CPE 3150 – PROJECT 2 TEAM #1 REPORT

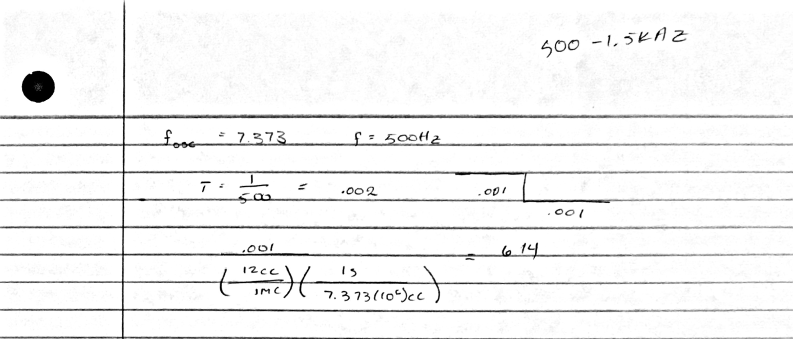
LanChau LeTran

Josh Cash

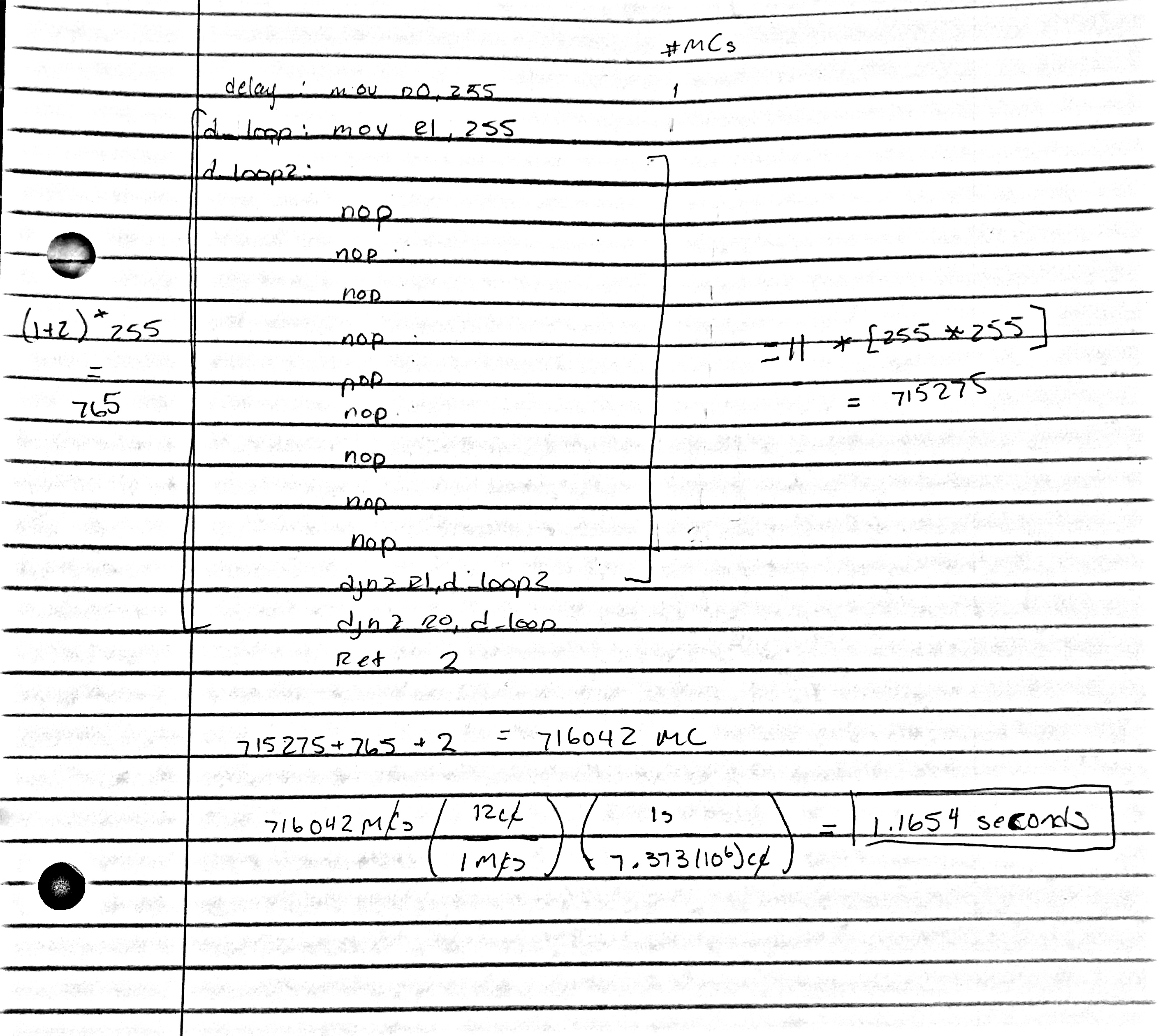
Alan Truong

SUMMARY:

