EE 308 – Homework 4

1. Suppose you started with the following register contents:

PC=200A Y=34B1 X=826F A=90 B=C2 SP=1C06

What address will be in the stack pointer, what values will be in the registers (A, B, X, Y, SP and PC), and exactly what is in the stack after the following instructions sequence is executed:

INST.	A	B	X	Y	SP	PC
	$90	$C2	$826F	$34B1	$1C06	$2009
PSHB PSHA PSHY PULX JSR $2270						

2. Below are some data in the MC9S12 memory:

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
1000	41	05	C3	12	15	B2	AA	79	C3	D6	8A	F1	91	43	0A	B9
1010	63	13	96	14	6B	24	80	C1	B1	1F	26	1A	31	83	DA	11
1020	10	1D	B2	61	91	09	A1	BB	67	11	78	11	4C	01	C8	95

Indicate the values in the registers after the MC9S12 executes the following instructions. Also write down the number of cycles needed to execute each instruction. Show what will be in the registers (in hex) after each of the instructions. If the instruction does not change a register, you may leave that entry blank. Note that the first instruction is located at address 0x2000.

Inst.	**A**	**B**	**X**	**Y**	**SP**	**N**	**Z**	**V**	**C**	**Addr. Mode**	**Eff. Address**	**Cycles**
	A0	0B	101C	1020	0200	1	0	1	0			
lds #$1020												
cpd $100C												
puly												
adda 5,-y												

3. Write a subroutine to display a counting pattern on PORTB, and return the next number (the number passed to the subroutine plus 1). The number to display is passed in accumulator A. Store this number into PORTB and return the next pattern in the sequence in accumulator A. The subroutine should return with all registers expect 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.

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 expect 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.

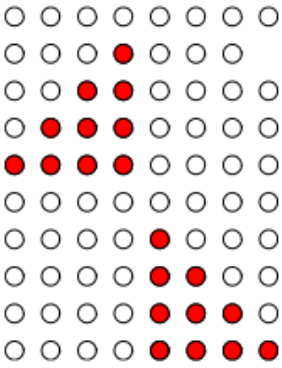
5. Write a subroutine to take the next entry out of a table, write it to PORTB, and update the index into the table. Here is an example of what the table might look like:

```
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 expect 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 pattern to display is shown below:



6. Write the program for Part 3 of Lab 2. The program will display three different patterns on the LED display connected to Port B. You will use the state of bits 1 and 0 of the onboard 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) a Johnson Counter, (iii) a Ford Thunderbird style turn signal based on the state of the DIP switches. (These are the three subroutines from Problems 3 to 5.) 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 variables to hold information on the three patterns. (You will need one variable for the binary and Johnson counter patterns, and one variable to hold the sequence number for the TBird Taillight pattern.) Initialize these three 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 to display:

PH1	PH0	Pattern
0	**0**	Binary Up Counter
0	**1**	Johnson Counter
1	**0**	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.