Embedded Processors UESTC
Homework Assignment 1

Author: Changgang Zheng UoG ID: 2289258Z UESTC ID: 2016200302027

Question 1 (15 points): There is a microcomputer embedded in a vending machine. List four operations the software must perform.

The aim of the vending machine is to sell things to customer. So the following operations should be performed:

1. The software must display the information about how many goods is remained in the vending machine by using some LEDs, or even the screen.
2. The software must respond to the chosen of consumers and receive the money and check how much money the customer is given.
3. The software must also deliver changes to consumers if they give more money or they decide not to buy it.
4. The software should also have the capacity to control the machine and offer the consumers the goods they choose.

Question 2 (25 points): Each row of the following table is to contain an equal value expressed in binary, hexadecimal, and decimal. Complete the missing values. Assume each value is 8 bits and the decimal numbers are signed. The first row illustrates the process.

binary	hexadecimal	decimal
11111110	0xFE	-2
11111101	0xFD	-3
10101110	0xAE	-82
10110011	0xB3	-77
10100100	0xA4	-92

Question 3.a (15 points): Give an approximation of sqrt(2) using the decimal fixed-point (0.001) format.

$$m = -3$$

$$Decimal\ fixed - point\ \ \Delta = 10^{-3}$$

$$Integer\ \ I = 1414$$

$$\mathrm{sqrt}(2) = 1.414 = 1\times10^{0} + 4\times10^{-1} + 1\times10^{-2} + 4\times10^{-3} = 1414\times10^{-3}$$

Question 3.b (15 points): An signed 16-bit binary fixed-point number system has a resolution of 1/256. What is the corresponding value of the number if the integer part stored in memory is 384?

$$Resolution = \frac{1}{256}$$

$$Binary\ fixed - point\ \ \Delta = 2^{-8}$$

$$m = -8$$

$$Integer\ \ I = 384 = 110000000_2$$

$$Value = 110000000 \times 2^{-8}{}_2 = 1.1_2 = 1 \times 2^{-0} + 1 \times 2^{-1} = 1.5$$

Question 4 (30 points): n and mask are two 32-bit integer variables. Using logical operations you have learned, answer the following questions:

1) What values mask should be used in order to set only bit 12 of n into 0 and 1 respectively?

If we want to change only bit 12 of n into 0:

$$We\ set:\quad mask = 1\mathrm{UL} << 12$$

$$mask = 0x00001000 = 0000\ 0000\ 0000\ 0000\ 0001\ 0000\ 0000\ 0000_2$$

$$n = n\&(\sim mask)$$

In this way (set mask like above and use 'and' operation), we can change only bit 12 of n into 0;

If we want to change only bit 12 of n into 1:

$$We\ set:\quad mask = 1\mathrm{UL} << 12$$

$$mask = 0x00001000 = 0000\ 0000\ 0000\ 0000\ 0001\ 0000\ 0000\ 0000_2$$

$$n| = mask$$

In this way (set mask like above and use 'or' operation), we can change only bit 12 of n into 1;

2) Write the program to check the value of the bit 2 and bit 10 of variable n. If the value of these two bits are different, swap their values.

The program is as follow:

```
//
//  main.c
//  EP_homework1
//
//  Created by Changgang Zheng on 4/18/18.
//  UoG ID: 2289258
//  UESTC ID: 2016200302027
//  Copyright © 2018 Changgang Zheng. All rights reserved.
//


#include <stdlib.h>
#include <stdio.h>

void print(int n){                  // define a function 'print' to
display the binory form of 'n'
    int N[32];                      // define an array to represent
its binary form
    for(int i=0;i<=31;i++){
        N[i]=0;                     // initialize the array to
represent its binary form
    }
    for (int j=0;j<=31;j++){

        N[31-j]=n%2;                // transfer n in to a array to
represent its binary form
        n=n/2;                      // transfer n in to a array to
represent its binary form
    }
    for(int k=0;k<=31;k++){
        printf("%d", N[k]);         // print the binary form of 'n'
    }
    printf("\n");

}

int main(int argc, const char * argv[]) {
    int  n, end;                    // define the variable 'n'
    end=0;                          // Initialize the 'end'
    while(end!=1){                  // After the input 'n' have
different bit2 and bit10, we just let the program to stop.
        printf("Please input decimal number for n which you want to
swap the bit2 and bit10 if they are different: \nn=");     // print
the second number I want to add.
        scanf("%d", &n);            // input the number 'n'.
        printf("binary n=");
        print(n);                   // print the 'n' in binary form.
        if(((n>>2)&1)!=((n>>10)&1)){// check if the bit2 and bit10 of
'n' are different.
           n=(((n&0x00000400)>>8)|((n&0x00000004)<<8))|(n&0xfffffbfb);
            // get the new representation of 'n' after swap bit2 and
bit10.
            // (n&0x00000400)>>8: move the bit8's value to the bit 2.
            // (n&0x00000004)<<8: move the bit2's value to the bit 8.
            // n&0xfffffbfb: keep the original binary n(beside bit2
and bit8 is changed to 0).

            printf("After swap bit2 and bit10\nbinary n=");
            print(n);               // after swap bit2 and bit10,
print the 'n' in binary form.
            printf("In decimal n=%d\n",n);
            end=1;                  // we let the program to fun out
of the while loop and stop.
        }
        else{
            printf("The bit2 and bit10 is same for this 'n', please
try again:\n\n");
        }
    }
}
```

There are descriptions for the program which is written as the comment in above.

The core of the program are in these lines:

```
if(((n>>2)&1)!=((n>>10)&1)){
    n=(((n&0x00000400)>>8)|((n&0x00000004)<<8))|(n&0xfffffbfb);
}
```

- (n&0x00000400)>>8: move the bit8's value to the bit 2.
- (n&0x00000004)<<8: move the bit2's value to the bit 8.
- n&0xfffffbfb: keep the original binary n (beside bit2 and bit8 is changed to 0).
- The 'or' operation combine these together and can swap the bit2 and bit8 successfully.

Output Sample of the program is as follow:

```
Please input decimal number for n which you want to swap the bit2 and
bit10 if they are different:
n=3
binary n=00000000000000000000000000000011
The bit2 and bit10 is same for this 'n', please try again:

Please input decimal number for n which you want to swap the bit2 and
bit10 if they are different:
```

```
n=4
binary n=00000000000000000000000000000100
After swap bit2 and bit10
binary n=00000000000000000000010000000000
In decimal n=1024
Program ended with exit code: 0
```