11.4.2 Flag Register

Q. Explain the flag register of 8086.

- Flag register is a part of EU. It is a 16-bit register with each bit corresponding to a flipflop.
- A flag is a flipflop. It indicates some condition produced by the execution of an instruction. For example the zero flag (ZF) will set if the result of execution of an instruction is zero.
- A flag can control certain operations of EU.
- Fig. 11.4.1 shows the details of the 16 bit flag register of 8086 CPU. As shown, it consists of nine active flags out of sixteen
- The remaining seven flags marked "U" are undefined flags.

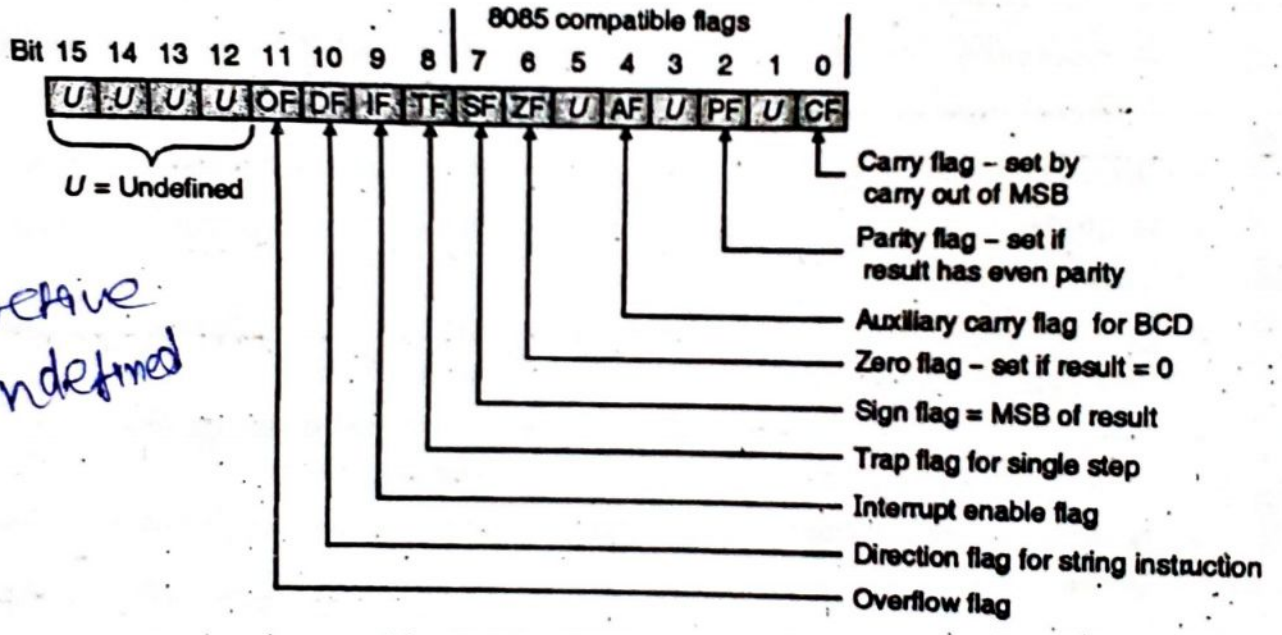


Fig. 11.4.1 : 8086 flag register format

Active flags

There are nine active flags out of 16, in the 8086 flag register. The remaining are undefined flags.

Control flags

- Out of the nine active flags, six are conditional (status) flags and the remaining three are called as the control flags, because they are used to control certain operations of the processor.
- The three control flags are :

  - The trap flag (TF)
  - The interrupt flag (IF)
  - The direction flag (DF)

1. **The trap flag (TF)**

   (i) Setting TF puts the processor into single step mode for debugging. In stepping, microprocessor executes a instruction and enters into single ste

   (ii) After that user can check registers or memory contents, if found ok, he/s proceed further, else necessary action will be taken. **This utility is, to the program.** If TF = 1, the CPU automatically gene es an in interrupt after each instruction, allowing a program to be inspected executes instruction by instruction.

   Used by debuggers for single step operation.

   TF = '1' - Trap on, TF = '0' - Trap off

2. **The interrupt flag (IF)**

   (i) If user sets IF flag, the CPU will recognize external (maskable) inte requests.

   (ii) Clearing IF disables these interrupts.

   (iii) IF has no effect on either non-maskable external or internally gener interrupt. The interrupt flag (IF) is used for allowing or prohibiting interruption of a program.

   IF = '1' - Interrupt enabled, IF = '0' - Interrupt disabled

3. **The direction flag (DF)**

   (i) This bit is specifically for string instructions. In string instruction we use (source index) and DI (destination index) registers as offset registers to po source area and destination area respectively. DF flag controls direction o and DI pointers.

