EE 472 – Embedded Systems

Dr. Shwetak Patel
Assistant Professor
Computer Science & Engineering
Electrical Engineering
Note Pack Solutions
// Constants
#define EXAMPLE_CONST_D 1234  /* DECIMAL value */
#define EXAMPLE_CONST_H 0x4D2  /* HEX value == dec: 1234 */
#define EXAMPLE_CONST_O 02322  /* OCTAL value */

int x = EXAMPLE_CONST_D;
int y = EXAMPLE_CONST_H;
int z = EXAMPLE_CONST_O;

// test yourself here: (which of these print?)
if (x == y) printf ("Hello there ... \n");  **YES**
else {};

if (y == 1234) printf ("What's up doc?? \n");  **YES**
else {};

if (z == 2322) printf ("The Rain in Spain ...\n");  **NO**
else {};

if (z == 2322) printf ("Falls mainly on the plains.\n");  **YES**
else {};
/* Use of bitwise logical operators */

#define Bit_Zero  0x01
#define Bit_One   0x02
#define Bit_Two   0x04
#define Bit_Three 0x08
#define Bit_Four  0x10
#define Bit_Five  0x20
#define Bit_Six   0x40

// etc etc etc ..

int x = 0x0B;       // x = [... 0 0 0 0 1 0 1 1]
int y = 011;        // y = [... 0 0 0 0 1 0 0 1]
int z = x << 2;     // z = [... 0 0 1 0 1 1 0 0]
int z1 = y & Bit_Three;  // z1 = [... 0 0 0 0 1 0 0 0]
int z2 = z | Bit_Four; // z2 = [... 0 0 1 1 1 1 0 0]
Convert 0x1A to decimal: 26

Convert 0xA2 to decimal: 162

Convert 020 to Hex: A0

Convert 0x20 to Octal: 40

/* Give the binary value of the least significant 8 bits of the following values: */

int aa = 0xC5; // aa = [1 1 0 0 0 1 0 1]
int ab = 0x7 + 1; // ab = [0 0 0 1 0 0 0 0]
int a = x ^ y; // a = [0 0 0 0 1 0 1 1]
int b = Bit_Two | Bit_Five | Bit_Seven;
   // b = [1 0 1 0 0 1 0 0]
int c = ~(z | Bit_Zero) << 1);
   // c = [1 0 1 0 0 1 0 1]
int d = !(z1 | Bit_Two);
   // d = [0 0 0 0 0 0 0 0]
Write a C function using bitwise logical operators etc. to print the binary value of a 16 bit word.

```c
void bin_display(unsigned int x)
{
    unsigned int i, j;
    for(i=0; i<16; i++)
    {
        j = 1<<(15-i);
        if(j & x) printf("1");
        else printf("0");
    }
}
```
Write a C code fragment which negates a signed (2's complement) integer using bitwise logical operators and addition. Include declarations for your variables. (do not just multiply by -1)

```
[blake@alec code]$ cat twocomp.c
/* twocomp.c
 *
 * Demonstrate 2's complement
 *
 */
void bin_display(unsigned int x);
main(argc,argv)
int argc;
char *argv;
{
    int a = -23;
    int b;
    int i,j;
    printf("Decimal: %d \n \n \n Binary: ", a);
    b = a;

    bin_display((unsigned int)b);
    printf("\n \n Complement: ");
    b = ~a;  // complement all bits
    bin_display((unsigned int)b);
    printf("\n i + Complement: ");
    b = i + ~a;  // complement all bits and add 1
    bin_display((unsigned int)b);
    printf("\n Compare: %d %d\n", a, -b);  // should be the same
}
```
C Pointers advance in increments of the thing they point to.

```c
int myInt;
float myFloat;

int* intPtr = &myInt;
float* floatPtr = &myFloat;

int szint = sizeof(myInt);
int szflt = sizeof(myFloat);

// Assume intPtr has the value 0x3000
// and floatPtr == 0x4000
// and szint = 4, szflt = 8 (bytes)

printf("intPtr = \%d\n", (int)intPtr);
printf("intPtr+1 = \%d\n", (int)intPtr+1);
printf("fltPtr = \%d\n", (int)fltPtr);
printf("fltPtr++ = \%d\n", (int)fltPtr++);
```

**Answers**

```plaintext
intPtr     = 0x3000
intPtr+1   = 0x3004
fltPtr     = 0x4000
fltPtr++   = 0x4008
```
Lewis Approach

#define DISP_BUFFER 0xB8000
int row, col;
char disp_char;
char *p;
// Let's display "A" in row 4, col 20
disp_char = 'A';
row = 4;    // for example ...
col = 20;   // "   "

p = (char *) (DISP_BUFFER + 2 *(80*row + col))
*p = disp_char;

1.6 Exercise in Class

Complete this example: write code to set the byte which controls the color of the character A we just displayed.

Assume the color byte should be assigned the value COLOR_BYTE.

The color byte has 2 4-bit fields which set the background color (16 possible colors) and the character color (16 possible colors).

Answer

#define COLOR_BYTE 0xBB // (made up hex value)

int row, col;
char *p;
row = 4;
col = 20;
p = (char *) ((DISP_BUFFER + 2 *(80*row + col)) + 1 )
*p = COLOR_BYTE;
Modify the scheme in Lewis, sec 3.6, so that we can address display characters and display colors each in a separate row/column array.

Answer

typedef char CELL[2]    // a CELL is 2 chars

CELL *graphics_chars = (CELL *)DISP_BUFFER;
CELL *graphics_colors = (CELL *) (DISP_BUFFER+1);
    // NOT: graphics_colors = 1 + graphics_chars -- Why?

graphics_chars[80*row + col] = 'A';
graphics_colors[80*row + col] = COLOR_BYTE;