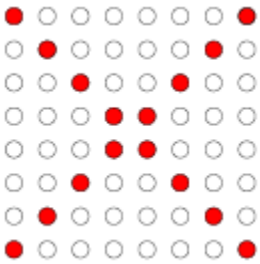
Note: This homework is the the same as the prelab for next lab.

1. Write a program to set up Port B as an eight bit output port (be sure to disable the seven-segment displays, and to enable the individual LEDS as you did in last week's lab), and to implement (i) a binary up counter, (ii) a shifting bit, (iii) a Johnson counter, and (iv) a Ford Thunderbird style turn signal based on the state of the DIP switches. Insert a 100 ms delay between updates of the display. Write the delay as a subroutine. Be sure to initialize the stack pointer in you program.

(Note: you will be referring to a number of MC9S12 registers in this and future programs. It is tedious and error-prone to look up and enter the addresses of the registers each time you write a new program. There is a file when you start a new project called 'hcs12.inc' that links to a file which has a list of all registers and their addresses for the 9S12DP256 version of the MC9S12 microcontroller. If you include that file in your program (by including the line INCLUDE 'derivative.inc' as the first line of your program), you can refer to all registers by name rather than having to look up their addresses.)

2. Write a subroutine to implement the binary up counter, have the LEDs count 0, 1, 2, 3, 4, ... It should take 256 counts from the time all LEDs are off until the next time all are off again.

3. Write a subroutine to display a shifting bit on PORTB which looks like the figure below. There is an easy way to calculate this. Start with two variables, one with a value of 0x80 and the other with a value of 0x01. OR the two variables together to get 0x81, the first pattern in the sequence. Then rotate the first variable right by one (to get 0x40), and rotate the second variable left by one (to get 0x02). OR these two together to get 0x42, the second pattern in the sequence. Continue rotating the first variable to the right, the second to the left, and ORing the two together.



4. Write a subroutine to generate the next pattern in the sequence for an eight-bit Johnson counter. The procedure to do this is as follows: Shift the present pattern to the right by one bit. The most significant bit of the next pattern is the inverse of the least significant bit of the present pattern. The number to convert is in accumulator A, and the next pattern in the sequence is returned in accumulator A. The subroutine should return with all registers, except A, the same as when the subroutine was called, so use the stack to save and restore any registers you need to use to implement the subroutine. The starting pattern is 00000000;

5. Write a subroutine to take the next entry out of a table, write it to PORTB, and update the index into the table. Here is an example of what the table might look like:

```
table_len:    equ (table_end-table)
              org data
table:        dc.b $00, $01, $02, $04, $08, $10, $20, $40, $80
table_end:
```

The index of the number to be displayed is passed in accumulator A. Your code should write the table entry corresponding to that index to PORTB. Return the index to the next table element in accumulator A. (For example, if accumulator A were 5, you would write the fifth element of the table, $10, to PORTB, and return a 6.) Make sure that the index stays between 0 and table_len - 1. The subroutine should return with all registers, except A, the same as when the subroutine was called, so use the stack to save and restore any registers you need to use to implement the subroutine. Fill out the table such that you get the following pattern. (This is called the TBird Taillights, because the old Ford Thunderbirds displayed this type of pattern for their turn signals. Current TBirds no longer display this type of pattern.)

6. Write the program that will display four different patterns on the LED display connected to Port B. You will use the state of bits 1 and 0 of the on board DIP switch to select which of the four patterns to display. Write a program to set up Port B as an eight bit output port (be sure to disable the seven-segment displays, and to enable the individual LEDs), and to implement (i) a binary up counter, (ii) pattern from part 2, (iii) a Johnson counter, and (iv) a Ford Thunderbird style turn signal based on the state of the DIP switches. Insert a 100 ms delay between updates of the display. Write the delay as a subroutine. Be sure to initialize the stack pointer in you program. Use four variables to hold information on the four patterns. Initialize these four variables to the first pattern in the sequence. You should have a loop which checks the DIP switches connected to Port H. If bit 7 of the DIP switches is high, end the loop and exit back to DBug-12 with a SWI instruction. If bit 7 of the DIP switches is low, check bits 0 and 1 to determine what pattern should be displayed.

PH1	PH0	Pattern
0	0	Binary Up Counter
0	1	Pattern from Part 2
1	0	Johnson Counter
1	1	TBird Turn Signal

For example, if bits 1 and 0 of Port H are 10, load accumulator A with the Johnson Counter variable, call the Johnson Counter subroutine, and save the returned accumulator A into the Johnson Counter variable. Call the Delay subroutine, then loop back to check the DIP switches again.