   (ii) If DF = 1, the string instruction will automatically decrement the pointer i.e. process string from high addresses to low addresses, or from right to left.

   (iii) If DF = 0, the string instruction will automatically increment the pointer i.e. process string from low addresses to high addresses or from left to right.

   It is used with string instructions

   DF = '1' - Up , DF = '0' - Down

Conditional (status) flags

- Six flags out of these nine active flags indicate status of the result produced after execution of an instruction. Such flags are called as the conditional flags.
- The names of the conditional flags indicate what conditions affect th For example the carry flag (CF) is set to 1 if addition of two numbers produce carry output.
- Some of the 8086 instructions check the status of the conditional flags, be execution.
- The six conditional flags are :

• The parity flag (PF)	• The zero flag (ZF)
• The sign flag (SF)	• The auxiliary carry flag (AF)
• The carry flag (CF)	• The overflow flag (OF)

1. **The parity flag (PF)**
   (i) This flag is set when the result has even parity, an even number of 1 bits.
   (ii) If parity is odd, PF is resetted.
   (iii) This flag is normally used to check for data transmission errors.
   PF = '1' - low byte has an even number of 1 bits, PF = '0' - low byte has odd parity

2. **The zero flag (ZF)**
   This flag is set, when the result of operation is zero, else it is reset.
   ZF = '1' - zero result, ZF = '0' - non-zero result

3. **The sign flag (SF)**
   (i) This flag is set, when MSB (most significant bit) of the result is 1.
   (ii) In other words it copies the MSB of the result
   (iii) Since negative binary numbers are represented (in the 8086 CPU) using standard two's complement notation, the MSB (copied in sign flag) indicates the number is positive or negative.
   (iv) SF indicates sign of the result only in case of signed operation
   (v) In case of unsigned operation, sign bit has no significance.
   SF = '1' - msb is 1 (negative), SF = '0' - msb is 0 (positive)

4. **The auxiliary carry flag (AF)**
   (i) It is a carry from lower digit to upper digit
   (ii) This flag is set, whenever there has been a carry out of the low nibble into the high nibble or a borrow from high nibble into the low nibble of an 8 bit quantity else AF is reset.
   (iii) This flag is used by decimal arithmetic instructions.
   AF = '1' - carry out from bit 3 on addition or borrow into bit 3 on addition
   AF = '0' - no carry out from bit 3 on addition nor borrow into bit 3 on addition

5. **The carry flag (CF)**
   (i) It can also be called as a final carry
   (ii) This flag is set whenever there has been a carry out of, or a borrow into, the MSB of the result (8 or 16 bit).
   (iii) The flag is used by the instructions that add and subtract multibyte numbers.
   (iv) Rotate instructions can also isolate a bit in memory or a register by placing it in the carry flag.
   CF = '1' - there is a carry out from the most significant bit (MSB),
   CF = '0' - no carry out from msb

6. **The overflow flag (OF)**
   (i) It indicates an overflow from the magnitude to the sign bit of result
   (ii) If OF is set, an arithmetic overflow has occurred; that is, a significant bit has been lost because the size of the result exceeded the capacity of its destination location.
   (iii) In 8086 interrupt on overflow instruction is available that will generate an interrupt in this situation. If result is not out of range, OF remains reset.
   OF = '1' - signed overflow occurred, OF = '0' - no overflow

Hence we can say for the following bit structure of data, the flags will be affected as indicated in the table

D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

Flags name	8-bit operation	16-bit operation
Auxiliary carry	Carry from D_3 to D_4	Carry from D_3 to D_4
Carry	Carry from D_7 to D_8	Carry from D_{15} to D_{16}
Overflow	Carry from D_6 to D_7	Carry from D_{14} to D_{15}
Sign	Copy of D_7	Copy of D_{15}

11.4.3 General Purpose Registers

Q. Explain the dedicate use of GPR.

- As shown in Fig. 11.4.1, the execution unit (EU) has four general purpose 16-bit registers.
- Each one of them can be used for temporary storage of 8 bit data, 16-bit data or 32-bit data.
- 8-bit Registers - AH, AL, BH, BL, CH, CL, DH, DL. Any of these registers can be used as an 8-bit operand.
- 16-bit Registers - AX, BX, CX, DX, SI, DI, SP, BP. Any of these registers can be used as a 16-bit operand.
- 32-bit Registers - DX : AX together can be used for 32-bit operand.
- These registers can be used for general purpose computing when their other specialized functions do not interfere.

Can we store 16 bit data words in the general purpose registers ?

