Homework 1

Shiyu Wang

1.

a. 53_{10}
   53 / 8 = 6……5 5
   6 / 8 = 0 …….6 65
   0 done
   So $53_{10} = 65_8$

b. FA_{16}
   FA_{16} = 1111(F) 1010(A) = 1111 1010_2
   $_{1111}$ 0_2 = 011(3) 111(7) 010(2) = 372_8

2.

a. 19_{10}
   19 / 2 = 9…..1 1
   9 / 2 = 4…..1 11
   4 / 2 = 2…..0 011
   2 / 2 = 1…..0 0011
   1 / 2 = 0…..1 10011
   0 done
   So $19_{10} = 10011_2 = 00010011_2$ (2's comp)

b. -13_{10}
   13 / 2 = 6…..1 1
   6 / 2…..3 01
   3 / 2 = 1……1 101
   1 / 2 = 0……1 1101
   0 done
   So $13_{10} = 1101_2 = 00001101_2$
   $-13_{10} = 11110010_2$ (1's comp) = 11110011_2 (2's comp)

c. -23_{10}
   23 / 2 = 11……1 1
   11 / 2 = 5…...1 11
   5 / 2 = 2……1 111
   2 / 2 = 1……0 0111
   1 / 2 = 0……1 10111
   0 done
   So $23_{10} = 10111_2$

$-23_{10} = 11101000_2$ (1's comp) = 11101001_2 (2's comp)

d. ED_{16}
$ED_{16} = 1110(E)\ 1101(D) = 11101101_2$ (2's comp)
$11101101_2 - 1 = 11101100_2$ (1's comp)
11101100_2 (1's comp) = $- 00010011_2 = -19_{10}$

3.

a. 0xABCD OR 0x9876
$ABCD_{16} = 1010(A)\ 1011(B)\ 1100(C)\ 1101(D) = 1010\ 1011\ 1100\ 1101_2$
$9876_{16} = 1001(9)\ 1000(8)\ 0111(7)\ 0110(6) = 1001\ 1000\ 0111\ 0110_2$

$1010\ 1011\ 1100\ 1101_2$

OR

$1001\ 1000\ 0111\ 0110_2$

= $1011\ 1011\ 1111\ 1111_2$

= $BBFF_{16}$

= 0xBBFF

b. 0xFEED AND (NOT(0xBEEF))
$FEED_{16} = 1111(F)\ 1110(E)\ 1110(E)\ 1101(D) = 1111\ 1110\ 1110\ 1101_2$
$BEEF_{16} = 1011(B)\ 1110(E)\ 1110(E)\ 1111(F) = 1011\ 1110\ 1110\ 1111_2$

Not(0xBEEF) = $0100\ 0001\ 0001\ 0000_2$

$1111\ 1110\ 1110\ 1101_2$
AND
$0100\ 0001\ 0001\ 0000_2$

= $0100\ 0000\ 0000\ 0000$

= 4000_{16}

= 0x4000

4.

a. 01000010 01100101 01110011 01110100 00100000 01101111 01100110

01000010_2 01100101_2 01110011_2 01110100_2 00100000_2 01101111_2 0110011_2

= $66_{10}\ 101_{10}\ 115_{10}\ 116_{10}\ 111_{10}\ 51_{10}$

By ASCII table

= B e s t o 3

b. 01001100 01110101 01100011 01101011 00100001
$01001100_2\ 01110101_2\ 01100011_2\ 01101011_2\ 00100001_2$

= $76_{10}\ 117_{10}\ 99_{10}\ 107_{10}\ 33_{10}$

By ASCII table

= L u c k !

