Homework 1

Shiyu Wang

* 1. 5310

53 / 8 = 6……5 5

6 / 8 = 0 …….6 65

0 done

So 5310 = 658

* 1. FA16

FA16 = 1111(F) 1010(A) = 1111 10102

* + - 1. 02 = 011(3) 111(7) 010(2) = 3728
  1. 1910

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 1910 = 100112 = 000100112 (2’s comp)

* 1. -1310

13 / 2 = 6…..1 1

6 / 2…..3 01

3 / 2 = 1……1 101

1 / 2 = 0……1 1101

0 done

So 1310 = 11012 = 000011012

-1310 = 111100102 (1’s comp) = 111100112 (2’s comp)

* 1. -2310

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 2310 = 101112

-2310 = 111010002 (1’s comp) = 111010012 (2’s comp)

* 1. ED16

ED16 = 1110(E) 1101(D) = 111011012 (2’s comp)

111011012 – 1 = 111011002 (1’s comp)

111011002 (1’s comp) = - 000100112 = -1910

* 1. 0xABCD OR 0x9876

ABCD16 = 1010(A) 1011(B) 1100(C) 1101(D) = 1010 1011 1100 11012

987616 = 1001(9) 1000(8) 0111(7) 0110(6) = 1001 1000 0111 01102

1010 1011 1100 11012

OR

1001 1000 0111 01102

= 1011 1011 1111 11112

= BBFF16

= 0xBBFF

* 1. 0xFEED AND (NOT(0xBEEF))

FEED16 = 1111(F) 1110(E) 1110(E) 1101(D) = 1111 1110 1110 11012

BEEF16 = 1011(B) 1110(E) 1110(E) 1111(F) = 1011 1110 1110 11112

Not(0xBEEF) = 0100 0001 0001 00002

1111 1110 1110 11012

AND

0100 0001 0001 00002

= 0100 0000 0000 0000

= 400016

= 0x4000

* 1. 01000010 01100101 01110011 01110100 00100000 01101111 01100110

010000102 011001012 011100112 011101002 001000002 011011112 01100112

= 6610 10110 11510 11610 11110 5110

By ASCII table

= B e s t o 3

* 1. 01001100 01110101 01100011 01101011 00100001

010011002 011101012 011000112 011010112 001000012

= 7610 11710 9910 10710 3310

By ASCII table

= L u c k !

* 1. 5 bits

14 / 2 = 7……0 0

7 / 2 = 3……1 10

3 / 2 = 1……1 110

1 / 2 = 0……1 1110

0 done

1410 = 01110

­-1410 = 10010 (2’s comp)

* 1. 6 bits

1410 = 001110

-1410 = 110010 (2’s comp)

* 1. 7 bits

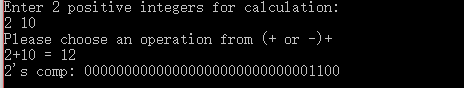
1410 = 0001110

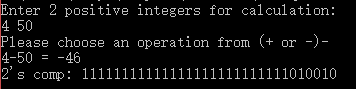
-1410 = 1110010 (2’s comp)

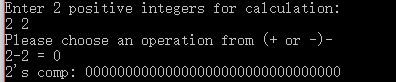
From the experiments, I found two’s complement is easier to store ints, when you need to extend the width of the register the value is being stored in. With two's complement, storing a less bit number in a high bit register is a matter of repeating its most significant bit. On the other hand, if we just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

1. 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;

}