Homework #8

Illya Starikov

Due Date: December 1st, 2016

This homework will be **extra credit.**

Problem #1

```
#include <reg51.h>

typedef enum { false, true } bool;
unsigned char R0;

void main() {
    TMOD = 0x06; // Counter 2, Mode 2
    TH0 = -0x60; // Initial value given

    R0 = 0; // Just in case
    TR0 = true; // Start the count

    do {
        if (TF0) {
            R0++;
            TF0 = false; // continue the count
        }
    } while (R0 != 60);

    TR0 = false; // Stop the count
}
```

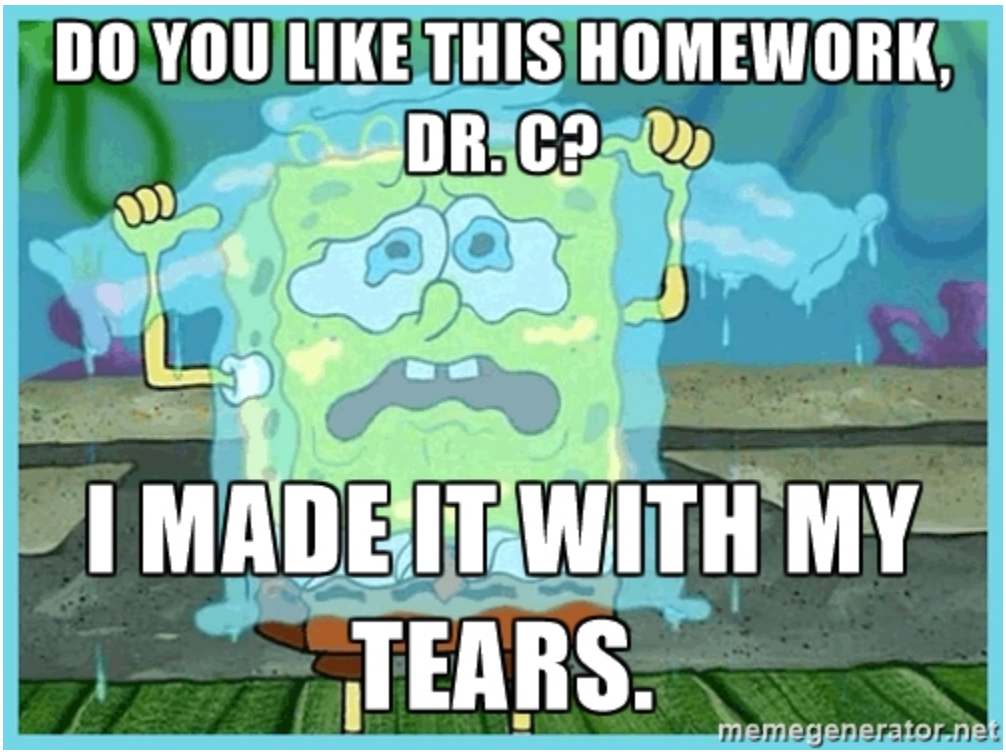
Problem #2

```
        ORG 0H

        LJMP MAIN

        /* Timer 1 Interrupt */
        ORG 001BH
        CPL P1.1
        RETI
```

```
9
10          /* Main */
11          ORG 0030H
12  MAIN:   MOV P2, #00H          ; P2 = output port
13          MOV TMOD, #20H        ; Timer 1, mode 2
14          MOV TH1, #-37D        ; 5 kHz => 25000 Hz => 4^-5 s
15                                ; => 40 us => 40/1.085 ticks => 37 ticks
16          MOV IE, #88H
17          SETB TR1
18
19  LOOP:   INC R0
20          MOV P2, R0            ; P2 = R0
21          ACALL DELAY
22          SJMP LOOP
23
24  ; so we need to generate a 10.85us delay
25  ; 10.85/1.085 = 10MC. Easier to just do a brute force
26  DELAY:  MOV R1, #4D           ; 2 MC
27  HERE:   DJNZ R1, HERE         ; 4 * 2 = 8 MC
28          RET                   ; 8 + 2 = 10 MC
29
30
31          END
```