For this project, we monitor the switch 8 (p0.3) as the positive key and switch 9 (p2.2) as the negative key. When each switch is pressed, we increment a counter, storing the value in register 4. As the user select the switch and the counter either increment or decrement, we display the counter value using 4 LEDS in binary (from 0-15) with a rollover if the value exceed 15 or goes below 0. The four LEDS from bit 4 to bit 0 is red (p2.4, p0.5, p0.6, p1.6). When the value rollover from 15 to 0, or 0 to 15, we send a square wave at the frequency of 500 Hz to the board. The math of using such frequency is shown below to get the value for the timer. When we tried to create the sound, we couldn’t get it to last long enough to be noticeable when it roll over. To deal with this, we wrapped the sound functionality code inside a delay loop to continuously send the square wave to the microprocessor long enough for the sound to last long enough, we ended up wrapping the sound function with a delay of 255 machine cycles.



Each time a switch is press, we increment or decrement accordingly, check for rollover, then display the lights. A big problem we had was when we press a switch and attempt to display a light, the board would display a binary number that did not match with what the value should be. The 8051 actually read in MULTIPLE input of the switch being pressed therefore incrementing or decrementing multiple time when we pressed the button. To fix this, after we press a switch and change the counter value, we added a delay so that the 8051 would only register only 1 incrementation or decrementation. The delay we added creates a 1.1654 seconds delay between each time a switch is triggered to change the counter and display the counter in binary. The math is shown below. This can be seen if you hold down the a switch, the light will be display for approximately 1.1654 seconds before the counter is changed.



SIMON BOARD:

The following pins and its respective functions where it is used are for the TEAM part of the project only. Additional pins used in the project that are only used in individual feature are listed by the individual themselves. The schematic for the board will be included below the list of pins used.

P0.3 – Increment switch

Functions:

* UP ( increment counter and check for roll over )
* LOOP ( Main function that detects any switch being pushed )

P2.2 – Decrement switch

Functions:

* DOWN ( decrement counter and check for roll over )
* LOOP ( Main function that detects any switch being pushed )

LIGHTS function:

This function is responsible for displaying the counter in binary. It moves the value of the counter from register 4 to ACC, then using bit addressing move each bit into c. The carry bit is moved to each in accordingly. The following pins were used for this feature from bit 0 to bit 3.

* P1.6 - (Blue light) Bit 0 of four-bit binary display
* P0.6 - (Amber light) Bit 1 of four-bit binary display
* P0.5 - (Yellow light) Bit 2 of four-bit binary display
* P2.4 - (Red light) Bit 3 of four-bit binary display

PROJECT CODE:

Also included as a separate file within the email as an .asm extension for better readibility.

#include <reg932.inc>  
  
cseg at 0  
  
    mov 0xA4, #0  
    mov 0x91, #0  
    mov 0x84, #0  
  
    mov r4, #0 ;start the counter at 0      
    mov r5, #0  
    mov r6, #0  
loop:  
    mov c, p0.3 ;detect if increment switch was pressed  
    jnc up ;jump to increment   
    mov c, p2.2 ;detect if decrement switch was pressed  
    jnc down ;jump to decrement  
  
    mov c, p0.0 ;Check if display parity switch was pressed  
    jnc display\_parity ;if parity switch pressed jump to function to display parity  
  
    mov c, p2.0  
    jnc ladd\_jmp  
    mov c, p0.1  
    jnc ladd2\_jmp  
    mov c, p2.3  
    jnc rand\_jmp  
    sjmp loop  
up:  
    setb p0.3 ;reset switch  
    inc r4 ;increment counter  
    acall delay ;delay in order to display correct value  
    cjne r4, #10h, lights ;jump to display lights if no roll over  
    sjmp over ;if check for no roll over is not true, jump to deal with this  
down:  
    setb p2.2  
    dec r4 ;decrement counter  
    acall delay ;delay to display correct value  
    cjne r4, #0ffh, lights ;jump to display lights if no roll over  
    sjmp under ;deal with roll over if check for no roll over was not true  
over:  
    mov r4, #00h ;set counter to 0 since its rolling-over  
    sjmp delay\_s ;jump to create sound  
under:  
    mov r4, #0fh ;set counter to 15 since its rolling-under  
    sjmp delay\_s ;jump to create sound  
  
rand\_jmp:  
    ljmp rand  
  
  
ladd\_jmp:  
    ljmp ladd  
ladd2\_jmp:  
    ljmp ladd2  
lights:  
    mov a, r4; move counter val to acc  
    cpl a; invert for lights                
    mov c, 0e0h  
    mov p1.6, c; set light 0  
    mov c, 0e1h          
    mov p0.6, c; set light 1  
    mov c, 0e2h          
    mov p0.5, c; set light 2  
    mov c, 0e3h      
    mov p2.4, c; set light 3  
    sjmp loop; return to loop  
;-------ALAN----------------------------------------------------  
;Display register 5 on the binary subject which contain teh added value  
lightsa:  
    mov a, r5; move r5 val to acc  
    cpl a; invert for lights  
    mov c, 0e0h  
    mov p1.6, c; set light 0  
    mov c, 0e1h  
    mov p0.6, c; set light 1  
    mov c, 0e2h  
    mov p0.5, c; set light 2  
    mov c, 0e3h  
    mov p2.4, c; set light 3  
    sjmp loop; return to loop  
;-------------------------------------------------------  
  
