EE 308 – Homework 4

Due Feb. 15, 2010

1. Find the values of the N, Z, C, and V bits of the CCR register after execution of each of the following instructions, given that (A) = $C5 and the condition flags are N=0, C=1, Z=0, and V=0. (Assume these are the values before each instruction starts e.g., do not use the flag state resulting from the instruction in part (a) as the initial state for part (b).)

   (a) `ADDA #$7A`
   (b) `ADCA #$3A`
   (c) `LSRA`
   (d) `ASRA`
   (e) `CMPA #$D0`
   (f) `SUBA #$AC`

2. Suppose you started with the following register contents:
   `PC=2007 Y=7892 X=FF00 A=44 B=70 SP=1F7F`
   What address will be in the stack pointer, what values will be in the registers, and exactly what is in the stack after the following instructions sequence is executed:

```
PSHA
PSHY
PSHB
PULX
JSR   $2007
```

3. 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	D6	05	35	CF	E0	00	FE	08	20	A6	00	47	6A	05	08	53
1010	26	F7	34	C6	C8	CD	9C	40	03	26	FD	53	26	F7	3D	3F
1020	07	C2	3A	68	F3	09	C2	67	9A	0F	AA	55	08	40	CD	CF

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`.

Instruction	D									Addressing Mode	Effective Address
	A	B	X	Y	SP	N	Z	V	C		
	0A	BB	1010	1020	0A00	1	0	1	0		
`lds #$1018`											
`cpd $100D`											
`pulx`											
`asrb`											
`stab $1013`											
`adda 2,-y`											

4. Suppose that we have the following instruction sequence to be executed by the MC9S12, what will be the contents of the topmost four bytes of the stack after the execution of these instructions?

```
lds     #$3C00
ldd     #$AA55
staa    1,-SP
stab    1,-SP
ldx     #$3377
stx     2,-SP
```

5. Draw the stack frame (the memory where the stack is located) and enter the value of each stack slot (if it is known) at the end of the following instruction sequence. Also, indicate the value of the stack pointer after execution of each instruction.

```
          org     $2000
          lds     #$2000
          leas    -2,sp     ; reserve 2 bytes for local variables
          clrb
          ldaa    #53
          psha
          ldaa    #$A2
          psha
          ldx     #$45AC
          pshx
          jsr     sub_123

...

sub_123:  pshd
          leas      -4,sp  ; reserve 4 bytes for local variables
```

6. 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.

7. Write a subroutine to display a shifting bit on `PORTB`. When you enter the subroutine, the three least significant bits of accumulator A represent the bit to display, and bit 3 of accumulator A represents the direction to shift the bit for the next pattern (1 for left, 0 for right). The upper four bits of accumulator A are ignored. The bit shifts until it reaches the end, then changes direction. For example, if A contained 00001100, this would indicate that you should turn on bit 4 of `PORTB`, and update accumulator A to turn on bit 5 next time. You should return a 00001101 in accumulator A. When you reach the left end (bit 7) change the direction bit from 1 to 0. When you reach the right end (bit 0) change the direction bit from 0 to 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.

8. 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.

9. 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.

10. Write the program for Part 3 of Lab 2. The program 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 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 shifting bit, (iii) a Johnson counter, and (iv) a Ford Thunderbird style turn signal based on the state of the DIP switches. (These are the four subroutines from Problems 6 to 9.) 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 to display:

PH1	PH0	Pattern
0	0	Binary Up Counter
0	1	Shifting Bit
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.