GOWHASRI DARLA BU21EECE0100486

HANDS ON ACTIVITY

EMBEDDEDSYSYTEMS

1] Write a program to count no. of bits which are set in given binary pattern .

Code::

```
#include <stdio.h>
int countSetBits(unsigned int num) { int count = 0;
while (num) {
count += num & 1; num >>= 1;
} return count; } int main() {
unsigned int num = 0b10101010;
printf("Number of set bits: %d\n", countSetBits(num)); return 0; }
```

Output: Number of set bits: 4

2] **Write a program to set 5th and 12th bits in a 16-bit unsigned integer Code:**

```
#include <stdio.h>
  unsigned int setBits(unsigned int num, int pos1, int pos2) { unsigned
    int mask = (1 << pos1) | (1 << pos2);
  return num | mask;
 }
int main() {
  unsigned int num = 0b00000000;num = setBits(num, 5,12);
  printf("Modified number: %d\n", num);
  return 0;
  }
```

Output: Modified number: 4864

3] Write a program to clear 6th and 19th bits in a 32-bit unsigned integer. Code:

```
#include <stdio.h>
unsigned int clearBits(unsigned int num, int pos1, int pos2) { unsigned
int mask = ~(1 << pos1) & ~(1 << pos2);
```

```
return num & mask;
}
int main() { unsigned int num = 0b11111111111111111; num =
clearBits(num, 6, 19);
printf("Modified number: %u\n", num); return 0;
}
```

Output: Modified number: 524287

4] Write a program to flip even positioned bits in a 16-bit unsigned integer

Code:

```
#include <stdio.h>
unsigned int flipEvenBits(unsigned int num) {
  unsigned int mask = 0xAAAA; // Binary pattern with even bits set return num ^
  mask;
}
int main() {
  unsigned int num = 0b1010101010101010; // Example 16-bit unsigned integer
  num = flipEvenBits(num); printf("Modified number: %d\n",
  num); return 0;
```

Output: Modified number: 2730

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

Code:

```
#include <stdio.h>
  int countSetBits(unsigned int num) { int count = 0;
  while (num) {
    count += num & 1; num >>= 1;
  }
  return count;
}
int main() {
  unsigned int num = 0b10101010; // Example binary pattern
  printf("Number of set bits: %d\n", countSetBits(num)); return 0;
}
```

Output: Number of set bits: 4

6] Convert MAC address into 48-bit binary pattern Code:

```
#include <stdio.h>
  void unpackIPAddress(unsigned int ip) { int a, b, c, d;
  a = (ip >> 24) & 255;
  b = (ip >> 16) & 255;
  c = (ip >> 8) & 255;
  d = ip & 255;
  printf("Unpacked IP address: %d.%d.%d.%d\n", a, b, c, d);
}
int main() {
  unsigned int packedIP = 3232235777; // Example packed IP address
  unpackIPAddress(packedIP);
  return 0;
```

Output: Unpacked IP address: 192.168.1.1

7] Convert 48-bit binary pattern as MAC address Code:

```
#include <stdio.h>

  void macToBinaryPattern(char *mac) { unsigned long
    long int binary = 0;
  sscanf(mac, "%2hhx:%2hhx:%2hhx:%2hhx:%2hhx:%2hhx", (unsigned char
*)&binary,
    (unsigned char *)&binary + 1, (unsigned char *)&binary + 2, (unsigned
    char *)&binary + 3, (unsigned char *)&binary + 4, (unsigned char
    *)&binary + 5);
  printf("Binary pattern: %llx\n", binary);
}

int main() {
  char mac[] = "12:34:56:78:9a:bc"; // Example MAC address
  macToBinaryPattern(mac);
  return 0;
}
```

Output: Binary pattern: 123456789abc

8] Convert 48-bit binary pattern to MAC address. Code:

```
#include <stdio.h>
void binaryPatternToMac(unsigned long long int binary)
{
    printf("MAC address: %02llx:%02llx:%02llx:%02llx:%02llx:%02llx\n", (binary
     >> 40) & 0xFF, (binary >> 32) & 0xFF, (binary >> 24) & 0xFF, (binary >> 16)
     & 0xFF, (binary >> 8) & 0xFF, binary & 0xFF);
}
int main() {
  unsigned long long int binary = 0x123456789abc; // Example binary pattern
  binaryPatternToMac(binary);
  return 0;
}
```

Output: MAC address: 12:34:56:78:9a:bc

Comparison Table between 8051 and Arduino

Feature	8051 Micro-controller	Arduino
Architecture	Harvard	Modified Harvard
Instruction Set	8-bit	8-bit (AVR) or 32-bit (ARM)
Clock Speed	Typically up to 12 MHz	8 MHz (Uno), 16 MHz (Mega), varies with different boards
Memory	ROM,RAM, EEPROM	Flash, SRAM, EEPROM
GPIO Pins	Limited	Abundant, typically 20 or more
Analog Inputs	Usually limited	Typically multiple, 6 or more
Digital I/O	Limited	Abundant
Development Tools	Limited availability	Extensive community support, IDE like Arduino IDE

Programming	**Assembly, C**	**Arduino Sketch (C/C++)**
IDE Support	**Limited**	**Arduino IDE, PlatformIO**
Debugging	**Limited**	**Limited (Serial debugging, LED blinking)**
Cost	**Affordable**	**Affordable**

Notes:
•Architecture: The 8051 microcontroller uses a separate memory space for instructions and data, typical of Harvard architecture, while Arduino uses a modified Harvard architecture.
•Instruction Set: The 8051 microcontroller has a more complex instruction set compared to the simpler, more efficient RISC architecture of Arduino.
•Ease of Use: Arduino is designed to be beginner-friendly with extensive documentation and a simple IDE.
•Memory: Arduino typically has more RAM and program memory than the 8051.
•I/O and ADC: Arduino generally provides more flexibility with built-in analog-to-digital converters (ADCs) and more I/O pins.
•Community Support: Arduino benefits from a large, active community that provides a wealth of libraries, shields, and support resources.