5.

a. 5 bits
14 / 2 = 7……0 0
7 / 2 = 3……1 10
3 / 2 = 1……1 110
1 / 2 = 0……1 1110
0 done
14_{10} = 01110
-14_{10} = 10010 (2's comp)

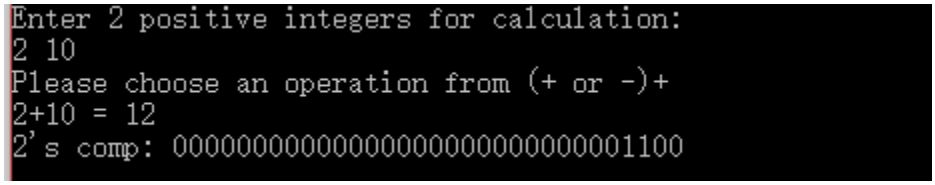
b. 6 bits
14_{10} = 001110
-14_{10} = 110010 (2's comp)

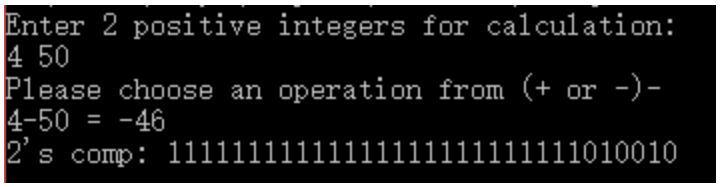
c. 7 bits
14_{10} = 0001110
-14_{10} = 1110010 (2's comp)

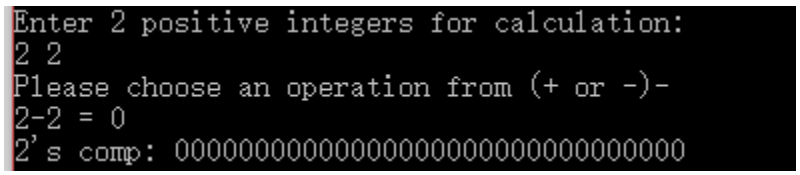
From the experiments, I found two's complement is easier to store ints, when you need to extend the width of the register the value is being stored in. With two's complement, storing a less bit number in a high bit register is a matter of repeating its most significant bit. On the other hand, if we just simply flip the first bit, we would need to clear the existing bit, which is an extra operation in addition to padding.

6. Code is attached

Test:








```
#include <stdio.h>
#include <string.h>

#define SIZE 8

//sum the two input positive integers by logic operations
int sum(int a, int b)
{
        // Iterate till there is no carry
   while (b != 0)
   {
       // carry now contains common set bits of x and y
       int carry = a & b;

       // Sum of bits of x and y where at least one of the bits is not set
       a= a ^ b;
```

```
        // Carry is shifted by one so that adding it to x gives the required sum
        b = carry << 1;
    }
    return a;
}

//compute the difference of two inut posiive integer by logic operations
int diff(int a, int b)
{
            //a - b = a + (-b)
             return (sum(a, sum(~b, 1)));
}

//int to binary
void decimal_to_binary(int n)
{
            //variables use to count
   int c, d, count;

   //store the binary
   char *pointer;

   count = 0;
   pointer = (char*)malloc(32+1);

   //transfer int to binary, bit to bit
   for ( c = 31 ; c >= 0 ; c-- )
```

```
  {
    d = n >> c;

    if ( d & 1 )
      *(pointer+count) = 1 + '0';
    else
      *(pointer+count) = 0 + '0';

    count++;
  }
  *(pointer+count) = '\0';

  printf(pointer);
  free(pointer);
}


int main(void) {
        /*scan the input of two positive integers*/
        printf("Enter 2 positive integers for calculation:\n");
        int a, b;
        scanf("%d%d", &a, &b);

        /*check the input integers are positive*/
        if (a <= 0 || b <= 0) {
        printf("Your inputs are invalid, please reenter 2 positive integers\n");
        scanf("%d%d", &a, &b);
        }
```

```
/*scan the operation to call the coresponding function*/
printf("Please choose an operation from (+ or -)");
char s;
scanf("%s", &s);

if (s == '+')
{
	//print the signed int
	printf("%i+%i = %i\n", a, b, sum(a, b));

	//print the 2's comp
	printf("2's comp: ");
	decimal_to_binary(sum(a,b));
	printf("\n");
}
else if (s == '-')
{
	//prin the signed int
	printf("%i-%i = %i\n", a, b, diff(a, b));

	//print the 2's comp
	printf("2's comp: ");
	decimal_to_binary(diff(a,b));
	printf("\n");
}
//print error if user entered invalid inputs
else
```

```
	{
		printf("Your input was invalid, please restart the program\n");
	}
	return 0;
}
```