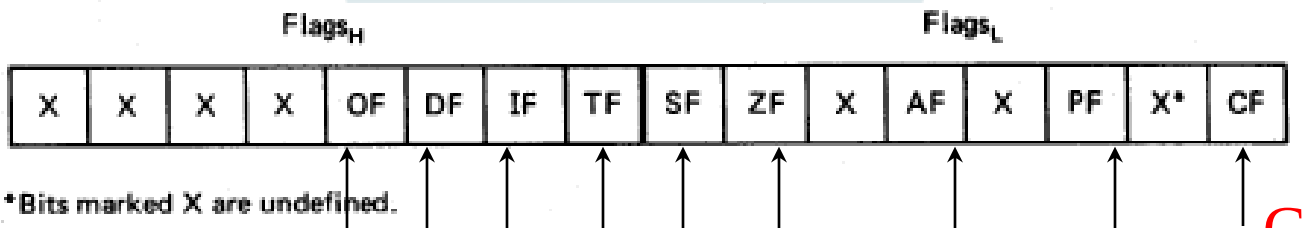
# *CSC222:* Computer Organization & Assembly Language

## 7 – Processor Status & Flags

# The FLAG Register

- Nine individual bits called as **flag** are used to represent the 8086 processor state.
- Flags are placed in **FLAG Register**.
- Two types of flags:
  - **Status Flags**: Reflects the result of a computation. Located in bits: 0, 2, 4, 6, 7 and 11.
  - **Control Flags**: Used to enable/disable certain operations of the processor. Located in bits 8, 9 and 10.

# Flags




Overflow

Direction

Interrupt enable

Trap

Sign

Zero

Auxiliary flag

Parity flag

Carry flag

6 are status flags

3 are control flag

| Flag Name | DEBUG Output | |
|---|---|---|
| | For Flag = 0 (CLEARED) | For Flag = 1 (SET) |
| Overflow | NV | OV |
| Direction | UP | DN |
| Interrupt | DI | EI |
| Sign | PL | NG |
| Zero | NZ | ZR |
| Auxiliary Carry | NA | AC |
| Parity | PO | PE |
| Carry | NC | CY |

Debug displays the flags in the following order:

*Overflow, Direction, Interrupt, Sign, Zero, Aux Carry, Parity, Carry*

# The Status Flags

1. **Carry Flag:**
   - Set (CY), Unset (NC)
   - The Carry Flag is set to 1 when there is a carry out from MSB on addition or there is a borrow into the MSB on subtraction. Also affected by shift and rotate instructions.
   - **Examples:**
   - 0FFh + 11h = 110h (If a register can store only 1 byte, then where to store the carry generated by MSB?)
   - 1000 0001b – 1000 0010b = 11111111b (How processor would know a borrow is required to perform subtraction?)

# Contd..

2. **Parity Flag**:

- Set (PE), Unset (PO)
- PE (Even Parity): If the low byte of a result has an even number of one bits. For Even parity, PF =1
- PO (Odd Parity): If the low byte of a result has odd number of one bits. For Even parity, PF = 0
- **Examples:**
- 1000 0001b – 1000 0010b = 11111111b (Number of one's in result = 8, so PF = 1)

# Contd..

3. **Auxiliary Carry Flag**:

- Set (AC), Unset (NA)
- The Auxiliary Carry Flag is set to 1 if there is a carry out from bit 3 on addition, or a borrow into bit 3 on subtraction.
- Used in Binary Coded Decimal (BCD) Operations.
- **Examples:**
- 1000 0001b – 0000 0010b = 01111111b (Borrow from bit 4 to bit 3)

# Contd..

4. **Zero Flag:**

- Set (ZR), Unset (NZ)
- Zero Flag is set when the result is zero.
- Zero Flag is unset when result is non zero.
- **Examples:**

  0FFh – 0FFh = 00h

# Contd..

5. **Sign Flag:**

- Set when MSB of a result is 1; it means the result is negative (signed interpretation)
- Unset when MSB is 0 i.e. result is positive.
- **Examples:**

  0FFh – 0FFh = 00h (MSB = 0, PF = 0)

# Contd..

6. **Overflow Flag:**
   - Set (OV), Unset (NV)
   - Set if signed overflow occurred, otherwise it is 0.
- **Overflow:**
  - Range of numbers that can be represented in a computer is limited.
  - If the result of an operation falls outside the defined range, Overflow occurs and the truncated result will be incorrect.
- Four possible outcomes of an arithmetic operation:
  - No Overflow
  - Only Signed Overflow
  - Only Unsigned Overflow
  - Both Signed and Unsigned Overflow

# Contd..

- Example of Unsigned Overflow Only:
  - AX = FFFFh, BX = 0001h
  - ADD AX, BX
  - Result: AX = 0000h but the correct answer is 10000h
  - But FFFFh = -1 and 0001h= 1, -1+1 = 0 so the stored answer is correct if consider signs as well. Therefore, no sign overflow occurred.