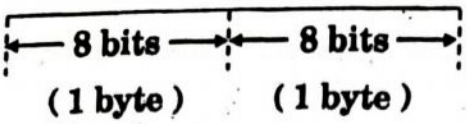
- The answer is yes. We can use certain pairs of the general purpose registers to store 16 bit data words.
- Such register pairs are AH and AL, BH and BL, CH and CL, DH and DL.
- The above mentioned register pairs are referred to as follows.

Pair of general purpose registers	Referred to as
1. Pair of AH and AL	AX register
2. Pair of BH and BL	BX register
3. Pair of CH and CL	CX register
4. Pair of DH and DL	DX register

for 16 bit

- In short, the four general-purpose 16-bit data registers : AX, BX, CX, and DX, each of these is a combination of two 8-bit registers which are separately accessible as AL, BL, CL, DL (the "low" bytes) and AH, BH, CH, and DH (the "high" bytes).

AX: Accumulator
BX: Base Register
CX: Count Register
DX: Data Register



- For example, if AX contains the 16-bit number 1234H, then AL contains 34H and AH contains 12H.
- The upper and lower halves of the data registers are separately addressable. This means that each data register can be used as a single 16-bit register or as two 8-bit registers.

Special functions of general purpose registers

Although the first four registers AX, BX, CX, and DX are called as "general-purpose", each of them is designed to play a particular role in common use:

1. Register AX

AX is the "16-bit accumulator" while AL is "8-bit accumulator"

Accumulator has the following special functions :

(i) Some of the operations, such as Multiplication and Division, require that one of the operands be in the accumulator and also the result is stored in accumulator.

Some other operations, such as Addition and Subtraction, may be applied to any of the registers (that is, any of the eight general- and special-purpose registers) but are more efficient when working with the accumulator.

(ii) It works as a via register for I/O accesses i.e. a data is routed through accumulator for the communication of the processor and I/O devices.

For OUT instruction the data in accumulator (AL for 8-bit data and AX for 16-bit data) can only be given to the output device

For IN instruction the data taken from the input device can be taken only in accumulator (AL for 8-bit data and AX for 16-bit data)

(iii) It also works as a via register for string instructions. Whenever a data is to be brought from memory or given to memory in case of string operations it is routed through accumulator only.

2. Register BX

BX is the "base" register;

(i) It is the only general-purpose register which may be used for indirect addressing. (various addressing modes are discussed in chapter 4.)

(ii) For example, the instruction MOV [BX], AX causes the contents of AX to be stored in the memory location whose address is given in BX.

3. Register CX

CX is the "count" register. It works as a default counter register for three instructions viz :

(i) The looping instructions (LOOP, LOOPE, and LOOPNE), to indicate the number of iterations

(ii) The shift and rotate instructions (RCL, RCR, ROL, ROR, SHL, SHR, and SAR), to indicate number of shifts or rotations (Here only CL is used and not entire CX)

(iii) The string instructions (with the prefixes REP, REPE, and REPNE) to in the size of the string block.

4. **Register DX**

DX is the "data" register

(i) It is used together with AX for the word-size MUL and DIV operations, the operand size is greater than the register AX i.e. operand is 32-bit

(ii) It also holds the port number for the IN and OUT instructions. For address accesses of I/O ports only DX can be used as a pointer.

Summary of Implicit use of General Purpose Registers

Registers	Operations
AX	Word multiply, Word divide, Word I/O and Word string
AL	Byte multiply, byte divide, byte I/O, byte string and decimal / ASCII arithmetic
BX	Store address information
CX	Counter for String operations and loops
CL	Counter for Variable shift and rotate
DX	Word multiply, word divide, Indirect I/O

11.4.4 Control Circuitry

The control circuit is a part of EU. It is used for directing the internal operations.

11.4.5 A Decoder

- "The process of translation from instructions into action is known as decoding"
- A decoder in the execution unit (EU) is used for translating the instructions fetch from the memory into a series of actions.
- The EU will actually carry out these actions.

11.4.6 Pointer and Index Register

- The execution unit also contains the following 16 bit registers.

• Base Pointer (BP) register	• Stack Pointer (SP) register
• Source Index (SI) register	• Destination Index (DI) register

These registers can be used as general purpose 16-bit registers. But mainly they used to hold the 16 bit offset of data word in one of the segments as given below :

Base Pointer (BP) register

- This is a 16 bit register in the E.U. It holds the 16 bit offset relative to the st segment (SS) register.
- But BP has a specific use. BP is used whenever we pass a parameter by way stack.
- The BP register can also be used as an offset register in the addressing mode call base addressing mode.