation instructions. Used in conjunction with DS register to point to data locations in the data segment.
- **Destination Index (DI):** is a 16-bit register. Used with the ES register in string operations. DI is used for indexed, based indexed and register indirect addressing, as well as a destination data address in string manipulation instructions..

b) Index and Pointer Register

- **Stack Pointer (SP):** is a 16-bit register pointing to stack, it is used to hold the address of the top of the stack. The stack is maintained as LIFO with its bottom at the start of the stack segment (Specified by the SS segment register). Unlike the SP register, the BP can be used to specify the offset of other program segments.
- **Base Pointer (BP):** is a 16-bit register pointing to stack segment. It is usually used by subroutine to locate variables that were passed on stack by calling program. BP register is usually used for based, based indexed or register indirect addressing.

2. Segment Registers

- Within the 1 MB of memory space the 8086/88 defines 16 segments, but four 64K-byte memory blocks are active at a time called the code segment, stack segment, data segment, and extra segment.
- Each of these blocks of memory is used differently by the processor.
- The four segment registers (CS, DS, ES, and SS) are used to "point" at location 0 (the base address) of each segment as shown in fig. (5)

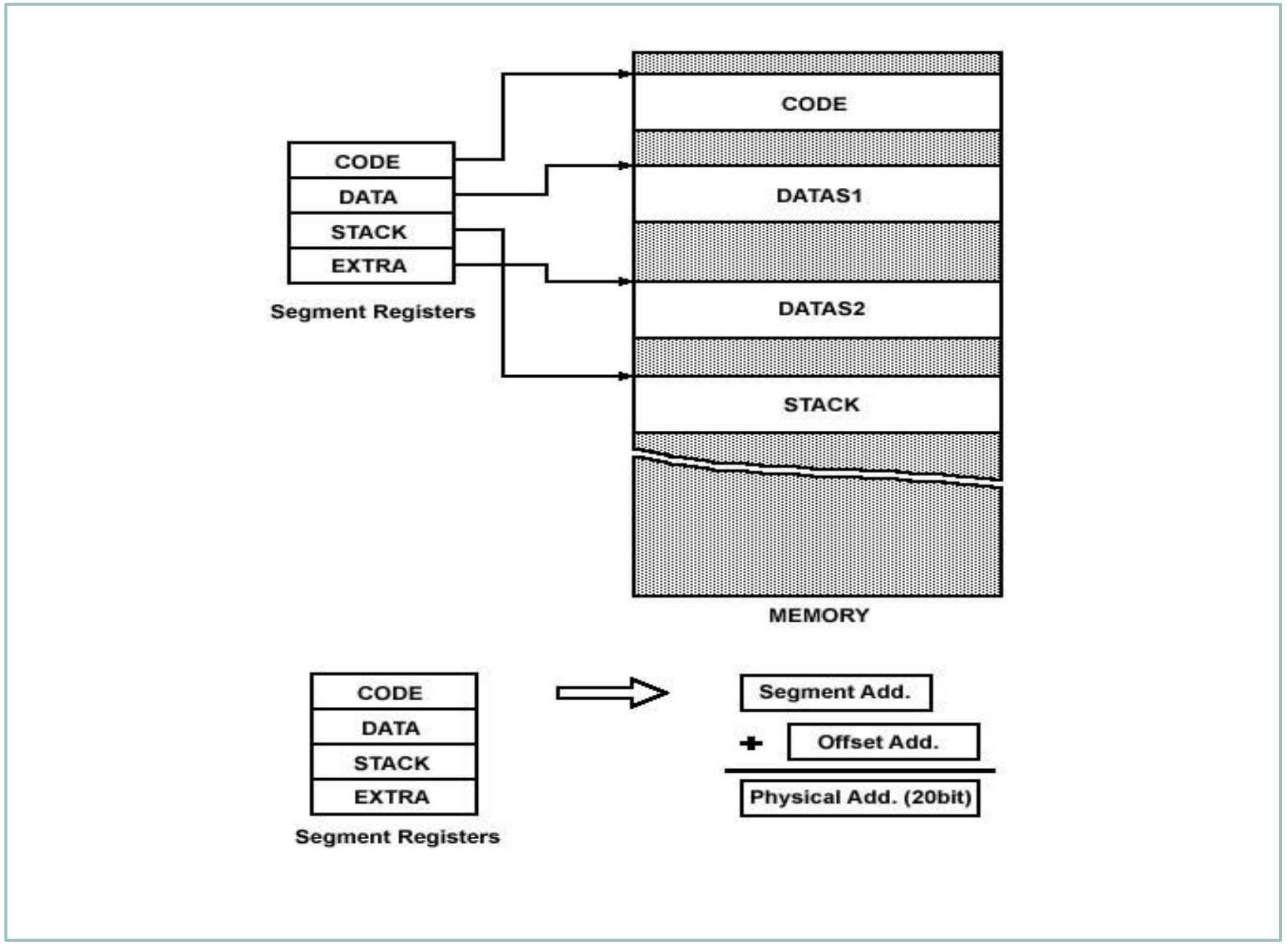


Fig. 5: Physical Memory Organization

2. Segment Registers

- **Code Segment (CS):** is a 16-bit register containing address of 64 KB segment with program instructions. CS register cannot be changed directly. The CS register is automatically updated during far jump, far call and far return instructions. .
- **Stack segment (SS):** is a 16-bit register containing address of 64KB segment with program stack. The stack segment in memory which the value of instruction pointer, status flags, and other registers are pushed in case of interrupt or subroutine call.

2. Segment Registers

- **Data segment (DS):** is a 16-bit register containing the starting address of the current data segment in which data are stored. It provides a read/write memory space.
- **Extra segment (ES):** used to hold the starting address of Extra segment. Extra segment is provided for programs that need to access a second data segment. Segment registers cannot be used in arithmetic operations..

3. Instruction Pointer (IP)

- Is a 16-bit register. This is important register which is used to control which instruction the CPU executes.
- The IP, or program counter, is used to store the memory location of the next instruction to be executed (offset address relative to CS).
- The CPU checks the IP to ascertain which instruction to carry out next, and then updates the IP to point to the next instruction.
- Thus the IP will always point to the next instruction to be executed.

4. Flag Register

- Determines the current state of the processor.
- They are modified automatically by CPU after mathematical operations, this allows to determine the type of the result, and to determine conditions to transfer control to other parts of the program.
- 8086 has 9 flags and they are divided into two categories as shown in figure below.

