ENCM 515: Digital Signal Processors – Lab 1

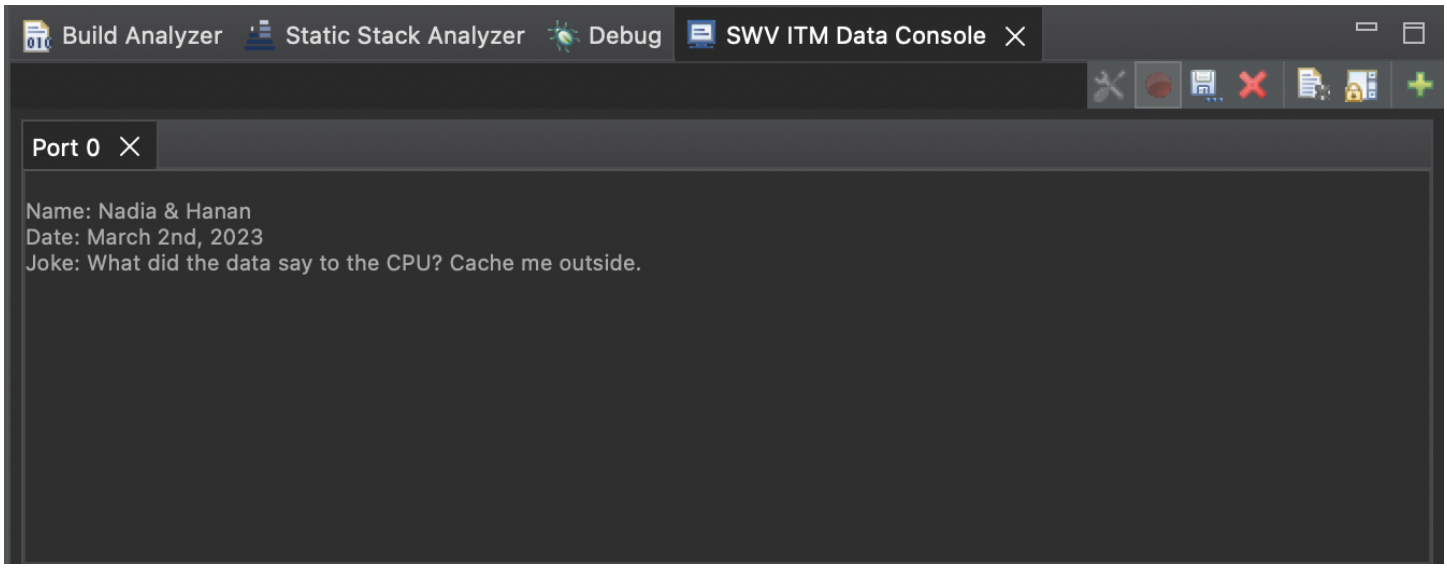
Group #	7	Date: March 2nd, 2023
Student name(s)	Hanan Anam	
	Nadia Duarte Amador	
Instructor/TA	Dr. Tan	

To submit:

- **Add your lab sheet (as PDF if possible) and your main.c to a zip archive and upload to D2L.**

PART ONE

Q1> Screenshot of the console output (name, date, joke)



Q2> Time stamps and cycles for the two ITM Port 31 entries (screenshot should suffice).

Index	Type	Data	Cycles	Time(s)
3	ITM Port 31	1	25634269	267.023635 ms
4	Sync	48	39634262	412.856896 ms
5	ITM Port 31	2	49647296	517.159333 ms

How many clock cycles are there between the two writes to the ITM Port 31?

24,013,027

How much time has elapsed?

250.14ms

What is the average amount of time taken for each loop iteration?

250.14/2000000 = 0.125ms per loop iteration

Floating point experiment

Q3 > > How many clock cycles are taken to perform the floating point function?

2	ITM Port 31	1	25450024	265.104417 ms
3	ITM Port 31	2	25450053	265.104719 ms

2	ITM Port 31	1	26309205	274.054219 ms
3	ITM Port 31	2	26309238	274.054563 ms

3	ITM Port 31	1	25894263	269.731906 ms
4	ITM Port 31	2	25894298	269.732271 ms

3	ITM Port 31	1	26877140	279.970208 ms
4	ITM Port 31	2	26877175	279.970573 ms

2	ITM Port 31	1	27123115	282.532448 ms
3	ITM Port 31	2	27123150	282.532813 ms

Test	Value of A	Value of B	Clock Cycles needed (6-5)
1	0.5	0.125	29
2	2.5	0.125	33
3	8.2	0.7	35
4	10.2	1.1	35
5	10.5	1.125	35

Q4 > How many clock cycles are taken to perform the floating point function now? Include a screenshot of the SWV Trace Log.

