**Bitwise and Parallel Port Programs**

This README documents a collection of C programs that demonstrate various bitwise operations and hardware interaction using the parallel port (0x378 and 0x370). These programs include binary manipulation, hardware LED control, and simple animations.

**1. Alternate Bit Pattern on Parallel Port (0x378)**

**Objective**: Generate alternating 0xAA and 0x55 patterns to blink LEDs in a striped manner.

outb(0xAA, 0x378);

usleep(500000);

outb(0x55, 0x378);

**2. Byte Representation in Decimal, Hex, and ASCII**

**Objective**: Print the decimal, hexadecimal, and ASCII representation of a byte.

unsigned char byte = 65;

printf("Decimal: %d\nHex: %X\nASCII: %c\n", byte, byte, byte);

**3. Walking One Pattern on Parallel Port (0x370)**

for (int i = 0; i < 8; i++) {

outb(1 << i, 0x370);

usleep(500000);

}

**4. Walking Zero Pattern on Parallel Port (0x378)**

for (int i = 0; i < 8; i++) {

outb(~(1 << i), 0x378);

usleep(500000);

}

**5. User-Controlled LED (ON, OFF, TOGGLE)**

unsigned char led = 0x00;

int bit, choice;

printf("Enter bit (0-7): ");

scanf("%d", &bit);

printf("1.ON 2.OFF 3.TOGGLE: ");

scanf("%d", &choice);

switch (choice) {

case 1: led |= (1 << bit); break;

case 2: led &= ~(1 << bit); break;

case 3: led ^= (1 << bit); break;

}

outb(led, 0x378);

**6. Bit Sequence Display**

void display\_bits(unsigned char byte) {

for (int i = 7; i >= 0; i--)

printf("%d", (byte >> i) & 1);

printf("\n");

}

**7. Set a Particular Bit**

byte |= (1 << n);

**8. Clear a Particular Bit**

byte &= ~(1 << n);

**9. Toggle a Particular Bit**

byte ^= (1 << n);

**10. Return the N-th Bit**

int bit = (byte >> n) & 1;

**11. Count Number of 1s in a Byte**

int count = 0;

for (int i = 0; i < 8; i++)

if ((byte >> i) & 1) count++;

**12. Count Number of 0s in a Byte**

int zeros = 8 - count;

**13. Shift All 1s to Left**

unsigned char left\_shift\_ones(unsigned char b) {

int count = 0;

for (int i = 0; i < 8; i++) if (b & (1 << i)) count++;

return ((1 << count) - 1) << (8 - count);

}

**14. Reverse Bits in a Byte**

unsigned char reverse\_bits(unsigned char byte) {

unsigned char rev = 0;

for (int i = 0; i < 8; i++) {

rev = (rev << 1) | (byte & 1);

byte >>= 1;

}

return rev;

}

**15. Find Size of Unsigned Int Without sizeof()**

unsigned int x = ~0;

int size = 0;

while (x) { size++; x >>= 8; }

printf("%d bytes\n", size);

**16. Get Mask from Bit i to j**

unsigned int get\_mask(int i, int j) {

return (~0U >> (31 - j + i)) << i;

}

**17. Insert Bits m into n Between Bits j and i**

unsigned insert(unsigned n, unsigned m, int i, int j) {

unsigned mask = get\_mask(i, j);

return (n & ~mask) | (m << i);

}

**18. Nearest Numbers With Same Number of 1 Bits**

int count\_ones(int n) {

int c = 0; while (n) { c += n & 1; n >>= 1; } return c;

}

void nearest(int n) {

int c = count\_ones(n), g = n + 1, s = n - 1;

while (count\_ones(g) != c) g++;

while (count\_ones(s) != c) s--;

printf("Smaller: %d, Greater: %d\n", s, g);

}

**19. Count 1s and 0s at Odd and Even Positions**

void count\_odd\_even(unsigned char n) {

int eo = 0, oo = 0;

for (int i = 0; i < 8; i++)

(n & (1 << i)) ? ((i % 2) ? oo++ : eo++) : 0;

printf("Even 1s: %d, Odd 1s: %d\n", eo, oo);

}

**20. Swap Bits at Even and Odd Positions**

unsigned int swap\_even\_odd(unsigned int x) {

return ((x & 0xAAAAAAAA) >> 1) | ((x & 0x55555555) << 1);

}

**21. Find Missing Value in Consecutive Array**

int missing(int arr[], int n) {

int sum = 0;

for (int i = 0; i < n; i++) sum += arr[i];

return n \* (n + 1) / 2 - sum;

}

**22. Animate 20 'Hut' Shapes in Circular Path**

#include <stdio.h>

#include <math.h>

#include <unistd.h>

#define PI 3.14159

#define R 10

int main() {

for (int t = 0; t < 360; t += 10) {

system("clear");

for (int i = 0; i < 20; i++) {

double a = (t + i \* (360 / 20)) \* PI / 180;

int x = (int)(R \* cos(a)) + 20;

int y = (int)(R \* sin(a)) + 10;

for (int j = 0; j < y; j++) printf("\n");

for (int k = 0; k < x; k++) printf(" ");

printf("H");

}

usleep(100000);

}

return 0;

}