;-----CHAU--------------------------------------------------  
display\_parity:  
    mov a, r4 ;move current counter to ACC  
    mov r0, #0 ;a count for number of 1 bits  
    mov r7, #4 ;decreasing count for number of bits read to find parity  
              ;only dealing with 4 bit words  so the count only need to read 4 bits  
loop\_parity:  
    rrc a ;rotate right and save bit 0 to carry bit   
    jc up\_parity ;jump to increase 1 bits  
    djnz r7, loop\_parity ;loop in order to read the right amount bits  
parity\_zero:  
;determins even or odd  
    mov a, r0 ;put the number of 1 bits into acc  
    mov c, 0e0h ;make carry bit acc.0  
    jc odd\_parity ;if carry bit is 1, its an odd parity  
    ljmp even\_parity ;if carry bit is 0, go to even parity  
      
up\_parity:  
    inc r0 ;increment number of 1 bits  
    djnz r7, loop\_parity ;go back to reading more bits if it has not been don 4 times  
    ljmp parity\_zero ;if we've read 4 bits, go to determine odd or even  
  
odd\_parity:  
;turns on the green light LED3 as long as parity switch is pressed  
    clr p2.7  
    acall delay  
    setb p2.7  
    ljmp loop ;go back to detecting switches  
  
even\_parity:  
;turns on second red light LED6 as long as parity switch is pressed  
    clr p0.4  
    acall delay  
    setb p0.4  
    ljmp loop ;go back to detecting switches  
;-----------------------------------------------------------  
  
delay\_s:  
    mov r0, #255 ;allow for continuous square wave  
sound:  
    ; Do sound here  
    mov TMOD, #00010000b ;timer 1 16 bit mode  
    mov TL1, #-614  
    mov TH1, #-614 shr 8 ;preload  
    setb TR1 ;turn on timer 1  
here:  
    jnb TF1, here ;wait for overflow  
    cpl p1.7 ;toggle speaker  
    clr TR1 ;turn off timer 1  
    clr TF1 ;clear overflow flag  
    djnz r0, sound ;send continous square wave to make sound longer  
    ljmp lights ;jump to display the new counter value after rolling over  
;------ALAN--------------------------------------------------  
ladd:  
    mov a, r4; move counter value to a   
    acall delay  
    mov r5, a  
    ljmp loop;jump back to loop for user to choose second number to add  
ladd2:  
    mov a,r4; move coutner value to a  
    acall delay  
    mov r6, a; store the 2nd number in register 6  
    mov a,r5;move r5 to a  
    acall delay; delay to correctly display and store number  
    add a, r6;add the two numver   
    mov r5, a;move the result in r5  
    ljmp lightsa;move to this function to display the register on the  
           ;device  
;--------------------------------------------------------------  
;-----Cash--------------------------------------------------  
rand:  
    clr p2.7; turn on indicator light  
    inc r4; increment count  
    cjne r4, #10h, next; reset if we make it to 15  
    mov r4, #00h;   
next:  
    mov c, p1.4; check for stop  
    setb p2.7; turn off indicator light  
    jnc light\_jmp; jump if we stopped  
    sjmp rand; otherwise keep incrementing   
;-----------------------------------------------------------  
  
light\_jmp:  
    ljmp lights  
delay:  
    mov r0, 255  
d\_loop:  
    mov r1, 255  
d\_loop2:  
    nop  
    nop  
    nop  
    nop  
    nop  
    nop  
    nop  
    nop  
    nop  
    djnz r1, d\_loop2  
    djnz r0, d\_loop  
    ret  
end

WORK DISTRIBUTION:

LanChau LeTran: 33%

Calculated the value for the timer in order to create the square wave to produce a sound when the counter roll over. Calculated the amount of time of the delay used in order to display the lights correctly. Wrote up the project report. Worked on the code and gave input on how to go about the problem.

Alan Truong: 33%

Worked on the code. Helped with debugging issues when the program did not function properly. Created and managed github files for easy sharing between members.

Josh Cash: 33%

Most prominent coder. Debug issues when it occurred. Managed github files. Fixed merging

issues between different stages of the code. Ensure that the feature worked properly.

Professor Stanley: 1%

We had 1% left over for contribution, you helped us with the square wave for the sound function in lab 105 so you contributed also.

FUTURE WORK:

In order to improve our project, we should have started a bit earlier. Though we finished everything in time, it’s still a bit close to the deadline. We started this at the beginning of the week it was due, and should have started the week before. Though we were busy with other classes, time management to meet up and start working on this group project could have been better.