19281858 / 84e6 = 229.55ms = .229s

- What processor (ECG, EFM, or EFH) you used.
  - **The ECG was used.**
- The list of changes you made to the fft files you ported (part 5).
  - **Had to comment out several printf statements most of which were error handling messages. As well as changing the #include statement for the .h file from being <fourier.h> and <ddcmath.h> to "fourier.h" and "ddcmath.h"**
- The execution times in cycles and seconds obtained in parts 9 through 16 in a table like the following:

Step	Configuration	Time in Cycles (decimal please)	Time in Seconds
9, 10	No Optimization, Cache Off, Prefetch Off, WS = 7	19281858	.229s
11	Optimization 1, Cache Off, Prefetch Off, WS = 7	16760551	.19953s
12	Optimization 1, L1 Cache Off, Prefetch Off, WS = 1	14297317	.17020s
13	Optimization 1, L1 Cache Off, Prefetch On, WS = 7	14022997	.16694s
14	Optimization 1, L1 Cache Off, Prefetch On, WS = 1	13386229	.15935s
15	Optimization 1, L1 Cache On, Prefetch Off, WS = 7	1324551	.01576s
16	Optimization 1, L1 Cache On, Prefetch On, WS = 1	1322637	.01574s
17	Optimization 1, L1 Cache On, Prefetch On, WS = 1 Second Pass	1320419	.01571s

- Your assessment as to how effective each of the 3 performance settings in steps 9 through 16 were. In particular, answer the following questions:

a) Which feature, by itself, improved performance the most?

The cache improved the performance the most by itself.

b) How do you account for the fact that turning the prefetch unit on (step

13) gives a greater performance boost than setting the wait states to 1 (step 12)? Would this be true for all programs? State why or why not.

Enabling prefetch which grabs more instructions at one time we reduce the number of wait states which would work with any program.

c) Why was there less of a performance gain from turning the prefetch unit on and setting wait states to 1 between steps 15 and 16 than there was between steps 11 and 14?

Because the cache is enabled if we get a hit in the cache the wait states and prefetch don't come into play so it has less of an effect on the overall time.

d) Why was the execution time reduced on the second pass in step 17?

Since we had just used that it was setup in the cache so that we had a cache hit and didn't have to go find it.

- Copy and paste your commented main routine source code (I don't need copies of any of the fft files) into the end of the document you used for your answers. Use a monospaced font (i.e. 10 pt Lucida Console) for the code section. Then save or print this as a pdf and submit it to your lab instructor electronically via Canvas. (**Note**, if you choose to print code from MPLab to pdf, I would ask you to please set the print options to change the default text font size to 12 points monospaced, turn on line wrapping, turn off the border and set the set the background color to white. The default header and footer settings are fine.)

```
/***************************************************************************
 * Author: Chris Thomas
 * Date: 10/11/2022
 * Project: Lab 2 - Performance Issues
 * Updates: N/A
 ***************************************************************************/
// PIC32MZ2048ECG144, EFM144 or or EFG144 based HMZ144 board Configuration Bit Settings
// DEVCFG2
```

```
#if defined(__32MZ2048EFG144__) || defined(__32MZ2048EFM144__)

#pragma config FPLLIDIV = DIV_4        // System PLL Input Divider (4x Divider) for 24MHz clock (Rev C1 board w
EFG) 24MHz/4 = 6MHz

                                        // also 24MHz clock rev C board w EFM (weird - went back to C. rev D also is EFM but with
Osc)

#pragma config UPLLFSEL = FREQ_24MHZ    // USB PLL Input Frequency Selection (USB PLL input is 24 MHz)

#else

#pragma config FPLLIDIV = DIV_2        // System PLL Input Divider (2x Divider) for 12 MHz crystal (Rev B and C
boards w ECG) 12MHz/2 = 6MHz

#pragma config UPLLEN = OFF            // USB PLL Enable (USB PLL is disabled)

#endif

#pragma config FPLLRNG = RANGE_5_10_MHZ // System PLL Input Range (5-10 MHz Input)

#pragma config FPLLICLK = PLL_POSC      // System PLL Input Clock Selection (POSC is input to the System PLL)

#pragma config FPLLMULT = MUL_112       // System PLL Multiplier (PLL Multiply by 112) 6MHz * 112 = 672MHz

#pragma config FPLLODIV = DIV_8        // System PLL Output Clock Divider (8x Divider) 672MHz / 2 = 84MHz


// DEVCFG1

#pragma config FNOSC = SPLL            // Oscillator Selection Bits (Primary Osc (HS,EC))

#pragma config FSOSCEN = OFF           // Secondary Oscillator Enable (Disable SOSC)

#if defined(__32MZ2048EFG144__)

#pragma config POSCMOD = EC            // Primary Oscillator Configuration EC - External clock osc

                                        // Rev C1 board w EFG uses an Oscillator (Rev D boards too))

#else

#pragma config POSCMOD = HS            // Primary Oscillator Configuration HS - Crystal osc

                                        // Rev B and C (w ECG or EFM) use Crystals

#endif

#pragma config FCKSM = CSDCMD          // Clock Switching and Monitor Selection (Clock Switch Disabled, FSCM
Disabled)

#pragma config FWDTEN = OFF            // Watchdog Timer Enable (WDT Disabled)

#pragma config FDMTEN = OFF            // Deadman Timer Enable (Deadman Timer is disabled)

#pragma config DMTINTV = WIN_127_128    // Default DMT Count Window Interval (Window/Interval value is
127/128 counter value)
```

```
#pragma config DMTCNT = DMT31          // Max Deadman Timer count = 2^31


// DEVCFG0

#pragma config JTAGEN = OFF            // JTAG Enable (JTAG Disabled)

#pragma config ICESEL = ICS_PGx2       // ICD/ICE is on PGEC2/PGED2 pins (not default)


#include <xc.h>

#include <math.h>

#include "FOURIER.h"

#define NSAMP 256

#define TPIN (2 * M_PI/NSAMP)

float in[256];                                         //set up arrays

float real[256];

float imag[256];

float mag[256];


int main()

{

   int n;

   int i;

   int j;

   SYSKEY = 0; // Ensure lock

   SYSKEY = 0xAA996655; // Write Key 1

   SYSKEY = 0x556699AA; // Write Key 2

   PB3DIV = _PB3DIV_ON_MASK | 0 & _PB3DIV_PBDIV_MASK; // 0 = div by 1, 1 = div by 2, 2 = div by 3 etc up to
128

   T2CON = ((0 << _T2CON_TCKPS_POSITION) & _T2CON_TCKPS_MASK);

   T2CON = ((1 << _T2CON_T32_POSITION) & _T2CON_T32_MASK);

   PRECON = (1 & _PRECON_PFMWS_MASK) | ((2 << _PRECON_PREFEN_POSITION) & _PRECON_PREFEN_MASK);


   for(n = 0; n < 256; n++)                            //fill arrays
```

```
    {
        in[n] = 256 * sin(8 * 2 * M_PI / NSAMP * n);
    }

    for(j = 0; j < 3; j++)
    {
        T2CONSET = 0x8000; // Start the timer
        fft_float(NSAMP, 0, in, NULL, real, imag);
        T2CONSET = 0x7000; // Stop the timer
        TMR2 = 0;
    }

    for(i = 0; i < 256; i++)
    {
        mag[i] = sqrt((real[i]*real[i]) + (imag[i]*imag[i]));
    }
}
```