3.2 Conditional Jumps

1. Introduction to Conditional Jumps:

Conditional jumps are essential in assembly language programming because they allow the program to execute different paths based on specific conditions. For example, instructions like **JZ** (Jump if Zero) and **JNZ** (Jump if Not Zero) check the Zero Flag (ZF). If two operands are equal, subtracting them will result in zero, which sets the ZF. Therefore, JZ can be used to test equality. Similarly, the renamed versions **JE** (Jump if Equal) and **JNE** (Jump if Not Equal) have clearer meanings but work the same way with the same opcode.

2. Use of Flags:

Many conditional jumps have multiple names for clarity. Intel has established these names for the iAPX88 architecture. Other flags, such as the Carry Flag (CF), are used with instructions like **JC** (Jump if Carry) and **JNC** (Jump if Not Carry). For example, the CF indicates if there was a carry during an unsigned addition or if two unsigned numbers were subtracted.

Jumps like **JB** (Jump if Below) and **JAE** (Jump if Above or Equal) help with unsigned comparisons, while **JG** (Jump if Greater) and **JL** (Jump if Less) help with signed comparisons.

3. Table of Jumps:

Here's a summary of how different jumps are conditioned based on flags:

- **JC** (Jump if Carry): **CF = 1** (Jump if there was a carry)

- **JNC** (Jump if Not Carry): **CF = 0** (No carry means the destination is larger)

- **JE** (Jump if Equal): **ZF = 1** (Jump if the last operation resulted in zero)

- **JNE** (Jump if Not Equal): **ZF = 0** (Jump if the last operation did not result in zero)

- **JA** (Jump if Above): **ZF = 0 AND CF = 0** (Jump if the unsigned source is greater)

- **JNA** (Jump if Not Above): **ZF = 1 OR CF = 1** (Jump if the unsigned source is less or equal)

- **JL** (Jump if Less): **SF ≠ OF** (Jump if the signed source is smaller)

- **JG** (Jump if Greater): **ZF = 0 AND SF = OF** (Jump if the signed source is greater)

4. Signed vs. Unsigned Comparisons:

The key distinction is how the CPU interprets the values during comparison. For example, the same bit pattern can represent different values depending on whether it's considered signed or unsigned. In signed comparisons, the most significant bit (MSB) represents the sign, whereas in unsigned comparisons, it simply holds data. This distinction is critical for accurate program behavior, especially when working with different data types.

5. Special Jump Instruction:

The **JCXZ** instruction (Jump if CX is Zero) is unique because it does not depend on any flags. It checks the CX register specifically, as it's treated as a counter. This jump is regardless of the zero flag and has no counterpart.

6. Example of Using Jumps:

An example is provided where a program adds ten numbers without using a separate counter, employing comparisons to determine when to stop. The CX register serves as both an index and a counter, streamlining the program structure.

7. Conclusion:

Conditional jumps are fundamental for control flow in assembly programming, enabling programmers to implement complex logic. Understanding how to utilize flags and conditional jumps effectively is essential for writing efficient assembly code. With this knowledge, programmers can construct robust applications that respond dynamically to varying conditions.