CS3280 Homework 3; Due: Monday, February 27, at beginning of class

1. For each of the following questions, assume that the registers are initialized to the following values (i.e., each part is independent of the others). Note: NZVC are the condition codes. Give the contents of the registers **(in HEX, except for the condition codes)** after each instruction is executed. All numbers shown are HEX numbers except when noted otherwise. DO NOT leave any box empty (even if the contents don't change) - empty boxes will be counted wrong! Also, do not use a DASH "-" in a box and do not forget the $-sign for hex numbers.

A. SUBA 1,X

	PC	A	X	Y	ML($A000)	ML($A001)	NZVC
Initial	$C000	$80	$9FFF	$A000	$FF	$1F	0111
After	$C002	$81	$9FFF	$A000	$FF	$1F	1001

B. ADCA 0,Y

	PC	A	X	Y	ML($A000)	ML($A001)	NZVC
Initial	$C000	$80	$9FFF	$A000	$FF	$1F	0111
After	$C003	$80	$9FFF	$A000	$FF	$1F	1001

C. STY 0,X

	PC	A	X	Y	ML($A000)	ML($A001)	NZVC
Initial	$C000	$80	$9FFF	$A000	$FF	$1F	0111
After	$C003	$80	$9FFF	$A000	$00	$1F	1001

D. BHS 3 (3 is the decimal branch offset)

	PC	A	X	Y	ML($A000)	ML($A001)	NZVC
Initial	$C000	$80	$9FFF	$A000	$FF	$1F	0111
After	$C002	$80	$9FFF	$A000	$FF	$1F	0111

E. BLE -3 (-3 is the decimal branch offset)

	PC	A	X	Y	ML($A000)	ML($A001)	NZVC
Initial	$C000	$80	$9FFF	$A000	$FF	$1F	0111
After	$BFFF	$80	$9FFF	$A000	$FF	$1F	0111

2. Suppose we have the following instructions:

```
LDAB        #$10
CMPB        #$FF
BLT         AHEAD
```

Will the program take the branch? Give a convincing reason. For example, just stating 'the branch will be taken because the number is greater' is insufficient. Rather, come up with the decimal value of the numbers to show that a number is greater than the other. Or, alternatively, come up with the flag settings (**show your work**) and show whether the branch condition is satisfied or not. A correct branch prediction with incorrect, missing, or insufficient reason will result in 0 points.

BLT -> signed numbers -> $10 = 16, $FF = -1; 16>-1 -> branch NOT taken

or:

$10 - $FF = $10 + $01

```
1<-0<-   0001 0000
        +0000 0001
        -------------
          0001 0001    -> unsigned overflow, no signed overflow; result non-zero and positive ->
```

Z=N=V=0, C=1 -> N XOR V = 0 -> branch condition not met -> branch NOT taken

3. Suppose we have the following instructions:

```
LDAB        #$10
CMPB        #$FF
BLS         AHEAD
```

Will the program take the branch? Give a convincing reason. For example, just stating 'the branch will be taken because the number is greater' is insufficient. Rather, come up with the decimal value of the numbers to show that a number is greater than the other. Or, alternatively, come up with the flag settings (**show your work**) and show whether the branch condition is satisfied or not. A correct branch prediction with incorrect, missing, or insufficient reason will result in 0 points.

BLS -> unsigned numbers -> $10 = 16, $FF=255; 16 < 255 -> branch taken

or:

Z=N=V=0, C=1 (see above) -> Z OR C = 1 -> branch condition met -> branch taken

4. Given the following pseudo code:

```
int count;
int result;

for (count=1, result=10; count < result; count++) {
        result--;
}
```

a) write an assembly language program that will implement this pseudo code using a while construct.

```
          ORG       $B000
count     RMB       1
result    RMB       1

          ORG       $C000
          LDAA      #1
          STAA      count
          LDAA      #10
          STAA      result
WHILE     LDAA      count
          CMPA      result
          BGE       ENDWHILE
          INC       count
          DEC       result
          BRA       WHILE
ENDWHILE  BRA       ENDWHILE
```

b) write an assembly language program that will implement this pseudo code using a do-until construct.

```
          ORG       $B000
count     RMB       1
result    RMB       1

          ORG       $C000
          LDAA      #1
          STAA      count
          LDAA      #10
          STAA      result
DO        INC       count
          DEC       result
UNTIL     LDAA      count
          CMPA      result
          BLT       DO
ENDDO     BRA       ENDDO
```

Note: For both a) and b), do not forget to include the data section in your program. ***count* and *result* are 1-byte variables to be implemented in memory, not in registers.** Your assembly program must match the pseudo code 1-to-1 (this also means that you should use a conditional branch that correctly implements the loop condition; **BNE or BEQ branches are not allowed**). Use labels such as IF, ENDIF, WHILE, ENDWHILE, DO, ENDDO, to show where your If/While/Do-Until constructs are. You don’t have to electronically submit anything (this is not a lab).

5. Consider the following assembly language program:

```
                ORG         $B000
ARRAY           FCB         1, 5, 10, 20, 34, 50
N               EQU         3
RESULT          RMB         N

* start of your program
                ORG         $C000
                LDX         #ARRAY
                LDY         #RESULT
                CLRB
LABEL           LDAA        0,X
                SUBA        1,X
                STAA        0,Y
                INX
                INX
                INY
                INCB
                CMPB        #N
                BLO         LABEL
DONE            BRA         DONE
                END
```

A. Come up with the pseudo code for this program. The pseudo code should match the program 1-to-1. **Note: use pointers in your pseudocode.**

```
int ARRAY[], RESULT;
#define N 3

int *POINTERX, *POINTERY, COUNTER;

                POINTERX=&ARRAY[0];
                POINTERY=&RESULT[0];
                COUNTER = 0
DO              {
                        *POINTERY =  *POINTERX  -  *(POINTERX+1);
                        POINTERX++;
                        POINTERX++;
                        POINTERY++;
                        COUNTER++;
                } until (COUNTER>=N)
```

B. Describe what this program is doing:

This program calculates the difference of pairs of 2 adjacent ARRAY elements and stores the difference in a RESULT array