Lab 4 Fixed-Point Expression and Operations

Lab Report

Student: Jeremiah Webb

Student ID: 2545328

Instructor: Dr. Jianhua Liu

Section #2

Introduction

Being able to express two real numbers in the fixed point Qm.n format, we can use real numbers such as floating point numbers. Being able to visualize how fixed-point mathematics operates in C shows how computers function in arithmetic, subtraction, and multiplication. One can see that using fixed Point mathematics develops error too, it however generates a close approximation of the real floating number.

Screenshots

```
Jeremiah Webb #2545328
804-2688
Task 2: Print Hexadecimal Forms of A1 and A2
A1: 0x324
A2: 0xFFFFF580

Task 3: Calculate results using floating-point numbers directly

Float Point Addition: -7.358500
Float Point Subtraction: 13.641500
Float Point Multiplication: -32.985748
Task 4. Calculate results using fixed-point numbers
Fixed Point Addition: -7.359375
Fixed Point Subtraction: 13.640625
Fixed Point Multiplication: -32.976562
```

Code Snippet

```
/*
Lab 4 CEC 322
Name: Jeremiah Webb
Student ID: 2545328
*/
#include <stdio.h>
#include <stdint.h>
#include <stdbool.h>
#include <math.h>
```

```
#include <stdlib.h>

float f1 = 3.1415;
float f2 = -10.5;
float f3 = 0;
float f4 = 0;

float f_sum, f_sub, f_mul;
int16_t A1, A2, A_sum, A_sub, A_mul;

// We use Qm.n format here:
int m = 7;
int n = 8;
//Converting to fixed, multiply by 2^n
float convertToFixed(float num1, int power){
     float temp = 0;

     temp = ((num1) * (pow(2, power)));
     return(temp);
}
//do round function to round numbers, return them
int convertToInt(float num1){
     int temp = 0;
     temp = round(num1);
     return(temp);
}
//Multiply fixedpoint number, don't forget to shift by n to save
bits.
int16_t fixedPointMul(int16_t a, int16_t b) {
     int32_t mul = (int32_t) a * (int32_t) b;
     return mul >> n;
}

float fromFixedPoint(A16_t i) {
     //Convert to a float first so that we will be able to
divide and get the result out without
     //having integer division lose our bits!
     float f = i;
     float coeffecent = 1.0f / powf(2.0f, n);

     float result = f * coeffecent;


     return result;
}
```

```
int main(void) {

     printf("\nJeremiah Webb #2545328\n");
     // Calculate f * 2^n for both numbers convert to Fixed
Point
     f3 = convertToFixed(f1, n);
     f4 = convertToFixed(f2, n);

     //Convert to integer from float
     A1 = convertToInt(f3);
     A2 = convertToInt(f4);

     printf("%d", A1);
     printf("%d", A2);

     printf("\nTask 2: Print Hexadecimal Forms of A1 and A2\n");
     printf("A1: 0x%X\n", A1);
     printf("A2: 0x%X\n", A2);

     //Do math Task 3;
     f_sum = f1 + f2;
     f_sub = f1 - f2;
     f_mul = f1 * f2;
     printf("\nTask 3: Calculate results using floating-point
numbers directly\n");
     printf("\nFloat Point Addition: %f\n",f_sum);
     printf("Float Point Subtraction: %f\n",f_sub);
     printf("Float Point Multiplication: %f\n",f_mul);

     //Do Math Task 4
     A_sum = A1 + A2;

     A_sub = A1 - A2;

     A_mul = fixedPointMul(A1, A2);
     printf("Task 4. Calculate results using fixed-point
numbers");
     printf("\nFixed Point Addition: %f\n",
fromFixedPoint(A_sum));
     printf("Fixed Point Subtraction:
%f\n",fromFixedPoint(A_sub));
     printf("Fixed Point Multiplication:
%f\n",fromFixedPoint(A_mul));
}
```

Questions:

Explanation of how you did the calculation of the multiplication using Q7.8.

Realizing that when multiplying the two numbers A1 and A2, the numbers would need to be shifted (n) times to get the proper format of bits. When C sees the bits, the right most bit is 1, however, in Fixed Point, we need ensure that when the two numbers are multiplied, shifting needs to occur to maintain data integrity.

Explanation of how you printed out `A_sum` as a real number.

I had to revert the fixed-point number into a float number by multiplying by the inverse of the coefficient: 1/(2^n), n being the n in Qm.n. Multiplying the inverse of the coefficient with the fixed-point number creates the floating-point number, and thus a real number.

Narrative

The lab went well and sparked an interest in how computers create accuracy. Discovering the use of fixed-point numbers presented the idea of how numbers in computers are generally computed. In the real world, exact calculations are needed to do anything, fixed point expressions are a computer's way of telling the exact amount in a number. However, with a small amount of error, this personally spiked my interest to go and investigate IEEE 754 on how computers find the exact calculations of numbers. Seeing how computers can store fixed-point numbers and floating-point numbers helps visualize the mathematics behind it too.

Results

I can understand how fixed-point intervals are used now, however, a numeric difference is shown amongst the original floating points numbers and fixed-point numbers. Although the errors are small, the error still exists.