**Flags**

In computer architecture, **flags** are special bits in a CPU's status register (often called the **flags register**) that provide information about the state of the processor and the results of arithmetic and logical operations. These flags help in controlling the flow of execution and are crucial for conditional operations and interrupt handling. There are total 9 flag registers, 6 are status flags and 3 are control flags.

**Status Flags**

Carry, overflow, zero, sign, auxiliary, parity

**Control Flags**

Direction, interrupt, Trap

**Bits**

3-bits = Tribit or triad

4-bits = Nibble

8-bits = Byte

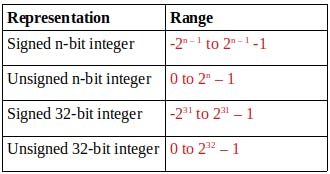
16, 32, 64-bits = Word

**Common Flags in the x86 Architecture**

1. **Carry Flag (CF)**:
   * **Description**: Indicates whether an arithmetic operation resulted in a carry out (for addition) or a borrow (for subtraction).
   * **Usage**: Useful for multi-precision arithmetic and detecting overflow in unsigned operations.
2. **Zero Flag (ZF)**:
   * **Description**: Set if the result of an operation is zero; otherwise, it is cleared.
   * **Usage**: Used to determine if an operation produced a zero result, which is useful for conditional jumps.
3. **Sign Flag (SF)**:
   * **Description**: Indicates the sign of the result of an operation (negative if the most significant bit is set).
   * **Usage**: Helps in determining the sign of the result of arithmetic operations.
4. **Overflow Flag (OF)**:
   * **Description**: Set if an arithmetic operation results in an overflow, meaning the result is too large to be represented in the destination operand.
   * **Usage**: Important for detecting errors in signed arithmetic operations.
5. **Parity Flag (PF)**:
   * **Description**: Set if the number of set bits (1s) in the result of an operation is even; cleared if odd. Meaning if the number of 1’s in the answer is even (0, 2, 4, ...), the flag will be set to 1, otherwise if odd then it will be set to 0.
   * **Usage**: Used for error checking in data transmission.
6. **Auxiliary Carry Flag (AF)**:
   * **Description**: Set if there is a carry from bit 3 to bit 4 or a borrow in binary-coded decimal (BCD) operations.
   * If the two numbers are in Binary, if a group of 3 or 4 bit number produces a carry, the flag will be set to 1.
   * If the two numbers are in hexadecimal, and if the right most bit generates a carry, the flag is set to 1, because each hexadecimal bit represents a nibble in binary.
   * **Usage**: Primarily used in BCD arithmetic.
7. **Interrupt Flag (IF)**:
   * **Description**: Controls the processor’s response to interrupt requests. If set, interrupts are enabled; if cleared, interrupts are disabled.
   * **Usage**: Used to enable or disable interrupt handling.
8. **Direction Flag (DF)**:
   * **Description**: Determines the direction of string operations. If set, operations proceed from high to low memory; if cleared, from low to high.
   * **Usage**: Used in string manipulation instructions.
9. **Trap Flag (TF)**:
   * **Description**: When set, it enables single-step mode for debugging. The processor will generate an interrupt after each instruction.
   * **Usage**: Useful for debugging and tracing execution.

**Determine Overflow & Carry Flag**

A screenshot of a computer

Description automatically generated

**Question:** Give the value of carry flag, zero flag, sign flag and overflow flag if AX = 0x1254 and BX = 0x0FFF

1. add ax, 0xEDAB
2. add ax, bx
3. add bx, 0xF001

**First line of code:**

|  |
| --- |
| 1254 |
| +EDAB |
| FFFF |

FFFF = Binary equivalent = 1111 1111 1111 1111

This addition is of unsigned numbers (positive number), but in the answer we can see the sign bit is ON. We can count the number of 1’s in our answer, its even, so parity flag is ON.

There is no carry and no overflow. And obviously the answer is not zero so the zero flag is not set.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Carry Flag | Zero Flag | Sign Flag | Overflow Flag | Parity Flag |
| 0 | 0 | 1 | 0 | 1 |

**Second line of code:**

|  |
| --- |
| FFFF |
| +0FFF |
| 1 0FFE |

--We know there is a carry generated in the left most byte, or we can nibble if we convert it into binary. So, auxiliary flag is set to 1.

--There is a carry, but to determine if its a carry or an overflow, we need to check if our answer is within the unsigned numbers range (0 to 2n - 1) n = number of bits.

Our answer is of 4-bytes, not including the carry. So n = 16-bit

216-1 = 65536 – 1 = 65535

Now, our answer should be within this range. 0FFE, convert it to decimal.

(0FFE)₁₆ = (0 × 16³) + (15 × 16²) + (15 × 16¹) + (14 × 16⁰) = (4094)₁₀

Our answer is within the unsigned numbers range, so the extra bit isn’t the overflow bit, rather it is simply a carry bit.

--For sign bit, we don’t consider the extra bit:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hexadecimal | 0 | F | F | E |
| Binary | 0000 | 1111 | 1111 | 1110 |

We can see the MSB/sign-bit is 0.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Carry Flag | Zero Flag | Sign Flag | Overflow Flag | Auxiliary Flag |
| 1 | 0 | 0 | 0 | 1 |

**Third line of code:**

|  |
| --- |
| 0FFF |
| +F001 |
| 1 0000 |

--We know there is a carry generated in the left most byte, or we can nibble if we convert it into binary. So, auxiliary flag is set to 1.

--For sign bit, we don’t consider the extra bit:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hexadecimal | 0 | 0 | 0 | 0 |
| Binary | 0000 | 0000 | 0000 | 0000 |

We can see the MSB/sign-bit is 0.

--There is a carry, but to determine if its a carry or an overflow, we need to check if our answer is within the unsigned numbers range (0 to 2n - 1) n = number of bits.

Our answer is of 4-bytes, not including the carry. So n = 16-bit

216-1 = 65536 – 1 = 65535

Now, our answer should be within this range. 0000, convert it to decimal.

(0000)₁₆ = (0 × 16³) + (0 × 16²) + (0 × 16¹) + (0 × 16⁰) = (0)₁₀

Our answer is within the unsigned numbers range, so the extra bit isn’t the overflow bit, rather it is simply a carry bit.

--We know our answer is 0000, so that means zero flag will be set to 1.

--The number of 1’s is 0, its even, so parity flag will be set to 1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Carry Flag | Zero Flag | Sign Flag | Overflow Flag | Auxiliary Flag | Parity Flag |
| 1 | 1 | 0 | 0 | 1 | 1 |