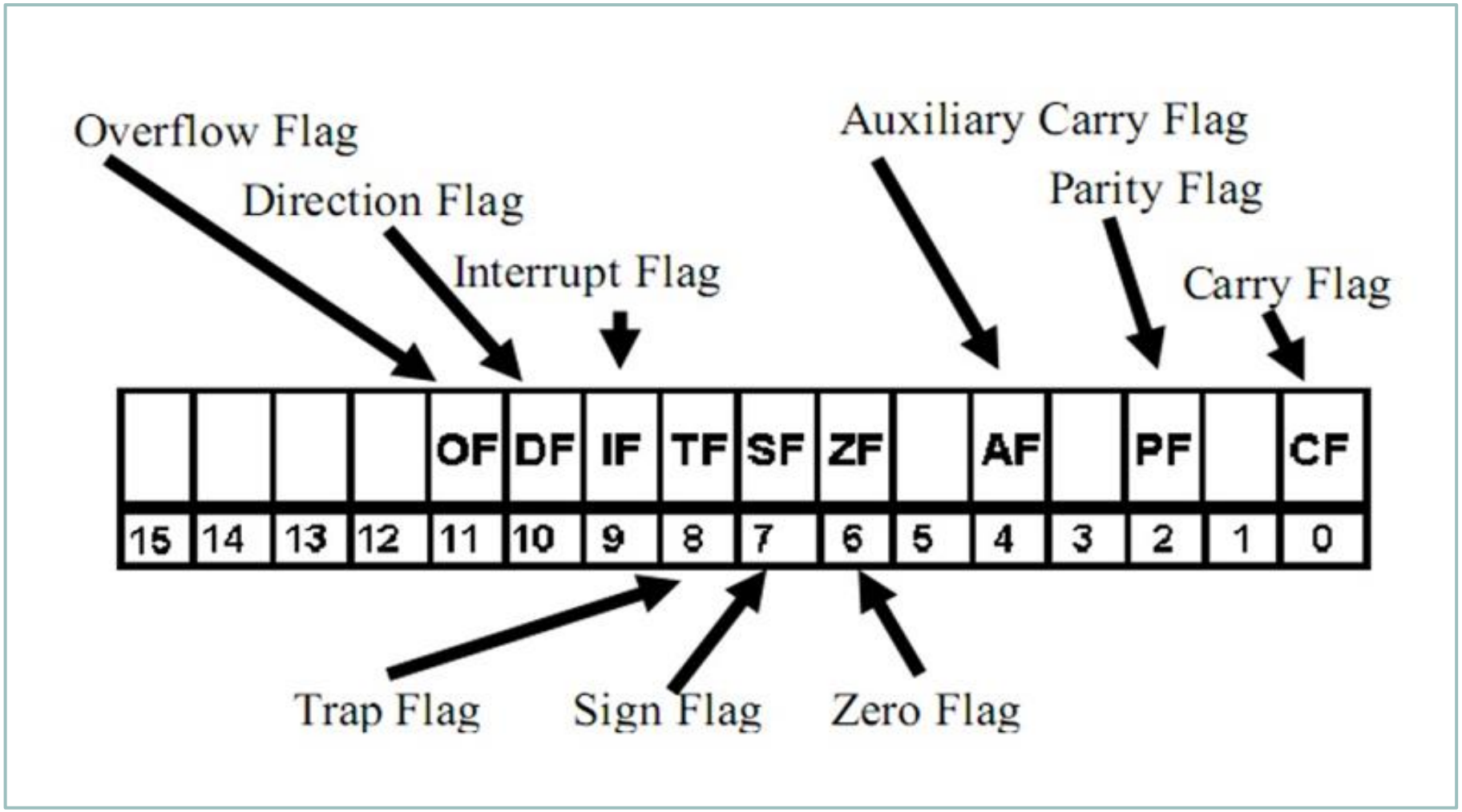


Fig. 6: Flag Register

1. Status Flags

The status flags are set/reset depending on the results of some arithmetic or logical operations during program execution:

- **Carry Flag (CF):** this flag is set to 1 if there is a carry out or borrow in for the most significant bit of the result during the execution of an arithmetic instruction otherwise, CF is reset.
- **Auxiliary Flag (AF):** If an operation performed in ALU generates a carry/barrow from lower nibble (i.e. D0 D3) to upper nibble (i.e. D4 – D7), the AF flag is set i.e. carry given by D3 bit to D4 is AF flag. This is not a general-purpose flag, it is used internally by the processor to perform Binary to BCD conversion.

1. Status Flags

- **Parity Flag (PF):** This flag is used to indicate the parity of result. If lower order 8-bits of the result contains even number of 1‟s, the Parity Flag is set and for odd number of 1‟s, the Parity Flag is reset.
- **Zero Flag (ZF):** It is set; if the result of arithmetic or logical operation is zero else it is reset.
- **Sign Flag (SF):** In sign magnitude format the sign of number is indicated by MSB bit. If the result of operation is negative, sign flag is set.
- **Overflow Flag (OF):** It occurs when signed numbers overflow. An OF indicates that the result has exceeded the capacity of machine.

1. Status Flags

❖ **Example:** Suppose AL= CFH if we execute the instruction ADD AL, C1H

AL	1100	1111	CF=1	AF=1	PF=1
C1H	1100	0001	ZF=0	SF=1	OF=0
90H	1001	0000			

2. Control Flags

Control flags are set or reset deliberately to control the operations of the execution unit.

- **Trap Flag (TP):** It is used for single step control. It allows user to execute one instruction of a program at a time for debugging. When trap flag is set, program can be run in single step mode.
- **Interrupt Flag (IF):** It is an interrupt enable/disable flag. If it is set, the maskable interrupt of 8086 is enabled and if it is reset, the interrupt is disabled. It can be set by executing instruction STI and can be cleared by executing CLI instruction.
- **Direction Flag (DF):** It is used in string operation. If it is set, string bytes are accessed from higher memory address to lower memory address. When it is reset, the string bytes are accessed from lower memory address to higher memory address.

Memory Segmentation:

- The memory in an 8086 based system is organized as segmented memory.
- The CPU 8086 is able to access 1MB of physical memory. The complete 1MB of memory can be divided into 16 segments, each of 64KB size and is addressed by one of the segment register.
- The 16-bit contents of the segment register actually point to the starting location of a particular segment. The address of the segments may be assigned as 0000H to F000h respectively.
- To address a specific memory location within a segment, we need an offset address. The offset address values are from 0000H to FFFFH so that the physical addresses range from 00000H to FFFFFH.
- A program can have more than four segments, but can only access four segments at a time.

