## Introduction

The goal of this lab is to understand the use of the interrupt requests from timers onboard the 8051 and from external inputs to the 8051. Interrupt requests are functions that are called when a certain condition is met. In the case of the onboard timers, an interrupt request can be generated after the timer overflows. The timing of these overflows can be calculated and be used in timing operations. In the case of external input, a pin is read for a value and the interrupt can be triggered by an edge or level in the voltage. Special syntax is needed in the C program to indicate such requests.

This lab is divided into three parts, the first involves and external input triggered interrupt. The second takes advantage of the internal clock’s dividers to provide a timing function. But because of these dividers, the timing functions can vary slightly due to rounding errors. Two copies of the timing will be written, one with rounding errors in the timing and one without. The last is a combination of the above by making a game based on measuring the time it takes the user to push a button in reaction to a timed event.

## Materials and Methods/Procedures (with Calculations)

The objective of part 1 of the lab was to write a program to respond to a push button switch through External Interrupt 0. This was accomplished by creating an int n to count the number of times the switch-- which was connected to the board through Port 0.2-- was grounded (ie. the input at P0.2 went from HIGH to LOW), and having that incremented and printed by an ISR (SW2\_ISR()) for interrupt 0. To deal with switch bouncing, Timer 0 (interrupt 1) was used to implement a delay before the switch ISR would increment and print again after being called. A .1uF capacitor was connected as well, between the switch and Ground, so its discharge time would provide extra insurance against bouncing.

The objective of part 2 was to use timers in different modes to generate an interrupt every (or approximately every) .1 secs. Timer 0 was set to use an external clock with a rate of 22118400 Hz as SYSCLK, counting at SYSCLK/12 with a 16-bit counter. Timer 2 was set to use SYSCLK/2, with a 16-bit counter, counting up. Both are set to overflow every .001 seconds, but print every .1 seconds.

Calculations are attached.

The objective of part 3 was to create a game that would measure response time using the push button, and print out the user's average time. Timer 2 was used to count the user's response time, overflowing once every millisecond; Timer 0 was used to deal with bounce, similar to how it was used in part 1. One ISR (SW2\_ISR()) was set up for the gameplay push button (using Port0.2), and another (SW1\_ISR()) was used for a reset-game button (using P0.3).

When the program is run, the game function playGame() is called in a main loop, which initiates the game. A wait-time between 0 and 100ms is generated, after which the screen flashes, signaling the player to press the button. The program then sets time ('overflows' variable) to 0 and waits for the player to press the button. Once the button is pressed, the elapsed time is recorded and the average recalculated, and the program then waits for the button to be pressed again to start the next round, or the other button to be pressed to reset the game variables. At this point the playGame() function ends and the program returns to main(), where playGame() gets called again. Additionally, a punishment for pressing the button too early (ie. before the screen flashes) was implemented, giving (arbitrarily decided) 200ms as the player's time for that round. A bit gameStarted is used to indicate whether or not the game has started, and is set to 1 by the push button ISR if it is 0 when it is called, so that if its value is 1 before the screen is flashed, the player gets penalized. Otherwise, gameStarted is set to 1 after the screen is flashed.

Results/Analysis

All objectives of this lab were successfully accomplished.

Button debouncing was added to part 1, and from that it was determined that ~10 overflows was a good enough wait time. A .1uF capacitor was added as extra insurance.

In part 2, Timer0 using SYSCLK/12 to overflow every .1ms was determined to be accurate, and Timer2 using SYSCLK/2 inaccurate, because of the necessity of rounding.

For part 3, the reset switch did not need to be debounced because when the ISR for said switch is called, the game variables are set back to 0; even if it gets called multiple times, the variables are already 0. An extra feature of penalizing the user for pressing the button too early was added to prevent the user from abusing the system by continuously pressing the button until the screen flashes.

## Conclusion

The results from the lab met the reasonable expectations of the goals. Given more time, refinement of the rounding errors in the timing in the first half of part two can be used to compare with the perfect timing of the second half to produce a noticeable difference and demonstrate the effects of the rounding errors. With even more time, the game in the third section of the lab could have been turned into a two player game, testing the reaction time of one player against the other’s reaction time.