MEGHANA P
BU22EECE0100339

**HANDS ON ACTIVITY:**
**EMBEDDED SYSYTEM FLOWCHART OF 7 PROGRAMS**

---

01.Write a program to count no. of bits which are set in given binary pattern2

**Code:**

```
def count_set_bits(binary_pattern):
    count = 0
    for bit in binary_pattern:
        if bit == '1':
            count += 1
    return count

# Test the function
binary_pattern = input("Enter a binary pattern: ")
count = count_set_bits(binary_pattern)
print("Number of set bits:", count)
```

**Output:**

```
Enter a binary pattern: 101010
Number of set bits: 3
```

02.Write a program to set 5th and 12th bits in a 16-bit unsigned integer

**Code:**

```
def set_bits(n, *positions):
    for pos in positions:
        n |= (1 << pos)
    return n

# Example usage
unsigned_integer = 0b0000000000000000  # Initialize a 16-bit unsigned integer
unsigned_integer = set_bits(unsigned_integer, 5, 12)  # Set the 5th and 12th bits
print("Resulting unsigned integer:", bin(unsigned_integer))
```

**Output:**

```
Resulting unsigned integer: 0b1000010000000000
```

01. Write a program to clear 6th and 19th bits in a 32-bit unsigned integer

**Code:**

```
def clear_bits(num, *positions):
    for pos in positions:
        mask = ~(1 << pos)
        num &= mask
    return num
# Example usage
num = 0b10101010101010101010101010101010  # Example 32-bit unsigned integer
cleared_num = clear_bits(num, 6, 19)
# Output
print("Original Number:", bin(num))
print("Cleared Number: ", bin(cleared_num))
```

**Output:**

Original Number: 0b10101010101010101010101010101010

Cleared Number:  0b10001010101010101010101010101010

02. Write a program to flip even positioned bits in a 16-bit unsigned integer
An IP Address will be in the form of "a. b, c. d" format, where a, b, c, d will be in the range of 0-255. Given a, b, c, d values (or string format) pack them into 32-bit unsigned integer.

**Code:**

```
def flip_even_bits(num):
    # Convert the number to binary representation
    binary_num = bin(num)[2:].fill(16)
    # Flip even-positioned bits
    flipped_binary = ''.join(['1' if i % 2 == 0 else bit for i, bit in enumerate(binary_num)])
    # Convert back to integer
    flipped_num = int(flipped_binary, 2)
    return flipped_num
def pack_ip_to_int(a, b, c, d):
    ip_int = (a << 24) | (b << 16) | (c << 8) | d
    return ip_int
# Example usage
a, b, c, d = 192, 168, 1, 10  # Example IP address values
ip_int = pack_ip_to_int(a, b, c, d)
flipped_ip_int = flip_even_bits(ip_int)
# Output
print("Original IP Address (in integer):", ip_int)
print("Flipped IP Address (in integer): ", flipped_ip_int)
```

**Output:**

(3232235786, 1077939210)

05.Given an unsigned 32-bit integer holding packed IPv4 address, convert it into "a. b. c. d" format.

**Code:**

```
def unpack_ip_from_int(ip_int):
    a = (ip_int >> 24) & 255
    b = (ip_int >> 16) & 255
    c = (ip_int >> 8) & 255
    d = ip_int & 255
    return a, b, c, d

# Example usage
ip_int = 3232235778  # Example packed IPv4 address
a, b, c, d = unpack_ip_from_int(ip_int)
ip_address = f"{a}. {b}. {c}. {d}"

# Output
print("Packed IPv4 Address (in integer):", ip_int)
print("Unpacked IPv4 Address (in 'a. b. c. d' format):", ip_address)
```

**Output:**

```
Packed IPv4 Address (in integer): 3232235778
Unpacked IPv4 Address (in 'a. b. c. d' format): 192.168.1.2
```

06.Convert MAC address into 48-bit binary pattern

**Code:**

```
def mac_to_binary(mac):
    # Remove colons from MAC address
    mac = mac.replace(":", "")
    # Convert each hex digit to binary and concatenate
    binary_mac = ''.join(format(int(char, 16), '04b') for char in mac)
    return binary_mac
# Example MAC address
mac_address = "A1:B2:C3:D4:E5:F6"
binary_mac = mac_to_binary(mac_address)
# Output
print("MAC Address:", mac_address)
print("48-bit Binary Pattern:", binary_mac)
```

**Output:**

MAC Address: A1: B2: C3: D4: E5: F6

07. Convert 48-bit binary pattern as MAC address

**Code:**

```
def binary_to_mac(binary):
    # Ensure the binary string is 48 bits long
    if len(binary) != 48:
        raise ValueError("Binary pattern must be 48 bits long")
    # Split the binary string into 6 segments of 8 bits each
    segments = [binary[i:i+8] for i in range(0, 48, 8)]
    # Convert each segment from binary to hexadecimal and format as two hex digits
    hex_segments = [format(int(segment, 2), '02X') for segment in segments]
    # Join the hex segments with colons
    mac_address = ':'.join(hex_segments)
    return mac_address
# Example binary pattern
binary_pattern = "101000011011001011000011110101001110010111110110"
mac_address = binary_to_mac(binary_pattern)
# Output
print("48-bit Binary Pattern:", binary_pattern)
print("MAC Address:", mac_address)
```

**Output:**

```
48-bit Binary Pattern: 101000011011001011000011110101001110010111110110
MAC Address: A1:B2:C3:D4:E5:F6
```