8.4 8086 REGISTER ORGANISATION

The 8086 microprocessor contains fourteen registers of 16-bit each. Figure 8.2 shows the register organization of 8086. The registers are classified into the following category as given below:

- General purpose registers
- Pointers
- Index Registers
- Segment Registers
- Instruction Pointer and Status register

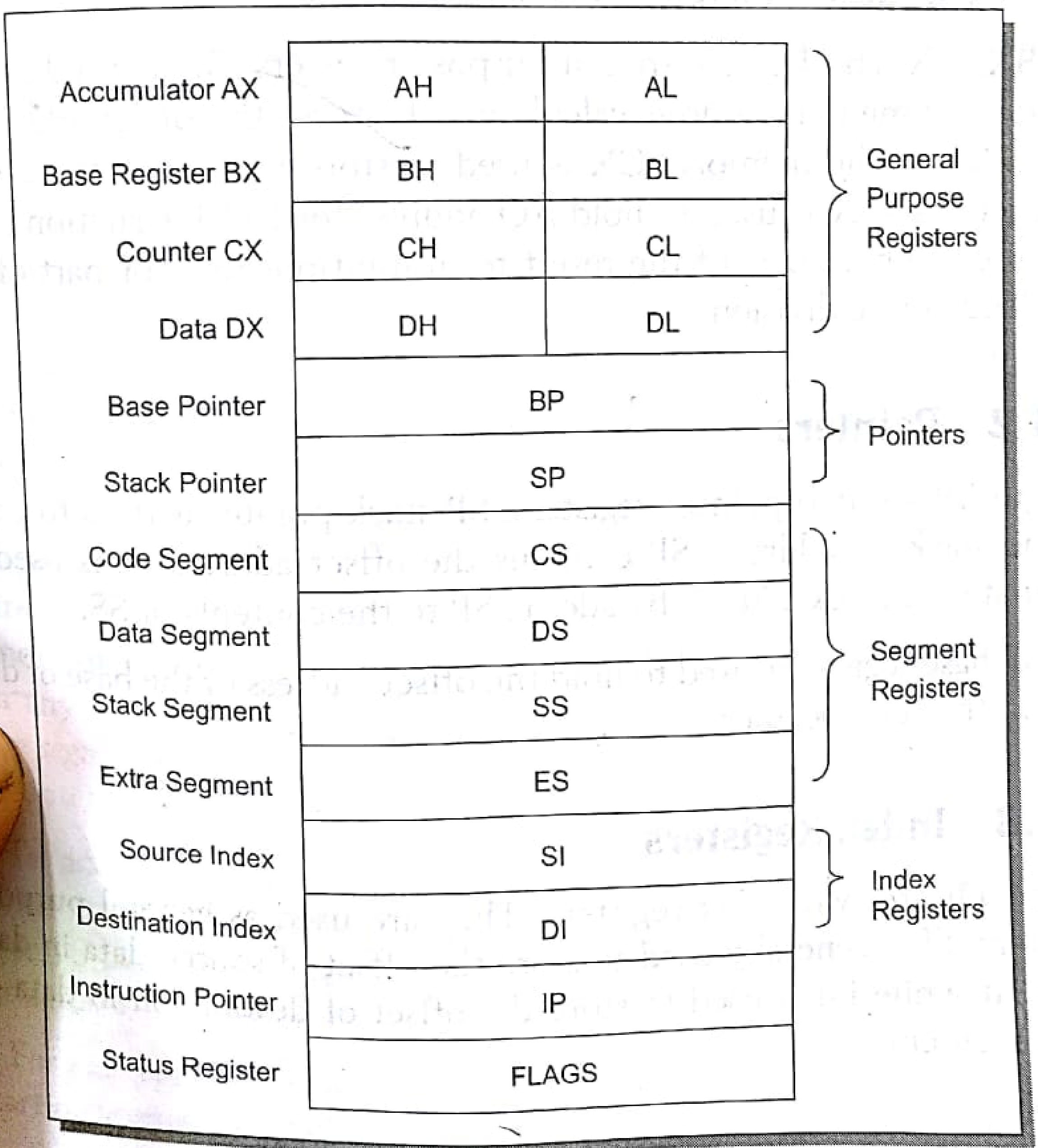

Figure 8.2 Register Organisation of 8086

8.4.1 General purpose registers

The general purpose registers are : AX, BX, CX and DX. Each of the register is referred as 16-bit register or 8-bit register as shown in the Table 8.4.

Table 8.4 *General purpose registers of 8086*

16-bit registers	high order 8-bit registers	low order 8-bit registers
AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL

AX (Accumulator) is a multipurpose register. It is used for instructions such as multiplication and division.

BX, CX, and DX are special purpose registers. BX is used as base register for memory address calculation. It stores the offset address of a location in the memory. CX is used to store the count for various instructions. DX is used to hold I/O address for I/O instructions. It is also used to hold part of the result from multiplication or part of the dividend before division.

8.4.2 Pointers

SP and BP are two pointer registers. SP stack pointer is used to point to the stacktop address. SP contains the offset address. It is used to calculate the stack address by adding SP to the contents of SS.

BP base register is used to hold the offset address of the base of data area in the stack segment.