Physical address is calculated as below:

Ex:

Segment address =1005H

Offset address =5555H

Segment address =1005H = 0001 0000 0000 0101

Shifted left by 4 Positions=0001 0000 0000 0101 0000 + Offset address = 5555H= 0101 0101 0101 0101

Physical address=155A5H =0001 0101 0101 1010 0101

Physical address = Segment address * 10H + Offset address.

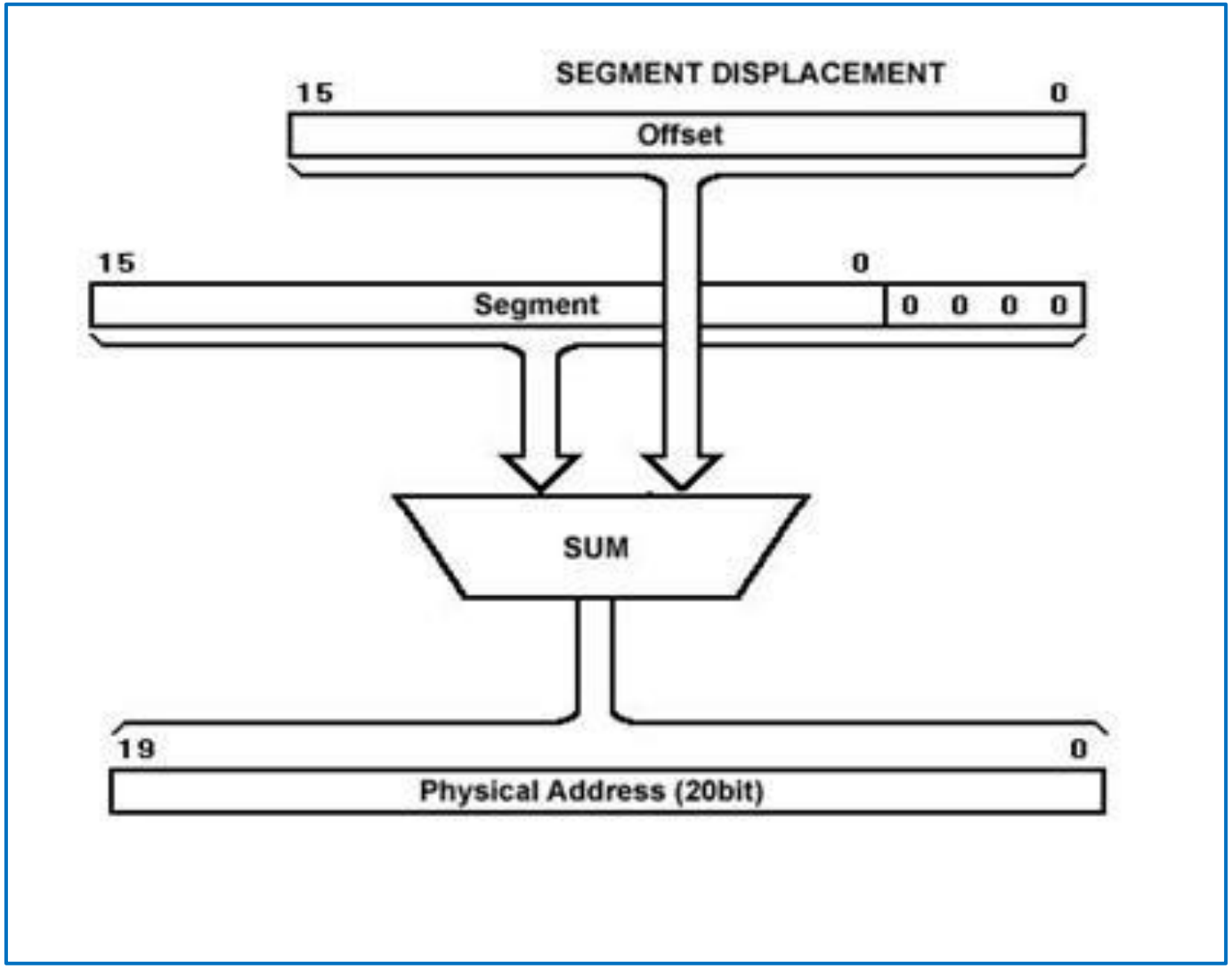


Fig. 7: Generating a Physical Address

The main advantages of the segmented memory scheme 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.