ECED 3204 Microprocessor

Assignment #4 Reference Solution

http://www.jasongu.org/3204/assignments.html

Assignment #4 contains the following problems:

E6.1 what is the value of ax/bx, assuming that ax=93, bx=19 ?

ax / bx = 93 / 19 = 4

E6.6 Write a program to find the median of an array of 8 bit unsigend integers. When the array has an even number of elements, th mdian is defined as th average of the middle two elements. Otherwise, it is defined as the middle element of the array. You need to sort the array in order to find the median.

Solution: We need to sort the array in order to find its median. We will invoke the bubble sort function to sort the given array.

```
#include <io.h>
void swap (unsigned char *px, unsigned char *py);
void bubble (unsigned char a[ ], unsigned char n);
#define   N      30
unsigned char xa[] =  {10,9,8,7,6,5,4,3,12,15,
                       20,29,28,21,27,25,24,23,99,43,
                       30,41,53,89,76,54,65,49,46,55};
unsigned char median;

void main (void)
{
    bubble(xa, N);
    if (N == ((N / 2) * 2)) { // is N an even value?
          median = (xa[N/2] + xa[N/2 - 1])/2;
    }
    else
          median = xa[N/2];
    while(1);
}
void  bubble (unsigned char a[ ], unsigned char n)    // n is the array count
{
    unsigned char i, j;
    char inorder;      // array in order flag
    for (i = 0; i < n - 1; i++){ // up to n - 1 iterations are performed
          inorder = 1;       // assume array is in order at the start of a new iteration
```

```
            for (j = 0; j < n - i - 1; j++)
                  if (a[j] > a[j+1]){ // are two adjacent elements not in order?
                        swap (&a[j], &a[j+1]);
                        inorder = 0;        // array not in order
                  }
            if (inorder) // array is in order, there is no need to continue.
                  break;
      }
}
void swap (unsigned char *px, unsigned char *py)
{
      unsigned char temp;
      temp = *px;
      *px = *py;
      *py = temp;
}
```

E6.11 Write a switfh sttement that will examine the value of an integer variable yy and assign one of the following values to the variable dd, depending on the value of yy :

- **a) 15, if yy==1**
- **b) 25, if yy==2**
- **c) 35, if yy==3**
- **d) 45, if yy==4**
- **e) 55, if yy==5**

Solution:

```
switch (yy) {
      case 1:
            dd = 15;
            break;
      case 2:
            dd = 25;
            break;
      case 3:
            dd = 35;
            break;
      case 4:
            dd = 45;
            break;
      case 5:
            dd = 55;
            break;
      default:
            dd = 10;
}
```

E6.16 Write a function to determine whether a three-digit number is an Armstrong number and a main program to find all the 3-digit (> 100) Armstrong numbers and store them in an array. (An armsrong number is a number that is equal to the sum of each digit raised to the nth power. For exmple, 153 equals $1^3+5^3+3^3$)

Solution:

```
#include <io.h>
unsigned int armStrong[6];
char ArmStrong3Test(unsigned int ax);
void main (void)
{
    unsigned int ax;
    unsigned char ix;
    ix = 0;
    for (ax = 100; ax < 1000; ax++){
        if(ArmStrong3Test(ax)){
            armStrong[ix] = ax;
            ix++;
        }
    }
    while(1);
}
char ArmStrong3Test(unsigned int ax)
{
    unsigned int x1, x2, x3, tmp;
    x3    = ax % 10;// one's digit
    tmp   = ax / 10;   // hundred's and ten's digits
    x2    = tmp % 10;       // ten's digit
    x1    = tmp / 10;// hundred's digit
    if ((x1 * x1 * x1 + x2 * x2 * x2 + x3 * x3 * x3) == ax)
        return 1;
    else
        return 0;
}
```

E6.21 Write a function that can convert a binary number into a NULL-terminated BCD string that represents the given binary number.

Solution: The function that performs the conversion and its test program are as follows:

```
#include <io.h>
void i2s (int i,c har *s);
char obuf[10];
void main (void)
{
    int ax;
    ax    = -32000;
```

```
    i2s(ax, &obuf[0]);
    while(1);
}
void i2s(int i,char *s)       // Convert Integer to String
{
    char  sign;
    short         len;
    char  *p;
    sign  = '+';
    len   = 0;
    p     = s;   // save a copy of the pointer
    if(I < 0) {
          sign = '-';
          i = -i;
    }
    do { // separate each decimal digit with lsd first
          *s = (I % 10) + '0';
          s++;
          len++;
          i /= 10;
    } while (i != 0);
    if(sign == '-') {
          *s = '-';
          s++;
          len++;
    }
    for(i = 0;i < len/2; i++){ // reverse the string
          p[len]       = p[i];
          p[i]         = p[len – 1 - i];
          p[len-1-i]   = p[len];
    }
    p[len] = 0;
}
```