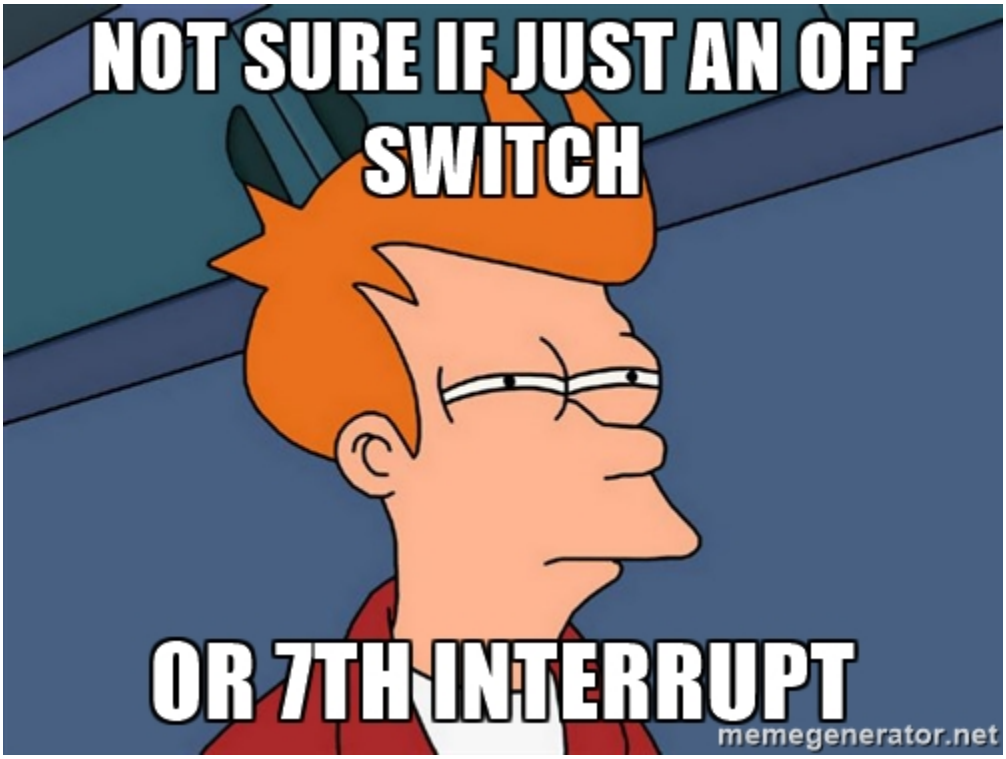


Problem #3

```
1          ORG 0
2          LJMP MAIN
3
4          /* Timer 0 Interrupt */
5          ORG 000B
6          CPL P1.1
7          RETI
8
9          /* Timer 1 Interrupt */
10         ORG 001B
11         CPL P1.2
12         RETI
13
14         /* Main */
15         ORG 30H
16 MAIN:   MOV TMOD, #12H          ; T0M2 & T1M1
17         MOV P1, #0              ; P1 = output port
18         ACALL WAVE0
19         ACALL WAVE1
20
21         MOV IE, #8AH
22 REPEAT: SJMP REPEAT
23
24 ; 5kHz => 1/5000 s = .0002 s => 200 microseconds
25 ; 200 us => 184 ticks PER PERIOD. Assuming a 50%
26 ; duty cycle. So actual value is 184/2 = 92 ticks
27 WAVE0:  MOV TH0, #-92
28         SETB TR0
29         RET
30
31 ; 500 Hz => .002 s => 2000 us
32 ; (FFFF - x + 1)(1.085) = 2000
33 ; x = FFFF - 1842 => E7 BD
34 WAVE1:  MOV TH1, #0E7H
35         MOV TL1, #0BDH
36         SETB TR1
37         RET
38
39
40         END
```

