Homework Assignment #3

Due: **March 23, 2021, before midnight**

1. Put **T/F** to each statement if it is true/false, respectively. If the statement is false, briefly explain why or correct the false statement.
   a. Microcontrollers are usually composed of a processor, a program memory, a data memory, input/output ports, and a set of peripherals.

      T

   b. Most of the internal registers in the PIC24 architectures are 24-bit.

      F, they are 16-bit.

   c. All assembly instructions change the status bits in the status register (SR) after their execution.

      F, some but not all.

   d. PIC24 architecture has an instruction for 32-bit addition.

      False. Only up to 16-bit value addition is supported by an instruction.

   e. PIC24 architecture has an instruction for 64-bit addition.

      False Only up to 16-bit value addition is supported by an instruction.

   f. Literal addressing embeds a constant value in the instruction and that value cannot be changed.

      T

   g. Program Counter (PC) contains the address of the instruction being executed at the current time.

      T

   h. Unconditional branch instruction (e.g. `BRA LABEL`) doesn't need to check any conditions, so it always takes 1 instruction cycle. pg 130

      F, always takes 2 cycles.

i. Conditional branch instruction (e.g. `BRA NZ, LABEL`) need to check the condition, so it always takes 2 instruction cycles.

F, only 1 cycle if branch not taken. 2 if taken.

j. Stack manages local variables, function parameters, and return address.

T

k. In PIC24, Stack Pointer (SP) is referring to the most recently added (pushed) element and W15 is used for SP.

F, SP refers to next available word (Top of stack).

l. Stack operation `PUSH Ws` is equivalent to `MOV Ws, [W15++]`.

T

m. Stack operation `POP Wd` is equivalent to `MOV [W15--], Wd`.

T

n. When we set up a timer, it is always a good practice to use high prescaler.

F, You don't always need a high prescalar, high prescalars when not needed lose precission.

o. It is generally recommended to use polling over interrupt.

T

2. Maximum delay using Timer 1
   a. What is the maximum delay that Timer 1 (16-bit) can generate? Assume that the PIC24 microcontroller is running at 16MHz.

$16\text{ MHz} = 16{,}000{,}000 \ \underline{\text{cycles}}$

$\left(\frac{1}{16\text{ MHz}}\right) = 62.5\times10^{-9} \frac{s}{\text{cycle}}$

Max Prescalar = 1:256
Max # = 65536

$256 \cdot 65535 \text{ cycles} = 16777216$

$$(16777216 \text{ cycles})\left(62.5\times10^{-9} \frac{s}{\text{cycle}}\right) = \boxed{1.048576\,s}$$

   b. Write a program in C to implement the maximum delay using Timer 1. Reuse the example code `int_first_t1.c` which is available in course Canvas. Implement both polling and interrupt versions. Submit your c file.

   c. Measure the time using Stopwatch and report the results. Set a breakpoint at the line `IFS0bits.T1IF = 0;` (or `_T1IF = 0;`) in the infinite loop (polling) and the `_T1Interrupt` ISR (interrupt) and measure the time between the line using Stopwatch. Report the results of the Stopwatch.

with Interrupt

Target halted. Stopwatch cycle count = 16777216 (1.048576 s)

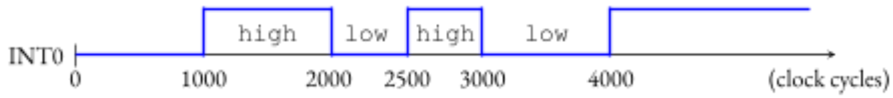
With Polling

Target halted. Stopwatch cycle count = 16777287 (1.04858 s)

Difference $= |16777216 - 16777287| = 71$ cycles more with polling.

$= .000004$ s more with polling.

3. Assume that INT0 pin receives the following electric signal.



Note that the unit of time is the clock cycle. The goal is to measure the time (in clock cycles) of the high period and the low period and save them to `high` and `low` variables. Use Timer 1 and its register `TMR1` to measure the time. To simulate the signal to INT0, use Stimulus (Pin/Register Actions)[1]. Reuse the example code `int_int0PinT1_determineDutyCycle.c` which is available in course Canvas.

a. Write a C program using **polling** and run the simulation. Report the value of `high` and `low` variables (i.e. 4 numbers).

