CSE2312-002, 004 (Fall 2021)
Homework #1

Notes:
All numbers are in base-10 unless otherwise noted.
If part of a problem is not solvable, explain why in the answer area.
The 0x number prefix indicates the number is base-16 as in the C language.

The target date to complete this homework set is two days prior to the next quiz.

This homework set will not be graded, but please solve all of the problems to prepare for the quizzes and exams.

1. Convert the following numbers between bases:

Leading zeros and gaps added to show byte or half-byte alignment are not required but helpful

a. 10111011 (base-2) = 187 (base-10)

b. 10111011 (base-2) = 0xBB (base-16)

c. 0x5249 = 5249 (base-16) = 0101 0010 0100 1001 (base-2)

d. 0x5249 = 5249 (base-16) = 21,065 (base-10)

e. 16383 = 0011 1111 1111 1111 (base-2)

f. 4095 = 0x0FFF (base-16)

2. What is the range of the following C99 variable types assuming the processor uses two's compliment arithmetic for signed number representation?

a. uint8_t — `0` to `255`

b. uint16_t — `0` to `65,535`

c. uint32_t — `0` to `4,294,967,295`

d. int8_t — `-128` to `127`

e. int16_t — `-32,768` to `32,767`

f. int32_t — `-2,147,483,648` to `2,147,483,647`

3. Write the binary representation of the C99 variables given below.

Example: for uint8_t x = 13, the answer would be answer is: 0000 1101 (base-2)

a. uint8_t x = 27;

```
0001 1011
```

b. uint8_t x = 122;

```
0111 1010
```

c. uint8_t x = 215;

```
1101 0111
```

d. uint8_t x = 40;

```
0010 1000
```

e. int8_t x = -40;

```
1101 1000
```

f. int8_t x = -103;

```
1001 1001
```

g. int8_t x = 103;

```
0110 0111
```

h. uint16_t x = 13000;

```
0011 0010 1100 1000
```

i. int16_t x = 13000;

```
0011 0010 1100 1000
```

j. int16_t x = -13000;

```
1100 1101 0011 1000
```

k. uint32_t x = 262144;

```
0000 0000 0000 0100 0000 0000 0000 0000
```

l. int32_t x = -50;

```
1111 1111 1111 1111 1111 1111 1100 1110
```

m. int32_t x = 50;

```
0000 0000 0000 0000 0000 0000 0011 0010
```

4. Write the status of the Carry (C), Zero (Z), and Sign (S) flags after an 8-bit ALU performs an ADD operation on the following 8-bit arguments (a and b):

Hint: Remember that the ALU just sees bits and does not know if the numbers represent signed or unsigned numbers.

a. uint8_t a = 91, uint8_t b = 23

C = 0, S = 0, Z = 0

```
 0101  1011    91
 0001  0111    23
 ---------    ---
 0111 0010    114
```

b. uint8_t a = 102, uint8_t b = 3

C = 0, S = 0, Z = 0

```
 0110 0110  102
 0000 0011    3
 ---------  ---
 0110 1001  105
```

c. int8_t a = 32, int8_t b = -22

C = 1, S = 0, Z = 0

```
  0010 0000   32
  1110 1010  -22
  ---------  ---
1 0000 1010   10
```

d. int8_t a = -32, int8_t b = 22

C = 0, S = 1, Z = 0

```
 1110 0000  -32
 0001 0110   22
 ---------  ---
 1111 0110  -10
```

e. int8_t a = 100, int8_t b = -100

C = 1, S = 0, Z = 1

```
  0110 0100  100
  1001 1100 -100
  ---------  ---
1 0000 0000    0
```

f. int8_t a = -130, int8_t b = 100

Not Possible, -130 < -128 => can't fit in 8 bit ALU

g. int8_t a = -32, int8_t b = 72

C = 1, S = 0, Z = 0

```
  1110 0000     -32
  0100 1000      72
  ---------      --
1 0010 1000      40
```

5. Assuming an 8-bit ALU, show the status of the Zero (Z), and Sign (S) flags and the result after each operation. Note the bases carefully.

a. arg1 = 33, arg2 = 2; result = arg1 OR arg2

Z = 0, S = 0

```
0010 0001
0000 0010
---------
0010 0011
```

b. arg1 = 0x23, arg2 = 0x14; result = arg1 OR arg2

Z = 0, S = 0

```
0010 0010
0001 0100
---------
0011 0110
```

c. arg1 = 0x2C, arg2 = 0x78; result = arg1 AND arg2

Z = 0, S = 0

```
0010 1100
0111 1000
---------
0010 1000
```

d. arg1 = 0xA5; result = NOT arg1

Z = 0, S = 0

```
1010 0101
---------
0101 1010
```

e. arg1 = 29; result = NEG arg1

Z = 0, S = 1

```
0001 1101
---------
1110 0010
        1
---------
1110 0011
```