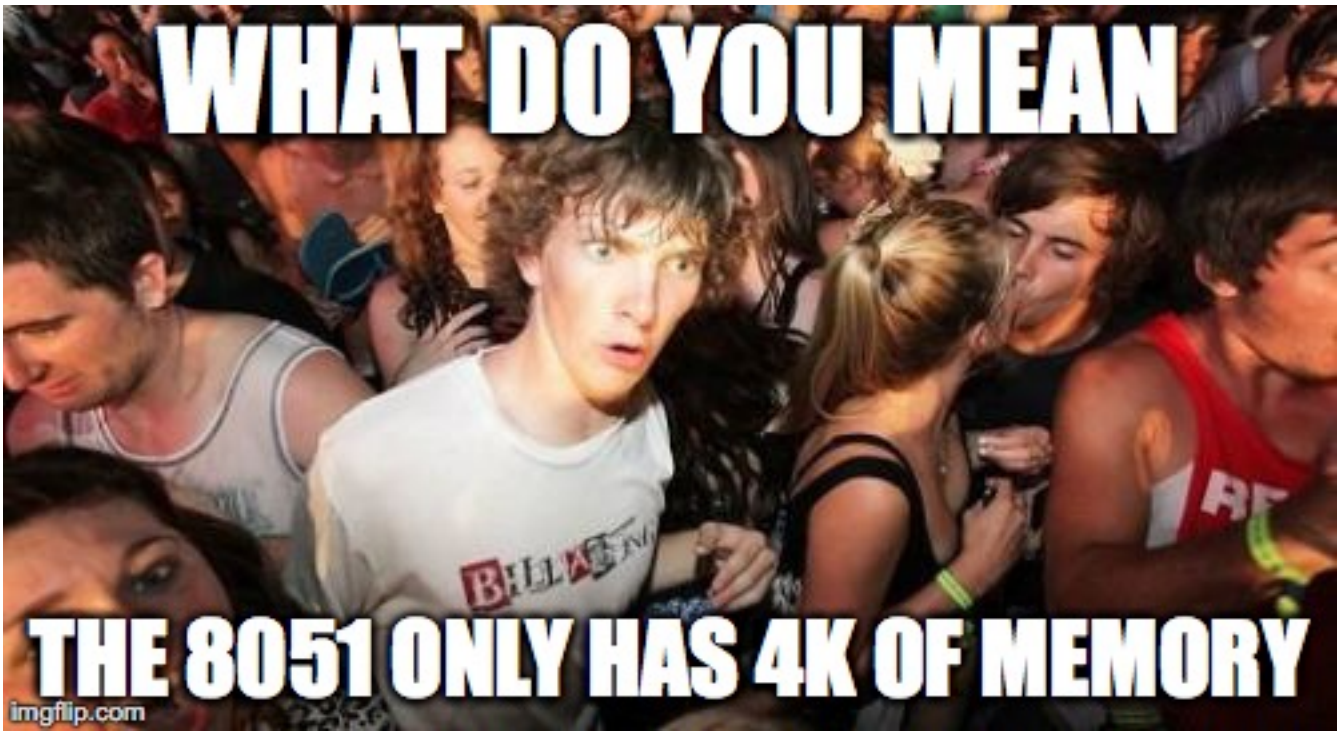
Problem #4

```
1          ORG 0H
2          LJMP MAIN
3
4          /* Interrupts */
5          ORG 001BH
```




```
6           LJMP SWAVE
7           RETI
8
9           /* Main */
10          ORG 30H
11  MAIN:   MOV R7, #1D
12          MOV P1, #0              ; Make P1 an output port
13          MOV IE, #88H            ; Enable interrupt on T1
14
15  LOOP:   MOV P2, R7              ; R7 holds the value of the div series
16          ACALL INFSER
17          SJMP LOOP
18
19  ; code for the square wave interrupt
20  ; determines via R6 what the last wave was
21  SWAVE:  MOV TMOD, #20H  ; Timer 2, mode 2
22
23          CJNE R6, #70, SKIP2LOW
24          ACALL HWAVE
25          SJMP NEXT
26
27  SKIP2LOW:
28          ACALL LWAVE
29
30  NEXT:   SETB TR1
31          RET
```

```
32
33  ; the wave has a duty cycle of 70%
34  ; so the high wave needs a high wave of 7kHz
35  ; & kHz => 7000 Hz => 1.42^-4 s => 142 us => 130 ticks
36  ; 130.8 *.7 = 92 ticks for 70% duty cycle of a 7kHz wave
37  HWAVE:
38          SETB P1.1
39          MOV TH1, #-92
40          MOV R6, #70             ; Just to keep track of what our last value
        was
41          RET
42
43  ; so, exact same reasoning as before, we need 130 ticks for
44  ; the period of the wave, but now we just subtract 92 from 130
45  ; so, 130 - 92 = 38
46  LWAVE:
47          CLR P1.1
48          MOV TH1, #-38
49          MOV R6, #30
50          RET
51
52  ; computes infinite divergent series incrementally
53  ; uses R7
54  INFSER:
55          MOV A, R7
56          ACALL TWOCOMP
57          ACALL ISNEGATIVE
58
59          JC RETURN
60          INC A
61
62  RETURN: MOV R7, A
63          RET
64
65
66  ; returns two's compliment of A
67  TWOCOMP:
68          CPL A
69          ADD A, #1D
70          RET
71
72
73  ; returns 1 in C if B is negative
74  ISNEGATIVE:
75          CJNE A, #10000000B, NEXT2
76  NEXT2:  CPL C
77          RET
78
79
80          END
```


WHAT DO YOU MEAN
THE 8051 ONLY HAS 4K OF MEMORY
imgflip.com