2	ITM Port 31	1	27396657	285.381844 ms
3	ITM Port 31	2	27396731	285.382615 ms

3	ITM Port 31	1	25904587	269.839448 ms
4	ITM Port 31	2	25904665	269.840260 ms

3	ITM Port 31	1	27325221	284.637719 ms
4	ITM Port 31	2	27325301	284.638552 ms

2	ITM Port 31	1	26362177	274.606010 ms
3	ITM Port 31	2	26362257	274.606844 ms

2	ITM Port 31	1	26555252	276.617208 ms
3	ITM Port 31	2	26555332	276.618042 ms

Test	Value of A	Value of B	Clock Cycles needed (6-5)
1	0.5	0.125	74
2	2.5	0.125	78
3	8.2	0.7	80
4	10.2	1.1	80
5	10.5	1.125	80

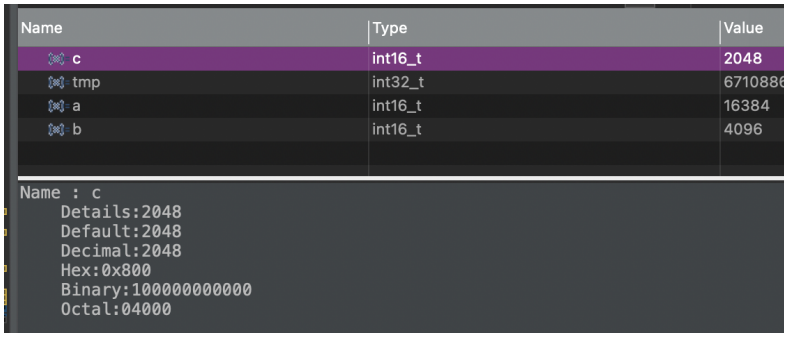
Fixed Point Experiment

Before fixing code:

Q5 > Fix the code snippet so that the correct values are loaded in a and b. Paste your completed function here.

After fixing code:

```
void FixedExperiment(void) {

int16_t c ;
int32_t tmp;

int16_t a = 0b0100000000000000; // should be 0.5;
int16_t b = 0b0001000000000000; // should be 0.125

tmp = a * b;

c = (int16_t)(tmp >> 15);

}
```



As we can see there result for c in binary is 0000100000000000 which in Q1.15 format corresponds to $2^{-4} = 0.0625$, which matches the result from 0.5*0.125 = 0.065. It took 33 cycles to complete this function, or 0.32 us.

Index	Type	Data	Cycles	Time(s)
3	ITM Port 31	1	25747280	268.200833 ms
4	ITM Port 31	2	25747313	268.201177 ms

CMSIS Experiment

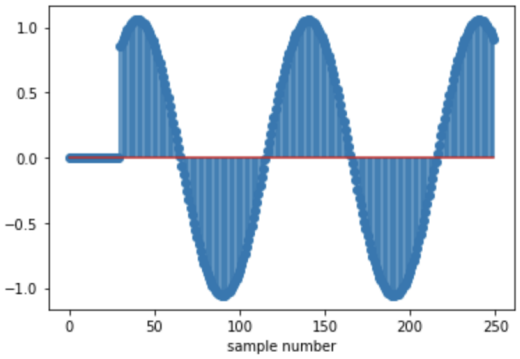
Q6> Take a screenshot of the Variables window after calling these functions. What do you observe?

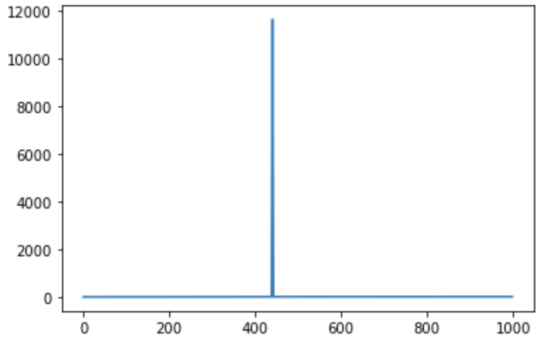
Name	Type	Value
arr1	float [10]	0x2001ffc0
arr1[0]	float	-0.200000003
arr1[1]	float	0.150000006
arr1[2]	float	0.800000012
arr1[3]	float	-0.5
arr1[4]	float	3
arr1[5]	float	-5
arr1[6]	float	10
arr1[7]	float	2
arr1[8]	float	0.899999976
arr1[9]	float	-2
arr2	float32_t [10]	0x2001ff98
arr2[0]	float32_t	-0.200012207
arr2[1]	float32_t	0.149993896
arr2[2]	float32_t	0.799987793
arr2[3]	float32_t	-0.5
arr2[4]	float32_t	0.999969482
arr2[5]	float32_t	-1
arr2[6]	float32_t	0.999969482
arr2[7]	float32_t	0.999969482
arr2[8]	float32_t	0.899993896
arr2[9]	float32_t	-1
arr3	q15_t [10]	0x2001ff84
arr3[0]	q15_t	-6554
arr3[1]	q15_t	4915
arr3[2]	q15_t	26214
arr3[3]	q15_t	-16384
arr3[4]	q15_t	32767
arr3[5]	q15_t	-32768
arr3[6]	q15_t	32767
arr3[7]	q15_t	32767
arr3[8]	q15_t	29491
arr3[9]	q15_t	-32768

The original array elements are shown in arr1. After being converted into Q1.15 in arr3, and then back into floats in arr2, the original values within the range [-1, 1) are the same, with a few small differences in rounding. Numbers outside of that range are rounded down to either -1 or close to +1. This implies saturation when numbers exceed the maximum range that can be represented by a Q1.15 number.