8.4.3 Index Registers

SI and DI are two index registers. They are used as general purpose registers. SI is generally used to store the offset of source data in data segment while DI is used to store the offset of destination in data or extra segment.

8.4.4 Segment Registers

8086 has 1 MB of memory which it can address. This memory is divided into four type of segments and each segment contains 64 KB of memory :

1. Code segment
2. Data segment
3. Stack segment
4. Extra segment

Code segment is a section of memory that contains the code of a program. Data segment contains most of the data used by a program. Extra segment is used as additional data segment for some string instructions to hold destination data. Stack segment is a section of memory kept aside for stack.

Each of the segment is of 64 K bytes. Thus to access any memory location only 16-bit effective address is generated. 8086 contains 4 segment registers:

1. Code segment register CS
2. Data segment register DS
3. Stack segment register SS
4. Extra segment register ES

The segment registers contain the starting address of the four segments in memory. 8086 has 20 lines for memory addressing, so it can address upto 2^{20} = 1MB of memory. But within 1MB of memory, 8086 can work with only four segments which are 64K bytes of size. Figure 8.3 shows the example of memory segmentation. Within 1MB of memory there can be several sets of four segments. The CPU is working with any four segments at any given time. These segments are called *active segments*. The CPU can work with other set of four segments, if the contents of the segment registers are changed.

To access a memory location , 8086 uses a 20-bit address. This address is the physical address. The segment registers contains the upper 16-bits of the starting address of a segment. The CPU appends four zeros to the lowest four bits of the 20-bit address.

For example: if the segment register contains 2000H, it can address a starting segment at location 20000H. Likewise, if the segment register contains 1234H, it addresses a memory segment beginning at location 12340H. W

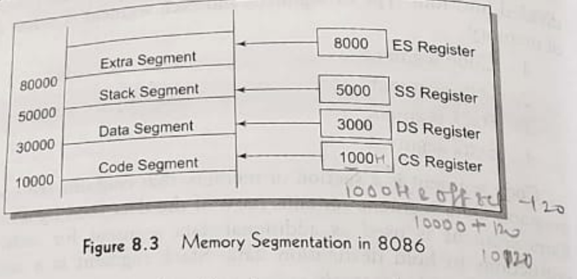

Figure 8.3 Memory Segmentation in 8086

While addressing a location in memory, the physical address is calculated by using the segment address and the offset. Offset or effective address determines how far is the memory location from the starting address of the segment register. To address a location, 8086 generates a 20-bit physical address. The contents of segments register are shifted left by 4-bits and the offset is added to it which generates the 20-bit physical address. For example:

CS=1300 H and offset=100, the 20-bit physical address is calculated as below:

CS contents are shifted left by 4-bits = 13000H

Physical address = 13000 + 100

= 13100 H

8.4.5 Instruction Pointer

IP Instruction pointer holds the offset address. It is used to point to the next instruction in a program currently located in the code segment. It gets incremented automatically as the program goes on executing.

8.4.6 Status Register / Flag Register

It is a 16-bit register also known as program counter or PSW (program status word). There are total nine flags . The rightmost five flags bits and the overflow flag bit change depending on the result of some arithmetic or logical operation. The remaining three flag bits are used

for control features in the microprocessor. Figure 8.4 shows the status flag of 8086.:

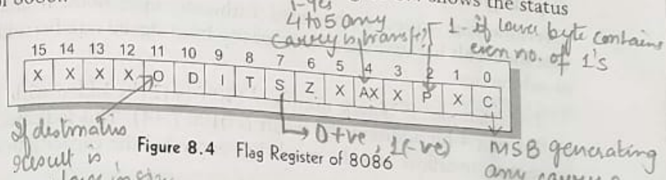

Figure 8.4 Flag Register of 8086

C (Carry)

It holds the carry resulting from addition or borrow after subtraction.

P (Parity)

It is set to 1 for even parity and 0 for odd parity.

AX (Auxillary Carry)

It is set to 1 when there is a carry after addition or borrow after subtraction between bits 4 and 3 of the result.

Z (Zero)

When Z=1, it shows that the result of an arithmetic or logic operation is zero. And when Z=0, the result is not zero.

S (Sign)

It holds the result is Positive or Negative after arithmetic or loic operation occurs. When S=0, the result is positive. And when S=1, the result is negative.

Control Registers

T (Trap)

It is used to allow the user to execute the program one instruction at a time for debugging.

I (Interrupt)

It is used to control the maskable interrupts in the system coming through INTR pin of 8086. If I=0, the interrupts are disabled. And if I=1, interrupts are ~~disabled~~ enabled. [maskable interrupts are recognized by CPU]

D (Direction)

It is used during String operations. If D=1, the DI or SI registers get decremented automatically. And if D=0, the registers are incremented automatically. [lowest addr to higher addr]

O (Overflow)

Overflow flag is used with signed arithmetic operations. It indicates that the result has exceeded the capacity from signed number addition or subtraction.

For example: If binary numbers 1100 (+12) is added with 1000 (+8) using 4-bit addition the result is 0100 (+4). This indicates that there is an overflow and the result has exceeded the capacity to store the data.