# Contd..

- Example of Signed Overflow Only:
  - AX = 7FFFh, BX = 7FFFh
  - ADD AX, BX
  - Result: AX = FFFEh (correct unsigned result, so unsigned overflow)
  - 7FFFh = 32767,
  - 32767 + 32767 = 65534, which is out of ranged for signed numbers, also FFFE = -2 so signed overflow occurred.

# How the Processor indicates overflow?

- The Processor sets:
  - Overflow Flag = 1 for Signed Overflow
  - Carry Flag = 1 for Unsigned Overflow

# How the Processor determines that overflow occurred?

- **Unsigned Overflow:**
  - Addition: Carry out from MSB
  - The correct Answer is largest than the biggest unsigned number FFFFh for a word and FFh for a byte.
  - Subtraction: Borrow into MSB
- **Signed Overflow:**
  - Addition: Same sign but sum has a different sign (e.g.: when you add two positive numbers and answer is negative)
  - Subtraction: If result has different sign than expected
- In Addition/Subtraction of two numbers with different signs, overflow is impossible. E.g.: A + (-B) = A – B

# How Instructions Affect the Flags

| Instructions | Affects Flags |
|---|---|
| MOV/XCHG | None |
| ADD/SUB | All |
| INC/DEC | All except CF |
| NEG | All (Carry Flag = 1 unless result is 0, Overflow Flag = 1 if word operand is 8000h, or byte operand is 80h |

# Example 5.1

- ADD AX, BX, where AX contains FFFFh, BX contains FFFFh
- **Solution**:
  - Actual Result = 1FFFEh
  - Result stored in AX = FFFEh
  - Flags:
    - SF = 1 because the MSB is 1
    - PF = 0 because there are 7 (odd number) of 1 bits in the low byte of the result.
    - ZF = 0 because nonzero result
    - CF = 1 because there is a carry out of the MSB on addition
    - OF = 0 because the sign of the stored result is the same as that of the numbers being added (in binary addition, there is a carry into the MSB and carry out from MSB also)

# Example 5.2

- ADD AL, BL where AL contains 80h, BL contains 80h
- **Solution:**
  - Actual Result = 100h
  - Result in AL = 00h
  - Flags:
    - SF = 0 because MSB is 0
    - PF = 1 because all bits in result are 0
    - ZF = 1 because result is 0
    - CF = 1 because there is a carry out from MSB
    - OF = 1 because the numbers being added are both negative but the MSB in result is 0 (in binary addition, there is a no carry into the MSB but there is carry out from MSB.

# Example 5.3

- SUB AX, BX where AX contains 8000h and BX contains 0001 h
- **Solution:**
  - Actual Result = Result in AX = 7FFFh
  - Flags:
    - SF = 0 because MSB is 0
    - PF = 1, Parity is Even because there are 8 one bits in the low byte of the result
    - ZF = 0 because result is nonzero
    - CF = 0 because a smaller unsigned number is being subtracted from a larger one
    - OF =1 because we are subtracting a positive number from a negative number but the result is positive (wrong sign of result)

# Example 5.4

- INC AL where AL contains FFh
- **Solution:**
  - Actual Result = 100h
  - Result stored in AL = 00h
  - Flags:
    - SF = 0
    - PF = 1
    - ZF = 1
    - CF = 0 because CF is unaffected by INC
    - OF = 0 because number of unlike sign are being added (there is a carry into the MSB and also carry out from the MSB)

# Example 5.5

- MOV AX, -5
- **Solution:**
  - Result in AX = -5 = FFFBh
  - None of the flags are affected by MOV

# Example 5.6

- NEG AX where AX contains 8000h
- **Solution:**
  - Result in AX = 8000h (2's complement)
  - SF = 1
  - PF = 1, in low byte of result, number of 1 bits is 0.
  - ZF = 0
  - CF = 1 because for NEG, CF is always 1 unless the result is zero
  - OF = 1 because there is no sign change

# Program

```
ORG 100h
.CODE
   MOV AX, 4000h          ;AX = 4000h
   ADD AX, AX             ;AX = 8000h
   SUB AX, 0FFFFh         ;AX = 8001h
   NEG AX                 ;AX = 7FFFh
   INC AX                 ;AX = 8000h
   MOV AH, 4Ch
   INT 21H
```

# Control Flags

1. **Direction Flag**: Controls the assumed direction used by string processing instructions. 1=Up, 0=Down
2. **Interrupt Flag**: Enable/Disable external interrupt.
3. **Trap Flag**: Determines whether or not CPU will be halted after each instruction.