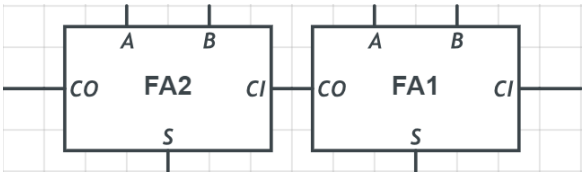
Name: Guneev Arora Student Number: 400477579

Assignment 1: Digital Logic and Representations

Task 1: Digital Logic (4 points)

a. You are given two 8-bit full adders. Devise a circuit that uses the adders to perform addition of two 16-bit numbers. You may, if necessary, use any additional NAND or NOR gates. Simply describe the motivation of your design. (1pt)



As each of the full adders take 8 bits from A and B 0-15 which would be 16 bits. Thus, to add 16 bits you would require 2 Full Adders as each add 8 bits when connected by the carry output (CO) of the first and input (CI) of the second.

b. What is a decoder chip and what is it used for? (1pt) If a decoder has 3 input pins, how many output pins will it have? (1pt)

A decoder chip takes inputs and converts it into a set of combinations/signals. It takes n inputs and converts it into 2^n outputs.

So as the decoder takes 3 inputs, it will have 2^3 output pins which would be equal to 8 combinations.

c. A three-bit binary counter receives the following sequence of inputs: 1100101001. What would be the output of the counter at each time step and for each output line? (1pt)

Step 1: 1 = Counter: 000 = 0

Step 2: 1 = Counter: 000 = 0

Step 3: 0 = Counter: 000 = 0

Step 4: 0 = Counter: 000 = 0

Step 5: 1 = Counter: 001 = 1

Step 6: 0 = Counter: 001 = 1

Step 7: 1 = Counter: 010 = 2

Step 8: 0 = Counter: 010 = 2

Step 9: 0 = Counter: 010 = 2

Step 10: 1 = Counter: 011 = **3**

Name: Guneev Arora Student Number: 400477579

Task 2: Data Representation (3 points)

a. Use mathematical induction to prove that k bits can represent 2k distinct values. (1pt)

K bits → 2^k values

Base Case:

1 bit = 2^1 values

2^1 = 2 values

1 bit = 2 distinct values

Inductive step:

k + 1 values = 2^(k + 1)

2^(k + 1) = 2^k . 2^1

2^k . 2^1 = 2 . 2^k

Due to the inductive hypothesis if
k bits = 2^k and base case this is True

b. A four-bit architecture uses a two's complement representation system to interpret a 4-bit byte. What are the maximum and minimum values that can be represented in this 4-bit memory? (1pt) What would the answer be if the byte is interpreted as an unsigned integer? (1pt)

When k = 4:

Signed:

Max:

2^(k - 1) – 1 = 2^(4 - 1) - 1

= 2^3 - 1 = 8 – 1

= 0111

= **7**

Min:

-2^(k - 1) = -2^(4 - 1)

= -2^3

= 1000

= **-8**

Unsigned:

Max:

1111

= **15**

Min:

As it can't go negative thus 0000

= **0**

Task 3: Integer Representations (4 points)

a. Write a C program that allocates a byte array in memory. (Note: at least 8 bytes) (1pt)

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

int main() {
    int bits = 8; //Change this to increase
    char *Data = (char *)malloc(bits);
}
```

b. Interpret the array pointer as an integer pointer and store the integer 0x04030201 in memory. Print out the bytes in order and determine if the integer was stored in little- or big-endian order. (1pt)

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

int main() {
    int bits = 4; //Change this to increase or decrease
    char *Data = (char *)malloc(bits);


    int value = 0x04030201;
    for(int i = 0; i < bits; i ++) {
        Data[i] = (value >> (8 * i)) & 255;
        printf("%02X ", Data[i]);
    }
    printf("\n");

 // Little endian due to the order being 01 02 03 04
}
```

c. Write a function that determines automatically if the architecture uses sign-magnitude or two's complement integer representation. Assume you know the endianness of the system. Hint: choose an appropriate integer to represent in memory as a byte array and compare the byte string to what you would expect to see under each representation. (2pt)

```c
void Question3c() {
    //Qc
    char testNum = (char)0b11111111;
    if(testNum == -1){
        printf("2's compliment");
    }

    else if(testNum == -127){
        printf("Signed");
    }

    else {
        printf("Neither/Wrong");
    }
}
```

Task 4: Floating-point representation (5 points)

In this task we will emulate an 8-bit floating point representation. We will assume a representation with a one-bit sign, 4-bit normalized mantissa and 3-bit biased exponent.

a. What are the smallest/largest numbers you can represent? (0.5pt) What is the smallest increment (i.e., step change between two neighboring numbers)? (0.5pt)

Smallest:

You can assume that the smallest value is -**15.5**

Due to a sign change.

Largest:

Sign = 0

Exponent = 110

Mantissa = 1111

= 1.1111 x 2^3 = 1111.1 = **15.5**

Smallest Increment:

= 0.001 x 2^1 = 0.01 = 1/8 = **0.25**

b. Write a function char toFloat(float arg) that takes a standard float variable, converts and stores it in a single byte and returns it as a char variable. (2pt) There are multiple possible ways to solve this problem. The recommended way is to use the function log2f to calculate the exponent. Next, check for limits and add the bias. Finally, copy the parts of the mantissa corresponding to the most significant bits. Make sure to shift and convert variables as appropriate

```
char toFloat(float num){
    int exponent_bit = 3;

    int conv = log2f(fabsf(num));
    int exponent = floor(conv);
    float mantissa = fabsf(num);
    mantissa = mantissa/pow(2, exponent) -1;
    int exp_bias = pow(2, (exponent_bit-1))-1;
    exponent = exponent + exp_bias;

    int start = 0b00000000;
    char *total = (char *)&start;
    if(num<0){
        *total |= 0x80;
    }
    else {
        *total &= 0x7F;
    }

    *total |= exponent << 4;
    *total |= (int)(mantissa * pow(2, 4));

    return *total;
}
```

c. Write a function to print out the converted char variable to screen as a bit string. Use the function to verify that your encoding works correctly for a range of input numbers. (2pt)

```
void printBit(char num){
    for (int i = 0; i < 8; i++){
        printf("%d", (num >> 7 - i) & 1);
    }
    printf("\n");
}


int main() {
    float num = -15.5; //change number accordingly
    char bin = toFloat(num);
    printBit(bin);
    return 0;
}
```