Assignment 2: Understanding Decimal Adjustment (DAW) in PIC Microcontrollers

Objective:

To delve into the fundamental principles of decimal adjustment in PIC microcontrollers and explore its significance in practical applications.

Task:

1. **Why Decimal Adjustment?**
    - Explain the inherent limitations of microcontrollers in handling decimal values directly.
    - Discuss the necessity of converting binary representations into human-readable decimal formats.
    - Highlight the role of decimal adjustment in enabling effective interaction with users and external devices.
2. **Practical Example:**
    - Develop a code example demonstrating decimal adjustment using a specific PIC microcontroller model.
    - Choose a relevant application, such as displaying numeric values on an LCD, interacting with keypad inputs, or performing calculations involving decimal numbers.
3. **Code Explanation:**
    - Provide a detailed explanation of your code, focusing on the steps involved in decimal adjustment.
    - Discuss the reasoning behind each operation and how it contributes to the overall conversion process.
4. **Applications:**
    - Disucss real-world scenarios where decimal adjustment is indispensable in PIC microcontroller-based systems.
    - Discuss the impact of decimal adjustment on user experience, device functionality, and overall system performance.

Q1

INHERENT LIMITATIONS OF MICROCONTROLLERS IN HANDLING DECIMAL VALUES

Microcontrollers, which are small computing devices used in embedded systems, often have inherent limitations when it comes to handling decimal values directly. Here are some key limitations:-

- Many microcontrollers are designed for efficiency and low power consumption, which often means they have limited processing power. Decimal operations are generally more computationally intensive than integer operations, leading to slower performance when handling decimal values.
- Microcontrollers typically have limited RAM and flash memory. Decimal numbers usually require more memory compared to integers. This can lead to inefficient use of memory resources, especially in applications where memory is at a premium.
- Decimal representation can introduce precision errors due to the way decimal values are stored in binary format. This can lead to inaccuracies in calculations, which may be critical in applications requiring high precision.
- Not all microcontrollers are equipped with a Floating-Point Unit (FPU). An FPU is a specialized hardware component that can perform floating-point arithmetic more efficiently. Without an FPU, microcontrollers must rely on software libraries to perform floating-point calculations, which can be significantly slower and less efficient.

THE NECESSITY & ROLE OF DECIMAL ADJUSTMENT

Converting binary representations into human-readable decimal formats is essential for several reasons, particularly in enhancing user interaction and ensuring effective communication with external devices. Here are a few reasons:-

- Most people are accustomed to working with decimal numbers in everyday life. Converting binary data into decimal formats makes it easier for users to understand and interpret the information, especially in applications like user interfaces, reports, and data visualization.
- In many applications, such as financial software, scientific calculations, or reporting tools, presenting data in decimal format is necessary for clarity and effective communication. Decimal formats are more intuitive for conveying results, making it easier for stakeholders to make informed decisions.
- Working directly with binary representations can lead to misunderstandings or errors, especially when interpreting values. Converting to decimal reduces the cognitive load on users, minimizing the risk of mistakes in calculations or data entry.
- Many industries have standardized on decimal formats for reporting and documentation. Converting binary data to decimal ensures compliance with these standards, which can be crucial for regulatory or quality assurance purposes.

- In applications where users need to input values (e.g., calculators, control systems), decimal formats are more user-friendly. Users can enter values in a familiar format, which can then be converted to binary for processing.
- Training users on systems that utilize decimal formats is generally easier than training them on binary systems. This reduces the time and resources needed for onboarding and support, leading to more efficient operations.

Q2

PRACTICAL EXAMPLE

```
MOVLW 0x57

ADDLW 0x77

DAW
```

Q3

CODE EXPLANATION

- The first instruction moves the literal (57H) into the working register (WREG = 57H)
- The second instruction adds the literal (77H) to the working register (WREG = 57H + 77H = 0xCE)
- The third instruction performs the DAW operation. When lowers nibbles are added (i.e. 7+7), the output is 0xE (i.e. 14) and the digital/auxiliary carry is set. When the lower nibble is greater than BCD 9(or when DC=1), the DAW operator adds BCD 6 to the lower nibble. Similarly, when upper nibbles are added (i.e. 5+7), the output is 0xC (i.e. 12). When the upper nibble is greater than BCD 9(or when C=1), the DAW operator adds BCD 6 to the upper nibble. Hence, now the output becomes 134 which is in accordance with BCD.

Q4

APPLICATIONS OF DECIMAL ADJUSTMENT

Decimal adjustment is a critical aspect of many real-world applications involving PIC microcontrollers, particularly in scenarios where precise numerical representation and manipulation are essential. Here are several scenarios where decimal adjustment is indispensable, along with a discussion of its impact on user experience, device functionality, and overall system performance:-

- PIC microcontrollers are often used in real-time clocks (RTC) and digital timers where the time needs to be displayed in a human-readable decimal format (e.g., 12:30:45). Decimal adjustment is essential in converting the binary-encoded time values into decimal numbers.
- In applications where a keypad is used for input (such as in security systems or calculators), the input from the keypad is usually processed in binary. However, the system needs to adjust the decimal values for meaningful display or further processing.
- Many temperature sensors interface with PIC microcontrollers and output their values in binary format. To display the temperature to the user in Celsius or Fahrenheit (e.g., 23.4°C), decimal adjustment is performed to ensure proper formatting.
- Decimal adjustment ensures that numerical data is presented in a clear and understandable format, which is crucial for user interaction. Users can quickly interpret values, leading to a more intuitive experience.
- Accurate decimal representation fosters trust in the device. Users are more likely to rely on devices that consistently provide precise and accurate information.
- Many applications require precise calculations (e.g., in control systems, financial transactions). Decimal adjustment allows for accurate arithmetic operations, which are essential for the correct functioning of the device.
- Proper handling of decimal values reduces the likelihood of errors in calculations, which can be critical in applications like medical devices or safety systems.
- Efficient decimal adjustment algorithms can enhance the overall performance of the system by minimizing processing time and resource usage. This is particularly important in resource-constrained environments like embedded systems.
- Systems that handle decimal adjustments effectively can be more easily scaled to accommodate additional features or increased complexity without compromising performance.