PART TWO

Q7> Screenshots of the first 250 filtered samples and the spectrum (from floating point FIR)





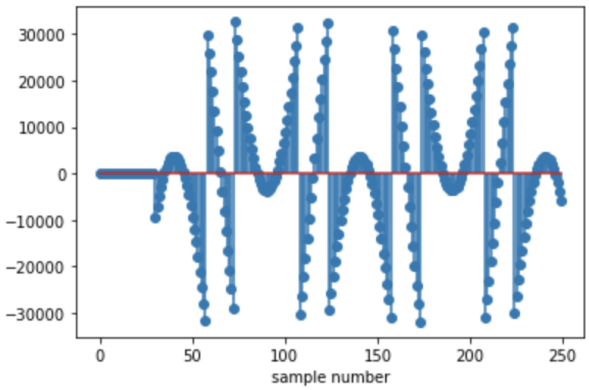
How much time is taken to filter the signal? Is this filtering going to be "good enough" for our sample rate?

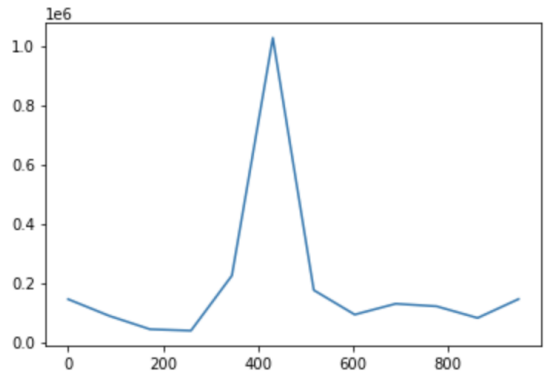
Index	Type	Data	Cycles	Time(s)	Extra info
3	ITM Port 31	1	26224564	273.172542 ms	
4	Sync	48	42224556	439.839125 ms	
5	Sync	48	58224548	606.505708 ms	
6	ITM Port 31	2	70306651	732.360948 ms	

It takes 732.360948-273.172542 = 459.2 ms to filter the signal. The total number of samples filtered can be found using the duration of the audio clip and the sample rate, 44100 Hz * 0.5s = 22050 samples. This means that on average, it takes 459.2ms/22050 = 20.83 us to process each sample. Assuming the samples are arriving in real-time, they arrive every 1/44100 Hz = 22.7 us.

Since the processing time is less than the sample interval, in theory the filtering should be good enough. But, the margin is very narrow (only 10% of the sample period) so missed samples are possible since some iterations of the filter may run more slowly than the average.

Q8> > Screenshots of the first 250 filtered samples and the spectrum (from fixed point FIR)





How much time is taken to filter the signal? Is this filtering going to be "good enough" for our sample rate?

Index	Type	Data	Cycles	Time(s)	Extra info
2	ITM Port 31	1	26095075	271.823698 ms	
3	Sync	48	34095071	355.156990 ms	
4	Sync	48	52095062	542.656896 ms	
5	ITM Port 31	2	63008376	656.337250 ms	

Now, it takes 384.51 ms to filter the signal. Using the same calculations as in Q7, it takes 384.51ms/22050 = 17.4 us to filter each sample. Compared to the sample time of 22.7 us, this is a much better cushion time to have between the arrival of each sample and the time required to process it. Note that overflow can be seen on the graph above and that saturation is required to fix it.

Q9> Write a reflection of what you have observed and learned in this lab [approx.. 200 words max.]. Which lecture concepts did you encounter here?

This lab covered the differences between floating and fixed point implementations for basic operations and filtering. Each implementation had its pros and cons, with fixed point operations running faster, but being limited in range [-1,1) for Qm.n format, where normalization is required. We saw a particularly dramatic effect of overflow in the filtered graph generated in question 8. Even if saturation had been implemented, the filtered plot would be quite distorted.

The timing improvements were quite dramatic (~25%) when we switched over to fixed point in part two of the lab. It seems like fixed point applications would be most useful in an application where the ranges of input and output data are known, to avoid distortion of the results due to overflow or saturation. Fixed point implementations would also be useful in a time-sensitive application. It is a little difficult to see the use of speeding up a process by a few microseconds when it's applied to such a small chunk of data like in this lab, but we can imagine how it could dramatically improve system performance with larger datasets or higher frequency sampling.