**Bitwise Operators Assignment**

Q1. WAP to read a 8 bit unsigned integer, interchange the adjacent bits i.e D0 with D1, D2 with D3….. D6 with D7. Display the final number.

Input: 0xAA

Output: 0x55

Output:

A.#include <stdio.h>

int main()

{

unsigned char num;

printf("Enter an 8-bit unsigned integer in hex (e.g., 0xAA): ");

scanf("%hhx", &num);

unsigned char even\_bits = num & 0x55; // Extract D0, D2, D4, D6

unsigned char odd\_bits = num & 0xAA; // Extract D1, D3, D5, D7

// Shift the even bits left and odd bits right

even\_bits = even\_bits << 1; // Shift even bits left by 1

odd\_bits = odd\_bits >> 1; // Shift odd bits right by 1

// Combine the shifted bits

unsigned char result = even\_bits | odd\_bits;

printf("Output after swapping adjacent bits: 0x%02X\n", result);

return 0;

}

A black background with white text

Description automatically generated

Q2. WAP to count the number of 1’s in a given byte and display

Output:

A. #include<stdio.h>

int count\_ones (unsigned char num)

{

int count = 0;

while (num) {

// Remove the rightmost 1 bit

num = num & (num - 1);

count++;

}

return count;

}

int main()

{

unsigned char num;

printf("Enter an 8-bit unsigned integer in hex (e.g., 0xAA): ");

scanf("%hhx", &num);

// Call the function to count 1's in the byte

int ones\_count = count\_ones(num);

printf("Number of 1's in 0x%02X is: %d\n", num, ones\_count);

return 0;

 }

A black background with white text

Description automatically generated

Q3. Generate odd and even parity bits for a given number. (consider a 32 bit number)

[Hint: You may reuse the solution created in Q2 and extend it further]

Output:

A.#include <stdio.h>

int count\_ones (unsigned int num)

{

int count = 0;

while (num) {

// Remove the rightmost 1 bit

num = num & (num - 1);

count++;

}

return count;

}

void generate\_parity(unsigned int num)

{

int ones\_count = count\_ones(num);

int even\_parity = (ones\_count% 2 == 0)?0:1;

int odd\_parity = (ones\_count% 2 ==0)?1:0;

printf("For the 32-bit number 0x%08X:\n", num);

printf("Even Parity Bit: %d\n", even\_parity);

printf("Odd Parity Bit: %d\n", odd\_parity);

}

int main()

{

unsigned int num;

printf("Enter a 32-bit unsigned integer in hex (e.g., 0xAA): ");

scanf("%x", &num);

generate\_parity(num);

return 0;

}

A screen shot of a computer

Description automatically generated

Q4. WAP to reverse the bytes in a 32 but unsigned integer using shift operator.

A.#include <stdio.h>

unsigned int reverse\_bytes(unsigned int num)

{

unsigned int byte1, byte2, byte3, byte4;

// Extract each byte using shift and mask

byte1 = (num & 0xFF) << 24; // Get the LSB and shift it to the MSB position

byte2 = (num & 0xFF00) << 8; // Get the second byte and shift it

byte3 = (num & 0xFF0000) >> 8; // Get the third byte and shift it

byte4 = (num & 0xFF000000) >> 24; // Get the MSB and shift it

return byte1 | byte2 | byte3 | byte4;

}

int main()

{

unsigned int num;

printf("Enter a 32-bit unsigned integer in hex (e.g., 0x12345678): ");

scanf("%x", &num);

unsigned int reversed\_num = reverse\_bytes(num);

printf("Reversed 32-bit number: 0x%08X\n", reversed\_num);

return 0;

}

A black background with white text

Description automatically generated