☑ high	unsigned int	0x806	0x03E5	997
☑ low	unsigned int	0x808	0x01F1	497
☑ high2	unsigned int	0x80A	0x01F1	497
☑ low2	unsigned int	0x80C	0x03E5	997
<Enter new watch>				

b. Write a C program using an **interrupt** and run the simulation. Report the value of `high` and `low` variables (i.e. 4 numbers).

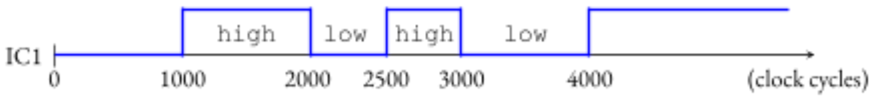
☑ high	unsigned int	0x800	0x03E5	997
☑ high2	unsigned int	0x804	0x01F1	497
☑ low	unsigned int	0x802	0x01F1	497
☑ low2	unsigned int	0x806	0x03E5	997

c. Are these `high` and `low` variables close to 1000, 500, 500, 1000? Briefly discuss why the results you obtained are not exactly the same as the ideal values.

They are close.
This is because there are a few instructions that occur after the interrupt bit is set that take a few cycles and occur before the timer value is pulled.

[1] **Stimulus**: You can find a video tutorial of using Stimulus in MPLAB X IDE at https://youtu.be/4gzeR4YnMFY.

4. Now let's assume that IC1 pin receives the same electric signal that we considered in the previous problem. Reuse the example code `ic_example_with_int.c` which is available in course Canvas.



a. Write a C program using **polling** and run the simulation. Report the value of `high` and `low` variables (i.e. 4 numbers).

☑ high	unsigned int	0x800	0x03E8	1000
☑ low	unsigned int	0x802	0x01F4	500
☑ high2	unsigned int	0x804	0x01F4	500
☑ low2	unsigned int	0x806	0x03E8	1000

b. Write a C program using an **interrupt** and run the simulation. Report the value of `high` and `low` variables (i.e. 4 numbers).

☑ high	unsigned int	0x800	0x03E8	1000
☑ low	unsigned int	0x802	0x01F4	500
☑ high2	unsigned int	0x804	0x01F4	500
☑ low2	unsigned int	0x806	0x03E8	1000

c. Are these `high` and `low` variables close to 1000, 500, 500, 1000? Briefly discuss the differences in the obtained values of IC1 and INT0 and explain why.

They are exactly 1000, 500, 500, 1000.
When you read from TMR1 with INTØ, it takes a couple cycles to read that TMR value but when you pull the cycles with IC1 you are polling the exact cycles at the time of that interrupt.

5. Consider the following assembly code:

```
.bss
Barr: .space 20
Aarr: .space 20
.text
_main:
      MOV   #10, W3         1
      MOV   #Bar, W10       1
      MOV   #Aar, W11       1
LOOP:
      MOV   [W10], [W11]    1
      INC2  WREG10          1
      INC2  WREG11          1
      DEC   WREG3           1
      BRA   Z, END_LOOP     1/2
      BRA   LOOP            2
END_LOOP:
      MOV   ...
```

10 cycles w3 > ∅
11 cycles w3 = ∅

a. Briefly explain what this code performs in one sentence.

This array Copies the elements in array Bar to Aar.

b. The code above is inefficient in terms of the number of instruction cycles it needs. Revise the code as efficiently as possible.

```
Barray : .Space  20
Aarray : .Space  20

   .text
 _main:
      MOV  #10, W3                  1
      MOV  #Bar, W10                1
      MOV  #Aar, W11                1

 Loop:
      MOV  [W10++], [W11++]         1
      DEC  W3                       1
      BRA  NZ, Loop                 2/1
```

7 w3 ≠ ∅
6 w3 = ∅